Vol.2011-SE-171 No.19
2011/3/15

ソフトウェア開発管理サービスのモデルと実行環境の提案

長澤 伸治† 青山 幹雄††

ソフトウェア開発はソフトウェアの大規模化に伴いオフショア開発やアウトソーシングによるグローバルな分散化が進んでいる. 複数の組織がネットワークを介して協調することから, ソフトウェア開発の構造が複雑になっている. そのためソフトウェア開発プロジェクト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サービスの複合体と考える. サービス指向を適用したソフトウェア開発サービスを基礎とし, 開発管理サービスモデルを提案する. ソフトウェア開発とソフトウェア開発管理の 2 つの視点からモデル化する. 次に, 提案したモデルを SOA(Service-Oriented Architecture)の基盤技術に基づき実行可能なサービスの設計方法を提案する. サービスモデルの実行環境はプロトタイプとして開発した. プロトタイプによりサービスが PMBOK に基づく管理モデルで管理可能かを評価し, 提案方法の妥当性を示す.

A Model and It's Design Method of Software Development Management Services

Shinji Nagasawa† and Mikio Aoyama††

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 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では複数の組織がネットワークを介して協調することから, ソフトウェア開発の形態が複雑になっている. このような形態のソフトウェア開発のプロジェクト管理は, 多種多様となることがありえる. そのためソフトウェア開発と開発管理の統一的な実行と管理が困難である.

本稿ではサービス指向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と呼ぶ. ソフトウェア開発サービスに開発管理可能なメタインタフェースを拡張する. このサービスをサービス指向に基づきモデル化し, ソフトウェア開発管理のサービスメタモデルとして提案する. このサービスを組み合わせ, ネットワークを介した統一的な実行を確立する. 開発管理のメタインタフェースを用いてネットワークを介した統一的な開発管理を確立する. この 2 点から, サービス指向基盤上で統一的な開発と開発管理の実行を実現する.

2. グローバルソフトウェア開発の課題

グローバルソフトウェア開発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) ソフトウェア開発サービスの統一的なサービスモデルが未確立

ソフトウェア開発サービスではプロダクトの入出力しか検討されておらず, 実行中にサービスが持つ内部情報は取得することができない. また, サービスの提供方法がプロバイダごとに異なるため, 統一的なサービスのモデルが必要である.

(2) ソフトウェア開発サービスの開発管理のためのインタフェースが未定義

人手の作業であるソフトウェア開発サービスの処理能力は作業担当者に依存する. そのため, サービスに開発管理を可能とし, 進捗を取得するためのインタフェースが必要である. しかし, ソフトウェア開発サービスは自律して実行できなければならない. そのため, プロダクトの入出力とは独立した開発管理可能なインタフェースをメタインタフェースとして定義する必要がある.

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[1, 6]の概念を用いたソフトウェア開発の実現方法である, サービス指向に基づくソフトウェア開発(SOSD: Service-Oriented Software Development)のモデル化方法が提案されている[10]. SOA の 3 階層に基づいたソフトウェア開発モデルを基にソフトウェアの開発工程をサービスと捉え, サービス連携によりネットワークを介したソフトウェア開発の実現方法が述べら

† 南山大学大学院 数理情報研究科
Graduate School of Mathematical Sciences and Information Engineering, Nanzan University
†† 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

れている. しかし, 開発プロセスを実行するための具体的な方法は定義されていない.

(2) 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

ソフトウェア開発プロセスのモデル化が提案されている[2]. さらにモデル化した開発プロセスをサービス指向基盤上で実行可能な記述である BPEL4People[7]に変換する方法が提案されている. しかし, 開発サービスの具体的な構造と開発管理方法が定義されていない.

(3) Web Services Human Task (WS-HumanTask)[8]

WS-HumanTask はサービス指向基盤上での人手の作業をヒューマンタスクとしてサービス化し, 記述する仕様である. ヒューマンタスクを Web サービスとして提供するために必要な構成要素と制御するためのインタフェースが定義されている.

4. アプローチ

本稿では, プロジェクト管理可能なソフトウェア開発サービスをモデル化するために, ソフトウェア開発のモデルとソフトウェア開発管理モデルの PDCA サイクルとモデル間の相互作用に着目し, 開発サービスに必要な開発管理の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, 開発管理のインタフェースが持つべき構造を示す.

4.1 ソフトウェア開発とソフトウェア開発管理の相互作用

ソフトウェアの開発はソフトウェア開発(以下開発)とソフトウェア開発管理(以下開発管理)から成り立つ. 開発と開発管理はソフトウェア開発を行うための基礎となるモデルを持つ. ここで, 開発管理モデルを PMBOK[13]のプロジェクト管理プロセスの組み合わせと捉える. ソフトウェア開発の PDCA サイクルの開発モデルと開発管理モデルへの対応付けを図 1 に示す.

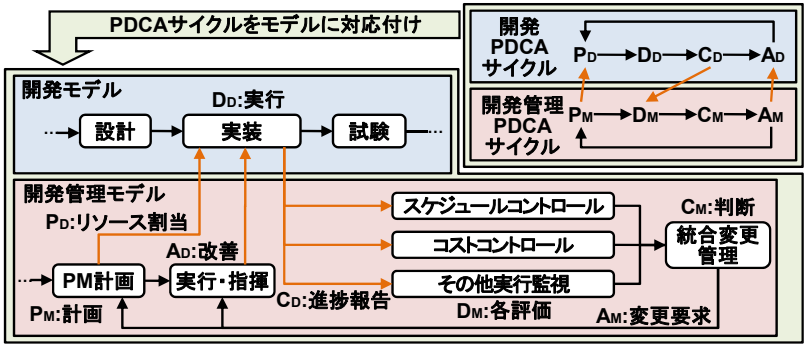

図 1 ソフトウェア開発の PDCA サイクルとモデルの対応付け

開発と開発管理の 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の改善を行う.

本稿では, 開発モデルと開発管理モデル間で情報の入出力を開発管理のメタインタフェースと定義する. 開発管理のメタインタフェース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インタフェースである.

4.2 開発管理のメタインタフェースの構造

開発管理は開発作業の進捗報告の情報に基づき, スケジュールやコストなどの知識領域に対応した評価を行う. さらにプロジェクトごとに同じ知識領域でも異なる評価指標を持ち, 進捗報告に求める情報は様々である. 本稿では, 同じ知識領域から様々な情報の入出力が可能な開発管理のメタインタフェースの構造を提案する. サービスのインタフェースを記述する WSDL の記述能力に着目し, 1 つの操作に対して様々なメッセージ交換が可能なインタフェースを記述する.

4.3 ソフトウェア開発サービスのモデル化

開発管理可能なサービスをモデル化する. SOA におけるサービスはサービス機能とサービスインタフェースから構成される. サービス機能はサービスを実行する機能であり, サービスインタフェースはサービスを呼び出すために必要な情報やメッセージ交換方法を定義する.

本稿では, サービスのモデル化は WS-HumanTask のアーキテクチャを基にサービス機能とサービスインタフェースの具体的な構造を定義する(図 2).

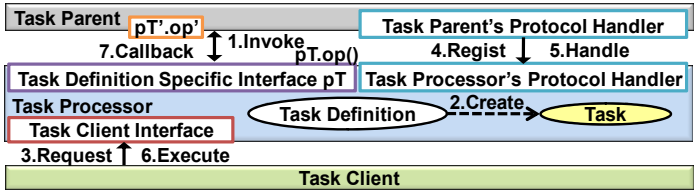

図 2 WS-HumanTask のアーキテクチャ

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

サービス指向基盤上で実行可能な開発サービスの基礎となるソフトウェア開発管理のサービスメタモデルを提案する. 提案サービスメタモデルのサービス機能とサービスインタフェースは, サービス指向基盤技術の WS-HumanTask のアーキテクチャに基づき構造を決定する. WS-HumanTask は仕様でヒューマンタスクを定義する記述の Task Definition が定義されており, これをサービス機能に対応付ける. 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が定義されており, これをサービスインタフェースに対応付ける. また, WS-HumanTask では 4.1 節の開発管理のインタフェースが未定義なため, WS-HumanTask が定義するインタフェースを拡張して開発管理のインタフェースを設計する.

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)である開発プロセスをサービスと定義する. サービスはプラットフォームとは独立に定義し, 統一的な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

開発と開発管理の 2 つの視点から 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.

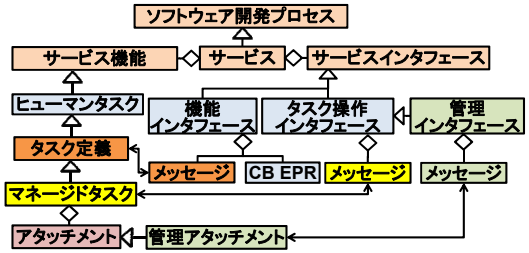

図 3 ソフトウェア開発管理のサービスメタモデル

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, タスクの実行を実現するための構造を持つ. 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される(図 4).

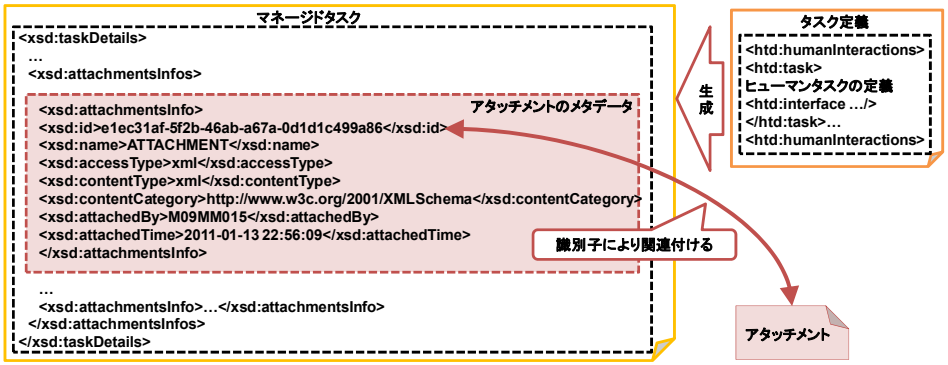

図 4 タスク定義, マネージドタスク, アタッチメント

(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

サービス呼び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 の仕様に基づき, 33 の要素を定義する. 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成果物の入出力インタフェースであり, WSDL でインタフェースを記述する. 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情報である, 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び出す. ヒューマンタスクの最終成果物はコールバック又はポーリングで取得する.

(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に対して制御を行う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で, 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 に基づく 8 つの知識領域に対して, PDCA サイクルのオペレーションを持つ複数のポートタイプを定義する(図 5). 使用するポートタイプによって同じ知識領域の管理情報であっても, 異なる指標で管理が可能になる. また, 管理情報はアタッチメントとして保持するため, 情報の要素や型にとらわれずに保持が可能になる.

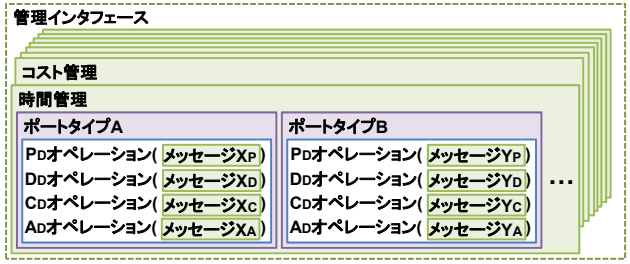

図 5 管理インタフェースの構造

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 6).

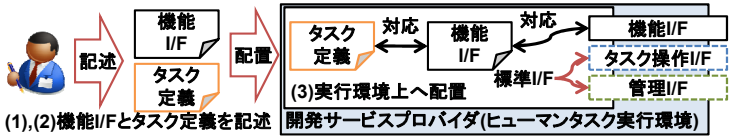

図 6 サービスプロバイダへの開発サービスの配置方法

(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できる環境に配置する. 配置したインタフェース記述を基に開発サービスを呼び出す.

6. ソフトウェア開発サービスの実行と管理

提案モデルに基づく開発サービス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

開発サービスを組み合わせることでソフトウェア開発は実現される. 開発サービスプロバイダはソフトウェア開発管理のサービスメタモデルに基づくサービスを提供する. 開発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と定義し，このサービスを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，開発の PDCA サイクルを実行する(図 7).

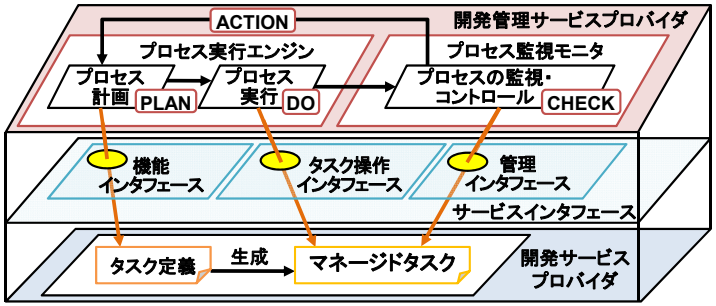

図 7 ソフトウェア開発の PDCA サイクル

6.2 ヒューマンタスクの状態と状態遷移

WS-HumanTask に基づくタスクは常に特定の状態を持ち，タスクの状態を変更す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 8).

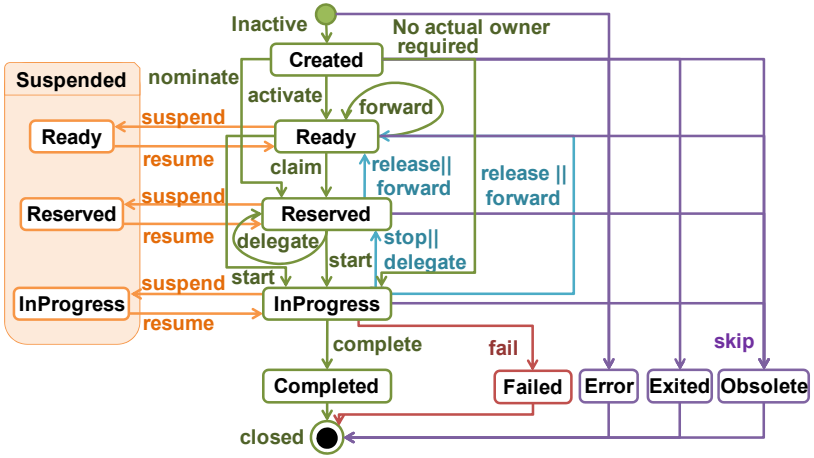

図 8 タスクの状態と状態遷移

タスクの作業担当者を指定する場合，タスクを呼び出してから正常に完了するまでに以下の手順でタスクを操作する必要がある. なお，この手順は最短の手順とする. タスクに作用するアクタを作業依頼者であるプロジェクトマネージャ(PM)と作業担当者と定義する.

(1) 開発サービスの呼び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び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) 作業担当者の指定
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び出すことで，作業担当者を決定する. Created 状態のタスクは Reserved 状態になる.

(3) タスクの開始
PM 又は作業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び出すことで，タスクを作業開始可能な状態にする. Reserved 状態のタスクは InProgress 状態になる.

(4) アウトプットの入力
作業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び出すことで，タスクの応答となるアウトプットを入力する.

(5) タスクの完了
PM 又は作業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び出すことで，タスクを完了状態にする. InProgress 状態のタスクは Complete 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び出すことで, 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

作業担当者が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び出すことで, 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作業依頼者に対して呼び出しの権限を与えるか否かを決定する.

6.3 全体の振舞い

PDCA サイクルに対応した実行環境の振舞いを定義し, 図 9 に振舞いを示す.

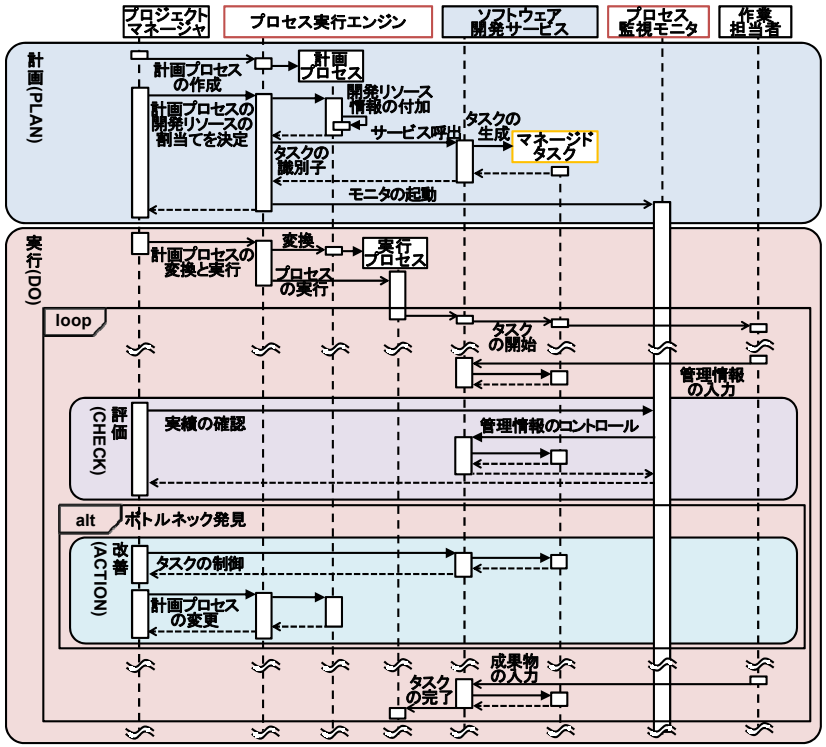

図 9 ソフトウェア開発管理の実行と実行環境の振舞い

PMBOK のプロセスを開発の PDCA サイクルへ以下の対応付けを行った.

(1)　計画(PLAN)のフェーズ

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)のフェーズ

プロジェクト実行の指揮・マネジメント

(3)　評価(CHECK)のフェーズ

スケジュールコントロール, コストコントロール, 品質管理, 統合変更管理など

(4)　改善のフェーズ

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指揮・マネジメント

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作成(再作成)

6.4 計画と実行のフェーズ

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を下記に示す.

(1)　プロセス計画(計画のフェーズ)

プロセス計画では SOSD の計画を実行するため, 以下 3 つの作業を行う(図 10).

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する.

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の割当てを行うための記述を行う. 本稿ではプロジェクトで扱う管理情報に対応するポートタイプの指定, 開発サービスの作業担当者の指定, 開発サービスの完成時総予算である BAC(Budget at Completion)の指定の 3 つのリソース割当てをプロセスに付加した.

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.

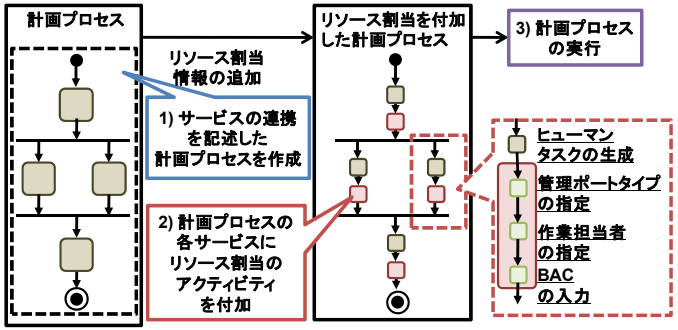

図 10 計画プロセスの記述と実行

(2)　プロセス実行(実行のフェーズ)

プロセス実行では SOSD の実行を行うため，以下 2 つの作業を行う(図 11).

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のプロセスを作業する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．本稿では実行プロセスへの変換方法については議論しない．

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．

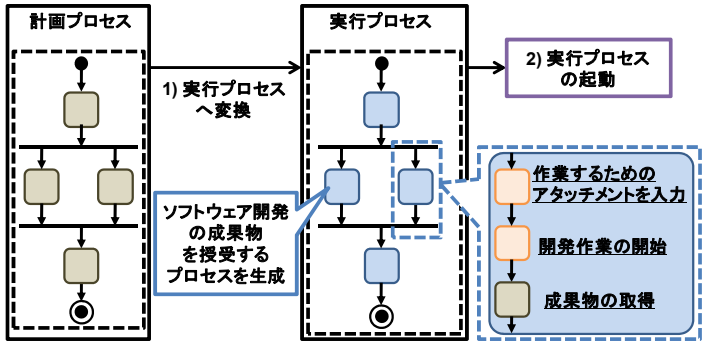

図 11 実行プロセスの起動

6.5 評価と改善のフェーズ

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)　プロセスの監視(評価のフェーズ)

使用する管理インタフェースを用いて，ヒューマンタスクの実行状況を監視する．取得した実績の情報を EVM(アーンドバリュー法)などの方法を用いて，プロジェクトの進捗やコストなどが計画通りに守られているかを測定する．

(2)　プロセスのコントロール(改善のフェーズ)

評価のフェーズで測定した実績に基づき，必要であれば開発のプロセスに対して特定のコントロールを行う．

1)　各サービスのリソース割当の変更

管理インタフェースを用いてサービスに割当てられたコストなどのリソース情報を変更する．これにより進捗遅れの立て直しを図る．

2)　開発プロセスの再設計

計画プロセスの再設計を行う．呼び出す開発サービスの変更や開発プロセスの順序変更などを行う．呼び出す必要の無くなった生成したヒューマンタスクは，タスク操作インタフェースを用いて凍結する．

7. 開発サービスプロバイダのプロトタイプ開発

6 章で提案した内容に基づき開発サービスプロバイダと開発管理サービスプロバイダのプロトタイプ開発を行った．

7.1 プロトタイプの構成

7.1.1 プロトタイプ全体の構成

開発したプロトタイプ全体の構成を図 12 に示す．

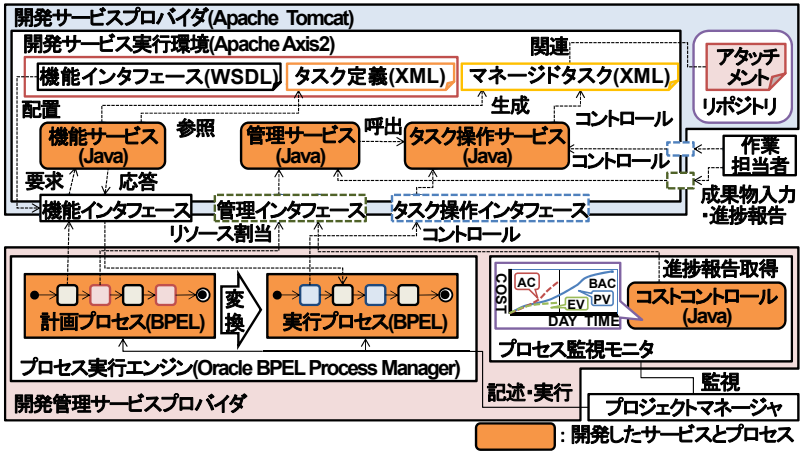

図 12 プロトタイプ全体の構成

7.1.2 開発規模

本稿で開発したサービス，プロセスの記述，監視モニタの開発規模を表 1 に示す．

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		Java	357
		タスク操作サービス		Java	2350
		管理サービス		Java	418
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に Apache Axis2[4]を使用した．

7.2.1 機能インタフェースの実装

機能サービスは，要求と応答の 2 つのオペレーションを基礎として持つ．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．応答を行うことで，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 通りの方法を実現する．ポーリングは要求時に決定したタスクの識別子を用いて行う．コールバックは WS-Addressing[5]を用いることで実現する．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 のプログラミング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．本稿では 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，1 つのオペレーションに対して複数のメッセージが存在する．そのため，オペレーションは様々な構造のメッセージを処理する必要がある．この問題を解決するために，オペレーションの引数や返り値に Axiom (AXIs Object Model)[3]を採用した．Axiom を使用することによって，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，1 つの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を可能にした．

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

7.3.1 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[11]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[12]を使用し，計画プロセスと実行プロセスを記述した．

7.3.2 プロセス監視モニタ

本稿ではプロセス監視モニタとして，PMBOK のコスト管理における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．本稿では，管理インタフェースから，コスト情報を扱う WSDL の portType を用いてコスト情報を取得し，JFreeChart[9]を用いて EV を視覚的に表現した．

8. 例題による開発と開発管理

開発モデルのプロセスに基づき実装と試験サービスを連携し，SOSD 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，コストの観点から EV を評価し，開発サービスに対する改善を行う．

(1) 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機能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 13)．

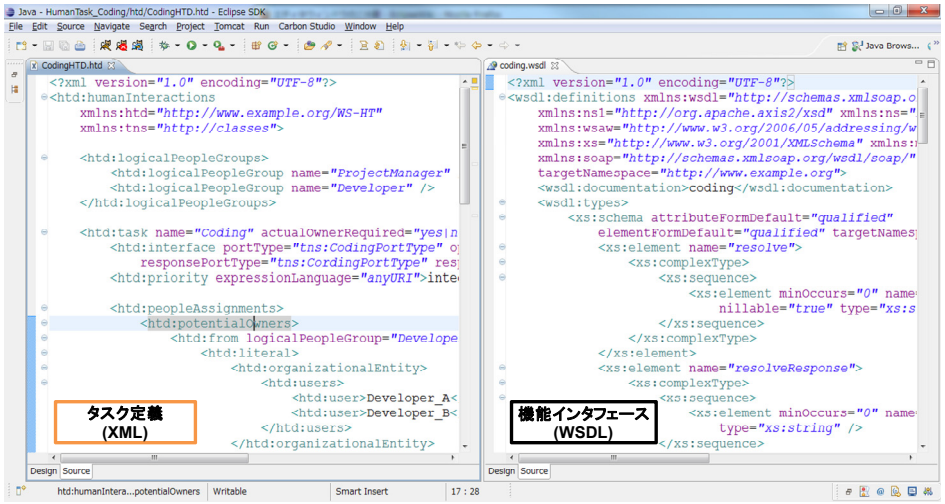

図 13 機能インタフェースとタスク定義

(2) 開発サービス連携の記述と実行

JDeveloper を用いて実装と試験サービスのサービス連携を BPEL で記述した．提案した開発の 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，記述したプロセスを BPEL サーバに配置し，起動することで開発を実行した．

(3) 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の進捗を一定時間ごとに入力した．この入力した管理情報は(4)で EV として視覚的に表現される．

(4) EV の測定と開発改善

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をコストの観点に基づき EV を測定した(図 14)．測定した EV(図 14 の A)から，AC(Actual Cost)が PV(Planed Value)よりも超過していることを確認した．確認した EV に基づき，管理インタフェースの改善のオペレーションを用いてプロジェクトの許容する範囲内で AC が PV を超えないように BAC を増やした．再び EV の測定(図 14 の B)から AC が PV を超過していないことを評価し，開発の改善を実現した．

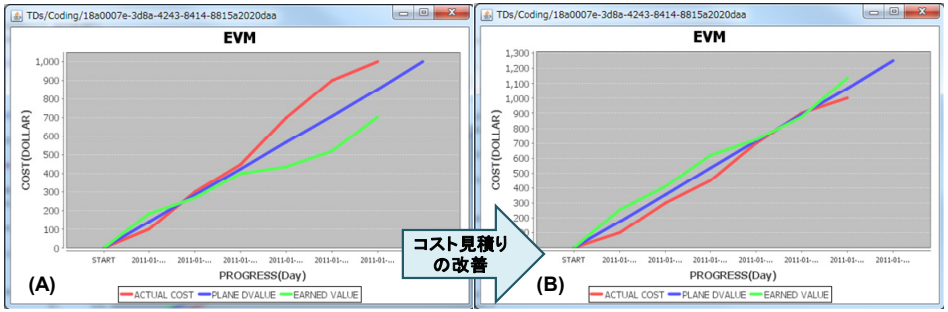

図 14 EV(A)と改善後の EV(B)

9. 評価

(1) WS-HumanTask に基づく開発管理サービスメタモデル設計の妥当性

提案モデルは 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．これにより提案モデルは特定の開発モデルに限定されずに適用可能である．

(2) メタインタフェースの妥当性

本稿ではタスク操作インタフェースと管理インタフェースを開発サービスのメタインタフェースとして設計した．メタインタフェースは開発サービスの実行とは独立したインタフェースである．また，メタインタフェースは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ち，開発活動の進捗から必要に応じて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を行うことが可能である．

(3) プロトタイプからのソフトウェア開発管理サービスメタモデルの妥当性

1) BPEL を用いた開発サービスの連携による SOSD 実現可能性の確認

本稿の例題として実装した実装(Coding)サービスと試験(Testing)サービスは BPEL 実行エンジンを用いて連携することで SOSD を実現可能とする．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プロセスで呼び出し可能になる．

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のタスクの進捗に EV を用いて確認した．確認した 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プロセスの変更を行った．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．

10. 今後の課題

(1) 開発管理サービスのサービス連携

本稿ではプロトタイプから開発サービスの連携と実行中の開発サービスの実行管理を行った．しかし開発管理サービスの連携方法については定義していない．PMBOK で定義される各アクティビティをサービスとして捉え，連携を記述することが必要になる．

(2) 開発サービスに対する作業担当者の決定方法

本稿では呼び出す開発サービスの作業担当者をあらかじめ決定した上でサービスの呼び出しを行っている．そのため，サービスを呼び出した上で複数の作業担当者のリストから最適な作業担当者を決定する仕組みが必要になる．

11. まとめ

本稿では WS-HumanTask の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た．また開発管理を行うための管理インタフェースは，開発プロジェクト要求による管理対象とする管理情報を WSDL の記述に使用される portType を選択することで，プロジェクト要求の管理対象を選択することを可能にした．また，提案モデルに基づき設計した開発サービスのサービス連携と PDCA サイクルで管理するための実行環境を提案した．

提案モデルを実行するための開発サービスプロバイダと開発管理サービスプロバイダをプロトタイピングし，サービスの連携と EV に基づく開発管理を行うことにより提案するモデルの妥当性を確認した．

参考文献

1) 青木 利晴, Web サービスコンピューティング，電子情報通信学会, 2005.
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., Web Services Addressing (WS-Addressing), 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) 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.
11) Oracle BPEL Process Manager, http://www.oracle.com/technetwork/middleware/bpel/
12) Oracle, JDeveloper, http://www.oracle.com/technetwork/developer-tools/jdev/
13) PMI, A Guide to the Project Management Body of Knowledge, 4th ed., PMI, 2008.