Journal of Theoretical and Applied Information Technology
31st December 2014. Vol.70 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A MIDDLEWARE AND POLICY-BASED ADAPTATION FRAMEWORK TO SIMPLIFY SOFTWARE EVOLUTION: AN IMPLEMENTATION ON UNIT TRUST ENTERPRISE SYSTEM

[1]N.H AWANG, [2]S. SHAHIBUDDIN, [3]W.M.N WAN KADIR
[1]HeiTech Padu Berhad, Kuala Lumpur Malaysia
[2]Professor, Advanced Informatics School, Universiti Teknologi Malaysia, Johor Baharu,Malaysia
[3]Assoc. Prof., Faculty of Computing, Universiti Teknologi Malaysia, Johor Baharu,Malaysia
E-mail: [1]hazila@heitech.com.my, [2]shamsul@utm.my, [3]wnasir@utm.my

ABSTRACT

Software evolution needs to be properly controlled to avoid huge problems during maintenance phase. Software needs to evolve to ensure it meets its development purpose. One of promising ways to address the issue of software evolution is via software adaptation. There are 4 main approaches to software adaptation i.e. architecture-based, component-based, agent-oriented and middleware-based. This research is adopting middleware-based approach to software adaptation. An adaptation framework called MiPAF, which uses middleware and policy-based concept is proposed to simplify software evolution. MiPAF comprises 6 components namely Service Manager, Adaptation Manager, Service Infrastructure, Device Controller, Policy Repository and Context Monitor. The use of MiPAF will affect 4 software development phases i.e. requirement, analysis, design, and development. MiPAF runtime is developed to enable adaptation of the device layer of a Unit Trust Enterprise System (UTES). A simple, XML-based policy language is developed to specify what action to be taken when certain condition happens. The adaptation requirements of this system is specified and an adaptation policy is developed according to the requirements. In this implementation, MiPAF runtime is developed using C language and it is installed on workstations together with UTES client. There are 2 adaptation requirements for this implementation. The first requirement is when a passbook printer fail, the system can proceed with printing using another passbook printer without interruption. The second requirement is that when the type of passbook printer is changed, the system should not be impacted. The evaluation is done against 6 evaluation criteria; scalability, context-awareness, performance, usability, heterogeneity, and dynamic-evolveability. MiPAF meets all the mentioned criteria.

Keywords: *Software Evolution, Software Adaptation, Middleware, Policy, Framework*

1. INTRODUCTION

The term "software evolution" can be briefly defined as all changes that happen to software during its entire lifetime (1). Software needs to evolve to cater for various changes that happen both to its operating environment and its business requirements. Thus, software evolution is a process, which is inevitable so that software can continuously supports its development purposes (2). Lehman argued in (3) that the quality of software will degrade when it evolves. When the quality of software degrade during evolution process, it becomes less reliable and operational risks will increase. In his research, Reiss mentioned that evolution is one of the reason for high coupling and low cohesion for software, hence becoming a major source of problems in the maintenance phase of software life-cycle (1).

Study on software evolution are classified into two main categories, namely, the research on tools and methods to enable software to evolve in a more controlled manner and the study on the nature of software evolution (4, 5). This research is taking the path of the first category, i.e. looking into approaches to minimize the impact caused by software evolution. Within the first category, there are a number of approaches that can be found in the literatures.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The first approach is called "context-based approach". In context-based approach, the researcher identified twenty-four context of change (6). All changes subjected to software are analyzed to map its context and evolution technique suitable to address the change. The second approach is called "semantic-changed" approach. In this approach, a technique is proposed to enable precise recording of semantic change of software during evolution. Based on the precise semantic information gathered, it is argued that a better analysis can be done to control software evolution (7). The third approach is called "story-driven approach". The aim of story-driven approach is to provide methods to enable explicit and implicit knowledge about the software during its lifecycle to be formally represented (8). The knowledge repository will provide useful information for software maintainers to perform evolution activities in a more controlled manner. The fourth approach, "requirement engineering approach" deals with improving requirement gathering process in order to properly control software evolution. In (9), Ferreira et al propose a framework based on organizational semiotic to enable requirements change to be predicted in advance. Souza et al proposed the study of requirements evolution to enable the development of methods to address the negative impact of software evolution (10). The fifth approach is "software adaptation approach". The aim of software adaptation approach is to provide adaptation methods so that software can adjust its behavior to suit the changes subjected to it. There are a wide range of research works addressing software adaptation topics. Example of such works can be found in (11-15)

This research is adopting software adaptation approach in finding a novel approach for controlling negative effects of software evolution in the domain of enterprise system. An adaptation framework called MiPAF, which uses the concept of middleware and policy is proposed to enable software adaptation, hence simplifying software evolution process. The organization of this paper is as follows: Section 2 will give background information and related work with respect to this research. In Section 3, MiPAF is described in detail. Section 4 will describe the implementation of MiPAF on an enterprise system. The last section will conclude this paper and future works will be briefly explained.

2. RELATED WORK

Researchers in software engineering field have proposed various approaches to software adaptation in recent years. The research works range from the effort to develop generic architecture framework to specific technique in particular domain. McKinley mentioned that software adaptation can be categorized into two; namely compositional adaptation and parameterized adaptation (16). Compositional adaptation enable software to be reconfigured during runtime, whereas parameterize adaptation involves reconfiguration of software before execution. In our earlier work (17), based on exhaustive literature review, we categorized the approaches to software adaptation into four categories; architecture-based approach, agent-oriented approach, component-based approach and middleware-based approach. The following subsection described each approach briefly.

2.1 Approaches to Software Adaptation

As mentioned earlier, there are four type of approaches to software adaptation. Each approach is discussed separately in the following subsections.

2.1.1 Architecture-based approach

In architecture-based approach, architecture model is used to enable dynamic adaptation of software at runtime. Composition of components and their relationship are reconfigured during execution time based on the changes that occurs to the software. Technique in implementing architecture-based adaptation includes using a formal specification language as in Darwin (18). Other approach includes the use of Aspect Oriented Modelling. In architecture-based approach, more focus is given at the design phase and no specification given on how adaptable software is going to be constructed. This situation may result in the loss of architecture knowledge and software construction may become more complicated, thus maintenance will become difficult.

2.1.2 Component-based approach

In component-based approach, software components are treated as a set of black-boxes that is reusable. The focus of component-based approach is during the construction of the software. During the runtime of the software, when changes happen, different variance of the components will be executed. Without proper component management, communication between newly loaded variants and the rest of the component may pose problems. In most implementation, existing component frameworks are used to manage the components. Popular frameworks include CORBA,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EJB, COM and DCOM. These frameworks are language specific and in the case of COM and DCOM, they are proprietary frameworks. Example of works that uses component-based approach are (13, 19).

2.1.3 Agent-based approach

In agent-based approach, software agents are developed to enable self-adaptation. Related works for agent-based approach can be found in (20, 21). Software agents provide external adaptation mechanism and communication between agents are using asynchronous messaging. There are two major drawbacks in using agent-based approach. The first drawback is that since agent-based approach uses higher abstraction level than object-oriented approach, novice developers may find it hard to use. The second drawback is that since asynchronous messaging is used as communication method, communication latency may affect software performance in a large software implementation.

2.1.4 Middleware-based approach

In middleware approach, software implementation is segregated into different layers thus changes to one layer will not affect another layers. It was argued that traditional middleware has limited capabilities in supporting the needs of software adaptation. However, traditional middleware concept can be enhanced to enable software adaptation. One of benefits in middleware-based approach is that no implementation technique is specified. Various technique can be incorporated in middleware approach including web services, aspect oriented programming and component based programming. Middleware-based approach offers scalability and heterogeneity which are very important for an enterprise system.

Comparative evaluation was performed on the approaches based on six defined criteria. The criteria are scalability, context-awareness, performance, usability, heterogeneity, and dynamic-evolveability. The result of comparative evaluation has shown that middleware-based approach scored more points as to compare with three other approaches.

3 THE PROPOSED FRAMEWORK

The framework proposed in this paper is called MiPAF (**Mi**ddleware and **P**olicy-based **A**daptation Framework). MiPAF main aim is to enable the compositional and parameterized software adaptation in order to simplify software evolution. MiPAF is developed based on a number of concepts namely middleware-based approach, close loop feedback system, web services and policy.

3.1 MiPAF Building Block

MiPAF comprises of six main components namely Service Manager, Adaptation Manager, Service Infrastructure, Policy Repository, Context Monitor and Device Controller. MiPAF building block is specified using class diagram in the following figure.

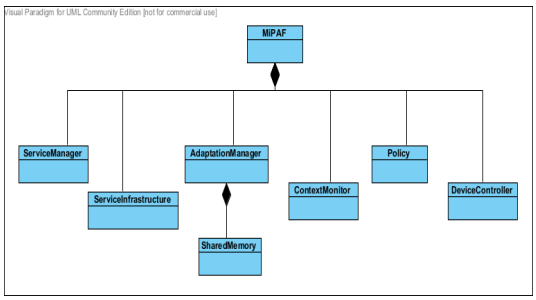

Figure 1: MiPAF Class Diagram

In the following subsections, MiPAF components are described in more detailed. Formal specification for each components are described using Z Notation. Z Notation is used since Z is already a matured language and sufficiently expressive to describe MiPAF.

3.2 Service Manager

Service Manager is the first point of contact between MiPAF and enterprise application that uses MiPAF runtime. In designing a service manager, five considerations need to be highlighted:-

a. Communication protocol between Service Manager and enterprise applications

b. Common message format for request and reply

c. Ability to handle multiple requests from enterprise systems

d. Internal communication with Adaptation Manager

e. Ability to parse XML message

The high level specification for Service Manager is as follows:-

ServiceManager ≙ Listener | DataHandler

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

There are two main components of Service Manager, namely Listener and DataHandler. The role of Listener is to accept request from enterprise applications and pass the data to the DataHandler for processing the XML messages. The reformatted data will be sent to Adaptation Manager.

3.3 Adaptation Manager

Adaptation Manager is the component of MiPAF that need to make adaptation decisions whenever changes occur. Prior to making adaptation decisions, Adaptation Manager must have a mechanism to detect changes that will impact an enterprise software. Adaptation Manager is the central component of MiPAF since it requires direct communication with Service Manager, Service Infrastructure, Device Controller and Policy Repository.

Adaptation Manager can be formally described in Z Notation as follows:-

AdaptationManager ≙ Initialize | PolicyHandler | AdaptationEngine | InterfaceToDevController | InterfaceToSvcManager

From the above specification, Adaptation Manager comprises of five components; Initialize, PolicyHandler, AdaptationEngine, InterfaceToDevController and InterfaceToSvcManager. Each component has its own responsibilities to enable Adaptation Manager to perform adaptation requirements. Initialize is an internal components to initialize all data used internally by Adaptation Manager. PolicyHandler has the role to load the right policy that will dictate the adaptation process. AdaptationEngine is the component that will translate the loaded policy into adaptation actions. Specific adaptation algorithm is developed for AdaptationEngine. InterfaceToDevController will handle all interactions with implemented Device Controller. InterfaceToSvcManager receives data from either Service Manager or Service Infrastructure.

3.4 Service Infrastructure

Service Infrastructure provides web server's capability to MiPAF since MiPAF supports web service communication. Web-based enterprise applications will communicate to MiPAF via Service Infrastructure. First level formal specification for Service Infrastructure is as follows:-

ServiceInfrastructure ≙ Listener | HTTPDataHandler

ServiceInfrastructure will continuously listen to the requests from enterprise application and process the HTTP command and strip the SOAP envelop from the incoming request. The data will be sent to Adaptation Manager.

3.5 Policy Repository

Policy Repository is where application policy is stored. MiPAF does not specify how the application policy to be stored. However, MiPAF specify that an application policy must be prepared in XML using a set of pre-defined keywords. The specification for Policy Repository is as follows:-

PolicyRepository ≙ PolicyName | RegisterPolicy

All policies must have a unique name and registered with Policy Repository.

3.6 Device Controller

Device Controller abstracts out the technical complexity of device communication. The problem that Device Controller is addressing is lack of standard interface on how to communicate with devices. Device Controller allows devices to be access using standard APIs. The specification for Device Controller is as follows:-

DeviceController ≙ SendReceiveAM | InterfaceToCM | InterfaceToDev

Device Controller communicates with Adaptation Manager via SendReceiveAM function. It will update the device's status to Context Monitor and it will issue device specific command in order to find out the status of the device.

3.7 Context Monitor

Context Monitor has the role to monitor devices of interest to the enterprise system. Context Monitor will alert Adaptation Manager when a device cease to functions. When there is a problem with a device, there are two possibilities; whether the actual device is problematic or the Device Controller has stopped. Context Monitor has the capability to restart the Device Controller. Specification for Device Controller is as follows:-

ContextMonitor ≙ MonitorResource | RegisterResource | UpdateStatus | RestartResource

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MiPAF is meant to be used by both enterprise application in maintenance mode and newly developed application. For a newly developed application, there are a number of processes that is specified during Requirement. Further details can be found in the following subsections.

3.8 MiPAF Process

Using MiPAF during new application development involves some new processes to be included in requirement, analysis, design, and development phase.

3.8.1 Requirement phase

During requirement gathering activity, apart from gathering functional and non-functional requirements, adaptation requirements must also be captured. Analyst should have a set of adaptation requirement questions in order to elicit adaptation requirement.

3.8.2 Analysis phase

During the Analysis Phase, two important process must be carried out. The processes are analysis of adaptation requirements and analyzing architectural needs. In this phase, the adaptation requirements will be analysed and the conceptual architecture that will support the required adaptation requirements will be decided.

3.8.3 Design phase

During the Design Phase, apart from traditional design process, two added activies are required, i.e. design of Device Controllers and the design of adaptation policy. The work product for this phase is Device Controller design and policy design which will be incorporated into Software Design Document.

3.8.4 Development phase

During the Development phase, the Device Controller for each device will be developed. In the development of Device Controller, Application Programming Interface (API) from device vendors will be encapsulated in a standard API specified by MiPAF. The adaptation policy can be developed using MiPAF Policy Editor. In the following subsection, MiPAF Policy Language (MPL) will be discussed.

3.9 MiPAF Policy Language

MPL is used to drive the adaptation mechanism in MiPAF. MPL uses action-based rules in making adaptation decision when changes occur. Adaptation rules will provide instructions to be followed by Adaptation Manager in response to each change. One of the benefits of policy based adaptation is the ability to totally decouple adaptation mechanism from the component that control the adaptation behavior. Therefore, changes in adaptation policy will not affect the Adaptation Manager and other components of MiPAF.

There is a number of existing policy language developed other researches such as PONDER and REI. However, both PONDER and REI are biased towards security. Other multipurpose policy language such as Esterel and Jess require developers to learn new languages. MiPAF requires less complex policy language, implementation independent but should be able to expand when required. Therefore, a simpler but extendable policy language, MPL, was developed. XML is used for MPL since it is expandable and most developers are familiar with XML.

The ontology for MPL is described in the following section.

3.9.1 MPL ontology

MPL Ontology is depicted in Figure 2.

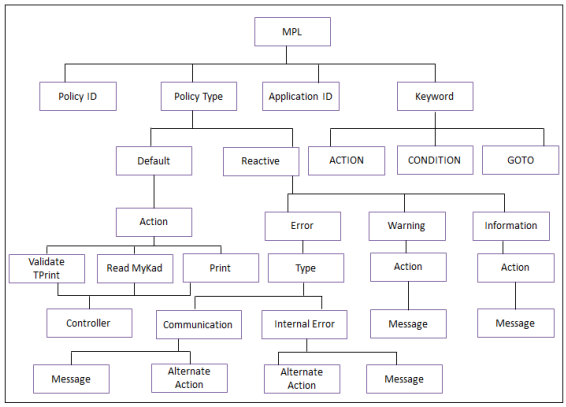

Figure 2: MPL Ontology

Each adaptation policy file will have Policy ID, Policy Type, Application ID, and Keyword. Policy ID is a unique identification of a policy file. Policy can be of two types, Default and Reactive. Each application need both type pf policy. Default policy is the normal flow of an application when using devices. Reactive policy will specify adaptation required in the event of change. Reactive policy comprises three categories namely error, warning and information. For each category, a message and alternate action must be specified. Message is an information to be communicate to users in the event of change and alternate action is the adaptation behavior that need to be executed in response to the changes.

Journal of Theoretical and Applied Information Technology
31st December 2014. Vol.70 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Currently, three keywords are supported by MPL namely ACTION, CONDITION and GOTO. Since MPL is XML based, an existing XML Parser, expat, is used to parse the policy file. Expat XML parser allows "handler" to be registered as ACTION whenever the CONDITION is triggered. GOTO keyword allows more flexibility to be incorporated in the policy file.

4. IMPLEMENTATION OF FRAMEWORK ON ENTERPRISE SYSTEM

4.1 Overview of Unit Trust Enterprise System

The implementation of MiPAF is done on existing enterprise system called Unit Trust Enterprise System (UTES). UTES is a client-server enterprise system which is currently implemented in Malaysia. UTES consists of two layers of application; front-end application and backend application. MiPAF implementation concerns only the front-end part of UTES since the scope is to manage changes at the device layer of an enterprise system. In the context of UTES, device layer is located at the front-end application. UTES front-end is a counter-based system used by tellers.

The purpose of UTES is to manage transactions and operations of unit trust transactions for an organization. UTES has been implemented for many years. Over the years, changes are introduced to its device layer due to changes in business requirements and to improve operational efficiency. For the purpose of this research, the main focus is only on one device used by UTES, passbook printer. Passbook printers are specialized printer used to print investors' passbook after transaction such as additional investment transaction or withdrawal transactions. The objective of MiPAF implementation in UTES is to control the impact of change due to evolution that occurs at the device layer of UTES. For a better perspective, the relationship between MiPAF and UTES can be found in the following diagram.

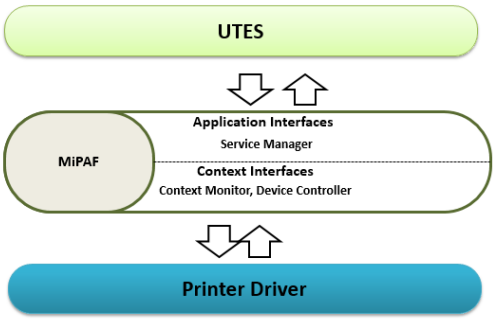

Figure 2: MiPAF Context Diagram

MiPAF sits in between UTES and the printer driver used by the passbook printer. Interface to UTES is via Service Manager and Interface to the printer driver is via Device Controller. Context Monitor will detect changes occurs at device layer.

4.2 Overview of the Implementation Environment

Testing environment for implementation of UTS using MiPAF is depicted as follows:-

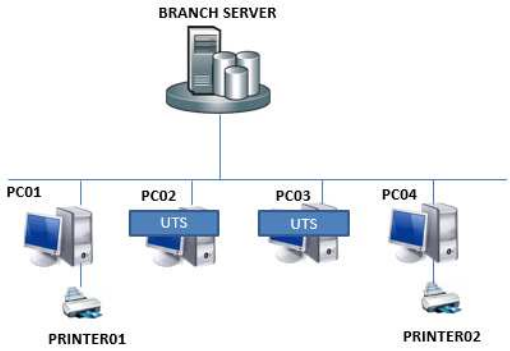

Figure 3: Implementation Environment

Four workstations are used for this purpose, PC01 to PC04. Printers are attached to PC01 and PC04. PC02 will share printer with PC01 and PC03 will share printer with PC04. MiPAF runtime environment is installed on PC01 and PC04. Adaptation requirements will be described based on the above configuration. For the test of scalability, initially, the evaluation was conducted on single workstation.

4.3 Adaptation Requirements

There are a number of possible change that can happen to the configuration specified in subsection 4.2. These changes bring about the adaptation requirements for UTES. The adaptation requirements for UTES are as follows:-

Requirement 1: Device Failure

During daily operation of UTES, there is a probability of printer failure due to mechanical problems. In the case of printer failure, UTES front-end should be able to perform printing using other printer within the same location.

Requirement 2: Change of Device Type

It is quite common for an organization to change the type or brand of printer. Device driver for each passbook printer is different. Therefore, changing printer type require change of application.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This type of change should not be affecting application system.

These two requirements will be simulated for the implementation using implementation environment specified in Figure 3.

4.4 Design of Adaptation Policy

To design the adaptation policy, adaptation requirements and rules need to be understood. The rules for adaptation for this implementation is as follows:-

- IF PRINTER01 fail, PC02 should be able to print using PC04 without the need to introduce any change and without recompilation of UTES required.
- Changes of PRINTER02 with different type of printer should not affect operations of UTES. For example, PRINTER02 will be changed from IBM9068 to TALLY GENICOM printer.

The default policy will specify the basic flow printing for UTSE. Part of the default policy will look as in the following:-

```
<?xml version="1.0" encoding="utf-8" ?>
<POLICY>
  <APPLICATION_ID> UTS Front-end</APPLICATION_ID>
  <POLICY_ID>1</POLICY_ID>

  <DEFAULT>

    <ACTION>
      <ACTION id="PRINT" dev_name="PRT_9068"/>
        <CONTROLLER>
          <CONTROLLER id="Printer Server"
                      ip_addr ="172.19.37.108"
                      port="1052" />
        </CONTROLLER>
    </ACTION>
```

In the default flow, dev_name is PRT_9068 which specify the printer type and the ip_address of the printer for basic flow is used. The reactive policy will specify conditions for adaptation and adaptation actions that will be executed by the Adaptation Manager. MPL Policy Editor is provided to generate MPL for ease of writing adaptation policy.

4.5 MiPAF Implementation

Each MiPAF components described in Section 3 need to be implemented to create a runtime environment so that it can be used by UTES. All MiPAF runtime components are developed using C programming language. Brief description on the runtime components of MiPAF is as follows.

4.5.1 Service manager

Since UTES is a client-server based enterprise application, Service Manager is implemented instead of Service Infrastructure. 2 components are implemented namely Listener and Data Handler. Listener is implemented using a common socket-based programming. Listener will continuously waiting for request from UTES and upon receiving the request, it will execute bind and accept command. During this time, UTES is in a waiting state. Data Handler will process the XML formatted request and pass the data to the Adaptation manager.

4.5.2 Adaptation manager

Adaptation Manager execute adaptation behavior for UTES. The algorithm for Adaptation Manager is lengthy to be included here. In summary, the Adaptation Engine is implemented using multithreaded and event based program. It uses shared memory for internal communication with the Context Manager. Adaptation Manager will execute Default Policy first and at specific interval, it will check a shared memory space that store printer status. If status change is detected, Adaptation Manager will execute Reactive Policy.

4.5.3 Policy repository

MiPAF Policy Repository is implemented using shared memory concept. A shared memory space is defined and can be accessed by all components. The structure the shared memory is as follows:-

```
char strApplicationName[20]
int strPolicyID
char* ptrStrData;
```

4.5.4 Device controller

For this implementation, the Device Controller is implemented using existing printer server program that is used by UTES. The printer server program is triggered by Adaptation Manager using event. Upon being triggered, it will get the data to be printed and send the data to the printer. It will return the status of printing to the Adaptation Manager.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.5.5 Context monitor

For this implementation, Context Monitor is implemented using an existing program called "Health Checker". Health Checker is a socket based program that continuously send connection request to each Device Controller. .Health Checker will update the status of each Device Controller in a shared memory. This status will be read by Adaptation manager at specific intervals.

5. CONCLUSION AND FUTURE WORKS

The evaluation of MiPAF using UTES was carried out based on the six evaluation criteria mentioned earlier, i.e. scalability, context-awareness, performance, usability, heterogeneity, and dynamic-evolveability. MiPAF meet all the criteria mentioned. In terms of scalability, it support both downward and upward capability. This is evidence when evaluation was done using one workstation and 4 workstations subsequently. Implementation of Context Monitor ensure context awareness criteria is met. Since MiPAF is installed on each workstation, there is no issue of performance. MiPAF is easy to be used since it uses open standard such as TCP-IP and XML. The adaptation policy is very simple and policy editor is provided. MiPAF is heterogeneous as it does not specify the language or platform for implementation. It meet dynamic-evolveability by means of using policy to specify adaptation that can be executed at runtime.

There are a number of future works that can be spin-off from this research:-

- The focus of this research is only on the change of devices in an enterprise system. Other changes that can be extended from this research is in terms of change in the database type and change in communication protocol.
- This research uses two type of communication to cater for different architecture type. Client-server applications use socket-based communication while web-based enterprise applications use web services to access the device. For future works, it is better to standardize the application interface so that changes in application architecture will not affect MiPAF execution
- There are other works in standardizing interface to devices such as works in Service Oriented Device Architecture (SODA). In future, when SODA is matured, Device Controller can be improved to incorporate SODA so that Device Controller can be standardized.

REFRENCES:

[1] Reiss SP, Editor. Evolving Evolution [Software Evolution]. Principles Of Software Evolution, Eighth International Workshop On; 2005

[2] Canfora G, Editor. Software Evolution In The Era Of Software Services. Software Evolution, 2004 Proceedings 7th International Workshop On Principles Of; 2004

[3] Lehman MM. Laws of Software Evolution Revisited. Proceedings of the 5th European Workshop on Software Process Technology: Springer-Verlag; 1996. p. 108-24

[4] Lehman MM, Ramil JF. An Approach to a Theory of Software Evolution. Proceedings of the 4th International Workshop on Principles of Software Evolution; Vienna, Austria: ACM; 2001

[5] Madhavji NH, Ramil JF, Perry DE. Software Evolution and Feedback: Theory and Practice: John wiley and Sons; 2006

[6] Ciraci S, van den Broek P, Aksit M. A Taxonomy for a Constructive Approach to Software Evolution. Journal of Software. 2007;2(2):84-97.

[7].Robbes R, Lanza M, Lungu M. An Approach to Software Evolution Based on Semantic Change. Fundamental Approaches to Software Engineering: Springer Berlin / Heidelberg; 2007. p. 27-41

[8].Riling J, Meng WJ, Witte R, Charland P. Story-driven Approach to Software Evolution. Software, IET. 2008;2(4):304 - 20.

[9].Ferreira MG, do Prado Leite JCS. Requirements Engineering with a Perspective of Software Evolution. 2011.

[10].Souza VS, Lapouchnian A, Angelopoulos K, Mylopoulos J. Requirements-driven software evolution. Comput Sci Res Dev. 2013;28(4):311-29.

[11].Zhang H, Ben K, Zhang Z, editors. A Reflective Architecture-Aware Framework to Support Software Evolution. Young Computer Scientists,2008 ICYCS 2008 The 9th International Conference for; 2008.

[12].Oreizy P, Medvidovic, N., Taylor, R.N., Gorlick, M.M., Heimbigner, D., Johnson, G., Quilici, A., Rosenblum, D.S., Wolf, A.L. An

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Architecture Based Approach to Self-Adaptive Software. IEEE Intelligent System. 1999:54 - 62.

[13].Holger K, Dirk N, Andreas R. 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; 2007.

[14].Garlan D, Cheng WC, Huang AC, Schmerl B, Steenkiste P. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. IEEE Computer Society. 2004:46 - 54.

[15].Lundesgaard SA, Arnor S, Oldevik J, France R, Aagedal JO, Eliassen F. Constriction and Execution of Adaptable Applications Using an aspect-Oriented and Model Driven Approach. Lecture Notes in Computer Science: Springer Berlin/Heidelberg; 2007. p. 76-89.

[16].McKinley PK, Sadjadi SM, Kasten EP, Cheng BHC. Composing adaptive software. Computer. 2004;37(7):56-64.

[17].Awang NH, Wan Kadir WMN, Shahibuddin S, editors. Comparative Evaluation of the State-of-the Art on Approaches to Software Adaptation. Fourth International Conference on Software Engineering Advances; 2009; Porto.

[18].Jeff M, Jeff K, editors. Dynamic structure in software architectures. Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering; 1996; San Francisco, California, United States: ACM.

[19].Kurt G, Mohammad Ullah K, Roland R, Arnor S, Svein H, Simon M. Modeling of component-based adaptive distributed applications. Proceedings of the 2006 ACM symposium on Applied computing; Dijon, France: ACM; 2006.

[20].Seungwok H, Sung Keun S, Hee Yong Y, editors. Dynamic Software Adaptation with Dependence Analysis for Multi-Agent Platform. Computational Science and its Applications, 2007 ICCSA 2007 International Conference on; 2007.

[21.]Qureshi NA, Perini A, editors. An Agent-Based Middleware for Adaptive Systems. Quality Software, 2008 QSIC '08 The Eighth International Conference on; 2008.