ISSN: 1945-3116 Vol. 3, No. 5, May 2010

Scientific Research

# Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz


ISSN: 1945-3116

www.scirp.org/journal/jsea

# Journal Editorial Board

http://www.scirp.org/journal/jsea

*J. Software Engineering & Applications*, **2010, 3, 419-516**
Published Online May 2010 in SciRes (www.SciRP.org/journal/jsea/).

# TABLE OF CONTENTS

**Volume 3 Number 5** **May 2010**



# Journal of Software Engineering and Applications (JSEA)

# Journal Information

## SUBSCRIPTIONS

The *Journal of Software Engineering And Applications* (Online at Scientific Research Publishing, www.SciRP.org) is published monthly by Scientific Research Publishing, Inc., USA.

### Subscription rates:

Print: $50 per issue.
To subscribe, please contact Journals Subscriptions Department, E-mail: sub@scirp.org

## SERVICES

**Advertisements**
Advertisement Sales Department, E-mail: service@scirp.org

**Reprints (minimum quantity 100 copies)**
Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.
E-mail: sub@scirp.org

## COPYRIGHT



## PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: jsea@scirp.org

*J. Software Engineering & Applications*, 2010, 3: 419-425
doi:10.4236/jsea.2010.35047 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

# A Reference Model for the Analysis and Comparison of MDE Approaches for Web-Application Development

**João de Sousa Saraiva, Alberto Rodrigues da Silva**

INESC-ID/Instituto Superior Técnico, Lisboa, Portugal.
Email: joao.saraiva@inesc-id.pt, alberto.silva@acm.org

Received January 26th, 2010; revised March 18th, 2010; accepted March 20th, 2010.

## ABSTRACT

*The emerging Model-Driven Engineering* (*MDE*) *paradigm advocates the use of models as first-class citizens in the software development process, while artifacts such as documentation and source-code can be quickly produced from those models by using automated transformations. Even though many MDE-oriented approaches, languages and tools have been developed in the recent past, there is no standard that concretely defines a specific sequence of steps to obtain a functional software system from a model. Thus, the existing approaches present numerous differences among themselves, because each one handles the problems inherent to software development in its own way. This paper presents and discusses a reference model for the comparative study of current MDE approaches in the scope of web-application development. This reference model focuses on relevant aspects such as modeling language scope* (*domain, business-logic, user-interface*), *usage of patterns, separation of concerns, model transformations, tool support, and deployment details like web-platform independence and traditional programming required. The ultimate goal of this paper is to determine the aspects that will be of greater importance in future web-oriented MDE languages.*

***Keywords*: *Model-Driven Engineering, Web Engineering, Software Development***

## 1. Introduction

The worldwide expansion of the Internet in the last years has made it a powerful platform for the deployment of a variety of artifacts and systems. This has led to the appearance of a myriad of frameworks and libraries that attempt to harness the power of Internet-based technologies in order to accomplish various objectives. Typical examples of widely-used web-application frameworks include Microsoft's ASP.NET [1], Sun's Java EE [2], PHP (PHP: Hypertext Preprocessor) [3], Ruby on Rails [4,5], Django [6], or Catalyst [7].

This paper proposes a reference model for the analysis and comparison of MDE (Model-Driven Engineering) approaches to web-application development. This reference model is focused on particularly relevant aspects such as the scope of modeling language scope (*i.e.*, if it addresses modeling of domain, business-logic, and user-interface), separation of concerns, time-saving features such as usage of patterns and/or model-to-model transformations, tool support available, and deployment aspects like web-platform independence and whether the approach still requires traditional programming (*i.e.*, development of source-code). The goal of this reference model is to identify concepts, principles and best practices that are likely to become important in future model-driven web-application languages.

It should be noted that, although currently there are not many MDE-oriented approaches to web-site or web-application development, there are some approaches – the analysis of which was not included in this paper due to length constraints – that we believe should be mentioned. Of those, we highlight WebML (Web Modeling Language) [8,9], UWE (UML-based Web Engineering) [10], XIS2 (eXtreme Modeling Interactive Systems) [11,12], the OutSystems Agile Platform [13], OOHDM (Object-Oriented Hypermedia Design Model) [14], or OPM/Web (Object Process Methodology for Developing Web-Applications) [15]. We also point out that there are other studies and tools regarding MDE-oriented approaches and languages, not directly related to web-application development, but rather to other important aspects such as 1) the usage of meta-metamodels [16], 2) user-interface modeling [17-21], or 3) the usage of prototyping tools to provide a way for stakeholders to draw and communicate

design ideas, as is the case with Microsoft Sketchflow [22].

This paper is organized as follows. Section 1 introduces the context of web-applications and frameworks. Section 2 describes the reference model that we defined for the analysis and comparison of MDE-oriented web-application development approaches. Section 3 provides a discussion regarding this reference model; this discussion also features some examples from web-application modeling approaches that we have analyzed in our research, and to which we have applied this reference model. Finally, Section 4 concludes this paper.

## 2. Reference Model

This reference model is defined according to a set of aspects that are relevant to MDE-based development, namely(see **Table 1**): 1) the language used; 2) the usage of model-to-model transformations; and 3) the tool support for the design, generation, and deployment of the web-application.

### 2.1 Modeling Language

The language used by the approach is analyzed in terms of the meta-metamodel used (if any), and whether it addresses a set of modeling concerns, such as domain modeling, business-logic modeling, and user-interface modeling.

*Meta-metamodel*. Besides the web-application modeling language, it is important to identify the language's meta-metamodel (if any) and the modeling elements that it provides. We consider that this aspect is important because: 1) this meta-metamodel's expressiveness can affect the number of elements that the language itself can provide, which in turn can have a direct impact on the language's own expressiveness and its adequacy to solve complex web-application problems; and 2) the meta-metamodel used by the language can have a profound influence on the tool support that is – or may become – available to the approach, e.g., languages specified as a UML 2.0 profile [23] are likely to be supported by the majority of UML modeling tools that support the Profiling mechanism.

*Domain Modeling*. Domain modeling concerns the identification of problem-domain concepts, and their representation using a modeling language (e.g., UML diagrams). This aspect is analyzed regarding: 1) whether it is independent of persistence or user-interface details (*i.e.*, domain models do not need to be "adjusted" to support those layers); and 2) the elements provided by the language, such as enumerations (which are useful to avoid specifying multiple concepts with no particular differences between themselves) or associations (namely their arity, *i.e.*, the number of possible associated entities).

*Business-Logic Modeling*. Although the definition of business-logic modeling can be considered somewhat subjective, in this work we consider it as the specification of the web-application's behavior. This aspect is analyzed regarding the following subjects: 1) whether it supports querying and manipulating domain concepts in a either textual or graphical manner, e.g., using SQL-like statements; 2) whether this querying and manipulation support is "low-level", in a manner similar to traditional source-code; 3) support for process specification; and 4) usage of patterns, such as the typical "create-read-update-delete" (CRUD) set of business-logic patterns. It should be noted that the "low-level support" subject is considered relevant because it often reflects the expressiveness of the language: typical source-code-oriented languages (such as C or C#), although complex, are nevertheless very expressive.

*Navigation Flow Modeling*. The approach's support for specifying the navigation flow (in the context of the modeled web-application) between different HTML pages, or even inside HTML pages, is also analyzed in this issue.

*User-Interface Modeling*. Another important aspect is the approach's support for specifying/modeling the user-interface (UI). The subjects analyzed are the following: 1) whether the UI modeling language is platform-independent (*i.e.*, does not require specific software to present the UI); 2) supports access-control specification (*i.e.*, certain controls are shown/hidden according to the authenticated user); 3) allows the definition of custom UI elements; 4) allows the usage of interaction patterns (e.g., create, edit, associate/dissociate); and 5) supports binding between UI elements and domain model elements.

### 2.2 Model-to-Model Transformations

This subsection examines whether the approach uses (or even considers) the usage of model-to-model transformations. This kind of transformations is typically used to accelerate the task of designing the web-application, by using model analysis and inference mechanisms to automatically specify some parts of the web-application model, thus releasing the model designer from some repetitive and error-prone tasks.

### 2.3 Tool Support and Deployment

This subsection analyzes the tool support that is available for the approach, regarding the following aspects.

*Modeling Tool*. This aspect determines whether the approach (and its language) is supported by a design tool, and if that design tool is proprietary (*i.e.*, if it is built specifically for supporting the approach), or if it is a generic tool, adapted to specific details of the language.

*Need for Traditional Programming*. This aspect analyzes if the tools supporting the approach allow the generation of (parts of) the web-application, and whether the generated application is complete (*i.e.*, it does not require the manual implementation of specific features by programmers).

*Deployment Environment*. Finally, this aspect determines the target platform(s) considered by the approach, and whether those target platforms are proprietary (in other words, if there is supposed to be a tight coupling between the approach and the target platform).

## 3. Discussion

This section provides a discussion regarding the criteria described in Section 2. Some of the arguments that we present in this discussion are based on our own application of this reference model to some MDE-oriented web-application development approaches (such as WebML and UWE), while others are based on our own previous experiences in this field of study (such as XIS2).

### 3.1 Modeling Language

It is frequent for this kind of development approaches to define a graphical modeling language, sometimes with textual elements involved (e.g., OCL constraints in UWE). Nevertheless, the language's users can certainly benefit from having textual specifications assisted by graphically-oriented features, such as using drag-and-drop to place such textual elements.

Regarding syntax, UML-based approaches and languages tend to use UML's graphical syntax (e.g., boxes for representing classes or enumerations, a "stickman" figure for representing a stakeholder), mainly due to their nature as UML-extending metamodels and as UML profiles. On the other hand, other languages typically define their own syntax, albeit some parts of those syntaxes are usually based on existing modeling languages, with the objective of facilitating their learning process.

Finally, regarding semantics, the difference between UML-based and non-UML-based languages is also quickly noticeable. The semantics of most UML-based languages are defined verbally – in their respective language specification documents – with some OCL restrictions defined; validation of models specified using these languages is dependent on the expressiveness of OCL and the modeling tool's support for OCL restrictions. On the other hand, the semantics of other languages, although not specified in a "formal" manner, are supported by custom-built tools that can easily address all aspects of the language's semantics, because of those tools' use of low-level programming languages. Nevertheless, we consider that this difference is present mainly because of the compromise, that each approach must assume, between the "tool support" and "model exchangeability" factors: 1) for "real-world" languages (with a high degree of complexity), it is usually easier to provide *adequate* modeling and validation facilities via a custom tool with a low-level programming language, than by a UML generic modeling tool with a "UML Profile + OCL restrictions" customization mechanism; 2) however, due to their typically academic nature, UML-based approaches are also concerned with exchanging models between tools, a concern not shared by commercially-oriented approaches, which accounts for the choice to use UML in such approaches.

*Meta-metamodel*. We have noticed that commercial-oriented approaches usually define their languages without using a standard meta-metamodel, while academic approaches tend to use existing meta-metamodels (e.g., UML [23]) as the basis for their web-application modeling language.

This is possibly due to the common belief that the meta-metamodels available today, such as UML or MOF, are too complex and attempt to address too many problems (which is one of the major factors that drive the discipline of Domain-Specific Modeling [24]). Considering that commercial approaches are supposed to be practical and "easy to use", the creators of such approaches are likely to opt for the creation of a language from scratch (even if this involves some initial extra effort), in order to avoid dealing with issues due to the inherent complexity of existing meta-metamodel languages.

On the other hand, academic approaches are typically more focused on obtaining and divulging results that contribute to advance the state-of-the-art, while aspects such as simplicity and "ease of use" are usually considered secondary. Thus, in this kind of environment, using a standard meta-metamodel is not only recommended, but it is also a way to ensure that other researchers can quickly build upon those results and advance the state-of-the-art even further.

*Domain Modeling*. Domain modeling is a subject that *must* be addressed by any application development approach. This is because an application, with the purpose of solving a certain problem, will undoubtedly manipulate a set of entities, directly or indirectly related to that problem; these entities, in turn, are described by a domain model.

Most software development approaches and best practices advocate the independence between the domain model and other parts of the application functionality (e.g., separating the domain model from persistence or UI layer details). However, most languages usually end up requiring that the domain model be "adjusted" to UI- or persistence-oriented details (e.g., in the Address Book example at the UWE tutorial [25], the "Address Book" class has an attribute "introduction", in order for the main screen of the application to show a small introductory message to the user). Although from a theoretical perspective this can be regarded as a bad thing, in practice it does have the advantage of endowing the designer with a greater level of control over the web-application implementation itself, instead of just relying on possibly fallible automated mechanisms. Nevertheless, this advantage can also be obtained by the technique of adding modeling elements, outside of the domain model, to establish the mappings between the domain model elements and the elements of other models.

Also, most languages usually provide some degree of support for important concepts like enumerations and associations. Associations are obviously important, as they enable the definition of relationships between entities. On the other hand, enumerations are also important because they enable the a-priori definition of "instances" while avoiding a potential explosion of entity types. UML-based approaches can inherit support for these concepts automatically; other languages typically provide support for associations, but enumerations are seldom supported (e.g., WebML does not support enumerations, but the OutSystems Agile Platform does).

*Business-Logic Modeling*. Business-logic modeling is usually addressed by this kind of modeling approaches, although how this is handled varies greatly according to the approach. Modeling features typically found are the usage of pre-defined business-logic patterns, the possibility for designers to define their own business-logic patterns, and the deference of business-logic specification to traditional, source-code-oriented, development. However, most approaches frequently adopt a combination of these possibilities: XIS2 provides the typical create-read-update-delete operations automatically, and the designer can add more operations; on the other hand, the OutSystems Agile Platform only supports its pre-defined operations, although the wide variety of operations that it supports should be adequate for most of the business-logic that the designer would want to specify.

A relevant issue that should be mentioned here is the trade-off between *abstraction* and *expressive power*. Although languages such as XIS2 or WebML try to raise the abstraction level at which designers have to think and specify their models (as opposed to just developing the web-application in a manual source-code-oriented fashion), the fact is that this higher abstraction usually also affects the expressive power available to designers in a negative manner. On the other hand, approaches that are not particularly oriented towards raising the abstraction level (e.g., business-logic modeling in the OutSystems Agile Platform is intended to provide *developers*, who have experience in dealing with web-application-related concepts such as "request parameters", "session values" or "asynchronous requests", with a graphical environment for them to do their tasks) suffer almost no impact on the expressive power available to business-logic modeling.

*Navigation Flow Modeling*. Navigation flow modeling is an aspect that is fundamental to any kind of web-application, and so these development approaches must address it. Due to their "web-application modeling" nature and all that this implies (e.g., the existence of HTML pages, hyperlinks to navigate between pages, using request parameters or a similar mechanism to provide necessary parameters), this kind of development approach typically all follow the same guidelines: a directed graph is drawn, where nodes correspond to HTML pages, and edges correspond to the possible navigation flows that are available to the user within the context of a certain page.

The difference between most approaches, however, lies in whether the designer can specify the sets of edges (*i.e.*, actions or links) available in each node. An example of this difference can be found in XIS2 and WebML: in XIS2 the designer can specify actions, associated to UI elements in a page, and the navigation flows will afterward be associated with the corresponding page's actions, while in WebML each Data-Unit node has a well-defined set of links, for input and output, and so the designer cannot specify additional links.

*User-Interface Modeling*. Most web-application development approaches address UI modeling in a graphical manner, using a WYSIWYG (What You See Is What You Get) approach. However, there are many variations on what can be modeled in the UI. For example, the WebML language, as described in [26], only allows the specification of *what* elements will be present on the page, but not *where* they will be located (although its commercial tool does provide support for this kind of UI modeling). On the other hand, other approaches (such as the OutSystems Agile Platform or XIS2) do allow the specification of the location of such elements.

An issue that should be mentioned, related to WYSIWYG-like UI modeling, is the possibility of using and or capturing UI patterns to address the UI's behavior. From our experiences in using this reference model, we have noticed that most approaches (such as XIS2, WebML and the OutSystems Agile Platform) usually address this behavioral aspect through the capture of UI patterns (e.g., associate/dissociate, list, create item), enabling applications to *interact* with users, instead of just displaying requested resources.

An aspect where most approaches actually differ between themselves is the possibility of reusing page or interface components. As an example, WebML and OutSystems' Platform support this feature, while XIS2 or UWE do not. Nevertheless, we consider this to be a very important feature, as it allows designers to specify certain screen sections only once, and "import" those sections into some/all of the application's screens, with the obvious added advantage that changes to such a section need to be done only in a single point in the model; a good example of such a component would be a web-site banner, or a web-site contents' navigation tree.

Regarding platform-independence, we believe that there is not much to be said, considering that these approaches have to generate regular HTML to be sent to a web-browser, and so can be considered as target-platform-independent.

Finally, it is very frequent for these approaches to allow binding UI elements to domain elements (e.g., configure the rows of a table to show the values of field in specific instances). However, these approaches usually differ on

whether they allow the customization of those bindings in the model (e.g., change a cell in a table row to show a different value for each instance). Although not being able to customize these bindings can certainly simplify the modeling task and avoid designer errors, in practice this is also a limitation on the expressiveness of the language, which can force developers to change generated source-code in order to address specific requirements.

### 3.2 Model-to-Model Transformations

From our own experience regarding both academic and non-academic web-application tools and approaches, support for model-to-model transformations is an aspect seldom addressed by these approaches, and typically found only on the UML-based ones. This is possibly because model-to-model transformations are typically implemented in such a way that user-defined information on the destination model may be lost. Additionally, most approaches use a single modeling language, and so the information that would be present in the destination "model" can be inferred from the information that has already been modeled.

The approaches that do consider the usage of model-to-model transformations typically use them to accelerate modeling tasks, by taking the information already modeled at the time the transformation is performed and generating new views. These transformations reflect how those views would *typically* be modeled, and the designer is free to change the generated views to suit the application's needs (e.g., showing an extra field in the UI, or adding an extra step in an activity diagram).

### 3.3 Tool Support and Deployment

For any web-application modeling approach to actually be usable, it has to provide some kind of tool support.

*Modeling Tool*. Tool support is an aspect typically influenced by the meta-metamodel used (if any), and by the academic/commercial orientation of the approach. Academic approaches based on UML are usually tool-agnostic and are supposed to be usable in any UML modeling tool that supports the Profiling mechanism. On the other hand, commercial approaches provide their own proprietary (and language-specific) modeling tools; nevertheless, it is sometimes possible to specify UML profiles that enable the modeling of such languages in regular UML modeling tools (albeit in a somewhat limited fashion). An example of this can be found in [27], which defines a WebML-oriented UML profile.

It should be noted, however, that such language-specific tools have a great advantage over "generic" tools, as they can assist the user throughout the entire development life-cycle (by means of mechanisms such as contextual help or helpful validation tips), in a manner that is well adjusted to the approach.

*Need for Traditional Programming*. Although it would certainly be desirable that a modeling approach required no programming efforts (for reasons such as avoiding programming errors, or reduced software development costs), the fact is that most web-application modeling approaches typically consider typical programming tasks (*i.e.*, manual editing of generated source-code artifacts) as an activity to occur during the development life-cycle, in order to account for particular requirements that may not be expressible by the approach's modeling language. Of the approaches that are known to us, only the OutSystems Agile Platform considers that source-code should not be edited in a manual fashion: designers/developers can *only* edit the model itself, and generated code and databases are always kept "behind-the-scenes".

This is an aspect that clearly illustrates one of the consequences of the traditional MDE question "how expressive should the modeling language be?" This question is actually a dilemma because increasing the expressiveness of a language typically involves adding elements to it, in order to address more semantic possibilities, which in turn can increase the difficulty for a new designer to learn the language, as well as possibly reducing its level of abstraction. On the other hand, if the language does not provide many elements, it is easier to learn, but it may also be unable to specify some desirable features. So, current modeling languages are actually the result of a "balanced compromise" between these factors, always taking into account the purpose of the language.

*Deployment Environment*. Most web-application modeling approaches are actually independent of the deployment environment, because they do not target a specific technology or platform (e.g., MySQL database, Apache web-server); nevertheless, although the concepts in the languages of some approaches may pose some technological restrictions, developers are sometimes able to use "workarounds" to remove such restrictions (e.g., replacing a Java servlet with an adequate ASP.NET class).

As an example of this kind independence, we can point out WebML, UWE and XIS2: because of their usage of high-level elements, they can generate code for any web-oriented platform (e.g., ASP.NET, Apache Tomcat); in fact, XIS2 can also generate code for desktop-based platforms. However, models that are created in the OutSystems Agile Platform can only be deployed to the OutSystems deployment stacks, which use either 1) the JBoss application server and Oracle database, or 2) Microsoft's IIS application server and SQL Server Express database.

Nevertheless, we consider it important to note that, although in the deployment aspect the Agile Platform is apparently much more limited than other approaches, this "disadvantage" is actually what allows the Agile Platform to automate most of the web-application development life-cycle, namely deployment and application upgrade/rollback scenarios, something that is clearly lacking in

most of the web-application modeling approaches that we know.

## 4. Conclusions

Most web-applications are still created in a traditional manner, by manually developing source-code, which is a slow and error-prone task. This is a problem, not specific to just web-applications but to the software engineering discipline in general, that MDE aims to address. Furthermore, the web-based development paradigm introduces a set of Internet and HTTP-related concepts that have to be addressed by the designer (e.g., page, navigation, request, GET, POST).

In this paper, we proposed a reference model for comparing current MDE-driven approaches for web-application development, according to relevant criteria such as support for domain, business-logic, UI and navigation-flow modeling, as well as usage of model-to-model transformations and tool support. **Table 1** summarizes the key issues of the proposed reference model. Finally, this paper discussed the defined reference model.

Some reflections can be made regarding the criteria identified in this reference model. The first aspect concerns the *compromise between language expressiveness, simplicity, and learning curve* for new designers. Although the *purpose* of the language must always be a factor to weigh in on this compromise, the fact is that very expressive languages can be harder to learn because of their great number of elements than languages with few elements, which are typically simpler. Moreover, if a creator of a language starts increasing its expressiveness by adding new elements, it is likely that its level of abstraction over another language is also diminished, because those new elements must reflect particular semantic possibilities of the lower-level language. Note that we say "very expressive languages" instead of "languages with many elements". This is on purpose, as a language can provide many elements, and still not be very expressive; in these cases, some of those elements have exactly the same semantic meaning but different syntax, what we typically call *syntactic sugar*. Java and C# e.g., have about the same level of expressiveness, although C# provides a greater number of elements (e.g., delegates).

Another aspect concerns the *compromise between tool support, and integration with the deployment platform*. Although, in theory, using a generic tool to create models and generate the desired source-code is a good idea, in practice it actually increases the difficulty of obtaining a *working system* from the model. A good example of this can be found in the OutSystems Agile Platform vs. a generic UML-based modeling approach such as XIS2: the latter requires the generation, compilation and deployment of artifacts (clearly a task to be performed by a programmer or a technical user), while the former handles all these deployment aspects in a manner completely transparent to the designer.

**Table 1. Main issues of the proposed reference model**

| Analysis Aspects | | | Examples of Possible Analysis Options |
|---|---|---|---|
| **Modeling Language** | **Meta-metamodel** | | UML 2.0 (Profile)<br>None |
| | **Modeling** | **Domain Modeling** | Independence of database<br>Independence of UI<br>Support for enumerations and associations |
| | | **Business-Logic Modeling** | Support for domain model query and manipulation<br>Usage of business-logic patterns |
| | | **Navigation Flow Modeling** | Designer-defined navigation links between pages |
| | | **User-Interface Modeling** | Custom interface elements<br>Designer-defined UI patterns |
| **Model-to-Model Transformations** | **Modeling levels concerned** | | PIM-to-PIM<br>PIM-to-PSM |
| | **Transformation languages** | | Programming-language specific<br>Independent of programming-language (e.g., QVT) |
| **Tool Support and Deployment** | **Modeling Tool** | | None<br>Integrated into (or an extension for) a generic CASE/IDE tool<br>Language-specific tool |
| | **Need for Traditional Programming** | | None<br>Yes |
| | **Deployment Environment** | | Platform-independent<br>Targets specific platform |
| | | | Platform-independent<br>Targets specific platform |

Finally, although there is work regarding MDE approaches for the development of web-applications, it does not address platforms of a more specialized nature, such as CMS (Content Management System) or DMS (Document Management System) platforms. Although this is understandable, as these platforms provide additional concepts and supporting them would introduce even more problems (e.g., what to do when a language element can not be mapped to an implementation concept), we consider that it also constitutes additional motivation to *use a set of languages and the corresponding model-to-model transformations*: each language could address a specific kind of platform, and the model-to-model transformations would be responsible for mapping concepts between those languages/platforms.

## REFERENCES


[1] "The Official Microsoft ASP.NET Site". http://www.asp.net

[2] "Java EE". http://java.sun.com/javaee

[3] "PHP: Hypertext Preprocessor". http://www.php.net

[4] "Ruby Programming Language". http://www.ruby-lang.org

[5] "Ruby on Rails". http:// rubyonrails.org

[6] "Django–The Web framework for perfectionists with deadlines." http://www. djangoproject.com

[7] "Catalyst–Web Framework". http://www.catalystframework.org

[8] http:// www.webml.org

[9] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai and M. Matera, "Designing Data-Intensive Web Applications," Morgan Kaufmann, 2003.

[10] "UWE–UML–Based Web Engineering". http://uwe.pst.ifi.lmu.de

[11] A. R. Silva, J. S. Saraiva, R. Silva and C. Martins, "XIS-UML Profile for eXtreme Modeling Interactive Systems," *4th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, IEEE Computer Society, Los Alamitos, March 2007, pp. 55-66.

[12] A. R. Silva, J. Saraiva, D. Ferreira, R. Silva and C. Videira, "Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools," "On the Interplay of .NET and Contemporary Development Techniques," *IET Software Journal*, Vol. 1, No. 6, December 2007, pp. 294-314.

[13] "Agile Software Development and Management," OutSystems. http://www.outsystems.com/agile

[14] "Object-Oriented Hypermedia Design Model". http://www.telemidia.puc-rio.br/oohdm/oohdm.html

[15] I. B. Reinhartz and D. Dori, S. Katz, "OPM/Web–Object–Process Methodology for Developing Web Applications," *Annals of Software Engineering*, Vol. 13, No. 1-4, 2002, pp. 141-161.

[16] J. S. Saraiva and A. R. Silva, "Evaluation of MDE Tools from a Metamodeling Perspective," *Journal of Database Management*, Vol. 19, No. 4, October-December 2008, pp. 21-46.

[17] W. Kozaczynski and J. Thario, "Transforming User Experience Models to Presentation Layer Implementations," *Proceedings of the Second Workshop on Domain-Specific Visual Languages*, Seattle, November 2002.

[18] P. P. Silva and N. W. Paton, "User Interface Modeling in UMLi," *Institute of Electrical and Electronic Engineers Software*, IEEE, Vol. 20, No. 4, July 2003, pp. 62-69.

[19] J. Van den Bergh and K. Coninx, "Towards Modeling Context-sensitive Interactive Applications: the Context-Sensitive User Interface Profile (CUP)," *SoftVis'*05: *Proceedings of the* 2005 *ACM Symposium on Software Visualization*, ACM, New York, 2005, pp. 87-94.

[20] P. Azevedo, R. Merrick and D. Roberts, "OVID to AUIML-User-Oriented Interface Modelling," *Proceedings of* 1*st International Workshop, Towards a UML Profile for Interactive Systems Development*, York, October 2000.

[21] N. J. Nunes and J. F. Cunha, "Towards a UML profile for interaction design: the Wisdom approach," *Proceedings of* 1*st International Workshop, Towards a UML Profile for Interactive Systems Development*, York, Springer Verlag, October 2000, pp. 101-116.

[22] "Microsoft Expression: Sketchflow Overview". http://www.microsoft.com/expression/products/SketchflowOverview.aspx

[23] "Unified Modeling Language: Superstructure–Specification Version 2.0," Object Management Group, August 2005. http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf

[24] "DSM Forum: Domain-Specific Modeling". http: //www.dsmforum.org

[25] "UWE–Tutorial". http://uwe.pst.ifi.lmu.de/teachingTutrial.html

[26] M. Brambilla, "The Web Modeling Language," Politecnico di Milano. http://home.dei.polimi.it/mbrambil/webml.htm

[27] N. Moreno and P. Fraternalli and A. Vallecillo, "WebML Modeling in UML," *Institution of Engineering and Technology Software*, Vol. 1, No. 3, June 2007, pp. 67-80.

*J. Software Engineering & Applications*, 2010, 3: 426-435
doi: 10.4236/jsea.2010.35048 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Mapping UML 2.0 Activities to Zero-Safe Nets

**Sabine Boufenara[1], Faiza Belala[1], Kamel Barkaoui[2]**

[1]LIRE Laboratory, Mentouri University of Constantine, Algeria; [2]CEDRIC-CNAM, Rue Saint-Martin, Paris, France.
Email: sabineboufenara@yahoo.com, belalafaiza@hotmail.com, barkaoui@cnam.fr

Received February 18th, 2010; revised April 6th, 2010; accepted April 6th, 2010.

## ABSTRACT

*UML 2.0 activity diagrams (ADs) are largely used as a modeling language for flow-oriented behaviors in software and business processes. Unfortunately, their place/transition operational semantics is unable to capture and preserve semantics of the newly defined high-level activities constructs such as Interruptible Activity Region. Particularly, basic Petri nets do not preserve the non-locality semantics and reactivity concept of ADs. This is mainly due to the absence of global synchronization mechanisms in basic Petri nets. Zero-safe nets are a high-level variant of Petri nets that ensure transitions global coordination thanks to a new kind of places, called zero places. Indeed, zero-safe nets naturally address Interruptible Activity Region that needs a special semantics, forcing the control flow by external events and defining a certain priority level of executions. Therefore, zero-safe nets are adopted in this work as semantic framework for UML 2.0 activity diagrams.*

***Keywords*: *UML Activity Diagrams Formalization, Interruptible Activity Region, Zero-Safe Nets***

## 1. Introduction

The Unified Modelling Language (UML) [1] has recently undergone a significant upgrade of its basic concepts, giving rise to a new major version, namely UML 2.0. Being widely used for specification and documentation purposes in the software development process**,** UML offers a spectrum of notations for capturing different aspects of software structure and behaviour. Activity diagram (AD) notations are intended to model behavioural aspects of software systems, particularly control and data flows.

Activity Diagrams (ADs) are widely used to model various types of applications fluctuating from basic computations to high level business processes, embedded systems and system-level behaviors. They facilitate the modelling of control and object (or data) flows by introducing a multitude of new concepts and notations such as collections, streams, loops and exceptions. Several semantics models have been defined to support these concepts. Nevertheless, many problems persist and reduce the usability of ADs [2,3]. This is mainly due to the new constructs and principles complexity and their formal semantics lack, leading to inconsistent interpretations of the model. For example, in a workflow process, described in terms of tasks and execution orders between them, Termination (or Cancelation) concept may be modeled via ADs Interruptible Activity Region. This modelling may have several interpretations since the used modelling concepts are still informal. Thus, a large gap has to be bridged prior to obtain an execution model and automated reasoning.

The abstract semantics of ADs have also completely changed in UML 2.0. They are no longer considered as a kind of state-machine diagrams and their semantics is being well explained in terms of Petri net concepts. But, basic Petri nets do not preserve semantics of new constructs of UML 2.0 ADs. We believe that this is essentially due to the locality character (local activations at transitions enabling) of basic Petri nets, whilst in UML 2.0, contrary to UML 1.x, the activation of computational steps may be not local.

Many attempts are currently led to give UML 2.0 ADs an operational semantics via some well known formal models such as high-level Petri nets [4], Abstract State Machine [5] and so on, for eventual analysis and simulation purposes. The objective of this paper is to describe how Zero Safe Nets (ZSNs for short) are very suitable to handle semantics of UML 2.0 ADs Interruptible Activity Region. ZSN is a new variant of Petri-net model introduced by Bruni [6] to define synchronization mechanisms among transitions without introducing any new interaction mechanism. On the basis of this formalism, we suggest a set of mapping rules to define a formal semantic of UML 2.0 ADs complex constructs. ZSNs semantics is then used to conduct control flow in the net guarantying atomicity and isolation of a transaction that is all what we need in the cancellation schema. This formal specification

is precise enough to enable a unique model interpretation at an utmost detail level. It can therefore serve as tools implementation basis. Finally, it participates to ensure that the specified behaviour meets the intended intuition of the modeller.

The remainder of this paper is structured as follows. A detailed discussion of related works is given in Section 2. Section 3 presents syntax and informal semantics of both Interruptible Activity Region in UML 2.0 ADs and ZSNs. In Section 4, we describe our contribution by presenting first the problematic, then the intuitive mapping to address the Interruptible Activity Region formalization and finally the formal definition of this mapping. Section 5 concludes the paper with remarks and outlook on future work.

## 2. Related Works and Paper Contribution

The state of the art concerning UML 2.0 activity diagrams semantics covers three different approaches: the first based on Petri-nets, the second using graph transformation rules and the third generating pseudo-code. Since the two latter approaches are out of the scope of the present paper, we therefore discuss UML 2.0 semantics related work only in terms of Petri-nets.

Since UML specification envisions a "Petri-like semantics" for activity diagrams, it is quite interesting to propose a mapping between the two notations. Barros [7] suggests translating a subset of ADs concepts to Petri nets ones. Actions considered as activities in Petri nets are no longer atomic, inducing to ADs semantics violation in the UML specification standard ([1] p. 203). Moreover, only locally behaved activities are considered in Petri nets, whereas non-locality semantics is one major innovative characteristic of UML 2.0 ADs.

In [8], an AD is transformed into FMC (Fundamental Modeling Concepts) for their attractive feature, and then, a Colored Petri net is constructed for execution and validation purposes. This approach focuses on abstract syntax and thus, does not preserve semantics, especially for the atomicity principle.

Störrle uses different variants of Petri nets (from colored to procedural and exception Petri nets) to propose a formal semantics to UML 2.0 ADs. The author tackles the formalization of many concepts [9-12] such as control-flow, procedure calling, data-flow, exceptions, loop-nodes, conditional-nodes and expansion-regions using various versions of Petri nets. However, different concepts can generally coexist in the same AD. Therefore, analysis of the whole system behavior is not possible due to non-unified formalism.

The development culminates in [13-15] concluding that Petri-nets might, after all, not be appropriate for formalizing activity diagrams. Especially, mapping advanced concepts, such as interruptible activity regions, is found to be not intuitive. Moreover, Petri-nets formalization of ADs concepts is not unified and integrates different variants of Petri-nets to map concepts belonging to the same diagram. Additionally the traverse-to completion semantics insurance is identified as being the major problem in Petri-nets mapping. In [16], we have proposed a generic mapping from UML 2.0 ADs to Zero-Safe Nets (ZSNs) and have shown by several examples how this Petri net variant can surmount this latter problem. Indeed, non-locality semantics of ADs is preserved via a global synchronization, offered by ZSNs, rather than a local one as in basic Petri nets. In [17], we have been interested by streaming parameters and exception outputs constructs in UML 2.0 ADs, their formal semantics has been defined in terms of these Petri nets variant without losing an important characteristic of those concepts that is atomicity.

In this paper, we extend these recent works by improving the proposed mapping to be more formal and general. Indeed, we examine semantics of Interruptible Activity Region construct in which actions need to be promptly cancelled on the reception of an external event. This can not be provided with basic Petri nets that are local and not reactive.

## 3. Basic Concepts

We are interested in this section to remind fundamental notions used in this study. For more details, the reader can consult [1,18] (for Interruptible Activity Region in ADs) and [6] (for ZSNs).

### 3.1 Interruptible Activity Region

The UML 2.0 specification made by the OMG [1] standard provides a meta-model to define the abstract syntax for activity diagrams including Interruptible Activity Region.

These ADs special regions are groups of nodes where all execution might be terminated, if an edge traverses an interruptible activity, before leaving the region. Interrupting edges must have their source node in the region and their target node outside it, but in the same AD containing the region.

An Interruptible Activity Region is notated by a dashed, round-cornered rectangle drawn around the nodes contained in the region. An interrupting edge is notated with a lightning-bolt activity edge.

During the process of an Interruptible Activity Region, the reception of an event (exception-event) triggers the block abort of that part of the Activity, and resumes execution with another action that may be the exception-handler. The standard specification [1] states, that "When a token leaves an interruptible region via edges designated by the region as interrupting edges, all tokens and behaviours in the region are terminated". Interruptible regions are introduced to support more flexible non-local termination of flow.

**Example 1**. **Figure 1** gives an example taken from [1] illustrating these concepts. This example illustrates that if an order cancellation request '*Order Cancel Request*' is made, while executing one of the actions (receive, fill, or ship orders), the '*Cancel Order*' behaviour is immediately invoked and the action being executed is aborted.

Cancellation is a very common behavior in the execution of workflows process. It is used to capture the interference of an event or an activity in other activities execution of a workflow preventing execution or termination. A cancellation can involve a cancellation area, a sub process or an entire workflow. In UML 2.0 ADs, Interruptible Activity Region has been defined to hold such behavior.

### 3.2 Introducing ZSNs

Zero-safe nets have been introduced by Bruni in [6] to define synchronization mechanism among transitions, without introducing any new interaction mechanism besides the ordinary token-pushing rules of nets. Their role is to ensure the atomic execution of complex transitions collections, which can be considered as synchronized.

Atomic execution of multiple coordinated transitions is forth possible in ZSNs thanks to a new kind of places, called zero places. From an abstract viewpoint, those transitions will appear as synchronized. Zero places are bound to zero tokens in a system observable state. A token in a zero place is equivalent to a system internal state that is non-observable. ZSN synchronized evolution must begin at an observable state, evolve in non-observable markings and must end at an observable state. Therefore, ZSNs define two sorts of places; stable places corresponding to net places, and zero places. A ZSN evolution is considered as a transaction. A stable token generated in a transaction is frozen all over the evolution; it is released only once the transaction is finished. We notice that a transaction in this case, is represented by a system activity possibly composed of a set of concurrent but atomic sub-activities.

Zero places coordinate the atomic execution of transi-

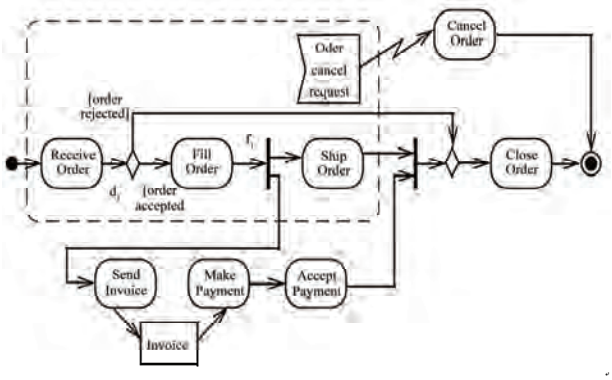



**Figure 1. Interruptible Activity Region enclosed by a dashed line. The Interrupting edge is expressed by a lightning-bolt style**

tions which, from an abstract viewpoint, will appear as synchronized. At the abstract level, we are not interested in observing the hidden state. Modeling of the well- known example of 'dining philosophers' problem' is sufficient to show how ZSNs are powerful to synchronize transitions in an atomic way (see [19] for more details).

**Example 2.** (taken from [19]). There are *n* philosophers (here, we suppose $n = 2$) sitting on a round table; each having a plate in front and between each couple of plates there is a fork, with a total of *n* forks on the table. Each philosopher cyclically thinks and eats, but to eat he needs both the left hand side fork and the right hand one of his plate. After eating a few mouthfuls, the philosopher puts the forks back on the table and starts thinking again.

It is not difficult to imagine conflict situations leading to a deadlock when each philosopher takes one fork and cannot continue. This is due to the fact that the coordination mechanism is hidden inside transitions ($Take_1$ and $Take_2$) that are too abstract (see **Figure 2(a)** modeling a centralized non-terminism). Places $Fk_i$ denote forks. A token in the place $Fk_i$ means that the ith fork is on the table.

The same model is redrawn using free choice nets. Decisions are now local to each place *i.e.* decisions are made independently (see **Figure 2(b)**) and deadlock situation is clear. One decision concerns the assignment of the first fork whether to the first or to the second philosopher, the other decision concerns the assignment of the second fork. Note that $Ch_{i,j}$ stands for Choice*i,j* where *i* denotes $Fork_i$ and *j* denotes $PHilosopher_j$. Then, it might happen that the first fork is assigned to the first philosopher ($Ch_{1,1}$) and the second fork is assigned to the second philosopher ($Ch_{2,2}$), and in such case the free choice net deadlocks and none of the $Take_i$ actions can occur. Thus, the translated net admits non-allowed computations in the abstract sub-system of **Figure 2(a)**.

Zero-safe nets surmount this deadlock problem by executing only some atomic transactions, where tokens produced in low-level resources are also consumed. In the example, the ow-level. This is possible, but at the expense of preseinvisible resources consist of places $Fk_{i,j}$ for $1 \le i, j \le 2$, that can be interpreted as zero places. In this way the computation performing $Ch_{1,1}$ and $Ch_{2,2}$ is forbidden, because it stops in an invisible state, *i.e.*, a state that contains zero tokens (see **Figure 2(c)**).

While basic Petri nets fail to conserve the system semantics at a low-level, free choice nets make local decisions possible at lrving execution semantics. Zero-safe nets are able to preserve execution semantics even when expressed in refined way.

**Formal Definitions [6]**

A ZSN is a 6-tuple $B = (S_B, T_B, F_B, W_B, u_B, Z_B)$ where $N_B = (S_B, T_B, F_B, W_B, u_B)$ is the underlying place/ transition

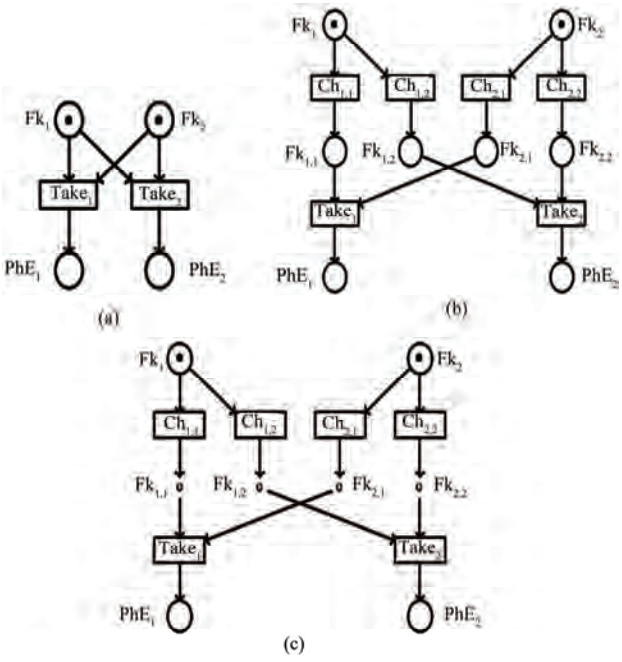



**Figure 2. Example of dining philosophers: (a) Centralized nondeterminisme, (b) Local nondeterminisme presenting deadlock, (c) Atomic free choice**

net. $S_B$ is a non-empty set of places. $T_B$ is a non-empty set of transitions. $F_B \subseteq (S_B \times T_B) \cup (T_B \times S_B)$ is a set of directed arcs. $W_B$ is the weight function that associates a positive integer to each arc. $u_B$ is the places marking associating positive tokens number to each place. $Z_B \subseteq S_B$ is the set of zero places (also called synchronization places). The places in $S_B \setminus Z_B$ are stable places. A stable marking is a multiset of stable places. The presence of one or more zero places of a given marking makes it unobservable, while stable markings describe observable states of the system.

Let B be a zero safe net and let $s = u_0[t_1>u_1 \ldots u_{n-1}[t_n>u_n$ be a firing sequence of the underlying net $N_B$ of B,

- The sequence *s* is a stable step of B if:

$\forall a \in S_B \setminus Z_B, \Sigma^n_{i=1} pre(t_i)(a) \leq u_0(a)$ (*Concurrent enabling*)

*$u_0$ and $u_n$ are stable markings of B* (*Stable fairness*)

Pre(t)(*a*) defines the weight of the arc from place *a* input of transition *t* to this one. Post(t)(*a*) defines the weight of the arc from transition *t* to its output place *a*. The concurrent enabling property insures the initial simultaneous enabling of all step transitions by stable places and not only those transitions allowing the initial triggering of the first execution. We notice that this property prohibits the consummation of stable tokens produced in the step by its transitions.

- Stable step *s* is a stable transaction of B if in addition:

*Markings $u_1, \ldots, u_{n-1}$ are not stable* (*Atomicity*)

$a \in S_B \setminus Z_B, \Sigma^n_{i=1} pre(t_i)(a) = u_0(a)$ (*Perfect enabling*)

The perfect enabling ensures the consummation of all initial stable tokens before the transaction ends.

In a stable transaction, each transition represents a micro-step carrying out the atomic evolution through invisible states. Stable tokens produced during the transaction become active in the system, only at the end of the transaction.

**Example 3.** Consider the zero-safe net example of **Figure 2(c)**. The firing sequence $\{Fk_1,Fk_2\}(Ch_{1,1}>\{Fk_{1,1}, Fk_2\}$ $(Ch_{1,1}>\{Fk_{2,2},Fk_{1,1}\}$ is not a stable step since the stable fairness is not satisfied. The marking $\{Fk_{2,2},Fk_{1,1}\}$ enables no transition, defining hence a deadlock situation. Since the sequence above is not a stable step and deadlocks at a non-visible state, so it is forbidden.

The two following firing sequences are the unique stable transactions:

$\{Fk_1,Fk_2\}(Ch_{1,1}>\{Fk_{1,1},Fk_2\}(Ch_{2,1}>\{Fk_{2,1},Fk_{1,1}\}(Take_1>\{PhE_1\}$.

$\{Fk_1,Fk_2\}(Ch_{1,2}>\{Fk_{1,2},Fk_2\}(Ch_{2,2}>\{Fk_{1,2},Fk_{2,2}\}(Take_2>\{PhE_2\}$.

In what follows, we exploit features offered by zero-safe nets to define a priority level in ADs actions executions, leading to the reactivity definition.

## 4. Handling Interruptible Activity Region via ZSNs

Formalizing ADs using Petri nets seems to be a good approach. The specification states that "*Activities are redesigned to use a Petri-like semantics*" [1]. Unfortunately, basic Petri nets present some limits.

In [16], we have shown that Petri nets, supposed to be a semantic framework for ADs, are not well suitable to handle new UML semantics such as traverse-to- completion principle. Indeed, the latter requires a global synchronisation and not a local one as defined by Petri nets. We defined a generic mapping from ADs to zero-safe nets that preserves ADs operational semantics while focusing on traverse-to-completion principle and synchronization of fork and join nodes. Therefore, we covered control/data flows and concurrency. Besides, in [17], we have focused on semantics of streaming parameters and exception outputs, and showed also that ZSNs are able to express such complex semantics. Atomic transactions have been defined in ZSNs under a token game based on freezing tokens that have been created in the transaction until it ends. This becomes possible thanks to the zero-places.

The contribution of this paper is to define a suitable mapping of ADs to ZSNs, dealing with more complex concepts of UML 2.0 ADs, namely the Interruptible Activity Region. We show how basic Petri nets are not able

to express semantics of this construct due to their non reactive aspect. We define a new net called $ZSN_{IAR}$ based on ZSNs, formalizing the Interruptible Activity Region as well as other important ADs principles and constructs such as global synchronisation of concurrent regions [16] and streaming parameters and exception outputs [17].

### 4.1 Petri Nets Limits

When dealing with the Interruptible Activity Region, two questions are to be considered: the first is about the raising and handling of exceptions and the second concerns the reactivity to external event.

1) Exceptions are a key example of non-local behavior. Raising and handling an exception means switching, from one of specified program states, to some other ones in one step (a kind of multi-goto).

In Petri nets, while system state is modeled via distributed marking over the whole net places, state changes are local. When mapping Interruptible Activity Region into Petri nets, state is hence distributed over many places of the region. To handle the cancellation semantics via Petri nets, we need to remove a set of place markings (of the interruptible region) at once. Moreover, the number of destructed tokens is only known at run time.

Yet, we can create some net structure warranting that all possible token distributions over places are covered. This is possible by adding arcs that will be connected to all potential combinations of all places in the region. It is obvious that this chaotic solution leads to a huge arcs number (spaghetti arcs). This, will greatly reduce the readability and understanding of such net. Reset arcs seem to be a good solution.

2) The reception of an external event triggers the activity block abortion in Interruptible Activity Region, and continues execution with another action that may be the exception-handler.

All actions of the Interruptible Activity Region are immediately aborted and no action outside the interruptible region can be executed before the handling of that event. This leads to a priority and isolation of execution.

Within the Petri nets semantics, there is no priority in executing two concurrent transitions. The choice of firing one of the enabled concurrent transitions is non-deterministic.

**Example 4.** In **Figure 3**, we give a naïve basic Petri net that formalizes the AD of **Figure 1**. The transcription follows mapping rules defined by Storrle in [3] (See **Table 1**). The author added a number of transitions, modeling the interruption event, equal to the cancelled actions in the region. Each transition is connected to the input place of a cancelled action and to transition *Cancel Order* via an output common place. When the *Cancel Order Request* is made, places of the Interruptible Activity Region, with dark gray, have to be emptied.

**Table 1. Mapping rules from UML activities to basic Petri nets [3]**

| Nodes and edges | UML Activity diagram | Petri Nets |
|---|---|---|
| | | *Fork/Join* |
| Control nodes | | |
| Activity edges | | |
| Executable nodes | *Activity* | |

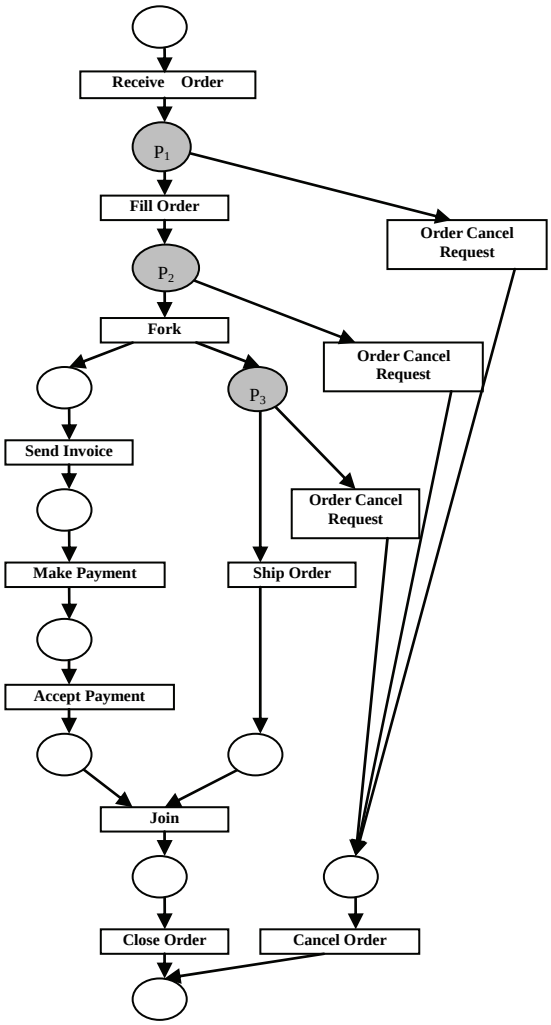



**Figure 3. Intuitive mapping of the AD of figure 1 to basic Petri nets [3]**

We notice that semantics of cancellation (first point) does not appear in the net: First, the cancel event is not visible (sketched by transitions *Order Cancel Request* that are enabled by internal places) and second, the net is supposed to be 1-safe.

Regarding the second point mentioned above: when place $p_1$ is marked, both transitions *Fill Order* and *Order Cancel Request* are enabled and have the same probability to fire (situation of effective conflict). Whereas in this context, we would like to fire the aborting transition, that is *Order Cancel Request*.

The ZSNs model offers transitions coordination thanks to zero places. It guarantees atomicity and isolation of transaction, and this is all what we need in the cancellation schema. In what follows, we use ZSNs semantics to conduct the control flow in the net.

## 4.2 Mapping Intuition

In what follows, we discuss two zero-safe nets based approaches to formalize AD interruptible region with regard to semantics via the running example 1.

The first solution introduces reset arcs and no new mechanism is necessary beyond the zero-safe nets semantics. In the net of **Figure 4**, we introduce a transition called '*cancel*', and then we connect all places in the Interruptible Activity Region to that transition by a reset arc for each. The firing of transition '*cancel*' empties all its input places at once, regardless of their marking. Thus, the net is no longer forced to be 1-safe. To overcome the second shortcoming pointed out, we add an input place '*interface place*' to transition '*cancel*'. This place represents the external cancel event. It is connected to transition *'cancel'* via an arc of weight 1. When the place '*interface place*' is marked, the transition '*cancel*' is enabled. Possibly, other transitions of the region are enabled at the same time. We need to guide the control to fire transition *'cancel'* first. This is known as isolation and atomicity. To achieve this, we assume that '*interface place*' is a zero place and not a stable one, so when marked, transition '*cancel*' is enabled and immediately fired. This is due to the *enabling property* of ZSNs. Then another problem arises: when combining both solutions *i.e.* reset arcs and the *interface* zero place, enabling of transition *cancel* is made via the zero place connected with a non-reset arc. Thus, if another input place that has to be emptied by *cancel*, has an other output transition, it could be possible to fire that transition first and then '*cancel*' transition indeterminably without impeding ZSNs rules. This is essentially caused by the presence of reset arcs. To overcome this problem, we can easily create a stable token in the transaction that is frozen until the transaction ends. The corresponding place is also an input one to cancelled transitions via reset arcs. (see **Figure4**).

In **Figure 4**, firing the external transition creates one stable token in the stable place $p_{freeze}$ and one zero token in

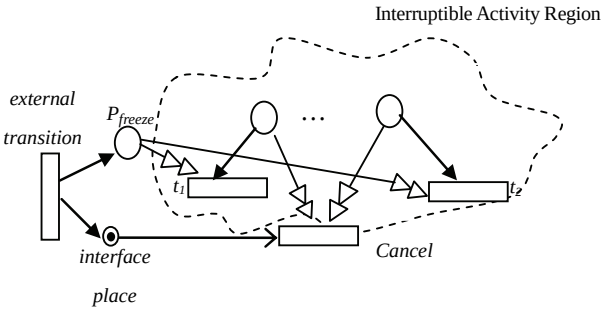



**Figure 4. Formal semantics of the Interruptible Activity Region via ZSNs augmented with reset arcs**

*interface place*. The stable token cannot be consumed until the transaction ends, hence prohibiting the firing of the region enabled transitions such as $t_1$ and $t_2$. The unique transition that satisfies firing conditions is *cancel*. The created token in $p_{freeze}$ can be consumed in the first next firing not being a cancellation procedure.

It is clear that such construction greatly improves modeling cancellation patterns and preserves semantics. However, adopting such technique has its drawbacks; the number of used reset arcs in this model depends always on the number of places in the interruptible region. This reduces considerably the net readability.

In **Figure 5**, we define a special cancellation transition *cancel* (pictured by an underlined rectangle) with its new enabling and firing semantics. *Cancel* may have many stable inputs and one zero input place, that is *interface place*. There are two different conditions to enable transition *cancel*:

1) Necessary condition but not sufficient to fire *cancel*: the input zero place is marked.

2) Effective firing condition: the instantaneous marking of *cancel* input places, *i.e.*, input places markings when the zero token is created. This marking is calculated at run time, and this one is the enabling marking. Thus, once firing *cancel* transition, all of its input places are emptied.

When the zero place is marked (via an external transition), *cancel* is enabled, the current marking is then calculated

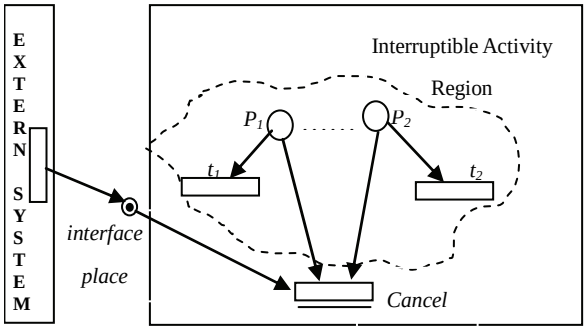



**Figure 5. Formal semantics of the Interruptible Activity Region via ZSNs and a special transition *cancel***

and is equal to the destroyed tokens. This latter is necessary to the transition firing. Hence, it is forbidden to fire $t_1$ or $t_2$ first. In such case, the *cancel* firing condition would not be satisfied leading to a deadlock situation in an invisible state (non-observable marking). Firing *cancel* will switch, from one of specified program states ($\{p_1\}$, $\{p_2\}$ or $\{p_1, p_2\}$), to some other ones in one step.

In our proposed semantics, event triggering cancellation is not formalized via a transition (this should be the intuition), but via a zero place. Hence, coordinating the execution of the termination action is made possible.

With basic Petri nets, this is not possible since it is agreed that an enabled transition can be fired or not, *i.e.* firing one of two concurrently enabled transitions is nondeterminitic. With ZSNs, interface place is modeled with a zero place rather than a stable one. Whenever, an out-transition (a transition not belonging to the system) is fired, a zero token is created in the *interface place* indicating that the system is actually executing a transaction. Transactions have a higher execution priority compared to transitions. Hence, firing *cancel* transition is prior to any other transition.

**Figure 6** presents the mapping of the Interruptible Activity Region part of **Figure 1**. When *Ship Order* is enabled, a cancellation event occurs. This is traduced by marking the zero place *interface place*. The effective firing condition of *cancel* is calculated and it is equal to {*interface place, p3*}. Two transitions are now enabled: *Ship Order* and *Cancel*. Firing transition *Cancel* is prior than transition *Ship Order*. Firing *Ship Order* first, leads to a deadlock

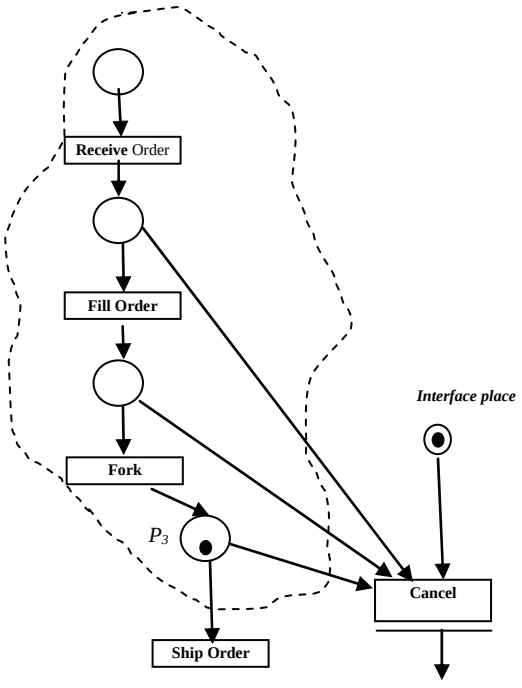



**Figure 6. Intuitive mapping of the Interruptible Activity Region of the AD of Figure 1 to ZSNs**

situation (non finishing transaction) caused by the consumption of *cancel* transition enabling tokens.

### 4.3 Formal Mapping

**Table 2** defines preliminarily hints on formalizing UML 2.0 ADs via ZSNs. This generic mapping covers basic constructs, concurrent-region, traverse-to-completion principle, streaming parameters, exception outputs and the Interruptible Activity Region.

Executable and fork/join nodes are mapped to transitions. Control nodes become stable or zero places, depending of the synchronization schema to be modeled. Specific Petri nets models are given in particular cases such as streaming parameters, exception outputs and the Interruptible Activity Region. Most of these notations have already been examined in earlier work. The semantics of the Interruptible Activity Region is discussed in this paper.

To formalize the mapping, we propose, for both basic activity diagram AD of UML 1.x and a complete one of UML 2.0, rigorous notations as given below. Extended activity diagram AD2 encloses new constructs and semantics, namely object nodes, traverse-to-completion principle, streaming parameters, exception outputs and Inturruptible Activity Region. Next, we define a formal semantic definition of AD2 in terms of ZSNs.

**Definition 1:**
An activity diagram is defined by a tuple *AD* = (*EN, BN, CN, iN, fN, CF*) where:

*EN*: denote Executable Nodes, *i.e.*, elementary actions. $EN = \{A_1, A_2, ..., A_n\}$.

*BN*: denote Branch Nodes *i.e.* decisions and merges. $BN = \{d_1, ..., d_k; m_1, ..., m_{k'}\}$, such as : $k \geq k'$.

*CN*: denote Concurrency Nodes *i.e.* forks and joins. $CN = \{f_1, ..., f_m; j_1, ..., j_{m'}\}$, such as: $m \geq m'$.

*iN*: denotes the initial Node.

*fN*: denotes the final Node.

*CF*: is a function denoting Control Flows. $CF \subseteq ((EN, BN, CN, iN) \times (EN, BN, CN, fN))$. A directed arc sketches the control flow where the source may be an action, a branch, a control or the initial node and the arc target may be an action, a branch, a control or the final node.

**Definition 2:**
An UML2.0 AD is defined by a tuple *AD*2 = (*AD, ON, OF, CR, SA, EA, IAR*) where:

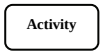
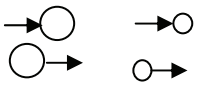
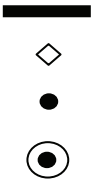
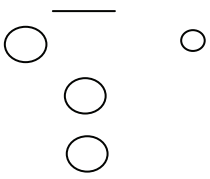
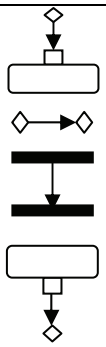
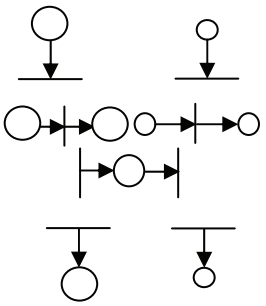
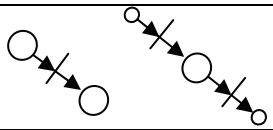
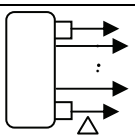
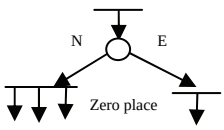
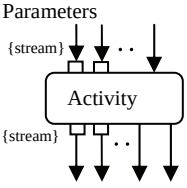
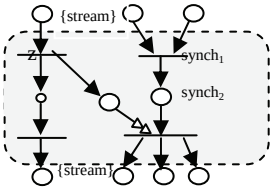
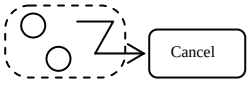
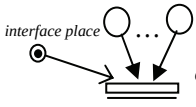
*AD*: is the corresponding basic activity diagram as defined above.

*ON*: denotes Object Nodes. In this work, we deal with pins. $ON = \{o_1, ..., o_r\}$. Objects may represent data or streams $\{s_1, \ldots, s_{r'}\}$ or exceptions $\{e_1, \ldots, e_{w'}\}$.

*OF*: is a function denoting Object (token) Flows. $OF \subseteq ((BN, CN, iN, ON) \times (BN, CN, fN, ON))$. $OF = \{of_1, \ldots of_x\}$. As tokens move across an object flow edge, they may undergo transformations. An object flow might carry a transformation behavior denoted *tb*.

**Table 2. The intuition of ADs formal semantics via ZSNs: Zero places are pictured with small circles**

| Activities Nodes | ZSNs Nodes: Basic Activity Nodes | ZSNs Nodes: Concurrent Region |
|---|---|---|
| Executable Nodes   |  | |
| Object Nodes  |  | |
| Control Nodes  |  | |
| Object Flows  |  | |
| Control Flows  |  | |
| Unless  |  | |
| Except  |  | |
| Exception Outputs  |   | |
| Streaming Parameters   |   | |
| Events  | zero place Interface  | |
| InterruptibleActivityRegion   |   | |

*CR*: denotes Concurrent Regions. A concurrent region is a *sub-AD2* delimited with a fork and a join. Concurrent regions have a special semantics under the traverse-to-completion principle. This one has been discussed in [16] along with a generic mapping to ZSNs.

*SA*: is a set of Streaming Actions $\{S_1, \ldots, S_r\}$. A streaming action is an elementary action $A_i$ with input/output streaming parameters $s_i$.

*EA*: is a set of Exception Actions $\{E_1, \ldots, E_w\}$. An exception action is an elementary action $A_i$ having exception outputs $e_i$.

*IAR*: denotes Interruptible Activity Regions. An interruptible region is a *sub-AD*2 bound to a special specification *Spec* that can, informally, be *Spec* (*IAR*) = (*evt, cancel*) where *evt* is the interruption triggering event and *cancel* is the cancellation action. We note that *evt* and *cancel* do not belong to the *IAR*. In the perspective of cancellation, only actions *EN* and control nodes *CN* may be interrupted, thus the enclosed actions and control nodes define an *IAR*. $IAR_i = \{A_g, \ldots, A_{g'}, f_h, \ldots, f_{h'}; j_v, \ldots, j_{v'}\}$, such that, A, f and j stand respectively for actions, forks and joins enclosed in the region.

For the sake of the presentation, we restrict our ZSN definition purpose to control flow, data flow and Interruptible Activity Region. Let $AD_{IAR}$ be a *sub-AD2*, such that, $AD_{IAR}$ = (*EN*, *BN*, *CN*, *iN*, *fN*, *CF*, *ON*, *OF*)) and let *Spec* be a specification such that:

*Spec* (*IAR)* = (*evt, cancel*). We define $AD_{IAR}$ by identifying, in addition to IAR nodes, the input and output object and branch nodes connected to each IAR node via a control or object edge, including edges. Recall that an $IAR_i = \{A_g, \ldots, A_{g'}, f_h, \ldots, f_{h'}; j_v, \ldots, j_{v'}\}$.

Next, we define a formal mapping from a sub *AD*2 $AD_{IAR}$ to a zero-safe net $ZSN_{IAR}$.

**Example 5.** Consider the AD of **Figure 1**: let $IAR_1$ be $IAR_1$ = {ReceiveOrder, FillOrder, ShipOrder, $f_1$}, where $f_1$ stands for the fork node and {ReceiveOrder, FillOrder, ShipOrder} $\subset$ EN.

We define $AD_{IAR1}$ by identifying inputs and outputs of $IAR_1$ nodes. $AD_{IAR1}$ = {ReceiveOrder, FillOrder, ShipOrder, $f_1$, $d_1$, ⟨ReceiveOrder, $d_1$⟩, ⟨$d_1$, FillOrder⟩, ⟨FillOrder, $f_1$⟩, ⟨$f_1$, ShipOrder⟩} where $d_1$ stands for the decision node and pairs of the form ⟨x, y⟩ stand for edges such as x is the edge source and y is its target. Spec ($IAR_1$) = (OrderCancelRequest, CancelOrder) where OrderCancelRequest stands for *evt* and CancelOrder for *Cancel*.

**Definition 3:**

$ZSN_{IAR}$ is a special ZSN defining the semantic of $AD_{IAR}$, an UML 2.0 sub activity diagram with the *Spec* specification.

$ZSN_{IAR}$ = (*ZSN*, $S_{IAR}$, $Z_{cancel}$, *Cancel*, *ip*, *sp*) where:

- There is a single source place *ip*, such that, $ip \in S_B$, $^\bullet ip = \emptyset$.
- There is a single sink place *sp*, where $sp \in S_B$,

$sp^{\bullet} = \varnothing$.

- Every node *n* in the instance net is on the path from *ip* to *sp*.

1) $Z_{cancel} \subseteq Z_B$*: is a set of zero places* $\{z_{canc1}, \ldots, z_{cancx}\}$, such that, $\forall z_{canci} \in Z_{cancel}, z_{canc}^{\bullet} = \{canc_i\}$

2) $S_{IAR} \subseteq P$. $S_{IAR}$ is a set of places. $S_{IAR} = \{p \mid (p \in S_B) \wedge (p = n \mid n \in \{AD_{IAR} \cap \{BN \cup ON - \{p_{oi} \mid \exists of \in OF$ and $of = o_i \times o_{i'}$ and $\neg\exists tb$ on $of\} \cup \{iN, fN\} \cup \{p_c \mid c \in CF\}\}\}$

3) *Cancel* $\subset T$*:* a set of special transitions $\{canc_1, \ldots, canc_x\}$, such that, the enabling condition to each transition $canc_i$ is the marking of $z_{canci}$ and the effective firing condition is the instantaneous marking calculated, when $z_{canci}$ is marked. Given a transition $canc_i$, the firing sequence is given by:

$\{z_{canci}, M_{SIAR}\}(canc_i > M'/ z_{canci}, \notin M' \wedge M'_{SIAR} = \varnothing$ and $^{\bullet}canc_i = \{S_{IAR}, z_{canci}\}$ where $M_{SIAR}$ and $M'_{SIAR}$ respectively stand for $S_{IAR}$ markings before and after firing $canc_i$.

4) *ZSN*: denotes a zero-safe net, *i.e.*, $ZSN = (S_B, T_B; F_B, W_B, u_B; Z_B)$ as defined in Section 3.2 such that:

$S_B = BN \cup ON - \{p_{oi} \mid \exists\ of = (p_{oi}, p_{oi+1})$ *and* $\neg\exists\ tb$ *on* $of\} \cup \{iN, fN\} \cup \{p_c \mid c \in CF\}$

For each branch or object node, we create a place. When two object nodes are connected via an edge not carrying a transformation behavior, just one place is created and takes the name of one of the two (since they have the same name).

$T_B = EN \cup CN \cup \{t_{oi} \mid \exists of \in OF$ and $of = o_i \times o_{i'}$ and $\exists tb$ on $of\} \cup \{t_{d\ id\ i'} \mid \exists of \in OF$ and $of = d_i \times d_{i'}$ or $\exists cf \in CF$ and $cf = d_i \times d_{i'}\} \cup \{t_{m\ im\ i'} \mid \exists of \in OF$ and $of = m_i \times m_{i'}$ or $\exists cf \in CF$ and $cf = m_i \times m_{i'}\} \cup Cancel$.

Executable and control nodes are mapped into transitions. An object flow gives rise to a new transition iff this edge carries a transformation behavior. For each control flow, we define a transition.

$F_B = \{\langle x, y\rangle \mid \langle x, y\rangle \in CF \wedge (x \in T_B) \wedge (y \in S_B)\} \cup \{\langle x, y\rangle \mid \langle x, y\rangle \in CF \wedge (x \in S_B) \wedge (y \in T_B)\} \cup \{\langle x, y\rangle \mid (x \in T_B) \wedge (y \in S_B) \wedge \exists A_i \mid x = A_i \wedge y = o_{i'}\}$.

$W_B: F_B \rightarrow$ lN.

$ip = iN$.

$sp = fN$.

$u_B = \{iN\}$

$Z_B = \{evt\}, z_{canc} = evt$.

The above definition, mapping an $AD_{IAR}$ to a ZSN, is faithful to the intuitive mapping given in **Table 2**. Concurrent regions, streams, and exceptions are not yet taken into account. The semantics of cancellation is deeply considered. So far, none of the previous works authors has considered the problem of reactivity in ADs cancellation behavior.

## 5. Conclusions

This paper is a continuation of our last two papers [16], [17]. Their main goal was to propose a generic mapping of ADs basic concepts to ZSNs ones. Especially, they handle formalization of concurrent-region, while considering the traverse-to-completion semantics and exception outputs streaming parameters via ZSNs.

This paper highlights also the failure of Petri nets to cover high semantics of ADs, namely the Interruptible Activity Region. Here, we have proposed, with the same spirit, the use of ZSNs as a formal semantic framework to handle this region.

A generic mapping from ADs to ZSNs, covering basic constructs, concurrent-region, traverse-to-completion principle, streaming parameters, exception outputs and the Interruptible Activity Region has been defined. Its formal definition based on ZSNs covers until now control flow, data flow and Interruptible Activity Region. Concurrent region, streaming parameters and exceptions are not yet covered, but they can be integrated very simply in the defined $ZSN_{IAR}$.

Some other constructs namely expansion-region and exception handling are to be considered in future works. Our aim is to define an EZS-Net (Extended Zero-Safe Net) for all new constructs defined in AD2. The EZS-Net will be dedicated to formalize UML ADs in a complete and unified way.

ZSNs are tile logic based models which is an extension of rewriting logic, taking into account the concept of side effects and dynamic constraints on terms. Mapping UML2 ADs to ZSNs can be followed by the projection of these latter in rewriting logic and thus, exploiting its practical system Maude for verification and validation aims.

## REFERENCES

[1] "OMG Unified Modelling Language: Superstructure," *Final Adopted Specification Version* 2.0,*Technical Report*, Object Management Group, November 2003. http://www .omg.org

[2] T. Schttkowsky and A. Föster, "On the Pitfalls of UML 2 Activity Modeling," *International Workshop on Modeling in Software Engineering*, Minneapolis, IEEE Computer Society, 2007.

[3] H. Störrle and J. H. Hausmann, "Towards a Formal Semantics of UML 2.0 Activities," *Software Engineering* Vol. 64, 2005, pp. 117-128.

[4] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989, pp. 541-580.

[5] E. Borger and R. Stark, "Abstract State Machines," Springer Verlag, 2003.

[6] R. Bruni and U. Montanari, "Zero-Safe Nets, or Transition Synchronization Made Simple," In C. Palamidessi and J. Parrow, Eds., *Proceedings of the 4th workshop on Expressiveness in Concurrency*, *Electronic Notes in Theoretical Computer Science*, Santa Margherita

Ligure, Elsevier Science, Vol. 7, 1997.

[7] J. P. Barros and L. Gomes, "Actions as Activities and Activities as Petri Nets," In Jan J¨urjens, Bernhard Rumpe, Robert France, and Eduardo B. Fernandey, Eds., *UML* 2003 *Workshop on Critical Systems Development with UML*, San Francisco, 2003, pp. 129-135.

[8] T. S. Staines, "Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling," "Concept Petri Net Diagrams and Colored Petri Nets," 15*th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, Belfast, 2008.

[9] H. Störrle, "Semantics of Exceptions in UML 2.0 Activities," *Journal of Software and Systems Modeling*, 9 May 2004. www.pst.informatik.uni-muenchen.de/stoerrle

[10] H. Störrle, "Semantics of Control-Flow in UML 2.0 Activities," In N.N. Ed., *Proceedings IEEE Symposium on Visual Languages and Human-Centric Computing*, Rome, Springer Verlag, 2004.

[11] H. Störrle, "Semantics and Verification of Data Flow in UML 2.0 Activities," *Electronic Notes in Theoretical Computer Science*, Vol. 127, No. 4, 2005, pp. 35-52. www. pst.informatik.uni-muenchen.de/-stoerrle

[12] H. Störrle, "Semantics and Verification of Data-Flow in UML 2.0 Activities," *Proceedings International Workshop on Visual Languages and Formal Methods*, IEEE Press, 2004, pp. 38-52. www.pst.informatik.uni-muench en.de/_stoerrle

[13] R. Eshuis and R. Wieringa. "Comparing Petri Net and Activity Diagram Variants for Workflow Modelling–A Quest for Reactive Petri Nets," In Weber *et al. Petri Net Technology for Communication Based Systems, Lecture Notes in Computer Science*, Vol. 2472, 2002, pp. 321-351.

[14] R. Eshuis and R. Wieringa. "A Real-Time Execution Semantics for UML Activity Diagrams," In H. Hussmann, Ed., *Fundamental Approaches to Software Engineering*, *Lecture Notes in Computer Science*, Genova, Springer Verlag, Vol. 2029, 2001, pp. 76-90.

[15] R. Eshuis and R. Wieringa. "An Execution Algorithm for UML Activity Graphs," *Proceedings of the* 4*th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, *Lecture Notes in Computer Science*, Toronto, Springer Verlag, Vol. 2185, 2001, pp. 47-61.

[16] S. Boufenara, F. Belala and C. Bouanaka, "Les Zero-Safe Nets Pour la Préservation de la TTC Dans les Diagrammes d'activité d'UML, " "Revue des Nouvelles Technologies de l'Information RNTI-L-3," Cépaduès éditions, 15*ème Conférence Internationnale sur les Langages et Modèles à Objets* : *LMO*, 2009, pp. 91-106.

[17] S. Boufenara, F. Belala and N. Debnath, "On Formalizing UML 2.0 Activities: Stream and Exception Parameters," 22*nd International Conference on Computers and Their Applications in Industry and Engineering CAINE*-2009, San Francisco, 4-6 November 2009.

[18] C. Bock, "UML 2 Activity and Action Models," *Part* 6: *Structured Activities*, 2005. http://www.jot.fm/issues/issue _2005_05/column4

[19] R. Bruni and U. Montanari, "Transactions and Zero-Safe Nets," In: H. Ehrig, G. Juhás, J. Padberg and G. Rozenberg, Eds., *Proceedings of Advances in Petri Nets*: *Unifying Petri Nets*, *Lecture Notes in Com- puter Science*, Springer Verlag, Vol. 2128, 2001, pp. 380-426.

*J. Software Engineering & Applications*, 2010, 3: 436-445
doi:10.4236/jsea.2010.35049 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

# Implementation of Hierarchical and Distributed Control for Discrete Event Robotic Manufacturing Systems

**Gen'ichi Yasuda**

Nagasaki Institute of Applied Science, Nagasaki, Japan.
Email: YASUDA_Genichi@NiAS.ac.jp

Received January 6th, 2010; revised January 22nd, 2010; accepted January 25th, 2010.

## ABSTRACT

*The large scale and complex manufacturing systems have a hierarchical structure where a system is composed several lines with some stations and each station also have several machines and so on. In such a hierarchical structure, the controllers are geographically distributed according to their physical structure. So it is desirable to realize the hierarchical and distributed control. In this paper, a methodology is presented using Petri nets for hierarchical and distributed control. The Petri net representation of discrete event manufacturing processes is decomposed and distributed into the machine controllers, which are coordinated through communication between the coordinator and machine controllers so that the decomposed transitions fire at the same time. Implementation of a hierarchical and distributed control system is described for an example robotic manufacturing system. The demonstrations show that the proposed system can be used as an effective tool for consistent modeling and control of large and complex manufacturing systems.*

***Keywords*:** *Implementation, Robotic Manufacturing Systems, Hierarchical and Distributed Control, Discrete Event Systems, Petri Nets*

## 1. Introduction

Because of robot's flexibility, industrial robots have been introduced into industry to automate various operations without significant redesign. This flexibility is derived from the generality of the robot's physical structure, control and reprogrammability, but it can only be exploited if the robot can be programmed easily. In some cases, the lack of adequate programming tools makes some tasks impossible to be performed. In other cases, the cost of programming may be a significant fraction of the total cost of an application. Further, it is quite obvious that a single robot cannot perform effective tasks in an industrial environment, unless it is provided with some additional equipment. It is usually required to integrate the robot into the manufacturing system, which includes NC machine tools, belt conveyors, and other special purpose machines. Further, the robot often must interact with such machines, other robots or operators.

These external processes are executing in parallel and asynchronously. It is not possible to predict exactly when events of interest to the robot program may occur. The signal lines are supported by most robot systems to coordinate multiple robots and machines, but this is a very limited form of communication between processes. Sophisticated tasks require efficient means for coordination and for sharing the state of the system between processes. The programming system should provide a mechanism for specifying the behavior of systems more complex than a single robot. Existing robot programming systems are based on the view of a robot system as a single robot weakly linked to other machines. Many machines may be cooperating during a task. The interactions between them may be highly dynamic. No existing robot programming system adequately deals with all of these interactions. No existing computer language is adequate to deal with this kind of parallelism and real-time constraints.

The overall structure of the working area in a large and complex manufacturing system consists of one or more lines, each line consists of one or more stations, and each station (shop or cell) consists of one or more machines such as robots and intelligent machine tools. Inside of a cell, machines execute cooperation tasks such as machining, assembling and storing. Inside of a shop, cells cooperate mutually and execute more complicated tasks. Furthermore each machine consists of several motion elements.

A task executed by a robot or an intelligent machine tool can be seen as some connection of more detailed subtasks. For example, transferring an object from a start position to a goal position is a sequence of the following subtasks; moving the hand to the start position, grasping the object, moving to the goal position, and putting it on the specified place. Thus the manufacturing system handles complicated tasks by dividing a task hierarchically in this structure, which is expected to be effective in managing cooperation tasks executed by great many machines or robots.

One of the effective methods to describe and control such systems is the Petri net which is a modeling tool for asynchronous and concurrent discrete event systems [1]. Conventional Petri net based control systems were implemented based on an overall system model. The description capability of the Petri net is very high; nevertheless, in case of manufacturing systems, the network model becomes complicated and it lacks for the readability and comprehensibility. Since in the large and complex systems, the controllers are geographically distributed according to their physical (hardware) structure, it is desirable to realize the hierarchical and distributed control.

The hierarchical and distributed control for large and complex discrete event manufacturing systems has not been implemented so far [2-4]. A Petri net model includes control algorithms, and is used to control the manufacturing process by coincidence of the behavior of the real system with the Petri net model. Thus, if it can be realized by Petri nets, the modeling, simulation and control of large and complex discrete event manufacturing systems can be consistently realized by Petri nets [5-6]. In this paper, the author presents a methodology by extended Petri nets for hierarchical and distributed control of large and complex robotic manufacturing systems, to construct the control system where the cooperation of each controller is implemented so that the aggregated behavior of the distributed system is the same as that of the original system and the task specification is completely satisfied.

## 2. Discrete Event Modeling of Robotic Manufacturing Systems using Petri Nets

A manufacturing process is characterized by the flow of workpieces or parts, which pass in ordered form through subsystems and receive appropriate operations. Each subsystem executes manufacturing operations, that is, physical transformations such as machining, assembling, or transfer operations such as loading and unloading. From the viewpoint of discrete event process control, an overall manufacturing process can be decomposed into a set of distinct activities (or events) and conditions mutually interrelated in a complex form. An activity is a single operation of a manufacturing process executed by a subsystem. A condition is a state in the process such as machine operation mode.

For example, a simple robot operation example, where the robot waits until a workpiece appears and then handles the workpiece and sends it out for the next operation, can be modeled as follows:

| | |
|---|---|
| condition | the robot is waiting |
| event | a workpiece arrives |
| condition | the workpiece has arrived and is waiting |
| event | the robot starts the handling |
| condition | the robot is handling the workpiece |
| event | the robot finishes the handling |
| condition | the handling has been completed |
| event | the workpiece is sent for other operation |

The above example illustrates a system where events and conditions are mutually connected. These systems are known as event-condition systems. Events simultaneously represent the end of the preceding condition and the beginning of the succeeding condition. Event-condition systems exhibit the following features:

1) Asynchronism

The system is essentially asynchronous. Events always occur when their conditions are satisfied.

2) Ordering

Before and after one condition there are always events, and each event is defined by preconditions and post conditions.

3) Parallelism

In one system two or more conditions can be held simultaneously and for this, events that do not interact may occur independently.

4) Conflict

One condition can be a precondition of various events and depending on which event is occurring, different conditions hold.

Because of these features, the following phenomena can occur in the event-condition system:

1) Deadlock occurs when the system enters into a state that is not possible for any event to occur.

2) Bumping occurs when despite the holding of a condition, the preceding event occurs. This can result in the multiple holding of that condition. When the system is free of this phenomenon, the system is called safe.

To represent discrete event manufacturing systems a modeling technique was derived from Petri nets [7-8]. Considering not only the modeling of the systems but also the actual manufacturing system control, the guarantee of safeness and the additional capability of input/output signals from/to the machines are required. The extended Petri net consists of the following six elements: 1) Place; 2) Transition; 3) Directed arc; 4) Token; 5) Gate arc; 6) Output signal arc.

A place represents a condition of a system element or action. A transition represents an event of the system. A directed arc connects a place to a transition, and its direction shows the input and output relation between them. Places and transitions are alternately connected using

directed arcs. The number of directed arcs connected with places or transitions is not restricted. A token is placed in a place to indicate that the condition corresponding to the place is holding.

A gate arc connects a transition with a signal source, and depending on the signal, it either permits or inhibits the occurrence of the event which corresponds to the connected transition. Gate arcs are classified as permissive or inhibitive, and internal or external. An output signal arc sends the signal from a place to an external machine. A transition is enabled if and only if it satisfies all the following conditions:

1) It does not have any output place filled with a token.

2) It does not have any empty input place.

3) It does not have any internal permissive arc signaling 0.

4) It does not have any internal inhibitive arc signaling 1.

An enabled transition may fire when it does not have any external permissive arc signaling 0 nor any external inhibitive arc signaling 1. The firing of a transition removes tokens from all its input places and put a token in each output place connected to it. The assignment of tokens into the places of a Petri net is called marking and it represents the system state. In any initial marking, there must not exist more than one token in a place. According to these rules, the number of tokens in a place never exceeds one, thus, the Petri net is essentially a safe graph.

If a place has two or more input transitions or output transitions, these transitions may be in conflict for firing. When two or more transitions are enabled only one transition should fire using some arbitration rule. By the representation of the activity contents and control strategies in detail, features of discrete event manufacturing systems such as ordering, parallelism, asynchronism, concurrency and conflict can be concretely described through the extended Petri net.

## 3. Design of Hierarchical and Distributed Control

The overall procedure for the design and implementation of hierarchical and distributed control is summarized as shown in **Figure 1**. The basic procedures of modeling and decomposition of robotic manufacturing systems are shown. A global, conceptual Petri net model is first chosen which describes the aggregate manufacturing process. At the conceptual level each task specification is represented as a place of the Petri net as shown, where the activity of each equipment is also represented as a place.

Based on the hierarchical approach, the Petri net is translated into detailed subnets by stepwise refinements from the highest system control level to the lowest machine control level. At each step of detailed specification, some parts of the Petri net, transitions or places, are substituted by a subnet in a manner, which maintains the structural properties.

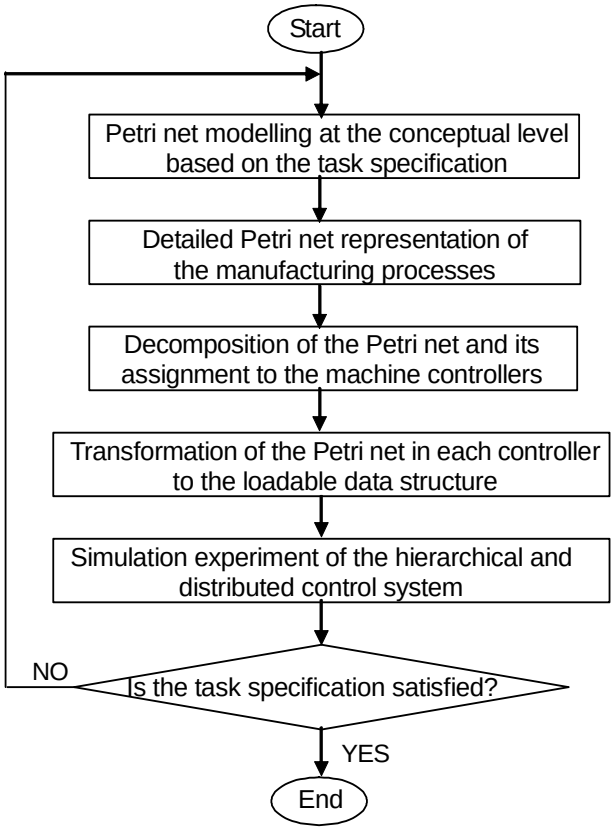



**Figure 1. Flow chart of Petri net based implementation of hierarchical and distributed control system**

It is natural to implement a hierarchical and distributed control system, where one controller is allocated to each control layer or block. For the manufacturing system, an example structure of hierarchical and distributed control is composed of one station controller and three machine controllers as shown in **Figure 2**, although each robot may be controlled by one robot controller. The detailed Petri net is decomposed into subnets, which are executed by each machine controller.

In the decomposition procedure, a transition may be divided and distributed into different machine controllers as shown in **Figure 3**. The machine controllers should be coordinated so that these transitions fire in union. Decomposed transitions are called global transitions, and other transitions are called local transitions.

Decomposed transitions must function in union, that is, the aggregate behavior of decomposed subnets should be the same as that of the original Petri net. By the Petri net model, the state of the discrete event system is represented as the marking of tokens, and firing of any transition brings about change to the next state. So the firing condition and state (marking) change before decomposition should be the same as those after decomposition. The firability condition and external gate condition of a transition *j* before decomposition are described as follows:

$$t_j(k) = \bigcap_{m=1}^{M} p_{j,m}^{I}(k) \wedge \bigcap_{n=1}^{N} \overline{p_{j,n}^{O}(k)} \wedge \bigcap_{q=1}^{Q} g_{j,q}^{IP}(k) \wedge \bigcap_{r=1}^{R} \overline{g_{j,r}^{II}(k)} \quad (1)$$

$$g_j^E(k) = \bigcap_{u=1}^{U} g_{j,u}^{EP}(k) \wedge \bigcap_{v=1}^{V} \overline{g_{j,v}^{EI}(k)} \quad (2)$$

where,

$M$ : input place set of transition $j$ ;

$p_{j,m}^{I}(k)$ : state of input place $m$ of transition $j$ at time sequence $k$ ;

$N$ : output place set of transition $j$ ;

$p_{j,n}^{O}(k)$ : state of output place $n$ of transition $j$ at time sequence $k$ ;

$Q$ : internal permissive gate signal set of transition $j$ ;

$g_{j,q}^{IP}(k)$ : internal permissive gate signal variable $q$ of transition $j$ at time sequence $k$ ;

$R$ : internal inhibitive gate signal set of transition $j$ ;

$g_{j,r}^{II}(k)$ : internal inhibitive gate signal variable $r$ of transition $j$ at time sequence $k$ ;

$U$ : external permissive gate signal set of transition $j$ ;

$g_{j,u}^{EP}(k)$ : external permissive gate signal variable $u$ of transition $j$ at time sequence $k$ ;

$V$ : external inhibitive gate signal set of transition $j$ ;

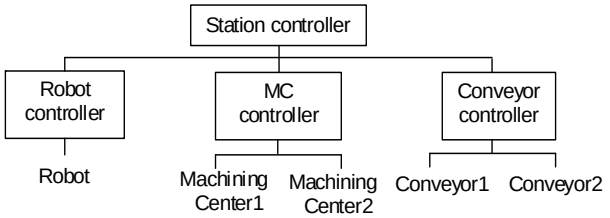



**Figure 2. Example structure of distributed control system**

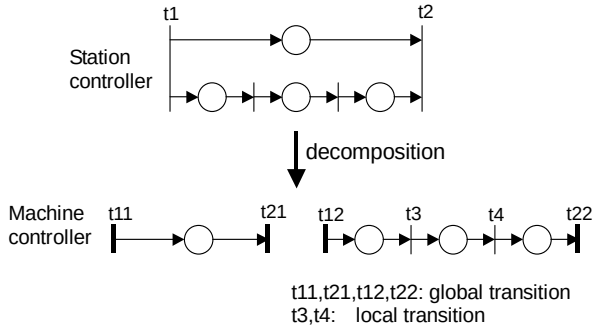



**Figure 3. Decomposition of transition**

$g_{j,v}^{EI}(k)$ : external inhibitive gate signal variable $v$ of transition $j$ at time sequence $k$ ;

The state (marking) change, that is, the addition or removal of a token of a place, is described as follows:

$$p_{j,m}^{I}(k+1) = p_{j,m}^{I}(k) \wedge \overline{(t_j(k) \wedge g_j^E(k))} \quad (3)$$

$$p_{j,n}^{O}(k+1) = p_{j,n}^{O}(k) \vee (t_j(k) \wedge g_j^E(k)) \quad (4)$$

If transition $j$ is divided into $s$ transitions $j1$, $j2$, , , $js$, the firability condition of a transition after decomposition is described as follows:

$$t_{jsub}(k) = \bigcap_{m=1}^{Msub} p_{jsub,m}^{I}(k) \wedge \bigcap_{n=1}^{Nsub} \overline{p_{jsub,n}^{O}(k)} \wedge \bigcap_{q=1}^{Qsub} g_{jsub,q}^{IP}(k) \wedge \bigcap_{r=1}^{Rsub} \overline{g_{jsub,r}^{II}(k)} \quad (5)$$

$$g_{jsub}^{E}(k) = \bigcap_{u=1}^{Usub} g_{jsub,u}^{EP}(k) \wedge \bigcap_{v=1}^{Vsub} \overline{g_{jsub,v}^{EI}(k)} \quad (6)$$

From Equation (1) and Equation (5),

$$t_j(k) = \bigcap_{sub=1}^{S} t_{jsub}(k) \quad (7)$$

From Equation (2) and Equation (6),

$$g_j^E(k) = \bigcap_{sub=1}^{S} g_{jsub}^{E}(k) \quad (8)$$

where,

$S$ : total number of subnets

$Msub$ : input place set of transition $jsub$ of subnet $sub$ ;

$p_{jsub,m}^{I}(k)$ : state of input place $m$ of transition $jsub$ of subnet $sub$ at time sequence $k$ ;

$Nsub$ : output place set of transition $jsub$ of subnet $sub$ ;

$p_{jsub,n}^{O}(k)$ : state of output place $n$ of transition $jsub$ of subnet $sub$ at time sequence $k$ ;

$Qsub$ : internal permissive gate signal set of transition $jsub$ of subnet $sub$ ;

$Rsub$ : internal inhibitive gate signal set of transition $jsub$ of subnet $sub$ ;

$Usub$ : external permissive gate signal set of transition $jsub$ of subnet $sub$ ;

$Vsub$ : external permissive gate signal set of transition $jsub$ of subnet $sub$ ;

The addition or removal of a token of a place connected to a decomposed transition is described as follows:

$$p^I_{jsub,m}(k+1) = p^I_{jsub,m}(k) \wedge \overline{(t_j(k) \wedge g^E_j(k))} \quad (9)$$

$$p^O_{jsub,n}(k+1) = p^O_{jsub,n}(k) \vee (t_j(k) \wedge g^E_j(k)) \quad (10)$$

Consequently it is proved that the firability condition of the original transition is equal to AND operation of firability conditions of decomposed transitions. If and only if all of the decomposed transitions are enabled, then the global transitions are enabled. To exploit the above results, the coordinator program has been introduced to coordinate the decomposed subnets so that the aggregate behavior of decomposed subnets is the same as that of the original Petri net.

There may exist a place which has plural input transitions and/or plural output transitions. This place is called a conflict place. The transitions connected to a conflict place are in conflict when some of them are enabled at the same time. In this case, only one of them fires and the others become disabled. The choice for firing is done arbitrarily using an arbiter program.

In case that a transition in conflict with other transitions is decomposed as shown in **Figure 4**, these transitions should be coordinated by the station controller. If arbitration of the transitions is performed independently in separate subnets, the results may be inconsistent with the original rule of arbitration. Therefore the transitions should be arbitrated together as a group. On the other hand, arbitration of local transitions in conflict is performed by local machine controllers.

The Petri net based control structure with introduction of coordinator is shown in **Figure 5**. The control software is distributed into the station controller and machine controllers. The station controller is composed of the Petri net based controller and the coordinator. The conceptual Petri net model is allocated to the Petri net based controller for management of the overall system. For cooperative or exclusive tasks between robots, global transitions at the station controller are used to communicate the status of the robots. The detailed Petri net models are allocated to the Petri net based controllers in the machine controllers. Each machine controller directly monitors and controls the sensors and actuators of its machine.

The control of the overall system is achieved by coordinating these Petri net based controllers. System coordination is performed through communication between the coordinator in the station controller and the Petri net based controllers in the machine controllers as the following steps.

1) When each machine controller receives the start signal from the coordinator, it tests the firability of all

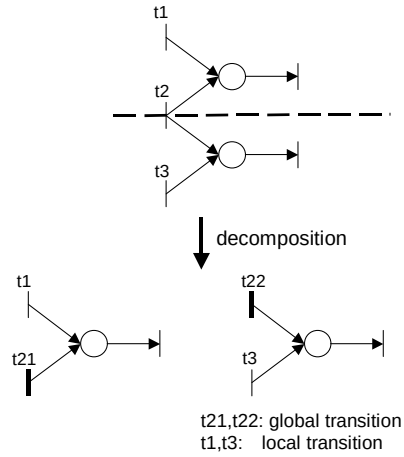



**Figure 4. Decomposition of transition in conflict**

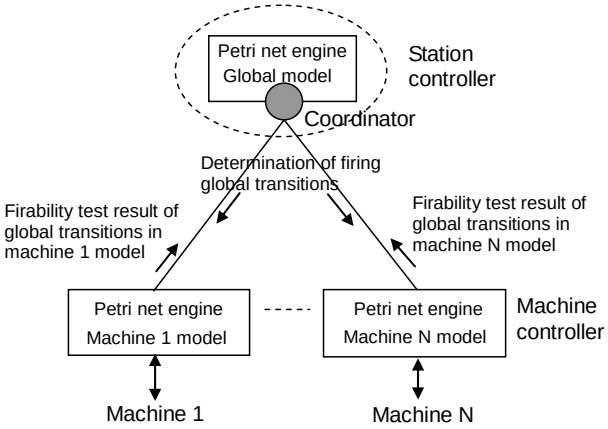



**Figure 5. Petri net based control structure with coordinator**

transitions in its own Petri net, and sends the information on the global transitions and the end signal to the coordinator.

2) The coordinator tests the firability of the global transitions, arbitrates conflicts among global and local transitions, and sends the names of firing global transitions and the end signal to the machine controllers.

3) Each machine controller arbitrates conflicts among local transitions using the information from the coordinator, generates a new marking, and sends the end signal to the coordinator.

4) When the coordinator receives the end signal from all the machine controllers, it sends the output command to the machine controllers.

5) Each machine controller outputs the control signals to its actuators.

Multilevel hierarchical and distributed control for large and complex manufacturing systems can be constructed such that the control system structure corresponds to the hierarchical and distributed structure of the general manufacturing system. The coordination mecha-

nism is implemented in each layer repeatedly as shown in **Figure 6**. The overall system is consistently controlled, such that a coordinator in a layer coordinates one-level lower Petri net based controllers and is coordinated by the one-level upper coordinator.

The details of coordination in a two-level control system composed of a global controller and several local controllers have been implemented as shown in **Figure 7**.

## 4. Implementation of Control System

### 4.1 Example of Workstation Task

The basic procedures of modeling and decomposition of robotic manufacturing systems are shown through a simple example. The example robotic manufacturing system is composed of two input conveyors, two machining centers, one handling robot, and one output conveyor as shown in **Figure 8**.

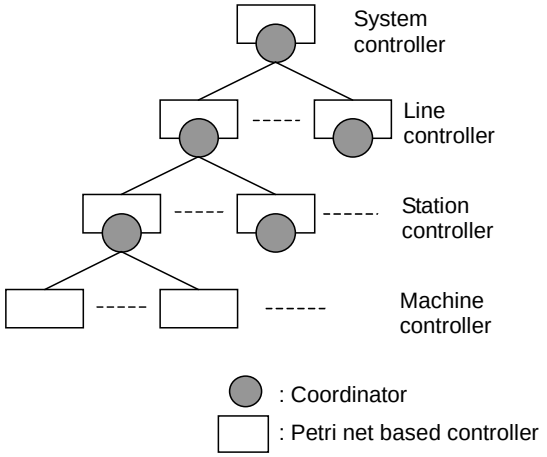



**Figure 6. Hierarchical and distributed control structure for overall manufacturing system**

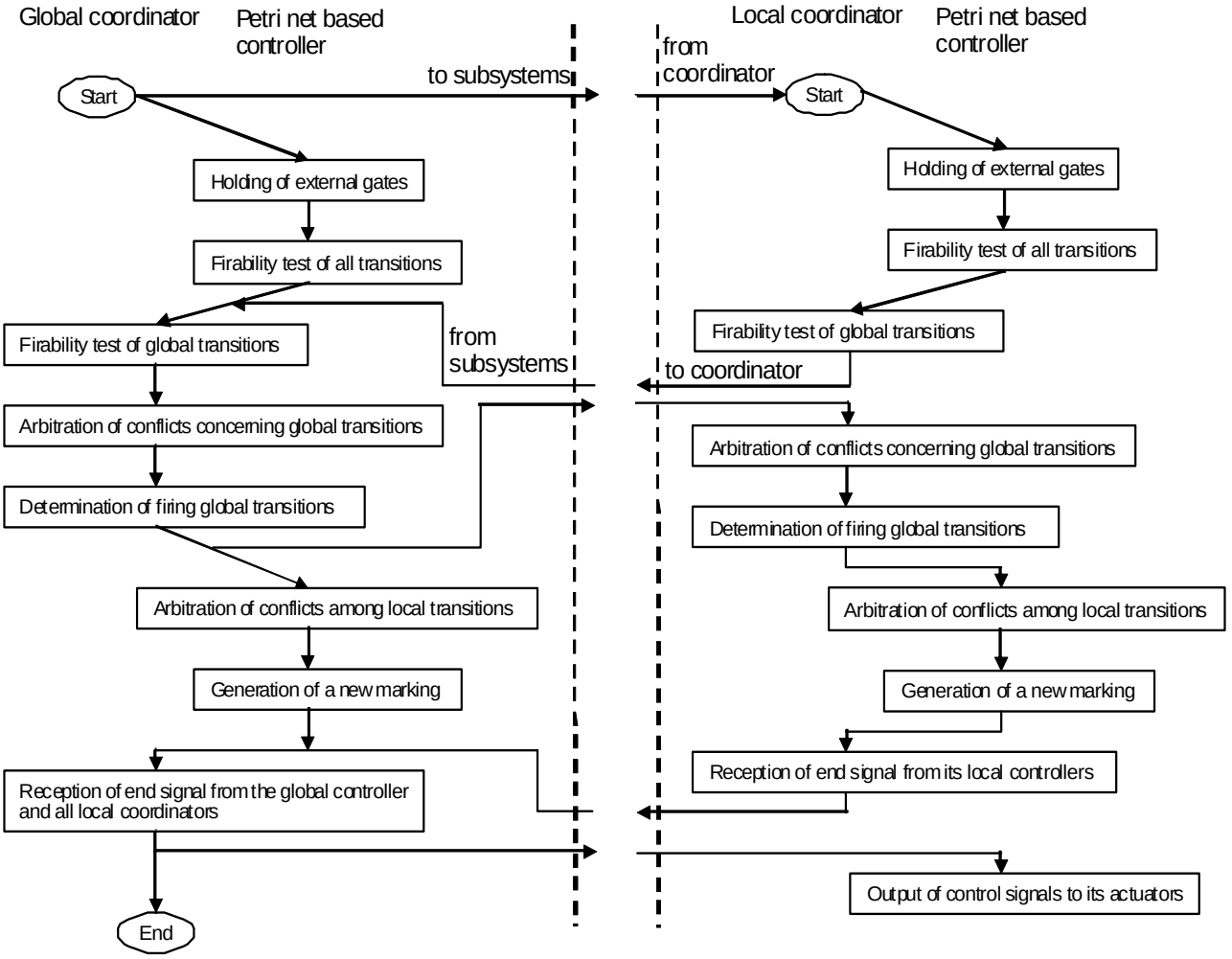



**Figure 7. Flowchart of coordination in two-level control system**

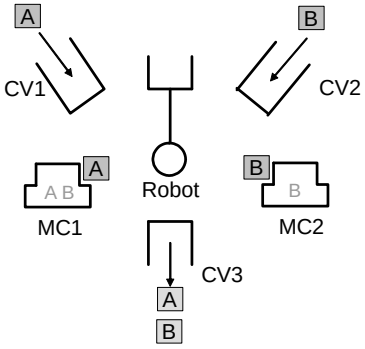



**Figure 8. Example of robotic manufacturing system**

The task specification of each equipment with the names of subtasks is as follows.

1) The conveyor CV1 carries a workpiece of type A into the workcell (CIN_A).

2) The conveyor CV2 carries a workpiece of type B into the workcell (CIN_B).

3) The robot loads the workpiece of type A into the machining center MC1 (LDA_MC1).

4) The robot loads the workpiece of type B into the machining center MC1 (LDB_MC1).

5) The robot loads the workpiece of type B into the machining center MC2 (LDB_MC2).

6) The machining centers process the workpieces each (PR_MC1, PR_MC2).

7) The robot unloads the processed workpiece from the machining center MC1 and carries them to the conveyor CV3 (UNLD_MC1).

8) The robot unloads the processed workpiece from the machining center MC2 and carries them to the conveyor CV3 (UNLD_MC2).

9) The conveyor CV3 carries the workpiece away (COUT).

## 4.2 Petri Net Based Modeling

From the viewpoint of the flow of workpieces, the task specification is summarized by Petri nets as shown in **Figure 9**.

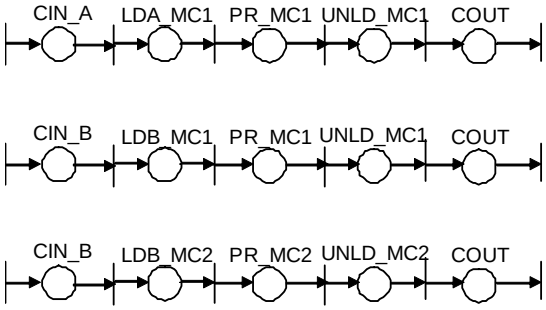



**Figure 9. Petri net representation of the task specification of the example system**

A global, conceptual Petri net model is first defined which describes the aggregate manufacturing process. At the conceptual level each task specification is represented as a place of the Petri net as shown in **Figure 10**, where the activity of each equipment is also represented as a place. Activities of the conveyor CV2, the machining center MC1 and the robot should be arbitrated based on the global Petri net model, because the places have two or more input/output transitions.

Based on the hierarchical approach, Petri nets are translated into detailed subnets by stepwise refinements from the highest system control level to the lowest machine control level [6]. At each step of detailed specification, some places of the Petri net are substituted by a subnet in a manner, which maintains the structural properties. **Figure 11** shows the detailed Petri net representation of subtasks: loading, processing and unloading in **Figure 10**.

The transitions among associated machines in the detailed Petri net representations imply the cooperative control structure in the overall system. For example, loading a workpiece of type A necessitates the cooperative or synchronized activities among the conveyor CV1, the machining center MC1, and the robot. First, "forward" operation to carry a workpiece is performed by the

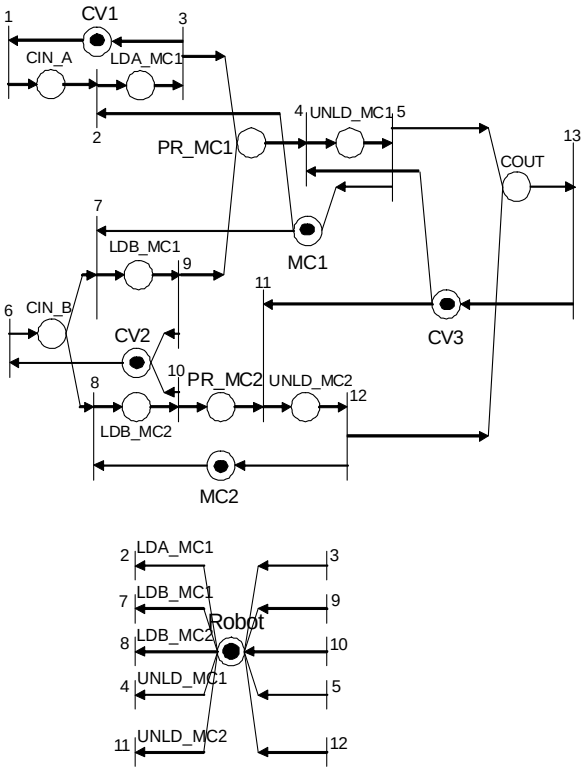
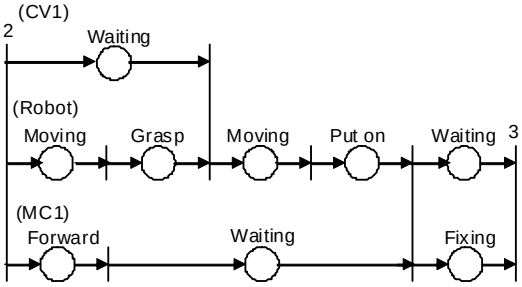
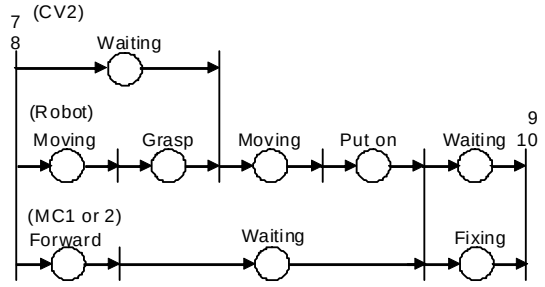



**Figure 10. Petri net representation of the example system at the conceptual level**

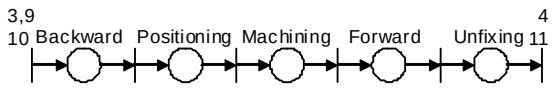
 



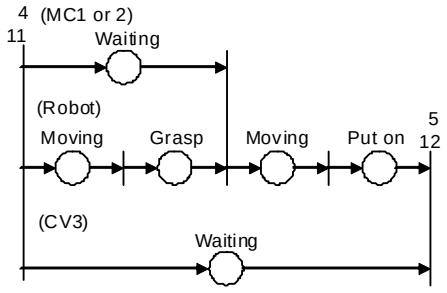

(a) Loading A to MC1 (LDA_MC1)




(b).Loading B into MC1 or MC2 (LDB_MC1, LDB_MC2)




(c) Processing (PR_MC1, PR_MC2)



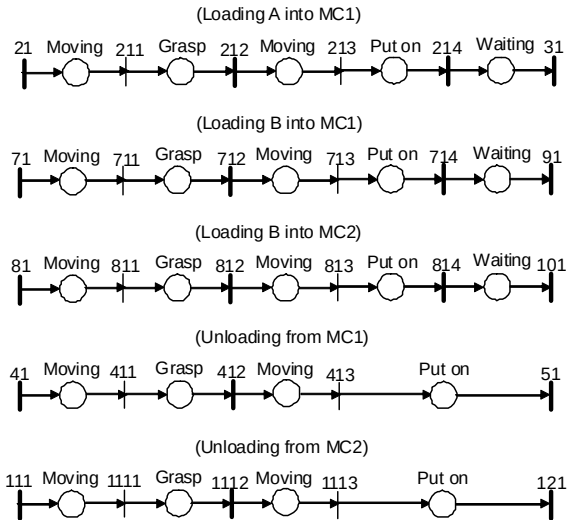

(d) Unloading A or B (UNLD_MC1, UNLD_MC2)

**Figure 11. Detailed Petri net representation of subtasks**

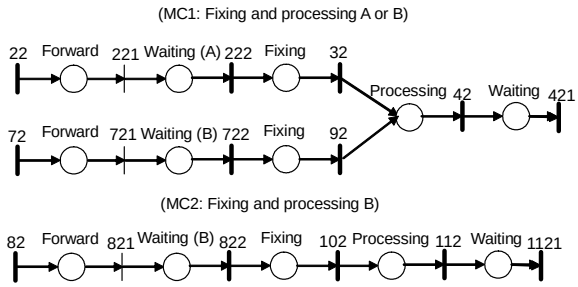
conveyor CV1. At the end of "forward" operation, when the robot is free, the "loading" operation is started. The conveyor CV1 starts waiting, the robot starts moving to grasp the workpiece, and the machining center starts moving forward to get the workpiece from the robot. After holding the workpiece, the robot starts moving to put it on the machining center and the conveyor CV1 is free. After putting on, the machining center starts fixing the workpiece, while the robot is waiting. After fixing the workpiece, the "loading" operation is finished.

## 4.3 Control System Design and Experiments

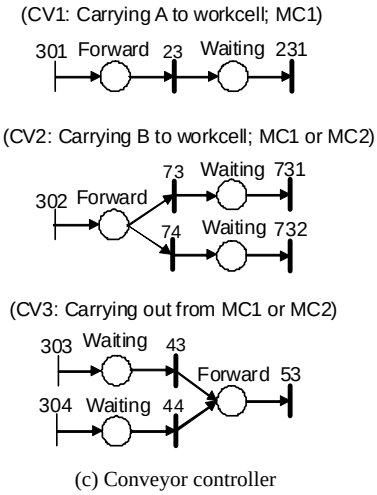
For the manufacturing system, an example structure of hierarchical and distributed control is composed of one station controller and three machine controllers (**Figure 2**). The Petri net executed in each machine controller is shown in **Figure 12**, simply by extracting the specified sequences of subtasks in the detailed Petri nets. For the implementation of the Petri net based control algorithm,




(a) Robot controller




(b) MC controller




(c) Conveyor controller

**Figure 12. Petri net representation of machine controllers ( ▌ : global transition)**

a transition of a Petri net model is defined using the names of its input places and output places; for example, t1-1=p1-1, -p1-11, where the transition no.1 (t1-1) of the subsystem no.1 is connected to the input place no.1 and the output place no.11. Using the names of transitions, global transitions are defined; for example, G1=t1-1, t2-1, t3-1 indicates that the global transition G1 is composed of the transition no.1of the subsystem no.1 (Robot controller), the transition no.1 of the subsystem no.2 (MC controller), and the transition no.1 of the subsystem no.3 (Conveyor controller). Then, the global transitions with comments of the example control system are as follows.

G1= t1-21, t2-22, t3-23 (start of loading A from CV1);
G2= t1-212, t3-231 (end of grasp A on CV1);
G3= t1-214, t2-222 (end of put A on MC1);
G4= t1-31, t2-32 (end of loading A into MC1);
G5= t1-71, t2-72, t3-73 (start of loading B from CV2);
G6= t1-712, t3-731 (end of grasp B on CV2);
G7= t1-714, t2-722 (end of put B on MC1);
G8= t1-91, t2-92 (end of loading B into MC1);
G9= t1-81, t2-82, t3-74 (start of loading B from CV2);
G10= t1-812, t3-731 (end of grasp B on CV2**);**
G11= t1-814, t2-822 (end of put B on MC2);
G12= t1-101, t2-102 (end of loading B into MC2);
G13= t1-41, t2-42, t3-303(start of unloading from MC1);
G14= t1-412, t2-421 (end of grasp on MC1);
G15= t1-51, t3-43 (end of put on CV3);
G16= t1-111, t2-112, t3-304 (start of unloading from MC2);
G17= t1-1112, t2-1121 (end of grasp on MC2);
G18= t1-121, t3-44 (end of put on CV3);

The hierarchical and distributed control system has been realized using a set of PCs. Each machine controller is implemented on a dedicated PC. The station controller is implemented on another PC. Communications among the controllers are performed using serial communication interfaces. A Petri net model includes control algorithms, and is used to control the manufacturing process by coincidence of the behavior of the real system with the Petri net model.

The names of global transitions and their conflict relations are loaded into the coordinator in the station controller. The connection structure of a decomposed Petri net model and conflict relations among local transitions are loaded into the Petri net based controller in a machine controller. By executing the coordinator and Petri net based controllers algorithms based on loaded information, simulation experiments have been performed. The Petri net simulators in the machine controllers initiate the execution of the subtasks attached to the fired transitions through the serial interface to the robot or other external machine. When a task ends its activity, it informs the simulator to proceed with the next activations by the external permissive gate arc. A machine controller controls one or more machines or robots using multithreaded programming [9]. Experimental results show that the decomposed transitions fire at the same time as the original transition of the detailed Petri net of the whole system task. Firing transitions and marking of tokens can be directly observed on the display at each time sequence using the simulator as shown in **Figure 13** [10].

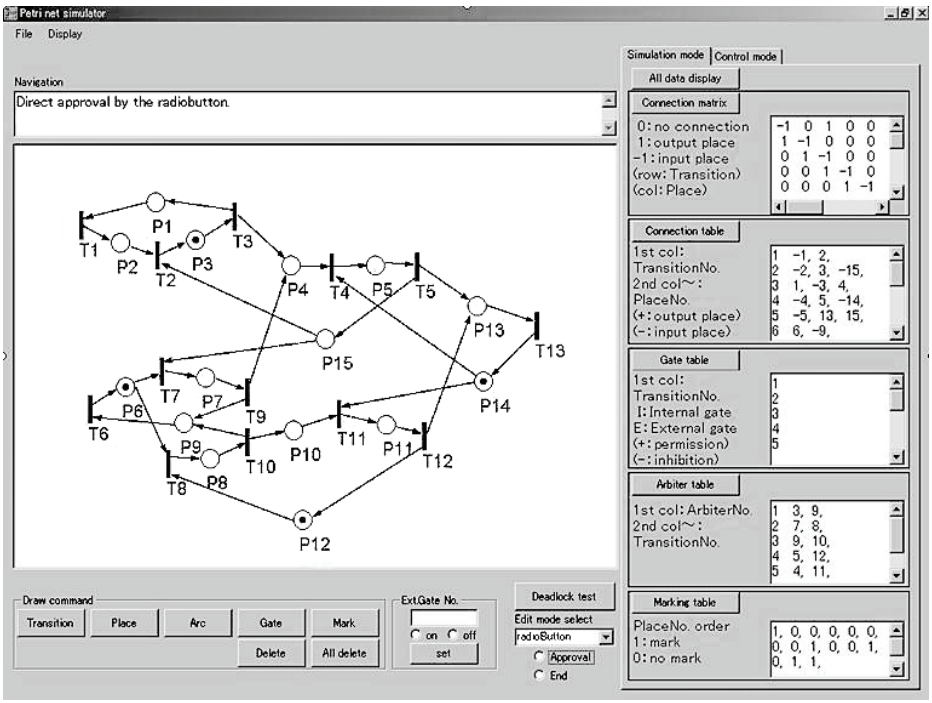



**Figure 13. View of Petri net simulator at the station controller**

## 5. Conclusions

A methodology to construct hierarchical and distributed control systems, which correspond to the structure of manufacturing systems, has been presented. The overall control structure of an example robotic manufacturing system was implemented using a communication network of PCs, where each machine controller is realized on a dedicated PC. The Petri net based control software is distributed into the station controller and machine controllers; the station controller executes the conceptual Petri net, and the machine controllers execute decomposed subnets. The controllers are arranged according to the hierarchical and distributed nature of the manufacturing system. The control software does not use the overall system model, and the decomposed Petri net model in each machine controller is not so large and easily manageable. Machine controllers are coordinated such that decomposed transitions fire at the same time and the task specification is completely satisfied. The Petri net model includes the control algorithm; control is executed in order that the behaviour of the Petri net model is in correspondence with that of the real system. Thus modeling, simulation and control of large and complex manufacturing systems can be performed consistently using Petri nets. The experimental control system uses conventional PCs with serial interfaces, but the performance of the control system can be improved using dual port shared memory and high-speed serial interfaces for communication between controllers.

## REFERENCES

[1] W. Reisig, "Petri Nets," Springer-Verlag, Berlin, 1985.

[2] M. Silva, "Petri Nets and Flexible Manufacturing," In G. Rozenberg, Ed., *Advances in Petri Nets* 1989, *Lecture Notes in Computer Science*, Springer-Verlag, Vol. 424, 1990, pp. 374-417.

[3] A. D. Desrochers and R. Y. Al-Jaar, "Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis," IEEE Press, 1995.

[4] E. J. Lee, A. Togueni, and N. Dangoumau, "A Petri Net Based Decentralized Synthesis Approach for the Control of Flexible Manufacturing Systems," *Proceedings of the IMACS Multiconference Computational Engineering in Systems Applications*, Lille, 2006.

[5] G. Bruno, "Model-based Software Engineering," Chapman & Hall, 1995.

[6] V. O. Pinci and R. M. Shapiro, "An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets," In G. Rozenberg, Ed., *Advances in Petri Nets* 1991, *Lecture Notes in Computer Science*, Springer Verlag, Vol. 524, 1991, pp. 227-252.

[7] K. Hasegawa, K. Takahashi and P. E. Miyagi, "Application of the Mark Flow Graph to Represent Discrete Event Production Systems and System Control," *Transactions of the SICE*, Vol. 24, No. 1, 1988, pp. 69-75.

[8] P. E. Miyagi, K. Hasegawa and K. Takahashi, "A Programming Language for Discrete Event Production Systems Based on Production Flow Schema and Mark Flow Graph," *Transactions of the SICE*, Vol. 24, No. 2, 1988, pp. 183-190.

[9] G. Yasuda, "Distributed Control of Multiple Cooperating Robot Agents Using Multithreaded Programming," *Proceedings of the* 16*th International Conference on Production Research*, Prague, 2001.

[10] G.Yasuda, "Implementation of Distributed Cooperative Control for Industrial Robot Systems Using Petri Nets," *Preprints of the* 9*th IFAC Symposium on Robot Control*, Gifu, 2009, pp. 433-438.

*J. Software Engineering & Applications*, 2010, 3: 446-454
doi:10.4236/jsea.2010.35050 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

# Experiences Analyzing Faults in a Hybrid Distributed System with Access Only to Sanitized Data*

**Ronald J. Leach**

Department of Systems & Computer Science Howard University, Washington DC, USA.
Email: rjl@scs.howard.edu

Received March 12th, 2010; revised March 26th, 2010; accepted March 28th, 2010.

## ABSTRACT

*In this paper we report on a work in progress assessing the faults observed and reported in a distributed, safety-critical, largely embedded system with both electrical and mechanical components. We illustrate why standard software testing techniques are not sufficient and indicate some of the technical and non-technical problems encountered in examining the faults and the initial results obtained. While the application domain is elevator operation, the techniques described here are general enough to apply to many other domains. Much of the data analyzed here would be considered imprecise in the software industry if it were used in software testing or to help increase fault tolerance. The paper includes a discussion of the use of multiple views of data, assessment of missing data, and analysis of informal information to produce its conclusions about fault avoidance and fault tolerance.*

***Keywords*: *Distributed System, Safety-Critical Systems, Fault Tolerance, Remote Monitoring***

## 1. Introduction

It is difficult to obtain useful information about the nature and distribution of faults in an actual distributed system, especially one that is safety-critical. Most companies and government organizations do not allow such information to be made available to external entities, even in sanitized form.

This lack of data poses a potentially enormous problem for researchers in fault-tolerance and distributed systems. It is very important to provide insights for researchers who might not have sufficient access to realistic data. Without such access, it is difficult to verify the practicality of research hypotheses. Hopefully the process described here, with a discussion of the analyses done, can provide insight and advance the research in this important field.

In this paper, we report on an evaluation of the root causes of faults in a safety-critical system and describe some of the partial solutions that were obtained. Our experience illustrates the difficulty in obtaining useful, realistic fault data from an operational safety-critical system. The system studied included several elevators in a high-rise building, with both internal and external monitoring and communications systems [1].

The situation examined in this paper is rather unusual as an example in the fault-tolerance community, because the fault and maintenance data analyzed was not reported in any sort of form that would ordinarily be used for a complete fault analysis, including analysis of either fault-tolerance or fault-avoidance [2,3].

We also observe that the reliability of electro- mechanical systems such as elevators might exhibit some of the characteristics of a "bathtub curve" typical in mechanical systems [4-6], or one more common in software [7]. The book [8] is devoted to systems with mechanical and electronic components, and the evolution of elevator control software systems is discussed in [9].

A 1996 version of a NASA standards document, Facility System Safety Guidebook NASA-STD-8719.7 states the following about software faults in hybrid systems [10].

Software faults may take three forms:

- The so-called honest errors made by the programmer in coding the software specification. These are simple mistakes in the coding process that result in the software behaving in a manner other than that which the programmer intended.
- Faults due to incorrect software specifications or the

*This research was partially funded by the National Science Foundation under grant number 0324818.

programmer's interpretation of these specifications. These errors may result from system designer's lack of full understanding of system function or from the programmer's failure to fully comprehend the manner in which the software will be implemented or the instructions executed. In this type of fault the software statements are written as intended by the programmer.

- Faults due to hardware failure. Hardware failures may change software coding. Thus such software faults are secondary in that they originate outside the software.

All these types of faults, as well as a considerable amount of human error, are present in this system. We note that a new draft standard STD-8719.7A is currently under NASA review. Other relevant research on the reliability of fault-tolerant, safety-critical; systems can be found in [11,12].

As will be discussed later in this paper, an informal verbal description of a problem with an on-site building manager and a conversation with a service company representative helped identify a set of faults that could be removed easily, leaving the system with a greater degree of resilience when other faults were encountered.

We note that some of the fault data was sanitized before it was made available to the author for the analysis that is described in this paper. Even so, some conclusions can be drawn about the major causes of faults, even with incomplete data.

We have removed all references to the particular companies that performed the initial installation and service of the set of elevators described here. The distributed card and password security system that the elevator access controls must interface with are described only at the highest levels, also. We have also sanitized the nature of any company database design in order to protect proprietary information.

Of course, simulation of elevator behavior in terms of picking up and letting off passengers is often used as a teaching tool. One of the earliest readily available such discussion is provided in Knuth [13]. A recent search on Google for the terms "elevator simulation" and "assignment" provided 517 matches.

## 2. The System Evaluated

The system evaluated in this work is a set of user- operated elevators that have multiple sets of controls, multiple alarms, and the capability to communicate with a remote monitoring device. All elevators are in the same high-rise building complex. The system is integrated with an access control system and electronic cards. The system currently complies with all existing safety codes in the geographical area.

The elevator system is over twenty years old and has some problems of age, wear and tear, and unavailability of parts.

Of course, it is not reasonable to expect that the programmers who wrote the original code for the microprocessors and related subsystems will still be with the company. In fact, there is no reason to expect that the company that originally designed and installed the elevator system is responsible for its maintenance. This is, of course, a typical situation in the software maintenance industry.

The entire system may be viewed as having several distinct features, most of which are illustrated below in **Figure 1**.

- The system contains a set of seven elevator cars that are positioned in three banks of two elevators each, with the remaining elevator essentially by itself, although another nearby elevator could be used in an emergency. The banks of elevators are several hundred feet apart.
- The alarm system in the elevators is audible to a local human monitoring system, with monitoring at all times of day and night. The on-site human monitor enters all problems into a log book and can call the elevator company's service center.
- There are also phones inside each elevator to enable a stranded user to contact the proper service personnel, or the fire department.
- In the late evening, the elevators automatically revert to limiting access to being controlled by electronic access cards.
- These electronic control cards are integrated into a building-wide security system with monitoring by the aforementioned human monitors and with each access entered into a database system.
- Microprocessors in each of the seven elevator cars can interact with communications devices that are able to transmit problem information to an off-site remote monitoring system.
- The microprocessors use a custom design and should

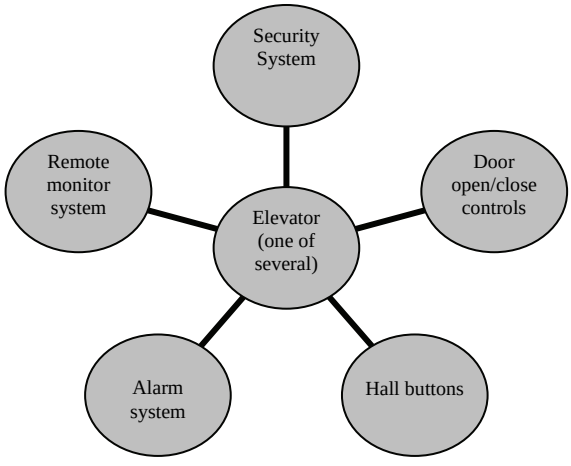



**Figure 1. An OV-1, high-level view of the interaction between several of the elevator's microprocessors and some of the other relevant computer-controlled systems**

be thought of as ASICs (Application-Specific Integrated Circuits). The lack of a standardized design makes the error rates of processors difficult to compare with other microprocessors of the same vintage. Hence it is impossible to use fault microprocessor data – even if it were made available – to determine if the reliability was typical of long-lived systems with high degrees of reuse.

- It appears that the microprocessors are not readily available for replacement in all of the elevator company's installed locations.
- Every call for elevator service is entered into a service database at the elevator company's central location. The elevator company's service supervisors can see this database monitoring system. This system can be viewed, in certain circumstances, by non-company personnel.

It is natural to ask why this system is an appropriate example to serve as the basis for a paper on software failures. Most modern elevators do not require a special operator and are operated by individuals who are, almost certainly, unaware of the safety, design, and control issues involves with their safe operation. Hence, there are multiple control and monitoring features, nearly all of which are computer-based for the system described in this paper.

There are microprocessors in several subsystems of this set of elevators. The microprocessors are custom designed and cannot be replaced easily by off-the-shelf components. Each elevator has the following computer components or computer system interfaces:

- Each elevator contains a microprocessor that selects options, based on the buttons that have been pressed. The microprocessor controls the operation of the doors (open, closed), as well as floor selection, based on the buttons pressed.
- Since there are separate controls on each side of the elevator cab, each side must have its own microprocessor.
- For six of the seven elevators, the buttons are rendered inoperable late at night by a security code set by a human operator at an in-building control center until a person uses their personal pre-assigned security code, which is entered using the in-car buttons on the keypad. Unless the code is entered correctly, the elevator car returns to the ground floor.
- For some of the higher floors, access also requires the swiping of an electronic security card.
- There are control units in sets of buttons, one for each floor, that allow the elevator to be called. Each of the control units contains a microprocessor for communication.
- There are sensors in each set of door panels. There are both interior and exterior doors in each elevator. These sensors make the doors stop closing if they encounter an obstacle, usually a human, but perhaps luggage or a grocery cart. These are controlled by microprocessors.
- Some doors have microprocessors to control smooth opening and closing of doors in the event of severe wind conditions affecting air flow within the elevator shafts. The elevator shafts have external air access, due to elevator safety regulations.
- All programming of the microprocessors is done off-site and, after testing, the microprocessors are deployed. There is only a minimal amount of on- site programming performed.

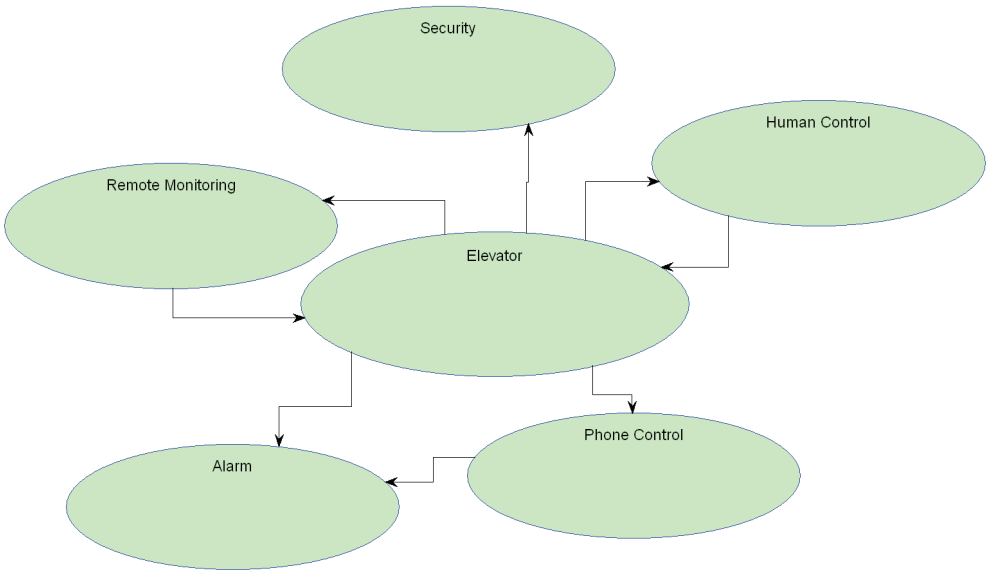



**Figure 2. An OV-2 view of the system, showing need lines**

- Each elevator contains a microprocessor and a communications path that sends a service code to the elevator company's central service location in the event of a malfunction.
- The company's central service location monitors all service calls, whether called in by an authorized human monitor or the electronic call system described above.
- There is a company proprietary database of service calls. In certain circumstances, the database may be made available for read-only access to selected customer representatives.

## 3. Modeling the System

To help understand and model the system's organization, we used the Department of Defense Architectural Framework, DoDAF and created the models using the System Architect for DoDAF tool from Telelogic. Representations of system operation were shown in what in DoDAF terminology is called "Operational Views." There are several types of standardized operational views:

- OV-1 consists of an informal, graphical representation of operations as well as explanatory text. It is informal in the sense that information provided in it is not included in any database or CASE tool. An OV-1 diagram of the system is provided in **Figure 1**.
- OV-2 is intended to track the need to exchange information from specific operational nodes that play a key role in the architecture to others. OV-2 does not depict the connectivity between the nodes.
- OV-3 (Operational Information Interchange Matrix) This view expresses the relationship between the three basic architecture data elements of an OV (operational activities, operational nodes, and information flow) in the form of an Excel spreadsheet, with a focus on the specific aspects of the information flow and the information content. This view is not provided in this paper, since it is somewhat redundant to the information included in the OV-2 and OV-5 diagrams.
- OV-4 (Organizational Relationships Chart) This view clarifies the various relationships that can exist between organizations and sub-organizations within the architecture and between internal and external organizations. Relevant organizations are the elevator service company, the company that built and installed the operator, the elevator inspector, the building management company, tenants, and, although informal, the organization of elevator users. This view is not provided in this paper, since it has been superceded by a new, somewhat confidential, contractual relationship that was developed as part of the analysis that was performed as a result of this study.
- OV-5 (Operational Activity Diagrams) The diagrams provided in this view represents the various activities that are performed by major components of the elevator management system. It is intended to do the following:
  - Clearly delineate the lines of responsibility for activities when coupled with OV-2
  - Uncover unnecessary operational control activity redundancy
  - Make decisions about streamlining, combining, or omitting activities
  - Define or flag issues, opportunities, or operational activities and their interactions (information flows among the activities) that need to be scrutinized further
  - Provide a necessary foundation for depicting activity sequencing and timing in OV-6a, OV-6b, and OV-6c

  In Telelogic's implementation of System Architect for DoDAF, three distinct OV-5 diagrams are created: an "Operational Activity Model Node Tree," a top-level "Node Activity Diagram," and a child-level "Node Activity Diagram." Each of these diagrams is discussed in detail. The methodology used in this diagram in System Architect is known as IDEF0, which is used to reflect data flows. The acronym IDEF stands for Integrated Computer-Aided Manufacturing (ICAM) DEFinition.

  The Operational Activity Model Node Tree Diagram indicates the major components of the elevator management system: human operation; elevator car operation; remote monitoring operation; security system operation, alarm system operation, and the phone system The tree structure indicates the major operational activity dependencies and their relation to the primary operational activity-management of the elevator's operation. For simplicity, only a few of the child nodes are shown in **Figure 3**.

  For each of the nodes in an operational activity diagram, a set of operations is allowed. We show a few of these in **Figure 4**, where we have presented an ICOM diagram. The acronym ICOM stands for Input Control Output Mechanism. Arrows for a few of each of these four types of interactions are shown in clockwise order, beginning at the left hand side of the highest level operational activity named "Manage elevator" in **Figure 4**.
- OV-6 (Operational Activity Sequence and Timing Descriptions) OV products discussed previously model the static structure of the architecture elements and their relationships. Many of the critical characteristics of a software architecture are only discovered when the dynamic behavior of these elements is modeled to incorporate sequencing and timing aspects of the architecture. Three standard types of sequence diagrams are in common use: Operational Rules Model (OV-6a), Operational State Transition Description (OV-6b), and Operational Event-Trace Description (OV-6c). Since our analysis of the failure data indicated that timing considerations did not appear to be a problem, these views are not discussed in this paper.

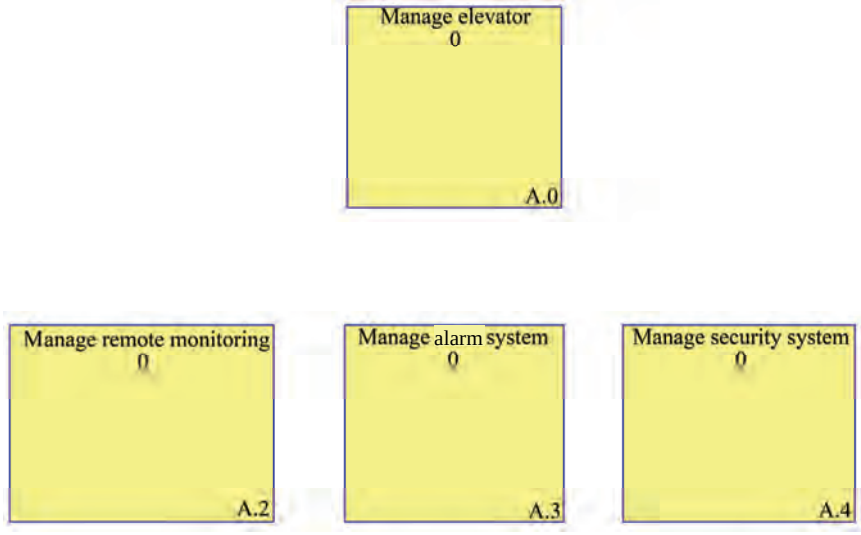



**Figure 3. An OV-5 Operational Activity Diagram, showing parent and some of the child nodes**

## 4. Relevant Non-Technical Issues

Elevators such as the ones described here are complex, far more so than one that might be found in, say, an expensive city townhouse. Therefore, the number of companies who can handle this type of installation is relatively limited to large companies with sufficiently large service staffs that can provide service at any time of the day or night.

It is common practice, but not uniformly guaranteed, that the company that performed the initial installation may not be given the service maintenance contracts once an initial warrantee period has expired. In order to protect confidentiality farther, we will always refer to two separate companies in this paper, although that may or may not be accurate in this particular situation, with the possibility that all service work was performed by a single company.

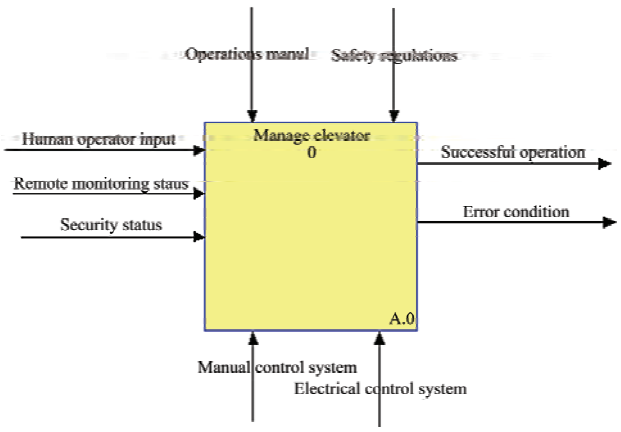



**Figure 4. An OV-5 diagram showing an operational activity with ICOM arrows**

To insure income streams, elevator service companies strongly prefer long-term service contracts. On the other hand, once the service contract is in hand, there is an incentive to not provide service beyond what is needed to maintain minimal operational service. Fortunately, safety is never ignored by any reputable elevator manufacturing or service company. Elevator safety systems are highly redundant; their designs resemble a multi-version programming scheme [2] with constant rollback states [5].

Of course, there are political issues about who pays for repairs beyond what is covered by these maintenance contracts, and who monitors the availability of the repairs of items not covered by these maintenance contracts. These issues suggest a somewhat adversarial relationship between customer and the elevator service company, especially if major repairs are anticipated. Independent analysis of faults by consultants is often of use. However, the dearth of companies with sufficient expertise to maintain elevator systems of this complexity encourages all parties to work together.

There are several sources of information that extend beyond the database discussed later. Either the building's manager or engineer, or both have been present during most of the elevator service calls during the period being examined. They have indicated verbally that some faults requiring service calls may have been caused by environmental conditions affecting microprocessors.

It is conceivable that some other problems may have been caused by interference with control microprocessors in individual elevator cars or near the hall buttons by cell phones. The elevators are over twenty years old and the design of the original shielding may not have considered the potential for cell phone interference.

There is one other non-technical issue that affects the analysis of the problem. It is conceivable that in certain

instances, data in the aforementioned company's proprietary database of service calls may provide some confidential information about failures of certain components. That might give some competitors an unfair advantage when bidding for maintenance or major upgrade contracts. This information must be kept within the security standards of the company. Hence, such data is sanitized considerably before release to anyone not employed by the company.

## 5. Current System Status

In **Figure 5**, we illustrate the availability of the individual elevators for service during a period of one year. The period shown was ended before the analysis described in this paper was undertaken. Of course, these percentages, while high, are never high enough for the elevator user who might be stuck in an elevator. The low availability of the first elevator is clearly a cause for concern.

The graph shows real data, but information on specific elevators has been deleted to preserve sensitive proprietary information. The diagrams are screen dumps taken directly from the elevator company's website.

While it is difficult to appreciate the differences between the percentages indicated, simple arithmetic shows that an elevator with an availability of 98.49% causes difficulty for its users 5½ days per year on average. Even the elevator with the highest availability was out of commission over ¾ of a day per year, on average.

Data for individual elevators was available for further analysis during the same reporting period. The results by month for the first elevator (the one most troublesome in **Figure 5**) are shown in **Figure 6**. Note that there was a wide range in availability of this particular elevator, which was the most troublesome of the elevators considered. Also, some of the other elevators had the desired 100% availability for multiple months.

Data for the other elevators has been omitted to save space.

It is important to understand the meaning of the data illustrated in **Figures 5** and **6**. A lack of availability might mean that a unit could not stop on a particular floor, that a hall button might not call the elevator unless it was pushed several times, or that a security code needed to be entered from a central location in the building. It did not mean that the elevator car was in any danger of falling. This does not happen on modern fail-safe elevators.

## 6. Analysis

In addition to the overall data on availability of the elevators during a one-year period illustrated in **Figure 5** and the monthly report for the same year, illustrated in **Figure 6**, data on this complex system were collected by the elevator maintenance company over an approximately nine-month period. There were a total of 74 service visits during that nine-month period. The results of each visit were entered into the company's service database, which is in the form of a Microsoft Excel spreadsheet. Since a spreadsheet normally contains less information than a database, and is less easily queried, data analysis is somewhat limited.

Initially, there was little concern about the discrepancy between the nine-month period of the service visits and the yearly data reported in **Figures 5** and **6**. This omission slowed down the analysis considerably, because it could have pointed out one of the most serious problems

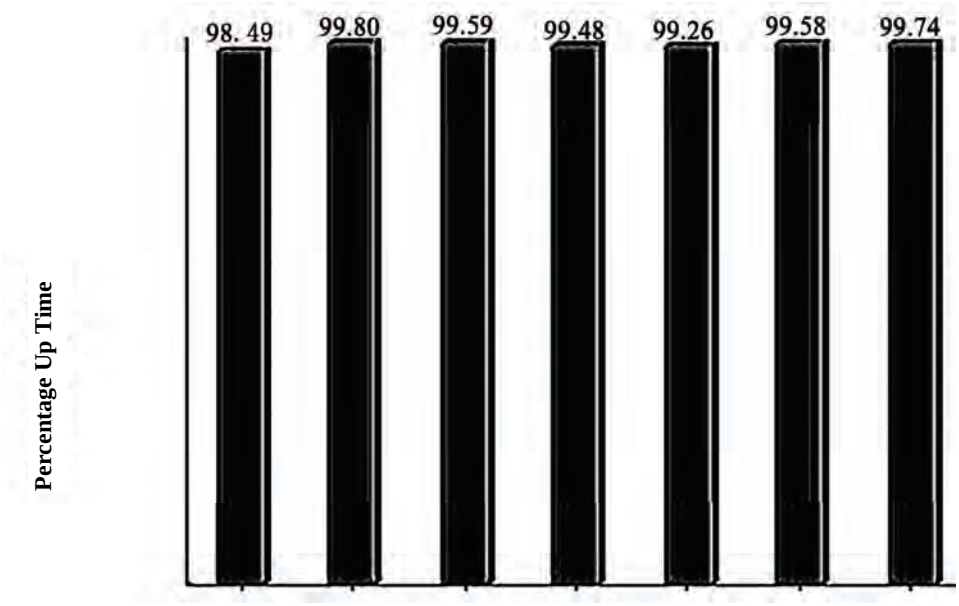



**Figure 5. Percentage of availability of operation of the elevators during a recent one-year period**

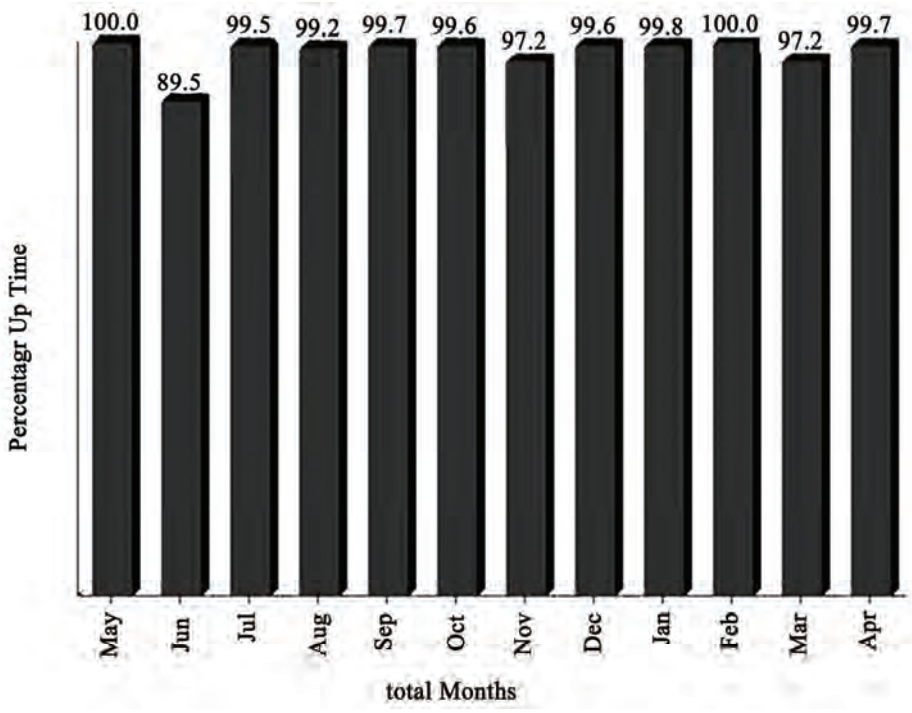



**Figure 6. Percentage of availability of operation by month for the most troublesome elevator during a recent one-year period**

immediately, had it been fully understood.

The entries in the database that, apparently, triggered the technician's maintenance service call are not very illuminating from the perspective of providing insight into computer faults. The categories indicated are limited to the following:

- Door_performance
- Checked/adjusted elevator operation and phone
- Maintenance on controller/mr_equipment
- Ropes
- Motor_generator
- General_maintenance_procedure
- Brake_elevator
- Hall_buttons
- Door operation/car doors
- Maintnance_on_car_door/operator/car_top/emg_ light

There were other views of this data that were somewhat more informative. One was a listing of 43 of the 74 service calls on which specific items that needed to be repaired or replaced were identified in more detail. These specific items could be classified as follows in this listing:

- There were 28 issues that required mechanical repairs.
- There were 12 issues that required the replacement of one or more specific mechanical parts.
- There were 5 issues that required computer hardware repairs.
- There were 2 issues that required computer software repairs.

In this listing, a few of the 43 service calls in which specific items that needed to be repaired or replaced were identified had multiple items, accounting for the 47 items described in the above list.

It is now obvious that there are discrepancies between the entries in the database of actions (repairs, replacements, hardware-specific repairs, software-specific repairs), the number of service calls, and, to some degree, the periods of unavailability of the elevators. It is natural to ask why there are such discrepancies.

One possibility that could be eliminated readily in the analysis of this data is the possibility of the elevator service company cutting corners. The elevators were under a long-term maintenance contract and, under the terms of the service contract, any unresolved issues would result in an additional service call to the elevator service company. Since the service calls required transportation of service personnel, it was in the elevator service company's best interest to minimize unnecessary extra travel trips. Hence, this possibility was rejected.

The elevator company's central dispatch office assigned technicians when faults were either detected or called in. Because of the redundancy in each of the elevator banks, service calls received lower priority in the dispatch office than locations with a single elevator. Occasionally, junior technicians were dispatched. For these reasons, it was felt that a statistical distribution of the time to fix problems would not produce more meaningful

data than simply reporting aggregated outages times.

It is clear that the entries in the technician's database (door_performance, hall buttons, checked/adjusted elevator_operation and phone, ropes, etc.) were restricted to match certain pre-defined categories. Thus, it is reasonable to assume that they might not provide much information on specific failures, especially for hardware and software failures.

When examining the discrepancies, it was noted that the time period were different. One set of data was for a nine-month period, while the other was for one year. It was important to know if the discrepancy was due to the way the elevator service company sanitized the data, or to the way data was collected. In particular, if the discrepancy was due to a problem data collection process, what caused this failure and did the result of this failure cause a cascade of related faults?

The explanation for this discrepancy was quite simple. Both the company's database and what we have called the secondary listing of which specific items that needed to be repaired or replaced were accurate, but did not show the failures at the times they were noted by human users and monitors. The data from the technician's service call database was accurate and reflected what was actually done (even though the codes were not always very helpful).

What happened is that the remote monitoring of what is called the "health and safety" of the elevators via the communications path between the elevator microprocessors had not been activated during the entire period. Reinitializing this communication allows microprocessors to be reset automatically if there were failures, providing much higher tolerance of hardware and software faults, thereby increasing availability.

How was it determined that the remote monitoring of elevator status was not working? (It was not clear from the documentation provided to the building – the customer – that there even was remote monitoring.) The information was obtained from the elevator company's newly appointed service manager, who gracefully provided access to the data.

A follow-up interview with the building manager of the building complex indicated another potential explanation for what had seemed to be an overly large number of microprocessor errors that required either resets or hardware replacement. The cleaning fluid used to clean the surfaces of both the in-elevator control panels and the much simpler hall buttons in several cases had seeped behind the decorative plates and caused electrical shorts. A simple change in the cleaning procedures reduced the number of observed faults.

The two actions–enabling the remote monitoring of microprocessor status and enacting new procedures for cleaning – caused a great reduction in faults, with almost no down time when failures did occur as a result of these remaining faults.

## 7. Conclusions and Suggestions for Future Work

Obviously, this was an unusual situation when compared to what is typically studied in the fault tolerance research and community. However, it may be more relevant to the practitioners of fault tolerance who are faced with solving a real-world problem.

The following techniques were especially useful in helping to determine the root causes of faults that led to system failures:

- While nearly all the reports in the maintenance service databases used pre-defined categories that, at first glance had little useful information, more detailed analysis indicated certain commonalities of faults.
- Interviews with knowledgeable people, such as the building's manager and the elevator service company's service manager, led to information that resulted in new policies (for keeping cleaning fluids and gels away from the microprocessors) and the proper use of the remote monitoring system.
- Unwritten information was useful, such as the existence of the remote monitoring database and the possibility of viewing this database by persons who are not employees of the elevator service company.
- Reasoning about missing things, such as the missing months in two different views of the maintenance database, led to an understanding of a major lapse in the use of the remote monitoring system.

It is likely that many of the lessons learned in this analysis can be useful to practitioners of fault tolerance who are faced with similar problems with the data available to them.

## REFERENCES

[1] Unnamed elevator company, Unnamed Service Database, 2008.

[2] A. Avizienis and J. P. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments," *IEEE Computer*, Vol. 17, No. 8, August 1984, pp. 67-80.

[3] B. Randell, "System Structure for Software Fault Tolerance," *IEEE Transactions on Software Engineering*, Vol. 11, No. 2, June 1975, pp. 220-232.

[4] R. Amuthakkannan, S. M. Kannan, K. Vijayalakshmi and N. Ramaraj, "Reliability Analysis of Programmable Mechatronics System Using Bayesian Approach," *International Journal of Industrial and Systems Engineering*, Vol. 4, No. 3, 2009, pp. 303-325.

[5] V. Dhudsia, "Guidelines for Equipment Reliability," *Technical Publication*, Sematech, Inc, 1997. http://www.sematech.org/docubase/document/1014agen.pdf

[6] G. K. Fourlas, "An Approach towards Fault Tolerant Hybrid Control Systems," *Control & Automation Mediter-*

*ranean Conference on MED*, Corsica, 27-29 June 2007, pp. 1-6.

[7] J. D. Musa, A. Iannino and K. Okumoto, "Software Reliability: Measurement, Prediction, Application," Mc-Graw-Hill, Inc. New York, 1987.

[8] R. Isermann, "Mechatronic Systems Fundamentals," Springer, London. 2003.

[9] K. Lee, K. C. Kang, E. Koh, W. Chae, B. Kim and B. W. Choi, "Domain-Oriented Engineering of Elevator Control Software: A Product Line Practice," *Proceedings of the First Software Product Line Conference*, Denver, August 2000, pp. 3-22.

[10] "Facility System Safety Guidebook," *NASA-STD*-8719.7, National Aeronautics and Space Administration, 1996.

[11] "The use of Computers in Safety Critical Operations," *Final Report of the Study Group on the Safety of Operational Computer Operations*, Health and Safety Commission, UK. http://www.hse.gov.uk/nuclear/computers.pdf

[12] N. Leveson, "Software Safety: Why, What, and How," *ACM Computing Surveys*, Vol. 18, No. 2, June 1986, pp. 125-163.

[13] D. E. Knuth, "Fundamental Algorithms," *The Art of Computer Programming*, 3rd Edition, Addison-Wesley, Reading, Massachusetts, Vol. 1, 1973.

*J. Software Engineering & Applications*, 2010, 3: 455-459
doi:10.4236/jsea.2010.35051 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Variability-Based Models for Testability Analysis of Frameworks

**Divya Ranjan[1], Anil Kumar Tripathi[2]**

[1]Dept. of Computer Science, Faculty of Science, Banaras Hindu University, Varanasi, India; [2]Dept. of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India.
Email: ranjan_divya@yahoo.co.in, aktripathi.cse@itbhu.ac.in

Received March 20th, 2010; revised April 3rd, 2010; accepted April 5th, 2010.

## ABSTRACT

*Frameworks are developed to capture the recurring design practices in terms of skeletons of software subsystems/ systems. They are designed 'abstract' and 'incomplete' and are designed with predefined points of variability, known as hot spots, to be customized later at the time of framework reuse. Frameworks are reusable entities thus demand stricter and rigorous testing in comparison to one-time use application. It would be advisable to guaranty the production of high quality frameworks without incurring heavy costs for their rigorous testing. The overall cost of framework development may be reduced by designing frameworks with high testability. This paper aims at discussing various metric models for testability analysis of frameworks in an attempt to having quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. The models considered herein particularly consider that frameworks are inherently abstract and variable in nature.*

***Keywords*: *Object-Oriented Frameworks, Variability, Customizability and Testability***

## 1. Introduction

Frameworks represent semi-codes for defining and implementing time-tested highly reusable architectural skeleton design experiences and hence become very useful in development of software applications and systems. As per Gamma *et al.* [1], famous in reuse literature as GoF, an object-oriented framework is a set of cooperating classes that make up a reusable design for a specific class of software which provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations. Being a reusable pre-implemented architecture, a framework is designed 'abstract' and 'incomplete' and is designed with predefined points of variability, known as ***hot spots***, to be customized later at the time of framework reuse [2]. A hot spot contains *default and empty interfaces*, known as ***hook methods***, to be implemented during customization [3,4]. Applications are built from frameworks by extending or customizing the framework, while retaining the original design. New code is attached to the framework through hook methods to alter the behavior of the framework. Hook descriptions provide guidance about how and where to perform the changes in the hook method to fulfill some requirement within the application being developed. With the help of *hook descriptions*, the framework developer passes his knowledge about what needs to be completed or extended in the framework, or what choices need to be made about parts of the framework in order to develop an application using the framework to framework users so that user is able to easily understand and use the hook. During framework reuse, the variant implementations of one or more hook methods, as needed, are created [5]. The code that the framework reuser writes, in order to create hook method implementations, is known as *application specific code* or *customized code.*

Frameworks are developed to capture the recurring design practices in terms of skeletons of software subsystems/ systems. It focuses mainly on the similarity amongst skeletons by identifying commonalities amongst them in terms of commonalities in the structure and functionality exercised by the concerned structure making use of the objects involved. It has been widely understood that the possibilities of variations provided by the hot spots and the corresponding hook methods etc. in the semi-code make it possible for a framework to be reused as extensively as permitted by the hot spots and the corresponding hook methods. The idea is simple that a framework permits variations across the application/ systems and attempts to design and code the common

parts and aspects of the structure.

One has to be very careful about developing *fault free reusable frameworks* because if the framework contains defects, the defects will be passed on to the applications developed from the framework [6]. The reusable framework thus demands stricter and rigorous testing in comparison to one-time use application [7,8]. It would be advisable to guaranty the production of high quality frameworks without incurring heavy costs for rigorous testing. This calls for analyzing testability of reusable artifacts so as to reduce the overall cost of framework based development.

Several techniques are specifically proposed to test object-oriented frameworks [1,6,9-12] and their instantiations [11,13-16].

There has been a little discussion upon the testability of frameworks in literature. As per Jeon *et al.* [2], the four factors that have direct influence upon framework testability are: controllability, sensitivity, observability and oracle availability. Ranjan and Tripathi [17] identified various factors and sub factors that affect the testability of frameworks so as to take care of those factors to bring high testability in frameworks. As per their observations, the factors that affect the testability of a framework are related to the characteristics of documentation of a framework, domain of a framework, design of a framework and the test support available for the framework testing like test tools, environments, reusable test artifacts and built-in tests etc. The concept of built-in tests is brought to frameworks with the intention of enhancing their testability [2,11,12,18].

In spite of wide importance and promotion of frameworks, over the last decades, a widely accepted set of measures to quantify its characteristics has not been established. Moreover, there is a complete lack of framework testability metrics related studies in literature that could produce quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. This paper proposes framework testability models that consider that frameworks are inherently abstract and variable in nature.

The paper is organized in four sections. Few important reasons behind the need of framework testability analysis are presented in Section 2, the proposed testability models for software frameworks appear in Section 3 and finally Section 4 presents conclusions.

## 2. Need of Framework Testability Analysis

Some obvious reasons for rigorous testability analysis of a framework could be summarized as:

1) A testable framework ensures *low testing cost* and helps in reduction of overall development cost of a framework which has been designed and implemented as a semi-code.

2) Frameworks are reusable entities and hence high testability is essential. As a testable system is known to provide increased reliability [19,20].

3) High testability brings *high reusability*. Many a times a framework reuser will want to test few features to assess its quality. If testing a framework is tough then framework reuser will hesitate in testing and using the framework and will seek to choose another framework or go for development without deploying a framework.

4) To calm obvious scientific curiosity that while writing test cases for frameworks why it is tougher in some case than the other cases or, so to say, why for one framework we had to think very hard before we were able to write a meaningful test suite, whereas for other frameworks we could generate test cases in a straightforward way.

5) Testability holds a prominent place as part of the *maintainability* characteristic in ISO 9126 quality model ISO, 1991, so this study also increases our understanding of software quality in general [21].

6) Framework testability analysis creates a base for formulating the strategy for designing highly testable frameworks *i.e. framework design for test* (FDFT).

## 3. Testability Models Considering Abstract and Variable Nature of the Frameworks

This section aims at discussing various metric models for testability analysis of software frameworks that consider that frameworks are inherently abstract and variable in nature and thereby they provide opportunity to develop multiple applications with its reuse.

### 3.1 A Testability Model Considering the Diversity/Commonality of Applications that the Framework Represents

Frameworks represent a set of applications that share commonalities. During design of a framework, the common aspects are concretely defined and are known as *fixed spots* whereas the variable aspects are designed abstract and are known as *hot spots*. This may not always be easy to work out a framework definition in terms of the variable aspects that it is going to deal with. However a crude measure may suggest that as many would be the variable aspects that difficult its testing is likely to be. Thus,

$$TE_{Fr} = f(\textit{Variable Aspect Fraction}) \quad (1)$$

where,

$$\textit{Variable Aspect Fraction} \propto \frac{Aspects_{Var}}{Aspects_{Var} + Aspects_{Com}} \quad (2)$$

$Aspects_{Var}$ = Variable aspects in a framework;

$Aspects_{Com}$ = Common aspects in a framework;

By variable aspect fraction, we mean the ratio of vari-

able aspects with respect to common aspects in the framework.

This fraction may be calculated before the design of the framework to get an idea whether the candidate framework will be testable or not.

$$Tb_{Fr} \propto \frac{1}{Variable\ Aspect\ Fraction} \quad (3)$$

This model is a preliminary one which just gives the idea of testability of frameworks before proceeding for its design. The next models give the idea of testability after a framework is coded or at least designed.

### 3.2 A Testability Model Considering Variability in terms of Hook Methods Provided by the Frameworks

Only an idea can be formed about the testability of the candidate framework through the above model. It can better be judged once the design of the framework is ready in terms of hot spots and the constituent hook methods. Hot spots consist of hook methods which represent points of variability by providing the calling interface to variable tasks [22]. Variability is number of possible variant implementations of framework's abstract behaviors. The variability within the family of architectural skeletons is constituted into the hot spots of a framework [4]. It is variability that makes one instantiation of the framework different from others [22].

Variability of a framework may be determined by summing up the measures of variability of each hot spot in the framework. The variability of a hot spot depends upon the number of possible alternative implementations of each its constituent hook methods.

We, thus, can express variability of a framework as below:

$$VAR_{Fr} \propto \sum_{i=1}^{N_{Hotspots}} \left( \sum_{j=1}^{N_{HMi}} N_{IHMij} \right) \quad (4)$$

where,

$VAR_{Fr}$ = Variability of a framework;

$N_{Hotspots}$= Total number of hot spots in the framework;

$N_{HMi}$ = Total number of hook methods in $i^{th}$ hot spot;

$N_{IHMij}$ = Total number of possible implementations of $j^{th}$ hook method in $i^{th}$ hot spot;

A discussion that the variability how affects testability of frameworks appears in [17]. Testing a framework requires testing possible implementations [6]. Hence, more the variability of a framework more the testing effort is required. We can write,

$$TE_{Fr} \propto VAR_{Fr} \quad (5)$$

Therefore, combining (4) and (5), testability of a framework is:

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{N_{Hotspots}} \left( \sum_{j=1}^{N_{HMi}} N_{IHMij} \right)} \quad (6)$$

The testability of frameworks, thus, is inversely proportional to the total number of possible implementations of all hook methods in all hot spots. Alternatively, more the number of possible implementations of hook methods, more the effort is required for the testing of the framework.

Further, while calculating testing effort we find that certain variations require less effort in their implementations and thus incur lesser share in testing effort. The above model does not take this aspect into consideration and thus a stronger model is required. The next testability model which is based on the customizability of the framework is a stronger model than the present one, as it also considers the effort required for implementing variants of hook methods.

### 3.3 A Testability Model Considering Customizability of Hook Methods Provided by the Frameworks

Framework is customized by framework reusers to create concrete application software systems. The *customizability* of a framework may be interpreted by knowing how easy it is to customize (tailor) the framework. A framework is customized either by sub-classing (in case of white-box frameworks) or by composing preexisting components (in case of black-box frameworks). A testable framework should be highly customizable so that during testing of the framework various instantiations can be easily produced and subsequently tested. A discussion that the customizability how affects testability of frameworks appears in [17]. The customization of a hook method may require following:

**1) Changing** some object or method by the means of the mechanisms like inheritance, extensions, configuration, parameterization, template instantiation etc. [23]. What changes to make is defined in the *changes* section of a hook method description and

**2) Making assumptions** regarding other objects or methods and **understanding the assumptions** that other objects or methods have to make regarding this hook method. What assumptions are to make is defined in the *pre-condition constraints*¸ *post-condition constraints*, *uses* and *participants* sections of a hook method description.

Thus, we may express *Customization Effort* (CE) required for implementing a variant of a *hook method*, as below:

$$CE_{ImHM} \propto \sum_{i=1}^{N_{Changes}} Change\_load_i \times \sum_{i=1}^{N_{Assump}} Assumption\_load_i \quad (7)$$

**Table 1. Applicability/intention of the proposed framework testability metric models**

| S.No | Category | Framework Testability Metric Model | Applicability of the Model |
|---|---|---|---|
| 1. | Testability models considering abstract and variable nature of frameworks | Testability Model Considering the Diversity/Commonality of Applications that the Framework Represents | To have an idea about framework testability even ***before starting the design*** of framework. |
| 2. | -- do-- | Testability Model Considering Variability in terms of Hook Methods provided by the Frameworks | When framework ***variability*** is prominent during design and development. |
| 3. | -- do-- | Testability Model Considering Customizability of Hook Methods provided by the Frameworks | When framework ***customizability*** is prominent during design and development. |

where,

$CE_{\text{ImHM}}$ = Customization Effort required for implementing a variant of a hook method (*HM*);

$N_{Changes}$ = Number of changes to be made in some object or method during implementation of the hook method;

$Change\_load_i$ = Heaviness of $i^{th}$ change;

$N_{Assump}$ = Number of assumptions involved regarding objects and their interactions;

$Assumption\_load_i$ = Heaviness of $i^{th}$ assumption;

It is for sure that in Equation (7) $\sum_{i=1}^{N_{Changes}} Change\_load_i$ as well as $\sum_{i=1}^{N_{Assump}} Assumption\_load_i$ will never be zero because customizing a hook method would require at least one *change* and involve assumptions regarding at least one *participant*.

Further the *CE* of one hook method, which consists of customizing or implementing all its possible variants, may be defined as following

$$CE_{HM} \propto \sum_{i=1}^{N_{IHMi}} CE_{ImHMi} \tag{8}$$

where,

$CE_{HM}$ = Customization Effort (*CE*) for a hook method;

$N_{IHMi}$ = Total number of possible implementations for the hook method;

$CE_{ImHMi}$ = *CE* required for $i^{th}$ implementation of the hook method;

Now we will consider the *CE* of the framework itself. It would consist of the *CE* of all the hook methods of all the hot spots. This relation can be expressed as follows:

$$CE_{Fr} \propto \sum_{i=1}^{N_{Hotspots}} \left( \sum_{j=1}^{N_{HMi}} \left( \sum_{k=1}^{N_{IHMij}} CE_{ImHMijk} \right) \right) \tag{9}$$

where,

$CE_{ImHMijk}$ = Customization Effort for implementing $k^{th}$ variant of $j^{th}$ hook method of $i^{th}$ hot spot;

$N_{IHMij}$ = Total number of possible implementations of $j^{th}$ hook method in the $i^{th}$ hot spot of the framework;

$N_{HMi}$ = Number of hook methods in $i^{th}$ hot spot of the framework;

$N_{Hotspots}$ = Total number of hot spots in the framework;

Since, the testing effort of the framework depends upon the customization effort of the framework. So, we can get the framework testability model based on the customizability of framework, from Equation (9) as below:

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{N_{Hotspots}} \left( \sum_{j=1}^{N_{HMi}} \left( \sum_{k=1}^{N_{IHMij}} CE_{ImHMijk} \right) \right)} \tag{10}$$

Each of these testability metric models has different intention or applicability which is discussed in the table (**Table 1**) below, however, more than one model may also be employed at the same time.

## 4. Conclusions

It is mainly the *incomplete* and *abstract* nature of frameworks that makes it difficult to test than an object-oriented software system. In the present paper we proposed few preliminary framework testability models that consider that frameworks are inherently abstract and variable in nature. These models could produce quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. Authors found a complete lack of framework testability metrics related studies in literature.

## REFERENCES


[1] E. Gamma, R. Helm, R. Johnson and J. M. Vlissides, "Design Patterns: Elements of Reusable Object-oriented Software," Addison-Wesley Professional Computing Series, 1994.

[2] T. Jeon, S. Lee and H. Seung, "Increasing the Testability of Object-oriented Frameworks with Built-in Tests," *Lecture Notes in Computer Science*, Vol. 2402, January 2002, pp. 873-881.

[3] G. Froehlich, H. J. Hoover, L. Liu and P. Sorenson, "Designing Object-oriented Frameworks," *CRC Handbook of Object Technology*, CRC Press, 1998, pp. (25)1-21.

[4] W. Pree, "Design Patterns for Object-oriented Software Development," Addison-Wesley, 1995.

[5] H. A. Schmidt, "Systematic Framework Design by Generalization," *Communications of the ACM*, Vol. 40, No. 10, October 1997, pp. 48-51.

[6] J. Al-Dallal and P. Sorenson, "System testing for Object-oriented Frameworks Using Hook Technology," *Proceedings of the* 17*th IEEE International Conference on Automated Software Engineering*, Edinburgh, September 2002, pp. 231-236.

[7] J. S. Poulin and J. M. Caruso, "Determining the Value of a Corporate Reuse Program," *Proceedings of the IEEE Computer Society International Software Metrics Symposium*, Baltimore, May 1993, pp. 16-27.

[8] E. J. Weyuker, "Testing Component-based Software: a Cautionary Tale," *IEEE Software*, Vol. 15, No. 5, September 1998, pp. 54-59.

[9] R. V. Binder, "Testing Object-oriented Systems: Models, Patterns, and Tools," Addison-Wesley Professional, 1999.

[10] M. E. Fayad and D. C. Schmidt, "Object-oriented Application Frameworks," *Communications of the ACM*, Vol. 40, No. 10, October 1997, pp. 32-38.

[11] Y. Wang, D. Patel, G. King, I. Court, G. Staples, M. Ross and M. Fayad, "On Built-in Test Reuse in Object-oriented Framework Design," *ACM Computing Surveys*, Vol. 32, No. 1, March 2000, pp. 7-12.

[12] T. Jeon, H. W. Seung and S. Lee, "Embedding Built-in Tests in Hot Spots of an Object-oriented Framework," *ACM Sigplan Notices*, Vol. 37, No. 8, August 2002, pp. 25-34.

[13] J. Al-Dallal and P. Sorenson, "Estimating the Coverage of the Framework Application Reusable Cluster-based Test Cases," *Information and Software Technology*, Vol. 50, No. 6, May 2008, pp. 595-604.

[14] J. Al-Dallal and P. Sorenson, "Reusing Class-based Test Cases for Testing Object-oriented Framework Interface Classes," *Journal of Software Maintenance and Evolution*: *Research and Practice*, Vol. 17, No. 3, May 2005, pp. 169-196.

[15] J. Al-Dallal and P. Sorenson, "Testing Software Assets of Framework-based Product Families During Application Engineering Stage," *Journal of Software*, Vol. 3, No. 5, May 2008, pp. 11-25.

[16] W. Tsai, Y. Tu, W. Shao and E. Ebner,"Testing Extensible Design Patterns in Object-oriented Frameworks Through Scenario Templates," *Proceeding of* 23*rd Annual International Computer Software and Applications Conference*, Phoenix, October 1999, pp. 166-171.

[17] D. Ranjan and A. K. Tripathi, "Testability Analysis of Object-oriented Frameworks," *The Journal of Defense Software Engineering*.

[18] M. E. Fayad, Y. Wang and G. King, "Built-in test reuse," In: M. E. Fayad, *et al.*, Ed., *The Building Application Frameworks*, John Wiley and Sons, 1999, pp.488-491.

[19] R. V. Binder, "Design for Testability in Object-oriented Systems," *Communications of the ACM*, Vol. 37, No. 9, September 1994, pp. 87-101.

[20] J. M. Voas and K.W. Miller, "Software Testability: the New Verification," *IEEE Software*, Vol. 12, No. 3, May 1995, pp. 17-28.

[21] "Software Engineering-Product Quality," *ISO/IEC* 9126, 2001.

[22] G. Succi, A. Valerio, T. Vernazza, M. Fenaroli and P. Predonzani, "Framework Extraction with Domain Analysis," *ACM Computing Surveys*, Vol. 32, No. 1, March 2000.

[23] G. Butler, "Object-oriented Frameworks," 15*th European Conference on Object-Oriented Programming*, Tutorial Budapest, 2001.

*J. Software Engineering & Applications*, 2010, 3: 460-471
doi:10.4236/jsea.2010.35052 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# A Conflicts Detection Approach for Merging Formal Specification Views

**Fathi Taibi[1], Fouad Mohammed Abbou[2], Md. Jahangir Alam[2]**

[1]University of Tun Abdul Razak, Malaysia; [2]Multimedia University, Malaysia.
Email: 1taibi@unitar.edu.my, {fouad, md.jahangir.alam}@mmu.edu.my

Received April 20th, 2009; revised June 2nd, 2009; accepted June 10th, 2009.

## ABSTRACT

*Specifying software requirements is an important, complicated and error prone task. It involves the collaboration of several people specifying requirements that are gathered through several stakeholders. During this process, developers working in parallel introduce and make modifications to requirements until reaching a specification that satisfies the stakeholders' requirements. Merge conflicts are inevitable when integrating the modifications made by different developers to a shared specification. Thus, detecting and resolving these conflicts is critical to ensure a consistent resulting specification. A conflicts detection approach for merging Object-Oriented formal specifications is proposed in this paper. Conflicts are classified, formally defined and detected based on the results of a proposed differencing algorithm. The proposed approach has been empirically evaluated, and the experimental results are discussed in this paper.*

***Keywords*: *Formal Specification*, *Object-Oriented*, *Collaboration*, *Merge Conflicts*, *Consistency***

## 1. Introduction

The development of large-scale software systems requires the collaboration [1] of hundreds of developers working on different aspects of the same system. Often, this leads to the creation of different but related documents. These documents (views) could be in the form of design models, software specifications, source code, etc. During a particular collaborative activity, a resulting local view needs to be merged [2] with the version of the document available in a shared repository. The latter shared document encloses all the modifications made locally and checked (integrated) into the repository at that point in time. A merging approach must integrate the changes made and must ensure that the merging result is consistent by detecting and resolving merge conflicts [3].

Specifying software requirements is an important, complicated and error prone task that involves the collaboration of several peopled specifying requirements that are gathered through several stakeholders. Studies have shown that most of the problems with software projects such as not meeting the needs of stakeholders, late delivery and budget overrun can be traced back to problems with the requirements [4].

Merging requirements specified informally is unpractical, inefficient, error prone, and time consuming due to the ambiguous and imprecise nature of natural languages and most of the graphical notations used. Formal methods [5] offer a better alternative because of their precise and accurate nature. Object-Oriented (OO) formal methods, such as Object-Z [6], combine the strengths of two worlds: the world of formal languages and the world of OO methods. When used to specify software requirements, they produce specifications that are precise, clear, and highly reusable. Thus, making them suitable to be used when developing specifications collaboratively why they can be manipulated systematically.

Conflicts detection requires the calculation of the delta(s) representing the modifications made locally compared to a shared view. Analyzing these deltas allows the detection of conflicts before integrating their content with the shared view. Rules could be formally defined for these conflicts to uniformly detect different type of violations such as those concerned with the loss of update, the well formedness of the view, the avoidance of all kind of redundancies, etc.

Efficient merging frameworks support collaboration and distributed development, and when used at an early phase of the software development such as when specifying software requirements, they enable detecting conflicts that will cost higher to detect and resolve during later stages of development.

Employing unique identifiers for the elements of the merged documents, such as in [7], introduce tool dependency. The latter term refers to approaches that are dependent on the tools used to create and modify the documents to be merged. In order to support the tool independence requirement [8], merging should not rely on elements' unique identifiers. Thus, a differencing approach is needed to produce a list of the created/deleted and modified elements as well as the created/modified relationships (or links) between the elements of the specifications. Differencing two specifications should be based their computed similarities (matches). The similarities between the elements of two specifications could be calculated accurately based on syntactic as well as structural similarity.

Several existing work on model differencing and version control such as CVS [9] adopts line-based (textual) management. Textual merging tools have been used to some extend in industry, and in [10] it has been reported that around 90% of the changed file could be merged without any problem. However, the remaining changes cannot be merged automatically because there is no consideration to the syntactic or semantic information of the files. In order to manage the changes of specifications there is a need to work at the granularity of a logical unit component such as a class rather than at the granularity of a line. Moreover, textual merging can only detect very basic conflicts, as it does not take into account the specific structure of the processed documents. Furthermore, it gives rise to unimportant conflicts [11] such as code comments that have been modified, line breaks, etc. Thus, transforming specifications into a textual format used by existing tools cannot solve the problem of merging specifications.

Most of existing merging approaches process the manipulated documents as trees, which is restrictive and not applicable to a large number of documents including software requirements specifications. Moreover, most of the surveyed approaches are inadequate for merging specifications due to their limitations in uniformly defining conflicts, and accurately detecting and resolving them. The domain independent approaches surveyed lack accuracy when used to merge specifications, and to our knowledge, there is no existing merging approach intended specifically for Object-Oriented formal specifications.

In this paper, an approach is proposed to detect and resolve conflicts when merging OO formal specifications. The approach comprises three parts. The first part consists of comparing specifications to produce deltas differentiating them. The second part consist of integrating (merging) the deltas into a shared specification and the third part consists of checking the deltas against defined consistency rules to detect and resolve any consistency violations that might arise during merging. Collaborative formal specification is discussed in the next section. This is followed by proposing an algorithm for differencing OO formal specifications. After that, an approach for detecting and resolving merge conflicts is discussed. Then, the proposed approach is empirically evaluated. This is followed by discussing related work and the last section concludes the paper and discusses future work.

## 2. Collaborative Development of Formal Specifications

Software development is a collaborative activity as it involves several people working on different aspects of the same software project. Most often, collaboration is the key to the success of a software project. During the specification of software requirements, developers working in parallel add, remove, and modify requirements until reaching a description of the system (or subsystem) that satisfies the stakeholders' requirements. This collaborative nature raises the needs for frameworks to support the merging of software specifications.

Asynchronous collaboration allows members of a group to modify copies of a shared specification in isolation, working in parallel and afterwards synchronizing their copies to reestablish a common view. This gives a great deal of flexibility, and matches the needs of collaborative requirements specification. In such environments, three basic operations are applied on a shared repository of specifications: check-out, modify, and check-in. The check-out operation consists of importing a copy of the latest version (for example at time $t_0$) of a shared specification ($S_{Base}$) from the repository in order to perform some modifications on it. These modifications represent new requirements that have yet to be specified or different views on the requirements that have already been specified. The modifications made lead to the creation of a new version $S_{Local}$ of the shared specification $S_{Base}$. Check-out is not applicable to the first developer creating the first version of a given specification and checking it into the repository. However, later on, he/she can perform a check-out operation to make any new changes if needed. In case at time $t_1$ ($t_1 > t_0$), the shared specification has evolved into a new version $S_{Rep}$ after undergoing some changes made by a different developer. The check-in operation consists of merging the local and shared versions of the specification in case of two-way merging. In case of three-way merging, the modifications made locally need to be adjusted according to those who have already been integrated in $S_{Rep}$, *i.e.* using two specifications ($S_{Local}$ and $S_{Rep}$) and their ancestor ($S_{Base}$) in the merge. Integrating all the modifications made with the base specification is possible after that. The following figure illustrates the basic operations of this collaborative environment:

A developer 'X' imports (check-out) a shared specification ($S_{Base}$) from a Specification repository. He/she

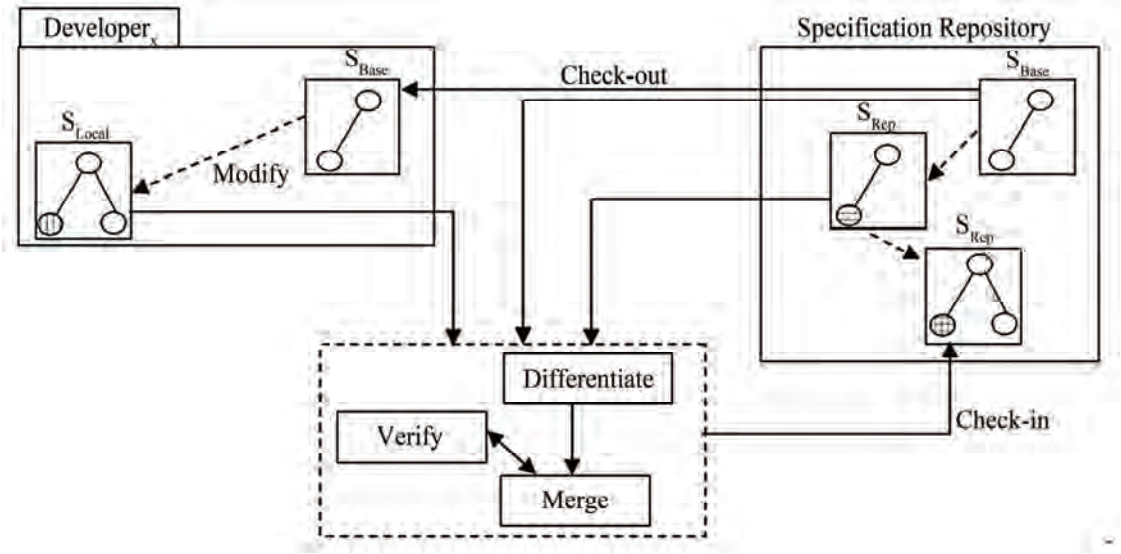



**Figure 1. The operations supporting collaboration**

performs some modifications on it locally. These modifications lead to the creation of a new version ($S_{Local}$) of the specification. For X to check-in $S_{Local}$ into the specification repository, there is a need to discover the exact modifications (delta) made. The operations contained in this delta allow detecting the conflicts that might arise during the merge. In case of three-way merging, there is a need to identify the exact modifications made locally and those that have evolved $S_{Base}$ into $S_{Rep}$ because of some parallel modifications made and integrated by a different developer 'Y'. In addition to the latest version of the shared specification, a history of all the deltas applied to it is also stored in the repository. Thus, re-obtaining $S_{Base}$ from which $S_{Rep}$ originates is achieved by reversing the effect of the last integrated delta. Using $S_{Base}$ and the two deltas, a three-way merging could take place where conflicts caused by the parallel modifications need to be detected and resolved before integrating them.

The proposed framework incorporates three approaches to perform the required tasks. The first approach "Differentiate" is intended to differentiate between the to-be-merged specifications. The differentiation process involves the production of deltas containing the exact modifications (operations) made. The latter deltas are obtained using the computed similarities that exist between the specifications' elements. Merging based on differencing is referred to as operation-based merging [11] and is efficient in case of large models as the number of operations that transform a version into another one is statistically smaller than the number of model elements. Furthermore, it provides a better platform for conflicts detection and resolution. The second approach "Merge" is intended to merge the modifications made with the shared specification. The specification obtained through this process must be consistent. Thus, the third approach "Verify" is intended to detect and resolve merge conflicts. The differencing and verifying approaches are discussed in detail in sections 3 and 4.

## 3. Differencing Object-Oriented Formal Specifications

Given a basic set *S* of all the specifications, differencing between two specifications $S_1$ and $S_2$ is the process of identifying the exact set of operations (transformations) that allow obtaining $S_2$ from $S_1$. As a motivation example, consider the following classes representing a shared specification, and two versions representing some parallel modifications made to it by two different developers. Object-Z notation has been used to specify the three versions.

The class *Professor* includes two operations *New* and *Affiliate* that are the only elements visible outside the class. The operation *New* is used to assign values to the state attributes *Id*, *Name* and *Expertise*, which represents a professor's personal data. The operation *Affiliate* is used to assign a value to the state attribute *Faculty*. The classes *Academician* and *TeachingStaff* are the result of some parallel modifications made to the class *Professor* by two different developers. In the class *Academician*, in addition to the class name that has been changed, the operation *Affiliate* has been removed and its functionality has been delegated to the operation *New* that is the only class' element visible. In the class *TeachingStaff*, in addition to the class name that has been modified, the attribute *Expertise* has been removed while the operations *New* and *Affiliate* have been renamed as *Add* and *Join* respectively. In addition, the operation *New* has been

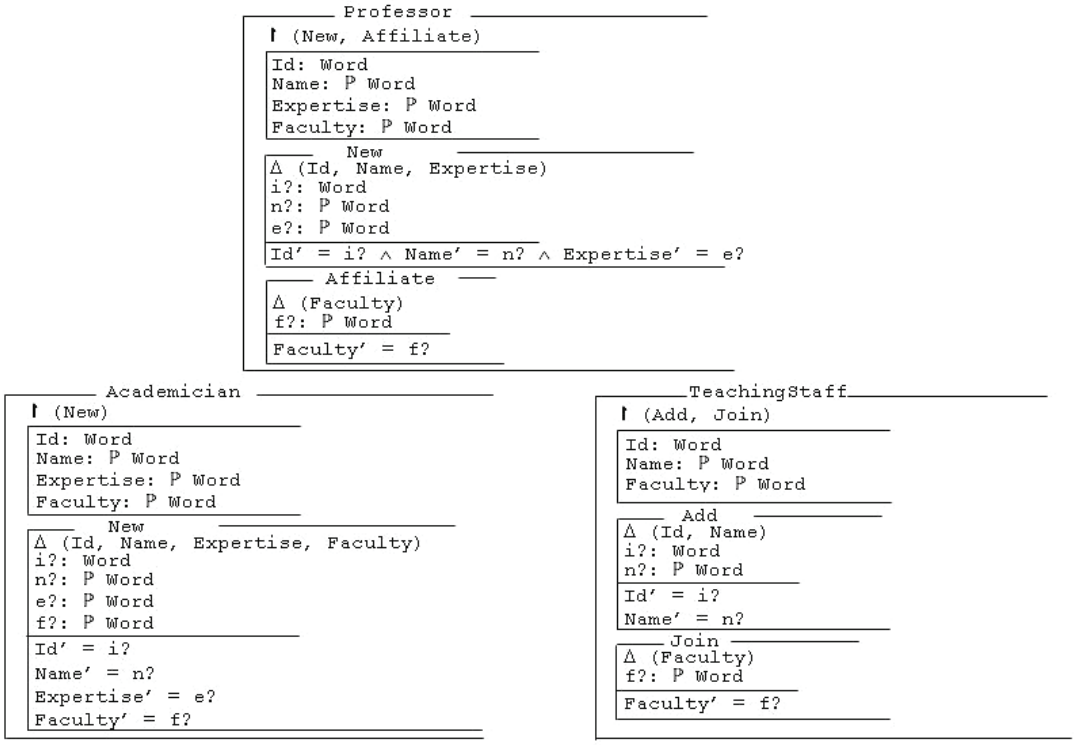



**Figure 2. Three versions of an Object-Z class**

modified by removing the part dealing with the deleted attribute *Expertise*.

The systematic identification of the exact differences that exist between the class *Professor* and the classes *Academician* and *TeachingStaff* respectively requires a formal definition of the change undergone by a specification. An algorithm to precisely compute this change can then be developed. **Table 1** shows the proposed operations defining a difference between any two given specifications.

In addition to the precise and accurate representation of a difference between two given specifications, the above operations could also be used to represent specifications' creation process itself. Moreover, the effect of a delta's operations can be inversed to obtain the old version of a specification. This is enabled by keeping track of old and new values (e.g. *Rename* and *Modify*), and the complementarity that exists between insertion and deletion operations, *i.e.* to revert the insertion of an element, we only need to delete it and vise versa.

The insertion of a node is concerned about four major meta-classes: *Class*, *Variable*, *Operation* and *Predicate*. In case of OO formal specifications, the *Variable* meta-class has three sub-classes. They are the *Attribute* (global and state attributes) of a class, the *Input* and the *Output* of an operation. Moreover, the *Predicate* meta- class has four sub-classes. They are *Invariant*, *Initialization*, *Precondition* and *Postcondition.*

**Table 1. Operations for differencing specifications**

| Operation | Effect |
|---|---|
| *insertNode*(*e, t*) | Inserts a new node *e* where *t* is the node's type. *t*={*Class, Variable, Operation, Predicate*}. |
| *setNodeProperty*(*e, p, v*) | Assigns for the 1st time a value *v* to the property *p* of the element *e*. |
| *insertLink*(*k*, $e_1$, $e_2$ , *t*) | Creates and inserts a new link *k* between the elements $e_1$ and $e_2$ where *t* is the link type, *t*={*aggregated_by, derived_from, associated_with, declared_in, used_by*}. |
| *deleteLink*(*k*) | Removes the link *k*. |
| *deleteAllLink*(*e*) | Removes all *links* and *references* associated with the element *e*. |
| *deleteNode*(*e*) | Removes the node *e*. |
| *Rename*(*e, oldname ,newname*) | Renames the element *e* (named *oldname*) with *newname* and updates (with *newname*) all *references* made to *e* in the specification. |
| *Modify*(*e, p,* $v_1$, $v_2$) | Modifies the content of *e* by changing the values of a set of properties *p* (excluding the name) whose values are in $v_1$ with a set of new values $v_2$. |

Renaming a specification's element requires updating all references made to it with its new name. For example, if a variable has been renamed, this name change is propagated to all elements that refer to this variable such as initialization, invariant, and pre (post) condition predicates. The same rule is applied when renaming classes and operations. Removing a specification's element requires removing its associated links, and all the references made to it as well (*deleteAllLink* operation). Furthermore, the operation *Modify* applies to both specifications' elements and links where a link's type (*p*) could be changed from $v_1$ to $v_2$. This reduces the number of operations in a delta by avoiding the removal of a link typed $v_1$ and the insertion of a link typed $v_2$. **Table 2** highlights the different attributes of the meta-classes representing specifications' elements:

Most of the attributes of **Table 2** are self-explanatory. However, there is a need to highlight the attributes *visibility* and *changes*. *Visibility* is similar to *public;* it applies to operations, some variables as well as some predicates. In case the visibility attribute is not applicable, the proposed value used is "n/a", such as in the case of inputs and outputs as well pre and post conditions. If an element needs to be visible outside the class the value "yes" is used otherwise "no" is used. The default visibility in Object-Z is "no", *i.e.* anything that needs to be visible outside the class has to be explicitly included in the visibility list. The *changes* attribute contains a set of variables that are changed by an operation. In case of a query operation, *i.e.* an operation that does not change the value(s) of the class variables it manipulates, the *changes* attribute is "empty".

Differencing is concerned about three classes of change. The insertion of elements/links, the modification of elements' contents, the modification of links' types, and the deletion of elements/links. We propose a differencing algorithm that is not based on elements' identifiers but rather on matching results. Using accurate matching results, differences between specifications can be precisely computed. Matched elements with different content are updates, matched elements with different links shows adding/removal of links and unmatched elements show adding/removal of elements.

The similarities that exist between specifications' elements are stored in a matching function:

$$\textbf{Match : ELEMENT} \times \textbf{ELEMENT} \times \text{TYPE} \rightarrow \text{R}$$

The returned value of *Match* is a real number (between 0 and 1) representing the exact similarity that exist between the two compared elements. The similarity scorings are added to *Match* if they are greater than or equal to a chosen threshold. The latter is a real number between 0 and 1 that defines the strictness of the matching process [12].

Each input specification is treated as a graph whose nodes are the specification's elements. Each link has a source and a target element as well as a type. For example in case of an operation *O* defined in a class *A*, a link is created to represent this relation. The link's source and target are *O* and *A* respectively, and its type is "declared_in" as in defined in **Table 1**. The difference between two given specifications is produced using the following algorithm. Given two specifications $S_1$ and $S_2$ representing by sets of nodes ($N_1$ and $N_2$) and sets of links ($L_1$ and $L_2$), the algorithm generates the exact set of transformation operations (*delta*) that allow obtaining $S_2$ from $S_1$. The algorithm starts by analyzing the unmatched elements of the two specifications. The unmatched elements of $S_1$ are added to *delta* as being *deleted* (lines 2-4). In this case, the nodes as well as their associated links are deleted. The unmatched elements of $S_2$ are added to *delta* as being *newly inserted* elements. Thus, all their associated properties and links are also added to *delta* (lines 5-7). The matched elements with different names are added to *delta* as renames (lines 9-11). For these elements, if they are not exact matches (*i.e.* similarity scoring < 1) then there is a possibility that their contents (other than names) have been modified, new links have been attached to them or that some of their links have been removed. The algorithm addresses this by detecting the changed properties other than names and adding them to *delta* (line 13). It also detects any new inserted links to them (line 14) and any removed links (line 15) and adds the changes to *delta*.

**Table 2. The attributes of specifications' elements**

| Element | Attributes |
|---|---|
| Class | - name: the class' name |
| Variable | - name: the variable's name<br>- data_type: is in the types supported by the formal language including class names.<br>- visibility is in {yes, no, n/a} |
| Operation | - name: the operation's name<br>- changes: is in {{some variable}, empty-set}<br>- visibility is in {yes, no} |
| Predicate | - value: the predicate content is a set of String<br>- visibility is in {yes, no, n/a} |
| Link | - type: the link's type is in {*aggregated_by, derived_from, associated_with, declared_in, used_by*} |

```
1.  delta=Ø
2.  for all nodes n in N1 that are not in the domain of Match do
3.     add "delete" operation to delta removing the node n and all the links associated with it
4.  end for
5.  for all nodes m in N2 that are not in the domain of Match do
6.   add "insert" operation to delta inserting the node m, setting its properties, and inserting the links associated with it
7.  end for
8.  for all x=(e1,e2,type) in domain of Match do
9.    if (e1.name≠e2.name) then
10.     add"rename" operation to delta renaming e1 with e2.name
11.   end if
12.   if Match(x) < 1 then
13.     select the properties (other than name) of e2 with values different from e1, add "modify" operations to delta
14.     select all links to/from e2 with no matching link to/from e1, add "insert" operations to delta
15.     select all links to/from e1 with no matching link to/from e2, add "delete" operations to delta
16.   end if
17. end for
18. return delta
```

**Figure 3. Differentiate algorithm**

Low similarity thresholds lead to a high rate of false matches while being able to detect most of the correct matches. High thresholds lead to few or no false matches while producing a high rate of missed matches. In [12], empirical results have shown that a reasonable threshold value (around 0.7) produces the most balanced results, which leads to a more precise delta calculation.

## 4. Detection and Resolution of Merge Conflicts

Detection and resolution of conflicts are treated as two separated phases. This is to allow each one of them to be fined-tuned without an influence on the other one. Conflicts detection should be systematic while conflict resolution might require some user interaction. Given two deltas ($delta_1$ and $delta_2$) and a base specification from which both originate, conflicts detection in concerned about discovering conflicts that might arise when the modifications contained in the deltas are integrated with the base specification. The goal of the approach is to produce a conflict-free delta whose operations are the unification of $delta_1$ and $delta_2$. The operations contained in this delta are then applied to the base specification to perform the merge.

### 4.1 A Formal Definition of Conflicts

The goal of merging is to combine the modifications made and preserve their intensions. Conflicts should be resolved accordingly. **Tables 3** and **4** show a classification and a formal definition of the most frequent conflicts observed and their causes. In these tables, *p* refers to a property or a list of properties, *v* ($v_i$) refers to a value or

**Table 3. Lost update conflicts**

| Conflict rule | How the confilict happens? |
|---|---|
| Rule 1: *modify-deleted-element* | $\exists$ e, p, $v_1 \neq v_2$: Modify(e, p, $v_1$,$v_2$) and deleteNode(e) *An element e is modified in one delta while it is deleted in the other one.* |
| Rule 2: *modify-deleted-link* | $\exists$ k, p="type", $v_1 \neq v_2$: Modify(k, p, $v_1$,$v_2$) and deleteLink(k) *The type of a link k is modified in one delta while it is deleted in the other one.* |
| Rule 3: *rename-deleted-element* | $\exists$ e, $v_1 \neq v_2$: Rename(e, $v_1$,$v_2$) and deleteNode(e) *An element e is renamed in one delta while it is deleted in the other one.* |
| Rule 4: *concurrent-update* | $\exists$ e, p, $v_1 \neq v_2 \neq v_3$: Modify(e, p, $v_1$,$v_2$) and Modify (e, p, $v_1$,$v_3$) *An element e undergoes different modifications in the two deltas.* |
| Rule 5: *concurrent -renaming* | $\exists$ e, $v_1 \neq v_2 \neq v_3$: Rename(e, $v_1$,$v_2$) and Rename (e, $v_1$,$v_3$) *An element e undergoes different renaming in the two deltas.* |
| Rule 6: *modify-moved-element* | $\exists$ e, p, $v_1 \neq v_2$: Modify(e, p, $v_1$,$v_2$) and Move(e) *An element e is modified in one delta while it is moved in the other one.* |
| Rule 7: *rename-moved-element* | $\exists$ e, $v_1 \neq v_2$: Rename(e, $v_1$,$v_2$) and Move(e) *An element e is renamed in one delta while it is moved in the other one.* |
| Rule 8: *concurrent-moving* | $\exists$ e, $e_1 \neq e_2 \neq e_3$: Move(e)=( $e_1$ , $e_2$) and Move(e)=($e_1$, $e_2$) *An element e undergoes different moving in the two deltas.* |
| Rule 9: *move-deleted-element* | $\exists$ e: Move(e) and deleteNode(e) *An element e is moved in one delta while it is deleted in the other one..* |

**Table 4. Structural conflicts**

| Conflict rule | How the confilict happens? |
|---|---|
| Rule 10: *modify-source-class* | $\exists$ A, $A_1$, p, $v_1 \neq v_2$, k, v: Modify(A, p, $v_1$,$v_2$) and insertLink(k, A, $A_1$, v) *An class A is modified in one delta while it is the source of a new link in the other one.* |
| Rule 11: *modify-target-class* | $\exists$ A, $A_1$, p, $v_1 \neq v_2$, k, v: Modify(A, p, $v_1$,$v_2$) and insertLink(k, A, $A_1$, v) *An class A is modified in one delta while it is the target of a new link in the other one.* |
| Rule 12: *link-without-source* | $\exists$ e, $e_1$, k, v: deleteNode(e) and insertLink(k, e, $e_1$, v) *An element e is removed in one delta while it is the source of a new link in the other one.* |
| Rule 13: *link-without-target* | $\exists$ e, $e_1$, k, v: deleteNode(e) and insertLink(k, e, $e_1$, v) *An element e is removed in one delta while it is the target of a new link in the other one.* |
| Rule 14: *double-containment* | $\exists$ k, e, $e_1$, v= "declared_in":insertLink (k, e, $e_1$, v) and ($\exists$ $k_1$:$k_1$.source=k.source $\wedge$ $k_1$.target$\neq$ k.target $\wedge$ $k_1$.type=k.type) *An element e is linked through a "declared_in" relation with an element $e_1$, in one of the deltas while it is linked through the same type of relation with another element in the base specification.* |
| Rule 15: *symmetric-link* | $\exists$ k, e, $e_1$,v in {"derived_from", "declared_in", "used_by"}: insertLink(k, e, $e_1$,v) and($\exists$ $k_1$: $k_1$.source=k.target $\wedge$ $k_1$.target=k.source $\wedge$ $k_1$.type=k.type) *An element e is linked through a "derived_from", "declared_in" or "used_by" relation with an element $e_1$, in one of the deltas while $e_1$ is linked to e through the same type of relation in the base specification.* |
| Rule 16: *cyclic-class-link* | $\exists$ A, $A_1$, k, v: insertLink(k, $A_1$, A, v) and ($\exists$ TransitiveClosure$_v$(A, $A_1$)) *An new link between two classes $A_1$ and A is inserted in one of the deltas while there a transitive closure between A and $A_1$ in the base specification.* |
| Rule 17: *redundant-link* | $\exists$ A, $A_1$, k, v: insertLink(k, A, $A_1$, v) and ($\exists$ TransitiveClosure$_v$ (A, $A_1$)) *An new link between two classes A and $A_1$ is inserted in one of the deltas while there a transitive closure between those classes in the base specification.* |
| Rule 18: *unwanted-reachability* | $\exists$ k, $k_1$, e, $e_1$, $e_2$, v: insertLink(k, e, $e_1$, v) and insertLink(k, $e_1$, $e_2$, v) *An new link between two elements e and $e_1$ is inserted in one delta while a new link of the same type is inserted in the other one conneting $e_1$ to an elememt $e_2$ leading to a transitive reachability between e and $e_2$.* |
| Rule 19: *redundant-element* | $\exists$ k, e, $e_1$, $e_2$, v= "declared_in": insertLink(k, $e_2$, e, v) and ($\exists$ $k_1$: $k_1$.source=$e_1$ $\wedge$ $k_1$.target=k.target $\wedge$ $k_1$.type=k.type $\wedge$ $e_2$ name=$e_1$.name) *An element $e_2$ is linked through a "declared_in"relation with an element e in one of the deltas while there is an elememt $e_1$, with the same name as $e_2$ that is linked with e through the same type of relation in the base specification.* |
| Rule 20 *double-definition* | $\exists$ $e_1$, $v_1 \neq v_2$:Rename($e_1$, $v_2$,$v_1$) and ($\exists$ e:e.name=$v_1$) *An element $e_1$ is renamed in one of the deltas while the name is already being used in the base specification for a different element e.* |

to a list of values, *e* ($e_i$) refers to specifications' elements, *A* ($A_i$) refers to classes, and *k* ($k_i$) refers to links between elements.

The conflicts have been classified under two categories.The first category is concerned about lost updates, which happens when the effect of the modifications made in one delta forbid those in the other one. Nine rules have been formally defined to allow the precise detection of this type of conflicts. The second category is concerned about the structural consistency of the specification obtained after integrating the modifications made in the delta(s), where eleven rules were formally defined. Structural consistency is a prerequisite to ensuring other forms of consistency such as those related to specifications' semantics. In fact, consistency checking, such as model checking in case of formal specifications, produces meaningful results only when applied to specifications that are well formed.

Several conflicts under the lost update category originate because of moving elements, thus, it is important to have a mechanism that detects moved elements based on the modifications made in a delta. A potential moved element is a *Variable*, an *Operation* or a *Class*. A *Variable* or an *Operation* is moved if a new added link $k_2$ of type "declared_in" connects it to a new class *B* and a link $k_1$ of the same type with an old class *A* is removed. A *Class* is moved if a new added link $k_2$ of type "derived_from", "aggregated_by" or "associated_with" connects it to a new class *B* and a link $k_1$ of the same type with an old class *A* is removed. An operation *Move* is used to perform the above verification; it accepts a *delta* containing a list of operations and a specification element *E* as parameters and returns an object containing the two elements representing the old and new link ends or a

"null" object if no moving has taken place. Formally, this verification can be written as:

$\exists\, k_1, k_2, E, A, B, t$ *in* {*declared_in, derived_from, aggregated_by, associated_with*},{*insertLink*($k_2$, *E, B, t*), *deleteLink*($k_1$)} $\in$ *delta*: $k_2$.*type*= $k_1$.*type* $\wedge$ $k_2$.*source*= $k_1$.*source*=*E* $\wedge$ $k_2$.*targe*$\neq k_1$.*target*

Finally, in order to detect some structural conflicts such as *cyclic-class-link and redundant-link*, an operation *TransitiveClosure*$_v$ is used. Given any two classes $A_1$ and $A_2$, the operation is used to verify the existence (*True* or *False*) of a path (of more than two links) between $A_1$ and $A_2$ whose links are all of type *v*. This can be formally written as:

$\exists\, k_i$: *i in* [1…*n*] *where n*>=2 *and* $\forall i$: $k_i$.*type*=*v*, $\exists\, B_j$:*j in* [1…*m*] *where m*=*n*-1: ($k_1$.*source*=$A_1 \wedge k_1$.*target*=$B_1 \wedge k_2$.*source*= $B_1 \wedge k_2$.*target*= $B_2 \wedge \ldots \wedge k_n$.*source*=$B_{n-1} \wedge k_n$.*target*=$A_2$)

## 4.2 Conflicts Detection

The proposed approach accepts as input a base specification ($S_{Base}$) and two deltas (*delta*$_1$ and *delta*$_2$) representing the parallel modifications made. Based on the formal definitions proposed previously, the approach generates a list *C* containing the details of all the conflicts discovered. **Figure 4** shows the algorithm used to discover these conflicts.

The elements of *delta*$_1$ and *delta*$_2$ are traversed to discover conflicting operations according to pre-defined rules (e.g. rules 1 to 20 of **Tables 3** and **4**). In case such operations are found, an object containing the indexes of the operations causing the conflict in *delta*$_1$ and *delta*$_2$, the type of conflict, and a conflict resolution (if any) is added to the conflict list *C* (line 9). Unlike lost updates, which are caused by two operations, structural conflicts may originate because of one operation only (from one of the deltas) or two operations (from both deltas). Thus, the conflict object added to *C* in this case may include only one index identifying the operation causing the conflict and a "null" value is assigned for the second index (lines 2-3 and lines 6-7). At the end of this process, the conflict list *C* will be storing the details of all discovered conflicts.

It is important to note that the most frequent structural conflicts originate mainly because of the creation of new links and modifying (including renaming) the elements of a base specification, and that it is possible to reduce the number of conflicts detected by enforcing conflict rules only to a specific high-level granularity such as the *Class* level.

## 4.3 Resolving Conflicts

A Conflict resolution is a set of transformation applied to the delta(s) so that a conflict is resolved. For example in case of a *visibility* attribute undergoes different modifications, a resolution is to keep the most restrictive one. Most often, a resolution consists of dropping or altering one of the conflicting operations. Manual conflict resolution is a tedious, error prone and time-consuming process especially for large specifications. Moreover, the interpretation of conflicts can differ from one developer to another one. Thus, there is a need for an approach to support the systematic resolution of as many conflicts as possible. Interacting with developers only when several (or no) resolutions are possible and a choice need to be made.

Let $N_x$ be a reference to every specification elements named x and $K_i$ (i=1...n) the new links inserted (if any). **Table 5** shows the deltas and the conflict list C associated with the classes of **Figure 2**.

Four lost update conflicts were discovered through the application of the proposed conflicts detection approach. The first conflict is a *concurrent-renaming* originating because of two different renaming of the class *Professor*. A resolution to this conflict consists of dropping the renaming operation of *delta*$_2$. The second conflict is a *rename*

**Input:** A base specification $S_{Base}$ and two deltas $delta_1$ and $delta_2$
**Output:** A conflict list C
1. for all operations $op_1$ in $delta_1$ {
2. if (effect of $op_1$ on $S_{Base}$ causes a conflict) then
3. C=C ∪ {New Conflict(indexOf($op_1$),0, getConflictType(),getResolution())}
4. if $delta_2 \neq \emptyset$ then {
5. for all operations $op_2$ in $delta_2$ {
6. if (effect of $op_2$ on $S_{Base}$ causes a conflict) then
7. C=C ∪ {New Conflict(O, indexOf($op_2$), getConflictType(),getResolution())}
8. if (combined effect of $op_1$ and $op_2$ on $S_{Base}$ causes a conflict) then
9. C=C ∪ {New Conflict(indexOf($op_1$), indexOf($op_2$), getConflictType(),getResolution())}
10. }
11. }
12.}
13. return C

**Figure 4. Verify algorithm**

**Table 5. A conflict list of two deltas**

| | |
|---|---|
| $delta_1$ | 0) Rename($N_{Professor}$, "Professor", "Academician")<br>1) deleteAllLink($N_{Affiliate}$)<br>2) deleteNode($N_{Affiliate}$)<br>3) Modify($N_{New}$, changes, "{Id,Name,Expertise}", "{Id,Name,Expertise,Faculty}")<br>4) insertLink($K_1$, $N_{Faculty}$ , $N_{New}$, "used_by")<br>5) insertLink($K_2$, $N_{f?}$ , $N_{New}$, "declared_in")<br>6) Modify($N_{postNew}$, value,"{Id'=i?, Name'=n?, Expertise'=e?}", "{Id'=i?, Name'=n?, Expertise'=e?, Faculty'=f?}") |
| $delta_2$ | 0) Rename($N_{Professor}$, "Professor", "TeachingStaff")<br>1) deleteAllLink($N_{Expertise}$)<br>2) deleteNode($N_{Expertise}$)<br>3) Rename($N_{New}$, "New", "Add")<br>4) Modify($N_{New}$, changes, "{Id,Name,Expertise}", "{Id,Name}")<br>5) deleteAllLink($N_{e?}$)<br>6) deleteNode($N_{e?}$)<br>7) Modify($N_{postNew}$, value, "{Id'=i?, Name'=n?, Expertise'=e?}", "{Id'=i?, Name'=n?}")<br>8) Rename($N_{Affiliate}$, "Affiliate", "Join") |
| Conflicts | $C_1$:{0,0, concurrent-renaming}<br>$C_2$:{2,8, rename-deleted-element}<br>$C_3$:{3,4, concurrent-update}<br>$C_4$:{6,7, concurrent-update} |

*-deleted-element* originating because the operation *Affiliate* has been removed in $delta_1$ while it has been renamed in $delta_2$. A resolution to this conflict could be to remove *Affiliate* (*i.e.* the renaming operation of $delta_2$ is dropped) as the task associated with it has been delegated to the operation *New* through the modifications made in $delta_1$. Clearly, this resolution requires user intervention. Another possible resolution to this conflict consists of dropping the delete operation of $delta_1$ that is causing the conflict. The third conflict is a *concurrent-update* originating because of concurrent modifications of the attribute *changes* of the operation *New*. Since the modified attribute is a set, it is possible to resolve this kind of conflicts by checking if one of the values is a subset of the other one, which is the case in this example. Dropping the update operation of $delta_2$ resolves this conflict. The last conflict is a *concurrent-update* originating because of concurrent modifications of the attribute *value* of the post-condition of the operation *New*. If a predicate is written as a conjunction of clauses, then it is possible to treat it as a set containing these clauses. Consequently, we could resolve this conflict by verifying if one of the values is a subset of the other, which is the case here. Dropping the update operation of $delta_2$ resolves this conflict.

Systematic conflicts resolutions based on operations' priorities could be applied. These priorities are chosen by users, which allow resolving a large number of conflicts. For example, in case of conflicts originating because of removals (e.g. *modify-deleted-element, modify-deleted-link, rename-deleted-element, move-deleted-element,link-without-source,* and *link-without-target*), a removal could be considered as an operation with less priority compared to an insertion, a modification or a renaming. Consequently, a resolution to all these conflicts is drop the delete operations causing them. The same heuristic could be used with other conflicts such as *rename-moved-element* and *modify-moved-element* if a moving operation is given a higher priority compared to a renaming or modification operation.

## 4.4 Merging

Merging is a direct process that consists of integrating (or applying) the changes made in the delta(s) after all conflicts have been discovered and resolved. Thus, merging is only possible when there is a resolution attached to every conflict discovered. These resolutions may require some user-interaction. Moreover, the order of operations in a delta is important to allow some changes to take place. For example, a rename operation involving an element *e* should always take place after any other modifications involving *e*. Merging consists of applying the operations contained in a unified and conflict-free delta to a specification. This delta is obtained through a process that involves combining the operations contained in two deltas while resolving the conflicts associated with them. **Figure 5** shows a possible result of merging the modifications made to the class *Professor* of **Figure 2**.

The above merge class integrates the modifications made after adopting specific conflict resolutions. The removal of the attribute *Expertise* does not satisfy the intension of the modifications made in one delta while the removal of the operation *Affiliate* does not satisfy the intension of the modifications made in the other delta. However, they have to be done to resolve the conflicts associated with these elements. Systematic conflict reso

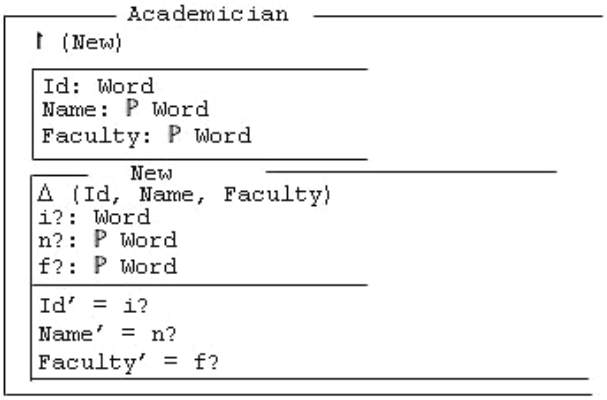



**Figure 5. The result of merging two classes**

lution saves time and effort and to some extend can preserve intensions. On the other hand, manual resolution provide a better platform for preserving intensions but they are time (and effort) consuming and are based on particular interpretations, which may lead to new inconsistencies in the merge results. Techniques to minimize conflicts could be used based on the idea of restricting the type of modifications a particular group of developers can make. Such as restricting modifications to addition only, creation of links only, etc. Thus, leading either to a reduction in the number of conflicts or to classes of conflicts that can be resolved easily and systematically.

## 5. Empirical Evaluation

A Java tool has been developed to evaluate and validate the proposed approach. It incorporates three major components. The first component processes a given OO formal specification and parses it into a graph. Currently, only Object-Z specifications are supported. The second component differentiates between two specifications represented as graphs after computing their similarities to produce a set of operations representing their delta. The third component integrates the edit operations contained in a unified delta to a base specification. This unified delta is obtained after combining the operations contained in two deltas according to the resolutions of the detected conflicts.

Merging is concerned about obtaining a consistent specification after the modifications contained in the delta(s) are integrated. Thus, it is critical to be able to detect and resolve as many conflicts as possible. Conflicts detection is dependent on the differencing algorithm used and the later is dependent on the accuracy of the similarity detection approach adopted. The proposed approach has been tested with three major case studies. They include a *university management system*, a *hotel management system* and an *online purchase system*. **Table 6** summarizes the details of the base version ($V$) of each specification as well as the modifications made to it ($V_1$ and $V_2$) and the actual number of conflicts arising because of the modifications made to the base specifications. Two different domain experts were in charge of modifying the base specifications according to requirements they think should be taken into consideration. The modifications made produced the specifications $V_1$ and $V_2$ respectively.

The combined base specifications contain a total number of 247 elements and links. The first versions of the specifications were obtained after performing 67 delta operations and the second versions were obtained through 82 delta operations made to the base specifications respectively. These modifications led to a total of 58 different conflicts.

The conflict detection and resolution approach was validated through the number of correct conflicts detected (positives), the number of all conflicts detected (positives and false positive) and the actual number of conflict that arise as a result of the modifications made (58 in the experiments made). Precision and recall metrics were used in the evaluation. Precision measures quality and is the ratio of the number of correct conflicts detected and resolved to the total number of conflicts detected. Recall measures coverage and is the ratio of the correct conflicts detected and resolved to the total number of correct conflicts. **Figure 6** shows the results obtained based on similarity threshold ranging from 0.5 to 0.9.

Perfect recall (100%) combined with a good precision (87%-98%) were obtained for thresholds raging from 0.5 to 0.7. Moreover, perfect recall (100%) combined with perfect precision (100%) were obtained for threshold ranging from 0.75 to 0.8. Furthermore, average to good recall (53%-86%) combined with perfect precision (100%) were obtained for thresholds ranging from 0.85 to 0.9. Out of 58 actual conflicts, only 8 conflicts were not detected for a high threshold of 0.85. These unidentified conflicts originate because of *too many* concurrent modifications and moving made to two classes namely *transactionInfo* and *shoppingCart* (and their elements), which led to many *concurrent-update* (4 cases), *concurrent-renaming* (2 case), *modify-moved-element* (1 case), and *concurrent-moving* (1 case) conflicts not being detected. For a threshold equals to 0.9, only 31 conflicts

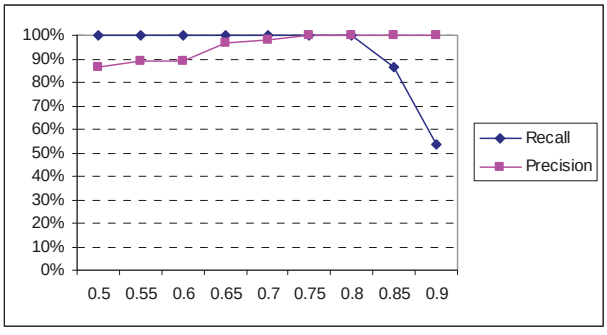



**Figure 6. Experimental results**

**Table 6. Details of the experiments**

| | V | | $V_1$ | | | $V_2$ | | |
|---|---|---|---|---|---|---|---|---|
| | #Element | #Link | #Insertion | #Deletion | #Modification | #Insertion | #Deletion | #Modification |
| Case 1 | 28 | 27 | 13 | 0 | 3 | 15 | 2 | 7 |
| Case 2 | 32 | 29 | 0 | 6 | 10 | 22 | 1 | 6 |
| Case 3 | 71 | 60 | 4 | 22 | 9 | 4 | 22 | 3 |
| Total | 247 | | 67 | | | 82 | | |
| #Conflicts | - | | | | 58 | | | |

were identified out of the actual 58 cases, which is indicated by the sharp drop of the recall. Consequently, if a high rate of positives is preferred while tolerating some negatives, a threshold value of 0.7 provides the best results.

The employed approach scales up well in terms of efficiency (performance and memory usage) as the size of the specifications increases. This is due to three main reasons. Firstly, the approach used to detect the similarities between specifications has an acceptable complexity bounded by $O(nm)$ where n and m are the number of elements of the specifications. In addition, only compatible elements are compared, *i.e.* classes with classes, variables with variables, etc. Thus, the actual number of comparison is far smaller than $n*m$. Secondly, during delta calculation only the difference between the specifications is calculated and stored, which leads to a more efficient memory usage. Finally, the proposed approach is operation-based which leads to a better performance because conflicts detection compares the operations contained in the deltas rather than comparing the input specifications themselves. Knowing that the number of operations a delta can have is statistically smaller than the number of specifications' elements.

## 6. Related Work

In [13], an approach is proposed to detect the changes to XML documents. The proposed approach uses a delta that includes insertions, deletions and updates. Moving and renaming were not considered in the delta definition, thus, leading to the detection of conflicts that originate only because of deletions and updates. Moreover, the proposed approach is based on tree representation of the analyzed documents, which restricts its applicability to documents that can be represented as trees.

In [14], model merging was used to check the structural consistency of homogenous conceptual models described as graphs. The proposed approach constructs a merge model using given mapping information that equates the correspondences between the elements of the two graphs to be merged. Then, verifies it against some consistency constraints of interest. Consistency checking rules were described using the Relational Manipulation Language (RML). The consistency diagnostics obtained over the merge are projected back to the original models and their relationships. Lost update conflicts were not considered as structural conflicts were the main focus of the work. The presented work did not include experimental data on the rate of the discovered inconsistencies in terms of precision and recall.

In [15], structural and methodological model inconsistency is verified. The proposed approach does not support model merging but rather defines a model as a set of elementary construction operations, and consistency rules are defined and verified based on the type of operation involved and the effect an operation has on the model. Inconsistency rules were translated to Prolog queries and model construction operations to Prolog facts. The approach is tool supported were an XMI file containing a model is parsed into a sequence of model construction operations then a check engine is responsible of detecting elementary operations violating consistency rules within the sequence. The validation process used employed large UML models, and the results provided are mainly about time factors, *i.e.* efficiency and scalability.

In [16], a differencing algorithm is proposed to detect the structural changes between the designs of subsequent versions of OO software. The algorithm reports the differences between them in terms of additions / removals, moves, and renaming of program elements such as packages and classes. The differencing algorithm computes an overall similarity based on name and structure similarity metrics. The proposed algorithm assumes that enough design entities remain the "same" between the two consecutive versions of the system. The latter assumption is weak as there is no guarantee that the developers of the new version of the system do not make too many modifications. The experimental results obtained reported limitations in detecting moved fields and methods. Moreover, any mistakenly identified renaming or moving of an entity is propagated to the class or the interface that contains it, and the latter will be reported as changed as well.

In [17] an algorithm is proposed to detect changes in XML documents. As a mean to improve change results, unordered tree representation of the analyzed models

were used. The matching part of the approach uses nodes signatures and prevents matching child nodes with different ancestors. This restriction affects the change detection by limiting the recognition of moved nodes. The experimental results obtained showed a slow running time while improving the accuracy of the change detection compared to the algorithm proposed in [18]. Finally, similar differencing algorithms were proposed in [19-21] to deal with different kind of software documents.

## 7. Conclusions and Future Work

An approach is proposed in this paper to detect and resolve conflicts that may occur when merging OO formal specifications. The differences between specifications are precisely calculated before a merge could take place. The difference is defined using primitive operations, acting on one element at a time, and containing traceability information enabling the reversal of their effects. The proposed approach deals with two major groups of conflicts: lost update and structural conflicts. Conflicts have been classified and formally defined as rules and a systematic approach is used to verify the calculated differences against these rules to detect any conflicts that may occur when integrating the changes made to a base specification. For every identified conflict a resolution is either derived systematically (pre-defined resolutions), or through user interaction (e.g. choosing among possible resolutions, etc). The experimental results obtained have validated the correctness and efficiency of the proposed approach as the majority of the conflicts contained in the studied specifications were systematically and cheaply (time and space) discovered. As an improvement to the proposed approach, optimizing the delta calculation and providing means to compress its content could be explored as it leads to a better efficiency. Finally, it is important to run more experiments using larger specifications.

## REFERENCES


[1] P. Sriplakich, X. Blanc and M. P. Gervais, "Supporting Collaborative Development in an Open MDA Environment," *Proceedings of* 22*nd IEEE International Conference on Software Maintenance*, Los Alamitos, 2006, pp. 244-253.

[2] A. Boronat, J. A. Carsi, I. Ramos and P. Letelier, "Formal Model Merging Applied to Class Diagram Integration," *Electronic Notes in Theoretical Computer Science*, Vol. 166, No. 1, 2007, pp. 5-26.

[3] C. L. Ignat and M. C. Norrie, "Flexible Definition and Resolution of Conflicts through Multi-level Editing," *Proceedings of IEEE* 2*nd International Conference on Collaborative Computing*, Atlanta, 2006, pp. 10-19.

[4] G. Kontonya and I. Sommerville, "Requirements Engineering Process and Techniques," John Wiley, UK, 2002.

[5] M. G. Hinchey, "Industrial-Strength Formal Methods in Practice," Springer, 2008.

[6] G. Smith, "The Object-Z Specification Language," Kluwer Academic Publishers, 2000.

[7] A. Mehra, J. Grundy and J. A Hosking, "Generic Approach to Supporting Diagram Differencing and Merging for Collaborative Design," *Proceedings of ACM/IEEE International Conference on Automated Software Engineering*, Long Beach, 2005, pp 204-213.

[8] S. Fortsch and B. Westfechtel, "Differencing and Merging of Software Diagrams-State of the Art and Challenges," *Proceedings of International Conference on Software and Data Technologies*, Barcelona, 2007, pp. 90-99.

[9] K. Fogel and M. Bar, "Open Source Development with CVS," 3rd Edition, Paraglyph Press, 2003.

[10] D. E. Perry, H. P. Siy and L. G. Votta, "Parallel Changes in Large Scale Software Development: An Observational Case Study," *Proceedings of International Conference on Software Engineering*, Kyoto, 1998, pp. 251-260.

[11] T. A Mens, "State of the Art Survey on Software Merging," *IEEE Transactions on Software Engineering*, Vol. 28, No. 5, 2002, pp. 449-462.

[12] F. Taibi, F. M. Abbou and M. D. Alam, "A Matching Approach for Object-Oriented Formal Specifications," *Journal of Object Technology*, Vol. 7, No. 8, 2008, pp. 139-153.

[13] S. Ronnau, C. Pauli and U. M. Borghoff, "Merging Changes in XML Documents Using Reliable Context Fingerprints," *Proceedings of* 8*th ACM symposium on Document Engineering*, Sao Paulo, 2008, pp. 52-61.

[14] M. Sabetzadeh, S. Nejati, S. Liaskos, S. Easterbrook and M. Chechik, "Consistency Checking of Conceptual Models Via Model Merging," *Proceedings of* 15*th IEEE International Requirements Engineering Conference*, New Delhi, 2007, pp. 221-230.

[15] X. Blanc, I. Mounier, A. Mougenot and T. Mens, "Detecting Model Inconsistency through Operation-Based Model Construction," *Proceedings of International Conference on Software Engineering*, Leipzig, 2008, pp. 511-519.

[16] Z. Xing and E. Stroulia, "Differencing logical UML models," *Journal of Automated Software Engineering*, Vol. 14, No. 2, 2007, pp.215-259.

[17] Y. Wang, "X-Diff: An Efficient Change Detection Algorithm for XML Documents," *Proceeding of* 19*th International Conference on Data Engineering*, Bangalore, 2003, pp. 519-530.

[18] A. Marian, "Detecting Changes in XML Documents," *Proceedings of* 18*th International Conference on Data Engineering*, San Jose, 2002, pp. 41-52.

[19] P. Apiwattanapong, N. Orso and M. J. Harrold, "A Differencing Algorithm for Object-Oriented Programs," *Proceedings of* 19*th International Conference on Automated Software Engineering*, Linz, 2007, pp. 2-13.

[20] U. Kelter, J. Wehren and J. Niere, "A Generic Difference Algorithm for UML Models," *Proceedings of Software Engineering Conference*, Brisbane, 2005, pp. 105-116.

[21] T. Oda and M. Saeki, "Generative Technique for Version Control Systems for Software Diagrams," *Proceedings of International Conference on Software Maintenance*, Budapest, 2005, pp. 515-524.

*J. Software Engineering & Applications*, 2010, 3: 472-476
doi:10.4236/jsea.2010.35053 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# A Novel Efficient Mode Selection Approach for H.264

**Lu Lu, Wei Zhou**

School of Computer Science & Engineering, South China University of Technology, Guangzhou, China.
Email: lul@scut.edu.cn

Received March 11th, 2010; revised April 2nd, 2010; accepted April 3rd, 2010.

## ABSTRACT

*H.264 video coding standard introduces motion estimation with multiple block sizes to achieve a considerably higher coding efficiency than other video coding algorithms. However, this comes at the greatly increased computing complexity at the encoder. In this paper, a method is proposed to eliminate some redundant coding modes that contribute very little coding gain. The simulation results show that the algorithm can remarkably decrease the complexity at the encoder while keeping satisfying coding efficiency.*

***Keywords*:** *Video Coding*, *H.264*, *Mode Selection*

## 1. Introduction

The JVT (Joint Video Team) introduced a number of advanced features in H.264 or MPEG-4 AVC. These improvements achieve significant gains in encoder and decoder performances [1-3]. One of the new features is multi-mode selection, which is the subject of this paper. In the H.264 coding algorithm, blockmatching motion estimation is an essential part of the encoder to reduce the temporal redundancy between frames. H.264 supports motion estimation and compensation using different block sizes ranging from 16 × 16 to 4 × 4 luminance samples, which is shown in **Figure 1**, with many options between the two. The luminance component of each macroblock can be split by four ways: 16 × 16, 16 × 8, 8 × 16 and 8 × 8. Each of the submacroblock partitions is called a macroblock partition. If the 8 × 8 mode is chosen, each of 8x8 macroblock partitions within the macroblock can be further split by four ways: 8 × 8, 8 × 4, 4 × 8 or 4 × 4, which are called macroblock sub-partitions. These partitions and subpartitions give rise to a lager number of possible combinations within each macroblock.[1]

H.264 standard uses computationally intensive Lagrangian rate-distortion (*RD*) optimization to choose the best block size for a macroblock. The general equation of Lagrangian *RD* optimization is given as:

$$J_{mode} = D + \lambda_{mode} \bullet R \quad (1)$$

where $J_{mode}$ is the rate-distortion cost (*RD* cost) and $J_{mode}$ is the Lagrangian multiplier; *D* is the distortion measurement between original macroblock and reconstructed macroblock located in the previous coded frame, and *R* reflects the number of bits associated with choosing the mode and macroblock quantizer value, Qp, including the bits for the macroblock header, the motion vector(s) and all the DCT residue blocks [4,5].

The computational complexity required by motion estimation, however, increases linearly with the number of used block types because block matching needs to be performed for each of them. In JVT reference software JM75C[6], it adopts full search method for each block type and selects the optimal block type as the final coding mode based on the *RD* cost function. Though it provides the best coding efficiency, the computational complexity is obviously much too high. In order to reduce the intensive computational requirement, Andy Cbang etc. proposed fast multi-block motion estimation [7]. They adopt an approach of early termination by skipping searching for mode 16 × 8 and mode 8 × 16, if the performance of mode 16 × 16 is "good enough", otherwise all coding modes will be performed. This method only considers three coding modes which are 16 × 16, 16 × 8 and 8 × 16 inter coding modes. Another approach, proposed by Andy C. Yu, is based on estimating block detail complexity [8]. It is an effective way judging by his simulation results, but there is more a critical factor, texture direction, which he does not think about but also can be useful to significantly improve coding efficiency.

In this paper, we propose a effective method to eliminate some redundant coding modes in mode selection.

[1]This paper is supported by Guangdong Technology Project (2009B010800048) and Guangzhou Technology Major Project.

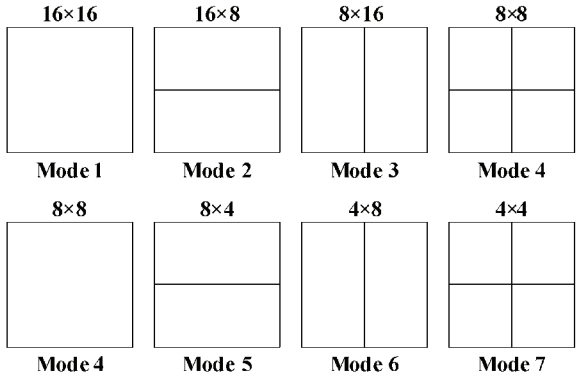



**Figure 1. Inter-prediction modes**

The paper will be organized as follows. The proposed algorithm will be described in detail in Section 2. Section 3 shows the simulation and the results. Finally, a conclusion will be given in Section 4.

## 2. Proposed Algorithm

### 2.1 Block details

**Table 1** shows the observations on how selected modes relate sequence characteristics.

The choice of partition size has a significant impact on compression performance. In general, according to **Table 1**, large partition sizes are appropriate for homogeneous areas of the frame and small partition sizes may be beneficial for detailed areas.

We derive an approach based on summing the total energy of the *AC* coefficients to estimate the block detail. The *AC* coefficients can be obtained from the DCT coefficients of each block. The definition is:

$$E_{AC} = \sum_{u=1}^{M-1}\sum_{v=1}^{N-1}(F\ (u,v))^2 \tag{2}$$

From (2), $E_{AC}$, the total energy of the *AC* components of an $M \times N$ block is the sum of all the DCT coefficients, $F(u,v)$, except for the DC component, $u = 0$ and $v = 0$.

$$F(u,v) = c(u)c(v) \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\cos[\frac{(2x+1)u\pi}{16}]\cos[\frac{(2y+1)v\pi}{16}] \tag{3}$$

where,

$$c(u), c(v) = \begin{cases} \sqrt{\frac{1}{M}}, \sqrt{\frac{1}{N}} & for\ u,v=0 \\ \sqrt{\frac{2}{M}}, \sqrt{\frac{2}{N}} & for\ u,v\neq 0 \end{cases} \tag{4}$$

According to the energy conservation principle, the total energy of an $M \times N$ block is equal to the accumulated energy of its DCT coefficients. Thus, (3) can be further simplified as

$$E_{AC} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f^2(x,y)) - \frac{1}{MN}[\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)]^2 \tag{5}$$

where the first term is the total energy of the image intensities within an $M \times N$ block, and the second term represents the mean square intensity. Equation (5) clearly shows that the energy of the *AC* components of a macroblock can be represented by the variance.

Evaluating the maximum sum of the *AC* components is the next target. By definition, the largest variance is obtained from the block comprising checkerboard pattern in which every adjacent pixel is the permissible maximum and minimum value. Thus, $E_{\max}$, the maximum sum of *AC* components of an M×N block is

$$E_{\max} = MN\frac{f_{\max}^2(x,y) + f_{\min}^2(x,y)}{2} - \frac{MN}{4}[f_{\max}(x,y) + f_{\min}(x,y)]^2 \tag{6}$$

Note that $E_{\max}$ can be calculated in advance. Then the criterion to assess the complexity of a macroblock detail is

$$r_d = \frac{\ln(E_{AC})}{\ln(E_{\max})} \tag{7}$$

In total, 7 different block sizes are recommended by H.264 for P-frames, namely, 16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8, 4 × 4 as well as SKIP, and other two INTRA prediction modes, I4MB and I16MB. However, in our complexity measurement, there are only 3 categories, which are denoted as MD16 category, MD8 category, and MD4 category, respectively.

The proposed algorithm provides a recursive way to decide the complexity of each macroblock. Firstly, a macroblock of 16 × 16 pixels is examined with the first

**Table 1. Selected modes for different sequences**

| Sequence | Skip | 16×16 | 16×8 | 8×16 | 8×8 | Intra16 | Intra4 |
|---|---|---|---|---|---|---|---|
| **Container** | 75.8 | 10.4 | 3.5 | 2.7 | 7.3 | 0.3 | 0.0 |
| **Foreman** | 23.7 | 39.9 | 39.9 | 7.3 | 7.6 | 7.3 | 9.3 |
| **Bus** | 3.5 | 22.0 | 12.1 | 14.4 | 40.5 | 1.0 | 5.5 |
| **Mobile** | 4.5 | 31.3 | 7.1 | 6.1 | 6.1 | 49.7 | 0.3 |

***IPPP, 5 reference frames, CABAC, CIF Format***

piecewise equation in (7). An LDB category is given if it is recognized as being a homogenous macroblock. Otherwise, the macroblock is decomposed into 4 blocks of 8 × 8 pixels. Note that an 8 × 8 block is recognized as high-detailed if it satisfies two conditions: 1) the RB in (7) is greater than 0.7, and it is decomposed into four 4 × 4 block, and 2) one of its four decomposed 4 × 4 blocks is highdetailed as well. If an 8 × 8 block satisfies the first condition but not the second, it is still recognized as low-detailed. After checking all the 8 × 8 blocks, an MDB category is given to a macroblock which possesses more than two high-detailed blocks, otherwise the HDB category is assigned. **Table 2** displays the relationship between the three categories in the proposed algorithm and the 9 inter-frame prediction modes. It is observed that the LDB category covers the least number of prediction modes, whereas the HDB category contains all the available modes. The table further indicates that the higher detailed the macroblocks are, the more prediction modes the proposed algorithm has to check.

The function of the natural logarithm is to linearize both $E_{\max}$ and $E_{AC}$ such that the range of $r_d$ can be uniformly split into 10 subgroups. In our evaluation, a macroblock that has the $r_d$ >0.7, is considered to be a high-detailed block.

## 2.2 Object Movement

More than one object is contained in a macroblock and is moving in different directions. This included objects moving over a background with different velocity. For example, in **Figure 2** the object is moving against a static background. In this case, the current block should be divided into two 8 × 16 sub-blocks whereas sub-block 0 should have a zero motion vector and sub-block 1 should have a motion vector such that the cost function can be minimized.

## 2.3 Texture Regions

When the edge of texture aligned perfectly with the sensor boundaries at a particular time instant, the texture edge is clear and sharp. We will describe this texture as having "integer-pixel location". When the texture undergoes an integer-pixel translational motion, the texture will look exactly the same in the two consecutive frames except that one is a translation to another. And the moved texture can be predicted perfectly by integer-pixel motion estimation.

If the edges of texture have a half-pixel offset relative to the senor, the edges may be blurred as shown in **Figure 3(b)** and said to have "half-pixel location". The original zero-pixel-wide (sharp) edge now becomes one pixel-wide (blurred). The pixel at the blurred edges may have only half the intensity of the original one, which can lead to difficulty in motion estimation.

Similarly, if the edges have a quarter-pixel offset it may be blurred as shown in **Figure 3(c)**. We will describe this texture as having "quarter-pixel location". The zero-pixel-wide (sharp) object edge becomes one pixel-wide (blurred). The pixels at the blurred edges may have 3/4 or 1/4 of the intensity. The use of sub-pixel motion estimation algorithm, like half-pixel or quarter-pixel estimation, uses interpolation to predict the sub-pixel shift of texture relative to the sampling grid.

Different type of texture (integer, half or quarter) has a different response to fractional motion estimation. For example, the texture in **Figure 3(b)** (half) can be predicted perfectly by the texture in **Figure 3(a)** (integer) using half-pixel motion estimation but not vice verse. Since it is possible for a macroblock to contain more than one kind of texture, using only one integer, half or quarter pixel motion vector will not be sufficient to describe the texture content. For example, a macroblock may contain two 8 × 16 sub-blocks where sub-block 0 contains "half-pixel" texture and sub-block 1 contains "integer-pixel" texture. In this case, the current macroblock should be divided into two 8 × 16 sub-blocks in which half-pixel motion vector should be used for sub-block 0 and integer-pixel motion vector for sub-block 1.

**Table 2. Block categories and corresponding modes**

| Detail Level | Enabled Modes |
|---|---|
| LDB | 16×16 |
| MDB | 16×16, 16×8, 8×16, 8×8 |
| HDB | 8×8, 8×4, 4×8, 4×4 |

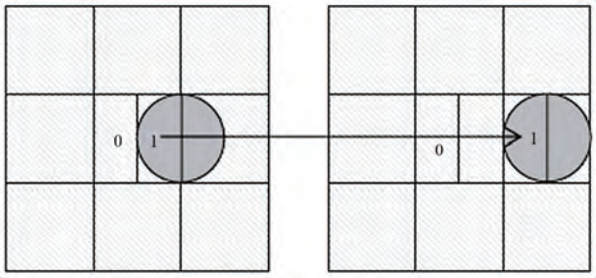



**Figure 2. Example of an object moving on a static background**

## 2.4 Algorithm

Former results [9] show that, often, about 70% of the macroblocks will choose mode 1 (16 × 16) as their final block type. In the proposed algorithm it determines the macroblock detail-level and analysis the information obtained from 8 × 8 block size ME to predict the mode 1 macroblock in advance, if possible, the optimal motion vector. If the macroblock is predicted to be mode 1 macroblock, searching will be stopped immediately.

As a result, computation can be saved for mode 2 and mode 3 block size ME and in some situation mode 1 as well. Three decisions are set up in handling different

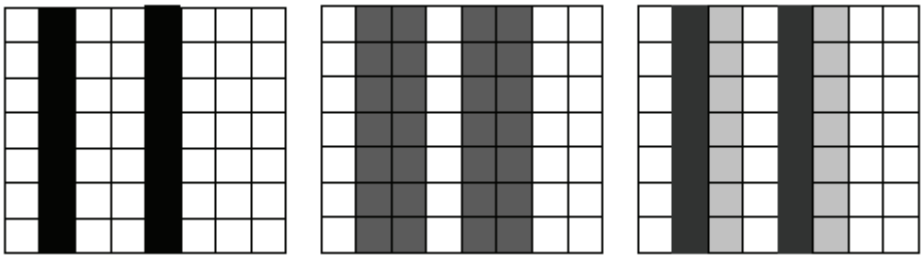


**Figure 3. (a) Integer-pixel texture; (b) Half-pixel texture; (c) Quarter-pixel texture**

video area-general area, slow moving area and fast moving area.

**Step1**: If $r_d$<0.3 then

- Select 16 × 16 as the only enabled mode (LDB)

Else if 0.3<$r_d$<0.7 then

- Disable 8 × 8, 4 × 8, 4 × 4 (MDB)

Else if If $r_d$>0.7 then

- Enable all of the modes (HDB)

Defining MV0, MV1, MV2, MV3 be the motion vector of 8 × 8 subblock of the current macroblock. Two conditions are checked:

**Step2**:

**C1**: If MV0=MV1=MV2=MV3 then

- choose mode 1 (16 × 16) as final block type
- no ME will be further performed
- 16 × 16 MV = 8 × 8 MV0

**C2**: If three subblock MV are the same AND

the forth unequal MV only differ by one quarter pixel (1/4)

distance

then

- choose mode 1 (16 × 16) as final block type
- no ME will be further performed
- 16 × 16 MV = dominated 8 × 8 MV

**C3**: If collocate MB in previous frame is mode1 AND {MV0, MV1, MV2, MV3} < 4 (*i.e.* one integer pixel

distance) AND MV0, MV1, MV2, MV3 has the same direction then

- choose mode 1 (16 × 16) as final block type
- 8 point local search around MV = {0, 0}

**C4**: If all magnitude of 8 × 8 MVx >= 3 integer distance OR all magnitude of 8 × 8 MVy >= 3 integer distance

- choose mode 1 (16 × 16) as final block type
- local search for surrounding 24 points of MV0

The reason for all the 8 × 8 motion vector having the same direction in decision C3 can be illustrated using **Figure 4(a)**. Suppose a macroblock is undergoing small rotational motion as shown in **Figure 4(a)**. The motion vector at the left size of macroblock will be downward and the right side will be upward. As a result, there is a high potential for the current macroblock segmented vertically even the magnitude of motion vector is very small.

**Figure 5** shows the performance of C1 + C2 + C3 + C4. We can see the hit rate increase for the fast panning part of foreman sequence which is much closer to the optimal one.

## 3. Simulation Results

The proposed algorithm was implemented in the reference JVT software.JM75C. We have tested our proposed method over a series of testing sequence with different resolution.

In this paper, two QCIF (176 × 144) sequences, "Foreman" and "Stefan" are selected to show the result. In the simulation, the sequences are encoded at 30 fps with qp = 10 to 20 with step size of two. The PSNR and bitrate comparison between proposed algorithm and full search is shown in **Table 3**.

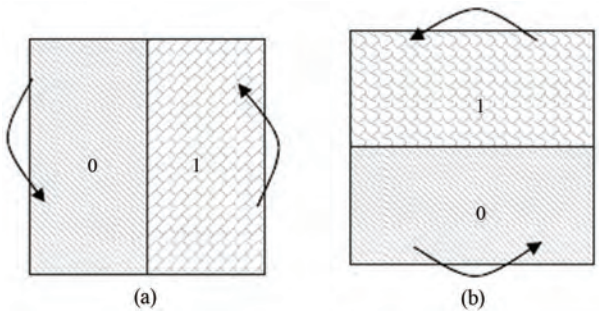



**Figure 4. Example of rotational motion in Macroblock that cause segmentation (a) vertical segmentation; (b) horizontal segmentation**

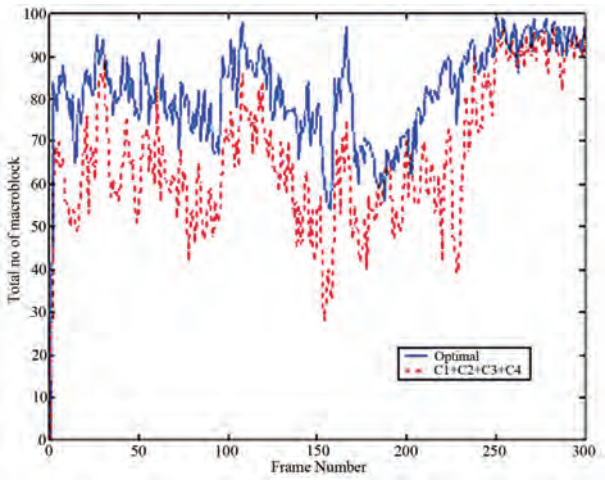



**Figure 5. Performance using Decision C1 + C2 + C3 +C4 using foreman QCIF sequence**

**Table 3. PSNR and Bitrate Comparison between the proposed algorithm and FS with QP = 10 to 20; (a) Stefan QCIF; (b) Foreman QCIF**

| Stefan QCIF | | | | | | | ForemanQCIF | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FMFME | | Full Search | | | | | FMFME | | Full Search | | | |
| QP | Psnr (dB) | BR (kbits) | Psnr (dB) | BR (kbits) | Gain (dB) | BR Gain | QP | Psnr (dB) | BR (kbits) | Psnr (dB) | BR (kbits) | Gain (dB) | BR Gain |
| 10 | 49.48 | 2602.9 | 49.48 | 2600.0 | 0 | -0.11% | 10 | 49.69 | 1457.1 | 49.69 | 1455.0 | 0 | -0.15% |
| 12 | 47.64 | 2203.3 | 47.64 | 2201.3 | 0 | -0.09% | 12 | 47.97 | 1149.2 | 47.97 | 1146.8 | 0 | -0.21% |
| 14 | 46.13 | 1891.5 | 46.14 | 1889.5 | -0.01 | -0.10% | 14 | 46.5 | 923.26 | 46.5 | 921.76 | 0 | -0.16% |
| 16 | 44.48 | 1611.5 | 44.48 | 1608.4 | 0 | -0.19% | 16 | 44.89 | 732.14 | 44.9 | 729.57 | -0.01 | -0.35% |
| 18 | 42.58 | 1323.5 | 42.58 | 1321.5 | 0 | -0.15% | 18 | 43.11 | 552.33 | 43.12 | 550.69 | -0.01 | -0.30% |
| 20 | 40.89 | 1095.2 | 40.89 | 1092.2 | 0 | -0.27% | 20 | 41.52 | 422.31 | 41.53 | 419.7 | -0.01 | -0.62% |
| 22 | 39.3 | 903.06 | 39.3 | 900.83 | 0 | -0.25% | 22 | 40.03 | 328.54 | 40.03 | 326.17 | 0 | -0.73% |
| 24 | 37.36 | 707.5 | 37.36 | 705.61 | 0 | -0.27% | 24 | 38.32 | 241.99 | 38.33 | 238.92 | -0.01 | -1.28% |
| 26 | 35.65 | 557.72 | 35.66 | 555.8 | -0.01 | -0.35% | 26 | 36.83 | 180.03 | 36.85 | 178.54 | -0.02 | -0.83% |
| 28 | 33.95 | 432.34 | 33.96 | 430.4 | -0.01 | -0.45% | 28 | 35.48 | 136.92 | 35.49 | 135.35 | -0.01 | -1.16% |
| 30 | 32.06 | 322.08 | 32.07 | 320.79 | -0.01 | -0.40% | 30 | 33.99 | 103.02 | 34 | 101.24 | -0.01 | -1.76% |
| 32 | 30.34 | 238.65 | 30.34 | 236.84 | 0 | -0.76% | 32 | 32.57 | 77.65 | 32.58 | 76.58 | -0.01 | -1.40% |
| 34 | 28.79 | 177.98 | 28.8 | 177.61 | -0.01 | -0.21% | 34 | 31.3 | 60.67 | 31.34 | 59.62 | -0.04 | -1.76% |
| 36 | 27.13 | 127.37 | 27.13 | 127.04 | 0 | -0.26% | 36 | 29.96 | 45.86 | 30.03 | 45.43 | -0.07 | -0.95% |
| 38 | 25.66 | 94.1 | 25.68 | 93.35 | -0.02 | -0.80% | 38 | 28.63 | 35.55 | 28.71 | 35.47 | -0.08 | -0.23% |
| 40 | 24.33 | 70.58 | 24.36 | 70.35 | -0.03 | -0.50% | 40 | 27.49 | 28.63 | 27.53 | 28.3 | -0.04 | -1.17% |
| | | | | Average | -0.00625 | -0.32% | | | | | Average | -0.02 | -0.82% |

(a) (b)

The complexity is shown in **Table 3**. The proposed algorithm can reduce computational cost by 58% on average (equivalent complexity of performing motion estimation on 1.7 block types instead of 4 block types) with negligibly small PSNR degradation (0.013dB) and slight-increase in bit rate (0.57%).

## 4. Conclusions

In this paper, we propose a method to eliminate some redundant coding modes, which speeds up the process of multi-mode selection. The simulation results show that the algorithm can remarkably decrease the complexity at the encoder while keeping satisfying coding efficiency.

## REFERENCES

[1] "Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG: Draft Text of Final Draft International Standard for Advanced Video Coding," *H.* 264|*ISO/IEC* 14496-10 *AVC*, ITU-T.

[2] M. Ghanbari, "Standard Codecs: Image Compression to Advanced Video Coding," IEE Publishing, 2002.

[3] E. G. Iain and Richardson, "H.264 and MPEG-4 Video Compression," Wiley, 2003.

[4] F. S. Yan, "Fast mode selection based on texture analysis and local motion activity in H.264/AVC," 2004 *International Conference of Communications, Circuits and Systems*, Chengdu, Vol. 1, 27-29 June 2004, pp. 539-542.

[5] G. W. Teng, Z. Y. Zhang, Y. J. Zhang and W. J. Zhang, "Fast Mode Decision Algorithm in Inter Pictures Based on H. 264/ AVC," *Journal of Optoelectronics·Laser*, Vol. 16, No. 7, July 2005, pp. 866-870.

[6] "JVT Reference Software JM75C". http://bs.hhi.de/~suehring/tm

[7] A. Chang, O. C. Au and Y. M. Yeung, "A Novel Approach to Fast Multi-block Motion Estimation for H.264 Video Coding," *Proceedings* 2003 *International Conference on Multimedia and Expo*, Maryland, Vol. 1, 6-9 July 2003, pp. 539-542.

[8] A. C. Yu, "Efficient Block-size Selection Algorithm for Inter-Frame Coding in H.264/MPEG-4 AVC," 2004 *IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, Vol. 3, 17-21 May 2004, pp.69-72.

[9] Y. S. Cui, D. G. Duan and Z. L. Deng, "Fast Motion Estimation Algorithm on H.264," *Journal of Liaoning Institute of Technology*, Vol. 24, No. 5, October 2004, pp. 12-15.

*J. Software Engineering & Applications*, 2010, 3: 477-486
doi:10.4236/jsea.2010.35054 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Test Cost Optimization Using Tabu Search

**Anu Sharma*, Arpita Jadhav, Praveen Ranjan Srivastava, Renu Goyal**

Computer Science and Information System Group, Birla Institute of Technology and Science, Pilani, India.
Email: {*anu11sharma1123, arpitajadhav, praveenrsrivastava}@gmail.com

Received January 5th, 2010; revised February 21st, 2010; accepted February 25th, 2010.

## ABSTRACT

*In order to deliver a complete reliable software product, testing is performed. As testing phase carries on, cost of testing process increases and it directly affects the overall project cost. Many a times it happens that the actual cost becomes more than the estimated cost. Cost is considered as the most important parameter with respect to software testing, in software industry. In recent year's researchers have done a variety of work in the area of Cost optimization by using various concepts like Genetic Algorithm, simulated annealing and Automation in generation of test data etc. This paper proposes an efficient cost effective approach for optimizing the cost of testing using Tabu Search* (*TS*)*, which will provide maximum code coverage along with the concepts of Dijkstra's Algorithm which will be implemented in Aspiration criteria of Tabu Search in order to optimize the cost and generate a minimum cost path with maximum coverage.*

***Keywords*****:** *Tabu Search, Test Cost Optimization, Dijikstra's Algorithm*

## 1. Introduction

Software engineering is not just to develop new software but also that product should be more reliable and cost effective so that client can effectively use that. According to Bezier B [1], Software testing is an important factor in software development life cycle in which one-third to one-half of the total cost of the product is consumed only on the testing process. Software testing is a process to trace out the errors in software that intended to meet the desired result of the programmer by satisfying all the pre condition factors setup by the tester. Since, software testing is becoming more popular and demanding area in the software development industry in past few years [2]. Resulting product is very reliable if testing covers maximum errors. But on the contrary, during testing the cost can increase more than the expected value due to inappropriate test cases. These inappropriate test cases cause wastage of organizational resources as well as time. There is a need to minimize the cost for getting an acceptable product.

In order to deal with the above issue and also to provide good quality software within desired time and least cost Researchers have applied several techniques on minimization of cost in testing of product, such as fuzzy logic, automatically generation of test data [3]. But here we are using Tabu Search [4]. Tabu Search will choose appropriate test paths and concepts of Dijkstra algorithm will be implemented in Aspiration Criteria to optimize the cost. If any test case does not provide maximum coverage, proposed algorithm will backtrack and starts with a new path.

This paper is organized as follow:

Section 2 describes the general introduction to software testing, Section 3 shows historical detail of the tabu search in testing area, Section 4 describes about the tabu search, Section 5 shows dijkstra algorithm used in proposed approach, Section 6 is the proposed technique which uses tabu search with dijkstra algorithm, Section 7 is showing experimental illustration of the proposed algorithm Section VIII is conclusion and further work.

## 2. Software Testing

Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software [1,2]. The software testing strategy is an important process in case of software development lifecycle model [2]. Testing is a process which leads delivery of higher quality products, low cost, more accurate and reliable results of developed computer software. The purpose of testing includes quality assurance, verification and validation, or reliability estimation. If all shortest paths which can be measured by plotting the control flow graph [2], with maximum coverage are known to the tester, estimated cost taken by testing process can be reduced.

## 3. Related Work

Organizations facing the challenge of solving testing cost problems. Classical approaches often suggested solution

to the coverage problem but there is very less number of solutions that control the cost while providing the reasonable quality to the software. On tabu search research is going on rapidly but no method exists to control the cost of testing, Even if we do the automation there should be some criteria to select the appropriate test cases.

Several attempts have been made over the years to develop such an algorithm for the Optimization of the cost to find out an efficient path automatically.

Research has applied many approaches for the same purpose [5]. The basic algorithm of Tabu Search is explained in [6]. The concept of Tabu Search has also been applied to optimize the Cost of the program with maximum code coverage [6]. Previously the work has been done on the automation of test cases using Tabu search algorithm on complex programs under test and large number of input variables. Using these automation tools cost can be reduced and saves the time [3,7]. Another approach suggests the use of tabu search algorithm for generation of structural software tests [4]. It also combines the use of memory with a backtracking process to avoid getting stuck in local minima [8]. To explore the regions and to avoid the revisiting of candidate list a position guided tabu search based on metric search space has been covered. To improve the intensification (search the local optimal solution) and diversification (exploring new regions)[9] ITS based approach has been used in a research Tabu search implemented to "genetic" methods and evolutionary method, to deal with the complex path tabu search have adaptive memory structure so it can also be applied to the neural networks. Tabu Search has been also applied to solve the Job Shop Scheduling problem using Genetic Algorithms [10]. Most of the scheduling problems require either exponential time or space to generate an optimal answer. Many researchers have been done for identifying the infeasible paths of a program [11]. Different algorithms and techniques have been proposed by the researchers to detect optimized paths [12,13].Generating test data automatically and identifying infeasible paths reduces the testing cost, time and effort [14].

Since cost and code coverage play an important role in testing. But not much of work has been suggested for cost optimization with maximum code coverage. This paper provides the solution for the above problem. Which employ Tabu Search algorithm with the concept of greedy approach to provide a minimum cost path by covering most of the nodes and storing the best path solution into memory.

## 4. Tabu Search [6]

Tabu search is a metaheuristic approach which is used to solve the optimization problems, [6,15]. It is designed to guide other methods to escape the trap of local optimality, also called local minima.

Overview of Tabu Search:

Three primary themes form the basis of Tabu search:

1) Flexible attribute-based memory structure: It is designed in such a way so that the evaluation criteria as well as historical search information can be exploited more thoroughly than by rigid memory structures (as in branch bound) or by memory less systems (as in case of simulated annealing).

2) An associated mechanism of control is embodied for employing the memory structures, which are based on the interplay between conditions that constrain the search process.

3) At different time spans, the incorporation of memory functions, from short term to long term, as well as to implement strategies for intensifying and diversifying the search process to give optimal results.

a) Intensification strategies, helps in reinforcing and moving combinations and solution features historically that are found good.

b) Diversification strategies, drives the search process into new regions as to explore every possible area and region.

Distinguishable Features:

Short-term Memory and Aggressive Search:

Short-term memory constitutes a form for aggressive exploration and captures the best move possible under tabu restrictions. Tabu restrictions prevent the reversal and repetition of certain moves by rendering selected attributes covered in previous moves (forbidden). Primary goal of tabu restriction is to permit the method to go beyond the points of local optimality during its iterations. Tabu restrictions help in preventing cycles and induce the search to follow a new trajectory, in case if cycling occurs.

Tabu Restriction [15]:

These are certain conditions which are imposed on moves that make some of the moves forbidden. These forbidden moves, in-turn are listed to a certain size called as tabu. And this list is considered as "tabu list". The reason behind to denote a move as forbidden is to prevent cycling and avoiding returning to the local optimum that has been visited. In order to identify a good tabu list size, simply watch the occurrence of cycling (if it occurs) when the size of the list is too small and deteriorate the quality of solution when the size of the list is too large which is caused by forbidding too many moves. Remember, that the size of tabu list should grow with the size of the problem. Also it prevents the added edges from being dropped as well as it prevents the dropped edges from being added.

A general approach for the tabu search [3,15] is as shown in **Figure 1** and the basic elements of TS are as follows:

1) Current solution: it comprises a set of the optimized parameter values for a given iteration. It plays a crucial role in order to generate the neighbor trial solutions.

2) Moves: These are related to current solution and

characterize the process of generating the trial solutions.

3) Set of candidate moves: It comprises the set of all possible moves or may be trial solutions.

4) Aspiration Criterion: It is a rule for overriding the tabu restrictions, for example if certain move is forbidden by tabu restriction, aspiration criterion has to be satisfied, once it is satisfied, it can only make this move allowable. One we considered here is to override the tabu status of a move if this particular move yields a solution which has better objective function (let it be *J*), than the one that was obtained earlier within the same move. The phenomenon behind using aspiration criterion is to add flexibility in the tabu search by directing it towards better moves.

5) Long Term and Short Term Memory: Tabu incorporate two type of memory long term memory and short term memory. Short term memory stores more recent moves and long term memory keeps all the related moves.

6) Stopping Criterion: When best solution already reached, maximum iterations have been performed and when certain conditions are not meet.

## 5. Dijkstra Algorithm

Dijkstra's algorithm [16] is a graph search algorithm, that traverses all the nodes and it helps in providing minimum cost path from source to destination node. It is also known as greedy approach and it finds the shortest path between single source to all other nodes.

That's why sometime it is also called as single source shortest path problem. It can also be used for finding costs of shortest paths from a single source node to a single destination node by stopping the algorithm once the shortest path to the destination has been determined. For example: in case of city problem, in which the vertices of the graph represent cities and edge represents the distance between the cities.

## 6. Proposed Solution (Tabu Search with Dijkstra Algorithm)

Since cost and coverage are two important factors in case of testing. Here in our proposed algorithm we are using the Tabu Search concept to resolve both of the issues.

The algorithm as shown in the **Figure 3.** Which we have developed uses tabu search [3,6,15] with the concepts of Dijkstra's Algorithm [16] which will be implemented in Aspiration criteria of Tabu Search in order to optimize the cost and generate a minimum cost path with maximum coverage.

For fulfilling the desired purpose we are require to inspect the program thoroughly to check the aspiration criteria. For this our software will generates the control flow graph of the statement automatically. Where node represents the statement and link represents the flow of control between the statements.

Our algorithm states that in case of fulfilment of all the three conditions given in aspiration criteria will move further to next iteration otherwise system goes in backtracking stage. The flow of all related activities is as depicted in **Figure 2**.

| |
|---|
| **begin** |
| Initialise some current solution |
| Calculate the cost of current solution and store it as best cost |
| Store current solution as new solution |
| Add new solution to tabu list |
| **do** |
| Calculate neighbourhood candidates |
| Calculate the cost of candidates |
| Store the best candidate as new solution |
| Add new solution to tabu list |
| **if (the cost of new solution<best cost) then** |
| Store new solution as best solution |
| Store the cost of new solution as best cost |
| **endif** |
| Store new solution as current solution |
| **while** NOT Stop Criteria |
| **end** |

**Figure 1. Basic tabu search algorithm [3]**

## 7. General Illustration of Proposed Algorithm

In this section we have presented a simple program for the calculation of the ship charge that depends on amount, tax and rush charge. And then we have made its corresponding control flow graph. This control graph is taken as input for the software. In the graph, the nodes represent the statements and the edges represent the flow between the statements. The cost for several edges can be calculated by using Halsted's Software Science [2] or with the prior experience of developer for the application.

**Case Study:** To test the proposed approach we have applied that to the program given in **Figure 4**, and we will check the solution iteration by iteration. At the end tabu List store the best optimized path in its memory.

1) **Figure 5**. represents the intial control flow graph. Here node 'a', represents the starting node and n represents ending node. Any graph can be provided as input. This algorithm will provide minimum cost and maximum code coverage.

2) Once the algorithm starts, here node 'b' is selected to be the next node as its cost is minimum.Same criteria will be used with respect to all other nodes.

3) Here the next node will be chosen as 'c' as c is the other node with minimum cost.

4) Here the next node will be chosen as 'd'.

5) Here the next node will be chosen as 'n'.

6) In this step the algorithm will backtrack since the path through 'd' did not give the least cost path. The next node will be chosen as 'e'.

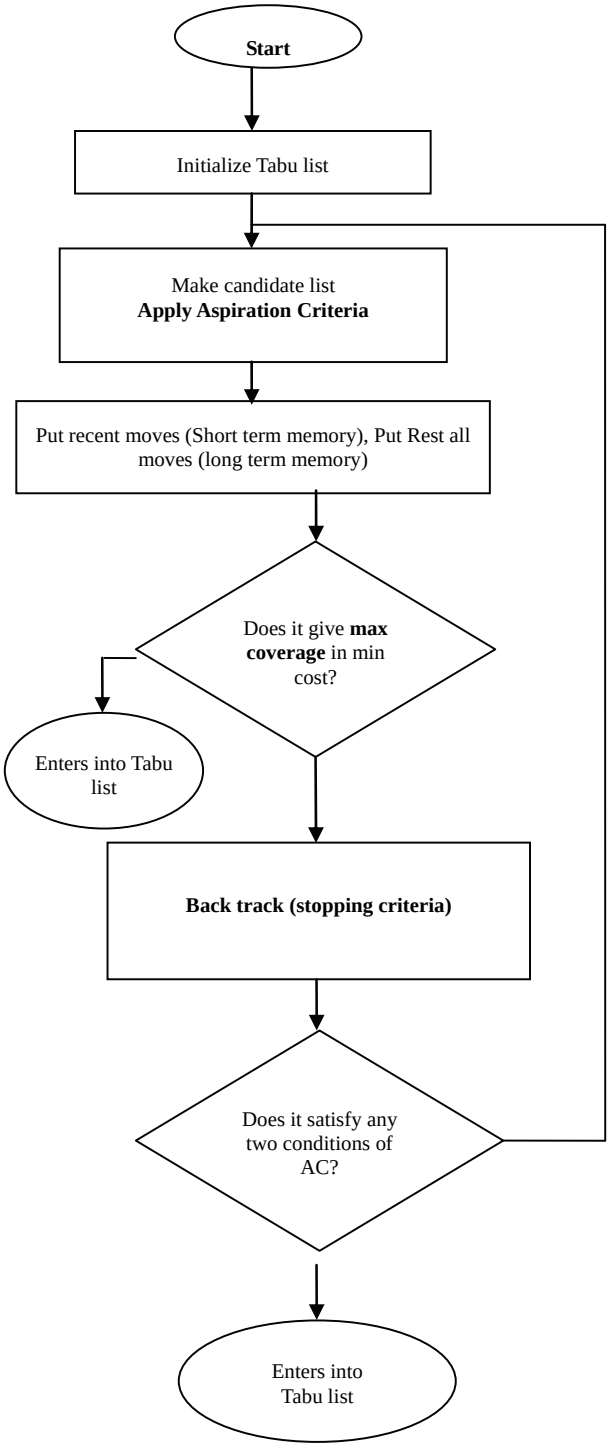



**Figure 2. Flow graph**

7) Here the next node will be chosen as ‘f’.

8) Here the next node will be chosen as ‘n’. In this step this algorithm will backtrack since the path through ‘f’ did not give the least cost path. The next node will be chosen as ‘g’.

9) Here the next node will be chosen as ‘h’.

**1**. Start algorithm
**For(N:=1; N<=candidate list C1; N++)**
**If**

**2. Aspiration criteria(Ac)(max. coverage && min. cost && reach to subgoal)**

**3.** Calculate cost

**4.** Enter in the tabu list

**Else**

**5. Backtracking beginning**
**Stopping Criteria (SC)[(max coverage && min. cost)!! (Max. coverage && reach to subgoal)**

**6.** Calculate cost

**7.** Enter in the tabu list

**End**

**Figure 3. Proposed algorithm**

```
Public double calculate(int amount)
{
Double rushcharge=0;
If(nextday.equals(“yes”)){
Rushcharge=14.50;   }
Double tax=amount*.0725;
If(amount>=1000){
Shipcharge=amount*.06+rushcharge;   }
Elseif(amount>=200)   {
Shipcharge=amount*.08+rushcharge;   }
Elseif(amount>=100)   {
Shipcharge=13.25+rushcharge;    }
Elseif(amount>=50)   {
Shipcharge=9.95+rushcharge;    }
Elseif(amount>=25)   {
Shipcharge=7.25+rushcharge;    }
Else   {
Shipcharge=5.25+rushcharge;    }
Total=amount+tax+shipcharge;
Return total;
End calculate
```

**Figure 4. Sample program**

10) Here the next node will be chosen as ‘n’. In this step the algorithm will backtrack since the path through ‘h’did not give the least cost path. The next node will be chosen as ‘i’.

11) Here the next node will be chosen as ‘j’.

12) Here the next node will be chosen as ‘n’.In this step the algorithm will backtrack since the path through ‘j’ did not give the least cost path. The next node will be chosen as ‘k’.

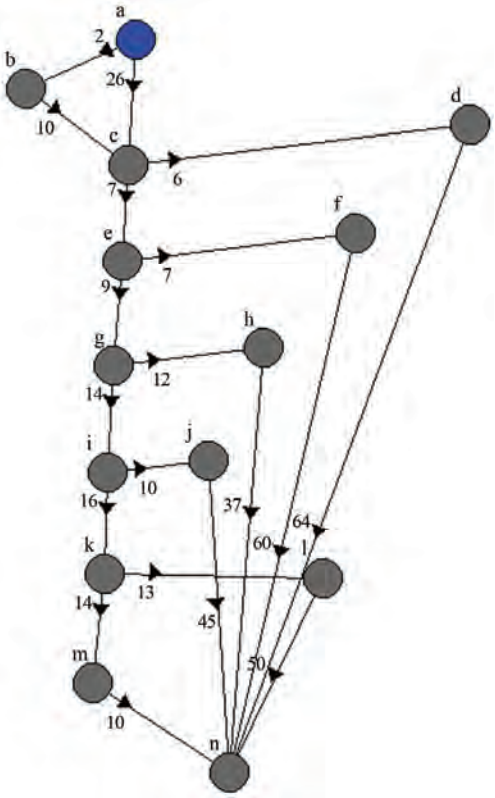



**Figure 5. Initial control flow graph**

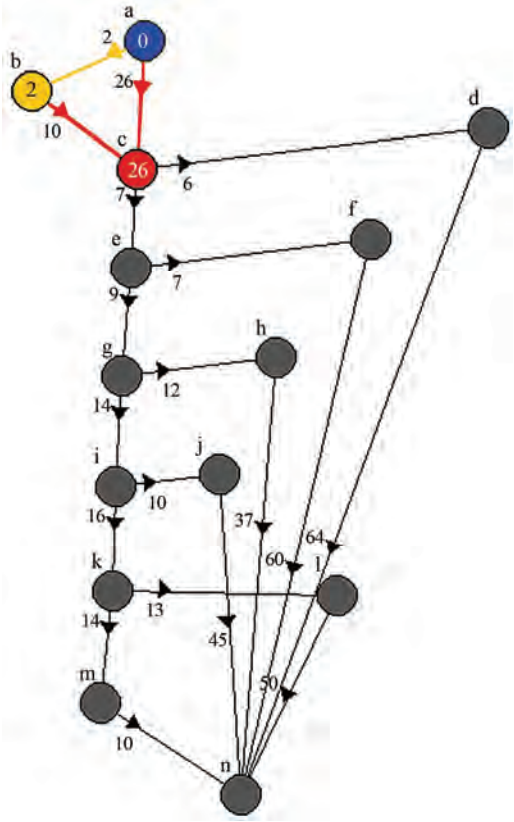



**Figure 7. Cost at node c**

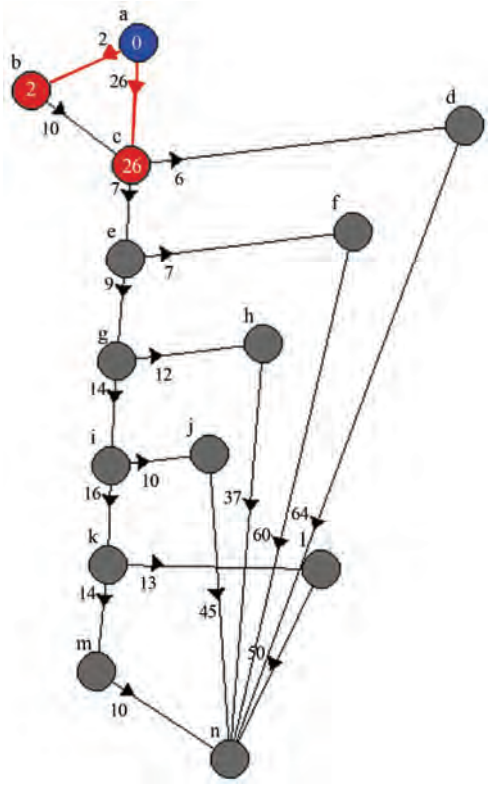



**Figure 6. Cost at node b**

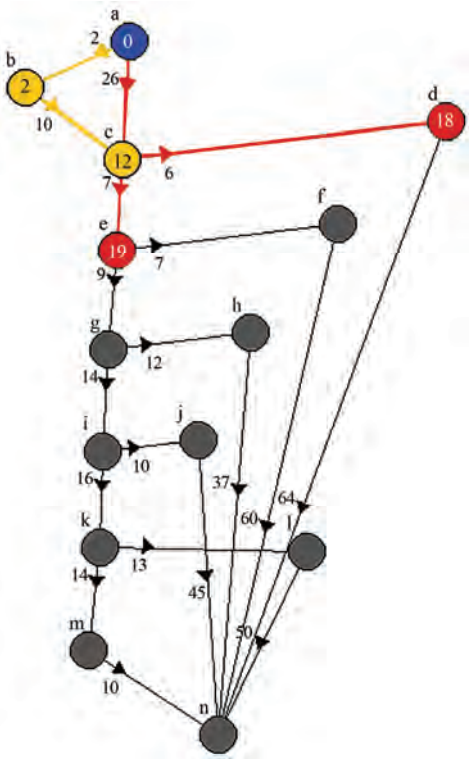



**Figure 8. Cost at node d**

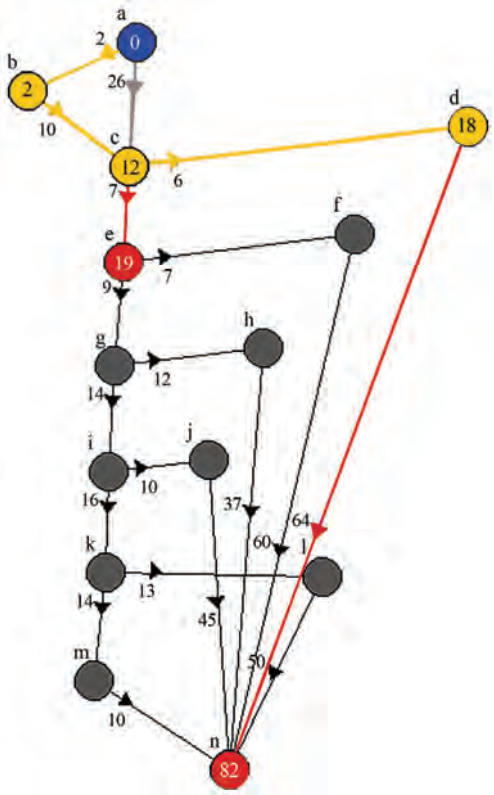



**Figure 9. Cost at node n**

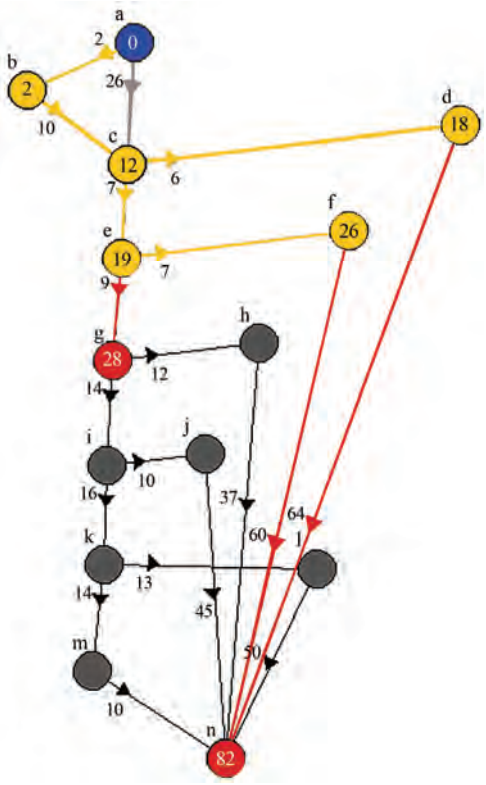



**Figure 11. Cost at node e**

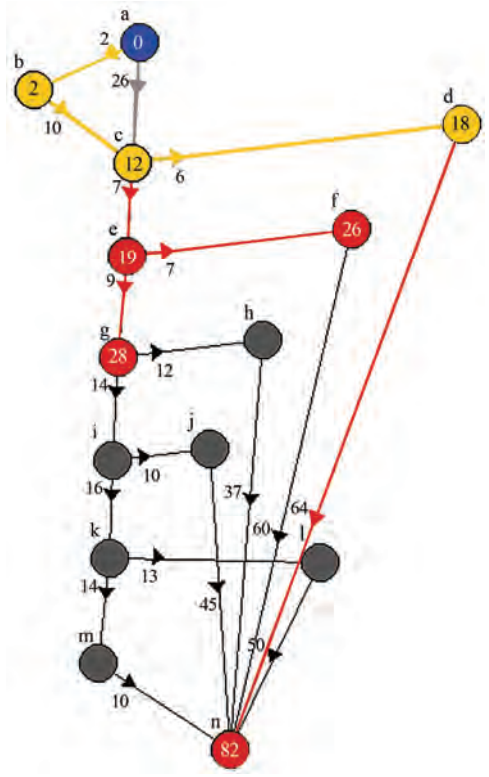



**Figure 10. Cost at node e**

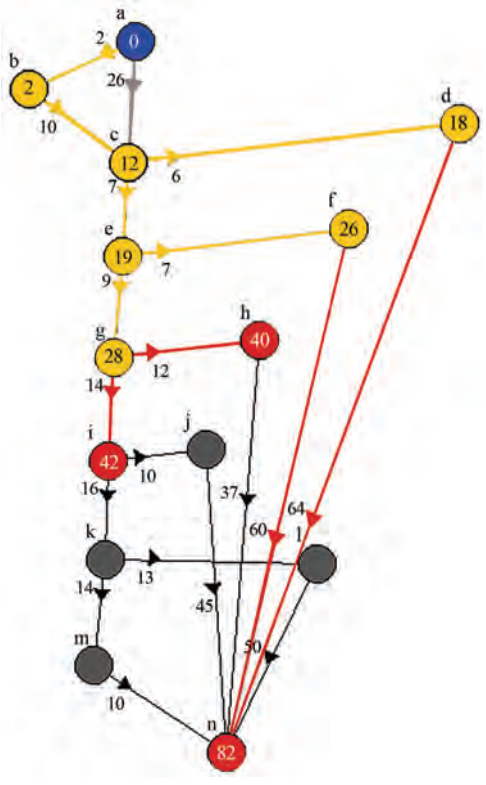



**Figure 12. Cost at node f**

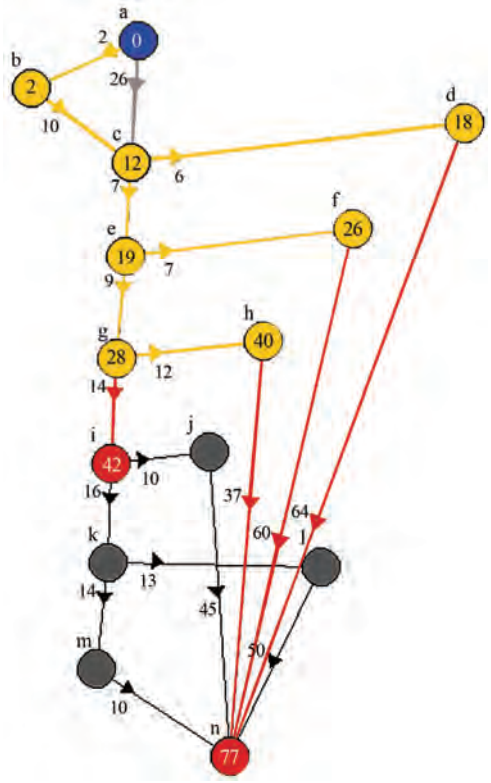



**Figure13. Cost at node g**

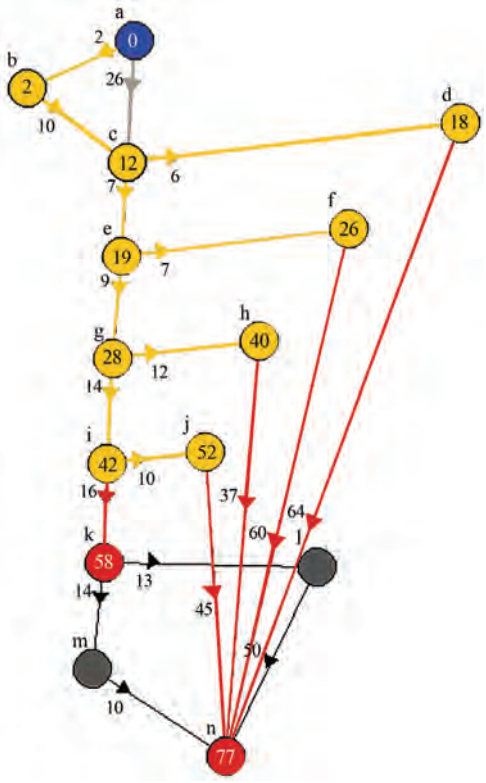



**Figure15. Cost at node i**

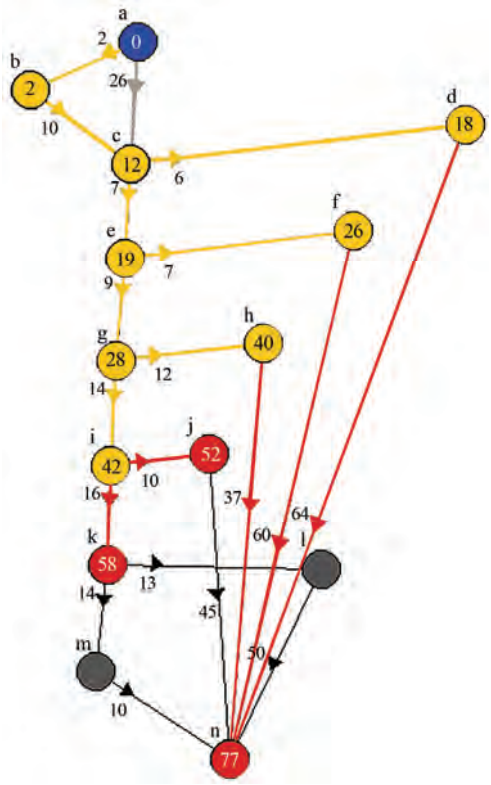



**Figure14. Cost at node h**

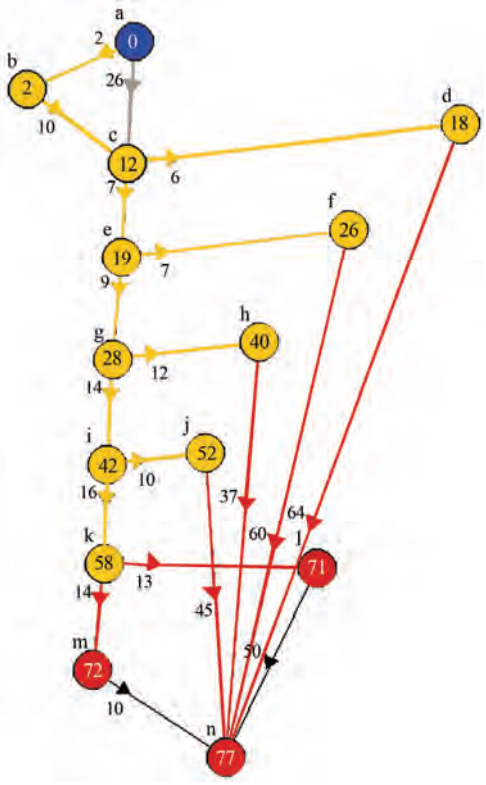



**Figure16. Cost at node j**

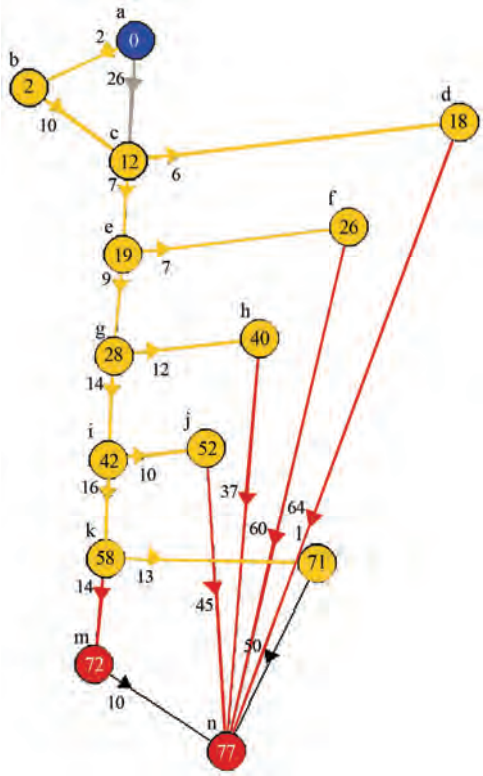



**Figure17. Cost at node k**

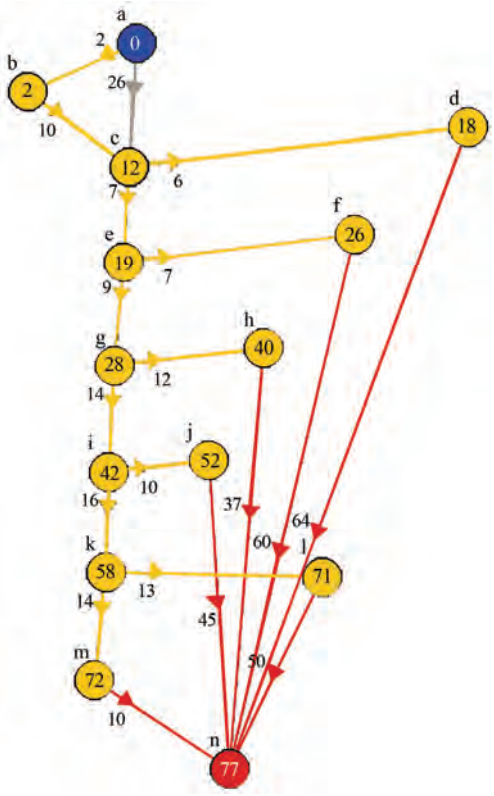



**Figure18. Cost at node l**

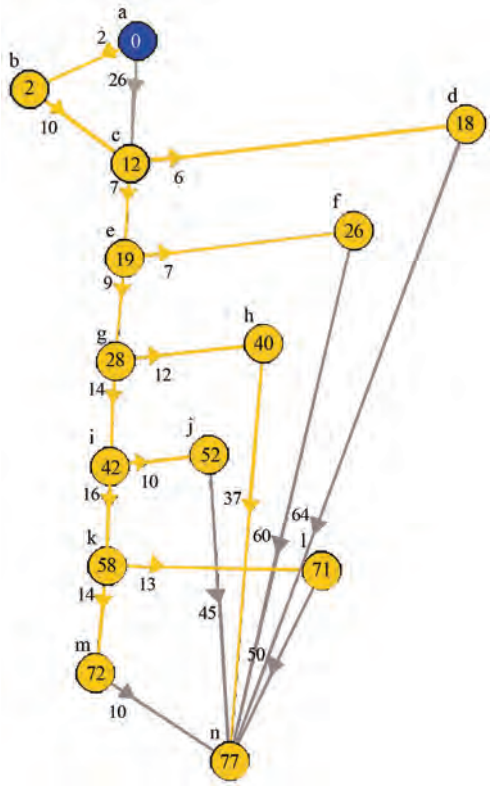



**Figure19. Cost at node m**

13) Here the next node will be chosen as ‘l’.

14) Here the next node will be chosen as ‘n’. In this step the algorithm will backtrack since the path through ‘l’ did not give the least cost path. The next node will be chosen as ‘m’.

15) Here the next node will be chosen as ‘n’.

Hence in accordance, with the above illustration we conclude that proposed algorithm gives maximum code coverage along with least cost.

Hence from the table we can see that in the long term memory all paths have been tested. Tabu list will provide least cost path with maximum coverage in the future if the similar type of module comes for the development then only tabu path can be tested.

## 8. Conclusions and Further Work

This paper presents a meta-heuristic search technique that depends upon the neighborhood solution. There are a lot of real world problems that have been solved by tabu search. Many authors have published their work on tabu search in area of software testing. But this is unique kind of work what we have proposed in this paper solves the problem of cost optimization in software testing. This paper presents use of tabu search with dijksra algorithm (a greedy approach) to provide an efficient path with maximum code coverage and minimum cost.

The structure of the paper shows that tabu search de-

**Table 1. Table for calculation of tabu path**

| Iteration | Long Term Memory | Short Term Memory | Tabu List | Cost |
|---|---|---|---|---|
| 1. | a-b (2)<br>a-c (26) | a-b (2) | Nil | 2 |
| 2. | a-b-c(12)<br>a-c (26) | a-b- (12) | Nil | 12 |
| 3. | a-b-c-d (18)<br>a-b-c-e (19) | a-b-c-d (18) | Nil | 18 |
| 4. | a-d-c-d-n (82)<br>a-b-c-e (19) | a-b-c-e (19) | Nil | 19 |
| 5. | a-b-c-e-g(28)<br>a-b-c-e-f (26) | a-b-c-e-f (26) | Nil | 26 |
| 6. | a-b-c-e-f-n (82)<br>a-b-c-e-g (28) | a-b-c-e-g(28) | Nil | 28 |
| 7. | a-b-c-e-g-h (40)<br>a-b-c-e-g-i (42) | a-b-c-e-g-h (40) | Nil | 40 |
| 8. | a-b-c-e-g-h-n (77)<br>a-b-c-e-g-i (42) | a-b-c-e-g-i (42) | Nil | 42 |
| 9. | a-b-c-e-g-i-j (52)<br>a-b-c-e-g-i-k (58) | a-b-c-e-g-i-j (52) | Nil | 52 |
| 10. | a-b-c-e-g-i-j-n (97)<br>a-b-c-e-g-i-k (58) | a-b-c-e-g-i-k (58) | Nil | 58 |
| 11. | a-b-c-e-g-i-k-l (71)<br>a-b-c-e-g-i-k-m (72) | a-b-c-e-g-i-k-l (71) | Nil | 71 |
| 12. | a-b-c-e-g-i-k-l-n (121)<br>a-b-c-e-g-i-k-m (72) | a-b-c-e-g-i-k-m (72) | Nil | 72 |
| 13. | a-b-c-e-g-i-k-m-n (82)<br>a-b-c-e-g-h-n (77) | a-b-c-e-g-h-n (77) | Path is a-b-c-e-g-h-n (77) | 77 |

pends upon searching the neighboring nodes and memorizing the best solution in its short term memory. All other related solutions are memorized in long term memory of the system which makes tabu search simple to apply in the problem, that is maximizing code coverage with minimum cost.

Furthermore, we have also introduced the concept of backtracking to distinguish those solutions that trapped us towards local minima. We tried to cover the uncovered nodes and the conditions in the program .The system have been tested for several examples. Furthermore the experimental results for one of them are detailed above what we have obtained with the proposed algorithm making it an effective technique in coverage area (branch, path, condition). There is further scope for future work because this strategy is applicable at low level and the moderate level of testing. More work in this area can be carried out to use this technique for projects dealing with complex level testing issues.

## REFERENCES


[1] B. Beizer, "Software Testing Techniques," 2nd Edition, van Nostrand Reinhold, New York, 1990.

[2] I. Sommerville, "Software Engineering, Pearson Education," 7th Edition, Tata Mc-Graw Hill, India, 2005.

[3] E. Diaz, J. Tuya and R. blanco "Automatic Software Testing Using a Metaheuristic Technique Based on Tabu Search," *Proceedings* 18*th IEEE International Conference on Automated Software Engineering*, Montreal, 2003, pp. 301-313.

[4] E. Díaz, J. Tuya, R. Blanco and J. J. Dolado, **"**A Tabu Search Algorithms for Structural Software Testing," *ACM proceeding*, Vol. 35, No. 10, October 2008, pp. 3052-3072.

[5] P. McMinn, "Search-based Software Test Data Generation: A Survey", *Software Testing, Verification and Reliability*, ACM library, Vol. 14, No. 2, 2004, pp 105-106.

[6] F. Glover, "Tabu Search Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, 1989, pp. 190-206.

[7] E. Díaz, J. Tuya, R. Blanco. "A Modular Tool for Automated Coverage in Software Testing," *Proceedings of the 7th Annual International Workshop on Software Technology and Engineering Practice*, Amsterdam, 2003, pp. 241-246.

[8] R. Blanco, J. Tuya and B. A. Diaz, "Automated Test Data Generation Using a Scatter Search Approach," *Information and Software Technology*, Vol. 51, No. 4, 2009, pp. 708-720.

[9] A. Misevičius, "Using Iterated Tabu Search for the Travelling Salesman Problem," *Information Technology and Control*, Vol. 32, No. 3, 2004, pp.29-40.

[10] R. Thamilselvan, D. P. Balasubramanie, "Integrating Genetic Algorithm, Tabu Search Approach for Job Shop Scheduling," *IJCSIS Transactions on Software Engineering*, Vol. 2, No. 1, 2009, pp. 134-139.

[11] J. Gustafsson, A. Ermedahl, and B. Lisper, "Algorithms for Infeasible Path Calculation," 6*th International Conference on Worst-Case Execution Time*, Dresden, Euromicro Conference on Real-Time Systems, 2006.

[12] R. Jasper, M. Brennan, K. Williamson and B. Currier, "Test Data Generation and Feasible Path Analysis," *Pro-*

*ceeding of the International Symposium on Software Testing and Analysis*, Seattle, ACM, 1994, pp.95-107.

[13] J. Carlos, M. Alberto and Francisco, "A strategy for Evaluating Feasible and Unfeasible Test Cases for The Evolutionary Testing of Object-oriented Software," 30*th International Conference on Software Engineering*, Leipzig, 2008, pp. 85-92.

[14] J. C. Lin and P. L. Yeh, "Automatic Test Data Generation for Path Testing Using GAs," *Information Sciences*, Vol. 131, No. 1-4, 2006, pp. 2380-2401.

[15] F. Glover and M. Laguna, "Tabu Search," Kluwer Academic Publishers, 1997.

[16] "Dijkstra's Algorithm," Wikipedia. http://en.wikipedia.org/wiki/Dijkstra's_algorithm

*J. Software Engineering & Applications*, 2010, 3: 487-494
doi:10.4236/jsea.2010.35055 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Research on Knowledge Creation in Software Requirement Development*

**Jiangping Wan[1,2], Hui Zhang[1], Dan Wan[1], Deyi Huang[3]**

[1]School of Business Administration, South China University of Technology, Guangzhou, China; [2]Institute of Emerging Industrialization Development South China University of Technology, Guangzhou, China; [3]E-jing Technologies, Hong Kong Science Park Shatin, Hong Kong SAR.
Email: {scutwjp, dandanwan42}@126.com, shidelan@163.com, huangdy3816@tom.com

Received February 3rd, 2010; revised March 15th, 2010; accepted March 17th, 2010.

## ABSTRACT

*After field survey and literature review, we found that software requirement development* (*SRD*) *is a knowledge creation process, and knowledge creation theory of Nonaka is appropriate for analyzing knowledge creating of SRD. The characteristics of knowledge in requirement elicitation process are analyzed, and dissymmetric knowledge of SRD is discussed. Experts on requirement are introduced into SRD process as a third knowledge entity. In addition, a knowledge creation model of SRD is put forward and the knowledge flow and the relationship of entities of this model are illustrated. Case study findings are illustrated in the following*: 1) *The necessary diversity of the project team can facilitate the implementation of the SRD.* 2) *The introduction of experts on requirement can achieve the transformation of knowledge effectively, thus helping to carry out the SRD.* 3) *Methodology and related technologies are important for carrying out the SRD.*

***Keywords***: *Requirement Engineering, Knowledge Creation, Dissymmetric Knowledge, Knowledge Conversion, Experts on Requirement, Methodology*

## 1. Introduction

Michael Polanyi divided knowledge into tacit knowledge and explicit knowledge [1]. Tacit knowledge exists in human brains, which is the knowledge that people don't know, in other words people don't know what they know. Verna Allee thought that tacit knowledge which exists in individuals is private and has its own special background, and it also depends on experience, intuition and discernment [2]. Nonaka figured that organizations create and make use of knowledge via the interaction of tacit knowledge and explicit knowledge, which is called knowledge conversion process [3]. Li Xiao-Ming *et al.* analyzed the requirement elicitation process (REP) in the view of knowledge management and put forward the countermeasures. In our understanding, it is necessary to discuss how knowledge is converted during REP in details, and present an embodying knowledge conversion pattern [4].Wan Jiangping researched on the knowledge integration support structure of quality software production [5], and established knowledge transfer model of software process improvement (SPI) and the conceptual framework of influencing factors [6].

This paper is organization as following: we firstly analyze the characteristics of knowledge during REP; then based on characteristics of SECI (socialization, externalization, combination, and internalization) knowledge spiral model, dissymmetric knowledge flow theory and knowledge communication, a knowledge conversion model is put forward, which is used to analyze the requirements development process of the NY Company's cost accounting system.

## 2. Literature Review

### 2.1 Tacit Knowledge and Knowledge Conversion

Organizational knowledge creation is the process of making available and amplifying knowledge created by individuals as well as crystallizing and connecting it to an organization's knowledge system. The concept of "tacit knowledge" is a cornerstone in organizational knowledge creation theory and covers knowledge that is unarticulated and tied to the senses, movement skills, physical experiences, intuition, or implicit rules of thumb. Knowledge of wine tasting, crafting a violin, or interpreting a complex

*This research was supported by Key Project of Guangdong Province Education Office (06JDXM63002), NSF of China (70471091), and QualiPSo (IST- FP6-IP-034763)

seismic printout of an oil reservoir are well-known examples of tacit knowledge. The concept of "knowledge conversion" explains how tacit and explicit knowledge interact along a continuum. Nonaka argued that knowledge is created and used in organization through knowledge is transformed process, including four patterns: socialization, externalization, combination, and internalization. All four of these patterns exist in dynamic interaction. Knowledge is really and truly created and used effectively and effectively depending on dynamic business system is established [3,7].

There nine questions are put forward in the following [8]: 1) What is the status of "truth" in the definition of knowledge? 2) Do tacit and explicit knowledge fall along a continuum? 3) Is the tacit/explicit knowledge distinction along the continuum valuable for organization science? 4) What is the conceptual basis of knowledge conversion? 5) Given the relationship between tacit knowledge and social practices, how can the concept of knowledge conversion be upheld? 6) What is the outcome of knowledge conversion? 7) What is the relationship between organizational knowledge creation and social practices in organizations? 8) When and why do social practices contribute to the conservation of existing tacit knowledge and existing routine rather than organizational knowledge creation and innovation? 9) How can leadership motivate and enable individuals to contribute to organizational knowledge creation by transcending social practices?

### 2.2 Enabling Knowledge Creation

There five knowledge enablers in the following: 1) Instill a knowledge vision, 2) Manage conversations, 3) Mobilize knowledge activists, 4) Create the right context, and 5) Globalize local knowledge. The effective knowledge creation depends on an enable context. What we mean by enabling context is shared space that fosters emerging relationships. This definition of context is connected to our first two points: knowledge is dynamic, relational, and rather than on absolute truth or hard facts. The essential thing for managers to remember is that all knowledge, as opposed to information or data, depends on its context. There are the five knowledge-creation steps in the following: 1) Sharing tacit knowledge, 2) Creating concepts, 3) Justifying concepts, 4) Building a prototype, and 5) Cross leveling knowledge [9].

### 2.3 Software Requirements Engineering

Requirements engineering (RE) is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families [10]. There are five core RE activities in the following: 1) Eliciting requirements, 2) Modeling and analyzing requirements, 3) Communicating requirements, 4) Agreeing requirements, and 5) Evolving requirements. This paper analyzes the eliciting requirements.

### 2.4 Characteristics of Knowledge in Software REP

REP is actually a process of developing conditions that satisfied people in the following: 1) Understanding requirements; 2) Participating in this project (in most cases); 3) Understanding how to work effectively as a team [11].The generic process of requirement elicitation is illustrated in **Figure 1**. First, we work with our customers to elicit the requirement, by asking questions, demonstrating similar systems, or even developing prototypes of all or part of the proposed system. Next, we capture those requirements in a document or database. Then, the requirements are often rewritten, usually in a more mathematical representation, so that the designers can transform the requirements into a good system design. A verification and validation step makes sure that the requirements are complete, correct, consistent, and the requirements are what the customer intends to [12].

In software REP, besides the requirements that clients or users are able to bring forward, they also have requirements that could not be clearly expressed. The characteristics of software tacit requirements are summed up in the following: 1) Tacit requirements are hard to express, convert, communicate and share; 2) Tacit requirements are often related to application domain; 3) Tacit requirements are often users' tacit knowledge; 4) Tacit requirements are experiential knowledge which developing team accumulates step by step in practice during a long period of time; 5) Tacit requirements are hard to encode and articulate; 6) Tacit requirements can be expressed hazily and crudely.

## 3. Establishment Knowledge Creation in Software REP

### 3.1 Software REP in the View of SECI Knowledge Spiral Model

SECI Knowledge Spiral Model illustrates the relationship between tacit knowledge and explicit knowledge and how they convert into each other, and it also illustrates the way tacit knowledge and explicit knowledge convert and share. Finally new knowledge is created in this conversion process.

1) **Socialization:** It is tacit knowledge from individual to individual, and in the process people actualizes the expression and conversion of knowledge by means of observation, imitation, etc. In software REP, knowledge socialization includes the internal conversion between users' knowledge, which is the principal part and the external conversion of tacit knowledge among users,

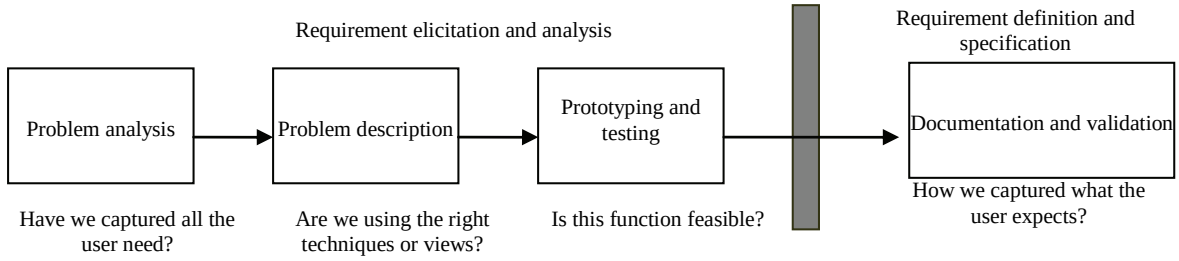



**Figure 1. Software REP**

experts on requirement, and developers. Finally they all reach consensus.

2) **Externalization:** It is the conversion from tacit knowledge to explicit knowledge, expressing tacit knowledge written down or stored in computer, etc. In software REP, mainly with the help of experts on requirement or consultants, knowledge externalization is the process in which users' tacit knowledge is converted into explicit knowledge, which could be directly accepted by developers.

3) **Combination:** It is the process from separate explicit knowledge to systematic explicit knowledge in which explicit knowledge is further systematized and complicated. It involves different kinds of external explicit knowledge system. In software REP, with knowledge of experts on requirement further systematized, developers integrate their own knowledge to make the requirements specification.

4) **Internalization:** It is the process from explicit knowledge to tacit knowledge; in the process individuals digest and adsorb the knowledge from different mediums, and make it into their own abilities. In software REP, experts on requirement pass the combined explicit knowledge (requirements specification or prototype) to users who may consequently have a further understanding of the system.

## 3.2 Analysis of Dissymmetric Knowledge Flow

Because of dissymmetric knowledge, knowledge flows from high knowledge level to low knowledge level (**Figure 2**). There are two main knowledge flows: the one from users to developers is mainly knowledge in business domain and other users' tacit knowledge, while the other one from developers to users is knowledge in software domain and system performance requirements, etc.

The flow of dissymmetric knowledge makes the conversion of knowledge type, namely integrating the knowledge of developers and users and making users' tacit knowledge articulate to form their very requirements, which are the critical parts of REP. After feeding back, the integrated knowledge flows to users, and then when it is confirmed, REP is initially finished. But this does not imply that the REP is ended; on the contrary, during the whole process, software requirements are gradually updated along with feedbacks of information from different stages [13].

Software REP is the process in which users, experts on requirement, developers and other demand sources communicate their information and knowledge, and it is also the process in which the knowledge of users, experts on requirement and software developers are integrated. Because of users' participation, experts on requirement and developers may absorb users' information to make it into knowledge, and update their knowledge with their experience, cooperation value, project goals and business principles, which may reach the goal of requirement elicitation. Therefore, REP is the process of knowledge flow, integration and innovation. Meanwhile, users improve themselves continuously in the process of knowledge flow.

## 3.3 Conditions for Knowledge Conversion

In software REP, the face-to-face communication which is aimed at knowledge conversion is different from generic communication [14], firstly, the two parties are quite dissymmetric in knowledge; then the communication is to elicit users' tacit knowledge and show developers' software knowledge; finally, the process of communication is an iterative negotiation process.

In the communication for requirement elicitation, users, experts on requirement, developers and relevant personnel should clearly understand each element in successful knowledge conversion, and only the honest speakers who own knowledge and listeners who trust and understand speakers may interact to reach the goal of knowledge conversion and tacit knowledge elicitation[15].

## 3.4 Model of Knowledge Conversion in REP Based on SECI

The model of knowledge conversion in REP based on SECI is illustrated in **Figure 3**.

In software REP, the externalization process of tacit requirements is the process in which users and developers communicate via experts on requirement. As the central part in REP, experts on requirement not only listen attentively to users' requirements and developers' description of technologies and computer, but also win the trust of users and developers as authorities and able persons,

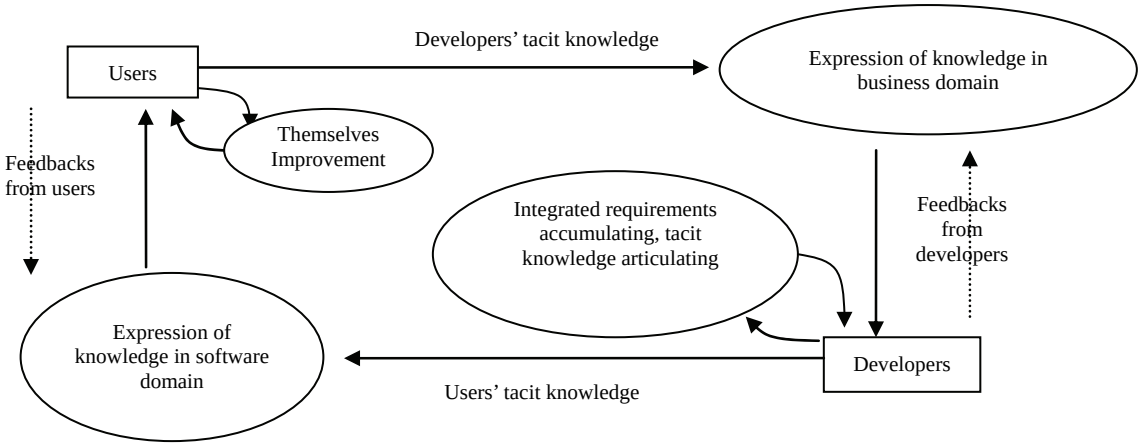



**Figure 2. Flows of dissymmetric knowledge**

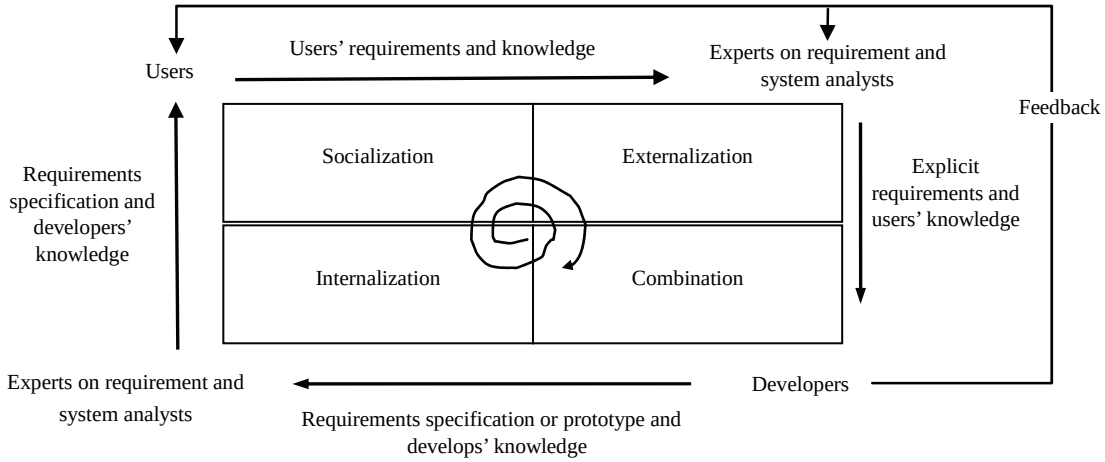



**Figure 3. Model of knowledge conversion in REP**

based on which they put forward users' tacit requirements in a way that developers can easily understand and accept, and present developers' solution in a way that users can easily understand and accept.

REP is a multi-stage process of knowledge conversion and implementation. In this iterative conversion process, experts on requirement who master "logic" domain knowledge face users and developers. They may perform concept design, decomposition and requirement integration for users, while performing consistency analysis, knowledge optimization and flexibility processing for developers. Owning to the dissymmetry of knowledge level, high-level knowledge flows to low-level knowledge, users' business domain knowledge and other tacit knowledge to developers, and developers' software domain knowledge to users. Experts on requirement bring users' knowledge to developers. Developers analyze requirements, feed back to users and experts on requirement, accept their feedbacks, and finally produce requirements specification, making the process of combination. After requirements specification is formed, experts on requirement make explanation to users and provide relevant training, making the process of knowledge internalization. In each step of SECI, participations should apply other four steps to creative thinking in the following: preparation, incubation, illumination, and verification [16].

### 3.5 Keys to Model of knowledge Conversion in REP Based on SECI

When using the model of knowledge conversion in REP based on SECI to guide requirement elicitation, there is something should be paid attention to in the following: 1) Choose the experts on requirement who are familiar with users' business domain and are trusted by both developers and users. 2) Users should take part in the REP actively, at least appointing customer representatives, including user mouthpieces and system users, etc. 3) Experts on requirement should make an observation and have a discussion with both users and developers on the spot. 4) The requirements specifications made by experts on requirement should be the negotiation by the three parties and be validated by users and developers. 5) Experts on re-

quirement should participate in the whole process of requirement elicitation, and harmonize all kinds of conflicts between users and developers. 6) All activities require a right context, called "Ba" in Japanese [9], such as creative workshops [16], meeting, and so on.

# 4. Case Study

## 4.1 Overview

The NY Company was founded in 1986 and has two branches now. In November 2006, NY Company's top executives contacted managers of JZ Company and initially established the project through the communication of the two sides. JZ Company was committed to achieve the initial system within 3 months and chose the branch as a pilot test run.

## 4.2 Asymmetry of the Project Knowledge

The NY Company is not satisfied with the system vision and planning established by the software providers. This issue is largely from software provider's deficient understanding of the business processes of the catering industry, particularly in its cost accounting processes. On the other hand, NY Company has deficient or inaccurate understanding of the software (**Table 1**).

## 4.3 The Requiring Expert of the Project

The project was officially approved by the project team in November 2006, including the NY Company as the customer, the JZ Company as the experts on requirement, and the SY Company as the developer. The knowledge traits of the three parties in the project team are illustrated in **Table 2**.

Firstly, JZ Company determines what system that NY Company needs and what system SY company should provide. JZ Company further deepens the requirements NY Company proposed. They analyze the project in the view of the entire company and identify the cost accounting containing operating costs, inventory costs, purchasing costs, as well as costs monitoring and budgeting, not just a simple ordering system or accounting analysis. Secondly, JZ Company also needs to analyze clearly what software system is compatible with the NY Company, transforming the enterprise business processes into a computer system processing and the computer's function modules into enterprise business functions.

## 4.4 Knowledge Creation Process in the Project

### 4.4.1 Situation of the Project Team

JZ organizes formal meetings within the project team actively, such as project-startup meeting, various seminars, weekly meetings, periodic meetings, etc. And the experts on requirements will keep a record for each meeting and store them into the project database as backup.

### 4.4.2 Project Process

NY Company introduced their initial requirements to JZ Company, including cost management businessprocess diagrams and related reporting requirements, and clarified them in depth and carried out business training.

JZ Company investigated respectively inspecting department, financial department, administrant department and had in-depth interviews with relevant personnel. JZ Company's staff has a good knowledge of accounting, business management knowledge, and system planning knowledge, so during the interview process they could soon be able to reach a consensus with the NY company personnel to develop the "Investigation and Analysis Specification of NY Company's Cost Accounting System." The specification clearly identified the main objective of the system, outlined the NY Company's organizational structure, identified the enterprise's overall operational processes, analyzed carefully the management of the catering industry and established the internal processes of the cost-accounting system (**Figure 4**), which divided the system into six modules, i.e. namely data collection, cost data, report output, initialization, query, system management and also encoded the information in the processes. After communication with the general manager of NY Company, it was clarified that the system would be used by the inspecting department, while the financial department would only use it for data queryand the administrant department would collect data. System requirements would be mainly determined by the inspecting department.

JZ Company exchanged ideas with NY Company and illustrated the figure so that both the two parties could understand and accept it. At the same time, the "NY

**Table 1. Asymmetry of the project knowledge**

| | Customer | Developer |
|---|---|---|
| **Asymmetry of direction** | 1. A system contributing to control the cost. | 1. It is not started from the information systems of the entire enterprise.<br>2. The goal of the system is not defined correctly, thinking it is just an ordering system. |
| **Asymmetry of distance** | 1. What the software system can do for cost control is not clearly learnt.<br>2. Lack the computer system knowledge.<br>3. Clear about the daily operating processes of the catering industry. | 1. Deficient understanding of the cost management in catering industry; Unclear about how to control the cost and which part to control.<br>2. Lack the cost management business processes knowledge.<br>3. Well-informed of the software development |

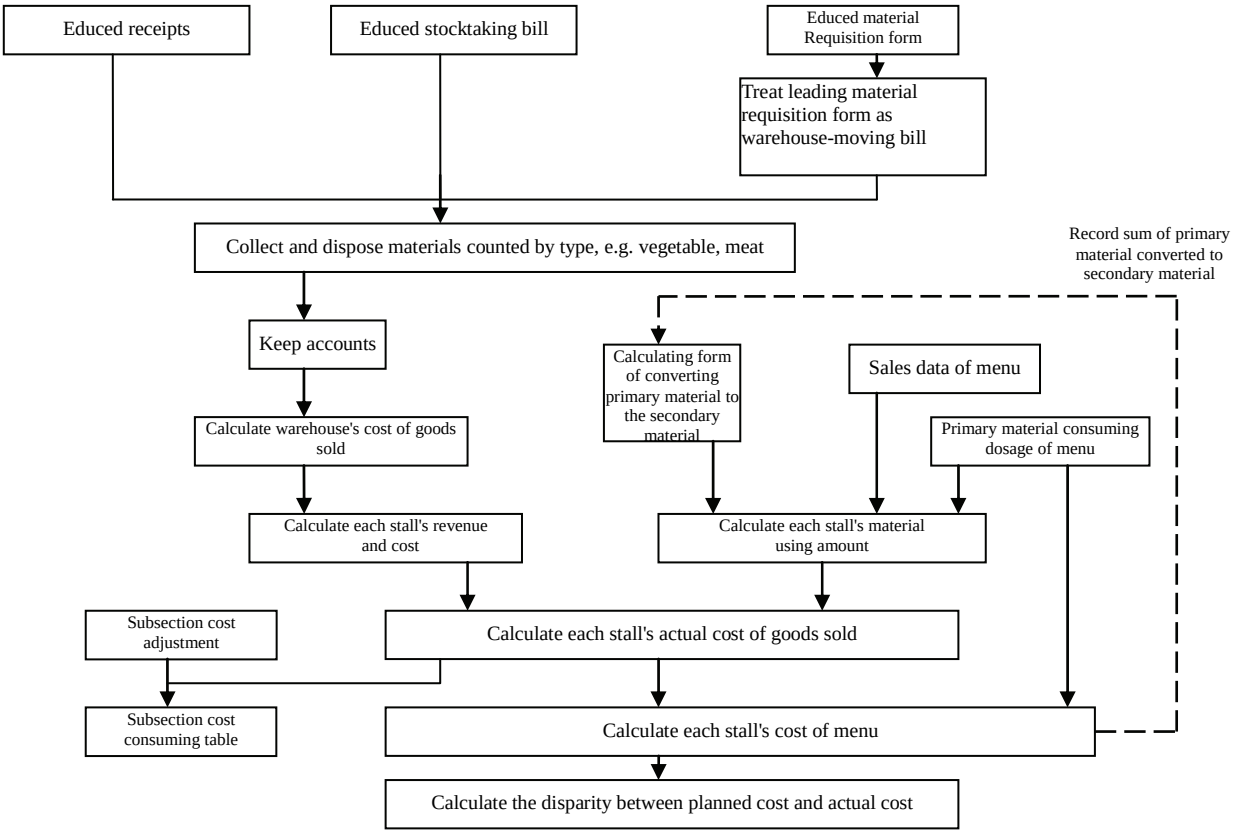



**Figure 4. Internal processes of cost accounting system**

**Table 2. Knowledge traits of the three parties in the project team**

| | **Knowledge traits** |
|---|---|
| **Customer（NY）** | Operating and cost management knowledge of catering industry |
| **Developer（SY）** | Software and system developing knowledge |
| **Requirements experts（JZ）** | Organization management knowledge, cost-control knowledge, system design and developing knowledge , project management knowledge and ability, requirements acquiring knowledge and ability |

Company's Cost Accounting System Design Specification V1.0" and simple system prototype, which included the interface, the core module and so on, were completed.

JZ company and SY company further refined the requirements, forming the "NY Company's Cost Accounting System Design Specification V2.0" and the improved system prototype, and trained the customers. The three parties signed on the "NY Company's Cost Accounting System Requirements Specification" and "NY Company's Cost Accounting System Design Specification", confirming the system entering into construction phase.

The key to effectively realizing requirements' transformation and requirements' knowledge creation are illustrated in **Table 3(a)** & **Table 3(b)**. 1) The three parties of the project should have mutual trust and actively participate in the project process, and basing on a clear project objective carry out the project with a detailed project plan. The appropriate methodology and technology should be adopted rather than merely seeking advancing. Face to face communication is mainly used while thoughts can be expressed in the form of prototype, ensuring effective communication and avoiding unnecessary confusion. 2) JZ Company needs to construct a platform for requirements' transformation and lead the entire project process. JZ Company conducts a comprehensive and in-depth analysis to NY company, including organizational structure, status of existing enterprises' information technology and so on, understands the cost accounting systems in the view of the whole organization and proposes extensible project blue print. Meanwhile, it should integrate professional knowledge of cost accounting, business operating knowledge and computer knowledge

**Table 3(a). Requirements creation process**

| **Project phase** | Project proposing | Project preparation | Project initiation | Relevant data collection | Requirements educing | Requirements classification |
|---|---|---|---|---|---|---|
| **Participators** | Customer | Customer, other software providers | Customer, experts on requirements, developers | Experts on requirements, customer | Experts on requirements, customer | Experts on requirements , developer |
| **Requirements statement** | Meeting consensus | Relevant document | Project objectives, initial cost management business process | Organizational structure, "NY company's proposal of informationization" | "Survey and analysis of NY company's cost accounting system", " General plan of NY company's cost accounting system project" | "Resolution to NY company's cost accounting system", " Interface specification of NY company's cost accounting system" |
| **Knowledge phase** | Customer internalization and socialization | | Project socialization | Internalization and socialization | | Externalization |
| **Key problems** | Enterprise development needs, executives strongly demand | In-depth learning about system through several contacts with software providers | Guidance for experts on requirement, acquiring customer's trust | Comprehensive investigation on company's structure, culture, existing systems, business processes, etc. | Adopt some requirements educing technology, e.g. Old system analysis, seminar, questionnaire survey, interview, etc. | Identify functional requirements and nonfunctional requirements, such as restriction, performance, revisability and so on |

**Table 3(b). Requirements creation process**

| **Project phase** | Initial modeling | Requirements discussion | Model refinement | Requirements validation | System realization | System test-run | Requirements alteration |
|---|---|---|---|---|---|---|---|
| **Participators** | Customer, requirements experts | Experts on requirements, developer, customer | Experts on requirements, developer | Experts on requirements, developer, customer | Developer | Experts on requirements, developer, customer | Experts on requirements, developer, customer |
| **Requirements statement** | "NY company's cost accounting system design specification V1.0", simple system proto-type, including the inter-face, the core module, etc. | "Summary specification of NY company's cost accounting system discussion" | "NY company's cost accounting system design specification V2.0", the improved system prototype | "NY company's cost accounting system requirements specification", "NY company's cost accounting system design specification" | NY company's cost accountting management system, "test report", "specifiction of system installation, setting, uninstall or upgrading" and relevant document | "NY company's cost accounting system's user manual", "NY company's cost accounting system's test-run report" | "NY company's cost accounting system requirements alteration specification" |
| **Knowledge trans- forming phase** | Combination | Combination, externalization, internalization | Combination | Internalization | Combination (realization) | Internaliza-tion | Requirements' transformation |
| **Key problems** | Use dynamic displaying document | Arrange special seminars to discuss the topics | Develop workable prototype, perfect system design specification | Take the requirements specification and system design as the original source to validate the project | Experts on requirements monitor the system realization as customer representa-tives | | |

and ability, and reduce the knowledge asymmetry between NY Company and SY Company.

### 4.5 Summary

The conclusions are in the following: 1) Friendly project environment. The three parties were very friendly, and in the course of the project gradually established a mutual trust relationship. 2) Executives' high consideration and the relevant members' active participation in the project. As customer exactly knew the project objectives and was very concerned about this project, so they actively participated in the project and cooperated with other members well. 3) Clear project objectives and plan. Face to face communication was mainly used to communicate the project so as to avoid unnecessary chaos and individual blind autonomy. The project team also encouraged informal communication, and the communication process should be recorded and summarized. 4) Introduction of requiring expert reduces knowledge asymmetry. Specialists are knowledgeable, well-informed of accounting knowledge, business management knowledge and software knowledge, so they play a role of knowledge communication platform. 5) Appropriate project management methodology and technology. If the team is wild about advanced methodology in the really critical project, the project will often end up with failure. That is because introduction of advanced methods and concepts is often a gradual process that requires adaptation. Furthermore, experts on requirement are good at project management, helping to carry out the project orderly.

## 5. Conclusions

In software REP, the knowledge of both the users and developers can be divided into explicit knowledge and tacit knowledge, and REP is an iterative process of knowledge socialization, externalization, combination and internalization. Knowledge dissymmetry is one of the forces that drive knowledge conversion. The model of knowledge conversion in REP based on SECI is put forward. Depending on experts on requirement, this model can reduce knowledge dissymmetry, and realize knowledge conversion and share more effectively and efficiently. This model is used to analyze the requirements development project of the NY Company's cost accounting system. Case study findings are in the following: 1) It can facilitate the implementation of the project to have the necessary diversity of the project team. 2) The introduction of requiring expert can achieve the transformation of knowledge effectively, thus helping to carry out the SRD. 3) Methodology and related technologies are important for carrying out the SRD.

## 6. Acknowledgements

Thank for helpful discussion with Mr. Li Jiangzhang, Mr. Chen Zhening, Mr. Wang Shuwen, Mr. Liu Bing, Brenda Huang etc.

## REFERENCES

[1] M. Polanyi, "Personal knowledge," University of Chicago Press, Chicago, 1958.

[2] V. Allee, "The Knowledge Evolution: Expanding Organizational Intelligence," Butterworth Heinemann, Boston, 1997.

[3] I. Nonaka, "The Knowledge Creating Company," *Harvard Business Review*, Vol. 69, No. 6, 1991, pp. 96-104.

[4] X. M. Li, *et al*. "Research on Software Requirement Management based on Knowledge Management," *R&D Management*, Vol. 2, 2005, pp. 28-39.

[5] J. P. Wan, "Research on Software Product Support Structure," *Journal of Software Engineering and Applications*, Vol. 2, No. 3, 2009, pp. 174-194.

[6] J. P. Wan *et al.*, "Research on Knowledge Transfer Influencing Factors in Software Process Improvement," *Journal of Software Engineering and Applications*, 2010, Vol. 3, No. 2, 2010, pp. 134-140.

[7] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science*, Vol. 5, No. 1, 1994, pp.14-37.

[8] I. Nonaka and V. K. Georg, "Perspective—Tacit Knowledge and Knowledge Conversion: Controversy and Advancement in Organizational Knowledge Creation Theory," *Organization Science*, Vol. 20, No. 3, 2009, pp. 635-652.

[9] V. K. Georg, *et al*. "Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation," Oxford University Press, New York, 2000.

[10] P. Zave, "Classification of Research Efforts in Requirements Engineering," *ACM Computing Surveys*, Vol. 29, No. 4, 1997, pp. 315-321.

[11] D. C. Gause and G. M. Weinberg, "Exploring Requirements: Quality before Design," Dorset House Publishing Co., Inc, Vancouver, 1989.

[12] S. L. Pfleeger, "Software Engineering: Theory and Practice," 2nd Edition, Higher Education Press, Beijing, 2002.

[13] S. Alshawi and W. A. Karaghouli, "Managing Knowledge in Business Requirements Identification," *Logistics Information Management*, Vol. 16, No. 3, 2003, pp. 341-349.

[14] Q. F. Shi, *et al.* "Characteristics and Modal Analysis of Implicit Knowledge Transfer," *Studies in Dialectics of Nature*, Vol. 20, No. 2, 2004, pp. 62-68.

[15] X. J. Xv, "On the Transmission of Knowledge," *Studies in Science of Science*, Vol. 23, No. 3, 2005, pp. 298-303.

[16] N. Maiden, *et al.* "Provoking Creativity: Imagine What Your Requirements Could be Like," *IEEE Software*, Vol. 21, No. 5, 2004, pp. 68-75.

*J. Software Engineering & Applications*, 2010, 3: 495-502
doi:10.4236/jsea.2010.35056 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Raising Awareness of the Constituents of Software Design – The Case of Documentation

**Lavy Ilana, Yadin Aharon**

Management Information Systems, Emek Yezreel Academic College, Emek Yezreel, Israel.
Email: {ilanal, Aharony}@yvc.ac.il

Received December 6th, 2009; revised December 21st, 2009; accepted December 23rd, 2009.

## ABSTRACT

*This research was performed within a software engineering workshop for Computer Science students. For addressing the soft skills issues required by the industry, the course was delivered as a workshop with various* (*inter and intra*) *team based activities. An additional objective of outlining the importance of software maintainability issues was achieved through team-based role play. There were three assignments in which each team had to continue the work performed by another team, thus creating a dependency between the teams as might happen during maintenance. The main research study objective was to examine the effect of employing this kind of a team-based role-play peer-review on the students' learning process regarding maintainability. Data referring to the students' perceptions is presented and analyzed in addition to student reflections on the workshop which demonstrate their expanded understanding of the design and application process.*

***Keywords*****:** *Team-Based Role Play, Software Engineering, Peer Review, Maintainability*

## 1. Introduction

The term software engineering appeared first at a NATO conference in 1968 [1] and it was intended to ignite discussions on the process of developing correct, testable, and understandable computer programs. At that time, the "software crisis", that was partially caused by the rapid developments in computer technologies combined with more, complex user requirements, and the lack of "engineering" methodologies for software development [2]. Since then the process of software engineering has matured and is accepted as a proven learning discipline. The Software Engineering course is an important part of the Computer Science (CS) and Information Systems (IS) curricula, however many students regard it as less applicable in their future careers [3]. Information systems and software based systems in general, require follow-up maintenance due to the existence of potential bugs that will have to be corrected, the high likelihood of functional enhancements to be introduced to the programs. For lowering the costs associated with these ever increasing needs for software maintenance adoption of proper software engineering methodologies is required. Thus software maintainability plays a critical function in the software engineering process, not unlike the role of software development itself.

Cognizant of the students' difficulties regarding non-technical knowledge such as critical thinking, interpersonal and team based skills, the Software Engineering workshop structure employs many inter-team and intra-team activities. Furthermore, to raise the students' awareness to the importance of documentation and the role it plays in maintainability, the workshop employs an incremental life-cycle involving each team in three activities: design (including documentation), development, and testing. However, unlike the ordinary software development life-cycle, in which each team performs the three activities for the same project the workshop structure employs a team-based role play. By team-based role play we mean that the design, development and testing is swapped among the teams. Initially all the teams are given the same project, and each team prepares his own design. The second assignment consists of developing the system according to the design specifications, but each team develops a system that was designed by a different team and not the system it has designed. The third assignment consists of defining the test specifications and testing the system, however, once again, each team tests a system designed by one team and developed by another. When proceeding to the next assignment, the team is required to ignore their prior knowledge or ideas and to concentrate only on the system as it has been designed (or developed) by another team of their peers.

The main research study objective was to examine the effect of employing this kind of a team-based peer-review on the students' learning process in a software engineering workshop with special emphasis on their perception regarding documentation. This paper describes the workshop structure and the quantitative and qualitative results obtained from employing it.

## 2. Theoretical Background

Software development is a collaborative task that employs teams of developers working together [4]. To enhance software readability and maintainability, software engineering practitioners have been striving to improve existing tools and methodologies. Among these various practices, documentation - used to describe the required software, its structure, logic and performance - is high on the list [5,6]. Without the proper documentation, the future maintainer will find it extremely difficult to understand the system or its processes. Poor and missing documentation is a major contributor to software quality degradation and aging, compounding an increase in maintenance costs [7]. In spite of this, many students fail to recognize the importance of documentation on maintainability [3].

Software engineering is an integration of many practices, methodologies and tools.. One of the initial practices is software documentation. However, even at present many systems are still developed and released without proper documentation [8]. In response, several methodologies have been developed to address the unsolved problem of the documentation lag [9]. The Agile Manifesto, for example, puts greater emphasis on the developed product while ignoring detailed documentation. This methodology welcomes change and stresses fast delivery of useful software, based on the close collaboration between developers and customers.

Software documentation is a general term that refers to two types of documentation: 1) documenting the user requirements that provide the basis for designing the system to be developed, and 2) documenting the software to be developed, or that was developed, for aiding development or future maintenance activities. These maintenance activities, according to many studies, are the most expensive part in the software development life-cycle [10]. The Agile methodology is helpful in reducing the amount of work needed during the first documentation process (requirement elicitation and project development). However, for future maintenance of the developed software, proper documentation is still required. For that reason, the documentation required to the Agile methodology is done at the end of the project and remains inadequate for maintenance purposes [11]. For years educators and practitioners have stressed the importance of documentation (during the design phase or after project completion), however many projects are still released without proper maintenance documentation.

### 2.1 Software Maintenance

A common definition for maintenance is that it is performed after product delivery. Software maintenance, as well, refers to the activities carried out after the development's completion. However, software maintenance is very different from "ordinary" equipment maintenance. While in other engineering disciplines maintenance in intended to fix a problem [12] and keep the equipment running so it will continue to provide the original functionality, software maintenance, in many cases is required to enhance the functionality based on the ever changing requirements due to the operational environment, the competition, and the business climate. Meeting these new and changing requirements is unique and basic software characteristic, as defined in Lehman's laws of software evolution [13,14]. Furthermore, software is constantly being modified to utilize new hardware equipment and for integration into new environments.

Introduction of the software development life-cycle has led several researchers to consider activities related to software maintenance, which are initialized while the software is being developed and not only after delivery. Additionally, some researchers emphasize that starting the maintenance activities after completing development leads to an unnecessarily more complex and costly task [15,16]. Others define software maintenance as a mix of activities, some performed after delivery and some performed during development. The pre-delivery activities include the necessary planning for the post-delivery activities [17].

### 2.2 Students' Perception Regarding Maintenance

There has been a great deal of improvement in software development over the last decade. Many new techniques, methodologies, languages and tools have been created to advance the development processes. Software maintenance, however, lags behind mainly due to its reactive nature. Introducing systemic approaches to software maintenance is inherently problematic [18]. The required software maintenance (error corrections or introductions of new features) cannot be postponed or circumvented. By nature software maintenance is a disorganized process which deteriorates the software's architecture (Lehman's second law of software evolution [13]). This deterioration is due in part to missing knowledge which is required for maintenance. In addition, any changes introduced deteriorate the system architecture further, making future maintenance even more difficult. The lack of correct and updated documentation is one of the main causes for this missing knowledge.

During their first and second years of study, students become acquainted with these facts, however in spite of the lecturers' efforts; software documentation continues to

be insufficient. More troubling is the fact that students do not assimilate the need for, and importance of, proper documentation [3]. Many students consider the development stage as the most important activity in the software development life cycle, totally ignoring the fact that successful software will have to be maintained for a long time.

### 2.3 Peer Review in Higher Education

Peer review is a form of external evaluation carried out by professional colleagues [19]. Peers can be experts in the field but can also be classmates who poses the same level of knowledge and assess the work of fellow students. Peer review is a widely practiced form of certifying quality in higher education [20], and has been described as a formative evaluation process in which participants work collaboratively to strengthen a product [21]. Peer review is generally said to encourage critical examination, promote the exchange of ideas, reduce non-academic interference, guide academic discourse, and reinforce academic values [22]. Peer review assumes the existence of norms by which a peer's work may be judged. Through critical examination, norms are used to compare a peer's work to accepted practices. If a peer's work deviates significantly from accepted norms, then an attempt to correct it will likely occur. Being aware of the advantages of peer review, it has been incorporated as an integral part of the workshop. The inter-team and intra-team peer review was used to enhance the students' learning abilities and to enforce critical thinking. However, In addition to the common peer review, the workshop requires the students not only to evaluate and assess their peers work, but also build on it. The students' success in performing their assignments depends on their ability to understand the work or their peers. This elevates the assessment process to a new and more important level.

## 3. The Study

In what follows we discuss the study performed while addressing the participating students' the workshop's structure including the assignments and the grading scheme.

### 3.1 About the Study Participants

The workshop is a mandatory course taken during the second year of study. A total of twenty-six college students participated in the present study. In the workshop the students were divided into seven teams (five teams of four students and two teams of three students). At this stage the students have already learned software modeling, UML usage, etc. In addition to the standard topics of the software engineering course, one of the workshop's important objectives is to prepare the students for their Final Project and the real world challenges they will face.

### 3.2 The Course

The systems engineering workshop's general objectives are to introduce software development life cycle concepts to the students while enhancing their understanding of documentation and product maintainability. Since software is considered one of the most complex systems produced by humans [23], students have to adopt proper working procedures for lowering the development risks and the high maintenance barriers. Documenting their ideas and thoughts during the design phase is crucial for future understanding of the software to be developed.

Other objectives relate to 1) practical understanding of the software development stages required for development of a modern Information System; 2) implementing these stages in a small project; 3) understanding the problems associated and caused by working in teams, and 4) developing the required "soft skills" (critical thinking, team work, interpersonal relationships, etc). For that reason, the workshop augments knowledge and understanding gained in current and previous courses, and is practical, "hands-on," and team-based.

All seven teams received and worked on an identical project. The project was a general description of a required system that was to be developed (an Internet based electronic auctions, or e-bidding system). As part of the first assignment, the students had to study the existing systems, address and assess various alternatives, and suggest ways (and a software based system) of providing an agreed upon functionality. The workshop structure followed the software development life-cycle and was based on three incremental assignments.

Each assignment required personal and individual work followed by team activities (in person or by using various collaborative tools).The students had four weeks for each assignment throughout the process the students consulted their instructor (via email, the workshop web site, and personal meetings) on various issues related to their assignment. In order to reinforce the importance of documentation and maintainability, the teams were engaged in role-based development in which the teams shared all responsibility for their success. While a specific project was designed by one team, developed by another team and tested by a third team, in the end each team had to work not only on the three stages of the assignment but on each one of the three design solutions (**Figure 1**). This way, each team was involved in developing a system designed by another team while trying to understand some of the undocumented intentions expressed in the design. This forced them to seek help from the designing team. On the other hand, this developing team had to help another team that was trying to develop the system based on their design document. The interdependence of these stages was stressed and made apparent to all teams. This workshop structure was designed to enhance the students' under-

standing regarding the importance of documentation through their own experience.

**Figure 1** depicts the workshop's structure. The long horizontal rectangles represent the seven versions of the same project, while the three vertical columns represent the assignments (A1, A2 and A3). As can be seen, each project consists of the three assignments performed by three different teams. Each team, on the other hand, worked on all three assignments, each one belonging to a different project.

The workshop requirements included two types of deliverables: 1) team assignments, and 2) a personal assignment.

### 3.2.1 Team Assignments

The software development life-cycle activities were divided into three team based assignments: 1) project definition and design; 2) project development, and 3) project testing.

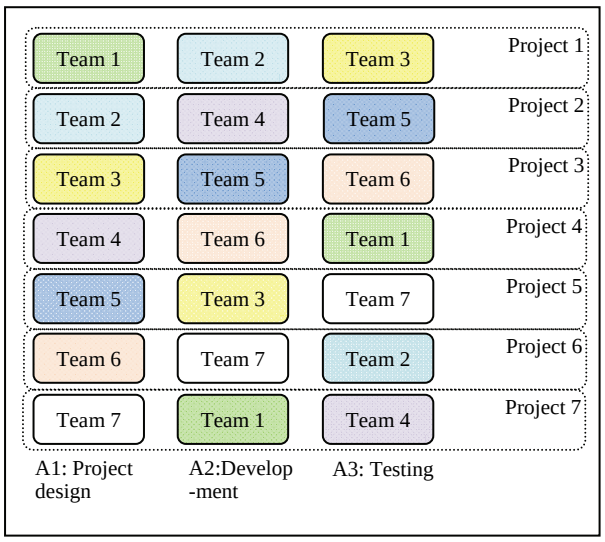



**Figure 1. The Workshop's Structure**

#### 3.2.1.1 Project Definition and Design

The first assignment started with a very brief description of the project, the functionality and the required development activities. The students studied available e-bidding systems, documenting their functionality, and used them as a basis for the system they were required to develop. Since such a large project cannot be completed during the semester, the students had to identify at least five different users to be supported by the system and for each user a set of Use-Cases had to be defined. In addition to the Sequence Diagrams supporting these Use-Cases, the students had to define the non-functional requirements associated with these Use-Cases. The system analysis phase (which is part of this assignment) included a high level design (System architecture and the Class Diagram) as well as a detailed design (Activity Diagram followed by a Program Design Language definition for the described functionality). All these activities required a great deal of individual work as well as collaborative work in which each student assessed and approved the work performed by other team members.

#### 3.2.1.2 Project Development

The second assignment consisted of the development of the system according to the Project Definition and design document (the first assignment). However, instead of developing the system according to their own design, each team had to develop the system as it was defined by another team. The developing team had to carefully follow the document they received, ignoring all their prior knowledge or ideas they may have expressed in their first assignment. Small code modifications were permitted, provided that the definition in the document they received was erroneous and could not be implemented. After completing the development, each team had to compile a 'difference' document, outlining the changes between the implementation and the document as received, with special emphasis placed on the reason behind these changes. At this stage stress is not placed on enhancing the product to be developed, but rather on developing it according to the exact specifications outlined in the definition document. An additional document which was part of this assignment was a short evaluation of the first assignment's quality as it was reflected in the implementation. The last document to be submitted as part of this assignment was a Unit Test Plan for each of the methods developed.

#### 3.2.1.3 Project Testing

The third assignment consists mainly of the testing phase. The students have to implement the Unit Test Plan as was designed by the previous team. Due to time constraints the workshop addresses only part of the required project development, so for testing the software pieces developed by the previous team, the testing team had to include additional developments (a test generator and a stub). These additional developments were required for building the testing infrastructure for the developed software pieces. As part of this assignment the team is required to correct mistakes that were discovered during the testing. The corrected code has to be tested once again. This process is repeated until everything runs according to the specifications, as outlined in the project definition document (the first assignment of this project). This third assignment also includes the testing report that summarizes the problems discovered and their corrections. In addition, this assignment includes a system test plan with at least ten detailed test cases. This plan is for the Quality Assurance staff, so it has to be detailed and based on the system functionality as derived from the project definition document (first assignment). The last part of this assignment is a quality test plan that concentrates on the non-functional attributes of the system, with a special

emphasis on the metrics to be used or defined.

### 3.2.2 Personal Assignment

The personal assignment is mainly an activity summary report, in which each student describes 1) the work done during every stage of the project; 2) his/her part in these activities; 3) the problems they (as a team) encountered during the project and 4) the problems he/she encountered personally. There is also a short reflection on the workshop, as well as a one sentence summary about the workshop's results. The last part reflects on the work distribution among the team members (100 points that the student divides between the other team members to express their relative contribution toward each of the three assignments).

### 3.3 The Workshop Grading Scheme

Since one of the important workshop goals is to strengthen team work, most of the grades are based on the team's activities. Each of the first two assignments makes up 33% of the grade, while the third, which is simpler, comprises 24%. The personal report, including the short reflection, contributes an additional 10%.

The grading scheme took into consideration the work distribution as was described by each team member. Five points (out of the 90 points allocated for team activities) were used as floating points among the team members, based on their average contribution to the team's success

## 4. Learning Process Evaluation Methodology

The evaluation method included a comparison between two questionnaires. The first questionnaire was part of a survey conducted during the workshop's first lecture, in which students were asked to rate their perception regarding the relative importance of the three project phases expressed by the planned assignments. A similar survey was conducted during the last lecture producing the second questionnaire. Since the end of semester questionnaire was identical to the questionnaire used in the first lecture, its intention was to measure the workshop's influence regarding the perceived importance of the three phases and especially the importance of documentation and testing on the software engineering activities. In addition, the evaluation process analyzed the student's reflections on their workshop experiences.

For implementing a successful inter-team role play, the workshop was highly structured. In addition, pre-defined templates were used for all the team based assignments. However, in contrast to these pre-defined templates the personal reports were composed of free style answers. The only data provided were the points to be addressed in these reflections. This open format encouraged students to concentrate on the issues s/he felt were important and offered a better understanding of the students' achievements during the workshop.

## 5. Results and Discussion

In what follows we present data and discuss the effect of the workshop's structure on the students' perceptions regarding the importance of documentation on the project's success, as well as their reflections regarding the benefits of the workshop.

### 5.1 The Assignment's Relative Importance

The first questionnaire results are outlined by **Figure 2.** It is no surprise that CS students regarded development as the most important activity (70% of the project). Testing was perceived to be of secondary importance (16%) and documenting the design phase was perceived to be the least important (14%) in the design and development of software.

The students explained the relatively low importance of documentation by citing the fact that the methodology and the tools used (UML – Unified Modeling Language as well as the code itself) provide all the necessary documentation. The results obtained in this survey were no different when compared to the results from the previous year. These consistent results were the trigger for the workshop structure and one of its objectives was to convince students, through their own practical experience about the importance of documentation.

As was demonstrated by the first questionnaire (**Figure 2**), most students view development as the most important activity of a project (70%). However, the second questionnaire revealed that the students began to realize the importance of the subsequent components and the role they play in determining the project's success. Therefore, by the semester's end, development's perceived importance was reduced by 31% (to 48% of the project), while the relative importance of the documentation assignment increased by 59% (from 14% in the first questionnaire to 23% in the second questionnaire). Testing's perceived importance also increased - by 83% (from 16% in first questionnaire to 29% in the second).

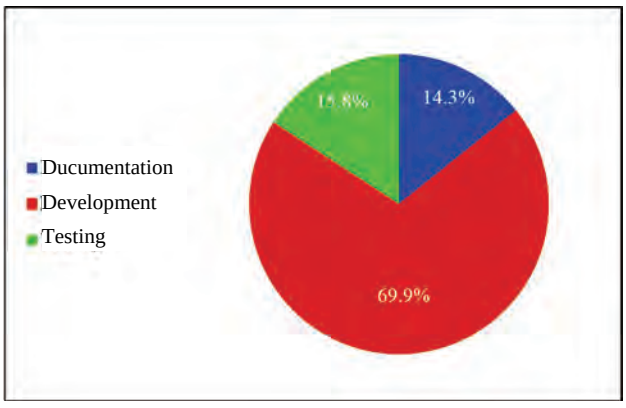



**Figure 2. Relative Assignment Importance (1st Lecture)**

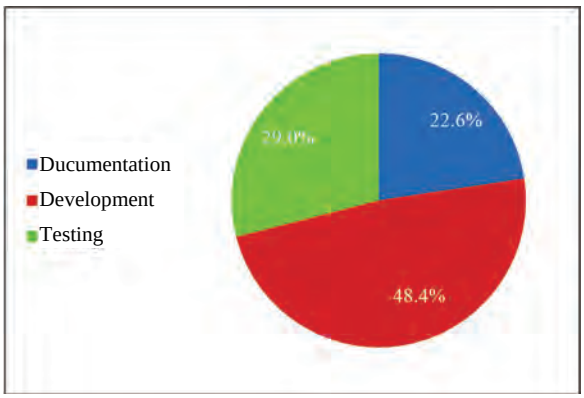



**Figure 3. Relative Assignment Importance (Last Lecture)**

There are many factors affecting the relative importance of the various life cycle stages and the amount of time required for each one. These factors, for example may include customer requirements, project type, the software development life cycle methodology, the programming languages and CASE tools, etc. In most cases the development stage requires less than 30% of the project estimated time. Glass [24] uses 20% for requirements elicitation, 20% for design, 20% for coding and 40% for testing. The requirements elicitation is not part of this workshop since the project and its general requirements were predefined and the students had to study available solutions and decide which parts to design and implement. At the end of the semester, the students still regard coding (development) as the most important component, but it is significantly (31%) less than its importance at the beginning of the semester. The end of semester percentages are closer to the numbers used by researchers and practicing software engineers.

The change in the students' perception regarding the relative importance of the various project components is directly linked to the workshop's structure. There were many instances during the second stage of the workshop, in which the students were trying to drop the design specifications they received from the previous team, claiming these specifications will not produce a viable solution and the project cannot be developed. In all cases it proved to be wrong. The solutions described in these design specifications provided a workable solution, however they were not properly documented, which hampered the students understanding. After discussing the design specifications with the responsible team, the project was developed as intended, with some minor modifications. This misunderstanding was repeated on the third stage, in which students had to design and execute the system testing. However, for designing the test environment and the test scenarios a full project understanding was required. In addition to the extra work needed due to the missing documentation it also changed the students' perception regarding the importance of the non-development activities.

The significant increase in the perceived testing importance (83%) can be explained by the fact that during the third stage the student had to design and develop the stub and the scenarios for the system to be checked. In all cases where the system did not function according to the specifications, the testing team had to correct the code and run it once again. This means that the testing team had to be familiar with the design specifications as well as with the developed code. The testing students acted as the "gate-keepers" making sure that only the fully functional system is released. Performing this task properly, in the workshop, requires some additional analytical skills for finding and correcting various bugs which may have been introduced during the development or the design stages. Unlike the real world situation, where in case of problems, Quality Assurance people usually return the system to the developers, here the testing team had to fix it by themselves, which led to their higher appreciation of the testing task and its elevated importance in the development process.

## 5.2 The Student's Perspective

Analysis of the students' reflections revealed emphasis of three main issues: 1) the importance of documentation; 2) team-based activities and 3) contribution to future vocation.

### 5.2.1 The Importance of Documenting the Project

Improving the students' understanding regarding documentation and the role it plays in the project and its future maintainability, was addressed by many of the reflections. For example:

> "I understood (unfortunately through bad experience) the importance of a development project's documentation."
>
> "It was only during the workshop that I began to grasp the importance of understanding and documenting the requirements."

From the above students' excerpts we can conclude that they developed a sense of appreciation for documentation, mostly arising from the need to spend many more of their own resources when it was missing. Furthermore, without proper documentation, the project may not be successful and might not deliver the expected outcome. The fact that they realized, for example, that undocumented specifications are misleading is consistent with Williams [25] stressing that students no longer view the teaching staff as their sole conduit of technical information.

### 5.2.2 Team-Based Activities and Implications

Students pointed out several advantages regarding their experience of working in teams, as well as what was required of them. Here are some common reflections:

> "Working in a team provided me with many

new views and possibilities for solving the problem."

"The most important lesson I learned during the workshop was to accept my friends' criticism and provide constructive feedback."

"The success and failure of the project depended mainly on the team members' activities and not on any single member."

From these reflections we learn that in general students found the teamwork method helpful in developing their critical thinking (receiving and providing constructive feedback) and in improving their ability to cooperate. However, in their reflections students also pointed out the shortcomings they experienced in team-based activities. For example:

"Team work can be a blessing, but sometimes it can also be a curse…"

#### 5.2.3 The Workshop's Contribution to Future Vocation

Here are some student reflections regarding the contribution of the workshop's assignments to their future employment.

"So far we learned that the most important stage in the project is the development. Here I understood that the process is equally important."

We conclude that the students found the detailed documentation very helpful. Furthermore, the understanding gained by working in teams helped them think as developers and enhanced the process of reaching the problem solution.

## 6. Concluding Remarks

From the students' reflections and the results received regarding the differences in their perception as reflected in the two questionnaires, it can be concluded that the workshop raised the students' levels of understanding [26], and as a result helped them cope successfully with the given workshop assignments. The role-based development, in which the students had to assume responsibility for activities partially performed by others, exposed them to ideas which were different from the ones they had decided to use in their own solutions. Especially, this workshop structure effected the students' appreciation of documentation and the role it played in their own success. This exposure, in many cases, made them rethink their task and prompted them to look for better, more efficient solutions. The collaborative team work exposed each team member to various ideas expressed by his/her peers and as a result caused additional thinking about available solution alternatives.

## REFERENCES

[1] P. Naur and B. Randell, "Software Engineering: Report of a Conference Sponsored by the NATO Science Committee," Garmisch, Germany, Scientific Affairs Division, NATO, 1968. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

[2] R. J. Veldwijk, M. Boogaard and E. R. K. Spoor, "Assessing the Software Crisis-Why Information Systems are Beyond Control," Vrije Universiteit, the Netherlands, 1992. ftp://zappa.ubvu.vu.nl/19920006.pdf

[3] J. Burge, "Exploiting Multiplicity to Teach Reliability and Maintainability in a Capstone Project," 20*th Conference on Software Engineering Education & Training*, Dublin, 2007.

[4] L. T. Cheng, S. Hupper, S. Ross and J. Patterson, "Jazzing up Eclipse with Collaborative Tools," *Proceedings of the* 2003 *OOPSLA Workshop on Eclipse Technology eXchange*, Anaheim, October 2003.

[5] S. C. Souza, N. Anquetil and K. M. Oliveira, "Which Documentation for Software Maintenance," *Journal of the Brazilian Computer Society*, Vol. 13, No. 2, 2007, pp. 31-44.

[6] S. Das, W. G. Lutters and C. B. Seaman , "Understanding Documentation Value in Software Maintenance," *Proceedings of the* 2007 *Symposium on Computer human interaction for the management of information technology*, Cambridge, Massachusetts, 30-31 March 2007.

[7] M. K. Mattsson, "Problems in Agile Trenches," *Proceedings of the Second ACM-IEEE international symposium on Empirical Software Engineering and Measurement*, Kaiserslautern, 09-10 October 2008.

[8] G. T. Daich, "Document Diseases and Software Malpractice". http://www.sstc.online.org/Proceedings/2003/PDFFiles/p961pap.pdf

[9] P. Clements, "Comparing the SEI's Views and Beyond Approach for Documenting Software Architectures with ANSI-IEEE 1471-2000," *Technical Note CMU/SEI*-2005-*TN*-017.http://www.sei.cmu.edu/pub/documents/05.eports/ pdf/05tn017.pdf

[10] R. C. Seacord, D. Plakosh and G. A. Lewis, "Modernizing Legacy Systems–Software Technologies, Engineering Processes and Business Practices," New York, Addison-Wesley, 2003.

[11] D. Brolund and Ohlrogge, "Streamlining the Agile Documentation Demonstration for the XP 2006 Conference," *Lecture Notes in Computer Science*, Springer, Vol. 4044, 2006, pp. 215-216.

[12] G. Canfora and A. Cimitile, "Software Maintenance." http://www.compaid.com/caiInternet/ezine/maintenance-canfora.pdf

[13] M. M. Lehman, "Lifecycles and the Laws of Software Evolution," *Proceedings of the IEEE, Special Issue on Software Engineering*, Vol. 68, No. 9, 1980, pp. 1060-1076.

[14] M. M. Lehman, "Program Evolution," *Journal of Information Processing Management*, Vol. 19, No. 1, 1984, pp. 19-36.

[15] N. F. Schneidewind, "The State of Software Maintenance," *IEEE Transactions on Software Engineering*, Vol. 13, No.

3, 1987, pp. 303-310.

[16] W. M. Osborne and E. J. Chikofsky, "Fitting Pieces to the Maintenance Puzzle," *IEEE Software*, Vol. 7, No. 1, 1990 pp. 11-12.

[17] T. M. Pigoski, "Practical Software Maintenance–Best Practices for Managing Your Software Investment," Wiley & Sons, New York, 1997.

[18] M. B. G. Dias, N. Anquetil and K.M. de Oliveira, "Organizing the Knowledge Used in Software Maintenance," *Journal of Universal Computer Science*, Vol. 9, No. 7, 2003, pp. 641-658.

[19] A. Yadin and I. Lavy, "Integrated Formative Assessment as a Vehicle towards Meaningful Learning in Systems Analysis and Design workshop," Paris, 2008.

[20] C. Herndon, "Peer Review and Organizational Learning: Improving the Assessment of Student Learning," *Research & Practice in Assessment*, Vol. 1, No. 1, 2006, pp. 1-7.

[21] L. Keig and M. D. Waggoner, "Collaborative Peer Review: Role of Faculty in Improving College Teaching," *ASHE-ERIC Higher Education Report*, Washington DC, Vol. 23, No. 2, 1994.

[22] K. Berkencotter, "The Power and Perils of Peer Review," *Rhetoric Review*, Vol. 13, No. 2, 1995, pp. 245-248.

[23] M. Lenic, M. Zorman, P. Povalej and P. Kokol, "Alternative Measurement of Software Artifacts," *ICCC* 2004 *Second IEEE International Conference on Computational Cybernetics*, Wien, 2004, pp. 231-235.

[24] R. Glass, "Facts and Fallacies of Software Engineering," Addison-Wesley, New York, 2003.

[25] L. Williams, "In Support of Student Pair-Programming," *Proceedings of the* 32*nd SIGCSE technical symposium on Computer Science Education*, Charlotte, 2001.

[26] J. Biggs, "Enhancing Teaching Through Constructive Alignment," *Higher Education*, Vol. 32, No. 3, 1996, pp. 347-364.

*J. Software Engineering & Applications*, 2010, 3: 503-509
doi:10.4236/jsea.2010.35057 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# A Line Search Algorithm for Unconstrained Optimization*

**Gonglin Yuan[1], Sha Lu[2], Zengxin Wei[1]**

[1]College of Mathematics and Information Science, Guangxi University, Nanning, China; [2]School of Mathematical Science, Guangxi Teachers Education University, Nanning, China.
Email: glyuan@gxu.edu.cn

Received February 6th, 2010; revised March 30th, 2010; accepted March 31st, 2010.

## ABSTRACT

*It is well known that the line search methods play a very important role for optimization problems. In this paper a new line search method is proposed for solving unconstrained optimization. Under weak conditions, this method possesses global convergence and R-linear convergence for nonconvex function and convex function, respectively. Moreover, the given search direction has sufficiently descent property and belongs to a trust region without carrying out any line search rule. Numerical results show that the new method is effective.*

***Keywords*:** *Line Search, Unconstrained Optimization, Global Convergence, R-linear Convergence*

## 1. Introduction

Consider the unconstrained optimization problem

$$\min_{x\in R^n} f(x), \tag{1}$$

where $f: R^n \to R$ is continuously differentiable. The line search algorithm for (1) often generates a sequence of iterates $\{x_k\}$ by letting

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0,1,2,\cdots \tag{2}$$

where $x_k$ is the current iterate point, $d_k$ is a search direction, and $\alpha_k > 0$ is a steplength. Different choices of $d_k$ and $\alpha_k$ will determine different line search methods [1-3]. The method is divided into two stages at each iteration: 1) choose a descent search direction $d_k$; 2) choose a step size $\alpha_k$ along the search direction $d_k$.

Throughout this paper, we denote $f(x_k)$ by $f_k$, $\nabla f(x_k)$ by $g_k$, and $\nabla f(x_{k+1})$ by $g_{k+1}$, respectively. $\|\cdot\|$ denotes the Euclidian norm of vectors.

One simple line search method is the steepest descent method if we take $d_k = -g_k$ as a search direction at every iteration, which has wide applications in solving large-scale minimization problems [4]. However, the steepest descent method often yields zigzag phenomena in solving practical problems, which makes the algorithm converge to an optimal solution very slowly, or even fail to converge [5,6]. Then the steepest descent method is not the fastest one among the line search methods.

If $d_k = -H_k g_k$ is the search direction at each iteration in the algorithm, where $H_k$ is an $n \times n$ matrix approximating $[\nabla^2 f(x_k)]^{-1}$, then the corresponding line search method is called Newton-like method [4-6] such as Newton method, quasi-Newton method, variable metric method, etc. Many papers [7-10] have been proposed by the method for optimization problems. However, one drawback of the Newton-like method is required to store and compute matrix $H_k$ at each iteration and thus adds the cost of storage and computation. Accordingly, this method is not suitable to solve large-scale optimization problems in many cases.

The conjugate gradient method is a powerful line search method for solving the large scale optimization problems because of its simplicity and its very low memory requirement. The search direction of the conjugate gradient method often takes the form

$$d_k = \begin{cases} -g_k + \beta_k d_k, & if\ k \geq 1 \\ -g_k, & if\ k = 0, \end{cases} \tag{3}$$

where $\beta_k \in R$ is a scalar which determines the different

* This work is supported by China NSF grants 10761001, the Scientific Research Foundation of Guangxi University (Grant No. X081082), and Guangxi SF grants 0991028.

conjugate gradient methods [11-13] etc. The convergence behavior of the different conjugate gradient methods with some line search conditions [14] has been widely studied by many authors for many years (see [4, 15]). At present, one of the most efficient formula for $\beta_k$ from the computation point of view is the following PRP method

$$\beta_k^{PRP} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{\| g_k \|^2}. \quad (4)$$

If $x_{k+1} \to x_k$, it is easy to get $\beta_k^{PRP} \to 0$, which implies that the direction $d_k$ of the PRP method will turn out to be the steepest descent direction as the restart condition automatically when the next iteration point is approximate to the current point. This case is very important to ensure the efficiency of the PRP conjugate gradient method (see [4,15] etc.). For the convergence of the PRP conjugate gradient method, Polak and Ribière [16] proved that the PRP method with the exact line search is globally convergent when the objective function is convex, and Powell [17] gave a counter example to show that there exist nonconvex functions on which the PRP method does not converge globally even the exact line search is used.

We all know that the following sufficient descent condition

$g_k^T d_k \le -c \| g_k \|^2$, for all $k \ge 0$ and some constant

$$c > 0 \quad (5)$$

is very important to insure the global convergence of the algorithm by nonlinear conjugate gradient method, and it may be crucial for conjugate gradient methods [14]. It has been showed that the PRP method with the following strong Wolfe-Powell (SWP) line search rules which is to find the step size $\alpha_k$ satisfying

$$f(x_k + \alpha_k d_k) \le f_k + \sigma_1 \alpha_k g_k^T d_k, \quad (6)$$

and

$$| g(x_k + \alpha_k d_k)^T d_k | \le \sigma_2 | g_k^T d_k | \quad (7)$$

did not ensure the condition (5) at each iteration, where $0 < \sigma_1 < \frac{1}{2}$, $\sigma_1 < \sigma_2 < 1$. Then Grippo and Lucidi [18] presented a new line search rule which can ensure the sufficient descent condition and established the convergence of the PRP method with their line search technique. Powell [17] suggested that $\beta_k$ should not be less than zero. Considering this idea, Gilbert and Nocedal [14] proved that the modified PRP method $\beta_k^+ = \max\{0, \beta_k^{PRP}\}$ is globally convergent under the sufficient descent assumption condition and the following weak Wolfe-Powell (WWP) line search technique: find the steplength $\alpha_k$ such that (6) and

$$g(x_k + \alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k. \quad (8)$$

Over the past few years, much effort has been put to find out new formulas for conjugate methods such that they have not only global convergence property for general functions but also good numerical performance (see [14,15]). Resent years, some good results on the nonlinear conjugate gradient method are given [19-25].

These observations motivate us to propose a new method which possesses not only the simplicity and low memory but also desirable theoretical features. In this paper, we design a new line search method which possesses not only the sufficiently descent property but also the following property

$\| d_k \| \le c_1 \| g_k \|$, for all $k \ge 0$ and some constant

$$c_1 > 0 \quad (9)$$

whatever line search rule is used, where the property (9) implies that the search direction $d_k$ is in a trust region automatically.

This paper is organized as follows. In the next section, the algorithms and other line search rules are stated. The global convergence and the R-linear convergence of the new method are established in Section 3. Numerical results and one conclusion are presented in Section 4 and in Section 5, respectively.

## 2. The Algorithms

Besides the inexact line search techniques WWP and SWP, there exist other line search rules which are often used to analyze the convergence of the line search method:

1) The exact minimization rule. The step size $\alpha_k$ is chosen such that

$$f(x_k + \alpha_k d_k) = \min_{\alpha \ge 0} f(x_k + \alpha d_k). \quad (10)$$

2) Goldstein rule. The step size $\alpha_k$ is chosen to satisfy (6) and

$$f(x_k + \alpha_k d_k) \ge f_k + \sigma_2 \alpha_k g_k^T d_k. \quad (11)$$

Now we give our algorithm as follows.

1) Algorithm 1 (New Algorithm)

Step 0: Choose an initial point $x_0 \in R^n$, and constants $0 < \varepsilon < 1$, $0 < \sigma_1 < \frac{1}{2}$, $\sigma_1 < \sigma_2 < 1$. Set $d_0 = -g_0 = -\nabla f(x_0)$, $k := 0$.

Step 1: If $\| g_k \| \le \varepsilon$, then stop; Otherwise go to step 2.

Step 2: Compute steplength $\alpha_k$ by one line search technique, and let $x_{k+1} = x_k + \alpha_k d_k$.

Step 3: If $\| g_{k+1} \| \le \varepsilon$, then stop; Otherwise go to step 4.

Step 4: Calculate the search direction $d_{k+1}$ by (3), where $\beta_k$ is defined by (4).

Step 5**:** Let $d_{k+1}^{new} = d'_{k+1} + \min\{0, -\frac{g_{k+1}^T d'_{k+1}}{\| g_{k+1} \|^2}\} g_{k+1} - g_{k+1}$, where $d'_{k+1} = \frac{\| y_k^* \| \| g_{k+1} \|}{\| s_k \| \| d_{k+1} \|} d_{k+1}$, $s_k = x_{k+1} - x_k$, $\| y_k^* \| = \max\{\| s_k \|, \| y_k \|\}$, $y_k = g_{k+1} - g_k$.

Step 6: Let $d := d_{k+1}^{new}$, $k := k+1$, and go to step 2.

Remark. In the Step 5 of Algorithm 1, we have

$$\frac{\| y_k^* \|}{\| s_k \|} = \frac{\max\{\| s_k \|, \| y_k \|\}}{\| s_k \|} \geq 1,$$

which can increase the convergent speed of the algorithm from the computation point of view.

Here we give the normal PRP conjugate gradient algorithm and one modified PRP conjugate gradient algorithm [14] as follows.

2) Algorithm 2 (PRP Algorithm )

Step 0: Choose an initial point $x_0 \in R^n$, and constants $0 < \varepsilon < 1$, $0 < \sigma_1 < \frac{1}{2}$, $\sigma_1 < \sigma_2 < 1$. Set $d_0 = -g_0 = -\nabla f(x_0)$, $k := 0$.

Step 1: If $\| g_k \| \leq \varepsilon$, then stop; Otherwise go to step 2.

Step 2: Compute steplength $\alpha_k$ by one line search technique, and let $x_{k+1} = x_k + \alpha_k d_k$.

Step 3: If $\| g_{k+1} \| \leq \varepsilon$, then stop; Otherwise go to step 4.

Step 4: Calculate the search direction $d_{k+1}$ by (3), where $\beta_k$ is defined by (4).

Step 5: Let $k := k+1$ and go to step 2.

3) Algorithm 3 (PRP$^+$ Algorithm see [14])

Step 0: Choose an initial point $x_0 \in R^n$, and constants $0 < \varepsilon < 1$, $0 < \sigma_1 < \frac{1}{2}$, $\sigma_1 < \sigma_2 < 1$. Set $d_0 = -g_0 = -\nabla f(x_0)$, $k := 0$.

Step 1: If $\| g_k \| \leq \varepsilon$, then stop; Otherwise go to step 2.

Step 2: Compute steplength $\alpha_k$ by one line search technique, and let $x_{k+1} = x_k + \alpha_k d_k$.

Step 3: If $\| g_{k+1} \| \leq \varepsilon$, then stop; Otherwise go to step 4.

Step 4: Calculate the search direction $d_{k+1}$ by (3), where $\beta_k = \max\{0, \beta_k^{PRP}\}$

Step 5: Let $k := k+1$ and go to step 2.

We will concentrate on the convergent results of Algorithm 1 in the following section.

## 3. Convergence Analysis

The following assumptions are often needed to analyze the convergence of the line search method (see [15,26]).

Assumption A (i) $f$ is bounded below on the level set $\Phi = \{x \in R^n : f(x) \leq f(x_0)\}$;

Assumption A (ii) In some neighborhood $\Phi_0$ of $\Phi$, $f$ is differentiable and its gradient is Lipschitz continuous, namely, there exists a constants $L > 0$ such that $\| g(x) - g(y) \| \leq L \| x - y \|$, for all $x, y \in \Phi_0$.

In the following, let $g_k \neq 0$ for all $k$, for otherwise a stationary point has been found.

**Lemma 3.1** *Consider Algorithm* 1. *Let Assumption* (ii) *hold. Then* (5) *and* (9) *hold.*

**Proof.** If $k = 0$, (5) and (9) hold obviously. For $k \geq 1$, by Assumption (ii) and the Step 5 of Algorithm 1, we have

$$\| d_{k+1}^{new} \| \leq \| d'_{k+1} \| + \| d'_{k+1} \| + \| g_{k+1} \| \leq (2\max\{1, L\} + 1) \| g_{k+1} \|.$$

Now we consider the vector product $g_{k+1}^T d'_{k+1}$ in the following two cases:

case 1. If $g_{k+1}^T d'_{k+1} \leq 0$. Then we get

$$\begin{aligned} g_{k+1}^T d_{k+1}^{new} &= g_{k+1}^T d'_{k+1} + \min\{0, -\frac{g_{k+1}^T d'_{k+1}}{\| g_{k+1} \|^2}\} \| g_{k+1} \|^2 - \| g_{k+1} \|^2 \\ &= g_{k+1}^T d'_{k+1} - \| g_{k+1} \|^2 \\ &\leq -\| g_{k+1} \|^2. \end{aligned}$$

case 2. If $g_{k+1}^T d'_{k+1} > 0$. Then we obtain

$$\begin{aligned} g_{k+1}^T d_{k+1}^{new} &= g_{k+1}^T d'_{k+1} + \min\{0, -\frac{g_{k+1}^T d'_{k+1}}{\| g_{k+1} \|^2}\} \| g_{k+1} \|^2 - \| g_{k+1} \|^2 \\ &= g_{k+1}^T d'_{k+1} - \frac{g_{k+1}^T d'_{k+1}}{\| g_{k+1} \|^2} \| g_{k+1} \|^2 - \| g_{k+1} \|^2 \\ &= -\| g_{k+1} \|^2. \end{aligned}$$

Let $c \in (0,1)$, $c_1 = 2\max\{1, L\} + 1$ and use the Step 6 of Algorithm 1, (5) and (9) hold, respectively. The proof is completed.

The above lemma shows that the search direction $d_k$ has such that the sufficient descent condition (5) and the condition (9) without any line search rule.

Based on Lemma 3.1, Assumption (i) and (ii), let us give the global convergence theorem of Algorithm 1.

**Theorem 3.1** *Let* $\{\alpha_k, d_k, x_{k+1}, g_{k+1}\}$ *be generated by Algorithm* 1 *with the exact minimization rule, the Goldstein line search rule, the SWP line search rule, or the WWP line search rule, and Assumption* (i) *and* (ii) *hold. Then*

$$\lim_{k \to \infty} \| g_k \| = 0 \tag{12}$$

*holds.*

**Proof.** We will prove the result of this theorem with the exact minimization rule, the Goldstein line search rule, the SWP line search rule, and the WWP line search rule, respectively.

1) For the exact minimization rule. Let the step size $\alpha_k$ be the solution of (10).

By the mean value theorem, $g_k^T d_k < 0$, and Assumption (ii), for any

$$\alpha_k^* \in \left[ \frac{1}{5L} \frac{|g_k^T d_k|}{\| d_k \|^2}, \frac{2}{5L} \frac{|g_k^T d_k|}{\| d_k \|^2} \right],$$

we have

$$\begin{aligned} & f(x_k + \alpha_k d_k) - f(x_k) \le f(x_k + \alpha_k^* d_k) - f(x_k) \\ & = \int_0^1 g(x_k + t\alpha_k^* d_k)^T (\alpha_k^* d_k) dt \\ & = \alpha_k^* g_k^T d_k + \int_0^1 [g(x_k + t\alpha_k^* d_k) - g_k]^T (\alpha_k^* d_k) dt \\ & \le \alpha_k^* g_k^T d_k + \frac{1}{2} L \alpha_k^{*2} \| d_k \|^2 \\ & \le -\frac{1}{5L} \frac{|g_k^T d_k|}{\| d_k \|^2} (-g_k^T d_k) + \frac{1}{2} L \frac{4}{25L^2} \frac{(g_k^T d_k)^2}{\| d_k \|^4} \| d \|^2 \quad (13) \\ & = -\frac{3}{25L} \frac{(g_k^T d_k)^2}{\| d_k \|^2}, \end{aligned}$$

which together with Assumption (i), we can obtain

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\| d_k \|^2} < +\infty . \quad (14)$$

This implies that

$$\lim_{k \to \infty} \frac{(g_k^T d_k)^2}{\| d_k \|^2} = 0 \quad (15)$$

holds. By Lemma 3.1, we get (12).

2) For Goldstein rule. Let the step size $\alpha_k$ be the solution of (6) and (11).

By (11) and the mean value theorem, we have

$$-\alpha_k g(x_k + \alpha_k \tau_k d_k)^T d_k = f_k - f_{k+1} \le -\sigma_2 \alpha_k g_k^T d_k ,$$

where $\tau_k \in (0,1)$, thus

$$g(x_k + \alpha_k \tau_k d_k)^T d_k \ge \sigma_2 g_k^T d_k .$$

Using Assumption (ii) again, we get

$$\begin{aligned} & -(1-\sigma_2) g_k^T d_k \\ & \le [g(x_k + \alpha_k \tau_k d_k) - g_k]^T d_k \le \alpha_k L \| d_k \|^2 , \end{aligned}$$

which combining with (6), and use Assumption (i), we have (14) and (15), respectively. By Lemma 3.1, (12) holds.

3) For strong Wolf-Powell rule. Let the step size $\alpha_k$ be the solution of (6) and (7).

By (7), we have

$$\sigma_2 g_k^T d_k \le g(x_k + \alpha_k d_k)^T d_k \le -\sigma_2 g_k^T d_k .$$

Similar to the proof of the above case. We can obtain (12) immediately.

4) For weak Wolf-Powell rule. Let the step size $\alpha_k$ be the solution of (6) and (8). Similar to the proof of the case 3), we can also get (12).

Then we conclude this result of this theorem.

By Lemma 3.1, there exists a constant $\delta_0 > 0$ such that

$$-\frac{g_k^T d_k}{\| g_k \| \| d_k \|} \ge \delta_0, \text{ for all } k. \quad (16)$$

By the proof process of Lemma 3.1. We can deduce that there exists a positive number $\delta_1$ satisfying

$$f_k - f_{k+1} \ge \delta_1 \left( \frac{-g_k^T d_k}{\| d_k \|} \right)^2, \text{ for all } k. \quad (17)$$

Similar to the proof of Theorem 4.1 in [27], it is not difficult to prove the linear convergence rate of Algorithm 1. We state the theorem as follows but omit the proof.

**Theorem 3.2** (*see* [27]) *Based on* (16), (17), *and the condition that the function* $f$ *is twice continuously differentiable and uniformly convex on* $R^n$. *Let* $\{\alpha_k, d_k, x_{k+1}, g_{k+1}\}$ *be generated by Algorithm 1 with the exact minimization rule, the Goldstein line search rule, the SWP line search rule, or the WWP line search rule. Then* $\{x_k\}$ *converges to* $x^*$ *at least linearly, where* $x^*$ *is the unique minimal point of* $f(x)$.

## 4. Numerical Results

In this section, we report some numerical experiments with Algorithm 1, Algorithm 2, and Algorithm 3. We test these algorithms on some problem [28] taken from MATLAB with given initial points. The parameters common to these methods were set identically, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$, $\varepsilon = 10^{-6}$, In this experiment, the following *Himmeblau* stop rule is used:

If $|f(x_k)| > e_1$ ,let $stop1 = \frac{|f(x_k) - f(x_{k+1})|}{|f(x_k)|}$; Otherwise, let $stop1 = |f(x_k) - f(x_{k+1})|$, where $e_1 = 10^{-5}$. If $\| g_k \| < \varepsilon$ or $stop1 < e_2$ was satisfied, the program will be stopped, where $e_2 = 10^{-5}$.

We also stop the program if the iteration number is more than one thousand. Since the line search cannot always ensure the descent condition $d_k^T g_k < 0$, uphill search direction may occur in the numerical experiments. In this case, the line search rule maybe failed. In order to avoid this case, the stepsize $\alpha_k$ will be accepted if the

searching number is more than forty in the line search. The detailed numerical results are listed on the web site

http://210.36.18.9:8018/publication.asp?id=34402

Dolan and Moré [29] gave a new tool to analyze the efficiency of Algorithms. They introduced the notion of a performance profile as a means to evaluate and compare the performance of the set of solvers $S$ on a test set $P$. Assuming that there exist $n_s$ solvers and $n_p$ problems, for each problem $p$ and solver $s$, they defined

$t_{p,s}$ = computing time (the number of function evaluations or others) required to solve problem $p$ by solver $s$.

Requiring a baseline for comparisons, they compared the performance on problem $p$ by solver $s$ with the best performance by any solver on this problem; that is, using the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

Suppose that a parameter $r_M \geq r_{p,s}$ for all $p$, $s$ is chosen, and $r_{p,s} = r_M$ if and only if solver $s$ does not solve problem $p$.

The performance of solver $s$ on any given problem might be of interest, but we would like to obtain an overall assessment of the performance of the solver, then they defined

$$\rho_s(t) = \frac{1}{n_p} size\{p \in P : r_{p,s} \leq t\},$$

Thus $\rho_s(t)$ was the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ was within a factor $t \in R$ of the best possible ration. Then function $\rho_s$ was the (cumulative) distribution function for the performance ratio. The performance profile $\rho_s : R \in [0,1]$ for a solver was a nondecreasing, piecewise constant function, continuous from the right at each breakpoint. The value of $\rho_s(1)$ was the probability that the solver would win over the rest of the solvers.

According to the above rules, we know that one solver whose performance profile plot is on top right will win over the rest of the solvers.

In **Figures 1-3**, NA denotes Algorithm 1, PRP denotes Algorithm 2, and PRP$^+$ denotes Algorithm 3. **Figures 1-3** show that the performance of these methods is relative to $NT = NF + m * NG$, where $NF$ and $NG$ denote the number of function evaluations and gradient evaluations respectively, and $m$ is an integer. According to the results on automatic differentiation [30], the value of $m$ can be set to $m = 5$. That is to say, one gradient evaluation is equivalent to $m$ number of function evaluations if automatic differentiation is used. From these three figures

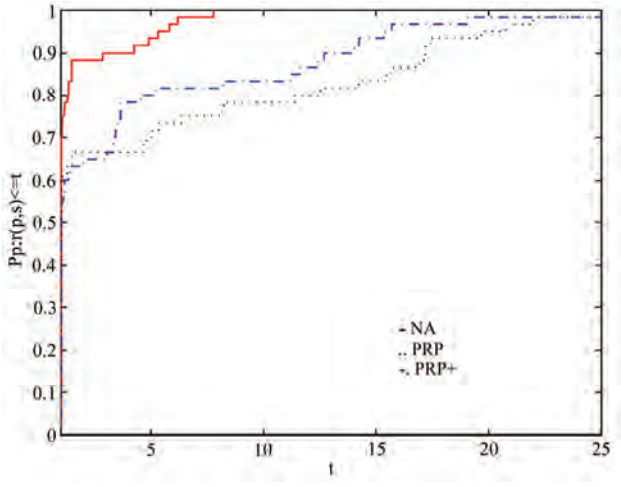



**Figure 1. Performance profiles(NT) of methods with Goldstein rule**

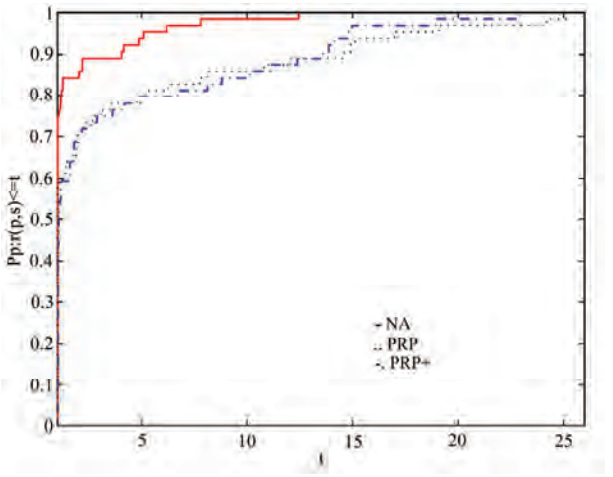



**Figure 2. Performance profiles(NT) of methods with strong Wolfe-Powell rule**

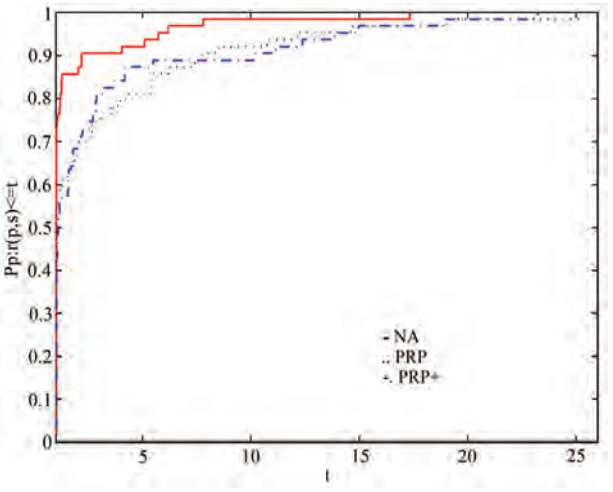



**Figure 3. Performance profiles (NT) of methods with weak Wolfe-Power rule**

it is clear that the given method has the most wins (has the highest probability of being the optimal solver).

In summary, the presented numerical results reveal that the new method, compared with the normal PRP method and the modified PRP method [14], has potential advantages.

## 5. Conclusions

This paper gives a new line search method for unconstrained optimization. The global and R-linear convergence are established under weaker assumptions on the search direction $d_k$. Especially, the direction $d_k$ satisfies the sufficient condition (5) and the condition (9) without carrying out any line search technique, and some paper [14,27,30] often obtains these two conditions by assumption. The comparison of the numerical results shows that the new search direction of the new algorithm is a good search direction at every iteration.

## REFERENCES


[1] G. Yuan and X. Lu, "A New Line Search Method with Trust Region for Unconstrained Optimization," *Communications on Applied Nonlinear Analysis*, Vol. 15, No. 1, 2008, pp. 35-49.

[2] G. Yuan, X. Lu, and Z. Wei, "New Two-Point Stepsize Gradient Methods for Solving Unconstrained Optimization Problems," *Natural Science Journal of Xiangtan University*, Vol. 29, No. 1, 2007, pp. 13-15.

[3] G. Yuan and Z. Wei, "New Line Search Methods for Uncons- trained Optimization," *Journal of the Korean Statistical Society*, Vol. 38, No. 1, 2009, pp. 29-39.

[4] Y. Yuan and W. Sun, "Theory and Methods of Optimization," Science Press of China, Beijing, 1999.

[5] D. C. Luenerger, "Linear and Nonlinear Programming," 2nd Edition, Addition Wesley, Reading, MA, 1989.

[6] J. Nocedal and S. J. Wright, "Numerical Optimization," Springer, Berlin, Heidelberg, New York, 1999.

[7] Z. Wei, G. Li, and L. Qi, "New Quasi-Newton Methods for Unconstrained Optimization Problems," *Applied Mathematics and Computation*, Vol. 175, No. 1, 2006, pp. 1156-1188.

[8] Z. Wei, G. Yu, G. Yuan, and Z. Lian, "The Superlinear Convergence of a Modified BFGS-type Method for Unconstrained Optimization," *Computational Optimization and Applications*, Vol. 29, No. 3, 2004, pp. 315-332.

[9] G. Yuan and Z. Wei, "The Superlinear Convergence Analysis of a Nonmonotone BFGS Algorithm on Convex Objective Functions," *Acta Mathematica Sinica, English Series*, Vol. 24, No. 1, 2008, pp. 35-42.

[10] G. Yuan and Z. Wei, "Convergence Analysis of a Modified BFGS Method on Convex Minimizations," *Computational Optimization and Applications*, Science Citation Index, 2008.

[11] Y. Dai and Y. Yuan, "A Nonlinear Conjugate Gradient with a Strong Global Convergence Properties," *SIAM Journal of Optimization*, Vol. 10, No. 1, 2000, pp. 177-182.

[12] Z. Wei, G. Li, and L. Qi, "New Nonlinear Conjugate Gradient Formulas for Large-Scale Unconstrained Optimization Problems," *Applied Mathematics and Computation*, Vol. 179, No. 2, 2006, pp. 407-430.

[13] G. Yuan and X. Lu, "A Modified PRP Conjugate Gradient Method," *Annals of Operations Research*, Vol. 166, No. 1, 2009, pp. 73-90.

[14] J. C. Gibert, J. Nocedal, "Global Convergence Properties of Conjugate Gradient Methods for Optimization," *SIAM Journal on Optimization*, Vol. 2, No. 1, 1992, pp. 21-42.

[15] Y. Dai and Y. Yuan, "Nonlinear Conjugate Gradient Methods," Shanghai Science and Technology Press, 2000.

[16] E. Polak and G. Ribiè, "Note Sur la Xonvergence de Directions Conjugèes," *Rev Francaise Informat Recherche Operatinelle* 3*e Annèe*, Vol. 16, 1969, pp. 35-43.

[17] M. J. D. Powell, "Nonconvex Minimization Calculations and the Conjugate Gradient Method," *Lecture Notes in Mathematics*, Springer-Verlag, Berlin, Vol. 1066, 1984, pp. 122-141.

[18] L. Grippo and S. Lucidi, "A Globally Convergent Version of the Polak-RibiÈRe Gradient Method," *Mathematical Programming*, Vol. 78, No. 3, 1997, pp. 375-391.

[19] W. W. Hager and H. Zhang, "A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search," *SIAM Journal on Optimization*, Vol. 16, No. 1, 2005, pp. 170-192.

[20] Z. Wei, S. Yao, and L. Liu, "The Convergence Properties of Some New Conjugate Gradient Methods," *Applied Mathematics and Computation*, Vol. 183, No. 2, 2006, pp. 1341-1350.

[21] G. H. Yu, "Nonlinear Self-Scaling Conjugate Gradient Methods for Large-scale Optimization Problems," Thesis of Doctor's Degree, Sun Yat-Sen University, 2007.

[22] G. Yuan, "Modified Nonlinear Conjugate Gradient Methods with Sufficient Descent Property for Large-Scale Optimization Problems," *Optimization Letters*, Vol. 3, No. 1, 2009, pp. 11-21.

[23] G. Yuan, "A Conjugate Gradient Method for Unconstrained Optimization Problems," *International Journal of Mathematics and Mathematical Sciences*, Vol. 2009, 2009, pp. 1-14.

[24] G. Yuan, X. Lu, and Z. Wei, "A Conjugate Gradient Method with Descent Direction for Unconstrained Optimization," *Journal of Computational and Applied Mathematics*, Vol. 233, No. 2, 2009, pp. 519-530.

[25] L. Zhang, W. Zhou, and D. Li, "A Descent Modified Polak-RibiÈRe-Polyak Conjugate Method and its Global Convergence," *IMA Journal on Numerical Analysis*, Vol. 26, No. 4, 2006, pp. 629-649.

[26] Y. Liu and C. Storey, "Efficient Generalized Conjugate Gradient Algorithms, Part 1: Theory," *Journal of Optimization Theory and Application*, Vol. 69, No. 1, 1992, pp. 17-41.

[27] Z. J. Shi, "Convergence of Line Search Methods for Unconstrained Optimization," *Applied Mathematics and Computation*, Vol. 157, No. 2, 2004, pp. 393-405.

[28] J. J. Moré, B. S. Garbow, and K. E. Hillstrome, "Testing Unconstrained Optimization Software," *ACM Transactions on Mathematical Software*, Vol. 7, No. 1, 1981, pp. 17-41.

[29] E. D. Dolan and J. J. Moré, "Benchmarking Optimization Software with Performance Profiles," *Mathematical Programming*, Vol. 91, No. 2, 2002, pp. 201-213.

[30] Y. Dai and Q. Ni, "Testing Different Conjugate Gradient Methods for Large-scale Unconstrained Optimization," *Journal of Computational Mathematics*, Vol. 21, No. 3, 2003, pp. 311-320.

*J. Software Engineering & Applications*, 2010, 3: 510-516
doi:10.4236/jsea.2010.35058 Published Online May 2010 (http://www.SciRP.org/journal/jsea)

# Experience in Using a PFW System – A Case Study

**Derrick Black[1], Elizabeth Hull[1], Ken Jackson[2]**

[1]School of Computing and Mathematics, University of Ulster, N Ireland, UK; [2]IBM Ltd., England, UK.
Email: mec.hull@ulster.ac.uk

Received October 26th, 2009; revised January 30th, 2010; accepted January 31st, 2010.

## ABSTRACT

*A safety document management system, in a domain such as the power industry, is known as a Permit for Work* (*PFW*) *solution. It is based on the issues prevalent in an environment and on the methods available to eliminate potential safety issues. This paper considers how a PFW system should be implemented. It does so by identifying an appropriate case study from a domain not usually associated with PFW systems, and applying a suitable process, **+PFW**.*

***Keywords*:** *Safety, Permit for Work, Systems Engineering, Health and Safety, Process, Modeling, Framework*

## 1. Introduction

For many years, process industries in the UK such as the mining and power generation industry have had government legislation applied to them which included the requirement to utilise a Permit for Work (PFW) system [1]. This has resulted in these industries developing a thorough understanding and competency in the implementation and operation of a safety document management system based on domain knowledge and operational experience. Research in requirements engineering [2,3] has recognized the need to ensure that systems are developed with safety considered as an integral part of requirements elicitation. Furthermore, it is generally understood that all stakeholders involved in the requirements process are fully conversant with the consequences of their decisions and the potential impact on the domain [4].

The introduction of the UK Health and Safety at Work Act [5-7] has widened this, and placed a requirement to operate a PFW system on all sectors of society where risks exist that cannot be eliminated or minimised sufficiently. Unfortunately these new sectors do not have the same experience or competency of safe systems. Thus the potential exists for this lack of operational knowledge to cause difficulties when a PFW system is introduced. When less experienced industry sectors start to introduce PFW systems (in response to risk assessments) it is important that they are implemented correctly and that the operational procedures applied to them are appropriate. The deficiency of user experience in these sectors may compound any problems and this is an area of concern.

This paper builds on work previously presented by the authors concerning the management of safety. First of all a Safety Framework [1,8] has been established. This allows a series of views to be identified that are relevant to safety in systems. These views convey different perspectives of the architecture including issues such as roles and organizational hierarchies, as well as rules and regulations. Hence a series of high level views can be established that may be applied to a system with safety as a core consideration. Secondly, a process **+PFW** [1,8], has been presented which ensures that a user will be in a position to utilise a PFW system without compromising safety. It is important to understand the Safety Framework and the process **+PFW** to fully appreciate the following sections of this paper.

This paper identifies a suitable candidate as a case study to use a PFW system. Intentionally, the domain chosen is outside the industries normally associated with PFW systems. This is described in Section 2. Having identified a suitable nominee as a case study, the paper considers the rationale for implementing a PFW system. Section 3 recounts the experiences of the identified user in implementing a PFW system in their environment. The evident shortcomings are presented and areas of concern still evident after implementation are highlighted. Section 4 then examines the application of the **+PFW** process to the implementation of a PFW system in an effort to eliminate the outstanding issues and provide an operational system designed to enhance the safety of users in the environment.

## 2. Identification of a Suitable Candidate

The need to use a PFW solution is based on the risks prevalent in an environment and on the methods available to eliminate these potential issues. The requirement to manage part of the safety process using a specialist system

such as a PFW solution is not an isolated decision. It consists of a series of considered assessments leading ultimately to a decision on whether an organisation needs to employ such a solution. Initially the concern is with the tasks performed. If all risks and hazards can be identified, managed and eliminated, or reduced to an acceptable level then there is no requirement for a PFW solution. However, if the risks cannot be managed successfully then a PFW is required. This concept is shown in **Figure 1**.

As an example, consider the domain of an academic institution. A university's goal is to develop a seat of learning for its students that is supported by world renowned research, innovation and teaching. However, to achieve this task the required infrastructure must be in place to support this objective. This infrastructure includes the provision of suitably equipped teaching and research facilities as well as accommodation and social provision for academics, students and support staff. All of these facilities need to be maintained and enhanced and it is here that many of the risks and hazards associated with this environment are present.

In providing the required facilities universities use high voltage equipment, heating and steam generating plant as well as scientific ancillaries such as fume extraction equipment. All of these items have associated risks and hazards such as electrocution, scalds, asphyxiation and toxicity. Many of these risks cannot be eliminated or reduced successfully and as a result a PFW is required for the maintenance environment of a university. Even though these items of equipment are commonplace across the university sector few if any universities have PFW systems in place and even fewer operate them successfully. Given the limited experience of using a PFW system in this sector, a university environment would appear to be ideally suited as a case study. A University in the UK was therefore chosen.

## 3. Initial Implementation of a PFW System

Having identified the need for a PFW solution based on a series of risk assessments and method statements, the University decided an appropriate solution would be to use a computerised PFW system. A tender exercise was carried out to source the most suitable solution. The result of this exercise was the procurement of the world leading computerised software system known as Eclipse. This product was development in the Power Generation Industry and is the standard system implemented in the majority of existing UK Power Stations. It has also been implemented in new power stations world-wide. The chosen system was installed with a minimal set of data at the request of the University.

The supplier carried out a series of training sessions on the operation of the system. This training was focused exclusively on the key presses required to deliver the required output rather than any concept of the operation of a PFW system. The result of the installation and training was the availability of a fully functionally PFW. However, because of the lack of data and understanding of the operational concepts of the system by the users, the installed system lay unused for eighteen months, with no safety documents being issued. The supplier returned to the University on a number of occasions to ascertain if they could be of assistance in implementing the full operation of the system but to no avail. No operational procedures existed and the data required for day to day operations, such as an asset list and the identification of the participants, were never established. Thus a system deemed necessary to fulfill health and safety obligations remained unused.

### 3.1 Issues with the Initial Implementation

The initial installation was performed to facilitate the requirement to provide a safety document management system. Identification of this need was made following a risk assessment exercise carried out by the Estates Directorate in the University. The assessment looked at some of the key activities performed by this department and concluded that a safety document management system was required. However, the group tasked with this initial assessment programme was made up of several members of staff some of whom had limited or no experience of PFW systems or had widely differing interpretations of the operational procedures required. These differences were left unresolved and the resultant system installation had no agreed operational process in place.

Despite the fact that no cohesive operational procedures had been developed and the data required to populate system tables had not be developed, nor agreed, the system was installed and training was undertaken. A number of issues remained to be resolved. These included:

- Individuals responsible for the operation of the system remained unidentified
- Management roles had not been established
- Users roles had not been identification
- No data was available to populate the system tables
- The areas to be addressed by the PFW system remained unidentified
- Establishment of operational procedures remained to be undertaken.

## 4. Using +PFW to Implement a Solution

As the University recognized that the most suitable solution available had been chosen, it was agreed that the problem lay not with the computerised solution but with the process applied to implement the system.

To facilitate the implementation and operation of the PFW system the process **+PFW** [9] was introduced to the University staff and its concept explained. Following detailed discussions it was agreed that **+PFW** should be used in the second attempt to implement the safety docu-

ment management system. A group was identified and tasked with fully implementing the PFW system.

The three key stages of **+PFW** are as follows:

1) Establishment of a maintenance list

2) Development of an equipment list based on the context of the maintenance list

3) Establishment of specified roles

The Safety Framework [10] facilitates the way in which the stages of the process can be realized. The framework identifies three potential groups of views:

- Operational Group, OG
- Safety Regulation Group, SG
- Requirements Group, RG
- And proposes a way in which each can be implemented.

### 4.1 Creating the Maintenance List

**+PFW** was developed to be used in a standalone environment were the requirements phase had been completed and a PFW was deemed necessary, but the implementation and operational procedures had not yet been discovered. The University scenario described previously is a perfect example of this situation since the requirement elicitation process had resulted in the installation of the PFW system but the implementation of the solution was the cause of concern. **Figure 2** shows the Maintenance List stages in **+PFW**.

This maintenance list defines those items of plant and equipment that require the issue of a safety document when repair tasks are being undertaken. The process suggests that use of the *Safety Framework* [8] is needed to achieve the correct maintenance list. Cognition must be made of the principles of operation of the system, the organisational roles in place, any existing safety rules utilised as well as identifying the intention behind any decisions made. Any maintenance list must be developed in the context set by these requirements. Thus the first task was to identify the objectives of the PFW system.

The University had decided that the PFW system was to

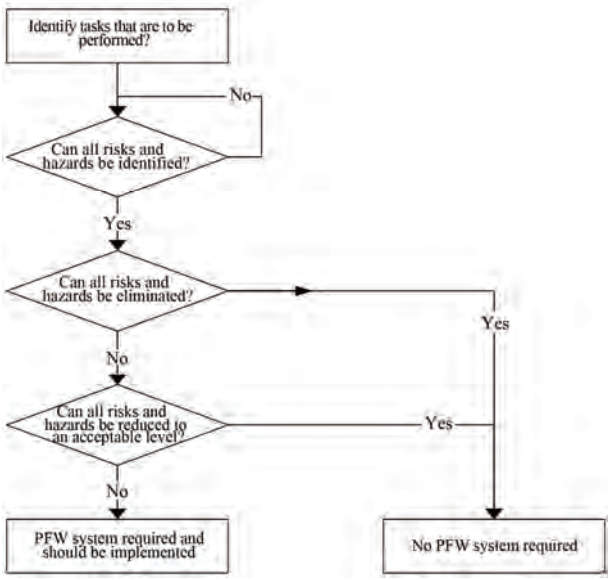



**Figure 1. Decision process involved in introducing a PFW solution**

be used to protect individual's safety rather than plant safety and that initially it was to be operated by the Estates Department in conjunction with its internal staff and external contractors. This objective clearly removed elements of equipment not maintained by this group of staff and as such excluded research equipment from consideration. Although risks may still be evident for these items of equipment their omission from the PFW system is justified on the basis that the initial implementation was for a particular group of staff.

Limiting the operation of the system to Estates staff and external contractors employed to perform maintenance activities for this group was another element that placed the operation of the system in an agreed context. Since the system's operation was limited to this group only the organisational hierarchy within the Estate's department needed to be considered in terms of who would be in-

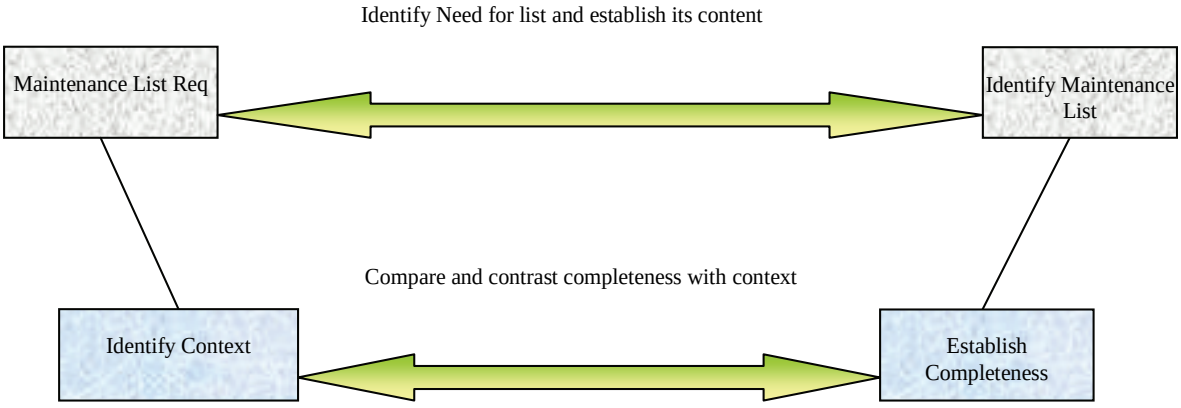



**Figure 2. Maintenance list stages of +PFW**

volved in the operation of the solution. Therefore only the identified roles within this structure had an input to the development of the maintenance list thereby restricting the number of potential stakeholders.

Before the maintenance list was developed, the rationale for including items of equipment and plant needed to be understood. The University decided the most appropriate method for this was to group items of plant and equipment and then decide if they were to be included in the maintenance list. An examination was undertaken using risk assessments of the tasks to be undertaken to ensure that the list was complete. The outcome was that a Maintenance List specific to the requirements of the University was created that could be justified in terms of its context, its completeness and the reasoning behind those elements included and those omitted. This first draft of the maintenance list was approved for use. It is considered dynamic and will be reviewed on a regular basis.

The decision on which type of equipment to included was influenced by the domain knowledge and experience of the Estate's Department staff and the current legislation.

## 4.2 Establishing the Equipment List

The second element of **+PFW** concentrates on the establishment of the *Equipment List* and is shown in **Figure 3**. This is based on the maintenance list using the same context.

The equipment list is used to identify all potential sources of energy that may cause an item of equipment to operate, or any potentially hazardous materials stored in the equipment or plant used by the University. Elements such as high voltage supplies, steam and high pressure water, as well as flammable and hazardous materials etc were all identified as potential sources of supply.

The equipment list is a dynamic document needing continual review to ensure that modifications to the plant and equipment and the overall electro-mechanical system are included as appropriate. Changes to potential sources of energy need to be updated to ensure an up to date, accurate list is maintained. In addition the equipment list needs to be reviewed in association with the agreed maintenance list to reflect changes, additions and deletions of items from this list. The University recognised this requirement and has established a procedure to actively review the contents of both the maintenance and equipment lists as well as auditing the overall operation of the system.

## 4.3 Identification of the PFW Roles

**+PFW** identifies the requirement to establish the roles and responsibilities associated with the implementation and operation of a PFW system as shown in **Figure 4**.

To operate a PFW successfully the roles to be performed by users must be clearly and unambiguously identified. The first of these roles was identified as the individuals charged with assessing the task to be undertaken to determine if a safety document is to be issued. Although the maintenance list identifies the equipment to be included that does not mean that in every instance work is performed on these items; a safety document is required. However, not all safety documents perform the same task. Although they are similar in format two distinct safety document types were identified by the University as being relevant to their procedures. These documents are referred to as the **Permit** and the **Limited Work Certificate** (LWC). Both documents state the work to be undertaken and the precautions applied to achieve safety. Where they differ is in the isolation applied to the equipment. In the case of the Permit the equipment is isolated completely from the potential sources of energy while for the LWC safety is achieved by limiting either the work to be undertaken or the area in which the task is to be carried out. For example working on a high voltage busbar would require a Permit while brushing the floor in front of the high voltage switch gear would require a LWC, because the work is in a dangerous area but no contact is possible with the live conductors.

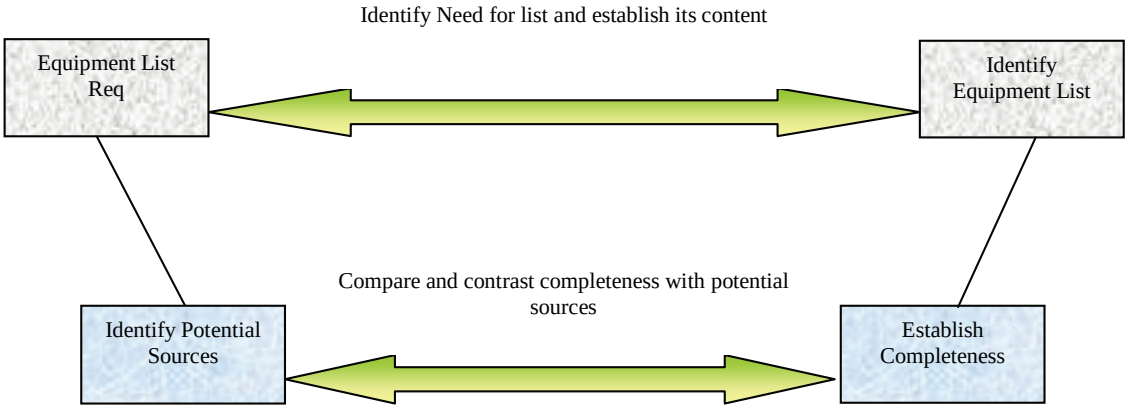



**Figure 3. Equipment list stages of +PFW**

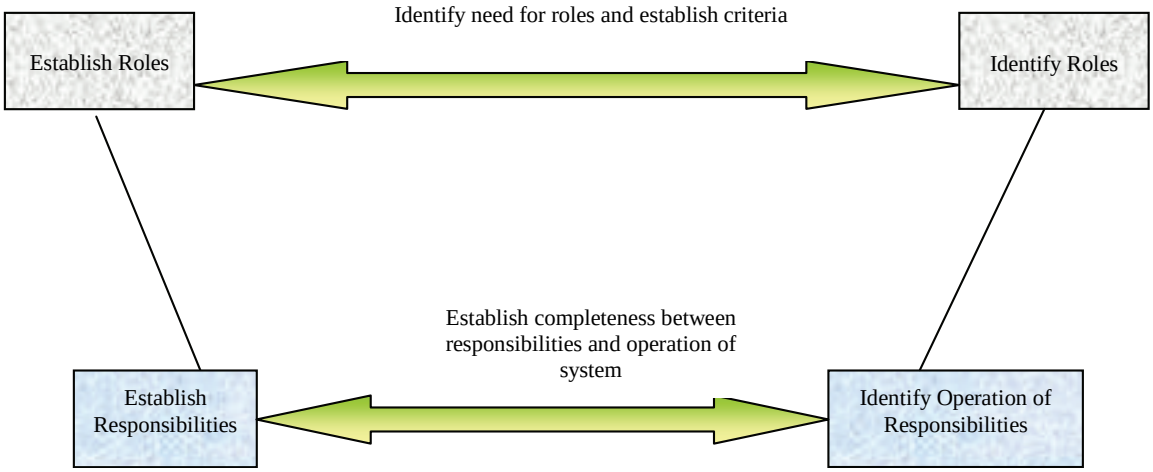



**Figure 4. Roles and responsibilities associated with a PFW system**

The creation of the safety document requires an individual skilled in the application of the PFW system as well as individuals with detailed knowledge and experience of the domain. This role must identify any precautions and isolation points to be applied to ensure safety. Although the equipment list identifies the isolation to be applied to a particular equipment item it is unwise to rely on this list completely, as there may be occasions when the isolation suggested may be inappropriate or unavailable.

Once a safety document is issued there is clearly a role to be played in the performance of the repair task, but equally a role needs to exist to ensure that the requirements of the PFW system are not breached.

Finally, a role was established that is only applicable in a very specific set of circumstances. The University proposed to use a 'Hot Work Certificate' in association with a safety document where the use of cutting or burning equipment is required in the repair task. Although this is common in the operation of PFW systems, it differs in that normally PFWs are issued in process industries that operate 24 hrs per day while the University's Estates department operates 9 to 5 daily. There is a risk that heated material may spontaneously combust. To prevent this, a safety document may stipulate a required time to undertake a 'Fire Watch' whereby someone is charged with remaining in situ for a period after the work has been completed. To ensure this has occurred, it is advisable for a nominated individual to visit the site of the repair when the safety document is returned as completed (after the fire watch). Since no maintenance engineering staff are likely to be present after hours the task has been delegated to the security staff and as such this is an identified role in the PFW operation.

#### 4.3.1 Naming and Assignment of Identified Roles

**+PFW** indicates that PFW roles should be established in association with the operational roles and organisational structure prevalent in the domain as well as using the interaction between these elements. In the University scenario referencing these aspects led to the decision that three roles would be utilised in the operation of the PFW system. One of the roles would be performed by the Estates Engineering and Project Managers and assistants, the second would be performed by competent maintenance staff and external contractor's staff while the third would be performed by the security staff as previously described.

The first role was named as an Authorised Person. This role was assigned the responsibility to issue a safety document (and its cancellation on completion of the task) and the decision to isolate equipment (and to de-isolate).

The second role was named as a Competent Person. The term 'Competent Person' is unlike the conventional definition of competent. To be considered a Competent Person in the PFW system a user needs to be competent in their own discipline, for example only qualified electricians can work at electrical installations, as well as being assessed competent in the use of the PFW system.

A Competent Person, using the University's definition, means an individual charged with supervising and/or undertaking the work required to complete the repair task while being responsible for requesting a safety document, receiving it when it is issued, ensuring general safety is maintained at the work site and returning the safety document on completion of the task.

The Security role has been discussed previously and the responsibility is to receive a completed safety document when it is returned out of hours, visit the site of a repair that has had a Hot Work Certificate issued on it and to return any safety documents to the Authorised Person.

The University identified an additional role that was considered important, although plays no part in the actual operation of the system, staff, students and contractors who are not involved in the repair task indicated by a safety document need to comply with the terms of the safety document, in terms of the access to a restricted area

etc. This role is not commonly included in the roles assigned in a PFW system but the University felt that it was appropriate to include this role so that no individual was overlooked when training was being undertaken. Plans are currently being drawn up to include this in the induction progress for contractors, new staff and students.

### 4.3.2 Documenting the Assumptions

The final stage of **+PFW** deals with the assumptions, methods of isolation and the operational rules relevant to the implementation and operation of PFW system and is shown in **Figure 5**.

The majority of assumptions made in implementing a PFW system are made at the maintenance list creation stage but these decisions need to be recorded to allow traceability on all decisions taken. They should also be tested to ensure that they are relevant to the domain. The assumptions made in this case study were that only equipment maintained by the Estates Department of the University would be included. All other plant and equipment even if it was on the University estate would be excluded. However, this raised a question with regard to what happened to the equipment when it was handed to an external contractor as part of a major refurbishment/replacement process. The outcome of deliberations on this point lead to the assumption that the equipment would be temporarily removed from the PFW system until the refurbishment had been completed.

The equipment list identified earlier detailed the isola tion applicable to each equipment or plant item but did not consider how this was to be achieved. Two possible scenarios are common in the operation of PFW system. One relies on the understanding of the stakeholders in the domain. In this instance the isolation is applied by closing valves, opening electrical switches and opening drain and vent valves on the item. Notices are then placed on the isolation points stating their use on a safety document system. The second option applies the same methodology to the isolation points, but in this instance locks are applied to the devices and the keys that from these locks are placed in a safe which is controlled by the safety document. Either method is suitable provided all the stakeholders involved understand the principles. Although the University believed that the second option might be more secure it has opted for the first since it is an easier method to implement and operate.

The final element of the process suggests that a set of operational rules are required. These will be developed in due course. The University felt that it was more appropriate to develop these rules following a period of operation so that the user community had gained a sound appreciation of the system and its nuances before committing to the operational rules.

### 4.3.3 Evaluation of the Implementation Process

Using **+PFW** highlighted a significant number of areas that had not been sufficiently addressed during the initial implementation procedure. They included the need to establish a full and comprehensive maintenance list based on the agreed groups of equipment to be included in the system operation. Hence **+PFW** delivered a positive impact almost immediately and this carried on throughout the implementation process.

Having identified the maintenance list, the requirement for an equipment list was clearly evident since knowing the equipment to be worked on as part of the system was

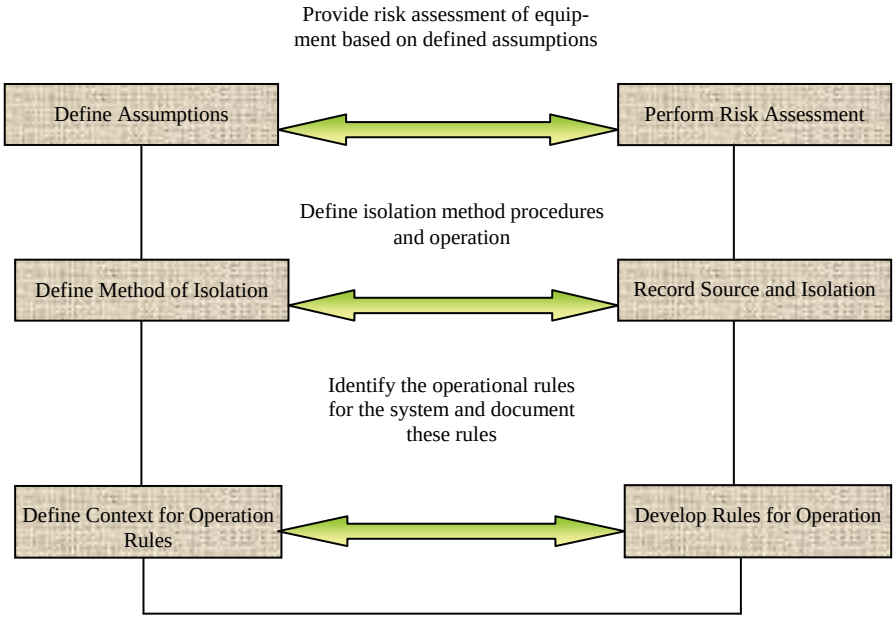



**Figure 5. Identifying the assumptions, methods of isolation and operational rules**

only part of the issue. The normal or potential sources of energy to each of these items was obviously required if the equipment was to be rendered safe for the repair tasks.

The identification of the roles involved in the PFW system was a much more contentious issue for the University. The need to establish the roles was not the issue, however the roles to be performed and the method of operation for the roles caused major differences of opinion between all the stakeholders. Part of the problem in this area was that several stakeholders had experience of PFW system gained in different environments. Each of these stakeholders had slightly differing views of what the correct procedures to employ should be and where the responsibility for the operations of the various elements resided. To facilitate the establishment of the roles and responsibilities the University sought advice from the supplier of the Eclipse product and other experienced PFW system users. This did not quite achieve the desired result since the supplier is heavily involved in the Power Generation domain and had what were considered strict interpretations of the requirements for the roles in the system while some of the other users consulted were more lax in their definitions. A compromise was eventually reached that combined the major roles suggested by some stakeholders and validated by the supplier with some more lenient aspects suggested by other stakeholders. The outcome has proved to be very satisfactory for the University. It has clearly established the key roles while addressing specific issues such as the fire watch scenario.

**+PFW** indicated that the desired outcome required documentation to enable users to operate the PFW system effectively. This has been achieved with the University now in possession of Safety Procedure Document. It provides a clear overview of the operational procedure to be applied while identifying the roles and responsibilities required to effectively operate the system. It establishes the concept behind the maintenance and equipment lists unambiguously.

At present no formal training has been undertaken in the concept of PFW systems. However, a contract has been prepared for issue to a Health and Safety company to provide the required training for all levels of staff in their identified roles. Additionally a request has been made to each contractor requesting the nomination of suitably qualified individuals to be trained as 'Competent Persons' within the PFW system.

## 5. Conclusions

This paper has described how, following an initial attempt at implementing a system, **+PFW** was utilised. The process highlighted the elements that needed to be established and validated for the implementation to be considered a success. Having reached an impasse after the first attempt to implement the system the University was sceptical that any progress could be made but **+PFW** clearly removed these doubts and an effective PFW system is now in operation. It has allowed the University to develop the information necessary to fully implement and operate a PFW system.

The initial implementation procedure resulted in a number of key elements being missed with the consequence that a poorly installed system, which could not be operated by the University, was provided. The implementation did not fulfil the University's identified requirement to protect the safety of individuals working on equipment when outstanding risks existed. By following the process, these missing elements were identified and provided the University with the skills necessary to establish the required outcomes in each area. These elements included the need to:

- Identify key individuals in the operation of the system
- Establish pivotal managerial roles
- Provide users with an identified set of tasks for which they are responsible
- Identify the roles required for the operation of the system
- Identify the activities requiring a safety document and their associated methods of isolation
- Identification of the equipment and plant to be included in the PFW system.

These areas were all fully addressed by the **+PFW** process using stakeholders with limited or no experience of the concepts associated with a PFW system.

## REFERENCES

[1] D. D. Black, "Management of Safety-A Systems Engineering Approach," PhD Thesis, University of Ulster, 2008.

[2] P. G. Bishop and R. E. Bloomfield, "The SHIP Safety Case Approach," *Proceeding of Safecomp* 95, Belgirate, 1995, pp. 437-451.

[3] N. G. Leveson, "Safeware System Safety and Computers," Addison-Wesley, 2001, pp. 171-184.

[4] M. E. C. Hull, K. Jackson and A. J. J. Dick, "Requirements Engineering," 2nd Edition, Springer, 2005.

[5] "Essentials of Health and Safety at Work," *HSE Books*, Health and Safety Executive, 2006.

[6] "Health and Safety at Work (Northern Ireland) Order," *Northern Ireland orders in Council*, No. 1039, 1978.

[7] "Permit-to-work Systems," *HSE Books Online*, Health and Safety Executive, 1997.

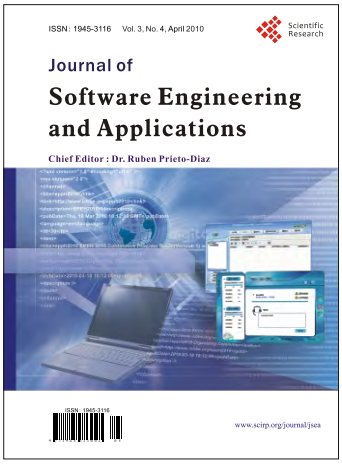
[8] D. D. Black, M. E. C. Hull and K. Jackson, "Combining a Safety Management Process with a Safety Framework," *Journal of Intelligent Information Management*, Vol. 2, No. 4, 2010, pp. 233-242.

[9] D. D. Black, M. E. C. Hull and K. Jackson, "+PFW-A Process for System Safety," 2009.

[10] D. D. Black, M. E. C. Hull and K. Jackson, "Systems Engineering and Safety-A Framework," 2009.

# Call for Papers




# Journal of Software Engineering and Applications (JSEA)

ISSN 1945-3116 (print) ISSN 1945-3124 (online)
www.scirp.org/journal/jsea

**JSEA** publishes four categories of original technical articles: papers, communications, reviews, and discussions. Papers are well-documented final reports of research projects. Communications are shorter and contain noteworthy items of technical interest or ideas required rapid publication. Reviews are synoptic papers on a subject of general interest, with ample literature references, and written for readers with widely varying background. Discussions on published reports, with author rebuttals, form the fourth category of JSEA publications.

## Editor-in-Chief

Dr. Ruben Prieto-Diaz, Universidad Carlos III de Madrid, Spain

## Subject Coverage

- Applications and Case Studies
- Artificial Intelligence Approaches to Software Engineering
- Automated Software Design and Synthesis
- Automated Software Specification
- Component-Based Software Engineering
- Computer-Supported Cooperative Work
- Software Design Methods
- Human-Computer Interaction
- Internet and Information Systems Development
- Knowledge Acquisition
- Multimedia and Hypermedia in Software Engineering
- Object-Oriented Technology
- Patterns and Frameworks
- Process and Workflow Management
- Programming Languages and Software Engineering
- Program Understanding Issues
- Reflection and Metadata Approaches
- Reliability and Fault Tolerance
- Requirements Engineering
- Reverse Engineering
- Security and Privacy
- Software Architecture
- Software Domain Modeling and Meta-Modeling
- Software Engineering Decision Support
- Software Maintenance and Evolution
- Software Process Modeling
- Software Reuse
- Software Testing
- System Applications and Experience
- Tutoring, Help and Documentation Systems

## Notes for Prospective Authors

Submitted papers should not have been previously published nor be currently under consideration for publication elsewhere. All papers are refereed through a peer review process. For more details about the submissions, please access the website.

## Website and E-Mail

Website: http://www.scirp.org/journal/jsea E-Mail: jsea@scirp.org

*J. Software Engineering and Applications,* **2010, 3**
Published Online May 2010 in SciRes (http://www.SciRP.org/journal/jsea).

# TABLE OF CONTENTS

**Volume 3, Number 5** **May 2010**

