DSLを利用したソフトウェア試験の効率化手法

甲斐　啓文†　　安井　照昌†
三菱電機株式会社　先端技術総合研究所‡

1　はじめに

現在のソフトウェア開発の分野において、RUP（Rational Unified Process）[1]やXP（eXtreme Programming）[2]といった、繰り返し型開発プロセスが注目されている。その背景として、変化の激しい現在のビジネス環境において、仕様変更への柔軟な対処やバグの早期発見によるリスクの削減を実現したいという要求が高まっていることが挙げられる。

繰り返し型開発プロセスでは、回帰試験の継続的な実施が要求される。一度完成したソフトウェアに変更を加えた場合、追加した機能に対して試験を実施するだけでは品質確保の観点で不十分であり、既存の機能に対して予想外の影響が現われていないことを同時に確認する必要がある。この際に実施する試験が回帰試験である。回帰試験を手動で実施すると大変なコストがかかるため、一般には自動化される。しかしながら、多くのソフトウェアでは、バージョンアップやバグフィックスが頻繁に繰り返され、その度に試験の規模が増大し、自動回帰試験の維持・管理が困難になる傾向にある。

本稿では、ソフトウェア試験の試験シナリオの記述に特化した専用言語（Domain Specific Language, DSL）を定義し、それを利用することにより、大規模な回帰試験の維持・管理を効率化するための手法を提案する。

2　本研究のアプローチ

我々は、試験シナリオを効率的に記述するためのDSLを利用することで、自動回帰試験を効率化する手法を考案した。本研究では、以下に示すアプローチを採用した。

1) 試験シナリオの形式化

試験シナリオを機械可読とするための形式化を行う。

2) DSLによる試験シナリオの記述

1)で形式化された試験シナリオを効率的に記述するためのDSLを定義する。

3) DSLで記述した試験の実行

DSLで記述された試験シナリオ（試験スクリプト）を読み込むことにより試験を自動実行する。

上記アプローチの詳細を以降で述べる。

Effective Method for Software Test Using DSL
†Hirofumi Kai　†Terumasa Yasui
‡Advanced Technology R&D Center, Mitsubishi Electric Corporation

3　試験シナリオの形式化

本研究では、ソフトウェアの試験を1) 事前条件を整える、2) 対象を操作する、3) 事後条件を検証するの3つの動作からなるものと定義する。

例えば、データベースにアイテムを挿入する操作について、表1に示すような事前条件、操作、事後条件が成り立つものとする。

表1: 事前条件、操作、事後条件の例

事前条件	データベースにアイテムが3件入っている
操作	データベースにアイテムを2件追加する
事後条件	データベースにアイテムが5件入っている

この場合、表2に示す試験シナリオを定義できる。

表2: 試験シナリオの例

Step	実施内容
1	データベースにアイテムが3件入っている状態を作り出す
2	データベースにアイテムを2件追加する
3	データベースにアイテムが5件入っていることを確認する

表2のStep1が事前条件の整備であり、Step2が対象への操作、Step3が事後条件の検証である。ただし、Step1については実施手順が不明確であるため、自動試験を構成することができない。本方式では、事前条件を他シナリオの実施によって成立させるものとする。表2に対し事前条件を成立させるための試験シナリオを追加したものを表3に示す。

表3: 試験シナリオの例

Step	実施内容	種別
1	データベースを起動する	操作
2	データベースが空であることを確認する	検証
3	データベースにアイテムを3件追加する	操作
4	データベースにアイテムが3件入っていることを確認する	検証
5	データベースにアイテムを2件追加する	操作
6	データベースにアイテムが5件入っていることを確認する	検証

つまり、表1を試験するための試験シナリオを、表3に示すように、対象への「操作」と事後条件の「検証」の連続パタンとして定義する。

4 DSL による試験シナリオの記述

我々は、「操作」と「検証」の連続パタンからなる試験シナリオを効率的に記述するための DSL を考案した。本 DSL の BNF による定義をリスト 1 に示す。

リスト 1: 本 DSL の BNF による定義

```
<script>::={<project>|<def>}*
<project>::="project""\n"<test>*"end""\n"
<test>::="\t""test"<name>"\n"
<def>::="def""\t"<name>"\n"<defbody>*"end"
<defbody>::="\t"{<name>|<op>|<as>}"\n"
<op>::="opq"<opscript>
<as>::="asq"<asscript>
```

script 本 DSL により記述される試験スクリプト全体。
project 複数の試験を一纏まりとして扱う。
test 実行対象の試験シナリオを指定する。
def 試験シナリオを定義する。
op 試験対象への操作を記述する。
as 事後条件の検証を記述する。
opscript 対象操作言語。試験対象により異なる。
asscript 対象検証言語。試験対象により異なる。

本 DSL の特徴を以下に述べる。

1) 手続き定義の集合としての試験シナリオの記述

本 DSL は、手続き定義の集合として試験シナリオを記述する。リスト 2 はノートパッドに対する試験シナリオの手続き定義の例である。2 行目の「opq」で指定される文字列は「操作」、3 行目の「asq」で指定される文字列は「検証」を表す。この例では、この 2 行を予約語「def」によって手続き「ノートパッド起動」として定義している。さらに手続き「ノートパッド起動」は、11 行目～14 行目において定義される手続き「テキスト挿入」から呼び出されている。

このように、手続き定義の集合として試験シナリオを記述することで、既存の試験シナリオの再利用が可能となり、試験シナリオの作成効率が向上する。

2) 様々な試験対象への対応

試験対象への操作と検証の内容（方法）は、試験対象によって異なる。そこで、操作内容、検証内容としてそれぞれ「対象操作言語」、「対象検証言語」という異なる言語を指定可能とすることで、様々な試験対象に対応する。リスト 2 では、GUI アプリケーション用言語を指定している。2 行目の「opq」の後に指定されている「[start \windows\notepad.exe]」は対象操作言語である。ここで指定されるのは、対象アプリケーションへのイベント送信であり、この場合「アプリケーションの起動」イベントである。3 行目の「asq」の後に指定された「ps 3000」は、対象検証言語である。ここでは、対象アプリケーションの画面ダンプの取得と、あらかじめ取得しておいた正解画面との比較を指定している。このような仕組みを備えることで、本 DSL の枠組みを維持したまま、様々な試験対象に対して統一的に試験シナリオを記述することができる。

リスト 2: 試験シナリオ「テキスト挿入」の手続き定義

```
1       def ノートパッド起動
2               opq [start \windows\notepad.exe]
3               asq ps 3000
4       end
5
6       def helloと挿入
7               opq hello
8               asq ps 1000
9       end
10
11      def テキスト挿入
12              ノートパッド起動
13              helloと挿入
14      end
```

5 DSL で記述した試験の実行

我々は、GUI アプリケーション用に、本 DSL で記述された試験シナリオを自動実行する試験ツール（図 1）を開発した。試験シナリオは DSL パーサによって一連の対象操作言語と対象検証言語に展開される。その後、それぞれのインタープリターによって解釈され、操作（イベント送信）及び検証（画面ダンプの取得及び正解画面との比較）として実行される。

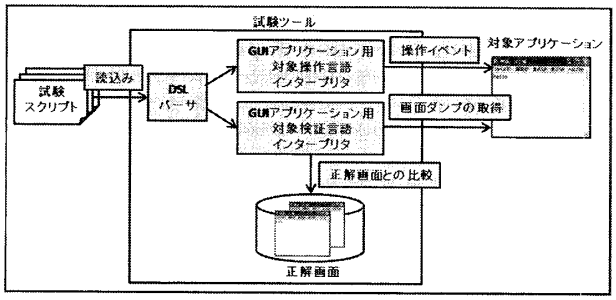

図 1: 試験ツールの概要

6 おわりに

今回考案した DSL は、試験シナリオの記述に特化した簡素な記述形式と、手続き定義と異なる言語の組み込みによる再利用性の向上によって、回帰試験の維持・管理の効率化を目指したものである。今後、本 DSL 内での変数の使用を認めるなど、改良により使い勝手の向上が期待できる。

参考文献

[1] Philippe Kruchten. *The Rational Unified Process: An Introduction.* Addison-Wesley, 2003.

[2] Kent Beck. *Extreme Programming Explained.* Addison-Wesley, 2004.