# SOA に基づくソフトウェア開発管理サービスのモデルと実行アーキテクチャの提案と評価

青山 幹雄[†]　　長澤 伸治[††*]

ソフトウェア開発はオフショア開発やアウトソーシングによるグローバル開発が進んでいる．複数の組織がネットワークを介して協調する必要があることから，開発の構造が複雑となり，開発の統一的な実行と管理が困難である．本稿ではソフトウェア開発を SOA (Service-Oriented Architecture)の概念に基づきサービスとしてモデル化する．このモデルに基づき，ソフトウェア開発を管理する開発管理サービスモデルを提案し，その実行のために開発とは独立したメタインタフェースの実装を提案する．提案モデルを SOA 基盤上で実行可能な開発環境のアーキテクチャを提案する．環境のプロトタイプを開発し，PMBOK に基づく管理モデルでその有効性を評価し，提案方法の妥当性を示す．

# A Model and Enacting Architecture of Software Development Management Services Based on Service-Oriented Architecture

Mikio Aoyama[†]　Shinji Nagasawa[††*]

With increasing the size of software, software development becomes globally distributed. Because two or more organizations cooperate over the network, the structure of the software development organization is complex. Therefore, unified execution and the management of the software development project are difficult. In this research, the authors regard the software development as a composition of services. It proposes the development management service model based on the software development services inspired by the service-orientation. It models from two aspects of the software development and software development management. Next, the authors propose the method of designing executable development services based on the SOA (Service-Oriented Architecture). The execution environment of the service model was developed as a prototype. Whether service can be managed with the management model based on PMBOK on the prototype is evaluated, and the proposed method is validated.

## 1. はじめに

グローバルソフトウェア開発(GSD: Global Software Development)では複数の組織がネットワークを介して協業する必要があることから，その開発形態が複雑になる．

一方，SOA(Service-Oriented Architecture)[6]やクラウドコンピューティングの普及により，ソフトウェアやビジネスが Web を介してサービスとして提供されている．今後，GSD などのソフトウェア開発環境も SOA やクラウドコンピューティングへ移行すると考えられる [1]．

本稿では SOA 基盤上で人を含む開発活動をサービスと捉え，ソフトウェア開発サービス(SDS: Software Development Service)と呼ぶ．SDS をサービスとして統一的に管理するために，SDS のインタフェースに管理のためのメタインタフェース拡張を提案する．さらに，SDS をサービスとしてモデル化し，ソフトウェア開発管理をサービスとして実現するためのサービスメタモデルを定義する．サービスメタモデルに基づき，SDS を組み合わせて，ネットワークを介した GSD を実行可能とする．この 2 つのアプローチを統合し，SOA 基盤上で GSD の統一的な実行と管理を可能とする．

## 2. SOA に基づく GSD の管理を実現する課題

GSD がネットワークを介して協調するための課題を以下に挙げる．

(1)　ソフトウェア開発と管理のための統一的なサービスインタフェースモデルが未確立

従来提案されている SDS のモデルはインタフェースを介した成果物の交換に限定されている．しかし，ソフトウェア開発の管理では，実行中に進捗などのサービスの状態も知る必要があるが，それを取得するためのサービスインタフェースは提供されていない．

(2)　ソフトウェア開発と開発管理をサービスとして実行する環境のアーキテクチャが未確立

人手の作業を含む SDS の実行は，コンピュータにより実行される Web サービスとは異なる．人手による作業を含むプロセスを定義する WS-HumanTask(Web Services Human Task)[8]が提案されているが，ソフトウェア開発への適用は報告されていないため，その適用可能性が不明である．さらに，WS-HumanTask を適用してソフトウェア開発・管理をサービスとして実行する環境のアーキテクチャは提示されていない．

## 3. 関連研究

(1)　サービス指向に基づくソフトウェア開発

SOA に基づくソフトウェア開発モデルが提案されている [1][11]．ソフトウェア開発の各工程をサービスと捉え，その連携により開発を実現するが，具体的な実現方法は示されていない．

[†]南山大学 情報理工学部 ソフトウェア工学科
Department of Software Engineering, Nanzan University
[††]南山大学 大学院 数理情報研究科 (*現在，富士通株式会社勤務)
Graduate School of Mathematical Sciences and Information Engineering, Nanzan University

(2)　ソフトウェア開発プロセスの実行可能モデルとその環境の提案

ソフトウェア開発プロセスの実行可能モデルが提案されている [2]. モデル化した開発プロセスを BPEL4People[7]に変換する. しかし, 開発サービスの構造と管理方法が示されていない.

(3)　WS-HumanTask (Web Services Human Task)と BPEL4People

WS-HumanTask は SOA 基盤上で人手の作業を記述する仕様である [8]. BPEL4People は人手の作業を含むビジネスプロセスを記述する [7]. しかし, これらの仕様の応用は, その特殊性から様々な課題がある [15]. 特に, ソフトウェア開発への応用は報告されていない.

## 4.　アプローチ

SDS を管理するために PDCA サイクルに着目し, ソフトウェアの開発と管理の共通モデルとして導入する. このモデルに基づき, 開発と管理に必要なインタフェースを定義する. 次に WS-HumanTask の仕様から開発サービスに必要な構成要素を示す. さらに, 管理のインタフェースが持つべき構造を示す.

### 4.1　ソフトウェアの開発と管理の共通振舞いモデル化による管理モデル

ソフトウェア開発を開発と管理の対として定義する. 1 つの SDS のインスタンスに対して 1 つの管理のインスタンスが対応する. ここで, 管理は PMBOK[14]のプロジェクト管理プロセスに従うものとする. しかし, PMBOK では管理対象毎にプロセスが定義されるので, 管理対象によらない共通のモデルが必要である. そこで, PDCA を開発と管理の共通振舞いモデルとし, 管理を開発との相互作用として定義した(図 1).

開発と管理の PDCA サイクルを実行するためには, 開発プロセスから開発作業の進捗を確認する必要がある. 進捗報告を PMBOK の知識領域の各コントロールや監視の実行に基づき評価($D_M$)し, 開発の問題を判断することで開発作業を改善できる.

開発モデルと管理モデル間で情報を交換するためにインタフェースを拡張し, 開発管理メタインタフェース(DMMI: Development Management Meta-Interface)と呼ぶ. これは, コンポーネントのメタインタフェース [10]を Web サービス, WS-HumanTask に拡張したものとなる. 従って, DMMI は全ての開発プロセスに配備される開発管理のための標準メタインタフェースとなる.

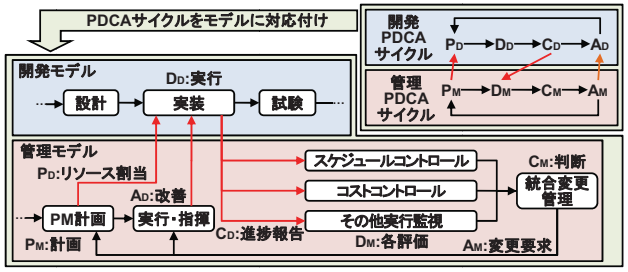


図 1 ソフトウェア開発の PDCA とモデルの対応

### 4.2　管理メタインタフェースの構造

開発管理は開発の進捗情報に基づき, スケジュールやコストなどの管理対象に応じた評価を行う. プロジェクト毎に同一管理対象であってもメトリクスが異なる場合があるので, 進捗報告に求められる情報は様々である. このため, 同一管理対象に対し異なるメトリクスのバインディングが可能な DMMI の構造を提案する. Web サービスインタフェース定義言語 WSDL の構造に着目し, 1 つの操作に対して複数のメッセージを交換可能なインタフェースとして定義する.

### 4.3　SDS のモデル化

SDS をモデル化する. SDS が人手の活動を含むことから, WS-HumanTask のアーキテクチャに基づき, サービス機能とサービスインタフェースの構造定義する(図 2).

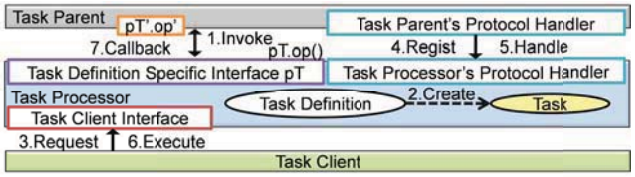


図 2 WS-HumanTask のアーキテクチャ

## 5.　ソフトウェア開発管理のサービスメタモデル

### 5.1　ソフトウェア開発管理サービスの設計

SOA 基盤上で実行可能な SDS の実行と管理の基礎となるサービスメタモデルを定義する. サービスメタモデルの機能とインタフェースは, WS-HumanTask のアーキテクチャに基づき決定する. WS-HumanTask の仕様では, ヒューマンタスクを Task Definition で定義する. これを開発管理サービスの機能に対応付ける. 実行中のヒューマンタスクを制御するためのインタフェースを開発管理サービスのインタフェースに対応付ける.

### 5.2　サービスの定義

ソフトウェア開発におけるコンピュータの処理や人を含む開発活動(ヒューマンタスク)を統一して SDS として定義する. SDS はプラットフォームと独立に定義し, 統一したインタフェースを提供する.

### 5.3　ソフトウェア開発管理のサービスメタモデル

WS-HumanTask に基づくソフトウェア開発管理のサービスメタモデルを図 3 に示す.

#### 5.3.1 サービス機能

サービス機能はヒューマンタスクであり, タスクの実行を可能とするための構造を持つ. 図 4 に示すように, SDS のサービス機能はタスク定義とマネージドタスクから構成する.

(1)　タスク定義

タスク定義は WS-HumanTask の Task Definition の仕様に基づき, task 要素に任意要素を含む interface などのヒューマンタスクを実行するために必要な 11 の要素を定義する. interface 要素にヒューマンタスクの入出力を定義する機能インタフ

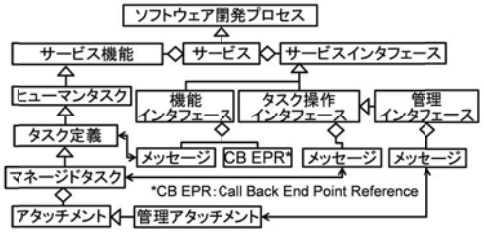


図 3 ソフトウェア開発管理サービスメタモデル

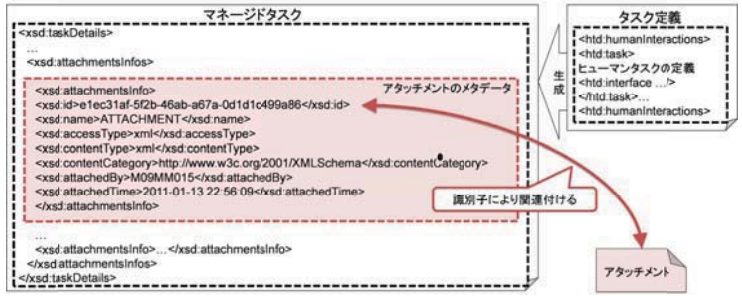


図 4 タスク定義，マネージドタスク，アタッチメント

ェースの記述をインポートすることで，タスク定義と機能インタフェースを対応付ける．

(2) マネージドタスク

サービス呼出し時にメッセージとタスク定義に基づき生成される．マネージドタスクはヒューマンタスクの状態や中間成果物を保持し，実行中のヒューマンタスクはマネージドタスクに基づき管理される．マネージドタスクは WS-HumanTask の Task 仕様に基づき，33 の要素を定義した．マネージドタスクはアタッチメントを持つ場合，マネージドタスク内にアタッチメントのメタデータを付加する．アタッチメントはメタデータ要素の id により，関連付ける．

#### 5.3.2 サービスインタフェース

サービスインタフェースは機能，タスク操作，管理の 3 つのインタフェースから構成される．

(1) 機能インタフェース

機能インタフェースはタスク定義で記述したタスクの入出力インタフェースとして WSDL で定義する．機能インタフェースを介してヒューマンタスクは生成される．ヒューマンタスクを生成するために必要なタスクのデッドライン，優先度，実行するために必要なプロダクトを要求として呼出す．ヒューマンタスクの成果物はコールバック，または，ポーリングによって取得される．

(2) タスク操作インタフェース

タスク操作インタフェースは生成されたヒューマンタスクを制御するための全タスク共通のインタフェースである．このインタフェースを介して個々のタスクが持つマネージドタスクの情報を取得できる．WS-HumanTask の定義するインタフェースを WSDL で記述したものになる．

(3) 管理インタフェース

管理インタフェースは実行中のタスクに対して開発リソースの割当や，進捗情報の取得を行うための全タスク共通のインタフェースである．管理インタフェースは PMBOK[14]に基づく 8 つの知識領域で定義される管理対象に対して，異なるメトリクスが利用できるように複数のポートタイプを定義する．また，管理情報はアタッチメントとして保持するため，情報の要素や型によらず保持が可能になる．

### 5.4 実行可能な開発サービスの配置方法

以下の 3 つのプロセスにより，実行可能な開発サービスを配置する(図 5)．

(1) 機能インタフェースを記述

プロダクトの入出力やオペレーションの契約などを記述する．

(2) タスク定義を記述

ヒューマンタスクの名前や内容，対応するインタフェースなどを記述する．

(3) 機能インタフェース記述とタスク定義を実行環境に配置

2 つの記述をヒューマンタスクの実行環境に配置する．配置したインタフェース記述を基に開発サービスを呼出す．

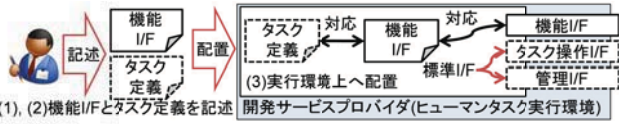


図 5 サービスプロバイダへの開発サービスの配置方法

## 6. SDS の実行と管理

提案モデルに基づく SDS の実行と管理を行うための方法を提案する．

### 6.1 開発管理の PDCA サイクル

SDS を組み合わせてソフトウェア開発を実現する．開発サービスプロバイダはソフトウェア開発管理サービスメタモデルに基づきサービスを提供する．管理モデルに基づくプロセスの組み合わせを開発管理サービス(DMS: Development Management Service)と定義し，開発管理サービスプロバイダ上で実行する．開発管理サービスプロバイダはプロセス実行エンジンとプロセス監視モニタを持ち，管理サービスを起動し， PDCA サイクルを実行する(図 6)．

### 6.2 ヒューマンタスクの状態遷移モデル

WS-HumanTask に基づくタスクの振舞いは規定の状態遷移に従う．タスクの状態を遷移させるためにはタスク操作インタフェースを介した操作が必要になる(図 7)．

タスクの実行担当者を指定する場合，タスクを呼出してから正常終了するまでに以下の手順でタスクを操作するものとする．なお，この手順は例外処理を含まないものとする．タスクに作用するアクタを開発依頼者であるプロジェクトマネージャ(PM)と開発担当者と定義する．

(1) 開発サービスの呼出しとタスクの生成

PM が開発サービスの機能インタフェースに記述される要求のオペレーションを呼出すことでタスクが生成される．生成されたタスクは Created 状態になる．

(2) 開発担当者の指定

PM がタスク操作インタフェースに記述される nominate オペレーションを呼出すことで，担当者を決定する．Created 状態のタスクは Reserved 状態になる．

(3) タスクの開始

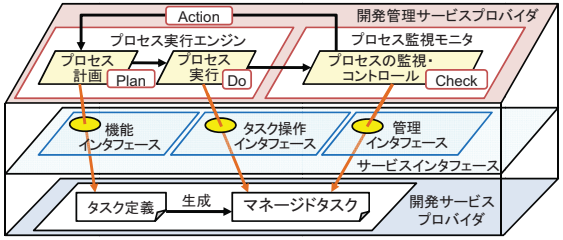


図 6 ソフトウェア開発の PDCA サイクル

PM 又は開発担当者が操作インタフェースに記述される start オペレーションを呼出すことで，タスクを開始可能状態にする．Reserved 状態のタスクは InProgress 状態になる．

(4) アウトプットの指定

開発担当者が操作インタフェースに記述される setOutput オペレーションを呼出すことで，タスクの応答時に受け渡すアウトプットを指定する．

(5) タスクの完了

PM 又は開発担当者が操作インタフェースに記述される complete オペレーションを呼出すことで，タスクを完了状態にする．InProgress 状態のタスクは Completed 状態になる．

(6) 開発サービスのアウトプットを取得

アウトプットを取得する方法はポーリングとコールバックの 2 つの方法がある．

1) ポーリングによるアウトプットの取得：PM が開発サービスの機能インタフェースに記述される応答のオペレーションをタスクの識別子を用いて呼出すことで，サービスのアウトプットを取得する．

2) コールバックによるアウトプットの取得：開発担当者が機能インタフェースに記述される応答オペレーションをタスクの識別子を用いて呼出すことで，アウトプットを PM に送信する．また，start や complete オペレーションは開発サービスプロバイダが開発依頼者に対して呼出しの権限を与えるか否かを決定する．

### 6.3 PDCA サイクルの振舞い

PDCA サイクルに対応した実行環境の振舞いを図 8 に示す．PMBOK のプロセスを次のように PDCA サイクルへ対応付けた．

(1) 計画(Plan)フェーズ：WBS 作成とプロジェクトマネジメント計画書作成

(2) 実行(Do)フェーズ：プロジェクト実行の統制

(3) 評価(Check)フェーズ：スケジュールやコストのコントロール，品質管理，変更管理など

(4) 改善(Action)フェーズ

1) 評価の結果，開発のプロセスに対してリソースの再割当が必要な場合：プロジェクト実行のリソースの再見積りと再割り当て

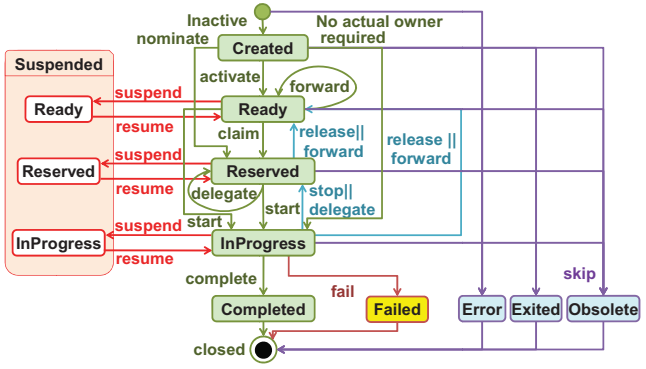


図 7 タスクの状態遷移

2) 評価の結果，開発のプロセス全体の変更が必要な場合：プロジェクトマネジメント計画書の見直し

### 6.4 計画実行フェーズ

開発管理サービスプロバイダのプロセス実行エンジン上のプロセスである．

(1) 計画フェーズ：プロセス計画では，以下 3 つの作業を行う．

1) 開発サービスの実行順序を記述：開発サービスの組み合わせをプロセスとして記述．

2) 開発リソース割当ての付加：各開発サービスに対して開発リソースを割当てるための記述を行う．プロジェクト管理情報を扱うポートタイプの指定，開発サービスの作業担当者の指定，開発サービスの BAC(Budget at Completion)指定の 3 つをプロセスに付加する．

3) 計画プロセスの実行：リソースの割当てを付加した計画プロセスを実行し，ヒューマンタスクを生成する．

(2) 実行フェーズ

プロセス実行では SDS を実行するため，次の 2 つの作業を行う．

1) 計画プロセスを実行プロセスへ変換：計画プロセスの記述に従い，各開発サービスの実行プロセスを行うためのアタッチメントの入力，開発作業の開始，アウトプットの取得の 3 つのアクティビティに変換する．アタッチメントの入力と開発作業の開始はタスク操作インタフェース，アウトプットの取得は機能インタフェースを介してメッセージ交換を行う．

2) 実行プロセスの起動：変換した実行プロセスをプロセス実行エンジンに配置し，実行プロセスを起動する．

### 6.5 評価改善フェーズ

開発管理サービスプロバイダのプロセス監視モニタ上のプロセスを下記に示す．

(1) 評価フェーズ

管理インタフェースを介してヒューマンタスクの実行を管理する．取得した実績情報から EV (Earned Value)などを用いて，プロジェクトの進捗やコストを評価する．

(2) 改善フェーズ

実績に基づき開発プロセスをコントロールする．

1) 各サービスのリソース割当の変更：進捗が遅れた場合，管理インタフェースを用いてサービスに割当てられたリソースを変更し，立て直す．

2) 開発プロセスの再設計：計画プロセスを再設計する．呼出す開発サービスや実行順序を変更する．

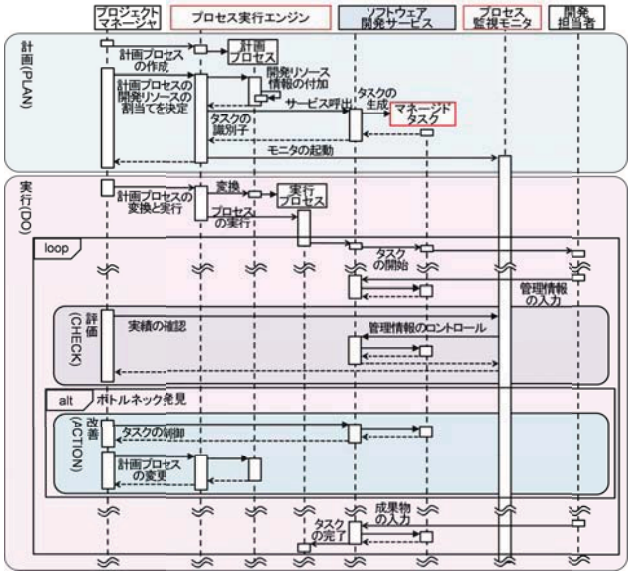


図 8 ソフトウェア開発管理の実行とその振舞い

## 7. プロトタイプ開発

開発と開発管理のサービスプロバイダのプロトタイプを開発した.

### 7.1 プロトタイプの構成

(1) プロトタイプの構成

プロトタイプの構成を図 9 に示す.

(2) 開発規模

サービス, プロセス記述, 監視モニタの開発規模を表 1 に示す.

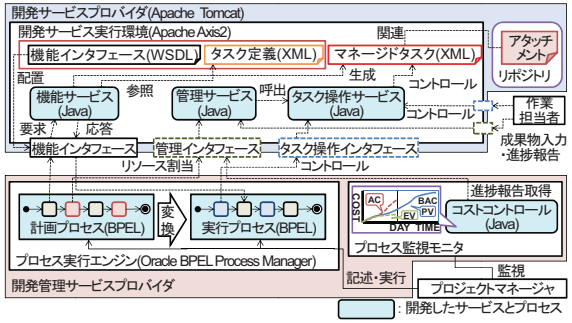


図 9 プロトタイプの構成

表 1 開発規模

| 配置環境 | | 開発した種類 | | 開発言語 | 開発規模(LOC) |
|---|---|---|---|---|---|
| 開発サービスプロバイダ | | 機能サービス, タスク操作, 管理サービス | | Java | 3125 |
| 開発管理サービスプロバイダ | プロセス実行エンジン | 記述プロセス | 計画プロセス | BPEL | 658 |
| | | | 実行プロセス | BPEL | 512 |
| | プロセス監視モニタ | コストコントロール(プロセス監視モニタ) | | Java | 212 |
| 合計 LOC | | | | | 4295 |

### 7.2 開発サービス実行環境(開発サービスプロバイダ)の開発

開発サービスプロバイダの実行環境として, アプリケーションサーバの Apache Tomcat を使用し, Web サービスフレームワークとして Apache Axis2[4]を使用した.

#### 7.2.1 機能サービスの実装

機能サービスは要求と応答の 2 つのオペレーションを持つ. 要求を行うことで, 要求メッセージとタスク定義に基づきマネージドタスクを生成する. 応答としてサービスのアウトプットを取得する. 応答メッセージの取得方法はコールバックとポーリングの 2 通りで実現した. ポーリングは要求時に決定したタスクの識別子を用いる. コールバックは WS-Addressing[5]を用いて実現した. 本稿では機能サービスにポーリングを用いた応答を実装した.

#### 7.2.2 タスク操作サービスの実装

WS-HumanTask のインタフェースで定義されるオペレーションの集合をタスク操作サービスとして実装した. 実装したオペレーション数は 34 個である. 9 個がプロセスを実行するための必須オペレーションとして使用し, 25 個はオプションである.

#### 7.2.3 管理サービスの実装

管理サービスのオペレーションはオペレーション毎に複数のメッセージが対応するため, オペレーションは異なる構造のメッセージを処理できる必要がある. このため, オペレーションの引数と返り値の定義に Axiom(AXIs Object Model)[3]を採用した. Axiom によって引数と返り値は構造や型を制限しない XML としてメッセージ交換が可能なため, 各オペレーションに対して複数の構造をとるメッセージを受け付けることが可能となった.

### 7.3 開発管理サービス実行環境(開発管理サービスプロバイダ)の開発

(1) プロセス実行エンジン

開発管理サービスプロバイダのプロセス実行エンジンの実行環境として, Oracle BPEL Process Manager[12]を使用した. プロセス実行エンジン上で実行するプロセスの作成は Oracle JDeveloper[13]を使用し, 計画プロセスと実行プロセスを記述した.

(2) プロセス監視モニタ

プロセス監視モニタとしてコストコントロールを実装した. コストコントロールは開発作業の進捗情報から, EV(Earned Value)を用いてプロジェクトの実績を評価する. 管理インタフェースからコスト情報を取得し, JFreeChart[9]を用いて EV を視覚的に表示する.

## 8. 例題による開発と管理の試行

(1) 例題のシナリオ

開発プロセスに基づき実装と試験サービスを連携し, ソフトウェア開発サービスを実行する. 実行中の開発サービスから進捗情報を取得し, EV を評価し, 計画を変更する.

(2) 開発サービスの配置

実装サービスと試験サービスの各サービスのインタフェースとタスク定義を記述し, 開発サービスプロバイダに配置した(図 10).

(3) 開発サービス連携の記述と実行

実装と試験サービスの連携を BPEL で記述した. PDCA サイクルに基づき, 計画プロセスと実行プロセスを BPEL で記述し, BPEL サーバに配置し, 起動することで開発を実行した.

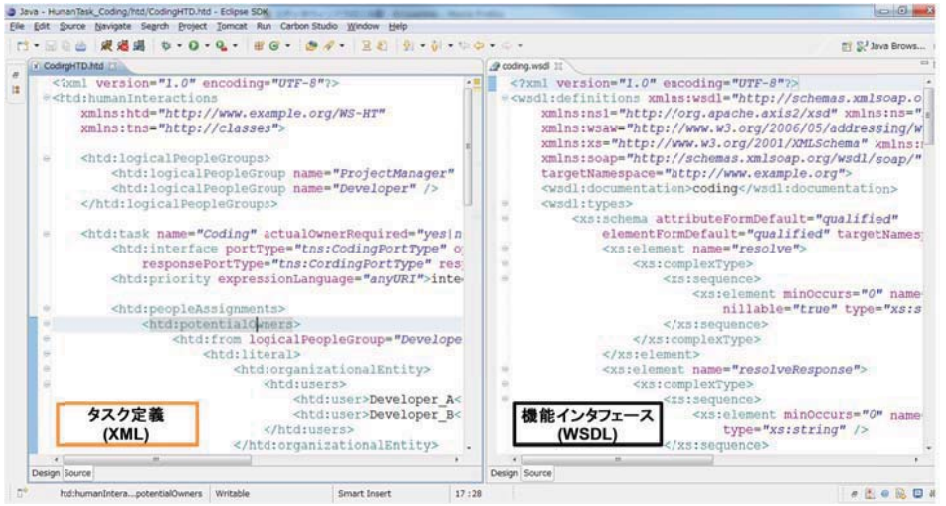


図 10 機能インタフェースとタスク定義の例

(4) 管理インタフェースを介した進捗情報の入力
実行中である実装サービスに対して，開発に費やしたコストを一定時間毎に入力した．
(5) EV の測定と計画の変更
プロトタイプ実装で開発したプロセス監視モニタのコストコントロールを用いて実装サービスのコストから EV を計算した．EV から，AC(Actual Cost)が PV(Planed Value)よりも超過していることが確認できた．EV に基づき，管理インタフェースの改善オペレーションを用いてプロジェクトの許容範囲内で AC が PV を超えないように BAC を増やした．再び EV の測定から AC が PV を超過していないことを確認し，開発が動的に変更できたことを確認した．

## 9. 評価

(1) WS-HumanTask に基づく開発・管理サービス設計の妥当性の評価
WS-HumanTask に基づき機能，タスク操作，管理インタフェースを設計した．WS-HumanTask 仕様のインタフェースを開発サービスに組み込むことによって，実行中の開発サービスに対して，中間成果物の取得やタスクの制御が可能になる．さらに管理インタフェースはプロジェクト要求から使用するポートタイプの選択により測定したい管理情報を取得可能になる．これは，特定の開発形態に依存しないので共通に適用可能である．Web サービスが Web 上の標準インタフェースを提供すると同様に，プロジェクト管理の標準インタフェースを提供できる点で意義がある．また，HS-HumanTask により人手のタスクが定義できるが，ソフトウェア開発への適用はない．本稿では，ソフトウェア開発における進捗管理などの固有の要求を満たす適用方法を提案し，その有効性を示した．
(2) メタインタフェースの妥当性の評価
タスク操作と管理のインタフェースをメタインタフェースとして提案した．メタインタフェースは開発サービスの実行とは独立で，全ての開発サービスに対して共通のオペレーションを持つ．メタインタフェースから内部情報の取得やコントロールが可能である．このことから，メタインタフェースによって，開発の進捗などを自動収集するツールが実現可能となる．
(3) プロトタイプによる提案モデルの妥当性の評価
1) BPEL を用いた開発サービスの連携による SDS の実現可能性の確認：例題として実現した実装サービスと試験サービスは BPEL 実行エンジンを用いて連携することで SDS を実現可能とした．また開発プロセスを再設計する時，実行中のサービスを組み替えることによって再設計したプロセス上で継続可能である．これにより実行中のサービスを異なる開発サービスから呼出し可能となることを確認した．
2) 管理インタフェースを用いた EV 測定とリソースの再割当てによるモデルの妥当性確認：プロトタイプでは実行中タスクの進捗メトリクスに EV を用いた．EV に基づき開発のボトルネックを発見し，リソース再割当てや計画の変更を行った．これより進捗やコストに応じて開発リソースの割当てが動的に変更可能になることを確認した．

## 10. 今後の課題

(1) 管理サービスの連携を可能とする
開発サービスの連携は可能となったが，メタインタフェースを用いる管理サービスの連携は未確立である．PMBOK で定義される管理アクティビティをサービスとして連携を可能とする．
(2) 開発サービス担当者の動的な決定と割当て可能とする
開発サービスの担当者を動的に決定し，割り当てる仕組みが必要になる．

## 11. まとめ

グローバルソフトウェア開発をクラウドや SOA 基盤上で統一的に実行，管理する方法とその支援環境のアーキテクチャを提案した．このため，WS-HumanTask 仕様に基づき開発管理可能なサービスのメタモデルを提案し，開発とは独立に管理を可能とするインタフェースをメタインタフェースとして実装する方法を提案した．さらに，提案モデルに基づき設計した開発サービス間の連携と各サービスを PDCA サイクルで管理する環境のアーキテクチャを提案した．提案モデルを実行する開発サービスプロバイダと管理サービスプロバイダのプロトタイプを実装し，サービス連携と EV に基づく開発管理を試行し，提案モデルの妥当性を確認した．

## 参考文献

1) 青山 幹雄，サービス開発のサービス化を実現する統一サービスシステム USS (Unified Service Systems)の提案，ウィンターワークショップ 2009・イン・宮崎 論文集，Jan. 2009, pp. 53-54.
2) 浅岡 奈津貴，ほか，ソフトウェア開発プロセスのサービスモデルとその実行環境の提案と評価，第 167 回ソフトウェア工学研究会，Mar. 2010, pp. 1-8.
3) Axiom, http://ws.apache.org/axiom/.
4) Axis2 Architecture Guide, http://axis.apache.org/axis2/java/core/docs/Axis2ArchitectureGuide.html.
5) D. Box, et al. (Eds.), Web Services Addressing (WS-Addressing), W3C, Aug. 2004.
6) T. Erl, Service-Oriented Architecture, Prentice Hall, 2005.
7) D. Ings, et al., WS-BPEL Extension for People (BPEL4People) Specification, Ver. 1.1, May 2010.
8) D. Ings, et al., Web Services Human Task (WS-Human Task) Specification, Ver. 1.1, May 2010.
9) JFreeChart, http://www.jfree.org/jfreechart/.
10) D. H. Lorenz and J. Vlissides, Pluggable Reflection: Decoupling Meta-Interface and Implementation, Proc. ICSE 03, IEEE Computer Society, May 2003, pp. 3-13.
11) 大原 晋吾，ほか，サービス指向に基づくソフトウェア開発モデル化方法論の提案，第 163 回ソフトウェア工学研究会，Mar. 2009, pp. 249-256.
12) Oracle BPEL Process Manager, http://www.oracle.com/technetwork/middleware/bpel/.
13) Oracle, JDeveloper, http://www.oracle.com/technetwork/developer-tools/jdev/.
14) PMI, A Guide to the Project Management Body of Knowledge, 4th ed., PMI, 2008.
15) N. Russell, et al., Work Distribution and Resource Management in BPEL4People: Capabilities and Opportunities, Proc. CAiSE 2008, LNCS 5074, Springer, Jun. 2008, pp. 94-108.