

ReqQA: ソフトウェア要求仕様書品質解析ツールの提案と評価

青山 幹雄^{†1} 中根 拓也^{†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。しかし、SRS の品質に関する研究は限定的である。本稿では、SRS のインスペクション設計方法論の成果に基づき、SRS の品質解析方法とその自動化ツール ReqQA(Requirements Quality Analyzer)を提案する。多様な SRS を統一的に解析するために、RDF で表現された中間言語 SRS-CIL(Common Intermediate Language)を提案する。Word 文書の SRS を SRS-CIL に変換し、パースペクティブに基づくプラグマティック品質を解析するツールのアーキテクチャを提案し、プロトタイプを REST サービスとして実装した。プロトタイプを実際の RFP に適用し、提案方法とツールの妥当性、有効性を示す。

ReqQA: A SRS Quality Analyzer and its Application

MIKIO AOYAMA^{†1} TAKUYA NAKANE^{†2}

The quality of SRS (Software Requirements Specification) influences the quality of software development and its product. However, few works have been done on the quality of SRS. This article proposes a method of quality analysis of SRS and its automated tool ReqQA (Requirements Quality Analyzer) based on the work on SRS inspection design methodology. To accommodate diverse forms of SRS, the authors propose SRS-CIL (Common Intermediate Language) represented by RDF. The authors propose the architecture of ReqQA consisting of the converter from SRS in Word, and plug-able tool sets, analyzing the SRS-CIL based on the pragmatic quality model from the reader's perspective. The prototype of ReqQA is implemented and applied to a real RFP, and proves the concept and demonstrates the effectiveness of the ReqQA.

1. はじめに

ソフトウェア要求仕様書(SRS: Software Requirements Specifications)の品質はソフトウェア開発の成否とソフトウェアシステムの品質を左右するため、SRS の品質確保は極めて重要である。SRS の品質を確認するためにレビューやインスペクションが広く実践されている[4]。しかし、これらは人手に頼ることから、コストを要し、かつ SRS の大部分が自然言語で記述されていることとあいまって誤りやすく、個人のスキルにも依存する。さらに、大規模ソフトウェア開発では SRS の規模が数 100 ページにもなり、レビューのコストに加え一貫性なども問題がある。

このような現状に対して、SRS の品質とその分析方法に関する研究は少ないのが現状である[9]。自然言語処理による SRS の解析も試みられてきたが、全文検索に頼ることから、構造や表現の多様性により解析の自動化が困難である。

近年、SRS の表現や解析の新たなアプローチが提案されている。OSLC(Open Services for Lifecycle Collaboration)では、異なるツール間で仕様書の情報を連携するために SRS の一部を RDF[18]を用いて表現する方法が提案されている[13]。また、SRS を異なる表現に変換して扱う方法も提案されている[1, 17]。しかし、これらの方法は表現が固有であり、解析も限定的であるなどの課題がある。

本稿では、SRS の品質を解析するために、SRS をオープンな意味定義言語 RDF で定義された中間言語 SRS-CIL(Common Intermediate Language)に変換し、解析する方法と

その自動化ツール SRS 品質解析システム ReqQA(Requirements Quality Analyzer)を提案する。ReqQA のプロトタイプを実装し、実際の RFP[10]に適用し、提案方法の妥当性と効果を示す。

2. 研究課題

本稿の研究課題は以下の 2 つとする。

- (1) RDF による SRS 共通中間言語(SRS-CIL)の実現可能性
SRS は企業やプロジェクト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解析が困難である。そのため、SRS の構造を表現する共通中間言語 SRS-CIL を、RDF を用いて定義する。さらに、一般に広く利用されている Word で記述された SRS から SRS-CIL への変換の実現可能性を明らかにする。

- (2) SRS-CIL を用いた SRS 品質解析自動化の実現可能性
SRS-CIL を用いた品質解析の自動化の可能性を明らかにする。具体的な方法として SRS インスペクション設計方法論 RISDM[14]に基づき、SRS 品質解析システム ReqQA を構成し、具体例を用いてその妥当性、有効性を評価する。

3. 関連研究

本研究の関連研究として、次の 3 つがある。

3.1 SRS の品質とその分析ツール

ソフトウェアシステムの品質に関する研究は数多いが SRS の品質に関する研究は極めて限定的である。

^{†1} 南山大学 理工学部 ソフトウェア工学科
Dep. of Software Engineering, Nanzan University

^{†2} 南山大学大学院 理工学研究科 ソフトウェア工学専攻
Graduate School of Science and Engineering, Nanzan University

IEEE 830 では、SRS の表現に関する 8 つの品質特性を規定している[5]. また、要求工学知識体系(REBOK)では 11 の要求特性を定義している[8]. このような特性に対し、SRS の品質を解析するツールの必要性が指摘されている[9]. しかし、SRS の構造は企業やプロジェクトごとに異なることからツールの実現は困難である. これに対し、SRS の形式的中間言語 RSL-IL が提案されているが[1], その文法は固有であり、解析の提案も限定的である[17].

また、SRS の記述に関する文法的、意味的な品質に対して、特定の読者を想定したプラグマティック品質の概念が提案されている[11]。しかし、それを用いた評価方法の具体化や適用は報告されていない。

3.2 SRS の表現とその支援環境

SRS を処理するために、適切な形式で表現する必要性が指摘されている。例えば、SRS を含む開発成果物を管理するためのリポジトリが IDE や ALM (Application Lifecycle Management) のコンポーネントとして提供されている。しかし、これらはベンダ固有であり、特定のツールに利用が限定されている。

これに対し、OSLC では、異なるツール間でデータを連携するために RDF[18]を用いてデータを表現する[13]. OSLC 上で SRS を管理する仕様が公開されているが、対象は管理データに限定されている. さらに、OSLC 上で SRS の内容を表現するために、IEEE 830 と REBOK に基づく SRS のデータモデルが提案されている[2]. しかし、その解析には至っていない.

3.3 SRS のインスペクションとその設計方法論

SRS のインスペクションの体系的な設計方法論が提案されている[14]. SRS の満たすべき品質としてプラグマティック品質特性 (PQC: Pragmatic Quality Characteristic)が定義され[11], PQC に対する PBR (Perspective Based Reading)[16]に基づきインスペクションを設計する. インスペクションを SRS の適切な箇所で行うため, 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生成される. インスペクションポイントで確認すべき内容が質問セットとして定義されている. しかし, インスペクションは人手で行われ, 自動化には至っていない.

4. アプローチ

4.1 SRS 品質モデル

本稿の対象とする、主として自然言語で記述された SRS はその表現(Syntax)と意味(Semantic)共に多様性が避けられない。このような SRS のリーディングの視点であるパースペクティブが多様性の抑制に効果があることが指摘されている[16]. これは、SRS の解析におけるパースペクティブの有効性を示唆するといえる。一方、自然言語で記述された SRS の品質として、表現、意味に加え、特定の読者に対する記述品質がプラグマティック品質として提案されてい

る[11]. これらを踏まえ, 本稿では, SRS の品質を次のように定義する.

定義[SRS 品質]: SRS の品質は, その想定読者のパースペクティブに対するプラグマティック品質である.

図 1 に SRS 品質のモデルを示す。SRS 参照品質とは、例えば、IEEE830[5]で規定されている 8 つの品質特性や REBOK[8]の 11 の品質特性である。SRS 品質は、これらの SRS 参照品質から読者のパースペクティブに適した品質特性に具体化したものである。

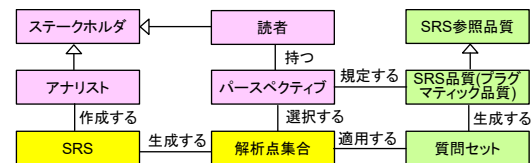


図 1 SRS の品質モデル

Fig. 1 A Quality Model of SRS

なお、この定義は、文献[14]において SRS のインスペクションのための品質特性として提案され、パースペクティブリーディングの方法と組み合わせられて利用されているプラグマティック品質の概念を、SRS の品質を評価するために一般化したものである。

4.2 SRS 品質解析のアプローチ

本稿で提案する SRS の品質解析は、上記の定義に基づき、文献[14]で提案された SRS のインスペクション設計方法論を、品質解析へ一般化し、自動化を図るものである。この SRS 品質解析を自動化するツールとして SRS 品質アナライザ ReqQA (Requirements Quality Analyzer)を提案する。

5. ReqQA アーキテクチャ

ソフトウェアによる SRS 品質解析を実現する SRS 品質アナライザ ReqQA のアーキテクチャを図 2 に示す。

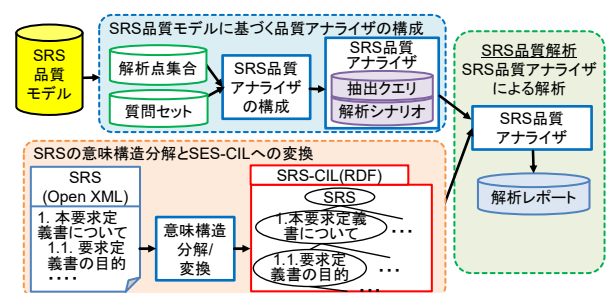


図 2 ReqQA のアーキテクチャ

Fig. 2 Architecture of ReqOA

ここで、前提条件として、解析対象の SRS は企業やプロジェクトごとに定められたテンプレートに則っており、内部が Open XML[6]で定義されている Word (Word 2003 以降) で記述されているものとする。また、こ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。

5.1 SRS-CIL: SRS の共通中間表現

SRS は企業やプロジェクトごとの構造の多様性がシステ

ムによる統一的な解析の妨げとなってきた。そこで、Word や Excel で記述された SRS を前提として、その内部表現である Open XML を意味構造に基づいて分解し、その意味構造を表現できる言語による表現へ変換する。この表現言語を SRS-CIL (Common Intermediate Language) と呼ぶ。

SRS-CIL は意味表現可能でオープンな標準言語である RDF[18]を用いて表現する。これにより、Web 上で特定ベンダによらず多様なツールによる解析が可能となる。

5.2 SRS 品質モデルに基づく SRS 品質アナライザの構成

SRS-CIL に変換された SRS を解析するための SRS 品質アナライザを構成する。SRS アナライザは、次の 2 つのコンポーネントから構成される。

- (1) 抽出クエリ: SRS 品質解析方法に基づく解析を行うため、SRS-CIL から解析すべき要素の集合である解析点集合に対して、質問セットの質問に応じた要素を抽出する SPARQL クエリの集合[19]
- (2) 解析シナリオ: 抽出した要素に対し解析目的に合わせて解析するために解析手順を記述したシナリオ

5.3 SRS 品質アナライザによる解析レポート生成

SRS 品質アナライザのシナリオを実行した結果を SRS の要素ごと、あるいは、品質特性ごとに適切な表現に変換し、レポートとして提示する。

6. ReqQA の実現

6.1 SRS のモデル

企業やプロジェクトごとに多様な表現をとる SRS を統一的に扱うために、図 3 に示すように、標準 SRS とその RDF 表現である参照 SRS-CIL を定義する。

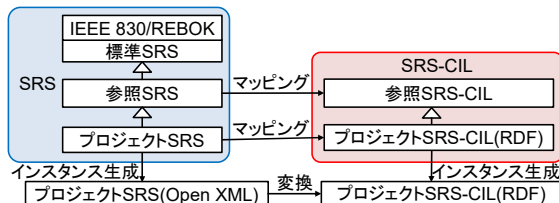


図 3 SRS-CIL のメタモデル
Fig. 3 Metamodel of SRS-CIL

標準 SRS として、本稿では、IEEE 830 と REBOK で定義されている SRS の構成を用いた。標準 SRS に基づき SRS とその管理情報であるメタデータを含めて一般化した表現として参照 SRS を定義した。この参照 SRS は IEEE 830 と REBOK と OSLC の要求管理の仕様に基づき定義している [2]。参照 SRS を企業やプロジェクト固有の SRS へ特殊化したものがプロジェクト SRS である。

参照 SRS の構造を RDF の語彙と名前空間で表現したものが参照 SRS-CIL である。従って、参照 SRS-CIL の要素は、SRS の内容とその意味を表現する対として RDF で表現される。語彙は参照 SRS の目次項目とその関係を定義する。参照 SRS-CIL の名前空間は OSLC[13]の名前空間を利用している。プロジェクト SRS-CIL は参照 SRS-CIL のプロ

ジェクト SRS へのマッピングである。

6.2 SRS リソースモデル

SRS を意味構造に基づき分割し、SRS-CIL のリソースとして定義する。RDF ではすべての表現された要素をリソースと呼ぶことから、本稿でも以下、リソースと呼ぶ。標準 SRS-CIL のリソース定義では OSLC Core と OSLC RM (Requirements Management)のリソースを基礎とすることから名前空間に Dublin core の dcterms に加え、oslc, oslc_rm も利用する。

標準 SRS-CIL のリソース定義の一部を図 4 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 プロパティ値として子リソースの URI を持ち、dcterms:description プロパティ値として、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 プロパティ値として SRS の本文を記述する。SRS 内の表から生成されたリソースは、oslc:element プロパティ値として表の要素名の URI を持つ。

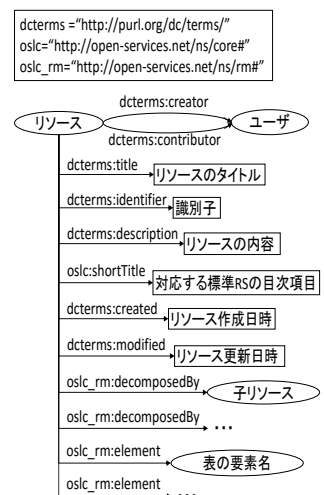


図 4 SRS-CIL リソース定義
Fig. 4 Resource Definition of SRS-CIL

6.3 SRS 品質解析プロセス

SRS-CIL を用いた SRS 品質解析プロセスを図 5 に示す。提案プロセスでは初めに解析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統一的な解析が繰り返し実行可能となる。

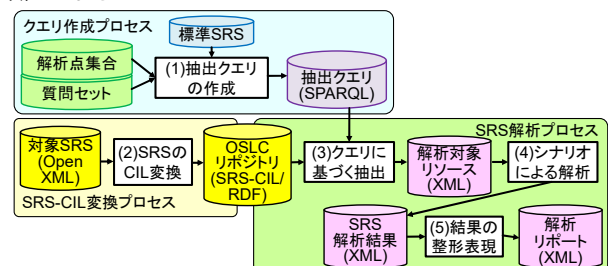


図 5 SRS-CIL を用いた SRS 品質解析プロセス
Fig. 5 Process of SRS Quality Analysis Based on SRS-CIL

(1) 抽出クエリの生成

解析点集合と質問セットを基に、解析に必要なリソースを抽出するクエリを生成する。クエリ言語は SPARQL[19]を用いる。

(2) SRS 文書の SRS-CIL への変換

Word 文書で記述された SRS を SRS-CIF へ変換する。変換した SRS-CIF は RDF リポジトリで管理する。

(3) クエリに基づく SRS のリソース抽出

上記(1)のクエリを用いて OSLC リポジトリ上の SRS-CIL 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の実行

上記(3)の結果に対し、質問セットの質問に基づいて品質を解析するためのシナリオに沿ってクエリを実行する。

(5) 解析レポートの生成

上記(4)の結果を人に理解しやすい形式に変換してレポートとして提示する。

6.4 SRS 文書の SRS-CIL への変換

SRS 文書の SRS-CIL への変換プロセスを図 6 に示す。

構造の異なる SRS を文献[2]に基づき定義された SRS プロパティを SRS-CIL へ変換する。さらに、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。

以下に各コンポーネントの詳細を示す。

(a) Word アダプタ

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 で表現された SRS-CIL へ変換する。

(c) リソース変換アダプタ

SRS テンプレートに対応付けられた標準 SRS の目次項目に基づいて、SRS-CIL 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。

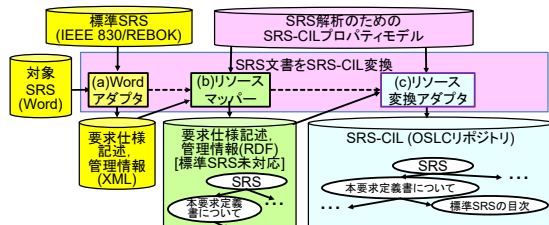


図 6 SRS 文書の SRS-CIL への変換プロセス
Fig. 6 Process of Converting from SRS to SRS-CIL

6.5 リソース抽出クエリの生成

解析に必要なリソースを抽出するため、解析点集合と質問セットを基に、リソース抽出クエリを作成し、SPARQL で記述する。リソース抽出クエリは PQC の項目と品質解析を行う標準 SRS の目次項目ごとに作成する(図 7)。

リソース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 7)。SRS-CIL に対して、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、解析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 8 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解析クエリの例である。

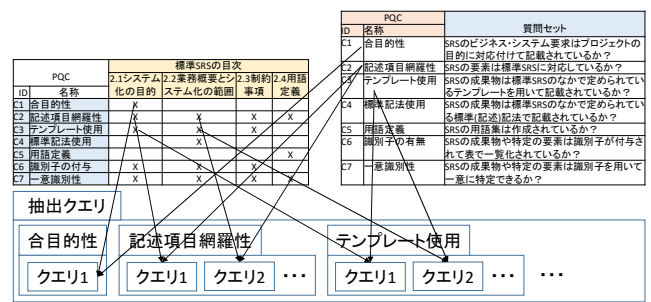


図 7 リソース抽出クエリの生成
Fig. 7 Generation of Resource Selection Query

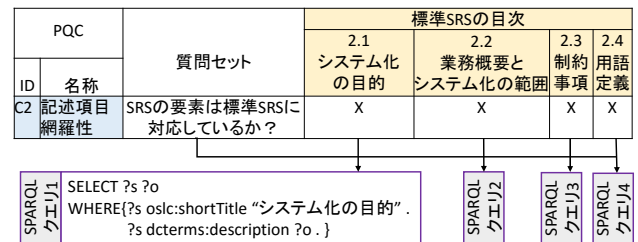


図 8 記述項目網羅性に関するリソース抽出クエリの導出
Fig. 8 An Example of Resource Selection Query

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。

6.6 シナリオに基づく SRS の解析

OSLC リポジトリ上の SRS-CIL に対して SPARQL の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 SRS の解析を行う(図 9)。

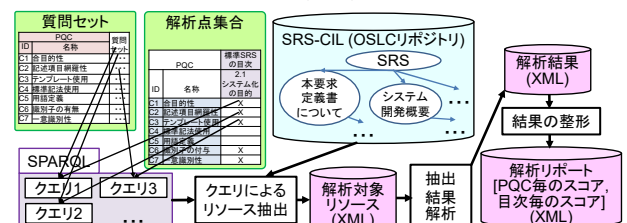


図 9 SRS 解析プロセス
Fig. 9 Process of Analyzing SRS-CIL

まず、OSLC リポジトリ上の SRS に対して SPARQL の抽出クエリを用いて解析に必要なリソースを抽出する。次に、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可否により、質問を満たしているかの判定を行う。

解析結果を整形し、解析レポートを作成する。解析レポートは人に理解しやすい表現にするため、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。品質スコアは未解析の質問を除いた総質問数に対し回答が Yes の質問の比率を示す。

6.7 品質アナライザの拡張

品質アナライザの機能を拡張し、SRS 内で用いられている曖昧用語を検出する機能を追加した(図 10)。

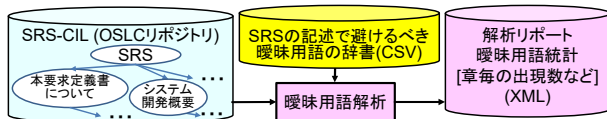


図 10 曖昧用語使用の解析
Fig. 10 Analysis of the Use of Ambiguous Terms

曖昧用語は文献[20]の「要求で避けるべき曖昧な用語」を用いて、曖昧用語辞書を CSV 形式で作成した。SRS-CIL の各リソースの内容を抽出し、それに対し曖昧用語辞書の用語が用いられているかチェックすることで SRS 内の曖昧用語を検出する。

検出結果は解析レポートとして提示する。解析レポートには、曖昧用語ごとの出現回数と対象 SRS の章ごとの曖昧用語の使用回数を算出し、記載する。解析レポートは XML 形式で出力している。

7. ReqQA プロトタイプの開発と例題への適用

SRS 品質アナライザ ReqQA の有効性を確認するために、プロトタイプを開発し、例題に適用して評価した。

7.1 プロトタイプの開発

ReqQA のプロトタイプを OSLC の参照実装である Eclipse Lyo[3]を拡張して RESTful Web サービスとして実現した。プロトタイプのシステム構成を図 11 に、その実装環境を表 1 に示す。Word アダプタ、SRS 解析器、曖昧用語解析器は Java、JSP を用いて実装した。リソースマッパー、リソース変換アダプタ、解析レポートジェネレータは Java で実装した。実装規模は Java が 879 (LOC), JSP が 68(LOC) である。以下、各コンポーネントの概要を示す。

(1) Word アダプタ

Word 形式の SRS を章、節ごとのまとまりに分割し、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

表 1 プロトタイプ実装環境
Table 1 Platform of SRS Analyzer

システム	バージョン
OS	Windows 8.1 64bit
JDK	JDK 1.6.0 45
Eclipse	Eclipse 4.4.1
Web Server	Jetty 7.3.0
RDF Store	Sesame 2.6.10

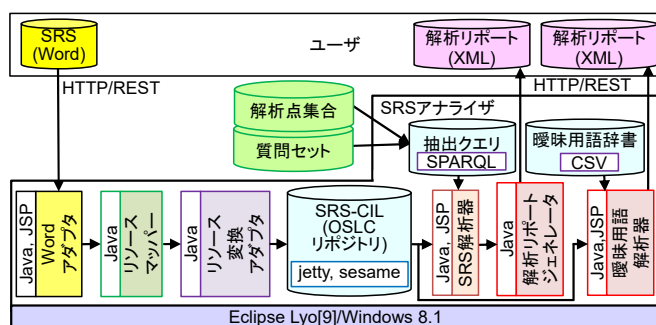


図 11 SRS 品質アナライザのプロトタイプの構成
Fig. 11 Configuration of the Prototype of SRS Analyzer

に変換する。XML 形式の SRS を章、節などの要素に分割し、URI を付与することで要素毎に RDF に変換する。

(3) リソース変換アダプタ

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 の要素が SRS-CIL のリソース型へ変換される。後述する RFP の例題で生成された SRS-CIL のリソースの一部を図 12 に示す。リソースは RDF リポジトリ Sesame[15]で管理している。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:oslc="http://open-services.net/ns/core#" xmlns:oslc_rm="http://open-services.net/ns/rm#"
xmlns:rio="http://open-services.net/rio/" xml:base="http://localhost:8080/rio-rm/requirement/11">
<oslc_rm:Requirement rdf:about="http://localhost:8080/rio-rm/requirement/11">
<rdf:type rdf:resource="http://open-services.net/ns/rm#Requirement"/>
<dcterms:title>1. 現行ホームページの概要</dcterms:title>
<oslc:shortTitle>section2_1</oslc:shortTitle>
<dcterms:description>現行システムは、平成 18 年 3 月に導入したもので、オープンソース CMS (Joomla!) を用い
てコンテンツを作成。FTP によるアップロードは行わず、ホームページアクセスがあった場合、CMS がリクエストを
受け、直接データベースからコンテンツを返す仕組みとなっている。現行 Web コンテンツ容量等(平成 22 年 10 月
31 日現在)
現行 Web サイト訪問者数(平成 21 年 11 月?平成 22 年 10 月のページビュー)
現行関連サービスの構成概要
現行ネットワーク環境
現行 IDC 内ネットワークシステムの機能概要
現行システム構成
</dcterms:description>
<dcterms:identifier>11</dcterms:identifier>
<dcterms:contributor rdf:resource="http://localhost:8080/rio-rm/_UNKNOWN_USER_"/>
<dcterms:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-01-
05T16:25:27.160+09:00</dcterms:modified>
<dcterms:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-01-
05T16:25:27.160+09:00</dcterms:created>
<dcterms:creator rdf:resource="http://localhost:8080/rio-rm/_UNKNOWN_USER_"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/12"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/18"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/22"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/35"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/38"/>
<oslc_rm:decomposedBy rdf:resource="http://localhost:8080/rio-rm/requirement/47"/>
<oslc_rm:stdTitle>調達内容・業務の詳細</oslc_rm:stdTitle>
<oslc_rm:stdTitle>現行システムとの関連</oslc_rm:stdTitle>
</oslc_rm:Requirement>
</rdf:RDF>
```

図 12 例題の SRS-CIL 表現(部分)
Fig. 12 SRS-CIL of the RFP (Part)

(4) SRS 解析器

SPARQL を用いた抽出クエリによって OSLC 上の SRS-CIL から解析に必要なリソースを抽出する。あらかじめ定義された抽出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各クエリに対する結果を一括して生成可能である。抽出したリソースは XML 形式で出力する。図 13 に一意特定可能性を解析するための SPARQL の抽出クエリの例を示す。

(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て解析を行う。すべての解析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」の場合は「○」、「No」の場合は「×」、未解析の質問は「?」を用いて表現する。

図 14 に後述する例題

の解析結果の出力例を示す。

この解析結果を人に分かりやすい表現に整形し、解析レポートを生成する。解析レポートは標準 SRS の目次項目ご

```
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX oslc_rm: <http://open-services.net/ns/rm#>
SELECT ?table ?elmt
WHERE {
  ?uri oslc_rm:stdTitle "提案の範囲";
    oslc_rm:decomposedBy ?table .
  ?table dc:shortTitle "table";
    oslc_rm:decomposedBy ?tableItem .
  ?tableItem oslc_rm:element ?elmt .
}
```

図 13 一意特定可能性を解析するための抽出クエリ
Fig. 13 SPARQL Query for Unique Identification

と PQC などの品質スコアを算出し、提示する。

(6) 曖昧用語検出器

SRS-CIL に対し、曖昧用語辞書を基に各リソースで使用されている曖昧用語の利用状況を解析する。曖昧用語は、文献[20]で定義されている「要求で避けるべき曖昧な用語」を用いた。曖昧用語辞書は CSV 形式で定義した。OSLC リポジトリの SRS-CIL 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辞書に記載されている用語が用いられているかチェックした。解析結果として、曖昧用語ごとの出現回数と対象 SRS の章ごとの曖昧用語の利用回数を算出し、解析レポートに提示する。

7.2 例題への適用

ReqQA の適用対象は SRS を想定しているが、SRS は一般に公開されていない。そのため、適用する例題を公開されている地方自治体のホームページリニューアルに関する RFP[10]とした。この RFP は 25 ページである。

```
<?xml version="1.0" encoding="Shift_JIS" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="result_style.xml"?>
<result>
  <R id="R1-1">
    <U id="U1-1">×</U>
    <U id="U1-2">?</U>
    <U id="U2-2">×</U>
    <U id="U3-1">○</U>
    <U id="U4-1">?</U>
  </R>
  <R id="R1-2">
    <U id="U1-1">○</U>
    <U id="U1-2">?</U>
    <U id="U3-1">○</U>
    <U id="U4-1">×</U>
  </R>
  <R id="R1-3">
    <U id="U1-2">?</U>
    <U id="U2-2">×</U>
    <U id="U3-1">○</U>
  </R>
  ...
</result>
```

図 14 解析結果の例
Fig. 14 An Analysis Result

一般に RFP の記述の標準化はされていない。このため、参照 SRS

に代わり「RFP 見本」[7]を参照 RFP として利用した。解析点集合と質問セットとして文献[12]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 を評価するための質問セットを利用した。この結果、RFP の品質評価も文献[12]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を用いた(表 2)。

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。

表 2 RFP のプラグマティック品質
Table 2 Pragmatic Quality of RFP

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

文献[12]で定義した質問セットは質問ごとにペルソナの視点による差分がある。これは、パースペクティブとして定義されるものである。本稿では、ペルソナごとの詳細な品質解析は必要ではないと判断し、複数のペルソナの視点をまとめてパースペクティブとして定義した。また、RFP 内の図に関する質問も解析の対象とはしていない。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く 68 の質問を有効質問とした。解析結果を図 15 に示す。

この図に示すように、68 個の質問セット中、52 個の質問

ID	プラグマティック品質	有効質問数	解析可能なプラグマティック品質	質問数
U1-1	システム目的の独立性	3	システム目的の独立性	3
U1-2	業務要求のシステム目的への適合	5	要求の独立性	3
U2-1	定量的具体性の有無	4	標準 SRS との整合性	42
U2-2	要求の独立性	3	文書の参照関係の明示	2
U3-1	標準 SRS との整合性	42	一意に特定可能	1
U3-2	文書の参照関係の明示	2	ランク付けの有無	1
U3-3	例外要求の網羅	0	合計	52
U3-4	変更可能性の明記	0	解析不可能なプラグマティック品質	質問数
U3-5	一意に特定可能	1	業務要求のシステム目的への適合	5
U3-6	用語集の存在	0	定量的具体性の有無	4
U4-1	ランク付けの有無	1	用語の整合	4
U4-2	用語の整合	4	制約条件と要求の整合	3
U4-3	動作の整合	0	合計	16
U4-4	制約条件と要求の整合	3		
合計		68		

図 15 解析結果
Fig. 15 Results of Analysis

が解析可能であった。ここで、「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」は図に対する解析が必要なため、これらの解析は対象外とした。さらに、文献[20]に基づく 75 種類の曖昧用語の利用解析の結果を表 3 と表 4 に示す。14 種類の曖昧用語の利用を検出した。

RFP 入力から解析結果出力までの実行時間は約 5 分であった。

表 3 対象 RFP の章ごとの曖昧用語検出結果
Table 3 Chapter-by-Chapter Usage Distribution of Ambiguous Terms in the Example of RFP

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 4 対象 RFP の頻出曖昧用語
Table 4 The Most Frequent Use of Ambiguous Terms

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

8. 評価と考察

8.1 評価方法

SRS 品質アナライザ ReqQA を評価するため、次の 3 つの評価尺度を定義した。

- (1) 網羅率: SRS の解析項目中で解析できた項目の比率。

- (2) 有効率: SRS の解析項目中で正しい結果を得た比率.
- (3) 品質スコア: プラグマティック品質尺度に対する SRS 全体の包括的な品質. 式(3)で定義する.

$$\text{網羅率} = \frac{\text{評価可能質問数}}{\text{有効質問総数}} \quad (1)$$

$$\text{有効率} = \frac{\text{想定した結果を得た質問数}}{\text{解析した質問総数}} \quad (2)$$

$$\text{品質スコア} = \frac{\text{Yesの数}}{\text{評価質問総数}} \quad (3)$$

8.2 評価可能性に関する考察

提案方法による評価の網羅率は 76.4%であった. 評価できたプラグマティック品質は以下の 6 項目である.

- (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は提案方法による評価が可能であると言える. しかし, 独立性に関する品質の評価は個々の目的, 要求が分割されているかという限定的な評価に留まった. これは質問をより具体化し, 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は, 現在の実装においては評価が不可能であった.

- (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間での対応がとれず, 評価が不可能となった.

- (2) リソースの値の有無

評価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の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の評価が困難である. また, RFP では用語集との関連が確認できないため, 用語集に関する評価も不可能となった.

- (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の確認ができず, 評価が不可能となった. これらのプラグマティック品質の評価では, 評価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確認が不可能となる共通の問題が明らかとなった. これに対しては, 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, 評価可能になると考えられる.

8.3 プラグマティック品質特性ごとの解析に関する考察

プラグマティック品質特性ごとの品質スコアを表 5 と図 16 に示す. 評価対象の 6 つのプラグマティック品質の網羅率は 100%であり, 漏れなく解析できている. このことから, 解析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解析が可能であると言える. これらのプラグマティック品質はシステムによる自動解析が可能であると言える.

表 5 プラグマティック品質ごとの品質スコア

Table 5 Pragmatic Quality Score

ID	総数	解析数	網羅率[%]	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

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」は解析数が 1 のため品質スコアの精度が低いと考えられる. 今後, リソース定義を詳細化し, 解析の細粒度化による品質スコアの精度向上が必要である.

8.4 目次項目ごとの解析に関する考察

標準 RFP の章ごとの品質スコアを表 6 と図 17 に示す.

「R3: 提案手続き」, 「R4: 開発に関する条件」, 「R6: 契約事項」は未解析数が 0 である. このことから, これらの章に対する ReqQA による解析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」は未解析項目があり, 質問総数に対して未解析の割合が高い. 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, 未解析項目の低減が必要である.

Word 形式の RFP を SRS-CIL で定義した結果, リソース間の対応やリソースの構造に関する品質評価が可能となった. これにより, 質問セットに基づく評価が, 記述の有無だけでなく, RFP の構造や意味に基づく評価が可能となっ

プラグマティック品質ごとの品質スコア

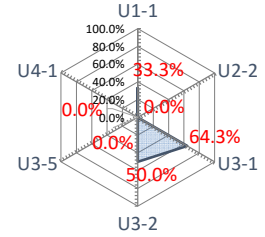


図 16 プラグマティック品質ごとの品質スコアグラフ
Fig. 16 Distribution of Pragmatic Quality Score

評価対象RFPの品質スコア

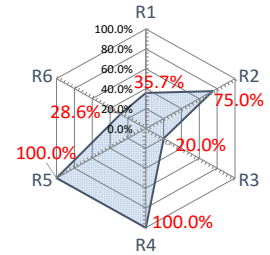


図 17 章ごとのプラグマティック品質スコアグラフ
Fig. 17 Distribution of Pragmatic Quality Score by Chapter

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評価が可能となった。SRS-CIL はオープンな標準に基づき、かつ、RDF を用いた意味表現が可能であることから、関連研究[1, 17]と比べ、SRS の中間表現として適していると考えられる。

表 6 章ごとのプラグマティック品質スコア
Table 6 Pragmatic Quality Score by Chapter

章 ID	総数	解析数	Yes 数	未解析数	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

提案方法による解析では、網羅率が 100%とならず、限定的な解析と未解析に留まった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解析は、リソースを解析可能な意味要素とその関係となる適切な粒度へ分割し、リソースへ対応づけることで評価可能になると考えられる。このためには、SRS-CIL とその RDF リソース定義を見直す必要があると考えている。

提案した品質解析方法により、ソフトウェアによる RFP に対する構造、表現の評価が可能となった。自動化により人手によらない解析が可能となる。提案方法は SRS の品質評価のコストの削減と品質の向上が期待できる。

8.5 曖昧用語解析結果に対する考察

表 3 に示す結果から章により曖昧用語の出現に偏りがあることが明らかになった。これは、章の内容、著者などに起因すると考えられる。また、表 4 に示す結果は特定の曖昧用語が頻用される傾向があることを示唆している。これは、曖昧用語解析によって特定の章の執筆、あるいは、著者ごとの曖昧用語の利用を抑制するための効果的なアドバイスを提供できることを示唆している。

8.6 提案解析システムの拡張性についての考察

提案解析システムに SPARQL のクエリとシナリオの追記のみで拡張を行い、曖昧用語の検出機能を実現した。これは、オープンな標準に準拠した共通中間言語である SRS-CIL の効果である。

9. 今後の課題

9.1 実際の SRS への適用と実践

実際の SRS に提案方法を適用し、品質改善効果を確認する。あわせて、ReqQA を実開発へ適用するために改善する。

9.2 網羅率の向上

評価不可能であった 4 つのプラグマティック品質に関する質問の具体化、SRS 要素の粒度の細分化とそのリソース定義を行い、網羅率を向上する。

9.3 SRS 規模の増大に対するスケーラビリティの向上

SRS の規模増大に伴い、変換した RDF のリソース数も

増大し、SPARQL クエリの処理時間が増加すると考えられる。処理のスケーラビリティと効率化を検討する。

10. まとめ

従来、実践における重要性は指摘されていながら研究が進んでいなかった SRS の品質解析について、SRS インспекション設計方法論と OSLC の成果を発展させ、品質の自動解析の方法と解析システム ReqQA を提案した。

本提案の意義は、SRS の多様性を扱える RDF を用いた意味構造を表現できる中間言語 SRS-CIL とその上での解析方法の自動化にある。これにより、研究課題を解決できる方法とツールを具体的に示した点に意義がある。提案方法により SRS の品質向上が期待できる。今後、ReqQA を実際の SRS へ適用し、実用性を評価する。

謝辞

本研究は(株)NTT データの斎藤忍氏らとの共同研究から発展したものである。斎藤氏と関係各位に感謝する。

参考文献

- 1) D. de Almeida Ferreira, et al., RSL-IL: An Interlingua for Formally Documenting Requirements, Proc. MoDRE 2013, IEEE, Jul. 2013, pp. 40-49.
- 2) 青山 幹雄 ほか, 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, 近代科学社, Dec. 2013, pp. 263-272.
- 3) Eclipse Lyo, <http://eclipse.org/lyo/>.
- 4) G. Fanmuy, et al., Requirements Verification in the Industry, Proc. of CSDM 2011, Springer, Dec. 2011, pp. 145-160.
- 5) IEEE Std. 830-1998, IEEE Recommended Practice for Software Requirements Specification, IEEE, 1998.
- 6) ISO/IEC 29500-1:2012, Information Technology - Document Description and Processing Languages - Office Open XML File Formats - Part 1: Fundamentals and Markup Language Reference, 2012.
- 7) 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.
- 8) JISAREBOK 企画 WG(編), 要求工学知識体系, 近代科学社, 2011.
- 9) A. Katsanov, et al. Requirements Quality Control: A Unified Framework, Requirements Eng. J., Vol. 11, No. 1, Mar. 2006, pp. 42-57.
- 10) 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>.
- 11) J. Krogstie, et al., Towards a Deeper Understanding of Quality in Requirements Engineering, Proc. of CAiSE '95, LNCS Vol. 932, Springer, 1995, pp. 82-95.
- 12) 森下 月菜, 青山 幹雄, 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спек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.
- 13) OSLC, <http://open-services.net/>.
- 14) S. Saito, et al., RISDM: A Requirements Inspection Systems Design Methodology, Proc. of RE 2014, IEEE, Aug. 2014, pp. 223-232.
- 15) Sesame, <http://rdf4j.org/>.
- 16) F. Shull, et al., How Perspective-Based Reading Can Improve Requirements Inspections, IEEE Computer, Vol. 33, No. 7, Jul. 2000, pp. 73-79.
- 17) A. R. da Silva, Quality of Requirements Specifications, Proc. SAC 2014, ACM, Mar. 2014, pp.1021-1022.
- 18) W3C, RDF 1.1 Primer, <http://www.w3.org/TR/rdf11-primer/>.
- 19) W3C, SPARQL 1.1 Query Language, <http://www.w3.org/TR/sparql11-query/>.
- 20) K. Wiegers, Software Requirements, 3rd Ed., Microsoft Press, 2013.