Advances in Engineering Research (AER), volume 124
2nd International Symposium on Advances in Electrical, Electronics and Computer Engineering
(ISAEECE 2017)

An Approach to Evolving Internetware Application

Elenga Danieland Lionel Vitrand[1, a], Xin Zuo[1, b,] Hualong Yu[1, c] and Shang Zheng[1, d]

[1]School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China

[a]lionelvldelenga@yahoo.com, [b]632343650@qq.com, [c]153094658@qq.com, [d] zs.hot@163.com

Keywords: Internetware, Architecture Engineering, Software Evolution

Abstract. Internetware is a new software paradigm evolved by the internet; it brings many challenges for the traditional software development techniques, especially for architecture engineering in the new paradigm. Self-adaptability is one of the most important capabilities and design goals of Internetware. Although the existing researches provide some support to the self-adaptability of the systems, there is still not enough. Some of them are focusing on high-level analysis and simulation, not taking into practice, and others discuss the evolution of existing architecture in Internetware environment. In order to save developing time, cost, and keep the original features, this paper proposed an approach to supporting existing architecture evolution in Internetware. There are three phases in the approach: 1) prediction 2) transformation 3) comparison, which cannot only make the systems to become more self-adaptive in Internetware environment, but also can be easily used to guide system evolution.

1. Introduction

Internet environment is becoming more open, dynamic and complex, which brings a great challenge on the existing software applications. Internetware [1] is proposed in the current internet environment. It is an abstract of new software patterns under internet and it is a new paradigm for internet application. The new software paradigm is constituted by software entities, supported by software components and other technologies. Due to the open environment, Internetware needs to keep evolving to fulfil the dynamic requirements. Architecture evolution can be thought as a central feature of all software systems. According to Internetware's concept and construction, it is important to select suitable architecture for the entities. At present, the researches mainly focused on Internetware modelling, Internetware developing, Internetware middleware technology and evolution based on services provide by entity. For example, Yang [2] proposed a kind of architecture-centred method, based on entity and structured cooperation on demand and specified the characteristic analysis by defining open computing model. From the view of Internetware developing technology and middleware development, Mei [3] suggested the architecture-based component composition method based on software architecture. Meanwhile, some typical researches based on application view in Internetware include: a series of Internetware projects based on agent [4, 5], the trust measurement for Internetware [6], a formal analysis of Internetware evolution [7] and Internetware evolution model and its implementation [8]. Although some of these studies discussed Internetware evolution, the researches only focused on analysis method of Internetware evolution and the evolution model based on entity. The implementation on Internetware architecture evolution is still far from completion. It is lack of systematic approach to consider setting architecture evolution in Internetware.

This paper proposes an approach to setting architecture evolution in Internetware, which contains three steps. First, the analysis on the existing architecture of entities in Internetware is needed to complete and the prediction analysis for target architecture is implemented by a method based on AHP evaluation [9]. Second, after the architecture selection analysis is completed, the ACME [10] is utilised to describe the updated architecture, meanwhile, WSL [11] in FermaT [12] workbench is used to transform the ACME to a better description for architecture and developer can add new properties through this process. Finally, a comparison approach based complexity of time and space is used to compare the old and new architecture.

2. Related Work

Researchers have done some work on designing Internetware model and architecture evolution. However, there are not the detailed approaches that have considered setting architecture evolution in the paradigm of Internetware. In [13], an agent-based autonomous component model for Internetware was proposed and it provided self-management support for constructing Internetware, however, they do not consider the quality properties and architecture styles. As introduced in [14] [15], the authors proposed the approaches to designing self-adaptive architecture and environment-driven model for Internetware, which advanced the development of the new paradigm, but they do not also discuss the evolution issues after the design process. Architecture evolution is a key feature of most software systems, especially for IT-based companies, the companies need to deploy their systems in the internet environment, and evolving the existing architecture to the suitable one can save the cost and design time. In the research area of architecture evolution, there are four types: the evolution of the code structures, the tool support for management, the formal approaches to architecture transformation and the tradeoff analysis for architecture evolution [16]. In this paper, the proposed approach is described from the architecture transformation in Internetware.

3. Proposed Approach

Internetware has been thought as a new paradigm, many entities consist the paradigm. So the architecture among those entities needs to keep changing to fulfil the dynamic requirements. This paper focuses on how to analyse, transform and evolve the architecture in Internetware. In the Fig.1, the framework of the proposed approach is shown.

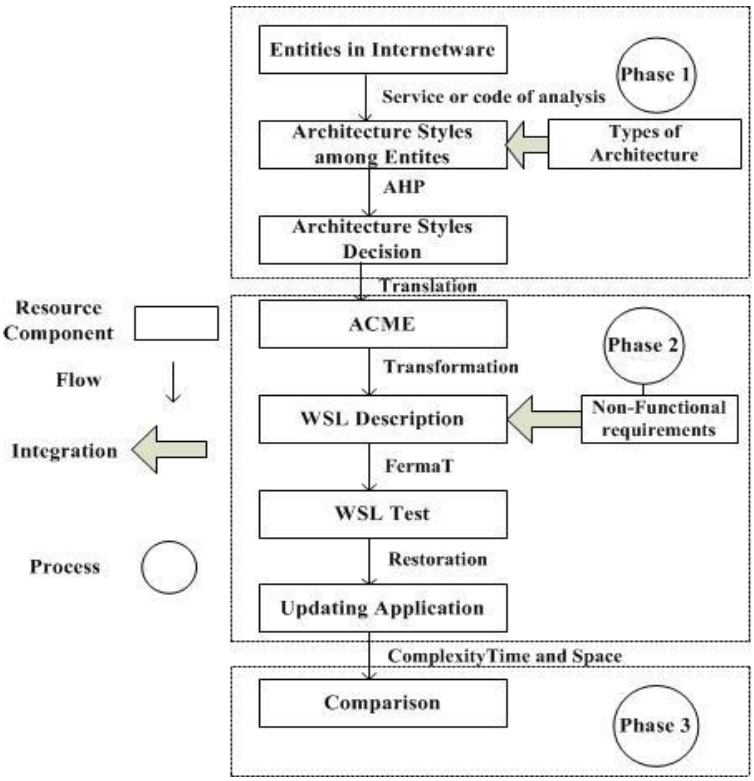

Fig.1.Framework of Internetware Application Evolution

There are three phases in the proposed approach. In phase 1, the entities in Internetware should be analyzed first, and then the architectures among the entities can be required according to the service type or code structure. Second, the evolution of architectures style for the entities needs to be completed. When the existing architecture style is known, the developers need to change it to fulfil the dynamic requirements, and they can take the existing architecture types as the criteria or update the existing architecture properties because many companies have considered architecture enhancement for their systems. About the introduction of architecture style, the details can be found.

By the current architecture styles, there is a prediction for the updated architecture, Analytic Hierarchy Process (AHP) are utilized to predict the possibility of the selection. The AHP result can describe the effects different styles on required aspects or the change of a style property. Fig.2 shows the AHP structure for evaluating the selection of architecture styles. This figure just chooses a style as the representation of each architecture type.

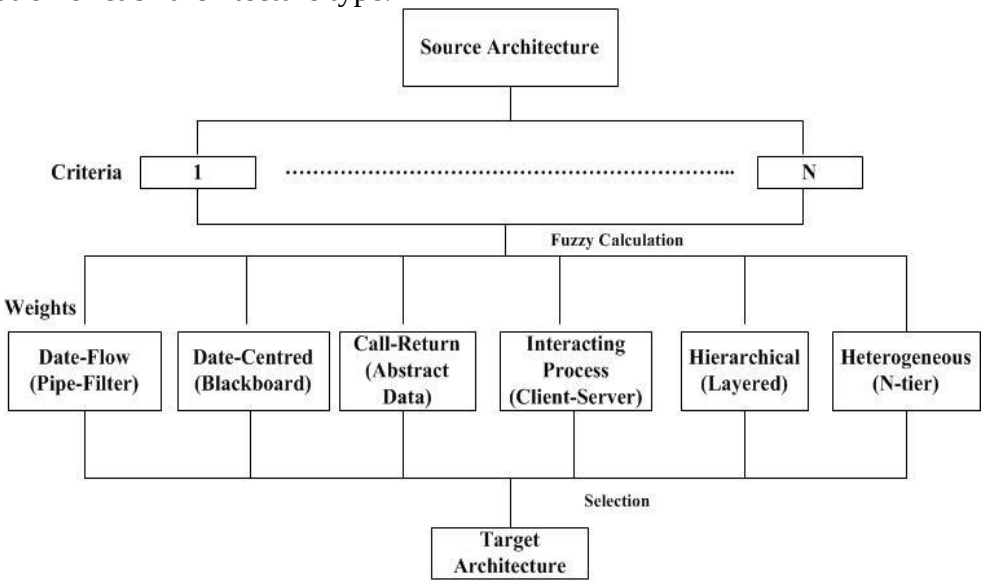

Fig.2.AHP Structure of Evaluating Architecture Selection

According to the complexity prediction, if the complexity does not exceed the limitation, the change can be selected and the suitable change of architecture style can be selected for the evolution. In Phase 2, ACME can be used to translate the decided style content. It is convenient to understand the style by ACME. After the description, WSL is chosen to transform ACME and test the integration of functions. It is not only easy to describe by WSL and restore code, but also the FermaT can provide many transformation rules for the integration and specification. Fig.3 shows an array output of the WSL description and FermaT workbench. It shows the WSL description does not need many syntax and there are many rules in FermaT to simplify the description.

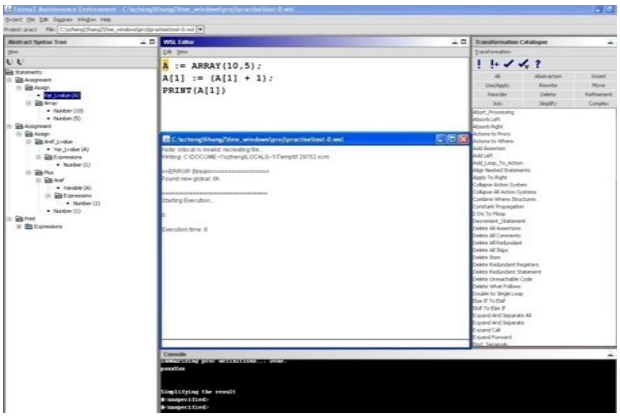

Fig.3.WSL Description

After phase 1 and phase 2, the work process has been finished. In order to compare the result with original architecture, phase 3 discusses the time complexity and space complexity between the new architecture and the old one to prove that the new architecture is better to deal with messages or services. By the evolution direction and the graph transformation in the tool support, the source architecture is transformed to the target architecture. With the source code and evolved architecture information, Eclipse and FermaT can implement the code evolution. In order to optimise the generated java code, the WSL in FermaT can be used to specify the program. Fig.4 shows the transformation state, there are the transformation operations in the right side of FermaT.

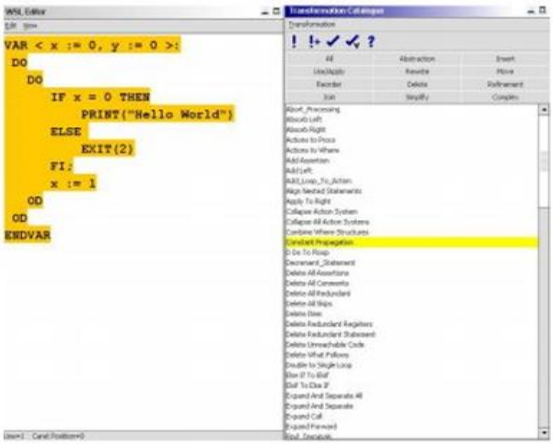

Fig.4.WSL Transformations in FermaT

Combining the proposed evolution process of architecture for Internetware, this paper provides the runtime environment of supporting evolution of system based Internetware in Fig.5. The supporting environment is described as Internetware infrastructure, supporting runtime architecture evolution by AGG & EMFT, completing runtime code generation and specification by Eclipse & FermaT and the interfaces between web and users. The graph transformation can guide the architecture evolution via static analysis and dynamic analysis. The code evolution can be executed by WSL description of generated code in Eclipse and FermaT after the architecture evolution. All the layers in the environment have constructed a mode of causal connection to implement the dynamic evolution of system based Internetware.

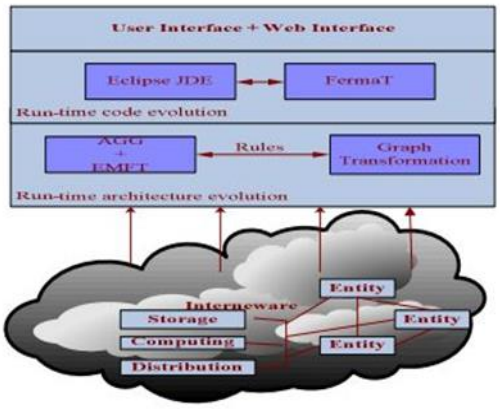

Fig.5. Runtime Environment of Supporting Internetware Application Evolution

4. Case Study

This paper illustrates the proposed approach by the case study Java Pet Store (JPS), which contains four main parts, pet store (PS) for interaction with users, processing centre (PC) for message communication, administration (AM) and supplier (SP). There are three steps to deal with the system: prediction, transformation and comparison.

4.1 Prediction. As proposed before, the fuzzy AHP model of modified LLSM (logarithmic least squares method) is utilised to estimate the architecture styles by the relevant criteria in this paper. There are three styles for the analysis, the original layered style, the blackboard style, the integrated style of layer and blackboard. The criteria are integrity, efficiency that come from ISO/2011. The fuzzy operational axioms and modified LLSM details can be required in [22], this paper does not introduce them again. Here, fuzzy rating scales are defined as:

Scale = {H, M, L, I, L^{-1}, M^{-1}, H^{-1}}, their triangular fuzzy numbers are defined as follows in this paper.

L = (1, 1.5, 2),

M = (2, 2.5, 3),

H = (3, 3.5, 4),

I = (1, 1, 1),

L^{-1}= (0.5, 0.67, 1),

M^{-1}= (0.67, 0.4, 0.5),

H^{-1} = (0.25, 0.29, 0.33).

Then the local fuzzy weights (LFW) under two criteria can be calculated in Table 1 and Table 2.

Table 1.LFW of Three Styles under Efficiency

	Integrated Style	**Layered Style**	**Blackboard Style**	**LFW**
Integrated Style	**(1, 1, 1)**	**(3, 3.5, 4)**	**(3, 3.5, 4)**	**(0.592, 0.593, 0.594)**
Layered Style	**(0.25, 0.29, 0.33)**	**(1, 1, 1)**	**(2, 2.5, 3)**	**(0.262, 0.281, 0.286)**
Blackboard Style	**(0.25, 0.29, 0.33)**	**(0.33, 0.4, .0.5)**	**(1, 1, 1)**	**(0.134, 0.125, 0.121)**

Table 2.LFW OF Three Styles under Integrity

	Integrated Style	**Layered Style**	**Blackboard Style**	**LFW**
Integrated Style	**(1, 1, 1)**	**(3, 3.5, 4)**	**(2, 2.5, 3)**	**(0.550, 0.556, 0.558)**
Layered Style	**(0.25, 0.29, 0.33)**	**(1, 1, 1)**	**(0.33, 0.4, 0.5)**	**(0.145, 0.134, 0.128)**
Blackboard Style	**(0.33, 0.4, 0.5)**	**(2, 2.5, 3)**	**(1, 1, 1)**	**(0.305, 0.310, 0.314)**

The estimation results can be easily required in the two tables. The LFW of integrated style is higher than other two styles. After the comparison under two criteria, a weight analysis result - w (1, 0, 0) is concluded, “1” represents the item can be considered and “0” shows not suitable under the criteria. Finally, the predicted architecture style can be proved more suitable and effective than existing architecture style under the required criteria.

4.2 Transformation. In order to make the new architecture design of JPS, WSL is utilised to transform the Acme description to a low level programming language that can be convenient to become high level language for application implementation, like C, Java and so on. During the process of the transformation, the developers also can add the extra functions or conditions in the WSL description. In order to simplify description, FermaT provides many rules to ensure the efficiency, security, maintainability and other qualities. Here just introduces several rules of them for efficiency and integrity.

- Combine_Wheres: will combine two nested WHERE structures into one structure which will contain the definitions from each of the original WHERE structures. The selected WHERE structure will be merged into an enclosing one if there is one or, failing that, into an enclosed WHERE structure.
- Delete_All_Redundant: Delete All Redundant searches for redundant statements and deletes all the ones it finds. A statement is `Redundant' if it calls nothing external and the variables it modifies will all be assigned again before their values are accessed.
- Reverse_Order: This transformation will reverse the order of most two-component items.

- Replace_Accs_With_Value: This transformation will apply Replace with Value to all variables with the names a0, a1, a2 and a3 in the selected item.

In order to complete the WSL transformation, Fig.6 shows the original architecture description in ACME. With the Acme description of JPS system, the WSL transformation for new system architecture can be ready to conduct. During the transformation of the process, the extra quality properties and functions can also be considered adding into the WSL code, which ensure the transformation more suitable for the dynamic requirements of Internetware. Meanwhile, if the additional functions are put into the code; the transformation rules can be used to simplify the description. Fig.7 shows the simplification of WSL description, and the ACME of new JPS system is in Fig.8.

```
System JPS : TieredFam = new TieredFam extended with {
    Component PS = {Port p: remoteUseT}
    Component SP = {Port p: remoteProvideT, provideT; Port p0: remoteUseT}
    Component PS2 = {Port p: remoteUseT}
    Component PS1= {Port p: remoteUseT}
    Component Backboard = {
        Port p: admin;
        Port p0: remoteUseT;
        Port p1: remoteUseT;
        Port p2: remoteProvideT, localUseT;
        Representation Board_Rep = {
            System Board_Rep : TieredFam = new TieredFam extended with {
                Component PC = {Port p: admin, localUseT}
                Component Admin = {
                    Port p: admin;
                    Port p0: admin}
                Component PC1 = {Port p : localUseT}
                Connector ConnAdmin = {Roles {caller,callee}}
                Connector ConnPC = {Roles {caller,callee}}
                Attachment PC.p to ConnAdmin.callee;
                Attachment Admin.p to ConnAdmin.caller;
                Attachment Admin.p0 to ConnPC.caller;
                Attachment PC1.p to ConnPC.callee;
            }}}
    Connector Send = {Roles {caller,callee}}
    Connector Response = {Roles {caller,callee}}
    Connector SendProvide = {Roles {caller,callee}}
    Connector ProviderResponseUser = {Roles {caller,callee}}
    Attachment PS.p to Send.caller;
    Attachment Backboard.p to Send.callee;
    Attachment PS1.p to Response.callee;
    Attachment Backboard.p1 to Response.caller;
    Attachment Backboard.p2 to SendProvide.caller;
    Attachment SP.p to SendProvide.callee;
    Attachment SP.p0 to ProviderResponseUser.caller;
    Attachment PS2.p to ProviderResponseUser.callee;
}
```

Fig.6.ACME Description of Original Architecture

```
BEGIN
 VAR < p := "", p0 := "", p1 := "", p2 := "" :
    exit_flag_1 := 0;
    ACTIONS PS:
    PS ==
        p := "remoteUser";
        CALL Blackboard
    END
    Blackboard ==
        p := "admin";
        p0 := "remoteUser";
        p1 := "remoteUser";
        p2 := "remoteProvider, localUser";
        CALL PC
    END
    PC ==
        p := "admin, localUser";
        CALL A■
    END
    A■ ==
        p := "admin";
        IF p0 = "remoteUser" THEN
            CALL PS
        FI;
        p := "remoteProvider, provider";
        CALL SP
    END
    SP ==
        p := "remoteProvider, provider";
        p0 := "remoteUser";
        CALL PS
    END
    ENDACTIONS
 ENDVAR
WHERE
PROC PS( VAR ) ==
 p := "p, p0, p1, p2"
END
END
```

Fig.7.WSL Transformation

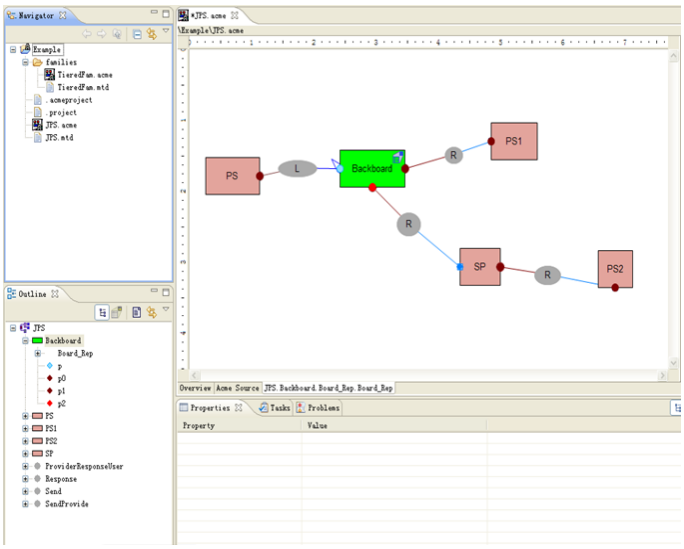

Fig.8. New Architecture in ACME

4.3 Comparison

4.3.1 The Comparison between Evolved Architecture and original One. If a large of operation commands need to implement, the integrated style is better than the original one. From the aspect of time complexity, the same command will be executed less than the original one and the system can have better performance. From the aspect of space complexity, the commands will take less space. The operations between them can also be optimised by programming developers. In a word, the evolved architecture can be better than the original one on the performance.

4.3.2 The Comparison between the Proposed Approach and the Existing Methods. Internetware is a new software paradigm and it requires different innovations in the development. Recently, there are not many papers that have discussed the architecture evolution in Internetware. Some papers just considered designing the autonomous or self-adaptive model for software engineering in Internetware and did not discuss architecture evolution for the dynamic internet environment. Distinguishing from the existing method, this paper proposes a general approach to supporting architecture evolution in Internetware, which will help the existing software model of Internetware to become more self-adaptive, evolvable and effective. Meanwhile, the approach can easily have integrated existing architecture styles and functional requirements and non-functional requirements into the evolution process, which can make software model more suitable in Internetware environment. The approach is a systematic process that can be effective to support architecture evolution in Internetware and it can also help other engineering methods more self-adaptive.

5. Conclusions

Internetware appears naturally as a new software paradigm that can be suitable in the open, dynamic and complex internet environment. Its engineering requires evolutions or innovations on the traditional software development techniques. In this paper, a three-phase approach has been introduced to support architecture evolution of entities in Internetware. According to “prediction”, “transformation” and “comparison”, the proposed approach can support the engineering of Internetware become more self-adaptive. Meanwhile, it can also be useful to link architecture styles and quality requirements in the engineering process. An example of internet application evolution process was experimented by the proposed approach. It can be seen that the approach can be easily applied to supporting architecture evolution in Internetware paradigm. In the future, the research will conduct complex systems and develop the approach more unified to cope with different paradigms.

Acknowledgements

This work was financially supported by Research Foundation for Talented Scholars of Jiangsu University of Science and Technology (1132921504).

References

[1] H. Mei, G. Huang, L. Lan, and J.G.Li, "A Software Architecture Centric Self-Adaptation Approach for Internetware," *Science in China Series F: Information Sciences*, Vol. 51(2008), p. 722-742.

[2] F.Q.Yang, J.Lv, and H.Mei, "A Kind of Architecture-centered Method," *Science in China Series (Series E: Information Sciences)*, Vol. 38(2008), p.818-828.

[3] H.Mei, J.C.Chang, and F.Q.Yang, "Software Component Composition based on ADL and Middleware," *Sci China SerF-Inf Sci*, Vol. 44(2001), p.136-151.

[4] W.T.Lu, P.Yu, X.X.Ma, X.P.Tao, and J.Lv, "Componentised Software Service and Its Application in Artemis-ARC," *Application Research of Computers*, Vol. 24(2007), pp. 169-172, 2007.

[5] C.Cao, X.X.Ma, and X.P.Tao, "Artemis-COOR:A Platform for Agent Based Dynamic Software Coordination," *Computer Engineering & Science*, Vol. 32(2010), p.1-5.

[6] Y.Wang, J.Li, F.Xu, and L.Zhang, "A Trust Measurement and Evolution Model for Internetware," *Journal of Software*, Vol. 17(2006), p.682-690.

[7] Q.Wang and Y.B.Wang, "Formal Analysis Method of Internetware Evolution," International Conference on Computer Science & Electronics Engineering, p.357-360(2012).

[8] T.Wang and G.S.Yin, "An Internetware Evolution Model and Its Implementation Based on Service Entity," Fifth International Conference on Internet Computing for Science & Engineering, p.130-134(2010).

[9] L.Mikehailov and P.Tsvetinov, "Evaluation of Service Using Fuzzy Analytic Hierarchy Process," *Applied Soft Computing*, Vol. 5(2004), p.23-33.

[10] D.Garlan, R.Monroe, and D.Wile, "Acme: An Architecture Decription Interchange Language," Proc Cascon, p.169-183(1997).

[11] H.Yang and M.Ward, *Successful Evolution of Software Systems*. Artech House, Boston, USA & London, UK(2003).

[12] "FermaT, http://www.cse.dmu.ac.uk/~mward/fermat.html".

[13] M.G.Wang, J.J.Jie, T.X.Shi, and X.Fang, "An Agent-Based Autonomous Component Model for Internetware," International Conference on Web Information Systems & Mining, p. 348-352(2010).

[14] H.Mei, G.Huang, L.Lan, and J.G.Li, "Architecture Based Self-Adaptive Approach for Internetware," *SCIENCE CHINA Information Sciences (Science in China Series F*, Vol. 38(2008), p. 901-920.

[15] J.Lv, X.X.Ma, X.P.Tao, C.Cao, Y.Huang, and P.Yu, "On environment-driven software model for Internetware," *Science in China Series F: Information Sciences*, Vol. 51(2008), p. 683-721.

[16] D.Garlan and B.Schmerl, "*Æ*vol: A tool for defining and planning architecture evolution," Proceedings of the 31st International Conference on Software Engineering, p. 591-594(2009).

[17] K.K.F.Yuen and H.C.W.Lau, "Evaluating Software Quality of Vendors using Fuzzy Analytic Hierarchy Process," Lecture Notes in Engineering & Computer Science, p. 126-130(2008).