# DDMan: A Management System for Distributed Software Development in Cloud Computing Environments

Chung Yung and Shao-Zong Chen

***Abstract*—In this paper, we present a management system for distributed software development in cloud computing environments, called *DDMan*. Cloud computing environments provide more flexibility than conventional computing environments. In particular, the platform as a service (PaaS) in cloud computing environments provides benefits for users in a variety of aspects, such as application design, development testing, software deployment, team collaboration, web service integration, database integration, and scalability. 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. For managing distributed software development in cloud computing systems, we present the DDMan system, which extends the WebSD model of distributed software development management. In DDMan, the interactions among the roles in the software development are precisely defined. In addition to the conventional web-based interface, the DDMan system also supports the interface for the Amazon cloud environment, which is an approach to a fully automatic service of distributed software development management in cloud computing environments.**

***Index Terms*—Distributed software development, software project management, cloud computing environments.**

## I. Introduction

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 [1]-[3]. It calls for the need of a new model of distributed software development management, which is specially designed for the use in cloud computing environments. This motivates us to develop a new management system that implements a new model of distributed software development for cloud computing environments.

More precisely, the management system presented in this paper is designed to meet the following goals:

1) To simplify the hierarchical architecture of conventional distributed software development,
2) To make the management system more flexible and suitable for outsourcing the development of parts in the software system to fellow companies, and
3) To allow various roles involved in the distributed software development, including project managers, software programmers, and software testers, to cooperate in the management system.

In the past decade, the globally distributed way of software development is applied to more and more software projects [4]. Altogether, based on the report by da Silva *et al.* [5], there are 30 challenges, 31 management best practices, 10 models, and 24 tools were collected from 54 works published between 1998 and 2009. The key finding is that the strong evidence about the effect of using the best practices, models, and tools in distributed software development projects is still scarce in the literature [5]. The intent of distributed software development is to fully use the resources, including computing devices and human resources, to achieve flexibility, quality and cost down. However, several challenges arise in globally distributed software development at the same time, such as formalization in communication, the management of formal changes, planning for system integration, project monitoring across distributed teams, standard development tools, and integrated management tools [6].

There are various architectures for distributed software development management proposed in the literature, majorly from 1997 to 2010. According to [6], we may classified them into five categories:

1) Communication based approach [4], [5],
2) Virtual roles based approach [7], [8],
3) Virtual roles and Communication based approach [9],
4) Knowledge management and compliance based approach [10]-[12],
5) Virtual machines based approach [13], [14].

We design and build a management system for distributed software development, called *DDMan*, which implements the WebSD model proposed by Yung et al. [6]. In addition, we make some improvement when developing the DDMan system. First, in DDMan, we formalize the communication architecture in the distributed software development, and thus, DDMan supports the process of all the roles of project members in the distributed software development; including the project owners and/or managers, the software programmers, the software testers, and the software debuggers. Second, compared with traditional management systems for software development, DDMan improves the usage of resources, especially when DDMan is used in combination with other services provided at the cloud computing platforms. And, DDMan also provides a solution to the common problems in conventional distributed software development, such as the integration and timely process of the programs developed in different distributed regions. Usually,

Manuscript received April 17, 2013; revised July 28, 2013. This work is supported in part by the National Science Council under grand number NSC 100-2218-E-259-002- MY3.

Chung Yung and Shao-Zong Chen are with Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan 97401, R.O.C. (e-mail: {yung, m9721505}@mail.ndhu.edu.tw).

such features require redundant investment in the software development and management systems in order for software transformation across difference platforms. DDMan eliminates such redundant investment by providing a common management platform for distributed software development.

This paper is organized as follows. The next section briefly describes the WebSD model. In Section III, we present the design of the DDMan system. An application of DDMan to a practical industry case is included in Section IV. And, at last is a brief conclusion.

## II. The WebSD Model

In this section, we briefly introduce a model of distributed software development management in cloud computing environments, called WebSD [6], based on which the DDMan system is implemented.

In WebSD, a project of distributed software development is described as a well-designed set of modules, which serve as the basic units of encapsulation. It is reported that separating the code into modules has the benefit of encouraging developers to design well-defined interfaces and thereby to shape coupling and cohesion of the code [3].

The management operations of distributed software development modeled in WebSD are listed in Fig. 1. We briefly describe each operation as follows.

1.1. Assign_to$_p$: A project manager assigns the job of programming a module to a group.
1.2. Accept$_p$: A group accepts the job of programming a module.
1.3. Finish$_p$: A group finishes the job of programming a module.
1.4. Reject$_p$: A group rejects the job of programming a module.
1.5. Withdraw$_p$: A group withdraws the acceptance of programming a module.
1.6. Reassign$_p$: A project manager reassigns the job of programming a module to a group.
2.1. Assign_to$_u$: A project manager assigns the job of testing a module to a group.
2.2. Accept$_u$: A group accepts the job of testing a module.
2.3. Finish$_u$: A group finishes the job of testing a module.
2.4. Reject$_u$: A group rejects the job of testing a module.
2.5. Withdraw$_u$: A group can withdraw the acceptance of testing a module.
2.6. Reassign$_u$: A project manager reassigns the job of testing a module to a group.
2.7. Report_bugs$_u$: A group reports bugs after testing a module.
3.1. System_test: A project manager schedules the job of integration testing involving a module.
3.2. Finish$_s$: A project manager finishes the job of integration testing a module.
3.3. Report_bugs$_s$: A project manager reports bugs after integration testing a module.
4.1. Assign_to$_d$: A project manager assigns the job of debugging a module to a group.
4.2. Accept$_d$: A group accepts the job of debugging a module.

| Phase | | Operation |
|---|---|---|
| P1 | Programming | 1.1Assign_top<br>1.2Acceptp<br>1.3Finishp<br>1.4Rejectp<br>1.5Withdrawp<br>1.6Reassignp |
| P2 | Unit Testing | 2.1. Assign_tou<br>2.2. Acceptu<br>2.3. Finishu<br>2.4. Rejectu<br>2.5. Withdrawu<br>2.6. Reassignu<br>2.7. Report_bugsu |
| P3 | Integration Testing | 3.1. System_test<br>3.2. Finishs<br>3.3. Report_bugss |
| P4 | Debugging | 4.1. Assign_tod<br>4.2. Acceptd<br>4.3. Rejectd<br>4.4. Reassignd<br>4.5. Finishd |

Fig. 1. Basic management operations in distributed software development.

4.3. Reject$_d$: A group rejects the job of debugging a module.
4.4. Reassign$_d$: A project manager reassigns the job of debugging a module to a group.
4.5. Finish$_d$: A group finishes the job of debugging a module.

In the WebSD model, the life-cycle of developing a module is presented by the state transition diagram shown in Fig. 2. 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 schedules 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 complete.

One of the advantages in the WebSD model is that the status of each module in a project of distributed software development is well-defined, and hence the project can be in good management [6].

**Definition** (Status of a software project using distributed development P)

Given a software project consisting of a module set M of $k$ modules, the status of the project P is defined as

$$P=\{p_i \mid 1 \leq i \leq k\} \tag{1}$$

where each $p_i$ is a pair $\langle m_i, s_i \rangle$, $m_i \in M = \{m_1, ..., m_k\}$, and

$$s_i \in S = \{A, B, C, D, E, F, G, H, I, J, K, L\}. \tag{2}$$

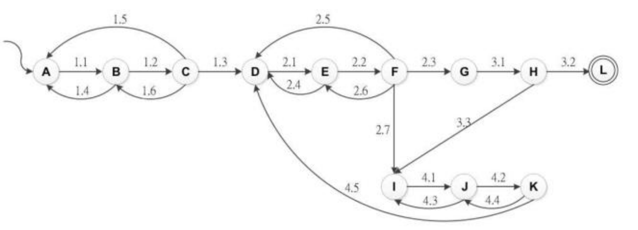


Fig. 2. The state transition diagram of the WebSD model.

With the definition of P, we may keep record of the progress in distributed software development with a formal manner.

## III. The DDMan System

In this section, we present a project management system of distributed software development, called DDMan, which implements the WebSD model described in the previous section. In addition to conventional web-based interface, DDMan also includes a distribution for Amazon cloud environment (http://aws.amazon.com), such that all members involved in the project may enjoy the benefit of cloud computing environments; that is, being able to report and proceed their jobs whenever they have access to the internet.

In this section, we introduce the DDMan system as follows. We start with a brief description on the system structure of DDMan. And then, we show the interface design of the DDMan system.

### A. System Structure of DDMan

The overall structure of the DDMan system is shown in Fig. 3. The major components in the DDMan system structure are described as follows.

1) **User Interface**: In our design, a few styles of user interfaces for DDMan are supported for various platforms. At the time of writing this paper, the interfaces for World Wide Web, and Amazon Web Service are already available.
2) **Management Agent**: DDMan includes a fully automatic management agent that implements the management operations defined in the WebSD model.
3) **Manager Subsystem**: The manager subsystem manipulates the operations needed by the project managers.
4) **Programmer Subsystem**: The programmer subsystem manipulates the operations needed by the programmers.
5) **Tester Subsystem**: The tester subsystem manipulates the operations needed by the software testers.

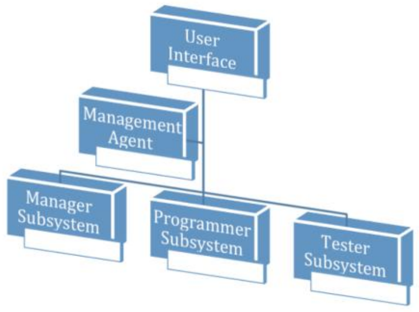


Fig. 3. System Structure Diagram of DDMan.

### B. User-Interface Design of DDMan

In the DDMan system, the user interface is designed as an individual module such that DDMan may be easily ported to various platforms. At the time of writing this paper, the DDMan user interface supports two platforms; namely, the World Wide Web and the Amazon cloud environment.

Some snapshots of the DDMan system are shown in Fig. 4. In Fig. 4 (a), group $g_1$ views the status of each module in the project. In Fig. 4 (b), group $g_1$ assigns the job of programming a module to a programming group. In Fig. 4 (c), group $g_2$ reports the finish of programming a module. In Fig. 4 (d), group $g_4$ accepts the job of testing a module.

## IV. A Case Study of ConsMan

We present the application of the DDMan system to a practical project called *ConsMan*, which is a distributed software development project for constructing an information management system for an energy and power company in Taiwan that our second author worked with. The ConsMan project is executed between 2006 and 2007. Note that the case study is applied after the project is closed, based on the documentation kept during execution, with simplification and adaption for the ease of presentation. Also note that in the following description, we only include the record of top-level activities.

TABLE I: A Practical Case Study of Applying DDMan to the ConsMan Project

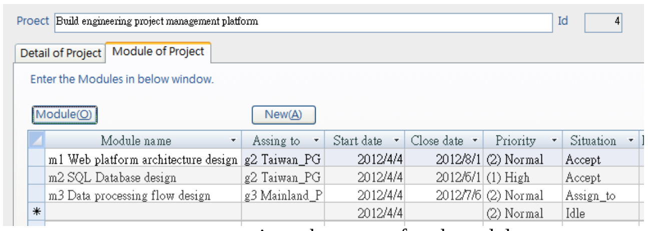
| Module | $m_1$ | $m_2$ | $m_3$ |
|---|---|---|---|
| Day 0 | A | A | A |
| | $g_1$:Assign_to$_p$(1,2) | $g_1$:Assign_to$_p$(2,2) | $g_1$:Assign_to$_p$(3,3) |
| Day 1 | B | B | B |
| | $g_2$:Accept$_p$(1) | $g_2$:Reject$_p$(2) | $g_3$:Accept$_p$(1,2) |
| Day 2 | C | A | C |
| | | $g_1$:Reassign$_p$(2,3); $g_3$:Accept$_p$(2) | $g_3$:Finish$_p$(3) |
| Day 3 | C | C | D |
| | $g_2$:Finish$_p$(1) | $g_3$:Withdraw$_p$(2); $g_1$:Reassign$_p$(2,2) | |
| Day 4 | D | B | D |
| | $g_1$:Assign_to$_u$(1,4) | $g_2$:Accept$_p$(2); $g_2$:Finish$_p$(2) | $g_1$:Assign_to$_u$(3,5) |
| Day 5 | E | D | E |
| | $g_4$:Accept$_u$(1) | $g_1$:Assign_to$_u$(2,5) | |
| Day 6 | F | E | E |
| | $g_4$:Report_bugs$_u$(1) | $g_5$:Accept$_u$(2) | $g_5$:Accept$_u$(3) |
| Day 7 | I | F | F |
| | $g_1$:Assign_to$_d$(1,3) | $g_5$:Finish$_u$(2) | |
| Day 8 | J | G | F |
| | $g_3$:Accept$_d$(1) | | $g_5$:Finish$_u$(3) |
| Day 9 | K | G | G |
| | $g_3$:Finish$_d$(1) | | |
| Day 10 | D | G | G |
| | $g_1$:Assign_to$_u$(1,4) | | |
| Day 11 | E | G | G |
| | $g_4$:Accept$_u$(1) | | |
| Day 12 | F | G | G |
| | $g_4$:Finish$_u$(1) | | |
| Day 13 | G | G | G |
| | $g_1$:System_test(1) | $g_1$:System_test(2); $g_1$:Report_bugs$_s$(2) | $g_1$:System_test(3) |
| Day 14 | H | I | H |
| | | $g_1$:Assign_to$_d$(2,2); $g_2$:Accept$_d$(2) | |
| Day 15 | H | K | H |
| | | $g_2$:Finish$_d$(2); $g_1$:Assign_to$_u$(2,5) | |
| Day 16 | H | E | H |
| | | $g_5$:Accept$_u$(2); $g_5$:Finish$_u$(2) | |
| Day 17 | H | G | H |
| | $g_1$:Finish$_s$(1) | $g_1$:System_test(2); $g_1$:Finish$_s$(2) | $g_1$:Finish$_s$(3) |
| Day 18 | L | L | L |

The ConsMan project, with three top-level modules, is developed by a team consisting of five groups, including an in-house group ($g_1$: PCM), and two outsourcing groups of agents and consultants ($g_2$: ACT, and $g_3$: ACC), and the other two off-site testing groups ($g_4$: PCN, and $g_5$: PCS).

We briefly describe the progress in the project as follows.

1) $g_1$ does Assign_to$_p$($m_1$, $g_2$), Assign to$_p$($m_2$, $g_2$), and Assign_to$_p$($m_3$, $g_3$).
2) $g_2$ does Accept$_p$($m_1$) and Reject$_p$($m_2$), and $g_3$ does Accept$_p$($m_3$).
3) $g_1$ does Reassign$_p$($m_2$, $g_3$), and $g_3$ does Accept$_p$($m_2$) and Finish$_p$($m_2$).
4) $g_2$ does Finish$_p$($m_1$), $g_3$ does Withdraw$_p$($m_2$), and $g_1$ does Reassign$_p$($m_2$, $g_2$).
5) $g_1$ does Assign_to$_u$($m_1$, $g_4$) and Assign_to$_u$($m_3$, $g_5$), $g_2$ does Accept$_p$($m_2$) and Finish$_p$($m_2$).
6) $g_4$ does Accept$_u$($m_1$), and $g_1$ does Assign_to$_u$($m_2$, $g_5$).
7) $g_4$ does Report_bugs$_u$($m_1$), and $g_5$ does Accept$_u$($m_2$) and Accept$_u$($m_3$).
8) $g_1$ does Assign_to$_d$($m_1$, $g_3$), and $g_5$ does Finish$_u$($m_2$).
9) $g_3$ does Accept$_d$ ($m_1$), and $g_5$ does Finish$_u$($m_3$).
10) $g_3$ does Finish$_d$($m_1$).
11) $g_1$ does Assign_to$_u$($m_1$, $g_4$).
12) $g_4$ does Accept$_u$($m_1$).
13) $g_4$ does Finish$_u$($m_1$).
14) $g_1$ does System_test($m_1$), System_test($m_2$), System_test($m_3$), and Report_bugs$_s$($m_2$).
15) $g_1$ does Assign_to$_d$($m_2$, $g_2$), and $g_2$ does Accept$_d$($m_2$).
16) $g_2$ does Finish$_d$($m_2$), and $g_1$ does Assign_to$_u$($m_2$, $g_5$).
17) $g_5$ does Accept$_u$($m_2$) and Finish$_u$($m_2$).
18) $g_1$ does System_test($m_2$), Finish$_s$($m_1$), Finish$_s$($m_2$), and Finish$_s$($m_3$).

The record of applying the DDMan system to manage the progress in developing ConsMan is shown in Table I.

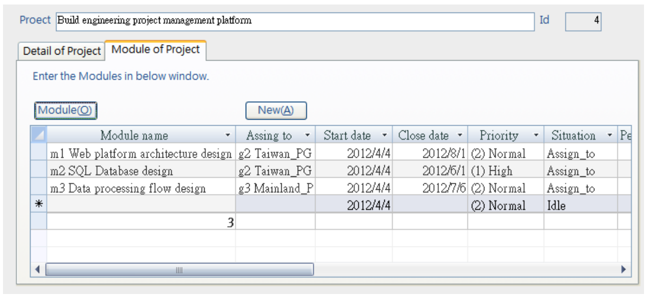


Proect Build engineering project management platform Id 4

Enter the Modules in below window.

| | Module name | Assing to | Start date | Close date | Priority | Situation |
|---|---|---|---|---|---|---|
| | m1 Web platform architecture design | g2 Taiwan_PG | 2012/4/4 | 2012/8/1 | (2) Normal | Accept |
| | m2 SQL Database design | g2 Taiwan_PG | 2012/4/4 | 2012/6/1 | (1) High | Accept |
| | m3 Data processing flow design | g3 Mainland_P | 2012/4/4 | 2012/7/6 | (2) Normal | Assign_to |
| * | | | 2012/4/4 | | (2) Normal | Idle |

(a) Group $g_1$ views the status of each module

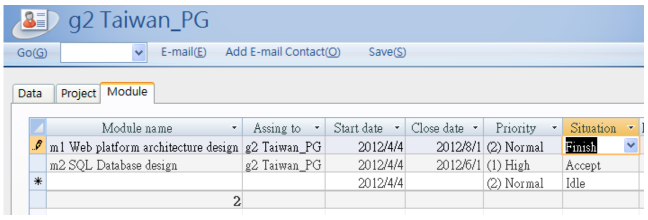


Proect Build engineering project management platform Id 4

Enter the Modules in below window.

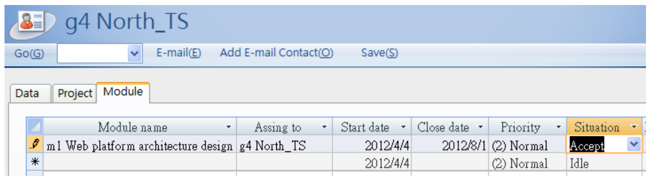
| | Module name | Assing to | Start date | Close date | Priority | Situation | Pe |
|---|---|---|---|---|---|---|---|
| | m1 Web platform architecture design | g2 Taiwan_PG | 2012/4/4 | 2012/8/1 | (2) Normal | Assign_to | |
| | m2 SQL Database design | g2 Taiwan_PG | 2012/4/4 | 2012/6/1 | (1) High | Assign_to | |
| | m3 Data processing flow design | g3 Mainland_P | 2012/4/4 | 2012/7/6 | (2) Normal | Assign_to | |
| * | | | 2012/4/4 | | (2) Normal | Idle | |
| | 3 | | | | | | |

(b) Group $g_1$ assigns the job of programming a module



| | Module name | Assing to | Start date | Close date | Priority | Situation |
|---|---|---|---|---|---|---|
| | m1 Web platform architecture design | g2 Taiwan_PG | 2012/4/4 | 2012/8/1 | (2) Normal | Finish |
| | m2 SQL Database design | g2 Taiwan_PG | 2012/4/4 | 2012/6/1 | (1) High | Accept |
| * | | | 2012/4/4 | | (2) Normal | Idle |
| | 2 | | | | | |

(c) Group $g_2$ reports finish of programming a module



| | Module name | Assing to | Start date | Close date | Priority | Situation |
|---|---|---|---|---|---|---|
| | m1 Web platform architecture design | g4 North_TS | 2012/4/4 | 2012/8/1 | (2) Normal | Accept |
| * | | | 2012/4/4 | | (2) Normal | Idle |

(d) Group $g_4$ accepts the job of texting a module

Fig. 4. Snapshots of the DDMan system.

## V. Conclusion

We present a management system for distributed software development in cloud computing environments, called DDMan, which implements the WebSD model [6]. We describe the overall structure of the DDMan system, and we show the user-interface design of the DDMan system. We also include a practical case study of applying DDMan to manage the ConsMan project.

Based on our experience, it is clear that there is need of extending the DDMan system to include more precise information of distributed software development so that the project may be better managed. We note the following directions of future work, most of which are currently under development.

1) To explore and integrate more management operations to provide precise views of distributed software development management for project managers.
2) To extend DDMan and facilitate the management of multiple projects concurrently developed by a common global virtual team.
3) To construct a cloud service of distributed software development management so that DDMan can be easily accessed and integrated with the other services in the cloud computing environments.

## References

[1] V. Matveev, "Platform as a service – new opportunities for software development companies," Master's thesis, Department of Information Technology, Lappeenranta University of Technology, Punkkerikatu 2 A 6, 53850 Lappeenranta, Finland, May 2010.

[2] Y. C. Zhou, X. P. Liu, X. N. Wang, L. Xue, X. X. Liang, and S. Liang, "Business process centric platform-as-a-service model and technologies for cloud enabled industry solutions," in *Proc. the Third IEEE International Conference on Cloud Computing (CLOUD 2010), IEEE Computer Society*, July 2010, pp. 534–537.

[3] J. S. Rellermeyer, M. Duller, and G. Alonso, "Engineering the cloud from software modules," in *Proc. the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE Computer Society*, May 2009, pp. 32–37.

[4] A. Piri, "Challenges of globally distributed software development analysis of problems related to social processes and group relations," in *Proc. the IEEE International Conference on Global Software Engineering (ICGSE) 2008, IEEE Computer Society*, August 2008, pp. 264–268.

[5] F. Q. B. da Silva, C. Costa, A. C. C. Franca, and R. Prikladinicki, "Challenges and solutions in distributed software development project management: a systematic literature review," in *Proc. the 5th IEEE International Conference on Global Software Engineering (ICGSE) 2010, IEEE Computer Society*, August 2010, pp. 87–96.

[6] C. Yung, S.-Z. Chen, S.-C. Wu, J.-T. Hsieh, and K.-J. Peng, "A web-based model of distributed software development management for cloud computing environments," *GSTF Journal of Computing*, vol. 2, no. 2, pp. 1–7, June 2012.

[7] H. Zhu, M. Zhou, and P. Seguin, "Supporting software development with roles," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 6, pp. 1110–1123, November 2006.

[8] B. Xu, X. Yang, and A. Ma, "Role based cross-project collaboration in multiple distributed software design projects," in *Proc. the 12th*

*International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008)*, April 2008, pp. 177–182.

[9] K. A. Johnston and K. Rosin, "Global virtual teams: How to manage them," in *Proc. the 2011 International Conference on Computer and Management (CAMAN)*, May 2011, pp. 1–4.

[10] I. S. Wiese and E. H. M. Huzita, "IMART: An interoperability model for artifacts of distributed software development environments," in *Proc. the 2006 International Conference on Global Software Engineering (ICGSE)*, October 2006, pp. 255–256.

[11] M. Ali-Babar and I. Gorton, "A tool for managing software architecture knowledge," in *Proc. the Second Workshop on Sharing and Reusing Architecture Knowledge-Architecture, Rationale, and Design Intent (SHARK/ADI 2007)*, May 2007, pp. 11–17.

[12] M. Ali-Babar, "A framework for supporting the software architecture evaluation process in global software development," in *Proc. the Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009), IEEE Computer Society*, July 2009, pp. 93–102.

[13] G. Lawton, "Developing software online with platform-as-a-service technology," *IEEE Computer*, vol. 41, no. 6, pp. 13–15, June 2008.

[14] H. Sun, X. Wang, C. Zhou, Z. Huang, and X. Liu, "Early experience of building a cloud platform for service oriented software development," in *Proc. the 2010 IEEE International Conference on Cluster Computing, Workshops and Posters. IEEE Computer Society*, September 2010, pp. 1–4.

**Chung Yung** 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.

**Shao-Zong Chen** is currently with Taiwan Power Company as a senior software engineer since 2002. He is working part-time for his master degree in Computer Science and Information Engineering at Compiler and Technology Application Laboratory of National Dong Hwa University, Taiwan. He received BSc degree in Computer Science and Information Engineering from Tamkang University, Taiwan in 1992. His research interests include software project management, program analysis and distributed software development in cloud computing environments.