ISSN: 1945-3116(Print), 1945-3124(Online)
Vol. 3, No. 4, April 2010

Scientific Research

Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

9 771945 311001 04

www.scirp.org/journal/jsea

Journal Editorial Board

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

J. Software Engineering & Applications, **2010, 3: 303-418**
Published Online April 2010 in SciRes (http://www.SciRP.org/journal/jsea/).

TABLE OF CONTENTS

Volume 3 Number 4 April 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: 303-311
doi:10.4236/jsea.2010.34036 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

Separation of Fault Tolerance and Non-Functional Concerns: Aspect Oriented Patterns and Evaluation

Kashif Hameed, Rob Williams, Jim Smith

University of the West of England, Bristol Institute of Technology, Bristol, UK.
Email: {Kashif3.Hameed, Rob.Williams, James.Smith}@uwe.ac.uk

Received January 1st, 2010; revised January 30th, 2010; accepted February 1st, 2010.

ABSTRACT

Dependable computer based systems employing fault tolerance and robust software development techniques demand additional error detection and recovery related tasks. This results in tangling of core functionality with these cross cutting non-functional concerns. In this regard current work identifies these dependability related non-functional and cross-cutting concerns and proposes design and implementation solutions in an aspect oriented framework that modularizes and separates them from core functionality. The degree of separation has been quantified using software metrics. A Lego NXT Robot based case study has been completed to evaluate the proposed design framework.

***Keywords*:** *Aspect Oriented Design and Programming, Separation of Concerns, Executable Assertions, Exception Handling, Fault Tolerance, Software Metrics*

1. Introduction

Adding fault tolerance (FT) measures and other non-functional requirements to safety critical and mission critical applications introduces additional complexity to the core application. By incorporating handler code, for error detection, checkpointing, exception handling, and redundancy/diversity management, the additional complexity may adversely affect the dependability of a safety critical or mission critical system.

One of the solutions to reduce this complexity is to separate and modularize the extra, cross-cutting concerns from the true functionality.

Although Rate of Change (ROC) based plausibility checks for error detection and recovery have been addressed by [1,2], unfortunately none of the previous studies propose the separation of these error handling concerns from true functionality to avoid complexity.

At the level of design and programming, several approaches have been utilized that aim at separating functional and non-functional aspects. Component level approach like IFTC [3], computational reflection and meta-object protocol based MOP [4] have shown that dependability issues can be implemented independently of functional requirements.

The evolving area of Aspect-Oriented Programming & Design (AOP&D) presents the same level of independence by supporting the modularized implementation of crosscutting concerns.

Aspect-oriented language extensions, like AspectJ [5] and AspectC++ [6] provide mechanisms like *Advice* (behavioural and structural changes) that may be applied by a pre-processor at specific locations in the program called *join point.* These are designated by *pointcut* expressions. In addition to that, static and dynamic modifications to a program are incorporated by *slices* which can affect the static structure of classes and functions.

The current work thus proposes some generalized aspect oriented design patterns representing fault tolerance error detection and recovery mechanisms like ROC plausibility checks, exception handling, checkpointing and watchdog. Moreover some additional design patterns for developing robust mission/safety critical software are also presented. Software metrics like coupling, cohesion and size have been applied quite successfully to access and evaluate the quality attributes of OO software systems [7, 8]. However separation of concerns (SOC) especially cross cutting ones in the light of new abstraction addressed by AO software development demands some additional metrics. The current work reviews these additional metrics like concern diffusion over the components (CDC), concern diffusion over the operations (CDO) and concern diffusion over the lines of code (CDLOC). The

SOC metric suite is later applied on the proposed AO patterns in an empirical case study. This helps evaluating the degree to which AOSD modularizes the FT concerns and its impact on other quality attributes.

The validation and dependability assessment of proposed AOFT patterns has already been done in an earlier work by the author [9].

2. Aspect Oriented Exception Handling Patterns

Exception handling has been deployed as a key mechanism in implementing software fault tolerance through forward and backward error recovery mechanisms. It provides a convenient means of structuring software that has to deal with erroneous conditions [10].

In [11], the authors addresses the weaknesses of exception handling mechanisms provided by mainstream programming languages like Java, Ada, C++, C#. In their experience exception handling code is inter-twined with the normal code. This hinders maintenance and reuse of both normal and exception handling code.

Moreover as argued by [12], exception handling is difficult to develop and has not been well understood. This is due to the fact that it introduces additional complexity and has been misused when applied to a novel application domain. This has further increased the ratio of system failures due to poorly designed fault tolerance strategies.

Thus fault tolerance measures using exception handling should make it possible to produce software where 1) error handling code and normal code are separated logically and physically; 2) the impact of complexity on the overall system is minimized; and 3) the fault tolerance strategy may be maintainable and evolvable with increasing demands of dependability.

In this respect, [4] has proposed an architectural pattern for exception handling. They address the issues like specification and signaling of exceptions, specification and invocation of handlers and searching of handlers. These architectural and design patterns have been influenced by computational reflection and meta-object protocol.

However, most meta-programming languages suffer performance penalties due to the increase in meta-level computation at run-time. This is because most of the decisions about semantics are made at run-time by the meta-objects, and the overhead to invoke the meta-objects reduces the system performance [13].

Therefore we propose generalized aspect based patterns for monitoring, error detection, exception raising and exception handling using a static aspect weaver. These patterns would lead to integration towards a robust and dependable aspect based software fault tolerance. The following design notations have been used to express aspect-oriented design patterns shown in **Figure 1**.

2.1 Error Detection and Exception Throwing Aspect

Error detection and throwing exceptions has been an anchor in implementing any fault tolerance strategy. This aspect detects faults and throws range, input and output type of exceptions. The overall structure of this aspect is shown in **Figure 2**. The *GenThrowErrExcept* join points the *NormalClass* via three pointcut expressions for each type of fault tolerance case.

RangeErrPc: this join points the *contexMethod*() only. It initiates a before advice to check the range type errors before executing the *contextMethod*(). In case the assertions don't remain valid or acceptable behavior constraints are not met, *RaneErrExc* exception is raised.

InputErrPc: this join points the *contextMethod*() further scoped down with input arguments of the *contextMethod***()**. It initiates a before advice to check the valid input before the execution of the context method. Incase the input is not valid it raises *InputErrExc***.**

OutputErrPc: this join points the *contextMethod*()

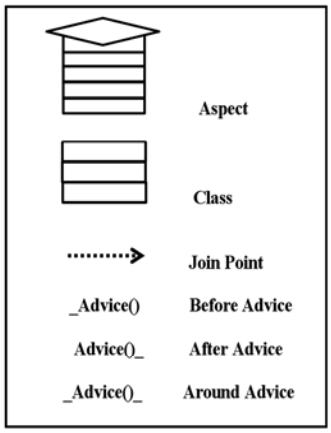


Figure 1. Aspect oriented design notations

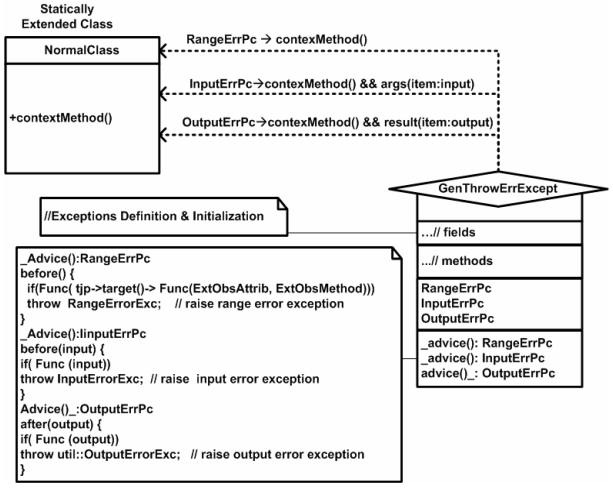


Figure 2. Error detection, exception throwing

further scoped down with results as output of the *contextMethod*(). It initiates an after advice to check the valid output after the execution of the context method. Incase the output is not valid it raises **OutputErrExc**.

2.2 Rate of Change Plausibility Check Aspect

This aspect as shown in **Figures 3** and **4** is responsible for checking the erroneous state of the system based on the rate of change in critical signal/data values. Once an erroneous state is detected, the respective exception is raised. Various exceptions are also defined and initialized in this aspect. The *pointcut GetSensorData* defines the location where error checking plausibility checks are weaved whenever a critical data/sensor reading function is called. The light weight ROC-based plausibility assertions are executed in the *advice* part of this aspect.

2.3 Catcher Handler Aspect

The *CatcherHandler* aspect as shown shown in **Figure 5(a)** is responsible for identifying and invoking the appropriate handler. This pattern addresses two run-time handling strategies.

The first strategy is designated by an *exit_main* pointcut expression. It checks the run-time *main*() function for various fatal error exceptions and finally aborts or exits the main program upon error detection. This aspect may be used to implement safe shut-down or restart mechanisms in safety critical systems to ensure safety, if a fatal error occurs or safety is breached.

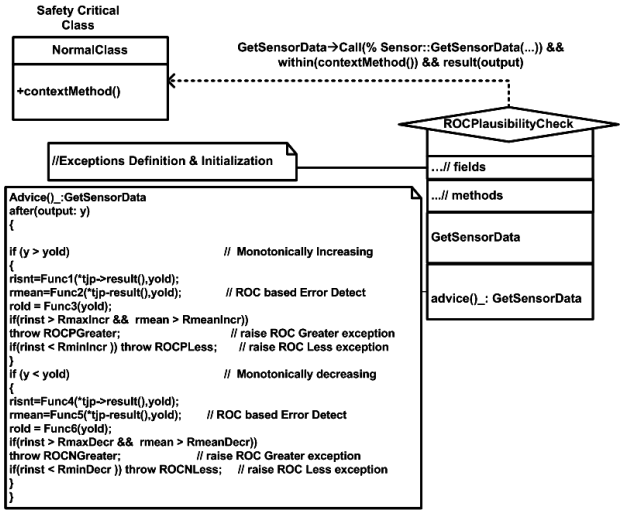


Figure 3. Rate of change aspect pattern structure

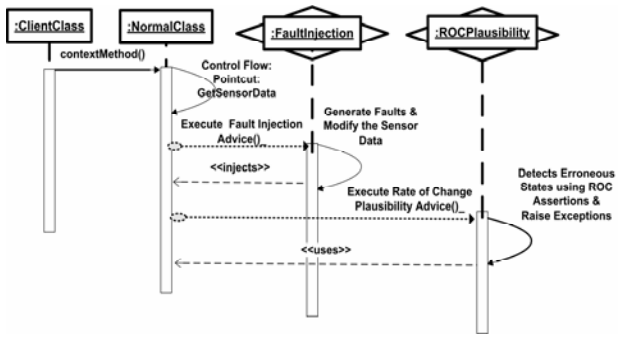


Figure 4. Rate of change aspect pattern dynamics

The second strategy returns from the called function as soon as the error is detected. The raised exception is caught after giving warning or doing some effective action in the catch block. This can help in preventing error propagation. Using this aspect, every call to critical functions is secured under a try/catch block to ensure effective fault tolerance against an erroneous state.

It can be seen in the **Figure 5(a)** below that *exit_main* pointcut expression join points the main() run-time function. Whereas *caller_return* pointcut expression join points every call to the *contextMethod*(). Moreover *exit_main* and *caller_return* pointcut expressions are associated with an around advice to implement error handling. The tjp→proceed() allows the execution run-time main() and called functions in the try block.

The **advice** block of the catcher handler identifies the exception raised as a result of in-appropriate changes in the rate of signal or data. Once the exception is identified, the recovery mechanism is initiated that assign new values to signal or data variables based on previous trends or history of the variable.

2.4 Dynamics of Exception Handling Aspect

This scenario shown in **Figure 5(b)** represents a typical error handling case. It simulates two error handling strategies. In the first case, control is returned from the caller to stop the propagation of errors along with a system warning. In the second case the program exits due to a fatal error. This may be used to implement shutdown or restart scenarios. Moreover the extension of a class member function with a *try* block is also explained. A client object invokes the *contextMethod*() on an instance of *NormalClass*. The control is transferred to *CatcherHandler* aspect that extends the *contextMethod*() by wrapping it in a *try* block and executes the normal code. In case an exception is raised by previous aspect, the exception is caught by the *CatcherHandler* aspect. This is shown by the catch message. The condition shows the type of exception *e* to be handled by the handler aspect. *CatcherHandler* aspect handles the exception e. the caller_return strategy warns or signals the client about the exception and returns from the caller. The client may invoke the *contextMethod*2() as appropriate. In exit_main strategy, the control is retuned to client that exits the current instances as shown by the life line end status.

3. Watch Dog Aspect

A watchdog is a common concept used in real time systems for detecting and handling errors in real time sys

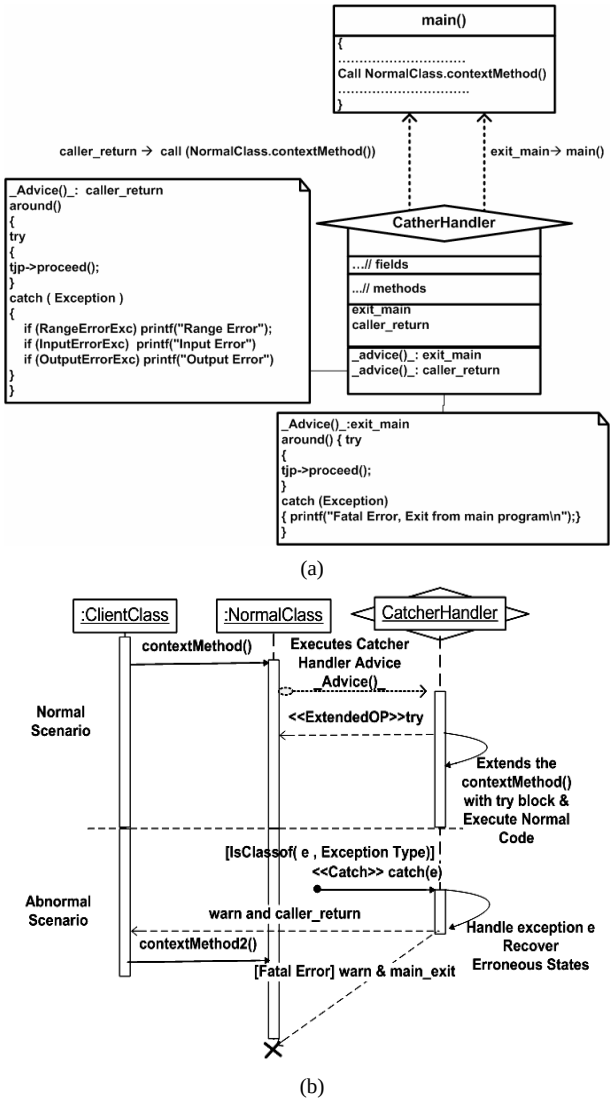


Figure 5. Catcher handler aspect. (a) Structure; (b) Dynamics

tems. It is a component that detects error by receiving a delayed or null service response. Based on such timing faults, it initiates a corrective action, such as reset, shutdown, alarm to notify attending personnel, or signaling more elaborate error-recovery mechanisms. Sometimes software watchdogs are more active by performing periodic built-in-tests (BIT). Synchronous tasks are more prone to such timing based faults resulting in mission failures.

In this regard we present a watchdog aspect (**Figure 6**) to make such tasks fault tolerant by weaving an advice code. Thus every synchronous mission critical task is monitored against a deadline that is derived from the worst case execution time of the overall task. As the dead line is expired, the mission is aborted. The watchdog aspect is presented below. It can be seen that every call to a **contextMethod()** of a **NormalClass** is weaved with a timing check to see whether time delay between current and previous call exceeds the dead line or not. The watchdog aspect communicates with an external clock interface to receive time stamps. Thus the watch dog aspect separates timing concerns from the true functionality. It also localizes the definition and signaling of exceptions.

4. Save Data and Checkpointing Aspect

Some tasks require context related critical data to be stored for post analysis and executing recovery mechanisms. Every call to these tasks is weaved with a data saving advice. **SaveData** aspect (**Figure 7**) provides checkpointed stable recovery data to be used in ROC based error detection and recovery mechanisms.

Whenever a critical function is called, **SaveData** aspect stores the contextual state information with the help of **MemoryInterface.**

5. System Configuration & Initialization Aspect

Most real time systems rely on sensors in-order to attain

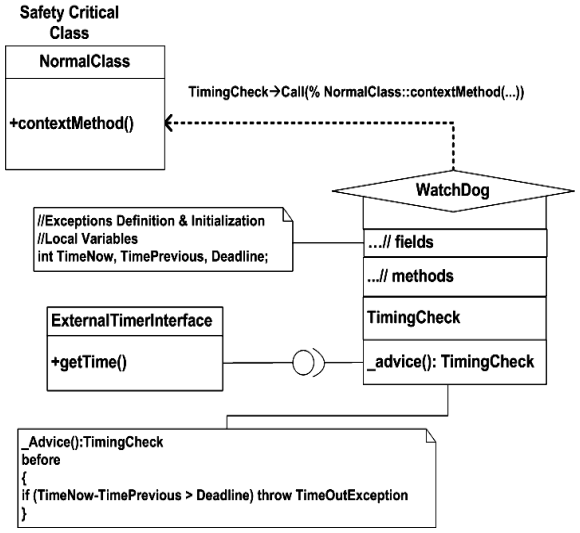


Figure 6. Watchdog aspect

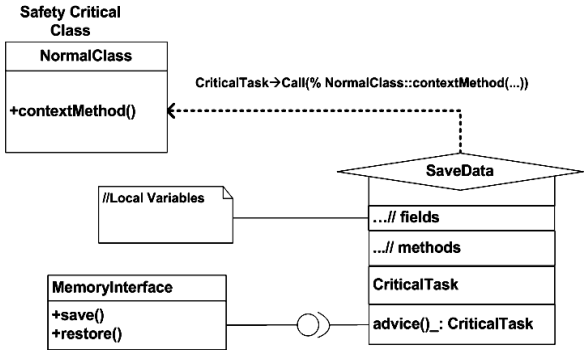


Figure 7. Save data & checkpointing aspect

physical information from external environment. These sensors need to be configured and initialized depending upon the modes of operation. For example a Lego NXT robot (Tribot) used in our case study uses light, ultrasonic and rotation sensors to carry out tasks. These sensors are mapped on respective ports of the NXT brick. Moreover they must be initialized before starting actual tasks. It has also been observed that rotation sensors are reinitialized as the direction of rotation changes (when Tribot start traversing backward). All such requirements either cut-across true functional concern or emerges as additional non-functional requirement. Such requirements have been implemented in an aspect thus separating them from true functional concern. This aspect is weaved as a startup advice in the control flow of main program.

6. Mission Pre-Conditions Aspect

Mission critical real time systems require some pre-conditions or constraints to be met before starting the core task. For example Tribot check the voltage level of batteries and ambient light before staring its mission so that it could fulfill its tasks reliably. If the above constraints are not met, mission is aborted. Such constraints are cross cutting to core functional requirements and thus implemented as a separate aspect as shown as shown in **Figure 8**.

As soon as the software finishes system initialization, the said aspect acquires environmental data from the **SensorInteface** and checks against the pre-conditions or constraints. If the constraints are not met, an exception is thrown and mission is aborted.

7. Case Study

In order to evaluate proposed AO design patterns, a case study has been carried out using a LEGO NXT Robot (Tribot). This uses an Atmel 32-bit ARM processor running at 48 MHz. Our development environment utilizes AspectC++ 1.0pre3 as aspect weaver [6].

The Tribot has been built consisting of two front wheels driven by servo motors, a small rear wheel and an arm holding a hockey stick with the help of some standard Lego parts. Ultrasonic and light sensors are also available for navigation and guidance purposes.

An interesting task has been chosen to validate our design. In this example Tribot hits a red ball with its hockey stick avoiding the blue ball placed on the same ball stand. It makes use of the ultrasonic and light sensors to complete this task. This task is mapped on a goal-tree diagram as shown in **Figure 9**.

Any deviation in full-filling the OR goals and corresponding AND sub-goals is considered as a mission failure.

8. Aspects Evaluation via Software Metrics

Although software metrics like coupling, cohesion and size has been used to access the software quality for quite some time, yet the separation of concerns especially cross cutting ones with the aid of aspect oriented software development demands some additional metrics suite for its assessment. In this regard [14-16] have proposed additional metric suite for separation of concerns. This metric suite has been utilized in [17] to access the quality of some large scale software systems.

These additional metrics measure the degree to which a single concern in the system maps to the design components (classes and aspects), operations (methods and advice), and lines of code. For all the employed metrics, a lower value implies a better result. Some of these metrics used in our study are explained below.

8.1 Separation of Concerns Metrics

Separation of concerns (SoC) refers to the ability to identify, encapsulate and manipulate those parts of software that are relevant to a particular concern. The metrics for

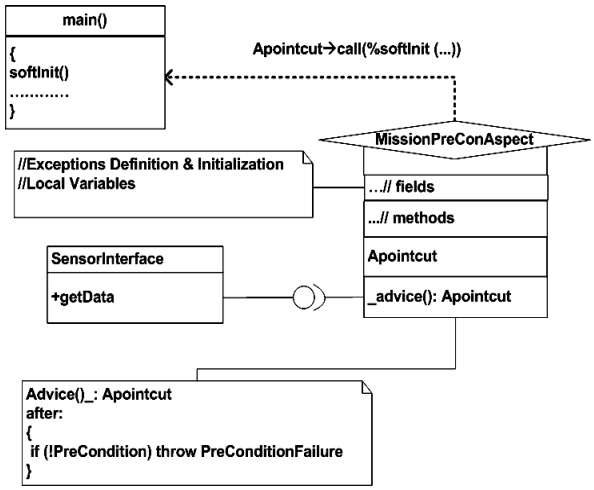


Figure 8. Mission pre-condition aspect

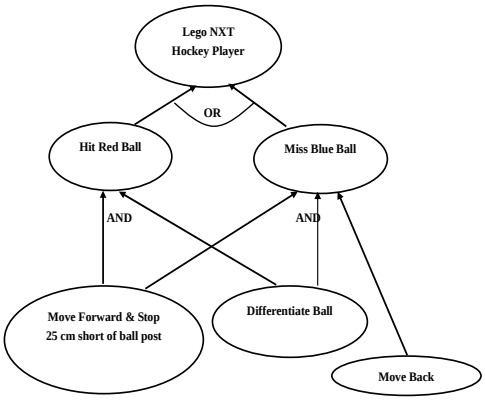


Figure 9. Lego NXT robot case study: Goal tree diagram

SoC measurement are:

Concern Diffusion over Components (CDC)

This metric measures the degree to which a single concern in the system maps to the components in the software design. The more direct a concern maps to the components, the easier it is to understand. It is also easier to modify and reuse the existing components.

Definition: CDC is measured by counting the number of primary components whose main purpose is to contribute to the implementation of a concern. Furthermore, it counts the number of components that access the primary components by using them in attribute declarations, formal parameters, return types, throws declarations and local variables, or call their methods.

Concern Diffusion over Operations (CDO)

One way of measuring the code tangling is by counting the number of operations affected by concern code. If a concern is scattered around more operations, it becomes harder to understand, maintain and reuse.

Definition: CDO is measured by counting the number of primary operations whose main purpose is to contribute to the implementation of a concern. In addition, it counts the number of methods and advices that access any primary component by calling their methods or using them in formal parameters, return types, throws declarations and local variables. Constructors also are counted as operations.

Concern Diffusion over LOC (CDLOC)

The intuition behind this metric is to find concern switching with in the lines of code. For each concern, the program text is analyzed line by line in order to count transition points. The higher the CDLOC, the more intermingled is the concern code within the implementation of the components; the lower the CDLOC, the more localized is the concern code.

Definition: CDLOC counts the number of transition points for each concern through the lines of code. The use of this metric requires a shadowing process that partitions the code into shadowed areas and non-shadowed areas. The shadowed areas are lines of code that implement a given concern. Transition points are the points in the code where there is a transition from a non-shadowed area to a shadowed area and vice-versa. An extensive set of guidelines to assist the shadowing process is reported in [15].

8.2 Coupling Metrics

Coupling is an indication of the strength of interconnections between the components in a system. Highly coupled systems have strong interconnections, with program units dependent on each other [14]. The larger the number of couples, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult. Excessive coupling between components is detrimental to modular design and prevents reuse. The more independent a component is, the easier it is to reuse it in another application [14]. The metrics in this category are Coupling between Components (CBC) and Depth of Inheritance Tree (DIT).

Coupling between Components (CBC)

This counts the coupling between classes, classes and aspects and between other aspects. It counts the classes used in attribute declarations *i.e.* C2 and C3 depicted in figure below. It also counts the number of components declared in formal parameters, return types, throws declarations and local variables. Moreover classes and aspects from which attribute and method selections are made are also included.

New coupling dimension are also defined in [14] in order to support aspect oriented software development (AOSD). For *e.g*. access to aspect methods and attributes defined by introduction (couplings C4, C5, C7, C8, C10), and the relationships between aspects and classes or other aspects defined in the pointcut (couplings C6, C9) as depicted in **Figure 10**. Thus overall this metric encompasses nine coupling dimensions (from C2 to C10). If a component is coupled to another component in an arbitrary number of forms, CBC counts only once.

Depth of Inheritance Tree (DIT)

DIT is defined as the maximum length from a node to the root of the tree. It counts how far down the inheritance hierarchy a class or aspect is declared. This metric encompasses the coupling dimensions C1 and C11 illustrated in **Figure 10**.

8.3 Lego NXT Software Measures Analysis

Software metrics are attained for the Lego NXT Robot case study as shown in **Figure 11**. In this case study, a C++ based true functionality has been made fault tolerant by weaving various concerns in 30 places using 7 aspects and 10 independent point cut expressions. These 7 aspects represent different concerns that otherwise may be added to actual true concern making the code more tangled, non maintainable and non reusable.

Separation of Concern Measures

Separation of concerns has been evaluated using CDC, CDO and CDLOC figures attained in the above case study.

The Concern Diffusion over the components (CDC) metrics measures the mapping of a single concern on various components. It can be inferred from **Figure 12** below that there is 64% reduction in mapping of true concern on the components present in the system due the introduction of aspects. Moreover the individual aspects implementing cross cutting concerns don't present large CDC figures that means, the aspects are loosely coupled with the system and thus can be more powerful candidates for reusability.

Same behavior has been observed in CDO measures as shown in **Figure 13**. As argued in [15,16], the code tang-

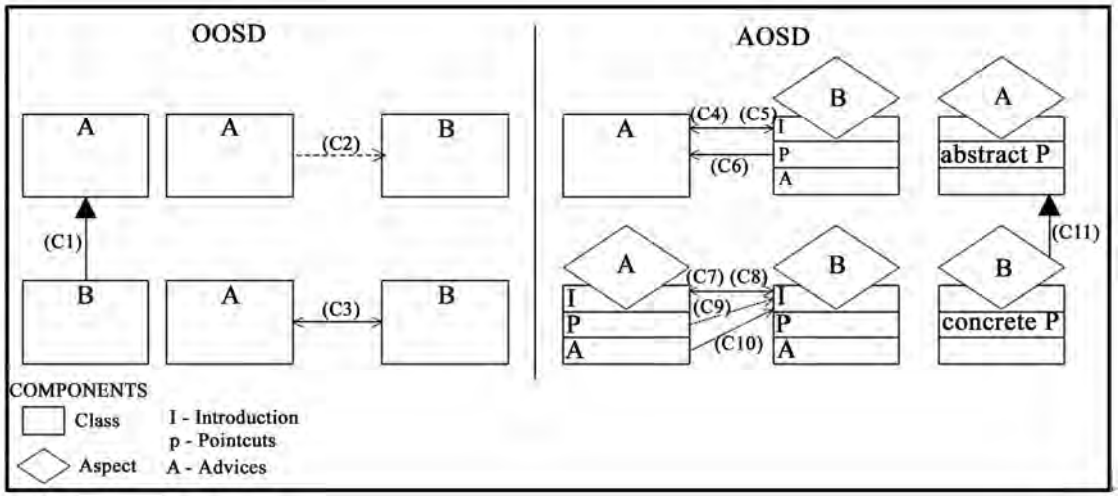


Figure 10. Coupling dimensions on AOSD [14]

Concerns	Type of Concerns (Cross Cutting Places)	Components		Operations				Exception Handling		Pointcut	Separation of Concerns			Couping
		Class/ Interface	Aspects	functions	Advice			Exception Definition & Throwing	try/catch blocks		CDC	CDO	CDLOC	
					Before	After	Around							
Tribot Case Study (TCS)	Functional Concern	3	0	17				0	0		7	17	2	3
Mission Pre-Condition (MPC)	Non-Functional (1 place)	1	1	2		1		2	0	1	3	3	0	3
System Configuration and Initialization (SCI)	Cross-Cutting (2 places)	1	1	1	2			0	0	2	2	3	0	2
Save Data (SD)	Cross-Cutting (5 places)	2	1	4	1			0	0	1	2	3	0	3
WatchDog (WD)	Cross-Cutting (5 places)		1	1	1			1	0	1	1	3	0	1
ROC Plausibiltiy Check Error Detection (ROC)	Cross-Cutting (2 places)		1			1		4	0	1	1	1	2	1
Exit-Main Catcher Handler (EMH)	Cross-Cutting (1 place)		1	2			1	0	1	1	1	3	1	1
Return Caller Catcher Handler (RCH)	Cross-Cutting (14 places)		1				2	0	2	3	2	7	1	3
True Functionality (TF)	0	3	0	17	0	0	0	0	0	0	7	17	2	3
Cross-Cutting Concerns (CC)	30	4	7	10	4	2	3	7	3	10	12	23	4	14

Figure 11. Lego NXT software metrics

ling may be visualized by observing the diffusion of a concern in different operations. Again the true concern seems more tangled in different operations as compared to cross cutting concerns implemented as aspects.

Concern diffusion over the lines of code (CDOLC) is a measure of how much tangled and inter-winded is the code implemented for a component. The larger the value, more tangled is the code with other concerns.

CDLOC for the core functionality (TCS) counts to 2 that seems a reasonable reduction as compared to non aspect oriented implementation. The seven non-functional/cross cutting concerns may add to present a larger CDLOC (**Figure 14**).

It can also be observed from the CDLOC dispersion that there are some indicators of bit code tangling with the true functionality especially for the aspects responsible for error detection and exception handling. Upon code reviewing it was observed that some critical contextual information is required that resulted in concern switching. Apart from that, overall concern switching for each component is reasonably small to be considered better candidates for reusability and maintainability. There were four concerns implemented with null concern switching in this study.

Coupling Measures

As observed in the study by [17], the coupling between components for various concerns has not increased a lot in our case as well. Coupling between components seem to be uniformly distributed with an average value of 2 as shown in **Figure 15**. Apart from core functionality (TCS), the increased coupling has been observed in the concerns implementing error detection and recovery

mechanisms. This is due to the fact that aspects implementing these concerns are coupled with the core concern for acquiring contextual information used in error detection and recovery mechanisms.

Exception Throwing & Handling Measures

It can be seen from **Figures 16(a)** and **16(b)** that exceptions definition and throwing have been localized in the components responsible for error detection like ROC plausibility Checks and Watch Dog. Moreover, exception handling has also been localized in their respective aspects without diffusing any other component.

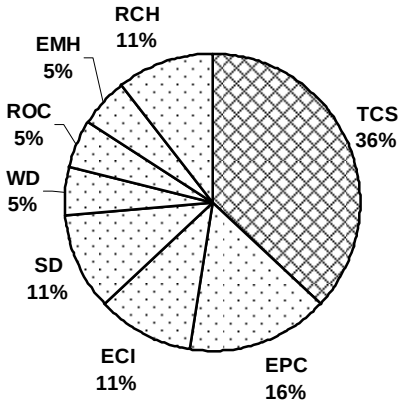


Figure 12. CDC dispersion

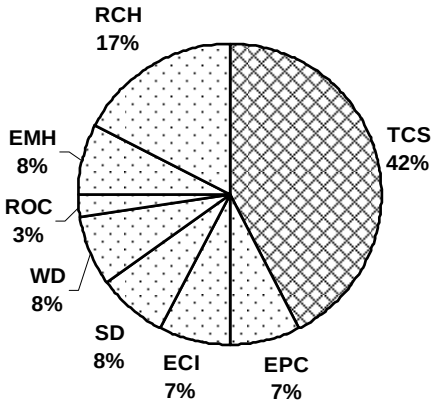


Figure 13. CDO dispersion

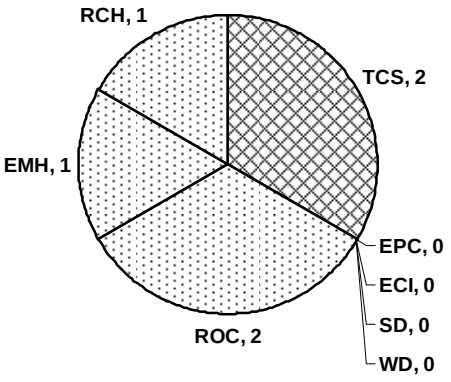


Figure 14. CDLOC dispersion

9. Conclusions & Future Work

The current work proposes AO design patterns for developing fault tolerant and robust software applications.

The aspect oriented design patterns under this framework bring additional benefits like the localization of error handling code in terms of definitions, initializations and implementation. Thus error handling code is not duplicated as the same error detection and handling aspect is responsible for all the calling contexts of a safety critical function. Reusability has also been improved because different error handling strategies can be plugged in separately. In this way, aspect and functional code may both be ported more easily to new systems.

Although a detailed analysis of concerns separation through aspects by refactoring large scale software ap-

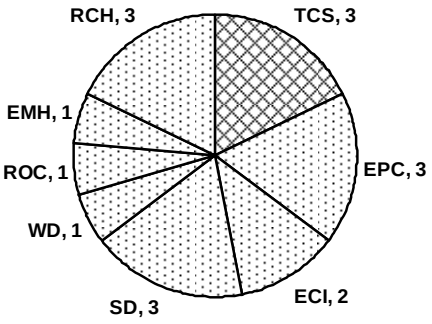
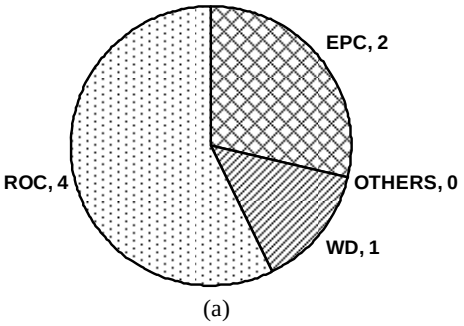
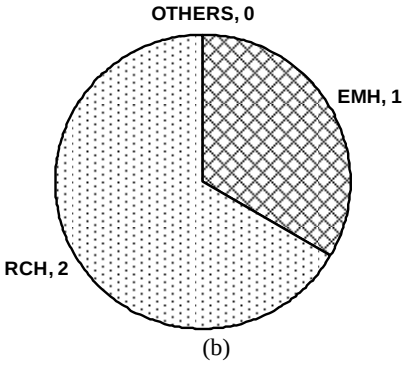


Figure 15. Coupling between components



(a)



(b)

Figure 16. (a) Exceptions define & throw; (b) Try/catch blocks

plication has been provided in [17]. Our case study also compliments some of the results. It has been observed that localization of exception management (definition, initialization and throwing) and exception handling improves modularity. It has been observed that fault tolerance concerns when implemented as aspects have resulted in considerable reduction in diffusion of concerns over the core functionality. The concern diffusion in terms of LOC does indicate clear separation and localization of error management related issues. However some code tangling has been observed with error detection based aspects. This is due to the sharing of context information needed for detecting erroneous states. Apart from that CDLOC measures too small in the true functionality. Coupling has been increased in the components responsible for error detection. Thus overall there has been an improvement in separation of concerns at the cost of slightly increased coupling.

This further probes the need for incorporating an error masking strategy like Recovery Blocks and N-Version Programming. An aspect oriented design version of these strategies is also under consideration.

REFERENCES

[1] M. Hiller, *et al*., "Executable Assertions for Detecting Data Errors in Embedded Control Systems," *Proceedings of the International Conference on Dependable Systems & Networks*, New York, June 2000, pp. 24-33.

[2] M. Hiller, "Error Recovery Using Forced Validity Assisted by Executable Assertions for Error Detection: An Experimental Evaluation," *Proceedings of the* 25*th EUROMICRO Conference*, Milan, Vol. 2, September 1999, pp. 105-112.

[3] P. A. C. Guerra, *et al*., "Structuring Exception Handling for Dependable Component-Based Software Systems," *Proceedings of the* 30*th EUROMICRO Conference* (*EUROMICRO*'04), Rennes, 2004, pp. 575-582.

[4] A. F. Garcia, D. M. Beder and C. M. F. Rubira, "An Exception Handling Software Architecture for Developing Fault-Tolerant Software," *Proceedings of the* 5*th IEEE HASE*, Albuquerque, November 2000, pp. 311-332.

[5] AspectJ Project Homepage. http://eclipse.org/aspectj/

[6] AspectC++ Project Homepage. http://www.aspectc.org

[7] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, June 1994, pp. 476-493.

[8] V. Basili, L. Briand and W. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, October 1996, pp. 751-761.

[9] K. Hameed, R. Williams and J. Smith, "Aspect Oriented Software Fault Tolerance," *Proceedings of* 4*th International Conference on Computer Science & Education* (*WCE*09), London, Vol. 1, 1-3 July 2009.

[10] L. L. Pullum, "Software Fault Tolerance Techniques and Implementation," Artech House Inc., Boston, London, 2001.

[11] F. C. Filho, *et al.*, "Error Handling as an Aspect," *Workshop BPAOSD*'07, Vancouver, 12-13 March 2007.

[12] A. Romanovsky, "A Looming Fault Tolerance Software Crisis," *ACM SIGSOFT Software Engineering Notes*, Vol. 32, No. 2, March 2007, p. 1.

[13] K. Murata, R. N. Horspool, E. G. Manning, Y. Yokote and M. Tokoro, "Unification of Compile-Time and Run-Time Metaobject Protocol," *ECOOP Workshop in Advances in Meta Object Protocols and Reflection* (*Meta*'95), August 1995.

[14] C. Sant'Anna, *et al*., "On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework," *Proceedings of the* 17*th Brazilian Symposium on Software Engineering*, Salvador, October 2003, pp. 19-34.

[15] A. Garcia, *et al.*, "Agents and Objects: An Empirical Study on Software Engineering," Technical Report 06-03, Computer Science Department, PUC-Rio, February 2003. ftp://ftp.inf.puc-rio.br/pub/docs/techreports/(file03_06_garcia.pdf)

[16] A. Garcia, *et al.*, "Agents and Objects: An Empirical Study on the Design and Implementation of Multi-Agent Systems," *Proceedings of the SELMAS*'03 *Workshop at ICSE*'03, Portland, May 2003, pp. 11-22.

[17] F. C. Filho, *et al*., "Exceptions and Aspects: The Devil in Details," *Proceedings of the* 14*th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Portland, 5 November 2006.

J. Software Engineering & Applications, 2010, 3: 312-330
doi:10.4236/jsea.2010.34037 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Specifying the Global Execution Context of Computer-Mediated Tasks: A Visual Notation and a Supporting Tool

Demosthenes Akoumianakis

Department of Applied Informatics & Multimedia, School of Applied Technologies, Advanced Technological Education Institution of Crete, Crete, Greece.
Email: da@epp.teicrete.gr

Received August 4th, 2009; revised September 2nd, 2009; accepted September 14th, 2009.

ABSTRACT

This paper presents the notion of the global execution context of a task as a representational construct for analysing complexity in software evolution. Based on this notion a visual notation and a supporting tool are presented to support specification of a system's global execution context. A system's global execution context is conceived as an evolving network of use scenarios depicted by nodes and links designating semantic relationships between scenarios. A node represents either a base or a growth scenario. Directed links characterize the transition from one node to another by means of semantic scenario relationships. Each growth scenario is generated following a critique (*or screening*) *of one or more base or reference scenarios. Subsequently, representative growth scenarios are compiled and consolidated in the global execution context graph. The paper describes the stages of this process, presents the tool designed to facilitate the construction of the global execution context graph and elaborates on recent practice and experience.*

Keywords**:** *Non-Functional Requirements, Software Evolution Artifacts, Global Execution Context, Tools*

1. Introduction

Over the years a plethora of techniques have been developed to manage functional requirements of a software system. Some of these techniques focus on modeling and implementing functional requirements using constructs such as goals, scenarios, use cases, or notations such as UML diagrams and dedicated UML profiles. More recent efforts shift the focus to packaging and deploying functional requirements as reusable components and Web services for program-to-program interactions. In particular, service-oriented architectures (SOAs) appropriate the benefits of Web services to make it easier to exploit software assets from many types of components in sophisticated new solutions, without complex integration projects. Nevertheless, in all cases current thinking is dominated by concerns focusing on the lower levels of an enterprise infrastructure—how to create, manage and combine business services providing data and logic. These efforts and the supporting techniques to managing functional requirements are characteristic of the prevailing paradigm in software development, which can be broadly qualified as construction-oriented. At the core of this paradigm is the goal of designing what a software system is expected to do, and to this end, the software design community has faced a variety of challenges in an effort to provide insights to the process of constructing reliable, robust and useful interactive systems and services.

The advent and wide proliferation of the Internet and the WWW have expanded an already over-populated software design research agenda, bringing to the surface the compelling need to account for a variety of non-functional requirements such as portability, accessibility, adaptability/adaptivity, security, scalability, ubiquity etc. Some of them are well known to the software engineering community, while others challenge established engineering methods and work practices. For instance, a long-standing premise of user-centered development is that of 'understanding users'; but users are no longer sharply identifiable, homogeneous or easily studied [1]. Furthermore, the tasks users carry out keep changing both in type and scope [2], with every new generation of technology, from desktop systems to mobile and wearable devices and the emergence of ubiquitous environments. The radical pace of these technical changes and the proliferation of myriad of network attachable devices introduce novel contexts of use, requiring insights, which are frequently beyond the grasp of

software designers.

Recognition of these challenges has motivated recent calls for departing from construction-specific software design techniques towards evolution-oriented methods and tools. In this vein there have been proposals aiming to provide creative interpretation of best practices (e.g., by devising new modeling constructs [3,4], building dedicated UML profiles [5], specifying architectural pattern languages [6], etc.) in an effort to establish mechanisms for analyzing and/or abstracting from salient features of software artifacts. Despite recent progress, designing systems to cope with change and evolution remains a challenge and poses serious questions regarding the design processes needed, the appropriate methodology and the respective instruments. One research path aiming to establish the ground for such informed design practices concentrates on non-functional requirements (NFRs) as a means to shift the focus away from what a software system is expected to do towards how it should behave under specified conditions. NFRs or quality attributes represent global constraints that must be satisfied by the software. Such constraints include performance, fault-tolerance, availability, portability, scalability, abstraction, security and so on. Despite their recognition by the software engineering community, it is only recently (*i.e.*, in the early 90s) that researchers have embarked in efforts aiming to assess their relevance to and implications for software development [3,4,7]. Nevertheless, in contrast to functional requirements their non-functional counterparts have proven hard to cope with for a variety of reasons [8]. Firstly, most of them lack a standard connotation as they are being treated differently across engineering communities and software development disciplines (*i.e.*, the same or similar NFRs hold different meaning for say, platform developers and usability experts). Secondly, they are abstract, stated only informally and requiring substantial context-specific refinement to be accounted for. Thirdly, their frequently conflicting nature (e.g., scale of availability may conflict with performance) makes step-by-step implementation or verification of whether or not a specific NFR is satisfied by the final product, extremely difficult. These are some of the reasons why NFRs are not easily incorporated into standard software engineering tools and practice.

SOA provide a new context for revisiting several NFRs and their management during software development. Nevertheless, current efforts are almost exclusively concentrated on qualities such as abstraction, messaging, service discovery, data integration, security, service orchestration/composition, etc, to facilitate two key principles: 1) creation of business services with defined interfaces so that functionality can be built once and then consumed as required and 2) separation of the provision of the services from their consumption. In this endeavor, the software design community continues to devise abstractions (*i.e.*, components, visual notation, models and tools) which make construction-oriented artifacts first-class objects, dismissing or undermining software evolution and the special value NFRs have in this context. On the other hand it is increasingly recognized that software evolution is steadily overtaking in importance software construction. Indeed, Finger [9] argues that '…the ability to change is now more important than the ability to create systems in the first place'.

An alternative approach to address this challenge may be grounded on establishing the appropriate level of abstractions to make evolution (rather than construction) artifacts explicit, traceable and manageable in the course of software development. This implies devising abstractions that allow us to expose how change is brought about, what it entails, how it is put into effect and how it may be traced and managed. To meet this goal, there are key milestones likely to catalyze future developments. Our understanding of this challenge leads to the conclusion that we need 1) modeling approaches directing analysis towards early identification of components that relate to the cause of change, the subject of change and the effect of change and 2) tools for effecting change in a compositional fashion, thus relating change to local components which are assembled without requiring global reconfiguration of the system. In this vein, the present work considers change management at a new level of abstraction by promoting a shift in the unit of analysis from task- or activity-level to task execution contexts. It is argued that managing change is synonymous to coping with complexity and entails a conscious effort towards designing for the global execution context of computer-mediated tasks. Our normative perspective is that software designers should increasingly be required to articulate the global execution context of a system's tasks, rather than being solely concerned with the development of an abstract task model from which incrementally, either through mappings or transformations, a platform-aware version of the system is generated. Moreover, designing for the global execution context is a goal to be satisficed rather than fulfilled. To this end, a new method and a supporting tool is described which allow designers to reason proactively (*i.e.*, from early concept formation through to design, implementation and evaluation) about the global execution context of designated tasks. Both the method and the tool provide a step in the direction of making change a first-class design object accounted for explicitly by articulating the parameters likely to act as 'drivers' of change. This is facilitated by an analytical approach aiming to unfold, identify, represent and implement alternative computational embodiments of tasks suitable for a range of distinct task execution contexts considered relevant and appropriate.

The remainder of the paper is structured as follows. The next section considers change management in in-

formation systems and frames the problem in two theoretical strands relevant to this work, namely change as evolution and change as intertwining non-functional requirements. This contrast offers useful insight to some of the research challenges preventing the development of methods for effectively coping with changes. Then, the paper elaborates on and defines the notion of a system's global execution context, which forms an abstraction for addressing complexity in software evolution rather than software construction. The following section describes the i-GeC tool, which allows incremental specification of the global execution context by using scenarios. Our reference example is a light ftp application initially designed for desktop use. The paper is concluded with a discussion on the contributions of this work and a brief note on implications and future work.

2. Motivation and Related Work

Change in interactive software is inherently linked with the context in which a task is executed. Typical context parameters include the target user, the platform providing the computational host for the task and/or the physical or social context in which the task is executed. Each may give rise to a multitude of potential drivers for change. Therefore, it stands to argue that change management is about coping with complexity in construction as well as in evolution. Managing complexity in construction has been coined with the handling of functionality. Specifically, through the history of software design the primary focus has been on accommodating functional requirements so as to develop systems that meet specific user goals. The resulting systems could cope with minimal and isolated changes, related primarily to the user, since no other part of the system's execution context (*i.e.*, platform or context of use) was conceived as viable to change. On the other hand, complexity in evolution is a more recent challenge attributed to the adoption of the Internet and the proliferation of Internet technologies and protocols. These developments have brought about an increasing recognition of the catalytic role of NFRs and have necessitated a paradigm shift in the design of interactive software so as to explicitly account for quality attributes such as abstraction, openness and platform independence, interoperability, individualization, etc. Despite the fact that complexity in construction and complexity in evolution may seem as competing at first glance (*i.e.*, evolution frequently implies improvements in the functional requirements), our intention is to argue that they bring about complementary insights, which may prove beneficial to the development of systems which are easier to manage and use.

2.1 Change as Evolution

The term evolution generally refers to progressive change in the properties or characteristics of the subject of evolution (*i.e.*, software). A common view to conceive software evolution is to focus on mechanisms and tools whereby progressive change in program characteristics (e.g., functionality) and growth in functional power may be achieved in systematic, planned and controlled manner [10]. This may be conceived from various perspectives and viewpoints. For instance, it may be viewed from the perspective of software engineering processes and thereby explain the proliferation of iterative software development models such as agile programming [11] and extreme programming [12], etc. It may also be related to evolution in requirements and requirements management [13], giving rise to methods for tracing evolving components [14,15], localizing changes to components and developing component interoperation graphs [16], framing change to scenarios and supporting scenario evolution [17,18], etc. Whatever the perspective adopted, it is widely accepted that the ability to change is now more important than the ability to create systems in the first place [9]. Change management becomes a first-class design goal and requires business and technology architecture whose components can be added, modified, replaced and reconfigured. The implication is that the complexity of software has definitely shifted from *construction* to *evolution*. As a result new methods and technologies are required to address this new level of complexity.

In the past, the software design community addressed complexity in construction by devising abstractions (*i.e.*, components, visual notation, models and tools) to make construction-oriented artifacts first-class objects of design. This paradigm has catalyzed developments and facilitated breakthroughs in areas such as: 1) data management (leading from the early conception of the relational model to more recent proposals [19]), 2) software design (progressively shifting from structured techniques to object orientation [20], the development of computer-aided software engineering tools [21,22], domain-specific design languages [23], architecture description languages [6] and software factories [24]), 3) user interface development (facilitating richer interactions with the advent of 2D and 3D graphical toolkits [25-28]), etc. The common theme in these developments is that complexity is addressed by establishing levels of abstraction, allowing software construction artifacts (expressed as models of some sort, code, or processes) to become first class objects. It can therefore be argued that the problem of complexity in software evolution amounts to establishing the appropriate level of abstraction to make evolution artifacts explicit, traceable and manageable. That is to say, we need to find the abstractions that allow us to expose how change is brought about, what it entails, how it is put into effect and how it may be traced and managed. As already stated, our understanding of this challenge leads to the conclusion that we need 1) modeling approaches directing analysis to the identifica-

tion of components that relate to the cause of change, the subject of change and the effect of change and 2) tools for effecting change in a compositional fashion, thus relating change to local components which are assembled without requiring global reconfiguration of the system.

2.2 Change and Non-Functional Requirements

Change as evolution of functional requirements is of course valid but it can only explain partially why modern information systems need to change. In fact, there is evidence to suggest that most of the changes in modern information systems do not concern functional components but their connections and interactions [9]. This explains recent efforts aiming to frame change in relation to NFRs [8] and architectural quality attributes [29,30] such as such as adaptability [31], portability [32], run-time adaptive behavior [2,33], etc. Through these efforts, it becomes increasingly evident that NFRs concern primarily environment builders rather than application programmers. Nevertheless, there is an equally strong line of research aiming to make NFRs visible and accountable for as early as possible in the software design lifecycle by developing conceptual models and dedicated notations. Specifically, there are techniques aiming to classify NFRs through taxonomies [34], develop representational notations for using NFRs [7], advance process-oriented instruments for working with them [7,29] and study their relationship to software architecture [30,35]. Although these techniques have evolved in separate engineering communities (each with its own point of view) and have typically been performed in isolation, they share common ground. For instance, they recognize the important role to be played by methodological concepts and supporting technology that promote architectural insight through suitable first-class objects. Phrased differently, software architecture quality attributes promote a gross decomposition of systems into *components* that perform basic computations and *connectors* that ensure that they interact in ways that make required global system properties to emerge. Thus establishing abstractions at the level of software architecture may help manage complexity in software evolution.

2.3 Framing Change to the Task's Execution Context

The present work focuses on treating change from the early stages of information systems development where a variety of critical decisions are taken regarding architecture, platforms, tools to be used, expected and foreseen behaviors. Our primary concern is to advance a proposal rooted in the anticipation of change and its incremental localization in components. To this end, we use the notion of task execution context to define a particular type of scenarios that are allowed to grow. Then the global execution context of a task (or a piece of functionality) is an instance in a continuum of revisions and extensions of the designated task's execution context. Consequently, managing change amounts to a conscious effort towards designing for the global execution context of computer-mediated tasks.

The execution context of a task is understood in terms of a triad <Users, Devices, Context>. Users represent the end (target) users - individuals or communities of users - who experience an interactive artifact through which the task is carried out. Devices refer to the technological platform used to provide the computational embodiment of the interactive artifact. Finally, context is a reflection on the (physical and social) context of use in which the task is executed. It is worth noting that none of these relate to functional properties of the task. Then, designing for the global execution context of a particular task with specified functional requirements is directly related to unfolding the rationale for and the artifacts encountered in a range of plausible task execution contexts. Interpreting the above rather theoretical concept in terms of practical design guidelines raises several issues with two standing out very promptly. The first is the commitment towards exploring and managing complex design spaces, while the second is the shift of engineering practice towards abstract and specification-based techniques. Although neither is entirely new to the software design community (e.g., see the works by MacLean [36] on design space analysis, the work on DRL by [37], etc, as well as recent advances in device-independent mark-up languages such as UIML (http://www.uiml.org/) and model-based development tools such as Teresa [32]), their meaning and exploitation is slightly different in the context of the present work.

3. The Global Execution Context of Tasks

The premise of the present work is that software design lacks a coherent and detailed macro-level method – in the sense defined in [38] – for the management of change during the early stages of development where critical decisions on architecture, tools and platforms to be used, are taken. Consequently, our interest is in establishing an integrated frame of reference for identifying and propagating change (*i.e.*, new requirements or evolution in requirements) across stages in the course of the design and development processes so as to facilitate designing for the global execution context (GeC) of tasks.

3.1 Motivating Example & Terminology

It is useful to conceive the global execution context of a task as a space of transformations depicting possible and/or desirable mappings of a task's abstraction to alternative non-functional contexts (*i.e.*, interaction platforms, contexts of use or user profiles). **Figure 1** provides an illustrative example from the field of user inter-

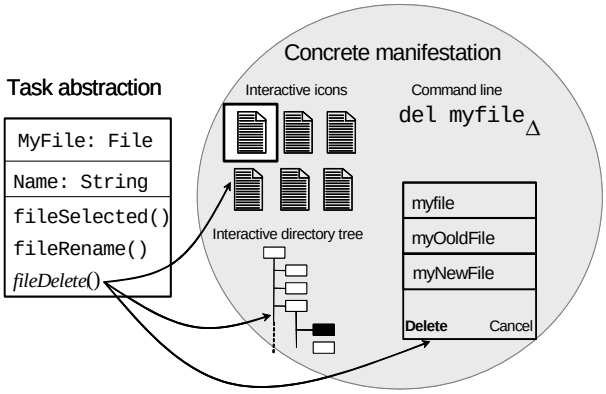


Figure 1. Possible transformations to derive the global execution context of a task

face engineering. Specifically, the figure presents schematically possible transformations for an abstract task 'fileDelete' of a hypothetical file management application to distinct concrete manifestations (potentially) suitable for different non-functional execution contexts. It is worth noticing that the example presents a case that challenges current conceptions of cross-platform or portable software in the sense that concrete manifestations need not be bound to native platform-specific elements; instead, they may use customized facilities, domain-specific and/or expanded components.

Being able to explicitly foresee and design a system so that it can cope with all possible changes in its execution context is probably a utopia, given the current state of the art in systems thinking and engineering. Nevertheless, if we delimit the qualification 'all possible changes in the task's execution context' to all known, or foreseen and explicitly modeled changes (within the scope of a service-oriented architecture), then it is possible to define a context-sensitive processing function which under certain circumstances will deliver the maximally preferred transformation of an abstract task to a concrete instance [39]. Consequently, understanding and designing for the global execution context of a system's task (*i.e.*, supporting the task's execution across all designated non-functional contexts) entails some sort of mechanism or service for linking to, rather than directly calling, different implemented components complying to/supported by a designated service-oriented architecture. Equally important is to consider how the service-oriented architecture is to view and link to radically different execution contexts and platforms. In recent wittings, both in research and development communities, this dimension is dismissed resulting in proposals for SOA that cannot cope with radically different non-functional execution contexts.

To provide further insight, let us abstract from the details of **Figure 1** to describe a more general situation where our abstract task T_2 (*i.e.*, delete a file) is assigned to two distinct realizations as shown in **Figure 2**. The first, denoted with the solid line, refers to task execution on a desktop devise, which requires that the selection list is presented (S_1), the user makes a choice (S_2) and subsequently the command is issued (S_3), followed by a confirmation dialogue (S_4). The second realization is using a mobile devise. Once again the selection list is presented (S_1), but this time in order for the user to make a choice the system augments interaction initiating a scanning interface S_2'. Once the selection is made the command is issued (S_3) followed by a confirmation dialogue (S_4).

It is worth pointing out that despite the simplicity of the example, it poses several challenges. First of all, for any given task one can easily identify several additional realizations (execution contexts) depending on the platform or toolkit, the context of use and/or the target user. Thus, one issue is enumerating requirements and encoding alternatives, but also allowing for incremental updates and evolution to accommodate new realizations. Secondly, irrespective of the task's execution context the functional requirement remains the same (*i.e.*, delete a file). The cause of change is therefore due to a designated set of NFRs. It may also be argued that prevalent NFRs such as portability or platform independence may not suffice to capture the essence implied by some of these changes. For instance, if scanning is implemented as a reusable interaction library, it signifies an *augmentation* of the target platform whereby the scanning functionality is introduced as new interaction technique assigned to designated interaction elements. This is totally different from a hard-coded implementation of the scanning inter face to suit a specific interaction scenario or system. Similarly, one could envisage alternatives to augmentation such as platform *expansion* (*i.e.*, to increase the range of interaction elements of an existing platform) or *new platform development* (*i.e.* for a designated modality) and *integration* (*i.e.*, mixing components from different platforms). All these represent intertwining (and frequently conflicting) goals to inscribing non-functional qualities such as usability, portability, individualization, etc. Moreover, they are not intuitively associated with or assumed by prevalent NFRs. It stands to argue therefore that designing for the global execution context of a task

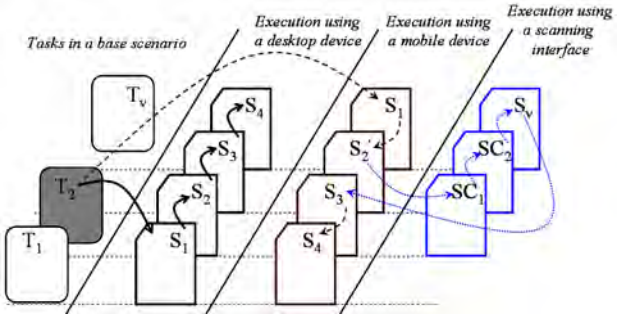


Figure 2. Tasks and execution contexts

entails an account of 'hidden' quality goals such as platform augmentation, expansion and integration, which are not so established in the software engineering literature. Furthermore, in many cases it is these 'hidden' quality goals that determine the type and range of non-functional contexts to be assigned to a task's global execution context.

3.2 Modelling the Global Execution Context

Conceptually, the GeC of an abstract task T can be conceived as a five tuple relation <T, g, S, f, C> where g is the task's goal to be achieved by alternative scenarios $s_i \in S$, and a context-sensitive processing function $f(s_i)$ which defines the maximally preferred instance of S, given a designated set of constraints C. Such a definition, allows us to model change in interactive software in terms of certain conditions or constraints which propagate alternative interactive behaviours to achieve task-oriented goals. Three types of constraints are relevant to the present work, namely user constraints, platform constraints and context constraints. User constraints are user-specific parameters designating alternative interaction and use patterns. Platform constraints relate to properties of a target device-specific execution environment. Context constraints designate external attributes of potential relevance to the task's execution. Then, change δ in the execution context of a software system occurs if and only if there is at least one constraint in C whose parameter value has been modified so as to justify system transformation. The result of recognizing δ and putting it into effect causes the deactivation of $s_i \in S$, which was the status prior to recognizing δ and the activation of a new $s_j \in S$, which becomes the new status.

3.2.1 Basic Vocabulary and Notation

In our recent work, we have been developing a scenario-based approach in an attempt to formalize elements of the global execution context of computer-mediated tasks. In terms of basic vocabulary, the approach makes use of three constructs namely base (or reference) scenarios, growth scenarios and scenario relationships. Base scenarios depict situations in an existing system or a prototype, which are defined in terms of functionality. Growth scenarios extend reference scenarios in the sense that they describe new execution contexts for the functionality associated to the reference scenario.

Scenario relationships are used to capture semantic properties of the reference and growth scenarios. Two categories of relevant scenario relationships have been identified, namely those describing internal structure of scenarios in terms of components as well as those describing scenario realization. The former type of relationships is well documented in the literature (see [17]) and may be applied to any scenario independent of type (reference or growth). For the purposes of the present work we have found two such relationships as being useful, namely subset-of and preference/indifference. Subset-of is the relationship defining containment between two scenarios. It declares that the functions of a scenario S_i are physically or logically part of another scenario S_j. S_i is termed the *subordinate scenario*, and S_j is termed the *superior scenario*. The subordinate scenario always encapsulates part of the action in the superior scenario. It should be noted that the subset-of relationship does not entail inclusion in the sense that execution of the superior scenario is suspended until the execution of the subordinate scenario is complete. Instead, it implies the set-theoretic notion of subset where the actions of the subordinate scenario are contained within or are the same as the set of actions of the superior scenario. Preference designates the existence of a preference order for two subordinate scenarios S_i and S_j of a superior scenario. Preference is specified by a preference condition or rule. When executed, the preference condition should place candidate subordinate scenarios in a preference ranking (indifference classes), while the most preferred scenario (first indifference class) is the one to be activated. The preference relationship is useful for specifying the context-sensitive processing function which activates/deactivates scenarios at run-time.

Scenario realization relationships provide details of the mapping (or transformation) between reference and growth scenarios and are intended to capture evolution of a reference scenario into growth scenarios. In general, two properties dictate the evolution of a base scenario into a growth scenario. The first relates to temporal aspects of growth scenario execution, while the second depicts the resources demanded for realizing the growth scenario. In terms of temporal aspects of execution these can be modeled either by alternative or parallel execution. The resources demanded can be modeled by relationships such as (platform) augmentation, (platform) expansion and (platform) integration. As the latter two are special cases of the alternative relationship, platform augmentation is the third scenario realization relationship used to complete the global execution context graph in the context of the present work. The alternative relationship links two scenarios when each serves exactly the same goals and one and only one can be active at any time. Alternative is the main operator for specifying adaptability of a system with regards to a designated quality attribute (e.g., platform independence). As already stated, two scenarios designated as alternative may be realized either by platform integration (typical case of multi-platform capability) or by platform expansion which assumes interoperability between the platform and another platform or third-party libraries used to expand the initial vocabulary of the platform. Moreover, two alternative scenarios are considered as indifferent with regards to all other quality attributes except the ones designated in the

alternative declaration (see preference). Augmentation captures the situation where one scenario in an indifference class is used to support or facilitate the mostly preferred (active) scenario within the same indifference class. For instance the scanning interface scenario in **Figure 2** augments file management when executed using a mobile devise. In general, two scenarios related with an augmentation relationship serve precisely the same goal through different (but complementary) interaction means. Finally, parallelism refers to concurrent activation of scenarios serving the same goal. At any time two parallel scenarios preserve full temporal overlap. Parallelism may have two versions. The simplest version is when the scenarios utilize resources of the same platform. In this case the relationship is synonymous to concurrent execution of the scenarios (*i.e.* deleting a file using command line or an interactive directory tree) with full temporal overlap. The second version of parallelism is relevant when the scenarios utilize resources of different interaction platforms (toolkits). In this case, it is assumed that the two platforms are concurrently utilized and an abstract user interface can link with each one to make use of the respective interaction elements. This type of parallelism does not require interoperability between the platforms, as platform-specific interaction elements are not mixed. A typical example of this type of parallelism is when two users (*i.e.* a blind and a sighted user) are engaged in a collaborative application (*i.e.*, file management session) and the concrete user interface in each case utilizes interaction resources of different toolkits (one graphical toolkit for the sighted user's interface and one non-visual toolkit realizing the blind user's interface). This type of parallelism is not common in interactive applications, but when properly supported, it can serve a number of desirable features such as adaptivity to suit individualized requirements, concurrent modality-specific interaction as well as multimodality.

It should be noticed that the relationships discussed above are intended to serve the analysis of the global execution context as described earlier. All of them except the subset-of relationship are intended to address primarily non-functional qualities of scenarios. Consequently, these relationships are complementary to others proposed in the relevant literature (see for example [17]) for capturing semantic properties such as scenario complements, specialization, temporal suspension of a scenario until another scenario is completed (*i.e.* 'includes' relationship) or exceptional scenario execution paths (*i.e.* 'extends' relationship).

3.2.2 The Global Execution Context Graph

Collectively, the notational constructs described earlier are presented in **Table 1** and constitute the basic vocabulary of the global execution context notation (GeCn). Using this notation, designers can specify the requirements of the global execution context of a task as a graph. This graph is typically a visual construction consisting of nodes representing scenarios and directed links representing scenario relationships. **Figure 3** illustrates an example of the global execution context graph of a task, namely select files, of a simple ftp application. This example will be further elaborated in the following section. The figure depicts, one reference scenario, namely 'Select files with desktop style' which through the containment operator links to 'Single file selection' and 'Multiple file selection'. Single and multiple file selection are parallel (*i.e.* weak notion of concurrency presented earlier, making use of resources of the same platform). The reference scenario as a whole (including the containments) is augmented with 'Select with scan on' which is a growth scenario containing two alternative options, namely 'One button/auto' and 'One button/manual' scanning. It is important to note that the designated growth scenarios do not represent change or evolution of the functional requirements of the application (either at the client or the server side). Instead, they designate a platform-specific non-functional requirement for supporting augmentation of interaction through scanning of interaction elements. On the other hand there is no pre-requisite as to how this augmentation is supported (*i.e.*, through programming or by augmenting toolkit libraries to facilitate scanning). This simple example suffices to make two claims regarding the global execution context graph of a task. Firstly, the technique is intended to represent 'hidden' requirements not commonly collected using conventional requirements engineering methods – thus it is complementary to rather than competing against such methods. Secondly, the technique is biased towards platform-oriented requirements leading to an improved insight on existing NFRs such as platform independence, portability, etc.

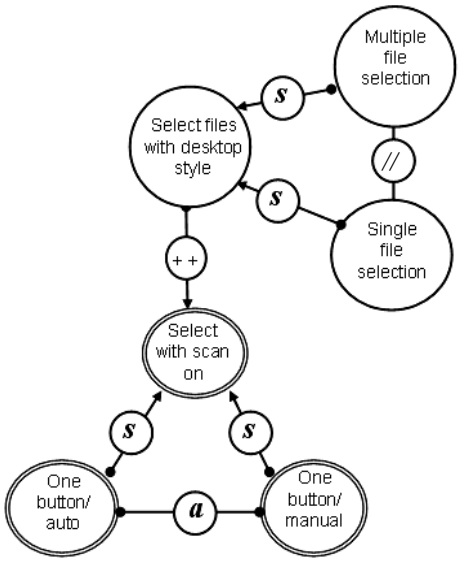


Figure 3. Example of a global execution context graph

Table 1. Basic notation

Symbol	*Interpretation*
A	Reference or base scenarios depict situations in an existing system or a prototype, which are defined in terms of functionality. They comprise at least one actor and one explicitly stated goal
B	Growth scenarios are always linked to a base scenario which they extend in the sense that they describe new execution contexts for the functionality associated to the reference scenario
— π —	'Preference' relationship designates the existence of a preference ranking between two or more scenarios; the preference ranking is conditional upon the preference rule (or condition)
— *i* —	'Indifference' relationship designates indifferent execution of two or more scenarios realized in the same design vocabulary (*i.e.* development platform)
— a —	'Alternative' relationship designates alternative realizations / embodiments of a scenario across distinct design vocabularies; at any time one and only one of the scenarios can be active
— ++ —	'Augmentation' relationship designates that a scenario is used to support or facilitate the mostly preferred (active) scenario within the same indifference class
— // —	'Parallel' relationship designates the concurrent activation of scenarios of a designated design vocabulary; at any time parallel scenarios preserve full temporal overlap
— ≈ —	'Interchangeable' execution designates parallel execution of scenarios in distinct design vocabularies; no requirement for interoperability as scenarios are not mixed
— s —	'Subset_of' defines containment of actions of one (subordinate) scenario into actions of another (superior) scenario; the subordinate scenario appears on the left hand side of the relationship

3.3 Stages in the Construction of Global Execution Context Graphs

Building the global execution context graph entails an iterative process of continuous refinement. It is both useful and important to be able to verify refinements of a task's global execution context graph so as to ensure consistency and correctness. To facilitate these tasks, a micro method and a supporting tool have been developed to provide guidelines for building the global execution context graph. The method and the tool serve two main goals, namely 1) encoding reference and growth scenarios in alternative representation forms and 2) incremental and evolutionary construction of the system's global execution context graph so as to allow incorporation of new requirements and requirements evolution (*i.e.*, versioning). **Figure 4** illustrates the conceptual stages involved in compiling the global execution context graph

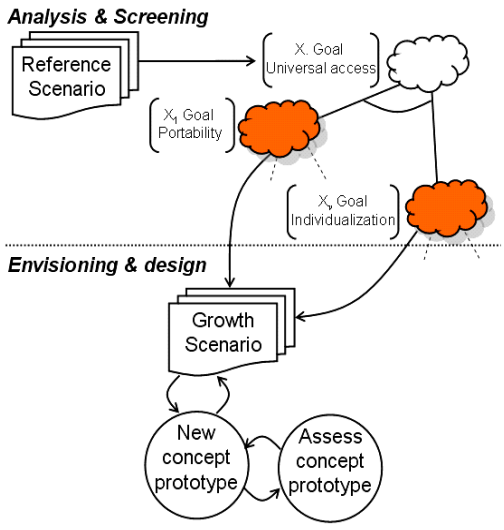


Figure 4. Process stages for building

using growth scenarios. As in the case of other scenario-based methods, it involves interplay between reflection (analysis and screening) and envisioning. The first step is usually a preparatory activity carried out by the analyst in collaboration with domain experts, and entails the formulation of reference scenarios. Once the scenario is formulated, typically in a narrative form, the screening process begins in an effort to compile the rationale for growth scenarios. To this end the choice of screening filters is important. One option is to screen the reference scenarios using designated NFRs so as to unfold breakdowns or deficiencies related to global system qualities (*i.e.* system architecture, platform commitment, interaction metaphor). In **Table 2** we provide an example of such NFRs-based screening of our reference ftp application.

Alternatively, screening may focus on other aspects of interactive software such as choice of interaction elements, dialogue styles, presentation, etc. In all cases, scenario screening assumes the availability of artifacts (e.g., narratives, pictures, user interface mock-ups, high fidelity prototypes, etc) and it entails a structured process whereby implicit or explicit assumptions embedded in an artifact (and related to the intended users, the platform used to implement the artifact and the context of use) are identified and documented. The essence of screening is in defining appropriate filters or adopting alternative perspectives to critique a base scenario. It is therefore a scenario inspection technique in the sense described in [40], which can be realized through different instruments.

Whatever choice of the screening instrument, it is imperative that screening should motivate growth scenarios so that the latter do not exist in vacuum. Instead, they should be related to the design breakdowns identified through screening and the designated new or evolving requirements. Consequently, the ultimate goal of growth scenarios is to capture evolution of requirements codified in base scenarios. Such evolution should depict new

Table 2. Screening using NFRs and corresponding design breakdowns

NFR		*Example of design breakdown*	*New or evolving requirement*
Platform independence		"… file transfer is not available as WWW or WAP application or service …"	*Allow choice of delivery medium or interaction style (i.e. HTML, WAP, Windows style)*
Scalability		"… the system does not exhibit scalability to platform or access terminal capabilities … "	*Detect context of use and allow operation in text-only style through a kiosk*
Adaptation	Adaptability	"…the system cannot be customized to diverse requirements…"	*Support manual or automatic customization of interaction style (e.g., scanning)*
	Adaptivity	"…when in operation the system does not monitor user's interactive behavior to adjust aspects of interaction…"	*Provide auditory feedback upon completion of critical tasks to inform users on task completion state*
Context awareness		'…the system takes no account of the context of use to modify its interactive behavior…"	*Allow context monitoring and switching between designated interaction styles*
Localization		"…the system can not be localized…"	*Allow choice of language*
Accessibility		"…the system is not accessible by certain target user groups…"	*Interview user to determine motor, visual, cognitive capabilities and define adaptation*

execution contexts for the tasks in the base scenario. In practice, growth scenarios result from relaxing the assumptions identified in the course of screening. Once agreed, growth scenarios may become more concrete through prototypes, which specify details of the new task execution contexts. It is important to note that our intention is to consider growth scenario management as an engineering activity [41] rather than a craft, and to contribute towards effective engineering practices for guiding the creation and refinement of scenarios. Consequently, our work links with recent proposals in scenario-based requirements engineering aiming to offer systematic scenario process guidance (see for example [42,43]) as well as key concepts and techniques of the Non-Functional Requirements Framework [8].

4. Designing for the Global Execution Context

To support designers in gaining insights and analysing the global execution context, a tool has been developed, namely interactive Global execution Context (i-GeC). i-GeC covers all three stages namely scenario recording, screening and growth scenario compilation. It does not however, embark into detailed design, which is beyond the scope of the present work. **Figure 5** depicts the logical view of i-GeC, summarising our notion of reference (or base) and growth scenarios as well as the scenario relationships relevant to this work. It should also be noted that the class model of **Figure 5** provides a scheme for interpreting the main components of the five tuple relation <T, g, S, f, C> used to conceptualise the GeC of a task. The only element not explicitly modelled is the context sensitive processing function *f*. However, this relates to the system's implementation and architectural model for processing (*i.e.* enabling/disabling) scenarios. As for the constraints they are assumed to be parameters of the class 'Artifact'. Another important consideration regarding the scheme of **Figure 5** is that, although there is a provision for goals, this should not be confused with functional requirements. In fact, this work is not concerned with this type of requirement. Instead, our interest is on non-functional requirements and how they are translated into quality goals. As already mentioned earlier, some non-functional requirements (*i.e.* adaptability, portability and individualization) are well established in the relevant literature both in terms of scope and techniques used to cope with them (*i.e.* [3,8]. Others however are not so well established (*i.e.* toolkit augmentation, expansion, integration) but are considered very important to modelling the global execution context of a task. The latter type of non-functional requirements, partly motivate the work presented in this paper.

To illustrate the above, we will continue to make use of our ftp application allowing authorised users to connect to a server and subsequently manipulate local files (*i.e.* transfer, delete). For the purposes of our discussion, we will consider both the incorporation of new requirements and requirements evolution. A new requirement is to support ftp portability to a new platform (*i.e.*, from desktop to PDA). As an example of requirements evolution we will consider various enhancements of the file selection task so as to support multiple selection by file category (*i.e.* select all files with an extension '.ppt') and selection through scanning on a PDA. Scanning is an interaction technique, which entails automatic manipulation of PDA interaction elements in a hierarchical fashion to reduce keystroke level actions. It is therefore conceived as a usability enhancement. Thus, in the remaining of this section, our aim is to show how from a given set of functional requirements we can progressively compile a specification of the system's new execution context supporting a PDA client with the enhanced file selection facilities.

4.1 Encoding/Recording Scenarios Using i-Gec

Encoding scenarios using a variety of media and representational tools is important, as it allows the designer to start with a high-level narrative description of a situation of use (see **Figure 6**, left hand side dialogue) and progressively transform it into a bulleted sequence, state diagram, use case model, etc., reflecting the designers' incremental improvement of understanding of the situa-

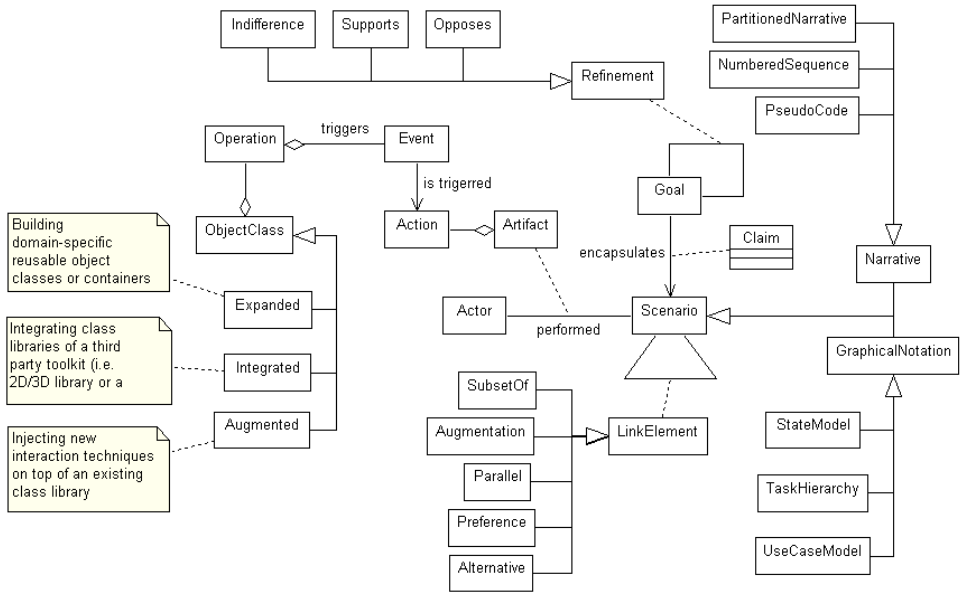


Figure 5. Class model of the global execution context of a task

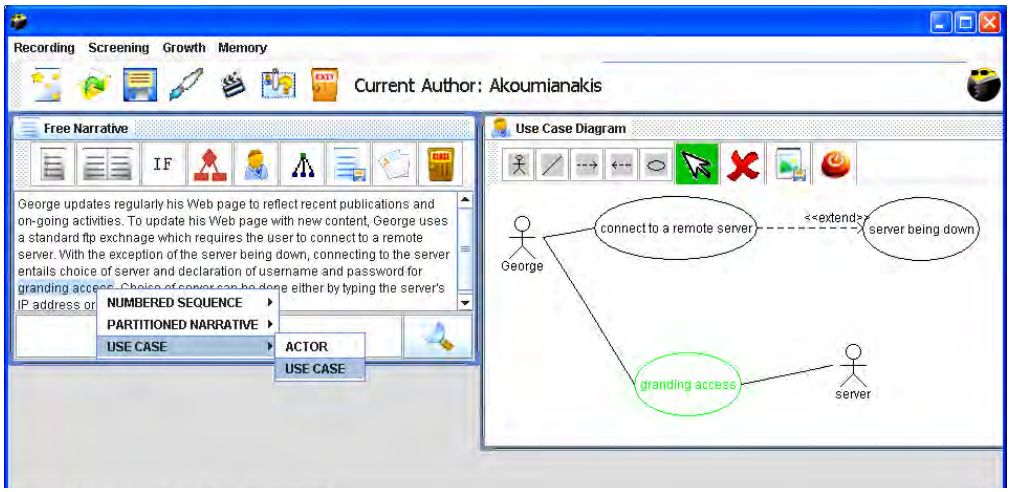


Figure 6. Encoding a reference scenario as use cases

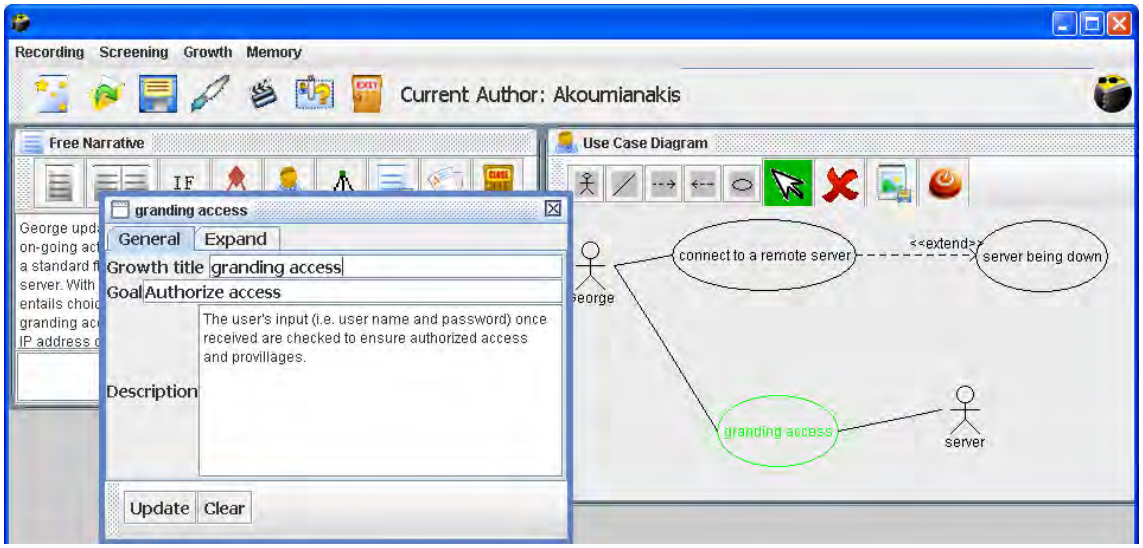
tion. This transformation is user-driven in the sense that the user can employee simple cut & paste techniques or menu-driven dialogues to map textual elements in the narrative description to graphical elements in a specific visual notation (see for example **Figure 6** for a transformation of a narrative to a use case model). Each reference scenario can be incrementally refined. Reference scenario refinement involves detailed description of the scenario and compilation of more analytic views of the scenario codified as numbered sequence of activities, partitioned narrative, exception steps, state transitions, etc., as shown in **Figure 7**.

4.2 Scenario Screening with i-Gec

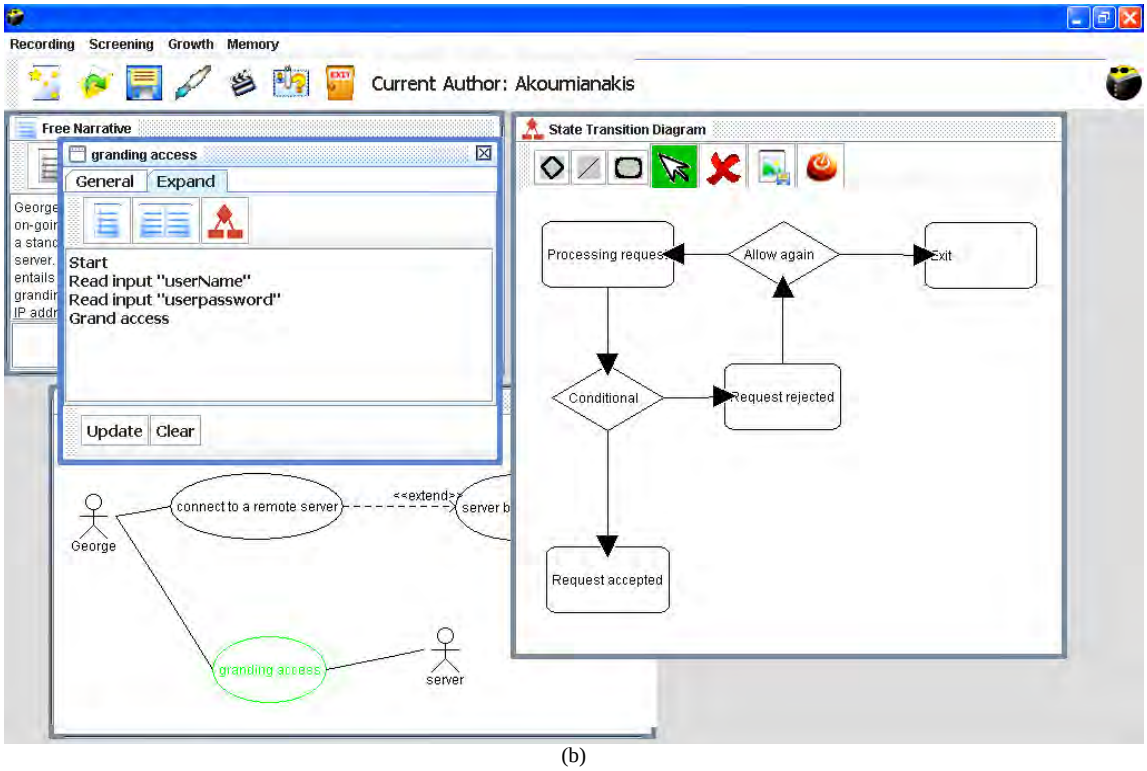
Following reference scenario recording, the screening stage seeks to provide a structured critique of the recorded scenario so as to designate issues (in anticipation of change) or shortcomings. These shortcomings provide the rationale and the motives for subsequent compilation of growth scenarios (see next section). In the current version, screening a scenario follows the tradition of design space analysis using Questions Options & Criteria [36]. The analyst can designate both issues and options (potential solutions) as shown in **Figure 8.** All designated issues and options are codified per scenario and can be explored through the memory tool. This type of screening is intended only to record and make persistent the results of analysis.

4.3 Compiling Growth Scenarios and Building Global Execution Context Graphs with i-GeC

In i-GeC, the compilation of growth scenarios entails



(a)



(b)

Figure 7. Manipulation of reference scenarios. (a) Describing the reference scenario; (b) Expanding the reference scenario

reformulation of a use case type representation of the base scenarios. Specifically, to define a growth scenario, the user should first declare the growth case and then assign the appropriate relationships between the growth case and the base or other growth scenarios. **Figure 9**. depicts an example where a growth scenario is introduced (**Figure 9(a)**) and subsequently elaborated (**Figure 9(b)**). In the example, 'iPAQ connection' is introduced as 'alternative to' the reference scenario 'connect to a remote server' which represents functionality already supported by the ftp application. The growth scenario is motivated by the screening criterion of 'user adaptability' and the issue 'how does the user type IP address' (see **Figure 9(b)**). For the same growth scenario there may be more issues assigned. As shown, reference and growth scenarios are distinct elements represented as single-line and double-line ellipses respectively. The semantic scenario relationship is represented as an annotated link. The

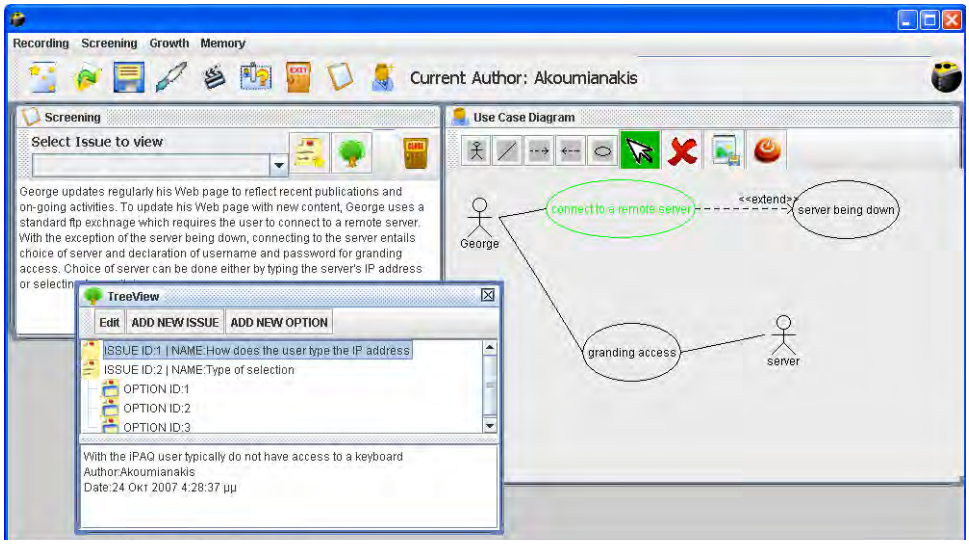
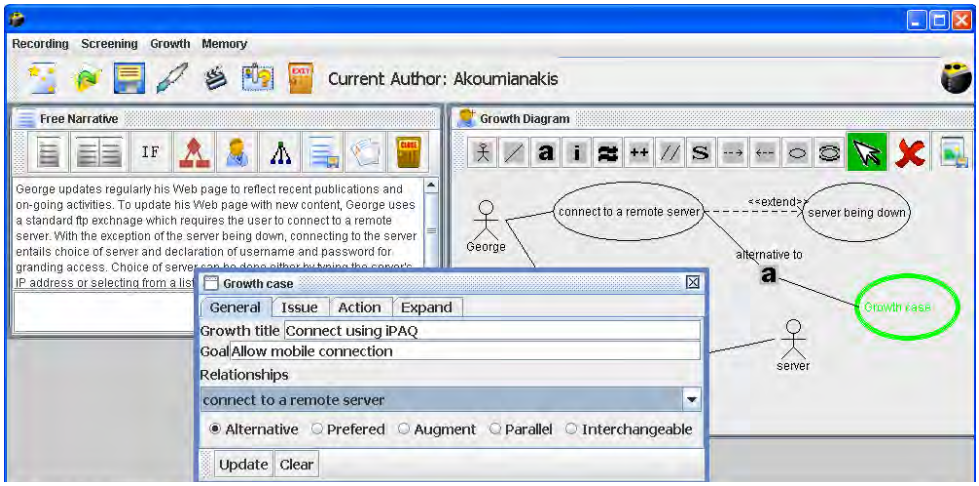
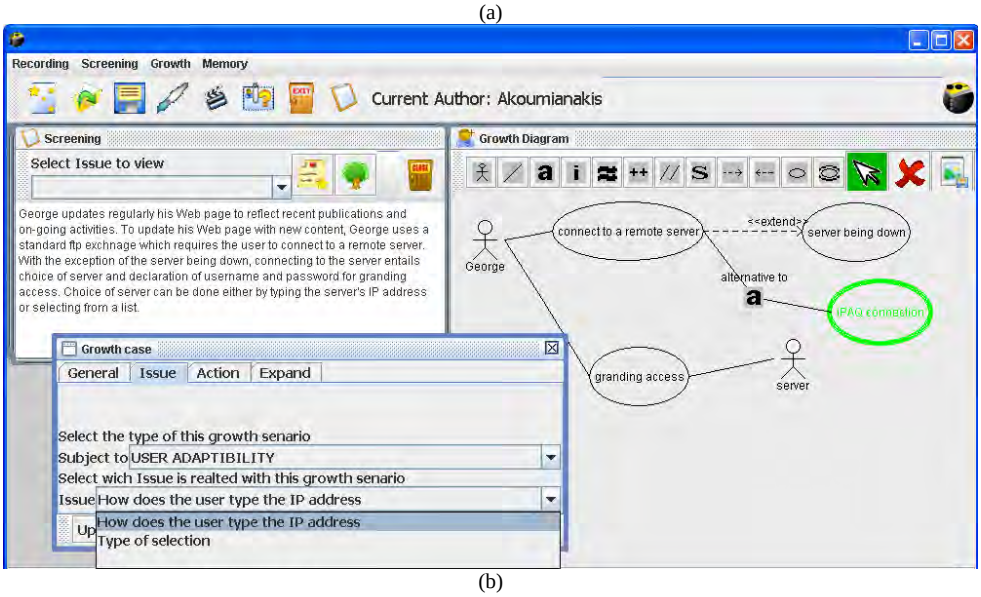


Figure 8. The screening stage



(a)



(b)

Figure 9. Populating a reference scenario. (a) Introducing a growth scenario; (b) Populating a growth scenario

growth scenario elaboration dialogue groups the properties of a growth scenario into four categories. The general properties of the growth scenario declare its name, description and relationship with other scenarios. In a similar fashion the user can assign the issues relevant to (or addressed by) the growth scenario, the pre- and post-conditions and supporting analysis (*i.e.* a state transition diagram, numbered sequence, partitioned narrative, etc). This provides a kind of verification for each growth scenario, since it ensures the minimum qualities required (*i.e.*, each scenario is assigned to a goal, each scenario is realized through a set of actions, etc). This issue is further elaborated later on in this paper.

Building the global execution context graph entails three steps: 1) devising growth scenarios; 2) assigning quality attributes to justify the derived growth diagram and 3) commenting on the pseudo verification applied to check the global execution context graph. **Table 3** provides a summary of the growth scenarios highlighting both the case of supporting ftp through PDA and the enhancement of the file selection task. These extensions are typically expressed as new/evolving requirements to be accommodated as growth cases of the initial base scenario. From the descriptions in **Table 3**, we can deduce that the global execution context of the new ftp application should include one additional growth scenario namely 'Select with scanning' and two parallel components designating that selection is augmented by two growth scenarios namely 'One button/Auto' or 'One button/Manual'. This is depicted in **Figure 10**. The relationships between the various growth scenarios define the scale and scope of the system's adaptable and/or adaptive behaviour. This offers useful insight to the range of anticipated changes and their implication on architectural abstraction, the choice of interaction techniques, as well as the conditions under which alternative styles of interaction are to be initiated.

At any time, designers can justify their decisions by rationalizing growth scenarios using non-functional quality models. Such models may be built in advance so as to establish global constraints on software design or in the course of building and rationalizing a task's global execution context. **Figure 11** presents an example decomposition of the 'accessibility' quality in terms of alternatives or claims softgoals in the vocabulary of the NFR Framework [8]. Specifically, the model in **Figure 11** details that accessibility can be satisficied either by augmenting interaction through scanning, or by expanding a toolkit library with new interaction object classes or by integrating another toolkit class library. The relationships qualify the degree of satisficing a goal. Thus, augmentation and expansion support (*i.e.*, have a positive influence) on accessibility, while toolkit integration is indifferent. **Figure 11** on the left hand side represents the link between the non-functional quality model and the global execution context graph. The rationale behind the combined model is intended to convey the following meaning: The iPAQ version of the ftp application should support a scanning interface which should allow selection in two alternative modes – one button with automatic scanning or one button with manual scanning of the highlighter.

Figure 12 depicts an interactive instance of the augmented ftp application with the scanning interface on and the multiple file selection facility (see **Table 3** and **Figure 10** for the rationale of the growth scenarios). As shown, scanning is activated through explicit function activation by pressing the button in the left bottom corner (see **Figure 12(a)**). Once activated the scanner gives focus in round-robin fashion to each control in a hierarchical fashion. It is worth noting the difference in the interactive behaviour for each object of focus. Thus, when scanning is activated and the object of focus is a text entry field the fill colour is changed (see **Figure 12(b)**) while when the object of focus is a button then the label is underlined (see **Figure 12(c)**). In **Figure 12(d)** the focus is on the Download Button (note the square around it). By pressing a hardware button on the device the scanner moves from one level to another (*i.e.* from scanning selection sets to scanning items within a selection set and vice versa). **Figure 12(e)** demonstrates the multiple file selection by checking files in a category. For example, when the "Remote Files" list has the focus, pressing the PowerPoint icon on the taskbar, all files with ppt extension in that list are selected. This multi-selection task adds checkboxes to the left of all items in the list, and files with 'ppt' extension are automatically checked. It should be noted that this type of selection is very useful as it reduces keystroke level interactions (*i.e.* avoids using the slider to locate files and multiple file checking), without changing the initial application in any other way.

4.4 Pseudo Verification of the Global Execution Context Graph

In its current version, the tool consolidates the global execution context graph in an XML document by implementing a pseudo-verification to ensure that the global execution context graph satisfies to some degree the criteria of completeness and redundancy. The tests performed aim to satisfy the following:

- Each scenario S_i in the global execution context graph GeCg(S) where S denotes the reference system should be either a base scenario or a growth scenario
- Given a system S, for each base scenario $SB_i \in$ GeCg(S) there is at least one growth scenario $SG_j \in$ GeCg(S) related with SB_i
- Given a system S, then a growth scenario $SG_i \in$ GeCg(S) can be related with a base scenario $SB_i \in$ GeCG(S) or another growth scenario $SG_j \in$ GeCg(S);
- All scenarios are assigned to goals – informally, this

Table 3. Elaboration of growth scenarios

	Base scenario (Desktop)		**Growth Scenario (PDA)**		**Growth Scenario (PDA & scanning)**	
Initiator	Professional user		Professional user		User at home	
Context of use	The user outside the office		The user enters the classroom and wishes to ftp the file containing his slideshow		The user is at home and wishes to review his slideshow	
	User	*System*	*User*	*System*	*User*	*System*
Flow of events	o The user connects to the server and logs-in o The user carries out file selection o The user issues a command (by button press)	o The system responds to notify user's login and presents the desktop embodiment of the user interface o The system notifies the user of current selection o The system executes the command o The system updates the display	o The user connects to the server and logs-in o The user makes a selection from the list of patients o The user issues a command (using the light pen)	o The system responds to notify user's login and presents the PDA embodiment of the user interface o The system list the currently selected file o The system executes the command	o The user connects to the server and logs-in o The user declares selection mode o The user selects all files with .ppt extension	o The system responds to notify user's login and presents the augmented PDA embodiment of the user interface o The system lists of files in a default style o The system initiates suitable style & informs the user o The system executes the command
Exceptions	› Network problems › Error in login procedure		› Network problems › Error in login procedure		› Network problems › Error in login procedure	
Preconditions	› User is authorized › Desktop interface is available › System has inferred the task's execution context resulting in HTML style being automatically initiated		› User is authorized › User is in possession of the designated terminal › System has inferred the task's execution context resulting in PDA style being automatically initiated		› User is authorized › User is in possession of the designated terminal User is familiar with scanning › System has inferred the task's execution context resulting in PDA style being automatically initiated	
Post-conditions	› Designated files are successfully transferred		› Designated files are successfully transferred		› Designated files are successfully transferred	
Relationships	› Alternative to base scenario		› Alternative to base scenario		› Augments PDA style › Parallel selection as separate growth scenarios	

ensures that a scenario is devised to facilitate a designated goal of the system. Thus, there are no scenarios beyond the scope of the envisioned system;

• Each scenario can be satisfied by at least one goal – informally, the proposition aims to assert that each scenario is linked to at least one goal.

The above propositions are checked before the global execution context graph is transformed into XML. This allows a pseudo verification of the completeness, redundancy and understandability of a global execution context graph. Specifically, the propositions can be considered as necessary but not sufficient conditions for ensuring completeness. Clearly, as the global execution context graph is subject to refinement, no sufficient condition for completeness can hold. As for redundancy, the propositions aim to support a weak notion of redundancy, which asserts that no scenarios are included that would not be designated to goals. Obviously, the global execution context graph could incorporate redundancy both at the level of growth scenarios (*i.e.* alternative growth scenarios may exist which satisfy the same goal) and at the level of actions (*i.e.* alternative action sets may be employed to satisfy a user goal). Finally, regarding understandability, the propositions aim to ensure that all scenarios included in the global execution context graph are understandable by tracing their designated goals, which are considered valid. On the other hand, the propositions offer no guarantee that the global execution context graph can be understood.

At any instance, the GeCg can be traversed by selecting and following a particular path from start to end and understanding the system's behaviour under certain conditions. Path differentiation is always associated with a scenario relationship of type alternative or augments. Referring to our example, we can define possible traversals of the global execution context graph, differentiated by colour. Activating, the reference scenario results in an iPAQ embodiment of the designated task, with single and multiple selection. Activation of scanning would augment the file selection process with scanning. As for the type of scanning, two alternative manifestations are available with only one being active at any point in time. In terms of system implementation requirements, it is important to note that the underlying intention is that both paths should co-exist, while through context-sensitive processing the system should decide on the choice of optimal path.

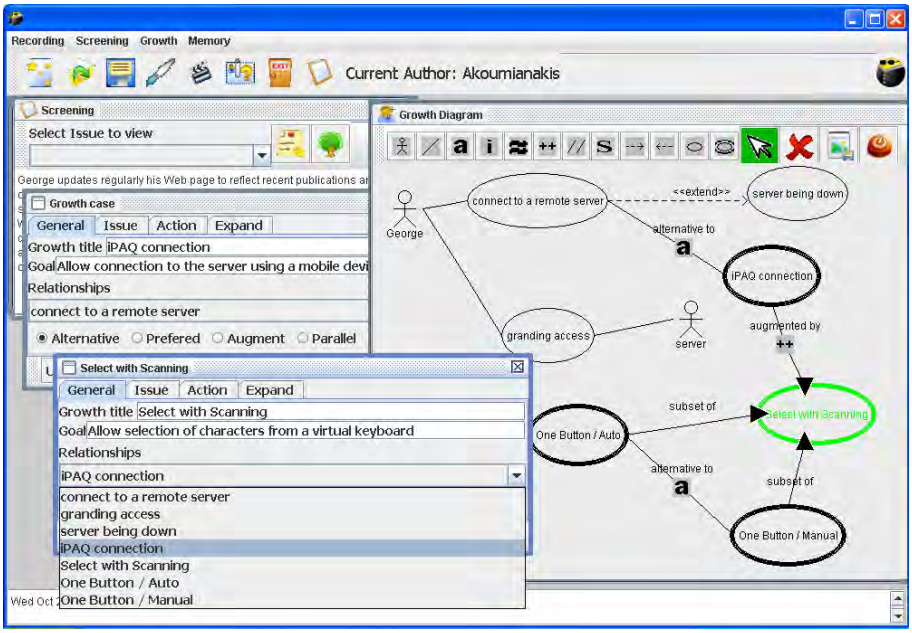
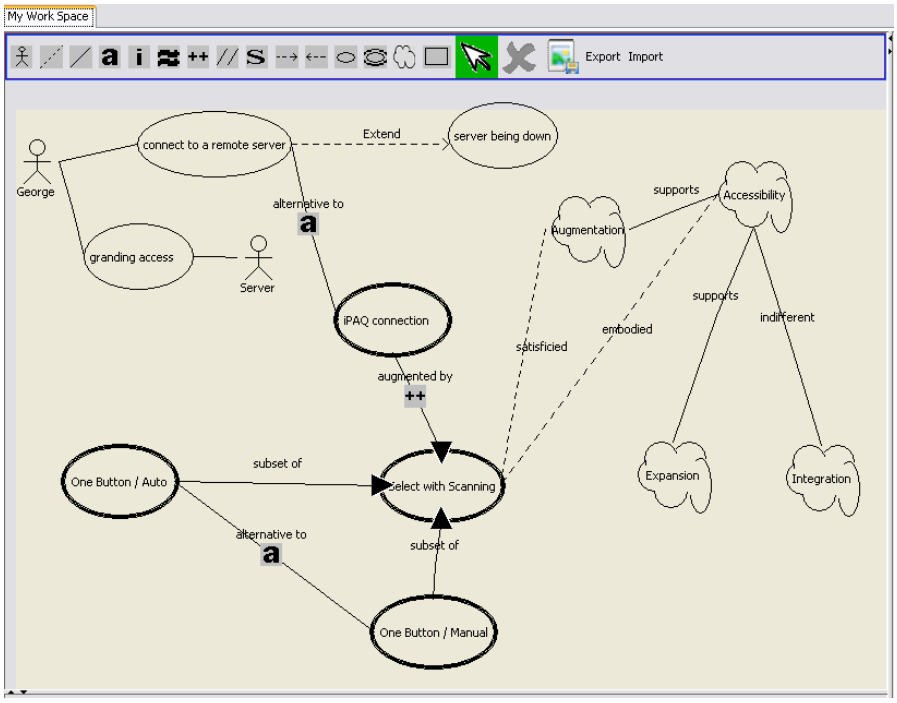


Figure 10. Designating an augmentation of a growth scenario



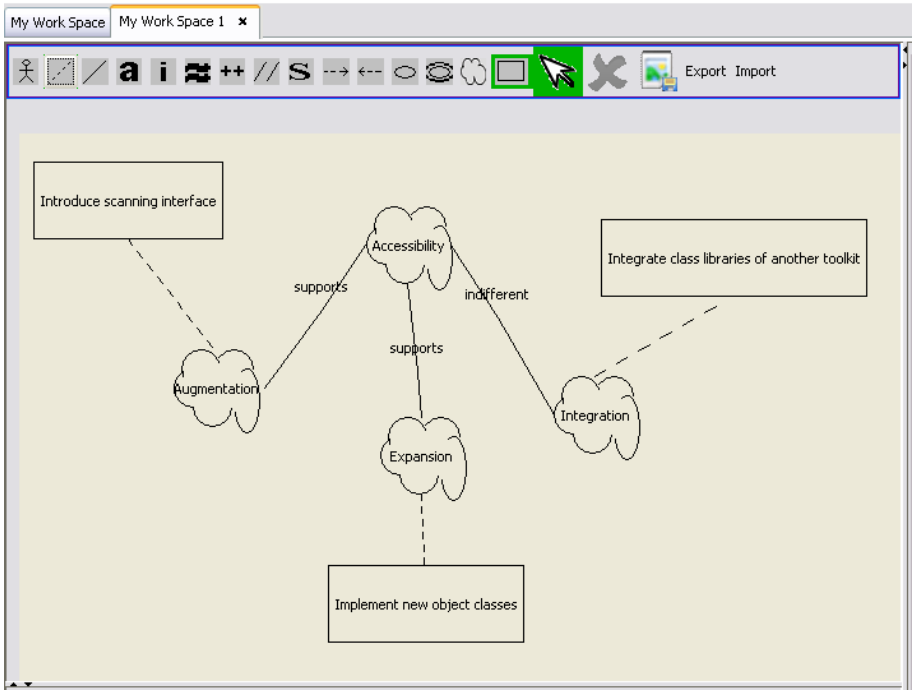
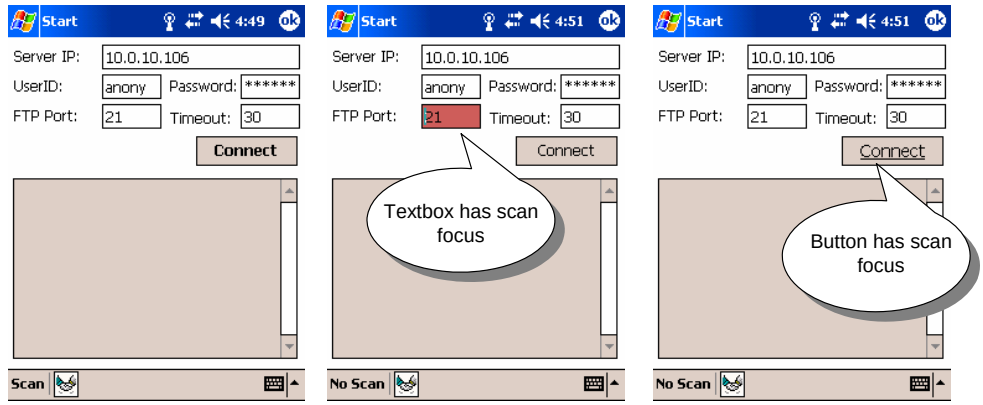
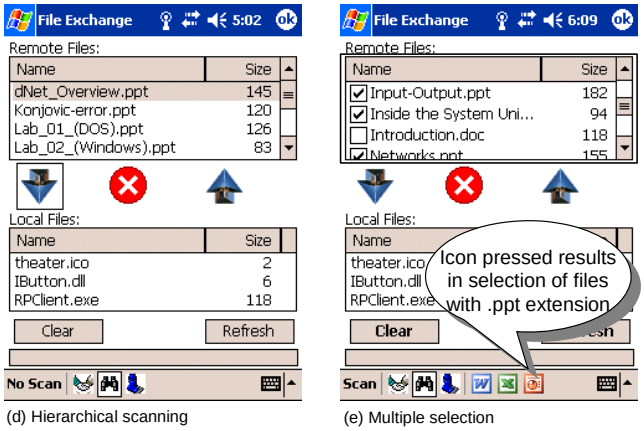


Figure 11. Quality models and the global execution context graph



(a) Initial screen (b) Scanning activated (c) Scanning activated (cont.)



(d) Hierarchical scanning (e) Multiple selection

Figure 12. Examples of the scanning interface

5. Discussion

The work presented in this paper differs from recent related efforts both in terms of orientation and underlying perspective. In terms of orientation, our interest is to frame the problem of software execution across different non-functional contexts as an issue of software evolution. To this end, change in functional requirements is of course valid but it can only explain in part why modern information systems need to change. In fact, there is evidence to suggest that most of the changes in modern information systems do not concern functional components but their connections and interactions. This explains recent efforts aiming to frame change in the context of non-functional requirements and architectural quality attributes.

In terms of underlying perspective, the present work pursues a line of research, which is motivated by the fact that complexity of software is increasingly shifting from construction to evolution. In the past, the software design community addressed complexity in construction by devising abstractions (*i.e.*, components, visual notations, models and tools), which make construction-oriented artefacts first-class objects. In a similar vein, an approach to addressing complexity in software evolution could be focused on making the software evolution artefacts explicit through modelling them as first class objects. This is especially relevant for service-oriented architectures (SOA), aiming to appropriate the benefits of reusability and maintainability to foster the design of applications in an implementation independent manner using network services and connections between network services.

The NfRn provides insights towards this end by promoting a shift in the unit of analysis from task- or activity-level to task execution contexts. Then, designing software systems for execution across different non-functional contexts is conceived as specifying the system's global execution context. This requires an explicit account of platform-oriented non-functional requirements such as augmentation, expansion, integration and abstraction, which are considered as quality goals inscribed in a SOA. Moreover as software designers will increasingly be required to articulate the global execution context of a system's tasks, there is a compelling need for tools supporting the management of designated software evolution artefacts. In our work, this is facilitated by extending the use case notation widely employed for documenting functional requirements in a manner facilitating the construction and refinement of the tasks' global execution context graph.

The global execution context notation and the supporting tool have now been applied in a number of case studies and applications (see [44-46]) in addition to the initial validation in the Health Telematics domain [47], providing useful insight to managing change in interactive software. These experiences provide evidence to support the claim that the basic vocabulary of the GeC and the method presented in this paper offer useful insight to modelling software design evolution necessitated either by new requirements or evolving requirements. The primary benefit of the method results from the fact that change becomes a first class design object modelled through designated growth scenarios that evolve from previously codified reference scenarios. Moreover, the GeCg as an artefact provides designers with useful information regarding:

- The range of alternative execution contexts considered appropriate at a point in time.
- The conditions which characterize activation/deactivation of growth scenarios; this entails an elaboration and justification of each of the relationships appearing in the graph.
- Guidance in the choice of what paths to traverse or walk through under specific conditions.
- Choice of suitable system architecture; for example relationships of the type alternative and augments designate the systems adaptable components, while the relationship type parallel points out adaptive features of the target implementation.

Consequently, the main contributions of the presented work are threefold. Firstly, we described a method for modelling change early in the development lifecycle. This is done by introducing a notation, which is simple and intuitive while resembling the vocabulary used by other popular notations such as UML. It is argued that using this notation to specify the current and anticipated contexts of use constitutes an improvement upon current practices. Specifically, the burden of using textual descriptions to codify goals (as in the case of RUP) is removed. Instead, visual constructs are used to codify design logic and rationale in a manner similar to other research proposals for visual goal-oriented requirements modelling [3-5]. Secondly, the method offers a frame of reference for considering scenarios as drivers for system evolution. This departs from contemporary views of scenario-based requirements engineering where scenarios are considered as static resources appearing at the beginning of a project and lasting until specifications or requirements are documented. In our work, scenarios remain 'live' and persistent resources driving future system evolution. Moreover, this is achieved in a systematic manner and it is documented using appropriate computer-based tools. Another contribution of the present work is that it is particularly suited to dealing with non functional requirements – such as adaptability, adaptivity, scalability and portability – which in contrast to functional requirements, are known to be hard to model and account for. This offers a perspective on scenario evolution, which is complementary to existing conceptions proposed in the relevant literature (e.g. [17]).

6. Summary and Future Work

In this paper, we have presented a method and a supporting tool for specifying the global execution context of computer-mediated tasks. Our motivation has been to make explicit the artefacts of evolution. Thus, our method considers evolution as a transformation from the current situation (codified through reference scenarios) to an envisioned situation (represented by semantically related growth scenarios). The links characterizing such transformations are a small set of scenario relationships such as alternate execution, concurrency, ordering, and set-oriented relationships between two scenarios, devised to encapsulate evolution as change of functional requirements as well as evolution as change in non-functional qualities. A system's global execution context can then be depicted as a visual construction, referred to as the global execution context graph, and can be populated by a supporting tool suite and transformed to XML.

Future work seeks to address several extensions both in the method and the i-GeC tool. In terms of methodological extensions, we are studying the development of a scenario specification language to formalize the description of scenarios. On the other hand several refinements of the tool suite are currently under development. Specifically, an on going activity seeks to expand the (currently primitive) user interface prototyping features supported by the tool so as to establish a link between scenarios (either reference or growth), their underlying rationale and their (possible) interactive embodiments. In this context, we are also exploring the possibility of linking the tool's outcome with existing task-based notations and model-based user interface engineering methods such as Teresa [32].

REFERENCES

[1] D. R. Olsen, "Interacting in Chaos," *Interactions*, Vol. 6, No. 5, September-October 1999, pp. 42-54.

[2] S. S. Anand, P. Kearney and M. Shapcott, "Generating Semantically Enriched User Profiles for Web Personalization," *ACM Transactions on Internet Technology*, Vol. 7, No. 4, October 2007.

[3] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," *Proceedings of the* 3*rd IEEE International Symposium on Requirements Engineering* (*RE*'97), IEEE Computer Society, New York, 1997, pp. 226-235.

[4] H. Solheim, F. Lillehagen, S. A. Petersen, H. Jorgensen and M. Anastasiou, "Model-Driven Visual Requirements Engineering," *The Proceedings of the* 13*th IEEE International Conference on Requirements Engineering*, IEEE Computer Society, New York, 2005, pp. 421-425.

[5] K. Cooper, S. P. Abraham, R. S. Unnithan, L. Chung and S. Courtney, "Integrating Visual Goal Models into the Rational Unified Process," *Journal of Visual Languages and Computing*, Vol. 17, 2006, pp. 551-583.

[6] M. E. Dashofy, A. Van der Hoek and N. R. Taylor, "A Comprehensive Approach for the Development of Modular Software Architecture Description Languages," *ACM Transactions on Software Engineering and Methodology*, Vol. 14, No. 2, 2005, pp. 199-245.

[7] J. Mylopoulos, L. Chung and B. Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach," *ACM Transactions on Software Engineering*, Vol. 18, No. 6, 1992, pp. 483-497.

[8] L. Chung, B. Nixon, E. Yu and J. Mylopoulos, "Non-Functional Requirements in Software Engineering," Kluwer Academic Publishers, Boston, 1999.

[9] P. Finger, "Component-Based Frameworks for E-Commerce," *Communications of the ACM*, Vol. 43, No. 10, 2000, pp. 61-66.

[10] M. M. Lehman and J. F. Ramil, "Software Evolution and Software Evolution Processes," *Annals of Software Engineering*, Vol. 14, No. 1-4, 2002, pp. 275-309.

[11] D. Thomas, "Agile Programming: Design to Accommodate Change," *IEEE Software*, Vol. 22, No.3, 2005, pp. 14-16.

[12] K. Beck, "Embracing Change with Extreme Programming," *IEEE Computer*, Vol. 32, No. 10, 1999, pp. 70-77.

[13] E. M. Shina and H. Gomaab, "Software Requirements and Architecture Modeling for Evolving Non-Secure Applications into Secure Applications," *Science of Computer Programming*, Vol. 66, No. 1, 2007, pp. 60-70.

[14] L. Naslavsky, A. T. Alspaugh, J. D. Richardson and H. Ziv, "Using Scenarios to Support Traceability," *Proceedings of the* 3*rd International Workshop on Traceability in Emerging Forms of Software Engineering*, ACM Press, New York, 2005, pp. 25-30.

[15] J. Cleland-Huang and K. C. Chang, "Event-Based Traceability for Managing Evolutionary Change," *IEEE Transactions on Software Engineering*, Vol. 29, No. 9, 2003, pp. 796-810.

[16] V. Rajlich, "Modeling Software Evolution by Evolving Interoperation Graphs," *Annals of Software Engineering*, Vol. 9, No. 1-4, May 2000, pp. 235-248.

[17] K. K. Breitman, J. C. S. P. Leite and M. D. Berry, "Supporting Scenario Evolution," *Requirements Engineering*, Vol. 10, No. 2, May 2005, pp. 112-131.

[18] D. B. Petriu, D. Amyot, M. Woodside and B. Jiang, "Traceability and Evaluation in Scenario Analysis by Use Case Maps," In: S. Leue and T. J. Systa, Ed., *Scenarios*: *Models, Transformations and Tools* (*Lecture Notes in Computer Science*), Springer-Verlag, Berlin/Heidelberg, Vol. 3466, 2005, pp. 134-151.

[19] J. Hammer and M. Schneider, "The GenAlg Project: Developing a New Integrating Data Model, Language, and Tool for Managing and Querying Genomic Information," *SIGMOD Record*, Vol. 33, No. 2, 2004, pp. 45-50.

[20] I. Jacobson, M. Christeron, P. Jonsson and G. Overgaard, "Object-Oriented Software Engineering – A Use Case Driven Approach," Addison-Wesley, White Plains, 1992.

[21] G. Avellis, "CASE Support for Software Evolution: A

Dependency Approach to Control the Change Process," *Proceedings of the* 5*th International Workshop on Computer-Aided Software Engineering*, IEEE Computer Society, New York, 1992, pp. 62-73.

[22] T. Mens and T. D'Hondt, "Automating Support for Software Evolution in UML," *Automated Software Engineering*, Vol. 7, No. 1, 2000, pp. 39-59.

[23] T. Kosar, E. P. M. Lopez, A. P. Barrientos and M. Mernik, "A Preliminary Study on Various Implementation Approaches of Domain-Specific Language," *Information and Software Technology*, Vol. 50, No. 5, April 2008, pp. 390-405.

[24] J. Greenfield and K. Short, "Software Factories – Assembling Applications with Patterns, Frameworks, Models & Tools," John Wiley & Sons, New York, 2004.

[25] B. Myers, "User Interfaces Software Tools," *ACM Transactions on Human-Computer Interaction*, Vol. 12, No. 1, 1995, pp. 64-103.

[26] J. Heer, S. Card and J. Landay, "Prefuse: A Toolkit for Interactive Information Visualization," *Proceedings of ACM CHI*, ACM Press, New York, 2005, pp. 421-430.

[27] B. B. Bederson, J. Grosjean and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Transactions on Software Engineering*, Vol. 30, No. 8, 2004, pp. 535-546.

[28] E. Adar, "GUESS: A Language and Interface for Graph Exploration," *Proceedings of the ACM Conference on Human Factors in Computing Systems*, ACM Press, New York, 2006, pp. 791-800.

[29] M. Barbacci, R. Ellison, A. Lattanze, J. Stafford, C. Weinstock and W. Wood, "Quality Attribute Workshops," 2nd Edition, Carnegie Mellon Software Engineering Institute, Pittsburgh, 2002. http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr019.pdf

[30] L. Bass, P. Clements and R. Kasman, "Software Architecture in Practice," Addison-Wesley, White Plains, 1998.

[31] L. Chung and N. Subramanian, "Adaptable Architecture Generation for Embedded Systems," *The Journal of Systems and Software*, Vol. 71, No. 3, 2004, pp. 271-295.

[32] G. Mori, F. Paternò and C. Santoro, "Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions," *IEEE Transactions on Software Engineering*, Vol. 30, No. 8, 2004, pp. 507-520.

[33] M. Salehie and L. Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges," *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 4, No. 2, 2009.

[34] M. Barbacci, M. Klein, T. Longstaff, C. Weinstocket, "Quality Attributes," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1995. http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html

[35] E. Folmer and J. Bosch, "Architecting for Usability: A Survey," *Journal of Systems and Software*, Vol. 70, No. 1-2, 2004, pp. 61-78.

[36] A. MacLean, V. Bellotti and S. Shum, "Developing the Design Space with Design Space Analysis," In: P. F. Byerley, P. J. Barnard and J. May, Ed., *Computers, Communication and Usability*: *Design Issues, Research and Methods for Integrated Services*, Elsevier, Amsterdam, pp. 197-219, 1993.

[37] J. Lee and K.-Y. Lai, "What's in Design Rationale?" In: T. P. Moran and J. M. Caroll, Ed., *Design Rationale*: *Concepts, Techniques and Use*, Lawrence Erlbaum Associates, Malham, 1996.

[38] J. S. Olson and T. P. Moran, "Mapping the Method Muddle: Guidance in Using Methods for User Interface Design," In: M. Rudisill, C. Lewis, P. B. Polson and T. D. McKay, Ed., *Human-Computer Interface Design*: *Success Stories*, *Emerging Methods*, *and Real-World Context*, Morgan Kaufmann Publishers, San Francisco, 1996, pp. 101-121.

[39] D. Akoumianakis, A. Savidis and C. Stephanidis, "Encapsulating Intelligent Interactive Behavior in Unified User Interface Artifacts," *Interacting with Computers*, Vol. 12, No. 4, 2000, pp. 383-408.

[40] J. C. S. P. Leite, J. H. Doorn, G. D. S. Hadad and G. N. Kaplan, "Scenario Inspections," *Requirements Engineering*, Vol. 10, 2005, pp. 1-21.

[41] K. Weidenhaupt, K. Pohl, M. Jarke and P. Haumer, "Scenarios in System Development: Current Practice," *IEEE Software*, Vol. 15, No. 2, 1998, pp. 34-45.

[42] C. Potts, K. Takahashi and A. Anton, "Inquiry–Based Requirements Analysis," *IEEE Software*, Vol. 11, No. 2, 1994, pp. 21-32.

[43] C. Rolland and C. B. Achour, "Guiding the Construction of Textual Use Case Specifications," *Data & Knowledge Engineering*, Vol. 25, No. 1-2, 1998, pp. 125-160.

[44] D. Akoumianakis and I. Pachoulakis, "Scenario networks: Specifying User Interfaces with Extended Use Cases," In: P. Bozanis and E. N. Houstis, Ed., *Advances in Informatics* (*Lecture Notes in Computer Science*), Springer-Verlag, Berlin/Heidelberg, Vol. 3746, 2006, pp. 491-501.

[45] D. Akoumianakis, A. Katsis and N. Bidakis, "Non-functional User Interface Requirements Notation (NfRn) for Modeling the Global Execution Context of Tasks," In: K. Coninx, K. Luyten and K. A. Schneider, Ed., *Task Models and Diagrams for Users Interface Design* (*Lecture Notes in Computer Science*), Springer-Verlag, Berlin/Heidelberg, Vol. 4385, 2006, pp. 259-274.

[46] D. Akoumianakis, G. Vellis, D. Kotsalis, G. Milolidakis and N. Vidakis, "Experience-Based Social and collaborative Performance in an 'Electronic Village' of Local Interest: The eKoNEΣ Framework," In: J. Cardoso, J. Cordeiro and J. Filipe, Ed., *ICEIS*'2007 – 9*th International Conference on Enterprise Information Systems, Volume HCI*, INSTICC, Funchal, 2007, pp. 117-122,

[47] D. Akoumianakis and C. Stephanidis, "Blending Scenarios and Informal Argumentation to Facilitate Universal Access: Experience with the Universal Access Assessment Workshop Method," *Behaviour & Information Technology*, Vol. 22, No. 4, 2003, pp. 227-244.

J. Software Engineering & Applications, 2010, 3: 331-340
doi:10.4236/jsea.2010.34038 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Test Effort Estimation Using Neural Network

Chintala Abhishek*, Veginati Pavan Kumar, Harish Vitta, Praveen Ranjan Srivastava

Department of Computer Science and Information System, Birla Institute of Technology and Science, Pilani, India.
Email: {*chabhishek123, pavanon9, harishvitta, praveenrsrivastava}@gmail.com

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

ABSTRACT

In software industry the major problem encountered during project scheduling is in deciding what proportion of the resources has allocated to the testing phase. In general it has been observed that about 40%-50% *of the resources need to be allocated to the testing phase. However it is very difficult to predict the exact amount of effort required to be allocated to testing phase. As a result the project planning goes haywire. The project which has not been tested sufficiently can cause huge losses to the organization. This research paper focuses on finding a method which gives a measure of the effort to be spent on the testing phase. This paper provides effort estimates during pre-coding and post-coding phases using neural network to predict more accurately.*

Keywords*: *Test Effort Estimation, Neural Network, Use Case Points, Halstead Model

1. Introduction

Software engineering [1] is a field that provides standardized approaches for the development, operation, and maintenance of software. Software Engineering as a discipline the need arose when there was software crisis [1]. The need for producing software of high quality and to have a control on the effort both in terms of the money and the person-hours gave software Engineering a higher prominence. It defines a process which helps for project management [1].

A crucial aspect of software Engineering is software testing [1]. Software testing is a phase of software development which deals with testing the developed product or project. A project /product which have been developed without sufficient testing might contain major bugs which can render the entire project useless and also cause losses of critical data.

Software testing [1] by definition is the process of validating and verifying a software product or a project or an application. It should be tested on the aspects of: meeting the requirements of the user, functionality, and characteristics of the developed software.

Generally software test life cycle involves several stages and it can be classified into three major phases.

Initial phase: This phase involves with identifying which aspects of the designed are to be tested followed by the creation of a test phase strategy.

Intermediate phase: This phase involves the development step in which the procedures and the scenarios all are defined. This is followed by the execution step which deals with implementing the developed plan and reporting any error found.

Termination phase: This is longest phase involves several activities, once the testing is finished a report indicating the fitness of the project/product to be released is created. Then the analysis is carried out with the client to deal with the problems faced during its real time implementation. Then the detection of any further existing defects is carried out. If there are any modifications done then the entire component is retested to determine any side effects that could have occurred because of changes in previous step (Regression testing). If the system meets the exit criteria the testing phase is terminated.

In the ideal scenario it is desirable to have exhaustive testing as this ensures that there are no bugs or errors. This is not possible even with a project of very less complexity. Thus the need for having an efficient testing strategy arises. Software testing phase needs to be planned to be carried out efficiently.

Artificial neural network [2-4] is a soft computing technique that tries to achieve the functionality of biological neural network. It consists of group of artificial neurons that work on mathematical model to process the information and to solve highly complex problems. It involves a network of simple processing elements called as neurons that are connected. The connections between the neurons help in realizing a complex functionality. As mentioned above Software

testing is a challenging field and this paper proposes and efficient methodology to estimate test effort estimation with more accuracy using artificial neural network.

The paper is written with the general introduction of the Software testing in Introduction, followed by the description about the background work (Section 2). Section 3 deals with actual problem while section four is fully devoted on proposed approach of the paper. Section 4 deals with the application of the proposed model and finally in Section 6, the results obtained are discussed.

2. Background Work

Estimation accuracy can be achieved by choosing an accurate model for measuring. This section provides with the information that has been gathered, on which the work is based upon.

2.1 Use Case Point [5]

The effort to be estimated for the pre-coding phase is based on the use case point analysis [5]. Nageswaran [5] proposes a strategy which calculates effort based on the unadjusted use case weight (UUCW), unadjusted actor weight (UAW) and the technical and environmental factors (TEF). Those factors are calculated based on the classification of actors and usecases into simple, average, complex and very complex classes. The obtained unadjusted use case point (AUCP) is multiplied by a factor to obtain the effort. The effort obtained in this is not accurate with the expected level of accuracy in estimation. Our proposed model is totally inspired by Nageswarn work. This paper provides an improvement over the method proposed by Nageswaran [5]. Nageswaran [5] model can be stated as:

During this phase the project manager has the design document based on which he can make an estimate of the effort that needs to be allocated to the testing phase.

The proposed method suggest the usage of adjusted unadjusted usecase weight, unadjusted actor weight (UCW), technical and environmental factor (TEF) as a measure for the test effort estimation. Back propagation in neural network is used for training the network.

The inputs are taken for a particular project based on the design document. The UUCW is calculated as

Usecase component:

UUCW = (No. of usecases of type simple*1 + No. of usecases type average*2 + No. of usecases of type complex*3 + No. of usecases of type very complex*4)

The usecase information **Table 1** is used for distinguishing and assigning the values.

Actor components:

The actor information is obtained from the **Table 2**.

TEF components:

The technical and environmental factors are assigned as indicated by the **Table 3**.

Table 1. Usecase weight assignment table [5]

Usecase type	Description	Weight
Simple	<=3	1
Average	4-7	2
Complex	>7	3

Table 2. Actor weight assignment table [5]

Actor type	Description	Weight
Simple	GUI	1
Average	Interactive	2
Complex	Low interaction	3

Table 3. TEF weight assignment factors [5]

Factor	Description	Assigned value
F1	Test tools	5
F2	Documented inputs	5
F3	Development environment	2
F4	Test environment	3
F5	Test ware reuse	3
F6	Distributed system	4
F7	Performance objectives	2
F8	Security	4
F9	Complex interface	5

UAW and TEF are calculated as:

UAW = $\sum$ Actor weight*number of actors

TEF = $\sum$ Assigned Weight*assigned value

2.2 Halstead Model [6]

The background study involved studying Halstead model [6]. A brief explanation of it is given here. It makes use of some primitive measures to determine the length and the volume of the program [6]. It makes use of the factors such as total number of operators (n_1), total number of operands (n_2), total number of their operator occurrences (N_1) and the total number of operand occurrences (N_2). He also proposes a formula for measuring the development effort and development time using such measures.

The length N is estimated according to Halstead as:

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

The program volume is given by the formula

$$V = N*\log_2(n_1+n_2)$$

A volume ratio is defined by him, represented by L, its value should not be more than one. It is represented by the formula

$$L = 2/n_1*n_2/N_2$$

The effort is given by the formula

Effort = $((n_1 * N_2)/ (\text{float } (2 * n_2)) * N *\log(n, 2)$

This is the effort as estimated by the Halstead model

and is obtained in elementary mental discriminations.

2.3 Cognitive Complexity [7]

Kushwaha [7] suggests that effort can estimated based on the total weighted information count of line of code and software and basic control structures. This method involves a complex estimation function. The effectiveness of which has not been established for large projects.

2.4 Effort Estimation Using Soft Computing Techniques [8]

Sandhu [8] shows that soft computing technique–neuronfuzzy can be applied for effort estimation by establishing its accuracy by comparing it with various other models. The estimation was done on NASA project data.

Neurofuzzy was able to estimate the nonlinear function with more accuracy. This paper helps in suggesting that effort estimation based on soft computing is indeed a right direction of accurate estimation.

Introduction to neural network:

2.5 Neural Network [2-4]

The neural network structure is used for solving complex problems [2-4]. The Backpropagation methodology is used for training the neural network. A set of input training data and the expected output is created and the network is trained with the training set. The network is trained over multiple iterations. Over the multiple iterations the network tries to converge towards the expected output and thus training itself with the required training function. The trained network is provided with the inputs from a test set and it gives the output which is the estimated output.

2.6 Neural Network Structure [9]

The neural network structure is realized using the freeware Neuroph neural network framework [9]. EasyNeurons [9] is the GUI application for it. It is a java library. The multilayer perceptron model is used for creating a neural network, as this would be the appropriate network structure which would help in realizing the problem. In a multilayer network there will be one input layer, atleast one hidden layer and one output layer. Backpropagation is used as the training methodology *i.e.*, the learning rule. It is a supervised learning algorithm. It is a learning methodology through which the network trains itself through multiple iterations over the test data. It does so by reducing an error function. The network eventually converges towards accurate values as it is trained with more and more training data.

The activation function used here is the Tanh function. The activation function is an abstraction of the action potential. It represents whether the cell should fire or not. The Tanh function is normalized .It is real valued differential curve, as represented in the **Figure 1**.

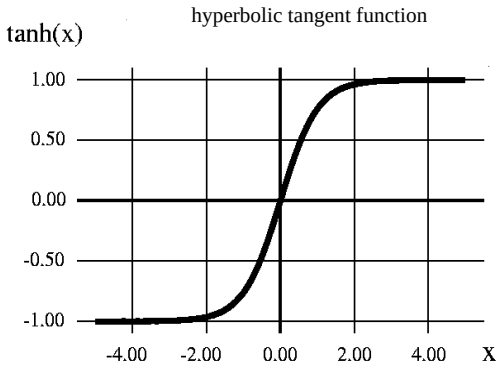


Figure 1. Tanh function

A brief description of the conventional pre and post coding effort estimation models is given here:

2.7 Conventional Methods for Pre Coding Effort Estimation [9]

1) The testing phase effort is not generally calculated. Once the product is designed the rest of the resources in terms of the budget and time are allocated to the testing phase. This methodology can be applied for mission critical system testing, as any compromise in the quality of the product would lead to huge losses [1].

2) Another method which is used for planning the testing phase effort is the percentage of the total development effort to be spent on testing. This also doesn't provide with efficient planning of resources.

2.8 Conventional Methods for Post Coding Effort Estimation [9]

1) Based on Software size:

The software size is available from the code and a productivity figure is applied to it. It involves the multiplication of number of function points and effort per function point. This approach is too simplistic, it involves estimations based on other project data which can lead to errors and it includes rigorous data maintenance [1].

2) Delphi Technique:

This technique involves a group of experts answering a questionnaire and arriving at a converging solution to the problem. The technique is time and resource consuming and generally doesn't lead to accurate predictions [1].

3) Test case enumeration based estimation:

It involves the enumeration of the entire test cases and the effort for each test case is estimated and beta distribution is applied over it. It is time consuming process.

3. Actual Problem

The test effort estimation is a big challenge in project

planning. There are no models presently available that can estimate the test effort accurately. The effort that needs to be spent on the testing phase needs to calculate precisely. The effort needs to be estimated both before the coding phase and after the coding phase. A comparison of the observed efforts should not be large, which is an indication of effective model. The problem is to propose a model which estimates the effort accurately. The proposed model should not be dependent on the type of project.

4. Proposed Approach

4.1 Architecture

The architecture involves two components: pre and post effort estimation components and learning rule used here is Back propagation algorithm as shown in the **Figure 2**.

The pre coding effort estimation consists of the three inputs components which get inputs from the design document. The three components are:

Actor components: This takes information about the actors involved in the system.

Usecase component: This takes information about the usecase involved in the design document.

TEF component: This takes the information regarding the technical and environmental factors involved in the system.

Further description about these components is given ahead in the paper.

The post coding effort estimation takes input from the code document and it has three components. They are:

Variables component: This takes the information regarding the variables involved in the system.

Complexity component: This takes the information regarding the complexity of the system.

Criticalness component: This takes information regarding the criticalness of the system.

These are further discussed ahead in the research paper. The activation function used is tanh. 'I' represent the inputs given to the system. 'X' represents the values after the application of activation function and 'w' represents the weights assigned.

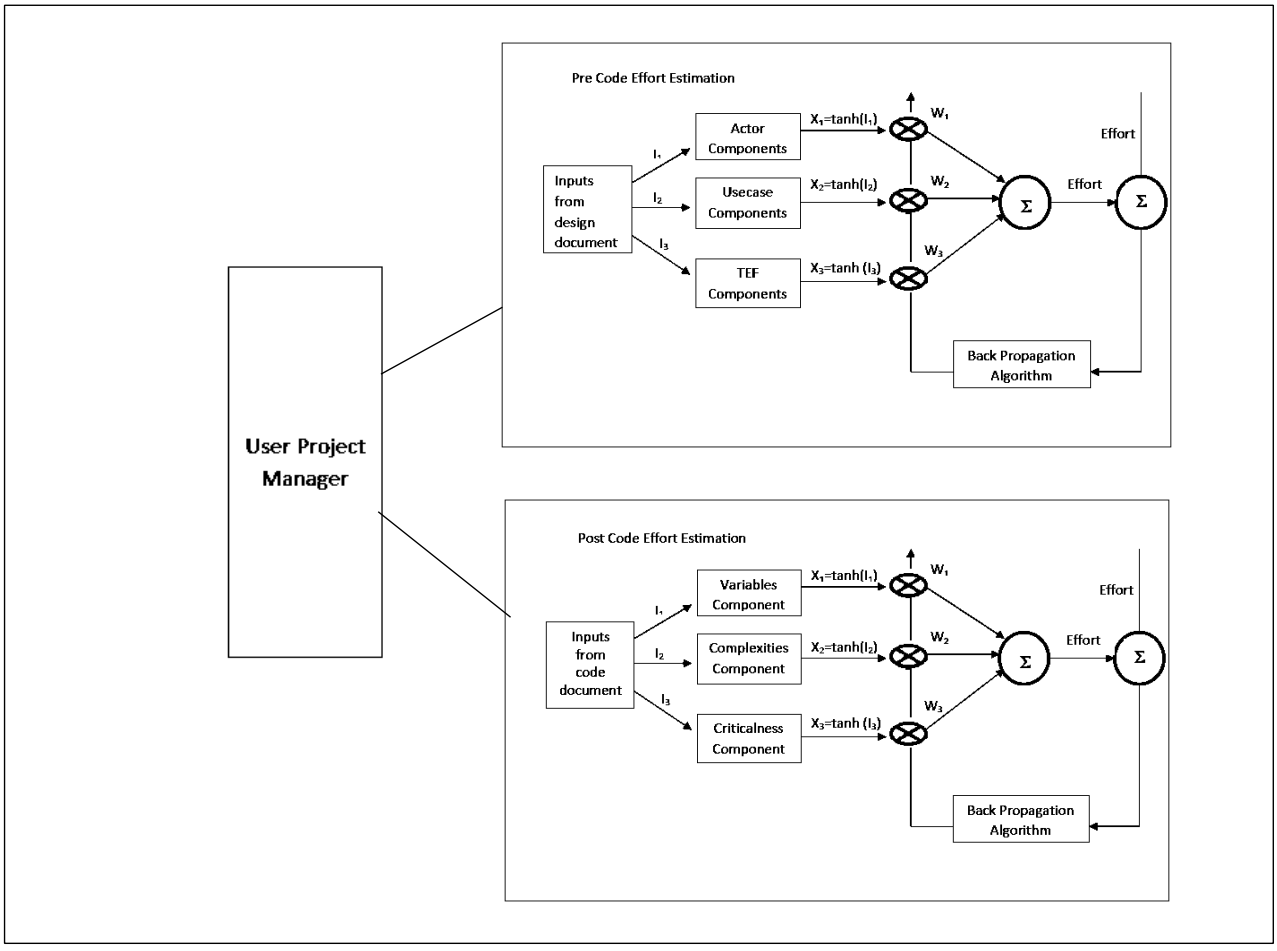


Figure 2. Architecture of the proposed system

4.2 Pre Coding Phase Effort Estimation

The proposed Pre coding effort estimation is based on the model proposed by Nageswaran [5].

Upon which this paper proposes a new improvement *i.e.*

The obtained values of UAW, UUCW, TEF and estimated test effort are trained to the network. The network trains itself to predict the values of weights and threshold values for the activation levels. The network is trained through test data over multiple iterations.

Then the network is provided with the information for the project for which an estimate needs to obtain. The information is derived from the design document. The network provides with the effort in terms of the person-months.

4.3 Neural Network Structure for Pre Coding Effort Estimation

1) Designing the network:

The network structure chosen for this phase involves three layers. One input layer through which the UAW, UUCW, TEF are given as the inputs to the network. The hidden layer consists of three nodes which are used for realizing the effort estimation function. The output layer consists of one node. The output of which gives the effort for the phase.

2) Training the network:

The network is trained with the test data that has been obtained from various sources. The test data is taken from Estimator Pal [9] and Use case Point [5] both of which contain the test data taken from a real time project, also we are using some of the real data for training purpose. This data would be helpful in training the neural network. UUCW, UAW, TEF are calculated for various projects and their test effort is provided as the training data. The network is trained for the data with maximum error rate of 0.2. The network gets trained with the provided test data over few thousands of iterations.

3) Testing the network:

The use case, actor, technical and environmental factors for the project whose test effort needs to be evaluated is taken as the input and is provided to the network which in turn provides the users with effort in person-months.

The **Figure 3** shows the neural network structure for the pre coding phase effort estimation model. It is developed in the easyneurons environment. It shows the thresholds, activation values for input, hidden, output nodes for the structure.

4.4 Post Coding Phase Effort Estimation

During this phase the project manager uses the coding document to make an estimation of the test effort.

The proposed method is based on the fact that the test effort is based on the number of inputs, number of outputs, and the complexity of the code and the criticalness of the code.

Different weightage factors are given a value each.

Variables component: As the number of inputs increases the number of test cases also increases. Different measures are given for different types of inputs. It can be observed from the **Table 4**. The method proposed makes use of the fact that a character data type doesn't need more than single test data, while an integer data would require more test cases and array variable would require even more test cases for testing [1]. Thus the assigned weights increase proportionately. var[i] takes the values of number of occurrences of each variable in the order mentioned in the **Table 4**. Var_comp[i] is the assigned weights which are taken from the **Table 4**. Thus the variable var_val is the summation of product of the number of occurrences of variables and their assigned weights.

Complexity component:

The complexity of the code is a measure of the number of test cases required for testing. Thus **Table 5** giving a measure for the complexity of the code is used. The assigned weight increases proportionately as the complexity of the code increases.

Criticalness component:

The number of test cases increases proportionately with increase in the criticalness of the system, the measure can be obtained from **Table 6**. The criticalness of the code is an indication of the importance of the code. If it is a general purpose code it is assigned a very less value (most of the project classifies under it). However if it is an essential mission critical code then the test effort increases proportionately as the number of test cases increases rapidly and thus the criticalness factor is assigned a very high value. As illustrated in the **Table 6** below.

A variable σ has been defined as an intermediate variable in measuring the effort. It is the product of var_val value, complexity value and the criticalness value.

Table 4. Complexity assignment table for variables

Input type	Assigned weight
Integer	3
Array variable	4
Character	1

Table 5. Complexity weight assignment for code

Complexity of the code	Assigned weight
O(n)	1
O(log n)	2
O(nlog n)	3
$O(n^2)$	4
$O(n^3)$	5
$O(n^4)$	6

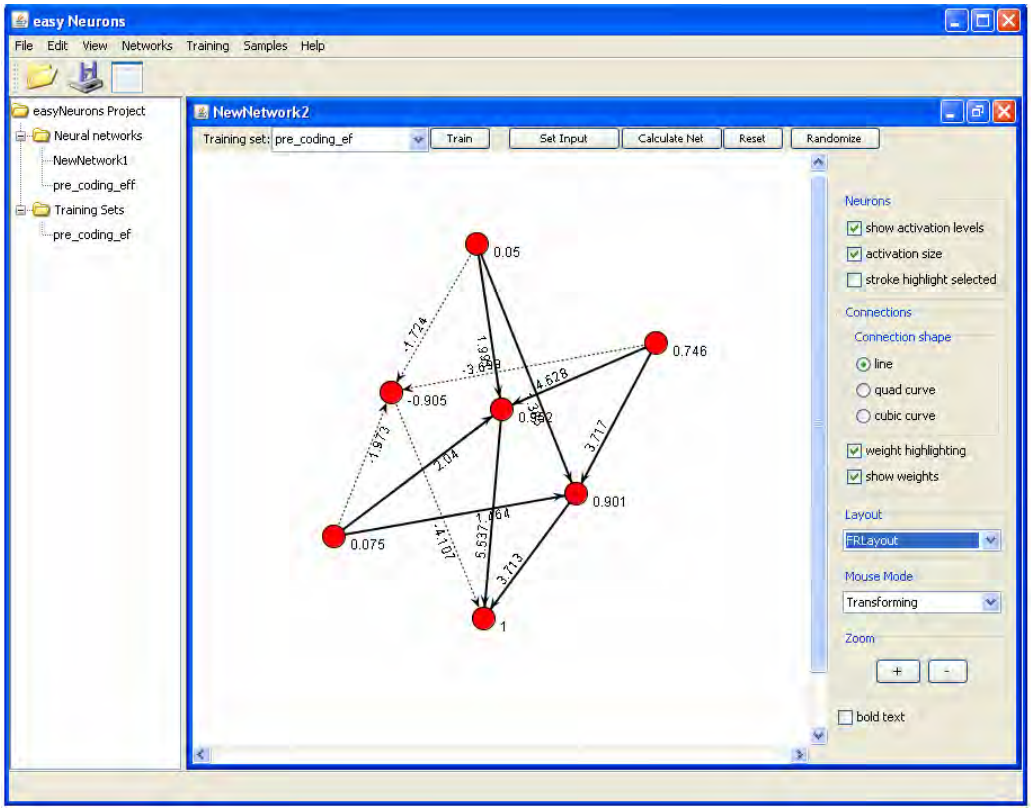


Figure 3. Neural network structure

Table 6. Criticalness assignment table

Criticalness of the code	Assigned weight
General purpose code	1
Higher critical code	2
Mission critical code	3

$$Var\_val = \sum (var[i]*var\_comp[i])$$

$$\sigma = var\_val*complexity*criticalness$$

$$Effort = (\sigma + 13.5)*10/3 \quad (1)$$

The equation is arrived based on the halstead effort estimation model. The effort is estimated on a large number of test cases (the test cases here being the source codes of quick sort, bubble sort, gcd program etc.,) the halstead effort is estimated for the test cases, the effort is obtained in elementary mental discriminations. For the same test cases the value of σ is computed and a large pool of values for the comparison of the proposed variable and the halstead estimated effort is obtained. The constant 13.5 and the multiplying factor 10/3 have been arrived from this large pool of values and their comparisons.

A relation is obtained for the obtained σ values and the estimated values. Thus Equation (1) has been derived.

The obtained values var_val and σ and estimated test effort according to the proposed model passed as training set to the network. The network trains itself to predict the values of weights and threshold values for the activation levels. The network is trained through test data over multiple iterations.

Then the network is provided with the information for the project for which an estimate needs to be obtained. The information is derived from the source code document. The various parameters are estimated from the source code like the variable occurrences, complexity of the code etc. The network provides with the effort in terms of the elementary mental discriminations (as the formula was derived using the Halstead model). The network gets trained with the proposed effort estimation function for the post coding phase.

4.5 Neural Network Structure for Post Coding Effort Estimation

1) Designing the network:

The designing of the network involves the selection of the network architecture. The architecture is chosen in such a way that it is in accordance with the proposed effort estimation function. The proposed effort estimation function for the post coding phase implies the design of the network structure with two input nodes, two hidden nodes and one output node. The two input nodes are provided with the values of var_val and σ at input

layer. The network gives effort in terms of the EMDs on the output layer which consists of only one node, the output node.

2) Training the network:

Training the network involves the compilation of the test data: the test data has been obtained by manually calculating the proposed model effort, var_val, σ, halstead effort for a substantial number of program codes. The training set is provided to the network designed as above. The acceptable error rate is set to 0.1. The network is trained with the compiled test data and the network converges over a period of thousands of iterations.

3) Testing the network:

The proposed model accepts the number of variables and their occurrences, complexity of the code, criticalness of the code as the input and it computes the values of var_val, σ and provides it to the network. The network calculates the estimated effort according to the proposed evolved model and produces an output in terms of elementary mental discriminations (EMD).

Figure 4 shows the neural network structure which has been obtained using the easyneurons freeware application. The figure shows the network structure, the thresholds, and the activation levels on various nodes.

The model developed takes the inputs from the users (project managers) estimates the intermediate values, passes it to neural network structure which was realized and retrieves the information from it and passes it to the model which then evolves the data to provide with the estimated effort as the final output.

5. Application of Proposed Model to Test Cases

The proposed model which has the effort estimation in pre coding phase in person-months and in post coding phase in elementary mental discriminations has been applied to various project data. The data has been obtained from Estimator Pal, Usecase point [14] which has a detailed design report. It has also been applied to other minor projects.

The post estimation model is very cumbersome. It has been applied to obtain the proposed estimated value as well as the value that is obtained from Halstead model.

6. Results and Discussion

The model has been applied to various projects as mentioned above. The following are the results obtained.

Figure 5 gives the comparison of pre code effort esti-

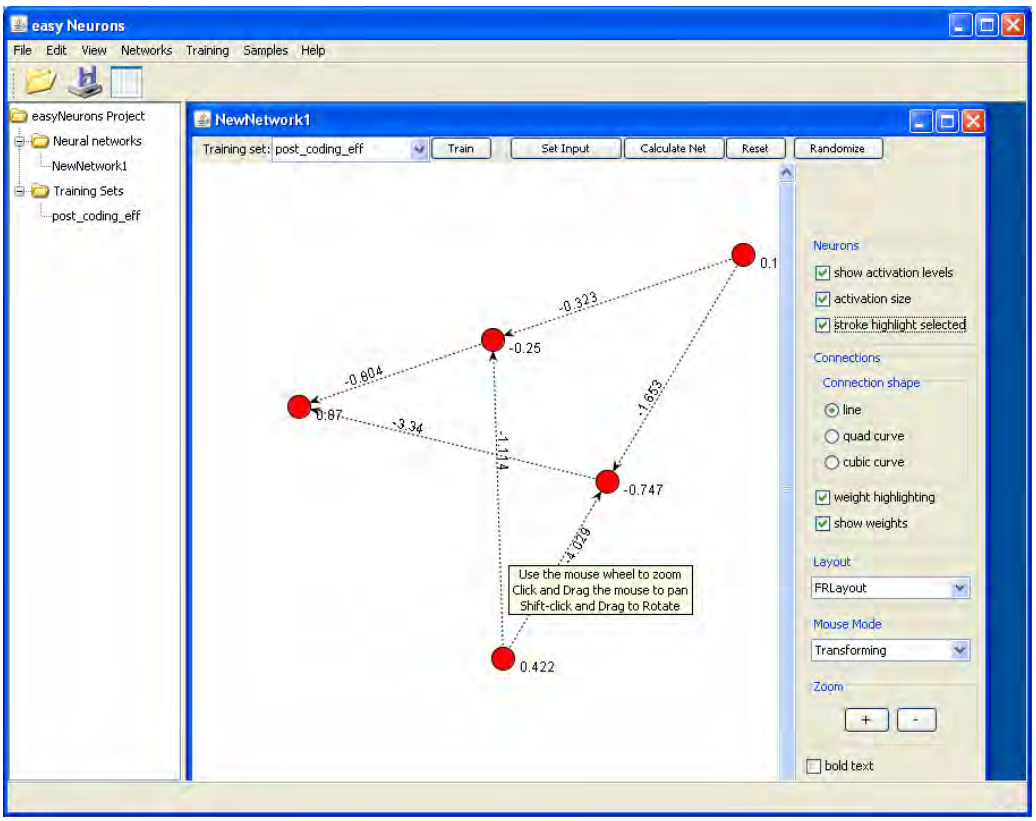


Figure 4. Neural network structure

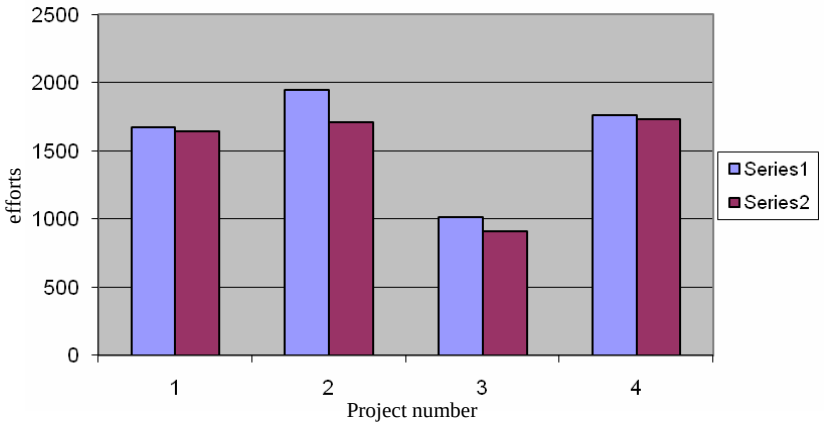


Figure 5. Comparison of pre code effort estimation

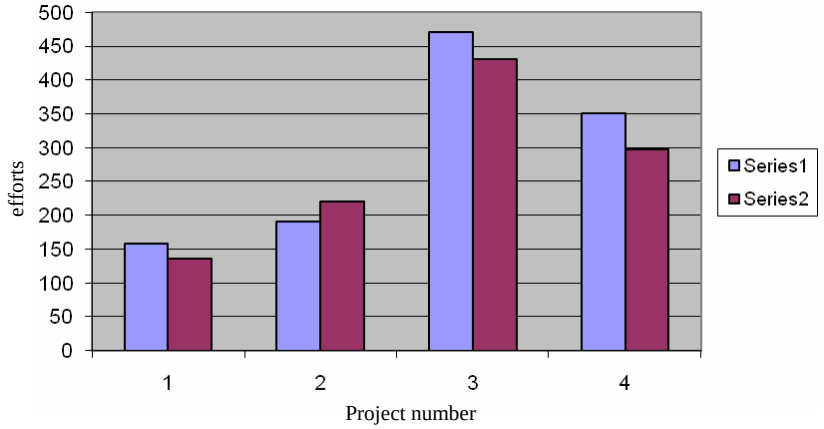


Figure 6. Comparison of post code effort estimations

mations. The X-axis represents the number of the test case and the Y-axis represents the effort interms of person-months. Series1 represents the test effort for pre-coding effort estimation based on the proposed model, while the Series2 represents the pre code effort estimation based on a traditional method. The method to which the proposed method is being compared to is [5] effort estimation based on usecase.

Careful analysis of the results obtained provides the information that the proposed estimation has a deviation of about 8% over the traditional method that has been chosen. This deviation is not much considering the fact that the effort estimated by the traditional method has also not being found to be accurate when applied to real time projects. The method based on usecase points [5] and several other traditional methods haven't produced an accurate estimate of the test effort. The proposed method has been applied on real time data from few of the projects that have been specified above and it has been found to produce an estimate of about 8% deviation from the mentioned effort.

The interpretation of the results obtained and mentioned in the above graph indicate another fact, that the estimated effort has been found to be always on the higher side of traditional method. The deviation found here is found to be on the positive side.

When the results were analyzed with the real time data the proposed model has been found to be more accurate than the traditional method that has been chosen. The proposed model estimated the effort more accurately.

Figure 6 shows the comparison of effort estimation for post coding phase. The X-axis represents the number of the project and the Y-axis gives the effort estimation in terms of elementary mental discriminations. Series1 represents the test effort estimation based on the proposed model which was evolved from the halstead model, cyclomatic model [7] and the application of neural network. Series represents the test effort estimation based on the Halstead model [6].

It can be observed from the graph and the analysis of the results which were obtained by applying the proposed model over several projects that there is about 10% deviation in the test effort estimation for halstead model. The model has been applied to various projects mentioned as above for post effort estimation.

The deviation has also been found to be varying and it has been seen that it is both on the positive side and negative side of the halstead effort. It can be observed

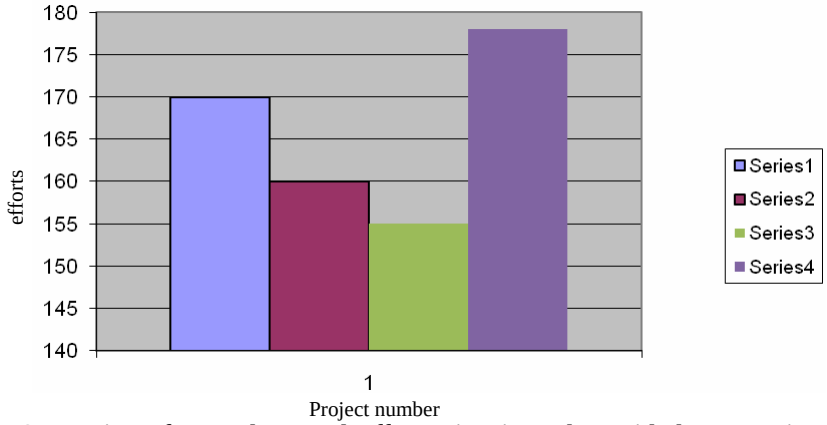


Figure 7. Comparison of pre and post code effort estimations, along with the conventional estimates

that the cyclomatic complexity model [7] and the halstead model [6] haven't been able to estimate the effort accurately. In general there has not been any model that could estimate the effort estimation accurately.

There is no accurate effort estimation for post coding phase. The proposed model has produced results which are in synchronization with the actual effort estimations and found to be more accurate.

In **Figure 7** the comparison of pre and post code effort estimations is given. The model developed has been applied to some student projects and the graph is plotted. In the figure the X axis represents the project number and Y-axis represents the effort. Series1 indicates the pre coding test effort for the proposed model and Series2 represents traditional method pre coding effort estimation, Series3 represents the post coding test effort for the proposed model and Series4 represents the traditional method post coding test effort estimation. It is showing a variation of about 8% over large number of projects. Thus it confirms the fact the estimated efforts both in pre and post coding phase have higher accuracy than the conventional models which as shown earlier show large deviation.

7. Conclusions

The models used for the traditional pre coding effort estimations use the usecase point or the function point.

The paper has covered brief details of the various traditional methods for effort estimations both in pre coding phase and in post coding phase. It then had the introduction of various keywords which are a part of the proposed model.

The proposed effort estimation models for pre coding phase based on usecase point and soft computing technique- neural network has been applied to improve upon the accuracy. The method that has been followed and the metric proposed have an advantage that it produces accurate results. For the post coding effort estimation the proposed model estimated the effort based on and used neural network to improve upon accuracy and the results have been found to show that the proposed estimation is in synchronization with the traditional effort estimation models.

The future scope for the proposed model is based in the direction that the model developed needs to be applied to large number of test cases *i.e.*, real time projects as the proposed model has a unique feature of learning through usage. The model converges towards more accurate values as it used over time. The model developed can be evolved even further in the view that more number of parameters which have a minor effect on the effort estimation be also considered for effort estimation and the model can be evolved.

REFERENCES

[1] R. S. Pressman, "Software Engineering – A Practitioner's Approach," 5th Edition, McGraw Hill, New York, 2002.

[2] B. T. Rao and B. Sameet, "A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network," *International Journal of Computer Science and Network Security*, Vol. 9, No. 6, June 2009, pp. 126-131.

[3] H. Zeng and D. Rine, "Estimation of Software Defects Fix Effort Using Neural Network," *IEEE* 28*th Annual International Computer Software and Applications Conference* (*COMPSAC*'04), Los Alamitos, 28-30 September 2004, Vol. 2, pp. 20-21.

[4] K. K. Agarwal, P. Chandra, *et al.*, "Evaluation of Various Training Algorithms in a Neural Network Model for Software Engineering Applications," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 4, July 2005, pp. 1-4.

[5] S. Nageswaran, "Test Effort Estimation Using Use Case Points (UCP)," 14*th International Software/Internet Quality Week*, San Francisco, 29 May-1 June 2001.

[6] T. E. Hastings and A. S. M. Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation," *IEEE Transactions on Software Engineering*,

Vol. 27, No. 4, April 2001, pp. 337-350.

[7] D. S. Kushwaha and A. K. Misra, "Software Test Effort Estimation," *ACM SIGSOFT Software Engineering Notes*, Vol. 33, No. 3, May 2008.

[8] P. S. Sandhu, P. Bassi and A. S. Brar, "Software Effort Estimation Using Soft Computing Techniques," World Academy of Science, Engineering and Technology, 2008, pp. 488-491.

[9] M. Chemuturi, "Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators," J. Ross Publishing, Lauderdale, July 2009.

[10] Free Software Foundation, "Neuroph Framework," Version 3, June 2007.

[11] M. Braz and S Vergilio, "Software Effort Estimation Based on Use Cases," 30*th Annual International Computer Software and Applications Conference* (*COMPSAC*'06), Chicago, 17-21 September 2006, Vol. 1, pp. 221-228.

[12] G. Banerjee, "Use Case Points – An Estimation Approach," Unpublished, August 2001.

[13] J. Kaur, S. Singh and K. S. Kahlon, "Comparative Analysis of the Software Effort Estimation Models," World Academy of Science, Engineering and Technology, Vol. 46, 2008, pp. 485-487.

[14] N. Nagappan, "Toward a Software Testing and Reliability Early Warning Metric Suite," 26*th International Conference on Software Engineering* (*ICSE*'04), Shanghai, 2004, pp. 60-62.

[15] C. Huang, J. Lo, S. Kuo, *et al*., "Software Reliability Modeling and Cost Estimation Incorporating Test-Effort and Efficiency," 10*th International Symposium on Software Reliability Engineering*, Boca Raton, 1-4 November 1999, pp. 62-72.

[16] O. Mizuno, E. Shigematsu, Y. Takagi, *et al*., "On Estimating Testing Effort Needed to Assure Field Quality in Software Development," 13*th International Symposium on Software Reliability Engineering* (*ISSRE*'02), Annapolis, 12-15 November 2002, p. 139.

J. Software Engineering & Applications, 2010, 3: 341-346
doi:10.4236/jsea.2010.34039 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

Sudden Noise Reduction Based on GMM with Noise Power Estimation

Nobuyuki Miyake, Tetsuya Takiguchi, Yasuo Ariki

Graduate School of Engineering, Kobe University, Kobe, Japan.
Email: takigu@kobe-u.ac.jp

Received January 5th, 2010; revised February 27th, 2010; accepted March 2nd, 2010.

ABSTRACT

This paper describes a method for reducing sudden noise using noise detection and classification methods, and noise power estimation. Sudden noise detection and classification have been dealt with in our previous study. In this paper, GMM-based noise reduction is performed using the detection and classification results. As a result of classification, we can determine the kind of noise we are dealing with, but the power is unknown. In this paper, this problem is solved by combining an estimation of noise power with the noise reduction method. In our experiments, the proposed method achieved good performance for recognition of utterances overlapped by sudden noises.

***Keywords*:** *Sudden Noise, Model-Based Noise Reduction, Speech Recognition*

1. Introduction

Sudden and short-term noises often affect the performance of a speech recognition system. To recognize the speech data correctly, noise reduction or model adaptation to the sudden noise is required. However, it is difficult to remove such noises because we do not know where the noise overlapped and what the noise was.

There have been many studies conducted on non-stationary noise reduction in a single channel [1-4]. The target of our study is mostly sudden noise from among these non-stationary noises. There have been many studies on model-based noise reduction [5-7]. These methods are effective for additive noises. However, these reduction methods are difficult to apply for sudden noise reduction directly since these methods require the noise information in order to be carried out.

In our previous study [8], we proposed detecting and classifying these noises before removing them. But there is a problem with this because the noise power is unknown from the classification results, although the kind of noise can be estimated. In this paper, we propose a noise reduction method that uses the results of noise detection and classification to accomplish the noise reduction. The proposed method integrates noise power estimation with the noise reduction based on GMM to solve the aforementioned problem.

2. System Overview

Figure 1 shows the overview of the noise reduction system. The speech waveform is split into small segments using a window function. Each segment is converted to a feature vector, which is a log Mel-filter bank. Next, the system identifies whether or not the feature vector is noisy speech overlapped by sudden noises using a non-linear classifier based on AdaBoost. The system clarifies the sudden noise type only from the detected noisy frame using a multi-class classifier. Then a noise reduction method based on GMM is applied. Even though we apply the proposed technique to the output from AdaBoost, it can be successfully applied to that from a binary identification technique such as SVM.

3. Clustering Noise

There are many kinds of noises in a real environment.

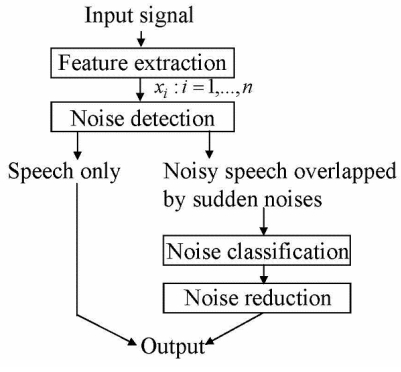


Figure 1. System overview of sudden noise reduction

The smaller the difference between the noise in training and the overlapped noise in the test, the better the performance of the noise reduction method in Section 5 is. But there are many kinds of noises, and potential noises need to be grouped by noise type in some way. Therefore, we made a tree of noise types based on the k-means method, where we used the log Mel-filter bank as the noise feature.

3.1 K-Means Clustering Limited by Distance to Center

K-means clustering usually sets the number of classes. In our method, the number of classes is decided automatically by increasing class so that distance *d* between the data and the center of a class must be smaller than an upper limit θ decided beforehand.

First, all data are clustered using the k-means clustering method. Next, we calculate the distance *d* between the data and the center of the class to which the data belongs. If the distance *d* is bigger than θ ($d > \theta$), this class is divided into two classes and k-means clustering is performed. This step is repeated until all the distances are less than θ.

The noise data for noise reduction is given as the mean value of each class data. So, the smaller the upper limit θ is, the higher the noise reduction performance is expected to be because the variance of the class becomes smaller.

3.2 Tree of Noise Types

One problem with the above k-means algorithm is that too many classes may be created when θ is set small. This problem is solved by making a tree using the above k-means clustering, while θ is set at a larger value and all the data are clustered. The bigger the level is, the less distance there is. In this paper, θ is set to be reduced by half with each level increment change on the noise tree.

Figure 2 shows an example of one such tree. In this paper, the clustering is performed using the mean vectors of each type of noise.

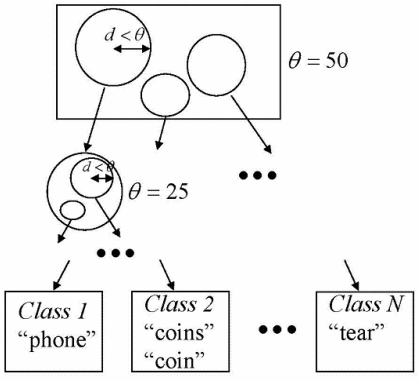


Figure 2. An example of a tree of noise types

4. Noise Detection and Classification

4.1 Noise Detection

Noise detection and classification are described in [8]. A non-linear classifier $H(x)$, which divides clean speech features and noisy speech features, is learned using AdaBoost. Boosting is a voting method using weighted weak classifiers and AdaBoost is one method of boosting [9]. The AdaBoost algorithm is as follows.

Input: *n* examples $Z = \{(x_1, y_1),...,(x_n, y_n)\}$ where y_i means a label of x_i and it is {-1,1}

Initialize:

$$w_1(z_i) = \begin{cases} \frac{1}{2m}, & if \quad y_i = 1 \\ \frac{1}{2l}, & if \quad y_i = -1 \end{cases}$$

where, *m* is the number of positive data, and *l* is the number of negative data.

Do for $t = 1,\ldots,T$

1) Train a base learner with respect to weighted example distribution w_t and obtain hypothesis $h_t : x \to \{-1.1\}$

2) Calculate the training error of h_t

$$e_t = \sum_{i=1}^{n} w_t(z_i) \frac{-y_i h_t(x_i) + 1}{2}$$

3) Set

$$\beta_t = \log \frac{1 - e_t}{e_t}$$

4) Update example distribution w_t

$$w_{t+1}(z_i) = \frac{w_t(z_i) \exp\{-\beta_t y_i h_t(x_i)\}}{\sum_{j=1}^{n} w_t(z_i) \exp\{-\beta_t y_i h_t(x_i)\}}$$

Output: final hypothesis

$$f(x) = \sum_t \beta_t h_t(x)$$

AdaBoost algorithm uses a set of training data, $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where x_i is the *i*-th feature vector of the observed signal, and *y* is a set of possible labels. For noise detection, we consider just two possible labels, $Y = \{-1, 1\}$, where label 1 means noisy speech and label −1, means speech only. In this paper, single-level decision trees (also known as decision stumps) are used as weak classifiers, and the threshold of $f(x)$ is 0.

$$H(x) = \begin{cases} noisy\ speech, & if \quad f(x) \geq 0 \\ clean\ speech, & if \quad f(x) < 0 \end{cases} \tag{1}$$

Using this classifier, we determine whether the frame is noisy or not.

4.2 Noise Classification

Noise classification is performed for the frame detected

as noisy speech. If the frame is noise only, it may be classified by calculating the distance from templates. But it is supposed that the frame contains speech, too. In this paper, we use AdaBoost for noise classification. AdaBoost is extended and used to carry out multi-class classification utilizing the one-vs-rest method, and a multi-class classifier is created. The following shows this algorithm.

Input: *m* examples $\{(x_1, y_1),\ldots,(x_m, y_m)\}$

$$y_i = \{1,...,K\}$$

Do for $k = 1,\ldots,K$

1) Set labels

$$y_i^k = \begin{cases} +1, & if \quad y_i = k \\ -1, & otherwise \end{cases}$$

2) Learn *k*-th classifier $f^k(x)$ using AdaBoost for data set $Z^k = (x_1, y_1^k),...,(x_m, y_m^k)$

Final classifier:

$$\hat{k} = \arg\max_k f^k(x)$$

This classifier is made at each node in tree. *K* is the total number of the noise classes in a node. In this paper, each node has from 2 to 5 classes.

5. Noise Reduction Method

5.1 Noisy Speech

The observed signal feature $X_b(t)$, which is the energy of filter b of the Mel-filter bank at frame *t*, can be written as the follows using clean speech $S_b(t)$ and additive noise $N_b(t)$

$$X_b(t) = S_b(t) + N_b(t) \tag{2}$$

In this paper, we suppose that noises are detected and classified but the SNR is unknown. In other words, the kind of the additive noise is estimated but the power is unknown. Therefore, the parameter α, which is used to adjust the power is used as follows.

$$X_b(t) = S_b(t) + \alpha \cdot N_b(t) \tag{3}$$

In this case, the log Mel-filter bank feature $x_b(t)$ $(= \log X_b(t))$ is

$$\begin{aligned} x_b(t) &= \log\{\exp(s_b(t)) + \alpha \cdot \exp(n_b(t))\} \\ &= s_b(t) + \log\{1 + \alpha \cdot \exp(n_b(t) - s_b(t))\} \\ &= s_b(t) + G_b(s(t), n(t), \alpha) \end{aligned} \tag{4}$$

The clean speech feature $s_b(t)$ can be obtained by estimating $G_b(s(t), n(t), \alpha)$ and subtracting it from $x_b(t)$.

5.2 Speech Feature Estimation Based on GMM

The GMM-based noise reduction method is performed to estimate *s*(*t*) [5,6]. (In [5,6], the noise power parameter α is not considered.) The algorithm estimates the value of the noise using the clean speech GMM in the log Mel-filter bank domain. A statistical model of clean speech is given as an M-Gaussian mixture model.

$$p(s) = \sum_m^M \Pr(m) \cdot N(s; \mu_{s,m}, \Sigma_{s,m}) \tag{5}$$

Here, *N*(*) denotes the normal distribution, and $\mu_{s,m}$ and $\Sigma_{s,m}$ are the mean vector and the variance matrix of the clean speech *s*(*t*) at the mixture *m*. The noisy speech model is assumed using this model as follows:

$$p(x) = \sum_m^M \Pr(m) \cdot N(x; \mu_{x,m}, \Sigma_{x,m}) \tag{6}$$

$$\mu_{x,m} \approx \mu_{s,m} + G(\mu_{s,m}, \mu_n, \alpha) \tag{7}$$

$$\Sigma_{x,m} \approx \Sigma_{s,m} \tag{8}$$

where μ_n is the mean vector for one of the noise classes, which is decided by the result of the noise classification. At this time, the estimated value of $G(s,n,\alpha)$ is given as follows:

$$\hat{G}(s,n,\alpha) = \sum_m P(m \mid x) \cdot G(\mu_{s,m}, \mu_n, \alpha) \tag{9}$$

where,

$$p(m \mid x) = \frac{\Pr(m) \cdot N(x; \mu_{x,m}, \Sigma_{x,m})}{\sum_m \Pr(m) \cdot N(x; \mu_{x,m}, \Sigma_{x,m})} \tag{10}$$

The clean speech feature *s* is estimated by subtracting $\hat{G}(s,n,\alpha)$ from feature *x* of the observed signal.

$$s = x - \hat{G}(s,n,\alpha) \tag{11}$$

5.3 Noise Power Estimation Based on EM Algorithm

The parameter α, which is used to adjust the noise power, is unknown. Therefore, (9) cannot be used because $\mu_{x,m}$ and *p*(*m*|*x*) depend on α. In this paper, this parameter is calculated by the EM algorithm. The EM algorithm is used for estimation of noise power α for maximizing *p*(*x*) which is the likelihood of a noisy speech feature. *p*(*x*) is written as (6), in which $\mu_{x,m}$ depends on α. So, we replace *p*(*x*) with *p*(*x*|α), and the noise power parameter α is calculated by maximizing likelihood *p*(*x*|α) using the EM algorithm.

E-step:

$$Q(\alpha^{(k)}, \overline{\alpha}) = \sum_m p(x, m \mid \alpha^{(k)}) \log p(x, m \mid \overline{\alpha}) \tag{12}$$

M-step:

$$\alpha^{(k+1)} = \arg\max_{\overline{\alpha}} Q(\alpha^{(k)}, \alpha) \tag{13}$$

where *k* is the iteration index. The above two steps are calculated repeatedly until $\alpha^{(k)}$ converges to optimum

solution. In M-step, the solution is found by calculating the following equation.

$$\frac{\partial Q(\alpha^{(k)},\overline{\alpha})}{\partial \overline{\alpha}} = 0 \tag{14}$$

This equation can be expanded as follows.

$$\begin{aligned}&\frac{\partial Q(\alpha^{(k)},\overline{\alpha})}{\partial \overline{\alpha}}\\ &= \frac{\partial}{\partial \overline{\alpha}} \sum_m p(x,m \mid \alpha^{(k)}) \log p(x,m \mid \overline{\alpha})\\ &= \sum_m p(x,m \mid \alpha^{(k)}) \cdot \sum_b \frac{x_b - \mu_{s,m,b} - \log(1+\overline{\alpha}\cdot\exp(\mu_{n,b}-\mu_{s,m,b}))}{\sigma_{b,b}^2(1+\overline{\alpha}\cdot\exp(\mu_{n,b}-\mu_{s,m,b}))}\end{aligned} \tag{15}$$

However, it is difficult to find a solution of this equation analytically. So, Newton's method is used for this equation. An approximation of the optimum solution is calculated repeatedly as follows using Newton's method.

$$\begin{aligned}f_1 &= \frac{\partial Q}{\partial \overline{\alpha}}(\alpha^{(k)},\overline{\alpha}^{(l)})\\ f_2 &= \frac{\partial^2 Q}{\partial \overline{\alpha}^2}(\alpha^{(k)},\overline{\alpha}^{(l)})\\ \overline{\alpha}^{(l+1)} &= \overline{\alpha}^{(l)} - \frac{f_1}{f_2}\end{aligned} \tag{16}$$

Equation (16) is calculated repeatedly until α converges. The initial value of Newton's method was set at 0.

6. Experiments

In order to evaluate the proposed method, we carried out isolated word recognition experiments using the ATR database for speech data and the RWCP corpus for noise data [10].

6.1 Experimental Conditions

The experimental conditions are shown in **Table 1**. All features were gotten in a 20 ms window by 10 ms frame shift. The word utterances of ten different people are recorded in the ATR database. There were 105 types of noises in the RWCP corpus [10]. The kinds of noises, for example, are telephone sounds, beating woods, tearing paper and so on. One kind of noise consists of 100 data samples, which are divided into 50 samples for testing and 50 samples for training. The noise tree was made using the mean vectors of the training samples, and these vectors were divided into 37 classes (which is the total number of leaves). Learning classifiers for detection and classification were performed using the noisy speech features. So, we made noisy utterances in each class, adding noises to 2,000 × 10 clean utterances of 10 persons (five men, five women) for training data. Clean utterances were in ATR database which were Japanese word utterances of 10 persons. In this case, SNR is adjusted between –5 dB and 5 dB. One model of GMM for noise reduction and HMM for recognition were learned using the same 2,000 × 10 clean utterances of 10 persons. In order to make test data, we used 500 × 10 different word utterances by the same 10 persons. Some noises overlapped one test utterance with adjusting SNR to –5, 0 and 5 dB and duration time of each noise to 10 ~ 200 ms. **Figure 3** shows an example of noisy speech.

Table 1. Experimental conditions

Making tree.

Feature parameters	24-log Mel-filter bank
Tree depth	5
Upper limit θ (in order of depth level)	50, 25, 12, 6

Detection and classification

Feature parameters	24-log Mel-filter bank
Number of weak learners	200

Noise reduction

Feature parameters	24-log Mel-filter bank
Number of components of GMM	16, 32, 64

Speech recognition

Feature parameters	12-MFCC+ Δ + Δ Δ
Acoustic models	Phoneme HMM 5 states, 12 mixtures
Lexicon	500 words

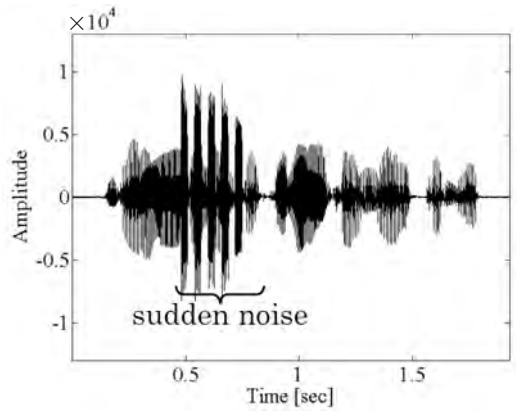


Figure 3. An example of noisy speech

6.2 Experimental Results

Table 2 shows the results of detection and classification. "Recall" is the ratio of detected true noisy frames among all the noisy frames, "Precision" is the ratio of detected true noisy frames among all the detected frames and "Classification" is the rate of true classification frames among the detected noisy frames. In this table, Recall rate and Precision rate are higher value, which mean noise is well detected. The classification rate was low, however. Even if the classification results are different

from the real noise label, though, if the noises are classified near to the real noise, the negative effect on noise reduction may be negligible.

Figure 4 shows the recognition rate for each SNR. In **Figure 4**, the baseline means noise reduction is not applied and "No estimation of noise power" means that power estimation was not performed in GMM-based noise reduction (calculated in (11) as $\alpha = 1$). "EM algorithm" means that noise power is estimated using the method written in section 5.3. "Oracle label" means that correct detection and classification results were given. In this case (Oracle-label), 64 Gaussian components were used. In cases where there were no noises, the recognition rate is 97.4%. As shown in **Figure 4**, the recognition rate was improved by using the proposed method. Furthermore, the proposed method has higher performance than no estimation.

Table 2. Results of detection and classification

	5 dB	0 dB	-5 dB
Recall	0.850	0.908	0.942
Precision	0.861	0.868	0.871
Classification	0.290	0.382	0.406

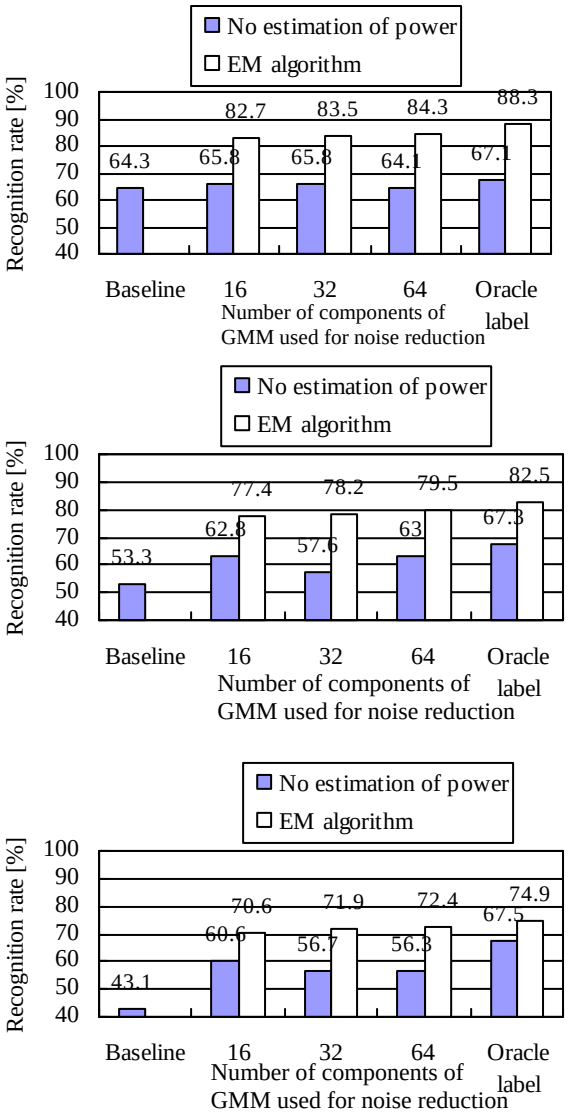


Figure 4. Recognition results at SNRs of –5 dB, 0 dB and 5 dB

6.3. Experiments for Unknown Noise

We examined the effectiveness of the proposed method for dealing with unknown noises using 10-fold cross validation of noise type. 105 types of noise were divided into 10 sets, with 9 sets for training and 1 set for testing. The noise tree and classifiers were created using training sets and test data were made using test sets. Experimental conditions were similar to those in **Table 1**, but we examined only 64 Gaussian mixture components for noise reduction. **Table 3** shows the detection results. Classification rate cannot be evaluated because the classes of the noises that overlapped utterances are not defined. **Figure 5** shows recognition rate for unknown noises for test sets. As shown in this **Figure 5**, the proposed method improved the word recognition rate for unknown noises. But, in comparison with the "Oracle label", the performance of speech recognition degraded due to differences between the training and test noise data.

7. Conclusions

In this paper, we have described a sudden noise reduction method. Noise detection and classification are performed using AdaBoost, and GMM-based noise reduction is performed using the detection and classification results. Combining an estimation of noise power with the noise reduction method, we solved the problem of word recog-

Table 3. Results of detection for unknown noises.

	5 dB	0 dB	-5 dB
Recall	0.831	0.886	0.926
Precision	0.849	0.856	0.860

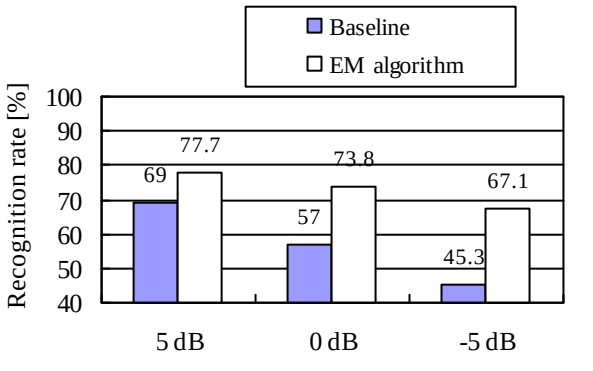


Figure 5. Recognition results for words utterances mixed unknown noises

nition when that noise power was unknown. Our proposed method improved the word recognition rate, although admittedly, the classification accuracy was not high. Furthermore, although this method was effective for unknown noises, it will need combination of a noise adaptation, tracking technique and so on. In future research, we will attempt to verify effectiveness of this new method in dealing with sudden noise when a large vocabulary is used.

REFERENCES

[1] M. Fujimoto, *et al*., "Particle Filter Based Non-Stationary Noise Tracking for Robust Speech Recognition," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), 2005, pp. 257-260.

[2] M. Kotta, *et al*., "Speech Enhancement in Non-Stationary Noise Environments Using Noise Properties," *Speech Communication*, Vol. 48, No. 11, 2006, pp. 96-109.

[3] T. Jitsuhiro, *et al*., "Robust Speech Recognition Using Noise Suppression Based on Multiple Composite Models and Multi-Pass Search," *Proceedings of Automatic Speech Recognition and Understanding* (*ASRU*), 2007, pp. 53-58.

[4] T. Hirai, S. Kuroiwa, S. Tsuge, F. Ren, M. A. Fattah, "A Speech Emphasis Method for Noise-Robust Speech Recognition by Using Repetitive Phrase," *Proceedings of International Conference on Chemical Thermodynamics* (*ICCT*), 2006, pp. 1-4.

[5] P. J. Moreno, B. Raj and R. M. Stern, "A Vector Taylor Series Approach for Environment Independent Speech Recognition," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), 1996, pp. 733-736.

[6] J. C. Segura, *et al*., "Model-Based Compensation of the Additive Noise for Continuous Speech Recognition. Experiments Using the AURORA II Database and Tasks," *Proceedings of Eurospeech*, 2001, pp. 221-224.

[7] L. Deng, *et al*., "Enhancement of Log Mel Power Spectra of Speech Using a Phase-Sensitive Model of the Acoustic Environment and Sequential Estimation of the Corrupting Noise," *IEEE Transactions on Speech and Audio Processing*, Vol. 12, 2004, pp. 133-143.

[8] N. Miyake, T. Takiguchi and Y. Ariki, "Noise Detection and Classification in Speech Signals with Boosting," *IEEE Workshop on Statistical Signal Processing* (*SSP*), 2007, pp. 778-782.

[9] Y. Freund, *et al*., "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, Vol. 55, 1997, pp. 119-139.

[10] S. Nakamura, *et al*., "Acoustical Sound Database in Real Environments for Sound Scene Understanding and Hands-Free Speech Recognition," *Proceedings of* 2*nd ICLRE*, 2000, pp. 965-968.

J. Software Engineering & Applications, 2010, 3: 347-363
doi:10.4236/jsea.2010.34040 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Mixed-Model U-Shaped Assembly Line Balancing Problems with Coincidence Memetic Algorithm

Parames Chutima, Panuwat Olanviwatchai

Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
Email: parames.c@chula.ac.th

Received December 15th, 2009; revised January 8th, 2010; accepted January 25th, 2010.

ABSTRACT

Mixed-model U-shaped assembly line balancing problems (*MMUALBP*) *is known to be NP-hard resulting in it being nearly impossible to obtain an optimal solution for practical problems with deterministic algorithms. This paper presents a new evolutionary method called combinatorial optimisation with coincidence algorithm* (*COIN*) *being applied to Type I problems of MMUALBP in a just-in-time production system. Three objectives are simultaneously considered; minimum number workstations, minimum work relatedness, and minimum workload smoothness. The variances of COIN are also proposed, i.e. CNSGA II, and COIN-MA. COIN and its variances are tested against a well-known algorithm namely non-dominated sorting genetic algorithm II* (*NSGA II*) *and MNSGA II* (*a memetic version of NSGA II*). *Experimental results showed that COIN outperformed NSGA II. In addition, although COIN-MA uses a marginal CPU time than CNSGA II, its other performances are dominated.*

Keywords: *Assembly Line Balancing, Mixed-Model U-Line, JIT, COIN*

1. Introduction

An assembly line comprises a sequence of workstations through which a predefined set of tasks are performed repeatedly on product units while they are moving along the line. It was originally developed to support mass production of single homogeneous standardised commodity to gain a competitive unit cost. Fierce competition in the current market as well as ever-changing customer requirements forces the mass production concept to become no longer attractive. Manufacturers need to redesign their production lines to accommodate mixed-model production known as mixed model assembly lines (MMALs). In MMALs, all models with the same standardised platform but different customisable product attributes are classified in the same family [1]. General-propose machines with automated tool changing equipment and highly flexible operators are necessary to realise an arbitrarily intermixed sequence of various models of a standardised product with similar process requirements to be assembled on the same line at negligible setup costs.

Typically, workstations on the assembly line are aligned straight along a conveyer belt. Monotone and boring types of work in the straight line layout may not challenge the working enthusiasm of operators, as well as being inflexible to manage changes in external environments. As a consequence of just-in-time (JIT) implementation, manufacturers aim to achieve continuously improved productivity, cost, and product quality by eliminating all wastes in their production systems [2]. However, the straight line cannot fully support the adoption of JIT principles to manufacturing especially in the utilisation of multi-skilled operators. Hence, such companies as Allen-Bradley and GE have replaced their traditional straight lines with U-shaped production lines, called U-lines hereafter [3]. **Figure 1** shows the configuration of the U-line.

In the U-line, the entrance and the exit are placed on the same position. A rather narrow U-shape is normally formed since both ends of the line are located closely

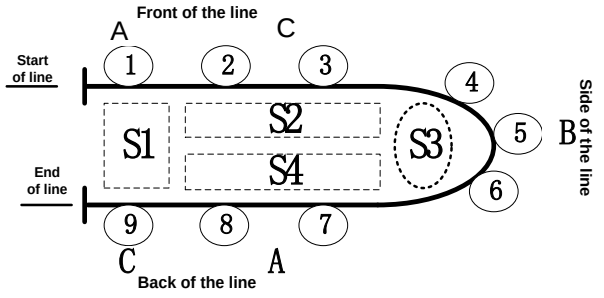


Figure 1. Configuration of a U-shaped assembly line

together. Tasks are arranged around the U-line and are organised into workstations. A part of the U-line comprising a set of tasks with the same directional alignment as the entrance is called *front* zone. The opposite side of the front zone where the exit-side is located is called *back* zone. The set of tasks joining front and back zones and being the base of the U-line is located in *side* zone. Two kinds of workstations can be formed in the U-line. A regular workstation comprises tasks located sequentially along the front (S2), back (S4), or side (S3) of the U-line; whereas, a crossover workstation (S1) includes tasks located on both the front and back of the line.

Multi-skilled operators are located inside of the line. Since some certain models have to visit a crossover workstation twice, the operator in charge of that crossover workstation may have to process two different models in the same cycle. For example, operator S1 performs task 1 of model A on the front side of the line, travels to the back side of the line to perform task 9 of model C, and then returns to the front side of the line to begin the next cycle. The salient characteristic of crossover workstations of the U-line poses additional challenges for improved performances.

Compared to the straight-line, the U-line gains popularity from its benefit offerings including improving visibility and communication, operator flexibility, rotatable multi-skilled operators, know-how sharing, enhancing teamwork, better quality control, prompt problem solving, faster corrective action on rework, higher product quality, easily adjustable output rate, volume flexibility, eliminating the need for special material handling equipment, fewer workstations, higher machine utilisation, and higher line effectiveness for breakdown prone systems [4-7].

The U-line is an inevitable element and becomes a cornerstone to obtain the main benefits of JIT production principles, *i.e.* one-piece flow manufacturing, smoothened workload, and multi-skilled workforce. The U-line is expected to gain much more popularity in industries in the future. The survey found that nearly 75% of available U-lines are configured to produce a product with different models or more than one type of product on the same line [8]. This type of production is called a mixed-model U-line (MUL). MUL has gradually superseded traditional mixed-model straight line due to its greater efficiency offerings, e.g. productivity, flexibility, cost, adaptability to demand changes, machine utilisation, and quality [9].

Successful utilisation of MUL needs effective solutions to mixed-model U-line balancing (MULB) and mixed-model U-line sequencing (MUS). MULB, a long to medium-term decision with a typical planning horizon of several months, is a problem of determining the number and sequence of workstations on the line or the cycle time of the line to accommodate the different models of products; whereas MUS, a short-term decision normally revising on a daily basis, is a problem of determining a production sequence of mixed models introduced to the line to achieve given objectives. Although these two problems are heavily interrelated, they are normally addressed independently and hierarchically due to their own computational complexities involved. This paper will focus on the MULB problem.

A great deal of research has been conducted on the line balancing problems since it was first published in mathematical form by Salveson [10]. Comprehensive literature reviews presented by [3,11-14]. Boysen *et al.* [15] indicated that very little has been done concerning the U-line balancing problem. Since Monden [4] brought U-lines to the attention of research community, the first pioneer study of the U-line balancing problems was published by Miltenburg and Wijngaard [5]. They developed a dynamic programming (DP) procedure for a single-model U-line to determine the optimal balance for Type I of U-line balancing problems (minimum number of workstations) with up to 11 tasks. However, DP was reported impractical for obtaining optimal balances for large-sized problems. They then developed a single-pass heuristic namely *U-line maximum ranked positional weight* to use for larger problems (111 tasks) where the priority of each task is given to either the time required to complete both that task and all the tasks that must succeed or must precede it, whichever is larger. The heuristics showed satisfactory performance for large-sized problems. Sparling and Miltenburg [16] proposed an approximate solution algorithm to solve the MMUALB problem up to 25 tasks. The algorithm transformed the multi-model problem into an equivalent single-model problem. The optimal balance was solved by branch and bound algorithm with exponential computational requirement to find minimum number of workstations. Smoothing algorithm was used to adjust the initial balance to reduce the level of model imbalance. Miltenburg [17] presented a reaching dynamic algorithm to balance and rebalance a U-line facility that consists of numerous U-lines connected by multiline stations. The objective when balancing such a facility is to assign tasks to a minimum number of stations while satisfying cycle time, precedence, location, and station-type constraints. A secondary objective is to concentrated idle time in a single station. The proposed algorithm can solve U-line balancing problem with no more than 22 tasks without wide, sparse precedence graphs.

Urban [18] presented an integer programming formulation for determining the optimal balance for the U-line balancing (ULB) problem. By eliminating some variables through the use of bounds, the size of the model was reduced. It was shown that the proposed formulation can optimally balance larger problems (21 to 45 tasks) than the DP procedure of Miltenburg and Wijngaard [5]. Ajenblit and Wainwritght [19] were the first who applied a genetic algorithm (GA) for Type I ULB problems with

the objectives of minimising total idle time, balancing workload among stations, or a combination of both. Several algorithms for assigning tasks to workstations were proposed. The fitness value of a chromosome is determined by applying all these algorithms to it and the one with lowest fitness value is selected. They found that these assignment algorithms proved to have merit and GA proved to be computationally efficient. Scholl and Klein [20] considered different types of the ULB problem, *i.e.* Type I, Type II, and Type E. A new branch and bound procedure called ULINO (U-line optimiser) adapted from their previous algorithm developed for the straight-shaped problem called SALOME was proposed. Computational results of up to 297-task problems showed that the procedures yielded promising results in limited computation time.

Erel *et al*. [21] proposed an algorithm with a coupling of a solution generator module and a simulated annealing (SA) module. The generator assigned tasks sequentially to separate stations and combines two adjacent stations with minimum total station time until infeasibility is found. Then, SA reconstructed feasibility for such solutions by reassigning the tasks in the combined station to other stations by minimizing the maximum station time. The algorithm was tested on a variety of data sets with up to 297 tasks and found quite effective. Aase *et al*. [22] proposed a branch-and-bound (B&B) solution procedure called U-OPT (U-line OPTimisation) for a ULB problem. Four design elements of the B&B procedure are investigated including branching strategies, fathoming criteria, heuristics to obtain upper bounds at each node, and identification of initial setting solutions. Paired-task lower bound was largely responsible for the dominance in the efficacy of U-OPT over existing methods. Aase *et al*. [23] conducted empirical experiments to confirm that the U-shaped layout can significantly improve labour productivity over the traditional straight-line one. Interestingly, the improvement tends to be higher during high demand periods when operators are assigned three or fewer tasks on average, when the problem size is small, and when assembly sequence is fairly well structured.

Martinez and Duff [24] applied heuristic rules adapted from the simple line balancing problem to the Type I UALB problems up to 21 tasks. Some heuristics were found to produce optimal results. To achieve improved solutions, each gene in a chromosome of GA representing the heuristic rule was used to break ties during the task assignment process. Balakrishnan *et al*. [25] modified 13 single-pass heuristics to balance U-lines with the existent of travelling time and investigated their effectiveness under various problem conditions. Gokcen and Agpak [26] developed a goal programming model for the ULB problems up to 30 tasks. This approach offers increased flexibility to the decision maker since conflicting goals can be dealt with at the same time. Urban and Chiang [27] considered the ULB problem with stochastic task times and developed a linear, integer program using a piecewise approximation for the chance constraints to find the optimal solution. The proposed method effectively solved practical-sized problems optimally up to 28 tasks. Chiang and Urban [28] developed a hybrid heuristics comprising an initial feasible solution module and a solution improvement module for the stochastic ULB problem. The heuristic can identify optimal or near-optimal solutions for up to 111-task problems. Kara *et al*. [29] developed a binary fuzzy goal programming for 8-task ULB with fuzzy goals that allow decision makers to consider the number of workstations and cycle time as imprecise values.

Baykasoglu [30] proposed multi-objective SA for ULB problems with the aim of maximising smoothness index and minimising the number of workstations. Task assignment rules were used in constructing feasible solutions. The optimal solutions for each problem were found in short computation times. Hwang *et al*. [31] developed a priority-based genetic algorithm (PGA) for ULB problems for up to 111 tasks. A weighted-sum objective function comprising the number of workstations and the workload variation was considered. The proposed model obtained improved workload variation, especially for large size problems. Hwang and Katayama [32] proposed an extension version of PGA namely an amelioration structure with genetic algorithm (ASGA) to deal with workload balancing problems in mixed-model U-shaped lines for up to 111 tasks. ASGA was able to find better solutions than PGA in terms of workload variation. Boysen and Fliedner [33] proposed a general solution procedure for U-shaped assembly line balancing using an ant colony optimisation (ACO) approach. Their procedure was versatile in the sense that various line balancing features found in practice can be incorporated into the model. Baykasoglu and Dereli [33] proposed ACO that integrates COMSOAL and ranked positional weight heuristics for solving ULB problems. The proposed algorithm found optimum solutions in short computational times.

The existent of crossover workstations in MUL opens a chance for MUS, a part from MULB, to smoothen workload distribution among workstations since the crossover workstation allows a model mix to be processed in a cycle. As a result, MULB and MUS can play significant roles in workloads moothening of MUL. Since these problems are highly correlated, especially when the workload smoothening objective needs to be achieved, several researchers have attempted to solve these two problems simultaneously in an aggregated manner. Miltenburg [34] modelled the joint problems of line balancing and model-sequencing for mixed-model U-lines operated under a JIT environment and proposed a solution algorithm for solving both problems simultane-

ously. Kim *et al*. [35] developed a symbiotic evolutionary algorithm called co-evolutionary algorithm (PCoA), which imitates the biological co-evolution process through symbiotic interaction, to handle the integration of balancing and sequencing problem in MUL for up to 111 tasks. Later, Kim *et al*. [36] proposed an endosymbiotic evolutionary algorithm (EEA), an extended version of the symbiotic evolutionary algorithm, to simultaneously solve line balancing and sequencing in MUL. The proposed algorithm obtained much better quality solutions than PCoA and a traditional hierarchical approach. Agrawal and Tiwari [37] demonstrated the superiority of the collaborative ant colony optimisation in simultaneously tackling disassembly line balancing and sequencing problem in MUL for up to 80 tasks. Sabuncuoglu *et al.* [38] developed a family of ant colony algorithms that make both sequencing and task assignment decisions simultaneously for ULB problems up to 111 tasks.

Kara *et al*. [9] proposed SA to deal with a multi-objective approach for balancing and sequencing MULs in JIT production systems for up to 30 tasks to simultaneously minimise the weighted sum of the absolute deviations of workloads across workstations, part usage rate, and cost of setups. Kara *et al*. [39] proposed SA based heuristic approach for solving balancing and sequencing problems of mixed-model U-lines simultaneously for up to 30 tasks. SA was capable of minimising the number of workstations and minimising the absolute deviation of the workloads among workstations. Kara [40] presented a mixed, zero-one integer, nonlinear programming for mixed-model U-line balancing and sequencing problems for up to 111 tasks with the objective of minimising absolute deviation of workloads. An efficient SA was also proposed and its performance outperformed PCoA and EEA.

Literature has demonstrated that the MULB is an important problem for modern assembly systems operated under JIT environment. Although several exact methods for their solutions were proposed, only small sized problems can be optimally solved due to the complexity of the problem. Hence, a computational more effective algorithm is needed for larger sized problem. Also, the algorithm has to be able to easily handle multiple objectives simultaneously. In this paper, such an algorithm that utilises the concept of evolutionary algorithm namely combinatorial optimization with coincidence algorithm (COIN) is proposed for multi-objective MULB problems. Three objectives including minimum number workstations, minimum work relatedness, and minimum workloads moothness are considered simultaneously. The performances of COIN are compared with a well-known algorithm namely non-dominated sorting genetic algorithm II (NS GA II) and their memetic versions The purpose of this study is to see the feasibility and effectiveness of the COIN approach which is one of the most recent meta-heuristics to solve this well-known problem and compare it against others in terms of quality of solutions and solution time.

The organisation of this paper is as follows. In the next section, the detailed description of the multi-objective optimisation problem is presented, followed by an explanation of the multi-objective MU LB problems. The proposed algorithm to solve MULB problems is elaborated next, and the experimental design and results are explained respectively. Finally, the concluding remark of the research is given.

2. Multi-Objective Evolutionary Algorithms

A multi-objective optimisation problem (MOP) is related to the problem where two or more objectives have to be optimised simultaneously. Generally, such objectives are conflicting and represented in different measurement units, preventing simultaneous optimisations of each one. MOP can be formulated, without loss of generality, as follows:

$$Minimize_{x\epsilon\Omega}\{f_1(x), f_2(x), \dots, f_k(x)\} \quad (1)$$

where solution x is a vector of decision variables for the considered problem; Ω is the feasible solution space; and $f_i(x)$ is the i^{th} objective function $(i = 1, 2, \dots, k)$. Two approaches often employ to solve MOP. The first approach is to combine each objective function into a single composite function, e.g. weighted sum method, utility theory, *etc*. The advantage of this method is straightforward computation. However, two practical problems are often experienced with this approach: 1) selection of the suitable weights can be very difficult even for those who are unfamiliar with the problem and 2) small perturbations in the weights can sometimes lead to totally different solutions [41]. As a result, the second approach, e. g. multi-objective evolutionary algorithms (MOEAs), has come into play. This approach determines a set of alternative solutions for (1) rather than a single optimal solution. These solutions are optimal in the wider sense such that no other solutions in the search space are superior to them when all objectives are considered. A decision vector x is said to dominate a decision vector y (also written as $x \succ y$) if:

$$f_i(x) \leq f_i(y), \text{ for all } i \in \{1, 2, \dots, k\} \quad (2)$$

and $f_i(x) < f_i(y)$, for at least one $i \in \{1, 2, \dots, k\}$ (3)

All solutions that dominate the others but do not dominate themselves are called *non-dominated (superior) solutions*. A *Pareto-optimal solution* is a global optimal solution which is not dominated by any other solutions in the feasible solution space. A set that contains all feasible Pareto-optimal solutions is called a *Pareto-optimal set* or *efficient set*. The collection of the points of the Pareto-optimal set (or the corresponding images of the Pareto-optimal set) along a curve in the

objective space that has a set of attributes collectively dominating all other points not on the frontier are termed the *Pareto-optimal frontier (front)* or *efficient frontier* (*front*). An example of the Pareto-optimal solutions for a two-objective minimisation problem is illustrated in **Figure 2.** It is obvious that an amount of sacrifice in one objective is always incurred to achieve a certain amount of gain in the other (inverse relationship) while moving from one Pareto-optimal solution to another. Providing Pareto-optimal solutions to the decision maker is more preferable to a single solution since practically, when considering real-life problems, a final decision is always based on a trade-off between conflicting objective functions.

MOEAs have recently become popular and have been applied to a wide range of problems from social to engineering problems [42]. In general, MOEAs are ideally suited to MOP because they are capable of searching a whole set of multiple Pareto-optimal solutions in a single run. In addition, the shape or continuity of the Pareto-optimal frontier has less effect to MOEAs than traditional mathematical programming. The approximation of a true Pareto-optimal set involves two conflicting objectives: 1.) the distance to the true Pareto frontier is to be minimised, and 2.) the diversity of the evolved solutions is to be maximised [43]. To achieve the first objective, a Pareto-based fitness assignment is normally designed to guide the search toward the true Pareto-optimal frontier [44, 45].

In the view of the second objective, some MOEAs successfully provide density estimation methods to preserve the population diversity. Although several versions of MOEAs have been developed [42], non-dominated sorting genetic algorithms-II (NSGA II) [46] is among the most promising one in terms of convergence speed to Pareto-optimal solutions and even distribution of the Pareto frontier. NSGA II is an elitist multi-objective genetic algorithm being introduced by Deb *et al*. (2002). It uses a fixed population size of N for both parent and offspring populations. Once a new offspring population is created, it is combined with its parent population. The size of the combined population becomes $2N$. A non-dominated sorting method is used to identify Pareto frontiers (F_1, F_2, ..., F_k) in the combined population. The first frontier (F_1) is the best in the combined population. The next population (archive) is created by selecting frontiers based on their rankings; the best Pareto frontier being selected first. If the number of members in the archive is smaller than the population size (N), the next frontier will be selected and so on. If adding a frontier would increase the number of members in the archive to exceed the population size, a truncation operator is applied to that frontier based on the crowded tournament selection by which the winner of two same rank solutions is the one that possesses the greater crowding distance (farther apart from its neighbours). This is to maintain a good spread of solutions in the obtained set of solutions.

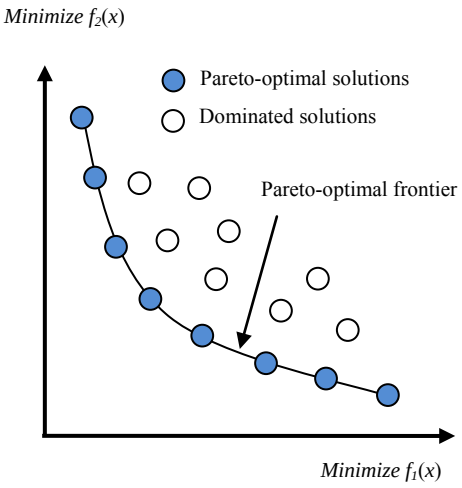


Figure 2. Pareto-optimal solutions

Memetic algorithms (MAs), a type of evolutionary algorithms (EAs), have been recognised as a powerful algorithmic paradigm on complex search spaces for evolutionary computing [47]. MAs are inspired by models of adaptation in nature systems that combine evolutionary adaptation of populations of individuals with individual learning with a lifetime. A *meme* is a unit of information that reproduces itself while people exchange ideas. Memes are adapted by the people who transmit them before being passed on to the next generation. MAs use EAs to perform exploration and use local search to exercise exploitation. A separate local search algorithm can be applied to improve the fitness of individuals by special hill-climbing. Local search in MAs is similar to simple hill-climbing with differences in that 1) the neighbourhood of the current solutions is searched systematically instead of random searching in the space of all candidate solutions, and 2) the neighborhood search is repeated until a locally optimal solution is found. An advantage of local search in MAs over other heuristics is that local exploitation around individual can be performed much more effectively; hence, good solutions in a small region of the search space can be found quickly.

3. Multi-Objective MULB Problem

3.1 MULB Problem

To plan an assembly process for any product on an assembly line, its total amount of work is partitioned into a set of elementary operations namely tasks. Assembly line balancing is the allocation of a set of tasks to workstations without violating any constraints to optimize some measure of performance. Typical constraints include

each task is allocated to one and only one workstation, precedence relationship that reflects technological and organisational constraints among the tasks is not violated, and total task time of any workstation does not exceed the given cycle time [13].

To perform a task on a workstation, not only tools, equipment, machinery, and labour skills have to be selected properly, but also its precedence relationship has to be followed strictly. A precedence diagram is often used to visually demonstrate such relationship. Nodes, node weights, and arrows on the precedence diagram represent tasks, task times, and precedence constraints between tasks, respectively. For MMAL, a merged precedence diagram is needed which can be created as follows [16].

1) Compute the weighted average task time for each task. Let M = the number of models to be produced during a planning horizon, D_m = the demand of product model m (m=1,2,...,M) task i (i=1,2,...,N) of model M has task time = t_{im}, The weighted average task time t_i is

$$t_i = \sum_{m=1}^{M}\{D_m t_{im}\}/\sum_{m=1}^{M} D_m \quad (4)$$

2) Merge the precedence diagram of each model to form the merged precedence diagram. It is assumed that the precedence relationship is consistent from model to model. The merged precedence diagram (MPD) is created by adding arrow xy to MPD if, for any model, task x is an immediate predecessor of task y.

MULB is more complex than the traditional straight line since not only can the set of assignable tasks be considered from the set of tasks whose predecessors have already been assigned (moving forward through MPD and allocating tasks on the front side of the U-line) as the straight line, but also from the set of tasks whose successors have already been assigned (moving backwards through allocating tasks on the back side of the U-line). This permission increases possibility on how to allocate tasks to workstations and often leads to a fewer number of workstations than the straight line. Based on MPD, literature always assumes that each task type is assigned to one and only one workstation regardless of the model [32].

3.2 Objective Functions

Although several measures can be used to evaluate the performance of line balancing in MUL, in this paper three main objectives that support JIT implementation to be simultaneously optimised are evaluated including number of workstations, variation of workload, and variation of work relatedness. Since the type I problem of MULB is considered, a fixed cycle time, assembly task time, and precedence relationship are given and our first objective is to minimize the number of workstations. Achieving this objective can result in low labour cost and less space requirement. If m is the number of workstation, the objective function is formulated as follows.

$$f_1(x) = minimize\ m \quad (5)$$

The second objective is to smooth (minimize variation of) the workloads distributed across workstations. Several benefits can be gained when MUL is operated in this manner including increased production rate, reduced line congestion, but more importantly, mitigates the concerns of inequity in task assignments among workers [35]. The workload smoothness objective can be formulated as follows.

$$f_2(x) = minimize\sqrt{\sum_{k=1}^{m}(S_{max} - S_k)^2/m} \quad (6)$$

where S_k = total time of workstation k, S_{max} = maximum total time of all workstations = $maximize\ S_k$ $(k = 1, 2, \dots, m)$.

The third objective is to minimize variation of work relatedness in a workstation. The purpose of this objective is to allocate interrelated tasks to the same workstation as many as possible. Not only can such an assignment improve work efficiency, but it is also useful to assembly line designers since they may have greater flexibility in locating facilities and workstations. The formulation of this objective is as follows.

$$f_3(x) = minimize\{m - m/\sum_{k=1}^{m} SN_k\} \quad (7)$$

where SN_k= number of relatedness of tasks in workstation k.

Although three objectives are considered simultaneously in this paper, for type I problem, the first objective dominates the others. As a result, if there are two candidate solutions, the one with lower number of workstations will always selected regardless of how good the other two objectives are.

4. Proposed Algorithm

4.1 COIN

Wattanapornprom *et al*. [48] developed a new effective evolutionary algorithm called *combinatorial optimisation with coincidence* (COIN) originally aiming for solving travelling salesman problems. The idea is that most well-known algorithms such as GA search for good solutions by sampling through crossover and mutation operations without much exploitation of the internal structure of good solution strings. This may not only generate large number of inefficient solutions dissipated over the solution space but also consuming long CPU time. In contrast, COIN considers the internal structure of good solution strings and memorises paths that could lead to good solutions. COIN replaces crossover and mutation operations of GA and employs joint probability matrix as a means to generate solutions. It prioritises the selection of the paths with higher chances of moving towards good solutions.

Apart from traditional learning from good solutions,

COIN allows learning from below average solutions as well. Any coincidence found in a situation can be statistically described whether the situation is good or bad. Most traditional algorithms always discard the bad solutions without utilising any information associated with them. In contrast, COIN learns from the coincidence found in the bad solutions and uses this information to avoid such situations to recurrent; meanwhile, experiences from good coincidences are also used to construct better solutions (**Figure 3**). Consequently, the chances that the paths always being parts of the bad solutions are used in the new generations are lessened. This lowers the number of solutions to be considered and hence increases the convergence speed.

COIN uses a join probability matrix (generator) to create the population. The generator is initialised so that it can generate a random tree with equal probability for any configuration. The population is evaluated in the same way as traditional EAs. However, COIN uses both good and bad solutions to update the generator. Initially, COIN searches from a fully connected tree and then incrementally strengthening or weakening the connections. As generations pass by, the probabilities of selection certain paths are increased or decreased depending on the incidences found in the good or bad solutions. The algorithm of COIN can be stated as follows.

1) Initialise the joint probability matrix (generator).
2) Generate the population using the generator.
3) Evaluate the population.
4) Diversity preservation.
5) Select the candidates according to two options: (a) good solution selection (select the solutions in the first rank of the current Pareto frontier), and (b) bad solution selection (select the solutions in the last rank of the current Pareto frontier).
6) For each joint probability matrix $H(x_i/x_j)$, adjust the generator according to the reward and punishment scheme as (4).

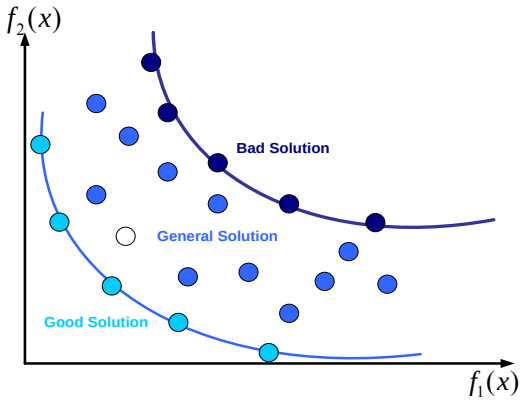


Figure 3. Good and bad solutions

$$x_{i,j}(t+1) = x_{i,j}(t) + \frac{k}{(n-1-np_i)}\{r_{i,j}(t+1) - p_{i,j}(t+1)\} + \frac{k}{(n-1-np_i)^2}\left\{\sum_{j=1}^{n} p_{i,j}(t+1) - \sum_{j=1}^{n} r_{i,j}(t+1)\right\} \quad (8)$$

where $x_{i,j}$ = the element (i,j) of joint probability matrix $H(x_i/x_j)$, k = the learning coefficient, $r_{i,j}$ = the number of coincidences (x_i, x_j) found in the good solutions, $p_{i,j}$ = the number of coincidences (x_i, x_j) found in the bad solutions, t = generation number, n = the size of the problem, and np_i = number of the direct predecessors of task i.

7) Apply a strategy to maintain elitist solutions in the population, and then repeat Step 2 until the terminating condition is met.

4.2 Numerical Example

The 11-task problem originated by Jackson [49] and later extended to accommodate a product mix by Hwang and Katayama [31] is used to elaborate the algorithm of COIN. Three models (A, B, and C) of the product mix with an equal minimum part set (MPS = [1,1,1]) are produced on MUL with 10-minute cycle time. Their precedence diagrams are shown in **Figure 4.**

Joint Probability Matrix Initialization

The number of tasks to be considered is 11. Therefore, the dimension of from-to joint probability matrix $H(x_i/x_j)$ is (11 × 11). The value of each element $(x_{i,j})$ in the matrix is the probability of selecting product j after product i. In order to incorporate some precedence relationship into the matrix, in each row, the element which belongs to the direct predecessor of the task is set to 0 to prohibit producing such task before its direct predecessor. For example, the direct predecessor of task 2 is task 1; hence, $x_{2,1} = 0$. Also, $x_{2,2} = 0$, since it can not move within itself. Initially, the value of the remaining elements in the 2nd row of the matrix is equal to $1/(n-1-np_2) = 1/(11-1-1) = 0.111$. Continue this computation for all the remaining tasks (rows), the initial joint probability matrix is shown in **Table 1.**

Population Generation

The order representation scheme is used to create chromosomes. The task order list in a chromosome is created by moving forward through MPD. If there is more than one task can be selected, the probability of selecting any task will depend on its value on the join probability matrix. For example, task 1 is selected for the first position since it is the only task to be considered. After selecting task 1, the set of eligible tasks comprises tasks 2, 3, 4 and 5. From row 1 of the joint probability matrix, a job is randomly selected according to its probability of selection ($p_{1,i} = 0.1000$, for $i = 2, \ldots, 11$). If the selected job is not in the set of eligible tasks, redo the

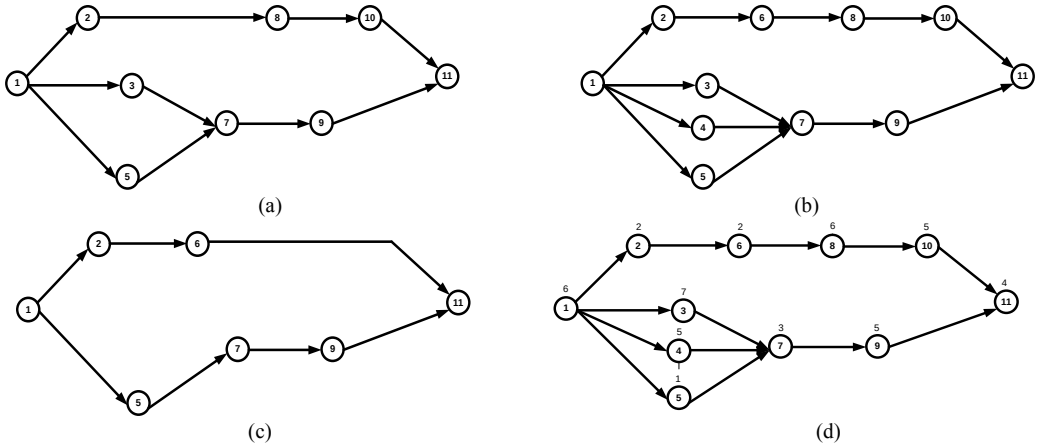


Figure 4. Precedence diagrams. (a) Precedence diagram of model A; (b) Precedence diagram of model B; (c) Precedence diagram of model C; (d) Merged precedence diagram of models A, B and C

Table 1. Initial joint probability matrix

i \ j	1	2	3	4	5	6	7	8	9	10	11
1	0	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
2	0	0	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
3	0	0.1111	0	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
4	0	0.1111	0.1111	0	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
5	0	0.1111	0.1111	0.1111	0	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
6	0.1111	0	0.1111	0.1111	0.1111	0	0.1111	0.1111	0.1111	0.1111	0.1111
7	0.1429	0.1429	0	0	0	0.1429	0	0.1429	0.1429	0.1429	0.1429
8	0.1111	0.1111	0.1111	0.1111	0.1111	0	0.1111	0	0.1111	0.1111	0.1111
9	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0	0.1111	0	0.1111	0.1111
10	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0	0.1111	0	0.1111
11	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0	0	0

selection. Suppose we select task 5, the new set of eligible tasks becomes tasks 2, 3 and 4. Continue this mechanism until all positions in the task order list are filled and we obtain the task order list of L_1={1,5,3,4,7,2,6,9,8,10,11}. Assume that the population size is 5 and the four remaining initial population consists of chromosomes L_2={1,4,5,3,7,9,2,6,8,10,11}; L_3={1,3,2,6,8,5,10,4,7,9,11}; L_4={1,4,3,2,6,8,10,5,7,9,11}, and L_5={1,5,4,3,2,6,8,7,9,10,11}.

Population Evaluation

To find tentative tasks to be allocated on the U-line, we have to search through the task order list in both forward and backward directions. The tentative task on forward or backward searching is the first found task that has its task time less than or equal to the remaining workstation cycle time and does not violated MPR. If both forward and backward tentative tasks are found, either one is selected randomly. But if none is found and the task order list still has some task not yet being allocated, a new workstation is opened. For example, for the task order list of L_1={1,5,3,4,7,2,6,9,8,10,11} and cycle time c = 10, the forward and backward tentative tasks are tasks 1 and 11. If task 1 is randomly selected, the remaining cycle time is 10 – 6 = 4, the new forward and backward tentative tasks are tasks 5 and 11 so on and so forth. Finally, a feasible line balance with m = 7 workstations and workstation load distribution given by S_1= {1,5}, S_2= {10,11}, S_3= {8}, S_4= {9}, S_5= {3}, S_6= {4,6,7}, S_7= {2}. Repeat this procedure for the remaining task order lists to obtain the number of workstations and workstation load distribution for each of them. Having obtained feasible line balances, three objectives have to be evaluated for each chromosome. **Table 2** indicates that all chromosomes give the same number of workstations; therefore, they are all eligible for Pareto ranking based on workload smoothness and work relatedness objectives. The Pareto ranking technique proposed by Goldberg [50] is used to classify the population into non-dominated frontiers and a dummy fitness value (lower value is better) is assigned to each chromosome (**Figure 5**).

Diversity Preservation

COIN employs a crowding distance approach [46] to generate a diversified population with uniformly spread over the Pareto frontier and avoid a genetic drift phenomenon (a few clusters of populations being formed in the solution space). The salient characteristic of this approach is that there is no need to define any parameter in calculating a measure of population density around a solution. The crowding distances computed for all solutions are infinite since only one solution is found for each frontier.

Solution Selection

Having defined the Pareto frontier, the good solutions are the chromosomes located on the first Pareto frontier (dummy fitness = 1), *i.e.* L_2={1,4,5,3,7,9, 2 ,6,8,10,11}. The bad solutions are those located on the last Pareto frontier (dummy fitness = 5), *i.e.* L_4={1,4,3,2,6,8,10,5,7,9,11}.

Joint Probability Matrix Adjustment

The adjustment of joint probability matrix is crucial to the performance of COIN. Reward will be given to $x_{i,j}$ if the order pair (i,j) is in the good solution to increase the chance of selection in the next round. For example, an order pair (1,4) is in the good solution L_2={1,4,5,3,7,9,2,6, 8,10,11}. Assume that k = 0.3, hence the value of $x_{i,j}$ where i = 1 and j = 4 is increased by $k/(n-1-np_1)$ = 0.3/(11 – 1 – 0) = 0.03. The updated value of $x_{i,j}$ of the order pair (1,4) becomes 0.1 + 0.03 = 0.13. The values of the other order pairs located in the same row of the order pair (1,4) is reduced by $k/(n-1-np_1)^2$ = 0.3/100 = 0.003. For example, the value $x_{i,j}$ where i = 1 and j = 4 is 0.1 – 0.003 = 0.0970. Continue this procedure to all order pairs located in the good solution; the revised joint probability matrix is obtained (**Table 3**).

In contrast, if the order pair (i,j) is in the bad solution, $x_{i,j}$ will be penalised to reduce the chance of selection in the next round. For example, an order pair (1,4) is in the bad solution L_4={1,4,3,2,6,8, 10,5,7,9,11}. Therefore, the value of $x_{i,j}$ where i = 1 and j = 4 is decreased by $k/(n-1-np_1)$ = 0.3/10 = 0.03. The updated value of $x_{i,j}$ of the order pair (1,4) becomes 0.130 – 0.030 = 0.100. The values of the other order pairs located in the same row of the order pair (1,4) is increased by $k/(n-1-np_1)^2$ = 0.3/100 = 0.003. For example, the value $x_{i,j}$ where i = 1 and j = 2 is 0.097 + 0.003 = 0.100. Continue this procedure to all order pairs located in the bad solution; the revised joint probability matrix is obtained (**Table 4**).

Elitism

To keep the best solutions found so far to be survived in the next generation, COIN uses an external list with the same size as the population size to store elitist solutions. All non-dominated solutions created in the current population are combined with the current elitist solutions. Goldberg's Pareto ranking technique is used to classify the combined population into several non-dominated frontiers. Only the solutions in the first non-dominated frontier are filled in the new elitist list. If the number of solutions in the first non-dominated frontier is less than or equal to the size of the elitist list, the new elitist list will contain all solutions of the first non-dominated frontier. Otherwise, Pareto domination tournament selection [51] is exercised. Two solutions from the first non-dominated solutions are randomly selected and then the solution with larger crowding distance measure and not being selected before is added to the new elitist list. This approach not only ensures that all solutions in the elitist list are non-dominated solutions but also promoting diversity of the solutions. According to our example, the current elitist list is empty and the solutions in the current first non-dominated frontier is L_2={1,4,5,3,7,9,2,6,8,10, 11}. When both sets are combined the non-dominated frontier is still the same. Also, the number of the combined solutions is less than the size of the elitist list; hence, both solutions are added to the new elitist.

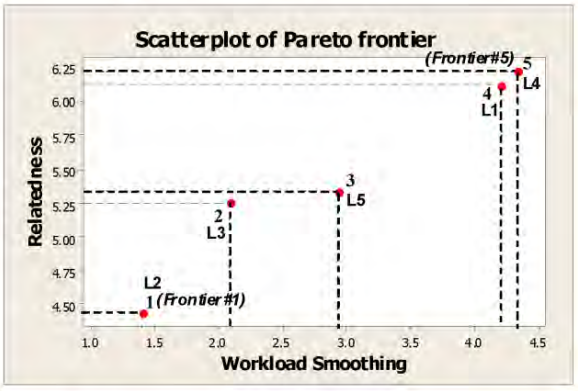


Figure 5. Pareto frontier of each chromosome

Table 2. Objective functions of each chromosome

Chromosome Number	Number of Workstations	Workload Smoothness	Work Relatedness	Pareto Frontier	Crowding Distance
2	4	1.4142	4.4444	1	Infinite
3	4	2.0817	5.2500	2	Infinite
5	4	2.9439	5.3333	3	Infinite
1	4	4.2088	6.1250	4	Infinite
4	4	4.3425	6.2222	5	Infinite

Table 3. Revised joint probability matrix (good solution)

i \ j	1	2	3	4	5	6	7	8	9	10	11
1	0	0.0970	0.0970	**0.1300**	0.0970	0.0970	0.0970	0.0970	0.0970	0.0970	0.0970
2	0	0	0.1074	0.1074	0.1074	**0.1444**	0.1074	0.1074	0.1074	0.1074	0.1074
3	0	0.1074	0	0.1074	0.1074	0.1074	**0.1444**	0.1074	0.1074	0.1074	0.1074
4	0	0.1074	0.1074	0	**0.1444**	0.1074	0.1074	0.1074	0.1074	0.1074	0.1074
5	0	0.1074	**0.1444**	0.1074	0	0.1074	0.1074	0.1074	0.1074	0.1074	0.1074
6	0.1074	0	0.1074	0.1074	0.1074	0	0.1074	**0.1444**	0.1074	0.1074	0.1074
7	0.1367	0.1367	0	0	0	0.1367	0	0.1367	**0.1858**	0.1367	0.1367
8	0.1074	0.1074	0.1074	0.1074	0.1074	0	0.1074	0	0.1074	**0.1444**	0.1074
9	0.1074	**0.1444**	0.1074	0.1074	0.1074	0.1074	0	0.1074	0	0.1074	0.1074
10	0.1074	0.1074	0.1074	0.1074	0.1074	0.1074	0.1074	0	0.1074	0	**0.1444**
11	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0	0	0

Table 4. Revised joint probability matrix (bad solution)

i \ j	1	2	3	4	5	6	7	8	9	10	11
1	0	0.1000	0.1000	**0.1000**	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
2	0	0	0.1111	0.1111	0.1111	**0.1111**	0.1111	0.1111	0.1111	0.1111	0.1111
3	0	**0.0741**	0	0.1111	0.1111	0.1111	0.1481	0.1111	0.1111	0.1111	0.1111
4	0	0.1111	**0.0741**	0	0.1481	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
5	0	0.1111	0.1444	0.1111	0	0.1111	**0.0778**	0.1111	0.1111	0.1111	0.1111
6	0.1111	0	0.1111	0.1111	0.1111	0	0.1111	**0.1111**	0.1111	0.1111	0.1111
7	0.1429	0.1429	0	0	0	0.1429	0	0.1429	**0.1429**	0.1429	0.1429
8	0.1111	0.1111	0.1111	0.1111	0.1111	0	0.1111	0	0.1111	**0.1111**	0.1111
9	0.1111	0.1481	0.1111	0.1111	0.1111	0.1111	0	0.1111	0	0.1111	**0.0741**
10	0.1111	0.1111	0.1111	0.1111	**0.0741**	0.1111	0.1111	0	0.1111	0	0.1481
11	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0	0	0

5. Experimental Design

5.1 Problem Sets

In order to compare the performances of COIN against several comparator search heuristics, three well-known test problems were employed as shown in **Table 5**. The problem set 1, 2, and 3 represent small, medium, and large-sized problems respectively

5.2 Comparison Heuristics

The performances of the proposed COIN applied to MULB problems are compared against such a well-known multi-objective evolutionary as NSGA II. In addition, the extended versions of COIN and NSGA II, *i.e.* MNSGA II and COIN-MA, are also evaluated.

Table 5. Test problems

	Problem Set	Number of Products	Number of Tasks	Cycle Time (sec)
1.	Thomopoulos[56]	3	19	120
2.	Kim[36]	4	61	600
3.	Arcus[57]	5	111	10,000

NSGA II

The algorithm of NSGA II [46] can be stated as follows.

1) Create an initial parent population of size N randomly.

2) Sort the population into several frontiers based on the fast non-dominated sorting algorithm.

3) Calculate a crowding distance measure for each solution.

4) Select the parent population into a mating pool based on the binary crowded tournament selection.

5) Apply crossover and mutation operators to create an offspring population of size N.

6) Combine the parent population with the offspring population and apply an elitist mechanism to the combined population of size $2N$ obtain a new population of size N.

7) Repeat Step 2 until the terminating condition is met.

MNSGA II

MNSGA II is a memetic version of NSGA II. Appro-

priate local searches can additionally embed into several positions of the NSGA II's algorithm, *i.e.* after initial population, after crossover, and after mutation [52]. The number of places to apply local search has a direct effect on the quality of solution and computation time. Hence, if computation time needs to be saved, local search should be taken only at some specific steps in the algorithm of MA rather than at all possible steps. In this research, we choose to take local search after obtaining initial solution and after mutation since pilot experiments and our previous research [53] indicated that these two points were enough to find significantly improved solutions, pull the solutions out of the local optimal, and reduce computational time. The algorithm of MNSGA II can be stated as follows.

1) Create an initial parent population of size N randomly.

2) Apply a local search to the initial parent population.

3) Sort the population into several frontiers based on the fast non-dominated sorting algorithm.

4) Calculate a crowding distance measure for each solution.

5) Select the parent population into a mating pool based on the binary crowded tournament selection.

6) Apply crossover and mutation operators to create an offspring population of size N.

7) Apply a local search to the offspring population.

8) Combine the parent population with the offspring population and apply an elitist mechanism to the combined population of size $2N$ obtain a new population of size N.

9) Repeat Step 3 until the terminating condition is met.

Four local searches modified from Kumar and Singh [54] originally developed to solve travelling salesman problems by repeatedly exchanging edges of the tour until no improvement is attained are examined including Pairwise Interchange (PI), Insertion Procedures (IP), 2-Opt, and 3-Opt. Three criteria are used to test whether to accept a move that a local search heuristic creates a neighbour solution from the current solution as follows: (1) accept the new solution if $f_1(x)$ is descendent, (2) accept the new solution if $f_1(x)$ is the same and $f_2(x)$ is descendent; (2) accept the new solution if $f_1(x)$ is the same and $f_3(x)$ is descendent; or (3) accept the new solution if it dominates the current solution ($f_1(x)$ is the same, and both $f_2(x)$ and $f_3(x)$ are descendent).

CNSGA II

In this heuristic, COIN is run for a certain number of generations. NSGA II then accepts the final solutions of COIN as its initial population and proceeds with its algorithm.

COIN-MA

In this heuristic, COIN is activated first for a certain number of generations. The final solutions obtained from COIN are then fed into MNSGA II as an initial population.

5.3 Comparison Metrics

Three metrics are measured to evaluate the achievement of two common goals for comparison of multi-objective optimisation methods as recommended by Kumar and Singh [54]: 1) convergence to the Pareto-optimal set, and 2) maintenance of diversity in the solutions of Pareto-optimal set. In addition, CPU time of each heuristic for achieving the final solutions is measured.

The convergence of the obtained Pareto-optimal solution towards a true Pareto-set (A^*) is the difference between the obtained solution set and the true-Pareto set. Mathematically, it is defined as (9) and (10)

$$convergence\ (A) = \frac{\sum_{i=1}^{|A^*|} dt_i}{|A^*|} \tag{9}$$

$$dt_i = min_{j=1}^{|A^*|} \sqrt{\sum_{k=1}^{2} \left[\frac{f_k(x) - f_k(y)}{f_k^{max} - f_k^{min}}\right]^2} \tag{10}$$

where $|A^*|$ is the number of elements in set A, dt_i is the Euclidean distance between non-dominated solution i^{th} in the true-Pareto frontier (y) and the obtained solution (x), f_k^{max} and f_k^{min} are maximum and minimum values of k^{th} objective functions in the true-Pareto set respectively. For metric A, lower value indicates superiority of the solution set. When all solutions converge to Pareto-optimal frontier, this metric is zero indicating that the obtained solution set has all solutions in the true Pareto set. Since the true Pareto frontier is unknown, its approximation is needed. The approximated true Pareto-optimal frontier is the result of combining all final non-dominated solutions obtained from of all algorithms, applying Goldberg's Pareto ranking technique to the combined solutions, and the first frontier of the combined solutions is the approximated true Pareto-optimal frontier.

The second measure is a spread metric. This measure computes the distribution of the obtained Pareto-solutions by calculating a relative distance between consecutive solutions as shown in (11) and (12).

$$spread\ (A) = \frac{sd_f + sd_l + \sum_{i=1}^{|A|-1} \|sd_i - s\bar{d}\|}{sd_f + sd_i + (|A|-1)s\bar{d}} \tag{11}$$

$$sd_i = \sqrt{\sum_{k=1}^{2} \left[\frac{f_k(x_i) - f_k(x_{i+1})}{f_k^{max} - f_k^{min}}\right]^2} \tag{12}$$

where sd_f and sd_l are the Euclidean distances between the extreme solutions and boundary solutions of the obtained Pareto-optimal, $|A|$ is the number of elements in the obtained-Pareto solutions, sd_i is the Euclidian distance of between consecutive solutions in the obtained-Pareto solutions for $i = 1, 2, \ldots, |A| - 1$,

$s\bar{d}$ is the average Euclidean distance of sd_i, and the operator "|| ||" means an absolute value. The value of this measure is zero for a uniform distribution, but it can be more than 1 when bad distribution is found.

The third measure is the ratio of non-dominated solutions $R_{NDS}(A_j)$ which indicates the coverage of one set over another. Let A_j be a solution sets $(j = 1, 2, \ldots, J)$. For comparing each J solution set $(A = A_1 \cup A_2 \ldots \cup A_J)$ the ratio of non-dominated measure of the solution set A_j to the J solution sets is the ratio of solutions in A_j that are not dominated by any other solution in A, which is defined as (13), where $y \prec x$ means the obtained solution x is dominated by the true-Pareto solution y. The higher ratio indicates superiority of one solution set over another.

$$R_{NDS}(A_j) = \frac{|A_j - \{x \in A_j \mid \exists y \in A : y \prec x\}|}{|A_j|} \quad (13)$$

All algorithms are coded in Matlab 7.0. The test platform is on Intel Core2 Duo 2.00 GHz under Windows XP with 1.99 GB RAM. The CPU time of each heuristic is kept after the program is terminated.

5.4 Parameter Settings

To tune MOEA for the MULB problems, an experimental design [55] was employed to systematically conduct and investigate the effect of each parameter to the responses of each heuristic. Recommendations from the past researches, e.g. Kim *et al.* [36], Chutima and Pinkoompee [53], etc. were used as a starting point for parameter settings. Extensive pilot runs were conducted around the vicinities of the starting point. The selection for each parameter setting was based on quality and diversity of solutions. If neither quality nor diversity of solutions was significantly different for several settings of the parameter, the one with lowest CPU time was selected. Having done that, **Table 6** shows the parameter settings found to be effective for each problem.

The process of finding appropriate local searches (LSs) for MNSGA II and COIN-MA for each problem set is worth mentioning. Four local searches that gave good performances from previous research [53] were investigated, *i.e.* Pairwise Interchange (PI), Insertion Procedures (IP), 2-Opt, and 3-Opt. Although LS can be located on 3 different places in MA, pilot runs indicated that putting LS after crossover did not help MA improve its performances. Therefore, LSs were placed only after initial population and after mutation for MNSGA II and after mutation for COIN-MA. Full factorial experiments were conducted to test the performances of LSs on each problem with 2 replicates. The number of experiment runs for each problem of MNSGA II and COIN-MA is 4*4*2 = 32 and 4*2 = 8 respectively. In total the number of runs is 120. ANOVA and Tukey's multiple range test were conducted to test significant different at 0.05 level.

Table 6. Parameter settings for each heuristic

Parameter settings	COIN	NSGA II	MNSGA II	CNSGA II	COIN-MA
Population size	100	100	100	100	100
Number of generations	Small = 100 Medium = 150 Large = 300	Small = 100 Medium = 150 Large = 300	Small = 100 Medium = 150 Large = 300	Small = 100 Medium = 150 Large = 300	Small = 100 Medium = 150 Large = 300
Crossover	-	Weight mapping crossover	Weight mapping crossover	Weight mapping crossover	Weight mapping crossover
Mutation	-	Reciprocal exchange	Reciprocal exchange	Reciprocal exchange	Reciprocal exchange
Probability of crossover	-	0.7	0.7	0.7	0.7
Probability of mutation	-	0.1	0.1	0.1	0.1
Learning coefficient (k)	Small = 0.1 Medium = 0.2 Large = 0.2	-	-	Small = 0.1 Medium = 0.2 Large = 0.2	Small = 0.1 Medium = 0.2 Large = 0.2
Percentage of generations of COIN to NSGA II	-	-	-	Small = 80:20 Medium = 60:40 Large = 60:40	Small = 80:20 Medium = 60:40 Large = 60:40

Table 7. Appropriate local searches

Problem set	MNSGA II		COIN-MA
	LS after initial population	*LS after mutation*	*LS after mutation*
1	IP	PI	IP
2	PI	3-Opt	IP
3	IP	PI	PI

The LSs appearing in **Table 7** were those that performed best with respect to solution quality, diversity, and CPU time. It is apparent that best LS combination for MNSGA II and COIN-MA depends on the problem set. However, for MNSGA II the combination of IP (LS used after initial population) and PI (LS used after mutation) appear more often than the other. For COIN-MA, IP seems to give better performances for small- and medium-sized problems; whereas, PI performed better than the others for large-sized problems. As a result, these settings were used for MNSGA II and COIN-MA in relative performance comparison.

6. Experimental Results

The behaviour of COIN was demonstrated with the 61 tasks' problem as shown in **Figure 6**. At the beginning (generation 1), a number of rather poor feasible solutions were created. As the number of generations increased, better solutions were found as observed from the moving downward trend to the left of the Pareto fronts. It was noticeable that not much improvement was gained in the first 20 generations. A leaped gain was noticeable from generations 20 to 30. However, the improvement was less and less after that and the Pareto front remained the same after generation 100.

The behaviour of CNSGA II (COIN plus NSGA II) and COIN-MA (COIN plus MNSGA II) were demonstrated in **Figure 7** and **Figure 8**. Both algorithms allowed COIN to run for 150 generations and the final solutions of COIN were considered as initial solutions of NSGA II and MNSGA II. Significant improvement was found after COIN was terminated and marginal gains from its previous solutions were obtained at the end of both algorithms. In other words, NSGA II (in CNSGA II) and MNSGA II (in COIN-MA) cannot provide much improvement to the final solutions of COIN.

For the small-sized problem (19 tasks), Table 8 showed that all algorithms gave the same number of workstations. NSGA II performed worst comparing with the others. By adding appropriate local search to NSGA II, its memetic version (MNSGA II) gained significant performance improvement. Although MNSGA II obtained the best spread metric, it was dominated by COIN, CNSGA II, and COIN-MA (**Figure 9**). These three algorithms obtained the same best Pareto front which can be seen from their ratio of non-dominated solution (R_{NDS} = 1) in **Table 8.**

For the medium-sized problem (61 tasks), COIN-MA obtained highest R_{NDS}, followed by CNSGA II, whereas the other algorithms have R_{NDS} = 0 (**Table 8**). This was confirmed by **Figure 10** meaning that some solutions of COIN-MA and CNSGA II were located in the Pareto front. MNSGA II outperformed NSGA II, but it was dominated by COIN. A big gap between the front of NSGA II and the fronts of three good performers (COIN, CNSGA II, and COIN-MA) was noticed indicating significant gains from using these three algorithms.

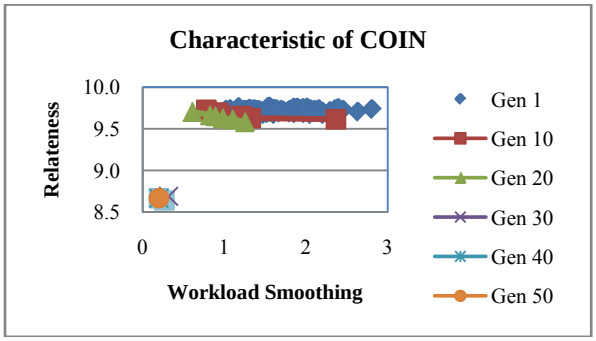


Figure 6. Characteristic of COIN

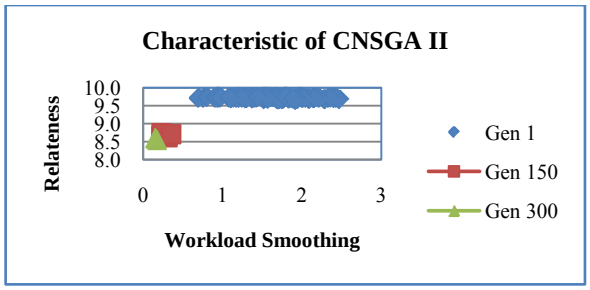


Figure 7. Characteristic of CNSGA II.

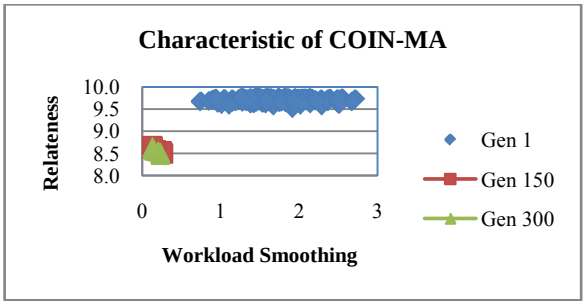


Figure 8. Characteristic of COIN-MA

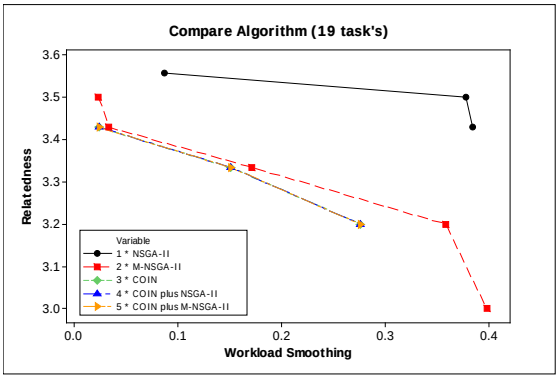


Figure 9. Pareto front of each algorithm (19 tasks)

Table 8. Performance comparison

Problem set	Performance Measure	NSGA II	COIN	MNSGA II	CNSGA II	COIN-MA
1	Number of workstations	4	4	4	4	4
	Convergence	0.4381	0.1317	0.0603	0.1317	0.1317
	Spread	0.6557	0.5390	0.4948	0.5390	0.5390
	R_{NDS}	0.0000	1.0000	0.4000	1.0000	1.0000
	CPU Time (min)	6	3	13	4	7
2	Number of workstations	10	9	10	9	9
	Convergence	0.9951	0.8966	0.4419	0.3058	0.0710
	Spread	0.4504	0.4945	0.8038	0.5514	0.4271
	R_{NDS}	0.0000	0.0000	0.0000	0.5000	0.6250
	CPU Time (min)	55	11	86	19	25
3	Number of workstations	16	16	16	15	15
	Convergence	1.0000	0.9907	0.8645	0.4100	0.0000
	Spread	0.7479	0.6951	0.4882	0.7211	0.6643
	R_{NDS}	0.0000	0.0000	0.0000	0.0000	1.0000
	CPU Time (min)	478	20	1089	32	44

For the large-sized problem (111 tasks), once again, COIN-MA performed best and NSGA II was ranked last (**Figure 11**). COIN outperformed NSGA and NSGA II. The performance of COIN was improved significantly with the cooperation of NSGA II (CNSGA II) and MNSGA II (COIN-MA). COIN-MA dominated all algorithms and, from **Table 8**, its solutions were all in the Pareto front ($R_{NDS} = 1$).

In terms of CPU time (**Table 8**), COIN used the lowest time to achieve the final solutions followed by CNSGA II, COIN-MA, NSGA II, and MNSGA II. As a result, COIN can be considered as a fast and smart algorithm since it can obtain good solutions very fast. It can be used as a good benchmark for other algorithms. In addition, if the good Pareto front needs to be discovered within a limited CPU time, COIN-MA is recommended as an outstanding alternative.

7. Conclusions

This paper presents a novel evolutionary algorithm

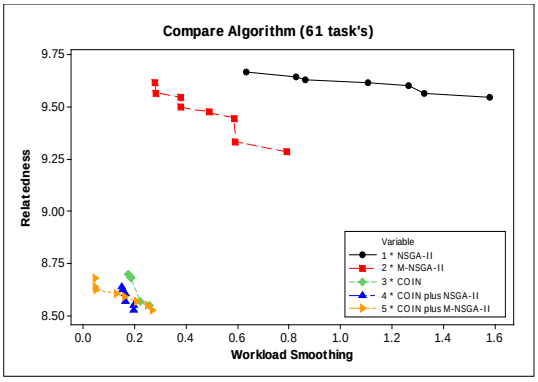


Figure 10. Pareto front of each algorithm (61 tasks)

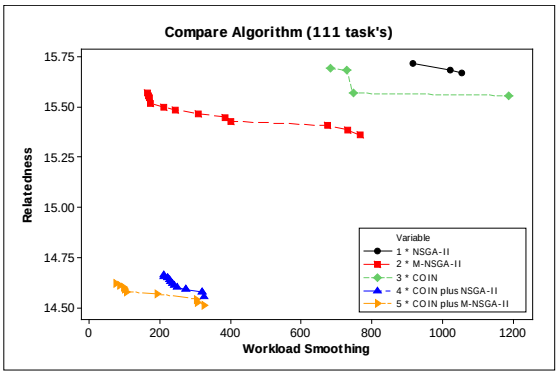


Figure 11. Pareto front of each algorithm (111 tasks)

namely combinatorial optimisation with coincidence algorithm (COIN) and its variances. The algorithms are applied to solve Type I problems of MMUALBP in a just-in-time production environment. COIN recognises the positive knowledge appearing in the order pairs of the good solution by giving a marginal reward (increased probability) to its related element of the joint probability matrix. In contrast, the negative knowledge found in the order pairs of the bad solution, which is often remiss in most algorithms, is also utilised in COIN (reduced probability) to prevent undesired solutions coincidentally found in this generation to be recurring in the next generation. The performances of COIN and its variances are evaluated on three objectives, *i.e.* minimum number workstations, minimum work relatedness, and minimum workload smoothness. Among these three, minimum number of workstations is dominated resulting in only the solutions with the same minimum number of workstations being considered and can be located on the first Pareto front. Experimental results indicate clearly that COIN outperforms the well-known NSGA II in all aspects. As a result, COIN can be considered as a new alternative benchmarking algorithm for MMUALBP. The COIN's variances (CNSGA II and COIN-MA) show significantly better performances than COIN, NSGA II, and MNSGA II. Although COIN-MA marginally uses more CPU time than CNSGA II, the other performances of COIN-MA are better than CNSGA. As a result, if we need to find an algorithm to search for an optimal Pareto front for MMUALBP, COIN-MA is recommended.

REFERENCES

[1] N. Boysen, M. Fliedner and A. Scholl, "Assembly Line Balancing: Which Model to Use When?" *International Journal of Production Economics*, Vol. 111, No. 2, 2008, pp. 509-528.

[2] R. J. Schonberger, "Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity," Free Press, New York, 1982, pp. 140-141.

[3] J. Miltenburg, "U-Shaped Production Lines: A Review of Theory and Practice," *International Journal of Production Economics*, Vol. 70, No. 3, 2001, pp. 201-214.

[4] Y. Monden, "Toyota Production System," 2nd Edition, Industrial Engineering Press, Institute of Industrial Engineering, Norcross, 1993.

[5] G. J. Miltenburg and J. Wijngaard, "The U-Line Balancing Problem," *Management Science*, Vol. 40, No. 10, 1994, pp. 1378-1388.

[6] C. H. Cheng, G. J. Miltenburg and J. Motwani, "The Effect of Straight- and U-Shaped Lines on Quality," *IEEE Transactions on Engineering Management*, Vol. 47, No. 3, 2000, pp. 321-334.

[7] G. J. Miltenburg, "The Effect of Breakdowns on U-Shaped Production Lines," *International Journal of Production Research*, Vol. 38, No. 2, 2000, pp. 353-364.

[8] J. Miltenburg, "One-Piece Flow Manufacturing on U-Shaped Production Lines: A Tutorial," *IIE Transactions*, Vol. 33, No. 4, 2001, pp. 303-321.

[9] Y. Kara, U. Ozcan and A. Peker, "Balancing and Sequencing Mixed-Model just-in-Time U-Lines with Multiple Objectives," *Applied Mathematics and Computation*, Vol. 184, No. 2, 2007, pp. 566-588.

[10] M. E. Salveson, "The Assembly Line Balancing Problem," *The Journal of Industrial Engineering*, Vol. 6, No. 3, 1955, pp. 18-25.

[11] I. Baybars, "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem," *Management Science*, Vol. 32, No. 8, 1986, pp. 909-932.

[12] S. Ghosh and R. J. Gagnon, "A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems," *International Journal of Production Research*, Vol. 27, No. 4, 1989, pp. 637-670.

[13] E. Erel and S. C. Sarin, "A Survey of the Assembly Line Balancing Procedures," *Production Planning and Control*, Vol. 9, No. 5, 1998, pp. 414-434.

[14] C. Becker and A. Scholl, "A Survey on Problems and Methods in Generalized Assembly Line Balancing," *European Journal of Operational Research*, Vol. 168, No. 3, 2006, pp. 694-715.

[15] N. Boysen and M. Fliedner, "A Versatile Algorithm for Assembly Line Balancing," *European Journal of Operational Research*, Vol. 184, No. 1, 2008, pp. 39-56.

[16] D. Sparling and J. Miltenburg, "The Mixed-Model U-Line Balancing Problem," *International Journal of Production Research*, Vol. 36, No. 2, 1998, pp. 485-501.

[17] G. J. Miltenburg, "Balancing U-Lines in a Multiple U-Line Facility," *European Journal of Operational Research*, Vol. 109, No. 1, 1998, pp. 1-23.

[18] T. L. Urban, "Optimal Balancing of U-Shaped Assembly Lines," *Management Science*, Vol. 44, No. 5, 1998, pp. 738-741.

[19] D. A. Ajenblit and R. L. Wainwright, "Applying Genetic Algorithms to the U-Shaped Assembly Line Balancing Problem," *Proceedings of the* 1998 *IEEE International Conference on Evolutionary Computation*, Alaska, 1998, pp. 96-101.

[20] A. School and R. Klein, "ULINO: Optimally Balancing U-Shaped JIT Assembly Lines," *International Journal of Production Research*, Vol. 37, No. 4, 1999, pp. 721-736.

[21] E. Erel, I. Sabuncuoglu and B. A. Aksu, "Balancing of U-Type Assembly Systems Using Simulated Annealing," *International Journal of Production Research*, Vol. 39, No. 13, 2001, pp. 3003-3015.

[22] G. R. Aase, M. J. Schniederjans and J. R. Olson, "U-OPT: An Analysis of Exact U-Shaped Line Balancing Procedures," *International Journal of Production Research*, Vol. 41, No. 17, 2003, pp. 4185-4210.

[23] G. R. Aase, J. R. Olson and M. J. Schniederjans, "U-Shaped Assembly Line Layouts and their Impact on Labor Productivity: An Experimental Study," *European Journal of Operational Research*, Vol. 156, No. 3, 2004, pp. 698-711.

[24] U. Martinez and W. S. Duff, "Heuristic Approaches to Solve the U-Shaped Line Balancing Problem Augmented by Genetic Algorithms," *Proceedings of the* 2004 *Systems and Information Engineering Design Symposium*, Charlottesville, 16 April 2004, pp. 287-293.

[25] J. Balakrishnan, C.-H. Cheng, K.-C. Ho and K. K. Yang, "The Application of Single-Pass Heuristics for U-Lines," *Journal of Manufacturing Systems*, Vol. 28, No. 1, 2009, pp. 28-40.

[26] H. Gokcen and K. Agpak, "A Goal Programming Approach to Simple U-Line Balancing Problem," *European Journal of Operational Research*, Vol. 171, No. 2, 2006, pp. 577-585.

[27] T. L. Urban and W.-C. Chiang, "An Optimal Piecewise-Linear Program for the U-Line Balancing Problem with Stochastic Task Times," *European Journal of Operational Research*, Vol. 168, No. 3, 2006, pp. 771-782.

[28] W.-C. Chiang and T. L. Urban, "The Stochastic U-Line Balancing Problem: A Heuristic Procedure," *European Journal of Operational Research*, Vol. 175, No. 3, 2006, pp. 1767-1781.

[29] Y. Kara, T. Paksoy and C. T. Chang, "Binary Fuzzy Goal Programming Approach to Single Model Straight and U-Shaped Assembly Line Balancing," *European Journal of Operational Research*, Vol. 195, No. 2, 2009, pp. 335-347. A. L. Arcus, "COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines," *International Journal of Production Researc*h, Vol. 4, No. 4, 1965, pp. 259-277.

[30] A. Baykasoglu, "Multi-Rule Multi-Objective Simulated Annealing Algorithm for Straight and U Type Assembly Line Balancing Problems," *Journal of Intelligent Manufacturing*, Vol. 17, No. 2, 2006, pp. 217-232.

[31] R. K. Hwang, H. Katayama and M. Gen, "U-Shaped Assembly Line Balancing Problem with Genetic Algorithm," *International Journal of Production Research*, Vol. 46, No. 16, 2008, pp. 4637-4649.

[32] R. K. Hwang and H. Katayama, "A Multi-Decision Genetic Approach for Workload Balancing of Mixed-Model U-Shaped Assembly Line Systems," *International Journal of Production Research*, Vol. 47, No. 14, 2009, pp. 3797-3822.

[33] A. Baykasoglu and T. Dereli, "Simple and U-Type Assembly Line Balancing by Using an Ant Colony Based Algorithm," *Mathematical and Computational Applications*, Vol. 14, No. 1, 2009, pp. 1-12.

[34] G. J. Miltenburg, "Balancing and Scheduling Mixed-Model U-Shaped Production Lines," *International Journal of Flexible Manufacturing Systems*, Vol. 14, No. 2, 2002, pp. 119-151.

[35] Y. K. Kim, S. J. Kim and J. Y. Kim, "Balancing and Sequencing Mixed-Model U-Lines with a Co-Evolutionary Algorithm," *Production Planning and Control*, Vol. 11, No. 8, 2000, pp. 754-764.

[36] Y. K. Kim, J. Y. Kim and Y. Kim, "An Endosymbiotic Evolutionary Algorithm for the Integration of Balancing and Sequencing in Mixed-Model U-Lines," *European Journal of Operational Research*, Vol. 168, No. 3, 2006, pp. 838-852.

[37] S. Agrawal and M. K. Tiwari, "A Collaborative Ant Colony Algorithm to Stochastic Mixed-Model U-Shaped Disassembly Line Balancing and Sequencing Problem," *International Journal of Production Research*, Vol. 46, No. 6, 2008, pp. 1405-1429.

[38] I. Sabuncuoglu, E. Erel and A. Alp, "Ant Colony Optimization for the Single Model U-Type Assembly Line Balancing Problem," *International Journal of Production Economics*, Vol. 120, No. 2, 2009, pp. 287-300.

[39] Y. Kara, U. Ozcan and A. Peker, "An Approach for Balancing and Sequencing Mixed-Model JIT U-Lines," *International Journal of Advanced Manufacturing Technology*, Vol. 32, No. 11-12, 2007, pp. 1218-1231.

[40] Y. Kara, "Line Balancing and Model Sequencing to Reduce Work Overload in Mixed-Model U-Line Production Environments," *Engineering Optimization*, Vol. 40, No. 7, 2008, pp. 669-684.

[41] A. Konak, D. W. Coit and A. E. Smith, "Multi-Objective Optimization Using Genetic Algorithms: A Tutorial," *Reliability Engineering & System Safety*, Vol. 91, No. 9, 2006, pp. 992-1007.

[42] C. A. C. Coello, D. A. Veldhuizen and G. B. Lamont, "Evolutionary Algorithms for Solving Multi-Objective Problems," Kluwer Academic Publishers, Dordrecht, 2002.

[43] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, 1999, pp. 257-271.

[44] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Proceedings of 5th International Conference on Genetic Algorithm*, Urbana, June 1993, pp. 416-423.

[45] C. M. Fonseca and P. J. Fleming, "An overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, Vol. 3, No. 1, 1995, pp. 1-16.

[46] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182-197.

[47] D. Corne, M. Dorigo and F. Glover, "New Ideas in Optimization," McGraw-Hill, London, 1999.

[48] W. Wattanapornprom, P. Olanviwitchai, P. Chutima and P. Chongsatitvatana, "Multi-Objective Combinatorial Op-

timisation with Coincidence Algorithm," *Proceedings of IEEE Congress on Evolutionary Computation*, Norway, 11 February 2009, pp. 1675-1682.

[49] J. R. Jackson, "A Computing Procedure for a Line Balancing Problem," *Management Science*, Vol. 2, No. 3, 1956, pp. 261-271.

[50] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, Boston, 1989.

[51] J. Horn, N. Nafpliotis and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," *Proceedings of the First IEEE Conference on Evolutionary Computation*, *IEEE World Congress on Computational Intelligence*, Orlando, 27-29 June 1994.

[52] P. Lacomme, C. Prins and M. Sevaux, "A Genetic Algorithm for a Bi-Objective Capacitated ARC Routing Problem," *Computer & Operations Research*, Vol. 33, No. 12, 2006, pp. 3473-3493.

[53] P. Chutima and P. Pinkoompee, "An Investigation of Local Searches in Memetic Algorithms for Multi-Objective Sequencing Problems on Mixed-Model Assembly Lines," *Proceedings of Computers and Industrial Engineering*, Beijing, 31 October-2 November 2008.

[54] R. Kumar and P. K. Singh, "Pareto Evolutionary Algorithm Hybridized with Local Search for Bi-Objective TSP," *Studies in Computational Intelligence* (*Hybrid Evolutionary Algorithms*), Springer, Berlin/Heidelberg, Vol. 75, 2007, pp. 361-398.

[55] D. C. Montomery, "Design and Analysis of Experiments," John Wiley & Sons, Inc., Hoboken, 2009.

[56] N. T. Thomopoulos, "Mixed Model Line Balancing with Smoothed Station Assignment," *Management Science*, Vol. 14, No. 2, 1970, pp. B59-B75.

[57] A. L. Arcus, "COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines," *International Journal of Production Research*, Vol. 4, No. 4, 1965, pp. 259-277.

J. Software Engineering & Applications, 2010, 3: 364-373
doi:10.4236/jsea.2010.34041 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Study and Analysis of Defect Amplification Index in Technology Variant Business Application Development through Fault Injection Patterns

Paloli Mohammed Shareef[1], Midthe Vijayaraghavan Srinath[2], Subbiah Balasubramanian[3]

[1]Trimentus Technologies, Chennai, India; [2]Mahendra Engineering College, Namakkal, India; [3]Anna University, Coimbatore, India.
Email: pmshareef@gmail.com, {sri_induja, s_balasubramanian}@rediffmail.com

Received December 26th, 2009; revised January 14th, 2010; accepted January 26th, 2010.

ABSTRACT

Software reliability for business applications is becoming a topic of interest in the IT community. An effective method to validate and understand defect behaviour in a software application is Fault Injection. Fault injection involves the deliberate insertion of faults or errors into software in order to determine its response and to study its behaviour. Fault Injection Modeling has demonstrated to be an effective method for study and analysis of defect response, validating fault-tolerant systems, and understanding systems behaviour in the presence of injected faults. The objectives of this study are to measure and analyze defect leakage; Amplification Index (*AI*) *of errors and examine "Domino" effect of defects leaked into subsequent Software Development Life Cycle phases in a business application. The approach endeavour to demonstrate the phasewise impact of leaked defects, through causal analysis and quantitative analysis of defects leakage and amplification index patterns in system built using technology variants* (*C#, VB* 6.0, *Java*).

Keywords**:** *Fault Injection, Amplification Index* (*AI*), *Domino Effect, Defect Leakage*

1. Introduction

Formulating reliable and fault tolerant software is difficult and requires discipline both in specifying system functionality and in implementing systems correctly. Approaches for developing highly reliable software include the use of formal methods [1-3], and rigorous testing methods [4].

Testing cannot guarantee that commercial and business software is correct [5], and verification requires enormous human effort and is subject to errors [6]. Automated support is necessary to help ensure software correctness and fault tolerance.

Fault injection modelling involves the deliberate insertion of faults or errors into a computer system in order to determine its response. It has proven to be an effective method for measuring and studying response of defects, validating fault-tolerant systems, and observing how systems behave in the presence of faults. In this study, faults are injected in key phases of software development of business application following a typical water fall software life cycle viz., SRS, Design and Source code.

2. Literature Review

The literature review consolidates the understanding on fault injection, associated topics and subsequent studies to emphasis the need to fault injections in business software application. It also crystallizes the need for awareness, tools and analyzes defect leakage/amplification.

Even after 20 years of existence the awareness of fault injection and associated modelling with tools are very rarely used and understood in the commercial software industry and used. The usefulness in the defect modelling and building fault tolerant software systems are not properly preached and/or practiced. Added, the availability of appropriate literature and software tools is very few and not used in commercial and business application design and testing.

After a detailed review of literature by the researcher it was concluded that there is an industrious interest software fault injection in the software industry to develop commercially reliable software.

3. Approach

In recent years there has been much interest in the field of software reliability and fault tolerance of systems and commercial software. This in turn has resulted in a wealth of literature being published around the topic, such as the

Fault Injection in the form of the 'Marrying Software Fault Injection Technology Results with Software Reliability' by Jeffrey Voas, Cigital Norman Schneidewind.

Many critical business computer applications require "fault tolerance," the ability to recover from errors or exceptional conditions. Error free software is very difficult to create and creating fault tolerant software is an even greater challenge. Fault tolerant software must successfully recover from a multitude of error conditions to prevent harmful system failures.

Software testing cannot demonstrate that a software system is completely correct. An enormous number of possible executions that must be examined in any real-world sized software system. Fault tolerance expands the number of states (and thus execution histories) that must be tested, because inconsistent or erroneous states must also be tested.

Mailing lists, websites, research and forums have been created in which all aspects of this fresh new niche software engineering area are discussed. People are interested, partly because it is a new area but also because the whole field of commercial software reliability is in itself so interesting; as it holds so many wide ranging disciplines, perspectives and logic at its core. Software reliability engineering is uniting professionals in disciplines that previously had little to do with one another, it is creating more opportunities for employment in the online environment, and it is changing the face and structure of all information that we seek out on the web. In the era of economic recession, customer demands reliable, certified and fault tolerant commercial and business software applications.

In this research, the focus is on software testing techniques that use fault injection. Several potentially powerful existing systems have drawbacks for practical application in business application development environment. We first examine existing fault injection techniques and evaluate their potential for practical application for commercial and business software applications. Available and accessible literature infrastructure including premium subscribed IEEE and ACM resources were studied and summarized for literature review from 1986 (20 years).

4. Fault Injection Modelling

Fault Injection Modelling (FIM) involves the deliberate insertion of faults or errors into a computer system in order to determine its response. It has proven to be an effective method for measuring and studying response of defects, validating fault-tolerant systems, and observing how systems behave in the presence of faults. In this study, faults are injected in all phases of software development life cycle viz., Requirements, Design and Source code.

4.1 Objectives

The objectives of conducting these experiments are to measure process efficiencies, statistically study, analysis and report defect amplification of defects (Domino's effect) across software development phases with a similar system constructed with technological variation.

The goal of this research is to understand the behaviour of faults and defects pattern in commercial and business software application and defect leakage in each phase of application development.

Throughout the literature certain questions reoccur, which one would anticipate when a new field emerges in commercial software fault tolerance? People are interested, and want to understand and define commercial software reliability and fault tolerance since the work on most fault injections and software reliability is found in life critical and mission critical application, so we try to answer the following questions;

- Why study Fault Injection Modelling?
- Why study business software fault tolerance requirements?
- Why are they called 'Fault Injection & Error Seeding'?
- Why review Software Implemented Fault Injection (SWIFI)?
- What work was performed, current status and work proposed?

These questions will be expanded upon throughout the research, and seek to bring clarity to those who want to find the answers to the above, or to see if there truly are any answers!

4.2 Background Concepts

A fault is a hardware or software defect, inconsistency, transient electrical field, or other abnormal circumstance. An error is an invalid internal state, which may or may not be detected by the system.

A failure is an invalid output. Thus a fault or error becomes a failure when it propagates to the output. There is a natural progression from fault to error to failure. Recovery code is the part of a program that is designed to respond to error states. Recovery code executes after the program recognizes that some erroneous or abnormal state has been entered. This code should gracefully restore the system to a valid state before a failure occurs.

Figure 1 shows the progression from faults to errors and finally to failures. The recovery code should serve as a safety net to prevent the progression from error to failure. A fault tolerant system should never fail, even if it has faults.

Testing recovery code requires the modeling of bad states that accurately simulate exceptional situations. As much as 50% of a fault tolerant program can consist of recovery code. Although testing might include invalid

data that executes some of the recovery code, often much of this code is never executed during normal testing.

Any recovery code testing technique must be based upon an assumed fault model [7]. We assume that all faults will behave according to some specific rules. Any fault model can only consider a subset of all possible faults.

For example, a common debugging practice is to insert a series of print statements in key positions. This debugging practice assumes a particular fault model.

Faults will cause the program to execute in the incorrect order and will be demonstrated

Figure 2 illustrates the taxonomy of Fault Injection Techniques in the printed output. Clearly, not all faults will adhere to this model.

No one fault model will fit all faults. However, a fault model can be very effective in detecting faults that fit the model.

Fault Injection technique of fault injection dates back to the 1970s when it was first used to induce faults at a hardware level. This type of fault injection is called Hardware Implemented Fault Injection (HWIFI) and attempts to simulate hardware failures within a system. The first experiments in hardware fault injection involved nothing more than shorting connections on circuit boards and observing the effect on the system (bridging faults). It was used primarily as a test of the dependability of the hardware system. Later specialised hardware was developed to extend this technique, such as devices to bombard specific areas of a circuit board with heavy radiation. It was soon found that faults could be induced by software techniques and that aspects of this technique could be useful for assessing software systems. Collectively these techniques are known as Software Implemented Fault Injection (SWIFI) [8].

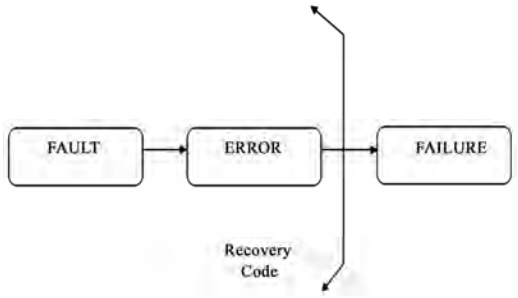


Figure 1. Fault tolerance terms

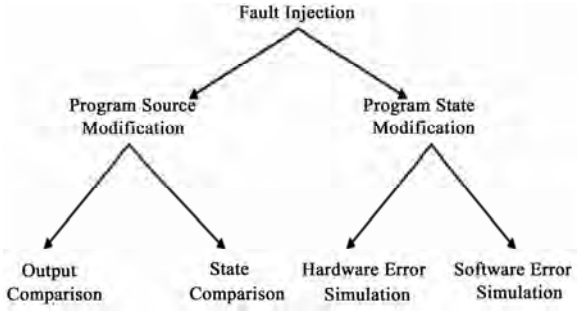


Figure 2. Taxonomy of fault injection techniques

Martin defines software fault injections as faults which are injected at the software level by corrupting code or data. So faults are applicable at the implementation phase when the code of the system is available, and it can be applied on an application to simulate either internal or external faults.

Internal faults represent design and implementation faults, such as variables/parameters that are wrong or not initialized, incorrect assignments or condition checks. External faults represent all external factors that are not related to faults in the code itself but that alter the system's state.

The injection of failures can discover errors that normal procedures cannot. First, it tests the mechanisms of exception and treatment of failures that in normal circumstances are not sufficiently proven and, helps to evaluate the risk, verifying how much defective can be the system behavior in presence of errors. All of the injection failures methods are based on concrete hardware or software characteristics associated to systems which are applied, then, to realize generalizations is a very complicated task.

4.3 Prior Work on Fault Injection

Fault injection can be used to modify either a program's source code text or the machine state of an executing program. **Figure 2** shows taxonomy of the key methods of fault injection. Fault injection techniques based on static analysis—program source modification—are modeled by the left subtree.

The most common static fault injection is mutation testing. The right subtree in **Figure 2** models dynamic fault injection techniques where changes are made to an actively running program's state. Much of the recent fault injection research is concerned with dynamic injection.

4.4 Domino's Effect

Domino's effect is the cascading effect of defects from the initial stages of the project to all the subsequent stages of the software life cycle. Errors undetected in one work product are 'leaked' to the child work product and amplifies defects in the child work product. This chain reaction causes an exponential defect leakage. E.g.: undetected errors in requirements leak and cause a significant number of defects in design which, in turn, causes more defects in the source code. The result of this study is to arrive at an "Amplification Index" which will characterize the extent of impact or damage of phase-wise defects in subsequent Software Development Life Cycle

(SDLC) phases.

The defect components in a work product and leakage into subsequent phases are illustrated in **Figure 3** below:

4.5 Trimentus Approach for Fault Injection Experiments

Defects were deliberately injected into each phase (work product) in the software development life cycle of a typical application development project and the effect of the defects injected was studied subsequently. The injected defects are typical defects that are characteristic of the software systems of a commercial application on Library Management System (LMS) and were chosen from the organizational defect database.

An approach was adopted towards studying the impact of defect amplification in a software system was causal analysis of the defects occurring in subsequent phases caused due to injected defects.

Fault injection can occur in several ways:

- Additional code can be linked to the target program and executed synchronously with the program flow.
- A separate process can perform the injection asynchronously with the flow of the target process.
- Separate hardware can directly access the memory to modify the state, thus not affecting the timing characteristics of the target process.

Overlay faults occur when a program writes into an incorrect location due to a faulty destination operand. Chillarege and Bowen claim that overlay faults account for 34% of the errors in systems programs. The experiment involved the use of failure acceleration, decreasing fault and error latency and increasing the probability that a fault will cause an error. The experiment applied failure acceleration by corrupting a large region of memory in a single injection. To inject an overlay fault, all bits in an entire page of physical memory are set to one. Because the page is in physical memory, the probability that the latency will be short is further increased. About 16% of the faults immediately crashed the system; about 14% caused a partial loss of service, which was usually recovered from soon after.

Half of the faults did not cause failures. These potential hazards are failures waiting to occur. The injection process used was manual and only 70 faults were injected during the entire experiment.

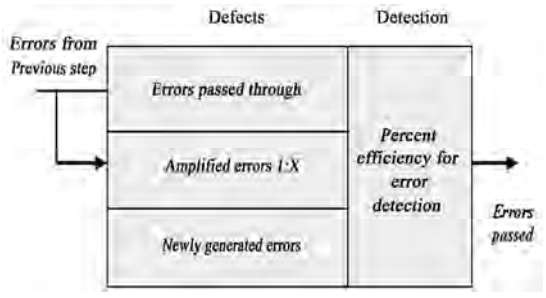


Figure 3. Fault injection pattern

Software faults introduced include:

- Initialization faults: incorrectly or uninitialized variables. They are modeled by dynamically replacing the initializing assembly instructions with incorrect values or no-ops.
- Assignment faults: incorrect assignment statements. Variable names on the right hand side are changed by dynamically mutating the assembly code.
- Condition check faults: missing condition checks, for example, failure to verify return values. Condition checks are either entirely overwritten with no-ops, or replaced an incorrect condition check.
- Function faults: Invalid functions. The assembly code for a function is dynamically replaced with the assembly code from a manually rewritten alternate version.

Initialization faults can be caught statically with a good compiler. The assignment and condition check faults are clearly relevant to the testing of recovery code, since an incorrect assignment or condition can be a condition that should force the execution of recovery code. Function faults are also relevant, especially if they could be automatically generated. Unfortunately, manual rewriting of sections of code is prohibitive in a large system.

4.6 Why Study Fault Injection Modelling?

Fault Injection Modelling has gradually crept into prominence over the last decade as one of the new buzz words in software design. However, as Martin observes:

"The main characteristic of fault injection software is that it is capable of injecting failures into any functional addressing unit by means of software, such as memory, registers, and peripherals. The goal of the fault injection software is to reproduce, in the logical scope, the errors that are reproduced after failures in the hardware. A good characterization of failure model should be allowed that this one was as versatile as possible, allowing a major number of combinations among the location, trigger conditions, kind of fault and duration, so that the coverage was maximum. Recent days, the Fault Injection technique has been considered as a very useful tool to monitor and evaluate the behavior of computing systems in the presence of faults. It's because the tool tries to produce or simulate faults during an execution of the system under test, and then the behavior of the system is detected."[9]

Figure 4 illustrates the relative cost factor in the defect resolution as the work product elaborates in the Software Development Life Cycle phases;

- The Carnegie Mellon Software Engineering Institute[1]

[1]Carnegie Mellon Software Engineering Institute, the Business Case for Requirements Engineering, RE' 2003, 12 September 2003.

reports that at least 42-50 percent of software defects originate in the requirements phase.

- The Defense Acquisition University Program Manager Magazine[2] reports that a Department of Defense study that over 50 percent of all software errors originate in the requirements phase.

1) MSDN (November, 2005) "Leveraging the Role of Testing and Quality across the Lifecycle to Cut Costs and Drive IT/Business Responsiveness".

2) Direct Return on Investment of Software Independent Verification and Validation: Methodology and Initial Case Studies, James B. Dabney and Gary Barber, Assurance Technology Symposium, 5 June 2003.

5. Description of Software System

A Library Management System (LMS) help in automating functions of the library. It helps in reducing the time spent in record keeping and management effectively. The management information system application was used to conduct the fault injection experiments. The same application was developed in the following technologies in 3G languages as listed in **Table 1** below.

LMS was simultaneously developed by independent project team and were made mutually exclusive. The application development for the projects followed the same process as described in the quality management system for software development of Trimentus. LMS was chosen to FIM because common MIS Domain knowledge for the application was high; it can be independently managed and developed; it covers the entire development life cycle; and the technology used is typical of current commercial applications and technologies in vogue.

SDLC, technology, exclusiveness allows different types of faults to be injected at various phases without bias and enables direct comparison.

In this paper, the system contains injected defects common across all projects. The same count of defects (5 numbers) were introduced in each phase of SDLC.

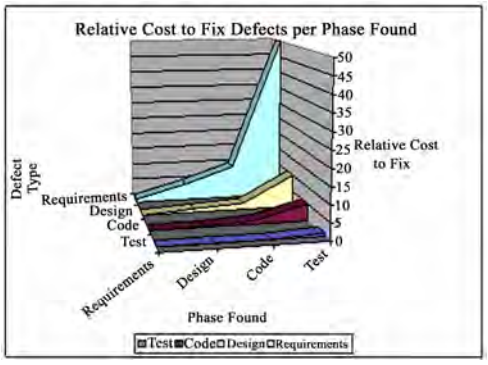


Figure 4. Relative cost to fix defects vs. development phases

Table 1. Library management system (LMS) experiment technology variants

Project Id	Programming Language	RAD Tool	Database
LMS 1	C#.Net	Visual Studio 2005	SQL Server 2005
LMS 2	Visual Basic 6.0	Visual Studio 6.0	Ms Access 2007
LMS 3	Java (jdk1.5)	NetBeans IDE 5.0	SQL Server 2005

[2]Defense Acquisition University Program Manager Magazine, Nov-Dec 1999, Curing the Software Requirements and Cost Estimating Blues

6. Results of the Experiments

The results from the independent experiments are derived at each stage of the Software Life Development Cycle Phase. The following section describes the detailed activities and step-by-step process followed in the introduction of defects in each software work product with results output.

6.1 Requirement Review

SRS (Software Requirement Specification) document was prepared and used as the basis for development of for all the experiments. SRS is identified as requirements documents. However, after the review of SRS, defects were injected into the same document. The SRS containing the defects were baselined by independent project team respectively to be used as basis for the Design.

The defects injected into the Requirement document are given in **Table 2.** The requirements defects are analyzed through causal analysis techniques to be classified and categorized.

6.2 Design Phase Analysis

Design document is prepared with (fault injected) SRS as basis. There were several defects observed with "source"

Table 2. Definition of defect types – requirements

Action taken	Defect Injected	Defect severity	Defect Type
Deleted	Reports based on classification by Type of books	High	Missed Requirement
Modified	Changed User Login to Student ID Changed the default status of the books given from "Pending" to "Borrowed" Add more records option not given as part of screen layout	Medium	Incomplete, Missed Requirement
Added	Obtaining the proposed date for return of books	High	Ambiguous
Deleted	Set the type of fine	High	Missed Requirement
Added	Set the number of times a books can be renewed by the members	Medium	Incorrect Requirement

as requirements. The Injected defects were major cause for design defects.

6.2.1 Design Review

Table 3 lists the number of defects injected independently in Requirements and inherent defects detected after Requirements document review. Further, it lists the defect leakage to the child work product (Design) with inherent defects detected after Design review for each experiment.

6.2.2 Design Defect Amplification: Technology Variant

Figure 5 represents the comparison of Amplification Index between the LMS developed on different technologies. The amplification of design defects caused due to the injected requirement defects in LMS is evidenced in all technologies and more prominent in VB Microsoft technology.

6.2.3 Amplification Index (AI) for Requirements

Table 3 and **Table 4** represent the methodology that was used to calculate Requirement Amplification Index (*i.e.* impact of requirements defects on Design).

6.2.4 Defects in Design

Table 5 lists the various types of known design defects that were introduced after design review. The defects are classified and categorized after causal analysis.

Table 3. Defects injected—requirements to design

	Source			
	SRS		Design	
	Injected	Inherent	Leaked	Inherent
LMS 1	5	4	4	8
LMS 2	5	8	7	6
LMS 3	5	5	7	9

Table 4. Defects amplification index computation—requirements to design

Application	Formula	AI (Requirements on Design)
LMS1	No. of design defects caused due to injected Requirement Defects / No. of injected Requirement defects	2/5 = 0.5 (rounded) ➔ *One requirement defect leaked causes 0.5 defect in design in C # technology*
LMS2	No. of design defects caused due to injected Requirement Defects / No. of injected Requirement defects	6/5 = 1.3 (rounded) ➔ *One requirement defect leaked causes 1.3 defect in design in VB technology*
LMS3	No. of design defects caused due to injected Requirement Defects / No. of injected Requirement defects	4/5 = 0.8 (rounded) ➔ *One requirement defect leaked causes 0.8 defect in design in Java technology*

Table 5. Definition of defect types – design

Action taken	Defect Injected	Defect severity	Defect Type
Removed	Validation and authentication of authorized students	High	Interface, Incomplete
Modified	Data Type Changed	Medium	Database, Incorrect
Review finding	Editing of book type by borrower	High	Incorrect
Modified	There is a possibility to add null records when no validations are made or no exceptions are handled.	Medium	Incorrect
Changed	A datagrid displays the content only when the recordset is open.	Low	Database Incorrect

6.2.5 Statistical Analysis and Validation

Based on the AI derived from the above requirement data analysis, a statistical study was carried out to understand and analyze the statistical significance and relationship of AI across phases.

A hypothesis was formulated based on the conditions of analysis as follows;

H0 : Requirements Amplification Index is same across technologies
H1 : Requirements Amplification Index is different between technologies

Minitab tool was used to analyze the data set of Requirement Amplification Index. A simple T-test was run to validate the statistical significance of the requirement AI data across technologies.

Minitab output on the Hypothesis Testing is listed in **Table 6** below.

The statistical rule of elimination is:

1) If the P- Value > 0.05, Then H0 is true and there is no difference in the groups. = Accept H0

2) If the Value < 0.05, Then H0 is false and there is a statistically significant difference. = Reject H0 and Accept H1

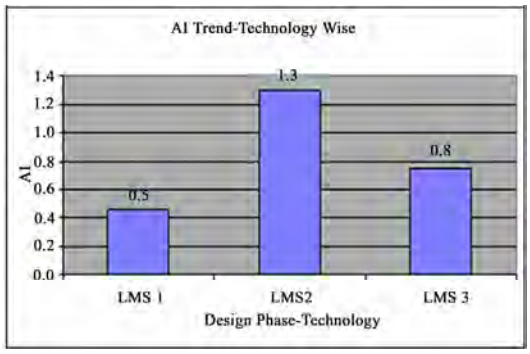


Figure 5. Amplification index trend – design

Table 6. Statistical analysis computation

One-Sample T:				
Test of mu = 0 vs mu not = 0				
Variable	N	Mean	StDev	SE Mean
AI	3	0.867	0.404	0.233
Variable	95.0% CI		T	P
AI	(–0.137,	1.871)	3.71	**0.065**

This results in: 0.065 > 0.05; so by the rule, Accept the Ho.

To conclude that, "Requirements Amplification Index is same across technologies and there is no statistical significant difference on AI across technologies in the Library Management System (LMS) developed in different technologies".

6.3 Coding Phase Analysis

Coding was performed with (fault injected) design as basis. There were several defects observed with "source" as Design and Requirements. The Injected defects were the major cause for Code defects detected in Code review.

6.3.1 Code Review

Table 7 appends to **Table 3** with the number of defects injected independently with leaked defected in design document. Further, it lists the defect leakage from Design to Code with inherent defects detected after Code review for each experiment.

6.3.2 Code Defect Amplification: Technology Variant

Figure 6 represents the comparison of Amplification Index between the LMS developed on different technologies.

The amplification of coding defects caused due to the injected design defects in LMS is evidenced in all technologies and more prominent in VB Microsoft technology.

6.3.3 Amplification Index for Design

Table 8 illustrates the methodology and computation details used to calculate Design Amplification Index (*i.e.* impact of Design defects on Code).

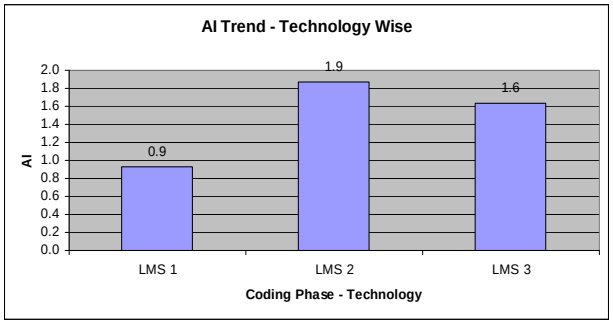


Figure 6. Amplification index trend—coding

Table 7. Defects injected—design to code

	Source					
	SRS		Design		Code	
	Injected	Inherent	Leaked + Injected	Inherent	Leaked	Inherent
LMS 1	5	4	4 + 5	8	7	7
LMS 2	5	8	7 + 5	6	9	6
LMS 3	5	5	7 + 5	9	10	6

6.3.4 Statistical Analysis and Validation

Similarly, based on the AI derived from the above design data analysis, a statistical study was carried out to understand and analyze the statistical significance and relationship of AI across design phases.

A hypothesis was formulated based on the conditions of analysis as follows;

H0 : Design Amplification Index is same across technologies
H1 : Design Amplification Index is different between technologies

Minitab tool was used to analyze the data set of design Amplification Index. A simple T-test was run to validate the statistical significance of the design AI data across technologies.

Minitab output on the Hypothesis Testing is listed in **Table 9** below.

The statistical rule of elimination is:

1) If the P- Value > 0.05, Then H0 is true and there is no difference in the groups. = Accept H0

2) If the Value < 0.05, Then Ho is false and there is a statistically significant difference. = Reject H0 and Accept H1

This results in: 0.038 < 0.05; so by the rule, Reject H0 and Accept H1.

To conclude that, "Design Amplification Index is different across technologies and there is a statistical significant difference on design AI across technologies in

Table 8. Defects amplification index computation—design to code

Application	Formula	AI (Design on Code)
LMS1	No. of Code defects caused due to injected Design Defects / No. of injected Design defects	4.9/5 = 0.9 (rounded) ➔ *One design defect leaked causes 0.9 defect in code in C # technology*
LMS2	No. of Code defects caused due to injected Design Defects / No. of injected Design defects	9/5 = 1.9 (rounded) ➔ *One design defect leaked causes 1.9 defect in code in VB technology*
LMS3	No. of Code defects caused due to injected Design Defects / No. of injected Design defects	8/5 = 1.6 (rounded) ➔ *One design defect leaked causes 1.6 defect in code in Java technology*

Table 9. Statistical analysis computation

One-Sample T:				
Test of mu = 0 vs mu not = 0				
Variable	N	Mean	StDev	SE Mean
AI	3	1.467	0.513	0.296
Variable	95.0% CI		T	P
AI	(0.192,	2.741)	4.95	0.038

the Library Management System (LMS) developed in different technologies".

6.4 Testing Phase Analysis

Testing was performed with (fault injected) code as basis. There were several defects observed with "source" as Coding, Design and Requirements. The injected defects were the major cause for Code defects detected in Testing.

6.4.1 Testing

Table 10 appends to **Table 7** with the number of defects injected independently with leaked defected in Code. Further, it lists the defect leakage from Code to Test Cases with inherent defects detected after Test Case review for each experiment.

6.4.2 Test Defect Amplification: Technology Variant

Figure 7 represents the comparison of Amplification Index between the LMS developed on different technologies. The amplification of test defects caused due to the injected code defects in LMS is evidenced in all technologies and more prominent in VB Microsoft technology.

6.4.3 Amplification Index for Code

Table 11 illustrates the methodology and computation details used to calculate Test Amplification Index (*i.e.* impact of Code defects on Test results).

6.4.4 Statistical Analysis and Validation

Similarly, based on the AI derived from the above code data analysis, a statistical study was carried out to understand and analyze the statistical significance and relationship of AI across test phase.

Table 10. Defects injected—code to test

	Source					
	Design		Code		Testing	
	Leaked + Injected	Inherent	Leaked + Injected	Inherent	Leaked	Inherent
LMS1	4 + 5	8	7 + 5	7	9	0
LMS2	7 + 5	6	9 + 5	6	14	4
LMS3	7 + 5	9	10 + 5	6	15	2

Table 11. Defects amplification index computation—code to test

Application	Formula	AI (Code on Test results)
LMS1	No. of Test results defects caused due to injected Code Defects / No. of injected Code defects	9/5 = 1.9 (rounded) ➔ *One code defect leaked causes 1.9 defect in test results in C # technology*
LMS2	No. of Test results defects caused due to injected Code Defects / No. of injected Code defects	11/5 = 2.1 (rounded) ➔ *One code defect leaked causes 2.1 defect in test results in VB technology*
LMS3	No. of Test results defects caused due to injected Code Defects / No. of injected Code defects	7/5 = 1.5 (rounded) ➔ *One code defect leaked causes 1.5 defect in test results in Java technology*

A hypothesis was formulated based on the conditions of analysis as follows;

Ho : Code Amplification Index is same across technologies
H1 : Code Amplification Index is different between technologies

Minitab tool was used to analyze the data set of Code Amplification Index. A simple T-test was run to validate the statistical significance of the code AI data across technologies.

Minitab output on the Hypothesis Testing is listed in **Table 12** below.

The statistical rule of elimination is:

1) If the P- Value > 0.05, Then Ho is true and there is no difference in the groups. = Accept Ho

2) If the Value < 0.05, Then Ho is false and there is a statistically significant difference. = Reject Ho and Accept H1

This results in: 0.009 < 0.05; so by the rule, Reject Ho and Accept H1.

To conclude that, "Code Amplification Index is different across technologies and there is a statistical significant difference on design AI across technologies in the LiBrary Management System (LMS) developed in different technologies".

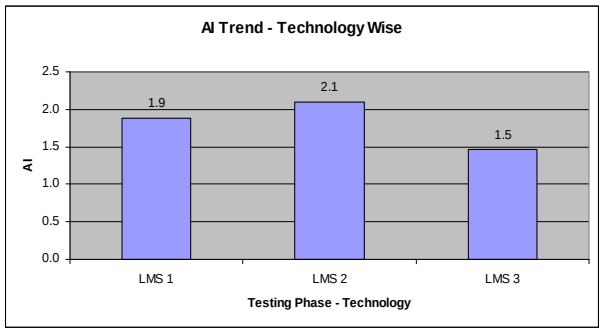


Figure 7. Amplification index trend—testing

Table 12. Statistical analysis computation

One-Sample T:

Test of mu = 0 vs mu not = 0

Variable	N	Mean	StDev	SE Mean
AI	3	1.833	0.306	0.176
Variable	95.0% CI		T	P
AI	(1.074,	2.592)	10.39	**0.009**

7. Conclusions

AI Trend Analysis

The Amplification Index indicates the extent of damage caused by a defect in various phases of the project. The index increases with every step in the life cycle of the project. This is evident in the case of Microsoft technologies (VB and C#.net) but AI in the case of open source technologies such Java, the AI increases in requirements and design but in code, it is found have marginal decrease compared to other technologies. It is also seen that defects amplification in the VB Technology show substantial increase in the amplification index across phases compared to other selected technologies.

The relative growth of AI across phases in Java technology is less compared to Microsoft technology. This indicates a better fault tolerance for Java technology.

It was concluded and validated statistically that;

- Requirement defects amplification index across on identified technologies remains are same.
- Design and Code defects amplification index across on technologies vary based on technologies for the common application developed in the same domain.

Figure 8 illustrates the consolidated Amplification Index trend across technologies classified under each phase of SDLC;

The defect leakage analysis emphasizes the importance of thorough and systematic reviews in the early stages of a software project with an emphasis on defect prevention. The analysis indicates a high increase of cost and effort to remove the defects at later stages. The number of defects increases exponentially as a direct result of defects leaked from previous stages.

Figure 9 consolidated the defect leakage pattern across technologies distributed each SDLC phase.

8. Future Experiments

Currently, the study is being extended to analyze the effect of the defects and amplification index in the development phases of the different domain based projects developed with same technology.

Guidelines for review time and effort estimation are being computed by analyzing and defining the review and test stop criteria. Error seeding during testing can be carried out to define the test stop criteria.

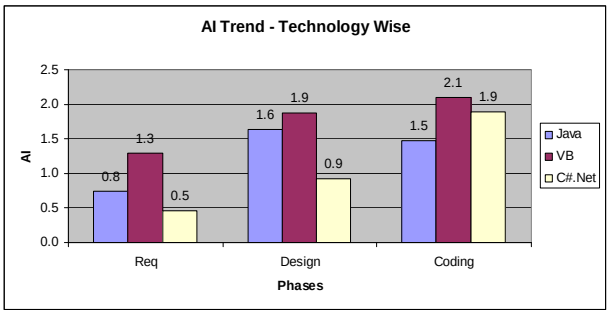


Figure 8. Amplification index trend – technology wise

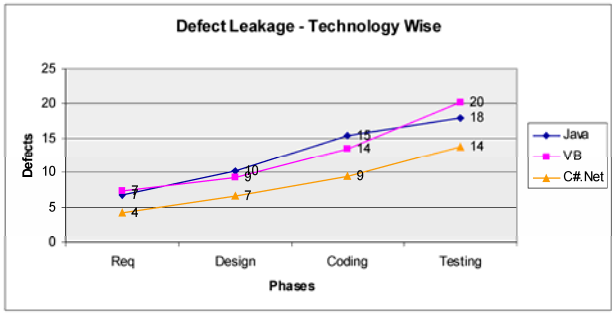


Figure 9. Defect leakage

9. Limitations of Experiments

The following are the limitations of the experiments:

- Causal analysis is relatively subjective to understand the cause of amplified defect. This required detailed review and discussion with project team and technical/technology experts.
- Defect removal efficiency percentage used for experiments in different technologies are based on a test in a sample requirement, design and code with known defects provided to project members and review efficiency percentage derived from the defects detected.
- It is verified that the skill set of the analysts and programmers working in the projects are same and/or similar across technologies.

The experiments do not consider specialized automated tools and techniques used in the development of software work products which could have impact of the work product output quality.

REFERENCES

[1] A. Hall, "Seven Myths of Formal Methods," *IEEE Software*, September 1990, pp. 11-19.

[2] C. B. Jones, "Systematic Software Development Using VDM," Prentice-Hall International, London, 1986.

[3] S. J. Garland, J. V. Guttag and J. J. Horning. "Debugging Larch Shared Language Specifications," *IEEE Transactions on Software Engineering*, September 1990, pp. 1044-1057.

[4] W. Howden, "A Functional Approach to Program Testing and Analysis," *IEEE Transactions on Software Engineering*, October 1986, pp. 997-1005.

[5] L. J. White, "Basic mathematical Definitions and Results in Testing," In: B. Chandrasekaran and S. Radicchi, Ed., *Computer Program Testing*, North-Holland, 1981, pp. 13-24.

[6] R. DeMillo, R. Lipton and A. Perlis, "Social Processes and Proofs of Theorems and Programs," *Communications of the ACM*, May 1979, pp. 803-820.

[7] B. W. Johnson, "Design and Analysis of Fault-Tolerant Digital Systems," Addison-Wesley, Massachusetts, 1989.

[8] D. Dreilinger and L. J. Lin, "Using Fault Injection to Test Software Recovery Code," November 1995.

[9] N. G. M. Leme, E. Martins and C. M. F. Rubira, "A Software Fault Injection Pattern System," *Proceedings of the* 9*th Brazilian Symposium on Fault-Tolerant Computing*, Florianópolis, 5-7 March 2001, pp. 99-113.

J. Software Engineering & Applications, **2010, 3: 374-383**
doi:10.4236/jsea.2010.34042 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Time Series Forecasting of Hourly PM10 Using Localized Linear Models

Athanasios Sfetsos, Diamando Vlachogiannis

Environmental Research Laboratory, INTR-P, National Centre for Scientific Research "Demokritos", Attikis, Greece.
Email: ts@ipta.demokritos.gr

Received October 15th, 20009; revised November 4th, 2009; accepted November 8th, 2009.

ABSTRACT

*The present paper discusses the application of localized linear models for the prediction of hourly PM10 concentration values. The advantages of the proposed approach lies in the clustering of the data based on a common property and the utilization of the target variable during this process, which enables the development of more coherent models. Two alternative localized linear modelling approaches are developed and compared against benchmark models, one in which data are clustered based on their spatial proximity on the embedding space and one novel approach in which grouped data are described by the same linear model. Since the target variable is unknown during the prediction stage, a complimentary pattern recognition approach is developed to account for this lack of information. The application of the developed approach on several PM*10 *data sets from the Greater Athens Area, Helsinki and London monitoring networks returned a significant reduction of the prediction error under all examined metrics against conventional forecasting schemes such as the linear regression and the neural networks.*

Keywords*: *Localized Linear Models, PM*10 *Forecasting, Clustering Algorithms

1. Introduction

Environmental health research has demonstrated that Particulate Matter (PM) is a top priority pollutant when considering public health. Studies of long-term exposure to air pollution, mainly to PM, suggest adverse long- and short-term health effects, increased mortality (e.g. [1,2]), increased risk of respiratory and cardiovascular related diseases (e.g. [3]), as well as increased risk of developing various types of cancer [4]. Hence, the development and use of accurate and fast models for forecasting PM values reliably is of immense interest in the process of decision making and modern air quality management systems.

In order to evaluate the ambient air concentrations of particulate matter, a deterministic urban air quality model should include modelling of turbulent diffusion, deposition, re-suspension, chemical reactions and aerosol processes. In recent years, an emerging trend is the application of Machine Learning Algorithms (MLA), and particularly, that of the Artificial Neural Networks (ANN) as a means to generate predictions from observations in a location of interest. The strength of these methodologies lies in their ability to capture the underlying characteristics of the governing process in a non-linear manner, without making any predefined assumptions about its properties and distributions. Once the final models have been determined, it is then a straight-forward and exceedingly fast process to generate predictions. However, ANN have also inherent limitations. The main one is the extension of models in terms of time period and location; this always requires training with locally measured data. Moreover, these models are not capable of predicting spatial concentration distributions.

Owing to the importance and significant concentrations of PM in major European cities, there is an increasing amount of literature concerned with the application of statistical models for the prediction of point PM values. For the purposes of the EU-funded project APPETISE, an inter-comparison of different air pollution forecasting methods was carried out in Helsinki [5]. Neural networks demonstrated a better forecasting accuracy than other approaches such as linear regression and deterministic models.

In [6], Perez *et al*. compared predictions produced by three different methods: a multilayer neural network, linear regression and persistence methods. The three methods were applied to hourly averaged PM2.5 data for the years of 1994 and 1995, measured at one location in the downtown area of Santiago, Chile. The prediction errors for the hourly PM2.5 data were found to range

from 30% to 60% for the neural network, from 30% to 70% for the persistence approach, and from 30% to 60% for the linear regression, concluding however that the neural network gave overall the best results in the prediction of the hourly concentrations of PM2.5.

In [7], Gardner undertook a model inter-comparison using Linear Regression, feed forward ANN and Classification and Regression Tree (CART) approaches, in application to hourly PM10 modelling in Christchurch, New Zealand (data period: 1989-1992). The ANN method outperformed CART and Linear Regression across the range of performance measures employed. The most important predictor variables in the ANN approach appeared to be the time of day, temperature, vertical temperature gradient and wind speed.

In [8], Hooyberghs *et al.* presented an ANN for forecasting the daily average PM10 concentrations in Belgium one day ahead. The particular research was based upon measurements from ten monitoring sites during the period 1997-2001 and upon the ECMWF (European Centre for Medium-Range Weather Forecasts) simulations of meteorological parameters. The most important input variable identified was the boundary layer height. The extension of this model with further parameters showed only a minor improvement of the model performance. Day-to-day fluctuations of PM10 concentrations in Belgian urban areas were to a larger extent driven by meteorological conditions and to a lesser extent by changes in anthropogenic sources.

In [9], Ordieres *et al*. analyzed several neural-network methods for the prediction of daily averages of PM2.5 concentrations. Results from three different neural networks (feed forward, Radial Basis Function (RBF) and Square Multilayer Perceptron) were compared to two classical models. The results clearly demonstrated that the neural approach not only outperformed the classical models but also showed fairly similar values among different topologies. The RBF shows up to be the network with the shortest training times, combined with a greater stability during the prediction stage, thus characterizing this topology as an ideal solution for its use in environmental applications instead of the widely used and less effective ANN.

The problem of the prediction of PM10 was addressed in [10], using several statistical approaches such as feed-forward neural networks, pruned neural networks (PNNs) and Lazy Learning (LL).The models were designed to return at 9 a.m. the concentration estimated for the current day. The forecast accuracy of the different models was comparable. Nevertheless, LL exhibited the best performances on indicators related to average goodness of the prediction, while PNNs were superior to the other approaches in detecting the exceedances of alarm and attention thresholds.

In view of the recent developments in PM forecasting, the present paper introduces an innovative approach based on localized linear modelling. Specifically, two alternative localized liner modelling approaches are developed and compared against benchmark models such as the linear regression and the artificial neural networks. The advantage of the proposed approach is the identification of the finer characteristics and underlying properties of the examined data set through the use of suitable clustering algorithms and the subsequent application of a customized linear model on each one. Furthermore, the use of the target variable in the clustering stage enhances the coherence of the localized models. The developed approach is applied on several data sets from the monitoring networks of the Greater Athens Area and Helsinki, during different seasons.

2. Modelling Approaches

Time series analysis is used for the examination of a data set organised in sequential order so that its predominant characteristics are uncovered. Very often, time series analysis results in the description of the process through a number of equations (Equation (1)) that in principle combine the current value of the series, y_t, to lagged values, y_{t-k}, modelling errors, e_{t-m}, exogenous variables, $\boldsymbol{x}_{t-j}$, and special indicators such as time of the day. Thus, the generalized form of this process could be written as follows:

$$y_t = f\,(y_{t-k}, \boldsymbol{x}_{t-j}, e_{t-m} \mid \textit{various } k,j,m \textit{ and special indicators}) \quad (1)$$

2.1 Linear Regression

This approach uses linear regression models to determine whether a variable of interest, y_t, is linearly related to one or more exogenous variable, $\mathbf{x}_t$, and lagged variables of the series, y_t. The expression that governs this model is the following:

$$y_t = c + \sum_{k}^{k} \beta_k y_{t-k} + \sum_{j}^{j} \gamma_j \mathbf{x}_{t-j} + \varepsilon \quad (2)$$

The coefficients c, β, γ are usually estimated from a least squares algorithm. The inputs should be a set of statistically significant variables, defined under Student's t-test, estimated from the examination of the correlation coefficients or using a backward elimination selection procedure from a larger initial set.

2.2 Artificial Neural Networks (ANN)

The multi-layer perceptron or feed-forward ANN [11] has a large number of processing elements called neurons, which are interconnected through weights (w_{iq}, v_{qj}). The neurons expand in three different layer types: the input, the output, and one or more hidden layers. The signal flow is from the input layer towards the output. Each neuron in the hidden and output layer is activated by a

nonlinear function that relies on a weighted sum of its inputs and a neuron-specific parameter, called bias, b. The response of a neuron in the output layer as a function of its inputs is given by Equation (3), where f_1 and f_2 can be sigmoid, linear or threshold activation functions.

$$y_i = f_1\left(\sum_{q=1}^{l} w_{iq} f_2\left(\sum_{j=1}^{m}(v_{qj}x_j + b_j)\right) + b_q\right) \quad (3)$$

The strength of neural networks lies in their ability to simulate any given problem from the presented example, which is achieved from the modification of the network parameters through learning algorithms. In this study, the Levenberg-Marquardt [12] algorithm is applied because of its speed and robustness against the conventional back-propagation.

The most important issue concerning the introduction of ANN in time series forecasting is "generalization", which refers to their ability to produce reasonable forecasts on data sets other than those used for the estimation of the model parameters. This problem has two important parameters that should be accounted for. The first is data preparation, which involves pre-processing and the selection of the most significant variables. The second embraces the determination of the optimum model structure that is closely related with the estimation of the model parameters. Although, there is no systematic approach, which can be followed [13], some useful insight can be found using statistical methods such as the correlation coefficients.

The second aspect can be jointly tackled under the cross-validation training scheme. The data set is split into three smaller sets the training (TS), the evaluation or validation (ES) and the prediction or testing (PS) sets. The model is initialized with a few parameters. The next step is to train the model using data from the training set and when the error of the evaluation set is minimized, the model parameters and configuration are stored. The number of parameters is then increased and a new network is trained from the beginning. If ES error is lower compared to the previously found minimum, then the parameters of this new model are stored. This iterative process is terminated when a predefined number of iterations are reached (**Figure 1**).

In this study, ES was formed using a Euclidean metric withholding a percent value (here 25% is used) of the TS data that are located nearest to other data. The strength of this approach lies in the fact that TS covers more distinct characteristics of the process, thus, allowing for the development of a model with better generalization capabilities.

2.3 Nearest Neighbours

This class of hybrid models includes a local modelling and a function approximation to capture recent dynamics

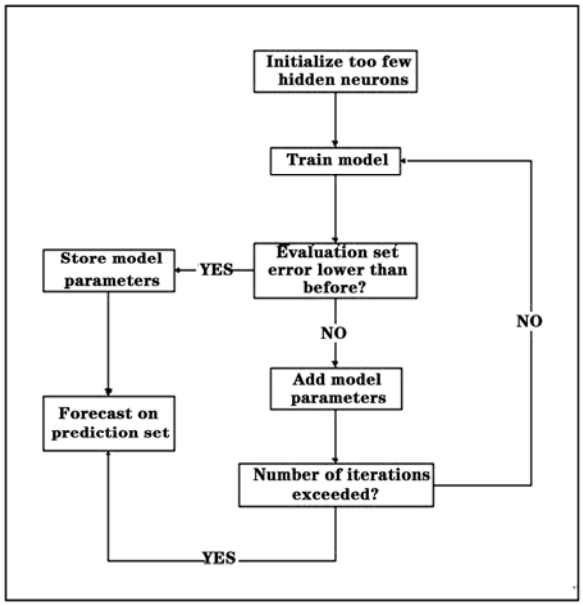


Figure 1. Iterative cross-validation training

of the process. The underlying aim of these predictors is that segments of the series neighbouring under some distance measure may correspond to similar future values. This claim was endorsed by the work of Farmer and Sidorowich [14] that showed that the chaotic time-series prediction is several orders of magnitude better using local approximation techniques rather than universal approximators. The tricky part in these models is the selection of the embedding dimension, which effectively determines segments of the series, and the number of neighbours. Initially, it is required to estimate the embedding dimension d and time delay τ of the attractor as follows:

$$Y(t) = \left[y(t-\tau),...,y(t-(d-1)\tau), \mathbf{x}(\mathrm{t-j})\right] \quad (4)$$

In this study, a value of $\tau = 1$ was used and $Y(t)$ had the same parameters as the linear regression model. The number of neighbours was not pre-determined but was set to vary between predefined limits. A small number of neighbours increase the variance of the results whereas a large number can compromise the local validity of a model and increase the bias of results. Once the nearest neighbours to $Y(t)$ have been identified, an averaging procedure is followed in the present study to generate predictions.

2.4 Local Models with Clustering Algorithms (LMCA)

The idea behind the application of clustering algorithms in time series analysis is to identify groups of data that share some common characteristics. On each of these groups, the relationships amongst the members are modelled through a single equation model. Consequently,

each of the developed models has a different set of parameters. The process is described in the following steps:

1) Selection of the input data for the clustering algorithm. This can contain lagged and/or future characteristics of the series, as well as other relevant information.

C(t) = [y_t, y_{t-k}, $\mathbf{x}_{t-j}$]. Empirical evidence suggests that the use of the target variable y_t is very useful to discover unique relationships between input-output features. Additionally, higher quality modelling is ensured with the function approximation since the targets have similar properties and characteristics. However, this occurs to the expense of an additional process needed to account for this lack of information in the prediction stage.

2) Application of a clustering algorithm combined with a validity index or with user defined parameters, so that n_{cl} clusters will be estimated.

3) Assign all patterns from the training set to the n_{cl} clusters. For each of the clusters, apply a function approximation model, $y_t = f_i(y_{t-k}, \mathbf{x}_{t-j}),\ i = 1...n_{cl}$, so that n_{cl} forecasts are generated.

Successful application of this method has been reported on the prediction of locational electricity marginal prices [15], Mckay Glass and daily electric load peak series [16], the A and D series of the Santa Fe forecasting competition [17] and hourly electric load [18].

In this study, the k means clustering algorithm was selected [19]. It is a partitioning algorithm that attempts to directly decompose the data set into a set of groups through the iterative optimization of a certain criterion. More specifically, it re-estimates the cluster centres through the minimization of a distance-related function between the data and the cluster centres. The algorithm terminates when the cluster centres stop changing.

The optimal number of clusters is determined using a modified cluster validity index, CVI, [20], which is directly related to the determination of the user-defined (here the number of clusters) parameters of the clustering algorithm. Two indices are used for showing an estimate of under-partitioning (U_u) and over-partitioning (U_o) of the data set:

$$U_u = \frac{1}{c}\sum_{i=1}^{c} MD_i$$
$$U_o = \frac{c}{d_{\min}} \quad (5)$$

MD_i is the mean intra-cluster distance of the i-th cluster. Here, $d_{\min}$ is the minimum distance between cluster centres, which is a measure of intra-cluster separation. The optimum number is found from the minimization of a normalized combinatory expression of these two indices.

2.5 Hybrid Clustering Algorithm (HCA)

The hybrid clustering algorithm is an iterative procedure that groups data, based on their distance from the hyper-plane that best describes their relationship. It is implemented through a series of steps, which are presented below:

1) Determine the most important variables.

2) Form the set of patterns H(t) = [y_t, y_{t-k}, $\mathbf{x}_{t-k}$].

3) Select the number of clusters n_h.

4) Initialize the clustering algorithm so that n_h clusters are generated and assign patterns.

5) For each new cluster, apply a linear regression model to y_t using as explanatory variables the remaining of the set H_t.

6) Assign each pattern to a cluster based on their distance.

7) Go to 5) unless any of the termination procedures is reached.

The following termination procedures are considered: a) the maximum pre-defined number of iterations is reached and b) the process is terminated when all patterns are assigned to the same cluster as in the previous iteration in 6). The selection of the most important lagged variables, 1), is based on the examination of the correlation coefficients of the data.

The proposed clustering algorithm is a complete time series analysis scheme with a dual output. The algorithm generates clusters of data, the identical characteristic of which is that they "belong" to the same hyper-plane, and synchronously, estimates a linear model that describes the relationship amongst the members of a cluster. Therefore, a set of n_h linear equations is derived (Equation (6)).

$$\hat{y}_{t,i} = a_{o,i} + \sum a_{i,k} y_{t-k} + \sum b_{i,j} X_{t-j} \quad , \quad i = 1\ldots n_h \quad (6)$$

Like any other hybrid model that uses the target variables in the development stage, the model requires a secondary scheme to account for this lack of information in the forecasting phase. For HCA and LMCA, the only requirement is the determination of the cluster number, n_h and n_{cl} respectively, which is equivalent to the estimation of the final forecast.

The optimum number of HCA clusters is found from a modified cluster validity criterion. An estimate of under-partition (U_u) of the data was formed using the inverse of the average value of the coefficient of determination (R_i^2) on all regression models. U_o indicates the over-partitioning of the data set, and $d_{\min}$ is the minimum distance between linear models (Equation (7)). The optimum number is found from the minimization of a normalized combinatory expression of these two indices.

$$U_u = \frac{1}{\frac{1}{h}\sum_{i=1}^{h} R_i^2}$$
$$U_o = \frac{c}{d_{\min}} \quad (7)$$

2.6 Pattern Recognition

A pattern recognition scheme with three alternative approaches was then applied to convert the LMCA and HCA output to the final predictions. Initially, a conventional clustering (k-means) algorithm was employed to identify similar historical patterns in the time series. The second was to determine n_{cl} / n_h at each time step, using information contained in the data of the respective cluster.

(p1) Select a second data vector using *only* historical observations $P_t = [y_{t-k}, \mathbf{x}_{t-k}]$

(p2) Initialize a number of clusters n_k

(p3) Apply a k-means clustering algorithm on P_t.

(p4) Assign data vectors to each cluster, so that each of the n_k clusters should contain k_m, m = 1,…, n_k data.

To obtain the final forecasts the following three alternatives were examined:

(M1) From the members of the *k*-th cluster find the most frequent LMCA / HCA cluster, *i.e.* n_{cl} / n_h number.

(M2) From the members of the *k*-th cluster estimate the final forecast as a weighted average of the LMCA/ HCA clusters. Here p_i is the percentage of appearances of the LMCA / HCA cluster in the *k*-th cluster data.

$$y_t = \sum_i p_i y_{t,n} \qquad i = 1,...,k \;\; and \;\; n = 1,...,n_h \text{ or } n_{cl} \tag{8}$$

(M3) From the members of the *k*-th cluster estimate the final forecast as a distance weighted average of the HCA clusters.

$$\begin{aligned} y_t &= \sum_i t_i y_{t,n} \qquad i = 1,...,k \;\; and \;\; n = 1,...,n_h \text{ or } n_{cl} \\ d_i &= \left\| P_t - P_i \right\| \\ t_i &= \frac{d_i^{-a}}{\sum_i d_i^{-a}} \text{ and } a = 2 \end{aligned} \tag{9}$$

The optimal number of clusters for the pattern recognition stage was determined using the modified compactness and separation criterion for the k-means algorithm discussed previously in section "Local Models with Clustering Algorithms".

3. Data Description and Results

The previously described forecasting methodologies were applied to eight different data sets both univariate and multivariate. The data sets were hourly PM10 concentration values from the monitoring network in the Greater Athens Area and in the cities of Helsinki and London, spanning over different seasons. It should be clarified that meteorological data were available only from the Helsinki station. The results returned by the applied algorithms for each station are discussed separately in the following sections.

In addition to the combined LMCA / HCA – PR methodology, the ideal case of a perfect knowledge of the n_{cl} / n_h parameter is also presented. This indicates the predictive potential, or the least error that the respective methodology could achieve. Also, the base-case persistent approach ($y_t = y_{t-1}$) is presented as a relative criterion for model inter-comparison amongst different data sets. The ability of the models to produce accurate forecasts was judged against the following statistical performance metrics:

Root Mean Square Error

$$RMS = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(O_i - P_i)^2} \tag{10}$$

Normalized RMS

$$NRMS = \frac{\sum_{i=1}^{k}(O_i - P_i)^2}{\sum_{i=1}^{k}(\overline{O} - O_i)^2} \tag{11}$$

Mean Absolute Percentage Error

$$MAPE = 100\frac{1}{k}\sum_{i=1}^{k}\frac{|O_i - P_i|}{O_i} \tag{12}$$

Index of Agreement

$$IA = 1 - \frac{\sum_{i=1}^{k}(O_i - P_i)^2}{\sum_{i=1}^{k}\left(|\overline{O} - P_i| + |\overline{O} - O_i|\right)^2} \tag{13}$$

Fractional Bias

$$FB = \frac{(\overline{O} - \overline{P})}{0.5 * (\overline{O} + \overline{P})} \tag{14}$$

3.1 Greater Athens Area – Aristotelous Str

The selected station from the Greater Athens Area monitoring network was Aristotelous Str. It is located at 23°43′39′′ North and 37°59′16′′ West, at an elevation height of 95 m above ground level. It is characterized as an urban station, positioned in the city centre with traffic dominated emissions. The training and the prediction sets covered the periods from 1/7/2001 to 14/8/2001 and 15/8/2001 to 31/8/2001, respectively.

The analysis revealed that the most influential variables were PM_{t-1}, PM_{t-2}, PM_{t-24}, PM_{t-25} and an indicator for the time of the day. This data set was used for the development of all methodologies and the input set for the pattern recognition scheme. The results on **Table 1** indicate that with the exception of NN, all other conventional approaches demonstrate a reduction of the prediction error by approximately 6% on the basis of the RMS error compared to the base case persistent method. The difference between LR and ANN was not found to be

statistically significant, although the later was marginally better under all criteria.

The application of the local linear models was able to reduce the predictive error by an order of magnitude depending on the pattern recognition scheme that was applied. Both LMCA and HCA are capable of reaching exceedingly lower prediction error, with IA above 0.98, if all n_{cl}/n_h clusters are predicted correctly at each time step. **Figure 2** presents a graphical description of the prediction error of the HCA-perfect cluster forecast. The HCA coupled with the M3 scheme returned the overall best prediction error that was approximately 8% lower than that of the persistent approach.

3.2 Greater Helsinki Area – Kallio

The data from the Helsinki monitoring network were from the suburban station of Kallio, with co-ordinates 25°52′92′′ W and 66°75′47′′ N and elevation height of 21 m above sea level. The training set was from 3/9/2003 to 9/11/2003, whereas the unknown prediction set spanned from 10/11/2003 to 30/11/2003.

The developed models for the prediction of PM10 val-

Table 1. Prediction results from Aristotelous

	RMS	NRMS	MAPE	d	FB	Nu of clusters
Persistent	9.5596	0.3112	13.006	0.9223	–0.0002	
LR	9.0193	0.277	12.6536	0.9007	0.0052	
ANN	8.9311	0.2716	12.3984	0.9152	0.0037	
NN	10.117	0.3485	14.3699	0.892	–0.0094	24
LCMA	n_{cl} = 4					n_k = 32
Perfect	4.6355	0.0732	7.2108	0.9813	–0.0043	
M1	9.6748	0.3187	13.3351	0.8999	–0.0107	
M2	9.0637	0.2797	12.434	0.9121	–0.0052	
M3	9.0559	0.2793	12.3804	0.9108	–0.009	
HCA	n_{cl} = 8					n_k = 13
Perfect	2.1522	0.0158	2.857	0.9961	–0.0002	
M1	9.6085	0.3144	12.5104	0.9105	–0.0134	
M2	8.8787	0.2684	12.3668	0.915	0.0048	
M3	8.8153	0.2646	12.3368	0.9178	0.0046	

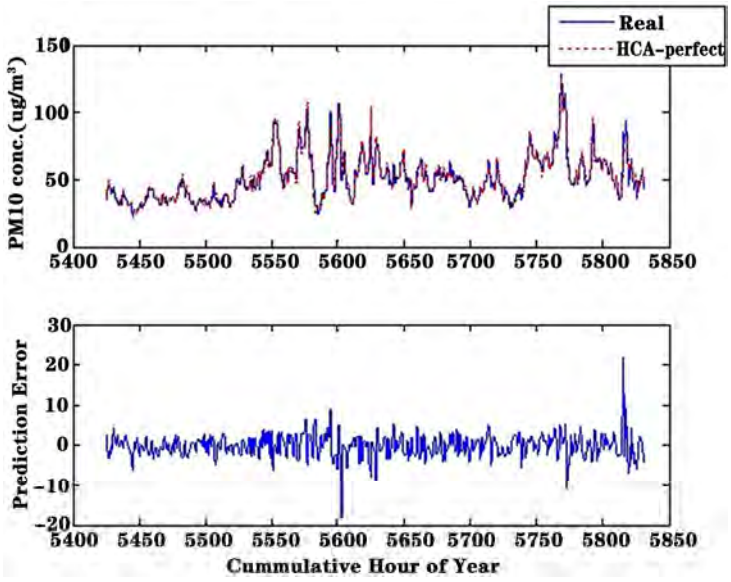


Figure 2. HCA perfect cluster forecast for the Aristotelous station (Athens)

Table 2. Linear regression model details for Helsinki 1

Variable	Coef.	St. Error	t-stat.	Variable	Coef.	St. Error	t-stat.
c	4.7626	1.1361	4.1921	T_{t-1}	1.0627	0.2753	3.8601
PM $_{t-1}$	0.7611	0.0247	30.8584	T_{t-2}	–1.0446	0.274	–3.8128
PM $_{t-2}$	0.0622	0.0246	2.5319	u_{t-1}	–0.749	0.2213	–3.3847
PM $_{t-24}$	0.0232	0.0136	1.7008	u_{t-2}	0.6094	0.2216	2.7493
RH $_{t-1}$	0.2055	0.0547	3.7582	v_{t-1}	0.6673	0.2242	2.9767
RH $_{t-2}$	-0.2361	0.0547	-4.317	v_{t-2}	–0.4508	0.2257	–1.9968

Table 3. Prediction Results from Helsinki 1

		RMS	NRMS	MAPE	d	FB	Nu of clusters
Persistent		5.1208	0.2793	33.3564	0.9301	0.0001	
LR		4.9654	0.2626	36.4317	0.9073	–0.0139	
ANN		5.1722	0.2849	39.5785	0.9085	–0.0591	
NN		5.6876	0.3446	43.8667	0.857	–0.0489	13
LCMA	n_{cl} = 3						n_k = 61
Perfect		3.033	0.098	18.1484	0.9724	0.0038	
M1		5.1044	0.2775	37.1176	0.9295	–0.0119	
M2		4.892	0.2549	37.5676	0.9193	0.0008	
M3		4.8416	0.2497	36.905	0.9229	0.0049	
HCA	n_{cl} = 7						n_k = 19
Perfect		1.5653	0.0261	8.9912	0.9932	–0.0051	
M1		5.2179	0.29	42.6351	0.9072	0.021	
M2		4.8139	0.2468	37.4036	0.9203	–0.0018	
M3		4.7612	0.2415	36.7128	0.9239	–0.0006	

ues from Helsinki contained meteorological parameters that were identified using a combination of statistical correlation properties and stepwise linear regression, discarding all those that were judged statistically as not significant under Student's t-test. The finally selected parameters and their estimation from the least squares fit are shown on **Table 2**.

The prediction results (**Table 3**) demonstrate that the forecasting ability of the conventional models is somewhat similar to that of the base-case persistent approach. The large prediction error of the ANN can be partly explained by the linear nature governing process that relates PM10 values to lagged values and from the over-fitting of the applied training scheme. The introduction of the LMCA and HCA localized models coupled with the M3 pattern recognition scheme returned the least overall prediction error that was approximately 5.5% and 7% respectively lower on the RMS criterion and double under NRMS. **Figure 3** shows the values of the prediction error of the LMCA-M3 modelling approach.

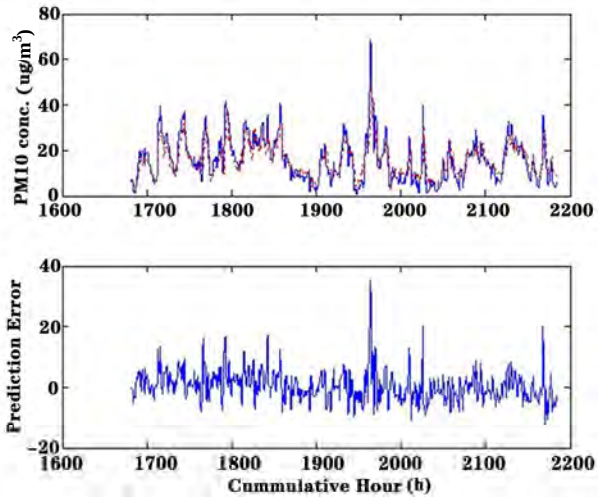


Figure 3. Prediction and error with LCMA – M3 approach

3.3 Greater London Area – Bloomsbury

The data from the Greater London Area were from the

Bloomsbury station located in the city centre of London (51°31'24" N, 0°7'54" W), characterised as an urban background station. The training set was selected to cover the period from 1/9/2005 to 22/10/2205, whereas the unknown prediction set comprised data ranging from 23/10/2005 to 6/11/2005.

The stepwise regression with a threshold value for the t-statistic of ± 1.96, corresponding to the 95% confidence interval, revealed as the most significant values PM_{t-1}, PM_{t-2}, PM_{t-24}. Additionally, an indicator for the time of the day was utilized. That data set was used for the development of all methodologies while the input set for the pattern recognition scheme. The analysis of the results (**Table 4**) indicated that none of the conventional forecasting approaches managed to return consistently lower prediction errors than the base case persistent approach. The least prediction error was returned from the ANN that was 3.6% lower than the persistent approach on the basis of the RMS error.

The developed localized linear model (HCA) has significant forecasting potential, as it can be observed in **Figure 4**, under the assumption of a perfect knowledge of the future cluster in the pattern recognition stage. The percentage improvement over the bench-mark persistent approach ranged from 40-70%. Similar results were found for the other two data sets

4. Discussion

The development and application of accurate models for forecasting PM concentration values in a rather fast and efficient manner is of primary concern in modern air quality management systems. The applied LR and ANN are nowadays mature approaches that have been integrated in many operational systems and could be used for the benchmarking of novel methodologies. The results of this work yielded that for the majority of the examined data sets, the linear approach marginally outperforms ANN. This indicates that the underlying process could possess predominantly linear characteristics.

The main focus of this work was the development and application of novel localized linear models. These were based on clustering algorithms as a means to identifying

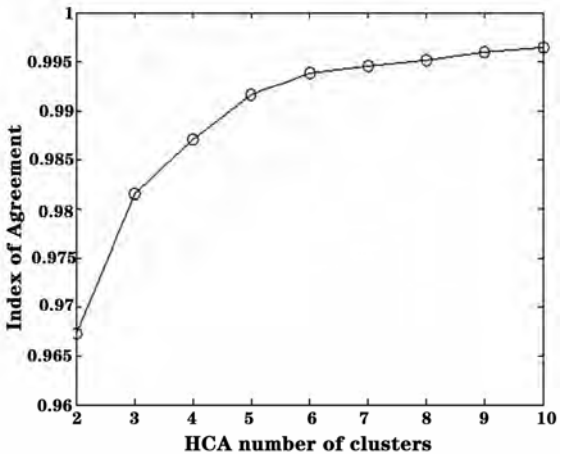


Figure 4. Index of agreement for HCA – perfect cluster forecast

Table 4. Prediction results from London Bloomsbury

	RMS	NRMS	MAPE	d	FB	Remarks
Persistent	4.4165	0.272	16.4202	0.9282	–0.0007	
LR	4.3119	0.2593	17.281	0.9257	0.0206	
ANN	4.256	0.2526	16.9101	0.9266	0.0221	
NN	5.1193	0.3655	22.466	0.8933	0.0075	14
LCMA	n_{cl} = 4					n_k = 16
Perfect	4.1665	0.2421	17.0711	0.9324	0.0193	
M1	4.4947	0.2817	17.9071	0.9137	–0.0136	
M2	4.2704	0.2543	17.051	0.9228	0.0069	
M3	4.3005	0.2579	17.9051	0.9229	0.021	
HCA	n_{cl} = 7					n_k = 29
Perfect	1.2401	0.0214	5.1061	0.9945	0.0064	
M1	4.3246	0.2608	16.8142	0.8877	–0.0188	
M2	4.2795	0.2554	16.8375	0.9163	0.0097	
M3	4.2513	0.2521	16.9179	0.8812	0.0046	

similar properties of the time series. The LMCA identified clusters based on their proximity on the embedding space, whereas HCA identified grouped points that were described by the same linear model. As both approaches included the target variable in the model development stage, a pattern recognition scheme was needed to account for this lack of information in the prediction stage.

The final prediction model was reached with the use of the modified CVI coupled with a pattern recognition scheme. The results suggested M3 as the most effective choice, because it produced consistently the least prediction error, under all metrics. For the RMS and MAPE errors, the improvement over the persistent approach ranged from 3.5% (London) to 7.7% (Athens and Helsinki). This value was almost doubled for NRMS and IA for the respective data sets. The HCA produced the least prediction error on every single examined data set, compared both to conventional approaches and the LCMA.

5. Conclusions

This paper introduced the application of localized linear models for forecasting hourly PM10 concentration values using data from the monitoring networks of the cities of Athens, Helsinki and London. The strength of this innovative approach is the use of a clustering algorithm that identifies the finer characteristics and the underlying relationships between the most influential parameters of the examined data set and subsequently, the development of a customized linear model. The calculated clusters incorporated the target variable in the model development phase, which was beneficial for the development of more coherent localized models. However, in order to overcome this lack of information in the prediction stage a complementary scheme was required. For the purposes of this study, a pattern recognition scheme based on the concept of weighted average distance (M3) was developed that consistently returned the least error under all examined metrics. The calculated results show that the proposed approach is capable of generating significantly lower prediction error against conventional approaches such as linear regression and neural networks, by at least one order of magnitude.

6. Acknowledgements

The assistance of members of OSCAR project (funded by EU under the contract EVK4-CT-2002-00083), for providing the Helsinki data for this research, is gratefully acknowledged. The Department of Atmospheric Pollution and Noise Control of the Hellenic Ministry of Environment, Physical Planning and Public Works is also acknowledged for the provision of the Athens data. The data from London are from the UK air quality archive (http://www.airquality.co.uk).

REFERENCES

[1] K. Katsouyanni, "Ambient Air Pollution and Health," *British Medical Bulletin*, Vol. 68, 2003, pp. 143-156.

[2] E. Samoli, A. Analitis, G. Touloumi, J. Schwartz, H. R. Anderson, J. Sunyer, L. Bisanti, D. Zmirou, J. M. Vonk, J. Pekkanen, P. Goodman, A. Paldy, C. Schindler and K. Katsouyanni, "Estimating the Exposure-Response Relationships between Particulate Matter and Mortality within the APHEA Multicity Project," *Environmental Health Perspectives*, Vol. 113, 2005, pp. 88-95.

[3] R. D. Morris, "Airborne Particulates and Hospital Admissions for Cardiovascular Disease: A Quantitative Review of the Evidence," *Environmental Health Perspectives*, Vol. 109, Supplement 4, 2001, pp. 495-500.

[4] E. G. Knox and E. A. Gilman, "Hazard Proximities of Childhood Cancer in Great Britain from 1953-1980," *Journal of Epidemiology and Health*, Vol. 51, 1997, pp. 151-159.

[5] J. Kukkonen, L. Partanen, A. Karppinen, J. Ruuskanen, H. Junninen, M. Kolehmainen, H. Niska, S. Dorling, T. Chatterton, R. Foxall and G. Cawley, "Extensive Evaluation of Neural Extensive Evaluation of Neural Network Models for the Prediction of NO2 and PM10 Concentrations, Compared with a Deterministic Modelling System and Measurements in Central Helsinki," *Atmospheric Environment*, Vol. 37, 2003, pp. 4539-4550.

[6] P. Perez, A. Trier and J. Reyes, "Prediction of PM2.5 Concentrations Several Hours in Advance Using Neural Networks in Santiago, Chile," *Atmospheric Environment*, Vol. 34, 2000, pp. 1189-1196.

[7] M. W. Gardner, "The Advantages of Artificial Neural Network and Regression Tree Based Air Quality Models," Ph.D. Dissertation, School of Environmental Sciences, University of East Anglia, Norwich, 1999.

[8] J. Hooyberghs, C. Mensink, G. Dumont, F. Fierens and O. Brasseur, "A Neural Network Forecast for Daily Average PM10 Concentrations in Belgium," *Atmospheric Environment*, Vol. 39, No. 18, 2005, pp. 3279-3289.

[9] J. B Ordieres, E. P. Vergara, R. S. Capuz and R. E. Salazar, "Neural Network Prediction Model for Fine Particulate Matter (PM2.5) on the US-Mexico Border in El Paso (Texas) and Ciudad Juαrez (Chihuahua)," *Environmental Modelling & Software*, Vol. 20, No. 5, 2005, pp. 547-559.

[10] G. Corani, "Air Quality Prediction in Milan: Feed-Forward Neural Networks, Pruned Neural Networks and Lazy Learning," *Ecological Modelling*, Vol. 185, No. 2-4, 2005, pp. 513-529.

[11] C. Lin and C. Lee, "Neural Fuzzy Systems," Prentice Hall, Upper Saddle River, 1996.

[12] M. Hagan and M. Menhaj, "Training Feed-Forward Networks with the Marquardt Algorithm", *IEEE Transactions on Neural Networks*, Vol. 5, 1996, pp. 989-993.

[13] T. Chernichow, A. Piras, K. Imhof, P. Caire, Y. Jaccard, B. Dorizzi and A. Germond, "Short Term Electric Load

Forecasting with Artificial Neural Networks," *Engine Intelligent Systems*, Vol. 2**,** 1996, pp. 85-99.

[14] J. D Farmer and J. J. Sidorowich, "Predicting Chaotic Dynamics, Dynamic Patterns in Complex Systems," In: J. A. S. Kelso, A. J. Mandell and M. F. Shlesinger, Ed., *World Scientific*, 1988, pp. 265-292.

[15] Y. Y. Hong and C. Y. Hsiao, "Locational Marginal Price Forecasting in Deregulated Electricity Markets Using Artificial Intelligence," *IEE Proceedings of Generation Transmission Distribution*, Vol. 149, No. 5, 2002, pp. 621-626.

[16] J. Mitchell and S. Abe, "Fuzzy Clustering Networks: Design Criteria for Approximation and Prediction," *IEICE Transactions on Information and Systems*, Vol. E79D, No. 1, 1996, pp. 63-71.

[17] A. B. Geva, "Hierarchical-Fuzzy Clustering of Temporal-Patterns and its Application for Time-Series Prediction," *Pattern Recognition Letters*, Vol. 20, No. 14, 1999, pp. 1519-1532.

[18] M. Djukanovic, B. Babic, O. J. Sobajic and Y. H. Pao, "24-hour Load Forecasting," *IEE Proceedings – C*, Vol. 140, 1993, pp. 311-318.

[19] J. B. McQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of* 5*th Berkley Symposium on Mathematical Statistics and Probability*, Berkeley, 27 December 1965-7 January 1966, pp. 281-297.

[20] D. J. Kim, Y. W. Park and D. J. Park, "A Novel Validity Index for Determination of the Optimal Number of Clusters," *IEICE Transactions on Information and Systems*, Vol. E84-D, No. 2, 2001, pp. 281-285.

J. Software Engineering & Applications, **2010, 3: 384-390**
doi:10.4236/jsea.2010.34043 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

Exploring Design Level Class Cohesion Metrics

Kuljit Kaur, Hardeep Singh

Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar, India.
Email: kuljitchahal@yahoo.com

Received November 17th, 2009; revised December 15th, 2009; accepted January 25th, 2010.

ABSTRACT

In object oriented paradigm, cohesion of a class refers to the degree to which members of the class are interrelated. Metrics have been defined to measure cohesiveness of a class both at design and source code levels. In comparison to source code level class cohesion metrics, only a few design level class cohesion metrics have been proposed. Design level class cohesion metrics are based on the assumption that if all the methods of a class have access to similar parameter types then they all process closely related information. A class with a large number of parameter types common in its methods is more cohesive than a class with less number of parameter types common in its methods. In this paper, we review the design level class cohesion metrics with a special focus on metrics which use similarity of parameter types of methods of a class as the basis of its cohesiveness. Basically three metrics fall in this category: Cohesion among Methods of a Class (*CAMC*)*, Normalized Hamming Distance* (*NHD*)*, and Scaled NHD* (*SNHD*)*. Keeping in mind the anomalies in the definitions of the existing metrics, a variant of the existing metrics is introduced. It is named NHD Modified* (*NHDM*)*. An automated metric collection tool is used to collect the metric data from an open source software program. The metric data is then subjected to statistical analysis.*

Keywords**:** *Design Metrics, Class Cohesion Metrics, Cohesion among Methods of a Class, Normalized Hamming Distance, Scaled NHD*

1. Introduction

In the object oriented paradigm, cohesion of a class refers to the degree to which members of the class are interrelated. Chidamber and Kemerer defined the first metric to measure cohesiveness of a class [1]. Since then, several class cohesion metrics have been proposed (discussed in the next section). Empirical studies report that class cohesion metrics are useful to assess software design quality [2,3], to predict fault proneness of classes [4-6], and to identify reusable components [7,8].Existing class cohesion metrics mainly fall into two categories – metrics which can be computed at design level (high level) and metrics which can be computed one step later *i.e.* at source code level (low level). Design level class cohesion metrics use the limited amount of information available about a class at this level *i.e.* only the class attributes, and method signatures. Method implementation is not completely defined at design level. So some assumptions are made. Different class cohesion metrics defined at design level are based on different assumptions.

1) One school of thought assumes that the types of method parameters match the types of the attributes accessed by the method. It is further assumed that the set of attribute types accessed by a method is the intersection of this method's parameter types and the set of parameter types of all the methods in the class [9,10].

2) Another school of thought assumes that the set of attribute types accessed by a method is the intersection of the set of this method's parameter types and the set of its class attribute types [11].

In this paper we review the design level class cohesion metrics based on the first assumption. Keeping in mind the anomalies in the definitions of the existing metrics, a variant of the metrics is introduced. The paper is organized as follows: Section 2 reviews the related work. Section 3 explains the existing design level class cohesion metrics and introduces a modified version as well. Section 4 presents the statistical analysis of the data collected from an open source project. Section 5 concludes the paper.

2. Related Work

A number of class cohesion metrics are defined in the low level metrics category [1,12-26]. However, there are only a few proposals for design level class cohesion metrics [9-11].

The metric, named Cohesion among Methods of a Class (CAMC) captures the information about parameter types of methods of a class [9]. A class is cohesive if all methods of the class use the same set of parameter types. Methods which use same type of parameter types are assumed to process related kind of information. CAMC metric values lie in the range [0, 1]. Counsell *et al*. point out some anomalies in definition of this metric, and propose a new metric named Normalized Hamming Distance (NHD) [10]. It is a normalized metric which measures average agreement between each pair of methods on their parameter types. A variant of the NHD metric called Scaled NHD (SNHD) is introduced in the same paper. It addresses shortcomings of both CAMC and NHD, as claimed by the authors [10]. This research finds anomalies in the definitions of NHD and SNHD as well, and proposes a modified version of the NHD metric - NHD modified (NHDM). The NHDM metric gives statistically significant results.

Dallal proposes another metric for measuring cohesion of a class at design level [11]. Similarity based Class Cohesion (SCC) metric is based on the second assumption discussed above. This metric is not analyzed in this paper as the automated tool developed for this research does not support collection of this metric.

3. Design Metrics

This section describes the class cohesion metrics computable with information available at design level. At design level, information regarding name of the class, its attributes (names, and data types), and method signatures is available. Method signature includes name of the method and its parameter list which describes names of the parameters and their data types. A Class does not have a detailed or algorithmic description of its methods available at this level.

3.1 CAMC

The *CAMC* metric measures the extent of intersection of individual method parameter type lists with the parameter type list of all methods in the class [9]. This metric computes the relatedness among methods of a class based upon the parameter list of the methods. It is assumed that methods of a class, having access to similar parameter types, process closely related information.

The CAMC metric uses a parameter-occurrence matrix (PO matrix) that has a row for each method and a column for each data type that appears at least once as the type of a parameter in at least one method in the class. The value in row *i and* column *j* in the matrix is 1 when the *i*th method has a parameter of the *j*th data type and is 0 otherwise. In the original version of the metric [9], the PO matrix has an additional column of all 1s. This column represents the 'self' parameter that corresponds to the type of the class itself which is by default one of the parameters of every method. In this discussion, the original version of the metric is referred to as $CAMC_s$ (Cohesion among methods of a class with 'self' parameter) and metric definition without the 'self' parameter is named as CAMC [10].

The CAMC metric is defined as the ratio of the total number of 1s in the PO matrix to the total size of the matrix.

$$\text{CAMC(C)} = \frac{\sigma}{kl} \quad \text{where } \sigma = \sum_{1}^{k}\sum_{1}^{l} PO[i][j]$$

CAMC suffers from the following anamolies:

1) CAMC gives false positives – the metric gives a non-zero value for a class with no parameter sharing in its methods.

2) CAMC can not differentiate between two classes having same number of 1s but with different patterns of 1s in their PO matrices.

3) Smaller classes take high values for the cohesion metric than the larger classes with same properties.

3.2 NHD

Counsell *et al.* [10] suggested an alternative of CAMC. It is based on the definition of hamming distance. NHD measures agreement between rows in the PO matrix. NHD metric for a class with k methods and l unique parameter types (set obtained from union of parameter types received by all its methods) is defined as:

$$\text{NHD} = \frac{2}{lk(k-1)} \sum_{1}^{k-1} \sum_{j+1}^{k} a(i, j)$$

where $a(i,j)$ is value of the cell at (i,j)th location in the PO matrix. Another easy way to compute NHD is to first find the sum of disagreements between methods for all the parameter types and then subtract it from 1.

$$NHD = 1 - \frac{2}{lk(k-1)} \sum_{1}^{l} c_j (k - c_j)$$

where c_j is the number of 1s in the *j*th column of the PO matrix.

A varaint of NHD (with self parameter), NHD_s can be defined for a PO matrix with an additional column of all 1s.

NHD suffers from the following anomalies:

1) NHD metric also gives false positives. The metric removes the first anomaly of the CAMC for a class with $k = l = 2$. The metrics fails to give correct answer for higher values of k and l (e.g. when $k = l = 3$, and there is no parameter sharing among methods, NHD metric gives a non-zero value).

2) NHD does not give different answers for classes with different properties – metric fails to distinguish a class with no parameter sharing in its methods from a class with substantial amount of parameter sharing in its methods.

3) Class size influences metric value. As size of the class increases, value of the NHD metric also increases (even if the PO matrix gets sparser).

3.3 SNHD

SNHD is the Scaled NHD metric proposed to interpret values of the NHD metric in a more varied range. Proponents of the NHD metric are of the opinion that NHD metric can take values at two extremes: the minimum or the maximum. But they admit that it is not clear as to which of these extremes represents a cohesive class. However without giving any clear explanation they state that classes at both the extremes may be cohesive. They define these extreme values as NHD_{min} and NHD_{max} respectively [10]. SNHD metric value helps to know how close the NHD metric is to the maximum value of the NHD value in comparison to the minimum value. SNHD is defined as follows:

$$SNHD = \begin{cases} 0 & if\ NHD_{\min} = NHD_{\max} \quad and\ \sigma < kl, \\ 1 & if\ \sigma = kl, \\ 2\left(\dfrac{NHD - NHD_{\min}}{NHD_{max} - NHD_{min}}\right) - 1, & otherwise \end{cases}$$

The SNHD metric values lies in the range [-1,1]. SNHD = –1 implies that NHD = NHD_{min}, and SNHD = 1 implies that NHD=NHD_{max}. NHD is closer to its minimum or maximum value depending upon whether SNHD is getting values close to –1 or +1 respectively. A class is considered non-cohesive if SNHD metric value for the class is 0.

$SNHD_s$ is defined by considering the 'self' parameter. SNHD suffers from these Anomalies:

1) Difficult to calculate and interpret.

2) False negatives – SNHD metric gives 0 value for a class with good degree of cohesion.

3.4 NHDM

Keeping in view the anomalies of the cohesion metrics discussed above, this research proposes a variation of the NHD metric. This variat is named as Normalized Hamming Distance Modified (NHDM) metric. The NHD metric ignores the method pairs with zero values in a column of the PO matrix. It counts only those methods pairs which do not agree, and ignores all other method pairs irrespective of whether they agree on a 0 or a 1. NHDM counts the method pairs which agree on a 0, as a disagreement. NHDM for a class with *k* methods and *l* unique parameter types, of all its methods, is defined as:

$$NHDM = 1 - \frac{2}{lk(k-1)} \sum_{1}^{l} (c_j(k - c_j) + \frac{1}{2} z_j(z_j - 1))$$

where c_j is the number of ones and z_j is the number of zeroes in the *j*th column of the PO matrix for the class.

Similarly NHDMs is defined by including the 'self' parameter in the PO matrix.

This metric removes the anomalies present in the defintion of CAMC, NHD, and SNHD metrics. NHDM gives correct results. It gives different results for classes with different properties. NHDM metric values are independent of the class size.

4. Data Analysis

Cohesion metrics discussed above are collected from an open source software system available at www.sourceforge.net. The software is a JAVA based charting library, and it consists of 884 classes. For automated collection of metrics, a tool CohMetric is developed.

4.1 Descriptive Analysis

Histograms in **Figures 1** to **4** show metrics distributions. **Table 1** presents the descriptive statistics. It can be observed that majority of the CAMC metric values lie close to 0 (see **Figure 1**). On average a class's cohesion value is 0.21. NHD metric takes values in a higher range (**Figure 2**). Average NHD metric value is 0.66. SNHD is 0 for maximum of the classes. Its values lie more on the

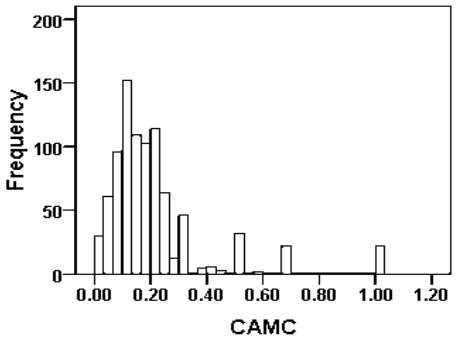


Figure 1. Distribution of CAMC metric

Table 1. Descriptive statistics for cohesion metrics

Metric	Average	Std Dev	Metric	Average	Std Dev
CAMC	0.21	0.18	$CAMC_s$	0.48	0.21
NHD	0.66	0.21	NHD_s	0.81	0.12
SNHD	-0.43	0.51	$SNHD_s$	0.63	0.42
NHDM	0.05	0.16	$NHDM_s$	0.38	0.22

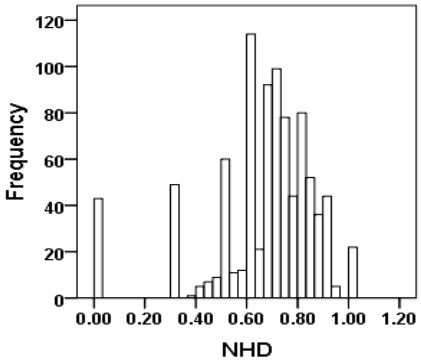


Figure 2. Distribution of NHD metric

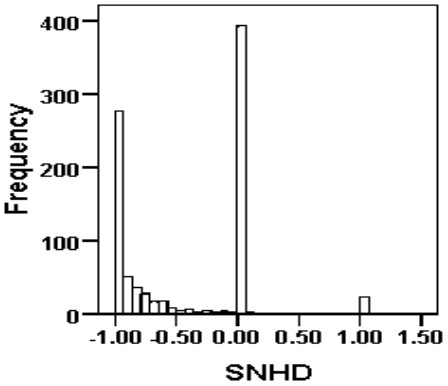


Figure 3. Distribution of SNHD metric

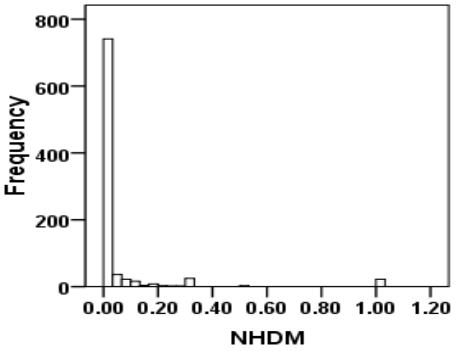


Figure 4. Distribution of NHDM metric

left side of 0 which implies that majority of the classes has NHD more close to NHD_{min} than NHD_{max}. Average SNHD for a class is –0.43 and standard deviation is also very high (**Figure 3**). NHDM takes very low values (**Figure 4**). For majority of the classes it is 0. Its average value is just 0.05. As earlier stated, it may be due to the reason that it does not give false positives.

4.2 Metric Variants

Variants of these cohesion metrics are defined on the basis of the assumption that all the methods of a class by default receive the class type itself (self) as one of the parameter types. $CAMC_s$, NHD_s, $SNHD_s$ and $NHDM_s$ are defined as variants of CAMC, NHD, SNHD, and NHDM respectively. Cohesion metrics which consider the 'self' parameter are expected to give higher values as the class methods agree on at least one parameter type. **Table 1** gives a comparison of averages of cohesion metrics and their variants. All the metrics in this category (which consider self parameter type) have higher averages than their counterparts. The observation is that metric variants, which consider 'self' as one of the parameter types, take values in higher range.It is also confirmed by the descriptive analysis of these metrics as shown in **Figures 5** to **8**. It is worth noting that $SNHD_s$ takes values in the range from 0 to 1 more frequently, in contrast to SNHD which takes values in the range from 0 to –1. It

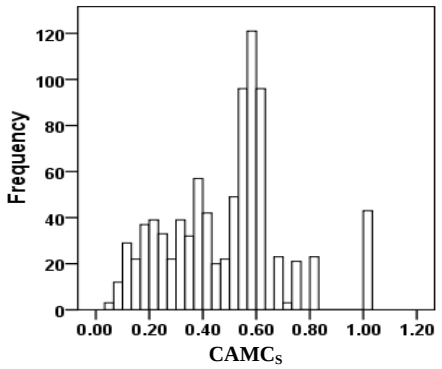


Figure 5. Distribution of $CAMC_s$ metric

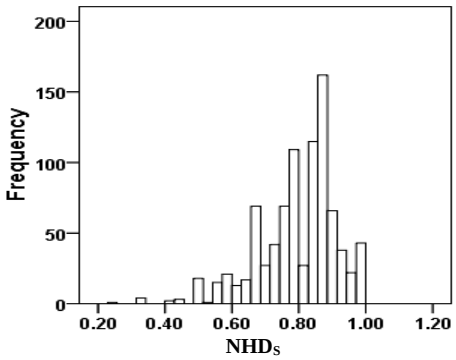


Figure 6. Distribution of NHD_s metric

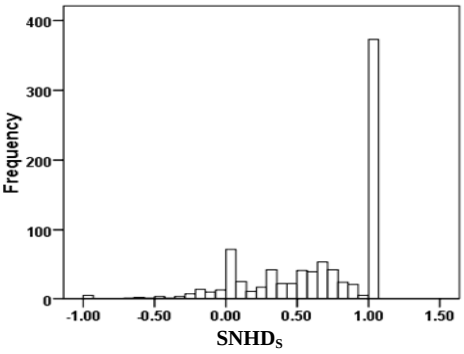


Figure 7. Distribution of $SNHD_s$ metric

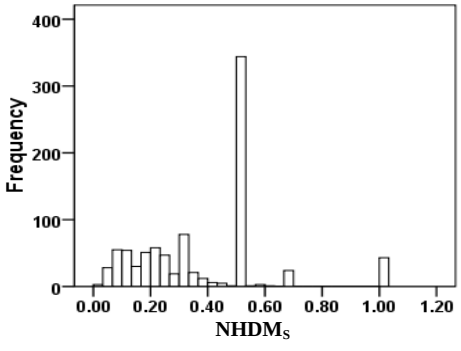


Figure 8. Distribution of $NHDM_s$ metric

implies that a class whose NHD value is more close to NHD_{max} is more cohesive.

4.3 Size Independence

Figures 9 to **12** present the relation between cohesion metrics and class size (measured in terms of number of methods). CAMC metric value is higher for small classes and is lower for large classes. NHD takes large values for classes with larger number of methods. This is in line with the earlier findings about these two metrics [10]. As shown in **Figure 11**, SNHD is close to 1 for some comparatively small classes. For larger classes, SNHD lies in the range [-1, 0]. NHDM takes values near 0 for most of the classes. However small classes have metric value in the higher range. However if size of the parameter occurrence (PO) matrix is taken into consideration then it is found that it does not have significant correlation with any of the metrics (see **Table 2**). Here *l* represents the number of parameter types, k is the number of methods of the class, and *lk* is the size of the parameter occurrence matrix. This result is unlike the previous studies on these metrics [10,27].

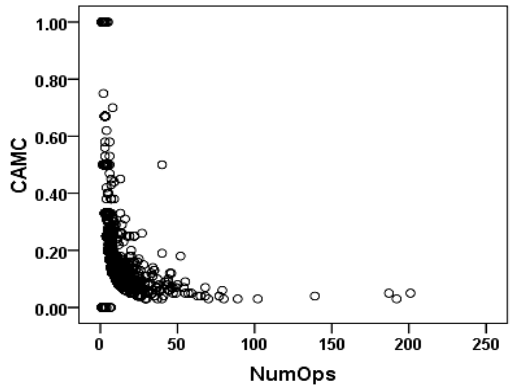


Figure 9. Scatter diagram of CAMC and class size

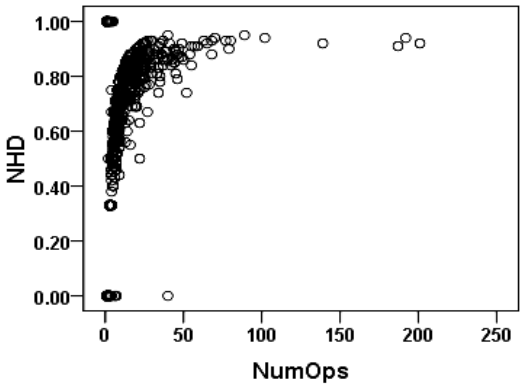


Figure 10. Scatter diagram of NHD and class size

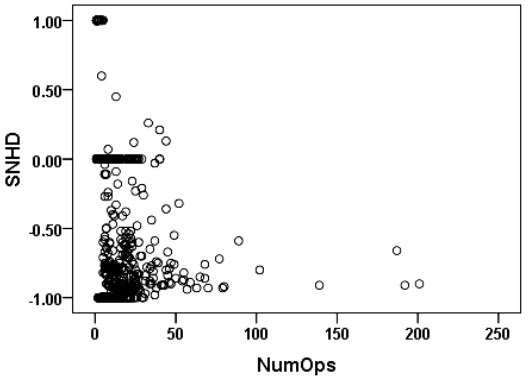


Figure 11. Scatter diagram of SNHD and class size

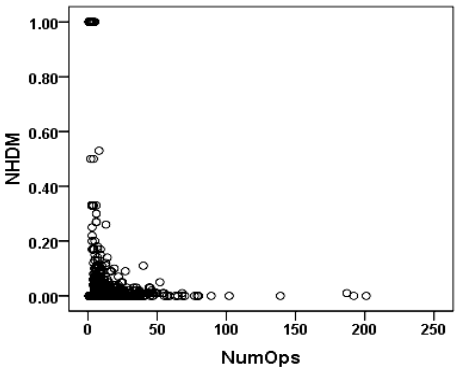


Figure 12. Scatter diagram of NHDM and class size

4.4 Metrics Inter-Dependencies

The parametric Pearson's correlation coefficient between each pair of cohesion metrics is given in **Table 3**. All the correlation figures are significant at p = 0.01 level. Metric variants such as $CAMC_s$, NHD_s, $SNHD_s$, and $NHDM_s$ are moderately correlated with their counterparts. NHD and NHD_s have the highest correlation coefficient in this category. NHDM and CAMC are strongly correlated. Similar is the case for their variants $NHDM_s$ and $CAMC_s$. SNHD is moderately correlated with $NHDM_s$ and $CAMC_s$. Unlike the previous studies, the correlation analysis for this data set does not show any significant correlation in NHD and CAMC [10,27]. However the scatter plot of values for these two metrics shows a negative trend. CAMC and NHD show a negative relationship in the scatter diagram given in **Figure 13.** CAMC is very low for the classes for which NHD is very high. On average the NHD metric takes values in higher range.This implies that this metric pair does not have a linear covariation.

Principal Component Analysis (PCA) is used to identify the metrics measuring orthogonal dimensions. Rotated principal components are obtained using the varimax rotation technique. Three principal components are extracted which capture 93.28% of the data set variance (shown in **Table 4**). Metrics with significant loading coefficients in a particular dimension are highlighted in bold. An analysis of the table shows that $NHDM_s$ and $CAMC_s$ and SNHD contribute significantly to the first

Table 2. Correlation in cohesion metrics and size

	CAMC	$CAMC_s$	NHD	NHD_s	SNHD	$SNHD_s$	NHDM	$NHDM_s$
l	**-.222**	**-.696**	**.337**	**.003**	**-.356**	**-.539**	**-.128**	**-.645**
k	**-.307**	**-.520**	**.350**	**.219**	**-.071**	**-.179**	**-.177**	**-.429**
lk	**-.158**	**-.357**	**.210**	**.138**	**.067**	**-.223**	**-.075**	**-.298**

Table 3. Correlation analysis among metrics

	CAMC	$CAMC_s$	NHD	NHD_s	SNHD	$SNHD_s$	NHDM
$CAMC_s$	**0.575**						
NHD	**-0.267**	**-0.542**					
NHD_s	**-0.372**	**-0.024**	**0.623**				
SNHD	**0.341**	**0.654**	**-0.043**	**0.334**			
$SNHD_s$	**-0.253**	**0.347**	**0.271**	**0.678**	**0.478**		
NHDM	**0.854**	**0.466**	**0.122**	**0.107**	**0.403**	**-0.043**	
$NHDM_s$	**0.456**	**0.962**	**-0.356**	**0.249**	**0.726**	**0.520**	**0.480**

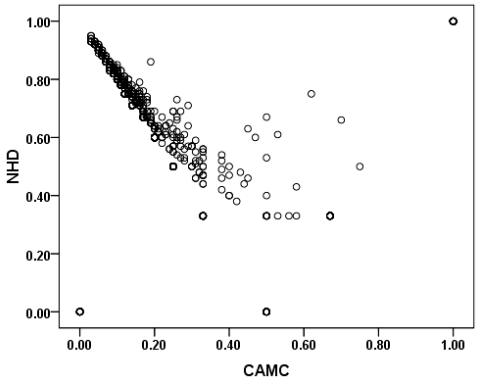


Figure 13. Scatter diagram shows correlation in CAMC and NHD metrics

Table 4. Principal Components Matrix

	PC1	PC2	PC3
Eigen Value	**3.72**	**2.39**	**1.35**
Percent	**46.45**	**29.93**	**16.90**
Comm. percent	**46.45**	**76.38**	**93.28**
CAMC	**0.25**	**-0.32**	0.90
$CAMC_s$	0.91	**-0.27**	**0.30**
NHD	**-0.39**	0.89	**0.11**
NHD_s	**0.27**	0.90	**-0.13**
SNHD	0.84	**0.21**	**0.27**
$SNHD_s$	**0.66**	**0.59**	**-0.31**
NHDM	**0.24**	**0.15**	0.95
$NHDM_s$	0.95	**-0.01**	**0.25**

dimension: PC1. $SNHD_s$ is moderately significant in two dimensions: PC1 and PC2. NHD and NHD_s both load significantly on PC2. NHDM and CAMC both load significantly on PC3.

It is worth mentioning here that NHDM and $NHDM_s$ have the maximum variance among all the metrics in this analysis. So metrics measuring different dimensions are:

PC1: $NHDM_s$, $CAMC_s$

PC2: NHD, NHD_s

PC3: NHDM, CAMC

5. Conclusions

Cohesion is one of the important design properties to realize a quality software product. Many empirical studies exist which relate the cohesion deign property with other properties of interest such as maintainability, reusability, and reliability. Several metrics have been proposed to compute cohesion at class level in object oriented systems. In this paper design level cohesion metrics such as CAMC, NHD, SNHD have been investigated using empirical data. In view of the anomalies present in the existing metrics' definitions, a modified version of the NHD metric is proposed and is named as NHDM (NHD Modified) ss. Statistical analysis of the metrics data shows that CAMC and NHD are influenced by the size of class (measured in terms of number of methods). None of the studied metrics correlates with the size of the Parameter Occurence matrix (PO matrix) of the class. Principal Component Analysis of the data shows that NHDM and CAMC both give similar results but NHDM has more variation in its values. Similar is the case for $NHDM_s$ and $CAMC_s$. SNHD or $SNHD_s$ does not contribute significantly to any dimension. NHD and NHD_s are not significantly related to any of the other metrics.

REFERENCES

[1] P. Chidamber and C. Kemerer, "Towards a Metrics Suite for Object Oriented Design," *Proceedings of* 6*th ACM Conference on Object Oriented Programming, Systems, Languages and Applications*, Phoenix, Arizona, 1991, pp. 197-211.

[2] L. Briand, J. Wust, J. Daly and D. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object Oriented Systems," *Journal of Systems and Software*, Vol. 51, No. 3, 2000, pp. 245-273.

[3] J. Bansiya and C. Davis, "A Hierarchical Model for Object Oriented Quality Assessment," *IEEE Transactions on Software Engineering*, Vol. 28, No. 1, 2002, pp. 4-17.

[4] T. Gyimothy, R. Ferenc and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Transactions on Software Enineering*, Vol. 31, No. 10, 2005, pp. 897-910.

[5] Z. Zhou and H. Leung, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low

Severity Faults," *IEEE Transactions on Software Engineering*, Vol. 32, No. 10, 2006, pp. 771-789.

[6] M. Marcus and D. Poshyvanyk, "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented System," *IEEE Transactions on Software Engineering*, Vol. 34, No. 2, 2008.

[7] J. Lee, S. Jung, S. Kim, W. Jang and D. Ham,"Component Identification Method with Coupling and Cohesion," *Proceedings of the Eighth Asia-Pacific Software Engineering Conference*, December 2001, pp. 79-86.

[8] G. Gui and D. Scott, "Measuring Software Component Reusability by Coupling and Cohesion Metrics," *Journal of Computers*, Vol. 4, No 9, Academy Publishers, 2009, pp. 797-805.

[9] J. Bansiya, L. Etzkorn, C. Davis and W. Li, "A Class Cohesion Metric for Object Oriented Designs," *Journal of Object Oriented Programming*, Vol. 11, No. 8, 1999, pp. 47-52.

[10] S. Counsell, S. Swift and J. Crampton, "The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design," *ACM Transactions on Software Engineering and Methodology*, Vol. 15, No. 2, 2006, pp. 123-149.

[11] J. Dallal, "A Design-Based Cohesion Metric for Object-Oriented Classes," *Proceedings of the International Conference on Computer and Information Science and Engineering*, 2007, pp. 301-306.

[12] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," *Journal of Systems and Software*, Vol. 23, No. 2, 1993, pp. 111-122.

[13] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, Vol. 20, 1994, pp. 476-493.

[14] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems," *Proceedings of International Symosium on Applied Corporate Computing*, 1995.

[15] J. Bieman and B. Kang, "Cohesion and Reuse in an Object-Oriented System," *Proceedings of the* 1995 *Symposium on Software Reusability*, ACM Press, 1995, pp. 259-262.

[16] B. Henderson-Sellers, L. Constantine and I. Graham, "Coupling and Cohesion (towards a Valid Metrics Suite for Object-Oriented Analysis and Design)," *Object Oriented Systems*, Vol. 3, 1996, pp. 143-158.

[17] L. Briand, J. Daly and J. Wust, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Engineering*, Vol. 3, No. 1, 1998, pp. 65-117.

[18] H. Chae, Y. Kwon and D. Bae, "A Cohesion Measure for Object-Oriented Classes," *Software Practice and Experience*, Vol. 30, No. 12, 2000, pp. 1405-1431.

[19] Z. Chen, Y. Zhou and B. Xu, "A Novel Approach to Measuring Class Cohesion Based on Dependence Analysis," *Proceedings of the International Conference on Software Maintenance*, 2002, pp. 377-384.

[20] L. Badri and M. Badri, "A Proposal of a New Class Cohesion Criter Ion: An Empirical Study," *Journal of Object Technology*, Vol. 3, No. 4, 2004.

[21] J. Wang, Y. Zhou, L. Wen, Y. Chen, H. Lu and B. Xu, "DMC: A More Precise Cohesion Measure for Classes," *Information and Software Technology*, Vol. 47, No. 3, pp. 176-180, 2005.

[22] C. Bonja and E. Kidanmariam, "Metrics for Class Cohesion and Similarity between Methods," *Proceedings of the* 44*th Annual Southeast Regional Conference*, ACM Press, New York, 2006, pp. 91-95.

[23] G. Cox, , L. Etzkorn and W. Hughes, "Cohesion Metric for Object-Oriented Systems Based on Semantic Closeness from Disambiguity," *Applied Artificial Intelligence*, Vol 20, No. 5, 2006, pp. 419-436.

[24] L. Fernández and R. Peña, "A Sensitive Metric of Class Cohesion," *International Journal of Information Theories and Applications*, Vol. 13, No. 1, 2006, pp. 82-91.

[25] S. Makela and V. Leppanen, "Client Based Object Oriented Cohesion Metrics," 31*st Annual International Computer Software and Applications Conference*, Vol. 2, 2007, pp. 743-748.

[26] A. Marcus and D. Poshyvanyk, "The Conceptual Cohesion of Classes," *Proceedings of* 21*st IEEE International Conference on Software Maintenance*, 2005, pp. 133-142.

[27] J. Dallal and L. Briand, "An Object-Oriented High-Level Design-Based Class Cohesion Metric," Simula Technical Report (2009-1), Version 2, Simula Research Laboratory, 2009.

J. Software Engineering & Applications, 2010, 3: 391-403
doi:10.4236/jsea.2010.34044 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

DSPs/FPGAs Comparative Study for Power Consumption, Noise Cancellation, and Real Time High Speed Applications

Alon Hayim, Michael Knieser, Maher Rizkalla

Department of Electrical and Computer Engineering, Indiana University Purdue University Indianapolis, Indianapolis, USA.
Email: mrizkall@iupui.edu, mrizkall@yahoo.com

Received December 24th, 2009; revised January 6th, 2010; accepted February 3rd, 2010.

ABSTRACT

*Adaptive noise data filtering in real-time requires dedicated hardware to meet demanding time requirements. Both DSP processors and FPGAs were studied with respect to their performance in power consumption, hardware architecture, and speed for real time applications. For testing purposes, real time adaptive noise filters have been implemented and simulated on two different platforms, Motorola DSP*56303 *EVM and Xilinx Spartan III boards. This study has shown that in high speed applications, FPGAs are advantageous over DSPs with respect of their speed and noise reduction because of their parallel architecture. FPGAs can handle more processes at the same time when compared to DSPs, while the later can only handle a limited number of parallel instructions at a time. The speed in both processors impacts the noise reduction in real time. As the DSP core gets slower, the noise removal in real time gets harder to achieve. With respect to power, DSPs are advantageous over FPGAs. FPGAs have reconfigurable gate structure which consumes more power. In case of DSPs, the hardware has been already configured, which requires less power consumption*? *FPGAs are built for general purposes, and their silicon area in the core is bigger than that of DSPs. This is another factor that affects power consumption. As a result, in high frequency applications, FPGAs are advantageous as compared to DSPs. In low frequency applications, DSPs and FPGAs both satisfy the requirements for noise cancelling. For low frequency applications, DSPs are advantageous in their power consumption and applications for the battery power devices. Software utilizing Matlab, VHDL code run on Xilinix system, and assembly running on Motorola development systems, have been used for the demonstration of this study.*

Keywords**:** *Four Quadrant* (4*Q*) *Converter, Interlacing, Traction Systems, Power Quality Analysis*

1. Introduction

The performance of real-time data processing is often limited to the processing capability of the system. Therefore, evaluation of different digital signal processing platforms to determine the most efficient platform is an important task. There have been many discussions regarding the preference of Digital Signal processors (DSPs) or Field Programmable Gate Arrays (FPGA) in real time noise cancellation. The purpose of this work is to study features of DSPs and FPGAs with respect to their power consumption, speed, architecture and cost. DSP is found in a wide variety of applications, such as filtering, speech recognition, image enhancement and data compression, neural networks, as well as analog linear-phase filters. Signals from the real world received in analog form, then discretely sampled for a digital computer to understand and manipulate. There are many advantages of hardware that can be reconfigured with different programming. Reconfigurable hardware devices offer both the flexibility of computer software, and the ability to construct custom high performance computing circuits. In space applications, it may be necessary to install new functionality into a system, which may have been unforeseen. For example, satellite applications need to adjust to changing operation requirements. With a reconfigurable chip, functionality that is not normally predicted at the outset can be uploaded to the satellite when needed. To test the adaptive noise cancelling, the least mean square (LMS) approach has been used. Besides the standard LMS algorithm, the modified algorithms that are proposed by Stefano [1] and by Das [2] have been implemented for the noise cancellation approach, giving the opportunity of comparing both platforms with respect

to their speed, noise, architecture, cost, and power.

2. Adaptive Filter Design on Motorola DSP56300

Adaptive filters have the ability to adjust their own parameters and coefficients automatically. Hence, their design requires little or no prior knowledge of the input signal or noise characteristics of the system. Adaptive filters have two inputs, x(n) and d(n), which are usually correlated in some manner. **Figure 1** gives the basic concept of the adaptive filter.

The filter's output y(n), which is computed with the parameter estimates, is compared with the input signal d(n). The resulting prediction error e(n) is fed back through a parameter adaption algorithm that produces a new estimate for the parameters and as the next input sample is received, a new prediction error can be generated. The adaptive filter features minimum prediction error. Two aspects of the adaptive filter are its internal structure and adaptation algorithm. Its internal structure can be either that of a nonrecursive (FIR) filter or that of a recursive (IIR) filter. An adaptation algorithm can be divided into two major classes; gradient algorithms and nongradient algorithms. A gradient algorithm is used to adjust the parameters of the FIR filter. The least mean square (LMS) algorithm is the most widely applied gradient algorithm. This adjusts the filter's parameters to minimize the mean-square error between the filter's output y(n) and the desired response input d(n) [3]. When an adaptive filter is implemented on the DSP56300 processer, address pointer to mimic FIFO (First-In-First-Out)-like shifting of the RAM data, modulo addressing capability to provide wrap around data buffers, multiply/accumulate (MAC) instruction top both multiply two operands and add the product to a third operand in a single instruction cycle, data move in parallel with the MAC instructions to keep the multiplier running at 100% capacity and Repeat Next Instruction (REP) to provide compact filter code are being used by the processor. The processor's capability to perform modulo addressing allows an address register (Rn) value to be incremented (or decremented) and yet remain within an address range of size L, where L is defined by a lower and an upper address boundary. For the adaptive FIR filter, L is the number of coefficients (taps). The value L-1 is stored in the processor's Modifier Register (Mn). The upper address boundary is calculated by the processor and is not stored in a register. When modulo addressing is used, the Address Register (Rn) points to a modulo data buffer located in X-Memory and/or Y-Memory. The address pointer (Rn) is not required to point at the lower address boundary; it can point anywhere within the defined modulo address range L. If the address pointer increments past the upper address boundary (base address plus L-1 plus 1), it will wrap around to the base address. Modulo Register M1 is programmed to the value NTAPS-1 (modulo NTAPS). Address Register R1 is programmed to point to the state variable modulo buffer located in X-Memory. Modulo Register M4 is programmed to the value NTAPS-1. Address Register R4 is programmed to point to the coefficient buffer located in Y-Memory. Given that the FIR filter algorithm has been executing for some time and is ready to process the input sample x(n) in the Data ALU input Register X0, the address in R4 is the base address (lower boundary) of the coefficient buffer. The address in R1 is M, where M is greater than or equal to the lower boundary of X-Memory address and less than or equal to the upper boundary of X-Memory address. The X-Memory map for the filter states, the Y-Memory map for the coefficients, and the contents of the processor's A and B Accumulators and Data ALU Input Registers X0, X1, Y0 and Y1 are shown in the **Figure 2**. The CLR instruction clears the A-Accu-

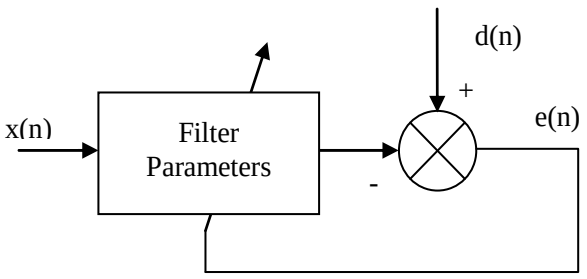


Figure 1. Basic concept of the adaptive filter

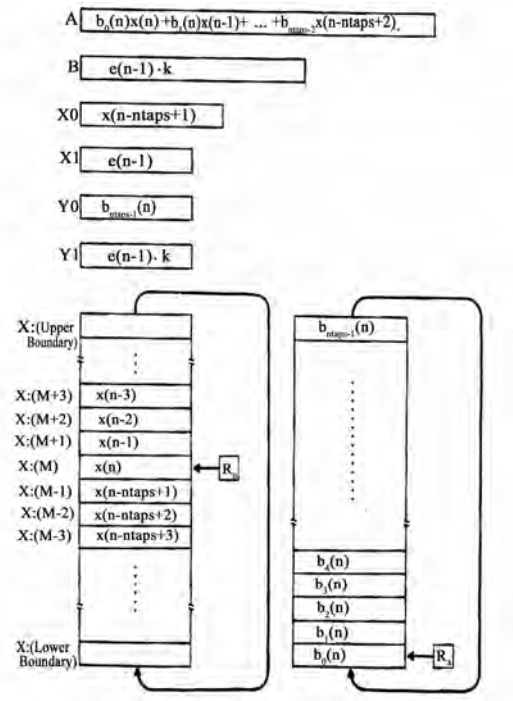


Figure 2. Memory map and data registers after last MAC instruction

mulator and simultaneously moves the input sample x(n) from the Data ALU's Input Register X0 to the X-Memory location pointed to by address register R1, and moves the first coefficient from the Y-Memory location pointed to by address register R4 to the Data ALU's Input Register Y0. Both Address Registers R1 and R4 are automatically incremented by one at the end of the CLR instruction (post-incremented). The REP instruction regulates execution of NTAPS-1 iteration of the MAC instruction. The MAC instruction multiplies the filter state variable X0 by the coefficient in Y0, adds the product to the A-Accumulator and simultaneously moves the next state variable from the X-Memory location pointed to by the Address Register R1 to the Input Register X0, and moves the next coefficient from the Y-Memory location pointed to by Address Register R4 to Input Register Y0. Both Address Registers R1 and R4 are automatically incremented by one at the end of the MAC instruction (post-incremented).

During the execution of the filter algorithm, Address Register R4 is post incremented to a total of NTAPS times; once in conjunction with the CLR instruction and NTAPS-1 times (due to the REP instruction) in conjunction with the MAC instruction. Since the modulus for R4 is NTAPS and R4 is incremented NTAPS times, the address value in R4 wraps around and points to the coefficient buffer's lower boundary location [3]. Also Address Register R1 is post incremented to a total NTAPS times; once in conjunction with the CLR instruction and NTAPS-1 times (due to the REP instruction) in conjunction with the MAC instruction. Also at the beginning of the algorithm, the input sample x(n) is moved from the Data ALU Input Register X0 to the X-Memory location pointed to by R1. Since the modulus for R1 is NTAPS and R1is incremented NTAPS times, the address value in R1 wraps around and points to the state variable buffer's X-Memory location M. The MACR instruction calculates the final tap of the filter algorithm and performs convergent rounding of the result. The data move portion of this instruction loads the input sample x(n) into the B-Accumulator. At the end of the MACR instruction, the accumulator contains the filter output sample y(n) as shown in **Figure 3**.

The two Move instructions transfers the loop gain K to the data register Y1 and the error sample e(n) to the Data Input Register X1. The first MOVE instruction in the "do loop" transfers the parameter $b_i(n)$ to the A-Accumulator and the filter state x(n-i) to the Data Input Register X0. Address Register R1 is incremented by one to point to the next filter state. The MAC instruction multiplies the filter state, in X0, by the product of the loop gain and the error sample, in Y1, and adds the product to the A-Accumulator. The result in the A-Accumulator is the updated parameter $b_i(n+1)$. The second Move instruction in the "do loop" transfers the parameter $b_i(n+1)$ to the Y-Memory location pointed to by the Address Register R4. R4 is incremented by one to point to the next filter parameter as shown in **Figure 4**. The LUA instruction decrements R1 by one, and R1 then points to the state variable buffer's X-Memory location M-1. When the algorithm is executed, a new (next) input sample x(n+1) will overwrite the value in X-Memory location M-1. Thus FIFO-like shifting of the filter state variables is accomplished by adjusting the R1 address pointer as shown in **Figure 5**.

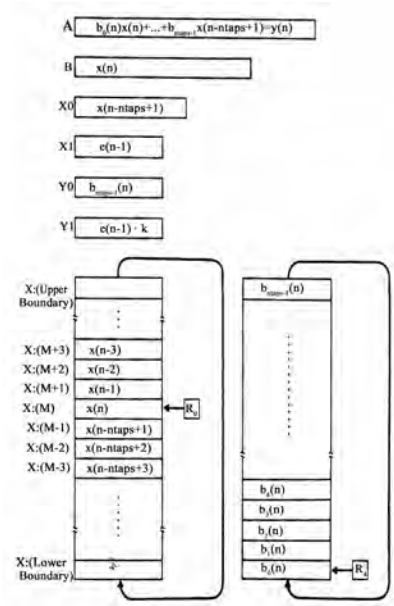


Figure 3. Memory map and data registers after MACR instruction

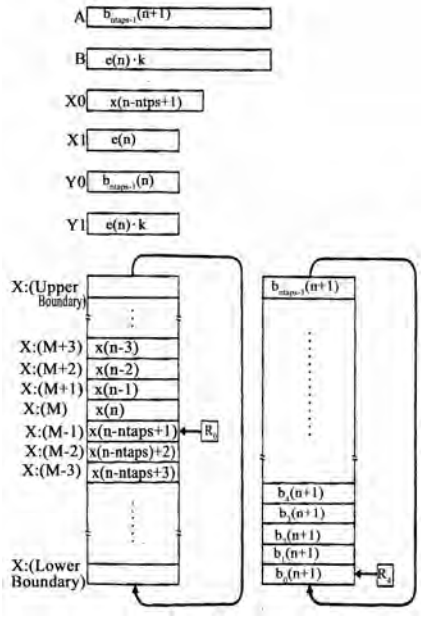


Figure 4. Memory map and data registers after last pass of do loop

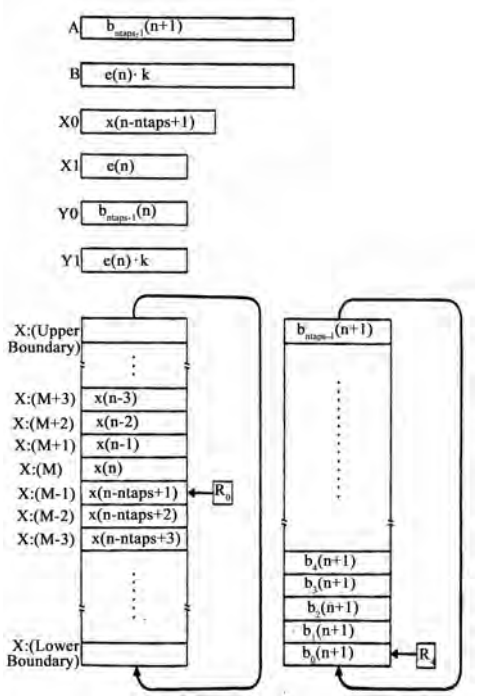


Figure 5. Memory map and data registers after LUA instruction

Consider the problem of finding the linear minimum mean square estimate (LMMSE) of a zero-mean signal vector, *S*, from a noisy zero-mean data vector, $X = S + N$, where *N* denotes the additive noise vector. A LMMSE of *S* is given in Equation (1), where *A* denotes a matrix of filter coefficients as given in Equation (2).

$$S = A \cdot X \tag{1}$$

$$S = C_{SS}\left(C_{SS} + C_{nn}\right)^{-1} X \tag{2}$$

Here, C_{SS} and C_{nn} denote the covariance matrices of signal and noise, respectively. Notice that if *X* has a non-zero mean vector, μ, Equation e becomes:

$$S = \mu + C_{SS}\left(C_{SS} + C_{nn}\right)^{-1}\left(X - \mu\right) \tag{3}$$

For point-wise processing of a non-stationary signal of a local mean, μ_S, and local variance, σ_S^2, and the noise to be zero-mean, white with a local variance, σ_n^2, the point-wise LMMSE will be given by:

$$S = \mu_S + \left[\frac{\sigma_S^2}{\sigma_S^2 + \sigma_n^2}\right]\left[x - \mu_S\right] \tag{4}$$

σ_n^2 is constant, while σ_S^2 and μ_S vary with the time index, *k*. Thus the filtered estimate at time, *k* can be written as:

$$S = \mu_S(k) + \left[\frac{\sigma_S^2(k)}{\sigma_S^2(k) + \sigma_n^2}\right]\left[x(k) - \mu_S(k)\right] \tag{5}$$

where $\mu_S(k)$ and $\sigma_S^2(k)$ denote the time varying estimates of local mean and local variance of $S(k)$. An improved version of Lee's adaptive wiener filter has been proposed by Das [4]. The main contributions of this algorithm include a better technique for estimation of noise variance, and incorporation of a data window for adaptive filtering. Lee's adaptive wiener filter suffers from two major drawbacks. First, it requires prior knowledge of noise power and second, its performance deteriorates when the signal-to-noise ratio (SNR) is low and noise power is imprecisely known. The improved wiener filter incorporates two modifications. First, the de-noising performance of the filter is improved by introducing a non-rectangular window to process weighted data samples and second, a scheme for online estimation of noise power is incorporated which is based on analyzing the power spectral density, $S(\omega)$, of the data. Assuming that the observed data consists of predominantly low-frequency signal components and additive white noise, then $S(\omega)$ can be modeled as a sum of the spectral density of the signal and a constant, σ_n^2, which represents the variance of noise. The estimated σ_n^2 is the average value of the high-frequency section of $S(\omega)$ [2]. The improved wiener filter can be done in a fashion similar to that of Lee's wiener filter, but Equation (2) now takes the form $S = AWX$, where *A* denotes a matrix of filter coefficients, and *W* is a (diagonal) data weighting matrix. The LMMSE of *S* is now given by Equation (6), where $X_W = WX$, and similarly, the point-wise LMMSE is given by

$$S = C_{SS}\left(C_{SS} + C_{nn}\right)^{-1} X_W \tag{6}$$

$$S = \mu_S + \left[\frac{\sigma_S^2}{\sigma_S^2 + \sigma_n^2}\right]\left[X_W - \mu_S\right] \tag{7}$$

3. FPGAs Adaptive Filter Design

The efficient realization of complex algorithms on FPGAs requires a familiarity with their specific architectures. The modifications needed to implement an algorithm on an FPGA and also the specific architectures for adaptive filtering and their advantages are given below.

3.1 FPGA Realization Issues

FPGAs are ideally suited for the implementation of adaptive filters. However, there are several issues that need to be addressed. When performing software simulations of adaptive filters, calculations are normally carried out with floating point precision. Unfortunately, the resources required of an FPGA to perform floating point arithmetic are normally too large to be justified. Another concern is the filter tap itself. Numerous techniques have been devised to efficiently calculate the convolution operation when the filter's coefficients are fixed in advance. For an

adaptive filter whose coefficients change over time, these methods will not work or need to be modified significantly [5]. The reconfigurable filter tap is the most important issue for high performance adaptive filter architecture, and as such it will be discussed at length.

3.2 Finite Precision Effects

Although computing floating point arithmetic in FPGA is possible, it is usually accomplished with the inclusion of custom floating point units, which are costly in terms of logic resources. Therefore, a small number of floating point units can be used in the entire design, and must be shared between processes. This does not take full advantage of the parallelization that is possible with FPGAs and is therefore not the most efficient method. All calculation should therefore be mapped into fixed point only, but this can introduce some errors. The main errors in DSP include ADC quantization error, coefficient quantization error, overflow error caused impermissible word length, and round off error. The other three issues will be addressed later.

3.2.1 Scale Factor Adjustment

A suitable compromise for dealing with the loss of precision when transitioning from a floating point to a fixed-point representation is to keep a limited number of decimal digits. Normally, two to three decimal places is adequate, but the number required for a given algorithm to converge must be found through experimentation. When performing software simulations of a digital filter for example, it is determined that two decimal places is sufficient for accurate data processing. This can easily be obtained by multiplying the filter's coefficients by 100 and truncating to an integer value. Dividing the output by 100 recovers the anticipated value. Since multiplying and dividing be powers of two can be done easily in hardware by shifting bits, a power of two can be used to simplify the process. In this case, one would multiply by 128, which would require seven extra bits in hardware. If it is determined that three decimal digits are needed, then ten extra bits would be needed in hardware, while one decimal digit requires only four bits. For simple convolution, multiplying by a preset scale and then dividing the output by the same scale has no effect on the calculation. For a more complex algorithm, there are several modifications that are required for this scheme to work [6]. The first change needed to maintain the original algorithm's consistency requires dividing by a scale constant any time and previously scaled values are multiplied together. Consider, for example, the values *a* and *b* and the scale constant *s*, the scaled integer values are represented by $s \cdot a$ and $s \cdot b$. To multiply theses values requires dividing by *s* to correct for the s^2 term that would be introduced and recover the scaled product $a \cdot b$.

$$\left(s \cdot a \times s \cdot b\right) \Big/ s = s \cdot ab \tag{8}$$

Likewise, division must be corrected with a subsequent multiplication. It should now be evident why a power of two is chosen for the scale constant, since multiplication and division by power of two results in simple bit shifting. Addition and subtraction require no additional adjustment. The aforementioned procedure must be applied with caution, however, and does not work in all circumstances. While it is perfectly legal to apply to the convolution operation of a filter, it may need to be tailored for certain aspects of a given algorithm. Consider the tap-weight adaptation equation for the LMS algorithm in Equation (9).

$$\hat{w}(n+1) = \hat{w}(n) + \mu \cdot u(n)\dot{e}(n) \tag{9}$$

where μ is the learning rate parameter; its purpose is to control the speed of the adaptation process. The LMS algorithm is convergent in the mean square provided in Equation (10).

$$0 < \mu < \frac{2}{\lambda_{MAX}\Pi} \tag{10}$$

where $\lambda_{MAX}\Pi$ is the largest eigenvalue of the correlation matrix R_x of the filter's input. Typically this is a fraction value and its product with the error term has the effect of keeping the algorithm from diverging. If μ is blindly multiplied by some scale factor and truncated to a fixed-point integer, it will take on a value greater than one. The affect will be to make the LMS algorithm diverge, as its inclusion will now amplify the added error term. The heuristic adopted in this case is to divide by the inverse value, which will be greater than one. Similarly, division by values smaller than one should be replaced by multiplication with its inverse. The outputs of the algorithm will then need to be divided by the scale to obtain the true output. The following algorithm describes the fixed point conversion:

Determine Scale

Through simulations, find the needed accuracy (# decimal places).

Scale = accuracy rounded up to a power of two.

Multiply all constants by scale

- Divide by scale when two scaled values are multiplied.
- Multiply by scale when two scaled values are divided.

Replace

For multiplication by values less than 1

- Replace with division by the reciprocal value.

Likewise, for division by values less than 1

Replace with multiplication by the reciprocal value.

3.2.2 Training Algorithm Modification

The training algorithms for the adaptive filter need some minor modifications in order to converge for a fixed-point implementation. Changes to the LMS weight update equation were discussed in the previous section.

Specifically, the learning rate μ and all other constants should be multiplied by the scale factor. When μ is adjusted it takes the form in Equation (11). With μ modification weight update Equation (11) can be modified as in Equation (12).

$$\hat{\mu} = \frac{1}{\mu} \cdot scale \tag{11}$$

$$\hat{w}(n+1) = \hat{w}(n) + \frac{u(n)\dot{e}(n)}{\hat{\mu}} \tag{12}$$

The direct form FIR structure has a delay that is determined by the depth of the output adder tree, which is dependent on the filter's order. The transposed FIR, on the other hand, has a delay of only one multiplier and one adder, regardless of the filter length. It is therefore advantageous to use the transposed form for FPGA implementation to achieve maximum bandwidth. **Figure 6** shows the direct and **Figure 7** shows the transposed FIR structures for a three tap filter. The relevant nodes have been labeled A, B and C for a data flow analysis. Each filter has three coefficients, and are labeled $h_0[n]$, $h_1[n]$ and $h_2[n]$. The coefficients' subscript denotes the relevant filter tap, and the n subscript represents the time index, which is required since adaptive filters adjust their coefficients at every time instance.

The direct FIR structure is shown in **Figure 6** and the output y at any time n is given by Equation (13), where nodes B and C are described in Equations (14) and (15) respectively.

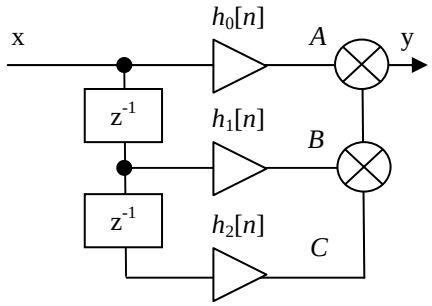


Figure 6. Direct form FIR structure

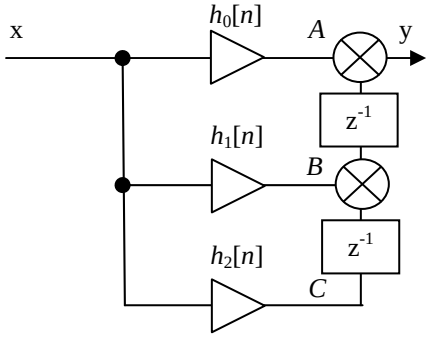


Figure 7. Transposed form FIR structure

$$y[n] = A[n] = x[n]h_0[n] + B[n] \tag{13}$$

$$B[n] = x[n-1]h_1[n] + C[n] \tag{14}$$

$$C[n] = x[n-2]h_2[n] \tag{15}$$

$$y[n] = x[n]h_0[n] + x[n-1]h_1[n] + x[n-2]h_2[n] \tag{16}$$

$$y[n] = \sum_{k=0}^{N=2} x[n-k]h_k[n] \tag{17}$$

The transposed FIR structure is shown in **Figure 7** and the output y at any time n is given below.

$$y[n] = x[n]h_0[n] + B[n-1] \tag{18}$$

$$B[n] = x[n]h_1[n] + C[n-1] \tag{19}$$

$$C[n] = x[n]h_2[n] \tag{20}$$

$$y[n] = x[n]h_0[n] + x[n-1]h_1[n-1] + x[n-2]h_2[n-2] \tag{21}$$

$$y[n] = \sum_{k=0}^{N=2} x[n-k]h_k[n-k] \tag{22}$$

Compared with the direct FIR output, the difference in the [n-k] index of the coefficient indicates that the filters produce equivalent output only when the coefficients don't change with time. This means if the transposed FIR architecture is used, the LMS algorithm will not converge differently from the direct implementation is used [7]. The change needed was to account for the weights as shown in Equation (23). A suitable approximation is to update the weights at every N input, where N is the length of the filter. This obviously will converge N times slower. Though simulations show that it never actually converges with as good results as the traditional LMS algorithm. It may be acceptable still though, due to the increased bandwidth of the transposed form FIR, when high convergence rates are not required.

$$\hat{w}(M-n+1) = \hat{w}(M-n) + \frac{u(n)\dot{e}(n)}{\mu \cdot scale} \tag{23}$$

3.3 Implementing Adaptive Noise Filter with FPGAs

Adaptive noise filtering techniques are applied to low frequency like voice signals, and high frequency signals such as video streams, modulated data, and multiplexed data coming from an array of sensors. Unfortunately in all high frequency and high speed applications, a software implementation of the adaptive noise filtering usually doesn't meet the required processing speed, unless a high end DSP processor is used. A convenient solution can be represented by a dedicated hardware implementation using a Field Programmable Gate Array (FPGA). In this case the limiting factor is represented by a number of

multiplications required by the adaptive noise cancellation algorithm. By using a novel modified version of the LMS algorithm, the proposed implementation allows the use of a reduced number of hardware multipliers. Moreover experimental data showed that the modified algorithm achieves the same or even better performances than the standard LMS version. There are many possible implementations for an adaptive noise filter, but the most widely used employs a Finite Impulse Response (FIR) digital filter, whose coefficients are iteratively updated using the LMS algorithm. The algorithm is described in Equations (24) to (26), leading to the evaluation of the FIR output, the error, and the weights update.

$$Y_i = X_i^T W_i \tag{24}$$

$$e_i = D_i - Y_i \tag{25}$$

$$W_{i+1} = W_i + 2\mu e_i X_i \tag{26}$$

In the above equations, X_i is a vector containing the reference noise samples, D_i is the primary input signal, W_i is the filter weights vector at the i^{th} iteration, and e_i is the error signal. The μ coefficient is often empirically chosen to optimize the learning rate of the LMS algorithm. The hardware implementation of the algorithm in an FPGA device is not trivial, since the FIR filter has not constant coefficients, so multipliers cannot be synthesized by using a look-up table (LUT) based approach. This however, should be straightforward in FPGA architecture. Multipliers with changing inputs instead need to be built by using a significantly greater number of internal logic resources (either elementary logic blocks or embedded multipliers). In an Nth order filter the algorithm requires at least 2N multiplications and 2N additions. Note the factor 2μ that is usually chosen to be a power of two in order to be executed by shifting. This makes it impractical for fully parallel hardware implementation of the algorithm as the value of N grows. This is due to the huge number of multipliers required. In order to reduce the complexity of the algorithm, the weights update expression (Equation (26)) is simplified as in Equation (27).

$$W_{i+1} = W_i + \alpha e_i \operatorname{sgn}(X_i) = W_i + \Delta_i \tag{27}$$

As a consequence the weights are updated using a factor proportional to the error and the sign of the current reference noise sample, instead of its value. This implies that weights can be updated by using an addition (or subtraction) instead of a multiplication. This simplified algorithm requires only N multiplications and 2N additions. However the simplification of the weights update rule usually results in worse learning performances, *i.e.* in a slower adaptation capability of the filter. To overcome this weakness, and significantly improve the learning characteristics, a dynamic learning rate coefficient α has been used. Generally this can be done by updating it with an adaptive rule, or, by using a heuristic function. Simulations of the above mentioned method shows that a dynamic learning rate gives an advantage not only in the learning characteristics, but also in the accuracy of the final solution (in term of improvement of the signal to noise ratio of the steady state solution). The product αe_i is used to update all weights; only one additional multiplication is required.

3.4 Architecture for Implementation on FPGA

The architecture of the adaptive noise filtering based on the modified LMS algorithm is shown in **Figure 8**. It was designed to implement 32 tap adaptive noise filter in a medium density FPGA device. It has a modular and scalable structure composed by 8 parallel stages, each one capable of executing 1 to 4 multiply and accumulate (MAC) operations and weights update. By controlling the number of operation performed by each block it is possible to implement an adaptive filter whose order can range from 8 to 32. In the first case, by exploiting maximum parallelism, the filter is capable of processing a data sample per clock cycle. In the other cases 2 to 4 clock cycles are requested. Some FPGA's internal RAM blocks were used to implement the tap delays and to store weights coefficients. Each weights update block is mainly composed by an adder/subtractor accumulator. The weights update coefficients Δ_i are computed by a separated block, which also handles the learning rate update function, following the above mentioned heuristic algorithm, and implements its multiplication with the error signal. By slightly modifying this unit, a more sophisticated adaptive function, can be easily obtained, thus enhancing the performances of the adaptive noise filtering for non stationary signals.

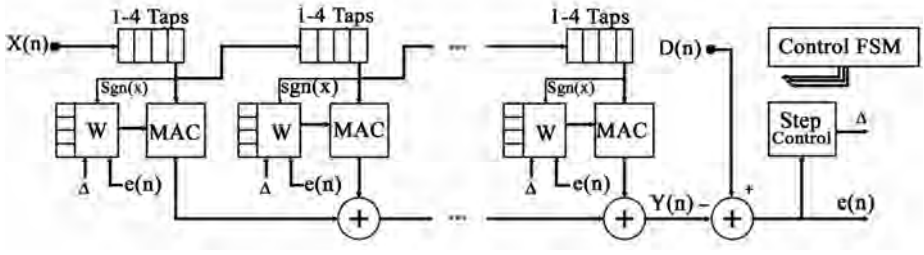


Figure 8. Architecture of the modified LMS filter

4. Simulations and Results

Adaptive noise filters have been implemented on DSPs and FPGAs. Motorola DSP56303 has been used for DSP platform, while Xilinx Spartan III boards are used to implement FPGA adaptive noise filtering. Matlab Simulink has been used to test the effectiveness and correctness of the adaptive filters before hardware implementation.

4.1 Matlab Simulink Simulations and Results

To test the theory and see the improvements visually that is proposed by Das, the adaptive filter that is proposed by Lee and Das has been compares through Matlab Simulink. (see **Figure 9**)

The target simulink model is responsible for code generation where as the host simulink model is responsible for testing. The host drives the target model with heavy wavelet noisy test data consisting of 4096 samples generated from wnoise function in Matlab. Matlab's fdatool is used for designing the bandpass filter to color the noise source. A colored Gaussian noise is then added to the input test signal. This noisy signal and the reference noise are inputs to the terminal of the LMS filter Simulink block. **Figure 10** Desired Signal (top), received signal (middle), output (bottom) This code has been implemented in C programming language. The LMS filter is placed in the virtual internal ram of the simulink model. In the code, breakpoints are placed in the corresponding section of the code where FIR filtering takes place. It takes 46, 213 and 266 clock cycles to run the filtering section. The time computation would be the clock cycles measured, divided by 225 MHz, which is the virtual clock speed. The execution time is 205 ms. The implementation of LMS filter takes worst case time of 38.95 ms to compute the filtering of heavy sine noisy signal consisting of 4096 samples per frame. **Figure 11** shows the comparison between the Das proposal of the wiener filter and the Lee's wiener filter proposal in the signal to noise ratio aspect. As it can be seen from the **Figure 11** the performance for the Das proposal is higher than the Lee's wiener filter. The improved adaptive wiener filter provides SNR improvement from 2.5 to 4 dB as compared to Lee's adaptive wiener filter.

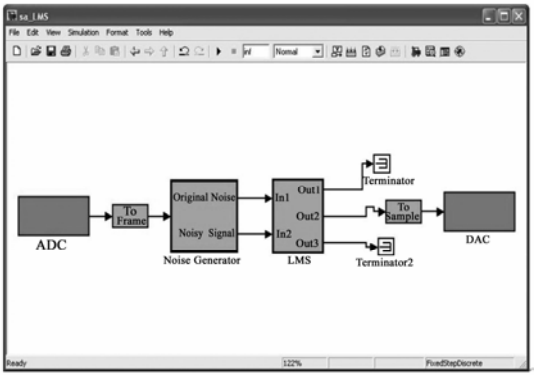


Figure 9. Block diagram of Matlab Simulink

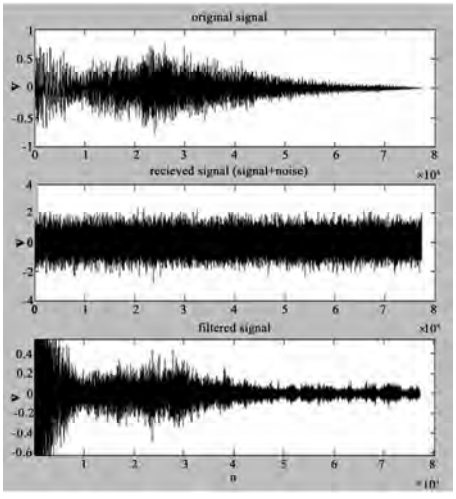


Figure 10. Desired signal

4.2 Motorola DSP56300 Results

The DSP system consists of two analog-to-digital (A/D) converters, and two digital-to-analog converters (D/A) converters. The DSP56303EVM evolution module is used to provide and control the DSP56300 processor, the two A/D converters, and the two D/A converters. The left analog input signal $x(t)$ consists of the desired input signal $s(n)$ plus a white noise signal $w(n)$. The left analog input signal $x(t)$ is first digitized using the A/D converter on the evaluation board. DSP Processor executes the adaptive filter algorithm to process the left digitized input signal $x(n)$, the left and right output signals y_1(n) and $y_2(n)$ will be generated. The left output signal $y_1(n)$ is the error signal. The right output signal $y_2(n)$ is the filtered version of the left digitized input signal $x(n)$, which is an estimate of the desired input signal $s(n)$. The two D/A converters on the evaluation board are then used to convert the left and right digital output signals $y_1(n)$ and $y_2(n)$ to the left and right analog output signals $y_1(t)$ and $y_2(t)$.

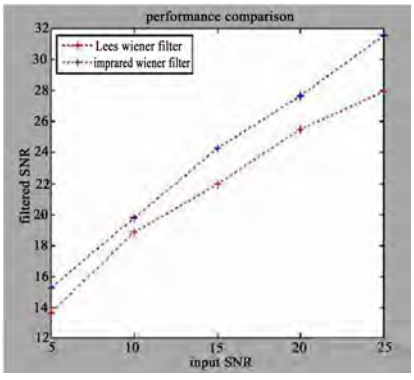


Figure 11. SNR performance comparison between Lee and Das proposals

The continuous analog signal was sampled at a rate of twice the highest frequency present in the spectrum of the sampled analog signal in order to accurately recreate the analog audio signal from the discrete samples. The analog audio signal was mixed with noise using a sum block which is bound to occur when the audio signal passes through the channel. The noise however, first low pass passed filter using a finite impulse response filter to make it finite in bandwidth. FIR noise filter was observed to have little or no significant effect on the signal with noise. The information bearing signal is a sine wave of $0.055\frac{cycles}{sample}$ is shown in **Figure 12**. The noise picked up by the secondary microphone is the input for the adaptive filter as shown in **Figure 13**. The noise that corrupts the sine wave is a low pass filtered version of the noise. The sum of the filtered noise and the information bearing signal is the desired signal for the adaptive filter. The noise corrupting the information bearing signal is a filtered version of noise as shown in the **Figure 14**. **Figure 15** shows that the adaptive filter converges and follows the desired filter response. The filtered noise should be completely subtracted from the signal noise combination and the error signal should only have the original signal. The results can be seen in **Figures 12** to **16**.

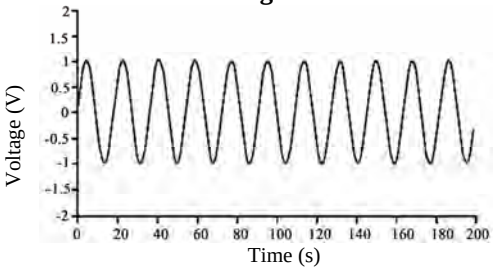


Figure 12. Plot showing the input signal

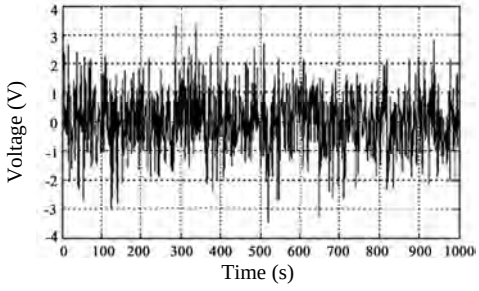


Figure 13. Plot of the noise signal

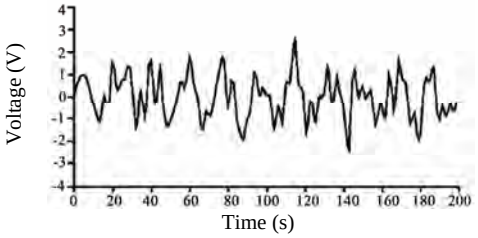


Figure 14. Noise corrupting the original signal

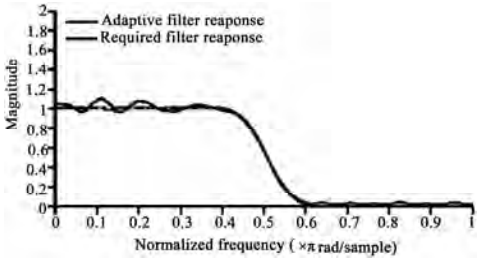


Figure 15. Convergence of the adaptive filter response to the response of the FIR filter

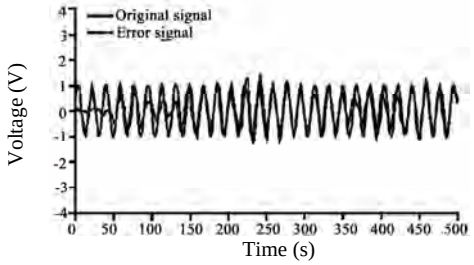


Figure 16. Plot of the error and the original signal

4.3 Xilinx Spartan III Results

The algorithm for adaptive filtering were coded in Matlab and experimented to determine optimal parameters such as the learning rate for the LMS algorithm. After the parameters have been determined, algorithms were coded for Xilinx in VHDL language.

4.3.1 Standard LMS Algorithm Results

The desired signal output was a sine wave, and it was corrupted by a higher frequency sinusoid and random Gaussian noise with a signal to noise ratio of 5.86 dB. The input signal can be seen in **Figure 17**. A direct form FIR filter of length 32 is used to filter the input signal. The adaptive is trained with the LMS algorithm with a learning rate $\mu = 0.05$. It appears that the filter with the standard LMS algorithm has learned the signal statistics and is filtering within 200-250 iterations. Since the results have shown that the standard LMS algorithm removes the noise from the signal, the next section. The timing analyzer has showed that the clock for standard LMS algorithm is 25 MHz. The input and output signals for the standard LMS algorithms are given in **Figures 17** and **18**.

4.3.2 Modified LMS Algorithm Results

The noise reduction obtained by both the standard LMS algorithm and the modified algorithm as applied to a sta-

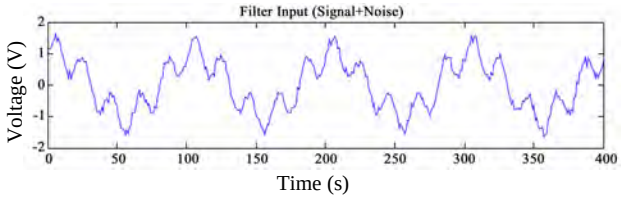


Figure 17. Input signal for standard LMS algorithm

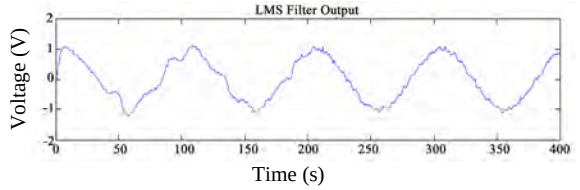


Figure 18. Output signal for standard LMS algorithm

tionary signal composed by 3 frequencies, corrupted by a random Gaussian noise, with signal to noise ratio of 5.86 dB were studied. Both algorithms used 16 bit fixed point representation for data and filter coefficients [14]. The frequency spectrum of the original signal, standard LMS, and modified LMS filter are given in **Figure 19**. The modified LMS used a dynamic learning rate coefficient α based on a heuristic function formerly proposed by Widrow [8], and consisted of 1/n decaying function, coefficients were approximated by a piecewise linear curve, starting from the value 0.1 down to 0.001 (in about 1000 steps). This heuristic function achieved a faster convergence, and les gradient noise. It has proved to be effective when applied to stationary signals. On the other hand the standard LMS used a static learning rate with the best performances obtained by setting the μ parameter equal to 0.05. The two algorithms reported noise attenuation greater than 40 dB and 36 dB respectively. As can be seen from the two learning characteristics in **Figure 20**

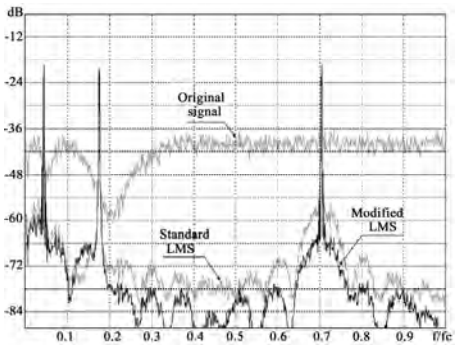


Figure 19. Frequency Spectrum of a signal processed with the standard and modified LMS

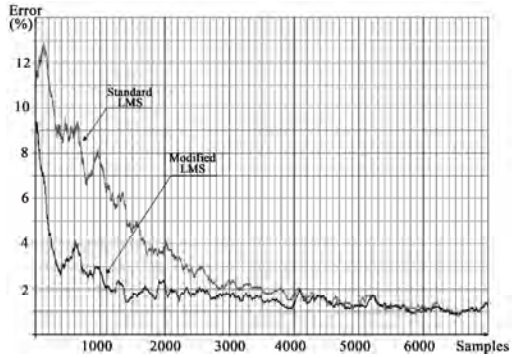


Figure 20. Learning Characteristics of both LMS algorithms

the modified LMS offered a faster convergence. A large class of signals (either stationary or short term stationary) and noises showed similar simulation results. The adaptive noise filtering was implemented using a 16 bit 2's complement fixed point representation for samples and weights. As it can be seen in **Figure 5**, the floor planned design required 1776 slices (logic blocks) of 3072 available (about 57%), and allowed a running clock frequency of 50 MHz (with a non optimized, fully automatic place & route process). It would require 2750 slices (89%) and would run at less than 25 MHz (due mainly to routing congestion). The Assembly file used for the simulation is given in Appendix A. The assembly code is provided elsewhere [26].

5. Conclusions

As discussed in the previous chapters, the concept of the adaptive noise filtering applications can be implemented in both DSP processors like Motorola DSP56300 series and also in the Field Programmable Gate Array such as Xilinx Spartan III boards. In high performance signal processing applications, FPGAs have several advantages over high end DSP processors. Literature survey has showed that high-end FPGAs have a huge throughput advantage over high performance DSP processors for certain types of signal processing applications. FPGAs use highly flexible architectures that can be greatest advantage over regular DSP processors. However, FPGAs come with a hardware cost. The flexibility comes with a great number of gates, which means more silicon area, more routing and higher power consumption. DSP processors are highly efficient for common DSP tasks, but the DSP typically takes only a tiny fraction of the silicon area, which is dedicated for computation purposes. Most of the area is designated for instruction codes and data moving. In high performance signal processing applications like video processing, FPGAs can take highly parallel architectures and offer much higher throughput as compared to DSP processors. As a result FPGA's overall energy consumption may be significantly lower than DSP processors, in spite of the fact that their chip level power consumption is often higher. DSP processors can consume 2-3 watts, while the FPGAs can consume in the order of 10 watts. The pipeline technique, more computation area and with more gates FPGAs can process more channels at the same time. Thus power consumption per channel is significantly less in the FPGA's [15]. DSPs are specialized forms of microprocessor, while the FPGA's are form of highly configurable hardware. In the past, the usage of DSPs has been nearly ubiquitous, but with the needs of many applications outstripping the processing capabilities (MIPS) of DSPs, the use of FPGAs has become very prevalent. It has generally come to be expected that all software, (DSP code is considered a type of software) will contain some bugs and that the

best can be done is to minimize them. Common DSP software bugs are caused because of, failure of interrupts to completely restore processor state upon completion, non-uniform assumptions regarding processor resources by multiple engineers simultaneously developing and integrating disparate functions, blocking of critical interrupt by another interrupt or by an uninterruptible process, undetected corruption or non-initialization of pointers, failing to properly initialize or disable circular buffering addressing modes, memory leaks, the gradual consumption of available volatile memory due to failure of a thread to release all memory when finished, dependency of DSP routines on specific memory arrangements of variables, use of special DSP "core mode" instruction options in core, conflict or excessive latency between peripheral accesses, such as DMA, serial ports, L1, L2, and external SDRAM memories, corrupted stack or semaphores, subroutine execution times dependent on input data or configuration, mixture of "C" or high-level language subroutines with assembly language subroutines, and pipeline restrictions of some assembly instructions [15]. Both FPGA and DSP implementation routes offer the option of using third party implementation for common signal processing algorithms, interfaces and protocols. Each offers the ability to reuse existing IP in the future designs. FPGA's are more native implementation for more DSP algorithms. **Figures 21** and **22** give the block diagrams of the DSP and FPGA respectively.

Motorola DSP56300 series can only do one arithmetic

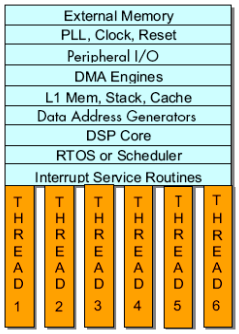


Figure 21. Digital signal processor block diagram

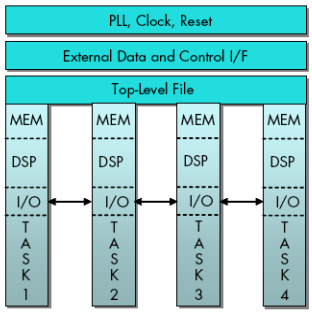


Figure 22. FPGA's block diagram

computation and two move instructions at a time. However, in the case of FPGAs, each task can be computed by its own configurable core and designated input and output interface.

5.1 Speed Comparison

Speed is one of the most important concepts that determine the computation time and also it is one of the most important concepts in the market. In the adaptive filters the parameters are updated with the each iteration and after the each iteration the error between the input and the desired signal get smaller. After some number of iterations the error becomes zero and the desired signal is achieved. According to the specifications from the manufacturer manuals, Motorola DSP56300 series has a CPU clock of 100 MHz, but this speed depend on the instruction fetch, computation speed and also the speed of the peripherals. On the DSP56303EVM board the audio codec runs on 24.57 MHz, this clock speed is determined by an external crystal. In the other hand Xilinx Spartan 3 has the maximum clock frequency of 125 MHz, but this speed can be reduced because of the number of instruction ns, gates and the congestion on the routing of the signals. Both of the modified adaptive noise filtering applications take about 200-250 iterations to cancel the noise and achieve the desired signal. In the Motorola DSP processor case because of the actual clock speed being lower, causality conditions and the speed limitation that is coming from the audio codec part of the board, the running time of the modified LMS algorithm is 20 MHz. in the case of the FPGA's the running speed is around 50 MHz. This due to discussions from the previous section, which is FPGA's flexibility and reconfigurable gates allows for the clock to be faster.

5.2 General Conclusions

As discussed in the previous sections, we have shown the differences between the DSP processors and FPGAs. As far as power and cost are considered, DSP processors in general have lower power consumption, which makes them suitable for battery powered applications. These applications can be done on audio applications. These voice applications are very straight forward and do not require sophisticated pipeline and parallel moves. Audio applications can be different filter applications. These are used especially in the voice transmission lines and cell phones. When it comes to the high frequency applications, DSP processors have some restrictions on their part when they are compared to the FPGAs. In high speed applications, FPGA's are much faster than the DSP processors. When it comes to high speed applications, the DSP boards have some limitations when compared to the FPGAs. FPGAs can offer more channels, and thus when cost per channel is considered because FPGAs can

offer more channels, the cost per channel is lower than the DSP's. Also the partitioning of the FPGA's can offer more throughputs as compared to DSP processors. Thus FPGAs can handle multiple tasks when their controls and finite state machines are configured correctly.

According to our study, the final conclusion is that for simple audio applications like adaptive noise cancelling, Motorola DSP56300 is more beneficial, because the requirements for audio applications are met with DSP processors. Also they are more power efficient and can be used for battery powered devices. But when adaptive noise filtering is considered in high speed applications like video streaming and multiplexed array signals, FPGA's are offering a faster approach and thus they are more suitable for high frequency applications.

5.3 Future Work

In the future, the adaptive noise filtering can be implemented on high frequency applications, such as noise removal from video streaming and noise removal from multiplexed data arrays. These applications may be applied first to FPGAs with Verilog HDL or VHDL. After application has been verified, hardware code can be converted to a net list and thru Synopsys a custom ASIC design can created. The ASIC design and FPGA design may be compared in the aspect of cost, power, architecture, noise removal and speed. These comparisons would be helping us to provide us a more educated choice for future applications.

REFERENCES

[1] A. Di Stefano, A. Scaglione and C. Giaconia, "Efficient FPGA Implementation of an Adaptive Noise Canceller," *Proceedings Seventh International Workshop on Computer Architecture for Machine Perception*, Palermo, 2005, pp. 87-89.

[2] M. El-Sharkawy, "Digital Signal Processing Applications with Motorola's DSP56002 Processor," Prentice Hall, Upper Saddle River, 1996.

[3] K. Joonwan and A. D. Poularikas, "Performance of Noise Canceller Using Adjusted Step Size LMS Algorithm," *Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory*, Huntsville, 2002, pp. 248-250.

[4] R. M. Mersereau and M. J. T. Smith, "Digital Filtering A Computer Laboratory Textbook," John Wiley & Sons, Inc., New York, 1994.

[5] J. Proakis and D. Manolakis, "Digital Signal Processing Principles, Algorithms, and Applications," 4th Edition, Pearson Prentice Hall, Upper Saddle River, 2007.

[6] G. Saxena, S. Ganesan and M. Das, "Real Time Implementation of Adaptive Noise Cancellation," *EIT* 2008 *IEEE International Conference on Electro/Information Technology*, Ames, 2008, pp. 431-436.

[7] K. L. Su, "Analog Filters," Chapman & Hall, London, 1996.

[8] B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, Eugene Dong, Jr., and R. C. Goodlin, "Adaptive Noise Cancelling: Principles and Applications," *Proceedings of the IEEE*, Vol. 63, 1975, pp. 1692-1716.

[9] S. M. Kuo and D. R. Morgan, "Active Noise Control: A Tutorial Review," *Proceedings of the IEEE*, Vol. 87, No. 6, June 1999, pp. 943-973.

[10] K. C. Zangi, "A New Two-Sensor Active Noise Cancellation Algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, Vol. 2, 1993, pp. 351-354.

[11] A. V. Oppenheim, E. Weinstein, K. C. Zangi, M. Feder, and D. Gauger, "Single-Sensor Active Noise Cancellation," *IEEE Transactions on Speech and Audio Processing*, Vol. 2, 1994, pp. 285-290.

[12] T. H. Yeap, D. K. Fenton and P. D. Lefebvre, "Novel Common Mode Noise Cancellation Techniques for xDSL Applications," *Proceedings of the* 19*th IEEE Instrumentation and Measurement Technology Conference*, Anchorage, Vol. 2, 2002, pp. 1125-1128.

[13] Xilinx Corp., "Spartan-IIIE 1.8V FPGA Family: Functional Description," November 2002.

[14] B. Dukel, M. E. Rizkalla and P. Salama, "Implementation of Pipelined LMS Adaptive Filter for Low-Power VLSI Applications," *The* 45*th Midwest Symposium on Circuits and Systems*, Tulsa, Vol. 2, 2002, pp. II-533- II-536.

[15] M. Das, "An Improved Adaptive Wiener Filter for De-noising and Signal Detection," *International Association of Science and Technology for Development, International Conference on Signal and Image Processing*, Honolulu, 2005, p. 258.

[16] K. Schutz, "Code Verification using RTDX," MathWorks Matlab Central File Exchange.

[17] S. Haykin, "Adaptive Filter Theory," Englewood Cliffs, Prentice Hall, Upper Saddle River, 1991.

[18] D. L. Donoho and J. M. Johnstone, "Ideal Spatial Adaptation by Wavelet Shrinkage," *Biometrika*, Vol. 81, 1 September 1994, pp. 425-455.

[19] J. Petrone, "Adaptive Filter Architectures for FPGA Implementation," Master's Thesis, Department of Electrical and Computer Engineering, Florida State University, Tallahassee, 2004.

[20] S. Manikandan and M. Madheswaran, "A New Design of Adaptive Noise Cancellation for Speech Signals Using Grazing Estimation of Signal Method," *International Conference on Intelligent and Advanced Systems*, Kuala Lumpur, 2007, pp. 1265-1269.

[21] K. Chang-Min, P. Hyung-Min, K. Taesu, C. Yoon-Kyung and L. Soo-Young, "FPGA Implementation of ICA Algorithm for Blind Signal Separation and Adaptive Noise Canceling," *IEEE Transactions on Neural Networks*, Vol. 14, 2003, pp. 1038-1046.

[22] S. M. Kay, "Fundamentals of Statistical Signal Process-

ing," Prentice Hall, Upper Saddle River, 1996.

[23] J.-S. Lee, "Digital Image Enhancement and Noise Filtering by Use of Local Statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, 1980, pp. 165-168.

[24] Alon Halim, "Real Time Noise Cancellation Field Programmable Gate Arrays," MSECE Thesis, Purdue University, Lafayette, 2009.

Appendix A

```
init_filter macro
      move      #states,r1
      move      #ntaps-1,m1
      move      #coef,r4
      move      #ntaps-1,m4
      endm
stafirmacro     ntaps,lg,foutput,ferror
clr   a     x0,x:(r1)+ y:(r4)+,y0
rep   #ntaps-1
mac   x0,y0,a           x:(r1)+,x0 y:(r4)+,y0
macr  x0,y0,a     x:(r1),b
move      a,x:foutput
sub   a,b
nop
move      b,x:ferror
move      #lg,y1
move      b,x1
mpy   x1,y1,b
move      b,y1
do    #ntaps,_update
move      y:(r4),a          x:(r1)+,x0
mac   x0,y1,a
move      a,y:(r4)+
_update
lua   (r1)-,r1
nop
move      x:foutput,b
move      x:ferror,a
endm
```


J. Software Engineering & Applications, 2010, 3: 404-408
doi:10.4236/jsea.2010.34045 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Intelligent Supply Chain Management

Mohammad Zubair Khan[1], Omar Al-Mushayt[1], Jahangir Alam[2], Jorair Ahmad[1]

[1]Faculty of Computer Science and Information System, University of Jizan, Jizan, Kingdom of Saudi Arabia; [2]University Women's Polytechnic, Faculty of Engineering and Technology, Aligarh Muslim University, Aligarh, India.
Email: Zubair.762001@gmail.com, oalmushayt@yahoo.com, Jahangir.uk786@yahoo.co.uk

Received December 27th, 2009; revised January 31st, 2010; accepted February 5th, 2010.

ABSTRACT

Fuzzy Logic is used to derive the optimal inventory policies in the Supply Chain (*SC*) *numbers. We examine the performance of the optimal inventory policies by cutting the costs and increasing the supply chain management efficiency. The proposed inventory policy uses multi-agent and Fuzzy logic, and provides managerial insights on the impact of the decision making in all the SC numbers. In particular, we focus on the way in which our agent purchases components using a mixed procurement strategy* (*combining long and short term planning*) *and how it sets its prices according to the prevailing market conditions and its own inventory level* (*because this adaptivity and flexibility are key to its success*). *In modern global market, one of the most important issues of the supply chain* (*SC*) *management is to satisfy changing customer demands and enterprises should enhance the long-term advantage through the optimal inventory control. In this paper an intelligent multi-agent system to simulate supply chain management has been developed.*

Keywords*: *SCM Supply Chain Management, Customer Agent, RFQ, Component Agent

1. Background and Introduction

1.1 Supply Chain Management

Supply chains encompass the companies and the business activities needed to design, make, deliver, and use a product or service. Businesses depend on their supply chains to provide them with what they need to survive and thrive [1-3]. Every business fits into one or more supply chains and has a role to play in each of them [3,4]. A multiechelon supply chain is illustrated in **Figure 1**. It includes different types of flows *i.e.* financial flow, Information flow and material flow. Third party logistics (3PL) services providers (3PL *is an organization that manages and executes a particular logistics function, using its own assets and resources, on behalf of another company.*) handle the inbound and outbound logistics for the shipper (*The person or company who is usually the supplier or owner of commodities shipped*). The inbound logistics take care of the material management while the outbound logistics deal with physical distribution of final products. The term "supply chain management" arose in the late 1980s and came into widespread use in the 1990s. Prior to that time, businesses used terms such as "logistics" and "operations management" instead. Some definitions of a supply chain are offered below:

- "A supply chain is the alignment of firms that bring products or services to market."—Lambert, Stock, and Ellram (1998).
- "A supply chain consists of all stages involved, directly or indirectly, in fulfilling a customer request. The supply chain not only includes the manufacturer and suppliers, but also transporters, warehouses, retailers, and customers themselves."—Chopra and Meindl (2001).
- "A supply chain is a network of facilities and distribution options that performs the functions of procurement of materials, transformation of these materials into intermediate and finished products, and the distribution of these finished products to customers."—Ganeshan and Harrison (1995).

Master strategist and a skillful general Napoleon once remarked, "An army marches on its stomach." Which holds true in business, as it moves on physical flow of materials, another saying that goes is also valid for the business, "Amateurs talk strategy and professionals talk logistics" [5-7].

Thus, we can say that Supply chain management is a set of approaches used to efficiently integrate suppliers, manufacturers, warehouses, and customers so that merchandise is produced and distributed at the right quantities, to the right locations, and at the right time in order to minimize system wise costs while satisfying service-level requirements.

Supply chain management (SCM) is the task that moves in a process from supplier to manufacturer to wholesaler to retailer to consumer [1,4]. Supply chain management involves coordinating and integrating these

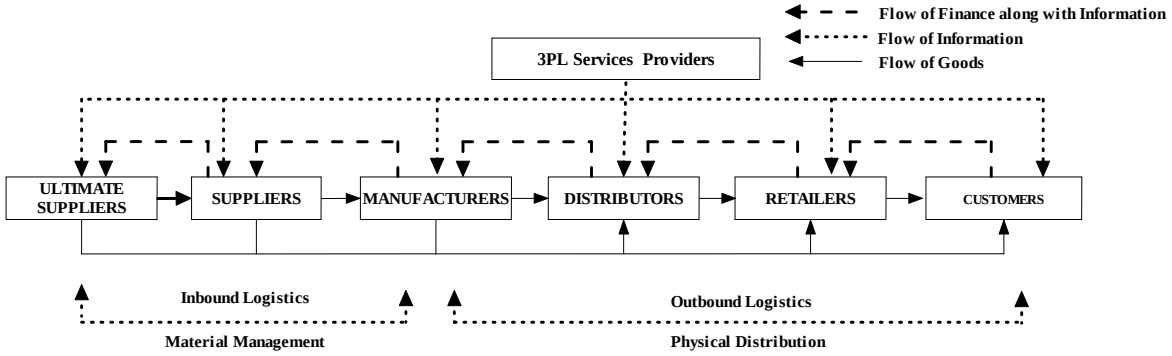


Figure 1. The supply chain process

flows both within and among companies [2,8]. It is said that the ultimate goal of any effective supply chain management system is to reduce inventory (with the assumption that products are available when needed).

Supply chain management flows can be divided into three main flows:

- The product flow
- The information flow
- The finances flow

The need of Supply chain Management in today's scenario is:

- Millions of dollars at stake!
- Excess Inventory costs
- Excess freight charges
- Lost sales/Stock outages
- Wasted time and energy
- Extra staff
- Listings/Delisting
- Customer dissatisfaction – privatization
- Capital costs
- Real Estate Costs
- Static and unresponsive SC policies
- Large inventories
- Unreliable deliveries
- Underperformance

1.2 Fuzzy Logic

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. Just as in fuzzy set theory the set membership values can range (inclusively) between any values, in fuzzy logic the degree of truth of a statement can range between any values and is not constrained to the two truth values {true, false} as in classic predicate logic [1]. And when *linguistic variables* are used, these degrees may be managed by specific functions, as discussed below.

Fuzzy Set Theory defines Fuzzy Operators on Fuzzy Sets. The problem in applying this is that the appropriate Fuzzy Operator may not be known. For this reason, Fuzzy logic usually uses IF-THEN rules, or constructs that are equivalent, such as fuzzy associative matrices. Rules are usually expressed in the form: IF *variable* IS *property* THEN *action.*

For example, an extremely simple temperature regulator that uses a fan might look like this: If temperature IS very cold THEN stop fan. If temperature is cold THEN turn down fan. If temperature IS normal THEN maintain level. IF temperature IS hot THEN speed up fan**.** Notice there is no "ELSE". All of the rules are evaluated, because the temperature might be "cold" and "normal" at the same time to different degrees. This paper proposes a multi-agent system to simulate supply chain management using the concepts of fuzzy logic. The rest of the paper is organizes as follows—Section 2 presents the proposed model, Methodology adopted conducting the experiments and results so obtained have been discussed in Section 3. Section 4 concludes the paper.

2. Proposed Model

The proposed SCM model has been depicted in **Figure 2**. In our model we are introducing the customer agent and Customer RFQ (Request for Quantities). The customer agent receives customer RFQ requesting a quantity of a particular commodity for delivery on a specified day. The customer agent is the key component while component agent is responsible for dealing with the component suppliers and aims to ensure that there are always sufficient components in stock to address the customers' changing demand for finished products**.**

In our model we propose customer agent only and leave the component agent for future studies.

2.1 The Customer Agent

The customer agent is the key component in our model (because we believe that offering the appropriate price at the right time is vital for success). If the price is too low, the agent will receive a low profit and if it is too high it will fail to win any orders (because the natural tendency of customers is to always choose the lowest offer price among those they receive). Given this, the key challenges are to determine which customer RFQ to bid for and at

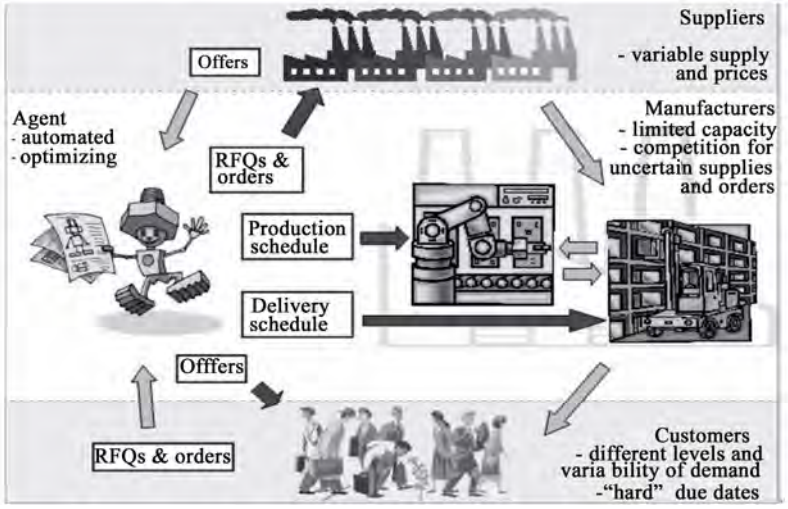


Figure 2. The SCM model

what price. To achieve this, we use fuzzy reasoning [9,10] to determine how to set prices according to the agent's inventory level, the market demand and the time into consideration.

2.2 Choosing Customer RFQ's

The customer agent receives customer RFQs requesting a quantity of a particular commodity for delivery on a specified day [2,8,11]. When selecting which RFQs to respond to, Our ISCM rates them according to the potential profit that they may bring and according to the inventory it holds. The latter inventory driven strategy offers customers commodity according to what is currently available and also what can be produced given the delivery date of pending components orders. In more detail, suppose a customer RFQ is represented as a tuple (**i, q, pres, cpenalty, pbase**), where i is the type of commodity the customer wants, q > 0 is the quantity, pres > 0 is the reservation price (maximum it will pay), cpenalty > 0 is the fine paid if the computers are not delivered on time, and ddue is the desired delivery date. On each day, the customer agent receives a bundle of such RFQs and sorts them in the decreasing order of the profit margin of the type of commodity requested. Here pbase is the cost of buying components (sum of the buying price for each component). The intuition is that the agent will first serve customers with high profit margins and low penalties. This is because the higher the pres, the more profit will be made (compared to selling the same product to a customer with a low pres). At the same time, the agent also wants to avoid getting high penalty orders because of the inherent uncertainties that exist in the scenario.

$$\textit{Profit Margin} = \textit{Pres}-\textit{Pbase}-(\textit{cpenalty}/q)$$

3. Methodology

We are applying the Fuzzy logic if-else algorithm using the following rules and simulated it using available data. Here we define fuzzy rules and then revisited the rules for our goal.

3.1 Fuzzy Rules for Calculating Offer Prices

- R1: if D is high and I is low then r1 is very-big
- R2: if D is medium and I is high then r2 is very-medium
- R3: if D is low and I is high then r3 is small

3.2 Fuzzy Rules Revisited

- If D is high I is High E is far then r1` is big.
- If D is high I is High E is in between then r2` is big.
- If D is high I is High E is near then r3` is big.

3.3 The Component Agent

The component agent is responsible for dealing with the component suppliers and aims to ensure that there are always sufficient components in stock to address the customers' changing demand for finished products. In doing so, it addresses a challenge that is common to all supply chains facing dynamically changing customer demand. That is, it must procure components at a low cost, whilst simultaneously maintaining a minimal component inventory in order to reduce the daily storage cost and also the possibility of being left with redundant stock if customer demands changes**.**

Types of procurements

- Far future procurement
- Near Future Procurement

3.3.1 *Far Future Procurement*

Of the two strategies, the far future procurement one is the simpler. For this, the agent assumes that in the far future, there will be a daily minimum need, and thus checks whether there is sufficient current and pending

component inventory to meet this need. If not, it submits an RFQ for a fixed amount to the relevant supplier, requesting delivery on the date at which the predicted inventory falls below the daily minimum need.

3.3.2 *Near Future Procurement*

The near future procurement strategy is more complex. It consists of two elements, a daily demand predictor that predicts the future demand of components, and a market tracker that generates the RFQs to be sent to the suppliers, both to order actual components required and to test the market to discern the most profitable order lead time and set appropriate reserve price

3.4 Demand Predictor

As described above, the agent buys components for the near future based on a prediction of customer demand. Now, according to the game specification, the number of RFQs that each agent receives from the customers is described by three independent random walks; one for each market segment (finished products are classified into three such segments: high, mid and low range). In more detail, the number of RFQs that an agent receives, within a single market segment, on day d is denoted by Nd, and is drawn from a Poisson distribution whose expected value is given by the parameter, Qd. Thus, for each market segment,

$$Nd = Poisson(Qd)$$

Having predicted the number of RFQs that will be received, within each market segment, on each day within the near future, the agent then calculates the expected daily usage of each component type (Did).

3.5 Price Tracker

The price tracker acts to maintain an estimate of the current market price of the components. Due to the behaviors of the competing agents, this market price depends on the due date with which components are requested. For example, if the competing agents are ordering components with very short lead times, then the supplier will have little spare capacity, and thus, the corresponding offer prices that the agent receives will be greater than those of orders with long lead times.

3.6 Factory Agent

One of the main challenges for the factory agent is scheduling what to produce and when to produce it (*i.e.*, how to allocate supply resources and factory time).

This strategy involves manufacturing commodities according to customer orders and satisfying orders with an earlier delivery date. Now, since the computers stored in the factory will be charged storage cost, each order will be delivered as soon as it is filled. The agent builds the commodity according to the customers' orders it has obtained (which has the advantage of ensuring that the factory always produces the needed computers on time). However, if on any day, there are still free factory assembling cycles available, and the numbers of finished PCs in stock are below a certain threshold, then the agent produces additional PCs of each kind uniformly (subject to the availability of components) in order to maximize the factory utilization. It is critical that this threshold is set appropriately; a high threshold will lead to excessive finished PC inventory, which may be hard to sell if demand is low.

3.7 Simulated Results

Profit Margin	Inventory	End of season	Predictors Decision
High	High	Far	yes
High	High	In Between	yes
High	High	Near	yes
High	Medium	Far	yes
High	Medium	In Between	yes
High	Medium	Near	no
High	Low	Far	yes
High	Low	In Between	no
High	Low	Near	no
Medium	High	Far	yes
Medium	High	In Between	yes
Medium	High	Near	yes
Medium	Medium	Far	yes
Medium	Medium	In Between	yes
Medium	Medium	Near	no
Medium	Low	Far	no
Medium	Low	In Between	no
Medium	Low	Near	no
Low	High	Far	no
Low	High	In Between	no
Low	High	Near	yes
Low	Medium	Far	no
Low	Medium	In Between	no
Low	Medium	Near	no
Low	Low	Far	no
Low	Low	In Between	no
Low	Low	Near	no

4. Conclusions

This mixture of baseline and opportunistic purchasing behavior is a common strategy in this domain and the technology we develop for achieving this can be readily transferred. Second, we believe our pricing model technology will also be useful in real SCM applications where just undercutting competitors' prices can significantly improve profitability. Specifically, to apply our model in other domains, the designers of the rule base would need to adapt the fuzzy rules to reflect the factors that are most relevant. Now we believe that customer demand and inventory level are highly likely to be critical factors for almost all cases and thus these rules can remain unaltered.

By using different rule bases, different factors can easily be incorporated (as we did here, in order to handle the additional need to reduce inventory towards the end of the season). The purpose model also will use in this area

focuses on the component agent. We would like to improve it so that it can adapt the quantity for far future and near future procurement automatically between the seasons according to the procurement behaviors employed by the opponents.

REFERENCES

[1] J. Collins, R. Arunachalam, *et al.*, "The Supply Chain Management Game for the 2005 Trading Agent Competition," Technical Report CMU-ISRI-04-139, School of Computer Science, Carnegie Mellon University, Pittsburgh, December 2004.

[2] S. D. Levi, P. Kaminsky and S. E. Levi, "Designing and Managing the Supply Chain," McGraw-Hill, Illinois, 2000.

[3] D. Pardoe and P. Stone, "TacTex-03: A supply Chain Management Agent," *SIGecom Exchanges*: *Special Issue on Trading Agent Design and Analysis*, Vol. 4, No. 3, 2004, pp. 19-28.

[4] K. Kumar, "Technology for Supporting Supply-Chain Management," *Communications of the ACM*, Vol. 44, No. 6, pp. 58-61, 2001.

[5] D. Pardoe and P. Stone, "Predictive Planning for Supply Chain Management," *Proceedings of International Conference on Automated Planning and Scheduling*, to appear.

[6] M. Sugeno, "An Introductory Survey of Fuzzy Control," *Information Sciences*, Vol. 36, 1985, pp. 59-83.

[7] M. Wellman, J. Estelle, S. Singh, *et al.*, "Strategic Interactions in a Supply Chain Game," *Computational Intelligence*, Vol. 21, No. 1, 2005, pp. 1-26.

[8] M. He, H. F. Leung and N. R. Jennings, "An ARTMAP Based Bidding Strategy for Autonomous Agents in Continuous Double Auctions," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 6, 2003, pp. 1345-1363.

[9] R. Arunachalam and N. Sadeh, "The Supply Chain Trading Agent Competition," *Electronic Commerce Research and Applications*, Vol. 4, No. 1, 2005, pp. 63-81.

[10] J. Collins, R. Arunachalam, N. Sadeh, J. Ericsson, N. Finne and S. Janson, "The Supply Chain Management Trading Agent Competition," Technical Report CMU-ISRI-04-139, Carnegie Mellon University, Pittsburgh, 2004.

[11] M. He, N. R. Jennings and H. Leung, "On Agent-Mediated Electronic Commerce," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 4, 2003, pp. 985-1003.

J. Software Engineering & Applications, 2010, 3: 409-418
doi:10.4236/jsea.2010.34046 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Designing a Fuzzy Expert System to Evaluate Alternatives in Fuzzy Analytic Hierarchy Process

Hamed Fazlollahtabar[1], Hamid Eslami[2], Hamidreza Salmani[2]

[1]Mazandaran University of Science and Technology, Babol, Iran; [2]Science and Research Campus, Islamic Azad University, Member of Young Researchers Club, Tehran, Iran.
Email: hamed@ustmb.ac.ir

Received June 23rd, 2009; revised July 18th, 2009; accepted July 25th, 2009.

ABSTRACT

This paper concerns with proposing a fuzzy logic based expert system to breakthrough the problem of alternatives evaluation in Analytic Hierarchy Process (*AHP*). *AHP as a multi criteria decision aid helped decision makers for analyzing and prioritizing the alternatives in a hierarchical structure. During times AHP encountered some problems. Hence, fuzzy analytic hierarchy process* (*FAHP*) *and some other extensions of AHP have been configured to solve those problems.*

***Keywords*:** *Multi Criteria Decision Making* (*MCDM*), *Analytic Hierarchy Process* (*AHP*), *Expert System*

1. Introduction

Analytic hierarchy process (AHP) [1] has been widely used as a useful multiple criteria decision making (MCDM) tool or a weight estimation technique in many areas such as selection, evaluation, planning and development, decision making, forecasting, and so on [2]. The AHP is expressed by a unidirectional hierarchical relationship amongst decision levels. The top element of the hierarchy is the overall goal for the decision model. The hierarchy decomposes to a more specific criterion on a level and each criterion may be related to some subcriteria. The AHP separates complex decision problems into elements within a simplified hierarchical system.

The AHP usually consists of three stages of problem solving: decomposition, comparative judgments, and synthesis of priority. The decomposition stage aims at the construction of a hierarchical network to represent a decision problem, with the top level representing the overall objectives and the lower levels representing the criteria, subcriteria, and alternatives. With comparative judgments, users are requested to set up a comparison matrix at each hierarchy by comparing pairs of criteria or subcriteria. A scale of values ranging from 1 (Equally Preferred) to 9 (Extremely Preferred), is used to express the users preferences. Finally, in the synthesis of priority stage, each comparison matrix is then solved by an eigenvector method for determining the importance of the criteria and alternative performance.

One major advantage of AHP is its applicability to the problems of group decision-making. In a group decision setting, each participant is required to set up the preference of each alternative by the AHP and the collective views of the participants are used to obtain an average weighting of each alternative.

The traditional AHP requires crisp judgments. However, due to the complexity and uncertainty involved in real world decision problems, a decision maker (DM) may sometimes feel more confident to provide fuzzy judgments than crisp comparisons. A number of methods have been developed to handle fuzzy comparison matrices. For example, Van Laarhoven and Pedrycz [3] suggested a fuzzy logarithmic least squares method (LLSM) to obtain triangular fuzzy weights from a triangular fuzzy comparison matrix. Wang *et al.* [4] presented a modified fuzzy LLSM.

Buckley [5] utilized the geometric mean method to calculate fuzzy weights. Chang [6] proposed an extent analysis method, which derives crisp weights for fuzzy comparison matrices. Xu [7] brought forward a fuzzy least squares priority method (LSM). Mikhailov [8] developed a fuzzy preference programming method (PPM), which also derives crisp weights from fuzzy comparison matrices. Csutora and Buckley [9] came up with a Lambda-Max method, which is the direct fuzzification of the well-known kmax method.

Among the above approaches, the extent analysis method has been employed in quite a number of applications [10-28] due to its computational simplicity. However, such a method is found unable to derive the true

weights from a fuzzy or crisp comparison matrix. The weights determined by the extent analysis method do not represent the relative importance of decision criteria or alternatives at all. Therefore, it should not be used as a method for estimating priorities from a fuzzy pairwise comparison matrix. The purpose of this paper is to show by examples that the priority vectors determined by the extent analysis method do not represent the relative importance of decision criteria or alternatives and that the misapplication of the extent analysis method to fuzzy AHP problems may lead to a wrong decision to be made and some useful decision information such as decision criteria and fuzzy comparison matrices not to be considered. We illustrate these problems to avoid any possible misapplications in the future. Here, we compare the Fuzzy AHP with a proposed expert system and illustrate our proposed expert system in an example.

2. Review of the Extent Analysis Method on Fuzzy AHP

A triangular fuzzy number is represented by $\tilde{a} = (l, m, u)$, with the membership function, $\mu_{\tilde{a}}(x)$, defined by the expression,

$$\mu_{\tilde{a}}(x) = \begin{cases} 1-(\frac{m-x}{l}) & x \le m \\ 1-(\frac{x-m}{u}) & x > m \end{cases}$$

where m is the center, l is the left spread and u is the right spread. For two triangular fuzzy number $\tilde{M}_1 = (l_1, m_1, u_1)$ and $\tilde{M}_2 = (l_2, m_2, u_2)$ the fuzzy operations are defined as follows:

$$\tilde{M}_1 + \tilde{M}_2 = (l_1 + l_2, m_1 + m_2, u_1 + u_2)$$
$$\tilde{M}_1 \times \tilde{M}_2 = (l_1 \times l_2, m_1 \times m_2, u_1 \times u_2)$$
$$\tilde{M}_1^{-1} = (\frac{1}{u_1}, \frac{1}{m_1}, \frac{1}{l_1}), \quad \tilde{M}_2^{-1} = (\frac{1}{u_2}, \frac{1}{m_2}, \frac{1}{l_2})$$

Consider a triangular fuzzy comparison matrix expressed by

$$\tilde{A} = (\tilde{M}_{ij})_{n\times n} = \begin{bmatrix} (1,1,1) & (l_{12}, m_{12}, u_{12}) & \dots & (l_{1n}, m_{1n}, u_{1n}) \\ (l_{21}, m_{21}, u_{21}) & (1,1,1) & \dots & (l_{2n}, m_{2n}, u_{2n}) \\ & \cdot \ \cdot \ \cdot \ \cdot & & \\ & \cdot \ \cdot \ \cdot \ \cdot & & \\ & \cdot \ \cdot \ \cdot \ \cdot & & \\ (l_{n1}, m_{n1}, u_{n1}) & (l_{n2}, m_{n2}, u_{n2}) & \dots & (1,1,1) \end{bmatrix}$$

where $\tilde{a}_{ij} = (l_{ij}, m_{ij}, u_{ij}) = \tilde{a}_{ji}^{-1} = (\frac{1}{u_{ji}}, \frac{1}{m_{ji}}, \frac{1}{l_{ji}})$, for i,j = 1,…,n and $i \ne j$.

To calculate a priority vector of the above triangular fuzzy comparison matrix, Chang [9] suggested an extent analysis method, which is summarized as follows.

Firstly, sum up each row of the fuzzy comparison matrix $\tilde{A}$ by fuzzy arithmetic operations:

$$RS_i = \sum_{j=1}^{n} \tilde{M}_{ij} = (\sum_{j=1}^{n} \tilde{l}_{ij}, \sum_{j=1}^{n} \tilde{m}_{ij}, \sum_{j=1}^{n} \tilde{u}_{ij},), \ i = 1,\dots,n.$$

Secondly, normalize the above row sums by

$$\tilde{S}_i = \frac{RS_i}{\sum_{j=1}^{n} RS_j} = \left(\frac{\sum_{j=1}^{n} \tilde{l}_{ij}}{\sum_{k=1}^{n}\sum_{j=1}^{n} u_{kj}}, \frac{\sum_{j=1}^{n} \tilde{m}_{ij}}{\sum_{k=1}^{n}\sum_{j=1}^{n} m_{kj}}, \frac{\sum_{j=1}^{n} \tilde{u}_{ij}}{\sum_{k=1}^{n}\sum_{j=1}^{n} l_{kj}} \right), \quad i = 1,\dots,n.$$

Thirdly, compute the degree of possibility of $\tilde{S}_i \ge \tilde{S}_j$ by the following equation:

$$V(\tilde{S}_i \ge \tilde{S}_j) = \begin{cases} 1, & \text{if } m_i \ge m_j \\ \dfrac{u_i - l_j}{(u_i - m_i) + (m_j - l_j)}, & \text{if } l_j \ge u_i, \ i, j = 1,\dots,n; j \ne i \\ 0, & \text{other} \end{cases}$$

where $\tilde{S}_i = (l_i, m_i, u_i)$ and $\tilde{S}_j = (l_j, m_j, u_j)$. The definition of possibility degree is shown in **Figure 1.**

Fourthly, calculate the degree of possibility of $\tilde{S}_i$ over all the other (n - 1) fuzzy numbers by $V(\tilde{S}_i \ge \tilde{S}_j \,|\, j = 1,\dots,n; j \ne i) = \min_{j\in\{1,\dots,n\}, j\ne i} V(\tilde{S}_i \ge \tilde{S}_j), \ i = 1,\dots,n)$.

Finally, define the priority vector $W = (w_1,\dots,w_n)^T$ of the fuzzy comparison matrix $\tilde{A}$ as

$$w_i = \frac{V\left(\tilde{S}_i \ge \tilde{S}_j \,|\, j = 1,\dots,n; j \ne i\right)}{\sum_{k=1}^{n} V\left(\tilde{S}_k \ge \tilde{S}_j \,|\, j = 1,\dots,n; j \ne k\right)}, \quad i = 1,\dots,n.$$

It must be pointed out that the normalization formula is wrong. The correct normalization formula for a set of triangular fuzzy weights should be as follows:

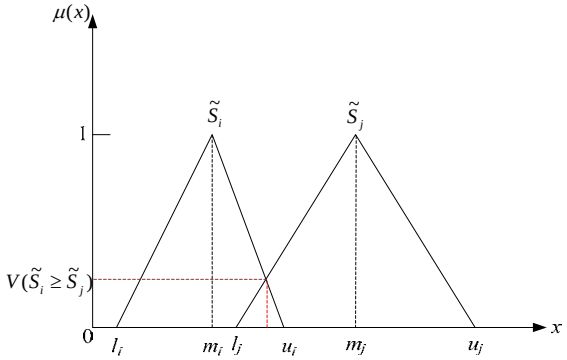


Figure 1. Definition of the degree of possibility of $V(\tilde{S}_i \ge \tilde{S}_j)$

$$\tilde{S}_i = \frac{RS_i}{\sum_{j=1}^{n} RS_j} = \left(\frac{\sum_{j=1}^{n} l_{ij}}{\sum_{j=1}^{n} \tilde{l}_{ij} + \sum_{k=1,k\neq i}^{n} \sum_{j=1}^{n} u_{kj}}, \frac{\sum_{j=1}^{n} m_{ij}}{\sum_{k=1}^{n} \sum_{j=1}^{n} m_{kj}}, \frac{\sum_{j=1}^{n} u_{ij}}{\sum_{j=1}^{n} u_{ij} + \sum_{k=1,k\neq i}^{n} \sum_{j=1}^{n} l_{kj}} \right), \quad i = 1,...,n.$$

Although Fuzzy AHP solved some of the problems of AHP, but still some problems arises:

Problem 1. The extent analysis method may assign a zero weight to a decision criterion or alternative, leading to the criterion or alternative not to be considered in decision analysis.

Problem 2. The weights determined by the extent analysis method do not represent the relative importance of decision criteria or alternatives and cannot be used as their priorities.

Problem 3. The extent analysis method may make a wrong decision and select the worst decision alternative as the best one when it is misused for solving a fuzzy AHP problem.

Problem 4. The extent analysis method cannot make full use of all the fuzzy comparison matrices information and may cause some useful fuzzy comparison matrices information to be wasted when it assigns an irrational zero weight to some useful decision criteria or sub-criteria.

Therefore, we propose an expert system which functions based on fuzzy logic, to improve decision making in uncertainties.

3. Fuzzy Logic

Fuzzy Logic (FL) is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded microcontrollers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make decisions, only much faster.

FL incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system.

FL requires some numerical parameters in order to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them. For example, a simple temperature control system could use a single temperature feedback sensor whose data is subtracted from the command signal to compute "error" and then time-differentiated to yield the error slope or rate-of-change-of-error, hereafter called "error-dot". Error might have units of degs F and a small error considered to be 2 F while a large error is 5 F. The "error-dot" might then have units of degs/min with a small error-dot being 5 F/min and a large one being 15 F/min. These values don't have to be symmetrical and can be "tweaked" once the system is operating in order to optimize performance. Generally, FL is so forgiving that the system will probably work the first time without any tweaking.

FL works as follows:

1) Define the control objectives and criteria: What am I trying to control? What do I have to do to control the system? What kind of response do I need? What are the possible (probable) system failure modes?

2) Determine the input and output relationships and choose a minimum number of variables for input to the FL engine (typically error and rate-of-change-of-error).

3) Using the rule-based structure of FL, break the control problem down into a series of IF X AND/OR Y THEN Z rules that define the desired system output response for given system input conditions. The number and complexity of rules depends on the number of input parameters that are to be processed and the number fuzzy variables associated with each parameter. If possible, use at least one variable and its time derivative. Although it is possible to use a single, instantaneous error parameter without knowing its rate of change, this cripples the system's ability to minimize overshoot for a step inputs.

4) Create FL membership functions that define the meaning (values) of Input/Output terms used in the rules.

5) Create the necessary pre- and post-processing FL routines if implementing in S/W, otherwise program the rules into the FL H/W engine.

6) Test the system, evaluate the results, tune the rules and membership functions, and retest until satisfactory results are obtained.

In 1973, Professor Lotfi Zadeh proposed the concept of linguistic or "fuzzy" variables. Think of them as linguistic objects or words, rather than numbers. The sensor input is a noun, e.g. "temperature", "displacement", "velocity", "flow", "pressure", etc. Since error is just the difference, it can be thought of the same way. The fuzzy variables themselves are adjectives that modify the variable (e.g. "large positive" error, "small positive" error, "zero" error, "small negative" error, and "large negative" error). As a minimum, one could simply have "positive", "zero", and "negative" variables for each of the parame-

ters. Additional ranges such as "very large" and "very small" could also be added to extend the responsiveness to exceptional or very nonlinear conditions, but aren't necessary in a basic system. Here, using fuzzy logic we define some rules that help the student to select the optimal department, course and teacher based on his age, average grade and skills.

4. Expert Systems

Knowledge-based systems are systems based on the methods and techniques of Artificial Intelligence. Their core components are the knowledge base and the inference mechanisms. Some particular types of knowledge-based systems are expert systems, case-based reasoning systems and neural networks.

Expert Systems (ES) are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI). AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behavior. It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine.

Of course, the term intelligence covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of those. But most progress to date in AI has been made in the area of problem solving; concepts and methods for building programs that reason about problems rather than calculate a solution.

AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called knowledge-based or expert systems. Often, the term expert systems is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often, the two terms, expert systems (ES) and knowledge-based systems (KBS), are used synonymously. Taken together, they represent the most widespread type of AI application. The area of human intellectual endeavor to be captured in an expert system is called the task domain. Task refers to some goal-oriented, problem-solving activity. Domain refers to the area within which the task is being performed. Typical tasks are diagnosis, planning, scheduling, configuration and design.

Building an expert system is known as knowledge engineering and its practitioners are called knowledge engineers. The knowledge engineer must make sure that the computer has all the knowledge needed to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as symbol patterns in the memory of the computer, that is, he (or she) must choose a knowledge representation. He must also ensure that the computer can use the knowledge efficiently by selecting from a handful of reasoning methods.

Every expert system consists of two principal parts: the knowledge base; and the reasoning, or inference, engine. The knowledge base of expert systems contains both factual and heuristic knowledge. Factual knowledge is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field.

Today there are two ways to build an expert system. They can be built from scratch, or built using a piece of development software known as a "tool" or a "shell". Before we discuss these tools, let's briefly discuss what knowledge engineers do. Though different styles and methods of knowledge engineering exist, the basic approach is the same: a knowledge engineer interviews and observes a human expert or a group of experts and learns what the experts know, and how they reason with their knowledge. The engineer then translates the knowledge into a computer-usable language, and designs an inference engine, a reasoning structure, that uses the knowledge appropriately. He also determines how to integrate the use of uncertain knowledge in the reasoning process, and what kinds of explanation would be useful to the end user.

Next, the inference engine and facilities for representing knowledge and for explaining are programmed, and the domain knowledge is entered into the program piece by piece. It may be that the inference engine is not just right; the form of knowledge representation is awkward for the kind of knowledge needed for the task; and the expert might decide the pieces of knowledge are wrong. All these are discovered and modified as the expert system gradually gains competence.

The discovery and accumulation of techniques of machine reasoning and knowledge representation is generally the work of artificial intelligence research. The discovery and accumulation of knowledge of a task domain is the province of domain experts. Domain knowledge consists of both formal, textbook knowledge, and experiential knowledge—the expertise of the experts.

Compared to the wide variation in domain knowledge, only a small number of AI methods are known that are useful in expert systems. That is, currently there are only a handful of ways in which to represent knowledge, or to make inferences, or to generate explanations. Thus, systems can be built that contain these useful methods without any domain-specific knowledge. Such systems are known as skeletal systems, shells, or simply AI tools.

Building expert systems by using shells offers significant advantages. A system can be built to perform a unique task by entering into a shell all the necessary knowledge about a task domain. The inference engine

that applies the knowledge to the task at hand is built into the shell. If the program is not very complicated and if an expert has had some training in the use of a shell, the expert can enter the knowledge himself.

Many commercial shells are available today, ranging in size from shells on PCs, to shells on workstations, to shells on large mainframe computers. They range in price from hundreds to tens of thousands of dollars, and range in complexity from simple, forward-chained, rule-based systems requiring two days of training to those so complex that only highly trained knowledge engineers can use them to advantage. They range from general-purpose shells to shells custom-tailored to a class of tasks, such as financial planning or real-time process control.

Although shells simplify programming, in general they don't help with knowledge acquisition. Knowledge acquisition refers to the task of endowing expert systems with knowledge, a task currently performed by knowledge engineers. The choice of reasoning method, or a shell, is important, but it isn't as important as the accumulation of high-quality knowledge. The power of an expert system lies in its store of knowledge about the task domain—the more knowledge a system is given, the more competent it becomes. Primarily, the benefits of ESs to end users include:

- A speed-up of human professional or semi-professional work—typically by a factor of ten and sometimes by a factor of a hundred or more.
- Within companies, major internal cost savings. For small systems, savings are sometimes in the tens or hundreds of thousands of dollars; but for large systems, often in the tens of millions of dollars and as high as hundreds of millions of dollars. These cost savings are a result of quality improvement, a major motivation for employing expert system technology.
- Improved quality of decision making. In some cases, the quality or correctness of decisions evaluated after the fact show a ten-fold improvement.
- Preservation of scarce expertise. ESs are used to preserve scarce know-how in organizations, to capture the expertise of individuals who are retiring, and to preserve corporate know-how so that it can be widely distributed to other factories, offices or plants of the company.
- Introduction of new products. A good example of a new product is a pathology advisor sold to clinical pathologists in hospitals to assist in the diagnosis of diseased tissue.

An expert system tool, or shell, is a software development environment containing the basic components of expert systems. Associated with a shell is a prescribed method for building applications by configuring and instantiating these components. Some of the generic components of a shell are shown in **Figure 2** and described below. The core components of expert systems are the knowledge base and the reasoning engine.

Knowledge base: A store of factual and heuristic knowledge. An ES tool provides one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use both frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

Reasoning engine: Inference mechanisms for manipulating the symbolic information and knowledge in the knowledge base to form a line of reasoning for solving a problem. The inference mechanism can range from simple modus pones backward chaining of IF-THEN rules to case-based reasoning.

Knowledge acquisition subsystem: A subsystem to help experts build knowledge bases. Collecting knowledge needed to solve problems and build the knowledge base continues to be the biggest bottleneck in building expert systems.

Explanation subsystem: A subsystem that explains the system's actions. The explanation can range from how the final or intermediate solutions were arrived at to justifying the need for additional data.

User interface: The means of communication with the user. The user interface is generally not a part of the ES technology, and was not given much attention in the past. However, it is now widely accepted that the user interface can make a critical difference in the perceived utility of a system regardless of the system's performance.

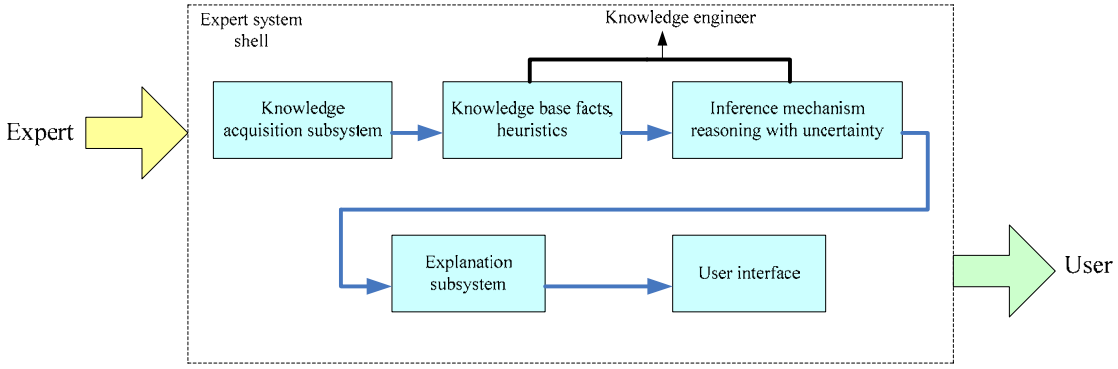


Figure 2. Basic components of expert system tools

5. Comparison between Expert System and Fuzzy AHP

Expert system has been applied for ranking [29]. Expert system in comparison with fuzzy AHP has the following advantages:

- The alternatives are analyzed using quantitative and qualitative criteria without normalization process
- More than seven alternatives can be processed against AHP which encountered problems in pairwise comparisons [30].
- By entrance of new alternatives, the ranking of the alternatives do not change
- The fuzzy expert system is able to consider a standard in evaluating the alternatives
- It is possible to apply group decision making of the experts in evaluating the alternatives
- The capability of sensitivity analysis for all the alternatives
- No limitation for evaluating many criteria
- The mistakes in computations such as the zero result will not occur in expert system
- The possibility of evaluating the alternatives using both quantitative and qualitative criteria
- The possibility of evaluating the alternatives while some information about some criteria are missing
- The possibility of keeping the same membership during the process of decision making [31].

In an expert system a membership function is proposed for criteria regarding to the experts idea. To propose an expert system the following steps should be taken:

1) Determining the objective, alternatives and criteria
2) Identifying the input and output variables
3) Proposing membership functions for input and output variables
4) Proposing rules to determine the relations between inputs and outputs
5) Selecting an appropriate inference mechanism
6) Placement of alternatives corresponding to each criteria
7) Extracting the evaluation result by the proposed expert system
8) Sensitivity analysis of evaluated alternatives

Net section presents a numerical illustration to indicate the application of the proposed expert system.

6. Numerical Illustrations

Here, we illustrate the proposed expert system in prioritizing four brands of mobile phone. We analyze HAD, IC, TA, HAM as alternatives using the criteria services, power of antenna, prestige, and price. The hierarchy of the model is shown in **Figure 3.**

The linguistic variables for criteria and their corresponded membership functions are as follows (**Figures 4-9**).

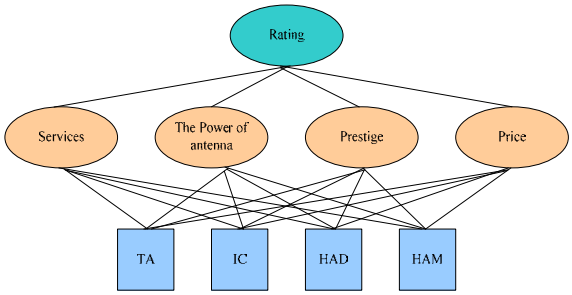


Figure 3. The hierarchy of the model

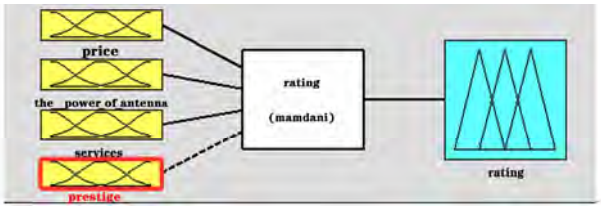


Figure 4. The inputs and outputs

Considering the experts the price has a Gaussian membership function with minimum price of 5000 and maximum of 700000.

For the power of antenna linguistic triangular fuzzy number (high, medium, low) is considered.

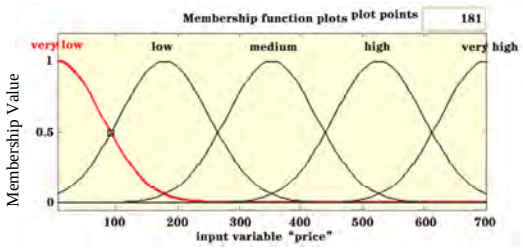


Figure 5. Price membership function

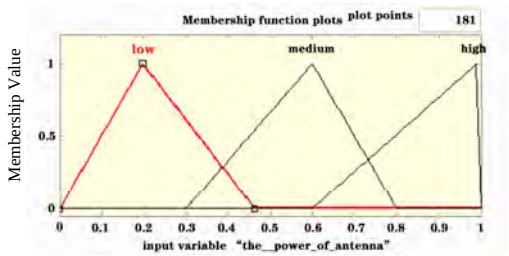


Figure 6. The power of antenna membership function

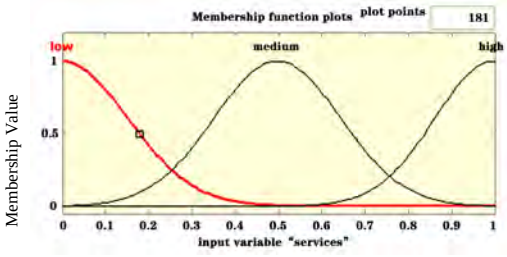


Figure 7. Services membership function

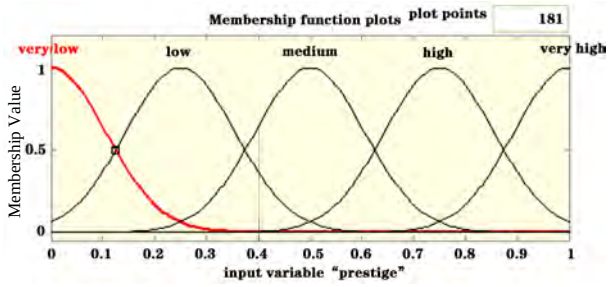


Figure 8. Prestige membership function

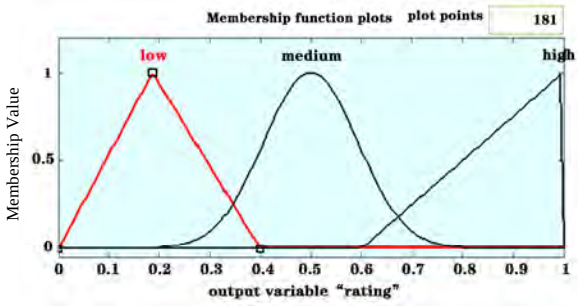


Figure 9. Rating membership function

The output of the system which is the evaluation result, is a combined linguistic fuzzy number with a Gaussian membership function for *medium* and triangular fuzzy membership function for *high* and *low*.

Regarding to the experts and taking the criteria into considerations, the following rules are derived:

1-If price is high Then rating medium.
2-If price is very low Then rating low.
3-If services is high Then rating high.
4-If services is medium Then rating low.
5-If services is low Then rating low.
6-If services is high Then rating high.
7-If the_power_of_antenna is high Then rating high.
8-If the_power_of_antenna is low Then rating low.
9-If the_power_of_antenna is medium Then rating low.
10-If price is medium Then rating high.
11-If price is very high Then rating low.
12-If price is low Then rating high.
13-If prestige is very low Then rating low.
14-If prestige is medium Then rating medium.
15-If prestige is very high Then rating high

Regarding to the proposed rules of the expert system, we evaluate the alternatives as follows. To facilitate the computations MATLAB package has been applied (Appendix A).

TA
Price: 200
Services: medium
Power of antenna: high
Prestige: medium
Output: 0.518

The graphical presentation is shown in **Figure 10**.

IC
Price: 15
Services: low
Power of antenna: high
Prestige: low
Output: 0.51

The graphical presentation is shown in **Figure 11.**

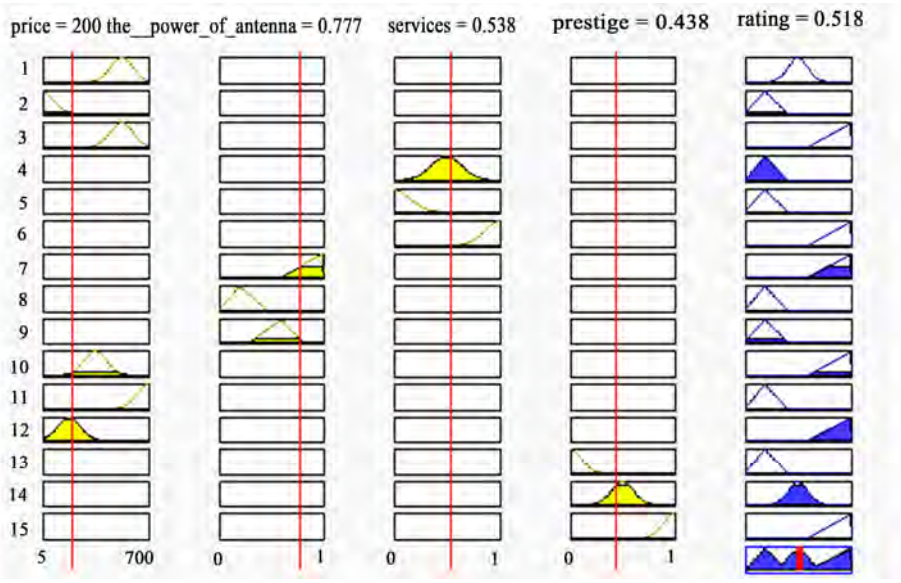


Figure 10. The rule viewer for TA

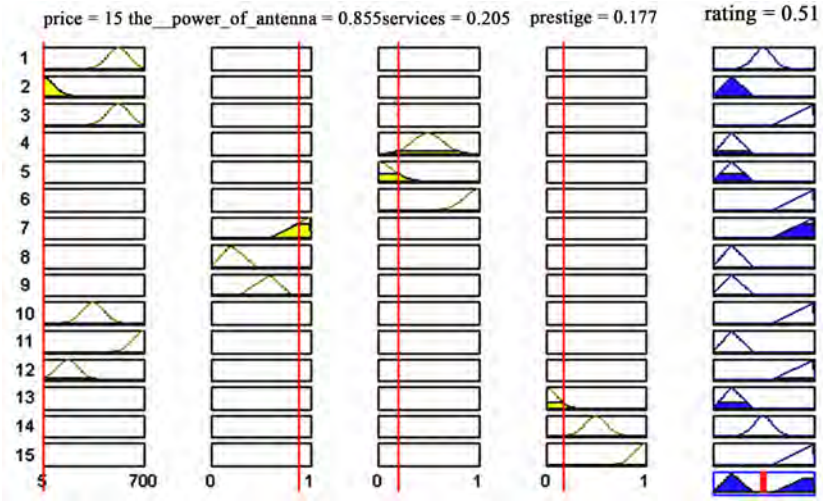


Figure 11. The rule viewer for IC

Figure 12. The rule viewer for HAD

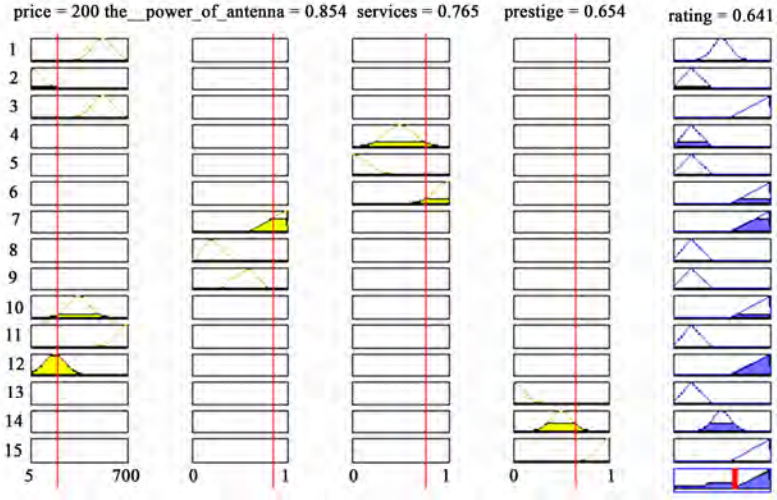


Figure 13. The rule viewer for HAM

HAD
Price: 500
Services: high
Power of antenna: high
Prestige: very high
Output: 0.654

The graphical presentation is shown in **Figure 12.**

HAM
Price: 200
Services: high
Power of antenna: high
Prestige: high
Output: 0.641

The graphical presentation is shown in **Figure 13**.

The ranking indicates the importance degree of each mobile brand.

7. Conclusions

In this paper, we developed a fuzzy expert system on the basis of rule base fuzzy logic to overcome the problems in AHP and Fuzzy AHP. The advantages of using expert system to prioritize the alternatives in comparison with fuzzy AHP are discussed. To present the validity and effectiveness of the proposed expert system a numerical example is illustrated.

REFERENCES

[1] T. L. Saaty, "Multicriteria Decision Making: The Analytic Hierarchy Process," RWS Publications, Pittsburgh, 1988.

[2] O. S. Vaidya and S. Kumar, "Analytic Hierarchy Process: An Overview of Applications," *European Journal of Operational Research*, Vol. 169, 2006, pp. 1-29.

[3] P. J. M. Van Laarhoven and W. Pedrycz, "A Fuzzy Extension of Saaty's Priority Theory," *Fuzzy Sets and Systems*, Vol. 11, 1983, pp. 229-241.

[4] Y. M. Wang, T. M. S. Elhag and Z. S. Hua, "A Modified Fuzzy Logarithmic Least Squares Method for Fuzzy Analytic Hierarchy Process," *Fuzzy Sets and Systems*, Vol. 157, 2006, pp. 3055-3071.

[5] J. J. Buckley, "Fuzzy Hierarchical Analysis," *Fuzzy Sets and Systems*, Vol. 17, 1985, pp. 233-247.

[6] D. Y. Chang, "Applications of the Extent Analysis Method on Fuzzy AHP," *European Journal of Operational Research*, Vol. 95, 1996, pp. 649-655.

[7] R. Xu, "Fuzzy Least-Squares Priority Method in the Analytic Hierarchy Process," *Fuzzy Sets and Systems*, Vol. 112, 2000, pp. 359-404.

[8] L. Mikhailov, "Deriving Priorities from Fuzzy Pairwise Comparison Judgments," *Fuzzy Sets and Systems*, Vol. 134, 2003, pp. 365-385.

[9] R. Csutora, and J. J. Buckley, "Fuzzy Hierarchical Analysis: The Lamda-Max Method," *Fuzzy Sets and Systems*, Vol. 120, 2001, pp. 181-195.

[10] F. T. Bozbura and A. Beskese, "Prioritization of Organizational Capital Measurement Indicators Using Fuzzy AHP," *International Journal of Approximate Reasoning*, Vol. 44, 2007, pp. 124-147.

[11] F. T. Bozbura, A.Beskese and C. Kahraman, "Prioritization of Human Capital Measurement Indicators Using Fuzzy AHP," *Expert Systems with Applications*, Vol. 32, 2007, pp. 1100-1112.

[12] C. E. Bozdag, C. Kahraman and D. Ruan, "Fuzzy Group Decision Making for Selection among Computer Integrated Manufacturing Systems," *Computers in Industry*, Vol. 51, 2003, pp. 13-29.

[13] G. Bu¨yu¨ko¨zkan, "Multi-Criteria Decision Making for E-Marketplace Selection," *Internet Research*, Vol. 14, No. 2, 2004, pp. 139-154.

[14] G. Bu¨yu¨ko¨zkan, T. Ertay, C. Kahraman and D. Ruan, "Determining the Importance Weights for the Design Requirements in the House of Quality Using the Fuzzy Analytic Network Approach," *International Journal of Intelligent Systems*, Vol. 19, 2004, pp. 443-461.

[15] G. Bu¨yu¨ko¨zkan, C. Kahraman and D. Ruan, "A Fuzzy Multi-Criteria Decision Approach for Software Development Strategy Selection," *International Journal of General Systems*, Vol. 33, 2004, pp. 259-280.

[16] F. T. S. Chan and N. Kumar, "Global Supplier Development Considering Risk Factors Using Fuzzy Extended AHP Based Approach," *Omega*, Vol. 35, 2007, pp. 417-431.

[17] T. Ertay, G. Bu¨yu¨ko¨zkan, C. Kahraman and D. Ruan, "Quality Function Deployment Implementation Based on Analytic Network Process with Linguistic Data: An Application in Automotive Industry," *Journal of Intelligent and Fuzzy Systems*, Vol. 16, 2005, pp. 221-232.

[18] Y. C. Erensal, T. O¨ ncan and M. L. Demircan, "Determining Key Capabilities in Technology Management Using Fuzzy Analytic Hierarchy Process: A Case Study of Turkey," *Information Sciences*, Vol. 176, 2006, pp. 2755-2770.

[19] C Kahraman, U. Cebeci and D. Ruan, "Multi-Attribute Comparison of Catering Service Companies Using Fuzzy AHP: The Case of Turkey," *International Journal of Production Economics*, Vol. 87, 2004, pp. 171-184.

[20] C. Kahraman, U. Cebeci and Z. Ulukan, "Multi-Criteria Supplier Selection Using Fuzzy AHP," *Logistics Information Management*, Vol. 16, No. 6, 2003, pp. 382-394.

[21] C. Kahraman, T. Ertay, and G. Bu¨yu¨ko¨zkan, "A Fuzzy Optimization Model for QFD Planning Process Using Analytic Network Approach," *European Journal of Operational Research*, Vol. 171, 2006, pp. 390-411.

[22] C. Kahraman, D. Ruan and I. Dogan, "Fuzzy Group Decision-Making for Facility Location Selection," *Information Sciences*, Vol. 157, 2003, pp. 135-153.

[23] O. Kulak and C. Kahraman, "Fuzzy Multi-Attribute Selection among Transportation Companies Using Axiomatic Design and Analytic Hierarchy Process," *Information Sciences*, Vol. 170, 2005, pp. 191-210.

[24] C. K. Kwong, and H. Bai, "Determining the Importance

Weights for the Customer Requirements in QFD Using a Fuzzy AHP with an Extent Analysis Approach," *IIE Transactions*, Vol. 35, 2003, pp. 619-626.

[25] Y. C. Tang and M. Beynon, "Application and Development of a Fuzzy Analytic Hierarchy Process within a Capital Investment Study," *Journal of Economics and Management*, Vol. 1, 2005, pp. 207-230.

[26] E. Tolga, M. L. Demircan and C. Kahraman, "Operating System Selection Using Fuzzy Replacement Analysis and Analytic Hierarchy Process," *International Journal of Production Economics*, Vol. 97, 2005, pp. 89-117.

[27] F. Tu¨ysu¨ z, and C. Kahraman, "Project Risk Evaluation Using a Fuzzy Analytic Hierarchy Process: An Application to Information Technology Projects," *International Journal of Intelligent Systems*, Vol. 21, No. 6, 2006, pp. 559-584.

[28] K. J. Zhu, Y. Jing and D. Y. Chang, "A Discussion on Extent Analysis Method and Application of Fuzzy AHP," *European Journal of Operational Research*, Vol. 116, 1999, pp. 450-456.

[29] S. Malagoli, C. A. Magni and G. Mastroleo, "The Use of Fuzzy Logic and Expert Systems for Rating and Pricing Firms," *Social Science Research Network*, Vol. 33, No. 11, 2007, pp. 77-120.

[30] E. Turban, J. E. Aronson and T. P. Liang, "Decision Support Systems and Intelligent Systems," 7th Edition, Prentice Hall of India, New Delhi, 2007.

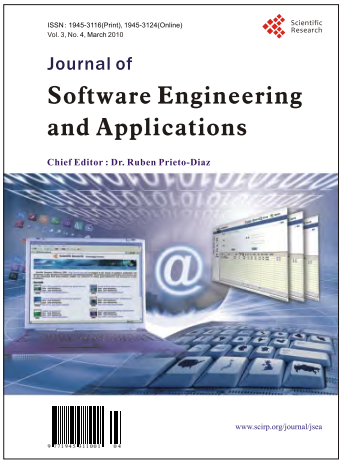
[31] Y. M. Wang, Y. Luo and Z. Hua, "On the Extent Analysis Method for Fuzzy AHP and its Applications," *European Journal of Operational Research*, Vol. 186, 2008, pp. 735-747.

Appendix A

Here, some useful MATLAB commands to work with the proposed fuzzy inference system (FIS) which is based on Mamdani are presented:

```
[System]
Name='AHP mobile'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=15
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
```


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 April 2010 in SciRes (http://www.SciRP.org/journal/jsea).

TABLE OF CONTENTS

Volume 3, Number 4 **April 2010**

