

## 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

大平 雅雄<sup>†</sup>      横森 励士<sup>††</sup>      阪井 誠<sup>†††</sup>      岩村 聡<sup>††††</sup>  
小野 英治<sup>†††††</sup>      新海 平<sup>††††††</sup>      横川 智教<sup>†††††††</sup>

Designing a System for Real-Time Software Development Management

Masao OHIRA<sup>†</sup>, Reishi YOKOMORI<sup>††</sup>, Makoto SAKAI<sup>†††</sup>, Satoshi IWAMURA<sup>††††</sup>, Eiji ONO<sup>†††††</sup>, Taira SHINKAI<sup>††††††</sup>, and Tomoyuki YOKOGAWA<sup>†††††††</sup>

あらまし 本研究は、信頼性や生産性に課題の多いソフトウェア開発の分野において、科学的根拠に基づく開発手法である実証的ソフトウェア工学 (Empirical Software Engineering) の確立を目指す EASE (Empirical Approach to Software Engineering) プロジェクトの一環として行われたものである。本論文では、より実践的なプロセス改善の支援を目的として構築したシステム、EPM (Empirical Project Monitor) について述べる。EPM は、現在広く普及している開発支援フリーウェア (CVS, Mailman, GNATS 等) と連携することによって開発履歴データをリアルタイムに収集し、定量的データ分析を可能にするシステムである。EPM を利用したソフトウェア開発では、管理者及び開発者は常時、プロジェクトの進捗状況や作業状況を客観的に把握することが可能となる。その効用として、プロジェクトの問題点を迅速に発見し的確なプロジェクト管理が行いやすくなるため、効果的なプロセス改善活動の実施を期待することができる。システムの有用性を検証するために、試作した EPM を現行の EPM 開発プロジェクトに適用した結果、開発者に大きな作業負荷を与えることなく当該プロジェクトの状況を分析することができた。

キーワード ソフトウェアプロセス改善, 自動データ収集, 定量的データ分析, 実証的ソフトウェア工学

### 1. はじめに

ソフトウェア開発は複雑な作業の組み合わせであり、高品質なソフトウェアを効率的かつ安定的に開発することは容易ではない。たとえ成果物が明確な作業であっても、開発作業の実施方法は多種多様であり、ソフトウェアの品質や作業効率に大きな影響を与える。より安定した開発を実施する目的で開発標準を策定し、同一の作業方法を実施可能とするための環境整備が行われているものの、単に開発標準を定めただけでは開発者の能力や開発対象、さらにはスケジュールや予算等の制約条件によって、開発に要する期間や成果物の品質は大きく異なってしまう。そのため、開発状況を随時把握しプロジェクトを管理することが必須となる。

開発状況の把握に基づくプロジェクト管理では、トヨタ式カイゼン方法の一つである「見える化」が有名である [1]。見える化は一言で言えば、問題点が常に「見える」ようにしておく工夫のことであり、ソフト

<sup>†</sup> 奈良先端科学技術大学院大学情報科学研究科, 生駒市  
Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5, Takayama, Ikoma, Nara 630-0192, Japan  
<sup>††</sup> 大阪大学大学院情報科学研究科, 豊中市  
Graduate School of Information Science and Technology, Osaka University, 1-3, Machikaneyama, Toyonaka, Osaka 560-8531, Japan  
<sup>†††</sup> 株式会社 SRA 先端技術研究所, 東京都  
SRA Key Technology Laboratory, Inc. 3-12, Yotsuya, Shinjuku, Tokyo, 164-0004, Japan  
<sup>††††</sup> NTT ソフトウェア株式会社, 横浜市  
NTT Software Corporation, 223-1, Yamashita, Naka, Yokohama, Kanagawa 231-8554, Japan  
<sup>†††††</sup> 日立公共システムエンジニアリング株式会社, 東京都  
Hitachi Government & Public Corporation System Engineering, Ltd., 2-4-18, Toyo, Koto, Tokyo 135-8633, Japan  
<sup>††††††</sup> (株) 日立システムアンドサービス 研究開発センタ, 大阪市  
Hitachi Systems & Services, Ltd., 2-10-70, Nanbanaka, Naniwa, Osaka 556-0011, Japan  
<sup>†††††††</sup> 岡山県立大学情報工学部情報システム工学科, 総社市  
Okayama Prefectural University, 111, Kuboki, Soja, Okayama 719-1197, Japan

表 1 文書中心のプロジェクト管理の問題点  
Table 1 Issues on document-based management

適用限界	より詳細な情報を得ようとするため、適用の限界がある
情報遅延	開発状況の把握に必要となる情報を入手できるまでにある程度の時間を要する
情報操作	客観性に欠ける状況報告など、人為的な情報操作が入る可能性がある

ウェア開発にも適用することができる [2]。プロセス改善のプラクティスをまとめた CMM [3]/CMMI [4] においても、レベル 4 及び 5 では定量的なプロジェクト管理を目的としてソフトウェア開発中のデータが収集・分析され、ソフトウェアプロセスの改善に用いることが要求される [5]。これらの方法では、GQM [6] と同様、プロセス改善のための目標を定め、ゴールと関連するメトリクスデータを収集し改善活動に用いることが一般的である。しかしながら、収集されたデータがプロジェクト管理にとって必ずしも効果的なデータになり得るとは限らない [7]。特に、従来用いられていた開発データは報告書等の文書が中心であるため、表 1 に挙げるような問題点がある。

CMM/CMMI や文献 [7] 等で述べられているように、類似の開発プロセスが繰り返され、誰もが理解できる程度にプロセスが明確に定義され十分に成熟した組織を対象とした場合ならば、良いデータを収集できる可能性がある。しかし、このような組織であっても、収集できるデータの粒度に限界があり、プロジェクトの規模や方法論等によってはデータ収集及びプロセス改善の実施にかかる負荷が大きくなる [8]。また、報告書等の情報は、次工程への移行を決定する会議等において利用されるが、そのような文書が必要となる工程の区切りまでは情報が作成されず、利用できない場合が多く、管理者が開発状況を把握するまでにはある程度の時間的遅延が生じる。さらに、早く次の工程に進めることや人事評価を良くすることを意図した情報の加工や主観的な情報の混入等によって、不自然な運用が行われる可能性がある。

本研究は、文部科学省リーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」の中の EASE (Empirical Approach to Software Engineering) プロジェクト [9] の一環として、信頼性や生産性に課題の多いソフトウェア開発の分野において、科学的根拠に基づく開発手法である実証的ソフトウェア工学 (Empirical Software Engineering) の確立を目指すも

のである [10]。

これまで我々は、実践的なプロセス改善支援を目的として EPM (Empirical Project Monitor) [11]~[13] を構築してきた。EPM は、現在広く普及している開発支援フリーウェア (CVS, Mailman, GNATS 等) と連携することによって開発履歴データをリアルタイムに収集し、定量的データ分析を可能にするシステムである。文献 [11]~[13] では主に、プロセス改善に対する我々のアプローチ及び支援システム EPM の提案を行った。本論文では、システムの有用性を検証することを目的として EPM を試作し、さらに現行の EPM 開発プロジェクトに適用した結果とその考察を述べる。

以下、2 章では、先行研究とのアプローチの違いについて述べ、本研究の位置付けを明らかにする。3 章では、EPM の実装について詳述する。4 章では、本研究が想定している EPM の分析結果利用例を説明し、本システムの適用可能性を示す。5 章では、システムの評価結果とその考察について述べる。最後に 6 章において本論文をまとめる。

## 2. プロセス改善とプロジェクト管理

本章では、先行研究とのアプローチの違いを述べ本研究の位置付けを明らかにする。

### 2.1 本研究におけるプロセス改善

これまで、ソフトウェアプロセスの改善を支援するためのモデル (CMM [3], CMMI [4], IDEAL [14] 等) が提案され、企業において適用されてきている。これらのモデルは共通して、段階的かつ継続的なプロセス改善を実施するための一連のガイドラインを提供している。現状プロセスの分析によって問題点を抽出し、改善へ向けた明確な目標を設定した上でプロセス改善活動を実施するという、トップダウンの改善モデルを採用している点に特徴がある。しかしながら実際のソフトウェア開発の現場では、プロセス改善を実施しようとしても目標の設定自体が困難な場合が少なくなく、小規模な企業や組織的なプロセス改善を実施するゆとりのない企業においては特に問題になる。

図 1 にプロセス改善支援に対する我々の基本モデルを示す。プロセス改善のための実施方法や手順をトップダウンに規定するのではなく、ボトムアップにプロセス改善を支援しようとする点で他の既存モデルとは異なる。本モデルは、(1) ソフトウェア開発データの大規模収集、(2) 集約的データ分析、(3) プロセス改善のためのフィードバックというサイクリックなプロセス

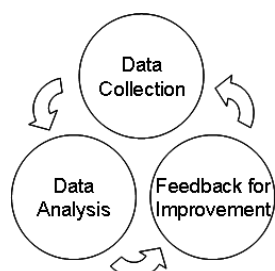


図 1 プロセス改善のための基本フェーズ  
Fig. 1 Basic phases for process improvement

改善活動を想定している。特に、定量的データを収集することに重点を置き、ソフトウェア開発プロジェクト及び組織の問題に沿って柔軟に改善活動を支援する。

本モデルを採用する理由は、進行中の開発プロジェクトの中で生じる、品質、コスト、納期等に関連する様々な問題を可能な限り迅速に発見し、的確にプロジェクト管理を行えるようにするためである。この意味において、プロセス改善支援に対する本研究の立場は、既存のものとは相反するのではなく補完可能なものであると考えることができる。粒度の整った一貫性の高いデータをあらかじめ収集しておくことで、データ分析を行いながら問題点を発見し、プロセス改善へつなげるためのフィードバックを随時提供していくことが可能となると考えている。

## 2.2 プロジェクト管理支援システムの設計

前章、表 1 に挙げたプロジェクト管理に関する諸問題を解決するための我々のアプローチについて述べる。

適用限界に対処するためには、開発者及び管理者に特別な作業負荷やコストを要さないデータ収集を行うことが必要である。次に、情報遅延の問題を解決するには、収集データを必要に応じた任意のタイミングで参照・分析できるようにすることが重要である。最後に、情報操作を防止するためには、人為的な情報が混入されにくい一貫性・客観性の高いデータを収集・計測する必要がある。

これらの要件を満たす支援システムを構築するために、本研究では表 2 に挙げる対処方法を採用する。まず、データの人為的な情報操作を防ぐことを目的として、構成管理、メーリングリスト管理、障害管理の各システムに残された開発履歴を開発データ収集源として利用することにした。これらは、ソフトウェア開発支援システムとして現在広く普及しており、多くの開発プロジェクトにおいて利用されている。日々の開発

表 2 支援システム構築のための本研究のアプローチ  
Table 2 Our approach

問題点	対処方法
適用限界	データを自動的に収集する機能の実装
情報遅延	リアルタイムにデータ分析する機能の実装
情報操作	開発作業の中で生成される開発履歴データの利用

作業の中でこれらのシステムに蓄積される履歴データを利用することによって、人為的に加工されにくい客観性の高いデータを収集することができるだけでなく、開発規模や開発形態に依存しない支援システムを構築することができると考えた。

このような開発履歴データを自動収集する機能を支援システムに実装することによって、開発者や管理者にデータ収集のための特別な作業負荷を与えずにデータを収集することが可能となる。さらに、開発履歴データをリアルタイムに分析する機能を実装することによって、開発状況を常に把握し問題が生じた場合に迅速に適切な対処を行うことが可能になると考える。

## 2.3 関連研究

関連研究として、開発データの自動収集という側面では、開発者のキーストロークや視線移動等のデータを収集する Ginger2 [15] がある。Ginger2 は、開発者の作業や行動をきめ細かな粒度で記録・分析することで、ソフトウェア開発に関する諸特性を解明するために開発されたシステムである。また、Hackystat [16] は、開発支援環境にセンサーとなるものを付加することで開発者の作業履歴を記録するシステムである。Hackystat の開発目的は、ソフトウェア開発における内因特性（サイズや時間等）と外因特性（品質や信頼性等）との関係を見出すことにある。これらの研究では、いわゆる実験室実験（統制実験）によって、ソフトウェア開発に関連する様々な仮説を検証することが主たる目的である。本研究の目的は、開発データの収集、分析、フィードバックというサイクリックなプロセスを通じたプロセス改善を支援することである。また、実際の開発現場でシステムを適用し、その効用を実証しようとする点が研究の方法として異なるものである。

開発履歴データの分析の側面としては、開発作業状況を可視化する Augur [17] や ROSE [18] 等がある。これら以外にも、構成管理や障害管理システムに蓄積された履歴データを分析する研究が近年盛んに行われている [19], [20]。これらの研究では、いずれか 1 種類の開発支援システムに蓄積された履歴データを分析し、ソフトウェアの変更理由の同定、分散環境下での情報

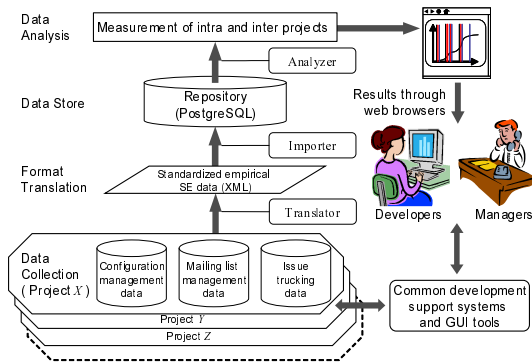


図 2 Empirical Project Monitor の概要  
Fig. 2 Overview of Empirical Project Monitor

遅延の原因解明，潜在的変更箇所や不完全な変更箇所の検出等の支援を目的としている．本研究は，3種類の開発支援システムから履歴データを収集し，それらを組み合わせた分析結果を提供することによってリアルタイムでのプロジェクト管理を支援する．複種類の履歴データを同時に分析利用する研究はまだほとんど見当たらず，本研究の新規性の高い部分である．

### 3. Empirical Project Monitor

本章では，2.2節の設計指針に従って実装を行った Empirical Project Monitor (EPM) を紹介する．

#### 3.1 EPM の概要

EPM は，リアルタイムでのプロジェクト管理を目的とした開発データの自動収集・分析システムである．EPM は，ソフトウェア開発において一般的に利用されている開発支援システムから開発履歴データを収集し，プロジェクト管理を行うために有益な分析結果をユーザに提供する．EPM は，実際の開発現場では困難が付きまとう一貫性のある定量的データ収集と収集データの分析利用を容易にする．

図 2 は，EPM の概要図である．EPM は，データ収集，データ変換，データ蓄積，データ分析・可視化の 4 つの基本機能から成る．以下では，EPM が処理するデータの流れに沿って各機能を説明する．

データ収集部では，構成管理，メーリングリスト管理，障害管理など，ソフトウェア開発において広く普及利用されている開発支援システムからデータを収集する．システムに保存される履歴データは，日常の開発作業の過程で蓄積されるものである．したがって，管理者・開発者がデータ収集のために特別な作業を行う必要はない．また，すべてフリーソフトとして利用

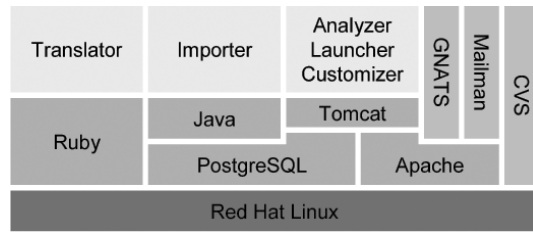


図 3 EPM のアーキテクチャ  
Fig. 3 The Architecture of EPM

表 3 EPM のサブシステム  
Table 3 Subsystems of EPM

サブシステム	説明
Translator	収集データを標準エンピリカルデータフォーマットと呼ぶ XML 形式のデータに変換
Importer	フォーマット変換したデータを PostgreSQL データベースに格納
Analyzer	格納されたプロセスデータをユーザ (管理者・開発者) の要求に応じて分析し結果を可視化
Launcher	Translator・Importer・Analyzer を Web 経由で実行
Customizer	Translator・Importer・Analyzer で使用する各種設定ファイルの作成・更新

可能なシステムであるため，EPM 導入にかかるコストは非常に低い．

データ変換部では，Ruby で記述されたスクリプトによって，収集データを標準エンピリカルデータフォーマットと呼ぶ XML 形式のデータに変換する．標準エンピリカルデータフォーマットは，各コンポーネントとのインターフェースとして公開されている．XML 形式のデータに変換する理由は，(1) 既存の開発環境で利用されている同種の様々なシステムへ対応するために入力データ形式を統一する必要があること，(2) 分析対象データの入力データ形式を統一することで EPM 上での分析環境の実現が容易になることの 2 点である．

データ蓄積部では，標準エンピリカルデータフォーマットに変換されたデータを PostgreSQL データベースに格納する．例えば，構成管理システム CVS からは，ソースコードの修正，チェックイン，チェックアウト等のイベント情報を取得可能である．これらのイベントには，CVS リポジトリ中のコンポーネントのサイズ，更新時期，バージョン等の遷移情報が含まれており，それらの情報もデータベースに格納する．また，メーリングリスト管理及び障害管理システムからは，メールのヘッダや障害報告を元に，投稿時間，件名，差出人等の情報を格納する．同時に，話題ごとに

表 4 カスタマイズ機能  
Table 4 Customization of EPM

機能	カスタマイズ内容	
環境設定	環境変数	EPM のインストール場所及びデータ保存場所の指定が可能．環境変数は Tomcat の引数経由でサブレットからも参照
	環境設定ファイル	プロジェクト毎の開発環境，履歴データ位置，出力位置を XML 形式で記述指定可能
機能追加	標準エンピリカルデータ	XML 形式を用いて中間ファイルのデータ構造を規定しており，EXCEL 等の外部システムによる分析や他システムからのデータ取り込みが可能
	リポジトリ (DB)	データの間接参照により可変データ構造を実現
	Translator	Ruby スクリプトの改造により各種開発システムに対応可能
	Importer	サブクラスの追加により各種開発システムに対応可能
	Analyzer	SQL 文の入力によりユーザ独自の分析 (カスタマイズ分析) ができ再実行も可能
分析機能	プロパティファイル	ユーザが過去に実行したデータ分析を保存しており呼び出しによる再実行が可能．DB 構造に依存する部分を独立させているため属性名等の変更にも対応可能
	カスタマイズ分析履歴	記録された分析履歴は，標準分析機能の拡張設計に利用される

集計した一覧表も作成する．

データ分析・可視化部では，格納されたプロセスデータをユーザ (管理者・開発者) の要求に応じて分析し，結果を提示する．データは随時収集・蓄積・分析され，プロジェクトを管理するために有益な分析結果をリアルタイムで得ることができる．分析は Java サブレットにより行われ，結果をグラフや表としてウェブブラウザから閲覧することができ，プロジェクトの分析結果の共有を促進する．分析結果共有の簡便化は，分析結果を理解するために行う議論の場で特に有効である．

このように，EPM は定量的な開発データを低いコストで収集し，開発プロジェクトのリアルタイムでの管理を支援する．基本的な 3 種類のシステムから人為的操作のされにくいデータを収集することによって，一貫性のある分析結果を提供することができる．

### 3.2 基本アーキテクチャと既存環境への適応

図 3 に EPM の基本アーキテクチャを，表 3 に EPM の各サブシステムの説明を示す．EPM は Red Hat Linux 上での動作を想定しており，Red Hat Linux 上にインストールした Ruby，PostgreSQL，Java，Apache，Tomcat を用いて，EPM の 5 つのサブシステム (Translator，Importer，Analyzer，Launcher，Customizer) を実現している．

図 3 中では，分析用データ収集源として CVS，Mailman，GNATS<sup>(注1)</sup> の利用を想定した場合を示しているが，Majordomo，fml，Bugzilla<sup>(注2)</sup> のデータも同様に収集・分析することが可能である．また，Java のライブラリとして JFreeChart，Log4J 及び PostgreSQL 用の JDBC ドライバを，Ruby には REXML，uconv

(注1)：利便性を考慮し，Gnatsweb との併用を推奨

(注2)：Bugzilla を利用する場合は，MySQL 環境が必要

表 5 開発者の振舞いと問題となる行為  
Table 5 Developer's behavior and problems

開発者の振舞い	問題となる行為
要求仕様確認作業	数少ない情報交換等
プロダクト提供作業	プロダクト提供が少ない・遅い，CVS 参照がない，CVS 更新連絡がない，プロダクトに関する議論が少ない等
障害報告及び対策作業	不具合報告が多い，一次対応が遅い，不具合対応の更新・完了が多い，不具合対応に伴う仕様変更が多い等

のライブラリを利用している．

EPM の導入にかかるコストをできるだけ減らすため，また，各組織の既存環境資産を有効に活用できるようにするために，表 4 に挙げるようなカスタマイズ機能を実装した．これらの機能によって既存環境への適応性の高いシステム構造となっている．

## 4. EPM の分析結果利用例

本章では，本研究が想定している EPM の分析結果利用例を説明し，本システムの適用可能性を示す．

### 4.1 単一プロジェクトを対象とした可視化

以下では，我々の研究プロジェクト (EASE プロジェクト [9]) における EPM 開発を例として利用し，EPM が可視化する 5 種類の標準分析結果を説明する．

5 種類の分析方法を標準とした理由は，プロジェクト管理に必要なとなる，開発者の振舞いに関する情報を提供するためである．ここでいう開発者の振舞いとは，大きく分けて次の 3 つの作業の中で行われるものから成る．(1) 要求仕様確認作業，(2) プロダクト提供作業，(3) 障害報告及び対策作業である．

これらの作業における問題は，プロダクトの品質，コスト，納期 (QCD) に大きな影響を与えるため，開発者の振舞いは常に管理者及び開発者自らがチェック

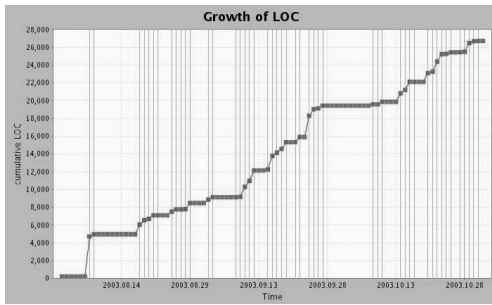


図 4 ソースコードの規模推移  
Fig. 4 Transition of LOC

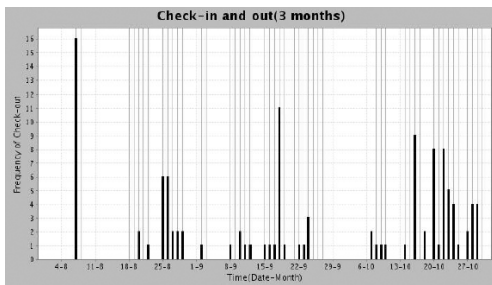


図 5 チェックインとチェックアウトとの関連  
Fig. 5 Check-in and Checkout

できることが望ましい。表 5 に、プロダクトの QCD に影響を与える開発者の振舞いを挙げる。5 種類の標準分析機能は、開発者の振舞いをリアルタイムで把握し、表 5 で挙げた問題となる行為を早期発見する上で必須の機能として実装された。

ソースコードの規模推移：図 4 は、ソースコード規模の推移（図中、黒の折線）を示すものである。図中、灰色の垂線は開発者が CVS リポジトリを更新した（チェックインした）時期を表しており、リポジトリ中のファイルに対する追加/修正/削除等の操作が行われたことを示している。グラフの規模推移と更新時期との関係から、何らかの原因で開発が停滞している事を確認できるだけでなく、例えば、順調な規模増加と頻繁な更新が同時に起きていれば、開発が活発に行われていることを知ることができる。リアルタイムでの開発状況の把握は、プロジェクト管理者だけでなく、定量的データを用いて開発状況の説明を行えることから、開発者にとっても有益である。

チェックインとチェックアウトの関連：図 5 は、チェックインの時期（図中、灰色の垂線）とチェックアウト回数（図中、黒の垂線）との関係を示している。CVS リポジトリに開発者がチェックインした場合、通常他の

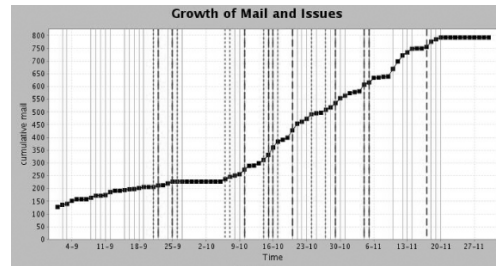


図 6 メール投稿数の推移と障害発生・解決時期との関連  
Fig. 6 Mails and bugs

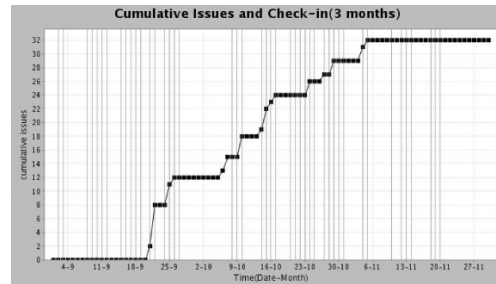


図 7 チェックインと障害件数との関係  
Fig. 7 Check-in and bugs

開発者はローカルにある自らのファイルを更新するためにチェックアウトの操作を行う。チェックイン後にチェックアウト数が少ない場合は、CVS 中のファイルが更新されているにもかかわらず他の開発者がそれを参照していないことになる。変更が局所的なものであり全体への影響が少ない場合と、開発者のコミュニケーションがうまくいかず変更が周知されていない場合等が考えられるが、後者の場合は改善が必要となる。

メールの投稿数の推移とチェックイン/障害発生/障害解決時期との関連：図 6 は、開発者用のメーリングリストを使用して投稿されたメールの累積（図中、太い黒の折線）、障害発生時期（図中、細かい点線）、障害解決時期（図中、粗い点線）、チェックインの時期（図中、灰色の垂線）との関係を表したものである。このグラフから、障害発生/解決時期と開発者間のコミュニケーションの関係を知ることができる。通常、障害報告がなされた場合、開発者らはメーリングリストを通じて対応策等について議論を行う。グラフ中に、障害が数多く報告されているにもかかわらず、開発者が議論を行っていない（メールの投稿数が少ない）状況が存在すれば、開発者間のコミュニケーションに問題がある可能性が高い。開発者間のコミュニケーションの問題はソフトウェア生産性や信頼性を減少させる原因

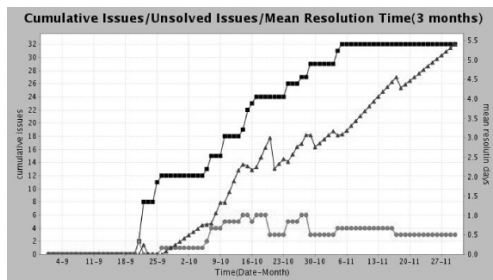


図 8 累積・未解決障害件数と平均障害滞留時間との関連  
Fig. 8 Resolved and unresolved bugs

となる [19], [21]. また, このグラフ中に表示される障害発生時期, (バグフィックスを行った) チェックイン時期, 障害解決時期の対応関係から, プロジェクトの障害対応の状況も確認することができる.

チェックインと障害件数との関係: 図 7 は障害報告の累積 (図中, 黒の折線グラフ) とチェックインの時期 (図中, 灰色の垂線) との関係を示している. 障害報告がなされた後, 障害修正結果を反映するために開発者がチェックインするケースが多いが, このグラフはバージョン毎の障害対応状況の確認に役立つ.

累積・未解決障害件数と平均障害滞留時間との関係: 図 8 は, 障害報告の累積 (図中, 上の折線), 未解決障害件数の推移 (図中, 下の折線), 平均障害滞留時間 (図中, 真中の折線) との関係を示している. このグラフは, 現在プロジェクトが抱えている障害の多寡や障害の解決状況を表しており, テスト段階において深刻な状況が発生しているか否かをプロジェクト管理者が把握するのに役立つ. また, プロジェクトが抱えている障害状況が一目でわかるため, 開発者がバグを減らすための動機付けとしても利用することができる.

#### 4.2 複数プロジェクトを対象とした可視化

EPM は, 前節で挙げた単一プロジェクトに対する分析のみならず, 複数のプロジェクトを対象とした比較分析を行うことができる. 過去の類似プロジェクトを比較することで, 進捗見積りや異常値の早期検出など, 主に管理者にとって有益な分析を行える.

複数プロジェクト間の比較: 図 9 は, 二つのプロジェクト間のソースコードの規模の変遷を比較したものである (上部の折れ線: SPARS プロジェクト [22], 下部の折れ線: EASE プロジェクト). 仮にこの二つのプロジェクトが同様な開発形態や開発規模をとるのであれば, 後発のプロジェクトの進捗状況や遅延の有無を大まかに見積もることができる. 例えば, SPARS プ

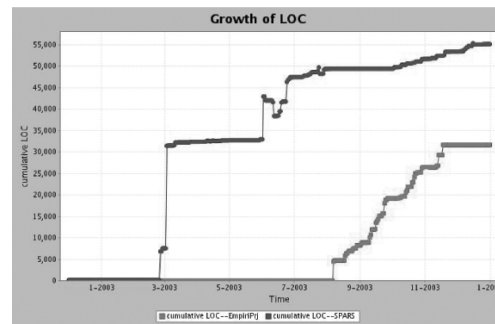


図 9 二つのプロジェクトの規模推移の比較  
Fig. 9 Comparison of LOC

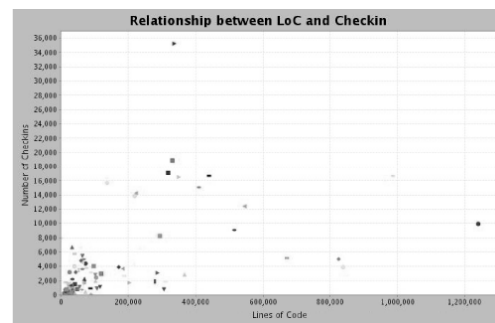


図 10 多数の OSS プロジェクトの散布図  
Fig. 10 Distribution map of 100 OSS projects

プロジェクトはメジャーバージョンをリリースするために急激に進化する 2 つのフェーズが存在することが見て取れる. EASE プロジェクトはちょうど最初のメジャーバージョンをリリースしたところである. 管理者は近い将来の EASE プロジェクトの進捗をある程度予想することができる (レビューやテストのためにソースコード規模はしばらく増加しない等).

複数プロジェクトの分布: 図 10 は, 複数プロジェクト間の比較分析を目的としてオープンソースソフトウェア (OSS) 開発プロジェクト 100 件分のデータを SourceForge.net から収集し<sup>(注3)</sup>, EPM を用いて分析した散布図の例である. 図 10 は, ソースコード規模 (X 軸) とチェックインの数 (Y 軸) の関係を表したものである. 図 10 から, 同程度の規模を持つプロジェクトであっても極端にチェックアウト数が多い/少ないプロジェクトや, 反対に, 同程度のチェックアウト数であっても極端に規模の大きい/小さいプロジェクトを見て取ることができる. このような散布図を利用す

(注3): SourceForge.net の active project list の上位 300 プロジェクトからデータ収集可能であったプロジェクト 100 件分の CVS データ.

表 6 EPM 開発プロジェクトの概要と評価環境  
Table 6 Evaluation of EPM development project

EPM 開発プロジェクトの概要	
対象プロジェクト	EPM 開発グループ (試作版完成後、次期版の EPM を開発)
利用言語	Ruby, Java
開発者数	企業派遣の開発者 4 名 (それぞれは異なる企業に所属)
分析対象システム	CVS, Mailman, GNATS
EPM 導入期間	EPM 試作版完成後から約 3ヶ月 (2003 年 11 月中旬から 2004 年 1 月)
評価に利用した計算機環境	
EPM 導入 PC	CPU Pentium4 2.53GHZ, Memory 1GB, HDD 200GB
LAN 環境	100Mbps で接続されたクライアント PC から分析結果を表示

ることで、年間数百あるいは数千規模でプロジェクトを運営管理する必要のある企業では、異常値の早期発見が容易になると考えられる。

## 5. 評価と考察

本章では、試作した EPM を現行の EPM 開発プロジェクトに適用した結果を考察し、システムの有用性を検証する。

### 5.1 EPM 適用環境と評価環境

表 6 に、EPM 開発プロジェクトの概要とシステムの評価に利用した計算機環境を示す。今回のシステムの有用性評価は、エスノグラフィの手法 [23] を用いて得られたデータ (開発プロセスの詳細な観察結果と、開発者からの意見及び感想) に基づくものである。

著者ら (4 名の開発者と 3 名の研究者) は、同じ職場で共同研究をしており、研究者はプロジェクトの進捗状況や各開発者の作業状況を常に観察することができた。週一回行われる開発進捗ミーティングにも研究者が出席することが許可され、随時、質問や意見を述べることができ、さらに工程管理表やその他ドキュメント類の閲覧も可能であった。

このような物理環境の中で、EPM 開発プロジェクトを細部に渡って観察・把握した研究者が、システムの有用性評価を行った。開発者には、次節で挙げる 5 つの評価項目に沿って、(定量的測定結果が利用できる場合はそれらを示し) インタビューを行い意見・感想を求めた。

EPM の有用性について定性的な評価を行った部分については、研究者らの観察結果と、開発者らへのインタビュー結果を併せて行ったものである。5 つの評価項目に沿ったインタビューだけではなく、(同じ職場

であることから機会の多かった) インフォーマルな議論の中から得られた意見や感想による部分も大きい。

### 5.2 評価方法

本論文ではまず、2.2 節で述べたシステム設計時における以下の 3 つの要件が満たされていたかどうかを評価する。

- 低い作業負荷とコストでのデータ収集
- 任意のタイミングでのデータ分析
- 人為的な情報操作の防止

これらの評価では、定量的な測定・分析結果を開発者に示し、研究者の評価結果についての妥当性に関して確認した。特に、人為的な情報操作の防止に関しては、履歴データの分析結果だけでは不十分と考えられたため、人為的情報操作の可能性について開発者を交えて議論し考察を行った。

次に、EPM の適用により得られる効用を以下の 2 点について評価し考察する。

- プロジェクト管理への効用
- プロセス改善への効用

これらの評価では、開発者及び開発状況の詳細な観察による定性的な評価結果と、開発者へのインタビュー結果を併せて考察し EPM 適用の有用性を判断した。

### 5.3 EPM 適用結果の評価と考察

今回得られた EPM 適用の有用性評価結果を表 7 にまとめる。以下では、各評価項目の評価結果とその考察について述べる。

低い作業負荷とコストでのデータ収集：EPM 及び関連するオープンソースソフトウェアのインストールには、約 2 人日要することがわかった。それらのシステムの管理・運用技術を開発者全員が習得する (システムの利用に慣れるまで) には、約 1 週間が必要であった。また、EPM 導入時には CVS 未経験者がいたが、類似のシステムの知識があったので 1~2 時間程度の学習で操作できるようになった。EPM の導入後は、(個々のプロジェクトが CVS 等のシステムを利用していることが前提ではあるが) ネットワーク経由で組織内のすべての開発プロジェクトを管理可能である。また、新規プロジェクトの追加設定等は GUI ベースの管理ツール (Customizer) を利用することができる。これらを考慮すると、EPM 導入に際しての作業負荷は多少必要ではあるものの、特に運用面においては大きな作業負荷が生じないことを確認できた。コスト面では、ハードウェアとして EPM をインストールする PC が必要な他は、フリーソフトを利用するため特別



表 7 EPM 適用の有用性評価

Table 7 Evaluation of EPM

評価軸	評価項目	評価結果の概要
システム設計時の要件	低い作業負荷とコストでのデータ収集	導入時の訓練等で多少の作業負荷は必要だが、運用面・コスト面では要件を満たせることを確認
	任意のタイミングでのデータ分析	妥当なリアルタイム性を確認
	人為的な情報操作の防止	情報操作の可能性は低いと考えられるが、さらに検証が必要
システム適用の効用	プロジェクト管理への効用	プロジェクトの状況理解・問題発見が容易に行える
	プロセス改善への効用	共同作業のための規約が徐々に作成された

表 8 リアルタイム性評価結果

Table 8 Result of Real-time performance

Translator によるデータフォーマット変換	約 3 秒
Importer によるデータベースへのデータ格納	約 74 秒
Analyzer による分析結果表示	平均約 4 秒

なコストはかからない。

データ分析のリアルタイム性：

表 8 は、EPM 開発プロジェクト 1 件分（ソース行数：約 32,000 行，メール総数：約 800 通，障害報告：32 回）の履歴データから，Translator による標準エンピリカルデータフォーマット変換，Importer による PostgreSQL データベースへのデータ格納にそれぞれ要した時間，Analyzer による（4.1 節の 5 種類の）分析結果表示に要した平均時間をまとめたものである。

開発の進捗状況を数分単位で管理することは考えにくく，通常は日単位でデータを格納すれば十分であることから，表 8 の結果は，履歴データの変換・格納時間及び分析結果の表示時間ともにリアルタイム性を示すものであるといえる。

参考として，今回の検証では OSS 開発プロジェクト 100 件分の CVS データも合わせてデータベースに格納した。それらを含めたデータ変換・格納に要した時間は約 7 時間であった。この結果は，市販の標準的な PC を利用したことを考慮すると妥当な値であると考えられるが，今回の実装ではプロジェクト毎にすべてのデータを一度にデータベースへ格納する方式を採っていたことも大きな原因である。現在，日々の作業更新分（差分）を適宜格納していく方式を実装中であり，多数かつ大規模なプロジェクト管理を行う場合のリアルタイム性確保に対応する予定である。

また，図 10 のプロジェクト 100 件分の散布図を表示するのに要した時間は約 46 秒であった。分析結果の表示時間については，データベースのチューニングやストアドプロシージャの定義によって，ある程度の処理時間の改善は可能であるが，今回 EPM を導入し

た PC 程度の性能では限界があることも事実である。EPM の適用規模に対してどの程度のハードウェア性能が必要かについては今後さらに検討が必要である。

人為的情報操作の防止：すべての履歴データを分析した結果，EPM 開発プロジェクトでは人為的な情報操作は発見されなかった。しかし，この結果から EPM が人為的情報操作を防止可能なシステムであると判断することは困難である。ここでは，EPM を利用したソフトウェア開発において，個々の開発者が意図的に情報操作を行えるかどうかについて考察する。

EPM は，開発者が日常頻繁に使用する開発支援システムに蓄積される履歴データを収集する。各工程終了時に開発者個人に手作業で入力させる従来のシステムとは異なり，蓄積された履歴データを EPM に取り込む際には人為的な情報操作は行えない。従って，各工程終了時に情報操作を行い，開発の進捗をより良く見せようとするなどは基本的に不可能である。

一方，悪意のある開発者が開発メンバの中に存在するのであれば，開発支援システム使用中の意図的な情報操作（ソースコードの水増し，不適切な障害報告など）を完全には防ぐことはできない。しかしながら，EPM を利用したソフトウェア開発では，個々人の開発者が意図的に不正を行おうとする動機を抑制する効用があると考えられる。この理由には大きく二つを挙げることができる。

一つ目は，各工程終了時に開発者個人が行っていた従来の手作業のデータ入力システムとは異なり，多人数参加型の開発支援システムを（履歴データ収集源として）利用していることが挙げられる。一部の上司が作業状況をチェックするのではなく，開発メンバ全員が各開発工程の中でソースコードや障害報告をリアルタイムで閲覧する機会が多数あるため，不正が発覚しやすい。また，不正なコードの混入や不適切な障害報告を行うと，結果的に他の開発者及びプロジェクト全体の作業を阻害することになるため，故意に不正を働きにくい。

二つ目は、開発者が日常的に頻繁に使用する開発支援システムを利用している点である。短期的な情報操作(生産性を一時的に高く見せることなど)は比較的容易であるが、EPMの分析機能によりリアルタイムでソースコードの規模推移やファイルの更新回数・時期をチェックできるため、中長期的に見れば短期的な情報操作を行った時期のデータは不自然であることが一目瞭然となる(不正が発覚しやすい)。従って、一度不正を行うと、つじつまを合わせるために日々コメントを増やすなどを継続する必要が生じる(手間がかかりすぎる)。

これらの理由から、EPMを利用したソフトウェア開発では、人為的な情報操作を行うメリットが開発者個人には少ないと考えられることから、情報操作の可能性は低いものと思われるが、今後様々なプロジェクトにEPMを適用し検証していく必要がある。

プロジェクト管理への効用: プロジェクトの状態は、グラフの特徴的な部分について確認することで容易に把握することができる。例えば、EPM開発プロジェクトの場合、図4中の中央やや右の部分では行数の成長が止まり水平になっている。この部分は担当者全員が出張し開発作業を行っていなかった期間である。実際の開発現場においても、他のプロジェクトのトラブルによって開発者の作業が中断するなど、同様の状況が生じる可能性がある。従来の方法では管理者が注視していない限り早くて週単位、場合によっては月単位でないと把握できない事象である。リアルタイムにデータを収集し可視化を行うことで、より容易に状況把握が可能になったといえる。

また、表示されるグラフからプロジェクトに問題が存在することも明らかになる。例えば、図8の中央の折れ線は、1件あたりの障害に対して、障害の報告から解決までに要する時間を示しているが、日々増加していることがわかる。この原因は、次バージョンで解決方法を決定するとして障害を放置していたためである。現在は件数が少ないため大きな問題ではないが、他に障害が多数発生した場合に混乱が生じる恐れがある。このような問題をグラフから容易に見ることが可能になったことから、プロジェクトを適切に管理するための対策が講じやすくなることがわかった。

このような議論を行うためのデータ集計・分析・可視化は、従来の手作業によるデータ入力支援システムの利用のみでは容易な作業ではない。EPMは、プロジェクトの状況理解・問題発見を行うための議論の場

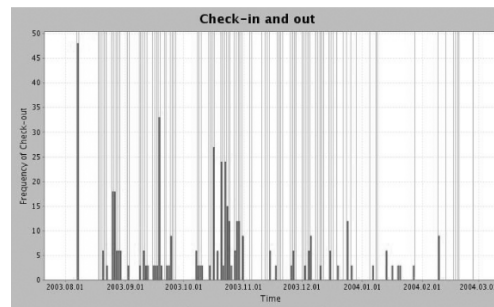


図 11 CVS の運用方法の変更  
Fig. 11 Change of CVS operation

で、客観的な情報を提供する有益なシステムであると考えることができる。

プロセス改善への効用: 複数の企業の開発者によって開発が行われたこと、EPM導入期間が約3ヶ月であったこと、EPM開発プロジェクトは研究的側面が強い(試行錯誤の過程が長く仕様も明確に決まりにくい)こと等から、アジャイルソフトウェア開発を行った。このため、従来のウォーターフォール開発プロセスとの比較がしづらく、今回の評価では開発プロセスの大幅な改善と結論付けるような事象を確認することはできなかった。

しかしながら、可視化された開発データに基づいて、EPM運用のために必要なルールを作成するなど、共同作業を円滑に進めるための規約が、徐々に開発チーム内に構築されていった。例えば、可視化結果の背景に示されるチェックイン縦線が予想以上に多く、各バージョンと各種データとの関係の判別が困難になることがあった(図11のグラフ左から3分の2部分)。日々チェックインを行う開発者がいたため縦線が多数表示されたことが原因である。これはCVSの運用上の問題であり、チェックインのルールを予め決めておく必要があることが認識された。対処方法として、バックアップ用とリリース用にCVSを分けておき、更新と参照の関係を見るにはリリースのCVSを、日々の規模の変化などを見るにはバックアップのCVSを参照するようになった<sup>(注4)</sup>。この結果、バージョンごとの作業の状況把握(図11のグラフ右3分の1部分)や日々の作業の把握が可能になった。

今回の例は小さな改善ではあるが、定量的データ収集に基づくプロセス改善では、このような積み重ねを行うことが問題点の「見える化」につながる。EPM

(注4): 2004年1月中旬以降、CVSの運用方法を変更。

を適用することによって、開発者自らが問題点を視えるように工夫を行なった点については、今後のプロセス改善へ向けて価値があるものであったと考えている。

具体的なプロセス改善の効用については、現在、約10社の企業が実プロジェクトにおいてEPMを導入中であり、これらのEPM適用結果が得られ次第、検証を行う予定である。

## 6. ま と め

本研究は、信頼性や生産性に課題の多いソフトウェア開発の分野において、科学的根拠に基づく開発手法である実証的ソフトウェア工学 (Empirical Software Engineering) の確立を目指し、EASE (Empirical Approach to Software Engineering) プロジェクトの一端として、産学官連携の共同研究として行われたものである。

本論文では、実践的なプロセス改善支援を目的として構築した Empirical Project Monitor (EPM) について述べた。EPM を利用したソフトウェア開発では、管理者及び開発者は常時、プロジェクトの進捗状況や作業状況を客観的に把握することが可能になる。その効用として、プロジェクトの問題点を的確に発見しやすくなるため、より効果的なプロセス改善活動の実施を期待できる。試作した EPM を現行の EPM 開発プロジェクトに適用しシステムの有効性を検証した結果、開発者に大きな作業負荷を与えることなく当該プロジェクトの状況を分析できることを確認した。

今後の課題としては、EPM 適用企業からの結果を分析し、EPM のプロジェクト管理効果およびプロセス改善への効果をより詳細に明らかにする必要がある。また、プロセス改善に有効な様々なシステムを評価検討し、EPM の分析・可視化用コンポーネントとして実装し提供していく予定である。

謝辞 本研究の一部は文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた。また、本研究の一部は文部科学省独創的革新技術開発研究提案公募制度による研究助成を受けて行われた。

本研究を遂行するにあたり多くの御協力を頂いた EASE プロジェクト関係者諸氏に感謝します。特に、大阪大学の井上克郎教授、奈良先端科学技術大学院大学の松本健一教授、同客員教授の鶴保征城氏に深く感謝致します。また、本論文をまとめるにあたり貴重なコメントを頂いた査読者の方々に感謝します。

## 文 献

- [1] 若松義人, 近藤哲夫, “トヨタ式改善力,” ダイヤモンド社, 東京, 2003.
- [2] 川又英紀, “IT 業界もトヨタ流企業改革から始めてみては?,” <http://itpro.nikkeibp.co.jp/free/ITPro/OPINION/20040123/138653/>, 2004.
- [3] M. C. Paulk, B. Curtis, M. B. Chrissis and C. V. Weber, “Capability maturity model (version 1.1),” *IEEE Software*, vol.10, no.4, pp.18–27, 1993.
- [4] “CMMI product team: capability maturity model integration (CMMI) version 1.1,” *CMMI for systems engineering and software engineering, continuous representation, (CMMI-SE/SW, v1.1, continuous)*, CMU/SEI-2002-TR-001, 2002.
- [5] W. S. Humphrey, “Process maturity model,” in *Encyclopedia of Software Engineering*, ed. J. J. Marciniak, vol.2, pp.851–860, John Wiley and Sons, New York, 1994.
- [6] V. Basili, “Applying the goal question metric paradigm in the experience factory,” Presented at the 10th Annual Conf. of Software Metrics and Quality Assurance in Industry, Amsterdam, 1993.
- [7] L. Briand, C. Differding and D. Rombach, “Practical guidelines for measurement-based process improvement,” Technical Report ISERN-96-05, Department of Computer Science, University of Kaiserslautern, Germany, 1996.
- [8] J. G. Brodman and D. L. Johnson, “What small businesses and small organizations say about CMM,” *Proc. of the 16th intl. Conf. on Software Engineering (ICSE’01)*, pp.331–340, Tronto, Canada, 1994.
- [9] The EASE (Empirical Approach to Software Engineering) project, <http://www.empirical.jp/>
- [10] 井上克郎, 松本健一, 鶴保征城, 鳥居宏次, “実証的ソフトウェア工学環境への取り組み,” *情報処理*, vol.45, no.7, pp.722–728, 2004.
- [11] 阪井誠, 大平雅雄, 横森励士, 松本健一, 井上克郎, 鳥居宏次, “EPM: 導入の容易な開発データ自動収集・分析支援システム—お手軽にリアルタイムの生データ—,” *ソフトウェアシンポジウム 2004 論文集*, pp.119–127, 2004.
- [12] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue and K. Torii, “Empirical project monitor: A tool for mining multiple project data,” *Proc. of 1st intl. Workshop on Mining Software Repositories (MSR2004)*, pp.42–46, Scotland, UK, 2004.
- [13] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue and K. Torii, “Empirical project monitor: automatic data collection and analysis toward software process improvement,” *第1回ディペンダブルソフトウェアワークショップ (DSW2004) 論文集*, pp.141–150, 2004.
- [14] J. Gremba and C. Myers, “The IDEAL model: a practical guide for improvement,” Appeared in the *Software Engineering Institute (SEI) Publication, Bridge*, Issue 3, 1997.

- [15] K. Torii, K. Matsumoto, K. Nakakoji, Y. Takada, S. Takada and K. Shima, "Ginger2: an environment for CAESE (computer-aided empirical software engineering)," *IEEE Trans. on Software Engineering*, vol.25, no.4, pp.474-492, 1999.
- [16] P. M. Johnson, H. Kou, J. M. Agustin, Q. Zhang, A. Kagawa and T. Yamashita, "Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from Hackstat-UH," *Proc. of the 2004 intl. Symposium on Empirical Software Engineering (ISESE2004)*, pp.136-144, Redondo Beach, CA, 2004.
- [17] J. Froehlich and P. Dourish, "Unifying artifacts and activities in a visual tool for distributed software development teams," *Proc. of the 26th intl. Conf. on Software Engineering (ICSE'04)*, pp.387-396, Scotland, UK, 2004.
- [18] T. Zimmermann, P. Weissgerber, S. Diehl and A. Zeller, "Mining version histories to guide software changes," *Proc. of the 26th intl. Conf. on Software Engineering (ICSE'04)*, pp.563-572, Scotland, UK, 2004.
- [19] J. D. Herbsleb, A. Mockus, T. A. Finholt and R. E. Grinter, "An empirical study of global software development: distance and speed," *Proc. of the 23rd intl. Conf. on Software Engineering (ICSE'01)*, pp.81-90, Toronto, Canada, 2001.
- [20] D. German and A. Mockus, "Automating the measurement of open source projects," *Proc. of the 3rd Workshop on Open Source Software Engineering*, pp.63-67, Portland, Oregon, 2003.
- [21] A. H. Dutoit and B. Bruegge "Communication metrics for software development," *IEEE Trans. on Software Engineering*, vol.24, no.8, pp.615-628, 1998.
- [22] The SPARS (Software Product Archiving and Retrieving System) project, <http://iip-lab.ics.es.osaka-u.ac.jp/SPARS/index.html>.
- [23] 茂呂雄二 (編著), "実践のエスノグラフィ," 状況論的アプローチ 3, 金子書房, 東京, 2001.

(平成 xx 年 xx 月 xx 日受付)

#### 大平 雅雄

1974 年生 . 1998 年京都工芸繊維大学工学部電子情報工学科卒業 . 2003 年奈良先端科学技術大学院大学情報科学研究科博士課程修了 . 同年同大学産学官連携研究員 . 2004 年同大学情報科学研究科助手 . 博士 (工学) . HCI, CSCW, CSCL 等の研究に従事 . 教育システム情報学会, ACM, IEEE 各会員 .

#### 横森 励士 (正員)

1975 年生 . 1999 年大阪大学基礎工学部情報工学科卒業 . 2003 年同大学院博士課程修了 . 同年同大産学官連携研究員 . 博士 (工学) . プログラム構造解析の研究に従事 . 情報処理学会, IEEE 各会員 .

#### 阪井 誠 (正員)

1962 年生 . 1984 年大阪電気通信大学工学部電子機械学科卒業 . 同年 SRA に入社 . 以来, ソフトウェア開発 . 研究開発に従事 . 2001 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了 . 同年 SRA 先端技術研究所シニア研究員 . 博士 (工学) . 開発支援環境, グループウェア, セキュリティに興味を持つ . 情報処理学会, IEEE, ソフトウェア技術者協会各会員 .

#### 岩村 聡

1975 年生 . 1999 同志社大学工学部電子工学科卒業 . 同年 NTT ソフトウェア株式会社に入社 . 以来, ソフトウェア開発 . 研究開発に従事 .

#### 新海 平

1974 年生 . 2001 年大阪大学大学院理学研究科博士後期課程中退, 同年 (株) 日立システムアンドサービスに入社 . 研究開発センタ, 生産技術センタで Web システム開発に従事 . 2003 年よりエンピリカル環境における EPM 開発に従事 .

#### 小野 英治

1975 年生 . 2000 年大阪大基礎工学部情報工学科卒業, 同年日立公共システムエンジニアリング株式会社に入社, 以来, ソフトウェア開発 . 研究開発に従事 .

#### 横川 智教

1977 年生 . 1999 年大阪大学基礎工学部情報科学科中退 . 2001 年同大学院基礎工学研究科博士前期課程修了 . 2004 年同大学院博士後期課程修了 . 同年岡山県立大学情報工学部助手 . 博士 (工学) . システムの形式的検証に関する研究に従事 . IEEE 会員 .

**Abstract** The goal of our research project (the EASE project) is to establish Empirical Software Engineering, a development approach based on scientific evidence, in the field of software development where there are many issues regarding reliability and productivity. We have been developing a computational environment for supporting practical software process improvement. This paper describes Empirical Project Monitor (EPM) which is an automatic data collection and analysis system. EPM automatically collects and measures history data accumulated during everyday software development activities. By providing graphical views of measurement results, EPM helps developers and managers keep their projects under control in real time without additional work.

**Key words** Software Process Improvement, Automatic Data Collection, Quantitative Data Analysis, Empirical Software Engineering