Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach

Vahid Rahimian, Raman Ramsin

***Abstract*—New Advances in mobile computer technology and the rapid growth of wireless networks in quality and quantity has introduced new applications and concerns in computer science and industry. The unique requirements and constraints associated with mobile systems have brought new challenges to software development for such environments, as it demands extensive improvements to traditional systems development methodologies in order to fulfill the special needs of this field.**

We examine the challenges of developing software for mobile systems, starting by reviewing mobile systems' characteristics and investigating the status quo of mobile software development methods. It has been shown that *Agile* methodologies are appropriate methods for the development of such systems; based on this assumption, we identify specific requirements for a mobile software development methodology, based on which a new agile method is engineered using the *Hybrid Methodology Design* approach. We claim that this methodology, and the approach used for its construction, can facilitate the application of a software engineering approach to the production of mobile software systems.

***Index Terms*— Mobile Software Development, Method Engineering, Agile Methods**

I. Introduction

New advances in mobile computer technology are too fast in emergence for the software engineering field to keep up with. The number of mobile devices with computation capabilities incorporated (such as third-generation mobile phones and personal digital assistants) is growing all over the world, and the ever-increasing demand for specialized software for these devices has caused new concerns for software developers, as this type of software has its own unique characteristics and requirements.

Although commercial mobile systems have not been as successful as originally predicted, mobile operators and mobile value-added software providers expect that the deployment of 3G technologies would have a dramatic effect on the mobile applications industry, by expanding the vast usage of mobile commerce applications and services. Examples of such commercial applications include: mobile information services, advertisement, content-based services, location-based services, and mobile payment applications.

Development of a mobile software system differs from traditional software development in many aspects, as mobile software should satisfy special requirements and constraints. Along with these specific constraints, software produced for mobile environments should be at a high level of quality, so that it can operate properly on different mobile devices that exist or are expected to hit the market in the future. There are numerous challenges that the designer of a software system for mobile environments has to cope with. These challenges mainly stem from ([1], [2]):

- Wireless communication issues (considerations such as availability and disconnection, bandwidth variability, heterogeneous networks, and security risks);
- Mobility issues (concerns such as address migration, and management of location-dependent information);
- Portability issues;
- Various standards, protocols and network technologies;
- Limited capabilities of terminal devices (factors pertaining to low power, risks to data integrity, small-sized user interfaces, and low storage capacities);
- Special privacy and customizability needs;
- Strict time-to-market requirements.

Some of these issues are due to deficiencies in current technology, but most of them are intrinsic to mobility. The design of mobile software systems is therefore much more complicated than that usually seen in software development projects, thus forcing developers to reconsider the use of traditional software development methodologies.

Despite the above-mentioned problems, research endeavors aimed at ameliorating the status quo through enhancing/devising methodologies for mobile-software systems development have been relatively few and far in between. Most of the work performed in this field has been focused on low-level (implementation-oriented) aspects of software development, while high-level (methodology-oriented) issues still remain to be properly addressed.

Manuscript received December 27, 2007. This work was supported in part by the Research Vice-Presidency of Sharif University of Technology.

Vahid Rahimian is with the Computer Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: rahimian@ce.sharif.edu).

Raman Ramsin is an Assistant Professor at the Computer Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: ramsin@sharif.edu).

We propose a new agile methodology developed through applying a Method Engineering (ME) approach [3]. The approach applied herein, called Hybrid Methodology Design [4], is used for iterative-incremental development of methodologies based on a predefined set of requirements and the knowledge acquired from existing methodologies and process patterns/metamodels. We have elicited the high-level requirements of a mobile-software development environment through examining the available literature, and have fed it into the Hybrid process to produce the target methodology. Ideas from the ASD (Adaptive Software Development) methodology [5] and New Product Development (NPD) [6] were used in constructing the methodology.

The rest of the paper is structured as follows: section 2 reviews the related research so far performed on mobile software development; in section 3, the appropriateness of agile methods in software development for mobile environments is discussed; in section 4, we identify the requirements of a methodology tailored for mobile software development; in section 5, the Hybrid Methodology Design approach is used for building a new agile methodology for developing mobile software, using the general patterns followed by agile software development processes, and ideas from ASD and NPD. The final section of the paper presents the conclusions and opportunities for furthering this research.

II. Related Research

Identifying mobile computing challenges [7], [8], adaptation of mobile software [9], [10], the notion of mobile agents [11], resource sharing in mobile environments [12], abstract architectures with special quality features [13], and design patterns for developing mobile software [14] are examples of the more popular, lower-level research so far performed on mobile software development, whereas research on specialized development methodologies is fairly limited. In the rest of this section, we will provide a general survey of methodology-level research conducted in the context of mobile software development.

Gerstheimer and Lupp [15] take the end-user perspective to propose a need-driven system design method for developing mobile applications. Vainio *et al.* [16] recognize the use of market elements as a leading factor in the success of a mobile software product; they recommend utilizing market-based New Product Development (NPD) for improving current methodologies for use in mobile software development.

New Product Development is a complete process for bringing a new product or service to the market [6]. There are two parallel paths followed in the NPD process: one involves idea generation, product design, and detailed engineering; the other involves market research and analysis.

Vainio *et al.* have also analyzed the activities performed in two commonly-used methodologies and have compared them with two NPD process models. They have concluded that current process models should contain more market-oriented activities in order to be efficiently incorporated into the development of mobile software products [16].

Another major work in the field of mobile software development is that conducted by Abrahamsson *et al.* [17], in which a methodology called *Mobile-D* is proposed as an agile approach to mobile application development. The approach is based on development practices borrowed from XP (eXtreme Programming), enjoys method scalability inspired by the Crystal family of methodologies, and provides life-cycle coverage as prescribed by RUP (Rational Unified Process) [17]. Through its phases and disciplines, Mobile-D tries to merge the classic software development process (i.e. traditional plan, design, implement, test, and release activities, as mapped to Mobile-D disciplines of Phasing, Architecture Line, Test-Driven Development, Continuous Integration, Pair Programming, and Off-Site Customer) with the necessary umbrella management/support processes (i.e. project management, software configuration management, and software process improvement, as mapped to Mobile-D disciplines of Metrics, Agile Software Process Improvement, and User-Centered Focus). Although the work of Abrahamsson *et al.* on mobile software development seems to be very promising, the description that they provide of their Mobile-D approach is cursory and incomplete.

III. Agile Development: A Potential Solution for Mobile Software Development

Our investigation of the problem and related research indicates that agile methods have a good level of suitability for the development of mobile applications. Although many agile methods have been introduced over the last decade, none of them has focused on the special requirements of mobile software development. Yet as discussed in the rest of this section, agile methods possess certain properties that make them applicable to the mobile software domain.

Boehm and Turner [18] recognize five main factors that affect agility: operating culture, team size, criticality of the software, competence of the developers, and stability of the requirements. Boehm argues that a software development method works best when it is applied to situations with specific traits [19]; he calls these situations the “home ground” of the software development method. Table I compares the home grounds for agile and plan-driven methods.

Abrahamson identifies agile methods as a potential solution for mobile software development [20]. Based on the home ground for agile methods, he performs a comparative analysis to prove the suitability of agile methods for development of mobile software, the results of which are shown in Table II.

While it has been suggested that agile methods are the most appropriate means for mobile software development among current software development methods, the special characteristics of mobile devices and mobile networks demand some adjustments to current development methodologies. In fact, in order to fulfill the special requirements of mobile software development, new methodologies are needed.

In the next section, we elaborate on the requirements of an ideal software development methodology for mobile software development.

TABLE I
HOME GROUNDS FOR AGILE AND PLAN-DRIVEN METHODS (ADAPTED FROM [19])

Area	Agile Methods	Plan-Driven Methods
Developers	Agile, knowledgeable, collocated, and collaborative	Plan-oriented; adequate skills; access to external knowledge
Customers	Dedicated, knowledgeable, collocated, collaborative, representative, and empowered	Access to knowledgeable, collaborative, representative, and empowered customers
Requirements	Largely emergent; rapid change	Knowable early; largely stable
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams and products	Larger teams and products
Primary objective	Rapid value	High assurance

IV. CHARACTERISTICS OF AN IDEAL MOBILE SOFTWARE DEVELOPMENT METHODOLOGY

In this section, we propose the traits that we believe a development method should have in order to be efficiently employed for mobile software development. Based on the properties we identify herein, a new methodology is later constructed, using the Hybrid Methodology Design approach.

A. Agility

As stated in the previous section, agile methods seem to be a good starting point for constructing a mobile software development method. Agile methodologies are believed to enhance software development flexibility and productivity, by providing means to adapt to changes in requirements and the environment, and also to learn from development experiences. To support early and quick delivery of working software, these methodologies use iterative-incremental development engines to produce artifacts tangible to the customer.

Agile characteristics of highest importance in the context of mobile software development include: iterative and incremental process (which leads to enhanced risk-management capabilities), test-driven development, adaptive process, continuous customer involvement, highly skilled developers, enhanced quality assurance, and continuous process-wide reviews. Furthermore, considering the competitive market for mobile software, shorter time-to-market is a precious advantage; this could be achieved via early releases of operational software, which is itself an important feature of agile methods. Prioritization of requirements is another agile practice that can prove essential in mobile software development, since it sets the stage for and governs risk management activities, and helps ensure that features of higher value to the customer take precedence.

B. Market Consciousness

As the current market for mobile software is biased towards relatively fine-grained software products, a general mobile development process should be mainly oriented towards *product* development, rather than *project* development. Consequently, such processes should focus on establishing the business case, thereby striving to identify the potential market.

The use of NPD practices for market analysis can enhance mobile-software development: NPD process activities utilize market information for mitigating uncertainties and risks. In a market-oriented process for mobile software development, market and customer needs should be carefully analyzed, and a strict release schedule, which meets time-to-market requirements, should be established and maintained during development. In contrast to the typical modus operandi, in which a process's main focus is on technical activities, a mobile software development process should maintain a balance between market-oriented and technical activities.

TABLE II
MAPPING AGILE HOME GROUND THEMES TO TRAITS OBSERVED IN MOBILE SOFTWARE DEVELOPMENT (ADAPTED FROM [20])

Ideal Agile Characteristic	Rationale	Mobile software
High environment volatility	Due to high change of requirements, less need for up-front design & planning, need for an incremental and iterative development approach.	High uncertainty, dynamic environment: Hundreds of new mobile phones produced each year
Small development teams	Small teams are able to react more rapidly, share information, need less documentation , etc.	Majority of mobile software is developed in micro or SME companies, or development teams.
Identifiable customer	To avoid business misunderstanding	Potentially unlimited number of end-users. Business customer easier to identify, e.g. distributor.
Object-oriented development environment	Most tools that support agile development exist for object oriented platforms.	E.g., Java and C++ used; some problems in proper tooling e.g. for refactoring and test-first approach
Non-safety critical software	Failures do not cause loss of lives. More agility can be pursued.	Majority of existing mobile software is for entertainment purposes. Mobile terminals are not reliable.
Application-level software	Large embedded systems require extensive communication & verification mechanisms.	While mobile systems are complex and highly dependent, mobile applications can be stand-alone applications.
Small systems	Less upfront design needed	Mobile applications vary in size, but are generally less than 10000 lines of code.
Short development cycles	For the purposes of rapid feedback	Development cycles vary. Typical mobile applications and services can be developed within a 1-6 month time-frame.

C. Software Product Line Support

A software product line is "a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [21]. The special benefits of applying the product line approach to mobile software development are mainly due to the fast pace of advances in mobile technology. This makes the life-cycle of mobile software products ever shorter. Consequently, software companies tend to develop a family of mobile software products in a bid to reduce development costs.

As a result, we suggest that a mobile software development process should provide means for supporting product line engineering. Component-based development, use of reconfigurable architectures, and product scoping can help in achieving this capability. The process should also provide adequate guidance on performing quality product line design.

D. Architecture-Based Development

The efficiency of the software product line approach depends on the development firm's ability to invest in the development of a common platform. This necessitates the development of a general architecture for such products.

E. Support for Reusability

The need for using functionally equivalent components (such as message boxes in a specific language), and functionally similar components (such as security operation classes), along with tight time constraints on the development of mobile software, requires the developer to use reusable components extensively. Having to develop these components from scratch each time they are needed increases the cost of mobile software development, delays product delivery, and makes the software more error prone. Support for component-based and layer-based development approaches is therefore essential in a mobile software development methodology.

F. Inclusion of Review and Learning Sessions

One of the most effective factors in a production company's success is its ability to abstract the knowledge obtained during product development. As mentioned above, today's mobile software development industry tends to be product-oriented; the methodology should therefore incorporate review sessions throughout the process to ensure product analysis, and include "lessons-learnt" sessions after delivering a working product to the market to ensure that experiences are analyzed and logged.

G. Early Specification of Physical Architecture

Mobile terminals' constraints should be considered from very early stages of software design. In fact, a high degree of technical risk can be traced to mobile hosts' limitations and their differences in the implementation of basic features. Consequently, the physical architecture should be elaborated in the early stages of software development. A prototype may also be required in order to mitigate technical risk.

V. Building a Mobile Software Development Methodology Using Method Engineering Techniques

Motivated by the belief that no one methodology fits all situations, Methodology Engineering was first introduced as a discipline aimed at constructing methodologies to match given organizational settings or specific development projects [22]. The discipline later came to be known as Method Engineering, with the definition broadened as: "The engineering discipline to design, construct, and adapt methods, techniques and tools for the development of information systems" [3].

There are several approaches to method engineering [23]:

- *Ad-hoc*: Constructing a new methodology from scratch;
- *Paradigm-based*: Instantiating, abstracting or adapting an existing meta-model to produce the target methodology;
- *Extension-based*: Enhancing an existing methodology with new concepts and properties;
- *Assembly-based*: Constructing the methodology through assembling method fragments retrieved from a repository.

In this section, we describe our work in utilizing an approach called Hybrid Methodology Design [4], through which the target methodology has been built based on the requirements defined in the previous section and knowledge gathered from existing methodologies and process patterns/metamodels [4], [6], [24]. The Hybrid Methodology Design process has been devised as a top-down iterative-incremental process consisting of the following tasks [4]:

- *Prioritization of the Requirements*: performed at the start of the process and repeated at the end of each iteration. The requirements are ordered according to their relevance to the current scope and level of abstraction, focusing the design process on satisfying requirements of higher significance. At the start of the process, abstraction is at its highest level and the scope encompasses the whole lifecycle, therefore requirements with lifecycle-level impacts are given precedence; as design progresses to lower levels of abstraction, priority is gradually shifted to requirements with finer-grained aspects.
- *Iterative Design Engine*: The following tasks are performed in each iteration:
  - Selection of the design approaches to be used in the current iteration: The possible approaches to designing the target methodology include:
    - *Instantiation*: instantiating an already available process metamodel;
    - *Artifact-oriented*: devising a seamless complementary chain of artifacts and building the process around it;
    - *Composition*: using one of the already available libraries of process patterns;
    - *Integration*: integrating features, ideas and techniques from existing methodologies.
  - Application of the selected design approaches aimed at defining the methodology at the current scope and level of abstraction: Special attention should be given to existing methodologies and process patterns/metamodels, thus implementing features of strength and avoiding common pitfalls. The prioritized set of requirements focuses the design effort on satisfying requirements of importance. The methodology elements designed are then integrated into the produced blueprint.
  - Revision, refinement and restructuring of the methodology built so far in order to accommodate the changes made in the current iteration.
  - Specification of the level of abstraction for the next iteration, and definition of the scope and intended level of detail.
  - Revision and refinement of the requirements, including prioritization according to the scope and level of abstraction intended for the next iteration.

The iterative-incremental engine at the core of the design

process generates the methodology in a top-down fashion – from the general lifecycle to the details of activities – using the requirements and methodology descriptions as a basis.

Of the four design approaches used in each iteration, two – i.e. Integration and Artifact-oriented – are relatively novel in this context. The Integration approach promotes integrating ideas and techniques directly from existing methodologies, instead of first dissecting the methodologies into fragments (as is common practice in assembly-based method engineering); the intention is to preserve synergy and avoid loss of functional capacity. Different approaches of the Hybrid approach have different uses depending on the scope and abstraction level of the design activity undertaken in the current iteration: Instantiation is more useful when designing high-level aspects of the methodology, Integration and Composition are more suited to the design needs of low-level aspects, and the Artifact-oriented approach comes in between, i.e. while less useful at the general lifecycle level, it is indispensable when addressing seamlessness at the inter-subprocess and intra-subprocess levels. We constructed the overall framework of the target methodology through four iterations of the Hybrid Design Engine, with the results shown in Figure 1. In following the Instantiation approach, the generic software development life cycle was set as the base process, i.e. at the highest level of abstraction. It provides adequate high-level coverage of generic software development activities and supports umbrella activities, thus providing the framework needed for satisfying the requirements through further iterations of the engine.

In the first iteration, the methodology was elaborated via the use of generic patterns for risk-based, architecture-centric, and test-based development. Analysis was split into Preliminary Analysis and Detailed Analysis, in order to mitigate development risks. To achieve architecture-based development, Design was split into Architectural Design and Detailed Design, with the relevant feasibility analysis, planning and architectural design activities duly added. The Implementation and Test sub-processes were combined in order to accommodate test-based development. These patterns are frequently used in agile methodologies. Moreover, since the main focus in current mobile software development is on product development, we incorporate the Commercialization phase; this would also facilitate product line support. The design approach in this iteration was mainly *instantiation*, using meta-models – including SPEM [25] and OPF [26] – and general object-oriented lifecycles – such as OOSP [27].

In the second iteration, market consciousness was focused upon through borrowing activities from New Product

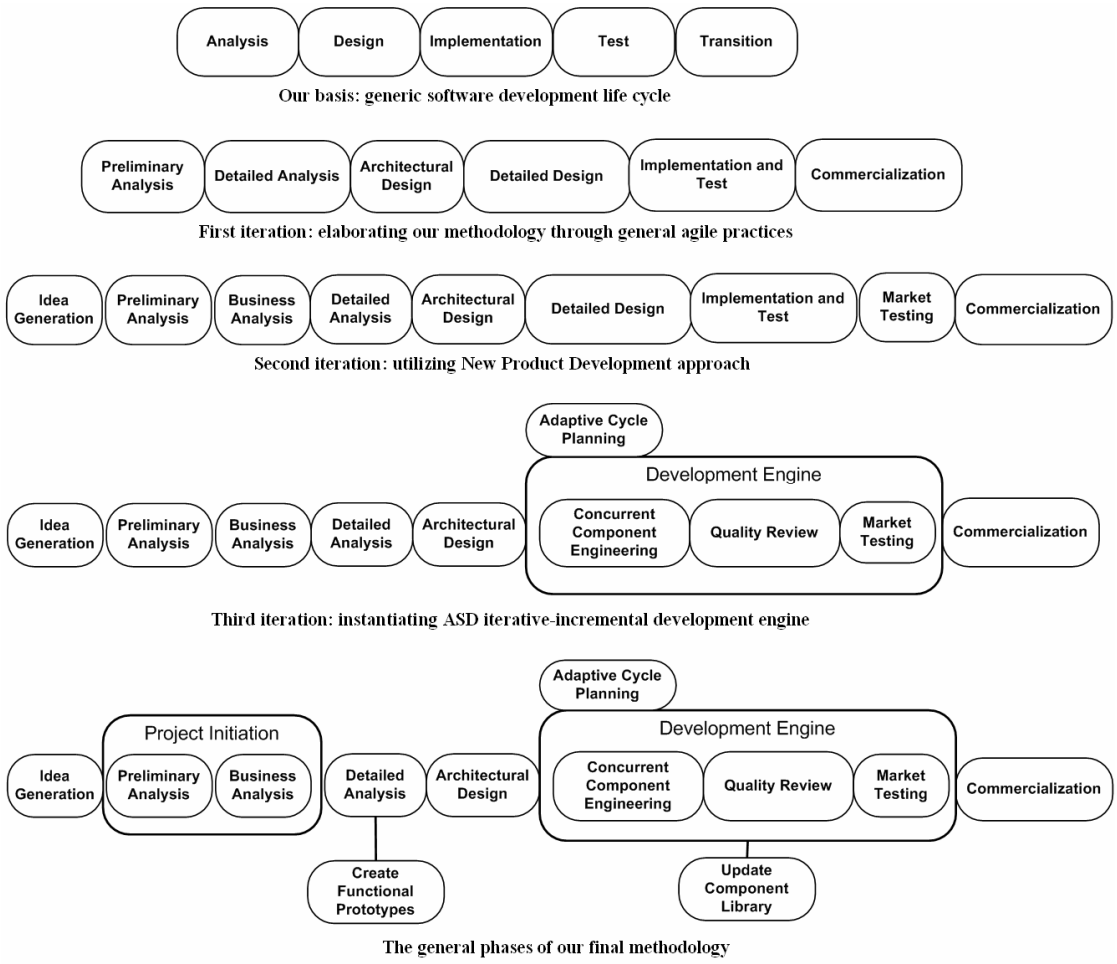

Fig. 1. Gradual refinement of our proposed methodology during iterations of the Hybrid Methodology Design process

Development. These include Idea Generation (incorporated into the beginning phases of the process), and Market Testing (performed before commercializing the software). The design approach in the second iteration was *integration*, using reusable parts of the NPD process.

In the third iteration, we enhanced our process's development engine by incorporating ideas from Adaptive Software Development (ASD) [28]. ASD is a component-based agile methodology especially rich in quality assurance measures. The Speculate-Collaborate-Learn cycle that forms the basis of the ASD process provides the means to cope with the uncertainties of software development. Using the *integration* approach, the iteration resulted in the incorporation of review sessions into the methodology. By benefiting from component-based development practices, we have also improved our process's support for reusability.

Considering the potential technology risks in mobile software development, the last iteration was mainly concerned with adding prototyping (as a process pattern [27]) to our process. Prototyping also facilitates the early specification of the physical architecture. Moreover, the process was refined through moving the Preliminary and Business Analysis activity into Project Initiation, with output artifacts based on those produced by ASD's Project Initiation phase. The design approach used was mainly *composition*, with *integration* used when reusing ideas from the ASD methodology.

VI. CONCLUSIONS

We have identified the main requirements of a mobile software development methodology, based on which a high-level methodology framework was built using the Hybrid Methodology Design approach. Our proposed methodology is an agile risk-based methodology, highly influenced by the ASD method and NPD approaches. The requirements-based nature of the Hybrid approach ensures that the requirements are properly addressed, and validating the resulting methodology against the requirements seems to confirm this.

In furthering this research, our next step would be to apply further iterations of the Hybrid Design Engine at lower levels of abstraction, thereby specifying the finer-grained tasks of the process. The methodology can then be put to test in developing commercial mobile software products; the process, and the requirements on which it is based, can thus be tuned. Another strand of research can focus on extending our work to the development of mobile software *projects*, rather than *products;* since technology advances with incredible speed, this is likely to be an industry demand in the future.

REFERENCES

[1] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing". *IEEE Computer*, Vol. 27, No. 4, April 1994, pp. 38-47.

[2] I. S. Heyes, *Just Enough Wireless Computing*, Prentice Hall, 2002.

[3] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools", *Information and Software Technology*, Vol. 38, No. 4, 1996, pp. 275-280.

[4] R. Ramsin, "The Engineering of an Object-Oriented Software Engineering Methodology". Ph.D. Thesis, University of York, York, UK, 2006. Available: http://www.cs.york.ac.uk/ftpdir/reports/YCST-2006-12.pdf.

[5] J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000.

[6] K.T. Ulrich, S.D. Eppinger, *Product Design and Development*, 3rd Edition, McGraw-Hill, 2004.

[7] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing", in *Proc. of the Fifteenth annual ACM symposium on Principles of distributed computing*, 1996, pp. 1-7.

[8] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", *IEEE Personal Communications*, August 2001.

[9] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, M. Ruberi, C. Sala, and D. Vitali, "Towards Highly Adaptive Services for Mobile Systems", in *Proc. of the Working Conference on Mobile Information Systems (IFIP TC8)*, 2004.

[10] H. Grine, T. Delot, S. Lecomte, "Adaptive Query Processing in Mobile Environments", in *Proc. of the 3rd international workshop on Middleware*, 2005.

[11] V. Pham, A. Karmouch, "Mobile Software Agents: An Overview", *IEEE Communications Magazine*, July 1998.

[12] E. Valavanis, C. Ververidis, M. Vazirgianis, G.C. Polyzos, K. Norvag, "MobiShare, Sharing Context-Dependent Data & Services from Mobile Sources", in *Proc. of the IEEE/WIC International Conference on Web Intelligence (WI 2003)*, 2003, pp. 263-270.

[13] M. Haahr, R. Cunningham, V. Cahill, "Towards a Generic Architecture for Mobile Object-Oriented Applications", in *Proc. of the 2000 IEEE Workshop on Service Portability and Virtual Customer Environments*, 2000, pp. 91-96.

[14] E.S. Chung, J.I. Hong, J. Lin, M.K. Prabaker, J.A. Landay, and A.L. Liu, "Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing", in *Proc. of the 2004 conference on Designing interactive systems: processes, practices, methods, and techniques*, 2004, pp. 233-242.

[15] O. Gerstheimer and C. Lupp, "Needs versus technology - the challenge to design third generation mobile applications", *Journal of Business Research*, Vol. 57, No. 12, Dec. 2004.

[16] A.M. Vainio, T. Tuunanen, P. Abrahamsson, "Developing Software Products for Mobile Markets: Need for Rethinking Development Models and Practice", in *Proc. of the 38th Hawaii International Conference on System Sciences (HICSS'05)*, Jan. 2005.

[17] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Mobile-D: An Agile Approach for Mobile Application Development", *in Proc of the OOPSLA'04 Conference*, 2004.

[18] B. Boehm and R. Turner, *Balancing agility and discipline: A guide for the perplexed*, Addison-Wesley, 2003.

[19] B. Boehm, "Get ready for agile methods, with care", *IEEE Computer*, Vol. 35, No. 1, 2002, pp. 64-69.

[20] P. Abrahamsson, "Keynote: Mobile software development – the business opportunity of today", in *Proc. of the International Conference on Software Development*, 2005, pp. 20-23.

[21] P. Clements, L. Northrop, "Software Product Lines", course notes of Product Line Systems Program, Software Engineering Institute, Carnegie Mellon University, 2003.

[22] K. Kumar, R. J. Welke, "Method Engineering: a proposal for situation-specific methodology construction", in *Systems Analysis and Design: A Research Agenda*, 1992.

[23] J. Ralyté, R. Deneckére, C. Rolland, "Towards a generic model for situational method engineering", in *Proc. of CAiSE'03 (LNCS 2681)*, 2003, pp. 95-110.

[24] R. Ramsin, R. F. Paige, "Process-centered review of object-oriented software development methodologies", *ACM Computing Surveys*, Vol. 40, No. 1 (February), 2008, Article 3, pp.1-89.

[25] OMG, *Software Process Engineering Metamodel Specification (v1.0)*, Object Management Group (OMG), 2002.

[26] D. Firesmith, B. Henderson-sellers, *The OPEN Process Framework: An Introduction*, Addison-Wesley, 2001.

[27] S. W. Ambler, *Process Patterns: Building Large-Scale Systems Using Object Technology*, Cambridge University Press, 1998.

[28] J. Highsmith, "Messy, exciting, and anxiety-ridden: Adaptive software development", *American Programmer*, Vol. 10, No. 4 (April), 1997, pp. 23-29.