ソフトウェアエンジニアリングの実践
～先進ソフトウェア開発プロジェクトの記録～

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター　編著

SE
SHOEISHA

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、誠にありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願い致しております。下記項目をお読みいただき、手順に従ってお問い合わせください。

● お問い合わせの前に

弊社Webサイトの「正誤表」や「出版物Q&A」をご確認ください。これまでに判明した正誤や追加情報、過去のお問い合わせへの回答（FAQ）、的確なお問い合わせ方法などが掲載されています。

正誤表	http://www.seshop.com/book/errata/
出版物Q&A	http://www.seshop.com/book/qa/

● ご質問方法

弊社Webサイトの書籍専用質問フォーム（http://www.seshop.com/book/qa/）をご利用ください（お電話や電子メールによるお問い合わせについては、原則としてお受けしておりません）。

※質問専用シートのお取り寄せについて

Webサイトにアクセスする手段をお持ちでない方は、ご氏名、ご送付先（ご住所/郵便番号/電話番号またはFAX番号/電子メールアドレス）および「質問専用シート送付希望」と明記のうえ、電子メール（qaform@shoeisha.com）、FAX、郵便（80円切手をご同封願います）のいずれかにて“編集部読者サポート係”までお申し込みください。お申し込みの手段によって、折り返し質問シートをお送りいたします。シートに必要事項を漏れなく記入し、“編集部読者サポート係”までFAXまたは郵便にてご返送ください。

● 回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

● ご質問に際してのご注意

本書の対象を越えるもの、記述個所を特定されないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

● 郵便物送付先およびFAX番号

送付先住所	〒160-0006　東京都新宿区舟町５
FAX番号	03-5362-3818
宛先	（株）翔泳社 編集部読者サポート係

はじめに

2004年10月1日、独立行政法人 情報処理推進機構（IPA）内に、ソフトウェア・エンジニアリング・センター（SEC）が設立された。設立当時、SECには以下の3つの事業があった。

- エンタプライズ系プロジェクト
- 組込み系プロジェクト
- 先進ソフトウェア開発プロジェクト

このうち先進ソフトウェア開発プロジェクト（以降「先進プロジェクト」という）は、他の2つのプロジェクトで検討、開発された手法を実際のソフトウェア開発で実践するためのプロジェクトであった。

SEC発足時点では、先進プロジェクトの対象となるソフトウェア開発事業は選定中であった。社会基盤となるソフトウェアであり、新たに作成されるものとしてプローブ情報プラットフォームソフトウェアが選定され、ソフトウェアを開発するために新たに技術研究組合を設立することとなった。2005年1月18日に創立総会が開催され、同年2月21日にソフトウェアエンジニアリング技術研究組合（COSE）が設立された。事務所は、文京グリーンコート センターオフィスに設置された。プローブ情報プラットフォームソフトウェアの開発は組合員企業各社から最新の技術を持ち寄って行われ、開発作業は各社分担分の結合試験までは各社で行い、各社の成果物を統合する結合試験からは品川のダヴィンチⅡビルで行った。

ソフトウェアの開発マネジメントにおいては、各社独自の方法を社内基準としているところもあったが、組合として統一した方式を採用した。そのため、一部では社内基準と併用した会社もあった。統一方式とするために、たとえば、障害の管理項目の検討などでは、項目の必要性や意味などが熱心に議論され、統一までに時間がかかったが中身の濃いものとなった。

2年弱で2回の開発サイクルを回し、その後は開発成果をデモンストレーションするためのソフトウェア開発作業を行った。その成果は2007年2月22日、23日に「Where 2.0 時代におけるリアルタイムプローブ情報の活用」として公開された。

このプロジェクトでは、プロジェクトの進行中にさまざまな定量データの計測を行い、収集されたデータの分析結果をプロジェクトマネジメントに反映するという実験が、これまでに例を見ない総合的な形で実施され、大きな成果を得た。

本書では、先進プロジェクトで実践されたソフトウェアエンジニアリングについて、実際のソフトウェア開発活動に沿って記述するとともに、プロジェクトで展開された主な技術について説明し、また、プロジェクトに参加された方からのメッセージをまとめた。

ソフトウェアエンジニアリング実践の一事例として参照していただき、品質や生産性の向

上、プロセスの改善など、高品質なソフトウェアを効率よく開発するための一助となれば幸いである。

謝辞

SECのソフトウェアエンジニアリング実践のためにご協力いただいたCOSE関係各位および本プロジェクトの計測・分析グループの各位に謝意を表します。

● EPM分析グループ
松村知子（奈良先端科学技術大学院大学、EASEプロジェクト）
森崎修司（奈良先端科学技術大学院大学、EASEプロジェクト）
玉田春昭（奈良先端科学技術大学院大学、EASEプロジェクト）

● コードクローン分析グループ
楠本真二（大阪大学大学院、EASEプロジェクト）
肥後芳樹（大阪大学大学院、EASEプロジェクト）
吉田則裕（大阪大学大学院、EASEプロジェクト）
馬場慎太郎（大阪大学大学院、EASEプロジェクト）

● チェックシート分析グループ
長岡良蔵（経済産業省 エンタプライズ系ソフトウェア開発力強化推進委員会）
ほか、プロジェクト見える化部会の各位

● ベンチマークデータ分析グループ
大杉直樹（奈良先端科学技術大学院大学、EASEプロジェクト、およびIPA SEC）
角田雅照（奈良先端科学技術大学院大学、EASEプロジェクト、およびIPA SEC）
柿元健（奈良先端科学技術大学院大学、EASEプロジェクト、およびIPA SEC）
菊地奈穂美（IPA SEC）
横山健次（IPA SEC）

2007年10月
独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター

目次

contents

第1部　進行中のプロジェクト計測とフィードバック

第2部　実践された技術とその活用方法

第3部　プロジェクトを支えた人々からのメッセージ

Report

ソフトウェアエンジニアリング技術研究組合（COSE）による先進的ソフトウェア開発事業

――プローブ情報プラットフォームソフトウェアの公開デモ

■先進のソフトウェアを先進の技法を適用して開発

タクシー、バス、物流車両など、道路にはさまざまな商用車両が走っている。それらの商用車両には、車両を有効に活用するため、タクシーであればタクシー配車システム、バスであればバス運行システムなどの各社各様のシステムが搭載されている。これら既存のシステムを使い、実際に走っている車両の走行情報を1つのサーバーに集めることで、高品質な交通情報をリアルタイムに提供できる「プローブ情報プラットフォームソフトウェア」が完成、2007年2月22日にデモが公開された。

公開デモの会場となった青山TEPIA

会場の青山TEPIAを出発するデモ車両

「プローブ情報プラットフォームソフトウェア」を開発したのは、経済産業省による新しいソフトウェアエンジニアリングを適用するための実証プロジェクトの1つである。このプロジェクトを実行するにあたり設立されたのが、NTTデータ、デンソー、トヨタ自動車、日本電気、日立製作所、富士通、松下電器産業の7社によるソフトウェアエンジニアリング技術研究組合「COSE」である。COSEによって、「プローブ情報プラットフォームソフトウェア」は開発された。


デモ車両の中に液晶ディスプレイを搭載して道路状況を表示


最新の情報に更新中であることを表示

COSEの最終目的は、「プローブ情報プラットフォームソフトウェア」を完成させることである。が、それと同時に目指したのが、ソフトウェアエンジニアリング適用による品質の確保と、ソフトウェアエンジニアリング手法の確立である。

ソフトウェアエンジニアリングの実践は、独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センターが主導した。今回、採用したソフトウェアエンジニアリング手法は、エンピリカル（実証的）アプローチである。それを実践するために利用したのが、文部科学省のリーディングプロジェクトであり、奈良先端科学技術大学院大学と大阪大学が中心になって活動するEASE（Empirical Approach to Software Engineering：ソフトウェア工学へのエンピリカルアプローチ）プロジェクトで研究開発された、「EPM（Empirical Project

Monitor)」などのソフトウェアツールである。これにより、定量データの自動収集から分析、フィードバックのプロセスが完成。そのほかにもEASEプロジェクトで開発された「コードクローン分析」「ロジカルカップリング分析」「相関ルール分析」「協調フィルタリング」などの、さまざまな分析ツールを適用。さらに、経済産業省のプロジェクト見える化部会で検討しているチェックシートを利用して、プロジェクト見える化部会の委員がヒアリングを実施するなど、SEC を通していろいろな手法の適用が行われた。これらを用いることで、高品質かつ効率的な開発が実現できたのである。

広域地図に現在位置を表示

立体表示画面に車線別の情報も表示できる

■プローブ情報プラットフォームソフトウェアとは?

プローブ情報プラットフォームソフトウェアとは、実際に走行しているタクシーやバス、物流車両から、時刻や位置などの固有の時空間データを、ネットワークを通じてリアルタイムに収集し、信頼度が付加された交通情報や交差点からの流出方向別の所要時間などを提供するような、付加価値の高いサービスに活用するためのプラットフォームである。交通情報は実際の走行車両が通過した隣接する2点の情報から、その間を通過した時間を基に、収集した車両台数や車両の種別を考慮して分析される。次世代ITSを実現する上でも、プローブ情報は現在、注目されている分野の1つだ。

その理由の1つは、車両自身がセンサーの役割を担うため、道路に車の空間情報を取得するためのセンサーの敷設が不要なことがあげられる。しかも、実際に走行している車両からリアルタイムに情報を収集して分析するため、より高精度な交通関連情報や運転支援情報が提供できるというわけだ。

今回のシステムが対象としたエリアは、東京23区を含む20Km四方である。そこに登録されている8500台にも及ぶ対象車両のうち、そのとき稼動している車両からのプローブ情報を対象に、3分30秒以内に収集から交通情報や推定情報の生成までの処理を完了する。そして5分ごとにリアルタイムな交通情報を提供することを視野に入れている。

携帯電話の画面も投影

車両台数、カバー率などを表示

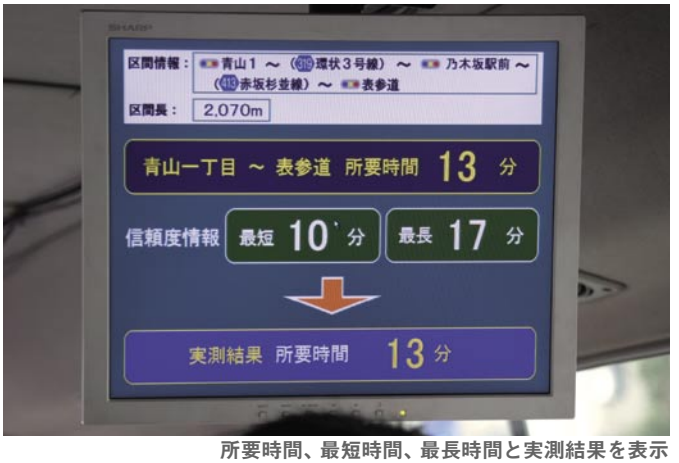
データの収集から送信までの仕組みは次のような流れとなる。まずタクシーやバス、物流車両などの各種車両走行の情報は、各事業者から送られ、センターのデータベースに一括して収集される。しかし、各事業者からは異なるデータ形式で提供されるため、フォーマットを統一する処理を行う。次に標準化されたデータを基に分析し、交通情報を生成する。そして生成された情報をインターネットを介して、無線LANや携帯電話（パケット通信）で送信する。このように書くと、「プローブ情報プラットフォームシステムは交通情報や旅行時間を算出するだけ」と思うかもしれないが、実はよりきめ細かい信頼性のある情報として提供できるような機能

を付加していた。それらの機能は、デモにより明らかになった。

■渋滞状況やリンク旅行時間をリアルタイムに提供

2007年2月22日および23日に行われたデモでは、実際にマイクロバスにプローブ情報プラットフォームシステムからの情報を受信するシステムを搭載し、渋滞状況やリンク旅行時間を算出する様子を披露した。まずは、このシステムが、23区内を含む20km四方のエリアの道路を、どれだけカバーしているかを示す受信画面が紹介された。デモ当日、稼動していたプローブカーは4600台強だが、どの道にどのくらいの車が走行しているかが一目で分かるように、「渋滞＝赤」「混雑＝橙」「順調＝緑」と渋滞度を状況別に色分けして、地図上に表示する。ちなみにデモ当日、このエリアの交通情報のカバー率は国道・都道府県道は約86%と表示されていた。

次に紹介された機能は、リンク旅行時間の算出機能である。外苑前から乃木坂にいたる交差点では、パケット通信を使って、右左折・直進リンク旅行時間が生成される場面を紹介。進行方向によって、目的地までの所要時間の違いが表示された。(x ページ下の写真参照) 具体的にいうと、直進して遠回りをする場合と、右折して通常のルートで目的地を目指す場合のリンク旅行



所要時間、最短時間、最長時間と実測結果を表示

車両によるデモと平行して講演会も開催された
(写真はパネルディスカッション)

時間が算出されるのである。事前に所要時間を表示することによって、ドライバーの選択肢が増えるというメリットが得られるというわけだ。実際にデモで表示された旅行時間はほぼ正確。その時間内に目的地にたどり着くことができた。

またこれら目的地到着までの時間は、「最短○分、最長△分」という形で表示されるのもこのシステムの特徴だ。信頼水準は70％。この幅で、最短、最長時間を算出し、より信頼性を考慮した情報を提供している。

これらの情報はマイクロバスに搭載されたパソコンで閲覧できるというだけではない。携帯電話でもこれらの情報を受信できるシステムが披露された。

■プローブ情報プラットフォームシステムは次世代ITSの基盤として活用することに期待

今回のプロジェクトで開発された「プローブ情報プラットフォームソフトウェア」は、タクシー会社の配車システムやバス会社の運行管理システムなど、既存のプローブ情報を活用したため、低コストながら、質の高い情報を提供できるシステムとなったと、経済産業省やCOSEでは評価している。

また新しいソフトウェアエンジニアリング手法の実践という意味でも、大きな評価を得ている。マルチベンダー開発におけるプロジェクト管理の仕組みづくりとともに、ベンダーごとの協調（公開）領域と競争（非公開）領域が混在した開発プロセスの見える化が、ある程度達成できたという。

今回の「プローブ情報プラットフォームソフトウェア」の開発プロジェクトは、2006年度で終了し、このソフトウェアは完成した。今後は、カーナビへの配信など、このソフトウェアをどのように実用化していくか、COSEに参加した各メーカーの取り組みが注目される。

IPA SECの展示コーナーでは、
「ソフトウェアエンジニアリングの実践」を解説

■公開デモ

正式には、「Where2.0時代におけるリアルタイムプローブ情報の活用　～先進的ソフトウェア開発プロジェクト公開デモンストレーション～」で、2007年2月22日（木）、23日（金）にTEPIA 4階ホール、3階エクジビションホール、および周辺道路で開催された。

TEPIA 4階ホールでは以下の講演が行われた。

2月22日（木）			
ご挨拶		経済産業省商務情報政策局	情報処理振興課長　鍛治 克彦
	COSE 理事長	株式会社 NTT データ	副社長　山下　徹
		独立行政法人　情報処理推進機構	ソフトウェア・エンジニアリング・センター　所長　鶴保 征城
基調講演	ソフトウェアエンジニアリングの実証的アプローチ	大阪大学大学院	教授　井上 克郎
基調講演	プローブ情報利活用の新たな可能性	東京大学生産技術研究所	教授　桑原 雅夫
パネルディスカッション	プローブシステムの可能性と更なる広がり	（コーディネーター）千葉工業大学	教授　赤羽 弘和
		財団法人　日本自動車研究所	香月 伸一
		株式会社アイ・トランスポート・ラボ	堀口 良太
		本田技研工業株式会社	柘植 正邦
		ヤマトシステム開発株式会社	玉川 雅浩
		株式会社 ゼンリンデータコム	清水 辰彦
2月23日（金）			
講演	プローブ情報プラットフォームの開発	COSE（株式会社ＮＴＴデータ）	SE部会長　勝又 敏次
講演	プローブ情報プラットフォームの実証実験による評価	COSE（株式会社デンソー）	評価WGリーダー　塚本　晃
講演	ソフトウェアエンジニアリングの実践	IPA/SEC	研究員　樋口　登

※役職は、公開デモ当時のものです。敬称略。

TEPIA 3階では、以下の展示が行われた。

COSE によるプローブ情報プラットフォームソフトウェアの展示	
COSE	COSE プローブ情報プラットフォームの6つの特徴
	COSE におけるソフトウェア開発の考え方
	COSE における V 字評価モデルの適用
IPA/SEC	ソフトウェアエンジニアリングの実践
COSE 組合員企業による展示（7社）	
利活用ケースの展示	
インターネット ITS 協議会	プローブ情報活用への取組みと研究事例紹介
NTT ソフトウェア株式会社	リアルタイムプローブ情報を活用した高度な車両位置情報の提供
キャンパスマップル株式会社	道路から地図を作る技術とリアルタイムプローブ情報の利活用

また、展示会場奥のミニセミナーコーナーでは、22日、23日とも以下のミニセミナーが行われた。

交通情報生成	松下電器産業株式会社	斉藤　貴光
補完・推定方法	株式会社日立製作所	横田　孝義
信頼度情報	日本電気株式会社	藤田　貴司
カバー率	富士通株式会社	市橋　文夫

TEPIA 周辺道路では、バス乗車によるデモンストレーションが行われた。プローブ情報プラットフォームソフトウェアの出力を受信するためのパソコンシステムと携帯電話をバスに持ち込み、実際にバスを走行させながらリアルタイムプローブ情報を受信するもので、以下の実験が行われた。

① ブロードバンド通信を想定した配信実験
② 右左折・直進交通情報の提供実験
③ 多くの道路をカバーする交通情報提供実験
④ 信頼度情報提供実験
⑤ 携帯電話による提供実験

両日とも天候にも恵まれ、多数の参加者があった。特に、22日には甘利明経済産業大臣、23日には渡辺博道経済産業副大臣が来場され、バス乗車によるデモンストレーションと展示会場を視察された。

第1部

進行中のプロジェクト計測とフィードバック

Section 1

先進ソフトウェア開発プロジェクトの概要

1.1 概要

独立行政法人 情報処理推進機構（IPA）ソフトウェア・エンジニアリング・センター（SEC）が提唱する手法を実証する場として、「先進ソフトウェア開発プロジェクト」（以下、先進プロジェクト）が実施された。先進プロジェクトの活動対象として経済産業省による中規模プロジェクトが選定された。それは「プローブ情報プラットフォームソフトウェア」と呼ばれるシステムソフトウェアを開発する計画で、2005年春に開始された。開発対象は公共的な情報システムで、その入力は、タクシー、路線バス、物流車両、などの各種車両をプローブカーとする車両位置情報、出力はこれらの情報を融合して分析することによって得られる国民生活に有意なリアルタイムの情報である。システムソフトウェアの開発とこれを用いた評価を行ったのは、7つの企業（トヨタ自動車株式会社、株式会社デンソー、株式会社ＮＴＴデータ、日本電気株式会社、株式会社日立製作所、富士通株式会社、松下電器産業株式会社）によって法律（鉱工業技術研究組合法）にもとづいて設立された技術研究組合である。この技術研究組合の名前が、ソフトウェアエンジニアリング技術研究組合、英語名称が、COnsortium for Software Engineering（ COSE ）である。

開発期間は2年余りで、この間に2回のシステム開発と公開デモンストレーション（以降「公開デモ」という）が行われた。本書はその2回の開発（フェーズ1、フェーズ2）および公開デモに関するものである。

ソフトウェアの開発は、COSEを構成するソフトウェア開発企業が分担したため、開発拠点が各地に分散することになり、いわゆる広域マルチベンダー開発となる。COSEの各企業は一部の開発対象領域においてはライバル関係にあり、この計画では互いに競争領域と協調領域を峻別し、競争領域では情報の開示を制限しながら、協調領域では情報共

有を図る必要があった。開発対象はLinuxサーバ上でRDBを使用するC++によるソフトウェアと、情報表示用のPC上のソフトウェアである。

1.2 ソフトウェアプロジェクト構成の特徴

このプロジェクトの運営の仕組みについては、下記に大略を紹介する。

COSEの組合員企業のうちトヨタ自動車が利用者としての役割を果たし、デンソーが利用者および開発者の役割を兼務し、デンソーを含む6社がソフトウェア開発を分担した。この中でNTTデータがもっぱら全体のプロジェクトマネージャ（PM）の役割を果たした。ソフトウェア開発の視点から見た概略の組織構成を図1-1に示す。この中で技術研究組合員企業各社で構成する技術委員会が実質的に全体の計画検討と、プラットフォームソフトウェアの仕様検討など、戦略会議として機能し、ソフトウェアエンジニアリング部会（SE部会）が、ソフトウェア開発の全般を管理するプロジェクト会議に相当する。技術委員会の活動については6ページのコラムで紹介する。

プローブ情報プラットフォームソフトウェアの要件定義書は、COSEの設立に先立ち経済産業省の委託により財団法人日本自動車研究所（JARI）のワーキンググループで策定された。COSEではこの要件定義を開発に移せるレベルにまで内容を詳細化したのちソフトウェア開発にとりかかった。

前述のようにCOSEでのソフトウェア開発は競争領域と協調領域を峻別して進められた。要件定義書に示された機能一覧にはその区別が明示されている。競争領域の情報は会社間で開示を制限し、協調領域の情報は共有される。

このような開発プロジェクトにおいて、プロジェクト計測はCOSE組合員各社とSECおよび連携するEASEプロジェクトの共同作業で進められた。

本書巻末の「対外発表と学術的な成果」に記載した資料を本文や図等で参照している場合は、[A.I]というように表記します。[A.1]は、「A.論文誌、国際会議等での発表」の資料番号1の資料を指します。

1.3 ソフトウェアエンジニアリング技術研究組合による開発のしくみ

仕様策定の構造

プローブ情報プラットフォームソフトウェアの基本的な要求仕様はプロジェクト開始に先立って、JARIのワーキンググループ（WG）で策定された。このWGに参加していた、トヨタ自動車、デンソー、ソフトウェア企業4社にNTTデータが加わって、技術研究組合が構成された。このWGでの要件定義では未定義の条件も多く、そのまま開発に移せるレベルではなかったが、ここでの規定事項はほぼすべて、技術研究組合での開発を拘束した。技術研究組合を構成した自動車関連製品メーカとソフトウェア企業4社では、このWGで要件定義作業に参加したメンバを設計作業の核要員として参加させた。

技術研究組合では、プロジェクト開始とともにこのWGでの要件定義を未定義事項の定義を含めて再定義した。この時点であわせて組合員各社の開発分担を決めた。

ついで、基本設計書、詳細設計書、プログラム設計書が作られ、製造工程となった。

情報共有の枠組み

技術研究組合でのソフトウェア開発は協調領域と競争領域を峻別して行われる。要件定義書に記述された機能一覧表には、その区別を明示している。

具体的には下記のように進められた。

要件定義書および基本設計書は共有。共同で基本設計書のレビューを行った。詳細設計書は共通機能と個別機能に分かれ、共通部のみ共有されほかは詳細を公開しなかった。

進捗報告では、設計工程では記述済ページ数、製造工程ではコーディング済モジュール数などによって、申告レベルで進捗が報告された。

ソースコードやソースコード行数は公開しなかった。

プログラム設計は製造工程に含まれ、工程全体の公開は控えられた。

PMの役割のシステムインテグレータ企業にもこの原則が適用され、基本設計書と詳細設計書の個別部分、製造工程、ソースコードは公にしないこととなった。ただし、ソースコード行数のみは、途中経過を含めて各社から知らされた。

プロジェクト計測の枠組み

プロジェクト計測とフィードバックの活動は、経済産業省の指導のもとに技術研究組合各社の全面的な協力を得て、組合員各社、SEC、EASEプロジェクトの共同作業で進められた。SECもEASEプロジェクトもシステム開発の受発注、組織系統の中には含まれないが、この活動はいわば発注者の意図によって実行された。SECは経済産業省の外部機関であり、EASEプロジェクトを主導する奈良先端科学技術大学院大学（以降「奈良先端大学」という）および大阪大学とSECの間には業務委託契約が結ばれた。また、技術研究組合と奈良先端大学の間にも業務委託契約が結ばれた。

このほかに、奈良先端大学と技術研究組合の間で、EASEプロジェクトの提供するプロジェクト計測プラットフォームEPM(注)の利用に関する共同研究契約が結ばれた。こうした契約のなかで、守秘義務や知財の扱いが担保された。

EASE プロジェクト

column

2003年から開始された文部科学省の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「データ収集に基づくソフトウェア開発支援システム」をテーマにしたプロジェクトで、奈良先端大学、大阪大学が中心となって活動している。

EASEとは、Empirical Approach to Software Engineering（ソフトウェア工学へのエンピリカルアプローチ）の略で、ソフトウェア開発において定量的なデータに基づく科学的手法を適用することで、生産性や品質の向上を目指そうというもので、定期的にエンピリカルソフトウェア工学研究会を開催するなど、エンピリカルソフトウェアエンジニアリングの普及と実践を行っている。

http://www.empirical.jp/

注：EPM: Empirical Project Monitor
ソフトウェア開発における開発データの自動収集・分析環境で、開発支援ツールから情報を得るため、開発者に対するデータ収集の負荷が少なく、リアルタイムに分析できるのが特徴。具体的には、ソフトウェア開発環境の中で使用されているCVSのような構成管理システム、GNATSのような障害追跡システム、およびメール管理システムから自動的に開発管理情報を収集し、グラフのような可視的な形に分析・整理して表示する。

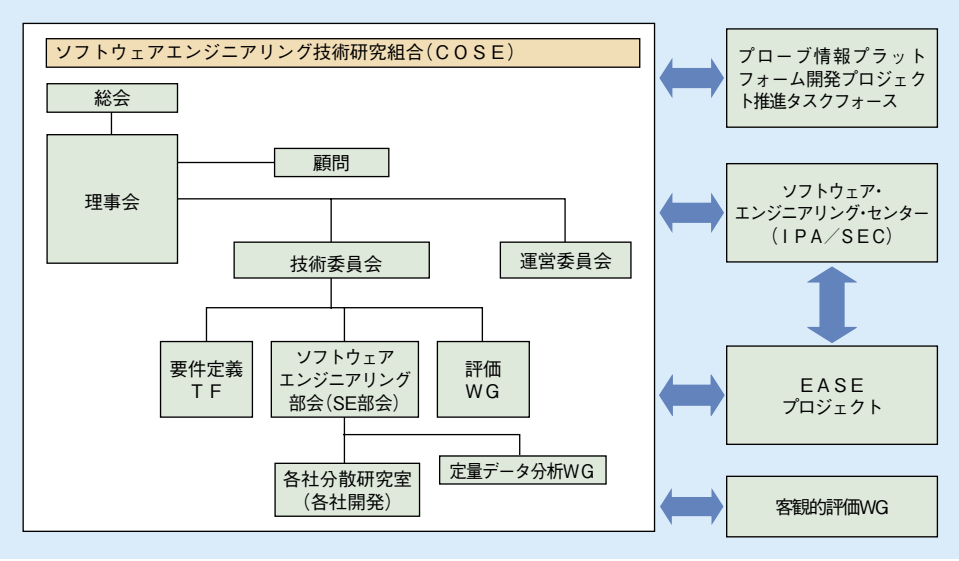


図1-1　ソフトウェアエンジニアリング技術研究組合（COSE）の組織図

技術委員会

column

COSEの活動に関する技術的な課題を討議するために技術委員会が設置された。COSE の活動期間中はほぼ毎週技術委員会が開催され、技術委員長を中心に熱い討議が繰り返された。1回の開催時間は平均すると約4時間だが、公開デモが近づくにつれて公開内容に関するレビューや検討が多くなり、深夜まで討議が続いた。

技術委員会の下に評価ＷＧが設置され、ユーザの視点からの評価項目や評価方法が検討された。評価ＷＧもほぼ毎週開催され、1回の平均開催時間は約5時間に及んだ。

技術委員会の下には、ソフトウェアエンジニアリングを推進し、ソフトウェア開発のマネジメントを行うソフトウェアエンジニアリング部会、要件定義を再検討する要件定義タスクフォース、公開デモの企画を検討する公開デモ企画ＷＧが設置され、それぞれのテーマの検討結果を技術委員会に持ち寄り、各社間の調整や COSE としての方針決定を行った。なお、開催回数としてはソフトウェアエンジニアリング部会が最も多く100回を超えている。

このような密度の濃いコミュニケーションが COSE としての一体感を強め、プロジェクトを成功に導いた大きな要因でもある。

Section 2

プロジェクト計測計画の概要

2.1 主要な計測項目

プロジェクト計測計画では、下記に示す6つの計測技術と方法を採用した。

EPMによる計測と分析（第2部1章参照）

EPMを用いて、開発プロセスとプロダクトの基本情報を取得した。そのために、COSEでは、各社の開発管理環境を構成管理システムはCVS、障害追跡システムはGNATSに統一した。EPMにより、たとえば、プログラム行数の推移、累積障害件数、残留障害数の推移、障害平均滞留時間の推移、社間メール件数の推移、これらとチェックインのタイミングなどの情報をグラフなどの可視的な形で提供することができた。

レビュー記録の収集とEPM Pro＊[注]による測定と分析

専用の電子フォームを用いて、レビュー記録について、基本設計と詳細設計に対して情報収集した。またEPM Pro＊を用いて、レビュー記録の分析、ファイル更新に関するさまざまな分析などCVSの情報を用いたより高度な分析を試みた。

注：EPM Pro＊
EPMの拡張機能。EPMには、各種の機能拡張用のインタフェースが装備されている。先進プロジェクトでは、このインタフェースを利用してデータ分析機能などさまざまな機能拡張が試みられ実際に適用された。EASEプロジェクトでは、プロジェクト終了後これらの機能を集約しEPM Pro＊として研究に供する体制を整備した。以降本書ではこのEPM拡張機能をEPM Pro＊と呼ぶ。[A.9]

コードクローン検出・分析ツールCCFinder/Gemini[注]（第２部２章参照）

コードクローンはソースコード中の類似するコード片のことである。このコードクローンの分布状況、含有率などからプロダクトの性質を推し量ることができる。先進プロジェクトでは提供されたソースプログラムをCCFinder/Geminiと呼ぶツールを用いて、ソースコード内のコードクローンの含有状況、含有率の分析を実施した。ソースコード内のクローンの含有状況は各種のメトリックスでも表現され、ソースコード全体での含有状況は散布図と呼ばれる図で可視化された。

ベンチマーク・データベースと協調フィルタリングツールの応用による分析

SECでは、プロジェクトに関する情報として400項目以上におよぶデータを専用の入力フォームで収集しており、これをベンチマークデータと呼ぶ。先進プロジェクトにおいても組合員企業におけるソフトウェア開発プロジェクトに対して同様の情報を収集した。この項目は国内の先進企業で収集されてきた項目や、ISBSG（International Software Benchmarking Standard Group）[注]が収集しているデータ項目を考慮して、経済産業省の定量データ分析部会における活動で検討されたものである。この集計データは毎年『ソフトウェア開発データ白書』[注]としてSECより出版されている。先進プロジェクトでは、基本設計終了時に基本設計工程の実績値とプロジェクト全体の計画値を集め、プロジェクト終了時に残りの実績値を収集した。

EASEプロジェクトで開発した、欠損のあるデータセットから類似のデータを選び出す協調フィルタリングの技術を応用したツールMagiを用いて、ベンチマークデータを分析した。SECに収集されている約1000プロジェクトのベンチマークデータ（以降「SEC定量データベース」という）を参照し、基本設計終了時のデータから類似するプロジェクトを探し出し、プロジェクトの途中でそのプロジェクトの将来を予測する試みを行った。

注：CCFinder/Gemini
CCFinderは、神谷年洋氏（産業技術総合研究所）が開発したコードクローン検出ツール。Geminiは、コードクローンを視覚的、対話的に分析するツール。

注：ISBSG
世界中からプロジェクトデータを収集しベンチマークデータベースを構築、運用してその情報を提供しているNPO。
http://www.isbsg.org/

注：ソフトウェア開発データ白書 2007：IPA/SEC 日経BP社 2007-8

チェックシートを用いたリーダへのヒアリング調査

SECは、プロジェクトの進捗や潜在するリスクの見えにくいソフトウェア開発プロジェクトの可視化を図るために、経済産業省の「プロジェクト見える化部会」を通じて研究を進めてきた。部会の成果として、要件定義が終了し、基本設計がある程度進んだ時点を想定し、プロジェクトのリスクを把握するための上流工程用チェックシートと、結合試験を実行中の時点を想定し、プロジェクトの失敗に結びつく事象を早期に検出するための下流工程用チェックシートの2つのチェックシートを作成した［D.2、D.3］。各チェックシートは2つのシートから構成され、1つは40項目程度からなる「自己評価シート」で、もう1つは第三者の専門家による2時間程度のヒアリングのための80項目程度の「ヒアリングシート」である。先進プロジェクトでは、組合員企業各社のリーダ、あるいはサブリーダを対象にこのチェックシートによる自己評価とヒアリングを実施した。このヒアリング調査を通して、プロジェクトに潜むリスクや管理体制などEPMのような自動収集ツールでは収集できない多くの情報を得ることができた。

プロジェクト会議への継続参加による情報の収集と分析への反映

プロジェクトの進捗に関する各種の情報を得るために、SECの研究員がプロジェクトを通して各種の会議に参加し、自動収集では得られない情報を取得した。また、必要に応じて工程区切りとなる会合にEASEプロジェクトの研究員も参加した。

2.2 計測データの収集法

プロジェクト計測計画では、次のような収集法が用いられた。

1) 計測データの収集・分析の場として、SEC内に機密室を確保し、ここにネットワークから独立したサーバを設置し、主要な分析作業を行うこととした。さらにEASEプロジェクトの運営するエンピリカルソフトウェア工学ラボ（大阪・千里）内にも機密室とサーバ環境を確保しEASEプロジェクトの研究員による分析作業を行った。
2) 開発を担当する5社にCVSとGNATSを導入し、運用してもらった。ソースコードを含むCVSのレポジトリ、GNATSのデータファイルを原則毎週媒体で送付、または

Webベースの情報共有システムにファイル転送してSECに提出してもらい、機密室のサーバに格納した。Webベースの情報共有システムから機密室にデータを移動する場合には、暗号化してから媒体に格納し、媒体を機密室に持ち込んだ。

3) レビュー記録は問題記述票という書式で、奈良先端大学提供のExcelマクロの電子帳票に記入してもらい、同じく媒体またはファイル転送で提供してもらった。問題記述票の記述項目、形式はプロジェクトで決めたものである。この帳票は1件1葉ではなく、1枚の帳票に15件までのレビューコメントを記入する。電子帳票のExcelマクロはこれを分解し集計を行う。

4) 障害追跡システムのGNATSは障害管理項目、すなわち障害票の形式を定義でき、開発プロジェクトではこれを定義して用いる。今回のプロジェクトでは、組合員各企業と計測・分析チームの十分な協議によって管理項目を決定した。

5) CVSの運用ルールについて、関係者間で十分に協議して定めた。

6) メールについても運用ルールを定め、開発プロジェクトの技術に関する社間メールを分析対象とした。

7) SECの機密室のサーバにEPMと関連ツールを導入した。また、フロントエンドにWindowsパソコンを接続し、Windows環境で走行するコードクローン分析ツールのCCFinder/Gemini、およびEPM Pro*を装備した。

8) ベンチマーク・データベース用のデータは、SECの提供するExcelフォームに記入する形でまとめられた。調査は2回行い、協調フィルタリングツールを用いた工数予測について評価した。

9) チェックシートによるヒアリングはプロジェクト見える化部会のメンバがチェックシートをプロジェクタで映しながらインタビューを行い、評価を行った。自己評価シートはヒアリング前に記入して提供してもらった。

10) 図1-1で示したSE部会などの会議にSEC研究員が参加し、情報と資料を収集した。

11) 組合員企業のうち、希望によって2社でEPMが導入され、自社内で評価された。

Section 3

全体の工程と主な計測・フィードバック契機

3.1 フェーズ１の工程区分と内容

計画全体の開発スケジュールを図1-2に示す。

要件定義は、JARIで策定された要件定義書をCOSEで再定義して開発の基盤文書を作成した。各社で分担して検討し、共同でレビューして文書を確定した。技術研究組合の技術委員会で承認のプロセスをとった。

基本設計は、書式を皆で合意したのち、この書式に沿って各社で分担して作成した。共通機能と個別機能に分かれ、共通機能は共同のレビューを経て、技術委員会で承認された。個別機能は分担する各社に任され、開示が制限された。プロジェクト会議では、各社自己申告レベルで、ページ数によって作業進捗が報告された。

詳細設計も共通機能と個別機能に分けて基本設計と同様のプロセスを経て作成された。個別機能は分担する各社に任され、開示が制限された。プロジェクト会議では、各社自己申告レベルで、ページ数によって作業進捗が報告された。

プログラム設計は、製造工程に含めて考えられ、各社に任され、内容、方法とも開示が制限された。

製造工程は各社にまかされた。自己申告レベルの進捗率が報告された。

社内結合試験も製造工程と同様に進められた。

社間結合試験以降の工程は、データセンター内にサーバ群を設置して共通の試験環境を整備し、各社が開発したオブジェクトコードを持ち寄り、ライブラリ管理の下で運用した。作業は各社の担当者が集まることができるオフィスを用意し、データセンターと専用線で結ばれた端末を使って行われた。社間結合試験以降のプロジェクト進捗管理はPMのリーダシップのもとで、技術研究組合各社の間で適宜試験設備と情報を共有しながら

共同作業で進められた。

社間結合試験工程以降の障害追跡システムは、共通のGNATSをこの試験設備に併設して準備し、SECではそのデータの提供を受けた。

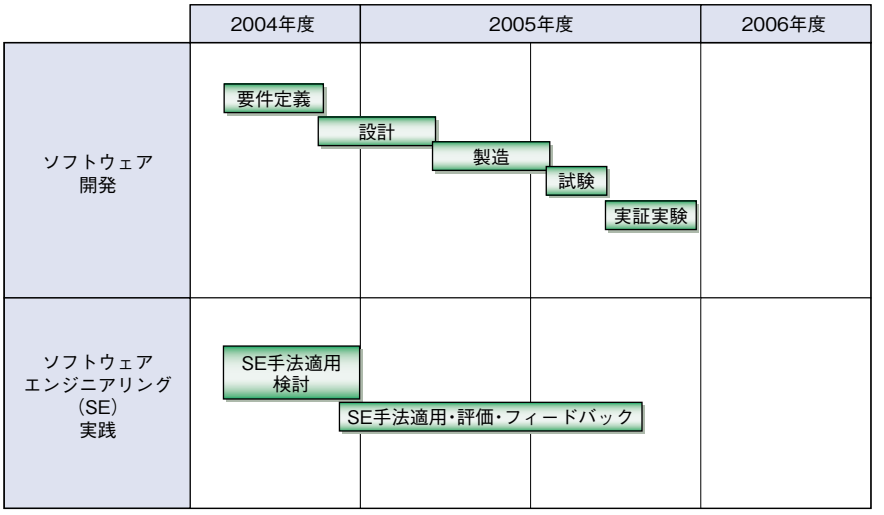


図1-2　フェーズ1の開発スケジュール

3.2 主なプロジェクト計測・分析体制

プロジェクトに関するデータの計測と分析、および分析結果のフィードバックは適用する技術に合わせて次のような分析グループを編成して実施した。

EPM分析グループ

EPMおよびEPM Pro*をおもな道具として、ファイル更新状況、障害発生状況、メール送信状況に関する計測と分析を行った。またレビュー記録を分析した。コードクローン分析についての支援もした。

コードクローン分析グループ

ソースコードからCCFinder/Geminiを用いてコードクローン分析を行った。

チェックシート分析グループ

自己評価シートおよびヒアリングシートにより、組合員企業各社ごとのプロジェクトにおけるリスクの明確化やプロジェクトを失敗に落とし込む問題の早期発見を試みた。また、ヒアリングによって見えたプロジェクトの課題をまとめ、課題に対するアドバイスをフィードバックした。

ベンチマークデータ分析グループ

SECの定める400項目を超すベンチマークデータ集計表でデータ提供を受け、1次分析を行い、さらに協調フィルタリングの技術を用いてSECの過去データの中から類似プロジェクト群を抽出し、見積もり予測を行った。

プロジェクト会議参加グループ

主要なプロジェクト会議に継続的に参加し、上記では収集できないプロジェクトコンテキスト情報を収集した。公式のプロジェクト会議における、自己申告レベルの進捗報告情報を得た。また、工程の区切りごとのPMと各社別の品質評価会議に参加し推移を観察した。

組合員各社へのフィードバックは、上記各グループ（プロジェクト会議参加グループを除く）のテーマごとに行った。また、それぞれについて事後、各社に簡単なアンケート調査を依頼し、主として計測とフィードバックの有効性、あるいはその方法について回答を得た。

3.3 フェーズ1のおもな計測契機

プロジェクト会議

SE部会は、要件定義の工程からフェーズ1のプロジェクト終了まで継続して毎週開催された（引き続き休みなくフェーズ2でも開催された）。

SE部会では詳細設計工程以降、各社より自己申告レベルで進捗報告があった。

レビュー記録収集

基本設計と詳細設計レビューについて、2005年6月上旬にレビュー記録の提供があった。

EPMツールの入力となる構成管理（CVS）データおよび障害管理（GNATS）データ収集

2005年7月から、各社ほぼ毎週、2006年1月まで継続してデータが提供された。

社間結合試験以降のGNATSデータについて、工程中適宜提供を受け、工程終了時、全障害クローズ時点で提供を受けた。

メールデータ

開発プロジェクトのメーリングリストによる社間メールについて常時全数提供を受けた。

チェックシートによるインタビュー

2005年8月下旬に実施した。

ベンチマークデータ収集

基本設計終了時にプロジェクト属性、プロジェクト全体の計画値、基本設計終了時点までの実績値を得た。プロジェクト終了後、全体の実績値を得た。

品質評価会議

2005年9月上旬（1回目）、9月下旬（2回目）、2006年1月下旬（3回目）を実施。

いずれの会合もSECメンバが参加し、EASEプロジェクトの大学側研究員もオブザーバの形で参加した。

3.4 おもな分析とフィードバックの契機

レビュー記録分析

基本設計分、詳細設計分あわせて2005年6月中旬に集計、グラフ化し、分析を加えたものを各社に社別に提示した。PMには全社分、モジュール名を伏せて示した。分析結果のフィードバックは8月中旬になってEPM分析結果とあわせて意見交換会の形で行った。

EPM分析

分析活動は最初のデータが揃った2005年7月中旬から9月中旬まで、各社当たり6回、そしてまとめの分析を2006年1月末に実施した。分析結果のフィードバックは、2005年8月中旬、9月中旬、2006年2月上旬に行った。これらのフィードバックに対して、3月上旬に各社へアンケート調査を実施した。分析結果は全社分をPMに対して示し、開発各社へはPM同席の状態で個別に示した。分析結果の提示は意見交換会の形で行った。

コードクローン分析

クローン分析は2005年7月下旬、9月中旬、2006年1月下旬の3回実施した。いずれもEPMによる分析と同期して行い、分析作業ごとにフィードバックを実施した。PMには原則ファイル名を秘して示し、開発各社にはPM同席の状態で個別に示した。分析結果の提示は原則、意見交換会の形で行ったが、資料提供だけの場合もあった。

これらの分析に関して2006年3月上旬に各社に簡単なアンケート調査を行った。

チェックシートによる見える化

チェックシートによるヒアリングは、まず2005年8月中旬に自己評価シートを送付し各社より回答を得た。SECではその集計・分析結果を持ち、8月下旬に各社を訪問して3時間弱のインタビュー調査を実施した。

9月下旬から10月上旬にかけて、個別に各社へフィードバックした。フィードバックは意見交換会の形をとる社と資料提供だけの社があった。PMへは個別にインタビュー調査するともに、全社の分析結果を提示した。

ベンチマークデータの分析

基本設計時までのデータの分析について、2005年9月中旬に全社共通事項について合同の概略報告会をもち、10月上旬に各社に個別に文書で提供した。11月上旬に各社に簡単なアンケートを実施した。

プロジェクト終了後全体の実績データの提供を受け分析した。

SE部会での進捗報告

SE部会はPMおよび開発担当社合同で実施された。

詳細設計工程では、開発各社より開発対象の機能別に設計書のページ数について、予定、実績、累積、ページ数からの進捗率が報告された。

製造工程では、モジュール別にクラス数で、予定、実績、累積、進捗率が報告された。

単体試験工程では、モジュール別に試験項目数の予定、実績、累積、消化率、発生バグ数、修正障害数が報告された。

品質評価会議での品質報告

品質評価会議はPMが主催し、開発各社個別に実施された。

開発各社からPMに対し、工程別、大きな機能項目ごとに、試験成績書兼検査成績書が報告された。記載事項の中に、試験規模（KStep）、試験項目数、障害件数、試験密度（件／KS）、障害検出密度（件／KS）、品質区分がある。このうち、試験密度と障害検出密度については目標値と許容上限、許容下限と実績値が示された。

3.5 フェーズ2の概要

フェーズ1に引き続き翌年度フェーズ2の開発が進められた。

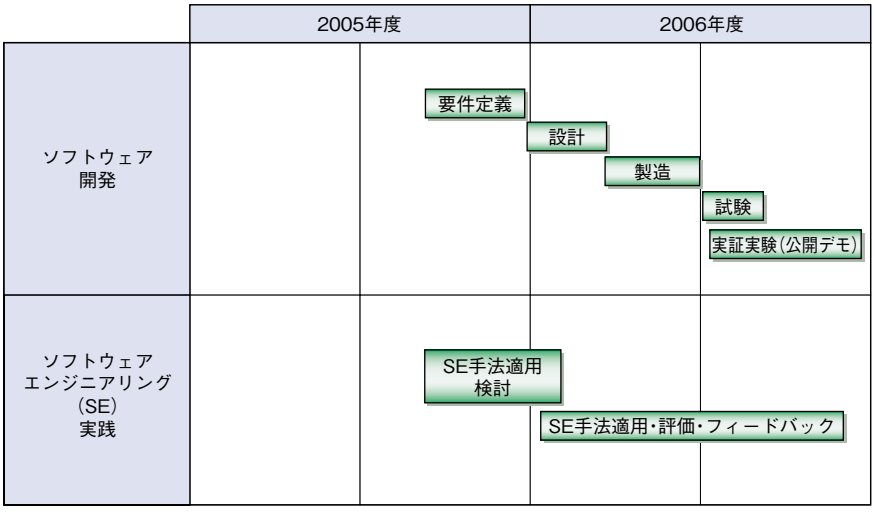


図1-3　フェーズ2および公開デモ向け開発スケジュール

フェーズ1の開発は、各企業のノウハウを持ち寄り、それぞれサブシステムを分担して、これを結合して全体の機能を実現した。全体に共通な機能として機能間のデータ受け渡しの基盤となるデータベース機能を1つだけ実現し、そのインタフェースを共通に規定した。そのため、この構成では、類似の機能がノウハウを提供した企業の数だけ並走する部分がいくつかあった。

フェーズ2の目的はこうした構成の冗長性を廃し、重複する機能の共通化を図り、さらに共通プラットフォームとして、機能ごとのプログラムの置き換えを可能にする構造整備と、さらに実用化を目指した機能拡充にある。このため、フェーズ1では無かった社間のプログラム供給を、フェーズ2では一部オブジェクトレベルで実現した。フェーズ2の機能実現に当たっては、可能な限りフェーズ1で実現した機能の転用を図った。

フェーズ1のプロジェクト計測とフィードバックは、それなしでもプロジェクト運用に支障の生じない、いわばモニター型で計画された。すなわち、プロジェクト運営は従来型の自己申告方式の予実管理で行われ、この運用にできるだけ擾乱を与えない条件でプロジェクト計測を進め、計測と分析の結果をプロジェクトマネージャ（PM）と組合員各社

の開発リーダ、サブリーダに各社別に切り出して提供された。分析データはプロジェクト運営にとっては参考情報の位置づけだった。

フェーズ2においては、フェーズ1での計測とフィードバックのプロジェクト運用への有効性に関する評価が高かったために、これをプロジェクト運用へより一層強く組み込むこととし、週次のフィードバックを実現した。

具体的には、下記の計測と分析、フィードバック作業を実施した。

設計書の分析

設計書を構成管理システム（CVS）に登録し、新しい分析方法を探究する。

フィードバック周期の短縮

データの収集はフェーズ1と同様に週次とし、さらに分析とフィードバックの契機も週次とする。通常の週にはドキュメント送付のかたちで分析結果をフィードバックし、工程区切りの品質評価会議で分析データを共有しながらの意見交換を行う。EPMによる分析はさらに内容を深める。

年度間のコードクローン分析

コードクローン分析において、フェーズ1との比較を行う。

SEC定量データベースと協調フィルタリング技術を用いた見積もり予測

フェーズ1と同様にSEC定量データベースをもとにして、各社のベンチマークデータ（基本設計終了までの実績とそれ以降の計画値）からプロジェクトの全工数を予測する。

相関ルール分析

各社の障害追跡システムに格納された障害データと構成管理システムに登録された障害に対応する修正の情報を分析することにより、障害情報の項目と障害の修正との間での関連性（ルール）を分析する。

スキル分析

ＩＴスキル標準に基づいたスキル診断を実施し、各社ごとにプロジェクトメンバ全員の

スキルを集計してチームとしてのスキルレベルを分析する。この時、ソフトウェア・エンジニアリングに関するスキル項目を詳細化し、ソフトウェア・エンジニアリングに関するスキル分析も行う。

3.6 公開デモ版開発の概要

プローブ情報プラットフォーム構築事業の成果を広く発表することを目指して、フェーズ2に引き続きその機能拡張版として、「公開デモ」版の開発が進められた。

開発方針として、フェーズ2の機能をそのまま複製し、そこに次の機能を付加して公開デモ版とした。

- 情報の配信機能
- 情報の提供機能

プロジェクト計測と運営の関係

公開デモ版の開発では、フェーズ2でのプロジェクト計測とフィードバックの形態はそのまま引き継いだが、それまでの自己申告による進捗報告は廃止し、計測データだけで進捗管理を進める体制とした。この段階で、EPM等によるプロジェクト計測とフィードバックはプロジェクト運営に不可欠な活動として組み込まれた形態となった。

走行実験

column

フェーズ1、フェーズ2、それぞれにおいて、作成されたソフトウェアをユーザの視点から評価するための走行実験が行われた。

フェーズ1の走行実験では、タクシーにGPS端末とビデオカメラを搭載し、決められた周回コースを走行し、位置と時刻を記録。同時に、周回コース上の何点かにビデオカメラを設置して定点観測を行い、コース上の車の移動時間を割り出した。これらの情報と開発したシステムの出力結果を照合して、システムの出力結果が正当かどうかを評価した。

フェーズ1の走行実験は、2005年12月の冬晴れの下、2日間に渡って実施された。1日目は昼食時と夕方の通勤時間帯、2日目は夜明け前から準備を行い、朝の通勤時間帯と昼食時を狙って測定を行った。フェーズ2では、さまざまな角度からの評価を行うためにデータ収集用の走行実験が2006年8月の炎天下（2日目は雨だった）から開始されるなど、数回に渡って実施された。フェーズ1の走行実験の分析結果をフェーズ2の開発に活かし、フェーズ2の走行実験の分析結果を公開デモ向け開発に活用した。

Section 4

データ収集・分析結果（フェーズ1を中心に）

4.1 基本設計、詳細設計のレビュー記録分析

レビュー記録は、開発各社の開発作業の特性を垣間見る最初の機会となった。レビュー記録は組合各社で協議、合意した問題記述票により集められた。記述票には約30項目の記述項目があり、1枚の記述票にレビューでの指摘項目を基本的に1件1行、最大15件まで記入する。この記述項目を1件ごとに分解し集計するためのExcelマクロを埋め込んだ奈良先端大学提供のExcelの電子帳票によって収集した。

設計書のレビュー記録の分析から、各社のレビューへの姿勢をある程度読み取ることができる。

図1-4に基本設計工程と詳細設計工程のレビュー記録の分析例を示した。図では、各社別にレビュー対象としたドキュメントのファイル数、ページ数、レビューのために稼動した延べ人数、延べ時間数、レビューで指摘された項目数（欠陥数）を比較している。レビュー対象としたドキュメントの分量に大きなバラツキがあることなどがわかる。

図中に示す「A社」「B社」などの名称は、具体的な社名に対応するものではありません。ある図にある「A社」と別の図にある「A社」は同じ会社を意味しません。

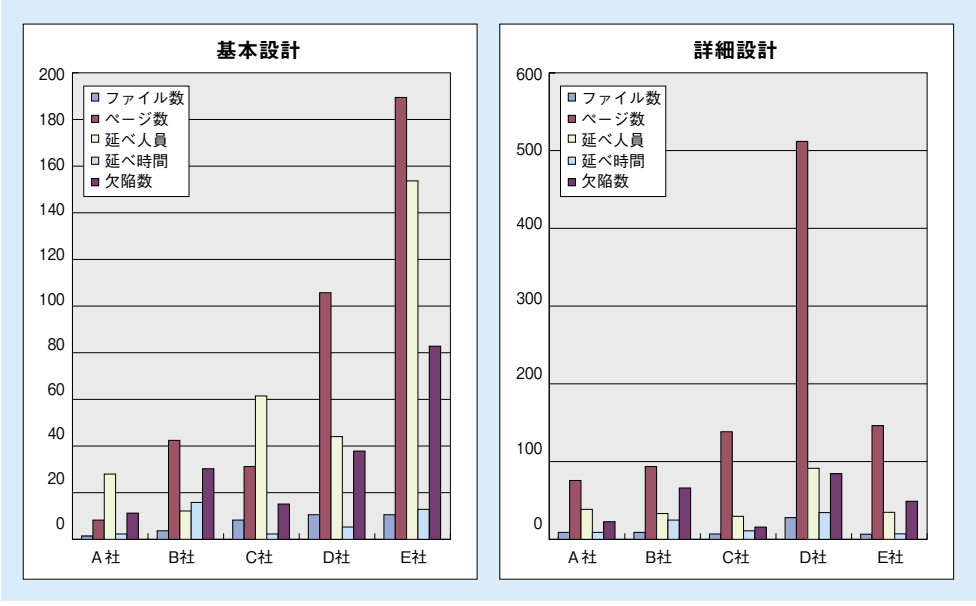


図1-4　レビュー記録（企業別問題記述票）分析（松村知子）

4.2 EPMによるプロジェクト・モニタリング

EPMを適用してソースコード行数を計測し、その推移を可視化した。ソースコード行数の推移の計測から、各社の開発規模感、開発着手時期や立ち上げ時期の相違、さらに各社でのソースコード管理の粒度について一目で比較・観測することができた。

図1-5に開発したソースコード規模の推移を全社俯瞰できる形で示している。プロジェクトの開始時期は各社同じであるが、各社の開発活動のスタート時点がばらばらで、五月雨式にスタートしている様子がわかる。また、スタートから3か月弱の時点で開発規模が各社間で大きく異なり、特にE社の規模が突出していることがわかる。このE社は大規模な流用母体を持っていること、そしてソースコードの管理方法が他社とは異なることが推測された。

またグラフの変化の様子から各社の管理体制を推し量ることができる。すなわち、細かい変化となっている場合は管理の粒度（きめ細かさ）が細かく、現場に近いところで管理が行われている可能性が高い。一方、粗い階段状になっている場合は現場から比較的遠い

階層で、粒度の粗い出荷管理的な管理が行われている可能性が高いことが読み取れる。

すなわち、ソースコード規模推移の階段が細かい場合は、データ提供のもととなる構成管理システムが開発現場に存在し、細かい単位でソースコードの作成が管理されていることを示唆している。ソースコード規模推移の階段が粗い場合は、データ提供のもととなる構成管理システムが日々の開発現場に無く、ある程度開発が進行したのち一定の単位で登録が行われている、いわゆる出荷管理システムとして機能していて、出荷単位となる開発が終わるまでの期間は、日々の開発状況はブラックボックスになっている可能性が考えられる。

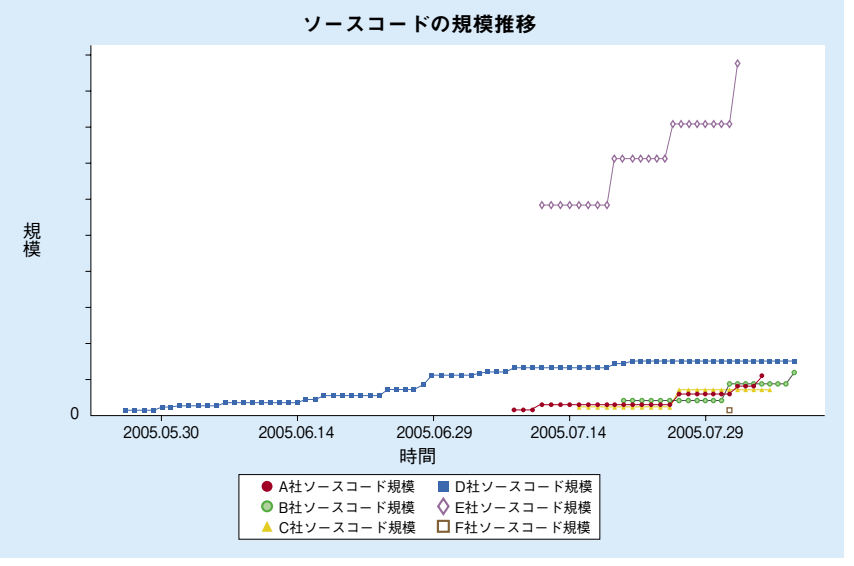


図1-5　ソースコード規模推移（プロジェクト立ち上げ時、全社分）（松村知子）

図1-5は全社を俯瞰して見るためのものであるが、1社のある期間を拡大して示したのが図1-6である。グラフ中の縦線はCVSへのチェックインの契機を示している。この社が開発した機能のリリースとその規模を把握することができる。ここでも、この社のソースコード管理の粒度を把握することができる。図1-6は粗い階段状になっており、リリースとリリースの間は現場の管理に任されていることが推測できる。

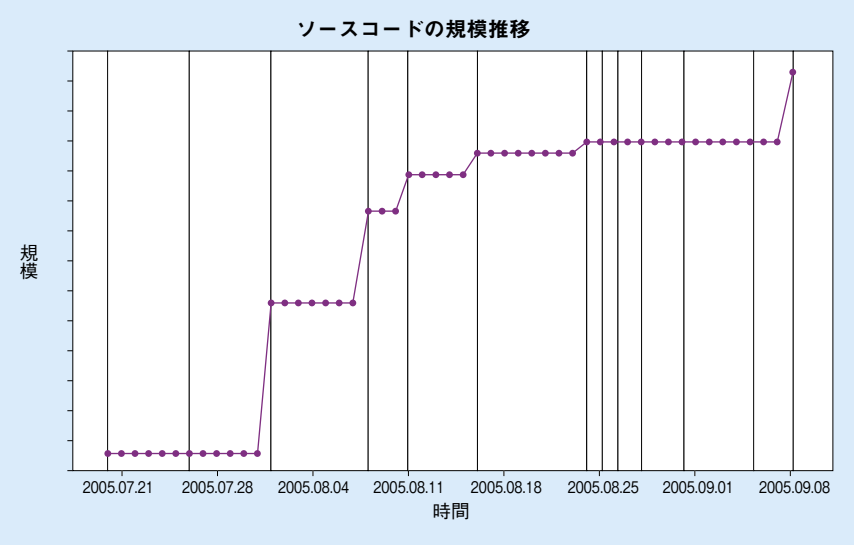


図1-6　ソースコード規模推移（1社例）（縦線：チェックイン契機）（松村知子）

EPMの拡張分析機能を用いることで、ソースファイルの更新履歴を追うことができる。ソースファイル更新履歴の観測から、ウォーターフォール型の開発、試行錯誤型開発などの各社ごとの状況を明確に把握できた。また、工程後半でのソースコードの安定度を把握できた。たとえば、仕様変更や修正のインパクト、工程途中でのコーディング標準の変更などのインパクトを明示的に把握できた。図1-7～図1-10にファイル更新履歴に関する分析グラフの例を示す。

図1－7は1社のある期間のファイル数とソースコード行数の推移を示している。この期間、順調に開発量を増やしている様子が見える。そして開発はまだ安定状態には入っておらず、この期間ではまだ終了していないことがわかる。

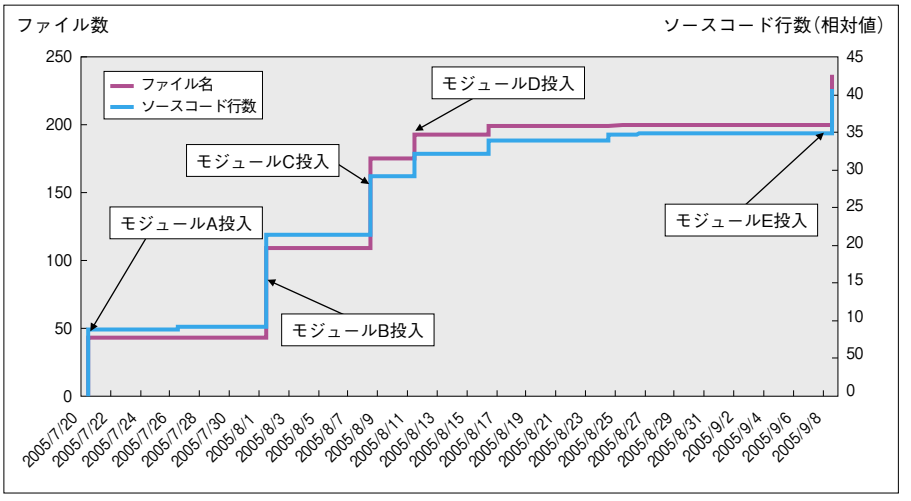


図1-7　ファイル数と行数の推移（1社例）（松村知子［A.10］）

図1-8はファイルやソースコードの変更量に着目した推移を示している。比較的大きな変更と考えられる5行以上削除された変更の割合（FCL）、追加・削除された行数のうち5行以上削除された行数の割合（LCC）、5行以上削除されたファイルの割合（FCB）、の遷移をプロットしている。グラフはこの期間、大幅な修正の後に、さらに2回障害対応の修正があったことと対応している。

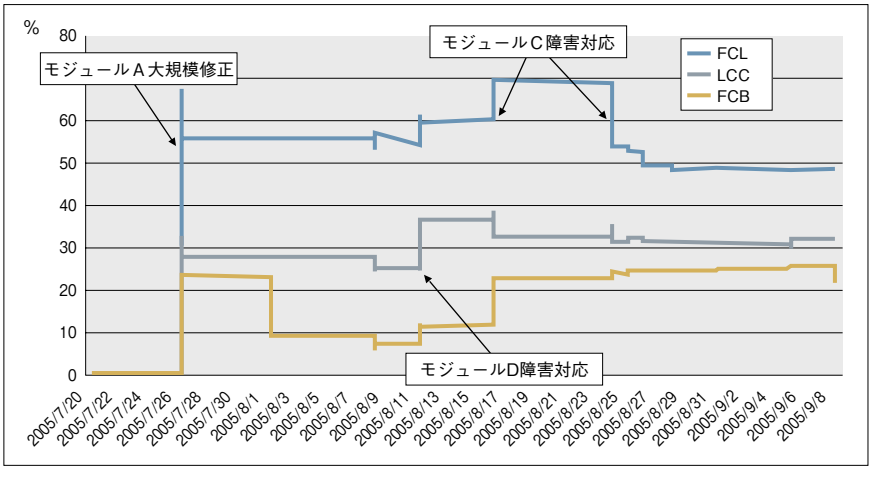


図1-8　変更頻度・変更規模・変更網羅性（1社例）（松村知子、森崎修司、玉田春昭［A.10］）

LCC　：全追加＋削除行数のうち、5行以上削除された行数の割合
FCL　：5行以上削除された変更の割合
FCB　：5行以上削除されたファイルの割合

図1-9は1社のある期間の、ファイル更新の別の側面に着目している。2回以上更新されたファイルの割合（FCM）、5%以上更新されたファイルの割合（LCC）、2人以上で変更されたファイルの割合（OWN）の推移をプロットしている。障害修理等インパクトのある時跳ね上がり、開発にともなう全体量（母体）の増加にともなって減少するパターンを観測できる。この例では修正と新規モジュール投入の動きを追うことができる。そしてまだまだ安定期ではないことがわかる。

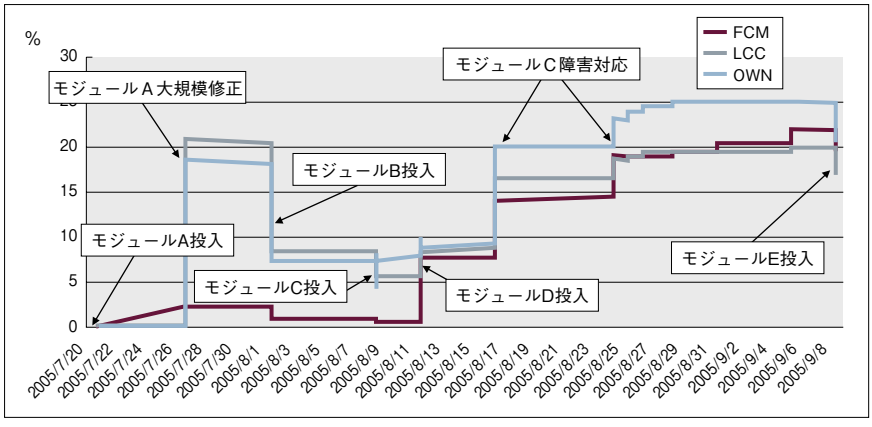
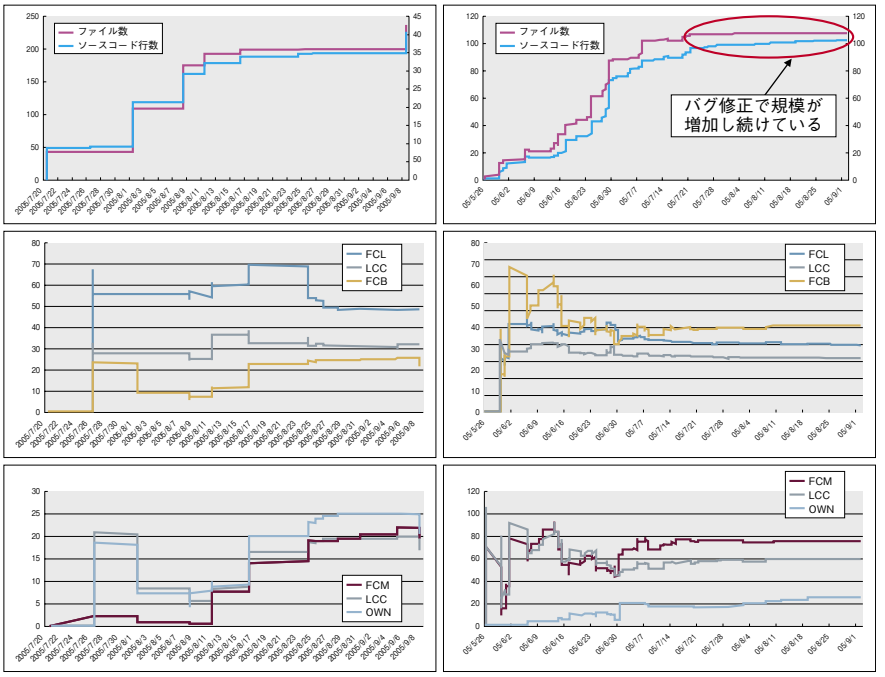


図1-9　変更者数・変更率・変更回数（1社例）（松村知子、森崎修司、玉田春昭［A.10］）

FCM　：2回以上変更されたファイルの割合
LCC　：5%以上変更されたファイルの割合
OWN：2人以上で変更されたファイルの割合

図1-10はこのようなファイル更新履歴に関する分析グラフを企業ごとに並べて俯瞰したものだ。図1-10はA社とB社のある時間の推移の様子を並べた例である。A社の例は図1-7から図1-9と同じものである。B社は前半に試行錯誤（カット＆トライ）型の開発を行っている。これはB社が今回の業務領域の開発について深い経験が無く、新しい方式を確認しながら開発したことを反映している。B社ではこうした開発方法に対して、開発期間や専門家の確保など一定のリスク管理を行って開発した。またB社のデータは開発現場から直接提供されたもので、開発管理の粒度が細かいことが推測される。B社に比べA社のデータは1段上の管理階層から提供されたもので、B社に比べ管理の粒度が粗いことが想定される。



A社　　B社

図1-10　ファイル更新履歴分析（社間比較例）（松村知子、森崎修司、玉田春昭［A.10］）

このように、1つのプロジェクトを分担（サブプロジェクト）し、並行に開発を進めている各社の分析グラフを俯瞰することで、プロジェクト全体の進捗をある程度推し量ることができる。

図1-10に示すようなファイルの更新状況の推移は、必ずしもそのパターンのみから特別なリスクやアクションを必要とする契機を捕捉することを目的としたものではない。これらのグラフに示されるような開発プロセスに対して、それぞれにふさわしい品質や工期が管理されていることが確かめられることが重要である。たとえばA社の場合は、これらのグラフには表れない開発階層において作られたモジュールが順次積上げられ、その都度試験を実施し、検出された障害の対応を進めるというプロセスとなっている。この場合、各モジュールの品質が確保され、内部試験での障害対応が円滑に進められていることが確認できれば良い。B社の場合は、試行錯誤型開発を行うための開発期間がきちんと確保され、当該分野の専門家や経験者へのアクセスが確保されるなどのリスク対策が採られていれば良い。

また、これらの計測と可視化した情報のフィードバックの効果は、可視化された情報から直接読み取れる事象への対応だけではない。ブラックボックスの多い管理階層の中での、ある視点からの抽象化された情報を開発者や管理者の間で共有することによって、チームのモチベーションの向上、見られていることへの肯定的なプレッシャー、見えていることによる心理的な安心感などがもたらされ、これらがプロジェクト全体にプラスの作用をあたえる。この効果は、今回の試みを通して定性的に観測され、また開発リーダへのアンケート調査によっても、こうした計測とフィードバックの効果に肯定的な回答が集まったことである程度実証された。

EPMが障害追跡システムから収集する情報の分析で、障害の発生状況、その種別や原因、修正状況をビジュアルに把握できた。また障害に関する混入工程・発見工程・発見すべき工程の分析から問題を潜在させている工程の推定ができた。障害修正に要している時間の推移の分析からプロジェクトの状況を推し量ることができた。図1-11～図1-18はEPMと関連ツールによる障害関連の分析グラフの例である。

図1-11は、累積の障害発生件数を全社分をグラフ化したものである。ソースコード行数の推移を表したグラフと同様、各社のスタートと工程の立ち上がりの相違がわかる。また、発生件数から試験の進捗の度合いの相違がわかる、またこのグラフでは、まだどの社も安定期に入っておらず開発と試験の途中であることがわかる。

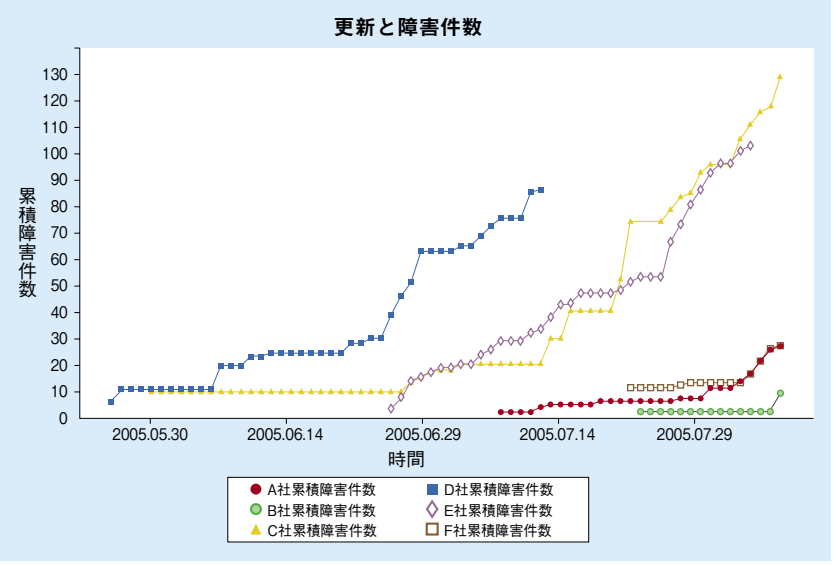


図1-11　障害件数の推移（全社分）（松村知子）

図1-12、図1-13は開発担当社のうちの2社について、ある時期の累積障害件数（●）、未解決障害件数（◇）、平均障害滞留時間（□）をプロットしたものである。試験の進行に伴って障害が検出され、未解決障害が残り、平均障害滞留時間が伸びた、そして収束に向かっている様子が見える。PMと開発グループの間でのこのようなグラフを共有することで、それぞれの企業の単体テストの期間、夏休みなどの作業状況を把握する支援になり、各社各様の状況を把握した形での深い議論に役立つ。

図1-12は、テストの開始に伴って累積障害件数、平均障害滞留時間とも伸びが見られ、一旦試験作業が休止状態となった後、障害対応によって未解決障害が減少し収束に向かっている様子がわかる。

図1-13は当初、障害対応をしていない模様で、平均障害滞留時間が伸びている。その後障害対応を行って、平均障害滞留時間が減少に転じるが、一方、試験作業が本格化し、累積障害件数の増加とともに未解決障害件数も増えている。しかしながら、そうした中で障害対応が行われ平均障害滞留時間の伸びが抑制され、試験も障害発生も収束に向かっている様子がわかる。しかしながらこの報告期間にはまだ未解決障害があり、完全には収束していない。

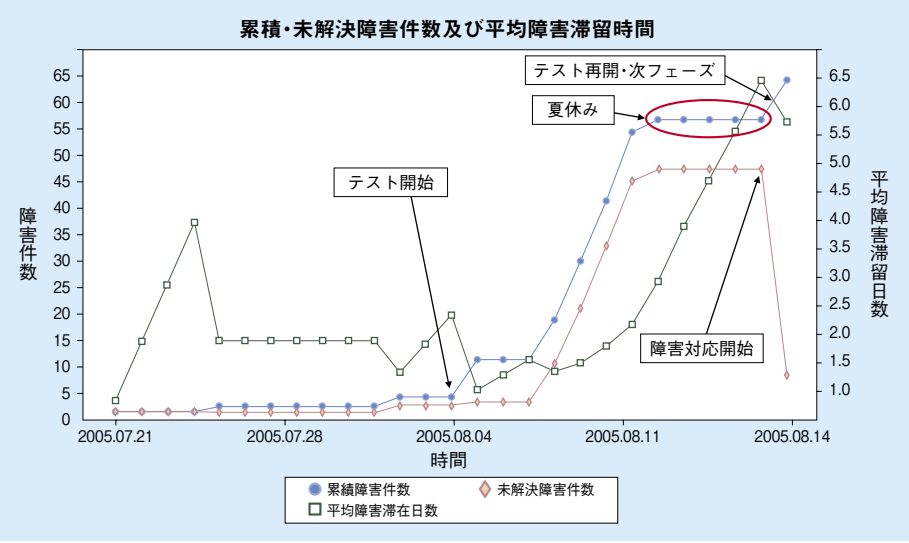


図1-12　障害件数と平均障害滞在日数の推移（X社例）（松村知子）

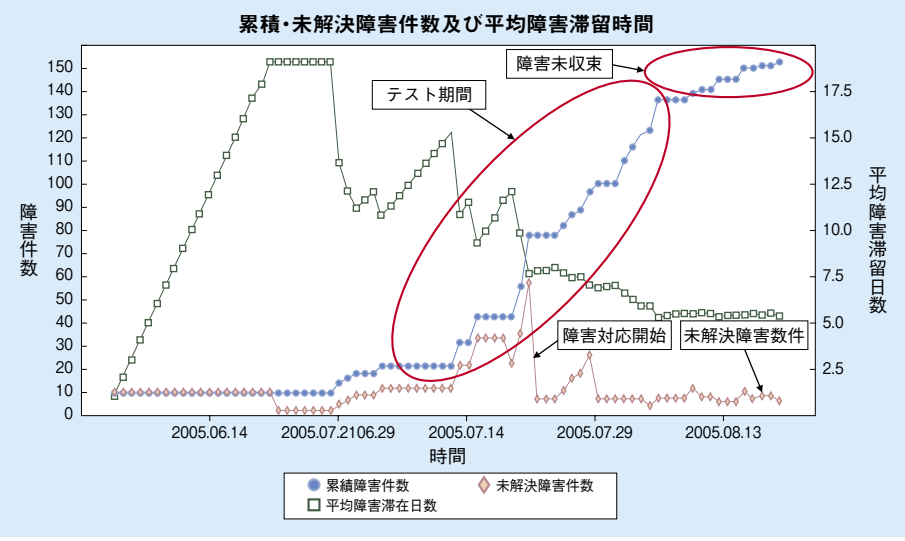


図1-13　障害件数と平均障害滞在日数の推移（Y社例）（松村知子）

図1-14、図1-15は開発担当社のうちの2社について、ある時期の障害件数、そのうち重要度の高い障害数、未解決な重要度の高い障害数、重要度の高い障害の平均滞留時間をEPM Pro*によってプロットしたものである。試験の進行に伴って障害が検出され、

未解決障害が残り、平均障害滞留時間が伸び、そして収束に向かっている様子が見える。PMと開発グループの間でのこのようなグラフを共有しての質疑が、それぞれの企業のテストの進行状況、作業の休止、障害対応などの作業状況を把握する支援になる。

図1-14は、テストの開始に伴って全障害数、平均障害滞留時間とも伸びが見られ、一旦試験作業が休止状態となった後、障害対応によって未解決障害が減少し収束に向かう様子がわかる。平均障害滞留時間が波打っている期間、試験に伴う障害の発生とその解決という作業が続き、平均障害滞留時間が延びるのを抑制している様子がよくわかる。後段、未解決障害が解消し試験も収束した様子がうかがえる。

図1-15は、試験の規模（あるいは開発規模）が図1-14の例より小さく、障害をある程度まとめてからその解消を図る作業スタイルが見て取れる。この場合も未解決障害が解消し試験作業が収束している。

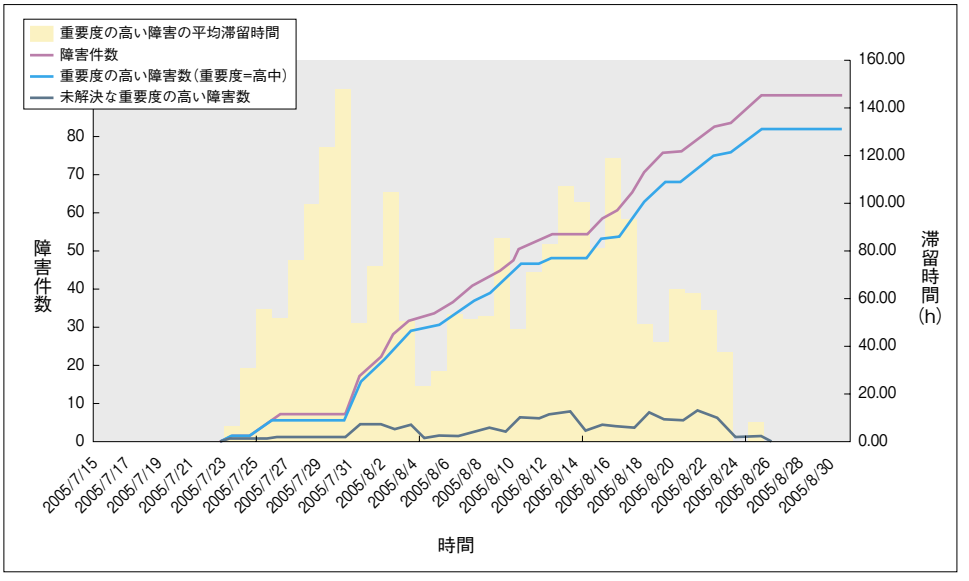


図-14　障害件数の推移（X社の例）（松村知子、森崎修司、玉田春昭［A.10］）

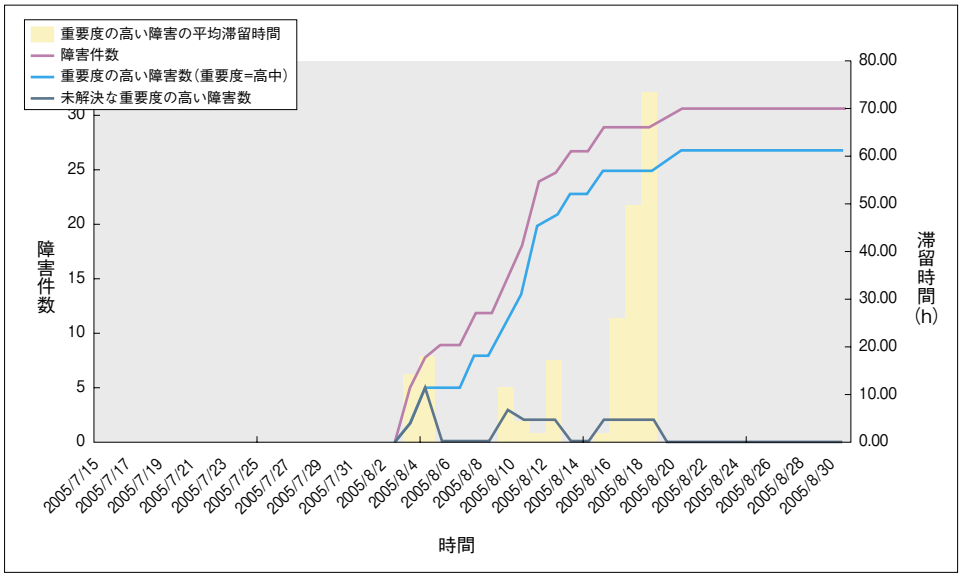


図1-15　障害件数の推移（Y社の例）（松村知子、森崎修司、玉田春昭［A.10］）

図1-16と図1-17は、障害に関する工程別の分析を行ったものである。図1-16は障害の混入工程と本来発見すべき工程の関係、図1-17は発見工程と本来発見すべき工程の関係を示している。表中の数字は、該当する障害件数を示している。

図1-17では、図中の注釈でいくつかの箇所について修正にかかった平均工数を示している。この分析から、発見工程が混入工程と同じ障害、1工程遅れの障害、2工程遅れの障害の修正工数の増加傾向を見ることができる。また発見工程が後ろになるほど工数を要していることもわかる。

	工程	割合	合計件数	混入工程							
				未入力	要件定義	基本設計力	詳細設計	コーディング(製造)／単体テスト	結合テスト	総合テスト	運用
本来発見すべき工程	未入力	2.67%	12	12							
	要件定義	0.00%	0								
	基本設計	0.00%	0								
	詳細設計	10.91%	49				48	1			
	コーディング(製造)/単体テスト	84.41%	379	4			2	372	1		
	結合テスト	2.00%	9					5	3		
	総合テスト	0.00%	0								
	運用	0.00%	0								
	合計		449	17	0	0	50	378	4	0	0
	割合			3.79%	0.00%	0.00%	11.14%	84.19%	0.89%	0.00%	0.00%

混入工程より後工程で発見されるべきとの見解

図1-16　本来発見すべき工程ー混入工程（1社例）（松村知子）

	工程	割合	合計件数	発見工程							
				未入力	要件定義	基本設計力	詳細設計	コーディング(製造)／単体テスト	結合テスト	総合テスト	運用
本来発見するべき工程	未入力	2.67%	12					7	5		
	要件定義	0.00%	0								
	基本設計	0.00%	0								
	詳細設計	10.91%	49					22	27		
	コーディング(製造)	84.41%	379	2				335	42		
	結合テスト	2.00%	9	1					9		
	総合テスト	0.00%	0								
	運用	0.00%	0								
	合計		449	2	0				83	0	0
	割合			0.45%	0.00%	0.00%	0.00%	81.07%	18.49%	0.00%	0.00%

平均工数　1.9人時
平均工数　5.5人時
平均工数　3.3人時
平均工数　1.7人時
平均工数　3.0人時

図1-17　本来発見すべき工程ー発見工程（1社例）（松村知子）

図1-18は1社の担当分に着目し、モジュールごとに障害原因をグラフ化したものである。ここでは、①複数のコンポーネントで発生している機能強化・拡張に起因する問題、②1つのコンポーネントで大量発生した操作ミスに起因する問題について、EPM分析グループから問いが提示された。フィードバックの場での協議の結果、複数のモジュールで発生した機能強化・拡張に関する障害は、外部のシステムからデータを取得する方式に関する共通の要因による障害で、すでに解決しており、また操作ミスとしてカウントされた障害は、試験の操作に関するもので、プログラムそのものの障害ではないことが判明した。いずれもこの時点で特別なアクションに繋がる障害ではなかったが、障害の内容によっては、社内のプロセスの改善や社間調整など、再発防止策、抑制策を必要とする場合もある。

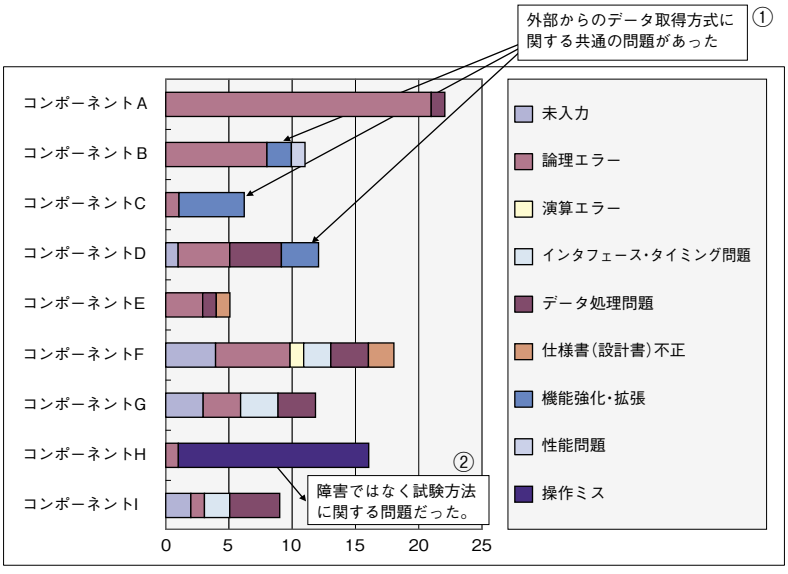


図1-18　コンポーネントごとの問題原因分析（1社例）（松村知子）

EPMは上記で示してきたようにソフトウェア開発のプロセスとプロダクトの推移を視覚的に捉えるための支援になるが、もう1つソフトウェア開発の重要な要素として、開発要員、グループ間のコミュニケーションの課題がある。

EPMにはメールの頻度を計測してグラフ化する機能がある。今回は開発者個人間のメールは計測されず、社間メールのみの計測となった。今回のプロジェクトでは、社間調整は週次のプロジェクト会議、開発事務所内の各社常駐メンバによる打ち合わせ、そして

社間結合試験体制では、開発事務所での毎朝会議などで進められることが多く、社間メールの利用は事務的な連絡、周知に使用されることが多かった。図1-19はEPMによるメール投稿数のグラフ化である。しいていうならば、プロジェクト立ち上がり時に急峻な線が見られ、活発なメールによる調整が進められたことがうかがえ、中段以降、参加人数が増えているにも関わらず、いくらか安定したメール頻度となっていることが読み取れる。

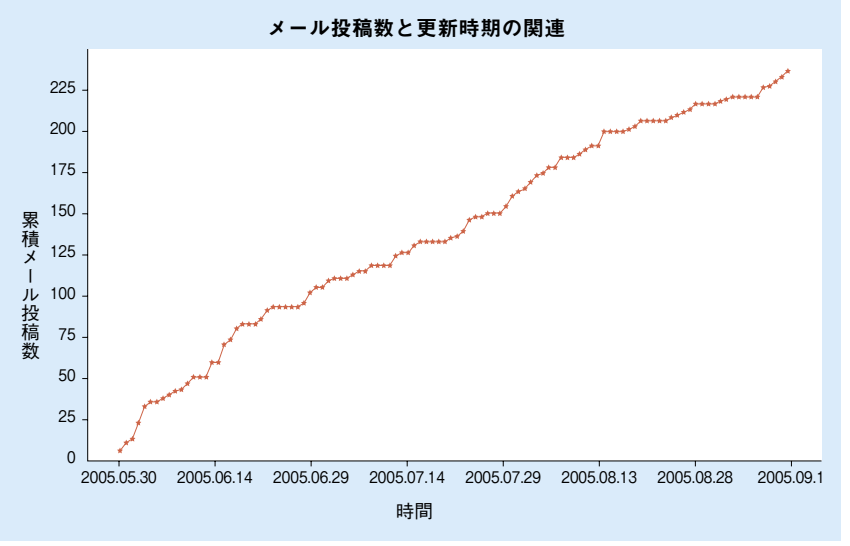


図1-19　社間メール件数の推移計測例（松村知子）

これらの計測・分析結果は、会社間の条件に沿って、組合員各社別にフィードバックされた。その効果はフィードバック先によって異なることが明確になった。

本プロジェクトでは競争領域を設定しているため、会社間に開示制限が存在し、全体のPMには一種のブラインド・マネジメントが要請される。そうした中で、EPMツール群による計測・分析データのフィードバックは、ある程度プロジェクトの透明性を確保するのに役立った。

開発担当各社で、開発現場に接する機会の少ない上位のマネージャにとっては、開発状況をある程度定量的に把握する支援となり、現場に接する機会の多いマネージャにとっては、新しい定量的な材料を提供することになった。また、品質管理部門など並行して多数のプロジェクトを主として書面上で管理している部門にとっては、新しい気づき材料を与えることがあった。

これらのデータ提供は工程区切りの品質保証会議などでの議論を深め、関係者間でプロジェクト運営に対する共通の認識を醸成した。これらの活動は従来の管理手法で得ていた状況認識に対してエビデンスを提供した。このような現場データの捕捉は、感覚的に把握している現場状況を、上位管理組織や他の関係者に早期にエスカレーションするのに役立つという指摘もあった。

また、全般に従来の自己申告方式による進捗報告に比べて、自動収集により省力化され、かつ人手の介入がないことにより正確な報告データを提供することができ、円滑なプロジェクト運営を支援する効果があった。

各グループはこれらの計測とフィードバック手法に接することで啓発される側面もあり、結果的にプロジェクト進行に肯定的な影響を与えたと推定できる。

4.3 ソースコード分析（コードクローン分析）

進行中のプロジェクトのプロセスとプロダクトの計測の一環として、プロダクトに関してコードクローン検出・分析ツールCCFinder/Geminiを用いて、プロジェクトの実行中、数回にわたってコードクローン分析を実施した。

コードクローンとは、ソースコード中に存在する一致または類似したコード片のことで、コピー＆ペーストなどのさまざまな理由により生成され、一般にソフトウェアの保守を困難にする要因と考えられている。たとえば、あるコード片に障害があると、そのコードクローン全てについて修正の検討を行う必要がある。また、機能を追加する場合も同様である。

ソースコードが会社間で開示が制限されている本プロジェクトにおいて、コードクローン分析は、ブラックボックス中のソースコードの特性を推測する材料を与えた。

またコードクローンの観測を実施していることが、コーダのモチベーションやモラルを高める役割を果たすことが示唆された。

図1-20は、コードクローン分析結果の表現法の1つである散布図（スキャターチャート）の概念である。散布図を描くために、まず、分析対象のソースコードを1本の1次元の文字列と捉え、これをコードクローン分析用に、プログラム構造に無関係な変数の除去などの前処理をして、プログラム構造だけに依存する1次元の単語列（これをコードクローン分析用トークン列と呼ぶ）に変換する。散布図はこの1本の単語列（トークン列）を水平、垂

直の2次元に配置したマトリックスを作成して一致箇所を探し、たとえば30単語（トークン）以上一致したマトリックスの交点にプロットを打つという方法で描かれる。単語列（トークン列）中のファイルとモジュールの区切りには、垂直・水平の線を引いて見やすくしている。

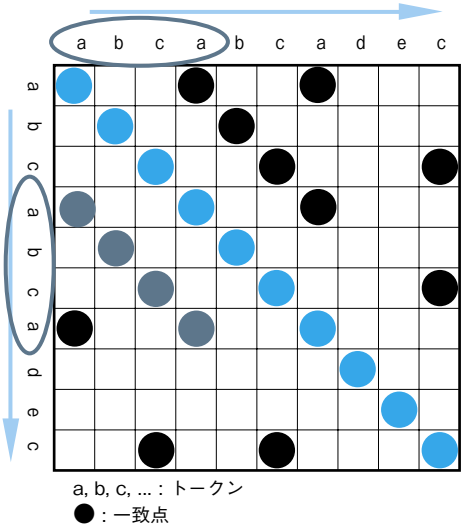
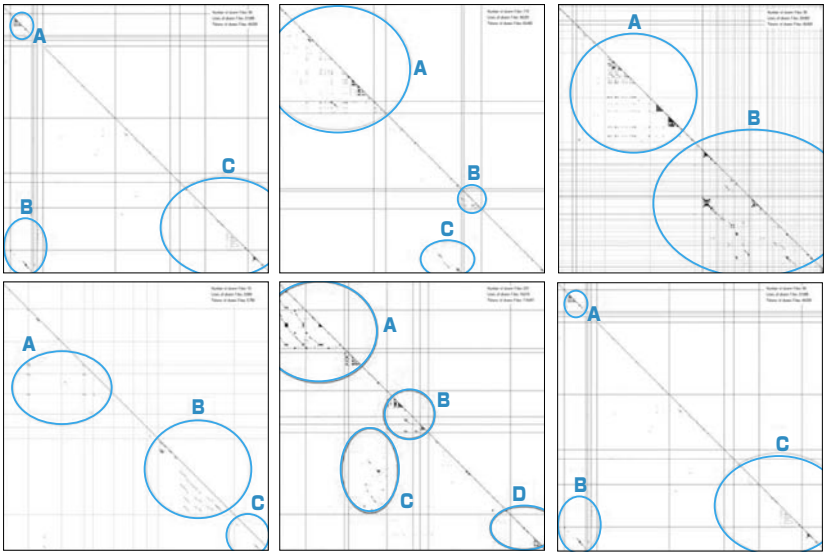


図1-20　コードクローン分析の散布図（スキャターチャート）の概念

図1-21は6グループのある時点の散布図を示している。図では規模やファイル数、モジュール数の異なる各グループのソースコードを同じ大きさの正方形で表現している。また、ファイルとモジュールの区切りに垂直・水平の線を引いて見やすくしている。水平・垂直線が濃いチャートは、規模が大きいことが想定される。チャート上にコードクローンの集中している箇所をいくつかマークしている。コードクローン検出・分析ツールCCFinder/Geminiはこれらの散布図と、コードクローンの存在箇所、コードクローン含有率などのコードクローンに関するメトリックスを表示できる。

散布図中の青丸とA，B，C等の文字は、分析者が気のついた大きなコードクローンの存在場所を示す。この部分に該当するモジュール名などをもとに分析者と開発者の間で、当該コードクローンの存在が既知であったかどうか、既知の場合その生成理由、今後の方針などについて協議することができた。このチャートを共有してコードクローン分析者と開発者が会話することで、他の方法では明らかにならない開発状況や課題を明らかにす

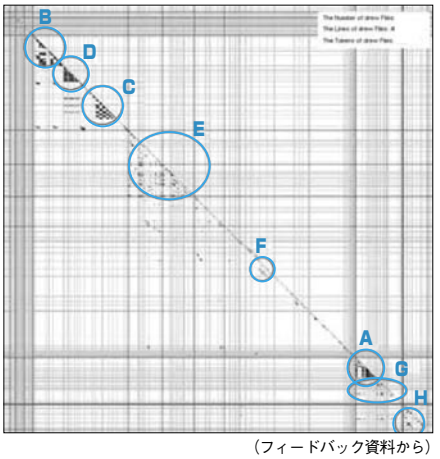
ることができた。また、散布図を比較することで、複雑なコードクローン分布を内包する内部流用の多い開発、コードクローンの少ないスクラッチ型の比較的新規の開発、開発者の保守に対する姿勢などを推測することができた。



（フィードバック資料から）

図1-21　コードクローン分析例（散布図、6グループ）（肥後芳樹［A.8］）

たとえば、図1-22はある社のある時点の散布図である。ここで顕著なコードクローンとしてAからHまで8個のコードクローンの存在が指摘され、それぞれについて開発者と議論があった。



（フィードバック資料から）

図1-22　コードクローン分析例（吉田則裕［A.8］）

プロジェクト実行中数回コードクローン分析を行い、結果を会社間の条件に沿って組合員企業にフィードバックした。全体のPMには全社分の結果を提供した。

ソースコードの開示が制限されたPMに対して、各社の開発したコードの属性をコードクローン分布の視点からマクロに把握するのを支援できた。

現場との接点の少ないマネージャは、自社のコードクローン分布から管理上の「気づき」、たとえば、新人の投入など把握していないことに気づくことがあった。

現場に接しているリーダにとっては、開発対象の機能充足だけでなく構造に関する深い議論ができ、モチベーションとモラルの向上に貢献した。たとえば、検出された中規模のコードクローンに対して、当該機能は現状では類似しているが、将来片方が大幅に進化することが想定され、保守上の配慮からあえて分けて（コードクローンを作って）開発したという設計思想を主張できた。また、別の例では、工程後段に中規模のコードクローンが検出されたが、その要因は、類似機能の片方の詳細仕様決定が社間調整等の仕様調整で遅れ、仕様決定時には先行機能の試験が終了していたため、コードクローンとして開発し、開発・試験工数の抑制と開発期間の短縮を図った、というような全体のマネジメントに起因する要因を説明できた。

コードクローン分析を中心としたソースコード分析はフェーズ2でも行い、2年間にわたる分析を進めた。

4.4 チェックシートによるリスク分析

開発担当各社のPMに対して自己評価シートを用いて自己評価を実施した。また、約80項目のヒアリングシートに基づいた専門家によるヒヤリングを各社2時間程度実施することにより、プロジェクト遂行上のリスクを把握する試みを行った。

この計測は、EPM、コードクローン計測のような自動収集あるいはそれに近い機械的なデータ収集方式では得られないプロジェクトのコンテキスト情報（文脈情報）を収集し、プロジェクト運営の実相を明らかにしようとするものである。本計測は、経済産業省のプロジェクト見える化部会で開発したチェックシートの実証実験として行われた。

当該チェックシートは約40項目の自己評価用と、約80項目からなる専門家によるヒアリング用のものがある。SECでは2005年度に下流工程用、2006年度に上流工程用を作成し、それぞれ書籍を出版している［D.3、D.2］。このチェックシートは表1-1、表1-2に示すように、プロジェクトマネジメントに関わる知識を体系化したPMBOKで定義する9つの知識エリアに、拡張知識エリアを加えたものである。拡張知識エリアとしては、組織や人間関係が大きな意味を持つ日本のすり合わせ的な開発と管理に配慮してヒューマンウェアの分野から顧客、組織、基本動作、モチベーション、技術の5項目、さらに課題管理の項目を加えて6つの知識エリアを設定した。合計15の知識エリアからプロジェクトマネジメント上の課題、リスクをあぶりだす設問を用意した。

表1－1　PMBOK知識エリアの定義

知識エリア	定義
統合	プロジェクトが全体としてうまくいくように調整する活動を扱う。プロジェクトの立ち上げと終結を定義し、プロジェクトの計画、実行、監視コントロールの各プロセス群において、他の８つの知識エリアを統合するための活動を取り扱う。具体的には、競合する目標や代替案間の調整などを行う
スコープ	プロジェクトに何が含まれ、何が含まれないかを明らかにし、コントロールする活動を扱う。プロジェクトに必要な成果物と作業を定義するWBSの作成はプロジェクト・スコープ・マネジメントに含まれる
タイム	プロジェクトをある期間内に完了させるために必要な活動を扱う。プロジェクトの成果物を作成するために必要な作業の定義とその作業に必要な資源と期間の見積りやスケジュールの作成と変更管理が含まれる
コスト	プロジェクトをある予算内に完了させるために必要な活動を扱う。プロジェクトを完了するのに必要なコストの見積りと予算化、プロジェクトに関するさまざまな変更がコストに及ぼす影響を管理する
品質	プロジェクトが取り組むべきニーズを満たすための活動を扱う。品質方針、品質目標、品質に対する責任を決定する組織の全ての活動が含まれる
人的資源	プロジェクト・チームを組織化し、マネジメントするための活動を扱う。プロジェクトに必要な作業（WBS）と組織図から責任分担を決める活動はプロジェクト・人的資源・マネジメントに含まれる
コミュニケーション	プロジェクトに関わる情報のやりとりをプロジェクトの利害関係者間で適時、適切、確実に行うための活動を扱う。プロジェクトの利害関係者のニーズを満たし、課題を解決するための活動はプロジェクト･コミュニケーション・マネジメントに含まれる
リスク	プロジェクトの期間、コスト、スコープ、品質に影響を与える不確実な事象であるプロジェクトリスクのマネジメントの計画、識別、分析、対抗、監視コントロールに関する活動を扱う
調達	プロジェクトの作業の実行に必要な製品、サービスをプロジェクトチームの外部から購入・取得する活動を扱う。契約に関するマネジメントも含まれる

出典　：プロジェクトマネジメント協会『プロジェクトマネジメント知識体系ガイド（PMBOKガイド）第３版』よりIPA/SEC作成

表 1－2　拡張知識エリアの定義

拡張知識エリア	定義
顧客	プロジェクトの利害関係者のうち、システムの仕様及び予算について最終決定権を持っている人もしくは組織のこと。受託型のシステム開発プロジェクトでは、顧客と協働で、最終的な成果物であるシステムを作るための合意形成を行っていく場合が多い
組織	システム開発に携わる組織のこと。システム開発プロジェクトでは、多段階の下請け構造を取る場合や複数の開発会社が参画するマルチベンダー方式が取られることもある。現実的には、開発組織構造によって、『人的資源』や『調達』のあり方にさまざまな制約が生じることになる
基本動作	システム開発における常識及び開発者が身に付けておくべき当たり前の動作のこと。システム開発マネジメントの内容も含まれる
モチベーション	システム開発に携わる人のやる気、動機付けのこと。システム開発に携わる人の心理的側面、労務環境や個々人の成長目標などのキャリアディベロップメントに関する事項も広く含まれる
技術	システム構築技術に関するマネジメント項目。システム開発マネジメントの内容であり、PMBOK には含まれない業種固有のマネジメント領域に当る
課題管理	システム開発プロジェクトにおける課題管理に関わるマネジメント項目。PMBOK では、監視・コントロールプロセスのマネジメントに該当するが、本書では、プロジェクトの下流工程に焦点を当てているため、特に重要な領域として拡張エリアに追加している。システム開発の現場で当然行われるべき課題管理のあり方を扱う

設問への回答は、自己評価シートの場合は3段階、専門家用のヒヤリングシートの場合は5段階で採点される。その結果は15個の知識エリアごとに集計され、レーダーチャートで表現される。また、自己診評価果とヒヤリング結果が比較可能な図で可視的に表現される。

プロジェクトでは、フェーズ1で全社に下流工程用を適用し、プロジェクトリスクの把握に努めた。同時に、チェックシートの適用について各社からコメントを収集し、チェックシートそのものの改善に反映した。また、公開デモ向け開発においては上流工程用を3社に適用し、同じくチェックシートの改善に反映した。

チェックシートの適用結果はヒアリングを行った各社別にレーダーチャートで示すとともに、プロジェクト見える化部会委員による課題の抽出とアドバイスを示した。

本分析活動では、自動計測では収集できない各社のプロジェクト体制やマネジメント

の仕組みに関するさまざまな情報を得ることができるとともに、これらの情報をプロジェクト全体の計測とフィードバック活動に役立てることができた。これらの状況からチェックシートの効用を認識することができると考える。ヒアリングに立ち会うことによって、公式的な報告・申告だけでは把握できない各社独自の状況や管理手法について把握することができ、得られた情報は、ほかの計測データの分析にも役立った。

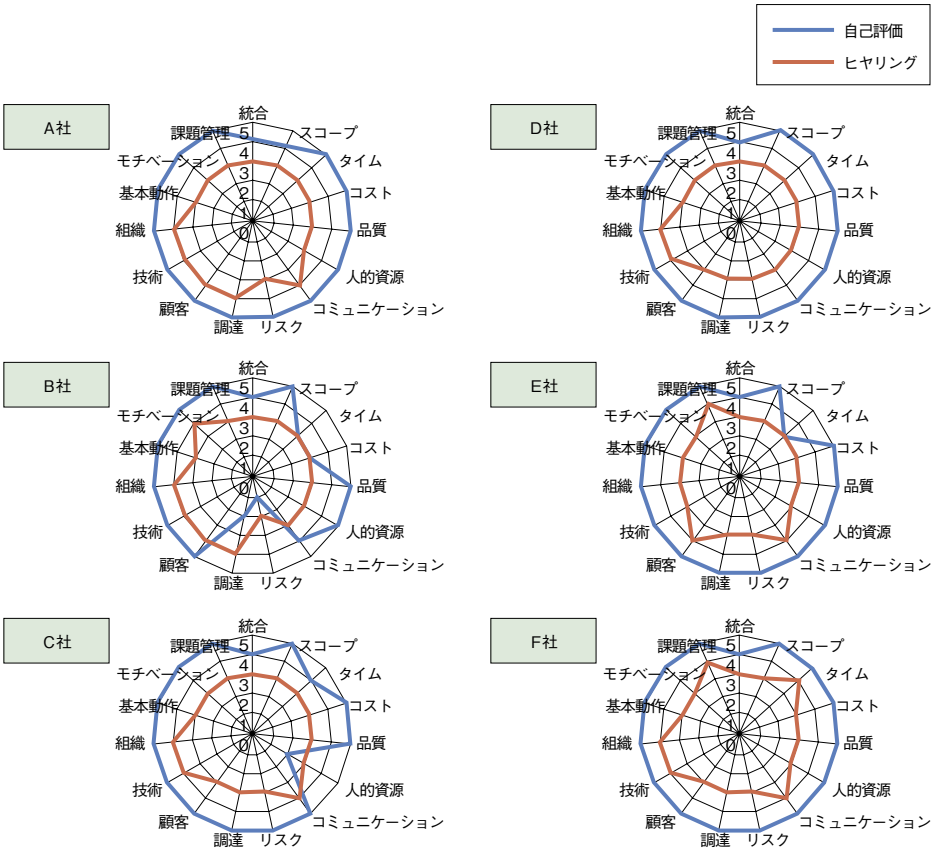


図1-23　チェックシート分析結果例（レーダーチャート、6社分、フェーズ1）

レーダーチャート上の分析では、①自己診断と専門家診断に乖離がない（認識があっている）こと、②点数が低い部分に対し、専門家がサポートすることができることの2つが重要である。実際、ヒアリングの中でリスク管理に関して点数の低い会社があり、これをもとに注意喚起を行って要員強化につながった。

自己評価結果とヒアリング結果の比較分析を行うことで、PMのプロジェクトリスクに

関する認識についてある程度把握できることを確認した。また、ヒアリングに参加した外部専門家の意見を集約してPMに説明することにより、プロジェクトを成功に導くための対策やヒントを提示できた。

Section 5

より高度なデータ収集と分析（フェーズ2、公開デモ向け開発を中心に）

先進プロジェクトでは、フェーズ1でのプロジェクト計測とフィードバックの状況から、EPMをはじめとする各種手法の有効性が認められた。フェーズ2においては、これらの手法をより積極的に活用するために、計測とフィードバックをプロジェクトマネジメントの中に組み込む試みを進めた。具体的にはフィードバックの頻度を週次にし、工程区切りの品質評価会議で可視化した分析データを共有しながら、プロジェクト進捗に関するより深い議論を進めた。また、分析方法もフェーズ1の経験を反映して高度化を図った。

さらに公開デモ向け開発では、開発期間も短かったことから、従来の自己申告方式の進捗報告をやめ、EPMによる計測と分析データの共用でこれに代え、EPMを完全にプロジェクトマネジメントに組み込む方式を試みた。

以下に、フェーズ2以降の計測と分析の中から特徴的な例を紹介する。

5.1 設計レビュー分析の高度化

設計レビュー分析についてフェーズ2では、フェーズ1との比較も含めてより深い計測と分析が実施された。

フェーズ2では、設計ドキュメントもCVSの管理対象とし、週次でデータを収集した。ドキュメントごとのページ数の推移や変更状況（修正箇所、変更文字数等）を把握し、レビュー結果や障害追跡システムのデータと合わせて分析を行った。

分析の結果、設計書のレビューと下流工程で検出された障害の関係について、いくつかの傾向が見られた。

フェーズ1とフェーズ2のレビューデータの差異からレビュー効率を比較する試みも行

われた。この成果は奈良先端大学テクニカルレポート［A.5］にまとめられている。

5.2 EPM Pro＊によるより高度な分析

フェーズ2では、EPMによって収集された情報をもとにEPM PRO＊を用いてより高度な分析が試みられた。たとえば、ソースコード行数やソースコードを格納したファイルへの操作頻度を時間軸で追うことによって、開発作業の進捗の一端を掴むことができる。

1） ファイルの更新回数を追うことで開発の完了状態を見ることができる
2） ファイル数の変化から進捗状況を推測できる
3） 追加行数と削除行数がともに大きく増加する時は大規模な追加が想定される
4） 追加行数のみが増加する時は新規開発が想定される
5） 削除量の大きい更新の比率が高い場合、あるいは削除行数の割合が高い場合、既存コードへの変更が頻繁、または大量で、変更の危険性や既存コードの修正の複雑さを推測できる

図1-24、図1-25に一例を示す。2つの図は横軸の時間軸を揃えてある。この例では、工程も進んだ段階になってそれまでの安定状態が急変し、ファイル数が減り、プログラム行数が減り、大幅な削除行数があり、ファイル変更者の人数が増加し、削除をともなう更新が急増している。このような事象に対し開発担当から申告が無い場合には理由をただす必要がある。なお、表1-3は、図1-25の計測項目A～D-Back Dの内容を表している。

EPM Pro＊を用いた分析について、またEPM Pro＊のアーキテクチャなどにつては、それぞれ奈良先端大学テクニカルレポート［A.10］［A.11］にまとめられている。

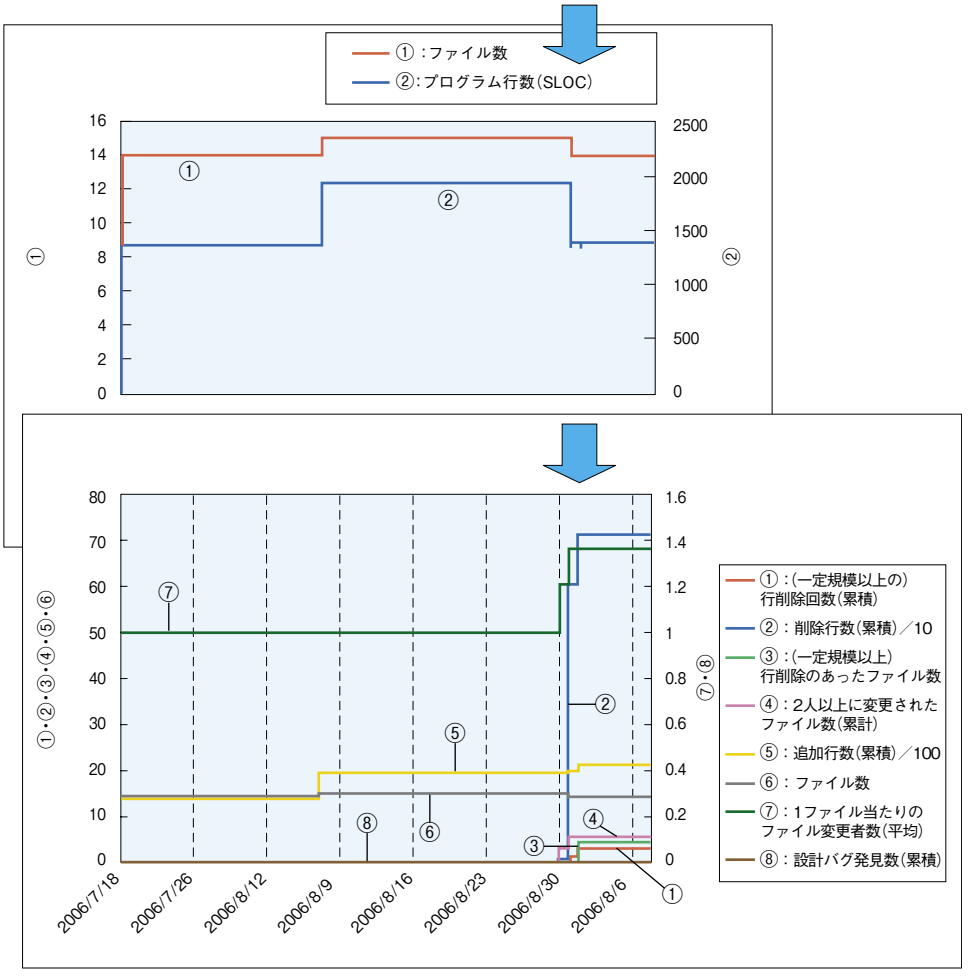


図1-24　EPM Pro＊分析例（1）（松村知子、森崎修司、玉田春昭）

表1-3　EPM Pro＊での計測項目（行数関連）（松村知子、森崎修司、玉田春昭）

メトリクス略号一覧

A	ファイルの更新回数（累積）
B	（一定規模以上の）行削除回数（累積）
C	追加行数（累積）
D	削除行数（累積）
A-BackA	更新回数（周変化量）
B-BackB	（一定規模以上の）行削除回数（週変化量）
C-BackC	追加行数（週変化量）
D-BackD	削除行数（週変化量）

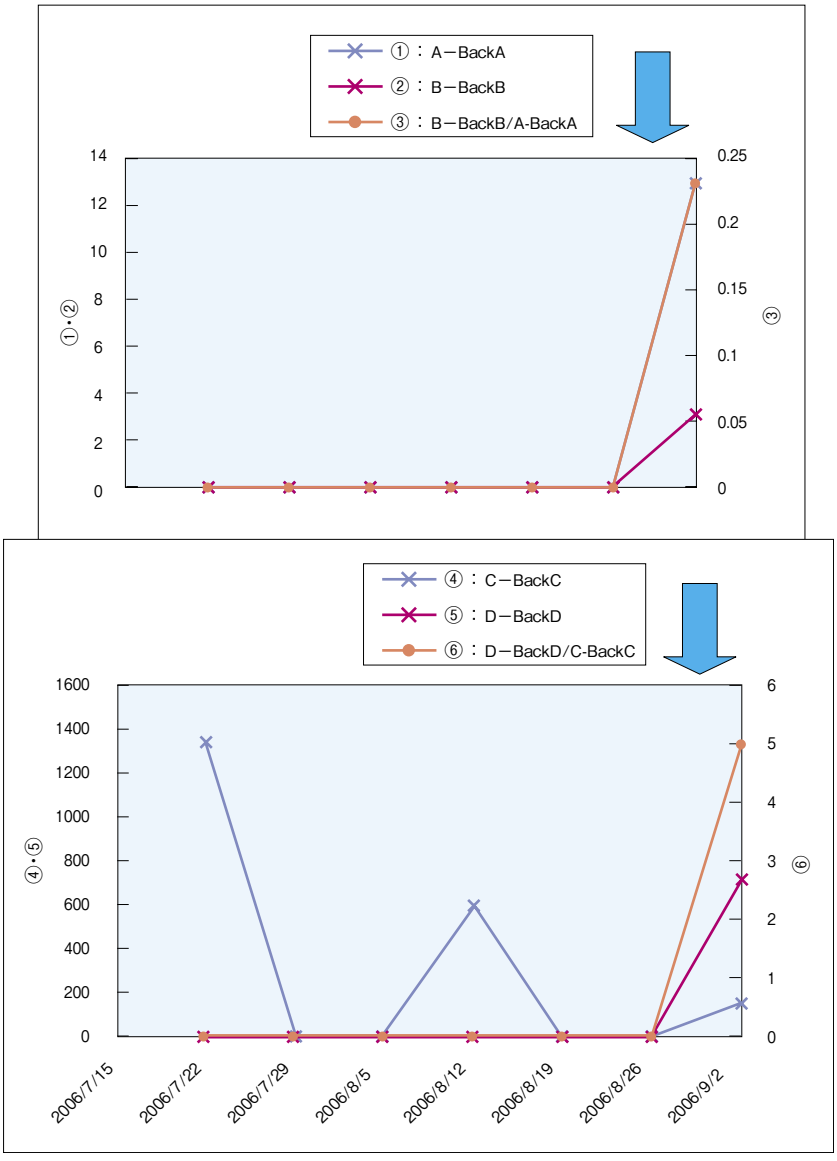


図1-25　EPM Pro＊分析例（2）（松村知子、森崎修司、玉田春昭）

図1-26は1ファイル当たりのファイル更新者数の推移を追うことで開発体制上の異変を捕捉する試みである。この例では、同じ開発グループの複数のモジュールで一斉にファイル更新者の人数が増加しており、開発体制の異変が想定される。

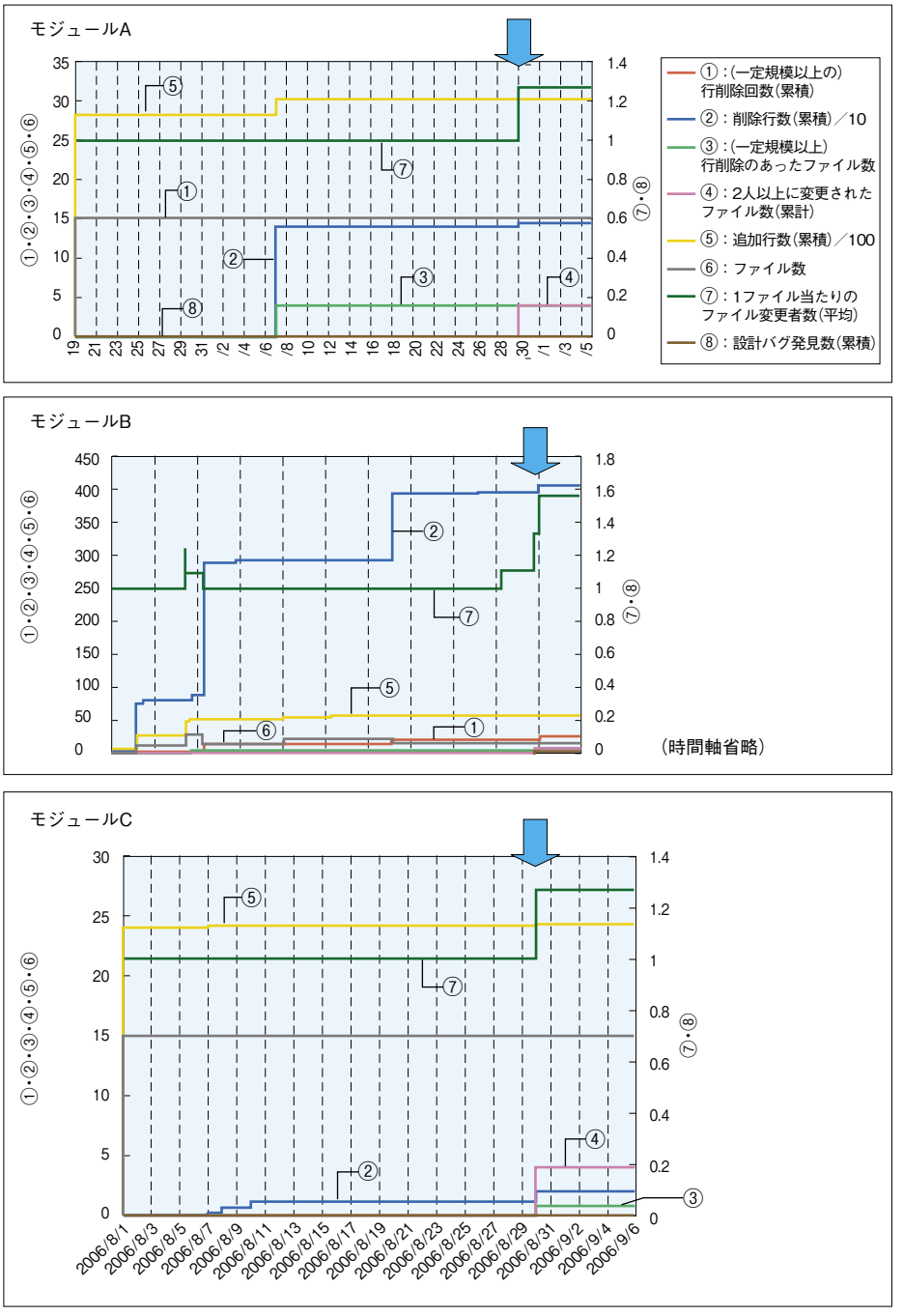


図1-26　EPM Pro＊分析例（3）（松村知子、森崎修司、玉田春昭）

※番号に該当するグラフがない場合、データがありません。

5.3 開発フェーズをまたがるソースコード分析（コードクローン分析を中心に）

フェーズ1、フェーズ2のそれぞれ、およびフェーズ1とフェーズ2間のコードクローンを散布図で分析した。図1-27は、フェーズ1のソースコードとフェーズ2のソースコードを直列に繋いだコード列のコードクローン分布を表示している。フェーズ間を比較することによって、フェーズ1に内在したコードクローンがフェーズ2でどのように扱われたか、および、フェーズ1からフェーズ2への流用開発の様子を推し量ることができる。図1-27の例では、フェーズ1を流用しながらフェーズ2を開発し、またフェーズ1のコードクローンをそのまま引き継いでいる様子もわかる。

図1-27右の図で①の領域ではフェーズ1のコードクローン、②の領域ではフェーズ2のコードクローンを表示している。フェーズ2では、フェーズ1と同じような分布のコードクローンが、図中のXで示す部分のように平行移動した形で認められる部分がある。これは、フェーズ2で再利用したフェーズ1のコードクローンがそのまま持ち込まれていることを意味する。また、図中Yで示す部分のようにフェーズ2のコードの中に新たなコードクローンが作られている部分も認められる。

③の領域はフェーズ1とフェーズ2の間のクローン分布を示している。もし、フェーズ2のコードが100%フェーズ1と同じだった場合には、図中のZで示す斜線のようなクローン分布が表示されることになる。今回は左の図のAで示される領域に、平行移動した形で大きなクローンが認められる。この部分は大きな単位で再利用されたことがわかり、全体の約50%の分量に相当する。フェーズ2の残りの部分は比較的新規開発に近く、その中に、B, C, Dで示される流用（再利用）部分が点在している、ということを認めることができる。

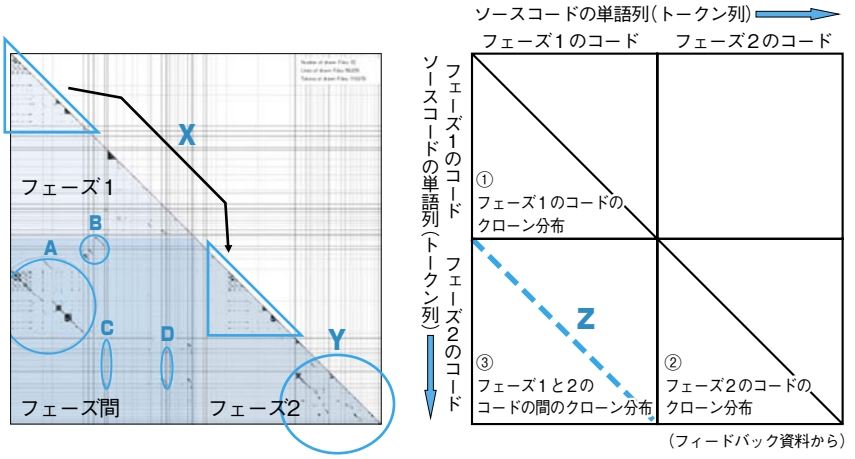


図1-27　フェーズ間のクローン分析例（1社例）（吉田則裕）

図1-28はコードクローン分析の分析者が、その分析から判定した2つのフェーズのプロダクトを構成する各モジュールの流用（再利用）・新規開発状況を示した例である。この構成はリーダによって概ね正しい認識であることが確認された。このことによってコードクローン分析が、プログラムの流用（再利用）・新規開発に関する設計情報に触れる機会の無い者にとっても、プログラム構造の変遷を追跡する有力な手段であることが実証された。

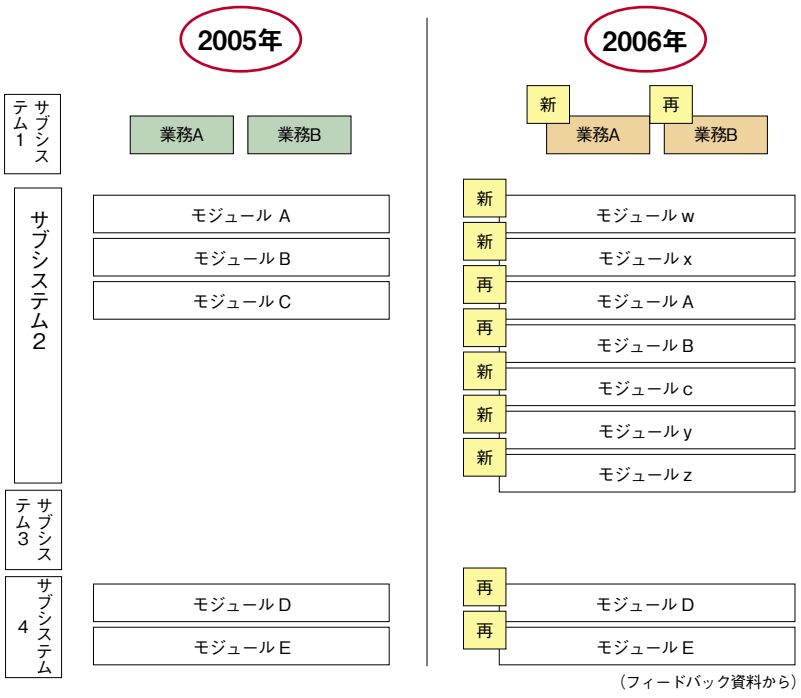


図1-28　クローン分析から判定したモジュールごとの流用・新規開発状況（吉田則裕）

フェーズ1、フェーズ2に関して、プロジェクト実行中数回コードクローン分析を行い、結果を会社間の条件に沿って組合員会社にフィードバックした。全体のPMには全社分の結果を提供した。

フェーズ1、フェーズ2、という大きなフェーズごとのコードクローン分布を追うことで、各社の各フェーズへの流用とソースコードの洗練に対する開発姿勢を類推するのを支援することができた。

こうした進行中のプロジェクトのコードクローン分析とフィードバックは、できるだけ保守性の高いコードクローンの少ないコードを作るという意味で、プロジェクト進行に肯定的な影響を与えたことが観測された。

開発担当社の1つからは、フェーズ1の開発でコードクローン分析による観測が行われ、コードクローンの存在を指摘されたために、フェーズ2ではできるだけ良いコードを作る努力をしたという報告があった。具体的には好ましくないコーディングを機械的に検出するツールであるソースコード静的チェッカを多用してソースコードの改善を図り、また

コードレビューを徹底したということである。

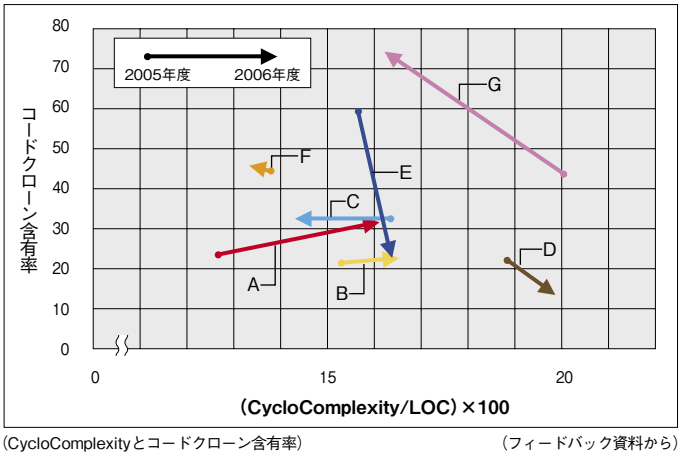
また、ある社のリーダは、コードクローン分析の存在がコーディング担当への肯定的なプレッシャーになると同時に士気向上に貢献したと述べた。

コードクローン分析に関して提案もとの大学と実データを用いた共同研究を開始した社や、社内での評価を検討している社もある。

さらに、複数の開発フェーズ（複数年）にまたがるソースコードのコードクローン分析により、こうした分析が、ソースコードの変遷を把握する有効な手段であることが示唆された。開発フェーズごとの継続したコードクローン分析、相互分析は、開発されたプログラムの保守を考えてゆく上で重要な情報を与えてくれることがわかった。

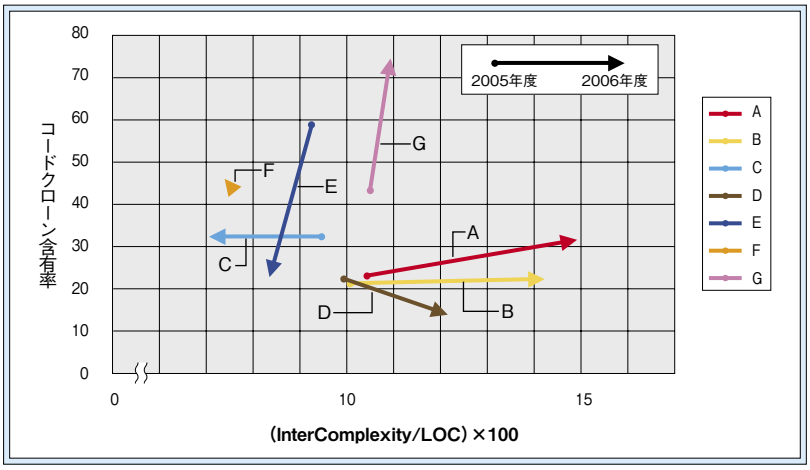
コードクローン含有率と他のソースコードに関するメトリクスを組み合わせて、2年間の開発の推移を分析する試みも行われた。図1-29、図1-30にその例を示す。全社分を俯瞰できるように重畳させてグラフ化している。このほかに、各社のモジュール別にグラフ化したものも描いた。図1-29はコードクローン含有率とCycroComplexity、すなわち分岐の数を軸とした推移、図1-30はコードクローン含有率とInterComplexity、すなわち関数中にある引数とReturnの総和を軸とする推移を示している。2年間比較することで、複雑度とコードクローン含有率の変化を見ることができる。

たとえば、G社は複雑度は増加しなかったのにコードクローン含有率を増加させ、一方E社はコードクローン含有率を下げたことがわかる。



(CycloComplexityとコードクローン含有率)　　(フィードバック資料から)

図1-29　コードクローンとプログラム複雑度の年次変化分析（全社）（馬場慎太郎）



(InterComplexityとコードクローン含有率)　　(フィードバック資料から)

図1-30　コードクローンとプログラム複雑度の年次変化分析(社別、全社俯瞰)(馬場慎太郎)

5.4 協調フィルタリングによる過去データを用いた工数予測

協調フィルタリングによる工数予測とは、データセット(複数のプロジェクトデータの組み合わせ)に対して、1つのプロジェクトデータを与えて類似のプロジェクトデータに基づき工数予測を行う技術である。その特徴は、データセットをベクトルとして捉え、ベクトルのなす角の大きさに基づいて類似プロジェクトを選び出す点にあり、対象となるデータセット中に多くの欠損を含んでいても、データに手を加えることなく高い精度で予測できる。協調フィルタリングによる工数予測は、eコマースシステムでリコメンデーションエンジン(推奨システム)に用いられている技術を応用したものである(例:インターネット書店サイト内での図書推奨システム)。

今回、SEC定量データベースに格納されているベンチマークデータを利用して、EASEプロジェクトで開発された協調フィルタリングによる予測ツールMagiを用いて工数見積りを行った。SEC定量データベースには、プロジェクトごとにその工数、工期、障害件数などの計画値と実績値のデータが主要なもので80項目、細部まで含めると400項目以上が収集されている。

具体的には、基本設計工程終了時に、全体の工程別工数見積もり値（計画値）と基本設計工程の実績値をキーとして、SEC定量データベースのなかから過去の類似プロジェクトを抽出して開発工数の予測を行う試みである。

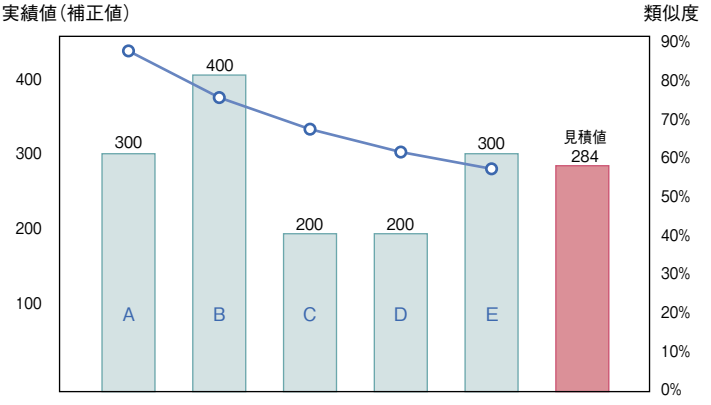


図1-31　協調フィルタリングによる見積りツールMagiの出力例（大杉直樹、角田雅照、柿元健）

図1-31に協調フィルタリングによる予測ツールMagiの出力例を示す。この場合、Magiは新規プロジェクトの工程途中までのデータをキーにしてSEC定量データベースの中から類似プロジェクトのベスト5を抽出している。棒グラフAからEが抽出されたベスト5の類似プロジェクトの実績値（左軸）で、グラフ上方の線が類似度（右軸）を現している。右はじの棒グラフがもともとの見積値である。

このベスト5の過去データは、値が比較的揃う場合と、ばらつく場合が見られた。そして、このMagiによる予測はプロジェクト終了後実績値と比較された。その結果は、予測値と実績値が近い場合と離れている場合があった。

Magiによる予測値が実績値と近かった会社は、この手法が利用可能と判断した。一方、予測値と実績値が異なる会社の場合でも、この手法が検索の母体となる過去データの性質に依存することから、SEC定量データベースではなく、たとえば自社の保有する類似度の高いデータベースを用いれば、予測精度が上がるのではないかと考え、自社データベースによる次の実証プロジェクトを検討するところもあった。

5.5 相関ルール分析

相関ルール分析は、分析対象データ間で何らかの関係があるかどうかを分析するもので、分析対象データ間に「AならばB」という関係が存在する時、それを相関ルールと呼び、関係の強さ（支持度）を確認することによって分析対象データ間の傾向を把握することができる。また、相関ルールに対してある条件を加えることにより結論が変わるものを例外ルールと呼ぶ。

障害追跡システムに入力された単体試験から結合試験までの障害に関して、障害の重要度、混入工程、発見工程、原因などの情報と構成管理システムに登録された修正量やファイル数の情報をEASEプロジェクトで開発された相関ルール分析ツールNEEDLEを使って分析を行った。

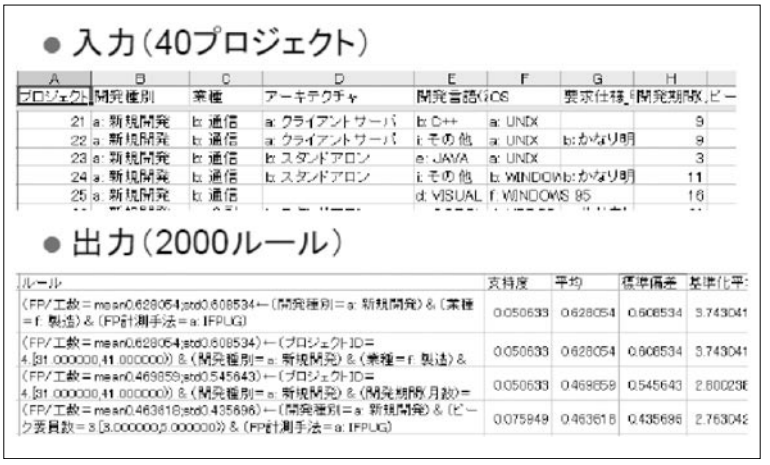
分析の結果、各社ごとにいくつかの相関ルールおよび例外ルールが検出された。たとえば、相関ルールとしては以下のような傾向が見られた。

- プログラム設計で混入され、社内結合試験で検出された障害の修正は、更新ファイル数が多くなる傾向がある
- 優先度が高い障害の修正は追加行数が多くなる傾向がある
- 重要度が高く、優先度も高い障害の修正は削除行数が多くなる傾向がある
- プログラム設計工程で検出すべき障害で、レビュー漏れで検出できなかった障害の修正は削除行数が多くなる傾向がある

すなわち、設計工程で混入された障害の修正や重要度・優先度が高い障害の修正は影響範囲が大きくなることが確認できた。

全体として大きな傾向はあるものの、抽出された個々のルールは各社ごとに異なり、各社間で同じルールは抽出されなかった。すなわち、障害の細かな傾向は各社ごとに特色があり、品質向上のための施策など、障害に対する対策は各社ごとに検討する必要があることがわかった。

図1-32に、相関ルール分析ツールNEEDLEの入出力表示例を示す。



A	B	C	D	E	F	G	H
プロジェクト	開発種別	業種	アーキテクチャ	開発言語(OS		要求仕様_	開発期間_ピー
21	a: 新規開発	b: 通信	a: クライアントサーバ	b: C++	a: UNIX		9
22	a: 新規開発	b: 通信	a: クライアントサーバ	i: その他	a: UNIX	b: かなり明	9
23	a: 新規開発	b: 通信	b: スタンドアロン	e: JAVA	a: UNIX		3
24	a: 新規開発	b: 通信	b: スタンドアロン	i: その他	b: WINDOWS	b: かなり明	11
25	a: 新規開発	b: 通信		d: VISUAL	f: WINDOWS 95		16

ルール	支持度	平均	標準偏差	基準化平
(FP/工数＝mean0.628054;std0.608534)←(開発種別＝a: 新規開発) & (業種＝f: 製造) & (FP計測手法＝a: IFPUG)	0.050633	0.628054	0.608534	3.743041
(FP/工数＝mean0.628054;std0.608534)←(プロジェクトID＝4.[31.000000,41.000000]) & (開発種別＝a: 新規開発) & (業種＝f: 製造) &	0.050633	0.628054	0.608534	3.743041
(FP/工数＝mean0.469659;std0.545643)←(プロジェクトID＝4.[31.000000,41.000000]) & (開発種別＝a: 新規開発) & (開発期間_月数)＝	0.050633	0.469659	0.545643	2.800238
(FP/工数＝mean0.463618;std0.435696)←(開発種別＝a: 新規開発) & (ピーク要員数＝8.[8.000000,8.000000]) & (FP計測手法＝a: IFPUG)	0.075949	0.463618	0.435696	2.763042

図1-32　相関ルール分析ツールＮＥＥＤＬＥの入出力表示例（森崎修司、松村知子）

各社の障害情報の分析により抽出された相関ルールおよび例外ルールを各社ごとに提示し、分析結果を総合試験の品質評価会議に提示した。この分析は障害情報がある程度蓄積されていないと実施できないため、テスト工程終了後のフィードバックとなった。同じプロジェクトメンバによって次の開発が行われる場合があれば、障害の混入防止策や早期検出の対策を検討するための参考情報となる。

なお、この結果は奈良先端大学テクニカルレポート［A.9］でまとめられているほか、国際ワークショップ［A.7］においても発表されている。

5.6 チームスキルと成果物品質との相関分析

本プロジェクトに参加したメンバで、開発に関連した全員に対して、ITスキル標準（Ver.1.1）に基づいたスキル項目とソフトウェアエンジニアリングに関して新たに定義した詳細なスキル項目について、スキル管理システム（SSI-ITSS）を用いて診断し、その情報を分析した。ITスキル標準バージョン1.1では、11の職種を定義しており、職種ごとに専門分野を設定し、合計38の分野に分けてレベルを規定している。

今回の分析では、ITスキル標準バージョン1.1 に基づいたスキルレベルの判定を行った。また、分析対象としては、ソフトウェア開発に関係の深い以下の職種に重点を置いた。

- プロジェクトマネジメント
- IT スペシャリスト
- ソフトウェアデベロップメント

各スキル項目に対して個人ごとに表1-5に示すようにランクを入力し、スキルフレームワークに従ったレベル判定を行った。

表1-5　スキル・ランク

ランク	意味
1	知識がある
2	上級者の支援があれば実施できる
3	単独で実施できる
4	後進の育成・指導ができる

レベル判定結果を専門分野ごとに、以下のようにレーダーチャートで可視化した。これによって、個人の強み･弱みが明確になった。たとえば、図1-33は、ソフトウェアデベロップメントのミドルソフトについてのスキル判定結果を示している。個人ａは､コミュニケーション、ネゴシエーション、リーダーシップ、品質マネジメントのスキルは高いが、プラットフォーム非依存設計のスキルはなく、分析・要求定義、開発方式設計、技術支援に関するスキルも低い。一方個人ｂは､「その他」を除いて高いスキルを持っていることがわかる。なお､「その他」には、設計ツール、開発言語、開発ツール、開発管理ツール、クライアントOS、データベース、グループウェアに関するスキル項目が含まれている。

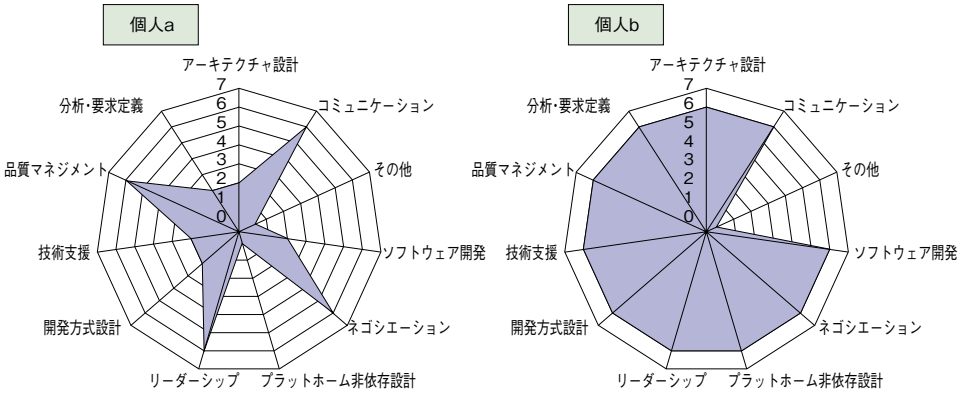


図1-33　スキルレベル表示例（個人）

チームとしてのスキルレベルを判断するために、各社のプロジェクトチームに所属するメンバのスキルレベルを集計し、最大値、最小値、平均値を示すことで、チームのスキルを可視化した。

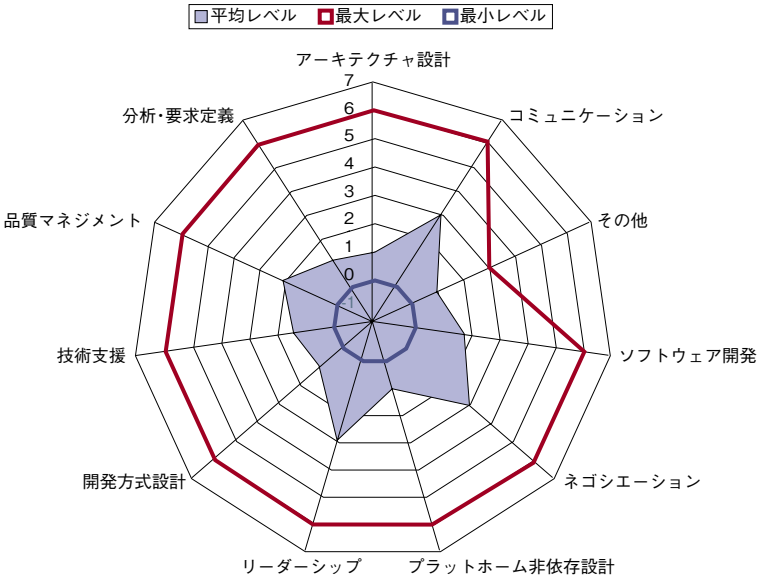


図1-34　スキルレベル表示例（チーム）

図1-34を見ると、ほとんどのスキル領域で高いレベルを持ったメンバがいる一方で、スキルのないメンバも含まれていることがわかる。また、チームを平均するとリーダーシップ、ネゴシエーションとコミュニケーションについてはある程度のスキルがあるが、分析・要求定義、アーキテクチャ設計、プラットフォーム非依存設計、その他についてのスキルは低いことがわかる。

このようにスキルレベルを可視化することで、チームに不足しているスキルが明確になり、必要に応じて要員の追加や研修の実施などの対策を立案することができる。

各チームのスキルレベルを専門分野ごとに比較した結果、以下の分野で差が見られた。

【IT スキル標準】

- ソフトウェアデベロップメントのミドルソフト
- プロジェクトマネジメントのソフトウェア開発

【ソフトウェアエンジニアリング】

- ソフトウェアエンジニアリングマネジメント
- ソフトウェア設計

そこで、上記専門分野におけるチームのスキルレベルと社内試験までに摘出した各チームの障害密度（件数／規模）について相関を分析したところ、ソフトウェアエンジニアリングマネジメントに関するスキルが高いチームほど、障害密度が低い（ソースコード規模当たりに検出される障害の数が少ない）傾向がうかがえた。障害密度が低いのは、設計ドキュメントに混入されている不具合が少ない、レビューによる障害の事前摘出が行われていることなどが考えられ、成果物としての品質が高いと判断できる。すなわち、ソフトウェアエンジニアリングマネジメントのスキルレベルの高いチームは成果物の品質が高い傾向が見られた。

各個人にとっては、チームあるいはCOSE全体のスキルレベルと自分のスキルレベルとを比較することにより、自分自身の位置付けを把握することができる。

各社のPMあるいはチームリーダにとっては、各社のチームのスキルレベルがCOSE全体の中でどのような位置付けかを把握することができる。

今回は、各社のチームメンバの構成が複数社におよぶ場合があったため、各社のPMにはプロジェクトメンバ個人のスキル情報は渡していない。しかし、社内メンバのみで構成されるプロジェクトにおいては、PMが個人のスキル情報を参照することによりプロジェクトメンバの育成に活用することができる。

第2部

実践された技術とその活用方法

Section 1

プロジェクト状況をリアルタイムに把握するEPMツール

先進プロジェクトでは、さまざまなソフトウェアエンジニアリングが実践され、その有効性について確認が行われた。第１部でも紹介したように、主に以下の技術に関する実践が行われた。

① EPM（Empirical Project Monitor）
② コードクローン分析（CCFinder/Gemini）
③ 協調フィルタリング（EASE CF 法、Magi）
④ チェックシートによる自己評価とヒアリング
⑤ ロジカルカップリング分析
⑥ 相関ルール分析（EASE NEEDLE）
⑦ スキル分析

上記のうち、ロジカルカップリング分析については、先進プロジェクトの成果を受けて、EPM の機能に組み込まれた。

第２部では、EPM、コードクローン分析、協調フィルタリングを中心にその活用方法を紹介する。

先進プロジェクトで実施されたソフトウェアエンジニアリングの全体イメージを図2-1に示す。この図は、先進プロジェクト実行後に IPA が機能強化を行った EPM（65ページ参照）とその検証プロジェクトを想定しているが、分析ツールの位置付けはほぼ同じである。

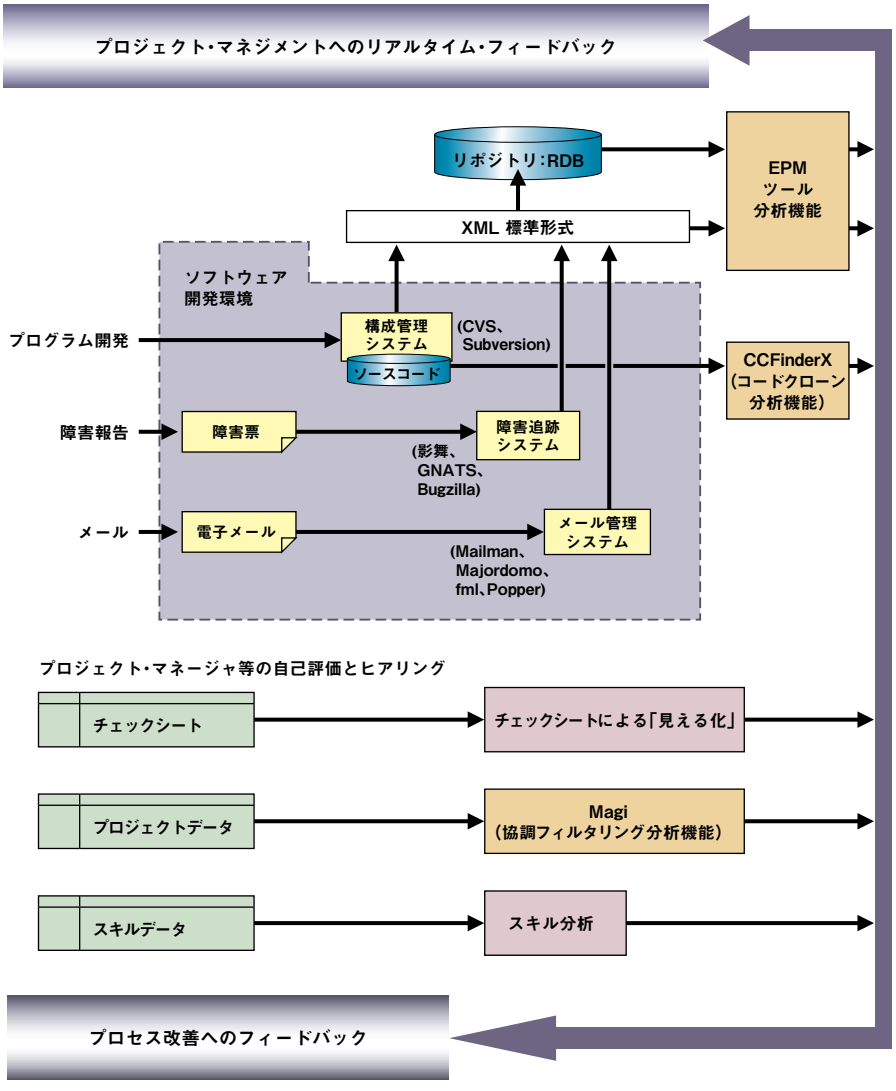


図2-1 ソフトウェアエンジニアリングの全体像

先進プロジェクトでは、EPMによるデータ分析、CCFinder（図では「CCFinderX」となっているが先進プロジェクトでは「CCFinder」を使用）を用いたコードクローン分析、Magiを用いた協調フィルタリングに基づく分析、チェックシートを用いたプロジェクトの「見える化」、そしてスキル分析によるチームスキルの把握を行った。

上記のツールによって分析された情報のうち、EPM とチェックシートによる「見える化」については、プロジェクトの進行中に分析結果をフィードバックすることにより、マ

ネジメントを支援した。また、すべての分析結果はプロジェクトの状態を把握し、プロセスを改善するための基礎データとなった。

EPMは、EASEプロジェクト（5ページ参照）で開発された、ソフトウェアを開発する過程のデータを自動収集し、分析するための環境である。ソフトウェアの開発現場で使われている構成管理システム、障害追跡システム、メール管理システムから情報を収集することにより、開発者に負担をかけずにプロジェクトの状況を定量的に、しかもリアルタイムに把握し、可視化できることが大きな特徴である。

本章では、先進プロジェクトで適用したEASEプロジェクトのEPMをもとに、IPAが機能強化したEPM（以降「EPM ツール」という）についての活用方法を紹介する。

IPA による EPM の機能強化 --- 導入の容易化、現場での活用を重点に

column

先進プロジェクトの成果を受けて、IPA では、EPM をソフトウェアエンジニアリングのためのプラットフォームとして活用できるよう機能を強化し、普及していくことにした。そのために、2006 年から EPM の機能強化を開始した。

以下に、EPM の主な機能と IPA が 2006 年に強化した機能を示す（色文字が強化機能）。なお、IPA での機能強化は 2007 年以降も継続する予定である。

■ データ収集機能
- 構成管理システム（CVS、【Subversion】）
- 障害追跡システム（GNUTS、Bugzilla、【影舞】）
- メール管理システム（Mailman、Majordomo、fml、【popper】）

■ 分析・グラフ表示機能
- ソースコードの規模推移
- 更新時期とチェックアウト数
- 累積・未解決障害件数および平均障害滞留時間
- メール投稿数と更新時期
- 更新と障害件数

■ 汎用分析機能
- 【パレート図】
- 【クロス分析】
- SQL フィルタリング
- ロジカルカップリング分析機能
- SRGM（信頼度成長曲線）表示機能

■ データ出力機能
- 【分析用データ出力機能】
  【分析用構成管理ツール】、【分析用障害追跡データ】
- 構成管理履歴詳細情報、メール詳細情報

■ システム制御
- サーバ方式：Linux サーバ＋ Web ブラウザ
- 【クライアント／サーバ方式（クライアント：Windows ＋ Eclipse）】

■ その他
- 英語環境に対応
- マニュアル、【チュートリアル】
- 【導入の容易なインストーラ】
- 【商用レベルの品質】

1.1 EPMツールでわかること

プロジェクトの進捗状況

プロジェクトを的確にマネジメントするには、開発現場の状況を常に正確に把握しておくことが必要になる。プロジェクトを遂行していくうえでは、さまざまな問題が発生するが、その問題に適切に対応するためには、プロジェクトがどのような状態にあるのかを判断し、その状態にとって最適な対策を実施することが重要である。

EPMツールは、ソースコードの規模推移、ソースコードの更新状況、障害の累積件数の推移、未解決障害数の推移等を表示することができ、プロジェクトの進捗状況を、開発者からの申告がなくても、正確に把握することができる。

プロジェクトの状況変化

プロジェクトを破綻に追い込むような問題も、その始まりは些細なことの場合がある。最初は担当者でも対処できる問題であったものが、適切に処理されなかったために、次第に大きな問題に発展し、やがてプロジェクトマネージャでも対処できなくなり、プロジェクトそのものを破綻させてしまう。問題をできるだけ早い時点で認識し、適切に対処することが、プロジェクトが失敗するのを未然に防ぐ最良の対策である。

EPMツールは、プロジェクトの状況の推移をグラフ表示することができ、グラフの変化点を検出することで、状況の変化を見出すことができる。具体的には、ソースコードの規模が急に増えたり、未解決障害件数が急増するといった変化をリアルタイムに把握することができる。

すなわち、EPMツールを用いて日々のプロジェクト開発データを収集し、分析することによって、大きな問題となりうる小さな変化を即座に把握することが可能になる。この小さな変化を見逃さずに、適切な対応をすることがプロジェクトの失敗を防ぐことに繋がる。

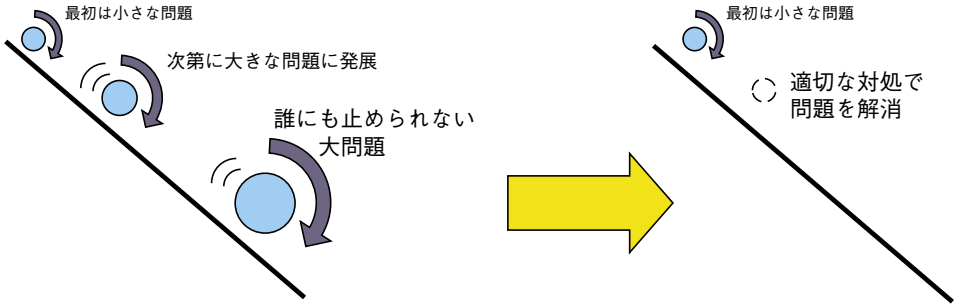


図 2-2 早期発見が問題解決への近道

プロジェクトの課題

プロジェクトが終了した時に、プロジェクト全体を振り返ってプロジェクトマネジメントやプロセスについての課題を抽出することは重要である。

EPM ツールと関連する開発環境（開発支援システム）を適切に運用することにより、プロジェクトの経緯を保存することができる。

保存した内容を分析することでプロセスの弱点や課題を見出し、改善に繋げることができる。また、プロジェクトの経緯を継続して蓄積しておくことで、過去のプロジェクトとの比較が可能となる。

EASE プロジェクトでは、EPM で収集したデータを使って、プロジェクトの状態を時間軸に沿って再現するプロジェクトリプレイヤを開発した（http://sdlab.naist.jp/prj-replayer.html）。このようなツールを使うことで、問題発生の経緯やイベントの前後関係などを鮮明に確認することができ、分析作業が効率化される。

今後は、ソフトウェア開発におけるプロセスに関するデータを蓄積し、活用することの重要性がますます高まってゆくと想定される。

1.2 EPMツールの活用

目的の明確化

EPMツールを導入する場合には、利用する目的を明確にしておくことが重要である。

EPMツールの分析結果の活用としては、大きく2つある。1つは、進行中のプロジェクトのマネジメントに活かし、マネジメントの効率化やプロジェクトの失敗を防止するために利用するもので、リアルタイムに収集されたデータを分析し、リアルタイムにフィードバックすることが重要になる。

もう1つは、プロジェクトが終了した後で、プロジェクトの経緯を分析するために活用するもので、そのプロジェクトの問題点を抽出し、プロセスの改善やマネジメント技術の改良に利用する。この場合には、改善する前と改善を実施したプロジェクトで同じデータが収集されていることが重要になる。

運用ルールの徹底

分析の精度を高めるには、EPMツールをはじめ、その入力となる構成管理システム、障害追跡システム、メール管理システムなどのソフトウェア開発環境で使われているさまざまなツール類の運用ルールを定め、それを徹底する必要がある。

たとえば、日々のソースコード規模を確認したい場合には、ソースコードの作成、修正に対して、毎日最低一度は構成管理システムにソースコードを登録する必要があり、モジュールなどの単位での進捗を把握したい場合には、モジュールのコーディングが終了した時点でモジュールごと登録する必要がある。

また、障害修正とソースコードへの影響を分析したい場合には、障害修正1件ごとに修正内容を構成管理システムに登録する必要があり、登録時には特定の障害のための修正であることがわかるように障害の管理番号などをソースコード登録時のコメントに入れるなどの対応が必要になる。

その他にもコーディング規約、メールの件名作成の規約、などの徹底が重要になる。

1.3 EPMツールの活用例

以下に、EPMツールを活用した分析例を示す。

各活用例は、EPMツールが出力するグラフの例、そのグラフを出力するための分析の観点、グラフから読み取れる現象や特徴、その現象を引き起こしている原因として考えられることと必要な対策を記述している。考えられる原因が複数ある場合には、それぞれの原因と対策を記載している。

なお、グラフの中で注目すべきところに○を付けた。

例に示すような変化を見出した場合には、原因が何かを確実に把握することが重要である。EPMツールで収集､分析された他の情報やプロジェクトメンバ等とのコミュニケーションによって原因を明らかにし、問題があれば速やかに対応する。EPMツールは状況の変化をリアルタイムに可視化するが、変化を見逃したり、対応が遅れてしまっては意味がない。

実際にEPMツールを活用し、同じ現象を把握した場合､「原因と対策」に記載されていない原因に起因するものであった時には、✎ マークのあるページの余白部分に書き込んでいき、本書をより充実したものにし、活用して欲しい。

注：本書に記載されていない原因と対策やご意見についてSEC へのご連絡をお待ちしております。SECへの連絡は Web サイト（http://sec.ipa.go.jp/index.php）からログインし（利用者登録してください）､「SECへのお問合せ」でお願いします。

ソースコードの規模推移（急増）

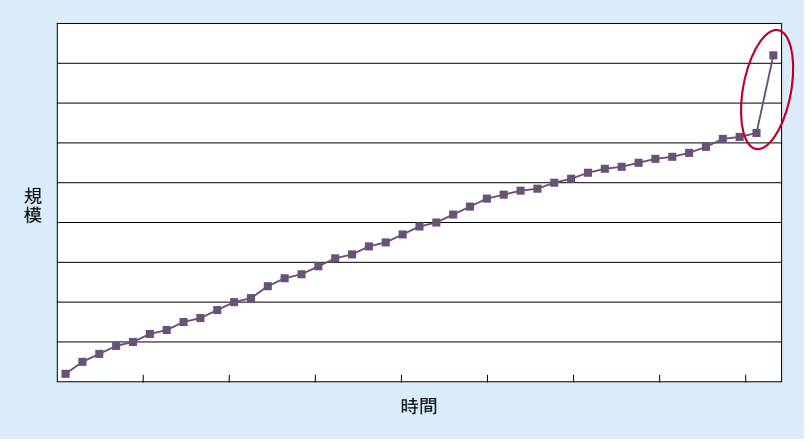


図2-3 ソースコードの規模推移（急増）

【利用する技術】

EPMツール

【分析の観点】

ソースコードの規模推移

【グラフから読み取れる現象】

規模が急増した

【原因と対策】

- 要員の増強

  当初から予定されていた要員増であり、増員数に見合ったソースコード規模の増加であれば問題ない。

  作業の進捗が遅れていることに伴う要員増の場合は、新規要員によって作成されたソースコードの品質を確認する必要がある。

  ファイル数の増加について確認したい場合には、EPMツールの分析用CVSデータ出力機能を使って情報を得ることができる。

- 新規モジュールの登録

  それまでコーディングしていたのとは別のモジュールが設計から製造に工程が進んだことによるソースコードの登録であれば問題ない。

- 流用するソースコードの登録

  流用するソースコードのレビューが終了したために、一括して登録した場合には問題ない。

  流用するソースコードの品質が担保されていることが重要。

- 大規模な障害修正

  ソースコードのレビューにより、大規模な修正が行われた結果、ソースコードの規模が大きくなった場合、修正されたソースコードが十分にレビューされていることが必要である。

- 運用ルールの不徹底

  運用ルールが徹底されておらず、本来決められた周期で修正したソースコードを登録することになっているにも関わらず、ルールを守らない担当者によって一度にまとめてソースコードが登録されている場合には、運用ルールの周知徹底が必要。

  特に、プロジェクトの途中から加わったメンバ対して、確実に運用ルールの説明を実施し、普段から徹底することを意識しておくことが必要である。

ソースコードの規模推移（急減）

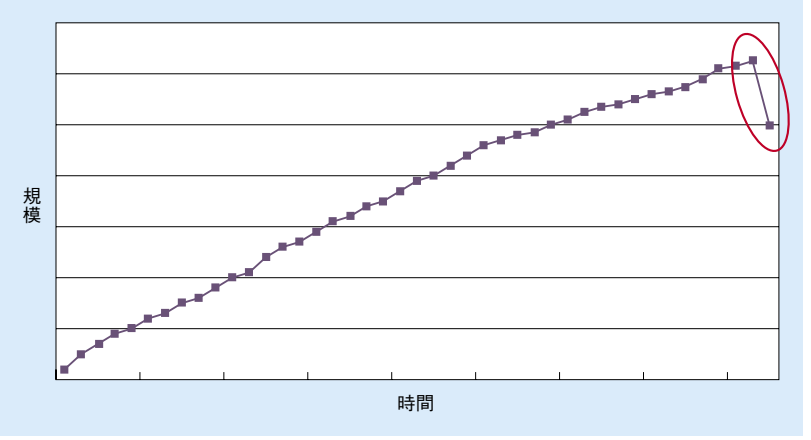


図2-4　ソースコードの規模推移（急減）

【利用する技術】

EPMツール

【分析の観点】

ソースコードの規模推移

【グラフから読み取れる現象】

規模が急減した

【原因と対策】

- ソースコードの共通化

  コードクローン分析等を用いて検出されたコードクローンを共通化したため、全体のソースコード規模が減少している場合は問題ない。

- 機能変更に伴う不要コードの削除

  当初予定していた機能が変更されたため、不要となったソースコードを削除した場合、削除部分が他のソースコードから引用されていないことを確認しておく必要がある。

- 品質の悪いモジュールの作り直し

  ソースコードレビューの結果、品質が悪いと判断され、新たな設計思想のもとでモジュールを作り直すために、すでに登録されているソースコードを削除した場合には、コーディング作業の遅れを挽回する施策の検討が必要である。

- 誤って必要なモジュールを削除

  削除する必要のないモジュールを誤って削除してしまった場合、そのモジュールの最新のソースコードを改めて登録し直す。ただし、なぜ削除してしまったのか、その原因を明らかにし、再発防止策を検討する必要がある。

ソースコードの規模推移（増減）

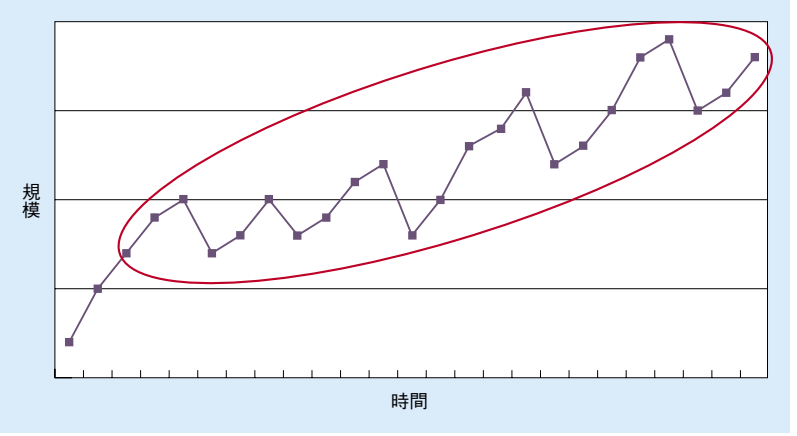


図2-5　ソースコードの規模推移（増減）

【利用する技術】

EPMツール

【分析の観点】

ソースコードの規模推移

【グラフから読み取れる現象】

規模の増減が繰り返される

【原因と対策】

- 試行錯誤

  開発を担当しているプロジェクトチームにとって、初めての分野でのソフトウェア開発で、ロジックの見直しなど、試行錯誤を行いながらコーディングを進めている場合には、ソースコードを登録する基準や試行錯誤をいつまで行うかなど、ソースコードのレビューや確定に関する承認ルールを決め、それに従って作業することが重要である。

- 仕様変更が頻発

  コーディングに入ってから仕様が変更され、それに伴ってソースコードの設計、製造をやり直している場合には、コーディング作業を中断して仕様の確定を先に行うなどの対策が必要である。

  なぜ、仕様が変更されたのか、その原因を明確にし、再発防止策を検討しておく。

- 五月雨式コードレビュー

  モジュール単位などでコードレビューを次々に繰り返す時、コードレビューによる削除行数が多い場合にはソースコード規模が増減する。ソースコードが削除された要因を明確にし、プログラム設計書の品質を向上させる、コーディング規約の見直しや周知徹底を行うなど、元々のソースコードの品質を高める対策を検討する必要がある。

ソースコードの規模推移（一定）

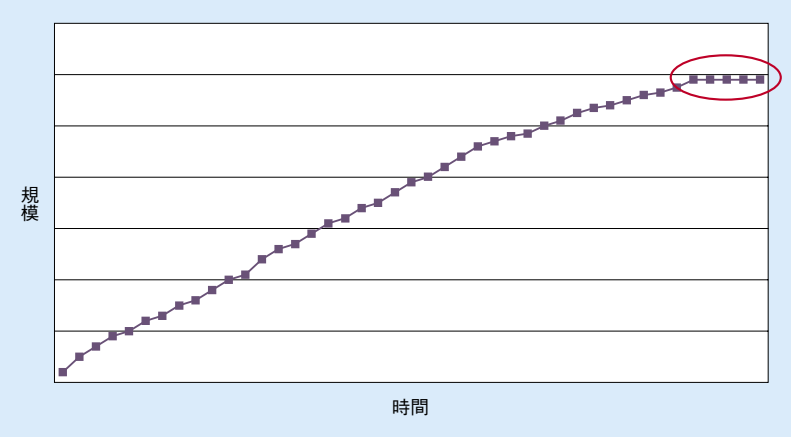


図2-6　ソースコードの規模推移（一定）

【利用する技術】

EPMツール

【分析の観点】

ソースコードの規模推移

【グラフから読み取れる現象】

規模の増減に変化がない

【原因と対策】

- 障害がなく安定

  コーディングおよびレビュー結果の反映による修正が終了し、単体試験の準備を行っている状態、あるいは単体試験が開始されたが、まだ障害が検出されていないのであれば問題ない。

- コーディングが中断

  仕様に問題があり、設計を見直しているなど、コーディング作業以前の工程に起因する問題が発生している場合は、早急にその問題を解決させる必要がある。問題の解決に伴って、ソースコードの作り直しが発生する恐れがあり、作業遅延を最小限に抑えるための検討を行う必要がある。

- コードレビューの結果が未反映

  コードレビューは終了しているが、その結果を反映したソースコードが登録されて

いない場合、修正が未完なのか、登録が漏れているのか、状況を確認して対処する必要がある。

- 長期休暇や出張

  担当者が長期休暇中や出張していてソースコードの更新が止まっている場合、それが予定されているものであれば問題ない。予定にない長期休暇や出張の場合には作業の遅れを挽回するための対応策を検討する必要がある。

ソースコードの規模推移（安定後変化）

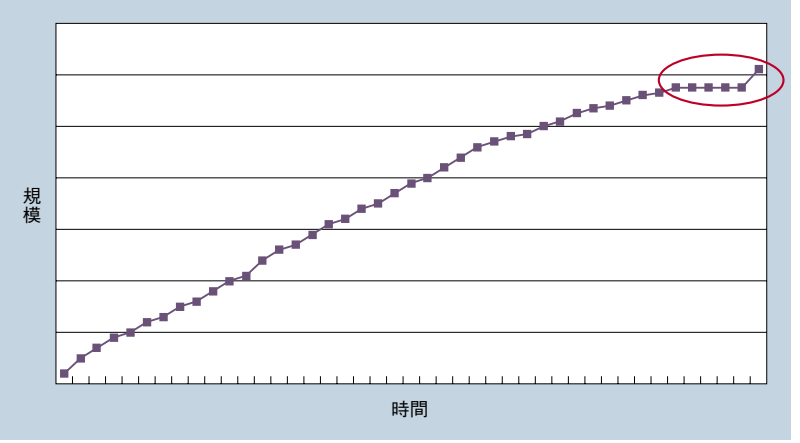


図2-7　ソースコードの規模推移（安定後変化）

【利用する技術】

EPMツール

【分析の観点】

ソースコードの規模推移

【グラフから読み取れる現象】

規模が安定後に変化した

【原因と対策】

- 試験で検出された障害の修正

  試験工程で検出された障害を修正したために多少の増減が生じた場合は問題ないが、規模が大きく変化した場合には、修正内容の確認やレビュー品質の確認が必要である。

- 仕様変更

  コーディング工程が終了してから仕様が変更され、ソースコードを修正した場合、機能設計書、詳細設計書など、ドキュメントも同期して変更されており、設計書に基づいてソースコードレビューが行われていることが重要である。また、仕様変更に伴う影響範囲の確認や調査が行われており、関係者に周知徹底されているか確認する。

グラフを重ね合わせる

column

ソースコードの規模が増減した理由を分析する上で、EPMツールが出力する他のグラフと比較することは有用である。たとえば、ソースコードの規模推移と累積障害件数の推移を比較することによって、障害の発見とソースコードの関連がわかる場合がある。EPMツールのカスタマイズ設定機能を使うと、このように比較したいグラフを同じグラフ上に表示することができる。

更新時期とチェックアウト数の関連（少ない）

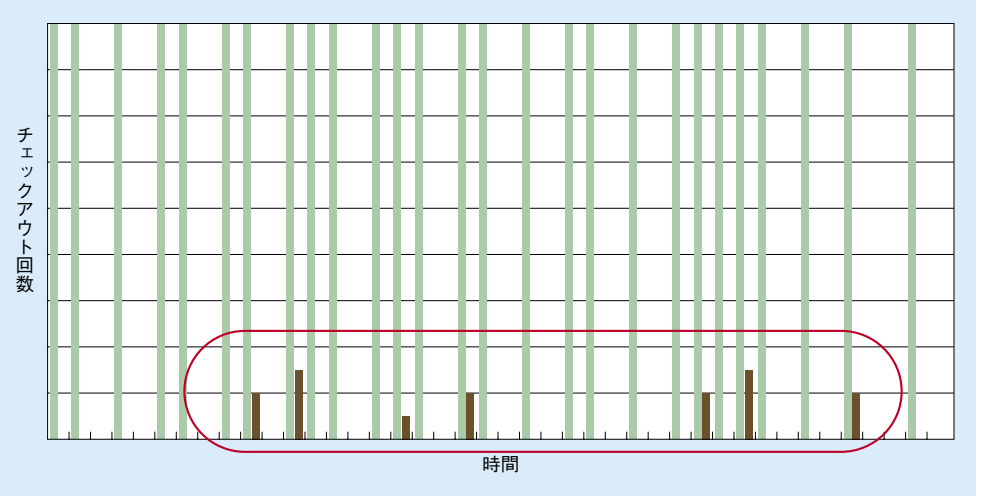


図2-8　更新時期とチェックアウト数の関連（少ない）

【利用する技術】

EPMツール

【分析の観点】

変更時期とチェックアウト数の関連

【グラフから読み取れる現象】

更新があってもチェックアウトが少ない

【原因と対策】

- 登録や更新の連絡が不徹底

  他のモジュールに影響するソースコードの登録や重要な変更が周知されていない場合、コミュニケーションルールや運用ルールの見直しあるいはルールが守られているかの確認が必要である。

- 変更の影響範囲が不明確

  他のモジュールに影響する変更が行われているにも関わらず、影響があることが知らされていない担当者がいる場合、コミュニケーションルールを見直し、影響のあるモジュール開発者に確実に情報が届くようにする。

  変更がどのモジュールに影響しているかわからない場合には、変更内容を明確にし

て周知するなど、情報共有を促進する必要がある。あるいは、既存のモジュールであれば、EPMツールのロジカルカップリング分析を行って依存性の高いモジュールやファイルを把握しておくことも重要である。

- モジュール間の影響度少

  設計上、モジュール間の影響度が少なく、更新に対して影響を受けるモジュールがない、あるいは少ない場合には、影響を受けるモジュール関係者が確実にチェックアウトしているかを確認しておく。

  ただし、設計上は影響がないのに実装方法によって影響を与えてしまっている場合も考えられるので、結合試験においては最新のソースコードをチェックアウトして使う必要がある。

- 関連モジュールの結合試験が未実施

  変更によって影響を受けるモジュールがまだ結合試験を開始していない場合であれば問題ない。ただし、インタフェースの変更など重要な情報があれば、関係者に周知されている必要がある。

更新時期とチェックアウト数の関連（急増）

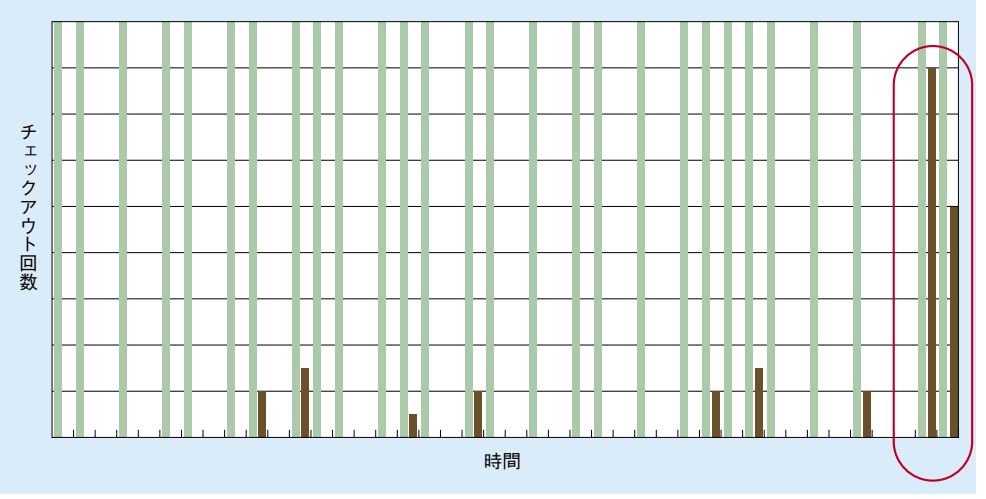


図2-9　更新時期とチェックアウト数の関連（急増）

【利用する技術】

EPMツール

【分析の観点】

変更時期とチェックアウト数の関連

【グラフから読み取れる現象】

チェックアウトが急増した

【原因と対策】

- 影響範囲の広い更新

  他の多くのモジュールに影響する重要な変更が行われた場合には、変更前との互換性があるかなど、レビューによって十分に確認が行われている必要がある。互換性がなくなる場合には、その情報が周知され、関係者間で調整が済んでいる必要である。

- 一斉に結合試験を開始

  各モジュールが一斉に結合試験を開始するために、チェックアウトを行っているのであれば問題ない。

- 要員の増加

  試験工程で要員が追加され、新たに加わった要員によってチェックアウトが行われて

いる場合、予定された増員であれば問題ないが、進捗の遅れに伴う予期せぬ増員の場合には、新たな要員に対する状況説明や導入教育が実施されていることを確認する。単に人を増やしただけでは思うような挽回は難しい場合があるので、既存要員と親規要員の役割分担を明確にして、コミュニケーションを十分確保することが重要である。

更新時期とチェックアウト数の関連（更新継続）

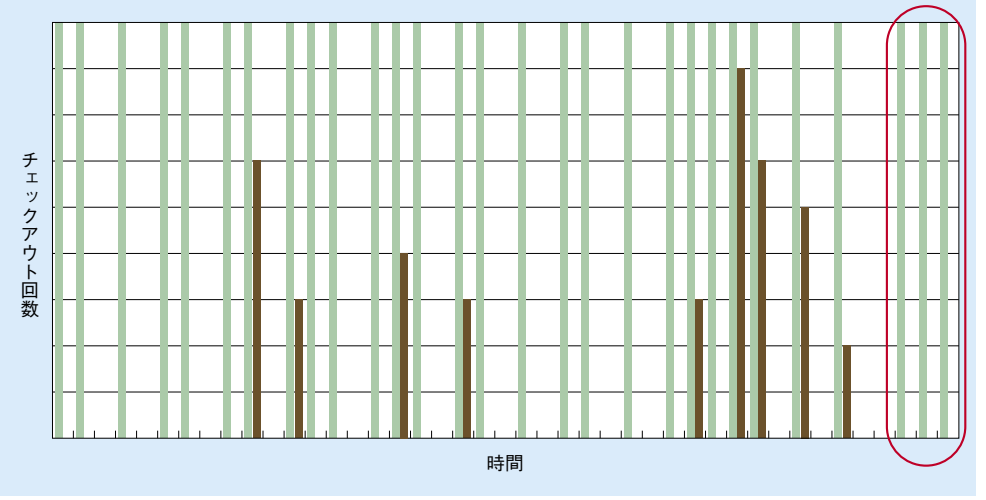


図2-10　更新時期とチェックアウト数の関連（更新継続）

【利用する技術】

EPMツール

【分析の観点】

変更時期とチェックアウト数の関連

【グラフから読み取れる現象】

更新が継続、チェックアウトなし

【原因と対策】

- 試験工程中で再評価前

  結合試験、総合試験を実施中で、検出された障害を修正したための変更だった場合、継続してテストが行われており、修正を反映したものの再評価が後日に予定されていれば問題ない。

- 他のモジュールに影響を与えない修正

  変更の影響範囲が更新単位（ファイルやモジュールなど）内に限られたものであれば問題ない。ただし、修正内容のレビューが実施され、ほかへの影響がないことが確認されている必要がある。なお、ファイル間の影響度については、EPMツールのロジカルカップリング分析によって確認することができる。

- 変更ルールが不徹底

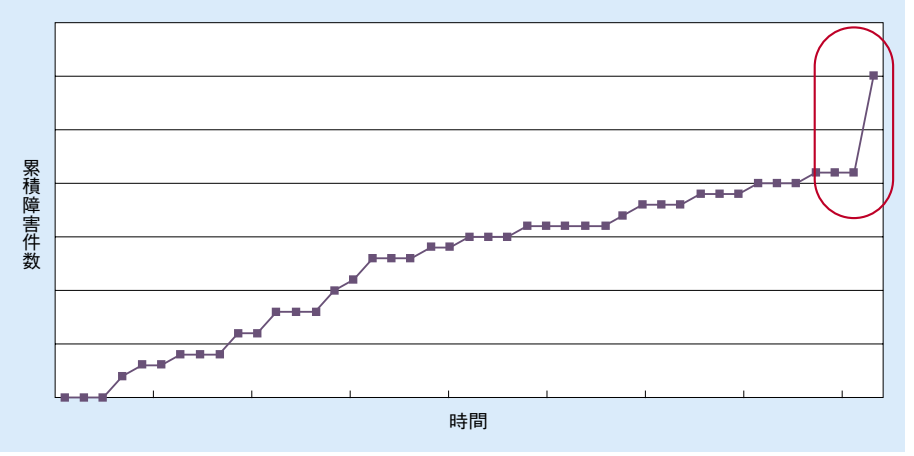
  他のモジュールに影響を与える変更が行われていて、その情報が関係者に周知されていない場合、変更に関する一連のやり方を見直し、変更ルールを周知する必要がある。

- 機能追加

  試験工程に入ってから、新たに機能の追加が行われている場合、その修正の影響範囲を正確に把握しておく必要がある。

  影響を受けるモジュールの試験を後にする、あるいは再度実施するなど、スケジュールの見直しが必要になる。

累積障害件数の推移（急増）



2-11　累積障害件数の推移（急増）

【利用する技術】

EPM ツール

【分析の観点】

累積障害件数の推移

【グラフから読み取れる現象】

累積障害件数が急増

【原因と対策】

- 試験の開始

  結合試験等が開始され、摘出された障害が登録された結果であり、累積障害件数が予定の範囲内であれば問題ない。

  予定を超えて累積障害件数が増加した場合には、どのモジュールに障害が多いか、場合によってはファイルや関数レベルで分析を行い、再レビューなどの対策を実施する必要がある。

- 試験の強化（要員の増加）

  試験工程用の要員を増加するなど、試験実施の強化が行われた場合、計画された増員で、あらかじめ準備（導入教育等）を行っていれば問題ない。試験工程の遅れを挽回する目的で、人員を投入している場合には、新たに参加したメンバへの教育や

ルール徹底が行われているか確認する。

- 品質の悪いモジュールの試験を開始

  新たなモジュールの試験が開始され、そのモジュールの品質が悪いために障害が多数検出されている場合は、コードレビューなどの対策を検討する。障害が設計に起因するものである場合は、設計書の再レビューを行う。

- 未登録障害の登録

  それまで、登録されていなかった障害を一気に登録した場合には、運用ルールがなぜ守られなかったのか、その原因を明確にし、運用ルールの見直しや周知徹底を行う。

障害情報の登録・更新

column

障害の登録や解決した時の情報変更などが正しい時期に行われていない場合、平均障害滞留時間が正しく計算できなくなる。プロジェクト状況の判断やプロジェクト終了後の分析に支障をきたすので、障害情報がある程度まとまってから一括登録や一括処理を行うのは避ける。

未解決障害件数の推移（急増）

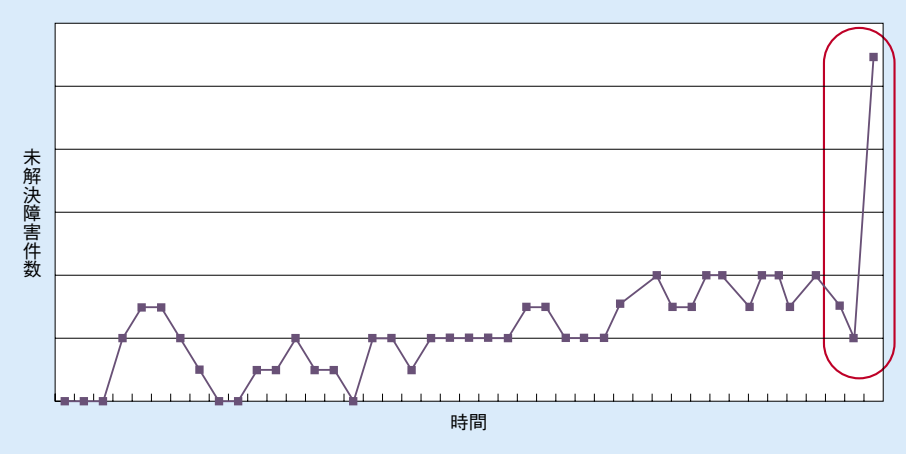


図 2-12　未解決障害件数の推移（急増）

【利用する技術】

EPM ツール

【分析の観点】

未解決障害件数の推移

【グラフから読み取れる現象】

未解決障害件数が急増

【原因と対策】

- 試験の開始

  結合試験等が開始され、摘出された障害が登録された結果であり、累積障害件数が予定の範囲内であれば問題ない。

  予定を超えて累積障害件数が増加した場合には、どのモジュールに障害が多いか、場合によってはファイルや関数レベルで分析を行い、再レビューなどの対処を行う必要がある。

- 難しい障害

  解決するのが難しい障害の対応に工数を取られている間に、新たな障害が検出されている場合、未着手の障害調査の分担を見直すなど、調査を進めるための対策を実施する。

- 担当者の交代

  障害調査の担当者が交代し、不慣れなメンバが調査を行っているために、効率が低下している場合、他のメンバとの情報共有が十分か確認し、不慣れなメンバの支援体制を整える。

- モチベーションの低下

  開発期間の長いプロジェクトなどで、所謂中だるみの状態に陥っている場合、目標の再確認や士気を高める場の設定などを検討する。

- 担当者の休暇

  障害の調査を担当するメンバが休暇などで不在となっている場合、一時的であれば良いが、長期の出張や休暇の場合には、役割分担を見直すなどの対策が必要である。

平均障害滞留時間の推移（増加）

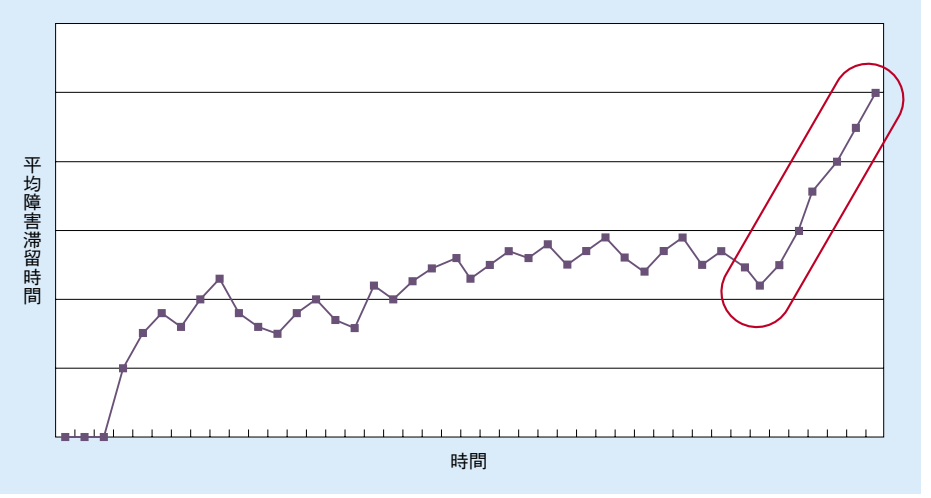


図 2-13　平均障害滞留時間の推移（増加）

【利用する技術】

EPM ツール

【分析の観点】

平均障害滞留時間の推移

【グラフから読み取れる現象】

平均障害滞留時間が増加する一方

【原因と対策】

- 障害調査の工数が不足

  他の作業に追われ障害を調査する工数が確保できていない場合は、作業分担の見直し等によって、担当者が障害を調査する工数を確保できるように調整する。

- 難しい障害

  調査あるいは解決するための検討に時間がかかっている障害があり、他の障害の調査が進んでいない場合、未着手の障害の調査を他のメンバで分担するなどの対策が必要である。

- 担当者の体調不良

  担当者が体調を崩して休暇を取っていたり、作業が進まない場合には、メンバの負

荷状況を確認し、メンバ間の負荷に大きな差がある場合には役割分担の見直しなどの対策が必要である。

更新と障害件数（追従）

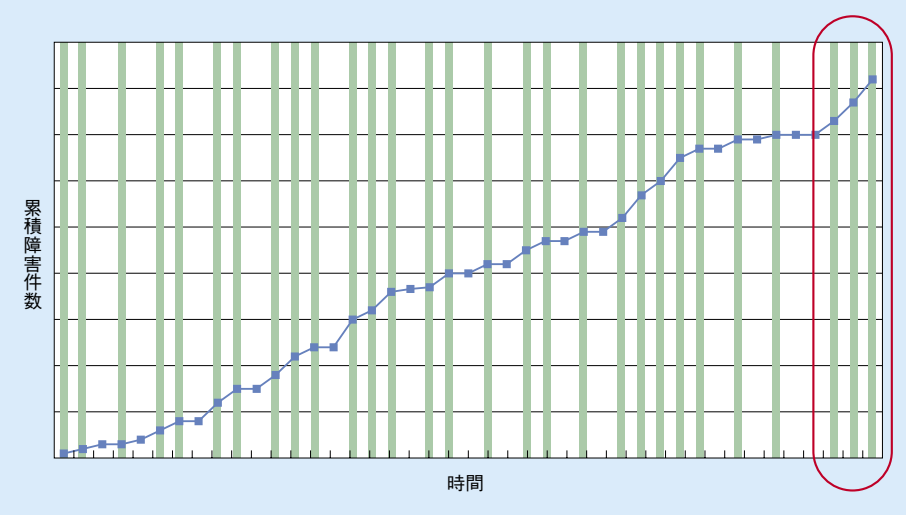


図 2-14　更新と障害件数（追従）

【利用する技術】

EPM ツール

【分析の観点】

更新時期と累積障害件数の関連

【グラフから読み取れる現象】

累積障害件数の増加に更新が追従している

【原因と対策】

- 検出された障害を順次解決

  累積障害件数が増加しているが、検出された障害を順次調査し、障害を解消するためのソースコードの変更を登録している場合は大きな問題ではない。

  検出された障害が順次解決されている場合は、平均障害滞留時間が増加していないことで判断できる。

- ソースコード修正レビューの実施

  障害を短期間で修正している可能性が考えられるが、ソースコードの修正についてきちんとレビューしているか確認する。簡単な修正であってもレビューを実施することで見落としを防止できる。

更新と障害件数（更新なし）

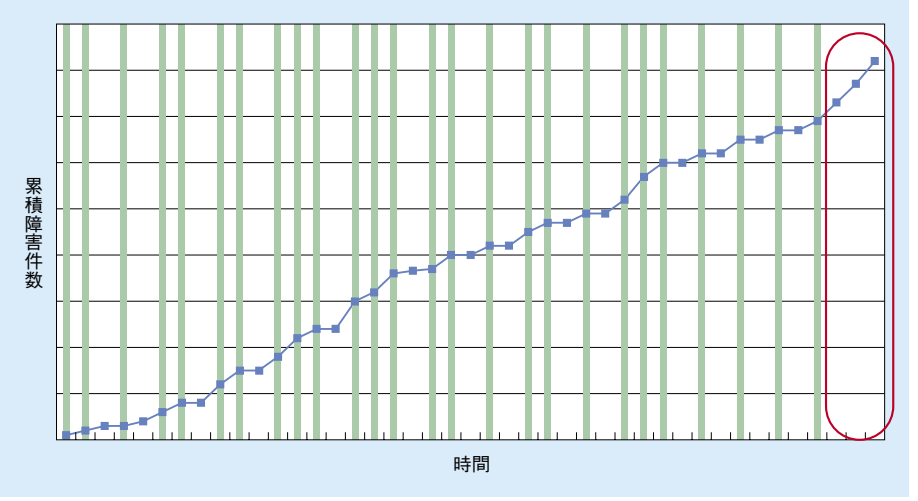


図2-15　更新と障害件数（変更なし）

【利用する技術】

EPMツール

【分析の観点】

更新時期と累積障害件数の関連

【グラフから読み取れる現象】

累積障害件数は増加しても更新がない

【原因と対策】

- 障害調査の工数が不足

  他の作業に追われ障害を調査する工数が確保できていない場合は、作業分担の見直し等によって、担当者が障害を調査する工数を確保できるように調整する。

- 難しい障害

  調査や修正方法の検討に時間がかかる障害に対応していてソースコードの変更がまだできていない場合、未着手の障害があれば他のメンバで分担して調査するなど、他の障害修正を推進させる必要がある。

- 修正の登録漏れ

  障害修正の登録漏れがないか確認する。登録漏れがあった場合には、その原因を明

確にし、作業手順を見直す。

- 試験の推進を優先

  プロジェクトの方針として、障害の調査より試験項目の実施を優先させている場合は問題ないが、平均障害滞留時間をチェックする時には注意が必要になる。また、プロジェクト終了後の分析で、障害の調査工数や調査期間を扱う場合も注意が必要となる。

更新と障害件数（更新のみ）

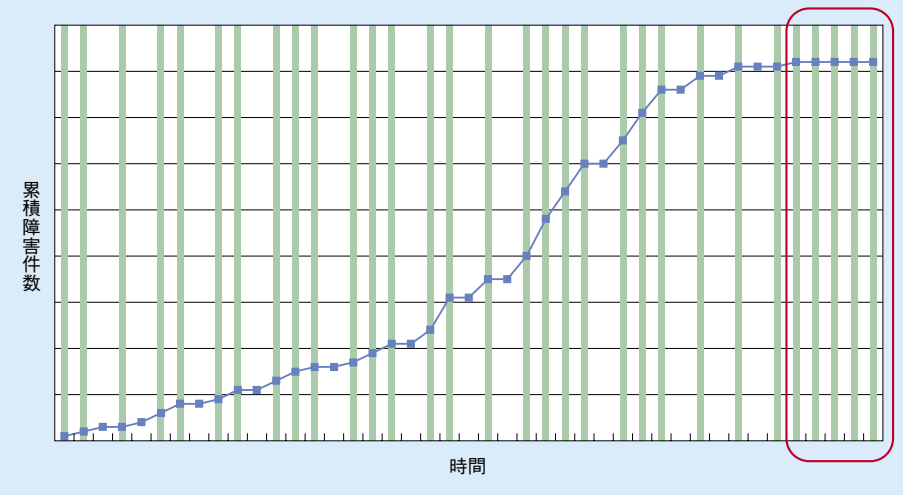


図 2-16　更新と障害件数（更新のみ）

【利用する技術】

EPM ツール

【分析の観点】

更新時期と累積障害件数の関連

【グラフから読み取れる現象】

更新が行われても障害は増えていない

【原因と対策】

- 機能追加

  新たな機能の追加要求があり、機能追加のためのソースコードの更新の場合、影響範囲を明確にしておくことが重要である。

  ただし、試験工程に入ってからの機能追加は本来あってはならない。この時期に追加しなければならない理由を明確にし、周知しておく必要がある。

- 未解決障害を解決

  すでに検出されている障害を解決するための更新が行われている場合、変更部分のレビューが実施され品質が確保されていれば問題ない。

- 原因不明

  試験工程後半になってからのソースコードの変更は、その影響範囲を明確に把握しておくことが重要である。

  障害対応ではない変更が行われた場合には、その原因と影響範囲を至急調査する必要がある。

- 承認されていないソースコードの修正

  障害修正や機能追加ではなく、性能改善などのためにソースコードを修正している場合、コードレビューを行って合意が取れている修正であれば良いが、担当者の勝手な判断でソースコードが修正されている時には、ソースコードの修正手順や基準を明確にし、周知する必要がある。

メール投稿数と更新時期の関連（メール急増）

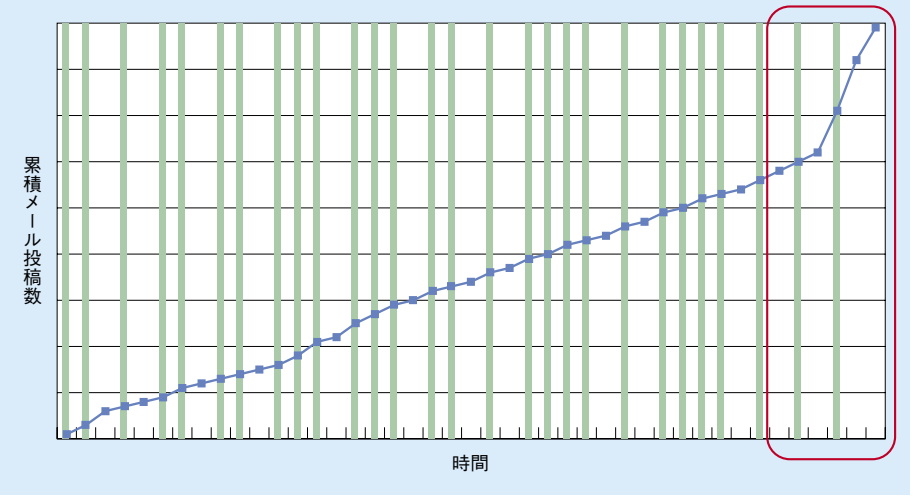


図 2-17　メール投稿数と更新時期の関連（メール急増）

【利用する技術】

EPM ツール

【分析の観点】

メール投稿数と更新時期の関連

【グラフから読み取れる現象】

更新が行われた後でメールが急増

【原因と対策】

- 変更内容について議論

  ソースコードの更新内容がメールで通知され、その内容についてメールでのやり取りが活発に行われている場合、メールの内容を確認しておくことが重要である。

  変更に対して想定外の影響があった場合には、関係者による見直しが必要になる。

- 修正ミス

  障害の修正方法が間違っていて完全に直っていない場合や新たな不具合を発生させている場合は、至急、修正内容の見直しを行う。

- 登録ミス

  修正されたソースコードの登録方法が間違っている場合には、登録をやり直すとともに、間違った原因を明確にし、手順の見直しを検討する。

メールの詳細分析

column

EPMツールには、メールに関して件名で分析する機能がある。「メール詳細情報」を選択すると、メールの件名（Topics）に関して、いつからいつまでリプライが続いたか、またその総数を表示することができる。この機能を利用して、メール投稿数が増加した時期に何についてメールがやりとりされていたのかを確認することができる。

パレート図（障害原因）

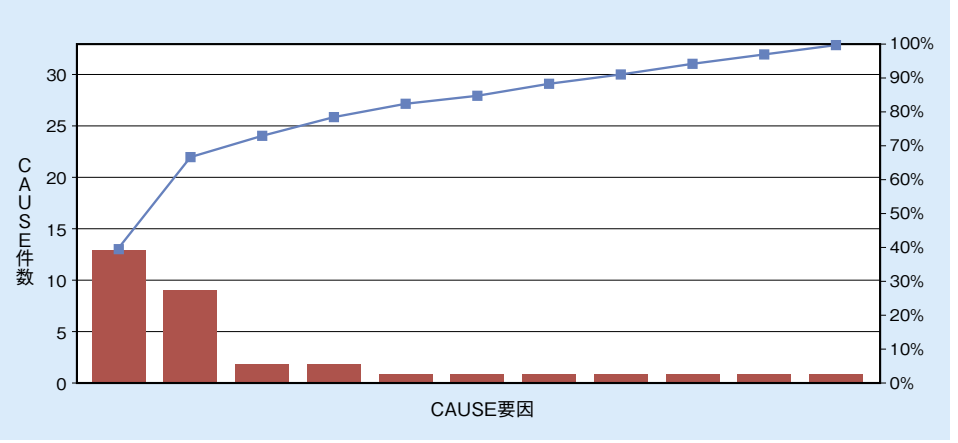


図2-18　パレート図（障害原因）

【利用する技術】

EPMツール

【分析の観点】

障害原因ごとの障害件数

（パレート図選択で「BUG CAUSE」を選択）

【グラフから読み取れる現象】

2つの原因が障害全体の70%を占める

【原因と対策】

- 原因の分析と対策を検討

  パレート図で示される原因の上位2つで、全体の70%を占めることから、この2つの原因に対して対策を検討する。

- 障害原因のグルーピング

  3番目以降の障害原因は、それぞれ単独では数が少ないが、グルーピングが可能かどうかを検討する。グルービングできる場合には、グループとしての対策を検討する。

- 他の要因による分析

  上記パレート図では件数で分析しているが、影響度や修正工数を加味した検討も必要である。件数が少なくても、修正のために多大な工数がかかる障害があった場合

には、そのような障害に対する再発防止策の検討は重要である。

重要度を考慮した分析

column

EPM ツールには、SQL フィルタリング機能があり、分析の対象となるデータを選別することができる。たとえば、パレート図による分析を行う時に、全ての障害ではなく、重要度が高い障害のみを対象として、その原因を分析したい場合には、SQL フィルタリングで重要度が大の障害のみを抽出する設定を行えばよい。EPM ツールでは、あらかじめいくつかの抽出条件を用意してあるが、抽出条件を自分で設定することも可能である。分析対象を絞り込むことにより、より深い分析が可能となる。

クロス分析（混入工程ー発見工程）

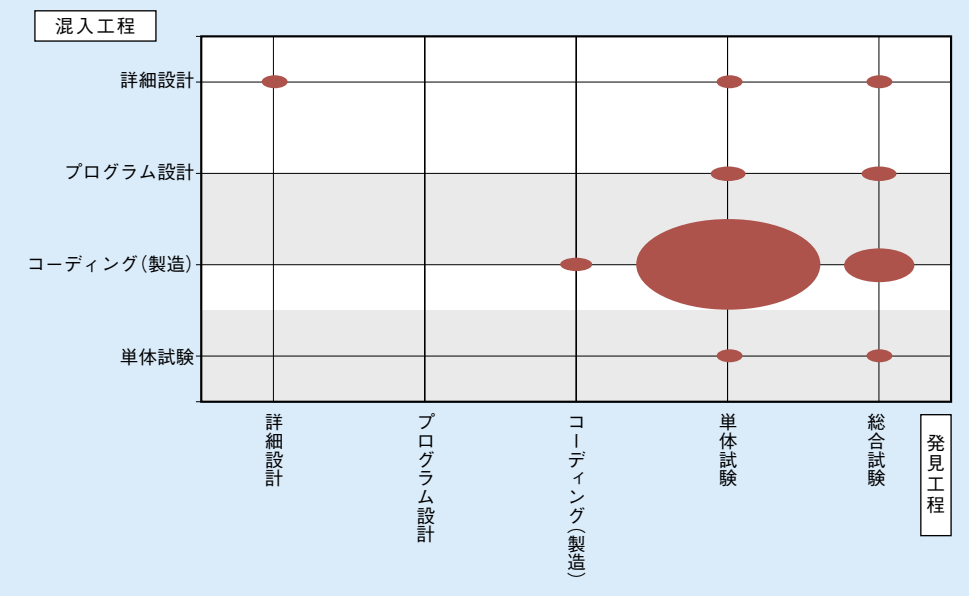


図2-19　クロス分析（混入工程ー発見工程）

【利用する技術】

EPMツール

【分析の観点】

障害が混入した工程と障害を発見した工程の関連（クロス分析）

（集計項目選択で「混入工程」と「発見工程」を指定、グラフの種類で「バブルグラフ」を選択）

【グラフから読み取れる現象】

コーディングで混入し、単体試験で発見される障害が最も多い

【原因と対策】

- コーディング工程での障害摘出不足

  コーディング工程で混入した障害の多くは単体試験、結合試験で検出されており、コーディング工程での検出が非常に少ない。コーディングレビューの実施状況や実施方法を確認し、コーディング工程で検出する障害を増やす工夫を行う。

- コーディング工程で混入される障害が多い

  コーディング工程で混入された障害がほとんどであることから、障害の内容を分析

し、コーディングで障害を混入させないための対策（コーディングルールの見直し、コードレビューの強化など）を検討する必要がある。

- 2工程以降後での発見

  詳細設計やプログラム設計で混入した障害は、プログラム設計や次のコーディングで検出されずに、2工程以降の単体試験や結合試験で検出されている。一般に混入工程と発見工程が離れるほど修正のための工数は多くなる。プログラム設計とコーディングで発見できなかった原因を明確にし、対策を講じる必要がある。

Section 2

ソースコードの類似部分を把握するコードクローン分析

本章では、CCFinderX を使ったコードクローン分析とその活用方法を紹介する。

なお、CCFinderXは、産業技術総合研究所の神谷年洋氏が開発したコードクローン検出ツールで、以下のWebサイトで公開されている。

http:/www.ccfinder.net/ccfinderx-j.html

2.1 コードクローン分析でわかること

CCFinderX を利用すると、ソースコード中のどこにコードクローン（類似したコード）があるのかを把握することができる。対応している言語は、Java、C/C++、COBOL などで、実際のソースコードを確認しながら分析することができる。

また、コードクローンがどの程度含まれているかなどを定量的に把握することができる。

2.2 コードクローン分析の活用

もっとも有効な活用法は、検出された類似部分を共通化することである。

共通化できない場合でも、ソースコード中の類似部分を把握しておくことで、保守を効率化できる。具体的には、障害があってソースコードを修正する場合、修正箇所のコードクローンに対しても同じ修正が必要となる可能性があるため、ソースコードの確認を行う必要がある。この時、あらかじめコードクローンがどこにあるのかを把握していれば、修正の確認が効率化される。

一般的にはコードクローンは少ない方が良いが、コードクローンがすべて悪いとは限ら

ない。類似した処理であっても性能上の問題から共通化しない場合も考えられる。また、将来的に処理を変えることがわかっていて、現状はあえて同じコーディングにしてある場合もある。

いずれにしても、保守や将来の変更を考えてコードクローンを把握しておくことは重要である。

2.3 コードクローン分析活用例

以下にコードクローン分析の活用例を示す。

コードクローン分析

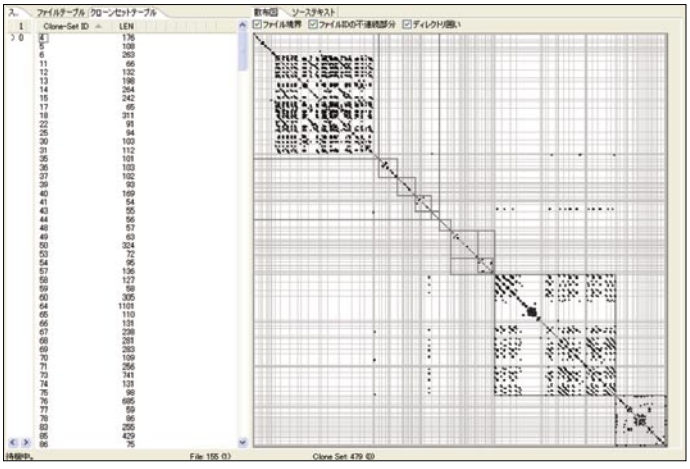


図 2-21　コードクローン分析

【利用する技術】

CCFinderX

【分析の観点】

製品全体のコードクローン分析

【グラフから読み取れる現象】

コードクローンの多いコンポーネントと少ないコンポーネントがあり、コンポーネント

間のコードクローンはほとんどない

【原因と対策】

散布図では、ソースコードが格納されているフォルダ間に太い線が引かれる。コンポーネント単位にフォルダを分けてソースコードを格納することで、コンポーネント単位でのコードクローンを見ることができる。

コンポーネントによってコードクローンの有無が大きく異なるため、コードクローンが多いコンポーネントは何らかの特徴を持っていることが考えられる。それぞれのコンポーネント内のコードクローンを調査し、共通化可能かどうかを検討する。

コードクローン分析（コードレビュー）

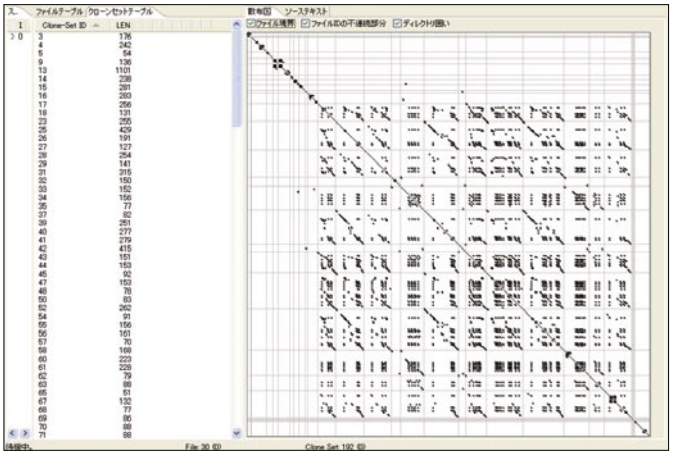


図2-22　コードクローン分析（コードレビュー）

【利用する技術】

CCFinderX

【分析の観点】

コンポーネント単位でのコードクローン分析

（コードレビュー時にコンポーネント単位でコードクローン分析を行う）

【グラフから読み取れる現象】

ファイル内およびファイル間でのコードクローンが多い

【原因と対策】

特定のコンポーネントにおいて、前処理、後処理など、同じ処理が複数のモジュール内に記述されている場合には、共通化可能かどうかを検討する。

スキルの未熟な技術者や、該当するコンポーネントに不慣れな技術者（新しく担当となった技術者）などが、新規にコーディングする場合には、そのコンポーネント内の類似処理のコーディングをコピーして作成することが考えられ、コピーを繰り返すとコードクローンが増加する。コーディングレビューで、コンポーネント内のコードクローン分析結果を参照することにより、処理の共通化を促し、コンポーネント内の構造の見直しや再構築についての検討を促進させる。

コードクローン分析（複数リリース）

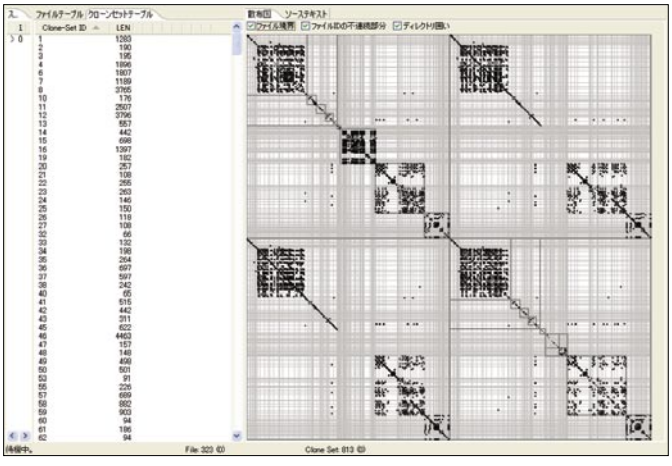


図2-23　コードクローン分析（複数リリース）

【利用する技術】

CCFinderX

【分析の観点】

連続したリリース間でのコードクローン分析

（直前のバージョンとのクローン分析を行う）

【グラフから読み取れる現象】

前バージョンから流用したコンポーネントと新規に作成したコンポーネントがあり、流

用したコンポーネント内のコードクローンはそのまま温存されている

【原因と対策】

直前のリリースのソースコードと今回リリースするソースコードをそれぞれ別のフォルダに入れ、その親フォルダを指定してコードクローン分析を行うとリリース間のコードクローンがわかる。

2つのリリース間で流用した部分はコードクローンとして表示され、作り変え、あるいは新規に追加されたコンポーネントに対しては、一般的にリリース間のコードクローンは少なくなる。

前リリースからの流用、新規作成について、計画通りになっていれば問題ないが、予期せぬ流用や予定にない作り直しが行われていないか確認する必要がある。

バージョンアップ等が計画された場合には、直前リリースのコードクローン分析を行い、コードクローンをなるべく減らす対策を盛り込んだ開発計画とし、保守性を高めて行くことが重要である。

Section 3

類似データから値を推測する協調フィルタリング分析

先進プロジェクトでは、EASE プロジェクトが開発した EASE CF 法による分析を行った。フェーズ２においては、協調フィルタリング分析ツール「Magi」を使ってプロジェクトの全体工数を見積もった。本章では、Magi による分析について簡単に紹介する。Magi の使用方法や出力結果の詳細については、Magi に添付されている利用説明書を参照のこと。

Magi への入力データは、カンマで区切られたデータ列（CSV）で、先頭行にデータ名称を並べたもので、各行の最初のデータがプロジェクトを識別するデータ（プロジェクト名、プロジェクト ID 等）と認識される。また、先頭行に並べられたデータ名称のうち、どのデータを対象として見積りを行うかを指定する（分析対象の CSV ファイルを指定すると、その先頭行のデータ名称が選択肢として表示されるので、その中から指定する）。

過去のプロジェクトで収集されているデータがあれば、それを同じ順番にカンマで区切って並べ、最初の行にデータ名称を並べる。収集されていないデータ項目がある場合は、空欄にしておく。進行中のプロジェクトがあれば、そのプロジェクトのデータを２行目（データ名称の次の行）に記入し、実積値以外は計画値を記入しておく。見積もり対象のプロジェクトが複数ある場合には、３行目以降に入力しておく。見積り対象データは何行目にあっても良いが、対象プロジェクトを指定する時に最初の方にあった方が見つけやすいためである。

3.1 協調フィルタリングでわかること

Magiにはいくつかの機能があるが、ここでは、「データ品質診断」と「見積もり」について説明する。

データ品質診断

データ品質診断機能を使うと、蓄積されているデータが見積もりの基礎データとして使えるかどうかを判定することができる。診断する項目としては、データの件数、ばらつき、見積もりに役立つ項目数、欠損率などである。それぞれの診断項目は10段階で評価され、評価結果の説明とともにレーダーチャートで表示される。

ばらつきが大きい場合には、見積もろうとしているプロジェクトの特性に合わせて、データ品質診断結果が良くなるよう過去データを取捨選択することも検討する必要がある。

以下にデータ品質診断結果の例を示す。

診断結果

診断項目	実績値	品質レベル	説明
プロジェクト件数	81	7	データは百件前後のプロジェクトを含んでいます。このデータを見積もりや統計分析に利用できます。見積や分析の結果を現場へフィードバックし、収集活動を助長しましょう。
平均誤差（見積精度）	50%	4	このデータを用いた場合、ときどき誤差が大きくなります。それを踏まえた上で、見積値を参考程度に利用できます。欠損率の低減に重点を置きましょう。
誤差のばらつき（見積安定性）	63%	6	このデータを用いた場合、誤差わずかにばらつきます。明らかに疑わしい場合を除き，見積値は信頼できます。真偽が定かでない値を削除・修正し、さらに改善を進めましょう。
的中率	47%	7	このデータを用いた場合、見積もりがよく的中します。その可能性は高く、実用レベルに達しています。データ品質を改善することで、さらに精度向上が期待できます。
見積に役立つ数値データ項目数	8	8	データが見積もりに役立つ数値データ項目をかなり多く含んでいます。仕様書や設計書の総ページ数、成果物の画面数、ユースケースのアクタ数も役立つことがあります。
数値データ項目の欠損率	0%	10	役立つ数値データ項目はほぼ完全に収集されています。他の診断項目の改善に努めると同時に、定期的に収集ポリシや項目を見直し、データ収集の形骸化を防ぎましょう。
見積に役立つ非数値データ項目数	9	4	データが見積もりに役立つ非数値データ項目を少し含んでいます。お客様の業務種別や分野など、お客様の特徴を表すデータも見積もりに役立ちます。
非数値データ項目の欠損率	0%	10	役立つ非数値データ項目はほぼ完全に収集されています。他の診断項目の改善に努めると同時に、定期的に収集ポリシや項目を見直し、データ収集の形骸化を防ぎましょう。

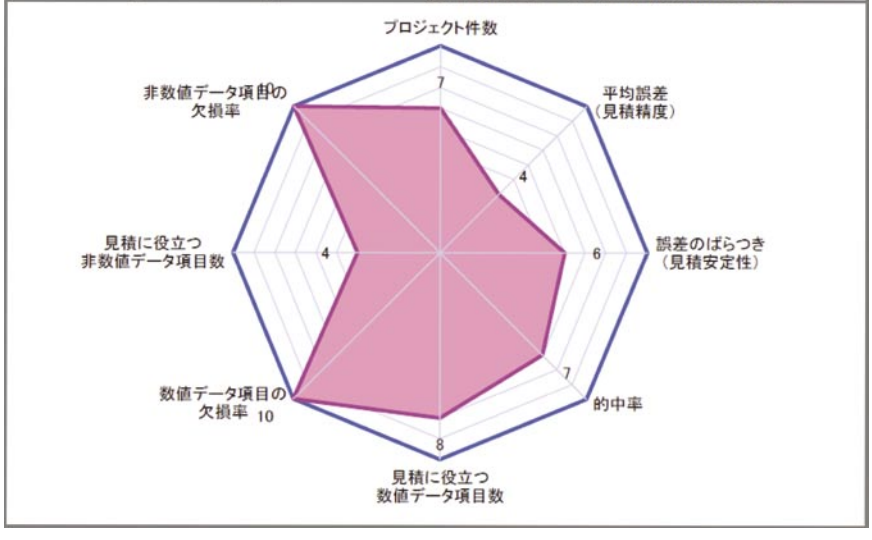


図 2-24　Magi によるデータ品質診断例

見積もり

入力されたデータ項目のうち、数値データについては他のデータをもとにして見積もることができる。実績として残っている過去データの値を「見積もる」ことで入力データに

よる見積もりの精度を検証することができる。進行中のプロジェクトに対して見積もりを行う場合には、見積もろうとするデータ項目に計画値を入力しておくことで、計画値と見積もり値の違いが明確になる。

以下に見積りを行った例を示す。

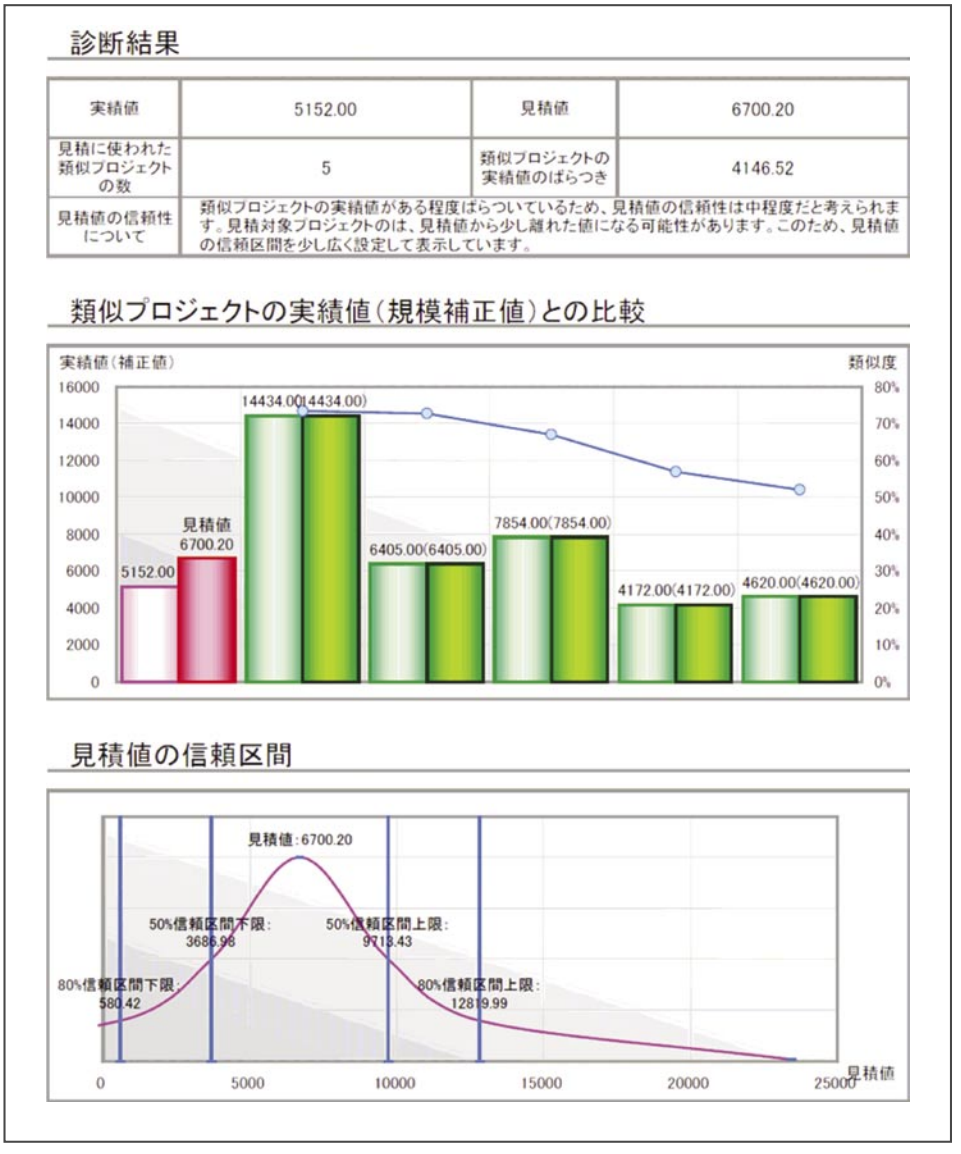


図2-25　Magiによる見積もり例

協調フィルタリングの活用

Magiを使った見積もりを行うにはデータの蓄積が必要不可欠であるが、精度の良い見積もりを行うには、蓄積されたデータのばらつきが重要になる。Magi は「見積もろうとするプロジェクトと類似しているプロジェクトを抽出し、類似しているプロジェクト」を使って値を推測するため、類似しているデータが多いほど精度が良くなるといえる。すなわち、過去データが多いほど精度良い見積もり結果が得られる可能性が高い。

ソフトウェア開発データは、開発する対象システムや開発メンバによって左右される場合が多いため、なるべく多くのデータを正確に、詳細に収集し、蓄積して行くことが非常に重要になる。

なお、SEC で収集しているベンチマークデータの収集フォームは『ソフトウェア開発データ白書』の付録に記載されているので、これからデータを収集する場合には参考として欲しい。

協調フィルタリングの活用例

以下に、Magi を使った活用例を示す。

プロジェクトに必要な全工数

【利用する技術】

Magi

【分析の観点】

協調フィルタリングによる全体工数の見積もり

【見積もるデータ】

プロジェクトに必要な全工数

【活用方法】

先進プロジェクトでは、基本設計までの実積工数と開発規模、工数などの計画値を入力し、全工数を見積もった。見積もりに使う過去データとしては、SEC で収集したベンチマークデータを利用した。

同じメンバによる開発を何度か行っている場合、その開発におけるデータが残っていれば、それを利用するのが最も簡単で精度も良いと思われる。同じメンバによる開発がな

かった場合でも、同じ部門の過去データや同じ業種における類似のシステム開発データなどを収集しておくことが重要である。

見積もりを行う時期としては、プロジェクトの計画時に開発規模などの計画値をもとに行うのが一般的であるが、工程が進むにつれて、工程ごとの実績工数を入れて見積もり直すことにより、より精度の高い見積もり値を得ることができる。

特定の工程に必要な工数

【利用する技術】

Magi

【分析の観点】

協調フィルタリングによる工程別工数の見積もり

【見積もるデータ】

特定の工程に必要な工数

【活用方法】

過去のプロジェクトデータとして工程ごとの工数が収集されていれば、工程ごとの工数を見積もることができる。

工程ごとに要員数が異なる場合には、要員数や期間などのデータも収集されていることが重要である。反復型開発の場合には反復回数も重要なデータである。

プロジェクト開始時には、工程ごとの計画値を入力して見積もりを行い、工程が進むにつれて実績値に置き換えて見積もることにより、徐々に精度が良くなる。見積もり値が計画値より大きくなった場合には、納期を考慮して、場合によっては要員を増強するなどの対策を検討する必要がある。

プロジェクトの開始から終了までの期間（工期）

【利用する技術】

Magi

【分析の観点】

協調フィルタリングによる工期の見積もり

【見積もるデータ】

プロジェクトの開始から終了までの期間（工期）

【活用方法】

工期を見積もることで、納期遅延等のリスクを回避するための早期対応を可能にする。工期に影響を与えると思われる要因に関するデータ（たとえば、仕様変更の回数や設計書の品質など）も収集されていると良い。

製造工程以降の障害数

【利用する技術】

Magi

【分析の観点】

協調フィルタリングによる障害件数の見積もり

【見積もるデータ】

製造工程以降の障害数

【活用方法】

設計工程終了時に、製造工程以降で検出すべき障害数を見積もることで、障害検出目標の設定や見直しを検討することができる。製造工程以降に検出される障害数に関係すると思われる設計工程での障害検出数も収集されていると良い。

第3部

プロジェクトを支えた人々からのメッセージ

第3部の記事は、2007年3月から4月にかけて行われたインタビューによってまとめています。
登場する方々の所属等はインタビュー当時のものです。

実践をめざしたソフトウェアエンジニアリングの研究開発で最先端の成果を実現

Interview 1

2007年2月22日、東京・青山TEPIAにて、ITS（Intelligent Transport Systems）向けの画期的なプラットフォームソフトウェア＝「リアルタイム・プローブ情報プラットフォームソフトウェア」（以下プラットフォームソフトウェア）のデモンストレーションが実施された。ソフトウェアエンジニアリング技術研究組合（COSE＝COnsortium for Software Engineering）を中心に産学官共同で進められたこのプラットフォームソフトウェアの開発プロジェクトを、「官」の立場から率いた経済産業省商務情報政策局情報処理振興課の安田篤氏に、政策的な側面から、このプロジェクトの意義と成果について、お話をうかがった。

経済産業省商務情報政策局
情報処理振興課
課長補佐
安田篤氏

"理論"を"実践"に生かす政策的テーマ

── まず、今回のプロジェクトで目指した最も重要なポイントについて、政策的な観点からお聞かせください。

今回のプロジェクトには2つのポイントがあって、1つは産学官が連携し、かつ異業種で構成する研究組合型マルチベンダーという環境で、実用性があり同時に品質と精度の高いソフ

トウェアを開発するという点。もう1つは、このソフトウェアの開発を通じて、ソフトウェアの開発手法、すなわちソフトウェアエンジニアリングの手法も確立したい、という点です。

ソフトウェアをいかに効率よく開発するかというソフトウェアエンジニアリングの手法は、ソフトウェア・エンジニア・リングセンター（SEC）と経済産業省が共同して2004年度から開発に取り組んできたわけですが、具体的な"製品"としての成果物がないと、やはり"理論"に過ぎないという部分はどうしても残ります。その"理論"であるソフトウェアエンジニアリングを"実践"にどう生かすことができるのか、という点が、まさに今回のプロジェクトの政策的なテーマでした。

また、複数のベンダーが集まって1つの目的に沿って開発するということは、いうまでもなく、ソフトウェア開発のシチュエーションとしては非常に難易度の高い状況です。今回のプロジェクトでは、そういう状況を敢えて構築した上で、ソフトウェアエンジニアリングの手法を磨き上げていきたいということも考えました。

なお、産学官の連携については、「産」の部分はCOSEに参加されているNTTデータ、デンソー、トヨタ自動車、日本電気、日立製作所、富士通、松下電器産業の7社（50音順）で、これら7社の異業種ベンダーがプラットフォームソフトウェア開発に当たりました。そして、こうしたマルチベンダー環境でいかに効率的かつ信頼性を保持して開発を進められるような基盤を整えるか、といったソフトウェアエンジニアリングの部分を、「学」の立場で奈良先端大学と大阪大学が、「官」の立場でSECと経済産業省が参加するという形です。

―― その具体的な成果物として、ITS向けのプラットフォームソフトウェアを選択した理由は。

どんな分野のソフトウェアを開発するかについては、いくつか選択肢がありましたが、せっかく実践に結び付けるなら、「より社会的な基盤となるようなもの、国民生活の利便性の向上につながる分野を」と指向した結果ですね。先進諸国同様、日本でも研究開発が進められ、カーナビゲーションシステムやETCなどの要素技術のいくつかは実用化され普及しているわけですが、プローブ情報システムについては、まだまだ研究開発を続ける余地はたくさんある。そこで今回、取り組んでみた次第です。

外部の人間からも見えやすいプロジェクト

―― 今回のプロジェクトがどんな風に進んだのかを、お聞かせください。

プロジェクトは全体的には、2004年度後半から2005年度にかけてを第1期、2006年度を

第2期として進めました。そして、第1期、第2期ともに、設計から開発、テストに至るいわゆるソフトウェア開発のV字モデルを1回ずつ回しました。第2期は、もちろん第1期の反省点、改善点を踏まえました。

そうしてPCDAサイクルを2回回すというソフトウェア開発プロセスを採ったことで、開発成果物であるプラットフォームソフトウェアの品質・精度向上はもとより、ソフトウェアエンジニアリングの高度化をも実現できたと思います。

── ソフトウェアエンジニアリングの実践強化という面では、どんな点が大きな成果とお考えですか？

EPM（Empirical Project Monitor）を導入し、その効果が確認できたことです。プロジェクトの第1期、第2期ともに、先に述べたマルチベンダー環境での開発進捗状況をSEC、奈良先端大学と大阪大学のチームがリアルタイムで取得・分析し、障害の発生などを時系列で"見える化"していくことで、マルチベンダー開発環境でありがちな"潜在的なリスク"を極力回避することができました。

ほかにも、プログラムの中で同じようなコードが使われているかどうかを分析するコードクローン分析や、ITスキル標準（ITSS）に基づく人材のスキル分析など、さまざまな角度からソフトウェアエンジニアリングの手法を磨き上げることを試みましたし、類似プロジェクトの累積データから、工数見積の予測なども行いました。

こうした成果は、青山TEPIAでの公開デモでも、プロダクト自体と同様に、大きく注目していただけたようです。異業種のマルチベンダーという開発環境の中で、どのベンダーのどのPMにもリアルタイムで状況がわかるような形を構築したことで、プロジェクトの外部の人間が見ても、プロジェクトの中でなにが起こっているのかが非常に見えやすくなったと考えています。

ソフトウェアエンジニアリングという分野は、一般的にはなかなかわかりにくいものだけに、こうした点も今回のプロジェクトの大きな成果といえると思います。

事業規模、実用的な出力、処理スピードという3つの目標をクリア

── 成果物であるプラットフォームソフトウェアの精度や品質については、開発に当たってどんな課題を設けたのでしょうか。

最大の課題は、ITSが実験段階から実用段階に限りなく近づいているということの実証で

した。つまり、現実の交通状況にできるだけ近いボリュームのデータを取得することと、現実の交通状況の改善に役立つアウトプット品質、そしてそのリアルタイム性を実現するということです。

そのために、まず「東京23区内で約8,500台のデータを取得」という目標を設定しました。厳密にいうと、都内の約20km四方という範囲ですが、その中を走るタクシー、バス、配送車約8,500台の走行データをリアルタイムに取得するというわけです。

そしてもう1つ、「取得データを3分半で分析し、実用的な情報に形を変えて出力する」という目標を立てました。実際、10分前、20分前といった情報では、実際の東京の交通状況には役に立たないですから。そうした現実をしっかり認識し、デモだからといって"参考値"ではない"実践的なアウトプット"を目指した次第です。

ちなみに、「東京23区内で8,500台のデータを取得」というプローブ情報システムの規模は、単位面積当たりのプローブカー台数という観点でいうと、世界に類を見ないものだと思います。

── その課題はクリアされましたか？

先に申し上げたように、2回のV字開発モデルを回したことで、最終的なプログラムの精度は飛躍的に向上し、事業規模、実用的な出力、処理スピード（リアルタイム性）という3つの目標はなんとかクリアできたと思います。

たとえば、実用的な出力と処理スピードというテーマについては、「A地点からB地点までどれくらい時間がかかるか」という情報や渋滞情報はもちろん、「右折車線が混んでいる」「直進車線は空いている」といった車線別の情報の出力も実現しているし、さらに前後の交通状況からの推測も高度に行えるようになっています。

こうした諸性能については、公開デモでもやはり高く評価していただけたと思います。公開デモでは、このシステムを搭載したマイクロバスを実際に走行させたりもしましたので、まだ完全には実用化されていないプローブ情報システムという分野のことをよく知らない方にも、かなりのインパクトを与えることができたのではないでしょうか。公開デモの際に取ったアンケートでも7割近くの方が、今後の実用化に期待を寄せていらっしゃいます。

現実の産業に受け入れられる構造を作ることが課題

── 今後、このプロジェクトをどんな風に発展させていこうとお考えですか？

政策的な観点から今後の展開を考えた場合、まずはやはり、現実のソフトウェア産業にいか

に活用していただくかということが大きな課題だと思います。ソフトウェアエンジニアリングの手法も、その成果物であるプラットフォームソフトウェアも、共にすでに実用レベルにはあると思いますので、次はやはり、ソフトウェアベンダーなどの開発企業が実際に営利を追求するための開発環境として、あるいはITSのプラットフォームとして、大いに活用していくような構造を作っていくことを考えなければならないでしょう。

また、今回はたまたまITSという分野でのソフトウェア開発だったわけですが、もっと別の分野の開発プロジェクトにも導入して手法の評価を行いたいという気持ちもあります。

── 最後に、今回のプロジェクトの成果に対する満足度をお聞かせください。

研究して理論的な部分を追求するだけではなく、実践を目指したという意味では、ソフトウェアエンジニアリングの研究開発では政策的にはじめての試みですから、今回の成果は最先端のものであり、その意味では非常に満足感、達成感があります。

そもそも2004年10月にSECが設立された際の政策的な意義が、「産学官が連携して日本のソフトウェア工学の拠点として活動していく」ということだったことを想起すれば、今回のCOSEのプロジェクトは、そもそもの目標の価値ある具現化だったのではないかと思います。その意味でも、ぜひ記憶に残ってほしいプロジェクトです。

「皆が使えるものを作るために『悪役』を買って出た」

Interview 2

「いいものを作る」「使えるものを作る」という信念のもとに、プロジェクトを牽引する役割を担ったのが近藤弘志氏だ。異業種、マルチベンダーという今回のプロジェクトは、ともすれば各社の意見がくいちがい、十分な成果を出せない危険性もあった。そのような状況で、徹底的な討論を展開し、参加者の方向をまとめる役割を担った近藤氏の存在は大きなものであった。

株式会社トヨタIT開発センター
専務取締役
近藤弘志氏

SECの立ち上げ構想からCOSEへ

── COSEに関係することになった経緯をお聞かせください。

私はSECを作る時から、関係者として参加しています。COSEのプロジェクトはその時の議論から生まれたといっていいでしょう。SECは業務系システム、組込み系システム、実証的な先進ソフトウエアプロジェクトの3本柱で実施することになりました。

その中で「では、実証的な先進プロジェクトは何をやるのか？」という話になると、これがなかなかないわけです。私は当時、（社）日本自動車工業会のITS部会長を務めていまして、ITSの全体構想を練る時から「ITSシステムを実現していくためには、国のプラットフォームとしてやらなければいけない」と考えていました。

その時からのITSのプラットフォーム構築という大きな目標に向けて、JARI、インターネットITS推進協議会などの関係機関やカーメーカーを含めた関連各社と意見調整し、事例としてなじみやすい交通情報をまずやろうと、経済産業省、自動車メーカーなどの合意となった。交通情報を扱うに当たっては、ITS各システム間の互換性を確保し、機能拡張性にも優れた次世代ITSプラットフォームが望ましい。こういったことから、交通情報のソフトウェアプラットフォームを作るプロジェクトCOSEプロジェクトがスタートしたわけです。

「V字モデルで2サイクル」にこだわる

── COSEの基本コンセプトと組織構造が決まっていった背景は？

COSEにおける開発の基本は、交通情報を、座標xyzとタイムスタンプを組み合わせた時空間データをうまく使う構造でやろうということです。本質は時空間データですが、わかりやすい「交通情報」というアプリケーションを表に出しました。そして経済産業省の製造産業局自動車課に協力していただき、自動車課がJARIに委託して要件定義を作成。この要件定義に基づいて、商務情報政策局情報処理振興課がソフトウェアプラットフォームを開発。このように、COSEの仕事はソフトウェア開発プロセスで大切な要件定義書の作成からプラットフォームソフトウエア開発へと進み出したわけです。

一方、COSEの会員企業がバラバラに入ってくるとうまくまとまらないため、どういう開発体制にすべきか、いろいろと議論がありました。しかし私がトヨタ自動車にいた時、改革プロジェクトの担当で得た、ある経験がありました。さまざまなベンダーやユーザーが混ざっている場合、プロジェクト全体をうまく進めるにはニーズの明確化、それを具体的な要件定義に落とし込み、具体的な開発アイテムを規定するというソフトウェア開発プロセス、つまりV字モデルの開発プロセスが一番良いと。そこで「COSEではV字モデルで行きましょう」と提案し、基本方針として採用されました。

もう1つ、これもトヨタ自動車で何度も味わった経験から得た教訓なのですが、V字モデルをしっかり回すためには、最低でも2サイクル回さないと駄目。最初の1年で、要件定義に従っ

てとにかくシステムを作る。実際、最初の要件定義は暫定仕様であり、わからないことがたくさんある。それを私は「エイヤで決めろ」といったわけです。要するに「知ったかぶり」をして一遍、流してみる。ここでは皆さんにものすごく協力していただきました。それからもう1つ。各ステージで、できたものをチェックする方法を決めておく。この2つが大事で、それでV字モデルを徹底的にやったのです。

その結果、どこに欠陥があるのかが明確に出てきますから、V字モデルの原点である要件定義をしっかり固めていくことができる。そして、2年目の2サイクル目でブラッシュアップするのです。

ソフトウェアの実装というのは、要件定義がきちんとしていれば、後はうまくやるか、下手かのどちらか。そうすればツールで解析して、同じコードが沢山ある、などのチェックができる。私は、ソフトウエア開発においては効率的開発も重要ですが、「自分たちは何を作らなければいけないのか」「その性能は何なのか」「期待するアウトプットは何なのか」などをしっかり共有化していないと仕事がいい加減になってしまうと考えていましたので、そのためにも、まず1サイクル回したのです。

── 要件定義を詰めていく上で、何が大事だとお考えでしょうか？

やはり、PMの役割が大きいと思います。PMは、仕組みを知らなくてもできるという議論がある。しかし私はソフトウェア開発では、全体の中の概念を理解する必要があり、チェックするだけのやり方では駄目、と信念で思っていた。そこで部下には「ゼネラルフローを書きなさい」といっています。データを使って、どういう処理をして、どのような答えを出すか。まず、処理フローを書く。それができたら、それぞれの項目をディテールに分解する。私はディテールフロー＝要件定義書だと思っている。その流れをしっかり見守り、是正していくのが、PMの役割だと考えています。

プロジェクトで結果を出すためには「悪者」が必要

── COSEの組織運営で、近藤さんが心がけていたことは？

COSEには、理事会の下に事務関連を担当する運営委員会とプロジェクトを推進する技術委員会が置かれています。ただ、当初の経済産業省の意向では、技術委員会のもとに、SE部会だけを置くことになっていた。そこにPMを任命して、一意専心開発するという構想です。

一番大切なのは、開発と、それによって何ができるか、をセットでやること。そうでないと

このプロジェクトは、世間からは何も評価されない。そこで、先にもいいましたが、V字モデルに基づいて、ユーザとソフトベンダーが同じ立場で議論できる場として、技術委員会はあるべきだ、と主張したわけです。ただ、そうすると今度は、7社の勝手な思いが強く出る部分もあるので、もう1つ、客観的な意見を聞く場として外にタスクフォースを置いてもらうよう、経済産業省にお願いしました。そこでは自動車メーカー、カーナビメーカー、電機メーカーなどがメンバとなり、今回のプロジェクトの成果を使う、使わない、を議論します。

というように、技術委員会の下に、SE部会とユーザ代表の評価ワーキングをセットで置き、外部の客観的意見を取り込む体制にしてもらった。一般に、ベンダーはユーザの下請的関係になっていることも多いのです。それをお互い同等の立場で、協力していいものを作ろう、という仕組みになったのがCOSEのいい面だと思っています。

もちろん、最初からうまく行ったわけではありません。当初、ほとんど知らない方が集まってきたCOSEでの議論を「皆さんのお手並み拝見」と、しばらく聞いていました。「これではいかん」と思ったのは、各社の利害が出始めたことです。最初は皆、口ごもっている。「こんなの、うちのノウハウだからしゃべれん」というわけです。異業種、異質の利害関係がある企業が集まって、特定のプロジェクトをやる時の難しさというのは、嫌というほど感じました。しかし、これができると誰かのために役立つとか、自分の企業にこういうフィードバックがある、など最後に目指すビジョンがものすごく明確になると、まとまりを生むのも実感しました。

しかし、期待したほどはうまく行かなかった部分もある。というのは、評価ワーキングのメンバが、同時にSE部会のメンバであったりするわけです。技術委員会であまり意見をいいすぎると、自分に火の粉がかかることもある。そこで、私が滅茶苦茶に怒り、意見をいう。そういう人間は、私しかいないから。それを経済産業省も認めてくれました。

参加企業の中には、経済産業省からお金が出るので、適当にやっておいて、経済産業省がOKといえばそれでいい、と思っていたところもあると思う。私もSE部会に対しては「組織の中身は何でもいい、あなた達に任せる」といって口をはさまなかった。

しかし最大の目標は、皆が使えるようなものを作ることです。だから私は、たとえ経済産業省から納品OKが出ても、納得しなければノーというと表明した。SE部会の運営にタッチしない代わりにチェック法は決める、それに対する答えがちゃんと出ない限り先には進ませない、という仕組みを作ったわけです。

COSEでの私の役割は、牽引役ではなく、アジテーターだと考えていた。なぜなら、プロジェクトには怒る人がいないと駄目だから。私は体が大きいこともあって、ぶっ続けで仕事をして、

午前2時、3時、徹夜でも平気。付き合わされた組合員企業の担当者やIPAの方などは大変だったと思いますが、最後、仕上げるための修羅場になった場合、ヨイショする人間ばかりでは、決していいものはできません。私は悪役を買って出たのです。

モチベーションを高めるソフトウェアエンジニアリングに期待

── ソフトウェアエンジニアリング関連の成果については、どう評価していますか？

私は、成果があったかどうかまで評価できる立場ではなかった。現在のソフトウェアエンジニアリングは、今までノー管理だったものを少しでも「見える化」して、トップからボトムまで見てどこにネックがあるのかを明らかにしようとするもの。だからそれはいい。

ただ一方、あまりにも管理手法に走りすぎで、少々物足りないものを感じているのも事実。ソフトウェアを組むという作業は、基本的に泥臭いと思っています。それほど格好いいものではない。困っているところに、「こういう知恵を入れた」「創意工夫した」という実装した人を評価ができるといい。知恵を加えた人を評価できる手法ができれば、日本だけでなく海外でも「自分の仕事を、こう評価してくれているんだ」と実感できる。モチベーションを高めるフィードバックをどうするか。SECの次のステップになるかもしれないですね。

── 分析・評価手法のほかにSECに期待していることはありますか。

標準化を目指す、今の活動はいいと思います。要望したいのは育成。大学に任せる教育と、実学の実践があるが、SECには社会人教育の機能、教育の場を持ってほしい。そうしないと、各企業の技術者は唯我独尊になり、組織は必ず垂直統合型になる。水平分業型の仕組みにならないから、いろいろな知恵が入らない。その実態を、企業のトップに見せてほしい。そうすると、自分たちのやっていることのまずい点がわかり、金食い虫になっていることに気づく場合もあると思う。

このままでは、日本のソフトウェア開発は、空洞化してしまう。なぜインドや中国のソフトウェアが伸びているのか、それは、どのような領域でどういう技術があるのか、真剣に探って、人材育成をしていく必要があります。

「勉強」となると企業もなかなか人を出さないので、共同プロジェクトはいい機会になる。小さいものでもいいと思う。COSEには、すばらしい人材が集まりました。やはり、国のプロジェクトでないと、なかなかここまでの人たちはそろわない。当然、SECにもそうしたプロジェクトの支援を期待したいと思います。

Interview 3

EASEプロジェクトの成果をCOSEプロジェクトで実証

EASE（Empirical Approach to Software Engineering）プロジェクトとは、「e-Society基盤ソフトウェアの総合開発：データ収集に基づくソフトウェア開発支援システム」として文部科学省が進めるリーディングプロジェクトである。2003年にプロジェクトが発足して以来、同プロジェクトを牽引してきたメンバの1人である、奈良先端大学の松本健一教授に、EASEが発足した背景からCOSEプロジェクトとの関わりについて、お話をうかがった。

奈良先端科学技術大学院大学
情報科学研究科
ソフトウェア工学講座教授
松本健一氏

理論を企業で実証する必要から

── EASEプロジェクトの目的について教えてください。

現在、大手SIベンダーの中には、ソフトウェア開発にかかった工数などのデータを定量的に収集し、分析を行うという仕組みを自社で構築している企業もあります。しかし圧倒的に、このような企業は少数派です。またデータを収集している企業はあっても、そのデータをなか

なかオープンにはしてくれません。私たちが考えた理論や技術の正しさを証明することがなかなかできなかったのです。従来の大学の研究室では、「便利なツールができました。でもその効果はわかりません」ということが多かった。その理由は、大学の成果を実際のデータで試すことができなかったからです。そこで、EASEプロジェクトを立ち上げ、自動的にデータを収集する仕組みを産学連携で作ることになりました。というのも、EASEは文部科学省主体のプロジェクトとしては珍しく、成果として「モノをつくること」と「産学連携すること」が求められていたからです。EASEの主要メンバは奈良先端大学と大阪大学のほか、NTTソフトウェア、日立製作所、日立公共システムエンジニアリング、SRA先端技術研究所という企業群で構成されています。

産学連携組織であるため、EASEには2つの視点の異なる役割があります。研究者の立場からすると、自分たちが考えた技術や理論の正しさを証明する根拠データを得るための場。一方、開発現場の人の立場からすると、開発している状況が把握でき、予定通りのコスト、納期、品質を実現できるよう、手助けしてくれるものを提供してくれる場。EASEにはこのように立場の違いによって、プロジェクトの捉え方が異なるという難しさがありました。

マルチベンダーで開発中のプロジェクトにはじめて適用

── EASEではどのような成果を上げているのでしょう。

モノづくりの面で、EASEが最初に上げた成果は、データ収集ツール「EPM」を開発し、2005年6月には「EPM 0.92 β日本語版」をオープンソースとして公開しました。COSEプロジェクトには、このバージョンを適用しました。

EPMは研究用のプロダクトだったため、COSEのような大規模なプロジェクトに適用するには難しい面もありました。しかし、COSEの開発現場の人たちからの声を聞くことで、大規模なプロジェクトでも十分、対応できるよう品質の向上、改良に努めました。2007年はSECにおいて、より多くの企業での検証を実施し、2008年には、いろいろな企業に容易に即、使ってもらえるようにパッケージ化したものの提供を開始します。

EPMのほかにもEASEでは、相関分析ツール「NEEDLE」、ワンクリック見積もり&データ品質診断ツール「Magi」などのツールを開発しています。

── EPMはCOSEに適用するには難しい面があったというのは、どういうことでしょう。

産学連携を基本とするEASEでは、COSEのプロジェクト以外にも、30社にも及ぶ企業との連携を行ってきています。しかしこれまでの産学連携のプロジェクトは、研究ベースの小さなものか、もしくは社内向けの小さなシステムが主だったのです。しかも1対1の契約がほとんどです。一方、COSEはエンタープライズ・ソフトウェアとしては大規模とはいえないまでも、これまでEASEが携わってきたプロジェクトに比較すると、規模が大きい。しかもマルチベンダーという、今日のソフトウェア開発の典型の1つともいえるものでした。

さらに分析のタイミングも異なりました。これまではすでに終了したプロジェクトに対して、分析するのが一般的だったのです。したがって、開発現場の人たちと実際に接触することはありません。しかしCOSEでは複数の会社と同時に連携し、しかも現在進行形のプロジェクトのソフトウェアを分析・評価します。もちろん、開発現場の人たちとも接触する機会がある。現実性という意味でも、これまでのプロジェクトとはまったく違っていたんです。

実務経験のある特任助手がデータ分析を担当

──開発現場の人たちとの会合は、どこで行われたのですか。

現場の人たちとの会合は、東京都港区・田町駅前のキャンパス・イノベーション・センター内にある「奈良先端科学技術大学院大学・東京事務所（リエゾンオフィス）」で行いました。また、具体的な分析作業は、大阪府豊中市・千里中央にある「EASE エンピリカルソフトウェア工学ラボ（千里ラボ）」内に設置した機密室で行いました。というのもCOSEのようにマルチベンダーのプロジェクトでは、どこか1社の会議室に集まって開発状況について議論する、また、その議論のもととなるデータ分析を行う、といったことが、セキュリティの面からも難しかったからです。

── COSEのデータ分析は、どのような体制で行われたのですか。

これまで述べてきたように、COSEプロジェクトはマルチベンダー開発であり、現在進行形でデータを分析・評価します。大量のデータを素早く分析し、現場の開発者にフィードバックするためには、データ分析の理論や技術に詳しいことはもちろんのこと、ソフトウェア開発現場のこともよくわかっている人間が必要となります。

そこでEASEでは、4名の特任助手を中心にデータ分析の体制を作りました。彼らは、博士号を取得しており、データ分析の理論や技術に詳しいだけでなく、企業での実務経験が豊富

な場合も多く、こうした産学連携においては欠かすことのできない存在です。COSEにおいても、厳しいスケジュールであったにも関わらず、ベンダーごとにきめ細かい分析を行い、その結果を開発者に毎週フィードバックしてくれました。そして、分析結果という具体的な証拠を手に、現場の開発者と対等に渡り合うことで、データ分析の重要性をCOSEの開発者の多くに理解してもらうことができました。本当にタフな仕事だったと思います。

彼らのような優秀な人材がいたからこそ、COSEプロジェクトにおけるデータ分析は成功したと思います。彼らの実力を評価する大学や企業は多く、すでに企業に就職した者、大学での採用が内定している者もいます。日本のソフトウェア工学研究の担い手として、COSEプロジェクトで得た数多くの知見を、今後の研究活動に生かしていってくれることと思います。

より多くのデータからソフトウェア開発の普遍的な条件を見つけていく

── EASEの目標を教えてください。

EASEでは、メガソフトウェア工学へ発展することを目指しています。これまで、ソフトウェアの品質を上げるために、大企業では独自の取り組みを行ってきているとはいえ、まだまだ個人的な取り組みがほとんど。また企業として取り組んでいる企業においても、データ収集以外のデータ分析や分析結果を基にした開発支援までは手が届いていません。つまりメガソフトウェア工学への発展とは、データ収集から開発支援までを業界全体に広げていき、また各プロジェクトから抽出した知見を、組織的な知識として再利用していけるようにすることです。

── メガソフトウェア工学へ発展するために、解決しなければならない課題について教えてください。

業界共通の知識として共通化を図るためには、まだまだ障壁があります。第1は「スケジュールがタイトなので、進捗管理を調べたい」「コストが超過気味なので、コスト管理を分析したい」というように、データ分析ニーズの個別性が高いこと。第2は、手法を適用して成果を得ても、それは「そのプロジェクトのみに該当したこと」「当社には当てはまらない」と、普遍性を問われることです。

たとえば医学の世界では、ある患者の症例報告も1つの成果として評価されます。一方、ソフトウェアの世界では、ある個別のプロジェクトにおける成果を報告しても、普遍性の議論になってしまいます。これらの障壁を乗り越えるためには、データを収集し、それらを分析・評価してプロジェクトがうまくいったという事例をより多く、集めていくしか方法がない。それ

らの成功事例の中から、普遍的な条件を見つけ出していくしか方法がないと思っています。そのためにも自動データ収集ツールであるEPMの普及は、欠かせないのです。

確かにEPMを導入したからといって、プロジェクト管理がすぐできるようになるわけではありません。しかし今では開発データのほとんどが、電子化されています。そのデータを生かし、今後のプロジェクトの品質向上に役立てていくことが重要なんです。このような意識が根付くよう、開発現場の人たちに直接、働きかけていきたい。EASEプロジェクト自体はあと1年を残すのみとなりましたが、データ分析の新しい手法が開発され、ツールの整備も進んでいます。何よりも、ソフトウェア開発データを分析し開発現場に役立てることのできる優秀な人材が数多く育ちつつあります。EASEのような取り組みは今後も頑張って続けていくつもりです。

ソフトウェアエンジニアリングの適用と実用性のあるソフトウェア開発を共有したプロジェクト

Interview 4

IPA SECの設立とともにSEC所長補佐に就任、COSEプロジェクトの立ち上げに関わった松浦清氏。同氏にCOSEプロジェクト立ち上げの経緯やCOSEプロジェクトを運営していく上で苦労した点、評価している点などについてお話しいただいた。

元独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター所長補佐
松浦清氏
（現メンター・グラフィックス・ジャパン株式会社
コンサルティング部ディレクター）

開発中のデータ取得、分析、フィードバックが重要

── COSEに関わることになった経緯について教えてください。

COSEに関わる大もとのきっかけは、SECの立ち上げに携わったことです。当時、外資系ソフト会社でコンサルティング部門を立ち上げ終わった私は、次の転職先を探していました。IPAに面接に行った際、たまたまSECを立ち上げる話を聞いたのです。SECの設立に関わる

ことになると、私の米国留学時代の恩師であり、現在はドイツのソフトウェアエンジニアリング研究機関であるIESE（Institute for Experimental Software Engineering）の所長ディータ・ロンバッハ氏に会えるという期待もありました。SECの活動には、3つの柱があります。第1にエンタープライズ系ソフトウェアの開発力強化、第2に組込みソフトウェアの開発力強化、そして第3がそれらの成果を実践・検証するための先進ソフトウェア開発プロジェクトを産学官の枠組みを超えて展開することです。

私はこの3本柱のうち先進プロジェクトの展開を担当。何を開発するかで曲折がありましたが、プローブ情報プラットフォームソフトウェアの開発に決定しました。そして、それを実際に開発する組織として、ソフトウェアエンジニアリング技術研究組合（COSE）を設立したのです。

── COSEプロジェクトを実施して評価している点は？

今回のプロジェクトでは、SECの目的の1つである産学官の橋渡しもできたと思っています。ソフトウェアエンジニアリングの観点からの成果もあります。従来、開発における生産性のデータを取得して分析することは何度も行われてきました。しかしいずれも、ポストモーテムなのです。ポストモーテムとは、検死とか司法解剖という意味の言葉を、プロジェクトの分野ではプロジェクトが終わってからの検証という意味で使います。それではダメなんです。開発中にデータを取得、分析して、開発チームにフィードバックする。そのサイクルを何度も回すことで、品質を高めていくことができる。COSEではそういう取り組みを実践できたことが、よかったと思います。

実際の開発現場では、ソフト開発優先で、ともすればソフトウェアエンジニアリング的な側面を忘れがちです。忘れさせないためにも、イベントを作るなど何らかの仕掛けを作り、気を引くことが必要です。今回の分析手法の中には、コードクローンや協調フィルタリングなど、新しい切り口がありました。このような新しい分析の切り口を提案できたことも、現場の意識を引き付ける1つの要因となり、よかったと思います。

分析・評価、フィードバックを週に1度のサイクルで回す

── 苦労した点について教えてください。

データを収集し、分析・評価を行い、フィードバックするというサイクルを回すのに最も苦労しました。というのも、データがスケジュール通りに収集できない、また収集してもフォー

マットがそろっていないなど、いろいろ大変なこともありました。分析し、フィードバックするEASEのメンバも、自分の研究もありいつでも手が空いているわけではありません。また、COSEを構成している各社は、機密保持の点から開発中のデータをお互い見せ合うことができませんでした。そのため、各社個別にフィードバックすることになり、時間と場所の調整に苦労しました。各社が互いに情報を出せる仕組みを考えていくことは、今後の課題でしょう。

もう1つは、COSEが求める成果が2つあったこと。COSEプロジェクトは、ソフトウェアエンジニアリングを適用するだけではなく、高品質のソフトが開発できなければ成功とはいえません。ソフトウェアエンジニアリング的に挑戦したいことがあっても、ソフトが完成できなければ失敗となります。このような課題があったため、COSEとしては開発を進めることに専念してしまい、ソフトウェアエンジニアリングに目が向かないこともありました。モノづくりとソフトウェアエンジニアリングのバランスを取ることの難しさを実感しました。これら、各社がCOSEで苦労したことは、今後、自社でソフトウェアエンジニアリングを適用する際の参考になると思います。

── エンピリカルアプローチを採用した理由について教えてください。

今回は、プロジェクト開始と同時に1から手法を探すのではなく、ある程度できた手法を使うことにしたのです。EASEプロジェクトは、COSEの約1年前にスタートした文部科学省のプロジェクトです。計測、定量化と評価、フィードバックによる改善という実証的手法（エンピリカルアプローチ）の実践を目指しており、COSEが立ち上がった時点である程度、適用できる手法があったのです。しかも、EASEプロジェクトは産業界の協力がなかなか得られず、実証する場を探していたのです。COSEの「すぐに適用できる手法が必要」というニーズと、EASEの「実証する場がほしい」というニーズが一致したというわけです。

ただ、EASEでの手法をそのまま適用することはしませんでした。というのもEASEでこれまで行ってきた手法は、ポストモーテム的だったからです。先ほど説明したとおり、データ収集から分析・評価、フィードバックを1週間に1度という早いサイクルで回していくこととしました。

ソフトウェアエンジニアリング手法の普及をSECに期待

── プローブ情報システムを開発することに決まった理由を教えてください。

JARIでは、プローブ情報システムを研究しているグループがありました。しかしまだ、実

際のソフトウェアにするまでには至っていなかったのです。そこでJARIの成果をたたき台に、さらに詳細な要件定義を行い、開発することになりました。このように今回のCOSEプロジェクトは、EASEプロジェクトの奈良先端大学と大阪大学やJARI、経済産業省、SEC、自動車メーカー、部品メーカー、SI企業などさまざまな組織、人たちが協力し、自分たちの持てる力を出し合えたからこそ、実現したと思います。

── COSEのような産学官のプロジェクトがうまくいった要因は何だと思いますか。

COSEの目的はソフトウェアエンジニアリングのベストプラクティスを適用することと、実際に動くソフトを作ることの2本柱でした。このことを常に皆で意識し共有してきたからこそ、成功できたのだと確信しています。これらの目標を達成し、COSEを設立した役割はうまくまっとうできた気がします。

── あとは出来上がったソフトウェアをどうビジネスに結びつけていくかですね。

実際、作ったソフトウェアをビジネスにしていくためには、JARIとSI企業、自動車メーカー、部品メーカーが話し合って決めていけばいいと思います。

適用したソフトウェアエンジニアリングとツールについては、CD-ROMにまとめて配布します。今回、CD-ROMにまとめるに当たり、マニュアルやインストーラーなど、誰もが使いやすいように整備しました。また操作画面などの見た目が悪かったところも、改善。つまり、EPMツールを研究者の手から一般の方が使えるツールとしたのです。業界共通のソフトウェア開発における品質管理の手法としてEPMツールを広く普及、定着させていきたい。そうすることで、ソフトウェア開発全体の生産性向上が実現するはずです。

ソフトウェアエンジニアリングの普及を促進するためにも、SECにはぜひ、ソフトウェアエンジニアリングに関するあらゆる情報を提供できる役割を担ってほしいと思います。

Interview 5

鉱工業技術研究組合法に基づいた組織COSEのバックオフィスを支える

先進ソフトウェア開発プロジェクトは、開発の主体となる7社がCOSEを組織して進められた。これは、昭和36年に「鉱工業の生産技術の向上を図るため、これに関する試験研究を協同して行うため」に施行された「鉱工業技術研究組合法」を根拠とする組織で、経済産業省の所管となる。ここで、総務部長を務めた小出氏には、多業種の7社が協同して作業を進めなければならない環境において、各社の文化の違いともいうべき微妙な差異を調整する苦労もあったという。

ソフトウェアエンジニアリング
技術研究組合総務部部長
小出満氏

富士通で長年経理畑を歩み、COSEの総務部長へ

――小出さんのご経歴と、COSEの総務部長就任の経緯をお教えください。

約40年、富士通で経理関係の業務を担当してきました。最近はさまざまな富士通グループ会社の資金繰りのサポートなどを行っていて、そうした関連の経験は豊富です。COSEの登記は2005年2月21日ですが、創立総会が1月18日に開催されており、その翌日、19日から組

合の事務所設立の作業を行いました。

── COSEにおける総務部の業務についてお聞かせください。

総務部はCOSEの「裏方」として、運営全般を支える役割を担いました。正規職員は私1人で、派遣の女性2人が常時在籍。それぞれ総務全般、経理を担当してサポートしてもらう体制でした。

COSEには、意志決定の議決機関としての総会、業務執行機関として理事会があります。その理事会をサポートするため、組合員企業の技術者が集まる技術委員会と、事務方が集まる運営委員会が置かれていました。両委員会の役目は会員企業7社のベクトルを1つの方向に向けることで、委員は基本的に各社別の方が就任していましたが、諸事情により兼任されていた方もいます。総務部は主に運営委員会を支える業務を行っていました。

総務部の仕事のなかで主なものは2種類で、その第1は決算処理。COSEは法人ですから、法人税の申告から納税までの業務がすべてあります。当初私は、経済産業省の所管する営利を目的としない組織なので非課税なのかと思っていましたが、税務署からは「法的には一般の中小企業と同じ。税金はしっかりいただきます」といわれました。

COSEの運営費、総務部の人件費やIPAに借りている事務所経費などは、会員企業からの会費でまかなっています。運営費で資金ショートしては大変ですから、少しだけ余裕が必要になります。すると、どうしても「利益」が出る。そこが課税の対象になってしまうわけです。

もう1つが経済産業省との調整や契約書関係の業務で、比率でいえば業務全体の8～9割になります。ソフトウェアの開発や実証実験は7社の組合員企業がすべて行っていますので、そのコストを全部集めて経産省でチェックしてもらう。それを確定検査というのですが、クリアして初めて実積として認定され、予算が交付されるわけですから、組合の運営上、非常に重要な業務となっていました。

組合運営に関する情報不足と7社連合ならではの苦労

── COSEの総務を担当して、大変だったこと、苦心したことはありましたか？

富士通のグループ会社で、さまざまな経験を積んできましたが、技術研究組合は初めてです。それまでの仕事では、必ず目上の人がいてアドバイスを受けることができましたが、今回は私がトップです。さらに、初めての組合運営なので業務の進行方法などでわからないことが数多くあり、その都度経済産業省のご担当者からアドバイスを受けました。

一番悩んだのは、経済産業省に提出する書類を作成するために、組合員企業7社がどういう書類を作成しなければならないかなどを記した業務マニュアルの作成です。実際に業務を開始してみると、あの書類がない、この書類が作られていない、などの問題が次々に発生しました。

結局、経済産業省の研究開発プロジェクトの推進を行っているNEDOの業務マニュアルを参考にしました。分厚いマニュアルの中から、必要な項目だけ抜き出したわけです。それでも最初の1年は、初めての仕事ということでの試行錯誤や、業務マニュアル作成で終わってしまいました。

また、経済産業省に提出する書類のまとめ方がある程度わかってきた後でも、別の大変さは続きました。COSEは7社による組合ですが、経済産業省の予算枠は決まっています。その予算を組合員企業に分配するためには、各企業がどのように仕事をしたかを示す確証が必要になります。それを収集することが総務部の重要な仕事の1つです。

その確証の中で、研究者の従事日誌を集める必要があります。私どもが作成した「業務マニュアル」に基づいて作成していただくのですが、COSEの組合員企業はそれぞれ就業規則が異なります。昼休みの開始、終了時間も違うし、17時頃に休憩があるところもあれば、すぐに残業がつく企業もある。また、7社それぞれメインで行っている仕事が違いますし、パートナー企業も使っています。さまざまな組織内で常時約50名が動いている。その日誌を集めてまとめるのですから、簡単ではありません。

さらに、提出された日誌には誤りもある。たとえば、毎回同じ曜日に開かれていた委員会の開催日が、何らかの都合で変更になった場合に、勘違いでいつもの曜日で計算して出席したと記してしまったケースなどです。その場合、総務で確認して、修正させる必要があります。さらに50名以上となれば、なかなか書類の提出期限を守れない人も出てきます。その催促などもしなければなりませんでした。

コミュニケーションが難しかった事例もあります。各社がメインで行っている仕事が違うために、こちらから同じ言葉で要請しても、受け取り方が変わってしまうのです。私は富士通のグループ会社で約40年経理畑にいましたが、7社の担当者に経理出身の方は1人もいません。そのため、うっかり経理用語を使ってしまうと特に伝わりませんでした。

さらに、それぞれの社内で使われている用語も異なっているため、生じた問題もありました。富士通のグループ会社では6分を0.1Hとして、時間を10進法で計算しています。それが当然だと思って、7社に一斉メールで流したら、富士通グループ以外の方にはわからない。「なぜわ

からない。常識だろう」と思ったら、実は富士通グループでしか通用しない「常識」だったのです（笑）。独自の用語というのがあるのは当然なのですが、最初はそういうことにも気づきませんでした。

組合運営の経験を伝えたい

── COSEの解散処理と総務部長として運営に関わられたご感想をお聞かせください。

COSEは、5月の通常総会で2006年度の事業・決算報告をして、31日の解散を決定。その後、財産をゼロにし、総会開催、報告書提出、確定申告納付などの後処理が10月ごろまでかかります。

会社を作る経験をした人は少ないと思いますが、会社を完全になくすところまでの手続きを行った人はもっと少ないと思います。ましてや技術研究組合となれば、なおさらです。そこで調べていきましたら、経済産業省で普段通っている部署がある本館ではなく、別館の方に組合の運営に関する情報を持っている部署があることがわかったのです。

最初の1年は、経済産業省のどの部署の誰に、何を質問すればいいのかがわからず、なかなか必要な情報が集まりませんでした。それがだんだん経験を積んで、省内のどこでも、初めての人にも「ちょっと教えて」と質問できるようになりました。

また、富士通グループの中にも似たような組織の運営に携わった経験のある方がいることも後になってわかり、アドバイスを受けることができました。ただ、すでにリタイヤした方も多く、そのために充分な情報を集められなかったのも事実です。

IPA内にあったCOSEオフィス

私がCOSEの総務部で得た経験、知識などは、次に同様の仕事をする人がいるのであれば、何らかの形で伝えられれば、と考えています。

3-2 プローブ情報プラットフォームソフトウェアの開発

Interview 1

良質なデータ作りをめざして変動リンク方式に挑む

今回のプロジェクトで交通情報の作成に当たって、「固定リンク方式」と「変動リンク方式」の2種類が採用された。プローブ情報の測定区間を交通状況によって変化させるという「変動リンク方式」は、新しい概念であり、ソフトウェア開発に当たってまったく新しいアルゴリズムを適用しなければならなかった。変動リンク方式に取り組んだ松下電器産業の山下氏にお話をうかがった。

松下電器産業株式会社
ITS事業推進センター
プロジェクトリーダー
山下哲郎氏

交通情報作成のために「変動リンク方式」を開発

── COSEにおける山下さんの役割と、松下電器産業の担当業務をご紹介ください。

私は松下電器産業が担当した業務関連の開発関係、技術開発の責任者です。COSEでは2007年2月に公開デモを行ったのですが、その企画も仰せつかりました。

実際の作業では最初に、バスとタクシーの走行情報データの収集を行いました。当社では

バスやタクシー関係のシステムを納入していますので、その関係先にご協力いただきました。

2点目は、集めた情報をもとにしたさまざまな処理です。バスとタクシーにはそれぞれ、一般の道路走行とは違う特殊な動きがありますので、その部分の情報を取り除くクレンジングと呼ぶ作業を行います。そして、処理済みデータを「共通車両情報データベース」に落とし込む。ここのデータ処理では、自分たちが集めたものだけでなく、富士通と日立製作所が収集したものについても担当しました。

── 交通情報の作成には「固定リンク方式」と「変動リンク方式」の2種類が採用されたとうかがいました。

固定リンク方式は主に日立製作所、富士通、NECなどの担当で、我々は変動リンク方式をやりました。変動リンクというのは、新しい概念です。プローブ情報のデータというのは、交通量が多い時には豊富に取得できます。交通情報を求める場合、通常はA地点からB地点までの区間の通過時間を計り、一定以上かかっていれば渋滞、そうでなければ順調と判断します。ところが、同じ計測区間でも一定期間内に通過した台数が少ない場合とない場合があり、台数によりデータ精度が不均一になります。

そこで編み出し、挑戦することにした方法が、変動リンク方式だったというわけです。通過台数に応じて計測の区間を広げたりすることによって、ある程度均一に交通情報を得ることをめざします。要求される処理は複雑なのですが、考え方としてはリーズナブルだと思います。

── 変動リンク方式を開発する際、大変だったことはありましたか？

やはり、どこにもない新しいアルゴリズムでしたので、最初から四苦八苦するところも多く、1つひとつ検証しながら開発しました。また、松下のプローブ情報のデータはわかっていますが、他社のものは不明でしたので、それをうまく融合させるための作業も苦心したところです。

変動リンクの効果ですが、2月の公開デモでは、交通量が少ない時の評価まではできませんでした。ただ、社内のテストで計測した結果、良好な値が出ています。台数が多い時には当然、正確な値を出力していますが、少ない時にも適切と思えるデータを得ることができました。

公開デモのコンセプトをどうするか、で議論を重ねる

── 2007年2月の公開デモについてうかがいます。準備はどのように進めたのでしょうか？

2006年4月よりWGを設立し、週1回のペースで検討してきました。9月までに、企画書にまとめ上げ、10月よりデモシステムの開発を行い、また、1月からデモ本番まで最終調整を行いました。

── 公開デモを準備するに当たって、苦心したことはありましたか？

一番大変だったのは、COSEとして何をアピールするかを決めることでした。各社がいろいろな技術を持ち寄って作ったものですから、それぞれに見せ場のようなものがあります。ただ、そのまま出してしまうと、まとまりがなくなってしまう。我々がCOSEとして行った成果で、何をアピールするか。どこに焦点を当てれば納得してもらえるのか、そこを見つけ出す作業が一番大変だった、といえるかもしれません。

2006年4月頃から公開デモ企画ワーキングを立ち上げ、週1回ずつ、8ヵ月ぐらいかけて議論に議論を重ねました。その間、議論の経過を技術委員会に報告してコメントをいただいたり、出た話を各社に持ち帰って検討したり、なども続きました。

公開デモである以上、見に来られる皆さんにとってわかりやすいものでなくてはいけません。それは当然、COSE関係者だけではない。交通問題やITSに詳しい方は問題ないと思いますが、そうではない人にどのように説明するか、が大きな課題でした。

中でも悩んだのは「言葉をどう選ぶか」です。たとえば、今回の公開デモでは「リアルタイムにポイントを置きたい」という話がありました。では、リアルタイムとは何か？　という議論になるわけです。その上で、リアルタイムで何をアピールしたいのか？　などと掘り下げていったわけです。

そして2月のデモ本番を終え、実施したアンケートでは非常に好意的な回答が多く、企画責任者としては、ほっとしています。

7社合同のCOSEで得られたものと今後

── 今回、7社の異なる文化、技術を持つ企業と共同作業をしてみて、得られたことはありますか。

技術者というのは、とかく井の中の蛙になりがちです。自分の技術が一番と思い込んでいるところがある。そのため一番良かったのは、他社のデータを使って、他社の処理したものと比較ができて、自分たちの技術の位置づけがどの辺かということが、客観的にある程度わかったことです。

また、今回の仕事のポイントは「どうやってデータを集めるか」だったのですが、成果を残すことができたのは皆さんの協力の賜（たまもの）です。皆で達成できた、という実積を残すことができたことも良かった点だと感じています。

今後、得られた成果を各社でどう生かすか、が課題になるわけですが、そこでもやはりデータを集めることが基本になります。それは各社ごとにやるべきなのか、皆が協力すべきなのかについては、じっくり考える必要があると思います。

今回、松下の開発メンバで反省会を開いたのですが、データを集めた後の1台ごとの処理で課題が残っていることが話題になりました。たとえば、タクシーにはいろいろな特徴があって、会社ごとの運用によって、動きが全然違う。タクシーだからといって、一概に処理はできないのです。なぜ違うのか、もっと要因を追求する必要があります。全国への展開を考えた時には、全国にどういうデータがあるのかなどを調査しなければならないでしょう。

多様なデータが存在するという事実からしても、データ集めは1社だけでは難しい。ですから、各社が協力する方向に向かえばいいと考えています。そうなれば、すごく円滑に市場化が進んでいくのではないでしょうか。

可能性を実感したソフトウェアエンジニアリングの実践

── COSEのもう1つの目的、ソフトウェアエンジニアリング的に検証しながら開発を進める、という面での成果はいかがでしょうか。

実は私は今回、ソフトウェアエンジニアリング関連には、あまり深くタッチしていないのです。ただ、一緒に研究開発を行った弊社の開発メンバが一生懸命取り組んでいるのですが、

IPAに解析していただいた結果が、非常にいい刺激になった、と語っています。開発の仕方やツール類が今のままでいいのか等を検討するきっかけにもなったというのです。

特にソフトウェアの現場は正直、国内だけでは厳しい状況になっています。海外まで含んだソフトウェアの構築が主流になっていく中で、EPMのようなリモートでのプログラム開発をサポートできる環境というのは非常に価値があり、可能性を感じるという意見が出ています。

―― 他のCOSE会員企業からは、自社ツールで行っていた解析をEPMでできるように変えるのが大変だったという声も聞かれます。

その種の問題は、あることはあったのですが、我々は最初からEPMを受け入れる、と決めていました。ですから、何も目立った抵抗はありませんでした。現場からも苦労したという声もなく、結果がちゃんと出てきましたので、評判もいい。EPMのようなロジカルな管理手法を実際のシステム開発で適用できたことは有益だったと評価しています。

今度は我々の大きなテーマ、カーナビや携帯電話など組込み系端末開発においても、役立てていきたいと希望しています。

新鮮だった共同プロジェクト。今後は実用化に向けた取り組みが「使命」

―― 最後に松下電器産業から見たCOSEプロジェクトの感想と、今後の展望についてのご意見をお聞かせください。

COSEの目的であるソフトウェアを作る、プラットフォームを構築する、ということでは、かなりのところまで追求できたのではないでしょうか。

また今回の経験は、プローブ情報とソフトウェアエンジニアリング関連の両方で得ることがあった、と両方で評価しています。

ただ、これが本当に実用化できて、カーナビゲーションや携帯電話などの端末に対して、どういう風にダイナミックな変化を与えるか、などは今後の課題として残されています。今回作ったソフトウェアやプラットフォームを本当に使っていただくためには、ドライバーなど利用者側がどういうメリットや楽しみを享受できるのかを示す必要があります。しかし、そのためにはまだまだ詰めていかなくてはならないことがあり、それはこれから、各社に任された世界だと考えています。

実証プロジェクトにおける成果だけで終えず、実用化に向けた開発を使命だと思って取り組んでいきたいと思っています。

Interview 2

COSEプロジェクトにより、プローブ情報システムを実用化へと導く可能性を実感

COSEが開発したプローブ情報システムの要件定義に携わっていた日立製作所の横田孝義氏。同氏は10年来、交通情報システムの開発に携わっている。交通情報の専門家として、COSEプロジェクトの成果をどう評価しているのか。またそこで得られた知見は何か。日立製作所の役割とともに、COSEに参加して得られたことなどについて、お話をうかがった。

株式会社日立製作所
日立研究所主管研究員
工学博士
横田孝義氏

世界的にも技術的に突き詰めた実験になった

── COSEプロジェクトにおける日立製作所の役割を教えてください。

当社が担当したのは収集系と処理系のプログラム開発です。収集系プログラムで当社が担当したのは2000～3000台の車両を保有するあるタクシー会社。そのタクシー会社が収集しているプローブ情報を共通車両データベースに格納するためのプログラムを開発しました。一

方の処理系プログラムとは、各事業者から集めたプローブ情報から渋滞情報や旅行時間などを生成するためのプログラムです。その中の固定リンク方式について担当しました。さらに携帯電話を想定した収集系プログラムの開発にも携わっています。当社は、プローブ情報プラットフォームソフトウェアの入口から出口までを担当したということです。私自身はプローブ情報システムの要件定義の部分に携わりました。

── COSEプロジェクトの中で最も大変だったことについて教えてください。

一番大変だったのは、各事業者が提供してくれるプローブ情報を統一フォーマットにしたことです。というのもそれぞれ、タイミングや特性が異なっているからです。またCOSEは7社のほか、官学が一緒に取り組んでいるプロジェクトです。いろいろな組織体にはそれぞれ固有の文化がある。その組織における文化の違いに慣れることも大変でしたね。

さらに今回、COSEに参加している7社はそれぞれ先進技術を持っており、しかも競合関係にある。それらの会社が一同に介して同じ目標に進んでいくわけです。どこまでどう話していいのかわからず、そのさじ加減が難しかったです。

── COSEでの成果をどのように捉えていますか。

実験としては世界有数の規模で、技術開発としてはかなり深いところまでできたと思っています。確かに、英国や米国など諸外国では、プローブ情報システムをすでに実用化している国もあります。しかしCOSEで行ったことのように、ここまで技術的に突き詰めたところまでできていないと思うのです。そういう意味では、当初の目的をほとんど達成できたと考えています。

もちろん、いくつかの課題は残っています。しかしそれらの課題でさえも、開発を進めていく中で明らかになったことも多い。課題が抽出できたことも、成果だと考えています。

営業車両の特徴抽出になお課題も

── どんな課題が明らかになったのでしょう。

プローブ情報とひと口にいってもさまざまです。COSEではバス、タクシー、物流車両など多様な車種をプローブカーとして扱いました。それぞれ車種ごとに走行挙動が異なり、これらをいかに共通的に扱うべきかで多くの議論を行いました。例えばタクシーの客待ちしている情報について、どう扱うかなどプローブ情報のクレンジング手法は奥の深い課題です。もう1つ、実証実験をしてわかったことがあります。それは旅行時間が予想以上にバラついていた

ことです。信号機の影響もあるのでしょうが、1～2kmの区間で旅行時間に3つの山があったのです。この結果を踏まえ、交通情報としてどう提供するか議論をしました。結局公開デモの際は、平均時間に幅を持たせた情報として提供しました。しかし実際には、平均時間ぴったりの人はそれほどいないはずです。旅行時間のバラつきに関する処理の仕方についても、今後、考えていく必要があるのかもしれません。解決しなければならない課題は多々ありますが、プローブ情報の実用可能性は、数が集まれば可能だと考えています。

── 日立製作所として、今後、このシステムをどう展開していく予定でしょうか。

どこまで今回、参画した他者と共存し、競争していくかは未定ですが、当社としては事業化を検討しています。COSEで得た知見により、どれだけの車が集まれば、どの程度の質や量の情報が出せるかがわかりました。ビジネスモデルをどうするかという問題はありますが、実用化の可能性は十分、あると考えています。

── ソフトウェアエンジニアリングが適用された成果についてはいかがですか。

直接、開発に携わっていないのですが、適用されたソフトウェアエンジニアリングについては、役に立った、参考になったという話は聞いています。またツールの導入に関しても、わりとスムーズに導入できたようです。

当社としてこの技法を導入するという話はまだありませんが、私が所属する研究所ではぜひ、COSEでのやり方を参考にし、従来のやり方を変えていきたいと考えています。研究所で開発されるソフトウェアは、どうしても個人の裁量に依存してしまいます。したがって、ドキュメントが整備されていなかったり、プログラミングに癖があったり、検証が不十分だったり、情報が共有できていなかったりなどの課題がありました。せっかくアイデアがよくても、ソフトウェアとしては完成度が低いために、社内に展開しづらいのです。それを解消するためにも、ソフトウェアエンジニアリングを導入して、改善したいと思います。

最後まで要件定義を行い課題として残ったことも整理

── COSEは産学官連携のプロジェクトですが、それがうまく機能したと思いますか。

私自身はモノづくりに携わっていないので、それほど実感としては感じていないのですが、第三者的な指摘があったことはよかったと思います。7社も民間企業が参画してプロジェクトを進めようとすると、どうしてもバラバラになりがちです。しかしそれを学や官の組織が串刺しにしてもらえるので、バラバラになることなく比較的、スムーズに進めることができたと思

います。特にSECなど公的な機関が介在したことは、企業間が協調するためのけん引力となりました。

── COSEでは要件定義をモノづくりと同時進行的に行うという珍しい取り組みだったと聞いていますが。

一般的なプロジェクトでは工期がCOSEほど長くないために、最初に要件定義を決めウォーターフォールで開発していくのが主流です。しかし、COSEは3年弱という開発期間が与えられていたため、2回、回すことができたという点でも、通常のプロジェクトとは大きく異なります。今回、開発したプローブ情報プラットフォームソフトウェアはもともと、JARIが作成した要件定義書をベースにしています。それを、実際に開発できるよう多少見直したものです。その要件定義書にさらに、初年度と最終年度に行った実証実験の結果から得られた知見をフィードバックし盛り込んだのです。フィードバックがしやすかったのは、要件定義とソフト開発のメンバが共通な会社から構成されていたためでしょう。そのため、最後の最後まで要件定義を作っていたというのが現状です。つまりCOSEの要件定義書は実装した要件にとどまらず、課題として残った部分も整理した形となっています。多少違和感もありますが、実用につなげていくためには、不可欠なことだと認識しています。

── COSEに参加した感想について。

これまでも官民共同プロジェクトに参加したことはありますが、毎週、検討会を実施するなど、COSEほど短期集中型で内容の濃いプロジェクトは初めてでした。特に参考になったのは、徹底的に議論する、課題管理をきっちりするなどというプロジェクトのやり方です。当社でも製造現場ではそれが実現できているとは思いますが、研究所だとどうしてもそういうことが甘くなってしまう。今後は、COSEでの経験を生かして、研究所におけるプロジェクトへの取り組み姿勢を変えていきたいと思います。

Interview 1

フラットな組織、ブラックボックス化されたノウハウなど、困難なベンダープロジェクトを率いた立場から

PMは、プロジェクト全体を統括し、その成否に責任を負う立場にある。そのため、通常のプロジェクトでは、PMは最高の権限を持ちプロジェクトを指揮することになる。しかし、COSEは組合組織であり、参加各社が並列の権限を持つ。PMはいわば調整的な役割を担わなければならない。その難しさやプロジェクトの評価について勝又氏に語っていただいた。

株式会社NTTデータ
公共ビジネス推進部技術戦略部
プロジェクト推進担当部長
勝又敏次氏

研究組合ならではの特徴とマネジメントの難しさ

── COSEプロジェクトにおける勝又さんの任務をお教えください。

プロジェクト全体のPMを務めました。マルチベンダーによるプロジェクトではどうしても各社の凸凹が出ます。それをいかに全体に影響が出ないようにするかが、PMのメインの仕事です。公の場で全体の調整を行った後、個別に要望を出し、解決を促すなどのマネジメントを

行いました。

── COSEは鉱工業技術研究組合法による研究組合で、一般的なプロジェクトとは違う面もあったと思います。PMとしてまとめていく上で、大変なことはありましたか？

難しさの種類は、どの角度から見るかによって、違ってきます。COSEは組合員ベンダーが並列のフラット組織でした。そのため、PMがプライムの立場ではなかったことによる、ある種の「面倒」がありました。

何よりもまず、各ベンダーそれぞれが持ち寄った技術、ノウハウがブラックボックス化されていたことがあります。本来PMは品質を高めるために、今回クローズされてしまっている詳細設計書、プログラム設計書、必要があればソースコードにまで入り込むのですが、それができなかった。そのため、開発結果についても測定したデータでしか見ていませんので、本当に満足できる品質のものができたかどうか確認していない、という思いがあります。

また今回、コストにはタッチできませんでした。普通のマネジメントで重要な部分となるコストを抜きにして、品質、納期のコントロールをしなければならなかったのが、やりにくかった点です。もちろん、よほどのことが無い限り、現場レベルでコストまで行き着くことはありません。スコープや考え方などに違いが出てきた時、追加仕様的に修正が入ったと時などに影響する場合が多いのですが、その場合は技術委員会で整理してもらって受け取るなど、上位レベルの意思決定機関を活用し対応するようにしました。

ただ1つ、最後の納入段階になって経済産業省から「成果を数字で示してほしい」といわれたのは、少し困りました。なぜなら、本来成果として一番効率を測ることができる部分（コスト）にタッチしていません。品質・納期に関連した部分を断片しかやっていない。そこが苦しいところでしたね。

── COSEは組合員7社、それぞれが分散して開発を行う体制でした。そのマネジメント上での難しさについてはいかがでしたか？

得意技を持つマルチベンダーの総合力で開発を行うというのは、当社本来のやり方です。実際、プロジェクトの規模が大きいほど、ある意味ではリスク分散につながる、という面もあるのです。

当社とトヨタ自動車を除いた5社による開発というのは、まあまあの陣容だと思います。ただ、プロジェクトの規模感から見ると多すぎるかもしれません。そこは今回、各ベンダーの持つノウハウでプラットフォームを作るという前提がありましたので、その意味では適材適所で入っていただいたと考えています。

── 複数のベンダーが参加して、しかも公的なプロジェクトということであれば、各社の足並みを揃えるのが大変なのではないかと想像しますが、その点はいかがでしたか？

COSEのスタートは2005年2月ですが、実は、私が参加したのは6月からです。立ち上げの大変な時期のマネジメントは前任者の役割になり、私は詳細設計以降、本当に実務レベルで動きだすところのコントロールから担当させていただきました。そういう意味では前任者がプロジェクト運営の基礎を作るのに苦労されたと聞いています。

── プロジェクトを途中から引き継ぐというのは、大変なのではないでしょうか。

そういう面も多少ありました。ある程度各社のカラーが出はじめた時点でしたから、最初はお互いに違和感もあったと思います。私も少し自分のやり方と違うと感じましたし、各ベンダーにも「せっかくやってきたものと違う」という部分があったのではないでしょうか。

とはいえ、私がマネジメントを引き継いだ際、前任者からは「淡々とやれば大丈夫」と言われていました。実際、大きな困難があったという印象はありません。

公開デモの実施により得られたものとは

── COSEプロジェクトは新たなプラットフォームの開発とソフトウェアエンジニアリングの2つのテーマがあったわけですが、成果についての評価をお聞かせください。

まず開発における成果ですが、何らかの形になるものを作ろうというテーマで、問題なく動くものを作ったわけですから、大体成功だったと思います。「大体」としたのは、実用化という部分で我々が商用ベースを考えると、まだまだやらなければいけないことがあるのが実情だからです。もちろん、そこまで踏み込むのはCOSEの目標ではありませんでした。そういう意味での品質確保などが今後の課題として残っています。

こうした実証実験では単に「やった」で終わってしまうことも多いのですが、公開デモによって全体像を示すことができたのは、確かに大きなことだと思います。そのために各社の自発性も求めましたし、コミットしてもらうために、ちょっと無理をいったこともあります。

── 公開デモについては、どう評価していますか？

具体的な形に表れたというのは、確かに感激です。実運用に近い状態での長時間運転に耐えうるものを作ったわけですから。中には、実際に動いているのを見せるのはリスクが高い、動いたところを記録しておいて、それを一般に流すぐらいがせいぜいではないか、と見ていた人もいたそうです。確かにシステム的には、そういう要素がありました。開発は単一サーバー

で、バックアップなしの環境でしかやっていませんでしたから。

難しかったのは、今回のシステムには正解値がないことです。銀行システムなどでは、決まった数値が出なければ、明らかにバグと指摘されます。ところが、いわゆる渋滞を表示する場合、感覚的なものと、設定した値にずれが生じる可能性がある。そのため、各社のプログラムに内在しているパラメータをぎりぎりまで触ってもらい、デモ直前まで実体に近い表示になるように調整しました。

デモ実施に対する評価ですが、世間一般に結果をさらしたというのは大きかったと考えています。経済産業省としても、あまりないタイプの研究開発だったのではないでしょうか。今後、プローブ情報は注目の分野です。課題は、周波数の割り当てで使えない、使ってもコストが高いなど、通信の問題だけです。そこさえクリアになれば、いろいろな情報が飛び交う、さまざまなサービスを提供できるようになると思います。

分析のデータ収集にかかる時間を１週間にまで短縮

―― ソフトウェアエンジニアリングに関する活動についての評価をお聞かせください。

今回試した手法・ツールの中でEASEで開発中のEPMを使った定量データ収集・分析サイクルについては、理解できましたし、一定の成果があったと思っています。ただ、その他のツール等に関しては、単発的な投入という印象です。どういう意図で、どのツールを入れたのかなど、SECに全体像を事前に示してほしかった、という感想を持っています。

当社もそうですが、各社は自分たちのプロジェクトマネジメントツールを持っています。ところがCOSEはフラットな組織ですから、私たちのツールを持ち込むことができません。そこで各社には納得していただいてSEC推奨のEPMを導入し、自社ツールと並行運用するなりしながら2年間回してきました。その結果、EPMを少しはいいものにできたのではないかと考えています。

EPMの基本スタンスは、あまり現場においてデータ収集作業を意識させずにプロジェクトデータを自動収集し、分析してフィードバックするというものです。1年目はこちらからSEC、EASEに「こういう分析ツールがほしい」とインプットし、対応していただくところから始めました。

研究者の場合は、ある程度データが貯まってから分析すればいいのでしょうが、現場では設計や単体試験など、工程の区切りごとには整理したいと思っています。そこで、なるべくリ

アルタイムに近づけるため、データの収集にかかる時間については、現場側でも協力し改善して1週間にまで短縮しました。一方、分析ツールも半自動化などを行っています。その結果、COSEの2年目には、各社のサブ的立場の方に分析データをフィードバックし、対応や応用した活用を要請できるようになったわけです。

ここで難しいのは、情報過多の問題です。サブマネは現場と密着していますので、分析データを見なくても感じることができる部分があります。逆に少し現場から離れているマネージャの場合は、分析の数字は問いかけの材料や根拠になる。つまり二面性があるわけです。シチュエーションや置かれている立場によって、データの使い方はさまざまです。その点を理解して、必要な情報の種類や、情報を有効に活用する方法などを検討しておかなければいけません。

いずれにしても、仕組みとして、リアルタイムに分析データを出せるというのはとても重要で、意義があると評価しています。実はこれからマネジメントするプロジェクトがありますので、「こういうモニター方法もあり、計測もしているぞ」と、現場に軽くプレッシャーを与えるのに使ってみたい（笑）。

まだまだ未熟で、道半ばという印象の手法・ツールもありますが、今後のSECの取り組みには期待したいし、ぜひ成果を現場にフィードバックしてほしいと思っています。

3-3 ソフトウェアエンジニアリングの適用

Interview 2

COSEで得た知見をカーナビソフト開発に展開したい

自動車用電装部品の大手メーカーデンソー。デンソーはカーナビメーカーの1社として、ユーザの視点からCOSEに参加した。デンソーのCOSEでの役割から具体的な業務内容とともに、COSEに参加して得られたことなどについて、ITS開発部 第1開発室主幹の塚本晃氏にお話をうかがった。

株式会社デンソー
ITS開発部
第1開発室主幹
塚本　晃氏

ユーザ企業としてノウハウを生かした

── COSEでの役割について教えてください。

COSEの7社のうち、当社とトヨタ自動車の2社は、トヨタ自動車は自動車メーカー、当社はカーナビメーカーの1社、つまりユーザ企業という立場で参加しました。ユーザの立場での参加という意味では、SI企業が開発したプラットフォームがよいかどうかを評価することが

重要な仕事でした。一方、もともとのCOSEプロジェクトの目的であるソフトウェアエンジニアリングを適用したプログラム開発も行いましたので、その面では、開発マネジメントと品質確保について取り組みました。

── 実際に担当したのはどの部分のプログラムですか。

担当したのは収集系プログラムの1つで、具体的には物流業者から収集したデータを、共通車両情報にするプログラムです。つまり提供事業者から収集した情報をもとに、共通のフォーマットにし、データベース化するためのプログラムです。私が携わったのは、このプログラムに関する大枠の仕様の作成と日程管理です。つまりプロジェクト全体を管理しているNTTデータが提示した大枠のスケジュールに間に合うよう、担当モジュール開発の日程を管理することが私の役割です。

今回のプロジェクトは2つのフェーズにわかれており、第1フェーズ（2005年2月～9月）、第2フェーズ（2006年4月～8月）の開発期間にデンソーが担当したのは、いずれも物流衛星通信という提供事業者から情報を集めて共通車両情報にするところまでのプログラムで、変わりません。違いは、第1フェーズは各社独自のソフトウェア開発だったのに対し、第2フェーズではポイントとなる共通部分をモジュール化し、それを使って開発を行ったことです。

当社ではこれまで、提供事業者からデータを集めて交通情報をサービスするという事業に携わったことはありませんでした。ただ、運行管理サービスをASPで提供しており、運行情報を交通情報に変換できるノウハウは持っており、それを生かすことはできました。

── たとえば、バスはバスレーンがあるので渋滞していてもスムーズに走っていたり、集配車両だと、集配の際に止まったりするので、交通情報にするには難しい面もあると思うのですが。

確かに各提供事業者から収集したデータは、同じフォーマットでデータベース化はできました。が、収集された情報には精度や意味にバラつきがあったのも事実です。当社が気をつけたのは、拠点間配送のデータです。拠点近辺の情報は削除しました。「近辺」にしたのは、工場内への配送などがあるからです。しかし、顧客先に向かい、そこで停止したという情報は削除しきれませんでした。

フィードバックがあることでモチベーションを維持できた

── エンピリカルなソフトエンジニアリングを適用した開発を経験した感想について教えてください。

当社が取り組んだプログラムにおいては、正しいスケジュールで品質の良いものができ、望むべきものが完成できたといいたいのですが、実際は試行錯誤の連続でした。当社ではこれまで、ソフトウェアエンジニアリングの経験がほとんどありませんでした。工数は管理していましたが、EPMのようにさまざまな詳細情報を入力するツールを使用するのも初めてのことでした。ですので、ソフトウェアエンジニアリングは難しいということを改めて実感しました。

EPMは入力するデータは一杯あり、作業をしながら入力するのは大変でした。しかし、このデータが後に解析され、フィードバックされると思うことで、モチベーション高く入力できました。いい経験ができたと思います。

── どんな内容がフィードバックされたのですか。

第2フェーズではフィードバックは週1回行われました。主なフィードバックの内容は、ソースコード行数やバグ件数の推移などです。しかし、当社が担当したプログラムの開発期間は短かったため、週次のフィードバックに効果があったというより、日々の入力結果の変化が後で見られることに効果を感じました。もう少し、開発期間が長ければフィードバックの効果を実感できたかもしれません。そこは残念に思っています。

COSEの成果はITS事業で大きな価値になる

── COSEのプロジェクトに参加したことで、デンソーが得たものについて教えてください。

まず、ユーザ企業の観点からすると、開発したプローブ情報プラットフォームソフトウェアが実用化できるかどうかによって、成果は変わってくると思います。2006年10月から、トヨタ自動車とともにユーザ側が使いこなすための実証実験に取り組んできました。その結果、非常によい交通情報が収集できたという実感があります。実現できそうな素地ができたという意味で、成果は得られたと思います。

ソフトウェアエンジニアリングの観点からも当社が得たものは大きかったと思います。今回のシステムは交通情報システムとはいえ、センター系システムだったので、当社から参加し

たのはカーナビの開発部隊ではなく、システム開発部門です。COSEで経験した手法を今後、ITS事業部全体にフィードバックしていくことができれば、大きな価値になると考えています。実はカーナビのソフト開発は今や何百人も関わるほど、機能が付加され大規模になっています。もはやソフトウェアエンジニアリングを使わないとやっていけないぐらい、切羽詰っているのです。そこで役立つのが今回の経験です。カーナビの開発にぜひ、ソフトウェアエンジニアリングを適用していきたいと思っています。

── 今回は国家プロジェクトなので、リソースにゆとりがありましたが、納期が厳しいカーナビの開発に適用するには、まだまだ難しい面もありますよね。

製品の開発スケジュールが決まっているため、すべてのカーナビソフト開発にすぐに適用するというのは、難しいと思います。スケジュールに余裕のあるものにスポット的に適用し、結果をみながらITS事業部全体に展開できればいいなと考えています。しかし、先にも言ったように、カーナビのソフト開発の現場では人も時間も足りず、追い込まれた状況にあります。時間をかけてもカーナビソフトの見える化に取り組んでいかないと、ダメだと思うのです。

── 社内に展開するためのキーポイントについて教えてください。

ソフトウェアの開発現場は、何月何日までに納品しないといけないなど、非常に厳しい条件が課せられている環境です。しかも、これまでソフトウェアエンジニアリングの文化もなかった。そこに適用しようというのですから、本当に効果がなければ、根付かないでしょう。ありきたりの結果を返すだけではなく、何をフィードバックすればうれしいのか、どんな項目を管理すればいいのか、などを突き詰めて考えていくことが必要だと思います。当社の独自の見える化ツールを作るぐらいの気持ちで取り組むつもりです。

開発現場には、役に立たないといわれることもあると思います。しかし、私たちもCOSEプロジェクトにおいて本当に苦労しました。粘り強く展開していきたいと思います。

3-3 ソフトウェアエンジニアリングの適用

Interview 3

1年目の分析結果を2年目に反映、品質向上活動に結びついた

今回のプロジェクトで富士通は、プローブデータ収集機能とデータベース管理機能、および評価画面表示機能の開発を担当した。三浦氏は、同時にプロジェクトのもう1つのテーマであるソフトウェアエンジニアリングにおいても、コードクローン分析、そして相関ルール分析とスキル分析に大きな関心を寄せる。

富士通株式会社
次世代IT・ITSプロジェクト室
テレマティクスプロジェクト
三浦　寿氏

主にデータベース管理機能、評価画面表示機能の構築を担当

── 今回のCOSEプロジェクトにおける富士通の担当と、三浦さんの果たした役割についてお聞かせください。

弊社では、交通情報の源泉となるプローブデータの収集機能開発、システムで共通利用するデータベースの構築、および同共通アクセスAPI機能の開発、生成した交通情報を評価す

るために用いる表示機能の開発を担当させていただきました。その中で私は開発責任者、ソフトウェアエンジニアリング実践責任者を兼任しました。

交通情報の源泉となるプローブデータの収集は、大手タクシー事業者と大手食品製造事業者の物流部門にご協力をお願いしました。実際のデータ収集に際しては、既存の配車管理システムに集まる車両ごとのリアルタイム情報のうち、位置情報（緯度、経度）と収集時刻情報、車両の動態情報を利用させていただいています。

これらの情報は、各事業者、COSEセンター間の専用線を通して、COSE側の機能でリアルタイムに収集し、COSEで扱える形式に変換した上で、共通車両情報データベースに格納します。

共通車両情報データベースには、「共通車両情報」ということで、COSE組合メンバ企業からのさまざまな業務車両プローブ情報が格納されます。このため、その「共通車両情報」はCOSE組合メンバ企業とフォーマットを調整させていただいた上で構築しました。さらに、データベースの共通アクセスAPI機能も提供しております。

収集したプローブデータから交通情報を生成する機能については担当外ですが、その生成した情報を格納するデータベースの構築は担当させていただきました。

さらに、生成された交通情報の評価に必要な表示機能の開発も担当し、プローブ交通情報プラットフォームにおいて、一連のデータフローが把握できる良いポジションで仕事ができました。

── タクシーや物流車両のデータを扱うに当たって、それぞれの特徴に配慮した点などはありましたか？

タクシーの状態には、乗客を乗せて走行している「実車状態」と乗せていない「空車状態」「迎車状態」、およびその他の状態があります。その中で「実車状態」の走行情報だけを拾い、その他の情報は除外するようにしました。というのは、一例ですが、「空車状態」の車両がどのように配置され、どこを走行したのか、という情報はタクシー事業者の営業情報であり、彼らが開示したくない情報だと考えるからです。一方、「実車状態」の情報は乗客の個人情報ではないかと考えられますが、特定の個人と結びつくわけではありませんので、あくまでも交通情報の源泉情報として扱う、ということで収集に臨みました。

一方、物流車両は、「荷卸し」「荷積み」という、停車状態があります。その状態を認識しないで交通情報の源泉としてしまうと、該当道路が渋滞していることになってしまいますので、そのような状況のデータは除外して収集しています。

苦労はあったが、「早く提供できた」という満足感が

── 今回の開発において、大変だったのはどういうことでしたか？

プロジェクト発足時、データベース管理機能の開発を担当することになり、初年度はCOSE組合メンバ企業で共通に利用できるデータベース環境の骨格を作りました。当初、プラットフォームソフトウェアのさまざまな機能が利用するデータベースは、「その種類の決定」と「データ項目の共通化」「APIの共通化」に費やす時間が膨大なものになると予想されました。

皆、データベースアクセスのためのAPI共通化には総論賛成なのですが、実際に実装するとなると、自分たちが利用しやすいものを求めます。さらに、共通APIの利用法は、各社さんが開発する機能のノウハウという理由で公開してもらえません。その一方で「共通化に関する仕様がだいたい決まったら、すぐに提供してほしい」と要求されます。

このようにマルチベンダー開発においては、往々にして共通API仕様の決定が遅れるため、当該機能の提供側にとっては、期限にいかに間に合わせるかに苦労することとなります。この点はCOSE組合メンバ企業も豊富な経験に基づいてよく理解されていました。

ここで私が採ったアプローチは、APIのうち、本当に共通化すべき範囲を明確化するというものです。そこで、インテグレータであるNTTデータさんを始め、開発担当各社さんにご協力をいただき、彼らのノウハウ部分を切り離しました。より具体的には、データベースのセキュリティに関わる部分と、使用データベースを決定する部分に仕様を絞り込んだことです。

その結果、苦労は多かったものの、各社さんのノウハウを侵すことなく、極めて短期間に高品質なものを提供でき、その満足感を得ることができました。

もう1つ苦労した点も、やはりデータベース機能にあります。共通機能を提供している立場から、データベース利用に関する問題が発生した場合、よく調査依頼を受けます。その中でも一番大変だったのは、データベースアクセス時の性能劣化問題です。他社開発のアクセス側アプリケーションは、ノウハウを含むため、ソースプログラムを入手することができません。プログラムが解析できないのに、問題解決のためのアドバイスをしなければならないわけです。原因は提供している共通APIにあるのか、本体にあるのか、現象のみから原因を究明するもどかしさ。1つ見れば済むかもしれないことを10～20も調査して、さまざまな角度から原因を検討しなければならなかったことは、それなりに辛い仕事でした。

── 逆に、共同作業だから良かったことは？

今回の実験のような交通情報の作成においては、いかに多くの情報を集められるかが命です。最終的に常時6000以上のデータをリアルタイムに収集できたのは、COSE組合メンバ企業が協調して情報を1つの器に入れたからです。そして、そうして集めたさまざまな情報から交通情報が生成できる、という確証を得られたのが最大の成果だと私は思っています。

コードクローン分析に一番の関心

── COSEプロジェクトにおけるもう1つの目的、ソフトウェアエンジニアリングの面では、どういう成果がありましたでしょうか？

私個人としては、ソフトウェアエンジニアリングの実践に大きな関心を抱いていました。その中で今回、COSEの活動を2年間続けさせていただいて、一番関心を持ったのはコードクローン分析です。開発メンバが自分なりに作ったソースコードを客観的に分析し、簡単にいえばコピー＆ペースト的なものがどこにあるかを探り、指摘していただけたことです。

複製したソースコードをそのまま放置しておくと、その一部に障害が発生した場合、「当該部分は直したけれども、コピーで作った他の部分は修正できていない」ということが当然あるわけです。この分析を行っておけば、障害が見つかった場合、他のどこに同様の問題が潜んでいるかという情報が得られ、関連障害の有無を容易に確認することができます。

開発メンバが新しいプログラムを開発する際、以前作成したソースプログラムからその一部をコピー＆ペーストすることは結構あります。実際、同一の記述を全体として多用しすぎているような場合、部品化を検討した方が良いと思うことも少なくありません。そして、個々のメンバがソースコード開発に従事している段階（プログラミング）では、全体を俯瞰して処理を共通化するといったことには、なかなか気が回らないのが現状でしょう。そのような場合、初めのうちこそコピー＆ペースト箇所を覚えているものの、工程上、次のステップに移る頃には、かなりの部分を忘れてしまうことが多いのではないかと思います。

実際、ソースプログラムのコンパイル時に、コピー＆ペースト部分にコードエラーが発生する段階では、同様なエラーが複数現れることも多く、すぐにそれとわかるものなのです。しかし、コンパイラを通った（試験段階に入った）ソースプログラムに対して、コードレビューを行い、コピー＆ペースト部分を見つけ出すのはかえって難しいものです。ですから、複製されたソースプログラムの部位を指摘するコードクローン分析は、試験工程のバグ発生時に用い

ると、類似バグの抽出が効率的に行え、プログラム全体の品質向上に役立ちます。

可能性を感じた相関ルール分析とスキル分析

もう1つ、「開発メンバ間、プログラムモジュール間の類似性分析」を行っていただいた結果も興味深いものでした。ここでの相関ルール分析により、「この人はこういう傾向があり、どういうバグを作り込みやすい」などの障害傾向が客観的にわかります。スキル分析では、「プロジェクトメンバのスキル構成はこのようになっています」という指摘が結構合っていました。

私どもが「ここは新人でも大丈夫」「ここはベテランでないと難しい」と考えて決定したプロジェクトの体制が正しかったかどうか、この相関ルール分析とスキル分析で判断できます。ここで得た情報は、今後、類似プロジェクトを推進していく際の参考になると考えています。

コードクローン分析、相関ルール分析による指摘はやはり、新人に集まりました。「この人はこういうロジックの時によくミスをする」という傾向がわかります。しかし、ベテランプログラマの場合にも、大なり小なり、なんらかの指摘はあるもので、それらの指摘から個々のプログラマの傾向を知ることにより、ソースコードのチェックを効果的に行うことができました。従来は、レビュアの主観的チェックがなされるのみでしたが、これらのツールの活用により、客観的、重点的なチェックを可能とし、レビューの効率と品質を高めることができました。

良かったことの3つ目は、スキル分析を細かく実施いただいたことです。その結果だけで開発体制の有効性を判断するまでには至りませんでしたが、ソフトウェア開発の各工程において、体制の強み弱みなどが明確になり、体制強化や人員配置の指針することができると感じました。このスキル分析は最終年度の後半に実施したため、実際に体制を変え効果を確認するまでには至らず残念です。

1年目の開発で指摘していただいた各種の分析結果を踏まえ、2年目の開発に臨みました。その結果、ソースコードのステップ数を削減し、プログラム単体テスト時の障害件数を低減することができました。そこには開発効率改善という面もありますが、より大きいのは、開発したアプリケーションの品質向上という点ではないかと思っています。

今回のソフトウェアエンジニアリング技法の実践で気づいた課題についてコメントすると、SEC さんに開発成果をさまざまな切り口で分析していただいたのですが、学術的な観点に留まり、「私たちのわかるレベルまでブレークダウンしていない」「実際の場面で使いこなせるよ

うになるには時間を要するであろう」と感じざるをえないものも多々見受けられました。たとえば、ソースプログラムの修正履歴から同時更新関係を推定するロジカルカップリング分析では、結果を具体的にどのように利活用すれば良いのか、いまひとつわかりませんでした。また、協調フィルタリングによる見積りについても同様です。今回は、設計工程終了までの開発実績データを使って、類似事例より製造工程以降の工数を予測していただきましたが、予測データと実績データを比較して何がいえるのかが明確でなかったと思います。

デファクトになりうる開発支援ツールの充実を

── 今回COSEに参加した経験などから、SECに期待することはありますか。

開発の現場で使える、各種支援ツールの充実です。コードクローン分析や相関ルール分析は今すぐにでも使用したい分析ツールですが、もっと使いやすくなることを望みます。

協調フィルタリングによる見積りは、根拠となる過去事例が増え、見積精度の高さが見極められれば、下流工程で有用な技法になっていくと想像しています。

今回のソフトウェアエンジニアリングの実践で利用した各種ツールが、無償または安価なソフトウェア品質向上ツールとしてリリースされて欲しいと思います。近い将来、開発支援ツールとしてスタンダードとなるべく整備されることを期待いたします。

3-3 ソフトウェアエンジニアリングの適用

Interview 4

COSEで採用した手法は中小規模プロジェクトにおいても選択肢の1つとなる

COSEプロジェクトのメンバ企業の1社である日本電気（NEC）で、プロジェクトマネジメントを支援する役割を担っていたのが、岡光彦氏である。岡氏は、比較的取組みが進んでいる大規模プロジェクトに対し、遅れがちな中小規模プロジェクトへ今回の成果が適用できるのではないかと注目する。

日本電気株式会社
第一官庁システム事業部
主任
岡　光彦氏

ソースの類似性という客観的な観点から対象を特定

── 実際にNECが担当した業務について教えてください。

COSEで開発したプログラムは、大きく情報収集系と情報処理系にわかれます。情報収集系はPMであるNTTデータを除く全社で取り組みましたが、情報処理系は松下電器産業、日立製作所とNECの3社のみで担当したため、当社では1社当たりの責任が重くなる情報処理

系に注力してプロジェクトに取り組みました。

情報処理系プログラムとは、実際に走っている車の情報をもとに、渋滞情報や交通情報などを作るプログラムのことです。旅行時間（ある区間における自動車の走行時間）を生成するプログラムは松下電器と日立製作所が担当しました。当社は、交通情報のない場所における現実の状況を、先の旅行時間をもとに推測してデータ化するプログラムを開発しました。このプロジェクトの肝となるのは、データベース連携だったため、3社間で仕様決めなどをする際は、よく話し合いました。とくに松下電器とは、あるプログラムの仕様を調整するため、技術者が参加する個別ミーティングを何度も設けるなど、かなり突っ込んだやり取りを行いました。

── COSE適用したソフトウェアエンジニアリングについての感想を教えてください。

CVSやGNATSにデータを入力したことで、品質だけでなく、進捗、構成管理まで適用の範囲が広がったと実感しました。当社でも、独自ツールを導入しプロジェクトを管理しています。今回のCOSEプロジェクトでは、CVSやGNATSとともに当社独自ツールへの入力というように、二重入力で対応しましたが、当社製のツールは品質会計（作り込んだバグを負債とみなし、レビューやテストでバグ摘出することで負債を返済。負債がなくなった時点で出荷するという考え方）を採用していますが、今回のような短期開発では効果を確認しにくく、その点でCVS、GNATSは有効でした。プロジェクトの規模を見積もり、このフェーズではどのくらいのバグが出るという予実を管理しているからです。

── 分析結果のフィードバックで、どのようなことが得られましたか。

今回のプロジェクトでは、ソースの類似性という観点で管理する共通認識ができていました。通常、不具合を水平展開する際、PMの個人的経験に基づく偏った視点や、不具合を出した開発担当者のスキルに問題があったのでは、というネガティブな視点で対象を掘り起こすことが行われます。しかし今回は、ソースの類似性という客観的な観点から、対象を特定することができました。これによりプロジェクトとしての網羅性、処置の即応性、開発担当者のモチベーション維持という3点に、非常に効果があったと感じました。

期待していたよりも、作る側の立場で分析されており、有益でした。社内でも同じような取り組みをしていますが、フィードバックが実際とかい離していることがあります。その点、今回は現場に近い結果で、素晴らしいと感じました。これも1つの成果です。

また、分析を担当したEASEのメンバとも思った以上に、ダイナミックなやり取りができました。残念だったのは「こういう傾向が出ているから、こんな行動をとるほうがよい」というところまでの分析はできなかったことです。

オープンソースツールは協力会社にも導入しやすい

── COSEの分析手法の社内への展開について。

当社での品質への取り組みは、大規模プロジェクトにはマッチしていますが、私が日ごろPM的な立場で携わる中小規模のプロジェクトでは効果が確認しづらいと考えていました。今回、COSEで適用したCVSやGNATSは、中小のプロジェクトにも使いやすいと感じました。しかもCVSなどのオープンソースは、当社だけではなく協力会社も使えます。また参考文献や事例も多い。このような標準的なものを使う方が、メリットも大きい気もしており、社内へも反映したいと思いますが、なかなか難しい面があります。というのも、社内ツールを今後も使っていかなければならないからです。したがって、それを中小向けにカスタマイズすることも視野に入れて考えていきたいと思います。

── マルチベンダー制のプロジェクトに参画した感想について教えてください。

通常のマルチベンダー体制のプロジェクトであれば、PMとしてプロジェクトを統括する元請会社と、各ソフトウェア開発を担当する会社が契約し、主従の関係で仕事を進めます。このようなプロジェクトの経験は、これまでもいくつかあります。しかし、今回のように全体を統括するPMと、開発を請け負う各社が対等な立場であるプロジェクトは初めての経験でした。各社が情報公開をせず、予算のコントロールもできない中で全体を統括しなければNTTデータにとっては、非常に難しいプロジェクトマネジメントだったと思います。

── 実際の成果物に対する評価はいかがですか。

2005年に開発したものと、2006年に開発したものを比べると、比較にならないほど進化を遂げました。それは当社だけの力ではできなかったことだと思います。各社の専門家が集まり、知恵を結集したからです。ここで開発した部品は、他のプロジェクトに展開できると思います。そういう意味では大きな成果が得られました。またプローブ情報プラットフォームソフトウェアの開発において、大きな役割を果たせたと思っています。

SECに中小規模プロジェクトのデータ収集と分析を期待

── COSEプロジェクトで得られたノウハウを、今後どのように生かしていきたいですか。

COSEの目的は、短期間で高品質なソフトウェア開発を可能にすることでした。これは成功

を収めたと思います。それを可能にした最大の要因が、各会社の窓口、技術担当者、プロジェクトマネジメントチームという、3つのリソースの競合が最大限配慮されていたことです。

たとえば、各会社の窓口のリソース競合が回避されていたことでは、次のような効果がありました。通常、このような複数のベンダーが関わるプロジェクトの場合、各会社にプロジェクトリーダー（PL）が立ち、窓口的な役割を担います。そしてこのリーダーはよほど大規模ではない限り、プロジェクトのかけもちをしているため、ボトルネックとなり、短期間という目標実現への障壁となります。COSEプロジェクトでは、ソフトウェア製造フェーズにおいて、各会社に窓口を介在させるのではなく、SE部会の独立性を高め、そこに参加している各社の技術担当者が中心となって進めることができ、短期間を実現できたわけです。

中心となった技術担当者も、ほぼ専任という形でこのプロジェクトに従事していました。さらにプロジェクトマネジメントチームも、調整対象はたくさんあったにも関わらず、PMのプロジェクト滞在率は高かったと思います。また常時3人体制を採用していたため、プロジェクトマネジメントチームが、プロジェクトルームから1人残らずいなくなるという事態はほとんどなく、ボトルネックの回避とメンバの緊張感の維持に貢献していたと思います。

これらのことは、プロジェクトマネジメントの本を見れば、当り前のように書かれていることです。しかし、現実にはなかなかできていません。今後、私が携わる中小プロジェクトにおいても、このようなリソース競合の回避を行った場合、行わない場合の期間や品質、そしてコストに与える影響などを比較し、検討していきたいと思っています。

── ソフトウェアエンジニアリングに期待することは？

中小規模のプロジェクトでは、プロジェクトマネジメント情報を改めて入力しなくても、PMには経験として蓄積されるため、情報入力そのものを敬遠する、有効なデータが入力されないなどの状況があります。しかし大規模プロジェクトにおいては、ソフトウェアエンジニアリングを用いる効果は大きく現われています。したがって、中小規模においても、適用していかなければなりません。そこでぜひ、SECにお願いしたいのは、中小規模のプロジェクトにおけるソフトウェアエンジニアリングデータを多く集め、その効果を広く発表してほしい。そうすることで、社内はもちろん業界全体が改善できると思うのです。

3-4 エンピリカルソフトウェアエンジニアリング

Interview 1

ソフトウェアの品質を高めるコードクローン分析ツールの開発

EASEプロジェクトのメンバである大阪大学大学院情報科学科研究科の井上克郎教授・楠本真二教授の研究グループでは、これまで約7年間ほど研究を続けているコードクローン分析手法をCOSEプロジェクトに適用した。分析作業を実際に行ったのが吉田則裕氏と馬場慎太郎氏だ。吉田氏はこの経験をもとに博士号取得を、修士1年生の馬場氏は、修士論文の作成を目指す。実際の開発プロジェクトにおいて、分析手法の有効性を各ベンダーに評価してもらったのである。コードクローン技術の概要から、COSEに適用したことで得られた知見などについて、お話をうかがった。

大阪大学大学院
情報科学科研究科
教授
楠本真二氏

コードクローン、良し悪しの評価は今後の課題

── EASEとCOSEの関係について教えてください。

楠本　EASEはご存知のように大学と企業が集まって、ソフトウェア工学へのエンピリカルアプローチを研究しているプロジェクトです。大学は社会的な観点からするとアカデミックな視点から研究を行う機関ですが、ソフトウェア工学をテーマとしている私たちは、現場に接す

ることも必要で、これまでにも企業と連携し実証の場を設けてきました。COSEもその場の1つという位置づけです。

ただこれまでの実証実験との違いは、COSEは規模も中規模と大きく、マルチベンダーのプロジェクトであることです。

吉田　これまで私たちがコードクローン分析手法を適用してきたものは、オープンソースソフトウェアなどで、規模的には小さいものがほとんどでした。COSEはソースコードの規模も、今までになく大きいものでした。

大阪大学大学院情報科学研究科
コンピュータサイエンス専攻
ソフトウェア工学講座
吉田則裕氏

── 今回適用された、コードクローン分析ツールの概要について教えてください。

楠本　コードクローンとは、ソースコードの中で類似もしくは一致した部分のことです。一般的には、コードクローンがプログラム中にたくさんあると、保守性を低下させるといわれています。というのも、あるコードクローンにバグを発見した場合、そのクローンと対応するコードクローンをすべて探して、修正を行わなければならないことが生じるからです。

世界各地で研究されているテーマですが、私たちが研究を始めたのは、大規模なレガシーシステムからコードクローンを効率的に見つけたいという要求があったからです。以来、7年間ほどかけてコードクローン分析ツールを開発してきています。オープンソースソフトウェアで有効性を検証し、ある程度使えることがわかりました。ただ、実際の開発現場に適用してみると、課題も見つかりました。

── どういう課題ですか。

楠本　企業が開発するソフトウェアコードをクローン分析する点で課題となるのは、すべてのコードクローンが悪いとはいえないことです。たとえば、開発ベンダー側からすると、コードクローンが数多くあっても、動くシステムができれば、プロジェクトは成功となる場合もあります。この場合、クローンはプロジェクトにとってよいものということができるでしょう。

一方、保守をするユーザ企業側からすると、クローンが多く含まれていることでシステムの保守性が下がってしまい、悪いクローンと考えられます。つまり同じクローンでも、視点や条件、開発プロジェクトの特性によっても定義が変わるのです。どういう見つけ方が効果的か、

条件や視点を含め、クローンの良し悪しの評価の方法を探るため、データを集め、研究していくことは、今後の課題です。

企業のコードとオープンソースのコードでは異なる

── COSEプロジェクトでは、どのようなタイミングで分析を行ったのですか。

吉田　私と馬場で昨年度は各社が開発し終わった後のソースコード、今年度は結合後のソースコードを分析しました。

楠本　今年度は新たに複雑さの分析も行っています。大変だったのは分析をする際に、1年目は東京都文京区にあるIPA内のSECのデータ分析室に行かなければならなかったことです。2年目は大阪府豊中市千里中央駅前にあるエンピリカルソフトウェアラボ内に機密室が設置され、そこでの分析が可能になりました。大学から10～15分でいける場所なので、助かりました。分析結果のフィードバックは2年間で数回行いました。

── COSEにツールを適用して、得られた知見や感想について教えてください。

楠本　本来、コードクローン分析は、既存のコードを機能拡張するなど、保守を継続するシステムに有効な手法です。しかしCOSEで開発されるシステムは、保守を継続していくというものではありません。むしろ新規開発案件です。その点では多少、コードクローン分析の目的とずれるところはありましたが、クローンの見つけ方の有効性がどこまで現場で受け入れられるかという評価の1つが得られたことは確かだと思います。

吉田　オープンソースソフトウェアの分析と、実際の会社が開発したソースコードを分析するのとでは大きく違うことがわかりました。オープンソースではうまくいっていた手法をCOSEに適用しても、うまくいかないこともありました。このことから、研究室で考えた手法が、そのまま産業界のシステムに適用できるわけではないということがよくわかりました。また、産業界で活躍する開発者との議論を通して、問題を引き起こすクローンであるかどうかを判断することはなかなか難しいということを実感しました。このような現実を理解できたことが、一番の収穫です。今後の研究に生かしていきたいと思います。

馬場　これまでにも会社のソースコードを分析したことはありましたが、COSEのように複数社のソースコードを一度に分析したのは初めてです。各社によって取り組みも異なります。そこで分析結果を各社ごと、分布図にまとめました。一度にこういう結果を得られたことは、非常に勉強になりましたし、1年後に就職を控える私にとっては、各企業の文化や風土も感じ

られ、就職を考える際の指針にもなりました。

企業の分析ツールへの関心を呼び覚ます契機になった

コンピュータサイエンス専攻
ソフトウェア設計学講座
馬場慎太郎氏

── 今後の展望について教えてください。

楠本　今回、COSEへの適用を機に、コードクローン分析ツールへの興味も多少、高まってきたと思います。コードクローン分析技術へ関心が向かなかったのは、クローンによる致命的な被害がまだ公表されていないからではないかと考えています。コードクローンに限りませんが、ソフトウェア開発の世界では、たとえ不具合があっても、その原因についてはあまり公開されていないと思います。

しかし今回、COSEプロジェクトで適用をした企業の中には、分析ツールに関心を抱いてくれた企業もあります。それらの企業と綿密に連携し、コードクローンの良し悪しを判断する条件を追求していき、一般化していきたいと思っています。

吉田　コードクローン分析ツールを世の中に広めていくことに貢献していくのはもちろんですが、それに限らず、リファクタリングなど、ソフトウェアのソースコードの品質向上をテーマに、今後も研究に取り組んでいきたいと考えています。

馬場　コードクローンとソースコードの複雑度を専門分野として研究を続けていきたいと思っています。

楠本　コードクローン分析に限りませんが、ソースコード分析の研究を続けていくためには、まだまだ乗り越えなければならない障壁があります。たとえば、守秘義務契約もその1つ。しかもベンダーだけではなく、ユーザ企業の許可をとることも必要でしょう。

しかし、今回のCOSEプロジェクトによって、エンピリカルアプローチというソフトウェアエンジニアリングの有効性が、参加企業には伝わったと思います。このような機会をまた作り、クローンの定義、見つけ方などを探っていくことと同時に、開発現場のコードクローンへの意識を高めていくことが、今後の課題ですね。

3-4 エンピリカルソフトウェアエンジニアリング

Interview 2

EASE協調フィルタリング法をSEC1000プロジェクトデータに適用

奈良先端大学ソフトウェア工学講座の門田暁人助教授と大杉直樹特任助手のグループでは、EASE強調フィルタリング法に基づく見積もりツール「magi」を開発、COSEプロジェクトへ適用を図った。協調フィルタリングの概要と開発したツールの特徴、COSEへ適用したことで得られた知見などについて、門田助教授と大杉特任助手に話を聞いた。

奈良先端科学技術大学院大学
情報科学研究科
ソフトウェア工学講座助教授
門田暁人氏

過去のデータから類似プロジェクトを探索

――協調フィルタリングとはどんなものか、教えてください。

門田　協調フィルタリングとは、通販サイトなどで「この商品を買っている人は、こんな商品も買っています」といった推薦に利用されている技術で、類似した商品を探し出すことにより推薦を行っています。私たちのグループでは、この技術を応用し、あるソフトウェア開発の工

数を見積もる際に、過去に同じようなプロジェクトがあったかどうか探し出し、その際にかかった開発コストやテスト時間などをもとに見積もり値を算出します。

今回、私たちが開発したのは、EASE協調フィルタリング法に基づく見積もりツール「Magi」。それをCOSEのプロジェクトに適用し、分析を行いました。分析に利用したプロジェクトデータは、SECが収集したものです。1000件のプロジェクトデータの中から各社の開発プロジェクトに似ているものを探し出し、そのデータを各社に提示しました。

奈良先端科学技術大学院大学
EASEプロジェクト特任助手
大杉直樹氏

── どんな項目で類似度を測るのですか？またどんな項目の予測が可能なのでしょう。

門田　過去のプロジェクトが残している項目にもよりますが、開発工数やテストの回数、期間、開発人数などを予測できます。

大杉　COSE では規模や業種、開発スタイルなど、SEC で収集しているデータ項目で類似度を測り、工数の予測を行いました。今はリスクの予測も行っています。何をもって類似とするかは難しい問題です。そして、ここがツールの「キモ」ですが、類似度を計算する時、見積りの精度（正確さ）が向上するデータ項目だけを、ツール側で自動選択する仕組みを組み込んでいます。つまり、うまく見積りできるようにツールが類似プロジェクトを選んでくれるということです。

── 協調フィルタリングを使って予測するよさとは。

門田　従来の方法では例えば、「試験工数＝26.5+ 設計工数×0.25」という式にあてはめて予測していました。しかし、開発プロジェクトが多様化している今、このような式に当てはまるとは限りません。そこで、その都度、類似プロジェクトを見つけて予測するほうが制度の高い見積もりができると考えたわけです。

実際、情報サービス産業協会が2004年に行った調査でも、「開発工数の見積もり方は過去の類似システムを参考にしている」と回答している企業が非常に多いです。しかし問題なのは、収集されているデータには多くの欠損値があることです。実はこのような虫食いデータをうま

く活用して予測できるのが、協調フィルタリングなんです。さらにMagiであれば、過去の類似プロジェクトも提示できるのも、メリットだと思います。

大杉　COSEでは、SECが2005年度に収集した1000件のプロジェクトデータをもとに予測をしました。

EPMとともにパッケージ化

── 予測の精度を上げるには、サンプル数が多ければ多いほどいいということですか。

大杉　やみくもに多くても駄目です。何らかの原因で見かけの生産性が極端に高いプロジェクトや、炎上したプロジェクトなど、いわゆるノイズのようなプロジェクトも混じってしまうためです。現在、そのようなノイズのようなプロジェクトを自動削除することに取り組んでいます。

── どの段階で見積もれば、より精度の高い見積もりができるのでしょう。

門田　基本的には、お客様からRFP などをいただいた直後に原価を見積もるために使います。その他、お客様へ提示する価格の決定や、受注判定時に適正受注価格を算出するためにも使えます。詳細設計の段階まで進んで工数を見積もるとSECのデータの場合はおよそ30～70%の誤差で見積もることができます。ただ、難しいのはプログラムの再利用の割合や外注先の働き度合いが、収集されたデータからは見えてこない。そのため、厳密には開発工数は測れないということになる。ここが問題です。

大杉　SECが収集しているデータは、経理上の開発工数であり、実際にかかっている工数とは異なるということも考えられます。しかし社内でご利用いただく時は、実際の工数がわかるので、そのデータを使えばより精度の高い見積もりはできると思います。

── 協調フィルタリングツール「Magi」の特徴を教えてください。

大杉　Magiは予測した値を出すだけではなく、類似プロジェクトの概要や特徴、元データも同時に提供されます。COSEにMagiを適用し、各社から意見を出してもらいました。役に立つ、自社でも適用したいという声が少なからずありました。

門田　Magiは見積もりのもととなる過去データの正確性を、的確に評価する機能も充実しています。これも大きな特徴です。

大杉　現在、さらにどのデータ項目が見積もりに役に立つ、役に立たないかを自動判別するツールも開発しています。

門田　予測のもととなるデータ自体を評価するツールは、これまでありませんでしたから、

Magiへの期待も高まると思います。

── Magiはどのような形で公開されるのでしょう。

門田　EPMとともにパッケージ化をして、2007年4月から配布する予定です。これは2006年9月に完成した実験用のツールです。新バージョンはまだ検討中ですが、大学のホームページでユーザ登録し、ライセンスキーを取得後、ダウンロードする形で公開する予定です。同時にユーザにアンケートをお願いし、ツールへの意見などフィードバックの仕掛けも作る予定です。意見をもらうことで、さらにツールの精度向上を図っていきたいと考えています。

COSEからの意見でより洗練された見積方法を

── COSEに関わったことで得られた知見とは。

門田　COSEプロジェクトでは、各企業でも独自に見積もりを行っていました。その見積もり結果とMagiによる見積もり結果に相違のある場合もありました。その時に、なぜツールがそのような見積もり結果を出したのか、根拠を示すことが重要だと感じました。単に見積もり値を提示するだけでは現場でツールを使ってもらうことは難しいと思います。より現場に受け入れられやすい方法を検討していきたいと考えています。

大杉　SECの1000プロジェクトデータのように、複数のベンダーのプロジェクトがデータに含まれる場合の課題は少しずつ見えてきました。データ中に多くの穴あき部分（欠損値）が含まれること、それから、性質が大きく異なるプロジェクト群が存在することです。これは、多くの部署やグループ会社を抱える企業でも起こり得る問題です。我々のMagiはそれらの問題にうまく対処できると考えています。

門田　協調フィルタリングはデータに基づくソフトウェア開発支援の1つでしかありません。今後も、この分野にこだわり、研究・開発にまい進していきたいと思います。なお、この研究プロジェクトは、博士課程学生の角田雅照さん、柿元健さん、戸田航史さんと協力して進めてきました。今後、彼らと協力して研究論文としても日本発の技術を世界に広めていきたいと思います

大杉　個人的にこれから取り組んでいきたいと思っているのが見積もりの方法の高度化です。協調フィルタリングでもファンクションポイント法でも手が届かないところはまだまだある。そういうところをカバーできる見積もり方法を提供できるよう、研究していきたいですね。

3-4 エンピリカルソフトウェアエンジニアリング

Interview 3

EPMをCOSEに適用、分析を担当して得られたもの

EPMを開発し、実際にCOSEプロジェクトから収集したデータの分析を行ったのが、EASEプロジェクトメンバで奈良先端科学技術大学院大学の特任助手である松村知子氏、森崎修司氏、玉田春昭氏の3人である。COSEにEPMを適用して得られたもの、苦労した点、研究者としての今後の展望などについてお話を聞いた。

奈良先端科学技術大学院大学
EASEプロジェクト特任助手
松村知子氏

ツール導入の普及活動からはじめた

── EASEでこれまで取り組んできた研究内容とCOSEプロジェクトでの関係について教えてください。

松村　EASEでは、EPMで収集した構成管理のデータを、時系列にできるだけリアルタイムにモニタリングして、早期に問題を発見できるような支援方法について研究してきました。

奈良先端科学技術大学院大学
EASEプロジェクト特任助手
森崎修司氏

COSEでの役割もこれまでの研究の延長で、EPMで収集したデータを時系列に分析した結果をレポートにまとめ、毎週、PMと各社にフィードバックしてその有効性を検証することに、2年間携わってきました。

玉田　私がこれまで携わってきた研究は、プログラムが動作するために不可欠な情報を抽出し、それを比較することでプログラムの類似性を計測する研究です。この時の特徴をバースマークと名付けました。開発現場では、既存のプログラムを流用して新しいプロダクトを作る、という流用開発が多く見られます。この流用度合いが、テストに割く工数を決める要因の1つになるのでは、と考えました。そこで実際に流用して開発されたプロダクトにおいて、改変が加えられた度合いをバースマークを用いて計測し、どれだけ工数を決める手掛かりになっているのかを研究しているというわけです。これはテスト工数の配分決定のための手掛かりだけではなく、プログラムのどの部分が変更されているかという差分を調べることもでき、外注したプログラムの検証箇所の特定にも役立つと考えています。

COSEとの関係は、松村さんが開発したEPMの可視化部分の拡張ツール「Pro*」のアーキテクチャの設計や、Pro*とEPMをどう連携すればよいのか、という検討を行いました。

森崎　私は情報通信系の民間企業で5 年ほど、開発/プロジェクトマネジメントの経験を積んでいたのですが、EASE の活動を知り、「商用ソフトウェア開発を対象としたエンピリカルソフトウェア工学の研究に携わりたい」という思いから、EASE プロジェクトおよび奈良先端大学に転籍しました。その一環としてCOSE プロジェクトを担当することになりました。私が日ごろ取り組んでいる具体的な研究テーマは3 つあります。1つ目が相関ルール分析です。相関ルール分析は、IBMの研究者の方が小売店舗やスーパーマーケットのPOSデータから併せ買いを分析するために適用した手法なのですが、それをソフトウェア開発中に収集できるデータに適用しようというものです。2 つ目はマイクロプロセス分析。奈良先端大学の飯田研究室(ソフトウェア設計学講座) の皆さんとのコラボレーションとしてテーマを進めています。EPMなどで自動収集をできるようになると、低コストで細かい開発プロセスが時系列に見られるようになります。時系列の詳細なプロセスを分析し、品質を計測するという研究です。3 つ目が

ネットワーク上における知識共有で、他の2つと比べるとこちらは細々とやっています。

── COSEでの作業期間と役割について教えてください。

松村　作業を開始したのは、2005年1月からです。2005年度はツール導入に関する普及活動を行いました。具体的に行ったことは、どうやってデータを集めるかというプロセスの定義の決定と集めるデータの取捨選択です。EPMの本格的な適用は、2006年度からです。各自のテーマにしたがって分析し、その結果を各社にフィードバックしたり、成果として発表したりしました。これらの活動の中で得られた企業の開発関係者や研究者からの意見をもとにEPMの仕様を変更したり機能を盛り込んだりして、バージョンアップしていきました。

森崎　COSEではデータの分析をサービスとして提供しました。したがって、ツールの使い方に関するサポートだけにとどまらず、意図通りのデータを集めることを目指した開発プロセスへの組み込みや指摘事項の検討など、高いレベルのサポート作業が多く含まれていました。

松村　分析作業に入る前の根回しから、分析が始まってからもちゃんとしたデータが集まるように各企業にお願いするなど、使えるデータを揃えることに、2年間を費やした気もします(笑)。実際のプロジェクトに新たなデータ収集・分析技術を導入するには前段階として考慮しないといけないことがあるということを、把握できたのは財産になりましたね。

自動収集しなければデータは集まらない

── 分析結果に関して、COSEではどのような評価を得られたのでしょう。

松村　開発メンバに分析した結果をフィードバックし、「こういう単位で分析した方がいいのでは」「このデータは現場では意味がない」「このデータに関しては毎日見たい」など、分析方法の用い方に関する現実的な意見をもらったことは、非常に大きな収穫となりました。

── 皆さん、企業で働いた経験もありますが、研究者として参加することで何か得たことはありますか。

森崎　集める側の立場になってわかったことは、自動収集しないと客観的なデータはなかなか集まらないということ。したがって、EASEの自動収集というアプローチは非常にいいと思います。また現場の開発者たちは大学の研究者というだけで、「現場がわからない」と思っている人も多い。そういう人たちとのコミュニケーションの難しさも感じました。

玉田　私は前職では、プログラマでした。したがってデータを収集される側の気持ちはよくわかります。手作業でデータを入力するような仕組みでは、どうしてもいい加減になり、分析に

耐えられるデータは集まらない。森崎さんも指摘しているように、EPMの自動収集はいいアプローチです。今、人が介在して収集しているデータも自動化するなど、分析のもととなるデータをどれだけ、自動的に収集できるようにしていくか、考えていきたいと思います。

奈良先端科学技術大学院大学
EASEプロジェクト特任助手
玉田春昭氏

松村　私は前職では機械の組込みソフトの開発に携わってきました。組込みの世界では、エンタープライズシステムほど、プロジェクトマネジメントがしっかり行われていない。だから素直に、「エンタープライズはすごい」と感動しました。考えてみれば、これまでデータを収集し分析するという文化がなかった組込み系にこそ、EPMの導入は役立つのではないかという気がしています。今後はぜひ、組込み系にも広げていきたいと思います。

── マルチベンダー型のプロジェクトに参加して気づいたことはありますか。

森崎　各社の文化や社風の違いは、データの取り方にも表れているような気がしました。体制は各社でかなり異なっていましたし、いろいろな側面から各企業の顔を見ることができたことは貴重な経験になりました。

松村　COSEでは、構成管理ツールと障害追跡ツール（バグ票）のデータを収集したのですが、会社によって、自社ツールを使ったり、自社ツールとEPM対応の管理ツールを併用したりと、社内のデータ収集への取り組み方はさまざまでした。自動収集で取得するデータ項目は決まっていたとはいえ、このような各社の取り組み方によって、文化や社風の違いが表れたのかもしれません。

現場でのデータ収集のノウハウが得られた

── ツールの有効性に関してはどのような評価が得られたのですか。

松村　COSEのプロジェクトでは、各社のデータ分析結果は各社に、プロジェクト全体を統括しているPMには、すべての分析結果をフィードバックしています。しかし分析結果のみを報告するという形だったので、EPMツール自体に対する評価は特になかったような気がします。

森崎　今回の結果は、サービスとして提供しているため、収集ツールの評価は聞いていますが、分析ツールの有効性についてはCOSE の方々の直接の評価は聞いていません。しかしながら、PMの勝又さんにとっては、ご自身が持っている情報にEPMの情報をプラスして判断ができるようになったと聞きましたし、ミーティングでは私たちがフィードバックをした資料を活用した会議も行うこともできました。またPMがリスク管理する際においても、「○○機能がおかしいのでは？」と具体的に指摘できるようになりました。これらを考慮にいれると、有効性はあったといえると思います。

── COSEのプロジェクトに参加したことで、研究者としてはどんな成果が得られたのでしょう。

森崎　このような機会を与えてくれたCOSEとSECには感謝しています。今回の結果をもとに記述した相関ルール分析に関する論文が、ワールドワイドの専門家が集まるシンポジウムで採録されることになりました。採録される割合は、全体の28%と非常に狭き門。今回の成果が認められたということです。研究者としては、ありがたいことです。しかしまだCOSEへ適用した手法は、手作業のサービスの域を出ず、誰もが使えるようなツールのレベルには達していません。攻略本をつけて、より多くの人に使ってもらえるようにするためにも、興味をお持ちの方にはぜひ私たちの仲間になっていただき、一緒に活動したいと思います。

玉田　COSEに適用した分析手法を、現在、他の企業にも展開しようとしています。しかしすでに自社製のデータ収集ツールを持っている企業に、EPMに置き換えてほしいとはいえません。そこで各企業の独自ツールでも収集、分析できるように、機能拡張を行っています。これが進むと、より多くのデータが集まり、いろんな分析ができるようになるのでは、と期待しています。

松村　COSEに参加したことで、開発現場でのデータの収集に関するノウハウがたまりました。私は2007年3月で（取材は2007年3月）EASEプロジェクトを離れることになりましたが、COSEでの経験を生かして、定量的にデータを測り、プロジェクトを管理する研究を続けていきたいと思います。

森崎　COSEに適用した手法はエンタープライズに限らず、組込みでもパッケージ開発でも応用できると思います。それらの分野にも広げていきたいですね。

「プロジェクト見える化部会」として第三者の立場からプロジェクトをサポート

Interview 4

IPA SECのエンタープライズ系プロジェクトには、経済産業省の「プロジェクト見える化部会」において、プロジェクトの失敗に結びつくようなリスクを見えるようにしようとする活動がある。その「プロジェクト見える化部会」での成果をCOSEのプロジェクトに適用して、第三者の立場からプロジェクトをサポートする取組みが行われた。その活動について、「プロジェクト見える化部会」主査の長岡氏に話を聞いた。

独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
エンタプライズ系プロジェクト研究員
長岡良蔵氏

プロジェクトの問題点を可視化してPMをサポート

── 長岡さんのご担当と、COSEとの関わりについてお聞かせください。

経済産業省に「プロジェクト見える化部会」(見える化部会)が設立されたのは2年前。私が、主査を務めることになりました。その目的は、PMの支援をするための可視化の実施でした。主に、プロジェクト失敗につながるリスクを見えるようにすることであり、それをCOSEプロ

ジェクトの成功のために適用する必要があったわけです。

我々の部会で1年目に対象にしたのは、プロジェクトの下流工程です。コスト、品質ならびに納期という制約条件で走っている中で、失敗するプロジェクトの不調が顕著になるのは下流になってからです。どのような予兆を察知して対処すると、品質を維持して納期に間に合わせることができるのか、が研究の主題だったわけです。

COSEの中にはもちろん、全体を見るPMがいるわけですが、会員各社もパートナー企業などに発注していることから、個々のPMが設置されています。その方々が何に悩んでいるのかを知るため、インタビューを行いました。第三者の立場で客観的にプロジェクトを見て、問題の解決を提案したいと考えました。

もちろん、見える化部会のメンバそれぞれが思いつきでインタビューしても、成果は期待できません。「誰もが一定ライン以上のレベルでインタビューでき、標準的な成果が得られるようにすべき」という意図で、我々はチェックシートを使うことを考えました。そのため、新たにPMBOKでいわれている知識エリアに、日本に根付いている文化を採り入れたエリアを独自に加えたものを開発したのです。

下流工程の検証で見えた問題点と改善例

── 下流工程を対象とした活動の成果はいかがでしたか？

実際に各社のPMなどをインタビューした結果、それぞれの方々が思い悩み、懸念しているところを解消できた部分もあります。逆にPM自身があまり不安に思っていなくても、第三者の立場でウィークポイントを見つけることもできました。「よりうまく進めるためには、こういうところをもう少し強化しなければいけませんね」と指摘できたわけです。とはいえ、それらの数は多くはありません。どちらのベンダーもプロジェクトの推進力が強く、開発作業は比較的順調に進行したと見ています。

── インタビューで問題点を発見して、改善できたことの具体例をご紹介ください。

まず、体制面があります。各ベンダーは、開発チームを組むに当たって「ここは3人でいいだろう」「メンバの技術レベルはこの程度で大丈夫」などと判断するための経験値を持っています。ただ我々は、経済産業省の代表としてより多くのプロジェクトを見てきているわけです。その経験から別の視点からの見解をお伝えすることもできました。

たとえば、人月当たりでどれくらいの生産量を見込んで仕事をしているかが、各ベンダーに

よって大きな開きがありました。調べてみると「非常に低い生産性を明記しているところは、逆に利益率を高く持ちすぎている」ことが推測できました。

加えて、プログラムを試験するためのチェック項目の指標が、各社でばらついていることもわかりました。ある企業では1000単位当たりでテスト項目が60用意されているのに、30しか用意していない企業もある。そこで調べてみたところ、チェック項目の記述レベルに差があったのです。60項目の企業では条件がマトリックスになっていて、たとえば「ある入力に対して、この出力が得られなくてはならない」というように、明快に書いてある。つまり、記述項目が細かいので、項目数が多くなっていたわけです。

一方、30項目の方は記述が非常にアバウト。「ある条件の時に、こうなればいい」程度しか書かれていませんでした。それではプログラマがアルゴリズムを作る際の「自由度」が高すぎて、テストの結果が本当に正しいかどうかの検証につながらないケースがあります。そうした「弱点」を見いだすこともできました。

上流工程で見えた問題点

── 2年目における部会の活動をご紹介ください

1年目とは一転して、対象にしたのは上流工程です。プロジェクトの入口となる工程ですから、ここで間違えたまま下流に行くと、必ず品質に悪影響が出ます。そのため、上流におけるリスクを洗い出すためのチェックシートを新たに作成し、再びPMなどのインタビューを実施しました。

── 上流でも下流のような各ベンダーの違いはありましたか？

上流工程では機能やプログラム構造などの設計がメインになるわけですが、アウトプットとして出されたドキュメントの記述レベルに差がありました。やはり、いいものを作り上げてくるベンダーは記述レベルが細かい、という印象です。代表的な例を挙げると、ここでも書き方がマトリックスになっている。要するに、プログラマにあまり考える余地を与えないレベルの記述になっているわけです。ですから、誤りも明快に見つかりますし、誤りがないと判断すれば、生産性を非常に上げることができるような書き方になっています。

一方、ドキュメントの記述が荒いベンダーもありました。「ここは、わかっている人が担当するので大丈夫」というような説明を受けるのですが、やはりプログラムの詳細設計を担当する人間から見ると、あいまいな部分があまりにも多いと思いました。そのため、機能設計の途

中で裏を取る必要が出てきてしまい、結果として二度手間になってしまいます。

もちろん「生産性、品質を上げるためによく考えている」ことと、ドキュメントの記述レベルが、100%リンクしているとまでは思えない部分もあります。しかし、記述が細かいところの方が、同じ間違いをするにしても、すぐに気づくことができるような作り方をしていると感じました。

この作業を通じて、記述レベルの最低ラインを提案できたのは、成果の1つだと考えています。

見える化部会の「成果」を今後にどう生かすか

── 見える化部会によるインタビューの成果などについて、COSEにはどうフィードバックしたのでしょうか？

まず、我々の中で一定の評価基準を決めて、レーダーチャートなどを使った報告書をまとめました。そこには、COSEの各担当者自身が評価した結果と、我々の評価の差分も作ってあります。その報告書をもとに各ディビジョン、ベンダーのPMならびに担当の方に「良かったのか、悪かったのか」インタビューさせていただき、同時にディスカッションしています。その際、他社のデータをそのまま見せることはできないのですが、ニュアンスは伝わり、ご理解いただけたようです。

実際、皆さんの評価は、おおむね好意的だということがわかりました。我々の評価がPMの理解と変わらなかったことから「やはり第三者が見てもそう感じるのですね。自分の考えが間違いではないし、自分の認識は確かだ」と確認できたことで、PMの心の支えになったのかな、と考えています。

── そのほかに「第三者」ならではの成果はありましたか？

やはり直接統括しているPMが行くと、そこには利害対立があり、言いたいことを言えないことがあります。そうした、普段表には出にくいような部分を浮き彫りにできたのが、我々が活動したメリットと考えていることの1つです。それから、ベンダーごとに持つ強み、弱みをある程度明確にすることができました。そうすると、プロジェクトのチーム力を増強する、標準化するなどの改善策を導き出すことが可能になります。

── 今回、積み残したことなどはありましたか？

今回の活動で、悪かったことやネガティブなことはあまりなかったと思っています。ただ、

プロジェクトに対して常に「標準」が通用するわけではありません。PMは物事や出来事に応じてダイナミックに動くべき時があります。もし、人的、時間的余裕があれば、各ベンダーの現場まで入り込んで、その事例研究をしてみたかったという思いはあります。

── 今回の見える化部会の大きなミッションの1つはPMのサポートだったわけですが、今後優秀なPMが出てくるための条件は何でしょうか？

PMは自分の仕事、判断に対する自信があれば育つと思います。その過程ではクライシスも必要でしょう。ただ、世の中には一度も失敗していないPMも存在します。なぜかを聞くと、プロジェクトの中盤などにいくつかのポイントを設けて、相談、報告をする機会を与えてもらえていることがわかりました。そこで、たとえば前提条件が変わったためにコストオーバーしそういう時、会社のスポンサーにコミットしてもらえたことが有効だったようです。優秀なスポンサーシップがないところに、優秀なPMは育たないと思っています。

── 最後に、いろいろなプロジェクトを見てきたお立場で、COSEの評価をお聞かせください。

COSEは新しい技術に挑戦したわけですが、世の中の初物プロジェクトのほとんどが失敗しています。しかしCOSEでは、そのような心配はあまりせずに推進することができました。それは参加したベンダーの基礎技術力の高さがあったからだと思います。その応用という面でも、非常に強い力を示していただきました。

3-4 エンピリカルソフトウェアエンジニアリング

Interview 5

協調フィルタリングによって適切な結果を得るための糸口が見えてきた

協調フィルタリングをソフトウェア開発プロジェクトへ適用し、類似した過去のプロジェクトを参照することで、進行中のプロジェクトの工数見積につなげることができる。当然、この技術には過去のプロジェクトで収集したデータが大きな意味を持つことになる。そこで活用されたのは、IPA SECが収集した千数百件におよぶプロジェクトのデータだった。奈良先端大と共同でプロジェクトデータ分析に取り組んだ菊地氏にお話をうかがった。

独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
エンタプライズ系プロジェクト研究員
菊地奈穂美氏

協調フィルタリングを使った工数見積もりの実証実験を担当

── 菊地さんが今回のCOSEの実証実験にはどう関わったのか、お聞かせください。

COSEは7社を組合員とする研究組合で、その中で実際の開発を担当したのは6社。SECはCOSEの開発プロジェクトにいくつかの技術で携わっていますが、その中には技術の有用性を評価するといった、実際の開発での直接的な関わりが少ないものもあります。

私はIPA SECのエンタプライズ系タスクフォース研究員の立場で参加し、奈良先端大学との共同でSEC収集データとCOSEのプロジェクトデータを使って分析に取り組みました。奈良先端大松本研究室の大杉直樹特任助手と松本研究室の方々が開発している協調フィルタリングのツールを使い、工数の見積もりを分析のゴールとして実証を行ったのです。

この種の分析はデータだけを大学に渡して依頼することもできます。しかし今までの経験からいえば、大学側だけでの作業ではいい結果が出ない場合も多いことを感じています。研究成果として論文を書くだけならいいのですが、それが開発現場で使えるとは限りません。そこで、現場を知る人間が分析結果を解釈して「これは使えそう」とか「もう少しチューニングをした方がいい」「これだけではわからないので、補足の情報を入れた方がいい」などと判断することが重要です。

今回の協調フィルタリングを使った分析実験では、初期における細かい分析のテクニックは奈良先端大の大杉さんたちにお願いして、私はパラメータデータの項目の選定や分析結果の解釈について共同作業を行いました。

── 協調フィルタリングについてのご説明と、今回のCOSEプロジェクトとの関わりについてお話しください。

協調フィルタリングは、データの中の似ている事例から、予測対象を類推する手法です。適用例で一番有名なのは、顧客の購入履歴からお薦め商品を導き出す書籍販売サイトの事例でしょう。

今回、19社からSECに提供された約1400件のプロジェクトデータを使って、COSEの開発プロジェクトの工数見積もりの予測値を導き出しました。SECのデータには項目が500近くあり、内訳は開発の規模、開発に係わった工数、期間などの基本情報に加えて、プロファイル情報としてアーキテクチャー、開発言語、開発メンバのスキル、外注比率など。統計分析に当たっては500項目の中で、重要なもの、データを得やすいものを選択して100程度に絞るのですが、それでもすべての項目を埋めることは困難です。しかも、プロジェクトデータごとに、空いている項目の種類が違います。だからといって、データ提供各社にヒヤリングもしきれないので、データの欠損状況がまだらになっています。

一方、何らかのデータセットを統計分析する場合、一般的な手法としては重回帰分析がありますが、データ項目が全部揃っているのが前提になっています。そのため、データに空欄があるままでは適切な結果が出るとは期待しにくいのです。そこで、欠損データがあっても類似度を計算するロジックがある協調フィルタリングをCOSEのプロジェクトに適用することに

よって、その予測精度の検証や、実用化に向けての模索を行うことにしたのです。

作業は、奈良先端大の大杉さんや柿元健さん、角田雅照さんたちが開発した協調フィルタリング実装ツールを使って行いました。1年目の2005年度はその時点でSECにあったプロジェクトのデータ約1000件を使い、作業はほとんど全部、奈良先端大の方でやっていただきました。そのような大量のデータを使った分析というのは、奈良先端大の方でも初めてだったと思います。翌2006年度の対象データは約1400件で、ツールをこちらにもいただいて私の方でも作業してみました。

選択したターゲットは開発の工数ですから、パラメーターとなるのはその工数以外。たとえば開発の規模で、計画時点で想定されていたプログラムのサイズなど。そしてデータをツールに入力して実行すると、類似のプロジェクトのトップ5などが抽出され、それをベースにターゲットのCOSEプロジェクト工数の見積もり値が導き出されるのです。

有意義だった奈良先端大との共同作業

――ご自分でも協調フィルタリングツールを使って作業してみていかがでしたか。

良かったと感じたことが、いくつかあります。1つは協調フィルタリングをツールの形で実装していただいているので、SECで用意しているデータセットの項目を用意すれば、予測させる作業が非常に短時間でできる点です。中のアルゴリズムは全部ブラックボックスになっていますが、ツールの基本的な意味さえわかっていれば、うまく使えました。加えて、ツールのアウトプットの見せ方も工夫されているのも、利用者からみてありがたかった点です。

また、ツールのアルゴリズム、ロジックのチューニングを、実証をしながら断続的にしていただけたことも有益でした。実際、ほぼ半年ごとにツールのパフォーマンスが向上しています。これはSECが独自に行っていてはできなかったことです。今回の検証実験では、アルゴリズムの研究者、ツールの開発者と、ツールに入れるデータの解釈や結果の分析を行うSECの研究者との組み合わせがうまくいったと思います。

さらにもう1つ良かったと思っているところ。それは、最良であろう予測結果を得るための「教訓」のようなものがわかってきたことです。

ポイントは、分析に使うデータの選択です。プロジェクトの初期では約20項目、プログラミングに入るとさらに数十項目増え、さらに進むともう少し増加、という具合に切っていくと、結果のばらつきが本当に減るのです。どのデータを選択してカットするかの判断は、ソフト

ウェア開発プロジェクトの経験が一定レベル以上あれば、常識ベースでできると思っています。いわゆる「あうんの呼吸が必要」、というほどではありませんでした。

データを削ることは、処理時間の面でも重要でした。当初、かなり多くのデータ量を持つプロジェクトを分析にかけたら、まる1日以上結果が戻らないことがありました。そこで、とても欠損率が高く、結果として分析で活用されそうにないデータを削るようにしたら、1～2時間で結果が出ました。

適切な分析結果を得るための「糸口」とは

――データを絞って分析した結果を見た感想と、精度を高めるための課題等についてご意見をお聞かせください。

分析結果を見ると実際の工数にかなり近いものがある一方で、大きく外れるものもあり、その理由を今回探ってみました。その結果、見積もり対象のCOSEプロジェクトと、分析に使った過去プロジェクトが持っているプロファイルとがかけ離れている場合、出てきた結果があまり使えないことが確認できました。

SECにあるプロジェクトのデータには、主要な規模と工数などの分類項目があります。その分布のゾーンに、これから見積もりをしようとしているプロジェクトの計画時の点がそれなりに入っていれば、近いデータがある可能性があります。その場合、両者の特徴が大きく外れているわけではありませんので、見積もりの結果も、そう外れない期待ができるわけです。

ただ、分布が似ているだけでは駄目です。たとえばアーキテクチャーがインターネット系のものをやろうとしているのに、分布図で選んで出てきた結果がスタンドアローンのものしかなかったら、残念ながらそれでは使えません。他を探すことになります。そうした判断材料に使えるのです。

データを地道に、淡々と見ていると、他にもわかってくることがあります。たとえば開発言語に関係した事例がありました。SECの分類では、開発言語を第一と第二に分けています。今回の分析では第一だけを使ったのですが、C#を第一言語にしていたものがあって、予測結果が悪かったのです。そもそもSECにあるデータはC++などが多く、C#は非常に少ないのです。

そこで、単に当たらないと捨て去るのではなく、C#とC++では開発規模、ソースコードの行数などが変わることに着目して改善策を導くなど、前向きな対応も考えられるわけです。

協調フィルタリングという技術の大きな特徴として、データ項目の「重み」を考慮しないことが挙げられます。そのためにどんな項目でも使えて、似ている項目を選べるともいえます。それは強みになっている一方で、場合によっては弱みにもなりうるところです。

今回の検証作業でも「協調フィルタリングのロジックにおいて」ということではありますが、必ず、見積もりを出すプロジェクトに似たもののトップ5などを出してくれる。これは今まで、人間がやろうとしてもあまり上手くできなかったことですから、価値があると思います。

そして、その結果を見て「似ているからある程度活用しよう」「あまりにもかけ離れているものばかりなので、他のものからもう1回探させてみよう」などの判断は、人間が行います。今回、協調フィルタリングによって工数を見積もる際、適切な結果を得るための拠り所、糸口が見えてきたと思っています。このように上手く使うための知恵を貯めていくことが、これから生きてくるのではないかと考えています。

協調フィルタリングを使いこなすための「知恵」をどう蓄積し、伝えるか

── その貯めた協調フィルタリングを使うための知恵を伝えるのはどのようにすればいいとお考えでしょうか。

確かに知恵の伝承は、今後の課題です。直接会って伝える機会は限られますので、きちんと前提や事例などを沿えた実証型の学術論文やレポートを書いて、業界内で発表しておくのも1つの方法です。また現在、SECに集まっているプロジェクトのデータベースを、インターネットなどでインタラクティブに参照できるようにする企画が検討されています。そこで協調フィルタリングを使って「似ているプロジェクトはこれ」と出てくるようにすることは可能ですし、検討の価値はあると思います。

やはり定期的にカンファレンスなどを開催して、各企業の現場で適応や改良した結果などを持ち寄って発表できると、知恵を集約して残していくことができます。それは1社だけでも、大学だけでもできませんので、IPA SECなどが束ねて継続的に行っていく事業の中で実現できるといいと思っています。

対外発表と学術的な成果

A. 論文誌、国際会議等での発表

COSEプロジェクトの中で実証したソフトウェアエンジニアリング手法に関する成果が学術論文や国際会議などを通して積極的に情報発信された。また、大学に登録されたテクニカルレポートが6編ある。

1. Yoshiki Mitani, Tomoko Matsumura, Mike Barker, Seishiro Tsuruho, Katsuro Inoue, Ken-Ichi Matsumoto: Proposal of a Complete Life Cycle In-Process Measurement Model Based on Evaluation of an In-Process Measurement Experiment Using a Standardized Requirement Definition Process, ESEM2007（Sep.2007）pp.11-20
2. Shuji Morisaki, Tomoko Matsumura, Kimiharu Ohkura, Kyohei Fushida, Shinji Kawaguchi, Hajimu Iida: Fine-grained Software Process Analysis to Ongoing Distributed Software Development, In Proc. Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects (Aug.2007) pp.26-30
3. 松村知子、森崎修司、玉田春昭、大杉直樹、門田暁人、松本健一：GQMモデルに基づく設計工程完成度計測手法の提案、奈良先端科学技術大学院大学、テクニカルレポート(2007-3) P.22
4. 松村知子、森崎修司、玉田春昭、大杉直樹、門田暁人、松本健一：プロセス改善を目的とするODCを用いた欠陥修正工数分析、奈良先端科学技術大学院大学、テクニカルレポート(2007-3) P.11
5. 松村知子、森崎修司、玉田春昭、大杉直樹、門田暁人、松本健一：設計書の再利用を考慮したレビュー効率の比較方法の提案と事例紹介、奈良先端科学技術大学院大学、テクニカルレポート(2007-3) P.16
6. 松村知子、門田暁人、森崎修司、松本健一：マルチベンダー情報システム開発における障害修正工数の要因分析、情報処理学会論文誌、2007-5
7. Shuji Morisaki, Akito Monden, Tomoko Matsumura, Haruaki Tamada and Ken-ichi Matsumoto: Defect Data Analysis Based on Extended Association Rule Mining, In Proc. International Workshop on Mining Software Repository (May.2007) pp.17-24
8. 肥後芳樹、吉田則裕、楠本真二、井上克郎：産学連携に基づいたコードクローン可視化手法の改良と実装、情報処理学会論文誌, Vol.48, No.2, pp.811-822 (2007-2)
9. 森崎修司、松村知子、玉田春昭、大杉直樹、門田暁人、松本健一：相関ルール分析を用いた障害対応データの特徴分析、奈良先端科学技術大学院大学、テクニカルレポート(2007-2) P.15
10. 玉田春昭、松村知子、森崎修司、松本健一：プロジェクト遅延リスク検出を目的とするソフトウェア開発プロセス可視化ツールProStar、奈良先端科学技術大学院大学、テクニカルレポート(2007-2) P.11

11. 松村知子、勝又敏次、森崎修司、玉田春昭、大杉直樹、門田暁人、楠本真二、松本健一：自動データ収集・可視化ツールを用いたリアルタイムフィードバックシステムの構築と試行、奈良先端科学技術大学院大学、テクニカルレポート(2007-2) P.23
12. 神谷芳樹：ソフトウェア計測の国際会議MENSURA2006に参加して（海外レポート）：SEC journal no.9 pp.18-19 (2007)
13. Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Naoki Ohsugi, Akito Monden, Yoshiki Higo, Katsuro Inoue, Mike Barker, and Ken-ichi Matsumoto:A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering, In-Process Measurementsユ and a Benchmarking ;International Conference on Software Process and Product Measurement: MENSURA 2006, Cadiz Spain(November 2006）pp.98-107
14. 森崎修司、松村知子、大蔵君治、伏田享平、川口真司、飯田　元：エンピリカルデータを対象としたマイクロプロセス分析、情報処理学会研究報告、ソフトウェア工学研究会、Vol.2006, No.125, 2006-SE-154, (Nov.2006) pp.9-15,
15. 神谷芳樹、菊地奈穂美、松村知子、大杉直樹、門田暁人、肥後芳樹、井上克郎、松本健一：進行中のプロジェクト計測とフィードバック実験に基づく計測データベース活用方式の提案
ソフトウェアエンジニアリングシンポジウム2006(ソフトウェアエンジニアリング最前線2006、近代科学社）(2006-10) pp.35-42
16. 神谷芳樹：ソフトウェア工学国際会議　ICSE2006　上海に参加して（海外レポート）
SEC journal no.7（2006-9）pp.22-29
17. Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Naoki Ohsugi, Akito Monden, Yoshiki Higo, Katsuro Inoue, Mike Barker, and Ken-ichi Matsumoto:A Proposed Method for Building a Database of Project Measurements and Applying it Using Collaborative Filtering:
5th ACM-IEEE International Symposium on Empirical Software Engineering 2006 (ISESE2006), Vol.2, Short papers：Rio de Janeiro (2006-10) pp.15-17
18. Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Yoshiki Higo, Katsuro Inoue, Mike Barker, Ken-ichi Matsumoto: Effect of Software Industry Structure on a Research Framework for Empirical Software Engineering:
28th International Conference on Software Engineering (ICSE2006), Far East Experience Track, Poster Session:Shanghai(2006-5) pp.616-619
19. 神谷芳樹、マイク・バーカー、松本健一、鳥居宏次、井上克郎、鶴保征城：現場データを産学で共有するソフトウェア工学研究のための枠組み：産学連携学、Vol.2.No.2　2006-3, 産学連携学会、pp.26-37
20. 松浦　清、神谷芳樹、樋口　登：Project Report 先進ソフトウェア開発プロジェクト　Part II；SEC journal no.5, 2006-2, pp44-49
21. Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Mike Barker, Ken-ichi Matsumoto: An empirical trial of multi dimensional in process measurement and feedback on a governmental multi-vendor software project:
International Symposium on Empirical Software Engineering (ISESE2005)vol.2 pp5-8; Noosa

Heads, Australia, 2005-11-18

22. Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Mike Barker, Ken-ichi Matsumoto: A Research Framework for Empirical Software Engineering Collaboration and Its Application in a Software Development Project:
International Workshop on Future Software Engineering (IWFST2005):Shanghai, 2005-11-8
23. 樋口　登：Project Report 先進ソフトウェア開発プロジェクト；SEC journal no.2、2005-4 pp.56-57

B. 商業誌等での掲載

商業誌等においても、COSEにおけるソフトウェア開発について紹介された。

1. 大杉直樹、松村知子、森崎修司：ソフトウェア開発の「見える化」を支援するデータ分析力、～エンピリカルアプローチによる既存データの有効活用～：JISA会報　№80　2006年1月
2. 取材記事：開発状況の定量化手法、SECが実地検討；狙いは中小ベンダーへの普及だが課題も残る、（森側真一）：日経コンピュータ　；2005-9-19

C. EPMツール改良版の配布

IPAはEPMをもとに、プロジェクト計測ツールの商用製品水準への改良版を実用化し、その配布事業を開始した。また、これを用いた実際のプロジェクトによる検証作業（実証実験）を開始した。EPMについては、COSEでの実践結果を反映し、導入の容易さ、分析手法の追加が行われた。配布ツールには、コードクローン分析ツールのCCFinderXと協調フィルタリングツールのMagi試供版が同梱され、先進プロジェクトで実施したプロジェクト計測環境を再現できるように工夫されている。

D. 出版物の発行

COSEにおけるソフトウェアエンジニアリングの実践に関連したいくつかの書籍が発行された。

1. エンピリカルソフトウェアエンジニアリングの勧め：IPA／SEC　翔泳社　2007-10
2. ITプロジェクトの「見える化」、上流工程編：IPA／SEC　日経BP社　2007-5
3. Tプロジェクトの「見える化」、下流工程編：IPA／SEC　日経BP社　2006-6
4. 「柔の力、剛の技」：IPA、アスキー出版、2006-5

E. 講演会、公開デモ等による普及活動

講演会や展示会等でもCOSEにおけるソフトウェエアエンジニアリングの実践について紹介された。

1. 樋口　登：SEC Forum 2007（IPAX2007講演）（東京ドームシティ）、2007-7-20 「エンピリカルソフトウェアエンジニアリングの実践」
2. 神谷芳樹：エンピルカルソフトウェア工学研究会(2007年度第2回）（東京、CIC）、2007-7-9 「IPAのEPM検証計画の紹介」
3. 森崎修司：エンピリカルソフトウェア工学研究会：（東京、CIC）、2007-7-9「EPM Pro*によるリスク検出例」
4. 玉田春昭:エンピリカルソフトウェア工学研究会:（東京、CIC）、2007-7-9「EPM Pro*アーキテクチャと活用例の紹介」
5. 井上克郎：組込み技術展関西コミュニティセッション：（大阪、マイドーム大阪）、2007-6-6「エンピリカルソフトウェア工学とEASEプロジェクト」
6. 森崎修司:組込み技術展関西コミュニティセッション:（大阪、マイドーム大阪）、2007-6-6「プロジェクトモニタリングの事例と蓄積データを用いた見積り／傾向分析」
7. 玉田春昭：組込み技術展関西コミュニティセッション：（大阪、マイドーム大阪）、2007-6-6「EASEプロジェクトの分析ツールと分析手法」
8. 樋口　登：ソフトウェア開発環境展（SODEC）（東京ビッグサイト）、2007-5-18 「EPMツールの紹介と実証プロジェクト募集案内」
9. 樋口　登：ソフトウェア開発環境展（SODEC）（東京ビッグサイト）、2007-5-16 「ITプロジェクトの『見える化』上流工程編　概説」
10. 樋口　登：Where2.0時代におけるリアルタイムプローブ情報の活用（COSE公開デモ講演）（青山）、2007-3-23 「ソフトウェアエンジニアリングの実践」
11. 塚本　晃：Where2.0時代におけるリアルタイムプローブ情報の活用（COSE公開デモ講演）（青山）、2007-3-23 「プローブ情報プラットフォームソフトウェアの実証実験」
12. 勝又敏次：Where2.0時代におけるリアルタイムプローブ情報の活用（COSE公開デモ講演）（青山）、2007-3-23 「プローブ情報プラットフォームソフトウェアの開発」
13 井上克郎: Where2.0時代におけるリアルタイムプローブ情報の活用（青山）,2007-2-22「ソフトウェアエンジニアリングの実証的アプローチ」
14. 鳥居宏次：e-Society シンポジウム2006、（本郷）、2006-12-19, pp.77-88. 「データ収集に基づくソフトウェア開発支援システム」
15. EASE:e-Societyシンポジウム併設展示（本郷）：2006-12-19　COSE－EASE連携プロジェクト紹介ビデオ
16. 神谷芳樹：エンピリカルソフトウェア工学研究会第11回（2006年度第2回）：（品川）、2006-12-18「スペインMENSURA2006参加報告」
17. 神谷芳樹：エンピルカルソフトウェア工学研究会(2006年度第1回）（東京、CIC）、2006-10-16「SECでのEPM実証実験（SEC先進プロジェクト）とIPAによるEPM普及施策の状況」
18. 神谷芳樹：エンピルカルソフトウェア工学研究会(2006年度第1回）（東京、CIC）、2006-10-16「ISESE2006参加報告」
19. 神谷芳樹：情報化月間特別行事、SECセッション（東京、全日空ホテル）；2006-10-2「先進プロジェクト報告」

20. Katsuro Inoue: "Code Clone Analysis and Application", Seminar at National University of Singapore, 2006-9-8
21. Katsuro Inoue: "Software Engineering Researches in Osaka University", Invited Talk at India Institute of Technology, Delhi, 2006-8-31
22. SEC：ソフトウェア開発環境展（SODEC）2006出展参加（東京、有明）：2006-6-28～30
23. Katsuro Inoue: "Code Clone Analysis and Application", State of the Art in Software Engineering 2006, at Rutgers University, 2006-6-16
24. 松浦　清：SEC Forum 2006, (サンケイプラザ)、2006-6-12. 予稿pp.84-96
「SEC先進プロジェクト」
25. Katsuro Inoue: "Software Engineering Researches in Osaka University", Seminar on Information Science: Progress in Research on Information Science and Technology in Osaka University, Japan, May 30, 2006, Peking University, 2006-5-30
26. 松村知子：エンピリカルソフトウェア工学研究会（2005年度 第4回）(東京、CIC)，2006-3-16.
「障害修正工数に関する要因分析事例報告」
27. 肥後芳樹: JASPIC 例会, 2006-3-16
「コードクローン検出技術とその利用法」
28. 松浦　清：沖電気工業社内講演会（蕨）：2006-7-11
「プロジェクト計測プラットフォームEPM」
29. 松村知子，ウィンターワークショップ2006・イン・鴨川，Vol.2006，pp.61-62，2006-1-26.
「組織横断的データ収集・分析技術の事例研究」
30. 神谷芳樹：エンピリカルソフトウェア工学研究会:（東京、CIC)、2005-12-22
「上海で開催されたIWFST2005等への参加報告」
31. 神谷芳樹：SECエンタープライズ系総合部会（東京、NRI)、2005-10-14
「SEC先進プロジェクト報告」
32. 神谷芳樹：情報化月間記念特別行事SECセッション（東京：全日空ホテル)、2005-10-3
「産学官連携プロジェクトにおける定量データ収集・分析環境導入の実現へ「実証プロジェクト＋可視化＋定量データ　報告」」
33. 松村知子：JISA経営者セミナー（東京:科学未来館）：2005-9-31
「ソフトウェア開発での科学的マネジメントの勧め」
34. 神谷芳樹：エンピリカルソフトウェア工学研究会:（東京、CIC)、2005-6-23
「先進ソフトウェア開発プロジェクトについて」
35. 松村知子：エンピリカルソフトウェア工学研究会（2005年度 第1回）(東京、CIC)，2005-6-23.
「先進ソフト開発プロジェクト報告」,
36. 近藤弘志：IPAX2005：特別講演：(東京、有明)、2005-5-18
「次世代ITSのソフトウェアアーキテクチャーとプラットフォーム」

F. 参考Webサイト

SEC関連の講演資料、印刷物の多くがSECのWebサイトからダウンロードできます。
http://sec.ipa.go.jp/

EASE関係の発表資料の多くがEASEプロジェクトのWebサイトからダウンロードできます。
http://www.empirical.jp/

奈良先端科学技術大学院大学のテクニカルレポート（Postscript）は、情報科学研究科のWeb サイトからダウンロードできます。
http://isw3.naist.jp/IS/TechReport/

おわりに

2年余にわたるSECの先進プロジェクトは、計測とデータを基盤とする実証的なソフトウェアエンジニアリングの分野に大きな足跡を残した。COSEは活動を終え解散したが、参加した各企業はここでの体験をもとに、それぞれ自らの資源を投入して新たなソフトウェアエンジニアリング施策への歩を進めた。用いられたツールと手法は、SECの手でブラッシュアップされ、普及へ向けて「EPMツール検証プロジェクト」として次の段階へ進められた。この検証プロジェクトには、本書執筆時に30社の参加があり、参加企業はCOSEプロジェクトの事例を背景に着実に増加しつつある。

また、連携したEASEプロジェクトの産業界からの中核メンバであるNTTソフトウェアでは、EPMの全社適用を開始し、その状況をEASEプロジェクトの研究会で発表した(注)。

EASEプロジェクトやSECの研究員からは、多数の学術論文、テクニカルレポートが発表された。収集、蓄積されたプロジェクトデータは保存され、ソフトウェア工学の学術研究に供される。開発されたプローブ情報システムについては、次の事業化計画が検討されている。

謝辞

本プロジェクトの推進に絶大なご支援、ご指導をいただいた、ソフトウェアエンジニアリング技術研究組合および組合員各企業、EASEプロジェクトの奈良先端科学技術大学院大学、大阪大学大学院、参加各企業、そして経済産業省の各位に心より感謝の意を表します。

2007年10月
独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター

注：NTTソフトウェア
NTTソフトウェア（株）におけるEPMの適用について、エンピリカルソフトウェア工学研究会、2007.7.9
発表資料はIPA SECのWebサイトから参照可能である。http://sec.ipa.go.jp/tool/EPMcase_nttsoftware.pdf

編著者紹介

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター

2004年10月に独立行政法人 情報処理推進機構（IPA）内に設立されたソフトウェア・エンジニアリング・センター（SEC）は、エンタプライズ系ソフトウェアと組込みソフトウェアの開発力強化に取り組むとともに、その成果を実践・検証するための実践ソフトウェア開発プロジェクトを産学官の枠組みを越えて展開している。

【所在地】 〒113-6591 東京都文京区本駒込2-28-8
文京グリーンコート　センターオフィス
電話 03-5978-7543　FAX 03-5978-7517
http://sec.ipa.go.jp/index.php

執筆者

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
研究員　樋口 登
研究員　神谷 芳樹

ソフトウェアエンジニアリングの実践
～先進ソフトウェア開発プロジェクトの記録～

2007 年 11 月 27 日　初版第 2 刷発行

編著　独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
発行人　佐々木幹夫
発行所　株式会社翔泳社（http://www.shoeisha.co.jp/）
印刷・製本　凸版印刷株式会社

ISBN978-4-7981-1408-8　◎ Printed in Japan