RDF を用いたソフトウェア要求仕様書分析方法の提案と評価

M2013SE009　中根　拓也

指導教員　青山　幹雄

1. はじめに

要求定義の成果物である SRS(ソフトウェア要求仕様)の品質を確保するためにインスペクションが広く実施されている. しかし, 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を要し, 誤りも多くなる.

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, SRS 解析方法を提案する.

2. 研究課題

(1)　SRS の意味構造分解

SRS は企業ごとに構造が多様であり, システムによる統一的な検証が困難である. そのため, 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必要がある.

(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で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 で表現することで RDF 形式の SRS に対する検証を可能とする.

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表現形式である[8]. RDF のクエリ言語に SPARQL (SPARQL Protocol And RDF Query Language)がある[9]. SPARQL は RDF グラフとのパターンマッチにより検索を行う.

3.2. OSLC に基づく要求管理方法と支援環境の提案

OSLC[6]と IEEE 830 に基づく SRS のプロパティを定義し, RDF へ変換し, Web を介した SRS の管理を実現した[1].

3.3. SRS のインスペクションデザイン方法論の提案

SRS の満たすべき品質として PQC (Pragmatic Quality Characteristic)を定義し, PQC に基づくインスペクション方法が提案されている[7]. さらに, PQC と標準 SRS の目次項目を対応付けたインスペクションポイントセットと, インスペクションで確認すべき内容を Yes/No で回答可能な質問形式で策定した質問セットの作成方法が提案されている.

4. アプローチ

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.

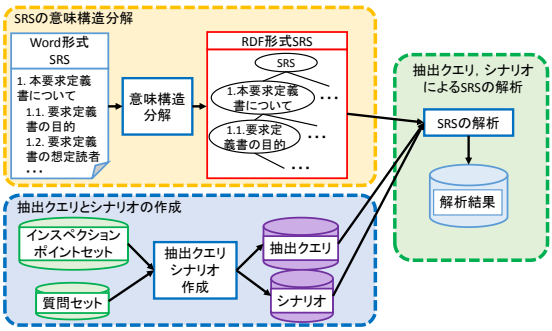

図 1　SRS 解析方法のフレームワーク

4.1. SRS の意味定義表現

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.

4.2. 抽出クエリとシナリオの作成

共通表現に変換された SRS に対し, 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抽出クエリとシナリオを作成する.

4.3. 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 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とでシステムによる検証を可能とする.

4.4. 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 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と IEEE 830 で定義されている標準 SRS の目次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. 文献[9]において 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, 標準 SRS に基づいて定義された参照 SRS として用いる. 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.

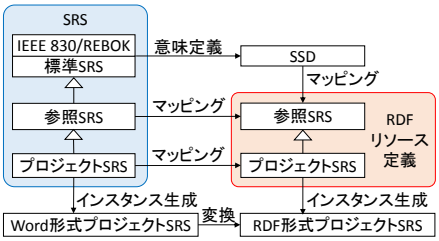

図 2 SRS の意味定義モデル

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. 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.

5.3. RDF を用いた SRS 解析プロセス

RDF を用いた SRS 解析プロセスを図 3 に示す. 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.

(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.

(2) SRS の RDF 変換

Word 文書で記述された SRS を 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.

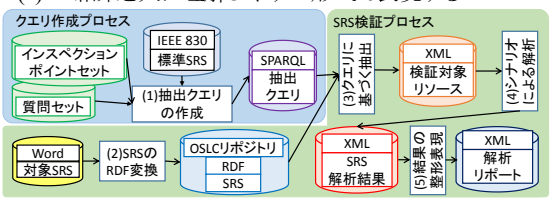

図 3 RDF を用いた SRS 解析プロセス

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. 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 4). 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. 同様にして，他の PQC に対応するクエリを作成する.

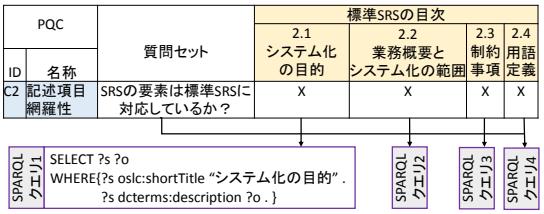

図 4 SPARQL を用いた抽出クエリの導出

5.5. SRS の RDF 変換

図 3 の SRS の RDF 変換の詳細を図 5 に示す. 文献[1]に基づき拡張した SRS プロパティを基に，構造の異なる SRS を 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. これにより構造が異なる SRS に対して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 SRS を任意の XML 形式に変換する.

(b) リソースマッパー

XML 形式に変換された SRS を 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意味付けを行う.

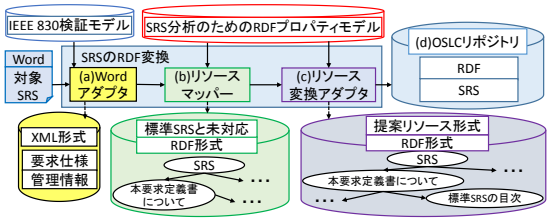

図 5 SRS の変換プロセス

5.6. シナリオに基づく SRS の解析

抽出クエリとシナリオを用いた SRS の解析を行う(図 6).

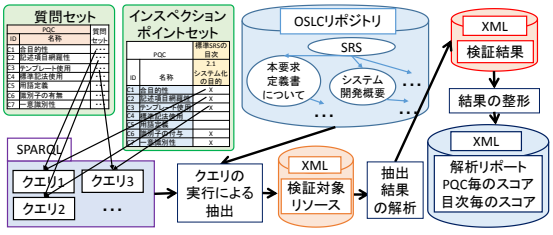

図 6 SRS の解析

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．

解析結果を整形し，解析リポートを作成する．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し，解析リポートに記載する．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．曖昧な用語は，文献[12]の「要求で避けるべきあいまいな用語」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．

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．

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構成を図 7 に示す．

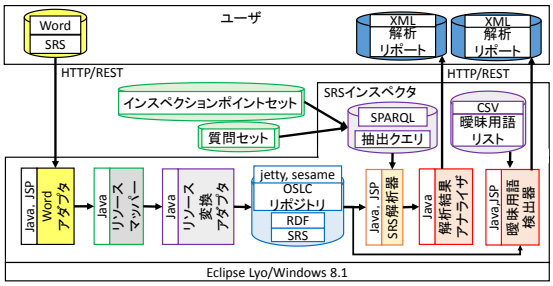

図 7 SRS インスペクタ

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．

6.2. 例題への適用

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)[4]とした．また，参照 SRS に代わり RFP/SLA 見本[3]を参照 RFP として利用し，文献[5]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し，RFP の品質評価も文献[5]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．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．

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．また，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．以下に考察を示す．

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 = 検証可能質問数 / 有効質問総数 (1)

検証率 = 想定した結果を得た質問数 / 検証数 (2)

品質スコア =Yes 数 / 検証数 (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．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
(2) 定量的具体性の有無
(3) 用語の整合
(4) 制約条件と要求の整合

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り，リソースの細分化により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ごと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 1 に示す．

表 1 プラグマティック品質ごとの品質スコア

	総数	検証数	検証率	Yes 数	品質スコア
システム目的の独立性	3	3	100%	1	33.3%
要求の独立性	3	3	100%	0	0.0%
標準 SRS との整合性	42	42	100%	27	64.3%
文書の参照関係の明示	2	2	100%	1	50.0%
一意に特定可能	1	1	100%	0	0.0%
ランク付けの有無	1	1	100%	0	0.0%
合計	52	52	100%	29	55.8%

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り，システムによる自動検証が可能であると言える．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, 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.

7.3. 目次項目ごと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 2 に示す.「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 であり, 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.

表 2 章ごとの品質スコア

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

7.4. 曖昧用語検出結果に対する考察

曖昧用語の検出結果を表 3 と表 4 に示す. 表 3 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. 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.

表 3 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 4 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

7.5. 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, システムによる統一的な検証が可能となった. システムによる検証を行うことで, 属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.

7.6. 提案システムの拡張性についての考察

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

(1) SRS への適用と実践

実際の SRS に提案方法を適用した場合の品質改善の効果を確認する必要がある.

(2) 検証網羅率の向上

検証不可であった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, 検証網羅率を向上させる必要がある.

9. まとめ

SRS の品質の自動検証実現のため, SRS 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, 抽出クエリとシナリオを用いて検証することで SRS に対しシステムによる自動検証を実現した. 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] 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.

[4] 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.

[5] 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.

[6] OSLC (Open Services for Lifecycle Collaboration), http://open-services.net/.

[7] 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.

[8] W3C, RDF 1.1 Primer, http://www.w3.org/TR/rdf11-primer/.

[9] W3C, SPARQL 1.1 Query Language, http://www.w3.org/TR/sparql11-query/.

[10] K. Wiegers, Software Requirements, 3rd Ed., Microsoft Press, 2013.