Towards Modeling and Validating Analysis Processes for Software Adaptation

Xiangping Chen  
National Engineering Research Center of Digital Life,  
Institute of Advanced Technology, Sun Yat-sen University,  
Guangzhou, 510006, China  
Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education,  
Beijing, 100871, China  
Chenxp8@mail.sysu.edu.cn

Gang Huang  
Key Laboratory of High Confidence Software Technologies, Ministry of Education,  
School of Electronics Engineering and Computer Science,  
Peking University,  
Beijing, 100871, China  
Huanggang@sei.pku.edu.cn

Lingshuang Shao  
Computer School of Wuhan University,  
Wuhan, 430072, China  
shaolsh@gmail.com

Abstract—With the complexity and the dynamic analysis requirements during software adaptation for software in the open and dynamic environment, the software analysis requires integrating multiple analysis methods and controlling the execution of analysis methods in specific sequence. As a result, the analysis process is hard to be communicated to its designers, implementers and users without an explicit model. In addition, if dependent methods are not included or encounter runtime failure, the data dependency and control dependency between analysis methods may cause correctness problem. In this paper, we proposed a domain-specific language for modeling the analysis process using subset of notations from BPMN. In order to cope with the dependency problem in model and implementation level, the dependency relationships are divided into static dependency and dynamic dependency. Dependency identification and matching algorithm is provided to assure the static dependency in the analysis process model. For the dynamic dependency, we provide a service to detect the analysis failure caused by dynamic dependency. We model and validate two analysis processes supporting software adaptation with performance and reliability concerns, and apply the analysis processes in analyzing a running system ECPerf.

Keywords- software analysis; analysis process, process validation, analysis process modeling

I. INTRODUCTION

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Software analysis technology is widely used during software lifecycle, and generates descriptive, predictive or prescriptive analysis results regarding to the system consistency, correctness, quality attributes and so on.

More and more software systems are running in the internet which is open and dynamic. In order to provide continues and high quality services in the internet, software may change the structure or behavior according to the changing user requirements and environment [1]. The increasing complexity of the software system brings challenges to the software analysis.

Software analyses play an important role during software adaptation. A general adaptation process contains four stages: monitoring, analysis, planning and execution. In the monitoring stage, the system model generated by monitoring system is usually low-level model containing raw data of different system aspects. These models are hard to be used without a transformation to high-level model or domain-specific model. In the analysis and planning stages, the system designer usually employs analysis methods to acquire status of the system, possible adaptation strategies and change impact, so as to gain a better understanding for decision making. For the information of different kinds: descriptive, predictive or prescriptive, usually more than one method are needed. Even in the execution stage, analyses of platform specific features and execution strategies are usually required for assurance of adaptation correctness.

During the software adaptation, multiple analysis methods are used and carried out in specific sequence. For example, an analysis method for adaptation planning may be used after the analysis methods for system quality attributes. In addition, the analysis results may affect the choice of analysis methods afterward. For example, the quality attributes acquired in the analysis stage may affect the choice of adaptation planning methods. If there is security problem, the designer may choose to analyze the possibility of adding fault-tolerant ability. If the system’s performance is too low to fulfill stakeholders’ requirement, the designer may choose to see possible reconfiguration with higher performance.

The analysis requirements for software adaptation bring new challenges. First, the analysis process is complex that it contains several related analysis methods. The set of analysis methods is chosen according to the system status and these methods should be carried out in specific sequence. The analysis process is repeatedly executed by hand. As a result, the execution process is error prone and hard for reuse. Second, the assurance of the analysis process correctness is hard because of complex relationships between analysis methods. There are two kinds of dependency relationship: the data dependency and control dependency. Data dependency means that an analysis method’s input relies on the output of another analysis method. Control dependency happens when an analysis method’s output decides whether another analysis method is executed. Thus, if an analysis method encounters runtime failure and can not
provide proper result for other analysis methods, the analysis process could not generate correct results or even could not finish execution.

Recent works on the integration of analysis methods have proposed methods and framework to ease the integration of analysis methods. These works focus on extracting proper model for analysis, controlling external analysis tool to carry out analysis task, and synthesizing analysis results back to the core model. Among these works, ADD [7] provides guidelines and standard interface for integration of analysis input, execution and output; XTEAM [5] and KAMI [3] focus on providing facilities for input adaptation and tool execution; our ongoing work focus on the integration on the synthesis of analysis results. However, these works assume that the analysis methods can be integrated independently and used together to finish an analysis process. The relationships between analysis methods during an analysis process are not considered.

In this paper, we propose a process based method to model and validate analysis process. We propose a domain-specific language for modeling the analysis process based on notations from BPMN. The modeling language considers the sequence relationship and dependency relationship between analysis methods in the process. In order to cope with the dependency problem in model and implementation level, the dependency relationships are divided into static dependency and dynamic dependency. Based on the analysis process model, dependency identification and matching are provided to assure the correctness of analysis execution. To ensure the static dependency could be guaranteed, the require input of each analysis method is evaluated. For the dynamic dependency, we provide a service to detect the analysis failure caused by dynamic dependency which is expressed by the elements with runtime data dependency definition, and this detection service is added to the process.

The remainder of the paper is organized as follows: Section 2 illustrates the modeling and validation requirement of analysis process. Section 3 and 4 introduce the modeling and validation method for analysis process. Going back to our example, Section 5 presents the use of our method to model and validate analysis process for software adaptation. Finally Section 6 discusses some related works before Section 7 concludes our work.

II. ILLUSTRATIVE EXAMPLE

In this section, we will introduce an analysis process including three analysis methods to show the modeling and validation requirement.

This example is a software adaptation analysis process, which first appeared in the work on performance adaptation [9]. The process includes three analysis methods for monitoring, analyzing and planning respectively: behavior model generation method COMPAS (Component Performance Assurance Solutions) [2], performance related pattern detection method [9], pattern-based reconfiguration method [9]. The user first uses the COMPAS analysis method to generate the behavior model based on runtime information contained in a software model from the monitoring stage. Then, the user uses an analysis method to detect Fine-grained Remote Calls structure in the software model. If instance of the bad structure pattern is found, the user will execute pattern based reconfiguration method to generate an adaptation plan. If bad structure does not exist in the software model, the analysis process is finished without planning.

In the example, the pattern detection method relies on the behavior model generated from the COMPAS method. Whether the planning method will be executed is decided by the result of pattern based detection method. The data dependency and control dependency may cause problem in design time and runtime. At design time, if the user first chooses the pattern based detection method instead of the COMPAS method, the detection method could not be executed without behavior model. We add the COMPAS method in the analysis process to solve the static dependency problem. However, dynamic dependencies may happen. For example, the COMPAS analyzer will not generate any result when it encounters runtime error. In this situation, all the analysis methods and control switchers rely on it will be affected. The analysis process will stop executing, or generates fault results or useless results.

Besides modeling the analysis process, we need to validate the process in order to correctly analyze the system.

III. MODELING ANALYSIS PROCESS

Our approach starts from the modeling of analysis process. The activities in the process are all of the type analysis method. An analysis process can be viewed as a group of analysis methods.

In order to simplify the analysis process modeling, we build our modeling method based on existing works on the integration of analysis methods. The integration of the analysis methods can be divided into three aspects: the integration of the input, the integration of the execution and the integration of the output. We assume that the analysis methods are already integrated that the input is provided by the software model through input adaptation; its tool is integrated and can be invoked; the analysis results are synthesized back into the software model.

Data processing and control logic are two important consideration in the molding of analysis process.

From the data level, all the analysis methods in an analysis process have the same analysis object: the software system. The analyzed system is usually abstracted and saved as model. The analysis results are the features of software system and can be modeled as elements or attributes of elements in the software model. As a result, the data flow in the analysis process includes the core software model and the analysis results. The analysis methods are organized as a star configuration.

In order to simplify the data processing modeling, we use our former work [11] on the integration of analysis results to update the core software model. Based on the framework, the software model is updating during the analysis execution. After an analysis method is executed, the software model will be extended with its results. For each analysis method, its required input can be provided by the software model and the analysis results produced of the analysis method executed
before. As a result, the analysis result integration framework maintains the models and manages the data processing in the analysis process. Based on the integration framework, modelers do not need to model the data processing in the analysis process; they can concentrate on modeling the control logic of analysis process.

The complex control logic of a process can be described using three kinds of structure: sequence structure, decision structure and repetition structure. However, an analysis method is seldom repeatedly executed during an analysis process. Because analysis is a way to generate software features from the software model, the analysis behavior does not affect the model. For a specific analysis method and a software model, the analysis result is the same if the analysis method is executed repeatedly. Thus, repetition structure is not needed in analysis process. It should be noted that an analysis method may be carried out more than once during an analysis process. This is because an analysis method can be applied to analyze different part of software model especially analysis results generated by other analysis methods.

Besides the sequence structure and decision structure, parallel structure is supported in the analysis process modeling for performance consideration. Because analysis methods are usually provided and implemented by different vendors, their analyzers may be running in different nodes. If some of the analysis tasks are time consuming and can be carried out by the analyzers in different node, the designer can model these analysis methods in a parallel structure.

We use a subset of notation of BPMN to model the analysis process. The modeling element types are listed in table 1. The start event and end event are used in the beginning and ending of the analysis process. Inter Event is only used to show possible exception in the process. A Task is mapped to an integrated analysis method which is the only kind of atomic activity in the analysis process. AND Gateway is used to describe the set of analysis methods which should be carried out in parallel. XOR Gateway and Sequence Flow are used to describe the sequence structure and decision structure.

<table>
<thead>
<tr>
<th>Table 1 modeling elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>Start Event</td>
</tr>
<tr>
<td>Inter Event</td>
</tr>
<tr>
<td>End Event</td>
</tr>
<tr>
<td>Task</td>
</tr>
<tr>
<td>AND Gateway</td>
</tr>
<tr>
<td>XOR Gateway</td>
</tr>
<tr>
<td>Sub-process</td>
</tr>
</tbody>
</table>

| Sequence Flow             | The Sequence Flow shows the order in which activities will be performed in a process. |

Our analysis process modeling language uses the notation of BPMN. Designer can use the BPMN visualization modeling tool and implementation generation tool to model the analysis process and get the executable process described using BPEL regarding semantic mapping between BPMN and BPEL.

IV. VALIDATING ANALYSIS PROCESS

There are two kinds of dependency: data dependency and control dependency. To ensure the static dependency could be guaranteed, the dependency occurs during the analysis process is detected and the depended analysis method is automatically added to the process. For the dynamic dependency, we provide a service to detect the analysis failure caused by dynamic dependency which is expressed by the elements with runtime data dependency definition, and this detection service is added to the process afterwards.

A. Dependency Definition

In order to validate the dependency relationship, we extend the integration definition in the analysis result integration framework. Fig 1 shows the extended meta-model, the integration model is extended with the element Relied Element. Analysis method integrator can specify the relied elements and their validation rules. The default validation rule for a relied element is that it should not be null. For the Gateway element in the analysis process, the switcher usually relies on some elements for decision making. The process modeler can define an integration model for a Gateway element. The integration model only includes a list of relied elements. We detect static dependency problem and validate the required input model at runtime based on the integration models of analysis methods and Gateway elements in the process.

B. Static Dependency Detection

There are two kinds of dependency relationship between analysis methods: data dependency and control dependency. An analysis process is incorrect if there is any analysis method or control switcher who relies on one or more analysis methods which could not generate needed analysis.
result for this method or control switcher. The dependency problem may occur when (1) the relied analysis method is not included in the analysis process; (2) the relied analysis method is included in the analysis process, but could be executed after the analysis method or control switcher in some situation; (3) the relied analysis method is included in the analysis process and could be executed before the analysis method or control switcher in any situation, but the analysis method has dependency problem.

For an analysis method \( m_0 \) in the process model, it does not have dependency problem if:

1. All the analysis methods which \( m_0 \) relies on are included in the process model.

2. All the process instances will carry out \( m_0 \)'s dependent analysis methods before \( m_0 \).

Because analysis methods do not affect each other directly even if they are in the same analysis process. Their dependency relationships are built based on the relationship between their input and output. \( AR_{\text{m0,dependent}} \) is the set of analysis results which analysis method \( m_0 \) relies on; \( AM_{\text{m0,dependent}} \) is the set of analysis methods whose results can be used by analysis method \( m_0 \); \( AR_{\text{m0,dependent}} \) is the set of analysis results produced by all the analysis methods in \( AM_{\text{m0,dependent}} \). An analysis process do not have static dependency problem when all the analysis methods in the process rely on the analysis results which can be generated before its execution in any process instances. The constraint can be expressed in OCL as below:

\[
P\text{.task} \rightarrow \forall \{ m_0 | (AR_{\text{m0,dependent}} \cup \text{CoreModel} ) \}
\]

\[
\supseteq AR_{\text{m0,dependent}} \subseteq P\text{.task}
\]

For an analysis method \( m_0 \) in the analysis process, its dependent analysis results are defined by the integrator. We developed an algorithm to find out the dependable analysis methods, the union of all the results produced by the dependable analysis methods is \( AR_{\text{m0,dependent}} \).

Initialization: \( AM_{\text{dependent}} = \phi \), \( \text{checkpoint} = \text{point}_{\text{start}} \), \( SF_{\text{temp}} = \text{checkpoint.outgoing} \);

1. if \( \text{checkpoint} = \text{point}_{\text{start}} \) and \( AM_{\text{temp}}\text{-size} = 1 \), then \( \text{checkpoint} = AM_{\text{temp}} \rightarrow \text{first}() \)
   \( AM_{\text{temp}} = \text{checkpoint.outgoing} \)
2. if \( \text{checkpoint} = \text{point}_{\text{end}} \), goto (9)
3. if \( \text{checkpoint} = m_0 \), goto (8)
4. if \( \text{checkpoint} \) is of the type Task and \( AM_{\text{temp}}\text{-size} = 1 \), then
   \( AM_{\text{dependent}} = AM_{\text{dependent}} \cup \{ \text{checkpoint} \} \),
   \( \text{checkpoint} = AM_{\text{temp}} \rightarrow \text{first}() \),
   \( SF_{\text{temp}} = \text{checkpoint.outgoing} \)
5. if \( \text{checkpoint} \) is of the type Gateway, then
   for \( m \) in \( AM_{\text{temp}} \)
   \( AM_{\text{dep_temp}} = \text{FindAMDepedent}(P, m_0, m, \text{checkpoint.jointpoint}) \)
   If \( AM_{\text{dep_temp}} \neq \phi \), then
   \( AM_{\text{dependent}} = AM_{\text{dependent}} \cup AM_{\text{dep_temp}} \)
   goto (9);
   endfor
   \( \text{checkpoint} = \text{checkpoint.jointpoint} \)
   if \( \text{checkpoint} \) is of the type AND Gateway and \( AM_{\text{temp}} = \phi \), then
   goto(6)
6. \( AM_{\text{dep_temp}} = \phi \)
   for \( m \) in \( AM_{\text{temp}} \)
   if \( m \) is of the type XOR Gateway, then
   \( AM_{\text{dep_temp}} = AM_{\text{dep_temp}} \cup m\text{.jointpoint} \)
   if \( m \neq \text{checkpoint.jointpoint} \) and \( m \) is of the type Task
   \( AM_{\text{dep_temp}} = AM_{\text{dep_temp}} \cup m\text{.outgoing} \)
   \( AM_{\text{dependent}} = AM_{\text{dependent}} \cup \{ m \} \)
   endfor
   if \( AM_{\text{dep_temp}} \neq \phi \), then
   \( AM_{\text{temp}} = AM_{\text{dep_temp}} \)
   goto (6)
7. goto (1)
8. return \( AM_{\text{dependent}} \)
9. return \( \phi \)
Using this algorithm, a process can be validated. If for each analysis method in the process, its relied input can be provided by the core model and the analysis results of its dependable analysis methods, this analysis process do not have static dependency problem.

C. Runtime Validation of Required Input Model

During the execution of an analysis process, runtime exception may affect the analysis execution and produce meaningless analysis results. The runtime exception may be caused by missing input from the core software model, exceptions from analysis tool execution, and so on. The fault or meaningless analysis results may affect other analysis methods which rely on these results. Thus, if this kind of fault can be detected at runtime, the process can be stopped so as to save time and avoid the misleading analysis results.

The runtime validation is the process to make sure that the input constraints are not violated. The validation is based on the dependency definition. The validation objects are the elements specified as relied elements in the integration model. There is a default rule that the required input element should not be null. This rule can be described using OCL:

$$\text{oclIsUndefined}() = \text{false}$$

Besides this rule, analysis method integrator can define rules as constraints for the \textit{reliedElement}.

The runtime validation of required input model can be abstracted as a special analysis method in the analysis process. Its analysis result is used to decide whether the process will continue or not. For each analysis methods or switchers with relied elements, we add a runtime validation method before it. We develop and integrate the validate tool in the integration framework. Then, the analysis process can detect validation of constraints at runtime.

V. CASE STUDY

In this chapter, we use our approach to model and validate the analysis process for the adaptation of a component-based system, the ECPerf system [6]. The ECPerf system is provided by Sun Microsystems, which has been adopted as a standard benchmark application for JEE systems. It is an online “Just-in-Time” manufacturing system. We used the SM@RT [10] monitoring platform to extract data from the running ECPerf system and to visualize the corresponding software model. We used the analysis method integration framework [11] to integrate the related analysis methods. For space limit, we focus on the modeling and validation of analysis process, detail of analysis methods integration is not included in this paper.

A. Performance Adaptation

In section 2, we introduce an analysis process for performance adaptation. For a maintainer who is not familiar with the analysis methods, he does not know that the pattern detection method relies on the behavior model which is not provided by the monitoring system. He may model an analysis process containing two steps: detect performance related bad pattern, and reconfigure the software if bad structure exists, as shown in Fig 2.

Fig 2 Analysis Process for Performance Adaptation

We use our static dependency detection algorithm to validate this analysis process. Because pattern detection method is the first method in the process, it can only rely on the core software model. In the integration definition of pattern detection method, two relied elements are defined: software configuration element \textit{Config} and the behavior model element \textit{SequenceDialog}. The configuration model is included in the software model. However, the software model only contains a list of invocation records. The software model does not contain any element of the type \textit{SequenceDialog}. There is static dependency problem in the process. If an analysis method COMPAS is added before the pattern detection method, as shown in Fig 3, it can generate behavior model based on the invocation record in the core model. This analysis process does not have dependency problem and can be used for performance adaptation.

Fig 3 Analysis Process for Performance Adaptation

B. Fault-tolerant Style based Software Reconfiguration

Reliability is one of the most important quality attributes for online services. Because reliability is important, the maintainer chooses two reliability analysis methods SBAR [12] and ABRAM [14] to analyze the runtime software model. If the value of any reliability result is less than 90%, the maintainer will choose a planning method ASPIRE [13] to generate adaptation plan, and then use the SBAR and ABRAM method again to analyze the planning result to make sure that the reliability will be improved. During the analysis process, the maintainer may stop analyzing if the system reliability is acceptable. We use our modeling language to describe this analysis process, as shown in Fig 4.

Fig 4 Analysis Process of Style-based Software Adaptation

In this example, the SBAR method and ABRAM method is used twice with different input. The XOR Gateway before
ASPIRE method relies on the reliability analysis results: SBARresult and ABRAMresult. The constraints for these two elements are the same that the value of the element should be in the range of 0 to 1. A validation analyzer is added before the gateway in the analysis process. This analyzer will check both the SBARresult and ABRAMresult when the SBAR and ABRAM method finishes execution.

When we apply this analysis process in analyzing the software model of ECPerf system, the process always stopped and through an exception before the XOR gateway. Because we re-implement ABRAM analyzer using the model transformation language ATL with too many nested statements, the analyzer will be short of memory when analyzing software models with more than 10 components. When analyzing the ECPerf software model, the ABRAM analyzer cannot return the result ABRAMresult until timeout. As a result, the runtime validation carried out before the XOR gateway will find that the required input ABRAMresult violates its constraint. The runtime validation is also helpful for us to detect improper usage of analysis method.

VI. RELATED WORK

The works on the Enterprise Application Integration (EAI) can be applied to integrate analysis tools. However, EAI most focuses on the translation the outputs of some application into inputs for other application, and the invocation of application interfaces [15]. The relationships between integrated analysis methods are not considered.

Many approaches and frameworks have been developed for complex analysis of software models. Some approaches are proposed to facilitate the use of multiple analysis methods during software design. ADD [7] and FADF [6] provide standard interfaces for integration of analysis methods. These works notice the requirement for the analysis process, but do not provide support for process modeling. In addition, these works simplify the relationship between analysis methods as data exchanging and leave the user to guarantee the process correctness.

Several research projects propose frameworks [3, 4, 5, 6] to simplify the integration of existing model-based analysis methods. XTEAM [5] is a framework focusing on solving the mismatch between SA model and required input of analysis methods. These frameworks provide facilities to simplify the development of model interpreters and model updaters, respectively. DUALLY [4] is an automated framework that allows architectural languages and tools interoperability. It provides the infrastructure for (semi) automatic generation of the weaving model to integrate analysis abilities provided by ADLs. These frameworks assume that analysis methods are independent and do not consider dependencies of the analysis methods. Our framework focuses on the modeling and validation of analysis process, and is build based on some of these works.

VII. CONCLUSION

This paper introduces an approach to model and validate the analysis process for software adaptation. We believe that an explicit model for analysis process can enhance the reuse of analysis process and save effect for analysis execution. Considering the dependency relationship in analysis methods, our method provide an algorithm to detect static dependency problem in the analysis process model and an analysis method to validate required input model at runtime, so as to avoid correctness problem caused by dependency.

VIII. ACKNOWLEDGMENTS

This effort is sponsored by the Joint Funds of NSFC-Guangdong under Grant No. U0835004, the National Natural Science Foundation of China under Grant No. 60873060, 60933003, and 61003072.

REFERENCES