A Storyboard-based Mobile Application Authoring Method for End Users

Jun-Sung Kim, Byung-Seok Kang and In-Young Ko
Dept. of Computer Science, Korea Advanced Institute of Science and Technology (KAIST)
291 Daehak-ro, Yuseong-gu, Daejeon, Korea
{junkim, byungseok, iko}@kaist.ac.kr

Abstract—Mobile computing focuses on supporting everyday activities of users by providing services that utilize mobile computing resources. More diverse types of users and computing resources have been engaged in mobile computing environments. Considering these characteristics, it is essential to support making end-users actively participating in mobile computing environments based on high-level goals of the users. To meet these requirements, we have devised a storyboard-based application authoring method. The main elements of this approach include a storyboard model to structure the mobile applications and a semantically-based abstraction method to represent complex applications in terms of abstracted scenes in a storyboard. This approach improves the existing visual programming paradigms which mostly focus on visually composing fine-grained programming elements. We have developed a prototype implementation of the application authoring tool and tested it with a group of users to prove the effectiveness of allowing end-users to create and manage mobile applications.

Keywords - Mobile Application Authoring; Application Storyboard; End-user Software Engineering; Visual Programming.

I. INTRODUCTION

To achieve the user-centricity goal in providing mobile applications [1], new computing paradigms such as service-oriented computing (SoC) and task-driven computing (TDC) have emerged [2][3]. These approaches focus on separating the concerns of selecting and coordinating specific services from the concerns of recognizing users' high-level computing goals. These allow end-users to more easily interact with numerous computing resources in a mobile computing environment with their high-level task goals. However, SoC and TDC in mobile computing have been considered mostly within the context of "use of applications" rather than "authoring of applications". Therefore, it is normally difficult for end-users to define their task goals and to arrange and access mobile computing resources that are necessary to accomplish their goals.

In this paper, we propose a storyboard-based mobile application authoring method by which end-users can intuitively specify their task goals as a storyboard and easily generate a mobile application from the storyboard. In a storyboard, users can specify the necessary functionalities and structure of an application as high-level activities (scenes). Once a mobile application is created, it can be validated, executed, evaluated, and personalized.

Our approach also allows users to identify and reuse common patterns of defining storyboards to accomplish a type of goal. The main elements of our approach include a storyboard model to structure the mobile applications and a semantically-based abstraction method to represent complex applications in terms of abstracted scenes in a storyboard. Fig. 1 shows the overview of the storyboard-based mobile application authoring method.

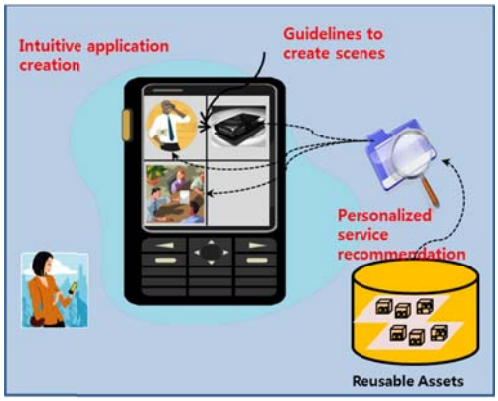


Figure 1. Overview of the storyboard-based mobile application authoring

Scaffidi *et al*. reported that most of the computer users these days are non-professionals who do not know much about conventional programming [18]. In mobile computing environments, there is even a bigger proportion of non-professional end-users.

End-user software engineering is a paradigm of allowing end-users to develop software to meet their own needs while bridging the gap between their high-level requirements and detail system capabilities [4]. Our approach provides an end-user software engineering framework that covers the overall lifecycle of software development including requirement specification, application generation, evaluation, and evolution. In comparison to the existing visual programming paradigms which mostly focus on visually composing fine-grained programming elements, the storyboard-based authoring method enables end-users to represent their requirements in a higher-level abstraction.

This paper is organized as following. We introduce the major requirements of end-user mobile application authoring and explain the related works in Section II and Section III, respectively. Section IV and Section V describe the core approach of the storyboard-based authoring. Section VI shows the evaluation results. Finally, Section VII concludes the paper by explaining the main contributions and future works.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9

II. REQUIREMENTS OF END-USER MOBILE APPLICATION AUTHORING

A. User-centricity – Task-driven abstraction and visualization

In a mobile computing environment, it is especially crucial for end-users to access mobile computing resources based on their task goals without considering any technical details. End-users need to be able to focus on describing what they need to achieve their task goals rather than expressing the detail structure and functions to be implemented in their applications. In addition, there must be a visual aid to allow users to intuitively represent their task requirements and to understand the core activities to be supported by a mobile application.

B. Efficiency – Automated and non-error-prone authoring and instantiation processes

Most end-users are non-programmers who do not have enough technical skills to create, recognize and compose services and to monitor applications [5]. Therefore, it is essential to minimize such technical burdens of users by automating the process of selecting and combining component services that are necessary to accomplish a task goal. In addition, there must be a mechanism of bridging the gap between a high-level task representation and a set of services available in a mobile computing environment, and making automated bindings between those two different abstractions.

End-users' authoring activities are normally error prone. Therefore, the authoring process should support the ways of resolving mismatches between required capabilities and available service functions, and validating an application against a user's task goal.

C. Reusability and Evolovability – Reuse of common application patterns and support of application evolution

In mobile computing environments, there is a wide spectrum of applications to be supported for diverse types of users. In these environments, reuse of common application patterns and evolution of applications based on users' feedback are critical to reduce development efforts of applications and to improve the quality of applications [11][12]. Therefore, it is necessary to provide a mechanism to identify a common structure and functionalities to support a similar set of user tasks and to enable these common application patterns to be refined and extended based on users' feedback. The application instantiation process also needs to meet these requirements by providing a mechanism of reusing successful task-service bindings for similar situations, and by making these binding patterns evolvable.

D. Mobile Usability – Usability support in user-interface-constrained mobile computing environments

Mobile usability is about allowing mobile users to effectively interact with an application by using User Interface (UI)-constrained computing environments such as smart phones and tabular PCs. The end-user mobile application authoring environment should support mobile usability such that users can effectively represent and recognize the core structure of task activities by using their mobile devices. The granularity of UI elements that comprise a task-driven abstraction of an application needs to be coarse-grained enough to be efficiently visualized on a small screen, and to be controlled by using constrained input methods. Especially, it is essential to make the high-level UI abstraction of an application consistent with the detail application integration structure [6].

III. RELATED WORK

End-user software engineering is a paradigm to allow end-users to create and manage software applications without having deep programming knowledge and skills [4]. Many approaches have been developed to help end-users conduct various development activities throughout the software development lifecycle.

A. Flowchart-based Approach

In this approach, end-users can draw a flow of component services and conditions by using a common format or template provided by developers [13][14]. Although this allows users to structure an application based on the main flow of activities and events that are important in a specific domain, it is often difficult for end-users to learn and understand detail notations (branches, loops, etc.) and options (e.g., sequential structure vs. parallel structure) to represent a flow.

The activities in a flow are normally represented at the same abstraction level as component services. Therefore, users need to associate each activity to a specific component service to be used. It is usually a difficult job for end-users to recognize, select and compose component services with understanding their functionality and other technical factors such as interfaces, preconditions and post-conditions. In addition, the detail flow structure cannot be shown effectively on a small screen of a mobile device.

B. Wizard-based Approach

In this approach, end-users can create and customize applications by creating forms and representing dialogues that are needed to be used in user interactions [15][16]. The step-by-step dialogue sequence and appropriate forms to provide at each step can be specified in the application definition. The wizard-based approach relatively does not need users to understand complex notations and technical factors. However, this approach requires careful modeling of the forms and dialogues. The forms need to be modeled such that the users can easily understand the desired inputs to be provided at a step. In addition, detail conditions and branches of the steps cannot be easily programed by end-users. In this approach, some error-toleration features can be incorporated to ensure the quality of data filled in a form.

C. Spreadsheet-based Approach

This approach allows end-users to create applications by putting values and assigning computations to designated cells in a spreadsheet [19]. Since spreadsheets are widely used by people, end-users can easily learn how to make applications by editing cells in a spreadsheet. In addition, some features to improve the dependability of application can be supported by adding interactive and dynamic testing capabilities described in [19]. However, the types of applications that can be developed by using this approach are limited to the ones that require management and computation of data in a tabular form. In addition, computational rules are normally hidden behind the

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9

visual representation of a spreadsheet, and novice users may have difficulty of creating and validating them.

D. Storyboard-based Approach

A storyboard is a series of visual illustrations sequentially arranged and displayed. It has been widely used in the movie, advertisement, and multi-media domains. There have been some attempts to use storyboards for designing and creating applications [9][10][17]. In this approach, service functions are abstracted to 'scenes' that can be arranged into a storyboard. A storyboard visualizes the structure and semantics of an application, and provides users with a series of interfaces to interact with the application. To create an application, end-users firstly identify available service functions via predefined mockup scenes, and then select and arrange the scenes in a storyboard template. Users can understand the semantics of an application by interpreting the sequence of the scenes selected.

This storyboard-based approach provides end-users with an intuitive interface to select and compose component services. In addition, a mockup scene can be used to effectively visualize the essential functionality of a component service. In addition, a storyboard that is composed of multiple scenes can be visualized and browsed effectively on the small screen of a mobile device.

However, similar to the wizard-based approach, most of the storyboard-based approaches cannot visualize and control detail application structures. In addition, large-scale applications that need to be composed of many scenes arranged in various structures cannot be efficiently created and managed by using storyboards. Some researchers have tried to overcome the limitations by providing detail structure representations on storyboards. However, the complex storyboard representations increased the difficulties of understanding and controlling component services to be accessed to accomplish a user task. In other words, although we can represent some control structures such as loops, branches and parallelism in a storyboard, a mobile application that is presented in a storyboard becomes too complex to understand and manage by end-users.

In each scene of a storyboard, users need to be able to represent their computational needs in their own perspective. However, most storyboard-based approaches lack the ability of hiding the technical details of component services and hardware devices in an environment. In addition, most of them do not support the reuse of existing storyboards to make new mobile applications based on previously defined compositional patterns. Without the support of finding and reusing existing storyboards, it is hard for the end-users to represent a service composition from scratch with considering different candidates of services in a local environment and the dynamically changing context of using an application. As we discussed in Section II, it is necessary for end-users to find most appropriate patterns of making storyboards for their needs, and generate an application by simply extending and customizing them. The previous storyboard approaches also do not provide facilities to validate the correctness of a service composition generated from a storyboard.

IV. Storyboard-based Mobile Application Composition Model

We have developed our storyboard-based application composition model based on the task-oriented application framework [20]. As depicted in Fig. 2, the model is composed of three different views: visualization model (called U-board), task meta-model, and instances.

The *visualization model* is for allowing end-users to describe their computing tasks as a story which is composed of multiple activity scenes. The *task meta-model* defines a decomposition structure of user-centric applications. A task is created for each computational goal of a user, and composed of unit tasks each of which defines a compositional pattern of services for performing an action in the task. A unit task defines a set of necessary component services and the interconnection structure among them. The *instances* are actual instances of applications and services that can be run in a local environment. Instances of a task are generated by considering system specific characteristics and environmental conditions.

As shown in Fig. 2, the task meta-model maps the user-centric visualization into the elements of an application instance. All the entities in our models are described and managed by using ontologies. In other words, the semantics of task stories, scenes, tasks, unit tasks, and services are described by using domain specific ontologies, and their semantic relationships and similarity can be inferred by using a reasoning engine [20]. In this section, we focus on explaining the content of the visualization model.

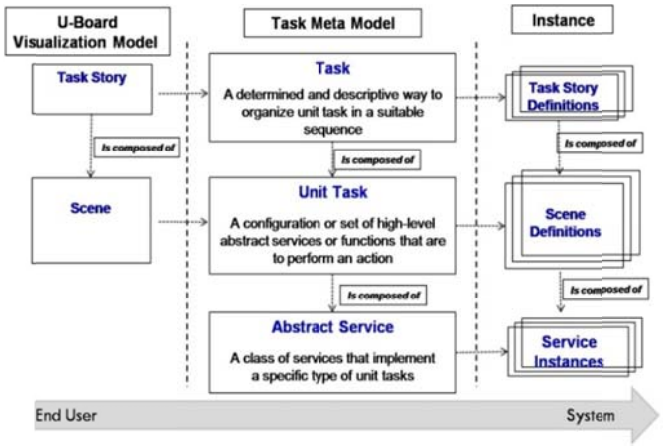


Figure 2. Three views of the storyboard-based mobile applcation composition model

A. Scene Visualization Model

As we discussed in Section II, each scene of a storyboard needs to be highly readable even on mobile devices. A shown in Fig. 2, a scene is composed of three parts: activity name, representative image, and context information. The activity name is a textual name of a scene, and the representative image visualizes the essential characteristics of a scene such as an object, movement, and operations. The context information represents the temporal and spatial context in which the scene is activated and becomes valid.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9

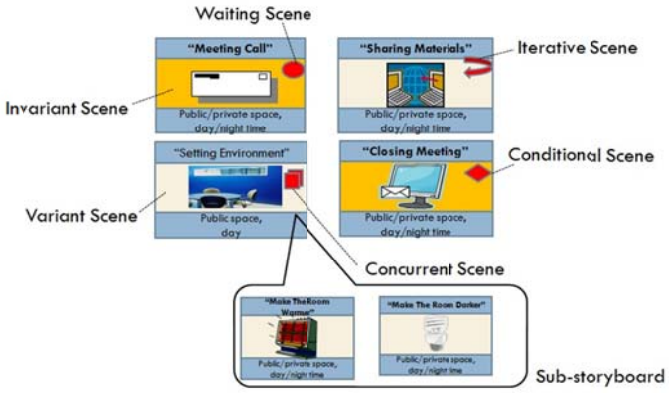


Figure 3. Examples of scene visualization (the scenes defined in the task story called 'Having a Meeting')

B. Scene Types

A task story can be composed of scenes that are either invariant or variant scenes. *Invariant scenes* are static parts of a story that are not changed over time and are common across different situations. At the example story shown in Fig. 3, 'Meeting Call' and 'Closing Meeting' are the scenes (marked with the yellow background) that are common in every meeting tasks, and defined as invariant scenes. *Variant scenes* are the ones that can be replaced by an alternative scene based on user preferences and environmental conditions. In the example above, 'Sharing Materials' and 'Setting Environment' are variant scenes that can be substituted to an environment-specific or customized scene. For example, 'Shareing Materials' can be replaced by a scene of exchaning secure emails to handle critical information. In most of the cases, variant scenes are replaced by personalized or more specialized scenes automatically based on user preferences or environmental conditions. This is to meet the efficiency requirement explained in Section II.

Scenes are also categorized into four groups based on their control structures: sequential, concurrent, iterative, and conditional scenes. The scenes that are arranged in a storyboard are *sequential* in default. The *concurrent scenes* that need to be executed in parallel can be grouped together into a scene as depicted in Fig. 3. The compound scene can be expanded into multiple, concurrent scenes by clicking on the control icon (overlapped rectangles). Scenes with a curved arrow are *iterative scenes*, which services are executed iteratively while a condition is met. The iteration condition of a scene is represented as a set of properties, and can be checked by double clicking on the curved-arrow icon. A diamond icon that is shown on a scene means that the scene is a *conditional scene*. A conditional scene is activated when a condition, which is represented as a set of properties, is met.

In our approach, control structures can be imposed only on each scene rather than across multiple scenes in a storyboard. Although this limits the representation power of controls, our simple and graphical controlling mechanism makes the authoring and management of storyboards much simpler and easier for end-users.

V. IMPLEMENTATION

We have implemented a prototype of our end-user mobile application authoring tool. Fig. 4 shows screen shots of the tool. The main screen shows a storyboard canvas and a task radar. The task radar allows users to visually control some contextual aspects such as spatial, social, temporal and personal aspects to narrow down the candidate storyboard to reuse for a task. The tasks found are shown in a hierarchical structure (based on their ontological relationships) in the task browser. When a task story is selected from the task browser, the sequence of scenes is displayed in the storyboard canvas, and the detail list of scenes and their decomposition structures are shown in the scene browser.

Our tool is developed by using Java SWT. We used jfreechart 1.0 for implementing the task radar. This client is installed in two ultra-mobile PCs (SAMSUNG Q1, Fujitsu U2010). In addition, the semantic descriptions of unit tasks and task stories are made by using the Protégé ontology editor [7]. The semantic reasoning of finding storyboards and scenes are implemented by using the Jena library [8].

By using this tool, a user can define an initial task story by finding and selecting a storyboard template that is most appropriate for his or her task. Then, the end-user can customize the initial task story by rearranging, adding, and deleting scenes on the storyboard. A task story is defined by arranging a set of scenes each of which has its URI, name, control structure type, variability conditions, and contextual properties.

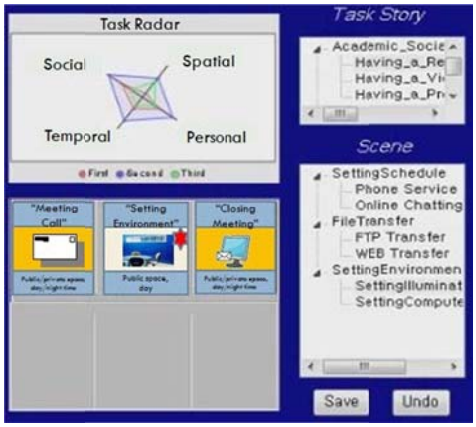


Figure 4. Layout of the mobile application authoring tool

While a task story is defined, the application authoring tool analyzes the relationships among the scenes and automatically suggests necessary modifications on the task story if it detects mismatch between inputs and outs of consecutive scenes or contextual inconsistency among scenes.

Once all authoring steps are finished, the end-user can hit the 'save' button to initiate the process of converting the storyboard representation into a concrete service composition that can be executed in the local environment. The storyboard is saved with some semantic description and can be found and reused for similar purposes.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9

VI. EVALUATION

To evaluate our approach whether it meets all the requirements to enable end-users to create mobile applications, we conducted a user test. We recruited sixty users and provided them with an operation manual (Table I) of the application authoring tool. The users are mostly graduate students at our school, but only few of them are expert programmers. To compare our approach against the existing storyboard approaches, we divided the sixty users into three groups. Each group participated in evaluating one type of approach to avoid learning effects because we measured the time to finish the customization of a task story according to a given scenario and counted the number of steps to accomplish the job.

TABLE I. OPERATION MANUAL OF THE APPLICATION AUTHORING TOOL

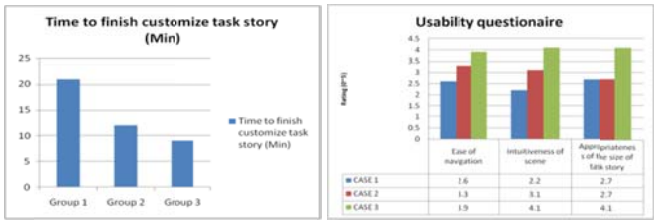
Operation	Descriptions
Add	Drag and drop scenes onto U-Board among the recommended scenes
Delete	Press delete key on the scene
Move	Drag and drop scenes onto the any cell in U-Board
Retrieve	Input the functionality of scenes to the retrieval window
Save	Press save button on task authoring tool
Undo	Press undo button to rollback to the original task story

TABLE II. EVALUATION CRITERIA AND METHODS

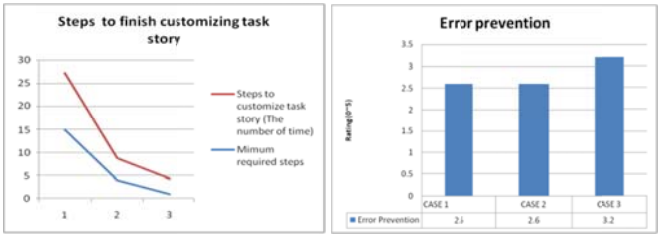
Requirements	Evaluation Methods
Usability	Easy to entry (measure time to finish customizing a task story according to a given scenario)
	Intuitiveness of scenes and task story Q) Difficulties to understand scenes and task story
	Appropriateness of the size of user interface considering mobile device Q) Appropriateness of the size of each scene
	Appropriateness of the volume of information Q) Do you think we provide too much information to bother the use of tool?
Efficiency	User efforts (measure the number of steps to customize a given task story)
	Error prevention Q) Do you think the guidelines are helpful to solve difficulties in customizing task story
Reusability	Ease of Add/Delete/Move/Save Scenes Q) Do you think Add/Delete/Move/Save functions are working properly
	Accessibility to existing scenes and task stories Q) Do you think it is easy to access to existing scenes and task stories

To the first group of users, we provided a tool that shows only a monolithic image for each scene and does not support the abstraction and mapping of scenes into task activities. To the second group of users, we provided a tool that visualizes scenes based on our scene visualization model, but does not support the task-oriented abstraction of scenes. Our task-oriented storyboard authoring tool was given to the third group of users.

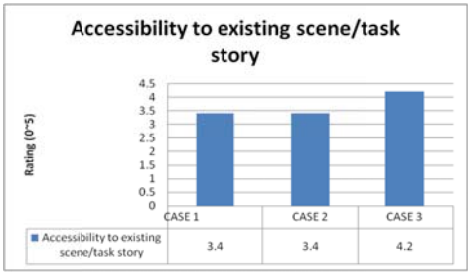
We evaluated the three groups to check whether they meet the usability, efficiency and reusability requirements that we explained in Section II. To measure the usability, we counted the time taken by the user groups to finish customization of a task story, and asked a series of questions to check various usability factors. Efficiency is measured by counting the number of steps to finish the customization of the task story, and by checking the effectiveness of error prevention facilities supported by the tools. Reusability is measured by asking a couple of questions that are about the easiness of managing scenes, and the effectiveness of finding and accessing existing scenes and task stories. Table II summarizes these evaluation criteria and methods.



(a) Usability



(b) Efficiency



(c) Reusability

Figure 5. Evaluation Results

Fig. 5 shows the evaluation results. As shown in Fig. 5(a), our approach (Case 3) lowered the barrier to entry to the application authoring job by reducing the time to learn how to customize and manage task stories. The answers to the usability questionnaires also show that our approach is much more intuitive and satisfactory than other two approaches.

In terms of efficiency, our approach contributed to reduce the number of steps to customize task stories and to prevent errors during the application authoring process. As shown in Fig. 5(b), by using our approach, the users had to perform 9 steps in average where the first approach required average 22 steps.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9

Finding appropriate storyboards to reuse is crucial to improve the reusability of service compositions. The survey result shown in Fig. 5(c) proves that our approach helped the users to find and access useful storyboards to reuse.

VII. CONCLUSION

In this paper, we proposed the storyboard-based end-user mobile application authoring method. The main goal of our approach is to allow end-users, who do not have sophisticated technical knowledge about developing mobile applications, to easily create and customize those applications. We identified three essential requirements (usability, efficiency, and reusability) of mobile application authoring for end-users to successfully represent their task goals and required contexts in an application description.

Our task-oriented storyboard approach provides an environment in which end-users can develop mobile applications without having technical knowledge. Users can visually browse through existing storyboard templates by controlling multi-dimensional aspects, and easily extend and customize them to generate mobile applications to achieve their computational goals. By using our tool, users can represent compound scenes and essential control structures that are effective to manage and dynamically instantiate storyboards according to the changes of environmental conditions and user requirements.

We are currently in progress on testing our application authoring tool by applying it to the public application domains in our campus and conducting a research to make the application authoring more evolvable by accepting and reflecting feedbacks from end-users. The accumulated feedbacks are analyzed in spatial, temporal, personal, and social perspectives. These can be automatically reflected in selecting or composing unit tasks and services for application authoring.

ACKNOWLEDGMENT

This work was supported in part by the IT R&D program of MKE/IITA [KI001877, Location/Societal Relation-Aware Social Media Service Technology]. This research was also supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.

REFERENCES

[1] Hansmann., Pervasive Computing: The Mobile World. Springer, 2003, ISBN 3540002189.

[2] Wang, Z. and Garlan, D., Task-Driven Computing. Technical Report, CMU - CS -00-154, 2000.

[3] W.T. Tsai and Yinong Chen., Introduction to Service-Oriented Computing, Arizona State University, http://www.public.asu.edu/~ychen10/activities/SOAWorkshop. <retrieved: July, 2011>

[4] Andy Ko., The State of the Art in End-User Software Engineering, http://www.sei.cmu.edu/interoperability/research/approaches/upload/Lewis-SEEUP2009-Workshop-20Review.pdf. <retrieved: July, 2011>

[5] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste., "Project Aura: Toward Distraction-Free Pervasive Computing", *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 22-31, Apr.-June 2002.

[6] Florian Daniel, Jin Yu, Boualem Benatallah, Fabio Casati, Maristella Matera, and Regis Saint-Paul., "Understanding UI Integration: A survey of problems, technologies, and opportunities," *IEEE Internet Computing*, vol. 11, no. 3, pp. 59-66, May/June 2007.

[7] The Protégé Ontology Editor and Knowledge-base Framework, http://protege.stanford.edu/. <retrieved: July, 2011>

[8] The Jena Semantic Web Framework, http://jena.sourceforge.net/. <retrieved: July, 2011>

[9] Yang Li and James A. Landay., "Activity-Based Prototyping of Ubicomp Applications for Long-Lived, Everyday Human Activities," Proc. *Twenty-sixth annual SIGCHI conference on Human factors in computing systems (2008)*, pp. 1303-1312.

[10] Agnes Ro, Lily Shu-Yi Xia, Hye-Woung Paik, and Chea Hyon Chon., Bill Organiser Portal: A Case Study on End-User Composition, Proc. *WISE 2008 Workshops*, Springer Berlin / Heidelberg, vol. 5176, pp. 152-161, 2008.

[11] Ivar Jacobson, Martin Griss, and Patrik Jonsson, Software reuse: architecture, process and organization for business success, ACM Press/Addison-Wesley Publishing Co., New York, NY, 1997.

[12] Nam-Yong Lee and Charles R. Litecky, "An Empirical Study of Software Reuse with Special Attention to Ada," *IEEE Transactions on Software Engineering*, vol. 23 no. 9, pp. 537-549, September 1997.

[13] W. M. Johnston, J. R. Paul Hanna, and R. J. Millar., "Advances in Dataflow Programming Languages, *ACM Computing Surveys (CSUR)*, vol. 36, no. 1, pp. 1-34, March 2004.

[14] Cycling 74 Max, http://www.cycling74.com/products/max.html. <retrieved: July, 2011>

[15] D. Draheim and G. Weber, Form-Oriented Analysis., A New Methodology to Model Form-Based Applications, Springer, October 2004, ISBN-10: 3540205934

[16] Zhiming Wang, Rui Wang, Cristina Aurrecoechea, Douglas Brewer, John A. Miller, and Jessica C. Kissinger., Semi-Automatic Composition of Web Services for the Bioinformatics Domain, http://cs.uga.edu/~jam/home/theses/z_wang_dissert/thesis/wsbiojournal/workflow-journal29.pdf. <retrieved: July, 2011>

[17] M.Haesen, J.Meskens, K.Luyten, and K. Conix., Supporting Multidisciplinary Teams and Early Design Stages Using Storyboards, Springer Berlin / Heidelberg, Human-Computer Interaction. New Trends, Volume 5610, pp. 616-623, 2009.

[18] Scaffidi, C. Shaw, C., and Myers, B., An Approach for Categorizing End-user Programmers to Guide Software Engineering Research. Proc. *First Workshop on End-user Software Engineering (WEUSE)*, pp. 1-5 at the 27th International Conference on Software Engineering (ICSE 2005), St. Louis, Missouri, USA, May 15-21, 2005.

[19] Margaret Burnett, Curtis Cook, and Gregg Rothermel. End-user software engineering. Commun. ACM 47, 9 (September 2004), pp. 53-58.

[20] In-Young Ko, Hyung-Min Koo, and Angel Jimenez-Molina. User-Centric Web Services for Ubiquitous Computing. J.D.Vel´asquez and L.C. Jain (Eds.): Advanced Techniques in Web Intelligence – 1, SCI 311, pp. 167–189, Springer-Verlag Berlin Heidelberg 2010.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-164-9