実効性のあるモデル駆動開発のための 関心事の分離による段階的な要求仕様の洗練手法

川合怜[†]　　松浦佐江子[†]

モデル駆動開発は高品質なソフトウェア製品を効率的に開発するための有用なアプローチである．我々は，インタラクティブシステムを対象に，設計でよく用いられる UI の分離を一つの関心事としたユースケース分析を提案してきた．しかし，開発初期の段階からソースコードを自動生成できる程，機能要求や非機能要求のトレーサビリティを保持して一貫性のあるモデルを定義することは難しい．そこで，UI と内部ロジックに対して例外処理の分離を行い，それぞれの関心事で明らかになったことをモデル要素と OCL により徐々に分離の意図を持たせる．これにより，設計としての統合時に要求のトレーサビリティを保証する．

A Refinement Method of Requirements Specification by “Separating Concerns” towards Effective MDD

SATOSHI KAWAI[†]　SAEKO MATSUURA[†]

Model driven development is a promising approach to develop high quality software products efficiently. We have proposed requirements analysis method based on use case analysis and separating concerns of UI and internal logic that is often used in such popular design model. However, it is difficult to define consistent model with requirements traceability from which generates source codes automatically at the early stage of development. Therefore, Separating exceptional concerns from UI and internal logic by using model elements and OCL description makes is possible to specify the concerns by modules. As a result, the traceability of requirements against the design can be guaranteed.

1. はじめに

モデル駆動開発[1]は，高品質なソフトウェア製品を効率よく開発するための有用な手段である．現在，実用的な MDD ツール[2][3]が複数提供されている．しかし、要求を分析したモデルの機能・非機能要求が，設計としてアーキテクチャに伴うフレームワーク等の要求を定義したモデルに対して，どう関連付けられているかを正しく意味付けできない限り，生成したソースコードの品質は期待できない．

そこで，インタラクティブシステムの設計に合わせて要求仕様を定義するために，UI と内部ロジックの分離を一つの関心事としてユースケース分析を行う．また，ユースケースの主系列に対して横断的な関心事である例外系列を分離する．このように関心事を定めて分析することで明らかになったことをモデル要素と UML 標準の制約記述言語 OCL（Object Constraint Language）[4]により分離した観点に意図を持たせながら要求分析を行う．これにより，分離した関心事を設計においてモジュール化することで要求のトレーサビリティが保証できる．

2. 要求分析の問題

業務系 Web システムを対象に，要求仕様の妥当性保証を目的とした要求分析モデル[5]では，要求分析から設計へ繋げやすくするために，インタラクティブシステムの設計でよく用いられる UI と内部ロジックの分離を一つの関心事としてユースケース分析を行う．このユースケース分析の特徴は，UML アクティビティ図によるアクション系列としての基本フロー，オブジェクトノードによる振る舞いとデータの関連，ガードの明記による例外フロー，パーティションによる振る舞い主体者の役割を定義する．そして，ユースケースの定義を Web UI に特化することで Web UI プロトタイプを自動生成する．これにより，生成したプロトタイプから顧客や開発者がシステムの妥当性を確認できる．

しかし，最終イメージに近いプロトタイプを生成するために，機能の呼び出しやキャンセル等のすべての操作性を実現できるようにあらゆるインターフェースを詳細に設計しようとするとモデルが複雑になる．また，インタラクションに係るデータ（入出力データ）からシステム内部に係るデータ（Entity データ）の成立観点や，システムの処理を表すアクションは，レビューによってモデルから目視で確認できても，仕様としての妥当性は確認できない．

3. 提案手法

3.1 提案プロセス

提案手法のプロセスを図 1 に示す．本手法では，要求分析モデルにおける UI を関心事とした基本フローの定義（図 1（1））を出発点とする．基本フローでは，ユースケースの成功シナリオとしてシステムへの入力，期待される出力結果，その結果を生成する過程を構成要素として定義するが，その中にはいくつかの関心事が混在する．そこで，UI に加えて不整合の要因である例外処理と内部ロジックを関心事

† 芝浦工業大学大学院 理工学研究科
Graduate School of Engineering and Science, Shibaura Institute of Technology

として段階的にモデルを洗練していく．以下，各関心事における要求を分析して定義する段階，定義した要素の整合性を検証する段階，保証した定義を設計へ系統化する段階について説明する．

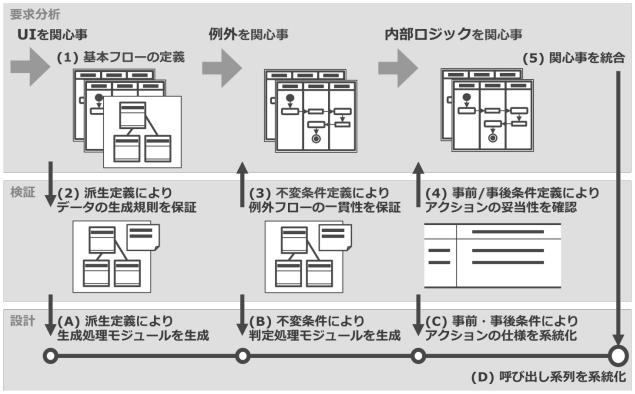

図 1　提案手法の概要

3.2 派生定義によるデータの妥当性確認

要求分析モデルでは，入出力データから Entity データの成立観点の十分性を確認できていない．これは，UI の分離による弊害であり，分離に対する統合としてそれぞれのデータを形成する関係を定義できなければならない．そこで，Entity データに対する入出力データの派生関係を OCL で定義する（図 1（2））．派生関係はデータの観点からアクションに対して制約を与える定義であるため，データ間で処理すべき計算方法を一意に識別できることから，データの生成規則として必要十分性を確認できる．また，設計時には派生定義から生成処理を得ることができる（図 1（A））．使用性向上のために入力方式が変化した場合には，計算方法を派生関係として定義し直すように，インターフェースの自由度を容易に変更できる．

3.3 不変条件による例外フローの定義

基本フローはユースケースを成功裏に実現するアクション系列であり，アクションの対象データからみた基本フローに対する不変条件に起因して例外フローが発生する．不変条件は，基本フローの定義によってデータ間の関係とデータと振る舞いの関係が見えてきているため，定義することができる．そこで，クラス図に対して不変条件を OCL で定義し，ユースケースの共通視点であるクラス図から不変条件名を用いてガードを記述することにより，分岐条件の曖昧性，ユースケース間の不整合を解消する（図 1（3））．また，複雑化の要因であるあらゆる例外応答を実現するために例外毎に判定と例外通知アクションを追加していた問題に対して，アクティビティ図における例外フローは，その遷移先を規定できればそれぞれのアクションを設計時に決定しても全体のフローに影響しないことから，これらのアクションを省略し，復帰位置が同一な例外フローを一つにすることでフローの複雑化を避ける．判定処理については，設計時に不変条件から生成することができる（図 1（B））．

3.4 事前・事後条件による内部ロジックの整理

内部ロジック内のアクションは，メソッドの候補となることから，アクションの妥当性を精査できなければならない．そこで，アクションの自然言語記述に対象となるインスタンス名を明記し，ユースケースの事前・事後条件は内部の振る舞いに対する定義にすることで，記述の一貫性を持たせる．また，各ユースケースにおける事前・事後条件，内部ロジック内のアクションを一覧できる表を生成し，OCL によってアクションの事前・事後条件を上記の自然言語記述から洗練していくことで冗長なアクションの選定を回避する（図 1（4））．前述のように，基本フローに対して不変条件や派生関係の観点からアクションやデータの整理により，判定や生成を行うモジュールを定義してきた．これにより，内部ロジック内のアクション系列は，基本フローの本質的な振る舞いとして注力できる．また，アクションの事前・事後条件は設計時のメソッドの仕様として系統化でき（図 1（C）），性能向上のために処理時間を考慮して派生属性を導出する際のきっかけにも利用できる．

3.5 関心事の統合

ユースケースの呼び出し関係を表すナビゲーションモデルをアクティビティ図で表す（図 1（5））．ユースケースの接合には，事前・事後条件を用いる．また，前述の定義からユースケースを成立させる各関心事の呼び出し関係や，それぞれの定義の結果から得られたモジュールを系統化できるため（図 1（D）），設計ではそれらの要素に対するコンポーネントを決定していけばよい．

4. おわりに

本稿では，関心事毎のユースケース分析により，実現する機能要求が明らかになりつつある場面で OCL 記述を含むモデル定義により，要求の曖昧性や矛盾に対する品質を作り込む方法を提案した．また，定義から得られた結果の呼び出し関係を決定することで要求を系統化する方法を提案した．ポスター発表では，クラス間の関係の明記が重要となるインタラクティブシステムとして図書管理システムを事例に提案手法の有効性を議論する．

今後は，物理的に存在する実体に障害が発生した場合のシステム上の正確性や，ステークホルダ毎に異なるシステムへの関心事について検討する．

参考文献

1) Object Management Group, Model Driven Architecture, http://www.omg.org/mda/
2) AndroMDA, http://www.andromda.org/index.html
3) BridgePoint, http://www.mentor.com/priducts/sm/bridgepoint
4) Object Management Group, Object Constraint Language, http://www.omg.org/spec/OCL/
5) 小形真平，松浦佐江子：UML 要求分析モデルからの段階的な Web UI プロトタイプ自動生成手法，コンピュータソフトウェア，vol.27，No.2，pp.14-32，2010.