ISSN: 1945-3116 Vol. 3, No. 6, June 2010

Scientific Research

Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

ISSN: 1945-3116

9 771945 311001 06

www.scirp.org/journal/jsea

Journal Editorial Board

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

J. Software Engineering & Applications, **2010, 3, 517-627**
Published Online June 2010 in SciRes (http://www.SciRP.org/journal/jsea/)

TABLE OF CONTENTS

Volume 3 Number 6 June 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, 517-524
doi:10.4236/jsea.2010.36059 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

The Topological Conditions: The Properties of the Pair of Conjugate Tress

Luis Hernandez-Martinez, Arturo Sarmiento-Reyes, Miguel A. Gutierrez de Anda

Electronics Department, National Institute for Astrophysics, Optics and Electronics, P.O. Puebla, Pue. Mexico.
Email: {luish,jarocho,mdeanda}@inaoep.mx

Received November 29th, 2009; revised January 4th, 2010; accepted January 6th, 2010.

ABSTRACT

This paper presents some important properties emanating from the pair of conjugate trees. The properties are obtained by resorting to the fundamental loops and cutsets in the circuit topology. The existence of such a pair is one of the conditions for a nonlinear resistive circuit to have one and only one DC solution.

Keywords*: *Pair of Conjugate Trees, Topological Conditions, Fundamental Cutsets and Loops, Analogue Circuits

1. Introduction

Graph Theory is used for the study of real-world Systems possessing a binary relation between elements of a certain set within the system description. Among other discipline (Circuit Theory) has received outstanding contributions from the study of graphs. Some of the contributions may be found in the solution to specific problems related to electrical network analysis, nonlinear circuit theory, circuit diagnosis and circuit synthesis [1,2].

A line of research developed in recent years has attempted to determine the relationships between the topology of a circuit and its functionality, which has derived in a deeper knowledge in the general problem of nonlinear circuits [3,4]. One important work has been reported in [5] and [6], where a topological criterion for the existence and uniqueness of the solution of linear circuits has been proposed. This criterion is based on two definitions of Graph Theory: *the pair of conjugate trees* and *the uniform partial orientation of the resistors*.

In this paper, the attention is focused on several properties of the pair of conjugate trees; these are highlighted by looking at the resulting loop and cutset matrices.

2. Preliminary Considerations

The scope of the work is restricted to certain types of basic circuit elements: resistors (R), voltage sources (V), current sources (I), nullators (O) and norators (P). However, it must be emphasised that these circuit components are used to model the original circuit through an equivalent circuit denoted as the nonlinear resistive circuit structure. Besides, an important condition concerning singular elements must be fulfilled: nullators and norators must appear in the circuit in equal numbers.

Other devices can be used by building up equivalent schemes consisting of models containing the set of basic elements. As an example, the **Figure 1** shows the equivalent circuit of a bipolar transistor.

Herein, we retake two definitions from [3] in order to set up the further development of the diagnostic method:

Definition 1

A nonlinear resistive circuit structure is a graph whose branches are labeled with the following six element types.

- Independent voltage sources
- Independent current sources
- V-resistors (voltage controlled)
- I-resistors (current controlled)
- Nullators
- Norators

A circuit **C** has a structure **S** if the graph of **C** and **S** coincide and if **C** has elements of the type prescribed by **S** on each branch.

A nonlinear resistor can be substituted by one of the following equivalents:

1) The element is converted into a linear resistor if it is strictly increasing.

2) The element is converted into a voltage source if it is voltage-controlled.

3) The element is converted into a current source if it is current-controlled. As shown in the **Figure 2.**

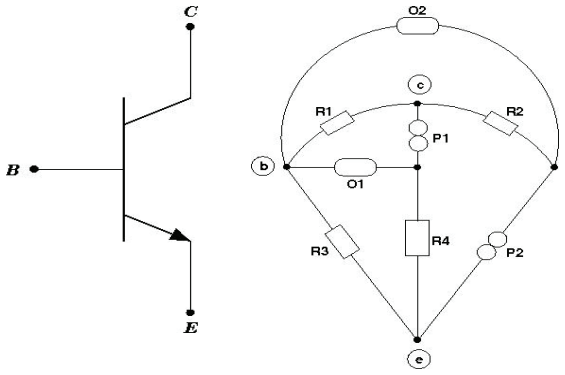


Figure 1. Equivalent circuito of a bipolar transistor

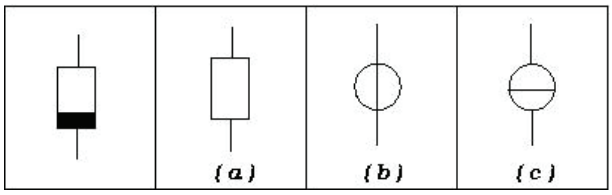


Figure 2. Equivalent of the nonlinear resistor

Definition 2

Two trees t' and t'' of a nonlinear resistive circuit structure constitute a pair of conjugate trees if:

- t' is composed of all norators, all voltage sources and a subset of the resistors, and
- t'' is composed of all nullators, all voltage sources and the same resistors as t'.

The subsets of the resistors may also be empty or contain all resistors.

The **Figure 3** shows the pair of conjugate trees formed according to the Definition 2. On one hand, the tree t' is formed by the norators, both voltage sources and the resistor $R1$, as given in the **Figure 3(a)**. On the other hand, the tree t'' is formed by the nullator, both voltage sources and the same resistor as given in the **Figure 3(b)**.

3. Properties of the Pair (t', t'')

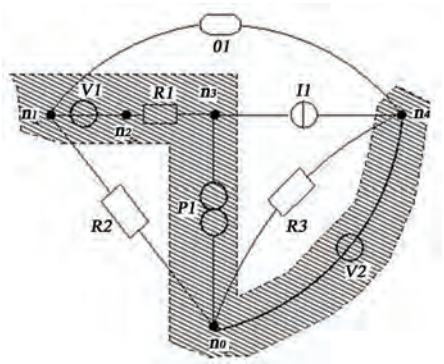
From the definitions above, two associated graphs of the same linear structure can be derived, namely g' and g''. It yields:

$$g' = t' \cup \tilde{t}_c \quad g'' = t' \cup \tilde{t}_c \tag{1}$$

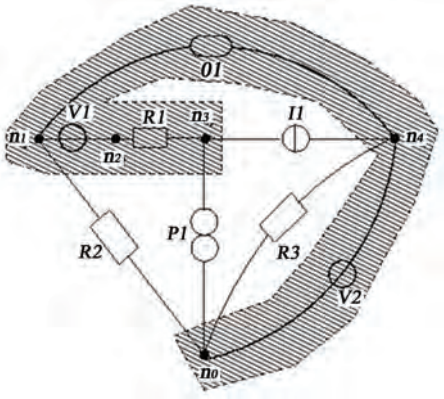
These graphs are depicted in **Figure 4**. However, it must be noticed that in fact, apart from the consideration of different trees, the relationship holds:

$$g' = g'' \tag{2}$$

As a result of the definitions given in Section 2, the pair of conjugate trees is formed as:



(a) Tree t'



(b) Tree t''

Figure 3. The pair of conjugate trees

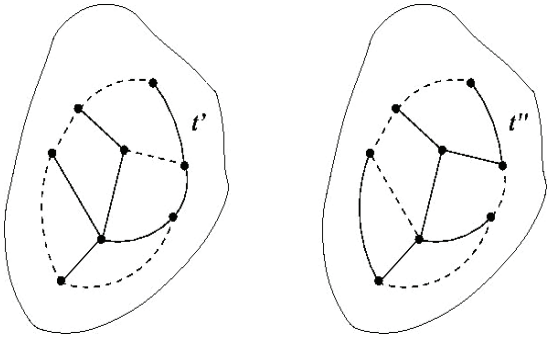


Figure 4. The graphs of a linear structure associated to the pair (t', t'')

$$t = \{P, V, R\} \quad t = \{O, V, R_a\} \tag{3}$$

where P, O, and V are the branches of all norators, all nullators and all voltage sources respectively. In addition, R_a is the set of branches of the common resistors $\in t'$ and t''. Furthermore, two pairs of conjugate co-trees ($\tilde{t}$, $\tilde{t}$) arise. They are formed as:

$$\tilde{t} = \{O, R_b\} \quad \tilde{t} = \{P, R_b\} \tag{4}$$

where R_b is the set of branches of those resistors not in t' nor in t''. It clearly results that the complete set of resistors R, is formed by:

$$R = R_a \cup R_b \tag{5}$$

Some dimensions may be mentioned now:

p – total number of norators

o – total number of nullators

v – total number of voltage sources

a – total number of resistors $\in R_a$

b – total number of resistors $\in R_b$

r – total number of resistors $\in R$

i.e. $r = a + b$

l – total number of fundamental loops

c – total number of cutsets

with the additional requirement of $p = o$. Besides, some variables must be defined:

u_p – norators voltages

u_o – nullator voltages

u_v – voltage sources

u_a – resistor voltages of R_a

u_b – resistor voltages of R_b

i_p – norators currents

i_o – nullators currents

i_v – current sources

i_a – resistor currents of R_a

i_b – resistor currents of R_b

4. Properties

Hereafter, the properties of the pair of conjugate trees (t', t'') are obtained by resorting to the fundamental loops and fundamental cutsets of the associated graphs. Because only one branch of the co-tree may be present in a fundamental loop, the structure of the fundamental loop matrix has the form:

$$C = [C_T | C_L] = [C_T | I_L] \tag{6}$$

where C_T is the tree-part of the fundamental loop matrix, and $C_L = I_L$ is the co-tree part. Because only one branch of the tree may be present in a fundamental cutset, the structure of the fundamental cutset matrix has the form:

$$D = [D_T | D_L] = [I_T | D_L] \tag{7}$$

where $D_T = I_T$ is the tree-part of the fundamental cutset matrix, and D_L is the co-tree part. In the following, Kirchoff's Laws for both graphs are analyzed in order to determine the properties of the loops and cutsets.

4.1 KVL for g'

Since the co-tree branch present in a fundamental loop must belong either to O or R_b, then the matrix in Equation (6) can have the form:

$$C = \left[\begin{array}{c|c} C_T & \begin{matrix} I_o & \\ & I_b \end{matrix} \end{array}\right]$$

where I_o and I_b are identity matrices of order o and b respectively. Two types of loops arise:

l'_o – a loop in g' having a link from O

l'_b – a loop in g' having a link from R_b

Furthermore, C_T can also be partitioned, which yields:

$$C = \left[\begin{array}{ccc|cc} C_{op} & C_{ov} & C_{oa} & I_a & \\ C_{bp} & C_{bv} & C_{ba} & & I_b \end{array}\right]$$

where every submatrix denoted as C_{xy} is a matrix of size x × y. Since o = p, C_{op} is a square matrix. Moreover, the size of C'_T is $(o+b)\times(p+v+a)$. KVL is given as:

$$C'u = 0$$

The partitioning above allows us to establish KVL as:

$$\left[\begin{array}{ccc|cc} C_{op} & C_{ov} & C_{oa} & I_o & \\ C_{bp} & C_{bv} & C_{ba} & & I_b \end{array}\right] \left[\begin{array}{c} u_p \\ u_v \\ \underline{u_a} \\ u_o \\ u_b \end{array}\right] = 0 \tag{8}$$

which is a system of $(o+b)$ loop equations in $(p+v+a+o+b)$ branch voltages. For the nullors, $U_o = 0$, then KVL can be re-written as:

$$\left[\begin{array}{ccc|c} C_{op} & C_{ov} & C_{oa} & \\ C_{bp} & C_{bv} & C_{ba} & I_b \end{array}\right] \left[\begin{array}{c} u_p \\ u_v \\ \underline{u_a} \\ u_b \end{array}\right] = 0 \tag{9}$$

4.2 KCL for g'

Since the tree branch present in a fundamental cutset must belong either to **P** or **V** or R_a, then the matrix in Equation (7) can have the form:

$$D = \left[\begin{array}{c|c} \begin{matrix} I_p & & \\ & I_v & \\ & & I_a \end{matrix} & D_L \end{array}\right]$$

where I_p, I_v and I_a are identity matrices of order

p , v and a respectively. Three types of cutsets arise:

c'_p – a cutest in g' having a twig from P

c'_v – a cutest in g' having a twig from V

c'_a – a cutest in g' having a twig from R_a

Furthermore, D'_L can also be partitioned, which yields:

$$D=\begin{bmatrix} I_p & & \\ & I_v & \\ & & I_a \end{bmatrix}\begin{vmatrix} D_{po} & D_{pb} \\ D_{vo} & D_{vb} \\ D_{ao} & D_{ab} \end{vmatrix}$$

where every submatrix denoted as D'_{xy} is a matrix of size x×y . Since $o=p$, D_{po} is a square matrix. Moreover, the size of D'_L is $(p+v+a)\times(o+b)$. KCL is given as:

$$Di=o$$

The partitioning above allows us to establish KCL as:

$$\begin{bmatrix} I_p & & \\ & I_v & \\ & & I_a \end{bmatrix}\begin{vmatrix} D_{pa} & D_{pb} \\ D_{vo} & D_{vb} \\ D_{ao} & D_{ab} \end{vmatrix}\begin{bmatrix} i_p \\ i_v \\ i_a \\ \hline i_o \\ i_b \end{bmatrix}=0 \qquad (10)$$

which is a system of $(p+v+b)$ cutset equations in $(p+v+a+o+b)$ branch currents. For the nullors, $i_o=0$, then KCL can be re-written as:

$$\begin{bmatrix} I_p & & \\ & I_v & \\ & & I_a \end{bmatrix}\begin{vmatrix} D_{pb} \\ D_{vb} \\ D_{ab} \end{vmatrix}\begin{bmatrix} i_p \\ i_v \\ \underline{i_a} \\ i_b \end{bmatrix}=0 \qquad (11)$$

Based on the orthogonality relationship:

$$C_T=D_T^T$$

where D'^{T}_{T} stands for the transpose of D'_T (see the **Figure 5**), the following equalities arise:

$$C'^{T}_{op}=D_{po} \quad C'^{T}_{ov}=D_{vo} \quad C'^{T}_{oa}=D'_{av}$$

$$C'^{T}_{bp}=D'_{pb} \quad C'^{T}_{bv}=D'_{vb} \quad C'^{T}_{ba}=D'_{ab} \qquad (12)$$

In order to illustrate the properties of the work, consider the linear circuit given in **Figure 6**, the fundamental loop and cutset matrices are given by:

C'=

	P_1	V_1	V_2	R_1	O_1	R_2	R_3
I'_1	1	1	1	1	1		
I'_2	1	1		1		1	
I'_3			1				1

D'=

	P_1	V_1	V_2	R_1	O_1	R_2	R_3
c'_1	1				1	1	0
c'_2		1			1	1	0
c'_3			1		1	0	1
c'_4				1	1	1	0

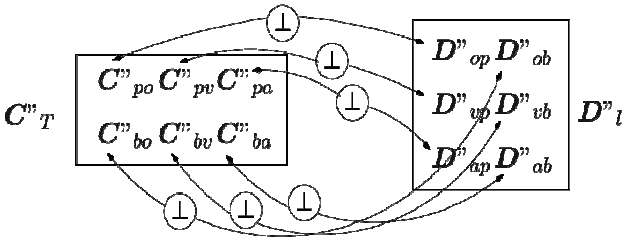


Figure 5. Orthogonality in g'

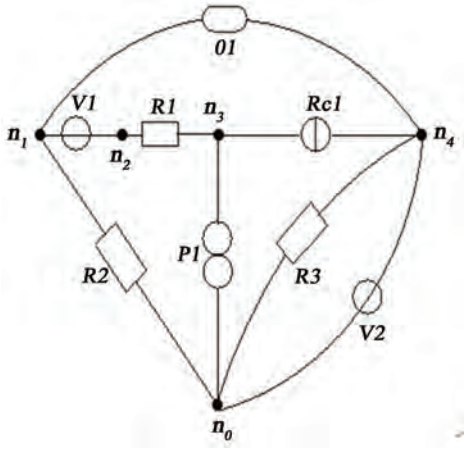


Figure 6. Case of study

where the columns have been labelled with the element names and the rows with the loop and cutsets respectively. The labels however, do not belong to the matrix.

4.3 KVL for g''

Since the co-tree branch present in a fundamental loop must belong either to P or R_b , then the matrix in Equation (6) can have the form:

$$C''=\left[\begin{array}{c|cc} C''_T & I_p & \\ & & I_b \end{array}\right]$$

where I_p and I_b are identity matrices of order P and b respectively. Two types of loops arise:

l''_p – a loop in g'' having a link from P

l''_b – a loop in g'' having a link from R_b

Furthermore, C''_T can also be partitioned:

$$C''=\left[\begin{array}{ccc|cc} C''_{po} & C''_{pv} & C''_{pa} & I_p & \\ C''_{bo} & C''_{bv} & C''_{ba} & & I_b \end{array}\right]$$

where every submatrix denoted as C''_{xy} is a matrix of size $x\times y$.

Since, C''_{po} is a square matrix. Moreover, the size of C''_T is $(p+b)\times(o+v+a)$. KVL is given as:

$$C''u=0$$

The partitioning above allows us to establish KVL as:

$$\left[\begin{array}{ccc|cc} C''_{po} & C''_{pv} & C''_{pa} & I_p & \\ C''_{bo} & C''_{bv} & C''_{ba} & & I_b \end{array}\right]\begin{bmatrix} u_o \\ u_v \\ u_a \\ \hline u_p \\ u_b \end{bmatrix}=0 \qquad (13)$$

which is a system of $(o+b)$ loop equations in $(o+v+a+p+b)$ branch voltages. For the nullors, $u_o=0$, then KVL can be re-written as:

$$\left[\begin{array}{ccc|cc} C''_{pv} & C''_{pa} & & I_p & \\ C''_{bv} & C''_{ba} & & & I_b \end{array}\right]\begin{bmatrix} u_v \\ u_a \\ \hline u_p \\ u_b \end{bmatrix}=0 \qquad (14)$$

4.4 KCL for g''

Since the tree branch present in a fundamental cutset must belong either to O or V or R_a, then the matrix in Equation (7) can have the form:

$$D''=\left[\begin{array}{ccc|c} I_o & & & \\ & I_v & & D''_L \\ & & I_a & \end{array}\right]$$

where I_o, I_v and I_a are identity matrices of order o, v and a respectively. Three types of cutsets arise:

c''_o – a cutset in g'' having a twig from O

c''_v – a cutset in g'' having a twig from V

c''_a – a cutset in g'' having a twig from R_a

Furthermore, D''_L can also be partitioned:

$$D''=\left[\begin{array}{ccc|cc} I_o & & & D''_{op} & D''_{ob} \\ & I_v & & D''_{vp} & D''_{vb} \\ & & I_a & D''_{ap} & D''_{ab} \end{array}\right]$$

where every submatrix denoted as D''_{xy} is a matrix of size $x\times y$. Since $o=p$, D''_{op} is a square matrix. Moreover, the size of D''_L is $(o+v+a)\times(p+b)$. KCL is given as:

$$D''i=0$$

The partitioning above allows us to establish KCL as:

$$\left[\begin{array}{ccc|cc} I_o & & & D''_{op} & D''_{ob} \\ & I_v & & D''_{vp} & D''_{vb} \\ & & I_a & D''_{ap} & D''_{ab} \end{array}\right]\begin{bmatrix} i_o \\ i_v \\ \hline i_a \\ i_p \\ i_b \end{bmatrix}=0 \qquad (15)$$

which is a system of $(o+v+b)$ cutset equations in $(o+v+a+p+b)$ branch currents. For the nullors, $i_o=0$, then KCL can be re-written as:

$$\left[\begin{array}{ccc|cc} I_v & & & D''_{op} & D''_{ob} \\ & & & D''_{vp} & D''_{vb} \\ & I_a & & D''_{ap} & D''_{ab} \end{array}\right]\begin{bmatrix} i_v \\ \hline i_a \\ i_p \\ i_b \end{bmatrix}=0 \qquad (16)$$

Based on the orthogonality relationship:

$$C''_T=D''^T_T$$

where D''^T_T stands for the transpose of D''_T (see the **Figure 7**), the following equalities arise:

$$C''^T_{po}=D''_{op}\quad C''^T_{vo}=D''_{ov}\quad C''^T_{ao}=D''_{oa}$$

$$C''^T_{pb}=D''_{bp}\quad C''^T_{vb}=D''_{bv}\quad C''^T_{ab}=D''_{ba} \qquad (17)$$

For the small circuit shown in **Figure 6**, the fundamental loop and cutset matrices are given as:

C"=

	O_1	V_1	V_2	R_1	P_1	R_2	R_3
l''_1	1	1	1	1	1		
l''_2	1	0	1	0		1	
l''_3	0	0	1	0			1

D"=

	P_1	V_1	V_2	R_1	O_1	R_2	R_3
c''_1	1				1	1	0
c''_2		1			1	0	0
c''_3			1		1	1	1
c''_4				1	1	0	0

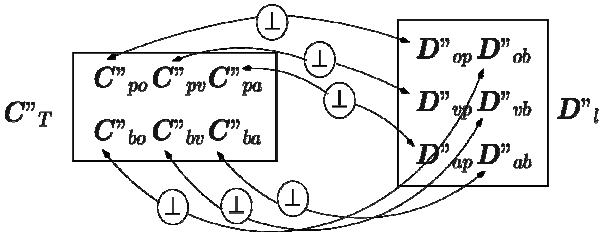


Figure 7. Orthohonality in g''

where the columns have been labelled with the element names and the rows with the loop and cutsets respectively. The labels however, do not belong to the matrix.

4.5 Loop Equations of g' and g''

The KVL Equations (8) and (13) are repeated here for easy reading:

$$\left[\begin{array}{ccc|c} C_{op} & C_{ov} & C_{oa} & I_o \\ C_{bp} & C_{bv} & C_{ba} & \end{array}\right. \left.\begin{array}{c} \\ I_b \end{array}\right] \begin{bmatrix} u_p \\ u_v \\ u_a \\ \hline u_o \\ u_b \end{bmatrix} = 0$$

$$\left[\begin{array}{ccc|c} C''_{po} & C''_{pv} & C''_{pa} & I_p \\ C''_{bo} & C''_{bv} & C''_{ba} & \end{array}\right. \left.\begin{array}{c} \\ I_b \end{array}\right] \begin{bmatrix} u_o \\ u_v \\ u_a \\ \hline u_p \\ u_b \end{bmatrix} = 0$$

Although these equations correspond to the fundamental loops of the graphs g' and g'', under the selection of *t'* and *t"* respectively, these equations refer in fact to the same graph and handle the same set of variables.

This means that both KVL equations contain redundant information. As can be observed schematically in the **Figure 8**, where the fundamental loop l''_1 in g'' results from the combination of loops l''_1 and l''_2 in g'. Therefore, it is possible to establish the next:

Statement 1

A fundamental loop in g' (resp. g'') may result from a combination of one or more fundamental loops in g'' (resp. g').

E.0 *A digression on the dimensions*

The considerations above lead us to define some special loops that may exist, namely:

L' – the longest loop(s) in g'

λ' – the shortest loop(s) in g'

L'' – the longest loop(s) in g''

λ'' – the shortest loop(s) in g''

The maximum lengths of longest loops are given as:

$$\max D(L) < (p+v+a)+1$$

$$\max D(L") < (o+v+a)+1$$

where $\max D$ stands for "the minimum length of". Because, $o = p$ then both bounds are the same, *i.e.*:

$$\max D(L) = \max D(L) = \max D(L") < (o+v+a)+1 \quad (18)$$

In the case that the longest loops have the maximum length, it occurs that:

$$L = L'' \quad (19)$$

i.e., a fundamental loop that appears in both graphs.

The minimum lengths of the shortest loops are given by:

$$\min D(\lambda) > \min(p,v,a)+1$$

$$\min D(\lambda") > \min(o,v,a)+1$$

where minD stands for "the minimum length of".

Because $o = p$, then both bounds are the same, *i.e.*:

$$\min D(\lambda) = \min D(\lambda) = \min D(\lambda") > \min(o,v,a)+1 \quad (20)$$

The minimum value for $\min(o,v,a)+1$ may be 2, *i.e.*, a loop formed by a parallel combination of two elements, then chances are that more than one shortest loop exist and also that a longer loop can be the result of a linear combination of several short loops. For example, these bounds are:

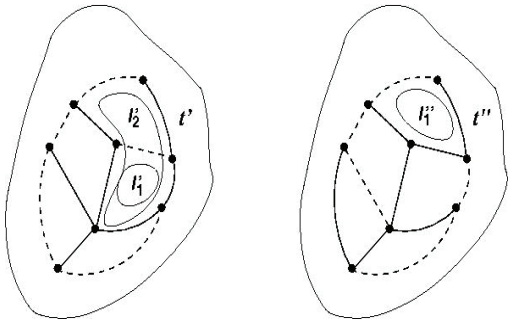


Figure 8. Redundant fundamental loops

$$\max\ D(L) < 5$$

$$\min\ D(\lambda) > 2$$

and the longest loops are given as:

$$L = l_1 = '\{P_1, V_1, V_2, R_1, O_1\}$$

$$L" = l"_1 = \{O_1, V_1, V_2, R_1, P_1\}$$

In addition, the shortest loops are given as:

$$\lambda = I_3 = \{V_2, R_3\}$$

$$\lambda" = I"_3 = \{V_2, R_3\}$$

i.e., in fact, the same shortest and longest loops arise in both graphs.

Because the Equations (8) and (13) refer in fact to the same graph, they can be combined in a single KVL. By re-ordering this equation according to the tree branches in *t'*, it yields:

$$\left[\begin{array}{ccc|c|c} C_{op} & C_{ov} & C_{oa} & I_o & \\ C_{bp} & C_{bv} & C_{ba} & & I_b \\ \hline I_p & C"_{pv} & C"_{pa} & C"_{po} & 0_{pb} \\ 0_{bp} & C"_{bv} & C"_{ba} & C"_{bo} & I_b \end{array}\right] \left[\begin{array}{c} u_p \\ u_v \\ u_a \\ u_o \\ u_b \end{array}\right] = 0 \qquad (21)$$

which is a system of $2p+2b$ loop equations, some of them being linearly dependent. Roughly speaking, some row-vectors in the equation above are wasted.

The number of linearly independent equations is given by the following:

Statement 2

The number of linearly independent loop equations is determined by:

$$= Rank\left(\left[\begin{array}{ccc|c|c} C_{op} & C_{ov} & C_{oa} & I_o & \\ C_{bp} & C_{bv} & C_{ba} & & I_b \\ \hline I_p & C"_{pv} & C"_{pa} & C"_{po} & 0_{pb} \\ 0_{bp} & C"_{bv} & C"_{ba} & C"_{bo} & I_b \end{array}\right]\right) \qquad (22)$$

Besides, the linearly independent loop equations are given by the following:

Statement 3

The linearly independent loop equations are determined by:

$$= Basis\left(\left[\begin{array}{ccc|c|c} C_{op} & C_{ov} & C_{oa} & I_o & \\ C_{bp} & C_{bv} & C_{ba} & & I_b \\ \hline I_p & C"_{pv} & C"_{pa} & C"_{po} & 0_{pb} \\ 0_{bp} & C"_{bv} & C"_{ba} & C"_{bo} & I_b \end{array}\right]\right) \qquad (23)$$

which constitutes in fact the row space of the matrix.

For the example of the previous section, the composed matrix of the Equation (21) is given as:

C_{both} =

	P_1	V_1	V_2	R_1	O_1	R_2	R_3
l'_1	1	1	1	1	1	0	0
l'_2	1	1	0	1	0	1	0
l'_3	0	0	1	0	0	0	1
$l"_1$	1	1	1	1	1	0	0
$l"_2$	0	0	1	0	1	1	0
$l"_3$	0	0	1	0	0	0	1

where the labels do not belong to the matrix. The **Figure 9** shows the graph with both trees and the set of linearly independent loops. It clearly results that:

$$l"_1 = l_1 \quad \text{and} \quad l"_3 = l_3$$

i.e., the shortest and longest loops. Therefore, the row basis of the matrix above is given as:

C_{base}=

	P_1	V_1	V_2	R_1	O_1	R_2	R_3
l'_1	1	1	1	1	1	0	0
l'_2	1	1	0	1	0	1	0
l'_3	0	0	1	0	0	0	1
$l"_2$	0	0	1	0	1	1	0

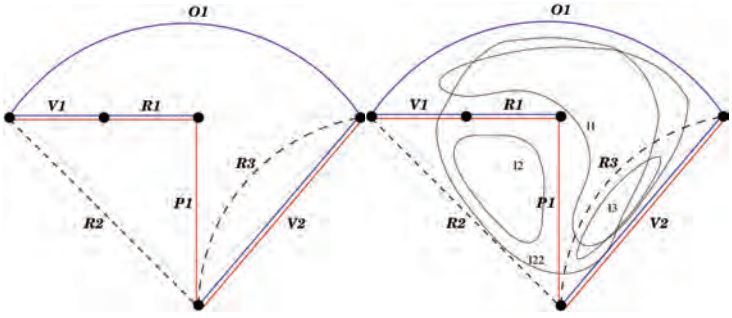


Figure 9. Combined graph & independent loops

5. Conclusions

A set of properties of the pair of conjugate trees of within the graph emanating from nonlinear resistive circuits has been presented. This approach is obtained by lookinkg at the resulting loop and cutset matrices.

REFERENCES

[1] A. F. Schwarz, "Computer-Aided Design of Microelectronic Circuits and Systems," Academic Press, Cambridge, Vol. 1, 1987.

[2] J. Vlach and K. Singhal, "Computer Methods for Circuit Analysis and Design," Van Nostrand Reinhold Company, New York, 1983.

[3] M. Hasler, "Stability of Parasitic Dynamics at a dc-Operating Point: Topological Analysis," *Proceedings of the IEEE International Symposium on Circuits and Systems*, Singapore, 1991, pp. 770-773.

[4] N. Tetsuo and C. Leon~O, "Topological Conditions for a Resistive Circuit Containing Negative Non-Linear Resistors to Have a Unique Solution," *International Journal on Circuit Theory and Applications*, Vol. 15, Vol. 2, July 1987, pp. 193-210.

[5] M. Fosseprez, M. Hasler and C. Schnetzler, "On the Number of Solutions of Piecewise-Linear Resistive Circuits," *IEEE Transactions on Circuits and Systems*, Vol. 18, No. 3, March 1989, pp. 393-402.

[6] M. Fossèprez and M. Hasler, "Resistive Circuit Topologies that Admit Several Solutions," *International Journal on Circuit Theory and Applications*, Vol. 18, No. 6, December 1990, pp. 625-638.

J. Software Engineering & Applications, 2010, 3, 525-535
doi:10.4236/jsea.2010.36060 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

New Approach for Hardware/Software Embedded System Conception Based on the Use of Design Patterns

Yassine Manai, Joseph Haggège, Mohamed Benrejeb

LA.R.A, Ecole Nationale d'Ingénieurs de Tunis, Tunis, Tunisie.
Email: yacine.manai@gmail.com, {joseph.haggege, mohamed.benrejeb}@enit.rnu.tn

Received February 5th, 2010; revised April 25th, 2010; accepted April 27th, 2010.

ABSTRACT

This paper deals with a new hardware/software embedded system design methodology based on design pattern approach by development of a new design tool called smartcell. Three main constraints of embedded systems design process are investigated: *the complexity, the partitioning between hardware and software aspects and the reusability. Two intermediate models are carried out in order to solve the complexity problem. The partitioning problem deals with the proposed hardware/software partitioning algorithm based on Ant Colony Optimisation. The reusability problem is resolved by synthesis of intellectual property blocks. Specification and integration of an intelligent controller on heterogeneous platform are considered to illustrate the proposed approach.*

Keywords*: *Embedded Systems, Design Patterns, Smartcell, Hardware/Software Partitioning, Intellectual Property

1. Introduction

There are two main orientations in embedded system research, the technological field and the methodological one [1]. The first is characterized by the increasing revolution in integration, the second tries to develop the embedded system design process by examining new design tools in order to front the complexity of embedded systems. There are three main problems during system design: the complexity, the hardware/software (HW/SW) partitioning and the reusability.

To simplify the design process, designers are recurring to raise the abstraction level, from Register Transfer Level (RTL) to system level. As a consequence, a gap between application development and architecture synthesis appears. In order to solve this problem, many frameworks are developed like transactional environments between application development and architecture synthesis [2,3], or many design tools are developed in order to improve embedded system performances [4,5]. In the domain of control system processor implementation, architecture and design framework for processor, solutions have been developed for linear time invariant (LTI) control and embedded real time control applications [6-8]. In [9], a design methodology based on a transactional model which is inserted between the application and the architecture is presented. In this way, the application is refined in an intermediate level which contains the architecture parameters. From this level, the implementation step is achieved in order to generate the RTL architecture.

Our contribution to resolve the complexity problem consists to develop two intermediate environments in order to minimize the gap between application development and architecture synthesis.

The second problem is the hardware/software partitioning. The HW/SW co-design is evolved in a way to automate all phases of design flow coming from physical phase to design one passing through the HW/SW partitioning and synthesis phases [10]. Our contribution to resolve HW/SW partitioning problem, based on ant colony algorithm development, presented in [11]. The work of [12] considers the hardware/software partitioning problem of the embedded system design of reconfigurable architecture. An automatic hardware/software partitioning methodology is proposed in order to develop the dynamically reconfigurable architecture. First, the system specification is developed with the *SyncChart* formalism based on the *Esterel* language. Next, the proposed partitioning method is applied, and the generated (C, Java) code is implemented on the heterogeneous target. To give a reusable solution of hardware/software partition-

ing, this paper presents a solution based on *Composite* design pattern development.

The third problem is the reusability in design process. Design patterns [13] have been operated in order to develop reusable design tools in different engineering fields. Many researches in this field are performed. The work of [14] developed an object analysis pattern for embedded system, further; a requirement pattern with design pattern approach was developed in [15]. A wrapper design pattern for adapting the behaviour of the soft IPs was proposed in [16]. The reusability of Intellectual Property (IP) blocks have been performed extensively for design hardware applications and IP blocks synthesis [16-18]. The development of IP blocks based on design pattern use *Unified Modelling Language* (UML) as specification language, [19] present design pattern modelling in UML. Many researches are performed for the reusability problem in order to develop new design tools that encapsulate all co-design phases in order to implement intellectual property (IP) blocks. One attempt proposed in [20,21] have as aim to develop the *smartcell* design tools in order to implement HW and SW IP blocks for heterogeneous platforms. This *smartcell* is developed with design pattern approach and oriented-object concept based on UML language. Our contribution to resolve the reusability problem consists in the synthesis of IP blocks for hardware and software solutions from direct acyclic graph (DAC). The proposed approach examines the *Builder* design pattern to produce IP blocks.

The remainder of this paper is organized as follows. Section 2 introduces the proposed hardware/software approach for embedded system design. A case study is discussed in next section which validates the proposed approach by design of induction motor controller system. The conclusion and the future works are presented in the last section of this paper.

2. The Proposed Hardware/Sotware Approach

2.1 Requirements of Proposed Approach

Three main problems are targeted by this paper: the first concerns the complexity mastering of embedded system; our contribution is to raise the abstraction level by investigating an object-oriented approach with the design pattern concept. The second is the reusability of IP blocks in order to minimize the time-to-market. Finally, the hardware-software partitioning is solved with a proposed algorithm, based on ant colony optimisation, in order to optimise task's deadline of a direct acyclic graph that models the embedded system. Further, this paper demonstrates the use of a design pattern concept for all phases in design flow.

The proposed hardware/software embedded system design process is presented in **Figure 1**. The proposed co-design flow operates in two levels, the system level and the smartcell one. First step consists to decompose the embedded system in a set of subsystems. Each subsystem is developed in the smartcell level.

The proposed approach considers the smartcell as a design agent that encapsulates the design process composed by specification, application development, architecture synthesis, the HW/SW partitioning, integration and validation phase. In the system level, we model the embedded system by the "smartcell system level", which have the following actions: the decomposition of the main system into subsystems, HW/SW partitioning process, the integration and the global validation of the main system.

In the second level of abstraction, each subsystem is modelled with a smartcell which have the following steps: the application development, the architecture synthesis, and the hardware/software partitioning.

2.2 Complexity Problem

The first step for smartcell system level is the decomposition of the system into a set of subsystems. We develop the design pattern smartcell Factory in order to do this. The decomposition's automation is guaranteed with this design pattern. Its intent is to allow an interface for creating a family of dependent objects without need to specify their concrete classes. The global system is decomposed into four subsystems, the input, the output, the physical subsystem and the controller one. Each one of these subsystems is managed by a smartcell (e.g., SCell_ Input in **Figure 2**).

We define the following actions: the application development, the architecture synthesis the communication management and the HW/SW partitioning. We define for each action an actor modelled with a class diagram in UML. Each actor has four missions corresponding to its smartcell.

Figure 2 presents the smartcell Factory. To implement each subsystem, the design pattern Factory_Method allows making use of the subsystem structure. It is named also virtual constructor and it defines an interface for creating an object instantiated from subclasses (concrete classes) [13]. The design pattern combined with abstract factory in order to decompose the global system into a class of subsystems.

2.2.1 Application Development

The application development is composed of two phases, the application modelled and the direct acyclic graph (DAG) development. First, the application model is developed with state space approach in order to extract the main block of the disturbances blocks. Second, the DAG is developed with the proposed MAC_Builder environment which builds application's graph. The application

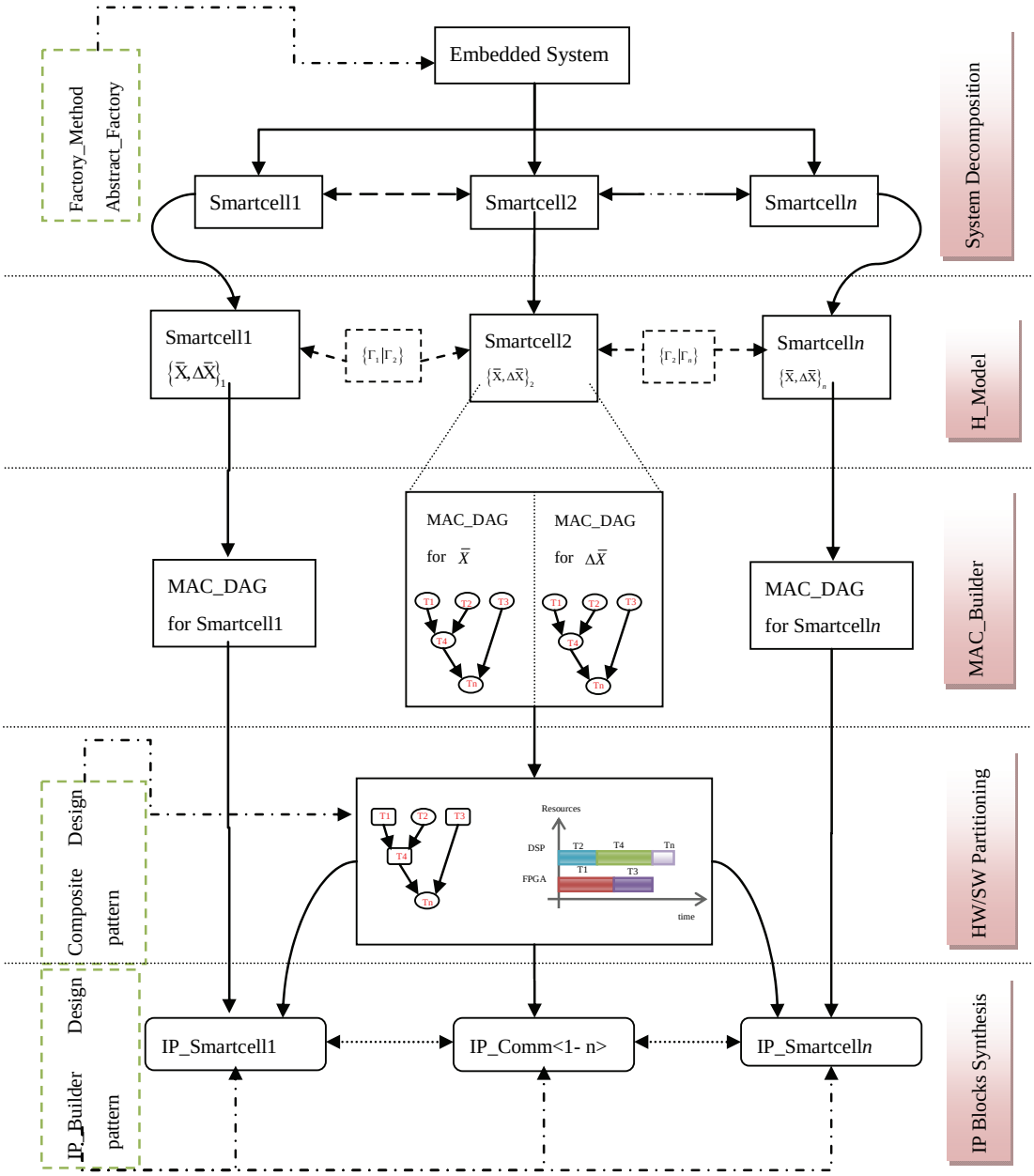


Figure 1. Proposed design process

development consists to following functionalities:

- the analytic model development,
- the MAC_Builder development.

1) The Analytic Model H

In this section, we present the analytic model corresponding to smartcell design pattern. This model allows developing the mathematical representation of subsystems with state space approach in order to characterise the corresponding subsystem.

The proposed analytic model H encapsulates the necessary information in order to carry out the smartcell. This model is a hybrid model that comports heterogeneous elements, presented by the Equation (1).

$$H = \{\bar{X}, \Delta\bar{X}, \Gamma, Y\} \quad (1)$$

where $\bar{X}$ is the nominal system model, $\Delta\bar{X}$ is the distur-bances model, $Y = \bigcup_{i=0}^{n} y_i$: the monitoring system, $\Gamma = \bigcup_{i=0}^{n} f_i$: the communication protocol system.

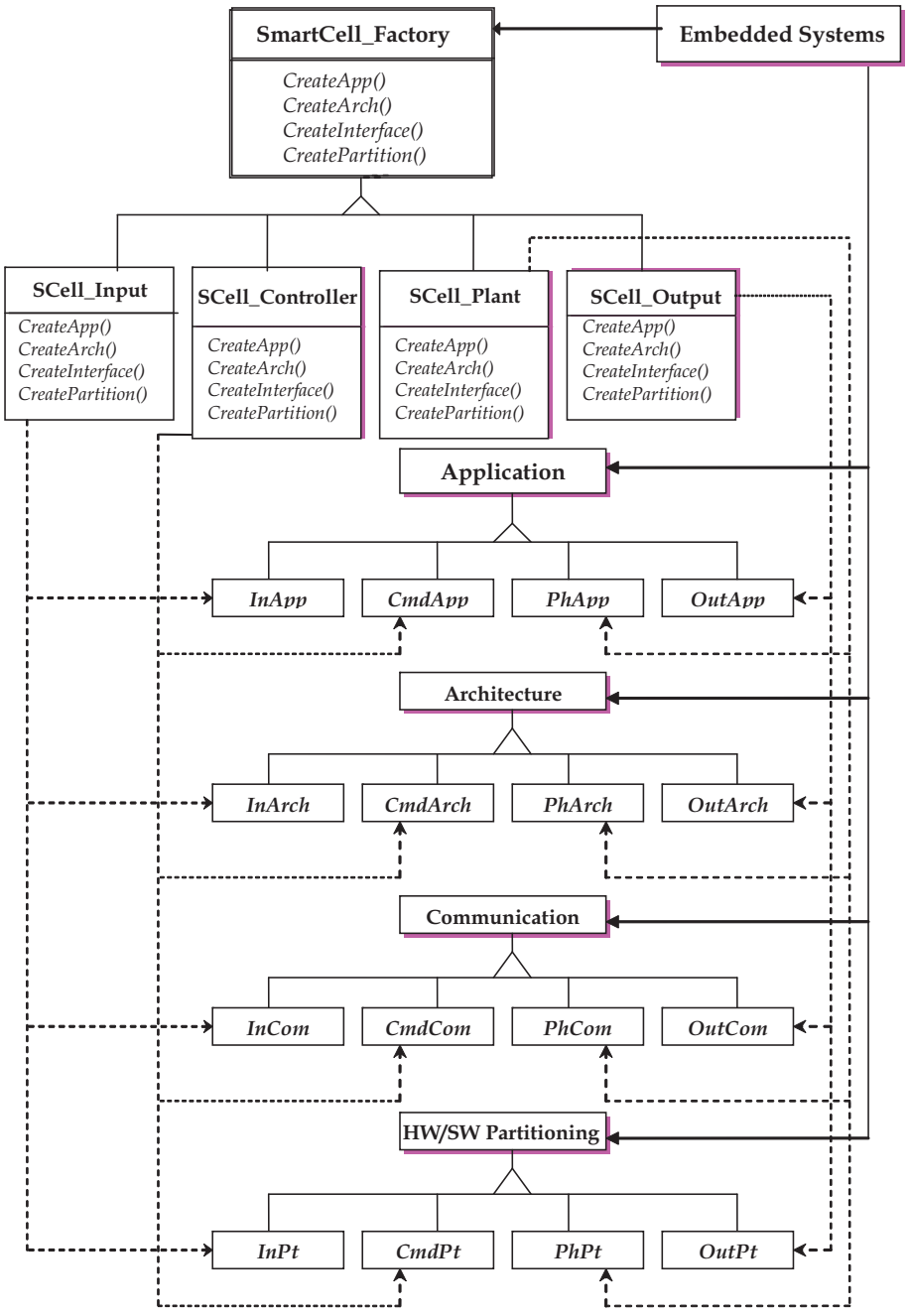


Figure 2. System decomposition with smartcell factory

In this equation, $\overline{X}$ encapsulate the system model described in state space in addition to state vector, input vector and output vector. The $\Delta\overline{X}$ function represents the disturbances applied on the system. This function is represented with sensitivity functions in order to model the disturbances.

The communication protocols are encapsulated with the Γ function, and the fault-handler control laws to be integrated in target architecture are encapsulated with Y function. Three phases must be distinguished for analytical model H; first, we elaborate the model with transfer function of smartcell. Next, we transform each transfer function in the state space using of *compagnon* form. Third phase consists in determination of the smartcell with delta representation.

The Strategy design pattern is developed in order to simplify the control law coding, and the choice of the correspondent control law for application. The control

law chooses is carried out through activation of *ControlLaw*() function of *Model* class. This function activates the *ControlLaw*() function of AbstractLaw class. Considering a linear system defined by Equation (2),

$$\begin{bmatrix} Y(P) \\ U(P) \end{bmatrix} = \begin{bmatrix} FT_{11} & FT_{12} & FT_{13} & FT_{14} \\ FT_{21} & FT_{22} & FT_{23} & FT_{24} \end{bmatrix} \times \begin{bmatrix} R(P) \\ D_i(P) \\ D_o(P) \\ D_m(p) \end{bmatrix} \quad (2)$$

we define an operator that allow extracting the i^{th} column of matrix (FT), we note this operator $X(.)$. To extract the main part of system, *i.e.* the output vector $\mathrm{Y}(P)$ and the control vector $\mathrm{U}(P)$ each one in function of reference vector $R(P)$, we apply the operator $X(.)$ to first column of equation, and we obtain:

$$X(1)(FT) = \begin{bmatrix} FT_{11} & 0 & 0 & 0 \\ FT_{21} & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

Otherwise, the $X(.)$ operator allows extracting the disturbance information, like input or output system disturbance, the $\Delta\overline{X}$ function of H model, is given with $X(.)$ operator as follow,

$$\Delta\overline{X} = \{\bigcup_{i=2}^{4} x(i)(FT)\}.\begin{bmatrix} R(P) \\ D_i(p) \\ D_o(P) \\ D_m(p) \end{bmatrix} \quad (4)$$

where $$\left\{\bigcup_{i=2}^{4} \chi(i)(FT)\right\} = \begin{bmatrix} 0 & \Delta\overline{X}_{12} & \Delta\overline{X}_{13} & \Delta\overline{X}_{14} \\ 0 & \Delta\overline{X}_{22} & \Delta\overline{X}_{23} & \Delta\overline{X}_{24} \end{bmatrix}$$

The second phase of H model is the transformation of transfer functions in state space. The main part of the system is given by Equation (5):

$$\overline{X} = \begin{bmatrix} Y(p) \\ U(p) \end{bmatrix} = \begin{bmatrix} \overline{X}_{11} \\ \overline{X}_{21} \end{bmatrix}\left[R(p)\right] = \frac{\begin{bmatrix} G(p)C(p) \\ C(p) \end{bmatrix}}{1+G(p)C(p)} \quad (5)$$

Each component of $\overline{X}$ vector is described with state space approach, as follow:

$$\overline{X}_{ij} = \begin{bmatrix} X_{ij}[k+1] \\ y_{ij}[k] \end{bmatrix} = \begin{bmatrix} A_{ij} & B_{ij} \\ C_{ij} & D_{ij} \end{bmatrix}\begin{bmatrix} X_{ij}[k] \\ U_{ij}[k] \end{bmatrix}; i = 1,2 \text{ and } j = 1 \quad (6)$$

where, the matrix A, B, C and D are given with a canonical representation like *compagnon* form. To model the disturbances parts of smartcell, we compute each $\Delta\overline{X}_{ij}$ vector as presented in Equation (7):

$$\Delta\overline{X}_{ij} = \begin{bmatrix} X_{ij}[k+1] \\ y_{ij}[k] \end{bmatrix} = \begin{bmatrix} A_{ij} & B_{ij} \\ C_{ij} & D_{ij} \end{bmatrix}\begin{bmatrix} X_{ij}[k] \\ U_{ij}[k] \end{bmatrix}; \quad (7)$$
$$i = 1,2 \text{ and } j = 2,\ldots,4$$

Then, we can develop the discrete model with delta operator, defined by Equation (8):

$$\delta(f(t)) \equiv \delta f[k] = f(k+1) - f(k) \quad (8)$$

Through delta representation of system, we can develop the reccurent equations as describe by Equation (9).

$$y[k] = c_1 x_1[k] + c_1 x_1[k] + \cdots + c_n x_n[k] + c_u u[k] \quad (9)$$

where,

$$\begin{aligned} x_t &= x_1 + x_2 + \cdots + x_n \\ x_1[k+1] &= x_1[k] + a_1 x_2[k] \\ x_2[k+1] &= x_2[k] + a_2 x_3[k] \\ &\vdots \\ x_{n-1}[k+1] &= x_{n-1}[k] + a_{n-1} x_n[k] \\ x_n[k+1] &= x_n[k] - a_n x_t + a_n u[k] \end{aligned}$$

2) The MAC Builder Model

An embedded system is modelled with a set of task graphs. Each task graph is composed by a set of nodes each one representing a task, and a set of edges that links between nodes. Each task can be implemented with software IP or hardware IP. An important property that characterizes the task graph building is the node granularity. There are three categories of granularity: the fine, the gross and the variable granularity. This paper introduces a new approach to building a task graph, that model an embedded system, based on a MAC_Operation granularity.

The MAC operations are composed of arithmetic operations, multiply and accumulate. The MAC builder environment consists in building graphs task from recurrent equations given by H model. Consider a recurrent equation; we can transform this in the list of MAC operations, for example, a fourth order linear system can modelled with MAC operation environment as present **Figure 3**. We use 13 MAC units for this system development. T_0 and T_N are fictive tasks, which indicate the start and end point respectively.

After modelling with task graph, the next step consists to realize the tasks partitioning into hardware and software targets. Indeed, the partitioning phase comports two main stages, space allocation and times scheduling.

Consider the fourth order linear system. The scheduling tasks of this system conduct to result presented in **Figure 4**. In this example, the *time execute* of one MAC operation is taken equal to 3 cycles.

The proposed MAC_Builder environment can be used to determine a task graph corresponding to any other type of system. For example, consider a non linear system given by the following equations:

$$y[0] = \sqrt{\frac{2}{3}} \times \left(u[0] \times \cos(i) + u[1] \times \cos(j) + u[2] \times \cos(k)\right);$$

$$y[1]=\sqrt{\frac{2}{3}}\times\left(-u[0]\times\sin(i)-u[1]\times\sin(j)-u[2]\times\sin(k)\right);$$

This system uses sinusoidal functions. In order to implement these functions, we develop new operations called MAC_cos and MAC_sin.

Consider the "cos" function development for example. First, we determine the approximation of "cos" by a polynomial of degree 12 on [0, π/4],

$\cos x \approx 1-\frac{x^2}{2}+C_1x^4+C_2x^6+C_3x^8+C_4x^{10}+C_5x^{12}$ where, $C_i, i=\{1,\ldots,5\}$ are the given constants. The proposed task graph of "cos" function is given in **Figure 5**.

The register R is initially loaded by the C5 constant. Six iterations are needed to compute the function "cos". The last example demonstrates how we can apply the proposed MAC_Builder for non linear applications.

For a generic aspect of a proposed approach, a "Composite" design pattern is carried out in order to building a task graph corresponding to this subsystem. The next section presents the hardware/software partitioning by use of this design pattern.

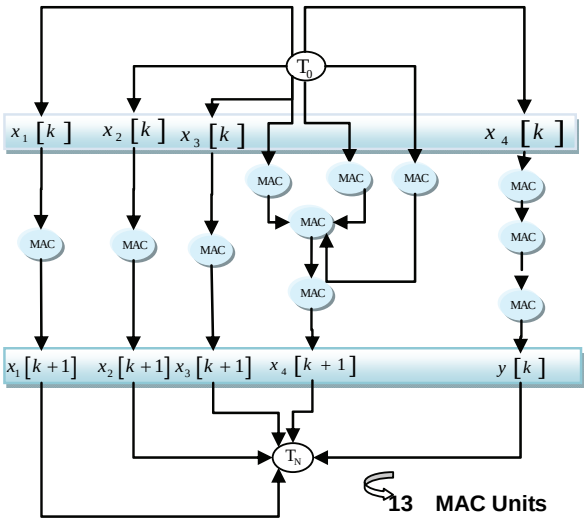


Figure 3. Task graph of four order system

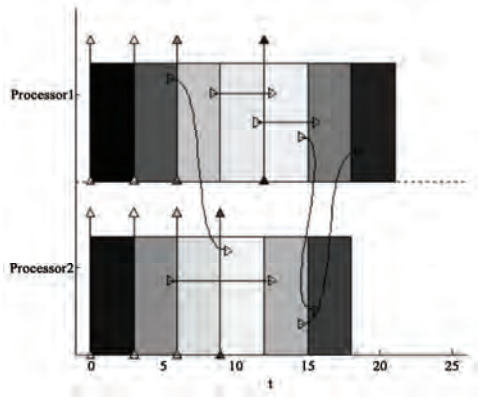


Figure 4. Scheduling in two processors

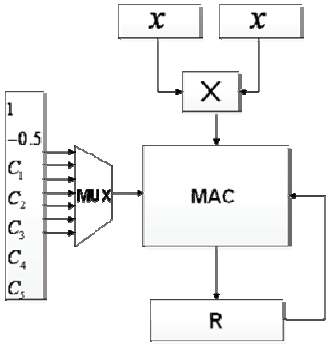


Figure 5. MAC_cos operation

2.3 Hardware/Software Partitioning

The hardware/software partitioning problem consists to respect a deadline of tasks in direct acyclic graph. The optimisation of this factor is function of the parallelism between tasks, and the good management of allocation tasks to hardware and software targets.

The partitioning problem is an NP-complete problem which it hasn't a polynomial resolution algorithm, but we can verify in polynomial time if S is a solution (S is a proposition of resolution).

2.3.1 MAC Operation as an Estimation Unit

An embedded system modelled with a smartcell, can be designed with state space models. This search examines the determination of MAC operation unit as an elementary block to represent a granularity of embedded system. The state vector, for example, can be represented with the MAC operation structure from its recurrent function.

2.3.2 Problem Formulation

Consider an embedded system modelled with a task graph G = {E, V}, E is edges set which rely two nodes and V is a set of nodes. Each node is defined with a start execution date and end of execution date.

An embedded system is a set of smartcells each one is modelled with a state space representation. For each smartcell, state vector is programmed with a recursive functions based on MAC operation.

Each node of task graph has a list of parameters, the time execution in DSP, the time execution in FPGA, and the silicon area. **Figure 6** presents the task graph parameteri-sation. Later on estimation parameters, we apply the proposed algorithm. Each node can be implemented either on DSP board or on FPGA one, then the complexity is equal to 2^n if n is the number of node.

The design pattern Composite is used for hardware/software partitioning problem formulation as a task graph. All successors' tasks are viewed as children tasks in relation to precedent task. The last task is viewed as a leaf by

the Composite design pattern. **Figure 7** present this design pattern.

2.4 IP Blocks Reusability

After the hardware/software partitioning phase, the next step in design process is to synthesise the intellectual property IP blocks. We distinguish two families of IP blocks, the Soft IP and the Hard IP. **Figure 8** presents the

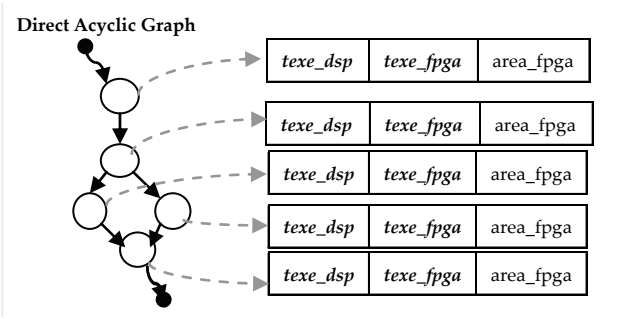


Figure 6. Resources estimation

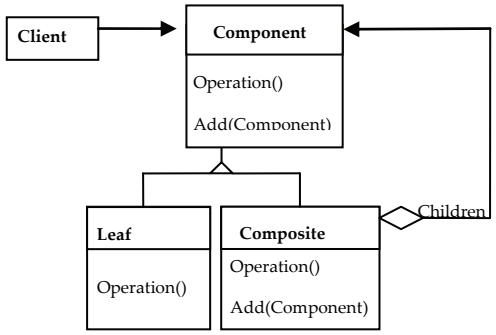


Figure 7. Composite design pattern

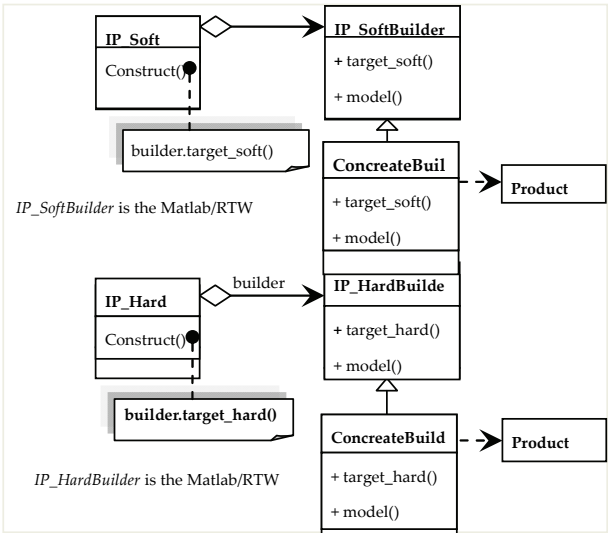


Figure 8. Hardware & software IP blocks

in the development of C/C++ code to be implemented in proposed IP design pattern. The synthesis of soft IP consists software target like DSP. In the other hand, for hard IP, we use the VHDL/Verilog code to be implemented in hardware target like FPGA. Each control law allows generating the C/C++ code in floating or fixed point implementation. This research investigates the development of IP soft and IP hard in order to synthesis the architecture of controller systems implemented in heterogeneous hardware/software target.

2.4.1 The IP Soft Development

The soft IPs blocks reusability in the design process, led us to introduce improvements in the development process of these blocks by investigating the aptitudes of design pattern approach. The IP soft development consists to convert a state space representation into a C/C++ file which can be implemented on software target. The "Builder" design pattern assumes the building of complex object by the specification of its type. The building details are hidden to user. The main motivation to use "Builder" design pattern is to simplify the code generation for building a complex object. The Builder pattern encapsulates the composite objects building, because this action is hard, repetitive and complex.

2.4.2 The IP Hard Development

The hardware synthesis of an application consists in the generation of VHDL/Verilog code to be implemented on target. We investigate two kinds of hardware architecture, FPGA and ASIC circuits. As seen for IP soft development, the IP hard development consist to model a subsystem with the state space approach and coding this model with corresponding hardware language. Each IP hard represent one MAC operation generated with MAC builder environment. We distinguish two kinds of MAC operation implementation, either hardware or software.

3. Case Study

This section presents the design of a control system in such a way that justify how our approach can be applied, in order to implement a hardware/software solution of embedded system by use of IP blocks.

The studied system, given in **Figure 9**, is an induction machine and we intend to implement its speed control system with our proposed approach. Then, we present the development of Hard/Soft IP blocks, and the HW/SW partitioning of this embedded system with *smartcell* design approach.

The induction machine control system is carried out with park transformation technique. Two blocks are developed with S-function, park_dq_abc function, and park_abc_dq fucntion. The variable measurement is carried out with estimator. The dynamic of the study system is presented in **Figure 10**.

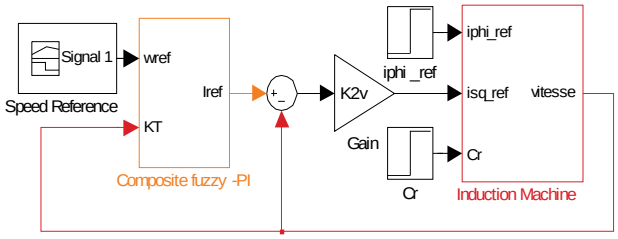


Figure 9. Induction machine control system

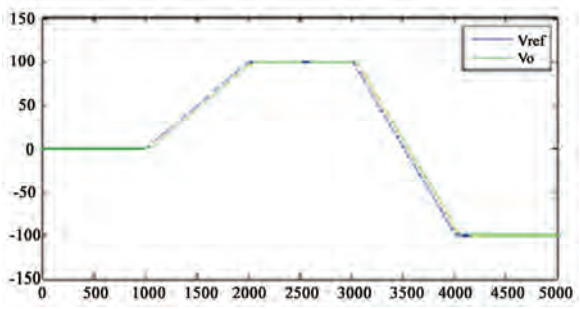


Figure 10. Speed response

3.1 Complexity Problem

We decompose the global system into four subsystems. The first subsystem composed with the input vector that contains the speed reference, the courant reference and the load torque. The control subsystem contains the flow controller, the torque controller and the speed controller. The second subsystem is the induction machine model composed with park transformation modules and induction machine model. The output subsystem contains the output vector, the courant estimator, and the speed estimator.

The control system is modelled as a smartcell in order to apply the proposed approach. First, the system is decomposed into four subsystems presented before with the SmartcellFactory design pattern. The development of task graph that model the embedded system is carried out in two phases, the H_model determination and the MAC_Builder development. From given recurrent equation we develop the task graph correspondent to each subsystem. The synthesis of Hard/Soft IP blocks is developed with VHDL and C/C++ code respectively by the mean of "Builder" design pattern. The HW/SW partitioning is carried out after development of each task graph with "composite" design pattern. The generated C/C++ code is implemented in DSP TMS320F2812 target, whereas the VHDL/Verilog code is integrated in FPGA Spartan 3 target.

The system decomposition is made with a developed abstract factory design pattern, the SmartCellFactory. This design pattern assumes the system decomposition and gives four main missions to each subsystem, the application development, the architecture synthesis, the hardware/software partitioning and the communication management.

Given that in oriented object concept the object creation is based on constructor function, the smartcell tool uses the Factory Method design pattern so that this function supports the heritage management by the mean of virtual property.

Indeed, the smartcell investigates the couple {*Abstract_Factory, Factory_Method*} design patterns in order to assume the decomposition process with oriented object approach. **Listing 1** presents the proposed SmartCellFactory that decomposes the initial system specification into a set of subsystems and presents the control subsystem development.

3.1.1 Analytic Model H of System

Consider the speed control system of induction drive. To extract the system from $\bar{X}$ vector, we apply the $\chi(.)$ operator

$$\bar{X} = \begin{bmatrix} Y(p) \\ U(p) \end{bmatrix} = \begin{bmatrix} \bar{X}_{11} \\ \bar{X}_{21} \end{bmatrix} \left[R(p) \right] \tag{10}$$

The process is modelled with transfer function $G(p) = e^{-\tau p}\bar{G}(p)$, $G(p)$ is the system model

```
/* System decomposition with  SmartCellFactory */

Class SmartCellFactory
{
Public :
    virtual Application CreateApp() const
    {return new Application ;}
    virtual Architecture CreateArch() const
    {return new Architecture;}
     virtual Communication CreateInterface() const
    {return new Communication ;}
    virtual Partitionnement CreatePartition() const
    {return new Partitionnement ;}
} ;
SmartCellControl*
SmartCell.Design ::CreateApplication(SmartCell.Factory &
factory)
{
   SmartCellControl *
UnifiedStructure=factory.CreateU.S() ;
```

Listing 1. Embedded system decomposition

$$\left[\bar{X}_{11}\right] = e^{-\tau p}\frac{\bar{G}(p)C(p)}{1+\bar{G}(p)C(p)} \quad (11)$$

The time delay represent the duration between control signal sending and its reception by the physical system. Its expression is approximated with first order Taylor series, then,

$$\left[\bar{X}_{11}\right] = \frac{1-\tau p/2}{1+\tau p/2}\times\frac{\bar{G}(p)C(p)}{1+\bar{G}(p)C(p)} \quad (12)$$

This fractional equation of $\bar{X}_{11}$ has the form,

$$\left[\bar{X}_{11}\right] = \frac{B(p)}{A(p)} = \frac{b_n p^n + b_{n-1}p^{n-1} + \cdots + b_1 p + b_0}{a_n p^n + a_{n-1}p^{n-1} + \cdots + a_1 p + a_0} \quad (13)$$

where $A(p)$ and $B(p)$ are polynomials that have n as maximum order. The next step consists to transform this equation with delta operator in order to discrete it. From this model, we can develop the corresponding recurrent equations.

3.1.2 MAC_Builder Environment

From recurrent equations given by the H model, the proposed MAC_Builder environment allows task graph development with MAC granularity. We propose to develop a task graph for the Park transformation function presented in **Listing 2**.

This function use trigonometric functions as sine function. We have developed a MAC structure corresponding to sinusoidal functions called MAC_sin and MAC_cos in order to develop a task graph corresponding to nonlinear elements. The **Figure 5** illustrates the proposed structure of MAC_cos. The task graph corresponding to Park Transformation function is given by **Figure 11**.

The process starts by computing the trigonometric functions of input vector by use of MAC_cos and MAC_sin functions; next we compute the output vector through investigation of elementary MAC operations.

3.2 Hardware/Software Partitioning

After the task graph development with MAC_Builder environment, the next step of proposed approach is the partitioning of these tasks between hardware and software platforms. With the aim to apply this approach on the induction motor control system, we can implement the developed task graph with the Composite design pattern. The advantage of this approach is to give an object which we can be reusable for several embedded system application just by modifying some parameters. In fact, each task is modelled with Composite design pattern given by **Figure 7**.

In order to affect tasks to hardware/software targets, we have developed a partitioning algorithm based on ant colony optimisation. We use the following notation, the visibility between two nodes of task graph and the pheromone's constants given by matrixes (h_{nn}) and (τ_{nn}) respectively:

$$(\eta_{nn}) = \begin{bmatrix} \eta_{11} & \eta_{12} & \cdots & \eta_{1n} \\ \eta_{21} & \eta_{22} & \cdots & \eta_{2n} \\ \vdots & \vdots & & \vdots \\ \eta_{n1} & \eta_{n2} & \cdots & \eta_{nn} \end{bmatrix} \qquad (\tau_{nn}) = \begin{bmatrix} \tau_{11} & \tau_{12} & \cdots & \tau_{1n} \\ \tau_{21} & \tau_{22} & \cdots & \tau_{2n} \\ \vdots & \vdots & & \vdots \\ \tau_{n1} & \tau_{n2} & \cdots & \tau_{nn} \end{bmatrix}$$

The transition rule is computed by a probability given by Equation (14), where α *and* β are parameters that control the visibility and pheromone respectively.

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}{\sum_{l\notin S_k}\left[\tau_{il}(t)\right]^{\alpha}\cdot\left[\eta_{il}\right]^{\beta}} \quad (14)$$

The pheromone matrix is updated by the function, $\tau_{ij}(t) = \rho\times\tau_{ij}(t) + (1-\rho)\times\Delta\tau_{ij}$ where $0 \prec \rho \prec 1$ and $\Delta\tau_{ij}(t) = \begin{cases} \dfrac{Q}{L(t)} \;\; si \;\; (i,j)\in T\;(t) \\ 0 \;\; si \;\; (i,j)\notin T\;(t) \end{cases}$; Q constant.

3.3 Complexity Problem

This section presents the synthesis of a hardware IP

```
void park_abc_dq_Outputs_wrapper
(const real_T *u, real_T *y)
{
const double pi=3.1416;
double i, j, k;
i=0; j=0; k=0;
   i=u[3];
   j=u[3]- 2*pi/3;
```

Listing 2. Park transformation function

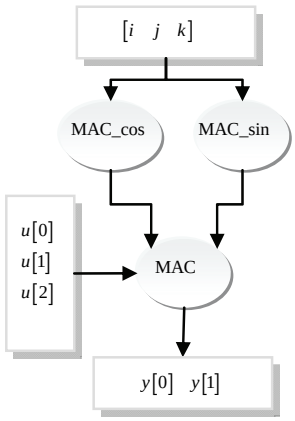


Figure 11. Park transformation DAG_MAC

block that implements the fuzzy controller in FPGA target. **Figure 12** illustrates three signals, the *error*, the *delta_error* and the *control* signal.

The application of partitioning algorithm contributes to affect the tasks into hardware aspect or software one. **Figure 13** present an obtained result for this phase. Further, a part of the logic circuit of this IP hard is presented in **Figure 14**.

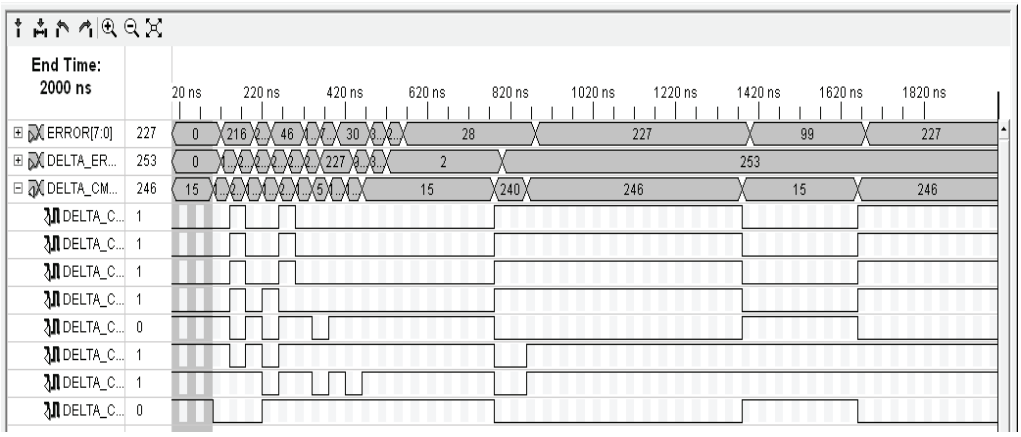


Figure 12. Simulation of fuzzy control system

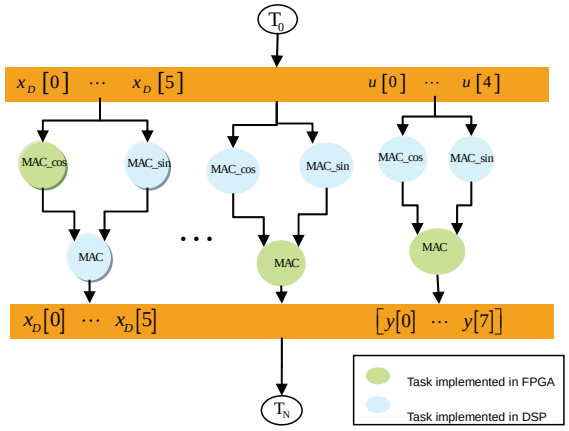


Figure 13. HW/SW partitioning

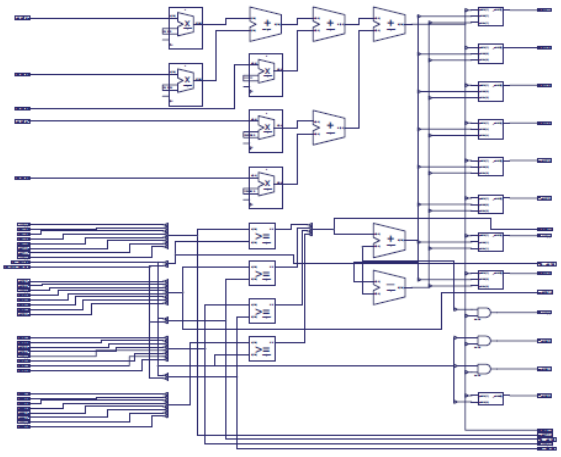


Figure 14. Logic circuit of fuzzy control system

4. Conclusions

In this work we carried out a multilevel design flow for embedded system through investigating the design pattern concept. In system level, the system decomposition is realised with the *Smartcell_Factory* design pattern. In the second level, each smartcell realises the model development, the DAG development, the hardware/software partitioning and the IP_hard/IP_soft blocks synthesis.

Three problems are resolved: the complexity, the hardware/software partitioning, and the reusability. Indeed, two intermediate models are developed in order to model the subsystem and to develop the task graph, the *H_model* that encapsulates the principal and the disturbances information of system in addition to communication protocol and control laws. The *MAC_Builder* is the second environment that allows developing a task graph corresponding to subsystem from the recurrent equation given by the *H_model*. The hardware/software partitioning problem deals with proposed *Component* design pattern which model the task graph given by the *MAC_Builder* in order to simplify the application of the proposed Ant Colony Optimisation algorithm. The IP blocks reusability is carried out through the result given by the partitioning phase, the IP_soft blocks are developed with C/C++ language and the IP_hard blocks are developed with VHDL/Verilog language.

As future work, the development of paradigm environment to execute the proposed approach is very required. In addition, we propose the development of complex control laws as fuzzy or neuronal control by the mean of the proposed *MAC_Builder* environment.

REFERENCES

[1] R. C. Dorf, "Systems, Controls, Embedded Systems, Energy, and Machines," Taylor & Francis, New York, 2006, pp. 486-511.

[2] K. Virk and J. Madsen, "A System-Level Multiprocessor System-on-Chip Modeling Framework," *Proceedings of SOC*, 2004.

[3] A. D. Pimentel and C. Erbas, "A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels," *IEEE Transactions on Computer*, Vol. 55, No. 2, February 2006, pp. 99-112.

[4] B. Zhou, W. Qiu and C. Peng, "An Operaing System Framework for Reconfigurable Systems," *Proceedings of CIT*, Salt Lake, 2005.

[5] S. Pasricha, N. Dutt and M. B. Romdhane, "Using TLM for Exploring Bus-Based SoC Communication Architectures," *Proceedings of ASAP*, Atlantic, 2005.

[6] R. Cumplido, S. Jones, R. M. Goodall and S. Bateman, "A High Performance Processor for Embedded Real-Time Control," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 3, May 2005, pp. 485-492.

[7] X. Wu, V. A. Chouliaras, J. L. Nunez and R. M. Goodall, "A Novel DS Control System Processor and its VLSI Implem-Entation," *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, Vol. 16, No. 3, March 2008, pp. 217-228.

[8] D. L. Sancho-Pradel and R. M. Goodall, "Targeted Processing for Real-Time Embedded Mechatronic Systems," *Control Engineering Practice*, Vol. 15, 2007, pp. 363-375.

[9] Y. Atat and N. E. Zergainoh, "Automatic Code Generation for MPSoC Platform Starting From Simulink/Matlab: New Approach to Bridge the Gap between Algorithm and Architecture Design," *Conference of ICTTA*, Bali Island, 2008.

[10] G. Wang, W. Gong and R. Kastner, "Application Partitioning on Programmable Platforms Using the Ant Colony Optimization," *Journal of Embedded Computing*, Vol. 2, No. 1, 2005, pp. 1-18.

[11] Y. Manai, J. Haggège and M. Benrejeb, "HW/SW Partitioning in Embedded System Conception Using Design Pattern Approach," *Conference of JTEA*, Hammamet, 2008.

[12] K. B. Chehida, "Méthodologie de Partitionnement Logiciel/Matériel Pour Plateformes Reconfigurables Dynami-Quement," PhD Thesis, Université de Nice-Sophia Antipolis, France, 2004.

[13] E. Gamma, *et al*., "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, Massachusetts, 1995.

[14] S. Konrad, H. C. Cheng and L. A. Campbell, "Object Analysis Patterns for Embedded Systems," *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, December 2004, pp. 970-990.

[15] S. Konrad and B. Cheng, "Requirement Pattern for Embedded System," *Proceedings of the IEEE Joint International Conference on Requirements Engineering*, Atlanta, 2002.

[16] R. Damasevicius, G. Majauskas and V. Stulikys, "Application of Design Patterns for Hardware Design," *Proceedings of DAC*, Anaheim, 2-6 June 2003, pp. 48-53.

[17] F. Rincon, F. Moya and J. Barba, "Model Reuse through Hardware Design Patterns," *Proceedings of Design, Automation, and Test in Europe*, 2005.

[18] P. Coussy, *et al*., "Constrained Algorithmic IP Design for System-on-Chip," *Integration, the VLSI Journal*, Vol. 40, No. 2, 2007, pp. 94-105.

[19] J. K. Mak, C. S. Choy and D. P. Lun, "Precise Modeling of Design Patterns in UML," *Proceedings of International Conference on Software Engineering*, 2004.

[20] Y. Manai, J. Haggège and M. Benrejeb, "PI-Fuzzy Controller Conception with Design Pattern Based Approach," 14*th IEEE International Conference on Electronics, Circuits and Systems*, Marrakech, 2007, pp. 483-489.

[21] Y. Manai, "Contribution à la conception et la synthèse d'architecture de systèmes embarqués utilisant des plates-formes hétérogènes," Ph.D. Dissertation, Ecole Nationale d'Ingénieurs de Tunis, Tunisia, 2009.

J. Software Engineering & Applications, 2010, 3, 536-540
doi:10.4236/jsea.2010.36061 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Testability Models for Object-Oriented Frameworks

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

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

Received April 10th, 2010; revised April 21st, 2010; accepted April 23rd, 2010.

ABSTRACT

Frameworks are time-tested highly reusable architectural skeleton structures. They are designed 'abstract' and 'incomplete' and are designed with predefined points of variability, known as hot spots, to be customized later at the time of framework reuse. Frameworks are reusable entities thus demand stricter and rigorous testing in comparison to one- time use application. The overall cost of framework development may be reduced by designing frameworks with high testability. This paper aims at discussing a few metric models for testability analysis of object-oriented frameworks in an attempt to having quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down.

Keywords*: *Object-Oriented Frameworks, Complexity, Framelet-Based Design and Testability

1. Introduction

Frameworks represent semi-codes for defining and implementing time-tested highly reusable architectural skeleton design experiences and hence become very useful in development of software applications and systems. The object-oriented paradigm provides a promising set of solutions to attain reusability with the help of objects, classes, templates, inheritance, overloading and genericity [1,2] so object-oriented frameworks are much more com- mon than non object-oriented frameworks and have become a synonym for software frameworks. It should be noted that a framework without object-orientation is also possible. As per Gamma *et al*. [3], famous in reuse literature as *gang of four*, an object-oriented framework is a set of cooperating classes that make up a reusable design for a specific class of software which provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations. The five elements of an object-oriented framework, as identified by Gurp and Bosch [4], are design documents, interfaces, abstract classes, components and classes. An abstract class usually has at least one unimplemented operation deferred to for its implementation during framework reuse.

Applications are built from frameworks by extending or customizing the framework, while retaining the original design. The framework-centered development lifecycle broadly consists of following three phases [5]: 1) the **framework development phase** aims at producing reusable design in a domain, consisting of domain analysis, architectural design, framework design, framework implementation, framework testing activities; 2) the **framework usage phase** is also referred to as the framework instantiation phase or application development phase. Once framework is developed, it is deployed for the development of framework-based applications that include the core framework designs or part of it and new classes need to be developed to fulfill the actual applications requirements and 3) the **framework evolution and maintenance phase**.

One has to be very careful about developing *fault free reusable frameworks* because if the framework contains defects, the defects will be passed on to the applications developed from the framework during framework usage phase [6]. The reusable framework thus demands stricter and rigorous testing in comparison to a one-time use application [7,8]. It would be advisable to guaranty the production of high quality frameworks without incurring heavy costs for rigorous testing. This calls for analyzing testability of reusable artifacts so as to reduce the overall cost of framework-based development [9].

Bache and Mullerburg [10] define testability as the minimum number of test cases required to provide total

test coverage, assuming that such coverage is possible. Several definitions of software testability are available in literature but intuitively, a software component that is testable has the following desirable properties [11]:

- test sets are small and easily generated,
- test sets are non-redundant,
- test outputs are easily interpreted and
- software faults are easily locatable.

In spite of wide importance and promotion of frameworks, over the last decades, a widely accepted set of measures to quantify its characteristics has not been established. Moreover, there is a complete lack of framework testability metrics related studies in literature that could produce quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. A recent work [9] proposed models for testability analysis of framework that particularly consider that the frameworks are inherently abstract and variable in nature. This paper proposes few more models for testability analysis of object-oriented frameworks, considering few design related aspects of frameworks.

Some obvious reasons for rigorous testability analysis of a framework could be summarized as [9]:

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

2) Frameworks are reusable entities and hence high testability is essential. As testable system is known to provide increased reliability.

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

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

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

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

The paper is organized in three sections. Proposed testability models for software frameworks appear in Section 2 and Section 3 presents conclusions.

2. Models for Testability Analysis of Object-Oriented Frameworks

This section aims at discussing various metric models for testability analysis of object-oriented frameworks considering few design related aspects of frameworks. Following discussion takes into account the factors that affect the testability of an object-oriented framework, as identified by Ranjan and Tripathi in [12]. They identified various factors and sub factors that affect the testability of frameworks so as to take care of those factors to bring high testability in frameworks. As per their observations, the factors that affect the testability of a framework are related to the characteristics of documentation of a framework, domain of a framework, design of a framework and the test support available for the framework testing like test tools, environments, reusable test artifacts and built-in tests etc.

2.1 A Testability Model Considering the Structural Complexity of a Framework

It is empirically proved that complexity metrics are good predictors of testing effort [13]. An interesting model of OO system complexity, proposed by Tegarden and Sheetz [14], consists of the *system complexity*, *structural complexity*, and *perceptual complexity constructs*. System complexity represents aspects of OO techniques and characteristics inherent to the problem. This construct influences the structural aspects of the system and the perceptual complexity of the OO system. Structural complexity deals with the measurable characteristics of the resulting OO system such as the number of classes and the interconnections between the classes whereas perceptual complexity deals with the ability of the developer to understand the problem, the structural components of the OO system, the use of OO techniques, and how to integrate these ideas to create an OO system.

As per this model, structural complexity identifies four levels of describing complexity of OO systems: variable, method, object, and system. At each level, measures are identified that account for the cohesion (Intra) and coupling (Inter) aspects of the system. Thus, the structural complexity of object-oriented framework may be expressed as:

$$SC_{Fr} \propto \sum_{i=1}^{N} CO_{Intrai} + \sum_{i=1}^{N} CO_{Interi} \quad (1)$$

where,

$$CO_{Intra} \propto CV_{Intra} + CM_{Intra} \quad (2)$$

and

$$CO_{Inter} \propto CV_{Inter} + CM_{Inter} \quad (3)$$

where,

N = Total number of objects in the framework
SC_{Fr} = Structural Complexity of the framework
CO_{Intra} = Intra Object Complexity in the framework
CO_{Inter} = Inter Object Complexity in the framework
CV_{Intra} = Intra Variable Complexity in an object
CV_{Inter} = Inter Variable Complexity in an object
CM_{Intra} = Intra Method Complexity in an object
CM_{Inter} = Inter Method Complexity in an object

It is very clear that framework testability is inversely proportional to its structural complexity. Therefore, using Equation (1) we can write,

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{N} CO_{Intrai} + \sum_{i=1}^{N} CO_{Interi}} \quad (4)$$

It for sure that $\sum_{i=1}^{N} CO_{Intra\,i}$ and $\sum_{i=1}^{N} CO_{Interi}$, in the above model, will never be zero together. Because total number of objects in the framework will always be >= 1. In case when N = 1, the value of $\sum_{i=1}^{N} CO_{Interi}$ may become zero but the value of $\sum_{i=1}^{N} CO_{Intra\,i}$ will not be zero then also.

2.2 A Testability Model Considering Complexity of Framework Interfaces

A framework may have its interfaces linked to external entities like other *frameworks*, *components*, or *library functions* etc. [15], known as external interfaces. More the number of other entities to be integrated with the framework, the more effort are required for their integration testing. This effort increases with the number and complexities of the constituent interfaces.

A framework testability model, considering complexity of framework interfaces, may be expressed as

$$Tb_{Fr} \propto \frac{1}{\left(1 + \sum_{i=1}^{N_{FItf}} C_{FItf\,i} + \sum_{i=1}^{N_{CInf}} C_{CInf\,i} + \sum_{i=1}^{N_{LInf}} C_{LInf\,i}\right)} \quad (5)$$

where,

C_{FItfi} = Complexity of framework's ith interface with other framework
C_{CInfi} = Complexity of framework's ith interface with component
C_{LInfi} = Complexity of framework's ith library interface
N_{FItf} = Total number of interfaces with other frameworks
N_{CInf} = Total number of interfaces with component
N_{LInf} = Total number of library interfaces

2.3 A Testability Model Considering Heaviness of Framework in Terms of its Size

Size happens to be an obvious influencing factor for testing effort [16]. A framework of huge size, *i.e.* consisting of large number of classes, methods, attributes and depth of inheritance etc., is considered heavy. We here make use of (a subset of) object-oriented metrics proposed by Chidamber and Kemerer [17].

The number and complexities of the methods involved in the framework is a predictor of how much time and effort is required to test. We define *WMFr* (the consolidated weighted method per framework in line with *WMC* proposed in [17]) as follows:

$$WMFr = \sum_{i=1}^{Nc} WMC_i \quad (6)$$

where

$$WMC_i = \sum_{j=1}^{NMi} c_{ij} \quad (7)$$

and

$WMFr$ = Sum of weighted methods of all classes in FUT
WMC_i = Weighted method per class of ith class in FUT
Nc = Total number of classes in FUT
C_{ij} = Complexity of jth method in ith class of FUT
N_{Mi} = Total number of methods in ith class

Framework testability model considering heaviness of framework in terms of its size may be defined as follows:

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{Nc}(\sum_{j=1}^{NMi} c_{ij})} \quad (8)$$

because,

$$TE_{Fr} \propto WMFr \quad (9)$$

Nc and N_M appearing in Equation 8 may further be expanded as below in Equations 10 and 11

$$N_C = NOCC_{Fr} + NOAC_{Fr} \quad (10)$$

where,
Nc = Total number of classes in FUT
$NOCC_{Fr}$ = Number of concrete classes in the FUT
$NOAC_{Fr}$ = Number of abstract classes in the FUT

And, methods in a class shall comprise of all the *concrete* methods, *abstract* methods, and *overridden* methods. Thus,

$$N_M = N_{CM} + N_{AM} + N_{OM} \quad (11)$$

where
N_M = Total number of methods in FUT
N_{CM} = Total number of concrete methods in FUT
N_{AM} = Total number of abstract methods in FUT
N_{OM} = Total number of overridden methods in FUT

2.4 A Testability Model Considering Framelet-Based Design of Frameworks

Framelets are the small, flexible, relatively independent and reusable assets, which are designed based on the concept of frameworks. They are mini-frameworks that usually contain less than ten classes and have a simple, clearly defined interface [18]. They evolved as a means of modularizing the frameworks where a family of interrelated framelets is an alternative to complex frameworks. It is basically *a design principle* that instead of designing one full fledged and complex framework, design it with a family of related framelets with lean interfaces [19]. Designing a framework in framelet-based fashion promotes reducing overall complexity of the framework and is like using divide and conquers approach to facilitate both, the design and testing of the framework.

Understanding a framelet-based framework is easier than a full fledged and complex framework because a framelet-based framework is made up of loosely coupled small frameworks, known as *framelets*. Testability of a framelet-based framework depends upon the testability of constituent framelets and the coupling among them. So, we may write,

$$Tb_{Fr} \propto \sum_{i=1}^{N_{Fmlt}} \left(\frac{1}{COUP_{Fmlti}} \times Tb_{Fmlti} \right) \quad (12)$$

where

Tb_{Fr} = Testability of the framework

N_{Fmlt}= Total number of constituent framelets

$COUP_{Fmlti}$ = Sum of measure of coupling of ith framelet with other framelets

Tb_{Fmlti} = Testability of ith framelet

Since, a framelet is nothing but a small framework, consisting of not more than ten classes and very lean interfaces with other framelets [19], so any discussion or metric model regarding framework's testability will be applicable for estimating testability of a framelet.

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

Table 1. Applicability of framework testability metric models

S.No.	Category	Framework Testability Metric Model	Applicability of the Model
1.	Testability models considering various kinds of complexities of frameworks	Testability Model Considering the Structural Complexity of a framework	When framework ***structural complexity*** is of concern for testability.
2.	-- do--	Testability Model Considering Complexity of Framework Interfaces	When ***framework integration*** with other frameworks, components and library functions is of concern for testability.
3.	Testability models considering size and framelet-based design of the framework	Testability Model Considering Heaviness of Framework in terms of its Size	When framework ***size*** is of concern for testability.
4.	-- do--	Testability Model Considering Framelet-based design of frameworks	When framework has ***framelet-based*** design and the testability of framelets are of concern.

3. Conclusions

It is quite obvious that the quality of the software system which has been developed with reuse depends heavily upon the quality of the underlying reusable artifacts. Frameworks are an important reusable artifact and are believed to be at the core of leading-edge software technology in the twenty first century [20]. Software frameworks and design patterns make reuse of design experiences possible but unlike design patterns (that may not have any code) software frameworks are semi-codes and hence call for thorough testing before they can be deployed as reusable entities. Although the technology for constructing frameworks and framework-based software is relatively advanced, we comparatively lack a sufficient theoretical basis for testing them. This paper attempted to discuss a few metric models for testability analysis of object-oriented frameworks in an attempt to having quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. The framework testability models presented here takes into account few design related aspects of object-oriented frameworks.

REFERENCES

[1] J. W. Hooper and R. O. Chester, "Software Reuse: Guidelines and Methods," Perseus Publishing, Cambridge, 1991.

[2] M. Smolarova and P. Navrat, "Software Reuse: Principles, Patterns, Prospects," *Journal of Computing and Information Technology*, Vol. 5, No. 1, 1997, pp. 33-49.

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

[4] J. Gurp and J. Bosch, "Design, Implementation and Evolution of Object-Oriented Frameworks: Concepts and Guidelines," *Software - Practice and Experience*, Vol. 31, No. 3, 2001, pp. 277-300.

[5] J. Bosch, P. Molin, M. Mattsson and P. Bengtsson, "Object-Oriented Framework-Problems and Experiences,"

Research Report, University of Karlskrona/Ronneby, Sweden, 1997.

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

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

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

[9] D. Ranjan and A. K. Tripathi, "Variability-Based Models for Testability Analysis of Frameworks," *Journal of Software Engineering and Applications*, Vol. 3, No. 6, 2010, pp. 455-459.

[10] R. Bache and M. Mullerburg, "Measures of Testability as a Basis for Quality Assurance," *Software Engineering Journal*, Vol. 5, No. 2, 1990, pp. 86-92.

[11] R. S. Freedman, "Testability of Software Components," *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, 1991, pp. 553-564.

[12] D. Ranjan and A. K. Tripathi, "Testability Analysis of Object-Oriented Frameworks," *The Journal of Defense Software Engineering*, accepted for publication.

[13] H. M. Olague, L. H. Etzkorn, S. L. Messimer and H. S. Delugach, "An Empirical Validation of Object-Oriented Class Complexity Metrics and their Ability to Predict Error-Prone Classes in Highly Iterative, or Agile Software: A Case Study," *Journal of Software Maintenance and Evolution*: *Research and Practice*, Vol. 20, No. 3, 2008, pp.171-197.

[14] D. P. Tegarden and S. D. Sheetz, "Object-Oriented System Complexity: An Integrated Model of Structure and Perceptions," Presented at *OOPSLA* 1992. http://www.acis.pamplin.vt.edu/faculty/tegarden/wrk-pap/OOPSLA92.pdf

[15] M. Mattsson and J. Bosch, "Framework Composition: Problems, Causes and Solutions," *Proceedings of Technology of Object-Oriented Languages and Systems*, Netherlands, 1997, pp. 203-214.

[16] P. Jalote, "An Integrated Approach to Software Engineering," Narosa Publishing House, Darya Ganj, 2009.

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

[18] W. Pree, "Design Patterns for Object-Oriented Software Development," Addison-Wesley Publishing Company, Massachusetts, 1995.

[19] W. Pree and K. Koskimies, "Framelets—Small and Loosely Coupled Frameworks," *ACM Computing Surveys*, Vol. 32, No. 6, 2000.

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

J. Software Engineering & Applications, 2010, 3, 541-547
doi:10.4236/jsea.2010.36062 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

User Session-Based Test Case Generation and Optimization Using Genetic Algorithm*

Zhongsheng Qian

School of Information Technology, Jiangxi University of Finance & Economics, Nanchang, China.
Email: changesme@163.com

Received March 22nd, 2010; revised April 12th, 2010; accepted April 13th, 2010.

ABSTRACT

An approach to generating and optimizing test cases is proposed for Web application testing based on user sessions using genetic algorithm. A large volume of meaningful user sessions are obtained after purging their irrelevant information by analyzing user logs on the Web server. Most of the redundant user sessions are also removed by the reduction process. For test reuse and test concurrency, it divides the user sessions obtained into different groups, each of which is called a test suite, and then prioritizes the test suites and the test cases of each test suite. So, the initial test suites and test cases, and their initial executing sequences are achieved. However, the test scheme generated by the elementary prioritization is not much approximate to the best one. Therefore, genetic algorithm is employed to optimize the results of grouping and prioritization. Meanwhile, an approach to generating new test cases is presented using crossover. The new test cases can detect faults caused by the use of possible conflicting data shared by different users.

Keywords*: *User Session, Genetic Algorithm, Test Case, Test Suite, Reduction, Prioritization

1. Introduction

Incapable Web applications can have far-ranging consequences on businesses, economies, scientific progress, health, and so on. Therefore, all the entities of a Web application, in essence, must be tested adequately to ensure that the application meets its original design specifications.

In Web application testing, some test methods and techniques were presented [1-4]. Kung, *et al*. [2] depicted an object-oriented Web Test Model to support Web application testing. Hieatt, *et al*. [1] introduced a method of acceptance testing, and developed a testing tool to show the system operations and the expected output results in XML. The approaches proposed by Kung and Hieatt, *et al*., however, focus primarily on unit testing without concerning the whole testing for Web applications. Liu, *et al*. [4] presented a data flow-based approach to testing Web applications. Most of these methods of testing Web applications are achieved through extending the testing methods for traditional software. Additionally, none of these methods [1-4] yields test data according to user sessions.

Elbaum, *et al*. [5] demonstrated the fault detection capabilities and cost-effectiveness of user session-based testing. Increment concept analysis [6] is used to analyze user sessions dynamically and minimize continuously the number of user sessions maintained. Khor, *et al*. [7] combined the genetic algorithm with formal concept analysis to trace the relationship between test data and corresponding test run.

Sthamer [8] analyzed deeply the test case optimization efficiency of different coding schemes and fitness functions of genetic algorithm for different structure-based software in his doctoral dissertation. Some researchers also studied on test case generation techniques using genetic algorithm [9-10]. However, they aimed for simplex optimization focusing on one-off optimization computation, while the testing is continuous and iterative. So, dynamically continuous optimization computation is more propitious to improve test performance. Jia, *et al*. [11] discussed the key problems of producing test data of covering designated paths using genetic algorithm, and introduced deeply the factors of influencing the genetic algorithm's efficiency through experimental results. Berndt and Watkins [12] summarized the advancement in generating data for generic algorithm-based testing recently. These studies [5,7-12] used genetic algorithm to

*This research has been partly supported by the National High-Tech Research and Development Plan of China under Grant No. 2007 AA01Z144; the Science and Technology Plan Project of the Education Department of Jiangxi Province of China under Grant Nos. GJJ10120 and GJJ10117; Shanghai Leading Academic Discipline Project of China with Project No. J50103; and the School Foundation of Jiangxi University of Finance & Economics of China under Grant No. 04722015.

analyze test problems with the exception of [5], but not focusing on Web application testing. In [5], the authors discussed user session-based Web application testing, but not concerning deeply the optimization of test cases.

Different from them, this paper investigates a key problem in Web application testing: test case generation and optimization. It delves into an approach to testing and optimizing Web applications based on user sessions using genetic algorithm. When converting a user session to a corresponding test case, we preserve the user input data.

2. Collecting and Reducing User Sessions

In the logs on each Web server, each access record corresponds to a request by a user each time. The contents of the record include request source (user IP address), request time, request mode (such as GET, POST), the URL of requested information, data transport protocol (such as HTTP), status code, the number of bytes transferred and the type of client, etc. It needs to scan the logs only once to resolve the original active historic records from current access logs. However, it is difficult to organize these original records directly, which must be preprocessed. We first purge irrelevant data including the records whose status codes are erroneous (the code 200 for success, 400 for error), embedded resources such as script files and multimedia files whose extension names are .gif, .jpeg or .css, etc., to obtain the set of user sessions for primitive analysis. Then, we create user sessions through scanning the logs on Web servers. Once a new IP address occurs, a new user session is created. The sequential requests sent from this IP address are appended to the new session under the condition that the time interval of two continuous requests is not greater than max-session-idle-time pre-determined, or else another new user session begins. The set of all the user sessions is finally achieved.

There are often large volumes of user sessions collected and a user session is converted to a test case transmitted to Web server. Therefore, we can eliminate *redundant* user sessions using reduction techniques, and then preserve *necessary* user sessions. For the convenience of discussion, some important concepts are given first.

Definition 1 (URL trace). *URL trace* is the URL sequence requested by a user session.

Let α be a URL trace. Its length is the number of URLs requested in the trace, denoted by $|\alpha|$.

Definition 2 (prefix). A trace α is the *prefix* of another trace β, if and only if α is the subsequence of β and they have the same initial symbol.

Definition 3 (common prefix). If a trace is the prefix of several traces, then this trace becomes their *common prefix*.

Definition 4 (greatest common prefix). The longest common prefix in all the common prefixes of two traces is their *greatest common prefix*.

The longer the greatest common prefix of two traces is, the more similar the two traces are, *i.e.*, their similarity is higher. Let we have four traces that are γ_1 = abcdefg, γ_2 = abcdeh, γ_3 = abcd and γ_4 = cde. Then, γ_3 is the common prefix of γ_1 and γ_2, but not the greatest one. The greatest common prefix of γ_1 and γ_2 is abcde. Although γ_4 is the subsequence of γ_1 and γ_2, it is not their prefix, for the initial symbol of γ_4 is not the same as that of γ_1 and γ_2. We define a function isPrefix(α, β), which decides whether or not α is the prefix of β, *i.e.*, whether or not the URL trace requested by a user session is *redundant* corresponding to that requested by another user session. If there is a prefix, then the function returns a BOOLEAN value TRUE, or else returns FALSE. The URL trace-based user session algorithm ReduceUSession is shown in **Figure 1**. Note that, an HTTP request here is regarded as a symbol of a trace.

The ReduceUSession algorithm decides whether or not a URL trace α requested by a user session is the prefix of β requested by another user session. If it is true, then the user session corresponding to α is removed. The number of user sessions obtained using this algorithm will be reduced greatly.

The ReduceUSession algorithm is different from other

```
Algorithm: ReduceUSession
input:
    The set of user sessions Λ = {s1, …, sk}, where k is the number
    of user sessions;
    The URL trace U1, …, Uk, which are requested by s1, …, sk
    respectively;
output:
    The reduced set of user sessions denoted by Γ;
begin
    Γ = Φ;
    while (another user session that is not marked in Λ exists)
        tag1 = FALSE;
        tag2 = FALSE;
        Select a user session si that is not marked in Λ, and then
        mark it with "USED";
        for (the URL trace Uj requested by each user session sj in
        Γ)
            if isPrefix(Uj, Ui)      //the URLs requested by si is
                                     //more, so sj is redundant
                Γ = Γ-{sj};
                tag1 = TRUE;
            endif;
            if isPrefix(Ui, Uj)      //here, Γ keeps unchanged,
                                     //and si is redundant
                tag2 = TRUE;
                break;   //exit for cycle
            endif;
        endfor;
        if tag1 || (!tag1 && !tag2)      //si is necessary
            Γ = Γ ∪ {si};
        endif;
    endwhile;
    Output the reduced set of user sessions Γ;
end.
```

Figure 1. The URL trace-based user session reduction algorithm ReduceUSession

user session reduction algorithms. Most other reduction algorithms analyze each test case in the test suites one by one and remove those test cases that cannot change or affect test requirements, *i.e.*, they distinguish *redundant* and *necessary* test cases, as is too difficult to manage in practice, for it is very intractable to discriminate that the test requirements satisfied by some test cases (*i.e.*, redundant ones) are also satisfied by other test cases before the test execution using the existential algorithms. While our ReduceUSession algorithm decides whether or not a URL trace requested by a user session is the prefix of another URL trace requested by another user session. Based on this, we can identify the *redundant* test cases, as can be easily done in practice. Besides, it covers all the URLs requested by the original set of user sessions and keeps the sequence of URL requests, *i.e.*, it guarantees that the original test requirements are satisfied.

3. Grouping and Prioritizing User Sessions

The user sessions reduced by the ReduceUSession algorithm are divided into subgroups, each of which is regarded as a test suite. The goal of grouping is to reuse test cases and the testing can be executed at different platforms in parallel (or concurrently), to lessen test time and improve test efficiency. Moreover, the interacting test can be conducted between a pair of user sessions in each group (see Subsection 4.4 for the details). We try our best to keep the property for the user sessions in the same group that the URL traces requested bear greatest common prefix of a certain length. Several discontinuous integral threshold values, denoted by $\zeta_1, \zeta_2, \ldots\zeta_k$ ($\zeta_i \geq 1$, $1 \leq i \leq k$), are defined. The user sessions, the lengths of whose greatest common prefix of URL traces requested fall in between a certain threshold values, are grouped together. Fox example, let we have three threshold values $\zeta_1 = 2$, $\zeta_2 = 4$ and $\zeta_3 = 7$, then the user sessions are divided into four groups that are S_1, S_2, S_3 and S_4, the lengths of whose greatest common prefix (denoted by α) are $|\alpha| \leq 2$, $2 < |\alpha| \leq 4$, $4 < |\alpha| \leq 7$ and $|\alpha| > 7$ respectively. The test cases in these four groups can be executed at different platforms in parallel (or concurrently), and the greatest common prefix is also reused in each group. Notice that this is not the unique grouping way. For example, in the four traces γ_1 = abcdefg, γ_2 = abcdeh, γ_3 = abcd and γ_4 = cde, we can divide them into two groups $\{\gamma_1, \gamma_2\}$ and $\{\gamma_3, \gamma_4\}$, the lengths of whose greatest common prefix are $4 < |\alpha| \leq 7$ and $|\alpha| \leq 2$ respectively, or another two groups $\{\gamma_1, \gamma_2, \gamma_3\}$ and $\{\gamma_4\}$, the lengths of whose greatest common prefix are $2 < |\alpha| \leq 4$ and $|\alpha| \leq 2$ respectively. Of course, it is unnecessary to require that the greatest common prefix of URL traces requested by any two user sessions in each group fall in between a certain threshold values; it is recommended that most of them satisfy this property (a percentage can be pre-designed or generated randomly for the measurement). A compromise way needs to be found to classify these URL traces (or test suites), *i.e.*, a tradeoff should be found between grouping and concurrent testing and test reuse. Suppose that N user sessions are divided into K groups. In general, there is almost the same number of user sessions in one group as that in another, *i.e.*, the number equals approximately to $\left\lceil \frac{N}{K} \right\rceil$. Additionally, this way of grouping is preparatory, called *elementary grouping*.

The common prefix indicates the users' common events, or the same or similar operations. It also shows that the users bear the same or similar interests. The longer the common prefix is, the more evident it is, as is the case of most users. In addition, there is a special group of user sessions, the length of whose greatest common prefix of URL traces requested is shortest. This group of user sessions often indicates different URL requests, which represent distinct requirements for a Web application. In these sessions, many aberrant events often occur with unwonted input data. They belong to boundary cases, which are very easy to go wrong for the Web application. Herein, we prioritize test suites. The test suite with shortest length of common prefix ranks first, then all the other test suites are arranged according to their lengths of common prefix in descending order. So, the test suite in final position is that whose length of common prefix is last but one. In the test suites S_1, S_2, S_3 and S_4, the lengths of whose common prefixes are $|\alpha| \leq 2$, $2 < |\alpha| \leq 4$, $4 < |\alpha| \leq 7$ and $|\alpha| > 7$ respectively, if we prioritize them, then the test executing sequence for those test suites are S_1, S_4, S_3, S_2. That is to say, the test cases in S_1 are executed first, then the test cases in S_4 and S_3 are executed respectively and finally, the test cases in S_2 are executed. In each test suite, the test cases are prioritized according to the coverage ratios of URLs requested, *i.e.*, the test case with longer URL trace requested is executed earlier. If the lengths of URL traces of several test cases in the same test suite are equal, then they are randomly executed.

Our approach of prioritization is different from others. The existing methods of prioritization add the strongest test case, which is of the maximal use for the coverage ratio of test requirements, to the new test suite. Those methods aim to execute earlier the test cases of high priority than those of lower priority, to satisfy some test requirements as soon as possible. These methods, however, are often difficult to find the (nearly) strongest test case each time before test run, while our approach divides test cases into several groups according to the idea of common prefix before prioritizing them. It is more convenient for the testers to selectively execute the test cases of some group by grouping all the test cases first, to detect some

types of faults, for the same or similar types of errors are often detected by those test cases in the same groups. In addition, test cases can be reused by grouping and the testing can be executed at different platforms in parallel (or concurrently), to lessen test time and improve test efficiency. Moreover, we have considered a special type of user sessions, the length of whose greatest common prefix of URL traces is shortest. This group of user sessions often contains specific requests, which are the primary source for errors.

4. Testing Web Applications Using Genetic Algorithm

After the process of grouping and prioritization above, we obtain several initial test suites and test cases with the initial executing sequences. However, the test scheme generated by the elementary prioritization is not fast in finding faults and can not satisfy the requirements earlier, *i.e.*, it is not much approximate to the best one. Therefore, genetic algorithm is employed further to optimize the grouping and prioritization.

In 1975, an American professor Holland first proposed the idea of genetic algorithm systematically [13]. It has attracted a large number of researchers and extended into those aspects of optimization, search and machine learning with a solid theoretic foundation. The genetic algorithm focuses on all the individuals in one population, and uses random techniques to search efficiently for a coded parameter space. Selection, crossover and mutation are the basic operators in genetic algorithm; parameter coding, the setting of initial population, the design of fitness function, the selection of genetic operations and control parameters consist of the critical part of genetic algorithm. As a global optimization search algorithm of high efficiency, the genetic algorithm has distinctive advantage in solving the difficult problems in the domains of big space, multiple peak, nonlinearity and parallel processing, etc.

In the following, we test Web applications using genetic algorithm to further optimize the initial test suites and test cases, and their initial executing sequences, in order to achieve better test suites and test cases that satisfy test requirements. The process of yielding a population of next generation using three basic operators that are selection, crossover and mutation once is called *an iteration*. To obtain a good result, much iteration is repeated. The following introduces the process of selection, crossover and mutation.

4.1 Selection

A pair of individuals is selected from a parent population with the probability of p_s. The probability that an individual is selected is in direct proportion to its fitness value, as is often implemented using the strategy of roulette wheel [13]. In selecting, the individual of high fitness value is duplicated into the population of next generation directly. The higher the fitness value of an individual is, the higher the probability of yielding its offspring is, as shows that it is more appropriate to the expected result. Let we get K test suites (constituting the initial population), which are S_1, S_2, …, S_K of the descending order according to the prioritization technique. In practice, we combine the error coverage ratios of test suites and the cost of test run to design fitness function. The fitness value is listed as f_1, f_2, …, f_K from high to low, where f_i is the fitness of S_i ($1 \le i \le K$). The probability that an individual is selected equals to the resulting value that its fitness value divides the sum of fitness values of all the individuals, *i.e.*, the selected probability of S_i, denoted by p_s^{Si}, is $f_i \Big/ \sum_{j=1}^{K} f_j$. Obviously, the sum of the selected probabilities of all the K test suites equals to 1.

According to the discussion above, the fitness f_1 and f_2 corresponds to two special test suites S_1 and S_2, the lengths of whose greatest common prefixes are shortest and longest respectively. S_1 and S_2 are selected to be the individuals of next generation directly (in practical use, we can also select more than two individuals of high fitness to become the next individuals); in case they do not be selected. Now, we randomly yield two numbers that are g_1, $g_2 \in [0, 1]$, and randomly select two test suites that are S_i and S_j whose probabilities are not less than g_1 and g_2 respectively. The two new test suites S_i' and S_j' (the individuals of next generation) are generated through crossovering and mutating their parents S_i and S_j (see Subsections 4.2 and 4.3). Repeat the process of selection, until adequate test suites of next generation are yielded. One point should be emphasized that some test suites of higher probabilities may not be selected as parents, while others of lower probabilities are selected. This case is reasonable and accords with the theory of biological evolution in nature, because any thing has its necessity and occasionality at the same time.

4.2 Crossover

The genes chain of two parent individuals selected are crossovered with the probability of p_c using the TSCrossover algorithm, where p_c is a system control parameter. The new individuals after crossovering are used to replace their parents. The TSCrossover algorithm is shown in **Figure 2**.

Let we have two test suites $S_i = < c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9 >$, which contains 9 test cases and $S_j = < c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, c_{15}, c_{16} >$, which contains 7 test cases. If the position of crossovering is 3, then two new individuals of next generation after crossovering S_i and S_j are: $S_i' = < c_1, c_2,$ **$c_{12}, c_{13}, c_{14}, c_{15}, c_{16}$**, $c_8, c_9 >$ and $S_j' = < c_{10}, c_{11},$ **c_3, c_4, c_5, c_6, c_7** $>$, where the test cases indicated in bold face are those interchanged one by one from corresponding test cases in parent test suites.

4.3 Mutation

Each bits of the genes chain of new individuals are mutated with the probability of p_m using the TSMutation algorithm, where p_m is a system control parameter. The probability of mutation is often small, or else it will raise questions over the system stability. The populations can be prevented from stagnating through mutation. If there is no mutation, then the test data of new populations will be confined to the initial values. The mutation process is conducted respectively using the TSMutation algorithm for the test suites S_i' and S_j' crossovered using the TSCrossover algorithm. The new individuals after mutating are employed to replace those before mutating. The TSMutation algorithm is shown in **Figure 3**.

Let $p_m = 0.015$ and $S_i' = <c_1, c_2, c_{12}, c_{13}, c_{14}, c_{15}, c_{16}, c_8, c_9>$, which is one of the test suites after using the TSCrossover algorithm. We randomly generate some data: $g(c_1) = 0.246$, $g(c_2) = 0.580$, $g(c_{12}) = 0.025$, $g(c_{13}) = 0.012$, $g(c_{14}) = 0.632$, $g(c_{15}) = 0.073$, $g(c_{16}) = 0.431$, $g(c_8) = 0.193$ and $g(c_9) = 0.059$. For p_m is not less than $g(c_{13})$, c_{13} is mutated, *i.e.*, the positions of c_{13} and c_{12} are interchanged. The test suites S_i' after mutating becomes $< c_1, c_2, \mathbf{c_{13}, c_{12}}, c_{14}, c_{15}, c_{16}, c_8, c_9 >$.

4.4 The Interacting Testing among User Sessions

Some new test cases, which contain the requested information of different users, can also be generated using crossover. The goal of the new test cases is to detect errors caused by the use of possible conflicting data shared by different users. The crossover way shows the idea of information interchange among different user sessions. Let *S* be a test suite, the steps of generating a test case using crossover are:

1) for a test case *c* in *S*, randomly generate a number $g \in [0, 1]$, if a given crossover probability is not less than *g*, then

a) copy the requests from r_1 to r_i in *c* to an interim test case, where *i* is a random number, $i \in [1, |c|]$; $|c|$ is the length of URL trace in *c*;

b) select a test case *d* (different from *c*) in *S* randomly, and then search r_j in *d* reversely, which is the first one to have the same URL as r_i. If not found, then another test case (also denoted as *d*) is selected, until it is found or up to a given time;

c) if *d* is found, then all the requests after r_j in *d* are appended to the interim test case, which is the new test case generated; otherwise go 2);

2) repeat 1) until each test case in the test suite is processed.

We search the requests in test case *d* with an inverse search method, for the length of greatest common prefix of two test cases *c* and *d* in the same test suite is often greater than 1; when the corresponding URL trace from r_1 to r_i is the prefix of this greatest common prefix, we can not obtain different test cases if using the sequential search method. For example, given two URL traces $c = u_1u_2u_4u_3u_5u_6u_8u_7$ and $d = u_1u_2u_4u_3u_7u_4u_5u_9$, we randomly copy $u_1u_2u_4$ from c to an interim test case t firstly (*i.e.*, *t* equals to $u_1u_2u_4$ temporarily). Here, $I = 3$ and $r_i = u_4$. Now, we search r_j in *d* reversely, which is the first one to have the same URL as r_i and then we have $j = 6$ and $r_j = u_4$. So, all the requests after r_6 in *d* (*i.e.*, u_5u_9) are appended to *t* such that $t = u_1u_2u_4u_5u_9$. If searching the requests in *d* with a sequential search method, we have $j = 3$ and $r_j = u_4$. This time, if copying all the requests after r_3 in d (*i.e.*, $u_3u_7u_4u_5u_9$) to *t*, we have $t = u_1u_2u_4u_3u_7u_4u_5u_9$, which equals to *d* itself. And no new test case is generated. The

```
Algorithm: TSCrossover
input:
    Parent test suites Si and Sj;
    The crossover probability of pc;
output:
    Si' and Sj' of next generation;
begin
    Si', Sj'←Si, Sj;
    Randomly generate a number g∈[0, 1];
    if pc is not less than g
        Randomly generate an integral number g'∈(0, min(|Si'|,
        |Sj'|)];    //|Si'| and |Sj'| are the number

                    //of test cases in Si' and Sj' respectively
        Interchange the test cases in Si' and Sj' from the position of
        g' one by one,
        until no more test case needs to be interchanged in any of
        them;
    endif;
    Output Si' and Sj';
end.
```

Figure 2. The crossover algorithm TSCrossover for test suites

```
Algorithm: TSMutation
input:
    The test suites Si' and Sj' after using TSCrossover algorithm;
    The mutation probability of pm;
output;
    The test suites Si' and Sj' after mutating;
begin
    for each S in {Si', Sj'}
        for each c in S
            Randomly generate a number g∈[0, 1];
            if pm is not less than g
                Interchange the positions of c and the test case
                before c;
            endif;
        endfor
        Output S;
    endfor
end.
```

Figure 3. The mutation algorithm TSMutation for test suites

reason is that the corresponding URL trace (*i.e.*, $u_1u_2u_4$) from r_1 to r_i (i = 3) in c is the prefix of greatest common prefix (*i.e.*, $u_1u_2u_4u_3$) of c and d.

In addition, it is often not to mutate a test case, for the new generated test case after exchanging the two adjacent requests r_i and r_{i+1} makes no sense. The possible reason is that the former request r_{i-1} may never reach the request r_{i+1} (note that, this time r_i becomes the next request of r_{i+1}).

5. Experimental Analysis

Consider a typical miniature Web application developed to demonstrate our approach: the SWLS (Simple Web Login System) was shown in **Figure 4**.

Starting at the first page (indicated by a dashed arrow, reasonably, a *blank* page can be used to request for the first page of a Web application), *i.e.*, a *home page* (p_1), the user can enter into the *news page* (p_2) to list the news by clicking on the *view* link, or enter into the *login page* (p_3) by pressing the *login* button. In page p_3, the user enters the *userid* and *password*, and presses the *submit* button. Upon this pressing, the *userid* and *password* are sent to the Web server for authentication. A *logged page* (p_4) will be loaded if both *userid* and *password* are correct. On the contrary, an *error page* (p_7) containing an error message is displayed if at least one of the submitted values for *userid* and *password* is wrong. From the *logged page*, it is possible to go to *info page* (p_5) for secure information viewing by just clicking on the *browse* link. The user can click on the intra-page link *continue* to view the different parts of the same page p_5 if it is too long. A *logout page* (p_6) will be displayed when the user presses the *exit* or *logout* button. Then, the user may come back to the *home page* for login again. Note that each time the *login page* is displayed; both the *userid* and *password* fields should be initialized to be empty.

We inject only one fault in each page respectively, so at least 7 faults exist totally (there may be other faults in the Web application originally). A set of user sessions are obtained after scanning user logs on the Web server and purging their irrelevant information; then 89 meaningful user sessions are finally created after scanning them again. Only 17 user sessions are used to generate test cases after the reduction of the meaningful user sessions using the URL trace-based ReduceUSession algorithm; and the reduction ratio reaches 80.9%. Impersonally, more user sessions there are for a given Web application, much higher is the reduction efficiency, for more user sessions mean higher possibility that the URL trace requested by a user session is the prefix of that requested by another in the view of statistics. The set of 17 user sessions[†] (or test cases) is denoted by Γ1. After grouping and prioritizing Γ1, an initial executing sequence <S_1, S_2, S_3> of test suites is obtained, which is denoted by Γ2, where S_1, S_2 and S_3 contain initial executing sequences of 7, 6 and 4 test cases respectively. We achieve the final executing sequence <S_1', S_2', S_3'> of test suites using genetic algorithm for further processing, which is denoted by Γ3, *i.e.*, S_1', S_2' and S_3' is obtained through crossovering and mutating S_1, S_2 and S_3.

Now, we run the test suites (and test cases) in Γ1, Γ2 and Γ3 for the Web application respectively, and find all the 7 faults injected as well as an additional fault (*i.e.*, the user may browse p2 just by directly entering its address in the URL address bar). The executing time of Γ2 and Γ3, however, is much shorter than that of Γ1, and it takes less time of Γ3 than that of Γ2 too. This means that the test suites (and test cases) grouped and prioritized run faster than those, which are not grouped and prioritized; and that the test suites (and test cases) processed further by genetic algorithm are much faster. It is predictive that the test approach proposed in this paper will yield more evident positive effects for larger Web applications.

6. Concluding Remarks and Future Work

Generating test cases of high quality is the premise of Web application testing. The approach in this paper captures a series of user events, *i.e.*, the sequences of URLs and name-value pairs in Web server (s). It then employs the reduction, grouping, prioritization and genetic algorithm to yield test cases and optimizes them. Compared to the methods of capturing user events in clients, our approach is very effective when a large volume of users exist, and it is a Web application testing method of high efficiency. The main contributions include:

1) an approach to generating and optimizing test cases is proposed for Web application testing based on user sessions using genetic algorithm.

2) several important definitions such as URL trace, prefix, common prefix, greatest common prefix, are given. These definitions are convenient for reducing and grouping user sessions.

3) a URL trace-based reduction algorithm is designed. The user sessions acquired are lessened greatly using the

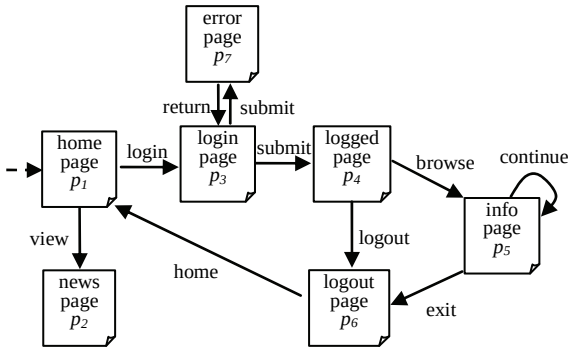


Figure 4. A simple web login system

[†] For the convenience of test run, we have preprocessed the user sessions appropriately.

algorithm. However, it covers all the URLs requested by the original set of user sessions and keeps the sequence of URL requests.

4) an approach to grouping and prioritizing user sessions is presented, as can improve the efficiency of test run.

5) an approach to generating new test cases is proposed using crossover. The new test cases generated include information requested by different users and can detect errors caused by the use of possible conflicting data shared by different users. That's to say, the crossover indicates the idea of information interchange among different user sessions, which helps to test the interacting of user sessions.

6) a strategy of test reuse and concurrence is provided, as can decrease the time of test run greatly as well as lessen test cost.

Web application testing is much complex systems engineering. It is not easy to acquire an effective and practical test scheme. Our approach only evaluates a test case according to its test coverage ratio. However, many factors need to be considered, such as the running cost of each test case itself (for example, CPU time), including the actual running time, loading time, time to save test state, and the influence of different test criteria on a test case. All these questions need to be answered in future research.

REFERENCES

[1] E. Hieatt and R. Mee, "Going Faster: Testing the Web Application," *IEEE Software*, Vol. 19, No. 2, 2002, pp. 60-65.

[2] D. C. Kung, C. H. Liu and P. Hsia, "An Object-Oriented Web Test Model for Testing Web Applications," *Proceedings of the* 1*st Asia-Pacific Conference on Web Applications*, New York, 2000, pp. 111-120.

[3] J. Offutt, Y. Wu. and X. Du, *et al*., "Bypass Testing of Web Applications," *Proceedings of the* 15*th IEEE International Symposium on Software Reliability Engineering*, Bretagne, November 2004.

[4] C. H. Liu, D. C. Kung and P. Hsia, "Object-Based Data Flow Testing of Web Applications," *Proceedings of the* 1*st Asia-Pacific Conference on Quality Software*, Hong Kong, 2000, pp. 7-16.

[5] C. Elbaum, G. Rothermel and S. Karre, *et al*., "Leveraging User Session Data to Support Web Application Testing," *IEEE Transaction on Software Engineering*, California, May 2005.

[6] R. Godin, R. Missaoui and H. Alaoui, "Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices," *Computational Intelligence*, Vol. 11, No. 2, 1995, pp. 246-267.

[7] S. Khor and P. Grogono, "Using a Genetic Algorithm and Formal Concept Analysis to Generate Branch Coverage Test Data Automatically," *Proceedings of the International Conference on Automated Software Engineering*, Austria, 2004, pp. 346-349.

[8] H. H. Sthamer, "The Automatic Generation of Software Test Data Using Genetic Algorithms," PhD. Dissertation, University of Glamorgan, Wales, 1996.

[9] D. Berndt, J. Fisher and L. Johnson, *et al*., "Breeding Software Test Cases with Genetic Algorithms," *Proceedings of the* 36*th Hawaii International Conference on System Sciences*, 2003, pp. 17-24.

[10] R. P. Pargas and M. J. Harrold, "Test-Data Generation Using Genetic Algorithms," *Journal of Software Testing, Verification and Reliability*, Vol. 9, No. 4, 1999, pp. 263-282.

[11] X. X. Jia, J. Wu, M. Z. Jin, *et al*., "Some Experiment Analysis of Using Generic Algorithm in Automatic Test Data Generation," in Chinese, *Journal of Chinese Computer Systems*, Vol. 28, No. 3, 2007, pp. 520-525.

[12] D. J. Berndt and A. Watkins, "Investigating the Performance of Genetic Algorithm-Based Software Test Case generation," *Proceedings of the International Symposium on High Assurance Systems Engineering*, Florida, 2004, pp. 261-262.

[13] J. H. Holland, "Adaptation in Natural and Artificial System," University of Michigan Press, Michigan, 1975.

J. Software Engineering & Applications, 2010, 3, 548-555
doi:10.4236/jsea.2010.36063 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Tabu Search Solution for Resource Confidence Considered Partner Selection Problem in Cross-Enterprise Project

Hanchuan Xu, Xiaofei Xu, Ting He

School of Computer Science & Technology, Harbin Institute of Technology, Harbin, China.
Email: {xhc, xiaofei, xuantinghe}@hit.edu.cn

Received April 1st, 2010; revised April 21st, 2010; accepted April 23rd, 2010.

ABSTRACT

Cross-enterprise project is the main implementation form in multi enterprises collaborative production environment. Minimizing the risk of failure and tardiness caused by the uncertainty of partner's resources in partner selection is the key problem to ensure success in Cross-enterprise project. In this paper, considering the factors and constraints of sub-project processing times, precedence of sub-project and project due date, especially the resource confidence, a 0-1 *integer programming model was presented with the objective to minimize the risk of failure and the tardiness of the project. A project scheduling algorithm was designed to search and evaluate selection solutions, and the project scheduling algorithm was embedded into a Tabu search algorithm to solve the model. Simulation experiments and comparisons with other algorithms showed that the proposed approach was possible to find the optimal solution with a faster speed and higher probability.*

***Keywords*:** *Cross-Enterprise Project, Partner Selection, Resource Confidence, Tabu Search*

1. Introduction

To be competitive in the global manufacturing environment with the rapidly increasing competitiveness, strategic collaborations between enterprises are needed in order to meet the market's requirements for quality, responsiveness, and customer satisfaction. Dynamic alliance, virtual enterprise, extended enterprise, and supply chain are the major management philosophies for multi enterprises collaborative production environment. Cross-enterprise project (CEP) management pattern arose as the main implementation form in these management philosophies. CEP is the formation of closer co-ordination in the design, development, costing and the co-ordination of the respective manufacturing schedules of co-operating independent manufacturing enterprises and related suppliers [1,2]. There are four stages in CEP life cycle: formation, operation, evolution and dissolution. A major issue in the formation phase is to select appropriate partners and allocate tasks between partners. During this process, the core enterprise comprehensively evaluates partners according to cost, quality, credit, delivery time, etc., and then based on certain criteria, find the best combination of partners to collaborate to complete the project.

Partner selection has attracted much research attention recently, because it is an important function for information management systems, such as project management (PM), enterprise resource planning (ERP) and supply chain management (SCM). Most of researches about the partner selection problem are based on qualitative analysis methods. However, quantitative analysis methods for partner selection are still insufficient. Many operation research methods, such as analytic hierarchy process (AHP), analytic network process (ANP) and fuzzy synthetic evaluation are widely used to the problem, but mathematical models and optimization methods for partner selection are still a challenge [3,4]. In partner selection process, although there are many factors needed to be considered such as friendship, credit, quality, and reliability, the cost and completion time are the two key factors and focused on by most researchers.

In CEP, the resources belong to different partners which often undertake other projects at the same time, so the available production capacity of partner will be tight during some periods. In addition, unforeseen exceptions also inevitably occur. All of these make resources have a certain degree of uncertainty. Although there are some

constraints between enterprises such as contracts, the case that partners can't provide promised resources on time or in right quality is not able to be completely avoided. When this happens, it will seriously affect the production schedule of core enterprise and even cause the whole cross-enterprise project failed. So the confidence level of resource that can be provided by partners on time and in right quality which we defined as *resource confidence* is a very important factor in the partner selection problem. However, most researchers mainly take into account the cost and completion time, and the objective is to minimize the total cost of the project or the project duration. The factor of resource confidence is neglected in most researches.

Yannis and Andreas [5], Sha and Che [6], Mikhailov [7] and other researchers proposed some models and approaches for partner selection by establishing a CEP, where cost, time and distance were considered. However, the other two important factors, the resource confidence and the precedence of sub-project, were not considered in their papers. As Wang *et al*. [8] indicates, in the cooperation relationship of sub-projects contracted by partners, it may be represented by an activity network with precedence. Thus, the problem could be considered as a partner selection problem embedded with project scheduling. Naiqi *et al*. [9] considered the completion time as a constraint and modeled the partner selection problem by an integer programming formulation to minimize the manufacturing cost. The formulation was then transformed into a graph-theoretical formulation and a 2-phase algorithm was developed to solve the problem. Wang *et al*. [8] took into consideration the factors of cost, completion time and precedence of sub-project, described the partner selection problem with a 0-1 integer programming formulation to minimize the total cost of the project. They then developed a fuzzy decision embedded heuristic genetic algorithm to find the solution for partner selection. The experiments showed that the algorithm was possible to quickly achieve optimal solution for large size problems. Taking into account the same factors and objective as Wang [8], W. H. IP *et al*. [10] and Zhibin *et al*. [11] separately proposed branch and bound solutions for partner selection problem in virtual enterprises and their solutions were especially effective to medium or small size problems. In all these papers mentioned above, their objectives were minimizing the total cost of project and didn't consider the impact of resource confidence to project implementation risk. To minimize risk in partner selection and ensure the due date of a project in virtual enterprise, W. H. IP *et al*. [12] described and modeled a risk-based partner selection problem. They developed a rule-based genetic algorithm with embedded project scheduling to solve the problem. In their paper, they assumed that each candidate partner had a fail probability of its contracted sub-project. In fact, the fail probability of the sub-project closely related to whether the partner's available resources were tight or not in the duration of the sub-project. The tighter the resources are, the higher the fail probability is, and vice versa. However, the production load of partner is varied in different periods, so the available resource and fail probability are also different. They used a whole fail probability for all periods and didn't consider the difference of fail probability in different periods.

To solve the partner selection problem in CEP, we first describe it with a 0-1 integer programming model considering the factors of process time, precedence of sub-projects, and resource confidence. Then a project scheduling algorithm is proposed to calculate the project completion time and the time window of each sub-project under a feasible solution. From this, we embed the project scheduling algorithm into a Tabu search algorithm to obtain the optimal partner selection solution. The computation results showed that the proposed approach is efficient to achieve the optimal solution.

The paper is structured as follows. In Section 2 the problem and the model are introduced. Then the solution space reduction method and the project scheduling algorithm to evaluate selection solution are presented in Section 3. The Tabu search algorithm embedded the project scheduling is presented in Section 4. Section 5 reports an experimental example and computational results obtained by testing the algorithm on some test instances. Finally, Section 6 presents our overall conclusions.

2. Model for Partner Selection Considering Resource Confidence

The problem of partner selection for CEP considering resource confidence can be described as follows:

Assuming an enterprise (core enterprise) obtain a big project consisting of several sub-projects. It is not able to complete the big project by itself from its own resources and has to find some partner enterprises to collaboratively finish the project. The partner selection procedure is divided into two phases. Firstly, the enterprise determines the payment and some basic requirements for the process time and quality of each sub-project. The partners who can accept the conditions will propose the process time they need to finish the sub-project according to their own capacity. They constitute the candidate partner set. In the second phase, the enterprise comprehensively evaluates all candidates and calculates the confidence of resource provided by the partner for the sub-project. At last the enterprise selects the most appropriate partner for each sub-project. There exists plenty methods to evaluate the individual candidate partner and calculate the resource confidence, for an extensive review we refer to MALONI and BENTON [3] and BOER *et al*. [4]. In this paper, we just focus on the second phase, *i.e.*, how to select partners according to the resource confidence.

Based on the risk-based partner selection model pre-

sented by W. H. IP *et al*. [12] and considering the resource confidence, we present a 0-1 integer programming model MPCPS for the partner selection problem. Suppose that the project consists of n sub-projects, there are precedence relationship between these sub-projects and they form a precedence activity network H. If sub-project k can only begin after the completion of sub-project i, we call sub-project i as the immediate predecessor of sub-project k and define the connected sub-project pair by $<i,k>\in H$. For the convenience of description, we label these sub-projects such that $i<k$. Without the loss of generality, the final sub-project is labeled as sub-project n. Thus, we can define that the completion time of final sub-project c_n is the completion time of the project. Each sub-project has some candidate partners, for sub-project i, $i=1,2,...,n$, there are m_i candidate partners, and for the candidate partner j of sub-project i, its processing times is q_{ij} periods. The resource confidence of candidate j to sub-project i in period t is noted as $d_{ij}(t)$ and there $d_{ij}(t)\in(0,1]$. The due date of the project is *D*. To simplify the problem, we assume that the core enterprise will select only one candidate to undertake one sub-project.

The objective is to select the optimal combination of partners for all sub-projects in order to maximize the whole resource confidence of the project and to finish the project within the due date.

The following decision variable is defined.

$$x_{ij}(t)=\begin{cases}1 & \text{candidate } j \text{ is selected to sub-project } i \text{ at period } t\\ 0 & \text{otherwise}\end{cases}$$

Then the problem can be modeled as follows:

MPCPS

$$\max_{x} F_d(x)=\prod_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{t=1}^{c_n}x_{ij}(t)\Big[\frac{1}{q_{ij}}\sum_{\tau=t}^{t+q_{ij}-1}d_{ij}(\tau)\Big]+\beta(1-[[c_n-D]^*]^-) \tag{1}$$

$$\text{s.t.}\quad \sum_{j=1}^{m_i}\sum_{t=1}^{c_n}x_{ij}(t)=1;i=1,2,...,n \tag{2}$$

$$(t+q_{ij})\sum_{j=1}^{m_i}\sum_{t=1}^{c_n}x_{ij}(t)\le t\sum_{j=1}^{m_k}\sum_{t=1}^{c_n}x_{kj}(t);t=1,2,...,c_n,\forall<i,k>\in H \tag{3}$$

$$\sum_{j=1}^{m_n}\sum_{t=1}^{c_n}(t+q_{nj})x_{nj}(t)=c_n \tag{4}$$

$$x_{ij}(t)\in\{0,1\}\forall i,j,t \tag{5}$$

where $[x]^+$ stands for $\max\{0,w\}$, $[y]^-$ stands for $\min\{1,y\}$ and β is the tardiness penalty coefficient. Formula (1) is the objective function, where $\frac{1}{q_{ij}}\sum_{\tau=t}^{t+q_{ij}-1}d_{ij}(\tau)$ is the mathematical expectation of resource confidence for candidate j of sub-project i in the q_{ij} continuous periods, and abbreviated as E_{ij} bellow. Constraint (2) ensures that each sub-project will be contracted to only one partner and constraint (3) is the precedence constraint of sub-projects. Constraint (4) gives the method to calculate the completion time of the project.

It is obvious that the operator $[x]^+$ and $[y]^-$ are non-analytical and the objective function is not continuous and differentiable, so it is difficult to treat the model by traditional mathematical programming methods. Therefore, we develop a project scheduling embedded Tabu search (PSTS) algorithm to solve this problem.

3. Solution Space Reduction and Evaluation Algorithm of Solution

3.1 Solution Space Reduction

The partner selection modeled as MPCPS is a complex combinatorial optimization problem. The number of feasible solutions (solution space) is very large, even for a small-scale problem. To simplify the solving process, W. H. IP *et al*. [12] defined the concept of *inefficient candidate* and proved that the optimal solution consists without any inefficient candidate. To efficiently reduce the solution space, all inefficient candidates can be ignored in the solving process without losing the optimal solution. Based on the definition presented by W. H. IP [12] and considering the characteristics of the model MPCPS, we define the *inefficient candidate* to our model as follows.

Definition 1. For the two candidates j and k of sub-project i, if $\forall t\in[ES_i^{\min},LF_i^{\max}]$ $\forall t\in[ES_i^{\min},LF_i^{\max}]$, $q_{ik}\le q_{ij}, d_{ik}(t)>d_{ij}(t)$, or $q_{ik}<q_{ij}, d_{ik}(t)\ge d_{ij}(t)$, then the candidate j is called inefficient candidate.

It is easy to prove that there exists at least one optimal solution which doesn't include any inefficient candidate. Therefore all the inefficient candidates are not considered in the model without loss of the optimal solution. In definition 1, $ES_i^{\min}$ is the earliest possible start time and $LF_i^{\max}$ is the latest allowable finish time of sub-project i. The longest possible time window $[ES_i^{\min},LF_i^{\max}]$ of sub-project i is only a part of the whole project time window $[1,c_n]$, so to judge whether a candidate is inefficient or not, we only need to do the judgment in time window $[ES_i^{\min},LF_i^{\max}]$ according to definition 1, rather than in the whole project time window. This method im-

proves the satisfiability of definition 1 and the effect of solution space reduction. Let $q_i^{\min}$ and $q_i^{\max}$ be the shortest and longest process time of sub-project i respectively, $q_i^{\min} = \min\{q_{i1,} q_{i2,...,} q_{im_i}\}$, $q_i^{\max} = \max\{q_{i1,} q_{i2,...,} q_{im_i}\}$, then $ES_i^{\min}$ and $LF_i^{\max}$ can be calculated with $q_i^{\min}$ and $q_i^{\max}$. This is a project scheduling problem with the objective of minimizing project makespan, *i.e.*, $PS|prec|C_{\max}$ and there exists some polynomial time algorithms [13].

3.2 Project-Scheduling-Based Solution Evaluation Algorithm

In our TS algorithm, the natural number string is selected as code description. Let $x = [x_1, x_2, ..., x_n]$, where x_i is an integer between 1 and m_i. This stands for that candidate x_i is selected for sub-project i. Thus, $x = [x_1, x_2, ..., x_n]$ is called a selection. For example, [3 4 1 5 2 3 4] is a partner selection of a project with 7 sub-projects. In the selection, the candidate 3 is selected for the sub-project 1, and the candidate 4 is selected for the sub-project 2, and so on.

Once a selection fixes the candidates for all sub-projects, to obtain the object function value, a project-scheduling-based solution evaluation algorithm *PSLP* can be done to calculate the variable values of c_n and E_{ij}, and also the object function value. The procedure of the algorithm PSLP is described as follows:

Algorithm PSLP

Step 1: Calculate the earliest starting time ES_i and the earliest finish time EF_i of each sub-project $i (i = 1, 2, ..., n)$. If do not exist $(k, i) \in H, \forall k \in \{1, 2, ..., n\}$, then $ES_i \leftarrow 0$; Else, $ES_i \leftarrow \max\{EF_k, \forall (k, i) \in H\}$. $EF_i \leftarrow ES_i + q_{ix_i}$.

Step 2: Calculate the project completion time c_n. Let $LF_n \leftarrow EF_n$, $LS_n \leftarrow ES_n$, $c_n \leftarrow LF_n$.

Step 3: Calculate the latest starting time LS_i and the latest finish time LF_i of each sub-project $i (i = 1, 2, ..., n)$. If do not exist $(i, k) \in H$, $\forall k \in \{1, 2, ..., n\}$, then $LF_i \leftarrow LF_n$; Else, $LF_i \leftarrow \min\{LS_k, \forall (i, k) \in H\}$. $LS_i \leftarrow LF_i - q_{ix_i}$.

Step 4: Calculate the float time FF_i of each sub-project $i (i = 1, 2, ..., n)$ and judge whether it is critical sub-project or not. $FF_i \leftarrow LS_i - ES_i$.

If $FF_i = 0$, then $U_c \leftarrow U_c \cup \{i\}$, Else $U_{nc} \leftarrow U_{nc} \cup \{i\}$.

Step 5: For each non-critical sub-project in U_{nc}, calculate the maximum mathematical expectation of resource confidence by the step-by-step right-shifted procedure.

Step 5.1: If $U_{nc} = \varnothing$, then go to Step 6; Else get the non-critical sub-project i from U_{nc} which has the maximum earliest start time. $U_{nc} \leftarrow U_{nc} \setminus \{i\}$, $Pos \leftarrow 0$, $E_{ix_i} \leftarrow 0$.

Step 5.2: For $j = 0$ to $FF_i - 1$ Do If $E_{ix_i} < \dfrac{1}{q_{ix_i}} \sum_{k=j}^{j+q_{ix_i}-1} d_{ix_i}(ES_i + k)$ then $E_{ix_i} \leftarrow \dfrac{1}{q_{ix_i}} \sum_{k=j}^{j+q_{ix_i}-1} d_{ix_i}(ES_i + k)$, $Pos \leftarrow j$ End For.

Step 5.3: Adjust the float time of each immediate preceding non-critical sub-project of sub-project *i*. For $\forall k \in U_{nc}$, and $(k, i) \in H$, let $FF_k \leftarrow FF_k + Pos$, go to Step 5.1.

Step 6: For each critical sub-project $i \in U_c$, calculate the maximum mathematical expectation of resource confidence, $E_{ix_i} \leftarrow \dfrac{1}{q_{ix_i}} \sum_{k=0}^{q_{ix_i}-1} d_{ix_i}(ES_i + k)$.

Step 7: Calculate the objective function value. $F_d(x) \leftarrow \prod_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{t=1}^{c_n} x_{ij}(t) E_{ij} + \beta(1 - [[c_n - D]^+]^-)$

Step 8: Over.

In the PSLP algorithm, the time window of each sub-project is calculated first. There, $[ES_i, ES_i + 1, ..., LS_i]$ and $[EF_i, EF_i + 1, ..., LF_i]$ represent the starting time window and finish time windows of sub-project *i*, respectively. In addition, the project critical path, the critical sub-project set U_c, the non-critical sub-project set U_{nc} and the float time of each sub-project can also be determined. In the second part, based on the idea of solving the resource levelling project scheduling with fixed project duration problem, and considered the characteristics that the resource confidence is various in each period and non-critical sub-project has float time, a step-by-step right-shifted procedure is employed to find the time section in which the non-critical sub-project has the greatest mathematical expectation of resource confidence. In the procedure, non-critical sub-projects are dealt with in accordance with descending order of the earliest start time. At the last part, the objective function value for a selection is obtained.

4. Tabu Search Algorithm Design

The Tabu search (TS) algorithm is an effective method for solving large-scale combination optimization problem. The TS algorithm can overcome the shortcomings of

some common heuristic algorithms which only adapt to special problems and easily sink into local optimal solutions. TS has been used on an increasing number of practice problem and has proven to be effective [14,15].

4.1 The Initial Solution

The initial solution is very important to TS and a good one is very useful to improve the constringency speed to optimal solution. Considering the fact that, in the widest feasible time window $[ES, ES+1, ..., LF]$ of the sub-project, the greater the resource confidence mathematical expectation of the candidate is, the higher the probability of being selected is, the initial solution generation procedure *PINI* is designed as the following steps.

Step 1: From sub-project $i=1$ to *n*, calculate the widest feasible time window $[ES_i, LF_i]$.

Step 2: From sub-project $i=1$ to *n*, find candidate x_i which has the greatest mathematical expectation of resource confidence in all the candidates of sub-project i.

Step 3: output the initial solution $x=[x_1, x_2, ..., x_n]$.

It is obvious that the initial solution is feasible.

4.2 Neighbourhood Structure and Candidate Solution

Considering the natural number string employed in this algorithm, neighbor is defined as all feasible solutions obtained through changing the value of one bit of the current solution, *i.e.*, changing a candidate partner of a sub-project. Moving from the current solution to a solution in the neighborhood is called a move. Therefore, one step in a move can change only one partner of the current solution. Let NB be the neighbor set. Evaluation value of each solution in NB can be calculated by the PSLP algorithm. The solution in NB will be selected as the candidate solution with meeting the conditions that it has the greatest evaluation value and the move from the current solution to it is not in the Tabu list.

In the solution evaluation algorithm PSLP described in 3.2, Step 5 is to precisely calculate the maximum mathematical expectation of resource confidence and the step-by-step right-shifted procedure has high CPU time cost. In fact, if a solution causes the whole project delayed, its evaluation value will be penalized with the penalty factor β in the Formula 1. Therefore, the solution has very low probability to be selected as the candidate solution. From this point of view, the following tardiness penalty evaluation procedure of candidate solution *CSTPE* is designed.

Procedure CSTPE:

Step 1: Implement the Steps 1-3 in the algorithm PSLP to calculate the time window of each sub-project and the completion time of the whole project.

Step 2: If the project isn't finished within the due date in the solution, then the Step 5 of PSLP will not be run and the subsequent steps will run with the time window $[ES_i, LF_i]$ obtained by the Steps 1-3, else the total PSLP algorithm will be run.

Experiments show that this candidate solution evaluation procedure can significantly reduce the time cost and still can find the optimal solution with high probability. The detailed analysis is described in the Section 5.

4.3 Tabu List

The Tabu list (TSL) is composed of a two-dimensional integer array. The number of rows is the length of the Tabu list, the first column is the code of the sub-project, and the second column is the code of the candidate partner corresponding to the sub-project in the first column. The code for every row records a solution in the neighborhood that has been deleted in recent movements. TSL is renewed according to the criterion of first in, first out.

4.4 Longer-Term Tabu List and Tabu Relaxation

To avoid getting into the local optimum and the cycling, two special techniques, longer-term Tabu list (TTL) and Tabu relaxation, are used. TTL is created to dynamically forbid moving overactive nodes in order to get diversification and help to prevent cycling. The algorithm incorporates a move frequency table to record the move frequency of each sub-project. When a sub-project's partner is changed, its move frequency is incremented by 1. If a sub-project's partner *x* has been moved more than two times and TTL is not full, it will be put into TTL. If TTL is full and if some sub-project's candidate partner *y* already in TTL has a lower move frequency than *x*, *y* will be dropped and *x* will be added into TTL.

Another technique used is the relaxation of Tabu lists. If a given number of iterations (*relaxed_tries*) has elapsed and TTL is full since the last best solution was found, which means the search process has plunged into a local optimal solution or a cycling, both TSL and TTL are emptied and using the current solution as the initial solution to continue the search. Relaxation of the Tabu lists will change the neighborhood of the current solution dramatically, which will lead to a rapid downhill movement and may lead to new search spaces.

4.5 The Aspiration Criterion and Stopping Rule

The Tabu status of a move can be overruled if a solution is feasible and is better than any feasible solution known so far.

In our PSTS algorithm, there are two ways of controlling the execution time: the maximum total number of iterations (max_*tries*) and the maximum number of iterations without improvement of the best known feasible solution (max_*unchanged*). The execution of the algo-

rithm is stopped when the number of iterations max_*tries* and max_*unchanged* are both attained, or when the number of iterations doubles max_*tries*. Therefore, the total number of iterations is not known in advance, depending on the evolution of the search. The combination of the max_*tries* and the max_*unchanged* as stopping criterion allows the search to continue if the algorithm is exploring a new promising region. Obviously, to give time for improvement after the restart, the max_*unchanged* should be greater than the *relaxed_tries*.

4.6 Global Description of the Algorithm

Algorithm PSTS:

Step 1: Specify the parameters. Set initial values of max_*tries*, max_*unchanged*, relaxed_*tries*, set iteration counter to $tries \leftarrow 0$, iteration counter for no improvement to $unchanged \leftarrow 0$.

Step 2: Generate an initial solution x using the algorithm PINI and calculate the evaluation value $F(\mathrm{x})$ of x using the CSE procedure. Let the current best solution $x^* \leftarrow x$, the best evaluation value $F^* \leftarrow F(\mathrm{x})$.

Step 3: $tries \leftarrow tries + 1$, if $tries > \max\_tries$ and $unchanged > \max\_unchanged$, or if $tries > 2 \times \max\_tries$, then go to Step 7, else go to Step 4.

Step 4: If $unchanged >$ relaxed $\_\, tries$, and *TTL* is full, then empty *TSL* and *TTL*.

Step 5: Find the neighborhood *NB* of x, calculate the candidate solution x_{NB} and the evaluation value $F(x_{NB})$. Calculate the best Tabu solution x_{TSL} and the corresponding evaluation value $F(x_{TSL})$ from *TSL*.

Step 6: Considering the TSL,TLL and the aspiration criterion, generate the new solution x from x_{NB} and x_{TSL}. If $F^* < \max\{F(x_{NB}), F(x_{TSL})\}$, $x^* \leftarrow x$, $F^* \leftarrow \max\{F(x_{NB}), F(x_{TSL})\}$, $unchanged \leftarrow 0$; else $unchanged \leftarrow unchanged + 1$. Go to Step 3.

Step 7: Output x^* and F^* is the optimal solution, algorithm is over.

If the CSTPE procedure is used to evaluate candidate solution, then the PSTS will be named as PSTS-P. Otherwise, if just the completed PSLP is used, the PSTS will be named as PSTS-NP.

5. Experiments Analysis

The algorithm was coded by JAVA and run on a Pentium Dual 2.2 GHz PC.

The example is a project that consists of 14 sub-projects and the core enterprise calls tenderers for the sub-projects. The precedence relationship represented by the active-on-node network is shown in **Figure 1.** The numbers

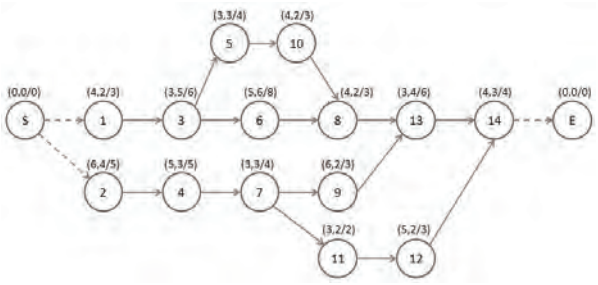


Figure 1. Example of a project represented by active network

in the parentheses are the number of candidates, the shortest process time and the longest process time, respectively. The project's due date is 24 periods and each sub-project has 3 to 6 candidates. The solution space size is 1.05×10^8. Through identifying and removing the inefficient candidates according to Definition 1, the size is reduced to 2.83×10^7. Different candidates may have different process time to the sub-project that they bid for. The resource confidence of the candidate in each period $d_{ij}(t)$ is calculated by the fuzzy comprehensive evaluation method, and where $d_{ij}(t) \in (0,1]$. For the limitation of the paper size, the detailed data is omitted.

The setting for the values of parameters is important for the efficiency of TS algorithm. In our algorithm, the "Best_rate" is used to evaluate and adjust the values of parameters, where "Best_rate" is the rate to reach the optimal solution in a certain number of runs. Based on the algorithms of IP WH *et al.* [10] and Zhibin *et al.* [11], considering the characteristics of our problem, a branch- bound algorithm (B & B) is designed to calculate the optimal solution. For different scale problems, the algorithm was run a certain number times according to the scale of the problem with different random seeds for each parameter setting. Therefore, the parameters with the highest "Best_rate" are selected. To the example in **Figure 1**, the values of the parameters are "max_*tries* = 700", "max_*unchanged* = 80", "relaxed_*tries* = 60", the length of TSL is 18, and the length of TLL is 70. The result of the example is shown in **Table 1**, and the objective value is 0.241.

For testing the performance of the PSTS algorithm, we randomly generated some problems with different scales. The comparison results of the PSTS-P, PSTS-NP and B&B are shown in **Table 2**. Where "N" is the number of sub-projects, "size" stands for the size of solution space, "CPU time" is the average computation time of each running.

The B & B algorithm is a kind of exact algorithm and can always find the optimal solution (best rate is always 100%), but it needs much more running time to deal with large scale problems. The two recommended algorithms, PSTS-P and PSTS-NP, can achieve the optimal solution with a high best rate and the computation time doesn't

Table 1. The selected partner list, processing time and resource confidence

Sub-project no.	Selected partner code	Processing time	Start time	Finish time	Resource confidence
1	A2	2	0	2	0.82
2	B3	4	0	4	0.80
3	C1	5	3	7	0.76
4	D4	4	6	9	0.55
5	E1	3	9	11	0.60
6	F5	7	8	14	0.72
7	G2	3	11	13	0.58
8	H1	3	15	17	0.63
9	I5	2	15	16	0.69
10	J3	2	12	13	0.72
11	K2	2	14	15	0.83
12	L4	3	17	19	0.82
13	M1	5	18	22	0.91
14	N2	4	23	26	0.87

Table 2. The comparison of PSTS-P, PSTS-NP and B & B

N	Size	PSTS-P				PSTS-NP				B & B		
		CPU time (s)	Best rate (%)	Best	Avg	CPU time (s)	Best rate (%)	Best	Avg	CPU time (s)	Best rate (%)	Best & Avg
12	5.24×10^{8}	13.32	100	0.231	0.231	33.32	100	0.231	0.231	48.56	100	0.231
18	4.63×10^{10}	33.67	100	0.256	0.256	69.41	100	0.256	0.256	101.72	100	0.256
24	7.51×10^{14}	46.59	100	0.217	0.217	85.73	100	0.217	0.217	201.34	100	0.217
33	3.04×10^{18}	79.41	98	0.134	0.132	98.41	100	0.134	0.133	579.31	100	0.134
38	1.46×10^{21}	88.76	96	0.158	0.156	108.37	99	0.158	0.157	783.85	100	0.158
45	2.35×10^{22}	97.58	94	0.179	0.176	120.78	97	0.179	0.178	1084.67	100	0.179
48	6.35×10^{24}	113.62	91	0.092	0.089	156.39	96	0.092	0.090	9457.36	100	0.092
52	4.53×10^{26}	130.68	88	0.087	0.083	210.65	94	0.087	0.085	17613.09	100	0.087
58	8.63×10^{27}	149.31	86	0.083	0.078	290.25	93	0.083	0.080	29465.06	100	0.083
64	9.27×10^{35}	173.24	82	0.062	0.053	362.47	90	0.062	0.056	38280.77	100	0.062

grow fast with the problem size increase. For PSTS-P using the CSTPE procedure to evaluate candidate solutions, it can solve large problems faster; on the other hand, PSTS-NP has higher rate to obtain optimal solution. In practice, we can select the appropriate one from the two algorithms according to the different requirements of speed and best rate.

6. Conclusions

Partner selection is a complicated and practical problem in CEP. Minimizing risk caused by the uncertainty of partner's resources in partner selection and ensuring the due date of the project are important to the success of the CEP. This paper introduces a description of the partner selection problem in CEP. The concept of resource confidence is used to characterize the uncertainty of partner's resources, then the non-linear integer programming model (1-5) provides a formal description of the partner selection problem of CEP with considering resource confidence, where the following features different from conventional methods are considered:

1) The precedence activity network describing the precedence relationship between sub-projects

2) The resource confidence of each partner

A project scheduling embedded TS algorithm for the problem was proposed. Its two variants, PSTS-P and PSTS-NP, focus on computation speed and optimizing efficiency, respectively. The computation results show its potential to solve practical partner selection and project management problems.

The suggested future research work includes: a) Find a better way to share information between the core enterprise and partners, and research how to evaluate and calculate the resource confidence of partners more accurately. b) Research project planning model and algorithms for the CEP with considering resource confidence.

REFERENCES

[1] X. F. Xu, "Virtual Organization - the Enterprise Organization Form in the Future," in Chinese, *China Mechanical Eengineering*, Vol. 7, No. 4, 1996, pp. 15-20.

[2] H. S. Jagdev and J. Browne, "The Extended Enterprise - A Context for Manufacturing," *Production Planning & Control*, Vol. 9, No. 3, 1998, pp. 216-229.

[3] M. J. Maloni and W. C. Benton, "Supply Chain Partnerships: Opportunities for Operations Research," *European Journal of Operational Research*, Vol. 101, No. 3, 1997, pp. 419-429.

[4] L. D. Boer, E. Labro and P. Morlacchi, "A Review of Methods Supporting Supplier Selection," *European Journal of Purchasing & Supply Management*, Vol. 7, No. 2, 2001, pp. 75-89.

[5] A. Yannis, Hajidimitriou and A. C. Georgiou, "A Goal Programming Model for Partner Selection Decisions in International Joint Ventures," *European Journal of Operational Research*, Vol. 138, No. 3, 2002, pp. 649-662.

[6] D. Y. Sha and Z. H. Che, "Virtual Integration with a Multi-Criteria Partner Selection Model for the Multi-Echelon Manufacturing System," *International Journal of Advanced Manufacturing Technology*, Vol. 25, No. 7-8, 2005, pp. 793-802.

[7] L. Mikhailov, "Fuzzy Analytical Approach to Partnership Selection in Formation of Virtual Enterprises," *Omega*, Vol. 30, No. 5, 2002, pp. 393-401.

[8] D. Wang, K. L. Yung and W. H. Ip, "A Heuristic Genetic Algorithm for Subcontractor Selection in a Global Manufacturing Environment," *IEEE Transactions on SMC Part-C*, Vol. 31, No. 2, 2001, pp. 189-198.

[9] N. Q. Wu and P. Su, "Selection of Partners in Virtual Enterprise Paradigm," *Robotics and Computer-Integrated Manufacturing*, Vol. 21, No. 2, 2005, pp. 119-131.

[10] W. H. Ip, D. Wang and K. L. Yung, "A Branch and Bound Algorithm for Sub-Contractor Selection in Agile Manu-Facturing Environment," *International Journal of Production Economics*, Vol. 87, No. 2, 2004, pp. 195-205.

[11] Z. B. Zeng, Y. Li and W. X. Zhu, "Partner Selection with a Due Date Constraint in Virtual Enterprises," *Applied Mathematics and Computation*, Vol. 175, No. 2, 2006, pp. 1353-1365.

[12] W. H. Ip, M. Huang, K. L. Yung, *et al.* "Genetic Algorithm Solution for a Risk-Based Partner Selection Problem in a Virtual Enterprise," *Computers & Operations Research*, Vol. 30, No. 2, 2003, pp. 213-231.

[13] P. Brucker, A. Drexl and R. Möhring, "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods," *European Journal of Operational Research*, Vol. 112, No. 1, 1999, pp. 3-41.

[14] F. Glover, "Tabu Search: Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, 1989, pp. 190-206.

[15] F. Glover, "Tabu Search: Part II," *ORSA Journal on Computing*, Vol. 2, No. 1, 1990, pp. 4-32.

J. Software Engineering & Applications, 2010, 3, 556-560
doi:10.4236/jsea.2010.36064 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

On Some Quality Issues of Component Selection in CBSD

Jeetendra Pande[1], Raj Kishore Bisht[1], Durgesh Pant[2], Vinay Kumar Pathak[3]

[1]Department of Computer Science, Amrapali Institute of Technology & Sciences, Haldwani, India; [2]Department of Computer Science, Kumaun University, Uttarakhand, India; [3]Uttarakhand Open University, Haldwani, India.
Email: jeetendrapande@yahoo.com

Received March 11th, 2010; revised March 27th, 2010; accepted March 29th, 2010.

ABSTRACT

Component based development offers many potential benefits, viz. software reuse, reduced time-to-market, interoperability, ease of quality certification etc. However, it is not always that benefits derived from addition of components from a component repository are more than the costs involved in developing the module from scratch. This work evaluates various software quality models and suggests recommendations for enhancing software quality in COTS (*component off-the-shelf*) *based software products by designing software quality metrics that would help in managing and enhancing quality in component-based software development.*

***Keywords*:** *Component Based Software Development* (*CBSD*), *Components-off-the-Shelf* (*COTS*), *McCall's Model, Dormey's Model, Bohem's Model*

1. Introduction

Software is at the heart of most industrial systems in use today. Rapid changes in industrial methods have led to a situation where industrial products are now, more often than not, systems consisting of software and hardware. Industries in which the use of software is now essential include, among others, Automotive, Medical-Systems, Process-Control and Manufacturing. In these and others, value added to products is provided largely by the associated software.

The economics of software projects has been seen as an important sub-discipline of software engineering for some time. Projects need to be considered in a wider business environment in which the issues of cost, schedule and productivity are of considerable significance. A key area in dealing with cost and quality of software systems is the ease and benefits accruing from reuse, *i.e.* how much can be saved by using existing software components when developing new software systems? To this end, Component Based Development (CBD) is widely accepted as the approach that could best yield the long sought after benefits of software reuse, reduced time-to-market, interoperability and ease of quality certification. The basis for reuse is the reliability of the components intended for reuse and the gains achieved through their application.

It is important to demonstrate to management and funding agencies that reuse makes good business sense. For this, it is necessary to have methods to collate and furnish clear financial evidence of benefits of reuse in real projects. To achieve this, we need to define good metrics that capture these benefits and develop tools and processes to allow effective use of these metrics. Deployment of component-based applications is a common practice in the area of commercial software. To meet market requirements, new functionality is frequently added to industrial products. Introduction of new functionality is often achieved by adding components to an existing system. This is why component-based development approach is attractive to industry.

Cost and time-to-market issues are addressed by taking recourse to the rapidly emerging component-based development (CBD) approach. Adding new functionality to existing products must result in lower cost and higher quality. In CBD, software systems are built as an assembly of components already developed and prepared for integration. The many advantages of CBD approach include effective management of complexity, reduced time -to-market, increased productivity, and greater degree of consistency and a wider range of usability. Thus, quality of the final software is highly influenced by use of CBD approach. Each component will have its own quality attribute profile, but when interfaced and used together with other components, the resulting composition may show a different quality attribute profile altogether [1]. A

large range of components, which perform the same function, are available from different vendors. This makes it very difficulty for a developer to decide which component to use and which to discard, based on the quality attributes of available competing components. Quality of an individual component is important but there is no guarantee that integration of components with high quality attributes will lead to a software product with overall high quality attributes. When multiple components are integrated, it is very difficult to reason about the overall quality of the final product and developers require some metric that helps them in evaluating and choosing components in such a manner that the final product is of high quality.

Project managers usually lay stress on the importance of improving estimation accuracy and techniques to support better estimates. This paper surveys the status of quality framework in component based software development and provides recommendations for future work in improving the quality of component based software development.

2. Software Quality Models

According to IEEE 610.12 standard [2], software quality is:

- The degree to which a system, component or process meets specified requirements.
- The degree to which a system, component or process meets customer or user needs or expectation.

The following section briefly explains the various quality models for software development.

2.1 Mccall's Model (1977)

The first quality model was proposed by Jim McCall *et al* [3,4], in 1977. This model is also known as the General Electric Model. McCall split eleven different quality attributes into three groups, as shown in **Table 1**.

1) Product Operation: It includes correctness, reliability, efficiency, integrity and usability.

2) Product Revision: It includes maintainability, flexibility and testability.

3) Product Transition: It includes portability, reusability and interoperability.

With the aim of improving quality of software products, McCall included a number of quality factors in his model that gave due weight to both users' views and developers' priorities.

Table 1. McCall's model

Product Operation	Product Revision	Product Transition
• Correctness		
• Reliability	• Maintainability	• Portability
• Efficiency	• Flexibility	• Reusability
• Integrity	• Testability	• Interoperability
• Usability		

The advantage of McCall's Quality Model is the relationship created between quality characteristics and metrics. The major disadvantage is that functionality of the software product is not considered.

2.2 Boehm's Model (1978)

Barry W. Bohem gave the second quality model in 1978. Boehm's model is similar to the McCall's Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate-level characteristics and primitive characteristics – each of which contributes to the overall quality level [5,6].

High-level characteristics represent high-level requirements of actual use. These characteristics address three main questions:

1) As-is utility
2) Maintainability
3) Portability

These three primary uses had quality factors associated with them, which represented the next level of Boehm's hierarchical model, the intermediate-level characteristics. Boehm identified seven quality factors, namely:
(a) Portability (b) Reliability (c) Efficiency (d) Usability (e) Testability (f) Understandability (g) Flexibility

The lowest level structure of the characteristics hierarchy in Boehm's model is the primitive characteristics metrics hierarchy, in which these quality factors are further broken down into *Primitive constructs* that can be measured. For example, *Testability* is broken down into accessibility, communicativeness, structure and self-descriptiveness. Bohem's model lays emphasis on maintainability of the software product. It includes considerations involved in evolution of a software product with respect to the utility of the program. Boehm's quality model is based on a wide range of characteristics but does not elaborate on the methodology of measuring these characteristics.

2.3 FURPS/FURPS+ (1992)

This model, proposed by Robert Grady and Hewlett-Packard Co, decomposes characteristics into two different categories of requirements:

- Functional Requirements (F): Defined by input and expected output.
- Non-functional Requirements (NF): Usability, Reliability, Performance and Supportability.

FURPS is an acronym representing a model for classifying software quality attributes (requirements):

- Functionality–Feature set, Capabilities, Generality, and Security.
- Usability–Human factors, Aesthetics, Consistency and Documentation.

- Reliability–Frequency/severity of failure, recoverability, predictability, accuracy and mean time to failure.
- Performance–Speed, efficiency, resource consumption, throughput and response time.
- Supportability–testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, installability, localizability and portability.

The main disadvantage of the FURPS model is it does not take into account the portability of the software product. Rational Software extended this model to FURPUS+, for use in corporate requirements such as design constrains, implementation requirements, interface requirements and physical requirements.

2.4 Dromey's Model (1996)

R. Goeff Dromey proposed this model in 1996 [7,8]. According to Dormey, high level attributes like reliability and maintainability cannot be built into software. Therefore, he identified a set of properties to cover these requirements. In this model, Dormey focused on those properties of the software product that affect the quality attributes. He connected software product quality with software quality attributes (**Table 2**).

The disadvantage of this model is that efficiency of software is not considered for determining the quality of software.

2.5 ISO Model 9126 (1991)

ISO 9126 model was proposed in 1991. Unlike its predecessors, MaCall's Model and Bohem's Model, upon which this model was built, functionality of a software product is also taken into consideration in this model. It proposes a generic definition of software quality in terms of six quality factors, which further covers some sub factors:

- **Functionality**–Suitability, accuracy, interoperability, compliance and security.
- **Maintainability**–Analyzability, changeability, stability and testability.
- **Usability**–Understandability, learnability, operability and likeability.
- **Efficiency**–Time-behaviour and Resource-behaviour.
- **Portability**–Adaptability, replaceability, installability, conformance and coexistence.
- **Reliability**–Maturity, recoverability, availability and fault-tolerance.

It defines the internal and external quality characteristics of the software product. The main disadvantage of this model is how these characteristics are to be measured.

3. Component Based System Development

A component is [11,12] a language neutral, independently implemented package of software services, delivered in an encapsulated and replaceable container, accessed via one or more published interfaces. While a component may have the ability to modify a database, it should not be expected to maintain state information. A component is neither platform-constrained nor is it application-bound.

Following the success of the structured design and OO paradigms, Component-Based Software Development (CBSD) has emerged as the next revolution in software development [13].

The idea behind CBSD is integrate the reusable components to develop the final product. This strategy reduces the development effort, time-to-market and cost. Quality and reliability come as inherited features in the product developed using CBSD approach, as the components used are time tested.

4. Quality of Component based software Development

In software product development using CBSD, the programmer's role is to integrate the pre-existing software components. The various software quality models discussed above are generic models. These models do not address the quality issues for CBSD. Bertoa [14] proposed a quality model for component based software development that allows an effective assessment of COTS components. This quality model is based on ISO 9126, adapted to deal with specific characteristics of COTS components. In the proposed model, five of the characteristics of the ISO 9126 model have been retained while the sixth characteristic, namely Portability, has been removed. Additionally, suitability and analyzability sub-characteristics have also been removed and two new sub-

Table 2. Dormey's quality model

Software product	**Implementation**			
Software product quality	**Correctness**	**Internal**	**Contextual**	**Descriptive**
Software quality attributes	Functionality, reliability	Maintainability Efficiency, reliability	Maintainability Reusability, Portability, Reliability	Maintainability Reusability, Portability, Usability

characteristics, namely compatibility and complexity, have been introduced (**Table 3**). The sub-characteristics have been further classified into two different categories, namely run time and life cycle sub-characteristics. Some of the characteristics (Usability) and sub-characteristics (learnability, understandability and operability) have changed meaning in the proposed model.

However, this model fails to validate itself on any application. Adnan & Bassem [15] has also done excellent work on developing a quality model for evaluating COTS components that support a standard set of quality characteristics. This model is developed based on study done on the six generic quality models. Staring with ISO 9126 model as a base, necessary changes have been made in it to customize it to suit COTS based development. A new characteristic, viz. Stakeholders, is proposed in this new model. It depends upon who are the members of the team responsible for developing, maintaining, integrating and/or using information system based on COTS systems (**Figure 1**).

The major drawback of this model is that it fails to explain how to measure the internal quality characteristics. Ali, Gafoor & Paul [P] proposed a new approach using a set of 13 system-level metrics, which are divided into 3 categories viz. Management, Requirement and Quality (**Table 4**).

This metrics helps managers select between appropriate components from a repository of software products and aid them in deciding between using COTS components or developing new components. This model, however, does not say much about how these metric are measured/quantified. Jasmine & Vasantha [Q] propose Defect Removal Efficiency–a quality metric that provides benefits at both project and process levels. They redefined the basic definition of defect removal efficiency in terms of the phases involved in reuse-based development and proposed a systematic approach in the defect removal process. Sharma *et al.* [R] proposed a quality model based on ISO9126 that defines the characteristics and sub-characteristics of the component and proposes to add some more sub-characteristics to it, which may be relevant in the CBSD context. This model can also be used to estimate the efforts required to achieve the required value of any characteristic.

5. Untouched Quality Related Issues in Component Selection Process

In an application developed using COTS approach, various components are integrated with each other to build the whole system of desired quality. This integration process is the most crucial part of the CBSD. There may be more than one component available in the repository or more then one vendor available for the same component that performs the desired task. Each individual component from the different sources has different quality attributes. When these individual components are integrated to develop a whole system, the quality attributes of components may change and affect the quality of the overall system.

There is an absence of any kind of metrics that can help in selecting between different components from different sources with the same functionality by evaluating relevant quality parameters of the component, such as cost-avoidance, reliability, productivity etc.

There is an increasing need for metrics meant specifically for component-based software, to help foster and manage quality in component-based software development. This is because traditional software product and process metrics are neither suitable for nor sufficient in managing the complexity of software components, which is necessary for quality and productivity improvement within

Table 3. Quality model for COTS components by Bertoa

Characteristic	Functionality	Reliability	Usability	Efficiency	Maintainability
Runtime Sub-characteristics	Accuracy, Security	Recove-rability		Time Behavior, Resource Behavior	
Life Cycle Sub-characteristics	Suitability, Interoperability, Compliance, Compatibility	Maturity	Learn ability, Understand-ability, Operability, Complexity		Changeability, Testability

Table 4. System level metric for component based system

Category	*Management*	*Requirements*	*Quality*
Metric	Cost Time to market Software engineering environment System resource utilization	Requirements Conformance Requirements stability	Adaptability Complexity of interfaces and integration Integration test coverage End-to-end test coverage Fault profiles Reliability Customer satisfaction

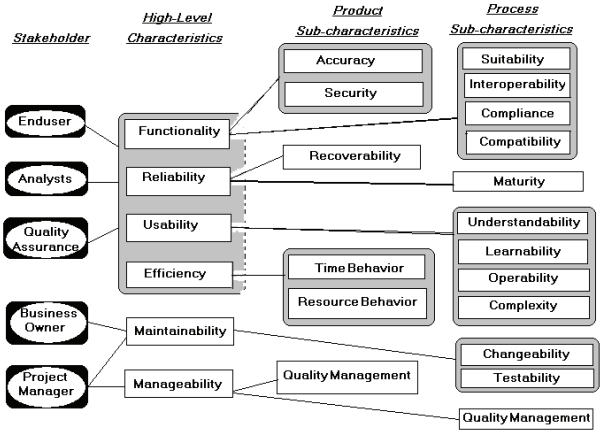


Figure 4. The new quality model for COTS-based systems

organizations adopting component-based software development.

6. Conclusions

This paper presents a survey of various generic quality models as well as quality models for COTS development. Based on this survey, some unanswered issues related to CBSD have been identified.

Future work in the development of CBSD research could include determination of quality metrics for components selection where there is more then one component available for the same task. This metric would help the developer/integrator in selecting between various components or decide upon develop the component from scratch. This metric should be easy to calculate and be feasible for practical use. The field of quality attribute determination of component-based system is extensive and more research can and should be performed in this field. This will help in increasing confidence in the use of research results, to solve problems in practical industrial settings.

REFERENCES

[1] J. A. Borretzen, "The Impact of Component-Based Development on Software Quality Attributes," http://www. toodoc.com/Software-quality-attribute-pdf.html

[2] "IEEE Standard Glossary of Software Engineering Terminologies," *IEEE Standard* 610.12-1990, 10 December 1990.

[3] J. A. McCall, P. K. Richards and G. F. Walters, "Factors in Software Quality," *Griffiths Air Force Base*, New York Rome Air Development Center Air Force Systems Command, 1977.

[4] J. A. McCall, P. K. Richards and G. F. Walters, "Factors in Software Quality," *National Technology Information Service*, Vol. 1, No. 2-3, 1977.

[5] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod and M. Merritt, "Characteristics of Software Quality," North Holland, 1978.

[6] B. W. Boehm, W. Barry, J. R. Brown and M. Lipow, "Quantitative Evaluation of Software Quality," *International Conference on Software Engineering, Proceedings of the* 2*nd International Conference on Software Engineering*, San Francisco, 1976.

[7] R. G. Dromey, "Concerning the Chimera (Software Quality)," *IEEE Software*, Vol. 13, No. 1, 1996, pp. 33-43.

[8] R. G. Dromey, "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, Vol. 21, No. 2, 1995, pp. 146-163.

[9] ISO 9126, "Information Technology-Product Quality-Part1: Quality Model," *International Standard* ISO/IEC 9126, International Standard Organization, June 2001.

[10] ISO 9126, "Information Technology-Software Product Evaluation-Quality Characteristics and Guidelines for their Use," ISO, December 1991.

[11] M. Sparling, "Is there a Component Market?" www. cbd-hq.com/articles/2000/000606ms_cmarket.asp

[12] M. Sparling, "Lesson Learned through Six Years of Component Based Software Development," *Communications of the Association for Computing Machinery*, Vol. 43, No. 10, October 2000, pp. 47-53.

[13] P. Vitharana, "Risk & Challenges of Component Based Software Development," *Communications of the Association for Computing Machinery*, Vol. 46, No. 8, August 2003, pp. 237-252.

[14] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components," *Proceedings of the* 6*th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering* (*QAOOSE*), Malaga, 2002.

[15] A. Rawashdeh and B. Matalkah, "A New Software Quality Model for Evaluating COTS Components," *Journal of Computer Science*, Vol. 2, No. 4, 2006, pp. 373-381.

J. Software Engineering & Applications, 2010, 3, 561-571
doi:10.4236/jsea.2010.36065 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

MDA (Model-Driven Architecture) as a Software Industrialization Pattern: An Approach for a Pragmatic Software Factories

Thomas Djotio Ndie[1], Claude Tangha[1], Fritz Ekwoge Ekwoge[2]

[1]Department of Computer Science, National Advanced School of Engineering, University of Yaounde 1, Yaounde, Cameroon; [2]Koossery Technology Cameroon, Douala, Cameroon.
Email: {tdjotio, ctangha, ekwogefee}@gmail.com

Received March 5th, 2010; revised April 30th, 2010; accepted April 30th, 2010.

ABSTRACT

In this paper we show that the MDA can be considered as a software industrialization pattern (*or a software factory*). *Nearly all industries today are haunted with how to reduce costs, improve quality, faster time-to-market and to maximize profits. These challenges are particularly relevant to the software industry, because it still lags behind other technology sectors as regards industrialization and the timely delivery of software products. Most software are still of poor quality, always finished after deadlines* (*most don't finish at all*), *and are very labour intensive. Here, we discuss the MDA as an approach that may help solving at the same time both problems of industrialization and ever-changing software infrastructures. We propose a MDA Engine based on a real case study in an IT services company. It is a proposal for a framework to create custom MDA tools, based on XMI, XSLT and the Visitor Pattern.*

Keywords**:** *Software Industrialization, Software Factories, MDA, MDA Engine, MDD, DSM*

1. Introduction

Software engineers are faced with the ever evolving nature of the software industry. New implementation infrastructures come and go at non negligible rates. What is "in" today may be "out" in just a few months, with little or no backward compatibility. A software factory's major concern is the industrialization of software development [1,2]. Just as a brewing industry has brewing factories that industrialize the production of beer, a software factory's main goal is the rapid production of high quality software components, at lower costs. According to Microsoft[1] "Software Factories provide a faster, less expensive, and more reliable approach to application development by significantly increasing the level of automation in application development, applying the time-tested pattern of using visual languages to enable rapid assembly and configuration of framework based components"[3].

The keyword that makes any industry factory productive is automation. But the automation process is a more complicated issue in the software industry compared to other industries. The software industry is constantly plagued by new technologies springing up very frequently. At one time everything was in C. Now most developers code in Java or any .NET language. At one time we had COM, now we equally have Web Services. The software industry has accepted the UML, which may help in describing systems, irrespective of implementation details. With UML, the model we describe will not change as often as the technology used to realize the system. The challenge then, in industrializing software components, will be the automatic transformation of UML models to concrete implementations. This is where tools like the MDA pattern come into play. When designing with UML, our level of abstraction is increased.

Software development is a complex issue. The complexity is aggravated by the fact that most developers build every application as though it is the first of its kind anywhere. We re-code the same Data Access Layers, design user interfaces from zero for each new product that comes, and we create services that are not reusable because deadlines are catching up on us. We may have gone a long way from writing code in assembly, but software development has always been regarded as an art by most

[1]Microsoft coins the term "software factory" in association with their .NET platform, but this description of software factories can be applied to other platforms.
LIRIMA (http://www-direction.inria.fr/international/lirima.html)

software developers, be they professional or casual developers. To industrialize the development of software, we need an increased level of abstraction, standardization and automation. According to the authors of the publication [1], "the key to industrialize software development is to leverage experienced developers by encapsulating their knowledge as reusable assets that others can apply. Design patterns demonstrate limited but effective knowledge reuse. The next step is to move from documentation to automation, using languages, frameworks and tools to automate more of the software life cycle". Most software has been written in the past decades to increase the productivity of workers in offices, a popular example being office automation suites. Software has been written to manipulate robots that assemble car parts in record time. For many years now, the software industry has been writing software that has helped increase the level of industrialization in other technical and non-technical sectors. It is time we seriously consider using software to industrialize the development of software. There are many ways we could automate the software development process. The most popular today being: Templates, Code Generators, MDA and DSL. This paper is separated into four parts. The first part will explore the state-of-the-art on the techniques used in industrializing software components, with particular focus on MDA. The second will focus on a specific case study of an IT Services Company: the Koossery Technology Framework (KTF). The third section will present our solution: the MDA Engine designed to serve as a guide for the creation of custom MDA tools. The last will present OptimaDev, the result of the application of the MDA Engine proposed.

2. Software Industrialization Techniques: The State-of-the-Art

2.1 Elementary Industrialization Techniques: Generators and Templates

A source code generation result in generating source code based on an ontological model such as templates. It is accomplished with a programming tool such as template processor or an Integrated Development Environment (IDE) [4,5]. The generation of source code also results from abstract models for particular application domains, particular organizations, or in simplifying the production process for computer programmers [5-7]. In the context of software engineering, the use of the term template implies many technical specifications, but it is generally identified as any processing element that can be combined with a data model and processed by a template engine to produce a result document[2]. Source code generators improve the quality of source code, provide some consistency, increase productivity and increase the level of abstraction to a certain degree. The most common form of generating code is by using templates. Though aiding in industrializing the development of software components, code generators and templates in general are too technology-oriented. It is to remove this coupling that the MDA comes into play.

2.2 Model Driven Architecture (MDA) and Domain Specific Modeling (DSM)

2.2.1 MDA (Model Driven Architecture)

The MDA is a development framework defined by OMG [8]. It "starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform" [9]. MDA addresses three main objectives which are portability, interoperability and re-usability through architectural separation of concerns. The MDA is a form of Model-Driven Development (MDD) that promises to allow the definition of machine readable application and data models which permits the long-term flexibility of implementation, integration, maintenance, testing and simulation [1,9-11]. Many basic concepts are described around it. We will just mention three of them: system, model, and viewpoint [9]. The system concept may include anything that can be a program, a single computer system, some combination of part of different systems, a federation of systems, each under separate control, a people, an enterprise or a federation of enterprises. The model of a system is a specification of that system and its environment for some purpose. "A viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system" [12-14].

The MDA specifies three viewpoints on a system: a platform independent view point, a computation independent viewpoint and a platform specific viewpoint. The '*Platform Independent*' is a quality which a model can exhibit and sets that the model is independent of the features of a platform of any particular type [14-16]. The *Computation Independent Model* (CIM) is a view of a system from that computation independent viewpoint which focuses on the environment of the system, and the requirements for the system. The details of the structure and processing of the system are hidden or are yet undetermined. The Platform Independent Model (PIM) relies from platform independent viewpoint. It focuses on the operation of a system while hiding the details necessary for a particular platform. "A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type." [14]. A very common strategy to achieve platform independence is to target a system model for a technology-

[2]http://www.nationmaster.com/encyclopedia/template-processor; (also at *Wikipedia*, accessed February 24, 2009)

neutral virtual machine [1]. The *Platform Specific Modeling* (PSM) combines the specifications in the PIM with the details that specify the way that the system uses a particular type of platform [13,14].

MDA is based on detailed platform models, for example, models expressed in UML and/or OCL, and stored in a MOF compliant repository [9,17-21]. There is the MDA pattern, by which a PIM is transformed to a PSM. This process of converting one model to one another of the same system is called "the model transformation" that forms a key part of MDA.

2.2.2 MDA vs. Domain Specific Modeling (DSM)

DSM is a software engineering methodology for designing and developing systems, most often IT systems such as computer software. It involves systematic use of a graphical Domain-Specific Language (DSL) to represent various facets of a system. DSM languages tend to require less effort and fewer low-level details to specify a given system [10,16,21,22]. MDA and DSM may appear to be the same concepts; both approaches will result in producing code automatically from a higher abstraction, thus increasing productivity. Though they both propose methods of solving the software industrialization problem based on Model-Driven Development (MDD), they both differ in approach. The principal differences can be summarized as follows: MDA promotes the use of the UML or any MOF-compliant modeling language, while DSM promotes the use of DSL for the description of a domain space. MDA is all about the use of models and their automatic transformations using a standard transformation language while on the other hand DSM is not limited to using models. The other main reason of choising MDA but not DSM is because the cost of creating and maintaining a new language not based on a standard will be too high for the compagny with no signficant added value and mostly because the nowaday IT compagny already has a lot of trained UML professionals. In the next section, we will present the Koossery Technology Framework, a real case study in an IT company.

3. Case Study: The Koossery Technology-Framework; Industrializing the Development of Koossery Technology Software Components

We exposed some basic concepts currently used to industrialize the development of software components. We have talked about the MDA Pattern, code generators, templates, and DSM. In this section, we will explore a real world example of industrializing software components in an IT services company called Koossery Technology (KT)[3].

3.1 Koossery Technology: Company Profile

KT adapts its services to the size of its customers, where ever the customer's location. Its center for technological support, Koossery Tech' Lab (KTL) does continuous technical tracking on predilected technologies, and knowledge management. KT's solutions include but are not limited to (1) J2EE and .NET Architecture and application development, (2) CORBA/C++ and CORBA/ Java Distributed Application Development, and (3) DataWareHouse, DataStage, Genio, BO/Webi. There are generally two phases involved in building .NET, Java or Corba C++/Java applications in KT. During the phase of architecture and UML design, engineers will model alternately **'use case diagrams'**; **'class diagrams'**; **'sequence diagrams'**; **'component diagrams'**; **'deployment diagrams'** to respectively identify the functional components; represent the relationship between objects; depict the dynamics of the objects and state the distributed character of the application.

During the phase of development of KT .NET applications, engineers can be allocated to the development of one or several layers. In the particular case of .NET applications we have: the Presentation Layer (be it a light client based on ASP.NET, or a rich client based on Winforms, or a smart client); the Business Logic Layer (BLL) (that can be implemented using an internal framework of the customer and the .NET framework, with target languages C# or VB.NET). The Data Access Layer (DAL) that can be implemented using the internal framework of the customer, or a commercial or open source Object/ Relational (O/R) mapping framework. The Database Management System (DBMS) familiar to KT engineers include but are not limited to: Sql Server, Oracle, and Sybase. The Inter-Layer Communications: different layers exchange data using some message queuing (like MSMQ, MQ-Series), the XML web service model in a heterogeneous environment, .NET Remoting in a homogeneous Microsoft environment, or PONOs[4] exposed as services via the Spring framework. In the following section, though mainly references will be made to the .NET framework, KT uses the same philosophy for other technologies (Java and CORBA/C++) [15,23,24].

3.2 Inhouse Framework

3.2.1 Overview

KT Lab has put in place the Koossery Technology Framework (KTF) (illustrated by the **Figure 1**), a reusable design for the development of complex distributed systems. This framework serves as a guide in building robust Service Oriented Architecture (SOA) applications, distributed components, and user interfaces.

The KTF is expressed as a set of abstract classes, ser-

[3] http://www.koossery-tech.com

[4] Plain Old .NET Objects

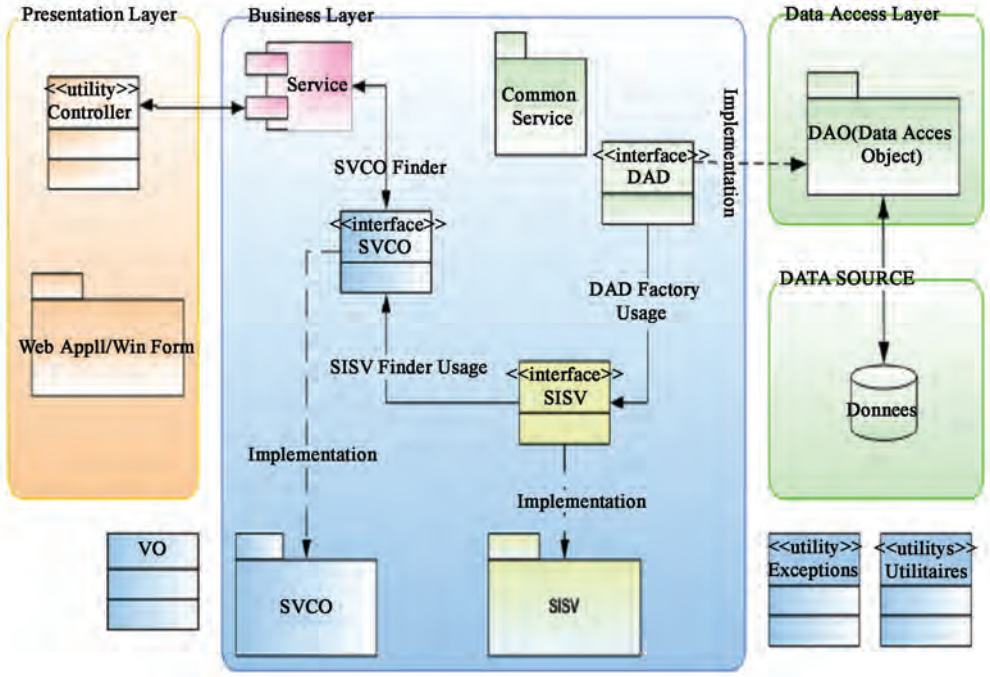


Figure 1. Koossery technology framework for n-tier applications[5]

vice locators, configuration files, and the way all instances of these collaborate for a specific type of software. It includes many support programs, code libraries, and other software to help develop and glue together the-different components of a software project, using popular patterns like MVC (Model View Controller), DAO (Data Access Object), DTO (Data Transfert Object), IoC/DI (Inversion of Control/Depedency Injection), Service Locators and other design patterns. It also uses popular tier software utilities like log4j/log4net[6], Ibatis[7], Hibernate[8], Spring, Struts, Maverick ... Various parts of the KTF are exposed though an API. With the KTF in place, developers spend more time concentrating on the business-specific problem at hand than on the plumbing code behind it. Also the KTF limits the choices during development to a certain extent, so it increases productivity, specifically in big and complex systems.

3.2.2 Development of .NET Server Components

Now we will describe how server components are developed. The development of a .NET server component, for example, is divided into 3 fundamental layers: DAL, SISV (Simple Service), SVCO (Service Controller). In the DAL, data can be stored in a Remote DBMS (RDBMS) or any other medium. To access data, the DAL uses a framework for O/R mapping, the two most popular used being iBATIS and NHybernate. Sometimes ADO.NET[9] is used directly, but it is used in a similar manner as an O/R mapping architecture. The DAL also possesses a service locator called DAOController which encapsulates the search for any DAO implementation.

The SISV is the layer for simple services. This layer manipulates the DAL directly, using the DAOController service locator to find the required DAO implementation. It is meant to be a stable layer since it is constituted of very simple functionalities which are just a combination of calls to the DAL. The SISV also has a service locator called SISVFinder to encapsulate the search for any SISV implementation.

The SVCO is the layer for composed services. This layer is constituted of very high level services which are obtained as a combination of services found in the SISV layer, using the SISVFinder to search the required SISV service. The SVCO layer should never access the DAL directly. It also has a SVCOFinder to encapsulate the search for its SVCO services.

The KTF applies dependency injection using the IoC pattern. The SISVFinder dependency of the SVCO is passed as a constructor dependency in an SVCO implementation, and the DAOController dependency of the SISV is passed as a constructor dependency in a SISV implementation. Integration is usually done in an external file using the Spring Application Framework[10].

The strategy most applied when developing a server component at KT is to separate the server component into two distinct software units, the CORE and the

[5]VO = Value Object/

[6]http://logging.apache.org/log4net/

[7]http://ibatis.apache.org/

[8]http://www.hibernate.org/343.html

[9]http://msdn2.microsoft.com/en-us/data/default.aspx

[10]http://www.springframework.org/

BACKEND. The CORE includes all different interfaces, exceptions and Data Transfer Objects (DTOs). The BACKEND includes all concrete implementations of the CORE. The CORE is the heart of the server component. It contains the CORE_CONTRACT and the CORE_BACKEND. CORE_CONTRACT includes all interfaces of the services offered by the server component to its clients. The CORE_BACKEND comprises all interfaces of the DAL and the SISV. The BACKEND contains the following packages: the DAO, SISV and SVCO. The DAO package contains all implementations of the DAL interfaces found in the CORE_BACKEND, the SISV package contains all implementations of the SISV interfaces found in the CORE_BACKEND, the SVCO package contains all implementations of SVCO interfaces found in the CORE_CONTRACT.

The framework proposes a method of realizing the concrete implementations of the DAL, SISV and SVCO. The services of the server component may be exposed locally using assemblies, as web services using Spring, or by using .NET remoting using Spring. Logging is usually accomplished via a logging framework e.g. Log4Net. All these layers are organized into separate Visual Studio projects that generate 6 principal artifacts: the CORE_CONTRACT, CORE_BACKEND, SVCO, SISV and DAO assemblies plus a set of configuration files, as summarized in **Figure 2**.

3.3 The Need to Industrialize [1]

KT, like most IT services companies, realized that with their actual methods close to handicraft, a lot of money was being lost when everything was done manually. The first approach to reduce the amount of craft was to capitalize all of the company's experience in a framework, the KTF; and organize methods for the realization of aproject. However, this first approach just permitted the engineers to have a working guide to the development of applications, with still a major part of the application done manually. There was thus a need for a second approach that will reduce the amount of manual input, and which from the modeling phase generates an application respecting their standards [15,23,24].

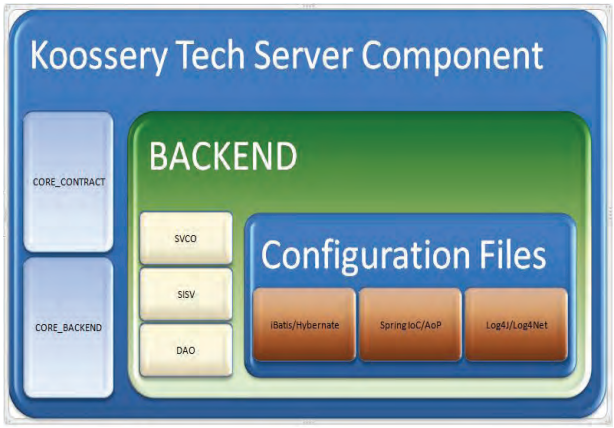


Figure 2. Koossery technology sample server component project structure [15]

We propose the MDA, amongst other software industrialization techniques, because it is the closest approach to really fulfilling this form of application generation. We expect the MDA approach from the UML model of an application, to generate all the application, all technical services, all configuration files, all CRUD (Create, Read, Update and Delete) functionalities so that the developer in the end will only have to complete with specific algorithms for only the most complicated business logic.

3.4 Preparations for the MDA: Identification of PIM, PSM and CM in KTF

To apply the MDA pattern to the KTF, let us identify what we will use as PIM, PSM and CM. Let us also define how our models will be marked, so as to perform automatic transformations from a higher level of abstraction (the PIM) to a lower level of abstraction (the CM).

- The choice of PIM has been natural: UML. Various enterprise UML tools are already used in KT including Rational Rose from IBM, Poseidon for UML from GentleWare and Enterprise Architect from Sparx Systems.
- The choice of PSMs has been limited to the various technologies used in KT at the present moment. Webservices or .NET remoting for exposing services, Hybernate or iBATIS for O/R mapping, Spring for Dependency Injection and exposure of PONOs and POJOs[11] as services, Log4J/Log4Net for logging etc.
- The CM can either be in Java, C# or CORBA/C++ for source code, and XML for configuration files.

Now that we have identified the various models, we have to perform automatic transformations from the abstract models to the code models. The strategy we have used to aid in this automatic transformation is by using marks, and the possible use of OCL to produce models of higher quality. So how could we mark UML diagrams for the KTF?

There are some standards respected in all modeling done in KT. All DAO, SISV, and SVCO interfaces are prefixed with "I" and suffixed with DAO, SISV and SVCO respectively e.g. IUserDAO, IUserSISV and IUserSVCO. All DAO, SISV and SVCO concrete implementations are suffixed with DAOImpl, SISVImpl and SVCOImpl respectively e.g. UserDAOImpl, User-SISVImpl, and UserSVCOImpl. All DTOs are suffixed with

[11] Plain Old Java Objects

DTO, e.g., UserDTO and all relational database tables are stereotyped with *entity*.

With this level of detail in the UML models, a choice was made to use UML classifier suffixes as a means of marking our models. These marks help us perform the automatic transformations from PIM (the UML model) to CM (the resultant code). e.g. a classifier that inherits from no other classifier and marked with the DAOImpl suffix in the PIM will, in the resultant CM (C# or Java code), inherits from the AbstractDataAccessObject abstract class defined in the KTF. Another method of marking will be the use of stereotypes; e.g. classifiers marked with the entity stereotype will be transformed into Data Definition Language (DDL) statements in Structured Query Language (SQL). Finally, we use the OCL to add some elements of business logic to the models, like saying "*an employee's age must be between* 18 *and* 65".

Now that we have identified the PIM, PSM and CM in the KTF and stipulated how the models will be marked, the framework is ready for MDA Transformations. The next step is using an MDA tool that performs the automatic transformations between models.

4. Designing a Lightweight MDA Engine

4.1 Motivation

So we have the problem of applying the MDA pattern in a company. Creating some custom software that will perform specific and not general automatic transformations from PIM to CMs will be more beneficial to the company on one hand, but may cost the company more time and money developing such software on the other hand. The custom software can be tuned to extract maximum benefit from the MDA pattern. It is to help create custom MDA tools that the idea of designing a lightweight MDA Engine sprung up. This MDA Engine will serve as a guide for the creation of custom MDA tools, which can be tuned for the specific enterprise, consistent, and uses as much as possible standard file formats, thus increasing the Return On Investment (ROI) for the creation of the custom MDA tool. It was designed to be lightweight so that the custom MDA tool developer will be able to start his/her project very rapidly.

4.2 Custom MDA Tool

4.2.1 Pragmatic Approach

To be pragmatic, we cannot possibly model every aspect of the business logic in UML. Maybe with the arrival of Executable UML this will be possible. But why should everything be modeled in UML? An argument in favor of modeling everything in UML is the ability to generate full working application only from the UML model. Arguments against relate the complexity and heaviness of these models. A pragmatic approach will involve some hybrid of UML and a 3GL. When computers were invented, everyone thought that paper usage in offices would reduce. Just the opposite is complete taking place today, with computers printing out more and more paper every day. Likewise, will the MDA eliminate the need for programmers? Not necessarily.

With present and near future technology, some parts of a software application will always require low-level coding. It just doesn't have to be a lot of very low-level coding. It is only by creating custom software that we can respect these criteria for each particular enterprise. The main reason why software developers sometimes react very critical on MDA is that MDA automates the heart of their profession. The generated code is not simply like how they code. This has prompted the development of a lightweight MDA Engine, from which developers can produce generated code from abstract models, their way.

4.2.2 The Broad View

The mere fact that we are trying to automatically transform a visual language like UML to some code may sound like a daunting task. Do we have to write Computer-aided design (CAD)-like software that understands shapes? That would be a very difficult thing to do. What would help is if we had an electronic format that represents these visual models, and permit us to access parts of these models. There exists such an electronic format: the XMI (ML Metadata Interchange) format.

***XMI*:** Now that our visual models can be transformed to electronic formats, we have to be able to perform MDA transformations on these models. Is there any standard for the transformation of XMI files? The QVT[12] exists, but does not suit our case since it cannot generate source code and no concrete implementation exists at the time of writing. So there is yet no implemented standard for transforming XMI files to any other type, but the problem can be solved indirectly. Since XMI files are XML files, we can address the problem by looking for a standard for transforming XML files. Fortunately, the XML format already has a standard for transformations from XML to any desired format, the XSLT (Extensible Stylesheet Language Transformations). Since XMI is an XML file, we can thus define a mapping between MDA transformations and XSL Transformations.

***ESLT*:** So to conclude, the proposed approach is simply to export UML models (our PIM) to the XMI format, then perform XSLT transformations to obtain specific code, configuration files or other text files(the CM). No need to depend on a proprietary format, or tool.

Now that we have a view of how transformations will

[12]http://smartqvt.elibel.tm.fr/, http://en.wikipedia.org/wiki/QVT; http://umtqvt.sourceforge.net/; http://www.omg.org/docs/ptc/07-07-07.pdf (accessed 24/103/2009).

be performed, we can build an MDA Engine that will help us do these transformations.

4.2.3 Architecture of the Lightweight MDA Engine

Since we have an XMI document that has to be transformed to various code models, it seemed natural to use the Visitor Design Pattern**,** where each visitor will visit the XMI document containing our UML model and generate corresponding code. We may have a visitor for the generation of each specific interface, configuration file, concrete class implementation or even other XMI files. Sometimes the order in which the visitors visit is important. The MDA Engine has to take care of that. The main concepts are illustrated by the **Figure 3**.

Figure 3. MDA engine main concepts

The **Figure 4** shown below illustrates the UML Class

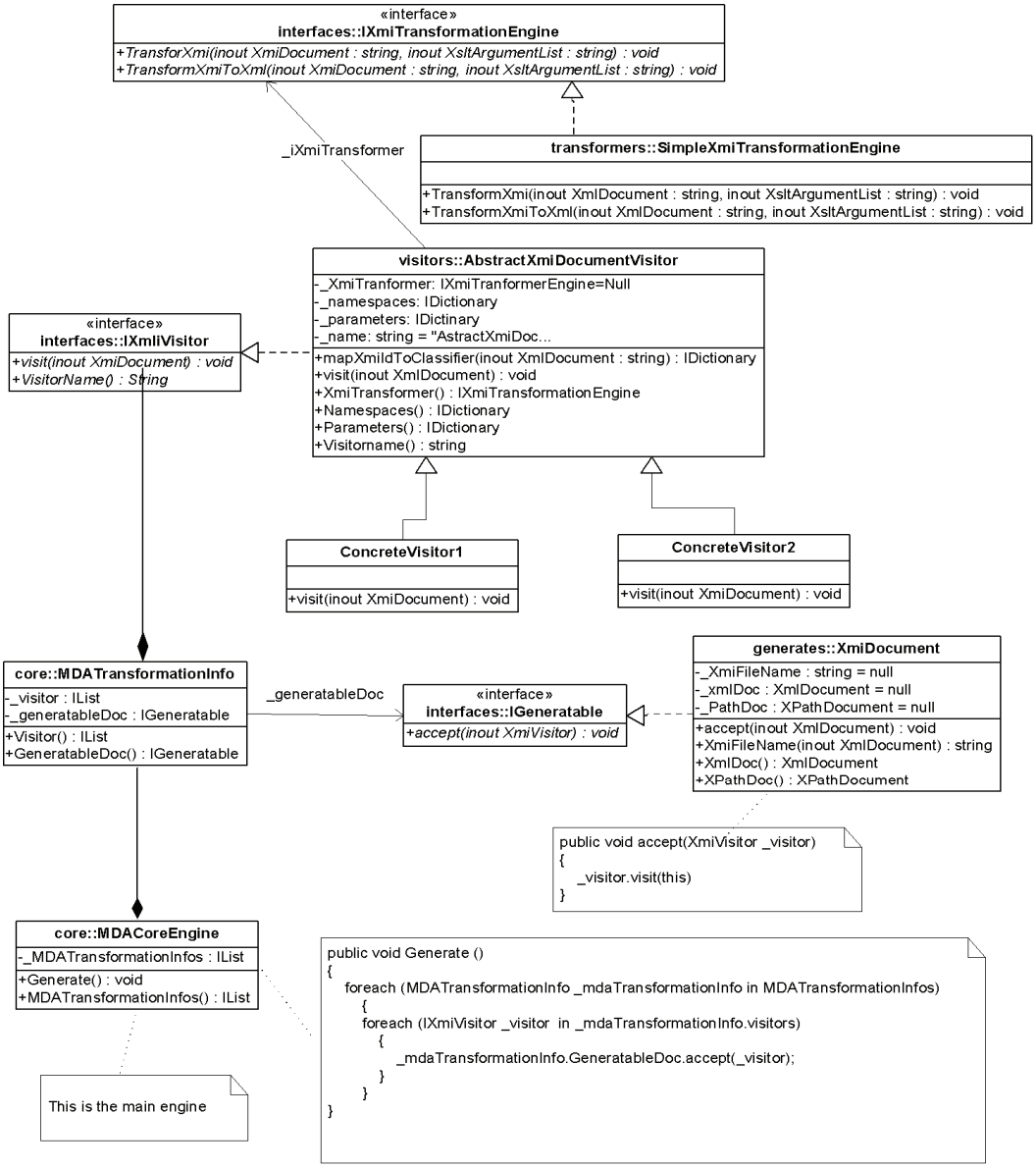


Figure 4. MDA engine

Diagram of the MDA Engine.

The participants of the UML Class Diagram are:

- IGeneratable: Interface that represents any generatable document. All generatable documents accept a visitor. In this particular case, all generatable documents accept an IXMIVisitor.
- IXmiVisitor: Interface that represents a visitor for an XMI document. Every visitor has a name. The visitor's operation is defined in the visit method.
- IXmiTransformationEngine: Interface that defines a common contract for all XSLT processors. There are two methods. One that transforms an XMI document to a text document, and another that specifically transforms an XMI document to another XML document.
- SimpleXmiTransformationEngine: A concrete XSLT processor that implements IXmiTransformationEngine.
- MDATransformationInfo: A data structure that holds a list of visitors for a particular generatable document.
- MDACoreEngine: This contains a list of MDATransformationInfos. It has a Generate method that calls each visitor sequentially as defined in each MDATransformationInfo object.
- XmiDocument: Data structure that represents a "generatable" XMI document. It has two properties that expose the DOM representation of the XMI file. One that is editable XmlDoc, and another that is not editable (but faster) XPathDoc. As soon as an XmiDocument accepts an IXmiVisitor, it calls this visitor's visit method on itself.
- AbstractXmiDocumentVisitor: All visitors can derive from this base class to facilitate their work. It has a reference to an IXmiTransformerEngine for XSLT processing, a dictionary of namespaces used in the XSLT files, a dictionary of Parameters that can be passed to the XSLT processor. It also has a utility function that maps an XMI ID to a classifier name called mapXmiToClassifier. Each visitor may use the XSLT processor for MDA Transformations. The AbstractXmiDocumentVisitor may also possess a log object based on a logging framework like log4J/log4Net for logging purposes.
- ConcreteVisitor1 and ConcreteVisitor2: These are concrete transformations to be performed in the IXmiDocument. Each operation is defined in the concrete class's visit method

This engine is distributed as a third party library, in the form of a .NET assembly or Java jar file. All the developer has to do now is to write visitors based on the AbstractXmiDocumentVisitor, and define a set of XSLT templates. The MDACoreEngine is then filled with a list of MDATransformationInfo objects, which in turn are filled with visitors either programmatically or using dependency injection (one may use the Spring IoC Framework for depency injection). To perform transformations, simply call the *Generate*() method of the MDACoreEngine object. The next section presents Optimadev, an application usage/case of the MDA Engine.

5. OptimaDev: A Prototype for MDA Engine

Creating a prototype for the MDA Engine will consist of creating an incomplete model of the future full-featured MDA Engine, which can be used to let the users have a first idea of the completed program. This prototype is called OptimaDev.

5.1 Preliminary Specification

The preliminary specification for OptimaDev was to create, automatically from the UML model, a set of artifacts. These included, but are not limited to artifacts needed for the CORE_CONTRACT, CORE_BACKEND, DAO, SISV and SVCO components. Some of the artifacts to be generated are displayed in **Figure 5** below.

5.2 Analysis

In order to fulfill the preliminary specifications, OptimaDev was designed as a custom MDA Engine that will perform MDA transformations for us, creating artifacts that respect the KTF. The choice of a custom MDA tool was taken because of the incapability of current MDA tools [13] in generating artifacts that respect the KTF. Based on the architecture in Subsection 5.1 and illustrate by the **Figure 5**, we created 9 visitors. All visitors inherit from the AbstractXMIDocumentVisitor base class as found in 4.2.3, and are listed in **Table 1**. OptimaDev was also furnished with a Graphical User Interface to ease the transformation process (see **Figure 6**).

Figure 5. OptimaDev: Preliminary specifications

5.3 Implementing the Prototype

The user interface permits the user to input his XMI file representing his model, and choose an output directory where OptimaDev will serialize results. The XMI file is obtained by exporting from a UML design tool. After generation, OptimaDev provides visual feedback on the status of the generation, like the number of files successfully generated. As shown in **Figure 6**, OptimaDev is cautious enough to detect if there was an error during transformations, and robust enough to continue functioning after having signaled the error. Finally, the generated artifacts are serialized in directory structures that closely resemble what is expected in a KT project.

5.4 Case Study or Application of OptimaDev: SoNetSec

5.4.1 Context

SoNetSec is a Real Estate Servicing company located in Cameroon. To ameliorate its services, SoNetSec has decided to have at its disposal an Information System in the form of a family of software that will guarantee at the same time its agility and its global competivity. Without entering into the details of the functional specifications, we will briefly list some of its non functional specifications:

- The principal application, which will serve to showcast, promote and ecommerce its products and services will be a transactional web based Internet application, capable of supporting high visiting rates.
- Some applications may have to be implemented using rich clients.
- The application will be conceived and implemented as a set of autonomous services.
- Scalability, security, robustness, response time, maintenance issues have to be considered in the conception and implementation of the different applications.

Fortunately enough, the KTF already facilitates the creation of software applications and software components with such non functional specifications. Some portions of the Information System to be realized were given to two software engineers and a senior software architect. What is interesting to recall is that within KT, most engineers communicate via UML models.

In the course of prototyping, a study of how the engineers and the senior architect modeled the Information System was done, including a study of how these models were implemented. It was the job of the prototype MDA Engine, OptimaDev, to automatically produce source code respecting the KTF from these visual representations. The feedback of the engineers, obtained through agile methods, was indispensable in perfecting the 9 visitors listed in **Table 1** below. This helped create source code and configuration files that were to be compared with what was done manually.

5.4.2 Preliminary Results and Benefits

The autogenerated code improves on quality, consistency, productivity and abstraction compared to manual code

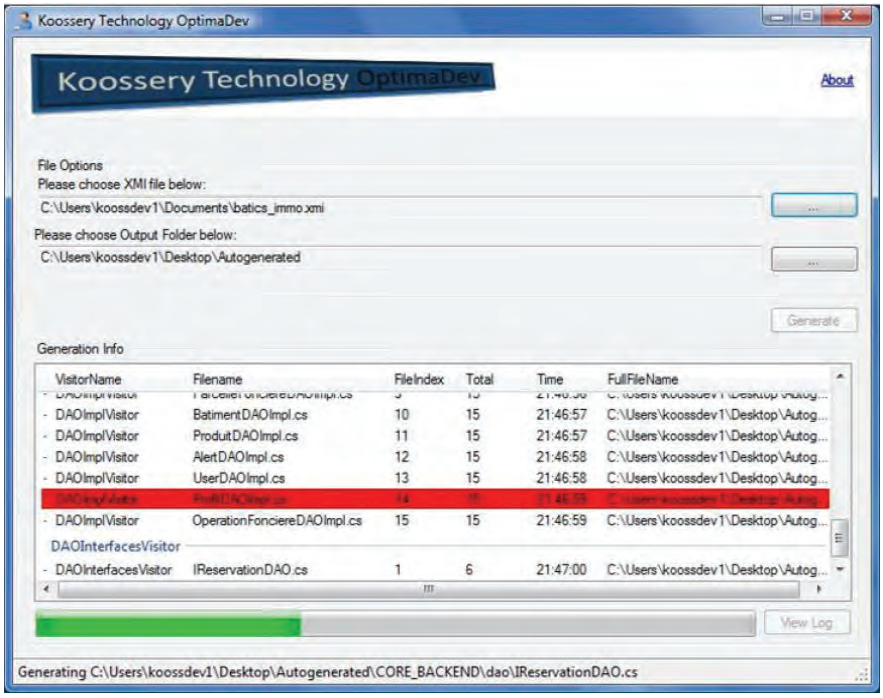


Figure 6. OptimaDev: Automatic error detection

Table 1. Visitors for the OptimaDev (MDA transformation visitors used in the OptimaDev)

	Visitor	Description
1	DTOVisitor	From the PIM, it extracts all DTOs from class diagrams and serializes each DTO in a separate file in the choice programming language. This visitor is meant to generate 100% of the code.
2	DAOImplVisitor	From the PIM, it extracts all DAO Implementations from class diagrams and serializes each implementation in a separate file in the choice programming language. This visitor is meant to generate stubs for the implementations with support of very common CRUD method signatures. This accounts for 95% of the code.
3	DAOInterfaces-Visitor	From the PIM, it extracts all DAO Interfaces from class diagrams and serializes each implementation in a separate file in the choice programming language. This visitor is meant to generate 100% of the code.
4	SISVImplVisitor	From the PIM, it extracts all SISV Implementations from class diagrams and serializes each implementation in a separate file in the choice programming language. This visitor is meant to generate stubs, accounting for 75% of code.
5	SISVInterfaces-Visitor	Same as DAOInterfacesVisitor, but for SISV interfaces. This visitor is meant to generate 100% of the code.
6	SVCOImplVisitor	Same as SISVImplVisitor, but for SVCO implementations. This visitor is meant to generate 75% of the code.
7	SVCOInterfaces-Visitor	Same as DAOInterfacesVisitor, but for SVCO interfaces. This visitor is meant to generate 100% of the code.
8	SpringVisitors	From the PIM, it extracts all DAO, SISV and SVCO Implementations from class diagrams and setups spring configuration files for Inversion of Control. It also sets up the service locators of each of these layers. This visitor is meant to generate 100% of the code.
9	IbatisRequestsVisitor	From the PIM, it extracts all DAO and DTO Implementations from class diagrams and creates iBatis request configuration files which are each serialized in a separate file in the choice programming language. This visitor is meant to generate stubs for the implementations with support of very common CRUD method signatures. This accounts for 95% of the code.

6. Conclusions and Perspectives

approaches which simply provides flexibility and control. Equally, Knowledge Base was a great benefit. The process of adopting the MDA pattern has forced the extraction of the best of individual KT experts into the MDA Engine, OptimaDev.

The approach we adopted helps us create pragmatic software factories that boost the industrialization of software development. We have particularly emphasized on the MDA pattern as a form of MDD and as a software factory. The best approach will be to create some custom tool that adapts the MDA pattern for each company. The MDA Engine is a proposal for a framework to create custom MDA tools, based on XMI, XSLT and the Visitor Pattern. It serves as a starter kit to help develop MDA tools that are tuned to a company's business logic, or software development strategies.

We have also described the use of this MDA Engine to build a prototype custom MDA tool (internal code name: OptimaDev) for Koossery Technology (KT). For OptimaDev, the KT MDA tool prototype, we realized some visitors based on the MDA Engine proposed. These visitors are designed to generate code for the development of a server side component following the KT Framework. Together with the addition of other visitors for the presentation layer, the support of the Object Constraint Language (OCL) especially for visitors targeting the business layer, we are very confident that with time the custom MDA tool's roadmap will be from code generator, to software component generator, and finally to a complete software application generator.

For the perspective point of view, there are many things we can add to this basic MDA Engine. Let's mention some here.

- Multi Agent System (MAS), where we will have intelligent agents instead of visitors that perform transformations.
- Expert System (ES), where the MDA Engine may instead be conceived as an inference engine with a set of inference rules that transform models. This permits the transformation process to be more declarative than imperative (see [7]).
- OCL Support, to be able to produce models of even higher quality.
- xUML or Executable UML support, to describe the dynamics of a domain [20].
- Round-trip engineering, to synchronize changes between model and code.
- AI/Fuzzy Logic: because the model itself can have some errors which some Artificial Intelligence or Fuzzy Logic can help.
- And others e.g. Velocity template language support, because it closely resembles the output code, unlike

XSLT.

7. Acknowledgements

Special thanks go to Professor Jean Claude Derniame of Institut Polytechnique de Loraine at Nancy France, to have reviewed this paper, and also to Koossery Technology Cameroon to have provided us with a real test environment.

REFERENCES

[1] J. Greenfield and K. Short, "Moving to Software Factories." http://www.softwarefactories.com/ScreenShots/MS-WP-04.pdf

[2] DoFactory.com, "Design Patterns in C#, VB.NET WPF, WCF, LINQ, PATTERNS," *Data & Object Factory™*, http://www.dofactory.com/Patterns/Patterns.aspx

[3] Microsoft, "Domain-Specigic Language Tools." http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx

[4] P. V. Hoof, "Code-Gen–about and technical documentation." http://forgeftp.novell.com//codegen/docs/Technical%20documentation/codegen_doc.pdf

[5] AndroMDA, "Extensible Code Generator." http://www.andromda.org

[6] Code Generation Network, "Code Generation Network." http://www.codegeneration.net/tiki-index.php?page=ModelsIntroduction

[7] ExpertCoder, "Code Generation Libraries for .NET, Mono and dotGNU." http://expertcoder.sourceforge.net/en/index.html

[8] A. Kleppe, J. Warmer and W. Bast, "MDA Explained: The Practice and Promise of the Model Driven Architecture." Addison Wesley, Massachusetts, 2003.

[9] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1." http://www.omg.org/docs/omg/03-06-01.pdf

[10] A. Kleppe, J. Warmer and W. Bast, "MDA Explained: The Model-Driven Architecture: Practice and Promise," Addison Wesley Professional, Massachusetts.

[11] J. S. Mellor, S. Kendall, A. Uhl and D. Weise, "MDA Distilled: Principles of Model-Driven Architecture." Addison Wesley Professional, Massachusetts, 2003.

[12] S. Sewall, "Executive Justification for Adopting Model Driven Architecture (MDA)." http://www.omg.org/mda/mdafiles/11-03_Sewall_MDA_paper.pdf

[13] Equipe SoftFluent, "Livre Blanc CodeFluent L'approche de Génie Logiciel de SoftFluent." http://www.softfluent.com/codefluent_home_en.aspx

[14] "What is MDA?" http://www.modelbased.net/mdi/mda/mda. html

[15] E. E. Fritz, "Pragmatic Software Factories: Industrialization of the Development of Software," Masters of Thesis of the National Advanced School of Engineering, University of Yaounde 1, 2007.

[16] J. M. Embe, "MDA: Applications de la Trans- formation des Modèles à la Génération d'Applications Trois Tiers," Ecole Nationale Supérieure Polytechnique, Université de Yaoundé 1, 2005.

[17] D. Pilone and N. Pitman, "UML 2.0 in a Nutshell," O'Reilly, 2005.

[18] S. Mellor and M. Balcer, "Executable UML: A Foundation for Model-Driven Architecture," Addison Wesley Prossional, 2002.

[19] J. Warmer and A. Kleppe, "Object Constraint Language, Getting Your Models Ready for MDA," Addison Wesley Professional, Massachusetts, 2003.

[20] 20nUML. http://numl.sourceforge.net/index.php/MainPage

[21] openArchitectureWare.organization, "Official Open ArchitectureWare." http://www. openarchitectureware.org

[22] S. Cook, J. Gareth, S. Kent and A. Cameron, "Domain-Specific Development with Visual Studio DSL Tools," Addison Wesley Professional, Massachusetts, 2007.

[23] M. Yacoubou, "Développement Industrialisé d'Applications n-tiers: Partie FrontEnd," Master's Thesis of the National Advanced School of Engineering, University of Yaounde 1, 2007.

[24] P. Djomga, "Développement Industrialisé d'Applications n-tiers: Partie BackEnd," Master's Thesis of the National Advanced School of Engineering, University of Yaounde 1, 2007.

J. Software Engineering & Applications, **2010, 3, 572-579**
doi:10.4236/jsea.2010.36066 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

Object-Oriented Finite Element Analysis of Metal Working Processes

Surendra Kumar

CSIR Centre for Mathematical Modelling and Computer Simulation, Council of Scientific and Industrial Research, Bangalore, India.
Email: surendra@cmmacs.ernet.in

Received March 10th, 2010; revised March 30th, 2010; accepted April 2nd, 2010.

ABSTRACT

Recently an object-oriented approach has been applied in the fields of finite element analysis with a view to treating the various complexities within these. It has been demonstrated that finite element software designed using an object-oriented approach can be significantly more robust than traditional codes. This paper describes a special kind of implementation of object-oriented programming which is rather hybrid in nature, in the development of a finite element code for engineering analysis of metal working problems using C++, and discusses the advantages of this approach.

Keywords*: *Finite Element Method, Data Abstraction Techniques, Object-Oriented Programming, C++ Programming Language, Metal Working

1. Introduction

The finite element method (FEM) has been developed and applied extensively in various fields of engineering. It is a purely computer-oriented numerical tool and requires an extensive amount of programming effort. Major concerns in the development of FEM systems are placed on the computational efficiency of numerical algorithms. Traditionally, procedure oriented programming techniques have been widely used and procedural programming languages such as FORTRAN have been strongly supported. Although the procedural approach has been proven effective in treating algorithmic complexity, this approach has intricate control strategies and internal data representation and does not address design and quality issues of the overall program. As a result, software developed using this approach is likely to face difficulties in its maintenance and extensions.

Recently, several investigations have been performed in applying the concept of object-oriented (O-O) methodology in the field of FEM [1-13]. It has been verified that object-oriented programming can provide strong support to desirable features of FEM systems such as reusability, extensibility, easy maintenance, etc. These benefits are achieved by well defined mechanisms of modular design and reusability of code. The object-oriented approach attempts to manage the complexity inherent in real-world problems by abstracting out knowledge and encapsulating it within objects. The various features of this approach consist of a class mechanism with inheritance and virtual function call mechanism, in addition to the facilities supporting data abstraction techniques and polymorphism. A detailed account of object-oriented programming can be found in several computer journals, language user guides and other literatures [14-16].

Mackerle [17] presents a list of published papers dealing with object-oriented programming (OOP) applied to FEM and BEM. In one of earlier investigations, Zimmermann *et al*. [2] discussed the concept of OOP as applied to the implementation of the finite element method. Huang *et al*. [4] have proposed a knowledge base system in which an object-oriented analysis in the FEM domain is carried out by means of introducing entity analysis concepts. Zimmermann *et al*. [5] discussed the key features of an integrated environment of finite element related technique which includes an object-oriented graphic interactive environment and object-oriented operators for symbolic mathematical derivations. Archer *et al*. [6] demonstrated an object-oriented architecture for finite element analysis based on a flexible and extendible set of objects that facilitate finite element modeling and analysis. Yu and Kumar [7] presented an object-oriented framework for implementing finite element method and explored ways to exploit the commonalities between various types of elements, loads, constraints and solvers so that duplication is reduced and software reuse is improved. Mackie [8] described a study into the object-oriented implementation of distributed finite element analysis on desktop computers

using the .NET framework. Heng and Mackie [9] proposed the use of software design patterns to capture best practices in O-O finite element programming.

Some research papers discuss the object-oriented techniques in the context of specific problems and also depict changes in the overall design or specific aspects in the design. Tabatabai [10] suggested an O-O finite element environment for reinforcement dimensioning of two- and three-dimensional concrete panel structures. Pantale [11] presented benefits of using an OOP approach in comparison with traditional programming language approaches in the analysis of inelastic deformations and impact processes. Kromer *et al*. [12] described an approach to the design and implementation of a multibody systems analysis code using an object-oriented architecture. Franco *et al*. [13] discussed the aspects of the OOP used to develop a Finite Element technique for limit analysis of axisymmetrical pressure vessels.

Although, the basic concepts of the design of an O-O finite element program are same, varying degrees of object orientation – even procedural design – can be accomplished using an O-O language depending upon a variety of factors including the software requirements, language features, executing environment and developer's methodology and viewpoint. Furthermore, one of the important challenges in developing O-O finite element codes is to find the balance between good abstractions and high computational efficiency, since the data abstraction and the associated polymorphism results in loss of numerical performance because it requires late (dynamic) binding. Compiler optimization and flow of execution in a process (computer program) are also more amenable to procedure-oriented code. Inspite of tremendous advancements in computer hardware capability, the numerical efficiency of finite element codes remains an important factor since demand for non-linearities, mesh refinement, coupled analysis and other complexities in the finite element solution are also growing. Unfortunately in early investigations, O-O philosophy has been considered as a systematic obligation and even numerical tools such as Gauss integration schemes have been abstracted out as objects. It is obvious that this extreme inclination towards data abstraction for each conceptual entity can lead to a serious loss of performance and difficulty in maintenance.

In the present investigation, a particular kind of object-oriented implementation has been applied in the design of FEM system for metal working analysis. The application of object-oriented programming to metal working analysis is not discussed in literature, although an important aspect of object-orientation is that it supports very general finite element codes, not tied to any particular application area. The metal working problems are multi-structure problems involving master (die) and slave (workpiece) structures and also include a large amount of non-linearities both in the element formulation and the solution process. The present design is rather hybrid in nature comprising of both object-orientation techniques and modular programming practices. This has been accomplished by abstracting out necessary real world objects in the finite element domain and implementing all numerical calculations and tools as member functions inside classes of these objects, wherever appropriate. This approach will apparently result in a good balance of benefits brought out by the two approaches. The present architecture also consists of suitable interface classes between primitive FEM classes (material, node, element, etc.) and the problem domain at different levels so that these primitive objects do not directly interact with the problem domain but through these interfaces. C++ is used in the development of the program which has several features to support object-oriented programming and can provide high computing efficiency because of its compatibility with C [18].

The paper is organized as follows. First, the key concepts of object-oriented programming are briefly outlined in the context of FEM and different features of present object-oriented framework are discussed. The issues involved in general object-oriented design and the benefits obtained by specific aspects of present implementation are also discussed. Next, present object-oriented system is applied in solving a general example problem of metal working and roles of different classes and interactions among them are explained.

2. Object-Oriented Design of FEM System

The most desirable types of general purpose finite element codes are those that are designed for comprehension, modification and updating. These desirable objectives can be easily met if the program is designed using object-oriented techniques. The FEM is by its nature a modular numerical tool. Object-oriented programming enables full advantage to be taken of this modularity. It reduces the scope for bugs by encouraging clearer thinking about the program design and allows easier incorporation of new types of element, solution techniques and other facilities as they become available.

While performing an object-oriented design, the first task is to identify classes of objects that will model the application domain. Fortunately, it is not difficult to identify the objects in the FEM domain, because several entities such as element types, material properties, nodal points, elements, etc. can be extracted from the fundamental concepts of FEM. Several solid model entities such as points, lines, surfaces, volumes, etc. also can be directly identified as objects. However, in the FEM domain, there are a large amount of problem-solving activities which are difficult to be directly identified as objects. Yet, their use and implementation may differ substantially from those in conventional codes. Some mathematical variables such as vector, matrix, etc. can also be designed

as objects so as to hide their implementing details.

The present framework consists of several basic classes such as **ElemType**, **Material**, **Node**, **Element**, etc. which are traditional classes used for the representation of finite elements. Several specific classes are derived from these abstract classes. For example, a class **Elem2DMfQ8** defining two dimensional eight noded metal working quadrilateral elements is virtually inherited from multiple base classes each having own set of base classes, as depicted in **Figure 1**.

In most of the earlier investigations, these primitive objects directly interact with the problem domain. However, it can be revealed from the real world concepts that in the FEM domain, some super objects can be identified which are either aggregates of the same objects or a superset of different objects. Further, new formalisms and new solution strategies evolve in a regular manner in the FEM field. The implementation of these new techniques in the main class or sub-class of the problem domain must not lead to frequent revisit of classes at lower levels and must not demand for redefinition or major changes in the software architecture. In order to achieve this to a possible extent, we create an interface between the primitive objects and the problem domain by defining classes such as **ElemTypeGroup**, **MaterialGroup**, **NodeGroup**, **ElementGroup**, etc., which deal with groups of the same type of objects. For example, the **ElementGroup** class is defined which deals with the lists of elements and performs several tasks including assembly of element stiffness matrices and load vectors, and the solution of the system equations as depicted in **Figure 2**. This class is derived from a **LinkedList** class template and so inherits all its operations for the proper management of the list. The **LinkedList** class has been defined in template form so that it can take different types of objects (**ElemType**, **Material**, **Node**, **Element**, etc.) as template arguments. A knowledge base has been incorporated to the **LinkedList** class which creates and maintains an array of pointers of objects so that an individual item in the list can be found out as efficiently as that in a normal array. The creation of the interface classes (which are properly optimized for numerical efficiency) has provided additional benefits in the modification and extension of the code. On one hand, the primitive classes can be modified, extended and made more efficient independently and on the other hand, modification or extension of main class of the problem domain may require revisit of only the interface classes and doesn't affect classes at lower levels. The direct involvement of interface classes such as the **ElementGroup** rather than primitive classes or their inheritances also improves the performance during the solution phase since most of the time data are required in vector or matrix form as an aggregate of all the elements or nodes. One of the

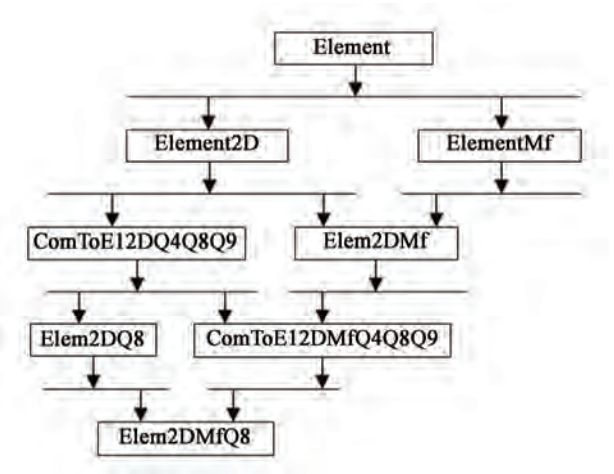


Figure 1. An example of inheritance of class Elem2DMfQ8

```
class ElementGroup : public LinkedList<Element> {
private:
/* . . . */
public:
ElementGroup(); // constructor
~ElementGroup(); // destructor
Element* operator[] (int who); // subscript
            // operator to reference an element
void Assembly(int loadcase); // assembly of
            // stiffness matrices and load vectors
void SkySolve(int loadcase); // skyline
            // reduction solution
void FrontSolve(int loadcase); // frontal
            // solution
/* . . . */
};
```

Figure 2. An ElementGroup class dealing with group of elements

advantages of this interfacing can also be seen in the class **NodeGroup**. Since the class **Node** here is not inherited mainly for efficiency reasons, the class **NodeGroup** is responsible for distinguishing the functionality of nodes in different types of analyses (static, multi-step, transient, etc.).

As discussed earlier, the metal working problem domain consists of multiple structures master as well as slave, each having own set of attributes and interacting with one another through the interface. Here, a **StructureGroup** class is defined which holds a pointer to two master structures (lower die and upper die) and slave structure (workpiece). Each of these structures belongs to

a class **Structure** defined to contain objects of classes such as **ElemTypeGroup**, **MaterialGroup**, **NodeGroup**, **ElementGroup**, etc. In many occasions while performing some tasks, the primitive classes may need to retrieve or request some data from the interface classes at higher level. This is done by defining a pointer to **Structure** within the primitive classes as depicted in diagram shown in **Figure 3**.

Several other classes, in addition to those discussed above, need to be defined in a complete finite element library. For example, several solid model classes such as **Keypoint** and **KeypointGroup**, **Line** and **LineGroup**, **Area** and **AreaGroup** and **Volume** and **VolumeGroup**, are defined that perform modal generation and meshing. Classes **Load** and **Constraint** are also defined dealing with loading and constraints as applied to solid model objects and/or nodes and elements. As has become common practice now, some mathematical variables required in the FEM domain such as vector, matrix, etc. have been represented in template form so that they can take variable type (integer, float, double, etc.) as an argument. Engineering variables such as strains and stresses have also been identified as objects. Several utility classes are also defined for the purpose of processing of finite element results and also for managing the finite element objects.

3. Object-Oriented FEM Analysis of Metal Working Process

The present object-oriented FEM system has been applied to solve several metal working problems after proper validation of the code [19]. It is worthwhile here to discuss qualitatively different aspects of present O-O FEM architecture in solving a metal working problem. In order to do so, a commonly known example problem of spike forging of a cylindrical steel billet in an impression die containing a central cavity is considered. The deformation characteristics of the spike forging are such that the portion of the material near the outside diameter flows radially, while the portion near the center of the top surface is extruded forming a spike. The problem is an axisymmetric rigid-plastic problem and a schematic drawing is shown in **Figure 4**. The steps required and role of different classes in analyzing the problem are briefly described below.

3.1 Discretization and Pre-Processing of Finite Element Model

The element type is defined to be eight-noded isoparametric quadrilateral element with axisymmetric option. This is done by requesting object of **ElemTypeGroup** class which dynamically creates an object of class **ElemType2DMfQ8** and inserts in the list of element types. While the object is created, several characteristics of the element such as number of nodes, number of nodal degrees of freedom, etc. are also defined. Material properties are defined by requesting **MaterialGroup** class which creates an object of an appropriate derived class of base class **Material** and assigns material constants values and defines flow rule which is a function of strain rate, strain and temperature.

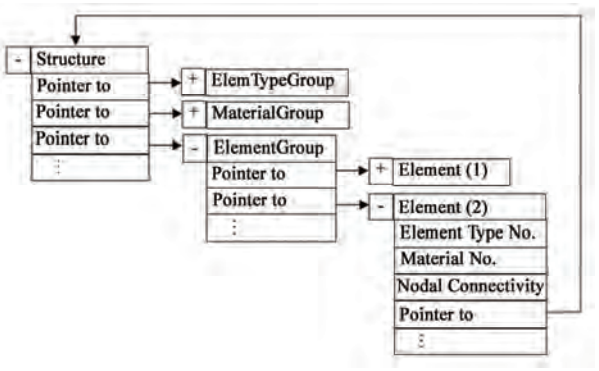


Figure 3. Basic architecture of present O-O FEM implementation ('+' sign at the left corner of each block indicates that the class is shown in unexpanded form while '-' sign indicates that the class is depicted in expanded form)

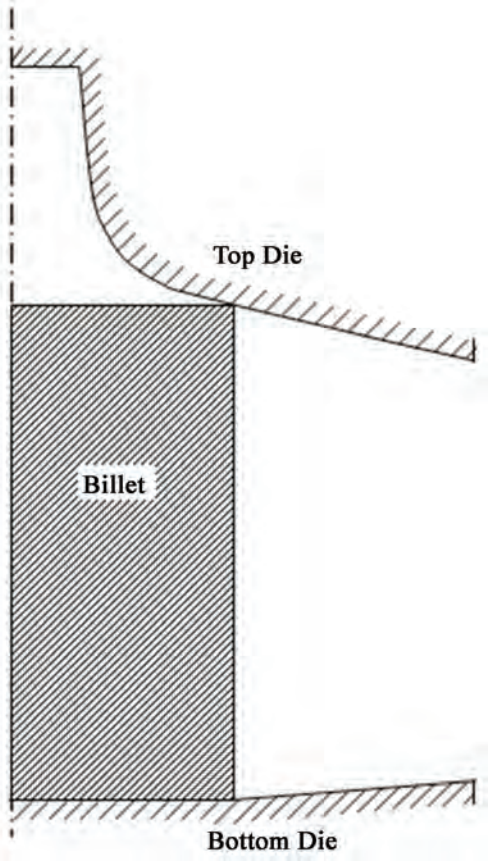


Figure 4. Schematic drawing of spike forging dies and billet

Solid modelling, mesh control and mesh generation are performed using classes such as **KeypointGroup**, **LineGroup** and **AreaGroup**. While meshing, a number of nodal points are established which are created once the solid model objects request object of class **NodeGroup** to do so. The coordinates, etc. are assigned to each nodal point during the process. Similarly elements are defined by an object of class **ElementGroup** which creates objects of class **Elem2DMfQ8** based on the element type currently set and arranges them in a list. The data input to each element include the element type reference number, material reference number and element connectivity.
Since the current problem is a multi-structure problem involving lower and upper dies and workpiece (**Figure 4**), most of the above steps are repeated for each of these structures. The meshes of dies and workpiece (deformed) are shown in **Figure 5**. In this figure, only segments of dies required to define constraints and loading conditions, are modeled and meshed.

3.2 Constraints and Loading

The material flow, which is characterized by spike height variation, depends on the interface friction between die and billet as well as the geometries of dies and billets. Here boundary conditions are prescribed on surfaces of master structures and/or slave structure as appropriate. The die boundary conditions along curved die-workpiece interfaces which constrain flow of material into the die are prescribed using objects of **Constraint** class defined appropriately within each master structure. Similarly, frictional boundary conditions are applied using objects of **Load** class.

3.3 Computation of Element Properties

The rigid-plastic approach states that for a plastically deforming body of volume Ω under surface traction $\{f\}$ prescribed on a part of the surface Γ_f and velocity $\{u\}$ prescribed on the remainder of the surface Γ_u, the variational principle (principle of virtual power) can be written as:

$$\delta\pi = \int_\Omega \bar{\sigma}\delta\dot{\bar{\varepsilon}}\,d\Omega + K\int_\Omega \dot{\varepsilon}_v\delta\dot{\varepsilon}_v\,d\Omega - \int_{\Gamma_f}\{\delta u\}^T\{f\}d\Gamma = 0 \tag{1}$$

where $\bar{\sigma}$ is the effective stress defined as $\bar{\sigma} = \bar{\sigma}(\bar{\varepsilon})$ and $\bar{\sigma} = \bar{\sigma}(\bar{\varepsilon}, \dot{\bar{\varepsilon}})$ for rigid-plastic and rigid-viscoplastic materials respectively, $\bar{\varepsilon}$ is the effective strain, $\dot{\bar{\varepsilon}}$ is the effective strain-rate and $\dot{\varepsilon}_v = \dot{\varepsilon}_{ii}$ is the volumetric strain-rate. K is a penalty constant introduced to impose incompressibility requirement.

In FEM, a continuous velocity field over each element can be defined uniquely in terms of velocities of associated nodal points by introducing the shape function. Equation (1) can now be expressed in terms of nodal point velocities $\{U\}$ and their variations $\{\delta U\}$. From arbitrariness of δU_I, a set of algebraic equations (stiffness equations) are obtained as

$$\frac{\partial\pi}{\partial U_I} = \sum_e\left(\frac{\partial\pi}{\partial U_I}\right)_{(e)} = 0 \tag{2}$$

where *(e)* indicates the quantity at the *e*th element. The capital-letter suffix (*I*) signifies that it refers to the nodal point number.

In metal-forming, the stiffness Equation (2) is nonlinear and the solution is obtained iteratively by using the Newton-Raphson method. The method consists of linearization by Taylor expansion near an assumed solution point $\{U\} = \{U_0\}$ (initial guess), calculating $\{\Delta U\}$ which is the first-order correction of the velocity $\{U_0\}$, and application of suitable convergence criteria to obtain the final solution. After linearization, (2) can be written in the form [20]:

$$[K_T]\{\Delta U\} = \{\Delta R\} \tag{3}$$

where $[K_T]$ is called the tangent stiffness matrix and $\{\Delta R\}$ is referred to as the vector of residual (out-of-balance) force increments. These are calculated as summation of contributions from all the elements as $[K_T] = \sum_e [K_T]_e$ and $\{\Delta R\} = \sum_e \{\Delta r\}_e$, in which $[K_T]_e$ is the element tangent stiffness matrix and $\{\Delta r\}_e$ is the vector of increments of element residual force.

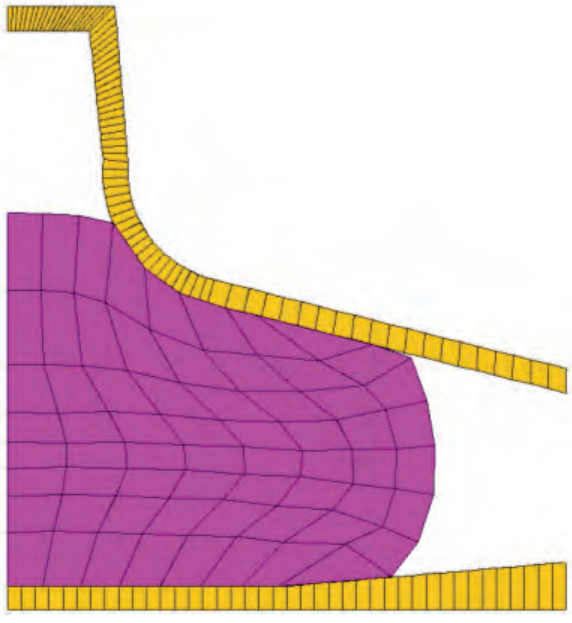


Figure 5. Mesh of the workpiece (deformed) and dies in spike forging of a cylindrical steel billet in an impression die containing a central cavity

For each element e, $[K_T]_e$ and $\{\Delta r\}_e$ are calculated by member functions **StiffMat**() and **LoadVect**() defined in the class **Elem2DMfQ8** or its base classes as appropriate. Calculating stiffness matrix and load vector require some subtasks to be performed such as calculating shape functions, their partial derivatives, strain-displacement matrix, etc. These subtasks are implemented in the corresponding element type class **ElemType2DMfQ8** or its base classes and are performed when a request is made by element objects. Numerical tools such as numerical integration scheme required for above calculations are embedded within an appropriate element class or element type class.

3.4 Assemblage of Elements and Solution of Equilibrium Equations

The element tangent stiffness matrices and residual load vectors are assembled to constitute the global tangent stiffness matrix and global residual force vector as discussed above. The function **Assembly**() defined in the class **ElementGroup** gets the stiffness matrix and force vector from each element and assembles them in skyline vector storage mode. This compacted storage is used by the skyline reduction solution method **SkySolve**() to solve the system of equations. Another solution scheme called frontal solution method is implemented by the function **FrontSolve**(). However, the complete assembly of all element contributions is not required in the case of frontal solution method (**FrontSolve**()) in which assembly and reduction of equations are performed at the same time. Each of the tasks performed by these member functions is decomposed into smaller tasks executed by different member procedures implemented in this class. Since this phase of finite element solution is computationally intensive, the present implementation of this segment is more inclined towards procedure-oriented methodology.

The solution of the system of equations using any of two methods determines the nodal velocity increments (represented in vector form) at a particular iteration in a particular load step. Iteration is continued in a particular load step until convergence is achieved and velocity correction terms become negligibly small. This is followed by calculation of nodal velocities at the load step by the function implemented in **NodeGroup** class. Here it seems that object-oriented approach has resulted in performance loss by first calculating the degree of freedom variable in vector form and then assigning these values to nodes which are identified as objects. Although for a simple linear static analysis this may be correct, the same is not true in the present case of non-linear multi-step analysis since the kind of variables (nodal velocity increments) calculated by solving the system of equations is different from that assigned to the node which is nodal velocities. Further, this assignment to the nodes takes place only at the end of a particular step and not at the end of each iteration.

3.5 Computation of Strains/Stresses and Post-Processing of Results

Once the nodal velocities are known, effective strain rates, effective strains and effective stresses within an element are calculated using the member functions defined in the **Elem2DMfQ8** class or its base classes. This calculation is invoked by the **ElementGroup** class before going to the next step. The **ElementGroup** class is also responsible to save these results in proper file and also prints/plots these results during the solution or at the end of the solution, if and when a request is made. For example, solution of effective strain rate in the mesh of the workpiece at die displacement of $0.6H_0$ is plotted in **Figure 6** as directly obtained by the present implementation of the code.

3.6 Remeshing

The metal working finite element analysis results in severe distortions of mesh and it is essential to frequently refine the mesh or modify some elements during the solution phase. Fortunately, the present implementation has one interface class as **ElementGroup** and another as **Structure** at a higher level. These interface classes effectively perform remeshing and map the data from the old mesh to the new mesh. **Figure 7** depicts the effective strain rate distribution mapped to the new mesh generated by remeshing after the die displacement of $0.6H_0$.

4. Discussion and Conclusions

In recent past, several investigators have implemented object-oriented techniques in FEM and reported benefits because of this. Object-oriented programming can provide stronger support to desirable features of finite element application programs such as easy testing, maintenance, extension and reusability, than the traditional programming.

In object-oriented design, the approach used is to identify and implement a library of finite element data types or classes identified from the real world concept. Each class has well-defined roles and interfaces and therefore can be developed, validated and maintained independently. This approach also permits efficiency concerns to be more easily addressed at the implementation level of each class. The concept of inheritance enables efficient and natural usability of finite element codes. Several new facilities such as new element types, materials and solution techniques may be incorporated with much reduced effort.

Although, the objective and general framework of the object-oriented code in these studies are the same, it is not surprising to find some differences in program design leading to the conclusion that a unique (optimized) O-O implementation of FEM system is difficult to conceptu-

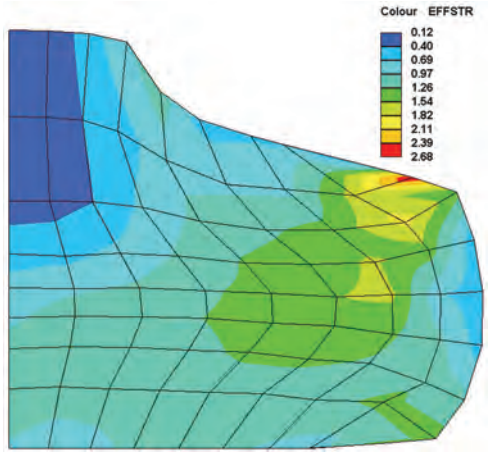


Figure 6. Effective strain rate at die displacement of 0.6Ho during spike forging of a cylindrical steel billet in an impression die

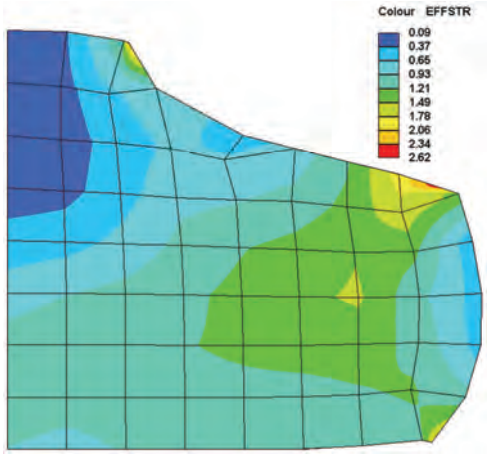


Figure 7. Effective strain rate mapped to the new mesh after remeshing at die displacement of 0.6Ho during spike forging of a cylindrical steel billet in an impression die

alize. Because of a number of factors involved such as software requirements, language features, programming environment and developer's methodology and perspective, varying degrees of object orientation techniques can be achieved, each having its own merit. One of the important challenges in developing O-O finite element codes is also to find the balance between good data abstraction and high computational efficiency. Further, extreme tendency towards data abstraction for each conceptual entity can increase the effort in testing and maintenance of code rather than decreasing it. Since numerous variables are created, used and destroyed during the full phase of FEM analysis, it is worthwhile to represent some of them in vector, matrix or normal structure form with proper naming in Hungarian notation rather than applying data abstraction.

The present paper discusses a special kind of implementation of object-oriented approach used in the design of FEM system for metal working analysis. This design is hybrid in nature consisting of both object-orientation techniques and procedure-oriented approach and can result in a good balance of benefits brought out by the two practices. C++ is used in the development of the program which has several features to support object-oriented programming. This object oriented code has been applied to solve an example problem of metal working and different aspects are presented.

The present object-oriented FEM system has been designed to contain necessary classes and their inheritances identified from the concepts and requirements of FEM. In addition, we create proper interfaces between these primitive classes and the problem domain at different levels so that these primitive objects do not directly interact with the problem domain but through some super objects. These super objects have been identified as either an aggregate of the same objects or a superset of different objects. This concept has provided additional benefits in modification and extension of the code without any compromise in efficiency, since the primitive classes can be modified, extended and made more efficient independently. Further, most of the numerical tools and algorithms are embedded appropriately within these interface classes instead of abstracting out them as objects, thus leading to a mixed design. The static member function concept and other facilities in C++ such as use of **this** pointer helped us to implement a hybrid approach wherever required.

REFERENCES

[1] G. R. Miller, "An Object-Oriented Approach to Structural Analysis and Design," *Computers & Structures*, Vol. 40, No. 1, 1991, pp. 75-82.

[2] T. Zimmermann, Y. Dubois-Pelerin and P. Bomme, "Object-Oriented Finite Element Programming. I. Governing principles," *Computer Methods in Applied Mechanics and Engineering*, Vol. 98, No. 3, 1992, pp. 291-303.

[3] X. A. Kong, "Data Design Approach for Object-Oriented FEM Programs," *Computers & Structures*, Vol. 61, No. 33, 1996, pp. 503-513.

[4] S. Y. Huang, S. Nakai, H. Katukura and M. C. Natori, "An Object-Oriented Architecture for a Finite Element Method Knowledge-Based System," *International Journal for Numerical Methods in Engineering*, Vol. 39, No. 20, 1996, pp. 3497-3517.

[5] T. Zimmermann, P. Bomme, D. Eyheramendy, L. Vernier and S. Commend, "Aspects of an Object-Oriented Finite Element Environment," *Computers & Structures*, Vol. 68, No. 1-3, 1998, pp. 1-16.

[6] G. C. Archer, G. Fenves and C. Thewalt, "A New Object-Oriented Finite Element Analysis Program Architecture," *Computers & Structures*, Vol. 70, No. 1, 1999, pp. 63-75.

[7] L. Yu and A. V. Kumar, "An Object-Oriented Modular Fra-

mework for Implementing the Finite Element Method," *Computers & Structures*, Vol. 79, No. 16, 2001, pp. 919-928.

[8] R. I. Mackie, "Object Oriented Implementation of Distributed Finite Element Analysis in .NET," *Advanced Engineering Software*, Vol. 38, No. 11-12, 2007, pp. 726-737.

[9] B. C. P. Heng and R. I. Mackie, "Using Design Patterns in Object-Oriented Finite Element Programming," *Computers & Structures*, Vol. 87, No. 15-16, 2009, pp. 952-961.

[10] S. M. R. Tabatabai, "Object-Oriented Finite Element-Based Design and Progressive Steel Weight Minimization," *Finite Elements in Analysis and Design*, Vol. 39, No. 1, 2002, pp. 55-76.

[11] O. Pantale, "An Object-Oriented Programming of an Explicit Dynamics Code: Application to Impact Simulation," *Advances in Engineering Software*, Vol. 33, No. 5, 2002, pp. 297-306.

[12] V. Kromer, F. Dufossé and M. Gueurya, "On the Implementation of Object-Oriented Philosophy for the Design of a Finite Element Code Dedicated to Multibody Systems," *Finite Elements in Analysis and Design*, Vol. 41, No. 3, 2005, pp. 493-520.

[13] J. R. Q. Franco, F. B. Barros, F. P. Malard and A. Balabram, "Object Oriented Programming Applied to a Finite Element Technique for the Limit Analysis of Axisymmetrical Pressure Vessels," *Advances in Engineering Software*, Vol. 37, No. 3, 2006, pp. 195-204.

[14] R. Wirfs-Brock, B. Wilkerson and L. Wiener, "Designing Object-Oriented Software," Prentice Hall, Englewood Cliffs, New York, 1990.

[15] I. Graham, "Object Oriented Methods," Addison-Wesley, Reading, Massachusetts, 1991.

[16] G. Booch, "Object-Oriented Design with Applications," The Benjamin/Cummings, Menk Park, 1991.

[17] J. Mackerle, "Object-Oriented Programming in FEM and BEM: A Bibliography (1990–2003)," *Advanced Engineering Software*, Vol. 35, No. 6, 2004, pp. 325-336.

[18] B. Stroustrup, "The C++ Programming Language," Addison-Wesley, Reading, Massachusetts, 2nd Edition, 1991.

[19] S. Kumar, "Finite Element Modeling of Thermomechanical Behavior and Microstructural Evolution in Steel during Hot Deformation Processes," *Project Report*, No. SR/FTP/ETA-31/2005, New Delhi, November 2009.

[20] S. Kobayashi, S.-I. OH and T. Altan, "Metal Forming and the Finite-Element Method," Oxford University Press, Oxford, 1989.

J. Software Engineering & Applications, 2010, 3, 580-587
doi:10.4236/jsea.2010.36067 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

The Exploratory Analysis on Knowledge Creation Effective Factors in Software Requirement Development

Jiangping Wan[1,2], Ruoting Wang[1]

[1]School of Business Administration, South China University of Technology, Guangzhou, China; [2]Institute of Emerging Industrialization Development, South China University of Technology, Guangzhou, China.
Email: scutwjp@126.com, mawangrt@gmail.com

Received March 16th, 2010; revised April 9th, 2010; accepted April 11th, 2010.

ABSTRACT

The knowledge creation effective factors were found in both necessary elements for stimulus of knowledge creation and the key influencing factors of software project success. The research was carried with the specific successful practices of Microsoft Corporation and William Johnson's analysis of R & D project knowledge creation. The knowledge creation effective factors in requirement development project are clarified through deeply interviewing the software enterprises in Guangdong province as well as other corporate information departments. The effective factors are divided with R & D project knowledge creation model in the view of organizational, team, personal and technical four levels through literature research and interview in enterprises, and the empirical study was done with questionnaire and exploratory analysis.

***Keywords*:** *Software Requirement, Knowledge Creation, Project, Organization, Empirical Study*

1. Introduction

The smooth development of software requirements needs an efficient organization to support [1], this paper discusses the knowledge creation factors in software requirements development process in the meta-level of the software process with the instance of Microsoft corporation [2-4]. Software requirement development as a knowledge creation process, Nonaka etc. have attributed the knowledge effect factors to four reasons: intention, autonomy, creative chaos, requisite variety. On this basis, Krogh etc. re-emphasized the importance of friendly relationship to build efficient "Ba" [5]. J. P. Wan etc. analyzed from knowledge management view, some of them proposed a number of effective factors: experience in the domain, knowledge gaps, user participation, administrative support, personal capability, comprehensive training, methodology and related technology and so on [6,7].

This paper is organized as follows: first knowledge creation effective factors are illustrated and the effective factors in the requirement development process are concluded. With deeply interviewing the software enterprises in Guangdong province as well as other corporate information departments, the knowledge creation effective factors in requirement development project are clarified, finally the empirical study is done with questionnaire survey and exploratory analysis.

2. Knowledge Creation Effective Factors

Nonaka attributed knowledge creation effective factors to intention, self-management, creative chaos, redundancy and requisite variety [8], and re-emphasizes the friendly environment in the organization [5].

2.1 Intention

Nonaka indicated that the organization intention is the most important criterion in judging the authenticity of intent. If there is no organization intention, the organization will not be able to judge the value of perceived information and creative knowledge, at the same time, the organization intention must be affected by the organizational value. William Johnson considers that it should give one intention for each project at last, and it is obviously that if there is no intention, the next research will not continue [9]. Software requirements development process

This research was supported by Key Project of Guangdong Province Education Office (06JDXM63002), NSF of China (70471091), and QualiPSo (IST- FP6-IP-034763)

is a knowledge creation process in nature [7]. For example, the first is to establish a shared vision to enhance the team's sense of identity, belonging in the Microsoft corporation's successful rules [3].

2.2 Self-Management

It is that the members or the teams take actions voluntarily to improve the organization creativity. Autonomy team refers to taking the team as operation mainstay voluntarily. For example, William Johnson discovers that personal autonomy is very important for knowledge creation with interviews [9]. It allows large teams to work like small teams by dividing work into pieces, proceeding in parallel but synchronizing continuously, stabilizing in increments, and continuously finding and fixing problems in Microsoft Corporation [10].

2.3 Creative Chaos

Nonaka etc. illustrate that turbulence and creative chaos accelerate the interaction between the organization and environment. Members will start to question the validity of the basic attitudes. It will be opportunity to amend the fundamental thinking and insight. It is obviously that turbulence and creative chaos contribute to organizational knowledge creation [5]. William Johnson discovered that only in a few projects, turbulence and creative chaos possess function which promotes knowledge creation, just same as Nonaka's description with R & D's research projects. In most projects, it is often closely linked with the problem's occurrence. There is no data illustrated that the creative chaos and knowledge creation have a strong correlation [9].

2.4 Redundancy

Redundancy usually refers to the repetition and share for group members and the unnecessary information. It is a kind of redundancy to adopt different technologies to solve the same problem during requirement development process. For example, it is an effective knowledge creation process to build a number of schemes and choose the optimal with review.

2.5 Requisite Variety

William Johnson concluded that all projects regard the requisite variety as a positive factor in the project knowledge creation on the R & D project study [9]. Microsoft Corporation emphasized the small teams, which should be diversification and even in a role. There are usually many different working ways and its members should have different job skills or experience levels in a project team [5].

2.6 Friendly Relationship

Krogh etc. considered that the friendly relationship can remove the distrust, fear and dissatisfaction in the knowledge creation process, and allow team members to explore new markets, new customers, new products and new manufacturing technologies in the unknown territory with enough reassurance [5].

3. The Effective Factors in the Requirement Development Process

The goal of software development is to exploit the high quality software which meets the customers' real requirements timely within the budget. The success of the project depends on good requirement management [11]. This paper discusses the effect factors of requirement development process in perspective of knowledge creation.

3.1 Domain Experiences

Cohen and Levinthal argued that if the organization had more relevant knowledge or experiences, and its absorptive capacity is better, it is a function of its past experiences accumulation [12]. Y. H. Ke etc. analyzed the importance of domain experience for system development, and discovered that the system development experience and deeply understanding of domain knowledge have a positive effect on knowledge transfer [13]. Pete Sawyer and Gerald Kotonya considered that one of the key resources in software requirement acquisition is the domain knowledge. Requirements engineers need to acquire effective knowledge on application domain. It can help them to know the tacit knowledge what stakeholders can not clearly illustrate and learn about the necessary balance between the conflict requirements [14]. For example, Microsoft's team model advocates on the basis of deeply understanding the client's business requirements and familiarly mastering related technologies to develop the project and decision-making. Therefore, the project team members should have the professional and deeply technology and business skills in themselves domains [15].

3.2 Knowledge Gap

It refers that the developer is lacking of business operation knowledge, knowledge of technology, and understanding of user business and software technology [16]. S. Alshawi etc. argued that it is important to have the business and technical knowledge for any enterprise [17].

3.3 User Participation

It is particularly important in information systems' development [18-20]. In Standish Group study, the most reason of project "disagree" factor was the lack of user participation, accounting for 13% in all failure projects. All successful projects illustrated that the most important factor was user participation, accounting for 16% of all projects [21,22]. Standish Group enumerated the top ten critical elements of software projects success with surveying 8380 software projects, the lack of user involve-

ment is listed in the top ten reasons of software project failure [23,24].

3.4 Administrative Support

For example, the Microsoft product development process explicitly specifies that when the projects passing the review and approval by higher managers, and the company will make sure the development progress is going smoothly, and appropriate human and resources for development will deploy through human resource department and finance department [3,4].

3.5 Personal Capability

System analyst is a typical compound talent, and his knowledge structure not only strides the social sciences and natural sciences, but also is the perfect combination of theory and practice. For example, Microsoft asked the staff who participates the software development project have good professionalism and excellent job skills. Staff qualities include: personal quality, passion for product, concerning customer feedback, having cooperation spirit and so on [25].

3.6 Comprehensive Training

For example, Microsoft pays much attention to the developer's re-improved process, including learning and training and so on. The training ways are various, such as professional skills training, many kinds of seminars, training of product plan and development and so on. It also pursues to learn from the past and current research projects and products in system way [4].

3.7 Methodology

Today, many software organizations implement the best industry practices as the software development methodology, such as the SW-CMM (Capability Maturity Model For Software) has been promoted the Software Engineering Institute (SEI) of Carnegie Mellon University in United States since 1987 and so on [15,24].

3.8 Related Technology

Eriksson and Dickson argued that people share the existing knowledge and the new knowledge are created in same time, and the IT infrastructure is one of the factors impacting knowledge creation and share, including supporting information circulation, integrating tools for group problem-solving, such as Intranet, Extranet, video conferencing etc. [26].

4. Interview in Enterprises

We interview some experienced requirement developers, project managers, technical directors and other staffs of the software enterprises in Guangzhou P. R. China for the effect factors of software requirement development. The results are summary as follows.

4.1 Positive Factors

Requirement developer generally plays by the veteran in a team with abundant project experience. These skills include: 1) domain knowledge; 2) communication skill; 3) analysis & arranging capability, comprehensive capability; 4) mastering a certain tool, specially the requirement analysis tools.

It has great importance on the methods and techniques of requirement development process in the software enterprise. First, it carries out the project generally according to the project management standards. Second, it uses prescriptive specification to develop requirement, e.g. the standard template, the standard development tool and so on. Finally, it will use variously interview methods, recording methods and tools in the requirement development process.

4.2 Uncertain Factors

Enterprises always hold uncertain attitude about autonomy. They considered that in the project management, whether the team processes autonomy is related to the project property. Employee must complete their work following the requirement specification and the standard format and submit the required report. However, they can complete independently in really operation.

4.3 Negative Factors

Software companies generally oppose chaotic environment, in particularly they do not like working in a tense environment. The creative chaos environment is not established, and tense working environment usually causes staffs turnoff.

5. The Classification on the Effective Factors of Knowledge Creation in Software Requirement Development

The knowledge creative factors in software requirement development are classified into three areas through the literature research and enterprise interviews (**Table 1**). 10 of which factors are positive, 3 are unable to determine clearly, there are two negative factors.

6. Questionnaire Design and Collection

The quantitative sample survey is taken to test the hypotheses of knowledge creation effective factors in software requirement development.

The questionnaire includes the following six areas: basic information, organizational characteristic, personal characteristic, technical characteristic, knowledge creation and requirement development characteristic relationship.

The first area is about the basic information, including industry type, system user, system type, the number of

system development team, the number of system requirement development team and testee related role in order to have a more clear understanding of the sample. The second area is about organizational characteristic scale, including 4 variables and 14 items. The third area is about the team characteristic scales, including 6 variables and 16 items. The fourth area is about the personal characteristic scale, including 3 variables and 8 items. The fifth area is about the technical characteristic scales, including 2 variables and 7 items (**Table 2**). The sixth area is about the knowledge creation and requirement development relationship characteristic scale, including 4 items.

Table 1. Classification on the effective factors

Relativity	Level	Effect factor	Remark
Positive	Organization	Management support Friendly environment	Literature research and enterprise interviews on the effective factors' classification is basically same.
	Team	Project intention Requisite variety User participation Comprehensive training	
	Personal	Domain experience Personal capability	
	Technology	Methodology Related technology	
Uncertain	Organization Team Personal	Redundancy Self-management Self-management	
Negative	Organization Team	Creative chaos Knowledge gap	

Table 2. Questionnaire detailed corresponding table

Level	Variable factors	Item	References
Organization	Management support	Area one O1～O3	Nonaka (2000), Johnson (2000), Standish Group (1994), Zhang Xianghui (2005), Chen Honggang (2003), James Emery (2002)
	Friendly environment	Area two O4～O10	Nonaka (2000), Johnson (2000), Krogh (1994), Zhang Xianghui (2005), Chen Honggang (2003)
	Creative chaos	Area two O11、O14	Nonaka (1995, 2000), Johnson (2000)
	Redundancy	Area two O12、O13	Nonaka (1995, 2000), Johnson (2000)
Team	Project intention	Area three T1～T2	Nonaka (1995, 2000), Johnson (2000), Zhang Xianghui (2005), Cheng Honggang (2003)
	Self-management	Area three T3～T5	Nonaka (1995, 2000), Johnson (2000), Zhang Xianghui (2005), Chen Honggang (2003)
	Requisite variety	Area three T6～T7	Nonaka (1995, 2000), Johnson (2000), Zhang Xianghui (2005)
	User participation	Area three T8～T10	Standish Group (1994, 1995, 1999), Johnson (2000), Zhang Xianghui (2005), Guinan (1998), Henri Barki (1994), Hirschheim (1994)
	Comprehensive training	Area three T11～T13	Humphrey (2002), Constantine (1995), Chen Honggng (2003)
	Knowledge gap	Area three T14～T16	Alshawi (2003), Linda (2000), Ian McBriara (2003), Gilbert (1996)
Personal	Self-management	Area four I1～I2	Nonaka (1995, 2000), Johnson (2000), Chen Honggng (2003)
	Domain experience	Area four I3～I6	Cohen, Levinthal (1990), Ke Yihua (2005), Chen Honggang (2003)
	Personal capability	Area four I7～I8	Johnson (2000), Zhang Xianghui (2005), Chen Honggang (2003), Tian Junguo (2003)
Technology	Methodology	Area five Te1～Te2	Johnson (2000), Zhang Xianghui (2005), Chen Honggang (2003)
	Related technology	Area five Te3～Te7	Johnson (2000), Zhang Xianghui (2005), Chen Honggang (2003), Ellen Gottesdiener (1999), Eriksson, Dickson (2000, 2003)
4	15	45	

7. The Exploratory Analysis of Requirement Development Effective Factors

7.1 Reclaiming Questionnaire

Questionnaire has surveyed during December 2006 to January 2007 in Guangdong region, including Guangzhou Ferryman Management Consulting Co., Ltd., Guangdong Visionsky Information Technology Co., Ltd., Guangzhou KeenFox Engineering Co., Ltd., Computer and Technologies Solution (Shenzhen) Co., Ltd., nearly 20 enterprises, issued totally 50 e-mails, and totally recovered 26s, all are valid.

7.2 Characteristic of Sample

The highest proportion is the software industry, the number is 17, accounting for 65.4%; the rest of the industry includes financial industry, service industry and other industries accounted for 11.5%, 11.5% and 11.8% correspondingly.

The products which belong to interviewee's team are generally provided to the external clients to use (sample number 10, accounting for 38.5%), internal requirement (sample number 8, accounting for 30.8%) and the combination of the two (sample number 8, accounting for 30.8 %). The products which belong to the interviewee's team, mainly MIS (sample number 17, accounting for 26.6%) and DSS (sample number 13, accounting for 20.3%), others such as ERP, EC, KM, special products, common products as well as other, accounting for 9.4%, 9.4%, 6.3%, 10.9%, 1.6% and 15.6% correspondingly.

The 51 persons and above (sample number 16) is dominated, in the software development team where the interviewee is accounting for 38.5%; 1 to 10, 11 to 20, 21 to 50 are accounted for 26.9%, 23.1 % and 11.5% correspondingly. The 4 to 5 persons is dominated in the requirement development team, accounting for 42.3%, while, 11 persons and above, 6 to 10, and less than 3, are accounting for 26.9%, 19.2% and 11.5% correspondingly. The main interviewees are team project management, the sample number is 12, accounting for 36.2%; developer, requirement person, designer, tester and others are accounting for 26.9%, 7.7%, 3.8% and 7.7% correspondingly. Software industry is dominated in the interviewee's enterprises, the main products is MIS and DSS. Interviewee's software development team usually are large, the number of requirement team is 4 to 5 persons. Mainly interviewees are project managers in order to make the data more persuasive.

7.3 Analysis on Reliability and Validity

The Cronbach's α value is used to determine internal consistency because this paper is exploratory research and items are limited. The reliability of every variable is more than 0.350 after deleting items I3 and Te7, and reliability can be basically acceptable (**Table 3**).

7.4 Statistical Analysis

7.4.1 Descriptive Statistics

The descriptive statistics is illustrated in the **Table 4** according to the variables in **Table 2**. The summary is in the following.

1) The average score of knowledge transformation & requirement development is 4.4712 and indicates that there is close relationship between knowledge transformation and requirement development, it is same as with literature research and enterprise interview.

2) Personal capability, comprehensive training, friendly environment, project intent, customer participation, domain experience and requisite variety and etc., score more than 4 and have a higher acceptance.

3) Redundancy, creative chaos, team self-management, individual self-management, methodology and technology, score lower than 3.5, are basically same as the expected results.

7.4.2 One-Sample T Test

It judges one-sample T test which the test value is 3.5, confidence interval is 95%. If the significant coefficient is less than 0.05, and the upper and lower bounds are greater than 0, indicating its value to more than 3.5 large (have passed the examination); if a significant factor greater than 0.05, or the upper and lower bounds are less than 0, then its value is smaller than 3.5. It is illustrated in **Table 5** that the items are passed the test except redundancy, creative chaos, the team self-management, individual self-management, methodology and technology.

Redundancy, creative chaos, team self-management, individual self-management, methodology and technology do not pass the test where the test value is 3.5. The reverse scoring one-sample T test results is illustrated in the **Table 6** where the test value is 3. Only the individual autonomy is significant, it specified that the individual autonomy plays a negative effect on knowledge creation of requirement development. The other variables do not pass the test, they are unclear type. In addition, the knowledge transfer and requirement development still passing the test where test value 4, it illustrates in **Table 7** that the relationship between the requirement development and knowledge transfer is recognized highly.

8. Conclusions

It is illustrated in **Table 8** that the management support, friendly environment, intention, requisite variety, customer participation, comprehensive training, knowledge gap, domain experience and personal capability and so on through the literature research, interview in enterprise and questionnaire survey, The nine variables have the positive effect on the knowledge creation of requirement development, where the knowledge gap is measured by reducing

Table 3. Reliability of variables

Variably	Item number	Cronbach's α value	Remove item	Reference value
Management Support	3	0.710		
Friendly environment	7	0.771		
Redundancy	2	0.447		
Creative chaos	2	0.683		
Intention	2	0.410		
Team self-management	3	0.532		
Requisite variety	2	0.555		
User participation	3	0.502		
Comprehensive training	3	0.824		0.350
Knowledge gap	3	0.379		
Personal self-management	2	0.703		
Domain experience	2	0.552	I3	
Personal capability	3	0.409		
Methodology	2	0.627		
Technology	4	0.469	Te7	
Knowledge transfer& requirements development	4	0.914		

Table 4. Descriptive statistics

	N	**Minimum**	**Maximum**	**Mean**	**Std. Deviation**
Management support	26	3.0000	5.0000	3.961538	.5360508
Friendly environment	26	3.4286	4.8571	4.131868	.3426739
Redundancy	26	2.0000	4.0000	3.096154	.6636148
Creative chaos	26	1.0000	5.0000	2.865385	.9225800
Intention	26	3.0000	5.0000	4.115385	.4540417
Team self-management	26	2.0000	4.3333	3.480769	.5931590
Requisite variety	26	3.5000	5.0000	4.019231	.3868015
User participation	26	3.6667	5.0000	4.423077	.4274752
Comprehensive training	26	3.3333	5.0000	4.192308	.5178852
Knowledge gap	26	3.0000	4.6667	3.987179	.4664835
Personal Self-management	26	1.0000	4.0000	2.500000	.7745967
Experience in the field	26	2.5000	5.0000	4.038462	.5463163
Personal capability	26	3.5000	5.0000	4.211538	.4043038
Methodology	26	2.5000	4.0000	3.403846	.4902903
Related technology	26	1.7500	4.0000	3.375000	.4962358
Knowledge transfer & requirements development	26	3.7500	5.0000	4.471154	.4707809
Valid N（listwise）	26				

Table 5. Variable one-sample T test

	Test Value = 3.5					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Management support	4.390	25	.000	.4615385	.245023	.678054
Friendly environment	9.402	25	.000	.6318681	.493459	.770277
Redundancy	-3.103	25	.005	-.4038462	-.671886	-.135806
Creative chaos	-3.507	25	.002	-.6346154	-1.007254	-.261977
Intention	6.911	25	.000	.6153846	.431993	.798776
Team self-management	-.165	25	.870	-.0192308	-.258813	.220351
Requisite variety	6.845	25	.000	.5192308	.362998	.675463
User participation	11.011	25	.000	.9230769	.750416	1.095738
Comprehensive training	6.816	25	.000	.6923077	.483129	.901486
Knowledge gap	5.325	25	.000	.4871795	.298763	.675596
Personal self-management	-6.583	25	.000	-1.0000000	-1.312866	-.687134
Domain experience	5.026	25	.000	.5384615	.317800	.759123
Personal capability	8.974	25	.000	.7115385	.548237	.874840
Methodology	-1.000	25	.327	-.0961538	-.294186	.101879
Related technology	-1.284	25	.211	-.1250000	-.325434	.075434

Table 6. Not pass the variable reverse scoring one-sample T test where the test value is 3.5

	Test Value = 3.5					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Redundancy (reverse)	–.739	25	.467	–.0961538	–.364194	.171886
Creative chaos (reverse)	.744	25	.464	.1346154	–.238023	.507254
Team Self-management (reverse)	–4.133	25	.000	–.4807692	–.720351	–.241187
Personal Self-management (reverse)	3.291	25	.003	.5000000	.187134	.812866
Methodology (reverse)	–4.200	25	.000	–.4038462	–.601879	–.205814
Related technology (reverse)	–3.853	25	.001	–.3750000	–.575434	–.174566

Table 7. Knowledge transfer and requirement development one-sample T test

	Test Value = 3.5					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Knowledge transfer and requirement development	5.103	25	.000	.4711538	.281001	.661306

Table 8. The summarized relationship between variables

Correlation	Level	Effect factor	Remark
Positive (+)	Organization	Management support	The same as with literature research and enterprise interview
		Friendly environment	The same as with literature research and enterprise interview
	Team	Intention	The same as with literature research and enterprise interview
		Requisite variety	The same as with literature research and enterprise interview
		User participation	The same as with literature research and enterprise interview
		Comprehensive training	The same as with literature research and enterprise interview
	Personal	Knowledge gap	The same as with literature research and enterprise interview
		Domain experience	The same as with literature research and enterprise interview
		Personal capability	The same as with literature research and enterprise interview
Uncertain (U)	Organization	Redundancy	The same as with literature research and enterprise interview
		Creative chaos	Chaos is a demon for the software business (Larry • Constantine), but the creative chaos has certain positive effect for enterprise management.
	Team	Self-management	The same as with literature search and enterprise interview
	Technology	Methodology	Small-scale projects require little, but large-scale projects need.
		Related technology	Small-scale projects require little, but large-scale projects need.
Negative (−)	Personal	Self-management	Requirement development projects generally obey the project management method, have clear work plan and method.

knowledge gap and it is positive. Considering the literature research and interview in the enterprise, individual independency is determined negative because it illustrates significance in reverse scoring. The others, including redundancy, creative chaos, team self-management, methodology and technology, are unclear. It concludes that the technology and the methodology are support factors of project development and would be very useful for large scale projects. On the contrary, redundancy, creative chaos and team self-management should be avoided as far as possible in the project, because it is inconsistence with the goals of requirement development.

9. Acknowledgements

Thanks for helpful discussion with Mr. Huang Deyi, Mr.Li Jiangzhang, Mr. Chen Zhening, Mr. Wang Shuwen, Mr. Liu Bing, Brenda Huang, and Ms. Zhang Hui etc.

REFERENCES

[1] X. M. Li, L. Y. Sun and Y. L. Wang, "Research on Software Requirement Management Based on Knowledge Management," *Management of Research and Development*, Vol. 17, No. 2, February 2005, pp. 28-32, 39.

[2] Swebok, "Guide to Software the Software Engineering Body of Knowledge," 2004. http://www.swebok.org

[3] X. H. Zhang, "Software Development Process and Management," Tsinghua University Press, Beijing, 2005.

[4] M. A. Cusumano and R. W. Selby, "The Secrets of Microsoft," Free Press, New York, 1995.

[5] G. von Krogh, K. Ichijo and I. Nonaka, "Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation," Oxford University Press, New York and Oxford, 2000.

[6] J. P. Wan, Q. J. Liu, D. J. Li and H. B. Xu, "Research on Knowledge Transfer Influencing Factors in Software

Process Improvement," *Journal of Software Engineering and Application*s, Vol. 3, No. 2, February 2010, pp. 134-140.

[7] J. P. Wan, H. Zhang, D. Wan and D. Y. Huang, "Research on Knowledge Creation in Software Requirement Development," *Journal of Software Engineering and Applications*, Vol. 3, No. 5, May 2010, pp. 487-494.

[8] I. Nonaka and H. Takeuchi, "The Knowledge Creating Company," Oxford University Press, New York, 1995.

[9] W. Johnson, "Technological Innovation and Knowledge Creation: A Study of Enabling Condition and Processes of Knowledge Creation in Collaborative R & D Project," Ph.D. Dissertation, York University, Toronto, 2000.

[10] H. G. Chen, *et al.*, "The Science and Art of Software Development," Electronic Industry Press, Beijing, 2002.

[11] Y. S. Zhang, "The Way of System Analyzer," Electronic Industry Press, Beijing, 2006.

[12] W. M. Cohen and D. Levinthal, "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly*, Vol. 35, No. 1, 1990, pp. 128-152.

[13] Y. H. Ke, "Research on Imparting Knowledge Transfer across Team: Based Information Systems," Master Thesis, Information Management of Institute, National Sun Yat-sen University, Taiwan, 2005.

[14] P. Sawyer and G. Kotonya, "Swebok: Software Requirements Engineering Knowledge Area Description Version 0.5," *IEEE and ACM Project on Software Engineering Body of Knowledge*, San Francisco, July 1999.

[15] J. P. Wan, "Research on Software Product Support Structure," *Journal of Software Engineering and Applications*, Vol. 2, No. 3, October 2009, pp.174-194.

[16] B. Jayatilaka, "The Role of Developer and User Knowledge Domains and Learning in Systems Development," *AMCIS*2000, 2000, pp.1323-1329.

[17] S. Alshawi and W. Al-Karaghouli, "Managing Knowledge in Business Requirements Identification," Logistics Information Management, Vol. 16, No. 5, 2003, pp. 341-349.

[18] H. Barki and J. Hartwick, "User Participation, Conflict and Conflict Resolution," *Information Systems Research*, Vol. 5, No. 2, December 1994, pp. 422-440.

[19] P. J. Guinan, J. G. Cooprider and S. Faraj, "Enabling Software Development Team Performance during Requirement Definition: A Behavioral vs. Technical Approach," *Information Systems Research*, July 1998, pp. 101-125.

[20] R. Hirschheim and H. K. Heinz, "Realizing Emancipatory Principles in Information Systems Development: The Case for ETHICS," *Management Information Systems Quarterly*, Vol. 18, No. 1, March 1994, pp. 83-109.

[21] Standish Group, "Chaos 1994," *The Standish Group International*, Massachusetts, 1994.

[22] Standish Group, "Chaos," *Standish Group Report*, 1995.

[23] Standish Group, "Chaos: A Recipe to Success," *Standish Group Report*, 1999.

[24] W. S. Humphrey, "Managing the Software Process," Reading, Addison-Wesley, Massachusetts, 1989, pp. 19-24.

[25] L. L. Constantine, "Beyond Chaos: The Expert Edge in Managing Software Development," Addison-Wesley, Boston, 2001.

[26] I. V. Eriksson and G. W. Dickson, "Knowledge Sharing in High Technology Company," *American Conference on Information System*, Vol. 36, No. 2, 2000, pp. 1330-1335.

J. Software Engineering & Applications, 2010, 3, 588-592
doi:10.4236/jsea.2010.36068 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

A Neuro-Fuzzy Model for QoS Based Selection of Web Service

Abdallah Missaoui[1], Kamel Barkaoui[2]

[1]LSTS-ENIT, Tunis, Tunisia; [2]CEDRIC-CNAM, Paris, France.
Email: abdallah.missaoui@enit.rnu.tn, barkaoui@cnam.fr

Received January 5th, 2010; revised March 8th, 2010; accepted March 10th, 2010.

ABSTRACT

The automatic selection and composition of Web services rely strongly on the manner to deal with ambiguity inherent to the description of functionalities of these services and the client's requests. Quality of Service (*QoS*) *criteria become crucial in Web services selection and the problem of checking that a web service satisfies a given level of QOS is considered in recent research works. This paper presents a QoS based automatic classification method of web services. These services give generally similar functionalities and are offered by different providers. The main feature of our Web service selection model is to take advantage of the neuro-fuzzy logic for coping with the imprecision of QoS constraints values.*

Keywords*: *Web Service, Selection, Neuro-Fuzzy, QoS, Constraint

1. Introduction

Web services are modular, self-contained, self-describing software components which are distributed over the Web. They can be readily located and checked-out online and dynamically, using a new directory and corresponding search mechanism known as Universal Description, Discovery, and Integration (UDDI).

The requester accesses the description using a UDDI or other types of registry, and requests the execution of the provider's service by sending a SOAP message to it (see **Figure 1**).

SOAP and HTTP provide exactly what they were designed for a simple, lightweight mechanism for interoperability and distributed communication. However, SOAP and HTTP do not provide the traditional enterprise qualities of service typically needed for an enterprise.

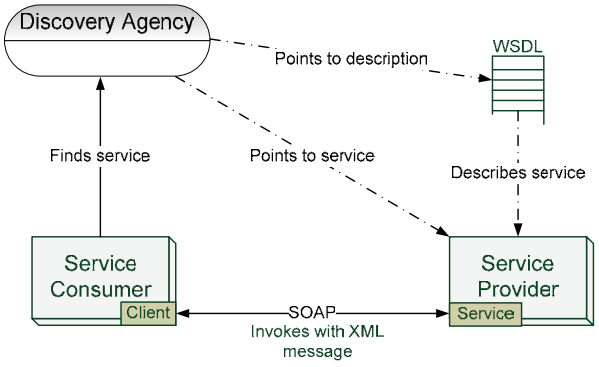


Figure 1. Basic web services architecture

Furthermore, SOAP was designed to be extensible, and it can be extended to support any desired QoS feature by adding SOAP headers to the SOAP messages and adding QoS features to the basic SOAP run-time facilities.

In recent years, several service providers offer QoS features to there customers. Then, multiple providers may provide similar functionalities with different values of non-functional properties.

Their non-functional properties need to be considered during service selection. There are characterised as quality of service (QoS). In many practice cases of business applications, it is recommended to be taken into account during the provider selection.

The human faculty of cognition and perception is very complex, but it possesses an efficient mechanism for information processing and expression [1,2].

This paper applies the neuro-fuzzy decision making approach in the process of selection and choice of the most appropriate web service with respect to quality of service criteria.

This paper is organized as follows: Section 2 presents web service QoS generic description. In Section 3, we discuss and evaluate related works on web service selection adopting a common fuzzy logic approach. Section 4,

we enlighten our QoS requirement description model exploiting neuro-fuzzy logic in order to deal with the imprecision of QoS constraints values. Comments and recommendations for our model are explicitly presented in Section 5. Finally, Section 6 draws a conclusion.

2. QoS Properties of Web Service

Many services are appearing on the Web, several requesters are presented to a group of service providers offering similar services. Different service providers may have different qualities of service.

QoS is one of the most important factors for user's choice of Web service. This will require sophisticated patterns of selection process. It is necessary to provide an appropriate negotiation mechanism between clients and service providers to reach mutually-agreed QoS goals.

QoS management in Web service architecture includes the definition of QoS attributes and the specifications of the following processes: QoS publication, discovery, validation, and monitoring. Many works have studied QoS management on web service. Several QoS languages and architectures are proposed.

The proposed approaches for QoS management can be classified into two groups: one based on extending web service technologies including SOAP, WSDL and UDDI to support QoS [3-5]. The second group use independent entities to perform QoS management [6].

Quality of service is defined by the ability to provide different priorities to different applications, users, or data flow, or to guarantee a certain level of performance to a data flow. A QoS property may include several sub-properties representing different evaluation criteria, e.g. availability, performance, accessibility. In addition, a QoS property can be evaluated by many metrics and therefore it is necessary to define the units of measurements.

QoS in web service architecture is a combination of several qualities or properties of a service, such as:

- Response time: the interval between a user-command and the reception of an action, a result or a feedback from the service.
- Availability: availability is the percentage of time that a service is available for use;
- Accessibility: Accessibility represents the degree that a system is normatively operated to counteract request messages without delay.
- Throughput: It means the max number of services that a platform providing Web services can process for a unit time.
- Reliability: Reliability is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web service.
- Price: represents the money that the customer should pay for this service. It is always associated with the value of the service's functionality, *i.e.* the more a service costs, the more complicated functions it provides.
- Security Level: represents the security level of a service. It includes the existence and type of authentication mechanisms the service offers, confidentiality and data integrity of messages exchanged, non-repudiation of requests or messages, and resilience to denial-of-service attacks [7].

3. Related Work

With the strong popularity of the development of service oriented application, quality of service becomes a central interest of more and more researchers and enterprises. QoS values are proportional to the reliability degree and performance of service and thus play a very important role in the provider choice. A large number of services are exposed constraint information's for comparison providers.

Many researches [5,8-10] have studied QoS issues to improve two processes of discovery and selection of services. Several QoS-aware web service selection mechanisms have been developed in recent years in order to perform the web service composition and to improve performance applications based on services. This mechanisms' main objective is how to how properly select a set of providers that most satisfy the constraints defined by the user in his business processes.

Menascé studies the problem of finding services that minimize the total execution time. It presents an optimized heuristic algorithm that finds the optimal solution without exploring the entire solution space. The solution provided in [11] covers only the important case of execution constraints but not all QoS properties.

Pfeffer proposed a fuzzy logic based model for representing any kind of non-functional service properties. This representation of user preferences enables the fast evaluation of the requested service composition by a fuzzy multiplication of the service composition properties. Thus service composition' properties are measured during or after execution [12].

Other works have been done in fuzzy logic based web service selection. In [12-17], various methods have been proposed for specifying fuzzy QoS constraints and for ranking Web services based on their fuzzy representation.

There is a more suitable technique to quantify functional properties: Linear Programming. These properties are not fitting well for measuring the non-functional attributes, because the majority of them are not easy to be quantified in numerical form. In the meantime, user's QoS constraints regularly remain imprecise or ambiguous due to various human mental states, and it is very difficult to distinguish the priority order among QoS criteria.

Furthermore, in web services selection, the applied QoS constraints are not explicitly defined. It is necessary to relax the constraints to make an optimal solution. The use

of fuzzy logic offers improvements in the overall satisfaction level. The QoS information's represented at abstract level such that it could efficiently select the best services.

However they are still initial efforts which need further investigation for more complete solutions. In the following, we specify several open issues that can be solved:

- When we use some kinds of fuzzy numbers like triangular fuzzy they may not be easy to be defined by end users.
- It is very important to correctly define the QoS properties that we use in the selection process. These criteria's QoS have important effects on ranking methods.
- How to improve fuzzy based web service discovery and the representation of QoS to achieve effective web service selection?
- How to automatically set the weights of service providers attributes?

4. Refinement of the Framework

Neuro-fuzzy technique is the combination of two artificial intelligence (AI) methods: fuzzy logic techniques and neural networks. Neuro-fuzzy system has the ability to handle the nonlinear and complex systems. It is constructed based on the learning algorithm of neural networks technique to adjust the appropriate parameters for fuzzy logic system [18].

In this paper, we aim to solve the selection of web services in a global and flexible manner by introducing a neuro-fuzzy way. For this purpose, we have developed a neural-fuzzy system based on the Sugeno Approach [19]. This is known as the ANFIS (*i.e.*, Adaptive Neuro-Fuzzy Inference Systems). We assume that semantic matchmaking has taken place to identify functionally equivalent services. When several of them are available to perform the same task, their quality aspects become important and useful in the selection process.

An ANFIS is a multi-layered feed forward network, in which each layer performs a particular task. The layers are characterized by the fuzzy operations they perform. **Figure 2** describes the global architecture of this neural-fuzzy system. It shows a *n*-input, type-5 ANFIS. Three membership functions are associated with each input.

We assume that the fuzzy inference system under consideration has *n* inputs $Q_1, Q_2,\ldots,Q_n$ (which are one service attributes). Each input has five linguistic terms, for example, the input Q_1 possesses the terms $\{M_{11}, M_{12},\ldots, M_{15}\}$.

For each input Q_i, we have defined linguistic expressions

L_i = {*Very Poor*(*vp*), *Poor*(*p*), *Medium*(*m*), *Good*(*g*), *Very Good*(*vg*)}

The common fuzzy *if-then* rule has the following type:

Rule 1: If (Q_1 is M_{11}) and (Q_2 is M_{21}) and … and (Q_n is M_{n1}) then f_1 (Q_1, Q_2,…, Q_n)

We denote the output of the i^{th} node in layer *k* as $O_{k,i}$. **Figure 2** shows the schematic diagram of the ANFIS structure, which consists of five layers.

Layer 1: Every node *i* in this layer transform the crisp values to a fuzzy one

$$O_{1,i} = \mu_{M_{1i}}(Q_1) \quad \text{for} \quad i \in \{1,2,...,5\} \quad \text{and}$$

$$O_{1,i} = \mu_{M_{2i}}(Q_2) \quad \text{for} \quad i \in \{1,2,...,5\} \quad \text{and, …, and}$$

$$O_{1,i} = \mu_{M_{ni}}(Q_n) \quad \text{for} \quad i \in \{1,2,...,5\}$$

where Q_K is the input to node *K* and M_{ki} ($k \in \{1,...,n\}$ and $i \in \{1,...,5\}$) is a linguistic label (very poor, poor, fair, good, very good) associated with this node. In other words, $O_{1,i}$ is the membership grade of a fuzzy set $M \in \{M_{11},...,M_{15}\} \cup \{M_{21},...,M_{25}\} \cup ... \cup \{M_{n1},...,M_{n5}\}$ and it specifies the degree which the given input Q_K ($k \in \{1,...,n\}$) satisfies the quantifier *M*.

We use the following generalized Bell function as the membership function (MF)

$$\mu_M(Q) = \frac{1}{1+\left|\frac{Q-c_i}{a_i}\right|^{2b_i}}$$

where a_i, b_i and c_i are the parameters set of MF. The bell-shaped function varies depending on the values of these parameters. Where the parameters *a* and *b* vary the width of the curve and the parameter *c* locates the center

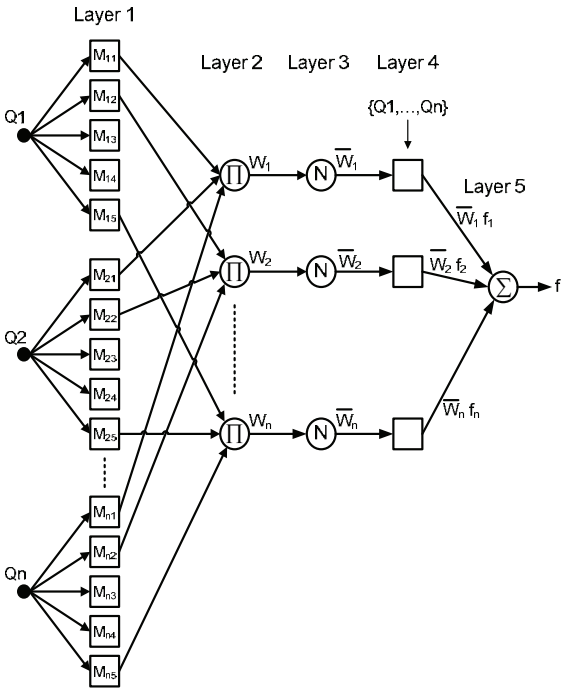


Figure 2. The structure of the neural fuzzy selector

of the curve. The parameter *b* should be positive. The parameters in this layer are referred to as premise parameters. The generalized Bell-shaped function is shown in **Figure 3**.

Layer 2: Every node in this layer is a fixed node labeled ∏. The weighting factor, w_k, of each rule is calculated by evaluating the membership expressions in the antecedent of the rule. This is accomplished by first converting the input values to fuzzy membership values by utilizing the input membership functions and then applying the *and* operator to these membership values.

The *and* operator corresponds to the multiplication of input membership values.

$$O_{2,i} = w_i = \mu_{M_{1i}}(Q_1)\mu_{M_{2i}}(Q_2)...\mu_{M_{ni}}(Q_n)$$

Each node output represents the firing strength of a rule.

Layer 3: Every node in this layer is a fixed node labeled N. The function of the fixed node is used to normalize the input firing strengths.

$$O_{3,i} = \overline{w_i} = \frac{w_i}{\sum_{j=1}^{n} w_j} \quad i \in \{1,...,n\}$$

Layer 4: Every node in Layer 4 is a parameterized function, and the adaptive parameters are called "consequent parameters".

The node function is given by:

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_1^i Q_1 + p_2^i Q_1 + ... + p_n^i Q_1 + p_{n+1}^i)$$

Layer 5: The single node in this layer is a fixed node labeled ∑, which computes the overall output as the summation of all inputs:

$$O_{5,1} = \sum_{i=1}^{n} \overline{w_i} f = \frac{\sum_{i=1}^{n} w_i f}{\sum_{i=1}^{n} w_i}$$

Thus, the ANFIS network is constructed according to the TSK fuzzy model. This ANFIS architecture can then update its parameters according to the backpropagation algorithm [20].

This algorithm minimizes the error between the output of the ANFIS and the desired output.

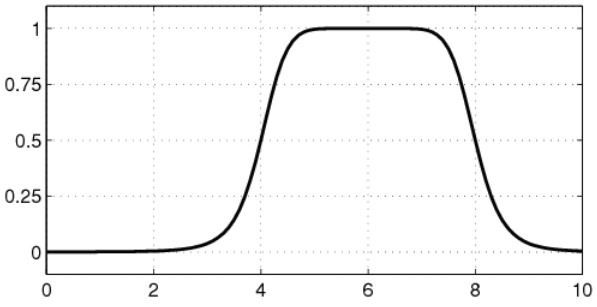


Figure 3. Generalized bell-shaped (a = 2, b = 4, c = 6)

Our neuro-fuzzy system allows classifying service providers in several categories: very poor, poor, fair, good, very good. It allows automating the selection process in the dynamic composition of services.

According to the QoS requirements of web service providers and the functions of Neuro-fuzzy system, we believe that each service invoked is appropriate candidate to increase the composition ability of web services and to decrease the burden of composition cognition and the minimal development cost.

5. Comments and Recommendations

In fuzzy inference system (FIS), The MF of the consequent of each rule is a constant of a fuzzy MF. There are two steps to construct this system: the specification of an appropriate number of input/output and the specification of the shape of MFs. The main problem is that structure identification requires human expertise to solve the parameter estimation. In our selector system we used a different approach, which take advantage of adaptive neural networks algorithms during fitting procedures. MF parameters are fitted to a dataset through a learning algorithm.

A significant number of samples of service providers are needed in order to have better result and to avoid having too many defect values during selection process. The database must be as complete as possible, including samples of providers attributes over a broad range. The number of samples depends on the context and on the runtime environment.

On the other hand, fuzzy logic sets are based on transparence, linguistic rules and establish a framework to include human expertise into modelling. The number of rules is decided by an expert who is familiar with the system to be modeled. In our work, however, no expert is available and the number of membership functions assigned to each input qualities is chosen empirically by examining the desired input-output data.

We merged the fuzzy logic approach with the ability of learning algorithms from neural networks to adjust the model.

6. Conclusions

Web service composition is an emerging area involving important technological challenges. Some of the main challenges are to correctly describe QoS of Web services, to compose them adequately and automatically, and to discover suitable providers and QoS composition issues.

Neuro-fuzzy logic can be seen as a promising formal technique for representing imprecise QoS constraints. In this paper, we have presented a solution to use neuro-fuzzy approach in Web service discovery and selection. We have proposed methods for ranking and selecting web services based on a neuro-fuzzy specification of fuzzy

QoS constraints. The user's constraints are formalized as fuzzy sets and the Qos criteria's are expressed as fuzzy expressions.

This model can be seen as a contribution towards a more complete solution for web service composition integrating fully QoS features.

REFERENCES

[1] J. S. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, 1993, pp. 665-684.

[2] J. R. Jang and C. T. Sun, "Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence," Prentice-Hall, Inc., Upper Saddle River, New Jersy, 1997.

[3] V. Diamadopoulou, C. Makris, Y. Panagis and E. Sakkopoulos, "Techniques to Support Web Service Selection and Consumption with QoS Characteristics," *Journal of Network and Computer Applications*, Vol. 31, No. 2, 2008, pp. 108-130.

[4] A. F. M. Huang, C. W. Lan and S. J. H. Yang, "An Optimal QoS-Based Web Service Selection Scheme," *Information Sciences*, Vol. 179, No. 19, 2009, pp. 3309-3322.

[5] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, 2004, pp. 311-327.

[6] D. A. Menascé, H. Ruan and H. Gomaa, "QoS Management in Service-Oriented Architectures," *Journal of Performance Evaluation*, Vol. 64, No. 7-8, 2007, pp. 646-663.

[7] D. A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing*, Vol. 6, No. 6, 2002, pp. 72-75.

[8] M. Sultana, M. M. Akbar and M. Rouf, "Network Flow Heuristic Algorithm for a Distributed Web Service Selection Problem," *IEEE Conference on Communications, Computers and Signal Processing*, 2009, pp. 465-470.

[9] D. Tsesmetzis, I. Roussaki and E. Sykas, "QoS-Aware Service Evaluation and Selection," *European Journal of Operational Research*, Vol. 191, No. 3, 2008, pp. 1101-1112.

[10] S. Chaari, Y. Badr and F. Biennier, "Enhancing Web Service Selection by QOS-Based Ontology and WS-Policy," *Proceeding of the* 23*rd ACM Symposium on Applied Computing*, Ceará, 2008, pp. 2426-2431.

[11] D. A. Menascé, E. Casalicchio and V. Dubey, "On Optimal Service Selection in Service Oriented Architectures," *Performance Evaluation Journal*, Vol. 67, No. 8, 2010, pp. 659-675.

[12] H. Pfeffer, S. Krüssel and S. Steglich, "A Fuzzy Logic based Model for Representing and Evaluating Service Composition Properties," *The Third International Conference on Systems and Networks Communications*, Bangalore, 2009.

[13] M. Lin, J. Xie, H. Guo and H. Wang, "Solving Qos-Driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction," *IEEE International Conference on e-Technology, e-Commerce and e-Service*, Hong Kong, 2005.

[14] P. Wang, K. Chao, C. Lo, C. Huang and Y. Li, "A Fuzzy Model for Selection of QoS-Aware Web Services," *IEEE International Conference on e-Business Engineering, IEEE Computer Society*, Shanghai, 2006, pp. 585-593.

[15] K. M. Chao, M. Younas, C. C. Lo and T. H. Tan, "Fuzzy Atchmaking for Web Services," *The* 19*th International Conference on Advanced Information Networking and Applications*, Taipei, 2005.

[16] L. Zhuang, Y. F. Huang, W. G. Jian, J. B. Zhou and H. Q. Guo, "Solving Fuzzy QoS Constraint Satisfaction Technique for Web Service Selection," *International Conference on Computational Intelligence and Security Workshops*, Harbin, 2007.

[17] H. Tong and S. Zhang, "A Fuzzy Multi-Attribute Decision Making Algorithm for Web Services Selection Based on QoS," *The IEEE Asia-Pacific Conference on Services Computing*, Guangzhou, 2006.

[18] M. A. Denai, F. Palis and A. Zeghbib, "ANFIS Based Modelling and Control of Non-Linear Systems: A Tutorial," *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, 2004, pp. 3433-3438.

[19] O. Nelles, A. Fink, R. Babuka and M. Setnes, "Comparison of Two Construction Algorithms for Takagi-Sugeno Fuzzy Models," *International Journal of Applied Mathematics and Computer Science*, 2000, pp. 835-855.

[20] P. Werbos, "The Toots of the Back Propagation: From Ordered Derivatives to Neural Networks and Political Forecasting," John Wiley and Sons, Inc, New York, 1993.

J. Software Engineering & Applications, 2010, 3, 593-602
doi:10.4236/jsea.2010.36069 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Scalable Varied Density Clustering Algorithm for Large Datasets

Ahmed Fahim[1], Abd-Elbadeeh Salem[2], Fawzy Torkey[3], Mohamed Ramadan[4], Gunter Saake[1]

[1]Faculty of Information, Otto-Von-Guericke University, Magdeburg, Germany; [2]Faculty of Computers & Information, Ain Shams University, Cairo, Egypt; [3]Kafrelshiekh University, Kafrelshiekh, Egypt; [4]Faculty of science, Menofyia University, Shebeen Elkoum, Egypt.
Email: ahmmedfahim@yahoo.com

Received November 11th, 2009; revised December 8th, 2009; accepted December 11th, 2009.

ABSTRACT

Finding clusters in data is a challenging problem especially when the clusters are being of widely varied shapes, sizes, and densities. Herein a new scalable clustering technique which addresses all these issues is proposed. In data mining, the purpose of data clustering is to identify useful patterns in the underlying dataset. Within the last several years, many clustering algorithms have been proposed in this area of research. Among all these proposed methods, density clustering methods are the most important due to their high ability to detect arbitrary shaped clusters. Moreover these methods often show good noise-handling capabilities, where clusters are defined as regions of typical densities separated by low or no density regions. In this paper, we aim at enhancing the well-known algorithm DBSCAN, to make it scalable and able to discover clusters from uneven datasets in which clusters are regions of homogenous densities. We achieved the scalability of the proposed algorithm by using the k-means algorithm to get initial partition of the dataset, applying the enhanced DBSCAN on each partition, and then using a merging process to get the actual natural number of clusters in the underlying dataset. This means the proposed algorithm consists of three stages. Experimental results using synthetic datasets show that the proposed clustering algorithm is faster and more scalable than the enhanced DBSCAN counterpart.

Keywords*: *EDBSCAN, Data Clustering, Varied Density Clustering, Cluster Analysis

1. Introduction

Over the past several years, even though the computing power has increased exponentially, hard-drive capacity has increased at an order of magnitude greater than that of processor power. Thus the capability to store data has greatly outpaced the capability to process it. As a result, large volumes of data have been generated. The result of this unceasing data collection is that organizations have become data-rich and knowledge-poor [1]. The main purpose of data mining is to extract knowledge from the data at hand, in other words data mining is the process of extracting hidden and interesting patterns from huge datasets.

Data clustering is one of the promising techniques of data mining, which groups a set of objects into classes or clusters such that objects within a cluster have high similarity in comparison to one another, but they are dissimilar to objects in other clusters. Data clustering algorithms can be classified into four categories; (1) partitioning, (2) hierarchical, (3) density-based and (4) grid-based. However, some algorithms may fall into more than one category. By clustering one can identify dense and sparse regions and, therefore, discover overall distribution patterns. Finding clusters in data is challenging when the clusters are being of widely differing sizes, shapes and densities and when the data contains noise and outliers. Although many algorithms exist for finding clusters of different sizes and shapes, there are few algorithms that can detect clusters of different densities.

Basic density based clustering techniques such as DBSCAN [2] and DENCLUE [3] treat clusters as regions of high typical densities separated by regions of no or low densities. So they are able to suitably handle clusters of different sizes and shapes besides effectively separate noise and outliers. But they may fail to identify clusters with varying densities unless the clusters are totally separated by sparse regions. There are some algorithms which handle clusters of different densities, like OPTICS [4] but it does not produce explicit clusters. Traditional

DBSCAN algorithm sometimes has trouble with clusters of varying densities. An enhanced DBSCAN algorithm has been developed to discover varied density clusters from uneven datasets [5]. The disadvantage of this algorithm is its high computational complexity and it does not scale well with the size of huge datasets. It requires O(n log n); where n is the size of the input dataset. Its main advantages are discovering varied density clusters and requiring only one input parameter; (*maxpts*) which determines a suitable value for *Eps* in DBSCAN for each cluster based on the local density. So it is very important to exploit these to two golden advantages and improve the scalability of this algorithm. The original DBSCAN algorithm has been merged with *k*-means algorithm in KIDBSCAN [6], and DBSK [7], and with CLARANS algorithm in [8].

This paper proposes an algorithm that merges among partitioning, density, and hierarchical based methods. It is based on ideas extracted from *k*-means [9], enhanced DBSCAN [5], and CURE [10]. The enhanced DBSCAN selects a suitable value for its parameter *Eps* in each cluster, based up on the local density of the starting point in each cluster, and adopts the traditional DBSCAN for each value of *Eps*. The idea of the enhanced DBSCAN algorithm depends on discovering the highest density clusters at first, and then the *Eps* is adapted to discover next low density clusters with ignoring the previously clustered points. For more details, one can refer to [5].

The proposed algorithm improves the scalability of the EDBSCAN algorithm via partitioning the dataset in order to reduce the search space of the neighborhoods. Instead of examining the whole dataset, the EDBSCAN searches only in the objects within each partition. Merging stage is needed to get the final natural number of clusters in the underlying dataset.

The rest of this paper is organized as follows; some related works are reviewed in Section 2. Section 3 presents the proposed algorithm and describes in details the three stages of it, and analyzes its time complexity. Section 4 presents some experimental results on different datasets to show the performance of the proposed algorithm. Finally the conclusion is presented in Section 5.

2. Related Work

Clustering is the organization of the objects in a dataset D into homogeneous and separated groups with respect to a distance or a similarity measure. Its ultimate objective is to assign the similar objects to the same cluster or group, and dissimilar ones to different clusters. Clustering methods can basically be classified into two main types; partitioning and hierarchical based methods [11]. Partitioning algorithms construct a partition of a dataset D of n objects into a set of k clusters; k is an input parameter for these algorithms. A partitioning algorithm typically starts with an initial partition of D and then uses an iterative control strategy to optimize an objective function. The square error criterion, defined below in (1), is the most commonly used (m_i is the mean of cluster C_i).

$$\sum_{i=1}^{k} \sum_{p \in c_i} (p - m_i)^2 \tag{1}$$

The square-error is a good measure of the within cluster variation across all the partitions. The objective herein is to find k partitions that minimize the square error. Thus, square error clustering tries to make the k clusters as compact and separated as possible and it works well when clusters are compact clouds that are rather well separated from one another. Each cluster is represented by the gravity center of the cluster (k-means algorithm) or by the object that is the most centrally located in the cluster (k-medoids algorithms). Consequently, partitioning algorithms use a two-step procedure. First, they determine k representatives minimizing the objective function. Second, they assign each object to the cluster with its representative "closest" to the considered object. This type of algorithms discovers only spherical shaped clusters of similar sizes, and requires k as input parameter.

Hierarchical algorithms create a hierarchical decomposition of D. The hierarchical decomposition is represented by a dendrogram; a tree that iteratively splits D into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of D. The dendrogram can either be created from the leaves up to the root (agglomerative approach) or from the root down to the leaves (divisive approach) by merging or dividing clusters at each step. Agglomerative hierarchical clustering (AHC) is more stable but its time and memory space requirements are consistently higher. Therefore it is unfeasible for a large dataset. Moreover, for examples, the single link approach is very susceptible to noise and differences in density. While group average and complete link are not as susceptible to noise, but they have trouble with varying densities and cannot handle clusters of different shapes and sizes [12]. Another hierarchical algorithm called CURE [10] has been proposed, it stops the creation of a cluster hierarchy if a level consists of k clusters, where k is one of several input parameters. It utilizes multiple representative points to evaluate the distance between clusters. Thereby, it is adjusting well to arbitrary shaped clusters and avoiding the chain effect problem of the single-link. This results in good clustering quality. But this algorithm has several parameters. The parameter setting does have a profound influence on the result.

Besides the partitioning and hierarchical approaches, density based clustering methods such as DENCLUE [3] and DBSCAN [2] form a third clustering type. These are often used in data mining for knowledge discovery. Density-based clustering uses a local cluster criterion, in

which clusters are defined as regions in the data space where the objects are dense, and remain, separated from one another by low-density regions. The basic idea of the DBSCAN algorithm is that for each point of a cluster the neighborhood of a given radius (*Eps*) has to contain at least minimum number of points (*MinPts*), where *Eps* and *MinPts* are input parameters. These two parameters are global for the dataset. Furthermore it is not easy to determine the best value for *Eps*, and hence DBSCAN can not discover clusters with varied density unless they are totally separated. Density-based clustering has some advantages over *k*-clustering and AHC in discovering clusters of arbitrary shapes and sizes.

However, in previous studies, it was shown that current density based clustering works well only on a simple dataset where cluster densities are similar [4]. Density based clustering is important for knowledge discovery in databases. Its practical applications include biomedical image segmentation [13], molecular biology and geospatial data clustering [14], and earth science tasks [2].

The enhanced DBSCAN algorithm [5] has previously been proposed to solve the problem of varied density clusters. In this paper, we improve the scalability of the enhanced DBSCAN by first partitioning the dataset to reduce the search space of the neighborhoods, then applying the enhanced DBSCAN on each partition separately, and finally applying merging procedure to get the actual number of clusters in the whole dataset.

3. The Proposed Algorithm

In this section, we describe the details of the proposed algorithm which called scalable enhanced DBSCAN (SEDBSCAN). This algorithm composed of three main stages; the first stage is to partition the dataset into *k* super-clusters, the second stage is to find out the sub-clusters within each partition (super-cluster), the third stage is to find out the natural number of clusters by merging dense sub-clusters from different partitions (super-clusters). **Figure 1** depicts these three stages.

3.1 Partitioning Stage

The main purpose of this stage is to partition the underlying dataset into a finite number of smaller datasets, because most algorithms perform efficiently well with small datasets more than large datasets. So this stage will improve the scalability of the proposed algorithm. In this stage, discovering varied-shaped clusters is not of high importance, but the most important issue is getting the initial partitions as soon as possible. To fulfill this goal, an algorithm with time complexity O(*n*) should be used. *k*-means algorithm [9] is the best choice for this stage. In the following, a brief review about *k*-means is presented. The *k*-means is classified as a partitioning method that requires only one input parameter, *k*, which represents the number of clusters. The main steps of this algorithm are as follows:

Input: the number of clusters *k*, and a dataset containing *n* objects.

Output: a set of *k* clusters which minimizes the square error as in Equation (1).

Method:

1) Arbitrary select *k* centers as the initial solution.

2) Compute membership the objects according to the current solution.

3) Update cluster centers according to new memberships of the objects.

4) Repeat steps 2 and 3 until convergence.

The *k*-means is scalable and efficient in processing large datasets due to its low time complexity (*i.e.* O(*n*)). Since the *k*-means works well with spherical shaped clusters of similar size, we expect that the result of this stage is not always correct. Consider the following example depicted in **Figure 2**. It is shown that the k-means produces six clusters. One can easily discover that this result is not really correct, as an actual cluster may be distributed over more than one partition, or some clusters are merged together.

The second stage will handle each partition as a new separate dataset. We used a scalable version of the *k*-means algorithm. This version was previously presented in [9]. It reduces the number of computations in the second step of the original *k*-means. It achieved its scalability through redistributing the objects that became far from their previous centers, while the objects which become closer to their centers will not be redistributed.

3.2 EDBSCAN Each Partition

This stage applies the enhanced DBSCAN on each partition. The main advantage of this algorithm is that it has the ability to discover varied density clusters. So the proposed

Figure 1. Main stages of the scalable EDBSCAN

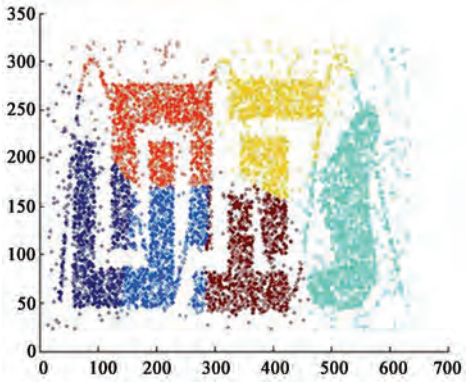


Figure 2. Initial partition resulted from k-means algorithm

algorithm will inherit this important advantage. A brief review about EDBSCAN [5] is presented. The EDBSCAN algorithm is based on the DBSCAN [2] algorithm, but it surmounts the problem of fixed *Eps* in DBSCAN. The EDBSCAN needs two input parameters; they are *Minpts* and *Maxpts*, *Minpts* < *Maxpts* < 20. These two parameters determine the minimum and maximum density for core points respectively. *Maxpts* also determines the *Eps* for each cluster according to the highest local density of its starting point. *Minpts* is fixed to 4 as in DBSCAN algorithm. Thus the user will set only one input parameter. The main steps in this algorithm are as follows:

Input: *Maxpts*, and dataset containing *n* objects.

Output: actual clusters discovered from the input dataset.

Method:

1) Find the *k*-nearest neighbors for each object *p*. (*i.e.* $N_k(p)$) and keep them in ascending order from *p*.
2) Set local density value for each object *p* as $DEN(p,y_1,\ldots,y_k)$ which represents the sum of distances among the object *p* and its *k*-nearest neighbors.
3) Rearrange the objects in descending order according to their local densities.
4) *ClusId* = 1.
5) Starting from the first unclassified object *p* in the sorted data do the following:

a) *Eps* = distance to *maxpts*-nieghbor for the object *p*.

b) Assign the object *p* to the current cluster (*ClusId*).

c) Append its unclassified neighbor q_i, wrt. *Eps* and *Minpts* to the seed list SL_p in ascending order of their distance to *p*, Continue expanding current cluster until no object can be assigned to it.

6) *ClusId* = *ClusId* + 1.

Assign the next unclassified object to the current cluster and go to step 5 until all objects are classified.

In EDBSCAN there are no border points as in DBSCAN. So it treats small clusters as noise and discards them from the data. Experimentally, a small cluster is the cluster that has less than 0.006 of the size of the dataset. **Figure 3** shows the sub-clusters discovered from each partition resulted from the first stage. This figure shows 16 sub-clusters discovered from the six initial partitions.

From **Figure 3** one can see that some sub-clusters are merged together to get the natural clusters in the underlying dataset. To merge sub-clusters, the idea of representative points proposed by CURE algorithm [10] will be used, but not using the same technique of the CURE. The *k*-means algorithm will be used for selecting these representative points. Based on these representtttatives, two clusters with the most near representatives will be merged in a hierarchical fashion until termination condition is hold. This process will be done in the third stage.

3.3 Merging Dense Clusters

This stage aims to merge the nearest dense sub-clusters detected from applying the EDBSCAN on each partition in the second stage. As it is known there is no border point can be detected by EDBSCAN algorithm, since it uses minimum and maximum density for core points. Therefore every point in the given dataset is a core point, and the maximum density determines the *Eps* value in each cluster according to the density of region where the initial point resides. Referring to the method presented in [8], one can find that the DBSCAN algorithm can detect large number of border points in each cluster. We need smaller number of points from each cluster as representatives to reduce computational complexity of the merging stage. So the merging stage will search for *k* representatives from each sub-cluster using the k-means algorithm. This means, we apply the *k*-means algorithm on each generated sub-cluster from applying the EDBSCAN on each partition in the second stage. Referring to **Figure 3** we have 16 sub-clusters, so the size of clusters is very small compared to the size of the whole dataset. Therefore, the time required to get these representatives using the k-means will be very small and can be neglected. We use these representatives for merging dense sub-clusters. **Figure 4** depicts the *k* representatives for each sub-cluster shown in **Figure 3**.

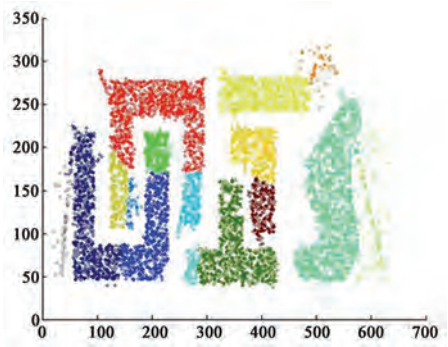


Figure 3. Result from applying EDBSCAN on each partition

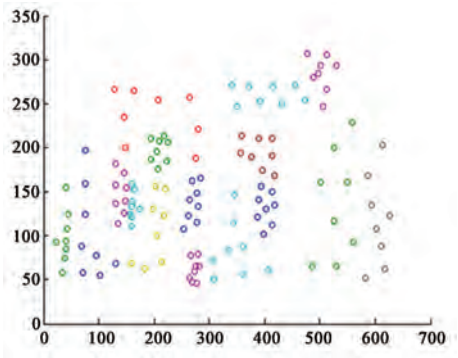


Figure 4.The *k* representatives for each cluster generated from the second stage (*k* = 8)

Two dense sub-clusters with the most nearest representatives will be merged together in a hierarchical form. This idea is similar to that was proposed by CURE algorithm [10]. We need a threshold to stop the merging process. This may be a simple problem because we already have some information about the data from the previous stages; like the number of sub-clusters, *Eps* in each cluster (density level of each cluster), distances among cluster's representatives. Intuitively, two sub-clusters in second stage are not allowed to be merged together if they are belonging to the same partition (super cluster) in the first stage.

The algorithm arranges the merging distances in ascending order. By examining the plot of distances, we can easily select threshold distance to stop the merging process. **Figure 5** presents the merging distances plot for the sub-clusters in **Figure 3**.

Since we have 16 sub-clusters from the second stage, we need 15 merging steps for merging all sub-clusters into a single cluster. We search for the gaps in distances plot. From **Figure 5** we notice that only one gap between distances 28.7 and 47.6. So any value between these two values will be a threshold distance. The merging process is applied seven times so that the final number of clusters will be 16-7 = 9 clusters as shown in **Figure 6**.

We can get rid of the smallest three clusters as noise, and return the remaining six clusters as a final result.

3.4 Complexity Analysis

The execution time of the proposed algorithm is composed of three components; the first one is the time for executing the *k*-means algorithm on the entire dataset which is O(*n*), where *n* is the size of the input dataset. The second component is the time for executing the EDBSCAN algorithm on each part resulting from the *k*-means in the first stage, the EDBSCAN requires O(m^2) if it doesn't use an index structure like R*-tree, or O(*m* log *m*) if it uses R*-tree; where *m* is the size of the input dataset, and *m* is very smaller than *n* since *m* is the size of one part of the original dataset. The third component is the time for getting the *k* representatives from each sub-cluster, and the merging process, this time is very small and can be neglected, because we apply the *k*-means algorithm on each sub-cluster which is very small compared with the size of the original dataset, and the total number of representatives is also very small compared to the original dataset size. Furthermore, we don't find the distances among representatives belonging to the same part in the initial partition. Hence the entire execution time of the proposed algorithm is O(*n* + *m* log *m*) which is definitely smaller than O(*n* log *n*).

4. Experimental Results

In this section the performance of the proposed algorithm is evaluated. This algorithm has been implemented in C++.

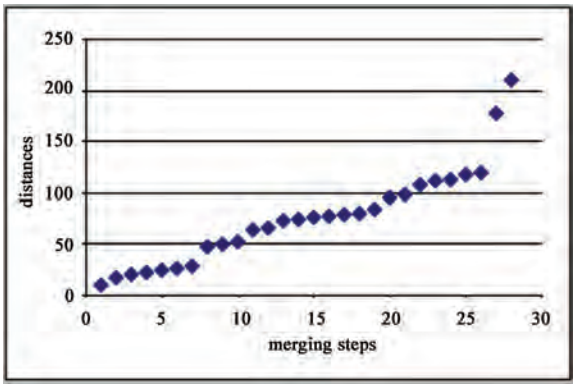


Figure 5. Merging distances plot

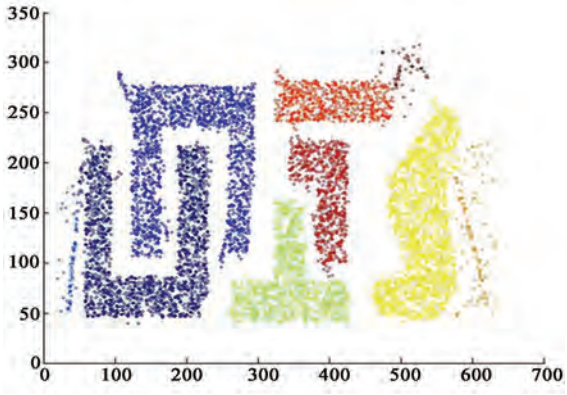


Figure 6. Natural number of clusters in dataset

We have used many synthetic datasets to test the proposed algorithm. The experiments have been done on seven different datasets containing 2D points. The first dataset has six clusters of different sizes, shapes, and orientation, as well as random noise points and special artifacts such as streaks running across clusters, this dataset is used as an example in **Figure 2**. The second dataset has six clusters of different shapes. Moreover, it also contains random noise and special artifacts, such as a collection of points forming horizontal streak, this dataset is shown in **Figure 7**. The third dataset has eight clusters of different shapes, sizes, and orientation, some of which are inside the space enclosed by other clusters. Moreover, it also contains random noise and special artifacts, such as a collection of points forming vertical streaks. This dataset is shown in **Figure 8**. The fourth dataset has eight clusters of different shapes, sizes, densities, and orientation, as well as random noise. A particularly challenging feature of this data set is that the clusters are very close to each other and they have different densities, this dataset is shown in **Figure 9**. The fifth dataset has four clusters of different shape, size, and density. A particularly challenging feature of this data set is that, the clusters are very close to each other and they have different densities, this dataset is shown in **Figure 10**. The sixth and seventh datasets have clusters of different

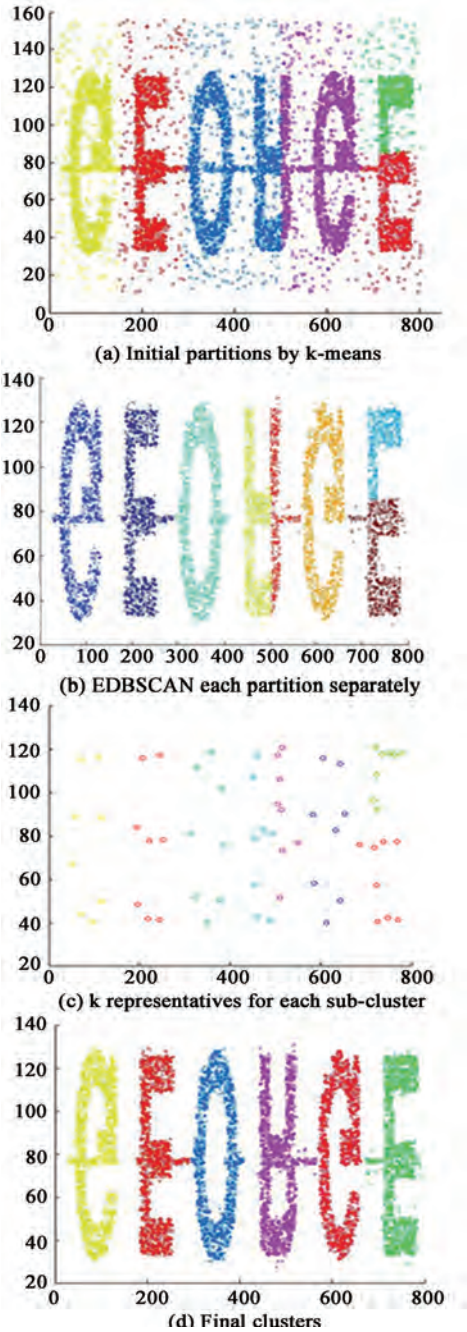


Figure 7. Clusters obtained from dataset 2

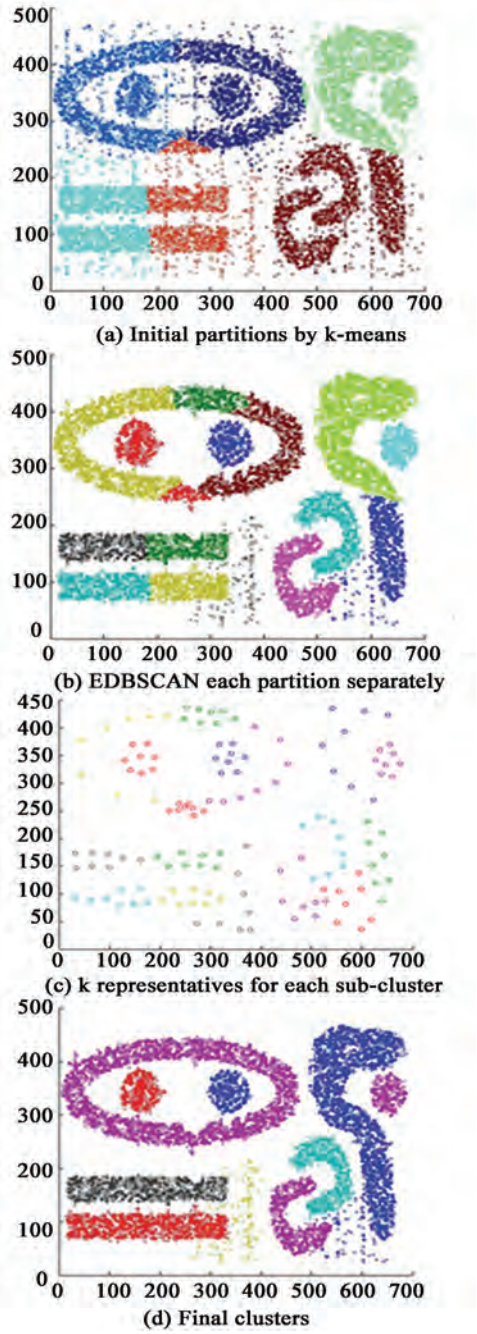


Figure 8. Clusters obtained from dataset 3

sizes and densities. These datasets are shown in **Figures 11** and **12**. The last three datasets are presented here to insure that the proposed algorithm is very efficient in discovering varied density clusters without requiring separating regions with low density.

We evaluate the performance of the proposed algorithm compared to the original EDBSCAN using different synthetic datasets include noise; the sizes of these datasets are ranging from 3147 to 10000 points in two-dimensions (2D), and they contain varied shaped, size, and density clusters. **Figure 7** shows the three steps of the proposed algorithm on dataset 2 of size 8000 points.

For comparing the proposed algorithm with the original EDBSCAN, the same value for the parameter *maxpts* in both EDBSCAN and the proposed algorithm is used in order to demonstrate the higher enhancement of the proposed algorithm. **Figure 8** shows the three steps of the proposed algorithm on the third dataset containing 10000 points.

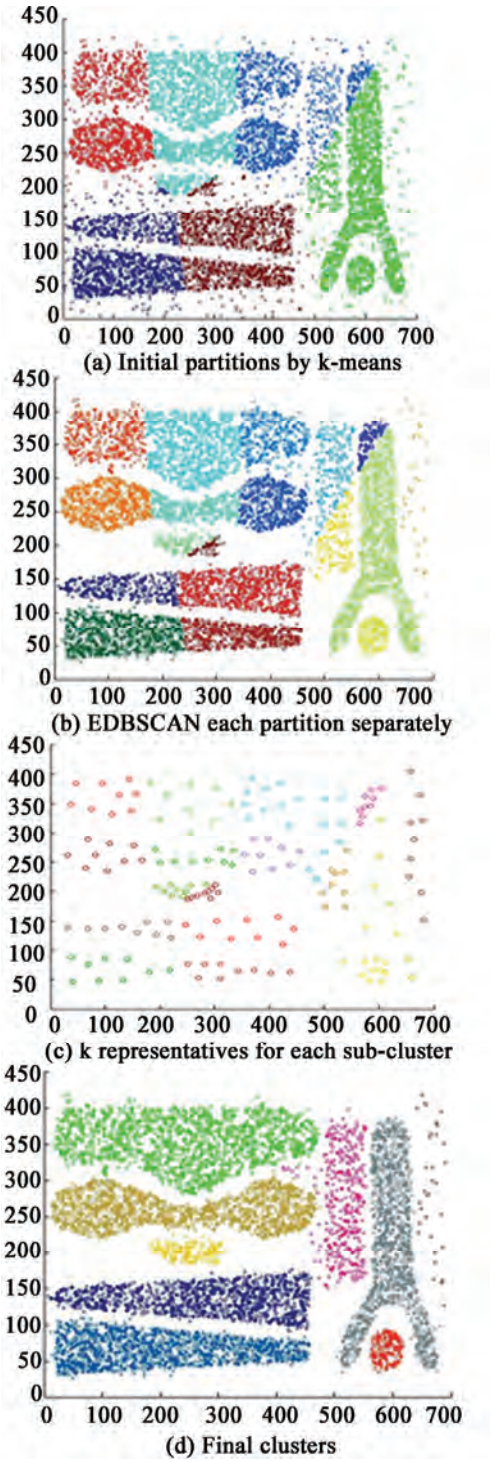


Figure 9. Clusters obtained from dataset 4

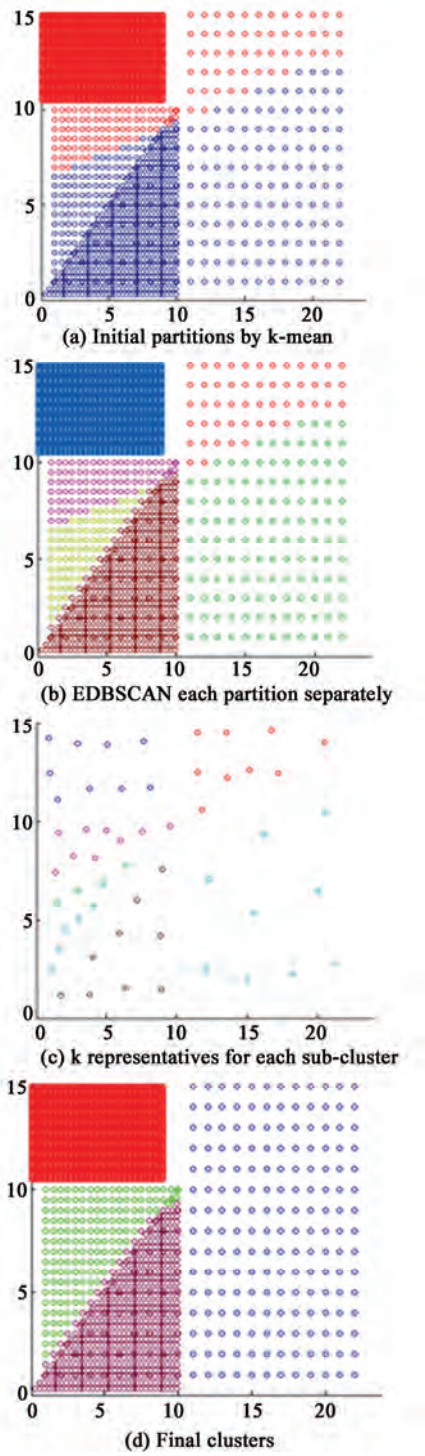


Figure 10. Clusters obtained from dataset 5

However dataset 3 has clusters of varied shapes that are nested with each other, the proposed algorithm could discover the right clusters in the data. There is a small mistake in the ring cluster that has two different spherical clusters within it. This cluster has three levels of density in its right half. This mistake resulted from the EDBSCAN which considered the small cluster as outlier and removed it from the data, but this problem can easily be resolved by assigning the outlier clusters to the nearest cluster. **Figure 9** shows the three steps of the proposed algorithm on

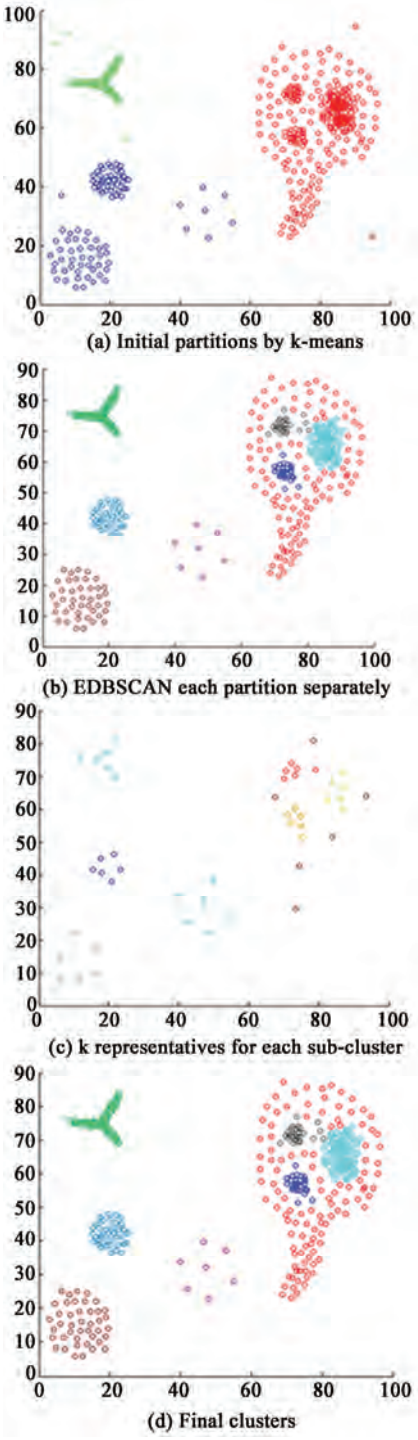


Figure 11. Clusters obtained from dataset 6

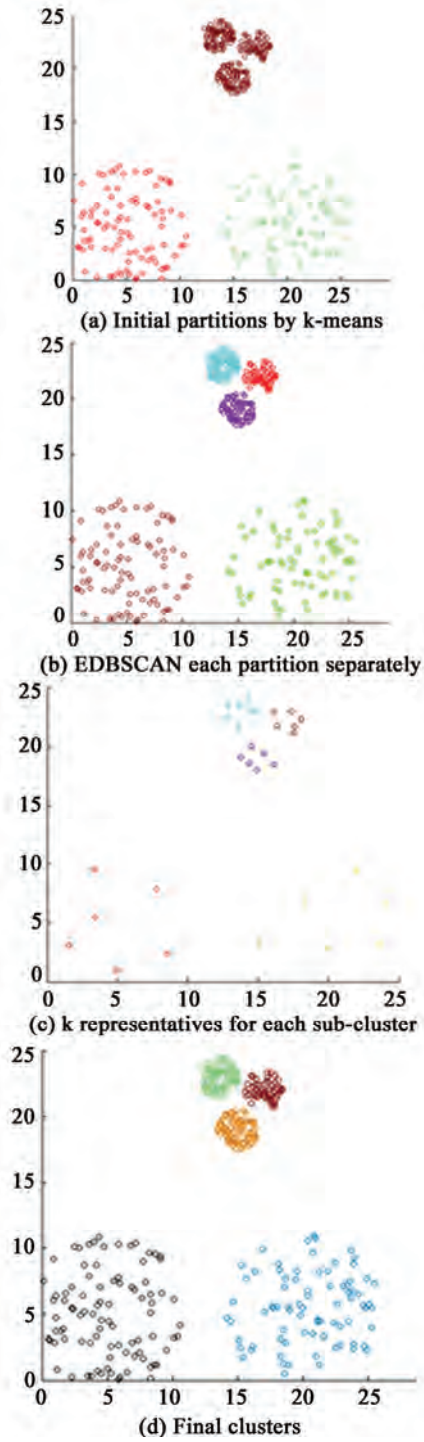


Figure 12. Clusters obtained from dataset 7

dataset 4 of size 8000 points.

Dataset 5 has four clusters of varied shapes, sizes and densities. The proposed algorithm could successfully discover the correct clusters. **Figure 10** shows the result on dataset 5 that has varied-density clusters with no separation among them.

From **Figures 11** and **12**, one can find that the final result is the same as the result from EDBSCAN in step 2 of the proposed algorithm. This is because the clusters in the same part are not allowed to be merged, and each part has clusters that are totally separated from clusters in other parts.

From the experimental result, we can deduce that the proposed algorithm is able to discover clusters with varied shapes, sizes, and densities efficiently. It can easily be noticed that the result obtained by the proposed algorithm is not exactly similar to the one of EDBSCAN on the entire dataset. Instead, the proposed algorithm produces additional small clusters that are discarded as noise by EDBSCAN on the entire dataset. This is not an irresolvable problem. If we want to get the identical result of EDBSCAN, we should remove the small clusters as noise. The results of the original EDBSCAN on the entire datasets are shown in **Figure 13**.

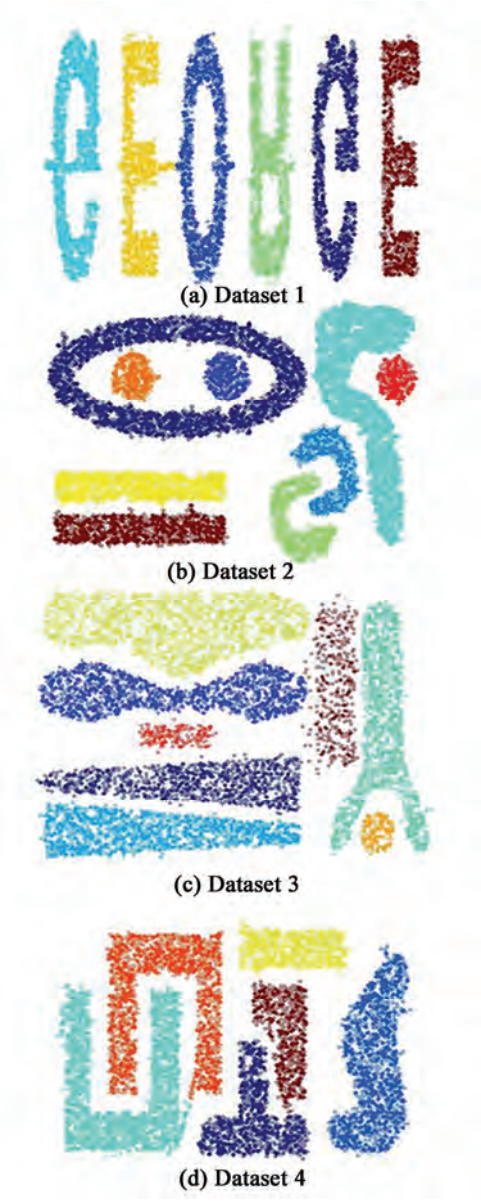


Figure 13. The results from the original EDBSCAN on the entire datasets

The proposed algorithm is more scalable than EDBSCAN algorithm, the scalability of the proposed algorithm comes from partitioning the original dataset into finite set of smaller datasets in the first stage, and depending on *k* representatives from each sub-cluster to get the actual clusters in the original dataset. The following **Figure 14** depicts the execution time of the proposed algorithm and EDBSCAN without using an index structure like R*-tree or canopies. We use only the fist four datasets because the other datasets are very small. From **Figure 14** it is noticed that the proposed algorithm (SEDBSCAN) reaches a speed up factor 5.25, 5, 5.33, and 5.25 for datasets 1, 2, 3, and 4 respectively.

5. Conclusions

This paper has introduced a scalable and efficient clustering algorithm for discovering clusters with varied shapes, sizes, and densities. The proposed algorithm has exploited all the advantages of previous different algorithms (e.g., the scalability of the *k*-means, and the ability of discovering clusters from uneven datasets of the EDBSCAN, and the idea of multiple representatives taken from the CURE algorithm, and finally the idea of local density taken from the DENCLUE Algorithm) and has overcame their disadvantages. So this algorithm collects ideas from partitioning, hierarchical, and density based methods. Generally speaking, combining all these ideas into the proposed algorithm allows it to be scalable and more efficient in discovering clusters of varied density. The experimental results have given a clear indication on the scalability of the proposed algorithm. Furthermore the proposed algorithm has better performance than EDBSCAN with speed up factor up to 5 times. Additionally, the algorithm can be partially implemented in parallel (in the second stage) which helps in improving the scalability by a significant factor.

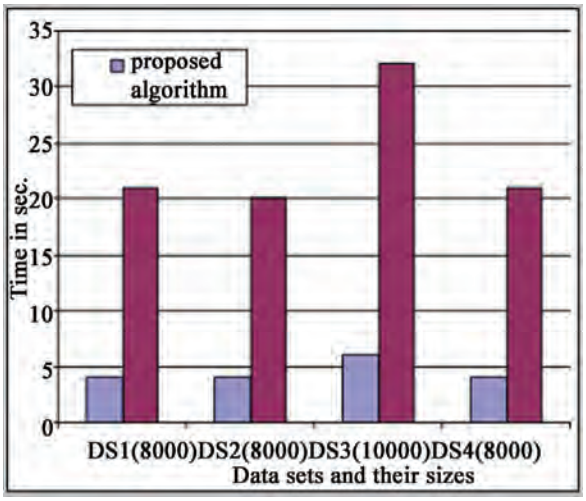


Figure 14. The execution time comparison

REFERENCES

[1] J. MacLennan, Z. Tang and B. Crivat, "Data Mining with SQL Server 2008," Wiley Publishing, Indiana, 2009.

[2] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A Density Based Algorithm for Discovering Clusters in large Spatial Datasets with Noise," *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226-231.

[3] A. Hinneburg and D. Keim, "An Efficient Approach to Clustering in Large Multimedia databases with Noise," *Proceedings International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 58-65.

[4] M. Ankerst, M. Breunig, H. P. Kriegel and J. Sandler, "OPTICS: Ordering Points to Identify the Clustering Structure," *Proceedings of the International Conference on Management of Data* (*SIGMOD*'99), 1999, pp. 49-60.

[5] A. Fahim, G. Saake, A. Salem, F. Torkey and M. Ramadan, "Enhanced Density Based Spatial clustering of Application with Noise," *in Proceedings of the* 2009 *International Conference on Data Mining*, Las Vegas, July 2009, pp. 517-523.

[6] C.-F. Tsai and C.-W. Liu, "KIDBSCAN: A New Efficient Data Clustering Algorithm," *Artificial Intelligence and Soft Computing-ICAISC*, Springer, Berlin/Heidelberg, 2006, pp. 702-711.

[7] R. Xin and C. H. Duo, "An Improved Clustering Algorithm," *International Symposium on Computational Intelligence and Design*, 2008, pp. 394-397.

[8] Y. El-Sonbaty, M. Ismail and M. Farouk, "An Efficient Density Based Clustering Algorithm for Large Databases," *Proceedings of the* 16*th IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*), 2004, pp. 673-677.

[9] A. Fahim, A. Salem, F. Torkey and M. Ramadan, "An Efficient Enhanced *k*-Means Clustering Algorithm," *Journal of Zhejiang University Science A*, Vol. 7, No. 10, 2006, pp. 1626-1633.

[10] S. Guha, R. Rastogi and K. Shim, "CURE: An Efficient Clustering Algorithms for Large Databases," *Proceedings of ACM SIGMOD International Conference on Management of Data*, Seattle, 1998, pp. 73-84.

[11] A. K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, Vol. 31, No. 3, September 1999, pp. 264-323.

[12] L. Ertoz, M. Steinbach and V. Kumar, "A New Shared Nearest Neighbor Clustering Algorithm and its Applications," *Workshop on Clustering High Dimensional Data and its Applications at* 2*nd SIAM International Conference on Data Mining*, 2002.

[13] M. Emre Celebi, Y. Alp Aslandogan and P. R. Bergstresser, "Mining Biomedical Images with Density-Based Clustering," *Proceedings of the International Conference on Information Technology*: *Coding and Computing*, Washington, DC, *IEEE Computer Society*, Vol. 1, 2005, pp. 163-168.

[14] J. Sander, M. Ester, H.-P. Kriegel and X. Xu "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications," *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, 1998, pp. 169-194.

J. Software Engineering & Applications, 2010, 3, 603-609
doi:10.4236/jsea.2010.36070 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Dynamic Two-phase Truncated Rayleigh Model for Release Date Prediction of Software

Lianfen Qian[1], Qingchuan Yao[2], Taghi M. Khoshgoftaar[2]

[1]Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, USA; [2]Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, USA.
Email: lqian@fau.edu, qingchuan_yao@yahoo.com, taghi@cse.fau.edu

Received October 23rd, 2009; revised November 13th, 2009; accepted November 15th, 2009.

ABSTRACT

Software reliability modeling and prediction are important issues during software development, especially when one has to reach a desired reliability prior to software release. Various techniques, both static and dynamic, are used for reliability modeling and prediction in the context of software risk management. The single-phase Rayleigh model is a dynamic reliability model; however, it is not suitable for software release date prediction. We propose a new multi-phase truncated Rayleigh model and obtain parameter estimation using the nonlinear least squares method. The proposed model has been successfully tested in a large software company for several software projects. It is shown that the two-phase truncated Rayleigh model outperforms the traditional single-phase Rayleigh model in modeling weekly software defect arrival data. The model is useful for project management in planning release times and defect management.

***Keywords*:** *Software Testing, Weekly Defect Arrival Data, Single-Phase Rayleigh Model, Two-Phase Truncated Rayleigh Model, Software Reliability*

1. Introduction

Software reliability is a key attribute of software quality. Various models have been developed for software reliability engineering [1]. The rising complexity, size and functionality of software systems make software reliability prediction difficult. The problem is compounded with short development times and strict release deadlines. Consequently, predicting the release date for achieving pre-specified system reliability has become a very important issue in software project development. Reliability modeling can not only assist in fulfilling commitments and project deadlines, but also aid in efficient resource management and planning.

Software reliability is the probability of failure-free software operation for a given period of time in a given operating environment. The key attribute in software reliability engineering is the number of defects observed in specified time intervals (e.g. weeks). Software reliability prediction models assess a software product's reliability or estimate the number of latent defects when it is released to the customers. Such an estimate is important for two reasons: 1) as an objective statement of the quality of the product and 2) for resource planning in the software maintenance phase.

There are two categories of software reliability models: static and dynamic models. Among the static models, Bayesian belief networks [2] and utilizing software process metrics are relatively popular. Related literature also proposes various models for software defect prediction which can be used to indirectly gauge software reliability [3,4]. The primary drawback among static models can not effectively capture the software process and its variations during the course of software project development. On the other hand, a dynamic software reliability model is reflective of the software testing phase and is generally applicable before product release.

Among dynamic models, the (single-phase) Rayleigh model has been shown suitable to fit software defect arrival patterns [5,6]. A single-phase Rayleigh model divides the whole software development life cycle into six stages that are in chronological order: High Level Design (HLD), Low Level Design (LLD), CODING, Unit Testing (UT), Integration Testing (IT) and System Testing (ST). The six stages are assigned to a sequence of numerical scales. That is: HLD = 0.5, LLD = 1.5, CODING = 2.5, UT = 3.5, IT = 4.5 and ST = 5.5 [5]. Those numerical assignments seem rather *ad hoc*. Instead, we

could assign the six stages to, for instance, $\{t_1, t_2, t_3, t_4, t_5, t_6\}$, as long as $\{t_i\}_{i=1}^{6}$ satisfies $t_1 < t_2 < t_3 < t_4 < t_5 < t_6$. With different numerical assignments for the six stages, the fitted single-phase Rayleigh models could show a much different accuracy pattern, as shown in **Figure 1**.

The defects/KLOC in **Figure 1** is reconstructed from the work of Thangarajan *et al*. [5]. The quadratic fit is shown to illustrate that the small pairs of data could be fitted well by an arbitrary model such as quadratic model, rather than just single-phase Rayleigh model. Also, by assigning one numerical number to each stage, the data set now contain only six pairs at most. For prediction purposes, most likely 3, 4 and 5 pairs available, such a small sample size offers no confidence in the reliability prediction.

During the software development life cycle, collecting one single representative number for each stage results in a very small sample size. Furthermore, it is more likely that the data of major software defects are followed weekly, hence allowing project management to monitor the dynamic progress of the software development process. Our motivated weekly software development defects data set, **Figure 2**, shows the serious inadequacy of the single-phase Rayleigh model. This leads to our research on developing a better dynamic software reliability model to estimate the number of major defects, hence predict software release date.

The existing organizational reliability prediction model for software release date prediction at a large software company, where the weekly data in **Figure 2** were collected, is the dynamic single-phase Rayleigh model [5,6]. The software process in the organization consisted of two or more development phases. This is due to the software production cycles, availability of supporting hardware (e.g. wingboard/test phones) in the earlier software development stages, man-power management (e.g. testers' rearrangement) during the software development phases, and other dynamic issues during development. **Figure 2**, for instance, shows that the scatter plot overlaid with the single and the newly proposed two-phase truncated (piecewise, for short) Rayleigh models for the data set from the large software company. It is clear that the two-phase truncated Rayleigh model fits the data much better than the single-phase Rayleigh model.

Motivated by the example, we propose a multiple-phase truncated Rayleigh model in this paper. Such a model is better suited to fit the weekly defect arrival patterns during software development process. For simplicity reasons, we focus on the two-phase truncated (piecewise) Rayleigh model. The model can be extended to include additional phases reflecting the development process. It is shown through empirical modeling that the model accuracy is significantly improved. Furthermore, using the two-phase truncated Rayleigh model, the release date is predicted with a much higher confidence level.

The paper is organized as follows: Section 2 summarizes the single-phase Rayleigh model and proposes the multi-phase model, with a focus on the two-phase truncated Rayleigh model. Section 3 presents the algorithms of nonlinear least squares estimators of the model parameters and flowcharts of the dynamic process. Section 4 applies the proposed two-phase truncated Rayleigh model to defect arrival data of a large real-world software project from the large software organization. Finally, Section 5 concludes the paper and provides suggestions for future work.

2. Multi-Phase Truncated Rayleigh Models for Software Reliability Prediction

The dynamic single-phase Rayleigh model is a standard technique for software reliability modeling, and has been widely used for the software project and quality management in the software industry. The software organization, from which our case study data is obtained, has utilized the dynamic single-phase Rayleigh model for several of their previous software project developments.

The single-phase Rayleigh model is a parametric regression model with the regression function specified by the Rayleigh distribution with a multiplier coefficient. When the parameters of the Rayleigh distribution are estimated based on the updated data from a software project, dynamic projections about the number of defects for the software can be made based on the model over the software development life cycle.

The Rayleigh distribution is a special case of Weibull distribution, and has various applications including reliability estimation and life cycle pattern modeling [7,8] in developing software projects, life testing experiments in clinical studies dealing with cancer patients [9]. We now summarize the Rayleigh distribution. Denote t_m be the time at which the single-phase Rayleigh density curve reaches its peak. The cumulative distribution function of Rayleigh distribution with the constant multiplier K (the total number of latent defects) is

$$F(t; K, \theta) = K\left[1 - e^{-\theta t^2}\right],$$

where $\theta = 1/(2t_m^2)$ is the scale parameter. The single-phase Rayleigh model has a regression function parameterized as,

$$f(t; K, \theta) = 2K\theta t e^{-\theta t^2} \quad (1)$$

where both K and θ are the two parameters that need to be estimated using the data.

The single-phase Rayleigh model (1) does not fit the

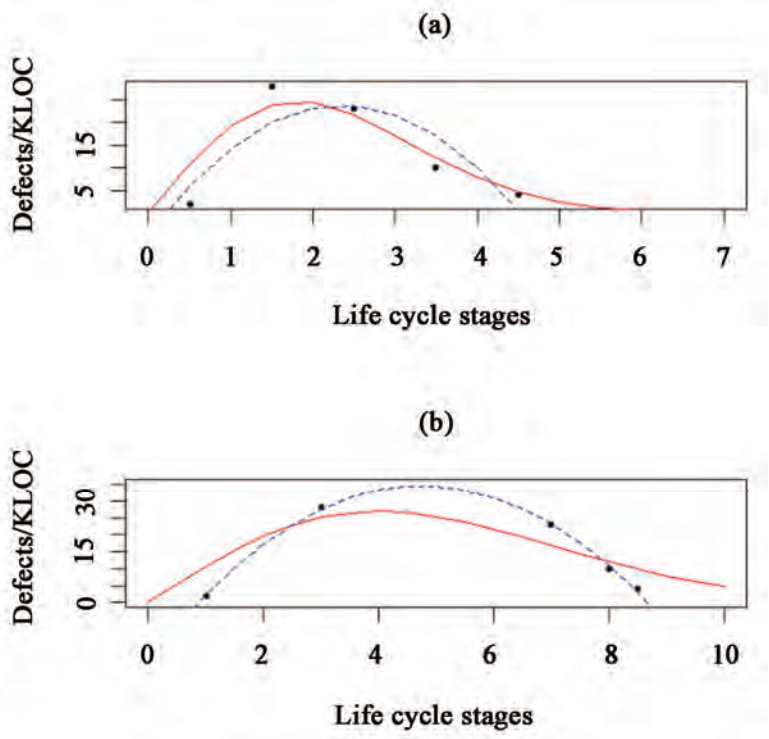


Figure 1. Single-Rayleigh model vs. quadratic model for two *ad hoc* numerical assignments for the ordinal stages in the software development life cycle. Solid line is for the single-phase Rayleigh model, while the dashed line is for the quadratic model. (a) (HLD,LLD, CODING, UT, IT, ST) = (0.5,1.5,2.5,3.5,4.5,5.5); (b) (HLD,LLD, CODING, UT, IT, ST) = (1,3,7,8,8.5,9)

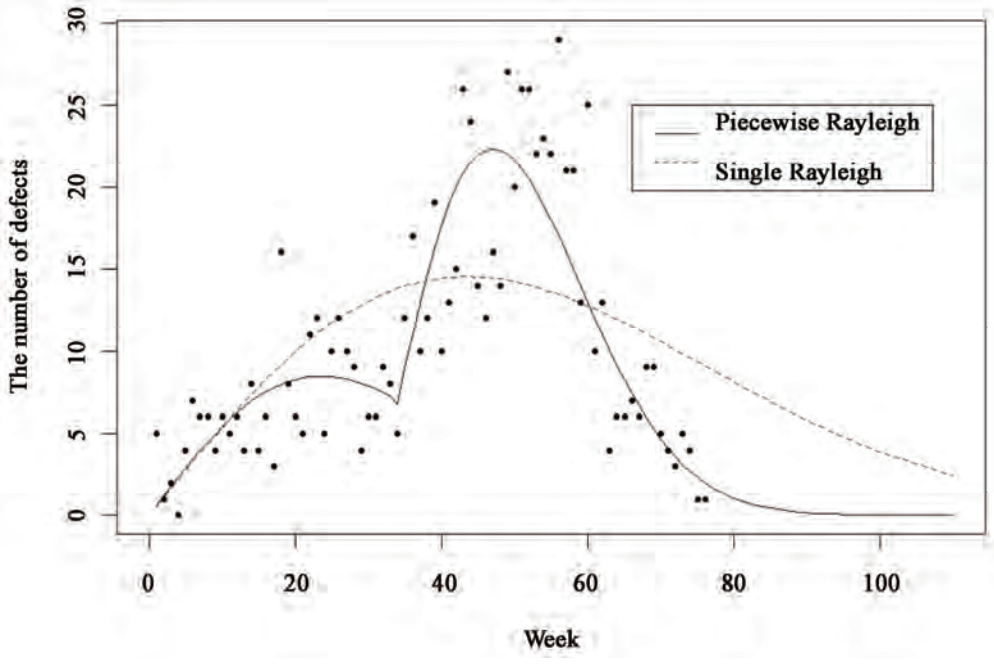


Figure 2. Major defects vs. development time in weeks

case study data set well. Actually it is a very poor fit as seen in **Figure 2**, and makes the case for a much needed improvement in modeling software defect arrival patterns. We propose a new multi-phase truncated Rayleigh model

defined as below:

$$g(t;\alpha)=\begin{cases} f(t;K_1,\theta_1), & 0\le t\le \tau_1 \\ f(t-\nu_1;K_2,\theta_2), & \nu_1\le\tau_1<t\le\tau_2 \\ \cdots & \\ f(t-\nu_{d-1};K_d,\theta_d), & \nu_{d-1}\le\tau_{d-1}<t<\infty, \end{cases}$$

where d is the number of phases and $\alpha^T=(\theta_1,\ldots,\theta_d,K_1,\ldots,K_d,\nu_1,\ldots,\nu_{d-1},\tau_1,\ldots,\tau_{d-1})$ is the model parameter vector. For simplicity, we will discuss the case with $d = 2$, the two-phase truncated (piecewise) Rayleigh model with regression function parameterized as follows:

$$g(t;\alpha)=\begin{cases} f(t;K_1,\theta_1), & 0\le t\le\tau \\ f(t-\nu;K_2,\theta_2), & \nu\le\tau<t, \end{cases} \quad (2)$$

where τ is the location of the phase change, ν is the starting location for the second phase. Due to the nature of the software defect data, we suggest to use the left truncated Rayleigh model for the second phase. Then $\alpha^T=(\theta_1,\theta_2,K_1,K_2,\nu,\tau)$ is the parameter vector, need to be estimated.

3. Algorithms for Piecewise Rayleigh Models

In this section, we describe the nonlinear least squares estimator of the model parameters. Let $\{(t_i,d_i)\}_{i=1}^n$ be the defect arrival data collected over time, where $t_i=i/n$ is the time index for the ith week, d_i is the total number of software defects detected during the ith week, and n is the number of weeks observed. Let

$$S(\alpha)=\sum_{i=1}^{n}\left[d_i-g(t_i;\alpha)\right]^2.$$

Then the nonlinear least squares estimator is the minimization of $S(\cdot)$. Notice that $S(\cdot)$ is not differentiable in the location of phase change point τ and the starting point of second phase ν. In conjunction with nonlinear least squares method and Gauss-Newton algorithm, we utilize a four-step technique (described below) to obtain the estimators of the parameter vector α. The package *nls* in **R** language is used to obtain the estimates of the model parameters.

Step 1: For any given location of phase change τ in (0, 1), fix a ν such as $0<\nu\le\tau<1$, we compute the nonlinear least squares estimators [10], $\tilde{\alpha}_{1n}(\nu,\tau)$ for the smooth parameters $\alpha_1^T=(\theta_1,\theta_2,K_1,K_2)$, by minimizing $S(\alpha)$ over α_1.

Step 2: Substitute $\tilde{\alpha}_{1n}(\nu,\tau)$ into $S(\alpha)$ to obtain the profile objective function, $\tilde{S}(\nu,\tau)$. Then we minimize $\tilde{S}(\nu,\tau)$ over $0<\nu\le\tau<1$ for the given τ to obtain $\tilde{\nu}(\tau)$. Notice that the minimizer $\tilde{\nu}(\tau)$ is a function of τ.

Step 3: Substitute $\tilde{\nu}(\tau)$ into $\tilde{S}(\nu,\tau)$ to get $\tilde{S}(\tau)$. The minimizer of $\tilde{S}(\tau)$ over $\tau\in(0,1)$ is called the change point estimator, denoted by $\hat{\tau}$.

Step 4: Substitute $\hat{\tau}$ into $\tilde{\nu}(\tau)$ to get $\hat{\nu}$ and $\tilde{\alpha}_{1n}(\hat{\nu},\hat{\tau})$ to get $\hat{\alpha}_{1n}$. Put them together, we obtain the nonlinear least squares estimator, $\hat{\alpha}_n^T=(\hat{\alpha}_{1n}^T,\hat{\nu},\hat{\tau})$ of α.

Figures 3 and **4** illustrate the flow charts of the dynamic process of the algorithm for single-phase and multi-phase truncated Rayleigh models, respectively. We provide the flowchart for the single-phase Rayleigh model for comparison purpose.

4. Application to a Real Software Defect Data Set

The data set motivated our research were collected from Feb-25-06 to Aug-04-07 at a large software company. There are 76 weeks software defects arrival data. Number of major defects during a week is reported.

4.1 Single-phase vs. Piecewise Rayleigh Models

We illustrate the two-phase truncated Rayleigh model by fitting the software defect arrival data set. From **Figure 2**, it is observed that using two-phase truncated Rayleigh model improves the model fitting significantly compared to the single-phase Rayleigh model with respect to model accuracy and model goodness-of-fit. For comparison

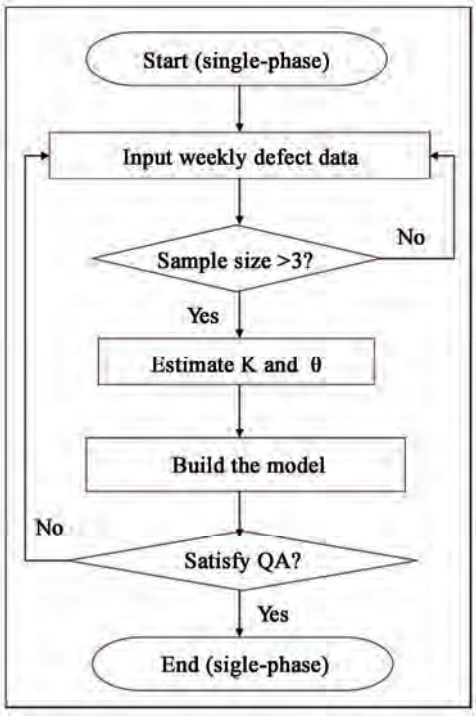


Figure 3. Algorithm for single-phase Rayleigh model

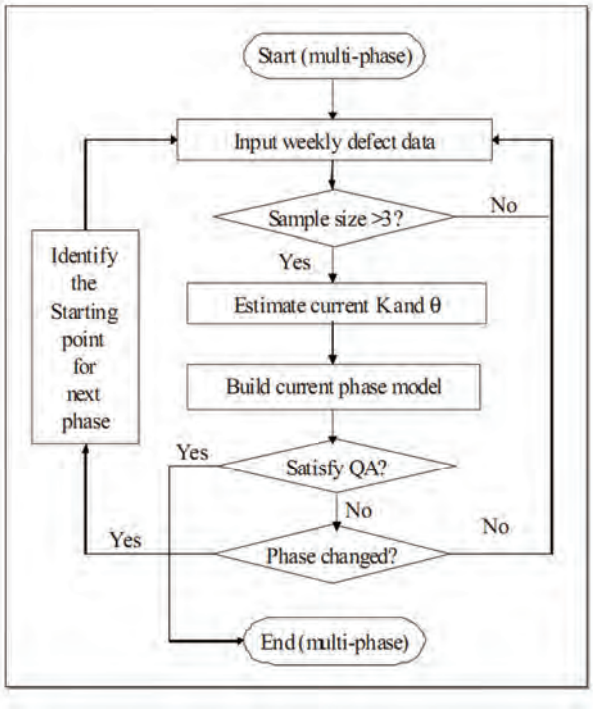


Figure 4. Algorithm for multi-phase Rayleigh model

purpose, the estimated single-phase Rayleigh regression function is given by,

$$\hat{g}(t) = 2\hat{K}\hat{\theta}te^{-\hat{\theta}t^2},$$

where $\hat{K} = 13.7111$ and $\hat{\theta} = 1.5297$.

For the two-phase truncated Rayleigh model (2), the estimated change is at the $\hat{\tau}$ = 33*rd* week with the starting point estimated at $\hat{\nu}$ = 31*st* week for the second phase. Hence phase one is from the first week to 33*rd* week and phase two is from the 34th week to the 76th week with estimated starting point at $\hat{\nu}$ = 31*st* week. The estimated first phase (right truncated) of the regression function is estimated as

$$\hat{g}(t) = 2\hat{K}_2\hat{\theta}_2(t-31/76)e^{-\hat{\theta}_2(t-31/76)^2},$$
$$\text{if } t > 33/76,$$

with $\hat{K}_1 = 4.3022, \hat{\theta}_1 = 5.2279,$ and the second phase (left truncated) of the regression function is estimated as

$$\hat{g}(t) = 2\hat{K}_2\hat{\theta}_2(t-31/76)e^{-\hat{\theta}_2(t-31/76)^2},$$
$$\text{if } t > 33/76,$$

with $\hat{K}_2 = 7.7731, \hat{\theta}_2 = 11.2445$. **Figure 2** shows the scatter plot overlaid with the two fitted curves using the single-phase and two-phase truncated Rayleigh models, respectively. From the fitted model, one can predict the future week's number of software defects and establish the quality assurance criterion and management for predicting the release date.

This proposed multi-phase truncated Rayleigh model can be utilized for modeling any future software development projects to obtain better prediction and provide more efficient estimation of the release date of the software product.

4.2 Quality Assurance Criterion for Release Date Prediction

In this section, we establish the quality assurance criterion for software release. The quality assurance criterion is determined by 95% and 99.9% confidence levels. That is, based on the fitted model, if the model shows that 95% or 99.9% of the total expected software defects has been detected, then we suggest that the software is ready for release.

For the single-phase Rayleigh model, we estimate the release date with 95% confidence level. We set $F\left(t;\hat{K},\hat{\theta}\right) = 0.95\hat{K}$ and solve for *t* or equivalently

$$1 - e^{-\hat{\theta}t^2} = 0.95.$$

This implies that the release date equals to the ceiling of $n * \sqrt{-\ln(1-.95)/\hat{\theta}} = 107$ weeks, where $\hat{\theta}$=1.5297. Hence, with 95% confidence the software project will

need 107−76 = 31 weeks of further testing before releasing the software product. That is the predicted release date using the 76 weeks of data is Feb-29-08 based on the single-phase Rayleigh Model. With 99% confidence, it will require even much longer testing time.

Alternatively, utilizing the two-phase truncated Rayleigh model (2), we set

$$\int_0^{\hat{\tau}/n} \hat{g}(t)dt + \int_{\hat{\tau}/n}^{i/n} \hat{g}(t)dt = 0.999\int_0^1 \hat{g}(t)dt$$

and solve for i to get the estimated release week with 99.9% confidence level. Equivalently, the estimated release week number, i, satisfies that

$$e^{-\hat{\theta}_2\left(\frac{i-\hat{\nu}}{n}\right)^2} = e^{-\hat{\theta}_2\left(\frac{\hat{\tau}-\hat{\nu}}{n}\right)^2} - \frac{\left[0.999\int_0^1 \hat{g}(t)dt - \int_0^{\hat{\tau}/n} \hat{g}(t)dt\right]}{\hat{K}_2} = A_0 - \frac{0.999A - A_1}{\hat{K}_2}, \quad (3)$$

where

$$A_0 = e^{-\hat{\theta}_2\left(\frac{\hat{\tau}-\hat{\nu}}{n}\right)^2} = 0.9922,$$

$$A_1 = \int_0^{\hat{\tau}/n} \hat{g}(t)dt = 2.6967,$$

$$A = \int_0^1 \hat{g}(t)dt = 10.2586.$$

Solving Equation (3) to obtain the estimated release week number:

$$i = \hat{\nu} + \left\{ n * \sqrt{-\left(1/\hat{\theta}_2\right)\ln\left(A_0 - \frac{0.999A - A_1}{\hat{K}_2}\right)} \right\} = 76 \text{ weeks},$$

where $\{x\}$ is the smallest integer greater or equal to x. This indicates that with 99.9% confidence that the estimated release week is the end of 76th week. That is the software is ready for release, with almost 100% confidence based on the two-phase truncated Rayleigh model. We note that the large software organization has adopted our new two-phase truncated Rayleigh model and is using it to predict the number of software defects dynamically and release dates for ongoing software projects. Our new two-phase truncated Rayleigh model has improved the software release life cycle a great deal and has saved a lot of man-powered resource for the large software organization.

4.3 Model Performance Check

We utilize three measures of goodness-of-fit to assess the performance of the models: root mean square error (RMSE), magnitude of relative error (MRE), and adjusted coefficient of determination R^2_{adj}. The root mean square error measures the model accuracy defined as the square root of mean squared residuals. That is,

$$RMSE = \sqrt{\frac{1}{n-5}\sum_{i=1}^{n}\left(d_i - \hat{d}_i\right)^2},$$

where d_i is the number of defects detected during the ith week, $\hat{d}_i$ is the fitted (predicted) value of d_i. The smaller the RMSE, the better the model fits.

The second criterion for assessment of the performance of model fitting used in the reliability literature is the mean magnitude of relative error, defined as

$$MRE = \frac{\sum_{i=1}^{n}\left|\frac{d_i - \hat{d}_i}{d_i}\right| I(d_i > 0)}{\sum_{i=1}^{n} I(d_i > 0)}.$$

The implicit assumption in this summary measure is that the seriousness of the absolute error is proportional to the size of the observations. The smaller the MRE, the better the model fits.

The third measure of goodness-of-fit used is the adjusted determination of coefficient R^2_{adj} which is the adjusted percentage of variation in the number of defects per week explained by the model. That is

$$R^2_{adj} = 1 - \frac{SSE/(n-5)}{SSTO/(n-1)},$$

where $SSE = (n\text{-}5)(RMSE)^2$ and

$$SSTO = \sum_{i=1}^{n}\left(d_i - \bar{d}\right)^2 \text{ with } \bar{d} = \sum_{i=1}^{n} d_i \Big/ n.$$

The higher of the R^2_{adj}, the better the model fits.

Table 1 summarizes the three performance criteria for the real-world weekly software defects data set using both single-phase and two-phase truncated (piecewise) Rayleigh models. Based on the reported *RMSE*, *MRE* and R^2_{adj} values, the two-phase truncated Rayleigh model is much better than the single-phase Rayleigh model. The *MRE* is reduced by about 50%, while the goodness-of-fit measure R^2_{adj} is roughly doubled for the two-phase truncated compared to the single-phase Rayleigh models. The two-phase truncated Rayleigh model explains the almost doubled variation in the number of defects than the single-phase Rayleigh model does. Thus, based on the given data, we conclude that the two-phase truncated Rayleigh model is an attractive model for predicting weekly software defects and release date of software projects.

Table 1. Model comparisons using RMSE, MRE and R^2_{adj}

	Criterion		
Model	RMSE	MRE	R^2_{adj}
Single-phase	5.97	0.76	36.6%
Two-phase	4.13	0.36	70.4%

5. Conclusions

The research was motivated by a real-world software defect arrival data over many weeks from a large software organization. The paper proposes a new multi-phase truncated (focusing on a two-phase truncated model) Rayleigh model in fitting weekly defect arrival data.

It is shown that the proposed model is much more accurate than the existing single-phase Rayleigh model. The single-phase model was previously used by the organization during software development. Using both MRE and R^2_{adj} performance measures, the proposed model almost doubled the prediction accuracy, hence, shortening the release date prediction with a higher confidence level. From a software reliability perspective, our proposed two-phase truncated Rayleigh prediction model will help in the management and planning of project resources toward bettering the software release cycle time.

The two-phase truncated Rayleigh model can be easily extended to a multi-phase truncated Rayleigh model. Hence it can be used to predict release date for future software projects with a higher confidence level. A general multi-phase Rayleigh software release prediction model can be developed to automatically detect and reflect all the change locations and the starting points of the software development phases so that the multiple-phase truncated Rayleigh software prediction model can be generated to automatically forecast the software release time.

REFERENCES

[1] M. R. Lyu, "Software Reliability: To Use or not to Use?" *Proceedings of* 5*th International Symposium on Software Reliability Engineering*, 66-73 November 1994.

[2] Y. Wang and M. Smith, "Release Date Prediction for Telecommunication Software Using Bayesian Belief Networks," *Proceedings of the* 2002 *IEEE Canadian Conference on Electrical and Computer Engineering*, 2002, pp. 738-742.

[3] T. M. Khoshgoftaar and N. Seliya, "Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques," *Empirical Software Engineering Journal*, Vol. 8, No. 3, 2003, pp. 255-283.

[4] T. M. Khoshgoftaar and N. Seliya, "Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study," *Empirical Software Engineering Journal*, Vol. 9, No. 3, 2004, pp. 229-257.

[5] M. Thangarajan and B. Biswas, "Mathematical Model for Defect Prediction across Software Development Life Cycle," *The SEPG* (*Software Engineering Process Group*) *Conference*, India, 2000. http://www.qaiindia.com/Conferences/SEPG2000/index.html

[6] S. H. Kan, "Metric and Models in Software Quality Engineering," 2nd Edition, Addison Wesley, Massachusetts, 2003.

[7] P. V. Norden, "Useful Tools for Project Management," *Operations Research in Research and Development*, B. V. Dean, Ed., John Wiley & Sons, New York, 1963.

[8] L. H. Putman, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transaction on Software Engineering*, Vol. SE-4, 1978, pp. 345-361.

[9] S. K. Bhattacharya and R. K. Tyagi, "Bayesian Survival Analysis Based on the Rayleigh Model," *Trabajos de Estadistica*, Vol. 5, No. 1, 1990, pp. 81-92.

[10] D. M. Bates and J. M. Chambers, "Nonlinear Models," Chapter 10 of Statistical Models in S. J. M. Chambers and T. J. Hastie, Eds., Wadsworth & Brooks/Cole, 1992.

J. Software Engineering & Applications, 2010, 3, 610-613
doi:10.4236/jsea.2010.36071 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

An Optimal Shape Design Problem for Fan Noise Reduction

Bahram Farhadinia

Department of Math, University of Mohaghegh Ardabili, Ardabil, Iran.
Email: farhadinia@uma.ac.ir

Received December 24th, 2009; revised April 10th, 2010; accepted April 12th, 2010.

ABSTRACT

The objective of the present article is to find an optimal design of a fan inlet to reduce the amount of noise radiated to the far field from the system. Against the gradient-based optimization algorithms, we employ here a method based on measure theory which does not require any information of gradients and the differentiability of cost function.

Keywords*: *Shape Optimization, Noise Radiation, Measure Theory, Linear Programming

1. Introduction

Noise generated by the rotating blades of an aircraft turbofan engines continues to pose a challenge for many researchers and of course, the reduction of the noise have been received more attention in the recent studies. Williams and Hall studied aerodynamic sound generation by turbulent flow around a scattering half plane [1]. Howe in [2] discussed the trailing edge noise at low Mach numbers. It is well known that the aircraft noise may be minimized through the two signification tasks: acoustic shape optimization of the inlet [3] and impedance optimization of the liner [4]. In much of the pervious work on this topic, the computational methods are gradient-based that bear some drawbacks such as mesh regularization and the requirement of knowledge of derivatives.

The aim of this article is to present a method to find an optimal solution of a shape design problem that models reducing the amount of noise radiated from aircraft turbofan engines.

Our method is based on measure theory and has some features, for instance, the design process is not iterative and therefore requires no initial design to be suggested; it is computationally efficient and flexible enough to accommodate general design problem.

The paper proceeds as follows. In Section 2 we first describe the mathematical formulation of the optimal control problem and then a brief review of the weak formulation of the problem is given. In Section 3, we metamorphose the optimization problem to a linear programming problem by the aid of our method. Finally, we present some numerical results in Section 4.

2. Problem Description and Weak Formulation

We follow here the same set up as considered in [4] in where the geometry of the axisymmetric fan is as shown in **Figure 1**. We suppose that the model composition of the aircraft noise source is specified on the source plane Γ_f. The inlet of fan is surrounded by two boundaries: fixed boundary Γ_c and flexible boundary Γ_α which is characterized by the function $y = \alpha(x)$. It is assumed an acoustic liner exists on the boundary Γ_c. We let Γ_∞ be enough far from the noise source. This implies that the radiated field treats locally as a plane wave at local incidence and in this case the Sommerfeld radiation boundary condition is satisfied. If we describe the treatment of the acoustic velocity potential u with the Helmholtz equation as

$$\Delta u + k^2 u = 0, \quad on\ \Omega,$$

Subject to the boundary conditions

$$u|_{\Gamma_f} = g(\alpha),$$

$$\frac{\partial u}{\partial n}|_{\Gamma_\alpha} = 0,$$

$$\frac{\partial u}{\partial n}|_{\Gamma_a} = 0,$$

$$(u + x\frac{\partial u}{\partial n})|_{\Gamma_c} = 0,$$

$$(iku + x\frac{\partial u}{\partial n})|_{\Gamma_\infty} = 0,$$

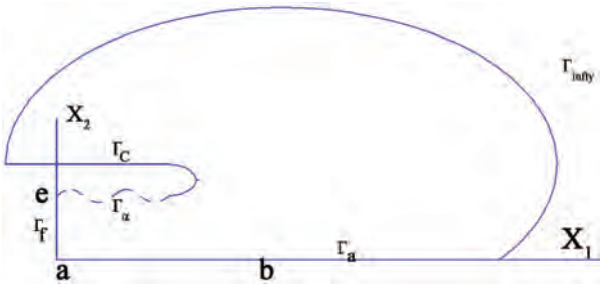


Figure 1. Geometry of the axisymmetric fan

then, the problem here is the manipulation of the flexible boundary Γ_α such that the least amount of noise propagates to the far field whereas some constraints on the boundary shape are satisfied. It is supposed that in the above equations, both the dependent and the independent variables are properly nondimensionalized.

Under consideration in [4], the optimal shape design problem may be stated as

$$
\begin{aligned}
(\mathrm{P}) \quad \min\{J &= A\int_\Omega u^2 d\Omega + B\int_\Omega |\nabla u|^2 \, d\Omega + \lambda\int_a^b (\alpha(x_1)-\alpha_0(x_1))^2 dx_1;\\
&\Delta u + k^2 u = 0, \quad on \quad \Omega,\\
&u|_{\Gamma_f} = g(\alpha),\\
&\frac{\partial u}{\partial n}|_{\Gamma_\alpha} = 0,\\
&\frac{\partial u}{\partial n}|_{\Gamma_a} = 0,\\
&(u + x\frac{\partial u}{\partial n})|_{\Gamma_c} = 0,\\
&(iku + x\frac{\partial u}{\partial n})|_{\Gamma_\infty} = 0\},
\end{aligned}
$$

where $x > 0$ and the three constants A, B and λ satisfy $A^2 + B^2 > 0, \quad \lambda \geq 0$.

Obviously the determination of domain of the problem, Ω, depends on the determination of $\alpha : [a,b] \to [c,d]$, which is a function to be determined by the optimization process. By virtue of this fact we let $\Omega = \Omega(\alpha)$.

In order for the optimal shape be exist, we define the admissible set of functions, α 's, as

$$
A_{ad} = \{\alpha \in C([a,b]); c \leq \alpha \leq d, |\alpha(t_1) - \alpha(t_2)| \leq \beta |t_1 - t_2|\}. \quad (1)
$$

3. Metamorphosis and Approximation

If we define $\varphi = u - u_0$ and $V_\alpha = \{u \in H^1(\Omega(\alpha)); u|_{\Gamma_f} = 0\}$, then the variational formulation of (P) with homogeneous boundary condition may be considered as (see [4])

$$
\begin{aligned}
(p_V) &\min\{J(\alpha,\phi) =\\
&A\int_{\Omega(\alpha)} |\phi + u_0|^2 \, d\Omega + B\int_{\Omega(\alpha)} |\nabla(\phi + u_0)|^2 \, d\Omega + \lambda\int_a^b (\alpha(x_1) - \alpha_0(x_1))^2 dx;\\
&(\nabla\phi, \nabla v) - k^2(\phi, v) + \langle x\phi, v\rangle_{\Gamma_c(\alpha)} + i\langle k\phi, v\rangle_{\Gamma_\infty(\alpha)} = (f, v), \forall v \in V_\alpha\}.
\end{aligned}
$$

In this portion, we define an artificial control $w(x)_1 : [a,b] \to R$ satisfying

$$
\begin{aligned}
&\frac{d}{dx_1}\alpha(x_1) = w(x_1), \forall x_1 \in [a,b],\\
&\alpha(a) = \alpha(a) = e.
\end{aligned} \quad (2)
$$

In fact, w is derivative of the boundary function Γ_α. Let $x_s : S \to \{0,1\}$ be the characteristic function defined on the set S.

We may now reformulate (p_v) involving the artificial control as follows:

$$
\begin{aligned}
(\hat{P}) \quad &\min\{J(\alpha,\varphi) = \int_{\Omega(\alpha)} (A|\varphi + u_0|^2 + B|\nabla(\varphi + u_0)|^2 + \lambda x_{D_1}) d\Omega;\\
&\int_{\Omega(\alpha)} (\nabla\varphi\overline{\nabla v} - k^2\varphi\overline{v} - f\overline{v}) d\Omega + \int_{\partial\Omega(\alpha)} x\varphi\overline{v} x_{\Gamma_{c(\alpha)}} + ik\varphi\overline{v} x_{\Gamma_{\infty(\alpha)}} \, d\Gamma = 0,\\
&\forall v \in V_\alpha \; \frac{d}{dx_1}\alpha(x_1) = w(x_1), \; \forall x_1 \in [a,b] \;\; \alpha(a) = \alpha(a) = e\},
\end{aligned}
$$

where $D_1 = \{(x_1, x_2) : x_1 \in [a,b], 0 \leq x_2 \leq (\alpha(x_1) - \alpha_0(x_1))^2\}$.

In order to apply the metamorphosis process, one needs first to have some background knowledge.

We say a triple $q = (w, \alpha, \varphi)$ is admissible if the following conditions yield: (i) The artificial control w takes its values in a compact set W and holds (2). (ii) The pair (α, φ) is the solution of the variational formulation of (p_v). We denote the set of all admissible triples by Q.

Let $\gamma_1 = \hat{\Omega} \times S_w \times S_\varphi \times S_{\varphi'} \times S_\alpha$ and $\gamma_2 = \partial\hat{\Omega} \times S_\varphi \times S_\alpha$ where all these sets are compact and $w, \varphi, \nabla\varphi$ and α get their values in $S_w, S_\varphi, S_{\varphi'}$ and S_α, respectively. Also $\hat{\Omega}$ is the fixed domain depicted in **Figure 2** and yields $\bigcup_{\alpha \in A_{ad}} \Omega(\alpha) \subset \hat{\Omega}$.

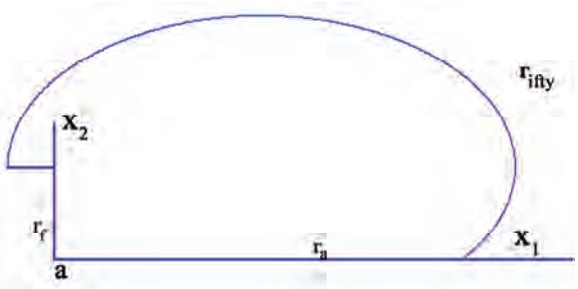


Figure 2. Geometry of the fixed domain $\hat{\Omega}$

Now we transfer $(\hat{P})$ over Q, into another nonclassical problem that has some interesting properties.

Define two linear, bounded and positive functionals corresponding to each admissible triple q as

$$\Lambda_{q,1}: F \in C(\gamma_1) \to \int_{\Omega(\alpha)} F(x_1, x_2, w, \varphi, \nabla\varphi, \alpha)\, d\Omega, \quad (3)$$

$$\Lambda_{q,2}: G \in C(\gamma_2) \to \int_{\partial\Omega(\alpha)} G(x_1, x_2, \varphi, \alpha)\, d\Gamma, \quad (4)$$

where $C(\gamma_i)$ is the space of all continuous real-valued functions on $\gamma_i, i = 1,2$.

By the use of the latter definition, one can observe that there exists an injection of Q into $C^*(\gamma_1) \times C^*(\gamma_2)$, the product of dual spaces of functionals defined in (3)-(4). (See Proposition 4.1 in [5]).

By Riesz representation theorem [6], corresponding to each $\Lambda_{q,i}$, there is a unique positive Radon measure μ_i, such that

$$\Lambda_{q,1}(F) = \int_{\Omega(\alpha)} F(x_1, x_2, w, \varphi, \nabla\varphi, \alpha)\, d\Omega = \mu_1(F), \quad \forall F \in C(\gamma_1), \quad (5)$$

$$\Lambda_{q,2}(G) = \int_{\partial\Omega(\alpha)} G(x_1, x_2, \varphi, \alpha)\, d\Gamma = \mu_2(G), \quad \forall G \in C(\gamma_2). \quad (6)$$

Generally, the metamorphosis process deals with integral form of relations, therefore, in order to convert the differential Equation (2) into an integral form, let B be an open ball in R^2 containing $\hat{\Omega}$, and $C^1(B)$ be the space of all real-valued continuous differentiable functions with continuous first partial derivatives on B. Let $\psi \in C^1(B)$ and define

$$\psi^w(x_1, x_2, w, \alpha) = \psi_{x_2}(x_1, \alpha) w(x_1) + \psi_{x1}(x_1, \alpha)$$

for each $(x_1, x_2, w, \alpha) \in \Omega_1 = [a,b] \times [c,d] \times S_w \times S_\alpha$.

Thus for each admissible triple q the system (2) is equivalent with

$$\int_a^b \psi^w(x_1, x_2, w, \alpha) dx_1 = \int_a^b (\psi_{x_2}(x_1, \alpha) w(x_1) + \psi_{x1}(x_1, \alpha)) dx_1 = \psi(b, \alpha(b)) - \psi(a, \alpha(a)) = \Delta\psi. \quad (7)$$

Now for simplifying the representation of optimal shape design problem, we set

$$f_0(\alpha, \varphi) = A \mid \varphi + u_0 \mid^2 + B \mid \nabla(\varphi + u_0) \mid^2 + \lambda x_{D_1},$$

$$F_1(\alpha, \varphi, v) = \nabla\varphi \overline{\nabla v} - k^2 \varphi \overline{v} - f\overline{v},$$

$$F_2^{\psi}(\alpha, w) = \psi^w x_{D_2},$$

$$G_1(\alpha, \varphi, v) = x\varphi \overline{v} x_{\Gamma_{c(\alpha)}} + ik\varphi \overline{v} x_{\Gamma_{\infty(\alpha)}},$$

where $D_2 = \{(x_1, x_2) : x_1 \in [a,b], 0 \le x_2 \le \psi^w\}$.

By virtue of (5)-(6) and the above arrangement, we may restate $(\hat{P})$ by the following measure representation form:

$$(P_M) \quad \min\{\mu_1(f_0); \ \mu_1(F_1) + \mu_2(G_1) = 0, \quad \forall v \in V_\alpha, \ \mu_1(F_2^{\psi}) = \Delta\psi, \quad \forall \psi \in C^1(B)\}.$$

In what follows, we employ a two-phase approximation procedure that converts (P_M) to a linear programming (LP) problem. The procedure for constructing a LP problem whose solution approximates the action of the optimal pair of measures of (P_M) is much like as that given in [5]. In the first phase of approximation procedure, we intend to deal with a problem subject to a finite number of constraints. Following from Proposition 5.1 in [5], our intension is fulfilled. Approximating measures involved in (P_M) by the unitary atomic measures (see Proposition 5.2 in [5]) is performed in continuation of the first phase of approximation procedure and leads to appear unknown supports. The second phase is devoted to approximate the unknown supports by specified points. This task is carried out in cooperation with Proposition 5.3 in [5].

Following the mentioned procedure, the resulting LP is as

$$(LP) \quad \min\{\sum_{k=1}^{N_1} \xi_k^1 f_0(z_1^k); \ \sum_{k=1}^{N_1} \xi_k^1 F_1(z_1^k) + \sum_{k=1}^{N_2} \xi_k^2 G_1(z_2^k) = 0, \quad l = 1,2,..., m_1, \ \sum_{k=1}^{N_1} \xi_k^1 F_2^{\psi j}(z_1^k) = \Delta\psi_j, \quad j = 1,2,..., m_2, \ \xi_k^i \ge 0, \quad k = 1,2,..., N_i, \quad i = 1,2\}.$$

The process of extracting a piecewise constant control function w form the optimal solution $\{\xi_k^i \ge 0,\ k = 1,2,...,N_i,\ i = 1,2\}$ of (LP) is based on the analysis given in [7].

4. Numerical Result

The data for the parameters of the model are listed as:

$$[a,b] = [0,2.5],\ e = 1,\ S_w = [-1,1],\ S_\alpha = [0.6,1.2],$$
$$k = 8,\ A = B = \lambda = 1,$$
$$v_l = \sin(lx_1),\ l = 1,2,..., m_1 = 4,$$
$$\psi_j^w = x_1^j,\ j = 1,2,..., m_2 = 5,$$
$$u_0(x_1) = 0,\ \alpha_0(x_1) = 1.$$

By using a code written in MATLAB 7.1 for solving (LP), we obtained the optimal approximated Γ_α and its corresponding optimal artificial control w depicted in **Figure 3**.

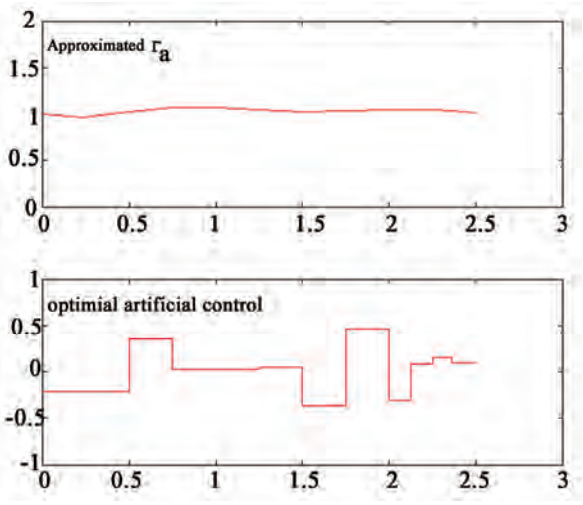


Figure 3. Optimal shape of the fan and optimal artificial control

5. Conclusions

This article discusses how to implement a numerical method based on measure theory for solving an optimal shape design problem for fan noise reduction. The proposed method has some advantages comparing to the gradient-based optimization methods. For instance, it does not require any information of gradients and the differentiability of cost function.

REFERENCES

[1] J. E. F. Williams and L. H. Hall, "Aerodynamic Sound Generation by Turbulent Flow in the Vicinity of a Scattering Half Plane," *Journal of Fluid Mechanics*, Vol. 40, No. 4, 1970, pp. 657-670.

[2] M. S. Howe, "Trailing Edge Noise at Low Mach Numbers," *Journal of Sound and Vibration*, Vol. 225, No. 2, 1999, pp. 211-238.

[3] A. L. Marsden, M. Wang and J. E. Dennis, "Constrained Aeroacoustic Shape Optimization Using the Surrogate Management Framework," Center for Turbulence Research, Annual Research Briefs, 2003.

[4] Y. Cao and D. Stanescu, "Shape Optimization for Noise Radiation Problems," *Computers and Mathematics with Applications*, Vol. 44, No. 12, 2002, pp. 1527-1537.

[5] B. Farhadinia, "Shape Optimization of a Nozzle with Specified Flow Field Including Viscosity Effect," *Acta Applicandae Mathematicae*, Vol. 104, No. 4, 2008, pp. 243-256.

[6] W. Rudin, "Real and Complex Analysis," McGraw-Hill, New York and London, 1970.

[7] J. E. Rubio, "Control and Optimization: The Linear Treatment of Nonlinear Problems," Manchester University Press, Manchester, 1986.

J. Software Engineering & Applications, 2010, 3, 614-627
doi:10.4236/jsea.2010.36072 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

An Interactive Method for Validating Stage Configuration

Abdelrahman Osman Elfaki, Somnuk Phon-Amnuaisuk, Chin Kuan Ho

Center of Artificial Intelligent and Intelligent Computing, Multimedia University, Cyberjaya, Malaysia.
Email: {abdelrahman.osman.06, somnuk.amnuaisuk, ckho}@mmu.edu.my

Received March 21st, 2010; revised April 6th, 2010; accepted April 8th, 2010.

ABSTRACT

Software product Line (*SPL*) *is an emerging methodology for developing software products. Stage-configuration is one the important processes applying to the SPL. In stage-configuration, different groups and different people make configuration choices in different stages. Therefore, a successful software product is highly dependent on the validity of stage-configuration process. In this paper, a rule-based method is proposed for validating stage-configuration in SPL. A logical representation of variability using First Order Logic* (*FOL*) *is provided. Five operations*: *validation rules, explanation and corrective explanation, propagation and delete-cascade, filtering and cardinality test are studied as proposed operations for validating stage-configuration. The relevant contributions of this paper are*: *implementing automated consistency checking among constraints during stage-configuration process based on three levels* (*Variant-to-variant, variant-to-variation point, and variation point-to-variation point*), *define interactive explanation and corrective explanation, define a filtering operation to guide the user within stage-configuration, and define* (*explicitly*) *delete-cascade validation.*

Keywords*: *Software Product Line, Variability, Stage Configuration

1. Introduction

Software Product Line (SPL) has proved to be an effective strategy to benefit from software reuse [1], allowing many organizations to reduce development costs and duration, meanwhile increase product quality [2]. It is an evolution from software reuse and Commercial Off-The-Shelf (COTS) methodologies.

Feature Model (FM) [3] appeals to many SPL developers as essential abstractions that both customers and developers understand. Customers and engineers usually speak of product characteristics in terms of those features the product has or delivers. Therefore, it is natural and intuitive to express any commonality or variability in terms of features [4]. Orthogonal Variability Model (OVM) is another successful approach proposed to document variability in SPL [5].

The principal objective for SPL is to configure a successful software product from domain-engineering process by managing SPL artifacts (variability modeling). Recently, in [6,7], validation is discussed as important issue in SPL community. Validating SPL intends to produce error-free products including the possibility of providing explanations to the modeler so that errors can be detected and eliminated. Usually, medium SPL contains thousands of features [2]. Therefore, validating SPL represent a challenge because it's a vital process and non-feasible to done manually.

The lack of a formal semantics and reasoning support of FM has hindered the development of validation methods for FM [8]. The automated validation of FM was already identified as a critical task in [9-11]. However, there is still a lack of an automated support for FM validation.

The configuration is a task of selecting a valid and suitable set of features for a single system, and it can become very complicated task [12]. Supporting user's needs in generating a valid and suitable solution is the basic functionality of a configuration system. In SPL, selecting a solution from domain-engineering process to use it in application-engineering process is the meaning of a configuration. As a conclusion, FM represents the configuration space of a SPL. An application-engineer may specify a member of a SPL by selecting the desired features from the FM within the variability constraints defined by the model, e.g., the choice of exactly one feature from a set of alternative features. In stage configuration, different groups and different people make configu-

ration choices in different stages [13].

2. Preliminaries

2.1 Software Product Line

SPL has been defined by Meyer and Lopez as a set of products that share a common core technology and address a related set of market applications [14]. SPL has two main processes; the first process is the domain-engineering process that represents domain repository and is responsible for preparing domain artifacts including variability. The second process is the application-engineering that aims to consume specific artifact, picking through variability, with regards to the desired application specification. The useful techniques to represent variability are FM and OVM. SPL has various members. A particular product-line member is defined by a unique combination of features (if variability modeled using FM) or a unique combination of variants (if variability modeled using OVM). The set of all legal features or variants combinations defines the set of product line members.

2.2 Feature Model

FM [3] is considered as one of the well-known methods for modeling SPLs [9]. According to Czarnecki and Eisenecker [12] the two most popular definitions of FM are 1) an end user visible characteristic of a system 2) a distinguishable characteristic of a concept (e.g., system, component, and so on) that is relevant to some stakeholders of the concept. Features in FM represent essential abstractions of the SPL (to both developers and customers). Customers and developers usually speak of product characteristics in terms of those features the product has or delivers. Therefore, it is natural and intuitive to express any commonality or variability in terms of features [4]. A FM is a description of the commonalities and differences between the individual software systems in a SPL. In more detail, a FM defines a set of valid feature combinations. The set of all legal features combinations defines the set of product line members. Each of such valid feature combinations can be served as a specification of a software product [12,15].

A feature model is a hierarchically structure of features and consists of: 1) relationships between a parent feature and its child features, and 2) dependency constraints rules between features, which are inclusion or exclusion. Czarnecki *et al.* [13] defined Cardinality-based feature modeling by integrating a number of extensions to the original FODA notation. According to [13] there are three different versions of FM: basic FM, cardinality FM, and extended FM. **Figure 1** illustrates basic FM. **Figure 1** is borrowed from [10]. In any type of FMs there are there are some common properties such as (examples are based on **Figure 1**):

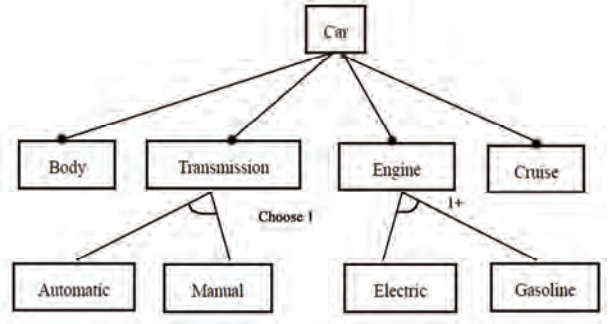


Figure 1. A car software product line represented by basic feature model

Parent Feature: This feature could be described as a decision point, in which some choices or decisions should be taken. A parent feature contains one or more of child feature(s). As example, transmission is a parent feature.

Child Feature: A feature belongs to parent feature is called a child feature, for instance, “Electric” is a child feature belongs to the parent feature “Engine”.

Common Feature: In a common relationship (between parent and child features), a child feature follows his parent feature in any product. For instance, “Engine”, “Transmission”, and “Body” are common features (belongs to the parent feature “Car”), which means it must be including in any product related to car SPL.

Option Feature: Option feature can follow his parent feature or not. For instance, “Cruise” is an optional feature.

Selection process: In selection process, one or more features could be selected from the parent feature. In basic FM, this operation is known as alternative, where one child feature can be included in a product when the parent feature is included. In cardinality FM, the selection process organizes by two numbers represent maximum and minimum numbers allowed to be selected from parent feature.

Constraint dependency: is known also as cross-tree relation. Require and exclude represent the constraint dependency. In the following require and exclude are defined.

Exclude: feature X excludes Y, means that if X is included in a product then Y must not be included and vice versa.

Require: feature X requires Y, means that if X is included in a product then Y must be included.

2.3 Orthogonal Variability Model

Orthogonal Variability Model (OVM) [5] is one of the useful techniques to represent variability, which provides a cross-sectional view of the variability through all software development artifacts. **Figure 2** illustrates OVM for e-shopping system. **Figure 2** borrow from [5]. Variabil-

ity is described (in OVM) by three terms. These terms are:

Variation Point: a variation point is a point that could be select one or more of its variants. For instance, "Member reward" in **Figure 2** is a variation point.

Variant: a variant is a choice belonging to specific variation point. For instance, "https", "SSL", and "SET" are variants belonging to "Secure payment" variation point.

Constraint dependencies rules: These rules describe the dependency relation between variation points and variants. Require and exclude signify the constraint dependency relation in OVM. These relations are: variation point requiring or excluding another variation point, variant requiring or excluding another variant and variant requiring or excluding variation point.

3. Related Work

In this section, we survey the works that related to validation of SPL regardless the method of modeling variability used. Inside these studies, we highlight the validation operations of the configuration. At the end, we justify our contributions.

Schlich and Hein proved the needs and benefits of using the knowledge-base representation for configuration systems in [16]. A knowledge-based product derivation process [17,18] is a configuration model that includes three entities of Knowledge Base. The automatic selection provides a solution for complexity of product line variability. In contrast to the proposed method, the knowledge -based product derivation process does not provide explicit definition of variability notations and for the selection process. In addition, knowledge-based product derivation process is not focused on validation operations.

Mannion [19] was the first who connect propositional formulas to FM. Mannion's model did not concern cross-tree constraints (Require and Exclude constraints). Zhang *et al*. [20] defined a meta-model of FM using Unified Modeling Language (UML) core package and took Mannion's proposal as foundation and suggested the use of an automated tool support. Zhang's model satisfies constraint dependency checking in the basic level (feature-to-feature). By define pre-condition and post-condition for each feature explanation operation was satisfied but there is no mentioned for propagation. Batory in [21] proposed a coherent connection between FM, grammar and propositional formulas. Batory's study represented basic FM using context–free grammars plus propositional logic. This connection allows arbitrary propositional constraints to be defined among features and enables off-the-shelf satisfible solvers to debug FM. Batory using solvers satisfied constraint dependency checking and explanation operations only. Sun and Zhang [22] proposed a formal semantics for the FM using first order logic. Sun used Alloy Analyzer (tool for analyzing models written in alloy) to automate constraint dependency checking (feature-to-

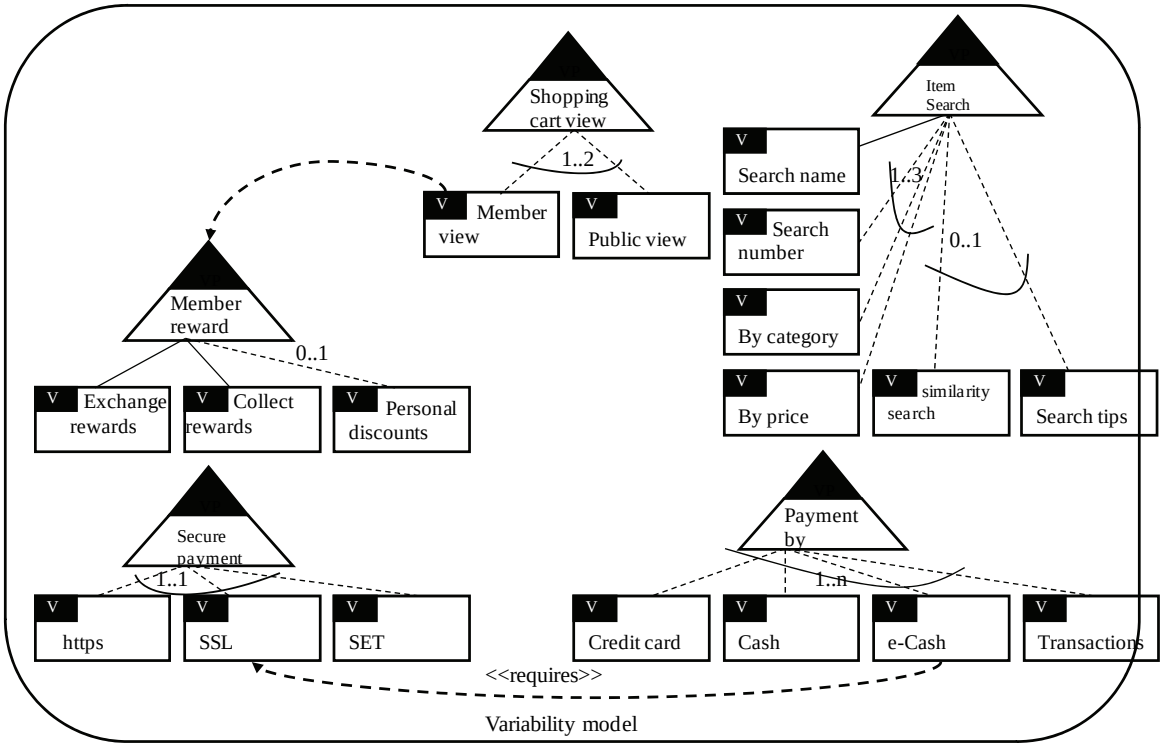


Figure 2. OVM represent variability in E-shopping system

feature level) and explanation operations in the configuration process. Asikainen *et al*. [15] satisfied constraint dependency checking and explanation by translate the model into Weigh Constraint Rule Language (WCRL). WCRL is a general-purpose knowledge representation language developed based on propositional logic.

Wang *et al*. [23] proposed Ontology Web Language (OWL) to Verify FM. Wang used OWL DL ontology to capture the interrelationships among the features in a FM. Wang supported constraint dependency checking and explanation by using FaCT++ (Fast Classification of Terminologies and RACER (Renamed ABox and Concept Expression Reasoner) reasoner tools. Falbo *et al*. [24] formalized domain-engineering process using ontology. Falbo *et al*. mapped the constraint relations in domain engineering to the synonymous primitive in set theory, and used hybrid approach based on pure first-order logic, and set theory for reasoning.

Dedeban [25] used OWL DL and rule based system to support constraint dependency checking. Dedban's method satisfy constraint dependency checking and explanation used FaCT++ reasoner. Shaofeng and Zhang [26] formalized the FM with description logic to reason constraint rules via description logic.

Transforming FM into Unified Modeling Language (UML) notations, representing, and documenting variability in SPL is proposed by different methods in literature. Usability is the main reward for using UML due to UML standardization. Clauss [27] suggested Object Constraint Language (OCL) to satisfy constraint dependency rules. Korherr and List [28] proposed a UML 2 profile to model variability. Korherr's model used OCL to satisfy the three levels constraint dependency rules. Ziadi *et al*. [29,30] used OCL in the form of meta-model level to satisfy constraint dependency rules. The works in [31-37] adopted UML (with different views) as a solution for modeling variability in SPL. These methods implemented OCL to satisfy dependency constraint rules. Czarnecki and Antkiewicz [38] proposed a general template-based approach for mapping FM. Czarnecki and Pietroszek [39] used object-constraint language (OCL) to validate constraint dependency rules.

The use of constraint programming for dealing with constraint dependency checking and explanation are suggested in [40-42] where FM is translated into a Constraint Satisfaction Problem (CSP) for automating FM analysis with a constraint solver. White *et al*. [43] proposed a method for automated diagnosis of product-line configuration errors in FM. White's method starts with transferring the rules of the FM and the current invalid configuration into a CSP. Later, the solver derives a labeling of the diagnostic CSP. Finally, the output of the CSP labeling is transform into a series of recommendations of features to select or deselect to turn the invalid configuration into a valid configuration.

Cao *et al*. [44] developed algorithm to transfer FM into data structures. This algorithm generates complete feature instances from a feature diagram under constraints. Cao used the Generic Modeling Environment (GME) to develop the algorithm. The Cao's algorithm satisfies constraint dependency checking and explanation operations. Deursern and Klint [45] proposed a feature description language to describe FM. Using their system constraint dependency checking and explanation operations are satisfied. Pohjalainen [46] described subset of regular expressions that can be used to express a FM. Pohjalainen presented a compiler for translating a FODA model to a deterministic finite state machine with support for implementing model constraints via post-augmentation of the compiled state machines. This model satisfies constraint dependency checking and explanation operation.

Cechticky *et al*. [47] proposed a feature meta-model and used an XSL-based mechanism to express complex composition rules for the features. Cechticky *et al*. described a compiler that can translate the constraint model expressed as a feature diagram into an XSL program and checks compliance with the constraints at application model level. Jarzabek and Zhang [48] described a variant configuration language that allows to instrument domain models with variation points and record variant dependencies. Implementation based on XML and XMI technologies was also described. Jarzabek's method satisfies constraint dependency checking and explanation.

Janota and Kiniry [46] formalized a FM using HOL. As the best of our knowledge this is the only one work used HOL for reasoning FM. This formalization satisfied constraint dependency checking and explanation operations. Lengyel *et al*. [49] proposed an algorithm (to handle constraints in FM) based on graph rewriting based topological model transformation. Implementation is done based on OCL semantics. Constraint dependency checking is satisfied based on feature-to-feature level.

Comparing with the literature, our propose method (to satisfy constraint dependency rules, explanation, corrective explanation, propagation and delete cascade, and filtering) is characterized by an interactive mechanism which is guide users systematically. In addition to constraint dependency rules (require and exclude), our validation rules are validate the commonality (is the feature is common or not), and the cardinality.

4. Modeling Variability Using First Order Logic

The most popular models for SPL variability modeling are FM and Orthogonal Variability Model (OVM). Therefore, the successful validation notations are those that can validate both FM and OVM. Roos-Frantz [50] illustrated the differences between FM and OVM. To overcome the differences we merge the FM and OVM in the proposed method (benefiting from their advantages),

e.g. a variation point is defined explicitly (mandatory or optional) and hierarchical structure is supported. This modeling is a prerequisite process for using the validation operations. OVM and FM can easily become very complex for validating a medium size system, *i.e.*, several thousands of variation points and variants are needed.

4.1 Upper Layer Representation (FM-OVM)

Figure 3 represents the upper layer of our proposed method. Optional and mandatory constraints are defined in **Figure 3** by original FM notations [3] and constraint dependency rules are described using OVM notations. The FM-OVM has three layers. Member reward is a variation point having a personal discount as a variant. A personal discount is also a variation point for three variants. A personal discount is a variant and variation point at the same time.

4.2 Lower Layer of the Proposed Method

Variation points, variants, and dependency constraint rules are described using predicates as a lower layer of the proposed method: (examples are based on **Figure 3**. Terms starting by capital letters represent variables and terms starting by lower letters represent constants):

4.2.1. Variation Point

The following five predicates are used to describe each variation point:

type: Define the type of feature; variation point, e.g.: type (view_type,variationpoint),

variants: Identifies the variant of specific variation point, e.g.: variant (view_type, not registered).

max: Identifies the maximum number allowed to be selected of specific variation point, e.g. max (payment_by, 4).

min: Identifies the minimum number allowed to be selected of specific variation poin, e.g. min payment_ by, 1). The common variant(s) in a variation point is/ are not included in maximum-minimum numbers of selection.

common: Describe the commonality of variation point, e.g. common (item-search, yes). If the variation point is not common, the second slot in the predicate will become No, as example, common (member reward, no).

4.2.2. Variant

The Following two predicates (from the above five predicates) are used to describe variants:

type: Define the type of feature (variant), e.g.: type (register, variant).

common: Describe the commonality of variant, e.g. common(search name, yes). If the variant is not common, the second slot in the predicate will become No -as example-common (by price, no).

4.2.3. Constraint Dependency Rules

The following six predicates are used to describe constraint rules:

requires_v_v: a variant requires another variant, e.g. requires_v_v (ecash, ssl).

excludes_v_v: a variant excludes another variant, e.g. excludes_v_v(by price, member view).

requires_v_vp: a variant requires variation point, e.g. requires_v_vp (member_view, member_reword).

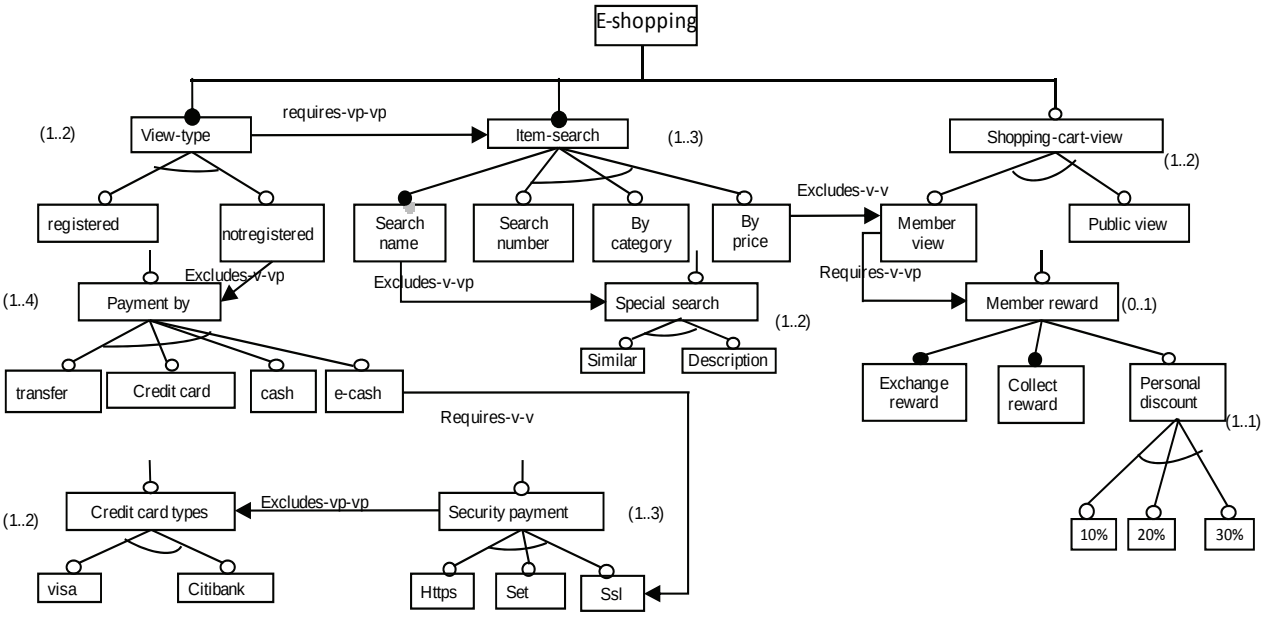


Figure 3. Representation of e-shopping system using the upper layer (FM-OVM)

excludes_v_vp: a variant excludes variation point, e.g. excludes_v_vp (notregistered, payment_by).

requires_vp_vp: a variation point requires another variation point, e.g. requires_vp_vp (item_search, view_ype).

excludes_vp_vp: a variation point excludes another variation point, e.g. excludes_vp_vp (security_payment, credit_card_type).

Table 1 shows the lower layer representation of the variation point *view-type* and the variant *not registered*. The lower layer models variability in one-to-one mapping. The predicate *variants* emphasize this point.

4.3 Generalization

FM-OVM might compose of many levels (variation point can contain one or more variation points) for example form **Figure 3**: variation point *member reward* has a variation point *personal discount* and variation point *personal discount* has a variant **30%**. A mathematical representation of this case is: *member reward* (*personal discount* (30%)). The following facts illustrate the modeling of this case:

type(member_reward,variationpoint)∧type(personal_discount,variation-point)∧type(30%,variant)∧variants(meber_reward,personal_iscount)∧variants(personal_discount,30%).

The following rule (transformation rule) concludes this relation:

$\forall x, y, z$: variants(x, y)∧variants(y, z) $\Longrightarrow$ variants(x, z).

In the next section, we illustrate how the proposed method can be used for validating stage-configuration in SPL.

5. Operations for Validating Stage-Configuration in Software Product Line

5.1 Validation Rules

To validate the configuration process, the proposed method triggers rules based on constraint dependencies. With regard to validation process result, the choice is added to knowledge-base or rejected, then an explanation of rejection reason is provided and correction actions are suggested. When a new variant is selected, new predicate (*select* or *notselect*) would be added to the knowledge-base and the backtracking mechanism validates the entire

Table 1. Snapshot of the lower layer representation

type(view-type, variationpoint). variants(view-type, registered). variants(view-type, not registered).common (view-type, yes). min(view-type, 1). max(view-type, 3). requires_p_p(view-ype, earch_tem). type(not registered, variant). common(not registered, no). excludes_v_vp(not registered, payment by).

Table 2. Predicates represent constraint dependency rules in the proposed method

requires_v_v: Variant requires variant equire_v_v(x,y)| x,y ∈{V}; V= variant	The selection of a variant *V*1 requires the selection of another variant *V*2 independent of the variation points the variants are associated with. e.g. requires_v_v (ecash, ssl).
excludes_v_v: Variant excludes variant exclude_v_v(x,y)| x,y ∈ {V}; V= variant	The selection of a variant *V*1 excludes the selection of the related variant *V*2 independent of the variation points the variants are associated with. e.g. excludes_v_v(By price, member view).
requires_v_vp: Variant requires variation point require_v_vp (x,y)| x,y ∈{V,VP}; V= variant.Vp=variation point	The selection of a variant *V*1 requires the consideration of a variation point *VP*2. e.g. requires_v_vp (member_view, member_reward).
excludes_v_vp: Variant excludes variation point excludes_v_vp (x,y) |x,y ∈ {V, VP}; V = variant. VP=variation point	The selection of a variant *V*1 excludes the consideration of a variation point *VP*2. e.g. excludes_v_vp (not registered, payment_by).
requires_vp_vp: Variation point requires variation point require_ vp_ vp (x,y) | x,y ∈ {VP}; VP= variation point	A variation point requires the consideration of another variation point in order to be realized. e.g. requires_vp_vp (item_ search, view_type).
excludes_vp_vp Variation point excludes variation point excludes_vp_vp (x,y)|x,y ∈ {VP}; VP= var- iation point	The consideration of a variation point excludes the consideration of another variation point. e.g. excludes_vp_vp (security_ payment, credit_card_type).

knowledge-base. At the end of the configuration process, *select* and not *notselect* predicates represent the product. **Table 3** shows the abstract representation of the main rules.

Rule 1:

For all variant *x* and variant *y*; if *x* requires *y* and *x* is selected, then *y* is selected.

Rule 2:

For all variant *x* and variant *y*; if *x* excludes *y* and *x* is selected, then *y* is assigned by *notselect* predicate.

Rule 3:

For all variant *x* and variation point *y*; if *x* requires *y* and *x* is selected, then *y* is selected. This rule is applicable as well if the variation point is selected first:

$\forall$ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ require_v_vp(x, y) ∧ select(y) $\Longrightarrow$ select(x)

For all variant *x* and variation point *y*; if *x* requires *y* and *y* is selected, then *x* is selected.

Rule 4:

For all variant *x* and variation point *y*; if *x* excludes *y* and *x* is selected, then *y* assigned by *notselected* predicate.

Table 3. Abstract representation of the main rules

1	∀ x, y: type(x, variant) ∧ type(y, variant) ∧ require_v_v(x, y) ∧ select(x) ⟹ select(y)
2	∀ x, y: type(x, variant) ∧ type(y, variant) ∧ exclude_v_v(x ,y) ∧ select(x) ⟹ notselect(y)
3	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ require_v_vp(x, y) ∧ select(x) ⟹ select(y)
4	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ exclude_v_vp(x, y) ∧ select(x) ⟹ notselect(y)
5	∀ x, y: type(x, variationpoint) ∧ type(y, variationpoint) ∧ require_vp_vp(x, y) ∧ select(x) ⟹ select(y)
6	∀ x, y: type(x, variationpoint) ∧ type(y, variationpoint) ∧ exclude_vp_vp(x, y) ∧ select(x) ⟹ notselect(y)
7	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ select(x) ∧ variants(y, x) ⟹ select(y)
8	∃x ∀y:type(x, variant) ∧ type(y, variationpoint) ∧ select(y) ∧ variants(y, x) ⟹ select(x)
9	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ notselect(y) ∧variants(y, x) ⟹ notselect(x)
10	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ common(x,yes) ∧ variants(y, x) ∧ select(y) ⟹ select(x)
11	∀ y: type(y, variationpoint) ∧ common(y,yes) ⟹ select(y)
12	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧variants(y, x) ∧select(x) ⟹ sum(y,(x)) ≤ max(y,z)
13	∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧variants(y, x) ∧select(x) ⟹ sum(y,(x)) ≥ min(y,z)

This rule is applicable as well, if the variation point is selected first:

∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ exclude_v_vp(x, y) ∧ select(y) ⟹ notselect(x)

∀ x, y: type(x, variant) ∧ type(y, variationpoint) ∧ exclude_v_vp(x, y) ∧ select(y) ⟹ notselect(x)

For all variant *x* and variation point *y*; if *x* excludes *y* and y selected, then *x* is assigned by *notselect* predicate.

Rule 5:

For all variation point *x* and variation point *y*, if *x* requires *y* and *x* selected, then *y* is selected.

Rule 6:

For all variation point *x* and variation point *y*, if *x* excludes *y* and *x is* selected, then *y* is assigned by *notselect* predicate.

Rule 7:

For all variant *x* and variation point *y*, where *x* belongs to *y* and *x* is selected, that means *y* is selected.

This rule determines the selection of variation point if one of its variants was selected.

Rule 8:

For all variation point *y* there exists of variant *x*, if *y* selected and *x* belongs to *y* then *x* is selected.

This rule states that if a variation point was selected, then there is variant(s) belong to this variation point must be selected.

Rule 9:

For all variant *x* and variation point *y*; where *x* belongs to *y* and *y* defined by predicate *notselect*(*y*), then *x* is assigned by *notselect* predicate. This rule states that if a variation point was excluded, then none of its variants must select.

Rule 10:

For all variant *x* and variation point *y*; where *x* is a common, *x* belongs to *y* and *y* is selected, then *x* is selected. This rule states that if a variant is common and its variation point selected then it is selected.

Rule 11:

For all variation point *y*; if *y* is common, then *y* is selected. This rule states that if a variation point is common then it is selected in any product.

Rule 12:

For all variant *x* and variation point *y*; where *x* belongs to *y* and *x* is selected, then the summation of *x* must not be less than the maximum number allowed to be selected from *y*.

Rule 13:

For all variant *x* and variation point *y*; where *x* belongs to *y* and *x* is selected, then the summation of *x* must not be greater than the minimum number allowed to be se--lected from *y*.

The *notselect* predicate prevents feature to be selected, e.g. rule 9.

Rules 12 and 13 validate the number of variants' selection considering the maximum and minimum conditions in variation point definition (cardinality definition). The predicate *sum*(*y*, (*x*)) returns the summation number of selected variants belongs to variation point *y*. From these rules, the full common variant (variant included in any product) can be defined as:

∀x,y:type(x,variant)∧type(y,variationpoint)∧variants(y,x) ∧common(y,yes)∧common(x,yes)⟹ full_common(x)

A full common variant is a common variant belongs to common variation point. A common variation point included in any product (rule 11), a common variant belongs to selected variation point is selected (rule 10).

The proposed rules (to validate the configuration) are based on two layers. The upper layer is a variation point layer where each rule applied to variation point reflect into all its variants, e.g. if variant excludes variation point that means this variant excludes all the variants that belong to this variation point, rule (9). The lower layer is a variant layer where each rule is applied for specific variant.

5.2 Explanation and Corrective Explanation

This operation is defined (in this paper) for highlighting the sources of errors within configuration process.
The general pattern that represents failure to select one feature in the configuration process is:

Feature A excludes Feature B and Feature A is selected then Feature B failed to select.

In the proposed method, there are two possibilities for the feature: variation point or variant and three possibilities for the exclusion: variant excludes variant, variant excludes variation point or variation point excludes variation point. Definition 1 describes these possibilities in the form of rules.

Definition 1:

Selection of variant *n*, *select* (*n*), fails due to selection of variant *x*, *select*(*x*), in three cases:

∀x,y,n:type(x,variant)∧select(x)∧type(y,variationpoint) ∧variants(y,x)∧type(n,variant)∧excludes_v_vp(n,y)⟹ notselect(n).

If the variant *x* is selected, and it belongs to the variation point *y*, this means *y* is selected (rule 7), and the variant *n* excludes the variation point *y*, this means *n* assigned by *notselect* predicate (rule 4 is applied also if the variation point is selected).

∀x,y,z,n:type(x,variant)∧select(x)∧type(y,variationpoint)∧variants(y,x)∧variants(z,n) excludes_vp_vp(y,z)⟹ notselect(n).

If the variant *x* is selected and *x* belongs to the variation point *y*, that means *y* is selected (rule 7), and if the variation point *y* excludes the variation point *z*, this means *z* is assigned by *notselect* predicate (rule 6), and the variant *n* belongs to variation point *z*, this means *n* is assigned by *notselect* predicate (rule 9).

∀x,n: type(x,variant)∧select(x)∧type(n,variant)∧excludes_v_v(x,n)⟹notselect(n).

If the variant *x* is selected, and *x* excludes the variant *n*, which means *n* is assigned by *notselect* predicate (rule 2). In addition to defining the source of error, these rules can be used to prevent the errors. The predicate *notselect*(*n*) validate users by preventing selection.

Example 1

Suppose the user selects *memebr_view* before entering a new selection and request to select *by price*, the system rejects the choice and directs the user to deselect member_*view* first. **Table 4** describes example 1. This example represents rule (3). The example illustrates how the proposed method guides users to solve the rejection reason. In addition to that, it can be used to prevent rejection reasons; example 2 explains this.

Example 2

The user asks to select the variant *https*, the system accepts the choice and adds *notselect*(*credit_card_types*) to the knowledge-base to validate future selections. **Table 5** describes example 2. Selection of the variant *Https* from *security_payment* variation point leads to the selection of *security_payment* (rule 7), and *secu rity_payment* excludes *credit_card_types* variation point, which means *credit_card_types* must not be selected (rule 6). The predicate *notselect*(*credit_card_types*) prevents the selection of its variants according to rule 9.

Table 4. Example 1

? select (by price). You have to deselect memebr_view

Table 5. Example 2

? select (Https). Yes notselect (credit_card_types) added to knowledge base.

The proposed method guides user step by step (in each choice), if the user's choice is invalid immediately reject it and suggest the correct actions (corrective explanation), see example 1. Moreover, *notselect* predicate can be assigned to some features according to user's selection, see example 2. The *notselect* predicate prevents user from future errors.

5.3 Filtering

Filtering operation guides the user to develop his product based on predefined conditions.

Example 3

Suppose *price* was defined as an extra-functional feature to *security_payment* variation point in **Figure 3**. As a result three new predicates (*price*(*https*,100), *price* (*ssl*, 200), and *price*(*set*,350)) were added. We want to ask about the feature with price greater than 100 and less than 250 (price(X, Y), Y > 100, Y < 250), the system triggers the variant *ssl* with price 200. **Table 6** describes example 3.

5.4 Propagation and Delete-Cascade

In this operation, some features are automatically selected (or deselected).

The general pattern that represents selection of feature based on selection of another feature is:
Feature A requires feature B and feature A is selected then feature B is auto-selected.

In the proposed method, there are two possibilities for the feature: variation point or variant and three possibilities for the requiring: variant requires variant, variant requires variation point or variation point requires variation point. Definition 2 describes these possibilities in the form of rules.

Definition 2:

Selection of variant *n*, *select*(*n*), is propagated fromselection of variant *x*, *select*(*x*), in three cases:

i.∀x,y,z,n:type(x,variant)∧variants(y,x)∧select(x)∧requires_vp_

Table 6. Example 3

? price(X, Y), Y > 100, Y< 250. X = ssl Y = 200

vp(y,z)∧type(n,variant)∧variants(z,n)∧common(n,yes)⟹ autoselect(n).

If *x* is a variant and *x* belongs to the variation point *y* and *x* is selected, that means *y* is selected (rule 7), and the variation point *y* requires a variation point *z*, that means *z* is selected also (rule 5), and the variant *n* belongs to the variation point *z* and the variant *n* is common that means the variant *n* is selected (rule 10).

ii.∀x,n:type(x,variant)∧type(n,variant)∧select(x)∧requires_v_v(x,n) ⟹ autoselect(n).

If the variant *x* is selected and it requires the variant *n*, that means the variant *n* is selected, (rule 1). The selection of variant *n* propagated from the selection of variant *x*.

iii.∀x,z,n:type(x,variant)∧select(x)∧type(z,variationpoint)∧requires_v_vp(x,z)∧type(n,variant)∧variants(z,n)∧common(n,yes) ⟹ autoselect(n).

If the variant *x* is selected and it requires the variation point *z* that means the variation point *z* is selected (rule 3), and the variant *n* is common and belongs to the variation point *z* that means the variant *n* is selected (rule 10). The selection of variant *n* propagated from the selection of variant *x*.

Example 4

Suppose the user enters this choice, *select(register)*, the system answered yes (acceptance of user selection) , user announced by selection of the variant *search_name*, as propagated from selection of the variant *register*. This example illustrates case 1: *view_type* variation point requires *item_search* variation point and *search_name* is common variant belongs to the variation point *item_search*. The direct selection of variant *register* makes v*iew_ type* variation point selected (rule 7), and the selection of *view_type* variation point makes the *item_search* variation point selected (rule 5), then the common variant *search_name* (belongs to *item_search* variation point) is selected (rule 10). The main result of this example is the additions of two new facts *select (register)* and *autoselect (search_name)* to the knowledge base. **Table 7** illustrates example one.

Delete-Cascade Operation:

This operation validates configuration process in the execution time. The following scenario describes the problem: If the variant *x* is selected in time *N* and *x* requires two variants *y* and *k*, then the solution (at time *N*) = {*x*, *y*, *k*}. In time (*N* + 1), the variant *m* is selected, and *m* excludes *x*, then *x* is remove from the solution. The solution at time (*N* + 1) = {*m*, *y*, *k*}. The presence of the variants *y* and *k* is not a real selection. Rule 2 (Subsection 5.1) assign *notselect* predicate (to the excluded variant) as a result from the exclusion process. The following rules added to the knowledge base to implement *delete-cascade*.

i.∀x,y:type(x,variant)∧type(y,variant)∧requires_v_v(y,x)∧autoselect(x) ∧select(y) ∧ notselect(y)⟹ notselect(x).

ii.∀x,y:type(x,variant)∧type(y,variant)∧requires_v_v(y,x)∧autoselect(x) ∧autoselect(y) ∧ notselect(y)⟹ notselect(x).

For all variants *x*, and *y*; if the variant *y* is requires *x*, y is selected or auto selected, *x* is auto selected and *y* assigned by *notselect* predicated, that means *y* is excluded within the configuration process, and *x* was selected according to selection of *y* (propagation) then the presence of *x* after exclusion of *y* is not true. The output for this operation is the assigning of the variant *x* with *notselect* predicate. This assigning permits backtracking mechanism to perform *delete-cascade* operation to verify the products.

5.5 Cardinality Test

Cardinality test operation validates the cardinality of each selected variation point. At the begging of this operation, all auto-selected variants must be assigned by *select* predicate, which constructs all selected variants in the configuration process assign by *select* predicate.
The following rule assiges *select* predicate to all auto-selected variants.

∀x: autoselect(x) ⟹ select(x).

The following rule converts all selected variants belonging to the variation point *y* to a list, *i.e.* the list contains only variants belonging to one variation point:

∀y,x:variants(y,x),select(x) ⟹ list(y,[x]).

The following rules test the maximum and minimum cardinality for each selected variation point.

∀y,x,len,m: length(list(y,[x]),len) ∧ max(y,m) ∧ len > m ⟹ error.

∀y,x,len,m: length(list(y,[x]),len) ∧ min(y,m) ∧ len < m ⟹ error.

In the above two rules, the length of a list compares against cardinality of its variation point and alert messages triggers out in case of error.

6. Implementation and Scalability Testing

In this section, technical details are present and discuss. **Table 8** shows an interactive configuration program. All programs are implemented based on prolog SWI software.

The program in **Table 8** guides user step by step to complete his selections. First, user enters his choice (the variant *X*). Later, two subroutines validate the selection. The first subroutine (interactive_validate_require) works to figure out all variants required by the selected variant *X* based on the three constraints: variant requires variant, variant requires variation point, and variation point requires variation point. The second subroutine (interacttive_ validate_exclude) works to figure out all variants excluded by the selected variant *X* based on the three constraints: variant excludes variant, variant excludes variation point, and variation point excludes variation point.

Table 9 shows a program to generate the maximum product. A maximum product defined as a product contains all variant in SPL considering the constraint dependency rules [42].

The program to generate the maximum product (**Table 9**) contains five subroutines: sel_common, sel_variant, validate_exclude, del_cascade, and make_product. In the following, each subroutine is discussed:

sel_common: this subroutine selects the common variants, *i.e.*, common variant belonging to common variation point.

sel_variant: select all the variants (that are not selected before as common variants) are the mission of this subroutine.

validate_exclude: This subroutine validates exclude constraint. In subroutine, all variants are compared against each other. The excluded variant is assigned by *notselect* predicate.

del_cascade: this subroutine implements the delete-cascade operation that is defined in Subsection 5.4 in this paper.

Table 7. Example 4

```
? select (view_type.register).
Yes
                You selected also…. search_name
```

Table 8. An interactive configuration program

```
sel:-
read(X),
interactive_validate_exclude(X),
interactive_validate_require(X).

interactive_validate_require(X):-
type(N,variant), X \==N,
((requires_v_v(X,N), write(' you have to de select '), write(N),nl);
( variants(M,N), requires_v_vp(X,M), common(N,yes),write(' you
have to select '), write(N),nl);
(variants(Y,X), variants(M,N), requires_vp_vp(Y,M), common
(N,yes), write (' you have to select '), write(N).

interactive_validate_exclude(X):-
type(N,variant), X \==N,
((excludes_v_v(X,N), write(' you have to de select '), write(N),nl);
( variants(M,N), excludes_v_vp(X,M),write(' you have to deselect '),
write(N),nl) ;
(variants(Y,X), variants(M,N), excludes_vp_vp(Y,M),write(' you
have to deselect '),write(N).
```

Table 9. A program to generate the maximum product

```
max_product:-
sel_common,
sel_variant,
validate_exclude,
del_cascade,
make_product.

sel_common:-
variants(Y,X),
common(Y,yes),
common(X,yes),
write('select'),write('('), write(X), write(').').

sel_variant:-
type(X, variant),
not(select(X)),
write('select'),write('('), write(X), write(').').

validate_exclude:-
select(N), select(X), X \==N,
((excludes_v_v(X,N), write('notselect'),write('('), write(X), write (').'), nl);
(variants(M,N),excludes_v_vp(X,M),write('notselect'),write('('),
write(X), write(').'),nl) ;
(variants(Y,X), variants(M,N), excludes_vp_vp (Y,M),write ('notse-
lect'), write('('), write(X), write(').'),nl)).

del_cascade:-
select(N),
requires_v_v(M,N),
notselect(M), % that means M was selected and then deleted
write('notselect'),write('('), write(N), write(').').

make_product:-
write('S/wproduct.'),
select(X),not(notselect(X)),
write(X),write(', ').
```

make_product: This subroutine print out the maximum product. The maximum product is represented by all variant assigned by *select* predicate and not assign by *notselect* predicate.

6.1 Scalability Test

Scalability is a key factor in measuring the applicability of the techniques dealing with variability modeling in SPL [51]. The output time is a measurement key for scalability. A system consider scalable for specific problem if it can solve this problem in a reasonable time. In the following, we describe the method of our experiments:

Generate the domain engineering as a data set: Domain engineering is generated in terms of predicates (variation points, and variants). We generated four sets containing 1000, 1500, 3000, and 50000 variants. White *et al.* [43] defines 5000 features as a suitable number to mimic industrial SPL. Variants are defined as numbers represented in sequential order, as example: In the first set (1000 variants) the variants are: 1, 2, 3,…, 1000. In the last set (5000 variants) the variants are: 1, 2, 3, …, 5000. The number of variation point in each set is equal to number of variant divided by five, which means each variation point has five variants. As example in the first

set (1000 variants) number of variation points equal 1000. Each variation point defined as sequence number having the term *vp* as postfix, e.g. vp12.

Define the assumptions: We have three assumptions: 1) each variation point and variant has a unique name, 2) each variation point is orthogonal, and 3) all variation points have the same number of variants.

Set the parameters: The main parameters are the number of variants and the number of variation points. The remaining eight parameters (common variants, common variation points, variant requires variant, variant excludes variant, variation point requires variation point, variation point excludes variation points, variant requires variation point, and variant excludes variation point) are defined as a percentage. The number of the parameters related to variant (such as; common variant, variant requires variant, variant excludes variant, variant requires variation point, and variant excludes variation point) is defined as a percentage of the number of the variants. The number of parameters related to variation point (such as; variation point requires variation point) is defined as a percentage of the number of variation points. We found that the maximum ratio of constraint dependency rules used in literature is 25% [51]. Therefore, we defined the ratio of the parameters in our experiments as 25%. **Table 10** represents snapshots of an experiment's dataset, *i.e.*, the domain engineering in our experiments.

Calculate the output: we tested two programs for each program, we made thirty experiments, and calculated the execution time as average. The experiments were done with the range (1000-50000) variant, and percentage range of 25%.

Experimental platform:

The experiments were performed on a computer with an Intel centrino Duo 1.73GHZ CPU, 2 gigabytes of memory, Windows XP home edition. In the following parts, the results are presented. The results show the execution time compared with number of variants, number of variation points, and the parameters.

6.1.1. Test Scalability of a Program to Validate Product in Interactive Mode

In this subsection, we test the scalability of interactive configuration program (**Table 8**). Instead of read the user's input one by one, we define additional parameter, the predicate select(c), where c is random variant. This predicate simulates the user's selection. Number of select predicate (defined as a percentage of number of variant) is added to the domain engineering (dataset) for each experiment. **Table 11** contains a program to validate the product in interactive mode. This program is modification of the program in **Table 8**. This modification allows us to test the scalability.

Table 12 shows the result of scalability test for a program to validate product in interactive mode.

6.1.2. Test Scalability of a Program to Define the Maximum Product

In this subsection, the scalability test for a program to define the maximum product is discussed. **Table 13** shows the results.

In [51] the execution time for 200-300 features is 20 min after applying atomic sets to enhance the scalability. With compare to the literature, our proposed method is scalable.

7. Conclusions and Future Work

In this paper, a method to validate SPL in stage-con-

Table 10. Snapshot of experiment's dataset

```
type(vp1,variationpoint).type(1,variant).
variants(vp1,1).
common(570,yes).
Common(vp123,yes).
requires_v_v(7552,2517).
requires_vp_vp(vp1572,vp1011).
excludes_vp_vp(vp759,vp134).
excludes_v_v(219,2740).
requires_v_vp(3067,vp46).
excludes_v_vp(5654,vp1673).
```

Table 11. A program to validate product in interactive mode

```
sel:-
interactive_validate_exclude,
interactive_validate_require.

interactive_validate_exclude:-
select(N), select(X), X \==N,
((excludes_v_v(X,N), write(' you have to de select    '), write(N),nl);
( variants(M,N), excludes_v_vp(X,M),write(' you have to deselect '),
write(N),nl) ;
(variants(Y,X),  variants(M,N),  excludes_vp_vp(Y,M),write('  you
have to deselect '), write (N), nl)).

interactive_validate_require:-
type(N,variant), select(X),  not(select(N)),
((requires_v_v(X,N), write(' you have to de select    '), write(N),nl);
( variants(M,N), requires_v_vp(X,M), common (N,yes), write (' you
have to select '), write(N),nl);
(variants(Y,X),  variants(M,N),  requires_vp_vp(Y,M),  com- mon
(N,yes), write (' you have to select '), write(N),nl)).
```

Table 12. Results of scalability test for a program to validate product in interactive mode

Number of variants	Time (Min)
1000	0.4
1500	1.6
3000	12.8
5000	59.6

Table 13. Results of scalability test for a program to generate the maximum product

Number of variants	Time (Min)
1000	1.98
1500	6.85
3000	54
5000	251

figuration process is presented. Firstly, modeling variability using FOL predicates was proposed. By this modeling, we can get formalized variability specifications, support and validate selection process within variability more precisely. The proposed method provides automated consistency checking among constraints (during configuration process) based on three levels (*i.e.* variant-to-variant, variant-to-variation point, and variation point-to-variation point). The proposed method guides users interactively step-by-step (in each choice). If the user's choice is invalid, immediately is rejected and correction actions are suggested (corrective explanation). Moreover, the proposed method can be used to correct future selections using *notselect* predicate (rule 9). All variants selected directly (by user) assigned by *select* predicate. All variants selected using propagation process assigned by *autoselect* predicate. The delete-cascade operation validates auto-select variants. Before cardinality test, all variants in the configured product converted to assign by *select* predicate. Finally, cardinality test validate the number of selection of each variation point.

Many methods are applying empirical results to test scalability by generating random FMs [43,52-54]. Comparing with literature, our test range (1000–5000 variants) is sufficient to test scalability. The proposed method is limited to work only in certain environment, *i.e.* where constraint dependency rules are well known in all cases.

We plan to complete the proposed method by defining operations for validating SPL in static mode. In addition, we plan to implement our method in real life case from industry.

REFERENCES

[1] J. Bosch, "Maturity and Evolution in Software Product Lines," *Proceedings of the Second International Software Product Line Conference*, Springer LNCS, San Diego, Vol. 2379, 19-22 August 2002, pp. 257-271.

[2] P. Clements and L. Northrop, "Software Product Lines: Practices and Patterns," Addison Wesley, Boston, 2001.

[3] K. Kang, J. Hess, W. Novak and S. Peterson, "Feature Oriented Domain Analysis (FODA) Feasibility Study," *Technical Report No. CMU/SEI*-90-*TR*-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[4] K. Kang, J. Lee and P. Donohoe, "Feature-Oriented Product Line Engineering, *IEEE Software*, Vol. 19, No. 4, 2002, pp. 58-65.

[5] K. Pohl, G. Böckle and F. van der Linden, "Software Product Line Engineering Foundations Principles and Techniques," Springer, Verlag Heidelberg Germany, 2005.

[6] D. Benavides, A. Ruiz-Cort´es, D. Batory and P. Heymans, *First International Workshop on Analyses of Software Product Lines* (*ASPL*'08), Limerick, Ireland, 2008.

[7] D. Benavides, A. Metzger and U. Eisenecker, "Third International Workshop on Variability Modelling of Software-intensive Systems," ICB-Research Report No. 29, University of Duisburg Essen, Duisburg, 2009.

[8] H. Wang, H. Li, J. Sun, H. Zhang and J. Pan, "Verifying Feature Models Using OWL," *Journal of Web Semantics*: *Science, Services and Agents on the World Wide Web*, Vol. 5, No. 2, 2007, pp. 117-129.

[9] D. Batory, D. Benavides and A. Ruiz-Cortés, "Automated Analyses of Feature Models: Challenges Ahead," *Communications of the ACM* (*Special Section on Software Product Lines*), 2006.

[10] K. Czarnecki and U. Eisenecker, "Generative Programming: Methds, Tools, and Applications," Addison-Wesley, Boston, 2002.

[11] T. Massen and H. Litcher, "Determining the Variation Degree of Feature Models," *Software Product Lines Conference*, *LNCS* 3714, Rennes, 2005, pp. 82-88.

[12] T. Asikainen, T. Männistö and T. Soininen, "Using a Configurator for Modelling and Configuring Software Product Lines Based on Feature Models," *Proceedings of the Workshop on Software Variability Management for Product Derivation, Software Product Line Conference* (*SPLC*3), Boston, 2004.

[13] K. Czarnecki, S. Helsen and U. Eisenecker, "Staged Configuration Using Feature Models," *Proceedings of Third International Conference of Software Product Lines SPLC*2004, Boston, 2004.

[14] H. Meyer and H. Lopez, "Technology Strategy in a Software Products Company, Product Innovation Management," Blackwell Publishing, Vol. 12, No. 4, 1995, pp. 294-306.

[15] T. Asikainen, T. Mnnistand and T. Soininen, "Representing Feature Models of Software Product Families Using a Configuration Ontology," *Proceedings of the General European Conference on Artificial Intelligence* (*ECAI*) *Workshop on Configuration*, Berlin, 2004.

[16] M. Schlick and A. Hein, "Knowledge Engineering in Software Product Lines," *Proceedings of the* 14*th European Conference on Artificial Intelligent Workshop on Knowledge-Based Systems for Model-Based Engineering*, Berlin, 2000.

[17] L. Hotez and T. Krebs, "Supporting the Product Derivation Process with a Knowledge Base Approach," *Proceedings of the* 25*th International Conference on Software Engineering ICSE*2003, Oregon, 2003.

[18] L. Hotez and T. Krebs, "A Knowledge Based Product Derivation Process and Some Idea How to Integrate Product Development," *Proceedings of the Software Variability Management Workshop*, Groningen, The Netherlands, 2003.

[19] M. Mannion, "Using First-Order Logic for Product Line Model Validation," *Proceedings of the Second Software Product Line Conference SPLC*2, San Diego, 2002.

[20] W. Zhang, H. Zhao and H. Mei, "A Propositional Logic-Based Method for Verification of Feature Models," *The*

6*th International Conference on Formal Engineering Methods ICFEM*04, *LNCS* 3308, 2004, pp. 115-130.

[21] D. Batory, "Feature Models, Grammars, and Propositional Formulas," *Proceedings of the* 9*th International Software Product Lines Conference SPLC*05, Rennes France, 2005.

[22] J. Sun and H. Zhang, "Formal Semantics and Verification for Feature Modeling," *Proceedings of the* 10th *IEEE International Conference on Engineering of Complex Computer Systems* (*ICECCS*05), Shanghai, 2005.

[23] H. Wang, H. Li, J. Sun, H. Zhang and J. Pan, "A Semantic Web Approach to Feature Modeling and Verification," *Proceedings of Workshop on Semantic Web Enabled Software Engineering* (*SWESE'*05), Galway, 2005.

[24] R. Falbo, G. Guizzardi and K. Duarte, "An Ontological Approach to Domain Engineering," *Proceedings of* 14th *International Conference on Software Engineering and Knowledge Engineering*, Ischia, 2002.

[25] V. Dedeban, "Ontology-Driven and Rules-Based System for Management and Pricing of Family of Product," Master Thesis, Norwegian University of Science and Technology Department of Computer and Information Science, Norway, 2007.

[26] F. Shaofeng and N. Zhang, "Feature Model Based on Description Logics," *Proceedings of* 10*th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems KES*2006, Springer-Verlag Berlin Heidelberg, 2006, pp. 1144-1151.

[27] M. Clauss, "Modeling Variability with UML," *GCSE* 2000 -*Young Researchers Workshop*, 3rd GCSE, Erfurt, 2001.

[28] B. Korherr and B. List, "A UML 2 Profile for Variability Models and their Dependency to Business Processes," 18*th International Workshop on Database and Expert Systems Applications*, IEEE, Regensburg, 2007.

[29] T. Ziadi, J. Jezequel and F. Fondement, "Product Line Derivation with UML," *Software Variability Management Workshop*, Groningen, Netherlands, 2003, pp. 94-102.

[30] T. Ziadi and J. Jézéquel, "Product Line Engineering with the UML: Deriving Products," *Chapter in Software Product Lines*, Springer, 2006, pp. 557-586.

[31] E. Oliveira, I. Gimenes, E. Huzita and J. Maldonado, "A Variability Management Process for Software Product Lines," *The* 2005 *Conference of the Centre for Advanced Studies on Collaborative Research*, IBM Centre for Advanced Studies Conference, Toranto, Ontario, 2005, pp. 225-241.

[32] S. Robak, B. Franczyk and K. Politowicz, "Extending the UML for Modelling Variability for System Families," *International Journal of Applied Mathematics and Computer Science*, Vol.12, No. 2, 2002, pp. 285-298.

[33] A. Schnieders, "Modeling and Implementing Variability in State Machine Based Process Family Architectures for Automotive Systems," *The* 3*rd International Workshop on Software Engineering for Automotive Systems ICSE*06, Shanghai, 2006.

[34] H. Gomaa and E. Shin, "Automated Software Product Line Engineering and Product Derivation," *The* 40*th Annual Hawaii International Conference on System Sciences*, Big Island, Hawaii, 2007.

[35] I. Philippow, M. Riebisch and K. Boell, "The Hyper/UML Approach for Feature Based Software Design," *The* 4*th AOSD Modeling with UML Workshop Collocated* 6*th International Conference on the Unified Modeling Language UML*, San Francisco, 2003.

[36] M. Riebisch, K. B¨ollert, D. Streitferdt and I. Philippow, "Extending Feature Diagrams with UML Multiplicities, 6*th World Conference on Integrated Design & Process Technology* (*IDPT*2002), California, 2002.

[37] D. Streitferdt, M. Riebisch and I. Philippow, "Details of Formalized Relations in Feature Models Using OCL," 10*th IEEE International Conference on Engineering of Computer–Based Systems* (*ECBS* 2003), Huntsville, IEEE Computer Society, 2003, pp. 45-54.

[38] K. Czarnecki and M. Antkiewicz, "Mapping Features to Models: A Template Approach Based on Superimposed Variants, *Proceedings of the* 4*th International Conference on Generative Programming and Component Engineering GPCE*'05, Tallinn, Estonia, 2005.

[39] K. Czarnecki and K. Pietroszek, "Verifying Feature-Based Model Templates against Well-Formedness OCL Constraints," *Proceedings of the* 5*th International Conference on Generative Programming and Component Engineering GPCE*'06, Oregon, 2006.

[40] D. Benavides, A. Ruiz-Cort´es and P. Trinidad, "Automated Reasoning on Feature Models," 17*th International Conference* (*CAiSE*05), Porto, 2005, pp. 491-503.

[41] D. Benavides, S. Segura, P. Trinidad and A. Ruiz-Cort´es, "Using Java CSP Solvers in the Automated Analyses of Feature Models," *Post-Proceedings of the Summer School on Generative and Transformational Techniques in Software Engineering* (*GTTSE*), LNCS 4143, 2006.

[42] D. Benavides, "On the Automated Analysis of Software Product Line Using Feature Models, A Framework for Developing Automated Tool Support," Ph.D. Dissertation, University of Sevilla, Sevilla, 2007.

[43] J. White, D. Schmidt, D. Benvides, P. Trinidad and A. Ruiz-Cortes, "Automated Diagnosis of Product Line Configuration Errors on Feature Models," *Proceedings of* 12*th International Conference of Software Product Line*, Limerick Irland, 2008.

[44] F. Cao, B. Bryant and C. Carol, "Automating Feature-Oriented Domain Analysis," *Proceedings of International Conference on Software Engineering Research and Practice* (*SERP'*03), 2003, pp. 944-949.

[45] A. Deursen and P. Klint, "Domain-Specific Language Design Requires Feature Descriptions," *Journal of Computing and Information Technology*, Vol. 10, No. 1, 2002, pp. 1-17.

[46] M. Janota and J. Kiniry, "Reasoning about Feature Models in Higher-Order Logic," *Proceedings of the* 11*th International Software Product Line Conference* (*SPLC*07), Kyoto, 2007.

[47] V. Cechticky, A. Pasetti, O. Rohlik and W. Schaufelberger, "XML-Based Feature Modeling," *Proceedings of the*

8*th International Conference on Software Reuse* (*ICSR*-8), Madrid, 2004.

[48] S. Jarzabek and H. Zhang, "XML-Based Method and Tool for Handling Variant Requirements in Domain Models," 5*th IEEE International Symposium on Requirements Engineering RE*01, IEEE Press, Toronto, 2001. pp. 116-173.

[49] L. Lengyel, T. Levendovszky and H. Charaf, "Constraint Handling in Feature Models," *Proceedings of* 5*th International Symposium of Hungarian Researchers on Computational Intelligence*, Budapest, 2004.

[50] F. Roos-Frantz, "A Preliminary Comparison of Formal Properties on Orthogonal Variability Model and Feature Models," *Proceedings of the* 3*rd International Workshop on Variability Modeling of Software-Intensive Systems*, Sevilla, 2009.

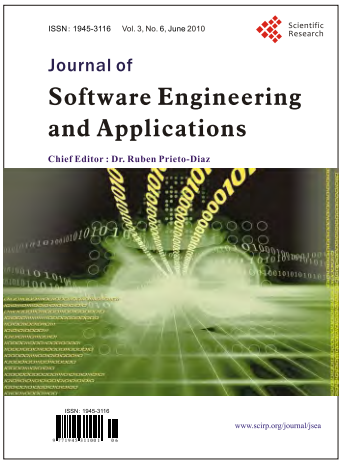
[51] S. Segura, "Automated Analysis of Feature Models Using Atomic Sets," *The* 1*st International Workshop on Analyses of Software Product Lines* (*ASPL*'08), *Collocated with SPLC*08, Limerick Ireland, 12-15 September 2008.

[52] P. Trinidad, D. Benavides, A. Dura´n, A. Ruiz-Cortes and M. Toro, "Automated Error Analysis for the Agilization of Feature Modeling," *Systems and Software*, Vol. 81, No. 6, 2008, pp. 883-896.

[53] P. Trinidad, D. Benavides, A. Ruiz-Cort´es, S. Segura and A. Jimenez, "FAMA Framework," 12*th Software Product Lines Conference* (*SPLC*), Limerick, 2008.

[54] H. Yan, W. Zhang, H. Zhao and H. Mei, "An Optimization Strategy to Feature Models' Verification by Eliminating Verification-Irrelevant Features and Constraints," *Book Chapter in Formal Foundations of Reuse and Domain Engineering*, Springer Berlin/Heidelberg, 2007, pp. 65-75.

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, 517-627.**
Published Online June 2010 in SciRes (http://www.SciRP.org/journal/jsea).

TABLE OF CONTENTS

Volume 3 Number 6 **June 2010**

