International Journal of Control and Automation
Vol. 9, No. 5 (2016), pp. 289-296
http://dx.doi.org/10.14257/ijca.2016.9.5.28

Research of Adaptive Software Design's Patterns

Jiangang Jin[1], Shibao Sun[2], Xiaoan Bao[3]

1. School of Software, North China University of Water Resources and Electric Power, Zhengzhou, Henan, China, 450011. 2. Network Information Center, Henan University of Science and Technology. Luoyang, Henan, China, 471023. 3. The College of Informatics and Electronics, Zhejiang Sci-Tech University, Hangzhou, Zhejiang, China, 310018
E-mail:hn_jjg@126.com

Abstract

The discussion is based on the background of adaptive software development needs according to the application requirements of the mobile terminal. The result, through many scholars' research and development about adaptive software, are analyzed and compared with both the advantages and disadvantages of adaptive software. A development trend of adaptive software design patterns is foreseen. A new feasibility design method is proposed, which is based on adaptive software development model of the combination of design patterns.

***Keywords**: adaptive software, design patterns, architecture, UML metal model, model assembly language*

1. Introduction

With the wider development of mobile terminals, the traditional disposition of software with manual and static technique which only provides passive service cannot meet the requirements of complex environment. More and more computer software is required to have the adaptive capacity to promptly respond to various environments. Adaptive software, because of its adaptive capabilities, has become a hot issue in the field of software system research.

Throughout the whole lifecycle of adaptive software, it stresses that, by detecting the demands and changes in the environment and then in accordance with these to readjust their behavior, the adaptive actions are taken to respond to the changes of the users' requirements and system errors so that it can gear up the software to develop constantly to adapt to the changes in the environment. Generally, adaptive software requires multiple functional modules to work together. According to the different environment and customers' demands, the developers of adaptive software system need to provide the software designs adapted to the demands of them. The current study of adaptive software is focused on architecture [1-4], formal description of the adaptive characteristics [5], adaptive algorithm [6] and so on. However, these studies all lack the support for systematic and engineering adaptive software development. Due to the inherent high complexity of the adaptive software and the great relevance of the demands, the development process is often much more complex and the errors are easy to occur, which makes it so difficult to ensure high availability and reliability of adaptive software as is desired in the development process.

Design patterns, commonly used in the modeling process of software system to improve the quality of software system design and reusability [7], are a common solution to describe a kind of software design problems. Design patterns have been widely used in the development process of computer software system [8-10]. In recent years, some experts have gradually begun trying to use design patterns to accurately describe the design of each part of the adaptive software system [5, 9], [11-13], grouping the existing and mature solutions into reusable design patterns, which can provide adaptive software design with the foundation of engineering development.

Due to the difference of application environment and business requirements, the design of adaptive software system must select a variety of design patterns to match the demands and put them together systematically in order to meet the business needs .It is indicated that at least six different design patterns are necessary in Ramirez and other studies[8]to design a simplified adaptive Internet servers, including three kinds of evolution modes, two kinds of monitoring modes, and one kind of decision-making mode. However, it is lack of accuracy and easy to generate different understandings in traditional design patterns which are often used natural language or semi-formal language UML. Using design patterns, especially multiple design patterns used in combination, remains heavily dependent on the knowledge of experts, which is lack of the support for systematic and engineering theory.

Model Driven Architecture (MDA) is a software development framework based on the model proposed by the International Organization for Standardization OMG. MDA aims to center software system from the code-based to model-based to improve the abstract level of the software development. Design patterns can be introduced into MDA to combine the advantages of both. On the one hand, design patterns can be used as a basic unit of modeling in software design to fully support the component-based model and reusability and increase reusable granularity. On the other hand, the combination process of the design patterns based on MDA can be highly accurate and automated. MDA can convert the combination of design patterns into the combination of schema model. This can effectively improve the automation of design patterns, reduce the randomness and uncertainty in the combinations of design patterns and improve the effectiveness and reliability of the development. Therefore, it is an effective way of improving the engineering and the reliability of adaptive software development by accurately describing dynamic design patterns and properly using a model-driven tool to realize the automatized combination of the model.

2. Evolution of Adaptive Mechanisms Based on Software Architecture

Domestic and foreign scholars have done a lot of work about adaptive software development, especially based on adaptive framework. Garlan [14] and others have designed a self-evolutionary software framework model Rainbow based on architecture. In the model an external running architecture was used and the transformation description language of adaptive software was proposed. Loyall [15] and others proposed a method on how to offer a service quality assurance framework of adaptive software in the distributed network environment. At the same time they raised a service quality measurement and control mechanism. K-Component framework proposed by Dowling [16] *et. al.* emphasized the separation of evolution logic from business logic. A figure transformation was used to indicate the architecture reconfiguration .The K-IDL was used to depict in the aspect of business logic. Capra [17] and others put forward the CARISMA system, introducing the idea of environmental awareness. When the operating environment changed, the system could make appropriate changes and adjustments. Li Chang-yun [18] and others proposed a dynamic evolution model in the system of space by retaining design architecture of the stipulation and decision-making, providing the support and basis for the evolution of software runtime .Yu Ping [19] *et. al.* put forward a kind of online evolution method for the adaptive software architecture. In this method, software architecture meta-model was designed and implemented to guide the runtime software evolution .Hong Mei team [20] from Peking University proposed a PKUAS system based on the reflection mechanism to achieve the overall control of the system starting from the architecture of the system and the global view of system structure to guide. Ding Bo [21] *et. al.* proposed a component model ACOE which could support the credible evolution of software in order to support fine-grained online adjustment of the software environment adaptability. Bao Xiaoan [4, 22] *et. al.* put forward a multi-objective evolutionary adaptive software framework Transformer to support the evolution of the software system in the complex and changing environment.

On the whole, a lot of related design patterns are usually integrated in these adaptive software frameworks, such as the monitoring mode, the reflective mode, the evolution mode, *etc.* so that specific application implementers can focus on specific details of the application itself. However, this software relies heavily on the implementation of the specific design of software framework and the support of language platform that is used. The reusability of software development and applicability are subject to certain restrictions. Aiming at this problem, recently some experts have proposed an adaptive software development model based on design patterns.

3. Adaptive Software Research Based on Design Pattern

Because design patterns can effectively improve the reusability of software solutions, they play a more and more important role in the adaptive software design and development.

Gomaa [12] and others put forward a model-based development approach which was introduced into the adaptive software development process, and four kinds of typical software dynamic reconfiguration models were presented, providing a new idea for adaptive software development. But the description way based on the text template model makes design patterns difficult to accurately reuse. Oluyomi [23] *et. al.* put forward a model description template to describe the interaction and evolution between different intelligent modules in order to use more standardized way to describe the self-evolutionary pattern of the discrete autonomous system. Weyns [5] and others put forward a Forms formal model using a formal description way to describe design patterns of various self-adaption systems, and at the same time studied the expansion of the formal model in the discrete system. Zhang [6] *et. al.* proposed a model-based adaptive software development model, describing the dynamic behavior of the cooperation in a number of software components, and combined with temporal logic to represent the dynamic evolution behavior constraints of the system. Ramirez [8] and others summarized and summed up 12 kinds of typical dynamic system design patterns, including perception patterns, behavior patterns and strategy patterns, and presented how to use design patterns to guide the adaptive software development process. He Chengwan [24] and others put forward a realizable way based on the roles of the design patterns modeling so as to better separate the business process and the model logic. Yi Guoding [25] *et. al.* proposed a method using the adaptive object model to develop the adaptive software and analyze five typical patterns in the object model architecture to create a dynamic model.

These studies of adaptive software based on design patterns have summed up a lot of design patterns involved in the adaptive software designs, laying a good foundation for the development of adaptive software engineering and rigorous. However, in these studies, the descriptions for design patterns are usually based on the combination of text and UML, relatively lacking more extended ways and the descriptions of the design pattern combinations, which make the combination of design patterns still heavily dependent on expert knowledge. So correctness and reliability of the combined software model are difficult to secure.

4. Research of Formal Descriptions of Design Patterns and Composition

Two major problems of design patterns automatically combinations are how to precisely model design patterns and how to automate the conversion process of the schema model.

In the aspect of design patterns accurately modeled, the concept of role [24, 26, 27] has been widely used in design pattern modeling. A lot of formal description technique [28-30] is using formal logic to describe the relationship between the model roles and interaction. Keller and Schauer [26] proposed a way to use design components in the design layer to make software portfolio. The study made a modular description on design patterns by using UML package diagrams. However, the study did not give the process through designing components to get structural combination. Riehle [31] proposed the combination of design patterns using role graph .By using the role graph; we could model the design patterns to achieve the distribution from the object to the role and using the object graph to create a role relationship matrix and through the analysis of it to complete the role graph of the composite design patterns. Sun Junmei and Miao Huaikou [32] proposed a role-based modeling and evolution of design patterns in the form of class, class attribute and the relationship between the class, *etc.* and treated them as a role. The study took the form of a modeling language Object-Z for modeling and used theorem proves to verify the consistency of the evolution model before and after. These studies usually use a dedicated model description language, which is difficult to combine with the existing UML tools.

Some studies choose to extend the UML meta-model [29, 33, 34] to describe the model so as to define new meta-model objects for different roles of design patterns, and use OCL to constrain these roles. Operations and attributes of the class can make a role assignment more expressive than the MUL collaboration method. However, these methods use the custom meta-model while at present most of the existing UML tools can't handle the meta-model defined by users, which makes these techniques difficult to obtain a wide range of applications. At the present time, the formal descriptions focus on the use of existing UML extension mechanism (UML Profile) to describe the design patterns.

The document [10] introduced a new method to describe the combination of design patterns, using UML extension--tag value to mark instantiated model information, such as model names, model methods and other attributes，*etc.* This method could clearly mark model instantiated information and have a good traceability, but it didn't separate the business logic and mode logic so that it could not be conducive to model instantiated. He Chengwan [24] proposed a design pattern modeling and implementation method based on role. Using UML extension mechanism, the role of model and the structure of the binding relationship between the business model and semantics were given, thus separating the business model and the pattern model to solve the problem of overlapping and traceability of patterns. However, this study for the unit with the model did not support the overall modeling mode and didn't involve the problem related to the evolution and combination based on design patterns.

In the automatic model transformation of design patterns, Alencar [35] *et. al.* was the first to use Prolog to obtain the process of structured evolution of design patterns, and defined it as Prolog rules. Wang [36] and others proposed a model transformation method for design patterns. The transformation framework included a series of atomic mapping and mode mapping, and could generate a model conversion code based on XSLT. Judson [37] and others put forward a design pattern based on the model transformation and reconstruction method, using the UML meta-model to characterize the source model and target model. The conversion model could be used to constrain the conversion defined in the model layer and can also check the consistency of the conversion. In order to automatically apply design patterns in the software system design model, Zhang Tian [33] and others treated design patterns as a full development unit to use in the framework of MDA. By way of extending MOF, the model of meta-model unit and the corresponding modeling support mechanism were defined. For the transformation from the model of the unit design patterns to EJB, a description based on QVT standard was used to support the model transformation from the unit design mode (PIM) to EJB-PSM, but currently the transformation rules do not support the evolution and combination of design patterns. Dong [38] and others put forward a description method based on UML language of graphical design patterns, which supported collaborative application description of multiple design patterns. Its follow-up [10] presented an evolution and description mechanism based on the design patterns of Predicate and on this basis supported the evolution of design patterns based on MOFQUERY/View/ Transformation (QVT) language [39] raised by OMG organization for standardization. The study was focused on providing design patterns with interior reconfiguration, which was lack of the support for multiple design patterns used in combination.

Bao and Gui [40] proposed a description language based on visual design patterns of meta-model extension to support the evolution mechanism of design patterns. Configurations of design patterns for users were converted to evolution models and used to achieve the model transformation of design patterns based on QVT, which could be compatible with most UML tools and was integrated into the Visual Paradigm UML(VP-UML) [41] modeling tool. But the work of the combination description way of design patterns should be strengthened and it has not been applied to more complex development process of adaptive software.

5. Prospects for Future Development of Adaptive Software

Comprehensive the above three aspects of the study can be seen, the adaptive software framework predefines a set of design patterns to support the adaptive software development, but there is heavily dependent on the support for the system framework and harder reconstruction according to the specific environment needs. Adaptive software research based on the existing design patterns is still confined to a summary of design patterns of the adaptive system, lacking an accurate description of design patterns and limiting the combined use of design patterns. Currently, theoretical studies of formal design patterns for instance configuration of design patterns and the ability of combination description are not enough. The research on self-combination of design patterns is still limited to some specific combination of design patterns. Because of these limitations, although there have been a lot of adaptive software-related design patterns, those are much more difficult to be used to construct adaptive software like software components.

The theoretical research on the combination of design patterns as well as its application in the adaptive software system is significant to strengthen the reuse and extend the theoretical research of adaptive software design, and promote the engineering standard of the adaptive software development. To get better studies in three aspects of the formal representation of design patterns, description language of model combination and combined model transformation engineer, here a design framework model is given, as is shown in Picture 1.As is seen in Picture 1,the following key problems need to be solved in designing adaptive software:

1) Design patterns of more accurate formalization description are needed. The premise of mode combination is to use object-oriented modeling language UML meta-model to abstract design, the role model of design patterns, and variant description and combination interface.

2) In the process of combination of design patterns, the key is to solve the problem of internal configuration mode. It is a key problem for combination mode to coordinate the relationship between mode combinations, solve the description of the role repeat mapping

and convert the description to evolution model for the identification of the model transformation engineer.

3) Design pattern evolution and the combination model of the part，which mainly solve the problem of describing various model combination provided by the users in order to facilitate UML to combine.

4) Define the transformation rules, which is mainly to solve the model transformation in the field of relevant or irrelevant model used to describe the transformation process. In the process of combination mode, the pattern model and the user input model element will get the corresponding change(combination, increase, change), which need to combine the definition of design patterns, the user input model and related knowledge in the field of adaptive software to design related rules of model transformation.

For these difficult problems in the study, designing a description schema language to support the formalization and modular description of design patterns; designing language combination of mode to provide configuration and combination description support of design patterns; designing portfolio strategy of mode, combined with domain knowledge of adaptive software; designing combination and transformation rules of mode to support the automatic mode evolution based on design patterns; designing composition engine of design patterns to provide efficient and reliable adaptive software development model based on the combination of design patterns, and *etc* are important issues to solve in the future. Based on the research in this area, perhaps one day it will become an important hot topic. Because of space, I can't do in-depth analysis here, but I'd like to play an active part in the research.

References

[1] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum and A. L. Wolf, “An architecture-based approach to self-adaptive software[J]”, IEEE Intelligent Systems & Their Applications, vol. 3, no. 14, **(1999)**.

[2] N. Paspallis, “Middleware-Based Development of Context-Aware Applications with Reusable Components [D]”, PhD Thesis, University of Cyprus, **(2009)**.

[3] R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A. Mamelli and U. Scholz, “MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments [M]”, Proceedings of the Software Engineering for Self-Adaptive Systems, **(2009)**, pp. 164-182.

[4] N. Gui, V. De Florio, H. Sun and C. Blondia, “Toward architecture-based context-aware deployment and adaptation [J]”, Journal of Systems and Software, vol. 2, no. 84, **(2011)**.

[5] D. Weyns, S. Malek and J. Andersson, “FORMS: A formal reference model for self-adaptation [C]”, Proceedings of the 7th international conference on Autonomic computing, Washington, DC, USA, ACM, **(2010)**, pp. 205-214.

[6] O. Moser, F. Rosenberg and S. Dustdar, “Domain-Specific Service Selection for Composite Services [J]”, Software Engineering, IEEE Transactions on. vol.4, no. 38, **(2012)**.

[7] E. Gamma, “Design patterns: elements of reusable object-oriented software [M]”, Reading, Mass.; Wokingham: Addison-Wesley, **(1995)**.

[8] A. J. Ramirez and B. H. C. Cheng, “Design patterns for developing dynamically adaptive systems [C]”, Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, **(2010)**, pp. 49-58.

[9] T. Holvoet, D. Weyns and P. Valckenaers, “Patterns of Delegate MAS [C]”, Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, **(2009)**, pp. 1-9.

[10] J. Dong, Y. J. Zhao and Y. T. Sun, “Design pattern evolutions in QVT [J]”, Software Quality Journal, vol. 2, no. 18, **(2010)**.

[11] H. Gomaa and M. Hussein, “Software reconfiguration patterns for dynamic evolution of software architectures [C]”, Proceedings of the fourth Working IEEE/IFIP Conference on Software Architecture, **(2004)**, pp. 79-88.

[12] H. Gomaa, K. Hashimoto, M. Kim, S. Malek and D. A. Menascé, “Software adaptation patterns for service-oriented architectures [C]”, Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre, Switzerland, ACM, **(2010)**, pp. 462-469.

[13] D. Weyns, R. Haesevoets and A. Helleboogh, “The MACODO Organization Model for Context-Driven Dynamic Agent Organizations [J]”, ACM Transactions on Autonomous and Adaptive Systems, vol. 4, no. 5, **(2010)**.

[14] D. Garlan, J. M. Barnes, B. Schmerl and O. Celiku, “Evolution Styles: Foundations and Tool Support for

Software Architecture Evolution [C]", Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, **(2009)**, pp. 131-140.

[15] J. P. Loyall, R. E. Schantz, J. A. Zinky and D. E. Bakken, "Specifying and measuring quality of service in distributed object systems [C]', Proceedings of the First International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '98), **(1998)**, pp. 43-52.

[16] J. Dowling, "The decentralised coordination of self-adaptive components for autonomic distributed systems [D]", PhD thesis, Trinity College Dublin, **(2004)**.

[17] L. Capra, W. Emmerich and C. Mascolo, "CARISMA: Context-aware reflective middleware system for mobile applications [J]", IEEE Transactions on Software Engineering, vol. 10, no. 29, **(2003)**.

[18] C. Li, Y. Li, J. Wu and Z. H. Wu, "A Service-Oriented Software Model Supporting Dynamic Evolution[J]", Chinese Journal of Computer, vol.7, no. 29, **(2006)**.

[19] P. Yu, X. X. Ma, J. Lv and X. P. Tao, "A Dynamic Software Architecture Oriented Approach to Online Evolution[J]", Journal of Software, vol.6, no. 17, **(2006)**.

[20] G. Huang, Q. X. Wang, H. Mei, F.Q. Yang. "Research on architecture-based reflective middleware[J]", Journal of Software, vol. 11,no. 14, **(2003)**.

[21] B. Ding, H. Wang, D. Shi and X. Li, "A Component Model Supports Trustworthiness-Oriented Software Evolution[J]", Journal of Software, vol. 1, no. 22, **(2011)**.

[22] N. Gui and V. D. Florio, "Transformer: an adaptation framework with contextual adaptation behavior composition support [J]", Software Practice & Experience, vol. 6, no. 43, **(2013)**.

[23] A. Oluyomi, S. Karunasekera and L. Sterling, "Description templates for agent-oriented patterns [J]", Journal of Systems and Software, vol. 1, no. 81, **(2008)**.

[24] C.W. He and K. Q. He, "A role-based approach to design pattern modeling and implementation [J]", Journal of Software, vol. 4, no. 17, **(2006)**.

[25] G. Yin, G. Ni and H. Yao, "Developing Adaptive Software with Adaptive Object-model [J]", Journal of PLA University of Science and Technology, vol. 1, no. 5, **(2004)**.

[26] R. K. Keller and R. Schauer, "Design components: toward software composition at the design level [C]", Proceedings of the 20th international conference on Software engineering, IEEE Computer Society, **(1998)**, pp. 302-311.

[27] R N Chakraborty and J. H. Burgess, "Conceptual modeling of innovation impact on business, technology and society [C]', Proceedings of the IEEE International Conference on Management of Innovation and Technology, **(2000)**, pp. 333-338.

[28] K. N. Loo, S. P. Lee and T. K. Chiew, "UML Extension for Defining the Interaction Variants of Design Patterns [J]", Software, IEEE, vol.5, no. 29, **(2012)**.

[29] R. B. France, D. K. Kim, S. Ghosh and E. J. Song, "A UML-based pattern specification technique [J]", IEEE Transactions on Software Engineering, vol. 3, no. 30, **(2004)**.

[30] T. Taibi and D. C. L. Ngo, "Formal Specification of Design Patterns -A Balanced Approach [J]", Journal of Object Technology, vol. 4, no. 2, **(2003)**.

[31] D. Riehle, "Describing and composing patterns using role diagrams [C]", Proceedings of the WOON, **(1996)**, pp. 137-152.

[32] J. Sun and H. Miao, "Formal Modeling and Evolution of Design Pattern Based on Role[J]", Computer Science, vol. 8,no.36, **(2009)**.

[33] T. Zhang, Y. Zhang, X. Yu, L. Wang and X. Li, "MDA Based Design Patterns Modeling and Model Transformation[J]", Journal of Software, vol. 9, no. 19, **(2008)**.

[34] H. Albin-Amiot, P. Cointe and Y.-G. Gué hé neuc, "Jussien N. Instantiating and detecting design patterns: Putting bits and pieces together [C]", Proceedings of the 16th Annual International Conference on the Automated Software Engineering, **(2001)**, pp.166-173.

[35] P. Alencar, D. Cowan, J. Dong and C. Lucena, "A pattern-based approach to structural design composition[C]", Proceedings The Twenty-Third Annual International Conference on Computer Software and Applications, **(1999)**, pp. 160-165.

[36] X. B. Wang, Q. Y. Wu, H. M. Wand and D. X. Shi, "Research and implementation of design pattern-oriented model transformation [C]', Proceedings of the Computing in the Global Information Technology, **(2007)**, pp. 24-24.

[37] S. R. Judson, R B France and D. L. Carver, "Specifying model transformations at the metamodel level [C]", Proceedings of the 2nd UML Workshop in Software Model Engineering, **(2003)**, pp. 15-23.

[38] J. Dong, S. Yang and K. Zhang, "Visualizing design patterns in their applications and compositions [J]", IEEE Transactions on Software Engineering, vol. 7, no. 33, **(2007)**.

[39] Object Management Group. Meta Object Facility (MOF) 2.0 Query/ View/ Transformation (QVT) http://www.omg.org/spec/QVT/, **(2012)**.

[40] X. Bao, N. Gui and T. Holvoet, "A Design pattern composition platform based on QVT -submitted for publication [J]", Software Quality Control, **(2012)**.

[41] Visual Paradigm, www.visual-paradigm.com, **(2012)**.

[42] J. O. Kephart and D. M. Chess, "The vision of autonomic computing [J]", Computer, vol. 1, no. 36, **(2003)**.

[43] N. Gui, X. Bao and T. Holvoet, "A Visualized Design Pattern Language for Compositions [J]", Software Testing, Verification and Reliability, submitted for publication, **(2013)**.

Authors

Jin Jiangang, M (1972 -), He was born in Gushi County, Henan Province, China on 20/11/1972. Lecturer, postgraduates. The main research directions: Software Engineering and Adaptive Software. National Natural Science Foundation of China (No.: 61202050).E-mail:hn_jjg@126.com. Tel:(+86)13523448299.

Sun Shibao, M (1970 -), He was born in Henan, China on 07/10/1970. M.S. His research interests can be summarized as intelligent information processing, machine learning. Particularly, he is currently interested in rough sets theory and approaches, as well as their applications in information processing.

Bao Xiaoan, M(1973-), Male, Ningbo County, Zhejiang Province, China. Professor. Doctor. The main research directions: Adaptive Software.