エンピリカルソフトウェア工学と ＥＡＳＥプロジェクト

奈良先端科学技術大学院大学 情報科学研究科 教授
松本　健一

１．はじめに

近年、ソフトウェアやその開発過程から得られる定量的データに基づいてソフトウェアの生産性や品質の向上を目指す実証的アプローチ（エンピリカルアプローチ）が注目されている［4］。経済産業省 産業構造審議会 情報経済分科会 情報サービス・ソフトウェア小委員会が2006年6月13日に公開した資料［3］でも、「情報サービス・ソフトウェア産業の変革の必要性を強調するとともに、変革後のあるべき姿を提示するため」として、エンピリカルアプローチの必要性、重要性を次のように説いている。

- 企業情報システムや組込み系のソフトウェアの信頼性・生産性の向上という観点からは、プロジェクトの計画、実装、試験、運用段階における定量的な運営手法等のソフトウェアエンジニアリングに関する実証的な研究を産学官が連携して進めるとともに、その普及に努めることが重要である。
- ソフトウェア開発の信頼性と生産性を高める新しい手法の現場への導入を促進するためには、理論やガイドラインのみでは十分ではなく、実際の現場へ適用して成功例を示すことや、現場からのフィードバックを得て、さらに理論等を改善していくことが重要である。

本稿では、組込みソフトウェア開発でも今後注目されるであろうエンピリカルアプローチの基本的な考え方を解説するとともに、その実践と普及をめざし活動中の文部科学省EASEプロジェクトの概要を紹介する。

２．組込みソフトウェア開発の現状

経済産業省の「2006年版組込みソフトウェア産業実態調査報告書」［2］によると、わが国における組込みシステム生産高は約59兆円、国内総生産比率で11.9％にものぼる。開発費はおよそ6兆7,700億円と推定され、その約40%にあたる2兆7,300億円が組込みソフトウェア開発費であり、組込みソフトウェア技術者数は約19万3000人に達する。2004年の調査結果と比較すると、開発費で36.5%、技術者数で28.7%も増加している。

表１　プロジェクト成功率の比較

観点＼プロジェクト種別	ソフトウェア開発全般[5]	組込みソフトウェア開発[2]	データ収集が行われているソフトウェア開発[10]
品質	46%	71%	71．5%
コスト	76%	43%	86．0%
納期	55%	36%	81．7%

なかでも、組込みシステム開発におけるソフトウェア開発の割合はますます大きくなってきている。開発費の約40%がソフトウェア開発費となっていることは、先に示したとおりであるが、開発に携わる技術者の中でソフトウェア技術者が占める割合も、2004年の47%から2006年には62%へと増加し、ハードウェア技術者を大幅に上回る状況となっている。

このように重要性が増し、社会に与える影響も大きくなりつつある組込みソフトウェアであるが、開発管理の点では、エンタープライズ系をはじめとする従来のソフトウェアには、まだまだ及ばないと表現されることがある。が、本当にそうであろうか。表１は、ソフトウェア開発プロジェクトの成功率を、品質、コスト、納期、それぞれの観点で事後評価した結果である。これによると、確かに、コストと納期の点では、組込みソフトウェア開発の成功率は低いが、品質の面では、エンタープライズ系など従来のソフトウェア開発を大きく上回っている。単純な比較は難しいかもしれないが、優劣があるというよりも、重視する点や優先順位が異なるということのようである。

３．エンピリカルソフトウェア工学

（1）概要

エンピリカルソフトウェア工学（Empirical Software Engineering）とは、ソフトウェア開発に関する実証データや実績データに基づいて、工業製品としてのソフトウェアの開発を効率よく確実に行う方法を追求する学問分野である［9］。ソフトウェアそのものやその開発過程を表すデータ（ソフトウェア開発データ）に基づいて、ソフトウェアの生産性や信頼性の向上を目指す技術の総称とも言える。

エンピリカル（empirical）という概念を日本語で端的に表現することはむずかしい。「経験主義」とすると、経験則しか認めないというニュアンスとなってしまう。翻訳のプロに意見を伺ったこともあるが、これというものはないという返事だった。ロングマン現代アメリカ英語辞典によると、

empirical: based on scientific testing or practical experience, not on ideas.

とある。これからすると、

Empirical（エンピリカル）
=Experiment（大学などでの実験）
+Experience（開発現場での経験（実績））。

といったところであろうか。便宜的には、「実証的」、「データに基づく」という表現が使われている。

（2）３つの段階

エンピリカルアプローチには、3つの段階があるといわれている［7］。

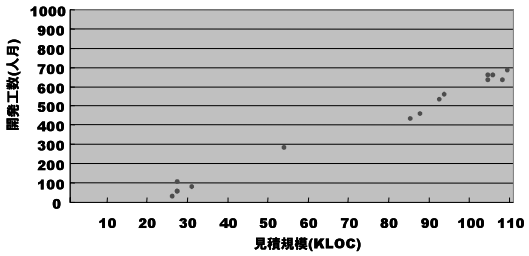
【第１段階】観察（Empirical observations）の実施
実験や調査で確認された事実が対象。
現象を表現できるようになる。

【第２段階】法則（Laws）の発見
繰り返される観察が対象。
現象が起こるコンテキストを理解し、現象を予測できるようになる。

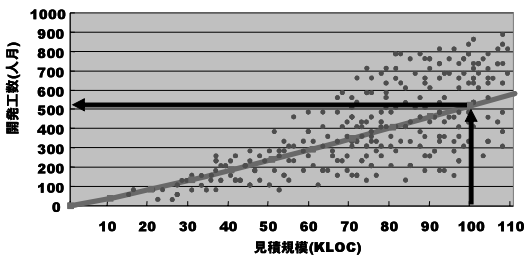
【第３段階】理論（Theories）の確立
因果関係が対象。
現象を説明できるようになる。

そして重要なことは、人間を含む数多くの要素が複雑に絡み合って構成されるソフトウェア開発では、「観察の実施」と「法則の発見」は可能であるが、「理論の確立」まで成し遂げることは、少なくとも現状では非常に難しいということである。

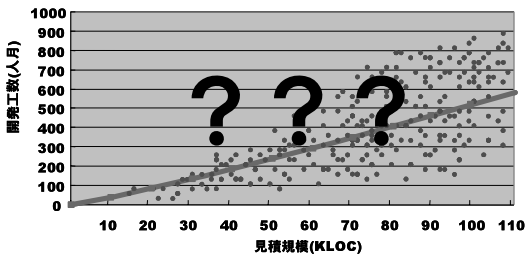
簡単な例を示す。今、自社のソフトウェア開発プロジェクトにおいて「ソフトウェアの見積規模（KLOC）」と「開発工数（人月）」を収集する場合を考える。見積規模と開発工数は実績値（事実）であり、図１（ａ）のようなグラフ上にプロットすることで現象を表現することができる。第１段階「観察の実施」にあると言える。



（ａ）観察の実施



（ｂ）法則の発見



（ｃ）理論の確立

図１　エンピリカルアプローチの３段階

次に、見積規模と開発工数の実績値が蓄積されていくと、図１（ｂ）に示すように、ばらつきがあるとはいえ見積規模が大きくなると開発工数も大きくなる傾向にあるように思えてくる。そこで、回帰分析など統計的手法を適用すると、工数見積モデルCOCOMOのように、

開発工数　＝　3.0×（見積規模）$^{1.12}$

といった式によって両者の関係を表すことができるようになる。この式を用いれば、例えば、見積規模が100KLOCのソフトウェアであれば、その開発工数はおよそ521人月になると予測できる。第２段階「法則の発見」に入ったと言える。

最後は第３段階「理論の確立」であるが、先に述べたように、この段階に進むことは容易ではない。COCOMOの関係式のようなものが示されると、理論が確立されたかのように考えがちになるが、関係式は、なぜ見積規模が大きくなると開発工数が大きくなるのか、その仕組みを説明するものではない。$y = a \times x^b$という式で二つの実績値の関係を表すとすれば、a=3.0、b=1.12とするのが統計的に最ももっともらしいというだけである。もちろん、理論の確立は非常に意義のあることであるが、はじめからあまり無理はしない方がよいかもしれない。

（3）メリットと教訓

理論の確立まで至ることは難しいが、だからといって、エンピリカルなアプローチが有用でないということではない、多様で個別性の高いソフトウェア開発プロジェクトにおいては、現象の予測ができることの開発管理上のメリットは非常に大きい。例えば、Putnamらは、定量的なソフトウェア開発データを用いることで次のような活動が可能になるとしている［16］。

- プロジェクトの工数、日程、品質を高い精度で見積る。
- 進行中のプロジェクトをコントロールする。
- 進行中のトラブルプロジェクトを再計画する。
- 組織内の全プロジェクトに対して、リソース割り当ての全体計画を作成する。
- 開発プロセスがどの程度改善されたかをモニターする。

また、ソフトウェア開発プロジェクトの状況を定量的に捉えようとする体制や活動が、間接的に、プロジェクトの成功をもたらすことがある。表１で紹介した日経コンピュータによる調査結果［5］でも、定量的なコスト管理を実施しているプロジェクトでは、納期順守率が全体平均より18.2ポイントも高くなっており、また、品質面でのプロジェクト成功率も高くなることが確かめられている。同様に、ソフトウェアエンジニアリングセンターが収集した1,419プロジェクトのデータ［10］によれば、ソフトウェア開発データが収集されているプロジェクトの成功率は、品質面で71.5%、コスト面で86.0%、納期面で81.7%と、先に表１で示した「ソフトウェア開発全般」や「組込みソフトウェア開発」と比べ、同じかより高い値となっている（表１参照）。

エンピリカルは、他の工業分野ではごく当たり前のアプローチであるが、相応の体制やコストが必要となることも確かである。活動の当初は、うまく機能しない場合もある。表２は、米国SEL（Software Engineering Laboratory）の研究者たちが、2001年にその25年の歴史を閉じるまでに得た「ソフトウェア開発におけるエンピリカルアプローチに関する13の教訓」である［6］。研究上の教訓ではあるが、ソフトウェア開発の現場にエンピリカルアプローチを導入する際にも参考になると思われる。

表２　エンピリカルアプローチに関する教訓

データ収集	教訓１：データ収集は厳密なプロセスと専門のスタッフを必要とする.
	教訓２：得られる情報の量には妥協も必要である.
	教訓５：組織の成果物，プロセス，目標のベースラインを作ることは，全ての改善プログラムにおいて非常に重要である.
	教訓６：データ測定の精度は常に疑わしい．その疑わしさにうまく対処し，また，その限界を理解しなければならない.
経営陣の積極的参加	教訓８：研究と開発の双方に共通のコミットメントを持つことは極めて重要である.
	教訓１１：より上層の経営陣の支援を得ることは，プロセス改善を継続的に成功されるために重要である.
	教訓１２：改善のためのプロセスが組織の外にあるとプロセス改善は成功しない.
	教訓１３：技術中心の組織では，任務を遂行する上でソフトウェア工学がいかに重要であるのかを認識させることは難しい.
研究指針	教訓９：ソフトウェア工学の研究と実践は共生的関係にある．双方の活動は，相互作用を通じて利益を得る.
	教訓１０：研究者と開発者が（文字通り）近い距離にいることが双方にとって良い結果を生む.
継続的スタッフ支援	教訓３：データ収集のためのスタッフトレーニングに終わりはない.
	教訓４：データ収集は重要であるが，ソフトウェアの納期はさらに重要である
	教訓７：開発者に情報を迅速にフィードバックすることの必要性と，収集データを十分に分析する時間を確保することの必要性の間には，いつでも緊張関係がある.

４．ＥＡＳＥプロジェクト

（1）概要

EASE（Empirical Approach to Software Engineering）プロジェクトは、文部科学省リーディングプロジェクト「e-Society基盤ソフトウェアの総合開発」の一環として2003年から５年計画で実施されている［1］［4］［9］。その目標は、エンピリカルソフトウェア工学の成果をソ

フトウェアプロダクトとして具体化し、産学官連携の下に、エンピリカルアプローチの普及と更なる発展を実現することにある。主な活動とこれまでの成果を、エンピリカルアプローチの３段階に対応させ以下に紹介する。

（2）観察の実施に向けて：EPMの提供

EASEプロジェクトがまず目指したのは、エンピリカルアプローチの第一段階「観察の実施」が容易に行える環境（エンピリカルソフトウェア工学環境）を開発者に提供することである。具体的には、EPM（Empirical Project Monitor）を開発し、2006年６月にはオープンソースとして公開した［1］。

EPMは、現在広く普及している開発支援フリーウェア（CVS、Mailman、GNATS 等）と連携することによって、ソフトウェア開発プロジェクトデータをリアルタイムに収集し、定量的データ分析を可能にするシステムである（図２参照）［9］。EPMを利用したソフトウェア開発では、プロジェクト管理者や開発者は、プロジェクトの進捗状況や作業状況を客観的に把握することが可能となる。プロジェクトの問題点を迅速に発見することで効果的なプロセス改善が期待できる。EPMの適用実験では、開発者に大きな作業負荷を与えることなく、対象プロジェクトの状況を分析可能であることが確認されている。

EPMは、構成管理、メーリングリスト管理、障害管理など、ソフトウェア開発において広く普及し利用されている開発支援システムからデータを収集する。プロジェクト管理者や開発者がデータ収集のために特別な作業を行う必要はない。データ収集が優先され納期が犠牲になるということもない。

データ収集のためにフリーウェアを導入することに抵抗のある企業もあるであろうが、特定の企業のツールやシステムを導入することに比べれば、導入可能な企業やプロジェクトの範囲は広い。もし自社製開発管理システムなどが既に導入されているのであれば、無理にEPMを導入せず、収集データの形式をEPMの標準データフォーマットに変換すれば、以降で述べるEASE分析技術の多くを適用することは可能である。

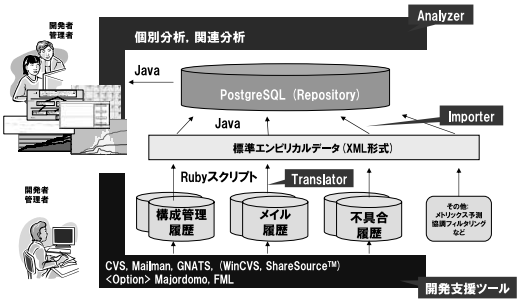
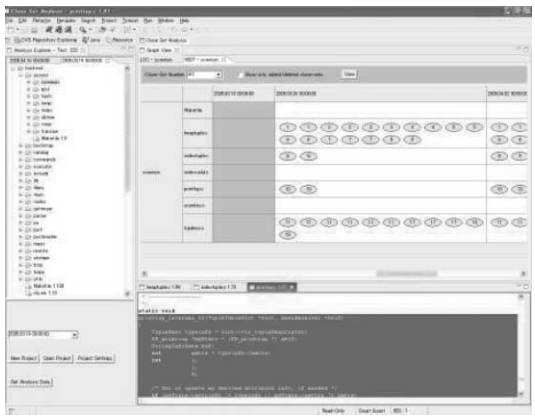
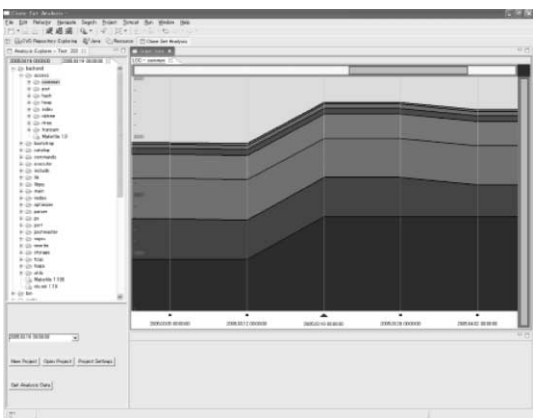


図２　EPM（Empirical Project Monitor）



（a）クローン履歴グラフ



（b）クローンLOC変化グラフ

図３　コードクローン

なお、2006年6月からは、情報処理推進機構（IPA）において、ソフトウェア開発技法普及ツール開発事業（公募）「ソフトウェア開発プロジェクト可視化ツールのパッケージ化」（EPMツール）が実施されている。これは、オープンソースとして公開されているEPMをベースとして、導入がより容易で、マニュアルやチュートリアルなども整備されたパッケージ版EPMを開

発するものである。キャッツ（株）と（株）日立システムアンドサービスにより作業が進められており、2007年1月には公開される予定である。

（3）法則の発見に向けて：分析ツールの開発

（3－1）見える化ツール

法則を発見する第一歩は、収集したデータそのものや、収集データから見てとれるソフトウェアやその開発過程の特性を図式化する、いわゆる「見える化」である。EASEプロジェクトにおける具体例を二つ示す。

・コードクローン分析

コードクローン（以下、クローンと表記）とは、ソフトウェア（プログラムコード）に含まれる重複したコード片である。クローンは、ソフトウェア開発の過程で生成されるが、その変遷を把握することで、開発の様子を明らかにすると共に、開発支援に役立てることが出来る。例えばコピー&ペーストによって誕生したクローンが、その後も頻繁に類似した変更が加えられていた場合には、プログラムを再構成してクローンを除去することで以降、同種の変更にかかるコストを削減できる可能性が高い。逆に発生以来ほとんど編集されていないようなクローンについては、クローンを除去してもあまり効果がないと考えられる。

クローンの変遷は、CVSなど構成管理システムが蓄積する開発履歴データを解析することで抽出できる。これにより、最新のプログラムコードに存在するクローンが過去の時点のどのクローンから発展してきたものかを対応づけることができ、クローンが時々刻々どのように変化していったのかを追跡することができる。

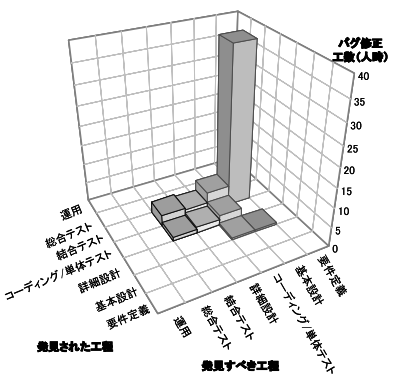


図４　直交障害分類

図３（ａ）は、クローン履歴情報をGUIとして可視化するプロトタイプシステムの画面例（クローン履歴グラフ）である［13］。本システムでは、解析を行ったそれぞれの時点において、おのおののクローンがどのファイルに含まれていたか、どの時点で変更が加わっているかを視覚的にユーザに提示することができる。検出されたクローンがそれぞれの時点においてどのような状態であったかを実際のソースコードから確認することも可能である。また、クローン量の変化を時系列に沿ったグラフとして表示することもできる（図３（ｂ）参照）。グラフから、クローンの量や増減を、ディレクトリやファイル単位で容易に管理することができ、コード劣化の様子を見ることができる。

・直交欠陥分類法

直交欠陥分類法（ODC）は、欠陥を2つの分類項目（発見工程、修正工程など）で分類したうえで、その特性値（件数、工数など）を計測することによって、対象システムにおける欠陥の特徴や傾向を把握し、効率的な品質改善を行うための分析方法である。

図４は、マルチベンダによるソフトウェア開発工程において作成されたバグ票に基づいて、バグ（欠陥）を、「発見された工程」と「発見すべき工程」で分類し、分類毎に平均修正工数を算出したものである［14］。「各ベンダが基本設計工程で発見しておくべきバグをベンダ間での結合テストまで見逃してしまうと、その修正工数が突出して大きくなる」という従来の定性的な指摘を具体的な数値で示している。バグを発見すべき工程と実際に発見した工程の不一致がバグ修正工数に及ぼす影響の大きさを把握できるだけでなく、発見すべき工程における作業の漏れやプロセスの問題点を改善することによるコスト削減の指標を得ることになる。

また、企業等で長年使用されているバグ票には、その必要性が不明な項目が少なくない。直交欠陥分類法などによる分析は、そうした、「ソフトウェアの品質や開発工数への影響の違いを示すことのできない項目」の見直しにもつながる。

（3－2）データマイニングツール

データマイニングは、データに基づいて法則（傾向、パターン、ルール、・・・）を発見するより直接的な方法である。多数のソフトウェア開発プロジェクトのデータが蓄積されている、あるいは、利用可能な場合に適用することが出来る。EASEプロジェクトにおける具体例を二つ示す。

・平行座標プロット

図５は、情報処理推進機構（IPA）のソフトウェア・エンジニアリング・センター（SEC）で収集されたソフトウェア開発データのうち、新規開発でＩＦＰＵＧ法によるファンクションポイント計測が行われた211プロジェクトのデータを平行座標プロットで表したものである[17]。この例では、「月当たりの工数が比較的大きい」プロジェクトと「月当たりの工数が比較的小さい」にそれぞれ着目し、その他の開発データ（プロジェクト特性）の値の傾向の差を図示している。特性間の厳密な関係を示しているわけではないが、月当りの工数の大小が、他の特性値をどのように変化させるか、また、させないかのおおよその傾向を把握することが出来る。

・アソシエーション分析

データマイニングの基本的な分析法の一つで、事象間の強い関係（アソシエーションルール、相関ルール）を発見することができる。ＰＯＳシステムにおける販売動向分析などがよく知られている。アソシエーションルールは

Ａ∧Ｂ　∧Ｃ　…　⇒Ｘ

という形式で表され、矢印の左側が「前提部」、右側が「結論部」と呼ばれる。前提部と結論部の間の因果関係を示すものではないが、前提部が成立する時に結論部も成立する確率などを知ることが出来る。図６は、ソフトウェア開発データから抽出されたアソシエーションルールの例である。ただし、あくまでもアソシエーションルールがどのように表現されるのかを示す例であり、実データに基づくものではない。

(4)「理論の確立」ではなく「開発支援」に向けて

・EASE協調フィルタリング見積り法

協調フィルタリングは、推薦システムなど情報検索の分野で研究されてきた手法である。これをソフトウェア開発データに応用すると、過

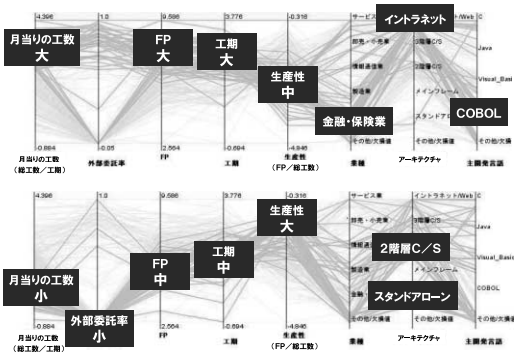


図５　平行座標プロット

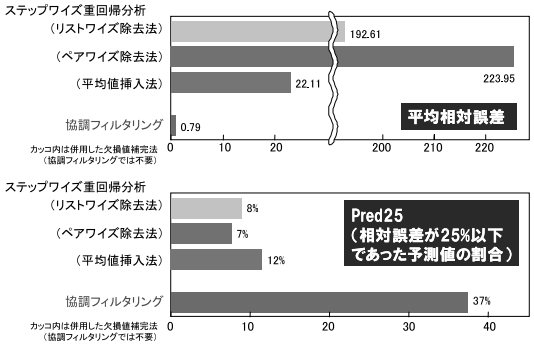
（顧客＝既存顧客）∧（業種・業務＝既存）⇒（外部委託率＝小）
（顧客＝既存顧客）∧（業種・業務＝既存）∧（業務パッケージの利用有無＝無）⇒（基本設計工数＝小）

図６　ソフトウェア開発における
アソシエーションルールの例

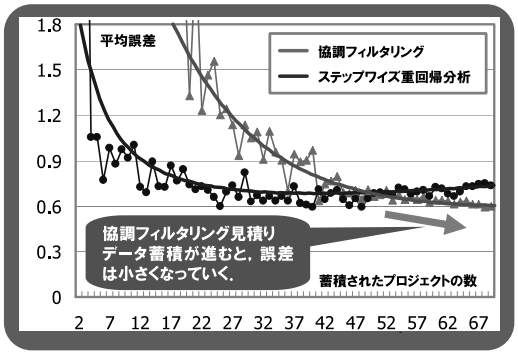
去の類似プロジェクトでの経験から新規プロジェクトの工数などを見積る、いわゆる「類推見積り」を系統的に行うことが可能となる［11］［15］［18］。重回帰分析やニューラルネットを用いた従来の方法では、すべてのソフトウェアプロジェクトに対して、ただ一つの見積り式（予測式）が構築されるが、協調フィルタリング見積り法では、見積り対象プロジェクトごとに個別に見積り式（予測式）が構築されるため、プロジェクトの個別性をより強く反映した見積りとなる。また、開発データ中の欠損値の量や分布（欠損値の偏り）が見積り精度に影響を与えにくいという特徴を持つ。

図７（ａ）は、ある企業の1,081プロジェクトで測定された14個の特性値を用いて試験工数を見積った結果である［18］。データ欠損率は約60%であった。従来の手法による見積り誤差が、協調フィルタリング法と比べて、著しく大きいことが分かる。（従来手法では、データ欠損率が30%を超えると見積り誤差が急激に大きくなるとされている）。

図７（ｂ）は、協調フィルタリング見積り法と従来の見積り法（ステップワイズ重回帰見積り法）における、見積りに利用可能な開発データ数（蓄積されたプロジェクト数）と見積り誤差の関係を示したものである［12］。従来の見積



（a）欠損率60%における見積り誤差



（b）プロジェクト数と見積り誤差

図７　EASE協調フィルタリング見積り法

り法では、データが蓄積され、利用可能なデータが増えても、見積り誤差は必ずしも小さくなっていかない。多様なプロジェクト全てを一つの見積り式（予測式）で表現しようとするためである。これに対して、協調フィルタリング見積り法では、データの蓄積に伴って、見積り誤差が小さくなる傾向にある。これは、見積り対象プロジェクトと類似のプロジェクトのデータが見積りに利用可能になる確率が高くなるためと考えられる。

・EPDG2

定量的管理を考慮してソフトウェア開発計画を立案する際には、指標や測定するデータについて理解し、プロジェクトの特性を考慮してデータの測定・分析活動を開発計画に組み込む必要がある。EPDG2（Electronic Process Data Guidebook 2）は、定量的管理に用いる管理指標とその利用に必要な測定データの理解、およびデータの測定・分析活動の調整作業を支援するシステムである（図８参照）[8]。

図８　EPDG2の実行画面

EPDG2 がプロジェクト計画者に対して提供する支援にはつぎのような効果があると考えられる。

① プロジェクトで適用する管理指標を一覧することができ、その場での取捨選択が容易に行えるため、一貫した視点に基づいて管理指標の導入が行える。

② 各工程やその各作業で必要となる測定作業の存在が視覚的に確認できるため、計測と管理に対する具体的なイメージを持ってプロジェクトに望むことができる。

③ 管理指標や測定量の概要および詳細を容易に参照できるため、指標や計測データに対する理解を深めることができ、管理や測定が正しく実践される。

④ テーラリングに関する情報を保存・収集する機能を利用して、系統的なテーラリングガイドの作成や、テーラリングパターンライブラリを構築し、経験の浅い管理者に提供することができる。

また、総合的な効果としては、EPDG2 の活用により、管理者と開発現場の両方において、定量的な管理の仕組みに対する理解がすすみ、形式主義的なデータ収集や管理体制に陥りにくくなることが期待できる。

つまり、開発現場は、なぜこのようなデータを測定して報告しなければならないのかを理解することで、データ収集に対してより協力的になると考えられる。また、管理者にとっても、自分が参照する管理指標が、実際にはどのようなデータからどのようにしてまとめられているかを知ることで、表層的ではなく、現場を意識した対応をとるようになると期待される。

5．まとめ

本稿では、組込みソフトウェア開発でも今後注目されるであろうエンピリカルアプローチの基本的な考え方を解説するとともに、その実践と普及をめざし活動中の文部科学省EASEプロジェクトの概要を紹介した。

ソフトウェア開発における定量的データの収集や分析は今に始まったことではない。しかし、最近では、開発で作成されるドキュメント類の多くが、その作成履歴を含め構成管理ツールなどで電子ファイルとして記録され、コンピュータやネットワークの処理能力の向上が大量のデータの分析ややり取りを可能にしている。今後は、ソフトウェア開発会社全体や業界全体でソフトウェア開発データの収集を行い、それらを比較、分類し、ソフトウェア開発の現状把握や改善につなげるようなアプローチが試みられるようになると思われる。

EASEプロジェクトでも、このような研究アプローチを「メガソフトウェア工学（Mega Software Engineering）と名付け、その推進を目指している。メガソフトウェア工学を支える基盤技術には次の４つがあると考えている。

- プロジェクトにまたがった大規模データを収集、蓄積する技術（Inter-project Data Collection）
- 得られたデータを大域的に解析・評価する技術（Global Analysis）
- 評価結果に基づいて経験や知識を資産化する技術（Software Asset Management）
- 開発者個人、プロジェクト、組織間の情報交換技術（Knowledge Circulation）

先に述べたとおり、プロジェクト成功率で見る限り、組込みソフトウェア開発は、品質の面において、エンタープライズ系など従来のソフトウェア開発を大きく上回っている。しかし、そうした品質面での優位は、開発工数のおよそ4分の1（24.3%）を不具合の修正に充てている結果でもある［2］。エンピリカルアプローチやそれに基づく個々の技術が、組込みソフトウェア開発の今後の発展に寄与することを期待するところである。

6．謝辞

本稿の一部は、文部科学省「e-Society基盤ソフトウェアの総合開発」の委託に基づいて行われた研究成果に基づくものである。

7．参考文献

［1］EASEプロジェクトホームページ. http://www.empirical.jp/.
［2］2006年版組込みソフトウェア産業実態調査報告書，経済産業省，2006. https://sec.ipa.go.jp/download/200606es.php.
［3］“情報サービス・ソフトウェア産業維新：魅力ある情報サービス・ソフトウェア産業の実現に向けて（案）”，経済産業省・産業構造審議会・情報経済分科会・情報サービス・ソフトウェア小委員会，2006. http://www.meti.go.jp/press/20060613010/torimatome-hontai-set.pdf
［4］“特集：日本のソフト開発力を取り戻せ”，日経コンピュータ，2004年12月13日号，pp.52-69，2004.
［5］“特集：プロジェクト成功率は26.7%”，日経コンピュータ，2003年11月17日号，pp.50-62，2003.
［6］V. R. Basili, F. E. McGarry, R. Pajerski, M. V. Zelkowitz, “Lessons learned from 25 years of process improvement: The rise and fall of the NASA Software Engineering Laboratory,” Proc. of the 24th International Conference on Software Engineering (ICSE2002) , Orlando, Florida, USA, May 19-25, 2002.
［7］A. Endres, D. Rombach（著），吉舗紀子（訳），ＥＡＳＥプロジェクト（監修），ソフトウェア工学・システム工学ハンドブック：エンピリカルアプローチによる法則とその理論，コンピュータ・エージ社，2005.
［8］伏田，川口，飯田，“定量的管理を取り入れたソフトウェア開発計画立案支援システムの提案”，ソフトウェア信頼性研究会 第3回ワークショップ論文集，2006.
［9］井上，松本，鶴保，鳥居：“実証的ソフトウェア工学環境への取り組み”，情報処理，Vol.45，No.7，pp.722-728，2004.
［10］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，ソフトウェア開発データ白書2006　～IT企業1400プロジェク

トの定量データで示す開発の実態～，日経ＢＰ社，2006.
［11］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），ソフトウェア開発見積りガイドブック ～ＩＴユーザとベンダにおける定量的見積りの実現：ＥＡＳＥ協調フィルタリング法，pp.190-201，2006.
［12］柿元，角田，大杉，門田，松本，“ 協調フィルタリングによる工数見積もり手法におけるデータ数と見積もり精度の関係の分析 ”，日本ソフトウェア科学会 FOSE2005，ソフトウェア工学の基礎Ⅺ，pp.77-86，2005.
［13］川口，松下，井上，“ 版管理システムを用いたコードクローン履歴分析 ”，電子情報通信学会技術研究報告，SS2005-31，pp.43-48，2005.
［14］大杉，松村，森崎，“ 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，No.80，pp.13-29， 2006.
［15］大杉，角田，門田，松村，松本，菊地，“ 企業横断的収集データに基づくソフトウェア開発プロジェクトの工数見積もり ”，SEC journal，No.5，pp.16-25，2006.
［16］L. H. Putnam，W. Myers（著），山浦恒央（訳），初めて学ぶソフトウエアメトリクス～プロジェクト見積もりのためのデータの導き方～，日経ＢＰ社，2005.
［17］角田，門田，宿久，菊地，松本，“ 外部委託率に着目したソフトウェアプロジェクトの生産性分析 ”，電子情報通信学会技術報告，ソフトウェアサイエンス研究会，SS2006-11，pp.19-24，2006.
［18］角田，大杉，門田，松本，佐藤，“ 協調フィルタリングを用いたソフトウェア開発工数予測方法 ”，情報処理学会論文誌「産学連携」特集号，Vol.46，No.5，pp.1155-1164，2005.