An Approach to Reconcile the Agile and CMMI Contexts in Product Line Development

Fredy Navarrete, Pere Botella and Xavier Franch

Universitat Politècnica de Catalunya {ffnavarrete, botella, franch}@lsi.upc.edu

http://www.lsi.upc.edu/~gessi

Abstract

Software product line approaches produce reusable platforms and architectures for products set developed by specific companies. These approaches are strategic in nature requiring coordination, discipline, commonality and communication. The Capability Maturity Model (CMM) contains important guidelines for process improvement, and specifies "what" we must have into account to achieve the disciplined processes (among others things). On the other hand, the agile context is playing an increasingly important role in current software engineering practices, specifying "how" the software practices must be addressed to obtain agile processes. In this paper, we carry out a preliminary analysis for reconciling agility and maturity models in software product line domain, taking advantage of both.

1. Introduction

The way software products are being developed has changed over the course of the time. Actually, software product development tends to assure the needs of both individual and grouped customers in order to adapt these needs to different market types [1]. Therefore, the ability to reuse and producing customizable software steer the companies to use product line engineering to improve software products development through reuse of systems whose functionalities overlap [2]. A software product line (hereafter, SPL) is a set of products that together are focused in a particular market segment or fulfill a particular mission [3]. The main goals in a SPL are addressed to provide customizable products at reasonable costs to satisfy the needs of the market. To achieve these goals it is necessary to define some important aspects, such as identifying specific business environments, performing the marketplace analysis, and defining a scope for the SPL. Because of these aspects, the SPL emerges by the recognition of different business opportunities having into account the tradeoffs between exploring commonality among software products, architecture-centric development, and two-tiered organizational structure [9]. Consequently, we can consider SPL engineering as a paradigm to develop software applications using platforms and mass customisation [4], which needs a strong discipline to produce products simultaneously, according with company schedule.

Capability Maturity Models (CMMs) [12] contain the essential elements of effective and disciplined software processes. The CMMs have wide acceptance in the industrial environment for their improvement guidelines in the software process. The most recent CMM model is the CMMI (Capability Maturity Model Integration), which has grouped different CMMs models in an integrated approach. Structurally, CMMI is built upon a major organization element, the *process areas*. In the capability context, the process areas refer to "*what to do*" rather than "*how to do it*" [3].

On the other hand, in the software development discipline, some agile methods such as eXtreme Programming (XP) [13], Scrum [14], Crystal methods [15], among others, are generating interest in the industry by the importance of their software development practices, which refer to "*how*" we can drive the software processes to obtain agility. These agile methods have generated controversy in software engineering context, because they propose foundations, processes, and activities to develop software that are different of plan-driven approaches, for instance as CMMs.

For this reason, the purpose of this paper is suggesting an agreement point where we could obtain mutual benefits for using together the maturity models and agile methods, taking advantages of the strengths

of both contexts to apply those advantages in a SPL domain to specify *what we can do* (with CMMI) and *how we can do it* (with the best practices of agile methods).

The structure of this work begins with the identification of the main processes involved in a SPL domain. Then, we study an approach that analyzes the CMMI in a SPL domain. Next, we study the influence of the agile context in SPL processes. Afterwards, we suggest a common point between the CMMI model and the agile methods, with the help of an example applied in a specific SPL process, for instance the selection of Commercial Off-The-Shelf (*COTS*) component. Finally, we provide the conclusions of our work.

2. SPL process and practice areas

In SPL the software-involved processes conform to common platform in which it is possible to build applications using a collection of reusable artifacts. For this reason, the SPL engineering paradigm separates two specifics processes [4, 16]:

- *Domain engineering:* are the set of processes specified to define the commonality and the variability of the SPL. The artifacts produced during domain engineering are interrelated by traceability to ensure the definition of the SPL.
- *Application engineering:* these sets of processes are responsible of reusing the domain components and the artifacts, deriving SPL applications from domain engineering to exploit the commonality and variability of the SPL.

Essentially, these processes are addressed to preserve the tradeoffs between the development of core assets, products development, and management of these developments for the organization benefits [17]. To preserve these tradeoffs it is necessary that any organization must master a collection of activities to carry out successfully the essential work in a SPL (*software engineering practice areas; technical management practice areas;* and *organizational management practice areas*). Based on [3], we describe briefly the main practice areas in SPL context:

- *Software engineering Practice Areas:* embrace all technical activities necessary to create and developing products. These activities are: Architecture Definition, Architecture Evaluation, Component Development, COTS Utilization, Mining Existing Assets, Requirements Engineering, Software System Integration, Testing, and Understanding Relevant Domains.
- *Technical Management Practice Areas:* represents all the management activities that are necessary to support the right way to develop the software engineering activities. These activities are: Configuration Management, Data Collection, Metrics, and Tracking, Make/Buy/Mine/Commission Analysis, Process Definition, Scoping, Technical Planning, Technical Risk Management, and Tool Support.
- *Organizational Management Practice Areas:* its responsibility is addressing the organization around the SPL processes coordinating the management activities. These activities are: Building a Business Case, Developing an Acquisition Strategy, Funding, Launching and Institutionalizing, Market Analysis, Operations, Organizational Planning, Organizational Risk Management, Structuring the Organization, Technology Forecasting, and Training.

3. Process maturity in SPL

The lack of maturity involved in the software processes has a negative impact on the successful development of SPL, because a good definition of software processes is necessary to help modelling the variability in a SPL. For this reason recognising CMMs models in SPL life-cycle could help us achieving a strategic discipline to address the processes improvement in SPL. Indeed, we must take into account the disciplined processes to provide the foundations and attain predictability and quality [7]. The CMMI model is defined in [5] as a "*process improvement approach that provides organizations with the essential elements of effective processes*". Like previous models [12], CMMI provides guidelines to specify "*what*" software process should possess. The CMMI provides the ability to generate multiple models that may reflect contents from different bodies of knowledge (e.g., systems engineering, software engineering, Integrated Product and Process Development) [6]. CMMI models have two structured representations: *staged representation* and *continuous representation*. These representations differ in how they organize the *processes areas*. These process areas are a set of related activities that are performed together to achieve the specific and generic goals.

Currently, some organizations applying the CMMI over the SPL domain to provide process improvement in their software processes [7]. We use the Table I which is extracted from [3], to present the influence of CMMI processes areas over SPL practices areas, where the process areas define where an organization should have processes, while the practices areas describe where an organization should have expertise [3].

Product Line Practice Areas	CMMI Process Areas
Software Engineering Practice Areas	
Architecture Definition	Technical Solution
Architecture Evaluation	Verification
Component Development	Technical Solution
COTS Utilization	Supplier Agreement Management
Mining Existing Assets	(none)
Requirements Engineering	Requirements Development
Software System Integration	Product Integration
Testing	Verification/Validation
Understanding Relevant Domains	(none)
Technical Management Practice Areas	
Configuration Management	Requirements Management/ Configuration Management
Data Collection, Metrics, and Tracking	Measurement and Analysis/ Project Monitoring and Control/Integrated Project Management
Make/Buy/Mine/Commission Analysis	Decision Analysis and Resolution/ Supplier Agreement Management
Process Definition	Organizational Process Definition
Scoping	(none)
Technical Planning	Project Planning
Technical Risk Management	Risk Management
Tool Support	(none)
Organizational Management Practice Areas	
Building a Business Case	(none)
Customer Interface Management	(none)
Developing an Acquisition Strategy	Supplier Agreement Management
Funding	(none)
Launching and Institutionalizing	(none)
Market Analysis	(none)
Operations	(none)
Organizational Planning	Project Planning
Organizational Risk Management	Risk Management
Structuring the organization	(none)
Technology Forecasting	(none)
Training	Organizational Training

Table I. *Associations between Product Line Practices Areas and CMMI Process Areas, taken from [3].*

Although the impact of process areas is not direct over a SPL, because some process areas do not cover the same ground of practice areas, CMMI is able to provide guidelines to improve the process discipline that can steer the SPL development. For example in [7], CMMI was adopted in a SPL environment obtaining an important foundation for SPL practices.

4. Agile methods in SPL context

Agile methods are proposed nowadays as a way to support software systems procurement. The agile context has had an increasing role in the practices of software engineering [8]. The starting point of agile methods was the "Agile Manifesto" [28]. Agile methods have emerged in software engineering for some important reasons: traditional methodologies like plan-driven methods are much automated to be used with a lot of detail, transforming them into a fictitious image seeking control over software processes [18]; there are a lot of standards and methods in software engineering that are not applied by the industry for ignorance, for their difficulty to be implemented or because they do not represent the reality of the organizations [19]. Therefore, agile methods have generated a wide debate for the controversy of their foundations, some important argued subjects are: the tacit knowledge [20, 25], innovation in agile methods [21], misconceptions about agile methods [22, 23, 25], among others. Beyond these controversies, the agile methods have gained in a few years a wide acceptance in industrial environment, and some software specialists have recommended their use [26, 27], specially because the agile methods suggest the *best practices* to specify "*how*" the software development could be driven to obtain agility.

In the SPL domain some important agile aspects may be considered, such as: the need of division of work in a SPL oblige us to consider sharing the knowledge between different disciplines involved [29]; there are not specialized techniques for any SPL inspections, reviews, or structured walkthroughs [4]; the need to obtain flexibility in a SPL architecture that may be adaptable either to different customers requirements or different software components (like COTS components) [3]. These aspects could be supported by agile methods, because these agile approaches have practices based on time-boxed iteration, evolutionary development, adaptive planning, evolutionary delivery, and inclusion of other values and practices that encourage agility in software development context [30]. In addition, the SPL may take advantage of three important aspects that define the agility to affront the SPL variability: creating and responding to change, being nimble and able to improvise, and balancing flexibility and structure [31].

Although there is not a lot of literature analyzing agile values and principles in the SPL context, we must take into account some important aspects of SPL before accepting agile foundations, such as: the SPL tends toward a long-lived life-cycle, for this reason to maintain the information is necessary; the conceptual integrity in a SPL is very important, for this aspect the requirements of customers specifics may affect it; the SPL targets satisfying the needs of various customers

rather satisfying the individual needs; among others. These aspects help us to evaluate the influence of agile principles over SPL practice areas. Therefore, in Table II we have analyzed this influence, representing it with a plus sign if the agile principle has a positive impact in the practice area, or with a less sign to represent the negative impact of agile principle over the practice area; besides, the zero number represents the absence of this principle over the practice area. In this analysis it is possible to find together both the plus and less signs indicating that the same principle may have contradictory effects. We identify the agile principles in the columns with a capital letters, and identify SPL practice areas in the rows.

Product Line Practice Areas	Agile Principles											
	A	B	C	D	E	F	G	H	I	J	K	L
Software Engineering	+ —	+ —	+	+	+ —	+ —	+ —	+	+	—	+ —	+ —
Technical Management	+	+ —	+	+	+ —	—	+ —	+	+	—	+ —	+
Organizational Management	+	+	0	+ —	+ —	+ —	0	+	+	—	—	+

Table II. *Impact of agile principles into Practice Areas.*

Next, we describe briefly the main outcome of our analysis by each agile principle over SPL practice areas:

A *Our highest priority is to satisfy the customer through early and continuous delivery*
- *"—"* in SPL there is not a unique customer to be satisfied,
- *"+"* but is important that the SPL can be driven to satisfy the multiple variability of grouped customers.

B *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage*
- *"—"*in some practice areas the requirements are freezing, e.g. in the core asset design,
- *"+"* but the architecture must be flexible to support the changing requirements of grouped customers to develop a product-specific.

C *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale*
- *"0"* this principle focuses in the software development,
- *"+"*although some practice areas need to develop software for a successful integration of COTS components (e.g., *glue code*) and to deliver specific software products.

D *Business people and developers must work together daily throughout the project*
- *"+"* many disciplines must work together in a SPL, to obtain a knowledge shared between the teams,
- *"—"* but the geographic distribution of SPL teams can avoid a fluid communication.

E *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done*
- *"—"* the SPL is market-driven, for this reason, it is not easy to embrace individual expectations of specific customer to build a SPL project,
- *"+"* although all practice areas needs well formed teams to achieve the goals project.

F *The most efficient and effective method of conveying information to and within a development team is face-to face conversation*
- *"—"* in a SPL it is common that different teams works in separate places,
- *"+"* but for the successful develop of practice areas it is necessary to have into account flow communication.

G *Working software is the primary measure of progress*
- *"0"* this principle focuses in software development,
- *"+"*although in SPL, delivering software for product-specific it is necessary,
- *"—"* but maybe, would be necessary using others measures in specifics practice areas to obtain the project progress.

H *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely*
- *"+"* in SPL context many disciplines and roles have to communicate between them.

I *Continuous attention to technical excellence and good design enhances agility*
- *"+"* the technical excellence in SPL process helps to achieve well-defined processes for the domain engineering and application engineering.

J *Simplicity--the art of maximizing the amount of work not done--is essential*
- *"—"* the SPL process embraces complex activities for this reason it is difficult reconciling simplicity with the practice areas.

K *The best architectures, requirements, and designs emerge from self-organizing teams*
- *"—"* a good definition of SPL structure depends on a lot of factors behind the self-organizing teams,

- *"+"* but practice areas need well-formed teams to define the main aspects to develop the SPL successful.

L *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*

- *"+"* the SPL involves different disciplines that need shared knowledge, the regular reflections can improve the behaviour of SPL teams,
- *"—"* but is not clear that the regular reflections can be applied over all SPL process.

The important thing with this analysis is how we can apply in an appropriate way the agile principles over SPL context, because some agile principles seem impact negatively or positively. Apparently, the best way to apply them depends on suitable or unsuitable of the principle at the moment of SPL processes development.

5. Common points between agile and CMMI contexts

In previous sections we have studied the agile and CMMI contexts applied in the SPL context. In this section, we suggest a balance that shows a situation which help us to determine an agreement point of reconciliation among the necessary discipline to develop SPL processes and the necessary activities to carry out agile processes, so we can define "*what we can do*" and "*how we can do it*" to develop successfully the SPL processes, taking advantage of CMMI and Agile contexts.

These two contexts have generated controversy in software development [21]. In section 4, we have introduced some subjects of this controversy. Next, we describe briefly some specific subjects about CMM: in CMM models, the people who work in the project development should make an effort to practice and to achieve skills which will be institutionalized by the organization, forcing them not to pay attention in the tasks and needs of the project, to pay attention in objectives and practices that have not been carried out still, besides, we need in the processes development to have many candidates practices rather that bureaucratic and fixed practices [32]; or some authors point out that the CMMI model help us to manage the bureaucracy and boilerplate with its emphasis on risk management and integrated teaming [27]. Beyond these controversy subjects that have been raised for these contexts [20, 21, 22, 23, 25], some authors are seeking the right way to work together with CMMs and Agile contexts, where it is possible to take advantage of two contexts. For example, Paulk analyzes XP from CMM perspective [31], he highlights the discipline and effectiveness of some XP practices; or Boehm, and Turner suggest to identify 5 critical dimensions (size, criticality, dynamism, personnel, and culture), that can be used to describe an organization or a project in terms of its agile and plan-driven characteristics [33].

In a SPL the practices areas are strategic-driven, so they require coordination, discipline, and commonality of an approach than a more independent effort [7]. If we are able to steer these dependencies properly, we can obtain high quality, which is the key to high speed [10]. For this reason we may use the CMMI organization, because it contains a set of standardized processes that refer to the organizational maturity level and includes management, techniques, and support for the organizational processes [11] specifying what we must do to address the needs of different projects. Furthermore, we may aim to take from the agile context its three main basic aspects such as project management, collaboration between stakeholders, and technical excellence [24] to suggest agile practices to provide a way to specify how the standardized processes can be driven.

In Figure I, we can observe a possible agreement point to work together with the disciplined processes (using CMMI), and the best practices to drive a main practice areas of SPL (using agile practices). This agreement point suggests a balance among the agility and discipline that may be achieved through improvement is provided over the SPL processes.

For this reason, this agreement point seeks to be complemented with two important dimensions that influence any development methodology which takes into account the system criticality of SPL processes (system criticality such as: lost of comfort, lost of discretionary money, lost of essential money, lost of lives) and the number of people that play a role the project [15]. With these two dimensions we would be able to regulate the necessary discipline inside the processes SPL, using practices that may be adjusted to the specific needs of the SPL projects, because the number of people that participate inside the SPL project and the criticality of SPL processes help us to apply more or less discipline depending on SPL processes ceremony.

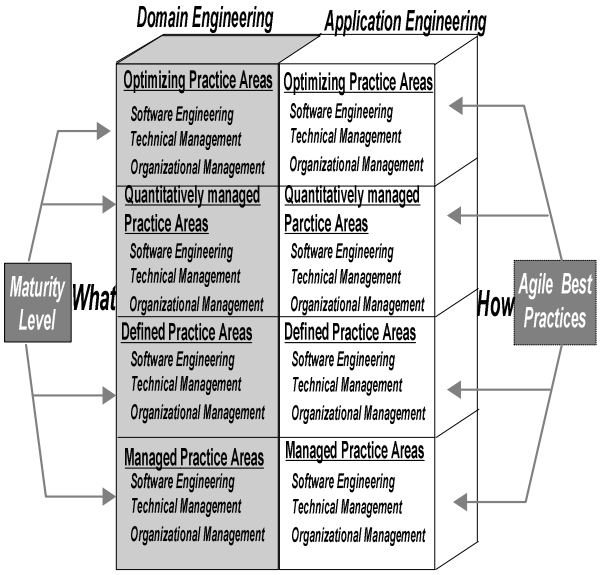

Figure I. *Maturity levels and best practices applied in a product line practice areas.*

This balance may be regarded in one particular SPL activities, namely acquiring or licensing a COTS component from the marketplace to be integrated into a SPL. In Table I we may see the importance and influence of COTS components over the practice areas of SPL, because COTS components selection demand new activities and processes that are different from software development processes. On the other hand, the formality and discipline that should be applied during the selection of a specific component varies according to component criticality and according to its impact over processes ceremony. For example, if we consider integrating two new components into a SPL that develops a family of products for decision making based on organizational information, one for the financial management of data, and another for the management of the organization news, the necessary degree of ceremony to acquire these components can vary, according to the number of people that participate during the selection and according to processes criticality. In Figure II, we may observed that the tool for financial management (represented with the black box) needs for their selection and integration between 7 - 20 people due to the criticality level of the financial tool, on the other hand, the tool for news management (represented with the grey box) needs less staff because it implies a smaller effort and a smaller criticality level than the financial tool. We can evaluate with this identification over which tool we need more planning, more qualified personal and less ceremony for SPL processes development.

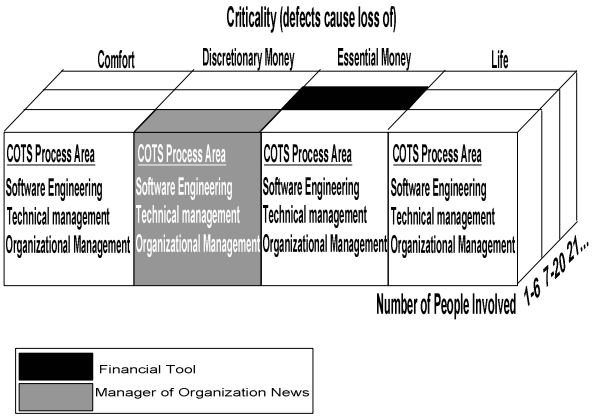

Figure II. *Maturity levels and best practices applied in a product line practice areas, based on [15].*

6. Conclusions

The processes involved in SPL development require activities and roles to apply coordination, discipline, and commonality, besides it is necessary to share the knowledge generated among different disciplines that participate to attain SPL, with the purpose of creating a family of products that satisfies the necessities of grouped customers.

This study analyzes the influence of agile and CMMI contexts over SPL processes, which have generated controversy inside the software engineering, with the purpose of suggesting an agreement point of reconciliation and balance among the necessary discipline required to develop a process SPL, and the agility that we are able to provide to develop a SPL. For this reason, we seek to take advantage of the discipline proposed in CMMI, and the agility of the best agile practices, to identify a point of balance that define the number of people involved in the SPL process development and the system criticality that should be having into account at the moment to carry out a SPL [15]. These contexts can be applied over SPL development in a suitable or unsuitable way, having into account the necessity of ceremony or formality that are required in SPL processes, helping us to define *what we can do* and *how we can do it* to develop a SPL satisfactorily.

Currently, the CMMI application over the SPL context has been studied by the Software Engineering Institute (SEI) reporting satisfactory results of different SPL projects where the CMMI model was applied inside the projects development, e.g., [7]. Moreover, we have not found literature that analyzes the agile methods over the SPL context in a similar way that we have done for COTS selection [34]. For this reason, this study also seeks to be a beginning point to generate

the necessary foundations to consider the inclusion of agile methods inside the SPL context.

References

[1] Ulkuniemi, P., and Seppänen, V., "COTS Component Acquisition in an Emerging Market". IEEE Software (Vol. 21, No. 6) 2004, pp. 76-82.

[2] Bayer, J., Gacek, C., Muthig, D., and Widen, T., "PuLSE-I: Deriving instances from a product line infrastructure". in Proc. Engineering of Computer Based Systems (ECBS 2000) 2000, pp. 237 – 245

[3] Clements, P., and Northrop, L., *Software Product Lines*, Addison-Wesley, 2002.

[4] Pohl, K., Böckle, G., and Linden, F., *Software Product Line Engineering*, Springer-Verlag Berlin Heidelberg 2005.

[5] CMMI Web Page. Software Engineering Institute. Available in: http://www.sei.cmu.edu/cmmi/cmmi.html (Last Accessed: April 2006).

[6] CMMI Team., "Capability Maturity Model® Integration (CMMISM), Version 1.1". Technical report CMU/SEI-2002-TR-011, ESC-TR-2002-011. 2002.

[7] Jones, L., and Northrop, L., "Product Line Adoption in a CMMI Environment". Technical report CMU/SEI-2005-TN-028. 2005.

[8] Cockburn, A., and Highsmith, J. "Agile Software Development: The People Factor". IEEE Computer, December 2002.

[9] McGregor, J., Northrop, L., Jarrad, S., and Pohl, K., "Initiating software product lines". IEEE Software (Volume 19), 2002, pp. 24 – 27

[10] Martin, C., *Agile Development: Principles, Patterns and Process*. Prentice Hall, 2002.

[11] Ahern, D., Clouse, A., and Turner, R., *CMMI® Distilled: A Practical Introduction to Integrated Process Improvement*. Addison Wesley, 2003.

[12] Paulk, M., Curtis, B., Chrissis, M., and Weber, C., "Capability Maturity Model SM for Software Version 1.1", Technical Report CMU/SEI-93-TR-024, ESC-TR-93-177, 1993.

[13] Beck, K., *Extreme Programming Explained: Embrace Change,* Addison Wesley, 1999.

[14] Schwaber, K., "The Scrum development process". in Proc. OOPSLA '95 Workshop on Business Object Design and Implementation, Austin, 1995.

[15] Cockburn, A., *Crystal Clear. A human-powered methodology for small teams, including The Seven Properties of Effective Software Projects*. Addison Wesley 2002.

[16] Linden, F., "Software product families in Europe: the Esaps & Cafe projects". *Software IEEE (Volume 19)*, 2002. pp. 41 – 49.

[17] Clements, P., Jones, L, Northrop, L, and McGregor, J, "Project management in a software product line organization" *Software, IEEE (Volume 22)*, 2005. pp.54 – 62.

[18] Nandhakumar, J., and Avison, D., "The fiction of methodological development: a field study of information systems development". Information Technology & People, 12(2), pp. 176-191, 1999.

[19] Wiegers, K., "Read my lips: no new models!". Software, IEEE Volume 15, 1998 pp. 10 – 13.

[20] Boehm. B., "*Get ready for Agile Methods, with care*". Computer (IEEE), pp. 64-69, 2002.

[21] Berard. E., "*Misconceptions of the Agile zealots*". Report the Object Agency, Available in: http://www.svspin.org/Events/Presentations/MisconceptionsArticle20030827.pdf, 2003. (Last Accessed April 2006).

[22] Jeffries, R., "Misconceptions about XP" an Agile Software Development Resource January 2002. Available in: http://www.xprogramming.com/xpmag/Misconceptions.htm (Last Accessed January 2006)

[23] Fowler, M.: "*Is design dead?*" XP2000 Proceedings. Available in: http://www.martinfowler.com/articles/designDead.html, Last Significant Update: 2004 (Last Accessed April 2006).

[24] Tate, K., *Sustainable Software Development: An Agile Perspective*. Addison Wesley Professional, 2005.

[25] McBreen, P., *Questioning Extreme Programming*. Addison Wesley, 2003.

[26] Jacobson, I., "A resounding Yes to Agile Processes – But also to more". Cutter IT Journal January 2002.

[27] DeMarco, T., and Boehm, B., "The agile Methods Fray". IEEE Computer, Vol. 35, No. 6, 2002, pp. 90-92.

[28] Beck, K., and et al., "Manifesto for Agile Software Development". Available in: http://www.agilemanifesto.org, 2001 (Last Accessed April 2006).

[29] Vesiluoma, S., "Ways of Knowledge Sharing in Agile and Product Line Based Software Development" in Proc. 27th Information Systems Research Seminar in Scandinavia, 2004.

[30] Larman, C., "Agile & Iterative Development. A Manager's Guide" Addison-Wesley, 2004.

[31] Marchesi, M., Succi, G., Wells, D., and Williams, L., *Extreme Programming Perspectives*. Addison Wesley, 2002.

[32] DeMarco, T., and Lister, T., *Peopleware—Productive Projects and Teams*, 2nd Ed., Dorset House, 1999.

[33] Boehm, B., and Turner, R., "Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods". in Proc. of the 26th International Conference on Software Engineering (ICSE'04), 2004.

[34] Navarrete, F., Botella, P., and Franch, X., "How Agile COTS Selection Methods are (and can be)?", 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 30 August - 3 September, 2005, Porto, Portugal, pp. 160-167.