社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

開発履歴データのリアルタイム収集・分析システム EPM の拡張について - SRGM を用いた予測グラフの実現および既存解析システムとの連携 -

横森　励士† 　市井　誠† 　新海　平†† 　井上　克郎†

† 大阪大学 大学院情報科学研究科
†† (株) 日立システムアンドサービス 研究開発センタ
E-mail: †{yokomori,m-itii,inoue}@ist.osaka-u.ac.jp, ††tshinkai@empirical.jp

あらまし　ソフトウェアの開発においてプロセス改善を行うためには，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要不可欠である．EASE プロジェクトでは，開発履歴データのリアルタイム収集および定量的なデータ分析を行うシステムとして，EPM(Empirical Project Monitor) を開発している．本研究では，EPM において実現した，もしくは実現を予定している EPM の拡張について説明する．具体的には，SRGM を用いた潜在フォールト数の予測グラフの EPM 上での実現手法について提案するとともに，既存の解析システムとの連携として，コードクローン解析システムである CCFinder や Java 部品検索システムである SPARS-J などを取り上げ，これらを用いて，EPM とどう連携させていくべきか，どのような情報が取得できるかについて考察する．

キーワード　ソフトウェアプロセス改善，実証的ソフトウェア工学，SRGM，コードクローン，ソフトウェア部品検索，ユースケースポイント法

Enhanced Real-Time Software Development Management System EPM: Graph using SRGM and Linkage with Source Code Analysis Systems

Reishi YOKOMORI†, Makoto ICHII†, Taira SHINKAI††, and Katsuro INOUE†

† Graduate School of Information Science and Technology, Osaka University
†† Hitachi Systems & Services, Ltd.
E-mail: †{yokomori,m-itii,inoue}@ist.osaka-u.ac.jp, ††tshinkai@empirical.jp

Abstract　Understanding the current development status is the essential activities for improvement of the development process. Our EASE project developed a real-time data collection and analysis system, named EPM(Empirical Project Monitor). EPM collects the development history data and make a quantitative data analysis. In this paper, we suggest an enhancing method for EPM. At first, we suggest a graph which predicts the number of latent defects by using several SRGM models. We also discuss about the cooperation method with the existing source code analysis tools, such as code-clone analysis system (CCFinder), and software component search system (SPARS-J), and so on.

Key words　software process improvement, empirical software engineering, SRGM, code clone, software component search, use case point

1. ま え が き

近年のソフトウェア開発現場では，ソフトウェアの大規模化と開発期間の短縮化という二つのリスク要因を抱えた状態での開発を余儀なくされている．これらのリスク要因は，開発が間に合わない，品質が低下するという形で顕在化し，たった一つのプロジェクトの失敗であっても，それが組織全体に大きな被害をもたらしている．

これらのリスク要因を軽減するために，ソフトウェアの再利用の推進やリファクタリングのようなコストを抑える取り組みや，アジャイルな手法の一部採用などの開発手法の選択も重要である．しかしやはり一番有効であるのは，問題の早期発見のための絶え間ないチェックと，問題の早期解決を目指すことで被害の拡大を防ぐことである．この問題の早期発見，早期解決を行うためには，トヨタ式カイゼン方法の一つである「視える化」のように，問題点が常に「視える」ようにしてお

く工夫が必要である [1]．プロセス改善のプラクティスをまとめた CMM [2]/CMMI [3] においても高いレベルでは，開発中に得られるデータの分析をもとにしたソフトウェアプロセスの改善が要求される．これらの「視える」化されたデータの分析においては，様々な仮説を考察し，それぞれをデータに基づいて検証していく，いわゆる実証的ソフトウェア工学 (Empirical Software Engineering) という観点が非常に重要である．

このような課題を抱えているソフトウェア開発の分野を対象として，我々の研究プロジェクトでは定量的なデータの収集・分析に基づく開発プロセスの改善手法の確立を目指し，EPM を構築してきた [4], [5]．EPM は，実際の開発においてよく利用されているような，構成管理システム（CVS），メーリングリスト管理システム（Mailman, Majordomo），フォールト情報管理システム（GNATS, Bugzilla）における履歴情報から様々なデータを抽出し，それらをグラフ化して表示する．

現在のところは，システムから得られるデータをそのまま集計して表示したり，複数のプロジェクトを比較のために並べてグラフ表示するなどの機能だけであるが，今後 EPM に対して「現状が問題ないか」や「今後どう推移するか」を予測する機能の追加が必要になると我々は考えている．また，CVS に登録されたソースコードや設計資料からより詳細な情報を得るためには，既存の解析システムとの連携も重要であると考えられる．

本研究では，これらの考えに基づいて行われた EPM の拡張について提案し，考察する．具体的には，フォールトの累積発見数の推移から潜在フォールト数を推測するためのモデルである SRGM を用いて，フォールト情報管理システムのログ情報から累積フォールト数の予測グラフを EPM 上で表示する機能を提案する．また，既存の解析システムとの連携として，コードクローン解析システムである CCFinder や Java 部品検索システムである SPARS-J，ユースケースポイントの測定支援システムなどを取り上げ，これらを用いて，EPM とどう連携させていくべきか，どのような情報が取得できるかについて考察する．

以下，2. 章では，EPM の説明を行う．3. 章では，現状の問題点について考察した後，SRGM を用いた EPM 上でのグラフ化機能を提案した上で，コードクローン解析システムや，ソフトウェア部品検索システムとどのように連携させるべきかについて考察する．最後に，4. 章でまとめと今後の課題について説明する．

2. EPM

ソフトウェア開発においては，通常，複数のメンバーがそれぞれの役割に応じて作業を分担し組織的に開発を行う．その中では，バグ報告書などを用いてテストの運用を行うなどの手法が一般化されており，メールをコミュニケーション手段として使うことが一般的になっている．その際，CVS などの構成管理システムを利用してソフトウェアを開発することや，Bugzilla や GNATS のようなフォールト情報の管理ツールなどが利用されることは珍しくない．

EPM は，これらの開発支援システムにおける履歴データを自動的に収集し，分析・グラフ化を行うシステムである．EPM は，実際の開発現場では困難がつきまとう一貫性のある定量的なデータ収集を支援し，プロジェクト管理を目的としたリアルタイムでのデータの分析・利用を容易にする．

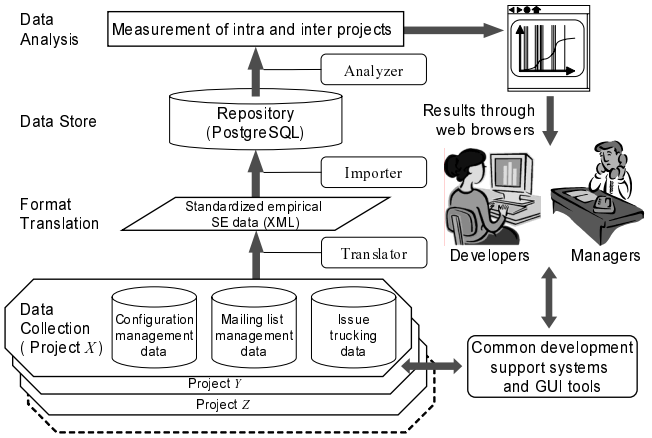

図 1　Empirical Project Monitor の概要図

図 1 は，EPM の概要図である．EPM は，Linux 上で動作するシステムで，データ収集，データ変換，データ蓄積，データ分析・グラフ化の 4 つの基本機能から構成されている．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よく利用される開発支援システムからデータを収集する．収集されるデータはソースコードのチェックイン，チェックアウト等のイベント情報や，投稿されたメールのヘッダや障害報告の投稿時間，件名，差出人等の情報である．これらは，ツールを用いた開発作業の過程で蓄積されるもので，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．

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 XML 形式のデータに一度変換する．XML 形式のデータに変換する理由としては，既存の開発環境では似たようなシステムが既に運用されていることが多いため，それらのシステムに対しても対応することができるようにするためという点が挙げられる．これにより，分析対象データの入力データ形式を統一でき，EPM 上での分析環境の実現や流用が容易になることが期待できる．

データ蓄積部では，変換された XML 形式のデータを読み込み PostgreSQL データベースに格納する．分析を行う際には，このデータベースが逐次必要な情報を提供する．

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．これらのデータ分析は Java サー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る．現在 EPM は,『ソースコードの規模推移』,『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』,『累積・未解決障害件数と平均障害滞留時間との関係』などの 5 種類のグラフや，CVS 内のイベントやメール情報の集計表などを提供する．グラフの表示の際には，複数プロジェクトのデータを同時に表示することもでき，類似プロジェクト間の比較も

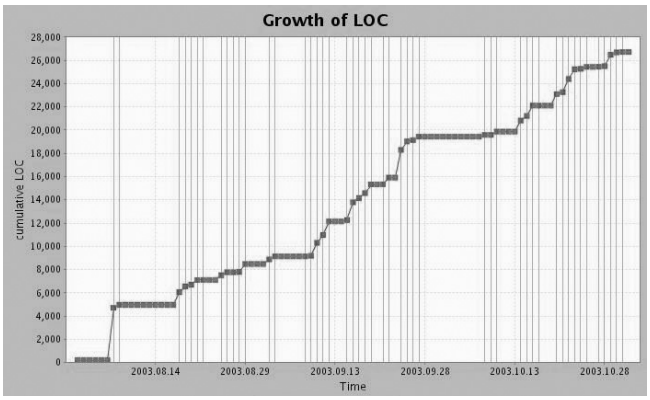

図 2　ソースコードの規模推移

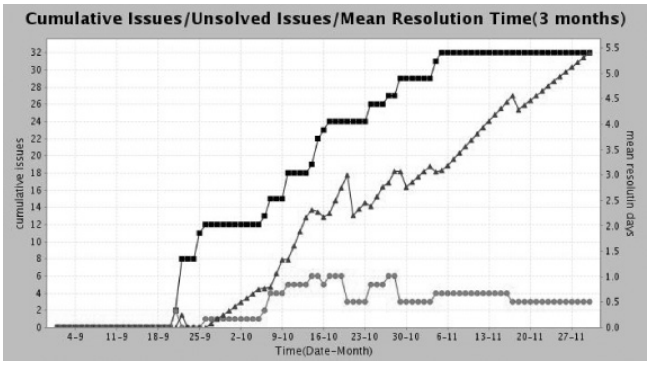

図 3　累積・未解決障害件数と平均障害滞留時間との関連

容易に行える．また，さらなる分析のために，SQL 文の入力によるユーザ独自の分析（グラフ化）も可能としている．

図 2 は表示されるデータの例で，このグラフでは『ソースコードの規模推移』を表し，ソースコード規模の推移 (図中，黒の折線) を LOC で示している．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，開発が停滞しているのか，開発やテストによる更新が活発に行われているのかを明確に確認することができる．このようなグラフを用いて説明を行うことで，プロジェクト管理者にとってリアルタイムで開発状況の把握をやりやすいということだけではなく，開発者にとっても定量的データを用いて具体的に開発状況が説明できるという点で有益であると我々は考えている．

また図 3 は『累積・未解決障害件数と平均障害滞留時間との関係』を表すグラフで，障害報告の累積 (上の折線)，未解決障害件数の推移 (下の折線)，平均障害滞留時間 (真中の折線) 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．

これらのグラフは，単一プロジェクトに対する分析結果であるが，過去のプロジェクトにおける情報を同時に表示することも可能である．これにより，類似プロジェクトの情報を予測に利用することも可能で，現状の把握および今後の予測に利用できる．

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．また，収集されるデータは人為的操作がされにくく，一貫性のあるデータを提供できる．これにより分析結果の共有化を容易にし，議論の場で現状についての認識を統一するのに有効であると考えられる．

3.　EPM の拡張について

前節で述べたように，EPM を用いることで定量的な開発データの収集や，分析を容易に行うことができると我々は考えているが，現在の EPM に対しては，次のような課題があるとも考えている．

- 予測機能の充実化

現在の EPM では，現在までの情報を取得し集計して，それをそのまま表示するだけの機能しか存在しない．現在の機能では，現状を確認するという意味では有効である．しかし，ユーザが「この状態が問題になりうるか」であるとか,「今後どう推移することが期待できるか」ということを知るためには，現状ではユーザが自分の手で類似プロジェクトを探索し表示し，類似プロジェクトの情報をもとに推測する方法しかサポートできていない．実際の利用者（開発者，管理者）に利用してもらうための動機付けとして,「予測機能の充実」や「類似プロジェクトの検出を支援する機能」が必要であると我々は考えている．

- 既存の解析システムとの連携

現在，CVS に登録されているプロダクトについては，どのようなソースコードに対しても適用可能なように，ログ情報から更新時期や各時点での LOC などの基本的なメトリクスを抽出するだけである．実際の現場では，コメントを省いた LOC や命令数などのメトリクスが利用されることも多いが，現状の EPM ではそのあたりを考慮していない．また，設計段階や要求定義での情報についても特に考慮しておらず，データ形式を解析し利用する仕組みも必要である．これらのデータを取得する際には，既存のツールを利用して登録されたプロダクトを直接解析することも一つの選択肢であると我々は考えており，それらのツールとの連携により具体的で有用な情報が抽出できると考えている．実際に，ソースコードを解析して情報を抽出するツールとして色々な提案がなされており [6], [8]，これらを EPM と有効に結びつける仕組みを提案することが重要であると考える．

我々は，これらの考察を踏まえ，ソフトウェアのフォールト発見数の推移から潜在フォールト数を推測するためのモデルである SRGM を用いることで，潜在フォールト数および累積フォールト数の予測グラフを作成する機能を，EPM の次バージョンである Ver 0.92 において実現した．このバージョン 0.92 では，この SRGM を用いたグラフ作成機能とともに，情報のグラフ化のカスタマイズを考慮するためのプラグイン化がなされている．また，既存のソースコード解析システムとの連携として，コードクローン解析システム CCFinder や部品検索シス

テム SPARS-J，ユースケースポイント計測支援システムなどを取り上げ，これらのシステムを EPM とどう連携させていくべきかについて考察する．

3.1 SRGM を用いたグラフ作成機能について

SRGM(Software Reliability Growth Model) は，ソフトウェアの信頼性を定量的に評価するためのモデルで，テスト段階においてフォールトが発見され，除去される過程をモデル化したものである．SRGM では，あるテスト区間 t までの累積フォールト発見数の推移から，モデル内のパラメータを推測することで，ソフトウェア内に残存するフォールト数を推定する．

これまでに多くの SRGM [11]〜[14] が提案されており，それぞれのモデルにおいて，異なる想定のもとにモデル化がなされている．例えば，指数型 SRGM [11] の場合は，フォールト一個当たりのフォールト発見率は一定であるという仮定が置かれているが，習熟 S 字型 SRGM [14] の場合は，時間と共にフォールト一個当たりのフォールト発見率が向上するという仮定のもとにモデル化がなされている．また，遅延 S 字型 SRGM [13] の場合には，故障が発生してからフォールトを特定するまでの時間のずれを考慮していたり，修正指数型 SRGM [12] の場合には，発見しやすいフォールトと発見しにくいフォールトの 2 種類の存在を考慮しているなど，様々な前提条件のもとに様々なモデルが提案されている．各手法についてのもっと詳しく知りたい場合は, [15] における解説をお勧めする．

予測モデルとして SRGM を用いたグラフ作成機能を作成するにあたり，これらの様々なモデルを利用することを考えた．過去の研究 [17] では，遅延 S 字型，習熟 S 字型の SRGM が比較的精度が高いと評価されている．しかし，現在の開発現場においては，オブジェクト指向型言語が一般的になり，開発手法や開発対象が大きく変化しているなど，その当時の環境とは大きく異なっている．そのため，現在の開発においてもこれらのモデルがそのまま利用できるかという点においては，検証を行う必要がある．

現在，複数の SRGM を統合したモデルとして，古山によって統合モデル [16] が提案されている．このモデルは，既存の複数のモデルを一つの式で表すことに成功しており，有用性が高いが，この手法は累積フォールト発見データ数の推移を曲線へ当てはめることに主眼が置かれている．我々は，パラメータだけではなく，それぞれのモデルを定義する際の前提も必要であると考えた．そのため，実際にどのような SRGM が現実的なモデルであるかを実証により確認することを目的として，それぞれのモデルに対して，パラメータの推定および検定を行い，検定によって適合していると判断できるモデルをグラフ上に表示するという手法を採用した．このときグラフを表示する際には，実際にどのようなモデルが現実のデータにあっているかを表示することで，各開発プロジェクトのデバッグ工程がどのような性質を持っているかを確認できる．

3.1.1 SRGM プラグイン

我々は，前述の考察に基づいて，フォールト管理システムが出力するログ情報から自動的に複数の SRGM を適用し，各 SRGM の累積発見フォールト数の予測結果をグラフとして表示する機能を EPM 上で選択可能なプラグインとして実現した．

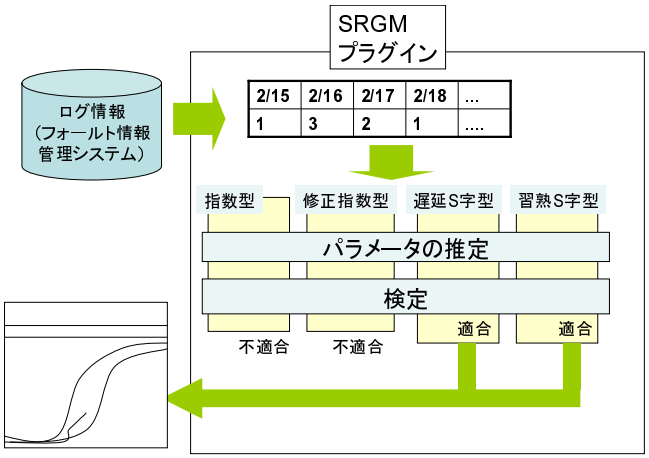

図 4　SRGM プラグインの構成

このプラグインでは，指数型 [11]，修正指数型 [12]，遅延 S 字型 [13]，習熟 S 字型 [14] の 4 種類の SRGM をサポートしており，入力方法として，標準設定とカスタマイズ設定の 2 種類の入力方法を提供する．標準設定では，予測対象となるテスト期間を入力するだけで，上記 4 種類のモデルに対して自動的にパラメータの推定を行う．一方，カスタマイズ設定では，モデルの種類や，いくつかのパラメータを指定した上で，SRGM を適用する．

図 4 は実現したプラグインの構成図で，処理の流れを以下に示す．

（1） PostgreSQL データベースに登録されているフォールト情報管理システムの履歴データから，当該プロジェクトの障害報告データを取得する．

（2） (1) で取得したデータから，指定された期間の各日における累積フォールト発見数を計算する．

（3） 4 種類の SRGM についてそれぞれ，(2) で得られた累積フォールト数の推移データから，各モデルにおけるパラメータを推定する．

（4） (3) において各 SRGM において推定されたパラメータをもとに，各モデルが実際のデータと適合しているかどうかをコルモゴロフースミルノフ適合度検定法を用いて検定を行い，適合しないモデルを棄却する．

（5） (4) の検定で適合している各モデルに対して，予測期間におけるフォールト累積データと，収束時の累積フォールト数を求め，グラフ表示プラグインを通じてグラフの表示を行う．

図 5 は，SRGM プラグインの出力画面例である．グラフ上では，実データにおける累積フォールト数の推移のグラフと共に，累積フォールト数の推移の期待（平均）値および最大・最小限界値，収束時の累積フォールト数に関する曲線がグラフ上に表示される．本プラグインにより，ソフトウェアのデバッグの進捗度や現在のソフトウェアの品質の予測がフォールトデータの集計や表の作成などのコストなしで可能となり，実際の開発において EPM を用いることで開発者は有益な情報を手軽に共有できると考えられる．

図 5　SRGM プラグインの出力例

3.2　ソースコード解析システムとの連携について

本節では，既存のソースコード解析システムとの連携として，コードクローン解析システムである CCFinder や Java 部品検索システムである SPARS-J，ユースケースポイント測定支援システムを取り上げ，これらのシステムを EPM と連携させることで，どのような情報が取得できるかについて考察する．

3.2.1　コードクローン解析システム CCFinder

CCFinder [6] は神谷らによって提案されたコードクローン検出システムである．コードクローンとは、ソースコード中の全く同じあるいは類似したコードの断片を指す．コードクローンが大量に存在するソースコードに対して修正を行う場合を考える．この時，一つの修正が他のコードクローンに対しても必要であることが多く，ソースコード中からコードクローンを探してきては修正を行うという作業が必要となり，保守が困難になる．保守作業を円滑に行うためには，コードの再構築を行う作業であるリファクタリングを定期的に行うのが望ましいと言われている．

CCFinder では C/C++, Java, COBOL, Fortran などの多様な言語を対象に，コードクローンの位置情報を抽出する他に，コードクローンに関するいくつかのメトリクスを計算している．EPM においては，コードクローンを EPM 上でそのまま見せることも考えているが，これらのメトリクスを CVS に登録されているソースコードの各リビジョンごとに計算し，その推移をグラフ化する機能も有用であると考える．

我々は，リファクタリングを行う際の目安として，コードクローンカバレッジ (Coverage of code clone) と呼ばれるメトリクスが重要であると考えている．コードクローンカバレッジは，モジュールに含まれる行のうち，コードクローン列に含まれる行の割合を指し，この割合が高いほど，ソースコード中にコードクローンが多く存在することを指す．一般的にリファクタリングにかかるコストは大きいため，これらの定量的な指標を目安にリファクタリングを行うタイミングを計ることが重要であると考えられる．

また，CVS への各更新作業について，その前後での差分に対するコードクローンカバレッジを取得することで，その更新の性質や位置付けなどがわかるのではないかと考えている．例えば，差分内や差分間のコードクローンを検出することで，パターン化された修正を検出できたり，コードクローンを埋め込んだ修正の特定に利用できるであろう．また，差分と本体とのコードクローンを検出することで，修正忘れなどがあるかないかのチェックなども効率的に行うことができる．このような情報の提供は，開発者にとって有効なもので，効率的な開発を支援できると考えている．

3.2.2　ソフトウェア部品検索システム SPARS-J

SPARS-J [8], [9] は Java のソースコードを対象としたソフトウェア部品検索システムである．SPARS-J は，依存や類似といったソフトウェア部品特有の特性を考慮しながら大規模なライブラリを自動的に構築し，キーワードとトークン種類を検索キーとした全文検索を提供する．SPARS-J は EPM と同様に Linux 上でも動作し，運用形態も EPM と同様にサーバに検索用のデータベースを構築した上で，ウェブブラウザを通じて分析結果や検索結果を提供するという形をとっている．

SPARS-J をソフトウェア開発に用いることで，ライブラリの知識が無い開発者も有用なソフトウェア部品やそれに付随する有益な情報を容易に入手することができる．[9] では，企業において実際の業務におけるソースコード管理支援を目的として SPARS-J を利用した際の評価実験が紹介されている．回答されたアンケートからはソフトウェアシステムの全体像の把握に非常に適しており，部品管理に有効であるとの結果が得られている．EPM と SPARS-J との親和性は非常に高く，EPM と SPARS-J を同時に運用するのは困難ではない．CVS からソースコードを抽出し，SPARS-J に登録することで，運用上のコストなしで部品管理のための環境を EPM に追加できると考えられる．

また，SPARS-J では順位付けの指標として，Component Rank を利用した評価手法を採用している．Component Rank は部品間の利用関係から，部品の重要度を判定する手法で，Component Rank が高いほど，その部品はよく利用されている，重要な部品から利用されていることがいえる．このとき，CVS リポジトリ中の各更新に対して，各更新時点での部品集合に対して，Component Rank を求めることを考える．開発の初期段階では，更新により機能が追加されていくため，Component Rank は大きく変動すると思われるが，開発の後期段階では，機能追加が少なくなり，部品間の関係が安定し始め，Component Rank の変動は少なくなっていくと考えられる．このことから，Component Rank の変動の度合いを測ることで，ソフトウェア開発プロジェクトの実装の収束度を計る目安になるのではないかと考えている．

3.2.3　ユースケースポイント測定システム

ソフトウェアの開発において見積もりをすばやく正確に行うことは，重要な活動の一つである．その時，見積もりを行う段階が早ければ早いほど，開発計画も立てやすくなり開発プロセスが潤滑に動きやすくなる．近年のソフトウェア開発において

は，様々な事情から要求が変更されることが頻繁にあるため，変更のたびに正しい見積もりを行うことが要求される．

設計前の要求定義の段階において工数の規模見積もりを行う手法として，ユースケースポイント法が提案されている [18]．松川らはユースケースポイント法に基づいてユースケースポイント計測の自動化を支援するシステムを開発した [19]．このシステムでは，アクタやユースケースに対する重み付けを支援することで，見積もり経験の浅い人間であっても，ある程度の精度で見積もりを行うことができる．

このシステムを EPM と連携させることで，設計情報をもとにした工数の予測機能を EPM に追加することができる．このシステムでは，重み付けに必要な情報が不足している場合でも，過去の情報から自動的に重み付けを定義する．このとき，CVS でプロジェクト内のユースケース図を管理することで，ユースケースに対する変更・追加があった場合でも，過去の登録された情報をもとに見積もりを早期に自動的に再計算でき，計画変更に対しても柔軟に対応できるようになると考えられる．

今節では，以上 3 つのシステムと EPM との連携を考察した．これらの連携を実現させるためには EPM 側に，

- 設定情報をもとに，定期的に解析システムを実行させ，
- 解析結果を XML 形式に変換しデータベースに登録し
- データベースの情報をもとにグラフを描く

ような汎用的なプラグインが必要であると思われる．このような形で EPM 上での連携を容易にすることで，様々な仮説を考察しデータに基づいて検証していく，エンピリカルソフトウェア工学の実現にふさわしい環境になると考えられる．

4. ま と め

本研究では，予測機能の充実化，既存の解析システムとの連携という点に着目し，EPM の拡張について提案および考察した．実現した SRGM プラグインを利用することで，EPM 上においてソフトウェアのデバッグの進捗度や，現在のソフトウェアの品質を予測することが可能となると考えられ，実際の開発において有益な情報を提供できると考えられる．今後の課題としては，SRGM プラグインの実際のプロジェクトに対する適用実験の他に，協調フィルタリングや情報のクラスタリングによる類似プロジェクトの検出手法の提案などが挙げられる．また既存の解析システムとの連携においては，CCFinder, SPARS-J，ユースケースポイント測定支援システムなどの既存の解析システムとの連携について考察し，連携の有用性を確認した．今後の課題として，これらの考察をもとにした EPM 上での連携機能の実現が挙げられる．

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．本研究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．

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] 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.

[3] 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-TR001, 2002.

[4] 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, pp.228–239, 2005.

[5] 大平 雅雄, 横森 励士, 阪井 誠, 松本 健一, 井上 克郎, 鳥居 宏次: ”Empirical Project Monitor: プロセス改善支援を目的とした定量的開発データの自動収集・分析システムの試作”, 電子情報通信学会技術報告, Vol.103, No.708, SS2003-48, pp.13–18, 2004.

[6] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, ”CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code,” IEEE Trans. Software Engineering, Vol. 28, No. 7, pp. 654–670, 2002.

[7] 井上克郎, 神谷年洋, 楠本真二, ”コードクローン検出法,” コンピュータソフトウェア, Vol.18, No.5, pp.47–54, 2001.

[8] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, Shinji Kusumoto: ”Ranking Significance of Software Components Based on Use Relations”, IEEE Trans. Software Engineering, to be published Mar. 2005.

[9] 横森 励士, 梅森 文彰, 西 秀雄, 山本 哲男, 松下 誠, 楠本 真二, 井上 克郎: ”Java ソフトウェア部品検索システム SPARS-J”, 電子情報通信学会論文誌 D-I, VolJ87-D-I, No.12, pp1060–1068, 2004.

[10] 横森 励士, 藤原 晃, 山本 哲男, 松下 誠, 楠本 真二, 井上 克郎: ”利用実績に基づくソフトウェア部品重要度評価システム”, 電子情報通信学会論文誌 D-I, Vol.J86-D-I, No.9, pp.671–681, 2003.

[11] Goel. A. L. and K. Okumoto :“ Time-dependent error-detection rate model for software reliability and other performance measures ”, IEEE Trans. Reliability, Vol. R-28, No. 3, pp.206–211, 1979.

[12] S. Yamada and S. Osaki:“ Nonhomogeneous error detection rate models for software reloability growth ”, in Stochastic Models in Reliability Theory, Osakiand Hatoyama(eds.), pp 120–143, Springer-Verlag, 1984.

[13] S. Yamada, M. Ohba and S.Osaki:“ S-shaped reliability growth modeling for software error detection ”, IEEE Trans. Reliability, Vol. R-32, No. 5, pp. 475–478, 1983.

[14] M. Ohba: “ Inflection S-shaped software reliability growth model ”, in Stochastic Models in Reliability Theory, Osakiand Hatoyama(eds.), pp 144–165, Springer-Verlag, 1984.

[15] 山田 茂, 高橋 宗雄:”ソフトウェアマネジメントモデル入門 ー ソフトウェア品質の可視化と評価法ー”, 共立出版, 1993.

[16] 古山恒夫：”ソフトウェア信頼度成長モデルに関する統合モデルの解析的パラメータ推定法”, 情報処理学会論文誌, Vol. 37, No. 12, pp. 2326–2333, 1996.

[17] S. Yamada and S. Osaki :“ Software reliability growth modeling: Models and applications ”, IEEE Trans. Software Engineering, Vol. SE-11, No. 12, pp. 1431–1437, 1985.

[18] B. Anda, H. Dreiem, D.I.K. Sjoberg and M. Jorgensen,“ Estimating Software Development Effort based on Use Cases - Experiences from Industry ”, Fourth International Conference on the UML, pp. 487–504, 2001.

[19] 松川 文一, 楠本 真二, 井上 克郎, 英 繁雄, 前川 祐介: ”ユースケースポイント計測支援ツールの実装とその適用”, 情報処理学会研究報告 (2004-SE-144), Vol. 2004, No.30, pp.91–98, 2004.