Optimal Decision “Build-or-Buy” for Component Selection for a Modular Software System

Shivani Bali¹, Vijay M. Ghantasala² and P.C. Jha³

¹Lal Bahadur Shastri Institute of Management, Dwarka, Delhi, ²DST Worldwideservices, Hyderabad, ³Department of Operational Research, University of Delhi, Delhi

ABSTRACT

Software Architecture is a complex process where architects have to face very challenging task of building the software that meet the constraints on quality, cost and reliability etc. After the advent of object oriented software, the dynamics of the software development has changed tremendously. Decision to choose the right components become extremely difficult because of the number of parameters to be considered while making the decision. If suitable components are not available, then the decision becomes very difficult further on whether to build the component in-house or to get this from the vendor. This paper focuses on the formulation of the optimization model that supports the decision whether to buy software components or to build them in-house upon designing a software structure. This paper presents an optimization model of software system. We devise a Bi-criteria optimization model based on decision variables indicating the set of structural components to buy and to build in order to minimize the software cost with simultaneously maximizing the system reliability.

KEYWORDS

Write key words used in your paper here. Write key words used.

1. INTRODUCTION

Software Architecture is a complex process where architects have to face very challenging task of building the software that meet the constraints on quality, cost and reliability etc. After the advent of object oriented software, the dynamics of the software development has changed tremendously. Similar to the other product domains where products are built by assembling various readily available or COTS components, software product development has seen the shift the traditional development approach and started using the components which are readily available in the market. These components are build separately and are assembled to make the product. Issues like which type of the software and component is needed, how much cost one has to pay, time for delivery etc are some of the key challenges faced by software architects. Decision to choose the right components become extremely difficult because of the number of parameters to be considered while making the decision. If suitable components are not available, then the decision becomes very difficult further on whether to build the component an in-house or to get these from the vendor. Even though in the last years numerous tools have been introduced to support decisions in different phases of the software lifecycle, the selection of the appropriate set of components remains a hard task to accomplish [1]. Functional requirements are easy to accomplish. There are models to select the COTS components based on the functional requirements they satisfy and also it is easy to verify the alternatives for such components.

Some relevant progresses have been made in building software architectures that meet their functional requirements. As an example, the automated synthesis of software connectors is becoming a well-assessed technique to improve functional compatibility between components. On the other end, only very recently methodologies appeared to support the automation of an assembling process that leads to software architectures able to meet their non-functional requirements, such as performance and reliability [2]. In practice, software projects suffer from limited budgets, and the decisions taken from software developers are heavily affected by cost issues. The best solution often might not be feasible due to high costs, and wrong cost estimation may become a critical impact factor for the project success. Therefore, the introduction of tools that able to drive (since the architectural phase) software developer decisions to meet functional and non-functional requirements, while keeping the costs within a predicted budget, would be very helpful to the software community.[2] Software developers always have to decide how non-functional requirements such as the performance of the developed product or deciding the components are critical with respect to achieving the required performance level throughout the development cycle. Also parameters like delivery time, estimated cost etc are some of the parameters which will have a key role in deciding the component selection. Developing a product requires huge budget. Organizations many times face the dilemma whether to build the product completely or get it from the available market source. Sometimes organizations may not have the sufficient technical skills to build the product. Even if they have the money, it makes sense to use the experts to help them in building the product. In the recent past, COTS based software development has become a necessary new activity in the development process. Experience of the development team members is very useful in selecting the...
components. In selecting the components, normally decision-making algorithms are not being used.

In this paper, various models which will help in selecting the components based on the non-functional requirements is discussed. Models which use non-functional requirements were discussed by V Cortellessa et.al [2] in their paper.

1. NOTATIONS

R : System quality measure
n : Number of components in the software architecture S
m : Number of COTS instances available for each component
T : Committed time to assemble the system

\[ N_{ij}^{tot} \] : Total number of tests performed on the in-house developed instance (i.e. alternative j of module i).

\[ N_{ij}^{suc} \] : Number of successful (i.e. failure free) tests performed on the in-house developed instance.

1.2 ASSUMPTIONS

1) Budget is limited
2) Each component might be developed in-house
3) For each component instance of a software architecture, several instances are available as COTS products
4) All COTS instances of the same component can be considered functionally equivalent
5) Adoption cost is embedded in the component cost
6) Delivery time, reliability attributes of interest of each COTS instance are assumed to be given from the vendor.
7) Development Cost, delivery time and reliability attributes of in-house instance can be estimated
8) One of the available COTS products must be chosen for components that cannot be in-house instance built.

2. COST MINIMIZATION UNDER DELIVERY TIME AND RELIABILITY CONSTRAINTS

Let \( S \) be a software architecture made of \( n \) modules, with a maximum number of \( m \) COTS alternatives available for each module.

2.1. COTS COMPONENT MODEL PARAMETERS

The parameters that we define for a COTS product are:

- Cost of the component, \( C_{ij} \)
- Delivery time, \( d_{ij} \)
- average number of invocations, \( s_{ij} \)
- probability of failure on demand, \( \mu_{ij} \)

The following expression is used to estimate the cost.

\[ C_{ij} = C_{ij}^{buy} + C_{ij}^{adopt} \]

Where \( C_{ij}^{buy} \) is the purchase cost and \( C_{ij}^{adopt} \) is the adoption cost.

Adoption cost takes into consideration for making the adopted software to work correctly. The changes required to make this work is represented by this.

2.2 IN-HOUSE COMPONENT MODEL

The parameters that we define for an in-house component are:

- Cost of the development, \( C_i \)
- Estimated development time, \( t_i \)
- Average time required to perform a test case, \( \tau_i \)
- Average number of invocations, \( s_i \)
- Probability that a single execution of software fails on a test case chosen from a certain input distribution, \( \pi_i \)

3. MODEL VARIABLES

In general, a Build-or-buy decisional strategy can be described as a set if 0-1 variables defined as follows, (for all \( i = 1, 2, .., n \))

\[ y_i = \begin{cases} 1 & \text{if the } i^{th} \text{Component is in-house developed} \\ 0 & \text{Otherwise} \end{cases} \]

\[ x_{ij} = \begin{cases} 1 & \text{if the } j^{th} \text{COTS instance of the } i^{th} \text{ component is chosen} \\ 0 & \text{Otherwise} \end{cases} \]

The following equation represents this constraint as well as the constraint that at most one COTS instance is bought for each development.

\[ y_i + \sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, 2, .., n \]

Finally, let \( N_{ij}^{tot} \) be an additional integer decision variable of the optimization model that represents the total number of tests performed on the in-house developed instance of the \( i^{th} \) component. The effect of testing on cost, reliability and delivery time of COTS products is instead assumed to be accounted in the COTS parameters. Based on the testability definition, we can assume that the number \( N_{ij}^{suc} \) of successful (i.e. failure-free) tests performed on the same component can be obtained as

\[ N_{ij}^{suc} = (1 - \pi_i)N_{ij}^{tot} \]

and it will be used to build the reliability constraint.

3.1 COST OBJECTIVE FUNCTION

The cost of the in-house instance \( C_i \) can be expressed as.

\( \overline{C_i t_i + \tau_i N_{ij}^{tot}} \)

The objective function to be minimized, as the sum of the costs of all component instances selected from the “build-or-buy” strategy, is given by

Minimize \( C = \sum_{i=1}^{n} (c_i (t_i + \tau_i N_{ij}^{tot}) y_i + \sum_{j=1}^{m} C_{ij} x_{ij}) \)

3.2 DELIVERY TIME CONSTRAINT

A maximum threshold \( T_i \) has been given on the delivery time of the whole system. In case of a COTS product the delivery time is simply given by \( d_{ij} \), whereas for an in-house developed instance the delivery time shall be expressed as \( t_i + \tau_i N_{ij}^{tot} \)

Therefore, the following expression represents the delivery time \( T_i \) of the component \( i \):

\[ T_i = (y_i (t_i + \tau_i N_{ij}^{tot}) + \sum_{j=1}^{m} d_{ij} x_{ij}) \]
With the assumption that sufficient manpower is available, the delivery constraint can be reformulated as:

$$\text{Max} \quad (T_i) \leq T$$

Which can be decomposed in the set of constraints

$$T_1 \leq T, \ldots, T_n \leq T$$

Although this assumption could not be too realistic, due to the overhead that likely incurs when a lot of components must be developed in-house, it reflects a common practice in production planning. In fact, resource capacity is considered as unbounded in the strategic and tactical planning where build or buy decisions have to be typically taken. Hence, more details on capacity and workload constraints are addressed into scheduling models for operational planning.

3.3 RELIABILITY CONSTRAINT

Failures that compromise the behavior of the whole system are only considered here. A minimum threshold $R$ has been given on the reliability on demand of the whole system. The reliability of the whole system can be obtained as a function of the probability of failure on demand of its components. The probability $\mu_i$ of failure on demand for COTS components has been discussed in the previous sections. In this section the probability of failure on demand for in-house components is formulated as a function of the testability $s_i$ and of the number $N_i^{\text{tot}}$ of successful test cases performed following the component operational profile. The possibility of reducing the probability that the component $i$ fails by means of a certain amount of test cases (represented by the variable $N_i^{\text{tot}}$) is expressed. Probability of failure on demand of an in-house developed component instance, under the assumption that the on-field users’ operational profile is the same as the one adopted for testing [3]. Let A be the event “$N_i^{\text{inc}}$ failure-free test cases have been performed” and B be the event “the instance is failure free during a single run”. If $\rho_i$ is the probability that the in-house developed instance is failure free during a single run given that $N_i^{\text{inc}}$ test cases have been successfully performed, from the Bayes’ theorem we get

$$\rho_i = P(B/A) = \frac{P(A/B)P(B)}{P(A/B)P(B) + P(A/B)P(B)}$$

The following equalities come straightforward:

$$P(A/B) = 1, \quad P(B) = 1 - \pi_i,$$

$$P(A/B) = 1 - \pi_i, \quad P(B) = (1 - \pi_i)^{N_i^{\text{inc}}},$$

$$P(B) = \pi_i, \quad$$

Therefore we have

$$\rho_i = \frac{(1 - \pi_i)}{1 - \pi_i + \pi_i (1 - \pi_i)^{N_i^{\text{inc}}}}$$

The probability of failure on demand of an in-house developed instance can be expressed as $(1 - \rho_i)$. The average number of failures $f_i$ of the component $i$ is follows:

$$f_i = (1 - \rho_i)s_i + \sum_{j=1}^{m}\mu_i s_i x_{ij}$$

The probability that no failure occurs during the execution of the $j^{th}$ component is given by $\phi_i = e^{-\phi_i}$ which represents the probability of no failures occurring in a Poisson distribution with parameter $\phi_i$. Therefore the probability of a failure-free execution of the system is given by $\prod_{i=1}^{n}\phi_i$

The reliability Constraint is the given by

$$\prod_{i=1}^{n}\phi_i \geq R_0$$

4. FORMULATION OF THE PROPOSED PROBLEM

The model solution provides the optimal “build-or-buy” strategy for component selection, as well as the number of tests to be performed on each in-house developed component. The solution guarantees a system reliability on demand over the threshold $R_0$, a system delivery time under the threshold $T$ while minimizing the whole system cost. The applied reliability model is a light-weighted one, as we work in favor of model solvability. However, it can be replaced by a profound reliability growth model from literature to increase the result accuracy. This can be done without essentially changing the overall model structure, with the side effect of increasing complexity. The main driver for the decision is Cost, Delivery time and Reliability. In taking the decision to buy or build, Cost is no doubt the primary driver however we cannot compromise on reliability at timely delivery of the software. This means this problem can be reformulated as Bi-Criteria optimization

Problem (P1):

Minimize $F_1(X) = \sum_{i=1}^{n}(c_i(t_i + \tau_i N_i^{\text{inc}}))y_i + \sum_{j=1}^{m}c_j x_{ij}$

Maximize $F_2(X) = \prod_{i=1}^{n}\phi_i$

Subject to $X \in S = \{x_{ij} \text{ and } y_i \text{ are binary variable/}

$T_i = ((Y_i(t_i + \tau_i N_i^{\text{inc}}) + \sum_{j=1}^{m}d_j x_{ij}) \leq T \quad i = 1,2,\ldots,n$

$y_i + \sum_{j=1}^{m}x_{ij} = 1 \quad \text{and } \phi_i = e^{-\phi_i} \quad i = 1,2,\ldots,n$

$f_i = (1 - \rho_i)s_i + \sum_{j=1}^{m}\mu_i s_i x_{ij} \quad i = 1,2,\ldots,n$

$N_i^{\text{inc}} = (1 - \pi_i) N_i^{\text{tot}} \quad i = 1,2,\ldots,n$

$\rho_i = \frac{(1 - \pi_i)}{(1 - \pi_i) + \pi_i (1 - \pi_i)^{N_i^{\text{inc}}}} \quad i = 1,2,\ldots,n$

5. NORMALIZATION
The optimization model is Bi-criteria optimization problem in which one hand system reliability is maximized and on other hand cost of selected components to form / assemble the system is minimized. The reliability which is unit free is measured between zero and one whereas cost has its unit. Two objectives can be converted to single objective programming problem either if both objectives are of same unit or if both objectives can be made unit free. To make cost function unit free, the following transformation is used.

\[ \frac{C_{ij}}{c + c}, \quad \frac{c_i}{c + c} \]

The problem (P1) further be rewritten as vector optimization problem.

**Objective:** Vector Max \( F(X) \)

Subject to \( \lambda \in S \)

where \( F(X) = (F_1(X), F_2(X))^T \)

**Finding Properly Efficient Solution**

**Definition 1 [4]:** A feasible solution \( X^* \in S \) is said to be an efficient solution for the below problem if there exists no \( X \in S \) such that \( F(X) \geq F(X^*) \) and \( F(X) \neq F(X^*) \)

**Definition 2 [4]:** An efficient solution \( X^* \in S \) is said to be a properly efficient solution for the problem (P2) if there exist \( \alpha > 0 \) such that for each \( r \)

\[ \left( F_r(X) - F_r(X^*) \right) \leq \alpha \left( F_j(X) - F_j(X^*) \right) \]

with \( F_j(X) \leq F_j(X^*) \) and \( F_r(X) > F_r(X^*) \) for \( X \in S \).

Using Geoffrion’s scalarization (P2) the problem reduces to

**Problem (P3)**

Maximize \( \lambda_1 F_1 - \lambda_2 F_2 \)

Subject to \( \lambda_1 + \lambda_2 = 1 \)

**Lemma 1 (Geoffrion[5]):** The optimal solution of the problem (P4) for fixed \( \lambda_1 \) and \( \lambda_2 \) is a properly efficient solution for the problem (P3) and consequently (P1).

4 Numerical Illustration

The problem is solved using software package LINGO (Thiriez [6]) for data given in table-1 and table-2. Following solution is obtained.

**Case 1:** Delivery time is assumed to be 20 units.

\( x_{13} = x_{24} = x_{31} = 1 \)

It is observed that all the alternatives are COTS components. The system reliability for the above solution is 0.95 and cost is 22 units.

**Case 2:** Delivery time is assumed to be 30 units.

\( x_{12} = x_{22} = y_{32} = 1 \)

It is observed that both inbuilt and COTS alternatives are chosen. The system reliability for the above solution is 0.97 and cost is 19.5 units.

**REFERENCES**


<table>
<thead>
<tr>
<th>Module</th>
<th>CO TS Alternatives</th>
<th>Cost</th>
<th>Deliver time</th>
<th>Averag number of invocations</th>
<th>Prob. of failure on demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Cij</td>
<td>dij</td>
<td>sij</td>
<td></td>
<td>( \mu_{ij} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>180</td>
<td>0.0002</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.5</td>
<td>4</td>
<td>180</td>
<td>0.0002</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>180</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>20</td>
<td>0.0002</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>20</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>20</td>
<td>0.0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>60</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>10</td>
<td>60</td>
<td>0.0004</td>
<td></td>
</tr>
</tbody>
</table>

Table-1
Optimal Decision “Build-or-Buy” for Component Selection for a Modular Software System

Table-2

<table>
<thead>
<tr>
<th></th>
<th>Development time, $t_i$</th>
<th>Testing time, $\tau_i$</th>
<th>Unitary development cost, $c_i$</th>
<th>Average number of invocations, $s_i$</th>
<th>Testability, $\pi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
<td>0.05</td>
<td>1</td>
<td>180</td>
<td>0.0002</td>
</tr>
<tr>
<td>$C_2$</td>
<td>3</td>
<td>0.05</td>
<td>1</td>
<td>20</td>
<td>0.0002</td>
</tr>
<tr>
<td>$C_3$</td>
<td>5</td>
<td>0.05</td>
<td>1</td>
<td>60</td>
<td>0.0002</td>
</tr>
</tbody>
</table>