Challenges of software-hardware co-design

Prestudy in TWINS project

Juha Takalo, Jukka Kääriäinen,
Päivi Parviainen & Tuomas Ihme

ISBN 978-951-38-7150-5 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)

Copyright © VTT 2008

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O.Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax +358 20 722 4374

VTT, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde 020 722 111, faksi 020 722 2320

VTT, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel 020 722 111, fax 020 722 2320

VTT Technical Research Centre of Finland, Kaitoväylä 1, P.O. Box 1100, FI-90571 OULU, Finland
phone internat. +358 20 722 111, fax +358 20 722 2320

Technical editing Anni Repo

Series title, number and report code of publication

VTT Working Papers 91
VTT–WORK–91

Author(s)
Takalo, Juha, Kääriäinen, Jukka, Parviainen, Päivi & Ihme, Tuomas

Title

Challenges of software-hardware co-design
Prestudy in TWINS project

Abstract

The purpose of this publication is to summarise the results of the survey carried out in the TWINS project during May 2007. The survey was carried out among TWINS ITEA project partners. The goal of the survey was to get an overview of the tools and methods used, and the challenges faced in requirements engineering and management, architectural design and product information management by TWINS partners.

The publication also presents the areas of co-design and some viewpoints that are addressed during the TWINS project.

ISBN
978-951-38-7150-5 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN		Project number
VTT Working Papers 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)		6086
Date January 2008	Language English	Pages 45 p. + app. 2 p.
Name of project TWINS	Commissioned by	
Keywords co-design, requirements engineering, requirements management, architectural design, product information management, PDM, PLM, CM, survey, results	Publisher VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4520 Fax +358 20 722 4374	

Preface

With the growth of electronic products' use in everyday life, the development of these systems (the design flow for co-design in electronic products) is increasingly becoming more complex and important. Modern electronics products have to meet high reliability requirements even under tough environmental conditions, such as variations in temperature or alternating electromagnetic radiation. Although different disciplines are closely linked to the end-products, the development of electronic products is often a rather sequential and mono-disciplinary process. Typically, the mechanical part is designed first, next the hardware infrastructure is fixed and finally the embedded software is developed. This way of developing electronic products creates problems above all for software engineers. Furthermore, the software's role in electronic products has increased to allow the implementation of functionality, parallel versions of products, and customisation of systems.

TWINS is an ITEA project (no 05004) that aims at optimising the design flow of software intensive systems, in a context where different disciplines (e.g. software and hardware) are involved. The TWINS project addresses the co-design problems of product development consisting of integrated hard- and software development. Challenging topics in this development mode are the co-specification and allocation of requirements, co-optimisation of HW/SW architectures, lifecycle management and the configuration management of evolving products and components (hardware or software), and the improvement of testing multidisciplinary products.

The authors would like to thank the TWINS project partners for participating in the survey.

Contents

List of symbols

ALM	Application Lifecycle Management
CFSM	Co-design Finite State Machine
CM	Configuration Management
DEVS	Discrete Event System Specifications
HW	Hardware
IPN	Interpreted Petri Nets
OMT	Object Modelling Technique
PDM	Product Data Management
PeaCE	Ptolemy extension as Co-design Environment
PLM	Product Lifecycle Management
SADT	Structured Analysis and Design Technique
SASS	Structured Analysis and System Specification
SW	Software
VHDL	Very High Speed Integrated Circuit Hardware Description Language

1. Introduction

The tight competition in electronic product development has led to hard lead-time requirements, i.e., software together with hardware must be developed more rapidly. This necessitates several improvements in the design flow, to design, implement and validate better HW/SW engineering and management methods, and techniques.

To improve this design flow, the TWINS project work package 2 focuses on creating and validating better technologies for requirements engineering (including development and management), architecture design and co-design management. The first step in the project was to find out the current state of both, industrial practice as well as research regarding these areas. To discover the state-of-the-art, literature surveys were carried out and to discover the state-of-the-practice a questionnaire was carried out. This publication describes the results of this questionnaire.

The TWINS project has 22 partners from Finland, Belgium, France, Spain and the Netherlands. The project consortium involves partners from a wide range of industries like automotive, avionics, copiers and printers, power supplies, automation systems, and telecommunication networks. The project wants to improve the project partner's competitive position by increasing the quality of the products while reducing the time to market.

This publication summarises the responses of TWINS partners to the questionnaire about tools, methods and challenges related to requirements engineering/management, architecture design and product information management.

The publication is organised as follows: Chapter one summarises the structure of the document. Chapter two presents the definition of software-hardware co-design and introduces the co-design activities. Chapter three introduces the goals of the survey, viewpoints that are covered by the survey and how the survey was carried out. Chapter four presents respondent and organisation profiles and summarises the results for each viewpoint presented in chapter three. Chapter five concludes the survey.

2. Software-hardware co-design

2.1 Definition

There are several definitions for software-hardware co-design in which different aspects are highlighted.

According to Wikipedia [2006] software-hardware co-design is an important approach to ensure an efficient final implementation of the product. That is due to the fact that the interface between hardware and software is a critical part of the overall design when designing systems where both hardware and software aspects are important.

Lockheed Martin [2006] defines co-design as a simultaneous consideration of hardware and software within the design process. It emphasises that it consists of the "co-development and co-verification of hardware and software through the use of simulation and/or emulation". On the other hand, Assimakopoulos [1998] states that communication and interaction between development teams is essential during the design process and co-design benefits from efficient systems engineering throughout the product lifecycle.

Based on the discussion above, the following extended definition of co-design describes how the term is used in this publication:

Co-design is an activity that involves cooperative specification, development, verification and management of complex multidiscipline products.

Therefore, it is a managed approach for collaborative and iterative activities in which the people involved with different product development disciplines work together in an attempt to specify, develop, verify and manage the product.

2.2 Systems engineering and co-design

The development of complicated multi-technology devices needs a discipline which brings together different skills, disciplines, development stages and stakeholders. This discipline, systems engineering, offers a viewpoint for coordinating the development of complex products. The role of systems engineering is different from discipline-specific engineering approaches, such as electronics engineering, software engineering and mechanical engineering [Stevens 1998]. Discipline-specific engineering approaches focus on their own special area providing mechanisms for supporting them. Systems engineering provides a framework for the work of other engineering disciplines and it

remains independent of discipline and product type. The role of systems engineering depends on the ability to communicate across the disparate groups involved in product development as seen in Figure 1 [Stevens 1998]. In large systems, it will be performed at multiple levels throughout the development by all disciplines (i.e. electronics, mechanics, SW) [Stevens 1998].

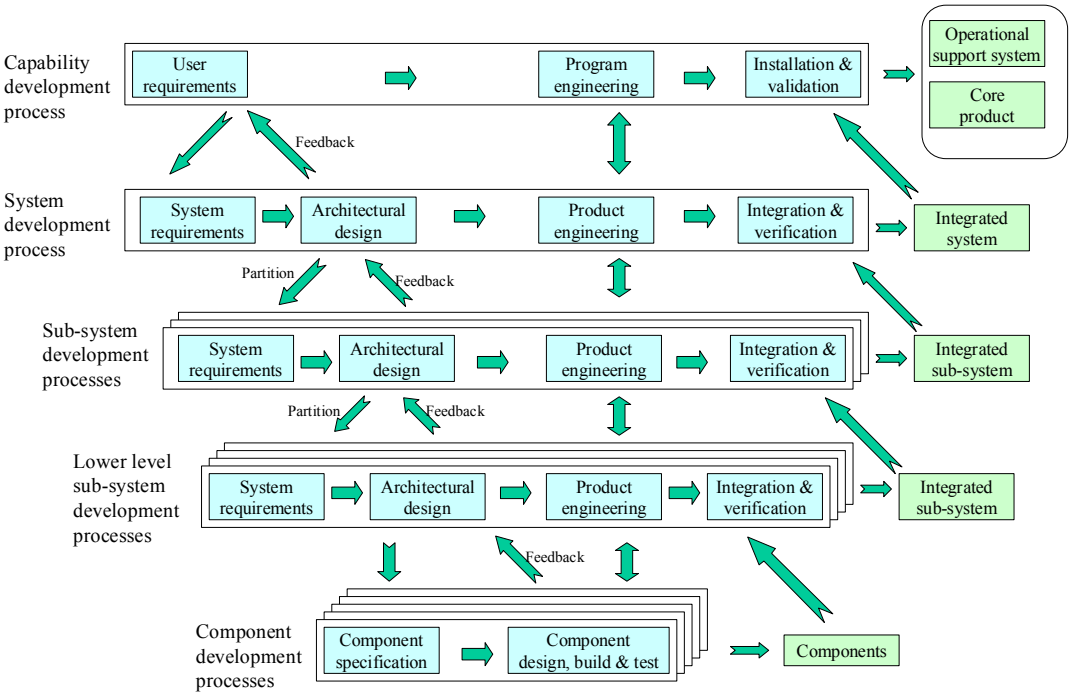

Figure 1. The development process of a multi-technology product.

Ernst [1998] presents the synchronisation and integration of hardware and software designs as a major problem in the design process. He also states that when the level of detail increases, the more time is needed to control consistency and correctness.

2.3 Co-design activities

According to Assimakopoulos [1998], the management of co-design identifies and solves technical and management obstacles by acting as a mediator between different development teams. The role of co-design is to manage complexity, design changes, and the shift of functionality between software and hardware domains [Assimakopoulos 1998]. Assimakopoulos [1998] also states that communication and interaction between development teams is essential during the design process.

Co-design includes:

- Co-specification, where the roles of software and hardware in implementing system functionality are considered and, based on the evaluation, the implementation is assigned to either of the two.
- Co-development, where the software, hardware and interfaces are developed.
- Co-verification to further optimise and refine the SW/HW partitioning, i.e. to aid design space exploration.
- Co-management that covers coordination, project management, requirements management and configuration management throughout system specification, development and verification.

These activities' relationship to the systems engineering phases is illustrated in Figure 2.

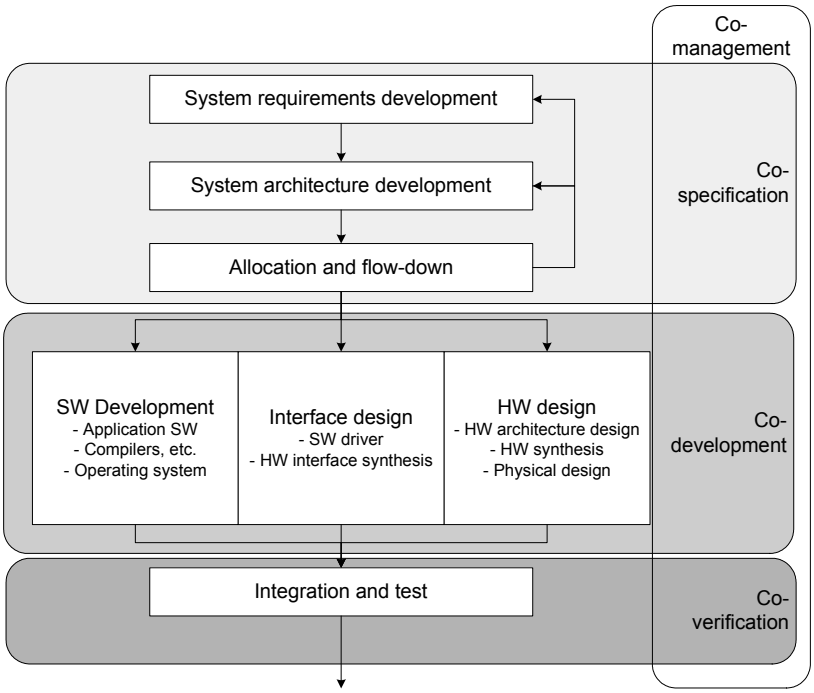

Figure 2. Co-design activities in the overall embedded system design process (adapted from Ernst [1998] and Stevens [1998]).

Next, each of the co-design activities is discussed in more detail.

2.3.1 Co-specification

Co-design starts with the specification of the system, describing its behaviour and requirements. The specification phase is similar to requirements engineering in general: the system functionalities are considered and the system requirements are specified. In co-design context, the specification is most relevant at the system level, where both hardware and software are considered.

The specification phase also includes modelling the system. In a co-design process, the overall system model must describe both hardware and software functionality. Different modelling paradigms can be used to create the models, for example, modelling languages or graphical formalisms (e.g. statecharts, petri nets) [Micheli & Gupta 1997, Gupta 2001, Ha et al. 2006]. The models used in this process are variants of different types of formal models. There is also variation in the types of models that can be used. For example, Micheli and Gupta [1997] describe using separate models for software and hardware, where the modelling language used can already determine much of how the model will be partitioned to the HW/SW parts.

Gupta [2001] describes using structural, functional and dynamic (e.g. data- and control-flow) models, with a set of modelling languages such as SystemC and SpecC for structural models, Discrete Event System Specifications (DEVS) for behavioural models and VHDL and other hardware description languages for hardware descriptions. Ha et al. [2006] describe models for different properties such as data- and control-flow and a task interaction model, with their own modelling languages for each model type. In all cases, the models must describe the system at sufficient level to enable the simulation and verification of the system from the models.

The models are used as input for software/hardware partitioning, which is also called design space exploration [Gupta 2001] and allocation. In this activity, the partitioning of the system functionality is divided among software and hardware components. The design space is the set of different configurations of hardware and software partitions for the models. Exploring this space is the process of optimising the partitioning of the model implementation for software and hardware with regards to given criterion. For example, large parts of the functionality can be implemented in a software component in the first release, which can be replaced by a hardware implementation in a later release to improve performance [Micheli & Gupta 1997]. This is an iterative process, where the models and partitioning are repeated, optimised, repartitioned and re-evaluated until the design goals are met. This process is considered crucial as once the partitioning is verified, the actual implementation phases are started and after this point changes are more difficult and expensive [Gupta 2001].

The co-design process can also consider architecture specification as a separate activity [Ha et al. 2006]. In this case, the possible (available) hardware components are considered and, for example, a hardware platform or a set of components is chosen as a basis for the possible architectures to consider. Similarly, the availability and any constraints set by the software components that can or must be used need to be considered, such as in an evolving product family with maximum reuse as a goal.

2.3.2 Co-development

Co-development involves the development of the software and hardware and their interfaces. Software and hardware development can be undertaken using their own development processes, and an important part of this activity is the interface development.

2.3.3 Co-verification

Co-verification makes use of co-simulation, where software and hardware are executed in parallel. Simulation is used by designers to simulate the model by executing test case scenarios and collecting data from model execution. The output from the model simulation is used to further optimise and refine the SW/HW partitioning, i.e. to aid in design space exploration. Thus the process is iterative and repeated to optimise the development of the models.

Different simulation engines can be used depending on the modelling languages and methods used. Some example simulation environments include the DEVS and PTOLEMY both of which use Java as a modelling language. VHDL is a modelling language used by many commercial simulators. An example of a domain-specific solution is the PeaCE co-design environment which aims to support the whole co-design process including the simulation [Ha et al. 2006]. PeaCE is targeted at multimedia applications with real-time constraints [Ha et al. 2006].

There are also formal models that have been used for co-verification and/or co-design, for example, CFSM and IPN.

- Co-design FSM (CFSM) [Chiodo et al. 1995, Balarin et al. 1997] is a formal model used in the POLIS co-design tool [Balarin et al. 1997]. Co-verification is performed by translating CFSM into traditional FSM and existing FSM-based verification techniques are applied.
- Interpreted Petri Nets (IPN) can be used for synthesising interfaces in [Vial & Rouzeyre 1997].

2.3.4 Co-management

Co-management covers coordination, project management, requirements management and configuration management throughout system specification, development and verification.

3. ITEA-TWINS survey

3.1 Goals

The survey was intended primarily for TWINS industrial partners. Research partners were to answer the questions when applicable.

The goal of the TWINS survey was to obtain an overview of the tools and methods used for:

- Requirements engineering (including development and management)
- Architecture design
- Product information management (PLM/PDM, CM).

The survey also collected information about the challenges encountered relating to the above-mentioned issues.

3.2 Viewpoints

3.2.1 Requirements engineering

Requirements engineering (RE) is a set of activities that cover discovering, analysing, documenting, validating and maintaining a set of requirements for a system. Requirements engineering is often divided into requirements development and requirements management. Requirements engineering is generally accepted to be the most critical and complex process within the development of systems. The main reason for this is that the requirements engineering process has the most dominant impact on the capabilities of the resulting product. Furthermore, requirements engineering is the process in which the most diverse set of product demands from the most diverse set of stakeholders is considered. These two reasons make requirements engineering complex as well as critical.

The main high level activities included in the requirements engineering process are:

1. System requirements development, including requirements gathering/ elicitation from various sources, requirements analysis, negotiation, priorisation and agreement of raw requirements, and system requirements documentation and validation.

2. Requirements allocation and flow-down, including allocating the captured requirements to system components and defining, documenting and validating detailed system requirements.
3. Software requirements development, including analysing, modelling and validating both the functional and quality aspects of a software system, and defining, documenting and validating the contents of software subsystems.
4. Continuous activities, including requirements documentation, requirements validation and verification, and requirements management. Requirements management controls and tracks the changes of agreed requirements, the relationships between requirements, and dependencies between the requirements documents and other documents produced during the systems and software engineering process.

The relationship of the requirements engineering activities to the co-design activities (see Figure 2), namely co-specification, co-development, co-verification and co-management, is as follows: The system requirements development, allocation and flow-down, and software requirements development are all related to co-specification, and requirements management is related to co-management. Continuous activities documentation is related to co-specification and validation and verification to co-verification.

3.2.2 Architectural design

A number of architectural design methods and techniques have been developed and documented over the last ten years. Architectural design methods and techniques for different domains emphasise different goals, rise to different challenges and show domain-specific characteristics. Architectural design includes the following three fundamental activity types: architectural analysis, architectural synthesis and architectural evaluation [Hofmeister et al. 2007]. These activities are executed repeatedly, at multiple levels of the granularity of a system as shown in Figure 1, in no predictable sequence.

Architecting is best understood in a lifecycle context because architecting contributes to every phase of the product's whole lifecycle from the system's initial conceptualisation to deployment, operation and retirement from use. The IEEE standard 1471-2000 [IEEE Std 1471-2000] specifies normative elements for architectural descriptions. An architecture description must identify architectural stakeholders and stakeholder concerns and the concerns are required to be addressed within the chosen architectural viewpoints and must be mapped to at least one of the provided architectural views. The standard defines view as "a representation of a whole system from the perspective of

a related set of concerns" and viewpoint as "a specification of the conventions for constructing and using a view". Each view should include a representation of the system with the languages, modelling techniques, notations, tools or analytical techniques of the associated viewpoint. Views and lifecycle models should be useful for the system at hand, and, therefore, approaches with a fixed set of views or a fixed lifecycle model conflict with the standard and a recent trend.

3.2.3 Product information management

The ability to produce quality products on time and at competitive costs is important for any industrial organisation. Nowadays, companies are seeking systematic and more efficient ways to meet these challenges. The modern approaches for product development need to take into account the business environment, and the product's whole lifecycle must be covered, from the initial definition up to maintenance. Such a holistic viewpoint means the efficient deployment of lifecycle management. Setting up a comprehensive, smoothly running and efficiently managed product development environment requires effective lifecycle processes and tool support. From the product information management point of view, this survey covered topics of Configuration Management (CM) as well as Product Data Management (PDM) and Product Lifecycle Management (PLM).

The discipline that keeps the evolving product under control is called Configuration Management (CM). It is a well-known concept in software engineering and it has been widely discussed in literature and articles. The roots of CM are in the defence industry environment as a discipline for resolving problems with poor product quality, parts ordering, and parts not fitting, which were leading to high cost overruns [Berlack 1992]. At the beginning, the focus was on the CM of hardware-oriented products. The need for the management of software artefacts became topical as the software engineering industry emerged. According to Estublier et al. [2005] software CM (SCM) emerged as a separate discipline in the 1970s, with the advent of tools such as SCCS, RCS and Make. The traditional products that were composed based on mechanical and electronics components included more and more software. Nowadays, software development as an engineering activity is also more complex. It ties up more developers from different cultural backgrounds, as globalisation removes national borders. Furthermore, the news that a product is bad and has a bug can spread very fast (e.g. newsgroups), which forces a company to provide the fixes and patches very quickly to save face and prevent its market share from dropping [Leon 2000].

PDM controls the products-related data and processes during the entire lifecycle of a product. Traditionally, the functionality of PDM systems is divided into user functions and

utility functions [Crnkovic et al. 2003]. User functions consist of data vault and document management, workflow and process management, product structure management, classification management, and program management. Utility functions consist of communication and notification, data transport and translation, image services, administration, and application integration. PDM systems are one of the most important components in PLM. Stark [2006] defines PLM as a business activity that manages the products during their lifecycles from early ideas until they are retired and disposed of – "from cradle to crave". Stackpole [2003] states, "*PLM is not so much a system as a strategy for integrating and sharing information about products between applications and among different constituencies such as engineering, purchasing, manufacturing, marketing, sales and aftermarket support*". Typically, PLM requires that separate databases are integrated to get the right information out of it for people working in different business functions of the company. PLM can be seen as a unifying framework for the definition of consistent and accurate product information and process management.

3.3 The set up

The survey was carried out by VTT Technical Research Centre of Finland. The survey was implemented as a web survey to enable the access and responses using web browsers. ZEF, that is web-based feedback collecting and analysis tool (see Figure 3), was used for the technical implementation of the survey.

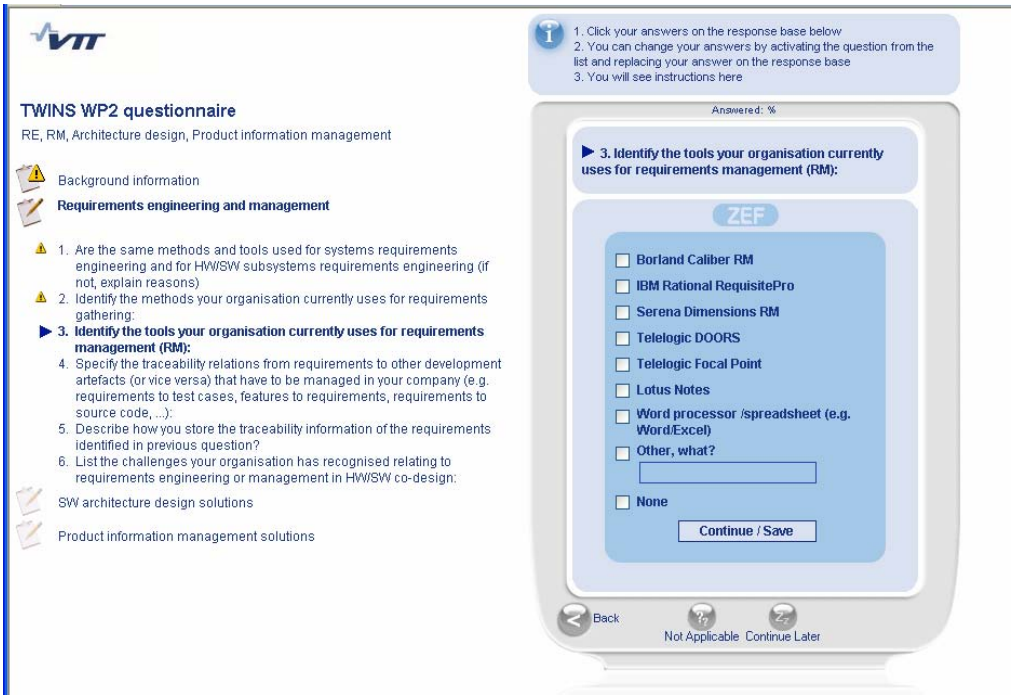

Figure 3. User interface of the ZEF-tool.

The survey was released on 25th of May 2007 to the TWINS consortium and it was closed on 11th of June 2007 after being online for two weeks. Altogether, the number of respondents totalled 17 persons and they represented 12 organisations out of 23 potential companies. Therefore, 52% of potential companies participated in the survey. The countries and number of organisations are presented in Figure 4.

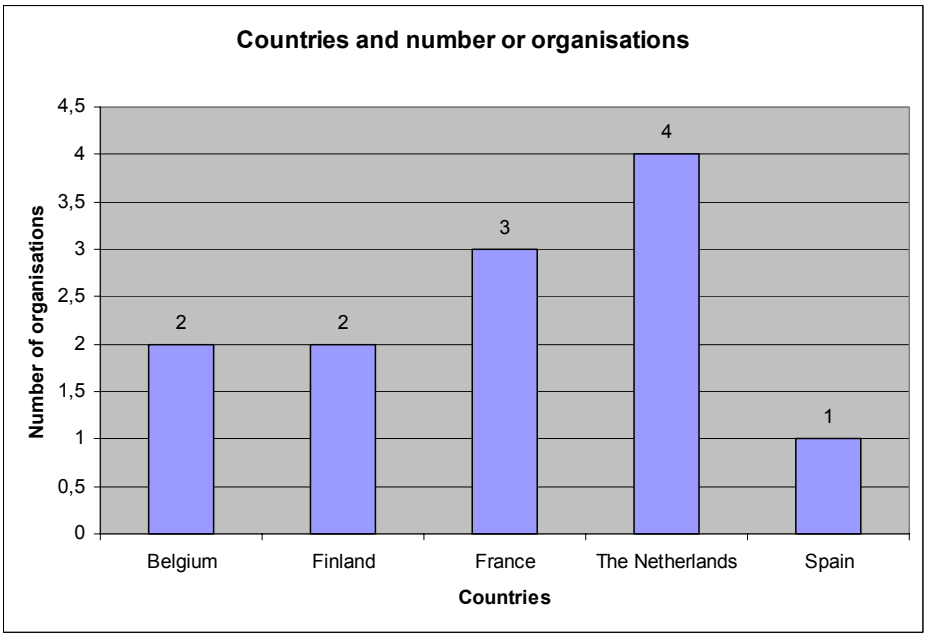

Figure 4. Countries and number of organisations.

Respondents have given their responses only to the questions they find relevant in their position and organisation. Therefore, the number of responses varies between the questions.

The survey was divided into four parts (see Appendix A) with several questions in them. Part A had 7 questions, Part B 6 questions, Part C 6 questions, and Part D 6 questions. Therefore, the total amount of questions in this survey was 25 questions. Figure 5 presents the different question types and the number of questions in the category.

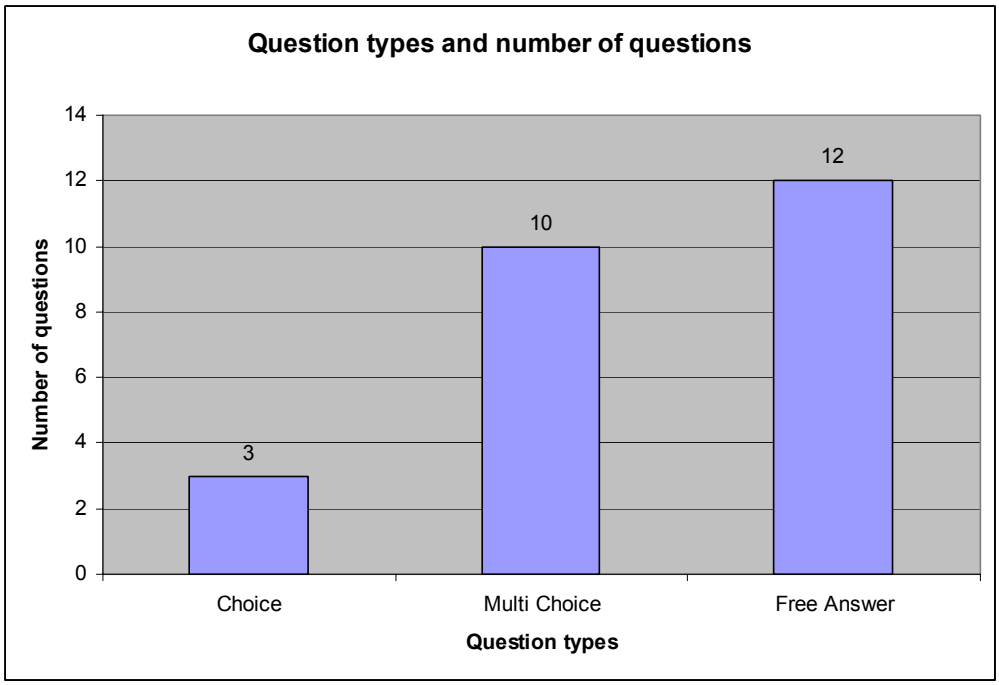

Figure 5. Question types and the number of questions.

4. Results

The questionnaire was sent to 23 organisations participating in the TWINS project. 12 organisations out of 23 answered the questionnaire so the response rate was 52%. The total number of individual respondents was 17 and there were five organisations with more than one respondent.

4.1 Respondent and organisation profile

4.1.1 Respondent's position

Most of the responses (see Figure 6) were provided by Designers (four respondents), Project managers (three respondents) and Chief architects or Architects (two respondents). The total number of respondents in the Other category was six. These responses were given by positions like Development Manager, Method Developer, CTO, Consultant, PhD Student, and innovation centre.

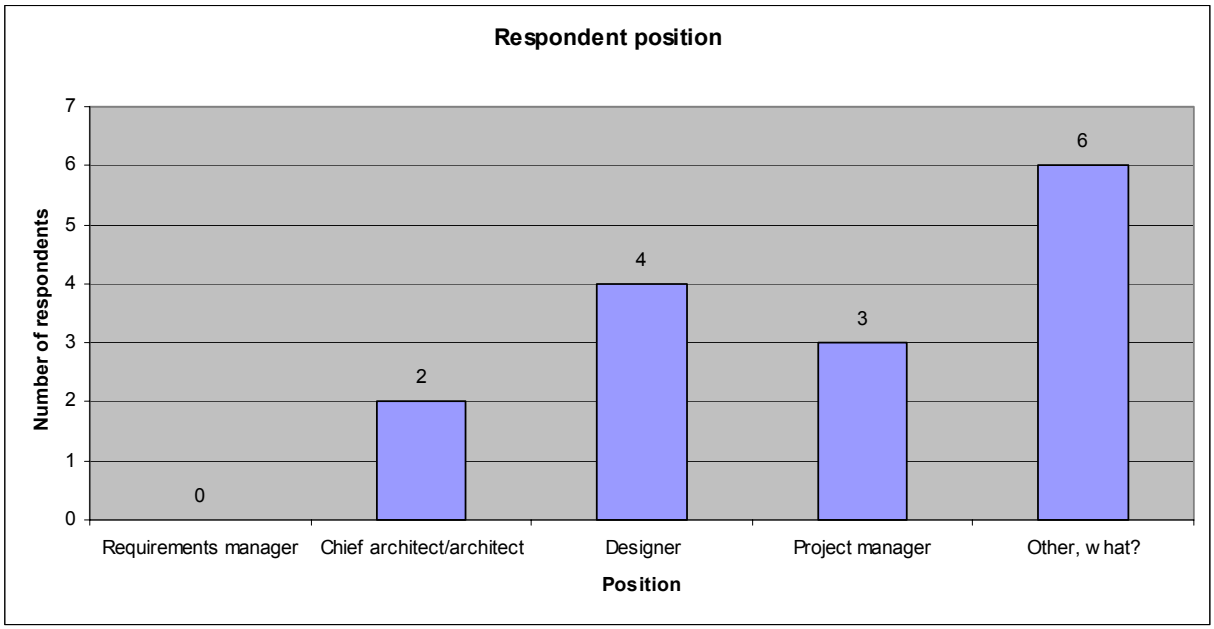

Figure 6. Respondent's position in organisation.

4.1.2 Size of the organisation

Most of the organisations (9 organisations out of 12) belong to the category of large organisation (see Figure 7), which means that the number of personnel in the

organisation exceeds 250 people. There was one organisation in each of the following categories:

- one organisation with less than 10 people
- one organisation with personnel varying from 10 to 49 people
- one organisation in which the number of personnel range from 50 to 250 people.

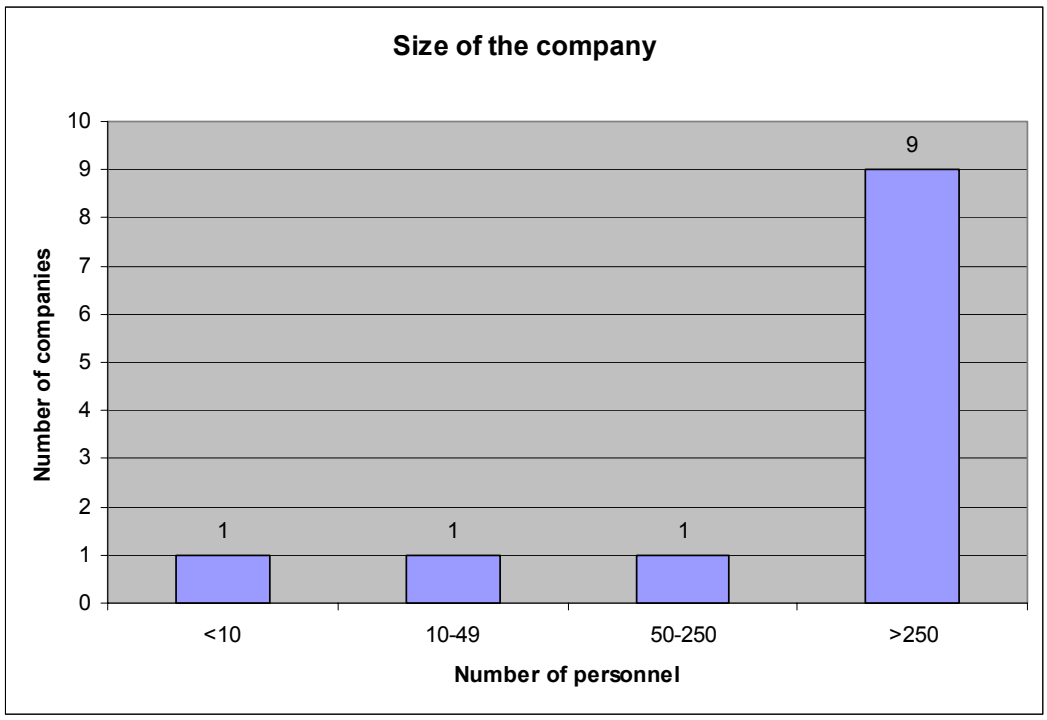

Figure 7. Size of the organisation.

4.1.3 Product life time

According to the respondents, the life time of the products (Figure 8) produced by the partners varied from 4 years to 15 years. The average value for the life time was 9 years.

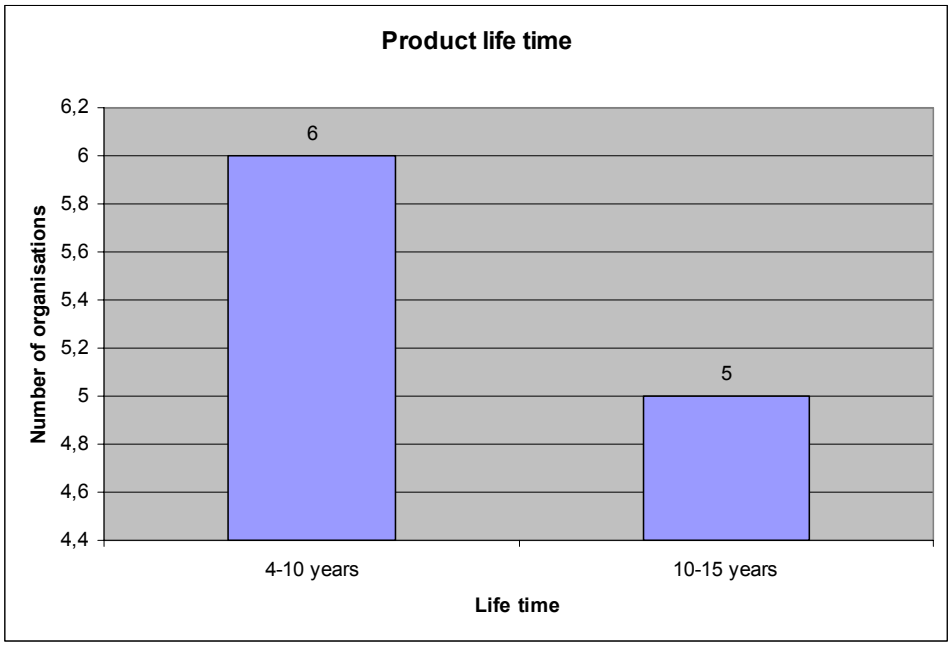

Figure 8. Product life time.

4.1.4 Collaboration mode and frequency

Many of the companies are using several collaboration modes (see Figure 9). One organisation advised they are using four different collaboration modes in their product development. Three organisations advised that they are using three different collaboration modes. Two collaboration modes were mentioned by four organisations.

The most popular collaboration type among the Twins consortium was "Customer supplier relationship". This collaboration type was mentioned by nine organisations. The second most common type was "Joint research and development partnership", which was mentioned by eight organisations. Third came "Multisite development", mentioned by four companies. The fourth common collaboration mode was "Technology exchange agreement or licensing" totalling three references. Only one organisation advised that they are not involved in any collaboration at all.

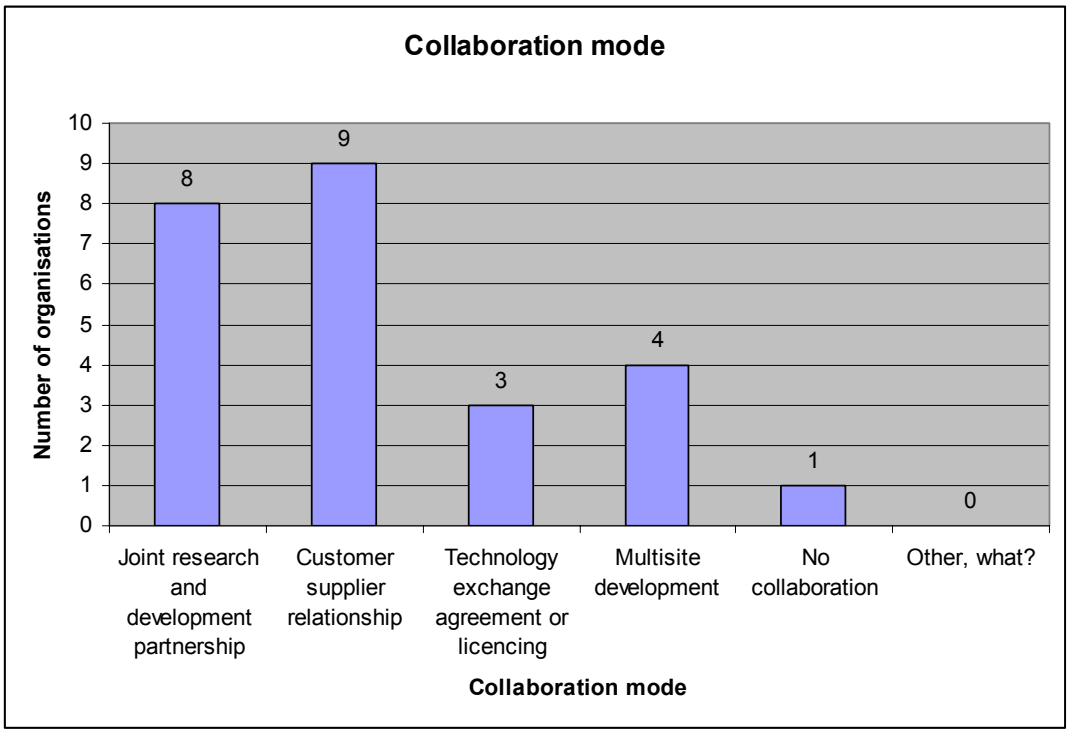

Figure 9. Collaboration modes.

Most of the companies estimated that from 10% up to 20% of their products are produced in collaboration (see Figure 10). Only one organisation is not developing its products in collaboration and one organisation develops all of its products through collaboration.

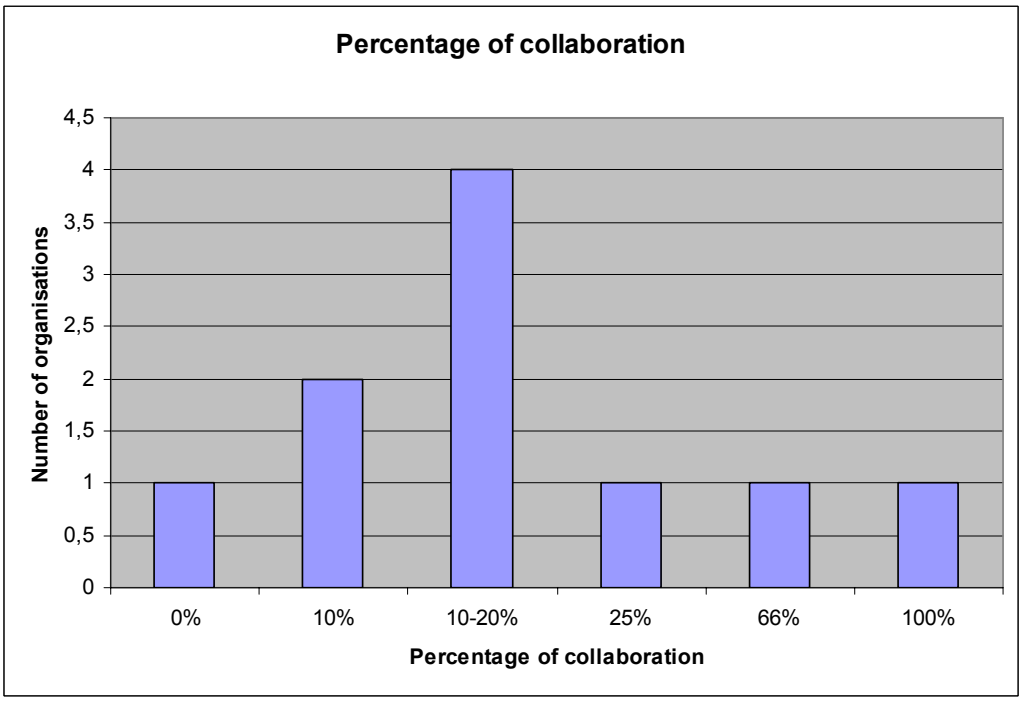

Figure 10. Collaboration frequency.

4.2 Requirements engineering and management

4.2.1 Number of requirements

Estimations on the number of requirements in typical projects in each organisation varied from 30 requirements all the way up to 3,000 requirements. The most common estimation about the number of requirements in a typical project was somewhere between 200 and 500 requirements. There were two companies that mentioned they are having typically over 1,000 requirements in their projects. One organisation also mentioned that the number of requirements is product-specific. (See Figure 11.)

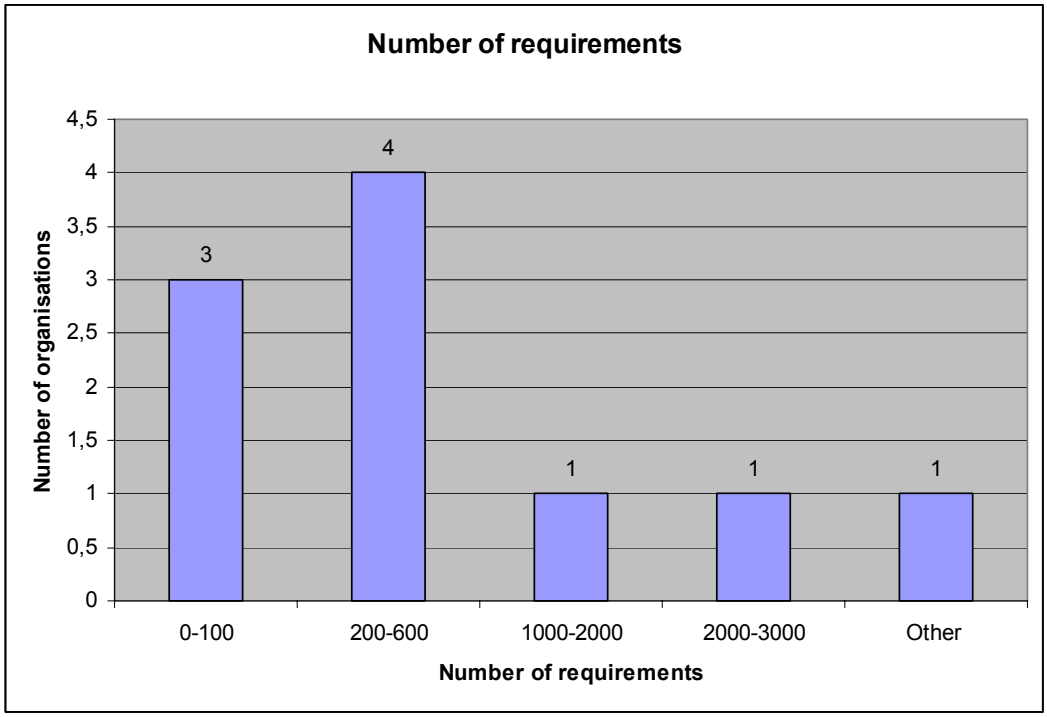

Figure 11. Number of requirements.

4.2.2 Requirements engineering tools and methods in system and HW/SW subsystem

Most of the organisations (six organisations out of nine) are using same methods and tools for systems requirements engineering and for HW/SW subsystems requirements engineering (see Figure 12). Three organisations are using different systems for systems requirements engineering and for HW/SW subsystems requirements engineering. The reasons for using different tools and methods were related to the following issues:

- For sys. req. eng. the DOORS tool is used. Due to less complexity and discipline-specificity at the SW/HW level, the "MS Word method" is used.
- Every discipline uses its own way of requirements specification.
- System engineering covers only system, component and SW. Requirements are only applied at the HW component level and not at the HW refined level.

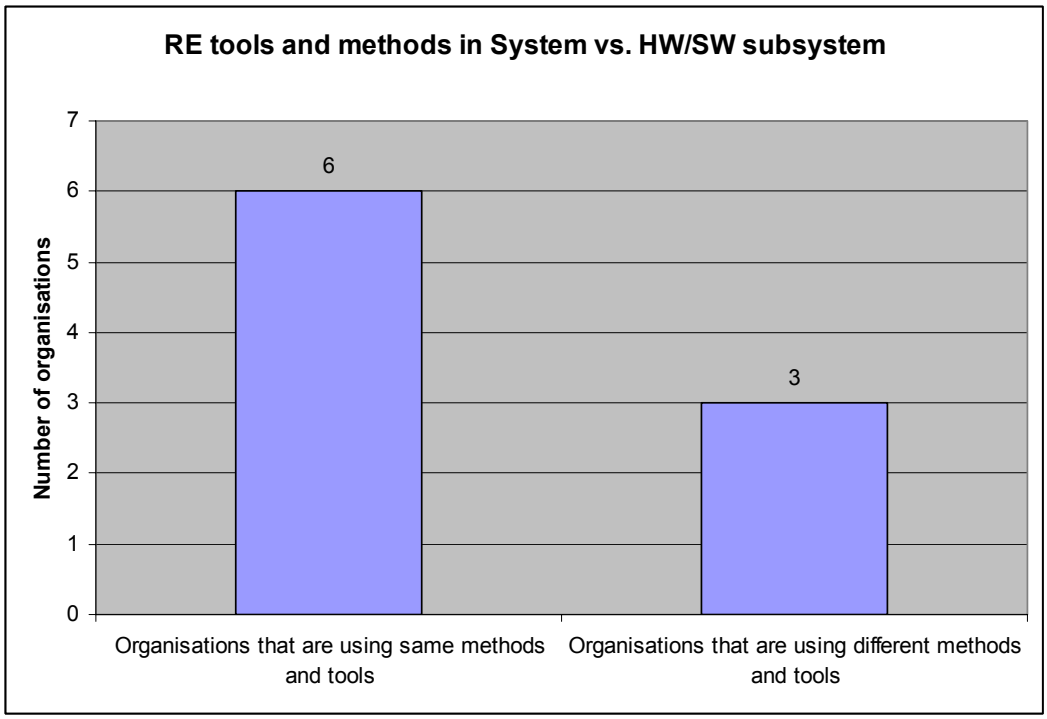

Figure 12. System and HW/SW subsystem RE tools and methods.

4.2.3 Requirements gathering methods

Every TWINS partner organisation is using "General techniques" like brainstorming, workshops, facilitated meetings, interviews, and use cases for requirements gathering (see Figure 13). The second most common techniques are "Structured methods" together with the "Agile technologies". Three companies out of nine are using both techniques. "Structured methods" cover methods like SADT (Structured Analysis and Design Technique) and SASS (Structured Analysis and System Specification). Methods like DSDM, XP, and SCRUM belong to the "Agile methods" category. "Object oriented methods" like OMT (Object Modeling Technique) and Shlaer-Mellor Object-Oriented Analysis Method are used by two companies out of the nine. "Dedicated SW RE methods" and "Viewpoint oriented methods" are used by one organisation. Other techniques the respondents mentioned once were "Prototyping" and "Matlab Simulink".

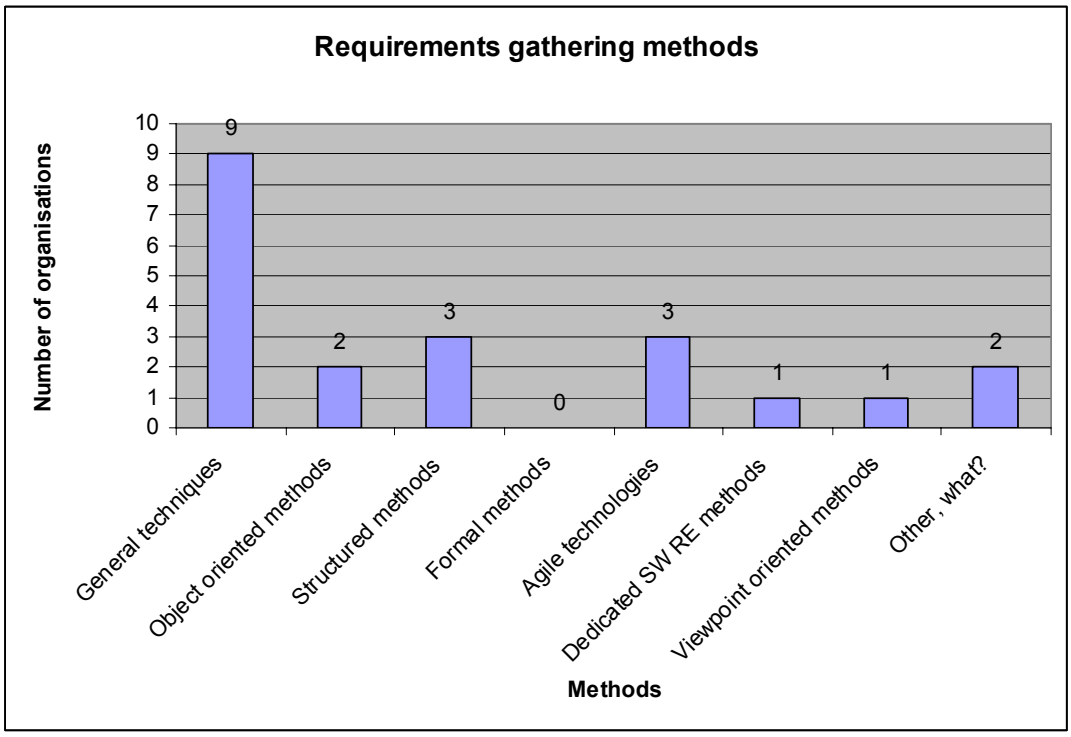

Figure 13. Requirements gathering methods.

4.2.4 Requirements management tools

Among TWINS partners, the most common tools used for requirements management are "Word processor/spreadsheet" tools (see Figure 14). Eight organisations out of nine use these tools for requirements management. The second most common tool was "Telelogic DOORS" that four organisations are using for requirements management. Two companies uses "Telelogic Focal Point" for requirements management and this makes it the third most common tool among the TWINS partners. "IBM Rational RequisitePro" and "Lotus Notes" is used by one organisation. In the "other" category, "proprietary tools" are mentioned as being used for requirements management.

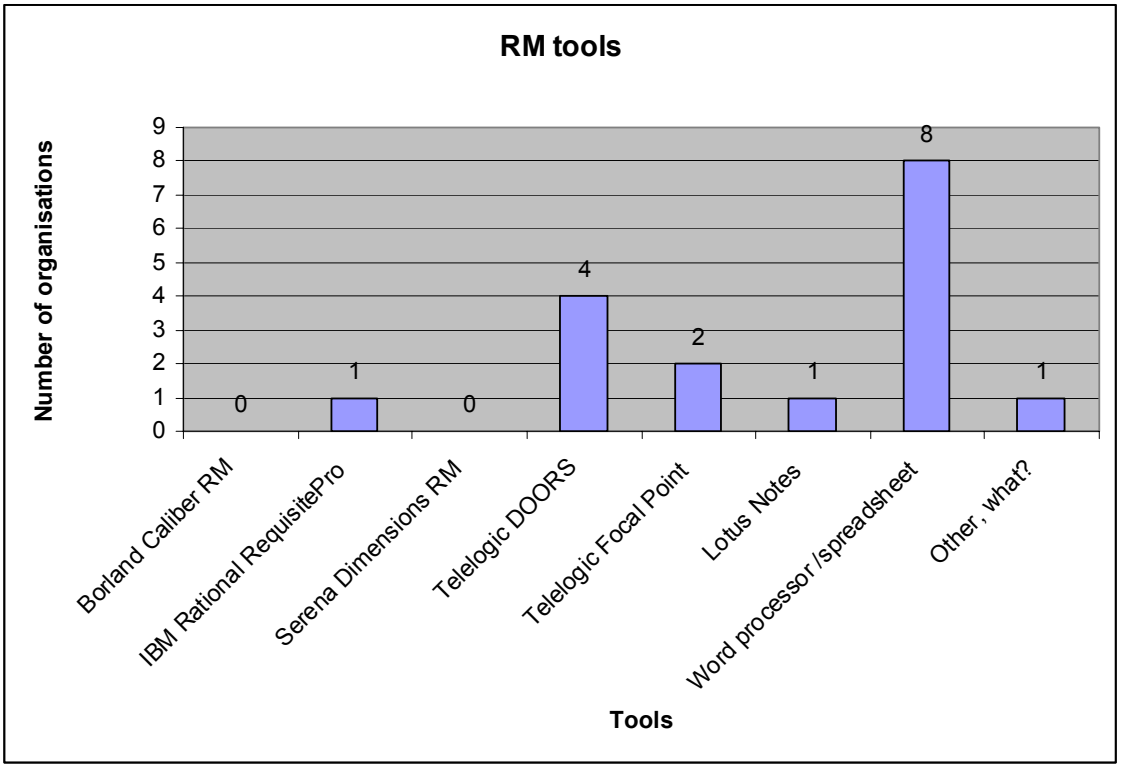

Figure 14. Requirements management tools.

4.2.5 Requirements traceablity

In this survey, respondents were asked to specify the traceability relations from requirements to development artefacts (or vice versa) that have to be managed in their organisation. The most common traceability (see Figure 15) that was mentioned was from requirements to tests (test cases, test results) (6 companies specified this option). The second most common was the traceability between requirements and specifications (5 companies specified this option). Three companies maintained traceability between requirements and design (models).

Other traceability information used by individual companies included traceability between requirements and bug reports, requirements and features, requirements and marketing documents, requirements and code and different requirements levels. Also, one organisation advised that it is not using any traceability information.

The most common way to store traceability information was a combination of partly manual and partly tool supported management of the links (4 companies). Two companies had full tool support in traceability and two were fully manual.

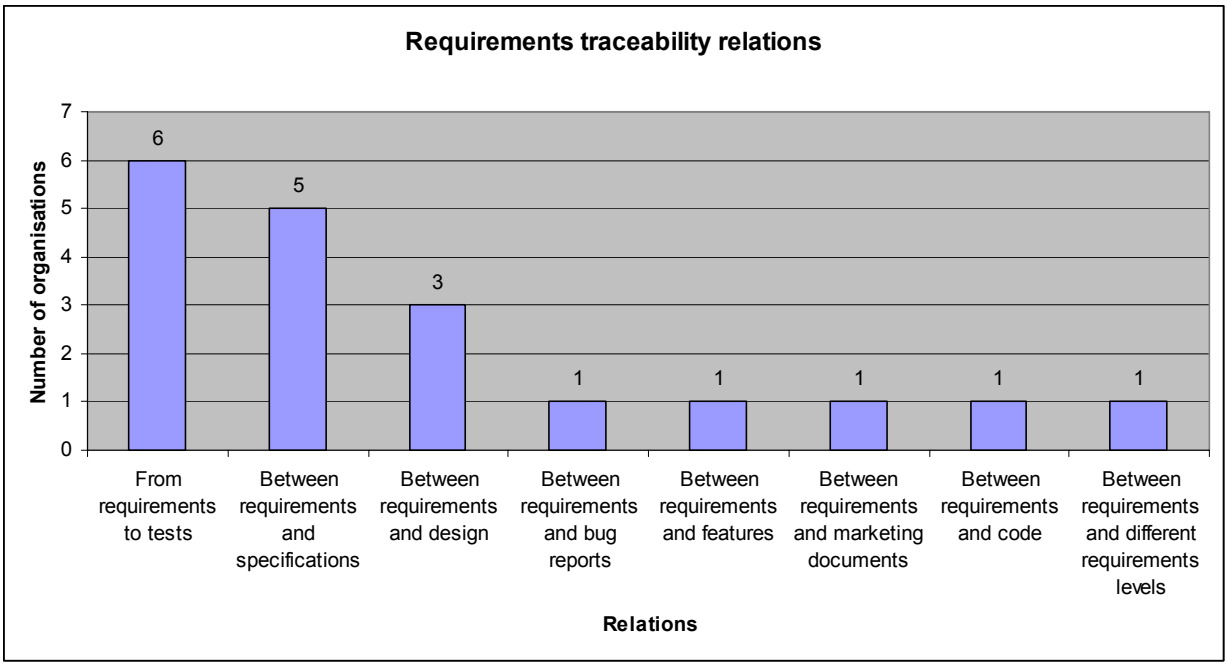

Figure 15. Requirements traceability relations.

4.2.6 Challenges in requirements engineering and management

Respondents were asked to list the challenges their organisations have recognised relating to requirements engineering or management in HW/SW co-design.

The recognised challenges varied quite considerably between the companies, however, the most frequently mentioned challenges were related to traceability, refining requirements and early verification. To follow is the list of the identified challenges:

Traceability:

- Traceability to code, bug reports (automated, e.g. by using Subversion).
- Traceability of software requirements to HS/SW design.
- Full traceability up to HW.

Refining requirements:

- Refinement of requirement after HW-SW partitioning decision.
- Refine System Requirements to HW/FW/SW requirements and design documents.
- Deriving discipline-specific requirements from general product RE.

Early verification:

- Early verification of choices.
- The requirements check if they are full filled must be as soon as possible in the project not on the work floor.

Other:

- Executable specifications are highly desired, but hardly present.
- The division of development needs and requirements between HW and SW.
- Better documenting the rationale behind design choices.
- Proper hardware/software modelling.
- Hardware/software design interaction.
- Hardware/software DFT implementation.
- Exploitation system prognostics.
- Improve the link between MKT and technical teams. – Better needs expression/definition. – Optimize the design process going faster to the right solution. Avoiding loops due to misunderstanding. – Inside technical teams generalise requirements engineering use.
- Trade off methods and criterion for requirement engineering towards HW-SW partitioning.

4.3 Architectural design

4.3.1 Lifecycle models of architectural descriptions

As a relevant lifecycle model for architectural descriptions, the "Architecture of single systems" and "Product line architecture" were the most commonly mentioned (see Figure 16). Six organisations out of 10 selected these options as relevant for their organisation. The third most common lifecycle was "Iterative architecture for evolutionary systems" that was mentioned by three companies. One organisation found the "Architecture is created from existing systems through reverse-engineering" option as relevant for their organisation.

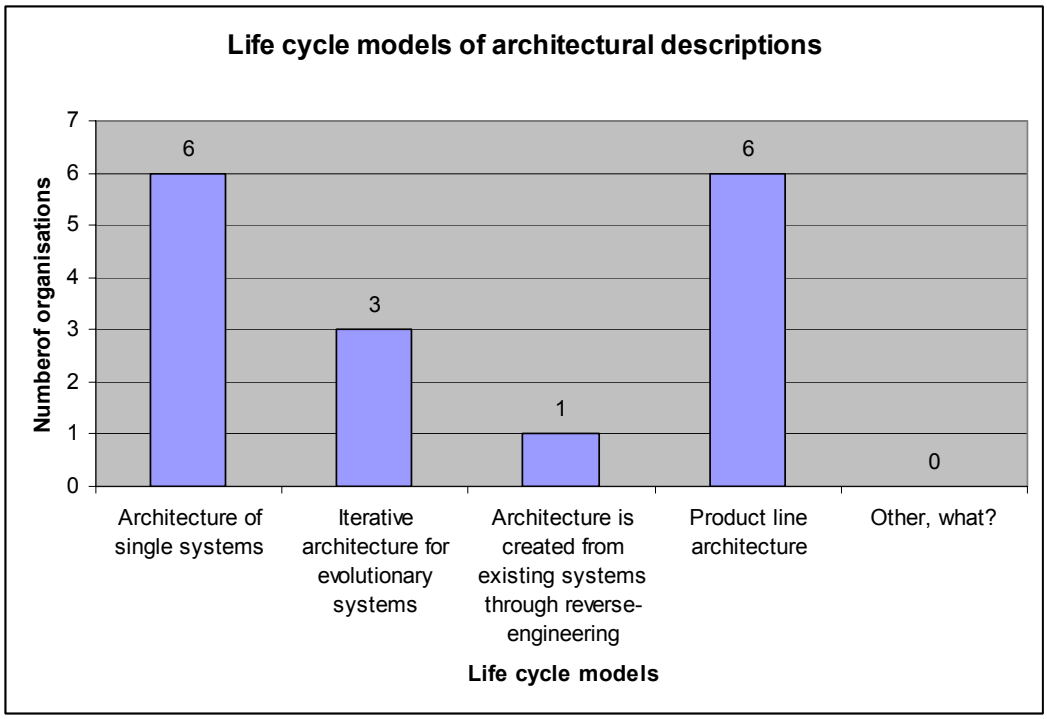

Figure 16. Lifecycle models of architectural descriptions.

4.3.2 Viewpoints in architectural descriptions

The respondents had to select the viewpoints that are used in architectural desciptions in their organisation (see Figure 17). The most commonly used viewpoint in architectural descriptions was "Structural", which was selected by nine out of 10 organisations. Seven organisation mentioned "Physical interconnections" as a viewpoint they use in architectural descriptions. The third most common viewpoint was "Behavioural", which was mentioned by six companies out of the 10.

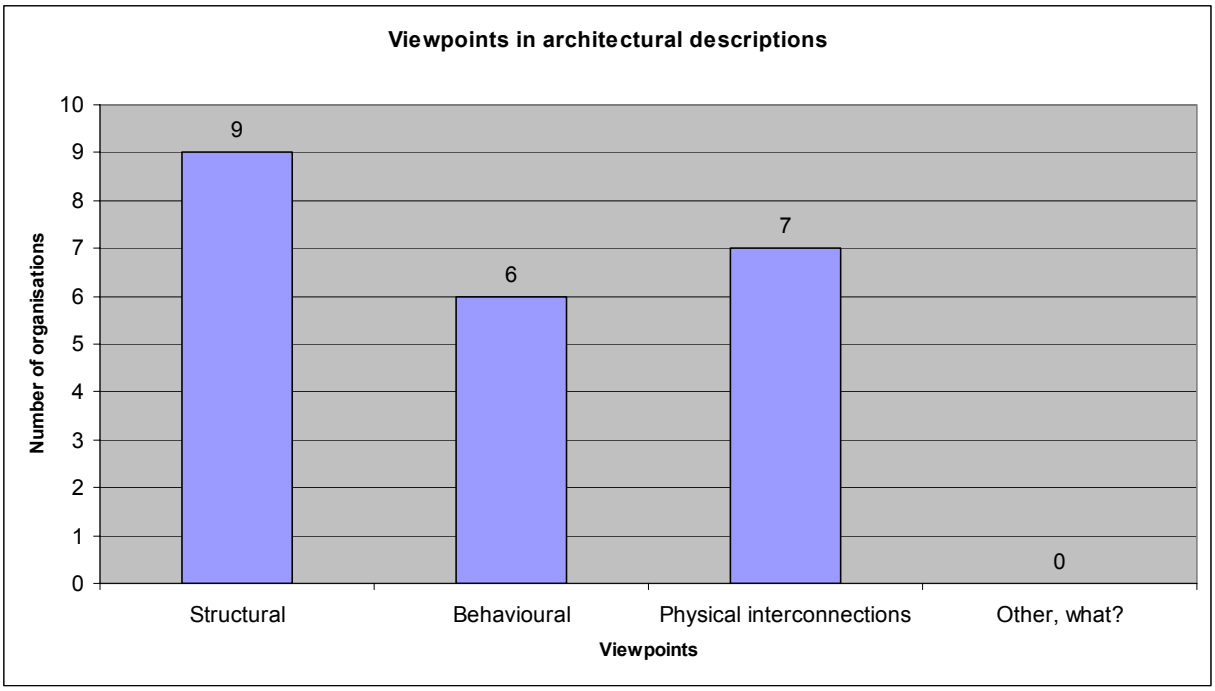

Figure 17. Viewpoints in architectural descriptions.

4.3.3 Architecture design methods and techniques

The diversity of organisations, applications, design approaches, languages, tools and environments resulted in the large variety of methods and techniques currently used for architectural design in the organisations (see Figure 18). Two organisations had no named method or technique for architectural design. Two organisations had a named method for architectural design. One organisation used Yourdon's functional decomposition method and another organisation the RUP method. Two organisations used development tool environments that support the model-driven development approach for system engineering and software development. One organisation organised architectural design elements into Kruchten's 4+1 architectural views. Another organisation used the logical and physical views approach. One organisation had selected the UML language for architectural modelling as an approach without any specific method.

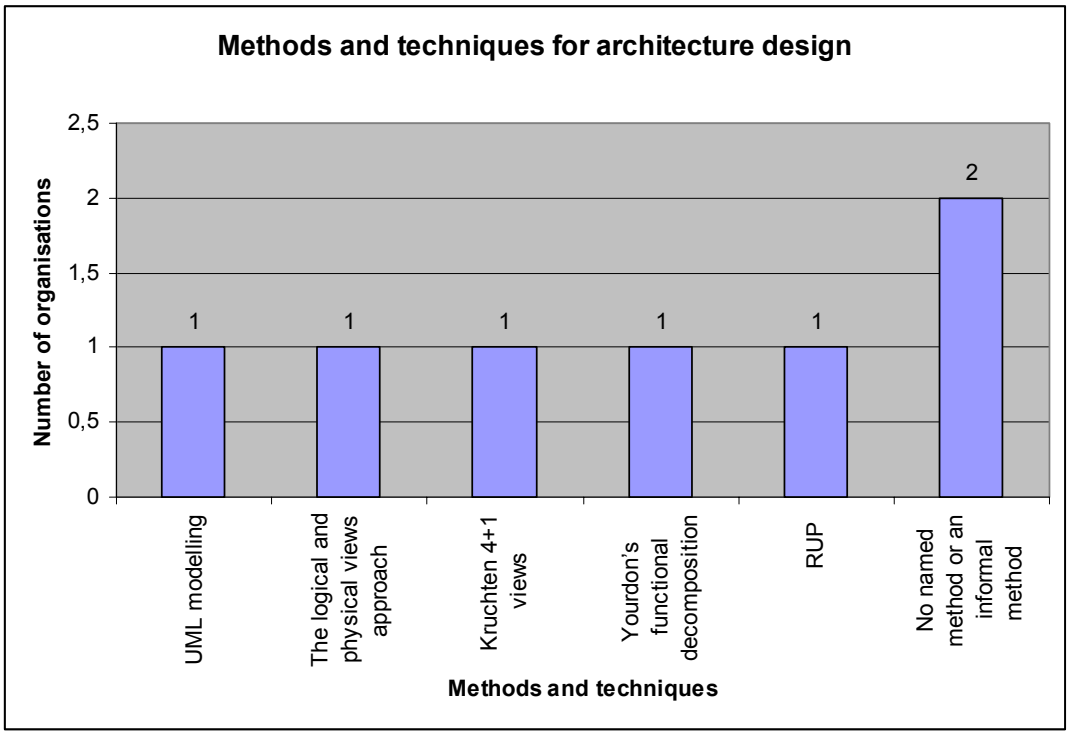

Figure 18. Architecture design methods and techniques.

4.3.4 Notations and models in architectural desciption

The respondents were asked to list the notations and models that are used in architectural descriptions in their companies. According to the answers, UML notations were used by six organisations, so that three of the six organisations widely used UML diagrams and three used UML diagrams for one or two specific purposes, for example, for static architectures or use cases, and other modelling notations or informal models for other purposes. Five organisations used informal modelling sketches and two of them used informal modelling solely. One organisation used Yourdon's functional models. (See Figure 19.)

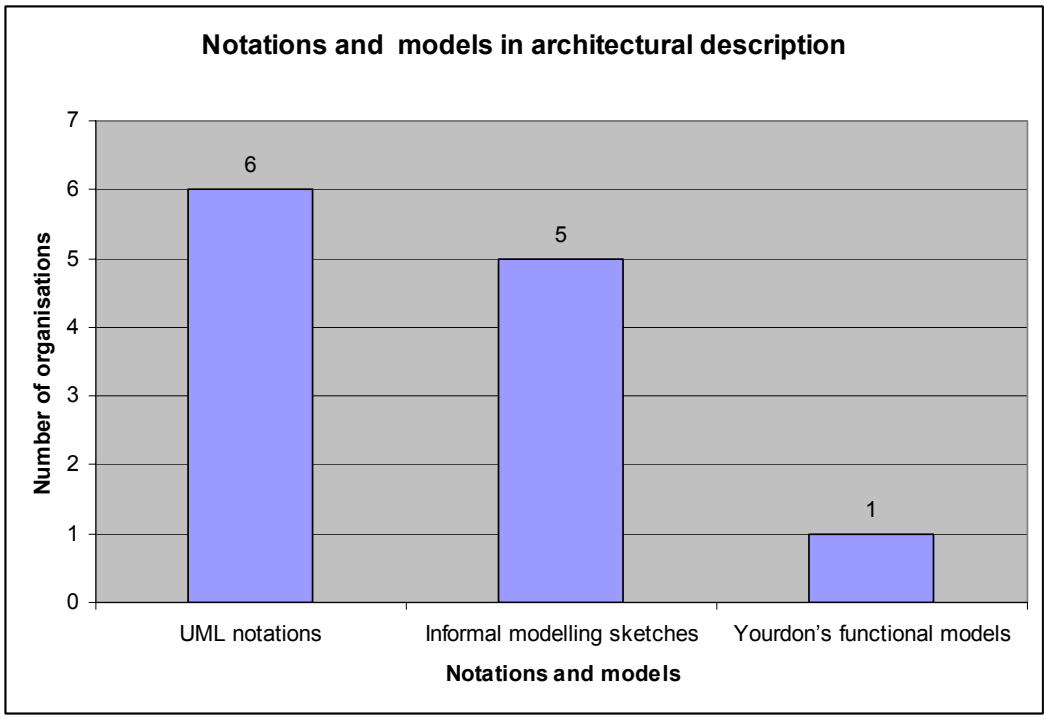

Figure 19. Notations and models in architectural description.

4.3.5 Architecture design tools

In general, it can be observed that, according to the results of the survey, some organisations are using several tools for architecture design at the same time (see Figure 20). Five organisations out of nine mentioned MS Visio as the tool they are using for architecture design. This made MS Visio the most commonly used tool among TWINS partners. In second place, there were three tools: MS Power Point, MS Word and Rational Rose-RT. Two organisations out of nine were using these for architecture design. The following tools received one mention for each: Enterprise Architect, MS Excel, Eagle, Telelogic Rhapsody, Matlab, Simulink, Autosar, Topcased, and Artisan Studio.

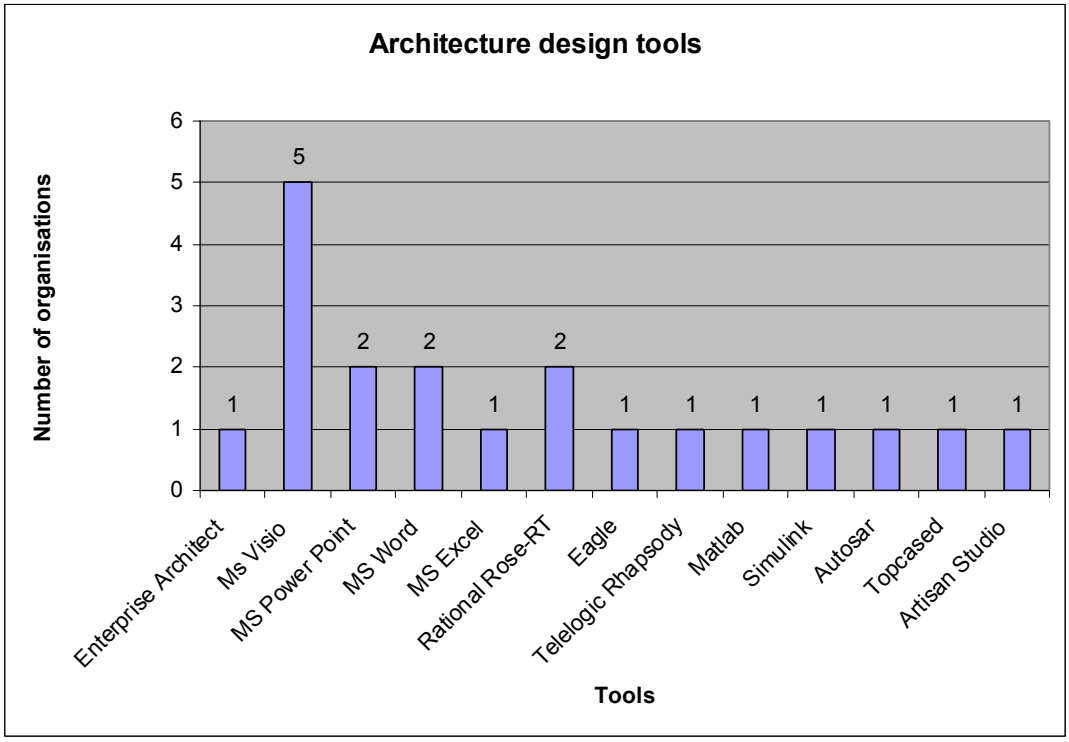

Figure 20. Architecture design tools.

4.3.6 Challenges in architectural design

Respondents were asked to list the challenges their organisation has recognised relating to the architectural design in HW/SW co-design.

The responses received complement each other because every organisation had recognised different architectural challenges in HW/SW co-design. One respondent identified the need for a new architectural co-design method with an open tool framework support for global products and their reliability and cost requirements. One respondent mentioned architectural co-design itself to be a challenging issue in HW/SW co-design. One respondent identified the lack of suitable architectural notations, model hierarchies and repositories. Early feedback on architectural choices via models and prototypes is a challenging architectural issue. Software requirements are needed to be taken into account at early stages of the project and this was considered as an architectural challenge by one respondent. Integration is a long process and one respondent recognised it as an architectural challenge. In addition, proper architectural documentation and documentation maintenance is needed.

Three challenges were related to HW/SW partitioning. One respondent mentioned that better support is needed for decision-making in HW/SW partitioning. Another

respondent mentioned that more discussions and support is needed for delaying HW/SW partitioning as far as possible. One respondent had identified difficulties in dynamical modelling and modelling interactions between SW and HW.

4.4 Product information management

4.4.1 Product data and lifecycle management tools

The respondents were asked to identify the product data management (PDM) or product lifecycle management (PLM) solutions they are using in their organisations.

The most common PDM or PLM solution among TWINS partners belong to the category of Other (see Figure 21). In this category, four organisations mentioned they are using a custom-made solution for PDM or PLM. Baan was also mentioned once in this category. The second most common PDM or PLM solution was SAP/PLM that is used in two organisations out of nine. MatrixOne, Vertex, Windchill, and Aton were mentioned only once. Two organisations mentioned that they are using two different systems. One organisation out of the nine is not using a PDM or PLM solution at all.

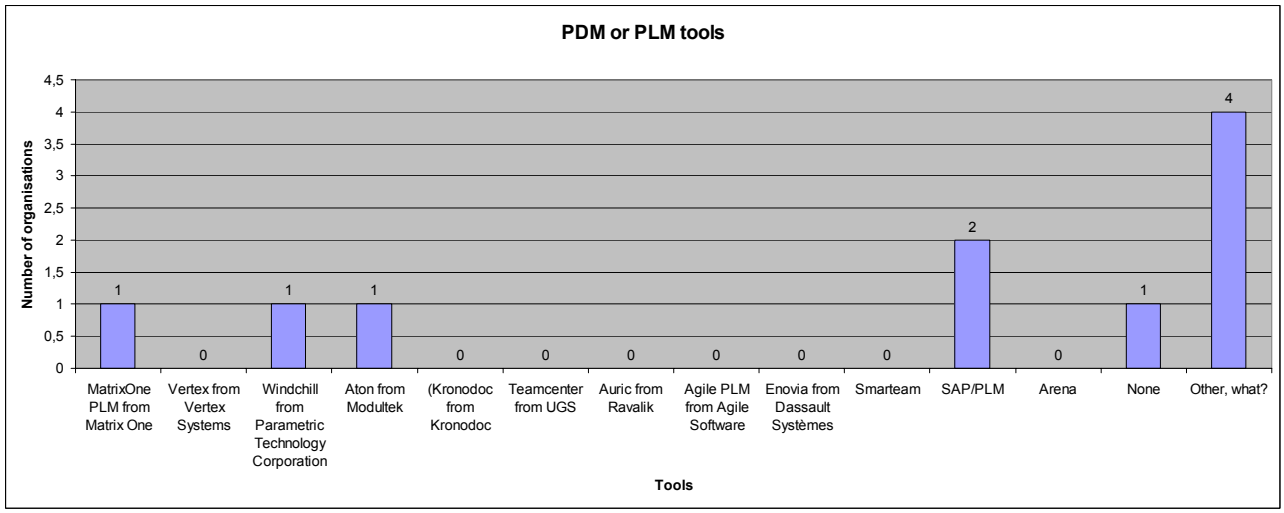

Figure 21. Product data and lifecycle management tools.

4.4.2 Functions of product data and lifecycle management tools

The respondents were asked to identify the functions of the product data and lifecycle management solution that their organisations are using.

Organisations most commonly are using the Data and document management functionality of their PDM or PLM tool (see Figure 22). Nine organisations out of ten mentioned this set of functionality in their response. Part and configuration management was mentioned by six organisations, which made it the second most used set of functionalities. Both Process and workflow management and Reporting got responses from five organisations. Four organisations mentioned that they use the Program and project management functionality of their PDM or PLM tool. Collaboration management was used by three organisations and Tool Integrations by two organisations. One organisation used the tools only for Data and document management.

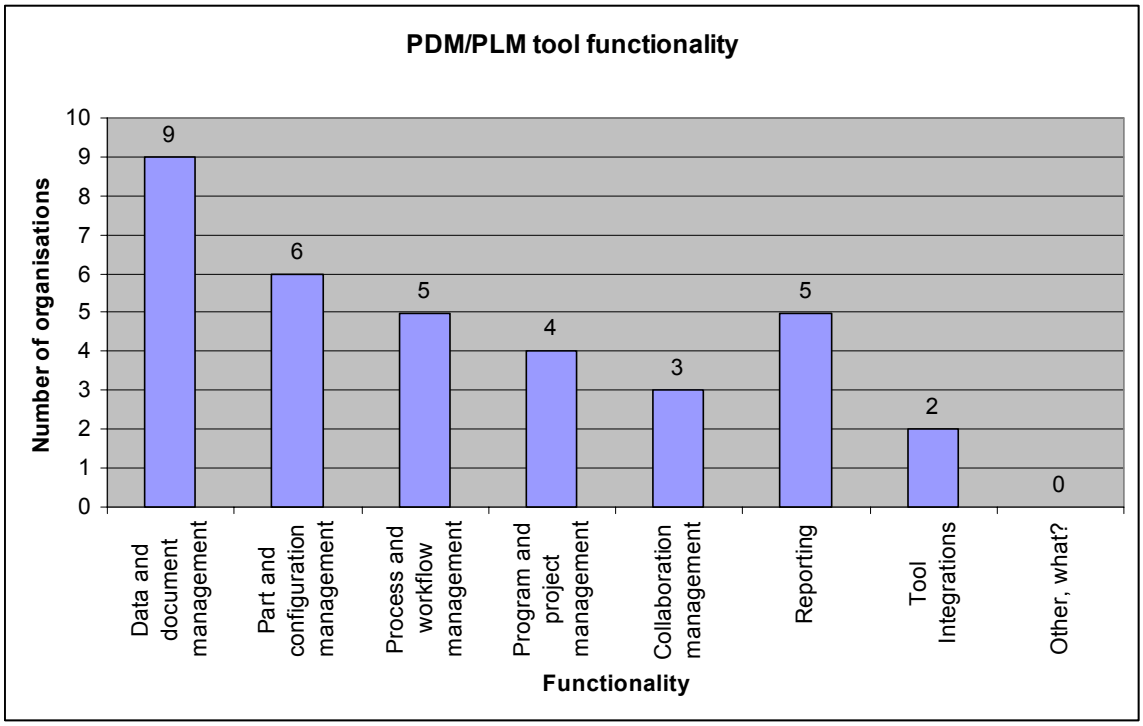

Figure 22. Product data and lifecycle management tool functionality.

4.4.3 Configuration management solutions

The respondents were asked to identify the configuration management solutions which are used in their organisations.

The most common CM solution among TWINS partners was IBM Rational ClearCase (see Figure 23). Four organisations out of 10 mentioned IBM Rational ClearCase as a tool they use for CM. Subversion was mentioned by three organisations, which made it the second most used solution for CM. CVS, MS SourceSafe, and Telelogic Synergy CM were each mentioned by two organisations. MS Team Foundation Server and

Serena Dimensions CM were mentioned only once. In the "Other" category, Jira and E-Matrix were mentioned once. In this category, one organisation mentioned that they have developed an in-house solution for CM.

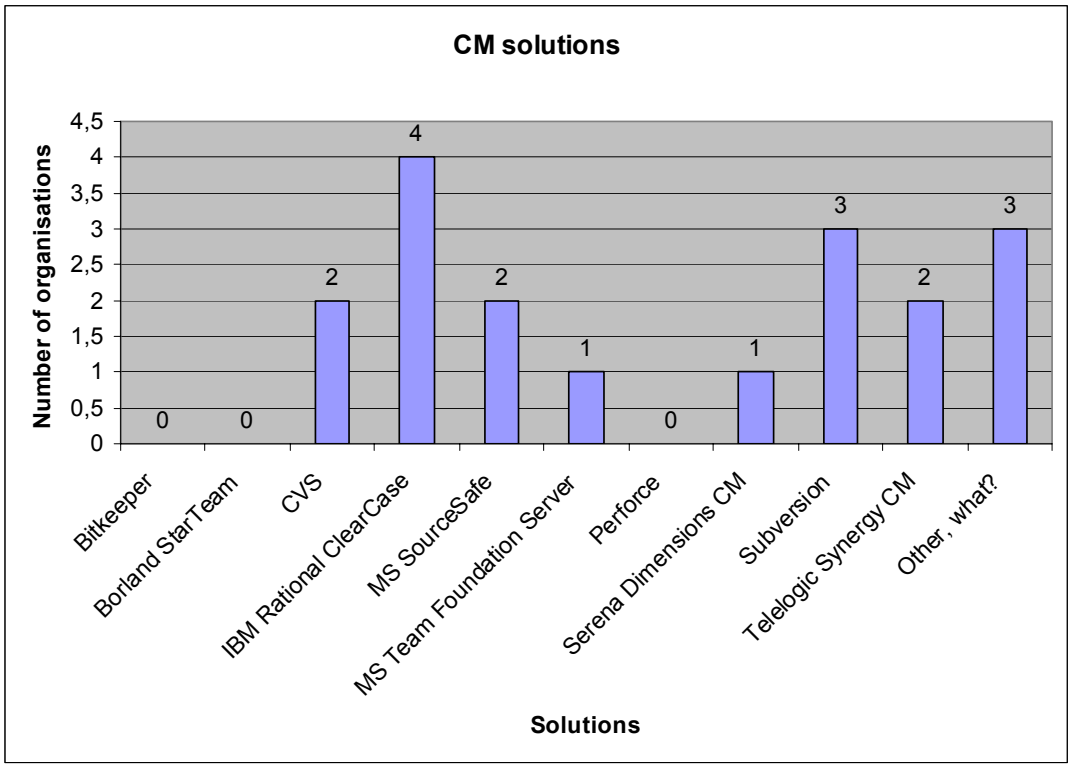

Figure 23. Configuration management solutions.

4.4.4 Functions of configuration management tools

The respondents were asked to identify the functions of configuration management tools that their organisations are using.

Organisations most commonly are using the version control function of their CM tool. Every organisation mentioned version control as a function they are currently using. Change management was mentioned by nine organisations, which made it the second most used function. Seven organisations advised that they are using the problem tracking function and five organisations mentioned build management. Both process support and collaboration management received responses from four organisations. Reporting was the most seldom used function of the CM tool and it was used by only three companies. There was one organisation that used all of the listed functionalities of its CM tool. (See Figure 24.)

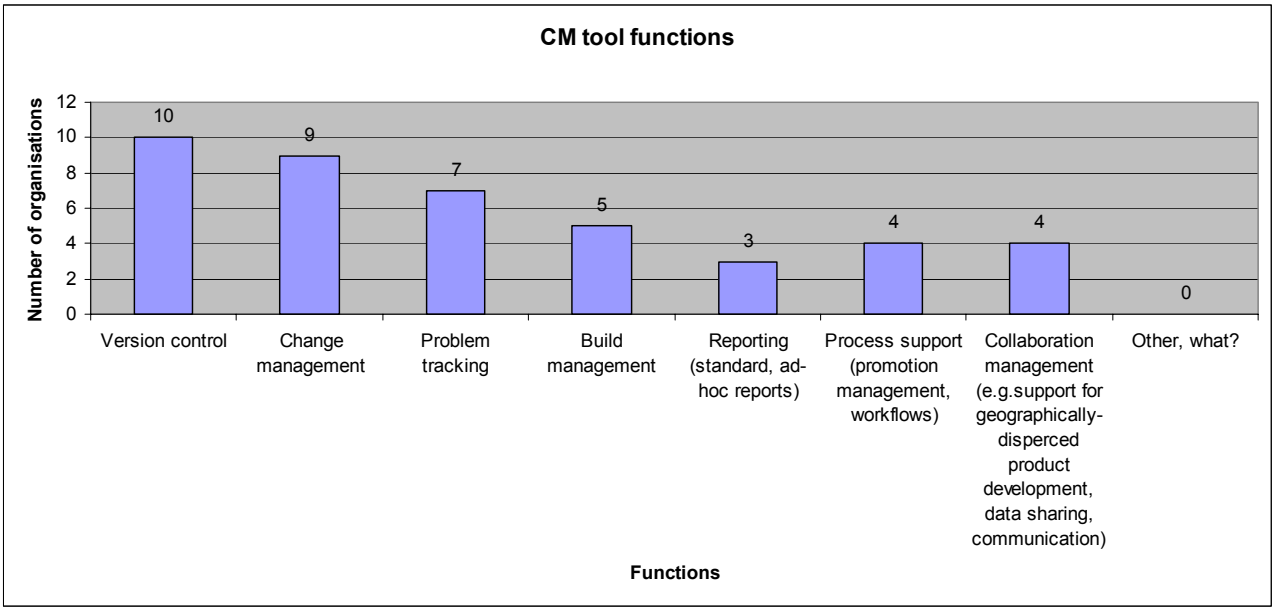

Figure 24. Configuration management tool functions.

4.4.5 Challenges in product information management

The respondents were asked to describe the challenges their organisation has recognised relating to product data management, lifecycle management or configuration management.

Every organisation had recognised different challenges when considering the using of PLM or PDM in their organisation. One organisation mentioned that PDM or PLM solution do not currently give enough support for HW/SW co-design. One organisation specified that the actors in HW and SW development should get only the relevant parts of information and not all of the information that is currently available in the documents. In that organisation, they also wish that PDM or PLM solutions improve common document management in their organisation. One organisation mentioned that they have need for better integration of PDM and CM. One organisation hopes that the use of PDM or PLM would unify the general processes (e.g. review process) that are used in HW and SW development.

Even though every organisation had recognised different challenges in CM, three answers highlighted HW/SW co-design. The questionnaire results show that interrelations between HW and SW development should be taken into account also from a CM point of view. These kinds of interrelations relate, e.g. to the ability of keeping track of the HW versions used relating to the correct SW, release management (dependencies) of components from different engineering disciplines and HW/SW co-design managed by CM. Other answers related to the integration of PDM and CM, improvement of documentation flow, rebuilding capabilities, increasing bug tracking / feature logging and traceability of evolutions.

5. Conclusions

This publication presents the results of the questionnaire used in ITEA-TWINS project. The TWINS project has 22 partners from Finland, Belgium, France, Spain and the Netherlands. The project consortium involves partners from a wide range of industries like automotive, avionics, copiers and printers, power supplies and telecommunication networks. The project wants to improve the project partner's competitive position by increasing the quality of the products while reducing the time to market.

This publication summarises the responses of TWINS partners to the questionnaire about tools, methods and challenges related to requirements engineering/management, architecture design and product information management.

The survey indicates that there are still many challenges in the areas of requirements engineering and management, architectural design and information management. Despite the fact that organisations use several methods and tools to manage their product development, there are still unsolved issues that have to be addressed in future development projects.

Although the survey was limited to only TWINS partners, it can be seen as giving directions towards state-of-the-practice in general in embedded systems development, as the participating companies are diverse and represent various domains within embedded systems. The findings are also in line with those presented in other published industrial experiences.

Acknowledgements

The authors of the publication acknowledge every organisation and respondents that gave their valuable time and contribution by answering the survey.

References

[Assimakopoulos 1998] Assimakopoulos, N. A. (1998) Systemic industrial management of HW/SW co-design. The Journal of High Technology Management Research, Vol. 9, No. 2, pp. 271–284.

[Balarin et al. 1997] Balarin, F., Chiodo, M., Giusto, P., Hsieh, H., Jurecska, A., Lavagno, L., Passerone, C., Sangiovanni-Vincentelli, A., Sentovich, E., Suzuki, K., and Tabbara, B. (1997) Hardware-Software Co-Design of Embedded Systems: The Polis Approach. Kluwer Academic Publishers.

[Berlack 1992] Berlack, H. (1992) Software configuration management. John Wiley & Sons.

[Chiodo et al. 1995] Chiodo, M., Giusto, P., Hsieh, H., Jurecska, A., Lavagno, L., and Sangiovanni-Vincentelli, A. (1995) Synthesis of software programs from CFSM specifications. In: Proceedings of the Design Automation Conference, June 1995.

[Crnkovic et al. 2003] Crnkovic, I., Asklund, U., and Persson Dahlqvist, A. (2003) Implementing and integrating product data management and software configuration management. Arctec House. ISBN 1-58053-498-8.

[Ernst 1998] Ernst, R. (1998) Co-design of embedded systems: status and trends. IEEE Design & Test of Computers, Vol. 15, No. 2, pp. 45–54. http://ieeexplore.ieee.org/iel4/54/14951/00679207.pdf?isnumber=14951&arnumber=679207

[Estublier et al. 2005] Estublier, J., Leblang, D., van der Hoek, A., Conradi, R., Clemm, G., Tichy, W., and Wiborg-Weber, D. (2005) Impact of software engineering research on the practice of software configuration management. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 14, Issue 4, pp. 383–430.

[Gupta 2001] Gupta, P. (2001) Hardware-software co-design. IEEE Potentials, Dec. 2001/Jan. 2002.

[Hofmeister et al. 2007] Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., and America, P. (2007) A general model of software architecture design derived from five industrial approaches. The Journal of Systems and Software, Vol. 80, No. 1, pp. 106–126.

[Ha et al. 2006] Ha, S., Lee, C., Yi, Y., Kwon, S., and Joo, Y.-P. (2006) Hardware-software Co-design of Multimedia Embedded Systems: the PeaCE Approach. In: Proceedings of the 12th IEEE Int'l. Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA'06).

[IEEE Std 1471-2000] IEEE Std 1471-2000. (2000) IEEE Recommended Practice for Architectural Descriptions of Software-Intensive Systems. New York: IEEE, The Institute of Electrical and Electronics Engineers. 23 p.

[Leon 2000] Leon, A. (2000) A Guide to software configuration management. Boston: Artech House.

[Lockheed Martin 2006] Lockheed Martin (2006) Hardware/software co-design appnote. http://www.atl.lmco.com/projects/rassp/RASSP_legacy/appnotes/HWSW/APNOTE_HWSW_INDEX.HTM

[Micheli & Gupta 1997] Micheli, G., and Gupta, R. (1997) Hardware/Software Co-Design. In: Proceedings of the IEEE, Vol. 85, No. 3, March 1997.

[Stackpole 2003] Stackbole, B. (2003) Product Lifecycle Management Promises to Streamline Development, Boost Innovation. CIO.com, 2007. http://www.cio.com/article/31906/Product_Lifecycle_Management_Promises_to_Streamline_Development_Boost_Innovation

[Stark 2006] Stark, J. (2006) Product Lifecycle Management (PLM) State of the Art Report. Geneva, Switzerland: John Stark Associates.

[Stevens 1998] Stevens, R., Brook, P., Jackson, K., and Arnold, S. (1998) Systems Engineering: Coping with Complexity. Pearson Education. 374 p.

[Vial & Rouzeyre 1997] Vial, C., and Rouzeyre, B. (1997) Hardware-software co-synthesis: Modelling and synthesis of interfaces using interpreted Petri nets. In: J.-M. Berge, O. Levia, and J. Rouillard, editors, Hardware/Software Co-Design and Co-Verification. Kluwer Academic Publishers.

[Wikipedia 2006] Hardware/software co-design. http://en.wikipedia.org/wiki/Hardware/software_codesign

Appendix A: Summary of the questions

Part A: "Background information" consisted of the following questions:

A1. Select your organisation

A2. What is the size of your company, i.e. total number of employees?

A3. How long is your product's life-time (used by the customer, maintenance and backwards compatibility)?

A4. Select the collaboration modes that are used in your company from the following list

A5. Give an estimation on how much of your product is developed in collaboration, as a percentage

A6. Specify the number of requirements in a typical project

A7. Identify your position in the company

Part B: "Requirements engineering and management" consisted of the following questions:

B1. Are the same methods and tools used for systems requirements engineering and for HW/SW subsystems requirements engineering (if not, explain why)?

B2. Identify the methods your organisation currently uses for requirements gathering

B3. Identify the tools your organisation currently uses for requirements management (RM)

B4. Specify the traceability relations from requirements to other development artefacts (or vice versa) that have to be managed in your company (e.g. requirements to test cases, features to requirements, requirements to source code...)

B5. Describe how you store the traceability information of the requirements identified in the previous question?

B6. List the challenges your organisation has recognised relating to requirements engineering or management in HW/SW co-design

Part C: "SW architecture design solutions" consisted of the following questions:

C1. Select the lifecycle model of architectural descriptions that is relevant for your company

C2. Select the viewpoints that are used in architectural descriptions from the following list

C3. List the methods and techniques your company currently uses for architecture design (e.g., SEI's Attribute-Driven Design (ADD), Siemens' 4 views, RUP's 4+1 views and Nokia's Architectural separation of concerns (ASC)

C4. List the notations and models that are currently used in architectural descriptions in your company (e.g. Informal modelling sketches, UML notations...)

C5. List the tools your organisation currently uses for architecture design (e.g. Visio, Borland Together...)

C6. List the challenges your organisation has recognised relating to architecture design in HW/SW co-design

Part D: "Product information management solutions" consisted of the following questions:

D1. Identify the PDM (Product Data Management) or PLM (Product Lifecycle Management) solutions your organisation currently uses

D2. Identify what PLM/PDM tool functions are used in your company

D3. Identify the CM (Configuration Management, Version management) solutions your organisation currently uses

D4. Identify what CM tool functions are used in your company

D5. Describe the challenges your organisation has recognised relating to PLM or PDM in co-design

D6. Identify challenges your organisation has recognised relating to CM (Configuration Management, Version management) in HW/SW co-design

ISBN 978-951-38-7150-5 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)