Inspection of SOFL Specifications through Building Traceability

Jinghua Zhang and Shaoying Liu
Graduate School of Computer and Information Sciences, Hosei University
jinghua.zhang.6w@stu.hosei.ac.jp, sliu@hosei.ac.jp

Abstract

When developing a formal specification for a software project using the SOFL three-step modeling approach, it is essential to ensure the conformance relation between every two level specifications. In this paper, we describe an inspection method through building traceability for rigorously verifying the conformance relation. The method consists of two steps: (1) traceability establishment and (2) inspection of the target specifications through building traceability. A small case study is given to show how the proposed method can be applied in practice.

1. Introduction

One of the primary problems in software projects is that the requirements documented in specifications may not be accurately and easily understood by the developers carrying out different tasks [1]. One way to improve the quality of specifications and therefore the quality of the corresponding software is to formalize specifications. We choose Structured Object-Oriented Formal Language (SOFL) as a formal notation in this paper.

The SOFL method provides a three-step approach to developing formal specifications. Such a development is an evolutionary process, starting from an informal specification, to a semi-formal one, to finally a formal specification [1]. However, as a formal method, there are still many places that can be improved in practice, especially about the purification through three level specifications. Our research mainly focuses on how to sustain the consistency between different level specifications.

In this paper, we present an inspection method through building traceability that helps users check the consistency in their specifications.

2. Inspection through building traceability

There are two steps in our inspection method through building traceability. Firstly, we generate the traceability between three different specifications. The traceability means the congruent relationships of elements which represent the same users' requirements in different specifications. For example, a function in the informal specification may be correlated to a process in the corresponding semi-formal specification. Secondly, by measuring the effectiveness of traceability, we inspect corresponding elements in different specifications together.

2.1. Building traceability between different specifications

Because there are three specifications, we put the traceability into two parts to make it more clearly: (1) traceability between informal and semi-formal specifications, (2) traceability between semi-formal and formal specifications.

During the first part, user's requirements will be purified and described more precisely. For covering user's requirements as many as possible, the structures are rough. They contain only three components: functions, data resources and constraints. Because of the partition in informal specification, the conversion to semi-formal specification is quite flexible and mainly depends on user's experience. However, we can still compare corresponding elements based on structures in different specifications as shown in Table 1 below.

Table 1.Connected elements in different specifications

Informal specification	*Semi-formal specification*
Function	Function, Process, Module
Data resource	Type identifier, Constant identifier, Variable
Constraint	Invariant

For the second part, structures are almost the same between semi-formal and formal specifications. We can generate traceability based on building *functional*

scenarios provided in [2]. Let $P(P_{iv}, P_{ov})[P_{pre}, P_{post}]$ denote the formal specification of an operation P, where P_{iv} and P_{ov} are the sets of all input and output variables. P_{pre} and P_{post} are the pre-condition and post-condition of operation P, respectively. Let $P_{post}=C_1 \wedge D_1 \vee C_2 \wedge D_2 \vee \cdots \vee C_n \wedge D_n$, where C_i ($i \in \{1, 2, ..., n\}$) is a guard condition and D_i is a defining condition. Then, a conjunction $\sim P_{pre} \wedge C_i \wedge D_i$ is called a functional scenario.

Because the corresponding operations in semi-formal and formal specifications describe same requirements, functional scenarios should be same. Then, we can generate traceability by put same functional scenarios together.

After generating two parts of traceability, we can inspect elements through the whole one.

2.2. Inspection through building traceability

When inspecting corresponding elements in different level specifications, we can put them together based on traceability. At the same time, measuring the effectiveness of traceability is a feasible way to check the traceability is good enough or not. This measuring focuses on different changes about corresponding elements in the traceability. To the specification base, a lack of elements in compared specification will get a negative value (-1) while an unexpected one which is not included in the base will get a positive value (+1), respectively. By calculating the percent of negative and positive values, we can measure the effectiveness of traceability and check possible errors easily.

2.3. A case study

Here, we use a case study to show how our inspection method through building traceability works. The case study is SOFL different level specifications used for describing requirements of the JTB system.

The JTB system mainly includes four functions: (1) making the tour plan, (2) reserving flights, (3) making bus arrangement, (4) reserving hotel. Firstly, we can generate traceability between different specifications. For example, when we try to find the traceability of the function called "Reserve for Flight" in informal specification, we can get the corresponding elements in semi-formal specification as shown in Figure 1.

After generating the traceability, we can put corresponding elements in three different specifications together to inspect them. According to the number of corresponding elements, we can measure the effectiveness of traceability shown in table 2.

```
FlightPlan = composed of
             date : Date
                    start_time : Time
                    arrival_time : Time
                    start_city : string
                    start_airport : string
                    arrival _city : string
                    arrival_airport : string
                    flight_no : given
                    price : nat
                    type : given
             status : {<NoVacancy>, <Available>}
           end;
process ReserveForFlight(reserve_for_flight_request : ReserveForFlightRequest)
          flight_reservation_result : FlightPlan, flight_confirmation_signal : bool
ext rd FlightPlanDB
     rd CustomerDB
     wr FlightContractDB
pre true
decom ReserveForFlight_Decom
end_process
```

Figure 1. "Reserve for Flight" in semi-formal specification

Table 2.Effectiveness of traceability

	Elements	Positive	Negative
Informal	19	--	--
Semi-formal	32	3	2
Formal	35	0	1

In the informal specification, we write 19 elements to describe user's requirements as the benchmark. Based on this, we purify 32 elements in the semi-formal specification. As measuring the effectiveness of traceability, we get 3 positive values and 2 negative values. That means when writing semi-formal specifications, we add three elements which can't be connected in informal specification. These may be errors or new requirements that should be added in the informal specification. For example, after checking this, we find the lack of "Confirm Tour Contract" in informal specification. At the same time, the negative values show that two corresponding elements are lost in the semi-formal specification. By using the measuring method, we can orientate possible errors quickly and help users inspect elements in different level specifications precisely.

References

[1] Shaoying Liu: Formal Engineering for Industrial Software Development Using the SOFL Method, Springer-Verlag, 2004

[2] Mo Li, Shaoying Liu: Automated Functional Scenario-based Formal Specification Animation, Proceedings of the 19th Asia-Pacific Software Engineering Conference (APSEC 2012), IEEE CS press, pages 107-115, 2012