要求仕様の一貫性検証支援ツールの提案と適用評価

位野木 万里※　　近藤 公久※

要求仕様の品質特性である「一貫性」に着目し，ベテラン技術者が経験的に得た検証知識を形式知化し，それら知識に基づき，要求仕様の一貫性検証支援ツールを実現した．実システムの要求仕様書を用いて本ツールの適用評価を行い，本ツールは技術者の効率的な仕様検証を支援する点において一定の効果があることを明らかにした．

Proposal of a Requirements Consistency Verification Support Tool and Its Evaluation

Mari Inoki※, Tadahisa Kondo※

In this paper, focusing on the consistency of the requirements quality, we propose a requirements consistency verification support tool. It helps a requirements analyst verify the requirements specification and improve the specification from the viewpoint of the requirements consistency. We have extracted tacit knowledge for verifying requirements and made it explicit as the verification knowledge which were incorporated into the tool. We have evaluated the tool by utilizing the actual requirements specifications. According to the evaluation, it was clarified that the tool is effective for an analyst to verify and improve the requirements specifications efficiently.

1　はじめに

近年，国内外において要求定義に関する標準化が盛んであり［ISOIECIEEE2011］［JISA2011］，真の顧客要求を反映させた魅力あるソフトウェアの要求を定義することは，産業界が重視する重要な課題である．要求定義に関する標準や知識体系が策定され，各企業はそのような標準や知識体系に基づき要求定義を実践しつつある．しかし，実際の要求定義は開発対象となる領域，組織が直面する課題，利用する技術などの条件に応じて工夫が必要となり，標準が示す一般化されたやり方のみでは対応が困難である．例えば，要求定義工程で定義した要求仕様の品質に関しては，要求工学知識体系REBOK［JISA2011］によれば，完全性，トレーサビリティ，一貫性などの品質特性が定義されているものの，現状では，各検証は組織のベテラン技術者が，各自の属人的な方法により実施している．初級の技術者が要求定義を実施することは失敗のリスクが高く，要求定義はベテランの技術者のみが従事することとなり，効率的な要求定義の実施は困難な状況にある［JISA2014］．

要求仕様書の検証のしやすさを考慮して，ダイアグラムや形式言語を用いて仕様を厳密に記述する取り組みも行われている．しかし，ソフトウェア要求は，構築してみなければ明らかにならない仕様が多いことや，要求仕様の分析者や要求の源泉となるステークホルダはそのようなダイアグラムや形式言語の専門家でないことから，自然言語により仕様を記述するという状況は増加傾向にある．例えば，アジャイル開発などの開発スタイルの需要が高まっているが［Hiranabe2013］［Rasmusson2011］，アジャイル開発ではユースケースシナリオ［Cockburn2000］や，ユーザストーリー［Cohn2004］［Wang2014］などの自然言語による仕様により要求仕様を記述している．したがって，自然言語で記述された仕様を対象に，仕様の品質を一定に保つための

※　工学院大学

取り組みは極めて重要である．

木村らは，自然言語で記述された要求仕様の検証支援ツールを開発し，同ツールを用いた要求仕様の高品質化への取り組み事例を報告している［Kimura2014］．［Kimura2014］による検証技術は，設計要素の一つであるアクターにフォーカスし，ベテラン技術者による，アクターの表記ゆれを防止するための検証ノウハウをツール化した点が特徴である．しかし，［Kimura2014］の成果は，アクター定義の観点で一貫した要求仕様を作成することに限定した，特定企業内での成功例であり，ノウハウやツールの一般化には至っていない．

本研究では，要求仕様の一貫性の検証の範囲に，機能への入出力となる「データ」，アクターとシステムとのインターフェースとなる「画面」，システムの働きに相当する「振る舞い」の設計要素を追加する．設計要素の定義漏れや表記ゆれの検証ノウハウは，対象ドメインや組織の条件により様々と考えられるため，組織がカスタマイズ可能なように柔軟な形式知化を図る．形式知化した検証知識を検証支援ツールに組み込み，検証を自動化し，初級技術者が効率的かつ適切に要求仕様を検証することを目指す．

以下，本稿は次のように構成する．2節において，自然言語で記述された要求仕様の検証技術の現状を整理する．3節では，2節で示す現状の技術動向を踏まえ，自然言語で記述された要求仕様の検証技術に対する本研究のアプローチを示す．4節では，要求仕様の一貫性検証知識の形式知化について説明する．5節では，開発した要求仕様の一貫性検証支援ツールについて説明する．6節では，実際の仕様書に対して，当該ツールを適用した結果を示す．7節では，ツールの有効性や妥当性について考察し今後の課題を整理する．8節で本稿をまとめる．

2　自然言語で記述された要求仕様の検証技術に関する関連研究

自然言語で記述された要求仕様を記述するために，Pohlらは機能要求仕様を記述するためのテンプレートを提案した［Pohl2015］．テンプレートでは，英語で記述された一つの機能要求文は，条件，主語（システム名），助動詞，動詞，目的語，目的語の詳細により構成するとしている．日本語で記述された要求に対して，Pohlらの提案をそのまま適用することは困難であるが，テンプレートが示す構成要素を満たすように要求を記述することは，要求仕様の品質向上に貢献すると考えられる．Pohlらのテンプレートにおいて，目的語には，データや画面が相当するが，それら用語自体に表記ゆれがあれば，要求文自体のあいまい性は解消されない．よって，要求文を構成する設計要素に焦点を当てた，使用する用語の一貫性の検証を支援することが必要である．

Lucassenらは，アジャイル開発においてユーザ要求の記述によく使われるユーザストーリーの品質を安定させるための品質モデルと支援ツールAQUSAの開発及び評価を報告している［Lucassen2015］．Lucassenらはユーザストーリーの品質評価のフレームワークとして，Syntactic, Semantic, Pragmaticの視点に分類される14個のメトリクスを定義した．AQUSAは，Atomic, Minimal, Explicit dependencies, Uniform, Uniqueの5つのメトリクスによる検証を自動化しているが，Semanticsの観点のUnambiguousの検証はスコープ外である．自然言語で記述された要求仕様の品質向上には，Semanticsの観点からのUnambiguousの検証，すなわち，表記ゆれなどの用語の一貫性検証が不可欠と考えられる．

日本語の自然言語で記述された仕様書の検証やレビューについても研究が行われている［Kouno2010］［Kuno2012］．河野らは，ドキュメント内において，あいまいさや不備につながりやすいキーワードの洗い出しと，そのようなキーワードのチェックツールを考案した．キーワードとしては，「～場合」などの条件を示す用語，「あれ」「これ」などの指示代名詞などが含まれる［Kouno2010］．

久野らは，同じ語句でも，使われ方によりあいまいの度合いが異なるとして，あいまい性の高い語句を選択的に検出することを目的に，語句のあいまい性を判定するツールを提案した［Kuno2012］．久野らが提案するツールでは，あいまい性のレベルを，曖昧語，準曖昧語，非曖昧語に分けて検出する．提案されたツールを実際の仕様書にて評価したところ，複数の解釈が可能な曖昧語を高いレベルで網羅的に検出でき，仕様書の修正に対して有効であることを確認している．

先行研究［Kouno2010］［Kuno2012］が着目した「あいまいさにつながりやすいキーワード」は重要なものの，要求仕様の品質向上には，対象ドキュメント内で定義された設計要素のあいまいさの解消が更に重要である．例えば，ある設計要素を示す用語が，あいまいさにつながりやすいキーワードには非該当でも，その設計要素が，別の語句で定義されたり，明確な定義なく機能要求などに使用されていれば，その機能要求の内容は，開発者や分析者など複数の読み手により解釈が異なる内容につながるリスクが高い．自然言語で記述された要求仕様に対しての検証支援のためには，特定のキーワードに加えて，要求仕様の本質的な定義内容に踏み込んで，設計要素のあいまいさや不整合を指摘する検証を行うことが必要である．

前述したように，木村らは，要求仕様の一貫性をツールにより検証することで要求仕様の安定化を図ることに取り組んだ事例を報告した［Kimura2014］．これは，設計要素の一つであるアクターにフォーカスし，ベテラン技術者による，アクターの表記ゆれを防止するための検証ノウハウを「用語不一致検証ルール」と「定義漏れ検証ルール」の機能を備えたツールとして実現した点が特徴である．例えば，表1のアクター定義と，図1のシナリオ記述が要求仕様書内に記述されているとする．シナリオ記述に出現するアクターは，ユーザ，レジ係，顧客，店員，顧客（会員）である．これらは，表1のアクター定義には未定義である．

表1　要求仕様書内のアクター定義例

No.	アクター定義
1	ユーザ（管理者）
2	ユーザ（担当者）
3	オペレータ
4	マスタ管理者
5	運用管理者

要求仕様の検証では，これらが現行のアクター定義の表記ゆれなのか，アクター定義からの定義漏れなのかを確認する必要がある．仮に，現行のアクター定義表のまま，システムへのアクセスコントロールを実装したとする．実装後，「レジ係」というアクターが別途必要になった場合，「レジ係」のアクセスコントロール機能の拡張のため，作業の手戻りが考えられる．

［Kimura2014］では，要求仕様書内のアクター定義とシナリオ記述を対象に，アクター定義表からの定義漏れを指摘するアクターの定義漏れ検証と，「ユーザ」と「ユーザ（担当者）」などの表記ゆれを指摘する用語不一致検証を自動化し，検証レポートを生成した．しかし，［Kimura2014］における要求仕様の検証のノウハウは，アクター定義の一貫性にフォーカスした，特定企業の分析結果に基づいて定義された内容（図2）であり，様々な組織での活用を想定した一般化には至っていない．一貫性検証の対象となる，要求仕様を構成する設計要素の種類を拡張することや，様々な組織での活用を想定した柔軟性のあるツールのアーキテクチャの提供が必要と考えられる．

（シナリオ1）

1. ユーザは，レジに購入したい商品を持参し，ポイントカードを提示する．
2. レジ係は，商品情報およびポイントカード情報を端末から入力し，ポイント残高，残金を顧客に提示する．
3. ユーザは，提示された金額（残金）を支払う．
4. 店員は，支払われた金額（残金）を入金する．
5. システムは販売管理システムと連動してレシート（購入商品，金額，利用ポイント，残高ポイント）を出力する．ポイント付与ルールに基づいて残高ポイントがアップデートされる．
6. レジ係は，商品とレシートを顧客（会員）に渡す．

図1　要求仕様書内のシナリオ記述例

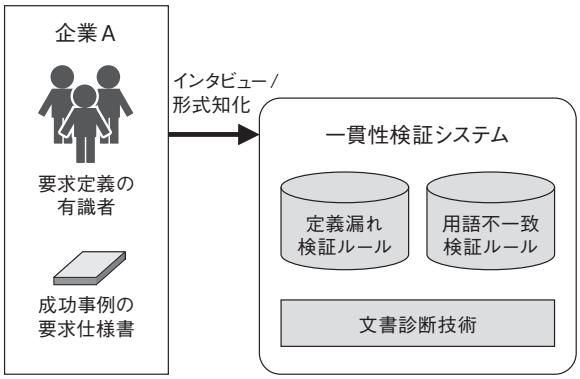

図2　一貫性検証支援ツールの構成

3　研究のアプローチ

2節の内容を踏まえ，自然言語で記述された要求仕様に対して，要求仕様の品質特性である「一貫性」に着目した検証手法のノウハウの形式知化とツールサポートに関する研究課題を以下に整理する．

課題1：要求仕様の一貫性検証知識の形式知化
課題2：要求仕様の一貫性検証の支援ツールの開発
課題3：要求仕様の一貫性検証の支援ツールの評価

以降，各課題の解決策のアプローチを整理する．

3.1　課題1：要求仕様の一貫性検証知識の形式知化

先行事例［Kimura2014］では，自然言語で記述された仕様説明内における「アクター」を対象とし，「用語不一致検証ルール」と「定義漏れ検証ルール」の2種類の対象企業に固有の検証知識の形式知化と支援ツールによる自動検証にとどまっていた．本研究では，研究成果を広く展開するため，検証知識の抽出範囲を複数企業に拡張する．また，検証の対象を要求仕様書全体に一般化した上で，検証の観点を，アクターに加えて「データ」「画面」「振る舞い」に拡張する．そして，複数企業の有識者インタビューを実施し，設計要素の一貫性に関する検証知識を抽出し，それらを共通部分と可変部分に仕分け，検証ルール及び辞書として形式知化する．

3.2　課題2：要求仕様の一貫性検証の支援ツールの開発

課題1で定義した検証ルール及び辞書を組み込んだ，要求仕様の一貫性検証を支援するツールを開発する．開発するツールは，要求仕様書内で言及されている，「アクター」「データ」「画面」「振る舞い」の記述が，要求仕様書内の記述と整合していることを検証する．検証の種類としても，用語定義完全一致，NGワード検証，NGワードを妥当な用語に置換するシナリオ自動生成の機能を追加する．開発するツールの考え方と構成を図3に示す．

［Kimura2014］では，技術や環境の変化に伴う検証ノウハウの拡張の方式については言及されていない．本研究では，検証ルール並びに辞書の共通部分を共有すると共に，各社が容易にそれらの維持管理と拡張を実施可能にするため，検証のエンジン，検証機能，検証ルール，辞書を独立させたアーキテクチャとする．更に，各社が独自に検証エンジンの拡張を可能とするために，形態素解析エンジンはOSSを用いる．

3.3　課題3：要求仕様の一貫性検証の支援ツールの評価

実システム開発で用いられた要求仕様書に対して適用評価を行い，有効性を検証する．

課題1で開発した検証知識を組み込み，課題2で開発したツールを，課題1でインタビューした有識者及びその関係者に提示し，適用評価し，検証のノウハウが妥当に具現化されていることを確認する．評価は，インタビューを実施した複数企業に対して，複数の視点から確認することとする．一貫性の検証にあたっては企業において実際に開発に用いられた仕様書を適用し評価する．

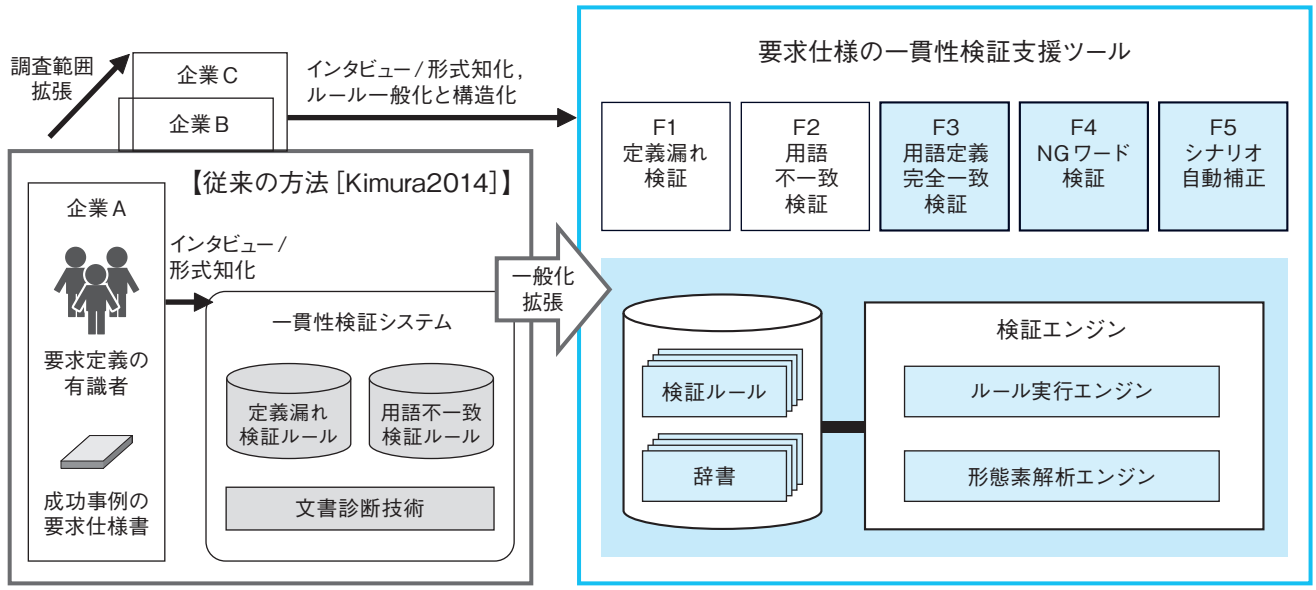

図3　要求仕様の一貫性検証支援ツール

4　要求仕様の一貫性検証知識の形式知化

先行研究[Kimura2014]の成果を踏まえ，既存の仕様書の分析，及び4社5部門の有識者にインタビューを行い，「アクター」「データ」「画面」「振る舞い」の一貫性に関する要求仕様の検証ノウハウを分析し整理した．分析・整理の結果得られた検証ルールと辞書の構造を図4に示す．

要求仕様の一貫性検証ルールは，定義漏れ，用語不一致，用語完全一致，NGワードの4つの検証ルールから構成した．これらのルールに基づき，要求仕様書を検証するには，自然言語で記述された要求仕様から設計要素を特定する必要がある．「アクター」「データ」「画面」「振る舞い」の識別ルールは，要求仕様に出現する名詞句や動詞句からそれぞれの用語を特定するルールである．各設計要素の識別ルールの実行には，各設計要素を特徴付ける用語を定義した識別辞書を用いる．また，同一意味で表現の異なる表記ゆれについても識別辞書と識別ルールを定義した．各ルールの定義を表2に示す．

例えば，アクター識別ルールには，「アクターとみなした用語の後ろに括弧で囲まれた文字列があれば，それらを連結してアクターとしてみなす」などが定義されている．要求仕様書内には，「ユーザ」「ユーザ(会員)」「ユーザ(非会員)」などのアクター名が記述されるが，本ルールに基づけば，これらはアクターとして特定することができる．要求仕様書内に，「ユーザ」しか定義されていなければ，用語不一致検証ルールによって「ユーザ(会員)」「ユーザ(非会員)」はユーザの表記ゆれ候補として指摘することができる．また，表記ゆれ辞書に，「ユーザ」と「利用者」が異音同義語として定義されていれば，「ユーザ」は「利用者」の表記ゆれとして指摘できる．

設計要素別の識別辞書の定義例を以下に示す．各設計要素では，「○○者」「○○ユーザ」などの表記や，辞書に定義された用語の複合名詞を当該設計要素とみなす．例えば，シナリオ中に，「顧客(会員)」「顧客(非会員)」などと記述されていれば，アクターを記述する辞書の「顧客」と「会員」の複合名詞として，これらアクターをアクター用語とみなす．

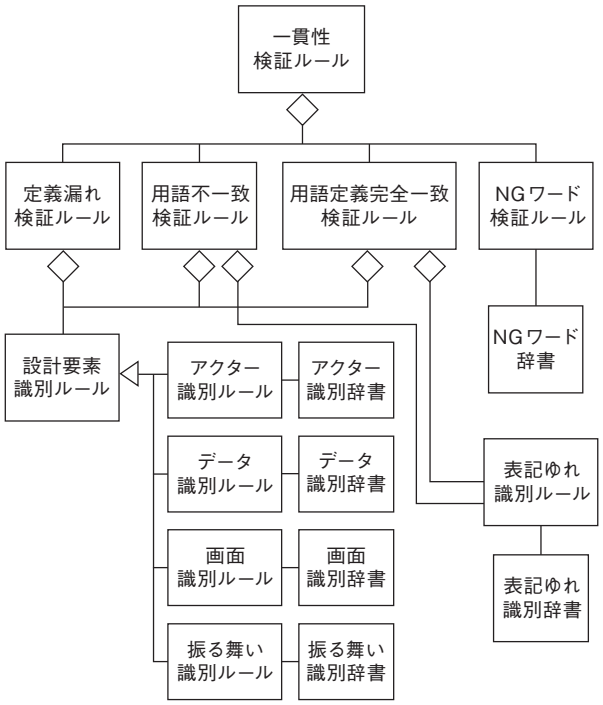

図4　検証ルールの構造

表2　検証ルールの説明

No.	ルール	ルール説明
1	定義漏れ検証	要求仕様書中に定義されていないにもかかわらず，本文中で利用されている設計要素を検証する．
2	用語不一致検証	要求仕様書では表記ゆれした形式で定義されている設計要素の用語不一致（表記ゆれ）を検証する．
3	用語定義完全一致検証	要求仕様書にて利用されている設計要素は，要求仕様書中に定義されており，かつ，定義された設計要素は，要求仕様書内で1回以上利用されていることを検証する．
4	NGワード検証	要求仕様書に出現するNGワードを検証する．NGワードはNGワード辞書にて定義する．
5	アクター識別	要求仕様書からアクター用語を識別する．アクター用語として特徴付ける単語はアクター識別辞書にて定義する．
6	データ識別	要求仕様書からデータ用語を識別する．データ用語として特徴付ける単語はデータ識別辞書にて定義する．
7	画面識別	要求仕様書から画面用語を識別する．画面用語として特徴付ける単語は画面識別辞書にて定義する．
8	振る舞い識別	要求仕様書から振る舞い用語を識別する．振る舞い用語として特徴付ける単語は振る舞い識別辞書にて定義する．
9	表記ゆれ識別	要求仕様書の複数の用語から表記ゆれ関係を識別する．表記ゆれ関係の用語は，表記ゆれ識別辞書にて定義する．

●**アクター識別辞書**：者，部，部門，会社，局，課，グループ，チーム，組織，ユーザ，会員，顧客，客，お客様，社員，従業員，員，委員，メンバ，オペレータ，運用者，管理者，監督者，人，市，市民，市職員，受託業者，オフィス，国，都道府県，市町村，業者，署，係，ユーザー，メンバー，オペレーター，市，町，村，都，道，府，県

●**データ識別辞書**：情報，データ，オブジェクト，帳票，書，ドキュメント，票，ファイル，調書，リスト，ID，コンテンツ，結果，パスワード

●**画面識別辞書**：画面，スクリーン，ディスプレイ，ページ，ウェブサイト，ホームページ，メッセージ

●**振る舞い識別辞書**：する，実施，実行，管理

●**NGワード**：全て，あらゆる，既存と同じ

●**表記ゆれ辞書**：利用者，ユーザ，ユーザー

5　要求仕様の一貫性検証支援ツールの開発

課題1を通して得られた，検証ルールと辞書に基づき，開発したツールの機能を表3に示す．本ツールは，F1～F4までの検証機能とF5のシナリオ自動補正から構成した．検証ルールと辞書は，組織や対象ドメインにより異なると考えられるため，拡張可能性を考慮し，検証ルール，辞書，検証エンジンを分離したアーキテクチャとした．更に辞書はテキストファイルで提供した．検証ルール及び検証エンジンの構成を図5に示す．検証エンジンは，ルールを実行するエンジンと，自然言語で記述された要求仕様を解釈する形態素解析エンジンで構成する．検証実行では，TemplateMethodパターンを用いて，設定ファイルに基づき，文書処理，用語抽出，検証，検証結果出力の一連の動作を実行する．各組織によって，用語抽出のエンジンを独自のものに変更するような場合には，検証実行クラスを別クラスに派生させて定義することを想定している．

本ツールは，Windows® 8.1 OS環境の下で動作可能とし，形態素解析ソフトウェアMeCab［MeCab］を用いた．開発言語はRuby［Ruby］であり，実装したクラス数は63，ステップ数は約4.4Kとなった．開発期間は6カ月，開発工数は約8人月，このうち，検証ルールと辞書の開発工数は約2人月である．

利用者が設定ファイルで書き分けることで，機能F1～F5を使い分ける．ツールは，記述された設定ファイルに基づいて，必要な検証ルールを選択・適用し検証を行う．検証結果は，検証レポート及び改善シナリオとして出力される．なお，入力シナリオ，設計要素定義とあるのは，要求仕様書に記述されたテキストファイルである．出力する検証レポート及び改善シナリオもテキストファイルである．

アクター用語の用語不一致検証のみの検証を行う条件を設定ファイルにおいて定義した場合の，一連の処理の流れを図6に示す．

表3　要求仕様の一貫性検証支援ツール機能一覧

ID	機能および概要
F1	定義漏れ検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，それぞれ，対応する定義表に定義されていることを確認し，未定義であれば指摘する．
F2	用語不一致検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，対応する定義表と別の表現（表記ゆれ）で記述されている場合に，その表記ゆれを指摘する．
F3	用語定義完全一致検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，対応する定義表に定義されていること，かつ，「アクター」「データ」「画面」「振る舞い」などの定義表に定義された各要素が，シナリオ中に1回以上出現していることを確認し，未定義または出現しない場合にそれを指摘する．
F4	NGワード検証
	検証対象の仕様（シナリオ）及び，NGワード定義表を入力として，NGワードの定義表に定義された用語がシナリオ中に出現していれば，それを指摘する．
F5	シナリオ自動補正
	検証対象の仕様（シナリオ）及び，NGワード定義表を入力として，NGワードの定義表に定義された用語がシナリオ中に出現していれば，同じくNGワード定義表に定義された置換候補用語でシナリオを置換し，改善シナリオを生成する．

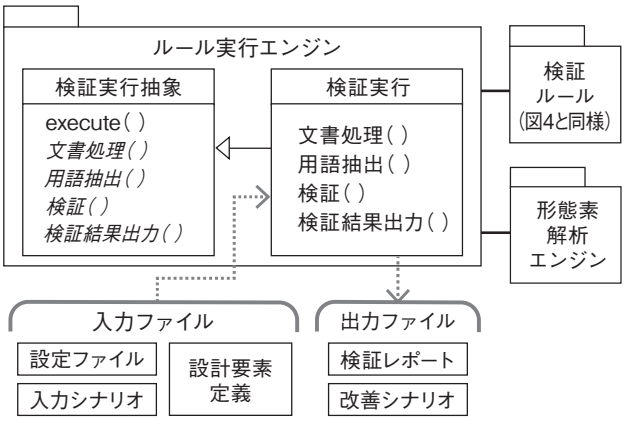

図5　検証ルール及び検証エンジンの構成

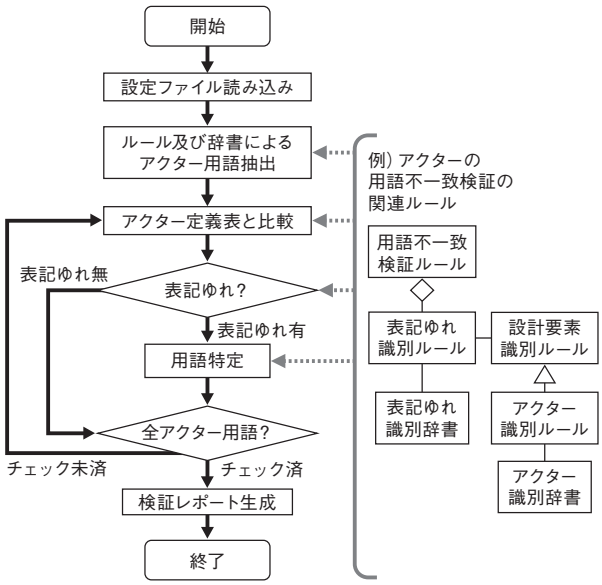

図6　アクターの用語不一致検証のフロー

6　要求仕様の一貫性検証支援ツールの評価

実システム開発で用いられた以下2件の要求仕様書に対して，要求仕様の一貫性検証支援ツールを用いて検証を実施した．これらは日本語で記述されたドキュメントである．それぞれの文字数は(a) 17,635，(b) 66,048である．

(a)某ウェブサイトリニューアル事業要求仕様書
(b)某制度システム構築・運用等業務調達仕様書

ツール評価は主に2つのステップで実施した．

ステップ1） ツールが組み込んだ検証ルールの意図に沿って，適切に指摘を行っていることを確認した．そのために，上記仕様書に対して，設計要素：アクター，データ，画面，振る舞い及びNGワードを対象に，定義漏れ検証，用語不一致検証，用語完全一致検証，NGワード検証，シナリオ自動補正を著者らの人手により実施し，その結果と，ツールによる検証結果を突き合わせて確認した．

ステップ2） 研究課題1の際にインタビューした有識者及び関係者に対して，ステップ1)の結果を提示して再インタビューを実施した．インタビューにおいて，検証のノウハウが妥当に具現化されていること，ツールによる検証結果の妥当性，ツールの有効性，操作性，適用可能性について評価した．

6.1　ステップ1)の結果

図7は仕様書(b)を本ツールで検証した結果の抜粋である．図7において，仕様書内の「交付申請情報」はデータ定義に定義されていないため，不一致として検出できることを示している．なお，「交付申請情報」が「交付金申請・決定情報」と同義語である(つまり表記ゆれである)ことまでは，現状のツールでは検出できる状態にはない．これは，文字列が部分一致するか，表記ゆれ辞書に記述するかによって，表記ゆれ候補を抽出しているためである．

表4にツールの定義漏れ，用語不一致，用語完全一致に関する再現率，適合率を示す．表記ゆれの特定と，統一化のための指針の抽出は100%には至らぬものの，平均して，再現率90%以上，適合率82%以上，F値0.84～1.00の結果を得た．NGワード検証は，NGワード辞書で定義した用語は100%特定でき，あらかじめ用意したNGワードに対する置換用語を用いてシナリオ自動生成が行えた．

仕様書(a)のアクターの定義漏れ，表記ゆれ検証では，ツールによる誤指摘が10件あるものの，53件の定義漏れ／表記ゆれの指摘が行えた．ツールによる適切な指摘としては，「管理者」「閲覧者」「利用者」など，アクターとして定義されていない，あいまいなアクター名が仕様書内に記載されていたことが含まれ

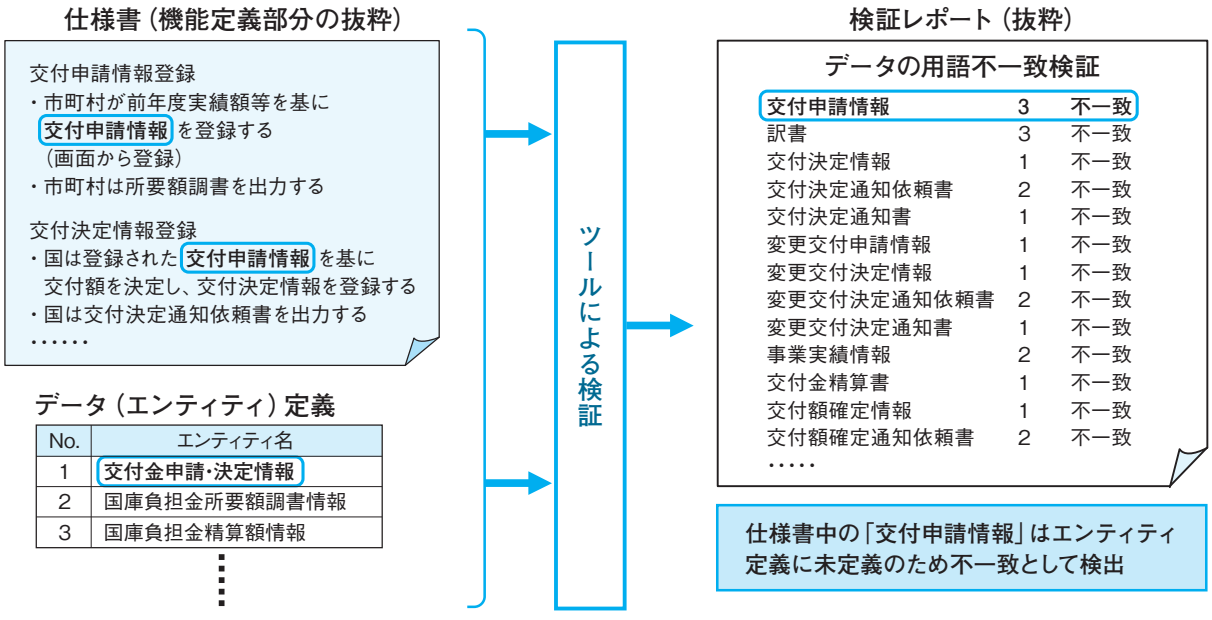

図7　仕様書(b)の検証結果例の抜粋

表4　検証結果抜粋

No.	仕様書	検証名	検証対象	人手の指摘数	ツールの指摘数	指摘漏れ数	誤指摘数	一致指摘数	再現率(%)	適合率(%)	F値
1	(a)	定義漏れ検証	アクター	62	63	9	10	53	85	84	0.84
2			データ	63	64	0	1	63	100	98	0.99
3			画面	30	31	2	3	28	93	90	0.91
4			振舞い	550	659	8	117	542	99	82	0.90
5		表記ゆれ検証	アクター	62	63	9	10	53	85	84	0.84
6			データ	63	64	0	1	63	100	98	0.99
7			画面	30	31	2	3	28	93	90	0.91
8			振舞い	550	659	8	117	542	99	82	0.90
9		完全一致検証	アクター	62	65	9	12	53	85	82	0.83
10			データ	63	64	0	1	63	100	98	0.99
11			画面	30	31	2	3	28	93	90	0.91
12			振舞い	550	659	8	117	542	99	82	0.90
13	(b)	定義漏れ検証	アクター	139	147	13	21	126	91	86	0.88
14			データ	209	210	12	13	197	94	94	0.94
15			画面	15	15	0	0	15	100	100	1.00
16			振舞い	1,370	1,644	13	287	1,357	99	83	0.90
17		表記ゆれ検証	アクター	139	147	13	21	126	91	86	0.88
18			データ	209	210	12	13	197	94	94	0.94
19			画面	15	15	0	0	15	100	100	1.00
20			振舞い	1,370	1,644	13	287	1,357	99	83	0.90
21		完全一致検証	アクター	139	147	13	21	126	91	86	0.88
22			データ	253	254	12	13	241	95	95	0.95
23			画面	70	70	0	0	70	100	100	1.00
24			振舞い	1,370	1,644	13	287	1,357	99	83	0.90

ている．なお，アクターとして，「高齢者・障がいのある人」「作成したページについての変更権限を持たない課」「専門知知識を有しない者」「コンテンツの所有部署」「コンテンツの担当(作成)部署」「担当部署」「所有部署」に対して，ツールは「アクター」用語として特定できず，定義漏れや表記ゆれの検証で指摘がすり抜ける結果となった．「部署」をアクター用語の識別辞書に登録し，再現率を向上させる対策が考えられる．

データに関しては，形態素解析エンジンMeCabの特性により，「内訳書」が「内」と「訳書」に分割して認識されたため，「所要額市町村別内訳書」「変更所要額市町村別内訳書」「精算額市町村別内訳書」は，データ用語として特定できなかった．これらはMeCabの特性に起因し，現状ではツール側での改善は困難なため，ユーザマニュアルにて識別困難であることを注意喚起する対策が考えられる．

6.2　ステップ2）の結果

本ツールによる仕様書の検証結果について，4社5部門の有識者にインタビューを行った．インタビューによれば，2つの実事例による検証結果は一定のレベルに達している，という評価を得た．すなわち，検証レポートの内容は有意義な指摘が提示されており，数千～数万文字からなる仕様書において，数十件以上の未定義やあいまいな表現の指摘を人手で実施することは困難であるが，本ツールを用いることで，初級の技術者でも検証レポートの指摘事項に従って仕様書を改善していくことが現実的に実施可能であると考えられる．今回は，既に開発が完了している案件に対しての要求仕様書への検証レポートである．したがって，定義漏れや表記ゆれの指摘が存在していたとしても，実際の開発において悪影響がどの程度あったかどうかまでは分析できていない．しかし，検証レポートに出力される抽出用語一覧を確認することで，出現頻度の高い用語や，用語の使われ方のパターンが把握でき，仕様書に対する理解が深まるという点で有用であるというコメントを得た．

また，検証レポートは数十ページにわたるボリュームになることや，テキストデータの羅列により直観的かつ効率的に指摘事項を特定して改善に取りかかることが困難であることから，検証レポートの見せ方の工夫が重要との指摘を受けた．また，検証レポートに基づき仕様書を改善する過程で，利用者が指摘結果を採用したかどうか，それに基づき修正を実施したかどうかを学習し，指摘事項を効率的に対処する仕組みが今後の課題であることを確認した．

本研究の成果は，主にはエンタープライズ系システムの企画立案，要求定義工程において，要求仕様書，提案書，調達仕様書，操作仕様書，製品取扱い説明書，業務マニュアルなどを対象に，初級技術者がセルフチェックするほか，次のシーンでの活用が考えられることを確認した．

- **担当者が作成した仕様書の管理者によるチェック**
- **検証ルールや辞書を用いた知識継承や人材育成**
- **抽出したアクターを用いたステークホルダ分析**
- **要求仕様書と基本設計書の検証レポートの比較による設計要素**

の不統一個所や設計漏れのチェック

- **プロダクトライン型開発の各モデル間の比較による，共通部分と可変部分の切り分けチェック**

7　考察

本研究では，要求仕様の一貫性検証のために，NGワードの指摘に加えて，システム開発で重要な設計要素となる「アクター」「データ」「画面」「振る舞い」の定義漏れと表記ゆれの指摘に着目した．実仕様書によるツールの適用評価や有識者インタビューにより，仕様書から設計要素を抽出すること，それらの定義漏れや表記ゆれを指摘することは，要求仕様の内容確認や誤った理解防止に効果的と考えられる．

本ツールでは，あらかじめ基本的な検証ルールと辞書を提供した．要求仕様の検証の経験のない組織でも，スムーズに要求仕様の検証に取り組むことが期待できる．様々な分野の要求仕様書に対して本ツールによる検証を実施するには，識別辞書や表記ゆれ辞書をカスタマイズする必要がある．本ツールはこれらの辞書をテキストファイルで提供している．辞書の拡張は，用語が確定していれば，1人H程度で対応可能と考えられる．なお，提供した検証ルールでは不足で，検証ルールを追加するには，検証ルールクラスの派生が必要なため，記述量にもよるが拡張コストは0.25人月程度必要と考えられる．

組織別に定義した設計要素の識別辞書は要求仕様の一貫性検証に関する専門知識であり，組織の資産として知識継承に活用可能である．また，本ツールはプロダクトライン型開発でのモデル間の仕様書の整合性の検証など，様々な活用が期待できる．

本ツールの評価者は，検証ルール及び辞書の構築の際にインタビューをした有識者である．したがって，本ツールにより初級技術者でも的確な一貫性検証が可能であることの十分な確認には至っていない．今後，評価者及び評価案件の対象を拡大し，ツールの妥当性の評価を継続し，評価過程で明らかにした課題の解決に取り組んでいく．

要求仕様の一貫性検証支援ツールを構築する際に利用した形態素解析エンジンの制約や，入力する仕様書の記述内容や対象組織での業務ルールや商習慣により，ツールが出力する検証結果には，誤指摘や当該組織にとり不要な指摘が15%程度含まれる場合がある．本ツールの適合率の向上には，形態素解析エンジンの条件や対象組織に固有の特性を考慮し，指摘対象からは除外すべき用語情報の学習と，ツールによる検証を対象組織用に進化させる技術開発に取り組むことが重要と考えられる．また，現状の検証レポートは数十ページに及ぶ場合がある．管理者や分析者の仕様書の改善作業を効率化するため，検証結果を図式化，定量化するなどが必要と考えられる．

8　まとめ

本研究では，要求仕様の品質特性である「一貫性」に着目し，「アクター」「データ」「画面」「振る舞い」の設計要素が，要求仕様書で一貫した定義で記述されていることを確認する検証ノウハウを，ルールと辞書として形式知化し，それら知識に基づく要求仕様の一貫性検証支援ツールを実現した．

本研究において，実システム開発で用いられた仕様書に対して適用評価を行い，有効性を検証した．有識者に対してインタビューを行い，当システムの有効性，妥当性，適用可能性の観点から考察し，システム開発の企画や計画段階での設計要素候補の抽出や，プロダクトライン型開発でのモデル間の仕様書の整合性の検証への活用が期待できることを明らかにした．

謝辞

本研究は，独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（SEC：Software Reliability Enhancement Center）が実施した「2015年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けたものである．

【参考文献】

[Cohn2004] M. Cohn, User Stories Applied: for Agile Software Development. Redwood City, CA, USA: Addison Wesley, 2004.
[Cockburn2000] Alistair Cockburn, Writing Effective Use Case, Addison Wesley, 2000.
[Hiranabe2013] 平鍋健児，野中郁次郎，アジャイル開発とスクラム～顧客・技術・経営をつなぐ協調的ソフトウェア開発マネジメント，翔泳社，2013.
[ISOIECIEEE2011] ISO/IEC/IEEE 29148:2011, Systems and software engineering — Life cycle processes — Requirements engineering, 2011.
[JISA2011] 情報サービス産業協会，REBOK企画WG，要求工学知識体系，近代科学社，2011.
[JISA2014] 情報サービス産業協会 REBOK企画WG，要求工学実践ガイド，近代科学社，2014.
[Kimura2014] 木村隼人，北川貴之，位野木万里，要求仕様書の品質向上に向けた活動報告 ～一貫性検証の形式知化および自動化～，情報サービス産業協会 SPES2014 経験報告 要求工学S4a，2014.
[Kouno2010] 河野哲也，猪塚修，藤森麻紀子，本間周二，茂中義典，キーワードベースドレビュー，ードキュメントのあいまいさや不備に着目したレビュー手法ー，ソフトウェアテストシンポジウムJaSST 2010，http://jasst.jp/archives/jasst10e/pdf/C2-3.pdf，2010.
[Kuno2012] 久野綾子，平尾英司，五藤智久，仕様書の曖昧性を検出するツールの試作と評価，電子情報通信学会，電子情報通信学会総合大会講演論文集 2012年_情報・システム（1），27，2012-03-06，2012.
[Lucassen2015] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper, Forging High-Quality User Stories: Towards a Discipline for Agile Requirements, Proc. of IEEE 23rd International Requirements Engineering Conference, pp. 126-135, 2015.
[MeCab] MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://taku910.github.io/mecab/#download
[Pohl2015] Klaus Pohl and Chris Rupp, Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant, 2nd Edition, Rocky Nook, 2015.
[Rasmusson2011] Jonathan Rasmusson（著），西村直人（監訳），角谷信太郎（監訳），近藤修平（訳），角掛拓未（訳），アジャイルザムライ―達人開発者への道，オーム社，2011.
[Ruby] Ruby: https://www.ruby-lang.org/ja
[Wang2014] X. Wang, L. Zhao, Y. Wang, and J. Sun, The Role of Requirements Engineering Practices in Agile Development: An Empirical Study, Proc. of the Asia Pacific Requirements Engineering Symposium, ser.CCIS., vol.432, pp.195–209, Springer, 2014.