グローバルなオフショア開発におけるビジネスプロセス知識の共有手法の提案

矢口隆明† 岩田彰† 伊藤孝行‡ Mark Klein‡ 福田直樹*
†名古屋工業大学大学院 ‡MIT スローン経営大学院 *静岡大学

1 はじめに

今日，世界的な情報通信技術の発展により，グローバルなアウトソーシングによるオフショア開発が活況を呈している [1]．日本においても，国内ソフトウェア技術者の人材不足やソフトウェア開発規模の拡大などで，近隣諸国へのオフショア開発が行なわれている．オフショア開発の主要な目的は，市場の低価格化と短納期要求に対して，開発コストの削減と優秀なエンジニア確保による品質や安定性・信頼性を確立していくことである．

以上のように，オフショア開発はグローバルなソフトウェア開発の競争を勝ち得るための重要な方法論でもある．しかし，日本の多くのソフトウェア開発企業は，オフショア開発の利点を有効に活用できず，開発コストの増大や開発納期の遅延などによる不適切なプロジェクトを多く抱えているという問題がある．

上の問題は，言語の違い，文化の違い，商習慣の違いなどによる，コミュニケーションの困難さに起因することが多く，表出化しづらい暗黙知の共有ができていないためである．

一方，一般的なソフトウェア開発方法論として，ウォーターフォールモデルが元来使われてきた．しかし，近年，ウォーターフォールモデルの欠点が多く指摘され，アジャイル／反復的なソフトウェア開発方法論が使われている [2]．

オフショア開発においても状況は同じで，アジャイル／反復的なソフトウェア開発 [3] が望ましい．しかし，オフショア開発では，もともとコミュニケーションが困難であるため，さらに多くのコミュニケーションが必要な方法論を導入するためには，効率的な方法論や支援ツールが不可欠である．

そこで，本稿では最適なオフショア開発を推進するために潜在的な知識と情報を正しく共有し，円滑なソフトウェア開発を効率的に実行するために，エンタープライズソフトウェア開発に関して，対象となるビジネスプロセス知識を共有する方法論とその支援ツールを提案する．

2 オフショア開発とその課題

既存の典型的なオフショア開発は，以下のように行われる [4]：プロジェクト全期間にわたって，小さな開発チームが顧客側に「常駐」し，システム統合，導入，テスト，など，顧客と直接コミュニケーションを取りながら行う．初期の要求仕様は顧客側で決定され，その詳細の作成はオフショアされる．次に，開発チームが構成され，開発がオフショアで開始される．そしていったんソフトウェアが完成したら，顧客側に送付され，顧客側にいる開発チームによって，統合，導入，テストが行われる．

明らかな課題は，コミュニケーションの充実である．既存の典型的なオフショア開発では，顧客側に常駐する開発チームが長期間滞在することが多い．大規模な予算を用いたソフトウェア開発でなければ成功しにくい一方，大規模なソフトウェア開発ではますますコミュニケーションが必要になってしまうというジレンマがある．既存のオフショア開発では，コミュニケーションを十分に行うために，様々な方法論，システム，及びツールを用いている．方法論としては，情報のオープンな共有，ツールとしてはビデオ会議システム，電話会議などがある．

本稿では，エンタープライズソフトウェア開発に関して，対象となるソフトウェアの仕様をビジネスプロセスモデルとして共有しながら，ソフトウェア開発を進める方法論とそのツールを提案する．

3 ビジネスプロセスモデルと共有支援ツール

近年，多くのビジネスプロセスモデルが提案されている．中でも最も良く知られているのは，MIT Process Handbook(PH) である [5][6]．PH におけるビジネスプロセスのモデル化は非常にシンプルであり，そのため，多くのビジネスプロセスを簡潔にモデル化することができる．PH の特徴は，ビジネスプロセスを行動（action）の依存関係（dependency）で表すことである．依存関係には，フロー（flow），シェア（Share），及びフィット（Fit）がある．フローは，既存のモデルで一般的に定義されるプロセスの遷移を表す．シェアは，いくつかの行動が資源などを共有することを示す（２人で１つのマシンを共有する）ために用いられる．フィットは，いくつかの行動によって，一つの資源が生成される（２人で１つの部品を生成する）ことを表す．資源への依存関係をシンプルに表現できる点が優れている点である．

本稿で提案するビジネスプロセスモデルは，PH をベースに，設計開発の対象となるビジネスプロセスをモデル化し共有する．一方，地域的に分散した設計方法論の観点から言えば，なぜそのようなプロセスを設計したかという理由付けが極めて重要になる [7]．理由付けは多人数による議論のヒストリであっても良いし，資源的な制約でも良い．文献 [7] では，グラフモデルによってそのような理由付けを表現できることが示されている．そのような設計や変更に関する理由付けをビジネスプロセスのモデル化に取り入れることは今後の課題である．

本稿で提案するビジネスプロセス共有ツール（図 1）では，対象となるビジネスプロセスを PH に基づいて編集・更新しながら，オントロジー記述言語 OWL(OWL-s)[8] と RuleML[9] を組み合わせて記述しておくことで，ウェブ上での，ビジネスプロセスの共有，検索，再利用を効率化する．また，ビジネスプロセスの推薦機構，および，ビジネスプロセスを売買するようなマーケット機構に関しても，現在検討を進めている．

4 ビジネスプロセス作成例

提案手法を用いて，例として内職を請け負う比較的小さな企業のビジネスプロセスのモデル化を紹介する．以下は，ビジネスプロセスの基本部分であり，このビジネスプロセスに基づいてエンタープライズソフトウェアが開発されるとする．

LaTeX 2ε Style File for National Convention of IPSJ
†Yoshio KAKIZAKI ‡Taro KYOUCHO
†Graduate School of Science and Technology, Tokai University Unified Graduate School
‡School of Information Technology and Electronics, Tokai University

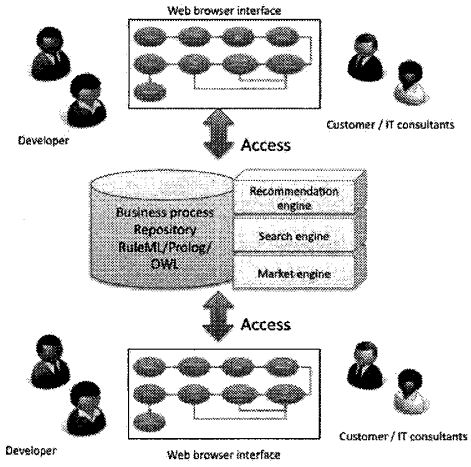

図 1: ビジネスプロセス共有ツール

(1) 顧客からの照会：タスク（例えば，１万個の小さなカレンダーの作製等）に関して，企業に照会が来る．**(2) 企業による見積り**：この企業は照会されたタスクについてのコストを見積る（例えば，１つのカレンダが２０円など）．**(3) 顧客からの注文**：見積に基づいて，顧客が注文をする．**(4) 顧客から材料の受け入れ**：顧客は製品を作るための材料を送付する（例えば，カレンダーを作るための紙や木枠など）．**(5) タスクの割当て**：タスクを内職をする人（内職人）に割り当てる（例えば，各内職人が１週間で１００－３００のカレンダーを作製する）．**(6) タスクの検査**：完了したタスクを検査する（例えば，完成した製品の検査，不良品の検査）．**(7) 最終検査**：全体的にタスクの最終検査をする（すべての製品の数などを検査する）．**(8) 納品**：顧客に製品を搬入する（１万個のカレンダーを顧客に届ける）．**(9) 支払い**：顧客からの支払いを受ける（２０万円の代金を受け取る）．

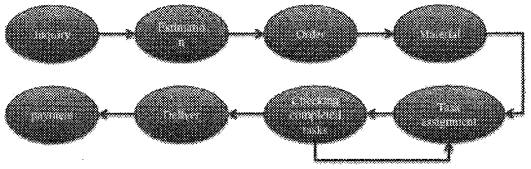

図 2: ビジネスプロセスの表現

図２に，上記の例をモデルとして表す．簡単な状態遷移モデルとして表すことができる．遠隔地で共同でソフトウェア設計を進める開発者らは，ウェブインターフェースを用いて，図２のようなモデルを設計し，さらに詳細化する．すべての変更はタイムスタンプとともに DB に保存される．遠隔地にいても，作るべきビジネスプロセスは明確に共有することができ，かつ，アジャイル／反復型のソフトウェア開発において，顧客からも変更要求を受付易い．また，すべての変更が保存されるため，設計や変更のための書類の共有が容易になる．

例えば，ユーザらは，ビジネスプロセスを最も単純な形，注文，生産，および納品から設計を始めることができる（図 3）．そして，少しずつ詳細化（具体化）していく（図 4）．最終的には図２の形になる．実際の現場では，さらに詳細化が行われ，それぞれのグループだけの詳細な部分まで共有し，すべて記録保管する．

図 3: 最も簡単なビジネスプロセス

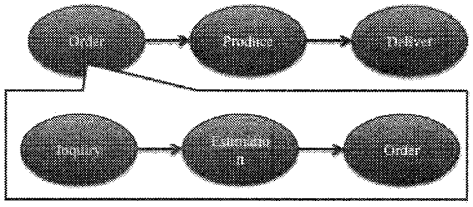

図 4: 段階的な詳細化

5 まとめ

実世界のニーズとして，オフショアによるグローバルソフトウェア開発を効率的に進めるための方法論の確立が求められている．近年のオフショアは次の２つの理由で，コミュニケーションや情報共有を以下に効率化するかが成功の鍵となっている．理由は，第１にアジャイル／反復的な開発手法の導入と，第２に地理的距離や文化的差異である．本稿では，ソフトウェアとして開発するビジネスプロセスを，シンプルなモデルで記述し，ウェブを使って共有することで，コミュニケーションや情報共有を促進する方法論を提案した．今後は，本モデルの精緻化，本ツールのより具体的な実現及び，本方法論を具体的に現場で活用する場合の新たなマネージメントの方法（現場知の収集方法など）に関して研究を行う．

参考文献

[1] Ravi Aron and Jitendra V. Sigh. Getting offshoring right. *Harvard Business Review*, pp. 135–142, 2005.

[2] Michael A. Cusumano, Alan MacCormack, Chris F. Kemerer, and Wiliam Crandall. Software development worldwide: The state of the practice. *IEEE Software*, pp. 2–8, 2003.

[3] Craig Larman and Victor R. Basili. Iterative and incremental development : A brief history. *IEEE Computer*, pp. 47–56, june 2003.

[4] Anandasivam Gopal, Tridas Mukhopadhyay, and Mayuram S. Krishnan. The role of software process and communication in offshore software development. *Communications of The ACM*, Vol. 45, No. 4, pp. 193–200, 2002.

[5] Thomas W. Malone. *The Future of Work : How the New Order of Business Will Shape Your Organization, Your Management Stype, and Your Life*. Harvard Business School Press, 2004.

[6] Thomas W. Malone. *Organizing Business Knowledge, The MIT Process Handbook*. The MIT Press, 2003.

[7] Mark Klein. Capturing design rationale in concurrent engineering teams. *IEEE Computer. Special Issue on Computer Support for Concurrent Engineering*, Vol. 26, No. 1, pp. 39–47, jan 1993.

[8] W3C. Owl web ontology language guide, 2004.

[9] The RuleML Initiative. Ruleml tutorial, 2005.