Evolution Pattern Verification for Services Evolution in Clouds with Model Driven Architecture

Zhe Wang, Chalmers Kevin, Xiaodong Liu
Edinburgh Napier University
Edinburgh, UK

Abstract

In this paper we will provide some discovered evolution patterns and its detailed underlying constitution components. Following the featured orientated model driven product line engineering approach for service evolution in clouds, an analysis on feature modeling, pattern selection and modeling will be show in this paper.

1. Introduction

This paper is organized as the following sections, section 1 is the introduction part, section 2 is the evolution pattern introduction part, section 3 is the MDA for evolution pattern part, section 4 is the clouds service evolution analysis part. Section 5 is the conclusion for the paper and Acknowledgment in the end.

2. The Evolution Pattern

The following evolution patterns will be described in four different angles, there are the underlying components that constitute the evolution pattern, the working orchestration of these underlying components, the design and feature modeling that related with these underlying components, evolution patterns' life-cycle that driven by MDA to adapt with the changing context of the service running environment and requirement.

2.1. Service Route Evolution Pattern

The service route evolution pattern is mainly focus on supporting the asynchronously interaction between services in clouds. Services in clouds are sometimes has time gap in delivering the messages, the time gap can be caused by internet message blocking , time consuming task processing , customer decision waiting or large data retrieving bottleneck. In order to compensate for the time that consumed by particular service which time consuming demand is less than the others, we can introduce the service callback evolution pattern to achieve the above goal.

Service A, Service B and Service C are three services that interacted with each other for the service user. Among them Service B is the service which sometimes under accessing burden and need more time for logic processing but such long time processing will delay the interaction between Service A and Service C, in order to compensate for the time for Service A and Service C, the service route evolution pattern is needed as the solution for this type of problem.

The service route evolution pattern is constituted by four components and one database, there are filter component, redirection component, retrieving component and weaving component. The filter component is used to recognize the message which is sent by Service A is processable by Service B or not, if the message is non-processable by Service B then it will be send to the redirection component, otherwise it will be send to Service B.

The redirection component is used to redirect the message from Service A directly to be send to Service C without go through Service B, the message that redirected by the component are non-processable by Service B, which can largely reduce the time for Service B to process redundancy message or non-processable message.

That can make the Service B can only focus on its own processable message. The database is used to store the message that need long time processing by service B, which means Service C can asynchronously work with Service B. Further message can be retrieved by the retrieving component from the database to Service C for further processing without affect the working efficiency between Service A and Service C. The weaving component is used to define the exact location where these components should be weaved into the currently running system and the weaving sequence of these components. Figure 1 is the Data Flow Model for the Service Route Evolution Pattern. Figure 2 is the Service Route Evolution Pattern.

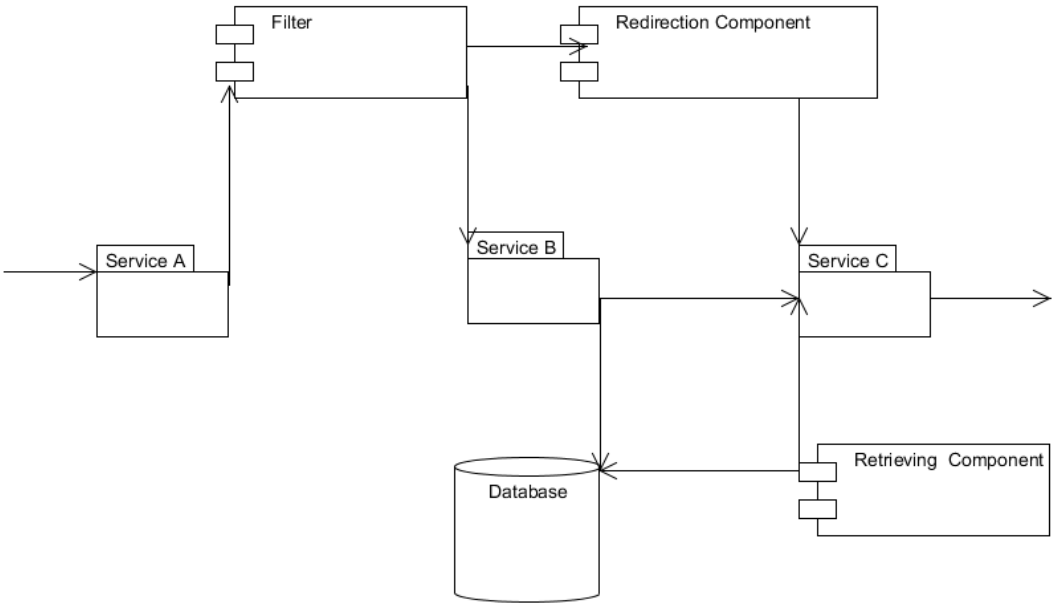

Figure 1. Data Flow Model for the Service Route Evolution Pattern

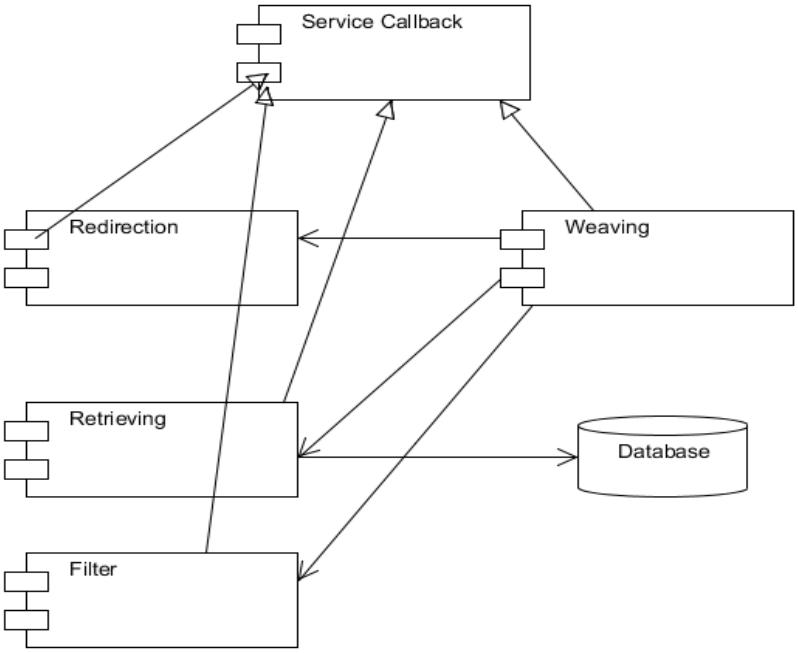

Figure 2. Service Route Evolution Pattern

2.2. Service Inventory Refine Evolution Pattern

The service inventory refine evolution pattern is mainly focus on making the service has more ability in re-composition and re-organization. As the service in clouds needs more flexibility in providing the justified the capability which customer needed in any circumstance and context, the capability of the service needs to be adapted as much as possible. In order to reach the above goal,

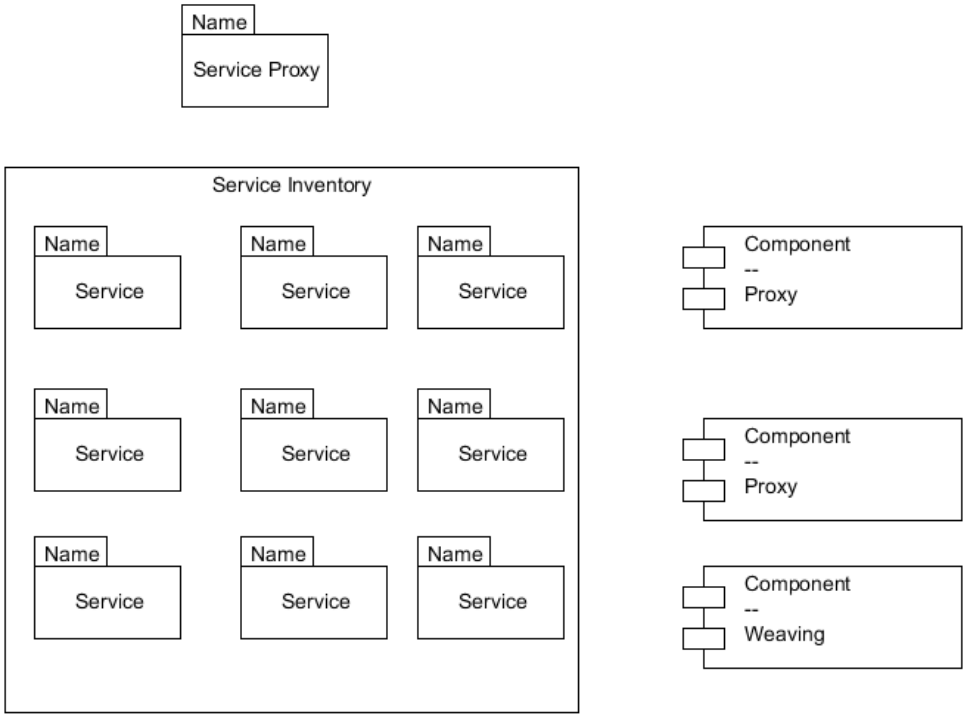

Figure 3.The Service Inventory Refine Evolution Pattern

the service inventory evolution pattern which can enhance the re-composition and re-organization ability of the service inventory is useful in such context. The service inventory refine evolution pattern is constituted by Service proxy, Proxy components and weaving aspects. The function of the service proxy is realized by the proxy components which used to encapsulate the function or refine the function provided by the service inventory. The interface of the service inventory can also be adapted through the accessing of the Service Proxy. By using that evolution pattern, the service inventory's interface and its underlying service logic can both by changed and adapted based on the new requirement of customer context. Such evolution can enhance the efficiency of the service inventory processing ability and highlight the service utility inside the service inventory. Figure 3 is the service inventory refine evolution pattern

2.3. Service Instance Evolution Pattern

The service instance in clouds can be virtualized expanded to support high burden accessing by the customer. In order to take advantage of the utility of clouds computing the service instance evolution pattern is introduce to solve this problem. Service A, B and C are service working in the clouds as a service group. The number of customer using Service A, B and C are dynamically changed based on whether it is peak time accessing or not. For the reason these three services are co-working to together as a group to achieve the same goal decided by the customer , so that we can enhance the efficiency of the service working ability by optimize its instance which belongs to each service. The service which is under burden can be created to have more instances for processing the information instead of having the same instances numbers as these non-burden services have. The optimization of the instances which belongs to the service can be solved by the service instance evolution pattern. The pattern is constituted by instance crater component, instance destroyer component and the weaving aspect. The Instance Creator can create instance for these service which is under burden and the instance destroyer can delete the instance for these service which is free accessing. Figure 4 is the service instance evolution pattern.

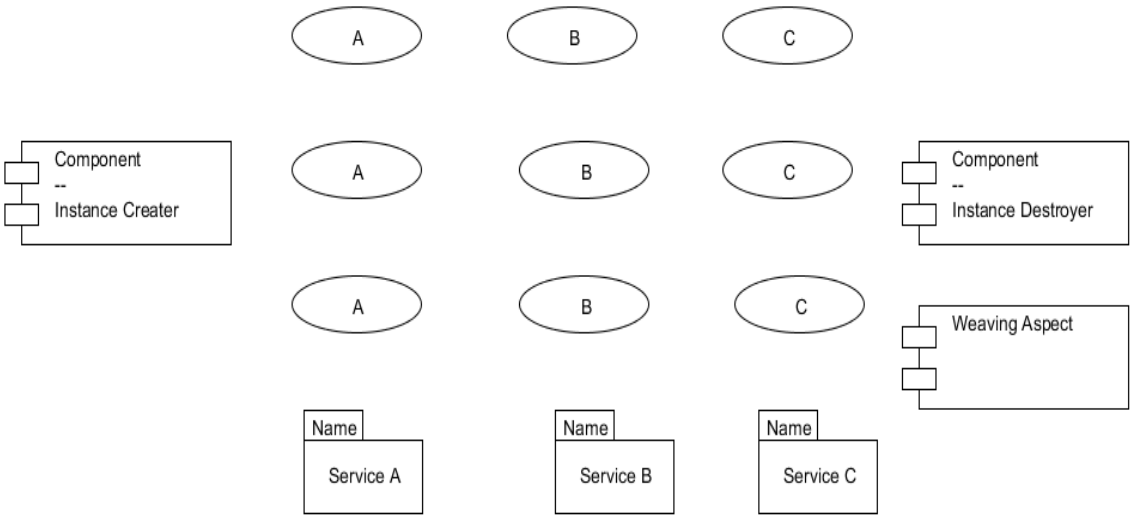

Figure 4. Service Instance Evolution Pattern

2.4. Service Accessing Security Evolution Pattern

With the growing number of customers to access the data store in the clouds storage drive, the analysis on data security is an issue in this area. Enhance the data security for customers with different requirements are a common problem for public clouds and private clouds. The service accessing security evolution pattern can further assure the data security in clouds for various customers. The service accessing security evolution pattern is constituted by Security Broker Service, Security Analysis component, Security Identity Database and Aspect weaving Component. These components will be weaved into the running system through the guidance of the Aspect weaving component. The Security Broker Service is used for verify the customer's intention of why the customer is need to accessing the underlying service and its database. The intention verification including service message sending, which means to verify the message format send by the customer. The content of the message send by the customer, which means the parameter that needs to be processed by the underlying service. The identity which the customer claimed themselves, which means the people's identity checking. All the above information will be compared with the information stored in the Security Identity Database. The customer that pass all the above verification can go continue to accessing the underlying service and database, otherwise the security service broker will redirect the message to other usable service for the customer. Figure 5 shows the Service Accessing Security Evolution Pattern. The weaving for all these component into the running system are based on the definition on the aspect weaving component, such definition is based on the context which the underlying service has, e.g. Which part of the service needs further protection and which part of the database is private or public. Such definition can be very detailed on service function, parameter and data identity.

3. MDA for Evolution Pattern

The Model Driven for these evolution patterns needs to be realized from its underlying components which constitute these evolution patterns. By modeling the evolution feature, the evolution pattern can be selected from the matching of the feature and the component belongs to the evolution pattern. The model driven from requirement to coding is just the process from evolution feature to component selection and then further refinement on the components. The components that constitute the evolution pattern are the entity to achieve the evolution task. It has the platform independent model to the platform dependent model based on which service running platform it works on.

4. Clouds Service Evolution Analysis

4.1. The Underlying Components that Constitute the Evolution Pattern

The underlying components of the evolution pattern are undertaking the entity function to realize the evolution activity, all these components should be designed as atomic element so as to be reused and composited in a timely fashion.

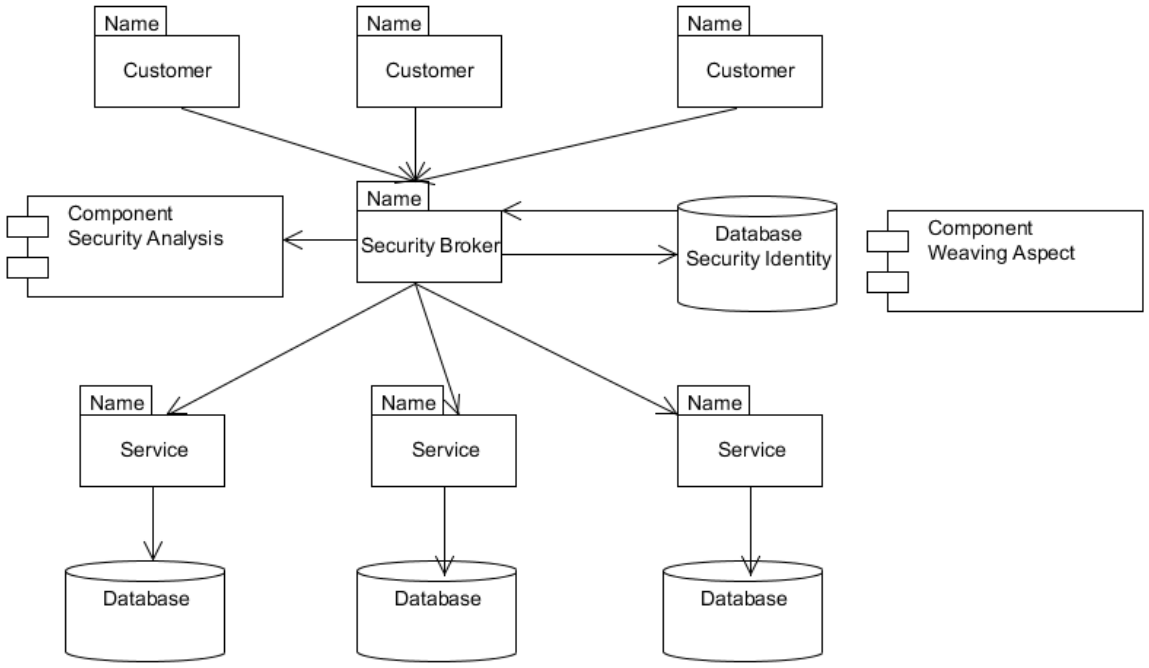

Figure 5.Service Accessing Security Evolution Pattern

For the reason that one evolution pattern is particular to solve one type of evolution problems, so the underlying components of different evolution patterns will have different functions and purpose. Composition or decomposition of these components can generate various types of evolution patterns.

4.2. The Working Orchestration of these Underlying Components

The working orchestration of these underlying components is the detailed working process of each evolution pattern. Adapting the orchestration can further refine the working process of the evolution pattern, which provided more flexibility for the evolution pattern to solve one type of evolution problem in various context and environment.

4.3. The Design and Feature Modeling that Related with these Underlying Components

As the underlying components that constitute the evolution pattern is atomic so it is more compatible to be connected with evolution feature modeling. The evolution feature modeling is a process to capture the evolution requirement in order to make such requirement more precise and accurate. A feature model based evolution model can be used to select the appropriate evolution pattern. As the evolution pattern is constitute by atomic components so such selection process can be done in more underlying layer to order to make the selection more precise and accurate.

4.4. Evolution Patterns' Life-Cycle that Driven by MDA to Adapt with the Changing Context of the Service Running Environment and Requirement

The model driven architecture that used for refine the framework to code always needs a goal to driven the whole process, the goal could be requirement, algorithm, testing standard, modeling parameter or any other factor that can deeply affect the model to code generation. In our approach such factor is the evolution requirement. The evolution requirements define the goal that the evolution should achieve. In order to achieve such goal we introduce the evolution feature to meet the demand of the evolution requirement. Base the evolution feature we select the appropriate evolution pattern through its underlying constitution components. In that way, the MDA model to code generation will mainly based on the refine of these underlying components which constitute the evolution pattern. Such refine process is more precise and accurate because all these atomic components undertaken

the evolution requirement.

5. Conclusion and Future Work

The above analysis is critical to the PhD project to achieve an approach for service evolution in clouds by using service evolution pattern. Such patterns in emergent needed both to academic and the industry. The project will continue to discover deep valuable evolution pattern in the following service evolution case study period. The following case study will mainly focus on scenario back ground testing for these evolution patterns, patterns will be driven by the real evolution requirement to undertake the evolution task in the coding level. The working will mainly focus on the current SOA evolution which affects clouds application and big data era. SOA, clouds and Big data are the three core concepts that sustain the software framework which popularly used in the emergent software solutions. The service evolution that used to enhance the SOA usability in clouds should also concern the co-affection that these three concepts produce as they are working synergically. As the service evolution's focusing is to enhance SOA based usability, such enhancement will inadvertently drive itself to the Big Data area which currently increasingly affects the Service-Oriented Architecture in Clouds. The 'Big Cloud' needs 'Bid Data' to constitute the 'Big Service' that provides to people. Such constitution needs sustentation from service evolution in clouds. For example, 'Big Data' is produced and growing because of today's 'Big People'. People need more data for analyzing and analyzing needs more data, such iteration progress created the 'Big Data' in a cyclicity process. We saw web2.0 growing in the past 5 years speedily more than any other period, web 2.0 is just provide the space and platform that all the people can create and share the data from themselves in a instantly manner. Such activity is originally where the 'Big Data' comes from. With the clouds platform's acceptance by people and industry, more data will be created and deployed to the clouds and data itself can have various formats for accessing. The service evolution problem for data in clouds can also be various, which just the chance for our approach to create such evolution pattern for solving these problems. The accumulation of data is the process that people creating 'Big Data', for the long term of generation of these data makes data more isolated from the other, which makes people introduced SOA initially already, but however, in the clouds environment with service evolution pattern we can found more innovated solution.

6. Acknowledgments

The work in this paper has been jointly sponsored by the British Royal Society of Edinburgh (RSE-Napier E4161) and the Natural Science Foundation of China (Ref: 61070030).

7. References

[1] Wang, Z., Chalmers, K., Cheng, G. (2013). Evolution Feature Oriented Model Driven Product Line Engineering Approach for Synergistic and Dynamic Service Evolution in Clouds. Journal of Industrial and Intelligent Information, The 2013 2rd International Conference on Database and Data Mining, Seoul, Korean, May 11-12, 2013.

[2] Wang, Z., Chalmers, K. (2013). Evolution Feature Oriented Model Driven Product Line Engineering Approach for Synergistic and Dynamic Service Evolution in Clouds:AO4BPEL3.0 Proposal. In: International Conference on Information Society (i-Society 2013). University of Toronto, Canada. June 24-26, 2013. Toronto, Canada.: IEEE.

[3] Wang, Z., Chalmers, K. (2013). Evolution Feature Oriented Model Driven Product Line Engineering Approach for Synergistic and Dynamic Service Evolution in Clouds: Four Kinds of Schema. In: Acadia University, Canada, The 4th International Conference on Ambient Systems, Networks and Technologies, June 25-28, 2013. Halifax, Canada: Procedia Computer Science 00 (2013) 000–000.

[4] Wang, Z., Chalmers, K. (2013). Evolution Feature Oriented Model Driven Product Line Engineering Approach for Synergistic and Dynamic Service Evolution in Clouds: Pattern Data Structure. In: The 7th International Conference on Complex, Intelligent, and Software Intensive System (CISIS 2013) July 3rd - July 5th, 2013, Taiwan. IEEE Computational Intelligence Society.

[5] Cheng, G., An, Y., Wang, Z., Zhu, K. (2012). Oil Well Placement Optimization using Niche Particle Swarm Optimization. In: In Proceedings of CIS'2012 (International Conference on Computational Intelligence and Security). Guangdong, China.

[6] Wang, Z., Liu, X., Chalmers, K., Cheng, G. (2012). Evolution Pattern for Service Evolution in Clouds. In: The 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012), 10-12, December 2012. London, UK: Copyright © ICITST-2012 Published IEEE UK Computer Chapter.

[7] Wang, M., Bandara, K., Pahl, C.Process as a Service - Distributed Multi-tenant Policy-based Process Runtime Governance. 2010 IEEE International Conference on Services Computing 978-0-7695-4126-6/10, Dublin City University, 2010

[8] Yuan, P., Jin, H., Yuan, S., Cao, W., Jiang, L. WFTXB: A Tool for Translating Between XPDL and BPEL. The 10th IEEE International Conference on High Performance Computing and Communications.978-0-

7695-3352-0/08 $25.00 © 2008 IEEE. Huazhong University of Science and Technology, Wuhan, 430074, China, 2008

[9] M. Oliver, R. Florian and D. Schahram ,"Non-Intrusive Monitoring and Service Adaptation for WS-BPEL", Distributed Systems Group, Technical University Vienn Argentinierstr. 8/184-1, 1040 Vienna, Austria, lastname@infosys.tuwien.ac.at

[10] T. Simon and Z. Uwe, "Runtime Process Adaptation for BPEL Process Execution Engines", University of Vienna, Faculty of Computer Science, Software Architecture Group, Vienna, {simon.tragatschnig, uwe.zdun}@univie.ac.at

[11] Diplomarbeit, L. Alexander and Dieburg, "Expressive Scoping and Pointcut Mechanisms for Aspect-Oriented Web Service Composition AO4BPEL 2.0", Technische Universität Darmstadt

[12] Garlan, D. and Schmerl, B. . A tool for defining and planning architecture evolution, 978-1-4222-3452-7/09, IEEELinthicum, D., Cloud Computing, 2009

[13] G Jaroucheh, Z., Liu, X., Smith, S., A MDD-based Generic Framework for Context-aware Deeply Adaptive Service-based Processes, 8th IEEE International Conference on Web Services, USA, 2010

[14] Lee, G. M. and Crespi, N. Shaping Future Service Environments with the Cloud and Internet of Things: Networking Challenges and Service Evolution, Leveraging Applications of Formal Methods, Verification, and Validation, Lecture Notes in Computer Science, Vol. 6415/2010

[15] Sindhgatta, R., Nanjangud, C. and Sengupta, B., Software Evolution in Agile Development: A case Study. Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, ISBN: 978-1-4503-0240-1, 2010

[16] Takabi, H., Joshi, J.B.D. and Ahn, G. Security and Privacy Challenges in Cloud Computing Environments, IEEE Security & Privacy, 8(6), 2010

[17] Cuadrado, F., Garcia, B., Duenas, J.C., Parada, H.A., A Case Study on Software Evolution towards Service-Oriented Architecture, 22nd International Conference on Advanced Information Networking and Applications, 2008

[18] Xie, G., Chen, J., Neamtiu, I., Towards a Better Understanding of Software Evolution: An Empirical Study on Open Source Software, Software Maintenance,. ICSM 2009. IEEE International Conference on, 2009

[19] C. Anis, M. Mira, AO4BPEL: An Aspect-oriented Extension to BPEL, World Wide Web (2007) 10:309–344 DOI10.1007/s11280-006-0016-3.

[20] H. Alejandro, C. Mariano, Extending an Open-Source BPEL Engine with Aspect-Oriented Programming, Also Databases and Distributed Systems Group, TU Darmstadt, Germany.

[21] C. Carine, F. Anthony, Towards an Aspect Weaving BPEL Engine, The Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS) March 2004, Lancaster, UK

[22] Jaroucheh, Z., Liu, X., Smith, S. (2012). An Approach to Domain-based Scalable Context Management Architecture in Pervasive Environments. Personal and Ubiquitous Computing, 16(6), 741-755.

[23] Yang, H., Liu, X. (2012). Software Reuse in the Emerging Cloud Computing Era. Pennsylvania, USA: IGI Global Publishing.

[24] Jaroucheh, Z., Liu, X., Smith, S., Zhao, H. (2011). Lightweight Software Product Line Based Privacy Protection Scheme for Pervasive Applications. In: Proceedings of the 35th IEEE COMPSAC. Munich, Germany: IEEE Computer Society.

[25] Jaroucheh, Z., Liu, X., Smith, S. (2012). A Unified Approach for the Dynamic Evolution of Context-aware Services. In: Proceedings of International Conference on Innovations in Computers, Information and Communication - ICICIC 2012. India: PSG Tech.

[26] SOA Principles of Service Design, Thomas Erl (ISBN:0132344823,Hardcover,Full-COlor, 240+ LLLustrations) Publisher: Prentical Hall www.servicetechbooks.com 2007

[27] SOA Design Patterns, Thomas Erl, (ISBN 13:9780136135166 ISBN 10:0136135161) Publisher: Prentical Hall

