Effective Software Assembly for the Real-time systems using Multi-level Genetic Algorithm

R. Amuthakkannan
Senior Lecturer,
Caledonian College of Engineering,
P.O. Box : 2322, CPO Seeb 111
Muscat, Sultanate of Oman.

Abstract:
Software Assembly of reusable components is one of the most important processes to improve the reliability of the product in the Commercial Off The Shelf (COTS) software. Software assembly and also continuous improvement of quality in software based systems is an important issue in software development process. In this paper, a multilevel genetic algorithm is used to find the sequence of the assembly order of the software components among the chosen reusable components. In GA, dependency between the components is also considered based on coupling factor and the assembly order of the identified software component is found effectively.

Keywords: COTS software, Software Assembly, Genetic Algorithm, Dependency Chart

R. Amuthakkannan received his B.E. degree in Mechanical Engineering from Bharathidasan University, Trichy, India in 1996 and an M.E. degree in Mechatronics from Madras Institute of Technology, Anna University, Chennai, India in 2001. He completed his MBA degree from Madurai Kamaraj University, Madurai, India in 2003. Then he completed his Ph.D. degree from Anna University, Chennai, India in 2008. Presently, he is working as a Senior Lecturer in the Department of Mechanical and Industrial Engineering at Caledonian College of Engineering, Muscat. He has published 12 research papers in International Journals and more than 50 research papers in International and National conferences. His research focuses on Software Quality, Real time systems, Software Reliability, Mechatronics system Design and its Reliability Assessment and Virtual Instrumentation.
1.0 Introduction

Today, the software-based products are commonly used in many industrial systems. There are various methods used to develop the real time software for software based systems. It is most important to develop the real time software with high quality as early as possible to construct a software based systems and also to update the implementation of latest technologies. In large scale software, component based software development playing a vital role to reduce time and also to increase quality. The software development for Commercial Off The Shelf (COTS) software includes planning, analysis, requirements specification, design, implementation, assembly, testing and integration, operation and maintenance. As a consequence, time spent on development of new products or release of the new versions of software products must be reduced. One solution to this emerging problem is to reuse software design, components and solutions in new versions of systems and products. Besides shortening development time, properly handled reused software will improve the quality and specifically reliability since the code is executed for longer time and in different contexts (Fenton and Fleeger, 1996). Moreover, assembly of all the components needs more expertise to get a high quality in the COTS software. This paper aims to give an effective solution for component assembly in Component-Based Software Engineering (CBSE).

2.0 Literature Review

One-way of approaching the issue of reuse is to develop the computer-based systems by the Component-Based Software Engineering (CBSE) or Component Based Software Development (CBSD) is normally done by assembling software systems from components. The use of Commercial-Off-The-Shelf (COTS) products as elements of large software systems is becoming increasingly common place due to shrinking budgets, accelerating rates of COTS enhancement and expanding system requirements (Heineman and Councill, 2001). The integration of COTS software into software based systems introduces many important issues. The primary issue is the assembly of different software components into usable form. The consumers who want to use these systems want reliable software system within the budget and also time bound is an important issue in component-based software.

Different COTS software selection and integrations can be modeled as different instances of Bayesian Belief Networks (BBN) and changes to the architecture are modeled as simple changes to BBN. Using COTS component software system architecture, it can be modeled and diagnosed the problems in this architecture and use of these emergent properties to test the architectures compliance to measurable system requirements through scenario (Fenton and Neil, 1999). But this work does not focus elaborately about software assembly.
CBSD can help the software industry to realize the quality and productivity improvement similar to those achieved in electronic and mechanical component assemblies (Brown, 2000; Szyperski, 2003). Vitharana et al. (2003) suggested that the weightage must be given to the cost effectiveness with the ease of assembly model in software industries when the business strategy is low cost and component-based development. He also pointed that the design models with quantified technical features support the component developers’ business strategy. Architecture-based software approach in which the architectural analysis and code synthesis are combined together in order to efficiently and correctly assemble a system out of a set of already implemented components. Bucchiarone et al. (2006) have proposed a software architecture-based approach in which architectural analysis and code synthesis are combined together in order to assemble a system out of a set of already implemented components. According to the past literatures, there is no flexible standard or step by step procedure or proper algorithm to make a perfect assembly in COTS system. In 2007, Vijayalakshmi et al., tried to give a best solution for the component based software assembly in their paper. Vijayalakshmi et al (2007) proposed a new algorithm for component assembly using dependency chart. It is an excellent algorithm for minimum number of components and it is a time consuming algorithm for large scale software assembly. The dependency based algorithm is taken as a base paper for the proposed problem and it is proposed to solve by using genetic algorithm in this research work.

3.0 Problem background

Software assembly is the process of designing a software system from off-the-shelf components and combining them to form a functional system. This process involves two steps

1. Selection of suitable components from repository
2. Assembly

Components are chosen such that the user defined specifications are satisfied. These design specifications include quality, reliability, complexity, maintainability, safety and cost etc. During assembly, the engineer must be concerned with the functional and nonfunctional properties of the components and how they interact as a system. Selecting components for a given configuration is a time consuming task that includes searching through the entire vendors catalog and matching it with required specification. Component selection is a challenging task because of the size of the solution space and the process of evaluating designs in a large software development system. (Inverardi and Tivoli, 2003a,b). The size of the solution space is a function of the number of components that are existing. The most difficult problem in selection is to determine the characteristic of the available COTS products so that a component which meets the requirements may be
selected. The result of this process is to reveal the necessary information to make selection and identify possible sources of conflict and overlap, so that the component can be effectively assembled and evolved. The next process after selection is the assembly of the software components. Assembly of the components also needs more expertise to achieve appropriate quality and particularly reliability (Dolbec, J. and Shepard, T., 1995). The following are the objectives of the effective software assembly.

- To increase the reliability of the software by minimizing the defects.
- To find a well-defined infrastructure for component based software system.
- To make a better coordination between components in the software architecture.
- To avoid intricacies and complexities in software integration.
- To minimize the overall development time.
- To maintain flexibility of arrangement and operation of components.
- To obtain a cost effective software system.

So, it is most important to concentrate on software assembly and also it is needed to develop a proper algorithm for the assembly process.

4.0 Genetic Algorithm

Genetic algorithms are used to optimize the specified problem using computer simulations. GAs are used to search in the very large space efficiently because of its probabilistic nature rather than deterministic. GA is easy to implement and also to solve intractable problems quickly. GAs start with initial set of random solutions called the population. There is no rule to determine the population size. Then the population will converge to minimize the population size efficiently. Once the population size is chosen, the initial population is randomly generated. The GAs are using three genetic operators-reproduction, crossover, and mutation. The successive generation in GAs are defined using chromosomes. Normally chromosomes with higher fitness value are more likely to be selected. In this work, Genetic Algorithm is used to search the best assembly order in the reusable components of Component Based Software Development (CBSD). In the implementation of the GA in software assembly, dependency between the components is taken as important parameter which is defined by coupling factors.

4.1 Multilevel Genetic Algorithm

The multi-level Genetic Algorithms are used to optimize the specific problem more effectively. It uses two genetic algorithms, working at two different levels. A first level GA generates new assembly sequence
populations, based on Genetic Operator Probability Setting (GOPS). The second level GA optimizes and updates GOPS for the first level GA. Figure 1 illustrates our multi-level genetic assembly planner, in block diagram form. In this, first level assembly sequence selection is dynamically optimized by the second level (level-2) GA GOPS selection.

5.0 Dependency based component assembly

In software based system, more components are combined into larger systems, more and more interconnections exist in these collections and relationships between their components become more numerous. Effective and reliable use of these systems involves cooperative operation among the components. Since complex distributed computer systems are composed of larger number of interconnected heterogeneous components from multiple vendors, comprise of any component may potentially affect many other components in unexpected and unacceptable way. (Harry S. Delugach, 2008)
It is necessary for the components to collaborate with one another within the system. However collaboration should be kept to an acceptable minimum. If the components are highly interactive in the system and it is difficult to implement, to test and to maintain with in the given period. This restriction suggests that if highly dependent components are assembled initially, the complexity of developing the systems is greatly reduced. Vijayalakshmi et al (2008) proposed a technique to solve the assembly problem. They described the algorithm based on the following techniques

1. Calculating the coupling parameter for the components
2. Constructing the dependency chart
3. Calculating the component dependency rating for selected components
4. Applying the proposed algorithm for proper component assembly

5.1 Coupling between the software components

Coupling is a measure of interconnection among the components in a software system. Coupling depends on the interface complexity between components, the point at which entry or reference is made to a component and what data pass across the interface.

Component coupling provides an indication of the connectedness of a component to other components, global data and the outside environment. Dhama(1995) has proposed a coupling metric for component coupling that includes data, control flow coupling, global coupling and environmental coupling. According to this approach the following parameters have been considered for data and control flow coupling.

For data and control flow coupling:

\[ d_i = \text{number of input data parameters} \]
\[ c_i = \text{number of input control parameters} \]
\[ d_o = \text{number of output data parameters} \]
\[ c_o = \text{number of output control parameters} \]

For global coupling:

\[ g_d = \text{number of global variables used as data} \]
\[ g_c = \text{number of global variables used as control} \]

For environmental coupling:

\[ w = \text{number of calls from other components} \]
\[ r = \text{number of calls to other components} \]
Coupling parameter $m_{ij}$ between component $i$ and $j$ is calculated as

$$m_{ij}= d_i + (a \times c_i) + d_o + (b \times c_o) + g_d + (c \times g_c) + w + r \quad \text{-------------------------- (1)}$$

Values for $a$, $b$ and $c$ must be derived empirically.

Using these factors component coupling indicator $m_c$ is defined as

$$m_c= 1- (k/m)$$

$K$-Proportionality constant which is also derived empirically.

As the value of $m_c$ decreases overall component coupling increases.

The coupling parameter $m_{ij}$ is a simple weighting factor that reflects the desirable connectedness between any two components and is calculated using equation (1). The calculated coupling parameter is converted to five-point scale value. Components that have complicated or high volume of interactions with other components acquire a high coupling rating. Conversely those components which are independent have negative coupling rating applied. The following table shows the various coupling values between the components according to the above assumptions.

<table>
<thead>
<tr>
<th>Five-point scale</th>
<th>Symbol</th>
<th>Value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>25-30</td>
<td>A</td>
<td>5</td>
<td>Very high rate of Coupling</td>
</tr>
<tr>
<td>20-25</td>
<td>B</td>
<td>4</td>
<td>High rate of Coupling</td>
</tr>
<tr>
<td>15-20</td>
<td>C</td>
<td>3</td>
<td>Moderate</td>
</tr>
<tr>
<td>10-15</td>
<td>D</td>
<td>2</td>
<td>Low</td>
</tr>
<tr>
<td>5-10</td>
<td>E</td>
<td>1</td>
<td>Independent</td>
</tr>
</tbody>
</table>

Table 1 Coupling Values

5.2 Dependency Chart

The Dependency Chart is used describe the “closeness” between the components. The chart allows capturing the interconnection information in an organized fashion. The first step is to take the various software components from the software architecture and list them in the first column of the dependency chart. The matrix to the right of that list (2$^{nd}$ to 6$^{th}$) provides a cell for the coupling between the components that is to be calculated using equation (1). Table 2 shows a dependency chart for a software system with 6 components.
Table 2. Sample Dependency Chart

Using this dependency chart, dependency rating is calculated to select components for assembly.

This chart is used to find the coupling between the components and also to identify the proper sequence in the execution of genetic algorithm program.

5.3 Component Dependency Rating (CDR)

Component dependency rating is used to select the components for assembly in a specified sequence Genetic Algorithm. Using CDR, components are chosen based on the coupling value with other components. Component dependency rating is computed for all components based on dependency chart values. This is also given as an input in Genetic Algorithm program to identify the sequence effectively. Component dependency rating of a component is calculated as the sum of the numerical values of the coupling parameter with all other components.

$\text{CDR} = \sum m_{ij}$

For the given dependency chart (Table 2), the CDR values are given below.

<table>
<thead>
<tr>
<th>Component</th>
<th>Component Coupling</th>
<th>CDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B+D+C+D+E=4+2+3+2+1+</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>B+E+B+C+C=4+1+4+3+3</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>D+E+E+E+D=2+1+1+1+2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>C+B+E+C+E=3+4+1+3+1+1</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>D+C+E+C+A=2+3+3+3+5+3</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>E+C+D+E+A=1+3+2+1+5</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 3. CDR calculation for the given coupling rating in Table 2
6.0 Case Study for the Software Assembly Using GA

The proposed technique is applied to component based Enterprise Resource Planning (ERP) software for manufacturing application. The components were developed to check the applicability of the Genetic Algorithm. A typical ERP system will use multiple components of computer software and hardware to achieve the integration. The ERP systems (Figure 2) are designed to plan the utilization of enterprise-wide resources. The modules in an ERP include: Manufacturing, Supply chain, Financials, CRM, Human resources, Project management and Warehouse management. A key factor is the integration of data from all aspects of an organization. To accomplish this, an ERP system typically runs on a single database instance with multiple software components providing the various business functions of an organization.

![Figure 2 Sample ERP Architecture for a manufacturing application](image-url)
The software components include the functionalities as follows.

1. Manufacturing component (C1) include classes for Engineering, Bills of Material, Scheduling, Capacity, Workflow management, Quality control, Cost Management, Manufacturing Process, Manufacturing Projects, Manufacturing flow.

2. Supply Chain Management component (C2) consists of classes for Inventory, Order Entry, Purchasing, Product configuration, Supply chain planning, Supplier Scheduling, Inspection of goods, claim processing, Commission calculation.

3. Financial component (C3) contains classes for General Ledger, Cash Management, Accounts Payable, Accounts receivable, fixed assets.

4. Project management component (C4) includes classes for Costing, Billing, Time and Expense, Activity management.

5. Human Resources management component (C5) contains classes for Human Resources, Payroll, Training, Time & Attendance, Benefits.

6. Customer relationship management component (C6) contains classes for sales and Marketing, Commissions, Service, Customer Contact and Call Center Support.

7. Data Warehouse Providing data for all the components

6.1 Dependency chart Construction

Component coupling is calculated for all the components using equation (1). Coupling of manufacturing component is shown in Table 4. The dependency chart is constructed by converting the coupling values to five-point scale and assigning symbols from A to E. The component that has the highest number of calls from and to other components gets the highest rating. The dependency chart for the ERP system is shown in Table 5.
<table>
<thead>
<tr>
<th>Component Parameters</th>
<th>Financial management</th>
<th>Supply Chain management</th>
<th>Customer Relationship management</th>
<th>Human Resources management</th>
<th>Project management Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of input data parameters $d_i$</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Number of input control parameters $c_i$</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Number of output data parameters $d_o$</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Number of output control parameters $c_o$</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Number of global variables used as data $g_d$</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Number of global variables used as control $g_c$</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Number of calls to other components $W$</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Number of calls from other component $R$</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Component Coupling $M$</td>
<td>26.5</td>
<td>21.4</td>
<td>18.7</td>
<td>8.1</td>
<td>26.9</td>
</tr>
</tbody>
</table>

Table 4 Coupling values of manufacturing component (Source: Vijayalakshmi et al. 2007)

| MC | - | D | E | D | C | C |
| FC | C | - | C | C | C | D |
| SCM | E | B | - | C | D | C |
| CRM | B | C | B | - | A | E |
| HRM | C | E | C | E | - | A |
| PMC | C | B | E | B | C | - |

Table 5 Dependency Chart for Enterprise Resource Management

Dependency rating for the component is calculated by adding all the coupling values of the components in the dependency chart and shown in Table 6.
<table>
<thead>
<tr>
<th>Component</th>
<th>Component Coupling rating</th>
<th>CDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C+E+D+C+C=3+1+2+3+3</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>D+B+C+E+B=2+4+3+1+4</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>E+C+B+C+E=1+3+4+3+1</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>D+C+C+E+B=2+3+3+1+4</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>C+C+D+A+C=3+3+2+5+3</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>C+D+C+E+A=3+2+3+1+5</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 6 Dependency rating for ERP

6.1 Application GA to find the Assembly order

For the component assembly, it is planned to apply multiple GA for the effective assembly. In Multiple GA assembly plan, the chromosome will be coded as a string which consists of assembly sequence numbers. This comes under the permutation encoding scheme as shown below.

Chromosome: 3 2 1 4 5 6

Each number in the chromosome represents the assembly component name. In this work, the position based crossover method is used as crossover operator and shift left operator is used as mutation operator. When these genetic operators are applied in a single level GA, it is observed that the static Genetic Operator Probability Setting (GOPS) did not yield a better offspring after each generation. This motivated the author to adopt a multi-level genetic algorithm to overcome this problem.

In the application of multiple GA, Level 1 genetic algorithm is used to find the assembly sequence of the selected components. Then this result is analyzed using level 2 GA and optimizes the genetic operator probability settings to get a maximized total dependency rating.

Level 1 Genetic Algorithm:

The permutation encoding scheme is adopted to code the assembly of the software component order as explained above. Level 1 GA considers the Component Dependency Rating (CDR) based on the coupling between the components and it is used to evaluate the total dependency rating for each chromosome. A position based crossover is adopted and the new offspring is generated as explained in the below steps.

Step 1: Select several loci randomly from Parent 1, eg 2, 4 and 6 in Parent 1 in Figure 3.

Step 2: Generate partial offspring 1 by copying the selected genes from Parent 1 into their corresponding loci.

Step 3: Mark the genes in Parent 2 that have been selected in Parent 1.
Step 4: From the beginning of Parent 2, sequentially select the genes that have not been marked and copy into the vacancy of partial offspring 1 from the beginning.

Further, Shift left operator is used as the mutation operator to get the new sequence and which is explained below.

The structure of the shift left operator:

Chromosome $<<$ Number of Displacement

The shift left operator shifts all the bits left as per the first operand. The second operator indicates the number of bits to be shifted. The left most bit positions in the string will be inserted at the backend in sequence. Following is the example for the shift left operator

Chromosome : $1 \ 2 \ 3 \ 4 \ 5 \ 6$

Operand : String $A << 2$

Mutated String: $3 \ 4 \ 5 \ 6 \ 1 \ 2$

A roulette wheel selection scheme is adopted to form the mating pool. Genetic Operator Probability Setting for the first level GA is updated by the second level GA during the function evaluation of the second level GA. In second level GA, real coded strings of GOPS (crossover probability and mutation probability) are used as the chromosome string. The roulette wheel selection scheme is also adopted in level 2 GA. Intermediate crossover method is applied to get a new offspring in real coded string. An incremental operator is used to mutate the offspring. A static probability setting is used in the second level GA. The performance of the second level GA was tested using Himmelblau's function. Himmelblau's function is one of the well-known multimodal functions commonly used to test the optimization algorithms. The above procedure is adopted for the ERP case study and coded in the MATLAB software and found the sequence of the assembly order. The convergent plot for the
assembly sequence which is received as the result from the software is given in the figure 4. The best sequence from GA is received as 546123, which yields the total dependency rating of 19. The sequence is HRM, CRM, PMC, MC, FC then SCM.

![Figure 4 Convergent Plots for the Assembly Sequence](image)

7.0 Discussion

Experimental results show that the multi-level genetic assembly sequence planner solves component assembly problem more quickly and reliably than prior classical single-level GA approaches. This result gives a better idea for the component based assembly for large scale software systems. In the Genetic based assembly, it is possible to assemble any number of components with the consideration of dependency rating. According to the base paper (Vijayalakshmi et al, 2007) the assembly sequence are HRM, CRM, PMC, SCM, MC, and FC. But, according to the GA, assembly sequence is coming as HRM, CRM, PMC, MC, FC and then SCM. In both the methods, assembly order of the components for the first three stages is same. In the fourth stage, there is a difference between SCM and FC components.

In the application of the dependency algorithm, Vijayalakshmi et al discussed about the criticality in the component order 4. In the dependency algorithm, there was a tie between the components in the stage four and the priority was given to the total number of lines. But it may not be true in the real situation. The dependency is the main criteria for the assembly of the components. So, this is taken as main parameter in the GA which is
solved more effectively than dependency based algorithm. So, the order from the fourth stage is changing and also it is believed that this assembly order is the right procedure to make the assembly sequence effectively.

8.0 Conclusion

This paper presents GA based assembly technique for the software based systems which is using large number of software components. The proposed algorithm can be applied to any number of software components and flexible for any component based software assembly. The algorithm is highly suitable for rapid development of large software systems. A simple case study for ERP system is used to illustrate the applicability of the proposed approach, and also, it can be extended to any system. The GA based software assembly is a new initiation in component assembly, and highly suitable for software industries.

References