ソフトウェアタグ・ガイドブック 2010

Software Tag Guide Book 2010

- **ソフトウェアタグ技術解説・日本語篇**

  Overview of Software Tag (in Japanese)

- **ソフトウェアタグ技術解説・英語篇**

  Overview of Software Tag (in English)

- **ソフトウェアタグ規格　第 1.0 版**

  Software Tag Standard Revision 1.0
  (in Japanese)

StagE プロジェクト

StagE Project

奈良先端科学技術大学院大学

Nara Institute of Science and Technology

大阪大学

Osaka University

Part1. ソフトウェアタグ技術解説・日本語編

技術解説「事故前提社会に向けたユーザ／ベンダ間での開発データ共有」（全三回）

「StagE プロジェクトとソフトウェアタグ」

SEC ジャーナル 2009 年 6 月号掲載

「ソフトウェアタグ規格とソフトウェアタグ支援ツール」

SEC ジャーナル 2009 年 7 月号掲載

「ソフトウェアタグ普及に向けた法的議論と利用技術基盤」

SEC ジャーナル 2009 年 8 月号掲載

事故前提社会に向けたユーザ・ベンダ間での開発データ共有 第1回
ー StagEプロジェクトとソフトウェアタグ ー

奈良先端科学技術大学院大学　ソフトウェア工学講座
文部科学省 StagE プロジェクト研究代表者
松本　健一

この技術解説では、ユーザ・ベンダ間でのソフトウェア開発データの共有を通じて、安心・安全なIT社会の実現を目指す、文部科学省 StagE※1 プロジェクトの活動とこれまでに得られた主な成果を数回にわたって紹介していく。

第１回となる本稿では、プロジェクトの概要、とくに、ソフトウェアタグとタグを用いた情報共有の基本概念、タグ普及に向けて研究開発を進めている基本技術のあらましと今後の取組み予定について紹介する。

1. はじめに

米国商務省国立標準技術研究所（NIST）が2002年に実施した調査によると、ソフトウェアのバグによる米国内での損失額は年間595億ドル、国内総生産の0.6%にも上る[NIST2002]。日米間でソフトウェア開発プロジェクトの成功率に大きな差が無いことから、日本の国内総生産515.8兆円（2007年、名目）を基に単純計算すると、ソフトウェアバグによる日本国内での損失額は年間約3兆円にも達しているということになる。

日米間で状況が異なる面もある。総務省が発表した情報通信白書平成20年版によると、日本の主要ITベンダの営業利益率は、日本国内で6.4%、海外ではわずか1.8%である（表１）[MIC2008]。一方、米国ITベンダは、米国内で15.6%、海外で13.7%である。日米間で営業利益率に数倍の差があることも問題であるが、日本ベンダの海外での利益率の低さは際立っている。「ユーザとの関係が確立されていない海外市場で日本ベンダは苦戦を強いられている」と白書では述べられている。電子情報技術産業協会（JEITA）が2006年に発表した「海外・国内企業におけるソフトウェアのオフショア開発についての調査・分析と提言」でも、『日本企業のオフショア開発能力レベルの平均は、「オフショア活用方針・基準」、「計画・契約」、「実行管理」、「評価」すべてにおいて、米国企業を下回る』と指摘されている。

「ユーザとの関係」、とくに、ユーザ・ベンダ間のコミュニケーションが良好でないと、オフショア開発でなくとも、コスト／スケジュール超過、品質低下、モチベーション／モラル低下、といったソフトウェア開発リスクが増大する場合がある。例えば、ユーザ・ベンダ間で非機能要件に関する事前合意が十分になされていなかった

表1　ITベンダ営業利益率

	国内	海外
日本	6.4%	1.8%
米国	15.6%	13.7%

出典：情報通信白書平成20年版（総務省）

※1　StagE：Software traceability and accountability for global software Engineering

ために、設計レビューにおけるバグの見逃し件数がおよそ3倍になったという事例もある。

ユーザ・ベンダ間で良好なコミュニケーションを実現する1つのアプローチは、ソフトウェアの開発状況を示す実証データ（開発データ）の共有である。これにより、ユーザは、ベンダが提示する開発・管理計画や進捗報告の妥当性を確認することが出来、自らのソフトウェア要求が、開発プロジェクトの進捗と共に、どのように実装、検証されていくのかを実感出来るようになる。一方、ベンダも、ユーザと共同で開発リスクの低減に取組むことが出来、自らの説明責任やコンプライアンスを明確にすることも容易になる。

内閣官房情報セキュリティセンターが2009年2月に発表した「第2次情報セキュリティ基本計画」においても、「事故前提社会」への対応力強化のため、「冷静で迅速な対応、説明責任の明確化」、「利用者にとっての安全・安心の確保」等を実現すべきであり、とくに、重要インフラ分野においては、「IT障害の発生は隠すべきものではなく、関係者間で共有すべき」とある[NISC2009]。我々が行ったアンケート調査でも、「ユーザの95%、ベンダの92%が、開発データ共有の有用性を強く認識している」との結果が得られている[StagE]。

本技術解説では、ユーザ・ベンダ間でのソフトウェア開発データの共有を通じて、安心・安全なIT社会の実現を目指す、文部科学省StagEプロジェクトの活動とこれまでに得られた主な成果を数回にわたって紹介していく。第1回の本稿では、StagEプロジェクトが提案するソフトウェアタグとタグを用いた情報共有の基本概念、タグ普及に向けた取組み等について概説する。

2. ソフトウェアタグ

StagEプロジェクトとは、文部科学省からの委託により奈良先端科学技術大学院大学と大阪大学が2007年8月から5年計画で実施している研究開発プロジェクトである[StagE]。正式名称は、「次世代IT基盤構築のための研究開発：ソフトウェア構築状況の可視化技術の普及：エンピリカルデータに基づくソフトウェアタグ技術の開発と普及」である。その目的は、ソフトウェア開発が適正な手順で行われたかどうかを表す実証データをユーザ・ベンダ間で共有する枠組みを構築し、その基盤となる技術を世界に先駆けて開発することである。ユーザ・ベンダ間でのデータ共有のメディアとして提案されているのが「ソフトウェアタグ」である。

ソフトウェアタグとは、ソフトウェア開発に関する実証データから、ソフトウェアやその開発プロジェクトの特徴量を算出し、ユーザにも理解しやすく、可視化や評価にも利用しやすい形式でとりまとめた情報パッケージである。ソフトウェア開発終了後にソフトウェア製品に添付され、ベンダからユーザへ提供されることになるが、開発途中に進捗報告書に添付するといった利用形態も考えられる。

ソフトウェアタグの基となる実証データは、次の3種類に大別される。

① 開発期間、開発規模、開発工数、総不具合数等、開発プロジェクトのプロファイルデータ
② 構成管理システムや不具合（障害）管理システムの記録（ログ）、進捗会議資料
③ 要求仕様書、設計ドキュメント、プログラムコード、テスト報告書等中間成果物や最終成果物の作成・更新履歴

いずれも、今日のソフトウェア開発プロジェクトでは、開発管理のために一般的に収集され電子的に保存されているデータである。ソフトウェアタグ生成のためのデータ収集コストは比較的小さいと考えられる。

ソフトウェアタグは、ベンダからユーザへの一方的な情報提供の手段と考えられがちであるが、必ずしもそうではない。例えば上記③には、ユーザから開発当初に出された仕様要求に関するデータが含まれ、②には、開発途中にユーザから出された仕様変更依頼とその結果発生した構成変更や不具合に関するデータが含まれる。他にも、受け入れテスト等において、ユーザ主導で行われる作業があれば、それらの実証データからは、ベンダより

もむしろユーザに関する特徴量が算出されることになる。

ソフトウェアタグが実現する安心・安全なソフトウェア開発のあらましを図１に示す。プロジェクト計画時、ベンダは、当該プロジェクトと類似するプロジェクトで生成されたソフトウェアタグを、自らの開発・管理能力を示す実績データとして開発・管理計画書に添付することが出来る。一方、ユーザも、保有するソフトウェア（ソフトウェア資産）に添付されているソフトウェアタグの情報と開発・管理計画書を比較することで、自らの組織や業務領域（ドメイン）において、開発・管理計画が妥当かどうかを確認することが出来る。

プロジェクト実行時、ベンダは、当該プロジェクトでその時点までに生成されたソフトウェアタグを進捗報告書に添付することで、報告書に嘘偽りはなく、自らの業界や社内標準に照らして妥当であると示すことが出来る。一方、ユーザも、自らの業界や社内標準に照らして進捗報告書の妥当性を確認することが出来るだけでなく、開発リスクを早期に検知し、必要な改善要求をベンダに出すことが可能となる。なお、ベンダが開発作業の一部を外部ベンダに委託している場合、外部ベンダからの進捗報告に対して、ベンダはユーザと同様の立場となる。もし妥当性の確認出来ない進捗報告があれば、その原因を究明し、対策を講じることで、説明責任やコンプライアンスの体制を強化することが出来る。

3. ソフトウェアタグ技術

StagEプロジェクトでは、ソフトウェアタグによる開発データの共有を実現する基本技術として、「ソフトウェアタグ規格」、「ソフトウェアタグ利用シナリオ」、「ソフトウェアタグ支援ツール」の研究開発を進めている。更に、それら3つの技術によって実現されるソフトウェアタグの法的意義についての検討も行っている（図2）。詳細については次回以降で解説するとして、ここではそのあらましを簡単に紹介する。

3.1 ソフトウェアタグ規格

ソフトウェアタグ規格は、タグの構造やタグ情報として含めるべき項目や形式を定義したドキュメントである。

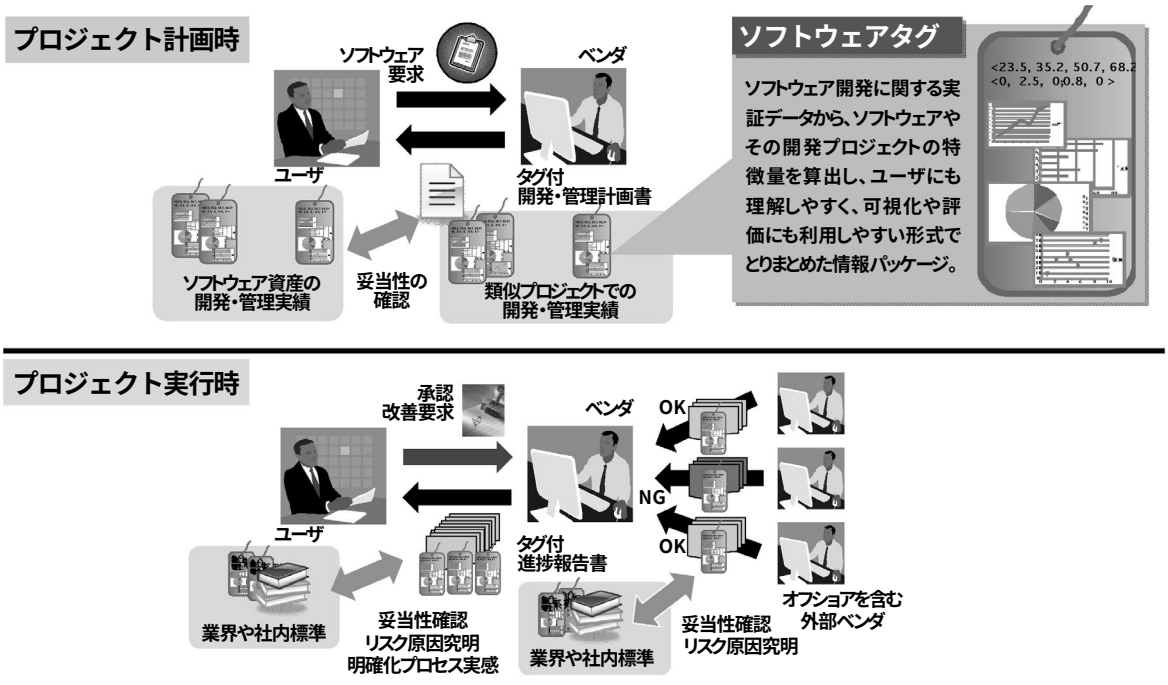

図1　ソフトウェアタグが実現する安心・安全なソフトウェア開発

13組織31名からなる「ソフトウェアタグ規格技術委員会」での議論を経て、2008年10月に「ソフトウェアタグ規格第1.0版」が公開された（表2）[StagE]。

ソフトウェアタグ規格第1.0版は、対象ソフトウェアが開発されたプロジェクトの特性と開発時の進捗状況を表す計41個のタグ項目で構成されている。タグ項目ごとに、具体化例や基となる実証データ例が記載されており、ソフトウェアタグ利用シナリオやソフトウェアタグ支援ツールのベースと言える。

3.2　ソフトウェアタグ利用シナリオ

ソフトウェアタグ利用シナリオとは、開発データ共有の目的や目標といった抽象的な概念を、ソフトウェアタグ規格で規定された具体的なデータに対応付けると共に、開発データを収集するタイミング、解釈方法や評価基準等を記したドキュメントである。タグ利用を効率良く、また、効果的に実施するためのドキュメントであるが、ユーザ・ベンダ間での合意形成を助ける道具とも言える。

現在、ソフトウェアタグ利用シナリオに求められる要件を明らかにするための実証実験を、株式会社東京証券取引所と株式会社日立製作所の協力を得て実施中である（図3）。

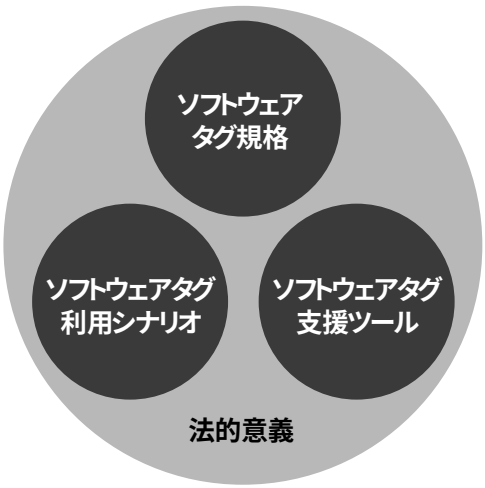

図2　ソフトウェアタグ技術

表2　ソフトウェアタグ規格第1.0版（一部）

分類	項番	タグ項目	具体化例	予定・実績の要否
要件定義	13	ユーザヒアリング情報	ユーザヒアリング実施件数（回）	○
			ユーザヒアリング項目数（件）、 ユーザヒアリング回答率（ユーザヒアリング回答数÷ユーザヒアリング項目数） 等	
	14	規模[推移]	画面、機能項目、ユースケース、アクター、顧客要件、機能、FP等	○
	15	変更[推移]	規模の計測単位に依存	
設計	16	規模[推移]	機能設計（ページ数・帳票数・画面数・ファイル数・項目数・UML図の数、クラス数、バッチプログラム数、重要な機能数等） 構造設計（データ項目数、DFDデータ数、DFDプロセス数、DBテーブル数 等） 等	○
	17	変更[推移]	規模の計測単位に依存	
	18	要件の網羅率	設計に取り入れられた要件数÷要件数	

3.3　ソフトウェアタグ支援ツール

ソフトウェアタグ支援ツールは、ソフトウェアタグ規格やソフトウェアタグ利用シナリオに準拠したデータ収集を容易にし、「見える化」や「開発計画策定」といったソフトウェア開発管理への応用を助ける道具である。

現在開発を進めているタグデータ収集システムの概要を図4に示す。システムへの入力は、開発データ履歴と収集設定である。開発データ履歴とは、IPA/SECで検証プロジェクトが実施されているEPM※2[EASE][EPM]等で収

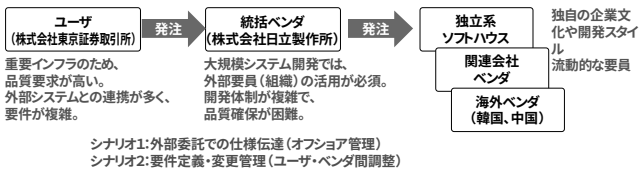

図3　ソフトウェアタグ利用シナリオ構築に向けた実証実験

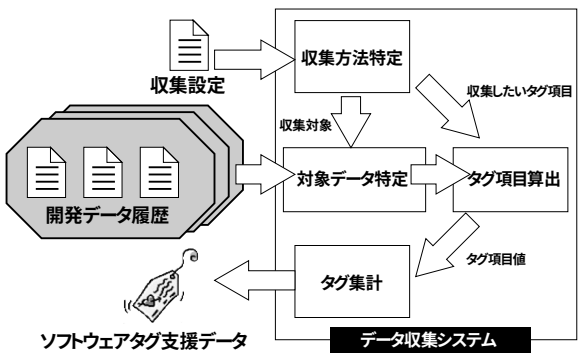

図4　タグデータ収集システム概要

集されたデータや、プロジェクト開発フォルダのバックアップ（の系列）等である。収集設定とは、開発データ履歴と収集すべきタグ項目の対応関係を示すものである。

タグデータ可視化ツールTagReplayerの画面例を図5に示す。TagReplayerは、「ある時点」でのプロジェクトの状態をあたかも録画したビデオを再生するように提示することが出来る。先述のタグデータ収集システムと同様に、EPMで収集されたデータを入力とすることが出来る。

3.4 ソフトウェアタグによる可視化に伴う法的諸問題

StagEプロジェクトにおける法的な議論の目的は次の2つである。

① コンピュータが動かなかったときに、事後的な紛争処理を分析することによって、未然に紛争を防止する。

② ソフトウェア開発において、ソフトウェアタグをプロジェクトマネジメントに利用するだけでなく、ユーザとベンダで共有し、これを活用することによって、紛争処理に係る労力を軽減し、ひいては、動かないコンピュータを減らす。

具体的には、ソフトウェア構築における紛争や失敗事例について、文献・判例分析、ソフトウェア開発企業に対するヒアリング／アンケート等を行い、ソフトウェア開発におけるトラブルの実態を明らかにすると共に、弁護士等法律の専門家とソフトウェア技術の専門家の文理融合領域からなる法的問題検討委員会を設立し、研究の方向性を定めている。

以上のように、ソフトウェアタグによる開発データの共有は、「見える化」に代表される定量的アプローチの1つである。ただし、ユーザ・ベンダ双方の視点を取り入れる等、新しい点も多い。関連する既存の組織・プロジェクトでの取組みとStagEプロジェクトとの違いを表3にまとめる。

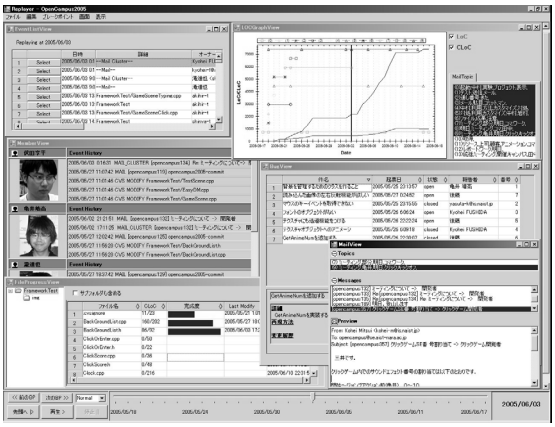

図5　タグデータ可視化ツール TagReplayer

表3　既存組織・プロジェクトでの取組みとStagEプロジェクトとの比較

関連研究を行っている組織・プロジェクト		StagEプロジェクト
ISGSG （オーストラリア）	ベンチマーキングデータの提供。 データ収集ツールが提供されておらず、 組織間でのデータの整合性に問題あり。	タグ規格準拠ツールを開発中。 組織間でのデータ共有が目的。
NASA Metrics Data Program （米国）	プログラムコードレベルのメトリクス、 障害データ、要求に関するデータが蓄積、公開されている。 開発プロセスを把握出来るデータは含まれていない。	プロジェクトのプロファイルや 進捗状況に関するデータも対象。
Fraunhofer IESE （ドイツ）	ドイツ最大の研究機関の一研究所で、 産学連携を通じたエンピリカルソフトウェア工学を推進している。 データや分析結果が広く公開されることはない。	研究開発の成果を、ソフトウェアタグ規格、 ツール、利用シナリオ等として公開。
HackyStat（米国）、 IPA/SEC、 JUAS（日本）、…	ベンダ、もしくは、ユーザの視点でデータが収集、 蓄積されている。	ベンダ・ユーザ双方の視点での、 ソフトウェア開発の見える化。 法的観点からも議論。
経済産業省 （日本）	情報システム・モデル取引・契約書等により、 ユーザ・ベンダ間でやり取りすべきドキュメントや 手順を規定している。	ユーザ・ベンダ間でやり取りされる ドキュメントの評価や組織間比較を可能に （左記とは補完関係）。

※2　EPM：Empirical Project Monitor

4. 実用化・普及に向けて

2007年8月のプロジェクト開始から約2年が経過し、様々な技術的検討を経て、「ソフトウェアタグ規格第1.0版」を始めとする具体的な成果も得られてきている。今後は、ソフトウェアタグの実用化・普及に向けた次のような取組みにより注力する予定である。

① ソフトウェアタグ規格やツールの標準化

ソフトウェアタグを一般的なものとし、多くのユーザ・ベンダで共有するためには、タグ規格やツールの標準化が必要である。StagEプロジェクトが実施したアンケート調査の結果でも、多くのユーザが、開発データの共有の実用化・普及の方策として、「データやその共有方法の標準化」を挙げている。ソフトウェアタグ規格についてはJIS化やISO化に向けた取組みをIPA/SEC等と、ツールについてはデファクトスタンダードに向けた取組みをツールベンダと、それぞれ行っていく予定である。

② ソフトウェアタグ実証実験の推進

ソフトウェアタグの有用性をユーザ・ベンダに具体的に示すことも実用化・普及には必要である。StagEプロジェクトが実施したアンケート調査の結果でも、多くのベンダが、開発データの共有の実用化・普及の方策として、「開発データの共有（ユーザへの開示）のメリットの明確化」を挙げている。開発データの共有に対するユーザ・ベンダニーズに基づくソフトウェアタグ利用シナリオ案を幾つか作成し、実際のソフトウェア開発プロジェクトに適用することで、タグ利用の具体的なイメージやメリットを明確にしていく予定である。

③ 国際連携の強化

オフショア開発は、ソフトウェアタグによる開発データ共有の主要な適用対象の１つである。日本からのオフショア開発圏となるアジア・太平洋諸国の研究開発機関と連携し、ソフトウェアタグの妥当性や有用性を検討することは、実用化・普及に大きく貢献すると考えられる。既に、アジア・太平洋圏ソフトウェア工学研究ネットワークAPSERN※3の設立に向けた準備会議を、オーストラリアNICTA※4、中国ISCAS※5の研究者と共に開催し、活動基本方針の策定、具体的な活動内容の検討を始めている。2009年12月には、APSERN設立記念ワークショップをマレーシアで開催する予定である。

5. おわりに

以上が、文部科学省StagEプロジェクトとソフトウェアタグの概要である。プロジェクトを進めるに当たり、多くのユーザ・ベンダ企業にご協力いただいてきたこともあり、最近では、国の戦略的取組みへの波及も見られるようになってきている。例えば、経済産業省商務情報政策局情報処理振興課が2009年3月に意見公募の対象として公表した「高度情報化社会における情報システム・ソフトウェアの信頼性及びセキュリティに関する研究会の中間報告書（案）」では、「複雑化する情報システム・ソフトウェアの信頼性及びセキュリティ水準を高度化するためには、ソフトウェアタグの高度化と現場への適用を検討する必要がある」とソフトウェアタグの重要性が述べられている[METI2009]。

次回以降では、ソフトウェアタグによる開発データの共有を実現する基本技術「ソフトウェアタグ規格」、「ソフトウェアタグ利用シナリオ」、「ソフトウェアタグ支援ツール」、及び、それらによって実現されるソフトウェアタグの「法的意義」について、具体的な成果を交えて詳しく紹介していく。

参考文献

[EASE] http://www.empirical.jp/
[EPM] http://sec.ipa.go.jp/tool/epm.html
[JEITA2006] 電子情報技術産業協会：海外・国内企業におけるソフトウェアのオフショア開発についての調査・分析と提言，2006
[METI2009] 経済産業省 商務情報政策局 情報処理振興課：高度情報化社会における情報システム・ソフトウェアの信頼性及びセキュリティに関する研究会の中間報告書（案），2009
[MIC2008] 総務省：情報通信白書平成20年版，2008
[NISC2009] 内閣官房情報セキュリティセンター：第2次情報セキュリティ基本計画，2009
[NIST2002] National Institute of Standards and Technology：The Economic Impacts of Inadequate Infrastructure for Software Testing，2002
[StagE] http://www.stage-project.jp/

※3　APSERN：Asia-Pacific Software Engineering Research Network
※4　NICTA：National Information/Communication Technology Australia
※5　ISCAS：Institute of Software Chinese Academy of Sciences

事故前提社会に向けたユーザ・ベンダ間での開発データ共有 第2回
ーソフトウェアタグ規格とソフトウェアタグ支援ツールー

大阪大学
大学院情報科学研究科　教授
井上　克郎

大阪大学
大学院情報科学研究科　教授
楠本　真二

奈良先端科学技術大学院大学
情報科学研究科　教授
飯田　元

第１回は、「ソフトウェアタグ」と、ソフトウェアタグの開発、普及を目指す「StagE※1プロジェクト」の概要について述べた。
第２回である今回は、ソフトウェアタグの規格の第１.０版の詳細と、ソフトウェアタグを効率良く作成するためのデータ収集支援ツール及びソフトウェアタグのデータを可視化し分析するツールについて述べる。

1. はじめに

前回第1回（SEC journal No.17）は、「ソフトウェアタグ」と、ソフトウェアタグの開発、普及を目指す「StagEプロジェクト」の概要について述べた[松本2009]。

ソフトウェアタグは、ソフトウェアシステムのユーザ（発注者、購入者、利用者）が、納品された、あるいは購入・流用したシステムを安心して安全に用いるために、ソフトウェアの開発プロセスやソフトウェア製品（プロダクト）に関する情報をベンダ（開発者、販売者）と共有する仕組みである。このように開発時に得られる種々のデータ（実証的（エンピリカル）データ）をユーザに提供することにより、

・ユーザによるソフトウェア品質の検証
・ユーザによる適正なソフトウェア製品の選択促進
・問題発生時の対応の迅速化
・透明性の拡大による法的問題の発生の予防と早期の公正な解決の促進

等が可能となる。

今回は、ソフトウェアタグの規格の第1.0版の詳細と、ソフトウェアタグを効率良く作成するためのデータ収集支援ツール及びソフトウェアタグのデータを可視化し、分析するツールについて述べる。

2. ソフトウェアタグ規格第1.0版

2.1 ソフトウェアタグの規格化

我々は、ユーザが購入し利用しようとするソフトウェアシステムの品質や性能に関して、定量的な評価を行えるようにするために、ソフトウェアタグをソフトウェア取引に導入することを提案している。

どのような指標があれば、ユーザが定量的な評価を行えるようになるかに関して、ソフトウェアのベンダ企業、ユーザ企業、及び大学や政府機関等の13機関31名が集まって、計13回の委員会（ソフトウェアタグ規格技術委員会）を開き、議論を積み重ねてきた（参加組織については第1回を参照）。そして、この委員会で、2.2節で述べる

※1　StagE：Software traceability and accountability for global software Engineering

41種類の指標（ここでは「タグ項目」と呼ぶ）の集合をソフトウェアタグ規格第1.0版とすることを決めた（2008年10月14日）[StagE2008]。以降、本稿では、このソフトウェアタグ規格第1.0版のことを単に「タグ規格」と呼ぶことにする。

2.2　タグ規格の全体像

タグ規格は、プロジェクト情報として12項目（表1）、進捗情報として29項目（表2）の合計41項目から構成さ

表1　タグ規格第1.0版（プロジェクト情報）

分類	項番	タグ項目	説明
基本情報	1	プロジェクト名	プロジェクトを一意に決定するための識別名
	2	開発組織の情報	当該プロジェクトの開発を担当する組織の情報。一般には、受注者となる組織情報となる。
	3	開発プロジェクト情報	開発プロジェクトの特徴や当該タグデータの対象とするプロジェクトの種類を示す情報。タグデータの解釈や分析時に必要なデータ。
	4	顧客情報	当該システムのユーザ、もしくは第1発注者となる組織の情報。
システム情報	5	システム構成	開発システム構成の特徴や当該タグデータの対象とするシステムの種類を示す情報。タグデータの解釈や分析時に必要なデータ。
	6	システム規模	開発システムの規模。計画値と最終実績値とする。進捗情報に同じ情報が含まれる場合は、省略可。
開発情報	7	開発手法	開発システム開発に用いたプロセスや手法についての情報。タグデータの解釈や分析時に必要なデータ。
	8	開発体制	開発側の要員に関する情報。タグデータの解釈や分析時に必要なデータ。 ※開示対象範囲は、発注者・受注者側での協議により決定する
	9	プロジェクト期間	当該プロジェクトの開発期間に関する情報
プロジェクトの階層構造情報	10	親プロジェクト情報	本プロジェクトが別のプロジェクトのサブ(子)プロジェクトである場合、付加
	11	サブ(子)プロジェクト情報	本プロジェクトがサブ(子)プロジェクトを持つ場合、その数やサブ（子）プロジェクトに関する情報
その他	12	特記事項	その他、タグデータの解釈や分析時に必要、もしくは有用なデータ。

表2　タグ規格第1.0版（進捗情報）

分類	項番	タグ項目	説明
要件定義	13	ユーザヒアリング情報	要件に関してユーザに行ったヒアリングに関する情報
	14	規模[推移]	開発側で作成した要件数
	15	変更[推移]	変更された要件数
設計	16	規模[推移]	設計成果物の規模 ※新規・改造・再利用(流用)毎に計測する
	17	変更[推移]	変更された設計成果物の数、もしくは変更量
	18	要件の網羅率	要件定義で作成された要件の実装率
プログラミング	19	規模[推移]	プログラミング成果物の規模 ※新規・改造・再利用(流用)毎に計測する
	20	変更[推移]	変更されたプログラムの数、もしくは変更量
	21	複雑度	プログラムの品質（保守性）
テスト	22	規模[推移]	テストの規模 ※新規・改造・再利用(流用)毎に計測する
	23	変更[推移]	変更されたテスト項目数や変更量
	24	密度	テストの品質
	25	消化	テストの進捗、プログラムの品質
品質	26	レビュー状況	成果物(仕様書、設計書、プログラムコード、テスト仕様書など）のレビューに関する情報
	27	レビュー作業密度	レビュープロセスの品質、もしくはレビュー対象の品質
	28	レビュー指摘率[推移]	レビュープロセスの品質、もしくはレビュー対象の品質
	29	欠陥件数[推移]	テスト設計の品質とコード品質
	30	欠陥対応件数	欠陥の対応進捗、対応内容
	31	欠陥密度	テスト設計の品質とコード品質
	32	欠陥指摘率	テスト設計の品質
	33	静的チェックの結果	プログラムの品質（保守性）
工数	34	作業工数	作業に要する工数、仕様変更作業工数
	35	生産性	工数に対する成果物の比率
計画・管理	36	プロセス管理情報	開発プロセスの管理に関する情報
	37	会議実施状況	ユーザ・ベンダ間、ベンダ間での情報共有状況を把握
	38	累積リスク項目数	リスク認識が十分であったかを把握
	39	リスク項目の滞留時間	リスク対策が適切になされていたかを把握
その他成果物	40	規模[推移]	対象成果物の規模 ※新規・改造・再利用(流用)毎に計測する
	41	変更[推移]	変更された対象成果物の数、もしくは変更量。

表3　具体化例や実証データ例(一部)

分類	項番	タグ項目	説明	具体化例	実証データ例	予定・実績の要否	備考
要件定義	13	ユーザヒアリング情報	要件に関してユーザに行ったヒアリングに関する情報	ユーザヒアリング実施件数（回）	ユーザヒアリング議事録ユーザヒアリング質問票など	○	
				ユーザヒアリング項目数（件）、ユーザヒアリング回答率（ユーザヒアリング回答数÷ユーザヒアリング項目数）など			
	14	規模[推移]	開発側で作成した要件数	画面、機能項目、ユースケース、アクター、顧客要件、機能、FPなど	要件定義書など	○	何を要件の基本単位とするかは、要合意事項
	15	変更[推移]	変更された要件数	規模の計測単位に依存	要件定義書 要件定義書の変更履歴など		

表4　あるプロジェクトのタグの例（一部）

分類	項番	タグ項目	計測値
基本情報	1	プロジェクト名	A大学新規教務システム
	2	開発組織の情報	要求定義：B株式会社 設計、実装、テスト：C株式会社
	3	開発プロジェクト情報	開発プロジェクト種別：新規
	4	顧客情報	開発プロジェクト形態：受託開発
システム情報	5	システム構成	顧客：A大学教務掛
			OS：Windows ブラウザ：Internet Explorer その他：Adobe Reader
	6	システム規模	26033行（Java、一部の実装サイズ）
開発情報	7	開発手法	オブジェクト指向開発
	8	開発体制	要求仕様作成チーム：B株式会社（5名） 設計仕様作成チーム：C株式会社（3名）
	9	プロジェクト期間	要求・設計期間：2007年3月27日～5月31日 実装期間：2007年12月3日～2008年2月28日まで
プロジェクトの階層構造情報	10	親プロジェクト情報	
	11	サブ（子）プロジェクト情報	
その他	12	特記事項	

分類	項番	タグ項目	計測すべきメトリクス	計測値
基本情報	13	ユーザヒアリング情報	ヒアリング回数	4
	14	規模[推移]	ユースケース図数	12
	15	変更[推移]	ユースケース図数	48
設計	16	規模[推移]	UML図数	128
	17	変更[推移]	UML図数	434
	18	要件の網羅率		
プログラミング	19	規模[推移]	行数（全体）	26033
	20	変更[推移]	変更量（追加＋削除行数）	88841
	21	複雑度	WMC※2	6.277551
			LCOM※3	10.955102
			NOC※4	0.7387755
			DIT※5	2.8081632
			CBO※6	10.43
			RFC※7	12.995918
テスト	22	規模[推移]	テストケース数	617
	23	変更[推移]	テストケース数	617
	24	密度	テストケース数/全体行数	0.0237
	25	消化	消化テスト数	584
品質	26	レビュー状況	レビュー回数	21
	27	レビュー作業密度	レビュー時間/全体時間	1
	28	レビュー指摘率[推移]	レビュー指摘数	649
	29	欠陥件数[推移]	全体欠陥件数	19
	30	欠陥対応件数	全体欠陥対応件数	19
	31	欠陥密度	全体欠陥件数/行数	0.000730
	32	欠陥指摘率	全体欠陥件数/消化テスト数	0.0325
	33	静的チェックの結果	FindBugs指摘件数	52
工数	34	作業工数	作業時間	152日
	35	生産性	行数/作業時間	171.3（行/日）
計画・管理	36	プロセス管理情報		
	37	会議実施状況		
	38	累積リスク項目数		
	39	リスク項目の滞留時間		
その他成果物	40	規模[推移]		
	41	変更[推移]		

れる。

プロジェクト情報は、基本、システム、開発、プロジェクトの階層構造の情報とその他に分類され、それぞれは1～4個のタグ項目を含んでいる。同様に、進捗情報は、要件定義、設計、プログラミング、テスト、品質、工数、計画・管理、その他成果物に分類され、それぞれ2～8のタグ項目を含んでいる。

各タグ項目は、それを表す名前（例えば「プロジェクト名」）とその簡単な説明（「プロジェクトを一意に決定するための識別名」）から構成されている。

タグ規格として決めているのはここまでで、具体的にどういう記述、メトリクス、データをタグ項目として開発者からユーザに受け渡すのかは、二者間で取り決めて決定する。

本タグ規格では、この二者間の取り決めを容易にするため、表3のように、メトリクスの例を示している。また、そのメトリクスを収集するための実証データの例も示している。実際にはこの中から適当なものを選んで利用しても良いし、また、別のメトリクスを用いることも可能である。「予定・実績の要否」は、対応する具体化例のデータを収集する前に、あらかじめ目標値（予定値）を設定し、その予定と実績の管理をすることが望ましいものであることを示している。

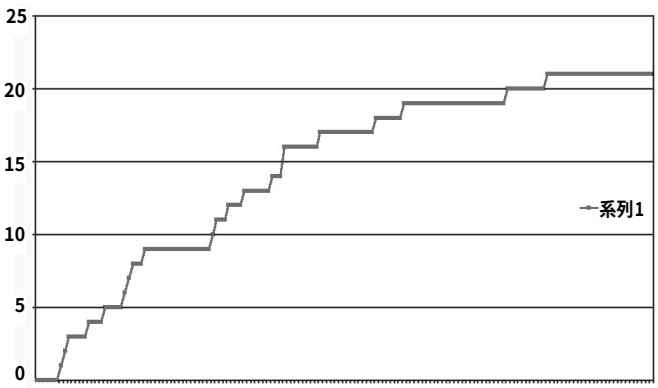

図1　レビュー回数（項番26）のデータの推移

※2　WMC：計測対象クラスの重み付きメソッド数
※3　LCOM：計測対象クラスの凝集性の欠如
※4　NOC：計測対象クラスのサブクラス数
※5　DIT：計測対象クラスの継承の深さ
※6　CBO：計測対象クラスに関係しているクラス数
※7　RFC：計測対象クラスに関係しているメッセージ数

2.3　タグの実例

表4に実際のプロジェクトの情報から作成したタグの例を示す。対象は、ある大学の教務システムの開発プロジェクトである。ここでは、具体的に計測したメトリクスを決めて、計測を行ったデータを集計したものを示している。この表では、例えば、複雑度に関しては6つのメトリクス（WMC※2、LCOM※3、NOC※4、DIT※5、CBO※6、RFC※7）を計測することにして、プロジェクト終了時点のプログラムに対して計測した値を示している。実際には、プロジェクト終了時、作成された29個のモジュールそれぞれの複雑度メトリクスを平均した値を示している。

この例で利用しなかった（計測しなかった）項目は斜線を引いている。この例のように、二者間で合意すれば41項目すべてを利用する必要は無い。

利用した各メトリクスの多くは、日単位で計測されており、そのデータもタグに含めている（大きくなるのでここでは表示していない）。図1は、レビュー状況（項番26）のメトリクスとしてレビュー回数を選び、その回数の増加をグラフ化したものである。ユーザは、このようなグラフや値を見て、プロジェクトの進捗や品質を評価する。

2.4　タグ規格の考え方

(1) ユーザ視点の実証的データの提供

通常、ベンダは、プロダクトの品質向上や組織の生産性向上等のために、種々のデータ（実証的データ）を収集し、フィードバックを行い、プロジェクトや組織を改善することが当たり前である。このようなフィードバックループは、もっぱらベンダ内に閉じているが、ソフトウェアタグの仕組みで目指すのは、ユーザを巻き込んだ大きなフィードバックループによる改善である。ユーザが提供される各タグ項目のデータを評価し、プロジェクトの品質について積極的に関与することで、プロジェクトの透明性が向上し、安心・安全なソフトウェアシステムの構築や運用につながる。

(2) タグ項目の選定のポリシー

ベンダ内で通常収集されている種々のデータの中で、ユーザにとって簡潔で理解しやすいものを、バランスに配慮してタグ項目とした。ユーザが持つ「対象のソフトウェアシステムはどんなものか？」、「どのように作られたか？」という疑問に対して、タグ項目の大分類のプロジェクト情報と進捗情報を用意している。

「どんなものか？」に対応するプロジェクト情報は、以下のような5種類の情報に分類し、対応するタグ項目群を設けた。

・プロジェクトの基本情報→基本情報
・稼動するシステムの情報→システム情報
・開発の基本的な情報→開発情報
・プロジェクト間の関連情報→プロジェクトの階層構造情報
・その他→その他

また、「どのように作られたか？」に対応する進捗情報は、主にISO/IEC 12207/SLCP※8の開発プロセスを元に、各工程の情報を用意すると共に、品質や工数の情報を加えた。

・要件定義、設計、プログラミング、テストの各情報
・品質の担保情報
・工数情報
・計画・管理情報
・その他

(3) 二者間の取り決め

本タグ規格は、ソフトウェアタグとしてベンダからユーザにどのようなデータ集合を提供するかを詳細に規定するものではない。提供すべき大枠を示しているのみで、詳細に関しては、ユーザ、ベンダの二者間で決める必要がある。決める必要のあるものとしては、

・41項目のどのタグ項目を利用するか（全部の利用が前提ではない）
・各タグ項目として用いるメトリクス
・各メトリクスの計測対象（全システム一括、サブシス

※8　SLCP：Software Life Cycle Process

テムごと、各ファイルごと等）
・計測頻度
・タグとしてユーザに提供する時期（毎週、毎月、工程ごと等）
等がある。

2.5 議論

ここでは、ソフトウェアタグ及び本タグ規格に関する幾つかの論点を示す。

（1）もっと多くのタグ項目が必要では？

現実のソフトウェアの開発現場では、もっと多様なデータを集め、分析を行って、改善活動を行っている。そのうちのごく一部だけをユーザに提示することに、どれだけの効果が得られるかは、不明な部分も多い。しかし、規格として一般化する場合の、タグの収集・構成コストや中小プロジェクトでの実現可能性等のバランスを考え、本規格を定義した。より大規模なプロジェクトでは、その他の項目をタグとして追加することも可能で、逆に小規模なプロジェクトでは、本規格の一部のみを利用することも可能である。

（2）進捗報告会議との違いは？

ソフトウェアタグの仕組みとほぼ同様な情報提示を、ユーザと定期的に開く進捗報告会議で行っているベンダは多い。二者間できちんと情報交換して品質を担保しようとする場合、タグ項目のデータは当たり前のものと言えよう。従って、ソフトウェアタグとそのような会議での情報交換は、同様な効果をもたらす。本規格は、このようなユーザを含めた改善活動を普及させるための基礎となる。

（3）タグの仕組みの分かりやすいアナロジーは？

人間ドックは、対象者の健康度を知るために数十項目のデータを収集、分析して対象者に示し、健康状態を知ることが出来るようにする。また、企業の決算報告等で用いられる財務諸表は、企業の資金や資本の大きさや動きを数十項目の金額で示し、投資家や取引先に開示し、その企業の健全性を示す。

これらと同様、ソフトウェアタグは、数十項目のソフトウェアプロジェクトやプロダクトに関するメトリクスデータをユーザに開示し、プロジェクトやプロダクトの健全性、品質等の評価をする際の重要な指標になる。

（4）タグ項目のデータが改ざんされるのでは？

このような可能性はあり得るが、整合性のあるデータを複数の項目、複数の版にわたって偽造や改ざんするのは容易ではない。一方、3節で述べるように、開発環境からデータを抽出しタグ化することは比較的容易である。このように、偽造や改ざんには大きなコストがかかる上に、発覚した場合のため一時は計りしれない。

タグ項目のデータの基礎となる開発時の詳細なデータ全体を第三者に預託しておき、紛争時にタグのデータの正当性を検証出来るような枠組みを作っておくのも良い方法かもしれない。

（5）タグ規格の今後の方向性については？

現在の第1.0版では、出来るだけ用途を広くするために、具体的なメトリクスやその収集方法を規格として規定していない。しかし、実際に対象とするプロジェクトを前にして、どのようなメトリクスを利用するか、41項目のタグすべてユーザとベンダが協議することは簡単ではない。

従って、ある程度よく利用されるパターンを想定して、メトリクスやその計測方法を具体化した追補規定の設定を考えている。例えば、「中規模エンタープライズシステム開発において、ベンダの開発活動を正しく伝えるための追補規定」、「新規開発案件において、要件の間違いや法的問題の発生を少なくするための追補規定」等が考えられる。また、メトリクスが具体化すれば、予想される標準値もSECのデータ等を参考にして盛り込むことも可能になる。さらにこれらの作成のために、タグ規格第1.0版を改良していくことも必要かもしれない。

3. ソフトウェアタグ支援ツール

ソフトウェアタグ支援ツールは、ソフトウェアタグ規

格やソフトウェアタグ利用シナリオに準拠したデータ収集を容易にし、見える化や開発計画策定といったソフトウェア開発管理への応用を助ける道具である。図2にソフトウェアタグ実用化技術俯瞰図を示す。図に示す通り、支援ツールはタグ実用化サービス基盤の要素として位置付けられ、タグ規格やガイドラインと共に、ソフトウェアタグ実用のために重要な役割を果たす。

各ツールは機能的観点から、タグの生成のための基盤、及び運用のための基盤に大別されるが、本稿ではとくに重要である以下の2つのカテゴリに属するツールについて紹介する。

① タグデータ収集基盤：ソフトウェアタグの生成に必要なデータを収集する仕組を提供する

② タグ可視化・分析基盤：ソフトウェアタグを活用するために、その内容を可視化し、分析するための仕組を提供する

3.1　タグデータ収集基盤

タグデータ収集基盤には、開発プロジェクトで収集されデータを基にソフトウェアタグを作成するツールが含まれる。ここでは、タグデータの事前選定と計測計画立案ツール「タグ・プランナー」と実際にタグデータの収集を行うツール「CollectTag」について述べる。

3.1.1　タグ・プランナー

タグ・プランナーは、タグデータの収集計画立案を支援するツールで、プロジェクトマネージャ等の役割を持つ利用者が、システムの発注時等、開発着手前、データ定義の閲覧者収集計画の作成、調整等の作業を容易に行えることを目的としている。対象プロジェクトで作成するタグの内容をあらかじめ策定し、具体的な機能としてタグデータの構造や定義作成を支援し、可視化して表示する機能を持つ。

図3は、タグデータ定義画面の例である。対象プロジェクトの作業構造はWBS※9の形で画面左に表示されており、画面下部には、タグデータ項目が一覧表示されている。画面右側のエリアでは、個々のタグデータ定義の詳

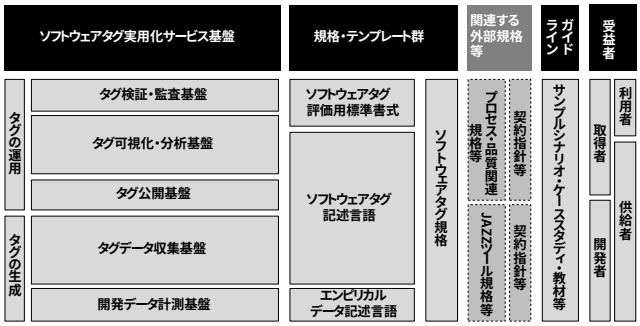

図2　ソフトウェアタグ実用化技術俯瞰図（簡略版）

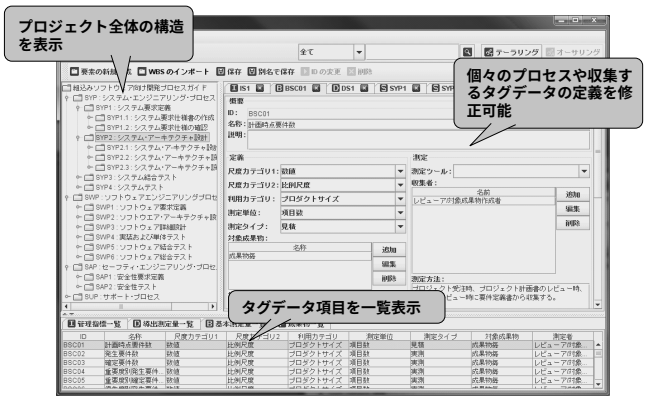

図3　タグデータ定義画面例

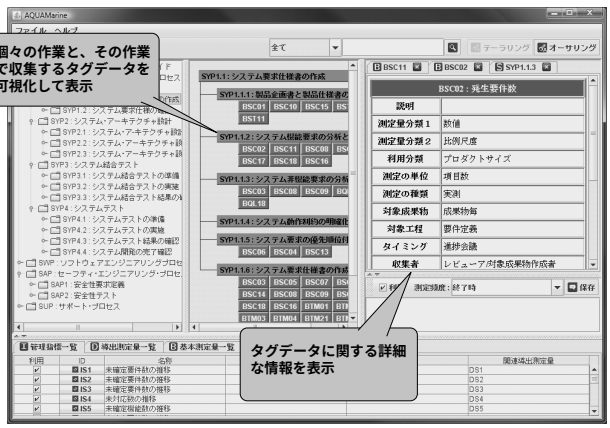

図4　策定したプロジェクト計画の閲覧画面例

細を閲覧し、編集を行うことが出来る。また、WBSやタグデータの構造自体をカスタマイズする機能も有する。本ツールで作成したタグデータ収集計画の内容は後述するXMLスキーマに従った形式で保存されるので、他のタグツールとの連携が可能である。

図4は、策定したプロジェクト計画中で扱われるタグ

※9　WBS：Work Breakdown Structure

データをWBS中の各作業からたどって閲覧する画面の例である。画面下部には対象プロジェクトで収集するタグデータの選択内容がチェックリスト形式で示されており、画面右には、各作業中で収集されるべきタグデータが表示されている。このように、タグ・プランナーで作成した収集計画は、ユーザ・ベンダ間でタグ収集項目を検討し、合意を取る際に利用可能であると共に、プロジェクト実行中のデータ収集やタグ生成の際のガイドラインとしても活用出来る。本ツールは、我々の開発したAQUAMarineツール[伏田2009]を基に試作を進めている。

3.1.2 Collect Tag（ソフトウェアタグデータ収集ツール）

本ツールは、開発プロジェクトで収集されている実証データを基にソフトウェアタグを実際に作成する。タグデータは、プロファイル情報と進捗情報に分けられる。プロファイル情報はプロジェクトを特徴付ける基本情報であるため、一度確定されるとほとんど修正が入らないと考えられる。一方、進捗情報は成果物や作業の進捗、成果物やプロセスの品質等を表すものであるため、適当な頻度で計測をする必要がある。以降では、進捗情報のデータ収集を中心に話を進める。

タグデータ収集ツールに求められる要求を次に示す。

① 汎用性が高い
② 低コストでタグデータが収集可能である
③ ソフトウェアタグを利用・分析しやすい形式で作成する

以降、これらに対する基本方針を述べる。

① 汎用性の要求

ソフトウェアタグの内容（個々のタグ項目に対応するメトリクス）は、契約時にユーザとベンダが協議して決めることになっているため、当然であるが個々のソフトウェアタグは特定のベンダの開発環境から収集されるデータより作成することになる。また、使用されるメトリクスの名前が同じであっても、ベンダ間で定義が異なる場合も多い（例えば、ソフトウェアの行数であっても、コメント行・空行を含むか含まないか等の違いがある）。従って、あらゆる実証データの収集やメトリクスの計測に対応することは難しい。そこで我々は、ソフトウェアタグ収集ツールを、ユーザとベンダが契約時に取り決めた個々のタグデータを適当なタイミングで入力し、その結果を標準タグデータフォーマットに変換するトランスレータと位置付ける。

② 低コストでの収集

タグ作成のために利用される実証データは今日のソフトウェア開発組織では開発管理を行う上で一般的に収集され、電子的に保存されているデータであると考えられるため、ソフトウェアタグ生成のためのデータ収集コストは比較的小さい。また、個々のメトリクスについても、ベンダは自社の定義に基づいて計測をしている。従って、①で述べた方法であっても、タグ作成のコストは少ないと考える。

しかし、すべてのタグ項目をベンダが入力することは現実的では無い。そこで、典型的な開発環境やメトリクスを利用する場合は、自動収集の機能を提供することを目指す。例えば、ソースコードがCVSやSubversionのような構成管理ツールで、またバグ情報がバグ管理ツールでそれぞれ管理されているような場合は、これらに依存したタグデータ項目は自動的に収集し、計測する機能を提供する。

③ ソフトウェアタグの出力

ソフトウェアタグデータは、XML形式で出力し、本プロジェクトあるいはツールベンダ等が開発する可視化・分析ツールとの効率的な連携を目指す。

3.1.3 プロトタイプ（Collect Tag）の開発

本ツールは現在、ソフトウェアタグ規格Ver.1.0に準拠して試作中である。図5に開発中のプロトタイプシステムの入力画面を示す。各タグ項目を順番に選べ、具体的なメトリクスを決めていく。

例えば、図5で「プログラミング」－「規模」を選択すると、図6の画面になる。規模のメトリクスとして「行数」と「ファンクションポイント」が選択可能になっており、「行数」を選ぶと図7の画面になる。この画面で、ツールが対応しているリポジトリから自動取得する場合は、必要な情報を入力する。ツールが対応していない場合は、「手動で入力」を選び、決められたタイミングで数

図5　画面例1（初期画面）

図6　画面例2（規模の設定）

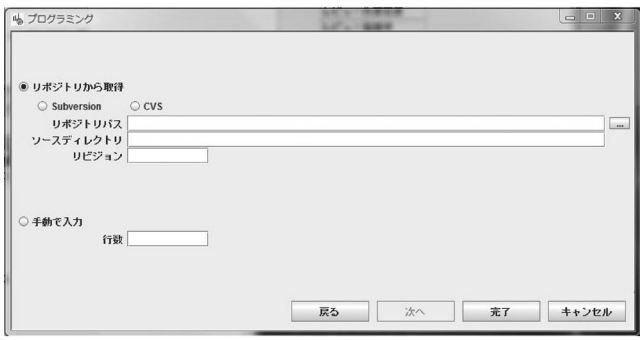

図7　画面例3（規模の自動収集・手動入力設定）

値を入力することになる。

個々のタグ項目に対しては、標準的なメトリクスを選択出来るようにすることを考えている。また、利用者が直接値を入力する項目については、その値の基となる実証データ名を合わせて入力する。

自動収集に対応していないメトリクスについては、システム利用者が自分でメトリクスの計測プログラムを作成すれば、収集システムが提供するAPIを通じて、収集方法の設定時に作成したメトリクス計測プログラムを指

表5　自動収集可能なタグデータ

分類	項番	タグ項目	タグデータ
要件定義	14	規模[推移]	ユースケース・アクターの数
	15	変更[推移]	追加，変更されたユースケース・アクターの数
設計	16	規模[推移]	UML図の数
	17	変更[推移]	追加，変更されたUML図の数
プログラミング	19	規模[推移]	コード行数
	20	変更[推移]	追加，変更されたコード行数
	21	複雑度	CKメトリクス
品質	29	欠陥件数[推移]	不具合数
	30	欠陥対応件数	不具合消化数
	31	欠陥対応密度	不具合数÷コード行数

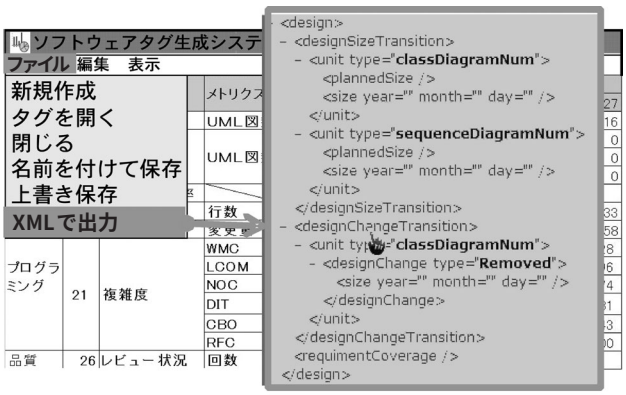

図8　XML形式への出力イメージ

定することでプラグインとして導入可能になる。

現在のプロトタイプでは、進捗情報の10項目を対象とし、表5に示すデータが自動収集可能になっている。

要件定義ではユースケース図の構成要素であるユースケース・アクターの数を収集する。設計では、UML図の数を抽出する。要件定義、設計共に計測対象はXMI形式で出力されたモデル図である。プログラミングでは、一般的に構成管理ツールで管理しているリポジトリよりデータを抽出する。現在、SubversionとCVSに対応している。複雑度メトリクスは、2.3節で述べた6つのメトリクス（CKメトリクス）を計測可能であるが、対象はJavaプログラムのみである。品質では、バグ管理ツールで収集されているバグデータから不具合数、不具合消化数等を計測する。

図8にタグデータのXML形式での出力イメージを示す。現在、分析・可視化ツール研究グループとの間でXMLによる標準タグ形式を検討中である。

今後の課題としては、自動収集するタグ項目の充実、

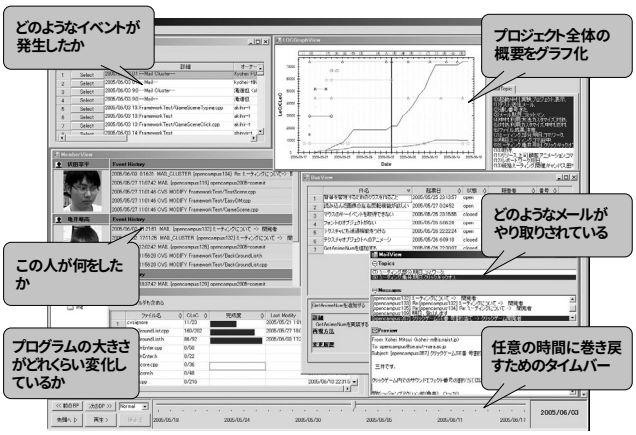

図9　タグ・リプレイヤーのメイン画面

タグ項目に対する基本メトリクスの設定やユーザビリティの向上が考えられる。

3.2　ソフトウェアタグ可視化・分析基盤

ソフトウェアタグ可視化・分析基盤にはソフトウェアタグデータ収集ツールに等により生成されたソフトウェアタグの内容を、利用シーンに応じて適切に提示するツール等が含まれる。タグに基づいた分析を行うに際しては、ソフトウェア信頼性予測モデルに代表されるような、様々な既存モデルの活用が想定されるが、既存の分析技術を一つひとつ基盤ツールとして実装するのはリソースの制約上非現実的である。従って、StagEプロジェクト期間中のゴールとしては以下の2点を設定している。

① タグの内容を元にプロジェクト推移に関する基本情報を再構成して分かりやすく提示する基盤機能を開発する

② 幾つかの分析手法について①の機能を使ってサンプル的な実装を示す。

現在は①の基本可視化ツールを中心に開発を進めている。ソフトウェアタグは進捗情報にかかわるデータを数多く含むので、これを中心とした可視化を考える。すなわち、時系列による推移を基本軸として、成果物、作業、組織等の視点による粒度を、可視化の用途に応じて変化させて提示出来る仕組みを、可視化の基盤として提供する。

表6　現在開発中のソフトウェアタグ活用支援ツール

ツール名	機能・用途
タグ・プランナー	タグ計測・利用計画策定支援
CollectTag	実証データからのタグ生成
タグ・リプレイヤー	タグデータを利用したプロジェクト可視化
タグ・シミュレータ	開発プロセスをシミュレートし、結果をタグとして出力
EPM/Core（仮称）	個人作業環境での実証データ収集

以降では現在プロトタイプ開発中の可視化・分析ツールとしてタグリプレイヤーを紹介する。

3.2.1　タグ・リプレイヤー

タグ・リプレイヤーは、タグデータに含まれる様々な進捗情報を複数の表示形式で再現する統合的な可視化ツールである。“ある時点”でのプロジェクトの状態を、録画ビデオを再生するように提示することが出来るため（図9）、プロジェクトのレビュー（事後分析）等に有効である。

(1) タグデータ可視化機能

タグリプレイヤーは、ソフトウェアタグデータ中に含まれる進捗に関する情報をプロジェクトにおいて発生したイベントとして捉え、時系列に沿って網羅的に表示することが出来る。サマリ的な情報としては、プロジェクト中で計測された各種データをグラフとして重畳表示することが出来る。また、分析のためにプロジェクトをさらに深く閲覧する機能として、開発者間の対話や障害報告等のテキスト情報をマイニングし、表示する機能も備える。

(2) プロトタイプの開発

タグ・リプレイヤーはEASE※10プロジェクトで開発された「プロジェクト・リプレイヤー」[OHKURA2006]を元に、ソフトウェアタグ規格1.0の様々なデータ項目への対応や、XML形式のタグ情報の読み込み、ソフトウェアメトリクスのグラフ表示、テキスト情報のマイニング機能

※10　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

等、大幅に機能を追加して開発中である。

3.4　その他のツールと全体のフレームワーク

誌面の都合上、本稿で紹介出来なかったツールを含め、現在開発を進めている支援ツールの一覧を表6に示す。これらのツールは来年度の適用実験を通じて評価され、最終的にはソフトウェアタグ支援基盤サービス群のリファレンス実装として公開予定である。

なお、図2に示したように、ソフトウェアタグ活用技術としては、タグデータの収集と可視化・分析の他に、タグの公開・運用や検証・監査のための基盤サービス等が必要である。これらについては今後開発を行っていく予定であるが、タグデータの閲覧制御や改ざん防止等、セキュリティ管理に相当する仕組みについては、現在のプロジェクト期間中には利用シナリオ等を通じたサービス仕様の策定までを行う予定である。

ソフトウェアタグ規格、及び具体的なタグデータの記述言語仕様は、これら基盤サービス群に共通して準拠すべき要素である。ソフトウェアタグの重要な目的の一つはシステム開発プロジェクト中で計測されるデータの構成を一定の自由度を保ちつつ、標準化することである。従って、一連のソフトウェアタグ支援ツールにおいても、流通するタグのデータ形式は一定の抽象度において（つまり、ソフトウェアタグ規格1.0相当の粒度において）互換性を持って取り扱われることが大前提である。

StagEでは、データ収集及び可視化・分析ツールの予備設計を通じて、XMLによる具体的なソフトウェアタグ記述用のスキーマ（仮称、StagE/ML）のドラフトを定め、標準エンピリカル形式の開発データや代表的な開発支援ツールのリポジトリ形式のデータからStagE/MLへの変換ライブラリ等の整備を進めている。今後、言語仕様の規格化・公開を行う予定である。また、プロジェクト外の開発ツール群との連携についても検討を進めており、例えば、IBM Rationalで開発中のチーム開発支援ツール群Jazzで収集されたデータによるソフトウェアタグの生成の可能性についても検討している。

4. おわりに

本稿ではStagEプロジェクトにおけるソフトウェアタグ規格化の経緯とその内容、及び、ソフトウェアタグ利用支援ツール群の設計と試作の現状について紹介を行った。

初年度及び2年目までの期間を通じて、まずはタグ規格の策定とそれを活用するツール群の概要設計を行ってきた。現在はこれらの成果を受けてツール群のより具体的な設計と試作を行っている段階である。このように、理論と実践、規格と実装の間を短期間でフィードバックするアプローチ自体もStagEプロジェクトの特徴であると考えている。

タグ規格については、今後、ツールやサンプルシナリオの開発、実証実験等の結果を基に、規格の内容自体を見直したり、タグ規格本体ではあえて規定していない具体的情報の補足や実践のためのガイドライン作成等も検討している。

現在開発中のタグ実用化支援ツール群は、本プロジェクトの運用・検証担当班で作成が進んでいる「ソフトウェアタグ利用シナリオ」において有効に活用されるように設計が行われている。また、これらのシナリオに基づいたソフトウェアタグ適用実験においては、これらの試作ツール群の有効性・可用性についても検証を行っていく予定である。

次回は、ソフトウェアタグの利用シナリオ・普及に向けての取り組み、法的意義について紹介する予定である。

参考文献

[OHKURA2006] Ohkura, et al.：Project Replayer with Email Analysis -- Revealing Contexts in Software Development，In Proceedings of the 13th Asia Pacific Software Engineering Conference（APSEC06），pp. 453-460，December 2006
[StagE2008]　StagEプロジェクトーソフトウェアタグ規格技術委員会：ソフトウェアタグ規格　第1.0版，2008年10月14日，http://www.stage-project.jp/seika_dl.php
[伏田2009] 伏田　他：AQUAMarine：定量的管理計画立案システム，SEC journal, to appear，2009
[松本2009] 松本健一：事故前提社会に向けたユーザ・ベンダ間での開発データ共有ーStagEプロジェクトとソフトウェアタグー，SEC journal, pp.198-203，2009

事故前提社会に向けたユーザ・ベンダ間での開発データ共有 第3回
―ソフトウェアタグ普及に向けた法的議論と利用技術基盤―

奈良先端科学技術大学院大学
先端科学技術研究調査センター
教授
久保　浩三

奈良先端科学技術大学院大学
先端科学技術研究調査センター
研究員
小柴　昌也

奈良先端科学技術大学院大学
情報科学研究科
特任助教
角田　雅照

奈良先端科学技術大学院大学
情報科学研究科
研究員
松村　知子

この解説では、文部科学省StagEプロジェクト※1の概要を紹介すると共に、同プロジェクトが作成したソフトウェアタグ規格と、ソフトウェアタグ支援ツールについて述べてきた。最終回となる今回は、ソフトウェアタグ普及に向けたより実践的な取り組みとして、ソフトウェアタグの意義と技術的課題についての法的観点からの議論、及び、ソフトウェアタグ利用の技術基盤となるユーザ・ベンダ協調型プロジェクト管理とソフトウェア開発データ分析モデリング言語について解説する。

1. はじめに

ソフトウェアタグとは、ソフトウェア開発に関する実証データから、ソフトウェアやその開発プロジェクトの特徴量を算出し、ユーザにも理解しやすく、可視化や評価にも利用しやすい形式でとりまとめた情報パッケージである。ソフトウェア開発終了後にソフトウェア製品に添付され、ベンダからユーザへ提供されることになるが、開発途中に進捗報告書に添付するといった利用形態も考えられる。ユーザ・ベンダ間でのデータ共有のメディアとも言えるが、実開発プロジェクトに適用するためには、開発データ共有の目的や目標といった抽象的な概念を、ソフトウェアタグ規格で規定された具体的なデータに対応付けたり、タグ収集・可視化・評価ツールを活用したりするためのより実践的な技術が必要となってくる。

本技術解説では、ソフトウェアタグ普及に向けた法的観点からの議論と2つの利用技術基盤について概説する。これらは、主にソフトウェアタグ利用シナリオとして具現化することになるが、ソフトウェアタグ規格やソフトウェアタグ支援ツールにも今後反映されていくものである（図1）。

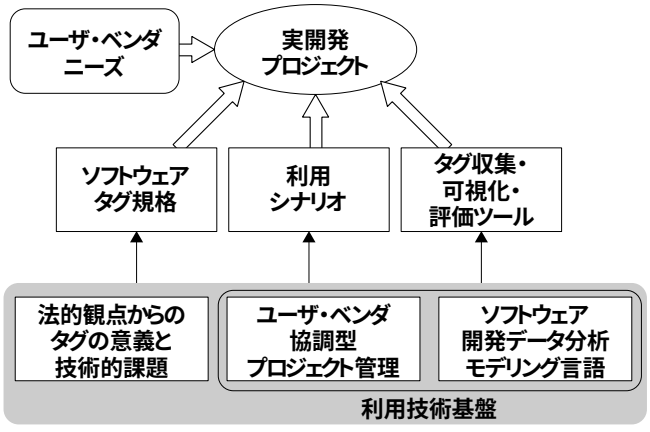

図1　ソフトウェアタグ技術を支える法的観点からの議論と利用技術基盤

2. ソフトウェアタグに対する法的議論

2.1　目的と背景

ソフトウェア開発を成功させるためには、要件定義、レビュー等、すべての段階で注意が必要であり、ユーザ

※1　StagEプロジェクト：Software Traceability and Accountability for Global software Engineering：エンピリカルデータに基づくソフトウェアタグ技術の開発と普及

とベンダの双方において、契約の明確化、プロセスの可視化、人材育成等のいろいろな試みが行われている。しかし、ソフトウェア開発において、品質、コスト、納期のすべてにおいてユーザ・ベンダ間での紛争が多発しており、現状においては減少傾向にはなく、問題がますます複雑化してきているようにも見受けられる。

ソフトウェアタグに対する法的観点からの議論の目的は、ソフトウェア不具合時の事後的な紛争処理を分析することで、紛争を未然に防止することにある。また、ソフトウェアタグを、プロジェクト管理に用いるだけでなく、ユーザとベンダで共有し、これを活用することによって、紛争処理に係る労力を軽減することも目的の1つである。

2.2 技術と法律の連携

ここで紹介する法的議論は、ソフトウェアタグに関する技術的議論との連携を目指したものである。例えば、後述するように、ソフトウェア開発における紛争の分析結果に基づいて、ソフトウェアタグへの要望をまとめている。技術と法律の連携を深めるため、法学部教授、弁護士等の法律の専門家、そして、ソフトウェアタグの技術的専門家の双方で構成される「ソフトウェア構築可視化に伴う法的諸問題委員会」を設置している。

このような技術と法律による連携は、一般的に見られるものではない。確かに、これまでも、建設に係る法、通信に係る法、情報と法、原子力と法等にも技術と法律の連携は見られる。しかし、これらは、新たな技術を社会に適用する場合に生ずる種々の問題を調整するために、法律を制定し、その交通整理を行うことが主である。

今回用いた手法は、ソフトウェア開発における紛争を紹介、分析することで法律家が関与し、新たな技術課題解決による新技術の開発手法の提案を行うという新規なものである（図2）。この手法が、他の技術分野でも応用出来るかどうかの検討も今後行っていく。

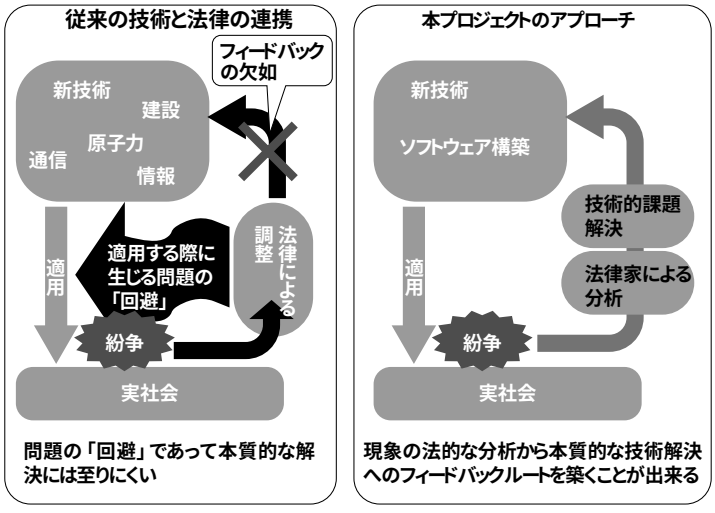

図2　法的分析から技術的課題解決へのループ形成

2.3 具体的な取り組み

2.3.1 裁判例の抽出と分析

分析対象としたのは、平成2年以後のソフトウェア開発委託取引を巡る20件の裁判例である。裁判事例は、TKC法律情報データベースにおいて、以下のキーワードにより抽出した。

（瑕疵（かし）＋仕様＋開発）×プログラム＝41件
（瑕疵＋仕様＋開発）×ソフトウェア＝26件
（瑕疵＋仕様＋開発）×ハードウェア＝13件

ここから知的財産権に係る紛争を除外し、さらなる調査により4件を追加した。

これらの裁判例を分析すると、ユーザのシステム開発要求をベンダに伝えることの困難さに紛争の原因の1つがあることが分かった。通常、ソフトウェア開発は、RFP※2により、ユーザの要求がベンダに伝えられる。また、RFPを更に具体化した要求仕様書が用いられることもあり、これに基づいてベンダが要件定義を作成する。しかし、ユーザの要求をすべて伝えることは容易ではない。

また、ユーザからベンダに対して、開発工程の段階でも仕様変更やユーザの要求が提示されることが多く、当初よりも見積りが大きく膨れ上がり、また納期も延びることとなる。ソフトウェア開発

※2　RFP：Request For Proposal，提案依頼書

委託契約は、いわゆる請負型で行われることが多く、ユーザは、自分の要求が満たされないと、業務が完成していないとして支払いを拒むケースがあり、紛争の元となっているように見受けられた。

2.3.2 アンケート調査

以上のように、ソフトウェア開発委託取引を巡る裁判事例は、非常に少ない。そこで、判例分析で導き出された観点である「ユーザの要求がベンダに正確に伝わっているか」という点についてアンケート調査を行った。調査では、ソフトウェア開発費の見積額と開発に実際に要した経費との差異（見積り差異）に着目した。2008年12月から2009年3月にかけて、インターネットも活用したアンケートの結果、ユーザ91名、ベンダ224名、合計315名から回答を受け取った。得られた主な結果は次の通りである。

- ベンダの回答によると、ソフトウェア開発における見積り差異は、人月計算で1.8倍、金額計算で2倍であった
- 見積り差異が生じたプロジェクトでの見積り方法は、「大まかな業務の説明」に基づくものがベンダ41%（最も多い原因）、ユーザ24%（2番目に多い原因）であった（図3）
- 見積り差異の発見段階は、ベンダでは「詳細設計」（21%）を筆頭に、「基本設計」（19%）、「要件定義」（13%）という意見が多く、プログラミングに着手するまでの早い段階で問題が発覚していることが多い（図4）

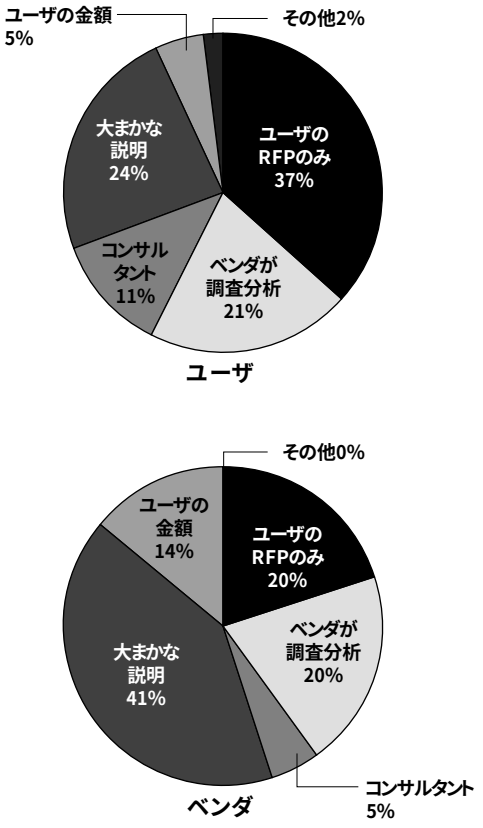

図3　見積り差異が生じたプロジェクトでの見積り方法

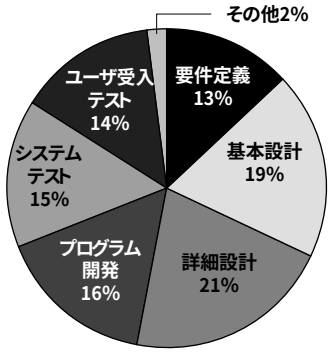

図4　見積り差異の発見段階

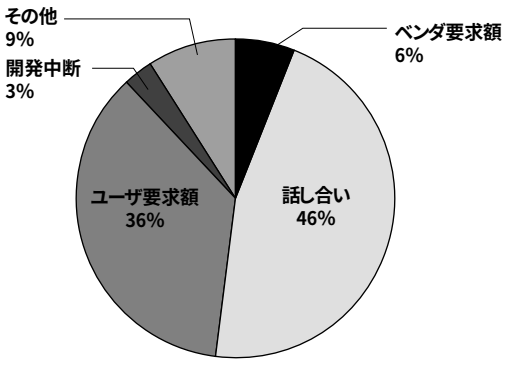

図5　見積り差異の解決手段

・見積り差異の原因は、ユーザ・ベンダ共に「現場からの予想外の要求」（ユーザ30%、ベンダ27%）が、最も多い
・見積り差異の解決手段は、「話し合い」（46%）を除き、ベンダが「契約通りの金額しか請求しなかった」（36%）が最も多かった（図5）

このように、アンケート全体を通じて、ユーザとベンダ間には意見の隔たりがあることが分かった。また、見積り差異が生じる原因として、あいまいな受発注による契約が根底にあることも分かった。

2.4　ソフトウェアタグの意義と技術的課題

裁判例等の紛争やアンケート回答を分析していくと、ソフトウェア開発における紛争は大きく2つの型に分類出来ることが分かった。そして、法的議論の1つの結果として、それら2つの型それぞれにおけるソフトウェアタグの意義と技術的課題（技術的議論への要望）が明らかとなった。詳しくは表1を参照されたい。

ソフトウェアタグの内容をどのように要求するかは、契約書の一部分として定めることであり、例えば、段階的な請負契約をすることを業界標準として定めたときに、その段階ごとに、当事者が内容を確定するために、どのようなソフトウェアタグを必要とするかを検討する必要がある。今後は、当事者自身が、ソフトウェアタグをどのように活用していくかの観点から契約とソフトウェアタグについて法的議論を深めていく予定である。

3. ユーザ・ベンダ協調型プロジェクト管理

本章では、ユーザ・ベンダにWin-Winの開発管理形態を実現するための枠組みの2つのポイント、ユーザ・ベンダ協調型プロジェクト管理サイクルと、このサイクルを実現するための管理プロセスの構築手順について説明する。前述した法的紛争の事例や現場アンケートからも、ユーザのプロジェクト管理における責任や役割の重要性は明確であるが、既存のソフトウェア開発の定量的管理の規格やガイドライン［CMMI2007］［PMBOK2004］は、いずれも開発組織の視点で作成され、ユーザは管理の対象の一部として扱われている。紹介する枠組みは、双方にとって透明性の高い管理プロセスを実現し、問題の早期発見・解決を支援する。

表1　ソフトウェア構築に関する法的諸問題の分析

紛争の型	法的責任	ソフトウェアタグの意義	ソフトウェアタグの技術的課題	今後の課題
未完成または欠陥により使用不可	ユーザまたはベンダの債務不履行責任	・ユーザに進捗を見せ、相互に管理を行う。 ・開発が順調に進んでいることをユーザに知らせる。 ・開発が順調に進んでいない場合の軌道修正を容易にする。	・ユーザの要望（要求定義）がベンダに伝わっているかどうかを明らかにすること。 ・実績が予定通り進んでいること（予実管理）を行えるようにすること。	・実証実験により、基準値、標準曲線を作成し、ユーザの視認性を高めること（例：人間ドックにおける基準値）。
使用可ではあるが、完成後のバグ発生または障害発生	ベンダの不法行為責任または債務不履行責任	・事故があった場合のトレースを容易にする。 ・早期復旧を容易にする。	・システムがダウンする前に、どこにバグまたは欠陥があるかのおおよその当たりをつけること。	・システムがダウンした場合の原因、修正履歴を残し、障害の発生率を下げることが出来るようにすること。

3.1 協調型プロジェクト管理サイクル

図6はPMBOK［PMBOK2004］に基づく従来のプロジェクト管理サイクルである。ユーザは、サイクル外に位置付けられ、ベンダから提供されるデータに対してチェックや要望を出す。これに対して、図7の我々が提案する協調型プロジェクト管理サイクルでは、計画・監視・コントロールのプロセスに関して、両者の合意・相互確認・各組織での合意に基づいた適切な是正処置を行うことを要求する。

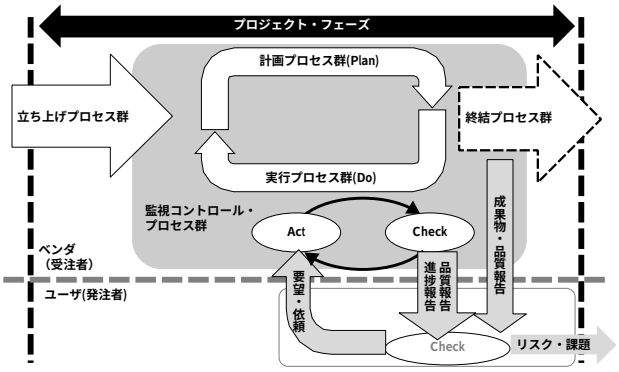

図6　従来のプロジェクト管理サイクル

3.2 管理プロセスの構築手順

次に、合意形成プロセスについて、我々はCMMI［CMMI2007］のプロセス領域「定量的プロジェクト管理」を参考に以下のような定量的プロジェクト管理プロセスの構築手順を提案する（図8）。

① プロジェクト管理目標を設定し、両者で合意
② ユーザ・ベンダ別に、目標に関わる作業（サブプロセス）を定義
③ 作業でユーザ・ベンダ間のコミュニケーションや共有される（べき）データを明確化
④ ②③に基づき、定量的に管理するフェーズを構成
⑤ 管理フェーズごとに定量的管理に関する目標、尺度、データの収集と格納方法、分析方法等で測定モデルを作成。このステップについては、前号で紹介したタグデータの事前選定と計測計画立案ツール「タグ・プランナー」を適用可能

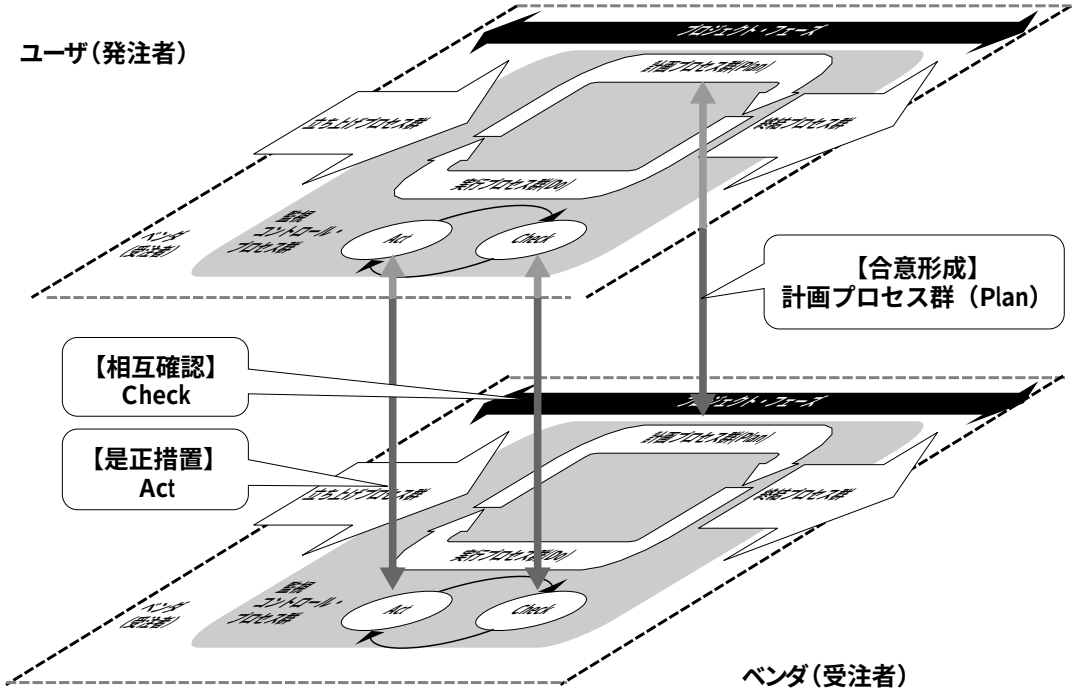

図7　提案する協調型プロジェクト管理サイクル

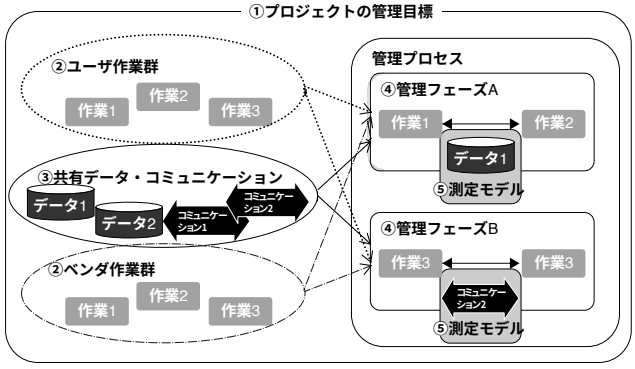

図8　定量的プロジェクト管理プロセスの構築手順

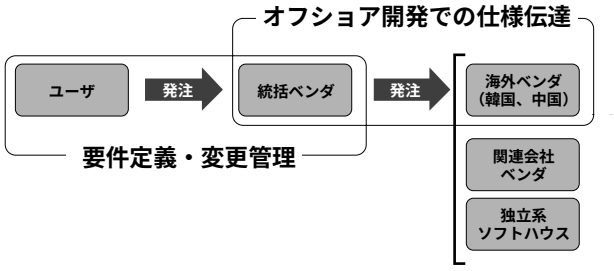

図9　実プロジェクトでの検証

3.3　実プロジェクト開発情報に基づく検証

上記の枠組みについて、数社から実プロジェクトのデータをご提供頂き、定量的データ分析とその結果に基づいた開発関係者へのヒアリングを繰り返し、管理プロセスの構築と実用性の検証を行った（詳細は、[松村2009-1][松村2009-2]）。図9は検証に用いた2つのプロジェクト体制（プロジェクトは異なる）と、おのおののプロジェクトの管理目標を示す。対象プロジェクトでは、データ共有やコミュニケーションは十分に行われていて、既に定量的な管理に活用出来る基盤があり、実際に第3者（StagEスタッフ）でも既存データの定量的分析からプロジェクトの状況を把握出来ることを確認した。また、検証過程で、実プロジェクトへの適用のための幾つかの要件が抽出され、次章のモデリング言語へも反映された。

4.　ソフトウェア開発データ分析モデリング言語

4.1　要件と構造

ソフトウェア開発データ分析モデリング言語SDAML※3とは、ソフトウェア開発における定量的データ計測・分析の活動を記述するための記法である［角田2009］。SDAMLの利点は以下の3つである。

・データ分析方法のノウハウの共有が容易になる
・データ分析の目的と方法を明確に示すことが出来るため、ユーザとベンダ間でのデータ共有に関する合意がしやすくなる
・テンプレートの機能を果たすため、全くテンプレートが無い状態に比べ、記述するコストが低い

SDAMLの役割は、直感的にはソフトウェアの仕様記述法であるUML※4の役割に、プログラム設計パターンのカタログであるデザインパターンの役割を加えたものであると考えると分かりやすい。

SDAMLは、データ計測・分析プロセスを記述するために、次の8つの要件を満たすように定義されている。

（R1）　データ分析の目的と、目的を果たすために着目すべきデータ項目を記述出来ること
（R2）　各データ項目の計測方法を記述出来ること
（R3）　各データ項目の分析方法を記述出来ること
（R4）　分析結果に基づいた、実施すべき対応策を記述出来ること
（R5）　データ計測・分析・共有の手順やタイミングを記

※3　SDAML：Software development Data Analysis Modeling Language
※4　UML：Unified Modeling Language

述出来ること

（R6） ユーザ、ベンダそれぞれの分析目的を個別に記述出来ること

（R7） プロジェクト進行中、及びプロジェクト完了時のデータ分析方法を区別して記述出来ること

（R8） データ分析モデルの適用条件を記述出来ること

（R1）～（R5）は、一般的なデータ計測・分析プロセスの活動を記述するための要件である。（R1）はデータ収集計画立案活動（PDCA※5サイクルのPlan）を記述するための要件、（R2）はデータ計測活動（PDCAサイクルのDo）を記述するための要件、（R3）はデータ分析活動（PDCAサイクルのCheck）、（R4）は対応策実施活動（PDCAサイクルのAct）を記述するための要件、（R5）はデータ計測・分析プロセスのフロー（PDCAサイクルのフロー）を記述するための要件である。

（R6）はユーザとベンダでデータを共有する場合に必要となる要件である。ユーザとベンダのデータ分析目的は完全に一致しない場合がある。例えば、「ソフトウェアの出荷後の欠陥数を抑える」はユーザ、ベンダ共通の分析目的となるが、「テスト効率を高める」は、（請負開発の場合）ベンダのみの分析目的としかならず、それぞれを明確に区別して記述する必要がある。

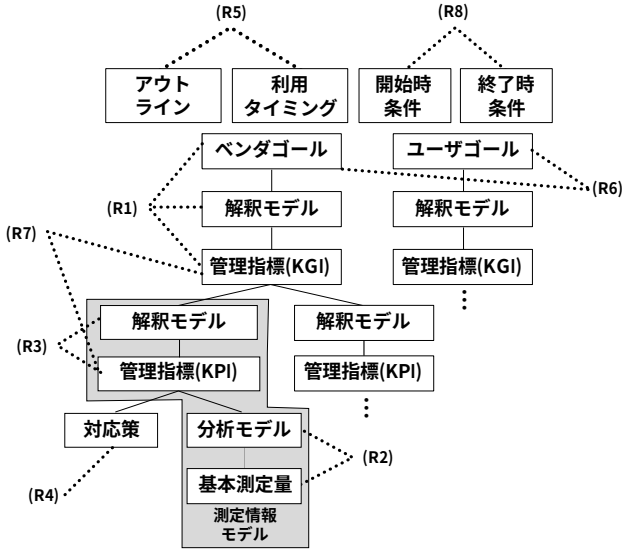

図10　SDAMLの構造と要件との関係

（R7）はユーザとベンダでデータを共有し、かつプロジェクト進行中とプロジェクト完了時の両方でデータ分析を行う場合に必要となる要件である。例えば、「高い品質のソフトウェアを開発する」ことを目的として、「出荷後欠陥密度」に着目して分析しても、プロジェクト進行中に目的が達成出来そうかを判断することが出来ない。逆に、同じ目的で、プロジェクト進行中に「結合テスト欠陥密度」と「システムテスト欠陥密度」に着目して分析することを決めても、最終的に目的が達成されたかを判断する基準を決めていなければ、ユーザとベンダで目的が達成されたかどうかの認識が一致しなくなる可能性がある。

（R8）はデータ分析モデルをカタログ化する際に必要となる要件である。過去のプロジェクトで作成されたデータ分析モデルを、新たなプロジェクトに適用する際、誤った評価をすること避けるためには、「どのようなプロジェクトに適用出来るのか」、「プロジェクト実施中にどういったことが起こると正しい評価が行えないのか」等が明らかになっている必要がある。

SDAMLは、ISO/IEC 15939で定義されている測定情報モデル［ISO/IEC 15939］をベースにした記述法であり、測定情報モデルと同様に、階層構造を持ったモデルである。測定情報モデルとは、データの計測から分析方法までを階層構造により表したモデルである。図10にSDAMLの構造を示す。図では各要素と要件との対応関係、及び測定情報モデルに該当する部分を示している。

次節では、SDAMLの構成要素について説明する（誌面の都合上、主要な構成要素のみ説明する）。

4.2　主要構成要素

（1）アウトライン、利用タイミング

アウトライン、利用タイミングは、「データ計測・分析・共有の手順やタイミング（R5）」を記述するための要素である。アウトラインでは、利用シナリオ（分析モ

※5　PDCA：Plan Do Check Action

デル）の適用場面、ユーザ/ベンダの要求、データ計測、分析手順を具体的に記述する。要求分析におけるユースケースと類似した内容を記述すると考えるとよい。利用タイミングは、データの計測、データのユーザへの受け渡し、データ分析のタイミングを示した図であり、UMLのアクティビティ図を使って表記する。

(2) 基本測定量、分析モデル、管理指標、解釈モデル

基本測定量、分析モデル、管理指標、解釈モデルは「各データ項目の計測方法（R2)」と「各データ項目の分析方法（R3)」を記述するための要素であり、ISO/IEC 15939で定義されている測定情報モデルに基づいている。各要素の役割は以下の通りである。

- 基本測定量：測定対象から直接計測される数値
- 分析モデル：基本測定量に基づいて管理指標を計算する方法を示したモデル
- 管理指標：分析モデルに基本測定量を与えることにより求められる数値
- 解釈モデル：管理指標の分析方法を示したモデル

(3) ユーザゴール、ベンダゴール

ゴールは、「データ分析の目的と、目的を果たすために着目すべきデータ項目（R1)」を記述するための要素であり、GQMパラダイム［BASILI1984］のゴールの概念に基づいている。GQMパラダイムとは、データ収集の目標設定からデータ収集のメトリクス決定までをモデル化したものであり、ゴールとは、計測の目標、計測対象、計測理由等を明確にした文である。ゴールと管理指標を対応付けて記述する。

ゴールにはユーザゴールとベンダゴールの2種類が存在する。これらは、「ユーザ、ベンダそれぞれの分析目的を個別に記述（R6)」するための要素である。

(4) KGI※6、KPI※7

管理指標にはKGIとKPIの2種類が存在し、ゴール、

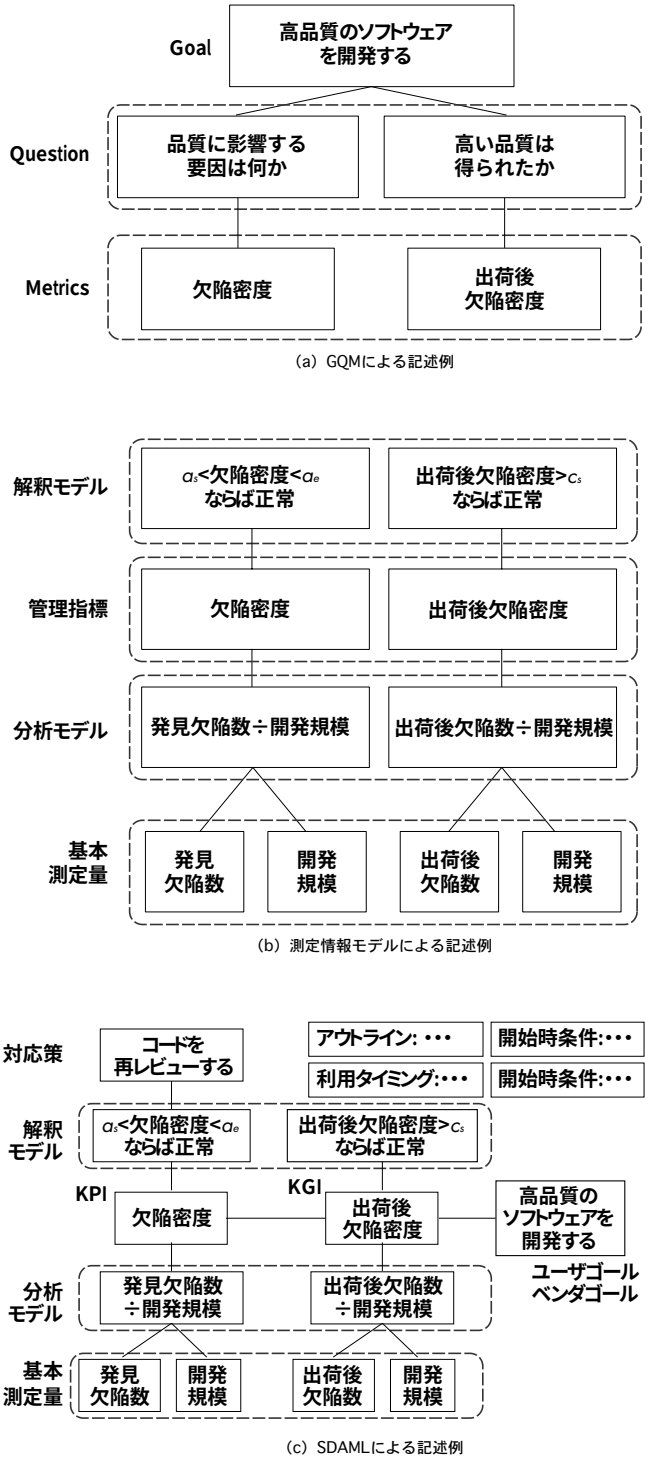

図11　モデル記述例

※6　KGI：Key Goal Indicator，重要目標達成指標
※7　KPI：Key Performance Indicator，重要業績評価指標

KGI、KPIは階層構造となっている。これにより、「プロジェクト進行中、及びプロジェクト完了時のデータ分析方法 (R7)」を記述出来るようになっている。KGIとKPIはビジネスマネジメントの分野で用いられている概念であり、KGIはゴールを達成したか否かを判断するための指標、KPIはプロジェクト進行中にプロセスを評価するための指標である。KPIが目標値をクリアするようにプロジェクトを遂行することにより、KGIも目標値をクリア出来るような関係となる。なお、(「テスト工程の進捗を把握する」等) 進捗を把握することが目的のゴール場合は、KGIを設定しない。それ以外では、まずゴールに対して1つのKGIを設定し、KGIに対して1つ以上のKPIを設定する。

4.3 SDAMLと従来法によるモデル記述例

GQM、測定情報モデル、SDAMLそれぞれを用いてモデルを記述した例を図11に示す。GQMは、測定情報モデル、SDAMLに比べ、記述出来る情報量が少ないことが分かる。SDAMLと測定情報モデルで記述出来る情報量の差は大きくはない。しかし、測定情報モデルの場合、ユーザゴール・ベンダゴール、KGI、KPIの関係を表すことが出来ないため、管理指標を何のために利用するのかが非常に分かりにくい。更に、測定情報モデルでは、開始時条件、終了時条件を記述出来ないため、データ分析モデルをカタログ化することが出来ない。

このように、SDAMLを用いてモデルを記述することにより、データ分析の目的と方法を明確に示すことが出来る。また、テンプレートの機能を果たすため、GQMや測定情報モデルを用いるよりも、モデル記述が容易となる。更に、データ分析モデルをカタログ化することが出来る。

5. おわりに

本技術解説では、文部科学省StagEプロジェクトの活動とこれまでに得られた主な成果を3回にわたって紹介してきた。2007年8月から5年計画の始まった同プロジェクトは、ちょうど折り返し地点を迎えたことになる。先日開催したプロジェクト主催の研究会での議論等を拝聴していると、開発データの共有に対するユーザ・ベンダの関心は、プロジェクト開始当初に比べて高まっており、ソフトウェアタグの具体的な利用イメージも明確になりつつある。ソフトウェア開発管理力に自信のあるユーザ・ベンダにとって、開発データの共有によるデメリットは少なく、その一方で、メリットを拡大する余地があると考えられ始めているのかもしれない。

現在のところ、同様の取り組みは海外では見られない。IPA/SECによるプロジェクトベンチマーキングや経済産業省によるソフトウェアメトリックス高度化の取り組み等と連携をとり、そして何より、ユーザ・ベンダ企業との連携の輪を広げることで、日本独自のソフトウェア技術としてソフトウェアタグの研究開発を推進していく予定である。なお、ソフトウェアタグ規格を始めとして、本技術解説で取り上げた取り組みの詳細やその他の研究成果についてはStagEプロジェクトのウェブページ［STAGE Web］にて公開中である。参照いただければ幸甚である。最後に、技術解説の機会を頂いた、SEC journal編集委員会に心から感謝致します。

参考文献

［AMBLER1984］ Ambler, S.W.：Process Patterns：Building Large-Scale Systems Using Object Technology, Cambridge University Press, 1998
［BASILI1984］ Basili, V. and Weiss, D.：A Methodology for Collecting Valid Software Engineering Data, IEEE Trans. On Software Eng., vol.10, No.3, pp.728-738, 1984
［CMMI2007］ CMMI 成果物チーム：開発のためのCMMI （CMMI-DEV）1.2版公式日本語翻訳版，技術報告書 CMU/SEI-2006-TR-008，2007
［ISO/IEC 15939］ International Organization for Standardization：ISO/IEC 15939:2002, Software Engineering - Software Measurement Process, International Organization for Standardization, Geneva, Switzerland, 2002
［PMBOK2004］ Project Management Institute：プロジェクトマネジメント知識体系ガイド（第3版）PMBOKガイド，2004
［STAGE Web］ http://www.stage-project.jp/
［角田2009］ 角田雅照，松村知子，松本健一：ソフトウェア開発データ分析モデリング言語の提案，ソフトウェアエンジニアリングシンポジウム2009 併設ワークショップ 「ソフトウェア開発マネジメントのための測定と分析」，2009年9月
［松村2009-1］ 松村知子，大平雅雄，森崎修司，松本健一：オフショア開発におけるユーザ・ベンダ間コミュニケーション情報の分析による仕様伝達の評価，奈良先端科学技術大学院大学情報科学研究科テクニカルレポート，NAIST-IS-TR2009005，2009年10月
［松村2009-2］ 松村知子，松本健一：ユーザとベンダ間の協調による要求品質確保のための定量化事例，奈良先端科学技術大学院大学情報科学研究科テクニカルレポート，NAIST-IS-TR2009006，2009年11月

Part2. ソフトウェアタグ技術解説・英語編

“Standardizing the *Software Tag* in Japan for Transparency of Development”

Reprint of NAIST Technical Report 2010001

(http://isw3.naist.jp/IS/TechReport/report/2010001.pdf)

Standardizing the *Software Tag* in Japan for Transparency of Development

Masateru Tsunoda†, Tomoko Matsumura†, Hajimu Iida†, Kozo Kubo‡,
Shinji Kusumoto††, Katsuro Inoue††, Ken-ichi Matsumoto†

†Graduate School of Information Science, Nara Institute of Science and Technology
{masate-t@is, tomoko-m@is, iida@itc, matumoto@is}.naist.jp

‡Research Center for Advanced Science and Technology, Nara Institute of Science and Technology; kubo@rsc.naist.jp

††Graduate School of Information Science and Technology, Osaka University; {kusumoto, inoue}@ist.osaka-u.ac.jp

ABSTRACT
In this paper, we describe the *Software Tag* which makes software development visible to software purchasers (users). A software tag is a partial set of empirical data about a software development project shared between the purchaser and developer. The purchaser uses the software tag to evaluate the software project, allowing them to recognize the quality level of the processes and products involved. With Japanese government support, we have successfully standardized the software tag named *Software Tag Standard* 1.0, and have developed various associated tools for tag data collection and visualization. For its initial evaluation, the software tag has been applied to several projects. This paper also presents various activities aimed at promoting the use of the software tag in Japan and the world.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics.*

General Terms
Management, Measurement.

Keywords
Information sharing, empirical data, project management, offshore development.

1. INTRODUCTION
Software systems are becoming huge and complex, with our everyday life heavily dependent on such software systems. One of the major concerns of software purchasers (users) in Japan is the quality of the software systems. Japanese society generally demands high-quality software systems with low fault rates and high operability levels.

On the other hand, many software purchasers in Japan are not knowledgeable about the nature of software. It is reported that only 40% of Japanese major companies employ a full-time Chief Information Officer (CIO) and that only 20% of all CIOs are confident of their knowledge about information technologies [7].

Without a sufficient understanding of software quality and software projects, many companies try to purchase software systems from software developers (vendors). This produces a very risky situation. For example, purchasers cannot specify system requirements very well, and they do not oversee the project properly. Such situations often lead to project failures. It is reported that only 31.1% of software projects are recognized as 'successful projects' in Japan [8]. To confront these issues, there is strong demand to provide transparency of software projects to the software purchaser and improve communications between purchaser and developer.

The *Software Tag* is a new scheme to provide information feedback about the project from the developer to the purchaser. It establishes transparency of the software development project by allowing purchasers to view and analyze the elements of the tag. It also provides support for quantitative and qualitative communications between stakeholders. The Software Traceability and Accountability for Global Software Engineering (*StagE*) project [1] is a government-supported project that pursues standardization and promotion of the software tag scheme. In this project, we have defined the detailed structure of the software tag and developed various support tools. The software tag has been applied to real projects of major Japanese organizations. Along with technical development, we have also started various promotion activities, such as formal standardization of the software tag in both domestic and international standards, and exploration of new trade laws for software using the software tag scheme.

An early concept of how software tags could be used for software maintenance was shown in [4]. In this paper, we mainly explain use of the software tag for software development, together with activities and outcomes from the StagE project. In section 2, we describe an overview of the software tag scheme, and in section 3 explain the details of the software tag structure. In section 4, we describe activities of the project. In section 5 we provide some discussion, while in section 6 we outline conclusions and future research topics.

2. OVERVIEW OF THE SOFTWARE TAG SCHEME
A software tag is a packaged data set about a software project. It is currently composed of 41 characteristic elements of project data and progress data, as defined in section 3.1. Figure 1 shows an overview of the software tag scheme.

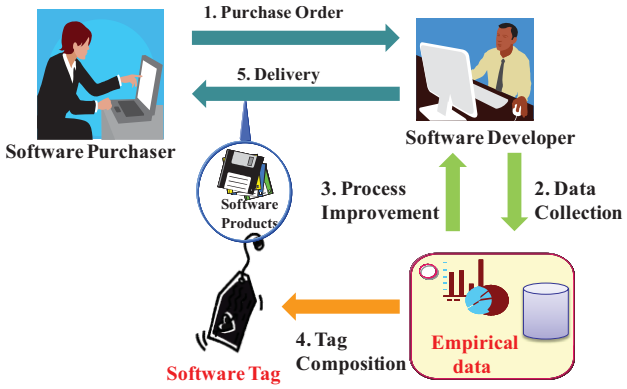

Figure 1. Overview of Software Tag Scheme

1. A software purchaser orders development of a software system. The purchaser includes both the final products and the software tag in their requirements.

2. During software development, various kinds of empirical data are created and generated. For example, requirements documents, software design documents, source code, test cases, issue tracking logs, manual documents, review logs, and quality analysis records may be produced. These are collected and archived. Note that we collect not only the final data at the end, but also interim snapshot data during development.

3. The collected data is analyzed for process improvement of the development organization, as is the usual process improvement scheme for software development organizations.

4. The collected data is used to construct the software tag. Parts of the empirical data are selected and abstracted into the software tag format.

5. The software tag is delivered to the software purchaser periodically during the development and/or finally at the end of the development together with the final software product. The software purchaser evaluates the software development by viewing and analyzing the tag, and accepts the delivered software product.

If a controversy such as a question about the quality of the product occurs between the software purchaser and the developer, the delivered software tag and (if necessary) the empirical data are analyzed, providing a basis for exploring a resolution to the controversy.

The software tag is a key to improving *transparency* of software projects. By examining the software tag, the software purchaser can identify and understand the development process, which has been mostly hidden from the purchaser. The purchaser can evaluate the quality of the processes and products of the project.

For the software developer, the software tag is useful to prove that they have conducted the proper activities in the software project. Also, it can be used to trace the quality of the activities of sub-contractors and sub-sub-contractors... (such contracting chains are very popular in Japan).

This scheme can be very useful for offshore and global development, because transparency and traceability of software development can be established with a fairly low overhead for the developers.

Standardizing the software tag will help to establish a minimum baseline for project quality, and to improve negotiations over software development contracts. Evaluation of software products and projects based on the objective empirical data contained in the software tag will lead to more healthy use of software in society.

3. DEVELOPMENT OF SOFTWARE TAG TECHNOLOGIES

3.1 Software Tag Standard 1.0

We have defined the elements of the software tag as shown in Table 1, named *Software Tag Standard 1.0*. It is composed of 41 tag elements, which are categorized into *project information* and *progress information*. The project information depicts the overall sketch of the project with various basic pieces of information. The progress information provides qualitative and quantitative indices of project achievement with various measures of the development phases. The tag standard provides more precise explanations and example metrics for each tag element which are not presented here.

It is not mandatory to use all 41 elements in the software tag in all cases. The purchaser and the developer can negotiate and select elements to use. Also, they can discuss and determine the details of the metrics. For example, #19, Scale of Programming, might be agreed to be measured by lines of code without comments.

In this standard, we have included various kinds of information that are considered important to the purchasers. The overall structure should be simple for the purchaser to understand, so we have tried to keep it as simple as possible. Also, we have tried to keep in mind the balance of the tag elements. This standard does not include tag elements that are computable from other tag elements. There are a number of standards and reports such as SWEBOK, CMMI, ISO/IEC 15939, and reports by the Software Engineering Center in Japan (SEC) which can help interpret the tag elements.

The definition process was based on discussions with industry and academic collaborators such as:

Purchasers: Tokyo Stock Exchange, Japan Aerospace Exploration Agency, DENSO.
Developers: Fujitsu Lab, Hitachi, NEC, SHARP, SRA Key-Tech Lab, Toshiba, NTT Data.
Others: Information Technology Promotion Agency, Ministry of Economy, Trade and Industry, Japan (IPA), Nara Institute of Science and Technology, Osaka University.

3.2 Support Tools

We are developing various support tools to promote the software tags scheme. In this paper, we introduce two essential tool prototypes that have been created for collection and visualization of the software tag.

(1) Software tag data collection tool (*CollectTag*)

CollectTag supports collection of empirical data from software projects and creation of a software tag. CollectTag uses a wizard as a user interface, allowing the user (developer) to easily input the necessary data for the software tag.

For each project, the purchasers and developers determine the metrics for the tag elements. To provide generality, we implemented CollectTag as a translator that converts a set of

Table 1. Software Tag Standard 1.0

Classification	Category	No.	Tag Element	Explanation
Project Information	Basic Information	1	**Project Name**	Unique name of project
		2	**Organization**	Information of development organization
		3	**Project Information**	Information needed to identify the project characteristics
		4	**Customer Information**	Information identifying the purchaser or owner
	System Information	5	**System Configuration**	Information identifying system configuration to label the type of system
		6	**System Scale**	Development system scale
	Development Information	7	**Development Approach**	Development process type or techniques
		8	**Organizational Structure**	Structure of development organization
		9	**Project Duration**	Information of development length
	Project Organization	10	**Super-Project Information**	Name of super project which creates this project
		11	**Sub-Project Information**	Name of sub projects which is created by this project
	Other	12	**Special Notes**	Other necessary or useful data for interpreting or analyzing tag data
Progress Information	Requirements	13	**User Hearing Information**	Information of user-requirements hearing
		14	**Scale**	Amount of requirements
		15	**Revisions**	Amount of changed requirement
	Design	16	**Scale**	Amount of design products
		17	**Revisions**	Amount of changed design
		18	**Design Coverage by Requirements**	Implementation ratio of design for requirements
	Programming	19	**Scale**	Amount of programming products
		20	**Revisions**	Amount of changed programs
		21	**Complexity**	Complexity of programs
	Test	22	**Scale**	Amount of testing
		23	**Revisions**	Amount of changed test
		24	**Density**	Ratio of test to system size
		25	**Progress Status**	Test progress to plan
	Quality	26	**Review Status**	Quantity information of review
		27	**Review Density**	Ratio of review to system size
		28	**Review Effectiveness**	Ration of found defects to amount of review
		29	**Defect Count**	Number of defects found by test
		30	**Fixed Defect Count**	Number of fixed defects
		31	**Defect Density**	Ratio of defects to system size
		32	**Defect Detection Rate**	Ratio of detected defects to consumed test
		33	**Static Check Results**	Report of static checker
	Development Cost	34	**Overall Cost**	Development and maintenance cost
		35	**Productivity**	Ratio of amount of products to overall cost
	Schedule and Management	36	**Process Management**	Information on management of development process
		37	**Purchaser-Developer Meeting Status**	Amount of user-vendor communication
		38	**Total Risk Item Count**	Number of risk items in the development
		39	**Risk Item Existence Period**	Time length between a risk item creation and deletion
	Other Products	40	**Scale**	Amount of product metrics not listed above
		41	**Revisions**	Amount of change in products not listed above

empirical data provided by the developer into the standard software tag format. That is, the developer periodically inputs values for each tag element, and then CollectTag outputs a software tag.

To reduce the effort of data input, CollectTag provides automatic data collection mechanisms for 11 of the tag elements in the progress information, if the target project uses common software development tools for configuration management and bug tracking. For example, LOC (#19: Scale) and CK (#21: Complexity) metrics [3] can be automatically collected and calculated from configuration management tools such as CVS or Subversion.

Finally, CollectTag generates the software tag elements in XML format (named *standard software tag format*). This makes it easy to provide the output to other visualization and analysis tools for further processing.

(2) Software tag visualization tool (*TagReplayer*)

TagReplayer provides fundamental features for integrated visualization of various historical data included in the software tag. TagReplayer employs the metaphor of video player manipulation for its user interface so that users can replay the progress of the project just like watching video on TV. Users can also instantly recall the details of any points along the timeline based on the software tag as shown in Figure 2. Our experience shows that this feature is very useful for postmortem project reviews.

TagReplayer aligns progress information from the software tag as a series of events. As a project summary, it displays multiple views including line charts, progress bars, and plain lists. For more details, clicking the items or points in the summary view can instantly recall empirical data such as problem descriptions or communicated messages. The tool also provides natural text mining and clustering that is useful for deeper analysis of human activity records, such as the problem reports or developers' communication messages, if the information is associated with the software tag.

3.3 Applications of the Software Tag

We present here three case studies of application of the software tags scheme to real software projects.

(1) A course registration system for a university with 26K LOC in Java was developed for five months by a medium-sized software company in Japan. 32 elements of tag data were collected and used to analyze the project status, *e.g.*, refactoring and the probability of insufficient testing density, by comparing the element values with the publicly available benchmark values from the Software Engineering Center in Japan (SEC).

(2) A medium-sized stock exchange system for a stock market was enhanced by a Japanese major software development company for more than two years. Using tag elements related to the requirements phase such as requirements revisions (#15), defect count (#29), and review status (#26), the purchaser and developer were able to identify problems with the requirements completeness caused by frequent changes.

(3) A Japanese software development company ordered several small-sized projects such as development of a project management support system from various offshore companies in China and Korea. Although remotely located from each other, the purchasers and developers could understand the progress of the specifications using tag elements such as the review status (#26), user hearing information (#13), and defect count (#29).

4. ACTIVITIES FOR PROMOTION AND DIFFUSION

The StagE project is also actively promoting and diffusing the software tags scheme in industry as follows.

(1) International/Domestic Standardization

Interviews with several Japanese software purchasers and developers, along with offshore software developers for Japanese companies in some countries, convinced us that most software purchasers and developers would strongly demand that the software tag and tools should be international and/or domestic technical standards in software engineering. To support this, we

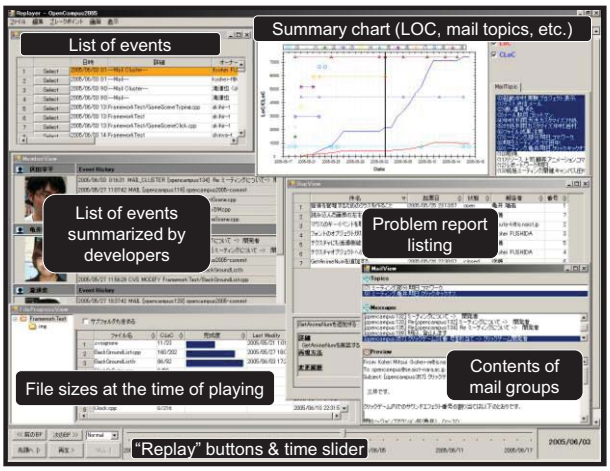

Figure 2. TagReplayer Screenshot

are now serving ISO as a committee member of the working group on process assessment, ISO/ITC JTC1/SC7/WG10. We are also working with some software tool developers to construct a de facto standard software project management system that includes the tag support tools mentioned in Sec. 3.2.

(2) International Collaboration

Offshore software development is one of the most useful application areas of the software tag scheme. To encourage and accelerate international collaboration to have various kinds of case studies and experiments of the software tag in offshore software development, we established Asia-Pacific Software Engineering Research Network (APSERN) in 2008 with software engineering researchers in NICTA (National ICT Australia), ISCAS (The Institute of Software, Chinese Academy of Sciences), and so on.

(3) Professional discussion of Legal issues

In the case of a legal dispute between software purchasers and developers, the software tag can clarify their liabilities and has the potential to help resolve such legal issues in software development. The StagE project has a committee examining the legal issues of software development. Members of this committee include lawyers, patent attorneys, and software engineers. The committee has interviewed many software developers in Japan and China to compile data about troubles between software purchasers and developers. It also distributed questionnaires to more than a hundred software developers in Japan to analyze the trends of such troubles. The software tag provides an opportunity for collaboration between software engineering and software trade law.

5. DISCUSSION

(1) There are metrics repositories aimed at improving and benchmarking development organizations [5], along with some software measurement paradigms [2] [6]. However, the software tag provides a unique approach to involve software purchasers in the quality improvement framework by providing development transparency. As far as we know, there is no similar approach presented in the technical literature.

(2) We believe that the benefits of the software tag scheme for software purchasers will be substantial because the development processes and the developed products become more visible and understandable. However, purchasers will need to collaborate more closely with developers, providing effort and enthusiasm to create successful projects.

(3) We have presented the first standard of the software tag with 41 elements. In some sense, these are very basic data for indicating development quality, and they may be insufficient to perform detailed analysis. However, as a standard used for various software development projects, the set should be minimal and low cost. As presented in Sec. 3.1 and 3.2, our tag standard 1.0 is a lightweight set with low collection and assembly cost. It is important to continue practical applications of the software tag, and to get feedback for further improvement of the standard.

6. CONCLUSIONS

We have introduced our activities for standardization of the software tag in Japan. Through these activities, the concept of the software tag is becoming well understood in Japan.

Our future work will focus on making international/domestic standards of the software tag. With such standardization, the software tag is expected to be used in various software industries, where we think it will strongly promote participation and understanding of software development by purchasers.

7. ACKNOWLEDGMENTS

This work is being conducted as a part of the StagE project, The Development of Next-Generation IT Infrastructure, supported by the Ministry of Education, Culture, Sports, Science and Technology. We are grateful to the members of the StagE project, especially to Michael Barker, Akito Monden, Makoto Matsushita, and Shuji Morisaki.

8. REFERENCES

[1] Barker, M., Matsumoto, M., and Inoue, K. 2008. Putting a TAG on Software: Purchaser-Centered Software Engineering. In Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization, Ramachandran, M., and Carvalho R. A., Eds. Information Science Reference, Hershey, PA, 38-48.

[2] Basili, V.R., and Rombach, H.D. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Trans. Softw. Eng. 14, 6 (June 1988), 758-773.

[3] Chidamber, S. R., and Kemerer, C. F. 1994. A Metrics Suite for Object Oriented Design. IEEE Trans. Softw. Eng. 20, 6 (June 1994), 476-493.

[4] Inoue, K. 2008. Software Tag for Traceability and Transparency of Maintenance. In Proceedings of, 24th IEEE International Conference on Software Maintenance (Beijing, China, Sep. 28 - Oct. 4, 2008). ICSM 2008, 476-477.

[5] International Software Benchmarking Standards Group (ISBSG) 2004. ISBSG Estimating: Benchmarking and research suite.

[6] Kitchenham, B. A. Hughes, R. T., and Linkman, S. G. 2001. Modeling Software Measurement Data. IEEE Trans. Softw. Eng. 27, 9 (Sep. 2001), 788-804.

[7] Nikkei Business Publications, Inc. 2008. Survey Report on IT Investment and CTO in Japan. Nikkei Information Strategy. March (in Japanese).

[8] Nikkei Business Publications, Inc. 2008. Second Survey of Japanese Software Projects. Nikkei Computer. Dec. 1, 36-53 (in Japanese).

ソフトウェアタグ規格

第1.0版

2008年10月14日

StagEプロジェクト
奈良先端科学技術大学院大学
大阪大学

目次

前文 1
(1) 経緯 1
(2) 目的 1
(3) 用語定義 3
(4) 留意事項 5
(5) 本規格が支援するポイント 5
(6) 本規格による効果 5
(7) 信頼性ガイドライン等とソフトウェアタグの関連 6
(8) まとめ 7
1. 本規格の構成 9
2. 規格の使用方法（運用ガイドライン） 9
3. データ共有の目的と関連タグ項目例 9
3.1 製品品質の把握 9
3.2 開発状況の把握 9
3.3 紛争処理 9
4. タグの構造 10
4.1 階層構造 10
4.2 項目の繰り返し 11
5. 標準タグ項目一覧の概要 11
5.1 一覧表の見方 11
5.2 標準タグ項目の位置づけ 11
6. 標準タグ項目一覧表 13
付録 16
体制 16
参考文献 17

前文

(1) 経緯

近年，ソフトウェア開発プロジェクトにおいて，ソフトウェア発注者（ユーザ[1]）の参画が重要視されてきている．従来，ソフトウェア発注者（ユーザ）は要件定義など上流工程からソフトウェア受注者（ベンダ）まかせであるケースが多かった．しかし，ユーザ不在の開発では，要件のあいまいさから設計ミスや頻繁な仕様変更が発生し，品質低下，重大な不具合によるシステムダウン，コスト増大，納期遅れなどの原因となっている．また，障害発生時には，ユーザによる原因や責任所在の追及が困難で，障害対応が遅れ，甚大な損害が発生するケースも多い．さらに，法的な係争に発展すると，裁判が長期化し，解決までユーザ・ベンダ双方の損害が増大する．一方，多重請負・オフショア開発などの複雑な開発体制や多重の受発注関係では，潜在的発注者（ユーザ）が多く存在し，統制されたプロジェクトの管理を困難にし，管理コストを増大させている．

一方，携帯電話や家電製品への組込ソフトウェアの分野では，社内ソフトウェア部品を流用した派生開発が頻繁に行われている．また，ソフトウェア開発ベンダが，品質の確保や納期短縮のため外部からパッケージソフトウェアを購入し，システムに組込む事例も多い．このようなケースでは，長期にわたる流用・改造によって設計情報などが散逸したり，開発した組織が異なるため必要な情報を得られないなどの理由で，開発が困難になったり，予期しない品質問題によってダメージを受けることも多い．

これまで，ソフトウェア開発におけるデータの収集・分析は，当該組織でのプロセス改善やプロジェクト内部の進捗・品質管理を目的とし，組織間共有や一般ユーザの視点に立って行われてこなかった．しかし，ソフトウェアの問題の社会的な影響度の拡大や，マルチベンダ開発やオフショア開発など開発組織の複雑化に対して，このような開発プロジェクトデータのより幅広い活用が求められてきている．例えば，ソフトウェアの不具合による法的紛争では，様々な開発データに基づいて責任の所在が追及される．

(2) 目的

StagE (Software Traceability and Accountability of Global Software Engineering)プロジェクト[2]では，ソフトウェア開発データのユーザとの共有のための基盤研究開発を目的とし，ソフトウェアの品質や由来をユーザに示すための技術として「ソフトウェアタグ」の研究・開発を行っている．ソフトウェアタグは，ソフトウェアの開発中に収集された管理データや品質情報など複数のデータから構成され，各データ種類をタグ項目と呼ぶ．タグ項目に従って，開発成果物や副次的に発生するデータ（実証データ/エンピリカルデータ）からタグデータが開発中，もしくは開発完了時に作成され，ソフトウェアに付随して流通する．ソフトウェアユーザは，タグの内容を見ることによって，対象ソフトウェアの開発過程や品質保証のプロセス，設計書やプログラムなどの成果物の品質情報を入手でき，安心して安全なソフトウェアを利用することが可能になる．ソフトウェアタグは，プロジェクト進行中にプ

---

[1] 本規格では，ソフトウェア開発関係者として，主に「ユーザ」「ベンダ」という 2 つの言葉を用いる．特記しない限り，「ユーザ」はソフトウェア発注者，ソフトウェア利用者（エンドユーザ）などを含み，「ベンダ」はソフトウェア受注者，ソフトウェア開発者などソフトウェアを供給する人・組織を示す．（詳細は，「用語定義」参照）

[2] 文部科学省のプロジェクト 次世代 IT 基盤構築のための研究開発「エンピリカルデータに基づくソフトウェアタグ技術の開発と普及」の略称

ロジェクト管理のための定量的データを共有するために用いることも可能で,プロジェクト進行中に発生する問題の早期発見・解決が可能になる．また，ユーザ・ベンダ間の紛争処理では実際の開発データに基づいた詳細かつ正確な検証が可能になる．

「ソフトウェアタグ規格」は，ソフトウェアタグに含まれることが望ましい標準的なデータ項目とタグの運用ガイドラインを提供する．例えば，ソフトウェア発注者（ユーザ）と受注者（ベンダ）は，本規格に基づいて，両者で共有するべきデータを取り決め，どのように運用・管理するかといった具体的な活用方法を検討することができ，ユーザ・ベンダ両者が参画する定量データによるプロジェクト管理の導入が容易になる．また，収集データの標準化によりデータの組織間の共有が容易になり，統一基準による適正な評価も促進される．

以下に,本規格の構築に先立って検討されたソフトウェアタグの具体的な利用目的を挙げる．利用目的は，プロジェクトやシステムに応じて異なり，下記に限定されるものではないが，産業界において必要性の高い場面を想定している．

- 製品品質の把握
  - 流用資産の保証書／部品として使う製品と高信頼システム全体の信頼性認証:開発プロセスとプロダクトの動作に関する試験結果を監査し,製品及び高信頼システムを認証するために用いる．（図 1 参照）
  - 製品情報：工業製品や農業製品についているプロファイル情報（生産者，原産地，原材料，添加物など）のソフトウェア版．使っている基盤ソフト，パッケージソフト，開発言語，開発規模，開発者（企業名），サービス提供者などのほかに，試験結果など品質情報もオプションとして加えても良い．
- 開発状況の把握
  - 要件定義の充実度:要件定義が曖昧であることがプロジェクトの成否の鍵を握るといわれているが，定量的な分析はあまりない．要件定義の充実度を図るメトリクス及びプロジェクトの成功/失敗についてのカテゴリを定義することで，どのような要件定義をすればよいか，明らかにする．仕様変更などもプロジェクトの成否にかかわる有益な情報である．
  - 開発中の情報共有：開発中や検収時に，ユーザ（顧客）に対して示すプロジェクトデータをタグとして共有する．納品時の事後分析だけでなく，開発中の進捗状況の把握などにも用いることができる．（図 2 参照）
- 紛争処理
  - 事故調査：ハードウェアの世界では一般的な事故調査委員会（飛行機/電車事故など）が検査するためのデータのソフトウェア版．原因を究明して，プロセス問題に帰着させ，プロセス改善に結びつける．
  - 法的係争時のプロセスの証拠： 障害や事故の責任の所在を明らかにする．プロセス遵守度・プロセス規定度・プロセス管理度など．（図 3 参照）

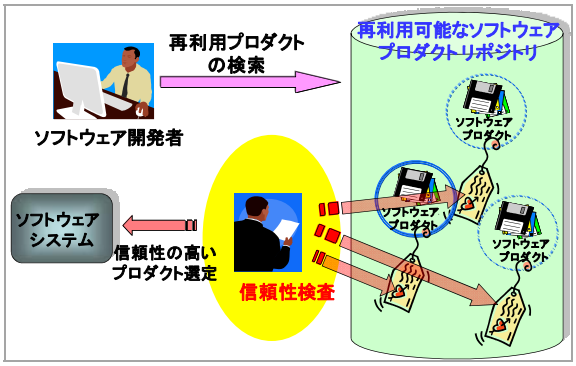

図 1　流用資産の保証書／部品として使う製品と
高信頼システム全体の信頼性認証

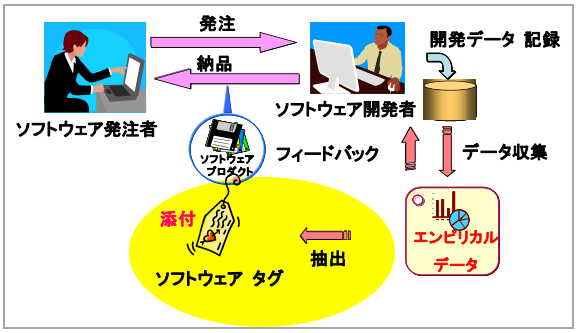

図 2　開発中の情報共有

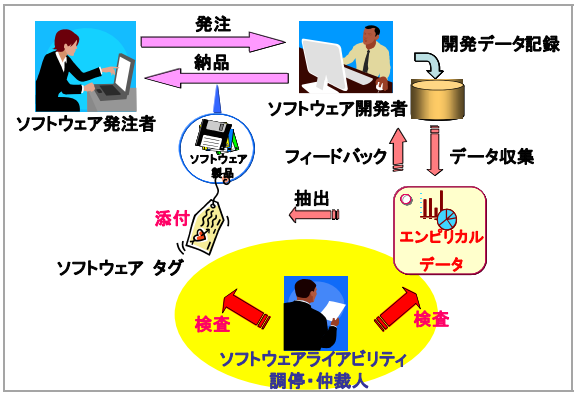

図 3　法的係争時のプロセスの証拠

(3) 用語定義

データ

記録の形態や方法にかかわらず，記録された情報，もしくはそれらから加工・抽出・算出された情報.

定量的データ

数値化されたデータ．規模や件数など対象物の計測で得られるデータのほか，欠陥密度など複数のデータから導出されるデータを含む．また，重要度，難易度などの定性的な属性も，集計などによって定量的に扱われる場合，定量的データに含まれる.

実証データ（エンピリカルデータ）

対象の実測により得られるデータ．ソフトウェア開発プロジェクトにおいては，最終成果物であるシステムから得られるデータだけでなく，開発過程で生じる副次的なデータ，例えば議事録，欠陥レポート，工程管理表なども含まれる．

メトリクス

定量的な評価尺度．

プロジェクト

一般的には，目標達成のための計画から実施を含む作業の単位．本文中では，ソフトウェア開発プロジェクトの意味で用いるが，必ずしもシステムの構築が目標ではなく，要件定義，設計など工程別に行われるプロジェクトも対象とする．標準タグ項目での「プロジェクト名」「サブプロジェクト」などは，タグの作成・流通単位としてのプロジェクトであり，ソフトウェア開発プロジェクト全体とは異なる狭義の意味を持つ．

プロダクト（成果物／製品）

ソフトウェア開発プロジェクトでは，顧客や最終利用者への納入のための作業成果物．プログラム，品質保証データ，設計書，マニュアルなど，プロジェクトによってプロダクトは異なる．

ベンダ

ソフトウェア開発における狭義のベンダを意味する．ソフトウェア開発における「ベンダ」は，ソフトウェア受注者，ソフトウェア開発者などソフトウェアを供給する人・組織を示す．本文中では，パッケージソフトウェアの販売代理店など，実際に開発過程にかかわらない組織は，対象としない．

ユーザ

委託開発における発注者，システムのエンドユーザ，パッケージソフトウェアの利用者，ソフトウェア部品を流用・改造するソフトウェア開発者などが挙げられる．

発注者・受注者

２者間契約による委託開発における受発注関係にある組織を指す．これは，ユーザと１次請けベンダの関係だけではなく，ベンダからの外部委託などの関係も含む．従って，１組織が２つの立場（発注者であり，受注者）になる場合もある．

ソフトウェアタグ

ソフトウェアの開発中に収集された管理データや品質情報をパッケージ化したもの．ソフトウェア製品に付随して流通し，ソフトウェアユーザは，タグの内容を見ることによって，対象ソフトウェアの開発過程や品質保証のプロセス，設計書やプログラムなどの成果物の品質情報を入手することができる．

タグ項目

ソフトウェアタグを構成する個々のデータ種類．

タグデータ

ソフトウェアタグに含まれる実際の値や情報．基本的に，実証データ（エンピリカルデータ）から抽出されるが，実証データそのものを用いることも可能である．タグデータは，実証データによって裏づけが可能なデータであることを前提とする．

(4) 留意事項

汎用性

本規格は，できるだけ汎用的な枠組みとするため，特定のプロジェクト・分野の個別性の高い内容は含めない．タグ項目などに具体性に欠ける場合，説明のための詳細情報として，具体例を示す．特定の目的・分野ごとに必要な具体化は，利用シナリオで行う．

強制力

本規格は，ユーザ・ベンダ間（受発注者間）での協議のベースなどに用いる枠組みを提供するものであり，そのまますべてのプロジェクトやソフトウェアに導入を強制するものではない．また，本規格の利用者による追加・変更・削除などを制限しない．

具体例の活用

本規格には，汎用性を保つため抽象的なレベルでとどめている記述がある．プロジェクトでの活用を容易にするため，データの具体的なデータ収集・加工の手法/手段として具体化例や実証データ例を示す．ただし，例示するにとどめ，すべての具体化方法を網羅するものではない．

(5) 本規格が支援するポイント

① データ収集

ソフトウェア開発プロジェクトにおいて，収集するべきデータを明確にする．このデータは，プロジェクトの進捗やコストをリアルタイムに管理し，納期の遅延やコスト超過を防止したり要求されている品質を保証したりするために，開発側（ベンダ）が必要とする情報をベースとするが，特にユーザと共有することが望まれるデータ（紛争時の解決に有効なデータなど）についても取り上げる．

② データ加工・可視化

本規格では，ユーザと効果的にデータを共有するために，収集されたデータを加工・可視化する方法や技術の例を示す．収集されるデータは膨大な量であり，そのまま開示しても分析や解釈のコストが増大し，効率的・効果的にデータが共有できない．そのため，ユーザと共有するデータは，双方の目的に応じて必要最小限の量であることが望ましい．本規格では，開発中の収集データからどのような定量的データを加工し，そのデータをどのようにユーザに見せるかといった点について，例を提供する．

③ ユーザ・ベンダ間のデータ運用

本規格では，ユーザ・ベンダの目的やプロジェクトの特性に合わせたデータ運用の枠組みを提供し，ユーザ・ベンダ間でどのようにデータを用いたプロジェクト・ソフトウェア管理を行うか，その手順を明確にする．具体的には，目的に応じた収集データの選択，契約への記載，データ共有タイミング，データ授受方法，それに伴うコストなどである．

(6) 本規格による効果

① データ収集の促進とデータ項目の標準化

一般的なデータ項目と収集・加工方法などの情報の利用により，データ収集の導入が容易になり，定量的プロジェクト管理・プロセス改善が促進される．データ項目の標準化によって，ツール・計測技術の開発が促進され，計測や分析のコストが低減する．

② ユーザ・ベンダ間のデータ共有の促進

組織間のデータ項目の統一により，データ共有・流通が容易になる．統一基準による組織・プロジェクト・システム評価により，適正競争が促進され，国際競争力が強化される．

③ ユーザのデータ収集・分析への理解・協力
　データ共有の目的の明確化やプロジェクト管理への理解向上により,ユーザの協力（コスト負担など）が得やすくなる.

④ 障害対応や紛争処理
　あらかじめ収集されているデータが明確になり,要約されたデータが共有されていることによって，障害発生時には，ユーザ・ベンダ間の協力による迅速な原因究明・問題解決が可能になる．また，ユーザ・ベンダ間の紛争に発展した場合にも，公正かつ迅速な解決が図れる.

(7) 信頼性ガイドライン等とソフトウェアタグの関連

経済産業省は，情報産業強化施策の一環として，「産業構造・市場取引の可視化」[3]を挙げ，ユーザ・ベンダ一体となった生産性向上の促進を目指している．「情報システムの信頼性向上に関するガイドライン」[4]（以降，信頼性ガイドライン）では，“情報システムの利用者及び供給者の双方が応分の責務を担う”とし，「企画・開発における，信頼性・安全性の水準検討」「障害発生時の手順の文書化・記録」「定量的な手法を取り入れたプロジェクトマネジメントの実行」などについて，受発注者双方の果たすべき役割について述べている．それを受けて，「情報システムの信頼性向上に関する評価指標（試行版）」[5]（以降，信頼性評価指標）では，ユーザ・ベンダ双方のガイドラインの遵守状況を評価のための評価指標を示し，具体的なチェックリストを公開している．一方，「情報システム信頼性向上のための取引慣行・契約に関する研究会」最終報告書[6]（以降，モデル契約書）では，ユーザの主体的なプロジェクトへのかかわりを促進することを目的として,契約書にユーザやベンダの役割・責任分担を明確に示すことが重要である，と述べている．契約に明記することによって，ユーザによるベンダへの“丸投げ”を防ぎ，仕様の固まりにくいプロジェクトの円滑な進行を促進するという効果を期待している．また，モデル契約書では，ユーザ・ベンダ間でのソフトウェア開発に関する紛争解決方法として，裁判外紛争処理（ADR）の活用を推奨している．専門的・技術的視点による詳細な事実認定や当事者間の話し合いによって，紛争処理を迅速かつ柔軟に行うことが可能である，と考えられている.

ソフトウェアタグは，これらのガイドライン等に対して，それぞれの目的に応じて具体的なデータに基づいて支援することが可能である（図 4 参照）．信頼性ガイドラインに挙げられている「定量データを活用した管理」は，本規格の標準タグ項目をベースにユーザ・ベンダ間で検討することによって，導入が容易になると考えられる．また，ソフトウェアタグとして収集されたデータを用いて,ガイドラインの遵守状況をより詳細に評価することが可能になる．一方，ソフトウェアタグに関する条項を契約書に盛り込むことによって，プロジェクト管理に関するユーザ・ベンダ双方の責任・役割を明確にすることができ，データ収集に伴うコストなども契約時に協議することが可能になる．一方，ADR による紛争処理がおこなわれる際には，実際の開発データに基づいた詳細かつ正確な検証が可能になる.

---

[3] http://www.meti.go.jp/policy/it_policy/softseibi/index.html
[4] 2006 年 6 月 15 日公表，経済産業省
[5] 2007 年 4 月 13 日公表，経済産業省
[6] 2007 年 4 月 13 日公表，経済産業省

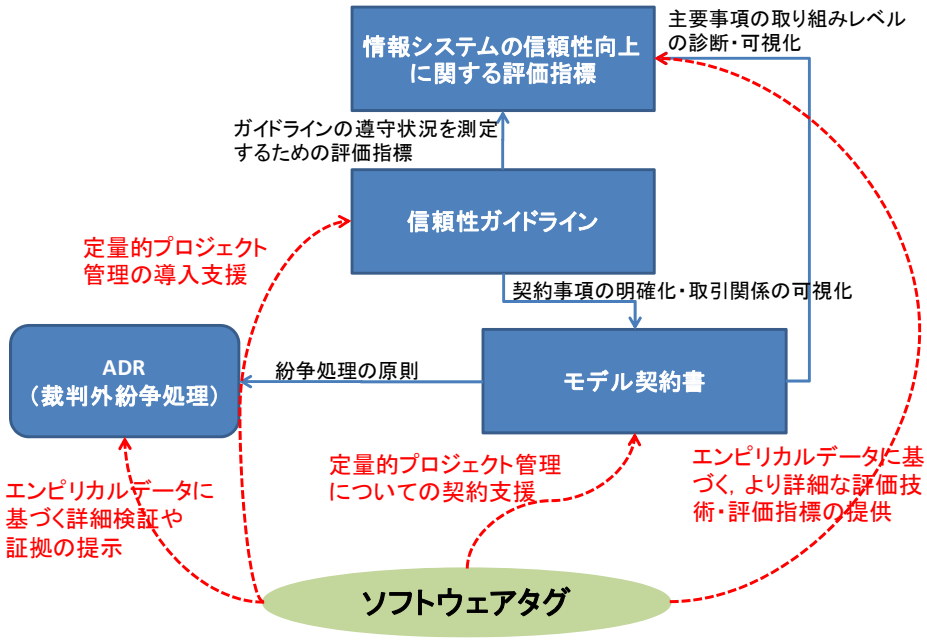

図 4　情報政策とソフトウェアタグの関連

一方，JISA(社団法人情報サービス産業協会)では，「信頼性向上ベストプラクティスを実現する管理指標調査報告書」[7]を刊行し，ベンダ視点から信頼性向上のための具体的な指標の効果や活用方法を示した．この中で，顧客や協力会社との対話における指標活用のポイントとして，「活用目的と意義の共有」「活用する指標の早期合意の重要性」「定例会議での指標の報告・確認と議論の効果」などが挙げられている．

ソフトウェアタグ規格は，ユーザに重点を置いた指標を選択することによって，組織間のデータ共有・活用を支援すると考えられる．

(8) まとめ

上記のとおり，ソフトウェアタグは，これまでのソフトウェア開発プロジェクト管理を大きく変更するものではなく，ユーザ視点によるデータ活用を促進するために，既存の技術や知識を再構成することによって実現される．本規格は，その具体的な方法，すなわちデータの種類や活用方法を検討するために，ユーザ・ベンダ間で用いられる 1 つの指針として作成される．

ソフトウェアタグに含まれるデータ項目やデータ構造などは，本規格で具体的に示されているが，今後の課題として以下のことが考えられる．

- 実プロジェクトにおいてどんなタイミングで情報を取るのかなど，本規格のユーザが運用の具体的なイメージを持つための，利用シナリオの充実
- 本規格に準じたソフトウェアタグの普及を促進するため，ユーザサイドへの啓発活動の促進（経済産業省，SEC などとの連携や，タグの認証や検証を行う第三者機関の設立など）
- 国際規格化
- データ収集・分析ツールの開発や既存ツールを活用した導入や普及の支援
- 本規格を保守・運用・管理する体制の整備

---

[7] 2008 年 4 月，社団法人情報サービス協会．概要は，以下 URL 参照．
http://www.jisa.or.jp/report/2007/19-J009.pdf

1. 本規格の構成

(1) 規格の使用方法（運用ガイドライン）
　ソフトウェア開発プロジェクトの企画段階で，ユーザ・ベンダ間でのデータ共有に関する取り決めを行うにあたっての観点・手順を示す.

(2) データ共有の目的と関連タグ項目
　ユーザ・ベンダ間で共有するデータの項目を決定するために，データ共有の主要な目的とそれに関連する標準的なタグ項目の例を示す.実際には,より詳細な目的を設定し，目的に応じた具体的なタグ項目を選択することが必要である.

(3) タグの構造
　開発プロジェクトの管理や部品の流用・再利用を考慮した，推奨するソフトウェアタグ構造を示す.

(4) 標準タグ項目一覧の概要
　(5)の標準タグ項目一覧表の見方と，運用上の位置づけを示す.

(5) 標準タグ項目一覧表
　標準的にプロジェクト管理に用いられるタグ項目の一覧表を示す.

2. 規格の使用方法（運用ガイドライン）

　ソフトウェア開発プロジェクトの企画段階で，ユーザ・ベンダ間でのデータ共有に関する取り決めを行うにあたっての観点・手順を示す．また，各ステップで参照するべき本規格の章を示す.

(1) データ収集・共有目的の設定
　「3. データ共有の目的と関連タグ項目例」参照

(2) 目的に応じたデータ項目の選択
　「3. データ共有の目的と関連タグ項目例」「5. 標準タグ項目一覧の概要」「6. 標準タグ項目一覧表」参照

(3) データ定義・収集方法・開示データレベルの決定
　「5. 標準タグ項目一覧の概要」「6. 標準タグ項目一覧表」参照

(4) システム・プロジェクト構成に応じたタグ構造の決定
　「4. タグの構造」参照

(5) データの収集・計測・開示タイミングの決定

(6) 開示データの活用方法（説明，議論，解釈/判定，フィードバック）の決定

(7) 開発完了後（保守，運用，再利用等)におけるタグ活用の検討

3. データ共有の目的と関連タグ項目例

　ユーザ・ベンダ間で共有するデータの項目を決定するために，データ共有の主要な目的とそれに関連する標準的なデータ項目の例を示す．実際には，より詳細な目的を設定し，目的に応じた具体的なデータ項目を選択することが必要である.

3.1 製品品質の把握

- 品質の把握に関する項目：バグ数，レビュー状況，静的解析結果など
- ソフトウェアを部品としての再利用するために必要な情報に関する項目 ：システム構成，テスト項目数，リスク項目数など

3.2 開発状況の把握

- 開発環境（リソース等）に関する項目 ：開発体制，開発手法など
- プロセス情報に関する項目 ：要件/設計変更，テスト消化，プロセス管理情報など

3.3 紛争処理

- 試験に関する項目：バグ数，レビュー状況，テスト項目数など

- ユーザ・ベンダ間合意に関する項目：ユーザヒアリング情報，要件変更，会議実施状況など

4. タグの構造

開発プロジェクトの管理や部品の流用・再利用を考慮した，推奨するソフトウェアタグ構造を示す.

4.1 階層構造

ソフトウェアタグは，システムの構成やプロジェクトの契約単位，開発チーム構成などに応じて階層構造で持つ.

図 5(a)のシステム A のように 2 つのサブシステム X,Y から構成されるシステムの場合，ソフトウェアタグは 1 つのメインタグとそれにリンクする 2 つのサブ（子）タグから構成することができる．このように進捗管理や品質管理の単位で構成すると，収集・分析データの流用も容易であり，タグの管理コストが低減できる.

また，タグを階層構造にすると，サブシステムの再利用や流用も容易になる．図 5(b)は，サブシステム Y を他のシステム B の部品として流用する際のタグの構造を示している．濃い網掛けのタグが同一タグになる．薄い網掛けのタグは，流用するサブシステムのタグ(濃い網掛け)から新規に作成され，システム B への組込み作業のデータを持つ．このような構造をとることによって，流用部品の過去のデータ管理が容易になり，ソフトウェア部品のトレーサビリティを確保できる.

開発されたシステムの運用・保守を別契約で行う際も，図 5(c)のようにタグを階層構造にすることが望ましい．この場合，メインタグとサブタグで重複する情報を省略し，各サブタグは，運用・保守プロセス作業で発生するデータのみを持つことになる.

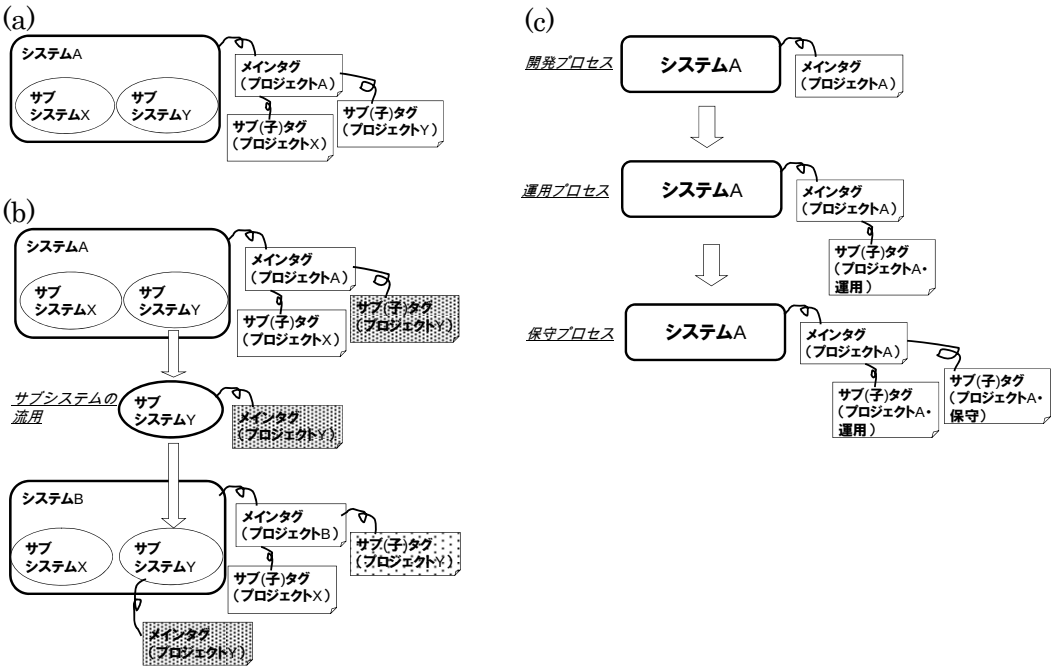

図 5　タグの構造

(a)システム構成時のタグ構造　(b)流用時のタグ構造　(c)保守・運用時のタグ構造

4.2　項目の繰り返し

タグ項目は，開発工程やプロダクトの性質に応じて，複数持つことが適切な場合，同一項目の繰り返しを許容する．例えば，進捗情報の「設計」分類に関して，基本設計，詳細設計などの工程が存在する場合，「設計」は同一タグ内に複数存在するのが適切と考えられる．また，進捗情報の「プログラミング->規模」についても，派生開発などで新規作成・改造・再利用（流用）など種別に計測することが必要な場合，「プログラミング->規模」項目が３つ存在することを許す．

5. 標準タグ項目一覧の概要

標準的にプロジェクト管理に用いられるタグ項目の一覧と，運用上の位置づけを示す．

5.1　一覧表の見方

分類	項番	タグ項目	説明	具体化例	実証データ例	予定・実績の要否	工程別の要否	備考
基本情報	1	プロジェクト名	プロジェクトを一意に決定するための識別名		プロジェクト計画書 など			必須項目
	2	開発組織の情報	当該プロジェクトの開発を担当する組織の情報．一般には，受注者となる組織情報となる．	ISO/IEC 15504 Process Assessment，米カーネギメロン大CMMIなどを用いたアセスメント結果のプロセス達成度一覧または組織成熟度レベル	発注仕様書 アセスメント結果に関する情報 体制表			類似対象業種のプロジェクトに関するプロセス領域ごとの能力レベルの達成度など，合意のもとに詳細を共有することも可能
				開発組織名，対象業種の経験 など	キャリアシート など			
	3	開発プロジェクト情報	開発プロジェクトの特徴や当該タグデータの対象とするプロジェクトの種類を示す情報．タグデータの解釈や分析時に必要なデータ．	開発プロジェクト種別：新規・改造・保守・運用・流用など	要件定義書 プロジェクト計画書			開発作業を伴わない場合も入力
				設計書再利用率・ソースコード再利用率・コンポーネント再利用率など	品質計画書 設計ドキュメント	○		進捗情報の各成果物の規模から導出可能な場合もある
				開発プロジェクト形態（商用パッケージ，受託開発），利用パッケージ など	ソースコード テスト計画書 など			

*注）進捗情報タグ項目には，「工程別の要否」欄はない．進捗情報は，管理単位を自由に設定することが可能で，工程別・日次・週次などが考えられる．

5.2　標準タグ項目の位置づけ

- 利用目的やプロジェクトの特性に応じたカスタマイズを前提としている．従って，すべての項目がすべてのプロジェクトにおいて網羅的にタグ化されることを求めない．また，必要なデータの追加も可能とする．
- データ収集の対象物，対象範囲，粒度，収集期間（工程）は，別途ユーザ・ベンダ間で取り決めなどに準じ，本規格では特に規定しない．
- 収集データのタグ化のタイミングやユーザへの開示タイミングは，ユーザ・ベンダ間の協議などにより決定されるものとする．納品時，工程毎，一定時間ごと(週次，日次)などが考えられる．
- タグの階層や項目の繰り返しなどのタグの具体的構造については，システムやプロジェクトに応じてカスタマイズをする．（「4．タグの構造」参照）

- ソフトウェアタグとするデータの具体化レベルは，選択可能とする．エンピリカルデータ・生データ，一次加工データ（数値化されたデータ），二次加工データ（2 つの測定値の比率など）が考えられる．たとえば，進捗情報の「プログラミング->規模」は，プログラムコードそのもの，時系列のステップ数，予定規模に対する完成割合などが考えられる．この点についても，ユーザ･ベンダ間で協議などにより決定されるものとする．
- 他の項目から算出可能な数値は，項目から除外する．ただし，欠陥密度など一般的なものは１項目として挙げている．
- 他の文書（契約書等）に記載され，ユーザ・ベンダ間ですでに共有されている内容（品質要件，受発注金額など）は標準タグ項目から除外しているが，変更管理などのため追加することも可能である．

タグ項目一覧

プロジェクト情報

プロジェクトに関する情報・プロジェクト中に不変なデータ・プロジェクトを特徴付けるタグ

※下記一覧は，ユーザ・ベンダ間で共有するタグ項目の候補である．このなかから目的に応じてタグ項目を選択し，ソフトウェアタグを構成する．
※タグのデータを解釈・分析するために必要なデータや他のプロジェクトと比較するための情報を含む．
※必要に応じて，具体的なデータの取り方の情報を含む．

分類	項番	タグ項目	説明	具体化例	実証データ例	予定・実績の要否	工程別の要否	備考
基本情報	1	プロジェクト名	プロジェクトを一意に決定するための識別名		プロジェクト計画書 など			必須項目
	2	開発組織の情報	当該プロジェクトの開発を担当する組織の情報．一般には，受注者となる組織情報となる．	ISO/IEC 15504 Process Assessment，米カーネギメロン大CMMI®などを用いたアセスメント結果のプロセス達成度一覧または組織成熟度レベル	発注仕様書 アセスメント結果に関する情報 体制表 キャリアシート など			類似対象業種のプロジェクトに関するプロセス領域ごとの能力レベルの達成度など，合意のもとに詳細を共有することも可能
				開発組織名，対象業種の経験 など				
	3	開発プロジェクト情報	開発プロジェクトの特徴や当該タグデータの対象とするプロジェクトの種類を示す情報．タグデータの解釈や分析時に必要なデータ．	開発プロジェクト種別：新規・改造・保守・運用・流用など	要件定義書 プロジェクト計画書 品質計画書 設計ドキュメント ソースコード テスト計画書 など			開発作業を伴わない場合も入力
				設計書再利用率・ソースコード再利用率・コンポーネント再利用率など		○		進捗情報の各成果物の規模から導出可能な場合もある
				開発プロジェクト形態（商用パッケージ，受託開発），利用パッケージ など				
	4	顧客情報	当該システムのユーザ，もしくは第1発注者となる組織の情報．	顧客名，新規顧客か否か など	顧客との議事録 プロジェクト計画書 など			通常は，発注者自身なので不要．ただし，受注者からの外部委託を行う場合，最終顧客(エンドユーザ)情報を記入

システム情報	**5**	**システム構成**	**開発システム構成の特徴や当該タグデータの対象とするシステムの種類を示す情報．タグデータの解釈や分析時に必要なデータ．**	利用したサブシステムの安定度：利用ハードウェア、ライブラリ、OS等、調達したサブシステムの安定度合い（初回、不安定等）	基本設計書 プロジェクト計画書 工程管理表 勤務実績データ など			はじめて使うサブシステムには学習工数が必要なため
				サブシステムの検証期間/工数：利用を検討したサブシステムの検証期間と工数				検証時に性能把握や設計情報の細部を詰めることができるため、検証期間/工数から設計、テスト品質を推測するため
				利用したサブシステムに関する法的要件 など				
	6	**システム規模**	**開発システムの規模．計画値と最終実績値とする．進捗情報に同じ情報が含まれる場合は，省略可．**	受注者側の想定する顧客要件の数，実装した要件数など	基本設計書 ソースコード 要件定義書 など	○		
				工程終了時のソースコード行数,機能,FP,処理データ量,処理データ数など		○		SLOCはプログラム単位，その他はプロジェクト単位
開発情報	**7**	**開発手法**	**開発システム開発に用いたプロセスや手法についての情報．タグデータの解釈や分析時に必要なデータ．**	適用プロセスタイプ（ウォーターフォール・アジャイル・プロトタイプ），適用手法（OOAD，SASD，DDE）など	プロジェクト計画書 プロジェクト生産管理計画書 など			
	8	**開発体制**	**開発側の要員に関する情報．タグデータの解釈や分析時に必要なデータ． ※開示対象範囲は，発注者・受注者側での協議により決定する**	組織図・人員配置図 作業体制の複雑さ・親密さ(結合度・分散度)：組織図・人員配置図から計測 プロジェクト体制の階層数　など	体制表 発注書 キャリアシート 勤務実績データ 工程管理表 など		○	
				外注率，ピーク時の開発人員，延べ開発人員，チーム別の開発要員　など		○	○	
				経験年数，レベル別割合(上級・中級・初級)，新人比率， 類似プロジェクトの経験(開発、ユーザともに類似プロジェクト参加の経験があるか：経験なし、3回未満、5回以上)， 要員スキル（PM,業務，分析・設計，言語・ツール，開発プラットフォーム：ITSSなどに準拠）　など		○	担当機能毎，担当職務毎メンバの学習工数を推測するため
	9	**プロジェクト期間**	**当該プロジェクトの開発期間に関する情報**	稼働時期、開始、終了を含む（時間、日、月）など	勤務実績データ 工程管理表 など	○	○	
				工期 など		○	○	発注から納品までのリードタイム(=全体工期)

プロジェクトの階層構造情報	10	**親プロジェクト情報**	**本プロジェクトが別のプロジェクトのサブ(子)プロジェクトである場合，付加**	親プロジェクトのプロジェクト名	プロジェクト計画書 機能設計書 構造設計書 など			「No.1　プロジェクト名」と一致するプロジェクトのタグが存在しなければならない
	11	**サブ(子)プロジェクト情報**	**本プロジェクトがサブ(子)プロジェクトを持つ場合，その数やサブ(子)プロジェクトに関する情報**	サブ(子)プロジェクトの数，プロジェクト名など	プロジェクト計画書 機能設計書 構造設計書 など			
その他	12	**特記事項**	**その他，タグデータの解釈や分析時に必要，もしくは有用なデータ．**					タグデータ管理情報(適用管理指標、収集データ群)など

タグ項目 一覧

進捗情報

成果物や作業の進捗状況，成果物やプロセスの品質を表すタグ

※下記一覧は，ユーザ・ベンダ間で共有するタグ項目の候補である．このなかから目的に応じてタグ項目を選択し，ソフトウェアタグを構成する．
※[推移]は，時系列もしくは一定間隔で計測された連続データも想定される．
　計測間隔は，工程毎・日次・週次・開発のm/n(m=0,1,2,・・n-1 ／ n=16 or 32 ….)などが考えられる
※進捗に関する各計測項目は，必要に応じて全体／工程毎／タスク毎に測定する．
※必要に応じて，具体的なデータの取り方の情報を含む．

分類	項番	タグ項目	説明	具体化例	実証データ例	予定・実績の要否	備考
要件定義	13	ユーザヒアリング情報	要件に関してユーザに行ったヒアリングに関する情報	ユーザヒアリング実施件数(回) ユーザヒアリング項目数(件)， ユーザヒアリング回答率（ユーザヒアリング回答数÷ユーザヒアリング項目数） など	ユーザヒアリング議事録 ユーザヒアリング質問票 など	○	
	14	規模[推移]	開発側で作成した要件数	画面、機能項目、ユースケース、アクター，顧客要件，機能，FPなど	要件定義書 など	○	何を要件の基本単位とするかは、要合意事項
	15	変更[推移]	変更された要件数	規模の計測単位に依存	要件定義書 要件定義書の変更履歴 など		
設計	16	規模[推移]	設計成果物の規模 ※新規・改造・再利用(流用)毎に計測する	機能設計（ページ数・帳票数・画面数・ファイル数・項目数・UML図の数、クラス数、バッチプログラム数，重要な機能数など） 構造設計（データ項目数，DFDデータ数，DFDプロセス数，DBテーブル数 など） など	基本設計書 機能設計書 構造設計書 詳細設計書 など	○	
	17	変更[推移]	変更された設計成果物の数，もしくは変更量	規模の計測単位に依存	基本設計書 機能設計書 構造設計書 詳細設計書 各設計書の変更履歴 など		
	18	要件の網羅率	要件定義で作成された要件の実装率	設計に取り入れられた要件数÷要件数	要件定義書 基本設計書 機能設計書 構造設計書 詳細設計書 など		

プログラミング	**19**	**規模[推移]**	**プログラミング成果物の規模 ※新規・改造・再利用(流用)毎に計測する**	LOCなど	ソースコード など	○	ファイル別・モジュール別
	20	**変更[推移]**	**変更されたプログラムの数，もしくは変更量**	変更回数，削除行数，追加行数など	ソースコード リポジトリ更新履歴 など		ファイル別・モジュール別
	21	**複雑度**	**プログラムの品質（保守性）**	Halstead，McCabe，CK，クローン含有率，コメント平均利用率など	ソースコード など		ファイル別・モジュール別
テスト	**22**	**規模[推移]**	**テストの規模 ※新規・改造・再利用(流用)毎に計測する**	テスト項目数など	テストケース テスト計画書 など	○	
	23	**変更[推移]**	**変更されたテスト項目数や変更量**	追加テスト項目数，変更テスト項目数など	テストケース 変更履歴 など		
	24	**密度**	**テストの品質**	テスト項目数÷システム規模（ソースコード行数など）	テストケース テスト結果 ソースコード 要件定義書 など	○	「No.6　システム規模」と「No.22 規模」から導出可
				C0(命令網羅)，C1(分岐網羅)，要件に対するカバレッジ率（結合/受け入れテスト）など		○	
	25	**消化**	**テストの進捗，プログラムの品質**	テストケース進捗，消化テスト項目推移，テストプログラム消化率，クリティカルパス進捗など	テストケース テスト計画書 テスト結果 工程管理表 など	○	
品質	**26**	**レビュー状況**	**成果物(仕様書，設計書，プログラムコード，テスト仕様書など）のレビューに関する情報**	レビュー回数，対象規模，実績回数，ユーザ参画の有無，レビュー時間など	レビュー議事録 など	○	
	27	**レビュー作業密度**	**レビュープロセスの品質，もしくはレビュー対象の品質**	レビュー工数率，レビュー密度など	レビュー議事録 レビュー対象 (ソースコード， 各設計書） など		

	28	**レビュー指摘率[推移]**	**レビュープロセスの品質，もしくはレビュー対象の品質**	レビュー指摘数，指摘密度，不具合指摘数推移など	レビュー議事録 レビュー対象 (ソースコード， 各設計書) など		
	29	**欠陥件数[推移]**	**テスト設計の品質とコード品質**		テスト結果 障害管理票 品質管理表 など	○	欠陥が発生した工程別，原因ごとにも集計
	30	**欠陥対応件数**	**欠陥の対応進捗，対応内容**	不具合消化数，障害滞留時間，検出欠陥の分析実施回数，類似欠陥調査実施回数など	テスト結果 障害管理票 品質管理表 など		
	31	**欠陥密度**	**テスト設計の品質とコード品質**	検出欠陥数をシステム規模で割る：不良摘出件数÷ステップ数など	テスト結果 障害管理票 品質管理表 ソースコード など	○	「No.6　システム規模」と「No.29　欠陥件数」から導出可
	32	**欠陥指摘率**	**テスト設計の品質**	不良摘出件数÷テストケース消化数	テスト計画書 テストケース テスト結果 障害管理表 品質管理表 など	○	「No.25　消化」と「No.29　欠陥件数」から導出可
	33	**静的チェックの結果**	**プログラムの品質（保守性）**	チェッカによる総エラー数，チェッカによる総警告数など	ソースコード など		
工数	34	**作業工数**	**作業に要する工数，仕様変更作業工数**		工程管理表 勤務実績データ など	○	
	35	**生産性**	**工数に対する成果物の比率**	規模（画面数・帳票数）÷工数 頁÷工数(設計工程) ステップ数÷工数（製造工程） 不良摘出件数÷工数(テスト工程）など	構造設計書 勤務実績データ ソースコード テスト結果 など	○	「No.34　作業工数」と各成果物の規模などから導出可

計画・管理	**36**	**プロセス管理情報**	**開発プロセスの管理に関する情報**	プロセス規定度（提出されたプロセス記述の詳細さに基づく:WBSのタスク数など）， 標準プロセスに対する網羅度，プロセス遵守度（プロセス記述と実際の作業間の適合の度合いに基づく）， プロセス管理度（各ステップごとに設定された会議体の数や提供された管理指標の数に基づく） など	プロジェクト計画書 プロジェクト生産管理計画書 勤務実績データ 工程管理表 議事録 など		
	37	**会議実施状況**	**ユーザ・ベンダ間，ベンダ間での情報共有状況を把握**	各会議の時間、参加人数、資料量、議事数、議事録量、情報共有者数など	議事録 など		会議:要件検討会，進捗会議，納入/検収など
	38	**累積リスク項目数**	**リスク認識が十分であったかを把握**	進捗会議で挙げられたリスク項目数の累積、軽微、重大、その他の項目でわける	議事録 リスク管理表 など	○	
	39	**リスク項目の滞留時間**	**リスク対策が適切になされていたかを把握**	リスク管理項目の最長、平均滞留時間	議事録 リスク管理表 など		
その他成果物	**40**	**規模[推移]**	**対象成果物の規模 ※新規・改造・再利用(流用)毎に計測する**		ドキュメント 計画書 など	○	対象成果物:マニュアル，計画書，スケジュール表など
	41	**変更[推移]**	**変更された対象成果物の数，もしくは変更量.**	変更回数，変更ページ数など	ドキュメント変更履歴 など		

付録

体制

- ソフトウェアタグ規格技術委員会
  第 1 回（2007 年 7 月 9 日）　～　　第 13 回（2008 年 10 月 14 日）
  構成員　14 組織　31 人

  アドバイザー（50 音順・敬称略）
  大杉　直樹　　株式会社 NTT データ
  木谷　　強　　株式会社 NTT データ
  鶴保　征城　　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（IPA SEC）
  神谷　芳樹　　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（IPA SEC）
  村山　浩之　　株式会社デンソー

  企業（50 音順・敬称略）
  岩崎　新一　　日本電気株式会社
  片平　真史　　独立行政法人宇宙航空研究開発機構（JAXA）
  清田　辰巳　　株式会社東京証券取引所
  阪井　　誠　　株式会社 SRA 先端技術研究所
  高木　富士雄　シャープ株式会社
  福地　　豊　　株式会社日立製作所
  松尾　昭彦　　株式会社富士通研究所
  松村　好高　　株式会社 SRA 先端技術研究所
  宮本　祐子　　独立行政法人宇宙航空研究開発機構（JAXA）
  森　　俊樹　　株式会社東芝
  山田　　淳　　株式会社東芝

  大学（50 音順・敬称略）
  飯田　　元　　奈良先端科学技術大学院大学
  井上　克郎　　大阪大学
  川口　真司　　奈良先端科学技術大学院大学
  楠本　真二　　大阪大学
  久保　浩三　　奈良先端科学技術大学院大学
  黒崎　　章　　元 奈良先端科学技術大学院大学
  小柴　昌也　　奈良先端科学技術大学院大学
  角田　雅照　　奈良先端科学技術大学院大学
  鳥居　宏次　　奈良先端科学技術大学院大学
  名倉　正剛　　奈良先端科学技術大学院大学
  松村　知子　　奈良先端科学技術大学院大学
  松本　健一　　奈良先端科学技術大学院大学
  森崎　修司　　奈良先端科学技術大学院大学
  門田　暁人　　奈良先端科学技術大学院大学
  リビエリ・シモネ　　大阪大学

- ソフトウェア構築可視化法的問題検討委員会
  第 1 回（2007 年 11 月 16 日）　～　　第 5 回（2008 年 10 月 8 日）
  構成員　7 組織　11 名（50 音順・敬称略）

青江　秀史	大阪大学
飯田　　元	奈良先端科学技術大学院大学
井上　克郎	大阪大学
江口　順一	帝塚山大学
大西　幸男	西銀座法律事務所
楠本　真二	大阪大学
久保　浩三	奈良先端科学技術大学院大学
小柴　昌也	奈良先端科学技術大学院大学
茶園　成樹	大阪大学
松田　貴典	大阪成蹊大学
松本　健一	奈良先端科学技術大学院大学

参考文献

[1] ISO9001：2000（JIS Q 9001/2000） 品質マネジメントシステム

[2] 開発のための CMMI®(CMMI-DEV) 1.2 版　公式日本語翻訳版，CMMI 成果物チーム，技術報告書 CMU/SEI-2006-TR-008.
http://www.sei.cmu.edu/cmmi/translations/japanese/models/dev-v12-abstract-j.html

[3] 保田勝通，ソフトウェア品質保証の考え方と実際，日科技連，1995 年 11 月

[4] IPA/SEC，ＩＴプロジェクトの見える化　下流工程編，日経 BP，2007 年 6 月

[5] IPA/SEC，IT プロジェクトの見える化　上流工程編，日経 BP，2007 年 5 月

[6] IPA/SEC 著作監修，「ソフトウェア開発データ白書 2005～IT 企業 1000 プロジェクトの定量データを徹底分析」，日経コンピュータ

[7] IPA/SEC，ソフトウェア開発データ白書 2007，日経 BP，2007 年 7 月

[8] XBRL　FACTBOOK :
http://www.xbrl-jp.org/download/pdf/factbook/2007/XBRLFACTBOOK_ver.9.pdf

[9] 飯塚悦功編：“ソフトウェアの品質保証-ISO9000-3 対訳と解説-”，日本規格協会(1992).

[10] “品質を考える技術”，組込みプレス，5, pp.13-34(2007).

[11]「情報システムの信頼性向上のための取引慣行・契約に関する研究会～情報システム・モデル取引・契約書～（受託開発（一部企画を含む），保守運用）＜第一版＞」（2007 年 4 月 13 日公表，経済産業省）

[12]「情報システムの信頼性向上に関するガイドライン」（2006 年 6 月 15 日公表，経済産業省）

[13]「情報システムの信頼性向上に関する評価指標（試行版）」（2007 年 4 月 13 日公表，経済産業省）

[14]「信頼性向上ベストプラクティスを実現する管理指標調査報告書」（2008 年 4 月 社団法人情報サービス協会）

ソフトウェアタグ規格　第 1.0 版
2008 年 10 月 14 日
StagE 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
StagE プロジェクト研究代表者
松 本　健 一

大阪大学　大学院情報科学研究科
ソフトウェアタグ規格技術委員会委員長
井 上　克 郎

E-mail:stage-contact@is.naist.jp
URL:www.stage-project.jp

注）StagE プロジェクトは，文部科学省の委託業務「次世代 IT 基盤構築のための研究開発：ソフトウェア構築状況の可視化技術の開発普及」として，奈良先端科学技術大学院大学と大阪大学が共同で実施する研究開発プロジェクト「エンピリカルデータに基づくソフトウェアタグ技術の開発と普及」の略称です．

ソフトウェアタグ・ガイドブック 2010

2010 年 1 月 31 日

StagE 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

StagE プロジェクト研究代表者

松本健一

E-mail: stage-contact@is.naist.jp

URL: http://www.stage-project.jp

本冊子は，文部科学省の委託業務「次世代 IT 基盤構築のための研究開発：ソフトウェア構築状況の可視化技術の開発普及」として，奈良先端科学技術大学院大学と大阪大学が共同で実施する研究開発プロジェクト「エンピリカルデータに基づくソフトウェアタグ技術の開発と普及」の成果をとりまとめたものです．StagE は，本研究開発プロジェクトの略称です．


StagE
StagEプロジェクト