Research Article

Mapping Component Diagram to Colored Petri Net for the Evaluation of Maturity Non-Functional Parameter

Amir Hossein Toni
Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
Email_amirtoni@hotmail.com

ABSTRACT:

The quality of Software is a fundamental matter in Software’s Development. Having a high quality design in software systems will be very useful in obtaining non-functional parameters. One of the most common tools in software design is the Unified Modeling Language and it is inherently suffers from the disadvantage of being semi-formal. This weakness will be effective in obtaining non-functional parameters. Many efforts and researches have been made to convert this semi-formal language into formal methods which among them as an example, we can name Petri Net. This article presents an algorithm for conversion of component diagram into equivalent Petri Net and it will be followed step by step. This mapping will be presented formalization of the Unified Modeling Language. Using formalization, validation and evaluation of non-functional parameters such as maturity and reliability in designing system may be possible.

Keywords: UML, Component Diagram, Petri Net, non-functional parameter, formalization

INTRODUCTION

Software engineering is a systematic approach to software development. Producing reliable and affordable software requires an appropriate architecture. Today, the design and software architecture are known as one of the most important branches of Software Engineering. The design quality of software systems is based on achieving non-operational parameters. If the software architecture before entering the implementation phase did not analysis and implement very well, using the architecture will have high risks and it is possible the resulting system fails to meet the objectives and it leads to wasting a lot of resources of the project such as time and cost. Today the Unified Modeling Language is not only the most important but also it is the most common tool for modeling engineering software architecture. According to the plausibility that the Unified Modeling Language or UML has among software engineers, however, it suffers from an inherent weakness which is being semi-formal. The weakness appears in the evaluation of non-operational parameters and validation. Due to this lack in these diagrams non-functional parameters such as Maturity and Reliability cannot be assessed before the implementation phase in the production cycle of software. Therefore, Petri Nets due to the mathematical basis are a good way to formalize this language.

The main objectives of this research are formalization of component diagram and evaluation of non-functional parameter software maturity and the impact they have on the reliability parameter on this diagram by using colored Petri nets. Many studies have been...
conducted to cover weaknesses of UML. Some of these studies used only algorithms mapping and conversion Diagram. These methods have changed the UML model to mathematical model and formal Petri nets that have a visual aspect as well and have done the validation operations more accurately and quickly [1]. In [2] presented an algorithm of sequence Diagram mapping to fuzzy Petri nets and ultimately assessed the reliability of the diagram. [3] deals with the modeling component diagram in the Unified Modeling Language using Petri nets. In [4] evaluation of parameter for understanding component diagrams were discussed, also [5] has been elaborated the evaluation of fault tolerance parameter in component diagrams by mapping it into colored Petri nets. In [6] changed the state fuzzy diagram into fuzzy Petri nets and with this process added the ability to assess non-operational parameters into the system.

1. Unified Modeling Language
Producing reliable and affordable software requires a proper architecture. The Software architecture word does not have a clear and precise definition. Software architecture for an application is computing system of the structure or structures of system which includes software components, observable characteristics of each of the components and their relationships [7]. Software architecture can be modeled by several tools. The reason is that modeling complex systems cannot conceptualized the system once and for all therefore, to fully understand and finding and displaying the relationship between different parts of the system, we discuss modeling. One of the most widely used modeling tools in the field of software engineering is the Unified Modeling Language. Unified Modeling Language is a language for specifying, visualizing, constructing, and documenting software and non-software systems and modeling business systems. Unified Modeling Language is a modeling language which is used for object-oriented analysis and design [8]. This language has several diagrams which used according to the application.

This study focuses on component diagram, including structural diagrams of the Unified Modeling Language.

1.1. Component Diagram
Component Diagram described how to split the system into its components and dependencies between the components of the system. Component diagram indicates relationship between components of the system. A component is an interchangeable unit of the system that hides details and its behavior shown by some interfaces and provides a physical view of the system. The purpose of this diagram is to show dependencies of a software component on other software components or even on components of other software on the system [9]. The major and important parts of component diagram are shown in Table 1.

Table: 1. Major and important parts of component diagram

<table>
<thead>
<tr>
<th>Name</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td><img src="image" alt="Component" /></td>
</tr>
<tr>
<td>Provided Interface</td>
<td><img src="image" alt="Provided Interface" /></td>
</tr>
<tr>
<td>Required Interface</td>
<td><img src="image" alt="Required Interface" /></td>
</tr>
<tr>
<td>Port</td>
<td><img src="image" alt="Port" /></td>
</tr>
</tbody>
</table>

1.2. Colored Petri Net
Petri Nets are a powerful tool for modeling. Petri Nets in addition to have a nominal structure and behavior, they have the ability to display graphics; that's why they make it easy for modeling. In Petri nets there are nodes as circles and lines, that circles represent the place and lines marker the transition. These place and transition are connected to each other by arc. To describe the behavior of Petri net signs are added to the graph. Adding signs make that we can define the concept...
Mapping Component Diagram to Colored Petri Net for the Evaluation of Maturity Non-Functional Parameter

of state in this graph [10]. Figure 1 shows the Petri Net components.

![Fig 1: Petri network components](image)

**METHODS**

Here an algorithm for mapping individual components will be described in the component diagram of Petri net [3]. This algorithm has the following steps so the component diagrams can be converted to Petri net according to the following steps.

2.1. Refining the Component

For determining the scope of the components behavior we use the formalizer of the phrases of tracks. We used path expressions to determine the order of operations between the components. These symbol used to display the main and primary operation as order (\(;\)), selection (\(,\)), repetition and more repeat for zero (\(*\)), repeated for at least one more repetition (\(+\)), parallel (\(||\)) and parentheses used to expand the impact. By adding the path expressions, we limit the responsibility of component and the restriction automatically covers cooperation components. Figure 2 illustrates this point.

![Fig 2: path phrases in the graph component](image)

2.2. Conversion of each component to Petri net equivalent

By adding states to the component operations we have a brief display of operation of each component. Petri net corresponding to Fig. 2 is drawn in Figure 3.

![Fig: 3. Diagram of component Petri net](image)

2.3. The combination of Petri nets

Components mixed together based on their interface. In this compound, as Figure 5 shows, Client components by using a shared place are connected to the transition of server component
and the output of this transition connects to the place of client components.

Fig: 5. Interaction between components

2.4. Evaluation of maturity parameter of system

In this paper quality model ISO / IEC 9126-3 is considered as a qualitative model. The model is a quality standard model that offered for both levels of character and sub-character. The main features of this model include performance, reliability, usability, efficiency, portability and maintainability of the system's capabilities. As mentioned, Reliability is one of the main characteristics of this model. This character has following sub-characters such as maturity, fault tolerance and recoverability that in this study we discussed the maturity system of in this model. This character includes the following metrics such as Fault Detection and Fault Removal to assess the maturity of software. Each metric respectively expressed are calculated by equation (1) and equation (2) [11].

Fault detection in fact, as its name suggests, checks the number of errors found in the system. This metric by counting the number of errors found in comparison and it is calculated by estimating number of errors per each step. The higher value of this metric represents a desirable level of quality in the system. Fault detection is achieved by the following equation [12, 13].

\[ X1 = \frac{NA}{NE} \]

NA: The number of errors found in Checking
NE: The estimated number of expected errors

Equation (1)

Fault Removal metric measures the number of errors discovered during checking which have been modified. This metric was calculated by counting the number of errors corrected during the review and comparison with the number of errors discovered in the checking. Higher result indicates higher number of errors that has been eliminated. Correcting the error is calculated by equation (2) [12, 13].

\[ X2 = \frac{NA}{ND} \]

NA: The number of corrected errors in Checking
ND: The number of errors discovered during Checking

According to the metric introduced, the maturity of software system is calculated by equation (3) [13].

\[ Mi = \sum_{i=1}^{n} W_i \times X_i \]

Mi: Metrics for system maturity
Wi: The weight of each metrics
Weight represents the importance of each metric in different parts of component.

To calculate each system metrics in Petri net, we define a success rate ‘f’. The success rate determines the possibility of shooting transmission. On the other hand, f-1 will be failure rate. If transmitting the rate that shows shoots, the success rate will change to D*f. (D is the result of the success rate up to that point). You can see the description in Figure (6). Min and max are calculated according to the statement of the problem.

Fig: 6. Evaluation of metrics on Petri net
[III] RESULTS

3.1. Case Study

The system under study is related to the online ordering system. Diagram component is designed in such a way that includes forms and major and standard concepts. In the studied system there are three components which were interconnected by the relevant interfaces. Implementations of the components have been identified by path phrases. The system acts in this way that initially the user enters the username and password in the form displayed by the component order. After connecting to the list of products, you will search the desired product, in case of availability of the product, order and in the absence can check similar products. Figure (7) shows system component diagram in the study along with path expressions. Figure (7) presented in Appendix. In Figure (8) you can see Petri net equivalent component diagram Figure (7). This network is expressed by conversion algorithm which has been made step by step. Figure (8) presented in Appendix

Entries in Figure (8) marked with the name of V11 to V31 are embedded to calculate metrics in each stage. To calculate the fault detection metric we use equation (1). Inputs identified in each metric represents a value of it at that stage. According to the way shown in Figure (6), entries values are multiplied during the step and give the final value of that metric system. How to calculate fault detection (Vx1) are followed as table 2 and Table: 3 initial entry values.

Table : 2. Input and output of fault detection metric

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>V11</td>
<td>0.95</td>
</tr>
<tr>
<td>V12</td>
<td>0.95</td>
</tr>
<tr>
<td>V21</td>
<td>0.92</td>
</tr>
<tr>
<td>V22</td>
<td>0.90</td>
</tr>
<tr>
<td>V23</td>
<td>0.90</td>
</tr>
<tr>
<td>V24</td>
<td>0.95</td>
</tr>
<tr>
<td>V31</td>
<td>0.96</td>
</tr>
<tr>
<td>V32</td>
<td>0.96</td>
</tr>
<tr>
<td>V33</td>
<td>0.96</td>
</tr>
<tr>
<td>Vx1</td>
<td>0.61</td>
</tr>
</tbody>
</table>

For calculating the metric of fault removal we use equation (2). Calculation methods are as the previous method. The results are shown below.

Table: 3. Input and output of fault removal metric.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>V11</td>
<td>0.95</td>
</tr>
<tr>
<td>V12</td>
<td>0.95</td>
</tr>
<tr>
<td>V21</td>
<td>0.95</td>
</tr>
<tr>
<td>V22</td>
<td>0.85</td>
</tr>
<tr>
<td>V23</td>
<td>0.91</td>
</tr>
<tr>
<td>V24</td>
<td>0.93</td>
</tr>
<tr>
<td>V31</td>
<td>0.97</td>
</tr>
<tr>
<td>V32</td>
<td>0.94</td>
</tr>
<tr>
<td>V33</td>
<td>0.95</td>
</tr>
</tbody>
</table>

The final amount of metric error detection and fault removal, are respectively, 0.61 and 0.60. To calculate the maturity of system, we proceed according to equation (3).

\[
\frac{U(Vx1) = \frac{(0.61 * Vx1) + (0.60 * Vx1)}{2}}{2}
\]

According to the type of system, the amount of weight of each metric depends on the importance of metrics of studied system that here the significance of both Metrics meant for us is 1 which considered hundred percent. As you can see, the final result is equal to 0.605. In order to have higher maturity system, system developers must increase the maturity of each component that this value in software development cycle and multiple iterations can be expanded in RUP process.

[IV] DISCUSSION

The Conversion of semi-formal model of UML model to formal Petri net will enable validation and evaluation of non-functional parameters. This mapping will have benefits, such as formalization, visualization, automated process and feasibility. As previously stated maturity of system is one of the characteristics of reliability in quality model ISO / IEC 9126 and it has direct relationship with it. According to the output of case study it can be concluded that the higher value of the character is much correlated with greater system reliability. By using this model non-functional parameters are
able to be calculated that evaluation of these parameters can have great significance on implementation phase.

[V] CONCLUSION
As future work other UML diagrams can be converted to Petri networks and other formal models. We can also evaluate other non-operational parameters in diagrams. Designing and construction of software tools for mapping from UML diagrams to Petri net that increase the speed of conversion of these diagrams will have some sort of automation.

REFERENCES
Mapping Component Diagram to Colored Petri Net for the Evaluation of Maturity Non-Functional Parameter

APPENDIX

Fig: 7. Component diagram of online ordering system

Fig: 8. Petri net equivalent to component diagrams of online ordering system