3 9 2

V T T P U B L I C A T I O N S

Veikko Seppänen, Kimmo Alajoutsijärvi &
Päivi Eriksson

Projects or products: seeking for the business logic of contract R&D

TECHNICAL RESEARCH CENTRE OF FINLAND ESPOO 1999

VTT PUBLICATIONS 392

Projects or products: seeking for the business logic of contract R&D

Veikko Seppänen
VTT Electronics

Kimmo Alajoutsijärvi
University of Oulu

Päivi Eriksson
University of Tampere

TECHNICAL RESEARCH CENTRE OF FINLAND
ESPOO 1999

ISBN 951–38–5391–8 (URL: http://www.inf.vtt.fi/pdf)
ISSN 1455–0849 (URL: http://www.inf.vtt.fi/pdf)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Sulautetut ohjelmistot, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Inbyggd programvara, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Technical editing Leena Ukskoski

Seppänen, Veikko, Alajoutsijärvi, Kimmo & Eriksson, Päivi. Projects or products: seeking for the business logic of contract R&D. Espoo 1999. Technical Research Centre of Finland, VTT Publications 392. 110 p. + app. 104 p.

Keywords core competence, industrial relationships, logic of action, embedded software, research and development

ABSTRACT

This research addresses the question of building and exploiting competence in connection with contract research and development (R&D), by means of a longitudinal case study. The research involves VTT as a supplier and internal research customer, Tekes as a funding body and several firms as external industrial customers. We are looking into their mutual relationships to gain a better understanding about the evolution and exploitation of competence on the so-called code generation techniques used in the development of software embedded in electronic products.

The analysis of the code generation case is based both on written material and on interviews of persons involved in code generation related activities at VTT, Tekes and industry from the mid-eighties to the present date. The building of the code generation competence of VTT is analysed and explained within the R&D process based on project relationships. The marketing and purchasing of the competence is also addressed. Differing logic of action of the interacting parties have been found to affect the evolution of competence, within networks established for creating and making use of the competence.

In the code generation case, the managers of VTT aimed at creating a growing portfolio of fully contractual project relationships, involving machine automation firms, in particular. The researchers favoured marketing the competence as a commercial style tool, with a minimum of tailoring done in projects. The customers of VTT had difficulties in coping with these two logic of action, in a rapidly and radically changing business environment. It may have been for this reason that the competence was, after all, utilised mainly by VTT itself in joint research projects.

This did neither benefit its developers nor did it advance the evolution of the competence. The differing logic of action of the two key parties, which resulted in the lack of any considerable portfolio of external customer relationships, lead to a rather rapid withering of the competence at VTT. However, the developed code generation technology has recently been sold to the former researchers, who have established a company based on their own business logic. This kind of competence survival through many years and despite conflicting viewpoints is, after all, one of the key factors in making business out of research.

PREFACE

The main purpose of this research was to study the evolution and exploitation of competence in the context of inter-organisational relationships. We used a longitudinal case study to analyse the change of competence in project-based relationships, as well as its utilisation for the needs of both VTT itself and its industrial customers.

We are grateful to Dr. Kari Leppälä and Mr. Harri Perunka for their comments on an early draft of this report. The persons who were actually involved in code generation related activities at VTT are also gratefully acknowledged. The people outside VTT, who informed us about their views to, participation in and use of the results of these activities deserve even more grateful thanks.

Mrs. Jaana Määttä, who carried out some of the interviews, did a very good job as a marketing trainee. Professor Kristian Möller, one of the key figures in the research of industrial networks, has reviewed this report for publication. We are indebted to him for his time to read the manuscript and the comments that helped us to improve it.

Veikko Seppänen, the first author, has translated pieces of the code generation case data included in appendices 2 and 3 for the needs of this report. Although the translations have been reviewed by the key informants, he is solely responsible for any possible mistakes remaining in the translations.

The Foundation for Economic Education provided a personal grant for Veikko Seppänen to carry out the research published in this report, which is most gratefully acknowledged.

Oulu, August 13, 1999

Veikko Seppänen, VTT Electronics

Kimmo Alajoutsijärvi, University of Oulu

Päivi Eriksson, University of Tampere

CONTENTS

SYMBOLS AND ABBREVIATIONS

AI	Artificial intelligence
ARA	Activity-resource-actor model
ASIC	Application-specific integrated circuit
AUT	VTT Automation
CASE	Computer-aided software engineering
CMM	Capability maturity model
COTS	Commercial off-the-shelf (software)
CUTE	C software unit testing tool
DSP	Digital signal processing
EDA	Electronic design automation
ELE	VTT Electronics (1994 -)
ELI	VTT Food technology laboratory (- 1993)
FIM	Finnish mark
IT	Information technology
IPR	Intellectual property rights
KE	Knowledge engineering
KTM	Ministry for Trade and Industry
OO	Object-oriented
PLC	Programmable logic controller
PC	Personal computer
QFD	Quality Function Deployment
R&D	Research and development
SA/SD	Structured analysis and design
SE	Software engineering
SDL	Structured design language
STUK	Finnish Centre for Nuclear Safety
TKO	VTT Computer technology laboratory (1983 - 93)
TTCN	Tree and tabular combined notation
VHDL	Very High-Level Design Language
VLSI	Very large scale integration of electronic circuits
VTT	Technical Research Centre of Finland

1 INTRODUCTION

This research addresses the evolution of competence of a *contract research* institute within the context of its project relationships. From the viewpoint of institute management, competence should contribute to the strategic business goal of ensuring an adequate volume of fully contractual activities. The exploitation of competence should also advance the business goals of the customers of the institute, either directly for the benefit of the customer's own customers or indirectly via the customer's product or process development activities. We are studying how the objectives, goals and viewpoints of interacting parties are managed throughout the initiation and carrying out of joint activities as part of their relationships. Since the goals may not be aligned with each other, a successful evolution of competence can not be guaranteed. The parties may disagree over what should be considered as an element of competence, in the first place.

VTT Electronics (ELE), our case organisation, carries out contractual R&D activities in the field of electronics. We refer to [Seppänen et al. 1998a] and to the Internet home page www.ele.vtt.fi for a brief description of the institute. Code generation addressed in the research involves *embedded computer systems*, built-in controlling computers incorporated in electronic products. Embedded systems *software engineering* was one of the focal areas important topic of TKO (Computer technology laboratory of VTT), one of the predecessors of ELE that was merged with three other laboratories to form ELE in 1994. When established in 1983, the embedded software engineering section of TKO included some fifteen researchers. At present, there are about a hundred software engineering experts at ELE.

Industrial software development emerged in the sixties. The phrase software engineering was taken in use in the seventies to denote a systematic approach to organising, managing and performing software design as part of product development. The need for embedded systems software development originated in the seventies, when microprocessors became commercially available. At first, embedded software was mostly developed by hardware designers. In the early eighties, TKO was one of the main contributors In Finland to bringing embedded software engineering principles, methods and tools to the attention of industry. It even proposed and made popular the Finnish translation of the term embedded software, "sulautettu ohjelmisto". TKO researchers took part in the development of a number of very successful electronic products as embedded software engineering experts. Moreover, the roots of some of the very first Finnish embedded systems software development tools are to be found at TKO.

In more general terms, embedded systems software engineering research started to flourish in Finland in the mid-eighties. One of the main reasons was that Tekes, the national technology development funding body, organised two quite extensive software-related research programs. Companies that had started to utilise microprocessors realised that the size of software incorporated in their products was rapidly growing and that they needed better means for managing the software development.

They also found out that the features used for arousing the customers' interest in products were often based on software. Industry was, for these reasons, quite eager to join the steering groups of public research projects dealing with embedded software engineering. Only modest fees, compared to the total budgets of the projects, needed to be paid for joining the groups. The research was in most cases performed by university or by VTT researchers. At the same time, industry started to hire experts so as to facilitate the use and development of new software engineering methods and tools. Quite many of them had previously been working at universities and at VTT, which would contribute to industrial interest in public research projects. While product development engineers were playing a major role, other people from industry, such as marketing and customer support professionals, were seldom involved. Most of the representatives in funding bodies, such as Tekes, were technology experts as well.

Tekes established the first national software-related research program called Finprit in the mid-eighties. TKO carried out several Finprit projects and produced prototypes of embedded software development tools. The second software engineering research program, called Finsoft, was launched by Tekes in the late eighties [Karjalainen 1991]. TKO prepared and carried out the biggest project within the Finsoft program, called Sokrates. The aim of this project was to produce a design method and to build a *code generator* prototype, which would provide for a systematic, seamless and automated embedded software development approach. The general goal of the project was characterised as creating *design automation* for embedded systems. The intention was to turn embedded software development from art to an engineering discipline relying on rigorous methods and automated tools.

Despite the efforts allocated to the project, neither a breakthrough in code generation succeeded, nor did any vendor commercialise the code generator prototype developed in the project. The rights to the code generator have recently been sold by VTT to a small company owned by some of the original researchers. Therefore, there's still a possibility for the venture to succeed. Yet, in this research we are mainly interested in investigating why the code generation competence could not be more extensively exploited earlier, to allow VTT to make use of its research results.

Conflicting viewpoints arose around code generation at VTT as early as the early nineties. One of the very basic controversies involved line managers and focal researchers. The conflict was related to the basic business logic of VTT based on a portfolio of three kinds of projects, shown in Figure 1. About twenty percent of projects involve innovative self-funded *“green”* research, usually followed by a few companies. The degree of industrial income in green projects, if any, may be only half to one percent of the total budget of the project. Technical risks are often high in such projects, but business risks are low both for VTT and the possible industrial followers of the research. These projects aim at "capability development" from the viewpoint of inter-organisational networks [Håkansson and Snehota 1995].

Some forty percent of the portfolio includes *“blue”* applied research projects carried out and financed jointly by public funding bodies, VTT and industry. In these projects, which involve "strategy development" from the network perspective [Håkansson and Snehota 1995], the typical industrial contribution amounts to twenty percent, the input from public sources fifty percent and VTT’s own funding thirty percent. This financing scheme covers all the expenses of the projects, but does not produce any profit for VTT. The rest of the portfolio, about forty percent, consists of *"red”* development projects purchased by industrial customers. The income from red projects should not only cover all the expenses of VTT, but also result in some, yet small, profit for VTT, in addition to the benefits for the customers. Contrary to the networking framework of Håkansson and Snehota, multi-party networks for strategy development thus usually precede dyadic customer relationships.

From the managerial point of view, internal investments made in capability development in green projects should be paid back by subsequent blue strategy development projects and red projects marketed successfully to industry. From researchers’ viewpoint, the need to create contractual business from self-funded research, based on an expanding flow of external income, may not be obvious at all.

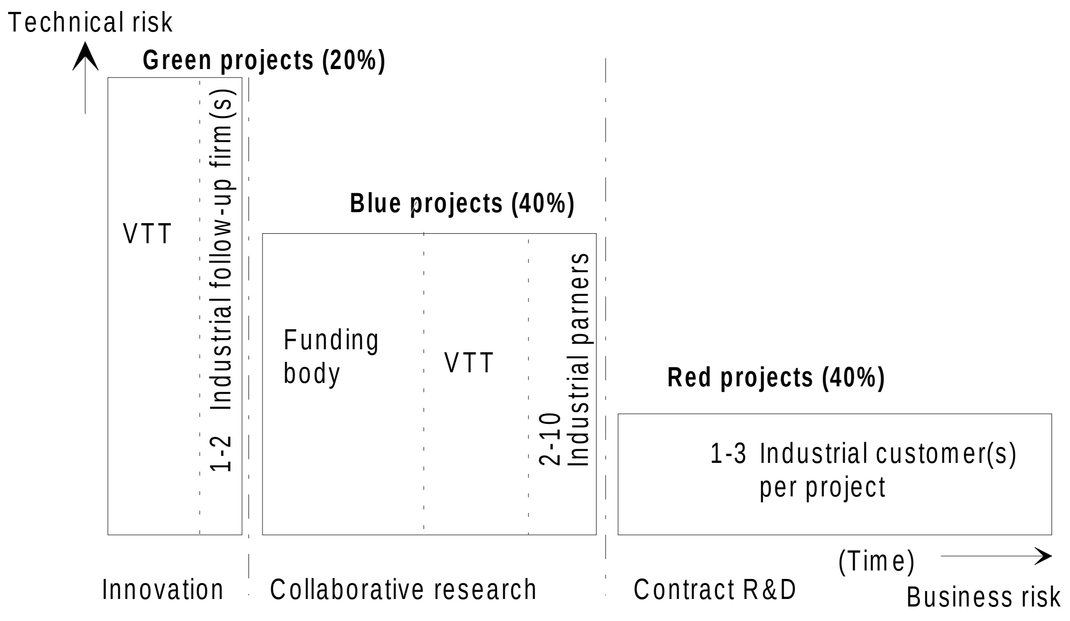

Figure 1. The intended project portfolio of VTT.

Code generation research started "backwards”, as a small red project paid by a company, whose R&D manager was the former section head at TKO. The subsequent Sokrates project was a very large blue project, in which VTT's own financing of was quite limited, due to the as high as 70% funding by Tekes and the fees paid by the over ten companies taking part in the project. Although the expectations for a number of subsequent red projects were high, after Sokrates both public and industrial investment on code generation related activities decreased. Two mid-size red projects were carried out for one of the industrial members of the Sokrates steering group, and two small projects for other companies.

Although some licenses of the code generator were sold to firms that had not been involved in the Sokrates project, the generator was mostly used in blue projects at VTT. On one hand, the code generation researchers felt that managers were preventing them from making a practical tool out of the research results. They were not allowed to continue their work by any considerable internal funding, while the managers were hoping to establish red projects paid by industrial customers. On the other hand, the managers' tolerance to wait for expanding industrial income from red projects gradually disappeared:

> *Code generation researcher: "The management of TKO said that this [the code generator] is not good, this is a prototype, a hack-hack, and we cannot seriously offer this to industry. Now, having been in industry for four years, I have seen what kind of tools are being used, for example in the telecommunication area: tools developed by ourselves and by universities, public domain tools, and our competitors' tools. I must say that Reagenix was, after all, a very reasonable tool. The impression that was given by the management of TKO was that industry uses only top-quality, well-packaged tools offered by reliable parties. This is the point, VTT did not see itself as a reliable tool developer."*

> *Manager: "Well, it might be that there was a belief that something would come out, but that belief gradually vanished. In closing, [of the Sokrates project] ... [the focal researchers] took it for granted that about six companies would take the results in use right away. They already wrote down ready-made contracts, but the fact was that there were no real deals at all. They told ... that everything would have been agreed, but the reality was completely different. We lost confidence in their judgement."*

Contrary to the conclusions of many studies of industrial relationships, the basic business strategy of a contract research organisation like VTT is not always the length of individual customer relationships, but the width of the portfolio of such relationships. The reason is that the width is, in practice, a better means of maintaining and advancing competencies.

This has been shown, for example, in [Ford et al. 1998], relying on a study carried out among 123 Swedish companies on technological relationships. Relationships with horizontal units, e.g. such external partners as research institutes, were the shortest, compared e.g. with the relationships with customers and vertical suppliers. Their duration was less than four years in 55% of the surveyed cases. The weighted average of technological relationships with horizontal units was eight years. In the code generation case, the longest relationships lasted for six years, and typically much shorter. Well over twenty relationships existed during a period of thirteen years despite the case being rather unsuccessful from the perspective of creating customer relationships.

Ford and others concluded that the most common type of firms with respect to technological relationships was “broad co-operators”, 48 of the 123 companies had a versatile portfolio of relationships with customers, suppliers and horizontal units. Only thirteen of the 28_“focused companies” interested in certain types of partners were dealing with horizontal units. VTT had also quite many broad collaborators interested in code generation during the Sokrates project. Only few of its customers chose to focus on code generation later in the form of red projects or to use VTT as a technology supplier of the code generator tool.

Code generation has not yet become a notable approach to embedded systems software development in more general terms either. However, one third of the industrial respondents of a global survey concerning the strategic needs and future trends of embedded software technologies carried out in 1996 [Seppänen et al. 1996] indicated that code generation techniques will be needed in the future. Only 5% of the respondents fully disagreed with the proposition of using such techniques.

We are addressing the code generation case by employing the framework presented in [Seppänen et al. 1998a], where it was used for explaining the evolution of fault diagnosis competence *within* the customer relationships of VTT. In the code generation case, VTT managers wanted the competence to be built_*for* a portfolio of such relationships, whereas the focal researchers focused on license selling in much shorter term business transactions.

This report is organised as follows. First, the research design of the case study that was performed is described in Chapter 2. Then, an overview of the framework used for explaining the development of the code generation competence is given in Chapter 3. Intertwined stories of the focal researchers and one of the VTT managers is presented in Appendix 3, to provide an overview of the developments that took place from the viewpoint of the two most influential groups of actors at VTT.

The case data is analysed and discussed by using the competence evolution framework in Chapter 5. Finally, Chapter 6 presents the conclusions drawn from the code generation case and a cross-case analysis with the fault diagnosis case reported in [Seppänen et al. 1998a]. Appendix 1 provides a comprehensive list of the written material and interviews used as data sources. Appendix 2 comprises an extensive summary of the case data, to supplement the code generation stories presented in Appendix 3.

2 RESEARCH DESIGN

This research on the evolution of code generation was carried out as a longitudinal case study. According to [Yin_1988], a case study strategy is relevant, when the research question is of the explanatory form “how” or “why”, and the research does not require control over the behavioural events that are being studied and the focus is on contemporary events. Yin defines case study as “an empirical inquiry that: investigates contemporary phenomenon within its real-life context; when the boundaries between the phenomenon and context are not clearly evident; and in which multiple sources of evidence are used.”

We are interested in explaining how competence emerges - or withers – and is being exploited within the context of the relationships of a contract research organisation. The boundary between this phenomenon and its context is all but clear cut. As an example, from a managerial viewpoint we would also like to control the context, i.e. to know how and why some individual competencies can or cannot be utilised in a business strategy based on the creation and maintenance of project relationships, or how other kinds of relationships - be they social, economic or spatial - could be created and utilised to advance project relationships.

Our focus is on events that have taken place during the past several years, but which are not past history. In other words, we are addressing contemporary events. In this research we do not need to have any control over the events, but we are not complete outsiders either. For example, the first author of this report, who is a line manager at VTT, sold the rights of the code generator technology, the object of the activities that we have studied, to a small company during the research project. The owners of the company are the former code generation researchers. Since the conditions for carrying out a case study given by Yin fit with our research, and participant observation involving one of us is not excluded from the method but seen to provide “unusual opportunities for collecting case study data” [Yin 1988], we have selected the case study approach.

A longitudinal perspective is necessary for understanding and explaining changes in real-life phenomena. The period of over ten years that we are focusing on starts from the first code generation related project carried out by VTT in the mid-eighties and ends at the present date. The contemporary events related to the code generation are thus well covered, from the viewpoint of VTT and the other parties involved in the case.

Although our study is substantive, aiming at understanding and explaining complex events in their real-life context, it is not strictly qualitative, because we have used the competence evolution framework presented in [Seppänen et al. 1998a] as a theoretical basis for explaining our case data. The case data has not been the sole source of information used for creating the framework - the modifications of the framework that we suggest in this report should be viewed as an attempt to improve the existing theory.

Yin points out, however, that the role of theory building *prior* to any case data collection has often been overlooked. Taking heed of this point we extended the framework before using it in structuring and analysing the code generation case data.

A case study research design includes at least the following elements: research questions, propositions (if any), units of analysis, the logic of associating case data to the propositions, and the criteria used for interpreting the results. Since our study is of explanatory nature, we also need to use some patterns to structure the explanations. The *research question* that we address comprises two parts: how competence evolves within the project relationships of a contract research institute, and why the competence can or cannot be exploited for the contractual business needs of the institute.

A research *proposition* is a means of focusing the study on a relevant topic. The main proposition of our research is that the skills and capabilities of a contract research organisation should evolve towards core competence [Hamel and Prahalad 1994], so as to make the organisation efficient enough with regard to its capabilities and flexible enough in its relationships. An important associated proposition is that core competence does not emerge only *for* customer needs within the supplier's internal activities, but *within* contractual activities. Concerning this proposition, we are analysing how the non-alignment of different logic of action of the involved parties will slow down or even prevent the evolution and exploitation of competence.

A *unit of analysis* specifies the elements of the case to be studied, usually relying on the formulation of research questions. The units of analysis of this research are the resources and activities involved in the R&D and business processes of a technology supplier and its customers. The main *patterns of explanation* are the actors and relationships associated with the processes, called focal nets in [Seppänen et al. 1998a] and *process nets* in this report. In terms of the actors of these nets, we focus on small groups of people, such as the code generation researchers, the managers of the focal organisation and the key representatives of customers. Their logic of action and interaction are made explicit and analysed. The *project cycle* is used for structuring the R&D process of both the case organisation and its customers. This cycle involves the planning, conducting and controlling of individual projects. The *competence marketing cycle* is used for structuring the marketing process of the focal organisation and the purchasing process of its customers. It includes not only initiating and contracting of projects, but also general planning and management activities.

The quality of the research design of a case study can be evaluated by using four tests related to the basic elements of the research design: *construct validity*, for which multiple sources of evidence, chains of evidence and informant review of the case study report can be used; *internal validity* by pattern matching, explanation building or time-series analysis; *external validity* by replication in multiple case studies; and *reliability* based on a documented case study protocol and case data base. One of the strengths of case studies is that it allows employing a full range of evidence from physical artefacts and documents to interviews and direct observation.

We have used all these sources, to pass the construct validity test. Chains of evidence will be presented in connection with the analysis of the case data, including both chronological chains and relations based on such elements of the competence evolution framework as actor bonds, resource ties and activity links in relationships between VTT and its external collaborators and customers. Moreover, a draft case data summary (Appendix 3) has been reviewed by the key informants interviewed for the study. [Yin 1988] claims that "for case studies, the most important use of documents is to corroborate and augment evidence from other sources". We do not fully agree with this, because it seems, according to [Seppänen et al. 1998a] and this research, that it is the other way round, at least in the case of a contract research organisation operating on a project basis.

Within this context, managerial and technical documents are the basic means for professional people to communicate and agree over technical and managerial issues. Most research results are documented either as internal papers or as public reports, and many of them are also evaluated in written form. In addition, problems in contractual projects are often noted in managerial documents. Individuals often keep personal diaries where they write down, for example, other people's opinions on joint plans, activities and results. To try to structure this kind of data around interviews, which are "most commonly, ... of an open-ended nature" in case studies [Yin 1988], would be very difficult. Therefore, we first produced an initial case data base from a chronological set of organisational and project documents and corroborated this documentary data with the results of interviews.

We interviewed over forty people altogether, including six out of the total of seven code generation researchers; five line managers of VTT directly involved in code generation related activities; seven VTT researchers associated indirectly with the code generation projects; ten VTT and industrial people as end users of the code generator, a subcontractor involved in the further development of the code generator; the two persons from Tekes involved in code generation related projects; three industrial customers who contracted code generation projects from VTT, including one line manager and two application engineers; and altogether eleven members of the steering group of Sokrates, the main code generation research project.

The interviews were conducted as face-to-face discussions, by phone, as a group rehearsal of the code generation researchers, and by using electronic mail. The group rehearsal was carried out in an unstructured manner. Direct observation was carried out in the rehearsal, but not as part of any ongoing project. Most of the other parties were interviewed by using a semi-structured approach, asking them to answer questions organised around the specific projects in which they had been involved. Some interviews were carried out by a marketing student as a training assignment. Frame 1 illustrates, as an example, the questions asked from the members of the Sokrates steering group. The inquiries concerned the resources expected and resulting from the project, as well as the events related to the project.

Frame 1. An example of semi-structured interview questions.

1. What skills and results did you seek from the Sokrates project?

2. How was the contact with the project established?

3. What information and experience did you get from the project?

4. Did you utilise any of the results and if you did, what results?
   SA/SD based design method
   Coding rules for the C programming language
   Prototype of the code generator
   Operating system nucleus
   Communication software package

5. What was unnecessary in Sokrates from your viewpoint?

6. Would you have needed generation of code for other programming languages than C or from other source languages than SA/SD design models?

7. Did you consider the use of code generation after the project, did you buy any commercial code generators, such as Prosac?

8. What do you think of code generation at the moment?

9. Do you still use the SA/SD method in design?

A tentative summary (cf. [Seppänen et al. 1998a]) of the code generation case was written by the first author of this report (Appendix 3) and sent to the code generation researchers and to two other VTT researchers for reading before their interviews. Its purpose was twofold, to facilitate the writing of a more elaborated case history and to make explicit the managerial perspective of the first author as a participating actor, the focal manager and "key decision maker in an organisational setting" [Yin 1988].

The benefits of the case summary that was written in the form of a story, especially in the latter sense, were considerable. The informants could see on paper what a representative of the management thought about code generation and had the opportunity of telling their own stories. Several extensive augmentations of the initial summary, two comprehensive yet different written outlines of the same subject, an 18-page report of a group rehearsal that lasted for three hours, and a 90-minute interview of a co-researcher resulted from the triggering effect of the initial story. As a result, the initial case summary could be accompanied by the story of the code generation researchers, the "multiple actor-informants own constructions of their situation in the context studied" [Tikkanen 1998].

According to [Yin 1988] "the most desirable option is to disclose the identities of both the case and the individuals". However, since the study is carried out on a controversial topic, as is the case with the code generation R&D, in which the viewpoints of the key participants clearly conflict, anonymity helps to protect the real case or at least its participants. Yin points out that "the anonymity of the individuals alone may be sufficient, thereby leaving the case itself to be identified accurately". We have followed this suggestion.

Several thousand pages of documentary data were retrieved, mainly from the archives of the focal organisation. The most extensive set of personal notes studied included eight 200-page R&D diaries from the mid-eighties to the late nineties. Artefacts other than documents were also available for investigation, e.g. the code generator and its commercial counterpart, a small-scale model of an elevator for which the software had been generated and a marketing video tape, to mention just a few. The data collection process took as long as ten months, resulting in a data base comprising approximately 170 pages of case data. This data base included classified extracts from the analysed documents in a temporal order, the original grand story, the results of the group rehearsal, as well as the free-formatted data gathered from the interviews, which "may be considered as the formal part of the data base and not part of the final case study report" [Yin 1988].

The analysis of this case study evidence relied on the theoretical propositions based on the competence evolution framework, as discussed above. The process net part of the framework was used as the basic means for explanation building, denoting a special form of pattern matching based on causal links. Since these links were highly complex and difficult to measure precisely, we based the explanations on the theoretical propositions of the study rather than on the case description only. We did not use the formal time-series analysis, other than organising causal activity links involving different actors and relationships over time, mostly in tabular form. However, since the period of time that we studied is only about ten years and the archived project material included plenty of details, it was quite easy to unfold the chronological relations of events from the case data (cf. Appendix 2).

Explanation building is always an iterative process in multiple case studies, as shown in Figure 2. We have followed an iteration cycle not only with regard to the case described in this report, but also in our research as a whole.

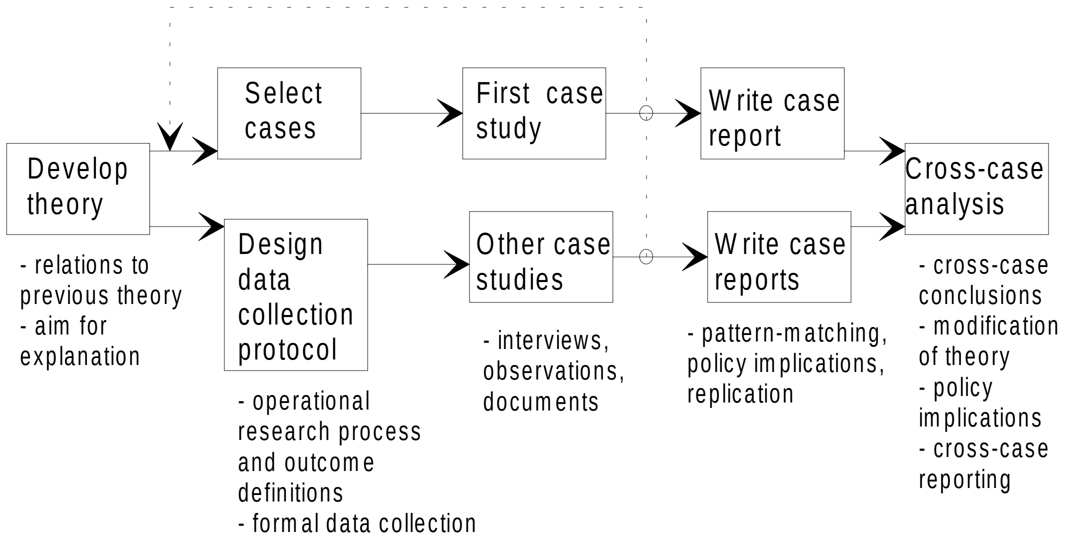

Figure 2. Case study method for multiple cases [Yin 1988].

The basic iteration loop consists of making the initial proposition based on relations with former theories, comparing the findings of the initial case study with the proposition, revising the proposition, comparing the details of the case with the proposition, revising the proposition again, and repeating this cycle for the other cases. Both in associating the case data with the research propositions and in interpreting the results, we rely on the competence evolution framework, which is being improved and validated in two different case studies.

[Seppänen et al. 1998a] presents the original framework through relating together existing theories on competence-based competition and industrial networks. The case of fault diagnosis systems is then analysed by means of the framework. This report describes the modifications made to the framework to help us to better explain goals and processes related to competence evolution. The code generation case is analysed, at the level of projects, by using the modified framework.

The “lesser mode of analysis” [Yin 1988] includes means for analysing embedded units, especially in multiple case studies. According to Yin, it is appropriate to conduct the analysis of embedded units first within each case, in order to consider the results as factors in pattern matching or explanation-building at the level of the individual case. Patterns of explanations should be compared between cases, rather than the results of the analysis of embedded units. We will follow this advice by analysing individual code generation projects, the most important embedded units, to explain the evolution of code generation competence. The patterns of explanations of the two cases will be compared at the end of this report. In this way, we aim at generalising the explanations analytically to research propositions.

According to Yin, a chronological structure of a case study report usually follows the early, middle and late phases of the case history, whereas a comparative case study report “repeats the same case study two or more times, comparing alternative descriptions or explanations of the same case” [Yin 1988]. We chose the first approach, structuring this report chronologically according to the development of the code generation case from the mid-eighties to the present date. However, in Appendix 3 we also present plots of an alternative story told by the code generation researchers.

3 COMPETENCE EVOLUTION FRAMEWORK

The framework presented in [Seppänen et al. 1998a,b] is introduced in this chapter, with a few extensions needed for explaining both the R&D process which was used for building code generation competence, and the business processes related to its marketing, leverage and exploitation.

3.1 ELEMENTS OF THE SUBSTANCE LAYER

The competence evolution framework, as presented in [Seppänen et al. 1998a], consists of two layers: the *substance layer* is used for outlining the basic elements of competence and relationships, whereas the *management layer* describes the evolution of these elements over time. The substance layer is based on the well-known activity-resource-actor (ARA) model of industrial networks [Håkansson and Snehota 1995], Table 1. It describes activities carried out in creating and utilising resources.

We consider competence as the ability of conducting purposeful activities on certain types of resources, much along the lines of the so-called resource-based perspective to strategic management of firms (e.g., [Foss 1997, Foss and Knudsen 1996, Lowendahl 1997, Rosenbröijer 1998]). The substance layer includes a typology of competence elements, developed by extending and reformulating the resource typology given in [Rosenbröijer 1998], and by associating it with a typology of R&D activities. The main characteristics (cf. attributes in [Easton and Araujo 1996]) of the elements are also addressed. The firm, relationship and network levels form the focal net dimension of the substance layer [Håkansson and Snehota 1995].

Typologies were used in [Seppänen et al. 1998a] to define classes and types of competence and relationship elements, and values to define their attributes. The existence of certain types of elements and the values of their attributes depend on time, which was thus also made explicit. [Rosenbröijer 1998] uses a resource typology based on *financial, physical, organisational, human, technological*, and *reputation* resources. This resource typology was modified and associated it with activities, actors, and relationships.

The research and development service is the main technological resource utilised in projects; the possession of some expert skills is required to carry out such services at the level of individuals. Physical resources can simply be considered as products, documents, or development tools. *Temporal resources* are missing from the typology used in [Rosenbröijer 1998], although they are crucial in project-based research and development. The basic types of temporal resources planned for, and controlled in projects, such as schedules, efforts, and calendar time, were therefore included in the typology. In addition to professional reputation, project management is one of the most important organisational resources in use, e.g. in marketing. Financial resources are almost always exchanged in contractual research and development, be it internal or involve external parties.

Research and development activities are required for acquiring resources, as well as planning for, carrying out, evaluating, and utilising R&D results. Additional activities include supporting individuals in developing and extending their expertise, taking care of project management, planning and controlling financial matters, following general technical developments, and acting as a member of the professional community.

Resource and activity attributes facilitate the study of the main characteristics of competence. In [Seppänen et al. 1998a] and in this research the focus was on the characteristics of technical competence, i.e. content of R&D services. It will be characterised by using the following four dimensions derived from the well-known market model of Abell [1980]:

- the *application* domain and particular products involved,
- the *functions* accomplished by products,
- the *techniques* on which the functions are based, and
- the implementation *technologies* used to realise products.

Although the typology was by no means comprehensive, but they were still used to explain the main concepts of the fault diagnosis case. However, during the analysis of the code generation case it became obvious that the typology was, after all, too complex. Therefore, we simplified it by considering human, technological and physical resources as *product* resources, temporal and financial resources as *process* resources and reputation as the most important *organisational* resource.

In terms of the ARA model, activities, resources and actors form the *substance* of contractual research and development. The firm, relationship and network levels form the functional dimension of the framework, called the focal net. Competencies are managed by the actors of the focal net, by carrying out activities on resources. Interrelationships between competence elements can be described as *resource collections, ties, constellation*s and *activity structures, links,* and *patterns*. We will use focal nets organised around projects as a key means of tracing and explaining the evolution of competence. A project is “a complex transaction covering a discrete package of products, services and other actions designed to create (capital) assets for the buyer over a certain period of time” [Tikkanen 1998]. We address competence development as a function of *project nets*.

However, contractual R&D does not involve only project personnel and activities, but also people involved in marketing and controlling projects and in results exploitation. We will address this by *business nets* organised around project marketing and purchasing. Their aim, from the supplier point of view, is to create and maintain portfolios of customer relationships that may be based on projects, but also on other kinds of competence leverage interactions. The activities of the process involve marketing, selling and general management by the supplier. On the customer side, business nets usually involve subcontracting, purchasing and general management, and sometimes also marketing to the customer’s own customers.

Table 1. The substance layer of the framework.

Focal net **Substance**	**Firm**	**Relationship**	**Network**
ACTORS	*Organisational structures:* e.g. project and research groups	*Actor bonds:* e.g. steering group of a joint blue project	*Actor webs:* e.g. joint project team of ELE, its customer and the customer's other subcontractor
COMPETENCE Activities	*Activity structures:* e.g. green project	*Activity links:* e.g. red project	*Activity patterns:* e.g. blue project
Resources	*Resource collections:* e.g. embedded systems modelling skills and tools	*Resource ties:* e.g. coding rules tailored to customer's needs	*Resource constellations:* e.g. integration of a CASE tool and a code generator

As already pointed out above, one of the most essential aspects of the substance layer is the content of product-related competence. It is modelled by using four dimensions extended from [Abell 1980]: the applications for which products are being developed, the functions that respond to the needs of the customers of these applications, the technologies enabling and used to implement the functions and the scientific or engineering techniques on which implementation is based. These dimensions can be illustrated by using the code generation case, taking into account both design (methods and tools) and solution (embedded software) related competence:

- *Solution*/applications: machine automation, electronic instruments,
- functions: real-time operating system functions,
- techniques: real-time software execution techniques,
- technologies: operating system, program library, protocol software.
- *Design*/applications: design of integrated circuits,
- functions: embedded system modelling, code generation,
- techniques: SA/SD design technique,
- technologies: code generator, graphical animator, software tester.

3.2 PROCESS-BASED VIEWPOINT TO CONTRACT R&D

The substance layer of the framework presented in [Seppänen et al. 1998a] makes the governance structure of relationships explicit through the actors and relationships of the focal net. In addition to this *organisational* viewpoint, the *product*-based viewpoint is included through certain types of resources. Although activity types are associated with resource types, the *process*-based viewpoint is not emphasised. The ARA model ties the elements of the substance layer together in a static way, whereas processes would link activity and resource elements together dynamically.

Processes would also facilitate the structuring of focal nets according to the roles played by different parties in carrying out activities on resources. For example, a steering group of a project typically controls the use of the financial and temporal resources of the project. It also accepts the results of the project. Since the competence of a contract research organisation that emerges trough the R&D process should be exploited in new customer relationships, it is important to make explicit also the business process related to competence marketing. The focal nets related to the two processes can thus be seen as special kinds of process nets, which is a concept used by Tikkanen [1997].

We will use Tikkanen as a starting point in introducing processes at the substance layer of the competence evolution framework, Figure 3. Tikkanen's work is one of the most recent and comprehensive studies involving a process-based viewpoint to industrial relationships: "the main idea explored in this book is whether it would be fruitful to study business processes through the increasingly rich conceptual arsenal offered by the industrial networks research tradition". According to Tikkanen, core processes are "central to actual business operations. Business network processes transcend organisational boundaries and connect organisations with their co-operative partners."

Tikkanen proposes process nets based on the ARA model as a means of making explicit and analysing business processes. In one of his two industrial case studies, Tikkanen uses the concepts of a focal business process and a focal project interchangeably, because "project supplies can also be regarded as meaningful business processes for certain organisations". Focal R&D process nets can thus be called project nets [Tikkanen 1998], because most R&D activities are carried out as projects. Business nets, called "the project marketing horizon" in [Tikkanen 1998], are used for marketing projects and other types of R&D services. From the customers' viewpoint, they represent the "project purchasing horizon".

The structure of project and business nets can be described by the governance structure concepts of the ARA model. These include internal and external actors, where the focal actor can be considered as the process owner, as well as their direct and indirect relationships. The resources and activities performed, controlled and co-ordinated by the actors form the production system of the nets, i.e. the functional process. In this way the process becomes an integrative concept for competence created and maintained by the actors of process nets.

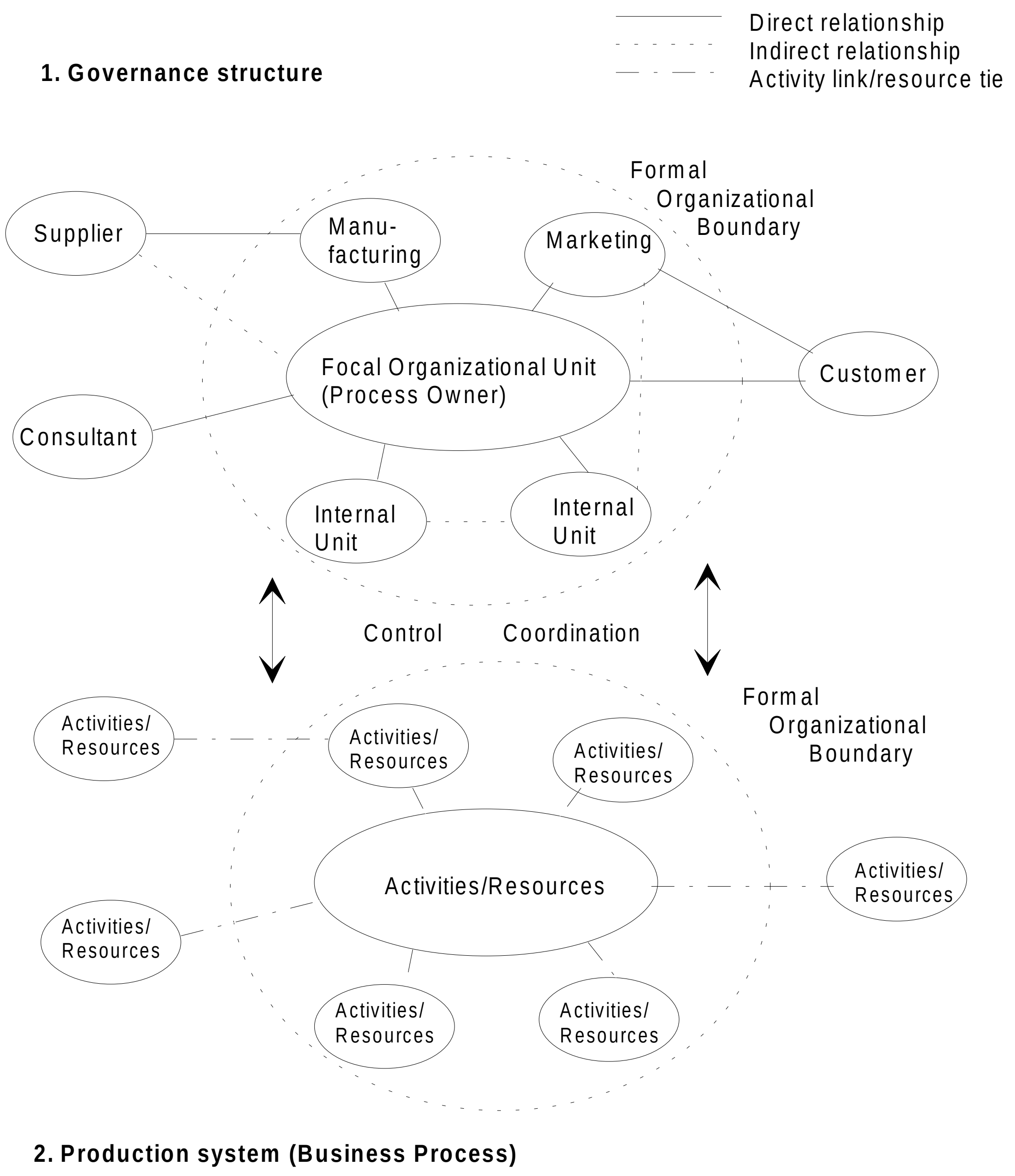

Figure 3. A hypothetical focal process net [Tikkanen 1997].

In the project-based business process of one of his case organisations, Tikkanen identifies four phases: bidding or negotiation, planning, design and implementation. He studies actor involvement and activities performed on resources in each of these phases, and actually also in a fifth phase, transition. These phases represent the seller’s point of view, i.e. the project marketing cycle [Tikkanen 1998].

In the context of VTT, the portfolio of green, blue and red projects provides a better basis for the life cycle of the R&D process, because it corresponds to the organisation's basic operational logic. We will consider green projects as competence innovation, blue projects as competence development and red projects as competence leverage according to this logic. The project marketing cycle concerns individual projects within the R&D process. There may also be a competence formalisation phase, such as writing of a doctoral thesis (cf. [Kurki 1995]) after carrying out several projects on the same topic.

Although the exploitation and evolution of competence are often tightly intertwined in project-driven organisations, we separate them conceptually. Since most customers of an engineering contract research organisation are product developers, a typical competence exploitation process of a customer involves using the competence for its own product or process development needs. In other words, the project net of the supplier is closely tied with the project net of the customer. Another, usually less tightly coupled, form of exploitation is the commercialisation of competence as intellectual property rights (IPR) purchased by customers.

We view the exploitation of competence through the marketing (supplier) and purchasing (customer) processes. At the supplier side, competence marketing aims at initiating or continuing research and development projects. For example, during the main code generation project Sokrates many activities were carried out to market customer projects, and a few such projects were contracted by industrial firms aiming at exploiting the developed competence. At the customer side, the competence purchasing process usually serves the development of new products or production processes. These four processes must be linked together by a fifth process, co-ordination and control of co-operation, Figure 4.

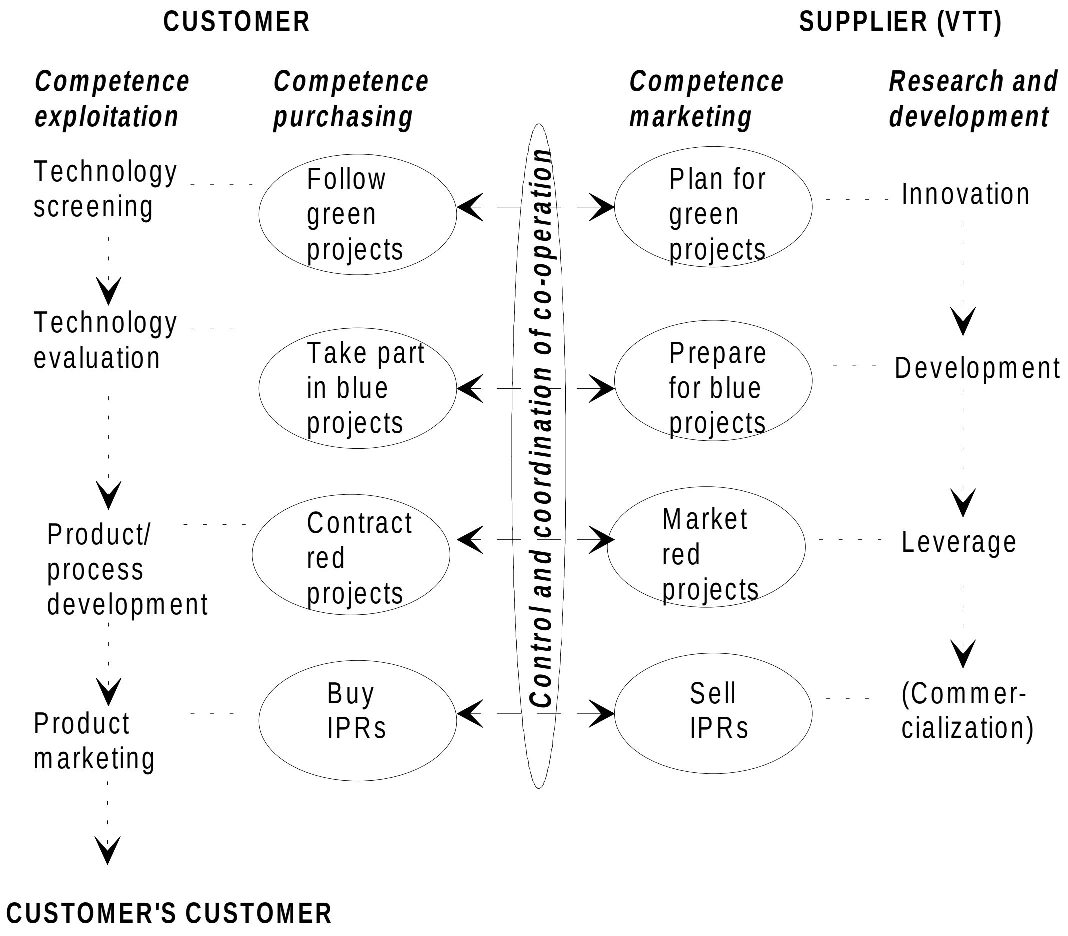

Figure 4. Interplay of the supplier and customer processes.

The code generation case analysed in this research shows that even within a single organisation the same processes can be seen from very different perspectives, depending on the goals and interests of the various actors involved. For example, managers usually consider the financial value of red projects as an important factor in maintaining the project marketing horizon, whereas researchers may see it as much less important:

> *Veikko Seppänen: "My point was the conflict that I as a section head, or ... [other line managers] was looking for hundreds of thousands or millions of marks [from red code generation projects]?"*
>
> *Code generation researcher: "Our point was that we were offering solutions to customers, which would allow them to produce things cheaply and fast. If the customer could get something at fifty thousand marks, for which they had spent half a million marks earlier, that was good. This was our way."*

Tikkanen claims that "individual project deliveries could be characterised as more or less unique" and that "it is practically impossible in most cases to identify clear beginnings and ends of business processes when these are viewed from the perspective of a core competence area". These claims do not hold for contract research organisations, in which projects are based on continuous and systematic evolution of certain competencies. For example, It would have been quite easy to identify not only activities related to individual code generation projects but also those related to the marketing and purchasing of the emerging competence.

One reason for Tikkanen's opinion may be that his analysis of focal process nets was based only on interviews, in which "it proved to be impossible to track any more specific sequential activity chains [in one of the case organisations] than the rather general ones". Furthermore, in "neither of the cases described and analysed in the focal net study was it possible to relate the resource dimension exactly to the specific activity structure of the focal process net", concerning especially "the more abstract intangible resources, such as technical know-how and personnel capabilities."

In the code generation case discussed in this report we found it very difficult to acquire facts related to individual activities and their results from managers. They simply appeared not to remember too much factual information, perhaps because they had had so many things to take care of. In Tikkanen's study, three of the five interviewed persons in the project delivery case were managers. Even the two interviewed project-level experts were some kinds of managers: a project foreman and a chief engineer of planning.

However, first by analysing documentary data and then by interviewing not only the supplier's and the customers' managers, but also the code generation researchers and their industrial counterparts, we found it relatively easy to identify the starting point of code generation research, the emergence of individual skills towards team-based competence, and the fading of the competence in the sense of a remarkably decreased volume of activities. Other important facts, such as times, persons, projects, technical results and financial figures could also be traced.

3.3 STRUCTURING OF PROCESS NETS

In a recent textbook Lowendahl proposes two dimensions, strategic focus and resource base, for classifying such professional service firms as contract research organisations, Table 2 [Lowendahl 1997]. The diagonal of this positioning consists of A, B and C types of firms. On one hand, the resource base can be controlled by the organisation as a whole (B), independently by the professionals themselves (A), or as a mixture of the two approaches (C). Along the other dimension, strategic focus on customers or customer groups aims at continuing interaction, whereas problem-solving based strategies involve a high degree of innovation and solution or output-based strategies seek to extend markets for uniform services. In practice, the three strategies are not mutually exclusive, but may co-exist in a single firm.

Table 2. Positioning of professional service firms [Lowendahl 1997].

Strategic focus/ Resource base	**Client relations**	**Creative problem Solving**	**Adaptation of ready solutions**
Organisationally controlled resources	(D) Insufficient adaptability	* → ↓	(B) Efficient
Team-based, individual and collective resources	* → ↓	(C) Both	↑ ← *
Individually controlled resources	(A) Flexible (Effective)	↑ ← *	(E) Lack of co-ordination, discipline

According to the logic of VTT, problem-solving dominates green and blue projects, whereas red projects are either customer driven or output based. During the competence innovation and development phases, resources are largely controlled by individual researchers. Competence leverage in red projects and commercialisation by selling intellectual property rights are, on the other hand, most often controlled by the organisation through managerial planning and co-ordination. The competence marketing process follows a similar pattern: green and blue projects are predominantly initiated and prepared by researchers aiming at creating research project nets, whereas the line management, focusing on maintaining the marketing horizon, is often quite extensively involved in the marketing of red projects and other customer-paid activities.

Pressures for the change of a position include, according to Lowendahl, moving away from the “stuck-in-the-middle” position for C types of firms, and moving towards the upper left (D) and lower right corners (E) for B and A types of firms. Contrary to Lowendahl, we believe that it would be beneficial for a contract research organisation to be “stuck-in-the-middle” with such *core platforms* as the fault diagnosis platform analysed in [Seppänen et al. 1998a], in order to be both efficient with regard to its competence and effective in its portfolio of customer relationships.

In order to emphasise the strategic aim at core platforms, at first we were planning to structure project nets according to the four dimensions determined by the content of technical competence. The governance structures of the processes shown in Figure 4 were associated with the applications, functions, techniques or technologies of the competence dealt with in projects. This was supposed to help us to explain the evolution of competence. For example, fault diagnosis functions were the dominating elements of the competence of VTT in the fault diagnosis case discussed in [Seppänen et al. 1998a], whereas the customers were focusing on their application skills. This combination seemed to be beneficial for both parties.

In the code generation case VTT was emphasising a specific design technique and certain implementation technologies, which resulted in conflicts with a commercial technology provider and with those technical experts who promoted different techniques. The evolution and exploitation of the competence was successful, when the specific technique was favoured by both parties and the technologies were accepted by the purchaser. Still, our attempt to structure the code generation project nets according to all of the four dimensions of competence failed, as there was simply not enough variation in the dimensions. Therefore, we structured the project nets according to the phases of the research and development process. Correspondingly, business nets were formed on the basis of the phases of the competence marketing and purchasing processes.

3.4 MANAGEMENT OF COMPETENCE CHANGES

The management layer of the framework makes explicit how and why the elements of the substance layer change over time. The substance layer is based on two dimensions: substance elements consisting of resources, activities, and actors, and the process net expressed at the internal firm, and the external relationship and networks levels. In the management layer, we try to answer the question of how purposeful activities carried out by actors to develop and use resources, i.e. the evolution of competence, affects the external relationships of the focal organisation.

We will address changes in project relationships rather than other types of supplier-customer transactions. The very basic types of changes in project relationships are rather simple: beginning and end of an internal or external project; change of an internal project to an external project and vice versa, which involves a change of the colour of the project; and change of a dyadic project to a multi-party project and vice versa. In principle, a project may change into any other type of a project. However, the strategic goal of VTT is first to extend its internal green projects into joint blue research projects, and then to several distinct customer projects, or if possible, to multi-party consortia. Shortcuts to contractual projects from internal activities are possible, although risky, whereas reverse changes are more seldom. The completion of internal projects without any continuing external projects is usually considered to be a failure, whereas the possible negative effects of the ending of an external project depend on the resources involved. A single project relationship may on the other hand, continue for several years.

Although their basic changes are simple, project relationships will evolve in a complex manner over time. We will use the four relationship attributes based on [Ford et al. 1986], *capability, mutuality, particularity*, and *inconsistency*, to identify and analysed changes of relationships that affect competence. Based on the interrelationship of the attributes, Ford and others propose four functions to manage changes of relationships. We will, however, use them only to analyse changes of process networks and thereby changes in resources and activities, i.e. competence:

- Balancing the particularity of resources (capability vs. particularity)
- Assuring the capability of resources (capability vs. inconsistency)
- Balancing the mutuality of activities (mutuality vs. particularity)
- Assuring the mutuality of activities (mutuality vs. inconsistency).

These functions involve distinct aspects of competence: balancing of particularity deals with the *codification* and *contextuality* of resources, assuring of capability is related to the life *cycle of the content* of resources, and balancing and assuring of mutuality address *institutionalisation* of activities. Since the relationship change management functions are interrelated as shown above, functions to manage changes in codification, content and institutionalisation of competence become also interrelated. Both the relationship and competence change management functions will now be discussed, providing examples from the code generation case.

Balancing particularity - codification/contextuality of competence

Balancing of the particularity of resources concerns the question about the extent to which resources should be tailored to relationships with particular actors. This involves codification and contextuality of resources.

Development of manual coding rules in a red spin-off project of Sokrates is an example of balancing of the particularity of resources (cf. [Toivanen 1992]). The change of the blue Sokrates network to the red project relationship required VTT to increase the particularity of certain product-related resources. This resulted into a more customer-specific and tacit piece of code generation knowledge, possessed by the researchers involved in the project. Development of the Sokrates code generator was based on the results of a former project, whose customer took part also in Sokrates. This is an example of a reversed change, decreasing of the particularity of product-related resources from the viewpoint of the customer company.

Particularity of process-related and organisational resources is also closely related to the project portfolio. Red projects may involve many different types of customer-specific arrangements, whereas blue projects usually follow uniform organisation principles set by the funding body.

We consider the codification and contextuality of resources at three levels: tacit alias context-specific, portfolio-specific and generic. Contextuality involves especially process-related and organisational resources and codification product-related resources.

Two types of competence change management functions are associated with codification/contextuality managed by balancing of the particularity of resources. *Problem-solving* refers to the process of codifying and giving a structure to tacit product-related knowledge possessed by certain actors, which may lead first to customer portfolio-specific and ultimately to fully generic solutions. Contextuality of process and organisational resources decreases as a result of this change, i.e. organisational and process-related resources become less dependent on a certain context.

Absorption is a reversed competence change management function. It means the application of codified knowledge to different situations or the transfer of context-independent process and organisational resources for some specific use. The Sokrates project focused extensively on problem solving regarding product resources, because its main goal was to produce fully generic design methods and tools. Its subsequent red projects are good examples of how the developed methods and tools were aimed to be absorbed by different industrial customer companies. In terms of process and organisational resources, this meant that the code generation researchers needed to make use of specific industrial funding and projects instead of a joint national information technology research program.

Problem solving and absorption are critical for the integration of interacting elements of competencies. [Rosenbröijer 1998] uses the concepts of "directed capability" and "connection function" to describe resource integration in relationships. We will take a more simple approach, by analysing if competence elements *strengthen*, *complement* or *compete* against each other when integrated, or if they are *neutral* in this regard.

Balancing and assuring mutuality - institutionalisation of competence

Because of the risks involved in the lack or excess of particularity, a firm must carefully consider the extent to which the mutuality of its activities is related to the particularity of its resources. The Sokrates project suffered from the lack of mutuality in documentation activities. Documents were expected by the steering group, to be used for controlling the project and making use of its results. Some of them were either not properly delivered or not written. On the other hand, most red code generation projects showed a very high degree of mutuality. The former Sokrates project members diffused the skills needed in these projects to each other, often informally and without any active involvement by the management.

Assuring the desired level of mutuality involves inconsistency. It is concerned with the learning of persons involved in a relationship. For example, inexperienced persons may be allocated to projects in which certain knowledge plays a central role. Four of the seven key members of the Sokrates project were research trainees. Their inexperience as professional researchers may have affected the mutuality of their activities. The industrial members of the steering group of the Sokrates project claimed, on the other hand, that it was especially the senior researchers who were not listening attentively enough to industrial feedback.

We consider the institutionalisation of activities to include the levels of individuals, project and organisational teams, the whole organisation, and inter-organisational relationships. Balancing of mutuality at these levels involves the *diffusion* and *scanning*. The former denotes sharing insights, knowledge or skills within a larger community of actors, the latter refers to the identification of opportunities hidden in some data. Scanning is often performed by individual researchers when planning for projects. Diffusion involves, for example, organisational control in managing the allocation of human resources to distinct projects.

From the viewpoint of knowledge creation, which is important in the code generation case due to its innovation-related characteristics, competence changes can be easily associated with the socialisation, externalisation, internalisation and combination functions proposed in [Nonaka and Takeuchi 1995]. Tacitness and explicitness of knowledge, i.e. codification, constitutes the epistemological dimension of the knowledge creation model of Nonaka and Takeuchi. Externalisation that transforms tacit to explicit knowledge can be likened to problem solving and the reversed internalisation function to absorption. The ontological dimension of the knowledge creation model involves social interaction between the people that create and share knowledge. Conversion from tacit to explicit knowledge is called socialisation. Explicit knowledge can be turned into other explicit knowledge through combination.

Problem solving, absorption, scanning and diffusion, as they are presented above, do not make any clear difference between tacitness and explicitness of knowledge with regard to its institutionalisation. The code generation case data indicates, however, that this is important. Much of the tacit knowledge was not externalised into organisational competence, as opposed to the fault diagnosis case discussed in [Seppänen et al. 1998a] where an explicit core platform emerged. To study this phenomenon, we will redefine diffusion and scanning as diffusion by socialisation or combination and scanning by socialisation or combination.

Assuring capability - coping with the evolving content of competence

Also assuring of the capability of resources involves inconsistency. The code generation researches were extremely capable in the SA/SD design method that was used as one starting point of the Sokrates project [Ward and Mellor 1985 - 86]. The skills are illustrated by the numerous professional, in-house and academic courses that the code generation researchers gave on the method in the eighties and nineties.

On the other hand, the researchers were later criticised of having neglected discontinuity in the life-cycle of embedded systems design techniques. By the end of the nineties the formerly dominating SA/SD method had been mostly substituted by the so-called object-oriented methods. Assuring of capability involves thus management of the life cycle of the content of resources. The life-cycle phases of incremental changes, discontinuity, substitution, competition and dominance are defined for each of the four dimensions of the content of product-related resources.

Life-cycle management means coping with these phases, concerning a certain dimension of the content. The life cycle of process and organisational resources is closely related to the project portfolio. For example, the life cycle of a financial resource may change from self financing to partial and full external funding thanks to green, blue and red projects. Figure 5 summarises the results of the above discussion, by showing the three directions of change, the four relationship management functions and the five competence change management functions.

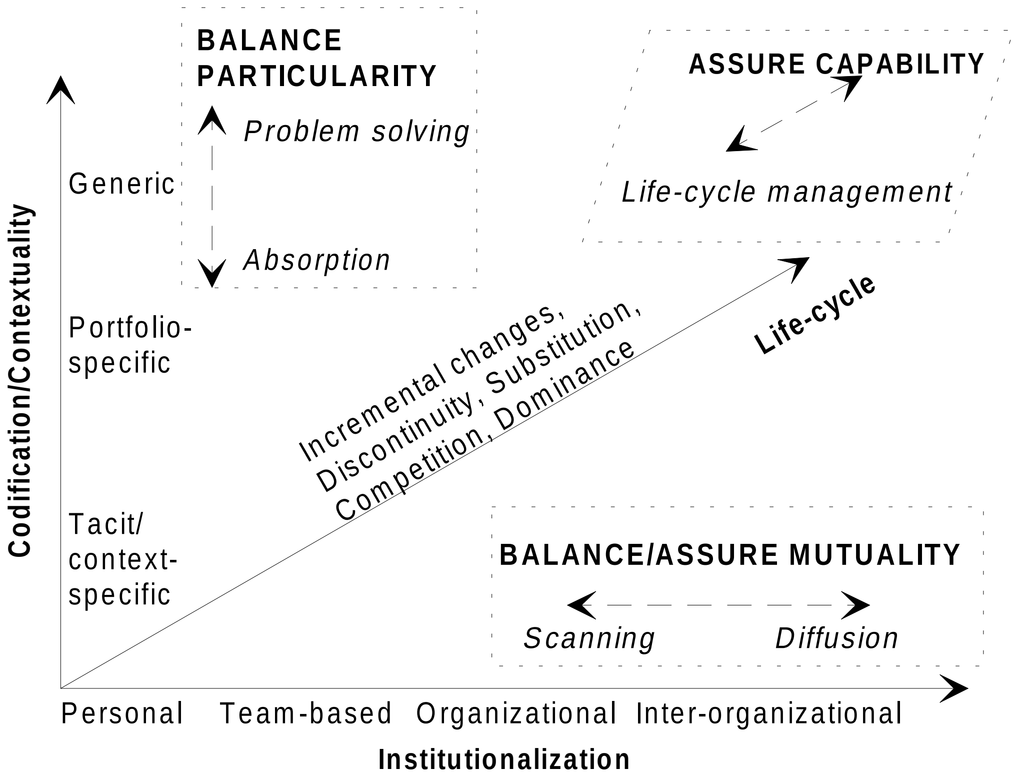

Figure 5. Management of competence changes.

The change of project nets as a whole could be characterised as the evolution of *innovation nets* [Miettinen 1998] to *competence exploitation nets,* by shifting the focus from knowledge creation *for* relationships to competence leverage *within* relationships. The characteristics of the innovation nets of VTT are discussed in [Miettinen 1998], mainly from a social perspective. Miettinen does not, however, address the evolution of innovations to commercial products and services.

This could be done, for example, by comparing the intended business nets and the actual project nets from the viewpoints of both suppliers and customers. Especially jointly funded blue projects can be considered a kind of no-man’s land in this regard, because the parties involved may strongly disagree on who has the right to use the results and how.

3.5 EXPLAINING THE LOGIC OF ACTION

The attributes proposed in [Seppänen et al. 1998a] for characterising the actors involved in project nets are very simple, such as the role of an individual either as a manager or a researcher, and the role of an organisation as a research institute or an industrial company. The code generation case data indicates various differences in the interests and several conflicts in the viewpoints of actors involved in project and business nets. This kind of social embeddedness [Holmlund and Törnroos 1997] can be used to explain the evolution of competence and the change of process nets. Miettinen [1993] discusses a similar phenomenon in innovation-related networks, stating that "*voices of actors*" reflect:

1. different *interests* of the institutions involved in innovation networks, such as the researchers, producers and users; because the actors have different roles in their organisations and in the network, they usually see the interests of the institutions from different viewpoints,

2. the educational *background*, job history and personal interaction network of each of the actors, which depend on the actor's view to the roles and positions of the other actors of the innovation network, and historically, illustrate how the actor's views have been formed from his or her consideration, opinions and experience from items that originate from many different sources and have been affected by other actors' views, and

3. the *position* of the actor in his or her organisation, and the division of labour in the organisation; depending on the position of the actor, contradictory concepts may be pointed out from the same network, depending in part also on the relations of power, competition and career ambitions of the actors.

In [Eriksson and Räsänen 1998], the concept *logic of action* is used for integrating and reasoning about the goals, means and justifications of groups of people dealing with changes of product mixes.

The content of a certain logic of action was constructed by Eriksson and Räsänen from their case data, based on the projects and actions that each group of actors actually proposed, championed or objected concerning product mix changes. The case data was also used to make explicit three types of interactions by which groups responded to the actions of the other groups over time, called dominance, compromise and integration.

We are using logic of action to make explicit the multivoicedness in process nets, including both project and business nets. Within the focal organisation we address the logic of action of certain types of people, such as line managers, code generation researchers and other VTT researchers. The logic of external actors is described only at the level of certain types of organisations, such as funding bodies, other research institutes and industrial customers. Our approach is compared with that of Eriksson and Räsänen in Table 3.

Table 3. Making logic of action explicit in process nets.

[Eriksson and Räsänen 1998]	**Process nets**
Manager groups	Groups of actors
Relationships between managers	Relationships of groups of actors
Form of interaction (dominance, compromise, integration)	Form of interaction (dominance/submission, co-operation, competition)
Objectives and goals	Objectives and goals
Means	Means based on resource creation, possession and mobilisation
Justification of the logic of action	Justification based on background, interest and position of actors
Source of influence	Control over and importance (value) of competence
Confectionery products	Competence
Product mix changes	Evolution of competence
Management of product mix changes	Management of competence

Groups of actors with various objectives and goals to pursue are interacting within the context of process nets. The logic of action of a group can be identified by analysing how its goals are associated with some means. The objectives, goals and interaction of actors illustrate the social embeddedness of relationships. The forms of interaction in industrial relationships have been extensively studied (cf. [Alajoutsijärvi 1996]). We will use the basic notions of dominance or submission, co-operation and competition.

As indicated in the table, the means of an actor group to pursue some goals and to affect other actors depend on the resources that the group creates, possesses and is capable of mobilising. This view emphasises the technological embeddedness of relationships. The main source of influence is based on the ownership and control over the resources which have a certain importance alias value. For example, VTT group managers usually possess all the financial resources needed for managing their groups, whereas individual researchers can control only the financial resources of their projects as project managers.

The actual content of some logic of action consists of the projects and other activities carried out by a certain group of actors. In [Eriksson and Räsänen 1998] these involve the management of changes of confectionery product mixes, in our case the management of competence. Within the context of inter-organisational relationships the *backgrounds* of certain types of actors justify in part their logic of action The background of actors in terms of the history of their relationships can be made explicit by the means of process nets. We will provide such a history for the code generation relationships.

However, backgrounds of actors include also histories of many other relationships, as well as other developments that do not involve any relationship. For example, the company for which the very first code generation pre-study was carried out, had a long co-operation history with VTT. The R&D manager of that company, who was responsible for a joint strategic alliance, used to work as a line manager at VTT. Several members of the steering group of the Sokrates project were former VTT researchers, and so on. We will make explicit these kinds of background to the extent that is beneficial to understanding the code generation case.

Miettinen [1998] suggests three types of artefacts alias resources to characterise the *interests* of actors, which also justify their logic of action. The "What" type of primary artefacts denotes tools that are used for carrying out activities by some actors. The Sokrates code generator studied in this research is an example of such an artefact. The interests of VTT and the tool vendor that participated in the steering group of the Sokrates project were competitive with regard to the commercialisation of the generator. Yet, the two organisations did co-operate during the project. The interests of the VTT managers and code generation researchers conflicted over the same issue, because the managers did not consider commercialisation to be aligned with the objectives of the research institute, as opposed to the use of the generator in red projects.

Secondary "how" type artefacts describe the use tools as part of textbooks, journals and papers. The Sokrates design method, a version of the SA/SD method developed in the Sokrates project, is a good example of this kind of resource. The code generation researchers showed a lot of interest in developing and transferring the method to industrial embedded systems designers. This was not regarded as significant by the VTT managers, in terms of their logic of action that aimed at creating industrial income from research investments.

"Why" type of secondary artefacts explain reasons for certain phenomena. Solving of concurrence problems, by which the focal researchers motivated their topic of interest, is an example of this kind of a resource. The researchers wished to see themselves as solution providers to concurrence problems, but their customers saw them more as tool developers that should take responsibilities similar to commercial tool vendors.

Tertiary "where to" artefacts describe the future developments of primary and secondary artefacts, typically as a vision that characterises the goals to be achieved. A design automation vision for embedded systems was presented at the beginning of the Sokrates project, Figure 6. Some of the industrial partners of the project had a very vivid picture of this vision even after ten years of its presentation.

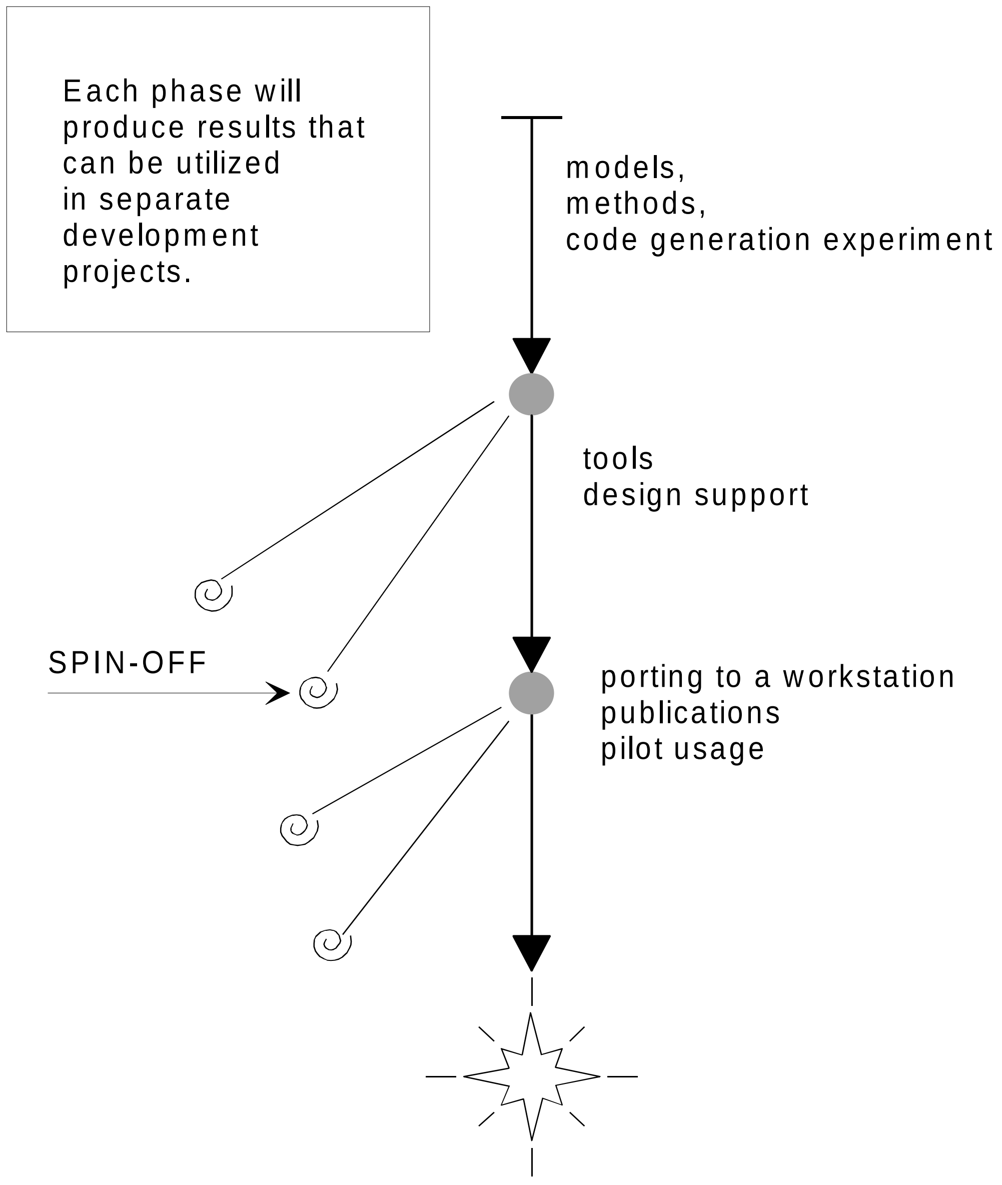
Each phase will produce results that can be utilized in separate development projects.
models,
methods,
code generation experiment
tools
design support
SPIN-OFF
porting to a workstation
publications
pilot usage

Figure 6. The Sokrates vision, shown as a "guiding star".

Another aspect justifying the logic of action of certain actors is their *position* in process nets. Johanson and Mattson [1997] define the position of an actor in a network of relationships as follows: "Thus, according to the extended definition, the position of an actor includes also the productive process - in a broad sense - in which it is involved and its direct and indirect network interdependencies. The production role has two dimensions ... The qualitative dimension describes which function the actor has in the production system. ... The quantitative dimension characterises the relative importance that the resources of the actor have in relation to the resources of the other actors, i.e. how much of the total quantity of sustainable resources are controlled by the actor". The position of an actor within a process net can thus be characterised by its activities on certain resources that have some relative importance.

We will address the importance, or value, of competence from the supplier and purchaser viewpoints, based on the longitudinal *expected* (before), *perceived* (during) and *historical* (afterwards) values. This kind of view fits with our research approach. The expected value of a competence element is usually documented in various kinds of plans describing how some resources will be developed or utilised, e.g. in contracts, project plans and minutes of the first project management group meeting. The expected value may also be evaluated by some external party.

The perceived value of a competence element evolves during its development and is documented e.g. in the minutes of project management meetings and in the feedback of the parties or some external evaluators given during or immediately after the work.

The historical value of a competence element can be evaluated after some time. In this research the opinions of the interviewed parties illustrate the historical value of code generation competence, after a few or as many as ten years.

One of the best examples of the change of the importance of competence from expected via perceived to historical value is provided by a project, in which a code generator was developed and used for the needs of a customer company. The customer's view to the value of the planned and realised results of the project were as follows (cf. Appendix 2):

> *Expected value (1993, before the project): "The purpose of the contracted work is to build a real-time software development environment for the MCS project of the customer. The starting point of the project includes the Finsoft/Sokrates technology taken in use and developed further in the RHU project [of the firm Nm]."*
>
> *Perceived value (1994, end of the project):"The specification of the project succeeded well and its implementation corresponds to the specification. VTT's attitude has been positive during the whole project, and problems have been solved fast."*
>
> *Historical value (1998, interview):"Thank you for the questions. I have waited for a possibility to give some feedback. ... Technically, the contribution of VTT was good. From the viewpoint of the business [of the firm Nm] it was fatal."*

4 ANALYSIS OF THE CASE DATA

The case data included in Appendix 2 and the code generation stories presented in Appendix 3 will be analysed in this chapter by using the revised competence evolution framework. The analysis is carried out chronologically, based on three main periods that we call Speco (1985 - 1987), Sokrates (1988 - 1991) and Reagenix (1992 - 1998). They involve the competence innovation, development and exploitation phases of the R&D process carried out by VTT during the past thirteen years.

The case data in Appendix 2 has been organised around two closely related processes of VTT as an R&D supplier: competence marketing, with a focus on the project marketing horizon, and competence building based on research and development. Competence purchasing and exploitation processes are addressed at the customer side. The four processes are associated together by a fifth process, the control and co-ordination of co-operation. Tekes, the public funding body, was closely involved in this process in 1988 - 1991. The resources and activities that resulted in and shaped the code generation competence took place, in practice, within the five processes carried out within the corresponding process nets. We have simplified the nets to two types of governance structures in Chapter 3, called the project and business nets.

As indicated in Figure 7, we will first analyse the outer context of the case and then proceed to the inner context. The six types of key actors involved in the case are then addressed: the managers of the focal organisation, the code generation researchers, their colleagues, industrial customers, research customers and Tekes. The goals and logic of action of these actors, which affected the code generation process nets and processes and thereby the code generation competence, are also analysed.

The actual code generation project nets are compared with the planned business nets, i.e. the intended project marketing horizon of the focal organisation. The project nets are also compared with the project purchasing horizon, to the extent that the intended purchasing of the code generation competence can be made explicit from the case data. The nets are structured based according to the phases of the R&D process.

The forms and means of interaction of the parties within the process nets are first addressed by evaluating the balancing and assuring of the mutuality of VTT activities, based on the diffusion and scanning functions. Then, the content of the logic of action is discussed by evaluating the balancing of particularity and assuring of capability of code generation resources, based on the life-cycle management, problem solving and absorption functions. Finally, the evolution and valuation of competence are analysed, concerning product-related, process-related and organisational competence elements. In closing, the results of the analysis are summarised from the viewpoint of building and exploiting core competencies.

The outer context (Section 4.1) affects

Actors ⇐ Objectives/goals: backgrounds, interests and positions

⇑⇓

Logic of action

⇑⇓ *… the process nets (Section 4.2)*

Relationships ⇐ Project marketing and purchasing horizons
⇐ Assuring and balancing of mutuality
* forms of interaction
* scanning/diffusion by socialisation or combination

⇑⇓ *… that enact the processes (Section 4.3)*

Processes
Customer: purchasing, exploitation
Supplier: marketing, research and development
Co-operation: control and co-ordination

⇑⇓

Content of the logic of action

⇑⇓

Activities on Resources ⇐ Assuring of capability by life-cycle management
⇐ Balancing of particularity
* absorption and problem solving

⇑⇓ *… which shape the competence (Section 4.4).*

Competence ⇒ Evolution of resource creation and usage
* evolution and valuation of the (core) competence

Figure 7. Analysis of the evolution of competence.

The bold-faced items on the left-hand side of the figure represent the elements of the substance layer of the competence evolution framework. The activities, which we view from a process-based perspective, realise the logic of action of the actors involved in focal nets. The analysis of the evolution of the code generation competence thus proceeds from causes to effects. The items on the right-hand side of the figure involve the management layer of the framework, i.e. management of changes in focal nets and processes. The relationship change management functions of assuring and balancing of mutuality affect focal nets, whereas the functions of assuring of capability and balancing of particularity affect processes. Scanning and diffusion support the former, life-cycle management, absorption and problem solving the latter.

4.1 OUTER CONTEXT – SOFTWARE ENGINEERING

[Tikkanen and Alajoutsijärvi 1998] define the outer context of industrial relationships as “an extension of the connected network directly relevant to a customer-supplier relationship and its inner context”, where the connected network focuses on “whole organizations as collective actors”. The phenomena that have directly affected the code generation relationships and internal activities of VTT and its customers as collective actors can be characterised as *industrial embedded software engineering*.

As indicated in the case data, the development started in the seventies, when microprocessors became incorporated in other kinds of products than general-purpose computers. Software development was at first done much as a side-job of hardware design, emerging as a new profession by the mid-eighties. As part of this development, Tekes started to actively support national research and development of embedded software engineering in Finland. Firm I started the Finnish business on embedded software engineering methods and CASE tools in 1985.

We will first outline the changes in the industries developing and applying embedded software in Finland, focusing on the electronics industry. The case data is used for discussing the position of the focal organisation in this context. The technological changes which have shaped embedded software engineering approaches are then summarised. The role of the focal organisation in regard to these changes is briefly evaluated.

Frame 2 lists the data sources that we have used in tracing the industrial changes related to embedded software engineering from 1985 to 1998. A considerable difficulty in this analysis arose from the fact that embedded software can be incorporated in almost any kind of electronic products. No studies have yet been carried out on the role of embedded software in even the best-known Finnish electronic products. The nineteen data sources listed in the frame provide thus only for a broad overview of the subject.

One of the most thorough analyses of the evolution of the Finnish electronics industry is given in [Lovio 1993], ending, however, with the year 1989. We have also used [Kivisaari and Lovio 1993, Mårtensen et al. 1985] and a few reports of the local activities in the Oulu region to provide some insights in the industrial context at the time when code generation related activities were initiated at TKO. Hannu Hakalahti, the director of TKO, was involved in carrying out some of the local studies. Several national information technology studies and strategic plans are available from the early nineties. We selected the ones that were acquired by the managers of the focal organisation and used by them to support the strategic organisational planning. Later in the nineties, both Tekes and the Association of the Finnish Electronics and Electrical Engineering Industries have published several surveys and forecasts, as indicated in Frame 2. Embedded software was, however, included as a distinct section in the forecasts only as late as in 1995 by Jukka Karjalainen, the former vice laboratory director of TKO. The corresponding section in [Hienonen 1997] which we have used in this research was authored by Veikko Seppänen.

Frame 2. Sources of data on the outer context from 1998 to 1985.

Anon. 1998. Huippuosaajat maailmalla. Sähkö- ja elektroniikkateollisuusliitto. 21 p. (in Finnish)

Hienonen, R. 1997. Elektroniikka- ja sähköalan kehitysnäkymät 1997...2002. VTT Automation. 229 p. (in Finnish)

Anon. 1996. Teknologia 2000. Osaamisella tulevaisuuteen. Tekes. 120 p. (in Finnish)

Anon. 1994a. Suomi tietoyhteiskunnaksi – kansalliset linjaukset. Ministry of Financing. 73 p. (in Finnish)

Anon. 1994b. Osaamisstrategia. Sähkö- ja elektroniikkateollisuuden menestystekijät, avainteknologiat ja osaamisen kehitystarpeet. Seteli. 29 p. (in Finnish)

Lovio, R. 1993. Evolution of firm communities in new industries. Acta Academiae Oeconomicae Helsingiensis. Sries A:92. The Helsinki School of Economics and Business Administration. 304 p.

Anon. 1993a. Teknologiakatsaus 1993. Tekes. 77 p. (in Finnish)

Anon. 1993b. Kansallinen teollisuusstrategia. Kauppa- ja teollisuusministeriön julkaisuja 1/1993. Ministry of Trade and Industry. 124 p. + app. (in Finnish)

Kivisaari, S., Lovio, R. 1993. Suomen elektroniikkateollisuuden merkittävien innovatiivisten liiketoimintojen menestyminen 1986 - 1992. VTT Technology Research Group. 51 p. (in Finnish)

Anon. 1992a. Teollisuuspoliittinen linjaus. Seteli. 16 p. (in Finnish)

Anon. 1992b. Sähkö- ja elekteroniikkateollisuus. Seteli. 7 p. (in Finnish)

Anon. 1990a. Valtion tiede- ja teknologianeuvosto: katsaus 1990. Tiede- ja teknologiapolitiikan suuntaviivat 1990-luvulla. 76 p. (in Finnish)

Anon. 1990b. Teknologiaohjelmatoiminnan linjat 1990-luvulle. Komiteamietintö 1990:2. 112 p. + app. (in Finnish)

Anon. 1989. Oulun läänin teknologiapoliittinen ohjelma. Oulun lääninhallitus. 54 p. (in Finnish)

Anon. 1987a. Uuden teknologian tuotekehitys- ja tutkimuskeskus, perustamisselvitys. Teknologian diffuusio, TEDI-87. Projektiraportti n:o 1. City of Oulu. (in Finnish)

Anon. 1987b. Uusia keinoja teknologiayhteistyön tehostamiseksi Oulun seudulla. Teknologian diffuusio, TEDI-87. Projektiraportti n:o 2. City of Oulu. (in Finnish)

Anon. 1987c. Pohjois-Pohjanmaan elinkeinoelämän tietotekniset toimintaedellytykset. Pohjois-Pohjanmaan Seutukaavaliitto. Publications Series A:84. 61 p. (in Finnish)

Klus, J.P., Markkula, M., Venho, J., Järvenpää, A., Sirkeinen, U., Ahlroos, R. 1985. Effective technology transfer. 128 p.

Mårtenson, G., Otala, M., Wiio, O.A. 1985. Tietotekniikka 1990-luvulla. Sitra. Series B, No. 78. 112 p. (in Finnish)

4.1.1 Evolution of the electronics industry 1985 - 1998

Matti Otala wrote the chapter “Information technology in products” in the forecast [Mårtenson et al. 1985]. He was using microprocessors used in cars as an example of embedded systems, but pointed out also other automation applications, such as elevators, cranes, heavy-duty machines, public and industrial transportation equipment, as well as robots and buildings. The section “Software production” (pp. 60 - 61) indicates that embedded software was applied especially in machines and equipment and would be developed “semi-automatically” by using special software engineering environments based in part on “artificial intelligence techniques”. Otala was expecting the automation in the development of embedded software to increase “slower than [in] the design of other electronics [technologies]”, three to four times by 1990. This was much the view of firm K, when it contracted the Speco project in 1985. In reality, embedded software became crucial in telecommunication, but only ten years later. The size of software incorporated in such products increased by the factor of four in 1994 - 1998, but the productivity in software development only by the factor of two.

[Lovio 1993] analyses the growth of the Finnish electronics industry from 1961 to 1989. After a deep decrease of growth in 1981, the level of a 15% annual increase was reached in 1985. Then, the growth ceased again and was well under ten percent in 1989. Lovio considers the period between 1975 and 1989 a competence enhancing phase of the Finnish electronics industry, comprising the era of ferment based on microelectronics from 1975 to 1984 and the era of incremental change between 1985 and 1989, with “new products in all old product groups” being introduced, and both exports and internationalisation increasing. Considering the firms that were involved in the code generation case, N, Nm, Nr and Nt as parts of a larger corporation and the firm V were among the top five, when ranked by their number of employees in 1985. Firm K was among the top ten and the firm W among the top twenty. In 1989, the top five firms had remained the same, but firm K had dropped down as many as nineteen positions. The firm W had risen among the top fifteen and the firm Th among the top twenty.

[Kivisaari and Lovio 1993] analyse “innovative businesses” of the Finnish electronics industry between 1986 and 1992 - during the innovative phase of code generation R&D. The firms K and V are included as large innovative firms in the product group “Industrial automation and measurement devices”, the firms N and Nt as large in “Telecommunication equipment”, the firm S as large in “Consumer electronics” and the firm W as medium-size in “Medical electronics”. Firm Nm had grown by 25% between 1986 and 1992, firm Nt 21% and even firm K 17%. The turnover of the firm S had increased by 9% and that of the firm W 5%. The turnover of the firm V had decreased by 4% and it had laid off 900 employees. The biggest product group in terms of the annual turnover was in 1992 the telecommunication sector with its 7.2 billion marks (43% of the total production, increase of 6% from 1986). The turnover of the automation sector was 4.8 billion marks (14%, decrease of 4%) and of consumer electronics 3.2 billion marks (6%, decrease of 14%). In contrast, the annual turnover of medical electronics was only 1.2 billion marks (2%, decrease of 2%).

TKO had at the time of the preparation of the Sokrates project three key customers. They included the firms K and N, but the former was seen as a “lowering customer” in terms of the volume of the co-operation. It had first increased greatly from half a million marks in 1984 to 2.3 million in 1985, but then decreased to 1.9 million marks in 1987. Income from the co-operation with firm N had increased rapidly from a half million in 1983 to 1.5 million in 1984, but then settled at the level of a million marks by 1987. The biggest increase and volume of income involved Tekes - from 0.8 million marks in 1984 to 1.5 million in 1985 and to 3.2 million in 1987. The total volume of the industrial income had increased from 4.9 million in 1984 to 6.8 million in 1985 and to 7.4 million in 1987. It was thus more than twice bigger than the income that TKO received from Tekes.

Two years later in 1989 TKO had about 25 industrial customers. The industrial income was about 6 million marks, i.e. it had started to decrease due to the extensive involvement of the institute in Tekes-funded research programs. The estimated total of annual R&D driven subcontracting volume in the electronics industry was 120 man-years. In the machine and equipment manufacturing industry the estimated annual embedded systems development volume was about 500 man-years, but increasing by 25% annually, especially in software development. Yet, R&D driven subcontracting was estimated at only 25 man-years annually. The share of TKO of the total annual R&D subcontracting market of the two sectors, 145 man-years, was thus approx. 14% in 1989.

Two years later, in 1991, when the Sokrates project was finished, 37% of the financing of TKO came from public funding bodies, mostly from Tekes. Another 37% came from industry. The situation had thus changed considerably from what it was in 1987 before Finsoft started. The number of the industrial customers of the laboratory had remained the same, but in addition to some fifteen electronics firms TKO had as many as ten machine and equipment manufacturing firms as customers. In 1987 it had had only three contractual customers from the machine automation industry.

Comparing the customer related data of TKO from the mid-eighties to the early nineties with the data on the evolution of the electronics industry, it is obvious that the rapid increase of the telecommunication sector was not foreseen – even though it was certainly missed by many others, too. The increase of the co-operation with firm N was not comparable to the 25% annual increase of its turnover. As an example, the co-operation concerning the further development of the firm’s in-house code generator was never realised. The fall of the consumer electronics segment had been rather dramatic as early as between 1986 and 1992, as much as 14%. Yet, the firm S, for example, was still considered an important industrial customer for code generation related projects by TKO.

One of the strategic visions followed by TKO at the turn of the nineties was that computers were rapidly becoming embedded in various kinds of machines and industrial production systems. The vision had been presented already in the eighties by Matti Otala, among others. Yet, according to [Klus et al. 1985] the change of the importance of electronics and real-time production control technologies in the metal industry, as seen by executives, had been estimated to remain less than 3 on the scale 1 - 4.

On the contrary, the importance of software technologies in the electronics industry had been estimated to increase from 3 to 3,5 by both executives and engineers. This kind of an evolution had indeed started already by 1989. In contrast, the annual R&D subcontracting value of the machine and equipment industry was estimated to be only one fifth of the corresponding value of the electronics industry in 1989. Still, TKO had increased the volume of its machine and equipment manufacturing customers to as high as 40% of its industrial customer base by 1992.

There was also a clear difference in public funding for electronics and information technology research, compared with machine automation [Anon. 1990b]. Tekes had spent 70 million marks and other organisations 45 million on the Finprit program during the period 1984 - 1988. For Finsoft the corresponding figures were 40 and 15 million marks between 1988 and 1990, when the program was in its busiest phase. The total sum of the two programs was well over 150 million marks. The funding of Tekes for the mechatronics research program between 1987 and 1989 was only 20 million marks, and an additional 10 million marks was received from other sources. This was altogether less than one fifth of the two software-related research programs.

At the end of the eighties VTT had established a Machine automation Laboratory in Tampere, which started to research and develop machine control solutions based in part on electronics and embedded systems. The laboratory had the advantages of being located at the heart of the Finnish machine industry and employing a considerable number of automation and control system experts. The VTT Electronics laboratory had also been carrying out R&D on mechatronics for several years. [Anon. 1987a] includes a piece of news (Insinööriuutiset, 13.4.1987) titled “VTT [Electronics laboratory in] Oulu leads research. Electronics to machine parts according to the conditions of mechatronics”. Mechatronics is mentioned as one of the foci in the Oulu region in [Anon. 1987c], which states that “the Electronics laboratory of VTT located in Oulu carries out research that falls in part in the area of information technology”.

The overall situation in the outer context of the code generation research and development during the birth of Reagenix in 1991 and 1992 was thus that TKO was extensively involved in an industrial segment that was small, had increased quite slowly and was not keen on using external R&D services. Moreover, there were competing offerings for this segment even inside VTT. In comparison, the electronics industry had increased very rapidly (in the Oulu region even faster than elsewhere in Finland [Anon. 1989]), was using external R&D services more extensively and had started to consider embedded software as a core product technology.

In addition, TKO had become largely dependent on external income, not only from the machine industry, but also from Tekes, when a very deep economic recession hit Finland at the turn of the nineties. Reagenix was not only competing against the commercial code generator launched by firm I during the deepest recession, but it was also marketed especially to the machine and equipment industry and to consumer electronics and instrumentation firms which suffered considerably from the recession.

[Anon. 1993b] points out that “many indicators show that the technological development of industry [in Finland] seems to have become slower. Investments in research have increased less than earlier. The increase of exports rested almost entirely on a few large companies. The role of new innovative small enterprises has been too small”. The telecommunication sector was estimated to “consists of a functional whole which has yet only weak cluster structures”, but the sector was “strongly oriented towards exports and has increased rapidly also during the recession”. In the middle of the recession in 1991 the value of the production of the electronics industry was 12 billion marks, of which 24% came from telecommunication, 11% from industrial automation and instrumentation and 6% from consumer electronics. In 1992, the corresponding shares of the sectors were 29%, 9% and 4%. By 1997, the value of the production had increased to 65 billion marks, of which almost one half came from the telecommunication sector.

The technology review 1993 of Tekes estimates that the foci of development in the automation sector include “hardware and software engineering”, but that a weakness is “the lack of companies that develop production automation systems” [Anon. 1993a], such as firm Nm. Concerning telecommunications, it is estimated that “the great importance and hard competition in the field mean that Finland cannot be a forerunner in critical issues. Big competitors can take in use standards that differ from the ones used by the Finnish companies.” This was a completely wrong vision, of course. According to the competence strategy of the electronics and electrical engineering industries written only one year later in 1994, the telecommunication sector had “reached the World Class level … and become involved in the creation of new standards” [Anon 1994b]. The same strategy indicates that the development of automation systems “is based strongly on the familiarity with the sector[s] using the systems”. Moreover, the strategy claims that “system design tools have become a central part of product design, because the efficiency and competitiveness of products depend critically on … the tools”.

The Tekes technology review written three years later states that “companies must be involved in affecting and specifying new standards for the field” [Anon. 1996]. Moreover, the “most important technical competence areas” of the telecommunication sector are explained to be “embedded real-time software, signal processing, ASIC design, radio engineering, telecommunication systems and system-level software”. The review includes for the first time a special section on embedded software (pp. 31 - 32). Between 1994 and 1997, 24 000 new jobs were created in the electronics and electrical engineering industries, in 1997 44% percent of all new employees held a Master’s degree in engineering and 33% an engineering college degree. During 1989 - 1993 the labour of the electronics and electrical engineering industries decreased to 34 400 persons, but already in 1996 it had risen again to the level of 51 000 persons [Hienonen 1997]. That year the value of the production of the electronics and electrical engineering industries was 54 billion marks, of which the share of the electronics industry was 41 billion marks. The share of the telecommunication sector was 46%, the automation sector 7%, and the consumer electronics only 2%.

The increase of production in the telecommunication sector had been over 30% in 1996, in industrial automation and instrumentation it was about 15%. In consumer electronics the production had decreased by 20%. About 31% of the personnel of the electronics industry was involved in R&D in 1996, but in the telecommunication sector as much as 47%. In 1987 the corresponding values were 19% and 28%. In industrial automation and instrumentation, the volume of R&D personnel was 15% in 1987 and 16% in 1996, in consumer electronics the volumes were 7% and 14%.

[Hienonen 1997] includes an estimate of the volume of personnel in electronics industry involved in software development. It shows that the telecommunication sector employed over 30 times more software engineers in 1996 than the second biggest sector, industrial automation and instrumentation. In 1997 the value of the R&D spending in the electronics industry was almost 17 billion marks, 61% of the total national R&D spending [Anon. 1998]. “Software plays[ed] a key role” in the electronics industry, and almost one fifth of the personnel of electronics and electrical engineering firms was carrying out software development – in telecommunication firms as much as one fourth.

VTT was restructured in 1992, at least in part due to the recession. The former small independent laboratories were integrated into nine large units. The mechatronics section of the Electronics laboratory was incorporated with the Machine automation laboratory and some other laboratories with VTT Automation. The Computer technology laboratory became a part of VTT Electronics, together with the rest of the former Electronics laboratory and two other laboratories. The hardware engineering related research groups of the new unit ELE started to specialise in telecommunication electronics. The software engineering groups had a wider technology portfolio and customer base, but were also deeply involved in the R&D of embedded telecommunication software and the methods and tools used in software development. By 1998, the annual volume of embedded software R&D at ELE had risen to over 70 man-years, of which almost 40% involved confidential customer projects. 50 to 70% of these projects were carried out for telecommunication firms. The annual size of the biggest project portfolio contracted by a single customer was about 15 man-years. It included several activities concerning software method and tool development as well as technology transfer.

The machine and equipment industry had become one of the key sectors for fault diagnosis and intelligent systems R&D at ELE [Seppänen et al. 1998a], but not for R&D on embedded software engineering. Only occasional customers had been gained from the consumer electronics sector. Comparing the dramatic increase of the telecommunication sector between 1993 and 1997, also in terms of the number of software engineers, with the evolution of the other sectors, it is very likely that the strategic market of Reagenix would have been there. The right window for stepping into the rapidly moving telecommunication sector might have been offered as early as in 1991, or during 1992 - 1993 at latest. However, at that time TKO was busily developing the code generation technologies further for the needs of machine and equipment manufacturers (e.g. firm Nm), as well as for the consumer electronics sector (e.g. the firm S).

4.1.2 Technological developments

Tables 4 and 5 taken from [Seppänen et al. 1996] present the technological developments in embedded software engineering in the nineties, from the viewpoints of system solutions and design methods and tools. The road maps shown in the tables are based on interviews with approx. 150 professionals around the world, the initial purpose being to support the planning of embedded software related research and development activities in an electronics research program launched by Tekes in the late nineties.

Table 4. Overview of an embedded software solutions road map.

Area of the road map	**1991**	**1996**	**2001**
System characteristics	Device control	Mass customisation, portable products	Networked systems
User interfaces	Alpha-numeric	Graphical and Customisable	Virtual, COTS
System software and hardware	In-house software and COTS hardware	PC-compatible software and hardware vs. ASIC-based co-designs	Open system platforms vs. COTS co-designs
Data access, management	Dedicated local data storage	Data management Systems	Real-time networked multimedia
Associated IT systems	Local information processing	Client-server system interconnections	Embedded and IT system networks
Communication infrastructure	Closed, local area	Open local area and Global networks	Hetero-geneous, Wireless

Table 4 indicates that embedded software solutions have evolved in the nineties from small, dedicated device control programs to large, heterogeneous and networked software systems. At the same time, many special in-house solutions have been replaced by commercial or at least standardised solutions. Interfaces that did not exist or were "closed" for internal use only, have become publicly defined and "open". Embedded software solutions have been integrated with new hardware solutions, such as ASICs, to form deeply embedded systems.

From the business perspective, these developments means that the earlier focus on special-purpose technological solutions has been directed to applications of electronic products and services provided by these. Many companies use standard software-hardware platforms and concentrate on delivering customer value for certain applications. Others still compete by using special technological solutions, if the markets are large enough or if there is either a need for protecting company-specific technological innovations or developing one-of-a-kind systems, such as space devices.

Considering the system solutions developed in the Sokrates project, such as a communication protocol package and an operating system kernel, the developments would have required focusing either on special solutions or accommodating to some standards. The use of the operating system kernel solution by firm N (cf. Appendix 2) and the communication protocol package in a space instrument represent the former. No serious attempts were made regarding the latter. An example could have been provided by developing a software bus concept starting from the protocol package, but aiming at a solution that would be compatible with an object-oriented communication standard.

Table 5. Overview of an embedded software process road map.

Area of the road map	**1991**	**1996**	**2001**
Development methods	Structured development, CMM/1	Early object-oriented development, CMM/2	Component-based development, CMM/3 - 4
Development tools	Structured CASE tools	Object-oriented CASE tools	Application-specific and simulation CASE tools
Environments	Isolated environments	Site-specific environments	Interoperable local & global environments

The evolution of embedded software development methods and tools has led from structured techniques, such as SA/SD, first to object-oriented and then to component-based techniques (Table 5). One of the main reasons for this is the increase in the use of standard software and hardware platforms and open system interfaces. Software engineering environments have changed from isolated and site-specific to heterogeneous and inter-operating support systems that sustain a process-based viewpoint to software production. A good example of this is provided by the Capability Maturity Model (CMM), which is an incremental process improvement framework, as opposed to the “quantum leap” envisioned in the Sokrates project. Moreover, the process perspective has not resulted in such an integration of methods and tools as was forecasted in Sokrates. Instead, the model was built mostly by software quality professionals, who were more interested in the management of software development than in specific methods and CASE tools.

The evolution of embedded software engineering technologies shown in the two tables has been extended and reformulated in Table 6, by using the elements of software engineering competence. The periods of art, craft, standstill and engineering shown in the table illustrate changes in embedded software development practices in Finland. In the mid-eighties, this field involved few hundred hardware-oriented system engineers. In the late nineties, thousands of industrial professionals are work with embedded software systems, which may involve millions of lines of program code.

Table 6. Elements of embedded software engineering during 1985 - 1998.

Period	**Applications**	**Techniques**	**Technologies**	**Functions**
Art (- 1985)	++ Control	*Program design* (SA/SD)	*Compilers Debuggers*	*Coding, Testing*
Craft (1986 - 1991)	+ Control + Automation + Telecom +/- Instruments +/- EDA	*SA/SD* Program design (Co-design) (OO design) (Testing)	*Programming environments* (CASE tools)	*System design,* System testing, Docu-mentation
Stand-still (1992 - 1993)	+/- Control + Automation + Telecom +/- Instruments +/- EDA	-"-	*CASE tools* Visual programming environments	-"- (Quality engineering)
Engi-neering (1994 - 1998)	- Control + Automation ++ Telecom +/- Instruments +/- EDA +/- Multimedia	*OO design* SA design? Co-design (Simulation) (SDL) (TTCN)	Visual CASE OO CASE (System design tools) (ASIC tools) (Web tools)	(Simulation) *Multi-paradigm design,* *Concurrent engineering*

The dominating elements of each period, if any, are shown in italics in the table, the competing elements as ordinary text and the emerging substitutive elements in parentheses. The basic device control applications that were prevailing ("++") in the eighties, have given way ("-") to telecommunication applications with regard to importance. Automation applications have grown ("+"), but considerably much less than what was expected in the eighties. The role of embedded software in electronic instruments has remained about the same ("+/-") or decreased as in the case of consumer electronics, which almost collapsed as an industrial sector. There has been no considerable growth in the Finnish CASE tool business either, although firm I, for example, has played a key role in providing embedded system design methods and tools for industry. Many companies have given up the building of in-house methods and tools, with an exception of some large telecommunication firms.

In the view of the table, and as pointed out by many of the interviewees of this research, Sokrates was carried out at the right time. The goal was to transform the craft of program design to a software engineering discipline. However, the economic recession resulted in a technological standstill from which automation industry, the main intended market of Reagenix, was slow to recover. The rapidly increasing telecommunication sector became interested in systematic software design methods and automated tools, but the opportunity window seems to have been open only for a short while.

4.2 ANALYSIS OF THE PROCESS NETS

Six groups of actors were directly involved in the code generation case. The objectives, goals and logic of these actors determined the shape of the code generation process nets. We will, therefore, start the analysis of the nets by analysing how much the logic of action were aligned or differentiated and for which reasons. The code generation project nets are then compared with the project marketing and purchasing horizons, based on data about the planned and actual exploitation of the code generation competence.

4.2.1 Alignment of the logic of action

Table 7 summarises the objectives and goals of the actors involved in the code generation case during the period 1985 - 1998. Four of the six groups of actors are classified further, as follows:

- VTT managers into directors and sections heads (i.e. high-level general managers and low-level technical managers),
- Code generation researchers into the Sokrates project team and the people involved in the development of Reagenix,
- Industrial customers into the Sokrates participants, broad co-operators, focused buyers and tool vendors, and
- Research customers into the focal organisation itself and other research organisations.

One of the main objectives of the VTT managers was to increase the volume of contract R&D, while minimising financial risks and earning only small profits as a national government-owned research organisation. The focal researchers wished to make a paradigm shift in the development of embedded software, by building and transferring in use innovative design methods and tools. In principle, the goals of the two groups should have been well-aligned, actually the same, but they appeared to be quite different.

The strategy of Tekes involved starting and co-ordinating national research initiatives as a funding body, to "investigate the limits of new technologies", as it was described by one of the interviewed Tekes representatives.

Industrial customers were keen on following up and evaluating new technologies, but mainly by making rather moderate investments and taking only limited technical risks. The sooner this kind of technology screening would result in direct business benefits, the better.

Research customers were aiming at certain project goals that were usually not related to code generation. They were looking for free or inexpensive use of effective technologies, which tended to irritate the focal researchers. The research customers and VTT managers did not recognise this, though. The other VTT colleagues, who were mostly informally related to the focal researchers, also had their own professional and personal objectives.

The goals of the actors involved in the code generation case were not static, but changed rather considerably as indicated in Table 7. The table shows the alignment of the goals of actor groups with other groups by "+" and differentiation by "-". A situation where the alignment or differentiation of goals clearly changed is shown by "+/-".

Table 7. Alignment of the goals of actor groups between 1985 and 1998.

Groups	**Actors**	**Alignment of the goals of actor groups**		
		Speco	**Sokrates**	**Reagenix**
A: VTT Managers	Directors Section heads	Serve firm K Nurture R&D projects +/- B + D	Follow research Market new red projects + B, D, E	Don't care Control human resources - B +/- D, E
B: Focal Resear-chers	Sokrates team Reagenix team	Implement technological innovations + A, C, F +/- D	Carry out world-class research in a different way +/- A, F + C, D, E	Create new type of product business at VTT - A +/- C, D, E, (B)
C: Funding Bodies	(KTM) Tekes	Finance the screening of technological limits + A, B, D, E	Finance generic research and aid in the transfer of technologies + A, B, D, E	Finance applied research based on explicit industrial needs +/- A, B, D
D: Industrial Customers	Sokrates parties Broad co-ops Tool vendors Focused buyers	- - - Try new technologies + A, C +/- B	Follow ongoing novel research Gain new ideas Consider using the technology Try new technologies + A, B, C, E	- Buy solutions Be alerted Buy certain new technologies +/- A, B
E: Research Customers	TKO/ELE Other institutes	- -	Make use of research results + A + B, C, D	Solve design-related problems cheaper or faster +/- A + B, C, (D)
F: Colleagues	VTT research scientists	Discuss about technologies + A, B, C, E	Argue about goals and results +/- A, B, (E)	Use in projects, if not too risky + A, B, C, (D)

The background of VTT managers can be characterised as collective R&D, as opposed to focused individual specialists. One of their main interests was to increase the volume of R&D in the network position of organisational decision makers. They focused on launching and nurturing projects. As an example, they took much the side of the key customer, firm K, in the dispute over the Speco project. Yet, only after project-based exploitation of the Sokrates results had in their view failed, their goals started to differentiate considerably from the goals of the focal researchers. They were not any more actively marketing red code generation projects in the late nineties, although they did not oppose the use of Reagenix in blue projects either.

The focal researchers, whose professional background involved rather small-scale R&D, were interested in making innovations in the position of high-level technology experts, not as project or organisational managers. Technological innovations, as understood by the focal researchers, were in the Sokrates project produced by conducting world-class research in a different way from all the other projects of TKO. This different approach caused problems with the VTT managers and colleagues. During the Reagenix period, a related novel aim was to create product-based business at VTT. The goals of the VTT managers started to differentiate quite remarkably from the goals of the focal researchers by the mid-nineties. The group rehearsal and interviews revealed differences also in the goals of the focal researchers themselves in this regard.

Tekes, the funding body, was acting as a kind of patron. Its interest was to ensure national success in developing and exploiting new technologies. However, it redirected its funding in the nineties from generic research, such as the Finsoft program, to projects with much more direct links to certain industrial needs. It did not provide any direct funding for code generation related work after 1994.

The background of many of the industrial customers involved in the code generation case revealed expertise in certain industrial applications rather than in embedded software engineering. They were interested, like firm K already in the late eighties, in early screening and evaluation of new technologies, mostly in the position of product development managers and engineers. After the Sokrates project, focused buyers acquired certain technologies from VTT, whereas broad co-operators were looking for comprehensive system design solutions. On the basis of our interviews, these goals were not, after all, considered properly either by the VTT managers or the focal researchers. Firm Nm is a good example of this, but also the lack of co-operation with firm N, which would have been one of the most obvious customers for code generation related R&D. No co-operation emerged with tool vendor companies either - the business goals of firm I clashed with the goals of the focal researchers during the Reagenix period.

Research customers were interested in achieving certain project goals, often in the position of project managers. They wanted to make use of Sokrates and Reagenix results to solve various kinds of problems, for the benefit of the industrial partners of their projects. Some of them criticised the VTT managers for not attending to Reagenix well enough, but all of them were quite satisfied with the support received from the focal researchers. They did not recognise any problem regarding the free support. The VTT colleagues, who did not utilise the results, were acting as observers rather than active participants even during Sokrates and especially later.

The logic of action of a group of actors is a combination of not only the goals and means of the group. It includes also the justifications and arguments supporting the viewpoints of the group, based on the backgrounds, interests and positions of the actors. Table 8 summarises the logic of code generation actions based on the goals, means and justifications of the actors.

Table 8. Logic of code generation actions during 1985 - 1998.

Groups of actors	**Objectives and goals**	**Means**	**Justifications**	**Logic of action**
A: VTT managers – research directors – section heads	Increase of the volume of contract R&D: minimise risks, earn small profit	Control of human and organisational resources	Collective R&D: Increase volume as decision makers	*Contractual R&D* – project business – resource control
B: Focal researchers – Sokrates project team – Reagenix team	Make a paradigm shift: transfer in use new design methods and tools	Control of project resources, personal technical skills, team-based problem solving	Small-scale R&D: Innovation as technology experts	*Small entrepreneurship* – pioneering research – product business
C: Tekes	Guide national research initiatives: investigate limits of technologies	Control of financial resources, program-level co-ordination	National success: Act as a "patron"	*Investments in the future*
D: Industrial customers – Sokrates parties – broad co-operators – focused buyers – tool vendors	Follow-up and evaluate new technologies: make moderate investments with small risks for the benefit of the business	Freedom to accept or reject the developed technology, business and application skills	Application focus: Early technology screening/adoption as development managers or experts	*Technology testing* – smooth audience – paradigm buying – solution buying – technology watch
E: Research customers – within TKO/ELE – other organisations	Aim at certain project goals: utilise effective technologies for free	Control of project resources, use of project relationships	Project-based R&D: Project goals as project managers	*Problem solving* – cheaper solutions – faster solutions
F: Colleagues	Ensure professional and personal careers: carry out applied engineering R&D	Personal skills, project relationships, (money, time)	Applied research: Personal goals as "observers"	*Professional interest*

The logic of action of the VTT managers has been described in Table 8 as "Contractual R&D". It meant project business based on the control of human and organisational resources. The focal researchers emphasised "Small entrepreneurship", in the form of pioneering research during the Speco and Sokrates periods and product-based R&D business during Reagenix. Tekes wished to make "Investments in the future", whereas the logic of action of the industrial customers can be characterised as "Technology testing". The Sokrates steering group was a rather smooth audience for innovative research. Broad industrial co-operators were looking for opportunities for exploiting comprehensive system development paradigms, whereas focused buyers were interested in specific technological solutions. Tool vendors were conducting technology watching by following the work during and immediately after Sokrates, while not actually participating in the work. The logic of action of the research customers was geared towards "Problem solving", by the use of inexpensive or faster solutions. VTT colleagues expressed "professional interest" in the work, but were not directly involved in it.

The driving force behind the logic of action of the VTT managers was the goal to increase the volume of R&D. As an example, TKO grew from about twenty to almost a hundred people in ten years, during the period 1983 - 1993. The close to non-profit nature of the services provided by VTT meant, according to the logic of the managers, that green and blue research topics should move to the red portion of the project portfolio after quite a short while. In other words, industrial customers should have covered the costs of the further refinement of capabilities built in green and blue projects. Profits from red projects have been allowed at VTT only since the mid-nineties. For example, the main red code generation project MCS-REA was offered to firm Nm at a prime cost, with no profit margin. Later during the nineties, the profits gained from Reagenix licenses appeared to be very small, no considerable red code generation projects were carried out and the tool was used for free in joint blue projects. The project business logic of the VTT managers was therefore actually most effective during the Sokrates period, when TKO was earning the highest external income from code generation R&D. The focal researchers should have, in principle, followed the same logic of action as project managers and researchers did. However, this was not the case. The early dispute with firm K during the Speco project had a lot in common with the disagreement with the VTT managers later in the nineties. The case data shows how the views of the two parties differentiated. The manager of firm K wanted to have well-established schedules and plans, while the innovators were solving problems at a hectic pace in the basement.

In the interviews, the focal researchers and some of their colleagues were criticising the unwillingness of the VTT managers to take risks, be they financial or technical. The managers of firm Nm described the consequences of serious risks that had been realised. The focal researchers still had the original belief that taking the risks of technology testing had paid off for firm Nm. The managers of firm Nm were, on the other hand, characterised as smart people in this regard. The VTT managers said that they rather soon lost confidence in the judgement of the focal researchers to realise code generation related business plans.

In Table 8, the logic of action of the focal researchers, resulting from their vision of changing the craft state of embedded software development to an engineering discipline, is characterised as small entrepreneurship. The background information of some of the focal researchers included in Appendix 2 shows the roots of this logic. The researchers were stressing in the interviews that some of them had "run real companies" before joining VTT. However, even the complaints of the Sokrates steering group regarding the preparation and delivery of plans, status reports and documents point towards the logic of small-scale business: paperwork was done after the more interesting technical questions had been answered.

Still, this logic of action was quite appropriate during Sokrates. According to Tekes, the Finsoft evaluators and also the steering group, the project succeeded in reaching its goals. The disagreements between the focal researchers and some of their colleagues regarding the appropriate way of carrying out innovative research were not considered any serious problem by the interviewed VTT managers. The logic run into difficulties only afterwards, during the Reagenix period. The focal researchers were then left with practically no project resources at all at their disposal, while the managers who controlled the human and organisational resources continued to follow their own logic. Yet, the managers did not sell out Reagenix either, until 1998. It can be questioned, why this was not done much earlier. One possible answer is that Reagenix was used in quite a number of blue projects during the whole of the nineties. A more likely reason is, however, the project-based business logic of the managers. They were neither keen on selling licenses, nor of making money by selling the IPRs of VTT for good.

The goals and logic of action of Tekes, industrial customers and research customers were quite well aligned with each other during the Sokrates period, when the logic of the focal researchers was prevailing. The logic of Tekes, VTT managers and the focal researchers did not conflict even after Sokrates until about 1994, because they were all wishing to make use of the results of Sokrates. However, due to the industrial logic of testing new technologies, as well as due to the recession, only a few and rather small exploitation projects arose. After firm I had introduced its commercial code generator, many of the companies that were interested in code generation bought the tool from firm I.

This can be understood as part of the emerging engineering period in the development of embedded software, as shown in Table 6. During the earlier Craft and Art periods industry was more willing to follow and test research results. The whole Finsoft program is a good example of this willingness. Industry, Tekes and the VTT managers were looking for more needs-driven software engineering research after the Standstill period 1992 - 1993. Such new topics as animation and unit testing used to renew the several years old code generation vision could not compete against the new visions of object-oriented and component-based software engineering. Code generation became an in-house means for VTT of solving other research problems in blue projects. The interest of these projects was not to carry out or pay for any code generation research. Yet, the logic of action of the focal researchers seems to have become captured with such an idea.

4.2.2 Planned and realised process nets

The first step towards analysing the content of the logic of action is to compare the project marketing and purchasing horizons with the realised project nets, Table 9. The project nets are then presented in more detail and the form of interaction between the different groups of actors is analysed. During the Scope period, embedded software engineering was still considered art, but it started to emerge as a distinct profession, in which more effective tools and methods were needed. Tekes had been established in 1983 and TKO was in many respects in a pivotal position for launching, carrying out and co-ordinating national research on embedded software engineering. For this reason, the focus of the code generation project marketing horizon of TKO involved joint blue projects. This was also the main project purchasing horizon for industry and Tekes. Yet, the ongoing phase of the code generation R&D process was innovation rather than technology development. Correspondingly, the ongoing phase of the competence exploitation process was technology screening.

The actual colour of the Draco project net was green, although KTM was involved as an external funding body. However, the Speco project net was red, involving both technology evaluation and early technology development. There was thus a mismatch not only between the project marketing and purchasing horizons and the realised process nets, but also between the phases of the R&D and competence exploitation processes. The problems encountered in the Speco project can be seen as one result of these mismatches: the researchers aimed at innovations, the managers expected to get value for the money they had spent (firm K) or earned (TKO). VTT had not yet expressed any considerable interest in internal purchasing of embedded software engineering technologies. No code generation project nets had been established within VTT.

During the Sokrates period embedded software engineering evolved from art to craft, but the period ended with a standstill caused by the economic recession. Industry was eager to take new methods and tools in use in product development, but the recession would decrease its interest towards the end of the period. The survival, or rather the miraculous growth, of the telecommunication sector was not yet foreseen. The ongoing phase of the R&D process was technology development. The project marketing horizon of the VTT managers included both red spin-off projects of Sokrates and blue projects, in which code generation could have been applied. The focal researchers were also looking for opportunities for selling tool licenses.

The competence exploitation phase related to the project purchasing horizon was technology evaluation: Tekes was expecting to see industrial spin-off projects, industry was also interested in the base techniques of Sokrates, such as SA/SD. The VTT Electronics laboratory was ready to exploit the early Sokrates results first in blue and then in red projects. The actual project nets fitted well with the marketing and purchasing horizons, although they involved not only technology evaluation but also some technology screening. For example, the choice of an Ada style programming language for Sokrates-SA was a result of such screening.

Table 9. Planned and realised process nets during 1985 - 1998.

Code generation vs. industrial software engineering periods	**Project marketing horizon (vs. phase)**	**Project purchasing horizon (vs. phase)**	**Realised project nets**
Speco/Art…craft: Embedded software engineering started to emerge as a distinct profession, tools and methods were needed	***Innovation*** Blue embedded systems projects: joint software engineering research	***Screening*** *Tekes*: blue projects *Industry:* blue projects *VTT*: --	***Evaluation*** Speco: red research Draco: pseudo-blue (green) research
*Sokrates/*Craft…standstill: New software engineering methods and tools had been developed and taken in use, but the recession decreased industrial interests	***Development*** *Managers*: red spin-off or blue research projects *Researchers*: red spin- off projects, red licenses (blue and green projects)	***Evaluation*** *Tekes*: spin-off projects *Industry*: tool piloting and system design projects, SA/SD courses *VTT*: blue/red projects	***Screening and evaluation*** Sokrates: blue research Kaapeli: red spin-off Synchro, Sasic, Sixa: blue research
*Reagenix/*Standstill…engineering: Industry was recovering from the recession led by the telecommunication sector, but its interest in in-house general-purpose methods and tools had decreased; new methods and tools were sought for managing large systems; the software process emerged as an integrative view on software development	***Leverage*** *Managers*: red projects (incl. VTT) especially in machine automation, blue projects (incl. joint European projects) ***"Commercialisation"*** *Researchers*: internal and external licenses, green tool development projects, red generator-based software projects	***Process and product development*** *Industry*: well-packaged effective methods/tools *Tool vendors*: buying of IPRs, elimination of likely competitors *VTT*: well-packaged and supported in-house (cheap/effective) tools *Research institutes*: problem solving means	***Process and product development*** Reagenix, Aniprosa: green research (Table 10): generator-based red projects, license selling (Table 10): usage in blue research

Sokrates, Synchro, Sasic and Sixa created together a considerable portfolio of blue projects including two VTT laboratories, Tekes and a large number of companies. Kaapeli was the kind of a red spin-off project that VTT and Tekes wished to launch already when Sokrates was ongoing. After the Sokrates project had been finished and the Reagenix period started, the standstill hit in 1992 - 1993 and changed only gradually to a true embedded software engineering era. The Finnish industry was recovering from the recession led by the telecommunication sector. The traditional consumer electronics sector had almost disappeared and the importance of the automation industry had decreased in comparison with the electronics industry. Industrial interest in developing in-house methods and tools had faded, because international tool vendors were well represented in Finland, not to speak of firm I and some other national EDA companies. The telecommunication sector needed new methods and tools for managing large software systems, but were also looking mostly for commercial solutions. The software process had emerged as an organisational rather than a tool-based view on software development.

The phase of the code generation R&D process expected by the VTT managers and Tekes was leverage. The focal researchers accompanied this with a "commercialisation" phase to be carried out by VTT itself, after the attempted co-operation with firm I had failed. The project marketing horizon of the VTT managers included red projects, targeted especially at machine automation. VTT was considered an internal customer. The managers were also aiming at establishing blue projects, so as to apply and develop further code generation techniques, including joint European research projects. The focal researchers focused on selling Reagenix licenses, to conduct the further development of the generator.

The code generation competence was in the phase where it could be exploited in process and product development. Industry was looking for well-packaged and effective embedded software engineering methods and tools, as did also internal research customers at VTT. Commercial tool vendors were interested in preserving and extending their markets, including buying of IPRs and elimination of competitors. The realised code generation project nets, as shown in Table 10, included small-scale green development, i.e. the use of the Reagenix account and the Aniprosa project. A few red code generation projects were launched. The main project, called MCS-REA, was carried out for firm Nm. Some licenses were also sold.

Table 10 summarises the realised project nets, from the viewpoint of different actors and colours of their relationships. Blue project nets dominate, green nets are almost missing and rather few red nets were created from the many blue nets. VTT Electronics laboratory established the kinds of red machine automation related projects nets at which TKO aimed in the early nineties, but Reagenix did not became any strategic technology for them. The recession finished the broad co-operation between TKO and firm Nm. The relationship with firm K that had expanded from the red Speco project to the joint blue Sokrates project broke, too. There was hardly any continuation in project relationships with focused buyers either.

Table 10. Code generation project nets during 1985 - 1998.

Actors	Relationships			Remarks
	Green	Blue	Red	
TKO + KTM	Draco: 1987			Pseudo-blue
TKO + firm K		Sokrates: 1988 - 91	Speco, Speco-2: 1985 - 87	Red-to-blue (expansion)
TKO + firm N		-"-	Osdyn: 1992, (Mots-2: 1997)	Blue-to-red (specialisation, focused buyer)
TKO + firm E, W, So, P, Nt, Pa, Ta, Th, T, V		-"-		Finished, except with the firms T and V
TKO + firm I		-"-		Relationship as a tool vendor 1992-1998
TKO + firms	Reagenix: 1992 - 1997		Reagenix: 1992 - 1997	License selling and internal tool development
TKO + firm El	Aniprosa, Aniprosa-2: 1993 - 1994			Subcontracting from firm El by VTT
TKO + firm Nm		-"-	Kaapeli: 1990 -91, MCS-REA: 1993	Blue-to-red (specialisation, broad co-op)
TKO/ELE + VTT Electr. laboratory + firms + Oulu university		Synchro, Sasic: 1989 - 90, Tulko: 1989 - 93	(several: 1991 -)	VTT Electronics laboratory: broad internal co-operator
TKO + firm R			Raski: 1992	Broad co-operator
TKO + firm S		Sokrates: 1988 - 91	Sympa: 1992 - 93 Cute: 1993	Blue-to-red (specialisation, focused buyer)
TKO + VTT/ ELI			Kaasu: 1992	Focused internal buyer
ELE + firm Kc			Nosto, Nosto-2: 1993-96	Focused buyer
TKO + Työ-suojelurahasto + firm T		Turva: 1991 - 92, Diag: 1992 - 93		Sokrates used as a design method (blue/no colour)
TKO + Työ-suojelurahasto + firm H		Rulla, Rulla-2: 1993 - 1996		Reagenix used as a CASE tool (blue)
ELE + Oulu university		Rekki: 1995 - 96		-"- (blue/no colour)
ELE + STUK + VTT/AUT		AVV: 1995 - 1997		-"- (blue/no colour)
TKO + Tekes		Sokrates: 1988 - 91, Sixa: 1990		Funding also Kaapeli and MCS-REA

Table 10 shows that during the period 1985 - 1998 there were altogether three green project nets, one blue project, in which code generation competence was developed, and nine others, in which it was exploited, and nine red project nets managed by the focal organisation. In three cases, the blue Sokrates project net that consisted of more than ten firms was specialised in red customer project nets. Firms N and S in these nets were focused buyers and firm Nm a broad co-operator. Not only code generation related co-operation but also other co-operation with most of the other Sokrates parties finished entirely. Firm I remained one of the commercial tool vendors of VTT. None of the blue projects in which code generation competence was exploited resulted in red code generation projects of the focal organisation. However, Reagenix licenses were sold to firms that participated in the blue projects.

Firms R and Nc and VTT Food technology laboratory represent customers with whom red project nets were established without any previous participation by them in blue code generation projects. The relationships with them involved single projects, none of them have contracted any other projects afterwards. Firm El did not become any strategic partner of VTT in the further development of Reagenix either. From the viewpoint of customer relationships this meant the following:

- one red relationship with a key customer (firm K, machine automation) was expanded to a blue network,
- one blue network with a key customer (firm N, telecommunication) was specialised in a red relationship,
- two blue networks with new customers (firm S, consumer electronics; firm Nm, machine automation) were specialised in red relationships,
- eight out of the eleven other customer relationships of the blue Sokrates network finished (firms Nt, V and T were former customers, the rest were new customers),
- the blue network with firm I changed to a series of tool purchasing transactions (from firm I to VTT),
- three new red customer relationships were created (firm R, consumer electronics; firm Kc, machine automation; VTT/ELI; research institute/scientific instruments),
- nine blue (in part colourless from the viewpoint of the focal researchers) code generation exploitation projects were created, involving three different funding bodies, several research partners and a large number of industrial firms,
- one green relationship was created (firm El), and
- a number of red license selling transactions took place.

The longest lasting customer relationships, six years, involved the original industrial research partner firm K and the key industrial exploiter firm Nm. The former lasted from 1985 to 1991, the latter from 1988 to 1993. Figure 8 shows how the number of green, blue and red project nets changed during 1985 - 1998, without considering the volume and parties of the nets.

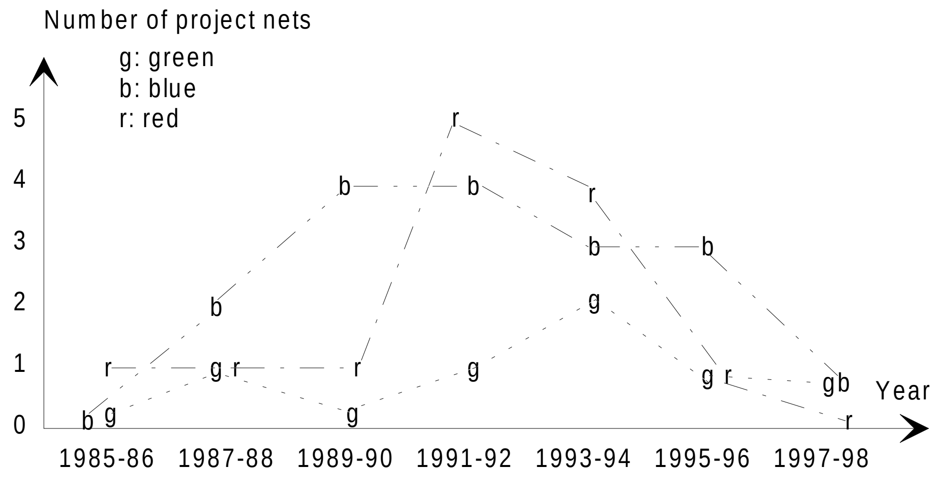

Figure 8. Change of the number of project nets during 1985 - 1998.

The four years from 1991 to 1994, after the Sokrates project, were the time of the most extensive networking. The figure shows that at that time the relative number of green, blue and red projects nets followed the ideal project portfolio. However, most of the blue projects were exploiting already existing results, not solving new code generation problems. As the focal researchers told in the interviews, this did not bring too much external income for them to carry out further research and development on code generation. Quite the opposite, it often meant extra work with no income.

If the blue exploitation projects were removed from the figure, actually only the Sokrates project would remain. In other words, there was no continuation at all in the joint code generation research, which was a rather typical situation in more general terms: public funding for a specific applied research topic may last for two to three years, but after that the topic should be considerably revised or integrated into other topics to ensure continuing funding. This did not succeed for Sokrates. It was attempted, but only once, after a few weeks of the massive Sokrates project had been finished, as a very small part of another research topic, and in the middle of the recession.

The number of red code generation project nets increased rapidly at the end of the Sokrates project, despite the standstill caused by the recession. However, their number also decreased rapidly. In about five years it was back to the level of one project from which it started in 1985. The number of green project nets was very low, when compared with the blue and red project nets. Yet, the figure provides a clear illustration of the fact that before Sokrates started in 1988, the number of green projects increased according to the ideal project portfolio principle.

The development of Reagenix after Sokrates was, on the other hand, carried out mostly in red projects. The number of green projects increased again in 1992 - 1994, when the number of red and blue projects had started to fall, but this could not stop the falling. The blue project nets decreased more slowly than the number of red project nets, but they also ended up at the level of one project in 1997 - 1998.

4.2.2.1 Forms of interaction

The forms of interaction between the focal researchers and the other groups of actors are illustrated in Table 11. The processes owned and the resources controlled by the actors are also shown, so as to associate the discussion with the contents of logic of action addressed in the next section. The most influential groups are shown in *italics* in the table, rather influential as ordinary text and less influential in (parentheses). Non-influential groups of actors are also listed and reasons for the lack of their influence outlined.

Table 11. Interaction of code generation parties in 1985 - 1998.

Period	**Influence of actor groups**	**Forms of interaction**	**Core processes**	**Key resources**
Speco (1985 - 1987)	*Researchers*	Co-operation	R&D (innovation)	Individual and team-based problem-solving skills, vision
	VTT managers	Co-operation	Project marketing	Contacts with Tekes/industry, project planning skills
	Firm K	Organisational dominance	Control, co-ordination, R&D	Money, decision making power, technical skills, company vision
Non-influential: KTM		**Reasons**: KTM did not want to control technology evaluation in Draco, but just finance the project.		
Sokrates (1988 - 1991)	*Researchers*	Dominance	R&D (dev.)	Sokrates project resources, vision
	(Tekes)	Submission	Evaluation, Co-ordination	Money, program-level resources
	(Industrial customers)	Co-operation	Project purchasing, Exploitation	Money, system design and application skills
Non-influential: VTT managers, Research customers, Sokrates parties, Colleagues		**Reasons**: VTT managers, research customers and Sokrates parties did not control the R&D process, only co-ordinated it. Colleagues could not even co-ordinate the process, only comment its results.		
Reagenix (1992 - 1998)	*Research customers*	Co-operation	R&D, inter-action with industry	Project resources, problem-solving skills
	VTT managers	Organisation al dominance	Project marketing	Organisational resources
	(Researchers)	Technical dominance	R&D (leverage)	Individual skills, Reagenix account
	(Tool vendors)	Competition/ Don't care	Commer-cialisation	Knowledge of EDA markets, business skills
Non-influential: Funding bodies, Industrial customers		**Reasons**: Funding bodies had no control over Reagenix, they just financed projects where it was applied. Industrial customers had, in practice, no control over Reagenix either.		

During the Speco period the most influential group of actors was the focal researchers, because they had the individual and team-based problem-solving skills for carrying out the innovative research needed for realising the code generation vision. Firm K put forward the original vision and had both technical skills and organisational resources for making it true, but more as a vision of the future of one company than as an innovators' dream of changing the whole paradigm of embedded software engineering. By the time of the Sokrates project proposal, the latter had won. The organisational dominance of firm K, based on its position as a purchaser and decision maker in a red project relationship, had also ended.

The VTT managers and focal researchers co-operated with each other and third parties to launch activities that would make the researchers' vision true. The managers used their contacts with Tekes and industry, as well as their project planning skills, to market the Sokrates project. They showed little interest in taking part in forming the technical code generation vision. As an example, Veikko Seppänen prepared a separate Finsoft project proposal on software reuse with some other researchers, based on his own software engineering vision and experiences from his former activities. This proposal along with the Sokrates proposal were sent to Tekes, and only the latter was accepted. KTM did not influence the other parties during the Speco period similarly to Tekes, although it did finance the Draco project. The reason was that it did not use any organisational means of controlling the work, such as project management groups. The Draco project was co-ordinated by an interest group of one manager and a few researchers.

During the Sokrates period, the focal researchers remained the most influential group of actors, but their form of interaction changed from co-operation to dominance. The reason was that they now had all the project resources needed for carrying out the work that would realise the code generation vision. They controlled tightly all the human, technical and physical resources of the project. As a small but illustrative example, the workstation computer that was purchased by TKO for the project was not connected to the computing network used by the rest of the laboratory. Other persons than the project members could not access the computer.

Tekes submitted to the dominance of the researchers. It carried out co-ordination and evaluation work at the Finsoft program level. As described in Appendix 2, industrial interest was enough for Tekes to be assured of the usefulness of the project. Such industrial firms as Nm co-operated with the focal researchers, because they needed support for exploiting the project results. The needs of firm Nm were taken well care of by TKO, due to the fact that it was necessary for it to show the industrial applicability of the research results. The VTT managers, research customers and the Sokrates steering group parties did not control the R&D process, they only co-ordinated it. Colleagues did not co-ordinate the process, but only commented its results. Therefore, these groups of actors were actually non-influential during the Sokrates period. The main reasons for the perhaps surprising lack of influence of the VTT managers was that they could not control the content of the research and that the researchers were now financially almost independent.

The Reagenix period shows a radical change in the forms of interaction of the groups of actors. The VTT managers dominated during the beginning of the period, thanks to their ability of controlling the project marketing process and organisational resources. The reason for the rapid change of the focal researchers from the most influential group of actors to an almost powerless group was the lack of financially considerable red or blue code generation projects after Sokrates. Although the focal researchers dominated technically due to their R&D skills, the income from the code generation projects after Sokrates did not cover the cost of their salaries. The Reagenix account would have ensured the continuation of some kind of financial independence, but the license income appeared to be very modest. As shown in Appendix 2, the annual expenses of the salaries of the Sokrates project team were close to one and a half million marks during the two last fiscal years of the project. By comparison, the income from the biggest red code generation exploitation project MCS-REA was less than one tenth, a hundred thousand marks.

Because of the dramatic decrease in financial resources, research customers later become the most important group of actors for the further leverage and exploitation of the code generation competence. They were both the biggest and most faithful group of actors during the middle and late nineties with regard to interest in utilising Reagenix. Moreover, industry was involved in their projects as research partners. Since Tekes had changed its funding policy more towards industrial needs, it was common that companies took part in blue research projects as active research partners with concrete problems to solve. Six out of the seven projects listed in Table 10, all but the Turva project, included such industrial partners.

The interaction with tool vendors became competitive after the introduction of Reagenix. However, since the tool failed to become a truly commercial alternative, e.g. for the code generator of firm I, the interaction changed to a 'don't care' situation. The EDA business grew in Finland during the nineties to a recognised sector, where knowledge of the domestic market and software business skills played a central role. The selling of Reagenix licenses did not became any comparable business at all. Although there are no exact figures available, the few licenses that were sold by VTT were most likely only a tiny fraction of the domestic CASE tool market.

During the Reagenix period, funding bodies and industrial customers remained non-influential groups of actors, although they participated both in the red and blue code generation related projects. These actors could not control the work that was carried out to apply and develop Reagenix further. The results of the interview of firm Nm show that even the most important industrial customers were led by the technology experts, i.e. the focal researchers. The firms that bought Reagenix licenses had even less influence, if there was no project involved in connection with which the license was purchased. Although funding bodies financed blue research projects in which Reagenix was applied, they did not have the means of affecting the technology either. Not even the technical experts of these projects had such means, because they were end-users rather than developers of the code generation technology.

4.2.3 Analysis of the change of process nets

The change of the process nets outlined in the previous sections will now be analysed at the level of relationships between groups of actors, using the functions of assuring and balancing of mutuality based on scanning and diffusion by socialisation or combination. The changes of the process nets are viewed from the perspective of the relationships of the focal code generation researchers in Table 12. The key elements of the relationships affected by the changes, be they resource, activity or actor related, are also identified. Increasing of the management effort is marked with "+", decreasing with "-" and the 'don't care' situation with "x".

During the Sokrates period, the managers co-operated with the focal researchers by assuring of mutuality based on diffusion by combination. The purpose was to produce proposals and plans together for launching code generation related projects, by referring to and integrating explicit pieces of knowledge. During the Sokrates period mutuality clearly decreased. The two groups of actors still handled project plans and non-technical reports together, but now the managers occasionally raised concerns together with the steering group about the way that information was made explicit and delivered. During the Reagenix period mutuality decreased further, although some planning and co-ordination of the exploitation of Reagenix was carried out jointly. During this period resource collections became more important than activity structures between the two groups of actors. Especially during Sokrates, joint project planning and management had been the key elements in their relationship. Later, when the number of common activity structures decreased, the groups were related to each other mostly via the collection of technical code generation resources.

Resource constellations and ties were the dominating elements of the relationships between the focal researchers and industrial customers, except perhaps for such red projects as MCS-REA, where the R&D activities of the customer and the researchers were linked tightly together. During the Speco period the researchers were scanning explicit knowledge to understand the product application of firm K, but the balancing of mutuality of the project activities failed and a dispute resulted. During the Sokrates and Reagenix periods mutuality between the two groups increased. It was then based on diffusing explicit code generation research results to industrial use. Yet, for example, the designers of firm K resisted the pilot use of the code generator vigorously.

Relationships with tool vendors as special kinds of industrial customers followed a similar path, except that mutuality of the activities of the vendors did not increase but decreased during the Reagenix period. The relationship between firm I and the researchers became competitive, which was why it was no longer more possible to carry out any joint activities. In more general terms, tool vendors were interested in the results of the research, i.e. in resource ties and constellations. Activity links and webs remained weak: firm I took part in the steering group of Sokrates, and some experimental studies were carried out at TKO based on discussions with another vendor. Yet, the two groups never linked their activities together as tightly as, for example, firm Nm as an industrial customer and the focal researchers in the Kaapeli and MCS-REA projects.

Table 12. Change of relationships with the focal researchers.

Actor groups	**Management of changes**			**Affected Elements**
	Speco	Sokrates	Reagenix	
VTT managers	Assuring of mutuality (+): Diffusion by combination (project proposals)	Balancing of mutuality (+/x): Diffusion by combination (Sokrates plans and results)	Balancing of mutuality (x/-): Diffusion by combination (Reagenix tools and methods)	Resource collections (Sokrates and Reagenix tools and methods), Activity structures: (project tasks)
Industrial customers	Balancing of mutuality (x): Scanning by combination (application understanding)	Balancing of mutuality (x/+): Diffusion by combination (Sokrates plans and results)	Assuring of mutuality (+): Diffusion by combination (Reagenix tools and methods)	Resource ties, constellations (Sokrates and Reagenix tools and methods), Activity links, patterns: (project tasks)
Tool vendors	Assuring of mutuality (+): Scanning by combination (related work)	-"-	Balancing of mutuality (x/-): Diffusion by combination (Reagenix tools)	Resource constellations (Sokrates and Reagenix tools and methods)
Research customers	-	Assuring of mutuality (+): Diffusion by combination (help to use Sokrates)	Assuring of mutuality (+/x): Diffusion by combination (help to use Reagenix)	Resource constellations (Sokrates and Reagenix tools and methods)
Funding bodies	Balancing of mutuality (x): Diffusion by combination (Draco proposal)	Assuring of mutuality (+): Diffusion by combination (Sokrates proposal, plans and results)	Balancing of mutuality (x): Diffusion by combination (Kaapeli and MCS-REA proposals)	Activity patterns (project management tasks), Resource constellations (project resources)
Colleagues	Assuring of mutuality (+): scanning by combination (Draco and Sokrates proposals)	Balancing of mutuality (+/x): diffusion by socialisation (failure to establish the R Group)	Assuring of mutuality (+): diffusion by combination (usage of Reagenix in blue projects)	Speco/Sokrates: organisational structures (co-ordination) Reagenix: resource collections (code generator)
Focal researchers	Assuring of mutuality (+): scanning by combination (Speco and Draco results)	Assuring of mutuality (+): diffusion by socialisation (joint work in the Sokrates project)	Assuring of mutuality (+/x): diffusion by socialisation/ combination (share of Reagenix)	Activity structures (informal and project related) and resource collections (R&D results)

The relationships between the focal researchers and their research customers were also based on resources ties and constellations, because the latter group carried out research activities on other subjects than code generation. The activity links and patterns that were created involved assistance in the use of code generation techniques and tools. The mutuality of such activities was high until the late nineties, when most of the original researchers had left VTT and those who remained had plenty of other things to do. As told by the researchers, they also became tired of the fact that hardly any financial or even other types of rewards were given to them by the assistance. Moreover, their help was not always asked for when they would have been interested in giving it. Earlier, when some of the research customers were the former Sokrates project team members, e.g. in the Tulko and Diag projects, such help was asked for and provided regularly.

During the Speco project the mutuality of activities between KTM as a funding body and the focal researchers was very low. During the Sokrates period it increased considerably, Tekes funded not only Sokrates but also the early exploitation projects. Since activity patterns based on project plans and reports dominated and resource collections involved only project resources, mutuality decreased very rapidly after the Sokrates project. The preparation of the MCS-REA project proposal, which included the acquisition of funding from Tekes to firm Nm, was the last considerable code generation related activity pattern in which both groups were involved.

During the Speco period mutuality was high between the focal researchers and their colleagues. They carried out joint planning and co-ordination activities to scan information on existing approaches and to plan for the Sokrates project. During the Sokrates period there was an attempt to continue increasing mutuality, in the form of an interest group that would have helped diffusion by socialisation. It failed, however, and the interest group was dissolved as an organisational structure. Mutuality increased again during the Reagenix period, but this time related to resource constellations based on the use of the generator in blue projects. The internal organisational structures and activity patterns related to the projects within ELE were weak. The focal researchers were not much involved in the external patterns of activities and webs of actors of these projects either.

The researchers kept their internal mutuality high trough all three periods. During the Sokrates project implicit knowledge was effectively diffused within the project group, which looked like a very homogenous team to the outsiders. Earlier, the explicit results of the Speco and Draco projects had been shared to prepare and launch Sokrates. Formal organisational structures did not play any considerable role, compared with activity structures and resource collections. Most of the shared resources were technical results and certain kinds of problem solving skills. Later, when the Sokrates project team had been dissolved, the diffusion based on explicit resources became important, due to the fact that joint project activities were decreasing considerably. Many of the researchers left VTT and certain differences in viewpoints seem to have emerged: "From my point of view, it was a mistake that it [Reagenix] was taken as the continuation of Sokrates and therefore no one was interested".

4.3 ANALYSIS OF THE CODE GENERATION PROCESSES

Table 13 provides an overview of code generation related processes carried out within the process nets discussed in the previous sections. Activities performed in these processes constitute the implementation of the logic of action of the groups of actors involved in the nets.

Table 13. Overview of code generation related processes.

Processes Owners	**Main types of activities**			**Process nets** (cf. Table 10)
	Speco	Sokrates	Reagenix	
Marketing: VTT managers Purchasing: industrial customers	Initiating of the Speco and Draco projects Planning of the Sokrates project	Marketing and initiating of spin-offs and use of Sokrates Planning of exploitation projects	Marketing and initiating of the use of Reagenix Purchasing of Reagenix licenses and IPRs	Speco: Tekes, KTM, industry Sokrates: VTT, Tekes, industry, tool vendors Reagenix: VTT, educational institutes, industry, IPR buyer company
R&D: focal researchers	Evaluation of Draco, Refine Development/ evaluation of a prototype of a PL/M code generator	Development of the Sokrates method, tools, Lego elevator, system solutions Screening of the results	Development /evaluation of Reagenix, Aniprosa, Cute and ReagOS	Speco, Speco-2, Draco, Sokrates, Reagenix account, Aniprosa, Aniprosa-2
Exploi-tation: research (and industrial) customers	Demonstration of the results of Speco Joining of the Sokrates steering group	Use of the Sokrates results in joint blue research projects and in the red Kaapeli project Use of technical documents Taking part in SA/SD courses	Conducting of red spin-off projects Exchange of licenses Acquisition of a patent Use in blue research projects Taking part in courses	Synchro, Sasic, Tulko, Kaapeli Turva, Diagnostics MCS-REA, VTT Electr. laboratory Osdyn, Raski, Kaasu, Sympa, Cute, Nosto, Nosto-2, Rulla, Rulla-2, Rekki, AVV, Mots-2
Commer-cialisation: focal researchers	--	Writing of Courseware and manuals	Writing of Courseware and manuals	License selling and training relationships
Control and co-ordinat-ion: VTT managers (Tekes)	Control of the Speco and Draco projects	Control of the Sokrates project and the Finsoft program Planning for spin-offs	(Planning for continuing projects) Control of spin-off projects	Project steering and management groups, TKO research council, schaffners, TKO management group, R Group
Other: --	Commenting the work	Evaluation of Finsoft and the Sokrates project	Use of Reagenix in education	Temporary issue-based relationships

In the following, we are discussing the evolution of each of the processes shown in Table 13 during the three periods. We will also outline the combined process portfolio of each period, as an introduction to the subsequent analysis of the processes that created and shaped the code generation competence by the functions of assuring of capability and balancing of particularity of resources.

The focus of the marketing and purchasing processes was on blue and red projects during the Speco and Sokrates periods. The corresponding activities involved the kinds of project marketing meetings and preparation of project proposals and plans illustrated in Appendix 2. To put it simply, the focal researchers and managers were the marketers of the projects and Tekes and industrial parties of the Finsoft research program the purchasers.

The marketing process was owned by the VTT managers, though, because they were controlling the organisational resources needed for planning, carrying out and co-ordinating the process as a whole. For example, a frame of hours was fixed for marketing in the annual plans of the line organisation groups. The hours that were actually spent on marketing were recorded to a certain activity code that was controlled by group managers. As the code generation stories in Appendix 3 and extracts from the case data in Appendix 2 indicate, the focal researchers felt that they had not enough control over marketing activities: "An ordinary researcher is not allowed to talk to industrial managers. The section head on his own makes the decisions on the use of the section's resources". The managers felt, on the other hand, that what the researchers were doing was talking rather than acting to launch new projects.

During the Reagenix period, the focal researchers themselves carried out most of the activities related to Reagenix license selling, without much co-ordination with the managers who prepared the operational and marketing plans. Reagenix was actively marketed by the VTT managers from about 1992 to 1994, especially when one of the focal researchers deputised Veikko Seppänen as a section head. Later in the nineties, almost the only considerable code generation marketing activity of the managers was to sell the Reagenix IPR rights to the company established by some of the former researchers. Some inquiries had been made earlier by other parties concerning the rights, but the researchers had turned them down.

Marketing to research customers was mentioned in organisational plans, but using the case data, it was impossible to recognise any planned marketing activities that would have been directed to research customers. Moreover, some customerships were created through former Sokrates project team members working in new projects. Although research customers, after all, became the most influential group of actors in the nineties, the true owners of the competence purchasing process in the business sense were the industrial customers. Since research customers did not provide too much income for the focal organisation, the VTT managers who owned the marketing process did not actively promote the use of Reagenix in blue research projects. The same holds for Reagenix licenses given to educational institutes in connection with courses that the researchers gave on the design method. This was not viewed as business by the managers.

The focal researchers were the owners of the R&D process throughout the three periods, because the skills to carry out research on code generation were neither diffused to customers nor to VTT colleagues. Rather than a linear sequence of activities from innovation to commercialisation, this process involved several intertwined research and development cycles. As exploiters of the results produced by the focal researchers, customers performed technology screening, evaluation and utilisation, but did not build much code generation competence of their own. Some customers, such as firm N that exploited one of the Sokrates system solutions, could have done that. Many others, such as firm Nm, did not have the necessary capabilities, although that was expected by the researchers.

The rather loose integration of the R&D process with the customers' competence exploitation process can be seen as part of the relationship strategy of the focal researchers: make it fast and cheap and deliver value, not work to customers. One of the results of this strategy was that there were only few, if any, relationships in which external parties would have been competent enough to develop code generation skills themselves. The focal researchers themselves stated that their purpose was to transfer the skills "of using technology" (Appendix 2) to customers, not the skills of developing the technology. This is a considerable difference between the code generation case and the fault diagnosis case analysed in [Seppänen et al. 1998a].

Research customers can be considered having owned the exploitation process, taking into account the role of the Sokrates steering group members in the early phases of exploitation and the number of blue projects in which Reagenix was used in the nineties. Although the results of the Speco project remained just a demonstration system, they were paving the way for the Sokrates project. Another demonstration system was also built in connection with Sokrates, but for most members of the steering group, technical documents remained the only tangible results of Sokrates. Yet, the research and industrial customers of the projects listed in Table 13 exploited the code generation competence elements in a rather versatile manner. The Aniprosa graphical debugger was perhaps the only element the usefulness of which was suspected even by the focal researchers themselves. The researchers emphasised also the usefulness of the courses that they gave to students and industrial professionals, although such courses did not belong to the core R&D activities of VTT.

Selling of licenses was a new type of exploitation activity in the focal organisation, as was actually also the writing of professional-style technical users manuals and course material on Sokrates and Reagenix. These activities were central to the "commercialisation" process that was managed by the focal researchers. Yet, the process of controlling other processes as a whole was owned by the VTT managers. Tekes controlled, in principle, the research process during Sokrates, but only indirectly through funding and the Finnsoft program level activities. It is difficult to identify any owners for the other processes, such as the participation of colleagues in the planning of Sokrates and the evaluation of its results. Most activities carried out within these processes were of temporary nature, and had rather little effect on the developments that took place.

During the Speco period, the focus of the process portfolio was twofold, project marketing and purchasing on one hand, and early innovative research and development of code generation solutions on the other hand. These processes progressed in parallel and in a reversed mode compared with the ideal project portfolio from green to red projects. One of the reasons for this was that a few key people both at VTT and at firm K shared a vision of automatic software production and could therefore give a jump start to the work. Both parties aimed at producing the first tangible results soon and using them for launching more comprehensive development activities. Much less emphasis was put on the other processes than the actual R&D process. For example, the control and co-ordination of the Draco project was carried out mostly on an informal basis. From the viewpoint of the process portfolio, the period could be characterised as "Testing of an industrial vision".

During the Sokrates period, the focus was also twofold, but now on the R&D and exploitation processes. Project marketing and purchasing were carried out on the side of the Sokrates project, involving mostly the firms who participated in its steering group. The Electronics laboratory of VTT that became interested in the results was located on the same premises as TKO and even had its people in the steering group of Sokrates at some point. No great marketing and purchasing efforts were thus needed for initiating the early exploitation activities. Writing manuals on the use of the code generation technology and giving of courses on the Sokrates-SA design method may have been more oriented towards marketing, in addition to discussions with firm I, which ultimately failed. Although the whole Finsoft program was co-ordinated in a very structured manner, the focal researchers kept a tight control over research and exploitation, due to their technical expertise. The period could be characterised as "Demonstration of a research vision". The limits of technology that Tekes wished to be explored were not in sight yet, because the major case example of the Sokrates project was only a toy system.

During the Reagenix period, everything changed in the process portfolio. The large-scale R&D activities carried out in Sokrates finished almost entirely. Reagenix was more of a garage product compared with the Sokrates results engineered during three years and using millions of marks. The competence marketing and purchasing processes became critical for the continuation of the R&D process. The exploitation of Reagenix was quite sporadic, only rather small activities were carried out for individual industrial and research customers. The exploitation of Reagenix in research projects was controlled by research customers, while the code generation researchers did not play any central role. In industrial projects, the situation was slightly different, but it was more important to conduct R&D than to plan for and control competence marketing. Also the organisational control of the R&D process corrupted rather rapidly due to the managers losing interest in Reagenix as a business. The informal co-ordination process between the focal researchers faded away slowly, as a result of people leaving VTT. From the viewpoint of the process portfolio, the Reagenix period could be characterised as "Fighting for the vision".

4.3.1 Assuring of capability

Table 14 summarises changes in code generation capability during the three periods concerning product, process and organisation-related resources. Assuring capability involves the management of the life-cycle of resources, on which competence is based. The life cycle of product-related resources consists of the phases of incremental changes, discontinuity, substitution, competition and dominance. They include design and solution elements of human, technical and physical resources. We view their content from the four dimensions of applications, functions, techniques and technologies. The process-related resources include temporal and financial resources, and reputation is one of the most important organisation-related resources.

The main purpose of capability assurance is to manage inconsistencies in relationships. In practice, this can be performed not only as part of the competence marketing and purchasing processes, but also in connection with the control and co-ordination of ongoing R&D and competence exploitation processes. In Table 14, the management of inconsistency with high (+), low (-) or moderate (+/-) capability is evaluated for each type of code generation related resources from 1985 to 1998.

Speco was a period of discontinuity in terms of the life cycle of code generation related product recourses: code generation would mean radical changes in software development as industrial work and code generators were new as a tool technology. Contrary to the scientific and early commercial approaches to code generation, the focal researchers chose to use the SA/SD system design technique as their starting point. This technique dominated industrial embedded systems design in the late eighties, and some of the focal researches had "fallen in love" with it (Appendix 2). It was not a bad choice, as regards helping the researchers and industrial embedded software engineers to understand each other.

During the Sokrates period, code generation design and solution elements were developed to substitute existing elements, except for the Sokrates-SA design technique, which was an incremental extension of the SA/SD technique. The solution elements were generic technological implementations. It was not necessary to have an detailed knowledge of the functions they implemented and the techniques upon which they were based, to be able to exploit them. This was also the goal of using the Sokrates code generator, but its achievement could not yet be shown: VTT researchers were the main group of users.

The life cycle of product-related resource elements changed again during the Reagenix period. Code generation functions had already been developed and only incremental changes were required to improve their efficiency. The dominance of the SA/SD design technique was rapidly weakening due to object-oriented techniques (cf. Table 6), competition emerged regarding commercial code generators, and in most applications where code generation was used, changes remained incremental rather than radical. This was the case also in firm Nm, which was the key customer of Reagenix. Technological solution elements remained in their substitutive phase.

Table 14. Assuring of code generation capability by VTT.

Speco	**Sokrates**	**Reagenix**
Product: human, technical and physical resources		
Design elements: - functions: discontinuity (innovation of code generation) - techniques: dominance (SA/SD widely used) - technologies: discontinuity (use of generators) - applications: discontinuity (change the way to develop software) Solution elements: --	Design elements: - functions: substitution (use of code generation instead of manual coding) - techniques: incremental changes (Sokrates-SA) - technologies: substitution (wide use of generators) - applications: discontinuity (change the way to develop embedded software) Solution elements: - technologies: substitution (replace existing solutions)	Design elements: - functions: incremental changes (improvement of code generation) - techniques: substitution (SA/SD dominance is over) - technologies: competition (commercial tools exist) - applications: incremental changes (many different applications addressed), Solution elements: - technologies: substitution (replace existing solutions)
Management of inconsistency of product resources between 1985 - 1998 Design elements: - applications (+/-): trying to cause a radical change in the development of industrial software, ending up to deal with many kinds of applications - functions (+): research, development and gradual improvement of code generation - techniques (-): improvement of SA/SD, while competing methods emerged rapidly - technologies (-): trying to commercialise the code generator in a competing situation Solution elements: - technologies (+/-): attempting to replace existing solutions, partially succeeding		
Process: temporal and financial resources		
Draco: resources gained and managed as planned Speco: disagreement over temporal resources caused problems with firm K, financial resources available to make the initial innovation	Sokrates: success in gaining financial resources was better than planned, problems with temporal resources Other projects: financial resources moderate, temporal resources good	Green projects: very modest financial resources Blue projects: no income from temporal resources Red projects and licenses: disagreement over financial resources with managers, minimal temporal resources
Management of inconsistency of process resources between 1985 - 1998 - temporal resources (-): management of resources by focusing on the analysis of problems and then trying to solve them very fast caused problems in Speco, resulted in unfinished work in Sokrates and was criticised by the VTT managers during Reagenix - financial resources (+/-): financial resources were good during Speco and excellent during Sokrates, but collapsed then rapidly due to the lack of considerable projects		
Organisation: reputation and other organisational resources		
Reputation: leading embedded software engineering research unit Other: strong role in national R&D initiatives	Reputation: few problems with the steering group and with some colleagues Other: investments in equipment, co-ordination	Reputation: conflict with firm I caused some problems, dispute with the managers Other: almost no resources provided by the organisation
Management of inconsistency of organisational resources between 1985 - 1998 - reputation (+/-): deterioration of the good reputation among the managers; rather good reputation sustained among the early and late internal and external exploiters - other resources (+/-): full organisational support during Speco and Sokrates changed to a doubtful situation during Reagenix, resulting in the loss of organisational visibility		

The capability of code generation related product resources is decreasing during the three periods, except code generation functions that were gradually improved from their early innovative stage. Enhancing the capability of the technological design resource elements, most notably the Sokrates and Reagenix code generators, did not succeed. The kind of commercialisation at which the researchers aimed also failed. Furthermore, there were problems in the capability of the SA/SD system design technique, which lost its dominating position by the mid-nineties.

The code generation capability concerning automation applications was high in the early nineties, not only among the focal researchers but at TKO as a whole. Yet, the automation sector recovered very slowly from the recession and the consumer electronics sector for which some code generation projects were also carried out almost disappeared from Finland. No special capabilities were built for the rapidly increasing telecommunication sector. Assuring of capability in terms of the life-cycle of applications was apparently not successful as a whole, the result was a mixed bag of research applications with no special attention paid to the enormously growing telecommunication sector. The developed technological solution elements were intended for replacing existing solutions; they succeeded at least partially, as indicated by the Osdyn project, for example. However, there was no capability of creating any considerable business based on these elements, such as a continuous flow of license fees or IPR sales income.

The life-cycle of the process-related temporal and financial resources depended much on the phases of the R&D and competence exploitation processes. Capability in terms of financial resources was sufficient during Speco and even better than expected during Sokrates, but it collapsed by 1994. There were problems in capability regarding calendar time and manpower in the Sokrates project, because some of the central tasks could not be finished. In the Speco project, this had caused a dispute with the customer. During Reagenix, the temporal resources for carrying out further research and development were very limited indeed. The managers, who controlled the time and money spend on marketing, did not want them to be invested in Reagenix. In terms of the management of inconsistency concerning process-related resources, the focal organisation thus succeeded mainly in assuring of financial capability during the Sokrates period, which was the competence development and early exploitation phase. It could not assure the continuation of financial capability after the Sokrates project.

Capability of organisation-related resources was good during Speco and Sokrates, when TKO was centrally involved in code generation related national research and development activities. Later, when the logic of action of the managers and researchers diverged, only few organisational resources were available to the researchers. Disagreements with firm I did not seriously affect the reputation of VTT, although they were noticed by some customer companies. The decrease of organisational resources resulted in a rapid loss of the visibility of code generation related activities. What used to be a promising idea during Speco and flagship research during Sokrates, became an almost invisible hobby of those of the focal researchers who stayed at VTT.

4.3.2 Balancing of particularity

Balancing of the particularity of resources for certain relationships is done by the absorption and problem solving functions that affect the codification of product-related resources and the contextuality of process and organisation-related resources. Table 15 provides and overview of the balancing of particularity of code generation resources from 1985 to 1998, by means of problem solving (+) for generic and context-independent resources and absorption (-) for tacit and context-specific resources.

The product-related code generation design elements were quite tacit during the Speco period, although the innovative skills of the focal researchers were used for producing the first explicit technical and physical resources. The code generation functions, the prototype of the PLM generator that was built, and knowledge of the applications of firm K were all particular to the Speco project. However, the idea of using the generic SA/SD technique that was explicitly described in textbooks and taught in courses was already emerging. As an alternative, it would have been possible to use application-specific design techniques and languages, as in the Draco and Refine tools that were evaluated in the Draco project. This would have meant high particularity of the developed design elements also with regard to the technique dimension.

During the Sokrates period, the code generation functions became portfolio-specific, because they were tailored to a few application areas. Also the Sokrates-SA design technique can be considered somewhat portfolio-specific, as the focal researchers emphasised its use in solving hard real-time and concurrent systems problems. Even the Sokrates generator was not fully generic, but designed to support code generation from structured system models, especially from the so-called state transition and functional models. Finally, when also the focus of the application dimension was directed to machine automation, it can be said that the particularity of the product-related design elements as a whole turned into portfolio-specific.

The solution elements that were developed in Sokrates were generic, and remained such also during the Reagenix period. The Reagenix code generation functions and generator were also generic, i.e. the particularity of the code generation functions and their implementation technology decreased. The Sokrates-SA based design technique was replaced by a generic component-based approach that could be tailored for specific needs. A generic visual modelling tool was used for defining and modifying the notation. This relieved the researchers of the need to use the CASE tool of firm I as a part of their tool environment. Earlier, the Sokrates-SA models needed to be drawn by using this tool. Since there was no focus on specific applications any more during the Reagenix period, the overall particularity of product-related resources decreased, all of the resources were quite generic. The continuous decreasing of the particularity of the resources fits with the logic of the focal researchers in creating a generic, well-packaged code generation solution. It can also be seen as a direct result from the change of process nets from one dyadic relationship via a large multi-party network to a number of small dyadic relationships and transactions.

Table 15. Balancing of particularity of code generation resources by VTT.

Speco	**Sokrates**	**Reagenix**
Product: human, technological and physical resources		
Design elements: - functions: tacit (specific code generation rules invented for firm K) - techniques: generic (Ward-Mellor SA/SD) - technologies: tacit (code generator prototype) - applications: tacit (individual skills in the application of firm K) Solution elements: --	Design elements: - functions: portfolio-specific (code generation applied in several projects) - techniques: portfolio-specific (Sokrates-SA) - technologies: portfolio-specific (SA/SD generator), - applications: portfolio-specific (focus on machine automation applications) Solution elements: - technologies: generic (widely usable solutions)	Design elements: - functions: generic (code generation applicable to many different problems) - techniques: generic (Reagenix models) - technologies: generic (Reagenix code generator), - applications: generic (no specific focus on certain applications or problems), Solution elements: - technologies: generic (widely usable solutions)
Absorption (-) and problem solving (+) of product resources in 1985 - 1998 Design elements: - applications (-/+/+): starting from a specific product application, trying to focus on machine automation and ending up with dealing with many different types of applications - functions (-/+/+): from specific via SA/SD based to generic code generation rules - techniques (-/+/+): changing of SD/SD to Sokrates-SA and then to Reagenix models - technologies (-/+/+): development of a commercial generic code generator Solution elements: - technologies (x/+/+): development and utilisation of generic solution elements		
Process: temporal and financial resources		
Draco: generic (loosely controlled evaluation of new technologies) Speco: portfolio-specific (resources provided as part of strategic co-operation between TKO and firm K)	Sokrates: portfolio-specific (resources provided as a part of the Finsoft research program) Other projects: context-specific (participation as experts on an individual basis and occasionally)	Green projects: context-specific (occasional) Blue projects: context-specific (occasional) Red projects/licenses: portfolio-specific/generic (resources provided within projects and from licenses)
Absorption (-) and problem solving (+) of process resources in 1985 - 1998 - temporal/financial resources (+/+/-): most of the resources were portfolio-specific during Speco and Sokrates, during Reagenix most resources were context-specific		
Organisation: reputation and other organisational resources		
Reputation and other resources: portfolio-specific (contractual project-based embedded systems R&D in Finland)	Reputation and other resources: portfolio-specific (no change to the earlier situation)	Reputation: context-specific (organisational restructuring resulted in a greater and more heterogeneous unit) Other: context-specific (project-based resources)
Absorption (-) and problem solving (+) of organisational resources in 1985 - 1998 - reputation (+/+/-): TKO created a portfolio-specific reputation based on embedded systems research and development, ELE needed to renew the reputation in the nineties - other resources (+/+/-): project-based planning and management of R&D activities increased uniformity of the institute; this caused problems in license selling and small-scale code generation problem solving tasks		

Particularity of process-related resources does not directly correspond to particularity of product-related resources, except during Sokrates when also most temporal and financial code generation resources were portfolio-specific. They had been portfolio-specific even in the Speco project, which was a part of the strategic alliance between firm K and TKO. The particularity of process-related resources became high at the end of the Sokrates period and especially afterwards. The resources depended largely on the occasional projects and license selling transactions that could be established or in which the focal researchers happened to participate.

Therefore, while the particularity of product-related code generation resources gradually decreased and they became very generic, the particularity of process-related resources increased rather rapidly, also due to changes in process nets. The consequence was that highly generic technical and physical resources were offered in connection with highly specific financial and temporal resources. The comparison of the Kaapeli (1990 - 1991) and the AVV (1995 - 1997) projects illustrates the change that took place.

The Kaapeli project was carried out for firm Nm, which wished code generation functions to be tailored for a specific programming language, with an intention to use them manually. The project was one of the first spin-offs of the Sokrates project, done as a Master's thesis work by a research trainee. The code generation rules and their use were highly particular to the needs of the customer, whereas the work was carried out as a part of Sokrates and its early exploitation projects. Special code generation functions were thus developed in connection with a portfolio of code generation related activities.

None of the focal researchers took part in the AVV project, where Reagenix was used as an off-the-shelf tool by three software safety researchers of VTT, to produce a certain simulation program. The schedule, volume, budget, tasks and human resources of the AVV project determined fully why and how the generator was used. The focal researchers offered to help in the use of the generator, but "no help was needed" (Appendix 2). They did not receive any income from the project either. A fully generic code generation solution was thus used in a one-of-a-kind project.

Particularity of the organisation-related resources follows a pattern that is similar to process-related resources. In the late eighties and early nineties TKO aimed at creating a reputation of an embedded systems expert organisation, whose main products were demanding research and development projects (cf. the characterisation given by the laboratory director Hannu Hakalahti in Appendix 2). In other words, it wanted to make use of portfolio-specific project-based organisational resources for the needs of industrial customers interested in developing or applying embedded systems in their products. The organisational uniformity of project-based R&D activities have become even more important since 1994, when ELE was formed as a much larger and more heterogeneous unit than TKO. The particularity of the idea of selling licenses and the focus on internal use of self-developed design techniques and tools were rather high, compared with the aim of increased organisational uniformity at ELE.

4.4 EVOLUTION AND VALUATION OF THE COMPETENCE

We will now discuss the evolution the code generation competence at VTT during the period 1985 - 1998. The integration of competence elements through activity and resource functions will also be addressed, as well as views on the valuation of competence elements by different parties. Competence has been classified into product, process and organisation-related elements. The integration of competence elements may involve resource collections and activity structures (in the focal organisation), resource ties and activity links (in relationships), or resource constellations and activity patterns (in networks). Competence elements may be strengthening, competing, complementary or neutral with regard to integration with other elements.

The longitudinal value of competence is viewed from the perspectives of one supplier and several purchasers, concerning the expected, perceived and historical values to the extent that they could be identified from the case data. Expected value refers to the valuation of competence before its development or exploitation, i.e. it is a kind of *market value* of the competence. The perceived or *delivered value* of competence is evaluated during and at the end of its development or exploitation. Expected and perceived values of competence were analysed in this research mostly on the basis of documentary data. Historical or *use value* refers to the valuation of competence after its development or leverage, analysed in this research on the basis of interviews.

4.4.1 Competence during the Speco period

Tables 16 summarises the competence elements possessed or created by VTT during the Speco period, as well as their planned and realised integration and their valuation. Product-related competence elements involve skills of developing embedded computer system technologies, knowledge of their design techniques, skills of evaluating and developing new technologies and capabilities in machine automation applications. The latter were in part absorbed from firm K, which possessed also the other competence elements. It wished to lead the Speco project also in technical terms, which created competition on how innovative research should be carried out. Yet, most of the elements strengthened both internal activity structures and resource collections of TKO and the activity links and resources ties with firm K. The capability of TKO of understanding machine automation applications complemented its actor bonds with firm K. The focal researchers considered the historical value of all of the product-related competence elements as very high (++) or high (+), whereas the VTT managers were somewhat more critical in their valuation. They considered the historical value of tool development skills as low (-), but the value of knowledge of system design techniques was considered high also by them. Tekes valued product-related code generation competence during the Speco period as high, but it was not yet directly involved in code generation related activities. Firm K considered the historical value of the elements as moderate (+/-) or low, especially the skills of TKO in developing new design tools.

Table 16. Code generation competence during 1985 - 1987.

Competence elements/ Owners	Planned/Realised integration of elements	Supplier valuation	Purchaser valuation
		(E)xpected, (P)erceived, (H)istorical	
Product-related competence elements			
Skills to develop embedded computer system technologies: TKO (possessed also by firm K)	Strengthening of internal resource collections/ok Strengthening of resource ties with firm K/ok	Researchers: H ++ Managers: E ++, P +, H +/-	Firm K: E ++, P +, H +/- Tekes: H ++
Capabilities in machine automation applications: TKO (absorbed in part e.g. from firm K)	Complementing of actor bonds with firm K/ok	Researchers: H + Managers: E ++, P +, H +/-	Firm K: H +/-
Knowledge of embedded system design techniques (especially SA/D): TKO (possessed also e.g. by firm K)	Strengthening of internal resource collections/ok Strengthening of resource ties with firm K/ok	Researchers: H ++ Managers: E ++, P +, H +	Firm K: E ++, P +, H +/- Tekes: H +
Skills to forecast, evaluate and develop new embedded system design tools: TKO (possessed also e.g. by firm K)	Strengthening of internal activity structures/ok Strengthening of activity links with firm K/in part competing	Researchers: H ++ Managers: E ++, P -, H - Colleagues: H +	Firm K: E +, P +/-, H - Tekes: H +
Process-related competence elements			
Skills in (team-based) solving of complex computer-related problems: focal researchers	Strengthening of internal activity structures/ok Strengthening of activity links with firm K/competing	Researchers: H ++ Managers: H + Colleagues: H +/-	Firm K: E ++, P -, H -
Capability to initiate and conduct embedded software engineering R&D: TKO (especially managers and senior researchers)	Strengthening of internal activity structures/ok Strengthening of activity links with industry/ok Strengthening of actor bonds (and activity links) with Tekes/ok	Researchers: H ++ Managers: H + Colleagues: H +/-	Tekes: P ++, H ++ Industrial parties: H +
Organisational competence elements			
Skills to establish and use contacts with Tekes and industry: TKO	Strengthening of actor bonds and webs with external parties/ok	Researchers and managers: P: ++, H ++	Tekes: H ++ Industry: H ++
Parties isolated from the competence elements: KTM: provided funding for the Draco project, but was not otherwise involved			

The two most important elements of process-related competence were team-based problem solving skills of the focal researchers and the organisational capability of launching R&D activities. They strengthened internal activity structures, external activity links with industry and external actor bonds especially with Tekes. An exception in this regard was the dispute with firm K on how to carry out the innovative research process. Yet, this did not break the mutual actor bond or even the activity link, although firm K perceived it as an indication of low competence.

Some interviewed colleagues criticised the organisational processes which were used for controlling the embedded software engineering research at TKO, but all the other parties evaluated the capability of TKO in this regard as high or very high. The fact that TKO was a key player in Finsoft, thanks to its remarkable involvement in conducting and co-ordinating applied engineering research, indicates the same. This capability was provided together with knowledge of embedded systems design techniques, which was clearly valued as the best element of product-related code generation competence of TKO during the Speco period. Together with the main organisational competence element, the skills of establishing and making use of contacts with both Tekes and industry, they were paving a way for a pragmatic, yet innovative approach to code generation. The organisational skills of networking strengthened actor bonds and webs. Considerable activity links existed only with firm K, but the skills of establishing contacts helped to create new activity patterns as a part of and related to the Sokrates project.

KTM was isolated from the competence of VTT, because it provided funding for the Draco project but was not involved in any other way in code generation related activities.

4.4.2 Competence during the Sokrates period

During the Sokrates period, the versatility of competence elements increased remarkably, as indicated in Tables 17, 18 and 19. Many product-related competence elements became portfolio-specific. Functional skills were made explicit via code generation rules. These skills were intended for complementing resource ties with industry, in which knowledge of how to generate computer programs from design models was still rare. In reality, most of the members of the Sokrates steering group did not need their competence to be complemented with these skills, and firm I came up with such skills without VTT. The skills in system design techniques turned into portfolio-specific kind in developing modelling languages and CASE tool technologies. This provided better means of strengthening resource collections and ties than the knowledge of the generic SA/SD method. The skills of developing embedded systems implementation technologies specialised in skills in system solutions that were widely exploitable. Application knowledge became also portfolio-specific, because there was no stricter focus on any single application. New application knowledge needed to be absorbed from customers and the Electronics laboratory of VTT, to complement the skills of the focal researchers. However, application knowledge did not play any central role during Sokrates.

Table 17. Product-related code generation competence during 1988 - 1991.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Skills of developing embedded computer system technologies (system solutions): focal researchers (in part absorbed from the Electronics laboratory)	Strengthening of resource ties (and constellations) with the steering group of Sokrates and with industrial and research customers/ok	Researchers: H ++ Managers: P +, H +/-	Steering group: P +, H +/- Tekes: H ++ Evaluators: P +/- Research customers: P ++, H + Industrial customers: P ++, H ++
Knowledge of embedded systems applications: TKO (in part absorbed from customers)	Complementing of actor bonds with industrial (and research) customers/mostly neutral	Researchers: H ++ Managers: P +, H +/- Colleagues: H+	Steering group: P +/-, H +/- Tekes: H + Evaluators: P + Industrial customers: P ++, H +
Skills of developing and transferring in use design languages, methods and CASE tools: TKO (possessed also by some customers and the Electronics laboratory)	Strengthening of internal resource collections/ok Strengthening of resource ties with research (and industrial) customers/ok	Researchers: H ++ Managers: E ++, P +/-, H +/- Colleagues: P+/-, H -	Steering group: E ++, P +, H +/- Tekes: H + Evaluators: P +/- Industrial and research customers: P +, H +/-
Skills of developing manual coding rules and automated code generation functions: focal researchers (also possessed by the firms K, S, N and I)	Complementing of resource ties with the steering group /mostly neutral, in part ok, in part competing Strengthening of resource ties with customers/mostly neutral, in part ok and competing	Researchers: H ++ Managers: E ++, P +, H + Colleagues: P +/-, H -	Steering group: E ++, P +, H +/- Tool vendors: P +, H - Tekes: H + Evaluators: P +/- Industrial and research customers: P +, H +/-
Parties isolated from the competence elements: Colleagues (especially the ones not involved in the activities of the R Group) (Members of the steering group: mostly did not directly exploit the results) (VTT managers, Tekes: only co-ordinated the R&D process)			

Technological skills mastered by the focal researchers were valued as high by most of the parties. The VTT managers considered their historical value as rather modest, and some evaluators criticised their inclusion in the project. One of the main competence elements of the focal researchers was seen in their functional code generation skills. The expected and perceived value of these skills was high, except for the opinions of some colleagues and the evaluators of the Finnish Academy (cf. Appendix 2). However, their historical purchaser value was quite moderate among the Sokrates steering group and the early exploiters. Firm I did not value the skills high enough to exploit them in its own code generator.

Skills of developing system design languages, CASE tools and methods can be considered part of the competence of TKO, not only of the focal researchers. They were valued higher by the researchers than the managers. Some of the colleagues considered the skills even as a kind of alchemy, or at least not good enough scientifically: "finding the highest point of a fence and going under it". Their historical value was rather low also among many of the steering group members. According to Tekes, it paid off creating and exploiting the skills, and the turn of the nineties was the right time for doing it. Customers had somewhat varying opinions regarding the historical value of the Sokrates results in this regard, while only few of them seized the opportunity of making use of the results in any spin-off projects. Knowledge of embedded systems applications was also part of the competence of TKO, but the managers valued its use in code generation research as rather moderate in interviews. The applications in which the Sokrates results were exploited were quite diversified. The toy elevator built for demonstrating the results was seen as a good choice by the focal researchers and their colleagues, but some of the steering group members and the VTT managers considered it as too trivial.

In summary, one of the main elements of product-related code generation competence during the Speco period, knowledge of embedded systems design techniques and SA/SD in particular, was reshaped to portfolio-specific skills in languages, methods and tools, while its dependence on certain kinds of applications weakened. The skills could be associated with knowledge of functions for the purpose of generating code from system models. The Sokrates-SA design method, which was described in technical documents, tied the elements together. Code generation functions were the most largely exploited part of the competence, as they were used in the Kaapeli, Synchro and Sasic projects. The tool technologies were still immature, as indicated in Appendix 2. In the Kaapeli project, they were not used at all. The Sokrates-SA design technique included elements, such as the Ada-like design language, the value of which was rated equally low by most of the parties involved.

Solution elements were also created, e.g. an operating system kernel and a communication protocol software package. They were more generic than the design elements to which they could be integrated. Their role can be seen as either replacements of other similar solutions independently from the design elements, or as filling of technological gaps in the design method so that it would became self-sufficient. The Osdyn project of the Reagenix period is an example of the former, while the development of the toy elevator as the Sokrates demonstration system of the latter.

Product-related competencies of colleagues were isolated from the code generation competence. Also the competencies of the other members of the steering group than firm Nm, who exploited the results, were rather isolated. Code generation competencies of the firms I and N are good examples of this isolation. Moreover, since Sokrates had all the necessary project resources, there was no urgent need for associating the competencies of the VTT managers, colleagues, Tekes and the focal researchers with each other. The researchers were running the show and the other parties were acting as the audience.

Table 18. Process-related code generation competence during 1988 - 1991.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Capability of planning, carrying out and managing contractual code generation R&D projects: focal researchers, (TKO)	Strengthening of internal activity structures/ok Strengthening of actor bonds with Tekes and industry/mostly neutral	Researchers: E+, P +, H +/- Managers: E ++, P+/-, H +/- Colleagues: H -	Steering group: P +/-, H +/- Tekes: H + Evaluators: P +/- Industrial and research customers: P +, H +
Skills of popularising, documenting and publishing R&D results: focal researchers	Strengthening of actor bonds (and activity links) with Tekes, industry and the R&D community /in part competing	Researchers: P +, H +/- Managers: P +/-, H +/- Colleagues: H -	Steering group: P +, H + Tekes: H + Evaluators: P + Customers: H -
Skills of training practitioners and educating students: focal researchers	Strengthening of actor bonds (and activity links) with educational institutes and industry /ok	Researchers: P ++, H ++ Managers: P +, H +/- Colleagues: H +	Industrial and research customers: P + Educational institutes: P ++
Skills of conducting (international) scientific research: focal researchers, TKO	Strengthening of actor bonds (and activity links) with the scientific research community/neutral	Researchers: P +, H +/- Managers: P -, H +/- Colleagues: H --	Evaluators: P + Scientific research community: P +/-
Parties isolated from the competence elements: Evaluators (Finsoft, Finnish Academy), because of the post-mortem analysis Some members of the steering group Colleagues (except members of the research council)			

The main process-related competence element during the Sokrates period involved conduction of the actual research process. It clearly strengthened both the external actor bonds of TKO and the internal activity structures among the Sokrates project group. However, it did not offer too much strengthening for internal resource collections and activity structures or external resource ties and activity links, because the project was almost totally self-sufficed. In comparison, the Speco project can be characterised as a close relationship of the focal researchers and firm K for the purpose of performing activities for creating and making use of code generation resources. This relationship was co-ordinated and controlled by the VTT managers, who were interested in the strategic alliance with the customer company. Although the Sokrates project involved a number of actor bonds, it only showed rather few activity links and resource ties. The training on the Sokrates-SA method given by the researchers to industrial professionals and students resulted in bonds with a large number of actors, but not in any considerable activity links or resource ties.

The focal researchers thus became quite distant from external parties with regard to the exploitation of code generation resources, with the exception of firm Nm. In particular, no activity links were established with firm N, the code generator expert of which joined TKO, but not the Sokrates project.

The Sokrates project team did not succeed perfectly in making use of technical and managerial documentation to link its activities and their results with the activities and resources of the steering group. The group complained regularly on delayed or missing pieces of information. There was an even more obvious lack of activity links and actor bonds concerning scientific and professional publications and interaction with recognised members of the software engineering research community. Many other Tekes-funded projects carried out by TKO at the turn of the nineties had many such bonds and some had also rather extensive activity links with foreign research institutions. Yet, this lack was not seen as problematic by the managers, although a few colleagues were criticising it.

The evaluation of the process-related competence elements shows some interesting differences. The managers and even the researchers themselves considered the historical value of the actual research results as somewhat mixed. As the hand-written comments of the final Sokrates report by Veikko Seppänen from 1991 (Appendix 2) and the remarks on the disappointing small number of spin-off projects of Sokrates indicate, the managers started to suspect the usability of the results even before the end of the project. The researchers explained in the interview that the Sokrates code generator from April 1991 was technically rather immature and could not yet be used in any serious software development work. This was also pointed out by one of the interviewed colleagues as a possible reason for the rather modest industrial interest in using the Sokrates results.

However, both Tekes and the early customers were quite satisfied with the both results and the way that things had been developed and exploited. Some members of the Sokrates steering group viewed the focal researchers as too self-satisfied, but rather many of them turned out to be just passive bystanders who did not associate with the researchers by any other way than by attending steering group meetings. The training given by the researchers helped to disseminate information on the results, which was, after all, what most of the steering group members were obviously expecting. Although the lack of interaction with recognised international software engineering researchers was criticised internally at TKO, the steering group and customers did not consider it any remarkable problem either. The foreign Finsoft evaluators indicated that the results were a match for the best international research. The evaluators performed, however, only a post-mortem analysis based on technical documents and a short interview.

The evaluation organised by the Finnish Academy illustrates, how such an isolated activity can lead to misleading remarks. Many of the Sokrates steering group members were actually also isolated, because they did not master the technology and their needs were not directly addressed in the project. Many of them did not mention code generation as their primary interest (cf. Appendix 2).

Yet another notable party isolated from the emerging code generation competence was the TKO researchers, who were not involved in other Finsoft projects and therefore did not participate in the research council, which held regular meetings to exchange information on the projects. These colleagues included researchers who had been involved in the initial planning of the Sokrates project, as well as others who used to work as members of the R Group, established as an internal interest group.

Table 19. Organisational code generation competence during 1988 - 1991.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Skills of making use of contacts with Tekes, other VTT laboratories and industry: TKO	Strengthening of actor bonds and webs/ok	Researchers: P +, H + Managers: P +, H +/-	Tekes: H ++ Research customers: H + Industrial customers: H +
Capability of marketing R&D results and making use of the good reputation of the institute: TKO	Strengthening of internal activity structures/neutral Strengthening of activity links with industry/partially ok, involving also VTT Electronics laboratory	Researchers: P ++, H +/- Managers: P +, H -	Steering group: P +, H +/- Tekes: H + Evaluators: P + Industrial and research customers: P +, H +
Capability of organisational planning and project quality management: TKO	Strengthening of internal activity structures/competing Strengthening of activity links with industry/competing	Researchers: P -, H -- Managers: P +/-, H -	Steering group: P +/-, H + Tekes: H + Industrial and research customers: P +, H +
Parties isolated from the competence elements: Management group of the Finsoft program (except J. Karjalainen, H. Hakalahti)			

One of the most important organisational competence elements during the Sokrates period was the skills of TKO in maintaining and making use of its contacts with Tekes and industry. This resulted in quite strong external actor bond and webs, although activity links and resource ties were much less common, and internally the Sokrates project team was rather isolated from the other researchers. Moreover, since Jukka Karjalainen and Hannu Hakalahti were almost the only contacts of Sokrates with the management group of the Finsoft program, the program level remained quite isolated from the project and its steering group.

Some mixed opinions were given in interviews regarding the capability of TKO to market the Sokrates results. As indicated in Appendix 2 and at the beginning of this report, the managers started to suspect the marketing plans of the researchers even before the end of the project, regarding them as wishful thinking. The two parties disagreed also regarding the appropriate management of R&D projects. Although the steering group seemed to take the side of the managers in this matter, it did not give remarkably negative feedback in this regard, after all.

The group was invited to launch Tekes-supported spin-off projects early, but only firm Nm used the opportunity. Historically, the managers themselves missed the opportunity of turning firm N to a key exploiter of the results. They helped the researchers in trying to establish an activity link with firm I, but it did not succeed. On the other hand, the VTT Electronics laboratory was an unexpected exploiter of the results, which seemed to provide a good case for internal marketing of the results among VTT.

4.4.3 Competence during the Reagenix period

The product-related competence elements became quite versatile during the Reagenix period, as shown in Table 20. Although the Sokrates generator was not taken in industrial use, the Sokrates-SA method and the system solutions were utilised by industrial and research customers. The value of the former was perceived as high by all parties, but the historical value was seen as remarkable only by the researchers and firm N which were making successful use of the operating system principle developed in Sokrates.

Although the applications in which Reagenix was utilised were more diversified than earlier, when the focus had been on machine automation, it was mainly the managers only that suspected the researchers' skills in this regard. Some of the representatives of the Electronics laboratory were also critical, while, on the other hand, claiming that the TKO and ELE managers were responsible for the lack of adequate support to the use of Reagenix in certain applications. Yet, it is fair to say that the focal researchers provided a great deal of help especially for research customers, considering that they were usually not paid for it.

Such broad co-operators as firm Nm sought for comprehensive embedded software development approaches. This aim did not create particularly strong resource ties with the customers - the projects listed in Table 10 were rather small and did not result in continuing projects. They were still perceived as good, except maybe for the Raski project carried out for the firm R, which was disappointed with the results. The VTT managers were not fully satisfied with the way some projects were implemented. The managers were, in their opinion, not extensively enough involved in the co-ordination and control of the projects. The focal researchers thought that the managers did not understand how Reagenix should be transferred in industrial use in such an efficient manner that would benefit the customers. Firm I viewed the selling of Reagenix licenses as a competitive attack against its own generator, but did not raise any public dispute over it.

Internal competition emerged at ELE, concerning Reagenix as "the best approach to embedded systems software development". One of the focal researchers characterised its marketing as "American style marketing that irritated everyone". The historical evaluation of Reagenix by firm Nm was quite devastating indeed. However, the representatives of the firm blamed in part themselves as incompetent buyers, who were trying to make too ambitious a leap in the middle of a recession. The perceived value of Reagenix during and immediately after the MCS-REA project by both firm Nm and the focal researchers had been high.

Table 20. Product-related code generation competence during 1992 - 1998.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Skills of applying embedded computer system technologies (system solutions): focal researchers	Strengthening of resource ties with industrial and research customers/in part ok, mostly neutral	Researchers: H ++ Managers: P +/-, H -	Research customers: P +, H +/- Industrial customers: P +, H +/-
Knowledge of embedded systems applications: focal researchers (in part absorbed from customers)	Complementing of actor bonds with industrial (and research) customers/ok	Researchers: H ++ Managers: P +/-, H +/-	Industrial customers: P +, H + Research customers: P +, H +/-
Skills of developing and transferring in use code generation driven embedded systems design approaches: focal researchers	Strengthening of internal resource collections/ competing Strengthening of resource ties with broad co-operators/ in part ok	Researchers: H ++ Managers: E +, P +/-, H -	Industrial and research customers (broad co-operators): P +/-, H +/-- Tool vendors: P +/--
Skills of developing and transferring in use code generation based design and testing tools: focal researchers	Strengthening of internal resource collections/neutral Strengthening of resource ties with focused buyers/ok	Researchers: H ++ Managers: E +, P +/-, H +/-	Industrial and research customers (focused buyers): P +, H + Tool vendors: P +/-
Capability of extending code generation functions further: focal researchers	Strengthening of resource ties with customers/mostly neutral, in part competing	Researchers: H ++ Managers: E +, P +/-, H +/-	Industrial and research customers: P +, H +/-- Tool vendors: P +/--
Parties isolated from the competence elements: VTT managers: did not control the development and application of Reagenix Research customers: only used Reagenix, did not control its development			

Focused buyers were more satisfied also in a historical sense, be they research or industrial customers. The VTT managers did not regard the exploitation of individual tools and system solutions as any great business success. According to the focal researchers, this kind of viewpoint was totally wrong, the purpose of VTT should have been to help customers as effectively and efficiently as possible, not to make big profits at their cost. Although the individual tools and system solutions were utilised also internally in several research projects, they did not became integrated into any considerable larger resource collections, but remained rather isolated from the other competence elements of the focal organisation.

One of the main reasons for this might have been that the other researchers were users of the Reagenix design and solution elements, not their active developers. The code generation functions that were made explicit as transformation rules during Sokrates became incorporated inside the code generator tool. There were plans to design code generation functions for other languages than C, but they did not materialise until in the Nosto project, which was carried out for firm Kc. Therefore, the functional dimension the competence became less distinct than, for example, in the Kaapeli project where it was explicitly used for the needs of firm Nm.

Table 21. Process-related code generation competence during 1992 - 1998.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Skills of planning, carrying out and managing self-funded code generation research projects: TKO	Strengthening of internal activity structures and resource collections/competing	Researchers: P +/-, H +/- Managers: P -, H -	-
Skills of planning, carrying out and managing contractual code generation R&D projects: TKO	Strengthening of internal activity structures/in part ok Strengthening of activity links and patterns with Tekes, industry/in part ok	Researchers: P +/-, H +/- Managers: E +, P -, H -	Tekes: P - Industrial customers: P +, H -
Skills of packaging R&D results into commercial products and IPRs/ focal researchers	Complementing (and strengthening) of resource ties (and activity links) with industry/competing	Researchers: P +, H + Managers: P +/-, H -	Industrial customers: P +, H +/- Research customers: P +, H +/- Tool vendors: P +/-
Skills of training practitioners and educating students: focal researchers	Strengthening of actor bonds (and activity links) with educational institutes and industry/ ok	Researchers: P ++, H ++ Managers: P +, H +/-	Industrial and research customers: P + Educational institutes: P ++
Parties isolated from the competence elements: Research customers: only used Reagenix, did not carry out code generation R&D			

Evolution of the process-related code generation competence elements became stalled after Sokrates (Table 21), because of the rapid collapse of the project portfolio in the financial sense. The autonomous position of the researchers in terms of process-related and organisational resources disappeared.

The VTT managers and Tekes expected the exploitation of the results to materialise as red projects – far too soon, as claimed by the focal researchers. The managers became quite disappointed with the actually realised volume of red activities. They did not prevent the researchers from following their product-based logic of action, but did not consider the results of license selling any great success either.

The use of Reagenix in quite a number of blue projects strengthened internal resource collections, but not activity structures, because the focal researchers were seldom involved in the projects as active participants. The self-funded green research related to code generation partially resulted in competition, not only because firm I was developing its own tool environment simultaneously, but also since ELE researchers were investigating alternative software development techniques, methods and tools in other blue projects.

One of the best process-related competence elements was, again, the researchers' skills of training students and customers. However, many of the research customers learned the use of Reagenix much by themselves.

Table 22. Organisational code generation competence during 1992 - 1998.

Competence elements/ Owners	**Planned/realised integration of elements**	**Supplier valuation**	**Purchaser valuation**
		(E)xpected, (P)erceived, (H)istorical	
Contacts with Tekes, other VTT institutes and industry: TKO/ELE	Strengthening of actor bonds and activity links/in part ok	Researchers: P +, H + Managers: P +, H +/-	Tekes: H ++ Research customers: H + Industrial customers: H +
Capability of marketing R&D skills and making use of the reputation: TKO/ELE	Strengthening of internal activity structures/competing Strengthening of activity links with industry/partially ok	Researchers: P ++, H + Managers: P +/-, H -	Research customers: H + Industrial customers: H +
Capability of organisational planning and project quality management: TKO/ELE	Strengthening of internal activity structures/ok Strengthening of activity links with industry/ok	Researchers: P -, H -- Managers: P +, H +	Research customers: H +/- Industrial customers: H ++
Parties isolated from the competence elements: Code generation researchers: were not involved in organisational planning and management regarding code generation R&D			

The main elements of the organisational competence (Table 22) during Reagenix involved making use of the extensive contacts and good reputation of VTT among potential customers, marketing of research and development skills, and planning for and carrying out R&D activities. From the viewpoint of Tekes and customers, the organisational competence of VTT was high. However, the VTT managers and the focal researchers both perceived and indicated in the interviews that they did not value each other's competence very highly in this regard. The use of Reagenix in blue projects was perhaps an exception, but otherwise the two groups of actors had completely different viewpoints. This is clearly visible in the two intertwined stories included in Appendix 3, but also in the extracts of the other case data shown in Appendix 2.

5 DISCUSSION

In the following we are discussing how well the modified competence evolution framework manages to explain the development of the code generation competence. We will also compare the code generation case with an earlier case study on industrial fault diagnosis systems [Seppänen et al. 1998a,b], by evaluating the competencies and relationships involved in the two cases. We will close the report by summarising the main empirical, research and managerial implications of this study.

5.1 ANALYSIS OF THE COMPETENCE EVOLUTION

The purpose of the competence evolution framework is to show and explain the development of competence, i.e. the changes of activities in creating and making use of resources, in terms of the development focal nets. The activities pursued by certain actors are means of implementing their logic of action, when aiming at reaching goals that fulfil specific objectives.

Various kinds of competence charts have been proposed for illustrating the integration of firm capabilities. [Klein et al. 1998] is a recent example of such charts, involving "clusters of R&D skills". Klein, Gee and Jones are proposing skills consisting of technical, processual and human aspects, which correspond to the product, process and organisation-related competence elements of this research. Their "company skill maps" are used for matching process-related competence elements to product-related elements in different organisations. An evaluation of skill levels is also included. Mutually supportive skills can be associated with each other, so as to form "skill cluster diagrams" representing, according to the authors, the core competence of the organisation. Although we have borrowed the idea of competence charts from Klein and the others, we have modified the charts to serve our needs of indicating the evolution of competence and its integration with the competencies of the most influential groups of external actors. The charts will also show the logic of the actors and characterise the process nets within which the logic have been carried out. The code generation competence evolution chart is presented in Table 23, with the elements that have strengthened each other shown in *italics*.

During the Speco period, a vision of code generation was developed and tested. The vision was, however, mostly created by firm K. VTT implemented the vision as part of its project business logic, and the customer company bought a technical solution to implement the vision, a prototype of a code generator tool. The generator was implemented by using the programming language Prolog that was familiar to both VTT and firm K. However, the opinions on technical matters of the customer and the focal researchers started to differ, not to speak of the process of innovative research. The functional skills of the researchers were related to compilation of program languages, to managing concurrence, as they explained in the interview, whereas the customer was familiar with the AI reasoning functions. The kinds of automation systems that firm K developed were considered key embedded systems applications by both parties.

Table 23. Code generation competence chart 1985 - 1998.

Actors/ Logic	**Product-related elements**	**Process-related elements**	**Organisational elements**
Speco - Testing of an industrial vision			
VTT: project business	Design elements Applications: (automation) Functions: compilation Techniques: *programming* Technologies: *Prolog*	*Innovative research* Project marketing and implementation	Strategic alliance *R&D groups* Domestic reputation *Utilisation of Tekes* Technological skills
Firm K: solution buying	Design elements Applications: *automation* Functions: AI reasoning Techniques: *programming* Technologies: *Prolog*	*Development of an* innovative integrated embedded *software engineering environment*	Company vision *R&D groups* Corporate research *Utilisation of Tekes* Automation business
Sokrates - Demonstration of a research vision			
VTT: pio-neering research	Design elements Applications: *automation* Functions: code generation Techniques: Sokrates-SA Technologies: generator Solution elements Technologies: SIC, ReagOS	*Technology development* *Applied research* in blue (red) projects Training of Sokrates-SA Technology popularising	Managerial role in the Finsoft program *Utilisation of Tekes*
Firm Nm: solution buying	Design elements Applications: *automation*	*Product development*	Machine automation business
VTT/Ele: faster solutions	Design elements Applications: *automation*, VLSI design	*Applied research and development* in blue and red projects	*Utilisation of Tekes* and industry R&D groups
Reagenix - Fighting for the vision			
VTT: product business	Design elements Techniques: (Sokrates-SA) Technologies: generator, testing tool, animator Solution elements (cf. above)	*Applied research and development* in blue and red projects	*Utilisation of Tekes,* industry and other VTT institutes *R&D groups*
Blue projects: cheaper solutions	(Project-specific competencies)	Use of project resources	*Utilisation of Tekes* and industrial contacts
Firm Nm: paradigm buying	(Cf. above)	(Cf. above)	(Cf. above)
Firm R: -"-	Design elements Applications: electronics	*Product development*	Consumer electronics business
Firm S: solution buying	Design elements Applications: electronics Functions: software design	*Product development* In-house design of software tools	Consumer electronics business *(R&D groups)*
Firm N: -"-	Solution elements Technologies: embedded software and hardware	*Product development* In-house design of software tools	Telecommunication business *R&D groups*
Firm Kc: -"-	Design elements Applications: automation	Product development *In-house tool design*	Machine automation business

The state of the outer context, with increasing industrial interest and active planning of joint national software engineering research, favoured the utilisation of the organisational and project planning skills of TKO for making use of the technical results of Speco and Draco in marketing Sokrates. The role of firm K cannot be neglected either, as its corporate research unit, for which the Speco project was carried out, was a forerunner in embedded software engineering and had close contacts with Tekes. Although the funding organisation was in a critical position regarding financial resources available to applied engineering research, its role in technology development and management was rather insignificant. This was paving a way for the autocracy of the focal researchers in Sokrates, where they could follow their own logic of action and had all the necessary technical, human and financial resources to make their own vision true.

The core concepts and implementations of code generation design elements were indeed produced during Sokrates, in addition to a few unforeseen solution elements. A remarkable difference with regard to the results of Speco was that these elements were more generic. Although the Kaapeli and Synchro projects in which the results were applied for the first time involved automation applications, the Sokrates-SA system design technique, the code generation functions and the implementations of the code generator and solution elements were application-independent. The understanding of the focal researchers on automation applications helped them to co-operate with firm Nm and the Electronics laboratory. VLSI design was new to them as an application, and after the Sasic project the Electronics laboratory carried the idea of code generation further with the University of Oulu, without any involvement of the focal researchers. They claimed in that VTT advertised the results of this co-operation more than the Sokrates results.

The Sokrates-SA method was the dominant product-related competence element of Sokrates, but the functional code generation skills also appeared as a useful purchase for firm Nm in the Kaapeli project. The generator itself was still immature. The opportunity window for producing an industrially usable generator had been closed already. Firm N had such a generator, a very simple one. Firm I had apparently already started developing its own generator, although it did not tell this to any external parties. The focal researchers and VTT managers could not launch any red projects involving generator development, although the possibility of Tekes-supported spin-off projects was actively marketed to the Sokrates steering group. This was a rather fatal failure, since such a work still needed to be done after Sokrates, but in a much more complex and financially limited setting. According to the researchers, some firms had "learnt too much" and did not need VTT any longer for developing their own generator solutions.

The process-related competence was considered good by the focal researchers during Sokrates, due to their emphasis on innovative, result-driven and pragmatic research rather than project management and organisational planning. Some other parties were complaining about the lack of managerial competence. Yet, the competence was good enough for carrying out the Sokrates project, and it was supported both by the key managerial role of TKO in the Finsoft program and the excellent training and technology popularising skills of the focal researchers.

While Sokrates was kind of golden time for developing code generation competence, Reagenix brought the secure focal net harshly down. The promise of extensive exploitation was, from the viewpoint of the VTT managers, rapidly fading away. The self-controlled huge project resource pool and the central position in the large Finsoft program were no longer available to the researchers. The large Sokrates steering group had dissolved and regular interaction with parties which had been interested in code generation for the past three years, was finished. Most importantly, it was now much more difficult for the VTT managers and the researchers to unite their forces to market the Sokrates results than it had been to launch the Sokrates project. The managers pursued the goal of initiating large enough red projects, whereas the researchers started to realise that the code generator needed to be improved, if not developed anew, without too much resources or external support, and facing competition from firm I.

It seems that the increasing non-alignment of the logic of action of the researchers and managers had evolved little by little. The managerial problems of the Speco project and the dissolution of the R Group may have left some tension between the parties, but as the managers pointed out in the interviews, they actually believed in Sokrates more than some of the colleagues of the focal researchers did. The idea of starting product business at VTT was tolerated by the managers, who even defended VTT's interests towards firm I on a few occasions. However, after the financial outcome of the business turned out to be just modest, their interest in the matter collapsed. The focal researchers thought that the managers were utilising customers rather than supporting efficient technology transfer. At the group rehearsal, they were strongly criticising the managers' distrust on the capability of VTT to develop useful software engineering tools for industry. The managers were claiming that the researchers themselves had missed the tool development opportunity during the Sokrates project already.

The diversity and small size of red projects during the Reagenix period is apparent also in Table 23. Some customers were interested only in certain solutions, others in more comprehensive offerings. Firms S and N, which had developed software design tools for internal use, did not, after all, make use of the design elements related to code generation. The role of SA/SD was decreasing, which according to the interviewed Sokrates steering group members affected their interest in using the results. The functional code generation skills remained hidden inside the generator, which made their exploitation difficult without the tool technology. Blue research projects seemed to offer alternative networks for exploitation, and the corresponding experiences from Sokrates were encouraging. However, the projects became a sort of extra front of colourless relationships, where the researchers spent their time without any considerable rewards. Although this kept Reagenix alive, it did not help either in the project sense or product business sense to improve the organisational status and visibility of code generation. The same holds for licenses that were sold or given for free to external parties. After many of the focal researchers had left VTT, the effective informal network that had helped to make use of Reagenix in blue projects weakened. Selling the code generation technology rights to the researchers was obviously an appropriate move, carried out at the last minute.

5.2 CROSS-CASE ANALYSIS

An earlier case study reported in [Seppänen et al. 1998a,b] involved the same focal organisation, but mostly a different group of actors and another area of competence, which was industrial fault diagnosis systems. The framework that was used to make explicit the evolution of fault diagnosis competence and relationships did not yet include the concepts of logic of action and processes applied in this research. We will therefore discuss them briefly in the context of the fault diagnosis case, as depicted in Table 24. The two cases are then compared from the viewpoint of competence evolution, based on the alignment of the logic of action of different parties and the management of the relationships of the focal organisation. The evolution of code generation competence has been summarised from the viewpoint of relationship management in Table 25.

Table 24. Evolution of fault diagnosis competence and relationships.

Period	**Competence elements**	**Dominant processes**	**Actors/Logic of action**	**Process nets**
1986 - 88 *Dawn of intelligence*	Product: AI and KE techniques	Green research	VTT/Establish a new research area at TKO	Mostly informal actor bonds
1989 - 91 *Demon-stration of knowledge-based systems*	Product: KE techniques, tool technologies Process: system design skills Organisational: knowledge engineers	Marketing of the promises of KE applications Blue and red projects: design of system prototypes for industrial trial	Process and machine automation firms and VTT/Improve the skills of knowledge engineers	Innovative project nets Informal interaction of knowledge engineering experts
1992 - 94 *Develop-ment of reliable automated products and processes*	Product: fault diagnosis functions, automation applications, KE techniques Process: system design skills Organisational: R&D projects	Marketing of fault diagnosis solutions Red and blue projects: development of diagnosis functions for automation applications	Machine automation firms/Improve reliability of products and processes VTT/develop intelligent fault diagnosis systems	Product and process development project nets Informal actor bonds between diagnosis and automation experts
1995 - 97 *Extending of the bridgehead*	Product: fault diagnosis platform, KE techniques Process: fault diagnosis related problem solving Organisational: references, R&D projects	Marketing of a core platform Analysis and capture of new R&D markets Green, blue and red projects: development and use of the core platform	Automation and telecom firms/ Manage complex systems better Funding bodies/ Support R&D of intelligent systems VTT/open up new application areas	Industrially-driven R&D project nets Informal actor bonds between diagnosis and application experts (and end-users)

Table 25. Management of code generation relationships.

Competence evolution	**Management of relationships**
Speco 1985 - 1987	
Testing of an industrial vision Innovative research results based on a shared vision with a key customer were successfully used for marketing and planning joint technology development as part of a strategic national initiative. A popular embedded system design method and a domestic CASE tool were used as a framework for the proposed approach. A core team was created. The Speco code generator was not utilised, but SA/SD was "found".	The aligned logic of action of the VTT managers and focal researchers created pivotal managerial and research positions in Finsoft. The Draco project had confirmed the need of an alternative pragmatic approach, which could be based on the SA/SD. Tekes let the researchers take the lead in Finsoft, because the focus was on generic techniques and not on industrial applications. The dispute with firm K was considered mostly a managerial problem.
Sokrates 1988 - 1991	
Demonstration of a research vision Code generation principles from Sokrates-SA to C were formalised and tested with firm Nm. Training of the improved method was arranged and associated system solutions produced. In-house exploitation of the generator was arranged to demonstrate the promises of the vision, no links to the code generation solutions of firms N and I realised. The core team was extended. Although the Sokrates code generator was not industrially utilised, Sokrates-SA emerged as a recognised design method.	The logic of action of the focal researchers guided the technology development work, because they had the technical skills and all the project resources at their disposal. Tekes, VTT managers and industry waited for the vision to realise, which was considered likely because of the early starting of the results exploitation. The managerial and scientific problems were not considered serious by the VTT managers, Tekes and steering group members. The final evaluation results were excellent, and the dispute with firm I was not made public. The Sokrates project manager became a line manager, which helped the researchers. The effects of the recession were largely neglected, though.
Reagenix 1992 - 1998	
Fighting for the vision Revised implementation of the design technology were made available with low costs and free exploitation support in a professional manner, despite the collapse of financial resources and organisational interest. Associated system solutions could be utilised rather successfully. The former Sokrates team members could help each others despite the intervention of the managers.	The financial foundation of the Sokrates network collapsed, before a well-engineered implementation of the code generator was available. Such an implementation could yet be produced in a very short time by the focal researchers. The logic of action of the managers and researchers were at first aligned well enough to boost the exploitation of Reagenix both in blue and red projects. Yet, this did not provide enough income from the managerial viewpoint, and green research did not improve the situation.

The idea of code generation as a practical means of helping professional embedded software engineers emerged during the Speco period. VTT, which had formed a strategic alliance concerning the development of an advanced software engineering environment with firm K, could use the technical results of the Speco and Draco projects and its organisational skills for creating a pivotal position in Finsoft. Tekes and industry bought the promise of code generation marketed by the VTT managers and researchers. The other parties let the researchers take the lead, because Finsoft addressed generic techniques rather than industrial applications.

The researchers had already formed a core problem solving team and were applying their radically new technical ideas to a well-known domestic CASE tool and the popular SA/SD method. This facilitated the acceptance of the idea, as did also the experiences reported on the Speco (it is possible to develop practical code generation technologies) and Draco (existing solutions are not practical enough or even working in embedded systems applications) projects. The code generator designed in Speco was thrown away to start the development again, this time based on the researchers' own vision and solutions. Even firm K had trust on the researchers, although it had been complaining about managerial problems in the Speco project.

The starting of fault diagnosis related R&D activities at TKO, characterised as the "Dawn of intelligence" in Table 24, bears some resemblance with the Speco period. In both cases, a new research area was opened by scanning existing approaches and solutions. However, the vision of the role of fault diagnosis systems in industry was very weak, if explicit at all, compared with the Speco and Sokrates visions. The vision was gradually created during 1989 - 1991, together with a few industrial knowledge engineers. A small team of VTT and industrial researchers started focusing on the so-called embedded expert systems at that time, which closely corresponds to the co-operation between firm K and TKO during Speco.

The industrial people involved in the early phase of fault diagnosis R&D were knowledge engineers, who wished to strengthen their own position and to improve their skills in applying knowledge engineering techniques. This resembles the relationships between the focal researchers and the representatives of firm K in Speco. The early fault diagnosis R&D results were considered demonstrations, rather than meant for daily industrial use. This was the status of the Speco results, too, and it had apparently also been the aim of the project. In both cases, generic techniques and quite novel technologies were developed and applied to solving problems in particular applications. This kind of marketing and demonstration of the promises of new technologies for specific customers in the context of some emerging or already established engineering techniques is one of the core activities in contractual engineering R&D.

In both cases, the logic of the two key actor groups of the focal organisation, the managers and the researchers, were well aligned. They were marketing and exploiting the initial customer projects together, so as to pave the way for subsequent competence elaboration and extension. In code generation, one big blue project was established, while in fault diagnosis several smaller red projects emerged.

During the Sokrates period, the focal researchers were enjoying a great deal of autonomy from the other parties thanks to their technical skills and the huge project resources. They were controlling the extension of the core problem solving team by themselves, hiring mostly newcomers, who from the viewpoint of VTT colleagues started to think and behave "the Sokrates way". This kind of human code generation skills management left a lot in the hands of the researchers. For example, the managers did not use their organisational authority for allocating the former code generation expert of firm N to the Sokrates team. From the managerial viewpoint, this could have greatly helped to exploit the firm as one of the key customers of TKO, who had developed similar solutions. The coming rapid growth of the firm's business sector, telecommunications, was also not foreseen. Since the exploitation of the Sokrates results at firm Nm and the VTT Electronics laboratory had started, the promise of industrially useful results seemed to realise anyhow.

Compared with the code generation case, in fault diagnosis the technology research and demonstration phase was much shorter and more fragmented. One of the critical differences can be found in the fact that fault diagnosis, as opposed to code generation, soon drew away from the generic AI techniques and special tool technologies upon which it had originally focused. By addressing fault diagnosis as a system design problem rather than an AI application, the VTT researchers could make use of a broader set of embedded systems design techniques and tools, which made the basic principles of fault diagnosis understandable to customers. This was necessary, as fault diagnosis functions were incorporated into industrial systems and processes designed and maintained by other people than knowledge engineers. The portfolio of automation applications helped the researchers to increase their application skills and understand the technical experts on the customer side. This focus was kept for a rather long time, by directing project marketing activities to the automation sector and using past projects as references. Despite, or perhaps due to, the recession, automation firms were investing in reliable products and processes, which created room for fault diagnosis systems. Although the former relationships of VTT with industrial knowledge engineers were allowed to break up, new relationships with industrial automation engineers and end users could be created.

The VTT managers and knowledge engineering researchers shared the goal of increasing red fault diagnosis projects, while the reliability requirements of the users of automated machines and processes were increasing the pressure on product manufacturers. The vision of knowledge-based fault diagnosis and its implementation could thus be carried further without any considerable disagreements within VTT and in its customer relationships. Firm K, which was disappointed with the results of a red fault diagnosis project for which it had contracted with VTT, was an unusual exception. In other projects, the reuse of fault diagnosis functions, the use of known techniques for modelling automation applications along with a variety of implementation technologies of the customers helped VTT to create sustainable portfolio-specific competence and allowed it to remain quite flexible at the same time.

This resulted in the evolution of a core platform, around the functional product-related competence dimension. It opened up both opportunities for capturing new markets and applying new generic techniques. A problem solving process also emerged, which tied product-related competence elements together.

Sokrates also resulted in portfolio-specific design competence. The Sokrates-SA method was well received by industrial professionals, and the generic system solutions provided with unexpectedly good spin-offs. Although the system solutions and the Sokrates-SA method could be exploited very much as such during Sokrates, they provided only little and sporadic income for the focal organisation. The generator that had been developed was, again, thrown away, and a new one was produced in a few months. Yet, the use of the Reagenix account that was thought to give the researchers financial freedom to carry out their business logic based on the new generator product, did not succeed as planned. The relationship with firm I broke up, and no alternative tool vendor partners could be found. The managers lost their interest in ensuring financial resources for further applications and development of Reagenix, while they were extensively involved in expanding the markets of fault diagnosis systems. The core fault diagnosis platform made it possible to direct competence marketing to the telecommunication sector, as soon as its enormous increase became apparent. No research customers were sought for, although they would have been available.

A national research program of Tekes was utilised to help to create the first telecommunication fault diagnosis project, but so that Tekes funded a few customer companies which contracted a red project from VTT. This helped Tekes to indicate the usefulness of its funding with regard to transferring new technologies to industry. The fault diagnosis skills were solely owned by VTT, but the companies took care of project management. Problems similar to Sokrates arise, the case application was complex and difficult to model, the first results remained a demonstration system and difficulties were encountered with some techniques chosen to be used. Yet, the first large red project was followed by two new projects. One of them was carried out for one of the initial customers, the other for a new customer. An extension of this bridgehead was supported by a rather large green project funded by VTT. In 1998, a new marketing manoeuvre was performed among instrument manufacturers, one of them getting involved in a green project, and the first red fault diagnosis project for this customer was initiated. The management of competence evolution through reuse and renewal of the core platform, associated with the creation of customer relationships in new application areas thus seemed to work, if the application knowledge could be modelled and no major mistakes were made in developing or taking in use the new generic engineering techniques and implementation technologies.

There were some product-based business opportunities involved also in the fault diagnosis case. For example, a distinct tool was developed for explaining faults detected by diagnosis systems. These opportunities have not yet been used, and some of them seem to have been lost due to their use in red projects.

At least one customer has sold the rights of a distinct fault diagnosis competence element to a third party without any consultation with VTT. Although this is completely correct in legal terms, it indicates that VTT has not been able to manage all the aspects involved in its competence. If competence elements are packaged well enough, the purchasers will re-sell them to other parties. This is possible, when a customer becomes the owner of project results based on reuse of explicitly codified competence elements. From the R&D supplier's viewpoint this involves the problem of "satisficing" codification and protection of its core competence.

Interestingly enough, as mentioned in passing above, the developments in the outer context during the period 1990 - 1995 apparently affected the two cases in opposite ways. The recession that considerably decreased the role of the automation sector as a potential application field for embedded software engineering techniques, seems to have boosted the need for developing more reliable products. This helped VTT to create portfolio-specific fault diagnosis competence with automation firms. The buyers and end users of expensive automated machines and processes wished to make use of their investments efficiently, and the hard times forced machine manufacturers to listen carefully to their customers.

Since many manufacturers did not have the capability of building advanced computerised fault diagnosis systems themselves, they were willing to use external experts. VTT researchers had developed an approach to fault diagnosis, in which engineering models familiar to manufacturers could be used. Intelligent techniques were applied in developing effective fault diagnosis functions and any technologies favoured by the customers. Relationships could be established where the interests and competencies of different parties matched. Firm K, which was itself competent in fault diagnosis, had needs for which the competence of VTT was unable to provide any solution - which was apparently the case also in the Speco and Sokrates projects.

If the code generation competence had been matched with the developments of the outer context, it would have been possible to make explicit the functional competence elements and to associate these with the kinds of new system design techniques that were emerging especially in the telecommunication sector. Code generation research started from a rather technological origin, which is understandable when considering the background of the key researchers. Most of the early embedded systems applications involved "low-level" device control software, but in the nineties the focus was directed to "higher-level" applications and software architectures providing some added value to the basic device control. System design techniques and tools should therefore help to develop and reuse software architectures, not only the kind of low-level device control software embedded, for example, in the toy elevator built in Sokrates.

Figure 9 illustrates, how code generation competence could have been packaged into a core platform. Since the target software had evolved towards technical, functional and application platforms built one on top of another, the functional code generation competence could have been associated with different design techniques and implementation technologies used at different platform levels.

For example, the device control platform of a digital telecommunication product involves digital signal processing software, the functional platform communication protocol software, and the application platform user interface software. The researchers concluded in the group rehearsal that software architecture concepts created during Sokrates were not utilised well enough. Focusing on a software platform architecture might have led to the creation of relationships with different customers or different experts of the same customer would have been required, similarly to the development of fault diagnosis systems at ELE.

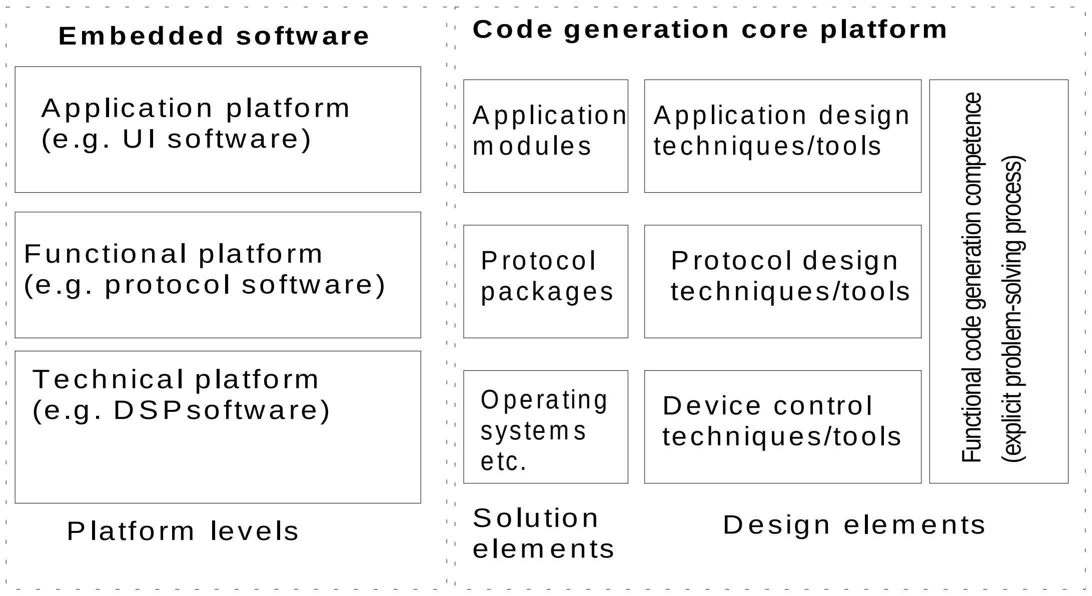
Embedded software
Application platform (e.g. UI software)
Functional platform (e.g. protocol software)
Technical platform (e.g. DSPsoftware)
Platform levels
Code generation core platform
Application modules
Application design techniques/tools
Protocol packages
Protocol design techniques/tools
Operating systems etc.
Device control techniques/tools
Functional code generation competence (explicit problem-solving process)
Solution elements
Design elements

Figure 9. Embedded software platforms vs. code generation platform.

5.3 LESSONS LEARNED

The empirical, research and managerial implications of our research are discussed in the following, focusing on lessons learned for subsequent investigations of competence in the context of industrial networks.

5.3.1 Empirical implications

We found the role of secondary data quite crucial in our study. Although secondary data may be overwhelming in longitudinal studies, digging deeply into it seems to pay off. In this research, the data played a central role in making explicit both the process nets and processes involved in the code generation case. We used the competence evolution framework to structure the data into meaningful chunks during three successive periods of time between 1985 and 1998. Yet, it was rather difficult to prevent the secondary data from becoming too fragmented, because of all the details included in the documents from which the data was acquired. For this reason, the data was organised around the five key processes of competence: marketing, purchasing, development, exploitation and mutual co-ordination. This helped much to follow the evolution of competence.

During the research, it became obvious than an important piece of secondary data was missing - information related to the outer context of the case phenomenon, not only from such a limited perspective as the documents describing the Finsoft research program that we had included in the data sources, but from the viewpoint of environmental macro forces affecting industries and markets as a whole. We decided to acquire the required data on the outer context, which was actually used by the managers for the purpose of positioning VTT in the changing environment.

This proved a rather good approach, because we managed to identify certain pieces of industry-level information that had remained unnoticed and, therefore, could not be considered in guiding the activities of the focal organisation. One of the best examples was the strong bias shown by TKO towards automation applications in embedded software engineering at the turn of the nineties, when the existing R&D potential of the sector was quite modest and rapidly decreasing, and the telecommunication sector was about to start its immense growth. The true potential of internal research customers was apparently similarly misjudged.

The secondary data does not include too much information on why certain actors took part in certain nets and were involved in certain activities. Although some of the key expectations and perceived values of the activities and their results were summarised in managerial documents, no data on the longitudinal or historical effects of the code generation competence was available. The evaluation of resources used in or produced by R&D activities is a topic constantly under debate, while almost invariably only the situational values are addressed. Various kinds of frameworks and criteria have been proposed for measuring the R&D process and its outcome, for example [Cooper and Kleinschmidt 1996, Roberts 1995, Schumann et al. 1995, and Tipping et al. 1995]. Schumann and others have specified a set of “total quality measurements” for R&D, Tipping and others a comprehensive “technology value pyramid”. [Werner and Shouder 1997] provide an exhaustive survey of the R&D measurement literature from 1956 to 1995.

From another perspective, [Holmlund 1997] proposes technical, social and economic dimensions for relationship quality. The content of the technical dimension of relationship quality proposed by Holmlund includes generic process types and characteristics, as well as technical outcome. The social dimension of quality includes the individual level and the company level and the economic dimension the benefits and costs of the relationship.

No distinction is made between the expected, perceived and historical values of relationships by Holmlund. The secondary code generation case data included information on the expected and perceived values of the competence, but we captured the historical value of the competence from the primary data. The primary and secondary data included a number of interesting and even dramatic differences in this regard.

Parts of the primary data were compiled into intertwined stories of the two main groups of actors of VTT. The stories provided both a summary of the case abstract enough for readers not familiar with the subject matter and an exciting history of the developments affecting the participants, which has never been put together earlier. Compared with focused interviews with the individuals involved in specific activities, such as customers of certain projects, the grand stories open up a considerably more comprehensive perspective to the case as a whole. The conflicting viewpoints of the managers and focal researchers became very obvious. Several parts of the original story written by Veikko Seppänen were pointed out as "wrong", misleading or incomplete by the researchers.

The help of information technology, especially electronic mail, was surprisingly beneficial in the process of gathering primary data. The interviewees were very open and willing to clarify their opinions. Some of them had been waiting for this opportunity in vain, which shows how little attention is paid to the past even in expert organisations aiming at transferring in use knowledge providing long-term value for their customers. We deliberately targeted the bulk of interviews on purpose at persons who had carried out other than only managerial tasks.

Although this focus certainly depends in part on the technical nature of the case, we wish to emphasise that business is not only what managers vaguely remember to have happened, but what their subordinates have personally done and experienced. This was important also because of the fact that very few informants used the opportunity of commenting or correcting the data that acquired and documented by us.

As opposed to using general managerial data, it is much easier to triangulate detailed data given by people who have carried out certain activities themselves. The size of the interview data of the code generation researchers was about ten times larger than the data provided by the two focal VTT managers. The data that individual Sokrates steering group members provided was typically less than one page.

5.3.2 Research implications

This research addressed the development of competence in one focal organisation as a result of its external relationships (and, to a lesser degree, due to its internal activities). Therefore, we wanted to use the existing research results of industrial networks as a means of explanation rather than as the subject of the research. Accordingly, we focused on testing and applying the relationship-based competence evolution framework proposed in [Seppänen et al. 1998a]. Yet, the framework needed to be extended by the concept logic of action, and to be associated with the network concepts of actors, resources and activities. Although investigations of industrial networks have utilised research methods familiar from subjective social science and organisational studies, such as unstructured personal interviews, they seem to have taken quite an objective and rational viewpoint to collective actors.

We chose the logic of action as a means of making a difference between groups of actors when associating them with relationships, not least because one of us, Päivi Eriksson, had used it successfully in her earlier research. It appeared to be a useful way to describe and explain the activities that were pursued by certain groups of actors within the context of certain process nets. For example, the basic dilemma between product-based and project-based exploitation of code generation competence could be made explicit and analysed by using the logic of action of two actor groups. Yet, further research is inevitably needed to gain a better understanding of the behaviour of organisational groups of actors in complex network settings. The interaction of "sustaining" and "radical" logic of action would make a particularly interesting to research topic. As an example, in the group rehearsal one of the focal researchers compared the aim of using code generators in software development to the idea of taking photographs instead of painting portraits at the end of the last century.

One of the great challenges in behavioural studies of network actors is that the focus may span from individual influential persons, such as the manager of a specific project, to whole organisations, industries and even nations. For example, the "logic of Finland" caused the recession, which had a strong influence on the developments visible in the code generation case. Tiittula [1994] uses the concept of "managerial logic" when analysing organisational changes at VTT as a whole. Consequently, this kind of extensive intertwining of inner and outer contexts can be regarded as an essential issue in real-life industrial networks.

Yet, a more demanding task than the use of the logic of action was the use of the competence evolution framework as a whole. Although the network-based approach to industrial marketing is deeply rooted in the resource-based perspective to firms, there are only a few in-depth studies on the role of resources in networks. As an example, the need to define the change of the value of a resource over time became obvious during this research, while there was little guidance available on how to treat this issue. Yet, we did not concentrate as much on the change of the value of resources than on the processes that caused such changes. We used [Tikkanen 1997] as a starting point in the process perspective on networks, while we still had to elaborate his approach considerably.

In our case the well-archived secondary data, fortunately, made it rather easy to follow the processes during the past years. As indicated by Tikkanen, this may be much more difficult in cases in which such data does not exist, e.g. due to the fact that the relationships are more informal and sporadic in their nature, in contrast to projects lasting for months or years. Despite possible difficulties, we would like to urge network researchers to keep a closer look on processes, because otherwise long-term phenomena may become trivially classified into activities, without the longitudinal "process flow" becoming explicit. Studying this flow is essential when analysing the true evolution of relationships against marketing plans and strategies. For example, the success of the strategic management of competence evolution at VTT through the portfolio of green, blue and red projects can only be evaluated if competence related processes can be identified from projects.

While strategic marketing used to be a popular school of thought in the seventies, it seems that later network researchers have been focusing more on individual firms than on markets as macroeconomic entities. It is therefore noteworthy that our view on product-related competence has been derived from the strategic marketing concept of [Abell 1980]. In other words, we are viewing the competence of a firm against a strategic market structure. The "techniques" dimension that we have added to Abell's market model is justified by the engineering and scientific foundation of the services that research organisations are offering to the market. We have also used another key concept of strategic marketing, the life cycle, and associated it with competence elements. The usefulness of the life cycle concept is illustrated, for example, by the need of making explicit the simultaneous rise of telecommunication applications and the fall of structured system design techniques in the nineties, which greatly affected the code generation case.

5.3.3 Managerial implications

The use of the market as a mirror of the content of product-related competence leads to one of the main managerial implications of this research: Strategic management of competence must be carried out *within* the context of external relationships, not *for* the relationships. Therefore, the strategic management of the relationship portfolio of a firm is actually its main competence management tool. Although self-funded green projects are an important asset for VTT, the true field test of competence evolution are red projects. The code generation case shows that market mechanisms greatly affect the passing of such a test. To put it simply, the focal researchers considered the market that of domestic CASE tools, but failed to carry out strategic competence management at VTT after the Sokrates project had been finished. The VTT managers considered the market as consisting of professional R&D services, and basically gave up the competence after the promise of red projects was not fulfilled.

Yet, both the managers and researchers seemed to have misinterpreted or neglected the coming industrial changes. Automation and electronic instruments were rapidly losing their customer potential, whereas telecommunication started to emerge as a killer application. Regarding this strategic market change, the decision to allocate the code generation expert of firm N who joined VTT to a completely different network of relationships in the middle of the opportunity window was a failure.

In more general terms, the versatility of code generation competence elements may have resulted in losing sight of functional skills, which was identified as a fundamental competence dimension in [Seppänen et al. 1998a]. In the code generation case, such skills involved knowledge of how to systematically produce computer programs from higher-level design models. These skills were hidden within the code generator technology, which was constantly changing and difficult to understand by the likely customers. Although this kind of mechanising of skills was apparently one of the key ideas of the focal researchers, the customers were not ready for it.

The researchers claimed that the initial grand story written by Veikko Seppänen was missing the point of functional skills of managing concurrence in embedded systems, describing the work "just as the development of yet another CASE tool". This was, however, exactly the opinion of at least one of the key customers interviewed; he regarded the focal researchers' claim of focusing on concurrence management as harshly as "bullshit".

The Kaapeli project, in which the functional skills were made explicit through manual coding rules without any tool technology, was quite successful also in historical evaluation. The MCS-REA project, where the skills were implemented in connection with the Reagenix code generator, was afterwards considered a business catastrophe by the customer, although the perceived value of the technology was seen as excellent.

Although the SA/SD skills helped to create and to extend the code generation network during the Sokrates period, they did not offer any foundation for strategic long-term management of code generation competence. Instead, the skills were lent core rigidity [Leonard-Barton 1992] that prevented a timely response to the change in design techniques.

Many people were trying to point out this to the focal researchers, but their first love was too strong to be replaced until the late nineties. In the group rehearsal, most of the researchers were still of the opinion that SA/SD was the right choice, and the same was indicated by various other interviewed parties. However, interestingly enough, one of the focal researchers told us, as his present opinion, that SA/SD was *not* the right choice.

The relationships between a research institute developing engineering methods and tools and a commercial tool vendor making business out of them appeared rather problematic in this research. The whole idea of the Finsoft research program was to study and transfer into industrial use new software development methods and tools. However, the co-operation between researchers and industry wishing not only to apply them as end users but also to make business out of the results was obviously not addressed well enough. TKO and firm I tried to clear up their relationship by themselves without the help of any common procedures, but did not succeed. The matter was still sensitive when the study was carried out, as the representatives of firm I told us that they did not want to think back to the past events related to the case. For this reason, they turned down the inquiry to interview them as a part of this research. All the other parties were willing to participate and were quite open in the interviews, even the VTT managers and focal researchers that had clearly conflicting viewpoints regarding each others' goals, activities and work-products.

The focal researchers were quite united in their view that the VTT managers were preventing them from following their logic of action and making business by selling tool licenses. This was obviously true in the sense that the whole portfolio of R&D activities at TKO and ELE was based on the logic of creating red projects from green research via blue projects, instead of focusing on collecting license fees. The researchers were not prevented from following their logic, but not much supported either.

Since all the organisational resources were controlled by the managers, the researchers were doomed to fail in their attempt. The researchers did not see the project portfolio as the basic organisational means of developing the competence further. One of the best examples of this is that Veikko Seppänen, as a lower-level manager who was most closely involved in the code generation activities, thought honestly in his original story that Reagenix was redesigned on the basis of the Sokrates code generator, following the project portfolio principle. The focal researchers reacted strongly against this view: "What was worst in this story was the indication that the Reagenix code generator was the same as Sokrates. My opinion is that Reagenix has been developing slowly here and there, and there is no clear link between Sokrates and Reagenix".

It is also interesting that the whole idea of project marketing seemed to be rather distant, if not terrifying, to the focal researchers: "I did not yet have any appropriate contacts and I had not received any positive feedback on my work [on Reagenix], so I was not interested in marketing". Many of their colleagues were continuously co-operating with the managers to create contacts, market projects and to share successes and failures in managing evolving competence. The opinion of the focal researchers that the VTT managers only wanted to milk customers - by selling work as projects with no unified frame, some projects addressing "parts of Lada" and others "parts of Porsche" - is radically different from this kind of collaboration.

Despite the frustration and disappointments involved in the code generation case, as in many real-life phenomena, it is relieving to see that the code generation competence survived the difficulties. This must be honestly credited to the true inventors and owners of the competence, the focal researchers. Therefore, one of the managerial lessons that can be learnt from the case is that the commitment of people should be taken better care of for the benefit of competence, whether it is managed according to any logic of action or not. In this regard, the key problem of a contract research organisation is that it continuously has to balance between commitment and a sufficient level of income.

An appropriate closing to this report will be provided by a small software extract taken from Appendix 2, a piece of code generated with Reagenix shown in Figure 10. The program has, of course, been generated for a small demonstration application - this time it a simulation of a traffic light control system. After all, the light is now green also for the generator itself, which means "go" for the business, as well!

```
/* xtlight1.c - control_traffic_lights () - 1997-06-17  13:07:10 */
/*-----------------------------------------------
      Diagram Information
      Title   : control_traffic_lights
-------------------------------------------------
   ReaGeniX Programmer  Version  Version 2.Beta-01
   Licensed to: …
   ReaGeniX is a trademark of
   VTT Electronics, Finland
-----------------------------------------------*/

/*  reactime compatibility check  */
#define reactime_style 1
#include "reactime.h"
#if reactime_level < 1
  #error Old reactime version included
#endif

/*  subprocess compatibility check  */
#ifndef no_flowcheck
  #include "xtlight1.v"
#endif

/*  data definitions  */
#include "xtlight1.h"

process_body(control_traffic_lights)
  #ifdef flag_init
    initial_value_reservation(initial_value(flag),flag) =
        {flag_init};
  #else
    #ifndef flag_uninitialized
      #error Uninitialized state variable of type flag
    #endif
  #endif

  initialization
    init_process(sequence_lights,C2);
    init_process(register_pedestrian_request,C1);

  init_phase_2
    link_own_flow_from(C2.ped_grant, R__30ped_grant);
    link_own_flow_to(R__30ped_grant, C1.ped_grant);
    link_out_flow(C2.b_red, b_red);
    link_out_flow(C2.b_amber, b_amber);
    link_out_flow(C2.b_green, b_green);
    link_out_flow(C1.ped_wait, ped_wait);
    link_own_flow_from(C1.ped_request, R__56ped_request);
    link_own_flow_to(R__56ped_request, C2.ped_request);
    link_out_flow(C2.ped_green, ped_green);
    link_out_flow(C2.ped_red, ped_red);
    link_out_flow(C2.a_green, a_green);
    link_out_flow(C2.a_amber, a_amber);
    link_out_flow(C2.a_red, a_red);
    init_2_process(sequence_lights,C2);
    init_2_process(register_pedestrian_request,C1);
  end_initialization

…

/* xtlight1.c - control_traffic_lights () - 1997-06-17  13:07:10 */
```

Figure 10. Piece of code generated by Reagenix.

REFERENCES

Abell, D.A. 1980. Defining the Business. Prentice-Hall, Englewood Cliffs, New Jersey. 257 p.

Alajoutsijärvi, K. 1996. Rautainen pari. Kymmenen ja Valmetin suhde, lähiverkosto ja makrovoimat 1948 - 90. Jyväskylä Studies in Computer science, Economics and Statistics 31. University of Jyväskylä. 279 p. (in Finnish)

Anon. 1987a. Uuden teknologian tuotekehitys- ja tutkimuskeskus, perustamisselvitys. Teknologian diffuusio, TEDI-87. Projektiraportti n:o 1. City of Oulu. (in Finnish)

Anon. 1987b. Uusia keinoja teknologiayhteistyön tehostamiseksi Oulun seudulla. Teknologian diffuusio, TEDI-87. Projektiraportti n:o 2. City of Oulu. (in Finnish)

Anon. 1987c. Pohjois-Pohjanmaan elinkeinoelämän tietotekniset toiminta-edellytykset. Pohjois-Pohjanmaan Seutukaavaliitto. Publications Series A:84. 61 p. (in Finnish)

Anon. 1989. Oulun läänin teknologiapoliittinen ohjelma. Oulun lääninhallitus. 54 p. (in Finnish)

Anon. 1990a. Valtion tiede- ja teknologianeuvosto: katsaus 1990. Tiede- ja teknologiapolitiikan suuntaviivat 1990-luvulla. 76 p. (in Finnish)

Anon. 1990b. Teknologiaohjelmatoiminnan linjat 1990-luvulle. Komiteamietintö 1990:2. 112 p. + app. (in Finnish)

Anon. 1992a. Teollisuuspoliittinen linjaus. Seteli. 16 p. (in Finnish)

Anon. 1992b. Sähkö- ja elektroniikkateollisuus. Seteli. 7 p. (in Finnish)

Anon. 1993a. Teknologiakatsaus 1993. Tekes. 77 p. (in Finnish)

Anon. 1993b. Kansallinen teollisuusstrategia. Kauppa- ja teollisuusministeriön julkaisuja 1/1993. Ministry of Trade and Industry. 124 p. + app. (in Finnish)

Anon. 1994a. Suomi tietoyhteiskunnaksi – kansalliset linjaukset. Ministry of Financing. 73 p. (in Finnish)

Anon. 1994b. Osaamisstrategia. Sähkö- ja elektroniikkateollisuuden menestystekijät, avainteknologiat ja osaamisen kehitystarpeet. Seteli. 29 p. (in Finnish)

Anon. 1996. Teknologia 2000. Osaamisella tulevaisuuteen. Tekes. 120 p. (in Finnish)

Anon. 1998. Huippuosaajat maailmalla. Sähkö- ja elektroniikka-teollisuusliitto. 21 p. (in Finnish)

Cooper, R.G., Kleinschmidt, E.J. 1996. Winning businesses in product development: the critical success factors. Research & Technology Management, Vol. 39, No. 4, pp. 18 - 29.

Eriksson, P., Räsänen, K. 1998. The bitter and the sweet: evolving constellations of product mix management in a confectionary company. European Journal of Marketing, Vol. 32, No. 3/4, pp. 279 - 304.

Ford, D., Håkansson, H., Johansson, J. 1986. How do companies interact? Industrial Marketing and Purchasing, Vol. 1, No. 1, pp. 26 - 41.

Ford, D., Gadde, L.-E., Håkansson, H., Lundgren, A., Snehota, I., Turnbull, P., Wilson D. 1998. Managing business relationships. John Wiley & Sons, Chichester. 292 p.

Foss, N. (ed.) 1997. Resources, firms and strategies. Oxford University Press.

Foss, N., Knudsen, C. 1996. Towards a competence theory of the firm. Routledge, London.

Hamel, G., Prahalad, C.K. 1994. Competing for the future. Harvard Business School Press, London. 327 p.

Hienonen, R. 1997. Elektroniikka- ja sähköalan kehitysnäkymät 1997...2002. VTT Automation. 229 p. (in Finnish)

Holmlund, M. 1997. Perceived quality in business relationships. Publications of the Swedish School of Economics and Business Administration No. 66, Helsinki. 336 p.

Holmlund, M., Törnroos, J.-Å. 1997. What are relationships in business networks? Management Decision, Vol. 35, No. 4, pp. 304 - 309.

Håkansson, H., Snehota, I. 1995. Developing relationships in business networks. Routledge, London. 418 p.

Johanson, J., Mattson, L.-G. 1997. Network positions and strategic action - an analytical framework. In: Ford, D. (ed.), Understanding business markets: interaction, relationships and networks. Second edition. The Dryden Press, London. Pp. 176 - 193.

Karjalainen, J., (ed.) 1991. Finsoft. Ohjelmistoteknologian ohjelma. Sulautetut järjetelmät 1988 - 1991. Tekes, Helsinki. 167 p. (in Finnish)

Kivisaari, S., Lovio, R. 1993. Suomen elektroniikkateollisuuden merkittävien innovatiivisten liiketoimintojen menestyminen 1986 - 1992. VTT Technology Research Group. 51 p. (in Finnish)

Klein, J., Gee, D., Jones, H. 1998. Analysing clusters of skills in R&D - core competencies, metaphors, visualization, and the role of IT. R&D Management, Vol. 28, No. 1, pp. 37 - 42.

Klus, J.P., Markkula, M., Venho, J., Järvenpää, A., Sirkeinen, U., Ahlroos, R. 1985. Effective technology transfer. 128 p.

Kurki, M. 1995. Model-based fault diagnosis for mechatronic systems. Technical Research Centre of Finland, Espoo. VTT Publications 223. 116 p.

Leonard-Barton, D. 1992. Core capabilities and core rigidities: A paradox in managing new product development. Strategic Management Journal, Vol. 13, pp. 111 - 125.

Lovio, R. 1993. Evolution of firm communities in new industries. Acta Academiae Oeconomicae Helsingiensis. Series A:92. The Helsinki School of Economics and Business Administration. 304 p.

Lowendahl, B. 1997. Strategic Management of professional service firms, Handelshojskolens Forlag, Copenhagen.

Miettinen, R. 1993. Methodological issues of studying innovation-related networks. VTT Group for Technology Studies, Espoo. Working Papers No. 4. 61 p.

Miettinen, R. 1998. Object construction and networks in research work: the case of research on cellulose degrading enzymes. Social Studies of Science, Vol. 28.

Mårtenson, G., Otala, M., Wiio, O.A. 1985. Tietotekniikka 1990-luvulla. Sitra. Series B, No. 78. 112 p. (in Finnish)

Nokaka, I., Takeuchi, H. 1995. The knowledge-creating company: how Japanese companies create the dynamics of innovation. Oxford University Press, New York. 284 p.

Roberts, E.B. 1995. Benchmarking the strategic management of technology. Research & Technology Management, Vol. 38, No. 2, pp. 18 - 26.

Rosenbröijer, C.-J. 1998. Capability development in business networks. A study of distribution in the fine paper sector in the United Kingdom. Doctoral thesis, Publications of the Swedish School of Economics and Business Administration No. 69, Helsinki. 255 p.

Schumann, P.A. Jr., Ransley, D.L., Prestfood, C.L. 1995. Measuring R&D performance R&D. Research & Technology Management, Vol. 38, No. 3, pp. 45 - 54.

Seppänen, V., Isomursu, P., Kähkönen, A.-M., Oivo, M., Perunka, H., Pulli, P. 1996. Strategic needs and future trends of embedded software. Tekes Technology Review 48/96. 103 p.

Seppänen, V., Alajoutsijärvi, K., Kurki, M. 1998a. Competence-based evolution of contractual R&D relationships. Technical Research Centre of Finland, Espoo. VTT Publications 346. 70 p.

Seppänen, V., Kurki, M., Alajoutsijärvi, K. 1998b. Competence-based evolution of contractual R&D relationships. Proc. of Information Systems: Current Issues & Future Changes, Joint Working Conference of IFIP8.2 and IFIP8.6, Helsinki, December 10 - 13, 1998. IFIP, Laxenburg, Austria. Pp. 197 - 214.

Tiittula, P. 1994. Farewell to bureaucracy: Technical Research Centre of Finland as a pathfinder in management change. Acta Academiae Oeconomicae Helsingiensis. Series A:95. The Helsinki School of Economics and Business Administration. 247 p. + app.

Tikkanen, H. 1997. A network approach to industrial business processes. A theoretical and empirical analysis. Publications of the Turku School of Economics and Business Administration. Series A-7:1997. 227 p.

Tikkanen, H. 1998. Research on international project marketing. A review and implications. In: Tikkanen, H. (ed.), Marketing and International Business - Essays in Honours of Professor Karin Holstius on her 65th Birthday. Publications of the Turku School of Economics and Business Administration. Series A-2:1998. Pp. 262 - 285.

Tikkanen, H., Alajoutsijärvi, K. 1998. Customer satisfaction in industrial markets: contextual understanding as a key for effective management. Proc. of the Annual CBIM/ISBM Meeting, Penn. State University, Atlanta, Georgia, January 17 - 20. 19 p.

Tipping, J.W., Zeffren, E., Fusfeld, A.R. 1995. Assessing the value of your technology. Research & Technology Management, Vol. 38, No. 5, pp. 22 - 39.

Toivanen, J. 1992. Ohjelmoitavien logiikoiden reaaliaikaisten ohjelmistojen määrittely ja toteutus Sokrates-SA-menetelmällä. Technical Research Centre of Finland, Espoo. VTT Julkaisuja 757. 74 p. + app. 78 p. (in Finnish)

Ward, T., Mellor, S.J. 1985 - 1986. Structured development for real-time systems. Yourdon Press, New York. Vols. 1, 2 and 3.

Werner, B.M., Souder, W.E. 1997. Measuring R&D performance - state of the art. Research & Technology Management, Vol. 40, No. 2, pp. 34 - 42.

Yin, R.K. 1988. Case study research. Design and methods. Revised edition. Applied Social Research Methods Series, Vol. 5. Sage Publications, Newbury Park, CA. 165 p.

APPENDIX 1: LIST OF THE CASE DATA

Documentary data

1987

Seppänen, V. 1987. R&D diary notes.

Hakalahti, H., Karjalainen, J., Ihme, T., Okkonen, A., Pulli, P., Taramaa, J., Valmari, A., Viinikka, J. 1987. Luonnos kansallisesta tietokonetekniikan tutkimushankkeesta. Memorandum, VTT Computer Technology Laboratory, 2.2.1987.

Karjalainen, J. 1987. R&D diary notes. 135 p.

Anonym. 1987. The state of automatic code generation. Part 2: Embedded Systems, CASE Outlook, Vol. 1, No. 6, pp. 11 - 22.

Seppänen, V. 1987. Memorandum on Software engineering section, annual plan, 8.10.1987.

Annual Review of VTT Computer Technology Laboratory, 1987.

Kalaoja, J. 1987. Draco-ohjelmiston soveltuvuus minispeksin transformointiin. VTT Computer Technology Laboratory, 10.8.1987.

Kalaoja J. 1987. Working meeting on Draco. Memorandum, VTT Computer Technology Laboratory 12.10.1987

Okkonen, A., Pulli, P., Taramaa, J. and Hakalahti, H. 1987. TEKES application Dno 16/32/87/TKO. Application for the Sokrates project 15.12.1987, including project proposal 23.12.1987.

1988

Seppänen V. 1988. R&D diary notes.

Memorandum on the Schaffner of VTT Computer Technology Laboratory 21. - 22.3.1988.

Seppänen, V., Research and development in software engineering at the Computer Technology Laboratory: a short history. 4.2.1988. 4 p.

Memorandum on the Schaffner of VTT Computer Technology Laboratory 19.9.1988.

Annual Review of VTT Computer Technology Laboratory, 1988.

Hakalahti, H., Karjalainen, J. 1988. Marketing plan 1988 - 1989 of VTT Computer Technology Laboratory, 11.8.1988.

Archived correspondence of the Sokrates project: A handwritten note of Ari Okkonen on a telephone conversation between him and M. Tervonen, 6.7.1988; “Embedded systems design automation”, 3.7.1988: an overview and a vision (“johtotähti”) of the Sokrates project.

Pelkonen, J. 1988. A prototype tool for embedded systems programming automation from Oulu. Tekniikka & Talous, 2.11.1988, p. 13.

Memorandum on the Sokrates seminar 17.11.1988 at Hotel Vihiluoto.

Seppänen, V. 1988. Software engineering section, strategic plan 1988 - 1993, 13.4.1988.

Karjalainen, J. 1988. R&D diary notes. 112 p.

Tolonen, P. 1988. Interview of Lauri Gröhn, Insinööriuutiset, 23.3.1988.

Brantberg, R. 1988. interview of Ari Okkonen and Antti Auer, Insinööri-uutiset, 23.3.1988.

Contract on the participation in Finsoft research, Dno 15/52/88/TKO, 1988.

Hakalahti, H. 1988. Tekes application Dno 25/32/88/TKO for the Sokrates project, 28.12.1988.

Memorandum on the press conference of the Sokrates project, 28.6.1988, VTT Computer Technology Laboratory.

Minutes, meeting of the joint steering group of the Oppi, SW/HW and Sokrates projects, 12.10.1988.

Sokrates project summary, 2.11.1988.

Minutes, meeting of the joint steering group of the Oppi, SW/HW and Sokrates projects 22.11.1988.

1989

Karjalainen, J. 1989. Finsoft-teknologiaohjelma, yleisesittely, 17.1.1989.

Contractual project reference list, VTT Computer Technology Laboratory, 22.5.1989.

Karjalainen, J. 1989. Update of the marketing plan of VTT Computer Technology Laboratory, 31.5.1989.

Hakalahti, H. 1989. Long-term plan 1989 - 1994 of VTT Computer Technology Laboratory, 19.4.1989.

Memorandum on the Schaffner of VTT Computer Technology Laboratory 22. - 23.3.1989.

Oikarinen, M. 1989. Analysis of selected machine industry companies.

Heikkilä, E. 1989. Review of the 1989 results of VTT Computer Technology Laboratory (draft), 30.11.1989.

Hakalahti, H. 1989. Annual plan of VTT Computer Technology Laboratory for 1990, 15.11.1989.

Annual Review of VTT Computer Technology Laboratory, 1989.

Auer, A., Levanto., M. 1989. Automaattinen ohjelman tuottaminen. Proceedings of Blanko-89, Oulu. 8 p.

Reinikka, A., Auer, A., Okkonen, A. 1989. Automatic synthesis of structural HDL descriptions from graphic specification of embedded ASICs. Microprocessing and Microprogramming, Vol. 27, No. 1 - 5, pp. 473 - 478.

Okkonen, A., Auer, A., Levanto, M., Okkonen, J., Kalaoja, J. 1989. SOKRATES-SA - A formal method for specifying real-time systems. Microprocessing and Microprogramming, Vol. 27, No. 1 - 5, pp. 513 - 520.

Karjalainen, J. 1989. R&D diary notes. 173 p.

Minutes, meeting of the joint steering group of the Oppi, SW/HW and Sokrates projects, 1.3.1989.

Savilampi, J. 1989. Toimilaitejärjestelmän ohjauksen mallinnus Sokrates-SA-menetelmällä. Diploma thesis, University of Oulu, Department of Electrical Engineering. 31 p. + app.

Hakalahti, H. 1989. Tekes application Dno 15/32/89/TKO for the Sokrates project, 21.12.1989.

Sokrates status report 2/1989.

Sokrates status report 30.8.1989.

Proposal for investment, 21.9.1989 (the proposal was accepted by the management board of the embedded systems subprogram 26.9.1989).

Memorandum on the Sokrates seminar 5.10.1989 at VTT Computer Technology Laboratory.

Leppänen, T. 1989. SASIC3 project: SA/SD-VHDL transformation, University of Oulu, 9.11.1989.

Seppänen, V. 1989. R&D diary notes.

1990

Seppänen, V. 1990. Memorandum for the meeting of VTT Computer Technology Laboratory and Kone Elevators, 11.1.1990.

Marketing plan of VTT Computer Technology Laboratory for 1990 - 1991, 2.4.1990.

Seppänen, V. 1990. Marketing memorandum 1/90, 16.4.1990.

Karjalainen, J. 1990. Business plan of VTT Computer Technology Laboratory, 1.8.1990.

Hakalahti, H. 1990. Long-term plan of VTT Computer Technology Laboratory for 1991...1995, 22.3.1990.

Aho, A.V., Bjorn-Andersen, N., Haberman, N., Neuhold, E.J., Rissanen, J., Simon, J.-C., Swanson, E.B. 1990. Research and teaching in computer science, computer engineering, and information systems. A critical evaluation. Publications of the Academy of Finland 3/90. VAPK-Publishing, Helsinki. 101 p.

Oikarinen, M. 1990. R&D needs of machine manufacturers, 30.3.1990.

Seppänen, V. 1990. Software engineering section, strategic plan 1990 - 1995, 4.6.1990.

Memorandum on the Schaffner of VTT Computer Technology Laboratory, 1990.

Hakalahti, H. 1990. Evaluation of the results of VTT Computer Technology Laboratory from 1990.

Annual Review of VTT Computer Technology Laboratory, 1990.

Auer, A., Levanto, M., Okkonen, A., Okkonen, J. 1990. Solution in software crisis. Microprocessing and Microprogramming, Vol. 30, No. 1 - 5, pp. 273 - 280.

Kemppainen, J. 1990. Prosessorien välisten liitäntöjen suunnittelu SOKRATES-menetelmällä. Diploma thesis, University of Oulu. 75 p.

Okkonen, A., Auer, A., Kalaoja, J., Levanto, M., Okkonen, J. 1990. AI approach to automatic programming. Proc. of STEP-90, pp. 376 - 387.

Seppänen, V. 1990. R&D diary notes.

Karjalainen, J. 1990. R&D diary notes. 164 p.

Minutes, meeting of the joint steering group of the Oppi, SW/HW and Sokrates projects, 18.1.1990.

Minutes, meeting of the joint steering group of the Oppi, SW/HW and Sokrates projects, 23.3.1990.

Minutes, meeting of the steering group of the Sokrates project, 21.6.1990.

Sokrates video tape, Sokrates - oikotie speksistä koodiin, 23.5.1990.

Minutes, meeting of the steering group of the Sokrates project, 1.11.1990.

Seppänen, V. 1990. Brochure of the software engineering section, 17.7.1990.

Seppänen, V. 1990. Annual plan of the software engineering section for 1991, 22.10.1990.

Hakalahti, H. 1990. Annual plan of VTT Computer Technology Laboratory for 1991, 4.10.1990.

Karjalainen, J. 1990. Status information of the embedded systems subprogram, 29.5.1990.

Holopainen, V. 1990. Japanese quality thinking becoming part of Finnish information systems, Kaleva, 11.10.1990.

Karjalainen, J. 1990. Status information of the embedded systems subprogram, 29.11.1990.

Minutes, meeting of the embedded systems subprogram management board, 18.10.1990.

Project plan for the Kaapeli project, 19.10.1990.

Minutes, meeting of the management group of the Kaapeli project, 13.6.1990.

1991

Heikkilä, E. 1991. Evaluation of the results of VTT Computer Technology Laboratory from 1990, 19.2.1991.

Memorandum on the Schaffner of VTT Computer Technology Laboratory 17. - 19.2.1991.

Update of the marketing plan of VTT Computer Technology Laboratory, 9.5.1991.

Hakalahti, H. 1991. Half-year report of VTT Computer Technology Laboratory (long version), 1.7.1991.

Hakalahti, H. 1992. Annual report of VTT Computer Technology Laboratory 1991, 29.1.1992.

Seppänen, V. 1991. Status of the annual plan of the software engineering section, 31.5.1991.

Hakalahti, H. 1991. Strategic plan 1992...1996 of VTT Computer Technology Laboratory, 25.4.1991.

Seppänen, V. 1991. Long-term plan of the software engineering section, 9.3.1991

Hakalahti, H. 1991. Long-term plan of VTT Computer Technology Laboratory, 4.3.1991.

Seppänen, V., Alanko, J., Taramaa, J. 1991. Memorandum on the marketing situation of the software engineering, real-time systems and knowledge engineering sections, 18.4.1991.

Seppänen, V. 1991. Memorandum on software engineering research topics, 1.8.1991.

Auer, A., 1991. User identification/market segmentation in 1991.

Auer, A., 1991. Software engineering section, annual plan 1992 (draft), 28.10.1991.

Hakalahti, H., 1991. Annual plan of VTT Computer Technology Laboratory, 8.10.1991.

Minutes, meeting of the steering group of the Sokrates project, 11.1.1991.

Minutes, meeting of the steering group of the Sokrates project, 17.4.1991

Sokrates project, admistrative final report, 18.6.1991, Dno 7/53/91TKO (with handwritten comments by Veikko Seppänen).

Guy, K., Quintas, P., Hobday, M. 1991. Evaluation of the scientific and technological status of Finsoft: The Finnish software technology programme. Tekes, Helsinki. 58 p. Appendix III: project notes (confidential). 24 p.

Saukkonen, S. 1991. Finsoft-ohjelman tulosten teollisen hyödyn ja hyödynnettävyyden arviointi. Tekes, Helsinki. 75 p. Appendix: projectwise evaluations (confidential). 92 p.

Karjalainen, J., (ed.), 1991. Finsoft ohjelmistoteknologian ohjelma. Sulautetut järjetelmät 1988 - 1991. Tekes, Helsinki. 167 p.

Hakalahti, H. 1991. Finsoft software technology program: Embedded systems. VTT Computer Technology Laboratory, 13.3.1991. 5 p.

Peltola, E. 1991. Telefax to Hannu Hakalahti on the role of VTT in the Finsoft program, 19.8.1991. 8 p.

Peltola, E., Hakalahti, H. 1991. On the participation of VTT in the software technology program (FINSOFT) and its evaluation, memorandum delivered to M. Mannerkoski, J. Forsten, E. Heikkilä, 20.8.1991.

Minutes, meeting of the Kaapeli management group, 29.11.1991.

Seppänen, V. 1991. R&D diary notes.

Karjalainen, J. 1991. R&D diary notes. 187 p.

Seppänen, V. 1991. Marketing letter, 14.8.1991.

Seppänen, V. 1991. TEKES project proposal: Reuse of embedded software, 7.6.1991.

Seppänen, V. 1991. Project planning memorandum, 1.8.1991.

Marketing brochure, (MCS-REA project), 2.11.1991

Press release, Reagenix generator makes real-time programming faster, VTT Computer Technology laboratory, 18.12.1991.

Seppänen, V. 1991. Memorandum on the marketing and continuing projects of Sokrates, 27.3.1991

Auer, A. 1991. RT-SA/SD training offer for Kemira Engineering.

Contract draft for the Kaapeli project, 27.5.1991. Appendix 1. Project plan: Application of the Sokrates technology, 15.4.1991.

Dno 24/52/91/TKO, contract on the Kaapeli project, 11.6.1991.

Contract Dno 40/52/91/TKO for the Raski project, 13.12.1991.

1992

Heikkilä, E. 1992. Evaluation of the results of VTT Computer Technology Laboratory from 1991, 19.2.1992.

Hakalahti, H. 1992. Strategic plan of VTT Computer Technology Laboratory, 5.5.1992.

Hakalahti, H. 1992. Half-year report of VTT Computer Technology Laboratory, 21.7.1992.

Strategic plan of VTT Electronics Laboratory for 1993 - 1996, 15.6.1992.

Auer, A. 1992. Long-term plan of the software engineering section, 15.4.1992.

Auer, A. 1992. Memorandum: analysis of the current customers, 13.3.1992.

Hakalahti, H. 1992. Notes on the Schaffner of VTT Computer Technology Laboratory.

Karjalainen, J. 1992. Marketing plan of VTT Computer Technology Laboratory for 1992 - 1993, 20.5.1992.

Heikkilä, E. 1992. Strategic plan of the Information Technology Department, 18.8.1992.

Hakalahti, H. 1992. Annual plan of VTT Computer Technology Laboratory for 1993, 21.10.1992.

Minutes of the meeting 2/92 of the industrial steering group of VTT Computer Technology Laboratory, 19.10.1992.

Marketing plan for machine industry, 21.12.1992.

VTT Computer Technology Laboratory, Annual Review 1992.

Karjalainen, J. 1992. R&D diary notes. 185 p.

Mälkki, Y. 1992. Order for the Kaasu project, VTT Food Technology Laboratory.

Okkonen, A. 1992. Project plan for the Raski project, 27.3.1992.

Minutes, meeting of the Raski management group, 11.03.1992.

Minutes, meeting of the management group of the Raski project, 5.5.1992.

Minutes, meeting of the management group of the Raski project, 9.6.1992.

Minutes, meeting of the management group of the Raski project, 6.11.1992.

Okkonen, A. 1992. Final report of the Raski project, 29.11.1992.

Auer, A. 1992. Memorandum on the marketing activities of the software engineering section.

Auer, A. 1992. Poster on the specification technologies of real-time programs, 17.9.1992.

Auer, A. 1992. Software technology reference sheet: efficiency and quality for software development.

Auer, A. 1992. Co-operation offer for Dassault.

Presentation slides on code generation, 27.8.1992.

Draft of the 1992 Annual review of VTT Computer Technology Laboratory.

Contract Dno 1/52/93/TKO for the Osdyn project.

Päivike, H. 1992. Project plan for the Osdyn project, 13.11.1992.

Minutes, meeting of the Osdyn management group, 8.10.1992.

Minutes, meeting of the Osdyn management group, 28.10.1992.

Minutes, meeting of the Osdyn management group, 13.11.1992.

Minutes, meeting of the Osdyn management group, 17.12.1992.

Contract Dno: 24/52/92/TKO, for the Cute project, 27.08.1992.

Summary of the Turva project, 12.2.1992.

Korhonen, J., Kalaoja, J. 1992. Turvallisten sulautettujen ohjelmistojen kehittäminen. Technical Research Centre of Finland, Espoo. VTT Research Notes 1430. 72 p. (in Finnish)

1993

Hakalahti, H. 1993. Annual report of VTT Computer Technology Laboratory, 2.1.1993.

Minutes, meeting of the Osdyn management group, 25.1.1993.

Minutes, meeting of the Osdyn management group, 5.2.1993.

Minutes, meeting of the Osdyn management group, 27.4.1993.

Okkonen, J. 1993. Project plan for the Cute project, 29.03.1993.

Minutes, meeting of the management group of the Cute project, 24.9.1992.

Minutes, meeting of the management group of the Cute project, 14.1.1993.

Minutes, meeting of the management group of the Cute project, 5.5.1993.

Minutes, final review of the Cute project, 11.8.1993.

Final report of the Cute project, 5.5.1993.

Ihalainen, J. 1993. Reaaliaikaohjelmistojen yksikkötestaus. University of Oulu, Department of Electrical Engineering, Diploma thesis. 59 p. + app.

Kurki, M., Hirvinen, J., Malm, T. 1993. Diagnosis of mechatronic systems. Final report. VTT Computer Technology Laboratory, VTT Safety Engineering Laboratory. Dno 6/53/93/TKO. 13 p. + app.

Minutes, meeting of the industrial steering group of VTT Computer Technology Laboratory, 4.5.1993.

OTE/RAI strategic planning: SWOT analysis, P. Pulli, A. Auer, 30.3.1993.

Strategic plan for software engineering 1993, V. Seppänen, 5.2.1993.

Auer, A. 1993. Rotu project plan, 1.3.1993.

Hakalahti, H. 1993. SWOT analysis of VTT Computer Technology Laboratory, 1.4.1993.

Hakalahti, H. 1993. Strategic plan of VTT Computer Technology Laboratory, 4.5.1993.

Material prepared for the Schaffner of VTT Computer Technology Laboratory, 10. - 11.5.1993.

Further specification of the marketing plan of VTT Computer Technology Laboratory for 1993, 8.6.1993.

Hakalahti, H. 1993. Half-year report of VTT Computer Technology Laboratory, 3.8.1993.

Competitor analysis of VTT Computer Technology Laboratory, 15.4.1993.

VTT Electronics, Embedded systems annual review 1993.

Seppänen, V. 1993. R&D diary notes.

Seppänen, V. 1993. Segmentation of the customers of the Embedded Software research area in 1993 - 1994.

Saari, H. 1993. Koneautomaatio-ohjelmistojen komponentointi. University of Oulu, Department of Electrical Engineering, Diploma thesis. 67 p. + app.

Offer, Development of an SA-level graphic debugger, 16.7.1993.

Lintulampi, R. 1993. Prosac-koodigeneraattorin soveltuvuusselvitys Lokki-projektin tarpeisiin. VTT Tietokonetekniikan laboratorio. 9 p. (in Finnish)

Rotu/Aniprosa-2 project report 4.10.1993: Graphical debugger Reagenix/Aniprosa.

Contract Dno 18/52/93/TKO, 26.3.1993 (MCS-REA), Appendix: Plan for R&D work.

Minutes, meeting of the MCS-REA management group, 5.4.1993.

Minutes, meeting of the MCS-REA management group, 30.6.1993.

Minutes, meeting of the MCS-REA management group, 9.9.1993.

Minutes, meeting of the MCS-REA management group, 12.11.1993.

Auer, A. 1993. MCS-REA continuation work option.

1994

Memorandum on the animation of Prosa diagrams, 3.2.1994.

Order, Dno 8/51/94/ELE3, 4.8.1994.

Offer, 26.6.1994: Finishing of the debugger developed for Reagenix models, including Aniprosa-2 project plan, 20.6.1994.

Minutes, meeting of the management group of the MCS-REA project, 22.4.1994. Appendix. Auer, A. 1994. Final report, 27.4.1994, version 1.0.

Minutes, final project review, MCS-REA, 30.5.1994.

Seppänen, V. 1994. R&D diary notes.

Häkli, R. 1994. Sulautettujen oliokeskeisten ohjelmien testaaminen. University of Oulu, Department of Electrical Engineering, Diploma thesis. 53 p. + app. (in Finnish)

Häkli, R., Seppänen, V., Ihme, T. 1994. State-based unit testing of object-oriented software. VTT Electronics. 20 p. + app.

Perunka, H. 1994. R&D diary notes.

1995

Hirvinen, J., Määttä, T. 1995. Final report, Improvement of the safety of a wood carving machine based on control system and mechanics implementation, 15.5.1995.

Virtanen, A., Mäkeläinen, T. 1995. Sulautetun ohjelmiston tuottaminen mallinnusohjelmalla. VTT Automation. Working report. 31 p. (in Finnish)

Auer, A., Korhonen, J. 1995. State testing of embedded software. Proceedings of EuroStar'95, 27 - 30 November 1995, London.

Seppänen, V., Kurki, M., Perunka, H., Päivike H. 1995. Marketing goals for embedded software 1994 - 1995.

Seppänen, V. 1995. R&D diary notes.

Perunka, H. 1995. R&D diary notes.

Röning, J., Kauniskangas, H., Kalaoja, J., Okkonen, A. 1995. From craftwork towards industrial production in the development of real-time machine vision software Proceedings SPIE, Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling. Philadelphia, PA, 23 - 26 October 1995. Vol. 2588.

1996

Röning, J., Kalaoja, J., Okkonen, A., Kauniskangas, H. 1996. Reaali-aikaisten konenäkösovellusten kehittäminen. Technical Research Centre of Finland, Espoo. VTT Tiedotteita 1777. 72 p. + app. 40 p. (in Finnish)

Oivo, M. 1996. R&D diary notes.

1997

Offer, 12.6.1997, Tunturipyörä Oy, Jouko Paavilainen.

Huuskonen, P. 1997. Final report of the Rulla-2 project, Dno 5/53/97/ELE3, 16.6.1997.

Haapanen, P., Heikkinen, J., Korhonen, J., Maskuniitty, M., Pulkkinen, U., Tuulari, E. 1997. Feasibility studies of safety assessment methods for programmable systems. Final report of the AVV project. STUK-YTO-TR 93. Finnish Centre for Radiation and Nuclear Safety, Helsinki. 54 p. + app.

Haapanen, P., Pulkkinen, U., Korhonen, J. 1997. Usage models in reliability assessment of software-based systems. STUK-YTO-TR 128. Finnish Centre for Radiation and Nuclear Safety, Helsinki. 48 p.

Interview data

1. Veikko Seppänen, case summary, 30 July 1997 (version 0.2 draft), commented by Ari Okkonen, Jarmo Kalaoja and Kari Leppälä.
2. Ari Okkonen, electronic mail, 5 May 1998.
3. Group rehearsal, 6 March 1998, from 15.10 to 18.10: Ari Okkonen, Mikko Levanto, Jyrki Okkonen, Jarmo Kalaoja, Jukka Kemppainen.
4. Antti Auer, electronic mail, 17 May 1998.
5. Pekka Pesonen, electronic mail, 18 March 1998.
6. Hannu Hakalahti, interview, 18 February 1998, from 18.15 to 19.45.
7. Jukka Karjalainen, interview, 25 February 1998, from 15.40 to 16.55.
8. Lauri Gröhn, electronic mail, 2 March 1998.
9. Matti Sihto, electronic mail, 24 March 1998.
10. Urpo Tuomela, electronic mail, 19 March 1998.
11. Timo Mukari, electronic mail, 17 March 1998.
12. Pekka Isomursu, electronic mail, 9 March 1998.
13. Antti Valmari, interview, 4 May 1998, from 14.20 to 15.35.
14. Tapio Halkola, interview (by Jaana Määttä), April 1998.
15. Mika Vanne, interview (by Jaana Määttä), April 1998.
16. Petri Pulli, electronic mail, 16 March 1998.
17. Tapio Hekkilä, interview, 10 March 1998, from 13 to 13.35.
18. Pekka Kemppainen, electronic mail, 10 March 1998.
19. Hannu Marjakangas, electronic mail, 19 March 1998.
20. Kari Hakkarainen, electronic mail, 10 March 1998.
21. Kari Tiensyrjä, electronic mail, 10 March 1998.
22. Eila Niemelä, electronic mail, 10 March 1998.
23. Markku Heikka, electronic mail, 9 April 1998.
24. Ilkka Kuuluvainen, electronic mail, 16 March 1998.
25. Jorma Taramaa, electronic mail, 9 March 1998.
26. Discussion with Jorma Taramaa, 9 March 1998.

27. Aija Vostrakov, electronic mail, 9 March 1998.

28. Juha Alanko, electronic mail, 11 March 1998.

29. Heikki Päivike, electronic mail, 9 Mar 1998.

30. Timo Mukari, electronic mail, 17 March 1998.

31. Andres Kull, electronic mail, 20 March 1998.

32. Gösta Silfver, electronic mail, 24 March 1998.

33. Markku Heino, electronic mail, 27 March 1998.

34. Juha Röning, electronic mail, 2 April 1998.

35. Discussion with Jukka Korhonen, 10 March 1998

36. Jouni Heikkinen, electronic mail, 12 March 1998.

37. Esa Tuulari, electronic mail, March 16, 1998.

38. Pentti Haapanen, electronic mail, 11 March 1998.

39. Marko Salmela, electronic mail, 16 March 1998.

40. Discussion with Pauli Räsänen, 7 April 1998.

APPENDIX 2: CODE GENERATION CASE DATA

The case data is summarised in this appendix chronologically. The primary data gathered through interviews is structured around the secondary documentary data. Examples of both types of data are given. The former were acquired through several interviews and one group rehearsal and the latter gathered from the archives of VTT, personal R&D diaries, and several technical and managerial documents. Parts of the data have been translated from Finnish to English by the first author of this report. The content of one of the most important pieces of the primary data, an English-written transcript of a group rehearsal of the code generation researchers, was sent to the informants for review. A considerable amount of the primary data was, however, collected through electronic mail and did not require any transcription and review by the informants.

All code generation related activities and their results are described in this appendix by using the concepts of the case data, rather than those of the competence evolution framework. The activities are structured around the code generation projects, because not only code generation research and development was carried out in projects, but also competence marketing, purchasing and exploitation were closely associated to projects.

A2.1 SETTING UP THE SCENE FOR INNOVATION

Table A1 shows the three projects in which code generation research started at VTT. The first two projects were carried out in the mid-eighties for a key customer with which TKO had established a strategic co-operation alliance in the early eighties. A pre-study was also carried out to evaluate related work on the so called transformational software development approaches. Although the study was in part financed from a public source, the actual investigation was planned and conducted solely by TKO. Therefore, the initial phase included pseudo-green and red projects.

Table A1. Code generation related projects at TKO in 1985 - 1987.

Project	**Type of project**	**Activities**	**Financing** (1000 FIM)	**Year**
Speco	Red project for the firm K	Research of code generation techniques for the PLM language	<100? (firm K)	1985
Speco-2	Continuing of the Speco project	Development of a prototype of a code generator	<100? (firm K)	1987
Draco	Pseudo-green TKO project	Evaluation of existing program transformation approaches and tools	<100? (KTM)	1987

A2.1.1 Marketing of the idea of code generation

The management of TKO analysed project marketing data from 1983 to 1988 in two planning meetings in 1988. As indicated in Frame A1, TKO had rather few regular key customers and plenty of occasional customers, many of which took part in blue projects. One of the main aspects of the emerging code generation marketing strategy was thus to make industry first involved in blue research projects and then in subsequent red projects, where the research results would be made feasible for exploitation.

In the annual report of TKO from 1987 the results of the two Speco projects are described as a compiler that had been "successfully tested ... in a real system example" and "will be further developed for production use". This business goal held for the next ten years, but its meaning and achievement became heavily debated issues.

Table A2 illustrates marketing events related to the continuation of the initial code generation research. The table is by no means complete, because little information has remained from these events in the archives of VTT. However, even the few examples indicate that both the managers and the researchers were actively marketing code generation as a joint research topic to industry and Tekes. A tool vendor was contacted in this early phase, in order to promote the expected commercialisation of the research results.

Frame A1. Extracts from the market analyses of TKO in 1998.

Seppänen, V. 1988. Personal R&D diary notes.

TKO schaffner 21. - 22.3.1988

Customer base analysis: 3 current key customers, 1 raising customer, 2 lowering customers, 9 one-shot customers, 5 previous customers, 23 potential new electronics customers; 12 customers of research projects (6 of which are not contractual customers), 9 new customers of planned new research projects. Machine industry's customers: 3 contractual customers; 5 research project customers; 13 potential customers. Altogether 21 customers in 1987.

TKO schaffner 19.9.1988

Analysis of the reasons for the starting of new projects in 1986, 1987 and 1988. 1986: 14 projects analysed: 4 "old contacts", 4 "TKO's own ideas", 6 "originated by the company"; 1 VTT project, 4 TEKES projects, 9 customer projects (5 different customers). Speco: company-originated, a part of the co-operation agreement. Did not start: 4 projects analysed, all "TKO's own ideas" (and targeted to funding bodies or VTT). 1987: 12 projects analysed: 5 TKO's own ideas, 5 company-originated, 1 scientific co-operation, 1 started by an industrial organisation; 1 VTT project, 4 TEKES projects, 1 NI project, 1 Sitra project, 5 customer projects (4 different customers). Did not start: 5 projects analysed, 1 VTT, 1 Ministry of environment, 2 TEKES, 1 customer project. 1988: 19 projects analysed: 10 TKO's own ideas, 4 company-originated, 2 old contacts, 1 foreign contacts, 1 VTT/ELE's pre-study; 4 VTT projects, 5 TEKES projects, 1 KTM project, 7 customer projects. Sokrates: Tekes/Finsoft, TKO's ideas, earlier industrial projects.

Table A2. Examples of code generation related marketing events in 1987.

Event	**Issue**	**VTT's persons**	**Target audience**	**Date**
Meeting with the Firm N	Co-operation on code-generation proposed	Line managers	Customer's managers and technology experts	22.4. 1987
Marketing meeting With the firm V	Discussion on code generation and CASE tools	Line manager	R&D manager of an automation firm	9.10. 1987
Meeting with the firm I	Discussion of the terms of the Tekes contract	Line manager	Managing director of a CASE tool vendor	24.11. 1987
Project marketing	Presentation of the Sokrates project proposal	Line managers, Researchers	Industrial experts, Representatives of Tekes	16.12. 1987

A2.1.2 Conducting of code generation pre-studies

A prototype of a code generator for the PL/M programming language was developed for the firm K in the two Speco projects. The work was a part of a large project portfolio established to build an application-specific industrial embedded systems software development environment for the firm K. The manager of the firm, who suggested the study of the code generator, was a former head of the software engineering section of TKO. The archives of VTT do not include much material of the two projects, but they were apparently rather small. The results of the projects were not taken in industrial use, but the firm K joined instead the subsequent joint code generation research project Sokrates.

The Draco project was a pseudo-green pre-study to evaluate existing program transformation methods and tools, funded by the Ministry of Trade and Industry (KTM). It was initiated by the line managers of TKO, with a help of senior researchers interested and personally involved in international co-operation. The results were reported in a diploma thesis [Kalaoja 1988], whose author became later a member of the Sokrates project group. One of the basic conclusions of the study was that the lack of automation would make the use of the existing code generation tools difficult in industrial embedded systems software development. Moreover, the code that the existing tools produced was not appropriate for embedded systems and the methods on which they based were not familiar to embedded systems practitioners. There was clearly a need to carry out further work.

A2.1.2.1 Individual background – putting novel ideas to work

The idea of code generation was put forward by the R&D manager of the firm K, and it fell into a fertile soil at TKO. The coming Sokrates project manager had carried out pioneering embedded software development work in the late seventies as an entrepreneur, before joining TKO in the mid-eighties. He was excited about the possibility to develop new innovative technologies that would solve some of the problems that he had faced in his practical software engineering career. He describes the background and motivation for the code generation research as follows.

"In 1973, I carried out modifications of the H1642 operating system at the computer center of the university … I gained hands-on experience from concurrent systems … and gave exercises at the information processing laboratory on computer system programming. In 1974 I programmed as a hobby a small real-time operating system kernel for the H316 process computer. It remained there to wait for the future needs. I tested it later in one night, between March 31 and April 1 in 1976. In about 1980 I designed together with [two other future Sokrates project members] software objects for a telecommunication software package ... The objects followed three key principles … [that ensured the independence of the software from the computer hardware on which the software was executed]. My task was to design state machines for the software. In 1976 I developed a taxation meter based on state machines.

In about 1977 I found out that … the so called traditional structured programming techniques [which were not based on state machines] were of almost no use for the design of telecommunication software systems. In about 1980 I designed together with [one of the future members of the Sokrates project] a descriptive macro language … based on a specification that was only a few pages long. In the textbook of Per Brinch-Hansen "Operating System Principles" that I and [the other member] had read already in 1973, some of the principles [used in the design of the language] had been presented thoroughly. I believed, after reading the book, that I managed everything on real-time systems ... but I was wrong.

In about 1981 I took part in the development of the Kajaani 400 bleaching process computer software. In this work I learned the Jackson Structured Programming (JSP) method. I also learned how to not design software components. In about 1982 I carried out another project for Kajaani … in which I did first a comprehensive system analysis. The analysis that was based on my own method took about three months, but the design and implementation of the software lasted then only for two weeks. In this work I learned the importance of a thorough system analysis. In 1983 I developed a programming language for the Kajaani 4 process controller together with [the above mentioned future Sokrates project member]. We got first familiar with the other similar systems available on the market … but designed [a new] programming language, for which it would be easy to build a pre-compiler. We designed in detail and tested the compiler and an interpreter. I got also familiar with the basics of digital signal processing.

I had read textbooks and taken university courses on information systems specification methods. They seemed to work in the given examples … [but] problems arose when they were attempted to be used for some real problem, whose solution was not yet known. Moreover, the methods did not give any support at all to [solving] real-time problems and using state machines. In 1985 I took part in the Yourdon RT/SA course. It was love from the first sight. The approach had almost everything that I wanted for designing real-time systems. There were [some minor negative] aspects that I thought could be improved by a few own extensions [to the method]."

This small story illustrates, on one hand, the practical problems of the early days of embedded software engineering, but on the other hand also the fact that there were some new methodological foundations on which innovators could build solutions from which practitioners could immediately benefit.

A2.1.2.2 Organisational background – software engineering

From the organisational viewpoint code generation research can be seen as a part the development of industrial embedded software engineering approaches. In the short "history" of software engineering research at TKO, written two months before the main code generation project Sokrates started, Veikko Seppänen describes how TKO had moved from earlier work on small-scale embedded software development towards more comprehensive activities involving larger applications and integrated embedded software production environments.

"In general the evolution of the SE [software engineering] related work has lead us from developing systems software for small portable embedded devices into the construction of larger embedded software systems, along with the research and development for the production environments of such systems. A shift from developing the basic tools to automate SE methodologies, to applying the currently existing CASE tools and enhancing them with more advanced features has taken place. Yet, we are still researching the basic SE automation matters, but in the context of more sophisticated software engineering paradigms (transformation systems, executable specifications, prototyping). Several proposals have been made for the Finsoft programme, to start in April, 1988. The proposals deal with reuse-centred, transformational and domain-based embedded systems production techniques, and also with a contracting management directed software production paradigm, especially the role of specification in such a scheme. Started in 1982, the series of projects [with the firm K] has evolved … to the new generation systems synthesis techniques based on transformational, domain-based production approaches".

In addition to the view of TKO moving towards more comprehensive problems in embedded software engineering, the story of the section head Veikko Seppänen illustrates his own experiences from the past few years on the so called domain-based and transformational software development approaches, considered as "more sophisticated" than the "basic" software engineering methods and tools. The Speco projects referred to in the historical overview are also seen from this methodological perspective.

A2.1.3 Planning and control of innovation

According to the customer of the two Speco projects, the firm K, the main reason for its interest in code generation was the view that software would become a key technology in its business and that plenty of new software would need to be developed in the nineties. More emphasis was attempted to be put on specification of the software, whereas the implementation of the specifications would benefit from automated program generation. “Considering it in retrospect, there was obviously no solid ground for this view. The plans were mostly rather rough ideas.”

Apparently for the sake of the strategic intent, the customer wanted the code generation idea to be studied in a “rather strictly planned way, to find solutions to research problems and to test hypotheses”. However, the researchers of TKO seemed to think that “this kind of an investigation cannot be planned according to any schedule at all.” Although both parties aimed at solving practical problems with innovations, the views of entrepreneurs who spent years to tackle small aspects of big issues at a time – day or night – and a manager seeking systematically for strategic solutions to corporate-level issues collided with each other.

One of the researchers “spent a year to model the behaviour of the product [of the firm K] using the RT/SA method”, but the analysis was stopped when the customer’s manager required another researcher to address the problem instead, using the customer’s own approach to produce results faster. This model was reformulated by using the customer’s approach. Although it took twenty minutes to execute the model by a computer, the result “was a … breakthrough”. The concepts of the SA/SD modelling language had been formalised to a degree that made it possible to execute them on a computer, and also to compile them to other computer languages.

The customer had already planned for the next step: “we were commanded to develop an RT-SA to PL/M-86 compiler”. The two researchers, the coming two project managers of Sokrates, presented this work to be done in the Speco projects to the personnel of TKO “by sketching the classical unidentified flying object to a drawing board and telling them that we will build an UFO”. The customer had also prepared some solution elements and a set of material on research related to code generation, which were handed over to the researchers. They carried out by themselves “a rather comprehensive information search … whose results remained quite poor”. Therefore, they developed their own approach in which “concurrency in the specification and in the software implementation were made completely separate from each other” - the basic principles of code generation from SA/SD models had been invented!

In parallel, Draco and another program transformation system were being evaluated and found out not to be suitable for real-time embedded software applications. The road was thus open for extending the self-made innovation and making it available also to other exploiters than the firm K. Therefore, the subsequent planning activities carried out jointly by the researchers and the managers of TKO served the preparation of a continuing joint research project.

This project was intended to become a part of a more comprehensive research program whose planning the management of TKO had activated already in 1986. The so called research council consisting of the managers and a few senior researchers was established to help to prepare the plan. The Speco researchers were responsible for the preparation of a part of the plan dealing with automated production of embedded systems software.

The research council met several times in 1987. The first two authors of the plan ([Hakalahti et al. 1987] in Appendix 1) were the director and deputy director of TKO. Altogether five researchers authored parts of the plan intended to be a seed for future Tekes project proposals. Code generation tools were described in the plan as “embedded systems application generators”. In other words, embedded systems code generation was associated to program generators used in data processing applications already for several years, not to the results of such academic program transformation research as Draco. Based on this plan, the Sokrates project was proposed to Tekes by TKO in December 1987 ([Okkonen et al. 1987] in Appendix 1).

The authors of the Sokrates project proposal were the laboratory director and three senior researchers of TKO, one of whom became the first manager of the project. In 1987, the idea of code generation for embedded systems was already being actively discussed in practically oriented professional journals (cf. [Anon. 1987] in Appendix 1), and such technically advanced firms as K had carried out small-scale studies of the subject. Yet, no practical code generators were commercially available for embedded systems software engineers.

The results of the project were described in the proposal as “methods and tools” for “the mechanisation of the software construction process” that would lead to “a quantum leap” in embedded systems software development. They were expected to be available for trial use as soon as in 1988 and for industrial use in 1989. They would be packaged into a “design support system” by 1992, if the industry was willing to finance the continuing work. The planned volume of the project was 15 man-years and the budget over six million Finnish marks, of which Tekes would pay almost 80%, VTT about 20% and industry less than 5%. Four firms were mentioned in the plan as interested in the project - K, N, E and S.

A2.2 BUILDING OF THE CODE GENERATION COMPETENCE

The Sokrates project carried out by TKO from April 1988 to May 1991 was a part of the Finsoft research program launched and funded by Tekes. The code generation capabilities of VTT were largely built in Sokrates, which was the biggest project in Finsoft. It remained also as the biggest project in terms of efforts and financial volume in the history of the code generation related activities of VTT. TKO had four projects in the Finsoft program.

Sokrates was as a project managed by the software engineering section led by Veikko Seppänen since September 1987. The size of the section was only about ten people in the late eighties, and Sokrates was thus one of the core projects of the section. The director of TKO since 1986 when he left the computer engineering research group of the local university, Dr. Hannu Hakalahti, was heavily involved in the planning and co-ordination of the Finsoft program. The deputy director Jukka Karjalainen, a diploma engineer in electronics and a manager at VTT since the early eighties, became the head of the subprogram devoted to embedded systems.

A2.2.1 Sokrates - the core code generation project

Altogether as many as fifteen firms, K, N, E, S, Nm, W, I, So, T, P, Nt, V, Pa, Ta and Th were members of the steering group of the Sokrates project during 1988 - 1991. The group is still the biggest in all embedded software related blue projects at VTT. The representative of the firm K, for which the Speco projects had been carried, was not the R&D manager but a knowledge engineering expert and a former researcher of TKO. He managed the software engineering department of the firm.

Most of the other firms of the steering group were not closely related to TKO either in terms of projects or personal contacts. However, N that was one of the key customers of TKO. Its representative in the steering group was a former VTT researcher, who used to work in projects carried out for the firm K. He was interested in joining the steering group, based on “what he saw and knew of the plans of the firm K”. The representative of the firm Th used to be a software engineer at the firm K and a research trainee at TKO. The firm I was a tool vendor, whose main business was to develop and sell a commercial CASE tool to support the structured SA/SD design method applied in the Sokrates project. The founders of this firm had left VTT in the mid-eighties. The firm aimed at including code generation features in its tool in the late eighties. The managing director was not satisfied with the intended role of the steering group of the Sokrates project:

> *Jukka Karjalainen’s R&D diary notes 1987. 135 p. 24.11.1987, meeting with the managing director of firm I: “The item of the Tekes project contract stating that the utilisation of the results is decided by the steering group cannot be accepted [by firm I]”.*

Five of the firms can be characterised as large with regard to the volume of their embedded software development activities. Their annual project participation fee was 25 000 Finnish marks. Others, who paid much less for their participation, were small or not primarily dealing with embedded systems software. As an example, the firm P was a government authority. Four of the firms were dealing with machine and process automation applications, two with electronic instruments and four with tele-communications. One of the firms was a software house and another an embedded systems subcontractor.

The industrial segments in which the potential customers for the code generation research results might be found in the nineties were thus well represented in the steering group, in addition to the tool vendor representing the so called electronic design automation (EDA) business in Finland.

The project involved seven key persons as researchers, four of whom joined TKO for preparing their Master's theses as research trainees. One of the researchers was affiliated with the knowledge engineering section and another with the real-time systems section of TKO, the rest with the software engineering section. In 1988, the first period of Sokrates was ongoing since April. The first results and the future goals of the project were summarised at the end of 1988 as shown in Frame A2.

Frame A2. Summary of the Sokrates project in November 1988.

Sokrates project summary 2.11.1988.

"VTT's Computer technology laboratory has investigated and successfully applied real-time systems' development and analysis methods for several years. Methods and tools will be developed in the research for the modelling, analysis, design and implementation of embedded systems. The project aims at a demonstration of the seamless computer-supported development of embedded systems from requirements to implementation".

The original project manager of Sokrates left – rather surprisingly to the management and colleagues – TKO to join firm Nt in 1989, and another person from the project group, the one with whom he had carried out the Speco projects, took his place. His stay at Nt lasted, however, only for a few months. He returned back to TKO and joined the project group again, but now as a chief researcher focusing on technical matters.

The Sokrates method on which the code generation solution was based was described as "formal" by the researchers, who justified the development of "completely new approaches" by the need to reduce the method's dependencies on specific implementation technologies, such as certain types of computer hardware and operating systems. The goal was thus to produce a generic solution for the code generation problem.

The general strategy to achieve this goal was to proceed bottom up, starting from the embedded software to be generated, Frame A3. This was opposed to many other research strategies at that time, which took a top-down approach by addressing first higher-level conceptual problems and then implementing technical solutions to those problems. The reversed approach was justified by the practical background of the researchers and by their aim at innovative but industrially applicable results.

Frame A3. Sokrates strategy for ensuring industrially applicable results.

Tekes application Dno 15/32/89/TKO, 21.12.1989, H. Hakalahti. Appendix: Research plan 1.4.1990 - 31.3.1991, Antti Auer, 20.12.1989 (revised 28.3.1990).

"The results of the second phase by 31.10.1989: The development of the design automation solutions started from the last design phases, implementation of the software, in order to ensure that industrially usable solutions to produce embedded software is developed. The methods and the compiler have been applied in a real-life example, a hydraulic controller (Synchro) developed by the Electronics Laboratory of VTT".

1990 was the busiest year of the Sokrates project in terms of the number of people involved and the volume of the research being carried out. Perhaps for this reason only three documents were produced as publicly available results: a conference paper, a diploma thesis and a presentation for a national engineering workshop. However, several technical reports and draft versions of users manuals were written for the use of the steering group.

In one of the publications, the developed results were seen as a "solution in software crisis" by the researchers, and they were explained to represent an "AI [artificial intelligence] approach to automatic programming". A reason for the latter may be the fact that one of the researchers was a member of the knowledge engineering section that focused on AI applications. Tekes, industry and research organisations were investing heavily in artificial intelligence in another national research program. The background of some of the existing program generator tools, such as Draco, was in artificial intelligence. Even the Sokrates code generator prototype was implemented in Prolog, one of the most common AI programming languages at that time.

A new research topic was addressed in the project in 1990, design of a communication protocol that would support the use of automatically generated code in distributed systems. This research was carried out by a research trainee, who later joined another blue project dealing with formal analysis techniques. An exchange visit of the chief researcher of that project to abroad was supported financially by Sokrates, based on a suggestion made by the management of TKO.

The two projects had something in common in principle, because the analysis techniques developed by the above mentioned researcher were intended to be included in Sokrates. However, this was later removed from the task list of the project. The arrangement of the visit was thus made mainly for gaining additional funding for the visiting researcher. The results of the visit were reported as a part of Sokrates, but never actually integrated into its results.

A piece of data from 1990 shows the first sign of a change in the methodological foundations adopted in the Sokrates project, at TKO and in industry in more general terms. SA/SD type structured design methods were expected to be challenged by the so called object-oriented methods.

Software engineering section's strategic plan 1990 - 1995, V. Seppänen, 4.6.1990: "Object-oriented techniques will emerge in specification".

Much of the effort of the last operational year of the Sokrates project was spent on documentation, whereas the evaluation of the results by an industrial case study was planned to be done only in four man-months:

Tekes application Dno 15/32/89/TKO, 21.12.1989, H. Hakalahti. Appendix: Research plan 1.4.1990 - 31.3.1991, Antti Auer, 20.12.1989 (revised 28.3.1990): "Tasks for the third phase: further development of the methods (24 man-months); experimentation with the method in real-life embedded systems design case (4 man-months), reporting and documentation (10 man-months)".

The aim at developing a working prototype of the code generator was not achieved. However, this was not considered as any major failure either at TKO or among the members of the steering group, because it was believed that the work can be completed in subsequent red projects funded by Tekes and industry on a fifty-fifty basis. One of the main targets for such projects would be to transfer the code generation tools entirely to a PC environment.

By the summer of 1991, the Finsoft program and the Sokrates project had been finished. The results of the project included a variety of methods, tools and pieces of software ("system solutions"). Licenses of the system solutions, which were initially not planned to be developed at all, were expected to be sold as intellectual property rights (IPR) to companies together with consultation and training services. These solutions included a program library, a parser generator, an operating system nucleus, an interface library and a communication protocol software package. Moreover, a patent application on a scheduling method based on the developed operating system had been made and was later accepted.

The high-level system requirements definition method QFD (Quality Function Deployment) had been integrated into the Sokrates design method in co-operation with a self-funded green research program of VTT in which members of the Sokrates project group participated. The two methods covered together the whole embedded systems development process. The developed Sokrates design method and modelling language were reported to be "taken in use, with some modifications, in one company of the support [alias steering] group" and the programming principles and solutions in two other companies.

The code generator had been used by VTT Electronics Laboratory in two blue projects. The firm Nm had used the operating system interface developed in the project and the firm N had produced, based on the Sokrates scheduling approach, an operating system for its telecommunication products sold world-wide in large numbers. A space instrument project carried out by TKO had utilised the communication protocol software package. A step towards the goal "to develop the code generator further for industrial use in a separate continuation project" was taken in a red project being carried out for the firm Nm.

In financial terms the Sokrates project went almost as planned, except that over two man-years more work was carried out, because of the large number of research trainees, whose salaries were rather low (Table A3). The total expenses of the project were about a million Finnish marks less than in the first project proposal in 1987. The realised industrial income was much higher than in the first proposal, although not as high as planned later. Investments in commercial computing resources (“bought items”) were rather modest, well under half a million Finnish marks in three years. This can be contrasted with the joint knowledge engineering research projects being carried out at the same time, where very expensive special computers and software products were bought by TKO [Seppänen et al. 1998a].

Table A3. Planned vs. realised figures of the Sokrates project.

Item	**Planned**	**Realised**	**Difference**
Efforts (man-months)	35+39+39=113	40+50+47=137	24
Expenses (1000 FIM, during three years)	1371+1681+ 1999=5051	1389+1722+ 1787=4898	153
Salaries	1257+1329+ 1841=4427	1109+1348+ 1460=3917	-510
Travels	235	302	67
Materials	29	56	29
Services	159	98	-61
Bought items	195	375	180
Income (1000 FIM)	4510	4340	-170
Tekes	880+1357+ 14631=3700	1100+1300+ 1300=3700	0
Industry	270+270+ 270=810	150+245+ 245=640	-170
VTT’s financing	220+55+ 266=541	139+177+ 242=556	15

A2.2.2 Planning and control of the Sokrates project

The planning, co-ordination and control of the Sokrates project involved VTT Information Technology Department in which TKO belonged, the focal organisation – especially its three research sections, the internal supervisory group called the research council, the steering group of the project and the management group of the whole Finsoft program, in which both the laboratory director and the vice director participated.

The steering group of the Sokrates project did not control heavily the work performed by the researchers, but was instead looking for the coming industrial application of the results. This is described in the external evaluation report of the Finsoft program as follows [Guy et al. 1991].

> *“Each project had a support group that included, among others, a number of industrial representatives who paid a relatively modest sum in order to sit on the Support Groups. The support groups were, in effect, a technology development and diffusion mechanism’.*

In the strategic plan of the software engineering section of TKO it was emphasised that the aim was to apply and carry further the results of basic research, such as the Draco system evaluated in [Kalaoja 1988]:

> *Seppänen, V. 1988. Software engineering section’s strategic plan 1988 - 1993. “As a part of research on automating software reuse processes, the section aims at applying the results of basic research on transformational systems.”*

The expectations of the members of the Sokrates steering group were somewhat different, but the goal of a more systematic, automated and manageable approach to embedded software development was shared:

> *Karjalainen, J. 1988. Personal R&D diary notes. 112 p. The first steering group meeting of Sokrates, 14.6.1988.*
>
> *“[Firm T]’s expectations: use of the SA method, support ‘close to the code level’.*
>
> *[Firm V]’s expectations: faster projects, faster software development, trial use of SA/SD, software configuration.*
>
> *The expectations of [firm S]: costs and elimination of errors important; automated code generation might be useful in simulation, software factory concepts; own operating system in the future (small).*
>
> *[Firm W]’s expectations: documentation support, semi-automatic code generation; software factory thinking.*
>
> *[Firm Nm]’s expectations: the use of SA, solving of maintenance problems, easiness of configuration.”*

Tekes, as the main funding source, considered it important to bring researchers and practitioners together, and to help to initiate early enough development projects based on prototypes developed in Finsoft.

The market of this kind of projects were seen as large as fifty companies, a considerable number of the total number of firms developing software-intensive products in Finland in the turn of the nineties:

> *Insinööriuutiset 23.3.1988: “Although Tekes will not aim at developing products but prototypes, the purpose of the industrial pilot projects to be carried out in the middle of the program is to get researchers closer the real world, in order to evaluate the prototypes. About 50 companies will be involved in the program.”*

The aim of the code generation researchers was to develop original methods and tools to address industrial embedded systems development as a whole:

> *Insinööriuutiset 23.3.1988: "The development of embedded systems by using traditional methods and tools is expensive, slow and difficult. One of the basic ideas in VTT's [Sokrates] project is to integrate different design and implementation phases to each other, to analysis tools and to commercial implementation tools. Methods and tools will be developed [by VTT] in the research for all these tasks."*

The following piece of the case data from 1989 indicates that the steering group was already looking towards the industrial application of the results:

> *Minutes of the Sokrates steering group meeting, 1.3.1989: "Regarding the continuation of the development of the code generator [in 1989], the goal [set by the steering group] is to be able to apply the generator in limited industrial usage".*

The planned industrial use of the results of Sokrates included pilot studies carried out both within the project and as distinct activities, Frame A4.

Frame A4. Early planning and control of the Sokrates project.

Minutes of the Sokrates steering group meeting, 12.10.1988.

"Discussions were conducted mainly on the evaluation of the developed methods by the pilot study. It was decided that the pilot system can be developed by the project group, but the industrial viewpoint must be taken into account in its design."

Tekes application Dno 25/32/88/TKO, 28.12.1988, Hannu Hakalahti. Appendix: Project plan for the period 1.4.1989 - 31.3.1990, Ari Okkonen, 7.12.1988.

"Utilisation of the results will be done in separate projects where tools are developed for industrial embedded systems developers, based on the pilot system produced as a part of the Sokrates project. This pilot system can be used for demonstration and small-scale utilisation".

The management of TKO saw code generation as a central research and development topic. Expectations of the usability of the results were high, the laboratory director and the department manager were looking forward to use the research results to carry out fully contractual projects:

> *TKO's annual plan 1990, H. Hakalahti, 15.11.1989. "In 1990, ... design automation [Sokrates] ... will remain as a central software engineering research area".*
>
> *TKO's long-term plan 1989 - 1994, 19.4.1989 by Hannu Hakalahti: "Embedded systems code generators for certain constrained application areas will become available at the end of the planning period (1994). [Contractual] product development projects serve as the application laboratory for the methodological research being carried out at TKO".*

TKO's 1989 results review (draft), department manager Esko Heikkilä, 30.11.1989. Research focus (d) [stands for "good"]: "Focus on embedded systems and research on new technologies that will be used in them meets the national challenges and creates a good basis for versatile application development".

This technology diffusion strategy was justified in the strategic plan of TKO from the viewpoint of the expected purchasing policies of the different types of its customers as follows:

TKO's long-term plan 1991...1995, H. Hakalahti, 22.3.1990: "[TKO participates in industrial R&D when]: a company wishes to have an external viewpoint to the development of its products, a company lacks expertise possessed by TKO, a company wants to have the latest technological knowledge, or a company wishes to manage market dynamics through R&D subcontracting".

The first outburst of disagreement between the management of TKO and the code generator researchers took place already in 1989. One of the code generation researchers, assigned to an internal parallel systems interest group, saw himself as a "clown" with regard to his responsibilities versus his formal authority to market research results and manage the resources needed to carry out projects:

Memorandum on TKO's Schaffner 22. - 23.3.1989. [Code generator researcher]: "An ordinary researcher is not allowed to talk to industrial managers. The section head decides by himself about the use of the section's resources. The [concurrent systems interest] group should not study the theoretical foundations of concurrent systems. I am not satisfied with the role of a clown. I'll quit now."

The remark that the group should not focus on the theoretical foundations of concurrent systems indicates some disagreements within the group. The code generation researcher describes the activities of the interest group from his viewpoint as follows: "The parallel and real-time systems interest group, the so called R Group, was established at TKO and I started to manage the group. One of the activities of the group was to help to organise continuing studies. We arranged, together with the great theorist [one of the members of the interest group] examinations on textbooks … I also performed a licentiate course titled to Theory of Automata (Gosh!)."

The steering group of Sokrates was not satisfied with the management of the project - concerning not the theoretical foundations of the research, but the question of how the plans, state and results of the project were made available to the steering group in technical and managerial documents.

Jukka Karjalainen's R&D diary notes 1988. 112 p. Minutes of the steering group meeting 22.11.1998: "Limited number of documentation criticised."

Minutes of the steering group meeting 18.1.1990: "By the next meeting, a list of the documents to be written during the next period must be provided. An updated list of documents must be presented starting from the next meeting."

Minutes of the steering group meeting 23.3.1990: "The plan of the next period must show in details the concrete tasks and reports that will be done before the next meeting."

Minutes of the steering group meeting 21.6.1990: "The agenda of the next meeting should include a specific item in which the documentation plan will be discussed."

Minutes of the steering group meeting 1.11.1990:"It was noted that the status report must in the future absolutely be sent to the steering group before the meetings."

A2.2.3 Evaluation of the Sokrates project and its results

Finishing of the Finsoft program involved several formal evaluations of the research results and process. One evaluation of the Finnish information systems and computer science research, initiated by the Finnish Academy and carried out by famous foreign scientists, had already been completed in 1990. The management of TKO included Sokrates in this evaluation:

Veikko Seppänen's R&D diary notes 1990. 5.5.1990: "[It was] Agreed that [the Sokrates project manager] will give a fifteen to twenty minute presentation on design automation (including slides and the Sokrates video), Sokrates demonstration possible."

The evaluation was very negative for TKO as a whole, and for the Sokrates project in particular. The project was deemed to deal with known problems and scientifically uninteresting solutions [Aho et al. 1990]:

"The Computer Technology Laboratory is clearly not oriented towards research. … Consequently, the best opportunities for the transfer of advanced technology and methodology from the laboratory (TKO) to industry could be missed. An example can be found in the software engineering section, where SA/RT was chosen because it is frequently used in industry. The project SOKRATES is even built around this technique, without the concern that more advanced approaches (VDM, Z, SOLE, or others) would lead to a technology jump in industry".

This view was neither accepted by the Sokrates researchers nor the managers of TKO, but was seen as untrue and unfair, based on a misunderstanding of the evaluators concerning the role of VTT. The results of Sokrates were considered as very promising and useful by TKO itself:

> *Evaluation of TKO's results from 1990, H. Hakalahti. "A pre-competitive version of the Sokrates-SA method is in the documentation phase. Industrial Sokrates-SA piloting carried out [in 5 companies that are listed, the firm Nm has] acquired for funding [from TEKES] to apply Sokrates-SA together with TKO".*

A group of British technology management and transfer experts evaluated Finsoft later in 1990. Sokrates did very well in the evaluation, this time the pragmatic aspects of the project were seen as a reason for success – also at TKO. The evaluators judged that the results were largely based on existing industrial practice, but also comparable to the best European results of academic research on automatic programming [Guy et al. 1991]:

> *"SOKRATES was an ambitious project in terms of scope, though it came to adopt a more pragmatic approach***.** *The work in this area is on a par with world level of achievement. The SOKRATES works stands comparison with some of the best automatic programming work in Europe. … Apart from developing its own real-time operating system, it built on existing industrial practice. This enabled good results to be achieved.*
>
> *The evaluators were particularly impressed with the ... SOKRATES ... projects, all of which demonstrated an admirable understanding of current scientific knowledge and methodological approaches in use within the international community".*

As a whole the evaluators saw the output of VTT in the Finsoft program poorer than what was anticipated [Guy et al. 1991], which further emphasised the weight of the positive evaluation of Sokrates:

> *"Work performed by the various units of the Technical Research Centre of Finland scored consistently less than projects performed either in industry or in the academic sector".*

The laboratory directors of VTT responsible for several Finsoft projects were not satisfied with these remarks. They emphasised the aim of industrially applicable results, i.e. the technology transfer goals:

> *Peltola, E., Hakalahti, H., On the participation of VTT in the software technology program (FINSOFT) and its evaluation, memorandum delivered to M. Mannerkoski, J. Forsten, E. Heikkilä, 20.8.1991."... the critic [of VTT's achievements in the program] by the British group involves especially the external impact of the projects ... the questions are such that basic research oriented projects in new areas do well, by definition. The industrial usability of the results especially in Finland is not enough emphasised in the questions."*

An industrial evaluation was also performed for the program. Sokrates earned very high scores indeed in this evaluation, as shown in Table A4.

Table A4. Industrial evaluation of the Sokrates project.

Characteristic of the project	**Score** (A best, E worst)
Goal setting	A
Timeliness with regard to needs	A
Timeliness wrt. technological readiness	A
Usefulness	A
Time-effectiveness	A
Interest of the support group	A
Validation of the applicability	A
Cost-effectiveness	B
Industrial impact	C
Goal achievement	C
Effects on competitiveness	C

In addition, the project was commented verbally by nine of the members and six of the managers of the steering group firms. The comments are summarised as follows in [Saukkonen 1991]:

> *Evaluation of the usefulness of the project (the steering group's view): "The goal was ok, but special method development was too much emphasised, because a particular version of the SA method was developed. The project group could concretise the results further and make the developed tools simpler. The project group should develop its activities based on the comments given by the support group and ensure that agreed tasks will be implemented."*

The results of the formal evaluations of the competence development phase can be contrasted with the case data acquired several years later, in this research, by interviewing the different parties involved directly in the Sokrates project. They include representatives of Tekes, members of the steering group, the code generation researchers, their colleagues at VTT and the former line managers of TKO.

A2.2.3.1 Tekes – the limits of technologies must be tried

The representative of Tekes in the steering group of the Sokrates project from 1988 to 1989 emphasises that "ideas are developed in research programs, not products … I do not consider immediate application of the results [of joint applied research] as important". However, in projects that aim at combining different ideas it may in his opinion be possible to find technological limits, which is very useful information for industry.

The experiences of Tekes from the Finprit program that preceded Finsoft were, according to the interviewee, rather negative concerning industrial help and feedback for guiding and evaluating research programs. Therefore, the opinions of researchers played a central role in creating the Finsoft program. The sole number of industrial representatives in the steering group of Sokrates was considered as a guarantee for the kind of industrial guidance that could be reasonably expected.

The diffusion of new ideas and possibilities during the project was more useful than immediate application of results – “it is no use to speculate on great the grandchildren [of applied research projects]”. Therefore, the “somewhat mixed goals of the [Sokrates] project that can be seen from the final report” could be tolerated by Tekes. The evaluations of Finsoft were heavily criticised by the interviewee: according to him they focused too much on the comparison of the initial goals and the final results, in a context where “the best that can happen is the changing of the goals during the project as a result of learning”.

After almost ten years, the promises of fully generic code generation technologies are still rather poor. The other representative of Tekes in the Sokrates steering group from 1989 to 1991 pointed out, however, that the promises were considered as good in the late eighties. Some of the ideas had already been tested in industrial pre-studies, such as the Speco projects conducted for the firm K. Therefore, the low number of the spin-off and continuing pilot projects of Sokrates in the early nineties was a pity also from the viewpoint of Tekes. One of the reasons might have been the weak role of the steering group. On the other hand, the researchers were in the opinion of Tekes “enthusiastic”, and only “in the final phase the pace slowed down a little”.

Because “Sokrates was loaded with expectations”, which were realised only partially, it was obvious that the formal evaluation of the results included also some negative remarks. Yet, Tekes did not know anything for example of the ultimately conflicting viewpoints of TKO and the tool vendor firm I. It was only “wondering why the firm was not that keen on applying the results”. Tekes considered the commercialisation of the results from a very pragmatic perspective: although the role of VTT “is not to go too much towards commercialisation, there are sometimes no alternatives”.

However, to be able to commercialise the results, a larger number of spin-off projects would have been needed. Later on in the nineties, “when the object-oriented approaches to component-based software development emerged, it became more difficult to continue investing in commercial code generation technologies”, but at the end of the eighties it was “a card that needed unquestionably to be played.”

A2.2.3.2 Industry – interested in learning the limits

Although the steering group criticised some managerial aspects of the Sokrates projects, it was interested in the topic and keen on seeing the results. Yet, many of the members of the steering group that we interviewed told that they did not consider to use the results immediately or at all.

Moreover, their interest in code generation from SA/SD specifications fell rather rapidly in the mid-nineties, in part due to the new object-oriented design methods that were becoming popular. In the late eighties "experiences from the comprehensive use of the SA/SD method" were still sought, in addition to new ideas on code generation. "A specific approach to the use of SA/SD [the Sokrates method], … information on the possibilities and tools that were available for code generation" and "knowledge of software development processes" were achieved according to the industrial participants of the Sokrates project.

Some steering group members suspected that the change of the "mind set of ordinary programmers" towards the use of automated programming tools was, after all, too difficult. Code generation was seen as a multi-dimensional problem where not only different types of programs but also different design styles and methods would have needed to be mastered. Some firms, such as N that actually took code generation as a part of its software development process, considered it "easier to control" their own code generation methods than such generic approaches as Sokrates.

Industrial code generation methods were often much more limited, but Sokrates was on the other hand suspected to "take a too big bite", instead of preceding in smaller and more concrete steps and acquiring feedback after each step. Interestingly, this is much the same difference of viewpoints as in the Speco projects, but in a reversed way – the industrial people would have liked to see smaller endeavours at the expense of a comprehensive vision.

The code generation researchers were claimed to be too self-satisfied, they "had their own firm views that were difficult to affect by a single party of a large joint project ". Yet, the opinion of Tekes on the high motivation of the researchers was shared by many industrial parties of the project: "I still remember the guiding star drawn by [the project manager of Sokrates] at which they were aiming".

The software designer of the firm N, who had built a simple code generator for producing C programs from SA/SD state machine models in the Winter 1988 - 1989 joined the knowledge engineering section of VTT in 1989. However, he did not work for the Sokrates project, but started to carry out knowledge engineering research instead. He points out that "thinking afterwards, I had surprisingly little to do with VTT, if the Sokrates project was already going on then [Winter 1988 - 1989]". It is even more surprising that TKO did not take any advantage of a person, who had just built an industrial code generator for a firm that took part in the Sokrates project.

Veikko Seppänen, the head of the software engineering section, initialised the recruiting of this person. However, he was actually affiliated with the knowledge engineering section that did not yet provide any human resources to the Sokrates project in early 1989.

The designer was himself "very interested indeed to continue towards a full code generator [instead of a simpler state-machine based generator]. No one apparently even asked me to join the Sokrates team. They seemed to have all scientific aspects [of code generation] already sorted out. I was interested in doctoral research, but ended up in choosing another topic." The code generator implemented originally by this person is still in use at the firm N. Its power lies in that "it does not try to be more than it really is. It does not have any fancy theories behind and its is tailored to produce code from certain kinds of state machines for the software of a certain company … [but this leads to] benefits in efficiency".

Two other firms of the steering group built similar kinds of application-specific program generators in the nineties. At least yet another two firms purchased and utilised the commercial code generator sold later by the firm I. Following of the Sokrates project was considered as "experimental" by these firms that wanted to use commercial tools in their actual software development process. The "extensive documentation" of the project was pointed out as one of the main results from the experiment. The SA/SD method was used for several years in the nineties by the firms which became familiar with it during the Sokrates project, until "the time ran over the method". One reason for this might have been that "the top-down approach to design [starting from a high-level specification and omitting implementation details] is not possible, because a certain context for reusing existing software components is always available."

The object-oriented methods that challenged structured methods as an alternative paradigm in the nineties provided explicit means to make use of existing software components. This situation can be contrasted to the design of completely new embedded software for industrial applications in the late seventies and early eighties, such as the work done by some of the key members of the Sokrates project. Legacy software was one of the reasons mentioned also by the firm K for not utilising the results of the Speco and Sokrates projects – the design of a main revision of the software embedded in the company's products had "proceeded rather far, until even close to appropriate tools were available". The timing of the development of the code generation technologies did not thus match with the planned schedule of the new products for which new software needed to be produced.

Another barriers for taking the results in use mentioned by the steering group members were the not-invented-here problem, the lack of measures for the possible benefits and the generator being "too much a black box that magically produces code". Some programmers of the participating companies had complained that "the basic principles and functionality of the generator were not visible enough", and that "the method required too detailed modelling". The members of the steering group claimed that they "did not have enough power to remove these doubts", especially if the programmers were professionals interested in "playing with the code".

A2.2.3.3 Researchers – the limits and possibilities were shown

The researchers commented the results of the Sokrates in 1998 by stating that "the main initial goals were achieved; it was possible to generate the full code of an embedded system from a specification". Co-operation with the firms participating in the project was considered as "at least reasonable". The four diploma theses, nine international publications and the first software-related patent ever gained by TKO were also mentioned as the main achievements of the project. The use of the Ada language as a part of the Sokrates method after "a hectic argumentation" of this decision was seen as a failure. Although some members of the steering group had spoken for the use of the C language instead, they "did not oppose the use of Ada".

The project group thought that they learned new skills in compilation techniques and in understanding the behaviour of real-time systems. The arrangement initiated by the management of TKO, by which the exchange visit of another researcher was supported, was considered as "a big mistake", because it consumed financial resources by which some of the Sokrates tasks that were not finished could have been completed. In the final report of the Sokrates project the researchers describe the results and the needs of further work in Spring 1991 as follows:

> *"The embedded systems design automation project (SOKRATES) has produced practical methods, languages and tools, by which the design of embedded systems can be made more efficient and the quality of its results can be improved. The quality of the code produced by the code generator [developed in the project] is fully suitable for production use, with respect of its size and speed."*
>
> *"The most remarkable need with respect to the practical utilisation of the results is to make the code generator available for industrial usage, in a PC environment. This is planned to be carried out in a distinct applied research project".*

The hand-written comments of Veikko Seppänen in the manuscript of the final report (Frame A5) indicate that he saw some of the conclusions as a twisted reality used to explain the results in the best possible way. For example, five of the eleven conference papers listed in the final report were actually produced by the exchange visitor, who only got financing for his trip from the Sokrates project and did not actually took part in the research. As another example, the close co-operation with the green research program of VTT emphasised in the report meant, according to the hand-written comments, that the same people worked in both projects.

The demonstration system built by the project, a toy elevator constructed from Lego pieces, was not considered as very convincing by Veikko Seppänen in 1991 – the demonstration of a change made for the control software of this toy elevator, by using the generator, was seen as too simple for reasons that were not explained. Making of this change was one of the highlights of the Sokrates video used to illustrate the results of the project.

Frame A5. Extracts from the summary of Sokrates results.

Sokrates project, administrative final report, 18.6.1991, Dno 7/53/91TKO.

“2.4 Results. Languages: The CML (Component Modelling Language) language developed for describing real-time phenomena has been created based on the Ward-Mellor and Ada languages.” **Veikko Seppänen’s hand-written comment: “Yet another impractical theoretical language.”** “Rea-C-Time extends, by the use of macros, the C language to include state machines and non-interruptable (non pre-emptive) scheduling. Other results: Dr. […] developed ... new methods for making reachability analysis more efficient. The results produced by the financing provided by the Sokrates project are shown in the section 2.6. Publications and reports.” **Veikko Seppänen’s hand-written comment: “Bullshit connection”.**

“4. Publication of the results. The results of the project has always been presented to the support group in its meetings, which have been held every three months. INSKO’s courses and company-specific training has been given on the methods and languages. The results of the project have been presented in the Finsoft seminars (3 times), in the STeP workshop in Fall 1990, in Blanko workshops in Fall 1989 and 1990, in a presentation given as a part of the Oulusoft training program, and in articles published in Tekniikka&Talous and Kaleva. The toy elevator was presented in Heureka’s exhibition “Do machines think?”, for three months in Fall 1990. The elevator was also presented in TV’s children’s program in the Uusi Suomi newspaper".

"5 International co-operation and its utilisation: 4 course travels [1 in OSTU]: CASE Symposium and company visit in Boston 1990, SD course and a company visit in London 1988, ESPRIT summer school in Nice 1989, software quality course in Methuen, USA, 1990. 3 conference travels [all to Euromicro: Zurich 1988, Köln 1989, Amsterdam 1990]. The project supported […]’s stay at Telecom Australia 9/88 - 6/89".

"Conclusions. In order to test the [developed] technology, an own pilot system (a toy elevator) was built. The development of the whole control software for the system too 6 man-days. By using this system, it could be demonstrated how a certain functional change of the requirements could be implemented as a visible system characteristic in 20 minutes.” **Veikko Seppänen’s hand-written comment: “If [the designer] knows elevators.”**

A2.2.3.4 Colleagues – but was it done in a decent way?

Although the Sokrates project group was commented to be rather self-sufficient, it did have many discussions with the colleagues working at TKO and some other VTT laboratories. Several other researchers were involved in the preparation of the project as a part of a larger proposal of an embedded systems research program. Later, both the research council and the parallel and real-time systems interest group of TKO were available for changing thoughts with colleagues. Even the management group of the Sokrates project was at some point combined with the management groups of a few other Finsoft projects. Yet, the general feeling of the colleagues was that the code generation researchers were united against other people, a homogenous group that even dressed similarly – in special outdoor trousers and coats called the “Sokrates uniform” according to an inside joke.

In principle, some of the topics being pursued by the colleagues could have been associated with code generation. The above mentioned formal analysis is an example of this kind of a possibility, another example is the so called executable specifications. However, integration was not easy managerially because of the strong role of individual projects.

Moreover, some of the colleagues claimed that it would have been difficult also technically, in part because the code generation principles were “at the alchemist level”, i.e. seen defined in an ad hoc manner and not explicitly documented. One colleague suspected that the exact documentation of the approach “would have required too much work” and was therefore of no interest to the code generation researchers, who not only aimed at rapid exploitation of the results but also did not have much experience from scientific research. Another claimed that the documentation of the “SA theory” was impossible:

> *“It is worth noticing that SA/SD lacks some very essential aspects: the so called thread alias the execution semantics. Therefore, it was an orthogonal approach to the others. It is not wrong, but it lacks a mechanism …In my opinion, the group had clear interest in theoretical and scientific questions. At the beginning, the group reacted positively to critic. There were many things … that the group treated well … later the opinions [of the group] become somehow distorted and tangled. My interpretation is that they were attempted to be forced into the “SA theory” – actually there is no such theory. SA is just a graphical notation and some fundamental but simple things”.*

These colleagues preferred the systematic, “decent” research approach based on analysis and application of related research, use of known theories and research methods and scientific publication of the results. Some of them saw the question as an educational problem: “One thing is that I and [a colleague] had a more theoretical education gained at the Helsinki University of Technology. This was then, and still is, an inflammable topic in Oulu”. The code generation researchers themselves thought that they followed the decent approach, but that their main goal was to produce industrially applicable results and not only research papers. The colleagues thought that both the papers and true industrial references would have been necessary, to “be credible” both scientifically and industrially.

One of them, however, shared the opinion of the code generator researchers that the management of TKO was not willing to support the commercialisation of the results, because it had a negative attitude towards taking of risks. The idea of generating C programs from SA/SD models was good also according to the colleagues, except that the power of the method was seen in its informal nature – the method was characterised as a system “sketching technique” by one of the colleagues. Code generation required very strict system models to be built, which was against the principle of informality. While the code generation researchers told that they performed thorough studies of traditional compilation techniques, one colleague claimed that they did not learn much from the studies: “if one does not ascertain anything, it is easy to believe having invented everything”.

The use of the Ada programming language as a part of the Sokrates system modelling method was seen “clearly harmful”. The use of C instead would have been much better. This was the case both in the commercial code generator of the firm I and the Reagenix generator of TKO developed later in the nineties.

The choice of the toy elevator as the main demonstration system of the project was considered as “excellent”, but one of the colleagues claimed that “it does not serve as an example of the generation of real-time software, although this was claimed. The software was modest in terms of computation and it was executed by using the primitive semantics that are completely inappropriate for real-time systems”. Scientifically, a colleague viewed the work as inexperienced: “Articles in the Step seminar do not have much of any value … no scientific results were produced in the project”. Comparison of the approach to Draco and another transformation system was seen as a wrong choice, and the use of a research trainee for this strategically important work as a mistake: “Another, better approach would have been to write an article for example for the IEEE Transactions on Software Engineering journal, to gain feedback from real top researchers”.

The management of TKO was criticised for tolerating the lack any scientific approach in the Sokrates project. One of the colleagues suspected that the reason for the limited interest in using the generator was that it was technically poor as a tool, i.e. it did not function well and was badly documented. While the evaluators of the Finsoft program characterised Sokrates as “an ambitious project in scope” in 1991, a colleague characterised is as too ambitious: “The fact that [the firm N] built a simple but widely used code generator was an important matter that should have taken into consideration”. The aim of usefulness was actually seen as one of the pitfalls of the project:

> *“The project maybe also illustrates the dangers of the so called useful research – in other words, the kind of research that we carry out [at VTT]. There is no shortcut to the truth. The goal to be practical did not solve any on the problems of the [Sokrates] project, but some theoretical investigations would have solved. In my opinion the project clearly drifted to the thinking of Lysenko, which is the danger of publicly funded contract research in more general terms … The theories of Lysenko were absolute nonsense, he collapsed the good reputation of the Soviet genetics and finally destroyed the farming”.*

A2.2.3.5 Managers – VTT cannot commercialise the results

The interviewed managers of TKO pointed also out the firm K, especially its R&D manager, as the father of the code generation research. At some point in the early eighties he “draw a picture of an automated machine that produced software”. The managers considered that code generation was “well marketed from the financial point of view” to the Finsoft program.

The managers opinion was that “the pragmatic approach taken by the Sokrates group was all right” and that the difference of viewpoints regarding the “missing theoretical foundations” between the project group and some other researchers of TKO was not a big problem: “The Sokrates project was based on industrial work [the Speco project], the timing was correct, and the project was not too scientific”. In other words, the management viewed the project more as engineering than science.

The relations to the firm I worried the management: “[The firm] saw VTT as a competitor, so that Sokrates would have spoiled their markets … There were some hard discussions …. [The firm] joined most likely the support groups of the Sokrates and Iptes projects in order to know what we were doing, not for really wishing to utilise the results or co-operate”. The management hoped that some other tool vendor would become interested in the Sokrates results, but “we did not manage to create a strategic alliance with any CASE tool vendor … [and] the idea of VTT selling the tool was not good.” When the work after all drifted towards the development of a packaged software tool, a failure could be expected, because at VTT “the traditional approach has been to take rather small risks by narrow financing”. A good touch to the end market would have been needed, as well as some business decisions: what are the constraints of this kind of work at VTT, how big a risk will be taken, for how long the results will be waited, what are the practical constraints of the tool?

Answering to these questions would have been difficult, because VTT as a service organisation has typically relations “only to the R&D departments of its clients, not to all decision makers”. An additional issue in this regard was that the “Sokrates way for doing was the one of a prima donna - not to look around, because the others do no not exist”. Although the researchers “had the artistic freedom for doing the research, … [the managers] controlled the direct marketing of the code generator. Our viewpoint was pragmatic, selling was not the business of VTT”. The managers tried to convince the researchers to establish their own company to commercialise the results, but without a success. They concluded that the researchers were not real entrepreneurs, after all.

According to the managers of TKO the Finsoft program was heavily researcher-driven, “industrial opinions were not taken that seriously, the steering group had little controlling effect”. The program-level steering group “criticised that Tekes had already made decisions concerning the projects to be launched”. Afterwards, one could speculate if indeed the plans were realistic at all. However, such industrial examples as the Speco project indicated that something could come out. Code generation was seen by the managers as relevant even today, but “it was after all unrealistic to believe that VTT would develop a tool that affects considerably the whole industrial software development process”. The managers did not recall in the interview the reasons for financing the exchange visit from the budget of the Sokrates project, but suspected that “maybe it was difficult to find money for [the] trip - arrangement of such trips was very bureaucratic at that time, and perhaps this was the easiest way. There was, of course, no real link between [the visitor’s] work and Sokrates”.

A2.2.4 Marketing of code generation during the Sokrates project

The situation of TKO in the domestic market of the research and development of embedded computer systems was rather good in the late eighties and early nineties, in the sense that it had a good share of the market and many existing and potential customers. Moreover, it was quite proactive in establishing and maintaining its customer relations. The definition of the strategic business areas of TKO was done in 1990 based on the blue and red portions of the project portfolio, Frame A6.

Frame A6. The SBA definition of TKO in 1990.

TKO's long-term plan 1991...1995, H. Hakalahti, 22.3.1990. TKO had 1989 about 25 industrial customers. The volume of industrial income was about 6 MFIM. 65 persons worked for TKO. The annual volume of embedded systems development in the national electronics companies is about 300 man-years, which is equal to about 1000...2000 MFIM/year. TKO's portion of the subcontracting market of 120 man-years is about 20 man-years or 16 %.

TKO's business plan, J. Karjalainen, 1.8.1990.

"SBA1. Applied research projects. "Customers include Tekes, (KTM) and companies. The needs of companies involve following of top-level research, identification of new ideas based on the follow-up, and development of thereby possibly produced prototypes into commercial products. The 'technologies' of SBA1 include embedded systems skills, contacts to outstanding international research and development of a framework for national co-operation. Development of business opportunities: TKO is a market leader in SBA1. Strategic goals: CASE and AI tools are best in Finland".

"SBA2. Development projects. Customers' needs involve rapid development of mass-products or processes, adjust the load of their own development resources, gain special knowledge of product development and external insights in addition to their own design expertise. The 'technologies' used by TKO include embedded systems computer technologies, project management principles, usage of the results of research projects and good contacts with funding bodies. The laboratory is a market leader in SBA2. Strategic goals: A special goal is to increase the number of development projects carried out for machine and equipment manufacturers. The central technological expertise that will be offered include artificial intelligence and software engineering".

Marketing of the Sokrates project had succeeded excellently, in terms of the number of parties involved and the amount of external funding gained. The manager of the project made an overview of his principles to market research and development services in a planning meeting of TKO in 1988:

> *Veikko Seppänen's R&D diary notes 1988. TKO's Schaffner 19.9.1988: A [code generator] researcher's view on marketing: "How to make the customer understand that his needs can be satisfied by using a technology developed in a research project: by a concrete demonstration system developed as a part of the research; by successful goal setting (concrete, technologically demanding, e.g. a piano playing robot). How research can help a customer to understand his needs: by showing of a concrete vision, mutual "analysis" discussions.*

How research can help a customer the believe in our skills: by a successful project, writing of articles, giving professional training (INSKO etc.), application of a demonstration system. How research can make a customer to give us an assignment: by good personal relationships (established in the steering group of the project), marketing of the use of technology, not resources."

TKO started to market code generation related red projects to its regular customers and the firms participating in Sokrates very early, with a goal to gain industrial references on the applicability of the approach. The expectations concerning the use of the generator were mostly positive:

Veikko Seppänen, Memorandum of a regular key customer meeting (the firm K), 11.1.1990: "Potential co-operation topics: pilot use of Sokrates. The goal would be to gain experimental information on the usability of the generator, the quality of the resulting code, the size of the code (vs. hand-written code), and the effects of differences of the Sokrates-SA method and the basic-SA method".

Jukka Karjalainen's R&D diary notes 1990, 164 p. 12.1.1990, Internal Finsoft meeting: what after Finsoft: "Customer-specific versions of Sokrates, Sokrates courses, textbooks. 30.3.1990, Internal Finsoft meeting: Next: code generator for the 8051 processor?" 30.3.1990, Marketing meeting (the firm S): "Own approach to code generation in use (produces code skeletons), does not believe in automatic code generation for several years, Sokrates produces too much code". 18.12.1990, discussion with an electronics firm: "Sokrates and code generation are interesting".

Veikko Seppänen's R&D diary notes 1990. 4.5.1990, marketing meeting (the firm S): "Sokrates trial use interests, Ada-type mini specifications are being used". 9.8.1990, marketing meeting with a telecom company: "Conclusions: use of Sokrates? Production of test cases from specifications."

The annual review of TKO from 1990 includes the first public references to red projects based on Sokrates results:

[R&D manager of the firm Nm]: "When developing our new [... product] we have adopted the design, implementation, and testing methods of the Sokrates project. In my opinion we have got clear and efficient improvement in our design".

Within the software engineering section, the outlook for subsequent technology transfer projects, after the "transferring of the understanding of technology" in Sokrates, were considered as very good in 1991:

Software engineering section's long-term plan, V. Seppänen 9.3.1991: "The possibilities for the application of the [Sokrates] results and starting of continuing projects based on them are good, discussions have been carried out with several companies and VTT's laboratories.

> *A plan on the application of the results and the preparation of continuing projects is being done. The goal is a 3 - 6 man-months' volume in 1991 - 4. The total technology transfer goal is 3 - 5 pilot projects in 1991 - 4. The main target is machine automation systems."*

The marketing plan of TKO for 1988 - 1989 aims at "the diversification of the customer base to include at least four big regular customers and several smaller customers".

One direction of diversification was machine automation, intended to be one of the main customer segments for red code generation related projects. Market analysis of selected machine and equipment manufacturing companies had been carried out for TKO as a part of a VTT level marketing project in 1989:

> *Market analysis of selected machine industry companies, M. Oikarinen, 31.3.1989. "It is difficult to evaluate the most interesting products of the industry from the viewpoint of [the VTT Computer Technology] laboratory. The volume of research in the industry has increased more rapidly than in all industries on the average... R&D expenditure was 1,4 billion FIM in 1986."*

Forty firms had been put into a marketing mailing list and sixteen analysed in more details. Two years later from this, in 1991, TKO planned that "the Sokrates code generator will be modified for the needs of machine automation firms". It co-operated at that time with ten machine and equipment manufacturing firms and with fifteen electronics firms. Several marketing planning meetings were conducted by the line managers and code generator researchers, to support the transfer of the results of the project into industry on a fifty-fifty basis required by Tekes.

Alternatives to commercial tools that could be integrated with the code generator were sought, due to the growing disagreement between TKO and the firm I. The firms N and S were developing their own code generators, but were not applying the results of the Sokrates project directly. The managers of TKO urged the researchers to plan for continuing red projects, which were seen as the products of the laboratory:

> *TKO's marketing plan 1990 - 1991, 2.4.1990. "It is very important from the viewpoint of the operating principles of the laboratory that the results of Finsoft will be applied in industrial projects dealing with the development of new products. From the viewpoint of marketing communications especially the Sokrates project is important."*

> *TKO's annual plan 1991, H. Hakalahti, 4.10.1990: "TKO is a service unit and its product type is project-based research and development done in its R&D area".*

> *Veikko Seppänen's R&D diary notes, 1991. 27.3.1991, internal Sokrates marketing meeting: "The present status: SA consultation for five firms including E and Nm. SPS exists as a product, the reference of [the firm N] is available indirectly, patent application?"*

"SIC package exists, the SIXA reference is available. Generator and meta tools: PC version could be developed on a 50:50 funding basis, could be given in use for universities, would require 6 man-months and two persons, about 600 kFIM: it was agreed that a project plan will be produced on this. Teamwork interface: [a telecommunication firm] could provide funding for about one man-month. Further research project proposals will be frozen. Reference sheets will be produced, market analysis will be carried out [by one of the researchers]".

Jukka Karjalainen's R&D diary notes 1991, 187 p. 8.4.1991, Sokrates marketing meeting: "Brochures will be produced, the prices of the Sokrates products will be defined, a market analysis will be carried out [by a code generation researcher], research projects will be prepared for the Fall season [of new Tekes project proposals]".

TKO considered VTT Electronics laboratory and VTT Machine automation laboratory as the most potential internal customers for the code generation research results. When evaluating the results of TKO from 1991, the department manager Esko Heikkilä requested that the "offering of own [TKO] expertise to other [VTT] laboratories should be improved". Yet, the laboratory director's report to the department manager only a few months after the finishing of the Sokrates project was cautious with regard to the future, realisation of continuing projects had not proceeded as expected:

TKO's half-year report 1991 (long version, H. Hakalahti, 1.7.1991): "In the software engineering area, the industrial application of the results of the Sokrates project is, despite of the good research results, not yet fully satisfying".

In the internal Finsoft meeting as early as in October 1990 it had been reported that a six-month spin-off project will most likely start with the firm Nm, but that "the firms K and N are not interested" in such projects. The failure of the expectations concerning red projects as a whole was explicitly noted in the final internal project review of Sokrates in October 1991:

Jukka Karjalainen's R&D diary notes 1991, 187 p. 28.10.1991, Final internal Finsoft meeting. "... 50/50 continuing projects did not succeed".

A2.3 EXPLOITATION OF THE COMPETENCE

Exploitation of the Sokrates design method and the first prototypes of the code generator started as early as in 1989, while the Sokrates project was ongoing. However, the exploiters were not industrial customers as planned in the initial project proposal, but researchers of two blue projects carried out by VTT Electronics laboratory, a sister institute from which TKO what extracted in 1983, located at the same premises. The applications involved machine automation and design of application-specific integrated circuits.

A2.3.1 Early exploiter – VTT Electronics laboratory

One of the very first versions of the Sokrates code generator was used in the joint research project Synchro that was being carried out by VTT Electronics laboratory as a part the machine automation research program of Tekes in the late eighties. A diploma thesis worker and a programmer used the Sokrates approach to design and implement a computer-controlled hydraulic control system [Savilampi 1989]. From the viewpoint of TKO this exploitation created a white relationship to the Electronics laboratory.

The technological level of the code generator was still modest in 1989, depending on the type of the computer used, it took about an hour or even three hours to compile a rather small system model to a C program. There were also some problems in integrating manually implemented and automatically generated pieces of code together. The program generated in the Synchro project was, however, comparable to a manually implemented program in terms of its size and operational efficiency. Compared to an earlier design, the same functionality could be developed in less than one half of the time by using the Sokrates method. The programmer describes her motivation for using Sokrates as follows:

> *"I had used RT/SA already for some time in system design. The need [to use the generator] resulted from the problem of not being possible to utilise the results of earlier projects (I was the only software engineer and always over employed). Moreover, there were no [automated] links from specifications to code".*

Another joint research funded by Tekes and carried out in part by VTT Electronics laboratory test used Sokrates in the late eighties and early nineties. This series of projects called Sasic involved the design of application-specific integrated circuits. The Sokrates project team saw this engineering domain as a possible application area for their approach. The circuit researchers were at that time missing both high-level system modelling methods and automated tools to produce lower-level hardware description "programs" from system models. Their first attempts to understand the Sokrates method were rather frustrating:

> *Timo Leppänen, SASIC3 project, 9.11.1989: "I got a feeling that it was very difficult to use the Sokrates-SA syntax, because of the lack of any decent manual. It would be important in the use of Sokrates-SA to know not only the syntax, but also the semantic thinking."*

The firms K, N and Nt took part both in the Sokrates and Sasic projects, but their representatives in the steering groups were different and specialised in different technologies. Technically, the Sokrates approach appeared to be "infeasible for the design of the application-specific integrated circuits". In part this was due to the rapid evolution of circuit design methods and tools, which made it difficult to develop any integrated design environment.

Another problem was that although some circuit designs could be generated by Sokrates, their "quality (the number of gates in the circuit design) was a decade worse than the quality of manual designs". The reason for the poor efficiency was, according to the circuit researchers, that "a good result would have required more intermediate design steps to be taken … Then we tried it [to produce designs] directly [from high-level specifications]".

In about 1990 the circuit designers started to investigate a new and rapidly emerging design language, VHDL (Very High-Level Design Language). A Sokrates style SA/SD-based system design method and modelling language were used in connection with VHDL. However, instead of continuing to develop the Sokrates code generator further for their needs, the circuit researchers developed their own design tool called Velvet – the first version was implemented only in three man-months.

In 1992 the strategic plan of the Electronics Laboratory for 1993 - 1996 stated that "in control system techniques, the goal is to investigate in a laboratory environment and transfer into industry functional specification and code generation techniques". However, the rights to Velvet were after all licensed to a small company established by a university researcher involved in the Sasic projects, but it "did not become any killer application". It was used in a few research projects and by some engineering schools in the nineties – much as the Sokrates code generator.

In 1998, the experiences from the early Sokrates exploitation were summarised by a former VTT circuit researcher as follows:

> *"If the present commercial tools are compared to the SA/VHDL approach, … they are not particularly smarter, although they include some new features. [The only difference is that] the support organisations have been able to market them [better]. Speaking of the Sokrates/Reagenix technique, I just point out that both the method and the tool were built based on an assumption that the target computing platform of the code is given.*
>
> *Viewing this from the point of application-specific integrated circuits, the biggest constraints are the modelling of true concurrency and differences in hardware architectures.*
>
> *One of the essential problems in SA/VHDL tools, Sokrates/Reagenix and many other "productised tools" is that they do not support the reuse of already existing solutions. The threshold to start [system development] from scratch is too big".*

A2.3.2 Struggling with the focus of exploitation

After the Sokrates project had been finished, the main goal of TKO was to transfer the results into industry. Although technology transfer was planned to be done, it was not clear which kinds of activities would be favoured:

- red and blue projects for the further development of code generation either with tool vendors or industrial customers,
- red projects based on the existing code generation competence,
- selling of the rights of the code generator solutions to a tool vendor for commercial exploitation, or
- selling of usage licenses and training packages to industrial customers.

In the last two cases the license fees could have been used for continuing distinct code generator related research, in the last two cases this should have been done as a part of the projects – TKO did not make much of any profits from red projects, customers were charged on a cost basis. Expectations for the immediate spin-off projects of Sokrates were high at the beginning of the nineties, as shown in Frame A7. Tekes was prepared to fund continuing industrial projects on a fifty-fifty basis. A considerable set-back was that these expectations were not met after the Sokrates project had been finished in Spring 1991. This meant great difficulties in continuing the work to implement an industrially applicable version of the code generator.

Frame A7. Expectations on the exploitation of the Sokrates results.

[Guy et al. 1991] Appendix III: project notes (confidential), 24 p. "The researchers took a pragmatic approach, basing the notations for the models on RT-SA/SD, which is well established within Finnish industry. The project goals were mainly attained, and the project represents a good, solid contribution which has the potential for wide applicability. The results offer ... a realistic development from existing industrial practice".

Saukkonen, S. 1991. Finsoft-ohjelman tulosten teollisen hyödyn ja hyödynnettävyyden arviointi. Tekes, Helsinki. 75 p. Appendix: projectwise evaluations (confidential), 92 p. Evaluation of usefulness (project's own view): "[Firm Nm] has carried out the development of its new product using, almost entirely, the Sokrates technology. An operating system derived from SPS has been used as a part of a mass product [by firm N]. A demonstration system has been developed, where an industrial PC controls a toy elevator. With the help of this system it has been shown how a full elevator control system ... has been developed in 6 days. It has also been shown that a functional software change can be done in 20 minutes.

One of the two demonstration systems was in a heavy use in an exhibition at Heureka, without any functional problems. The SYNCRO project of VTT/ELE [that has used the Sokrates technology] has observed that the effectiveness of programming increased remarkably and the resulting code was more efficient than in manual coding. One of VTT/TKO's contractual project [that has used Sokrates] has reported that hundreds of PLM code lines have been produced daily. As a result of consultation and training, different parts of the technology have been taken in use in 5 - 10 companies. Demand for consultation and training has increased remarkably during the last year."

Veikko Seppänen wrote a memorandum "What Sokrates is 1.4.1991?" for an internal meeting of the managers and the code generation researchers, where the exploitation of the results of Sokrates during the coming few years were discussed – maybe it was just a coincidence that the date was the All Fools' Day. The memorandum is a rather extensive summary of the managers' view to the possibilities to exploit the Sokrates results. It specifies the following exploitable items: the Sokrates-SA design language, the Sokrates-SA design method, the ReaCtime language, the SIC, SPS and SPI system solutions, the Sokrates generator and the meta tools used to define the languages. Several kinds of activities that could be taken to utilise and develop further these items were discussed in the memorandum. The following activities were listed in the memorandum:

- existing items that could be "easily sold and tailored for companies";
- the use of the Sokrates method and tools in industrial product development projects by a customer company itself or by TKO on behalf of the customer – the volume of one to six man-months was expected for each such project;
- new kinds of tools "based mostly on Sokrates" and developed for industrial customers, such as new operating system interface solutions based on SIC, productisation of the SPS operating system "together with an appropriate partner" and replacement of Ada as the mini specification language of the Sokrates-SA method by the C language;
- porting of Sokrates to a PC or a portable workstation environment,
- integration of Sokrates with new SA/SD tools or with object-oriented CASE tools;
- generation of other languages than C from system specifications – including languages used in non-embedded software applications;
- the use of Sokrates in new research projects – about ten possible topics involving three other VTT laboratories are listed;
- the further development of Sokrates in new research projects: "idea papers or proposals related to this have not been written, but would be needed as a starting point" – a few ideas are discussed;
- small-scale consultation for companies and VTT laboratories: "a more detailed marketing plan should be written … and the consultation packages specified, possibilities [to incorporate consultation] to the own customer projects of TKO should be evaluated";
- training that "should not be done on a cost basis and should support other activities, such as R&D projects and long-term marketing"; and
- textbooks that "could be best written as a private hobby".

The memorandum ends with a brief analysis of the possibilities to co-operate with domestic or international CASE tool vendors. Veikko Seppänen notes that “discussions on co-operation with [the firm I] were miscarried in 1990, new possibilities are unlikely”, and that “there are little contacts with other parties (with an exception of the importer of [an American CASE tool]”.

Discussions with the firm I that was involved in the Sokrates project had not resulted in any continuing co-operation, because the two parties apparently started to see each other as competitors. The representatives of the firm did not want to be interviewed as a part of this research - they considered that it is no use to dig up the old events. Yet, at the beginning of the Sokrates project the relations seemed to be rather warm. VTT researchers expected that the firm I would apply the results in close co-operation with them:

> *Archived correspondence of the Sokrates project. A hand-written note of the manager of the project on a telephone conversation between him and the managing director of the firm I, 6.7.1988: “The view [of firm I] to the development of justified new features for [its CASE tool], based on our research, is positive”.*

The main conclusion drawn from the situation by Veikko Seppänen was that “a good analysis should be done (in Finland, the Nordic countries, Europe and the USA) and contacts should be taken; the use of an external consultant could be beneficial or at least some kind of marketing support would be needed”. However, TKO did not have well-established means in the early nineties to market and sell distinct result items of its research.

The content of the Sokrates license agreement was, for example, designed from scratch by the code generation researchers themselves. The management’s viewpoint to commercial exploitation of research results was expressed in the long-term plan of TKO in 1991 as follows:

> *“Researchers are encouraged, based on VTT’s principles, to establish companies based on their innovations. Selling of licenses is considered on a case-by-case basis”.*

The managers, who wished to see red code generation projects paid by customers, ensured together with the researchers that the interests of VTT were appropriately taken into account in project contracts: “the rights of the [results of the] Sokrates project will remain at VTT, which can utilise them without any restrictions”. They did themselves some marketing of the Sokrates results, as part of general marketing activities, Frame A8.

Frame A8. Example of a marketing letter, Veikko Seppänen 1991.

Antti S…, B-o S-t Oy, T…katu 28, 33300 Tampere

14.8.1991

Dear Antti:

Referring to our telephone conversation, I am enclosing some material on VTT Computer Technology Laboratory for your information. ... We are interested in transferring in use, among others, the results of the newly finished research projects on design automation and software subcontracting. …

A2.3.3 Reagenix – birth of a new code generator

One of the technological trends in the early nineties was the use of personal computers rather than more expensive workstation computers in the design of embedded systems. Therefore, TKO aimed at developing a PC-based version of the Sokrates code generator. The plan was first attempted as a part of a Tekes project proposal on software reuse research, Frame A9. The PC-based generator was planned to be developed in only one man-month.

Frame A9. Tekes proposal including a PC-based version of Sokrates.

Project proposal, 7.6.1991, V. Seppänen: Reuse of embedded software
…
3. Simplified code generator from state machine and data flow models to C
- *Automation in the back end of the software development process is inevitable. Automation proceeds, if there are seamless links from one abstraction level to another (for example, from the state machine and data flow models of RTSA to the structures of the C language). This topic has a connection to the Sokrates project.*

…
I/3. Development and experimentation of a simplified code generator from state machine and data flow models to C (1 man-month).
- *Goal: A simplified PC-based version of the Sokrates code generator will be implemented that produces C code from physical RTSA models (data flow diagrams, state machines, ReaCTime macro language. The generator is experimented with a simple example.*
- *Tasks: The existing Sokrates code generator is fitted with a PC environment, by removing some features and by replacing the Ada-based mini-specification language with the ReaCTime language.*
- *Results: A PC-based code generator that has been validated by a simple example and can be used in the design of Sokrates-based projects and in the marketing of the Sokrates results.*

The proposal was not accepted, due to the lack of industrial interest. The likely reasons for the lack of interest were the increasing signs of a major recession and the tiredness of industry in joint software engineering research after the recent finishing of the Finsoft program:

TKO's strategic plan 1992...1996, H. Hakalahti, 25.4.1991. "The volume of generic research will decrease, at least temporarily [due to the recession]."

Two alternative plans existed: co-operation with some other commercial CASE tool vendor than the firm I and the development of the simpler code generator by the own funding of TKO. As described in the next section, the managers were not investing much in the latter, although they knew that the researchers were aiming at that direction: (Project planning memorandum, Veikko Seppänen, 1.8.1991) "Productisation of code generation and further research: [a code generation researcher] has carried out market analysis and is developing the first prototype of a PC version of Sokrates".

A2.3.3.1 Development of Reagenix

A PC-based code generator called Reagenix was after all developed by one of the code generation researchers, in a very short time in 1991 and without much of any financial support from TKO. The managers, colleagues and apparently also the customers thought that Reagenix was a PC-based version of Sokrates. This was, however, not the understanding of the researchers themselves. They told in the interview that the generator was developed anew, based on experiences not only from Sokrates but also from the related systems analysed in the late eighties:

Code generation researcher: "This was not the case. Reagenix is not any redesigned Sokrates. It is designed completely anew based on my experience from the pitfalls of Draco, Refine and Sokrates. It was developed in a pre-study, and made ready with surprisingly little effort. From my point of view it was a mistake that it was taken as the continuation of Sokrates and therefore no one was interested. On the other hand, no one knew which skills were needed to develop it".

The other alternative plan, co-operation with commercial CASE tool vendors, was also pursued. The idea was to replace the CASE tool of the firm I used as the user interface of Sokrates by another tool. A small pre-study on building a link between an American CASE tool and the Reagenix code generator was carried out. This study caused problems with the firm I, which burst out in a software engineering seminar, where one of the section heads of TKO was giving a presentation. He describes the situation and his conclusions from it as follows.

"Two things made me to be involved in this matter. I gave a presentation in the … seminar, because Hannu Hakalahti had asked me to that. [The managing director of the firm I] had been there, too. [The organiser of the seminar] was keen on taking modern software engineering methods and tools in use. A cousin of Hannu took care of the matter [on the behalf of the seminar organiser]. She had been told [by the firm I] that VTT has a tool that is competing against [their tool]. I am remembering that [the American tool vendor] was somehow involved, but not exactly how".

"[The code generation researchers] had at that point already made some marketing material on Reagenix, which had been delivered to industry, and [the firm I] had also seen them somewhere. [The firm I] considered VTT as its competitor. … I told them that they would have had the opportunity to utilise the work done in the Sokrates project, but they did not want to do it. My opinion is that if VTT begins to sell software products, it must have everything related to product maintenance, delivery etc. arranged. Therefore, I did not see Reagenix as a commercial product of VTT. The basic problem of the marketing of Sokrates/Reagenix … was that it did not start from the customer's problems, but aimed at pushing the technology on the take-or-leave principle. If one chooses the latter, he does not understand what is best for him. Therefore, I marketed myself Sokrates and Reagenix only at a general level. However, I always mentioned them in my marketing meetings at companies".

Although the managers did not like the idea of VTT making a product out of the code generator, they let the researchers to sell licenses, open a special account to collect the license fees, 25 000 FIM each - and use the income to develop the generator further. In addition to the use of this account, only one green project devoted solely to the development of the generator was carried out after the Sokrates project, Table A5.

Table A5. Green R&D of code generation in 1992 - 1997.

Project	**Parties**	**Results**	**Financing** (1000 FIM)	**Year**
Reagenix (account)	TKO	Reagenix code generator, methods	< 150?	1992 - 1997
Aniprosa, Aniprosa-2	TKO, firm El	Animator	150+45 (by VTT)	1993 - 1994

The total funding of TKO for the continuing work was very modest, when compared to the volume of the Sokrates project. Taking into account that some routine work was needed to sell the licenses and that an external subcontractor was hired for the Aniprosa project, the researchers had at most a few man-weeks available for the further development of the code generator that they could control by themselves in 1992 - 1997. The average volume of Sokrates had been as much as 180 man-weeks a year during 1988 - 1991. The change of the financial resources was dramatic indeed.

The self-funded research that was carried out in the Aniprosa projects aimed at the development of an additional tool related to the code generator, the so called animator. The need for it had arisen in a red industrial code generation project contracted by the firm Nm. Similar animation features were also being offered as part of commercial CASE tools. A prototype of the animator was implemented by using a small software house as a subcontractor. The owner of the firm had worked as a visiting researcher at TKO in the turn of the nineties and was familiar with Reagenix.

The firm could implement the animator at a very reasonable price and it had hands-on experience from the technologies on which the animator was based. The work continued until 1994, despite of some financing problems. The special Reagenix account was used, in addition to small funds obtained from two blue research projects: “The work [for the debugger] was not finished in the ROTU project due to project changes in the turn of the year, but the work was continued as a part of the Reagenix account”. The firm Nm did not, however, take the animator in use. The results were presented to a few other companies, including the firm N, but industrial interest could not be awakened. Instead, the animator was used in some joint research projects carried out by VTT. Thy were characterised by the researchers as follows:

> *“After all, the result is better than we could believe. The ReAnimator concept is according to our best knowledge both technically and functionally at a top level, also in international terms. Corresponding animators are usually available in workstations, but do not operate at the same speed”.*

The managing director of the co-operating firm commented the work as follows in an interview in 1998:

> *“The idea and the results were excellent. I have always liked Reagenix and Aniprosa was like a cream on the top. I believe that the biggest mistake was that SA/SD was already out of fashion at that time … [Moreover] software designers do not trust generated code, because it is difficult to read the code and locate errors from it”.*

A2.3.3.2 Marketing and planning related to Reagenix

The former manager of the Sokrates project deputised Veikko Seppänen as the head of the software engineering section of TKO from November 1991 to March 1993. The long-term plan of the section for 1992 - 1997 authored by him in April 1992 describes the state of the code generation and the plans for the exploitation of Reagenix as follows:

> *“Code generation features are available in commercial CASE tools, but not comprehensively (the same tools do not support both prototyping and implementation) and not for such special target environments as micro-controller software … A goal is also to acquire commercial code generators … and evaluate them.*

Six years later, he commented the situation as follows:

> *“I am still wondering, how it should have been marketed … There was only one direct exploiter, [the firm I], with which we had negotiations that failed, because they wanted to develop it solely by themselves”.*

Already earlier – actually only one month after Veikko Seppänen had started his leave, the new section head and the code generation researchers organised a press conference to make Reagenix public:

Press release, VTT/TKO, 18.12.1991, Reagenix generator makes real-time programming more efficient. "The use of the generator in all phases from specification to programming decreases the number of specification, design and implementation errors, and makes the final testing shorter. … The use of the generator increases the productivity of software development as much as ten times. The generator makes it possible to test a modification of a design document within a few seconds in an ordinary PC environment. The generator can solve automatically problems of concurrent system functions and program structures. Reagenix is a general-purpose tool … several extensions have been added to the original SA/SD method to improve the expression power and implementability of designs. We provide compiler guarantee … [and] user support. You are welcome to contact our help person, who can estimate for free the applicability of the ReaGenix generator for your needs. … Hotline service. Registered users are eligible for free telephone, telefax and e-mail help for one year after the delivery … Other support services are available according to the enclosed price-list."

The marketing plan of TKO for 1992 - 1993 written by the deputy director Jukka Karjalainen in May 1992 states that "the most considerable change when referred to the earlier plans is the difficult economic situation in industry, which also affects its possibilities for using external R&D services". Also the code generation researcher deputising Veikko Seppänen as the section head in 1991 - 1993 pointed out this in the interview:

"Maybe we were after all doing correct things, but the timing was wrong. The recession took place, unfortunately, at the same time. This kind of a topic [code generation] was simply not the most important issue in companies [at that time]".

The marketing plan of TKO for 1992 - 1993 is a good example of how the managers considered Reagenix as a new version of Sokrates: "the code generator developed in the Sokrates project will be used in at least three industrial development projects. An additional goal is that the immediate results of Sokrates would be used widely in at least one, but hopefully in more, industrial software development environment and tool projects. In 1993, integration of the Iptes and ReaGenix approaches are sought in a Tekes-funded project".

The integration with Iptes refers to the possibility of utilising Sokrates as a part of a large international research project on executable specifications. This plan was never realised, because of the fundamental differences in the approaches adopted by the projects. It illustrates, however, the managers' hope to associate the results of Sokrates with the research topics being investigated at TKO in joint European projects in the early nineties.

The annual plan of TKO for 1993 includes another goal related to this strategy: "in the area of code generation an attempt is made to join a Brite/Euram project" – this attempt never resulted in any project. No goals for the productisation of Reagenix or for its use in customer projects are detailed in this annual plan.

However, own funding is planned to be used for the "integration of the ReaGenix code generator with the SPS and SIC programs (2 man-months)" and for the "development of PC-animation techniques". The latter goal resulted to the Aniprosa project. An internal meeting on software tools was held a few months earlier. Jukka Karjalainen wrote in the meeting to his R&D diary that "no investments in marketing [of Reagenix is done], further development [of the tool will be done] in connection with some application projects (diagnostics, Tulko, Lokki), the same kind of an approach will continue". In other words, the vice director of TKO who co-ordinated the marketing activities was not prepared to spend any money for the general marketing of Reagenix. Instead, he looked for its use and further development in projects. The Tulko and Lokki projects mentioned were joint blue research projects. Reagenix was indeed successfully applied in Tulko, in addition to several fault diagnosis projects and in a small consultation work done for an industrial participant of the Lokki project.

The newly appointed section head's plan for the marketing of the services of the software engineering section in 1992 included a "product" called the "automation of the software process: pre-studies for and development of … code generation … in customers' environments", based on the strategy of comprehensive R&D services not provided by any other organisation. A marketing brochure was prepared to illustrate the methods and tools "used and developed" by TKO, including Reagenix, Table A6.

Table A6. Methods and tools used and developed (*) by TKO.

Development phase	**Methods**	**Tools**
Project planning and control	ISO 9001, 9000-3	VTT Project Management System (*)
Requirement analysis	QFD (Quality Function Deployment)	ReQueX (*), QFDesigner
Specification and design	RT SA/SD Statecharts OOA, OOD Lotos	Prosa, Teamwork, StP Statemate OOATool ARA(*)
Implementation	Automatic code generation Program compilation	ReaGeniX(*), ProsaC C, C++, PLM, Ada, ASM
Testing	Functional testing Static analysis	Unit Tester (CUTE)(*), Integration Tester (MOSIM)(*) Teamwork/Ensemble, PC-Lint
Maintenance	Version control Configuration management	PVCS, SCCS, RCS Polymake

A software technology reference sheet also prepared for marketing needs in 1992 states that TKO has “developed embedded software and their production environments for over ten years and transferred new technologies to industry”. Concerning software specification and design the reference sheet indicates that TKO “designs appropriate functionality and implementation architecture and develops the customer’s design automation” by using, for example, the real-time SA/SD method (Prosa, TeamWork, Reagenix and Iptes design tools).

To implement embedded software systems, “automatic code generation (the Reagenix tool)” is told to be available. The reference sheet was criticised by some of the researchers and line managers of TKO, because it mixed self-made tools – even such research-prototypes as the specification execution environment Iptes – with fully commercial tools. The reference sheets ends with a remark on “comprehensive services” being offered, including “expert help for design and implementation, training of methods, tools and counselling”.

A co-operation offer on embedded software engineering was sent to a potential foreign customer in 1992, but did not result in co-operation. It includes a proposal for the “evaluation of some of VTT/CTL tools (ReQueX, ReaGeniX, Mosim, operating system or protocol)”, using examples of and criteria set by the customer. This work was estimated to take two to three man-months and demonstrate that “new tools increase the reliability of code and efficiency of work”. Reagenix was suggested to be integrated into the customer’s embedded software production environment, in two to six man-months. The use of Reagenix was illustrated as shown in Figure A1, including the benefits of:

- reduced number of implementation errors (less re-design, shorter time-to-market),
- reduced volume of design work (improved productivity),
- increased reuse in design and implementation (improved productivity), and
- increased quality of documentation (improved communication).

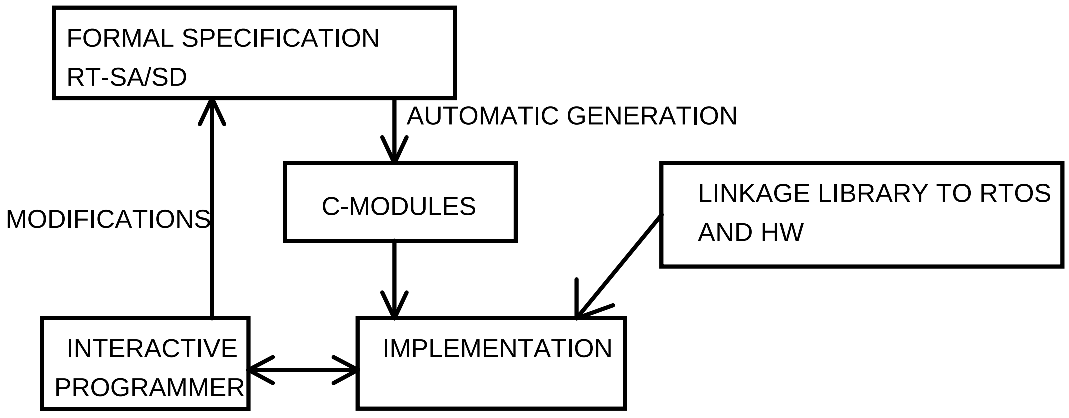

Figure A1. The use of Reagenix, as illustrated in a marketing brochure.

A SWOT analysis of the software engineering R&D services of TKO was done in March 1993. It involved both a researcher not dealt with the code generation research and the former manager of the Sokrates project. The researcher lists the strengths “SA/RT, process models”, the weaknesses “formal methods/synthesis, graphics”, the opportunity “process improvement” and the threat “object-oriented techniques replace SA/RT”.

The Sokrates project manager describes, on the other hand, code generation as the main strength of TKO, including Reagenix and the ReagOS operating system that involve “outstanding competence, relatively large user community, rather many references”. The weaknesses related to code generation include “poor marketing material, limited documentation”, the opportunities “independent from other tool vendors and method developers” and the threats “software development automation is a distant topic for some companies to even understand, the things are related to highly personal and strategic issues, lack of competition”.

The “life-cycle position” of code generation is estimated by the researcher to be 3/5, where 3 stands for “apply” and 4 for “sell”. The strategic plan for software engineering 1993, put together by Veikko Seppänen, who had returned back to his post as a section head, describes the strategic outlook related to code generation as follows:

> *“Embedded systems specification projects have been carried out in many application areas, involving developing, follower and well developed companies. In these projects, especially the SA/RT method has been used, with its commercial support tools (such as Prosa and in some cases Teamwork) and self-made support tool prototypes (such as Reagenix) … The threats [include] rapid ageing of the basic skills, for example due to the decreasing use of structured techniques”.*

The role of the SA/SD method is thus seen as decreasing and Reagenix as a “self-made support tool prototype”, as opposed to a “commercial support tool”. The customers were segmented to “developing, follower” and “well-developed” companies instead of industrial segments.

A competitor analysis was also done as a part of the strategic planning. It did not address competition concerning such individual R&D topics as code generation, but evaluated competition at a more general level. The competitors included universities, other VTT laboratories and corporate R&D departments:

> *“University groups. Strengths: deep technological knowledge, international contacts, long-term activities, better possibilities to cope with research evaluations; Weaknesses: professor-centred, often weak contact to industry, usually small groups, personnel changes frequently, few full-time researchers.*
>
> *Other VTT laboratories: no remarkable differences wrt. TKO.*
>
> *Industrial, independent R&D units. Strengths: application and needs expertise, knowledge of the customer’s key persons, opportunity to affect R&D decisions, public funding favours industrial R&D, access to early R&D planning activities; Weaknesses: corporate staff, difficult to sell services outside the own corporation, actual needs knowledge may be weak, target area often very wide, the overhead of a big corporation, often less focused on embedded systems; Goals: technology expertise wrt. business units, generation of new businesses in 3 - 5 years, technology mediator and buffer, collaboration with universities, international R&D projects”.*

The laboratory director Hannu Hakalahti does not mention in the 1993 strategic plan of TKO any strengths or weaknesses related to Reagenix and structured methods. The industrial development contracts, i.e. red projects, are told to involve 50% of electronics companies, of which 20% were telecommunication firms, 25% of machine automation companies, 15% of process automation companies and 10% of other firms. Machine automation that was expected to be the main customer segment for code generation, was one fourth of the customer base of TKO. Yet, the marketing plan for machine and equipment industries prepared in 1992 by Jukka Karjalainen does not include anything of code generation.

However, one of the “most remarkable R&D results” listed in the half-year report of TKO in August 1993 is that the “application of code generation developed in the Finsoft program continued in one company [the firm Nm]”. The rapid lose of organisational visibility of code generation as a strategic R&D topic and a marketable piece of competence is rather striking, but it may result simply from the fact that the plans and result reports were heavily related to projects. There was only one major project related directly to code generation going on in 1993, and that project was mentioned in the half-year report of the institute.

In 1994, the whole organisation of VTT was restructured. The former independent but rather small laboratories were integrated into much larger research units. TKO was merged with three other laboratories, including VTT Electronics Laboratory, to form ELE. The researchers of TKO were allocated to two different research departments.

Most of the former Sokrates researchers were allocated to the “Control systems” research group managed by Harri Perunka, a former researcher of TKO who had spent a few years in industry and returned back to VTT in 1993. Hannu Hakalahti left VTT, whereas Jukka Karjalainen became the marketing manager of ELE, with a focus on embedded systems. Veikko Seppänen started as the head of the embedded software department.

One of the very first tasks of the new organisation was to analyse the existing customer base and the key technologies offered to different industrial segments in a strategic planning meeting in April 1994. The segments included telecommunication, electronic instruments, “other” electronics firms, machine and equipment manufacturers and “other” customers. The average size of projects carried out for each segment was also estimated, using the scale of “very big” (over a million marks), “big” (over 500 000 marks), “intermediate” (over 100 000 marks) and “small” (less than 100 000 marks).

Three intermediate projects, two of which were red ones and carried out for the former Sokrates participants – the firms Nm and S, were analysed. In addition, one blue intermediate research project on fault diagnosis carried out together with VTT Automation (a research department based on the former VTT Machine automation laboratory) was included. The key technologies listed to be used in these projects were Reagenix and the Cute unit testing tool associated with Reagenix.

The marketing plan of ELE for 1994 - 1995 includes “the marketing of the Cute unit testing tool”, with an intention to “demonstrate its integration with a commercial tool”. Moreover, “marketing of Reagenix type solutions to firms, whose core competence does not include distributed real-time systems” was planned, with “a special target” of machine vision firms. The marketing vision for Cute was thus the same as for Sokrates and Reagenix earlier – integration with an appropriate commercial tool. However, Reagenix itself was planned to be marketed to firms that were not interested in solving real-time problems – along the lines of the view to the needs of machine automation firms. Machine vision was mentioned as a potential customer segment most likely because the developer of Reagenix was at that time involved in a joint research project dealing with the design and implementation of embedded software for machine vision products.

From 1994 to 1997 the line management of ELE marketed Reagenix much as it used to do after the Sokrates project, mainly for potential new customers and as a part of the general marketing of the embedded software engineering R&D services of ELE. As an example, Veikko Seppänen and Jukka Karjalainen visited the company Np in 1995, to market participation in a joint research project on embedded software process improvement services. Reagenix and Cute were also discussed and it was agreed that “demonstrations will be given in Oulu later”. The firm Np became later, according to the code generation researchers, as one of the most satisfied Reagenix customers.

In a strategic co-operation meeting with the firm N in 1995 the possibilities to integrate Reagenix and the company's "simple" code generator that was still being heavily used after five years from its development, were discussed. Veikko Seppänen wrote the following note to his R&D diary: "No needs for designing new state models, therefore possible continuing development of Genera may involve configuration issues. State-model level graphical debugging "not the most serious problem … Could test cases be generated using Reagenix/second generation Genera?"

The graphical debugging developed as one of the new features for Reagenix did not thus interest the firm N. The idea of generating automatically test cases from high-level specifications became one of the main features of new, commercial telecommunication-oriented CASE tools in the late nineties. Reagenix was used for this kind of a purpose only in a small piece of work done at ELE for the firm N in 1997. At the same time, the code generation researchers continued to market Reagenix licenses and training courses. Several licenses were given for free to educational institutes, in connection with Sokrates-SA courses. In addition to these white relationships, a few licenses and courses were also sold – mainly to new customers developing software for electronic instruments and small, portable devices. This is also the customer segment from which the small company that bought the rights to Reagenix and ReagOS in 1998 started.

The original manager of the Sokrates project is the managing director of the company, and the rest of the owners include several of his former project staff members. In October 1998, he told that he had been "busily working for a customer project until two a.m. in the previous night" – the innovator seemed to have returned back to his roots after twenty years, but with the code generator at his and his customers' disposal.

A2.3.4 External exploitation in red projects

Table A7 summarises the red projects in which the code generation competence was exploited by the industrial – or in one case an internal – customers of VTT in the nineties. Mots-2 was a pseudo-red project from the viewpoint of Reagenix, the use of the generator was only a minor task. Several pseudo-red projects were carried out by the Electronics Laboratory (later VTT Automation). The total sum of the income to VTT from the other projects is 1900 000 Finnish marks, i.e. slightly less than one half of the total expenses of Sokrates, but almost four times of the expenses of TKO.

Table A7. Exploitation of the code generation competence in red projects.

Project	**Customer**	**Results**	**Income** (1000 FIM)	**Year**
Kaapeli	Firm Nm, (Firm Ac)	Sokrates-based coding rules	200 (firm Nm)	1990 - 1991
MCS-REA	Firm Nm (Firm Ac)	Development of a code generator	100 (firm Nm)	1993
(Several)	(Several)	Generation of embedded machine control software	(pseudo-red: Electronics Laboratory)	1991 -
Raski	Firm R	Consultation of software design Methods and tools	250 (firm R)	1992
Osdyn	Firm N	Development of an Operating system	150 (firm N)	1992
Sympa	Firm S	Software design handbook	100 (firm S)	1992 - 1993
Cute	Firm S	Development of a software test tool	300 (firm S)	1993
Kaasu	VTT Food technology laboratory	Generation of embedded control software	80 (VTT/ELI)	1992
Nosto, Nosto-2	Firm Kc	Development of a software assembly system	120 + 600 (Firm Kc)	1993 - 1996
Mots-2	Firms N, Nt	Generation of test case software	(pseudo-red)	1997

A2.3.4.1 Broad co-operators – seeking for a paradigm shift

The Kaapeli and MCS-REA projects shown in the table were carried out for the firm Nm that became the key partner of TKO not only in the exploitation but also in the further development of code generation. The firm’s applications involved different kinds of factory automation systems, ranging from individual machines to complete production lines, part centres, and so on. The co-operation started as early as in 1990, when the Sokrates programming principles were encapsulated into a set of manual coding rules for software to be executed by a certain type of computer hardware.

The purpose of this project, Kaapeli, was to “develop methods and tools for the client’s environment, by which software can be produced for programmable logic controllers (PLCs) according to the Sokrates modelling method”. Moreover, “the goals of the project involve especially the creation of a common design culture at [the firm Nm]”. A small subcontracting company of Nm was also involved both in the Kaapeli and MCS-REA projects, mainly in the development and testing of the software for the pilot products. The contract of the Kaapeli project was written about six months after the work had already started: “the work has been started at 1.12.1990 and will be finished by 31.8.1991”.

The estimated size of the work was fifteen man-months, of which eight man-months would be carried out by TKO and about four months by the small subcontracting company. TKO charged only its prime costs, based on a certain overhead coefficient. Sokrates was described as a foundation of the planned work:

> *"The Sokrates project of the Finsoft technology program of TEKES has produced technology that can now be transferred in to industry. In this case the benefits of the technology project will be seen at [the firm Nm] already in products launched during the coming Spring and Summer [1991]. More powerful tools are needed in the design and documentation of PLCs, and the application of the Sokrates method and tools also to PLCs seems as a natural solution. The designers can then use the same tools and methods in design and testing. [The firm Nm] has taken part in the management group of the Sokrates project from the beginning. [It] has used the results of the Sokrates project for about a year in the design, implementation and testing of a new generation of device controls. The experiences are encouraging: the time used for testing and implementation has decreased considerably. The level of abstraction in design has been raised, as well as productivity. Based on these experiences, the results of the Sokrates project will be used in a wider scale. In the development of the methods and tools the Sokrates project of VTT Computer Technology Laboratory has paid special attention to industrial usability".*

The work was planned as the customer company's project, not as a subcontracting project of TKO for the firm Nm. The project manager was the R&D manager of the customer, the representatives of the project management group included, in addition to him and his colleague, two code generation researchers of TKO. The research trainee of TKO, who was one of the main resources of the project, acted as the secretary of the management group. The project plan states that "Veikko Seppänen of VTT/TKO will also take part in the management group, as the representative of VTT/TKO, as well as a representative of TEKES that will be named later."

This arrangement was against the project management principles of TKO, being described at that time in an ISO9001 quality management system. According to these principles, the section head of TKO and the R&D manager of the customer would have been the only representatives of the management group, and TKO would have named its own project manager for the work that it was carrying out as a part of the larger project of the customer. The project manager would have been the secretary of the project management group, not its member, because he did not sign the contract but the managers of the two organisations. In practice, Veikko Seppänen had difficulties in taking part in the management group meetings, because they were usually integrated into technical project meetings organised when needed. In the group rehearsal the code generation researchers told that this was, however, not done on purpose, but it was just their style of working.

The MCS-REA project carried out in 1993 - 1994 was in many ways the continuation of the Kaapeli project, but based on Reagenix. In the contract the researchers describe Reagenix as a result of the Sokrates project, which they strongly denied in the group rehearsal (see Appendix 3): "The starting points for the contracted R&D work are … the Reagenix code generator developed by the supplier in the earlier Sokrates project". The same was indicated in the project plan "… the Reagenix code generator developed by VTT/TKO that is based on the same Sokrates technology."

The contract states that "the customer has usage rights for the Reagenix code generator and the additional features of the generator developed in this project (the features implemented in phase 3 of task 1), to develop its own products within the limits of the [Reagenix] license. VTT has the owner's rights for these additional features of Reagenix, and can apply them without any restrictions … A separate license agreement and order will be prepared on the usage and content of ReaGenix." This text was included in the contract, because Reagenix was extended considerably as a part of the project. Yet, the project was rather small, only two man-months with an option of two additional months, which was not contracted by the customer. This time the organisation of the management group of the project followed the quality management principles of TKO.

The project group proceeded in a very rapid pace in its problem solving work. As an example, one project management meeting preceded the specification, implementation and delivery of new features of the generator to the customer: "It was noted that the changes required by the program generator were implemented in the morning, and PgmGen version 4 was delivered to the customer. This version was accepted [by the management group], its functional testing will be carried out later".

In the final project management group meeting the "technical results of the project" are described as "successful and correspond to the goals except documentation, the level of which is not quite what was expected." The representative of the small subcontracting company that was involved also in the MCS-REA project considered that "the generators which have been developed are usable in the real-life environment and product delivery projects." The customer characterised the project as follows:

> *"The specification of the project succeeded well and its implementation corresponds to the specification. VTT's attitude has been positive during the whole project, and problems have been solved fast. Co-operation [with VTT] has been excellent, when compared for example to other suppliers [of the firm Nm]".*

VTT Information Technology institute was used as a consultant in the project, and its work was also considered as useful:

> *"The results are working well and no 'ghost errors' have been found, so that the results are reliable. The work done by Antoni Wolski of VTT Information Technology for the data base definitions have also been found useful in the MCS project."*

The code generation researchers were satisfied with the results, too:

> *Co-operation has functioned well, as well as the different roles of the participants (application – software engineering – generation technology), which has made the use of resources efficient … Co-operation between experts has resulted in mutual benefits and has been particularly efficient … I was involved in the project only for a short time, but the given tasks were well specified and therefore easy to carry out."*

Harri Perunka commented the work as a section head as follows: "The project was successful and unbiased pioneering work has been done in the area of code generation. I was involved in the project only for a short time, and my earlier belief was that the project is one that never finishes." About a month earlier he had written into his personal R&D diary the following remark on the project:

> *"Project has lengthened and contracting of the continuing work has become difficult; small project but full (project) bureaucracy; problems to solve larger than expected; license fees do not cover the extra expenses needed (for additional work to solve the problems); continuing efforts: maintenance (invoicing based on hours of actual work done"*.

In the internal final project review in May 1994 the results of the project are summarised as follows: "methods: message coding/decoding based on C macros (CUTE technique), code generation techniques based on Prolog macros; programs: PgmGen, Reagenix v 1.2; and documents: ReaGeniX Quick References, User Manual. The quality of the results is characterised as follows: "only small errors have been found that could be corrected in less than one day". However, "the users documentation could not be finished, due to the size of the project (100 kFIM)". The industrial usability of the results is described as follows: "PgmGen and Reagenix have been over six months in a real use at the customer site. [The customer's] MCS software has been produced completely by using the PgmGen and ReaGeniX technologies. The code produced by the generators will be used in June, as a part of … [a] product delivery to Saudi Arabia (about 100 000 lines of C code). The customer is also porting its earlier … software to the generator (about 200 000 lines of C code)."

The formal feedback acquired from the customer by using the customer feedback form and evaluated in the final internal project review was very good. It included the following rates (out of 5 as the maximum): contacting 5, offer/contracting 4 - 5, project management 4, customer service 4, quality of reports 4, usefulness 4, customer's own support 3 - 4, quality of research 4, quality of software development 4, overall satisfaction 5. These rates can be contrasted with the results of the interviews of the two key persons of the customer company in 1998. Their story illustrates the experiences from the use of the results of the MCS-REA project by the firm Nm.

Product manager: “Thank you for the questions. I have waited for a possibility to give some feedback. My point of view is more that of a business and a customer than a technology. Sokrates interested us, because we had a need to raise the productivity of programming and application development. We had the MCS project ongoing, targeting to the control of a material flow in a … factory. VTT persons visited us [to market code generation], which was critical [for our decision to co-operate]. We had earlier contacts related to the … manipulator [the Kaapeli project]. [Yet, the use of the generator] was not discussed at length internally or with VTT. Little by little, code generation and Reagenix became a part of the picture, and we were in a situation that was certainly not planned initially. Namely, we used the … CASE tool [of the firm I], to which these [code generation techniques] were integrated.

Yet, Reagenix remained as a wild package, because it was not a commercial product of any vendor, which would have had maintenance support and had been developed along with the other software. Our aim was and still is to develop applications – not to develop software tools. This does not exclude participation in tool development, but responsibility [of the development] must be elsewhere. We had a positive view from the RHU [Kaapeli] project, and the technique interested us. We had much money and time, and the feeling was that we could do whatever we wished. The purpose was good, but it appeared later that the cake was too big and the means were wrong. We have used the code generator and the SA method, too. Productivity has been poor. It takes too much time and is difficult to learn to use them. The SA method has been too widely used instead of much better methods. In addition, the QNX operating system [interfaced to Reagenix in the MCS-REA project] has been a problem, because there are extremely few persons around who would be familiar with it. The result has been that awfully many hours were spent on playing with the tools. The functionality of the application and its user interface suffered. Our product [to be developed by using the code generator] did not meet the customer’s needs for functionality. There was no added value for the customer in the product, and the recession hit us before we could correct the situation. Our business [related to this product] was decided to be finished.

Technically, the contribution of VTT was good. From the viewpoint of the business its was fatal. Reagenix should have been developed together with a tool vendor ([the firm I]?), or not at all. In principle, code generation is good, one tool among many others. I am not any more a programmer, but I believe that there is still something to do until the programmer can control the behaviour of the code without knowing the principles of the code generation and the content of the generated code. I have also started to suspect the wide applicability and effectiveness of the SA/SD method. Everyone should think of his own role and where he is and can be really good. It should be thought over in every project what is needed as a whole, and the real experts and responsible persons should be collected around that whole. If one succeeds in this, the project may also succeed. We learned a lot in the MCS-REA project, and our own ignorance has been revealed. Yet, it was a good project in which many good things were produced, and I wish to thank all VTT participants”.

R&D manager: “Our goals were the increase of the manageability of software-intensive projects and products, as well as the understandability and maintainability of the software. We made the first contacts [with the code generation researchers] in the events of the continuing engineering education organisation Insko and the Finnish Quality Association. These involved presentations of the use of the SA method. Later, there were other discussions. [We wished to be] more systematic in software development, [and to gain] insights and experiences.

The manual coding rules developed [in the Kaapeli project] for the RHU project [of the firm Nm] were seen as positive. Based on these experiences, we believed that it was possible to transfer the coding rules to the generator. The generator was taken in use and all necessary modifications were done in the next big development project of the involved [product] area. The generator has been used in product deliveries based on the development projects. Some designers involved in the projects have used SA modelling also in other projects. [Yet], the learning curve for effective modelling and use of the generator is very high, [they are] the tools of gurus.

The technical contribution of VTT was alright. Our product business had obviously at the beginning enough critical mass to start [using the SA method and the code generator], but we did not have any real conditions to keep the product, the tool and the [design] culture alive and developing, while the turnover [of the product business] was decreasing. This would have required a much bigger investment in terms of efforts, people and money. The choice [of the method and the tool] was wrong, considering the longer-term situation of the company. It would have required a larger-scale software development environment [to be successful].

Code generation is alright, I mean coding done by using some kind of higher-level tools. Typically, mainstream tools have survived in our use. Both our personnel and customers shun exotic solutions (longer-term responsibility). Of course, we do not have any crystal balls for making the right choices. The quality of the choices becomes visible along time.

I wish to thank everyone, both at VTT and in the MMS group [of the customer]. RHU with the manual coding rules and MCS with the Reagenix generator were good projects”.

A manager of TKO described his experiences from the Kaapeli and MCS-REA projects as follows: “My viewpoint is that if researchers use other researchers’ tools, it is not a convincing case for industrial usefulness. Certainly the case [of the firm Nm] is the best in this regard. They used the results in my opinion at least the some extend. I was not involved in the case, though. [The code generation researchers] took care of it. They were perhaps a little bit jealous, too. It was [their] way for managing things. The guys just did something, and if there were some meetings, they never told anyone about them. I don’t know, if that was on purpose, or if it was just their way for doing things. We tried several times to get some visibility to their work. … I believe that we lost our faith in their work, and they must have been taken that negatively”.

VTT Electronics laboratory used first Sokrates and then Reagenix not only in joint research projects, but also in several red customer projects. The former researcher and later group manager of VTT Electronics laboratory involved in customer projects where Reagenix was applied told in an interview that “originally, the purpose [of the use of the generator] was to get the overall software process in control, which is a kind of permanent problem to us, who are end users of software technologies. It started at the end of the eighties, when you [at TKO] had the [code generation] projects. We had used the SA method, which was good for documentation etc. and the generator was a natural extension of that. We had noticed the gap between SA models and code, which could be closed by the generator”.

When asked, if he saw the generator as a tool for sort of getting rid of software development, as automation engineers who did not want to be software engineers, he answered: “No, but software development and maintenance is painful, and we just aimed at to easing that pain by tools. … I am remembering that we made the initiative [to use the generator]. [A software designer] used Sokrates first, in customer projects. Then she used Reagenix, in customer projects also. … The reason for using it was that it had been marketed as a tool that saves time, and that was not a lie - because [the designer] had been using so that she already knew everything”.

He considered the need of the customers to pay license fees from the use of the code generator as inappropriate: “In customer projects the generator was used to implement application software. The customer needed to pay the license fee, too. This was not good at all. It helped us, but the customer did not have any possibilities to use it later at all. We had that possibility only because we had an experienced software designer … we must have some common sense at VTT, too, about maturing technologies for use. We should start collecting money from the developed technologies only after a certain level of maturity has been achieved. Usually, we are too impatient in collecting the money. …The real status of the developed technology must be evaluated to see what needs to be done to proceed. Otherwise we are just swindling”.

The biggest problems in Reagenix were related to its usability: “You [at TKO] should have lowered the threshold of using Reagenix, that would have requested just a modest sum of money. But that money should not have given to [the code generation researchers], by saying that here you have three hundred thousands more to spend, please – you can say good-bye to that money at the same time. … It has been a management problem [at ELE], money has been spent to many nice features, instead of making it more user-friendly, producing good examples, and so on”. The software designer, who used Reagenix in 1991 - 1993, described her experiences in an interview as follows: “I used the generator in perhaps five projects. It suited well for the design of machine control systems, but the customers did not like it. The reason was that state machines [on which the SA/SD models were much based] were too difficult to understand, they were not the ordinary modelling means. … [SA/SD based] code generation is good in small projects and in research use, but not in the development of larger systems, because it is difficult to maintain the top-down system models”.

Another broad co-operator was the firm R, which was developing large volumes of different types of computer-controlled devices used by the general public. The firm was not involved in the Sokrates project, but it had been the customer of TKO in the early eighties. The Raski project that was established in 1992 addressed embedded systems software development methods and tools from a rather broad perspective. The planned volume of the work was six man-months, and the budget about 250 000 Finnish marks. The manager of the project was a former Sokrates project manager.

The project plan describes the goals of the work as follows: “the goal is to develop the methods, tools and guidelines of the software process [of the customer]. The developed methods aim at improving the documentability, generality, predictability and productivity of the software process, in addition to such quality aspects of the software as testability and reliability”. Evaluation of the applicability of the SA method and tools was planned to be done, based on which the software process would be described and recommendations for tools made. According to the final report of the project, the following tasks had been accomplished and the corresponding results produced: SA training, needs analysis, development of the software for one of the customer’s products, evaluation of design methods and tools, and modelling of the software process. Veikko Seppänen, to whom the situation of the Raski project was described after his return back to the post of a section head, wrote the following into his R&D diary in March 1993:

> *“Raski project meeting [the firm R]: done late 1992, Sokrates trial usage, SA training, did not proceed in all aspects, software development handbook skeleton will be delivered, [the customer] was not very satisfied with the results, [the deputised section head] contacted them in January 1993: development of a unit testing environment; will call again and ask the solutions, tools situation”*.

Another section head took also part in the preparation and management group of the project, and commented it in an interview as follows: “I was myself involved in the preparation and management of the project of [the firm R], in which Sokrates/Reagenix techniques were used. The project did not succeed very well, and the customer gave certain negative feedback”. No new project have been carried out for the firm R by ELE after the Raski project this far.

A2.3.4.2 Focused buyers – looking for specific items

There were a few customers that wished to utilise rather specific parts of the code generation competence. One of the earliest exploiters was the firm N, who applied the operating system principle developed in the Sokrates project in its products. However, the code generation researchers were not themselves involved in the Osdyn project, in which a new operating system version based on this principle was designed. The project was carried out by a researcher, who had been working in several earlier red projects contracted by the firm N, including a few related to operating systems. The planned volume of the Osdyn project was about five man-months.

The original goals set in the contract were: “implementation of OS dynamic timer feature for … in … target [processor], adding of task load monitor feature, design and implementation of dynamic reconfigurability. In addition the client has requested a set of maintenance tasks to be done for OS. These maintenance tasks are divided into two categories: bug fixing and nice to have features”.

The former Osdyn project manager commented the work in an interview as follows: “I am remembering that the idea of the scheduling [used in the development of the operating system] was told by [the manager of the Sokrates project] to [the representative of the firm N in the steering group of Sokrates] in some seminar. He wrote the first version of the operating system based on that idea. Finishing of the work was left to VTT apparently, because he had no time to do it. [The firm N] needed an operating system for [a certain product] that would not require much Ram and ROM memory”.

According to the manager of the Osdyn project the customer’s representative, the same person who took part in the steering group of Sokrates, took care of the subcontracting arrangement. He did know, why the Sokrates researchers were not used in the Osdyn project. Yet, he discussed several times during the project with the code generation researchers, “who certainly knew what was done, and helped as much as was needed”. Over ten million individual pieces of the product, in which the operating system is incorporated, had been sold in 1998. The operating system is likely to be “among the fifty most widely used in the world”.

The customer’s representative confirmed the basic need for developing the operating system, saving of the memory. Yet, he also told that one of his colleagues at the firm N had already developed a “somewhat similar” operating system, although “little simpler”, and therefore “the idea did not came solely from VTT”. He saw the development of the operating system as very specific to the client’s needs, and therefore it was carried out as a confidential red project. According to him the code generation researchers “consulted a little bit, and changed some ideas”.

The Sympa project carried out for the firm S resembled the Raski project in the sense that one of the main goals was to produce a software design handbook. However, the focus of the work was on testing, rather than on the whole software development process. A special test tool, to be later called Cute, was proposed: “After consulting sales material on several vendors, it seems that there are very few applicable test aids for parts of real-time systems. So, we propose development of a test environment consisting mainly of tailored software”. The work was related to Reagenix in the sense that it included an optional task: “Test bed interface generator for data flow diagram is to be built on the basis of similar part of ReaGeniX test bed generator”. The first representative of the customer in the management group was the same as in the steering group of the Sokrates project, but he left the project soon after its beginning because of the change of his duties at the firm S. The manager of the project was the original Sokrates project manager.

The management group commented “the short development times [of the results] estimated in the [status reports of the project]. The VTT researchers explained that “VTT/TKO has much experience from this kinds of tasks and readily available partial solutions than can be used”. The Cute project was established to continue to work on unit testing started in Sympa, the purpose was to “design and implement the unit test environment” and to write a “testing methodology handbook”.

A research trainee was hired to carry out the work and document the public parts of it as a diploma thesis ([Ihalainen 1993] in Appendix 1). The project manager was one of the former Sokrates researchers, the software quality manager of TKO was also involved to help in writing the handbook. In the internal final review in August 1993 the project and its results were evaluated as follows: “Results: CUTE software tester, Users Manual 3.0, Implementor Guide 3.0, Technical Reference 3.0 (Jari Ihalainen’s diploma thesis: Unit testing of real-time software, University of Oulu, 1993); Quality of the results: the CUTE software itself has not been tested, the language of the manuals has not been proof-read, the test material for the CUTE implementation was only 5+2 modules; Further application of the results: customer letter will be produced (VSE), discussions have been conducted with [the firm Nm], unit testing guidelines for TKO’s quality handbook?, presented to 10 persons at TKO, [the customer] does not have money for subcontracting? (JUK will keep contact)”.

The customer’s feedback in the scale from 1 to 5 was the following: contacting 5, offer/contracting 4, project management 5, customer service 4, quality of reports 4, usefulness 4, customer’s own support 3, quality of software development 4 (integration testing was not done?), overall satisfaction 5.

In addition to the diploma thesis, a scientific paper based on Cute was published in an international conference in 1995, where it was chosen as the best paper. Moreover, Veikko Seppänen wrote a summary of the diploma thesis that was used in customer letters sent to Sweden as a part of the marketing campaign of TKO. Although the Cute tool was introduced not only to the firm N but also to some other companies, it was not bought by any customer – its use would have required company-specific tailoring, but this was also considered as a reason for its effectiveness when compared to general-purpose testing tools. Marketing of continuing work based on Cute to the firm S did not succeed either, apparently in part because the firm was restructured as a consequence of the merging with its competitor, and the product development department was reorganised.

The former product development manager of the firm S, who took part in the management group of the project, commented the use of Cute in an interview in 1998 as follows: “the use has been small, but our experiences are still positive. The use requires work and time (heavy), which is the biggest obstacle for utilisation”. The reorganised firm S has taken part in several joint blue research projects of ELE after the Sympa project.

The Kaasu project is an example of internal “red” business between the different units of VTT. A control system implementation for a two-phase gas chromatograph was contracted from TKO by VTT Food technology laboratory. The estimate of the work to be carried out done by a code generation researcher in 1992 included the following tasks: “logical nucleus 5 days, device controllers 3 days, user interface 3 days, integration and testing 4 days, relays and their cases 2 days, Reagenix training 3 days, training of the implemented software 1 day, installation of the software 1 day: altogether 1 man-month”. A Reagenix license worth of 20 000 Finnish marks was also offered, including “the VTT discount of 20% – 5000 marks”. It was not bought the customer laboratory, though, most likely because they did not consider themselves as regular software developers. The software development work was carried out and succeeded well.

The Nosto projects addressed automatic configuration of PLC programs for the firm Kc, a manufacturer of large automated mechanical devices formerly owned by the same corporation as the firm K. The roots of the idea of automatic software assembly at the firm Kc are therefore most likely in the similar system developed by TKO for the firm K in the eighties: the pre-study that was first done was indeed carried out a knowledge engineering researcher of TKO who was a project manager in the earlier project of the firm K. However, since the target of the assembly was PLC software and there did not seem to be many knowledge-based features in the assembly problem, the code generation researcher who developed coding rules for the firm Nm in Sokrates, was allocated as the manager of the continuing project. The assembly system was built in that project, including a code generator based at least indirectly on Sokrates and Reagenix.

The customer described the nature of the project as “experimental” and was very pleased that despite of this the schedule and budget were kept well. “The commitment of VTT personnel to the project” and the co-operation between the firm Kc and VTT were considered as “excellent” by the customer. The feedback of VTT was much along the same lines, and it was noted that the subject of the project was related to “the core competence of VTT Electronics, component-based control software, reuse and automatic software assembly.” Nosto and Nosto-2 were the first projects carried out by TKO and ELE for the firm Kc. No further projects have been carried out for Kc later in the nineties, despite that several topics for continuing work were proposed at the end of the Nosto-2 project.

Reagenix was used in 1997 in the Mots-2 project, to generate parts of the code of the Mosim test environment developed and applied for the needs of the firms N and Nc (and earlier also for the firm K) since the late eighties. The approach taken by VTT to Mosim, building of a customer-specific tool environment and conducting several big project for its use and further development, was heavily criticised by the code generation researchers in the group rehearsal. They claimed that VTT is “stealing money” from industry by tying it into this kind of an expensive and long-term relationship. The code generator produced “efficient code”, although “it was impossible to produce an implementation-free design model as the method would have required”. Support for designing and implementing message buffering was also “missing from the generator”.

A2.3.5 Selling of Reagenix licenses

Although Reagenix was used in several red – and also blue – projects, the goal of selling Reagenix licenses independently from the projects did not succeed too well. The total number of the sold licenses, worth 25000 Finnish marks each, was closer to five than ten. In many cases the work needed to deliver, install and take the generator in use consumed a considerable part of the license income. Several licenses were given out for free, to universities and engineering schools – Reagenix was used perhaps by hundreds of engineering students, but this resulted in no income to ELE.

In Summer 1998, the rights of Reagenix were sold to a small company run by the original Sokrates project manager and owned also by some of the other code generation researchers. A modest royalty fee for ELE was agreed on any Reagenix licenses sold by the company during the coming few years, but no such income has realised yet. The firm is, however, busily using Reagenix to generate embedded software for a few customers that bought a Reagenix license from ELE earlier in the nineties.

One of the basic differences with regard to the present Reagenix licenses, when compared to the situation in the early nineties, is that there is no more any need to use the CASE tool of the firm I as the user interface of the generator. A replacement was designed in a blue research project on software reuse in the late nineties, by the former code generation researcher involved in the project. The notation supported by the user interface is no more strictly based on SA/SD, but on a more general software component modelling approach. An example of the notation on the control of a traffic light is shown in Figure A2. The generator produces C programs from software component models, Figure A3.

The user interface is based on a general-purpose commercial graphical software package, the cost of which is only a few thousand Finnish marks. The package can be tailored to support almost any graphical modelling language. This solution seems to have put an end to the deadlock between the code generation researchers and the firm I. Moreover, the two firms can from now on compete or co-operate as equal organisations that make business on the same grounds.

In 1998, the firm carried out generator-based software development for at least one industrial customer, and sold one Reagenix license to an educational institute.

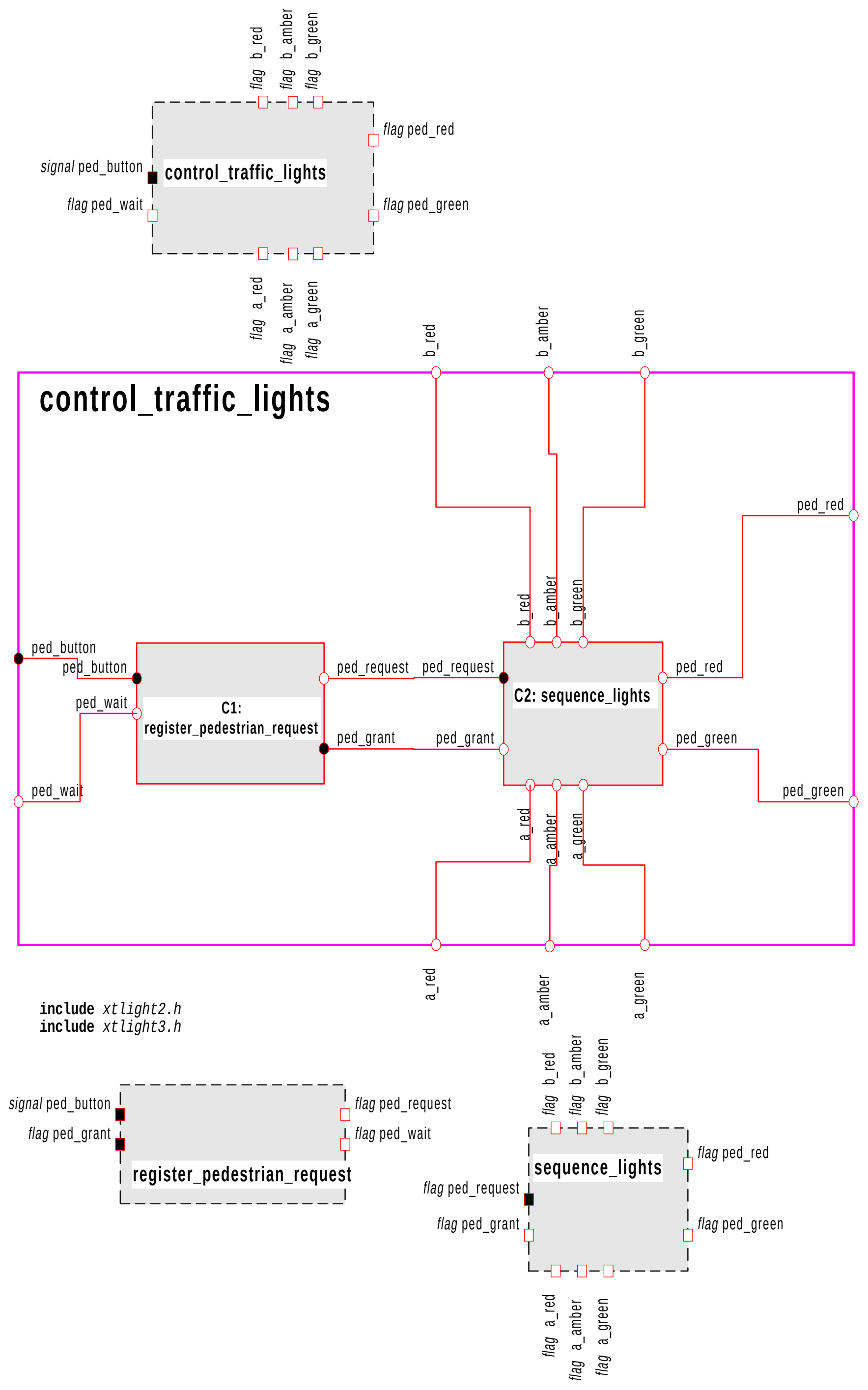

Figure A2. Reagenix-based model of traffic light control.

```
/* xtlight1.c - control_traffic_lights () - 1997-06-17  13:07:10 */
/*----------------------------------------------
      Diagram Information
      Title   : control_traffic_lights
------------------------------------------------
   ReaGeniX Programmer  Version  Version 2.Beta-01
   Licensed to: …
   ReaGeniX is a trademark of
   VTT Electronics, Finland
----------------------------------------------*/

/*  reactime compatibility check  */
#define reactime_style 1
#include "reactime.h"
#if reactime_level < 1
  #error Old reactime version included
#endif

/*  subprocess compatibility check  */
#ifndef no_flowcheck
  #include "xtlight1.v"
#endif

/*  data definitions  */
#include "xtlight1.h"

process_body(control_traffic_lights)
  #ifdef flag_init
    initial_value_reservation(initial_value(flag),flag) =
        {flag_init};
  #else
    #ifndef flag_uninitialized
      #error Uninitialized state variable of type flag
    #endif
  #endif

  initialization
    init_process(sequence_lights,C2);
    init_process(register_pedestrian_request,C1);

  init_phase_2
    link_own_flow_from(C2.ped_grant, R__30ped_grant);
    link_own_flow_to(R__30ped_grant, C1.ped_grant);
    link_out_flow(C2.b_red, b_red);
    link_out_flow(C2.b_amber, b_amber);
    link_out_flow(C2.b_green, b_green);
    link_out_flow(C1.ped_wait, ped_wait);
    link_own_flow_from(C1.ped_request, R__56ped_request);
    link_own_flow_to(R__56ped_request, C2.ped_request);
    link_out_flow(C2.ped_green, ped_green);
    link_out_flow(C2.ped_red, ped_red);
    link_out_flow(C2.a_green, a_green);
    link_out_flow(C2.a_amber, a_amber);
    link_out_flow(C2.a_red, a_red);
    init_2_process(sequence_lights,C2);
    init_2_process(register_pedestrian_request,C1);
  end_initialization

…

/* xtlight1.c - control_traffic_lights () - 1997-06-17  13:07:10 */
```

Figure A3. Piece of program code generated by Reagenix.

A2.3.6 Exploitation in joint research projects

As shown in Table A8, several different blue research projects were used as internal code generation "customers" – similar to the first exploitation activities performed already during the Sokrates project. None of the projects focused on code generation per se, which meant that rather small design assignments were given to the code generation researchers. In most cases these were supporting tasks, because the generation techniques were applied by the other researchers.

The projects did not pay any license fees or other kinds of compensations for the use of the code generation methods and tools, only the work expenses on a non-profit basis. For this reason, Table A8 does not show any distinct financing for the use of the code generation techniques. The volume of the tasks in which the techniques were applied varied from a few days to several man-months, the corresponding external financing thereby from a few thousand to hundreds of thousand of Finnish marks. For example, the external income from the work done by TKO for the last phase of Tulko that was heavily based on Reagenix, was about a million marks.

The lack of any internal usage fee was criticised by the code generation researchers. The management of TKO did not want to charge anything from the use of its own tool in projects that were carried out by itself or in close co-operation with its key research partners, such as VTT Electronics laboratory and the University of Oulu.

Table A8. Exploitation of the code generation competence in blue projects.

Project	**Customer**	**Results**	**Year**
Tulko	(VTT Electronics laboratory)	Generation of machine control software	1989 - 1993
Turva	(Työsuojelu-rahasto)	Design and analysis of safe embedded software	1991 - 1992
Lokki	(Machine automation firms)	Incl. evaluation of Reagenix	1991 - 1994
Diagnosis	(Firm T, Työsuojelu-rahasto)	Generation of diagnostic software	1992 - 1993
Rulla, Rulla-2	(Firm H)	Generation of diagnostic and machine control software	1993 - 1996
Rekki	(Machine vision firms, University of Oulu)	Design environment for real-time machine vision software	1995 - 1996
AVV	(VTT Automation, STUK)	Software safety analysis	1995 - 1997

Tulko was a large series of joint Tekes-funded projects on “intelligent machines of the future”, of the scale of Sokrates – well over five million Finnish marks. It was carried out by VTT Electronics laboratory, TKO, VTT Machine automation laboratory, University of Oulu and almost ten companies. It was mainly funded by Tekes.

Reagenix was used in the last phase of the project, to design and implement a demonstration system of an intelligent machine for the firm St, which was one of the customers of VTT Electronics laboratory. In the final management meeting of the project the feedback of the industrial participants was much similar to Sokrates – new “experiences” had been gained, and it might be possible to utilise the results “in the future”.

One of the key persons in the Tulko project at VTT Electronics laboratory summarised his experiences from the use of Reagenix in an interview as follows:

> *“In Tulko, [one of the code generation researchers] demonstrated Reagenix and built a working whole by it, … [but] a completely wrong direction was taken. The great software experts did it the wrong way. It went just the wrong way. Usually, the specifications are made in this kind of projects, but the implementation must go bottom-up, to ensure that it succeeds. We could implement the system anyway. The case system was the manipulator [of the firm Sc]. We actually had a continuing project with them afterwards, which I joined because [the software designer left us and] no one knew Reagenix. I produced the basic software by Reagenix, which was rather efficient after I learnt the cryptic Reagenix representational formalism. [The former Sokrates project manager] and the others helped me willingly, they were not watching the work over or asking for any money of that kind of use of Reagenix”.*

According to the interviewee Reagenix did not help to understand concurrence, because that should have been addressed as part of the analysis of the application, not as a software engineering problem:

> *“No, it is the other way round. We though that we already have a model that includes concurrent phenomena, and we just prototyped that model by using Reagenix. It was a fast way for implementing concurrence, which was described at the SA level, basically just by pushing a button. … If someone says so [that Reagenix provided a language to understand concurrence], I just say bullshit. We can see it in the field even nowadays. Tool providers tell about languages, but they do not understand that what matters is the analysis of the application”.*

Reagenix was used by TKO at the beginning of the nineties in the Turva project dealing with the design and analysis of safe embedded software. The person in charge of the use of Reagenix was the code generation researcher, who actually implemented Reagenix. The project was funded by a private foundation focusing on the improvement of safety and health in work.

The results of the project were described in the final report as follows: “Methods and tools were developed to be applied in the analysis of software safety, when the software development is based on Real Time Structured Analysis method. … A tool was developed to design fault trees, and a generator built to produce code from the fault trees. The generator produces code from the fault trees by which the software is augmented".

The elevator in case was not an industrial product, but the demonstration system of Sokrates, a toy built from Lego blocks. No industrial partner was involved in the project. The results of the project were published in several papers and reports, and as a part of a continuing engineering education course. However, no customer projects on the safety of embedded software could be established in the nineties, due to the lack of industrial interest.

Reagenix was used at least in two fault diagnosis related research projects in the early nineties. The Diag and Rulla projects were financed by the same foundation as the Turva project, but industrial companies were also involved. The code generation researchers were involved in both projects, in part without the knowledge of the line managers, who had actually allocated them to work in other projects. The decision to use the code generator in the Diag project was motivated as follows:

> *“The control software contains information on the operation of the machine. If it is implemented in a traditional way, it is difficult to get hands on the needed information. Therefore, we decided to use the SOCRATES technology [Okkonen&89, Okkonen&90] to implement the control system. This method is a refinement of RT-SA/SD introduced by Ward-Mellor [Ward&86]. It includes a code generator, which compiles the graphical diagrams to C. The code generator takes care of traditional real-time programming”.*

The Rulla project was a fault diagnosis research project in which the design of a safe wood carving machine for a small company was used as case example. The machine was a one-of-a-kind solution, but taken in production use. In the final report of the project the role of code generation was described as follows:

> *“The control system was implemented by using the Sokrates-SA method and the code generator, which compiles the program code directly from SA models. By using this method, the development of the control system remained clear and manageable. The use of the code generator prevents differences between SA models and the program code. In this way, program documentation remains up to date in an understandable form, which helps re-development and maintenance. In addition, the use of the code generator decreases the coding phase remarkably.”*

The Rulla-2 project carried the development of the machine further:

> *"The Rulla-2 project … based on the earlier work, focused on the improvement of control system design methods so that problems caused by design could be reduced in the future. As a result of the project, improved management of disturbances in the wood carving machine was expected. The project aimed at reaching its goal by the means of modelling. Visualisation and integration of the simulation control system models were seen as the most important target of modelling.*
>
> *The control system implemented in Reagenix carries out the necessary controlling actions based on this [model] information. During the specification phase of the [modelling] subtask it was found out that when the control software is described by the Reagenix method in a formal enough manner, it can be used as a distinct model. The result is remarkable in knowledge engineering in more general terms: if the same computer software can be used both in control and in analyses, modelling work can be considerably decreased".*

The results were thus encouraging, but Reagenix did not yet become routinely used in fault diagnosis projects. The reason was apparently because the researchers involved in the projects could use also fully commercial visual programming environments, such as Visual Basic and Visual C++ that were supported by manuals, textbooks and courses and widely used by software engineering practitioners and researchers.

The Lokki project was one of the activities of TKO directed towards machine and equipment manufacturers. The original goal of the project was, very broadly, help machine automation experts in solving embedded software engineering problems. Veikko Seppänen asked one of the code generation researchers to prepare the first draft of the project proposal and supported this choice in the management group of TKO, where the other managers suspected that the proposal would turn out just as reborn Sokrates.

The proposal was in the opinion of Veikko Seppänen poor, not seriously prepared. The industrial needs had not been clarified at all. The code generation researcher claimed that the proposal was based on the understanding of the real technological requirements of machine automation applications, as opposed to some impractical research questions. A fight followed, after which Veikko Seppänen prepared the proposal together with Jukka Karjalainen. The industrial participants wanted the project to be planned based on several case studies and focus on software reuse. The two-year project was followed by another two-year project addressing application-specific multi-method software design approaches.

Reagenix was, however, test used as part of the Lokki project and compared against the code generator of the firm I. The results of the evaluation were commented as follows: “The executable code of the manually coded program would be at least in this case (a part of a drilling jumbo’s boom control software) about five and a half times smaller [than the code generated by Reagenix]. In addition, the manually coded software would be much easier to understand than the software produced by the generator, because the latter includes many macros defined by the generator. On the other hand, it should be remembered that the code produced by the generator is not meant to be manually maintained, but all changes should be made to the model from which the code was produced. … Finding of errors is more difficult from the generated code, because at least for the moment no debugger is available for the generator that would allow to execute the SA description”.

The code generator of the firm I was evaluated for the needs of machine automation applications as follows: support to SA execution rules: poor (1), need to understand C programming: inevitable (1), easiness of functional modelling: satisfactory (3), validation of the developed models: fair (2), usefulness of the error messages: poor (1), easiness of debugging: poor (1), overall usability: satisfactory (3), modifiability for the users’ needs: good (4), support to reusable software components: satisfactory (3), and easiness of interconnecting software components: fair (2). The generator was described as not being “strictly speaking any generator for SA models … [because its] use requires to think of the implemented code all the time”. On the other hand, the generator “can be easily integrated into the present industrial software development environments”.

In the continuing project two different code generators incorporated in computer-aided control system design environments were evaluated. The main experiences were the following: “A rapid prototyping tool (MATRIXx) was evaluated in the [Kiuru] project to produce embedded software. A model was designed in the project of a garden tractor, whose control was implemented as a distributed system. The software of one node of the control system was modelled by MATRIXx, after which the real-time C code was generated. ... programs were manually added to the code. The real problems appeared when the [generated] code was ported to the embedded target environment. Technical problems prevented the testing of the code in the real target environment. The size of the generated code was much bigger than the corresponding manually coded software. ... The use of MATRIXx appeared to be easy, but for this kind of [rapid prototyping and code generation] use it would require some additional features.”

The Rekki project was carried out as a part of the Machine vision research program of Tekes. The topic of the project, integration of algorithm and real-time software development methods and tools for machine vision applications, was proposed by Veikko Seppänen to the University of Oulu, which was one of the key players of the research program.

Veikko Seppänen wrote the project proposal for Tekes together with his former student mate, a researcher at the university. One of the code generation researchers was asked to carry out the work allocated to VTT, another researcher was also involved mostly for providing SA/SD training to the industrial partners of the project. Most of the participating firms were small ones and dealing with machine vision based electronic measurement instruments. The applicability of Reagenix is described in the technical report of the project as follows: “Transfer of the [machine vision] algorithm into the target system can be made considerably easier by the ReaGenix too. The use of the ReaGenix and ReaGOS methods in as early a phase as possible provides for a fast (automatic) code transformation from the development environment to the target equipment”.

In the appendix of the report several CASE tools are listed and briefly evaluated. Reagenix and Reanimator are listed without any remarks that they are the only non-commercial tools in the table. ELE is given as the “manufacturer” of the two tools, all the other manufacturers are commercial firms. The CASE tool of the firm I and Reagenix are further evaluated, without making any remarks on this matter there either. When analysing testing tools, CUTE is also indicated to be manufactured by ELE. A listing of real-time operating systems is also given, with an indication that ELE is the “provider” of ReagOS. All the other real-time operating system providers in the list are commercial firms. On the other hand, the evaluation of the machine vision algorithm tools shown in another appendix written a university researcher points out that Khoros, the algorithm design tool used in the project by the university, is a public domain software package.

The volume of the work done by ELE in the Rekki project was six man-months. The results were considered as good. Some of the participating firms, as well as foreign machine vision experts, indicated that digital signal processing techniques “were missing from the [SA and Reagenix based] approach” to the development of real-time machine vision software. The main research party, University of Oulu, that subcontracted parts of the work from ELE, criticised that the work was only one of several ongoing projects of the VTT researcher. This resource distribution problem was pointed out also by the researcher in the group rehearsal.

In an interview in 1998 the manager of the project at the university told that “Reagenix does not give any particular support to solve machine vision problems” and that “during Rekki, Reagenix was by no means a mature. It had serious problems in documentation but also in part in the code, for example in the interoperation of different parts”. The integration of the machine vision algorithm design tool and Reagenix did not succeed very well either, according to the project manager. The university has not used the approach in any further machine vision projects. The firm Ha which provided a case example for the approach was, however, “quite satisfied, but the ultimate application of the results was unfinished”. In the final project management group meeting VTT pointed out as its opinion that the industrial partners “would have benefited from the development of a real [software design] environment” for their needs, but “they did not have readiness for this ... [because] they do not yet have the necessary volume to invest in the development of design methods and environments.”

Reagenix was used in the AVV project, where ELE was a subcontractor of VTT Automation and the customer was the Finnish Center for Radiation and Nuclear Safety (STUK). The researcher of TKO responsible for the AVV project was the former manager of the Turva project. Therefore, the idea of using Reagenix in software safety related R&D was carried further. According to the researcher, Reagenix models were used two industrial safety analysis cases, involving the foreign automation firms Ab and Si. The firms “saw the use of the code generator as useful, perhaps because some previously unknown faults had been identified by using the models developed with the help of Reagenix. The firms were interested in Reagenix, but never actually conducted any further discussions”. According to the researcher at least the firm Si, a very large multinational corporation, had already comprehensive methods and tools at its disposal.

Two other researchers of TKO were involved in the project. One of them “did not know, where the idea to use the generator came, I was not involved in the decision making”. The other explained that the project manager had made the decision. The researchers did not have any earlier experiences from Reagenix. One of the researchers considered Reagenix as “very good in executing and testing … logical [system] models, easy to use” and Reanimator as “good in unit testing”, whereas their documentation was not good. The other researcher was “not enthusiastic over” the tools. The code generation researchers “consulted the project for a few hours”.

The manager of the AVV project, a researcher of VTT Automation, confirmed that the decision to use Reagenix was made solely by TKO. He thought that the models developed by Reagenix “could apparently be produced rather easily and with relatively small efforts”, although the systems to be modelled were also “relatively simple”. According to the project manager, the firms Ab and Si that were involved in the project, “did not commented the use of Reagenix”, and “STUK did not provide any comments either”. The technical reports published by the project include several comments on Reagenix, which was compared to the well-known Statecharts modelling approach supported by many different commercial CASE tools:

> *“Two most potential of the considered methods, ReaGeniX and Statecharts, were compared to each other with respect to the defined criteria. ... while ReaGeniX has a stronger methodological background and is thus easier to apply, Statecharts has the better tool support. ... ReaGenix probably results to a better model because it has a clearly defined development process. The size of the executable software shows a noticeable difference: the Statecharts model has a size of 240 kB, while the ReaGeniX model uses only 25 kB. … ReaGeniX is clearly better regarding the costs of the environment and the tools themselves. … To conclude ... the ReaGeniX method is a better choice. … During the two years of the project no such novelties in the area of formal methods have been noticed that for instance would clearly exceed the reliability of the ReaGeniX models or cut down the amount of work of the modelling. Therefore it seems that the method is a useful choice also in the future.”*

REFERENCES OF APPENDIX 2

Aho, A.V., Bjorn-Andersen, N., Haberman, N., Neuhold, E.J., Rissanen, J., Simon, J.-C., Swanson, E.B. 1990. Research and teaching in computer science, computer engineering, and information systems. A critical evaluation. Publications of the Academy of Finland 3/90. VAPK-Publishing, Helsinki. 101 p.

Guy, K., Quintas, P., Hobday, M. 1991. Evaluation of the scientific and technological status of Finsoft: The Finnish software technology programme. Tekes, Helsinki. 58 p.

Kalaoja, J. 1988. Reaaliaikaohjelmiston kuvaus ja toteutus eräissä muunnosjärjestelmissä. Diploma thesis, University of Oulu. 68 p. (in Finnish)

Saukkonen, S. 1991. Finsoft-ohjelman tulosten teollisen hyödyn ja hyödynnettävyyden arviointi. Tekes, Helsinki. 75 p. (in Finnish)

Savilampi, J. 1989. Toimilaitejärjestelmän ohjauksen mallinnus Sokrates-SA-menetelmällä. Diploma thesis, University of Oulu, Department of Electrical Engineering. 31 p. (in Finnish)

Seppänen, V., Alajoutsijärvi, K., Kurki, M. 1998a. Competence-based evolution of contractual R&D relationships. Technical Research Centre of Finland, Espoo. VTT Publications 346. 70 p.

APPENDIX 3: CODE GENERATION STORIES

In the following, the original case summary written by Veikko Seppänen is accompanied with the *story of the code generation researchers*, created on the basis of interviews and a group rehearsal.

Code generation research and development started at TKO in the mid-eighties. There was a strong, global belief among software engineering researchers that the so-called automatic programming techniques would become practical within a few years. At TKO it was thought that this would be the case also for the development of embedded software. The problem of producing computer software automatically from higher-level system models is known from the very early days of computing. The so-called high-level programming languages had been developed in the sixties and seventies to help programmers to develop software by using higher-level concepts than those directly related to computer hardware. Many kinds of compilers, software tools for generating executable machine code from programming languages, had been available since the sixties.

The input language of compilers, understood and produced by programmers, is called source code. The output language of compilers is executed by computer hardware and called object code. The tools needed in writing source code are, typically, called editors. Different kinds of source code manipulation tools can be packaged into programming environments. Tools needed for managing object code are debuggers, whereas testing tools are used to facilitate executing the code and analysing the execution results. A typical programming task would thus involve the cycle of creating a source code, compiling it into an object code, and debugging the execution results. Many compilers and debuggers are dependent on hardware, the physical computer system that executes the object code. Therefore, software tool vendors sell families of tools tailored to certain computer hardware.

In the seventies and early eighties researchers were trying to "extend" compilation techniques to even higher abstraction levels, to support code generation from even more abstract software modelling concepts. Some researchers and commercial tool vendors were wishing to integrate high-level code generation techniques into more comprehensive Computer-Aided Software Engineering (CASE) tool environments, which emerged in the eighties. Another approach to code generation was to build pre-compilers, tools producing source programs which can then be compiled to object code by means of existing compilers. High-level languages, especially the C programming language, started to become popular in the embedded software development community in the eighties. The first CASE tools had also been taken into use by embedded software developers. These tools usually supported certain graphical modelling languages and methods used for software development. One of the most popular methods was SA/SD.

A version of the SA/SD method suitable for modelling real-time embedded systems was proposed in the early eighties, soon becoming soon highly popular. A Finnish tool vendor developed and sold a commercial tool supporting this method. Textbooks and courses on the method became available, sometimes organised around the use of certain tools. A typical software development task would include, firstly drawing graphical models of the behaviour of the system according to the method and perhaps using a CASE tool, and then editing, compiling and debugging the corresponding executable program by means of a C programming environment, for example. A code generator associated with a CASE tool would have automated this task by producing either the source code or the object code from the graphical system models.

At TKO, people had also become interested in SA/SD. In the mid-eighties, in a very short period of time, almost all software engineering researchers took courses on the use of the method, and were applying it in several industrial embedded software development projects. Industrial practitioners took the same course of action, and the Finnish CASE tool vendor succeeded in selling its tool to many of the actual and potential client companies of TKO. Thus, SA/SD based embedded software development emerged rapidly. As researchers, TKO persons became interested in the problem of extending the method and automating the CASE tools that supporting it. One of the ideas for extension was code generation from high-level system models into source code. System modelling notations would be used as a kind of high-level graphical programming language, and the code generator would serve as a pre-compiler for producing embedded software. A pre-study project called Speco on code generation was carried out for an industrial client (firm K) in the mid-eighties, but the input language was not SA/SD and the generated code was not the C programming language. The customer was also not fully satisfied.

This is interesting. I newer fully realised it, but perhaps there was some kind of conceptual fathering away between [the R&D manager of the firm K] and us … First of all, the problem was very complex. [The R&D manager] was wondering why I was proceeding that slowly ... the plan might have been 18 man-months for Speco as a whole. [He] would have needed some kind of journal of what we were doing, and he always had proposals for all problems. I could not write down any kind of journal … At the beginning, I remember that we were having meetings with the walls of the room filled with papers on the semantics of the design elements …We were in the basement of the TKO building then. We were solving the problems at a hectic pace. Minute after minute we were trying to solve some new problem with [yet another researcher]. [The R&D manager] would have needed a detailed journal on how the problem solving process was proceeding … to see, if he was getting some value for his money … Yes, and the preparation of a journal would have required a video recorder, to prevent disturbing the problem solving process … Returning back to my problems with [the R&D manager]. We were both software guys. He could just come tome and say that a particular problem could be solved in that way, and I, after having thought about it for a month, said no, it was not possible … [He] wanted to be right.

A decision was made to propose a code generation research project to the Finsoft software engineering research program established by Tekes, the engineering research funding body, in the late eighties. The researchers closely involved in the Speco project supported this proposal. The proposal was written on the basis of the original ideas of these persons rather than, for example, on the analysis and extension of existing results of automatic programming research. Another possible approach would have been to make a literature survey and to propose how the existing techniques could have been modified for the embedded systems domain.

We performed an extensive literature survey on automatic programming and code generation in the Speco project and collected one big folder full of references. We had become familiar with formal methods for modelling parallel systems in a licentiate course. [The R&D manager of the firm K] presented us the refinement and transformation concepts for Draco. The problem with the traditional approach was its tree structure, it would be better to describe parallel systems as networks.

Now, the idea was more practical in the sense that the proposal addressed the problem of how to transform SA/SD models into C source programs. Both the input and output languages of a tool that would support this transformation were well known to the embedded systems software design community. The proposed code generator would obtain the SA/SD system models from a CASE tool and produce a C source program for a compiler, which in turn would produce the corresponding object code.

I had evaluated related systems, though in a limited way, but using the 'hands-on' style. I surveyed the Draco and Refine transformation tools. I was very interested in Draco, for example, until I realised its problems. On the other hand, I found out its benefits, too. Draco's refinements are actually code components, which the system has been built of. The main obstacle of Draco was addressed by Reino Kurki-Suonio [one of the best known Finnish software engineering professors] at some meeting. With Draco, it is extremely hard to define domain-specific languages. Moreover, the principle of defining the semantics of languages was not good in Draco. The definition of semantics was done by making refinements. In practice, the definition should be based on a more formal model. Refine was based on a so-called wide-spectrum language. This was a strength and a weakness at the same time. The maintenance and management of the language concepts was rather problematic. Yet, I obtained ideas from Refine that were later implemented in ReaGeniX 1.0.

A number of automatic programming researchers had proposed special input languages for code generators so as to generate a full program code. One could say that no practical, industrially usable code generators existed for this reason. Practitioners were not using the kinds of abstract language notations by means of which automatic programming researchers had produced programs. Moreover, many of the languages for which small-scale research prototypes of code generators had been developed, were not suitable for programming embedded systems.

We used Prolog instead, which has actually turned out to be very good for the purpose. Refine, for example, includes a number of features similar to those in Prolog. In Draco, the same features have been implemented by using the Lisp language.

In this regard, the starting point of the code generation research at TKO was quite practical and promising. It was believed that the problem to solve was how to define and implement a mechanism – a code generator tool – that could transform SA/SD models into C source programs. The input and output languages and the technologies supporting the manipulation of these languages existed and could be used in the development of the code generator. It was thus believed, based on the experiences gained from the Speco project, that the project would succeed in producing an industrially usable code generator. Such a tool would have been a remarkable breakthrough indeed.

The persons behind the proposal pointed this out, their aim was to introduce the world's first embedded systems code generator. It could really change the way practitioners were working. This belief was shared by quite a number of industrial embedded systems practitioners. More than ten firms became interested in the proposal, joining the venture as follow-up participants paying a small fee to be able to use the results. A Finnish CASE tool vendor was one of these firms. The Sokrates project was launched in 1988 and it lasted for three years, as part of the Finsoft research program.

A3.1 FIRST PHASE – MAKING OF THE INNOVATION

The Sokrates project, although financed by Tekes, by TKO and to some extent also by the participating companies, was carried out by the researchers who initially proposed the project and then became its key resources. The researchers designed the technical specifications of the code generator based on their own original ideas, and established a research environment where the generator would be developed. This environment, a workstation computer, was physically separated from the internal computer network of TKO, so that no one else could disturb the ongoing work. This decision was made by the project group and accepted by the managers of TKO. The development of the code generator did not require any modifications to the CASE tool which was chosen for the task of producing SA/SD-based system models. There was thus no need to establish any particular co-operation relationships with the Finnish or other tool vendors.

[This Finnish CASE tool vendor, which] took part in the steering group [of Sokrates], did not, however, want to be involved in any real co-operation within the project.

However, the SA/SD method needed to be extended, as it was described in the literature. New development tasks and new kinds of models were defined, according to which the input of the code generator would be produced. The main reason for this was that the original method was meant to be read and understood by software engineers, not by a computer.

The original method was, therefore, not rigorous enough for generating program code. In other words, the rather informal and abstract graphical modelling language was redefined and extended to a rigorous programming language. The method and the language were defined through the project and named as Sokrates-SA. A part of Sokrates-SA was a subset of the Ada programming language, popular in American embedded systems military applications, but not used in practice in Finland. The manager of the project gave courses on the redefined method not only to the industrial participants of the project, but to a great number of other embedded systems practitioners and software engineering students as well. The courses were generally considered useful.

The code generator, as a tool, affected both the development process of embedded software and the kinds of software solutions produced in the process. It was not directed to any particular application domain, such as telecommunication or automation, but was rather meant to serve the needs of generic embedded software engineering. However, the input and output languages were fixed, and therefore also the computer hardware, for which source code was be produced.

New input and output languages would have required developing new code generation rules, as well as adding links to new CASE tools and compilers. The rules were, however, not entirely hard coded into the generator. Although the code generator was designed as a generic software engineering tool, the developers did not establish many relationships to other researchers, while using the results produced by these in developing generic automatic programming tools. Most of the existing tools were based on different input languages, or were rather useless in generating full embedded source code. This situation led to the fact that there was no rigorous theory of the generic code generation technique that was implemented, which could have been scientifically evaluated by other researchers dealing with similar problems.

Well, there was no such theory anywhere in the world. [Two code generation researchers] had made an attempt at publishing the principles of the coding rules, but I have to admit that the paper was not too well written (due to the lack of training and experience), which is why it was not published … Our approach was commented by [a colleague] saying that "we raised the bar so high that it was easy to go under it"… [Another researcher claimed that] No, his comment was that "you will find the highest position of a fence to go under it".

Our approach was all the time that if there was some theoretically interesting problem, we would try to figure out how to do things without solving this theoretical problem. Moreover, we all have been developing these things further afterwards. For example, we all knew what objects were, although they were not included in the approach. The reason was that, in our opinion, objects would not bring anything new to solving concurrence problems. We were reading up-to-date object-oriented literature, but its message was that the operating system would solve all the problems.

Attempts were made at publishing some of the code generation principles in the early phase, but the papers were rejected by reviewers.

The code generation technology was published in the STEP-90 conference.

This consolidated the view of the code generation researchers that they were producing something that was unique, a comprehensive approach that was different from all the other approaches. Later, however, a few papers were published on parts of the work. There was a vivid discussion within the Sokrates project on which case application the code generator would be applied to. Various proposals were made, the project, however, decided to build its own test application, a toy elevator constructed from Lego blocks.

Later, during Sokrates, when I went to [visit firm K] it came to a conflict. They had started a new generation [product] development, new chief designers had come with a new software architecture. [The representative of firm K in the steering groups of Sokrates] and I were talking to the chief designers trying to offer a Sokrates case, but they reacted very strongly, saying that if the Sokrates case were implemented, they would leave [the firm], or something like that… The case would have involved a small processor, on which they had already spent dozens of man-years.

The elevator controlling case had been used as a classical embedded systems text book problem. The project group decided to show how it could be solved by means of the Sokrates-SA method and the code generator. A summer trainee implemented the elevator, which was controlled by an industrial PC computer executing C programs.

Since, despite many requests, we did not manage to get any pilot case from the members of the steering group, the Lego elevator was chosen as a demonstration. We had made an alternative plan [to develop the Lego elevator] for this situation. We activated the alternative plan, because the deadline for starting to work for the case study was coming closer. The Lego elevator was chosen as the alternative for the following reasons: its control was not a trivial matter, including some real problems on concurrence and real-time issues; the problem domain was close to the businesses of the steering group members; I was familiar with the elevator control problem; the elevator would provide good demonstration equipment; the elevator could be developed and tested in a laboratory without any extra travel; damages resulting from errors would be small, and, finally, it would be available as a test environment after the project.

The demonstration succeeded, although rather many of the industrial participants of the project and other researchers were suspicious of using Ada as a part of the Sokrates-SA modelling language.

The reason for choosing Ada was to provide ways of generating different kinds of code using a single input language, and the purpose was also to do research on system analysis – and Ada provided better means of satisfying these needs than C. Yet, I overestimated the increase of the use of Ada. I did the Ada parts and noticed that it was completely senseless to do all the features with Ada. I was never able to get them [the other code generation researchers] to believe that.

A researcher not involved in the code generator project had developed a simpler code generator for one of the key customers of TKO, who took part in the steering group of the Sokrates project. This approach was based on transforming certain parts of the SA/SD models directly into C. It was proposed by the Sokrates steering group and the line managers of TKO that the developers of the code generator would follow a similar path, but the proposal was rejected. This, in practice, prevented the use of the developed generator by the mentioned customer company.

I had taken a look at [the other generator], too. It produced [C] code only from state machines and did not have much for handling data flow diagrams – although the problem in generating code for real-time systems is the management of concurrence [designed by using data flow diagrams in the SA/SD method]. At that time Sokrates was not suitable for letting C code go from specifications to the implementation [as in the other generator].

After demonstrating the generator in the toy elevator example, the researchers talked to the Finnish CASE tool vendor for joining forces to commercialise the generator. The discussions failed to produce any co-operation, the tool vendor had already started its own code generator development and did not wish TKO to become involved.

But the vendor did not admit it. It was clear that [the managing director of the firm] did not want that.

Discussions with other, American vendors, did not result in co-operation either. Yet another line of co-operation was pursued, namely the generation of a hardware description language from SA/SD models instead of C. Also these discussions failed, as the hardware engineering researchers considered the Sokrates-SA modelling language too complex for their needs and were not satisfied with the results of an experiment that had been made. Instead, they started to co-operate with the local university and developed a simpler SA-based notation, upon which hardware description programs could be generated. A university researcher later established a small company to commercialise a tool supporting this approach.

Did the company ever sell the generator? Were there any continuing projects? By the way, VTT advertised the generator in its brochures even before the company was established.

The explicit code generation rules aroused interest in a number of the industrial participants of the Sokrates project. One of them was using so-called programmable logic controllers instead of microprocessors, for implementing distributed automated production lines. In addition, a set of code generation rules was developed for the programming language of the controllers by a student, and the rules were taken in use by the same company. As opposed to the key customer mentioned above, this company was dealing with automation applications instead of telecommunication, the role of software being a supporting technology for the applications.

The emphasis of product development in the company was on other engineering design domains, such as mechanical, process and automation engineering. Most of the company's software designers were lower-level engineering graduates, while subcontracting services were used for implementing automation systems and machine control programs.

Yet another side-track of the code generator development was taken in use by this company, a communication protocol software package developed by the code generation researchers. This software was originally built to demonstrate that a ready-made, generic communication software package for distributed systems could be produced using Sokrates-SA. The protocol software was, most likely, used by the company as a replacement for commercial communication software packages, because it was modelled by using the same Sokrates-SA method as some other parts of the software.

Actually, this is an incorrect statement. Code generation itself is a side-track of a system design philosophy. Another side-track of the philosophy is the communication protocol package. The Sokrates group was trying to emphasise the management of concurrence. The story does dot address this at all. Concurrence was a topic which also interested me and [a colleague]. I've already said that it was not made explicit enough in the story that Sokrates really was a way of solving concurrence problems. This topic was always present in the discussions: how to help designers to manage concurrence. This is not obvious at all in the story, the topic has been treated more as the development of yet another CASE tool.

A3.2 SECOND PHASE – PACKAGING OF THE RESULTS

The Sokrates project ended without any commercialisation or large-scale use of the code generator – rather typically of many research projects during the late eighties. It was more disturbing for the managers of TKO, however, that the only subsequent project based on the results was the development of special code generation rules for the machine automation firm, mentioned above. This project was small, only about two man-months. No further research projects were established, perhaps due to the fact that the original manager of the code generator project had decided to join an industrial company at a late phase of the project, and the new manager did not have enough time to plan any continuing projects.

The generator was a mammoth, it would not have been used by anyone then. It was just a prototype, a demonstration that code generation was possible. Nothing else. It was not even a prototype, just a demonstration. [The firm W] used an 8051 processor, they wanted to be a leading edge firm, they were just looking around. Many others did not have the applications for which the results could be used.

Moreover, Sokrates was really terrible at that time. Someone made their own generators, [the firms S and N] … They took the results from Sokrates, but not from VTT … They could do things based on the Sokrates results, because they had learnt something "too well", and they did not need VTT. The same was actually true with regard to [the CASE tool vendor firm I].

I still believe that the companies who were involved [in Sokrates] managed to make good use of the results. At VTT, we were always told that this was no tool development project … This was another reason for what happened: they told that it was just a prototype … The generator itself was not a ready-made tool. I can't help wondering why there has not been any active attempt at selling the coding architecture. There aren't any brochures on it available either, for example, indicating that we have software architecture knowledge for sale. I would imagine that it has not been sold in any other way than in projects in which it has for some reason been used. Software architecture knowledge has merely been taken in use in projects by chance, and it has not been viewed as a marketable thing.

Towards the end of the Sokrates project, we had extensive discussions with the companies involved on how to utilise the results. Something had, however, happened in the companies. In many cases, the persons had changed. The structure of their products had changed to such an extent that these things were not topical any more.

On the other hand, since there were no scientific references on the developed code generation techniques, the hopes were only faint for proposing any new European code generation research based on the results. This kind of thinking, from Tekes-funded national projects to EU-funded European projects, was followed at VTT during the early nineties.

No one ever mentioned anything of this.

The original manager of the Sokrates project returned to TKO after his leave of a few months. He helped the project group to prepare for the external evaluation of the project at the end of the Finsoft research program, carried out by national and international experts on behalf of Tekes. The evaluation results, both industrial and scientific, of the project were very good indeed.

There were also some foreigners involved. Their conclusion was that "the research group is on a par with top code generation research in the world". In other words, we were right at the top of code generation research.

However, another evaluation performed by foreign scientific experts on behalf of the Finnish Academy produced results of completely different nature. The evaluators were very unhappy with the lack of any scientific evidence of the principles and novelty of the code generation techniques.

I believe that they did not have anything against code generation, but the design philosophy, because it was based on SA/SD. They did not say anything negative about code generation (you should check this out, if you wish to hold on to your claim).

The researchers wanted TKO to invest money in the commercialisation of the code generator. In connection with this proposal, discussions were conducted if it was right for a national research institute to invest taxpayers' money on developing software tools that would compete against truly commercial CASE tool vendor companies. A decision was, however, made to invest some money on the further development of the generator.

In particular, the Ada language was now replaced by C, as was proposed to the project group already some time earlier. Some brochures and usage manuals were also written, and a licensing policy established. The generator was renamed as Reagenix.

This was not the case. Reagenix is no redesigned Sokrates. It was designed completely anew based on my experience from the pitfalls of Draco, Refine and Sokrates. It was developed in a pre-study, and made ready with surprisingly little effort. From my point of view it was a mistake that it was taken as the continuation of Sokrates and therefore no one was interested in it. On the other hand, no one knew which skills were needed for developing it.

What was worst in this story was the indication that the Reagenix code generator would be the same as Sokrates. My opinion is that Reagenix has been developing slowly here and there, and that there is no clear link between Sokrates and Reagenix.

It is interesting that Reagenix was referred to as some external tool [in the projects carried out by TKO], but nothing was paid for its use. It should have been told that we had this kind of tool available, and some money should have been allocated from the projects to its development.

We had already made our own generator, before the generator of [firm I] was introduced, and we had set its price as we thought was appropriate. [Firm I] launched its product some six months later, and they had a much higher price. In other words, we did not know their price, otherwise we would have used the same price. By the way, Reagenix was introduced in the Technopolis Oulu, before [firm I] introduced its own generator. You [Veikko Seppänen] were not there were you? You were in Japan and therefore we could do what we wanted. [The former manager of the Sokrates project] was the section head.

When Reagenix was introduced, [firm I] needed to reduce the price of its … code generator to half of the original, which was more than 60 thousand marks, but they had to start to sell it at the price at which we were offering Reagenix, 30 thousand marks. I believe that we really caused business problems for [firm I], due to the fact that Finland is such a small country. [The firm] had its own marketing organisation which had to face the problem of everyone asking them about Reagenix, especially since the front end was the same, …, and we had claimed that Reagenix can do everything. Yet, [their generator] was good for generating sequential code only.

Another important extension of the code generator was developed, an operating system model and its prototype implementation. The operating system could be used instead of commercial operating systems as a part of programs produced by the code generator. The idea behind the operating system was patented by the code generation researchers, and it was thus truly original. Some parts of the idea were used in an industrial project carried out for the above-mentioned key customer of TKO. The code generator developers were, however, not involved in this project. The work was performed by another person who was an expert in the customer applications and had conducted several contractual projects for them.

Except that we told [the colleague] how to do it! … It was [his] project. We sold the idea of the single stack operating system. [A person from firm N] was familiar with the idea and supported us – he took part in the steering group of Sokrates.

Investments in the further development of the code generator became a difficult matter after some time, from the viewpoint of the VTT managers, because no considerable contractual or research projects had resulted from the investments.

Yes, the point is that it is not reasonable to think that as early as one year after finishing a [research] project the customers would be there. It may take three years.

The developers of the generator focused, instead of large-scale projects, on small-scale consultation and teaching of the Sokrates-SA method. A new side track of testing was, however, established. The idea was to use parts of the input models used by the code generator for testing embedded programs. This work was a small project done for a private company, and it resulted in yet another tool prototype. The tool was marketed to other companies, but without much success. There were, again, some Finnish and foreign competitors, and the particular testing problem tackled with the tool was perhaps not considered the most serious one by industry.

We first carried out the Sympa project, an analysis of the improvement of their software development practices. Unit testing evolved through that analysis. Once again, it is all the better that small expenses yield the great benefits … as happened in this case, but the organisation [TKO] did not support us in the further development of CUTE. Yet, it was transferred for example to [the firm A], although they had a horrible situation there. Their subcontractor had developed a massive software architecture that needed to be tested before it was taken in use. However, it was not just testing, we also had to deal with the architecture. As the well-known subcontractor and was using a new object-oriented technology, we had to tell them how to test the architecture. Therefore, we just took the consulting role, by commenting what [the firm A] itself was doing. I still believe they got what they wanted, although they could not invest in CUTE as such.

The original developer of the Sokrates code generator aimed at writing his doctoral thesis on the results of the Sokrates project, which would have resulted in a comprehensive scientific evaluation of the basic principles behind the method and the tool. The plan has, however, not yet materialised.

The dissertation would actually not have resulted in this, had it been finished.

We did not have the university background. It [scientific documentation of the code generation principles] should have been done as a part of the Sokrates project. … In my opinion, it was the university attitude [that was missing]. … There was no one to guide us in doing it … [a colleague] had high criteria for publishing, for us they were clearly too high, we should have studied how to describe these things formally.

Another problem was that as we always had two to three customer projects going on at the same time, there was quite simply not enough time for writing any documentation. At the end of the Sokrates project we made a mistake in trying to develop Sokrates [further] without publishing anything at that point. We should have stopped the development during the last year, to write publications. … Later, SA/SD was not any more successful with regard to publishing.

Increasing interest in object-oriented methods also made further investments in an SA/SD code generator more difficult. The focus of research on generic software engineering methods was moving towards these methods instead of the SA/SD method.

At that time, I was following the evolution of object-oriented techniques quite closely. I had already become familiar with these techniques during my diploma thesis work. I was listening to the praising of the techniques by Risto Suitiala [a researcher at VTT Information technology laboratory] and I realised that there was a lot of potential in these techniques. Refine supported some object-oriented features and transformations from these to code. I had discussions with [a colleague at TKO] on how object-orientation and real-time systems would fit together. I was using Object Pascal in my spare time to design a Windows-based user interface program. I even wrote tentative transformation rules from object life-cycle state diagrams to an object-oriented programming language and presented them [to another researcher], who applied them in [a customer project].

The original Sokrates project group had been dissolved, but kept together, in practice, and worked even during their spare time trying to advance code generation techniques. The biggest problem, from their viewpoint, was that the line managers were not willing to invest much money, but were rather looking for larger contractual application projects based on the existing generator, or for an extension of the generator's input language to an object-oriented formalism.

Actually I did not work that much in my spare time, except correcting some errors [in the generator]. As the [research] customers at VTT using the generator did not pay any license fees, these small tasks needed to be done [in my spare time].

The few application projects involving the code generator, lasting a few man-months at most, resulted in a very small return of investment from the viewpoint of the VTT managers.

Who tried to initiate these projects? I was never given any information, although I would have had some ideas and the skills for the job.

The relationships between the line managers and the code generation researchers started, therefore, to cool down. The managers had already been somewhat suspicious, as they had been expecting interaction with other researchers, scientific validation of the results, and a considerable number of continuing projects. The researchers, on the other hand, had been expecting support from the managers for the further development and marketing of the generator.

The relations had cooled down before Reagenix already. [Another code generation researcher] had at that point already lost his interest in project marketing. It had happened several times that he had sold an idea to a company, and then either the representative of [firm I] or [the manager of] VTT had come to criticise Reagenix. I did not yet have any appropriate contacts and I had not received any positive feedback on my work [on Reagenix], so I was not interested in marketing. I have to say that for this reason I was not fond of project marketing even later.

It is difficult to find any motivation for marketing, if your ideas are first heavily criticised within VTT. If you succeed in marketing a project, there is always a fear that if something goes wrong, somebody will come and tell you that the failure could have been expected at the very beginning …The [MCS-REA] project was carried out when [the former Sokrates project manager] was working as the section head. Indeed, the … project was the only one that succeeded, from the viewpoint of TKO, and [the former project manager] happened to be the head of the software engineering section then.

The managers of TKO said that this was not good, this was a prototype, a hack-hack, and we could not seriously consider offering it to industry. Now, after being in industry for four years, I have seen what kinds of tools are being used for example in the telecommunication area: the tools developed by ourselves and by universities, public domain tools, and our competitors' tools. I must say that Reagenix was, after all, a very reasonable tool.

The impression that was given by the managers of TKO was that industry was using only top-quality, well-packaged tools, offered by reliable parties. This was the point, VTT did not see itself as a reliable tool developer.

During the Sokrates project, the big generator did not arouse interest in anyone, because it was run on a workstation and was slow, expensive and difficult [to use]. But afterwards, the light-weight [code generator] version Reagenix was produced, which was running on a PC. However, TKO did not want to sell it any more, because [firm I] was seen as a strong competitor. We should have gone abroad, or establish a company. … But the problem with such a company would have been the fact that we had the front end [CASE tool from firm I]. … That was a clear problem. In other words, the developer of the graphical front end of our generator had its own competitive product.

During Sokrates, which was a research project, the participating companies got what they wanted. Afterwards, the generator should have been commercialised and taken abroad. In Finland, the big customers had already taken part in the Sokrates project. Otherwise there were only small companies who still could have bought the generator and used it.

By the way, I established several contacts with companies abroad, and in some cases I had quite lengthy interactions with the contacting parties. At the phase, at which an offer was requested and I sent them the information, including the fact that the graphical front end which was needed could be acquired from [firm I], the contact would always end.

I guess that they contacted [firm I] and asked about Reagenix, and [they]would certainly give their opinion of the matter!

The story does not describe clearly enough that whenever we were trying to take the Reagenix results to the market, the organisation [VTT] would always say no, in one way or another. For example, I was having discussions with [an American CASE tool vendor] concerning the integration of their CASE tool as the front end of Reagenix, and I had also ensured a case example from the Sodankylä observatory, but then [the section head] came to argue with me about me dealing with foreign competitors [of firm I], which was not suitable. The co-operation failed, because the managers said that we would not compete against [firm I] in this matter. I don't know, [they] perhaps felt that I must be hauled over the coals.

However, what was good about the development of the code generator was that we did implement it, after all. What went wrong was that the generator failed to get sold. I believe that we simply did not know how to sell it, then. There were no previous experiences of this kind of a matter [at VTT]. I would say that we did not have the culture of selling. We were improvising. We, or at least I, did not know what would be the ordinary way for selling such things at VTT. We were just doing our best to invent something, but that was much harder than finding out how to implement the generator.

I implemented the first "big generator" as my diploma thesis. One of the first feelings that came over me when reading the story was a feeling of sorrow, considering the amount of effort that was spent and how bright the people involved were, and how someone else [at VTT] would then be looking for some formal aspects to make this group of people look incompetent, and to create a public opinion of the group not knowing anything about any real problems. Another thing that went wrong was that the SA/SD method was not a very good choice. [The CASE tool of firm I] was a nice SA/SD drawing tool, but the choice of the SA/SD method was a failure.

We were [also] thinking about establishing a company. But we had, for example [another code generation researcher] and I, already been running a firm for several years. I had been the managing director of my own firm for seven years, [the other researcher] for three years. The firm, as such, was not an interesting idea. We knew what that would have meant. To start a company, we should already have had some customers for the product as well as an established market.

A3.3 THIRD PHASE - EXTENDING THE TOOLS

VTT was reorganised in 1994 to include nine big research units. TKO became a part of VTT Electronics (ELE) and the former section of VTT Electronics laboratory dealing with automation systems a part of VTT Automation (AUT). The researchers of AUT were still interested in the code generator. The basic need of the generator was in rapidly implementing and testing new machine automation algorithms. The generator appeared to be a very effective tool for this purpose, if its users were willing to design the algorithms by using SA/SD. For AUT researchers this was no problem, they had been using the method for several years.

For them, the code generator offered a higher-level, graphical programming language and environment for systematic and rapid production of implementations involving machine control algorithms. Also the line managers of AUT considered the code generator as a very good tool and could not understand, why such an excellent tool had not been utilised in a larger scale at ELE.

But they did not pay anything for it.

The viewpoint of ELE had, however, been to develop and market the tool to professional embedded software developers working in organisations where software design dominated embedded systems development, software design was carried out on a daily basis, and the software needed to be maintained for a long time. This kind of use was different from the needs of, for example, small-scale implementation of new machine control algorithms by researchers focusing on other technologies than embedded software as a means of implementing computerised systems.

I carried out the Kaasu project, of which the customer was very satisfied indeed … We had an article in Tekniikka & Talous, the technical newspaper, about the code generator, and they called us – to me, if I remember it correctly – right away, saying that they would need a control system [to be developed for a gas cromatographer]. My name was mentioned in the article. That must have been why we got the project.

The user gave a profile on how to implement the valve in the equipment, the control system followed the process. They produced a commercial system of the equipment … In that phase, it was a prototype though. They did not have software designers of their own. We did not sell Reagenix, but the control software for the equipment. It was done with Reagenix. An easy job. The schedule held. Nice equipment.

Similar small-scale success stories of the use of the code generator emerged also within ELE, e.g. in connection with the research on fault diagnosis systems. In a few projects, especially in one research project where a computer controlled wood carving machine and its diagnosis system were developed, the code generator was successfully used for implementing both the wood carving algorithm and pieces of the diagnosis software. The needs of this project with regard to the development of embedded software were much similar to the needs of the researchers of AUT.

Tulko is an important reference that is missing in the story. The Tulko demonstration system was built by using Reagenix and the SIC communication protocol. The final demonstration system was a full-scale real-time robot, with a perception system and distributed robotics tasks.

I never got any positive feedback from the small-scale success stories. Yet, at that point the whole [Reagenix] generator consisting of about 5000 lines of Prolog code had been implemented by myself.

Despite the success of these projects, no larger-scale use of the code generator would emerge. One of the main reasons was that there were still no large projects in which the generator could have been applied.

All the time there was the problem that I should have been selling projects, but the only thing that I could do was to go and ask if they had any projects which I could carry out using Reagenix … There was no reason to sell Reagenix to any project. I would not have earned any extra value for developing Reagenix. The projects did not pay anything for the use of Reagenix … We were lacking internal charging on the use of Reagenix … Instead, I have been blamed for having produced hell of a system. I have, however, developed a system that is being used by at least a hundred people … I have also been blamed of always doing everything with Reagenix.

I also see that Reagenix, the technology itself, is not all that beneficial to VTT. We could sell design projects instead, and use it for analysis and simulation. I am just saying that the project portfolio has not been good: the benefits go to the users of the tool, not to the developers.

I will now tell you a typical story. We were at a Hi-tech conference to demonstrate Reagenix. People from [a possible customer company] came to talk to us. We were talking about design methods, generator, etc. and agreed on a visit. I went there with [a former Sokrates member] then, carried out some requirements analysis and summarised what TKO could do for them. They needed, for example, DSP in their system. The real problem for them was the management of the entire product. We proposed Reagenix for this. We had analysed the situation. The problem was to manage the entire product. They were very interested in Reagenix, as it could be used as integrating technology for DSP, communications, etc. We returned to Oulu, told what the customer had said, and Jukka Karjalainen heard the word "DSP" and sent some DSP project offer there, and that was the end of it. In conclusion, all the things we were managing by ourselves went well, but if the organisation got involved, difficulties would arise.

I have never met a dissatisfied [Reagenix] customer, if the customer sincerely wished to learn how the things were. I have also carried out projects in which Reagenix could not have been used, the Monitori project, for example, in which the system architecture had already been fixed, including DSP and other parts.

The developers of the generator had perhaps also made the other researchers of VTT tired. At a personal level, some people seemed to discriminate anything related to the code generator.

No one ever told me about the success stories. I could never take part in the reporting of the results of the projects [where Reagenix was applied]. I just gave consultation and corrected small errors, if I had some extra time available. It is possible ascribe the fact to the industrial projects related to code generation being small in two ways. The first is that it was small business for VTT, and therefore an unfruitful line of research. The second is that the techniques and skills which were developed were highly efficient. The problems of the customers were solved with little effort and very fast. A good example is the development of the two-phased gas chromatograph [for one of the institutes of VTT]. The work took only little over one man-month.

Our idea was to solve problems and not to begin to profit from the customers. Now we come to the fact that I and [another code generation researcher] had been involved in business for a very long time ... and there were some differing viewpoints. We learnt how to solve the customers' problems and when they learnt that by associating with those guys their problems would be solved fast, they would come again to us in some other matters … In other words, we took a customer-oriented approach … We thought always about the value-for-customer. We have in my opinion taken care of this aspect much better than VTT projects on the average.

A3.4 FOURTH PHASE - STRUGGLING WITH LOW INTEREST

Yet another attempt was made to extend the code generator, by investing money in the development of a graphical debugger integrated with the generator. This debugger would have been a novel mechanism, making the generator look even more like a graphical programming environment. Such tools were available only for ordinary programming languages, such as C. The CASE tool sold by the Finnish vendor did not include any debugger. Again, no contacts could be established with this vendor, but instead a small software house was hired to implement the debugger.

It was proposed by [the managing director of the software house] … It looked interesting. Moreover, some people had been complaining about Reagenix not including any graphic debugger. Aniprosa seemed to offer a possibility of carrying out research on graphic debugging.

No internal contacts were made to the group of ELE which had been doing research on the animation of graphical system models for almost ten years.

This is not completely true. We had knowledge of how the other animator was designed. It appeared to be too much tied with a specific SmallTalk/Petri net based implementation. The small software house was used, because it had already implemented an animator for SA/SD diagrams. Politically, this was apparently unwise, because the animator turned out to be a in [firm I's] side.

The debugger was implemented, but it did not result in any new contractual or research projects where the generator could have been applied.

It was used in one project, though … It was a mistake to allocate the Reanimator work to the [Reagenix] account. I have been developing

Reagenix with the money I have earned. A hundred thousand marks was spent on debugging, which was not of any use after all.

On the other hand, no serious attempts were made to extend and modify the code generation techniques for the needs of the rapidly emerging object-oriented software development methods. An example would have been a means of generating C++ programs (the most typical object-oriented programming language) from an object-oriented system modelling language, using the generator and some object-oriented CASE tools.

I would have had ideas, and I was trying to present them occasionally. Yet, no one of the managers of VTT was interested. I even designed a prototype of a generator in a few hours for the object life-cycle method after I had taken a C++ course. This principle was later applied [in a customer project].

Several attempts had been made by the line managers to make the code generation researchers interested in other subjects, so as to ease the situation resulting from the fact that internal investments in the generator could not be continued and that there were no considerable projects in which the generator could have been applied and extended.

I was irritated by the ordinary research results of TKO that did not form any synergetic entity, but were useless when put together – as we would have developed the rear shaft of a Porsche, the gearbox of a Lada, the engine of a Cessna, the body of a Zetor, and so on. Yet, I am satisfied with having expressed honestly what I was going to do. The customary way was to write a proposal using normal research liturgy on how the research would result in new challenges and produce immediately applicable results, and then continue the old line of research. I did not like this kind of humbug. Still, I was able to make the management's idea on my thinking crystal clear.

In one of these "new" projects methods for the implementation of machine vision software was studied, another focused of software component reuse techniques. In both cases, however, the code generation researchers used the projects, in part, to extend the code generation techniques.

If you are referring to the Rekki project, I must say once again that for the most part I did all kinds of tool evaluations and tried to find a reasonable approach to carrying out the work together with the university. Yet, the university was interested only in continuing its own Khoros-related research, which was completely non real-time oriented. We did not do any code generation related research at all in the project. If this [component reuse techniques] refers to the Komppi project, you are presenting just another one-sided viewpoint. I was not able to participate in the literature survey part of the Komppi project. Code generation was not developed further at all in Komppi. I made it very clear, because I was aware of the attitude [of the managers]. I find it very unpleasant, if you use Komppi as an example.

This somewhat irritated the managers, and also some of the other researchers involved in the projects. The results of the projects were accepted by the funding bodies and the industrial partners who were involved, but did not result in any larger-scale use of the code generation techniques.

The American style marketing irritated everyone. They perhaps thought that we would produce some equipment that would free the designer from thinking of anything any more. Instead, we were aiming at defining a language that could be used for solving difficult problems. This was in contrast with what everyone thought. They did not understand … It caused trouble inside TKO, though not outside.

Some code generator licenses, each worth of two man-weeks' contractual development work, had been sold to companies and given for free to some educational institutes. Often this was done as a side job of some development project.

Educational institutes paid modest license fees, VTT institutes used the generator for free ... I still remember how difficult it was for me when I first joined VTT thinking that I could not produce anything. The contribution of investing five thousand in something which would allow industry to make millions was not realised … Our point was that we were offering solutions to customers by which they could produce things cheaply and fast. If the customer could get something at fifty thousand marks, for which they had spent half a million marks earlier, that was only good. This was our way. … And that [kind of projects] could have been sold … One thing is that we were not allowed to sell other than our own work. It was only at the time when I was already leaving VTT when it was told that money could be earned from projects.

Was the line organisation thinking that after investing five millions and it resulting in such small income, it was not sensible to use the developed technique? It could have been developing slowly, after all. A related example is an article published in the Harvard Business Review on the investing and harvesting phases of technologies, as well as the decline. When people read these papers and thought that the whole cycle was five years, General Electric gave up computers, before the decline phase would start. Xerox also gave up computers, although they had object methods and windowing systems [already then].

By the mid-nineties, the original group of code generator researchers had dissolved totally, because the persons were working in several projects not involving code generation techniques. Two of the original developers left VTT in 1996 and 1997, and it seems that this line of research has finished at ELE.

You mean the Sokrates group. I am still here, with all the knowledge. I have sold three licenses during the past six moths, and two other firms have shown interest in it.

I have also organised three courses, each two or three days. I answered to the most recent customer request last week. The total income from industry was about 100 000 marks last year, and the expenses about 30 000 marks. How right were you, after all [when stating that the story was finished]?

Afterwards, the same license plus training was sold to another company, for about 30 thousand marks. Another license was sold separately. They paid my travel expenses, too. In summary, three licenses have been sold after Summer 1997. According to your story, this line of research would be dead! It isn't. The front end is different, it is not [from firm I] any more. They wondered [the buyers of the license], why we had not been marketing the tool. I told them that I did not have the time at that point.

From the viewpoint of my short industrial experiences, I would say that VTT has no intrinsic value by itself. It should support the Finnish industry. If the guys have made themselves unnecessary, VTT can be closed and the persons can go and find jobs in industry. Then, a new research organisation can grow if it is noticed that industry does not know enough of some new topic, and the organisation can look five to ten years forward on behalf of industry. VTT is needed for such a purpose, to hold a position at the border of industry, to do what industry itself is not doing. And to help small companies. VTT should not have any intrinsic value by itself.

But for some reason we did not dare to sell this idea of using Reagenix inside TKO. We did everything with traditional methods even inside TKO. In the projects where I was involved, the initial setting was already such that Reagenix could not be used. I always joined the projects in a phase when it was not reasonable to use Reagenix any more … It was not seen at the level where the projects were prepared. It was just the few guys who were involved in Reagenix. For this reason, I should have spotted the customer first, and I should have realised that I needed to sell a certain kind of project to the customer, and then it [the use of Reagenix] may have emerged. Inside TKO, they did not believe in it, they preferred programming from scratch in all projects.

Moreover, if you charge on the basis of working hours, it would not be reasonable to use Reagenix. If the charge had been by the hour and the use of Reagenix had resulted in less hours, the use of Reagenix would not have been reasonable. We made a mistake in the sense that Reagenix was used in many places, in fault diagnosis projects and in the Electronics laboratory, and so on, but we could not get any benefit from the use of Reagenix. The license fee should have been included in the expenses of the projects.

The effort that was saved by using Reagenix could have been used for the further development of Reagenix … Not even licenses were sold, because that was considered an extra. VTT got it for free … Is there even any information of how extensive the use of Reagenix in the projects was, if it is not explicitly mentioned. There were rather many people who became interested in Reagenix.

Reagenix should have been used as part of research projects. And it was used, wasn't it. Another thing is that we should have had Reagenix-based projects. However, if a firm comes to VTT asking about the development of some product, the firm will contact a person who can be found on the list [VTT's service directory, other lists of contact information] and that person actually decides which methods and tools will be used. No one contacts me. I do not have any contacts myself.

A3.5 SUMMARY OF THE CODE GENERATION STORY

There are certain key technical and human reasons for the developments of code generation activities that took place at VTT, which are closely related to each other. From the technical viewpoint, the developed code generation solutions seem to have stuck with the SA/SD method, when industry was already gradually taking object-oriented methods in use.

Did it really took object-oriented methods in use? After all, the SDL method with code generation features has become increasingly popular, and it resembles SA/SD … One can also think that the five million that were invested were for the need of industry. What if the investment had not been made? The good point that was not told in the story is that the SA/SD method was brought to industry. [One of the code generation researchers] has a remarkable role in that. SA/SD is perhaps the first real design method in the area of embedded systems. [The researcher] is among the first persons who giving training on this method in Finland. Earlier, there had only been foreigners ... At present, when I talk to anyone out there, they do not know me and they are saying that they use SA/SD. When I ask why they are using it, they will tell me that VTT has brought SA/SD further … Yes, indeed … Some of those persons had taken my course.

Object-oriented methods emerged rapidly, but no serious attempts were made to link the code generation expertise to them.

There is a clear difference between Reagenix and the object-oriented world. Reagenix takes care of real time, concurrent control and supervision sequences and asynchronous communication between them. Object-oriented methods, in contrast, can be used to take care of traditional computing and management of stored data … The group [of code generation researchers] did not omit object-oriented methods due to ignorance or arrogance. The concepts were familiar [to the group] already from the computer science journals of the beginning of the eighties … The possibilities and problems were known.

Yet, the developed solutions were very flexible with regard to applications, target hardware and even CASE tools. Although the code generation functions that were developed were hard coded inside the code generator, their principles were made explicit by coding rules.

Why were there no serious attempts? I was never even dropped a hint of this kind of possibility. Usually anything related to code generation was flatly knocked out as Reagenix bullshit.

My experience was hardly ever used. On one occasion, I may have made a one week evaluation on the generation of code from Statecharts models.

From the viewpoint of networking, the isolation of the code generator researchers both internally at VTT and externally from the key customers proved to be a problem.

Yes, indeed, from outside it looked outside like a homogenous group that did exactly the same things. By the way, [a colleague] always said that whatever he asked from anyone of us, he always got the same answer. We had thus the same understanding, although we did not write it down, perhaps could not write it down … It was a well functioning unofficial organisation! … In the MCS-REA project, for example, we had very difficult problems that were solved by means of brainstorming.

Moreover, disagreements arose with some technology experts, such as hardware designers, who may have benefited from co-operation. Yet, several successful usage cases of the code generator can also be identified. Most notably, the tool proved promising for the prototyping needs of other than industrial software professionals, for the needs of researchers, actually! Networking with these people, and perhaps with commercial tools vendors that they used, might have resulted in some kind of commercialisation of the code generator technology. Internally, a number of persons adopted a negative or at least suspicious attitude to the code generation research.

The message that the idea of the code generator with all its supporting systems being capable of solving "all problems" was particularly irritating. One could speculate whether a similar development had taken place in industry, if the developers of the generator had succeeded in marketing it to a greater number of customers.

Why wasn't any bigger business created around code generation? VTT did never come to an agreement with itself upon what it wanted from code generation. The lines of continuing work that were brought down include:

- *Co-operation with the American tool vendor – could have resulted in a market worth of millions; and*
- *Development of code architecture for the 8051 processor – would have resulted in a market of hundreds of thousands of marks, because Finland was full of machine firms that were not interested in programming operating systems calls.*

There were a lot of people showing interest at [automation] fairs. The idea was to investigate the design phases that preceded coding. The core competence is there, but it can not be seen, in other words, the knowledge of how to solve concurrence problems … and the development method, analysis and so on.

It seems that those industrial practitioners who actually used the generator or the code generation rules sought for solving the software development problem by focusing on, for example, automation design and trying to make programming a rather mechanical task that could be carried out with a tool.

For them, the message mentioned above was promising. For people who where developing software as their profession, the message did not go through.

Why real software professionals not interested, but just machine automation people? Did portrait painters become enthusiastic about cameras in the late eighteen hundreds?

The developed code generation technique seems at the first sight to make some central software engineering skills obsolete … I do not know, if software professionals became interested in the first Fortran compilers, because it made obsolete a great deal of central skills [in the fifties], but at the same time it made it easy for mathematicians to program algorithms by themselves.

I myself took even greater interest in code generation than many others, because for seven years I had been risking my own money in tasks contracted at a fixed price. It always paid to think thoroughly how to do the job at the smallest effort. One very satisfied customer was [the firm Se]. How long do you believe that the development of the software package for their product took?

It took only 17 hours! I did a demonstration system during two nights in a hotel, then I showed it to [them] and asked, if it would be embedded in the target hardware. We spent five hours on a Friday evening doing that, then … Software developers still have many problems these [Sokrates/Reagenix] things have not become obsolete at all.

During one course, a technician said to me that when he had told his colleagues about this method [Sokrates-SA] and the generator, they had really become frightened and thought that if these kinds of tools were used, they would lose their jobs … They could not see that they would then be able to focus on higher-level issues, new designs and so on.

I would say that this is a rather non-democratic issue. There were smart guys [in firm Nm]. Things can be non-democratic … It was an appropriately sized group. The chief designers were programming themselves. In industry, some people just work from eight to four and do not want to learn new things. They use what they already have learnt. Other groups may realise that they can utilise some new things … The people at [firm Nm] were very smart indeed … And they were very busy, too … They were not interested in coding, but in getting their equipment ready. They did not get any kick out of programming, but of the fact that the equipment was completed … They saw their job not as coding, but as implementing the equipment.

Published by

VTT
Vuorimiehentie 5, P.O.Box 2000, FIN-02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication
VTT Publications 392
VTT–PUBS–392

Author(s)
Seppänen, Veikko, Alajoutsijärvi, Kimmo & Eriksson, Päivi

Title

Projects or products: seeking for the business logic of contract R&D

Abstract

This research addresses the question of building and exploiting competence in connection with contract research and development (R&D), by means of a longitudinal case study. The research involves VTT as a supplier and internal research customer, Tekes as a funding body and several firms as external industrial customers. We are looking into their mutual relationships to gain a better understanding about the evolution and exploitation of competence on the so-called code generation techniques used in the development of software embedded in electronic products.

The analysis of the code generation case is based both on written material and on interviews of persons involved in code generation related activities at VTT, Tekes and industry from the mid-eighties to the present date. The building of the code generation competence of VTT is analysed and explained within the R&D process based on project relationships. The marketing and purchasing of the competence is also addressed. Differing logic of action of the interacting parties have been found to affect the evolution of competence, within networks established for creating and making use of the competence.

In the code generation case, the managers of VTT aimed at creating a growing portfolio of fully contractual project relationships, involving machine automation firms, in particular. The researchers favoured marketing the competence as a commercial style tool, with a minimum of tailoring done in projects. The customers of VTT had difficulties in coping with these two logic of action, in a rapidly and radically changing business environment. It may have been for this reason that the competence was, after all, utilised mainly by VTT itself in joint research projects.

This did neither benefit its developers nor did it advance the evolution of the competence. The differing logic of action of the two key parties, which resulted in the lack of any considerable portfolio of external customer relationships, lead to a rather rapid withering of the competence at VTT. However, the developed code generation technology has recently been sold to the former researchers, who have established a company based on their own business logic. This kind of competence survival through many years and despite conflicting viewpoints is, after all, one of the key factors in making business out of research.

Keywords
core competence, industrial relationships, logic of action, embedded software, research and development

Activity unit
VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland

ISBN 951–38–5391–8 (URL: http://www.inf.vtt.fi/pdf)			Project number
Date September 1999	Language English	Pages 110 p. + app. 104 p.	Price
Name of project SOH.JULKA		Commissioned by VTT Electronics, The Foundation for Economic Education	
Series title and ISSN VTT Publications 1455–0849 (URL: http://www.inf.vtt.fi/pdf)		Sold by VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374	