PROMCODE

次世代プロジェクト管理データ交換アーキテクチャ協議会

インタフェース仕様書

第 1 版

2013 年 10 月 22 日

南山大学
日本アイ・ビー・エム株式会社
富士通株式会社
日本電気株式会社
株式会社 NTT データ
株式会社日立製作所
株式会社野村総合研究所

目次

1　概説

本書は，次世代プロジェクト管理データ交換アーキテクチャ協議会が提案する PROMCODE アーキテクチャに基づき，PROMCODE ドメインモデル仕様と PROMCODE サービス仕様を規定する．

1.1 解決すべき課題

協調的プロジェクト管理における管理データのリアルタイムな交換を阻害する主な要因はプロジェクトごとに異なる管理データモデルにある．

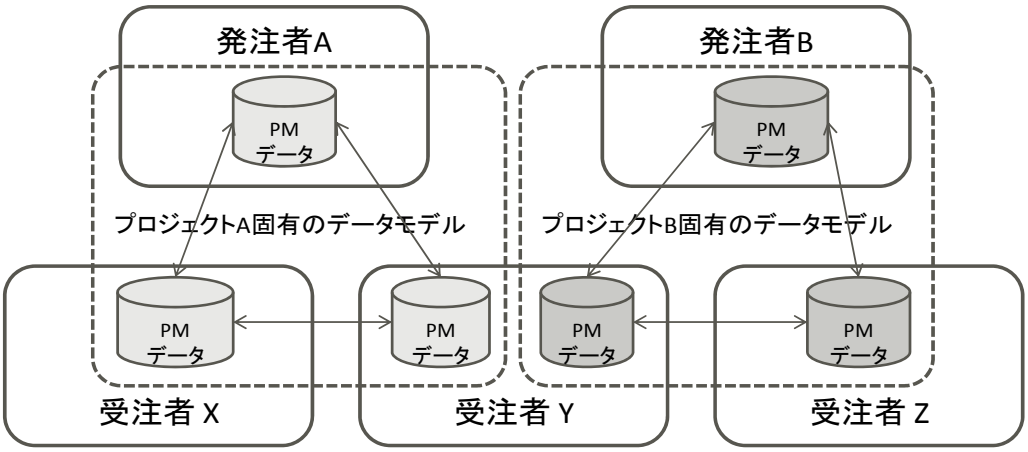

図 1　プロジェクトごとに異なる管理データモデル

例えば図 1 では，A 社と B 社がそれぞれ発注者として管理する二つのプロジェクトに Y 社が参画している．Y 社は，プロジェクト A と B の異なる管理データモデルに個別に対応する必要があるため，プロジェクト管理ツールを導入するメリットを得にくい．結果として，プロジェクトごとに異なるデータ形式のスプレッドシートを使って，手作業で管理するケースが多くなる．図 1 は 2 階層のみを示しているが，実際の大規模プロジェクトでは受発注関係が多階層に及び，参画する組織も多数であり，プロジェクト全体の間接工数が多大なものになっている．

組織間の管理データの相違は，大きく二種類に分類できる．一つ目は管理対象の違いであり，各社の開発プロセスの定義が異なるためである．例えば，作業工程の粒度や階層構造，名称が異なり，それらによって生産される成果物も異なり，その目的，内容，名称が異なっている．

二つ目は形式の違いである．二次元テーブル形式という共通点はあるものの，行と列の定義が異なる．予定と実績を行と列のどちらに配置するか，プロジェクト全体の情報を一つのテーブルで管理するか複数に分割するか，など様々なバリエーションがある．さらに，一つの企業内でも必ずしも統一されず，部門ごと，プロジェクトごとにバリエーションがある．

このような状況で，例えば図 1 のプロジェクト A 向けのツールを，プロジェクト B 向けにカスタマイズするのは容易ではない．その理由は，第一に，物理的なテーブル形式に管理データの意味構造が束縛されているため，異なる形式どうしを対応付けることが困難だからである．第二に，プロジェクトごとに異なる開発プロセスを採用する以上，異なる管理対象物どうしを対応付けることもまた困難だからである．

1.2 アプローチ

管理データの物理形式に束縛されず，任意の管理対象に対応付けが可能な，共通のインタフェースを策定する必要がある．プロジェクト管理データは，本来は，多種類の管理対象からなる多次元の情報空間を構成すると考えられる．それをプロジェクトごとに異なる方法で二次元に射影したものが現状のデータ構造であり，そのために共通構造を見出すことが難しい．本来の意味構造を適切に表現できる形式が利用できれば，共通インタフェースを設計し易いと期待できる．

Linked Data 技術は意味構造をリンクとして柔軟に定義できることから，任意の意味構造に基づいたデータ交換が共通のインタフェースで実現できる．例えば，図 1 のシナリオでは，受注者 Y が Linked Data に基づくインタフェースを実装する開発ツールを採用すれば，発注者 A と B のそれぞれ異なる意味構造を持つ管理データを交換できる．

さらに，プロジェクト間のデータモデルのバリエーションを吸収できるように，適切な抽象度のドメインモデルを定め，それに基づいてリソースの標準スキーマを定義する．これによって，例えば受注者 Y は，管理データを発注者 A 向けと発注者 B 向けのリソースに容易に変換できようになる．

1.3 PROMCODE モデリングフレームワーク

図 2 は，ドメインモデルとリソース定義の関係，及び実プロジェクトにおけるそれらの利用方法を示すフレームワークである．

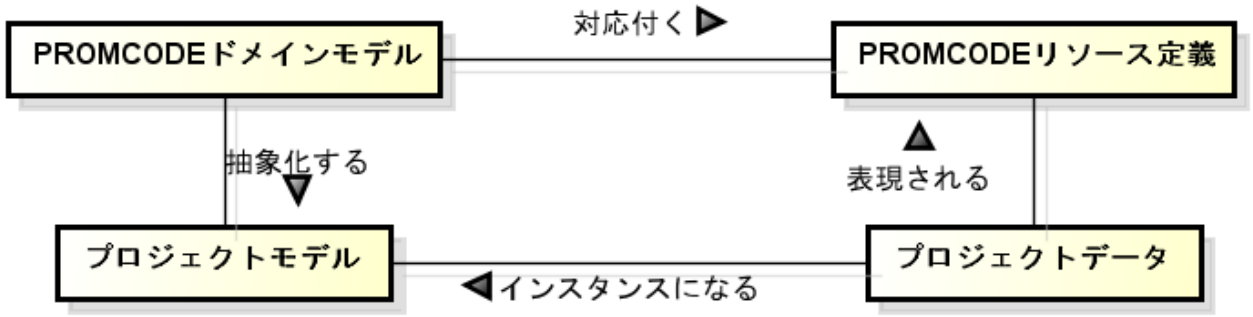

図 2　PROMCODE モデリングフレームワーク

このフレームワークにおいて，「PROMCODE ドメインモデル」は，協調的プロジェクト管理ドメインの抽象データモデルである．

「PROMCODE リソース定義」は，PROMCODE ドメインモデルに基づき，プロジェクト管理データを Linked Data リソースとして交換するためのスキーマを規定する．各リソース定義は，ドメインモデルの各クラスから 1 対 1 にマッピングして定義する．

「プロジェクトモデル」はプロジェクト個別の具体的なデータモデルであり，PROMCODE ドメインモデルが定義する抽象クラスのサブクラス群としてプロジェクトごとに定義する．

プロジェクトモデルのインスタンスデータが「プロジェクトデータ」となる．プロジェクトデータは，PROMCODE ドメインモデルの抽象クラス群のインスタンスとなるので，PROMCODE リソース定義に基づく Linked Data としてそのまま表現でき，リンクや交換が可能となる．

フレームワークの左半分はデータ交換の実装技術から独立したモデルであり,右半分は Linked Data という特定の実装技術によるその表現を表している.モデルと表現を分離することにより，Linked Data 以外の実装技術にも容易に対応できる. また,フレームワークの上半分はプロジェクトの違いを吸収する抽象レベルのモデルと表現であり,下半分は個々のプロジェクト毎のモデルと交換データを表している.

本仕様書は上半分の二つ,「PROMCODE ドメインモデル」と「PROMCODE リソース定義」を規定する.個々のプロジェクトはこれらを利用して,固有のドメインモデルを策定するだけで,具体的なプロジェクト管理データの Linked Data 表現を決めることができる.この結果,プロジェクト管理ツールが PROMCODE リソース定義を一度実装すれば, これを通して実際のプロジェクト管理データが容易に交換・収集できる．特に，複数の発注者に管理データを提出する場合でも，発注者ごとのデータ変換が不要なため，同じツールがカスタマイズすることなく利用できるようになる.

2　PROMCODE ドメインモデル仕様書

2.1 ドメインモデル

図 3 に PROMCODE ドメインモデルを示す.

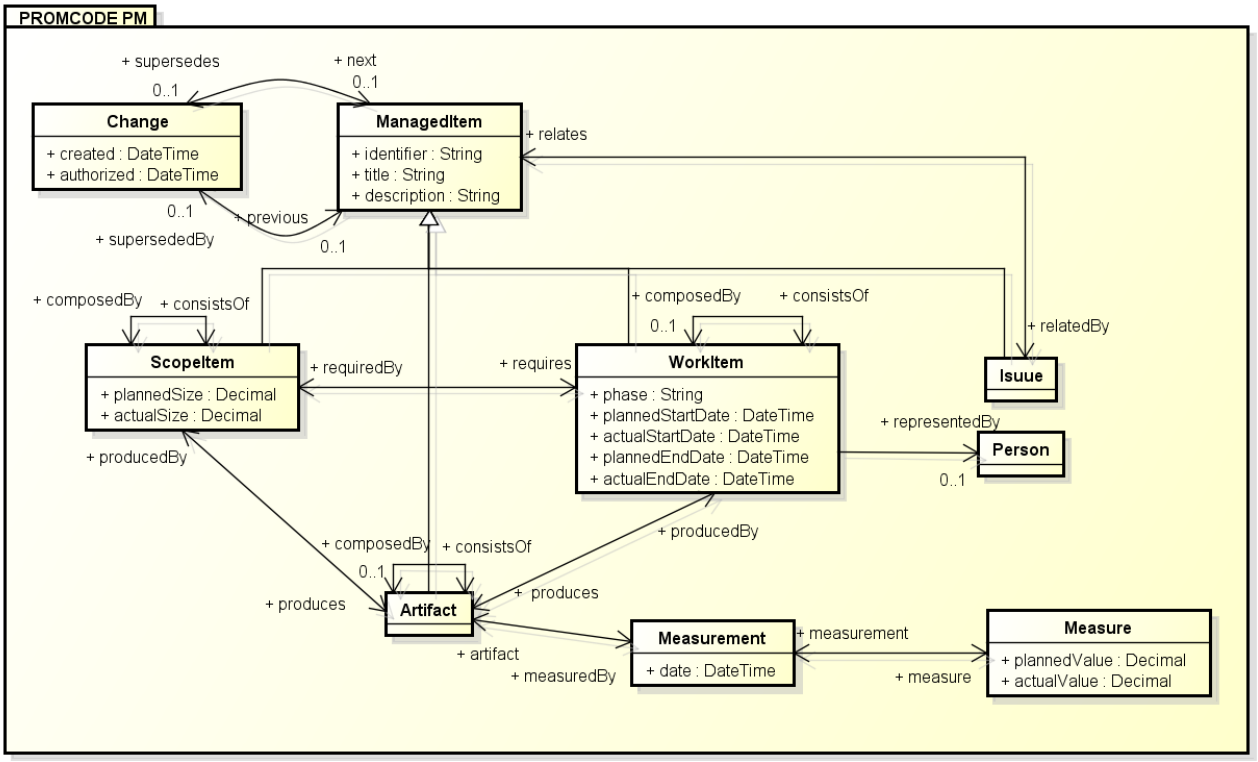

図 3　PROMCODE ドメインモデル

リンクの多重度は表記していないものは「*」である．属性の多重度は煩雑になるために省略しているが原則として「0..1」であり，例外的に「1」（必須）のものについては以下で述べる．

以下では各クラスを説明する．

2.1.1　ScopeItem

協調的開発における発注者(顧客)と受注者間で取り交わされる開発契約に対して，開発対象ソフトウェアが提供する「価値の単位」を表現するオブジェクトである．例えば要求，提供機能，ユースケース，画面などがある．

ScopeItem は開発作業ではないので，開始・終了というイベントを持たない．また，ScopeItem は成果物ではないので，品質を測定可能な実体を持たない．

ScopeItem は，顧客や発注者が想定する開発のスコープを，発注者が具体的に管理できる粒度に分解したものである．ScopeItem は，発注者と受注者の間で発注の規模を見積もるために利用するので，その１つ１つの規模を見積もれるものである.「何を ScopeItem とするか」やそれぞれの見積もり規模は，発注者と受注者の間の合意で決定し，管理対象やその見積もり規模が変わる場合にも合意が必要である.

ScopeItem は，より細粒度の ScopeItem に分解でき，より詳細な管理ができる．逆に，粒度が大きな ScopeItem を，下位の ScopeItem を分類するだけのために導入する場合がある．

顧客の視点から管理することは，協調的プロジェクト管理における本質的な特性である．

【スーパークラス】

- ManagedItem

【属性】

- planedSize: Decimal [0..1]
  発注者と受注者の間で合意した規模の見積もり値．単位は任意．
- actualSize: Decimal [0..1]
  発注者と受注者の間で合意した規模の実際値．

【リンク】

- composedBy: ScopeItem [0..1]
  上位の ScopeItem.
- consistsOf: ScopeItem [*]
  下位の ScopeItem.
- requires: WorkItem [*]
  この ScopeItem を開発するのに必要な WorkItem.
- produces: Artifact [*]
  この ScopeItem を開発する際に生産する Artifact.

2.1.2 WorkItem

プロジェクトを遂行する上で受注者が実施する「活動」を表現するオブジェクトである．例えば分析，設計，実装，テストのような開発工程を表すものもある．一方，設計書執筆，レビュー，指摘反映のような具体的な作業を表すものもあり，様々な粒度のオブジェクトが存在する．

WorkItem は，ScopeItem を実現するために，あるいは，Artifact を生産するために必要な作業を，具体的に管理できる粒度に分解したものである．

WorkItem は開始と終了という二つのイベントが存在する点で共通性があり，それらの予定と実績の日付を比較することで進捗を管理する．

WorkItem は，より細粒度の WorkItem に分解でき，より詳細な管理ができる．逆に，粒度が大きな WorkItem を，下位の WorkItem を分類するだけのために導入する場合がある．

【スーパークラス】

- ManagedItem

【属性】

- phase: String [0..1]
  工程としての分類名．例えば，「分析」「設計」「実装」という工程がある場合，これらを WorkItem の三種類のサブクラスと考えることもできるが，「工程」というサブクラスであってその phase が三種類あると考えてもよい．phase を使うことで具象サブクラスの数が膨大にならずに済む．
- plannedStartDate: DateTime [0..1]
  開始日の計画値.
- actualStartDate: DateTime [0..1]
  開始日の実績値.
- plannedEndDate: DateTime [0..1]
  終了日の計画値.
- actualEndDate: DateTime [0..1]
  終了日の実績値.

【リンク】

- representedBy: Person [0..1]
  この WorkItem の進捗に責任を持っている人．この WorkItem の実際の担当者とは限らない.
- composedBy: WorkItem [0..1]
  上位の WorkItem.
- consistsOf: WorkItem [*]
  下位の WorkItem.
- requiredBy: ScopeItem [0..1]
  この WorkItem を必要とする ScopeItem.
- produces: Artifact [*]
  この WorkItem によって生産する Artifact.

2.1.3 Artifact

開発プロジェクトの成果物として作成される「具体的な実体」を表現するオブジェクトである．例えばプログラムやモジュールのように計算機上で動作するものもある．仕様書，設計書，テスト結果報告書などのドキュメントもある.

Artifact は，ScopeItem を実現するために，WorkItem が生産する.

Artifact は，何らかの指標を測定して品質を評価できる点に共通性がある．品質指標の中には，一度測定するだけではなく日々測定してその変化を管理する場合もある.

Artifact は, より細粒度の Artifact に分解でき, より詳細な管理ができる. 逆に, 粒度が大きな Artifact を，下位の Artifact を分類するだけのために導入する場合がある.

【スーパークラス】

- ManagedItem

【リンク】

- composedBy: Artifact [0..1]
  上位の Artifact.
- consistsOf: Artifact [*]
  下位の Artifact.
- producedBy: WorkItem [*]
  この Artifact を開発するのに必要な WorkItem.
- producedBy: ScopeItem [*]
  この Artifact を開発することで生み出される ScopeItem.
- measuredBy: Measurement [*]
  この Artifact を測定する Measurement.

2.1.4 ManagedItem

ScopeItem，WorkItem，Artifact の三種の管理単位を抽象化したスーパークラスである.

【属性】

- identifier: String [1]
  オブジェクトの識別子.
- title: String [1]
  オブジェクトの名称.
- description: String [0..1]
  オブジェクトの内容を記述するテキスト.

【リンク】

- relatedBy:Issue [*]
  関連する Issue.
- supersedes: Change [0..1]
  この ManagedItem に置き換わる前の ManagedItem を指す Change．Change.previous は，この ManagedItem と同じサブクラスの ManagedItem でなければならない.
- supersededBy: Change [0..1]
  この ManagedItem に置き換わった後の ManagedItem を指す Change．Change.next はこの ManagedItem と同じサブクラスの ManagedItem でなければならない.

2.1.5 Change

ManagedItem の履歴を管理する.

【属性】

- created: DateTime [0..1]
  ManagedItem の変更日.
- authorized: DateTime [0..1]
  ManagedItem の変更の承認日.

【リンク】

- previous: ManagedItem [0..1]
  置き換わる前の ManagedItem．next の ManagedItem と同じサブクラスの ManagedItem でなければならない．previous が無い Change は，ManagedItem の生成を意味する.
- next: ManagedItem [0..1]
  置き換わった後の ManagedItem. previous の ManagedItem と同じサブクラスの ManagedItem でなければならない．next が無い Change は，ManagedItem の削除を意味する.

2.1.6 Issue

To Do やバックログなど，ManagedItem に関わる課題を表すクラスであり，今後，課題管理に利用することを想定している.

【スーパークラス】

- ManagedItem

【リンク】

- relates: ManagedItem [*]
  関連する ManagedItem.

2.1.7 Measure

Artifact の品質測定値を表現する.

【属性】

- plannedValue: Decimal [0..1]
  測定の計画値.
- actualValue: Decimal [0..1]
  測定の実績値.

【リンク】

- measurement: Measurement [1]
  この Measure を測定した Measurement.

2.1.8 Measurement

Artifact の品質の測定日を表現する.

【属性】

- date: DateTime [1]
  測定日.

【リンク】

- measure: Measure [*]
  この Measurement で測定した Measure. 一度の Measurement で複数の Measure を測定できる.
- measures: Artifact [1]
  この Measurement の対象である Artifact.

2.2 プロジェクトモデルのサンプル

2.2.1 進捗管理への適用例

表 1 に典型的な進捗管理テーブルの例を示す. 進捗の管理単位は「サブ機能」であり,「機能」を分割したものである. 各サブ機能は分析, 設計, 実装工程を通じて開発される. 表 1 は管理テーブルの本質的構造のみを示し, 実際の管理テーブルはより多くの列から成り, 機能-サブ機能構造は数レベルの木構造を持つ.

表 1 進捗管理テーブルの例

機能	サブ機能		分析		設計		実装	
			開始	終了	開始	終了	開始	終了
A	A1	予定	6/4	6/11	6/12	6/19	6/20	6/27
		実績	6/4	6/10	6/11	6/19	6/20	6/26
	A2	予定	6/4	6/11	6/12	6/19	6/20	6/25
		実績	6/4	6/12	6/13	6/20	6/21	6/26
B	B1	予定	6/4	6/11	6/12	6/19	6/20	6/27
		実績	6/4	6/12	6/13	6/20	6/21	6/28

表 1 に相当するプロジェクトモデルの例を図 4 に示す. Function（機能）と SubFunction（サブ機能）は ScopeItem のサブクラスであり, Analysis（分析）, Design（設計）, Coding（実装）は WorkItem のサブクラスである. Function は 1 つ以上の SubFunction に分解され（consistsOf）, さらに三種類の WorkItem に分解される(requires). このプロジェクトモデルにより表 1 のすべての管理データは, PROMCODE ドメインモデルが定める抽象クラスである ScopeItem, WorkItem のインスタンスとして表現できる.

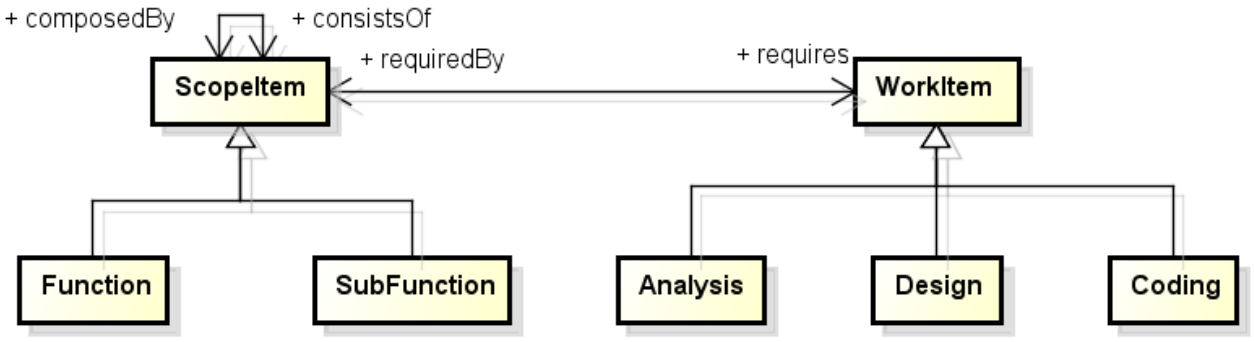

図 4 進捗管理のプロジェクトモデルの例

2.2.2 品質管理への適用例

表 2 に典型的な品質管理テーブルの例を示す．品質の管理単位は「モジュール」である．モジュールは「要求」で分類されている．各モジュールは，行数，テストケース数，障害数などの Measure で品質を測定する．

表 2 品質管理テーブルの例

要求	モジュール	コード行数		テストケース数		障害数	
		計画	実績	計画	実績	計画	実績
R1	M1-1	2,000	2,130	60	62	5	5
	M1-2	1,500	1,450	45	43	3	2
R2	M2-1	2,000	1,980	60	65	5	4
	M2-2	1,000	950	30	35	2	2

表 2 に相当するプロジェクトモデルを図 5 に示す．Requirement（要求）は ScopeItem のサブクラスであり，Module（モジュール）は Artifact のサブクラスである．Measure のサブクラスは LOC(行数)，#TC(テストケース数)，#Defect(障害数)の三種類がある．Requirement は，それを実現するために開発する Module に分解される(produces)．このプロジェクトモデルにより表 2 のすべてのプロジェクト管理データは，PROMCODE ドメインモデルが定める抽象クラスである ScopeItem，Artifact，Measure のインスタンスとして表現できることになる．

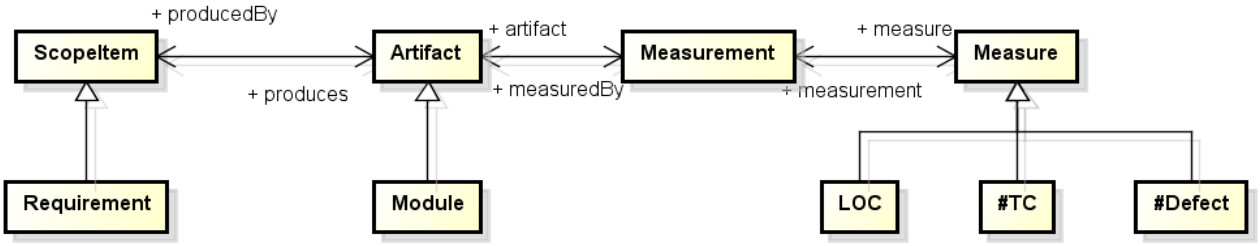

図 5 品質管理のプロジェクトモデルの例

3　PROMCODE サービス仕様書

3.1 概要

PROMCODE サービスは PROMCODE ドメインモデルに基づいて記述されたプロジェクトデータを交換するためのサービスである.

PROMCODE サービスは OSLC(Open Services for Lifecycle Collaboration)[1] の OSLC Core 仕様バージョン 2.0 バージョン 2.0 [2] に基づくものとし,PROMCODE ドメインモデルを RDF リソースにマップした PROMCODE リソース定義,および PROMCODE リソースに対して HTTP プロトコルによるアクセスを可能とするための PROMCODE サービスプロバイダを規定する.

3.2 基本仕様

以下のテーブルは OSLC Core 仕様バージョン 2.0 からの要求に対して PROMCODE サービスが提供するサービスの概要を示す.
以下のキーワードは RFC-2119[3] で定義されているものとして扱うものとする.
"MUST", "MUST NOT", "REQUIRED, SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", "OPTIONAL"

Requirement	Level	Meaning
Unknown properties and content	MAY	PROMCODE サービスプロバイダは未知のコンテントを無視してもよい.
	MUST	PROMCODE クライアントは未知のコンテントを保持しなければならない.
Resource Operations	MUST	PROMCODE サービスプロバイダは標準の HTTP プロトコルを通してリソースに対する操作を提供しなければならない.
Resource Paging	MAY	PROMCODE サービスはリソースのページ分割を提供してもよい.
Partial Resource Representations	MUST	PROMCODE サービスプロバイダは HTTP GET の URL パラメータに oslc.properties が指定された場合リソースの指定されたプロパティのみを返さなければならない.
Partial Update	MAY	PROMCODE サービスプロバイダは patch を用いることのよる部分更新をサポートしてもよい.
Service Provider Resources	MUST	PROMCODE サービスプロバイダはサービスプロバイダリソースを提供しなければならない.
	MAY	PROMCODE サービスプロバイダはサービスプロバイダカタログを提供してもよい.
Creation Factories	MAY	PROMCODE サービスプロバイダは HTTP POST メソッドによるリソースの作成を行うリソース作成ファクトリーを提供してもよい.
Query Capabilities	MUST	PROMCODE サービスプロバイダはリソースの検索を行うためのクエリ機能を提供しなければならない.
Query Syntax	MUST	クエリ機能は OSLC Core Query Syntax をサポートしなければならな

Requirement	Level	Meaning
		い.それに追加して他のクエリシンタックスをサポートしてもよい.
Delegated UI Dialogs	MAY	PROMCODE サービスプロバイダは委譲 UI（リソースの作成用およびリソースの検索用）を提供してもよい
UI Preview	MAY	PROMCODE サービスプロバイダはプレビュー用の UI を提供してもよい
HTTP Basic Authentication	MAY	PROMCODE サービスプロバイダは HTTP Basic 認証をサポートしてもよい.その場合には HTTPS を利用すべきである.
OAuth Authentication	MAY	PROMCODE サービスプロバイダは OAuth 認証をサポートしてもよい.
Error Responses	MAY	PROMCODE サービスプロバイダは OSLC Core 仕様バージョン 2.0 で定義されたエラーフォーマットを利用してエラー情報を返信してもよい.
RDF/XML Representations	MUST	PROMCODE サービスプロバイダは HTTP GET リクエストに対して RDF/XML 形式での返信はサポートしなければならない.
	SHOULD	PROMCODE サービスプロバイダは HTTP の POST および PUT リクエストに対して,RDF/XML 形式での送信をサポートしてもよい.
XML Representations	MAY	PROMCODE サービスプロバイダは HTTP GET, POST および PUT リクエストに対して OSLC Core 仕様バージョン 2.0 のガイドラインに基づいた XML 形式での通信をサポートしてもよい.
JSON Representations	MAY	PROMCODE サービスプロバイダは HTTP GET, POST および PUT リクエストに対して OSLC Core 仕様バージョン 2.0 のガイドラインに基づいた JSON 形式での通信をサポートしてもよい.
HTML Representations	MAY	PROMCODE サービスプロバイダは HTTP GET リクエストに対して HTML 形式での返信をサポートしてもよい.

3.2.1 仕様バージョニング

OSLC Core 仕様バージョン 2.0 の仕様バージョニングを参照.

3.2.2 名前空間

PROMCODE のリソース定義では OSLC Core 仕様バージョン 2.0 で定義されている oslc, rdf[4], dcterms[5], foaf[6] の各接頭語と対応する名前空間を利用し,PROMCODE の定義するリソースに関しては名前空間として **http://promcode.org/ns/pm#**,その標準接頭語として promcode_pm を用いる.

3.2.3 リソースフォーマット

PROMCODE 仕様ではリソースフォーマットとして以下のフォーマットを用いるものとする.

リソースに対しての HTTP GET リクエストにおけるフォーマット：

1. PROMCODE リソースのプロバイダーは RDF/XML 形式の提供をしなければならない（MUST）.RDF/XML 表現に関しては,OSLC Core のリソース表現ガイダンス を参照.

リソースに対しての,HTTP PUT/POST リクエストにおけるフォーマット：

2. PROMCODE リソースのプロバイダーは RDF/XML 表現を受け取ってもよいとする（MAY）.

クエリに対しての,HTTP GET リクエストにおけるフォーマット：

3. PROMCODE リソースのプロバイダーは RDF/XML 表現での提供をしなければならない（MUST）.

3.2.4 認証

リソースへのアクセスの際の認証に関しては,OSLC Core 仕様バージョン 2.0 の認証セクションを参照.

3.2.5 エラーレスポンス

OSLC Core 仕様バージョン 2.0 のエラーレスポンス セクションを参照.

3.3 リソース定義

3.3.1 リソース定義

図 6 は PROMCODE ドメインモデルの各クラスを RDF リソースにマップしたものを示す.

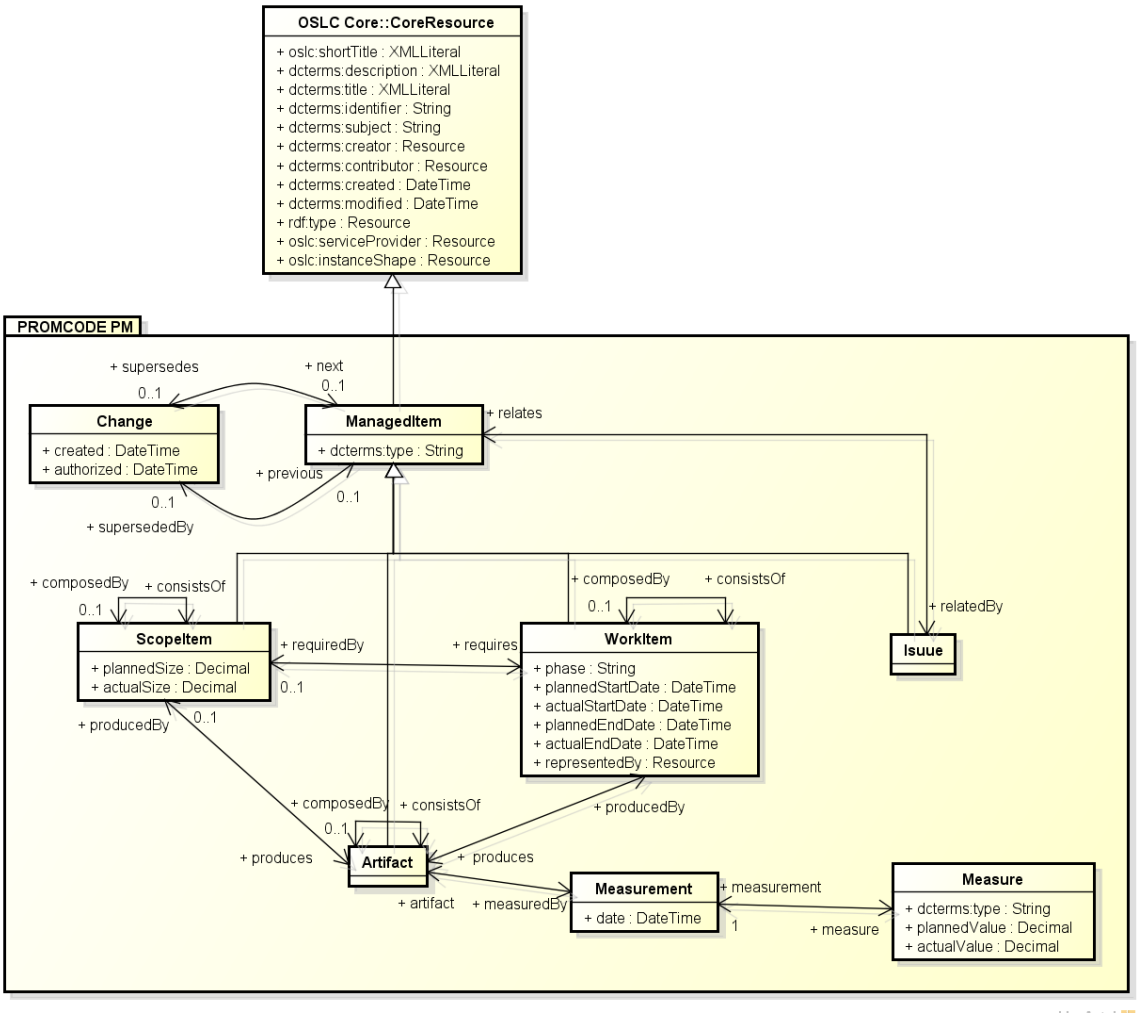

図 6 PROMCODE リソース定義

以降の説で各リソースの定義を記述する.
なお,各プロパティは PROMCODE 名前空間の場合には接頭語を省略して表記するものとする.

3.3.2 OSLC Core プロパティ

PROMCODE の下記のリソースにおいて共通に使用する OSLC Core のプロパティを示す.

- ScopeItem
- WorkItem
- Artifact
- Issue

名前	多重度	読み取り専用	型	範囲	説明
OSLC Core: 共通プロパティ					
oslc:shortTitle	zero-or-one	unspecified	XMLLiteral	n/a	リソースを表す短い名前.ユーザーに表示するための省略 ID が使われることが多い.

					XHTML のタグの内部に使用できる文字列のみを利用すべきである(SHOULD).
dcterms:description	zero-or-one	unspecified	XMLLiteral	n/a	リソースの詳細を記述する XHTML テキスト(Dublin Core[5]を参照). XHTML の タグの内部に使用できる文字列のみを利用すべきである(SHOULD).
dcterms:title	exactly-one	unspecified	XMLLiteral	n/a	リソースのサマリーを記述する XHTML テキスト(Dublin Core[5]を参照).通常1行のテキストで表される. XHTML の タグの内部に使用できる文字列のみを利用すべきである(SHOULD).
dcterms:identifier	exactly-one	TRUE	String	n/a	リソースのユニークな識別子. サービスプロバイダによって自動的に割り振られる. ユーザーに表示されることは意図していない.
dcterms:subject	zero-or-many	FALSE	String	n/a	リソースに付加するタグあるいはキーワード.
Dcterms:creator	zero-or-many	unspecified	Either Resource or Local Resource	any	リソースの作成者(Dublin Core[5]を参照). 通常 FOAF[6]で定義されている Person が使用されるが限定はされない.
Dcterms:contributor	zero-or-many	unspecified	Either Resource or Local Resource	any	リソースに対する責任者 (Dublin Core[5] を参照). 通常 FOAF[5]で定義されている Person が使用されるが限定はされない.
Dcterms:created	zero-or-one	TRUE	DateTime	n/a	リソースの作成日時 (Dublin Core[5] を参照).
Dcterms:modified	zero-or-one	TRUE	DateTime	n/a	リソースの最終更新日時(Dublin Core[5] を参照).
rdf:type	zero-or-many	unspecified	Resource	n/a	リソースの型の URI . 例: http://promcode.org/ns/pm/ScopeItem
oslc:serviceProvider	zero-or-many	unspecified	Resource	oslc:ServiceProvider	リソースのスコープを表すサービスプロバイダの URI .

oslc:instanceShape	zero-or-one	Unspecified	Resource	oslc:ResourceShape	リソースの持つプロパティの型や設定可能な値に関する情報を提供する Resource Shape.

3.3.3 PROMCODE ManagedItem プロパティ

PROMCODE の下記のリソースにおいて共通に使用する PROMCODE ManagedItem のプロパティを示す.

- ScopeItem
- WorkItem
- Artifact
- Issue

名前	多重度	読み取り専用	型	範囲	説明
Managed Item: 共通プロパティ					
dcterms:type	zero-or-many	FALSE	String		具体的な型を表す名前. 例: ● 設計(WorkItem) ● テスト文書(Artifact)
名前	多重度	読み取り専用	型	範囲	説明
関連のためのプロパティ					
supersedes	zero-or-one	FALSE	Resource	Change	
supersededBy	zero-or-one	FALSE	Resource	Change	
relatedBy	zero-or-many	FALSE	Resource	Issue	

3.3.4 リソース ScopeItem

- 名前：ScopeItem
- URI：http://promcode.org/ns/pm#ScopeItem

ScopeItem プロパティ

ScopeItem は OSLC Core の共通プロパティおよび PROMCODE ManagedItem プロパティに追加して以下のプロパティを持つ.

名前	多重度	読み取り専用	型	範囲	説明
:ScopeItem 特有のプロパティ					
plannedSize	zero-or-one	FALSE	Decimal		
actualSize	zero-or-one	FALSE	Decimal		
名前	多重度	読み取り専用	型	範囲	説明
ScopeItem における関連のためのプロパティ					
consistsOf	zero-or-many	FALSE	Resource	ScopeItem	階層化された ScopeItem の子 ScopeItem への参照
composedBy	zero-or-one	FALSE	Resource	ScopeItem	階層化された ScopeItem

					の親 ScopeItem への参照
requires	zero-or-many	FALSE	Resource	WorkItem	
produces	zero-or-many	FALSE	Resource	Artifact	

3.3.5 リソース WorkItem

- 名前：WorkItem
- URI：http://promcode.org/ns/pm#WorkItem

WorkItem プロパティ

WorkItem は OSLC Core の共通プロパティおよび PROMCODE ManagedItem プロパティに追加して以下のプロパティを持つ.

名前	多重度	読み取り専用	型	範囲	説明
WorkItem 特有のプロパティ					
phase	zero-or-one	FALSE	String		
plannedStartDate	zero-or-one	FALSE	DateTime		
actualStartDate	zero-or-one	FALSE	DateTime		
pllanedEndDate	zero-or-one	FALSE	DateTime		
actualEndDate	zero-or-one	FALSE	DateTime		
名前	多重度	読み取り専用	型	範囲	説明
WorkItem における関連のためのプロパティ					
consistsOf	zero-or-many	FALSE	Resource	WorkItem	階層化された WorkItem の子 WorkItem への参照
composedBy	zero-or-one	FALSE	Resource	WorkItem	階層化された WorkItem の親 WorkItem への参照
requiredBy	zero-or-one	FALSE	Resource	ScopeItem	
produces	zero-or-many	FALSE	Resource	Artifact	
representedBy	zero-or-one	FALSE	Resource	Any	通常 FOAF[5]で定義されている Person が使用されるが限定はされない.

3.3.6 リソース Artifact

- 名前：Artifact
- URI：http://promcode.org/ns/pm#Artifact

Artifact プロパティ

Artifact は OSLC Core の共通プロパティおよび PROMCODE ManagedItem プロパティに追加して以下のプロパティを持つ.

名前	多重度	読み取	型	範囲	説明

		り専用			
Artifact における関連のためのプロパティ					
consistsOf	zero-or-many	FALSE	Resource	Artifact	階層化された Artifact の子 Artifact への参照
composedBy	zero-or-one	FALSE	Resource	Artifact	階層化された Artifact の親 Artifact への参照
producedBy	zero-or-many	FALSE	Resource	ScopeItem or WorkItem	
measuredBy	zero-or-many	FALSE	Resource	Measurement	

3.3.7 リソース Measurement

- 名前：Measurement
- URI：http://promcode.org/ns/pm#Measurement

Measurement プロパティ

名前	多重度	読み取り専用	型	範囲	説明
Measurement 特有のプロパティ					
date	zero-or-one	FALSE	DateTime		
名前	多重度	読み取り専用	型	範囲	説明
Measurement における関連のためのプロパティ					
measures	zero-or-one	FALSE	Resource	Artifact	
measure	zero-or-many	FALSE	Resource	Measure	

3.3.8 リソース Measure

- 名前：Measure
- URI：http://promcode.org/ns/pm#Measure

Measure プロパティ

名前	多重度	読み取り専用	型	範囲	説明
Measure 特有のプロパティ					
dcterms:type	zero-or-many	FALSE	String		具体的な型を表す名前.
plannedValue	zero-or-one	FALSE	Decimal		
actualValue	zero-or-one	FALSE	Decimal		
名前	多重度	読み取り専用	型	範囲	説明
Measure における関連のためのプロパティ					
measurement	zero-or-one	FALSE	Resource	Measurement	

3.3.9 リソース Issue

- 名前：Issue

- URI：http://promcode.org/ns/pm#Issue

Issue プロパティ

Issue は OSLC Core の共通プロパティおよび PROMCODE ManagedItem プロパティに追加して以下のプロパティを持つ.

名前	多重度	読み取り専用	型	範囲	説明
Issue における関連のためのプロパティ					
relates	zero-or-many	FALSE	Resource	ScopeItem, WorkItem, Artifact or Issue	

3.3.10 リソース Change

- 名前：Chnage
- URI：http://promcode.org/ns/pm#Change

Change プロパティ

名前	多重度	読み取り専用	型	範囲	説明
Change 特有のプロパティ					
created	exactly-one	TRUE	DateTime		
authorized	zero-or-one	FALSE	DateTime		
名前	多重度	読み取り専用	型	範囲	説明
Change における関連のためのプロパティ					
previous	zero-or-one	FALSE	Resource	ScopeItem, WorkItem or Artifact	
next	zero-or-one	FALSE	Resource	ScopeItem, WorkItem or Artifact	

3.4 サービスプロバイダ機能

3.4.1 サービスプロバイダリソース

PROMCODE サービスプロバイダは OSLC Core 仕様バージョン 2.0 で定義されているサービスプロバイダリソースを提供しなければならない.
PROMCODE サービスプロバイダは OSLC Core 仕様バージョン 2.0 で定義されているサービスプロバイダカタログリソースを提供してもよい.

3.4.2 クエリ機能

PROMCODE サービスプロバイダは OSLC Core 仕様バージョン 2.0 で定義されているクエリ機能を提供しなければならない.

3.4.3　ユーザインタフェース委譲

PROMCODE サービスプロバイダは OSLC Core 仕様バージョン 2.0 で定義されているユーザインタフェース委譲の提供をしてもよい.

3.5　実プロジェクトへの適用

実際のプロジェクトではインスタンスデータである「プロジェクトデータ」を用いてデータ交換を実現する.「プロジェクトデータ」は「プロジェクトモデル」のインスタンスを RDF リソースの形式で表したものである.
以下「プロジェクトモデル」を「プロジェクトデータ」にマッピングする方法について記述する.

3.5.1　サブクラスの定義

プロジェクトモデルでドメインモデルのサブクラスを定義して活用する場合、そのサブクラスをリソースにマップする際には、ManagedItem の type 属性を用いて表現することができる.
例えば 2.2 節のサンプルである ScopeItem のインスタンスとして定義される Function オブジェクトは、ScopeItem の type 属性に“Function”と記述することによって表すことができる.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:promis_pm="http://promis.jp/ns/pm/">
  <promis_pm:ScopeItem rdf:about="http://localhost:8080/oslc-excel/rest/services/projectA/data/SI1_1">
    <dcterms:type>FunctionItem</dcterms:type>
    <dcterms:title>予約システム</dcterms:title>
    <dcterms:identifier>SI1_1</dcterms:identifier>
  </promis_pm:ScopeItem>
</rdf:RDF>
```

3.5.2　拡張属性の定義

拡張属性は独自の名前空間を定義し、その名前空間においてプロパティを定義することで実現できる.
例えば、WorkItem に独自の属性として“担当部署”のような属性を定義したい場合には、以下のような定義を行い、WorkItem に付加して利用すればよい.

- 名前空間の定義： http://my.bbb.com/prj/
- プロパティ名： ownerGroup

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:promis_pm="http://promis.jp/ns/pm/"
    xmlns:myext="http://my.bbb.com/prj/">
  <promis_pm:WorkItem rdf:about="http://localhost:8080/oslc-excel/rest/services/projectA/data/WI3_1.1.2">
    <dcterms:title>予約UI設計</dcterms:title>
    <dcterms:identifier>WI3_1.1.2</dcterms:identifier>
    <myext:ownerGroup>グループ1</myext:ownerGroup>
    ...
  </promis_pm:WorkItem>
</rdf:RDF>
```

参考文献

[1] Open Services for Lifecycle Collaboration,
http://open-services.net/

[2] OSLC Core 仕様バージョン 2.0,
http://open-services.net/bin/view/Main/OslcCoreSpecification

[3] RFC-2119, Key words for use in RFCs to Indicate Requirement Levels,
http://www.ietf.org/rfc/rfc2119.txt

[4] Resource Description Framework (RDF)
http://www.w3.org/RDF/

[5] Dublin Core Metadata Element Set, Version 1.1,
http://dublincore.org/documents/dces/

[6] Friend of a Friend (FOAF) 0.98,
http://xmlns.com/foaf/spec/