1P2-1in

ユースケース記述の構造化とその活用

Modeling of Use Case Description and Its Application

竹内広宜[*1*2] 中村大賀[*1] 山口高平[*2]
Hironori TAKEUCHI Taiga NAKAMURA Takahira YAMAGUCHI

[*1]日本アイ・ビー・エム株式会社 東京基礎研究所 [*2]慶應義塾大学
IBM Research - Tokyo Keio University

In this research, we consider to create a model from the use case description in free text format. In the development of the large software system, the many requirement documents are prepared and updated frequently. In such a situation, it is difficult to review all documents manually and generate test cases by keeping the traceability between requirements and tests. Therefore in this researh, by using the model constructed from the use case description, we also consider the system for the quality analysis of the use cases and the automated test case generation. Through the real use cases in the system development using the packaged application, we evaluate how we can reuse the existing use cases by the proposed system. As results, it is found that we can generate test cases successfully from about 90% of the real use cases through the manual improvement of the descriptions based on the feedback from the quality analysis system,

1. はじめに

ソフトウェアシステムの開発では，その上流工程で，仕様書をはじめとした文書成果物が作成される．大規模な開発では開発者同士の効果的なコミュニケーションがプロジェクトの成功に必要不可欠であり [3]，多くの開発現場で前工程までの成果物をコミュニケーション手段として利用し作業を進める成果物ベースの開発プロセスが適用されている．この成果物ベースの開発プロセスにおいて，仕様書などの文書成果物がコミュニケーション手段の大部分を占め，大量の文書成果物が作成される．この時，文書の品質が悪いと，誤った情報を伝えてしまい，後工程に重大な影響を及ぼすことが多い．例えば，テスト工程で対応する要求と異なる結果が得られた場合，大きな手戻りが発生する可能性がある．したがって，上流工程においてあらかじめ成果物に含まれる問題を取り除くことが重要となる．また，上流工程の段階で，成果物から対象システムのテストケースを作成しテスト計画を進めることで開発からテストを効果的に進めることが期待できる．しかしながら，文書成果物は大量に作成され，かつ，頻繁に更新される．このような状況で人手によるレビューを中心とした品質分析やテストケースの作成は非常に困難である．そのため，品質分析やテストケース作成の半自動化といった技術支援が必要となる．そこで，本研究では，テキスト分析技術を用いたユースケース分析支援システムを考える．

テキスト分析技術を開発工程で作成される文書成果物に適用した既存研究として，仕様書の品質上問題となる曖昧な記述を検出する試みが行われている [1, 11]．また，作成した仕様書の中から開発者間での共通理解のため，対象分野において定義を明確にすべき専門用語を抽出することも行われている [7]．さらに，専門用語だけでなく，対象分野の知識をドメインオントロジーとして構築し，要求の整合性を分析する試みも行われている [4, 5]．

本研究では，ユースケース記述を対象とし，記述をモデル化して分析・活用することを試みる．具体的には，テキスト分析技術を用いて，ユースケース記述からシステムの振る舞いとその過程を表すモデルを構築する．そして，モデルを用いた記述内容の品質分析，および，モデルを変換しテストケースを生成する支援システムを提案する．システムの活用シナリオとしてパッケージアプリケーションを利用したシステム構築を想定する．そして，システムが提示した品質分析結果を元に利用者がユースケース記述を洗練することで，テストケースが効果的に生成できることを実データを用いて検証する．

本論文の構成は以下の通りである．まず，分析対象となるユースケース記述とそのモデルとモデル化手法について 2 節で述べる．そして，3 節でモデルを利用した支援システムとその活用方法について述べ，4 節でシステムを利用することによる効果について評価を行う．その後，考察をし，結論を述べる．

連絡先: 竹内広宜 e-mail: hironori@jp.ibm.com

2. ユースケース記述のモデル化

2.1 ユースケース記述のモデル

ユースケースはある実現する機能におけるシステムとその利用者との間のインタラクションを記述したものである [2]．構築するシステムを利用する顧客視点でシステムの振る舞いが記述されているため，開発者とエンドユーザーの意思疎通に役立ち，要求を引き出しやすいという利点がある．要求の獲得後，ユースケースを元に，外部設計，詳細設計が行われる．そのため，ユースケースの記述が不明確であると，間違った設計が行われる可能性がある．ユースケースは利用者に見える振る舞いを示しているため，本来ユースケースで意図していた内容と違う理解で設計がなされた場合，最終的にできあがるシステムはユーザーが求めるものと異なる可能性が高い．そのため，ユースケースに書かれている内容はどんな単純な記述であっても，読み手によって一意に決まる内容かどうかを吟味する必要があり，記述に関する様々な留意点がまとめられ共有化されている [2]．

しかしながら，ユースケースをはじめとした文書成果物の品質は書き手の技術力や対象分野での経験に依存する．そのため，レビューなどを通してその品質を高める必要があるが，大規模システムになると大量のユースケースが作成され，頻繁に更新される．したがって，人手を中心としたレビューを限られた期間内に行うことが難しくなっている．また，仕様書を元

にテストケースを作成することが可能であるが，前述の通り，仕様書は頻繁に更新されるため，ユースケースとテストケースとの間のトレーサビリティが確保されなくなる可能性がある．

本研究では，ユースケースをモデル化し，ユースケース記述の品質チェックやテストケースの生成などに利用することを考える．ユースケースに対して自動的にモデル化することができれば，品質チェックの一部が機械化でき，レビューを効果的に行うことができる．本研究では，ユースケース記述のモデルとして Text-To-Test[9] で定義されたモデル (Use Case Description Model) を用いて記述をモデル化する．使用する Use Case Description Model とモデル化の例を図 1 および図 2 に示す．

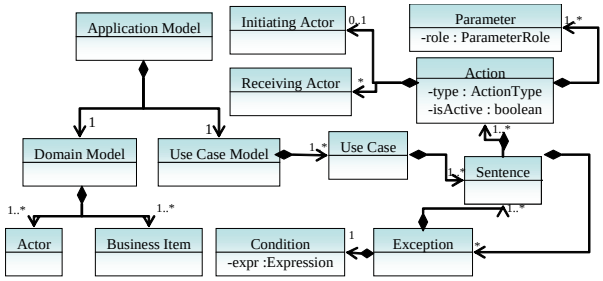

図 1: Use Case Description Model

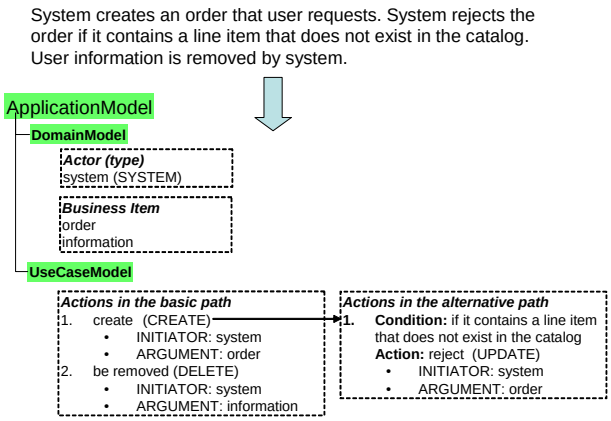

図 2: ユースケース記述のモデル化の例

2.2 モデル化の概要

図 3 にモデル化の概要を示す．入力となるユースケース記述に対して，処理コンポーネントが逐次的に適用される．各処理について以下に述べる．

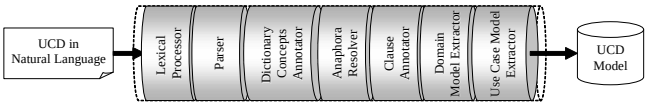

図 3: 分析の流れ

- **Lexical Annotator and Dependency Parser:** 本処理では，まず形態素解析が適用され，同定された単語に品詞，原形といった情報が割り当てられる．その後，係り受け解析（構文解析）が適用され，単語と単語の間の係り受け情報が付与される．その結果，記述中の各動詞に対して，係り元および係り先となっている単語の情報 (述語項構造:Predicate Argument Structure) を得ることが可能になる．

- **Dictionary Concepts Annotator:** 本処理では，動詞として同定された単語に対して，辞書を適用し，意味ラベルを付与する．辞書には，意味ラベルとして [8] に元に，INPUT, OUTPUT, READ, WRITE, GIVE, GET, CREATE, QUERY, UPDATE, DELETE, DELEGATE, START, STOP, BROWSE を定義した．そして各意味ラベルに該当する語を辞書に登録する．辞書が適用されない動詞には UNCLASSIFIED が付与される．

- **Anaphora Resolver, Clause Annotator:** 本処理では，省略されている主語や代名詞のの補完，そして文中の条件節と主節を同定する．

- **Domain/Use Case Model Extractor:** Domain Model の構築においては動詞の意味ラベルと主語，目的語の関係を元にしたルールを事前に定義しておく．例えば，INPUT という意味ラベルを持つ Action の主語は USER である，といったルールを定義する．そして，ルールに適合した Action ごとに主語・目的語にルールで定義された意味ラベルを付与し，Domain Model を構築する．Use Case Model の構築においては，主節に含まれる動詞のうち，連体修飾しているものを除き Action として抽出する．その時，Predicate Argument Structure から得られる主語，目的語を Action Parameter として Action に付与し，基本フロー (Basic Path) の中にモデル化する．条件節を含む文は直前の文で記述されている振る舞いに関する例外を述べている記述と判断し，主節内の Action を代替フロー (Alternative Path) の中にモデル化する．

本研究では，ユースケース記述を言語非依存なモデルに変換し，様々な分析に利用することを考えるが，モデル化の処理が言語依存ではなく，多言語化が容易である必要があり，そのためのアプローチが検討されている [10, 12]．上記で述べた処理プロセスのうち，形態素解析および係り受け解析については多くの言語でツールが作成されおり，日本語についてもいくつかの形態素解析器や構文解析器が公開されている．このようなツールを利用することで，言語によらず，各動詞に対して，Predicate Argument Structure を構築することができる．

得られた Predicate Argument Structure 中の動詞に対する意味ラベルの付与は辞書を用いて行うため，言語ごとに辞書リソースを作成することで対応する．本研究では，あるシステムに関するユースケース記述を含む要求仕様書群から頻出する動詞を抽出し，日本語と英語それぞれで 200 語を辞書に意味ラベルとともに登録した．

以上から主に言語に依存するのは，Anaphora Resolver および Clause Annotator であり，多言語を考慮した分析システムを容易に構築することが可能である [12].

3. モデル化したユースケース記述の活用

3.1 活用場面

ERP，CRM，SCM などのパッケージアプリケーションを用いた企業システムの構築では，あるプロジェクトで作成したユースケースを他の類似プロジェクトで再利用できる可能性が高い．そのため，既存のユースケースをビジネスプロセスの階層分類*1に沿ってサーバー上で保存し，顧客の要求に沿って，

*1 例えば American Productivity and Quality Center(APQC) が定義している Process Classification Framework

クライアント上で類似ユースケースを見つけ，編集し，成果物を作成する成果物管理システムが提案されている [6]．このようなシステムでは，成果物を効果的に再利用するために記述を明確にする必要となる．したがって，記述内に意味ラベルが不明な Action が存在するといった分析結果を，ユースケース記述を作成する際に提示することは有用であると考えられる．また，モデルから自動的にテストケースのテンプレート作成することができれば，ユースケース記述の変更とともに，テストケースを自動的に更新することが可能となる．

そこで本節では，ユースケースからのテストケーステンプレート作成を目的とし，ユースケース記述の検証とユースケース記述からテストケースへの変換の 2 種類の活用方法について述べる．

3.2 ユースケース記述の検証

ここでは，モデルを利用したユースケース記述の検証分析システムを示す [12].

結果はエラーと警告の 2 種類に分類され表示される．エラーとして

- Actor が不明 (Missing Actor)
- Action が分類できない (Unclassified Action)
- システムの内部動作が記載されている (System Internal Behavior)

を検出する．これらの項目は，ユースケースには，利用者とシステムとの間の振る舞いを記述し，動作とその動作主を正確に記載するというユースケース作成のガイドラインに沿ったものである．また警告として

- 1 つの記述内に複数の Action（振る舞い）が記載されている (Multiple Actions)
- 記述が受身形になっている (Passive Voice)

を検出する．これらの項目は，ユースケースは簡潔に書くというガイドラインに沿ったものである．

図 4 に記述検証分析システムにおける分析結果を示す.

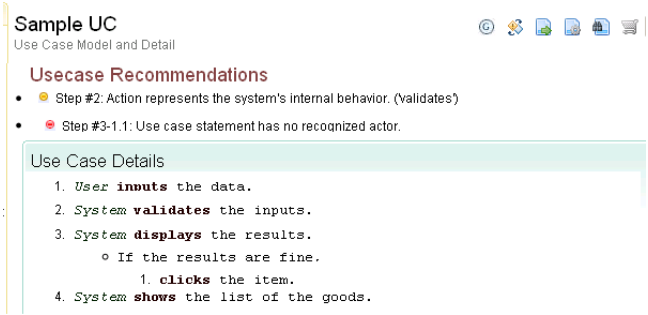

図 4: ユースケース記述の分析結果

利用者はエディタ内でユースケース記述を編集する．記述を保存すると，ユースケース記述のモデル化処理が自動的に起動し，2 節で述べた処理が適用されモデルが生成される．生成されたモデルを元に Actor や Action の情報がエディタ中でアノテーションされる（斜体や太字）．そして，モデルを用いて記述内容の品質分析して得られたフィードバックがエディタ上側に表示する．利用者がフィードバックを元に記述を修正し，保存すると再度分析が実行される．

(http://www.apqc.org/process-classification-framework) がある

3.3 テストケーステンプレートの自動生成

ここでは，ユースケースから作成されるユーザーテストで用いるテストケースを生成するシステムを示す．ユーザーテストには，ユーザーが実行する手順と，その手順を実行した際に期待されるシステムの振る舞いが明記される必要がある．そこで，ユースケースの各 Statement（記述）の Actor と Action を元にテストケーステンプレートを下記の手順で生成する．

- Actor の意味ラベルが USER の場合，ユーザーが実行する Execution Step を生成する
- Actor の意味ラベルが SYSTEM の場合，システムの振る舞いを確認する Verification Step
- Execution Step の生成では，Actor を含まない命令文形式で出力する
- Verification Step の生成では，Validate that ¡Action 記述¿の文形式で出力する
- 条件記述を含む記述はテスト実行の際のパラメータ情報を含むので，そのままの形式で出力し，利用者が必要に応じて変更する

なお，上記の品質分析で検出された項目を含む記述からはテストケースを生成せず，元の記述をそのまま出力する（テストケース生成時にユーザーに，”記述内に問題点を含む記述があり元記述がそのまま出力される”とい内容のメッセージを提示する）．生成されるテストケーステンプレートの例を図 5 に示す.

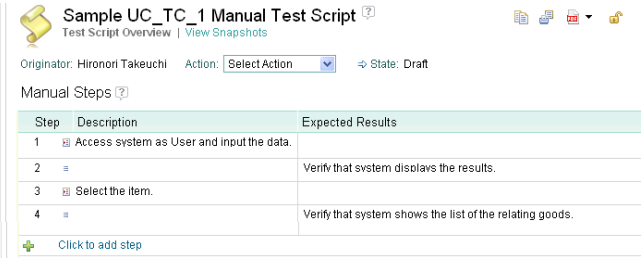

図 5: テストケーステンプレートの生成結果

4. 評価

パッケージアプリケーションを利用し人事システムの構築した際に作成したユースケースから 72 記述を評価に用いた．これらのユースケースは別の人事システムを構築する際に再利用できる可能性がある．本節では，ユースケース記述のモデル化を活用したシステムでユースケースが効果的に再利用できるかどうかの評価を行う．各ユースケース記述に含まれるステップの数（文数）は平均 16.4 であった．モデルを用いた品質分析を行ったところ問題点が検出されなかったユースケース（データセット A）は 28 であった．これらに対しては直接テストケース生成処理を適用した．次に，何らかの問題点が検出された残り 44 ユースケース（データセット B）については検出項目に基づいて記述の変更を人手で行い，その後テストケース生成処理を適用した．生成されたテストケーステンプレートに対して，記述が可読で，正しいテスト記述が生成されているかどうかを人手で評価した．表 1 に評価結果を示す.

この結果から，品質分析を適用しユースケース記述を洗練することにより，既存ユースケースを再利用する際に，同時にテストケースを効果的に作成することができることがわかる．

表 1: テストケーステンプレートの生成結果

	ユースケース数	精度
データセット A	28	28/28 (100%)
データセット B	44	38/44 (86.4%)
全データ	72	66/72 (91.2%)

5. 考察

データセット B から生成されたテストケーステンプレートに対する評価結果から，品質分析結果を元にユースケース記述を書き換えても完全なテストケースが生成できない場合があることがわかる．これは，品質分析で検出できなかった問題点が存在するために，正しいテストケースが作成できないからである．例えば，記述中に括弧付きで”Note that ···”のような但し書きが書かれている場合などで正しいテストケースが作成されなかった．このような表現はユースケース記述としては，制限事項として別途構造化して記述する内容であり，全ての問題をモデル化を通して検出するのは難しい．しかしながら，データセット A と B の比較から何らかの問題が検出されたユースケースは潜在的な問題も含んでいる可能性が高い．したがって，品質分析で問題点が検出されたユースケース記述について検出箇所以外も精査することで，問題点の大部分を網羅することが可能である．

ユースケース記述のモデル化では，記述中の Action に対して意味ラベルを付与する．Action の抽出のために，動詞に意味ラベル付与する辞書リソースを用意する必要があるが，本研究ではコーパスに出現する高頻度の動詞を元に同じ数 (200) の辞書リソースを準備した．しかしながら，適用するデータで初めて出現する動詞もあり，そのような場合は”Action が分類できない”という問題点が検出されてしまう．分析対象データを用いて効果的に辞書リソースを拡張する方法の検討が今後の課題の一つである．

6. まとめ

本研究では，ユースケースを対象にシステムの振る舞いとその流れをモデル化して分析することを考え，テキスト分析技術を用いたユースケース記述分析支援システムの構築を試みた．具体的にはユースケース記述のモデル化を通して品質分析をし記述を洗練化するシステムとテストケーステンプレートを自動生成するシステムを提案した．パッケージアプリケーションを用いてシステム構築で作成されたユースケース記述を用いた評価の結果，システムによるフィードバックを元に利用者が記述を洗練化することにより，約 90%のユースケースからテストケースを正しく生成できることがわかった．この結果より，ユースケースが再利用可能な場面において，本研究で提案したシステムが効果的に利用できることがわかった．今後の課題としては，対象ごとに必要となる辞書などのリソースの整備方法や，対象データを利用したリソースの拡張方法を効果的に行うことが考えられる．

参考文献

[1] D. M. Berry and E. Kamsties. Ambiguity in requirements specification. In *Perspectives on Software Requirements*, pp. 7–44, Kluwer, 2004.

[2] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000. (邦訳 : ユースケース実践ガイド, ウルシステム株式会社 監訳, 山岸 耕二, 矢崎 博英, 水谷 雅宏, 篠原 明子 訳, 翔泳社, (2001)).

[3] B. Curtis, H. Kransner, and N. Iscoe. A field study of the software design process for large scale systems. *Communications of the ACM*, 31(11):1268–1287, 1988.

[4] R. Hasegawa, M. Kitamura, H. Kaiya, and M. Saeki. Extracting conceptual graphs from japanese documents for software requirements modeling. In *Proceedings of the 6-th Asia-Pacific Conference on Conceptual Modeling (APCCM)*, pp. 87–96, 2009.

[5] H. Kaiya and M. Saeki. Ontology based requirement analysis: Lightweight sementic processing approach. In *Proceedings of the Fifth IEEE International Conference on Quality Software (QSIC)*, pp. 223–230, 2005.

[6] P. Mazzoleni, S. Goh, R. Goodwin, M. Bhandar, S. K. Chen, J. Lee, V. S. Sinha, S. Mani, D. Mukherjee, B. Srivastava, P. Dhoolia, E. Fein, and N. Razinkov. Consultant assistant: a tool for collaborative requirements gathering and business process documentation. In *Proceedings of ACM SIGPAN*, pp. 807–808, 2009.

[7] P. Sawyer, P. Rayson, and K. Cosh. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *IEEE Transactions on Software Engineering*, 31(11):969–981, 2005.

[8] A. Sinha, M. Kaplan, A. Paradkar, and C. Williams. Requirements modeling and validation using bi-layer use case description. In *Proceedings of MoDELS*, pp. 97–112, 2008.

[9] A. Sinha, A. Paradkar, P. Kumanan, and B. Boguraev. A linguistic analysis engine for natural language use case description and its application to dependability analysis in industrial use cases. In *Proceedings of IEEE/ACM DSN*, pp. 327–336, 2009.

[10] A. Sinha, A. Paradkar, H. Takeuchi, and T. Nakamura. Extending automated analysis of natrural language use cases to other languages. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pp. 364–369, 2010.

[11] H. Yang, A. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh. Extending nucuous ambiguity analysis for anaphora in natural language requirements. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pp. 25–34, 2010.

[12] 竹内, 中村, 山口. テキスト分析技術を用いたユースケース分析. 信学技法 KBSE2010-32, pp. 55–60, 2010.