5K-06

自動コード生成を目的としたテンプレートベースの UML 上位設計に対する整合性検査

A Consistency Check for Template-based UML Upper Level Design for Automated Code Generation

畑瀬尚之 Takayuki Hatase
和崎克己 Katsumi Wasaki

信州大学総合理工学研究科
Graduate School of Science and Technology, Shinshu University
信州大学工学部
Faculty of Engineering, Shinshu University

1 はじめに

ソフトウェア開発の早い段階でのプロトタイピングによる要求確認は，欠陥発覚の遅れに対して有用な手段である．本研究の最終的な目的は，上位設計群を利用したプロトタイピングの自動コード生成であり，その準備研究として VDMJWeb サービス [1] がある．これは上位設計群を用いることで，ユーザによる早期の妥当性確認を可能にするものであるが，用いる上位設計群の間での整合性確認は行われていない．整合性が取れていなければ，プロトタイプへの不整合の混入や，コード生成が困難となる事が考えられるため，上位設計群に対し整合性検査を行うことが必須である．本研究では，検査対象は，プロトタイピングの自動コード生成に必要となる，テンプレートベースで記述された UML クラス図，UML アクティビティ図を対象とし，クラス図とアクティビティ図間での要素の一致検査と，アクティビティ図上のロジック層の構造検査を行う．

2 テンプレートを用いた UML 図記述

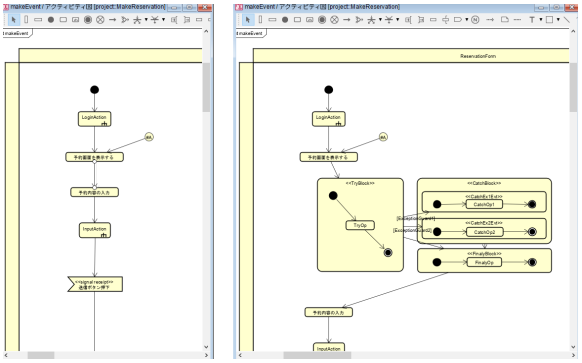
2.1 記述支援の必要性

提案手法による整合性検査は，検査を行う上での曖昧さの排除といった目的から，複数の UML 図記述規則をユーザに強いる．これにより，UML 図記述のコストの増大や不整合混入といった問題が考えられるため，テンプレートベースでの UML 図記述により，図入力形式の統一を図る．

2.2 各種制御構造のテンプレート挿入支援

システムの実行系列を記述する際，頻出構造と考えられる制御構文･例外処理をテンプレートとしてアクティビティ図上に作成し，それらをアクティビティ図内の任意のフローへの挿入を支援する機能を `astah*professional`[2] にプラグインとして実装した．

挿入時には，ノードの表示が重ならないよう，元のノードの位置を自動調整し挿入を行う．挿入可能な制御構造はワークフロー型となっており，挿入部であるフローの選択も必ず 1 つのみとしているため，制御構造の挿入によって元のアクティビティ図のフローとしての整合性が崩れることはない．図 1 に制御構造の挿入例を示す．図 1(a) の選択されたフローに対し，制御構造が挿入されたものが図 1(b) となっている．



(a) 挿入前　　(b) 挿入後

図 1　制御構造のテンプレート挿入例

2.3 デザインパターンを用いた UML 図記述

対象ドメインを定めたデザインパターンを作成することにより，ユーザがこれをベースとし，仕様に合わせた要素の追加や変更を行うことで，UML 図記述のコスト削減や不整合混入の抑制が可能となる．デザインパターンが記述された UML プロジェクト内に制御構文・例外処理の構造が記述されたファイルやログインシステムといった，汎用性があり広く用いられるシステムモデルを同時に提供することで，コスト削減・不整合混入の抑制を図る．

3 図に対する整合性検査

3.1 UML 図要素に対する一致検査

クラス図とアクティビティ図間で記述された要素が一致するかの検査を行う．このとき，クラス図を設計としてアクティビティ図よりも上位のものであるとし，クラス図基軸による検査を行う．また，検査に際し，記述規則を以下の通り設ける．

1. クラス図
   (a) 最低 1 つのクラスを持つこと
   (b) 変数の記述は VDM++でのインスタンス宣言と同様に記述すること
2. アクティビティ図
   (a) 最低 1 つのパーティション，開始・終了ノードを持つこと
   (b) パーティションにクラス名，アクションノードにメソッド名，その入力フローのアクションに引数，出力フローのガードに返し値を記述すること

表 1　不整合内容に対応するユーザフィードバック例

評価値		不整合内容	ノート内容
1.Critical	11	図内にクラスを一つも持たない	Unable to find the Class in this ClassDiagram
	21	アクティビティ図の要素がクラス図に無い	Unable to find the [element] in ClassDiagrams
	31	ノードが途中で途切れている	Flow is broken at [node]
	41	分岐・合流/ループの開始と戻りの不一致	Unable to match any node with [node]
2.Warning	11	クラス図の要素がアクティビティ図に無い	Unable to find the [element] in ActivityDiagrams
	31	開始から終了の間で出現しないノード	[node] does not appear until FinalNode

3.2　アクティビティ図に対する構造検査

アクティビティ図に記述された実行系列に対し，構造検査を行う．このとき，コード生成対象言語により，記述可能となる構造も変わる．例えば，C++言語では goto 文によるラベルジャンプが可能であるが，Java 言語では記述することはできない．提案手法では，Java 言語を対象としたコード生成を目標とするため，対応する構造も Java 言語でサポートされた制御構文・例外処理となる．

今回，記述可能とする制御構文･例外処理は，if/switch-case 文や for/while 文，break/continue 文，try-catch 文とする．開始ノードから終了ノードへと到達できない処理手順や到達不可能なノード，サポートする各制御構文･例外処理に対して設けた記述規則を逸脱した構造を，不整合であるとして検査を行う．

4　各構造に対する記述規則

4.1　制御構文の構造記述

UML アクティビティ図において，if/switch-case 文や for/while 文といった構造中の分岐・合流表現には，ディシジョン・マージノードを用い，分岐中の処理と合流後の処理の明確化を図る．分岐･合流構造記述として，ディシジョンノードのすべての出力フローが同じマージノードの入力フローとなる構造として判定を行う．

ループ構造記述では，ループ条件となる true/false が記述されているか，ループの入りと戻りを表現するノードのフロー数が適当であるかを確認する．また，ループの戻りから入りへの遷移間は他のノードの記述を行わないものとして構造判定を行う．

break/continue 文の記述はコネクタを用いて行う．このとき，記述可能な箇所はループ構造下のみとし，遷移先はループ終了先またはループの先頭と明示的であるため記述は行わないものとする．

4.2　例外処理の構造記述

try-catch 文のアクティビティ図上での記述表現は，if/switch-case 文や for/while 文といった構造記述と同様に行うことは困難であるため，ステートチャート図を用いた例外処理 [3] を参考にし，依存関係を用いた疑似階層化による try-catch 文記述を行った．各階層とその内部の開始･終了ノードに依存関係を作ることで階層を表現している．階層となるノードは，TryBlock 層，CatchBlock 層とそれに内包される各 Catch 処理層，FinallyBlock 層であり，このとき，開始・終了ノードと各階層とを依存線で結ぶことで，間にあるアクションノードも一意に同じ階層下にあると表現できる．

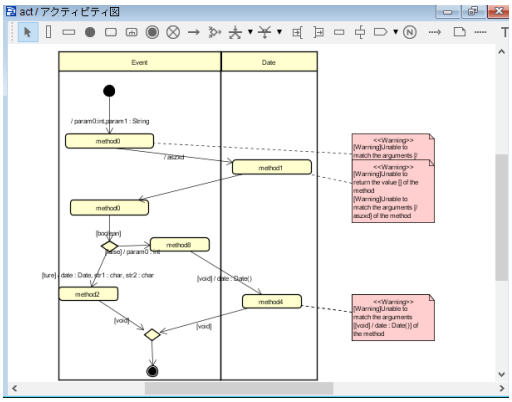


図 2　検査結果のユーザフィードバック UI の例

5　astah*プラグインの試作

提案手法によるテンプレートを用いた UML 図記述と整合性検査，ならびにユーザフィードバックを `astah*professional` のプラグインとして実現した．拡張タブ内にプロジェクト全体での不整合要素の要素名，型等の出力と，表 1 に示す不整合内容に対応したエラー文を不整合要素と紐づけた形で図上にノートとして生成し，フィードバックを行う．このとき，示された不整合内容に対し修正が行われ，不整合が解消されていれば，次の検査時にはノートが削除される．図 2 に，検査結果のユーザフィードバック UI の例を示す．例えば，ノード method4 の引数である date:Date() はクラス図上の定義と一致していないため，対応したエラー文がノートとして生成されている．

6　まとめと今後の課題

自動コード生成を目的とした上で利用される上位設計群の，テンプレートベースでの作成と整合性検査の提案，検査部の一部実装を行った．今後の課題として，生成対象言語別にサポートされる制御構文・例外処理の構造テンプレートの作成や，対象ドメイン別デザインパターンの作成を進めることで，生成対象の拡大を行う．また，構造テンプレート内の要素にラベリングを行うことで，整合性検査やプロトタイピング生成の効率化を図る．

参考文献

[1] 村林, 多田, 和崎：VDMJ と Apache Axis2 を用いた上流工程におけるモデル実行環境の構築, FIT2014 講演論文集, (B-108), pp.155–158, 2014

[2] 株式会社チェンジビジョン：astah*professional, http://astah.change-vision.com/ja/product

[3] G.Pintér, I.Majzik：Modeling and Analysis of Exception Handling by Using UML Statecharts, LNCS 3409, pp.58–67, 2004