Implementing TOAST, a Tool for Agile Software Project Management in Cloud Computing Environments

Chung Yung*, Yu-Tang Lin
Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan.

* Corresponding author. Tel.: 886-3-8634017; email: yung@mail.ndhu.edu.tw
Manuscript submitted April 25, 2015; accepted August 28, 2015.
doi: 10.17706/jsw.10.11.1310-1318

Abstract: In this paper, we present a software project management of agile development in cloud computing environments, called as TOAST, which is constructed based on the MOAT model. MOAT is a two-layer model of software project management using agile methodology, which is especially designed for managing software projects in cloud computing environments. In MOAT, the primary operations of agile software development are categorized into two layers; namely, the project layer and the task layer. With MOAT, the progress status of an agile software development project is represented easily and clearly. TOAST implements all the MOAT operations using a client/server architecture, of which the TOAST client is deployed as an Android app and the TOAST server is designed to provide services in cloud computing environments. Since there is not many, if any, software management tools of agile development available, TOAST is considered as an explorative tool especially for the software industry that aims at cloud computing environments as the primary computing platform.

Key words: Agile software development, cloud computing environment, software development methodology, software project management.

1. Introduction

The agile software development methodology achieves agility by promoting self-organizing teams, customer collaboration, higher quality, less documentation, and reduced time to market [1], and there are a few frameworks proposed for applying agile methodology to software development projects in recent years [2]. On the other hand, cloud computing is not only a technological term that refers to data, processing, or experiences that live out there, somewhere in the cloud that we call as the Internet, but also a silent revolution in the way how companies operate with data and applications in the processes of inventing, developing, deploying, scaling, updating, maintaining and paying for resources that undergo the changes [3]. Recently, Yung, et al. propose a two-layer model of software project management using agile methodology, called MOAT, which is especially designed for managing software projects in cloud computing environments [4].

Traditional software development applies the waterfall model, in which each of the requirements, design, implementation, verification, and maintenance phases flows sequentially and cascades downward to the following phase. However, since it is typical that software projects do not have sufficient requirements specified early on, the requirements phase usually either takes much longer than expected or is rushed to an unfinished result, and hence the remainder of the project gets negative impact and may eventually fail

[5].

The agile methodology emphasizes rapid and flexible software development, which transforms the development process from being process-centric to being human-centric [6]. The major characteristics of agile development include short releases, flexibility, and minimal documentation. It is reported that about 31% of the industry practices agile methodology, including tailored versions [7]. Agile methodology has been found to be of great value to software development, improving quality and shortening time-to-market of software products [8].

Cloud computing technology provides more flexibility than conventional computing environments [9], [10]. Although the process of distributed software development evolves and changes by a lot in the past decade, there are still some user needs remaining unsatisfied. Various models have been proposed for software project management in cloud computing environments [11]. However, most of these proposed models are based on the traditional waterfall model of software development [12]. To satisfy the needs, it calls for new models of software project management using agile methodology that is particularly appropriate for the use in cloud computing environments.

MOAT is a two-layer model for software project management in cloud computing environments [4]. In the top (project) layer, MOAT lists the primary operations of agile software development and describes the progress in a software project with a project state diagram. In the bottom (task) layer, MOAT lists the primary operations of the task development in a sprint cycle and describes the progress of each task with a task state diagram. With MOAT, the current development status of the agile software project may be described easily and clearly, such that the software project can be well managed in cloud computing environments.

This paper describes our construction of a software project management tool called TOAST that implements all the MOAT operations with a client/server architecture, of which the TOAST client is deployed as an Android app and the TOAST server is designed to work in the cloud computing environments. Since there is not many, if any, software management tools of agile development available, TOAST is considered as an explorative tool especially for the software industry that aims at the cloud computing environments as the primary computing platform.

The remainder of this paper is organized as follows. We briefly discuss the background and preliminaries in the following section. Section 3 officially introduces the MOAT model. We briefly describe the primary project management operations that are modeled in MOAT. In section 4, we present our implementation TOAST, the software project management tool of agile software development in cloud computing environments. And, at last is a brief conclusion.

2. Background and Preliminaries

In this section, we discuss about the background and preliminary of our research. This section includes two parts: the agile software development methodology, and the web-based management model of software development projects in cloud computing environments.

2.1. Agile Software Development Methodology

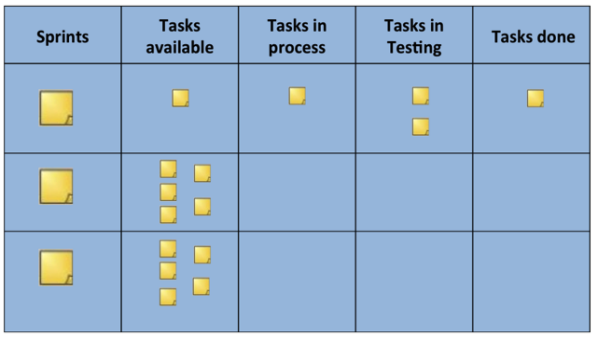
Agile software development methodology facilitates to produce high quality software in shorter period of time. The term “agile” refers to the software development team as active, swift, and responsive [13]. The common agile models of agile software development methodology include Scrum, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), and Test Driven Development (TDD) [14].

With traditional waterfall methodology, a software development project is divided into several phases, usually including the following:

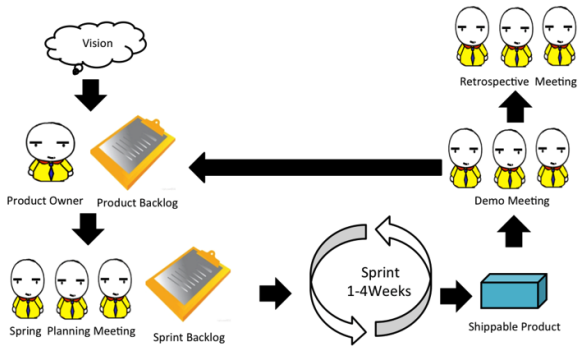
1) Launch,
2) Requirements,
3) Design,
4) Implementation,
5) Test, and
6) Rollout.

The agile software development methodology modeled by MOAT is similar to scrum methodology [3], with some minor modifications. The first phase is the project launch phase, and the next phase of the agile project is the sprint cycle, which includes the activities of the requirements, design, implementation, test, and rollout phases and combines into one big repeatable activity [15]. Usually, any design work or documentation becomes a user story on the backlog in order to allocate time to be completed, as shown in Fig. 1 (a). The project activities from launch phase to rollout phase are processed by the sprint cycles, as shown in Fig. 1 (b).

Note that similar to the other agile software development, the MOAT software development methodology provides more flexibility with late requirement changes allowed, and hence that software projects using agile methodology are developed more efficiently.



(a) A project backlog and tasks.



(b) Activities of an agile project.

Fig. 1. The MOAT software development methodology.

2.2. Software Project Management in Cloud Computing Environments

WebSD is a web-based management model of distributed software development project for cloud computing environments [11], [12], [16]. In WebSD, a software project of distributed development is described as a well-designed set of modules, which serve as the basic units of encapsulation.

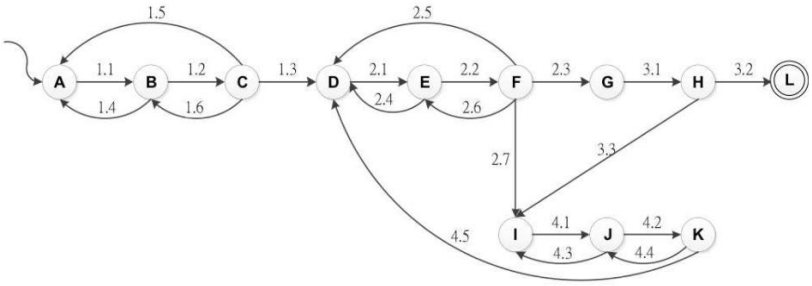
The primary operations in software development projects modeled in WebSD are listed in Fig. 2 (a). Note that WebSD includes the operations in the implementation phase, the testing phase, the debugging phase, and the integration phase. Furthermore, the life-cycle for developing a module in a software system is defined by the project state diagram shown in Fig. 2 (b). Note that with the project state diagram, the current status of a software development project may be described easily and clearly, and hence the software project can be well managed in cloud computing environments.

As an example, the state of a module is initially *A*. After it is assigned to a group for programming, the state goes to *B*. Once the assigned group accepts the job of programming, the state goes to *C*. When the group reports the finish of programming, the state goes to *D*. Then, it is assigned for testing and the state goes to *E*. Once the assigned group accepts the job of testing, the state goes to *F*. When the group reports the finish of testing and no bug is found, the state goes to *G*. And then, the project manager performs the integration tests, and the state goes to *H*. If it passes the integration tests, the state goes to *L* and the

development of the module is completed.

	Phase	Operation
P1	Programming	1.1 $Assign\_to_p$
		1.2 $Accept_p$
		1.3 $Finish_p$
		1.4 $Reject_p$
		1.5 $Withdraw_p$
		1.6 $Reassign_p$
P2	Unit Testing	2.1. $Assign\_to_u$
		2.2. $Accept_u$
		2.3. $Finish_u$
		2.4. $Reject_u$
		2.5. $Withdraw_u$
		2.6. $Reassign_u$
		2.7. $Report\_bugs_u$
P3	Integration Testing	3.1. $System\_test$
		3.2. $Finish_s$
		3.3. $Report\_bugs_s$
P4	Debugging	4.1. $Assign\_to_d$
		4.2. $Accept_d$
		4.3. $Reject_d$
		4.4. $Reassign_d$
		4.5. $Finish_d$

(a) Primary operations in distributed software development.



(b) Project state diagram.

Fig. 2. The WebSD model of software project management.

Note that the WebSD model has the advantage that the status of a software project using distributed development may be described easily and clearly, and hence that we may manage the progress of the software project in an official manner.

3. The MOAT Model

For managing software development projects using agile methodology in cloud computing environments, Yung, et al. propose a two-layer model called MOAT, meaning Model Of Agile Technology [4]. They categorize the primary operations of agile software development into two layers; namely, the project layer and the task layer. MOAT models the progress in a project with a project state transition diagram. And for each task, MOAT models the progress with a task state transition diagram.

For clarity, we describe the MOAT model with two sections. In the first section, we describe the operations modeled in the project layer. The second section includes the operations modeled in the task layer.

3.1. Project Layer

In the project layer, MOAT models the following twelve primary operations in agile software development:

p_1: Sort requirements and split into sprints	p_7: Start next sprint
p_2: Distribute into tasks	p_8: Integrate sprints
p_3: Develop tasks	p_9: Project test
p_4: Integrate tasks	p_{10}: Report project bugs
p_5: Sprint test	p_{11}: Fix project bugs
p_6: Report sprint bugs	p_{12}: Complete project

In MOAT, the progress in an agile software project is modeled using a project state transition diagram, as shown in Fig. 3. The initial state of a software project is A when the project is just created. As the project leaders sort the requirements and split into sprints, the project goes to state B. When the jobs in the sprint are distributed into tasks, the project goes to state C. When the project starts to develop tasks, the state goes to D. In state D, MOAT uses a second-layer state transition diagram to model the progress of each task, which is introduced afterward. As soon as all the tasks finish development and get integrated, the project goes to state E. Then the project starts sprint tests, and the state goes to F. If there are bugs reported, the state goes back to C, and the team starts working on the tasks of debugging. If the sprint passes the test, the project either goes back to state A for another sprint cycle, or goes to state G for starting integration. When the project starts integration tests, the state goes to H. If there are bugs reported, the state goes to I, and when the team finishes debugging, the state goes back to G for more integration tests. Otherwise, if the project passes the integration tests, the project goes to state J, and the project is completed.

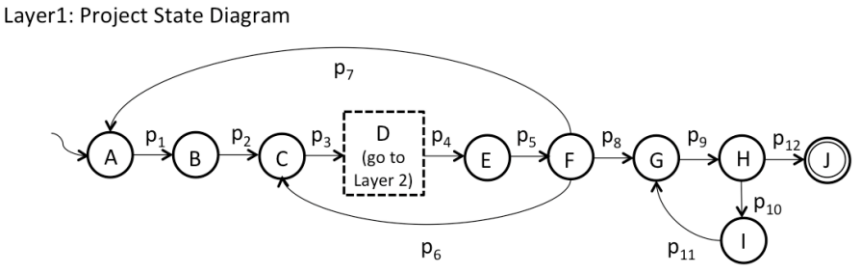


Fig. 3. MOAT project state diagram.

3.2. Task Layer

In the task layer, MOAT models the following three primary operations in agile software development:

q_1: Pick up task
q_2: Withdraw task
q_3: Finish task

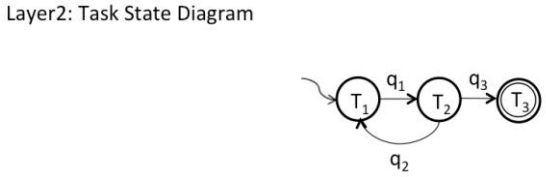


Fig. 4. MOAT task state diagram.

In MOAT, the progress in each task is modeled using a task state transition diagram, as shown in Fig. 4. Initially, a task is at state T_1. When an engineer picks up the task, the state goes to T_2. The engineer may

withdraw the task for someone else to pick up, if he encounters with some difficulty, and the state goes back to T_1. When the task is finished, the state goes to T_3.

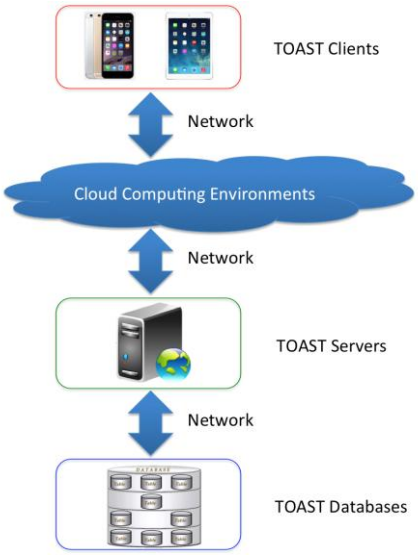


Fig. 5. The overall architecture of TOAST.

Day	Operation	State
d_0	(Project starts)	A
d_1	$p_1 \rightarrow \{ A, B \}$	B
d_2	$p_2(A) \rightarrow \{\alpha_1, \alpha_2, \alpha_3\}$	C
d_3	$p_3(\alpha_1, \alpha_2, \alpha_3)$	D
d_6	$p_4(\alpha_1, \alpha_2, \alpha_3)$	E
	$p_5(A)$	F
d_7	$p_7(A)$	A
	$p_1 \rightarrow \{ B \}$	B
d_8	$p_2(B) \rightarrow \{\beta_1, \beta_2\}$	C
d_9	$p_3(\beta_1, \beta_2)$	D
d_{11}	$p_4(\beta_1, \beta_2)$	E
	$p_5(B)$	F
d_{12}	$p_6(B) \rightarrow \{\beta_3, \beta_4\}$	C
	$p_3(\beta_3, \beta_4)$	D
d_{13}	$p_4(\beta_1, \beta_2, \beta_3, \beta_4)$	E
d_{14}	$p_5(B)$	F
	$p_8(A, B)$	G
	$p_9(A, B)$	H
d_{15}	p_{12}	J

(a) Progress in project layer.

Day	α_1		α_2		α_3	
d_3	-	T_1	-	T_1	-	T_1
	q_1	T_2	q_1	T_2	q_1	T_2
d_4	q_3	T_3	q_2	T_1		
d_5			q_1	T_2	q_3	T_3
d_6			q_3	T_3		

(b) Progress of tasks in sprint A.

Day	β_1		β_2	
d_9	-	T_1	-	T_1
	q_1	T_2	q_1	T_2
d_{10}			q_3	T_3
d_{11}	q_3	T_3		

Day	β_3		β_4	
d_{12}	-	T_1	-	T_1
	q_1	T_2	q_1	T_2
d_{13}	q_3	T_3	q_3	T_3

(c) Progress of tasks in sprint B.

Fig. 6. Application of TOAST to manage the CSCI project.

4. TOAST

In this section, we describe the software tool that we construct based on the MOAT model. We call it TOAST, meaning a Tool Of Agile Software Technology. With our experience in building project management systems [4, 11, 12, 16], TOAST is especially designed for the use in cloud computing environments with an advantage that the progress status of the software project in development is described clearly and easily.

TOAST implements all the 12 project operations and 3 task operations modeled by MOAT. The overall structure of the TOAST system is shown in Fig. 5. There are 3 subsystems in TOAST: the TOAST client

subsystem, the TOAST server subsystem, and the TOAST database subsystem.

1) The TOAST client subsystem has two kinds of interfaces implemented. The first is a web-page version viewed with web browsers; the second is a packaged stand-alone Android app.
2) The TOAST server subsystem includes a bunch of programs coded in Java and Java Server Pages (JSP).
3) The TOAST database subsystem consists of a few tables in a SQL database that are in charge of manipulating user data, project data, and operation data.

For validating, we apply MOAT to a practical project CSCI, which is a web-based cooperative sale system of car insurance developed by a credit cooperative bank in Hualien, which our third author works with. The CSCI project was finished in three weeks in November 2013, and the CSCI software system is developed using agile methodology with two sprint cycles A and B. There are three tasks α1, α2, and α3 in sprint A. For sprint B, there are initially two tasks β1 and β2, and two more tasks β3 and β4 are generated later for debugging. The way that MOAT models the progress in CSCI project is shown in Fig. 6.

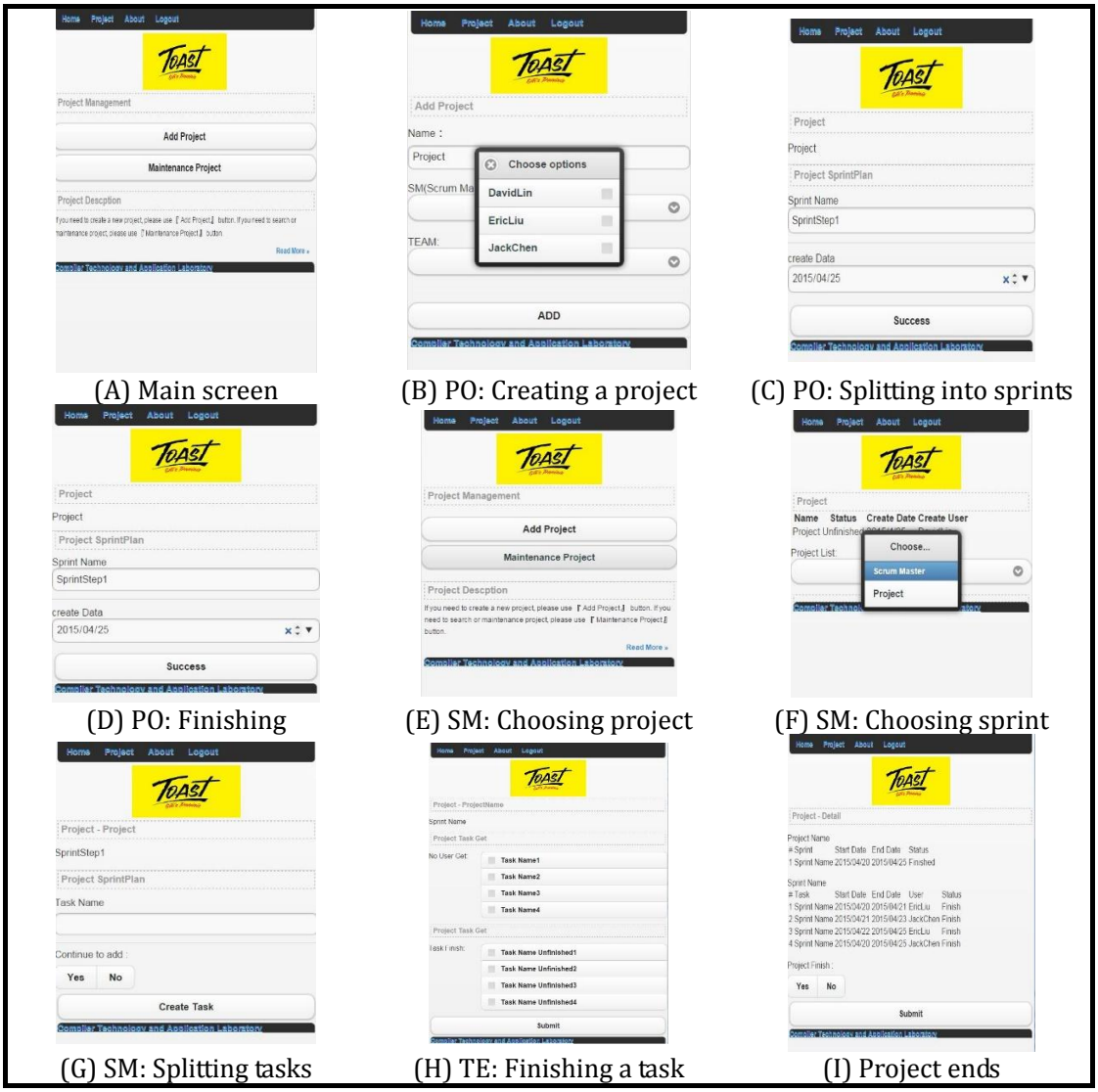


Fig. 7. Snapshots of applying TOAST to CSCI project.

Here, we include a few snapshots from using the TOAST system in the CSCI projec, as shown in Fig. 7. Fig. 7(A) is the main screen after logging into TOAST. Fig. 7(B) is the snapshot of creating a project by project owner. Fig. 7(C) shows the process of splitting requirements into sprints. Fig. 7(D) is a snapshot that

project owner finishes the process of splitting sprints. Fig. 7(E) is a snapshot when scrum master logs in and chooses project to work on. Fig. 7(F) is a snapshot of choosing a sprint. Fig. 7(G) shows the process of splitting stories into tasks by scrum master Fig. 7(H) is a snapshot when a team engineer chooses a task and report finish of the task. Finally, the project ends with a snapshot of Fig. 7(I).

5. Conclusion

A software project management tool of agile development methodology, called TOAST, is presented in this paper. TOAST implements all the agile development operations in MOAT, and is designed especially for managing agile software projects in cloud computing environments. The TOAST system is implemented using client/server architecture. We demonstrate that managing the progress in an agile software development project is simple and easy. Our future direction is to cooperate with software industry in applying the TOAST system to manage industry-level agile software development projects in cloud computing environments.

Acknowledgment

This work is partially supported by the Ministry of Science and Technology, Taiwan, under Grant No. MOST 103-2221-E-259-014-MY3.

The authors appreciate the discussion with Mr. Ming-Chen Li and his contribution in the early part of this project while he worked with us.

References

[1] Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010). Agile software development: impact on productivity and quality. *Proceedings of the IEEE International Conference on Management of Innovation and Technology* (pp. 287-291).

[2] Cristal, M., Wildt, D., & Prikladnicki, R. (2008). Usage of scrum practices within a global company. In *Proceedings of the IEEE International Conference on Global Software Engineering* (pp. 222-226).

[3] Silva, F. Q. B., Costa, C., Franca, A. C. C., & Prikladinicki, R. (2010). Challenges and solutions in distributed software development project management: A systematic literature review. *Proceedings of the 5th IEEE International Conference on Global Software Engineering* (pp. 87–96).

[4] Yung, C., Li, M. C., & Lin, Y. T. (2014). A new model of software project management using agile methodology in cloud computing environments. *Proceedings of the International Scientific Conference on Management and Information Science* (pp. 233-243).

[5] Lehman, T. J., & Sharma, A. (2011). Software development as a service: Agile experiences. *Proceedings of the Annual SRII Global Conference* (pp. 749-75).

[6] Gregorio, D. D. (2012). How the business analyst supports and encourages collaboration on agile projects. *Proceedings of IEEE International Systems Conference* (pp. 1-4).

[7] Hayata, T., & Han, J. (2011). A hybrid model for IT project with Scrum. In *Proceedings of the IEEE International Conference on Service Operations, Logistics, and Informatics* (pp. 285-290).

[8] Hadar, I., & Sherman, S. (2011). Agile vs. plan-driven perceptions of software architecture. In *Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 50-55).

[9] Sun, H., Wang, X., Zhou, C., Huang, Z., & Liu, X. (2010). Early experience of building a cloud platform for service oriented software development. *Proceedings of the IEEE International Conference on Cluster Computing* (pp. 1–4).

[10] Zhou, Y. C., Liu, X. P., Wang, X. N., Xue, L., Liang, X. X., & Liang, S. (2010). Business process centric

platform-as-a-service model and technologies for cloud enabled industry solutions. *Proceedings of the Third IEEE International Conference on Cloud Computing* (pp. 534–537).

[11] Yung, C., Chen, S. Z., & Hsieh, J.-T. (2014). Automatic schedule control for distributed software development in cloud computing environments. *Journal of Industrial and Intelligent Information, 2(1),* 39-44.

[12] Yung, C., Chen, S.-Z., Wu, S.-C., Hsieh, J.-T., & Peng, K.-J. (2012). A web-based model of distributed software development management for cloud computing environments. *GSTF Journal of Computing, 2(2),* 1-7.

[13] Melo, C., Cruzes, D. S., Kon, F., & Conradi, R. (2011). Agile team perceptions of productivity factors. *Proceedings of the Agile Conference* (pp. 57-66).

[14] Shalaby, M., & El-Kassas, S. (2011). Applying Scrum framework in the IT service support domain. *Proceedings of the IEEE Asia-Pacific Services Computing Conference* (pp. 9-15).

[15] Sutherland, J., Schoonheim, G., Kumar, N., Pandey, V., & Vishal, S. (2009). Fully distributed Scrum: Linear scalability of production between San Francisco and India. *Proceedings of the Agile Conference* (pp. 277-282).

[16] Yung, C., & Chen, S.-Z. (2013). DDMan: A management system for distributed software development in cloud computing environments. *International Journal of e-Education, e-Business, e-Management and e-Learning, 3(6),* 446-450.

Chung Yung was born in received PhD degree in computer science from New York University (USA) in 1999 and BSc degree in computer science and information engineering from National Chiao Tung University (Taiwan) in 1988.

He has been with the Department of Computer Science and Information Engineering of National Dong Hwa University (Taiwan) since 2000. He was a part-time senior consultant and project manager within the intelligent digital content industry between 2003 and 2007. He is currently leading Compiler Technology and Application Laboratory in National Dong Hwa University. His research interests include semantic methods of program analysis, optimizations for cloud software systems, compiler supported software engineering, and programming languages.

Dr. Yung has been a committee member of CTHPC (Taiwan) since 1999. He received the best presentation award in the 2015 International Conference on Information Technology. He has been awarded a couple of times for his excellence in research by National Dong Hwa University.

Yu-Tang Lin was born in Hualien, Taiwan. He received BSc degree in management information science from Kuan-Shan University (Taiwan) in 2007.

He currently works as a senior software engineer in Hualien Second Credit Cooperative (Taiwan). He is also a part-time student in the Department of Computer Science and Information Engineering, National Dong Hwa University (Taiwan). His research interests include software development of Java programming language, system analysis and design, and software project management.