# Table of Contents

1. **THE DESIGN OF TEXT TO USE CASE DIAGRAM TOOL** ........................................... T2-1  
   *Siti Hafizah Ab Hamid, Wan Hasnira Wan Husin, Mohd Hairul Nizam Md Nasir*

2. **EFFORT REDUCTION OF QUALITY ASSURANCE FOR COMPLEX SYSTEMS THROUGH FAULT ESTIMATION** ................................................................. T2-8  
   *Matthias Wiemann, Jörg Gericke*

3. **INTEGRATING QUALITY ASSURANCE TECHNIQUES WITH REQUIREMENTS NEGOTIATION ACTIVITIES: AN ANALYSIS OF SOFTWARE DEVELOPERS PRACTICES** ........................................................................................................ T2-15  
   *Hazrina Sofian, Siti Salwah Salim*

4. **MODEL Driven ARCHITECTURE FOR INTEGRATED ENTERPRISE APPLICATIONS – USING MOF** ................................................................. T2-22  
   *Subaji Mohan, Eunmi Choi*

5. **AN OBJECT ASSERTION LANGUAGE FOR INTEGRITY CONSTRAINTS IN OODBS** ................................................................................................. T2-29  
   *Belal Zaqiibeh, Hamidah Ibrahim*

6. **PERISA: A PERSONAL ISLAMIC ASSET MANAGEMENT SYSTEM USING OBJECT-ORIENTED APPROACH** ................................................................. T2-37  
   *Nor Fazlina Iryani Abdul Hamid, Zarinah Mohd Kasirun*

7. **REALISING THE MYCL PROCESS MODEL THROUGH THE DEVELOPMENT OF A COMPONENT-ORIENTED SPECIFICATION LANGUAGE (COSL) COMPILER** ................................................................. T2-43  
   *Hazleen Aris, Siti Salwah Salim*

8. **SOFTWARE VISUALIZATION TOOL TO SUPPORT FOR OUTSOURCED SOFTWARE MAINTENANCE** ......................................................................... T2-50  
   *Eswaran Narasimhan, Udayan Banerjee*

9. **TOWARDS A HIGHLY EXTENSIBLE AND FLEXIBLE MIDDLEWARE FOR DATA MINING** ................................................................................................. T2-57  
   *Lai Ee Hen, Sai Peck Lee*

10. **TOWARD A METHODOLOGICAL FRAMEWORK TO SUPPORT PROFESSIONAL VIRTUAL COMMUNITIES** ................................................................. T2-64  
    *Fadila Kaouane, Imed Boughzala*
11. WSEP: AN E-PROCUREMENT SYSTEM OF SUPPLY CHAIN MANAGEMENT USING WEB SERVICES TECHNOLOGY.................................................. T2-72
   Yap Khay Seng, Siti Hafizah Ab Hamid,
   Mohd Hairul Nizam Md Nasir
THE DESIGN OF TEXT TO USE CASE DIAGRAM TOOL

Siti Hafizah Ab Hamid, Wan Hasnira Wan Husin, Mohd Hairul Nizam Md Nasir
Faculty of Computer Science and Information Technology,
University of Malaya, 50603 Kuala Lumpur, Malaysia.
Email: sitihafizah@um.edu.my, aira2125@yahoo.com, hairulnizam@um.edu.my

ABSTRACT

Use case diagram is a diagram that helps requirements analysts to discover the requirements of target system from its users’ perspective. It describes what the system does from the standpoint of external observers. Use cases are employed during the analysis phase of a software project to identifying and partitioning the system’s functionality. There are many existing tools that assist end-user to draw Use Case Diagram. Event Studio allows the end-user to enter requirements by using FDL language to be generated as a diagram. Therefore, the end-user has to learn its syntax and semantics to use the tool. This paper presents a project’s design named Text to Use Case Diagram Tool or T-UCD tool that will help its end-user to identify actors, use cases, and relationships from a simple scenario or case study that will be generated as a Use Case Diagram. The system design covers both static and dynamic part of the system development which is Use Case Diagram, Class Diagram, and Interaction Overview Diagram. All of these diagrams illustrate on what and how the tool shall be implemented.

Keywords: Use Case Diagram, Text to Diagram

1.0 Introduction

Recently, the object-oriented methodology has been more popular and widely used. Its usage starts from the first level of software development lifecycle, known as the analysis level until the last level, which is called the maintenance level. One of the object-oriented methodologies is called Unified Modeling Language or UML [1], has now become a standard language in specifying object-oriented software.

The UML has its own notations and provides multiple views of a software system. It is a very useful modeling language in designing software. It acts as an interface between the natural language of user requirements and the programming language of implementation. It is also a communication medium between software designers and programmers. Under Object Management Group (OMG), UML has been released in many versions [4]. Even though OMG has produced a white paper on latest UML version which is version 2.0 in 2003, it seems that people still come out with different types of diagram when it comes to drawing a UML diagram for the same requirements (or case study). Hence, this is the main point that gives an idea for this research to develop a tool in order to help the users to draw a correct use case diagram.

Since the important part of UML is its facilities in drawing a use case diagram [3], this research is only focused on the use case diagram. In UML, the use case diagram plays an important role in the design level where it clearly visualizes all requirements that have been gathered. According to the UML specification (Object Management Group 2001), a use case diagram is a diagram that shows the relationship among actors and use cases within a system. A use case diagram is a diagram that helps analysts to discover the requirements of the target system from the user’s perspective. It describes what a system does from the standpoint of an external observer. Use cases are employed during the analysis phase of a software project to identify and partition system functionality.

The title of this project is named as “Development of a Text to Use Case Diagram Tool” or “T-UCD”. T-UCD is developed to help and guide students in identifying the elements of a use case diagram from a short case study and subsequently draw a use case diagram based on the case study. T-UCD which acts as a tutor is designed to guide the student, where he or she is only required to follow all the steps to drawing a correct use case diagram.

2.0 System Functional Requirements

There are five modules in the tool, which are notes, case study, elements, relationships, and diagram. The notes module is for the end-user reads notes/information about the UML diagram. The case study module is for the end user to key in the simple scenario/case study that he/she wants it to be converted to Use Case Diagram. Once the end user has entered the scenario, the tool will highlight all actors and use cases involved. The elements module allows the end user to modify the elements. The relations module is for the end user creates and modifies the relationship.
The diagram module is for the end user plays around with the diagram. Below is the Figure 1 illustrates a use case diagram of the system’s requirements.

**Notes Module:** In the Notes module, user will be able to select notes to view the notes. In this situation, “Select notes” use case is realized by using a class named Notes. Figure 2 illustrates the partial of the system Use Case Diagram (for Notes Package).

**Case Study Module:** In the Case Study module, users are allowed to key-in requirement, edit requirement and delete requirement. “Key-in requirement” use case allows the user to either “Key-in system name” or “Key-in case study”, while “Edit requirement” use case allows the user to either “Edit system name” or “Edit case study”. The “Delete requirement” use case allows the user to either “Delete system name” or “Delete case study”. Once the users key-in the system name, edit the system name and delete the system name, all of these tasks can be traced by Diagram class which is defined as Tutorial.vb. On the other hand, once the users key-in the case study, edit the case study and delete the case study, all these tasks can be traced by Case Study class which is also defined as Tutorial.vb. Figure 3 illustrates the partial of the system Use Case Diagram (for Case Study Package).

**Elements Module:** In this module, users are allowed to add or delete elements. “Add elements” use case allows the user to either “Add actor” or “Add use case”, while “Delete elements” use case allows the user to either “Delete actor” or “Delete use case”. All tasks are traced by Elements class then defined as Tutorial.vb. Data of actor is saved in Data Actor database while data of use case is saved in Data UseCase database. Figure 4 illustrates the partial of the system Use Case Diagram (for Element Package).

**Relations Module:** In this module, users are allowed to select elements, select element’s name, select relationship and edit relation. “Select element” use case allows the user to either “Select actor” or “Select use case” while “Select element’s name” use case allows the user to either “Select actor’s name” or “Select use case’s name”. The “Edit relation” use case allows the user to either “Add relation” and “Delete relation”. All of use cases are traced by Relations class which is defined as Tutorial.vb and these elements are subsequently saved in the Relationship database. Figure 5 shows the partial of the system Use Case Diagram (for Relations Package).

**Diagram Module:** In this module, users are able to adjust, save and print the diagram. “Save diagram” use case provides additional function that maybe required in the “Print diagram”. “Adjust diagram” and “Save diagram” are traced by Diagram class which is defined as Tutorial.vb. Figure 6 shows the partial of the system Use Case Diagram (for Diagram Package).
3.0 System Non Functional Requirements

Non-functional requirements are requirements that constrain the design of a system, but do not describe the service provided. Below are the non-functional requirements for T-UCD.

i. Simple – T-UCD is a system created using simple user interfaces. In this system, user will be able to master the function provided due to the available simple user interface. T-UCD has only five main pages and all pages provide the “Next” button for the user to proceed to the next task.

ii. User friendly – As the system is built under the Windows environments, the interface should be friendly and easy to understand. It should also be intuitive and consistent in purpose and use within themselves. Generally, the design of T-UCD interfaces is as follows:

- Consistent, in terms of screen design and error message display.
- Accommodating for any level of end user.
- Appropriate error handling with associated error messages.
- High degree of understandability and avoids too much memorization of events and commands for the users.
- Provides help assistance. Screen maintains help assistance for user who needs online help. This does not mean that the screen is no more user friendly, but simply made available to provide more features to user.
iii. Interactive – Interactive refers to programs or applications that respond directly to the end user, taking instructions and giving feedback. The end users are allowed to navigate to the previous and next task easily by clicking the provided buttons. Help menu is available and can be accessed anytime.

iv. Usefulness – T-UCD is a useful system not only for students who are learning the UML subject, but also for system designers and system analysts. People who are involved in developing systems can also use T-UCD to obtain a graphically requirements before the system development begins.

v. Clarity – Use case diagram provided in T-UCD is clear for the users to view and save.

Fig. 7: Class Diagram

4.0 System Class Diagram

A class diagram is defined as a diagram that shows a collection of classes and interfaces along with the collaborations and relationships among classes and interfaces. A class diagram is a pictorial representation of the detailed system design also known as a static view of the system [6]. Figure 7 illustrates the class diagram for T-UCD.

5.0 System Interaction Overview Diagram

Interaction overview diagram is used to illustrate a control flow serving an encompassing purpose. Figure 8 illustrates the interaction overview diagram for T-UCD. In reference to the diagram, upon entering the T-UCD, a user may choose to either proceed to the Notes module or the Tutorial module. In the Notes module, the user may view all the provided notes. After that, they may proceed to the Tutorial module which includes four submodules known as Case study, Elements, Relations, and Diagram modules. Once a user enters the Tutorial module, they must key-in the requirements; system name (will appear on the “Diagram” page) and a case study in “Case Study” page. T-UCD will highlight actors and use cases which are identified from the case study. In the Elements module, all elements identified from case study will be listed. User may also add or delete elements. In addition, the user may progress to the Relations module. This process involves Data Actor and Data UseCase databases. In the Relations module, the user must select a few
parameters such as the related element, element’s name and relationship for the use case diagram. In this module, the user may also add or delete a relation which is involved of Relationship database. Finally, the user may view the use case diagram that is generated by T-UCD in the Diagram module.

![Diagram](image)

**Fig. 8: Interaction Overview Diagram**

### 6.0 How to Identify Actors and Use Cases from a Text?

In order to answer above question, the code of the system using Visual Basic is shown below.

```vbnet
Private Sub RTB_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles RTB.TextChanged
    ScanRTB() 'Call the Function
    ScanRTB1() 'Call the Function
    End Sub
Public Sub ScanRTB() 'blue
    Dim i
    Dim sArray
    sArray = Split(wArray, ",", -1, 1) 'splits all the words into an array to check if the word needs to be changed
    For i = 0 To UBound(sArray) 'loop through the words in the array
        SendMessage(RTB.Handle, WM_SETREDRAW, 0, 0) 'reduces flicker
        Dim sLength As Integer
        Dim sStart As Integer = Len(RTB.Text) - Len(sArray(i)) 'gets the start length of the selection
        sLength = Len(sArray(i)) 'gets the length of the word you are searching
        If sStart < 0 Then GoTo Continue 'if length of text is less than length of word, bypass next code or it will error out
        RTB.Select(sStart, Len(sArray(i))) 'selects the previous characters based on the length of the word to search
        If UCase(RTB.SelectedText) = sArray(i) Then
            RTB.SelectionColor = Color.Blue ' if the upper case matches, change the font
            checkstring("blue", RTB.SelectedText)
        End If
    Continue:
    RTB.Select(Len(RTB.Text) + 1, 0) 'this positions the cursor back to where you are typing.
    RTB.SelectionColor = Color.Black 'sets the font color back to black
    SendMessage(RTB.Handle, WM_SETREDRAW, 0, 0) 'more code on the flicker
    If UCase(RTB.SelectedText) = sArray(i) Then
        RTB.Select(Len(RTB.Text) + 1, 0) 'this positions the cursor back to where you are typing.
        RTB.SelectionColor = Color.Black 'sets the font color back to black
        SendMessage(RTB.Handle, WM_SETREDRAW, 0, 0) 'more code on the flicker
    End If
Next
End Sub
Public Sub ScanRTB1() 'red
    Dim i
```

©FCSIT, UM 2007
Dim tArray

tArray = Split(yArray, ",", -1, 1) 'splits all the words into an array to check if the word needs to be changed
For i = 0 To UBound(tArray) 'loop through the words in the array
SendMessage(RTB.Handle, WM_SETREDRAW, 0, 0) 'reduces flicker
Dim sLength As Integer
Dim sStart As Integer = Len(RTB.Text) - Len(tArray(i)) 'gets the start length of the selection
sLength = Len(tArray(i)) 'gets the length of the word you are searching
If sStart < 0 Then GoTo Continue 'if length of text is less than length of word, bypass next code or it will error out
RTB.Select(sStart, Len(tArray(i))) 'selects the previous characters based on the length of the word to search
If UCase(RTB.SelectedText) = tArray(i) Then
RTB.SelectionColor = Color.Red 'if the upper case matches, change the font
checkstring("red", RTB.SelectedText)
End If

Continue:
RTB.Select(Len(RTB.Text) + 1, 0) 'this positions the cursor back to where you are typing.
RTB.SelectionColor = Color.Black 'sets the font color back to black
SendMessage(RTB.Handle, WM_SETREDRAW, 1, 0) 'more code on the flicker
RTB.Refresh() 'refreshes the rtb to allow the change
Next
End Sub

7.0 Acknowledgment

This work was previously supervised by Nazean Jomhari who is currently doing PhD in Human Computer Interaction in University of Manchester, UK.

8.0 Conclusion

T-UCD is a tool that assists end user to draw a use case diagram based on a simple scenario or case study by identifying its actor, use case, and relationship. In general, any end user has problems in identifying all the three elements of a use case diagram. Therefore, the main purpose of developing this tool is not only to help the end-user but also to make them learn on drawing a use case diagram. There are few designs have been presented to visualize the tool development, which are Use Case Diagram, Entity Relationship Diagram, Data Flow Diagram, Class Diagram, and Interaction Overview Diagram.

REFERENCES


BIOGRAPHY

Siti Hafizah Ab Hamid is a lecturer at Department of Software Engineering, Universiti Malaya. Her research areas include Electronic Learning, Mobile Learning, Applied Object-Oriented Methodology, Applied Electronic and Mobile Technologies, Learning Objects, Software Metrics, Applied Game Techniques.
Wan Hasnira Wan Husin obtained her Master of Software Engineering from Universiti Malaya in 2007. She is interested in doing research in software design and CASE tool.

Mohd Hairul Nizam Md. Nasir is currently serving as a lecturer in the Department of Software Engineering, Universiti Malaya. His area of specialization includes web-based application development, software quality, and software process improvement as well as CASE tool.
EFFORT REDUCTION OF QUALITY ASSURANCE FOR COMPLEX SYSTEMS THROUGH FAULT ESTIMATION

Matthias Wiemann, Jörg Gericke
Siemens AG, Corporate Research and Technologies, CT PP 6
Otto-Hahn-Ring 6, 81739 Munich, Germany. Email: joerg.gericke@siemens.com

ABSTRACT
The structural complexity of hybrid systems increases constantly and thus leads to an increasing number of faults. Hence, the proportions of project costs that are due to quality assurance grow accordingly. Methods to reduce these costs have to be developed. In this paper, encouraging experiences of deducing and applying fault estimation models in industrial projects are presented. These models support the identification of extremely complex and fault-prone systems and their components. Thus, quality assurance arrangements like testing might focus on critical components, which is supposed to improve the quality of systems and their components more effectively and should lead to notable cost-cuttings. Although we focus on object-oriented systems, the presented estimation method is conceptual and therefore applicable to arbitrary target factors to deduce models for arbitrary estimations.

Keywords: Complexity Measures, Fault Estimation, Software Metrics, Quality Assurance

1.0 INTRODUCTION
Today, the structural complexity of software is constantly growing and leads to an increase of the number of faults in software systems. This trend delays the delivery of software systems within agreed time and causes increased efforts for finding faults. Thus, the proportion of the total development costs of software products that are due to quality assurance grows constantly. New methods and techniques are to be developed to act contrary to this trend. This is supposed to enable both: a structured software design and a direct support for quality assurance.

The focus of this work lies on estimating the number of faults in object-oriented software products, which may serve as instances for complex systems in general. Mathematical models will be deduced from measurement results and processed fault data from previous projects. These models may be used to localize faults and also to estimate the number of faults of particularly complex and fault-prone parts of software products. These concepts include software metrics, complexity analyses and regression procedures. As a consequence, quality assurance arrangements might begin at and focus on critical parts of software systems in order to improve the quality or design of software more effectively.

We developed a supporting tool that computes various software metrics, and derives as well as applies fault estimation models. These steps have been fully automated for the most parts. The implemented tool is in use at Siemens Corporate Research and Technologies and has been successfully applied to several real-world projects.

2.0 APPROACH
The UML 2.0 activity diagram in Figure 1 depicts the described technique of target factor estimation using the example of estimating the number of faults in software products. The general and abstract technique can be applied at various points within a software project, for example, at the beginning of a project, when solely design documents are available. This enables fault estimation indeed at an early stage, however it is based on relatively few input data. Fault estimation can also be applied during implementation considering (parts of) source code. That way, more input data is available, which is supposed to improve quality and accuracy of the fault estimation. Nevertheless, it must be assessed whether fault estimation at this stage still can contribute to an effort reduction, since it might be applied too late to take effect.

2.1 Selection of Metrics
Fault estimation requires evaluating reference data (see Figure 1). For this purpose existing software products are compared using so-called software metrics. They should quantify all properties that support understanding the structure and execution of the considered software products. This information is comprised in so-called cognitive complexity aspects. They might indicate increased number of faults and increased complexity of development goals such as maintainability and changeability. The selection of appropriate metrics is a crucial step, since selected metrics must provide a holistic view of the system in order to derive suitable fault estimation models.
Unfortunately, a single metric can only quantify certain complexity aspects and thus does not consider all complexity aspects. For example, complexity of inheritance is usually influenced by depth and breadth of the inheritance tree, and the numbers of overridden, reused and newly added methods. Various metrics are required to consider all these aspects of inheritance complexity. Thus, a coverage of all complexity aspects of a software product can only be achieved using a set of various metrics. Selecting appropriate metrics is often challenging since measurement results of various metrics might overlap and correlate. For example, an increase of the metric “number of operations” due to adding a method to a class might also result in an increase of the metric “lines of code”. On the other hand, the inheritance metric “number of children” remains unaffected.

Our fault estimation model should focus on object-oriented software. Thus, a set of selected metrics has to cover the complexity aspects of object-oriented software. In 0 a set of complexity aspects for object-oriented software has been identified (Figures 2 and 3) using the Goal-Question-Metric-Method (GQM, 0). To cover these complexity aspects, we chose a widely accepted basic set of 18 metrics that has been proposed by Lionel Briand 0. Additionally, we added five metrics so that our set of 22 metrics provides a much better coverage of the complexity aspects shown in Figures 2 and 3. The selected metrics are described in more detail in section 4.1.

In addition to software metrics, target factors must be selected and obtained from previous projects (see Figure 1). Target factors are the characteristics of a system that are to be estimated by the estimation model – in our case type, number and location of faults. Generally, other target factors such as costs for developing a component could also be estimated. Target factors of previous projects also serve as input for the computation of an estimation model.

2.2 Principal Component Analysis (PCA)

An easy interpretation of measurement results requires that measurement results do not correlate. For example, it is difficult to compare a deep with a broad inheritance tree using the metrics depth and breadth of the tree. Therefore, the next step in deducing a fault estimation model uses these correlated metrics to compute uncorrelated virtual metrics, so-called complexity aspects (see Figure 1). Isolation of complexity aspects is achieved using a statistical method called Principal Component Analysis (PCA) 00. The PCA originates in statistics and belongs to the multi-variant procedures.
In contrast to the strongly related factor analysis, which is based on statistical models, the PCA is a numerical method. It was introduced in the 1930s by Harold Hotelling.

PCA summarizes partly independent dimensions of selected software metrics and generates complexity aspects that are independent from each other. The independent complexity aspects are also called Principal Components (PC) and are represented through independent linear combinations of metrics, for example $PC1 = 0.6 \times LOC + 1.12 \times NOP$. Thus, their interpretation has to consider the correlation with various other metrics. For example, PC1 might correlate with LOC and NOP. PCs can also be interpreted separately as virtual metrics for properties that are detectable only indirectly. For example, a PC that is strongly influenced by several inheritance metrics could represent the overall complexity of inheritance.

### 2.3 Multivariate Adaptive Regression Splines (MARS)

In the next step, an estimation model is computed on the basis of the PCs and the investigated target factors (see Figure 1). This step utilizes the relatively new regression technique **Multivariate Adaptive Regression Splines (MARS)**, which was published in 1991 by Jerome H. Friedman. In contrast to other linear regression techniques, MARS allows the approximation of arbitrary functions. This part is straight forward and therefore not discussed in detail.

### 2.4 Estimating Faults

An estimation model that was generated using the MARS technique can be used to estimate target factors for new software systems (see Figure 1). This estimation begins with measuring the new software system using the previously selected metrics (see Section 2.1). Then, PCs for the new system are computed using their representation as linear combinations of metrics (see Section 2.2). Finally, the computed values for the PCs are inserted into the estimation model which provides the estimated values of the target factors for the new systems.

For example, the fault value “5” for a certain class might imply the following interpretation: “Classes that have comparable complexity as the currently considered class have shown 5 faults in the past. Therefore, it is likely that the considered class comprises approximately 5 faults.”

Similarly to most statistical procedures, it must be assured that reference objects, in this case previous software products, are similar in system size, structure, complexity, team experience, team size and application domain in order to achieve results of high quality. For example, it does not seem reasonable to apply fault estimation to a project involving four persons using statistical reference data from projects that involved two hundred persons.

### 3.0 APPLICATION OF FAULT ESTIMATION

#### 3.1 Localization of Fault-Prone System Parts

The computation of PCs (see Section 2.2) can be performed for whole as well as for parts of a new system. Thus, it is possible to compute PCs on package-, class- or even method-level. That allows identification of extremely complex and fault-prone system parts (e.g. packages, classes or methods).

#### 3.2 Reducing Structural Complexity during Design

Fault estimation can be based on class diagrams and therefore applied in early project phases such as the design phase. If faults are estimated on class-level, then redesign of complex parts might be applied to avoid making faults in implementation (see Figure 4). After having identified presumably fault-prone system parts, the system might be...
Proceeding of the 2nd International Conference on Informatics, 2007

redesigned for simplification. Then, fault estimation is applied to the redesigned system in order to ensure that complexity has been reduced. Once again, complex parts might be simplified. The process may be applied iteratively (see Figure 5). That way, fault estimation may increase system’s quality by preventing faults that have not yet been made. This early fault prevention is very favorable since fault removal costs increase exponentially with the project phase.

3.3 Increasing Effectiveness of Tests

Usually, time for testing is too short so that testing activities must be focused 0. Since fault estimation can also be applied directly before testing, it might help identifying critical system parts such as classes or methods that should be tested more intensively. Thus, testing might start at the identified complex parts.

Unfortunately, fault estimation predicts the number but not the criticality of faults. That means fault estimation might assign one fault to class A and ten faults to class B. Nevertheless, the one fault in class A may be very critical and thus causes great damage. On the other hand, the ten faults of class B together might not even be observable in the system’s behavior since they are caught due to defensive programming. In this case it seems favorable to begin with removal of the single fault in class A. Estimation of the criticality of faults is an open issue and subject to further research. As a first step, we assume that all predicted faults have the same criticality.

Another issue in testing is the so-called stopping criterion 0. When using coverage criteria, stopping criteria are usually defined by ensuring certain coverage. For example, a 100% branch-coverage should be achieved. If such criteria are not applicable, then other stopping criteria must be defined. When knowing the number of faults, stopping criteria might be defined more easily. For example, 90% of all faults have to be found. Such criteria might be defined using fault estimation. In this case, the inaccuracy of fault estimation must be kept in mind, otherwise a criterion might be impossible to achieve. For example, a system contains 10,000 errors. Fault estimation predicts 12,000 errors. If the stopping criterion has been defined to remove 90% of the estimated faults, then 12,000×0.9=10,800 faults have to be found to stop testing. Due to the inaccuracy of fault estimation, this goal cannot be achieved since the system does not contain 10,800 faults.

4.0 CASE STUDY

4.1 Generic and Automated Test Environment (GATE)

The previously described algorithms have been implemented as part of our test suite GATE (Generic and Automated Test Environment). GATE comprises various test case generators and test case reduction algorithms for state machines, condition coverages, function tests and boundary value analyses. The goal was to add a component for estimating the number of faults for each class and method in both, early and late phases of projects.

We hoped that GATE supports the in Chapter 3 mentioned benefits: Early fault estimation is supposed to help finding faults as early as possible and thus to avoid more costly fault-removal activities in later phases of projects. Additionally, consequent faults might be avoided through redesign of presumably fault-prone parts. Fault estimation in later phases is used to increase accuracy of the estimation towards the end of implementation. Testing activities may focus on presumably fault-prone system components and yield the possibility to define an adequate stopping criterion.
for testing. Usually, a stopping criterion is defined as finding a certain proportion of the estimated faults, for example, “85% of the estimated faults have to be identified”.

With these goals in mind, various metrics have been chosen that are applicable to both: early available design documents – such as class diagrams – and source code. An instance for such a metric might be the depth of the inheritance tree (DIT), which can be determined on the basis of design documents as well as source code.

For our implementation, we chose 22 metrics (see Section 2.1). These metrics focus on three major characteristics of software:

- quantitative metrics (6 metrics: lines of code (LOC), Halstead distinct operands (OPN), Halstead distinct operators (OPT), sum of attribute variable span (SAVS), maximum of attribute variable span (MAVS), number of public attributes (NOPA)),
- complexity metrics (3 metrics: cyclomatic complexity (CYC), sum of attribute access count (SAAC), maximum of attribute access count (MAAC)) and
- coupling metrics (13 metrics: ancestor method-method import coupling (AMMIC), other method-method import coupling (OMMIC), ancestor class-attribute import coupling (ACAIC), other class-attribute import coupling (OCAIC), ancestor class-method import coupling (ACMIC), other class-method import coupling (OCMIC), information flow based coupling (ICP), class to leaf depth (CLD), number of methods available (MTH), number of methods inherited (NMI), number of children (NOC), number of parents (NOP), depth of inheritance tree (DIT)).

As already mentioned, to derive an estimation model, the MARS algorithm requires data from previous projects to adapt to characteristics like application domain, project size or team’s experience. GATE’s fault estimation model should target projects in the automotive area, thus data from several projects of the same application domain (automotive software), team size and team’s experience has been evaluated. Thus, the resultant model is optimized for this particular type of projects.

4.2 Results of Case Studies

The computed fault estimation model was used in several real-world industrial projects to evaluate its applicability. Figure 6 shows the development of fault estimation during the life cycle of a project, which was typical for all surveyed projects. The shown project is relatively large with almost 10,000 identified faults over a time-period of 2.5 years. Here, we applied fault estimation independently from development. Thus, fault estimation did not influence design, implementation or testing and was solely applied to evaluate the quality of the fault estimation model.

Since the actual number of existent faults in the system remains unknown, we make the approximation that the number of faults that were found is equivalent to the number faults that actually existed in the system. Through evaluation of faults, we were able to identify the development phases, in which faults were made. Thus, it was possible to estimate the number of existent faults for each phase of the project (dotted line in Figure 6). The solid line in Figure 6 indicates the number of identified faults over time.

During the design phase, fault estimation was based on few design documents such as class diagrams. Since the data base has been small at that time, the observed deviation of the estimated number of faults compared to the existing number of faults was relatively high and averaged between 25% and 30% (see Figure 6). With time, more and more design documents became available and were taken into consideration. Thus, the estimation became more precise and showed an average deviation of 15% to 20% at the end of the design phase. Please keep in mind that we did not redesign the system in this example. Thus, simplification activities were not applied to reduce the number of following faults.

During implementation, the deviation was further reduced by evaluating an increasing amount of source code. At the end of the implementation phase, we estimated approximately 10,400 faults. Since the number of existent faults was approximately 9,800, the deviation of the predicted number of faults compared to the actually found number of faults was with 6% less than the 10%, which was the average over all surveyed projects. At this point, only approximately 2,000 faults had been identified. Fault estimation models from test managers predicted between 3,000 and 4,000 faults. A year later, GATE’s fault estimation has been used to define a stopping criterion for testing, after test managers’ fault estimation methods had converged GATE’s predicted number of faults.
5.0 CONCLUSION

The presented approach for fault estimation can be applied in early phases of projects. Early fault removal activities such as design reviews may help reducing the number of resulting faults. To be more effective in these reviews, the estimation also points out modules or parts of the system that seem to be especially fault-prone. Such an early fault removal is usually much less cost-intensive than later fault removal activities (fault localization, removal, regression testing). Literature describes dramatically increasing factors for fault removal costs for different project phases: (specification phase):(design phase):(implementation):(module test):(system test) $\approx$ (1):(3-6):(10):(10-40):(30-70) $0 \approx (1):(1):(1):(10-60):(n.a.):(20-120) 0 $\approx (1.5):(1):(10):(60):(n.a.) 0$.

The application of our GATE-tool has shown that fault estimation can be applied in early and late project phases. The accuracy of the estimation increases as more data is available. At the end of the implementation phase, the averaged deviation of the estimated number of faults from the existing number of faults was less than 10%.

5.1 Future Work

So far, our set of metrics to evaluate design documents considers class diagrams only. But the automotive industry also uses state charts and message sequence charts intensively. The next step will be to define and evaluate metrics for this kind of specification and design documents. This supports predicting the number of faults in early project phases before the actual implementation has started. Furthermore, state charts and message sequence charts are often also part of model-based specifications. Thus, a first fault estimation might already be applicable at the end of the specification phase.

Currently, we are evaluating great amounts of error reports to optimize our reference data which is crucial to derive fault estimation models. Future estimation results using the newly evaluated reference data should be compared as soon as the evaluation has been completed.

As already mentioned, the criticality of faults has not been evaluated yet. In our future work we plan to combine the number and type of faults with costs for fault removal in order to be able to estimate the criticality of faults.

REFERENCES

K. Bender, Embedded Systems-qualitätsorientierte Entwicklung. Springer. 2006. pages 1-10
M. Burghardt, Einführung in Projektmanagement. Publicis-MCD-Verlag. 1995

©FCSIT, UM 2007
D. Grell, Rad am Draht - Innovationslawine in der Autotechnik. in „c’t“, Ausgabe 14/2003
Gresse, C., Hoisl, B., Wust, J. A Process Model for GQM-Based Measurement. Software Technology Transfer Initiative (STTI), University of Kaiserslautern, Germany. 1995
H. Hotelling, Analysis of a Complex of Statistical Variables with Principal Components. Journal of Educational Psychology, 24:498–520, 1933
I. Jolliffe, Principal Component Analysis. Springer. 2002
Website of the project EQUAL. Methoden zur Unterstützung der entwicklungsbegleitenden Qualitätssicherung von eingebetteter Software (EQUAL). http://www.embedded-quality.de/

BIOGRAPHY

Matthias Wiemann obtained his M.Sc. in software engineering from the University of Potsdam, Germany, in 2006. Currently, he is pursuing his Ph.D. studies at the Technical University of Munich in cooperation with Siemens Corporate Research & Technologies. He advances techniques for SW and system testing as well as fault estimation in the automotive industry since 2004.

Jörg Gericke obtained his diploma in computer science from the University of Erlangen-Nürnberg, Germany. He works in the field of software engineering since 1990, specialized in the medical and automotive industry. Since 2004, he is a project manager at Siemens Corporate Research & Technologies and focusing on system integration tests and system tests of object-oriented and safety-critical systems.
INTEGRATING QUALITY ASSURANCE TECHNIQUES WITH REQUIREMENTS NEGOTIATION
ACTIVITIES: AN ANALYSIS OF SOFTWARE DEVELOPERS PRACTICES

Hazrina Sofian 1, Siti Salwah Salim 2
Department of Software Engineering, Faculty of Computer Science and Information Technology
University of Malaya, 50603 Kuala Lumpur, Malaysia. 1 Email: hazrina_sofian@perdana.um.edu.my
2 Email: salwa@um.edu.my

ABSTRACT

Realizing the importance and the needs of integrating Quality Assurance (QA) techniques with Requirements Negotiation (RN) activities, as this integration is able to produce a consensus agreement and at the same time is able to reduce the number of defected requirements in the negotiation results, this research investigates the RN practices among software developers in Malaysia. Multimedia Super Corridors (MSC) status companies are selected as the participants of the study. The Repeatable Quality Assurance Requirements Negotiation Methodology is used as a framework of this survey study. This paper reports the findings that emerged from this investigation which highlights both, the status of RN activities practices and the status of integration of QA technique. The investigation also highlights on the needs of an electronic system to support the RN process, which leads to the proposed QARN (Quality Assurance Requirements Negotiation) tool which integrates QA techniques with RN activities.

Keywords: Requirements negotiation, Quality Assurance techniques, EasyWinWin activities, Software developer requirements negotiation practices, Requirements negotiation methodologies.

INTRODUCTION

Requirements Negotiation (RN) is a process of deciding which requirements are to be accepted and capable to make the most significant impact on the system’s value by the involvement of a collaboration of numerous stakeholders [1], [9]. According to [13], RN produces robust requirements which address important concerns of all the stakeholders. However, the negotiation results derived from RN often contain defects. These defects should be removed to avoid failure to a software project because even a minor defect in a negotiation results can become serious in other software development life cycle as it serves as an input to the following software development phase such as contracts, specifications, project plans or architecture models [4], [9]. Defects can be detected and removed using Quality Assurance technique (QA) by performing rapid quality checks before, during and after RN. These techniques have been empirically evaluated on its effectiveness and efficiency in detecting and removing defects [4]. Therefore, there is a need of integrating QA techniques with RN activities as this integration is able to produce consensus agreement and at the same time is able to reduce the number of defected requirements in the negotiation results [6].

Realizing the importance and the needs of integrating Quality Assurance (QA) techniques with Requirements Negotiation activities, this paper presents the status of RN practices among MSC status companies. Figure 1 shows the approach employed in this study. It starts with a review on existing RN methodologies which becomes the basis for selecting an appropriate methodology as a framework of the survey study. The selected methodology is used to capture the RN practices among software developers. Important findings and conclusion derived from the survey study are highlighted in the paper.

Fig. 1: Approach employed in this study
REPEATABLE QUALITY ASSURANCE REQUIREMENTS NEGOTIATION METHODOLOGY

Four existing requirements negotiation methodologies were reviewed: WinWinNegotiation [7][14][15], MPARN [11][12], EasyWinWin(EWW) [2][3][5][8][10][16], and Repeatable Quality Assurance for Requirements Negotiation(RQA-RN) [4][6]. These methodologies are correlated to one another as MPARN and EWW adopt WinWin Negotiation, and RQA-RN adopts EWW as the basis of the methodologies. Fig.2 shows the evolution of RN methodologies.

![Evolution of RN Methodologies](image)

Table 1 shows the supported activities of each methodology in which RQA-RN methodology supports most of the activities in requirements negotiation with the exception of quantitative assessment.

<table>
<thead>
<tr>
<th>Supported Activities/Techniques</th>
<th>RN Methodologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. WinWin Negotiation activities; support stakeholders in reconciling win conditions</td>
<td>✓    ✓    ✓    ✓</td>
</tr>
<tr>
<td>2. Multi-Criteria Preference Analysis activities; support quantitative assessment on how the developing system should behave.</td>
<td>✓</td>
</tr>
<tr>
<td>3. EasyWinWin activities; facilitate software developer to collaboratively-engaged in discovering, elaborating, prioritizing and negotiating requirements.</td>
<td>✓    ✓</td>
</tr>
<tr>
<td>4. Collaborative activities; support stakeholders to collaboratively-engaged in throughout RN</td>
<td>✓</td>
</tr>
<tr>
<td>5. Quality Assurance techniques; identify and remove defects in requirements.</td>
<td>✓</td>
</tr>
</tbody>
</table>

RQA-RN methodology adopts EasyWinWin(EWW) negotiation model as the basis for the requirements negotiation activities, integrate these activities with QA techniques and supported them with collaborative activities. Collaborative activities are essential as there are many stakeholders involved throughout RN phases. Figure 3 shows the representation of QA-RN methodology [4], [6] where it consists of 3 main phases: RN Phase 1, Phase 2 and Phase 3. RN Phase 1 includes the preparation activities (P0) to prepare sufficient information that are needed during the negotiation, and the pre-process QA techniques (QA0) to ensure the preparation activities produces the desired results. RN phase 2 includes the value-creation activities (P1) to negotiate requirements and the in-process QA techniques (QA1) to support quick quality checks as this phase is highly dynamic and interactive. RN Phase 3 includes the finalization activities (P2) and the post-process QA techniques (QA2) to perform inspection of potential defects in the negotiation results.
SURVEY ON THE RN PRACTICES

Based on the above selected RN methodology, 2 sets of online questionnaires were produced, which are Questionnaire A and Questionnaire B. RN activities and QA techniques were grouped under three main phases, namely RN phase 1, RN phase 2, and RN phase 3. In these online questionnaires, all questions associated with RN activities and QA techniques were set with compulsory restriction to prevent participants from leaving out any questions. A ‘Do not know’ option was provided as an alternative choice for participants who are facing uncertainty in answering any questions concerning RN activities and QA techniques.

Questionnaire A aims at investigating the requirements negotiation practices within software development companies. It explores into the:
- Background of RN practices within software development companies: the employment of a specific RN methodology and the use of RN tools.
- Practices of each RN activity and QA technique. This section used a rating scale of ‘Yes’ and ‘No’.
- Other RN activities or QA techniques which were not included in the selected methodology.

Whilst Questionnaire B aims at investigating how software developers feel about the importance of each RN activities and QA techniques towards their companies. It examines:
- The importance of each RN activities and QA techniques using a rating scale of 1 to 4, where 1 represents totally not important, 2 represents not important, 3 represents important and 4 represents totally important.
- Opinions on employing QA techniques in negotiating requirements.

Participants

The online questionnaires were sent out to nine MSC status companies from the following three categories:
1. IT consulting companies – Provide IT advisory services by understanding industry-specific business issues and trends
2. Software development companies - Provide software development services
3. IT development companies – Provide in-house software development services.

A total of 18 software developers responded to the Questionnaire A and 23 to the Questionnaire B. They were involved in both, private and government software projects.

RESULTS

Figure 1 presents the results on the employment of a specific RN methodology and the use of RN tools. Tables 2 to 5 present the detailed survey results on the RN activities and the QA techniques being practiced by the software
companies, as well as on their views on the importance of the RN activities and QA techniques towards their companies.

**The employment of a specific RN Methodology and the use of RN Tools**

![Fig. 4: The % Ratings on the RN Methodology and RN Tools.](image)

Most of the software companies do not establish a specific approach on RN. As shown in the Figure 4, only 21.74% of them practice a structured and standard methodology. All participants in the study stated that they do not use any RN tools in their working environment, and they encouraged the use of an electronic system to support RN activities.

**RN Activities practices**

As shown in Table 2, the five main RN activities being practiced by the software companies are brainstorming stakeholder interest, prioritizing win conditions, identifying issues and constraints, revealing options, and negotiating agreements. The mode values indicate that converging win conditions is being less practiced in comparison to all other RN activities.

<table>
<thead>
<tr>
<th>Table 2: The % Ratings of RN Activities Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RN activities</strong></td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td><strong>RN PHASE 1</strong></td>
</tr>
<tr>
<td>1. Preparation activities</td>
</tr>
<tr>
<td>1.1 Prepare negotiation purpose</td>
</tr>
<tr>
<td>1.2 Prepare context information</td>
</tr>
<tr>
<td>1.3 Prepare major objectives</td>
</tr>
<tr>
<td>1.4 Conduct specific tailoring activities</td>
</tr>
<tr>
<td><strong>RN PHASE 2</strong></td>
</tr>
<tr>
<td>2. Value-creation activities</td>
</tr>
<tr>
<td>2.1 Brainstorm stakeholder interest</td>
</tr>
<tr>
<td>2.2 Converge win conditions</td>
</tr>
<tr>
<td>2.3 Capture glossary of terms</td>
</tr>
<tr>
<td>2.4 Prioritize win conditions</td>
</tr>
<tr>
<td>2.5 Identify issues and constraints</td>
</tr>
<tr>
<td>2.6 Reveal options</td>
</tr>
<tr>
<td>2.7 Negotiate agreements</td>
</tr>
<tr>
<td><strong>RN PHASE 3</strong></td>
</tr>
<tr>
<td>5. Finalization</td>
</tr>
<tr>
<td>5.1 Inspection preparation</td>
</tr>
</tbody>
</table>

**Importance of RN Activities**

©FCSIT, UM 2007 T2-18
The mode values of the importance of RN activities show that they are all important to be practiced. Even though the converge win conditions is rated as being less practiced (as in Table 2), it is still rated as an important activity.

### Table 3: The % Ratings of the Importance of RN Activities

<table>
<thead>
<tr>
<th>RN activities</th>
<th>Rating</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Do not know</th>
<th>Mean</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RN PHASE 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Preparation activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 Prepare negotiation purpose</td>
<td></td>
<td>-</td>
<td>17.39</td>
<td>78.26</td>
<td>4.35</td>
<td>3.82</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1.2 Prepare context information</td>
<td></td>
<td>-</td>
<td>43.48</td>
<td>47.83</td>
<td>8.69</td>
<td>3.52</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1.3 Prepare major objectives</td>
<td></td>
<td>-</td>
<td>4.35</td>
<td>34.78</td>
<td>60.87</td>
<td>-</td>
<td>3.57</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Conduct specific tailoring activities</td>
<td></td>
<td>-</td>
<td>8.69</td>
<td>39.13</td>
<td>39.13</td>
<td>13.04</td>
<td>3.35</td>
<td>4</td>
</tr>
<tr>
<td><strong>RN PHASE 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Value-creation activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Review negotiation topics</td>
<td></td>
<td>-</td>
<td>8.69</td>
<td>47.83</td>
<td>47.83</td>
<td>-</td>
<td>3.39</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Brainstorm stakeholder interest</td>
<td></td>
<td>-</td>
<td>13.04</td>
<td>86.96</td>
<td>-</td>
<td>3.87</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2.3 Converge win conditions</td>
<td></td>
<td>-</td>
<td>17.39</td>
<td>21.74</td>
<td>60.87</td>
<td>-</td>
<td>3.43</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Capture glossary of terms</td>
<td></td>
<td>-</td>
<td>8.69</td>
<td>21.74</td>
<td>60.87</td>
<td>8.69</td>
<td>3.57</td>
<td>4</td>
</tr>
<tr>
<td>2.5 Prioritize win conditions</td>
<td></td>
<td>-</td>
<td>17.39</td>
<td>21.74</td>
<td>60.87</td>
<td>-</td>
<td>3.43</td>
<td>4</td>
</tr>
<tr>
<td>2.6 Identify issues and constraints</td>
<td></td>
<td>-</td>
<td>47.83</td>
<td>52.17</td>
<td>-</td>
<td>3.52</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2.7 Reveal options</td>
<td></td>
<td>-</td>
<td>4.35</td>
<td>39.13</td>
<td>47.83</td>
<td>8.69</td>
<td>3.48</td>
<td>4</td>
</tr>
<tr>
<td>2.8 Negotiate agreements</td>
<td></td>
<td>-</td>
<td>30.43</td>
<td>69.57</td>
<td>-</td>
<td>3.69</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>RN PHASE 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Finalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1 Inspection preparation</td>
<td></td>
<td>-</td>
<td>17.39</td>
<td>60.87</td>
<td>13.04</td>
<td>8.69</td>
<td>2.95</td>
<td>3</td>
</tr>
</tbody>
</table>

### QA Techniques Practices

As shown in Table 4, the three main QA techniques being practiced by the software companies are checking initial set of win conditions and glossary of terms, producing WinWin Tree, and inspecting negotiation results. The mode values indicate that the technique ‘review checklist containing typical defects’ is being less practiced in comparison to all other QA techniques.

### Table 4: The % Ratings for QA Techniques

<table>
<thead>
<tr>
<th>QA techniques</th>
<th>Rating</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Do not know</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RN PHASE 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Pre-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 Review checklist containing necessary steps</td>
<td></td>
<td>72.22</td>
<td>27.78</td>
<td>-</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>1.2 Review checklist containing typical defects</td>
<td></td>
<td>33.33</td>
<td>61.11</td>
<td>5.56</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3 Conduct risk analysis</td>
<td></td>
<td>72.22</td>
<td>16.67</td>
<td>11.11</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>RN PHASE 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. In-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Check context information and negotiation topics</td>
<td></td>
<td>66.67</td>
<td>33.33</td>
<td>-</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2.2 Check initial set of win conditions and glossary of terms</td>
<td></td>
<td>83.33</td>
<td>16.67</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 Produce WinWin Tree</td>
<td></td>
<td>83.33</td>
<td>16.67</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4 Check the completeness of negotiation results</td>
<td></td>
<td>66.67</td>
<td>27.78</td>
<td>5.56</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5 Check the completeness of documentation</td>
<td></td>
<td>72.22</td>
<td>27.78</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RN PHASE 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Post-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1 Inspection preparation</td>
<td></td>
<td>72.22</td>
<td>27.78</td>
<td>-</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>3.2 Inspect negotiation results</td>
<td></td>
<td>83.33</td>
<td>16.67</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3 Meeting of inspectors</td>
<td></td>
<td>61.11</td>
<td>38.89</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4 Report and rework</td>
<td></td>
<td>66.67</td>
<td>33.33</td>
<td>-</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Importance of QA Techniques
The mode values of the importance of QA techniques show that they are all totally important in detecting and removing defects throughout the requirements negotiation activities, except for the technique ‘review checklist containing typical defects’ where it is rated as less important in comparison to the rest of the QA techniques. These results are consistent with the current practice of their QA techniques (as in Table 4).

Table 5: The % Ratings of the Importance of QA Techniques

<table>
<thead>
<tr>
<th>QA Techniques</th>
<th>Rating</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Do not know</th>
<th>Mean</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RN PHASE 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Pre-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 Review checklist containing necessary steps</td>
<td></td>
<td>-</td>
<td>8.69</td>
<td>47.83</td>
<td>39.13</td>
<td>4.35</td>
<td>3.32</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Review checklist containing typical defects</td>
<td></td>
<td>-</td>
<td>21.74</td>
<td>30.43</td>
<td>39.13</td>
<td>8.69</td>
<td>3.19</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Conduct risk analysis</td>
<td></td>
<td>-</td>
<td>-</td>
<td>21.74</td>
<td>78.26</td>
<td>-</td>
<td>3.78</td>
<td>4</td>
</tr>
<tr>
<td><strong>RN PHASE 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. In-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Check context information and negotiation topics</td>
<td></td>
<td>-</td>
<td>-</td>
<td>30.43</td>
<td>52.17</td>
<td>17.39</td>
<td>3.63</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Check initial set of win conditions and glossary of terms</td>
<td></td>
<td>-</td>
<td>-</td>
<td>60.87</td>
<td>30.43</td>
<td>8.69</td>
<td>3.33</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Produce WinWin Tree</td>
<td></td>
<td>-</td>
<td>-</td>
<td>47.83</td>
<td>39.13</td>
<td>13.04</td>
<td>3.45</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Check the completeness of negotiation results</td>
<td></td>
<td>-</td>
<td>-</td>
<td>47.83</td>
<td>47.83</td>
<td>4.35</td>
<td>3.50</td>
<td>4</td>
</tr>
<tr>
<td>2.5 Check the completeness of documentation</td>
<td></td>
<td>-</td>
<td>-</td>
<td>47.83</td>
<td>52.17</td>
<td>-</td>
<td>3.52</td>
<td>4</td>
</tr>
<tr>
<td><strong>RN PHASE 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Post-process QA techniques</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1 Inspection preparation</td>
<td></td>
<td>-</td>
<td>-</td>
<td>52.17</td>
<td>43.48</td>
<td>4.35</td>
<td>3.45</td>
<td>4</td>
</tr>
<tr>
<td>3.2 Inspect negotiation results</td>
<td></td>
<td>-</td>
<td>-</td>
<td>47.83</td>
<td>47.83</td>
<td>-</td>
<td>3.43</td>
<td>4</td>
</tr>
<tr>
<td>3.3 Meeting of inspectors</td>
<td></td>
<td>-</td>
<td>8.69</td>
<td>30.44</td>
<td>47.83</td>
<td>13.04</td>
<td>3.45</td>
<td>4</td>
</tr>
<tr>
<td>3.4 Report and rework</td>
<td></td>
<td>-</td>
<td>-</td>
<td>52.17</td>
<td>47.83</td>
<td>-</td>
<td>3.48</td>
<td>4</td>
</tr>
</tbody>
</table>

Participants were also asked to give their views on the need of integrating QA techniques with RN activities. With the percentage of 84.62%, majority of the participants agree that Quality Assurance techniques are able to enhance the quality of the negotiation results.

DISCUSSION AND CONCLUSION

This study commenced with the review and the comparative analysis on 4 existing RN methodologies, which initiated the selection of Quality Requirements Negotiation Methodology as the framework of the survey. Following are the important findings derived from this study with the participation from nine MNC status companies:

- Most of the software companies do not establish a specific approach on RN and they acknowledged the importance of having a structured and standard RN methodology.
- The software developers agreed on the ability of Quality Assurance Requirements Negotiation Methodology in achieving consensus agreement and enhancing the quality of negotiation results.
- All the software companies do not use any RN tools in their working environment. Majority of the software developers encouraged the implementation of an electronic system to support their RN activities.
- All the RN activities extracted from the Quality Assurance Requirements Negotiation Methodology are in practice by the software developers. The focus of their activities is centered on the RN phase 2 (the value-creation activities). There are no new RN activities being specified by the software developers. All the activities are rated as totally important to be practiced, except for the activity of ‘converging win conditions’ which is rated as being important.
- Similar to RN activities, all the QA techniques are being practiced by the software companies and they are rated as totally important in detecting and removing defects throughout the requirements negotiation activities, except for the technique of ‘reviewing checklist containing typical defects’ which is rated as important.
- Most of the software companies do integrate QA techniques with their RN activities in their RN approach, but majority of the software developers feel that the integration of the QA techniques and the RN activities are not structured and standardized.

©FCSIT, UM 2007
Requirements negotiation is imperative to ensure the right requirements are selected while quality assurance activities are crucial in detecting requirements’ defects. Hence, both requirement negotiation and quality assurance activities shall be implemented in the development of a requirement negotiation tool in order to support the decision-making process and enhance the quality of negotiation results (i.e. Quality Assurance Requirements Negotiation Tool). The Repeatable Quality Assurance Requirements Negotiation Methodology used in the survey study covers all the activities and techniques currently being practiced by the software developers, thus, suitable as the basis for the design and implementation of the Quality Assurance Requirements Negotiation Tool.

REFERENCES


AUTHOR’S BIOGRAPHY

Hazrina Sofian is a masters student at the Faculty of Computer Science and Information Technology, University of Malaya. She is currently involved in the implementation of a Requirements Negotiation Tool that integrates QA techniques with RN activities.

Dr Siti Salwah Salim is Professor of the Department of Software Engineering, at the Faculty of Computer Science and Information Technology, University of Malaya.
MODEL DRIVEN ARCHITECTURE FOR INTEGRATED ENTERPRISE APPLICATIONS – USING MOF

Subaji Mohan, Eunmi Choi
School of Business Information Technology
Kookmin University, Jeongmueng-Dong, Seongbuk-Gu, Seoul, 136-702, Korea
Email: subaji@kookmin.ac.kr, emchoi@kookmin.ac.kr

Abstract

Integration of business applications are hindering companies as there is rapid change in business requirements and the complexity in developing enterprise-spanning applications. To cope with the changing requirements and high complexity, it is essential to integrate applications on the business process and conceptual level. Because of the diversity of models and modeling languages for developing enterprise applications, a meta-model approach using the concept of MDA is discussed in this paper. The two major models facilitated by MDA are PIM & PSM. PIM is a model with a high level abstraction and independent of any implementation technology and PSM contains all the syntactic, semantic, and presentation information regarding the domain. In this paper, the basic concepts of meta-model integration of enterprise applications is described, which will provide flexibility and interoperability.

Keywords: Model Driven Architecture (MDA), Meta-Model, Meta Object Facility (MOF), Platform Independent Model (PIM), Platform Specific Model (PSM)

1.0  Introduction

The problem of aligning and integrating business and Information Technology (IT) is restraining many companies in their strategic and tactical development. In modern business process, an integrated approach of business and IT is indispensable [1]. It is clear that even though IT has evolved from its traditional orientation of administrative support towards a more strategic role within an organization [2]. Most organizations are using packaged software’s for their key business processes like Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Customer Relationship Management (CRM), and Electronic Commerce (EC), which helps to improve the leverage on using information systems to support their operational and financial goals [3]. Each of the applications is developed in different platforms and organizations are facing problems to make it work together.

Model Driven Architecture (MDA) provides an open neutral approach to overcome the challenge posed by business and technology change. MDA has the ability to separate business and application logic from underlying platform technology [5]. Platform Independent Model (PIM) of an application (or) integrated system’s business functionality and behavior built using UML and other associated modeling standards, can be realized through MDA on virtually any platform. These Platform Independent Models document the business functionality and behavior of an application separate from the technology specific code that implements it and this can be transformed to Platform Specific Models (PSM) using MDA mapping. MDA is an approach that facilitates the means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification. In this paper we are proposing the concept of meta-model through MDA approach, which will help to improve the flexibility and interoperability among the Enterprise Applications.

The structure of the paper is as follows. In Sections 2 and 3, an overview of Integration Issues and various Enterprise Integration approaches are highlighted. In Section 4, the concept of MDA is reviewed. In Section 5, we introduce our proposed MOF architecture and how it is used to support the Integration of Enterprise Applications. Finally, in Section 6, we provide an analysis of the proposed Model Driven Architecture and our future work

2.0  Integration Issues

The term integration has diverse meanings in different disciplines. For example, in Information Systems (IS) research integration deals with the combination of different systems and data sources, whereas in business integration it is characterized as an interactive or collaborative process. Therefore a comprehensive definition is elusive in an interdisciplinary setting, mainly due to the domain-specific terminology and lack of comparable structures.

©FCSIT, UM 2007
Different approaches had summarized that integration means cross and inter-functional co-operation as well as process overlap. To facilitate this four 'mechanisms', namely strategy, process, technology, and organization are to be considered to enable integration. Efficient coordination and control of these business processes can only be achieved with integrated information systems that can deliver right information at right time at every point in the value chain. However, the results have not lived up to the expectations. The reasons behind this include:

**Enterprise Issues**: Enterprises encounter constraints as they grow. One common constraint is the system inability to scale with the increasing enterprise processing needs. Another constraint is the geographical location which brings the challenge of distribution. It also specifies the importance of bringing the same level of services to a diversely located enterprise.

**System Issues**: One of the major constraints is connecting existing separate systems. Legacy systems were typically designed with specific purposes in mind and were not specifically conceived with integration with other systems.

**Software Issues**: The inability for one standard software provider to deliver hundred percent of the business software today's organizations require combined with the need to integrate legacy applications has created heterogeneous IS architectures.

**Financial Issues**: Organizations spend at least forty percent of their IT budget for integration. The costs of integration are playing a major part when implementing and maintaining packaged applications in the early 21st Century the amount spent has already crossed US $ 85 billion approx around the world.

**Internal Issues**: In an enterprise which involves processes, people and technology internal integration play a vital role. It forces enterprises to be in a position to re-organize its business processes and the ability to facility integration with the legacy or internally developed systems.

**External Issues**: The integration of business partners, such as customers and suppliers, not only at the data but also at the process level, requires a time and cost efficient integration approach.

In this complex and dynamic environment, homogenous IS architectures are no longer practical options. Many organizations are changing their strategies from a single sourcing or traditional point-to-point integration strategy to a more proactive approach. It would not be realistic to see that companies will throw their existing design practice and tools overboard and replace these by an entirely new approach. Enterprise modeling should focus on bringing together already existing techniques and integrating these at the appropriate level of abstraction.

### 3.0 Overview of Enterprise Application Integration Approaches

The enterprise application integration approaches have focused on one particular area, such as data or processes. The following are some of enterprise application integration approaches:

- **Data integration** is a simple and inexpensive approach to implement [7]. Data can be moved between different business domains and can be shared, regardless of the database systems, operating systems or networking platform. However, this approach assumes bypassing of the application logic, therefore limiting 'real time' transactional capabilities [9].

- **Process integration** is more advanced than data and object integration, as the logic, and reasoning, for conducting business are included. It removes the data integration limitation by focusing on the actual process and not only on the data. It requires a high level of investment and is complex to architect and develop [9].

- **Object integration** encapsulates the business logic and data within objects allowing the objects to be linked together in a plug and play manner to interoperate. It allows the legacy system to be wrapped into objects, and participate in the object integration [9]. However, this approach is complex and the most difficult approach [7].

- **User interface integration** approach is the easiest and the faster way of integration. However it has many limitations. This approach is synchronous in its communication style, and requires the original application to be online [7]. It only takes place at the presentation, and not at the actual inter-connection between applications and data [8].
Application interface integration allows external applications to access existing application services without modifying the existing application. However, this approach has many limitations; it is synchronous in nature where the function call is not returned until the specific action is completed, and requires both applications to be available, which may result in a tightly coupled system [10].

4.0 Model Driven Architecture

The Model Driven Architecture (MDA) is a fairly new resource in the software development field. It facilitates in creating good designs that can cope with multiple technology deployments of a software system and is based on widely used standards like the Unified Modeling Language (UML). The intention of the MDA is to create machine-readable models that can be understood by automatic tools that generate schemas, code skeletons, testing models, test packs, and integration code for multiple platforms and technologies. The core of the MDA depends on the models that are created as part of the software development process. There are two models at the heart of the MDA as follows:

Platform Independent Model (PIM) - The PIM is a highly abstracted model that is independent of any implementation technology. It describes a software system that supports a part, or the whole, of business. The software system is modeled from the perspective of how it best supports the business. The PIM may include generic functions, scenarios (process descriptions of how the system realizes a business requirement) and class descriptions.

Platform Specific Models (PSM) - Using the PIM as a foundation, it is then transformed into one or more platform specific models, which describes in detail how the PIM is implemented on a specific platform, or technology. Depending on the platforms across which the software system is going to be deployed PSM's will be created - one per platform, or technology.

When modeling of the MDA is realized, there are a number of benefits it would bring to the software development community. The two main benefits are:

Productivity - The developer will focus on the development of a PIM. From the PIM, the PSM's and codes can be automatically created via transformations. With less focus on the coding and detail design for specific platforms, the developers can spend more time in accommodating business problems. This will ensure better business fit and hopefully a happier user community.

Portability - Portability is achieved via the PIM that is transformed into PSM's for the multiple platforms on which deployment will take place. With the transformation between PIM and PSM automated the PIM becomes totally portable.

5.0 Proposed MDA model for Enterprise Architecture

In the dynamic business environment, applications such as CRM, SCM, ERP and integrated collaborative environments have become imperative for organizations that need to maintain their competitive edge. It is essential to integrate applications on the business and conceptual level as well as using models such as business models, business process models, products models, interaction models, and interface models [6]. We try to use MDA to achieve better integration among the disparate enterprise applications.

The three primary goals of MDA are to provide portability, interoperability, and reusability. Even though standardizations on modeling level still to be found, almost the prominent tool is the UML for modeling object-oriented systems [12]. Apart from this, OMG community started to establish a relatively new vision with the Meta Object Facility (MOF) and the Model Driven Architecture (MDA) to improve productivity in software development, applying object orientation, meta-level concepts and modeling [13, 14].

In order to enable the integration of models of an enterprise, the necessary prerequisite is the integration of their underlying meta-models [15, 16]. MOF will be used to describe the meta-meta-models as MOF specification defines the set of CORBA Interface Definition Language (IDL) interfaces that can be used to define and manipulate a set of interoperable meta-models and their corresponding models. It also specifies precise mapping rules that enable the CORBA interfaces for meta-models to be automatically generated thus encouraging consistency in manipulating meta-data in all phases of the distributed application development cycle.

In the figure 1, the level 3 of the architecture shows the business, process, and product in the meta-meta-model form. Since UML and MOF share the same core semantics, it allows MOF to reuse the UML notation for visualizing meta-
models. MOF allows interoperability across the application development cycle by supporting the definition of a single Object Analysis & Design (OA&D) model.

The OA&D facility defines the interfaces and semantics needed to support the creation and manipulation of a core set of OA&D models that define the structure, meaning, and behavior of object applications within the OMG Object Management Architecture [13]. The OA&D facility will play a key role in achieving semantic interoperability between OA&D tools, such that users of these tools may be ensured that models created by one tool are meaningful to another tool.

MOF meta-model is the meta-model for the UML meta-model. UML meta-model may be considered an instance of the MOF meta-meta-model. It is also referred as loose modeling. The MOF meta-meta-model will be used in the MDA as platform independent model which will be transformed to platform specific model with the help of the transformation engine. The transformation of meta-model in MDA is shown in the figure 2.

![Fig. 1: Representation of MOF Architecture](image)

The platform independent meta-model is the Business Process Model and the platform specific meta-model is a MOF model of a work flow engine. The transformation specification that will be used is a MOF QVT transformation model. The transformation will be carried out by a transformation engine created by a tool, which uses a pair of MOF models to build an engine for a specific transformation. In addition to successive transformation of complete models, models need to be woven together with other models to produce a system. The models linked together and translated into code, the executable models become systems. To effect this combination, a mapping function can be defined. These mappings can be between any two kinds of identifiable entity in an executable UML model. Once these mappings are defined, then all the models can be combined into a single populated meta-model from which code can be generated. The mechanism responsible for compiling is a model compiler, which weaves together the several models according to a single set of architectural rules (Figure 3).

![Fig. 2: Meta-Model Transformation](image)

Patterns will be used in the specification of mapping. The mapping includes a pattern and selection of components with marks corresponding to some element of that pattern. In model instance transformations the specified marks are then used to prepare a marked PIM. The marked elements of the PIM are transformed according to the pattern to produce the

![Fig. 3: Merging of Different PIMs to PSM](image)

![Fig. 4: Target Models from PIM](image)
PSM. The platform specific models or target models can be CORBA Component Model (CCM), Enterprise Java Bean (EJB) or .NET as shown in Figure 4.

A decorator pattern with two roles, decoration and decorated supplied a mark, decorated. When this mark is applied to a class in a model, the transformation might produce a class corresponding to that class, with additional operations and attribute, a new class, corresponding to the decoration, and an association between those classes.

In model type transformations rules will specify that all elements in the PIM which match a particular pattern will be transformed into instances of another pattern in the PSM. The components selected by marks will be used to bind values in the matched part of the PIM to the appropriate slots in the generated PSM. In this usage the target patterns can be thought of as templates for generating the PSM, and the use of marks as a way of binding the template parameters.

A mapping from EDOC (Enterprise Distributed Object Computing) ECA (Enterprise Collaboration Architecture) to EJB might include a pattern of ECA types identifying appropriate Process Components and their associated document types as suitable for mapping to EJB Entities and their Remote Interfaces and container managed data classes.

In our approach, MOF the well-established meta-meta-model will be used to build meta-models. The PIM reflects the functionalities, the structure and the behavior of a system. The PSM is more implementation-oriented and corresponds to a first binding phase of a given PIM to a given execution platform. The PSM is not the final implementation, but has enough information to generate interface files, a programming language code, an interface definition language, configuration files and other details of implementation. Mapping from PIM to PSM determines the equivalent elements between two meta-models.

Two or more elements of different meta-models are equivalents if they are compatible and they cannot contradict each other. Model transformation is accomplished by a transformation engine that executes transformation rules. Transformation rules specify how to generate a target model (i.e. PSM) from a source model (i.e. PIM). To transform a given model into another model, the transformation rules map the source into the target meta-model. The transformation engine takes the source model, executes the transformation rules, and gives the target model as output.

As shown in Figure (5) in the level of Meta-Meta-Model the Business Process on work flow model and Business information on Data and Reference Models are designed. A business process can be defined as a set of interrelated tasks linked to an activity that spans functional boundaries. Applying workflow to a business process certainly brings the details of that process into focus as we lay out the business process definition and add the required business rules. This is mapped into session beans in the level of meta-model as they are typically used for business process or control logic. It is a model-specific entity containing workflow and access to special transient business services. Similarly, we design the Data and Reference models in the meta-meta-model level and mapped into entity beans. An entity bean represents a business object in a persistent storage mechanism. Entity beans are persistent, allow shared access, handle information and may participate in relationships with other entity beans. The session and entity beans can generate remote interfaces and integration of objects with service oriented design. The remote interface serves to identify interfaces whose
methods may be invoked from a non-local distributed objects and services. The System and Application Integration Model is derived from the specific model level, deploying the EJB target system of the model.

6.0 Conclusion

MDA has shown to be a valuable aid in application development projects. The diversity of modeling languages, programming languages, target systems, development methodologies, and standardizations, lead us to a flexible meta-model integration approach using Meta Object Facility concepts and technologies. The major advantage using this approach is that it helps to establish integration at meta-meta-model level and increase the quality of delivered solutions, and enhance acceptance level because of direct mapping of the domain. The usage of meta-model integration patterns will help us to standardize the procedures of meta-model integration. We are currently working on concepts and mechanisms to integrate on meta-meta-model level and trying to integrate by identifying the pragmatics, syntactics and semantics of various application systems. Modeling always means effort in time and costs. Nevertheless, to handle complexity in business and application engineering, we see models moving towards more applications oriented.

REFERENCES


BIOGRAPHY

Subaji Mohan is a Doctoral student at School of Business Information Technology, Kookmin University, Seoul, Korea. He was working as Asst. Director (International Relations) at VIT University, India since 2000 before joining for his PhD study at Kookmin University in 2007. His research areas include Business Process Modeling, Model Driven Architecture and Customer Relationship Management. He received his Bachelors Degree in Mechanical Engineering from MS University, India and Masters Degree in Business Administration from University of Madras, India, in 1993 and 1999.

Eunmi Choi is an Associate Professor in the School of Business IT at Kookmin University, Korea, where she has been since 2004. Before joining Kookmin University, she was an Assistant Professor in Computer Science and Electronic Engineering at Handong Global University, Korea. Her current research interests include distributed system, middleware system, system architecture modeling, ubiquitous computing, and grid & cluster computing. Professor Choi received MS and a PhD in computer science from Michigan State University, U.S.A., in 1991 and 1997, respectively, and a BS in computer science from Korea University in 1988.
AN OBJECT ASSERTION LANGUAGE FOR INTEGRITY CONSTRAINTS IN OODBS

Belal Zaqaibeh\textsuperscript{1}, Hamidah Ibrahim\textsuperscript{2}

\textsuperscript{1}Department of Information Technology, Zarqa Private University, Jordan, Email: belal@zpu.edu.jo
\textsuperscript{2}Department of Computer Science and IT, Universiti Putra Malaysia, Email: hamidah@fsktm.upm.edu.my

Abstract

In this paper an Object Assertion Language for Integrity Constraints (OALIC) is proposed. The OALIC is used to create classes and collect attributes and their constraints that are derived from composition and inheritance hierarchies. The OALIC is designed to support enforcing and maintaining integrity constraints for Object-Oriented Databases (OODBs). A new technique called detection method is designed to check the object metadata to detect and catch the OODBs violation before it occurs. Furthermore, we have implemented the OALIC and supported set of definitions that are for checking attribute values validity, OODB consistency, and also a method for verifying attribute values when inserting, deleting, and updating objects.

Keywords: Assertion language, object-oriented databases, integrity constraints, constraints violation.

1.0 Introduction

Checking the Integrity Constraints (ICs) in OODBs is a fundamental dilemma in database design. The current OODB Management Systems (OODBMSs) lack the capability of an ad-hoc declarative specification of enforcing and maintaining ICs that appear as a result of association, composition, and inheritance hierarchies. ICs are conditions that data within a database must satisfy. Integrity maintenance or constraint enforcement is a set of activities to keep OODBs in a consistent state, where all instances in the OODB satisfy ICs.

This paper presents our contribution for this research, in which it clarifies the Assertion Model of ICs (AMIC) properties and specifications including Object Assertion Language for ICs (OALIC), Object Metadata (OMD), and Detection Method (DM). The AMIC has made a big challenge in the OODM environment as it can represent constraints and complex relationships among attributes and classes that are derived from composition and inheritance hierarchies, whereas the current OODBs are deficient in such properties.

This paper is organized as follows. Section 2 presents groundwork of our research. Section 3 explores the OALIC and their structure format and grammar. Consequently, the AMIC architecture and components are presented in Section 4. Section 5 presents the DM and Section 6 explore the OMD. Subsequently, the AMIC features are shown in Section 7 and related works are discussed in Section 8. Naturally, we end this chapter by a conclusion and future work in Section 9.

2.0 Preliminary

The increased emphasis on process integration is a driving force for the adoption of OODBSs [2]. For example, the CAD area is focusing heavily on using OODB technology as the process integration framework. Advanced office automation systems use OODBSs to handle hypermedia data. Image processing and designing systems use OODB technologies for ease of use. All of these applications are characterized by having to manage complex, highly interrelated information, which is the strength of OODBSs. Clearly, relational database technology has failed to handle the needs of complex information systems [7].

ICs in OODBs are maintained either by rolling back transactions that produce an inconsistent state, or by disallowing operations that may produce an inconsistent state for the constraints [2] [4] [10]. Existing OODBMSs lack the capability for an ad-hoc declarative specification of maintaining the ICs. An alternative approach is to provide automatic detection of inconsistent states.

Some ICs are represented naturally and maintained in OODBs, by capturing the violation using the type system and the class hierarchy [11]. Checking the ICs in OODBs is a fundamental dilemma in database design, because current OODBMSs lack the capability of an ad-hoc declarative specification of maintaining ICs that appear as a result of composition, inheritance, and association hierarchies. The AMIC can represent ICs and their relationships over the composition and inheritance hierarchies [12].
3 The OALIC

The OALIC is the assertion language that is proposed to handle classes and their attributes and constraints. The OALIC is design to simplify and support constraints to the AMIC.

3.1 The Structure Format of OALIC

The general structure of the OALIC format is illustrated in Figure 1. All attributes are gathered under the specifier ATTRIBUTE, methods under the specifier METHOD, and CONSTRAINT is added to gather constraints [13]. Also a new method called DM is introduced to express the status of the constraints.

![Figure 1: The General Structure of OALIC Format](image)

The user can manipulate (insert, delete, or modify) attributes, methods, and constraints. The AMIC has been designed to maintain redundant, inconsistent, and duplicate constraints, also to enforce ICs and keep the OODB in a consistent state. If a violation is expected to be occurred, the AMIC will do several actions as follows:

1. Send the current user request and the constraint derivation path to the maintenance technique.
2. For each user request to the OODB, the DM checks the OMD and assigns the new values to the DM variables.
3. The AMIC verifies the DM then specifies the dependences among constraints.
4. If the constraints cannot be maintained then the error handler technique keeps the violation path and type. Moreover, the current user request is aborted.

3.2 The OALIC Grammar

The grammar of OALIC as per the ISO/IEC rules format is shown in Figure 2 where “|” means a multiple-choice, optional “[ ]” means zero or once, repetition “{ }” means zero or many times. “< >” means non-terminals. “::=” means the equivalence from the right hand side to the left hand side, and “+” means many times but at least once.

![Figure 2: OALIC Grammar](image)
The constraints are the conditions that control attributes values in intra-class or inter-class. The constraint operands (Constant value, Literal, Attributes, Expression, or Aggregate function) that enforce attribute values. Once a class is created, all members (attributes, methods, and constraints) are gathered and their relationships are specified. The idea behind gathering class members is to find the relationships, specify the dependences, and keep the derivation path. Typically, the AMIC keeps the relationships and the derivation path for each class member. Therefore, all constraints (constraint base) and relationships (constraint knowledge) will be collected, analyzed, optimized, and stored in the OMD.

The AMIC uses the hierarchy model to keep the derivation path, constraint base, and constraint knowledge. When a violation occurs in any attribute, the AMIC can provide the violation path and the constraint that causes the violation. The maintenance technique in AMIC will handle the violation automatically.

4 The AMIC Framework

The AMIC framework is illustrated in Figure 3. The framework consists of five components namely, OODB, OMD, DM, rules, and applications. Users use OALIC to create the classes and their attributes, methods, and constraints. The AMIC gathers all attributes and constraints, identifies the relationships, optimizes constraints, and stores them in the OMD. The DM is used to update and retrieve information from the OMD.

The OODB stores objects with their attributes, methods, and constraints. The attributes and methods represent entities and their behaviors. A constraint can be defined on attributes. The participating objects are represented in AMIC by translating User Defined Constraints (UDCs) into rules and these rules are stored in the OMD. The user does not have to specify detailed execution procedure for constraint checking and propagation in the database. The constraints are translated into rules and relationships regardless whether the propagated constraints are derived from composition or inheritance hierarchies.
The OMD is used to manage constraints and also to store the attributes and their paths, constraint knowledge, and constraint base. Furthermore, the OMD provides operations that eliminate conflicts of constraints. All the operations for the AMIC are based on the OMD contents.

The simplest application is based on the access to only objects in the database. In this case, it is guaranteed that the applications always read consistent data and cannot make inconsistent update into the database. Other kind of applications can share objects and methods that can change the state of objects. If an application has interface to the database that embeds the data control language, it can manage the constraints as well as objects and methods because all the objects, methods, and constraints are defined with resources from the database.

The DM checks the constraints in the OMD when a user request is received from applications. It is not easy to enforce the ICs when composition or inheritance hierarchies exist. The DM reads the OMD, finds the involved constraints, checks the involved attributes that are needed to be modified during the objects creations, and also checks the objects. Moreover, if maintenance is needed, the DM calls the maintenance technique and updates the OMD with the result. The user request must be checked in the OMD before it updates the database. When a transaction is received, the DM gets all information about the involved attributes and their constraints, and verifies the new changes that may happen due to the user request. If there is no violation then the DM gets the new changes and updates the OMD. But if the transaction causes inconsistent state, the DM gets the required information about the violation from OMD and stores them in its variables then sends them to OODBMS to abort the running transaction.

Before the AMIC maintains the violated constraints, it recognizes which constraints may violate the database and what is the repairable action. Therefore, a practical working technique in maintaining the constraints when any violation or unexpected circumstances occur is already implemented in AMIC. The AMIC overcome the limitations of Do’s approach [5] by collecting the constraints information in the OMD and calls only the involved constraints when an event occurs.

5 Detection Method

The DM is an overloaded method that can access and modify the OMD. The DM is designed for constraint validation checking purpose. The DM has two functions that are differentiated from each other by their arguments.

\[
\text{DM (CID, AID, RCID, RAID, } \{\text{AC}\}, \{\text{UDC}\}, \{\text{SC}\})}
\]

\[
\text{DM (CID, AID)}
\]

Which Class ID (CID), Attribute ID (AID), Related Class ID (RCID), Related Attribute ID (RAID), set of Antecedent Constraints (ACs), set of User Defined Constraints (UDCs), and set of Supplement Constraints (SCs) are the DM arguments. The CID and AID are the composite key to reach the information about any attribute in the OMD. This information includes constraints, derivation path, domains, derived attributes, and superclasses. The AID represents the ID for an attribute in a particular class, where the ID is unique under the class level; this means the AID can be repeated in different classes. The RCID represents the ID for the superclass if the attribute is a result from inheritance or composition hierarchies. The RAID represents the ID of an attribute if the current attribute is derived from other attribute. For instance, the following example shows an attribute in an independent class with no constraints.
The UDC value indicates whether constraints exist or not on a particular attribute, in other words, if the value of UDC is null, this means there are no constraints on the particular attribute, and if the value of UDC is not null, this means there is a constraint on that attribute. For instance, \( \text{DM}(1, 1, 0, 0, \emptyset, \emptyset, \emptyset) \) gives information about an attribute with \( \text{CID} = 1, \text{AID} = 1 \), not derived from classes (no Generalization), with no dependent attributes (no Specialization), no constraints that depend on, no constraints that have been specified on it, and there is no other attributes that it depends on.

6 Object Metadata

Each attribute has a domain, which is the valid values that can be stored in a particular attribute. A domain attribute is the range of its data type or set of values that are controlled by constraints in different ways like constant values, literals, attributes or aggregate functions. An essential step is that, simplifying the constraints in domains, the attribute domain controls the attribute values in the OMD. In the AMIC model the OMD stores all classes, attributes, constraints, and their relationships.

The OMD has been designed to manage constraints that are in independent, inherited, composed, and associated classes. The OMD consists of three classes as shown in Figure 4, namely: OMD Constraint Optimization (OMD\(^{CO}\)), OMD Constraint Knowledge (OMD\(^{CK}\)), and OMD Knowledge Base (OMD\(^{KB}\)).

![Figure 4: The OMD Structure](#)

The OMD classes are used to describe objects structure. The OMD has all the required information about the constraint base and constraint knowledge. The following sections describe the OMD classes, where these classes are connected with association and composition relationships.

6.1 Constraint Optimization Class

The OMD\(^{CO}\) is the constraint optimization class that optimizes the constraints and the domains. The OMD\(^{CO}\) includes the constraints, domains and Domain ID (DID). The DID is the hashing key that creates a method for searching as opposed to simply scanning a large data with all the nodes. In addition to, it makes addition and removal of nodes more efficient. Furthermore, the DID is associated with the OMD\(^{CK}\) class with M:N as shown in Figure 4. Thus, the DIDs indicate domains for associated attributes.

6.2 Constraint Knowledge Class

The OMD\(^{CK}\) is the class for collecting the attributes, constraints, and domains knowledge. For each attribute in the class there is a unique identifier called AID. Thus, the DM can access and control any attribute using AID. The AMIC can enforce attributes integrity in classes that are result of association, composition, and inheritance. However, to keep the derivation path the OMD\(^{CK}\) keeps the RCID and the RAID that are derived into the present class.
Users can declare their own constraints, which are called UDCs (e.g., parent.age > 18). Typically, a UDC may depend on an Antecedent Constraints (ACs) that are derived from superclasses, associated, or composed classes. Thus, the ACs must be verified before the UDCs verification (e.g., child.age < parent.age). Subsequently, a Supplement Constraints (SCs) are constraints that depend on UDCs, thus, the SCs must be verified after the UDCs verification.

Since the OMD\textsuperscript{CK} has an association relationship with the OMD\textsuperscript{CO}, the AC, UDC, and SCs take their values from the OMD\textsuperscript{CO}. The AC and UDC are declared from sequence data type of DID, and SC from sequence data type of DM. Whereas the AC represents the DID of the domain of dependent attributes, so before any update is the domain in the OMD\textsuperscript{CO} must be satisfied. The domains of the DIDs in the UDC must be satisfied too to remain a consistent state for the database. Accordingly, to prevent a violation that may occur in other attributes, the DIDs in the SC must be checked and satisfied. If a violation occurs in any of AC, UDC, or SC the OODBMS will abort the current user request.

6.3 Knowledge Base Class

The OMD\textsuperscript{KB} is the structure for the OMD object. The OMD\textsuperscript{KB} includes knowledge about all attributes and their relationships, constraints, and domains. Each class has a unique internal identifier called Class ID (CID) and a unique name. Consequently, the OMD\textsuperscript{KB} composes a sequence of OMD\textsuperscript{CK} in the CN attribute. The OMD is the instance of OMD\textsuperscript{KB} class as shown in Figure 5.

In the OMD the OMD\textsuperscript{CK} is a composed object for the CN attribute. Each object in OMD represents class knowledge. So to enforce a particular constraint we need to read only the related object for that constraint and only the involved constraint will be verified, so this reduces the execution time and avoids multitasking. Accordingly, this increases the AMIC efficiency.

![Figure 5: An Instance of OMD\textsuperscript{KB}](image)

7 AMIC Features

At this point, we concentrate on the ICs and data integration for satisfying a set of rules. We extend the formal OODM with standard operators by including two aggregation operands. One of the most significant problems is the incorporation of UDCs with the composition and inheritance mechanisms in the OODM. We have overcome this problem by layering the AMIC over the OODM. This approach has several benefits:

- Represent complex relationships (e.g., M:N) and relationships that propagate from association, composition, and inheritance hierarchies.
- ICs can constrain the action of computationally methods.
- ICs can be applied to arbitrarily complex objects including hierarchy structures.
- ICs are at a higher level of abstraction and thus easier for users to read and write.
- The AMIC is able to maintain and enforce ICs in a single class (intra-class constraints) and also among distributed classes (inter-class constraints).
- Support multiple processors to maintain the constraints simultaneously when sets of objects or constraints of different relationships are completely independent from each other.
- The AMIC can be integrated with any existing or specialized constraint services.

©FCSIT, UM 2007
• The AMIC has an assertion language, which is called the OALIC that makes it able to declare the relationships among the attributes, constraints, and behaviors. Subsequently, the constraints derivation path can be presented easily.

8 Related Work

The proper handling of ICs is essential to any data storage and management. Handling ICs is an essential premise to managing semantically rich data [7]. In OODBs, checking the ICs is a fundamental problem in the database design [7]. The automated verification of constraints and their enforcement provided by current OODBMSs is limited [6] [7] due to the user participation is required. OODBMSs do not have adequate support for certain types of constraints especially the ones defined in a class composition and inherence hierarchies [1] [3] [5] [7]. The ICs must be maintained in the backward direction along the class hierarchies as well as in the forward direction. It seems to be no obstacles in extending the proposed model to deal with constraints.

Constraints are represented as SQL queries, hence are very difficult for inter-constraint maintenance. The inter-constraint maintenance problem and the contradiction or lack of proper functionality of a set of constraints is addressed in [7]. Also in [8] issue of a commercial semantic databases that extensively supports structural integrity enforcement and arbitrary constraint checking, is elaborated. Other work in [9] constraints has been done from the aspect of constraint satisfaction and constraint logic programming languages, where the emphasis is to use constraint propagation.

9 Conclusion and Future Work

Since the ICs are conditions that data within a database must satisfy, so database must have a set of activities that enforce integrity and maintain constraints to keep the database in a consistent state. The OALIC format and grammar facilitate the usage of the AMIC and make it competent to be used by any existing OODB. Also it can enforce the ICs for constraints with two operands that is not supported by the current OODBMSs. Consequently, the AMIC included the OMD and DM into the OODB, which helps to accelerate accessing and checking attribute values and their constraints.

The OMD has three classes namely: $\text{OMD}^{\text{CO}}$, $\text{OMD}^{\text{CK}}$, and $\text{OMD}^{\text{KB}}$, to keep track the constraint paths in the backward direction as well in the forward directions. The $\text{OMD}^{\text{CO}}$ has a special new technique that is built based on DAG to reduce coupling among attribute relationships and domains. The $\text{OMD}^{\text{CK}}$ keeps the constraint knowledge to ease accessing them. Furthermore, the $\text{OMD}^{\text{KB}}$ is designed to include knowledge about all attributes and their relationships, constraints, and domains by composing $\text{OMD}^{\text{CK}}$ that are associated with $\text{OMD}^{\text{CO}}$.

More optimization techniques can be developed for constraint compilation. OODBs face new challenges to semantic integrity especially to both constraint representation and constraint maintenance. More work can be done when copying an object of a superclass to another object of a subclass and vise versa. For such problem down-casting and slicing must be taken in account. Moreover, when a multiple inheritance occurs and the same attribute name existed in more than one superclass, then a virtual class is needed.

References


**BIOGRAPHY**

Belal Zaqaibeh received his BSc degree with the first honor degree in Computer Science from Irbid National University, Jordan, in 1998. In 1999, he was the manager of Makkah Center for Computer. In 2000, he continued to graduate school at Universiti Putra Malaysia (UPM) and received his MSc in Distributed Computing in 2001 and also his PhD in Object-Oriented Databases in 2006. During his enrollment, he was employed as a lecturer at the Multimedia University, Malaysia, where he did research on Object-Oriented Database. In 2006, he joined Zarqa Private University, Jordan, where he is currently working as an assistant professor of computer science. His research interests include object-oriented databases, mobile databases, integrity constraints, and object-oriented software engineering.

Hamidah Ibrahim is currently an associate professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She obtained her PhD in computer science from the University of Wales Cardiff, UK in 1998. Her current research interests include databases, transaction processing, and knowledge-based systems.
PERISA: A PERSONAL ISLAMIC ASSET MANAGEMENT SYSTEM USING OBJECT-ORIENTED APPROACH

Nor Fazlina Iryani Abdul Hamid1, Zarina Mohd Kasirun2
Faculty of Computer Science and Information Technology, University Malaya
50603, Kuala Lumpur, Malaysia
1fazlina@imrnd.com.my 2zarinahmk@um.edu.my

ABSTRACT

Many Islamic asset management systems have been implemented over the past few years in Malaysia, but none of them focuses on personal asset management. Furthermore, existing systems are developed with different management of different assets. There is no system that manages all types of Islamic assets in one single system. PERISA or Personal Islamic Asset Management System is a portal that integrates the personal management of Islamic assets. It is developed using object-oriented approach to address the current situation. Emphasising on various assets such as zakat, fidiah, heritage and waqaf, PERISA contributes to the management of personal Islamic assets as well as object-oriented software engineering research. Exploiting the Object-oriented Approach, PERISA is developed using Rational Unified Process (RUP) model in web-based environment. Usability test and software metrics measurement are carried out to measure the usability, reusability and complexity of PERISA.

Keywords: Islamic asset management, Object-oriented approach, Rational Unified Process, usability test, reusability and complexity measurement

1.0 INTRODUCTION

Recently, many systems related to Islamic asset management exist in Malaysia but most of them focus on specific category and do not cater all types of Islamic assets. For example, the zakat system or e-zakat which is being implemented by Pusat Zakat Selangor only focuses on zakat and fidyah. Amanah Raya Berhad on the other hand has a system to manage the distribution of harta pusaka while Jabatan Agama Islam has a system to manage all the funds and harta wakaf. Of all the systems mentioned above, only e-zakat is an interactive and informative web-based system, whereas the rest are only internal closed systems.

The existing systems exhibited some problems and weaknesses. They are as follows:

- Each system is separated according to the organization that handles it.
- The users have to go to different sites if they want to browse different type of assets.
- Only e-zakat has an interactive and informative web base system but it only manages zakat and fidyah.
- The closed systems which handle harta pusaka, wakaf and funds can only be accessed by the employees of the organizations. The public can only get these information from the officer, pamphlet or related books.
- The language used is Malay, which cannot be understood globally.

To overcome these problems, a system that has the information about all types of Islamic asset management is implemented. This system must be interactive and attractive as well as informative to attract users. This system must also be web-based so that it can be easily accessible via Internet anywhere and anytime. The system will be applicable to asset managers, professionals and also normal users who are interested with the Islamic asset management. English language has been chosen as the language to make the system globally understood.

This paper focused on system design, system development, system implementation and system evelopment using object-oriented approach. It is divided into several sections. Section 1 briefly introduces the problems and its background. Section 2 identifies the sources of Islamic assets together with the procedure to manage them. It is followed by understanding the way to manage the Islamic asset using object-oriented model in Section 3. Section 4 describes in details the implementation of PERISA. Section 5 highlights the measurements carried out and Section 6 wraps the paper.

2.0 SOURCE AND PROCEDURE OF MANAGING ISLAMIC ASSETS

From the Islamic point of view, assets can be classified into valuable assets, invaluable assets, static assets and dynamic assets. Valuable asset is every asset that is kept by someone and syara' [6] allows the usage of it. Valuable assets comprise of static and dynamic assets. An invaluable asset is an asset that is, not to be kept by anybody, such as fishes in the sea, birds on the air and minerals inside the earth. A static asset is an asset that cannot be moved from its location.
like buildings and houses. A dynamic asset is the asset that can be moved or restructured whether its value or its physical such as money and business properties. This paper focused on four types of Islamic assets, which are zakat, fidiah, heritage and waqaf.

There are two types of zakat, which are Zakat on Properties and Zakat Fitrah. The way of calculating the Zakat on the Properties is different from one asset to another. The amount of zakat due on these types of wealth is 2.5% of their total values. However, this rate is different for other types of wealth such as agricultural products, livestock and natural resources. For Zakat Fitrah the amount to be given is 1 cubic measure that is equivalent to 2.3 kilograms of staple food of a particular country or the equivalent value of money.

In Malaysia, Fidiah is paid using rice or money for those who cannot fasting for some reason based on syara’ or missed to replace the fasting days after the year lapse. The amount of the rice is about ¾ kilogram per day or equivalent of cash and the amount is accumulated. The amount of rice or money with equivalent value will accumulate until she replaces her fast.

Amanah Raya Berhad plays its role as the heritage asset administrator. Amanah Raya Berhad will manage all assets including the money in saving account, employee provident fund (EPF), retirement allowance, compensation and so on. Amanah Raya Berhad also identifies the heir of the asset, their portion and method of distribution.

Nowadays, in Malaysia, Jabatan Agama Islam controls the waqaf properties. In Islam, Waqaf has the high value and is viewed in three dimensions. Firstly, it gives multiple merits to the people who give waqaf their property. Secondly, it helps the poor people. Thirdly, it upgrades the image of Islam.

3.0 DEVELOPMENT of PERISA USING OBJECT-ORIENTED APPROACH

The PERISA has been developed using object-oriented approach by implementing Rational Unified Process (RUP) model. Using the RUP, software product lifecycles are broken into four main phases that are termed as inception, elaboration, construction and transition phase [7].

Inception phase is the process where it analyses the depth and breadth of any architectural prototype that was developed by literature review and requirement understanding process through data collection and analysis. Elaboration phase is where the project started to take shape. In this phase the problem domain analysis has been made and the architecture of the project gets its basic form. It elaborates on the functional and non-functional requirements of the system based on the interview and the literature review in several related web sites. From the functional requirement, the system use cases (refer to Figure 1) are developed and the actors of the system are identified. Elaboration phase also described in the system architecture and development plan for the system that elaborates the system design starting from system design and modelling, database design up to interface design. The system design has been supported by layered architecture, class diagram (refer to Figure 2), sequence diagram and entity relational diagram to illustrate the real system.

Construction phase focuses on the development of components and other features of the system. This is the phase where the bulk of the coding takes place. This phase produces the first external release of the software. This is the phase where system has been implemented. The system functionality is based on classes and methods as defined in elaboration phase. In transition phase the system has moved from software development organization to the end user. The activities in this phase include the system evaluation and system measurement. Usability test is a test to measure user’s satisfaction of the system and their comments are analyzed in order to improve the system. It will cover the testing procedure such as evaluation survey and survey outcome analysis. The results of the survey are presented using graphs and will be discussed. System measurement is the measurement of the system using object-oriented metric. This system has been measured Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT) and Number Of Children (NOC).
4.0 IMPLEMENTATION OF PERISA

PERISA is developed using PHP language and MySQL database. It consists of eight tables, which are profile, archive, zakat, fidiah, heritage, heir, counter and waqaf table. This system is tested and able to run under windows and Linux environments. For client, since the system is a web-based application, it is able to be opened from any environment as long as there is an Internet connection and a web browser.

This system consists of nine modules, which are general information, admin, profile, archive, registration, zakat, fidiah, heritage and waqaf. General Information module can be accessed by public where it gives the brief information about personal Islamic asset management including kifayah calculator. Kifayah calculator is a calculator to indicate whether a particular person is considered as fuqaraa’ or maskeen who are eligible to receive zakat and fidiah. Admin module can only be accessed by the system administrator to manage the system. Profile module can be accessed by all registered...
users to update their own profiles. Archive module is the module where the registered users can search for the related articles. Registration module is the module to enable publics to register as a member to the system. Upon successful registration, the system produces username and password. Zakat module is a module to manage zakat, which consists of zakat details information, zakat calculator, personal zakat organizer and list of zakat counters. Fidiah module is a module to manage fidiah, which consists of fidiah details information, fidiah calculator, personal fidiah organizer and list of fidiah counters. Heritage module is a module to manage heritage, which consists of heritage details information, heritage calculator, personal heritage organizer and list of heritage counters. Waqaf module is a module to manage waqaf, which consists of waqaf details information and list of waqaf counters. As example, interface for fidiah calculator module is as Figure 3.

![GUI for Fidiah Calculator](image)

**Fig. 3: GUI for Fidiah Calculator**

### 5.0 Usability and Metrics of PERISA

PERISA is evaluated in terms of its usability as well as its reusability and complexity. The usability test is used to measure if PERISA has met user’s requirements while the reusability and complexity measures the PERISA’s maintenance. The usability test covers ease of use, functionality of each module, overall functionality and potential effectiveness of PERISA.

At this stage, the system usability test verifies that the whole system is functioning properly and several end users are identified for the test. It also ascertained that the objectives and requirements are fulfilled. In order to make the test more credible and reliable, the program testers are randomly chosen and ranged from diploma to master degree holders.

![Potential Effectiveness of Personal Islamic Asset Management System using Object-oriented Approach](image)

**Fig. 4: Potential Effectiveness of PERISA Graph.**

From the usability test, it shows that the system is user friendly and easy to use for various background of users. For the system functionality of each module, most of the respondent agree that the system is well functioning. In overall functionality of the system, most of the respondents rate fairly good and up to their expectation. In potential effectiveness of the system, most of them rates likely to be greatly improved, which shows that the system has a great
potential in improving on how individual manages their personal Islamic assets. Graph in Figure 4 shows the potential effectiveness of PERISA.

The reusability and complexity measurements are carried out using object-oriented metrics. In the beginning of 1990s, Chidamber and Kemerer, proposed six new object-oriented metrics to overcome the limitations of more traditional code-based metrics [9]. As far as PERISA is concerned, it will only be tested using Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT) and Number Of Children (NOC).

Weighted Method per Class (WMC) is defined as the sum of the complexities of all methods of a class [10]. The number of methods and the complexity of methods involved is a predictor of how much time and effort is required to develop and maintain the class. The formula of Weighted Method per Class (WMC) is as below:

\[
\text{Total number of methods} = \frac{\text{Average number of method per object class}}{\text{Total number of object classes}}
\]

Hence, the Weighted Method per Class (WMC) for Personal Islamic Asset Management system using Object-oriented Approach is as follows:

Total number of method = 37
Total number object classes = 20
Average number of method per object class = \(\frac{37}{20} = 1.85\).

Therefore, the Weighted Method per Class (WMC) for PERISA is 1.85. The Weighted Method per Class (WMC) for the system is small, so it reduces the complexity and easy to be extended.

The next measurement metrics is Depth of Inheritance Tree (DIT) that defined as maximum length from the node to the root of the tree [10]. The deeper a class in hierarchy, the greater the number of method it is likely to inherit, making it more complex to predict its behaviour. The formula of Depth of Inheritance Tree (DIT) is as below:

\[
\text{Inheritance tree depth} = \max (\text{inheritance tree path length})
\]

In Personal Islamic Asset management System using Object-oriented Approach there is only one inheritance relationship as figure 5:

![Inheritance Relationship](image)

**Fig. 5: Inheritance Relationship**

Thus, the Depth of Inheritance Tree (DIT) for PERISA is 2 because the maximum inheritance tree path length of the system is 2. The Depth of Inheritance Tree (DIT) for the system is small, so it does not have greater method sharing, easy to test and comprehensive.

The final measurement metric for the system is Number Of Children (NOC), which defined as a number of immediate subclasses [10]. The greater number of children, the greater the reuse, since inheritance is a form of reuse. Consequently, the Number Of Children (NOC) of PERISA is 3 as shown by Figure 5. The Number of Children (NOC) of the system is considered small, so the reusability is also small, but it has the proper abstraction of the parent class.
6.0 CONCLUSION AND FURTHER WORKS

This paper has described the development of PERISA using object-oriented approach. It shows that the approach is suitable to be used from the analysis until the evaluation stage. As far as the contribution, PERISA is capable of managing user’s personal Islamic asset such as zakat, fidiah, heritage and waqaf. In addition, it also gives detailed information about these assets and allows user to calculate zakat, fidiah, heritage as well as kifayah if they are entitled to. Based on the user evaluation, the features of PERISA can still be improved by integrating online payment, online registration for zakat and fidiah receiver, online financial report, and online asset registration for waqaf.

7.0 REFERENCES


8.0 BIOGRAPHY

Nor Fazlina Iryani Abdul Hamid just finished her Master of Software Engineering from University Malaya. Currently, she is a researcher at the TM Research and Development, a subsidiary of TM. Her research areas include object-oriented approach in developing web-based application and managing assets in Islamic way.

Zarinah Mohd. Kasirun is a lecturer at the Faculty of Computer Science and Information Technology, University of Malaya. Her research areas include Requirements Engineering, Computer-Supported Collaborative Learning, Object-Oriented Analysis and Design.

©FCSIT, UM 2007
REALISING THE MYCL PROCESS MODEL THROUGH THE DEVELOPMENT OF A COMPONENT-ORIENTED SPECIFICATION LANGUAGE (COSL) COMPILER

Hazleen Aris¹, Siti Salwah Salim²
¹College of Information Technology, Universiti Tenaga Nasional, 43009 Kajang, Selangor, Malaysia. Email: hazleen@uniten.edu.my
²Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia. Email: salwa@um.edu.my

ABSTRACT

This paper describes our approach to the construction of a compiler for component-oriented specification language (COSL). The compiler is part of the component engineering subprocess of the MyCL process model; a simplified component-oriented software development (COSD) process model that we proposed from our previous research after reviewing the impracticality causes of the existing COSD process models. The MyCL process model was evaluated, and the evaluation results indicate a number of improvements that can be made to ensure smooth application of the process model. One such improvement is by having a dedicated specification language for COSD because most of the COSD at present depends largely on object-oriented programming language. A dedicated specification language for COSD no doubt requires a new compiler due to differences in its syntaxes, semantics and et cetera. The existence of the compiler will bring the current component technology closer to its requirements of being independent of any vendor and available for composition in executable form.

Keywords: component-oriented, process model, compiler construction.

INTRODUCTION

Application development through components composition, known as component-oriented software development (COSD) is gaining more and more attention lately. In order to keep pace with this growing interest in COSD, a number of COSD process models have been proposed. However, not many of these COSD process models are being applied in the software development industries as reported in [1]. Therefore, in our previous study, we had performed a review of the existing COSD process models and came out with a list of impracticality causes that hinder these models from being applied in the software development industries. As a result, we introduced a simplified COSD process model called the MyCL process model. The MyCL process model was constructed by overcoming the impracticality causes of the existing COSD process models while retaining their strengths [1], [2]. There are eight processes that constitute the MyCL process model. These processes are requirement analysis, domain engineering, architectural design, component development, component testing, component composition, system testing and component updates, which will be elaborated in the next section.

Evaluation of the MyCL process model through the use of the process model in coming out with an application discovers a number problems associated with the current approach to COSD. First of all, it is still tied to the object-oriented programming language at the implementation stage. Secondly, software developers who would like develop application using this methodology is also required to have a significant amount of skill in programming language. Finally, the components available are only composable using the component framework where they come from. These problems, if not treated, will hamper the successful application of COSD in software development. This paper therefore suggests one possible approach to overcome these problems. We propose the construction of a compiler to compile components written using a component-oriented specification language (COSL). The outcome of this compilation process is the corresponding components in target language that can be assembled and linked to produce executable components, which are composable regardless of component frameworks.

Section 0 of this paper summarises the MyCL process model, focusing on the component development process and describing the problems discovered after evaluating the process model, which leads to the need of the compiler. A detailed description of the formulation and evaluation of the MyCL process model is reported in [1], [2], and [3]. The position of the compiler with respect to the MyCL process model is explained in section 0. Section 0 describes our approach towards the construction of the compiler and section 0 concludes the paper.
As mentioned above, there are eight processes that constitute the MyCL process model. These processes, with the exception of component development process, are represented by rectangles as shown in Fig. 1. The component development process is represented by a frame as it contains a number of subprocesses. Requirement analysis process analyses the requirements received from user to produce component specification. In this process, cooperation from a group of domain engineers is required to identify possible reuse of existing components from the repository through the domain engineering process. Domain engineering process also produces design models that are fed into the architectural design process and a group of specific languages, application generators and candidate components that are fed into the component development process.

The outcomes of the component development process are individual composable components. Each component is tested in the component testing process prior to composition by the component composition process. The composed application is tested in the testing process before an application software is produced. Any changes to the existing components and the creation of new components resulting from the component development process are updated by the component updates process before being deposited into the repository for future use.

Evaluation of the MyCL Process Model

The MyCL process model was evaluated using two evaluation methods; first, through the structured interview sessions with the experts and second, through the application of the process model in coming out with a component-oriented application. These evaluation methods were chosen due to the constraints in getting suitable software developers to evaluate the process model [3], [4]. The former evaluation method results in a number of improvements made on the process model, which, among others is the expansion of the component development process to expose the component selection, component adaptation and component engineering subprocesses as can be seen from Fig. 1. A complete list of improvements made can be found in [3].

The latter evaluation method applies the process model in coming out with an ‘online pharmacy store’ application as explained in [5] with the aim of demonstrating the simplicity of the process model. On top of the online pharmacy store application produced, the use of the process model has also discovered the following problems, which are not specific to the MyCL process model, but also true to most of the existing COSD process models. These problems are:
Dependency on object-oriented programming language. The unavailability of a programming language for COSD forces the component development and composition processes of the MyCL process model to adopt the current object-oriented programming language. This is also the case for most of the existing COSD process models.

Dependency on programming language skill. The notion of COSD as a ‘plug-and-play’ approach in developing software is still far reaching. Using the currently available component frameworks, software developers are still required to have a significant amount of knowledge on programming language.

Dependency on component framework. This is another challenge of COSD. Components offered by a specific component framework thus far are only meant for composition using the framework. An ideal component should be composable regardless of any framework.

To overcome these problems, a number of suggestions for future work have been recommended [4]. These suggestions, amongst others, include:

- the invention of a dedicated programming language for COSD
- the creation of a self-explanatory executable component and
- the creation of rules that will allow component composition across various component frameworks

The invention of a dedicated programming language for COSD is needed because most of the available component frameworks are still relying on the object-oriented programming language in the implementation stage, such as the Expresso framework [6]. The use of object-oriented programming language in COSD does not seem to be a good solution as a number of fundamental concepts and ideas in COSD have to be compromised to enable the use of the object-oriented programming language. Such action will hamper the progress towards COSD as some of its foundations will have to be abandoned. A number of work has been done on the specification of dedicated programming languages for COSD. Notably the component definition language (CDL) [7], component composition language (CoCo) [8] and construction and composition language (CCL) [9]. Therefore, the obvious need now is to turn these component-oriented specification languages into executable components that are composable across various component frameworks in an effort to fulfil the requirements of a component stated above.

**positioning the compiler**

Even though a number of component-oriented specification languages have been proposed, most of the proposals do not continue to the implementation stage. Even if they do, the resulting components are only composable using a specific frameworks where they come from. Thus, the challenge here is to have a compiler that compiles a component-oriented specification language and produce cross-framework executable components. Since compilation is part of the component development process in the MyCL process model, we first describe the component development process in the next subsection and introduce the compiler in section 0.

**Component Development Process**

The component development process of the MyCL process model is further divided into three subprocesses; component selection, component adaptation and component engineering as can be seen from Fig. 1. The component development process begins by first selecting the appropriate components from the list of components short-listed by the domain engineering process. This basically means browsing through the component repository of the chosen component-based framework to find the suitable components for use. If these components are already sufficient to fulfil the system requirements, they can be composed straightaway to produce a working system. If some kinds of modifications are required in order for the components to perform the intended functions, the selected components that require modifications will be identified for component adaptation subprocess.

Component adaptation usually requires the writing of component wrapping; a locally developed code that masks a particular component’s unwanted and incompatible behaviour [10]. If the selected and adapted components are already sufficient for the development of the new application, the component development process ends here and the component composition process can take place. Otherwise it will continue to the component engineering subprocess where new components will be created, with the consideration of making them reusable in later projects.

Component testing process tests the individual components prior to composition. It is especially required for the engineered components. Once tested, the selected, adapted and engineered components will be assembled for composition and that concludes the component development process. Successful application of the component development process described in this subsection is subject to the components used meeting the requirements of a true
component [11]; independent of any vendor and available in executable form. These requirements, as far as the current component technologies are concerned, are still the subject of research.

Compilation as Part of Component Engineering Subprocess

Fig. 2 below refines the component engineering subprocess to consist of the following elements; preprocessor, compiler, assembler and linker. Preprocessor is responsible for providing input to the compiler, which is the source language to be compiled. Compiler, by definition, is a program that reads a program written in one language and translates it into an equivalent program in another language [12]. In this case, it takes in the source language produced by the preprocessor and produces an equivalent component in the target language for the assembler. The compiler element is shaded in Fig. 2 to indicate the focus of this research on compiler construction. The assembler further processes the target language, which is normally produced as assembly code. It translates the assembly language into relocatable machine code that can be passed directly to the linker.

Linker performs the two functions of loading and linking. The process of loading consists of taking relocatable machine code, altering the relocatable addresses and placing the altered instructions and data in memory at the proper locations. The linking process allows us to make a single program from several files of relocatable machine code. These files may have been the result of several different compilations, and one or more may be library files of routines provided by the system and available to any program that needs them.

A proposed approach to the compiler construction

We propose the approach shown in Fig. 3 to construct the compiler. This approach begins with the selection of a component model to be used. We recommend selecting one of the existing component models because there are already many of them being researched and made available. After deciding on a component model to use, there is an option to either use an existing component-oriented specification language (COSL) or develop a new COSL. Most of the proposed component models come together with their proposed COSL. In our case, we have decided to use the COSL that comes together with the selected component model so that we can concentrate our research effort on the construction of the compiler. As mentioned before, preprocessor is responsible for the task of collecting the source program to produce source language statements to be fed to the compiler. In our proposed approach, the preprocessing covers the task of writing components using the chosen COSL. Therefore, as an alternative, this process can also be called ‘coding’.

The first three phases of the compilation process concerns with analysing the source language, i.e. the components written using the chosen COSL. We make the source language more specific by using the term ‘source component code’ in Fig. 3 and from this point onwards. In compilation, source component code analysis consists of three phases;
lexical analysis, syntax analysis and semantic analysis. Lexical analysis is also known as scanning. In lexical analysis, the stream of characters making up the source component code is read from left-to-right and grouped into tokens, i.e. sequences of characters having a collective meaning. The blanks separating the characters or these tokens would normally be eliminated during lexical analysis. Syntax analysis is a phase where characters of tokens from the source component code are grouped hierarchically into grammatical phrases with collective meaning. The grammatical phrases of the source component code are usually represented by a parse tree. Another name for this phase is parsing. Semantic analysis phase is responsible to check that the grammatical phrases of the source component code written fit together meaningfully. Other than semantic checking, it also gathers type information for the subsequent intermediate code generation phase. An important task of semantic analysis is type checking, where it checks that each operator has operands that are permitted by the chosen COSL specification.

After the analysis phases, the compiler, through its intermediate code generator generates an explicit intermediate representation of the source program. Two important properties that this intermediate representation should have are easy to produce and easy to translate into the target language, which, in this case are the equivalent components in independent assembly language. The code optimisation phase improves the intermediate code produced where possible, resulting in faster-running machine code. The final phase of the compilation process is the generation of the target code by the code generator. The target code normally consists of relocatable machine code or assembly code. Intermediate instructions are each translated into a sequence of machine instructions that perform the corresponding task.

Other than the selection of component model to be used and the selection or development of the COSL, our proposed approach is by far generic to other specification languages. Refinement of the approach, involving the addition or modification of the phases in the compilation process will be made as this research progresses and more details on COSL chosen are discovered.

![Fig. 3: COSL compiler construction approach](image-url)
As can be seen from Fig. 3, two other activities are interacting with the six phases of compilation process; symbol-table manager and error handler. A symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier. When an identifier in the source component is detected by the lexical analyser, the identifier is entered into the symbol table. The remaining phases also enter information about identifiers into the symbol table and then use this information in various ways. Each phase of the compilation process can possible encounter errors, albeit of different types. A phase should somehow deal with that error by means of an error handler, so that compilation can proceed to allow further errors in the source component to be detected at the remaining phases.

**Conclusion**

This paper first provides the summary of the MyCL process model, a simplified process model for COSD with the focus given to the component development process. Results from the evaluation performed on the process model show a number of problems associated with it, which are also true with other COSD process models. In an effort to overcome the problems, we have proposed the construction of a compiler to compile component-oriented specification language into composable executable components. The compiler is part of the component engineering subprocess of the MyCL process model. Our approach in constructing the compiler is described towards the end of this paper. The existence of the compiler will bring the component technology one step closer to meeting the requirements of a component.

**Acknowledgement**

This research is partly funded by the University of Malaya under the ‘Peruntukan Penyelidikan Pascasiswa (PPP)’ grant number P0278/2007A.

**REFERENCES**


BIOGRAPHY

Hazleen Aris is currently holding the position as a lecturer at the Department of Computer Science, College of Information Technology, Universiti Tenaga Nasional. She obtained her Master in Software Engineering from the University of Malaya, Malaysia and Bachelor of Computer Engineering (Hons) from the University of Southampton, UK. Her current research interests include the development of component-oriented programming (COP) language and its compiler and components composition at a higher level of abstraction.

Siti Salwah Salim is currently a Professor at the Faculty of Computer Science and Information Technology, University of Malaya. She obtained her PhD in Computer Science from the University of Manchester Institute of Science and Technology (UMIST), United Kingdom. Her current research interests include computer supported collaborative learning, human computer interaction, software requirements engineering and animated pedagogical agents.
SOFTWARE VISUALIZATION TOOL TO SUPPORT FOR OUTSOURCED SOFTWARE MAINTENANCE

Eswaran Narasimhan 1, Udayan Banerjee2

1Technology Innovation Centre, NIIT Technologies Ltd., Chennai, INDIA
Email – eswaran.narasimhan@niit-tech.com
2Technology Innovation Centre, NIIT Technologies Ltd,
31/2 Hosur Road Bangalore, India. Email: udayan.banerjee@niit-tech.com

ABSTRACT

With increasing outsourcing of software code maintenance and re-engineering some of the most difficult issues faced by the outsourcing companies and those who execute them are in the area of program comprehension. They need different programming languages and the cycle-time available is always very short. A flexible software visualization tool is needed to accomplish the re-engineering. To aid this process we created a software visualization tool called INSIGHT. It is programmable and has a flexible architecture, so that it can be adapted to any programming language and any form of visualization.

Keywords: Program Comprehension, Program Querying, Reengineering.

1. INTRODUCTION

There has been increased outsourcing of software program maintenance in the recent years. Those who undertake the maintenance task face problems in managing the process of comprehending the program. There is pressure to comprehend large amount of code in a short time. The complexity of the exercise is further compounded by the absence of updated documentation and skill sets on the existing code. The projects are further characterized by new team members in the initial phases of the project and a relative short time available for becoming a productive contributor towards the project.

These comprehension requirements may begin with a simple meaningful stocktaking of the code inventory which may run into millions of line. Subsequent activities involve fixing bugs in the program and enhancing the programs based on the change requests made by the user. The change requests may arise from business enhancements, field expansions and changes to computational logic etc. Typically these activities are carried out by a team who has not taken any part in the actual development of the programs and hence they would not be aware of the internal structure of the program and would not have formed a mental model of the programs. This is very critical to ensure that changes do not break the intended behavior of the system [5]. The problem is compounded by the fact that there is usually a service level agreement in place which necessitates a response to individual problem within a specified period of time.

Therefore, it is imperative that pre-work on comprehension needs to be carried out in advance and recorded in form of documents which can be searched when the need arises. This process of preparing the documentation from code is nothing but software reverse engineering and it can be defined as “the process of analyzing a subject system to identify the system’s components and their relationships, and to create representations of the system in another form or at a higher level of abstraction” [1]. The goal of this exercise is to construct a mental model and good software visualization tool is a powerful asset in the building of such a mental model [2], [3], [4]. One important point that needs to be kept in mind is that very often a team member who uses these documentations will be different from the person who has prepared it.

The team member who uses the existing model often has to add annotations to make the visualization relevant to the current situation. Sometimes there is a need to hide unnecessary details to improve the visualization at a higher level. The annotations thus evolve and they become richer as the project progresses.

When an actual change that needs to be performed, the first question which needs to be answered is “among the millions of lines of code, where do I start?” This precedes the exercise of forming the mental model and the documentation should be able to guide the programmer. To answer this question link needs to be established between the documentation that describes the functionality from the top and the documentations created by reverse engineering the program. Hence the visualization tool should have facility to establish hyperlinks among documents and between document and program. After the programmer has located where the change needs to be made, they will explore the code and would like to navigate through relationships. During this exploration process, also called “Just in Time...
Comprehension” [9], a mental map of the visualized information is constructed [12]. This will help in deciding what changes need to be made. The next question that needs to be answered is “what else will be affected if I make these changes?” This can partially be answered through easy navigation which is an important in any software comprehension tool [11].

The languages that had to be addressed by the visualization tool would vary from project to project. A large variety of languages needed to be covered. Not only is it necessary to handle typical legacy languages like COBOL, RPG and PL/I, a wide range of other languages needs to be handled. These includes second generation language like Assembler and TPF, 4GL like Power Builder, Delphi Pascal and special languages like JCL, CICS Maps, and QTP testing scripts.

The best way to visualize a program in a specific language will vary. What is best for a second generation language may be quiet inappropriate for an object oriented language. So, the visualization tool that we needed not only had to support any language of our choice, but also support any form of visualization that we may need. We could not locate a tool which satisfied our entire requirement. So, decided to write our own tool and called INSIGHT.

In this paper we present the design considerations and the architecture of our software visualization tool that we built. We also share the current status of the tool including its limitations and the future enhancements that we intend to undertake.

2. DESIGN CONSIDERATIONS

The main characteristics of the tool can be stated in form of a wish list:

- Ability to support multiple languages
- Extendable to include support for any language encountered in the future
- Support multiple forms of visualization
- Support form of visualization suitable for each language
- Extendable to include support for any form of visualization
- To have ability to edit a specific visualization to
  - Include annotation and explanation and hide details where needed
- Facility for hyper linking and navigation
- Every visualization exportable into standard formats like
  - Word document, HTML file and Visio diagram

The tool had to be easily deployable in different project situation. We wanted to distribute only the executable. This implied that it should be possible to add a new language and a new form of visualization without making any changes to the core tool. Therefore, the tool had to be programmable and we needed a scripting engine. We looked at different scripting engines and chose the VBA scripting engine of Microsoft and by extension Microsoft Visual Basic as the platform for the tool. We also needed to have intermediate representation of the source and a data structure to store a specific visualization. XML was chosen because the advantages it offered. The structure is flexible, extensible and also read to understand. It is also easy to process. We needed a repository and decided to use MS-Access for single user implementation and MS-SQL Server database for a multi user implementation.

3. ARCHITECTURE

To reduce complexity we decided to partition the tool into three parts which are loosely coupled where interaction between them should take place between them through a repository. The repository would contain the code, parsed metadata which we will refer to as “Parsed XML”, and structured representation suitable for each type of visualization, which we will refer to as “Visualization XML”.
3.1. The Parsing Module

The speed with which we could adopt to a new language was an important criterion. When we encounter a new language we are likely to face the following situations:

- Time may not be available to build a complete parser
- Full language syntax may not be available at the start of the exercise
- Top level comprehension may be required very quickly

We did not attempt a complete language parsing, as our code set never used all the features of a language. As we often needed a top level comprehension very quickly, we resorted to an incremental parsing approach, thus building a parser that could be extended as required. We decided to build a heuristic parsing engine which follows the principle of parse as much as needed. The parsing engine provides an environment for iterative definition of rules for parsing the source code. It also provides a facility for step by step execution of the rules. Rules can be written to operate on the specific pattern so that the source code can be sliced accordingly. The parsing module has the following behavior.

- A Rule can consist of multiple commands
- The main commands are:
  - Locate and Register a Token; and a Matching Token
  - Register a Statement bounded by two tokens
  - Register an Attribute
- The base rule will apply on the whole source
- Once a statement is registered, different set of rules can be applied to it
- Multiple rules can be applied to one type of statement
- Each command advances the current position within the statement
- Commands are available for manipulating the location of the current position
- A rule can be repeatedly applied on the same statement till any command in the rule fails
- When a rule is repeated, it starts from the current position
- Column limit for the source can be set so that sequence numbers can be ignored
- Once a token / statement / attribute is registered, it is ignored by subsequent rules

Each of the rules is written in a VB Script where each command is a function call. Since we used VBA scripting engine, complete procedural programming facility was also available. The parsed code gets represented as Parsed XML is stored in the repository. The tool has visualization for the Parsed XML structure which is shown below.
3.2. Degree of Parsing

In our approach, it is not necessary to parse the program to the last detail. The depth of parsing that need to be carried out will depend on questions regarding the code that needs to be answered. While the basic structure of nested statements, usage of tokens and usage of attributes to punctuate the description of statements remains same for any program, the degree of nesting and the actual schema of the structure will depend on the language and the degree of comprehension expected for the purpose of the exercise under discussion.

For example when the requirement is to estimate the program complexity, it is not necessary to do a complete parsing. We may only parse the program to the extent of identifying the types of statements that we may be interested in. It may be restricted to identifying the control structures, input-output statements etc. It may not be necessary to parse individual expressions. This will probably be sufficient to arrive at the program complexity.

3.3. The Processing Module

The processing module was designed to take processing instructions on the parsed code in the repository and create more intermediate internal models that the visualization engines can use to render the visualization artifacts. The processing engine incorporates a facility to program in VBA procedure that can operate on the parsed code and create new relationship to be used for further processing.

The processing module is the indirect link between the parsing module and the visualization module. It is programmable and procedures can be written in VBA to create the Visualization XML from theParsed XML. In addition it has a querying sub-module which allows a procedural definition of a query which operates on the Parsed XML and creates a list of notes which meets the selection criteria. The querying sub-module supports simple query which can select XML nodes based on a selection criteria. The criteria can be build using different operator the statement, attribute and token. The operators can be Begins with, Contains, Does not exist, Equals, Exists, In list, Match pattern, Not in list. In addition compound queries can be created. Set operation like Union, Intersection and Not in is also possible on the result.

In the processing module, procedures can be written in VBA. Functions are provided which can initiate a specific query. Access to the query result is also provided. In addition direct access to the Parsed XML is also possible. Similarly functions are provided which can be used to construct different Visualization XML. Since the basic scripting engine is VBA, full VB Scripting language support is available.

3.4. The Visual Rendering Module

The visual rendering module has two primary functions 1) to render the Visual XML and 2) to allow modification to the visualization. It has facility to create diagrams as well as reports. The report can be stored as an HTML and it can be
exported to a word document. All diagrams can be copied to a word document and it can be exported as a Visio diagram. Like all the other modules in INSIGHT, this module is also programmable and any type of visualization can be created. Here we give example of two diagrams and one textual report.

**Flow Charts:** One of the useful visualization techniques for an assembler program is a flow chart. The programs are usually long and tracing a coherent path is difficult. The flowcharts are made up of many traces. A trace is a set of lines represented by their numbers that participate in the flowchart segment. A trace can be a set of sequential lines or a branch. A branch is a jump from one line to another.

The intermediate representation maintains the traces and the branches. A set of such traces form a flowchart. A procedure can detect sequential lines in an assembler program and also map out the traces defined by the Branch instruction. Once the traces are created another procedure translates them to a flowchart using the line numbers and the branch details. While seeing the flowchart, clicking on a flowchart segment automatically highlights the corresponding segment in the source code for better understanding. Flowcharts can be raw and without comments. When comments are added by knowledgeable users then the same can be stored and a new flowchart can be created with comments for better understanding by others. The user can click on the flowchart to open the source code at the corresponding line so that he can see the code and its context.

Using API calls provided by Visio, the drawing was exported in a Visio format. This format can be saved and opened in Visio allowing further annotations and usage.

**Structure chart:** A COBOL program makes more meaning when seen as a structure chart. The intermediate form stores the calling paragraph name and the called paragraph names. These names are extracted from the PERFORM statement in a given paragraph. Once this is done, the relationships are shown as a structure chart. Once done for COBOL extending this for PL/I are easy. The way the paragraphs are executed in PL/I are different. Replacing the logic for detecting calls within a program automatically created relationships for PL/I and structure chart for PL/I is ready with little additional effort. Here too comments can be added to structure chart and seen. Using API calls provided by Visio, the drawing was exported in a Visio format. This format can be saved and opened in Visio allowing further annotations and usage. The user can navigate between the structure chart and the relevant source code.

**Data Cross Reference:** More complex visualizations can be achieved through procedural control over XML structure. Procedures allow set operations on the XML to produce the desired output. Linkages for called programs, included files and copy books can be set for producing statistical reports such as where-used reports. In the adjoining figure a data cross reference report has been generated. This is an example of a nested report. For example, the variable DATE21N defined in line 138 is used in lines 131 and 141. This cross references can then be used to navigate through the source for easy visual understanding.

This hyper linking on the sources also helps in visually seeing the code in another dimension. The intermediate storage stores the linked line numbers. These line numbers along with their content can be displayed when ever the user click on a line and wishes to see related lines. This relationship is produced by first generating a report of related items along with their line numbers. One this is done the line numbers are then used to annotate the line numbers in the XML document.

©FCSIT, UM 2007
4. Related Work

Software comprehension is the most costly activity in the software development life cycle [13], [14], [15] and comprehension support is needed for software maintenance [15]. Many tools have already been developed to support the understanding of software systems. Some of them focus on the visualization facilities. Three well-known examples are Rigi [18], SHriMP [8], [19] and CodeCrawler [7]. The other category tools like JQuery [16], tksee [9] and SEXTANT [17] focuses on software exploration provide sophisticated support for navigating along relationships between the different elements of software. However, each one of these tools addresses a specific class of language.

The focus of INSIGHT is on flexibility and programmability. The environment in which the tool has been deployed requires it to be easily extendable to cover new languages and visualization. It has the ability to handle all types of languages starting from second generation to the latest object oriented languages. It can also be extended to handle all types of non-procedural and scripting languages.

For developing the tool, simplicity of programming and rule writing has taken precedence over efficiency. So, some of the activities which with efficient algorithm can be achieved in seconds may require minutes in this tool. However, in the overall context, the execution time has been acceptable. With the current rate of increase in processing power, lack of efficiency is not likely to be critical.

5. Summary and Future Work

A large degree of flexibility is required from the software visualization tools [6]. Our intention for creating INSIGHT was to have the flexibility. Though the basic architecture is in place, we need to go a long way before we can claim that INSIGHT is truly generic in nature. We intend to extend INSIGHT to cater to more methods of visualizations.

We have covered a large range of legacy languages including Assembler, COBOL, PL/1, RPG and Pascal. We have also developed parsers for scripting language like QTP. The time taken to develop the rules has shown a declining trend for each new language taken up. This can be attributed to maturing of the parsing engine as well as to our ability to reuse rules across languages. Rule development for a new language is not a bottleneck any more.

The languages that we have chosen has been driven by organizational need. So far we have not attempted any comprehension exercise on an object oriented language or the modern scripting languages. To comprehending these languages we will need to create different forms of visualizations suitable for such languages. Since the architecture of the tool is flexible, it is very much possible and it is part of our future work plan.

Another area that we need to work on is to make the visualization engine more generic. This will include looking at the usability issues and providing easier editing features. Providing navigation feature between a manually created document and a specific visualization is also a pending activity.

REFERENCES


**BIOGRAPHY**

Eswaran Narasimhan obtained his Bachelor of Engineering degree from Indian Institute of Technology, Kharagpur, India in 1981. Currently, he is the Lead Architect at Technology Innovation Centre in NIIT Technologies Ltd., India. His research areas include Model Driven Software Engineering, Knowledge Management and Code comprehension.

Udayan Banerjee obtained his Bachelor of Engineering degree from Indian Institute of Technology, Kanpur, India in 1977. Currently, he is the Vice President and head of Technology Innovation Centre in NIIT Technologies Ltd., India. His research areas include Model Driven Software Engineering, Service Oriented Architecture, Web 2.0, Knowledge Management and Code comprehension. He has published papers in some of these areas. He also has a pending patent application in the area of Model Driven Software Engineering.
TOWARDS A HIGHLY EXTENSIBLE AND FLEXIBLE MIDDLEWARE FOR DATA MINING

Lai Ee Hen¹, Sai Peck Lee²
¹Department of Software Engineering
Faculty of Computer Science & Information Technology
University Malaya. Email: laiee@perdana.um.edu.my
²Department of Software Engineering
Faculty of Computer Science & Information Technology
University Malaya. Email: saipeck@um.edu.my

ABSTRACT

Information are critical assets to an organization. In order to uncover useful information, data mining tools are used to help the organization extract any hidden patterns to assist in decision making. However, software development is always in the process of evolution in order to keep up with the current dynamic environment. Thus, supporting a predefined set of data mining techniques, data sources and reporting formats is insufficient due to frequent change of requirements and expectation from business users. In this paper, we propose the architecture of a middleware for data mining which adopts plugins mechanism to help this so-called data mining middleware achieve extensibility and flexibility by providing wide spectrum of data mining techniques, data sources and reports.

Keywords: Data Mining, Middleware, Extensible Plugin, Customizable Plugin.

1.0 INTRODUCTION

Knowledge Discovery in Databases (KDD) is defined as a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [2]. Data mining is one of the processes in KDD which helps in applying intelligent methods such as data mining techniques to walk through the huge amounts data from different data sources from different organizations with the purpose to extract any hidden patterns. This useful information can then be used in business decision-making and problem-solving.

Data mining can bring significant benefits to an organization. As such, it has attracted a great deal of attention in recent years by many sectors of interest such as engineering, medicine, education, business and so forth. This has potentially led to the boom of many data mining tools, each offering a wide range of functionality to users. Some of the tools specialize mainly on the performance attributes, while others concern with other attributes such as user friendliness, and the number of data mining techniques available. However, the flexibility and extensibility of the tools to support any data sources and data mining techniques is also one of the integral parts that a data mining tool should have. According to Sanjiv, one of the challenges of a data mining system is the inability to provide platform and data sources independence [3]. This is due to the fact that different users may have different interests of knowledge. With the increase in the amount and speed of data interchanged nowadays, this has led to the increasing number of data sources [4]. In order to address to this issue, it is necessary for a data mining middleware to support a wide spectrum of data sources.

In addition, data mining techniques are created with the result of a long process of research and development [5]. It is unavoidable that new data mining techniques will be introduced to facilitate data mining processes due to the influence of dynamic environment and ever changing business requirements. For this reason, a data mining middleware should be extensible and flexible enough to allow any data mining techniques to be used in order to assist organizations to derive useful information through data mining with the purpose to gain a competitive edge in their businesses. This is made possible by applying a plugin-based architecture into the data mining middleware. Plugins are components that are optionally being added to the existing system to extend its functionality [7].

This research aims to propose an extensible architecture for the data mining middleware. The primary intention of this proposed architecture is to design the middleware so that it is able to support a variety of data mining functionalities. The architecture provides great flexibility for an unlimited type of data mining techniques from an unlimited type of data sources through the use of plugins. Users are allowed to plugin any new mining techniques into the system to mine the data. In addition, it provides better features as compared to existing data mining systems for it is designed to be a platform- and language-independent. Furthermore, the middleware also allows organizations to mine data through a wide range of data sources such as relational databases, object-oriented databases, text files, spreadsheets, XML files and others, to solve real-world business problems.
2.0 RELATED WORK

In this research, we have studied on a few data mining tools in the market namely IBM Intelligent Miner, SPSS Clementine, SAS Institute Enterprise Miner, Oracle Data Miner, and Microsoft Business Intelligence Development Studio. Our study mainly focuses on the flexibility of the tools based on whether the tools support plugins of new data sources and data mining techniques. In addition, studies are also conducted on the performance of the tools based on five attributes: memory shortages, excess paging with a disk bottleneck, paging file fragmentation, memory leaks, and cache manager efficiency.

At the time of our study, tools like Microsoft Business Intelligence Development Studio, Oracle Data Miner, and SAS Institute Enterprise Miner only support a predefined set of data sources. For instance, Microsoft Business Intelligence Development Studio only supports ODBC, OLEDB and other types of predefined data sources. Oracle Data Miner, on the other hand, only supports JDBC compliant driver such as OCI-based drivers. Implementing new data sources into such tools are difficult. For example, if users wish to mine new data sources such as Object-Oriented data sources and Multi-dimensional data sources, the implementation of those data sources are difficult and often involves a considerable amount of time to study.

Platform dependence is one of the key constraints of a data mining tool. In order to support a wide range of users, it is recommended that a data mining tool be platform independent. However, it is observed that most of the data mining tools are still platform dependent except for Oracle Data Miner. The Microsoft Business Intelligence Development Studio, for example, is platform dependent. The tool depends on .NET Framework which currently only supports the Windows platform. Even though it is possible for Microsoft Business Intelligence Development Studio to run on Unix and Linux by implementing Mono technology, unfortunately the necessary implementations are somehow complicated. As for SPSS Clementine, in order to support users of different operating systems, different binaries are released on different platforms. Lastly, Oracle Data Miner uses the same binaries on different platforms, and as such, is platform independent.

With the evolution of data mining, there are many new data mining techniques being developed and are supported in these data mining tools. All data mining tools studied offer a predefined set of data mining techniques with the exception of Microsoft Business Intelligence Development Studio that supports plugins of additional third party’s data mining techniques that comply with OLEDB data mining specification.

3.0 PROPOSED DATA MINING MIDDLEWARE

The proposed data mining middleware is referred to as Java-Based Data Mining Middleware (JDMM). This data mining middleware will be designed as a platform-, data source-, and data mining technique-independent middleware which is accessible from front-, back- and web-office environments. In short, users are able to use JDMM in any platform to mine any type of data sources at any place without geographical boundaries using a wide range of data mining techniques to help them in business decision-making. This is made possible through the use of plugins such as data mining technique related plugins. Any new data mining techniques can be plugged into JDMM through click-based type of configuration.

This middleware will sit between the knowledge seeker interface and the organization memory system’s repository. JDMM will provide data mining techniques to facilitate knowledge discovery to aid in organization-wide decision-making. The proposed high-level system architecture of JDMM is portrayed as in Fig. 1.

JDMM would be able to support an unlimited number of data sources and data mining techniques due to its ability to provide extensive pluggable adapter and flexible features. JDMM is proposed with three possible roles of users: JDMM Administrator, JDMM Implementor, and JDMM Business Analyst. JDMM Administrators will administer all JDMM instances and is responsible for the uptime of JDMM. JDMM Implementors, on the other hand, are technical users who are able to plug new adapters through the JDMM Web Configurator. Lastly, JDMM Business Analysts are non-technical users who are responsible on business decision-making through JDMM to solve real-world business problems such as planning a marketing campaign, forecasting product growth and other types of business decisions. JDMM Web Configurator is a web application that provides a user-friendly interface, allowing users to specify different data sources such as XML, relational database, object-oriented database and legacy system. After data retrieval, JDMM automatically organizes the data to create a data mining model using a specific data mining technique. Each technique is governed by adapters which are pluggable rule adapters. This means JDMM is able to use other open-source
components to apply a specific data mining technique. The result can be stored into a data mining repository or directly to a persistent data store such as a relational database. The final result is a XML file that will then be published and delivered to the users as either PDF file, XLS file or any proprietary formats that are incorporated into JDMM. The user interacts with JDMM through the Web JDMM which is deployed in the Tomcat Servlet Container.

4.0 JDMM ADAPTER FRAMEWORK PLUGIN ARCHITECTURE

In order for JDMM to support unlimited type or data sources, data mining techniques and reporting formats, the system architecture needs to be designed to be highly reusable, modularize and flexible. By limiting the number of data mining techniques in data mining tools, it will restrict organizations on their contributions toward new or customized data mining techniques. No matter how promising the functionalities offered by data mining tools, there are always special features that are needed by the organization. Hence, a data mining tool should provide the flexibility to allow users such as business analysts, who are usually the experts in a specific business domain, to customize, create and plugin their own specialized data mining techniques that best suit their environment. This is also applicable for the customization and extension of any new data sources and reporting formats.

The primary goal in promoting modularity and extensibility in JDMM is to build a plugin-based infrastructure library for both new and existing plugins (adapters) of JDMM. Extensibility is based on the concept of extension points (variation points) and extensions (variants). An extension point is an opening that may be added to by code later (superclass-abstract class or interface in Java). An extension is the code that adds onto an existing extension point (subclass-concrete class). This relationship is similar to that of traditional jigsaw puzzle [9]. Typically extension points are declared in a plug-in manifest (XML Metadata file).

In general, we believe that it doesn't matter where (in what plugin) the actual code and/or its resources are placed. It is much more important to define where an application can be extended, and then design and develop extension points to support this extensibility. JDMM Adapter Framework plugin architecture is based around the concept of plugins from Eclipse. Eclipse is the most prominent representative of plug-in platforms which introduce the idea of “Everything is a plug-in” [6].

This adapter framework allows JDMM implementors to configure different adapters (plugins) which need to be plugged into JDMM. The proposed JDMM Web Configurator allows JDMM implementors to configure different adapters which need to be plugged into JDMM. The proposed JDMM will accept a wide range of data sources such as relational databases, object-oriented databases, flat file and so forth. JDMM provides an extensible JDMM Adapter Framework that allows multiple different adapters (i.e., Data Source Adapter) connecting different data sources to be added. Any new data source available will only need to be plugged into JDMM Adapter through minimal configuration in an XML file. The JDMM Adapter Framework also allows different data mining techniques to be plugged and configured into JDMM similar to Data Source Adapter. We will call these adapters as Data Mining Technique Adapter. Examples of the proposed Data Mining Technique Adapters are adapters implementing different data mining algorithms such as hypothesis testing, time series, normal distribution, binomial distribution, poisson distribution and other types of algorithms. These adapters are also known as extension points. Lastly, the proposed JDMM also provides another type of adapter called Reporting Adapter through the JDMM Adapter Framework. The Reporting Adapter is used to describe different mined data into formats such as pdf file, Excel spreadsheet, csv file, text file, xml file, html/htm file and other types of format.

According to Ian et al, plugins are written in a standard object-oriented fashion and they utilize inheritance structure to promote reusability [8]. This concept will be adopted in the design of JDMM architecture where a Java API will be
developed and will expose certain interfaces which serve as the specification for the plugins. Anyone who intends to extend or customize any data sources, data mining techniques or reporting formats will need to implement the interfaces exposed by the API.

Fig. 2 depicts the plugin architecture of JDMM Adapter Framework. Through JDMM Web Configurator, administrators or implementers will inject adapters together with an XML metadata file into Plugin Repository which is the repository for all the adapters. During injection process, the location and validity of the adapters are validated through either a DTD or a XML schema. Successful injection of the adapters will then be registered into the JDMM Plugin Controller. The core function of Plugin Controller is to load and activate the registered adapters before they are used by the data mining engine. It is possible for the adapters to be developed using external libraries which will be incorporated into the Plugin Repository during injection process.

The Plugin Descriptor encapsulates a group of individual plugin factories such as Data Source Plugin Factory and Data Mining Technique Factory. The client, in this case, the JDMM Web Configurator is able to create concrete objects, which in this case, is a type of JDMM plugin through the Plugin Descriptor. The Plugin Descriptor is designed to separate the details of implementation of JDMM plugins. In short, the Plugin Descriptor serves as an interface to create families of related plugins through dynamic method invocation.

5.0 DETAILED ARCHITECTURE OF JDMM

The internal architecture of JDMM is divided into two threads namely Inbound Threads and Outbound Threads as shown in Fig. 3. There are two different priorities for Inbound Threads and Outbound Threads: Low and High. A high priority inbound thread or outbound thread ensures all operational activities such as data mining activities are finished before a shut down can be triggered within JDMM, and vice versa for a low priority inbound thread or outbound thread. The number of Inbound Threads and Outbound Threads, and the priorities are configurable through a standard Java properties file. An Inbound Thread manages all incoming uninterpreted operational data (also referred to as raw data), whereas an Outbound Thread manages all outgoing interpreted operational data (also referred to as mined data).

Within the Inbound Thread and Outbound Thread consist of Receive Adapter and Send Adapter which are part of a framework known as the Adapter Framework. Through the JDMM Web Configurator, implementors of JDMM are able to plug in different adapter components to connect to different data sources. Each adapter is configurable and each configurable parameter is stored in a typical Java properties file.

Java-Based Data Miner (JDM) is a pure Java API for developing data mining applications. The idea is to have a common API for data mining that can be used by clients without users being aware or affected by the actual vendor implementations for data mining. The JDM architecture consists of three logical components, the API, the Data Mining Engine (DME), and the metadata repository (MR). The API is the exposed programming interface that provides access to the services provided by the DME. The API shields the data mining user from the actual implementation in the DME and any associated sub-components used by the DME. The DME is the engine providing the services that can be used...
by JDMM users through the API defined above. The third component is the MR which is used to persist data mining objects. These persisted data mining objects are again used by the DME for data mining operations. The metadata repository can exist as a flat file system or a relational database. The three logical components are grouped into one physical system or they can exist independently as separate components.

JDM Extension is an extension to JDM that includes additional data mining models, data scoring and data transformations. JDM Extension is based on a highly-generalized, object-oriented, data mining conceptual model using Data Mining Group’s Predictive Model Markup Language (PMML) data mining standards. PMML is a XML markup language to describe statistical and data mining models. Finally, the interpreted results are sent to different data sources as output through send adapters.

JDMM will be designed as an information-intensive software, and as such, a proper caching is needed to allow maximum performance. Data Mining Repository (DMR) stores serialized objects captured from data sources at EJB Server. JDMM allows caching policies such as “no caching”, “least recently used”, “most recently used” and “caching duration” to be configured in an XML file. These policies are intended to tune the amount of cached data. The sole objective of the memory repository is to reduce any I/O during the process of mining the data sources. Majority of the data mining processes within JDMM are performed using the cached data from DMR. DMR is further divided into Caching Repository and Non-Caching Repository as shown in Fig. 3. The Caching Repository stores both interpreted and uninterpreted data into JDMM Directory Service. The JDMM Directory Service catalogs each transformation and descriptor that occurs during clusterings, associations and others into a volatile directory. Implemented as part of JDMM is a snap-in which allows JDMM implementors to configure the JDMM Directory Service. All configurations within JDMM Directory Service are stored in multiple configurable XML files. The Caching Repository is a copy of transaction data specifically structured within JDMM. The repository can be normalized or denormalized. The repository needs to be denormalized in the case if the objects residing in the repository is highly coupled. This is a typical workaround suggested to minimum the overhead of coupled objects. JDMM will support common relational databases, multidimensional databases, flat files, hierarchical databases, object-oriented databases and other types of data sources as the data stores in the repository. The Non-Caching repository stores each transformation and also descriptor that occurs during clusterings, associations and so forth into a database.

Fig. 3: JDMM Internal Architecture

6.0 PROPOSED LOGICAL COMPONENT OF MIDDLEWARE

The application systems such as JDMM Web Configurator and Web JDMM are built on top of the business specific component systems, Adapter Framework and JDM. Adapter Framework will be the component system to enable JDMM users to establish connections to a wide variety of data sources. JDM will be the engine to perform any data mining process. Each of these application systems may need to be executed on a different machine, thus leading to the needs to interoperate using middleware component systems such as J2EE, J2SE and Tomcat to host the application systems. JDMM is a platform independent system, hence it will be run on top of different operating systems such as Windows and Unix. The application systems in the Application layer are dependent on the component systems in the Business specific layer. Component systems residing in the business specific layer will not be reusing any component in the higher layer to avoid any static dependencies of a component system on more specific systems in the higher layer, which increases the volatility of the system.

As depicted in Fig. 4, the JDMM Web Configurator and the Web JDMM application systems communicate with each other during data mining process. For instance, during data sources configuration, the decision-maker actor may wish to mine a different data source, as a result, a different adapter corresponding to the data source will need to be configured and deployed into the system through Web JDMM. In the Web JDMM, the decision-maker actor will be able to perform data mining process from the adapter that is deployed.
7.0 CONCLUSION

The implementation of JDMM is still in-progress. We strongly believe that JDMM will be able to contribute to business areas such as human resources, business management/finance, IT/operations, sales/marketing and specialized/vertical (government). However, there is always a continuation for the evolution of every software development phase [1]. As for JDMM, although with the promising of extensibility and flexibility features, there are still rooms for improvement. According to Peyment et al, shutting down and restarting the system for upgrades will often incur unacceptable delay, increase cost and risk [10]. The current features provided by JDMM still requires one to restart the middleware if any new adapters are to be plugged in into the middleware. In the near future, our approach is to minimize the delay arise from a typical software restart. Another potential challenge that JDMM plugins might face is the upgrade of the Java Software Development Kit (SDK). In such a case, there might be possibilities that several components of the middleware need to be re-implemented due to deprecation of classes or methods at the API level.

REFERENCES


Proceeding of the 2nd International Conference on Informatics, 2007


BIOGRAPHY

Hen Lai Ee is a Software Engineer at Knowledge Networks Sdn. Bhd. She is currently following the Master of Software Engineering program at Faculty of Computer Science & Information Technology, Universiti Malaya. This paper is the part of the outcome of her research project in software engineering.

Sai Peck Lee is currently a professor at Faculty of Computer Science & Information Technology, University of Malaya. She obtained her Master of Computer Science from University of Malaya, her Diplôme d’Études Approfondies (D.E.A.) in Computer Science from University of Pierre et Marie Curie (Paris VI) and her Ph.D. degree in Computer Science from University of Panthéon-Sorbonne (Paris I). Her current research interests include Software Engineering, Object-Oriented (OO) Methodology, Software Reuse and Framework-based Development, Information Systems and Database Engineering, OO Analysis and Design for E-Commerce Applications and Auction Protocols. She has published an academic book and more than 80 papers in various local and international conferences and journals. She had also served as the executive editor of a journal for 2 years, and has been in the programme and reviewer committees of several local and international conferences.
TOWARD A METHODOLOGICAL FRAMEWORK TO SUPPORT PROFESSIONAL VIRTUAL COMMUNITIES

Fadila Kaouane¹, Imed Boughzala²

¹Department of Information system, National Institutes of Telecommunications. France. Email: fadila.kaouane@int-edu.eu
²Department of Information system, National Institutes of Telecommunications. France. Email: imed.boughzala@int-edu.eu

ABSTRACT

The fast progress of information and communication infrastructures, especially networking technologies over the past few years, has offered new possibilities for the knowledge creation and sharing within organizations through the emergence of collaborative networks such: virtual community, community of practice, community of interest.... The CSCW tools or groupware are the famous tools to support these forms, in particular, can be deployed to help organizations manage their empiric knowledge more efficaciously. The complexity of CSCW tools is growing from classical workstation to capillary CSCW; however, current solutions usually are not very customisable and interoperable and are not utilizing the full potential of the community support idea. In our research; we have focused ourselves on virtual professional communities, and we are interested on the specification of a collaborative framework for supporting and sustaining the development of such communities. In this paper we introduced our architecture model for a generic community support platforms designed to fulfil all theirs requirements.

Keywords: Community of practice, Professional virtual community, Collaborative tools, CSCW, Groupware, Knowledge management.

1.0 INTRODUCTION

Computer supported cooperative work (CSCW) is a large field of investigation, is focused on infrastructure enable persons or groups of persons to communicate, cooperate, perform a common task, solve a common task on a set of joint artefacts, shared their knowledge, etc. The development of these technologies of information and communication has promoted the emergence and the apparition of new forms of collaborative networked organizations: virtual enterprises, virtual teams, virtual communities, etc. In fact, these communities: of practice [26], of learning [9], of interest [15], of knowledge [3] [14], of project or professional virtual seem as a large body of empiric knowledge; they allow creation, sharing and capitalization of knowledge within the organizations. The professional virtual communities present a kind of aggregation of collective expertise on subjects or particular fields, this organizational form has gave place to many research works emanating from various fields: computer science, management, social and human science, etc. The complexities found in this research area are namely in the concepts, terminologies and associated meanings (for a broad overview of collaborative networks, we refer to [5]). These studies started by being interested to the comprehension and understanding of "why" appear these new organizational forms, and "in what" arises and resumes the objectives of their establishment. Recently, some work dealt with this subject and they focused rather on "how" manage and sustain these communities [23]; [27]; [10]; [16]. However, rarely are those devoted on the «how» identify, build or successfully support such community [22]; [20];[10]; the various proposal introduced on the few recent years, are limited to methods and guidelines which are more or less specific and meet to fulfill a specific need. In our work, we has been undertaken in how to build integrated professional virtual collaborative environment (PVCE) with the aim of supporting and successfully sustaining large set of collaborative networked organizations, especially professional virtual community across organizational boundaries and over their whole lifecycle.

Many models, tools, groupware and interaction patterns have been developed for supporting collaborative work. Each one focused on specific or narrow aspects of collaborative work. Groupware products include, amongst others, structured communication systems, shared whiteboards, group scheduling systems, videoconferencing systems, co-authoring systems and workflow management systems (for a broad overview of groupware products, we refer to [21]). As the number of CSCW is increasing, new collaborative networked forms are emerged, and so; new sociological consequences of this new way of working are identified. One of the most important results about these collaborative systems is that, most of the time, they offer a little interoperability and they are designed to support one specific kind of community, or well adapted for one narrow aspects of collaborative work, but can not be applied in large and dynamic work situations as envisaged by collaborative networked organizations. An overview of the literature [5] found that the behaviour of virtual community specially professional virtual community is not understood well enough and a solid
theoretical basis to support developments in these areas is missing, when the basic concepts are not understood, it is difficult to develop support services for set up, operation and disclosure of PVC.

For all these reasons, and for the lack and the absence of reference models for collaborative networked forms (Fig. 1), we consider that the actually challenge is to establish a groupware which support large kinds of collaborative organizations such virtual groups of project, virtual communities, virtual enterprises, etc., and provide a very wide range of collaborative tools adapted to their multiplicity and complexity collaborative work’s requirement, en consequence; they must be generic, very flexible; and more importantly, they should provide services which allow the users to personalize their environment, to switch at run-time from one tool to another, to develop their own collaborative processes, to specify their suitable workflow, and more then functionalities.

Considering these different issues, our aim is not to specify new patterns, collaborative tools or so, although, we based ourselves on the earlier work based understanding and analyzing of the nature, the characteristics, the requirements and the structure of professional virtual community, in order to propose a methodological framework and an overall architecture providing a generic, flexible, configurable and malleable collaborative/cooperative environments.

This paper presents our research in progress which falls under the collaboration engineering [25] and which is located at the crossroads of management, of the sociology of the organizations and of information system (CSCW, CIS). This paper introduce our proposal framework, a collaborative environment to support the increased mobility of workers and the flexibility of work and interaction, it leads the challenge of the definition of reference models for supporting and sustaining users in different organizations, with different roles within the collaborative process, using a wide range of different tools with different functionalities, thus this environment, need to be extensible and adaptable to varying work situations and collaboration forms, support reuse and combination of tools for different team settings, the users need to be able to extend, tailor, and personalize their work environment in order to adapt their changing needs. This paper is organized on two parts, in the first one; we introduce an overview of the literature on the related works especially the existing models and platforms introduced for supporting and sustaining collaborative works. In order to better determine our positioning and contribution, we present the limits of these models and the actual tendencies of the collaborative works.

The last part which is the focal part of this paper, it presents our different contributions in progress focalized on the specification of a flexible collaborative platform support for professional virtual community (and more collaborative forms). Finally we conclude this paper by drawing our future work and various prospects.

2.0 APPROACHES TO SUPPORT COLLABORATIVE WORK

Computer Supported Cooperative Work (CSCW) emerged as a research area in the mid 80’s [2]. Several types of collaborative applications recently advertise their contribution to “community support” and labelled this area of research as Collaborative work environments (CWEs) which aimed to provide “the ability to collaborate over time and space, within and between organizations or communities […] to achieve flexibility by making best use of the knowledge and competences available”[1]. However, none of the proposal models and platforms until present seems to be general and complete enough, more of them are concentrated at the specification of collaborative tools (co-authoring, synchronous communication, workflow, etc) rather then collaborative environments (we refer to the typology of collaborative tools given in [5]).

In general, community support is seen as a kind of cooperative information system, it includes models and methods for supporting communication, coordination and collaboration (and more cooperation) in a group of people. It includes support for synchronous and/or asynchronous communication, support for exchange of information and knowledge, support for coordination of tasks, support for collective resolution of problems, etc. In the literature, we can distinguish three principal approaches adapted in the development of the cooperative systems (CSCW): ad-hoc development, use of the patterns, and platforms for CSCW.
a) In the ad-hoc development, the application is usually established from nothing, making little or not call to the use of the components or models refined. It is the result of an unstructured or semi-structured process adapted and defined by the owners of the application, the famous examples are the open sources appeared at the few recent years (egroupware, phpcollab, etc.).

b) The patterns of CSCW, make it possible to establish an application (a tool) based on a whole of preset modules which can be re-used and combined in various ways, such as DistEdit (employed to adapt the individual editors to the shared edition [13]), and Thinklets (employed to develop repeatable collaborative processes [24]).

c) Platforms or methodologies of CSCW (models), make it possible to design a collaborative framework or cooperative information systems based on models defined, it is enough to instantiate and adapt it to there applicability, such as MeDICIS for the design of the framework of co-operation between firms [4].

The several proposal methods and frameworks do not take into consideration the adequacy of tools and functionalities integrated with the needs and the requirements of the large wide of diversity users. Therefore, this new alteration led to find more flexible and efficient solutions to implement environments adapted to all collaborative situations. The establishment of such solution reposes mainly on the specification of a cooperative information system (CIS) in which, users have the possibility to configure, personalize, specify there ones collaborative environments. In our research, we proposed a specification of new collaborative framework that exploits the existent theories, methods and models (MeDICIS, Thinklets, Ontology…), and profits from strengths of each particular model and encloses the three different approaches mentioned above (ad-hoc, pattern, and methodology).

3.0 TOWARD A METHODOLOGICAL FRAMEWORK TO SUPPORT PVC (MiCS)

Recently, certain survey and research projects are interested to the problematic of supporting communities, and aimed to introduce platforms, frameworks [11] [18] [17], the absence of a reference models and structured methodology requires a preliminary engineering and research task to identify and model the main elements of this collaboration, and which must answer certain questions, for example: what are the characteristics of PVC? Why building such communities? Which are theirs requirements and needs? How ensure their best evolution and development? Which are the adequate technologies to support them? Which are the resources necessary to mobilize?... To answer these questions and to define our methodology named MiCS (Fig. 2), we are based on MeDICIS methodology and we are extended them to be more dynamic and not limited to one domain and support a large set of collaborative networked organizations. In our approach, we are categorizing the problematic of supporting collaborative networked forms (especially PVC) into three dimensions: Design, Identification and Development dimension (Fig. 3). We focus ourselves currently in this present paper to the first dimension and we extension thereafter our research on the two last dimensions by the definition of a tool for identify PVC through ontology networked analysis based on PVC features and the development of a structural model for development of PVC which assure the good management, development and also to support the transfer, the sharing, and the creation of knowledge inside the PVC.

The first dimension: Design dimension refers to the cooperative information system and the associated technologies in particular the collaborative tools support to the PVC. The aim is being to put collaborative tools at the service of the communities in order to support theirs evolution and to strengthen their cooperation. At this stage, it is advisable to study the adequacy of the collaborative tools and the intensity of cooperation developed within the professional virtual community, like their contributions in the process of creation of knowledge.
The **Identification** dimension refers to the definition or the surround of the potential resources that form a community based on a list of a crucial structural PVC characteristics. And the last dimension: **Development** refers to the definition of a structural model of management which guide and assure a successfully evolution and expansion of PVC during their lifecycle.

![Fig. 3: A map modelling of our methodological framework](image)

We are investigating an ad-hoc process for building and sustaining PVC (Fig. 4), our ad-hoc process gathers the three great dimensions mentioned earlier (Fig. 3). This process has the particularity to support such sponsored and spontaneous PVC, it has for objective to support exchange, to capitalize the new knowledge created and emerged from the interaction inter-members and to develop and disseminate best practices.

![Fig. 4: Ad-hoc Process for building and sustaining PVC.](image)

To achieve our intent and for illustrate our proposal ad-hoc process, we had to specify models and tools which incorporate the three phases mentioned above (design, identification, and development), for that, we had analysed and studied a set of collaborative models and theories with emphasis on coordination and organisation aspects of work practices to obtain enough information for developing own framework; we based ourselves on MeDICIS methodology for designing inter-enterprises cooperative information system [4], the Thinklets (design patterns for collaboration engineering), and we adopted an ontological approach for the implementation of MeDICIS’s models. This coupling between MeDICIS, Thinklets, ontology, and inference engines (rules, reasoning, deductions...) makes it possible first, to design the community and the collaborative processes in which they operate, interact and base. And second, to deduce new collective knowledge starting from the initial knowledge shared and capitalized in the bases of MeDICIS.

Fig. 5 shows our overall architecture model of the virtual space shared by professional virtual communities specifically and virtual networked organizations generally (administrations, public and private organizations, professional, etc.). Our specification of the framework provides an adaptive, flexible and evolutionary support for collaborative work (virtual project teams, virtual communities, virtual enterprise…) and especially for successfully sustains PVC through their lifecycle. It is based on MeDICIS methodology and RDF ontology as technologies of conception and implementation. This platform provides a static part with basic modules sharable even by the whole users of the platform and gives access to the common information and knowledge, and a dynamic part with specific modules accessible through web services interface, and specified according to user’s profile (member of virtual project team, member of VPC, etc.) and gives access to specific collaborative environments with configurable tools, functionalities, etc.

The dynamic part consist mainly on two sort of web services: generic web services, on which over collaborative tools are integrated and specified according to the requirements of the collective (project team, PVC, etc.) the intensity of the collaboration, and work mode (cohesion, collision, …); and made-to-measure web services, on which they are developed and specified according to the organization business needs. In this category and for assuring a successfully sustain of PCV, we were specified four web services, which we qualify as necessary but not sufficient to support PCV: Intelligent search engine, Workflow Editor, PVC Identification tool, PVC proactive management tool.
A dynamic library of collaborative tools is available through web services (whiteboards, chats, wiki, blog, shared calendar, etc.) and which are specified in function of the collaborative work situation designed by the workflow processes based on coupling the coordination model of MeDICIS and the Thinklets. These collaborative tools are classified through their cooperative intensities and they could be dynamically involving with others tools due to the use of web services as technology of integration. In our approach of supporting community work by integrating collaborative tools and workflow editor, we aimed to facilitate ad hoc collaboration and dynamic work contexts, and so the emergence of collaborative networks (PVC).

Even when community members collaborate and share their requests through this virtual space, for example: help or answer in the threaded discussion, discuss a particular problem or potential solution, etc., new knowledge has been explicited, reviewed, modified and generalized into guidelines, reports, directives, or practices through the use of the ontology and the inference engines included to the platform, members can access to the knowledge by interrogate the research web service through a query interface passing request to an intelligent search engine. This search engine through the use of the ontology can retrieves new and relevant knowledge not only related directly to the given request but explored more advanced knowledge.

We can resume our purposes of the definition of this framework outlined above into:

- Supporting and sustaining a large set of collaborative networked organizations especially PVC;
- Personalising and configuring the workspace in function of the different needs and constraints of the collaborative work;
- Providing a workflow editor and features tools for designing collaborative processes;
- Providing a PVC identification tool to identify collaboration opportunities;
- Providing a PVC proactive management tool to successfully manage and assure the best evolution of the collective through their lifecycle;
- Assuring the traceability and the capitalisation of the flow of the shared information and knowledge;
- Managing the collective knowledge sharing through the collaborative environments;
- Improving flexibility, adaptability and co-evolution of the environment by the use of the web services as technology of integration;
- Providing a common base effective of sharable information/knowledge through software tools;
- Increasing the chances of PVC involvement even from distributed and geographic dispersed organizations;
Our proposal collaborative framework and associated design patterns, collaborative tools has for objective to support and sustain different collaborative networked organizations such as project teams, communities of practice, and more especially professional virtual communities -our purpose of research-. In earlier research project ECOPICS² [6], a virtual environment based MeDICIS methodology has been developed and experimented to support virtual inter-enterprises project teams. In our work we have inspired from to specify our collaborative environments which coupled the access and work in different modes: project, community, etc. This collaborative environment will enable all knowledge workers across organizations to network and form groups and professional virtual communities, sharing and creating knowledge, and developing best practices.

We have adopted .NET, Web services and ontology technologies to specify and develop our collaborative platform. By the way, users have the possibility to create, specify and personalize their collaborative processes, their collaborative spaces, their collaborative tools and so on, professional virtual community can be emerged or intentioned by companies and collaborative tools can be specified and integrated to the community shared space in function of the level of the cooperation and the state of there development. This is the main subject of our ongoing work into the two others dimensions of our methodology approach: identification and development, and which will be addressed in future papers.

4.0 RELATED WORKS

In the few recent years, a large number (over 40) of CWE research works based projects can be found in the literature such us Legal-IST project³, Ecospace project [18], Laboranova project⁴, Ecolead project [17], etc. These projects put a strong emphasis on the CSCW tools and infrastructures to support collaboration within organizations due to the lack of consolidated and holistic approaches. The Ecolead model and Ecospace project are the mainly contributions examples of platforms support for virtual organizations. Ecospace project’s aim is to create and deploy an eProfessionals collaborative working environments, and Ecolead project’s mission is focused on the specification of framework for Virtual Organizations. These research works are currently in progress, and at the difference of there, the key initiatives and benefits of our proposal model are that is more generic and support both spontaneous and sponsored PVC, virtual project teams, and all collaborative networked organizations. It leads to offer the most flexibility and liberty to member to specify and personalize their own collaborative environments. The use of Web service and .NET as technologies of integration permits plus the flexibility of the environments and more nomadism of users.

5.0 CONCLUSION AND PERSPECTIVES

Professional virtual community is a recent field of investigation; several works have been developed on this topic treating various points of view: sociology, management, information system, etc. In this paper, we have introduced an overview and an analysis of works related to this field in order to outline better our problematic and our research work focused on the definition of a reference methodology to support VPC through three dimensions design, identification and development. We have introduced and presented our framework designed to sustain and support professional virtual communities over their lifecycle. Our proposal solution is based on the coupling of MeDICIS methodology, Thinklets patterns, ontology and web services. It is expected to allow a high flexibility for communities by supporting their collaborative processes, and their development processes throughout this overall virtual space architecture, also our platform allows the members of a community to create and design their own collaborative process through the workflow editor integrated. In our future steps of research, we work into the definition of a tool for the identification of PVC through ontology cooperation network analysis and the definition of a development model assuring the evolution and the growth of VPC.

REFERENCES


² ECOPICS
³ LeGAL-IST
⁴ Laboranova

©FCSIT, UM 2007


Licklider J. C. R, Taylor R. W. (1968), the computer as a communication device.

Nickols, F. (2000). Communities of practice: Overview; Start up kit; Roles and responsibilities; http://home.att.net/~discon/KM/CoPs.htm


**BIOGRAPHY**

Fadila KAOUANE is a PhD student at Department of Information system, National Institutes of Telecommunications, France. His research areas include Cooperative Information System (CIS), Collaborative technologies and Professional Virtual Communities (PVC).

Imed BOUGHZALA is an associate professor at Department of Information system, National Institutes of Telecommunications, France. His research’s interests include topics related to Cooperative Information System (CIS), and Knowledge Management (KM).
WSEP: AN E-PROCUREMENT SYSTEM OF SUPPLY CHAIN MANAGEMENT USING WEB SERVICES TECHNOLOGY

Yap Khay Seng, Siti Hafizah Ab Hamid, Mohd Hairul Nizam Md Nasir
Faculty of Computer Science and Information Technology,
University of Malaya, 50603 Kuala Lumpur, Malaysia.
Email: ksyap@hotmail.com, sitihafizah@um.edu.my, hairulnizam@um.edu.my

ABSTRACT

This paper presents a system called E-Procurement system in the context of supply chain management. The current Information Technologies (IT) such as Electronic Data Interchange (EDI) and Enterprise Application Integration (EAI) has been used by major organizations to integrate the business processes in the supply chain. However, these technologies are expensive, inflexible and not dynamic. Web Services seem to be a way to solve these problems. By using the standard internet protocol, Web services can easily provide interoperable software service functions that can be accessible by various hardware and platform, even passing through the firewalls. The purpose of this project is to develop a system using Web Services to implement real time information sharing and procurement operations in the context of Supply Chain Management. Web Services for E-Procurement (WSEP) System is a web-based procurement system that provide real time knowledge sharing and procurement operations. The system is deployed using Web Services which is not only easier to implement but also integrated well with legacy system.

Keywords: Supply Chain Management, Web Services

1.0 INTRODUCTION

“A supply chain is a network of facilities that procure raw materials, transform them into intermediate goods and then final products, and deliver the products to customers through a distribution system” [5]. By uniting the disparate business processes, enterprises in the supply chains can effectively improve transaction performance and reduce costs. In the context of procurement, extending the business processes across the enterprise involves collaboration of suppliers, which includes the activities of involvement in production design, daily coordination of purchased material flows and etc. These activities require system integration as well as sharing information in a real time manner. However, current supply chain implementations in Electronic Data Interchange (EDI) systems do not really meet these requirements.

Web Services are the consequence of the evolution of the World Wide Web (WWW) that provides a standard protocols and data formats for implementing distributed components. By using ubiquitous and standard internet protocol, Web services can easily provide interoperable software service functions that can be accessible by various hardware and platform, even passing through the firewalls. Therefore, enterprises can use Web Services to expose their business processes to the external trading partners. The advent of Web Services has simplified Business-To-Business (B2B) integration.

With the implementation of Supply Chain Management (SCM), the organizations can streamline the business processes, improve response time to customer requests and reduce costs. SCM coordinates the processes in production management, inventory management, order fulfillment, purchasing and distribution. A Supply Chain typically extends the business process across and beyond organization. Information is passed through different systems of the companies across the supply chains. Electronic Data Interchange (EDI) and Enterprise Application Integration (EAI) have been used by major organizations to integrate the business processes in the Supply Chain. However, these technologies are expensive, inflexible and not dynamic. If some of the participating systems change, the integration must be redone which is a time consuming and expensive process [8]. These problems hinder the applications of dynamic supply chain management.

Web Services seem to be a way to solve these problems. This project uses Web Services technologies to overcome such problems. Web Services for E-Procurement System is a web-based E-procurement system that demonstrates the using of Web Services to provide dynamic procurement operations and real time information sharing over the internet.

The purpose of this research project is to develop a system using Web Services that can be used to implement a real time information sharing and procurement operations specifically for small businesses that have non-complex process
business flow. The architecture and design of the system presented in this research project are based on an n-tier enterprise design pattern.

2.0 Issues and Problems of IT in the context of Supply Chain Management E-Procurement System

A supply chain typically involves multiple parties, including manufacturers, transportations, warehouses, retailers as well as the customers. Information Technology (IT) has been playing a key role in improving the efficiency of the supply chain by integrating information system across multiple organizations.

For more than two decades, enterprises have been using Electronic Data Interchange (EDI) systems for basic B2B document exchange, i.e. orders, invoices, shipment notices and so on. EDI enables a company’s trading partners to transmit requests or documents directly into their business systems. For instance, an electronic version of a purchase order can be transmitted from the buyer and delivered directly into then order processing system of the seller. EDI translation software converts the incoming electronic purchase order from a commonly understood format into the proprietary input representation expected by a backend business application.

The EDI standard is monolithic, complicated, and coordinated through a heavy committee process. As a result, high costs are introduced in the way of expensive translation software, consultant, VAN services and dedicated staff required for implementation and maintenance. Even with these disadvantages, EDI is still currently used in SCM for sharing information and communication.

The XML acronym stands for eXtensible Markup Language. XML can represent almost any type of information. It is relatively easy to learn, and many inexpensive tools are available for working with XML-formatted data. In response to these benefits, XML-based technologies are infusing major emerging e-business initiatives and standards, ranging from B2B documents, application interfaces, B2B communication protocol and complete framework.

As we know, the transactions in the supply chain need to have real-time response. However, both EDI and XML documents are often process in batch. This causes the information of each entity in the supply chain is not refreshed timely and up-to-date. Thus, supply chain business process by using EDI and XML do not really support real time response.

Web Service is a new way for companies to integrate the business software applications within the organization and with trading partners. Web Services use standard protocols and data formats such as HTTP, SOAP, and XML to connect to other software applications. This enables the information in the supply chain can be exchanged easily and in real-time manner.

3.0 Scopes

The scope of the project covers the provision and consuming of Web Services over Internet. This project will cover the development of Web Services for E-Procurement (WSEP) System that includes the following modules:

Authentication Module: This module is to authenticate and authorize the user. This involves accepting credentials from the users that include username and password, and validating them against database. Only purchase officers with access permission are allowed access to the system.

Maintenance Module: This module is to manage and maintain the system entities such as products, suppliers and suppliers’ Web Services Uniform Resource Identifier (URI).

E-Procurement Transaction Web Services Module: This module is the main component that handles the interactions between the E-Procurement system and the suppliers’ Web Services. It queries the database in order to get the qualified suppliers information as well as suppliers’ Web Services URI. Based on the information, it will create a proxy to dynamically bind the suppliers’ Web Services for procurement operations.

Suppliers’ Web Services Module: The Suppliers’ Web Services are necessary as part of the WSEP System. All the suppliers who want to participate in the E-Procurement System have to implement Web Services that conform to the Web Services Description Language (WSDL) file. These Web Services expose their internal inventory system and allow the suppliers to submit the quotation in real time based on the customer requests. The suppliers are not restricted to any Web Services product for implementing this module since Web Services can easily communicate with each other no matter what language is used.
4.0 System Architecture

4.1 How different the system architecture compare to other existing architecture?

In order to ensure the system architectural design is standard, there are three existing architectural designs [1][7][8] from existing projects that have been reviewed. All of these designs used registry or UDDI to register Web Services. UDDI design center is a high level business or service registry. It is focused on the discovery aspect of businesses, web services, and the technical interfaces that they make available. The support for and use of UDDI is currently very limited [2]. If the needs are more than simple publish and discovery of business or service metadata, UDDI should not be used.

From the perspective of E-Procurement, the public UDDI is not suitable for the proposed WSEP System since it does not provide sufficient information, such as product data. Therefore a private database system has been used, as compared to UDDI, to store product and supplier information as well as supplier Web Services URI. The database, in this case, is easier to maintain and is more efficient because there are less security issues.

4.2 Architectural Design

Based on Figure 1, Authentication module is to validate the identity of a user to allow or deny the access to the system. ASP.NET provides built-in support for user authentication through several authentication providers [4]: Forms-based authentication, Microsoft Passport authentication and Windows authentication. This project has used the Form-based authentication. User has to provide the username and password in order for the system to validate against the database.

Maintenance module is for the Purchase Officer to maintain the entities of the system, which include suppliers, products and suppliers’ Web Services. This module simply performs CRUD (Create, Read, Update and Delete) operations against the entities in the database.

E-Procurement Transaction Web Services module is a module implemented using .NET XML Web Services and Reflection. The main function of this module is to bind the supplier Web Services and handle the real-time transactions. It acts as the middle-tier component and provides the necessary business logic rules for procurement processes.

Basically, this module will receive user events from user input. Based on the user selection, this module will query the database to retrieve the suppliers’ Web Services URI addresses. It will then dynamically create the proxy to consume the supplier Web Services for information retrieval or transaction.

4.3 Design of Dynamic Aspect of WSEP System

UML Sequence Diagram is used to model the flow of logic within the WSEP system. It helps in identifying the behavior within the system and validating the business logic and interfaces by describing the sequence of actions that need to be performed to complete a task or scenario. Figure 2 visualizes the request for quotation process.

The process starts with the user initiates a request for a product quotation. The user enters the Product Code in the Quotation Request form and submits to the E-ProcurementTransactionWS module. The E-ProcurementTransactionWS, which is implemented as Web Services, will retrieve the Supplier WSDL from the Database based on the Product Code. For each WSDL retrieved, the module will send a Quotation Request in XML format to the respective Supplier Web Services. The Supplier Web Services will retrieve the product price along with the quantity on hand from their legacy system and return the information to the E-ProcurementTransactionWS. The returned results are displayed to the user in grid format.

From the results returned, user can select multiple suppliers to place the purchase order with different quantity. Again, the E-ProcurementTransactionWS will retrieve the Supplier Web Services information from Database in order to send the Purchase Order to the selected Suppliers. A confirmation of success or failure of the transaction will be sent back to the user. This ends the e-Procurement process.
5.0 Database Design

The information of the WSEP system is stored in a relational database. The selected DBMS is Microsoft SQL 2000 Server, which can be managed from within Microsoft Visual Studio.NET, or from the SQL Server Enterprise Manager. The database design of the system is presented in Figure 3 below. These tables are in Third Normal Form. Normalization is a formal technique for analyzing relations based on their primary key (or candidate keys) and functional dependencies. This is done to avoid data redundancy and update anomalies. The Third Normal Form (3NF) is a relation that is in first and second normal form, and in which no non-primary key attribute is transitively dependent on the primary key (Thomas and Carolyn, 2002).

The AuthUser table is used to store username and password for authentication purpose. The Product table and Vendor table are the entity types that keep product information and vendor information respectively. The VendorProduct table is the entity that provides associative relationship between Vendor and Product. It tells what products are provided by the vendors. The VendorWS table is used to keep information for the Vendor Web Services. The WSDL for accessing the Vendor Web Services is stored here. For all the purchase orders made, records are kept in the PurchaseOrder table.

6.0 UI Design

User interface design must take into account the needs, experience and capabilities of the system users. Table 1 lists the design principles (Ian, 2001) that are incorporated into the user interface design of the WSEP system. The user interface is built on a consistent pattern across all pages that all share the same navigation menus, basic data grids, graphic themes and layouts. The goal is to be consistent and predictable so that the users would feel comfortable and confident while using the system. The menu consists of three parts namely Maintenance, Procurement and Common which
As a system is used, users will have a basic concept of the system works. Users will become frustrated and irritated if a system behaves in an unexpected way. The user interface design of the WSEP system is striving towards the principle of "least surprise". If one attribute of an object is changed in a particular way, then it is to be expected that other attributes of the object would also be changed in a similar fashion. Users may make mistakes when using the system. The WSEP system provides some resilience to user errors and allows the user to recover from errors. This includes undo facility and confirmation of destructive actions as shown in Figure 4.

Table 1: User interface design principles

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User familiarity</td>
<td>The interface should use terms and concepts which are drawn from the experience of the anticipated class of user.</td>
</tr>
<tr>
<td>Consistency</td>
<td>The interface should be consistent in that comparable operations should be activated in the same way.</td>
</tr>
<tr>
<td>Least Surprise</td>
<td>Users should never be surprised by the behavior of the system.</td>
</tr>
<tr>
<td>Recoverability</td>
<td>The interface should include mechanisms to allow users to recover from their errors.</td>
</tr>
</tbody>
</table>

Fig. 4: Screen Shot
7.0 Testing

1. Security Testing

The system is protected from unauthorised access. This is done by providing user authentication and using secure channel like SSL to encrypt the data transfers. The system should provide a means to enter user names and passwords. All the web pages are protected. If an unauthorized user tries to access any pages, the system will redirect the user to the login page to enter username and password.

2. Dynamic Requirement Testing

The system should provide dynamic and real time operations to the user. The Supplier Web Services should be invoked at run-time instead of design time. In order to check if the assemblies are created during run-time, a break point is set in the DynamicWebServicesProxyLib class where the assembly is compiled and created. By tracing into the code, the temporary path of dynamic assembly is identified, as shown in Figure 5. There are a few assemblies created in the folder during the execution of the codes.

![Fig. 5: Assemblies that are Created Dynamically](image)

In order to examine the assembly created, Visual Studio .NET Object Browser is used. The assembly is referenced and imported. As shown in Figure 6, the assembly has the namespace WSEP.Tools.WebServices.DynamicProxy and has a class PriceQuote with the function GetInventoryInfo(). This assembly acts as the dynamic proxy that will bind to Supplier Web Services.

![Fig. 6: Visual Studio .NET Object Browser Is Used to Examine the Dynamic Assembly](image)

3. Performance Testing

In software engineering, performance testing is the testing that is performed to determine how fast some aspect of a system performs under a particular workload. Performance testing is to demonstrate that the system meets performance criteria. It involves obtaining data concerning how well the system executes the functions they were designed for.

During the testing process, system attributes such as CPU usage, memory usage, SQL Server response time are captured to gauge the performance of the system. In order to do this, a Windows Form test program written in .NET is built. The performance test screen is shown in Figure 7. All the tests are conducted using the same algorithm as in Table 2. The test involves looping of M records of products multiply with X records of suppliers for each product. The result can be seen in Table 3.
Table 2: Algorithm for Performance Testing of Request For Quotation Transaction

| Get N number of quotations from N suppliers for M products |
| Loop through M products |
| For Each Product |
| Get X suppliers and suppliers WSDL |
| End Loop Products |
| Display Products |

Table 3: Test Result of Performance

<table>
<thead>
<tr>
<th>Number of Products</th>
<th>Total Records Returned</th>
<th>CPU Usage (%)</th>
<th>Memory Usage (%)</th>
<th>SQL Duration (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2.0</td>
<td>3.0</td>
<td>60</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>4.0</td>
<td>4.0</td>
<td>60</td>
</tr>
<tr>
<td>20</td>
<td>200</td>
<td>5.0</td>
<td>7.0</td>
<td>60</td>
</tr>
<tr>
<td>50</td>
<td>500</td>
<td>8.0</td>
<td>11.0</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>19.0</td>
<td>19.0</td>
<td>240</td>
</tr>
<tr>
<td>200</td>
<td>2000</td>
<td>27.0</td>
<td>29.0</td>
<td>303</td>
</tr>
<tr>
<td>500</td>
<td>5000</td>
<td>47.0</td>
<td>49.0</td>
<td>430</td>
</tr>
<tr>
<td>400</td>
<td>4000</td>
<td>43.0</td>
<td>44.0</td>
<td>370</td>
</tr>
<tr>
<td>500</td>
<td>5000</td>
<td>47.0</td>
<td>49.0</td>
<td>430</td>
</tr>
</tbody>
</table>

8.0 Conclusion

The current Information Technologies (IT) such as Electronic Data Interchange (EDI) and Enterprise Application Integration (EAI) has been used by major organizations to integrate the business processes in the supply chain. However, these technologies are expensive, inflexible and not dynamic. Web Services seem to be a way to solve these problems. By using the standard internet protocol, Web services can easily provide interoperable software service functions that can be accessible by various hardware and platform, even passing through the firewalls. The purpose of this project is to develop a system using Web Services to implement real time information sharing and procurement operations in the context of Supply Chain Management. Web Services for E-Procurement (WSEP) System is a web-based procurement system that provide real time knowledge sharing and procurement operations. The system is deployed using Web Services which is not only easier to implement but also integrated well with legacy system. The system is running under the Microsoft .NET 1.1 Framework and Microsoft SQL Server 2000 as the backend database.

REFERENCES


Proceeding of the 2nd International Conference on Informatics, 2007


BIOGRAPHY

Yap Khay Seng obtained his Master of Software Engineering from Universiti Malaya in 2007. Currently, he is working as a Team Leader in a small UK software company. He has been working with Visual Basic .NET, C# and ASP.NET for many years. He specializes in software architecture and development using .NET framework.

Siti Hafizah Ab Hamid is a lecturer at Department of Software Engineering, Universiti Malaya. Her research areas include Electronic Learning, Mobile Learning, Applied Object-Oriented Methodology, Applied Electronic and Mobile Technologies, Learning Objects, Software Metrics, Applied Game Techniques.

Mohd Hairul Nizam Md. Nasir received his Bachelor of Computer Science (Hons.) specializing in Software Engineering and Masters of Computer Science (by full research) from University of Malaya in 2004 and 2005 respectively. He is currently serving as a lecturer in the Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya. His area of specialization includes web-based application development, software quality, and software process improvement as well as Web Services.