A MIDDLEWARE BASED, POLICY DRIVEN ADAPTATION FRAMEWORK TO SIMPLIFY SOFTWARE EVOLUTION

N. H. Awang, W. M. N. Wan Kadir
Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Johor Baharu, Malaysia

S. Shahibuddin
Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Johor Baharu, Malaysia

Keywords: Software evolution, Software adaptation, Middleware, Web service, Close-loop feedback system.

Abstract: Evolution is said to be one of the main causes of problems for software. Unplanned evolution exposes an organization to high software maintenance cost. Due to these facts, we embark on this research to create a framework for simplifying software evolution. This paper presents a framework, called Middleware-based Policy-driven Adaptation Framework (MiPAF). MiPAF has the aim to control the negative effects of software evolution using the concept of software adaptation, supporting both parameterized and compositional adaptation.MiPAF is implemented using well established foundations, i.e. middleware and web service. These two concepts are well accepted by software developer's community; therefore the chances of MiPAF to be accepted and used by this community are increased. The adaptation mechanism of MiPAF is driven by XML based policy. To evaluate MiPAF, we implement the framework using C language and run it on Windows platform. An existing unit trust system (UTS) is used for evaluation.

1 INTRODUCTION

Change is something that is inevitable in the software lifecycle due to business requirements and technology advancement (Godfrey and German, 2008, Roland, 2001). The term *evolution* in the context of software refers to changes that happen to software during its lifetime (Reiss, 2005). The system must continue to evolve to correct defects, add or remove functional behaviours, and adapt to the operating environment.

As software evolves, its quality will degrade (Lehman, 1996). Evolution is said to be one of the main cause of problems related to software systems such as high coupling and low cohesion (Reiss, 2005). Subramanyam indicated that high cost and high risk are associated with unplanned software evolution (Subramanyam, 2008). Evolution that occurs to any software at a later lifecycle is costly (Stephens and Rosenberg, 2003). With these facts and challenges surrounding software evolution, it is very important that a method or approach is introduced to simplify software evolution with the aim to control the negative effects of software evolution.

Controlling negative effects of software evolution requires a proper support for managing changes in software development (Mens et al., 2005b, Mens et al., 2005a). One of the promising ways to control software evolution is via software adaptation. Software adaptation refers to the ability of software to readjust itself whenever changes happen in order to meet its development purpose (Mens et al., 2005b). The work in software adaptation ranges from the development of generic architectural framework to specific middleware for specialized domains. In this paper, we propose a framework to enable enterprise software to become adaptable due to changes in non-functional requirements. The framework, which is called Middleware-based Policy-driven Adaptation Framework (MiPAF), is developed using middleware-based approach, policy driven and make use of web services to enable the adaptability of enterprise software.

The organization of this paper is as follows: in Section 2, we describe some background and related

Awang N., Wan Kadir W. and Shahibuddin S..
A MIDDLEWARE BASED, POLICY DRIVEN ADAPTATION FRAMEWORK TO SIMPLIFY SOFTWARE EVOLUTION.
DOI: 10.5220/0003467502450253
In *Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2011), pages 245-253
ISBN: 978-989-8425-57-7
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

work of this research. In Section 3, MiPAF is described in detail. Some experimentation result is described in Section 4. We end this paper with conclusions and intended future works.

2 RELATED WORK

Software evolution occurs due to changes subjected to the software during its entire lifetime. A mechanism to adapt to these changes must exist in order to avoid the software from being defective or unnecessarily complex (Mens et al., 2005b). Software adaptation is one of the mechanisms in simplifying software evolution. There are many researchers that focus on software adaptation as a mean to control software evolution such as in (Garlan et al., 2004, Holger et al., 2007, Lundesgaard et al., 2007, Oreizy, 1999, Zhang et al., 2008).

In general, two main approaches have been identified by researchers for the implementation of adaptive or adaptable software, namely *compositional adaptation* and *parameterized adaptation* (McKinley et al., 2004). Compositional adaptation refers to the ability of the software to dynamically reconfigure itself at run-time to suit the changing operating environment. Parameterized adaptation or static adaptation involves modification of program variables that directly affect the system behaviour.

Researchers have proposed various ways in enabling software adaptation. In researches such as (Frei et al., 2003, Maciel da Costa et al., 2007) Aspect Oriented Programming (AOP) is used in providing dynamic adaptation capability. Reflection technique is used to enable adaptation in (Ghoneim, 2007, Keeney, 2003). One of the weaknesses of these two approaches is that they are supported by limited programming languages such as AspectJ and Meta Java. Furthermore, specific knowledge which is not common among application developers is required in order to use this approach. Therefore, we argue that there is a need to implement adaptation capability using mechanisms that are well accepted by application developers to increase the chances of the approach to be easily used.

Architecture-based approach to software adaptation is used in (John and Richard, 2008, Oreizy, 1999, Michel et al., 2001). Software architecture is typically used at the design time and it does not specify how the system is to be constructed (Garlan et al., 2004). However, to enable adaptation, the resulting architecture model will be used during run-time. There is a gap between the architecture modelling and the actual construction of the adaptive software. This gap can cause loss of knowledge on the software architecture since some of the information about system properties and constrain are not made explicit (Zhang et al., 2008).

Agent-oriented approach is used by a number of researchers to achieve software adaptation. Related works can be found in (Qureshi and Perini, 2008, Seungwok et al., 2007). To achieve useful works or goals, a group of agents must cooperate and communicate efficiently. Communication between agents depends on asynchronous messaging; therefore, there are higher possibility of communication latency in agent-based system that will affect performance and latency of the system (Tarkoma and Laukkanen, 2003). Novice developer will not get many benefits from agent-oriented approach since it requires change of paradigm in developing software (Paek and Kim, 1999).

MiPAF is developed to address the existing limitations in the current approaches mentioned above. The following sub-sections describe the related approaches that influence and motivate our current research.

2.1 Middleware-based Approach

We have identified four main approaches to software adaptation namely Architecture-based, Component-based, Agent-based and Middleware-based. A comparative evaluation studies on these four approaches has been carried out based on a set of criteria (Awang et al., 2009) The result of the comparative evaluation studies shows that Middleware-based approach obtained the highest score. Therefore, Middleware-based approach has been chosen to implement MiPAF.

Several mechanisms that support adaptations are used in the development of adaptive middleware. Among the popular ones include computational reflection, Aspect-Oriented Programming, and the most recent mechanism is by using web service. MiPAF makes use of web service to provide supports for adaptation.

2.2 Web Service Approach

Web service is described by Dietel as a “mechanism that facilitates computer application to communicate over the Internet using a set of accepted standard” (Deitel et al., 2003). Other definitions of web service such as in (Meyer-Wegener, 2005) and (San-Yih et al., 2007) also emphasize that the communication of web services is using the Internet.

However, for the implementation of MiPAF, the web service can also be used in a non-Internet environment. Web Service is selected in implementing MiPAF due to the following reasons:-

- Web service provides means to implement compositional adaptation since it can be invoked by a running application. The dynamic adaptability of web services is achieved through their loosely-coupled components and run-time method or service invocations. (Foggon et al., 2004).
- Web service is governed by well accepted standards and widely used nowadays therefore the probability of software developer to accept MiPAF is higher.
- Web service offers a scalable implementation since services can be hosted in different machines.

2.3 Policy-based Approach

Policy based approach is selected to be used as a method to drive adaptation. The policy comprises some logics in the form of CONDITION and ACTION; IF a CONDITION occurs, what ACTIONS need to be executed. CONDITION is closely related to the source of change, whether it is known prior to run time or it is detected by the Context Monitor when the application system is running.

The content of the policies are the adaptation decisions. The adaptation decision is captured from the designer's or developer's knowledge of the business rules. Application developers will specify the adaptation logic in the policy; therefore, it becomes apparent that the way the policy can be written must be simple and easy to learn. Difficult way of writing the adaptation policy will hinder the usage of MiPAF. This fact becomes one of the important requirements in the development of policy language for MiPAF.

MiPAF only requires a simple but extendable policy language to ensure its expansion in the future. Existing generic policy based framework such as PONDER is too complex to be used as policy management for MiPAF. Therefore, MiPAF policy language makes use of XML to specify CONDITION and ACTION. XML is chosen due to its popularity, open standard and there are many existing tools available to parse the XML. MiPAF is using Expat XML Parser to parse its policy document.

3 THE PROPOSED APPROACH

Our research focuses on simplifying or controlling the negative effect of software evolution using software adaptation. We develop a framework called MiPAF using foundations described in Section 2.

The following figure shows the composition diagram of MiPAF.

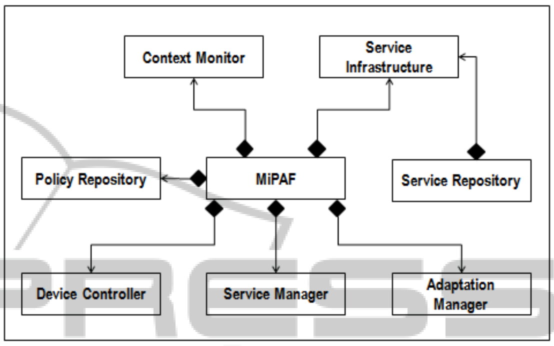

Figure 1: MiPAF Composition Diagram.

Figure 1 shows that MiPAF composes of six main components. The description of each component is as follows:-

3.1 Context Monitor

Context Monitor refers to a program that monitors a set of environment elements (such as devices' state). The Context Monitor will only monitor environment elements that are meaningful to the software, i.e. changes of states of these elements will require some adaptation to be carried out to ensure the software meet its development purposes. The requirements of the Context Monitor are as follows:-

- The implementation must be independent from the elements or components it monitors. Failures occurred at the monitored components should not affect the behaviour of the Context Monitor
- The Context Monitor must be generic so that only one Context Monitor is implemented to monitor various context components
- The Context Monitor must inspect the state of monitored components at a specific interval. Software developer should be able to change this inspection interval without the need to stop the Context Monitor
- The Context Monitor must have a way to communicate the monitoring status to the Adaptation Manager in real time manner. Furthermore, a trigger mechanism must be in

place to ensure that any context change can be quickly analysed and adapted.

These requirements form a basis for the evaluation of existing Context Monitor or for developing a new Context Monitor. For the implementation of MiPAF in this research, an existing Context Monitor called Health Checker will be modified to suit the above requirements. Health Checker is an application software developed by HeiTech Padu to monitor the "health" of the devices used by application system.

3.2 Service Infrastructure

MiPAF uses web services to implement dynamic behaviour of the application that need to be adapted. We are aware that one important source of change is the change in user requirements that affect the business functionalities of the application system. However, changes in business functionalities are not within the scope of this research. In this research scope, dynamic behaviour that is implemented using web services are related to the non-functional requirements such as the use of specific devices.

The requirements for the Service Infrastructure are as follows:-

- There must be a way to store services hosted by the Service Infrastructure. This web service storage will be referred to as Service Repository.
- The Service Infrastructure must support SOAP, therefore, as a prerequisite; the infrastructure must be able to handle HTTP.
- The Service Infrastructure must contain an XML Parser to parse SOAP messages

In a typical implementation of web services, the role of Service Infrastructure is played by a web server. However, for the implementation of MiPAF, the use of web server will unnecessarily complicate the implementation. With the above requirements, the Service Infrastructure can be thought as a "mini" web server with limited capabilities. Having said that, it is stressed here that, the limited capabilities is enough to serve our purpose.

3.3 Device Controller

Device Controller encapsulates the complexity of device integration into an application system. The problem with device integration is always the "non-standard" way of accessing device. MiPAF provides a method for an application system to systematically accessing devices using web service. However, some legacy devices such as IBM 4722 printer does not provide a ready support for web service. To overcome this problem, MiPAF provides Device Controller as a "bridge" to access legacy devices.

The Device Controller can not be accessible directly by application developer. The requirements for a Device Controller are as follows:-

- Device Controller must be written for each device since it is highly dependent on specific device command. Therefore, the number of Device Controller is equal to the number of devices used by the application.
- There must be a way of communication between Device Controller and Context Monitor. The Context Monitor will get the device states from each Device Controller.
- There must be a way of communication between the Service Manager and the Device Controller. This is due to the reason that request for accessing devices will come via the Service Manager.

3.4 Adaptation Manager

Adaptation Manager is the main intelligence of MiPAF. The Adaptation Manager provides runtime mechanism to enable the right adaptation to be executed when changes occur. In order to adapt to the changes, in general, the Adaptation Manager must:-

1. Realize that changes have occurred.
2. When the changes occur, decide what to do, i.e. what kind of adaptation need to be executed.

Change to the application system can be classified into two main sources i.e. changes that are introduced by users and changes due to the environment factors (changes in environment context). All changes that the application is interested in must be able to be further categorized to ensure the best adaptation to be carried out. It is very important to define the changes extensively to ensure all interested changes are catered by the Adaptation Manager. Among the changes that are catered by MiPAF are as follows:-

- *NOT EXIST* - where device is not exist at the expected location or the device maybe malfunction
- *CHANGE LOCATION* – where device is changed from one workstation to another
- *NEW DEVICE* – where new brand of device or new device is added to the system.

As stated earlier, two sources of changes are considered i.e. changes introduced by the users and changes in the environment context of the software.

Changes in the Environment Context
Changes in the environment context of the software is detected by the Context Monitor and communicated to the Adaptation Manager. The processes that occur in order for the Adaptation Manager know about the changes are as follows:-

- Context Monitor monitors the environment of an application system continuously. This is shown in the diagram by the arrows from "Any Change" decision that point back to the "Monitor Environment" process regardless of the result of "Any Change" decision
- If relevant changes are detected, the Context Monitor will update device status in a shared location. This shared location can be implemented using shared memory concept and an event will be fired indicating that there is a change that occurs
- The event will trigger Adaptation Manager and the Adaptation Manager will execute necessary adaptation.

Changes Introduced by Users
Changes introduced by users can be implemented prior to the running of application. Referring to our example, the example of such changes may include implementing a new device to the application system. In this case, Adaptation Manager will not be invoked. However, MiPAF allows this type of change to occur in a controlled manner. The software developer needs to reflect the current change by editing the policy.

3.5 Service Manager

Service Manager is the first point of contact between the application system and MiPAF. Communication between the Service Manager and the application system is done either via socket based communication or via SOAP. This implies the requirements of the Service Manager that it must support two communication interface; socket and Simple Access Object Protocol (SOAP). The reason for the two interfaces is that, for a local application (where application and MiPAF runtime located on the same workstation), socket based communication is preferred due to its simplicity and better performance. If MiPAF runtime is located on different workstation, SOAP is more preferable.

Service Manager implements a listener that continuously listens for service request. This statement implies another requirement of the Service Manager that it must be implemented using multi-threaded technology. Upon receiving the request to use a device, the Service Manager will locate a specific web service that will fulfil the request. The location of the service is kept in the service profile. Service profile will make use of the Service Infrastructure to invoke the actual web service that handle the device. The web service will access the device via a Device Controller.

3.6 Policy Repository

Policy Repository consists of all policy documents that dictate how access to the device is to be done. Each software or application that uses MiPAF will have a default policy that describes the non-functional requirements such as the type of devices used. The policy also describes the adaptation logic of an application system. The language used to specify policies is XML.

3.7 Service Repository

Service Repository consists of all available web services that can be used by the software. The repository acts as a host for the web services.

3.8 MiPAF Policy Language

MiPAF uses the concept of policy in driving the adaptation. Logic related to what action to be taken when certain conditions occurs is described in the policy of the application. MiPAF Policy Language is developed based on XML. Users need to create the policy file for application system that needs to be adapted. This policy file will be amended whenever required to ensure adaptability of the application. A tool will be developed to assist software developer in creating and editing policy files.

As mentioned in previous section, MiPAF will not make use of full blown policy based language since the requirements for MiPAF policy is not complex.

4 EVALUATION OF THE PROPOSED APPROACH

In order to evaluate the proposed approach, we apply MiPAF to an existing system, UTS. UTS is a counter-based application that manages unit trust investment. This system is currently in used by a public fund manager in Malaysia. It is installed in all branches of the said fund manager all across Malaysia.

Using MiPAF requires segregation between non-functional requirements that requires adaptation and functional requirements of the system. For UTS, the non-functional requirements that require adaptive capability is related to the use of devices. UTS is a client/server based system that uses three type of devices as described below:-

- IBM 4722 Printer
  This is a legacy printer that is used to print passbook. This printer can not be shared using default sharing mechanism provided by the operating system. In the existing implementation of UTS, this printer is shared between two workstations to save cost.
- MyKad Reader
  MyKad Reader is used to read Malaysian Citizen Identity Card or better known as MyKad. The device is also sharable between two adjacent workstations.
- Thumbprint Scanner
  This device is used to scan the thumbprint images of customers. This device is not sharable between workstations.

4.1 Scenario for Evaluation

For the evaluation, we create the following scenario of adaptation requirements:-

A. Introduction of a new device. Initially, the only device required by UTS is IBM 4722 passbook printer. Then we add another device, Sekure 2 MyKad Reader.
B. Device failure or device not exist
C. Location of the sharable device is changed – for example, initially, MyKad Reader is attached to Workstation A but it then changed to Workstation B.

From the above requirements, we segregate the changes into two types; changes that are known before the software is loaded and changes that happen while the software is running. A is classified as change of the first type while B and C are of the second type.

4.2 MiPAF Adaptation Mechanism

Adaptation for changes in MiPAF is parameterized into the system in the forms of policy. For the first type of change, MiPAF will read the default policy and choose appropriate services. For the second type of change, Context Monitor will detect the changes and choose the right policy from Policy Repository.

Part default policy for UTS is as follows:-

```
<app_name = “UTS” ver = “1.0 >
<print>
<dev-printer = “ibm4722” svc_name =
“svc_ibm4722” buffer-in = “1470”
buffer_out = “1471”host = “Y” port=
“1”>
<prt_fail = “Y” error = “error.log”
retry = “3” host = “172.19.37.102”>
</dev-printer>
</print>
</app_name>
```

In this implementation, when no adaptation required, UTS issues “print” command to indicate the need to invoke printing functions. To communicate with MiPAF, two options are supported; i.e. via socket or SOAP. UTS used socket-based communication since it resides on the same PC with MiPAF. Upon receiving the request, the Service Manager will load the right policy from Policy Repository. The Service Manager will parse the policy file and get the service name and related information under the tag <dev_printer>. The Service Manager will search the service repository to know the location of the service and invoke the service using the Service Infrastructure. The services will use the Device Controller to actually send data to the passbook printer. Data is passed among different components in MiPAF using shared memory.

For Scenario A, when MyKad Reader is added to the system, the policy file needs to be edited. New entry will be added to the policy file:-

```
<MyKad>
<dev-mykad= “sekure” svc_name =
“read” buffer-in =”1472” buffer_out
= “1474” host = “Y” port = “usb”>
<mykad_fail = “Y” error =
“error.log” retry = “3” host =
“172.19.37.102”> port=”usb”</dev-
mykad>
</Mykad>
```

Scenario B is tested by removing MyKad reader from the workstation. Device Controller of each device will update the health of each device in the shared memory periodically. Context Mo nitor on the other hand, will scan the devices’ statuses in the shared memory and alert the Adaptation Manager if any errors occur. The Adaptation Manager will scan through the policy and decides on what to dobased on this statement:-

```
<mykad_fail = "Y" error =
"error.log" retry = "3" host =
"172.19.37.102"> port="usb"
comm.="socket"</dev-mykad>
```

The policy stated that if the defice fail, error is logged in error.log file, Device Controller have to retry communicating to the device for three times. If the failure persists, Adaptation Manager will call Device Controller located at 172.19.37.102 using socket-based communication and port used is USB. This flow also applies to scenario C.

4.3 Results of Evaluation

We have listed six evaluation criteria to compare approaches to adaptation in our previous paper (Awang et al., 2009). The same criteria are used to evaluate MiPAF. The results of the evaluation are as follows:-

- Scalability: MiPAF is scalable since it is designed to work in a distributed environment. In this implementation, MiPAF is installed on two workstations, managing separate devices. The Service Manager supports multi-threaded technology, therefore, as applications grow, or as more applications are using MiPAF on single PC, the number of threads will increase thus making MiPAF scalable as needed.
- Context Awareness: Context awareness refers to the ability of MiPAF to detect changes in the operating environment. MiPAF achieve context awareness through the implementation of Context Monitor. Context Monitor periodically scan for the statuses of devices attached to the workstation. Not all changes to the context are relevant to applications, therefore, in MiPAF, we have defined the type of changes that are of interest to the application.
- Performance: We have not performed any quantitative observation on the performance of MiPAF. However in MiPAF, the concerns for performance are address earlier, during the design time. As a result to that, MiPAF supports two ways of communication; socket-based and SOAP-based communication. The intention of using socket-based communication is to increase performance.
- Usability: This attribute refers to how easy the approach 0can be used by developers. We designed MiPAF with this requirement in mind to ensure MiPAF is well accepted by developer's community. Therefore, we used well accepted approach such as XML based policy language and web services.
- Heterogeneity: MiPAF can be developed using any programming languages and executed in any platforms. Our implementation is using C and the platform used is Windows. Any developers can use MiPAF specification and develop it using other language such as Java. Another aspect of heterogeneity is the ability of MiPAF to entertain requests from different type of applications since the supported communications are based on open standard.
- Dynamic Evolveability: MiPAF enables a controlled, dynamic evolveability of an application by segregating codes that perform business function and codes that enable the execution of non-functional requirements. As changes to non-functional requirements occur, business functions are not affected.

From our initial evaluation results, MiPAF has shown promising opportunities in simplifying software evolution.

5 CONCLUSIONS AND FUTURE WORK

This research mainly consists of the development of MiPAF, its policy language that drive the adaptation and the process of using the framework to simplify software evolution in an enterprise system. Further, this research provides tools to assist software developer in using MiPAF. MiPAF is a framework that can be implemented using any programming language. In our experimentation, we implement the framework using C programming language and the target platform is Windows.

Due to the advantages offered by middleware approach, particularly the benefits of segregating codes that drive adaptation and codes that implement the business rules, MiPAF is implemented using this approach. The benefit of implementation includes scalability, heterogeneity, better performance and evolvability of the application systems to be adapted.

The compositional or dynamic adaptation is implemented using web services. This approach is selected due to several reasons such as it is an open standard and widely accepted in the IT industry. As a result, it will increase the chance of practitioners to embrace MiPAF.

MiPAF make use the concept of policy to drive adaptation. This approach allows user to segregate adaptation requirements into separate components. Adaptation logic can be specified outside the Adaptation Manager thus enable loose coupling between the components in the framework. The use of policy enables user to easily specify the adaptation logic in an XML format. Since XML is well accepted by the IT community, the representation of policy using XML is thought to be a good choice since it will increase the software developer acceptance of the framework.

We are now in the midst of implementing a more elaborated implementation of the framework. In the above experimentation, we use socket-based communication. In future, we plan to experiment with SOAP-based communication. Further, we plan to do more evaluation testing on other systems such as banking front-end system. To ensure that the framework can be used by both client/server and web based application, we plan to test the framework implementation with both architectures.

Currently we have not developed any tools to ease developers in using MiPAF. In the near future, we plan to develop a tool to specify adaptation policy, without the need for them to create raw XML file. The tool will also verify the validity of the policy created.

REFERENCES

Awang, N. H., Wan Kadir, W. M. N. & Shahibuddin, S. (2009) Comparative Evaluation Of The State-Of-The Art On Approaches To Software Adaptation. *Fourth International Conference On Software Engineering Advances.* Porto.

Deitel, H. M., Deitel, P. J., Waldt, B. D. & Trees, L. K. (2003) *Web Services A Technical Introduction*, Prentice Hall.

Foggon, D., Maharry, D., Ullman, C. & Watson, K. (2004) Programming Microsoft .Net Xml Web Services. Microsoft Press.

Frei, A., Popovici, A. & Alonso, G. (2003) Event Based Systems As Adaptive Middleware Platforms. *Workshop Of The 17th Europeean Conference For Object-Oriented Programming, Darmstadt, Germany.*

Garlan, D., Cheng, W. C., Huang, A. C., Schmerl, B. & Steenkiste, P. (2004) Rainbow: Architecture-Based Self-Adaptation With Reusable Infrastructure. *Ieee Computer Society*, 46 - 54.

Ghoneim, A. M. A. (2007) Reflective And Adaptive Middleware For Software Evolution Of Information Systems. *Fakultät Für Informatik* Germany, Otto-Von-Guericke-Universität Magdeburg.

Godfrey, M. W. & German, D. M. (2008) The Past, Present, And Future Of Software Evolution. *Frontiers Of Software Maintenance, 2008. Fosm 2008.*

Holger, K., Dirk, N. & Andreas, R. (2007) A Component Model For Dynamic Adaptive Systems. *International Workshop On Engineering Of Software Services For Pervasive Environments: In Conjunction With The 6th Esec/Fse Joint Meeting.* Dubrovnik, Croatia, Acm.

John, C. G. & Richard, N. T. (2008) Policy-Based Self-Adaptive Architectures: A Feasibility Study In The Robotics Domain. *Proceedings Of The 2008 International Workshop On Software Engineering For Adaptive And Self-Managing System.* Leipzig, Germany, Acm.

Keeney, J., Cahill, V. (2003) Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework. *Ieee 4th International Workshop*, 3-14.

Lehman, M. M. (1996) Laws Of Software Evolution Revisited. *Proceedings Of The 5th European Workshop On Software Process Technology.* Springer-Verlag.

Lundesgaard, S. A., Arnor, S., Oldevik, J., France, R., Aagedal, J. O. & Eliassen, F. (2007) Constriction And Execution Of Adaptable Applications Using An Aspect-Oriented And Model Driven Approach. *Lecture Notes In Computer Science.* Springer Berlin/Heidelberg.

Maciel Da Costa, C., Da Silva Strzykalski, M. & Bernard, G. (2007) An Aspect Oriented Middleware Architecture For Adaptive Mobile Computing Applications. *Computer Software And Applications Conference, 2007. Compsac 2007. 31st Annual International.*

Mckinley, P. K., Sadjadi, S. M., Kasten, E. P. & Cheng, B. H. C. (2004) Composing Adaptive Software. *Computer,* 37, 56-64.

Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R. & Jayazeri, M. (2005a) Challenges In Software Evolution. *Principles Of Software Evolution, Eighth International Workshop On.*

Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R. & Jazayeri, M. (2005b) Challenges In Software Evolution. *Principles Of Software Evolution, Eighth International Workshop On.*

Meyer-Wegener, K. (2005) Thirty Years Of Server Technology - From Transaction Processing To Web Services. *Lecture Notes In Computer Science.* Springer Berlin / Heidelberg.

Michel, W., Antonia, L. & Jose, L., Fiadeiro (2001) A Graph Based Architectural (Re)Configuration Language. *Proceedings Of The 8th European Software Engineering Conference Held Jointly With 9th Acm Sigsoft International Symposium On Foundations Of Software Engineering.* Vienna, Austria, Acm.

Oreizy, P., Medvidovic, N., Taylor, R.N., Gorlick, M.M., Heimbigner, D., Johnson, G., Quilici, A., Rosenblum, D.S., Wolf, A.L. (1999) An Architecture Based Approach To Self-Adaptive Software. *Ieee Intelligent System*, 54 - 62.

Paek, K. & Kim, T. (1999) *Aom: An Agent Oriented Middleware Based On Java*, Springer Berlin/Heidelberg.

Qureshi, N. A. & Perini, A. (2008) An Agent-Based Middleware For Adaptive Systems. *Quality Software, 2008. Qsic '08. The Eighth International Conference On.*

Reiss, S. P. (2005) Evolving Evolution [Software Evolution]. *Principles Of Software Evolution, Eighth International Workshop On.*

Roland, T. M. (2001) Software Evolution: Let's Sharpen The Terminology Before Sharpening (Out-Of-Scope) Tools. *Proceedings Of The 4th International Workshop On Principles Of Software Evolution.* Vienna, Austria, Acm.

San-Yih, H., Ee-Peng, L., Chien-Hsiang, L. & Cheng-Hung, C. (2007) On Composing A Reliable Composite Web Service: A Study Of Dynamic Web Service Selection. *Web Services, 2007, Icws 2007. Ieee International Conference On.*

Seungwok, H., Sung Keun, S. & Hee Yong, Y. (2007) Dynamic Software Adaptation With Dependence Analysis For Multi-Agent Platform. *Computational Science And Its Applications, 2007. Iccsa 2007. International Conference On.*

Stephens, M. & Rosenberg, D. (2003) Extreme Programming Refactored: The Case Against Xp. Apress.

Subramanyam, R. (2008) Position Statement: How Well Technology Supports Software Evolution. *Computer Software And Applications, 2008. Compsac '08. 32nd Annual Ieee International.*

Tarkoma, S. & Laukkanen, M. (2003) *Adaptive Agent-Based Service Composition For Wireless Terminals*, Springer Berlin/Heidelberg.

Zhang, H., Ben, K. & Zhang, Z. (2008) A Reflective Architecture-Aware Framework To Support Software Evolution. *Young Computer Scientists,2008. Icycs 2008. The 9th International Conference For.*