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

M2009MM015　長澤 伸治

指導教員　青山　幹雄

1. はじめに

グローバルソフトウェア開発では，統一的な実行と管理が困難である．本稿では人を含む開発活動をサービス(ソフトウェア開発サービス)と捉えモデル化し，さらに開発管理可能なインタフェースを拡張する．これをソフトウェア開発管理のサービスメタモデルとしモデルの実行環境を提案する．プロトタイプより評価し，その有効性を示す．

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ため，サービスの開発管理を可能とし，進捗を取得する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である必要がある．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メタインタフェースを定義する必要がある．

3. 関連研究

3.1. 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(SOSD: Service-Oriented Software Development)のモデル化を提案している[1]．モデル化した開発プロセスを SOA[2]基盤上で実行可能な記述の BPEL4People[3]に変換する方法を提案している．しかし，開発サービスの具体的な開発管理方法は未定義である．

3.2. WS-HumanTask

WS-HumanTask は SOA 基盤上で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[4]．ヒューマンタスクの構成要素と制御するためのインタフェースが定義される．

4. アプローチ

プロジェクト管理可能なソフトウェア開発サービスのモデル化を以下の手順で行う(図 1)．

(1) 管理インタフェース定義

開発のモデルと開発管理モデル(PMBOK[7])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管理インタフェースを定義する．

(2) WS-HumanTask に基づく開発サービスのモデル化

WS-HumanTask の仕様から開発サービスに必要な構成要素を示す．

(3) 管理インタフェースの拡張と構造定義

(1)で定義した管理のインタフェースを(2)のモデルに拡張し，さらに管理インタフェースが持つべき構造を定義する．

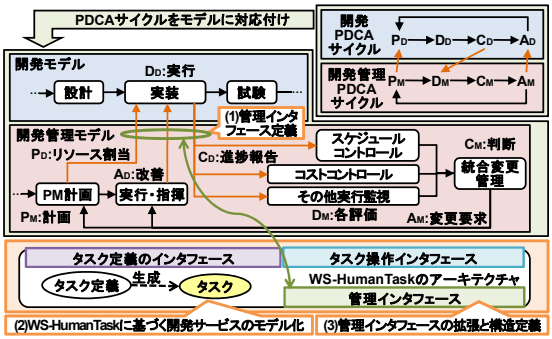

図 1 開発管理サービスのモデル化プロセス

5. ソフトウェア開発管理のサービスメタモデル

5.1. 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．

5.2. 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 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示す(図 2)．

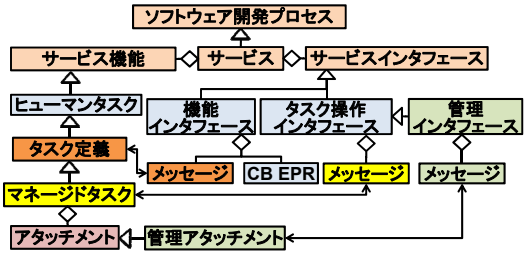

図 2 ソフトウェア開発管理のサービスメタモデル

図 2 のメタモデルは次の要素から構成される．

(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，ヒューマンタスクの実行を実現するための構成を持つ．

1) タスク定義

ヒューマンタスクであるサービスの実行に必要な要素である作業内容や制約を記述する．

2) マネージドタスク

サービス呼出し時に呼出しのメッセージとタスク定義に基づき構成される．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．

(2) サービスインタフェース

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．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．

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 3)．

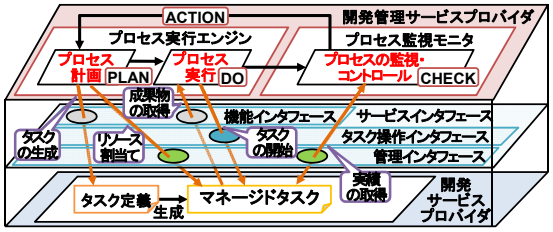

図 3 ソフトウェア開発の PDCA サイクル

(1) プロセス実行エンジン

プロセス計画ではヒューマンタスクの組合せをプロセスとして定義する．ヒューマンタスクの実行順序と各ヒューマンタスクの開発リソースの割当てを記述する．プロセスに基づき機能インタフェースを用いてヒューマンタスクを生成する．プロセス実行はプロセス計画で記述したプロセスに基づき実行可能なプロセスに変換し実行する．タスク操作インタフェースを用いて生成したヒューマンタスクを開始する．

(2) プロセス監視モニタ

プロセス実行エンジンから実行するヒューマンタスクの識別子を取得する．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態を監視，コントロールを行う．

6.2. 開発管理の振舞い

(1) 開発管理の計画と実行のフェーズ(図 4)

プロジェクトマネージャ(PM)は計画プロセスとしてサービスであるヒューマンタスクの連携を記述する．記述した計画プロセスにサービス呼出しの要素として，作業担当者や作業のデッドラインをリソースの割当て情報として付加する．計画プロセスを実行し，開発のタスクを生成する．タスク生成後，PM は計画プロセスを実行プロセスに変換し，開発を起動する．実行プロセスは，各サービスの開始と終了の制御と成果物を授受するメッセージ交換を記述する．

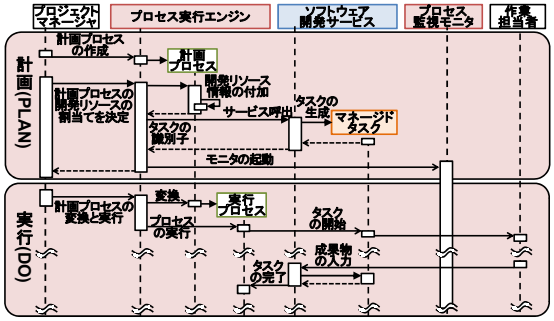

図 4 開発管理の計画と実行のフェーズ

(2) 開発管理の評価と改善のフェーズ(図 5)

開発管理の評価は計画フェーズで起動したプロセス監視モニタからタスクの進捗報告を取得する．タスクの進捗は作業担当者が管理インタフェースを用いて入力する．

改善は評価フェーズで確認した EV(Earned Value)を基に開発時のボトルネックを発見することで，計画プロセスの変更や各タスクへのリソースの再割当てを必要に応じて行う．

計画プロセスの変更は計画フェーズと同様にサービス連携のプロセスを変更する.

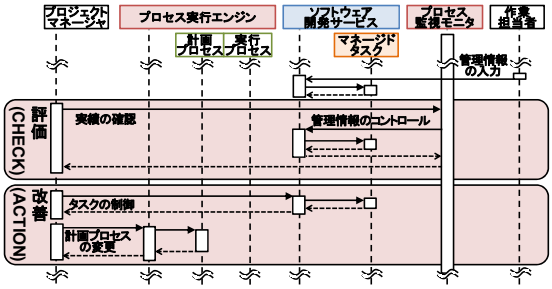

図 5 開発管理の評価と改善のフェーズ

7. 実行環境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と開発規模

開発したプロトタイプ全体の構成を図 6 に示す.

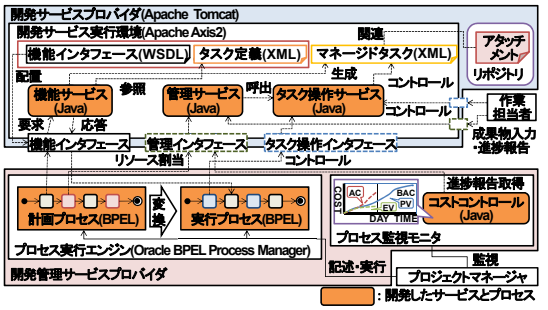

図 6 プロトタイプ全体の構成

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 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はコールバックとポーリングの 2 通りの方法で実現する.

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するオペレーションをタスク操作サービスとして実装した.

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)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になる.

7.3. 開発管理サービス実行環境の開発

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[5]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[6]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 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 を用いて EV を表現した.

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.

(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プロセス実行エンジンに配置し, 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の使用から実装サービスをコストの観点に基づ

き EV を測定した(図 7). 測定した EV(図 7 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(Budget at Completion)を増やした. 再び EV を測定(図 7 の B)し, AC が PV を超過していないことを評価した.

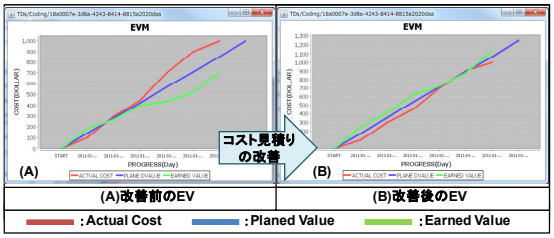

図 7 EV(A)と改善後の EV(B)

9. 評価

9.1. 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行えるようになる. 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が取得可能になる. 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.

9.2. 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オプションのインタフェースであり, 機能インタフェースの呼び出しのみでも開発サービスは実行可能である. 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.

9.3. 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 が実現可能になる. 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

プロトタイプにより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. 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, 開発サービスの連携と EV に基づく開発管理を行い, 提案するモデルの妥当性を確認した.

参考文献

[1] 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, 情報処理学会第 167 回 ソフトウェア工学研究会, Mar. 2010, pp. 1-8.

[2] T. Erl, Service-Oriented Architecture, Prentice Hall, 2005.

[3] D. Ings, et al., WS-BPEL Extension for People (BPEL4People) Specification, Ver. 1.1, May 2010.

[4] D. Ings, et al., Web Services Human Task (WS-Human Task) Specification, Ver. 1.1, May 2010.

[5] Oracle BPEL Process Manager, http://www.oracle.com/technetwork/middleware/bpel/

[6] Oracle, JDeveloper, http://www.oracle.com/technetwork/developer-tools/jdev/

[7] Project Management Institute, A Guide to the Project Management Body of Knowledge, 4th ed., PMI, 2008.