1S-5

デザインパターンに基づいたTraceability Linkの構築とSoftware Evolutionのサポート

佐々木宏太 表秀和 海谷治彦 海尻賢二
信州大学大学院工学系研究科

1. はじめに

ソフトウェア開発において、仕様変更、機能拡張の要求などによるsoftware evolutionは頻繁に発生し、そのサポートは必須である。適切なsoftware evolutionを行うためには、ソフトウェアの要求仕様、設計モデル、実装の間に適切なtraceability linkを確立することが重要である。

本研究では設計モデルの記述にUMLを使用したソフトウェアを対象とし、software evolutionのサポートを行う。設計に使用したUML図間でのtraceability linkと共に、デザインパターンに基づいたtraceability linkを構築し、より高度なサポートを提供する。

2. デザインパターンと要求仕様

デザインパターンは、ある設計問題とそれに対する解法の集合体であり、そのソフトウェアに求められた機能要求と、拡張性、保守性などの非機能要求を同時に満たすために用いられる。

従来のUML内でのtraceability link[1]では、機能要求と下位のプロダクトとの関係を辿ることはできるが、非機能要求と下位のプロダクトとの関係性が曖昧であり、その関係を辿ることはできない。デザインパターンに基づいたtraceability linkを使用することにより、非機能要求に関する情報を含めたtraceability link[2]を実現することが出来る。

3. デザインパターンとevolutionの分類

デザインパターンを用いたソフトウェアの場合、想定している変更要求に対して柔軟になり、容易に拡張を行うことが出来るようになる。しかし一方ではデザインパターンの利用により制約、前提条件が発生し、それを覆すような変更要求に対しては対応することが困難となり、このような変更要求を実現するためにはパターンの適用自体も解消する必要がある。

Evolutionをこのような観点から分類すると

- デザインパターンに適したもの
- デザインパターンとは関係ないもの
- デザインパターンに適さないもの

の３つに分類することが出来る。各パターンにおいて、このようなevolutionを識別することにより、上記分類に基づく変更方法の示唆が可能となる。

4. evolutionの種類の識別

各パターンはパターン固有のクラス構造を持ち、各クラスには役割名が与えられている。パターンにおける制約となる役割のクラスと、パターンの利点を生かすための役割のクラスが存在する。あるevolutionが発生した時に、デザインパターンに基づいたtraceability linkによって、変更がどの役割のクラスに当たるのかが分かれば、evolutionの種類を推測することが可能となる。

5. UML間でのtraceability link

本研究では、従来のUML間でのtraceability linkとデザインパターンに基づくtraceability linkを併用する。Evolutionの対象となった箇所がデザインパターンとは関係ない部分だった場合は従来のtraceability linkにより変更箇所の特定を行う。

6. デザインパターンに基づくtraceability linkの構築

本研究ではデザインパターンモデルを使用し、デザインパターンに基づくtraceability linkを実現する。現段階ではデザインパターンモデルの厳密な記法を定義していないが、デザインパターンモデルは、該当するパターンのパターン構造、含まれるクラス群の役割に関する情報、そのクラスもしくはクラス内の要素に対する変更が3種類の内のどのevolutionに当たるかの

Support for Software Evolution using Traceability Link based on Design Pattern
Kota SASAKI
Shinshu University, Graduate School, Engineering Course, Information Engineering

情報を保持している。

デザインパターンモデルと関連付ける対象として、ユースケース図内の各ユースケースに対応したアクティビティ図と、実装レベルに最も近いクラス図を使用する。

アクティビティ図はユースケース図よりも粒度が細かく、ユースケースシナリオの内容をほぼ忠実に再現できるため関連付けにおける最上位のモデルとして使用する。

アクティビティ図内のアクティビティとアクティビティ内の名詞をデザインパターンモデルと関連付ける。また実装レベルのクラス図については、クラス、クラス内の要素と関連付けを行う。デザインパターンモデルを介してアクティビティ図とクラス図を関連付けることにより、変更要求発生時に、アクティビティ図から、変更箇所、変更方法を特定する。

7. デザインパターンに基づく traceability link を使用した例

Traceability link を用いた実例として、リンク集作成システムの場合を取り上げる。

図 1 のように主に３つのユースケースからなるリンク集作成システムは、階層構造を持った URL のブックマークを作成し、それを HTML に出力することができる。

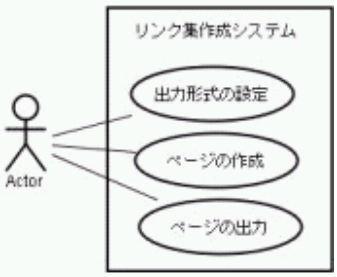

図 1

2 種類の出力形式に対応し、リンク形式とテーブル形式での出力を可能としている。このシステムは Abstract Factory パターンを使用し実装している。

このプログラムを作成する際に設計モデルとして、前述のユースケース図、さらにその各ユースケースに対応する 3 つのアクティビティ図、さらに実装レベルに最も近いクラス図の 5 つの UML 図が存在する。

このプログラムに対しての変更要求を想定しデザインパターンに基づいた traceability link が有効に働く事を実証する。

8. 変更要求：出力形式の追加

リンク集作成システムでは、リスト形式、テーブル形式での出力に対応している。この 2 つに、「数字リスト形式の HTML を出力する機能」を追加する要求を想定する。

図 2

図 2 は UML 図とデザインパターンモデルの関連付けの状況を示している。

「形式」という言葉をキーワードとして、「出力形式の設定」に関するアクティビティ図に注目する。アクティビティ「形式の選択」はデザインパターンモデルの AbstractFactory クラスと ConcreteFactory クラスと関連付けられている。さらにアクティビティ「リスト形式に設定」からは、ConcreteFactory クラス、「テーブル形式に設定」は同じく ConcreteFactory クラスに関連付けられている。デザインパターンモデルの情報から ConcreteFactory クラスへの変更は、デザインパターンに適した Evolution であり、この変更による他の部分への影響は最小限となる。変更内容は ConcreteFactory クラスとそのインスタンス作成部分の修正、それに付随するクラスの追加のみとなる。

9. 結論と今後の課題

デザインパターンに基づく traceability link を構築することにより、変更要求の対象となる変更箇所の特定と、evolution の分類を行うことが出来る。これらにより evolution 実施時のコストを軽減できる。

現段階では、デザインパターンモデルの定義、traceability link の記述方法の定義を行っていない。今後、これらに対する考察を行う。通常の設計に加え traceability link を構築することは開発者にとって負担となる。開発プロセスの中で、link を容易に構築できるようなサポートが必要である。また一度構築したリンクを維持するためのサポートも必要となる。

10. 参考文献

1 T. Tsumaki, et al, A Framework of Requirements Tracing using UML, APSEC 2000
2 Daniel Gross, et al, From Non-Functional Requirements to Design through Patterns, RE2000