EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Human-centered software development methodology in mobile computing environment: agent-supported agile approach

Hyo-Eun Eom[1] and Seok-Won Lee[2*]

Abstract

Due to the large demand of the services from the wireless networked computing environment, the capacity and performance of wireless devices, platforms, and applications are improved. Especially, in recent years, the use of mobile applications has increased dramatically along with the concept of 'smart' phone. As a result, the center of human e-life has been transferred from a conventional desktop space to a mobile wireless computing environment. These rapid changes also generate additional requirements for more sophisticated and complex functionalities in wireless computing environment such as personalization or adaptation which result in the requirements of 'human'-centered software development methodology in wireless computing environment. Although there are well-known classical software development methodologies such as waterfall, they do not fit to the characteristics of wireless computing environment. Instead, the agile software development methodology is generally used to deal with dynamically changing requirements in wireless computing environment. However, agile software development methodology is knowledge- and labor-intensive and does not fully cover the requirements that are essential to the wireless computing environment. In this paper, we analyze the requirements of mobile applications in wireless computing environment and propose a human-centered software development methodology which is to integrate agile philosophy and agent technology. Additionally, we employed the concepts of software product line engineering in order to understand and represent the variability of dynamically changing requirements. Consequently, what we propose in this paper is an agent-supported agile-based mobile software development methodology in wireless computing environment for supporting adaptive requirement changes and their automatic implementation. In order to demonstrate the feasibility of the proposed approach, we have performed two experimental case studies by developing android mobile applications.

Keywords: Agile software development, Mobile computing environment, Adaptive software, Agent-oriented software engineering, Multimedia mobile application, Requirements discovery

1. Introduction

Recently, we are witnessing an explosion of mobile applications with a lot of multimedia services driven by the various mobile technologies in wireless computing environment. Furthermore, the user's expectation toward functionalities and qualities keeps increasing through the appearance of 'smart' applications in the wireless computing environment. Users cannot be satisfied with one-dimensional functionalities in such environment. They want the mobile applications to be more intelligent. This intelligence includes the ability of recognizing the user's intention and context and the ability of adapting to the dynamically changing environment by itself. In other words, the demands for the software adaptation requirements move to the next level, which requires more intelligence, in the mobile computing environment (MCE). In order to support such software adaptation, a large scale of data should be continuously monitored and supervised. A large amount of information must be collected and analyzed as an information lifecycle. This is shown in Figure 1. Moreover, the

* Correspondence: leesw@ajou.ac.kr
[2]Department of Information and Computer Engineering, Ajou University, San 5 Woncheon-dong, Youngtong-gu, Suwon-si, Gyeonggi-do 443-749, Republic of Korea
Full list of author information is available at the end of the article

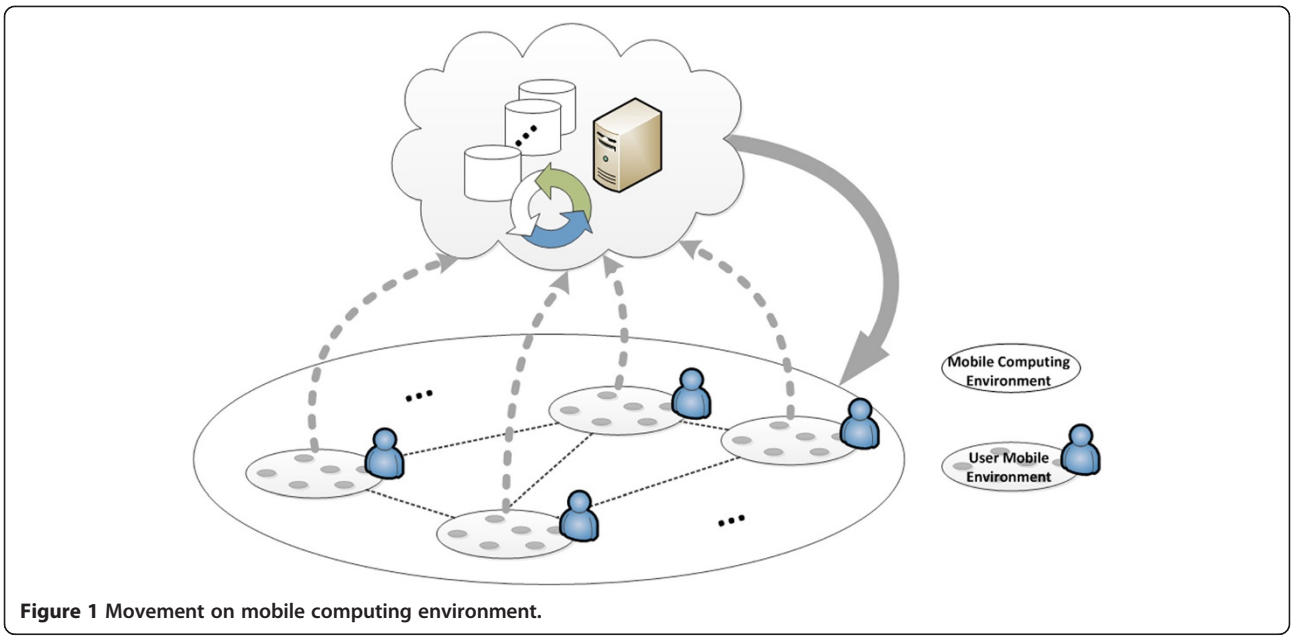

Figure 1 Movement on mobile computing environment.

additional requirements of users, systems, and their interactions, which are not elicited during the requirement engineering process, should be discovered from the actual usage of the application by the users.

From the above observations, we focus on three problems. Firstly, there is no existing applicable software development methodology that reflects the characteristics of MCE and its adaptation requirements. Secondly, additional requirements of software adaptation keep accumulated on top of existing requirements by adding the increased complexity of software and its engineering process. Lastly, a well-structured information processing lifecycle is needed to model, analyze and retrieve the complex and sophisticated information from the dynamically changing users, systems and the environment. Thus, our approach integrates agile philosophy and agent technology to resolve the above problems. The proposed framework (in Section 5.4), which adopts the human-centered software development methodology, is based on the hybrid use of the above two concepts.

2. Problem statement

2.1. Problems

As described in the previous section, there are some problems on the process of mobile software development. Those problems are as follows:

- Conventional software development methodology cannot be applied to the mobile software development.
- The complexity of mobile software engineering is increasing.
- Knowledge- and labor-intensive processes create additional burdens.

Firstly, the MCE is different from the conventional computing environment. The lifespan of mobile software is quite dependent on the usability, and this usability is also dependent on the user requirements. Requirements from mobile software users are variable and capricious. Thus, if we want to make a long-term mobile application on the MCE, it is critical to reflect and satisfy the dynamically changing requirements. Also, the accuracy of processed information for adaptation is important. The MCE has limited resources, including memory, battery, and display. Therefore, developers are required to consider these limited resources. These differences and restrictions make the conventional software development methodology inapplicable for the mobile software development. The additional requirement of adaptation also pulls variable cases and variants, which should concern the developer. In other words, the applications in the MCE need to be more intelligent. They need to recognize the dynamically changing environment and properly react to those changes. To develop such application, the conventional development methodology is not sufficient. Therefore, some labor- and knowledge-intensive processes and complexities exist in the background of realizing adaptation, and this is why the novel development methodology is proposed in this paper.

2.2. Objectives and propositions

In this research, our study aims to analyze the design of the software development methodology specialized in

adaptive mobile-software development. The purpose is to find a way to control the complexity and to reduce burdens during the process of adaptation realization. In detail, we have several propositions which are as follows:

- Reflecting features of the MCE
- Reflecting the adaptation requirement
- Controlling complexity on adaptive mobile software engineering
- Reducing loads from labor- and knowledge-intensive process of developers
- Improving accuracy and functionalities

Basically, the features of MCE have to be reflected onto the methodology, not to miss the factors affecting the usability of the mobile application. Through this, developers can avoid the risk of an early buried and dead application. Then, we should reflect the adaptation of the found factors. These factors become the requirements of the development, and they need to be adapted into the mobile software development process by the adaptive functions. After reflecting additional requirements, the complexity, which is increased by the dynamically changing requirements for the adaptation, needs to be managed. Finally, for more usability of adaptation, sufficient accuracy of information and functionalities of application must be achieved.

3. Related works

3.1. Features of mobile computing environment

MCE includes the environment of the mobile software's execution and the environment of the mobile software's development. The MCE has more limitations compared to the conventional computing environment. Most of them are the constraints on resources, which are as follows:

- Memory. The amount of memory that the application can use is limited by constraints of resources.
- Battery. The amount of the power consumption is limited by the physical size of the mobile devices and the impossibility of power supply.
- Display. The size of display is limited, so that the arrangement of information and view needs to be considered.
- Signal. As many functions of the application are based on network environments, the stability of signals is required.
- Speed and smoothness. To keep the real-time usability present in the user's life, the speed and smoothness of mobile software have to be assured.
- Maintenance. To satisfy changing requirements and updating new information and functions, continuous upgrade and communication with users are needed.

Because of these constraints, mobile applications avoid constructing huge data structures and weighted computations, and pursue compactness and lightness. In addition, users on MCEs need to be active to express opinions on applications they frequently use, and these requirements are various and variable.

3.2. Agile software development methodology on MCE

The agile software development methodology was proposed to resolve the problems of conventional software development methodology, which shows stiffness and intensive labor on the preparation and document process [1]. Thus, the agile methodology focuses on the member's capability, synchronization, and sustained deployment of products. By these principles, the software development process based on agile methodology generally includes the iterative and evolutional development processes [2] such as Scrum and XP [3,4]. Due to flexibility and short-term iteration of the agile process, the agile methodology is usually applied to mobile software projects [5]. Thus, the agile methodology has been discussed for mobile software development like the Mobile-D project [6]. However, there is no widely agreed software engineering process for the MCE which considers the dynamic requirement adaptation.

3.3. Agent and software product line

An agent is an encapsulated computer system that is situated. From the establishment of this definition, the study of agent-oriented software engineering (AOSE) or agent-based approaches for software engineering has been researched steadily [7]. AOSE is to adapt the agent concept to software engineering in order to control the complexity field, called as AOSE by Jennings and Wooldridge [8-10].

Software product line engineering (SPLE) is a field of software engineering involving the production of serial applications through distinguishing variable parts from unchangeable parts. The domain and variability concepts of SPLE are used to express the variable and common components in a software system. Presently, these concepts are used to describe the architecture of an adaptive software system as a changeable software at runtime. In SPLE, adaptive software is regarded as the system containing the different software at a different case in one product line. Through using SPLE concepts on engineering an adaptive software system, systems and developers can realize where and how the system changes.

4. Approaches

Now, we introduce our solution and approaches to the problems described above. Basically, our solutions are based on the hybrid of three concepts: agile, AOSE, and agent technology. That is, these approaches were developed through combining three ideas, which look isolated at first glance. Therefore, in this section, we explain how those key ideas are combined and used to build our solution.

4.1. Mobile computing environment and agile

As seen through the explosion of the popularity of mobile devices and applications, mobile software supports many services with or without the support of a desktop settled on human life. Hence, requirements for improved function and service approach need to be considered. To satisfy their requirements, developers have to speed up the progress of engineering and strive to produce good performances during relatively tight schedules. Moreover, mobile software and devices have inherent constraints, which produce additional requirements concerning processes.

Agile software development methodology is already used for the development of mobile software for the reason that the agile methodology is capable of covering some critical features of the MCE:

- Various requirements. In the MCE, mobile software requirements are changeable for various reasons like the user's needs, capabilities of devices, platforms, etc. Thus, for mobile software development processes and maintenance, frequent tests and revisions are needed to reflect various requirements of users and markets.
- Labor- and knowledge-intensive. For dynamically changing requirements and relatively strict terms for development, project members go through many turning points and iterative processes. Moreover, the quality of their results and development processes rely on the capabilities of the project members. Therefore, modern mobile software requires much knowledge and labor.
- Human-centered. As functionality and usability are directly related to the life of mobile applications, human-centered functions and interfaces are required to improve usability and functionality.

Generally, a conventional software development methodology cannot deal with these features of the MCE. Unlike the conventional methodology, agile seeks to accept changes in the middle of processes and to continuously communicate with project members, including customers and business members. Agile places the highest importance on delivering valuable software. Thus, the above features of MCE, except for the second characteristic, can be supported by the process and principles of the agile methodology. Consequently, in this study, we choose agile as the basic philosophy of our mobile software development methodology.

4.2. Agile and agent technology

As we mentioned, the software development process based on agile basically has iterative and evolutional processes to reflect requirement changes actively. Because of the rapid process, intensive labor and knowledge are required of developers, indicating that agile cannot solve the second aforementioned characteristic of MCE. In particular, the burdens on developers of the processes increase when they want to satisfy the additional adaptation requirements because adaptation includes a high-level information processing. Also, the rapid process repeats tests and inspections until the requirements are sufficiently met. Suppose we have a technical support on these burdening processes, and then we can reduce the load of knowledge engineering. In order to do that, the technical support has to possess the ability to collect widely spread information and to process collected information, like reasoning and knowledge processing. Agent technology has features that could supply those abilities and agility, as it is autonomous, reactive, and proactive [11,12]. Agent operates on the basis of knowledge (e.g., belief-desire-intention agent model [13-17]). Thus, agent technology can support the burden process based on information collecting and processing.

4.3. MCE and agent-oriented software engineering

In the field of software engineering, there exists a study about controlling the complexity of complex software systems by abstraction, using the agent concept called AOSE [8]. In this study, systems are considered composite components with several sub-components recursively representing specific goals or objectives. Each component could be expressed as an agent, which is reactive, proactive, autonomous, and social. That is, the complex software system is an organic set of components operating autonomously to meet the objectives. In another study about controlling the complexity of software intensive systems through the agent-oriented approach, researchers said that there are remarkable improvements for some decomposable and organizational problems. Therefore, an agent-oriented approach for the MCE is beneficial. In addition, the features of an agent give flexibility and scalability to the system organization, which is needed to realize adaptation. Consequently, the AOSE approach reduces complexity and other potential abilities to adapt to the system.

Hence, in this research, we plan to apply the AOSE approach to mobile software engineering to control

complexity. From AOSE viewpoint, the MCE includes elements related to mobile computing, like mobile devices, mobile software, users, and developers. Such an MCE is generally composed of various mobile elements, but is not apart from a conventional computing environment. Such components of MCE communicate mutually in sharing their services and information, and sometimes even collaborate and cooperate like family products. Thus, the architecture of MCE could be expressed as seen in Figure 2 which shows that an MCE is generally composed of user mobile environments and can be decomposed to smaller systems composed of mobile devices and applications.

As we discussed above, each of these three concepts has specific features, and these features have supplementary relations with each other as seen in Figure 3. In the MCE, requirements are easily changed, and frequent tests and revisions are necessary. Because of this, iterative implementation in the short term requires intensive labor and knowledge of developers. Furthermore, human-centered functions and interfaces are needed. Consequently, the loads of developers are the result of these busy processes and additional requirements of adaptation. Agent technology can support part of this load, specifically in the areas of knowledge engineering and inferences. Moreover, an agent-oriented approach can deal with the complexity of an MCE for mobile software engineering. Therefore, we plan to combine these three concepts in a hybrid approach to address the problems of mobile software engineering.

5. Body of methodology

The mobile software development methodology proposed by this research is to resolve the problems caused by the complexity and variability of MCE. The proposed methodology uses the flexibility of the agile methodology, the technical capacity of agent systems, and the abstraction of AOSE concepts. For designing such a methodology, we discuss four phases: principles, concepts, processes, and technical supports.

5.1. Principles

Though agile is generally used for mobile software development instead of conventional software development methodology, it does not focus on mobile software development. Thus, we need to reconsider the principles of agile from a mobile software development view, but as original agile principles are old-fashioned, we referenced recently proposed agile principles in an article by Williams [18]. The five principles, used by the proposed methodology, are selected from them. They are shown in Table 1. The principles in italics are the selected principles. Principle 9 was slightly revised to mention usability and functionality, which are necessary to consider during the process.

As stated in principle 1, the most important principle of mobile software development is delivering valuable software to users continuously in the short term. Flexible processes and willing minds are needed to accept dynamically changing requirements from various users. Mistakes and misunderstandings have to be reduced

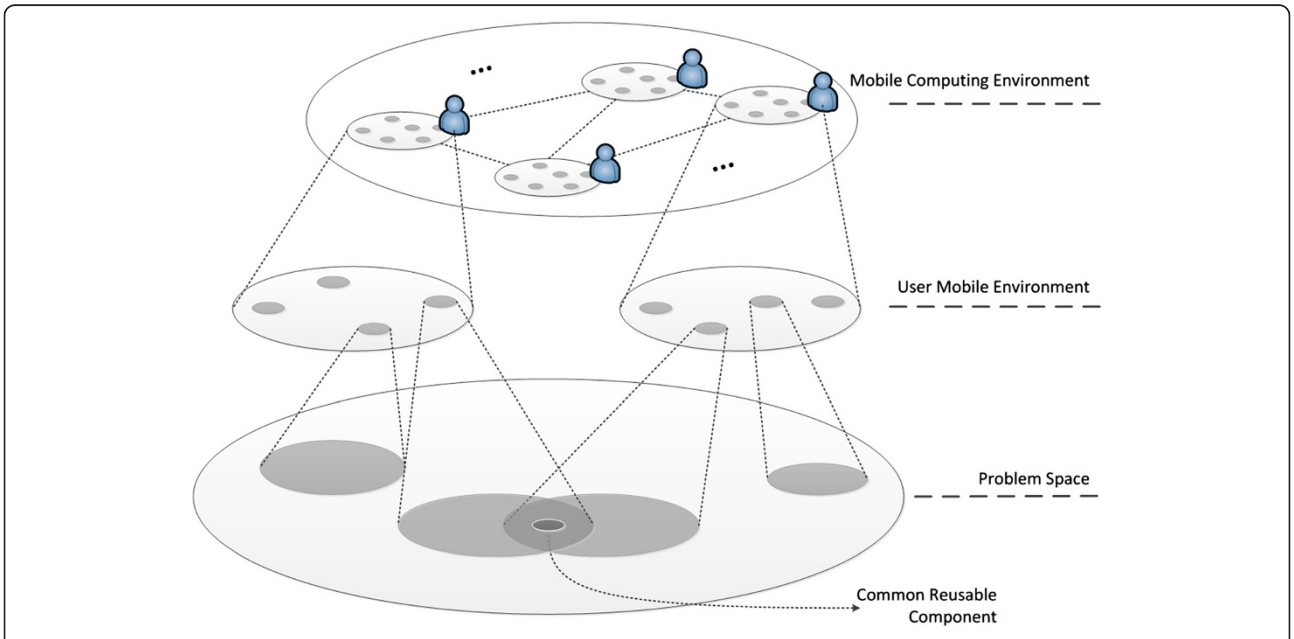

Figure 2 The architecture of an MCE based on an agent-oriented approach.

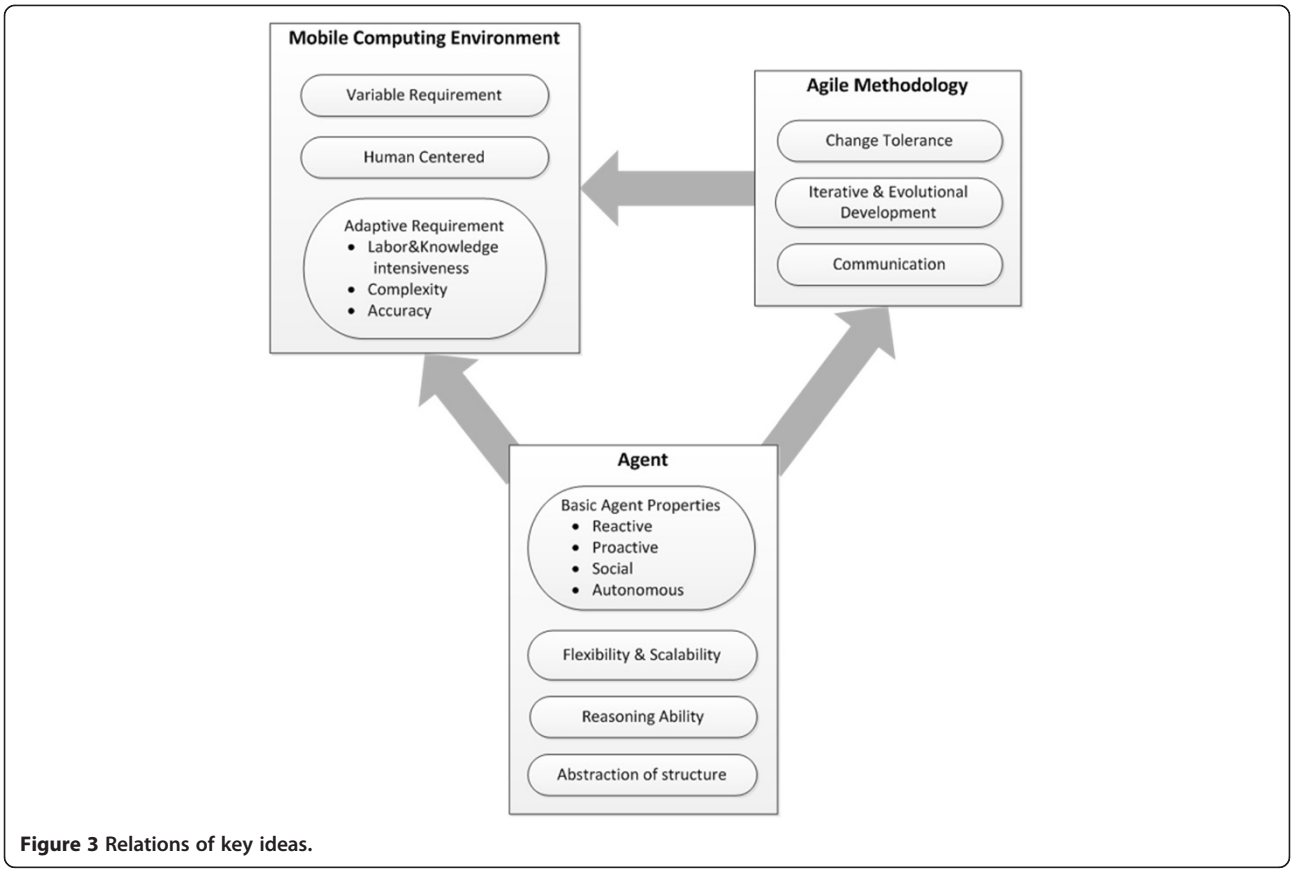

Figure 3 Relations of key ideas.

through frequent communication and specifications of important changes and feedback. During these steps, developers should pay attention to technical excellence and preserve usability in order to make the product valuable to users. Moreover, important change-centered documentation is pursued instead of heavy documentation. The ultimate purpose of these principles of mobile software development is to produce a human-centered application valuable to real users through flexible processes and active communications.

5.2. Concepts

We referenced several terms to explain certain concepts and artifacts within our process.

- Requirement elicitation. A method of eliciting requirements, which are based on the user's situation and context, for mobile software development.
- Conti. The material for description of a scenario composed of components like mobile software, agent, and external components. The material includes UI and view, data flows, and simple descriptions about the internal computation.
- Profile. Information used for figuring out the environment and users for adaptation. Profiles are composed of processed information and a data structure.
- Variable point. An asset differing in the behavior and its situation.
- Variants. Descriptions of required behaviors to make a choice for the next branch in variable point at the given situation.
- Feed report. A record of important changes and their results for feedback of each iteration.

5.3. Process

Basically, our mobile software development process is based on AOSE and follows agile principles. Our process includes an iterative process for the development of each component, and this manner can satisfy the dynamically changing requirements rapidly. As seen in Figure 4, our process of mobile software development starts from setting up the basis with ideas, categories, objectives and services of the system. The component is divided into simple mobile applications, sharable services, and agents. It could be evolutionally implemented through iterative processes. The initialization and test processes would be supported by agent technology in order to reduce the burden of information processing and repeated testing. Each step will be explained as follows.

Table 1 Agile Principles as outlined from an article by Laurie Williams [5]

Principles	Agile principles as outlined from an article by Williams [18]
Principle 1	The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Principle 2	Changing requirements are welcome at the start of each iteration, even late in development; agile processes harness change for the customer's competitive advantage.
Principle 4	Business people and developers must work together daily throughout the project.
Principle 5	Projects should be built around empowered, motivated individuals with a shared vision of success; give them the environment and support their needs, clear their external obstacles, and trust them to get the job done.
Principle 6	The most efficient method for conveying information to and within a development team is through synchronous communication; important decisions are documented so that they are not forgotten.
Principle 7	Valuable, high-quality software is the primary measure of progress at the end of a short time boxed iteration.
Principle 8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Principle 9	Continuous attention to technical excellence and good design enhances agility and makes the product valuable through its usability and functionality.
Principle 10	Simplicity—the art of maximizing the amount of work not done—is essential.
Principle 11	The best architecture, requirements, and designs emerge from self-organizing teams guided by a vision for a product release.
Principle 12	For each iteration, the team should candidly reflect on the success of the project, its feedback, and how it could be more effective, then tune and adjust its plans and behavior accordingly.

5.3.1. Initialization

The initialization process includes idea generation and the requirement elicitation process to initiate the target system and its scope. Firstly, the mobile applications, which are chained with parallel usages, are derived through a survey about the status of mobile software usage. Such derived applications and requirements present the requirement that is reflected on the subconscious of users. Thus, the emerging requirement that users want to get is elicited through analyzing the requirements and implicit relations among the existing applications. Figure 5 represents this proposed method. Based on such emerging requirements, the target system will be settled, and ideas and the scope will be edited.

5.3.2. Design architecture

After the initialization of a target system, we have to define the architecture of target system based on AOSE, composed of components that are described by specific objectives. From the predefined requirements of the target system, engineers find uncovered problem spaces and design a component that has specific objectives to solve the problem. Shared problems between components could be solved by defining a reusable service of the other components. In addition, an independent component could be added to solve the problem. Moreover, variable points and common points are described in this architecture based on SPLE concepts to explain which components are available for the change and which are not, as shown in Figure 6.

5.3.3. Decomposition and prioritization

The architecture is decomposed into segments and prioritized in order of development priority in several levels according to the relationships among the components. This step is to increase the reusability of each segment and understandability of the overall architecture and to respond to the dynamically changing environment and requirement more easily.

5.3.4. Iterations

For each component in a segment that is not developed yet, conventional software development process is repeated. During this phase, we reject weighted documentation in order to make progress rapidly, but we recommend simplified artifacts for specifying important changes, feedback, and progress.

5.3.4.1. Scenario Based on the architecture of the target system, we have to define the exact scenario. In scenarios, the composition of the mobile application's elements, functional flows, and interactions among the elements is described. In the simple case of just one mobile application, we can draw the scenario as a flow of views and processes. In more complex case of composite applications, we can draw interactions among components with some objectives and functions. Here is an example of developing the family applications for a bank company. There should be some considerations of security, identification, and so on. Each of these parts could be divided into some smaller components. In addition, if there are plans for adaptation, then variable points and variants are also described in the scenario. Basically, the variability description is composed of a variable point and variants. Usually, variable points are caused by information processing as seen in Figure 7. In this case, for adaptation, the information analysis process could be replaced by the agent. At the end of this step, a conti is produced as a result. A simple example of conti is shown on the case study of this paper, in Figure 8.

5.3.4.2. Requirement analysis In this step, the functional and non-functional requirements for each component are described. In particular, requirements for

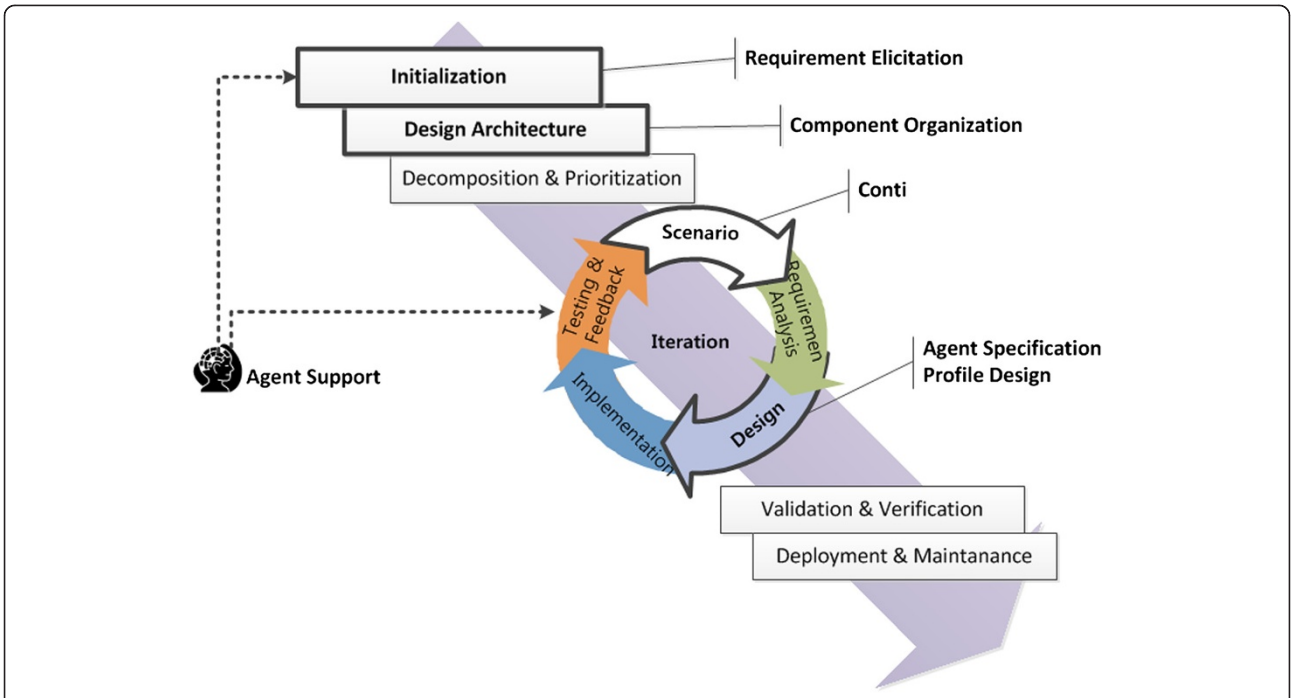

Figure 4 The process of an agent supported mobile software development methodology based on agile.

adaptation have to be described based on the distinct situation. Also, the attributes of agent, which is used in the target system, have to be preliminarily described for agent specification (e.g., reactivity, pro-activeness, social ability, and autonomy [9]).

5.3.4.3. Design For every component in the prescribed scenarios and requirements, actual data and functional structure are defined. Components are specified by the features of the agent - generally goals and behaviors. Moreover, profiles about the system and users related to adaptation and functions should be designed for each adaptation plan, which is described in the previous step. Several kinds of profiles have to be designed for adaptation. The type of profiles could be distinguished as shown in Table 2.

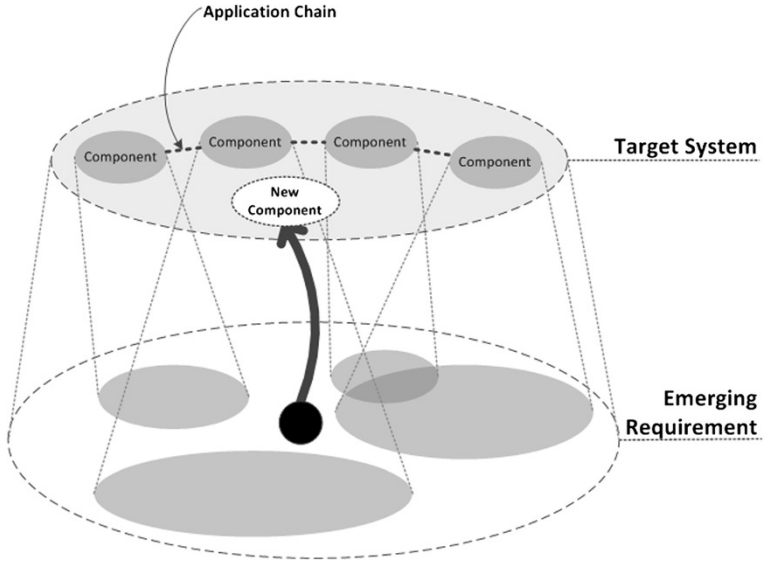

Figure 5 The emerging requirement elicitation.

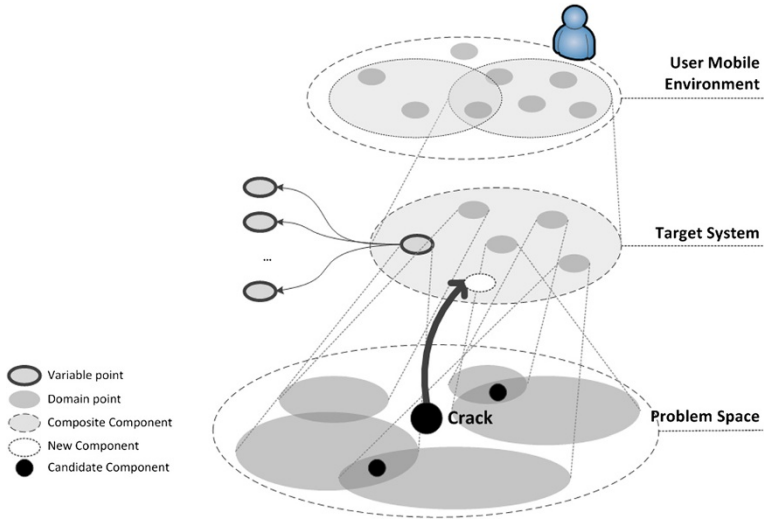

Figure 6 Designing architecture.

General profiles include the shared information that could be accessed by all components. It usually includes the general information about users and the system that could be used to extract the implied information, like patterns or the context. This general information has to be provided by the mobile framework. If additional information is needed for the component's adaptation, then the restricted profiles will be designed for an extension of information.

5.3.4.4. Implementation In this step, the designed components are implemented. This implementation is restricted to the previous design result. If new attributes or requirements occur in this step, the process needs to go back to design step. No functionalities, which are not described in design step, should be implemented. As this methodology is employing an agent module, the agent is also implemented in this step. This agent module is made using the predesigned specifications, models and platforms, and profiles.

5.3.4.5. Testing and feedback For a present component, engineers verify the agent in a syntactic and semantic way. If it passes the formal test, then engineers will deliver the latest version of the component to users and collect reactions and feedback. At the end of this step, developers analyze artifacts of previous steps, and then make a choice whether they should go through iteration again or go over to the next iteration for the other components. During this phase, the agent-supported

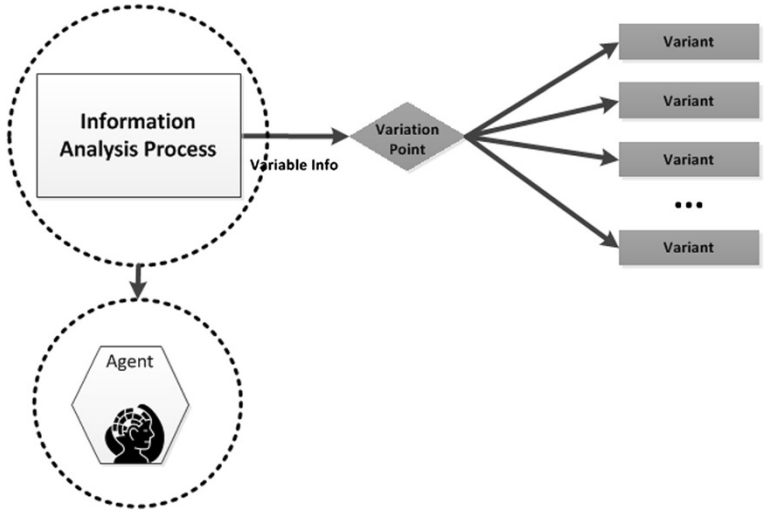

Figure 7 A conti example for adaptation.

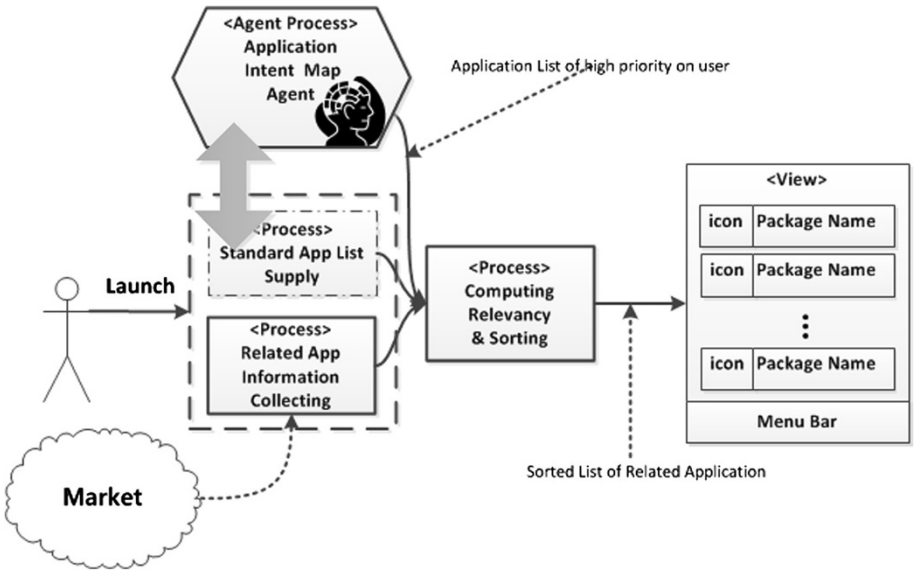

Figure 8 Conti for experiments.

development environment could conduct the formal test on the components and collect the results of the testing and feedback instead of the team. Consequently, the specifications of changed requirements, the revisions for the satisfaction, and progress should be documented.

5.3.5. Verification and validation

For the target system operated by all working components, engineers must ensure the satisfaction of the original purpose and objectives they planned in the first step of this process. They also should check whether the combined target system is adapted in the predefined contexts.

5.3.6. Deployment and maintenance

After the above steps are done, the goal product is made. In this step, the deployment of the product and its maintenance are executed. To keep the usability of the product and satisfaction of users, the team has to make a window for continued communication with users and the changes of the market. If needed, they could go back to the iterations and do their process again.

Table 2 Type of profiles

Type	User profile	System profile
General	General information over applications about a user and the user's life logs, including patterns and the context of the user.	General information over applications about a mobile system, including the state and the context of environment.
Restricted	The restricted information collected and analyzed by the component for adaptation. These profiles are used for more sophisticated information processing.	

When we comply with these described whole processes, some artifacts are made for each phase, as shown in Table 3.

5.4. Framework

Our mobile software development methodology includes technical agent supports to help the process of the adaptation and test. Agent technology can support information processing on the MCE. The supported environments are largely the development environment and the mobile software environment. As seen in Figure 9, the in-process information during the development and use of applications is collected and analyzed by information-managing components, and the newly produced information is used for developers to implement a new application and for mobile applications to adapt. This process of information collecting and processing can be executed in real-time by the help of the agent, who supports the information lifecycle. Thus, autonomous knowledge-based agent technology is appropriate to support these distributed real-time knowledge intensive processes.

Consequently, previous principles, processes, and technical support compose the framework for developing a mobile software. As shown in Figure 9, our adaptive mobile software development framework adopts the simplified principles based on agile, the process of AOSE, and the agent technology support. The five principles of agile are well matched to the features of MCE and made iterative processes based on that. Each step of the process is based on the AOSE and is supported by an agent technology with labor- and knowledge-intensive processes. Moreover, we propose agent technology support adaptations of mobile software.

Table 3 Artifacts of the process

Process	Artifacts
Initialization	Specification of ideas, category, and objectives of target system through emerging requirement elicitation
Design architecture	Agent oriented architecture of the target system
Decomposition and prioritization	Segments and priorities of components composing the target system
Iteration	
Requirement analysis	Functional/non-functional requirements specially based on mobile constraints and adaptation
Scenario	Conti, including variability
Design	Design including the specification of an agent and provision of profiles
Implementation	Implementation of specified components including agent
Testing and feedback	Specification of requirement satisfaction, testing reports, prototype evaluations and plan for revisions, including agent verification
Verification and validation	Verification and validation of the target system
Deployment and maintenance	Products and user guide

6. Case study

In this study, we have made an experiment referred to case study research design components [19] to show the effectiveness of an agent support and illustrate a way to take into account mobile user customization functions. Thus, we have planned two applications, which are to recommend new services according to the user's behavior history in the mobile environment. One was developed by the classic development methodology, and the other was developed by the proposed mobile software development process.

6.1. Objectives and propositions

This experiment has two main objectives. One is to show that the utilization of an agent support is valuable. The other is to show that the agent can actually reduce the complexity of the development process. Therefore, the following statements are considered in this experiment:

- Reflecting the user customization requirement;
- Reducing loads over the developers;
- Improving accuracy and functionalities.

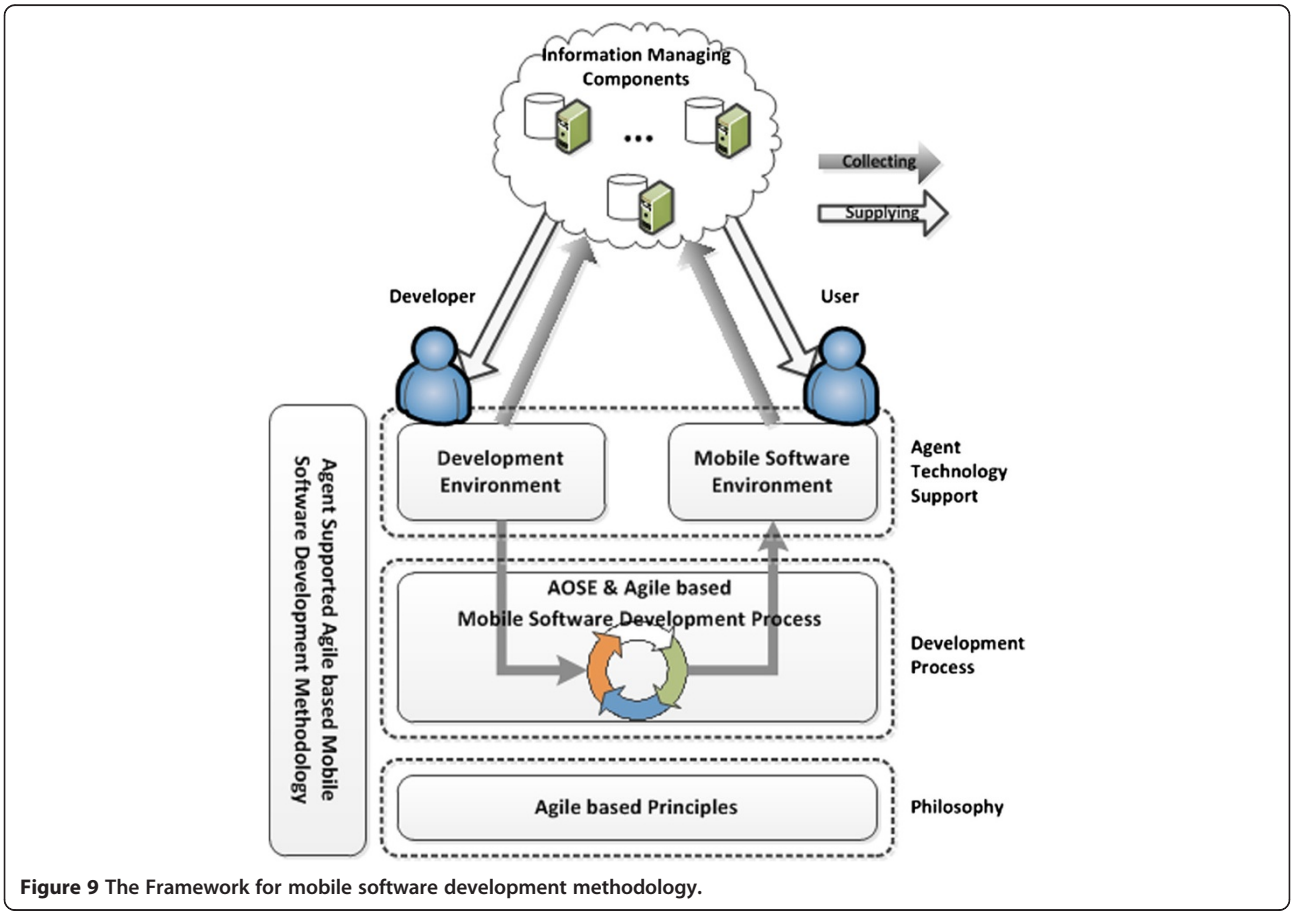

Figure 9 The Framework for mobile software development methodology.

Concerns for developing customization functions were reflected to the proposed process to make the process easier. Through agent support, we can reduce the loads on the information processing aimed at customization. Lastly, the accuracy and functionality will be improved through the capability of agent reasoning. The evaluation metrics for these propositions are shown in the next chapter.

6.2. Metrics

Table 4 describes the metrics and comparison of classic software development methodology with our proposed methodology. In this experiment, we have focused on the effectiveness of agent support. So, we have used four attributes in these metrics. These attributes are 'level of difficulty', 'cost of time and labor', 'customization usability', and 'capability of information processing'.

6.3. Experiment

In this experiment, we have made two applications. These applications have the same function which is to recommend new services to be used. One application is developed with agent support in managing intent map, and the other is developed without the support. In this section, we will explain the core concepts and results of customization.

6.3.1. Scenario

In the proposed methodology, we introduce some concepts for customization and adaptation requirements. Among them, we have adopted three concepts: conti, variable points, and profiles. Firstly, our program has a simple structure composed of three processes and just one view as seen in Figure 8. As you see in this conti, this scenario is composed of units in mobile applications. Generally, units of android applications are recognized as layouts or activities. However, the more general terms need to be used. Among these processes, the standard app list supply makes variability in this application. According to the information supplied by this process, the actual list of recommended services for the user is decided. Though it does not have visible variants, it has actual variability at that point. For the first application, a standard service list is collected from the user's mobile environment. For this, we used the list of present launched services beforehand. As the proposed methodology is used, this process is replaced by an agent module. In this experiment, application intent map agent, which has information about each service's user priority, is the agent module. Through this replacement, we can get more customized and meaningful information to be used as standard for extracting related services.

6.3.2. Design

As a part of the design, we defined two profiles: intent map and application relation map. An intent map is a kind of general system profile supplied by an application intent map agent. This profile is composed of applications, their counts of launch, and counts of launching dependency as seen in Figure 10. We implemented the agent to calculate the priority of the applications of the user based on a PageRank algorithm. The gray circle

Table 4 Metrics for evaluation of methodology

Before	Metrics	After
Difficulties in designing information processing for customization exist	Level of difficulty	Through agent reusing, length and complexity of process for design are shortened.
The amount of cost to develop functions instead of agent will be increased.	Cost of time and labor	The cost can be reduced by the reusability of the agent.
By limitation of information closed to users and capability of reasoning, usability of customization is bad.	Customization usability	By agent, collecting information and reasoning from the information are possible. Thus, the usability of customization increases.
Limitations of collecting the user's life information exist. All methods of information processing are implemented by developers each time.	Capability of information processing	Through agent support, we can get huge amounts of information about users and user behavior. Agent reasoning can discover meaningful information remarkably from data.
Generally, as the software development methodology is aimed at reducing the costs from the changing of designs and requirements, the process will fluctuate during the mobile application development.	Change tolerance	According to iterative and evolutional processes, changes can be easily accepted. Moreover, reducing the load of preparation before the actual implementation reduces the cost of revision.
Mobile constraints can be missed.	Quality of mobile apps	Through the reflection of mobile application requirements in the process, the possibility of missing concerns related to the quality is reduced.
By the complexity of high level functionality development, possibility of error is high.	Possibility of error	By using agent supported reasoning and specific algorithms, the possibility of error will be reduced.
Independency of components and the stiff structure of the application require the participation of developers for functional extension.	Scalability	Scalability and flexibility of agent reduce the cost of development for functional extension.

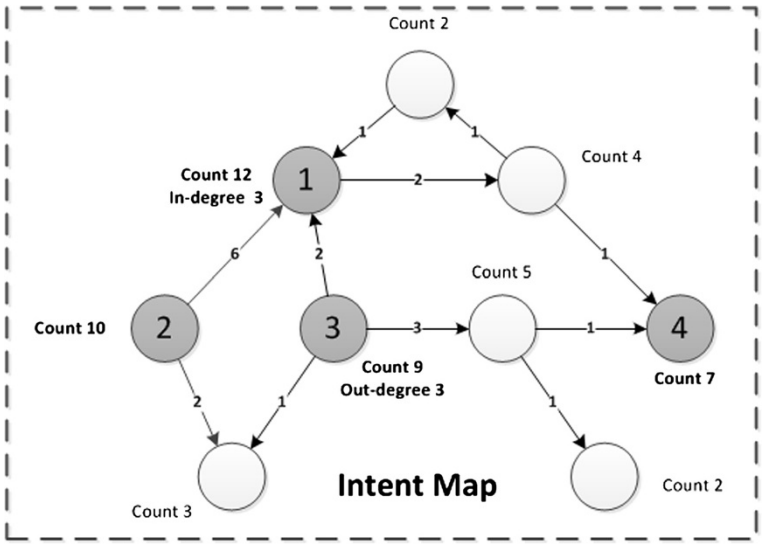

Figure 10 An intent map as a general system profile for experiments.

seen in Figure 10 represents the applications that have high priority and will be used as the standard for looking for related applications.

An application relation map is a kind of app-specific profile composed of applications with connections to each other as seen in Figure 11. The link between applications means that many users using one application can also use another application together. The former picture shows that the candidate has duplicate relations with some applications. The later one shows that the linear relation can be shown in a family product.

Consequently, in the first experiment, we simply get the list of currently used applications as standards and then gather additional applications related to the standard applications, but in the second experiment, we use the agent computing with an intent map and make a list of most important applications to the user, instead of just utilizing the currently used applications list.

6.3.3. Results

Figures 12 and 13 show the results of our experiments. As seen in Figure 12, recommended applications are actually not usable for the users in some cases. Specifically, users who generally use the bank and card company applications also use another application showing arriving bus information. The services are recommended to this user at the given time. However, the wrong service, which is for C university students, is recommended to this user who is a student of A university. In Figure 13, for the second experiment, the application with agent support keeps the application that has a high probability of use at the top of the list.

As a result, some concepts like conti, variable points, and profiles for customization and adaptation are reflected in the process of this experiment. Loads, by implementing the process extracting high priority applications from the user, are reduced using agents instead of people. Lastly, accuracy and usability of application have improved through the utilization of meaningful information from an agent. Consequently, in these experiments, we show three propositions, which are explained in Section 6.1.

Actually, these applications do not always recommend fantastic applications to the user. The reason is that the information is not enough to realize the taste of the user. The information about each service's user priority is not enough to make a practical user customization. However, we expect that it would be possible to get information and increase the accuracy of adaptation if the infrastructure is constructed to support an information lifecycle.

7. Discussion

In this research, we attempted to reflect the requirements of the MCE on the methodology and to solve the problems caused by variability, approaching from the hybrid method of combining agile software development methodology and agent technology to make the process effective. Also, we used concepts of SPLE and AOSE to control the complexity of engineering process and to facilitate the realization of adaptation. Lastly, we made experiments to show the potential of agent support in complex mobile developments, e.g., customizing and adaptive mobile software. However, we still have drawbacks in our process, which could yield to further studies as follows:

- Standard specification of agent for MCE for adaptation

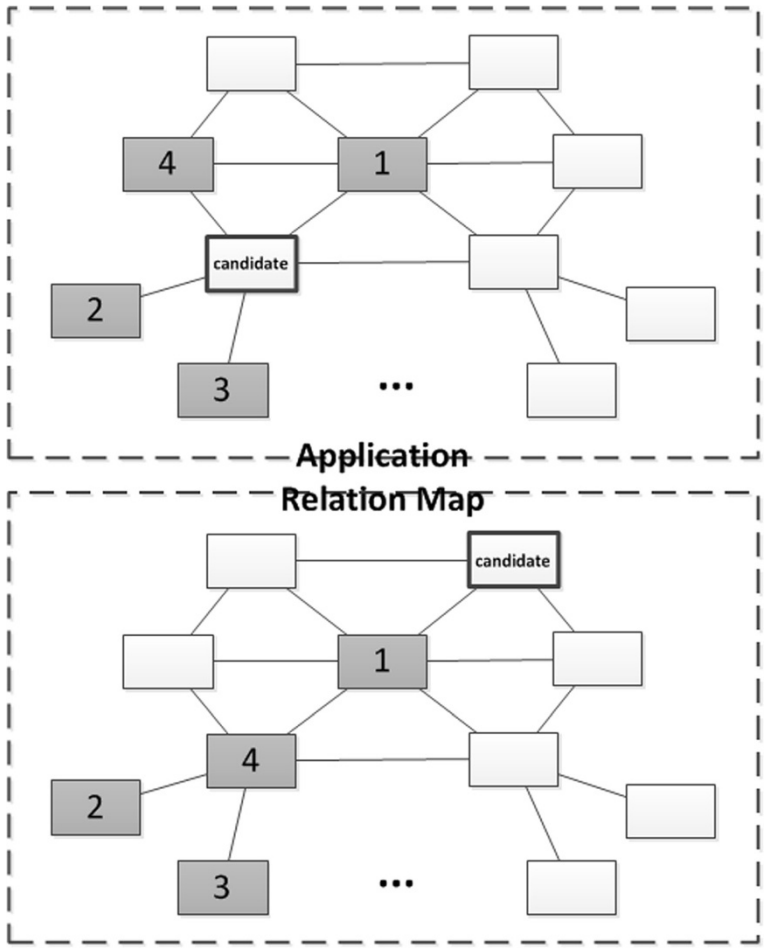

Figure 11 An application relation map as an app-specific profile for experiments.

- Autonomous agent verification and validation
- Evaluation of adaptive mobile software
- Realization of agent-supported development environment
- Agent-supported mobile software platform.

Moreover, mobile software engineering and an AOSE approach for customized/adaptive MCE are scarce. However, we are on the stepping stone to adaptive and tailored functions for user and environments. So if we face the needs of an adaptive MCE and the difficulties with complex mobile software engineering, then we could learn more about issues such as the following:

- Organization of components in MCE
- Collaboration between applications and mobile components in MCE
- Adaptation of mobile application and complex MCE
- Infrastructure to supply tailored service and information processing.

8. Conclusion

In this paper, we described the need for an agile-based software development methodology for dynamically changing requirements of mobile software in wireless networked computing environment. This approach can be used to provide automatic computation support for suitably decomposed tasks.

The proposed methodology includes these two features. One is the requirement discovery and the use of variable point concepts in SPLE for the representation of adaptive software requirements. The other is a process satisfying the changing requirements and accommodating the information lifecycle. In the designing of the methodology, we reflected the adaptation requirements as well as the constraints of MCE onto the methodology. Finally, using those concepts, processes, and technical support, we proposed a framework for mobile adaptive software development in wireless computing environment. Moreover, we also described the possible effects of agent support by performing two experiments.

Figure 12 Result of first experiment without agent support.

As an experiment, we found a fragmentary scenario to understand the effect of agent support. From this test, we could realize the advantages of the infrastructure which supports the proposed methodology.

In short, what we would like to show in this paper are like these. One is a new mobile software development methodology, which reflects the feature of wireless computing environment. Another is the mobile software development framework based on hybrid solution of agile philosophy and agent technology. The other is the incorporation of 'requirements discovery' stage which is a new unit of mobile software development process to support the need of software adaptation. With these objects, the design of a human-centered software development methodology and framework for mobile software development in wireless computing environment is proposed in this paper.

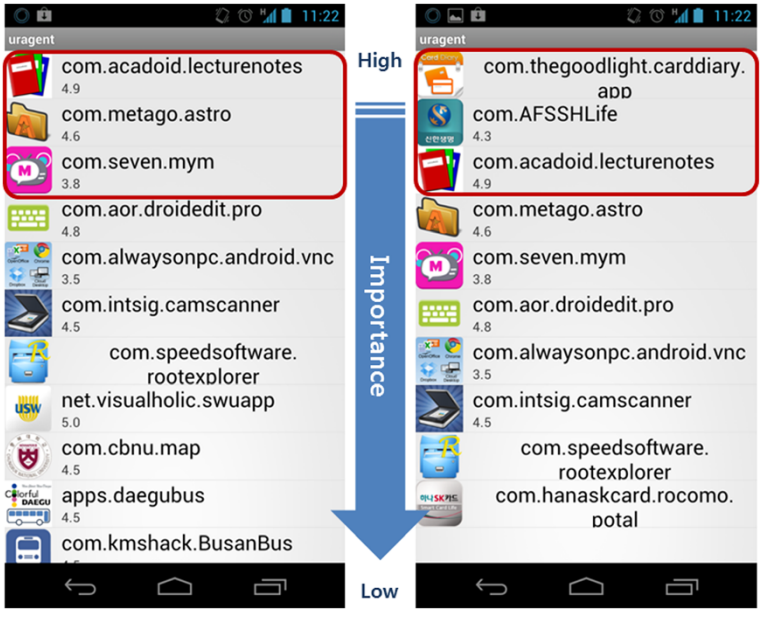

Figure 13 The results of a second experiment with agent support.

Competing interests
The authors declare that they have no competing interests.

Acknowledgment
This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012M3C4A7033343 and No. 2012M3C4A7033346).

Author details
[1]Graduate School of Computer Engineering, Ajou University, San 5 Woncheon-dong, Youngtong-gu, Suwon-si, Gyeonggi-do 443-749, Republic of Korea. [2]Department of Information and Computer Engineering, Ajou University, San 5 Woncheon-dong, Youngtong-gu, Suwon-si, Gyeonggi-do 443-749, Republic of Korea.

Received: 17 March 2013 Accepted: 3 April 2013
Published: 24 April 2013

References

1. V Rahimian, R Ramsin, Designing an agile methodology for mobile software development: a hybrid method engineering approach, in *Second International Conference on Research Challenges in Information Science (RCIS)*, ed. by O Pastor, A Flory, JL Cavarero (IEEE, New York), pp. 337–342
2. P Abrahamsson, O Salo, J Ronkainen, J Warsta, *Agile Software Development Methods* (VTT, Espoo, Finland, 2002), p. 112
3. L Rising, NS Janoff, The Scrum software development process for small teams. IEEE Software **17**, 26–32 (2000)
4. WA Wood, WL Kleb, Exploring XP for scientific research. IEEE Software **20**(3), 30–36 (2003)
5. T Wasserman, *Software Engineering Issues for Mobile Application Development* (FoSER, New Mexico, USA, 2010), pp. 397–400
6. P Abrahamsson, A Hanhineva, H Hulkko, T Ihme, J Jaalinoja, M Korkala, J Koskela, P Kyllonen, O Salo, *Mobile-D: an agile approach for mobile application development* (Paper presented at the 19th annual ACM SIGPLAN conference on object-oriented programming systems, languages, and applications, Vancouver, BC, Canada, 2004). 24–28 October
7. H Mubarak, Developing flexible software using agent-oriented software engineering. IEEE Software **25**(5), 12–15 (2008)
8. NR Jennings, M Wooldridge, Agent-oriented software engineering, in *Proceedings of the 12th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems: Multiple Approaches to Intelligent Systems, Cairo, Egypt*, ed. by IF Imam, Y Kodratoff, A El-Dessouki, M Ali (Springer, New York, 1999), pp. 4–10
9. M Wooldridge, Agent-based software engineering. IEEE Proceedings on Software Engineering **144**, 26–37 (1997)
10. MJ Wooldrige, NR Jennings, Software engineering with agents: pitfalls and pratfalls. Internet Computing, IEEE **3**(3), 20–27 (1999)
11. R Srinivasan, Artificial intelligence methodologies for agile refining: an overview. Knowledge and Information Systems **12**(2), 129–145 (2007)
12. M Wooldridge, *An Introduction to MultiAgent Systems* (Wiley, New York, 2009), p. 484
13. MP Georgeff, B Pell, ME Pollack, M Tambe, M Wooldridge, *The belief-desire -intention model of agency* (Paper presented at the 5th international workshop on intelligent agents V: agent theories, architectures, and languages, ATAL '98, Paris, France, 1998). 4–7 July
14. AS Rao, MP Georgeff, BDI agents: from theory to practice, in *Proceedings of the First International Conference on Multiagent Systems*, ed. by L Gasser, V Lesser (AAAI, Menlo Park, CA, 1995), pp. 312–319
15. MJ Huber, *JAM: A BDI theoretic mobile agent architecture* (Paper presented at the proceedings of the third annual conference on autonomous agents, Seattle, WA, 1999). 1–5 May
16. MJ Huber, *JAM manual (IRS, 2001)*. http://www.marcush.net/IRS/Jam/Jam-man-01Nov01.doc. Accessed 1 November 2001
17. I Rahwan, R Kowalczyk, Y Yang, PRICAI 2000 Workshop Reader, Virtual enterprise design—BDI agents vs. objects, in *Advances in Artificial Intelligence*, ed. by R Kowalczyk (Springer, New York, 2001), pp. 147–150
18. L Williams, What agile teams think of agile principles. Commun ACM **55**(4), 71–76 (2012)
19. SW Lee, DC Rine, *Case study methodology designed research in software engineering methodology* (Paper presented in the proceedings of the sixteenth international conference on Software Engineering and Knowledge Engineering (SEKE'04), Banff, Alberta, Canada, 2004). 20–24 June

doi:10.1186/1687-1499-2013-111
Cite this article as: Eom and Lee: **Human-centered software development methodology in mobile computing environment: agent-supported agile approach.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:111.