独立行政法人情報処理推進機構 委託

2015 年度ソフトウェア工学分野の先導的研究支援事業
「要求定義の高品質化のための要求仕様の整合性の検証
知識の形式知化と一貫性検証支援ツールの開発」
成果報告書

平成 28 年 2 月

工学院大学

本報告書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」の公募による採択を受け工学院大学情報学部（研究責任者　位野木万里）が実施した研究の成果をとりまとめたものである.

目次

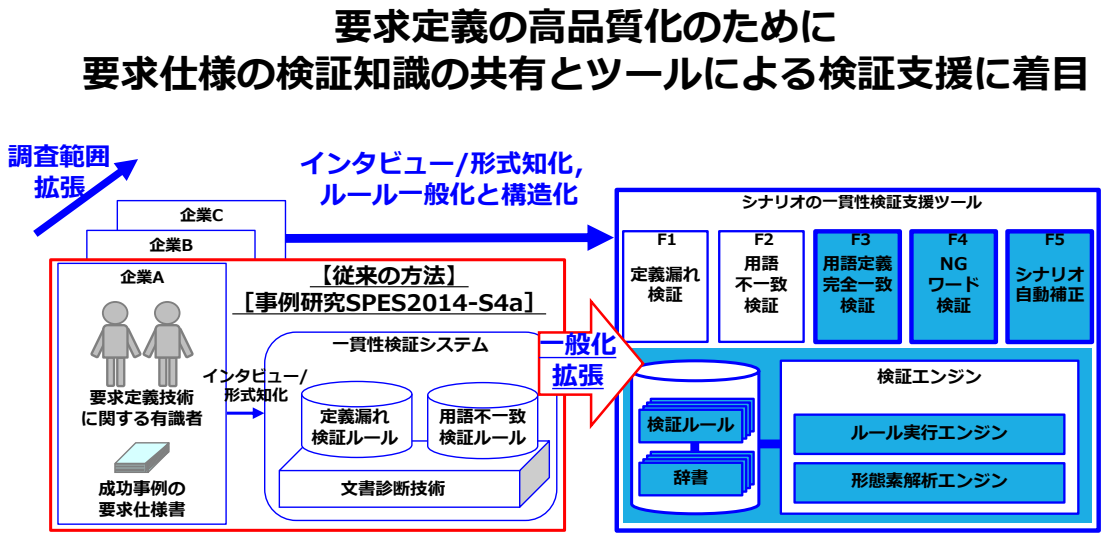
研究成果概要

1. 研究の概要

本研究は要求定義の高品質化のために，要求仕様の検証知識の共有と，ツールによる検証支援に関する研究である．本研究により，業界全体で要求定義のノウハウを共有するとともに，我が国における要求定義の技術力のレベルアップを目指すものである．本研究では，先行研究である「技術シンポジウム SPES2014 の SPES 事例研究（経験報告）2014 年　10 要求工学　S4a　東芝ソリューション株式会社による「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～」の有益な成果に着目した．本成果は，一企業に限定された固有の検証ルールによるツールであったことから，検証ノウハウの調査範囲を複数企業に拡張し，有識者へのインタビューを行い，ノウハウをルールの形式に一般化かつ構造化しシナリオの一貫性検証支援ツールとして組み込むことに取り組んだ．本研究の全体像を図[研究成果概要]-1 に示す．

本研究では，要求仕様の構成要素であるシナリオをとりあげ，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シナリオの整合性の検証知識を形式知化し，それら知識に基づくシナリオの一貫性検証支援ツールを実現した．開発するシナリオの一貫性検証支援ツールは，シナリオ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，要求仕様書中の記述と整合していることを検証する．

本研究において，実システム開発で用いられたシナリオに対して適用評価を行い，有効性を検証した．有識者に対してヒアリングを行い，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，プロダクトライン開発での仕様書の整合性の検証等への活用が期待できることなどを明らかにした．



図[研究成果概要]-1　要求定義の高品質化を狙った本研究の全体像

2. 研究の手順と成果物

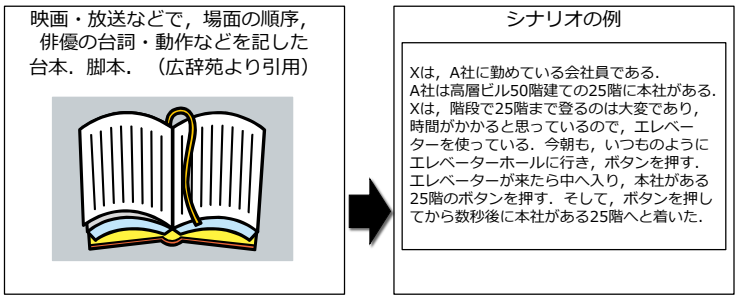
上述した本研究の目的を達成するために，次の３つの研究課題を設定し，研究を行った．課題１は，シナリオの一貫性検証知識の形式知化（形式知化）である．そのために，複数企業の有識者へのインタビューし，仕様を検証するための知識を抽出した．今回の検証対象は「シナリオ」として，検証観点は，アクター，データ，画面，振る舞いである．すなわち，シナリオ中に，アクター，データ，画面，振る舞いの観点で，定義漏れや表記ゆれなどの不整合を特定することを中心に，検証を行う．得られた検証ノウハウは，検証ルールおよび辞書として形式知化した．

課題２は，シナリオの一貫性検証の支援ツールの開発（ツール開発）である．課題１を通して得られた，検証ルールと辞書に基づき，シナリオの一貫性検証を支援するツールを開発した．検証ルール，辞書は，組織や対象市場により異なると考えられるため，拡張可能性を考慮し，ツールのアーキテクチャを定義した．

課題３は，シナリオの一貫性検証の支援ツールの評価（評価）である．研究課題１および２の結果をインタビュー先企業へ提示し有効性を確認するとともに，実案件の仕様書を用いて評価を行った．

本研究では「シナリオ」を検証の対象とした．シナリオとは，ユーザの行動やシステムの振る舞いを自然言語で物語のように記述したもので，共通フレーム 2013 中でも要件定義を実施するプロセスにおいてシナリオを記述することが定義されている．図[研究成果概要]-2 に「シナリオ」の定義を説明したイメージ図を示す．シナリオの語源は，台本や脚本などのシナリオと同様である．脚本は，登場人物のセリフにより構成されるが，ソフトウェア開発における「シナリオ」は，アクターとシステムの相互作用により構成される．つまり，シナリオとは，システムの使い方の手順を具体的に記述したもので，様々な上流工程のドキュメントにおいて記述される表現形態といえる．

現状，開発のパラダイムは，ウォーターフォール型，アジャイル型，ユーザエクスペリエンス型など，広がりを見せており，要求獲得や要求仕様化のタイミングや詳細度は異なっている．しかし，ソフトウェア・システム開発において，ステークホルダ間での要求の検証と合意形成はどのようなパラダイムでも重要であり，ステークホルダ間の理解の共有には，シナリオを用いることが有効である．つまり，シナリオの作成や理解は発注者側にも容易であり，例えばユースケースモデリングの設計手法とも親和性が高く，シナリオからスムーズに設計に落とし込みやすいという利点がある．さらに，シナリオは，テストシナリオやユーザマニュアルに活用できるため，シナリオの高品質化は，開発全体の最適化にも有効である．



図[研究成果概要]-2　「シナリオ」とは

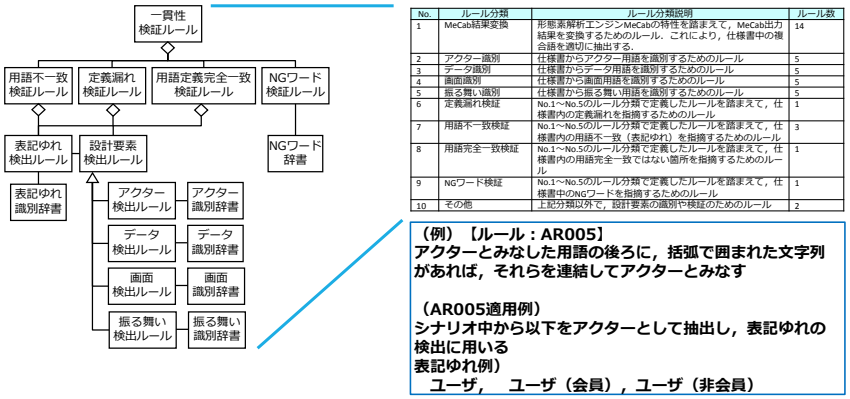
3. 開発成果

3.1 課題１：シナリオの一貫性検証知識の形式知化（形式知化）成果

先行研究の成果と，今回特に分析した 2 点の仕様書に基づいて，仕様書の分析，および 4 社 5 部門の有識者にインタビューを行い，仕様書の検証ノウハウを分析し，約 40 個のルールとして定義した．ルールはツールに組み込むためのクラス／モジュールに対応づけて実装した．ルールは，複合語の特定，アクター，画面，データ，振舞い用語の識別，定義漏れ，表記ゆれ，用語定義完全一致，NG ワードの特定などの種類に分類した．

検証ルールの構造とルールの定義結果を図[研究成果概要]-3 に示す．例えば，AR005 というルールは，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，それらを連結したアクターとしてみなす，というルールである．これにより，シナリオ中に，記述されたアクター，ユーザ，ユーザ（会員），ユーザ（非会員）などを特定し，表記ゆれ候補として指摘することができる．また，このような検証ルールは，検証対象となるシステム・ソフトウェアの対象ドメインや，組織の標準や商習慣により異なると考えられる．検証ルールおよび識別辞書を拡張可能とするために，検証の種類，設計要素の種類に応じて独立したクラスとして実装することとした．

設計要素のアクター，データ，画面，振る舞いを識別するためのキーワードに相当する辞書を，エンタープライズ系の仕様書を対象として図[研究成果概要]-4 のように定義した．



No.	ルール分類	ルール分類説明	ルール数
1	MeCab結果変換	形態素解析エンジンMeCabの特性を踏まえて，MeCab出力結果を変換するためのルール．これにより，仕様書中の複合語を適切に抽出する．	14
2	アクター識別	仕様書からアクター用語を識別するためのルール	5
3	データ識別	仕様書からデータ用語を識別するためのルール	5
4	画面識別	仕様書から画面用語を識別するためのルール	5
5	振る舞い識別	仕様書から振る舞い用語を識別するためのルール	5
6	定義漏れ検証	No.1～No.5のルール分類で定義したルールを踏まえて，仕様書内の定義漏れを指摘するためのルール	1
7	用語不一致検証	No.1～No.5のルール分類で定義したルールを踏まえて，仕様書内の用語不一致（表記ゆれ）を指摘するためのルール	3
8	用語完全一致検証	No.1～No.5のルール分類で定義したルールを踏まえて，仕様書内の用語完全一致ではない箇所を指摘するためのルール	1
9	NGワード検証	No.1～No.5のルール分類で定義したルールを踏まえて，仕様書中のNGワードを指摘するためのルール	1
10	その他	上記分類以外で，設計要素の識別や検証のためのルール	2

（例）【ルール：AR005】
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みなす

（AR005適用例）
シナリオ中から以下をアクターとして抽出し，表記ゆれの検出に用いる
表記ゆれ例）
ユーザ，　ユーザ（会員），ユーザ（非会員）

図[研究成果概要]-3　検証ルールの構造およびルール定義結果

エンタープライズ系要求仕様書を想定し設計要素識別辞書を定義

●アクター
- 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
- ○○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アクターとみなす
- シナリオ中に，「顧客（会員）」，「顧客（非会員）」などと記述されていたら，アクターを記述する辞書の「顧客」と「会員」の複合語として，この２つのアクターをアクター用語とみなす

●データ
- 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
- ○○情報，○○データなどの表現や辞書に定義された用語の複合名詞をデータとみなす

●画面
- 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
- ○○画面，○○スクリーンなどの表現や辞書に定義された用語の複合名詞を画面とみなす

●振る舞い
- する，実施，実行，管理
- ○○する，○○実施などの表現や辞書に定義された用語の複合語を振る舞いとみなす

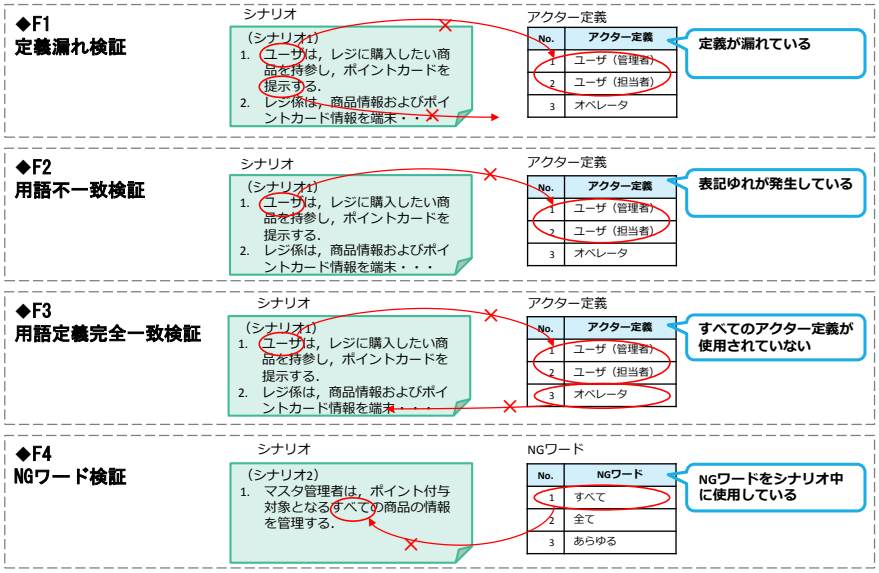
図[研究成果概要]-4　設計要素を識別する辞書

3.2 課題 2：シナリオの一貫性検証の支援ツールの開発（ツール開発）成果

課題 1 を通して得られた，検証ルールと辞書に基づき，シナリオの一貫性検証を支援するツールを開発した．開発したツールの機能を表[研究成果概要]-1 に示す．また，F1～F4 の検証機能については，検証の種類の違いの説明を図[研究成果概要]-5 に示す．

表[研究成果概要]-1　シナリオの一貫性検証の支援ツールの機能一覧

No.	機能名	ID	処理概要
1	定義漏れ検証	F1	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，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．
2	用語不一致検証	F2	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，それぞれ，対応する定義表と別の用語表現(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．
3	用語定義完全一致検証	F3	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，それぞれ，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，振る舞い等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指摘する．
4	NGワード検証	F4	検証対象の仕様(シナリオ)および，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．
5	シナリオ自動補正	F5	検証対象の仕様(シナリオ)および，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，おなじく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[研究成果概要]-5　検証の種類

検証ルール，辞書は，組織や対象市場により異なると考えられるため，拡張可能性を考慮し，ツールのアーキテクチャを定義した．本ツールでは，F1～F4 までの検証機能と F5 のシナリオ自動補正からなる機能と，検証ノウハウにあたる検証ルールと辞書，検証エンジンは分離した構造とした．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シナリオを解釈する形態素解析エンジンで構成する．

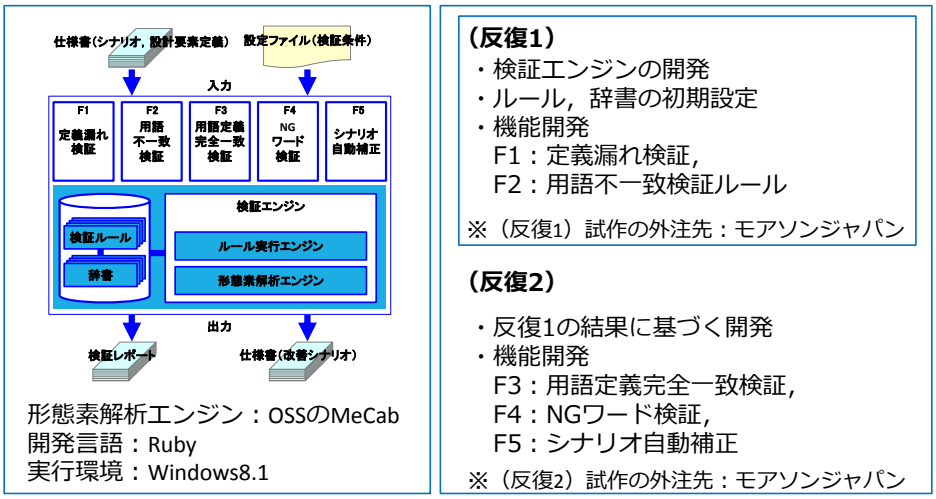
シナリオにはソフトウェアの要求仕様に固有の表現が含まれるため，OSS の形態素解析エンジン（MeCab）をそのまま用いただけでは，要求仕様の構成要素を特定することは困難であり，複合語や係り受けの関係性を辞書等で抽出可能なようにしておく必要がある．これには，調整しながらの開発が必要と考えられ，本研究では，開発期間を考慮し，2 回の反復による開発を行い，検証エンジンと機能間の調整を行いながら開発した．本ツールのアーキテクチャと各反復の開発内容の概要を図[研究成果概要]-6 に示す．

【各反復の内容】

反復 1：検証エンジン，ルール，辞書の初期設定と，定義漏れ検証及び用語不一致検証ルールを試作しツールα1 を開発する．

反復 2：残りの機能（F3～F5）について試作しツールα2 を開発する．

統合：反復 1 と 2 を統合し，チューニングを行い，ツールβ版としてまとめる．ツール試作部分は外注先に委託した．



図[研究成果概要]-6　シナリオの一貫性検証支援ツール開発アーキテクチャと反復による開発概要

シナリオの一貫性検証支援ツールのクラスモデルを図[研究成果概要]-7 に示す．ここでは，ルール実行エンジン，ルールと辞書，各種データセットに分類し，オブジェクト指向技術，デザインパターンを駆使して可変ポイントを明確化し，拡張しやすい構造とした．

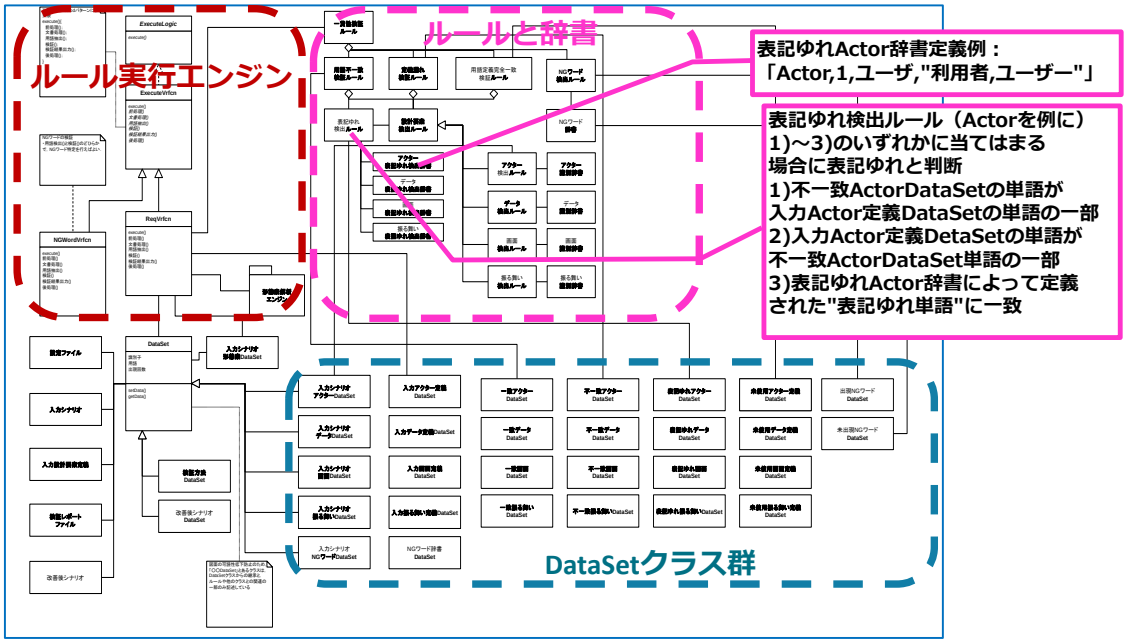
例えば，Actor に着目すると，表記ゆれ検出ルールは，シナリオに記述された Actor 用語（定義済みかどうかの処理済み）に対して，次の 1）～3）のいずれかに当てはまる場合に表記ゆれと判断する．

1)不一致 ActorDataSet の単語が入力 Actor 定義 DataSet の単語の一部となっている

2)入力 Actor 定義 DetaSet の単語が不一致 ActorDataSet 単語の一部となっている

3)表記ゆれ Actor 辞書によって定義された″表記ゆれ単語″に一致する

例） 表記ゆれ Actor 辞書に「Actor, 1, ユーザ, ″利用者, ユーザー″」と設定されている場合，シナリオから「利用者」と抽出された時，表記ゆれ候補として「ユーザ」を表示



図[研究成果概要]-7　ツール開発：設計構造

3.3 課題 3：シナリオの一貫性検証の支援ツールの評価（ツール評価）成果

ツール評価では，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．ツールの検証結果の妥当性の評価においては，以下の実際の案件での要求仕様書／調達仕様書を対象として，上述したツールの受け入れテストにて実施したテスト実施方法にて，設計要素：アクター，データ，画面，振る舞いを対象に，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実施した結果を作成した．

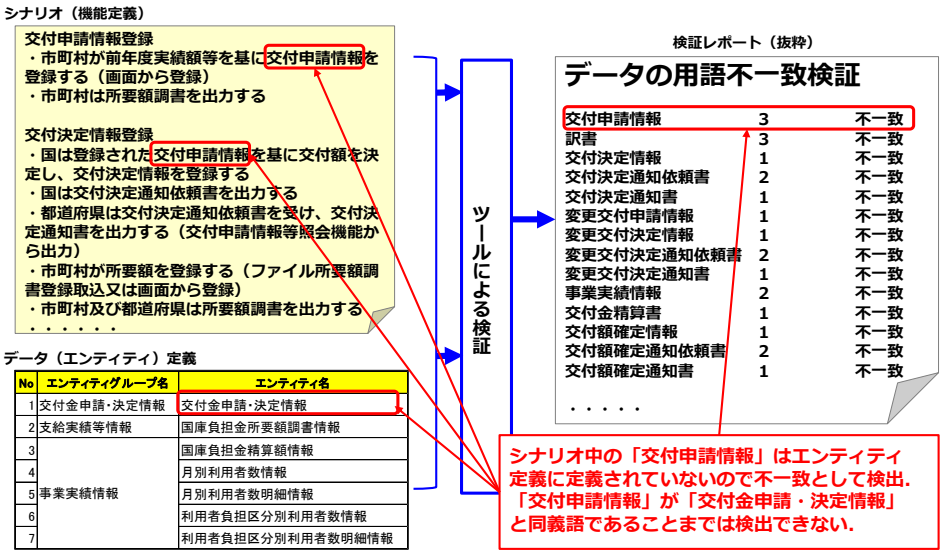
(a) 某市公式ウェブサイトリニューアル事業要求仕様書

(b) 某新制度全国総合システム構築・運用等業務調達仕様書

これらの実仕様書を用いてルール分析，テストデータ作成，ツールテストを実施した．

図[研究成果概要]-8 は，(b) 某新制度全国総合システム構築・運用等業務調達仕様書をツールにて検証した結果の抜粋である．図[研究成果概要]-8 において，シナリオ中の「交付申請情報」はエンティティ定義に定義されていないので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．このような表記ゆれの特定と，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 84%以上の結果を得た．

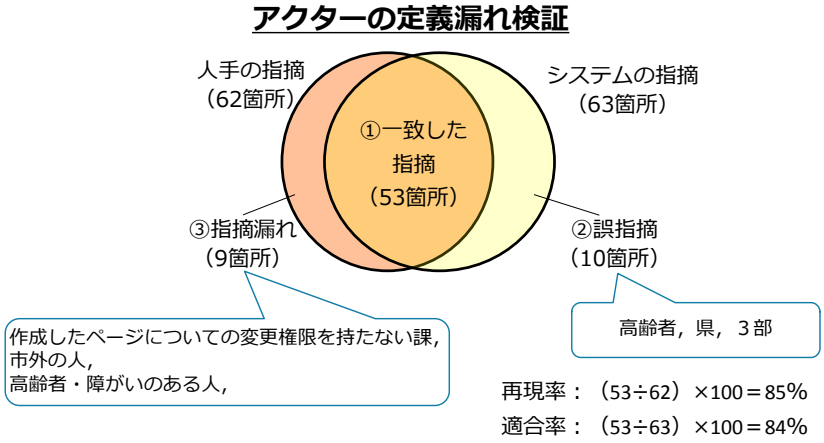


図[研究成果概要]-8　実際の仕様書による検証結果例

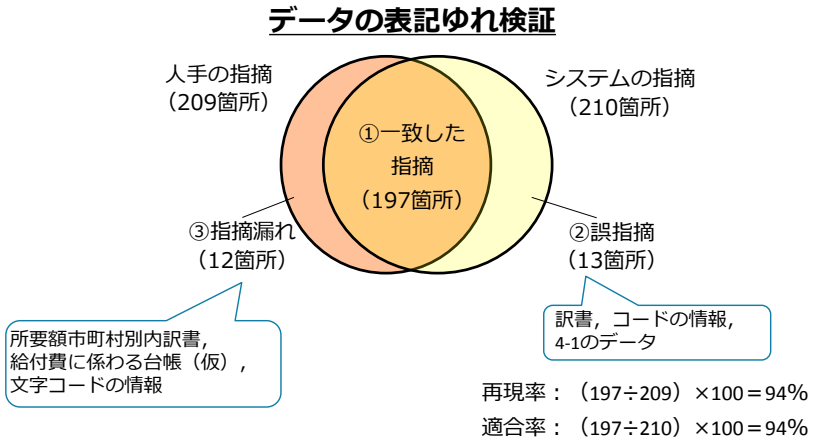
ツールの再現率，適合率について評価した結果の抜粋を，図[研究成果概要]-9 および図[研究成果概要]-10 に示す．アクターとしては，「高齢者・障がいのある人」，「作成したページについての変更権限を持たない課」，「専門知識を有しない者」，「コンテンツの所有部署」，「コンテンツの担当（作成）部署」，「担当部署」，「所有部署」について，ツールは「アクター」用語として特定できず，その結果，定義漏れや表記ゆれの検証で指摘がすり抜ける結果となった．「部署」をアクター用語の識別辞書に登録すれば，再現率は向上するなどの対策が考えられる．

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」に分割して分かち書きされるため，「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．これは，MeCab の特性に起因するものであり，現状では，ツール側での改善は困難であるため，ユーザマニュアルにおいて，識別困難であることなどを促すことで対策が考えられる．

上述したように，検証結果には誤指摘が含まれるものの，高い再現率と適合率を示しており，実際の仕様書に対する検証結果としては，現実の業務への適用に適合する一定レベルの評価結果になっていると考えられる．



図[研究成果概要]-9 ツール評価：再現率および適合率結果（1/2）



図[研究成果概要]-10　ツール評価：再現率および適合率結果（2/2）

本ツールによる仕様書の検証について，4 社 5 部門の有識者にインタビューを行った．インタビューの内容は，ツールの有効性や操作性，適用可能性，ツールへの期待である．インタビューに際して，ツール実行と検証レポート提示の実演と，上述した実仕様書の検証結果を提示した後，ディスカッションを通して，意見を聞いた．

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，初級の技術者が作成した仕様書をツールに入力し，検証レポートを得て，当該検証レポートの指摘事項にしたが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．加えて，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も得た．

しかし，検証レポートを印刷などしようとすると，数十ページに渡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掛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．

さらに，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していく仕組みが今後の課題であることなどを確認した．

4. 考察

4.1 研究による効果や問題点等

[研究による効果]

我々の提案した研究成果では，シナリオの曖昧さにつながる NG ワードの特定に加えて，システム・ソフトウェア開発で重要な設計要素となる，アクター，データ，画面，振る舞いにフォーカスした一貫性検証が行える点，これら設計要素の識別のための知識を組織別にカスタマイズし拡張が可能になっている点，検証レポートにより，シナリオに出現する用語分析結果が蓄積されるので用語の出現回数や状況から，シナリオの理解や確認が可能となる点で有効である．

検証ルール，辞書，ツールを活用することで，若手技術者でも高品質なシナリオを効率的に作成することが可能になり，産業界全体で，要求定義技術のレベルアップが期待できると考えられる．あらかじめ基本的な検証ルールと辞書を提供するため，要求仕様の検証の経験のない組織でも，スムーズにシナリオ検証に取り組むことが期待できる．

[今後の課題]

シナリオ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．要求定義の高品質化には，本ツールによる検証結果の適合率をさらに向上させるなど研究の余地が残されている．

本ツールの適合率を向上させ，誤指摘を防止し，利用者がツールの検証結果に基づいて効率的に仕様の改善が行えるように，形態素解析エンジンの条件や対象組織に固有の特性な

どから，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ていく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，シナリオの改善作業を効率化するため，検証結果の定量化およびビジュアル化が重要である．

4.2 産業界への展開と今後の研究の進め方

[成果の活用シーン]

本研究の成果は，主にはエンタープライズ系システム／サービスの企画立案，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のドキュメントを対象に，次のシーンでの活用が考えられる．使われ方のシーンをツールのユーザマニュアルに示すとともに，各企業へツールを提供し活用を促進する．

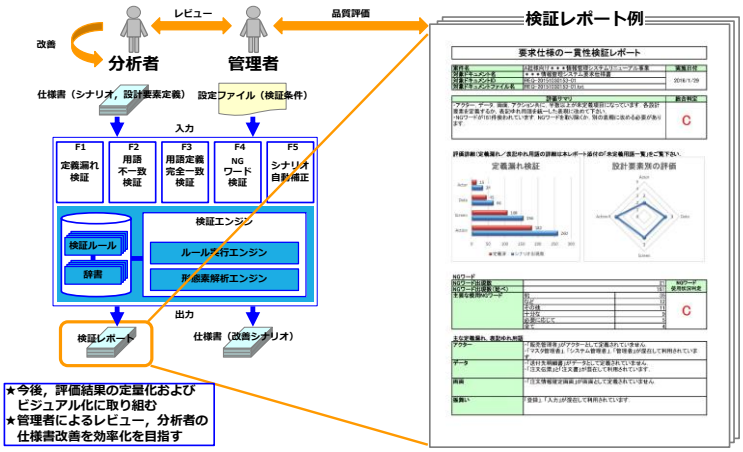
- 初級技術者が要求仕様書や提案書をセルフチェックする
- 担当者が作成した仕様書を管理者が品質チェックする
- プロジェクトメンバー全員で検証レポートを使い要求仕様書をレビューする
- 組織で蓄積した検証ルールや辞書を用いて知識継承や人材育成に取り組む
- 業務マニュアルからアクター用語抽出し，ステークホルダ分析等に利用する
- 要求仕様書と基本設計書のそれぞれの検証レポートを比較し，設計要素の不統一箇所や設計漏れのチェックを行う
- プロダクトライン型開発の各モデル間の検証レポートを作成し，共通部分が継承され，可変部分が考慮されているかを比較する

今回開発したツールはソースコード，ユーザマニュアルも含めて公開する．公開するツールは研究課題 3 の評価結果を反映した版である．研究課題 3 の評価結果に基づき改善したのは，辞書，検証ルール，ユーザマニュアルである．とくに，検証レポート中の誤指摘の発生理由や対処方法についての説明をユーザマニュアルに記述した．

ツールの公開 URL：　http://www.ns.kogakuin.ac.jp/~wwa1076

[今後の研究の進め方]

開発したシナリオの一貫性検証支援ツールの企業への普及展開活動を進め，同ツールへの学習機能の付与，検証レポートの定量化・ビジュアル化の方法を検討する．定量評価を付与した検証レポートによる仕様検証と改善サイクルのイメージを図[研究成果概要]-11 に示す．また，本ツール支援による仕様書の検証の有効性や妥当性を評価し，ツールの改善と利用ノウハウの蓄積を継続していく．



図[研究成果概要]-11　今後の研究の進め方：定量評価を付与した検証レポートによる仕様書検証と改善サイクル

1 研究の目的・背景と期待される効果

1.1 研究目的とその背景

[研究の目的]

ソフトウェア開発において，真の顧客要求に応えるためには，上流工程からの品質の作り込みは必至であり，対象ソフトウェアの機能や非機能の範囲を定める要求定義工程は極めて重要であるが，現状では，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，要求定義にはベテランの技術者のみが対応し，効率的に高品質な要求定義を実施することは困難な状況にある．

「要求定義の高品質化のための要求仕様の整合性の検証知識の形式知化と一貫性検証支援ツールの開発」（以下，「本研究」とする）では，要求仕様の構成要素であるシナリオをとりあげ，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シナリオの整合性の検証知識を形式知化し，それら知識に基づくシナリオの一貫性検証支援ツールを実現する．開発するシナリオの一貫性検証支援ツールは，シナリオ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，要求仕様書中の記述と整合していることを検証する．本研究では，シナリオの整合性の検証知識とシナリオの一貫性検証支援ツールを提供することで，業界全体で要求定義のノウハウを共有し，我が国における要求定義技術のレベルアップを目指す．そして，魅力あるソフトウェア製品の要求定義の効率化，高品質化により，ソフトウェア開発の国際競争力向上に貢献する．

[背景となる産業界の課題]

今後，情報化社会は高度に複雑化し，その中でソフトウェアのコモディティ化はさらに加速すると考えられる．高度で複雑化した社会の中でなくてはならない存在であるソフトウェアは，その品質の安定化は必至である．ソフトウェア開発において，真の顧客要求に応えるためには，上流工程からの品質の作り込みが必至であり，対象ソフトウェアの機能や非機能の範囲を定める要求定義工程は極めて重要である．

近年，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，必要となる技術等の条件に応じて工夫が必要となり，標準等の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[1]によれば，完全性，トレーサビリティ，一貫性等の品質特性が定義されているものの，現状では，各検証は各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．

[研究分野各内外での現状と動向]

◆国内外の標準化動向

要求定義の国際標準として ISO/IEC/IEEE 29148[2]が策定された．これは要求定義プロセスの作業標準であり「ステークホルダ要求」を重視している点が特徴である．国内標準とし

ては共通フレーム 2013[3]が改訂され，ISO/IEC/IEEE 29148 を踏襲し，ステークホルダ要求の明確化がタスクとして定義された．情報産業の業界団体の一つである一般社団法人情報サービス産業協会（JISA）から，要求工学知識体系 REBOK[1]とその実践ガイド REBOK 実践ガイド[4]が発行され，産業界では要求定義を重視する傾向が顕著である．

◆開発パラダイムと要求仕様の関係

近年，アジャイル型の開発が盛んに行われるようになってきた[5, 6]．アジャイル型の開発では，重厚な要求仕様を開発の初期段階で作成することは重視していない．その代わりに，ユースケースモデル[7]の定義と，主要なユースケースのシナリオ等の軽量な仕様に基づいて，プログラムを早期に開発し，振る舞いや操作性をユーザらと確認しながら開発を進めている．

従来のウォーターフォール型の開発に加え，アジャイル型の開発でも，対象システムのユースケースを定義し，使われ方のシーンを自然言語で記述したシナリオを，要求定義の成果物として活用している．また，シナリオは，ステークホルダ間での仕様理解やシステムテストシナリオへの再利用にも活用されるなど，その用途は多岐に渡り重要である．

利用者にとり魅力的なソフトウェア製品を開発するために，メンタルモデル[8]や Situation Awareness 要求等のユーザエクスペリエンスに基づく開発[9]にも注目が集まっている．このような開発対象における要求獲得の中心は，アクターの定義とアクター行動のシーンの仕様化であり，仕様化されたアクター行動，すなわちシナリオをステークホルダ間で共有することの重要性が高まっている．

ウォーターフォール型，アジャイル型，ユーザエクスペリエンス型のいずれの開発においても，要求仕様の高品質化は重要であるが，ウォーターフォール型の問題点と同様，要求仕様の検証は，属人的な方法で行われる傾向にあり，要求仕様，とくに，自然言語で記述されたシナリオを対象とした一定の検証手法が必要とされていると考えられる．

◆シナリオを用いた要求仕様化の有効性と課題

ウェアラブル端末を用いた情報支援サービス分野等の新しい分野におけるソフトウェア開発では，アジャイル型の開発やユーザエクスペリエンスに基づく開発が行われると考えられる．このような場合，重厚な要求仕様を開発初期の段階から確定することは困難である．しかし，ソフトウェア・システムが提供するサービスの使われ方のシーンをあらかじめできる限り明確化し，ステークホルダ間での理解の齟齬を防止し，ステークホルダ間での合意形成をスムーズにすることは極めて重要である．ステークホルダ間での理解の齟齬を防止するために，アクターとシステムとの相互作用を自然言語で記述する「シナリオ」[10]を用いることは，合理的かつ有効な手段と考えられる．

このような合理的かつ有効な手段である「シナリオ」を用いた要求仕様化においても，シナリオの記述内容に矛盾がある等の品質が保たれていない場合には，矛盾の解消のための問い合わせや確認の手間が発生することや，誤った理解のままの開発による不具合の発生等のリスクが高まる．したがって，「シナリオ」においても要求間に矛盾のないことを示す一貫性の検証は重要である．しかしながら，現状ではベテランの技術者に依存した属人的な検証が行われており，検証の品質にバラつきが発生するリスクがあると考えられる．

◆技術シンポジウム SPES2014 の事例研究による一貫性検証の取り組み

JISA が主催した技術シンポジウム SPES2014 の SPES 事例研究（経験報告）2014 年　10 要求工学　S4a　東芝ソリューション株式会社による「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～」（文献[11]．以下，［事例研究 SPES2014-S4a］とする）は，要求仕様の一貫性をツールによって検証することで要求仕様の安定化を図るために取り組まれた事例である．本事例は，SPES2014 のベストプラクティス賞を受賞し，産業界での共通となる要求仕様の高品質化という重要な課題に対して，実践的な解決策を示した極めて有益な事例である．

［事例研究 SPES2014-S4a］において提案された技術は，ベテラン技術者の検証のノウハウをルールとして形式知化し，それらをツールに取り込み，ツールを用いて要求仕様の検証を支援し，要求仕様への指摘事項を検証レポートとして出力する．［事例研究 SPES2014-S4a］では，検証の対象をウォーターフォール型の開発で定義された機能仕様に限定した検証を行っていたが，上述した理由から，検証の対象を「シナリオ」に拡張することで，ツールの適用範囲を向上させる必要があると考えられる．

1.2 期待される効果

(1) 研究成果の具体的な内容

本研究による成果は，SPES2014 でベストプラクティス賞を受賞し，産業界の関心が極めて高い［事例研究 SPES2014-S4a］の手法および支援ツールの取り組みを，特定企業内に限定した内容から，産業界全体で共有かつ拡張可能な方式に進化させた検証知識と検証支援ツール一式である．

検証知識とは，製品ソフトウェア開発の上流工程である要求定義工程において作成する「シナリオ」，すなわち，自然言語で記述された機能や状況の説明に対して，アクター，画面，データ，振る舞い等の要素の用語表現の一貫性を検証するためのノウハウを形式知化した「検証ルール」と「辞書」一式である．

検証支援ツール一式とは，上記知識を組み込み，かつ拡張やカスタマイズが可能な，検証を自動化するシナリオの一貫性検証支援ツール一式である．

(2) 研究成果が産業界へもたらす効果

◆研究成果が適用される場面

第一に，製品ソフトウェア開発の上流工程である要求定義工程において，ソフトウェア・システムの全体像を俯瞰する際に記述されるユースケースシナリオやシナリオ等の機能や状況説明を，複数の技術者で作成した際に，本研究成果である検証ルール，辞書，ツールが適用されることを想定している．

ウォーターフォール型の開発にとどまらず，アジャイル型，ユーザエクスペリエンス型の開発においても，ソフトウェア・システムの使われ方のシーンをシナリオにより記述する機会は増加傾向にある．したがって，第二に，産業界におけるさまざまな製品開発において，ソフトウェア・システムの全体像や使われ方のシーンをステークホルダ間で合意形成する場面で，本研究成果の適用が期待できる．

◆期待される効果

第一に，上記の検証ルール，辞書，ツールを活用することで，若手技術者でも高品質なシナリオを効率的に作成することが可能になり，産業界全体で，要求定義技術のレベルアップが期待できると考えられる．

第二に，あらかじめ基本的な検証ルールと辞書を提供するため，要求仕様の検証の経験のない組織でも，スムーズにシナリオ検証に取り組むことができる．第三に，検証ルール，辞書は，検証支援ツールに対して独立した構造となっているため，対象組織の経験に基づいて得られた独自の検証ノウハウを，ルールおよび辞書として容易に拡張でき，検証ノウハウそのものを組織の資産として蓄積することができる．

2 実施内容

2.1 研究アプローチ

2.1.1 研究の全体像

本研究は要求定義の高品質化のために，要求仕様の検証知識の共有と，ツールによる検証支援に関する研究である．

本研究では，先行研究である「技術シンポジウム SPES2014 の SPES 事例研究（経験報告）2014 年　10 要求工学 S4a　東芝ソリューション株式会社による「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～」［事例研究 SPES2014-S4a］の有益な成果に着目した．本成果は，一企業に限定された固有の検証ルールによるツールであったことから，検証ノウハウの調査範囲を複数企業に拡張し，有識者へのインタビューを行い，ノウハウをルールの形式に一般化かつ構造化しシナリオの一貫性検証支援ツールとして組み込むことに取り組んでいる．本研究により，業界全体で要求定義のノウハウを共有するとともに，我が国における要求定義の技術力のレベルアップを目指すものである．

2.1.2 関連するこれまでの研究について

［当該委託研究以前に実施されていた研究責任者以外の方が携わっていた、関連する研究テーマについて］

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，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，必要となる技術等の条件に応じて工夫が必要となり，標準等の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[1]によれば，完全性，トレーサビリティ，一貫性等の品質特性が定義されているものの，現状では，各検証は各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．

要求仕様書はダイアグラムを用いた記述よりも，自然言語で記述されることが一般的である．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[12]．提案されたテンプレートを図 2-1 に示す．Pohl らは本テンプレートに沿った形式で要求仕様を記述するために，変換 Step1～5 からなる記述手順もあわせて提案している．

- Step1 Determine the Legal Obligation
- Step2 The Requirement Core
- Step3 Characterize the Activity of a System
- Step4 Insert Objects
- Step5 Determine Logical and Temporal Conditions

Step1 は，要求の優先度を明確化することを提案している．すなわち，要求の優先度を，

必ず必要，必要，あっても良い，不要といったレベルに仕分けることを推奨している．Step2 は，要求の本質となる内容，すなわち，機能の振る舞いに相当する動詞句を明記することとしている．例えば，「登録を行う」，「削除を行う」等よりも「登録する」，「削除する」といったシステムの役割を明確に記述するべきであるとしている．

Step3 は，Step2 に関連して，システムの活動の範囲を明確に記述する必要があるとしている．すなわち，システムが自律的に実行する動作，ユーザや他システムとのやり取りを通して実現される動作があるが，これらを明確に区別して記述すべきであるとしている．

Step4 では，Step3 の記述内容に対して，1 つ以上の目的語の記載を提案している．要求文に目的語が存在しないと，Step2 で定義した要求のコア，すなわち，システムの振る舞い自体も曖昧になるとしている．

Step5 では，要求が論理的および時間的な条件がある場合，これらを容易に区別するため，文頭に条件を記載すること，としている．例えば，例外的な振る舞いが存在する場合には，「地震発生時には」等の時間的な条件として明示することが重要としている．

Pohl らの提案は，英語で記述された自然言語による機能要求を対象にしている．本研究では，日本語で記述されたシナリオを対象としているため，Pohl らの提案をそのまま適用することは困難である．一つの要求文を構成する語順は英語と日本語では異なるため，テンプレート自体は直接日本語の要求文には適用できないものの，Step1～Step5 の手順が示す内容は重要である．Pohl の提案するテンプレートに適合した要求文は，曖昧さの解消につながると考えられる．Pohl らのテンプレートにおいて，動詞に対する目的語は，データや画面が相当すると考えられるが，それらの用語自体に表記ゆれがあれば，要求文自体の曖昧性も解消されない．本研究は，要求文を構成する設計要素に焦点をあてて，使用する用語の一貫性の検証を支援する点で，Pohl らの提案とは違いがある．

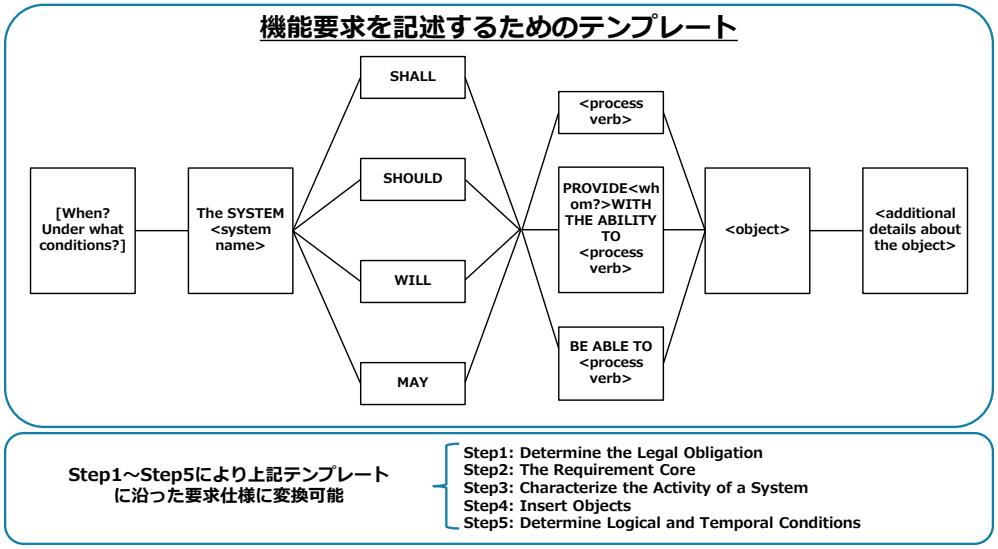


図 2-1　Pohl らによる要求記述のテンプレート

Lucassen らは，アジャイル開発においてユーザ要求の記述に良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 AQUSA と支援ツールの開発および評価を報告している[13]．Lucassen らはユーザストーリーの品質を評価するフレームワークとして，Syntactic，Semantic，Pragmatic の 3 視点からなる 14 個のメトリクスを定義した．また，ユーザストーリーを上記メトリクスで評価し，検証レポートを生成するツールを開発し，実際の案件で評価し，効果を実証した．Lucassen らが開発したツールでは，Semantics の観点からの検証はスコープ外として，Syntactic や Pragmatic の観点からの検証を行っている．

本研究では，「Semantic」における「Unambiguous」の指摘を行うことに主眼を置いており，内容に関わる用語の一貫性を検証することに重きを置いている点で Lucassen らの提案と差別化できる．自然言語で記述された要求仕様を対象として仕様の検証を行う方向性は同一であり，アジャイル開発などの開発スタイルの需要が高まるほど，自然言語による要求仕様の記述が増加し，記述した仕様については，品質を一定に保つことの必要があることを改めて確認できる．

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．河野らは，ドキュメント内において，曖昧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[14]．

久野らは，同じ語句でも，使われ方により曖昧の度合いが異なるとして，曖昧性の高い語句を選択的に検出することを目的に，語句の曖昧性を判定するツールを提案している[15]．久野らが提案するツールでは，曖昧性のレベルを，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真の曖昧語の検出において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．

本研究では，先行研究[14][15]が着目している「曖昧さにつながりやすいキーワード」は，NG ワード検証として指摘している．本研究が NG ワードよりも重要視する検証の対象は，対象となるドキュメント内で定義された設計要素である．当該設計要素を示す用語が，曖昧さにつながりやすいキーワードには該当しなくとも，一つの設計要素が，別の表現で定義されていることや，要求仕様書内で言及されている設計要素が，明確な定義なく機能要求等に記述されていれば，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．本研究では，NG ワードに加えて，仕様書の本質的な定義内容に踏み込んで，設計要素の曖昧さや不整合を指摘する検証を行っている点で，先行研究とは差別化ができている．

［研究責任者が当該委託研究以前に実施していた研究と当該委託研究との関係について］

研究責任者はソフトウェア開発企業において，高品質な要求定義を実施可能とするための手法や支援技術の研究開発に取り組んできた[16～20]．技術シンポジウム SPES2014 の SPES 事例研究（経験報告）［事例研究 SPES2014-S4a］は，要求仕様の検証知識の形式知化とその知識を組み込んだ支援ツールによる一貫性検証の実践事例であり，要求仕様の品質安定化に貢献する成果である．しかし，［事例研究 SPES2014-S4a］における要求仕様の検証のノウハウは，機能一覧表等を対象にアクター定義の一貫性にフォーカスした，東芝ソリューションの分析結果に基づいて定義された内容であり，他の企業での適用可能性は確認されておらず，検証支援ツールも一般公開はされていない．そこで，本研究では，図 2-2 に示すように，［事例研究 SPES2014-S4a］を特定企業向けから産業界全体で共有可能な形式に一般化し，さらに，検証ノウハウを各企業が独自に拡張可能なように，検証ルールを共通と可変部分に分離構造化し，産業界全体で共有する．

形式仕様記述言語や状態遷移図等のダイアグラムにより形式的に記述された仕様は，記述ルールを明確に定義し易く，検証の自動化の研究が活発である．しかしながら，自然言語で記述された「シナリオ」の検証技術は成熟している状況には至っていない．本研究では，自然言語で記述されたシナリオにおいて，製品ソフトウェアを構成する要素である，アクター，データ，画面，振る舞い等に着目し，これらの用語の定義漏れや表記ゆれを指摘することで，要求仕様の一貫性を検証することを支援するため，新規性ならび独創性が高いと考えている．

先行事例［事例研究 SPES2014-S4a］では，特定企業の検証ノウハウをツールに組込み，検証自動化を実現していたが，本研究では，複数企業からのインタビューに基づき仕様検証のノウハウを抽出し，それら検証ノウハウを一般化し，検証ルールと辞書を分離独立化し，関係性を考慮して構造化する．要求仕様の検証ノウハウを，構造化された検証ルールと辞書により組織の資産とすることを可能にする点で，独創的である．

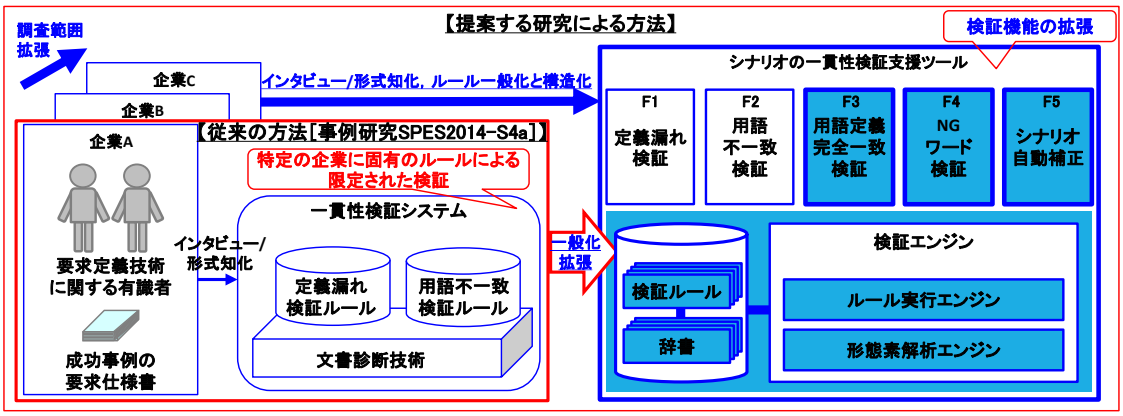


図 2-2　先行事例［事例研究 SPES2014-S4a］と本研究との関係

2.1.3 研究目標と研究課題

(1) 研究目標

本研究では，要求仕様の構成要素であるシナリオをとりあげ，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シナリオの整合性の検証知識を形式知化し，それら知識に基づくシナリオの一貫性検証支援ツールを実現する．開発するシナリオの一貫性検証支援ツールは，シナリオ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，要求仕様書中の記述と整合していることを検証する．本研究では，シナリオの整合性の検証知識とシナリオの一貫性検証支援ツールを提供することで，業界全体で要求定義のノウハウを共有し，我が国における要求定義技術のレベルアップを目指す．そして，魅力あるソフトウェア製品の要求定義の効率化，高品質化により，ソフトウェア開発の国際競争力向上に貢献する．

製品ソフトウェア開発において，上流工程からの品質の作り込みは重要であり，要求定義工程で定義した要求仕様の品質検証が極めて重要である．しかしながら，現状では，このような検証作業は，経験のあるベテラン技術者に依存しており，属人的な方法で実施されている．その結果，要求仕様の検証の品質にばらつきが発生し，効率的かつ統一的な要求定義の検証の実施は困難な状況にある．

検証のしやすさを考慮して，要求仕様書として，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，アジャイル開発ではユースケースシナリオや，ユーザストーリー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取り組みは極めて重要である．

また，2.1.2 で述べたように，自然言語で記述された要求仕様などのドキュメント中に記述された，指示代名詞や条件を表す曖昧語に着目して，自然言語の形態素解析ソフトウェアを用いて，それらの曖昧語を特定する取組みや支援ツールの研究が行われている．従前の研究における仕様の検証の対象は，曖昧な用語や表現の特定と修正が中心であり，仕様の中身に踏み込んだ検証は成熟した状況に至っていない．

本研究では，要求仕様の本質的な内容である機能要求に着目し，システムの利用者または連携システムである「アクター」，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」の設計要素に着目した．そして，本研究では，それら設計要素の定義漏れや表記ゆれなどによる要求仕様の一貫性の検証を行うことで，要求仕様の品質を安定化させることを狙う．なお，設計要素の定義漏れや表記ゆれの特定に関しては，ベテラン技術者の暗黙知となっていると考えられるため，有識者へのインタビュー等により，そうした検証知識を抽出し，形式知化する．形式知化した検証知識を検証支援ツールに組込み，検証を自動化することで，初級の技術者でも，効率的かつ妥当な要求仕様検証を行うことを目指す．形式知化された検証知識を業界全体で共有することにより，産業界における要求定義の検証技術のレベルアップを実現する．

(2) 研究目標に向けた研究課題の設定

産業界全体における要求定義の品質安定化とスキルレベルの向上を達成するために，次のアプローチにより研究を進める．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も適用し評価する．

・シナリオの一貫性検証知識の形式知化（形式知化）
・シナリオの一貫性検証の支援ツールの開発（ツール開発）
・シナリオの一貫性検証の支援ツールの評価（評価）

上記アプローチの元，本研究では次の 3 つの研究課題を設定する．

◆研究課題 1：シナリオの一貫性検証知識の形式知化（形式知化）

先行事例［事例研究 SPES2014-S4a］では，特定企業かつ，機能一覧表等の自然言語で記述された仕様説明内における「アクター」を対象として，「用語不一致検証ルール」と「定義漏れ検証ルール」に基づく 2 種類の検証の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，研究成果を広く複数企業へ展開するため，検証知識の抽出範囲を複数企業に拡張する．また，検証の対象をシナリオに一般化した上で，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，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，検証ルールおよび辞書として形式知化する．

◆研究課題 2：シナリオの一貫性検証の支援ツールの開発（ツール開発）

研究課題 1 で定義した検証ルールおよび辞書を組込んだ，シナリオの一貫性検証を支援するツールを開発する．先行研究［事例研究 SPES2014-S4a］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，検証ルールならびに辞書の共通部分を共有するとともに，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．さらに，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．なお，開発した検証ルール，辞書，検証エンジンを公開することを計画している．

本来，要求仕様の高品質化においては，システムの持つ働きである機能要求，定義した機能の操作性や保守性などの非機能要求の妥当性を追求すべきである．開発する機能に付加価値を付与することや，顧客の真の要求に対応するために非機能的側面の能力を向上させることは，発注者と開発者らを含むステークホルダ間でリソースを投入して追求すべき事柄である．しかし，合意形成に用いる要求仕様書の定義内容に矛盾や不足があれば，定義内容の些末な確認や表記ゆれの指摘にために確認作業や手戻りが発生するリスクがある．このような確認作業や手戻りの発生は，ステークホルダ間の本質的な議論と合意形成の妨げになる．本研究では，ツールにより機械的に指摘と修正が可能な部分はツールにまかせ，要求仕様の機能や非機能の要求の本質的な中身の充実に十分なリソースを集中させることを狙い，本ツールを開発する．

図 2-3 に提案するツールによる検証の一例を示す．図 2-3 において，ツールは，シナリオとアクター定義の仕様を入力として検証レポートを出力し，シナリオ中に表記ゆれがあ

ることを指摘する．本検証例では，「アクター定義に定義されていないアクターが，シナリオに記述されていると，例えばアクセスコントロールの仕様の定義漏れや手戻り確認等の発生リスクが高まるため，事前に一貫性不備を解消すべきである」というベテラン技術者が仕様の検証の際に用いるノウハウが用いられている．

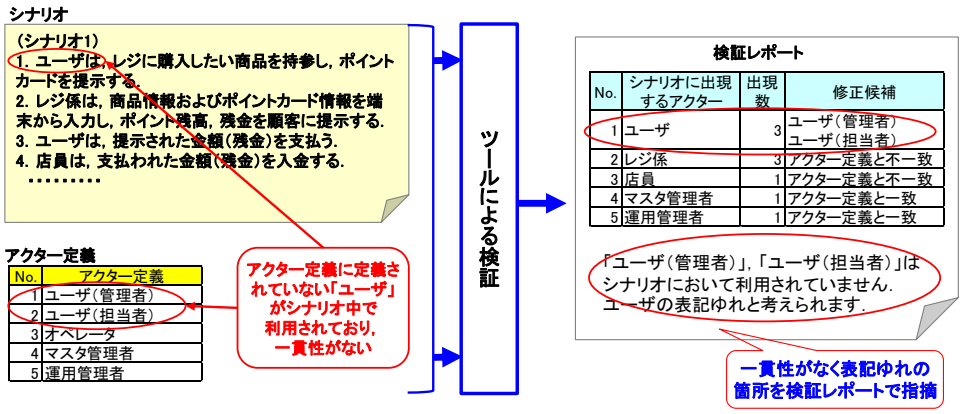


図 2-3　シナリオ検証の事例

◆研究課題 3：シナリオの一貫性検証の支援ツールの評価（評価）

研究課題 1 で開発した検証知識を組み込み，研究課題 2 で開発したツールを，研究課題 1 でインタビューした有識者およ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 3 社程度の企業に対して，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．ツールを用いた要求仕様の検証方法の新たなノウハウを抽出し，ツールの利用マニュアル等に追記し改善を行う．改善後，シナリオの一貫性検証支援ツール，マニュアル等関連ドキュメントをリリース成果物としてまとめる．

2.2 研究の活動実績・経緯

設定した 3 つの研究課題を順次解決する方法で研究を実施した．研究活動予定に対する実績を示した進捗予定管理表を表 2-1 に示す．研究課題 1「シナリオの一貫性検証知識の形式知化（形式知化)」では，①研究技術動向調査分析，②有識者インタビューを行い，これらの結果を反映させて検証ルールならびに辞書を定義した．

有識者インタビューは 4 社 5 部門に対して行った．研究技術動向調査分析においては，要求工学の国際会議 RE2015 に研究責任者が参加し，仕様検証の研究開発およびツールを中心に調査した．

研究課題 2「シナリオの一貫性検証の支援ツールの開発（ツール開発)」では，開発範囲を反復 1 および反復 2 に分割して，ツールを試作した．試作は外注に発注した．反復 1 では，検証エンジンの構築ならびに，定義漏れ検証と用語不一致検証の 2 つの機能を実装することとし，要求定義および基本設計を大学側で実施し，プログラム試作は外注に発注した．出来上がった試作プログラムに対して，大学側で受け入れテスト仕様書を準備し，リアルな仕様書に対して，人手で検証した結果および検証レポートの正解案を作成し，試作したツールで検証した結果と突き合わせることで，プログラムのテストを行った．ここで得られた検証ルールや設計に関する気づきを整理し，反復 2 の開発において改善項目として対応することとした．

反復 2 では，上述した改善項目の対応ならびに，用語定義完全一致検証，NG ワード検証，シナリオ自動補正の機能を開発した．プログラム試作は反復 1 と同様に外注発注した．反復 2 においては，反復 1 同様にリアルな仕様書を用いた検証を行った．また，検証ルールの内容についてはルール定義書にフィードバックするとともに，検証結果の解釈の方法や検証レポートの使い方などのノウハウをユーザマニュアルにまとめた．

研究課題 3「シナリオの一貫性検証の支援ツールの評価（評価)」においては，ツールの開発状況を有識者にインタビューし，ツールによる検証の妥当性，有効性，操作性，適用可能性などについて意見を伺った．有識者インタビューは，4 社 5 部門に対して行った．インタビューによって得られたコメント等をとりまとめ，適用可能性については，使われ方のユースケースをユーザマニュアルに反映させた．

表 2-1　進捗予定管理表（実績）

作業項目		6月	7月	8月	9月	10月	11月	12月	1月	2月	進捗状況
1. 研究課題	予										
	実										
1.1 研究課題1「形式知化」	予										
	実										
1.1.1 研究技術動向調査分析	予	→		→	→						
	実	→		→	→						完了
1.1.2 有識者インタビュー：インタビュー準備	予	→									
	実	→									完了
1.1.3 有識者インタビュー実施（1）	予		→								
	実		→								完了
1.1.4 有識者インタビュー実施（2）	予		→	→							
	実		→	→							完了
1.1.5 有識者インタビュー実施（3）	予			→	→						
	実			→	→						完了
1.1.6 検証ルールの定義	予				→	→					
	実				→	→					完了
1.2 研究課題2「ツール開発」	予										
	実										
1.2.1 反復1：要求定義・設計	予	→	→								
	実	→	→	→							完了
1.2.2 反復1：開発・テスト	予		→	→	→						
	実		→	→	→	→					完了
1.2.3 反復2：要求定義・設計	予			→	→						
	実			→	→	→					完了
1.2.4 反復2：開発・テスト	予					→	→				
	実					→	→	→			完了
1.2.5 反復1と2の統合	予						→	→			
	実						→	→			完了
1.2.6 まとめ・マニュアル作成	予						→	→			
	実						→	→	→		完了
1.3 研究課題3「評価」	予										
	実										
1.3.1 評価項目定義	予			→	→	→					
	実			→	→	→					完了
1.3.2 有識者評価（1）	予						→				
	実						→	→			完了
1.3.3 有識者評価（2）	予						→	→			
	実						→	→	→		完了
1.3.4 有識者評価（3）	予							→	→		
	実							→	→	→	完了
1.3.5 改善項目整理	予							→	→		
	実							→	→	→	完了
1.3.6 改善と成果まとめ	予								→	→	
	実								→	→	完了
2. 外注	予										
	実										
2.1 仕様作成・確認	予	→	→	→	→						
	実	→	→	→	→						完了
2.2 外注実施①ツール試作：反復1の機能試作	予		→	→	→						
	実			→	→	→					完了
2.3 外注実施②ツール試作：反復2の機能試作	予					→	→				
	実					→	→	→			完了
3. 中間報告	予										
	実										
3.1 中間報告：準備	予				→	→					
	実				→	→					完了
3.2 中間報告	予					△中間報告					
	実					▲中間報告					完了
4. 最終成果のとりまとめ	予										
	実										
4.1 成果報告書の構成案	予						→	→			
	実							→	→		完了
4.2 最終報告準備	予							→	→		
	実							→	→		完了
4.3 最終報告	予								△最終報告		
	実								▲最終報告		完了
4.4 成果報告書の作成	予								→		
	実								→		完了
5. 成果物の提出	予								成果報告書△		
	実								成果報告書▲		完了

2.3 研究実施体制

研究は次の体制で実施した.

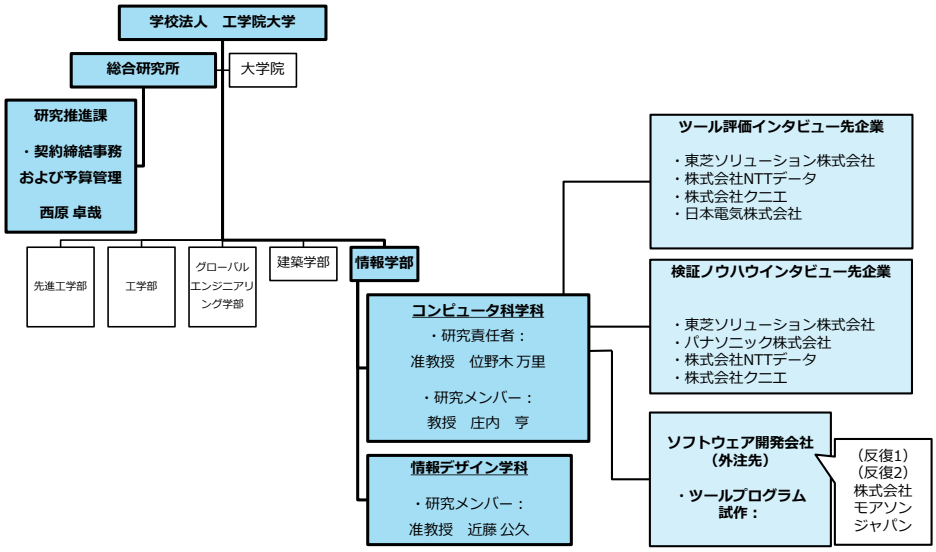


図 2-4　研究の実施体制

研究責任者のプロフィールと研究者の役割分担は次の通りである.

[研究者プロフィール]

(ふりがな) 氏　名	い　の　き　ま　り 位野木　万里
所属機関	学校法人工学院大学
所属	情報学部　コンピュータ科学科
役　職	准教授
住　所	〒163-8677　　東京都新宿区西新宿 1-24-2
TEL	03-3340-2889
E-mail	m_inoki@cc.kogakuin.ac.jp

【学歴（大学卒業以降）】 ・1989 年 3 月 15 日 早稲田大学理工学部数学科卒業　理学士 ・1991 年 3 月 15 日 早稲田大学大学院理工学研究科 数学専攻修士課程修了　理学修士 ・2008 年 3 月 15 日 早稲田大学大学院理工学研究科 情報・ネットワーク博士課程 修了 博士（工学）	【職歴】 ・1991 年 4 月 1 日　(株)東芝　社員 ・2003 年 10 月 1 日東芝ソリューション(株) SI 技術開発センター　主任 ・2013 年 4 月 1 日東芝ソリューション(株) 生産技術センター主幹 ・2014 年 4 月 1 日～現在： 工学院大学　情報学部　コンピュータ科学科 准教授

[研究チームの役割]

・　位野木　万里　　研究責任者　（准教授）

研究責任者として，契約締結，予算管理，外注管理，成果物提出等の契約実施の全体を統括した．また，企業インタビューと検証知識の抽出，検証手法およびツール仕様の考案，開発，検証を担当した．

・　庄内　亨　　研究メンバー（教授）（2015 年 12 月 31 日まで）

企業インタビューと検証知識の抽出及び検証ノウハウの形式知化を担当した．

・　近藤　公久　研究メンバー（准教授）

企業インタビューと検証知識の抽出を担当するとともに，ツール（検証エンジン）仕様化および検証を担当した．

・　学生アルバイト

工学院大学情報学部コンピュータ科学科の 4 年生を 7 名採用した．研究の作業状況に応じて，研究スタート時の 2015 年 6 月は 4 名，2015 年 8 月から 3 名を追加した．学生アルバイトは，技術調査，有識者インタビューでの議事録作成，デモ担当，設計書等のドキュメント作成支援，ツールのテスト仕様書ならびに実仕様書の分析と検証レポートのテストデータ作成，ツールのテスト実施とテスト成績書の作成まとめを担当した．

・　株式会社モアソンジャパン

シナリオの一貫性検証支援ツールの（反復 1）および（反復 2）の開発の試作を担当した．

・　検証ノウハウインタビュー先企業

要求仕様などの検証ノウハウの抽出のため 4 社 5 部門にインタビューを行った．4 社とは，インタビューした日程順に，東芝ソリューション株式会社，パナソニック株式会社，株式会社 NTT データ，株式会社クニエである．

・　ツール評価インタビュー先企業

開発したシナリオの一貫性検証支援ツールの評価のために 4 社 5 部門にインタビューを行った．4 社とは，インタビューした日程順に，東芝ソリューション株式会社，日本電気株式会社，株式会社 NTT データ，株式会社クニエである．

3 研究成果

3.1 研究課題 1「シナリオの一貫性検証知識の形式知化（形式知化）」

3.1.1 当初の想定

(1) 研究内容

先行事例［事例研究 SPES2014-S4a］において，仕様の検証の範囲は，特定企業かつ，機能一覧表等の自然言語で記述された仕様説明内における「アクター」を対象として，「用語不一致検証ルール」と「定義漏れ検証ルール」に基づく 2 種類の検証の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，検証知識の抽出範囲を，複数企業に拡張する．また，検証の対象をシナリオに一般化した上で，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，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，検証ルールおよび辞書として形式知化する．

(2) 想定問題と対応策

［想定問題］

先行研究［事例研究 SPES2014-S4a］では，検証知識は，一社内かつアクターの用語不一致と定義漏れに限定された内容であった．このような限定された検証知識では，他の組織におけるシナリオの一貫性検証への適用が困難である．

［対応策］

アクターの用語不一致と定義漏れ以外のシナリオの構成要素に検証の範囲を拡張することおよび，他の組織での適用を実現するためことが重要である．そのために，次のように解決策を考案する．

- 関連文献並びに国際会議での研究動向調査分析を通して，検証の観点を，アクターに加え，データ，画面，振る舞いを追加し拡張する．
- 複数社の有識者インタビューを通して検証のための知識を抽出し，共通部分と可変部分の知識に仕分けた上で，検証ルールおよび辞書として形式知化する．図 3-1-1 に検証ルールの構造化案を示す．本案は，各検証知識を部品化し，組織毎に容易に拡張やカスタマイズが可能なように定義した．
- 用語不一致検証，定義漏れ検証に加えて，用語完全一致検証と NG ワード検証を追加する．図 3-1-2 に「用語完全一致検証」の処理フローを示す．この検証は，「シナリオ中に出現したアクター用語はアクター定義仕様に定義されており，アクター定義仕様に定義された全てのアクターが，シナリオ中に少なくとも 1 回は出現する」ことを確認する．
- 有識者へのインタビューとしては，事業部門での要求定義の業務に関わった経験を持つ技術者または研究者に対するインタビューを行う．
- 検証知識の共通化と可変部分をスムーズに特定するため，インタビュー内容を次の観点で統一化する．

- あらかじめ「ポイント管理システム」や「道案内サービス」等の具体事例を想定し，改善前のシナリオと改善後のシナリオ例を用意する．改善前のシナリオを提示し，対象のベテラン技術者（インタビュー先）が検証する際のポイントを問う．また，改善後のシナリオを提示して，改善に用いた知識が妥当であり共通化すべきかどうか等を確認する．

・ インタビュー先のベテラン技術者の独自の視点からのシナリオ検証のポイントを自由に述べていただき，検証ノウハウを整理する．

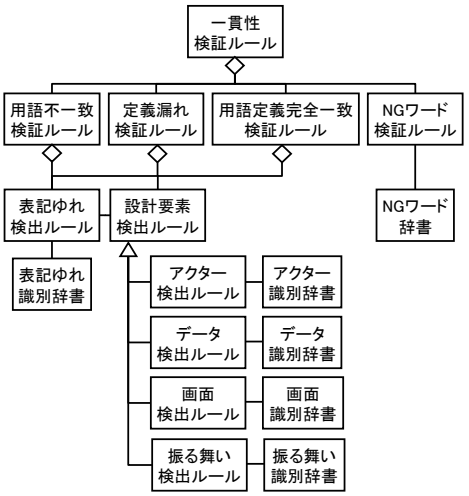


図 3-1-1　検証ルールと辞書の構造化

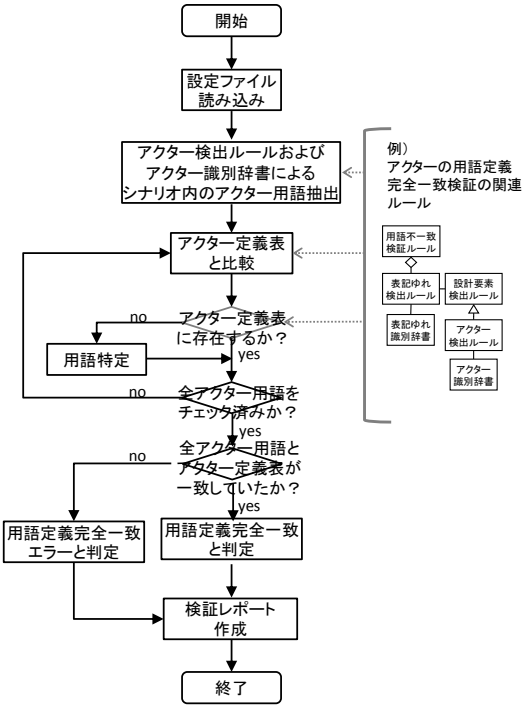


図 3-1-2　検証ルールを用いた「用語完全一致検証」処理の流れ

3.1.2 研究プロセスと成果

(1) 研究プロセス

研究は次の①～③の手順で実施した.

①研究技術動向調査分析

3.1.1(2)「対応策」で示した，図 3-1-1 の検証ルールと辞書の構造化の考え方に沿って，関連文献ならびに国際会議での研究調査を行い，検証ルールの具体化と辞書の構造化に関する方向性を明らかにした.

②有識者インタビュー

3.1.1(2)「対応策」で示した有識者インタビューの方法に基づき，インタビュー項目の準備，事業部門での要求定義の業務に関わった経験を持つ技術者または研究者に対するインタビューを行った．インタビュー項目としては，本研究で想定している用語不一致（表記ゆれ）の検証の有用性や，検証の種類の妥当性，有識者の方々が実際に仕様書を検証する際に気を付ける留意点などである.

③検証ルールの定義

①②の分析結果に基づき，図 3-1-1 の検証ルールと辞書の構造化の考え方に沿って，検証ルールを定義した.

(2) 具体的な研究成果の内容

①研究技術動向調査分析

1)要求仕様化ならびに検証に関する研究動向の調査結果

関連文献から，ユーザ中心設計等における要求仕様化の動向，要求工学に関する国際会議 RE2015 における要求仕様化および検証に関する研究動向についてまとめた結果を示す.

[ユーザ中心設計等における要求仕様化の動向]

真の顧客要求に応えるためには上流工程からの品質の作り込みが必至であり，対象ソフトウェアの機能や非機能の範囲を定める要求定義工程が重要である[1,4,21]．現状，発注者要求は多様化しており，業務システム導入による事務業務の品質向上，コスト削減，リードタイム短縮への貢献だけでは，発注者が満足しないリスクが考えられる．発注者は，競合他社と異なる特性を持つことの差別化や，システムの利用による心地良さや楽しさをもたらす刺激を求める傾向にある．したがって，要求獲得，要求記述，要求検証，要求管理からなる一連の要求工学の技術としても，多様化する発注者要求を想定した内容にすることが重要である．多様化する発注者要求を明らかににし，それらを具現化するために，ユーザ中心設計[9,22]やアジャイル開発[5,6,23]などが試みられている.

ISO 9241-210:2010 によればユーザエクスペリエンス(以下，UX と略す)とは，製品，システムまたはサービスを使用した時，および／または使用を予測した時に生じる個人の知覚や反応のことである[24]．UX を考慮したユーザ中心設計では，ユーザ側が多様化していることを考慮する必要がある[9,22]．考慮する事柄としては，文化や言語，子どもや高齢者なども含めた利用者の世代，ベビーカーや所有物がある，車いすで移動している，その他何ら

かの障がいがあるなどの条件も考慮する必要がある．加えて，同一のユーザが様々な状況下で感じ方やそれに基づく行動が異なることを考慮することも必要である．その状況とは，季節，時間帯，気候，その時の服装や持ち物，その時の健康状態や気持ちなどである．

UX を考慮した要求獲得のための手法として Goal Directed Task Analysis（以下，GDTA と略取）がある[9,25]．GDTA は，Endsley により考案されたユーザの状況認知（SA: Situation Awareness）に着目した要求獲得手法である．SA とは，ある環境下での現状認識，現状理解，それによる変化予測からなる一連の認知活動のことをさす．GDTA では，人は，「現状認識→現状理解→未来予測」のサイクルに従い，次の行動の意思決定をするため，そのような性質を考慮した要求獲得をする必要があるとしている．

UX で重視されるのは，多様化するステークホルダ間での要求の検証と合意形成である．関係するステークホルダを洗い出し，製品ソフトウェアの使われ方を明らかにし，ステークホルダ間で理解を共有するためにシナリオが有効である．UX やアジャイルな方法による要求獲得が推進されるに伴い，要求をシナリオとして，自然言語で記述するシーンが増加すると考えられる．したがって，シナリオを用いたステークホルダ間の合意形成において，シナリオの品質を向上させておくことは極めて重要である．

［要求工学に関する国際会議 RE2015 における要求仕様化および検証に関する研究動向］

2015 年 8 月 24 日～28 日まで，オタワ大学（カナダ・オタワ）において，要求工学に関する国際会議 IEEE 23rd International Requirements Engineering Conference（以下，RE2015 と略す）が開催された．要求獲得や要求定義段階において，獲得した要求が曖昧にならぬようにするにはどうすべきか，というのは，要求工学分野において継続的に取りあげられているテーマであり，RE2015 の Demos and Posters Session でも統一したテーマとして着目されていた．

自然言語で記述された要求仕様には曖昧さや矛盾が混入されるリスクが高いため，それを防止するためのツールや手法の開発は重要である．曖昧さや矛盾が混入する余地のない形式的な言語やダイアグラムの利用は，解決策の一つではあるが，上流段階では，達成したいゴール，直面する問題，解決の範囲そのものが曖昧なため，ノウハウの再利用が重視されている．例えば，[26]～[29]では，要求をパターン化しておく Requirements Pattern の活用が各所で取り組まれている．

[27]の ReqPat は，機能名や入出力情報などの情報を記述する際に，要求仕様中にタグを埋め込み，そのタグを用いて，要求仕様のパターンの候補を提示し，そこから要素を選択することで，記述の自由度を制約して，形式的な仕様に近づける工夫をしている．パターンをあらかじめどの程度定義できるのか，パターンを選択するツールの操作性などが課題となる．

[27]のようなアプローチも要求仕様を定義する側に仕様の記述方法を規定することになる．要求の企画や検討段階では，IT を専門とはしないユーザ側が要求仕様を記述することを考えると，仕様の記述方法を規定することは現実的ではなく，自由度の高い自然言語による仕様記述が妥当である．したがって，シナリオ等で仕様を記述した後，本研究による要求仕様の検証を行うことは有効と考えられる．

Mahmoud は，テキストで記述された要求仕様書から非機能要求を抽出するための手法を提

案した[30]．これは，要求仕様書から抽出した用語の意味を特定するために Normalized Google Distance（以下，NGD と略取）[31]を用いている．すなわち，各非機能要求に関連する用語が，NGD の観点で同じ文脈で使われていると判断されるとき，当該用語をその非機能要求に関連する用語であると定義する．本研究では，表記ゆれ辞書を用いて，異音同義語を洗い出している．このために，プロジェクトや組織として合意された異音同義語を，あらかじめ定義する必要がある．技術用語や概念は変化することが予測される．表記ゆれ辞書の維持管理過程において，NGD を用いた用語のメンテナンス支援は有効であると考えられる．

Lucassen らは，アジャイル開発においてユーザ要求の記述に良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 AQUSA と支援ツールの開発および評価を報告している[13]．ユーザストーリーとは，“As a <type of user>, I want <goal>, [so that <some reason>]”で記述される要求であり，アジャイル開発において獲得した要求を記述する方法として利用されている．Lucassen らはユーザストーリーの品質を評価するフレームワークとして，Syntactic，Semantic，Pragmatic の 3 視点からなる 14 個のメトリクスを定義した．また，ユーザストーリーを上記メトリクスで評価し，検証レポートを生成するツールを開発し，実案件で評価し，効果を実証した．図 3-1-3 に Lucassen らが開発したユーザストーリーの検証ツールの概念図を示す．なお，Lucassen らが開発したツールでは，Semantics の観点からの検証はスコープ外として Syntactic や Pragmatic の観点からの検証を行っている．

Lucassen らの研究では，入力する要求仕様のフォーマットが限定するが，Syntactic や Pragmatic の観点から品質評価を行っている．ユーザストーリーは，要求の洗い出しの出発点であり，厳密性にフォーマットを意識して記述していないことも多いため，どの段階でユーザストーリーを検証するのかは検討の余地がある．

本研究では，「Semantic」における「Unambiguous」の指摘を行うことに主眼を置いており，内容に関わる用語の一貫性を検証することに重きを置いている点で Lucassen らの提案と差別化できる．自然言語で記述された要求仕様を対象として仕様の検証を行う方向性は同一であり，アジャイル開発などの開発スタイルの需要が高まるほど，自然言語による要求仕様の記述が増加し，記述した仕様については，品質を一定に保つことの必要があることを改めて確認できる．

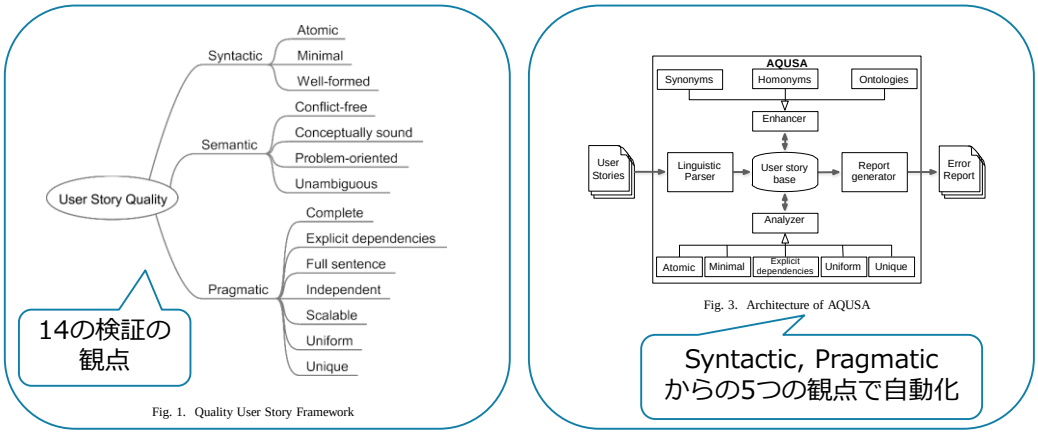


図 3-1-3 Lucassen らによるユーザストーリーの高品質化ツール

Ferrari らは，インタビューによる要求抽出における曖昧さ「Ambiguity」の役割について分析し,Ambiguity のタイプをフレームワークとして定義した[32].Ferrari らによれば，Ambiguity は，要求獲得の障害になると考えがちであるが，むしろ，暗黙知の存在を気づかせるための resource（源）であると主張している．

本研究では，仕様書における曖昧さの排除を狙っているため，曖昧さに着目した文献[32]の研究は関係性が高い．ソフトウェアの利用者は多様化し，開発技術が成熟しつつあるものの，要求の曖昧さをどのようにとらえ，真の顧客要求に対応すべきか，ということは追求すべき価値あるテーマといえる．

②有識者インタビュー

3.1.1(2)「対応策」で示した有識者インタビューの方法に基づき，本研究で想定している用語不一致（表記ゆれ）の検証の有用性や，検証の種類の妥当性，有識者の方々が実際に仕様書を検証する際に気を付ける留意点などのインタビュー項目を定義した．インタビューは，東芝ソリューション株式会社では 2 部門，パナソニック株式会社，株式会社 NTT データ，株式会社クニエはそれぞれ 1 部門のソリューション／システム開発の要求定義の経験豊富な有識者に対して実施した．得られた主なコメントを以下にまとめる．

[定義漏れ/用語不一致検証の有用性]

- 定義されていない設計要素名および表記ゆれの指摘は有効である．発注者の考える「ユーザ」と開発者の考える「ユーザ」は意味が異なることがある．このような事柄をツールで指摘し検証のための気づきを与えることは重要である．
- 表記ゆれをしている概念は，専門家であれば意味を補って解釈することは可能であるが，その場合の「あたりまえ」や「思い込み」によってすり抜けた結果が，その後のシステム開発での矛盾や不具合につながるリスクもある．本ツールにより「あたりまえ」や「思い込み」のリスクを指摘できる可能性がある．
- 表記ゆれの事例として例えば以下が考えられる
  - 「ユーザ」,「ユーザー」,「利用者」が混在
  - 「ウェブサイト管理者」が一般化されて「管理者」となり「コンテンツ管理者」と混同
  - 「センサー」と「センサ」が混在
  - 登録，蓄積，保存が同一の意味で混在して利用
  - 更新，改善，修正が同一の意味で混在して利用
  - チーム　→　課，　グループ　→　部，など組織の変更が反映されず旧組織名を使用

[要求仕様書の検証ノウハウ]

- NG ワードとして,「ある条件下では」,「・・・の時には」があげられる．ある状況を限定して仕様説明をする場合，例えば「ある条件下では・・・」と記述するのみにとどまると，その条件以外の場合が明確ではないことになり，仕様が曖昧になる原因になる．

- 「既設システムと同様である」,「全て」,「約」,「だいたい」,「最新版」，二重否定（～しないわけではない）等も NG ワードとして指摘する必要がある.
- 一文一文の不整合より全体としての網羅性や整合性を見ることも重要である.
- コストに直接反映される要求に細心の注意を払う．例えば，ブラウザの種類，サーバの種類，対応デバイスの数，準拠すべき標準や法令などである．とくに「最新のブラウザに対応すること」などは注意が必要である.
- 要求と根拠の対応づけの確認も重要である．要求仕様書には，システム化の目的とその実現手段を明記することが基本である．これらの網羅性と記述の妥当性の検証が重要である．よって，振る舞い用語の「目指す」を中心とした，要求仕様の記述の検証は必要である.
- 検証する際のポイントとしては，システムを導入することにより実現したいことが明記されていることの確認である．システムを導入することによる Tobe の姿，市民にとって何がもたらされるのかという観点の記述網羅性が重要である.
- 一見，一般的と思われる業務仕様に留意する必要がある．例えば，発注者側の規模が小さいほど，開発者側との業務の認識のズレが発生するリスクが高まる.「締め処理」など普通に存在する処理の，対象，期間，訂正方法等は，発注者側と開発者側が考えがちな「一般的な概念」が通用しないケースがあるので留意が必要である.
- スマート端末や IoT などの利用形態の進化に留意することも重要である．イントラネットのエンタープライズ系システム開発に関して，これまで蓄積されてきた常識と，スマート端末を用いた，コンシューマー向けのシステム開発では，従来の常識が適用できなくなるケースがある．例外処理などが発生しうるリスクも変化しており，その対策をあらかじめ講じておく必要がある.
- 非機能要件の事前の洗い出しが重要である．IT ベンダーとして開発者として期待されている，備えておくべき知見が，幅広くなっているとともに，変化している．こうした予期せぬ事態への対策については，開発者側から非機能要件として，発注者にあらかじめ問い合わせ，要件を洗い出しておく必要がある.

[検証時の留意点やツールへの期待]

- 表記ゆれや定義漏れに加えて，記述していなければならない用語や概念の指摘ができると良い．表記ゆれのパターンとしては，長音の有無，全角／半角，同義語（更新／改善／修正）．これらの用語を学習し，辞書が蓄積されると良い．何度も出現する用語を統計的に蓄積しておくことが重要である.
- NG ワードの逆があると良い.CPU の性能とピークとか，あるべき用語のセット（集合）が指摘されると望ましい.
- 仕様書の文章中に記述された，開発量，サイズ，性能等，開発規模やスコープ，すなわち開発費用に影響を与えるファクターに着目して，数値やデータのバリエーションを抽出し，リスト化して示すと，ステークホルダ（例えば，営業担当者）が確認するための情報として有効である.
- ツール利用を通して，単なる単語の指摘にとどまらず，単語間の関係性や，記述はされていないが前提としている暗黙知との不整合を確認するきっかけを得ることや，フ

ェーズ間の一貫性がトレース可能になることを期待する.

- 検証対象として名詞（アクター，データ，画面）と動詞（振る舞い）をおのおの単独で検証しているようだが，アクターとデータ，アクターと振る舞いとデータとの関係性に矛盾や定義漏れが発生することが多いため，組み合わせによる指摘も重要である.
- 要求仕様中から，画面候補やデータ候補が自動抽出されると，基本設計結果と突き合わせることで，開発過程の履歴が蓄積されるので有用である．さらにこれらの情報が機械的に自動生成されると有用性が高まる.

③検証ルールの定義

1)検証ルール

①②の結果を踏まえて，要求仕様の一貫性の検証ノウハウをルールとして定義した．検証ルールは，定義漏れ，用語不一致検証，用語定義完全一致検証，NG ワード検証のルール，形態素解析エンジンによる分かち書きの結果から設計要素（アクター，画面，データ，振る舞い用語）を識別するためのルール，形態素解析エンジン MeCab の特性や条件を踏まえて，名詞句や動詞句を適切に抽出するためのルールを定義した.

ルールの分類と定義数を表 3-1-1 に示す．各ルールの詳細は表 3-1-2～表 3-1-4 に示す．例えば表 3-1-3 の AR005 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，それらを連結したアクターとしてみなす，というルールである．これにより，シナリオ中に，記述されたアクター，ユーザ，ユーザ（会員），ユーザ（非会員）などを特定し，表記ゆれ候補として指摘することができる．なお，これらの表には定義したルールの実装対応先クラス名も示す.

表 3-1-1 検証ルールの分類と定義数

No.	ルール分類	ルール分類説明	ルール数
1	MeCab 結果変換	形態素解析エンジン MeCab の特性を踏まえて，MeCab 出力結果を変換するためのルール．これにより，仕様書中の複合語を適切に抽出する.	14
2	アクター識別	仕様書からアクター用語を識別するためのルール	5
3	データ識別	仕様書からデータ用語を識別するためのルール	5
4	画面識別	仕様書から画面用語を識別するためのルール	5
5	振る舞い識別	仕様書から振る舞い用語を識別するためのルール	5
6	定義漏れ検証	No.1～No.5 のルール分類で定義したルールを踏まえて，仕様書内の定義漏れを指摘するためのルール	1
7	用語不一致検証	No.1～No.5 のルール分類で定義したルールを踏まえて，仕様書内の用語不一致(表記ゆれ)を指摘するためのルール	3
8	用語完全一致検証	No.1～No.5 のルール分類で定義したルールを踏まえて，仕様書内の用語完全一致ではない箇所を指摘するためのルール	1
9	NG ワード検証	No.1～No.5 のルール分類で定義したルールを踏まえて，仕様書中の NG ワードを指摘するためのルール	1
10	その他	上記分類以外で，設計要素の識別や検証のためのルール	2

表 3-1-2　検証ルール（1/3）

項番	分類	ルールID	ルール内容	例	対応するクラス／メソッド名
1	MeCab結果変換ルール	BR001	基本的にはMeCabによって出力された解析結果をそのままセットする．ただし，BR002～BR014の条件にマッチする場合は文字列を連結した形で保持する．	_	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execute
2	MeCab結果変換ルール	BR002	対象となる単語が"動詞"かつ1つ前の単語が"名詞[サ変接続]"の場合，単語を連結させる．さらに前が名詞で連続している場合は，その全てを連結の対象とする．	注文，できる　→「注文できる」 考慮，する　→「考慮する」	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr002
3	MeCab結果変換ルール	BR003	対象となる単語が"名詞[接尾]"かつ1つ前の単語が"名詞"の場合，単語を連結させる．さらに前が名詞で連続している場合は，その全てを連結の対象とする．	要求，仕様，書　→「要求仕様書」として名詞句として変換 #「書」は接尾	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr003
4	MeCab結果変換ルール	BR004	対象となる単語が"名詞[サ変接続]"かつ1つ前の単語が"名詞"の場合，単語を連結させる．さらに前が名詞で連続している場合は，その全てを連結の対象とする．	基本，情報，登録　→「基本情報登録」として，動詞として変換	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr004
5	MeCab結果変換ルール	BR005	名詞[数]＋記号[一般]＋名詞[数]の組み合わせは，単語を連結させる．名詞[数]と記号[一般]が連続している場合は，その全てを連結対象とする．	1，","，000，人　→「1,000人」	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr005
6	MeCab結果変換ルール	BR006	対象となる単語が"助動詞"かつ1つ前の単語が"動詞"の場合，単語を連結させる．	し，ない　→　しない ログイン，し，ない　→「ログインしない」 #「し」は動詞，「ない」は助動詞	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr006
7	MeCab結果変換ルール	BR007	対象となる単語が"名詞"かつ1つ前の単語が"接頭詞[名詞接続]"の場合，単語を連結させる．	本，仕様，書　→「本仕様書」 各，仕様，書　→「各仕様書」 各，社　→「各社」 全，利用者　→「全利用者」 再，発行，する　→「再発行する」	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr007
8	MeCab結果変換ルール	BR008	対象となる単語が"動詞[接尾]"かつ1つ前の単語が"動詞"の場合，単語を連結させる．	考え，られる　→「考えられる」として動詞句 #「られる」が動詞の接尾	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr008
9	MeCab結果変換ルール	BR009	対象となる単語が"動詞(原形が「する」)"かつ1つ前の単語が"名詞"の場合，単語を連結させる．	ログイン，する　→「ログインする」 安定，し　→「安定し」	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr009
10	MeCab結果変換ルール	BR010	【括弧の扱い】 ・名詞＋"（"＋"文字列"＋"）"の表現は，全体で一語として拾う． ・拾う括弧：全角（），半角()，半角全角()，全角半角（） ・括弧の中の括弧，括弧の後の括弧も，名詞（文字列）が名詞とみなされるため，再帰的に本ルールを適用すれば，全体で一語として拾う．	・国（都道府県）は，「国（都道府県）」を一語としてみなす ・ユーザ(オペレータ)，ユーザ(担当者) ・注文(明細) ・「給付台帳（仮）（現行の保育所運営費支弁台帳（総括表）」は，一語（名詞）としてみなす．	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr010
11	MeCab結果変換ルール	BR011	【「〇〇等」の扱い】 ・名詞＋"等"or"毎"　の表現は，全体として一語として扱う． ・動詞＋"等"の表現は，全体として一語として扱う． ・"各"or"本"＋名詞　の表現は，全体として一語として扱う．	・各社，学協会等，部門毎　を一語としてみなす ・本ツールは要求仕様書が一貫性等を保持していることを検証する等を支援する． 「一貫性等」，「検証する等」を一語として拾う．	BR003，BR007，BR008に相当
12	MeCab結果変換ルール	BR014	【つなぎ文字　の，・，／，/　扱い】 ・名詞＋"つなぎ文字"＋名詞　の表現は全体として一語として扱う． ・つなぎ文字が複数回かつ混在して出現していれば，その前後も一語として拾う． ・つなぎ文字は，"env.xml"にて定義された文字とする．	以下「」内の文字列は一語としてみなす： 「品質特性の一貫性の検証」 「各都道府県の住民の個人情報（秘密）」 「交付金申請・決定情報」 「入力・確認・確定情報」 「登録／検索/修正／削除」	クラス名：ExecuteMorphemeAnalysisMecab メソッド名：execBr014
13	アクターを識別	AR001	入力シナリオ形態素DataSetの単語を連結する事で，アクター識別辞書の単語と一致する文字列が存在した場合，その文字列をアクターとみなす．	アクター識別辞書： Actor,26,知識を有しない者 入力シナリオ形態素DataSet： 専門， を， 知識， 有しない， 者 アクター識別： 知識を有しない者	クラス名：ActorRules01 メソッド名：execAr001
14	アクターを識別	AR002	アクター識別辞書の単語が後方一致する"名詞"をアクターとみなす．	_	クラス名：ActorRules01 メソッド名：execAr002
15	アクターを識別	AR003	アクター識別辞書の単語の前が名詞だった場合は，それを連結した形でアクターとみなす．	_	クラス名：ActorRules01 メソッド名：execAr003

表 3-1-3　検証ルール（2/3）

項番	分類	ルールID	ルール内容	例	対応するクラス／メソッド名
16	アクターを識別	AR004	アクターとみなした用語の後に，env.xmlで指定した接尾語が存在する場合は，それらを連結した形でアクターとみなす． env.xmlに記述する用語例：毎，等，など，等々，etc.,上，中，下，内，外	以下の「」内の用語はアクターとして抽出できなければならない． 「担当者毎」 ※識別辞書に「者」と記述されている．担当者がアクターとみなされ，さらに「毎」が語尾に付与されているので，全体としてアクター用語とみなされなければならない	AR002，AR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
17	アクターを識別	AR005	アクターとみなした用語の後に，括弧で囲まれた文字列が存在する場合は，それらを連結した形でアクターとみなす． 括弧は，BR010で定義した4パターンの括弧ならびに，その再帰的表現とする．	以下の「」内の用語はアクターとして抽出できなければならない． 「都道府県（国）」 ※識別辞書に「都道府県」と記述されている．「（国）」は語尾に付与されているので，全体としてアクター用語とみなされなければならない． 「担当者（オペレータ）」 ※識別辞書に「者」と記述されている．担当者がアクターとみなされ，さらに「（オペレータ）」が語尾に付与されているので，全体としてアクター用語とみなされなければならない．	AR002，AR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
18	データを識別	DR001	入力シナリオ形態素DataSetの単語を連結する事で，データ識別辞書の単語と一致する文字列が存在した場合，その文字列をデータとみなす．	－	クラス名：DataRules01 メソッド名：execDr001
19	データを識別	DR002	データ識別辞書の単語が後方一致する"名詞"をデータとみなす．	－	クラス名：DataRules01 メソッド名：execDr002
20	データを識別	DR003	データ識別辞書の単語の前が名詞だった場合は，それを連結した形でデータとみなす．	－	クラス名：DataRules01 メソッド名：execDr003
21	データを識別	DR004	データとみなした用語の後に，env.xmlで指定した接尾語が存在する場合は，それらを連結した形でデータとみなす． env.xmlに記述する用語例：毎，等，など，等々，etc.,上，中，下，内，外	以下の「」内の用語はデータとして抽出できなければならない 「注文情報等」 ※識別辞書に「情報」と記述されている．担当者がデータとみなされ，さらに「等」が語尾に付与されているので，全体としてデータ用語とみなされなければならない	DR002，DR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
22	データを識別	DR005	データとみなした用語の後に，括弧で囲まれた文字列が存在する場合は，それらを連結した形でデータとみなす． 拾う括弧は，BR010で定義した4パターンの括弧ならびに，その再帰的表現とする．	以下の「」内の用語はデータとして抽出できなければならない． 「注文情報（明細）」 ※識別辞書に「情報」と記述されている．「（明細）」は語尾に付与されているので，全体としてデータ用語とみなされなければならない． 「企画提案書（様式5号）」 ※識別辞書に「書」と記述されている．企画提案書がデータとみなされ，さらに「（様式5号）」が語尾に付与されているので，全体としてデータ用語とみなされなければならない． 「給付台帳（仮）（現行の保育所運営費支弁台帳（総括表））」 ※識別辞書に「台帳」と記述されている．給付台帳がデータとみなされ，さらに，（仮）（現行の保育所運営費支弁台帳（総括表）が語尾に付与されているので，全体としてデータ用語とみなされなければならない．	DR002，DR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
23	画面を識別	SR001	入力シナリオ形態素DataSetの単語を連結する事で，画面識別辞書の単語と一致する文字列が存在した場合，その文字列を画面とみなす．	－	クラス名：ScreenRules01 メソッド名：execSr001
24	画面を識別	SR002	画面識別辞書の単語が後方一致する"名詞"を画面とみなす．	－	クラス名：ScreenRules01 メソッド名：execSr002
25	画面を識別	SR003	画面識別辞書の単語の前が名詞だった場合は，それを連結した形で画面とみなす	－	クラス名：ScreenRules01 メソッド名：execSr003
26	画面を識別	SR004	画面とみなした用語の後に，env.xmlで指定した接尾語が存在する場合は，それらを連結した形で画面とみなす． env.xmlに記述する用語例：毎，等，など，等々，etc.,上，中，下，内，外	以下の「」内の用語は画面として抽出できなければならない 「CMS編集画面上」 ※識別辞書に「画面」と記述されている．CMS編集画面が画面とみなされ，さらに「上」が語尾に付与されているので，全体として画面用語とみなされなければならない	SR002，SR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．

表 3-1-4　検証ルール（3/3）

項番	分類	ルールID	ルール内容	例	対応するクラス／メソッド名
27	画面を識別	SR005	画面とみなした用語の後に，括弧で囲まれた文字列が存在する場合は，それらを連結した形で画面とみなす． 拾う括弧は，BR010で定義した4パターンの括弧ならびに，その再帰的表現	以下の「」内の用語は画面として抽出できなければならない． 「注文入力画面（仮）（その1）」 ※識別辞書に「画面」と記述されている． 「（仮）」「（その1）」は語尾に付与されているので，全体として画面用語とみなされなければならない．	SR002，SR003をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
28	振舞いを識別	VR001	入力シナリオ形態素DataSetの単語を連結する事で，振る舞い識別辞書の単語と一致する文字列が存在した場合，その文字列を振る舞いとみなす．	－	クラス名：ActionRules01 メソッド名：execVr001
29	振舞いを識別	VR002	振る舞い識別辞書の単語が後方一致する"名詞"を振る舞いとみなす．	－	クラス名：ActionRules01 メソッド名：execVr002
30	振舞いを識別	VR003	動詞は，振る舞いとみなす．	－	クラス名：ActionRules01 メソッド名：execVr003
31	振舞いを識別	VR004	名詞[サ変接続]は，振る舞いとみなす．	－	クラス名：ActionRules01 メソッド名：execVr004
32	振舞いを識別	VR005	振る舞いとみなした用語の後に，env.xmlで指定した接尾語が存在する場合は，それらを連結した形で振る舞いとみなす． env.xmlに記述する用語例：毎，等，など，等々，etc.,上，中，下，内，外	以下の「」内の用語は振る舞いとして抽出できなければならない． 「実施する等」 ※識別辞書に「実施する」と記述されている．「実施する」が振る舞いとみなされ，さらに「等」が語尾に付与されているので，全体として振る舞い用語とみなされなければならない．	VR002をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
33	振舞いを識別	VR006	振る舞いとみなした用語の後に，括弧で囲まれた文字列が存在する場合は，それらを連結した形で振る舞いとみなす． 拾う括弧は，BR010で定義した4パターンの括弧ならびに，その再帰的表現	以下の「」内の用語は振る舞いとして抽出できなければならない． 「登録する（基本情報）」 ※識別辞書に「登録する」と記述されている．「（基本情報）」は語尾に付与されているので，全体として振る舞い用語とみなされなければならない．	VR002をおこなう際，判定する単語に対して当処理を行う． CommonUtil.getDetectionRulesWordを実行すると，括弧と接尾語が除去された単語が抽出できる．
34	定義漏れ	TR001	不一致○○DataSetの単語は，定義漏れとみなす． すなわち，入力シナリオ○○DataSetのうち，入力○○定義DataSetと一致しない単語を，不一致○○DataSetとして定義する．	－	モジュール名：DefinitionOmissionVerificationRules メソッド名：execute
34	表記ゆれ	IR001	不一致○○DataSetの単語が入力○○定義DataSetの単語の一部となっていれば，表記ゆれとみなす．	－	モジュール名： OrthographicVariationVerificationRules メソッド名：execute
35	表記ゆれ	IR002	入力○○定義DetaSetの単語が不一致○○DataSet単語の一部となっていれば，表記ゆれとみなす．	－	モジュール： OrthographicVariationVerificationRules メソッド名：execute
36	表記ゆれ	IR003	表記ゆれ○○辞書によって定義された"表記ゆれ単語"に一致すれば，表記ゆれとみなす．	表記ゆれ○○辞書に「Actor,1,ユーザ,"利用者,ユーザー"」と設定されている場合，シナリオから「利用者」と抽出された時，表記ゆれ候補として「ユーザ」を表示する．	モジュール： OrthographicVariationVerificationRules メソッド名：execute
37	用語定義完全一致	CR001	入力○○定義DataSetに定義された用語と入力シナリオ○○DataSetが一致した場合に，用語定義完全一致とみなす．	－	モジュール名：PerfectMatchingVerificationRules メソッド名：execute
38	NGワード検証	NR001	出現NGワードDataSetの単語は，NGワードとみなす． すなわち，入力シナリオにおいて，NGワード辞書に定義された単語と一致する単語が，出現NGワードとなる．	－	モジュール名：NGWordVerificationRules メソッド名：execute
39	その他	OR001	アクター～振る舞いの抽出において同じ単語を2重でカウントしないために，それぞれの設計要素の検出の優先順位は以下の通りとする． データ＞画面＞アクター＞振る舞い （設定ファイルで指定する）	－	①優先順に処理を実行 クラス名：GetIdentificationDictionaryInfo ②優先順を設定ファイルから取得 クラス名：GetConfigInfo
40	その他	OR002	識別辞書の情報は1件も存在しなくても，検証エラーでははない．ただし，識別辞書ファイル自身が存在しない場合は，エラーとする．	－	①識別辞書情報の取得 クラス名：GetIdentificationDictionaryInfo ②ファイル存在チェック クラス名：GetConfigInfo

2）辞書及びNGワード

①②の結果を踏まえて，設計要素（アクター，データ，画面，振る舞い）を識別するための辞書を定義した．本辞書に記述された用語に基づき，表3-1-1のルールのうちNo.2～No.5の各識別ルールを用いて，入力された仕様書から，設計要素を特定する．

◇アクター：

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

○○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アクターとみなす

シナリオ中に，「顧客（会員)」，「顧客（非会員)」などと記述されていたら，アクターを記述する辞書の「顧客」と「会員」の複合語として，この２つのアクターをアクター用語とみなす

◇データ：

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

○○情報，○○データなどの表現や辞書に定義された用語の複合名詞をデータとみなす

◇画面：

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

○○画面，○○スクリーンなどの表現や辞書に定義された用語の複合名詞を画面とみなす

◇振る舞い：

する，実施，実行，管理

○○する，○○実施などの表現や辞書に定義された用語の複合語を振る舞いとみなす

NGワードについては，①②の結果を踏まえさらに，文献[21]のWiegersによるNGワード用語を追加して定義した．定義した結果を表3-1-5に示す．

表 3-1-5　NG ワード定義

No.	NGワード
1	すべて
2	全て
3	あらゆる
4	既存システムと同様
5	既設システムと同様
6	ユーザー
7	ユーザ
8	利用者
9	受け入れ可能な
10	適切な
11	そして
12	または
13	実行できる程度に
14	約
15	だいたい
16	最新版
17	しないわけではない
18	少なくとも
19	最低でも
20	せいぜい
21	上回らない
22	最高の
23	最善の
24	最大の
25	の間で
26	次第である
27	効率的な
28	速い
29	迅速な
30	急速な
31	柔軟な
32	融通が利く
33	i.e.
34	e.g.
35	改善された
36	より良い
37	より速い
38	より優れた
39	より高品質の
40	含めて
41	含めるがこれに限定するものではなく
42	その他
43	など

No.	NGワード
44	のような
45	たとえば
46	ほとんどの場合
47	一般的に
48	通常は
49	ほぼ必ず
50	匹敵する
51	等しい
52	一致する
53	同じ
54	最大化する
55	最小化する
56	最適化する
57	正常時は
58	理想的には
59	任意で
60	おそらくは
61	するはずである
62	したい
63	合理的な
64	必要に応じて
65	適切な場合
66	可能なら
67	当てはまる場合
68	堅牢な
69	シームレスな
70	透過的な
71	エレガントな
72	数個の
73	いくらかの
74	多くの
75	少数の
76	複数の
77	多数の
78	するべきではない
79	しない
80	最先端の
81	十分な
82	サポートする
83	可能にする
84	ユーザーフレンドリーな
85	簡単な
86	容易な

3.1.3 発生した問題および今後の展望

(1) 発生した問題

設計要素であるアクター，データ，画面，振る舞いを識別するためのキーワードや，NG ワードは，対象とするシステムの業務業種領域，顧客側の商習慣，開発部門内の業務ルールによって異なる．システム対象が異なれば，新たなキーワードを追加する必要がある．また，業種業務領域，商習慣，業務ルールの違いによって，新たなキーワードの追加だけではなく，不要なキーワードも発生することが明らかになった．そこで設計要素の識別辞書については，ツールの利用者側でカスタマイズが可能な仕組みにすることを決めた．

(2) 今後の展望

設計要素を識別するための辞書や NG ワードは，対象ドメインや組織の状況に応じて異なることに加え，状況に応じて改善することも必要である．指摘不要の情報や，組織として除外すべき事柄を学習し，識別辞書，NG ワード，検証ルールとして自動変換する仕組みの研究が重要と考えられる．

3.2 研究課題 2「シナリオの一貫性検証の支援ツールの開発（ツール開発)」

3.2.1 当初の想定

(1) 研究内容

研究課題 1「シナリオの一貫性検証知識の形式知化（形式知化)」で定義した検証ルールおよび辞書を組込んだ，シナリオの一貫性検証を支援するツールを開発する．開発するツールに搭載する機能を表 3-2-1 に示す．先行事例 SPES2014-S4a では，F1 および F2 に相当する機能が開発されていた．本研究では，F1 および F2 を一般化するとともに，F3～F5 を追加する．F3 と F4 は新規追加機能であり，F5 は NG ワードを自動的に適切な用語に置き換える補正機能である．

表 3-2-1　シナリオの一貫性検証の支援ツール機能一覧

No.	ID	機能名
1	F1	定義漏れ検証
2	F2	用語不一致検証
3	F3	用語定義完全一致検証
4	F4	NGワード検証
5	F5	シナリオ自動補正

(2) 想定問題と対応策

[想定問題]

シナリオの一貫性検証をツールにより支援する場合，検証の品質は，検証ルールおよび用語辞書の充実度，形態素解析エンジンに与える条件のきめ細かさにより影響があると予測される．検証ルール，辞書，エンジンへの条件を，分析結果から一意に決定すると，対象とする組織の条件により，適切な検証結果が得られないリスクがある．

[対応策]

・各社間共通と固有の分離ならびに拡張・改善をしやすいアーキテクチャとする

開発対象のツールのアーキテクチャを図 3-2-1 に示す．本ツールでは，F1～F4 までの検証機能と F5 のシナリオ自動補正からなる機能と，検証ノウハウにあたる検証ルールと辞書，検証エンジンは分離した構造とする．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シナリオを解釈する形態素解析エンジンで構成する．検証ルールと辞書は図 3-1-1 に示したように構造化し維持管理と拡張を容易にする．さらに，検証ルールと辞書は，研究課題 1 で明らかにした，各社共通部分と各社固有部分に分けて定義する．ツール利用において，ユーザが選択した検証の範囲を設定ファイルで定義することにする．

・2 回の反復による開発により，1 回目の反復での問題点を，2 回目の反復で反映できるように工夫する．

シナリオにはソフトウェアの要求仕様に固有の表現が含まれるため，OSS の形態素解析エンジンをそのまま用いただけでは，要求仕様の構成要素を特定することは困難であり，複合

語や係り受けの関係性を辞書等で抽出可能なようにしておく必要がある．これには，調整しながらの開発が必要と考えられ，本研究では，開発期間を考慮し，2 回の反復による開発を行い，検証エンジンと機能間の調整を行いながら開発する．

【各反復の内容】
反復 1：検証エンジン，ルール，辞書の初期設定と，定義漏れ検証及び用語不一致検証ルールを試作しツールα1 を開発する．
反復 2：残りの機能（F3～F5）について試作しツールα2 を開発する．
統合：反復 1 と 2 を統合し，チューニングを行い，ツールβ版としてまとめる．

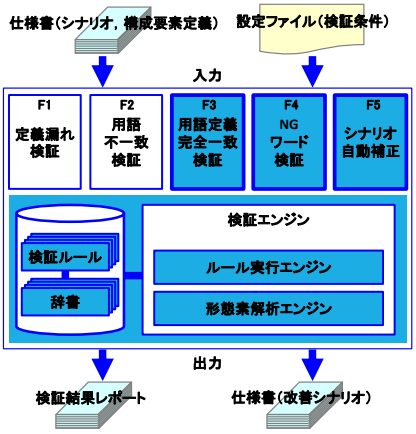


図 3-2-1 シナリオの一貫性検証の支援ツールのアーキテクチャ

【役割分担】
本ツールの開発では，要求定義，設計，開発，テストの一連を工学院大学の研究メンバーで行い，ツールプログラム各機能の試作は外注化する．反復毎に試作した各機能を工学院大学側で統合，チューニングし，ツールとして仕上げる．なお，設計ドキュメント整理，ツールのシステムテストやテスト成績書の記載，利用マニュアル等ドキュメント作成の補助として，アルバイト等を用いて効率的に進める．

3.2.2 研究プロセスと成果

(1) 研究プロセス

研究は次の①～⑥の手順で実施した．

①反復 1：要求定義・設計

検証エンジン，ルール，辞書の初期設定と，表 3-2-1 の F1：定義漏れ検証および F2：用語不一致検証の要求定義ならびに設計を行った．ツール試作のための外注仕様書を作成した．なお，本要求定義および外注仕様書作成において，反復 1 の試作範囲である機能 F1，F2 に対して，機能 F3～F5 が追加されることを想定し，ツール全体の仕様も定義した．

②反復 1：開発・テスト

①の結果に基づき，検証エンジン，ルール，辞書の初期設定と，表 3-2-1 の F1：定義漏れ検証および F2：用語不一致検証からなる機能を試作し，ツールテスト，チューニングを実施し，ツールα1 を開発した．

③反復 2：要求定義・設計

表 3-2-1 の機能（F3～F5）についての要求定義ならびに設計を行った．反復 1 での要求定義の結果に基づき，反復 1 でのチューニングで得られた結果を反映させた．また，反復 1 で開発した外注仕様書のツール全体仕様に基づき，機能 F3～F5 の試作のための外注仕様書を作成した．

④反復 2：開発・テスト

③の結果に基づき，表 3-2-1 の機能（F3～F5）について試作し，ツールテスト，チューニングを実施し，ツールα2 を開発した．

⑤反復 1 と 2 の統合

反復 1 と反復 2 で開発した，ツールα 1 とツールα2 を統合し，一連のツールテストを行った．

⑥まとめ・マニュアル作成

⑤の結果に基づき，ルール，辞書を調整するとともに，利用マニュアルを作成し，ツールβとしてとりまとめた．

(2) 具体的な研究成果の内容

「①反復 1：要求定義・設計」では，検証エンジン，検証エンジン，ルール，辞書の初期設定および定義漏れ検証，用語不一致検証の試作プログラムの外部仕様設計結果として，「反復 1 外注仕様書」を作成した．「②反復 1：開発・テスト」では，外注先を選定の上，「反復 1 外注仕様書」に基づき，検証エンジン，ルール，辞書の初期設定および定義漏れ検証，用語不一致検証が実装されたツールプログラムを開発した．また，ツールプログラムのテストを実施した．本テストでは，実際の案件で作成された要求仕様書に対して，人手による検証データを作成し，ツールプログラムでの検証結果を突き合わせるテストを行い，発生した不具合の改修ならびに辞書やルール等の改善案を洗い出した．

「③反復 2：要求定義・設計」では，反復 1 の開発結果に基づいて，用語完全一致検証，NG ワード検証，シナリオ自動補正についての要求定義ならびに外部仕様設計を行い，その結果として，「反復 2 外注仕様書」を作成した．

「④反復 2：開発・テスト」では，外注先を選定の上，「反復 2 外注仕様書」に基づき，検証エンジン，ルール，辞書の初期設定および定義漏れ検証，用語不一致検証，用語完全一致検証，NG ワード検証，シナリオ自動補正が実装されたツールプログラムを開発した．また，反復 1 と同様に，ツールプログラムのテストを実施した．テストでは，反復 1 で作成したテストデータに加え，新たに作成した用語完全一致検証，NG ワード検証，シナリオ自動補正に対しても人手により検証データを作成した上で，ツールプログラムによる検証結果

と突き合わせるテストを行い，妥当性を確認した．

「⑤反復 1 と 2 の統合」では，「④反復 2：開発・テスト」の結果を踏まえて，辞書，ルールを整理した上で，「ツールプログラム一式」としてとりまとめた．また，テストデータとともに再現率と適合率のデータを算出し「ツールテスト結果一式」としてとりまとめた．

「⑥まとめ・マニュアル作成」においては，本ツールに利用者が所属する組織に向けにツルールや辞書をカスタマイズしやすくするために利用するドキュメントを作成した．作成したドキュメントは，「ルール定義一覧」と「ユーザマニュアル」である．「ルール定義一覧」では，検証ルール定義およびプログラム対応表を作成した．「ユーザマニュアル」には，ツールのインストール方法，検証ルール，検証レポートのカスタマイズ方法を説明した．また，ツール利用のためのユースケースを示した．

ツールは，反復 1 と反復 2 からなる 2 回の反復型により開発した．反復 2 では，反復 1 の開発結果を包含して，インクリメンタルな開発をした．反復型開発を行った理由は，シナリオにはソフトウェアの要求仕様に固有の表現が含まれるため，OSS の形態素解析エンジンをそのまま用いただけでは，要求仕様の構成要素を特定することは困難であり，複合語や係り受けの関係性を辞書等で抽出可能にしておく必要があるが，これには調整しながらの開発が必要と考えられるためである．そこで，本ツール開発では，開発期間を考慮し，検証エンジンと機能間の調整を行いながら反復型の開発を採用し実行した．

以下では，開発した「シナリオの一貫性検証支援ツール」について，設計，開発，テストによる一連の開発成果物を説明する．なお，反復 1 は反復 2 に包含されるため，反復 2 実施後にとりまとめたツールプログラム一式ならびにドキュメント類一式について説明する．

[ユースケース]

シナリオの一貫性検証支援ツールのユースケースモデルを図 3-2-2 に示す．本ツールの利用者は，要求仕様であるシナリオを記述する開発者または仕様のとりまとめを行う管理者などの技術者である．本ツールにより，開発者または管理者は，記述したシナリオにおいて，アクター，データ，画面，振る舞いの定義漏れ検証，用語不一致検証，用語定義完全一致検証，NG ワード検証，NG ワードを妥当な用語に置き換えるシナリオ自動補正を行う．

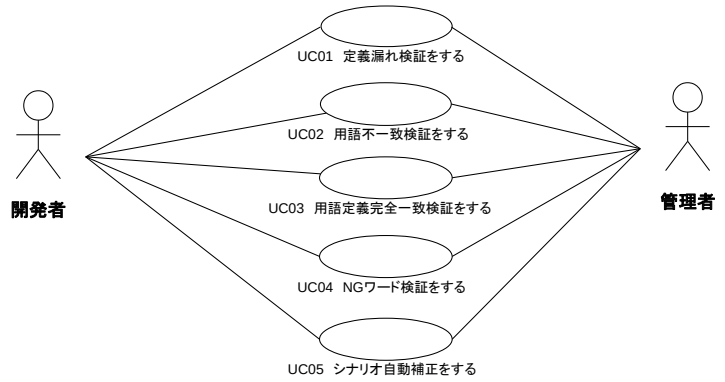


図 3-2-2　シナリオの一貫性検証支援ツール　ユースケース図

[機能一覧]

上記ユースケースを実現するために，本ツールが備える機能を表 3-2-2 に示す.

表 3-2-2　シナリオの一貫性検証支援ツール　機能一覧

No.	機能名	ID	処理概要
1	定義漏れ検証	F1	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，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.
2	用語不一致検証	F2	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，それぞれ，対応する定義表と別の用語表現(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.
3	用語定義完全一致検証	F3	検証対象の仕様(シナリオ)および，アクター，データ，画面，振る舞い等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，振る舞い等が，それぞれ，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，振る舞い等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指摘する.
4	NGワード検証	F4	検証対象の仕様(シナリオ)および，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.
5	シナリオ自動補正	F5	検証対象の仕様(シナリオ)および，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，おなじく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.

[設計アーキテクチャ]

本ツールの設計アーキテクチャを図 3-2-3 に示す．本ツールでは，F1～F4 までの検証機能と F5 のシナリオ自動補正からなる機能と，検証ノウハウにあたる検証ルールと辞書，検証エンジンは分離した構造とする．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シナリオを解釈する形態素解析エンジンで構成する．検証ルールと辞書は構造化し，各企業共通部分と各企業固有部分に分けて定義可能とすることで，維持管理や拡張を容易に実現しやすくする．また，利用者が選択した検証の範囲は，設定ファイルにおいて定義することにする.

課題 1 において，シナリオの整合性の一貫性検証の知識を検証ルールと辞書として形式知化した．検証ルールと辞書を，一般化かつ組織の違いに応じた拡張性を持たせたて，ツールに組込むため，オブジェクト指向モデリングにより，ルールを種類別に整理分類し，拡張可能な形式にした．また，検証を実行するエンジンについても，拡張性を考慮したクラス構造となるように工夫した．ここでは，デザインパターンの Strategy パターンと TemplateMethod パターンを適用することとした.

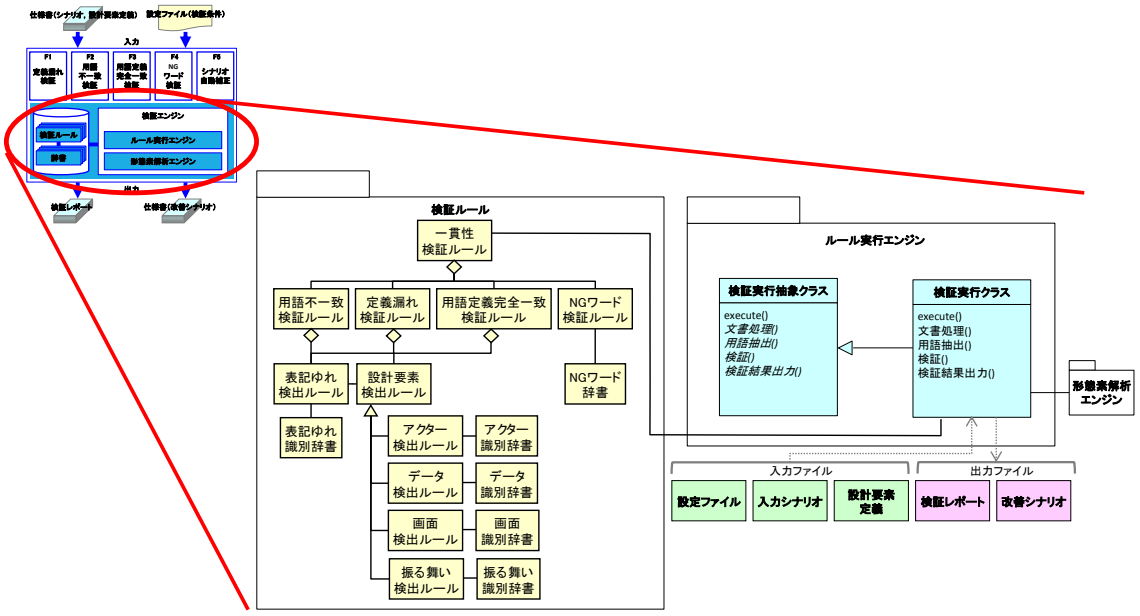


図 3-2-3　シナリオの一貫性検証支援ツールの設計アーキテクチャ

[動作環境]

・本ツールは，OS　Windows® 8.1 環境の下で動作可能とした．

・本ツールによる検証の実行は，コマンドラインでの実行ならびに exe ファイル等のダブルクリックによる実行とした．

[開発環境]

・OSS の形態素解析ソフトウェア MeCab[33]を用いた．

・ツール（アプリ）開発言語は，スクリプト言語である Ruby[34]とした．

[機能仕様定義]

各機能 F1～F5 の仕様を以下に示す．各仕様は，各機能への入力，入力に対する処理，各機能からの出力の順に説明する．

(F1)定義漏れ検証

(F1)定義漏れ検証入力：

(1)設定ファイル

(2)入力シナリオ

(3)入力設計要素定義表

(F1)定義漏れ検証処理：

・　（共通）設定ファイルより，検証対象の設計要素（アクター，データ，画面，振る舞い）と，検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシ

ナリオ自動補正の実施有無，検証レポートへの品詞別計測ログの出力有無を読み取る．
・ 設定ファイルの検証の種類に「定義漏れ」が設定されていた場合に本機能を実行する．
・ 入力シナリオと入力設計要素定義表を読み取り，形態素解析を行い，出現用語，品詞，出現数からなる中間データセットを作成する．
・ 形態素解析で設定ファイルから読み取った設計要素の観点で，上記中間データセットに対して，定義漏れ検証を実施する．すなわち，設定ファイルから読み取った検証対象の仕様（シナリオ）および，アクター，データ，画面，振る舞い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．アクター，データ，画面，振る舞いのうち，どの設計要素の観点で検証するかは，設定ファイルでの指定に基づく．
・ 上記検証結果を，検証レポートに出力する．
・ （共通）設定ファイルに指定された検証の種類ならびに NG ワード検証，シナリオ自動補正の実施有無に基づき，検証を行う．

(F1)定義漏れ検証出力：
(1)検証レポート
・シナリオに出現する設計要素名，出現数，設計要素定義との一致/不一致
・品詞別用語，出現数

(F2)用語不一致検証

(F2)用語不一致検証入力：
(1)設定ファイル
(2)入力シナリオ
(3)入力設計要素定義表

(F2)用語不一致検証処理：
・ （共通）設定ファイルより，検証対象の設計要素（アクター，データ，画面，振る舞い）と，検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシナリオ自動補正の実施有無，検証レポートへの品詞別計測ログの出力有無を読み取る．
・ 設定ファイルの検証の種類に「用語不一致」が設定されていた場合に本機能を実行する．
・ 入力シナリオと入力設計要素定義表を読み取り，形態素解析を行い，出現用語，品詞，出現数からなる中間データセットを作成する．
・ 形態素解析で設定ファイルから読み取った設計要素の観点で，上記中間データセットに対して，用語不一致検証を実施する．すなわち，設定ファイルから読み取った検証対象の仕様（シナリオ）および，アクター，データ，画面，振る舞い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，振る舞いが，それぞれ，対応する定義表と別の用語表現（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．アクター，データ，画面，振る舞いのうち，どの設計要素の観点で検証するかは，設定ファイルでの指定に基づく．

・ 上記検証結果を，検証レポートに出力する．
・ （共通）設定ファイルに指定された検証の種類ならびに NG ワード検証，シナリオ自動補正の実施有無に基づき，検証を行う．

(F2)用語不一致検証出力：
(1)検証レポート
・シナリオに出現する設計要素名，出現数，設計要素定義との一致/不一致，不一致の場合の定義表における候補
・品詞別用語，出現数

(F3)用語定義完全一致検証

(F3)用語定義完全一致検証入力：
(1)設定ファイル
(2)入力シナリオ
(3)入力設計要素定義表

(F3)用語定義完全一致検証処理：
・ （共通）設定ファイルより，検証対象の設計要素（アクター，データ，画面，振る舞い）と，検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシナリオ自動補正の実施有無，検証レポートへの品詞別計測ログの出力有無を読み取る．
・ 設定ファイルの検証の種類に「用語定義完全一致」が設定されていた場合に本機能を実行する．
・ 入力シナリオと入力設計要素定義表を読み取り，形態素解析を行い，出現用語，品詞，出現数からなる中間データセットを作成する．
・ 形態素解析で設定ファイルから読み取った設計要素の観点で，上記中間データセットに対して，用語定義完全一致検証を実施する．すなわち，設定ファイルから読み取った検証対象の仕様（シナリオ）および，アクター，データ，画面，振る舞い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，振る舞いが，それぞれ，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指摘する．アクター，データ，画面，振る舞いのうち，どの設計要素の観点で検証するかは，設定ファイルでの指定に基づく．
・ 上記検証結果を，検証レポートに出力する．
・ （共通）設定ファイルに指定された検証の種類ならびに NG ワード検証，シナリオ自動補正の実施有無に基づき，検証を行う．

(F3)用語定義完全一致検証出力：
(1)検証レポート
・シナリオに出現する設計要素名，出現数，設計要素定義との一致/不一致

・シナリオ中に出現しない設計要素定義，設計要素数
・品詞別用語，出現数

(F4)NG ワード検証

(F4)NG ワード検証
入力：
(1)設定ファイル
(2)入力シナリオ
(3) NG ワード定義表

(F4)NG ワード検証
処理：
- （共通）設定ファイルより，検証対象の設計要素（アクター，データ，画面，振る舞い）と，検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシナリオ自動補正の実施有無，検証レポートへの品詞別計測ログの出力有無を読み取る．
- 設定ファイルにおいて，「NG ワード検証」が設定されていた場合に本機能を実行する．
- 入力シナリオと入力設計要素定義表を読み取り，形態素解析を行い，出現用語，品詞，出現数からなる中間データセットを作成する．
- 上記中間データセットに対して，NG ワード検証を実施する．すなわち，検証対象の仕様（シナリオ）および，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．
- 上記検証結果を，検証レポートに出力する．
- （共通）設定ファイルに指定された検証の種類ならびに NG ワード検証，シナリオ自動補正の実施有無に基づき，検証を行う．

(F4)NG ワード検証
出力：
(1)検証レポート
・NG ワード，シナリオへの出現数，置換候補用語
・品詞別用語，出現数

(F5)シナリオ自動補正

(F5)シナリオ自動補正入力：
(1)設定ファイル
(2)入力シナリオ
(3) NG ワード定義表

(F5)シナリオ自動補正処理：

- （共通）設定ファイルより，検証対象の設計要素（アクター，データ，画面，振る舞い）と，検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシナリオ自動補正の実施有無，検証レポートへの品詞別計測ログの出力有無を読み取る
- 設定ファイルにおいて，「シナリオ自動補正」の実施が設定されていた場合に本機能を実行する．
- 入力シナリオと入力設計要素定義表を読み取り，形態素解析を行い，出現用語，品詞，出現数からなる中間データセットを作成する．
- 上記中間データセットに対して，NG ワード検証を経て，シナリオを自動補正する．検証対象の仕様（シナリオ）および，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，おなじく 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．

(F5)シナリオ自動補正出力：
(1)改善シナリオ

[機能とその構成要素の考え方]

検証ルールの構造は，図 3-1-1 に示した通りである．検証ルールは，各検証知識を部品化し，組織毎に容易に拡張やカスタマイズが可能なように構造化した．図 3-1-1 のうち，用語定義完全一致検証ルール，NG ワード検証ルール，NG ワード辞書は，反復 2 の開発範囲である．なお，反復 2 の範囲であるこれら 3 つのクラス（用語定義完全一致検証ルール，NG ワード検証ルール，NG ワード辞書）以外は，反復 1 の開発対象範囲である．反復 1 の開発結果に基づき，表記ゆれ検出ルールの構造を詳細化し，表記ゆれ識別辞書を参照することにした．

図 3-2-4 に「定義漏れ検証」の処理フローを示す．この検証は，シナリオ中に出現したアクター用語が，アクター定義に定義されていない場合に，その定義漏れを指摘し，検証レポートに検証結果を出力する．

図 3-2-5 に「用語不一致検証」の処理フローを示す．この検証は，シナリオ中に出現したアクター用語が，アクター定義に類似表現で定義されている場合に，その表記ゆれを指摘し，検証レポートに検証結果を出力する．

図 3-2-6 に「用語定義完全一致検証」の処理フローを示す．この検証は，シナリオ中に出現したアクター用語が，アクター定義に定義されており，かつ，アクター定義に定義されたアクターがシナリオ中に 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は指摘し，検証レポートに検証結果を出力する．

図 3-2-7 に「NG ワード検証」の処理フローを示す．この検証は，シナリオ中に NG ワードが出現していれば，その NG ワードを指摘し，検証レポートに検証結果を出力する．

図 3-2-8 に「シナリオ自動補正」の処理フローを示す．これは，シナリオ中に NG ワードが出現しており，かつ，当該 NG ワードの置き換え用語が定義されている場合に，シナリオ中の NG ワードを置き換え用語で置換したシナリオを自動生成し，NG ワード検証および置き換え結果を検証レポートとして出力する．

図 3-2-9 に「ルール実行エンジン」の構成を示す．ルール実行エンジンは，入力された設

定ファイル，シナリオ，設計要素定義情報に基づき，図 3-1-1 に示した各種検証ルールと辞書，形態素解析エンジンを制御して，検証を実行し，結果を検証レポートに出力する一連のアクションを実施するフレームワークである．ルール実行エンジンにおいて，検証を実行するための「検証実行クラス」はデザインパターンの TemplateMethod パターンを用いて実装することで，各組織による検証のカスタマイズを容易にする．

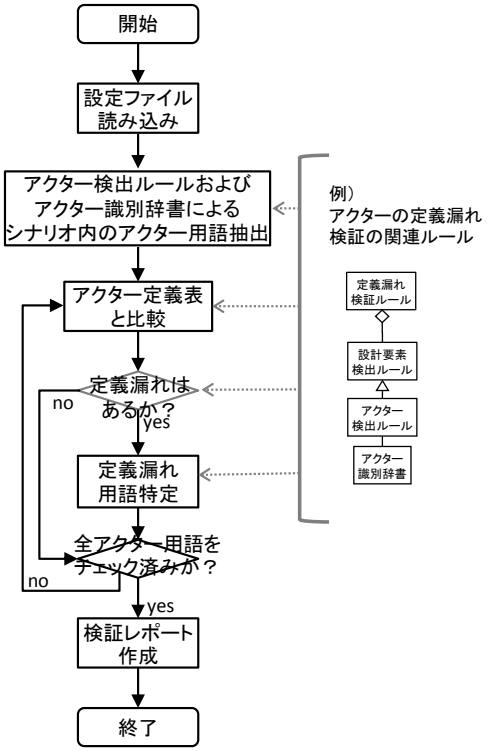


図 3-2-4　検証ルールを用いた「定義漏れ検証」の流れ

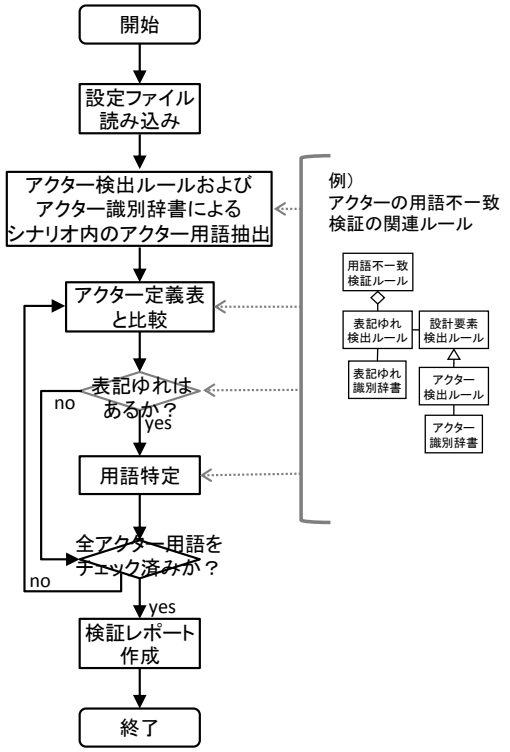


図 3-2-5　検証ルールを用いた「用語不一致検証」の流れ

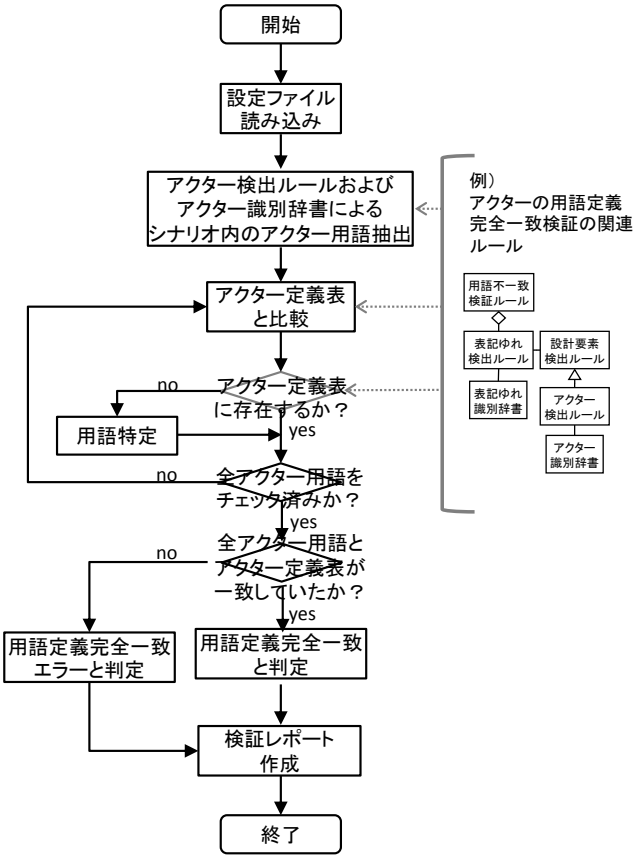


図 3-2-6　検証ルールを用いた「用語定義完全一致検証」の流れ

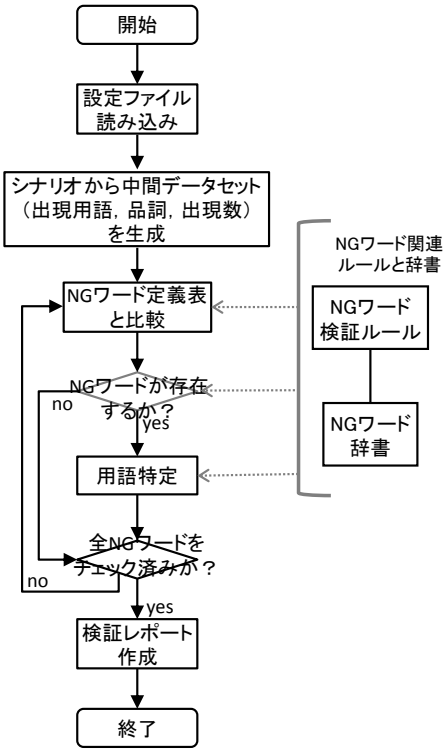


図 3-2-7　検証ルールを用いた「NG ワード検証」の流れ

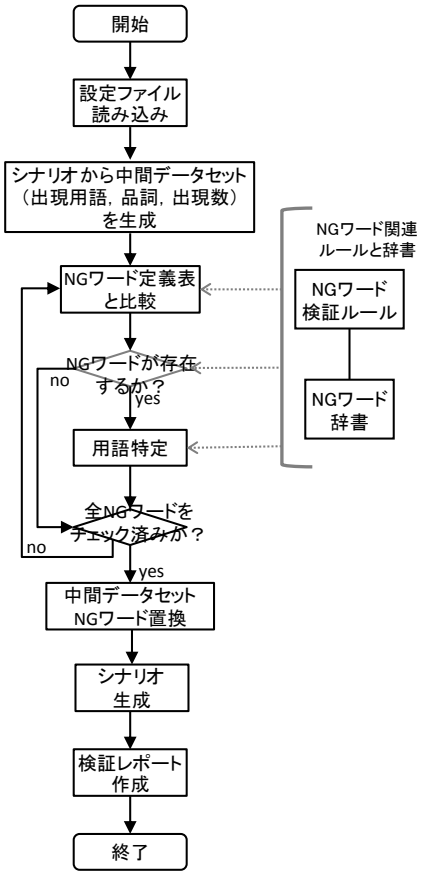


図 3-2-8　検証ルールを用いた「シナリオ自動補正」の流れ

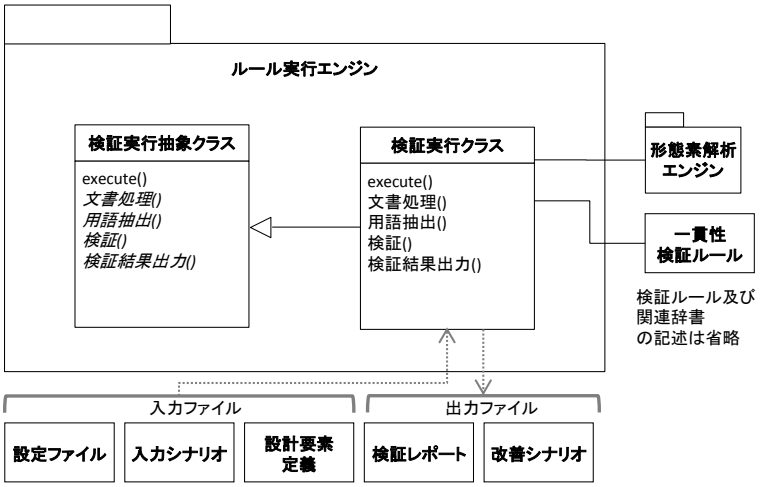


図 3-2-9　ルール実行エンジンの構成

［設定ファイル］

本ツールを実行する際に，利用者が検証条件である検証対象（アクター，データ，画面，振る舞い），検証の種類（定義漏れ，用語不一致，用語定義完全一致），NG ワード検証およびシナリオ自動補正の実施有無，品詞別計測ログの出力有無，利用者指定の辞書および検証ルールを指定するためのテキストファイルを XML 形式で記述する．

図 3-2-10 および図 3-2-11 に設定ファイルの記述例を示す．タグに挟まれた“1”は，対象とする検証および検証対象を実施することを示す．なお，本設定ファイルは，反復 1 での開発結果を反映させて，タグ名をわかりやすくするなどの改善をした．

［シナリオ］

入力シナリオは，テキストファイルで記述されたものとする．Pdf，Microsoft Word®，Microsoft Excel®等で記述されたシナリオを，あらかじめテキストファイル形式に変換しておくものとする．

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration>
  <アクター>
    <定義漏れ検証>1</定義漏れ検証>
    <表記ゆれ検証>1</表記ゆれ検証>
    <完全一致検証>1</完全一致検証>
  </アクター>
  <データ>
    <定義漏れ検証>1</定義漏れ検証>
    <表記ゆれ検証>1</表記ゆれ検証>
    <完全一致検証>1</完全一致検証>
  </データ>
  <画面>
    <定義漏れ検証>1</定義漏れ検証>
    <表記ゆれ検証>1</表記ゆれ検証>
    <完全一致検証>1</完全一致検証>
  </画面>
  <振る舞い>
    <定義漏れ検証>1</定義漏れ検証>
    <表記ゆれ検証>1</表記ゆれ検証>
    <完全一致検証>1</完全一致検証>
  </振る舞い>
  <NG ワード検証>1</NG ワード検証>
  <シナリオ自動補正>1</シナリオ自動補正>
  <品詞別用語ログ出力>1</品詞別用語ログ出力>
  <識別辞書>
    <アクター>
      <辞書>識別辞書１.csv</辞書>
    </アクター>
    <データ>
      <辞書>識別辞書１.csv</辞書>
    </データ>
    <画面>
      <辞書>識別辞書１.csv</辞書>
    </画面>
    <振る舞い>
      <辞書>識別辞書１.csv</辞書>
    </振る舞い>
  </識別辞書>
```

図 3-2-10　設定ファイル記述構成（1/2）

```
    <検出ルール>
      <アクター>
        <ルール>actor-rules-01.rb</ルール>
      </アクター>
      <データ>
        <ルール>data-rules-01.rb</ルール>
      </データ>
      <画面>
        <ルール>screen-rules-01.rb</ルール>
      </画面>
      <振る舞い>
        <ルール>action-rules-01.rb</ルール>
      </振る舞い>
    </検出ルール>
    <表記ゆれ検出辞書>
      <アクター>
        <辞書>表記ゆれ辞書１.csv</辞書>
      </アクター>
      <データ>
        <辞書>表記ゆれ辞書１.csv</辞書>
      </データ>
      <画面>
        <辞書>表記ゆれ辞書１.csv</辞書>
      </画面>
      <振る舞い>
        <辞書>表記ゆれ辞書１.csv</辞書>
      </振る舞い>
    </表記ゆれ検出辞書>
    <設計要素検出優先順位>
      <アクター>3</アクター>
      <データ>1</データ>
      <画面>2</画面>
      <振る舞い>4</振る舞い>
    </設計要素検出優先順位>
  </Configuration>
```

図 3-2-11　設定ファイル記述構成（2/2）

[辞書]

本ツールでは2種類の辞書を扱う．一つは，設計要素の識別辞書であり，これは，シナリオ中に記述された用語から，設計要素（アクター，データ，画面，振る舞い）に該当する用語を識別するために用いる情報である．辞書の記述形式は以下とする．また，それぞれ設計要素を検出するためのルールの仕様は以下とする．辞書および設計要素検出ルールは，利用する組織別に様々なバリエーションが考えられるため，拡張が可能な形式とする．辞書ファイルと検出ルールのファイルは，設定ファイルによって，取り換え可能にする．初期データとして提供する辞書は，3.1.2(2)で述べた辞書とする．

もう一つの辞書は，表記ゆれ識別辞書である．表記ゆれとみなす異音同義語を定義するための辞書である．例えば，アクターを表す用語である，ユーザ，利用者，ユーザーは，異音同義語である．この場合，利用者，ユーザーを「ユーザ」として統一化するため，表記ゆれを指摘するためのアクターの表記ゆれを識別する辞書の記述例は次のようになる．

Actor, 1, ユーザ, “利用者,ユーザー”

[設計要素の識別辞書の記述形式]

・検出したい設計要素
・項番
・用語名

[表記ゆれ識別辞書の記述形式]

・検出したい設計要素
・項番
・用語名
・“異音同義語の用語名リスト”

[入力設計要素定義表]

入力設計要素定義表とは，要求仕様中に定義された，設計要素（アクター，データ，画面，振る舞い）の定義情報（インスタンス）のことである．本定義表は，csvファイルで記述されたものとする．pdf， Microsoft Word®, Microsoft Excel®等で記述されたシナリオを，あらかじめ csv ファイル形式に変換しておき，それを入力する入力設計要素定義表とみなすものとする．

[入力設計要素定義表の記述形式]

・対象となる設計要素識別タグ（Actor, Data, Screen, Actionのいずれか）
・項番
・用語名

「用語名」以降に，用語の説明等が記述されている場合でも，上記3項目の処理が行えるようにする．

［検証レポート］

検証レポートは，入力されたシナリオの一貫性検証を実行した評価結果を出力したテキストファイルである．検証レポートの出力形式を以下に示す．なお，検証レポートのインスタンスは，設定ファイルで指定した内容に基づいて生成される．ここで，〈品詞別用語ログ〉とは，シナリオ中に出現した単語とその出現数を，品詞別に分類して出力したものである．本ツールで取りあげる品詞としては，名詞または動詞に限定する．

［検証レポートの出力形式］

〈対象ファイル名情報〉
・入力シナリオファイル名
・文字数

〈定義漏れ検証〉
・シナリオに出現する設計要素名
・出現数
・設計要素定義との一致/不一致
（以上，設計要素の数分の繰り返し）

〈用語不一致検証〉
・シナリオに出現する設計要素名
・出現数
・設計要素定義との一致/不一致
・不一致の場合の定義表における候補
（以上，設計要素の数分の繰り返し）

〈用語定義完全一致検証〉
・シナリオに出現する設計要素名
・出現数
・設計要素定義との一致/不一致
（以上，設計要素の数分の繰り返し）

・シナリオ中に出現しない設計要素定義
・設計要素数
（以上，設計要素の数分の繰り返し）

〈NG ワード検証〉
・NG ワード
・シナリオへの出現数
・置換候補用語
（以上，NG ワードの数分の繰り返し）

〈品詞別用語ログ〉
・品詞名
・品詞別用語
・出現数
（以上，用語の数分の繰り返し）

[クラス設計]

図 3-2-12 にシナリオの一貫性検証支援ツールのクラスモデルを示す．本モデルは，主としてルール実行エンジン，ルールと辞書，各種データセットから構成している．本モデルの設計においては，オブジェクト指向技術，デザインパターンを駆使して可変ポイントを明確化し，組織毎に拡張しやすい構造をねらっている．各クラスの定義を表 3-2-3，表 3-2-4，表 3-2-5 に示す．

表記ゆれ検出ルール（Actor を例に）は，シナリオに記述された Actor 用語（定義済みかどうかの処理済み）に対して，次のいずれかに当てはまる場合に表記ゆれと判断する仕組みとした．

・不一致 ActorDataSet の単語が入力 Actor 定義 DataSet の単語の一部となっている
・入力 Actor 定義 DetaSet の単語が不一致 ActorDataSet 単語の一部となっている
・表記ゆれ Actor 辞書によって定義された″表記ゆれ単語″に一致する

これにより，表記ゆれ Actor 辞書に「Actor,1,ユーザ,″利用者,ユーザー″」と設定されている場合，シナリオから「利用者」と抽出された時，表記ゆれ候補として「ユーザ」を表示する．なお，表記ゆれの候補となる単語が複数存在した場合はカンマで連結して複数表示する．

各クラスの論理設計後，実装したプログラムおよびクラスの設計結果を表 3-2-6 および表 3-2-7 に示す．

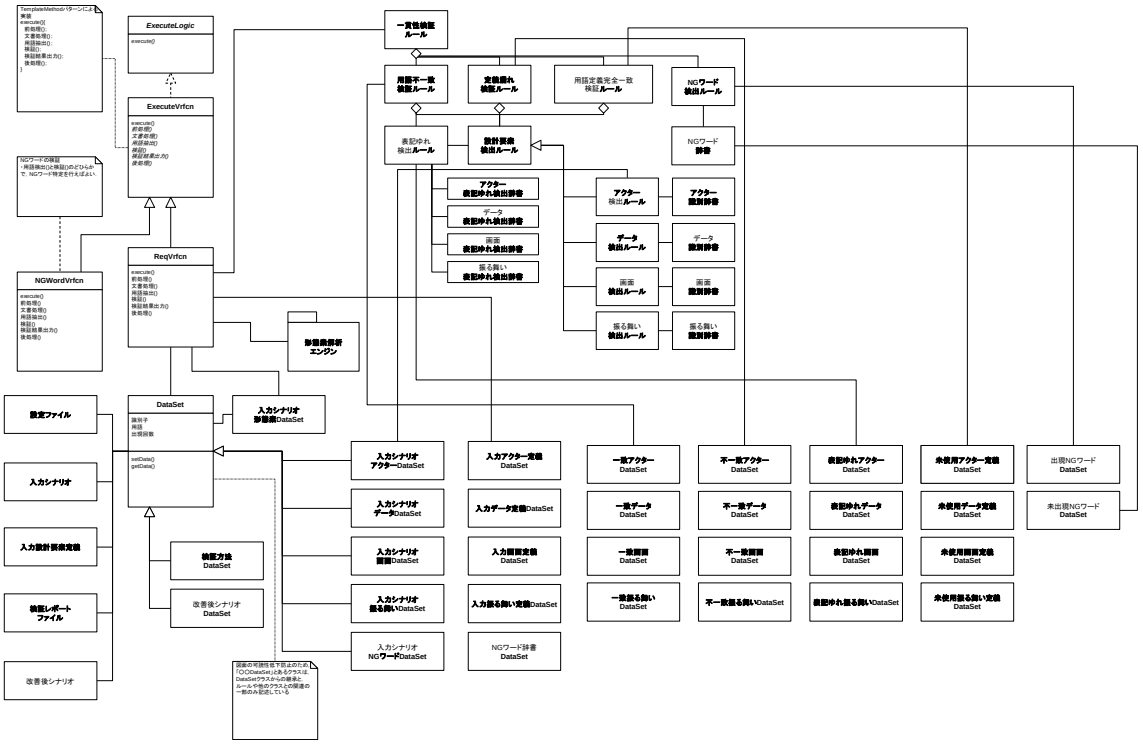


図 3-2-12　シナリオの一貫性検証支援ツールのクラスモデル

［設計クラス一覧］

表 3-2-3　設計クラス一覧（1/3）

No.	クラス名	概要	親クラス/インタフェース	分類
1	ExcecuteLogic	ロジックの実行のインタフェース	なし	Engine
2	ExecuteVrfctn	検証を実行の枠組みを定義した抽象クラス	ExcecuteLogic	Engine
3	ReqVrfctn	要求仕様の検証の方法を定義したクラス	ExecuteVrfctn	Engine
4	NGWordVrfctn	要求仕様のNGワードの検証方法を定義したクラス	ExecuteVrfctn	Engine
5	DataSet	テンポラリな内部データの設定と取得の枠組みを定義した抽象クラス	なし	Dataset
6	設定ファイル	検証の対象，検証の種類，NGワード検証／シナリオ自動補正の実施有無，検証レポートの表示方法，独自辞書などの指定を記述した，検証の実行条件を記述した入力ファイル	なし	File
7	入力シナリオ	検証対象となる，シナリオを記述したテキストファイル	なし	File
8	入力設計要素定義	要求仕様書に記述された，アクター／データ／画面／振る舞いの設計要素の定義ファイル	なし	File
9	検証レポートファイル	検証結果を記述するテキストファイル	なし	File
10	改善後シナリオ	NGワード検証後にNGワードを置き換えたシナリオを記述したテキストファイル	なし	File
11	検証レポートファイル	検証結果を記述するテキストファイル	なし	File
12	NGワード辞書	NGワード検証で利用する，当該要求仕様において，使ってはいけない用語とその言い換え用語を定義した辞書	なし	File
13	形態素解析エンジン	入力された自然言語で記述されたシナリオから，品詞別の単語に抽出するための，自然言語解析エンジン．OSSをベースに定義．	なし	Logic
14	検証方法DataSet	入力された設定ファイルにおいて定義された，検証方法（検証対象，実施する検証の種類，NGワード検証実施有無，シナリオ自動補正実施有無，検証レポートへの出力方法）を読み取った結果を，保管するためのDataSetクラス	DataSet	DataSet
15	入力シナリオ形態素DataSet	形態素解析エンジンによって，入力されたシナリオを解析した結果を保管するためのDataSetクラス	DataSet	DataSet
16	改善後シナリオDataSet	入力されたシナリオを解析した結果を保管するためのDataSetクラスに対して，NGワードを置き換えたを保管するDataSetクラス	DataSet	DataSet
17	入力シナリオアクターDataSet	入力されたシナリオをから，アクターに相当する用語を特定した結果を保管するためのDataSetクラス	DataSet	DataSet
18	入力シナリオデータDataSet	入力されたシナリオをから，データに相当する用語を特定した結果を保管するためのDataSetクラス	DataSet	DataSet
19	入力シナリオ画面DataSet	入力されたシナリオをから，画面に相当する用語を特定した結果を保管するためのDataSetクラス	DataSet	DataSet
20	入力シナリオ振る舞いDataSet	入力されたシナリオをから，振る舞いに相当する用語を特定した結果を保管するためのDataSetクラス	DataSet	DataSet
21	入力シナリオNGワードDataSet	入力されたシナリオから，NGワード検証の対象用語に相当する用語を特定した結果を保管するためのDataSetクラス	DataSet	DataSet
22	入力アクター定義DataSet	入力されたアクターの定義ファイルを，定義されているアクターを保管するためのDataSetクラス	DataSet	DataSet
23	入力データ定義DataSet	入力されたデータの定義ファイルを，定義されているデータを保管するためのDataSetクラス	DataSet	DataSet
24	入力画面定義DataSet	入力された画面の定義ファイルを，定義されている画面を保管するためのDataSetクラス	DataSet	DataSet
25	入力振る舞い定義DataSet	入力された振る舞いの定義ファイルを，定義されている振る舞いを保管するためのDataSetクラス	DataSet	DataSet
26	NGワード辞書DataSet	入力されたNGワード定義表から特定したNGワードを，保管するためのDataSetクラス	DataSet	DataSet
27	一致アクターDataSet	入力シナリオアクターDataSetのうち，入力アクター定義DataSetと一致する，アクターを保管するためのDataSet	DataSet	DataSet
28	一致データDataSet	入力シナリオデータDataSetのうち，入力データ定義DataSetと一致する，アクターを保管するためのDataSet	DataSet	DataSet
29	一致画面DataSet	入力シナリオ画面DataSetのうち，入力画面定義DataSetと一致する，アクターを保管するためのDataSet	DataSet	DataSet

表 3-2-4　設計クラス一覧（2/3）

No.	クラス名	概要	親クラス/インタフェース	分類
30	一致振る舞いDataSet	入力シナリオ振る舞いDataSetのうち，入力振る舞い定義DataSetと一致する，アクターを保管するためのDataSet	DataSet	DataSet
31	不一致アクターDataSet	入力シナリオアクターDataSetのうち，入力アクター定義DataSetと一致しない，アクターを保管するためのDataSet	DataSet	DataSet
32	不一致データDataSet	入力シナリオデータDataSetのうち，入力データ定義DataSetと一致しない，アクターを保管するためのDataSet	DataSet	DataSet
33	不一致画面DataSet	入力シナリオ画面DataSetのうち，入力画面定義DataSetと一致しない，アクターを保管するためのDataSet	DataSet	DataSet
34	不一致振る舞いDataSet	入力シナリオ振る舞いDataSetのうち，入力振る舞い定義DataSetと一致しない，アクターを保管するためのDataSet	DataSet	DataSet
35	表記ゆれアクターDataSet	不一致アクターDataSetにおいて，入力アクター定義DataSetに定義された用語の表記ゆれと考えられるアクターと修正用語候補を保管するための	DataSet	DataSet
36	表記ゆれデータDataSet	不一致データDataSetにおいて，入力データ定義DataSetに定義された用語の表記ゆれと考えられデータと修正用語候補を保管するためのDataSet	DataSet	DataSet
37	表記ゆれ画面DataSet	不一致画面DataSetにおいて，入力画面定義DataSetに定義された用語の表記ゆれと考えられる画面と修正用語候補を保管するためのDataSet	DataSet	DataSet
38	表記ゆれ振る舞いDataSet	不一致振る舞いDataSetにおいて，入力振る舞い定義DataSetに定義された用語の表記ゆれと考えられる振る舞いと修正用語候補を保管するための	DataSet	DataSet
39	未使用アクター定義DataSet	入力アクター定義DataSetにおいて，入力シナリオアクターDataSetに出現しないアクターを保管するためのDataSet	DataSet	DataSet
40	未使用データ定義DataSet	入力データ定義DataSetにおいて，入力シナリオデータDataSetに出現しないアクターを保管するためのDataSet	DataSet	DataSet
41	未使用画面定義DataSet	入力画面定義DataSetにおいて，入力シナリオ画面DataSetに出現しないアクターを保管するためのDataSet	DataSet	DataSet
42	未使用振る舞い定義DataSet	入力振る舞い定義DataSetにおいて，入力シナリオ振る舞いDataSetに出現しないアクターを保管するためのDataSet	DataSet	DataSet
43	出現NGワードDataSet	入力シナリオ形態素DatSetにおいて，NGワード辞書Datasetと一致する用語（NGワード）を保管するためのDataSet	DataSet	DataSet
44	未出現NGワードDataSet	NGワード辞書DataSetにおいて，シナリオ中に出現しなかったNGワード用語を保管するためのDataSet	DataSet	DataSet
45	一貫性検証ルール	要求仕様を検証するためのルールの集約クラス	なし	Logic
46	用語不一致検証ルール	定義漏れ検証ルールにより，定義漏れを特定し，表記ゆれ検出ルールによって表記ゆれを検出し，その結果，入力シナリオ形態素DataSetにおいて，入力＊定義DataSetと一致および一致しない用語を洗い出し，それぞれ，一致＊DataSetと，不一致＊DataSetに振り分けて定義し，表記ゆれ＊DatSetを定義する．＊は，アクター/データ/画面/振る舞いのうち，設定ファイルで指定されたもの．	なし	Logic
47	定義漏れ検証ルール	入力シナリオ形態素DataSetにおいて，入力＊定義DataSetと一致および一致しない用語を洗い出し，それぞれ，一致＊DataSetと，不一致＊DataSetに振り分けて定義する． ＊は，アクター/データ/画面/振る舞いのうち，設定ファイルで指定されたもの．	なし	Logic
48	用語定義完全一致検証ルール	定義漏れ検証ルールを用いて，定義漏れを特定し，入力＊定義DataSetにおいて，シナリオ中に出現しない用語を，未使用＊定義DataSetに定義する． ＊は，アクター/データ/画面/振る舞いのうち，設定ファイルで指定されたもの．	なし	Logic
49	NGワード検出ルール	入力シナリオ形態素DatSetにおいて，NGワード辞書Datasetと一致する用語を洗い出し，出現NGワードDataSetに保管する．また，NGワード辞書DataSetにおいて，シナリオ中に出現しなかったNGワードを，未出現NGワードDataSetに保管する	なし	Logic
50	表記ゆれ検出ルール	不一致＊DataSeにおいて，入力＊定義DataSetの表記ゆれに相当する用語を洗い出し，表記ゆれ＊DataSetに定義する． ＊は，アクター/データ/画面/振る舞いのうち，設定ファイルで指定されたもの．表記ゆれをしている語とは，次の2パターンとする - 定義された用語を部分的に含む複合語 - あらかじめ定義した異音同義語	なし	Logic

表 3-2-5　設計クラス一覧（3/3）

No.	クラス名	概要	親クラス/インタフェース	分類
51	設計要素検出ルール	アクター，データ，画面，振る舞いなどの設計要素を検出するためのルールの親クラス	なし	Logic
52	アクター検出ルール	アクター識別辞書を用いて，入力シナリオ形態素DataSetに定義された名詞句から，アクターに相当する用語を特定するためのロジック． 識別辞書に記述された用語そのもの，その用語が語尾に付与された複合語等を特定する．複合語のパターンは個別に定義する．	設計要素検出ルール	Logic
53	データ検出ルール	データ識別辞書を用いて，入力シナリオ形態素DataSetに定義された名詞句から，データに相当する用語を特定するためのロジック． 識別辞書に記述された用語そのもの，その用語が語尾に付与された複合語等を特定する．複合語のパターンは個別に定義する．	設計要素検出ルール	Logic
54	画面検出ルール	画面識別辞書を用いて，入力シナリオ形態素DataSetに定義された名詞句から，画面に相当する用語を特定するためのロジック． 識別辞書に記述された用語そのもの，その用語が語尾に付与された複合語等を特定する．複合語のパターンは個別に定義する．	設計要素検出ルール	Logic
55	振る舞い検出ルール	振る舞い識別辞書を用いて，入力シナリオ形態素DataSetに定義された名詞句から，振る舞いに相当する用語を特定するためのロジック． 識別辞書に記述された用語そのもの，その用語が語尾に付与された複合語等を特定する．複合語のパターンは個別に定義する．	設計要素検出ルール	Logic
56	アクター識別辞書	アクターを識別するための用語辞書．	なし	File
57	データ識別辞書	データを識別するための用語辞書．	なし	File
58	画面識別辞書	画面を識別するための用語辞書．	なし	File
59	振る舞い識別辞書	振る舞いを識別するための用語辞書．	なし	File

[実装クラス一覧]

表 3-2-6　実装クラス一覧（1/2）

No.	ファイル名	クラス／モジュール名		Step数	備考
		物理名	論理名		
1	verification-tool.rb	-	-	15	検証ツール起動用プログラム
2	action-detection-rules.rb	ActionDetectionRules	振る舞い検出ルール	75	設定ファイルにて指定された振る舞い識別ルールを順次実行する為のクラス
3	actor-detection-rules.rb	ActorDetectionRules	アクター検出ルール	75	設定ファイルにて指定されたアクター識別ルールを順次実行する為のクラス
4	appearance-ng-word-data-set.rb	AppearanceNGWordDataSet	出現NGワードDataSet	19	シナリオ中に出現したNGワードを保持するためのDataSetクラス
5	common-util.rb	CommonUtil	共通ユーティリティ	78	ツール内で使用する共通メソッドを定義したクラス
6	consistency-verification-rules.rb	ConsistencyVerificationRules	一貫性検証ルール	30	要求仕様を検証するためのルールの集約クラス
7	data-detection-rules.rb	DataDetectionRules	データ検出ルール	75	設定ファイルにて指定されたデータ識別ルールを順次実行する為のクラス
8	data-set.rb	DataSet	データセット	14	テンポラリな内部データの設定と取得の枠組みを定義した抽象クラス
9	design-element-detection-rules.rb	DesignElementDetectionRules	設計要素検出ルール	21	アクター、データ、画面、振る舞いなどの設計要素を検出するためのルールの親クラス
10	execute-morpheme-analysis.rb	ExecuteMorphemeAnalysis	形態素解析処理	10	形態素解析処理の実行の枠組みを定義した抽象クラス
11	execute-morpheme-analysis-mecab.rb	ExecuteMorphemeAnalysisMecab	形態素解析処理（"MeCab"を使用）	337	MeCabによろ形態素解析の実行を定義したクラス
12	execute-vrfctn.rb	ExecuteVrfctn	検証実行	53	検証を実行の枠組みを定義した抽象クラス
13	get-config-info.rb	GetConfigInfo	設定ファイル情報取得	264	設定ファイルの読み込みを行うクラス
14	get-design-element-definition-table-info.rb	GetDesignElementDefinitionTableInfo	入力設計要素定義表ファイル情報取得	71	入力設計要素定義表ファイルの読み込みを行うクラス
15	get-env-info.rb	GetEnvInfo	環境設定ファイル情報取得	168	環境設定ファイルの読み込みを行うクラス
16	get-identification-dictionary-info.rb	GetIdentificationDictionaryInfo	識別辞書情報取得	96	識別辞書ファイルの読み込みを行うクラス
17	get-ng-word-definition-table-info.rb	GetNGWordDefinitionTableInfo	NGワード定義表ファイル情報取得	60	NGワード定義表ファイルの読み込みを行うクラス
18	get-orthographic-variation-dictionary-info.rb	GetOrthographicVariationDictionaryInfo	表記ゆれ検出辞書情報取得	108	表記ゆれ検出辞書ファイルの読み込みを行うクラス
19	get-scenario-morpheme-info.rb	GetScenarioMorphemeInfo	入力シナリオ形態素解析情報取得	44	入力シナリオファイルを読み込み1行毎に形態素解析を実行し，入力シナリオ情報（テキスト名，文字数を保持）と入力シナリオ形態素情報を取得するためのクラス
20	input-action-definition-data-set.rb	InputActionDefinitionDataSet	入力振る舞い定義DataSet	19	入力設計要素定義表から取得した"振る舞い"を保持するためのDataSetクラス
21	input-actor-definition-data-set.rb	InputActorDefinitionDataSet	入力アクター定義DataSet	19	入力設計要素定義表から取得した"アクター"を保持するためのDataSetクラス
22	input-data-definition-data-set.rb	InputDataDefinitionDataSet	入力データ定義DataSet	19	入力設計要素定義表から取得した"データ"を保持するためのDataSetクラス
23	input-scenario-action-data-set.rb	InputScenarioActionDataSet	入力シナリオ振る舞いDataSet	19	入力シナリオ内の"振る舞い"を保管するためのDataSetクラス
24	input-scenario-actor-data-set.rb	InputScenarioActorDataSet	入力シナリオアクターDataSet	19	入力シナリオ内の"アクター"を保管するためのDataSetクラス
25	input-scenario-data-data-set.rb	InputScenarioDataDataSet	入力シナリオデータDataSet	19	入力シナリオ内の"データ"を保管するためのDataSetクラス
26	input-scenario-info-data-set.rb	InputScenarioInfoDataSet	入力シナリオ情報DataSet	19	読み込んだ入力シナリオのファイル名と文字数を保管するためのDataSetクラス
27	input-scenario-morpheme-data-set.rb	InputScenarioMorphemeDataSet	入力シナリオ形態素DataSet	19	形態素解析エンジンによって、入力されたシナリオを解析した結果を保管するためのDataSetクラス
28	input-scenario-screen-data-set.rb	InputScenarioScreenDataSet	入力シナリオ画面DataSet	19	入力シナリオ内の"画面"を保管するためのDataSetクラス
29	input-screen-definition-data-set.rb	InputScreenDefinitionDataSet	入力画面定義DataSet	19	入力設計要素定義表から取得した"画面"を保持するためのDataSetクラス
30	match-action-data-set.rb	MatchActionDataSet	一致振る舞いDataSet	19	入力シナリオ内の"振る舞い"で入力振る舞い定義に存在する振る舞いを保管するためのDataSetクラス
31	match-actor-data-set.rb	MatchActorDataSet	一致アクターDataSet	19	入力シナリオ内の"アクター"で入力アクター定義に存在するアクターを保管するためのDataSetクラス

表 3-2-7　実装クラス一覧（2/2）

No.	ファイル名	クラス／モジュール名		Step数	備考
		物理名	論理名		
32	match-data-data-set.rb	MatchDataDataSet	一致データDataSet	19	入力シナリオ内の"データ"で入力データ定義に存在するデータを保管するためのDataSetクラス
33	match-screen-data-set.rb	MatchScreenDataSet	一致画面DataSet	19	入力シナリオ内の"画面"で入力画面定義に存在する画面を保管するためのDataSetクラス
34	message-util.rb	MessageUtil	メッセージ出力ユーティリティ	14	メッセージの出力に必要なメソッドを定義したクラス
35	mismatch-action-data-set.rb	MismatchActionDataSet	不一致振る舞いDataSet	19	入力シナリオ内の"振る舞い"で入力振る舞い定義に存在しない振る舞いを保管するためのDataSetクラス
36	mismatch-actor-data-set.rb	MismatchActorDataSet	不一致アクターDataSet	19	入力シナリオ内の"アクター"で入力アクター定義に存在しないアクターを保管するためのDataSetクラス
37	mismatch-data-data-set.rb	MismatchDataDataSet	不一致データDataSet	19	入力シナリオ内の"データ"で入力データ定義に存在しないデータを保管するためのDataSetクラス
38	mismatch-screen-data-set.rb	MismatchScreenDataSet	不一致画面DataSet	19	入力シナリオ内の"画面"で入力画面定義に存在しない画面を保管するためのDataSetクラス
39	ng-word-definition-table-data-set.rb	NGWordDefinitionTableDataSet	NGワード定義表DataSet	19	NGワードと置換候補ワードを保持するためのDataSetクラス
40	not-appearance-ng-word-data-set.rb	NotAppearanceNGWordDataSet	未出現NGワードDataSet	19	シナリオ中に出現しなかったNGワードを保持するためのDataSetクラス
41	orthographic-variation-action-data-set.rb	OrthographicVariationActionDataSet	表記ゆれ振る舞いDataSet	19	表記ゆれと考えられる振る舞いと修正用語候補を保管するためのDataSetクラス
42	orthographic-variation-actor-data-set.rb	OrthographicVariationActorDataSet	表記ゆれアクターDataSet	19	表記ゆれと考えられるアクターと修正用語候補を保管するためのDataSetクラス
43	orthographic-variation-data-data-set.rb	OrthographicVariationDataDataSet	表記ゆれデータDataSet	19	表記ゆれと考えられるデータと修正用語候補を保管するためのDataSetクラス
44	orthographic-variation-screen-data-set.rb	OrthographicVariationScreenDataSet	表記ゆれ画面DataSet	19	表記ゆれと考えられる画面と修正用語候補を保管するためのDataSetクラス
45	output-verification-result.rb	OutputVerificationResult	検証結果出力処理	668	検証レポートの出力、補正後シナリオの出力を定義したクラス
46	req-vrfctn.rb	ReqVrfctn	要求仕様の検証	239	要求仕様の検証の方法を定義したクラス
47	rules-loader.rb	RulesLoader	識別ルールのクラスを読み込む	21	設定ファイルにて指定された識別ルールを動的に読み込むメソッドを定義したクラス
48	screen-detection-rules.rb	ScreenDetectionRules	画面検出ルール	75	設定ファイルにて指定された画面識別ルールを順次実行する為のクラス
49	struct-aggregate.rb	－	－	67	システムで使用する全ての構造体を定義したファイル
50	unused-action-definition-data-set.rb	UnusedActionDefinitionDataSet	未使用振る舞い定義DataSet	19	入力振る舞い定義に存在するが、入力シナリオに存在しない振る舞いを保管するためのDataSetクラス
51	unused-actor-definition-data-set.rb	UnusedActorDefinitionDataSet	未使用アクター定義DataSet	19	入力アクター定義に存在するが、入力シナリオに存在しないアクターを保管するためのDataSetクラス
52	unused-data-definition-data-set.rb	UnusedDataDefinitionDataSet	未使用データ定義DataSet	19	入力データ定義に存在するが、入力シナリオに存在しないデータを保管するためのDataSetクラス
53	unused-screen-definition-data-set.rb	UnusedScreenDefinitionDataSet	未使用画面定義DataSet	19	入力画面定義に存在するが、入力シナリオに存在しない画面を保管するためのDataSetクラス
54	verification-method-data-set.rb	VerificationMethodDataSet	検証方法DataSet	19	設定ファイルに定義された検証方法を保持するためのDataSetクラス
55	xml-util.rb	XMLUtil	XMLユーティリティ	80	xmlファイルの読み込みに必要なメソッドを定義したクラス
56	definition-omission-verification-rules.rb	DefinitionOmissionVerificationRules	定義漏れ検証ルール	79	定義漏れ検証のルールを定めたモジュール
57	ng-word-verification-rules.rb	NGWordVerificationRules	NGワード検証ルール	50	NGワード検証のルールを定めたモジュール
58	orthographic-variation-verification-rules.rb	OrthographicVariationVerificationRules	表記ゆれ検証ルール	86	表記ゆれ検証のルールを定めたモジュール
59	perfect-matching-verification-rules.rb	PerfectMatchingVerificationRules	完全一致検証ルール	61	完全一致検証のルールを定めたモジュール
60	action-rules-01.rb	ActionRules01	振る舞いルールその１	195	振る舞いを抽出するクラス
61	actor-rules-01.rb	ActorRules01	アクタールールその１	199	アクターを抽出するクラス
62	data-rules-01.rb	DataRules01	データルールその１	199	データを抽出するクラス
63	screen-rules-01.rb	ScreenRules01	画面ルールその１	199	画面を抽出するクラス
			ステップ数合計	4,396	

［テスト実施］

プログラム実装後，以下の実仕様書を用いてルール分析，テストデータ作成，ツールテストを実施した．なお，これらの仕様書は Web 上等で一般公開されている仕様書である．

(a) 某市公式ウェブサイトリニューアル事業要求仕様書

(b) 某新制度全国総合システム構築・運用等業務調達仕様書

図 3-2-13 は，シナリオの一貫性検証支援ツールによる検証例を示したものである．ツールには，シナリオとアクター定義表を入力すると，ツールが検証を行い，検証レポートを出力する．ここでは，シナリオ中に，アクター定義で定義されていない CMS 利用者，ウェブサイト閲覧者が使われていることを，検証レポートで指摘している．

テストにおいては，人手によりあらかじめ，仕様書内のアクター，データ，画面，振る舞用語の抽出し，さらに，定義漏れ，用語不一致，用語完全一致，NG ワードのそれぞれの検証結果を指摘しておく．ツールから出力された検証レポートと，あらかじめ用意していた人手による指摘事項を突合せ，ツールの検証結果の妥当性を評価した．なお，評価結果全体については，3.3.2(2)においてまとめて示す．

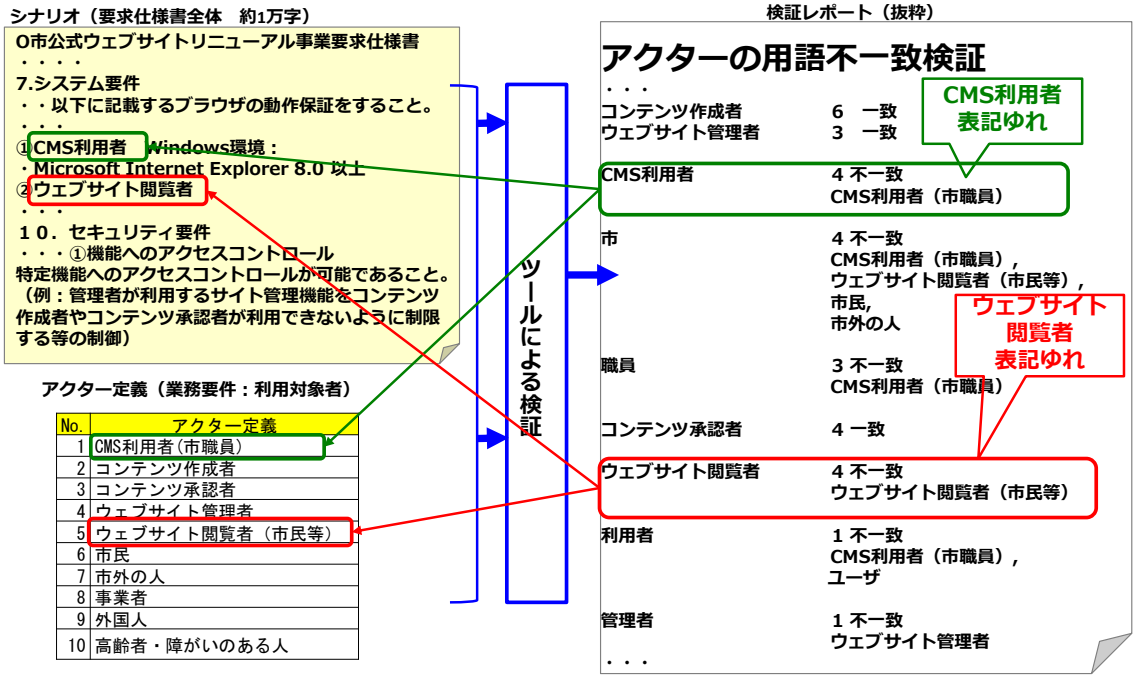


図 3-2-13　シナリオの一貫性検証支援ツールの評価事例

3.2.3 発生した問題および今後の展望

(1) 発生した問題

シナリオの一貫性検証支援ツール（反復 1）の開発が，9 月末時点で完了しておらず，（反復 2）の開発着手が研究計画立案時の予定に対して，2 週間ほど遅れた．（反復 1）の開発遅れに起因して，ツールの開発完了が 12 月にずれ込んだ．したがって，インタビューを行った有識者へのツールの評価の時期が，年末年始にずれ込む見込みで，スケジュール調整が難航するリスクがあった．

対策として，（反復 1）の受け入れが完了次第，企業側への評価のインタビューの日程調整に着手した．また，実案件の仕様書を，ツールの受け入れテストにて用い，本結果を有識者にお見せすることで，企業側での評価の負荷を軽減するなどして評価方法を工夫した．

シナリオの一貫性検証支援ツールのルール実行エンジンにおいて利用した MeCab のデフォルトの特性により，入力した要求仕様書内において名詞の用語抽出結果に不適切なデータが存在し，したがって検証レポートに誤検出が発生した．現状の MeCab の特性として，誤検出となる部分については，ユーザマニュアルにおいて明示することとした．

(2) 今後の展望

自然言語で記述された仕様書の分析において，MeCab による形態素解析以外の構文解析による解析を加味すること，ユーザが指摘事項に関して除外設定をすることによる誤検出の排除支援をすることが必要と考えらえる．

3.3 研究課題 3「シナリオの一貫性検証の支援ツールの評価（評価）」

3.3.1 当初の想定

(1) 研究内容

研究課題 1「シナリオの一貫性検証知識の形式知化（形式知化）」で開発した検証知識を組み込み，研究課題 2 シナリオの一貫性検証の支援ツールの開発（ツール開発）」で開発したツールを，研究課題 1 でインタビューした有識者およ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．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用いて評価し，評価結果をツール等に反映させる．

(2) 想定問題と対応策

[想定問題]

定義したシナリオの整合性の検証ルールおよびそれら知識に基づくシナリオの一貫性検証支援ツールが，現実の要求定義業務において，どれほどの効果をもたらすのかが曖昧であると，実務担当はこれらルールならびにツールを実際の要求定義の場面でスムーズに導入することが困難になり，本研究の成果が現実の要求定義の場面で活用できないものになる．

[対応策]

検証ルールおよびツールの評価においては，企業において実際に開発に用いられた仕様書も適用し評価することで，具体的な適用方法や，適用することによる効果を明らかにする．

3.3.2 研究プロセスと成果

(1) 研究プロセス

次の①～③のプロセスにより研究を実施した．

①評価項目定義

検証ルールおよびツールの評価の実施にあたり，実際の開発で用いられた仕様書の評価を考慮の上，評価項目を定義した．

②有識者評価

研究課題 1 でインタビューを実施した有識者を中心に，①で定めた評価項目に基づき，インタビューし，ツール評価を実施した．

③評価と成果まとめ

②の有識者による評価結果をまとめ，ツールに対する改善事項，ツールの利用に関するユースケースを整理した．また，辞書，検証ルール，ユーザマニュアルの改善を行い，ツール一式をまとめた．

(2) 具体的な研究成果の内容

①評価項目定義

1)評価項目

ツール評価では，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することを定義した．ツールの検証結果の妥当性の評価においては，以下の実際の案件での要求仕様書／調達仕様書を対象として，上述したツールの受け入れテストにて実施したテスト実施方法にて，設計要素：アクター，データ，画面，振る舞いを対象に，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実施した結果を作成した．

(a) 某市公式ウェブサイトリニューアル事業要求仕様書

(b) 某新制度全国総合システム構築・運用等業務調達仕様書

これらの実仕様書を用いてルール分析，テストデータ作成，ツールテストを実施した．それぞれの実施結果を表 3-3-1，表 3-3-2 にまとめた．また，表 3-3-3 には，検証種類別の平均値をまとめた．

表 3-3-1　仕様書（a）に対する設計要素別ツール検証の再現率および適合率

No.	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	一致した指摘数	再現率（%）	適合率（%）
1	定義漏れ検証	Actor	62	63	9	10	53	85	84
2		Data	63	64	0	1	63	100	98
3		Screen	30	31	2	3	28	93	90
4		Action	550	659	8	117	542	99	82
5	表記ゆれ検証	Actor	62	63	9	10	53	85	84
6		Data	63	64	0	1	63	100	98
7		Screen	30	31	2	3	28	93	90
8		Action	550	659	8	117	542	99	82
9	完全一致検証	Actor	62	65	9	12	53	85	82
10		Data	63	64	0	1	63	100	98
11		Screen	30	31	2	3	28	93	90
12		Action	550	659	8	117	542	99	82
13	NGワード検証	_	21	21	0	0	21	100	100

表 3-3-2 仕様書（b）に対する設計要素別ツール検証の再現率および適合率

No.	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	一致した指摘数	再現率（%）	適合率（%）
1	定義漏れ検証	Actor	139	147	13	21	126	91	86
2		Data	209	210	12	13	197	94	94
3		Screen	15	15	0	0	15	100	100
4		Action	1,370	1,644	13	287	1,357	99	83
5	表記ゆれ検証	Actor	139	147	13	21	126	91	86
6		Data	209	210	12	13	197	94	94
7		Screen	15	15	0	0	15	100	100
8		Action	1,370	1,644	13	287	1,357	99	83
9	完全一致検証	Actor	139	147	13	21	126	91	86
10		Data	253	254	12	13	241	95	95
11		Screen	70	70	0	0	70	100	100
12		Action	1,370	1,644	13	287	1,357	99	83
13	NGワード検証	–	21	21	0	0	21	100	100

表 3-3-3 仕様書（a）および（b）に対するツール検証の再現率および適合率まとめ

No.	仕様書	検証名	文字数	再現率平均（%）	適合率平均（%）
1	(a)	定義漏れ検証	17,635	94	89
2		表記ゆれ検証		94	89
3		完全一致検証		94	88
4	(b)	定義漏れ検証	66,048	96	91
5		表記ゆれ検証		96	91
6		完全一致検証		96	91

表 3-3-1，表 3-3-2 は，それぞれ仕様書(a)，(b) に対して，定義漏れ検証，用語不一致検証，用語完全一致検証，NG ワード検証からなる検証の種類別に，設計要素アクター(Actor)，データ (Data)，画面 (Screen)，振る舞い (Action) における検証結果を示したものである．なお，NG ワード検証は，設計要素別には実施していない．検証は，人手による指摘，ツールによる指摘数，指摘漏れ数 (人手による指摘があったもののツールで指摘できなかった数)，誤指摘数 (ツールが指摘したものの，人手では指摘していない不適切な指摘数)，人手とツールによる一致した指摘数をカウントし，その上で再現率，適合率を算出した．

なお，再現率，適合率は次の数式にて算出した．

再現率：(一致した指摘数) ÷ (人手の指摘) ＊100
適合率：(一致した指摘) ÷ (システムによる指摘) ＊100

表 3-3-3 に示すように，各仕様書に対して，3 つの検証については，90%以上の再現率，88%以上の適合率となっている．なお，指摘できなかった代表的な事項としては，次が考えられる．

[指摘できなかった主な用語]

アクターとしては，「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」について，ツールは「アクター」用語として特定できず，その結果，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．

「高齢者・障がいのある人」は，ツールは「高齢者」，「人」はアクターとして認識するものの，「障がいのある」という，名詞句「人」を修飾する表現については，名詞と組み合わせて一つの名詞として認識してはいないため，特定していない．同様にして，アクターを識別する「課」や「者」についても，その用語が修飾されている場合には，全体としてアクター用語として特定していない．また，「コンテンツの所有部署」等の「部署」を修飾する用語については，「部署」がアクターを識別する辞書として登録していないため，アクター用語として特定していない．「部署」をアクター用語の識別辞書に登録すれば，再現率は向上すると考えられる．

画面としては，「トップページ以外のページ」なる画面名が特定できなかった．これは，「ページ」が画面を識別するための辞書に登録されているものの，上述したページを修飾する表現を含めて「ページ」として認識しなかったためである．

データに関する検証結果の指摘漏れもアクター，画面と同様のパターンで発生している．本ツールは「XML 形式のファイル」，「CSV 形式のファイル」，「登録するための情報」，「文字コードの情報」はデータ用語として認識していない．加えて，MeCab の特性により，「内訳書」が「内」と「訳書」に分割して分かち書きされるため，「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せず，結果として，定義漏れ検証，用語不一致検証，用語完全一致検証において指摘漏れとなった．これは，MeCab の特性に起因するものであり，現状では，ツール側での改善は困難であるため，ユーザマニュアルにおいて，識別困難であることを補足説明した．

振る舞いとしては，「ページの変更・削除」，「画像の代替テキストの有無のチェック」，「機種依存文字のチェック」，「ページ本文の作成」，「貼り付け」，「一括メンテナンス」が特定できていない．サ変接続名詞は振る舞いと特定するルールを設定してはいるものの，上述したように，名詞を修飾する表現を含めての特定が困難なため，現状では，これらの用語は振る舞いとして特定していない．

[ツールの誤指摘となった用語]

アクターに関しては，前述したアクター用語に相当する名詞が修飾された表現で記述されている場合，ツールは全体としてアクター用語と特定が困難となり，部分的に一致した箇

所をアクターとして誤指摘した．例えば，「市外の人」，「高齢者」，「課」，「者」などを誤って定義漏れ用語として指摘した．

画面に関してもアクターと同様の理由で，「トップページ以外のページ」を特定せずに，その部分である「ページ」だけを特定する結果になった．

データについても，アクター，画面と同様であり，部分一致での用語の誤指摘が散見された．例えば，「訳書」，「ための情報」，「コードの情報」，「形式のファイル」などが相当する．

本ツールの振る舞い用語の識別ルールでは，サ変接続名詞を振る舞いとして特定することにしている．よって，システム化の目的や機能仕様以外の仕様説明の箇所に記述された様々なサ変接続名詞を，「振る舞い」用語としてツールが抽出している．結果として，バナー広告，メール，携帯電話，RAID 構成，セキュリティ対策などの周辺機能やサービスの説明仕様から抽出された用語を，振る舞いとして機能実装されるべき要素であるとして，誤指摘している．本件については，ツールに入力する範囲を機能仕様に限定することにより，このような振る舞いの誤指摘は防止可能と考えられる．

②有識者評価

1)有識者インタビュー結果

東芝ソリューション株式会社では 2 部門，日本電気株式会社，株式会社 NTT データ，株式会社クニエはそれぞれ 1 部門のソリューション／システム開発の要求定義の経験豊富な有識者に，ツールの実現結果に対するインタビューを行った．ツールの開発前のインタビューの結果を踏まえて，開発したツールの有効性や操作性，適用可能性，ツールへの期待について意見を伺った．なお，インタビューに際して，ツール実行と検証レポート提示の実演と，上述した実仕様書の検証結果を提示した．インタビュー結果を以下にまとめる．

[ツールの妥当性]

- 仕様書事例 2 点（某市公式ウェブサイトリニューアル事業要求仕様書（1 万 7 千字），某新制度全国総合システム構築・運用等業務調達仕様書（約 6 万 8 千字））を入力として，定義漏れ検証，用語不一致検証，用語定義完全一致検証，NG ワード検証を実施した結果については，おおむね妥当であると考えられる．また，検証の処理速度（数秒程度）は，実運用上の妨げにはならないと考えられる．
- 開発部門において実際に利用することを想定すると，検証レポートの見せ方（検証結果データのビジュアル化，定量化，基準値を設定した品質評価，評価結果のサマリの提示），ツールの実行環境（インタフェース）の工夫が必要である．

[ツールの有効性]

- ツールの評価実験で利用した「某新制度全国総合システム構築・運用等業務調達仕様書」は，実際の開発時に熟読しているものの，今回出力された検証レポートには，さまざまな用語が抽出されており，新たな気づきを得ることができた．しかし，同レポートには，設計とは比較的関係性の低い用語も同様に抽出されているため，検証レポートの結果の仕分けが必要である．
- 仕様書中に「○○（△△）は◇◇（○○）の情報をとりまとめたうえで確認する」とい

う表現があるが，これは，「〇〇は◇◇の情報をとりまとめたうえで確認し」，「△△は〇〇の情報を取りまとめたうえで確認する」，という２つの文章を，1 つの文章で表現している．文中の括弧「()」は，直前の用語の別名を表現するケースだけではない．記述した側に固有の意図が盛り込まれており，機械的な解釈は簡単ではない．このような文章を曖昧表現として指摘することができるのは有効である．

- 用語が不一致になる情報が数百行レベルで一覧化されて提示され，それらの中に誤指摘が含まれるとなると，実務者にとり，全ての「不一致」項目を改善するモチベーションが低下する．不一致の指摘に対しても，優先度を設定して，過去の経験から手戻りコストが高い箇所や品質に重要な影響を与えるような不一致項目を指摘し，改善を促せると有用である．
- 今後の適用評価については，ツールが指摘できた事柄で，実際の開発では確認が曖昧だった箇所について，一貫性を指摘し修正することのメリットを整理することが重要である．例えば，曖昧な用語がそのままであったために確認のため作業が１週間止まってしまったなどのケースがあれば，これが解決できることが示されれば有効性を説明することが可能である．
- 「交付申請情報」が，「交付金申請・決定情報」のことではないか，という指摘に対して，例えば，エンティティ定義の「交付金申請・決定情報」は，２つの情報から構成されていて，「交付金申請情報」と「交付金決定情報」とすべきではないか，といった基本設計以降のボリュームに変更が加わるようなケースに対して提案ができると有効である．

［操作性や適用時の工夫］

- 入力した要求仕様書から，用語を検出するためのルールでは，名詞と名詞を助詞「の」で接続している場合に複合語として抽出している．このため「〇〇情報の登録」という表現の場合，これを振る舞い用語として抽出することができない．これを振る舞い用語として認識するには「〇〇情報を登録する」というように，入力した仕様書を改める必要がでてくる．しかし，「〇〇情報の登録と△△情報の登録を行う」という記述の場合では，単に「の」を「を」に置き換えただけでは，意味が通じなくなる．助詞でつながれた名詞の複合語の抽出や，体言止めされた単語の扱いなど，辞書，検出ルールの調整の積み重ねが必要である．
- 要求仕様書では，発注者の「課題」について記述する領域と，開発するシステムの「インフラ」について言及する領域では，用語の使い方が異なると考えらえる．同じアクターやデータの辞書でも言及する対象領域の違いにより，辞書を使い分ける工夫が必要と考えられる．
- 本ツールの利用用途としては，作成した提案書の検証，分担して作成した設計書（統合時），機能仕様書と基本設計書などの工程別の成果物間での表記ゆれの指摘が考えられる．
- アクターなどが明確に定義されていない仕様書に対しては，仕様書内にこれらの用語があることを示すことを目的とした使い方ができる．

[ツールへの期待]

- ツールへの入力ファイルがテキストファイルに限定されている点は拡張する必要がある．一般的に，仕様書は，Pdf，Microsoft Word®，Microsoft Excel® ，Microsoft PowerPoint®で記述されているため，利用者側がテキストファイルに変換するのではなく，ツール側でテキストファイルに変換できると望ましい．
- 検証レポートで指摘された未定義の用語には，仕様書内で設計要素として定義する必要のない用語が含まれている可能性があるため，そのような用語は「除外」して再検証がかけられると望ましい．除外した用語リストを考慮して仕様検証を実行し，指摘事項がクリアされたら，次の工程に進めるなどの運用が実現できると有効である．
- 検証結果に点数をつけて評価できる仕組みがあるとより使いやすいと考えられる．例えば，一定の合格ラインを設け，そこに達するまで，改善と検証を繰り返すことで，仕様書の品質をあげる仕組みが望ましい
- 検証レポートでは，未定義／表記ゆれの用語の存在と出現回数を提示してはいるが，入力する仕様書中の出現箇所までは提示していない．指摘した用語が入力した仕様書中の何行目に出現しているかなどを明示できると，さらにツールの使用性が向上すると考えられる．
- NG ワード検証結果に対しては，指摘された用語の周辺文章が検証レポートに出力されていると，本文の修正が妥当かどうかを効率的に判断しやすいと考えられる．また，NG ワードの逆となる「重要ワード」の検出機能があると望ましい．
- 検証レポートによる定義漏れや表記ゆれ用語の指摘結果に対する対応履歴を蓄積する機能があると望ましい．すなわち，次の事柄を実施したかどうかなどを記録する：指摘結果の採用／不採用，修正の実施有無，修正箇所（仕様書本体，辞書，アクター等の定義表）さらに，プロジェクト内で対応履歴を学習し検証ルールを拡張する機能があると望ましい．

③評価と成果まとめ

1)ツール一式

②の有識者による評価結果をまとめ，ツールに対する改善事項，ツールの利用に関するユースケースを整理した．また，評価結果を反映させて，辞書，検証ルール，ユーザマニュアルの改善を行い，ツール一式としてとりまとめた．改善された辞書，検証ルールを含むツール一式の内容については前述した 3.2.2(2)を参照されたい．また，ツールの利用に関するユースケースについては，4.2.1 研究成果の産業界への展開に示す．ツールの利用に関するユースケースは，ユーザマニュアルにも記述した．

3.3.3 発生した問題および今後の展望

(1) 発生した問題

仕様書事例 2 点（某市公式ウェブサイトリニューアル事業要求仕様書（1 万 7 千字），某新制度全国総合システム構築・運用等業務調達仕様書（約 6 万 8 千字）を入力として，定義漏れ検証，用語不一致検証，用語定義完全一致検証，NG ワード検証を実施した結果については，おおむね妥当であるとの評価を得られた．ツールの適合率は，80%以上を達成できた

ものの，100%には至っていない．今回適用対象としたような実際の仕様書を本ツールで検証し，F1～F5 までのすべての機能を同時実行すると，検証レポートは数十ページに渡り，誤指摘についても合計数十件に渡るケースがある．長文で誤指摘が含まれる検証レポートを読み解いて仕様書を改善するモチベーションが低下するリスクが高い．実適用するには，検証レポートの簡略化，可視化，ユーザインタフェース定義が必要である．

(2) 今後の展望

本ツールの適合率を向上させ，誤指摘を防止し，利用者がツールの検証結果に基づいて効率的に仕様の改善が行えるように，形態素解析エンジンの条件や対象組織に固有の特性などから，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ていく技術開発に取り組むことが重要と考えられる．

また，検証レポートについては，現状，数十ページに及ぶ定義漏れや用語不一致結果のリストから利用者が状況を把握し，改善作業に直ちに取り掛かれるようにするために，評価結果を図表等で分かりやすくすることおよび，網羅率や定義率などの観点で定量化評価した成績書にあたるサマリレポートを提示することを進める．そのために有効な評価項目，定量評価の方法を明らかにすることが今後の課題である．

4 考察

4.1 研究による効果や問題点等

[研究による効果]

第一に，検証ルール，辞書，ツールを活用することで，若手技術者でも高品質なシナリオを効率的に作成することが可能になり，産業界全体で，要求定義技術のレベルアップが期待できると考えられる．第二に，あらかじめ基本的な検証ルールと辞書を提供するため，要求仕様の検証の経験のない組織でも，スムーズにシナリオ検証に取り組むことができる．また，第三に，検証ルール，辞書は，検証支援ツールに対して独立した構造となっているため，対象組織の経験に基づいて得られた独自の検証ノウハウを，ルールおよび辞書として容易に拡張でき，検証ノウハウそのものを組織の資産として蓄積することができる．

類似の研究成果が「曖昧さにつながりやすいキーワード」を中心として検証しているのに対して，我々の提案した研究成果では，そのようなNGワードに加えて，仕様書の中身に相当する設計要素の定義漏れや表記ゆれを検証可能な点で特徴的である．すなわち，システム・ソフトウェア開発で重要な設計要素となる，アクター，データ，画面，振る舞いにフォーカスした一貫性検証が行える点，これら設計要素の識別のための知識を組織別にカスタマイズし拡張が可能になっている点，検証レポートにより，仕様書に出現する用語分析結果が蓄積されるので用語の出現回数や状況から，仕様書の理解や確認が可能な点が差別化要素として有効と考えられる．

[設定した研究目標をどこまで達成できたのか]

本研究では，シナリオの整合性の検証知識とシナリオの一貫性検証支援ツールを提供することで，業界全体で要求定義のノウハウを共有し，我が国における要求定義技術のレベルアップを目指し，そして，魅力あるソフトウェア製品の要求定義の効率化，高品質化により，ソフトウェア開発の国際競争力向上に貢献することを目的としている．本研究では，各企業の有識者の検証のノウハウを洗い出し，設計要素の整合性のチェックと，NG ワードの指摘の観点から検証ルールを形式知化し，シナリオの一貫性検証支援ツールを開発した．仕様書事例2点（某市公式ウェブサイトリニューアル事業要求仕様書（1万7千字），某新制度全国総合システム構築・運用等業務調達仕様書（約6万8千字））を入力として，定義漏れ検証，用語不一致検証，用語定義完全一致検証，NG ワード検証を実施した結果から，実際の仕様書の検証に対して一定のレベルに達した．すなわち，初級の技術者が本ツールを活用することにより，設計要素の整合性のチェックと NG ワードの指摘に関する検証については，ベテラン技術者のノウハウを継承し，検証のスキルアップにつながると考えられる．

[達成できなかったあるいは残っている目標とその理由]

シナリオ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．要求定義の高品質化には，本ツールによる検証結果の適合率をさらに向上させるなど研究の余地が残されている．

［新たに見出された課題・問題］

本ツールの適合率を向上させ，誤指摘を防止し，利用者がツールの検証結果に基づいて効率的に仕様の改善が行えるように，形態素解析エンジンの条件や対象組織に固有の特性などから，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ていく技術開発に取り組むことが重要と考えられる．

また，検証レポートについては，現状，数十ページに及ぶ定義漏れや用語不一致結果のリストから，利用者が状況を把握し，改善作業に直ちに取り掛かれるようにするために，評価結果を図表等で分かりやすくすることおよび，網羅率や定義率などの観点で定量化評価した成績書にあたるサマリレポートを提示することを進める．そのために有効な評価項目，定量評価の方法を明らかにすることが今後の課題である．

4.2 産業界への展開と今後の研究の進め方

4.2.1 研究成果の産業界への展開

本研究の成果は，主にはエンタープライズ系システム／サービスの企画立案，要求定義工程において図 4-1 に示すようなシーンでの活用を想定している．活用のシーンとは，初級技術者による要求仕様書や提案書のセルフチェック，担当者が作成した仕様書の管理者による品質チェック，検証レポートを用いた要求仕様書のレビューなどである．さらには，組織で蓄積した検証ルールや辞書を共有することで，若手技術者への知識継承や人材育成に活用が可能である．なお，ツール評価インタビュー先企業の一部にツールを提供した．

図 4-1　成果の活用のシーン

3.3.2(2)で示した通り，実際の仕様書 2 点を入力として，定義漏れ検証，用語不一致検証，用語定義完全一致検証，NG ワード検証を実施した結果については，おおむね妥当であるとの評価を得た．とくに，検証レポートの抽出用語一覧を確認することで，新たな気づきとして，仕様書への理解が加速した，という有用というコメントがあった．このことから，本ツールは仕様書中の設計要素の整合性のチェックに加えて，仕様書の理解や設計要素の候補を抽出するための支援ツールとしての活用も有効と考えられる．

加えて，次のようなシーンでの使われ方が想定可能であり，このような使われ方を想定した普及展開を進めていく．

[シナリオの一貫性検証支援ツールの利用ユースケース]

- 操作仕様書や製品取扱い説明書の品質向上
  製品の操作パネルなどのイメージに対して自然言語で操作手順を説明するドキュメントに適用して，利用している用語の一貫性検証，NG ワード検証を行う．

- 要求定義書と基本設計書間の比較
  それぞれに対して設計要素を抽出し差分を比較することで，追跡可能性や一貫性の検証に活用する．

・ プロダクトライン型開発のモデル間の比較
それぞれに使用されている設計要素を抽出し結果を比較することで，利用していないデータや登場しないアクター等の矛盾を指摘する．例）モデル A で用いるセンサーは「X センサー」であり，モデル A'で用いるセンサー「Y センサー」である．モデル A'の仕様書をモデル A の仕様書を再利用して作成した場合に，モデル A'の仕様書内に「X センサー」の記述が残らぬように，デバイス「センサー」の用語不一致検証を行う．

・ 業務マニュアルからアクター用語抽出
要求仕様になる前の様々な業務文書から，ツールを用いてアクター用語を抽出する．抽出したアクターと識別できる用語を，新システム提案時に，ステークホルダ定義への入力ソースとして活用する．

4.2.2 今後の研究の進め方

企業への普及展開活動を進め，学習機能を付与すること，検証レポートにおいて検証結果の定量化表示を行うことに取り組み，本ツール支援による仕様書の検証の有効性や妥当性を評価し，ツールの改善と利用ノウハウの蓄積を継続していく．

現状，本ツールの適合率は 80%以上にはなるものの，形態素解析エンジンの条件や対象組織に固有の特性などから，指摘対象からは除外すべき用語が存在する．指摘結果からそのような指摘不要の情報を学習し，ツールによる検証を対象組織用に進化させていく技術開発に取り組む．第 1 ステップとしては，利用者が，検証レポートの結果から，除外すべき指摘事項を選択し，選択した結果を取り込んで，2 回目以降の検証を実施するような利用者からの除外情報に基づく検証の技術開発が有効と考えられる．第 2 ステップとしては，そのような除外情報を積み重ねた上で，除外するパターンを学習し，検証ルールへの除外ルールとして形式知化を自動化する仕組みの研究開発が重要と考えている．

また，現状の検証レポートは数十ページに渡る．そこで，定義漏れや用語不一致結果のリストから利用者が状況を把握し，改善作業に直ちに取り掛かれるようにするために，評価結果を図表等で分かりやすくすること，網羅率や定義率などの観点で定量評価した成績書を提示することに取り組む．図 4-2 に，検証レポートの可視化例を示す．

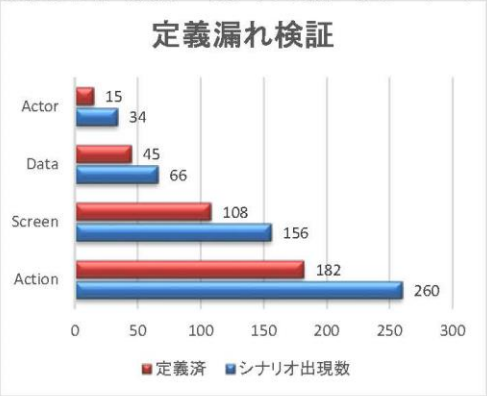
図 4-2 は，ある要求仕様書の検証内容を 1 枚のレポートで要約した結果を示している．これは，評価サマリ，総合判定，定義漏れ検証結果のグラフ，用語不一致検証の評価，NG ワード利用率，主な NG ワードの利用状況，主な定義漏れや表記ゆれ用語のサマリからなる．検証した仕様に出現する各設計要素の用語一覧のリストを含む検証レポートの詳細は，従来の検証レポートの結果を用いることを想定している．このように，詳細な検証レポートを分析する前に，定義漏れや表記ゆれの出現数が基準よりも多いか少ないかを把握可能な総合評価や，当該仕様書に使われている主な NG ワードや定義漏れ・表記ゆれ用語を提示することで，仕様書の品質の傾向を素早く把握し，利用者自身が，仕様書の品質を理解し，改善するモチベーションを高めることをねらっている．そのために有効な評価項目，定量評価の方法を明らかにすることが今後の重要課題の一つである．

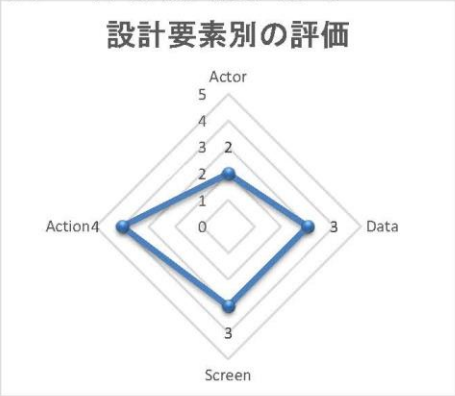
要求仕様の一貫性検証レポート

案件名	A社様向け＊＊＊情報管理システムリニューアル事業	実施日付
対象ドキュメント名	＊＊＊情報管理システム要求仕様書	2016/1/29
対象ドキュメントID	REQ-20151030153-01	
対象ドキュメントファイル名	REQ-20151030153-01.txt	

評価サマリ	総合判定
・アクター，データ，画面，アクション共に，半数以上が未定義項目になっています．各設計要素を定義するか，表記ゆれ用語を統一した表現に改めて下さい． ・NGワードが181件使われています．NGワードを取り除くか，別の表現に改める必要があります．	C

評価詳細（定義漏れ／表記ゆれ用語の詳細は本レポート添付の「未定義用語一覧」をご覧下さい．





NGワード

NGワード出現数		21	NGワード使用状況判定
NGワード出現数（延べ）		181	
主要な使用NGワード	約	35	C
	など	12	
	その他	11	
	十分な	9	
	必要に応じて	5	
	全て	4	

主な定義漏れ，表記ゆれ用語

アクター	・「販売管理者」がアクターとして定義されていません． ・「マスタ管理者」，「システム管理者」，「管理者」が混在して利用されています．
データ	・「送付先明細書」がデータとして定義されていません． ・「注文伝票」と「注文書」が混在して利用されています．
画面	・「注文情報確定画面」が画面として定義されていません．
振舞い	「登録」，「入力」が混在して利用されています．

図 4-2　検証レポート可視化例

4.2.3 産業界への要望

今回開発したツールはソースコードも含めて公開する．要求仕様書を記述する技術者，記述された結果を確認する管理者，提示された要求仕様書に基づいてシステムを開発する開発者の方々に利用していただき，要求仕様書の検証にとどまらず，内容理解や設計要素の抽出支援などに活用していただきたい．加えて，利用対象を要求仕様書に限定せず，様々なドキュメントの検証や分析に用いて，要求定義のレベルアップに活用していただきたい．

【ツールの公開 URL】
http://www.ns.kogakuin.ac.jp/~wwa1076

参考文献

[1] 一般社団法人 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.

[2] ISO/IEC/IEEE 29148:2011, Systems and software engineering -- Life cycle processes -- Requirements engineering, 2011.

[3] 独立行政法人情報処理推進機構(IPA)技術本部ソフトウェア高信頼化センター(SEC), 共通フレーム 2013～経営者、業務部門とともに取組む「使える」システムの実現～, 2013.

[4] 一般社団法人 情報サービス産業協会 REBOK 企画 WG, 要求工学実践ガイド REBOK シリーズ 2, 近代科学社, 2014.

[5] 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.

[6] 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.

[7] Alistair Cockburn, Writing Effective Use Case, Addison Wesley, 2000, ウルシステムズ株式会社 (監訳), 山岸耕二, 矢崎博英, 水谷雅宏, 篠原明子(訳), ユースケース実践ガイド―効果的なユースケースの書き方 (OOP Foundations), 翔泳社, 2001.

[8] Indi Young, Mental Models Aligning Design Strategy with Human Behavior, Rosenfeld Media, 2008, 田村大 (監訳) , 酒井洋平, 澤村正樹, 重村将之, 羽山祥樹, 吉岡典子 (訳) , メンタルモデル ユーザーへの共感から生まれる UX デザイン戦略, 丸善出版, 2014.

[9] Mica R. Endsley and Debra G. Jones, Designing for Situation Awareness, An Approach to User-Centered Design Second Edition, CRC Press, 2011.

[10] John M. Carroll, Making Use: Scenario-Based Design of Human-Computer Interactions, MIT Press, 2000, 郷健太郎 (翻訳), シナリオに基づく設計ーソフトウェア開発プロジェクト成功の秘訣―, 共立出版, 2003.

[11] 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　技術シンポジウム SPES2014 SPES 事例研究 (経験報告) 2014 年 10 要求工学 S4a, 2014.

[12] 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.

[13] 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.

[14] 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.

[15]久野愛子，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，一般社団法人電子情報通信学会，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.
[16]位野木万里，高品質な要件定義のための暗黙知の形式知化と共有手法，東芝レビュー，Vol.66，No.9，pp.64-65，2011.
[17]位野木万里，要件定義の生産性を向上させるための最適化への取り組み，情報サービス産業協会 SPES2011，査読有（SPES2011 ベストプラクティス賞受賞），2011.
[18]位野木万里：要求獲得におけるステークホルダ識別手法の実適用評価，情報処理学会デジタルプラクティス，Vol.4，No.2，pp.152-160，2013.
[19]Mari Inoki，Takayuki Kitagwa，Shinichi Honiden，Application of Requirements Prioritization Decision Rules in Software Product Line Evolution，Proc. of 2014 IEEE 5th International Workshop on Requirements Prioritization and Communication (RePriCo)，pp.1-10，2014.
[20]可児佑介，位野木万里，ドメイン特化型要求抽出の取り組み～交差点における運転行動による環境変化を考慮した要求抽出，情報サービス産業協会，SPES2014，2014.
[21]Karl Wiegers and Joy Beatty，Software Requirements，Third Edition，Microsoft Press，2013，渡部洋子（訳），宋雅彦（監修），ソフトウェア要求 第3版，日経BP社，2014.
[22]樽本徹也， ユーザビリティエンジニアリング（第2版）―ユーザエクスペリエンスのための調査、設計、評価方法―，オーム社，2014.
[23]Marja Käpyaho and Marjo Kauppinen，Agile Requirements Engineering with Prototyping: A Case Study，proc. of IEEE 23rd International Conference on Requirements Engineering 2015，pp.334 - 343，2015.
[24]ISO 9241-210:2010，Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems，ISO，2010.
[25]Abdulrhman Alkhanifer and Stephanie Ludi，Towards a Situation Awareness Design to Improve Visually Impaired Orientation in Unfamiliar Buildings: Requirements Elicitation Study，proc. of IEEE 22nd International Conference on Requirements Engineering，pp.23 - 32，2014.
[26]Edith Zavala，Xavier Franch，Jordi Marco，Alessia Knauss，and Daniela Damian，SACRE: A Tool for Dealing with Uncertainty in Contextual Requirements at Runtime，Demos and Posters Session，IEEE 23rd International Requirements Engineering Conference，2015.
[27]Markus Fockel and Jörg Holtmann，ReqPat: Efficient Documentation of High-Quality Requirements using Controlled Natural Language，Demos and Posters Session，IEEE 23rd International Requirements Engineering Conference，2015.
[28]Luxi Chen，Linpeng Huang，Hao Zhong，Chen Li，and Xiwen Wu，Breeze: A Modeling Tool for Designing，Analyzing，and Improving Software Architecture，Demos and Posters Session，IEEE 23rd International Requirements Engineering Conference，2015.

[29]Jaison Kuriakose and Jeffrey Parsons, An Enhanced Requirements Gathering Interface for Open Source Software Development Environments, Demos and Posters Session, IEEE 23rd International Requirements Engineering Conference, 2015.
[30]Anas Mahmoud, An Information Theoretic Approach for Extracting and Tracing Non-functional Requirements, Proc. of IEEE 23rd International Requirements Engineering Conference, pp.36-45, 2015.
[31]Cilibrasi, R. L. and Vitanyi, P. M. B., The Google Similarity Distance,IEEE Transactions on Knowledge and Data Engineering, 19(3), 370-383, 2007.
[32]Alessio Ferrari, Paola Spoletini, and Stefania Gnesi, Ambiguity as a Resource to Disclose Tacit Knowledge, Proc. of IEEE 23rd International Requirements Engineering Conference, pp. 26-35, 2015.
[33]MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://taku910.github.io/mecab/#download
[34]Ruby: https://www.ruby-lang.org/ja/about/
（以下よりインストール実施：http://rubyinstaller.org/downloads/）