Vol.2015-SE-187 No.19
2015/3/12

SRS インスペクタ：
RDF を用いたソフトウェア要求仕様書解析ツールの提案と評価

中根 拓也[†1]　　青山 幹雄[†2]

ソフトウェア要求仕様書(SRS)の品質はソフトウェア開発とその成果物の品質を左右する．SRS の品質を確保するためにインスペクションが広く実践されている．しかし，SRS の大部分は自然言語で記述されており，インスペクションでは人による自然言語の読解に時間を要し，誤りも多くなる．本稿では，インスペクションの生産性と品質向上のために RDF を用いた SRS 解析方法とその支援ツール SRS インスペクタ提案する．Word 形式の SRS を意味に基づき構造分解し，RDF 形式に変換する．インスペクションポイントセットと質問セットを基に SPARQL の抽出クエリと解析シナリオを定義する．RDF 形式に変換した SRS から抽出クエリを用いて解析に必要なリソースを抽出し，シナリオに沿った解析を行う．例として，実際の RFP(Request For Proposal)に適用し，その有効性を示す．

SRS Inspector: A Software Requirements Specifications Analysis System Using RDF Descriptions

TAKUYA NAKANE[†1]　　MIKIO AOYAMA[†2]

The quality of SRSs (Software Requirements Specifications) influences the quality of software development and product. Inspection is widely practiced to ensure the quality of SRSs. However, because of the major part of SRS is written in a natural language, inspection takes longer time to read and is error prone. This article proposes a SRS inspection method using RDF and SRS inspector, a system to improve the quality and productivity of inspection. SRS Inspector decomposes a SRS in Word document based on semantic structure and transforms the SRS into RDF document. This method defines a set of SPARQL queries for extracting resources of RDF document and generates a scenario to analyze the resource based on the “inspection point set” and “question set”. SRS Inspector extracts a set of resource needed for analysis by using a set of SPARQL queries, and analyzes along with a scenario. We applied the proposed method to a RFP (Request For Proposal), and demonstrated the validity.

1. はじめに

ソフトウェア開発では開発規模の増大とプロジェクトメンバの専門化により，要求定義とそれ以降の開発プロセスでは異なるチームが担当することが多くなっている．そのため，要求定義の成果物であるソフトウェア要求仕様書(SRS: Software Requirements Specifications)の重要性が高まっている．SRS の品質はソフトウェア開発とその成果物の品質を左右する．SRS の品質を確保するために要求の検証や妥当性確認が行われており，その手法の 1 つとしてインスペクションが広く実施されている．しかし，SRS の大部分は自然言語で記述されているため，SRS のインスペクションでは人による自然言語の読解作業に時間がかかり，その結果として誤りも多くなる．また，SRS のインスペクションは開発経験やスキルに依存する傾向があるため，属人的な要素の軽減が必要である．

本稿では，SRS のインスペクションの生産性と品質向上のため，RDF を用いた SRS 解析ツールを提案する．実際の RFP(Request For Proposal)[8]に対し提案する SRS 解析ツールを用いて検証を行い，提案方法の妥当性を示す．

†1 南山大学 大学院 理工学研究科 ソフトウェア工学専攻
Nanzan University
†2 南山大学 理工学部 ソフトウェア工学科
Nanzan University

2. 研究課題

本稿の研究課題は以下の 2 つとする．

(1)　SRS の意味構造分解

わが国では, SRS は Word 文書や Excel で記述されることが多い．また，SRS は企業ごとに構造が多様であるため，SRS に対するシステムによる統一的な品質検証が困難である．そのため，企業ごとに構造が異なる SRS を意味構造に基づき分解し，RDF を用いた共通表現に変換する．これによりシステムによる統一的な検証を可能とする．

(2)　共通表現に基づく SRS インスペクションの自動化

システムによる SRS インスペクションの自動化を実現するため，(1)の共通表現に基づくインスペクション方法を定義する．SRS のインスペクション方法において定義されているインスペクションポイントセットと質問セットを SPARQL で表現することにより RDF 形式の SRS に対するインスペクションを可能とする．

3. 関連研究

3.1 RDF (Resource Description Framework)

RDF は Web 上のリソースのメタデータの表現形式である[10]. リソース間の関係を主語, 述語, 目的語の三要素(トリプル)で表現するデータモデルである．

RDF のクエリ言語に SPARQL (SPARQL Protocol And RDF Query Language)がある[11]．SPARQL は主語，述語，目的語からなる基本グラフパターンと RDF グラフとのパターンマッチにより検索を行う．SPARQL には以下のクエリ形式がある．

(1) SELECT

クエリパターンにバインドされた変数のすべてまたは一部を返す．

(2) CONSTRUCT

グラフテンプレートで指定された 1 つの RDF グラフを返す．

(3) ASK

対象の RDF グラフがクエリパターンに一致するかどうかを返す．

(4) DESCRIBE

指定したリソースに関する RDF データを含んだ RDF グラフを返す．

本稿では検証に必要なリソースを抽出するため，SELECT 形式のクエリを使用する．

3.2 OSLC に基づく要求管理方法と支援環境の提案

Web 上で SRS の管理を実現するため，OSLC[7]に基づく要求管理方法が提案されている[1]．

この研究では，SRS の標準として IEEE 830[3]，管理情報の標準として要求工学知識体系(REBOK)を用い，これらを基に要求管理プロパティモデルを定義している．また，OSLC の要求管理，変更管理，構成管理で定義されているプロパティを要求仕様の管理と管理情報の管理に対応する情報に分類し，OSLC に基づくプロパティモデルを定義している．さらに，要求管理プロパティモデルと OSLC に基づくプロパティモデルをマッピングすることで，要求仕様と管理情報の表現，管理が可能な OSLC に基づく要求管理プロパティモデルが定義されている．このプロパティモデルに基づき，SRS を RDF 形式に変換することで OSLC 上で管理を行い，Web を介した SRS の管理が実現されている．

3.3 SRS のインスペクションデザイン方法論の提案

インスペクションの体系的なデザイン方法論が提案されている[9]．この研究では，SRS の満たすべき品質として PQC (Pragmatic Quality Characteristic)が定義されている．PQC は PBR (Perspective Based Reading)を導入し，IEEE 830 の品質特性から導出されている．「顧客」,「プロジェクトマネージャ」,「ソフトウェア技術者」,「アーキテクト」の 4 つのアクタに対するパースペクティブとして「要求する」,「管理する」,「実装する」,「アーキテクチャに割り当てる」が選定され，そこからさらに詳細化された 7 つのパースペクティブが選定されている．また，それらのパースペクティブに基づき SRS に求められる参照 SRS 品質特性が選択されている．これらのパースペクティブと参照 SRS 品質特性から各パースペクティブに基づいて SRS が備えるべき特性が PQC として定義されている．

PQC に基づくインスペクションを SRS の適切な箇所で実施するため，標準 SRS の目次項目と PQC を対応付けたインスペクションポイントセットが作成されている(表 1)．

表 1 インスペクションポイントセット

PQC		標準 SRS の目次			
ID	名称	2.1 システム化の目的	2.2 業務概要とシステム化の範囲	2.3 制約事項	2.4 用語定義
C1	合目的性	X			
C2	記述項目網羅性	X	X	X	X
C3	テンプレート使用	X	X	X	
C4	標準記法使用		X		
C5	用語定義				X
C6	識別子の付与	X	X	X	X
C7	一意識別性	X	X	X	X

また，インスペクションポイントで確認すべき内容が Yes/No の質問形式で策定され，各 PQC に対応した 7 つの質問(質問セット)が作成されている(表 2)．インスペクションポイントセットと質問セットを参照することで，インスペクションの実行者は，SRS のどの箇所を何の PQC に基づいて評価すべきか正確に把握することができる．

表 2 質問セット

PQC		質問セット
ID	名称	
C1	合目的性	SRS のビジネス・システム要求はプロジェクトの目的に対応付けて記載されているか？
C2	記述項目網羅性	SRS の要素は標準 SRS に対応しているか？
C3	テンプレート使用	SRS の成果物は標準 SRS の中で定められているテンプレートを用いて記載されているか？
C4	標準記法仕用	SRS の成果物は標準 SRS の中で定められている標準(記述)記法で記載されているか？
C5	用語定義	SRS の用語集は作成されているか？
C6	識別子の付与	SRS の成果物や特定の要素は識別子が付与されて表で一覧化されているか？
C7	一意識別性	SRS の成果物や特定の要素は識別子を用いて一意に特定できるか？

4. アプローチ

4.1 SRS 解析方法のフレームワーク

ソフトウェアによる SRS 解析方法のフレームワークを図 1 に示す．本稿では，ソフトウェアによる SRS 解析方法を提案するため以下のアプローチを取る．

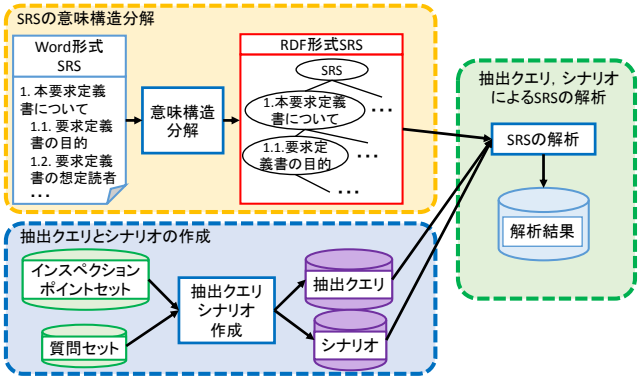

図 1 SRS 解析方法のフレームワーク

4.2 SRS の意味定義表現

SRS は Word や Excel で記述されていることがほとんどである．また，SRS は企業ごとに構造が多様であるため，システムによる統一的な検証が困難である．そこで，SRS を意味構造に基づいて分解し，共通表現をとれるようにする．この抽象表現を SRS 意味定義(SSD: Specification Semantic Definition)と呼ぶ．SSD は RDF を用いて表現することにより，ツールによる統一的な検証が実行可能となる．

4.3 抽出クエリとシナリオの作成

共通表現に変換された SRS を解析するための抽出クエリとシナリオを作成する．SRS のインスペクション方法に基づいた検証を行うため，インスペクションポイントセットと質問セットを基に検証に必要なリソースを抽出する抽出クエリと抽出したリソースに対する検証方法を記述したシナリオを作成する．

4.4 抽出クエリ，シナリオによる SRS の解析

作成したクエリを用いて共通表現に変換さえた SRS から検証に必要なリソースを抽出し，検証する PQC ごとにシナリオに沿った品質検証を行う．これにより，SRS に対しソフトウェアによる PQC に基づく検証が可能となる．

4.5 前提条件

適用対象として，入力する SRS は企業ごとのテンプレートに則っており，Word 文書で記述されているものとする．また，企業ごとのテンプレートの目次項目と IEEE 830 で定義されている標準 SRS の目次項目が対応付けられているものとする．

5. 提案方法

5.1 SRS の意味定義モデル

SRS の意味定義モデルを図 2 に示す．本稿では，IEEE 830 と REBOK で定義されている SRS の構成を標準 SRS として用いる．標準 SRS に基づいて定義された参照 SRS として，参考文献[9]で，IEEE 830 や企業で作成された SRS の内容に基づいて作成された SRS の構成を用いる．参照 SRS を特殊化したものがインスペクションするプロジェクト SRS である．参照 SRS の構造を RDF で表現することで，SRS の意味を定義することができる．これにより RDF を用いた SRS の SSD を定義する．

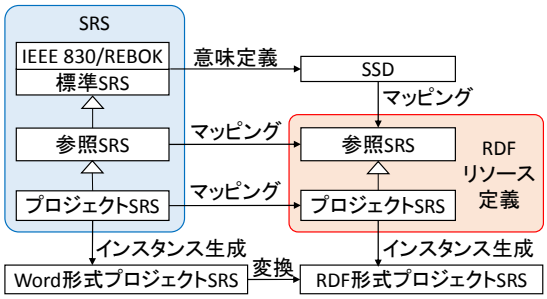

図 2 SRS の意味定義モデル

5.2 SRS リソースモデル

SRS を意味構造に基づき分割し，複数の SSD に基づくリソースに変換する．参考文献[1]を基に拡張したプロパティを用いて，参照 SRS の RDF 表現を定義する．参照 SRS のリソース定義を図 3 に示す．リソースはプロパティとして識別子，タイトル，内容，リソース作成日時などを持つ．子リソースを持つ場合は oslc_rm:decomposedBy プロパティの値として子リソースの URL を持ち，dcterms:description プロパティの値として，子要素となる章節の目次項目を記述する．子リソースを持たない場合は，dcterms:description プロパティの値として SRS の本文を記述する．また，SRS 内の表から生成されたリソースの場合，oslc:element プロパティの値として表の要素名の URL を持つ．

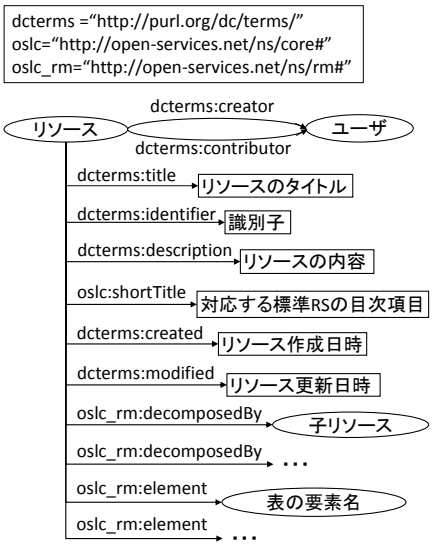

図 3 参照 SRS のリソース定義

5.3 RDF を用い SRS 解析プロセス

RDF を用いた SRS 解析プロセスを図 4 に示す．提案プロセスでは初めに検証対象リソースを抽出するクエリを作成することで，複数の SRS に対するソフトウェアによる検証が繰り返し実行可能となる．

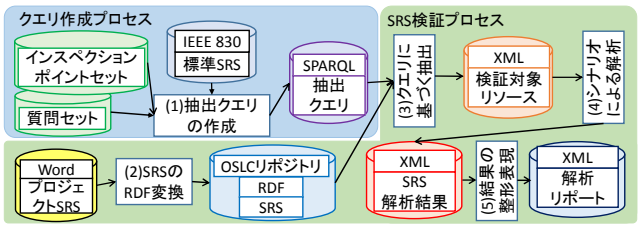

図 4 RDF を用いた SRS 解析プロセス

(1) 抽出クエリの作成

インスペクションポイントセットと質問セットを基に，検証に必要なリソースを抽出するためのクエリを作成する．クエリは SPARQL を用いて記述する．

(2) SRS の RDF 変換

Word 文書で記述された SRS を[1]を基に拡張した SRS 解析のためのプロパティに基づき RDF 形式に変換する．変換

した SRS は OSLC リポジトリ上で管理する．

(3)　クエリに基づく抽出

(1)のクエリを用いて OSLC リポジトリ上の SRS から検証に必要なリソースを抽出する．

(4)　シナリオに沿った解析

(3)の結果に対し，質問セットの質問に基づいて検証をするためのシナリオに沿った検証を行う．

(5)　結果の整形表現

(4)の結果を人に理解しやすい形式で表現する．

5.4 抽出クエリの作成

検証に必要なリソースを抽出するため，インスペクションポイントセットと質問セットを基に，抽出クエリを作成し，SPARQL で記述する．抽出クエリは PQC の項目と検証を行う標準 SRS の目次項目ごとに作成する(図 5).

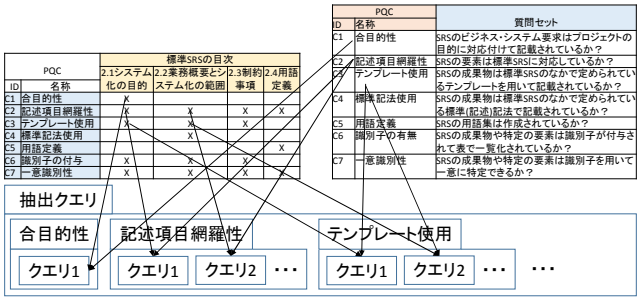

図 5　抽出クエリの作成

抽出クエリの導出過程を記述項目網羅性の例を用いて示す(図 6). RDF 形式に変換した SRS において，記述項目網羅性は検証対象となる目次項目の内容を表すリソースの有無により判別可能である．そのため，検証対象の目次項目に対応するリソースの内容を抽出するクエリを作成する．図 6 のクエリ 1 はシステム化の目的に関する記述項目網羅性の検証に用いるクエリである．対応する標準 SRS の目次を表す oslc:shortTitle プロパティの値が「システム化の目的」であるリソースの内容を示す dcterms:description プロパティの値とそのリソースの URI を取得するクエリとなっている．また，これに対応するシナリオは「dcterms:description プロパティの値の有無のチェック」となる．

同様に，他の標準 SRS の目次に関するクエリとシナリオを作成する．

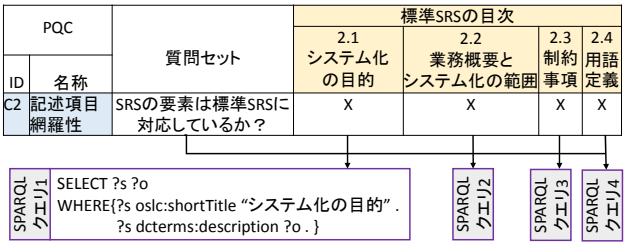

図 6　記述項目網羅性に関するクエリの導出

5.5 SRS の RDF 変換

図 4 の SRS の RDF 変換の詳細を図 7 に示す．構造の異なる SRS を[1]に基づき拡張した SRS プロパティを基に，RDF 形式に変換する．さらに，企業ごとの SRS テンプレートを基に, RDF 化した SRS の各リソースに対して標準 SRS の目次項目を対応付けることで意味付けを行う．変換した SRS は OSLC リポジトリ上で管理する．これにより企業ごとに構造が異なる SRS を参照 SRS に基づく共通表現に変換でき，SRS に対する統一的な検証が可能となる．以下に各コンポーネントの詳細を示す．

(a)　Word アダプタ

入力された Word 形式の SRS を任意の XML 形式に変換する．

(b)　リソースマッパー

Word アダプタにより任意の XML 形式に変換された SRS を SRS 分析のためのプロパティモデルに基づき RDF 形式に変換する．

(c)　リソース変換アダプタ

企業ごとの SRS テンプレートに対し対応付けられた標準 SRS の目次項目に基づいて，RDF 形式の SRS の各リソースに対して標準 SRS の目次項目をプロパティとして付与する．作成したリソースは OSLC リポジトリ上に配置する．

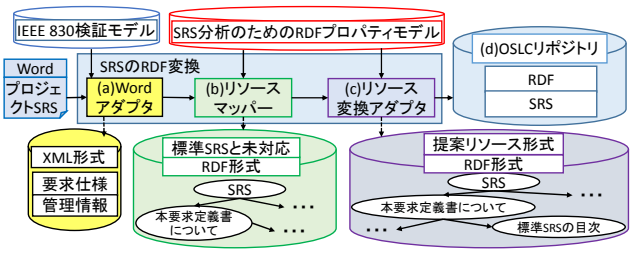

図 7　SRS の変換プロセス

5.6 シナリオに基づく SRS の解析

OSLC リポジトリ上の SRS に対して SPARQL の抽出クエリを用いて検証に必要なリソースを抽出し，シナリオに沿った SRS の解析を行う(図 8). OSLC リポジトリ上の SRS に対して SPARQL の抽出クエリを用いて検証に必要なリソースを抽出する．抽出したリソースに対し，対応する質問セットを基にしたシナリオに沿った解析を行う．解析では，リソースの記述の有無やリソースの構造，リソース間の対応の検証により，質問を満たしているかの判定を行う．

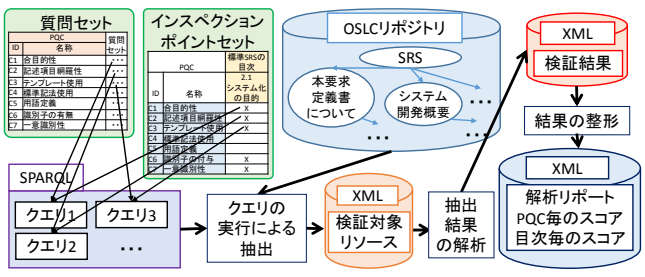

図 8　SRS の解析

解析結果を整形し，解析リポートを作成する．解析リポートは人に理解しやすい形式にするため, 解析結果を基に，PQC ごとの品質スコアと標準 SRS の目次項目ごとの品質

スコアを算出する．品質スコアは未検証の質問を除いた総質問数に対し回答が Yes の質問の割合を示す．この結果を参考にインスペクションを行うことで，インスペクションにかかる時間の短縮，誤りの低減が期待できる．

5.7 提案システムの拡張

提案システムを拡張し，SRS 内で用いられている曖昧な用語を検出する機能を追加した(図 9)．曖昧な用語は参考文献[12]の「要求で避けるべきあいまいな用語」を用いる．参考文献[12]では，要求の曖昧さを生み出す共通の原因が説明されており，その中で検証不能な要求につながる曖昧な用語と曖昧さを除去する方法が提案されている．それを基に，曖昧な用語のリストを作成し CSV 形式で記述する．OSLC 上の SRS の各リソースの内容を抽出し，その内容において曖昧な用語リスト内の用語が用いられているかチェックすることで SRS 内の曖昧な用語を検出する．

検出結果として解析リポートを作成する．解析リポートには，曖昧な用語ごとの出現回数と対象 SRS の章ごとの曖昧な用語の出現回数を算出し，記載する．解析リポートは XML 形式で出力する．

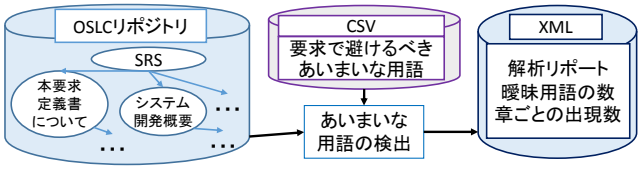

図 9　曖昧な用語の検出

6. プロトタイプ開発と例題への適用

提案方法の妥当性確認のため，プロトタイプを構築し例題に適用することで，RDF 形式の SRS に対する抽出クエリとシナリオを用いた自動検証が可能であることを示す．

6.1 プロトタイプの開発

OSLC の参照実装である Eclipse Lyo[2]を拡張して実装したプロトタイプの開発環境を表 3，構成を図 10 に示す．

Word アダプタ，SRS 解析器，曖昧用語検出器は Java，JSP を用いて実装し，リソースマッパー，リソース変換アダプタ，解析結果アナライザは Java を用いて実装した．規模は Java が 879 (LOC)，JSP が 68 (LOC)である．また，リソースを検証する際に用いるシナリオは解析結果アナライザ内で実装により実現している．

表 3　開発環境

システム	バージョン
OS	Windows 8.1
JDK	1.6.0_45
Eclipse	4.4.1
Jetty	7.3.0
RDF Store	Sesame

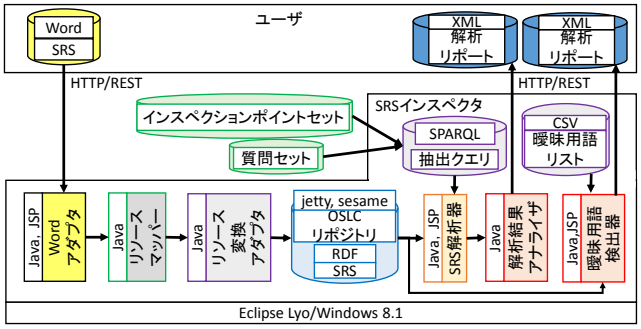

図 10　SRS インスペクタ

以下に各コンポーネントの詳細を示す．

(1)　Word アダプタ

Word 形式の SRS を章や節ごとのまとまりの構築などを行い，任意の XML 形式に変換する．

(2)　リソースマッパー

Word アダプタにより処理された SRS を RDF 形式に変換する．XML 形式の SRS を章，節などの要素ごとに分割し，URI を付与することで複数の RDF に変換する．

(3)　リソース変換アダプタ

企業ごとのテンプレートに対する標準 SRS の目次項目の対応付けを基に，リソースマッパーにより RDF 形式に変換された SRS の各リソースに対し，対応する標準 SRS の目次項目をプロパティとして付与する．これにより，SRS を提案したリソース型へ変換する．作成したリソースは OSLC 上に配置する．

(4)　SRS 解析器

抽出クエリを用いて OSLC 上の SRS から検証に必要なリソースを抽出する．抽出クエリをあらかじめ作成しておき，クエリセットの親ディレクトリを入力で指定することにより一度に各クエリに対する結果を生成可能である．抽出したリソースは XML 形式で出力する．

(5)　解析結果アナライザ

SRS 解析器により抽出された SRS の各リソースに対してシナリオに沿った解析を行う．本稿のプロトタイプではシナリオは実装により表現している．すべての検証対象リソースに対する解析結果を XML 形式で出力する．解析結果は，質問に対する回答が「Yes」に相当する場合は「○」，「No」に相当する場合は「×」，未検証の質問は「？」を用いて表現する．

さらに，検証結果を基に解析リポートを作成する．解析リポートは，標準 SRS の目次項目ごとの品質スコアと PQC ごとの品質スコアを算出し，記載する．

(6)　曖昧用語検出器

OSLC 上の SRS に対し，曖昧用語リストを基に各リソースで使用されている曖昧な用語の検出を行う．曖昧な用語は，参考文献[12]で定義されている「要求で避けるべきあいまいな用語」を用い，それらの用語の一覧を CSV 形式で記述する．OSLC リポジトリ上の SRS からすべてのリソー

スの内容を取得し，取得した内容に対して曖昧用語リストに記載されている用語が用いられているかチェックを行う．検証結果として，曖昧な用語ごとの出現回数と対象 SRS の章ごと曖昧な用語の出現回数を算出し，解析リポートに記載する．

6.2 例題への適用

本稿では SRS に対する解析ツールを提案しているため SRS を用いる必要があるが，実際に企業の開発で用いられている SRS は一般に公開されていない．そのため，適用する例題を実際の SRS に代わり，山梨県甲府市のホームページリニューアルに関する RFP(総 25 ページ)[5]とした．RFP に対する検証であるため，参照 SRS に代わり IT コーディネータ協会の RFP[4]を参照 RFP として利用し，インスペクションポイントセット，質問セットに代わり参考文献[6]で提案されている RFP に対するインスペクションマトリクスと RFP を評価するための検証質問セットを利用する．そのため RFP の品質評価も参考文献[6]で定義されているプラグマティック品質に基づく品質評価とする(表 4)．

表 4　プラグマティック品質

ID	品質特性	ID	副品質特性
U1	合目的性	U1-1	システム目的の独立性
		U1-2	業務要求のシステム目的への適合
U2	生産効率性	U2-1	定量的具体性の有無
		U2-2	要求の独立性
U3	堅実性	U3-1	標準 SRS との整合性
		U3-2	文書の参照関係の明示
		U3-3	例外要求の網羅
		U3-4	変更可能性の明記
		U3-5	一意に特定可能
		U3-6	用語集の存在
U4	充足性	U4-1	ランク付けの有無
		U4-2	用語の整合
		U4-3	動作の整合
		U4-4	制約条件と要求の整合

また，本稿の提案方法では検証対象の RFP に対する標準 RFP の目次項目の対応付けが必要であるが．適用対象 RFP は標準 RFP との対応付けがされていない．そのため，適用対象 RFP の目次項目と標準 RFP の目次項目の対応付けを行ってから例題へ提案方法を適用した．

RFP を評価するための検証質問セットでは質問ごとにペルソナの視点による差分があるが，提案方法ではペルソナの視点を表現することができない．また，RFP 内の図に関する質問も本稿の提案方法では検証が不可能である．そのため，図に関する質問とペルソナの視点による差分を除いた 68 の質問を有効質問とし，検証を行った．「例外要求の網羅」，「変更可能性の明記」，「用語集の存在」については RFP に対する質問項目が無く，「動作の整合」については図に対する検証が必要なため，これらのプラグマティック品質に関する検証は行わない．

提案方法による検証の結果，52 の質問について検証できた(図 11)．また，参考文献[12]に基づく 75 種類の曖昧な用語の検出結果を表 5 と表 6 に示す．検証の結果，14 種類の曖昧な用語が検出された．RFP の入力から検証結果の出力までの検証時間は約 5 分であった．

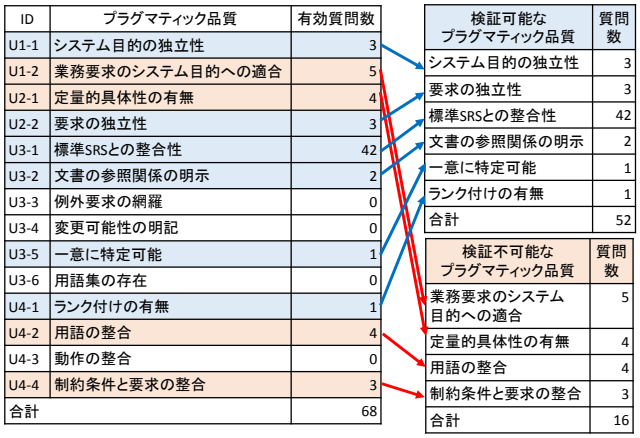

図 11　検証結果

表 5　対象 RFP の曖昧用語検出結果

章	文字数	種類数	出現数
1	790	2	2
2	1,032	0	0
3	916	0	0
4	1,114	7	8
5	3,101	7	8
6	3,960	4	7
7	4,732	8	9
8	2,879	2	3
9	617	0	0
合計	19,141	37	

表 6　対象 RFP の頻出曖昧用語

用語	出現数
その他	6
～しない	5
i.e.	5
など	4
～から～まで	3
含めて	3

注：種類数：重複なし
　　出現数：重複あり

7. 評価と考察

評価と考察において本稿では，検証率，検証網羅率，品質スコアを式(1)~(3)で定義する．

$$検証網羅率 = \frac{検証可能質問数}{有効質問総数} \quad (1)$$

$$検証率 = \frac{想定した結果を得た質問数}{検証した質問数} \quad (2)$$

$$品質スコア = \frac{Yes数}{検証数} \quad (3)$$

7.1 検証可能性の考察

提案方法による検証の結果，検証網羅率は 76.4%であった．検証できたプラグマティック品質は以下の 6 つである．

(1)　システム目的の独立性
(2)　要求の独立性
(3)　標準 SRS との整合性
(4)　文書の参照関係の明示

(5)　一意に特定可能
(6)　ランク付けの有無

これらのプラグマティック品質については提案方法による検証が可能であると言える．しかし，独立性に関する品質の検証は，個々の目的，要求が分割されているかという限定的な検証となった．これは質問をより具体的にし，分割することで改善可能と考えられる．

また，以下のプラグマティック品質に関しては検証不可であった．

(1)　業務要求のシステム目的への適合

システム目的が 1 つの文章で記述されており，RFP を RDF 形式に変換する際，単一のリソースとして変換されたため，業務要求のリソースとシステム目的のリソースにおいて，それぞれのリソース間の対応による検証ができず，検証不可となった．

(2)　定量的具体性の有無

検証対象となるリソースが細分化できず，対象リソースの内容について具体的な値や表現がされているか判別できず，検証不可となった．

(3)　用語の整合

文章中の用語がその用語の意味にそって正しく用いられているかについて検証困難であり，また，RFP では用語集との関連が検証されないため用語集に関する検証も不可であるため検証不可となった．

(4)　制約条件と要求の整合

要求および制約条件が複数のリソースではなく，単一のリソースとして変換されたため，制約条件のリソースと要求のリソースでの対応による検証ができず，検証不可となった．

これらのプラグマティック品質の検証では，検証対象のリソースが分割されず単一のリソースとして変換されたため，リソース間の対応による検証が不可であった．そのため，細粒度のリソース定義を行い，複数のリソースに分割することで，検証可能になると考えられる．

7.2　プラグマティック品質毎の品質スコアに基づく考察

プラグマティック品質ごとの品質スコアを表 7 と図 12 に示す．

検証を行った 6 つのプラグマティック品質については検証率が 100%であり，漏れなく検証できている．このことから，検証対象の目次項目によらず，リソースの構造やリソース間の対応による検証が可能であると言える．そのため，これらのプラグマティック品質に関しては，システムによる自動検証が可能である．

また，「要求の独立性」，「一意に特定可能」，「ランク付けの有無」の品質スコアは 0%となっている．特に，「一意に特定可能」と「ランク付けの有無」は検証数が 1 のため真偽の結果となっており，品質スコアの精度が低いと考えられる．そのため，細粒度のリソース定義を行い，検証の細粒度化による品質スコアの精度の向上が必要である．

表 7　プラグマティック品質毎の品質スコア

ID	総数	検証数	検証率[%]	Yes 数	品質スコア[%]
U1-1	3	3	100.0	1	33.3
U2-2	3	3	100.0	0	0.0
U3-1	42	42	100.0	27	64.3
U3-2	2	2	100.0	1	50.0
U3-5	1	1	100.0	0	0.0
U4-1	1	1	100.0	0	0.0
合計	52	52	100.0	29	55.8

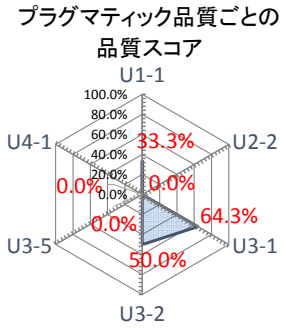

図 12　品質スコアを表すグラフ

7.3　目次項目毎の品質スコアに基づく考察

標準 RFP の章ごとの品質スコアを表 8 と図 13 に示す．

表 8　目次項目毎の品質スコア

章 ID	総数	検証数	Yes 数	未検証数	品質スコア[%]
R1	21	14	5	7	35.7
R2	28	20	15	8	75.0
R3	5	5	1	0	20.0
R4	4	4	4	0	100.0
R5	3	2	2	1	100.0
R6	7	7	2	0	28.6
合計	68	52	29	16	55.8

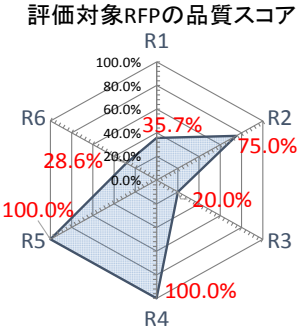

図 13　品質スコアを表すグラフ

「R3: 提案手続き」，「R4: 開発に関する条件」，「R6: 契約事項」については未検証数が 0 である．このことから，これらの章に対するシステムによる自動検証が可能であると言える．しかし，「R1: システム概要」，「R2: 提案依頼事項」，「R5: 保証要件」は未検証項目があり，質問総数に対して未検証の割合が高い．そのため，これらの章に関する品質スコアは正確性が低いと考えられる．品質スコアの正確性を向上させるため，未検証項目の低減が必要である．

7.4 RDF を用いたことに対する考察

Word 形式で記述された RFP を意味構造に基づき分解し，RDF 形式に変換することで複数のリソースに分割した．そのため，リソース間の対応やリソースの構造に関する検証が可能となった．これにより，質問セットに基づく検証が，記述の有無だけでなく，RFP の構造や意味に基づく検証が可能となった．また，構造が異なる RFP を共通表現に変換したことにより，構造が異なる文書に対し統一的な検証が可能となった．

提案方法による検証では，限定的な検証を行ったプラグマティック品質と未検証のプラグマティック品質があった．これらのプラグマティック品質に対する検証は，リソースを細粒度にし，検証対象リソースを分割することで検証可能になると考えられる．提案方法を用いた検証における検証網羅性向上のため，より細粒度のリソース定義が必要であると考えられる．

提案方法を用いることで，ソフトウェアによる RFP に対する構造，表現の検証が可能となった．システムによる検証を行うことで，俗人的な要素が低減可能である．このことから，提案方法を用いることでインスペクションにかかる時間の短縮と品質の向上が期待できる．

7.5 曖昧用語検出結果に対する考察

表 5 から章の文字数の増加に伴い，曖昧用語数が多くなる傾向があると言える．このことから，RFP の文章量が多くなると，曖昧な用語の出現数も多くなり，RFP の品質が低下すると考えられる．提案システムを用いることで，曖昧な用語が自動で検出でき，RFP のインスペクションにかかる時間の短縮が期待できる

7.6 提案システムの拡張性についての考察

本稿では提案するシステムを拡張し，曖昧な用語の検出機能を追加した．検証の結果，対象 RFP 内で使用されている曖昧用語リストの用語はすべて検出できており，提案方法による曖昧な用語の検出が可能であると言える．このことから，提案システムを拡張することで RFP に対する検証機能を追加することが可能であると言える．

RFP を参照 RFP に基づいた共通の形式に変換することで，構造が多様である RFP に対して統一的な検証が可能である．そのため，対象 RFP の構造によらず，参照 RFP の構造に対応した機能を追加するだけで，対象 RFP に対する機能の追加が可能である．このことから，提案システムに対する機能の追加は容易であると言える．

8. 今後の課題

8.1 SRS への適用と実践

本稿では SRS に代わり RFP に提案方法を適用した．そのため，実際の SRS に提案方法を適用した場合に品質改善の効果を確認する必要がある．そのため，実際の SRS に対し，提案方法を適用することが必要である．また，実際のプロジェクトやインスペクションの現場で，本稿の提案方法が有効に機能するか確認する必要がある．

8.2 検証網羅率の向上

例題への適用の結果，4 つのプラグマティック品質が検証不可であった．これらのプラグマティック品質に関する質問の具体化，細粒度のリソース定義を行い，検証方法を定義することで，検証不可のプラグマティック品質に対する検証を可能とし，検証網羅率を向上させる必要がある．

8.3 リソース数の増加による検証時間の増加傾向の測定

提案方法では SRS を RDF 形式に変換した際のリソース数の増加に伴い，検証時間が増加すると考えられる．そのため，リソース数の増加による検証時間の増加傾向を測定する必要がある．

9. まとめ

SRS の品質の自動検証実現のため，要求仕様書の管理方法とインスペクションのデザイン方法論に着目し，RDF を用いた SRS 解析方法を提案した．意味構造に基づき SRS を分解することで RDF 形式に変換した．また，インスペクションポイントセットと質問セットを基に抽出クエリと検証のためのシナリオを作成した．RDF 形式に変換した SRS に対し，抽出クエリとシナリオを用いて検証することで SRS に対しソフトウェアによる自動検証を実現した．提案方法により SRS に対するインスペクションの生産性，品質の向上とインスペクションの所要時間の短縮が期待できる．提案方法を実際の RFP に適用し，評価した．

参考文献

1) 青山 幹雄, 壁谷 孝洋, OSLC に基づく要求管理方法と支援環境の提案と評価, ソフトウェア工学の基礎 XX (FOSE2013 論文集), 近代科学社, Dec. 2013, pp. 263-272.
2) Eclipse Lyo, http://eclipse.org/lyo/.
3) IEEE Std. 830-1998, IEEE Recommended practice for Software Requirements Specification, IEEE, 1998.
4) IT コーディネータ協会, RFP/SLA 見本, 2004, http://www.itc.or.jp/foritc/useful/rfpsla/rfpsla_doui.html.
5) 甲府市, ホームページリニューアル業務に関わる受託事業者選考の事業広告, 2012, http://www.city.kofu.yamanashi.jp/koho/shise/koho/hp/renewal.html.
6) 森下 月菜, 他, ペルソナ観点からの利用品質に着目したソフトウェア要求仕様書のインスペクション方法の提案と評価, 情報処理学会 第 183 回ソフトウェア工学研究会, Mar. 2014, pp. 1-8.
7) OSLC (Open Services for Lifecycle Collaboration), http://open-services.net/.
8) B. Porter-Roth, et al., Request for Proposal, Addison-Wesley, 2002.
9) S. Saito, et al., RISDM: A Requirements Inspection Systems Design Methodology, Proc. of the 22nd IEEE Int'l Requirements Engineering Conference (RE 2014), IEEE, Aug. 2014, pp. 223-232.
10) W3C, RDF 1.1 Primer, http://www.w3.org/TR/rdf11-primer/.
11) W3C, SPARQL 1.1 Query Language, http://www.w3.org/TR/sparql11-query/.
12) K. Wiegers, Software Requirements, 3rd Ed., Microsoft Press, 2013.