OSLC に基づく分散リソースに対する統一的変更管理方法の提案と評価

M2012MM002　朝倉　知也

指導教員　青山　幹雄

1. はじめに

近年，ソフトウェアの大規模化，複雑化により，アウトソーシングやオフショア開発などの分散開発が進展している．分散開発では開発リソースが各拠点に分散するため，PMBOK などが定義するソフトウェア構成管理を各拠点で実施，連携して分散リソースに対する変更要求を管理する必要がある．しかし，変更要求の共通的な表現や，複数拠点で連携管理するための統一的方法は確立していない．

2. 研究課題

2.1. Web を介して管理可能な変更要求モデルが未定義

OSLC[3]は，変更要求の Web を介した管理を実現するためのリソースプロパティを定義している．しかし，OSLC の変更要求のプロパティは OSLC 独自の仕様であり，IEEE Std.828-2005[2]が定義する標準規格のプロパティと異なる．

2.2. 変更要求の統一的な時系列管理が困難

各拠点で発生する変更要求を統一的に管理するためには，発生した変更要求を OSLC リソースとして時系列で管理し，相互参照可能にする必要がある．しかし，OSLC リソースを時系列管理する変更管理システムのアーキテクチャは未確立であり，管理することは困難である．

3. 関連技術

(1) IEEE Std.828-2005

IEEE Std.828-2005 は，ソフトウェア構成管理の計画に関する標準規格である．この標準規格では，変更要求の持つべきプロパティが定義されている．

(2) OSLC(Open Services for Lifecycle Collaboration)

OSLC は，Web を介した開発ツール間の連携を実現するためのリソースプロパティを，CM(Change Management)などのドメイン毎に定義している．OSLC リソースは RDF で表現され，REST を用いて操作する．

(3) TRS(Tracked Resource Set)[4]

TRS は，対象リソース群の追加や更新などの変更の時系列管理を目的とした仕様である．TRS は，対象リソース群の変更情報を時系列で管理する Change Log と，時系列管理対象外のリソースを列挙する Base で構成される．変更は event として Change Log に追加する．event は event URI，変更の種類，発生番号，対象リソース URI で構成される．

4. アプローチ

本研究では共通的な変更要求を表現するため，IEEE Std.828-2005 が定める変更要求プロパティを用いて変更要求の共通モデルを定義する．この共通モデルのプロパティを OSLC CM が定義するプロパティにマッピングすることで OSLC リソースに変換することができ，変更要求の共通モデルの Web を介した管理を可能にする．また，複数拠点間で変更要求を時系列管理するため，変更要求の共通モデルから変換した OSLC リソースを時系列管理対象とする TRS を用いた分散変更管理システム(DCMS: Distributed Change Management System)のアーキテクチャを提案する(**図 1**)．TRS の event の URI にタイムスタンプとして ISO 8601 の拡張表記を用いることで，他拠点との時系列に基づいた TRS 連携を可能にする．

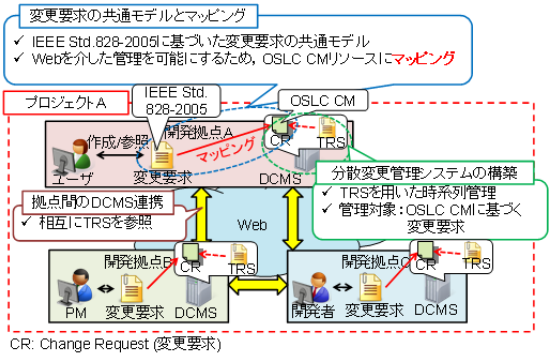

図 1　変更要求の Web を介した時系列管理方法

5. 変更要求の共通モデルの定義

5.1. プロパティマッピング

IEEE Std.828-2005 が定義する変更要求のプロパティと OSLC CM が定義する変更要求のプロパティの意味を対応させ，相互の変換を可能にする．IEEE Std.828-2005 が定義する変更要求の全プロパティを OSLC CM の変更要求プロパティにマッピングした結果を**表 1** に示す．

表 1　IEEE Std.828-2005 と OSLC CM のマッピング

IEEE Std.828-2005	OSLC CM	IEEE Std.828-2005	OSLC CM
作成者	dcterms:creator	番号	dcterms:identifer
作成日	dcterms:created	状態	oslc_cm:state
分類	dcterms:subject	優先度	oslc_cm:priority
説明/必要性	dcterms:description, dcterms:title	関連リソース	Relationship properties

5.2. 変更要求の共通モデル

プロパティマッピング結果から，IEEE Std.828-2005 が定義するプロパティを全て満たすための OSLC CM プロパティを抽出した(**図 2**).

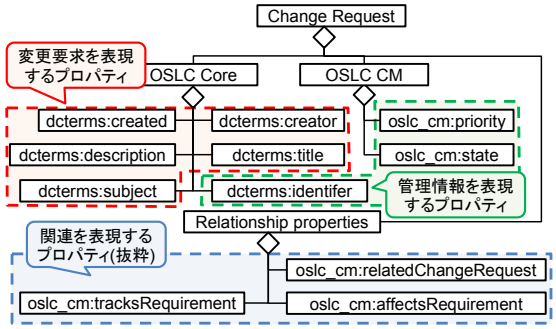

図 2 管理対象の変更要求モデル

(1) 変更要求を表現するプロパティ

変更要求の作成者や作成日，分類，説明など，変更要求の内容を表現する.

(2) 管理情報を表現するプロパティ

変更要求の番号や状態，優先度など，変更要求を管理するための情報を表現する.

(3) 関連を表現するプロパティ

変更要求の実施により影響を与えるリソース，変更要求からトレースされる要求仕様，関連する変更要求など，変更要求に関連するリソースを表現する. このプロパティは全て URI で記述され，関連リソースを参照できる.

6. DCMS アーキテクチャの提案

変更要求の共通モデルを管理対象とする，DCMS のアーキテクチャを提案する.

6.1. DCMS のユースケース

提案する DCMS のユースケースを**図 3** に示す.

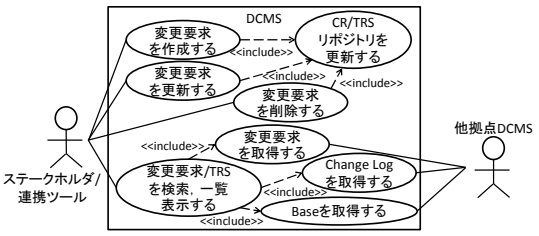

図 3 DCMS のユースケース

アクタはステークホルダ/連携ツール，他拠点 DCMS の二つである. ステークホルダ/連携ツールは HTTP メソッドを用い，変更要求の作成などを行う. 作成などの実行後，その変更を event として Change Log を更新する. また，他拠点 DCMS の Change Log を取得し，マージ，ソートすることで全拠点の変更要求の時系列変更一覧を作成できる.

6.2. DCMS のアーキテクチャ

提案する DCMS のアーキテクチャを**図 4** に示す.

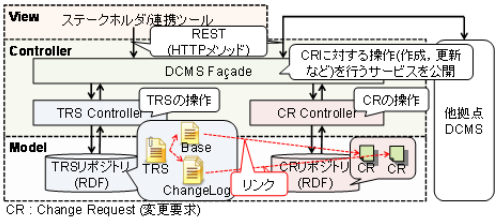

図 4 DCMS アーキテクチャ

DCMS は MVC アーキテクチャに基づき，View，Controller，Model で構成する.

(1) View

ステークホルダはブラウザを用い，変更要求が管理できる. また，公開する Web API を用いることで，連携ツールや他拠点 DCMS と連携できる.

(2) Controller

外部に公開する変更要求処理 API を定義する Façade，TRS の操作を行う TRS Controller，CR の操作を行う CR Controller で構成する. 変更要求の作成やプロパティの追加など様々な処理に対応するため，一連の操作をまとめた高レベルな API を提供する Façade を用いる. ステークホルダや連携ツールから発生したリクエストに対し Façade で定義された適切な処理が実行され，CR と TRS が自動更新される.

(3) Model

TRS を管理する TRS リポジトリと，CR を管理する CR リポジトリで構成される. これらのリソースは RDF で表現されるため，RDF リポジトリを用いる.

6.3. DCMS の振舞い

(1) 変更要求作成の振舞い

DCMS の変更要求作成の振舞いを**図 5** に示す.

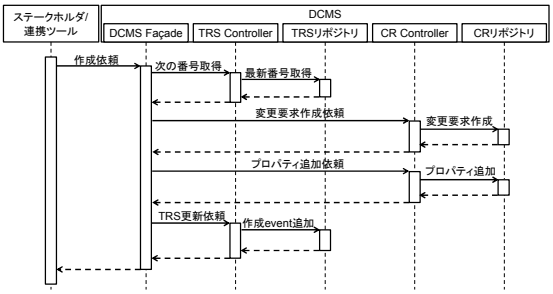

図 5 変更要求作成の振舞い

ステークホルダ/連携ツールは Façade が定義する変更要求作成 API を用いることで，変更要求を作成できる. ステークホルダ/連携ツールは変更要求作成依頼として変更要求のタイトルや説明などを記述し，送信する. Façade は TRS Controller を介して次の event 番号を取得し，CR Controller を介して変更要求リソースを作成する. 次に，TRS Controller を介し Change Log リソースに event を追加する.

(2) 時系列変更一覧作成の振舞い

時系列変更一覧作成の振舞いを**図 6** に示す.

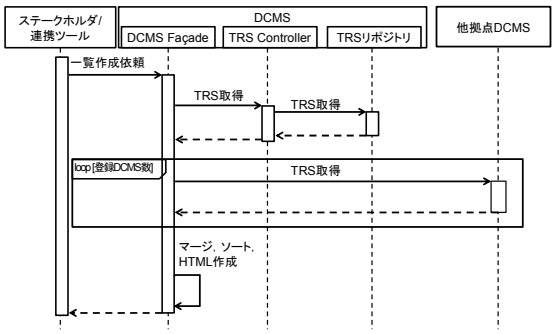

図 6 時系列変更一覧作成の振舞い

前提条件として，他拠点 DCMS の URI は事前に登録されているとする．ステークホルダ/連携ツールからの一覧作成依頼により，Façade で実装される時系列変更一覧作成 API が実行される．自拠点の TRS に加えて登録されている他拠点の TRS を取得，マージし，時系列順にソートして出力する．TRS の Change Log リソースや Base リソースには対象変更要求の URI が記述されているため，取得したい event を選択して変更要求の実データにアクセスできる．

7. プロトタイプの開発

7.1. プロトタイプ開発の目的

以下の項目を検証するため，プロトタイプを開発した．

(1) 変更要求の Web を介した時系列管理

変更要求の共通モデルを OSLC リソースとして作成し，Web を介して管理可能か検証する．また，複数の DCMS で並行して変更要求を作成，更新し，時系列に基づいた連携管理が可能か検証する．

(2) 他ツールとの連携

連携ツールとして，プロジェクト管理アプリケーションである Redmine を用いる．本研究で定義した変更要求の共通モデルを，Redmine のチケットとして登録可能か検証する．

7.2. プロトタイプの構成

プロトタイプの構成を**図 7** に示す．

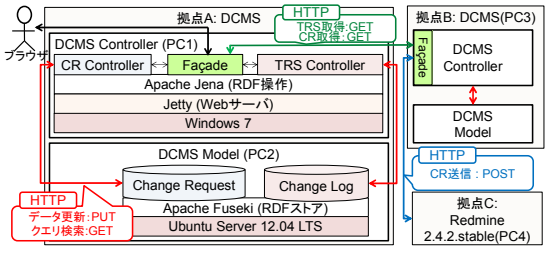

図 7 プロトタイプの構成

プロトタイプの DCMS は Eclipse Lyo 上の TRS のリファレンス実装を基に作成した．Web サーバに Jetty，RDF モデル操作に Apache Jena，RDF リポジトリに Apache Fuseki[1]を用いた．変更要求連携確認のため，拠点 A，拠点 B にプロトタイプを配置した．また，Redmine との連携には Redmine の REST API を用い，変更要求を送信した．

8. 例題への適用

8.1. ユースケース

SWEBOK の変更要求処理プロセス(**図 8**)に従い，ブラウザ又は REST で変更要求を作成，更新，参照する．

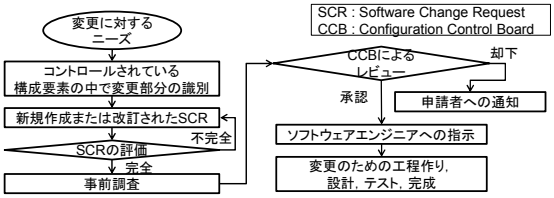

図 8 SWEBOK の変更要求処理プロセス

拠点 A，拠点 B でそれぞれ 50 件の変更要求をランダムに作成，更新し，各プロセス終了時に TRS を参照して時系列変更一覧を取得する．拠点 A ではブラウザを用いて，拠点 B では REST を用いて変更要求の作成などを行う．また，拠点 B では Redmine と連携し，変更要求の作成，更新が発生する毎に Redmine の REST API を用いて変更要求を送信し，チケットとして登録する．拠点 A で変更要求を管理するアクタを開発者 A，拠点 B で変更要求を管理するアクタを開発者 B，拠点 B から時系列変更一覧の取得と Redmine のチケットを確認するアクタをマネージャとする．例題のユースケース記述を**表 2** に示す．

表 2 例題のユースケース記述

名称	分散変更要求管理
アクタ	開発者A(拠点A)，開発者B(拠点B)， マネージャ(拠点B)
事前条件	拠点A，BのDCMSが連携可能
事後条件	マネージャが時系列変更一覧を取得
基本フロー	1. 2から5を繰り返す 2. 開発者Aはブラウザを用いて変更要求を作成又は更新 3. 開発者BはRESTを用いて変更要求を作成又は更新 4. マネージャが時系列変更一覧を取得 5. マネージャがRedmineを確認

8.2. プロトタイプへの適用

(1) 変更要求の作成と更新

開発者 A はブラウザを用いて，開発者 B は REST を用いて変更要求を作成，更新する．変更要求の共通モデルの必要とするプロパティを満たすため，変更要求のタイトルと説明，分類，作成者，優先度を入力する．また，任意で関連リソースを入力する．開発者 B は REST を用いた変更要求の作成，更新を実行するため，Firefox アドオンの REST Client を用いる．変更要求を作成，更新することで，各拠点で自動的に Change Log リソースが更新される．作成される変更要求リソースの例を**図 9** に示す．

図 9 管理対象の変更要求リソース

(2) 時系列変更一覧の取得

マネージャは拠点 B の DCMS を用い，登録済みの全ての DCMS で発生した event を時系列で取得する．マネージャがブラウザを用いて拠点 B の時系列変更一覧作成依頼を行うと，拠点 B の DCMS は拠点 A の TRS を取得し，拠点 B の TRS とマージ，ソートした結果を HTML として出力する．開発者 A，B が変更要求の作成，更新を行う毎にマネージャは時系列変更一覧を取得し，発生した event が時系列で表示され，event の対象リソース URI から対象の変更要求を取得可能であることを確認した．

(3) Redmine との連携確認

開発者 B が変更要求を作成，更新する毎に，変更要求の共通モデルのプロパティを Redmine が定義するチケットスキーマに変換し，Redmine に対してチケットの登録，更新を行った．そして，マネージャは開発者 B が変更要求の作成，更新をする毎に Redmine のチケット情報を取得し，登録を確認した．

9. 評価と考察

9.1. 時系列変更一覧取得方法の比較

TRS を用いない場合は，対象の変更要求リソースを全て取得して更新時間を抽出後，マージとソートを行う必要がある．しかし，本研究で提案した手法を用いることで TRS に記述された変更 event のみを取得できるため，個々の変更要求リソースに対する取得処理と解析処理，抽出処理が削減できる(**図 10**).

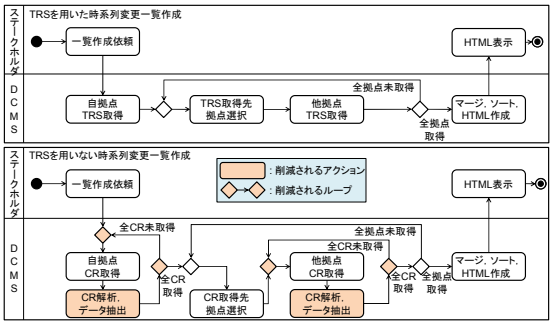

図 10 時系列変更一覧取得方法の比較

また，本研究で定義した変更要求の共通モデルは OSLC リソースとして表現する．TRS の変更対象リソースに変更要求の共通モデルの URI を用いることで，時系列変更一覧から変更要求リソースの URI を取得できる．そのため，各開発拠点に分散する変更要求リソースを，Web を介した統一的な方法で横断的に管理することが可能である．

9.2. 他ツールとのデータ連携の評価

変更要求の共通モデルのプロパティと Redmine のチケットフィールドをマッピングすることで，Redmine と任意の OSLC 対応ツール間で相互にデータ交換が可能になる．その結果，Redmine とバージョン管理システムなどの連携ツールで構成されるプロジェクトマネジメント情報システムの変更要求管理を行うシステムとして運用でき，分散開発におけるプロジェクトマネジメントを支援することができる．

10. 今後の課題

10.1. DCMS 連携のセキュリティ

DCMS が変更要求や TRS を連携する際には，認証や暗号化によるセキュリティ機能が必要である．しかし本研究で提案したアーキテクチャではセキュリティを考慮していないため，セキュリティ機能を実現する必要がある．

10.2. 時系列変更一覧の RDF 出力

時系列変更一覧を他の連携ツールから参照し機械的処理可能にするため，RDF をレスポンスする Web サービスとして実装する必要がある．

10.3. 変更要求と Redmine チケットのマッピング

プロトタイプではタイトルや説明，優先度など汎用的なプロパティはマッピング可能であったが，プロジェクトが独自に定義するプロパティは対応できない．そのため，プロジェクト毎の差異を吸収するマッピング方法が必要である．

11. まとめ

本研究では課題として，Web を介して管理可能な変更要求の共通モデルが未定義であること，OSLC リソースを時系列に基づいて統一的に管理することが困難であることを挙げた．この課題に対し，IEEE Std.828-2005 と OSLC CM のプロパティをマッピングし，Web を介して管理可能な標準規格に基づいた変更要求の共通モデルを定義した．更に，TRS を用いた分散変更管理システムのアーキテクチャを提案し，複数拠点間で変更要求の時系列管理を可能にした．また，プロトタイプを作成し例題に適用することで，提案アーキテクチャの妥当性を検証した．

参考文献

[1] Fuseki, Serving RDF Data over HTTP, http://jena.apache.org/documentation/serving_data/.
[2] IEEE Std.828-2005, IEEE Standard for Software Configuration Management Plans, IEEE, 2005.
[3] OSLC (Open Services for Lifecycle Collaboration), http://open-services.net/.
[4] TRS (Tracked Resource Set), September 4, 2013, http://open-services.net/wiki/core/TrackedResourceSet-2.0/.