Design Process for Intelligent Algorithm Intensive Systems

Juhani HIRVONEN
VTT
Box 1000, FIN-02044 VTT, Finland
and
Olli VENTÄ
VTT
Box 1000, FIN-02044 VTT, Finland

ABSTRACT

A variety of algorithms and mostly software-based technologies have been developed in order solve complex problems in technical systems. For example, machine learning, artificial intelligence, pattern recognition, neural networks, fuzzy logic, statistical methods, operation analysis, and most recently sensor networks have been actively studied. It is widely acknowledged that advanced algorithms have large potential, but due to their impractical applicability they are neglected in the engineering processes. The paper outlines systematic engineering design practices for algorithmic or knowledge-intensive intelligent systems and services.

Keywords: semantics, modeling, design, intelligent systems

1. INTRODUCTION

A large number of advanced methods and algorithms for solving complex problems in information processing have been developed during the recent decades. On one hand, these methods often show remarkable performance but, on the other hand, they often fail to proceed to true industrial use. Their performance may collapse easily during practical usage, and solutions are seldom transparent or well understood in field use, which leads to a lack of trust. It is also typical that an expert with these methods is constantly needed once these methods have been installed.

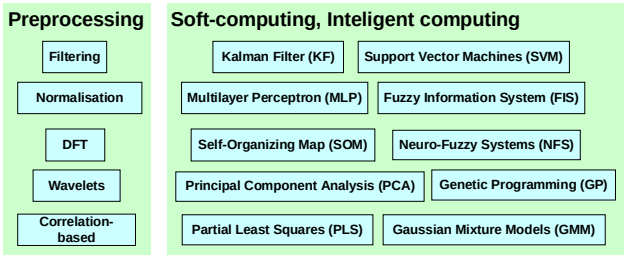

Figure 1. Well-known popular intelligent algorithms exercised and piloted extensively already over decades

Advanced algorithms and mostly software-based technologies require considerable knowledge and experience to be applied effectively, correctly, and transparently, necessitating participation of skilled professionals. From practical point of view, this makes applications expensive to design and maintain. A further challenge is created by the need for integration of algorithms and continuous system modifications [1]. The extensive research in this field has generated numerous demonstrations and pilots but only few practical solutions or commercial products [2], [3], [4], [13]. It must be admitted that these methods are not in the menu of industrial designers.

The purpose of this paper is to outline the development of a systematic engineering practice for algorithmic or knowledge-intensive intelligent systems and services. We believe that advanced technologies and algorithms can not be economically feasible unless standardized or systematic design practices, tools and system components are available [1]. There exist many kinds of engineering and project management frameworks to carry out effective software engineering, mechanical engineering, civil engineering, and construction. These frameworks have proven their merits in numerous activities. Constructing algorithms intensive applications should benefit of the same systems engineering principles, analogously.

There are many elements in a design framework. At first, it should have clear design life-cycle stages. Concept creation, feasibility study, design, manufacture, use, upgrades, and disposal represents a typical and even generic stage sequence. Secondly, engineering processes are requirements-driven rather than technology-driven. Engineering means constructing viable solutions or meeting the requirements but not piloting the possible merits of interesting technologies. Designers also want to study various design alternatives to a reasonable depth. Industrial design means constructing solutions that are known or ascertained to meet the requirements. Industrial design does not mean piloting in a laboratory how good your algorithms may be, or to prove that your special algorithm is just better than those benchmarked in the experiment.

The design tools should help the designers effectively in constructing the representations peculiar at various

stages. Effective tools also provide features that automate the generation of subsequent representations along the design flows. Reuse is an important point-of-view. The engineering environment should also guide designers in arriving in good partial solutions, practices, and at the same time allow the necessary creativity and flexibility. In summary, a good design practice is foremost a knowledge-management challenge: both in encoding the relevant knowledge and then effectively and accurately benefiting of it.

Figure 2 clarifies the focus of the paper. A product or system life-cycle starts with user needs and requirements (problem or opportunity space) and then goes deeper into product design, realization, verification, and validation ending up with a ready-to-deliver product or system. Typically, the so-called state-of-art engineering processes are adequate to manage most of the needs or requirements and the subsequent engineering task. However, some of the challenges are so special or hard that they can not be satisfied by using state-of-the-art methods. Instead algorithmic or intelligent solutions are needed. Advanced algorithms do have large potential but, due to their impractical applicability, they are often neglected in the engineering processes.

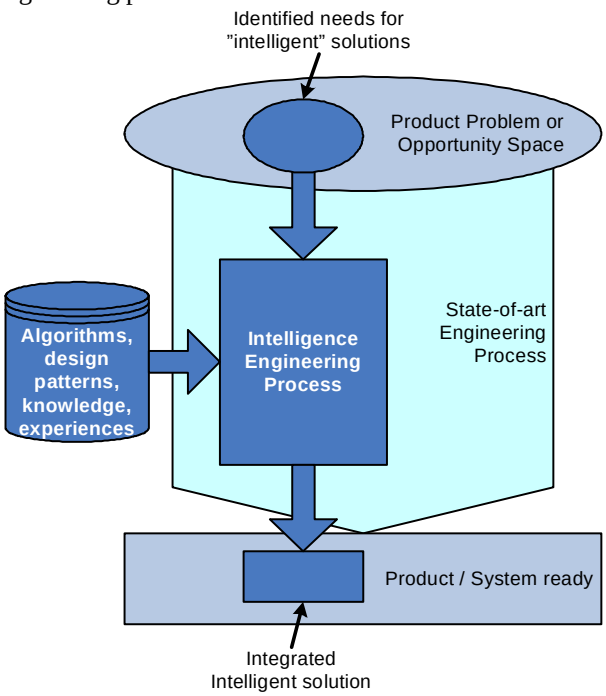

Figure 2. How to use advanced techniques to create systems that function in an intelligent way

2. DESIGN OF ALGORITHMIC SYSTEMS

Algorithmic intelligence

Because the concept of intelligent system is vague we will first state what we mean by intelligent system:

"A system is intelligent if it behaves appropriately with respect to its purposes, not only in circumstances it was explicitly designed for, but also in rare, unexpected and even new kinds of situations."

It should be firstly noted that what we mean by intelligence refers to the *functionality* of a system, not to its static features. Secondly, the appropriateness of the behavior is evaluated in terms of system values and goals. Thirdly, intelligence requires the system to adapt itself to changing and unfamiliar situations. To enable intelligent behavior, systems must have concrete capabilities like advanced computation and reasoning, plug & play, reconfiguration, adaptation, learning, co-operation, associating values with goals, data and alternative actions, rational decision making, etc..

In order to restrict the problem space and the design target we have selected a narrower scope by *focusing on the design of systems that exploit computational algorithms for identifying system states and adapting to varying circumstances. Regarding design methods, we are focusing on model-based approaches as they appear to give the best tools to design complex entities.*

Algorithmic intelligence design

Inspired by other domains Figure 3 depicts the structure of the intended engineering process for algorithm-intensive system design showing also the respective models, tools, platforms, and libraries.

With such an engineering system the designer will e.g. be able to:

- write requirements specifications that 1) are capable of describing the essentials of the tough items in their contexts, and 2) instruct and support the subsequent concept creation and early design choices
- deal with prospective alternative design options, judge the merits of various choices
- objectively choose algorithms that are known or expected to fit to the needs; in contrast to ad hoc experimenting, or, sticking to own approaches
- benefit of and increase the necessary knowledge, expertise, and experience related to the technology and application domains
- tailor, refine, tune, learn, assemble the algorithms appropriately, in a manageable way
- ascertain, verify and validate the performances of the design options, at various design stages
- work in a tools-assisted manner, i.e., with the help of appropriate design user interfaces, design editors, solution generators, various browsers, etc.
- as a consequence of all the above, speed-up the uptake of new methods in commercial actually used products and services

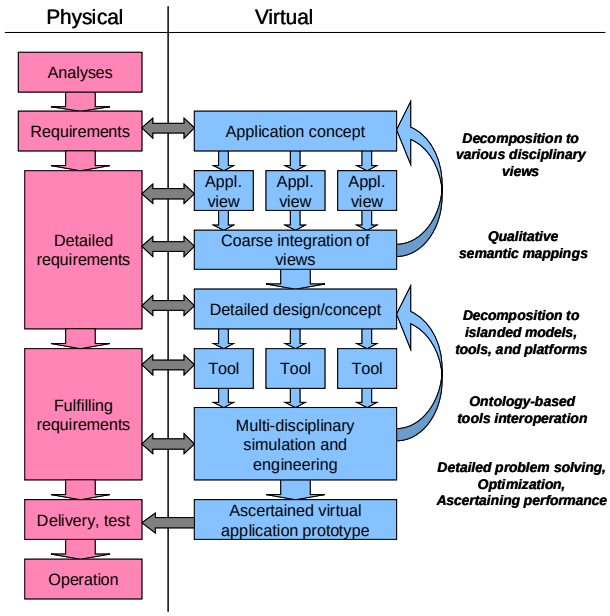

Figure 3. Schematic algorithmic engineering process

3. SAMPLE APPLICATIONS

We have approached our framework development by sample applications which are *monitoring or control of industrial processes, critical care unit.*

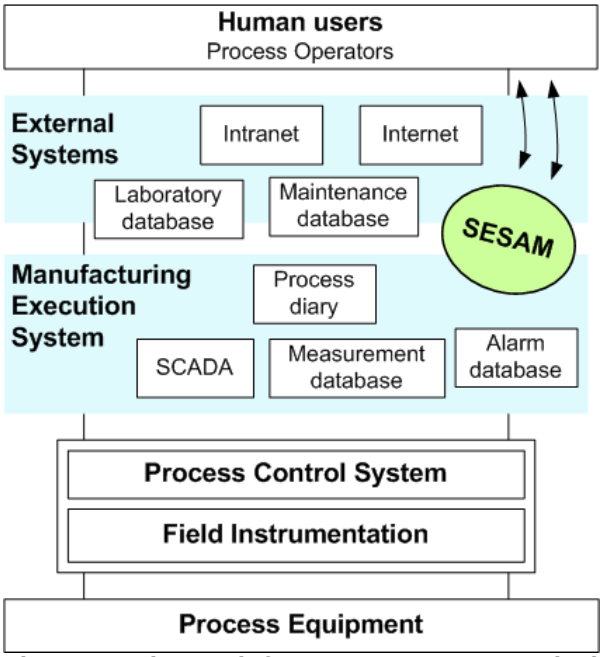

Figure 4. Plant IT infrastructure augmented with an intelligent system

Critical care application

The measurement of (continuous) intravenous blood pressure is essential for critical care patient management, but the measurement method is prone to occurrence of artifacts that increase the risk of making wrong interpretations of the blood pressure waveform, thus the ***validation of intravenously recorded blood pressure data during on-line recording*** is considered a key service of the system.

Complex plant monitoring

Figure 4 shows how an intelligent, situation aware system (SESAM) relates to an existing plant IT infrastructure. The main purpose of the intelligent system is to support control room operators and other plant personnel in maintaining their situational information and using it in their actions.

4. IMPLEMENTATION OF DESIGN FRAMEWORK

Our strategy is that the intended tool shall be **primarily based on knowledge-management.** We intend to use semantic modeling to describe the various aspects of the algorithms, to represent suggested algorithms constructs, to generate the design concepts and the actual design prototypes, to manage with training and testing data, to encode the various experiences both on the methods and application domains, etc. Another knowledge-management technology to be utilized is design patterns [14]. Eventually the intended design tool shall employ other technologies, too, but the core will be knowledge-management.

For systematic usage of algorithms an effective analysis or requirements specification will help the designers in identifying the “tough items” in their problem space, i.e., issues which either call for exceptional solutions. Secondly, we must learn how to describe such tough items so that it is both communicative to the stake-holders (end-user, etc.) of the intended device or system, and also instructive for the designers in their subsequent design activities, decision-makings, methods assessment, etc. Intelligent algorithms represent often complex features, properties, and behaviors, which are hard to specify and describe.

Conceptual design

Conceptual design means developing alternative algorithm or system architectures tentatively capable of fulfilling the requirements. At this stage, the design tool should be able to review and suggest algorithm constructs that by experience are potential or capable to meet the respective items in the requirements specification. Ideally, the tool should be able to map the formal requirements to the set of admissible alternative solutions.

In order to have such a tool the respective methods and algorithms must be described in the tool in such a way that the intended mappings or at least reviews against requirements are practical. As described below, we suggest the use of semantic modeling together with appro-

priate architectural design patterns to encode such knowledge of the characteristics, capabilities, properties, performances, experiences, restrictions of known algorithms. Since assemblies of algorithms are often used such a knowledge base must encode and store the various interconnection rules of the algorithms. Similarly, the knowledge base should contain principles, guidance, and experiences of the use of algorithms or algorithm constructs.

Virtual prototype generation
The design concepts or system architectures of the previous stage should directly indicate how to implement the detailed algorithm components, or how to choose algorithms from the tool library, or from a commercial or open domain package of algorithms. Current state-of-art of the availability of software offers already plenty of opportunities for efficient design automation.

The result of this stage, on the other hand, are expected to be executable virtual prototypes directly ready for learning, tuning, adaptation, calibration, verification, validation, and testing, i.e., activities leveling up the prototype constructs into intended and expected behaviors and giving grounds to choose the most appropriate construct for final solution.

Verification, validation, and testing
Verification, validation, and testing formally means ascertaining the performance made explicit in the requirements specification. In practice, these activities mean many kinds of data-management challenges. Data-management and algorithm-selection are highly mutually dependent issues. The kind of learning and testing data which is available very much dictates what algorithms may be used. Or vice versa, the required performance criteria typically restrict what algorithms or algorithm constructs must be used and, consequently, indicate what kind of data is necessary to obtain. Knowledge about various data management aspects are vital elements of the knowledge base discussed earlier

5. MODELING ASPECTS

The proposed design flow and tool will be based on models and application concepts so the following requirements have to be taken into account.

- The models created during design process must cover both the intelligent technical system itself and its environment or context (e.g. process to monitored). In addition, they will describe the current situation and other input information to design.
- Models should also include various types of information at different abstraction levels, including, requirements engineering data (problems, goals, design constraints, etc.), system functions and implementation.
- Models must be well structured (formal) and computer-processible allowing electronic information management and exchange at the minimum. As far as possible, however, the models, should be "computer-understandable" so that they support advanced analysis, inference and integration of partial models
- Complex applications are hard to understand, learn and test. Therefore the models should be executable. This would allow system animation and simulation early during the design stage.
- Process must rely heavily on the reuse of design knowledge. For the early design stages it needs, for example, patterns and measurable features for application domains, requirements and available solutions.
- Powerful libraries and methods like similarity measures or case-based reasoning must be used to match application requirements with suitable solutions, such as architectural patterns and advanced algorithms.

Semantic modeling
Our approach is to use semantic modeling on all levels of the design. We are in the process of developing ontologies for e.g. Functions, Information (Data), and Techniques (Algorithms) for the purposes of matching problems and solutions, see Figure 5.

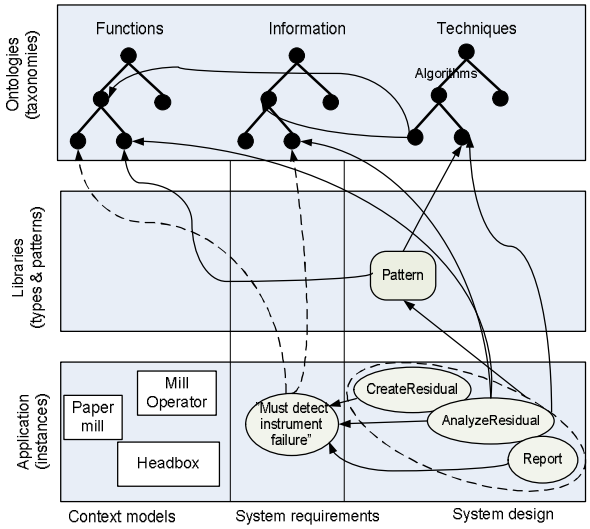

Figure 5. Matching problems and solutions

For the time being there exists a plethora of ontologies for different purposes. For example, in mechanical engineering, artifact functions have for a long time been the starting point of product design [6]. In the beginning this century the US National Institute of Standards and Technology (NIST) collected heterogeneous knowledge and data in to a design repository. On the basis of several previous efforts [7] suggested a combined taxonomy of functions and flows in the electro-mechanical design space. These taxonomies result from analysis of a large number of existing artefacts and are based on traditional concepts of functions and flows of material, energy and

signals (information). In a more recent study [8] has criticized these taxonomies as internally inconsistent and unsatisfactory from the viewpoint of formal ontologies.

Taxonomy for functions of algorithmic systems

There are also taxonomies and ontologies dealing with algorithms and data see e.g. [9]. Anyhow, taxonomies for functions of intelligent algorithmic systems seem to be missing. Hence, we will adapt to the following top level classification of these functions, depicted in Figure 6. Basically we will adopt the classical input/process/output paradigm with an addition of Tasks.

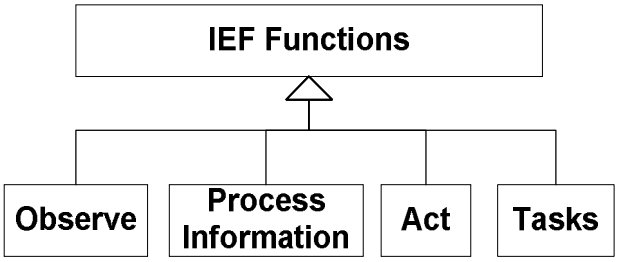

Figure 6. Top level classes

Furthermore we have defined the necessary subclasses (SOA stands for state of the affairs, double colon denotes superclass):

Table 1. Examples of functions of algorithmic systems

Term	Definition
Produce::Act	Producing a desired SOA
Maintain::Act	Keeping a desired SOA
Observe	Inputting information from external and internal SOA
Act	Causing changes in external and internal SOA
Detect::Process information	Becoming aware of a certain (predefined) kind of SOA
Validate::Process information	Ensuring that information or a statement about SOA is correct or true
Estimate::Process information	Calculating approximation of (a part of) SOA which is usable even if input data may be incomplete, uncertain, or noisy.
Predict::Estimate	A rigorous statement forecasting what will happen under specific conditions
Classify::Process Information	Determining the category that an entity being classified belongs to
Filter::Process Information	Changing, removing, or reducing entities by amplifying relevant aspects or attenuating/removing irrelevant aspects
Match::Process Information	Testing entity for being similar to a template entity

6. FURTHER ISSUES OF INTELLIGENT SYSTEMS AND THEIR DESIGN

The purpose of this chapter is to draw a "big picture" of the problem domain. The topics to be discussed are: 1) characteristics of intelligent systems and 2) their implications for the design principles.

Characteristics of intelligent system

Integration of heterogeneous components: There are stand-alone intelligent systems, but often there is a need to combine knowledge and services provided by several more or less intelligent systems. Such combinations can be referred to as "systems of systems". They are characterized by large-scale networked integration of heterogeneous (technical and human) components [1], [13]. Consequently, intelligent systems combine services provided by different nodes and subsystems developed, owned and maintained by separate companies. As a service can be used in more than one distributed application, various intelligent systems can overlap, i.e. system boundaries depend on the viewpoint.

Rationality: Intelligent behavior should be guided by stated or implied goals. These goals may be in conflict with each other. In addition, the actors within an intelligent system must base their actions on lacking and uncertain information. Therefore, rational decision making and management of uncertainty are features required for real intelligence.

Adaptation: Intelligence requires a system to change its behavior according to the current situation. In other words, adaptation is a key characteristic of intelligence. Adaptation can be achieved e.g. by modifying system composition (reconfiguration) or by fine-tuning the functionality of existing parts. Adaptation can also be achieved by making use of the human actor's capabilities to interpret situational requirements and change behavior to maintain aimed results.

Encapsulated and emergent intelligence: The intelligence can be encapsulated within individual components or emerge from the co-operation and dynamic reconfiguration of less intelligent elements. In both cases the mechanisms required for intelligent behavior call for flexible system architectures, powerful data models and computationally advanced algorithms. In case of "encapsulated intelligence", the emphasis is often on the use of known algorithmic solutions, whereas "emergent intelligence" is based on complex interaction patterns.

Implications for design practice

The functional and technical features of intelligent systems give rise to the major design challenges that need to be solved. These implications are shortly discussed below.

Enabling system transformations and reconfiguration during the lifecycle: Intelligent systems tend to be complex and often can not be fully completed during design time. As applications and technologies change, intelligent systems are modified more or less continuously. The system should provide mechanisms that make

these changes flexible, efficient and safe. There is a need for generic solutions and platforms that would enable future changes in the systems. Experimental data should be used in guiding further development of the systems.

Efficient and appropriate use of intelligent solutions: For design flow to be efficient practical solutions and advanced methods must be encapsulated into reusable knowledge and components that can be easily applied in integrated engineering tools, in runtime environments, or both. This may happen within a company or a group of firms operating in the same application area. However, only global commercial markets of "intelligent engineering components" provide true benefits for component developers and system integrators. This means that standardized and open (but obviously domain-specific) application frameworks are needed for both design tools and actual execution platforms.

Selection and appropriate use of technical solutions like algorithms in specific design problems requires thorough understanding and explication of both the problem area and the specific features of the technical solutions themselves. If the method is not appropriate for the application at hand an otherwise reliable solution may fail. In other words, intelligent solutions can not be seen as a separate issue but as part of the overall design process.

Diversity of elements of the system: Intelligent systems are often composed of heterogeneous components that must be able to act together. Therefore the ways of functioning of the system, i.e. the coordination and the communication mechanisms, must be defined. Furthermore, communication protocols and other technical solutions that enable adaptive patterns of the joint system need to be created. Ontology is one way of creating common language that the diverse elements of the intelligent system can understand. Implementing adaptive patterns in intelligent systems requires close cooperation of various engineering disciplines and human factors experts.

Maintaining reliability, safety and security: The application areas selected as examples for design flow development have high quality standards. This may be in contradiction with the rapid system evolution and use of advanced techniques that are hard to verify. Therefore, there is the need for structured and formalized models that support systematic, computer-assisted analysis, testing and simulation of design artifacts. In addition, the design process and working practices must be organized in a way that makes effectively use of these facilities.

7. FUTURE WORK

For the time being we are in the process of developing the required ontologies. Furthermore the design tool will require extensive libraries of patterns, templates, and algorithms. We will start to explore similarity measures between problems and solutions to be able to guide the designer in the process. One possible solution to these measures can be obtained through fuzzy ontologies [11], [12]. Finally we need ways to express statements like "solution X is suitable for problem Y in the given context Z and overall goal G".

8. REFERENCES

[1]. ARTEMIS, "**Design Methods and Tools**", European Platform on Intelligent Embedded Systems Strategic Research Agenda http://www.artemis-office.org/

[2] K. Leiviskä ed, "**Smart Adaptive Systems: State-of-the-art and challenging applications and research areas**", http://www.eunite.org/eunite/roadmap/02Roadmap1.pdf, 2004

[3] P. P. Angelov, **Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems**, Springer-Verlag, Heidelberg, New York, 2002

[4] S. Riedel, and B.Gabrys, "Combination of Multi Level Forecasts**". Int. J. of VLSI Signal Processing Systems**. 2007, Vol. 49 (2), pp. 265-280,

[5] B. Giovanni and A. Pietrosanto, "Instrument Fault Detection and Isolation: State of the Art and New Research Trends" **IEEE Transactions on Instrumentation and Measurement**, IEEE, February 2000, pp. 100-107

[6] G. Pahl, and W. Beitz**, Engineering Design - A Systematic Approach**, Springer, London, 1996

[7] J. Hirtz, R. Stone, D. McAdams, S. Szykman, and K. Wood, "A functional basis for engineering design: Reconciling and evolving previous efforts", **Research in Engineering design** 13 , 2002, pp. 65 – 82.

[8] P. Garbacz, "Towards **a standard taxonomy of artifact functions**", http://www.pawelgarbacz.webpark.pl/taxonomy.pdf.

[9] P. E. Black, ed., "**Dictionary of Algorithms and Data Structures**", U.S. National Institute of Standards and Technology. http://www.nist.gov/dads

[10] **Helsinki Testbed**. http://testbed.fmi.fi

[11] Sanchez, E., and Yamanoi, T., "Fuzzy Logic and Semantic Web**", IPMU Proceedings** 2004

[12] C. Carlsson, R. Fullér, and P. Majlender, "A Fuzzy Real Options Model for R&D Project Evaluation", **Proceedings of IFSA 2005**, Beijing, 2005, pp 1650-1654

[13] B, Gabrys, K. Leiviskä, J. Strackeljan (Eds), **Do Smart Adaptive Systems Exist?: Best Practice for Selection and Combination of Intelligent Methods**, Springer-Verlag, Berlin Heidelberg, 2005

[14] J. Coplien, B. Woolf, **A Pattern Language for Writer's Workshops, Pattern Languages of Program Design** 4, pp. 557-580, Addison-Wesley Longman, 2000.