成果物に基づいたソフトウェア開発プロジェクトの階層的な可視化フレームワーク

A Framework for Visualizing Software Development Project by Layered Mapping based on the Deliverables

大蔵 君治* 中道 上† 飯田 元‡

あらまし ソフトウェアに起因するシステム障害は今なお頻発しており，その原因の一つとして，ソフトウェアの持つ「目に見えない」という特徴が，開発プロジェクトの進捗把握を困難にしていることが挙げられる．本研究では，成果物に着目したプロジェクトの可視化手法を提案する．ソフトウェア開発プロジェクトにおける成果物は，プロジェクトの進捗を直接的に表すものであり，また，プロジェクトマネージャとプログラマの双方にとって理解が容易である．本研究では，コンセプト層・モデル層・プロダクト層の三つの層と時間軸から成る可視化手法を用いてプロジェクトの進捗管理を支援する．

1 はじめに

継続的に成長を続ける現代の情報化社会において，ソフトウェア開発の需要は高まる一方である．しかしながら，銀行のオンラインシステムや空港管制システムの障害といったソフトウェアに起因する事故は今なお頻発している．その原因の１つとして，ソフトウェアの「不可視性」が挙げられる [1]．ソフトウェアは建築等におけるモノ作りとは異なり，成果物が目に見えるものではないため，欠陥（バグ）の発見が難しく，開発中のシステムが現在どの程度完成しているのかといった全体像が掴みづらい．そのため，ソフトウェア開発プロジェクトを成功させるためには，進捗管理が重要となる．

これまで，プロジェクトの進捗管理には WBS (Work Breakdown Structure) [2] やガントチャート [3] といった手法・図が用いられてきた．また，それらを描いたり，管理したりするためのツールとしては Microsoft Project や Excel が広く用いられている．こういった方法は，プロジェクトマネージャ（以下 PM と言う）が見積もったスケジュールを可視化し，進捗状況に合わせて適宜修正していく為には大変有用である．しかし，正しくスケジュールを組むためには，PM が該当プロジェクトを常に調査する必要があり，高い管理能力，そして経験と勘が求められる．このような作業を主軸とした進捗管理では，PM と開発チームとの間に意識のずれを生みやすく，プロジェクト全体で成果物に対するイメージが異なるという状況に陥りがちである．

本研究では，従来のヒト（タスク＝作業時間）を主軸とした管理ではなく，成果物観点からプロジェクト構造を可視化し，進捗の把握を支援する手法を提案する．ソフトウェアを始め，設計書や仕様書といったソフトウェア開発プロジェクトにおける成果物は，プロジェクトの進捗を直接的に表すものである．我々は，成果物が作成されていく段階を記録し，その変遷を可視化することで，PM だけでなく開発チームにとっても理解しやすい進捗の表現が可能であると考え，その可視化手法についてのフレームワークを制定した．なお，本フレームワークは開発プロセスの制定や管理計画を行うものではなく，プロジェクト構造の全体を可視化することで「意識共有の支援」を行い，構造の変遷から「進捗の遅れている箇所を発見」するもので

*Kimiharu Ohkura, 奈良先端科学技術大学院大学 情報科学研究科

†Noboru Nakamichi, 南山大学 数理情報学部 情報通信学科

‡Hajimu Iida, 奈良先端科学技術大学院大学 情報科学研究科

ある．よって，従来のタスク指向の進捗管理に置き換わるものではなく，PM の補助，及び開発における協調作業の円滑化のために用いることが望ましい．

2　関連研究

成果物に着目したソフトウェア開発プロセスのモデル化手法としては，田中らの提案した PReP モデル [4] がある．CMM [5] あるいは CMMI [6] 等に基づいた開発プロセスの改善活動では，組織のベストプラクティスから定義する「標準プロセス」の導入が推奨されている．しかし，従来のタスク（作業）指向であるプロセス記述法は現場の開発要員にとっては理解が難しく，そのような記法で標準プロセスを定義しても，実際の開発現場においては用いられずに形骸化しがちであることを問題点として挙げている．PReP モデルはこの問題に対し，現場の開発要員にとっても直感的な理解が可能である成果物に着目し，成果物間の関連から開発プロセスを記述するアプローチをとっている．モデルはその理解容易性と記述の柔軟性から，開発プロセスの定義のみならず，プロジェクトの進捗管理へも適用可能であり，実企業においても一定の評価を得られている．

リアルタイム性を持ったソフトウェア開発プロジェクトの可視化システムとしては，Ohira らの提案した Empirical Project Monitor [7]（以下 EPM と言う）がある．EPM は，ソフトウェア開発において一般に用いられる版管理ツールやバグ追跡システムといった開発支援ツールと連携して動作する．EPM は開発支援ツールが自動で記録するログデータを逐次取得することで，開発者に負担を掛けることなく，且つリアルタイム性を持ったプロジェクトの分析表示を実現している．EPM が分析した各種のメトリクス（測定指標）は，Web サーバを介して利用者に図示されるため，導入後は Web ブラウザ上から容易に分析結果を閲覧できる．また，EPM を実際に導入している実企業も多く，一定の評価が得られている．

以上をまとめると，次の 2 つのことが言える．

- 成果物観点の記述法は現場の開発要員にとっても理解が容易であり，利用しやすい
- 開発要員にとって利用しやすい分析環境は，実際の開発現場においても受け入れられる

我々は上記を踏まえ，従来のタスク指向であった進捗管理ではなく，成果物に着目した進捗管理のための可視化フレームワークを提案し，PM のみならず，開発要員全体にとって利用しやすい分析システムの構築を目指す．

3　フレームワーク

ソフトウェア開発には構造化プログラミングやオブジェクト指向といった様々な開発パラダイムが存在するが，開発形態に関わらず，コーディング作業に入る前にはモジュールに関する種々の設計を行うのが一般的である．本研究では，最終成果物ができるまで（リリースまで）に個々の成果物がどのように出来上がってきたかに着目し，プロジェクト構造の可視化を行う．提案するフレームワークでは，抽象化されたシステムの全体図を示すコンセプト層，作成すべきモジュールの具体的な設計を示すモデル層，作成したモジュールを配置し，設計と関連づけるプロダクト層という 3 つの層が定義され，開発パラダイムに依存しないプロジェクト構造の可視化を行う．コンセプト層は一般的なソフトウェアライフサイクルにおける内部設計，モデル層はプログラム設計に相当するが，組織によっては内部設計とプログラム設計が明確に分かれていなかったり，フェーズの分け方が異なっていたりするため，コンセプト層，モデル層にはその役割と抽象化された表現記法のみを定義し，

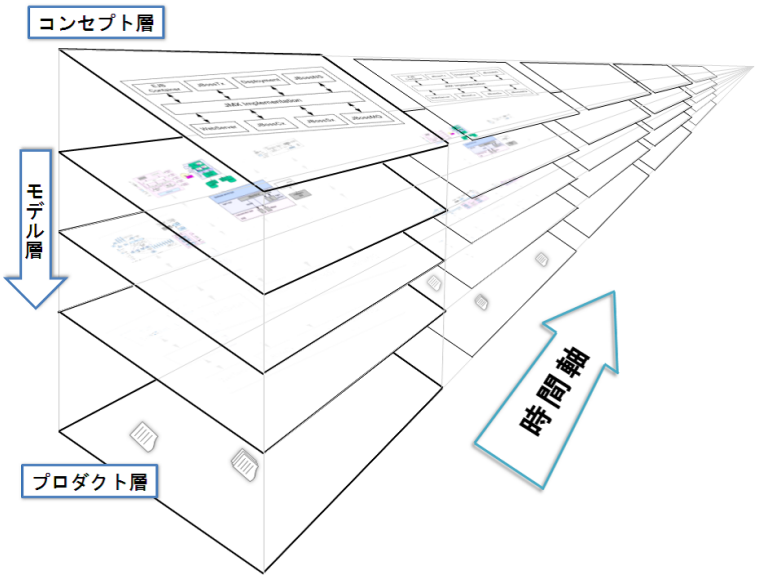

図 1　フレームワークの概念図

特定の図法には依存しないようにした．各層はそれぞれ反復可能である．例えばコンセプト層を 2 層，モデル層を 4 層に分けてもよい．また，前述した 3 層だけでは静的なプロジェクトの状態しか可視化できないため，プロジェクト構造の変化を記録するための時間軸を設けた．図 1 にその概念図を示す．

本フレームワークの主な利用者は PM である．PM が各要員の成果物をまとめあげ，各層に配置していくことで PM の視点からプロジェクト構造の可視化を行う．ただし，閲覧はプロジェクトメンバ全員が行う．これにより，開発チームと PM との間でプロジェクトの全体図を共有することができ，意識のずれを抑えることができる．

次に，それぞれの層の役割について述べる．

3.1　コンセプト層

コンセプト層は本フレームワークを構成する 3 つの層の中で最も上位に位置する．コンセプト層の役割は，開発するシステム全体の，開発者から見た完成予想図を提示することである．既存の手法を用いるとすれば，例えば構造化プログラミング開発であればコンテキストレベルの DFD，オブジェクト指向開発であれば UML におけるパッケージ図，あるいは配置図等が相当する．プロジェクトメンバはコンセプト層から開発するシステムに対する共通のイメージを掴むことができる他，モデル層の設計情報と照らし合わせることで大まかなシステムの完成度合いを掴むことができる．

3.2　モデル層

モデル層は，コンセプト層で示されたシステムの具体的な設計を示す役割を持つ．例えば，各モジュールの DFD，UML におけるクラス図やオブジェクト図などが該当する．モデル層はコンセプト層で示されたシステムを構成するにあたって必要な，全てのモジュールに関する設計情報を最終的に（プロジェクト終了時には）含んでいる必要がある．モデル層の設計情報は，その設計に従って作成された実体がプロダクト層に配置されている場合と，設計情報のみで実体が存在しない場合とで区別できる必要がある．例えば，設計情報のみで実行可能なコードが存在しない場合は

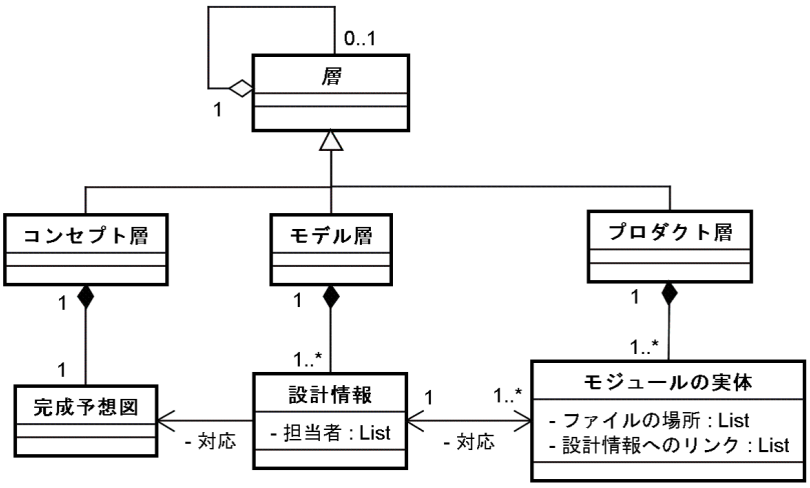

図 2　各層の関係

薄い色で表記するといった描き方が考えられるが，本フレームワークでは図に汎用性を持たせるため，具体的な表記法は定義しない．また，各設計情報には開発担当者（チーム，または責任者）に関する情報が含まれている必要がある．担当者の情報は，成果物指向の進捗管理とタスク指向の進捗管理を結びつける役割を持つ．

モデル層により，各開発要員は自身の担当している箇所が他の開発要員と比べてどれほど遅れているか，あるいは進んでいるかを知ることができる他，各モジュールの設計情報に工数等の規模情報を付加しておくことで，PM はクリティカルパスを視覚的に把握することが出来る．

3.3　プロダクト層

プロダクト層は，モデル層で設計された各モジュールの実体を配置する．モジュールの実体とは，ソースコード，あるいは結合されたライブラリといった，開発者によって実際に作成された各々の成果物を指す．本フレームワークではアイコンを用いたシンボルで表現する．モデル層の設計情報は，プロダクト層に配置された少なくとも 1 つのシンボルと最終的に結びついている必要がある．また，実体への物理的なアクセスパス（ソースコードであれば，そのファイルが置いてある場所）を記しておく必要がある．この 2 つの制約はソフトウェアの設計と実体を関連づけて，システムの複雑化を防ぐために存在する．プロダクト層にシンボルが置かれる毎に，プロダクト層とモデル層の関連，モデル層とコンセプト層の関連が更新され，システムの全体構造を構築していく．

各層の関係を図 2 に示す．図は本フレームワークを使用する上で最低限必要な情報を表したクラス図である．PM は，前述した「工数」のように，各層に対して自由に情報やコメントを付加することができる．構造化プログラミングを用いた小規模な開発プロジェクトを可視化した簡単な例を図 3 に示す．コンセプトレベルの DFD をコンセプト層に，具体的なモジュール設計を下位レベルの DFD で行い，モデル層に配置している．コンセプト層は内部設計にあたるため，ある程度の規模を持ったシステムであれば，2 階層以上のコンセプト層も考えられる．プロダクト層には，単体テストを終了したソースファイルのシンボルを配置し，モデル層で提示した設計情報と対応付けている．

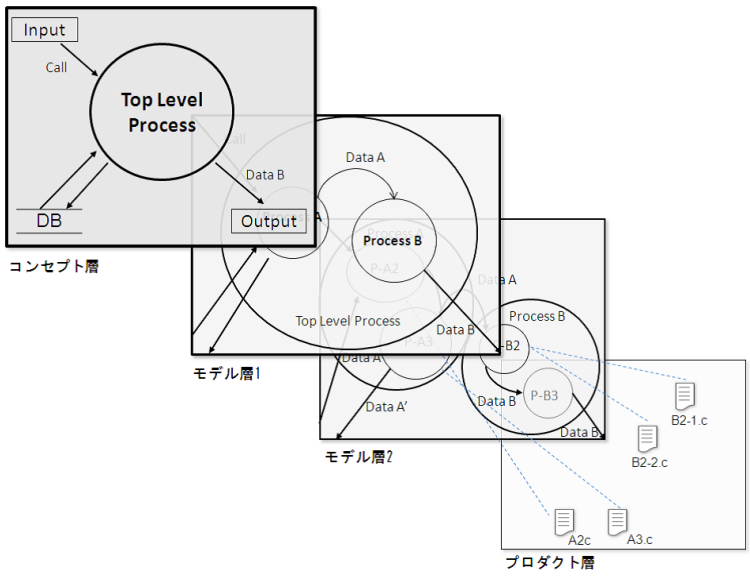

図 3　内部設計とプログラム設計に DFD を用いた場合の例

3.4　時間軸

上記 3 層のビューは，プロジェクトが進むに従い，PM が設定したスパンで定期的に更新を行う．PM は，ビューの変更履歴を見ていくことで，その変化の傾向から開発が遅れている部分，あるいは遅れそうな部分を早期に発見することができる．

4　システムの実装

我々は提案手法の有効性を確認するために，手法に基づいてプロジェクトを可視化するためのシステム「Lap-MAP (Layered Project Map)」を試作した．提案するフレームワークの利用者が PM だけでなく，プロジェクトを構成する要員全てであることを先に述べたが，その場合，システムを実装するにあたって以下の 2 点を満たしている必要があると考える．

- 直感的な操作が可能なユーザインターフェイスと，閲覧に充分な応答速度を持つこと
- 複数人のユーザから，同時にアクセスできること

我々は上記の条件を満たすため，フレームワークを複数のコンピュータから同時にアクセス可能であり，且つ多くのユーザにとって操作の理解が容易である Web ブラウザ上で動作するアプリケーション（Web アプリケーション）として実装した．また，AJAX (Asynchronous JavaScript + XML) と呼ばれる非同期通信技術を用いて，Web ブラウザの画面遷移に掛かる時間的コストを最小限に抑えた．本システムのスクリーンショットを図 4 に示す．ユーザは，階層移動ボタン又はマウスホイールの操作によって階層を上下できる．また，時間軸移動ボタンによってプロジェクトの経過時間を操作することができる．

我々はオープンソースプロジェクトとして公開されているソフトウェア (JBoss Project) のシステム設計図を基に入力データサンプルを作成し，階層表示のテストを行った．その結果，本システムが分析を行うにあたって，複数人からのアクセスに対しても十分な速度で実行可能であることを確認した．

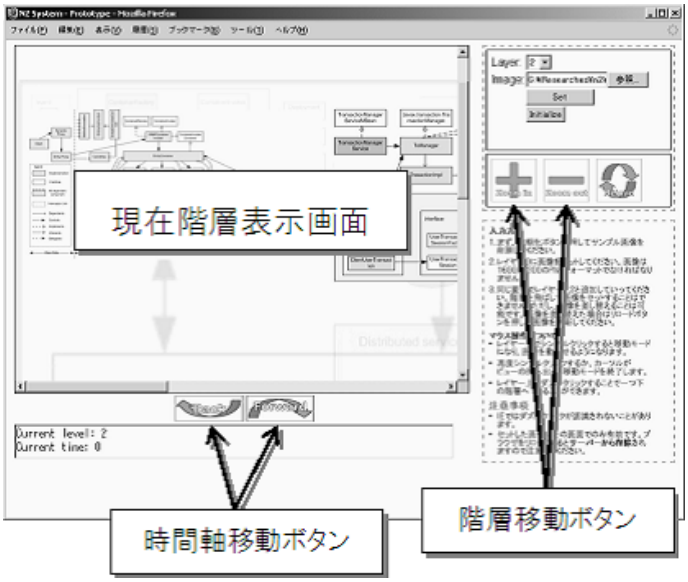

図 4　システムのプロトタイプ

5　まとめと今後の課題

本稿では，成果物の視点からソフトウェア開発プロジェクトを可視化し，進捗管理を支援するためのフレームワークを提案・試作し，手法がもたらす有効性の一部と実現可能性を確認した．しかし，試作したシステムはプロダクト層におけるシンボルの配置や複数層の同時表示といった，本フレームワークを実際の開発現場に適用する上で必要となるであろう多くの事項が実装されておらず，実用段階には至っていない．今後もフレームワークに沿って実装を進め，可能な限りの自動化とユーザビリティの向上を目指す．

謝辞　本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．本研究の一部は南山大学 2008 年度パッヘ研究奨励金 I-A-2 の助成を受けた．

参考文献

[1] Frederick P. Brooks, Jr., The Mythical Man-Month Essays on Software Engineering, Addison-Wesley, 1975.
[2] Robert C. Tausworthe, Work Breakdown Structure in Software Project Management, Journal of Systems and Software, Vol. 1, pp. 181-186, 1980.
[3] Clark, Wallace, The Gantt Chart, a Working Tool of Management, second edition, Sir Isaac Pitman & Sons, Ltd., London, 1942.
[4] 田中康, 飯田元, 松本健一, 成果物間の関連に着目した開発プロセスモデル：PReP, 情報処理学会論文誌, Vol.46, No.5, pp. 1233-1245, May 2005.
[5] Paulk, M., Curtis, B., Chrissis, M. and Weber, C., Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-024, 1993
[6] CMMI Product Team, CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, CMU/SEI-2002-TR-011/TR-012, 2002
[7] Masao Ohira, Reishi Yokomori, Makoto Sakai, Ken-ichi Matsumoto, Katsuro Inoue, Michael Barker, and Koji Torii, Empirical Project Monitor: A System for Managing Software Development Projects in Real Time, In Proc. International Symposium on Empirical Software Engineering (ISESE2004), Vol.2, pp. 37-38, May 2004.