An Industrial Case of Exploiting Product Line Architectures in Agile Software Development

Muhammad Ali Babar1, Tuomas Ihme2, Minna Pikkarainen2,
1Lero, University of Limerick, Ireland, 2VTT, Oulu, Finland,
1malibaba@lero.ie, 2{Tuomas.Ihme, Minna.Pikkarainen}@VTT.fi

Abstract

There has been an increased interest in exploring the ways of integrating agile software development and software product line approaches. Both approaches share several common goals, which provide the motivation for integrating them. However, there has been little empirical research for understanding how these approaches can be integrated in industrial settings. This paper presents the findings from a case study of a software development company that has successfully integrated software product line architecture and agile software development practices. The company’s processes are described based on product line and agile practices. The results are expected to provide useful insights into the mechanics of exploiting product line practices in agile software development despite apparent philosophical clashes between the two approaches.

Keywords: Software product lines, Agile approaches, software architecture, empirical research

1. Introduction

Agile and Software Product Lines (SPL) software development paradigms are being promoted as means of reducing time to market, increasing productivity, improving quality and gaining cost effectiveness and efficiency [1]. Both approaches assume that requirement changes occur and be managed appropriately [1]. Though, both approaches apply different means of achieving the promised benefits, there is a clear motivation to combine software product line engineering and agile software development because both paradigms appear to possess complementary properties.

These approaches are considerably independent of each other and they have few conflicts [2]. However, there appears to be several challenges involved in integrating Agile and SPL practices because of the philosophical differences namely upfront design and change management strategies [1, 3]. Moreover, agile software development approach do not purport to develop flexible artefacts for reuse [3, 4] or pay sufficient attention to documentation for maintenance and evolution as required by SPL [4].

Despite the perceived incompatibilities and potential challenges, there has recently been an increased interest in exploring the possibilities and prerequisites of combining agile and product line approaches [2, 5-9]. We assert that one way of helping researchers and practitioners in gaining insights into the mechanics of successfully integrating agile and SPL approaches is to gather and disseminate empirically found evidence. However, there has been little empirical research in this area. The goal of the research reported in this paper is to increase the empirically founded understanding of combining SPL and agile approaches. This study is one of our initial efforts towards achieving that goal. As such the objective of this study is to empirically study organizational processes aimed at integrating SPL approach and agile practices. In this paper, we present our findings from a case study of Finnish company that has successfully integrated SPL and agile practices in order to leverage product line architectures for improving its agile software development.

The structure of the paper is as following. Section 2 provides an overview of related research on integrating agile software development and software product line practices. Section 3 presents research method used and procedures followed as well as analysis approach used. Section 4 presents the findings of this study whereas Section 6 concludes the paper with final remarks.

2. Research Context and Objectives

One lesson from the history of software architectures, components and reuse is that it is profitable to focus on narrow application and technology domains or sub domains [10]. These observations have led to domain specific software architectures and product line approaches [11, 12]. The core asset development, product development and management form the triad of essential software product line activities [12].

Another lesson is that a useful component collection has to be small, often only 100 components, so that library retrieval systems are not required [10]. In addition, the adoption time and effort of the heavyweight approaches of product line pioneers have become a significant adoption
Both agile and product line approaches promote collaboration [8]. However, agile approaches emphasize the collaboration between customers and developers whereas product line approaches expect collaboration between core asset builders and product developers. Agile software development principles\(^1\) even encourage changes in requirements. Product line approaches also accept anticipated changes in requirements and they are in a position to accommodate unanticipated changes [8]. The scope of a product line identifies those entities that are within the context of a product line [12]. Agile project teams work within an implicit project scope defined by customers but the scope of each iteration cycle is explicit. Agile approaches emphasize the importance of producing working software since the first iteration cycle. Product development teams in a product line produce working software early as well by assembling and configuring core assets [8].

Despite the obvious synergies between agile and software product line approaches, their competing philosophies can make their integration difficult [6, 8]. Product line engineering can be proactive, reactive or extractive [13], but its product development activity is proactive when a product team creates product-specific solutions by specializing a product line platform. On the contrary, agile methods such as eXtreme Programming (XP) [17] and Scrum [18] are reactive and the architecture is expected to emerge in daily design of a system. Software architecture is a key factor in a software product line’s success [15] whereas agile approaches (e.g., XP and Scrum) are known for paying very little attention to software architecture.

The findings of a recent case study of Hanssen and Fægri [2] indicate that software product line engineering and agile software development can work together and complement each other. Motorola’s tailored version of XP [19] includes the development of a baseline architecture that provides sufficient guidance to support the creation of core assets. However, the description of such architecture may have less detail than typically produced architectures for a product line. Agile methods weakly support cross-team communication problems. Nokia’s solution to this problem is to minimize the need for cross-team communication [19]. For example, each product team usually has its own interface with the core asset team. Each product team decides which agile processes and practices are suitable to its needs independent of the teams of other products and core assets.

This paper makes the following two contributions to the software product line research and practices:

- It presents the methodological details and results of an empirical study aimed to find some answers to a question on which two different communities are converging.
- It provides information about and insights into the processes and practices of a company who has been leveraging product line architectures for improving its agile software development.

3. Research Methodology and Procedure

This section describes the methodology and data analysis approach and procedures used for the reported research.

3.1 Research method

We used a case study method for this research. The case study research method is valuable in the situations where researchers’ intention is to understand phenomena [20] in the complex, real life context [21]. The case study has been said to be useful especially when context and actors of the organization are critical for the implemented study [20]. Typically, the selection of a case to be studied is done using some specific context factors [22]. The case of our study was selected because the studied company had been using agile practices and principles (i.e., XP and Scrum methods) since 2005. Recently, they also have introduced a product line approach in order to leverage the architectural level reusability for improving agile development processes. The case study research was designed and implemented based on Yin’s [21] steps for conducting case study research.

3.2 Case context

The company selected for this research is a market leader of specific products that are produced for dynamic, global market environment. It is a large enterprise that has the key development group in Finland and market offices all over the world. It produces services for the global market in more than 90 countries. The company has deployed agile methods and practices since 2005 due to a need to respond to the changing market situation and to deliver products to the market fast and efficiently.

3.3 Source of data

This study is based on the four years’ (2005-2008) continuous collaboration between the case company and the research team. This longitudinal collaboration provided the researchers with several opportunities to gain insights into the required background and domain information for selecting an appropriate case and gathering and analyzing the relevant data. However, the findings reported in this paper are mainly based on the analysis of the data gathered using focus group sessions [23] and one workshop organized for sharing the findings.

\(^1\) http://Agilemanifesto.org
of the focus group discussions. The focus group sessions were designed to gain understanding of the mechanics and practices of the company’s software development process that exploits product line architectures in agile development by exploring the experiences and views of the software architects and managers. Hence, all the relevant software architects and managers from the same site of the company working with the same product line platforms participated in the focus group meeting.

As a secondary data source, we also reviewed various documents such as product roadmaps and development process descriptions, as well as the data gathered on the agile development practices from the same company by other researchers in VTT, Technical Research Centre of Finland.

3.4 Focus Group

As previously mentioned, we used a focus group as the main data gathering approach for the findings reported in this paper. The focus group is considered a proven technique to obtain the perceptions, views, and experiences of a group of selected people on a defined area of interest [24]. Our focus group intended to gain a deep understanding of the processes and practices of combining product line architecture and agile software development by exploring the experiences and views of the software development practitioners from the studied company. Focus group sessions are well planned and structured discussions involving 3 to 12 participants. Focus group discussion is largely free-flowing, but discretely guided by a moderator, who is responsible for keeping the group discussion focused on relevant topics and make sure that everyone has an opportunity to participate. Focus group discussion enables a researcher to explore the way people feel and think about the issues to be studied [25].

Compared to other qualitative data collection methods (e.g., interviews), focus group research can generate candid and insightful information inexpensively and faster. However, focus group research also has several weaknesses such as subjective self-reported data based on personal opinion and interpretation of a particular event/situation, biased moderation, and small sample size, which makes generalizing the results difficult [26, 27]. To combat this, we followed a number of practices as recommended in various books on focus group methodology such as Miles & Huberman [28]. These include structured questions, allocated time for each participant, transcription of sessions, and external checks of the coding labels used by the researchers.

3.5 Study procedure

Our study design intended to organize separate focus group sessions for the two main groups namely technical (i.e., software architects) and management (i.e., project or team managers). Three software architects (one for each of the platforms in the product line) participated in the first one focus group session and five managers (project and team leaders) participated in the second focus group session. Each session started with a brief introduction of the participants and researchers. The introduction included the name of the attendees, their organizations, current position, professional background, experience, and application domain and the type of industry. Each of the focus group sessions lasted for approximately two hours. The sessions were audio recorded with participants’ consent. One of the researchers also took extensive notes. The focus group sessions were held in October 2008. The feedback workshop was organized in February 2009.

3.6 Data analysis

The data analysis step involves transcribing the recorded discussion using appropriate coding schemes. The transcribed data can be analyzed using one or more of the qualitative data analysis techniques reported in [27]. The focus group sessions of this study resulted in approximately four hours of audio recording and extensive notes. The audio recording of both sessions were transcribed by a professional company. One of the researchers read the transcription and compared it with the notes taken during the focus group sessions. During the data verification step, some missed obscure words were corrected. After this stage, we performed content analysis of the gathered data. As proposed by Benbasat et al. [20], the data analysis was carried out by two researchers working together in order to enable the capture of greater richness, validity and accuracy of the data. The results especially from the viewpoint of combing SPL and agile software development were discussed and verified in the feedback workshop with the participants of the focus group sessions.

4. Results and Discussion

This section presents and discusses the results from our analysis of the data gathered during the focus group meeting discussions and project documents and artifacts. The results presented in this paper particularly focus on the part of the discussion that addressed the SPL and agile software development.
4.1 Demographics

Three software architects, each from one of the three platform teams, who had worked as a software architect on 27 projects and had two to eight years of experience in designing software architectures for the similar kinds of systems. The five managers had an average of 15 years of experience of developing software and had worked on several dozens of projects in technical as well as managerial roles. At the time of the study, they had various roles in the company including the director of agile practices, team manager, and project manager. Almost all of them had held the role of Scrum master since 2005 when the company introduced agile practices. All the participants were purposefully selected to maximize diversity of experiences and views. Many of the participants had worked extensively with both agile and plan-driven software development approaches. The selection of these participants was intentional, to allow informed reflection and comparison between the processes and practices being used before and after introducing the use of product line platforms for supporting agile baseline projects. However, this study just focuses on the current processes and practices that have been institutionalized for support product line and agile approaches.

4.2 Product line platform

Figure 1 highlights the role of the assets of the company’s product line platform in agile software development including the exploration and product development phases. Technology roadmaps and target domain analyses are the sources of architectural requirements for product roadmaps and product backlogs. Our investigation has found that each technology roadmap has a time span of about one to two years. As shown in Figure 1, the company’s product line consists of two main sub-systems: mobile clients on different mobile platforms and the Services Platform (background). The architecture of mobile services includes both the server and the client parts of the system. The end users (consumers) see and use only the clients. The service platform serves corporations, operators, and internal sales. Hardware architecture underlies software architecture. The client teams in the company are organized based on operating system (i.e., platform) and platform teams are organized based on the channel it serves.

Figure 1. Agile software development using a product line platform
In order to facilitate the communication and coordination among teams, a common integration document is used for documenting the interfaces between the Backend system and client subsystems as well as for interfaces for external systems. The document is reviewed by all the involved teams in joint meetings and it is updated by the responsible team when needed. The changes in the integration document are communicated to all other teams and need to be accepted by all teams. This integration document is also delivered to the customers. That means customers have a very good knowledge about the team organization for each of the platforms that usually underpins the services or solution provided to them.

The architectural overview model is another key artifact to support the utilization of product line platforms. This artifact contains an overall architecture and a high level architectural skeleton model about the whole system. The overall architectural skeleton has been very well specified but all internal parts of the skeleton have not been described in detail in order to provide maximum flexibility to the agile baseline projects teams. Each platform team maintains the architectural document of the sub-system of the team. The architectural artifacts allow baseline project teams to delay design decisions about connections between the architecture and software items. During the focus group discussions, the project managers and software architects of all three platforms agreed that the overall architecture, key interfaces and protocols of the product family were quite stable now, after a long-lasting development period.

It was also revealed that retrospective meetings of agile baseline projects are used for identifying and discussing the changes that need to be introduced in the overall architecture. However, such architectural level change requests are not made frequently rather there appears to be process pattern of raising such requests quarterly.

On the technical aspects of the architectural design, it was found that the Model-View-Controller (MVC) and layered architectural styles were the main design strategies applied for designing the architectures of all three main parts of the whole system. Both of the architectural styles are well-known for supporting maintainability and customization. However, we found that the customizations were not visible in the overall architecture.

Feature description documents are used to describe new features and modules and their updates including their effects on the architecture of the system. The goal of feature descriptions is to provide the iteration cycles (i.e., hereafter called Sprints) of baseline projects with pre-planned and clear features and workload estimates. The company has standardized certain coding conventions for all platforms and baseline projects. The code conventions define micro architectures and guide the implementation of the architectures and interfaces. It was also revealed that the company had changed its practices of documenting software architectures twice since the introduction of agile software development along with product line platforms. One of the architects described the changes in the architectural documentation practices in these words: “When we started the Agile, we went the opposite direction. We did not write any documents, and that proved not a very good thing. It’s a danger that the information about key architectural artefacts is only in the heads of the people who are doing the work. But now we, I think there has been about three years we are developing with agile practices, and we have found a kind of way to use the documents the right way. We are writing short descriptions when we need, and we are doing some other documents like feature descriptions, if somebody needs it, but not if nobody needs it.”

4.3 Exploration phase before agile product development

Our study has found that the company has introduced an intermediary phase between the domain engineering and application engineering phases of software product line development. This intermediary phase is called exploration phase. During this phase research projects are carried out in order to carry out most of the upfront architecture design work to tease out the big issues or by developing feature descriptions before the Sprint planning days of product development projects. The study concept is used for smaller research problems. For this phase, there are no standard company policies or processes for designing and evaluating architectures for each of the baseline projects based on the constraints placed by the architectures of the platforms to be used.

For this phase, a product backlog can include prioritized items for about two years. The items include high-level estimates of the size of the items to be delivered by managers and architects. The items also include information about connections to specific product releases and dependencies on other interest groups. Both functional and non-functional requirements are described in feature descriptions, product backlogs and Sprint backlogs.

4.3.1 Agile research projects

Agile research projects have varying goals such as developing a new product or introducing a big feature, or conducting the feasibility study of existing features to be introduced in the new version of the used software platform. The projects have a separate research backlog that is not shown in Figure 1. One of the software architects described the purpose of introducing agile research projects in these words: “We have separate research projects when we have some new idea or solution to be tested or prototyped based on the product line platforms. It’s a separate project, like one person working
for two months, and she or he is not participating in the other project sprints at that time.” We observed that architectural re-factoring is one of the most commonly used techniques when adding novel and big features. Architectural re-factoring is largely started by a research or study prototype. Research projects do not participate in agile product development projects. Hence, the research projects do not have significant maintenance activities or deliverables to customers.

We found that architecture documents are usually created in a research project. However, documentation of the architecture can be deferred until the agile baseline project decides to incorporate a solution proven feasible in the research project. This practice is allowed when an agile baseline project needs to start before the results of the research project are completely known. This is a recently introduced practice that is being trialed in the company. It was interesting to find that research projects have two weeks’ Sprints and their steering groups meet after every Sprint. The resource (i.e., time and budget) estimation of tasks in research projects help managers to allocate resources to study the problems of the tasks and no time estimates are usually available to complete the implementation of the tasks. The steering group evaluates the results after the allocated time and budget for a task have elapsed and decides whether or not further research is needed.

We were interested in finding out why the Sprint duration for the agile research projects is shorter than the Sprint duration of the agile baseline projects. Our investigation revealed that it was an intentional decision of the company’s management to keep the research project Sprint shorter. The main reason for this decision was that the tasks of research projects are expected to often bring surprises and may need more time as planned. That is why a shorter two weeks’ Sprints are needed for controlling and directing the progress of the tasks in these projects.

4.3.2 Feature analysis

Feature description documents are usually produced upfront and they are brought into the product backlog. At that stage, the features are not necessarily connected to a certain Sprint. According to the current practices, new feature proposals and change requirements come from a product manager, the platform team, and the technology roadmap. Once a feature request is received, a feature description document is created. The product manager describes the requirements. Then one or more platform team members identify and evaluate the potential ways of implementing the requirements as well as interfacing the features to be implemented with the rest of the system without introducing negative impact on the overall architecture. At this stage, a project manager and software architect also estimate the amount of implementation work required based on inputs from the developers and available historical data.

It was also found that the most common way of estimation is expert opinion. Feature descriptions are often changing in the course of time. The completeness of the detailed design in a feature description depends on the complexity of the problems at hand. The length of a feature description is usually about five pages at maximum. The product manager reviews and accepts the feature description before it is taken into the Sprint backlog for implementation. The preplanning of features varies from one platform team to another. Hence, there are no prescribed practices for this.

4.4 Agile product development

4.4.1 Sprint pre-planning

We have found that each baseline project has Sprint preplanning sessions, which are usually attended by the manager and architect of the baseline project. There can be one or more project team members who have experience with the high priority items of the product backlog at hand during the preplanning activity. While the participation of product manager and architect is mandatory in the Sprint preplanning sessions, the practice of involving other team members in these sessions varies. The participants of these sessions evaluate whether high priority product backlog items include enough information for inserting them into the backlog of the next Sprint or the subsequent ones. The participants can ask the product manager to add and refine the information of the items in the product backlog when needed. At the end of a Sprint, any unfinished tasks are transmitted into the next Sprint. The participants discuss the potential problems and effort estimates of the items and tasks to be inserted into the next Sprint. Sprint preplanning practices are slightly different in each platform team.

4.4.2 Sprint planning

Feature descriptions provide Sprints with pre-planned and clear features and workload estimates so that Sprint planning can focus on the selection of features to be implemented during the next Sprint. Designers re-factor continuously code without planned architectural tasks. Sometimes special review fixing tasks are planned based on the results of the reviews.

4.5 Architectural communication

The company management is fully aware of the importance of institutionalizing good practices for communicating architectural knowledge as an integral part of integrating product line architectures and Agile
practices. Our study has found that all designers regularly read the specified overall architecture and every new designer is expected to read the whole lot starting from the beginning of the document and all updates to it. One of the architects described the role of the platform’s architectural documents in their processes in these words: “Our architectural documents provide one way to communicate the architecture to research and baseline project teams. And the architectural documents are also usable when new fellows come into the team, so I don’t have to give them two hours’ presentation, but I can just give them the documents and...” Another strategy for supporting architectural knowledge sharing among the platforms teams’ members is to encourage them all to get involved in the architectural communication in daily conversations. To achieve this objective, the workspaces of the team members of all the platforms have been located very close to each other. To avoid any communication gap, distributed development is not encouraged.

The platform teams are usually working with the same product suite all the time. Therefore, the designers in the platform teams are expected to have a good knowledge of the architecture of the existing products. The lead architect and the baseline project architect communicate with each other on regular basis. A good working relationship and mutual trust are considered the vital ingredient for them to achieve the desired results from integrating product line and agile practices in the company.

For exchanging knowledge and experience between research and baseline projects, research project designers often start working in a baseline project on the technology or solution for which they have recently completed the research project. It is a common practice to rotate the staff between agile research and baseline projects. This practice of rotating staff for sharing architectural and process knowledge about the product line platforms with agile baseline projects was explained in these words: “for the team 1, we have now four persons in research, and basically two of those are permanent, and two are cycled, so I started in autumn, and I will be in research until the end of this year, and in the beginning of the next year I start again in the development.”

It was found that the development process usually includes no special architectural meetings but architectural design is continuously discussed whenever there are features which are expected to require architecturally significant changes. We also discovered that team members even use the “Daily Meetings” to discuss and communicate architectural decisions and strategies. According to agile principles, the “Daily Meetings” are strictly restricted to only three things: 1) What has been done; 2) What is going to be done; 3) What are the showstoppers. Everything else is discussed in separate sessions. However, this is not the case in this company. We learned that as a result of integrating product line and Agile practices, architecture is a vital medium of communication in the company. Hence, a discussion on architectural issues has been incorporated in the “Daily Meetings”. The team members use draft papers, flip charts, and emails are used to support the architectural communication during the “Daily Meetings”. In the case of a fully new application, a quite long document is often needed but a new feature can be developed in an existing application without specification. Many new ideas are firstly described and discussed around a white board and then modeled and described in an electronic format and distributed to all stakeholders.

This study has found that the overall architecture description is very useful for subcontractors, new team members and big architectural modifications. At the beginning of the development of a new product, the overall architecture description as well as interfaces to others services are needed. One of the mechanisms used for communicating architectural decisions, changes, and updates is confluence\(^2\). Since the introduction of product line architectures along with agile approaches, each of the platforms of the product line has its own confluence to share architectural documents and knowledge. One of the architects describes the use of confluence in these words: “Basically we write down everything about the architectural decisions in our Confluence, so we don’t specifically update each of the development team. If someone wants to see the status, he can go to confluence. Earlier those decisions were communicated while talking in the lobby or in e-mail discussions, but nowadays there is at least something in Confluence. It’s much better that way.”

4.6 Architectural responsibilities

Currently, a lead architect is responsible for all products on a specific platform. He or she works for research activities but does not participate in daily software development activities of baseline projects. The lead architect makes the final decisions in conflicting architectural situations. The role of software architect has been institutionalized in this company before the adoption of agile development approaches. While the software architects were required to gain a good understanding of the agile approaches. According to the company’s current practice, each baseline project has a team architect, who is expected to have a good understanding of the overall architecture of the product. The team architect has a good understanding what is the final product of the project to be developed and what is the current implementation status of the features to be developed in the project. Moreover, the software architects are also responsible for documenting (or updating) and communicating the architectures with which they are working.

\(^2\) [http://confluence.atlassian.com/](http://confluence.atlassian.com/)
5. Limitations

Like any empirical study, this study also has limitations. Our study was conducted with participants having different roles and representing all the platforms in the company’s product line. All of them had worked with both plan-driven and agile software development approaches. However, the study was conducted in one company. But we hope that a reader may be able to identify experiences and practices that are transferable to his/her environment. Our focus group sessions involved only 8 employees from the company that can be a completeness issue. However, we believe that the findings are based on the experiences and views of those who can represent both the technical and managerial groups in the company and are decisions makers for technological and process issues. Despite these and potentially other limitations, the empirical findings from this study are expected to provide useful information that can make significant contributions to the current knowledge about integrating software product lines and agile practices in industrial settings.

6. Conclusions

The main objective of this study was empirically study organizational processes and practices aimed at integrating SPL approach and agile practices. Based on the findings of this study, we make the following conclusions.

Agile methods such as XP and Scrum assume that the features required by the project customer are small and discrete so that they can easily be implemented in short fixed-time iterations without any up-front explorations [29]. The findings from this study have revealed that such assumption may not be valid in the context of the product being developed for the company’s own use for building various services and solutions as was the case in the studied company.

The results also revealed that the company’s agile product development projects necessarily need documented architectural background information about the product line platforms in addition to tacit architectural knowledge and experience. The assets of the product line platforms are mainly used manually because the rapid evolution of products makes automated production unfeasible.

We can also conclude that the role of a separate research project is important in successfully integrating Agile and SPL approaches. The research projects can be used to explore and solve potential problems between the product line and an intended agile product development project or projects. They can also be used to study the feasibility of existing features during the maintenance or evolution step of the product line.

This study has also revealed that the research projects can also be carried out using agile concepts and practices (e.g., Scrum backlogs, Sprints), however, these projects may not qualify to be agile development because every Scrum Sprint must deliver at least a new executable business functionality [18] and these research projects do not do this.

It should also be noted that there is usually quite a lot of proactive explorative work that has to be carried out in research projects. This upfront work is expected to allow an agile project team to follow practices of agile software development methods with small exceptions. The Sprint planning and Sprint working sessions include no architectural tasks or architectural re-factorings but designers reactively re-factor code continuously for achieving the best and simplest solution. We have also found that the development practices are slightly different in each platform team particularly due to the differences in the used software platforms and related technologies.

Agile methods emphasize collaboration and communication between developers as well as between the customer and developers. This is not the requirement of SPL approach. However, the studied company introduced two roles to communicate architectural decisions and information. There are a lead architect and a team architect that ensure architectural communication between research teams and agile production project teams. For minimizing the need for cross-team communication, the team organization of the company follows the used software platforms in the product line. The company also tries to avoid distributed development whenever possible. And the designers are rotated between research and product development projects.

We believe that the presented findings about the processes being used in the studied company provides a valuable contribution to the growing need and body of knowledge about the mechanics and practices of successfully combining various product line practices (i.e., product line architectural practices in this case) and Agile software development.

Acknowledgements

We are thankful to the participants of this study. Lero is funded by Science Foundation Ireland under grant number 03/CE2/1303-1. This research has been performed in FLEXI ITEA2 research project. Authors also acknowledge the support from National Technology Agency of Finland (Tekes).

7. References

Engineering (APLE'06), Baltimore, Maryland, USA, 2006.


