An Overview of Architecture-based SPL Test Generation

Jihyun Lee

1 Department of Software Engineering, Chonbuk National University, Jeonju, 54896, Republic of Korea
jihyun30@jbnu.ac.kr

Abstract. Software architecture can be used as an important input for generating test artifacts, including test plans and test cases. Particularly, in the case of testing codes related to connections and interaction mechanisms defined in architecture, architecture based testing makes easy test automation. As software architecture provides benefits such as buildability, reusability and evolvability for software development, we can expect that architecture based testing has similar benefits on software testing. This paper conceptually introduces the architecture-based SPL test generation approach. Architecture-based SPL test generation approach generates SPL test cases by using SPL test architecture defined from architecture-level SPL use case scenarios and SPL architecture.

Keywords: software product line, SPL testing, architecture based testing, test architecture

1 Introduction

Software architecture can be used as an important input for generating test artifacts, including test plans and test cases. Particularly, in the case of testing codes related to connections and interaction mechanisms defined in architecture, architecture based testing makes easy test automation. Accordingly, we need to rethink about the existing software testing approaches on the basis of software architecture[1]. As software architecture provides benefits such as buildability, reusability and evolvability for software development, we can expect that architecture based testing has similar benefits on software testing. The benefits of architecture based testing in error detection and error locating have been already validated[2]. Therefore, we can naturally expect that architecture based testing can provide the same benefits for software product line (SPL) testing.

This paper conceptually introduces the architecture-based SPL test generation approach. The complexity of SPL testing is high because of undetermined, i.e. unselected variability[3]. Product line architecture[4] incudes the detailed descriptions for variable components, variable connections and variable interaction mechanisms. Architecture-based SPL test generation approach[5] generates SPL test cases by using SPL test architecture defined from architecture-level PL use case scenarios and PL
architecture. SPL use case scenarios are refined scenarios at architecture level, including interactions among components. SPL test architecture is a refined architecture from SPL architecture by adding testing elements such as testing relevant decisions, point of observation/point of control and observation (PO/PCO) and relations between PO/PCO and components. Moreover, variability can be newly added at testing stage for dealing with different test requirements and test coverage decisions. SPL test case scenario can be derived by using SPL test architecture and SPL use case scenario. Architecture-based SPL test generation approach derives test case scenarios for a member product from variability binding results at the relevant application engineering stage and at testing stage.

This paper is organized as follows: Section 2 describes our approach with motivating example and in Section 3, we make concluding remarks.

2 Architecture-based SPL Test Generation Approach

Fig. 1 is a diagram that describes the activities of our architecture-based SPL test generation approach together with inputs and outputs. In this section, we explain the details of each activity with an example.

Fig. 1. Architecture-based SPL test generation approach

This section explains our approach through the pre-defined BankAccount PL for experiments and enhancing understandability of our approach. BankAccount PL consists of two components, BankProgram and vpAccount. vpAccount can be either checkingAccount or savingAccount.

2.1 A1: Derive SPL Test Architecture

A1 adds POs/PCOs to the relevant locations of SPL architecture in accordance with test coverage decision. The major inputs of A1 are SPL architecture and test coverage decision as shown in Fig. 1. The major outputs are SPL test architecture and variability model in testing. SPL test architecture includes binding information at testing stage. SPL test architecture includes newly added variability at testing stage besides variability due to those defined in test targets, i.e. domain artifacts. For
example, a number of POs/PCOs and their locations can be defined as variability at testing stage.

Fig. 2 is an example test architecture defined from the BankAccountSystem PL architecture. Test coverage decision adds POs or PCOs to ‘all interactions changing values of variables’. In the case of observing an intermediate value, we add a PO. Whereas, in the case that inputs from other components or changed variable have critical impacts on the behavior of a test target, we add a PCO to the interaction point. Decisions about either adding a PO or PCO can be causes of variability.

![Test architecture defined from the BankAccountSystem PL architecture](image)

Fig. 2. Test architecture defined from the BankAccountSystem PL architecture

Test architecture for the BankAccountSystem PL can include variable test elements like named vpPNT in Fig. 2 because vpAccount is a variable component. In addition, variability at test stage for selecting a PO or PCO can be added to the test architecture.

As shown in Fig. 2, a PO or PCO can be added to an interaction point between BankProgram and vpAccount for observing or controlling/observing interactions between the two. vpPNT is a variability introduced because of vpAccount and a variable decision about PO or PCO. Binding of vpPNT differs from decisions about vpAccount. Namely, the possible PO or PCO in vpPNT also differs from decisions about vpAccount.

2.2 A2: Apply SPL use case scenarios to SPL test architecture

A2 generates architecture-based PL test case scenarios by applying SPL use case scenarios[6] at architecture level to SPL test architecture, that is an output of A1. The major inputs of this activity are SPL test architecture and SPL use case scenarios, and the major output is SPL test case scenarios. SPL use case scenario includes both commonality and variability, and its variable parts have links with variability model.

Test architecture-based PL test case scenario is obtained as like scenario in Fig. 3. vSD1 is the case that checkingAccount variant is bound to vpAccount and vSD2 is the case that savingsAccount variant is bound respectively. The below part of Fig. 5
shows test case scenario replaced vpPNT as vpPNT_CA and vpPNT_SA. vpPNT_CA and vpPNT_SA are still variables because decisions about POs/PCOs are not made yet.

![Diagram](image)

**Fig. 3.** Example test architecture-based PL test case scenarios

In the case of checkingAccount, vPO_CA is a PO that observes whether the component sets the right value of deposit variable after credit operation while vPO_SA is a PO for observing the value of balance variable after deposit operation of savingsAccount. Each PO and PCO should implement operations for supporting these behaviors and additional implementation may be added to vpAccount, if necessary.

SPL test case scenario for credit operation can be defined as following:

\[
\text{Credit(amount), } \{(\text{setLastDeposit(amount)}, (\{ }_{\text{vpPNT}}{\text{CA}_1}, \{ }_{\text{vpPNT}}{\text{SA}_2}\})_{\text{vpAccount}}
\]

\[
\text{where, } (\{ }_{\text{vpPNT}}{\text{CA}_1} = ((\text{getBalance})_{\text{vPO}}{\text{CA}_1}, (\lambda _{\text{PCO}}{\text{CA}_1})_{\text{vpPNT}}{\text{CA}_1}
\]

\[
(\{ }_{\text{vpPNT}}{\text{SA}_2} = ((\text{getBalance})_{\text{vPO}}{\text{SA}_2}, (\lambda _{\text{PCO}}{\text{SA}_2})_{\text{vpPNT}}{\text{SA}_2}
\]

### 2.3 A3: Apply test case (scenario) bindings to PL test case scenarios

A3 derives test scenario for a member product from SPL test scenario through binding. The major inputs of this activity are bindings occurred during application engineering, bindings decided at testing stage and PL test case scenarios. By using these inputs A3 derives test scenario for a member product. First of all, tester and artifacts related to PO/PCO are automatically derived according to variability binding results. Product test case scenario is also automatically derived in accordance with the binding results. Trace links between variability model and artifacts are orthogonally established, so we do not need to maintain which SPL test case scenario is related to which products[7]. A test case scenario for a member product that selects checkingAccount and vPO_CA is as follows:

\[
\text{Credit(amount), setLastDeposit(amount), getLastDeposit()}
\]

Copyright © 2016 SERSC
3 Conclusion and Future Works

This paper proposed an architecture-based SPL test case generation approach that supports efficient and effective test case generation. The contributions of the paper are as follows: It introduced architecture-based SPL test generation approach with inputs, activity and outputs. To show how the approach works, we showed the approach with motivating example; we showed how to uniformly derive product line test cases from the defined test architecture and finally how to derive test cases to be executed for a member product. However, SPL use case scenarios, currently used to describe the behavior of SPL architecture make it difficult to automate test case generation. Test data that also can be variable and should be properly treated during domain engineering. As future works, we tackle the two issues, test automation and variability in test data.

Acknowledgment. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2013R1A1A3005162).

References