IZMI: 開発者と成果物の編集頻度に着目したソフトウェア開発リポジトリの可視化ツール

大蔵 君治　飯田 元

ソフトウェアが社会において重要なインフラとなっている近年，ソフトウェア開発の改善や効率化のためのプロジェクト分析が重要視されてきている．本研究では，ソフトウェア開発リポジトリに登録されている成果物の編集回数と，開発者の編集頻度に着目した可視化分析手法を提案する．ソフトウェア開発リポジトリを対象とした可視化手法は過去にも提案されてきたが，分析の事前準備にコストが掛かったり，開発時に分析用のツールを導入することが前提とされていたりするという問題があった．本研究では，事前準備が必要なく，開発者に負担を掛けない軽量な可視化分析を実現する．本手法は可視化ツール「IZMI」として実装を行った．

Software project analysis for improvement of the process is becoming an important factor in today's information-oriented society. In this research, we propose a method for visualization of software repository based on the frequency of artifact modifications by developers. Previous studies in this area have some problems caused by cost such as pre-processing or setting environment for this analysis. We propose a lightweight method that achieves a low cost analysis for developers. Our method has been implemented using the analysis tool "IZMI".

1　はじめに

社会においてソフトウェアが果たす役割は非常に重要なものとなっており，ソフトウェア開発の需要は高まる一方である．しかし，近年でもなおソフトウェア開発におけるトラブルは起こり続けており，多くの組織に対し社会的，あるいは金銭的な損失をもたらしている [4]．開発におけるトラブルを防止し，作業の効率化や成果物 (ソースコードやドキュメント等) の品質を高めるためには，開発プロジェクトの分析が必要不可欠である．しかし一方で，開発プロジェクトの分析には人的，時間的，あるいは経済的なコストが掛かるといった問題がある．本研究では，分析環境の組織的な導入を必要とせず，開発者に負担を掛けない軽量な可視化分析手法を提案する．可視化はソフトウェア開発において一般的に用いられる版管理システムのリポジトリを対象とし，次のような情報に着目した．

(1)　編集回数の多い成果物
(2)　編集頻度の高い開発者
(3)　開発者と成果物の関連

上記の情報はいずれもリポジトリから取得可能であり，ソフトウェア開発プロジェクトの分析において重要であると考えられる．例えば，(1) は開発システムにおいて影響の強い要素であると考えられ，障害の原因となりやすい成果物である．(2) はプロジェクト，あるいはチーム内において中心的に開発を行っている要員を示すほか，開発効率や各開発者に掛かる負荷を俯瞰することが可能となる．(3) は，バグを発見した際や人員の入れ替わりが発生した際に，適切な修正担当者や開発担当者の割り当てを行うために必要となる情報である．本研究の目的は，これらの情報を低コストで把握することである．

IZMI: A Tool for Visualization of Software Repository based on Frequency of Artifact Modifications by Developers.

Kimiharu Ohkura, Hajimu Iida, 奈良先端科学技術大学院大学情報科学研究科, Graduate School of Information Science, Nara Institute of Science and Technology.

コンピュータソフトウェア, Vol.28, No.3 (2011), pp.147–152.
[研究論文 (レター)] 2010 年 12 月 15 日受付.

2 関連研究

ソフトウェア開発リポジトリを対象とした分析支援手法は，リポジトリマイニングやソーシャルネットワーク分析の分野において広く研究されている．Hattori らは，プロジェクト情報をリアルタイムにチームで共有するためのチーム開発支援ツール「Syde」の開発を行った[3]．既存の版管理システムでは，開発者によってコミットの頻度や粒度が異なるために，異なるプロジェクトや開発者を横断的に分析し，比較することができないという問題があった．Syde は，統合開発環境である Eclipse を用い，コード編集に同期して自動で編集履歴を記録することで，属人性を排除したリポジトリ分析を行うことができる．また，Hattori らは誰がそのコードに最も詳しいかを示すコード所有権という概念を用いて成果物と開発者の関連を分析している．しかし，Hattori らの手法は開発環境に独自のツールを導入することを前提としているため，組織的な開発においては準備にコストが発生する．

Sarma らは，ソーシャルネットワーク分析によって抽出された開発者同士のネットワークと，モジュール依存度をもとに抽出されたファイルネットワークを関連づけて提示するツール「Tesseract」を開発した[5]．Tesseract を用いることで，例えばある開発者に着目し，その開発者と関連の強いほかの開発者やモジュールを同時に提示するといったことが可能となる．Tesseract を用いて分析を行う際は，特別な開発環境を導入する必要はない．しかし，モジュールと開発者の関連づけはすべて手動で行う必要があるため，分析のための事前準備に掛かるコストは高い．

既存の版管理システムのデータを用いて可視化分析を行う先行研究としては，D'Ambros らの研究がある．D'Ambros らはモジュールの依存関係を可視化する手法[2]や，コミット数，バグ報告数などから各モジュールにおける開発の活発性を計測，可視化する手法[1]などを提案している．しかし，D'Ambros らの可視化手法は個別のビューによって実現されており，成果物と開発者の両方の観点から時系列を揃えて分析することは困難である．また，一部の手法は版管理システムに加え Bugzilla などの障害管理システムの導入を前提としている他，FAMIX メタモデル[6]に関する理解も必要となる．本研究で提案する可視化手法は，版管理システムが導入されているプロジェクトであれば特別な開発環境の導入や手動による事前処理を行うことなく適用可能であり，特別な前提知識を必要としない．

3 可視化手法

3.1 概念

提案する可視化手法のキーアイデアは，成果物に対する単位時間あたりの編集回数を色の濃淡で表現することである．成果物を可視化する場合の概念を図 1 に示す．単位時間 (図 1 では日) あたりの編集回数を時系列で並べて色分けすることで，開発が活発な時期や，当該プロジェクトにおいて重要だと判断できる成果物を視覚的にわかりやすいかたちで把握することができる．また，縦軸を成果物ではなく開発者に置き換えることで，編集頻度の高い開発者を把握することができる．

3.2 ツールの実装

我々は，提案手法を可視化ツール「IZMI[†1]」として実装した．IZMI は「成果物表示モード」と「開発者表示モード」という 2 種類の画面をもち，分析者が不必要と判断した情報はフィルタリング機能によって不可視にすることが可能である．実装は C# .NET を用いて行った．

3.2.1 入力データ

IZMI は Subversion または CVS のリポジトリに記

図 1 可視化の概念図

†1 http://sdlab.naist.jp/members/tetsuya-o/izmi_introduction.html

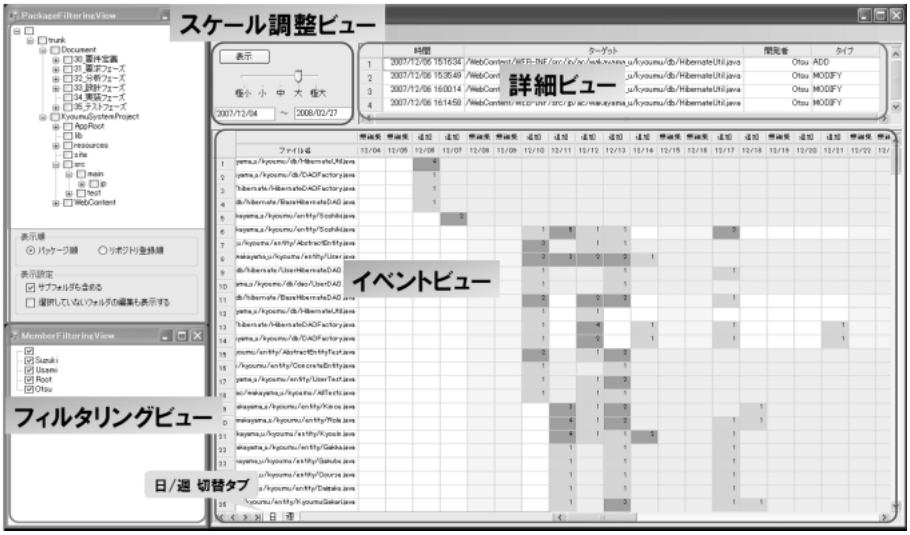

図 3 IZMI の画面 (成果物表示モード)

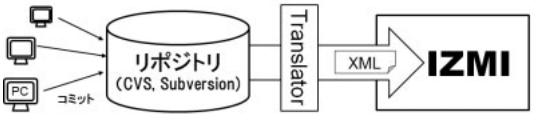

図 2 入力までの流れ

録された開発履歴を用いて可視化を行う．リポジトリのデータは一度 XML フォーマットへ変換される．XML にはコミットの日時，タイプ (追加，修正，削除等)，ファイルパス，オーナー (コミットを行った開発者名) といった情報が記録される．データの変換には EPM(Empirical Project Monitor) [7] の付属ツールである EPM Translator を用いている．EPM Translator はローカルで実行可能な単機能のアプリケーションであり，セットアップは容易である．入力までのデータの流れを図 2 に簡単に示す．

3.2.2 成果物表示モード

成果物表示モードは，縦軸に成果物 (リポジトリに登録されているすべてのファイル)，横軸に時間をとり，各成果物の編集回数を時系列で表示するモードである．縦軸の成果物は，リポジトリに登録された順番，またはパッケージ名の昇順でソートすることができる．起動時は登録順で表示される．図 3 にスクリーンショットを示す．単位時間は日単位と週単位をタブ操作によって切り替えることができる．これらの情報は IZMI のメインとなる画面 (イベントビューと呼ぶ) にテーブル形式で表示される．各セルにはコミット情報が関連づけられている．表示されている数値は編集回数である．編集が 1 度も行われなかった日・週は空白となっている．編集回数が多いほどセルの色は濃くなる．また，編集が 1 度も行われていない日・週であっても，リポジトリにファイルが存在すれば薄い色で表示される．なお，日単位の表示でリポジトリにファイルが存在せず，かつその列が休日 (本来であれば開発が行われていない日) であればセルは薄い青色となる．これにより，例えば「休日に複数の開発者が何度もコミットしているような期間は納期までの余裕がない」といった推察を行うことができる．現在は単純に土日のみを休日としているが，将来的には外部ファイルによってユーザが休日を自由に定義できるようにする予定である．

これらの色分けにより，頻繁に編集が行われている成果物や，長い期間をおいて編集された成果物，あるいは新規作成以降ほとんど編集されていない成果物といった成果物ごとの特徴を一目で把握することができる．また，各セルをマウスクリック等により選択することで，上部に設けられたフレーム (詳細ビューと呼ぶ) に選択中の成果物に関するコミット履歴を表

示することができる．詳細ビューを確認することで，その成果物を中心的に編集している開発者や，具体的な編集内容を把握することができる．なお，すべての色はユーザが自由に変更可能である．

3.2.3 開発者表示モード

開発者表示モードは，成果物表示モードの縦軸を開発者に置き換えたものである．スクリーンショットを図 4 に示す．各セルには，その日・週に当該開発者が行った編集の回数が表示される．成果物表示モードと同様に編集回数が多いほどセルの色は濃くなる．開発者表示モードでは更に，リポジトリ上のディレクトリをチェックボックスによって選択することで「選択したディレクトリのファイルのみを編集」(図 4 の画面では赤で表示),「選択したディレクトリ以外のファイルを編集」(図 4 の画面ではグレーで表示),「選択したディレクトリのファイルとそれ以外のファイル両方を編集」(図 4 の画面では青色で表示) の 3 パターンを異なる色で表示することができる．これらの色も編集回数によって濃淡が変化する．また，3 つの色はユーザが自由に設定することができる．開発者表示モードの詳細ビューは，その日・週に当該開発者が行ったコミット履歴がすべて表示される．これにより，各開発者がどの成果物を中心に編集しているかといった情報を把握することができる．このように，開発者表示モードと成果物表示モードを相互に分析することで，開発者と成果物の関連を調査することが可能となる．

3.2.4 フィルタリング機能

開発が進むにつれ，リポジトリに登録されているファイルは増加する．表示するデータが膨大な場合，着目したい箇所を探すのに時間が掛かってしまい，分析の障害となる．IZMI は，フィルタリング機能によって分析者が必要とする情報のみを表示することができる．フィルタは特定のファイル名 (拡張子)，ディレクトリ (パッケージ)，開発者，開発期間を指定して絞り込むことができる．また，イベントビューはスケールの調整が可能で，画面を縮小してより多くのデータを俯瞰したり，特定の箇所を拡大して精査したりすることが可能である．3ヶ月程度の開発プロジェクトを縮小表示した際のスクリーンショットを図 5 に示す．

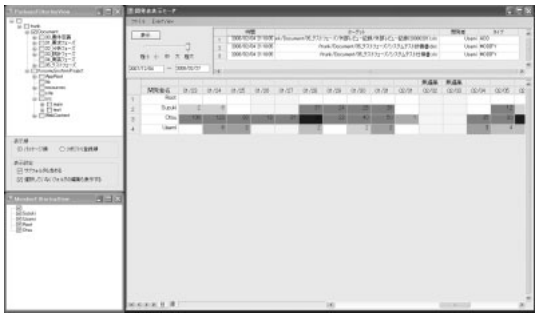

図 4　IZMI の画面 (開発者表示モード)

図 5　縮小表示させた IZMI の画面

4 適用実験

4.1 概要

本章では，IZMI の有効性を検証するために行った適用実験について述べる．実験では，小規模なソフトウェア開発プロジェクトを IZMI で可視化し，被験者が当該プロジェクトの情報を適切に把握することができるかどうかを検証した．

4.2 実験手順とデータセット

分析対象のプロジェクトは，7 人の学生によって約 1ヶ月間 (リリース後の保守を含む) 実施されたプロジェクトである．対象プロジェクトの概要を表 1 に示す．このプロジェクトは，開発が順調に進んだ期間と順調ではなかった期間があることがメンバのヒアリングによって事前にわかっている．本実験では被験者に対して次のような質問を与え，事実関係を適切に把握できるかどうかを検証する．

表 3 指摘事項の実例

指摘例	分析項目	記述内容
1	(1)	6/5(順調ではない)
	(2)	編集したファイル数が多く，ほとんど全ての編集を同一開発者が行っている
	(3)	仕様の変更のような大きな変化により，多くのファイルが編集されたのではないか
2	(1)	5/28〜6/2 がうまくいっていない
	(2)	この期間の開発が全く行われていない
	(3)	休暇中か，プログラミング以外の作業を行っていた
3	(1)	おおよそ順調だが，5/22〜5/27 および 6/3〜6/11 の期間は順調でない
	(2)	セルが赤色で示されていたり，トータルの編集回数が突出したファイルがある
	(3)	設計が不十分なまま開発が行われているため編集回数が増えたのではないか

分析項目は 4.2 節で述べた質問と対応

表 1 対象プロジェクト

プロジェクト	ゲーム開発
開発要員	7 人 (大学院生)
開発期間	27 日
総行数	9,578 行

表 2 被験者の実験時間と指摘事項件数

	指摘事項 (件)	実験時間 (分)
被験者 A	3	50
被験者 B	6	50
被験者 C	2	50
被験者 D	2	55
被験者 E	4	60
被験者 F	1	60
延べ	18	—

(1) 対象プロジェクトで開発がうまくいっていないと思われる期間はどこか．反対に開発が順調に進んでいると思われる期間はあるか．

(2) ツールのどの画面を見て判断をしたか．

(3) 対象プロジェクトにおいてどのような事象が発生していると考えられるか．

(1) と (2) はツールからわかる事実を，(3) は被験者が考えた推測をそれぞれ記述するように説明した．また，(1) から順に記述し，新たな発見があれば (1) に戻って再度記述するように指示した．回答は質問事項を印刷した分析シートを配付し，そこへ記入してもらった．指摘事項の正解判定は，対象プロジェクトの開発メンバへのヒアリングによって行った．

被験者は 6 人の大学院生で，対象プロジェクトに対して事前知識をもたない．ただし，いずれの被験者もソフトウェア開発に関する基礎的な知識を有している．また，被験者には実験前に IZMI の機能および操作方法と，分析対象プロジェクトが大学院生 7 人による小規模な開発であることを伝えた．実験は，1 時間を目安に分析を行ってもらい，新たな発見がなくなった場合は 1 時間未満であっても任意のタイミングで実験を終了してもよいこととした．また，1 時間が経過した後も分析を続ける意志があれば継続してもよいものとした．

表 4 開発メンバによる指摘事項の判定

判定	該当事項 (件)
大きく異なる	1
異なる	3
どちらともいえない	5
ほぼ一致している	6
一致している	3

4.3 実験結果と考察

被験者からは延べ 18 件の指摘を得た．表 2 に，実験に要した時間と指摘事項件数を，表 3 に指摘事項の実例を示す．表 3 より，被験者はプロジェクトの開発期間全般にわたり，特定のファイルや日付など幅広い観点から分析を行っていることがわかる．

4.3.1 開発者による事実確認

被験者が IZMI を用いて推測した 18 件の事象が事実であったかどうかの判定を当該プロジェクトの開発メンバに依頼した．判定結果を表 4 に示す．「ほぼ一致している」と「一致している」を正解とすると，被験者が推測した事象のうち 50%が事実と等しい判断を下せていたことがわかった．事実と判断された項目としては，編集回数が極端に多いファイルがあることから設計が不十分であったというものや，プロジェ

クト中盤で開発が全く行われていない期間に問題が発生していたことを推察したものがあった．事実と異なると判断された項目の多くは開発期間を見誤ったものであった．

4.3.2 考察

正答でないと判断された被験者の指摘は，納期時期の誤認に起因するものが多かった．対象プロジェクトはリリース後の保守工程を含んだものであるが，被験者の多くは IZMI で表示されている開発期間の最終日を納期として分析を行っていた．IZMI では，どの日にプロジェクトが納期を迎え，どの日から保守工程に入っているのかを明示する手段を有していない．そのため，開発期間に関する推察の多くが不正解となったと考えられる．より分析の精度を高めるためには，メーリングリスト等と照らし合わせて開発のコンテキストを把握する等，他のデータを併用する必要があろう．なお，対象プロジェクトは短期間ではあるものの，コミット数はおよそ 1,600 件にのぼり，実験と同様の分析を既存のリポジトリブラウザやリビジョングラフなどを用いて行った場合には多大な時間を要すると考えられる．また，対象プロジェクトは極めてインフォーマルな開発であったため，設計ドキュメントや障害管理に関するデータはほとんど残されていなかった．IZMI は版管理システムのログのみを入力とするため，このようなインフォーマルな開発プロジェクトにおいても分析を行うことができた．更に，被験者 F を除いた全員がプロジェクトの停滞期間に関する事象を正しく指摘しており，個人の能力に依らない分析ができていると考えられる．被験者 F は休日のコミット数に着目した分析を行っていたが，前述した納期時期の誤認により正しい判断を行うことができなかった．

正答率が 50%という結果は一概に良好とは言えないものの，1 時間に満たない分析実験で具体的な指摘を相当数得られたことは，IZMI が実プロジェクトの軽量な分析に有用であることを示唆していると言えよう．

5 まとめと今後の課題

本稿では，成果物の編集回数と開発者の編集頻度に着目したソフトウェア開発リポジトリ可視化ツール「IZMI」について述べた．また，実験によって IZMI が実際の開発リポジトリを可視化できること，また，プロジェクト外の分析者が IZMI を用いることで，当該プロジェクトの情報をある程度把握できることを確認した．しかし，納期の日付といったプロジェクトに関するコンテキスト情報は IZMI で把握することができず，分析の障壁となっていることも確認された．今後は，他の分析ツールとの連携や，開発者と成果物の関連がよりわかりやすく提示できるようなインターフェースの改良を行っていきたい．

謝辞 IZMI の実装を含め，本研究に対し長期にわたって尽力して頂いた奥村哲也氏に感謝申し上げます．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．また，本研究は日本学術振興会特別研究員奨励費の助成を受けたものである．

参 考 文 献

[1] D'Ambros, M. and Lanza, M.: Visual software evolution reconstruction, *J. Softw. Maint. Evol.*, Vol. 21(2009), pp. 217–232.

[2] D'Ambros, M., Lanza, M., and Lungu, M.: Visualizing Co-Change Information with the Evolution Radar, *IEEE Transactions on Software Engineering*, (2009), pp. 720–735.

[3] Hattori, L. and Lanza, M.: Mining the history of synchronous changes to refine code ownership, in *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories*, 2009, pp. 141–150.

[4] 中村建助, 矢口竜太郎: 2008 年情報化実態調査 プロジェクトの成功率は 31.1%, 日経コンピュータ 2008 年 12 月 1 日号 (2008), pp. 38–53.

[5] Sarma, A., Maccherone, L., Wagstrom, P. and Herbsleb, J.: Tesseract: Interactive visual exploration of socio-technical relationships in software development, in *ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, Washington, DC, USA, IEEE Computer Society, 2009, pp. 23–33.

[6] Tichelaar, S., Ducasse, S. and Demeyer, S.: FAMIX and XMI, *Reverse Engineering, Working Conference on*, Vol. 0(2000), p. 296.

[7] 大平雅雄, 横森励士, 阪井誠, 岩村聡, 小野英治, 新海平, 横川智教: ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム, 電子情報通信学会論文誌 *D-I*, Vol. J88-D-I, No. 2(2005), pp. 228–239.