CHALLENGES FOR AGILE DEVELOPMENT OF COTS COMPONENTS AND COTS-BASED SYSTEMS
A Theoretical Examination

Iva Krasteva
Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier Blvd, Sofia 1165, Bulgaria

Per Branger, Rikard Land
Department of Computer Science and Electronics, Mälardalen University, Västerås, Sweden

Keywords: Agile software development, component-based development, COTS-based development.

Abstract: Component-based software engineering has had great impact in the desktop and server domain and is spreading to other domains as well, such as embedded systems. Agile software development is another approach which has gained much attention in recent years, mainly for smaller-scale production of less critical systems. Both of them promise to increase system quality, development speed and flexibility, but so far little has been published on the combination of the two approaches. This paper presents a comprehensive analysis of the applicability of the agile approach in the development processes of 1) COTS components and 2) COTS-based systems. The study method is a systematic theoretical examination and comparison of the fundamental concepts and characteristics of these approaches. The contributions are: first, an enumeration of identified contradictions between the approaches, and suggestions how to bridge these incompatibilities to some extent. Second, the paper provides some more general comments, considerations, and application guidelines concerning the introduction of agile principles into the development of COTS components or COTS-based systems. This study thus forms a framework which will guide further empirical studies.

1 INTRODUCTION

As software systems are increasingly built using pre-fabricated components, there is a need to consider how the processes need to be changed compared to "classical" development where all parts of the software are built in-house. In particular, one needs to distinguish between development of components and development of systems made of components. It is possible to define three types of development (Crnkovic, Larsson and Chaudron, 2006), characterized mainly by the business relationships between these two processes (whether they occur in the same or different organizations): *architecture-driven development* (where components are defined from a top-down decomposition and developed internally or outsourced), *product line development* (where components are built internally but reused in several products) and *COTS-based development* (where components are developed and made available on an open marketplace; COTS stands for "Commercial Off-the-Shelf"). This paper focuses on *COTS-based development* including the production of COTS components as well as the development of systems made of COTS components.

In recent years agile methodologies for software development have proved very effective in the current dynamic business environment. More and more, the agile approach broadens its areas of application to domains and projects previously considered unsuitable for agile development (Turk, 2002). An EU project intended to scaling up agile approaches in a globally distributed environment is ongoing (FLEXI[1]), research papers and case studies report on using agile ideas in safety-critical systems (Greening, 2001; Bowers, 2002; Wayrynen, 2004) and hardware intensive projects (EUREKA-ITEA AGILE Project[2]; Manhart, 2004).

[1] http://flexi-itea2.org/index.html
[2] http://www.agile-itea.org/

Krasteva I., Branger P. and Land R. (2008).
CHALLENGES FOR AGILE DEVELOPMENT OF COTS COMPONENTS AND COTS-BASED SYSTEMS - A Theoretical Examination.
In *Proceedings of the Third International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 99-106
DOI: 10.5220/0001763400990106

The research on introducing agile ideas in component-based systems and component development itself is still quire sporadic and isolated and most of the suggested solutions are partial. The Evolutionary Process for Integrating COTS-based systems (EPIC) (Albert and Brownsword, 2002) is a framework for building, fielding and supporting COTS-based solutions. It presents an alternative approach for acquisition, management, and engineering practices of COTS-based systems which is based on Rational Unified Process (RUP). Another study (Cooper, 2006), which sets EPIC as a ground, extends the research question further towards agile ideas and presents a set of questions that need to be considered to introduce agility into the process from a requirements engineering perspective. Another part of COTS development process, the selection of COTS components and the applicability of agile principles for component selection, is examined by (Navarrete, Botella and Franch, 2005). A description of using CLARiFi system (CLear And Reliable Information for Integration) as an agile approach for retrieving components from large repositories is provided by (Scotto et al., (to appear)).

The aim of this paper is to present comprehensive and systematic evaluation of the applicability of agile approach in component-based development processes. In addition, this theoretical work lays the foundation for further empirical studies in an industrial setting.

This paper is oriented towards a reader who wishes to learn what to consider before introducing agile methods into a component development project or system development with components. For more details, we refer the interested reader, as well as the reader who wishes to scrutiny how we have arrived at our conclusions, to a more detailed report (Krasteva, Branger, Land, 2007).

The research method and the motivation behind it are described in section 2 of the paper. Further details about the systematic structure of the research can be found in section 3. Section 4 reports the main findings, followed by a discussion in section 5. Section 6 concludes the paper.

2 RESEARCH METHOD

Since the combination of component-based development (CBD) and agile approach is a novel and scientifically largely unexplored area, there are two research approaches, which are complementary and are both necessary:

- **Theoretical reasoning**, where the fundamental assumptions and inherent characteristics of the two fields are compared, and any theoretical incompatibilities are reported. The risk in this type of study is that when comparing two models of reality, which have evolved in parallel from different needs with different concepts and terminology, the theoretical reasoning may become too disconnected from reality.
- **Empirical studies** of projects and organizations where agile practices have been adopted in component-based software processes. However, without a theoretical basis for such studies, it becomes extremely difficult to formulate research questions which are relevant and concrete enough, and to design study settings to actually allow examination of the topic intended to be studied.

This work represents the first of these two approaches and should be seen as a first phase, laying the foundation for further empirical studies in an industrial setting. These two steps are well-defined parts of the research agenda of the established PROGRESS Centre for Predictable Embedded Software Systems[3] and also the ITEA2 FLEXI project[1] of which we are part.

When comparing Agile and CBD as two independently evolved research areas, one first step is to bridge all gaps between differences in their respective self-representation, i.e. terminology and concept formation. During this type of comparison, the fundamental set of "facts" of each field should be identified so that the majority of practitioners and researchers in this field would agree with this choice, and of course to make this choice explicit. At each step of the logical reasoning, we have been careful to document the basis for our conclusions, to make choices explicit and to motivate them, thus opening up our work for external scrutiny and criticism.

Our aim has been to:

- Use the most established basis for terms and concepts in these two fields, and present our interpretation for scrutiny (thus ensuring *construct validity*);
- Make our logical conclusions as explicit as possible (thus ensuring *internal validity* and *reliability*, also called *conclusion validity*).

In the paper, in order to compare an *approach* – an agile approach, and a *process* – the CBD process, we study the application of agile values and principles

[3] http://www.mrtc.mdh.se/progress/

for each of the activities of CBD processes. The structure of the research is a subject of the next section.

3 RESEARCH STRUCTURE

Agile values, stated in the Agile Manifesto[4] in 2001, outline the ideas of agile development. The twelve agile principles behind the manifesto support all of the values and provide directions for applying these ideas in a software project. In the proposed study it is examined how agile principles can be mapped to concrete tasks and activities of CBD processes. Although a subjective step of interpretation and application is needed, we have made sure to externalize the interpretation as much as possible by supporting our conclusions with practices of different agile methods such as XP (eXtreme Programming) (Beck, 1999), Scrum (Schwaber, 2004), Crystal Clear (Cockburn, 2004), DSDM[5] (Dynamic Systems Development Method) and Lean Development (Poppendieck, 2003).

For processes in the component-based community there is not definite set of "facts", nor any standard text to refer to in this matter. We have chosen to use (Crnkovic, Larsson and Chaudron, 2006) for the fundamental description of the component-based processes and their relation, and complemented this with other descriptions of characteristics of component-based processes, such as (Crnkovic and Larsson, 2002; Heineman and Councill, 2001; Wallnau *et al*, 2001).

The study and the presentation are structured in the following way. First, the development process of components is separated from the process of system development with components (Crnkovic, Larsson and Chaudron, 2006; Heineman and Councill, 2001; Wallnau *et al*, 2001). For each of the two processes, the basic development activities according to Sommerville (Sommerville, 2006) are listed. Then the meaning of the agile principles in the context of given activity is studied and exemplified with appropriate practices. Finally, the observations of applicability of agile principles into CBD process activities and tasks are summarized into several groups:

- **Comments** point out some important characteristics of a particular activity or agile principle
- **Contradictions** between agile principles and specifics of given process activities and tasks (sometimes with proposed **Solutions**)
- **Considerations** describe the things that should be taken into account when trying to apply agile principles to an activity
- **Applications** provide summary of the observations and guidelines on how to introduce the agile ideas to the development lifecycle in practice.

The approach is summarized in Figure 1. Activities which cross the complete development processes are further discussed in Section 5.

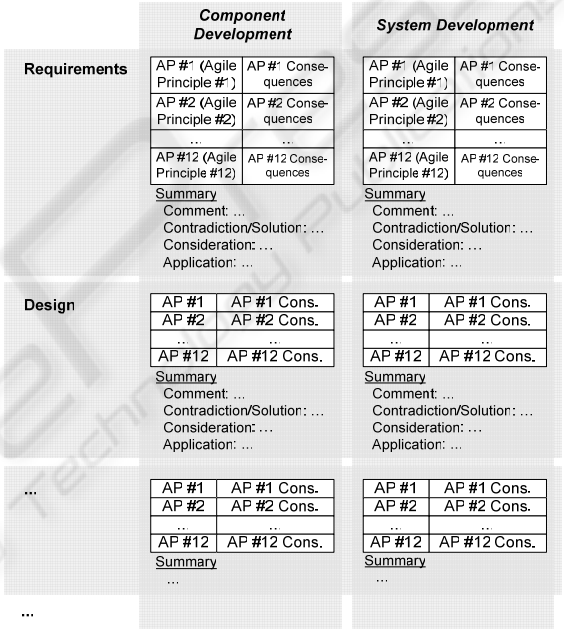

Figure 1: The break-down structure of the study.

We would like to point out that no particular order of the process activities is assumed or advocated (e.g. like a waterfall model). Also, as the focus is only on the development activities, not all of the principles are discussed. Some of the principles fall into the scope of Project Management or are relevant to the whole process not to the particular activity. Two issues which cross-cut the activities, but which turn out to be of big importance when discussing the activities, are discussed in section 5: number of customers and test-driven development. That part of the study should be seen as more speculative.

Due to space limitation, this paper does not list the full per-principle tables, but reports the main findings, i.e. contradictions, solutions, considerations, and notes on application; for all

[4] http://www.agilemanifesto.org/
[5] http://www.dsdm.org/version4/2/public/

details the interested reader is referred to the above-mentioned report (Krasteva, Branger, Land, 2007).

4 AGILITY IN CBD ACTIVITIES

The section presents our observations when introducing agile ideas in COTS-based development processes. They are presented in terms of comments, contradictions, considerations and applications. Description is organized by process activities and makes explicit distinction between development activities for component creation and activities for development of systems based on components.

4.1 Requirements Specification

Component Development

Contradiction: COTS components should meet the requirements of many different customers and users, while the agile approach assumes that business people from the (singular) customer are heavily involved during the component development process.

Solution: Customer representatives can be used (see "Application" below).

Consideration: As component interfaces cannot be changed very often issues such as backward and forward compatibility and compliance with standards, should be decided early in the requirements phase. The requirement on backward and forward compatibility means that "enough" time should be spent early to predict future changes, in order to make those changes easier and backwards compatibility easier (Examples: file formats, APIs.)

Application: In order to benefit from close collaboration with the business, as agile approach suggests, some additional steps for identifying a 'proxy' customer of a component should be done (this is further discussed in section 5.1). However, the requirements are identified to satisfy as many different business scenarios as possible. The overall requirements specification is needed in advance so that component interfaces are not changed during subsequent releases of a component. Non-functional requirements for a component are specified and addressed along with functional ones.

System Development

Component selection is a basic part of the process of system development with components. It is performed in parallel with the system activities and includes requirements specification, design, implementation, testing and integration. In the current study the discussion of component selection process is presented along with requirements development activity. It is further extended in section 5.

Contradiction: The contradiction that exists in the context of requirements specification is about the responsiveness to change and the possibility to introduce change late in the development process of component-based systems. Requirements for systems based on components should be pretty well defined in advance. The reason for this is that changing a COTS component is a very hard task. COTS are delivered to the team as a black-box, sometimes without source code and often without detailed specification. The way of changing a component (if possible at all) is to contact the supplier. This includes sending a mail with the proposed changes, waiting for a response, meetings with the supplier, negotiating schedule and costs, etc., which can significantly disturb the development process. That is why introducing changes in the requirements of a component-based systems involves either reconfiguration of components or replacing components. However, both activities are limited to the extent they can actually meet the changed requirements.

Solution: Although a significant part of the overall requirements specification should be done in advance, the processes of requirements elicitation and component selection are very liable to applying agile principles. Initially requirements should not be specified in too much detail, because it is practically impossible to find a component which fulfils all requirements. Instead, the requirements are refined in more detail iteratively during component selection and evaluation. Through "gap analysis" (Ncube and Dean, 2002) along with the customer the component which gives the most and leaves the least (in terms of effort and cost) is identified.

Application: Agile ideas benefit the requirements engineering activity for component-based systems mainly by their adoption in the processes of requirements elicitation and component selection. Two directions for application can be mentioned: customer participation and iterations. Close collaboration with the customer would assure that difficult decisions and trade-offs that should be made during the selection process as well as the appropriate requirements prioritisation are based on business value. An iterative selection process supports a fertile dialog with the customer and makes the requirements elicitation and selection process manageable and observable.

During the requirements specification process of systems based on components, the architectural and

business requirements should be considered along with the functional and non-functional ones.

An approach that adds additional value in requirements specification of component-based system is prototyping. Prototyping is a common practice for identification and clarification of customer requirements. Prototypes of systems that are based on components are easily produced. However, some policy issues, such as trial versions, should be considered.

4.2 Design

Component Development

Contradiction: The biggest contradiction in component design and agile design activities is about the simplicity and generality. Additional decisions should be made and considered during component design specification. Some such decisions are about component interfaces: how should functional interfaces be specified and designed to be as reusable as possible? How can the number of interfaces that requires functionality be minimized to make the component as independent as possible? What configuration interfaces are needed in order to support the adaptability? In addition, the component interfaces should change as little as possible from one version to another so they are usually specified in the very first versions of the component.

Another thing that should be considered during the component design is about the component technology and supported standards. Component has to provide support for both the common standards and the evolving ones in order to stay competitive on the market.

Furthermore, COTS complexity and additional non-functional requirements involve a more formal approach to design and architecture than currently suggested by agile methods.

No solution: Additional considerations about component reusability, independence, adaptability and dependability should be done when designing a component. The overall component design should be specified in advance so that all the interfaces are kept the same during subsequent versions of the component.

System Development

Application: Involving business people in design activity for COTS-based systems is easily achievable as the architecture specified by means of components, is enough simple and understandable for all involved project stakeholders (Stojanovic, 2003). An important goal for the design is to minimize architectural mismatch (Crnkovic, 2002), which can be done by considering compatible sets of components as candidates. Similar to requirements, replacement of components comes with a high cost in terms of required redesign and reimplementation. So developers must predict enough of future changes to select future-proof components.

4.3 Implementation

Component Development

No particular constraints and restrictions exist in applying agile principles to implementation activity of components.

System Development

Comment: The coding activity when developing a system with components is reduced by almost 50% (Crnkovic, 2002). It involves adapting components and writing wrappers and glue code, thus building component assemblies to provide system functionality. The integration process is central when a system is developed out of components (Larsson, 2007) and should be an integral part of the component selection process. It is possible that components need to be reconfigured when a new component is added to the integrated system. Issues with component integration exist even in run-time, when components are added dynamically to the system.

Application: Agile principles for early and continuous delivery of working software are supported to a great extent of the characteristics of implementation activity of COTS-based systems. Part of component integration is performed during component selection process. Reduced coding time allows for early receiving of feedback. Such an implementation process supported by automated tests can be very beneficial as it would assure safe reconfiguration of the system.

4.4 Verification & Test

Component Development

Consideration: An additional complexity is that of verifying the component in the absence of a context (Fredriksson and Land, 2007; Alvaro, Land, and Crnkovic, 2007), which is fundamental for the idea of certification of components by independent third parties (Alvaro, Almeida and Meira, 2005).

Consideration: One important part of component verification in absence of a system context is verification of its "integrateability", i.e. the accompanying documentation of interfaces, standard compliance, and perhaps illustrating its possible

usage by shipping it with code and applications which illustrate the possibilities by using the component (while also teaching how to use it).

System Development

Comment: In general ideas are not contradictory and are common for software development. The restriction that exists is that system verification is restricted to black-box testing.

Consideration: Since component behaviour is known by its specification which is not always sufficiently detailed, comprehensive test coverage is not possible for an acquired component. The system developer should focus its test suites on the component features desired and/or used in a system, and in practice extensive test coverage will be achieved only for these features. Since components are constructed to be general and to suit different situations and environments, there will be many features which are thus only partly tested by the system developer

Consideration: Experimentation, formal testing and prototyping would be an excellent way to learn about the component behaviour (especially for non-functional properties), and the tests would then be stored to verify that no (bad) changes has been made when a new version of the component is released (or at least it becomes apparent what the changes are, and you have the choice to adapt your system and use the new version anyway). In this way, the automated tests of component features would be used and re-executed many times during the whole system development process: first created and executed during the selection process, then as part of integration testing and system testing, and then during subsequent iterations (if any) as regression testing.

4.5 Integration

Component Development

Comment: No particular contradictions or constraints seem to exist since component development is not significantly different from common application development as far as integration is concerned.

Consideration: In agile methods integration happens continuously, while when dependability issues are addressed integration process should be controlled.

System Development

There are no restrictions in applying agile ideas to integration activity of COTS-based systems.

5 DISCUSSION

Two important issues which cross-cut the activities are discussed in this section: the number of customers and test-driven development.

5.1 Number of Customers

A fundamental difference in assumptions between agile methods and the usage of COTS is the relation to the customer(s). The agile principles assume that there are one or more customers that initiate the project and for whom the product is created, while COTS products are developed and then offered to an open market with many potential customers. This fundamental difference in the assumptions is the cause of many of the contradictions mentioned earlier, and this needs to be addressed. There are two ways, which can be combined, to alleviate the problem outlined earlier:

- Someone internally, who knows the market well, such as marketing people or domain experts, would act as customer in an agile project.
- The component development organization could involve real customers for e.g. requirements gathering and evaluation of various alternatives early during development.

When developing components for a larger market it is the component vendor who finally defines and prioritizes requirements. The goals are not decided ultimately by the customers (or their representatives) but by the one that develops the component. The component developer should of course listen to the customer but it is not the customer who makes the decision. This also means that the contract between component vendor and customer representatives has to be different. Some open questions that have to be answered are: Can a component developer require an on-site customer? Who pays who? etc.

5.2 Test-Driven Development and Selection Methods

The general idea of test-driven development (TDD) applied to system development with components would mean that functional tests are specified before implementing a function, after which the glue code for the function is created, followed by tests execution. When changes are made, these tests are used for regression testing. The TDD approach can easily be extended to also include component selection: functional tests are specified together with the customer, in parallel with a search for suitable

components. There are some established component selection methods where the selection is closely intertwined with requirements specification (Alves and Castro, 2004; Chung and Cooper, 2004; Liu and Gorton, 2003; Maiden and Ncube, 1998; Ncube and Maiden, 1999). Selection and design also influence each other in both directions: components must be selected which fit the specified architecture, but the availability of components will influence the design; components can for similar reasons with advantage be evaluated and selected in compatible sets simultaneously (Bhuta and Boehm, 2005; Morisio *et al*, 2002). The evaluation of components need to start with some exploration and experimentation to learn the component, but should then mainly consist of the implementation of the features specified by the functional tests. This ensures that the development efforts are kept focused and that the evaluation is relevant. This applies not only to functional testing but also to quality tests. Performance tests would by construction accurately reflect the usage expected in the real system and it is possible to find the relevant limitations and bottlenecks.

The component evaluation can thus be seen as a verification of the suitability of certain components and a certain design as well as the suitability for implementing the system requirements. Verification of the design includes architectural properties and "integrateability", i.e. how well the components integrate in practice. In this way, it is possible to show something to the customer very early in the process. It also becomes possible to involve the customer in the selection decision, if we have, say, three alternative implementations of the same function to show, which is very much in line with agile principles.

6 CONCLUSIONS

Nowadays agile approach for software development continuously enlarges its area of application. The presented study is a step towards introducing agility into the building of COTS-based systems, by being a systematic, theoretical examination of the applicability of the agile approach to COTS-based development. This examination is organized by mapping agile values and principles to each of the activities of the two development processes: development of systems based on COTS components and COTS production itself. The aim of the paper is to present the picture of agile adoption as completely and thoroughly as possible. However, to cover this broader picture some details have been left out of the discussions; these are published elsewhere (Krasteva, Branger, Land, 2007).

We can expect the findings of the study to be subject to adjustment and refinement (and rejection) as data are collected in empirical studies, which is the next phase of our research within the PROGRESS Centre for Predictable Embedded Software Systems[3] and the ITEA2 FLEXI project[1]. The examination presented in this paper will form the theoretical foundation for these empirical studies; for example, each of the contradictions proposed in this paper are easily transformed to study questions for empirical research, such as "how serious is the contradiction in practice?", and "how can the contradiction be overcome in practice?" Furthermore, the empirical adoption can benefit from analysis on the severity and criticality of the contradictions outlined in the paper and the impact and effectiveness of mitigation activities proposed as solutions.

In addition, the results of the study can be used as a starting point when a company searches for an appropriate agile development process (in terms of lifecycle, products, roles, techniques and application guidelines). Another direction for future work is to include more development activities in the examination, such as maintenance and evolution, and also supporting activities such as project management, configuration management, and documentation. The same type of study can also be made for other business contexts, such as *architecture-driven development* and *product line development* (Crnkovic, Larsson and Chaudron, 2006).

ACKNOWLEDGEMENTS

This work is partly funded by the Swedish Foundation for Strategic Research and Bulgarian Ministry of Education and Science.

REFERENCES

Albert, C., Brownsword, L., 2002. *Evolutionary Process for Integrating COTS-Based Systems (EPIC)*, Technical Report CMU/SEI-2002-TR-005, Carnegie Mellon University

Alvaro, A., Almeida, E. S., Meira, S. R. L., 2005. "Software Component Certification: A Survey", In *31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering (CBSE) Track*, Euromicro

Alvaro, A., Land, R., Crnkovic, I., 2007. *Software Component Evaluation: A Theoretical Study on Component Selection and Certification*, MRTC report, Mälardalen Real-Time Research Centre, Mälardalen University

Alves, C., Castro, J., 2001. "CRE: a systematic method for COTS components Selection", In *Proceedings of the XV Brazilian Symposium on Software Engineering (SBES)*, Rio de Janeiro

Beck, K., 1999. *EXtreme Programming EXplained: Embrace Change*, Addison Wesley

Bhuta, J., Boehm, B., 2005. "A Method for Compatible COTS Component Selection", In *Proceedings of the 4th International Conference on COTS-Based Software Systems*, LNCS, Vol. 3412, Springer

Bowers, J., May, J., Melander, E., Baarman, M., Ayoob, A. 2002. Tailoring XP for large system mission critical software development. In *Extreme Programming and Agile Methods - XP/Agile Universe 2002 Second XP Universe and First Agile Universe Conference*, Lecture Notes in Computer Science Vol.2418, Springer-Verlag

Chung, L., Cooper, K., 2004. "Defining Goals in a COTS-Aware Requirements Engineering Approach", *Systems Engineering*, Volume 7, Issue 1, pp. 61-83, Wiley

Crnkovic, I., Larsson, M. 2002. *Building Reliable Component-Based Systems*, Artech House

Crnkovic, I., Larsson, S., Chaudron, M., 2006. Component-based Development Process and Component Lifecycle. In *27th International Conference Information Technology Interfaces (ITI)*, IEEE Computer Society

Cockburn, A., 2004. *Crystal Clear: A Human-Powered Methodology for Small Teams*, Addison-Wesley Professional

Cooper, K., 2006. Can Agility be Introduced into Requirements Engineering for COTS Component Based Development?, In *International Workshop on Software Product Management (IWSPM'06),* IEEE Computer Society

Fredriksson, J., Land, R., 2007. "Reusable Component Analysis for Component-Based Embedded Real-Time Systems", In *29th International Conference on Information Technology Interfaces (ITI)*, IEEE Computer Society

Greening, J. 2001. Launching Extreme Programming at a Process-Intensive Company, *IEEE Software, Volume 18*, IEEE Computer Society

Heineman, G. T., Councill, W. T. , 2001. *Component-based Software Engineering, Putting the Pieces Together*, Addison-Wesley.

Krasteva, I., Branger, P., Land, R. 2007. *A Systematic Comparison of Agile Principles and the Fundaments of Component-Based Software Development*. MRTC report, Mälardalen Real-Time Research Centre, Mälardalen University

Larsson, S., 2007. *Key Elements of Software Product Integration Processes,* Ph D Thesis, Mälardalen University Press

Liu, A., Gorton, I., 2003. "Accelerating COTS Middleware Acquisition: The i-Mate Process", *IEEE Software*, Volume 20, Issue 2, pp. 72-79, IEEE Computer Society

Maiden, N. A., Ncube, C., 1998. "Acquiring COTS Software Selection Requirements", *IEEE Software*, Volume 15, Issue 2, pp. 46-56, IEEE Computer Society

Manhart, P., Schneider, K., 2004. Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report, In *26th International Conference on Software Engineering,* IEEE Computer Society

Morisio, M., Seaman, C. B., Basili, V. R., Parra, A. T., Kraft, S. E., Condon, S. E., 2002. "COTS-based software development: Processes and open issues", *Journal of Systems and Software*, Volume 61, Issue 3, pp. 189-199, Elsevier

Ncube, C., Dean, J. C., 2002. "The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components", In *Proceedings of the First International Conference on COTS-Based Software Systems*, LNCS 2255, p176-187, Springer-Verlag

Ncube, C., Maiden, N. A., 1999. "PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm", In *Second International Workshop on Component-Based Software Engineering*, Los Angeles

Navarrete, F., Botella P., Franch, X., 2005. How Agile COTS Selection Methods are (and can be)?, In *Proceedings of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'05 ,* IEEE Computer Society

Poppendieck, M., Poppendieck T., 2003. *Lean Software Development: An Agile Toolkit*, Addison-Wesley Professional

Schwaber, K., 2004. *Agile Project Management with Scrum*, Microsoft Press

Sommerville, I., 2006, *Software engineering*, Pearson Education, Eight Edition

Stojanovic, Z., Dahanayake, A., Sol, H. 2003. Component-oriented agile software development, In *Extreme Programming and Agile Processes in Software Engineering. 4th International Conference,* Lecture Notes in Computer Science Vol.2675, Springer-Verlag

Scotto M., Sillitti A., Succi, G., Vernazza T., (to appear) Agile Methods, In *CBSE: state of the art, practices, and future directions,* World Scientific, available at http://www.unibz.it/web4archiv/objects/pdf/cs_library/agilemethods.pdf

Turk, D., France, R., Rumpe, B. 2002. Limitations of Agile Software Processes, In *Third International Conference of eXtreme Programming and Agile Processes in Software Engineering*

Wallnau K. C., Hissam S. A., Seacord R. C., 2001. *Building Systems from Commercial Components*, Addison-Wesley.

Wayrynen, J., Boden, M.; Bostrom, G. 2004. Security engineering and extreme programming: an impossible marriage? In *Extreme Programming and Agile Methods - XP/Agile Universe 2004. 4th Conference on Extreme Programming and Agile Methods.* Proceedings, Lecture Notes in Computer Science Vol.3134, Springer-Verlag