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。

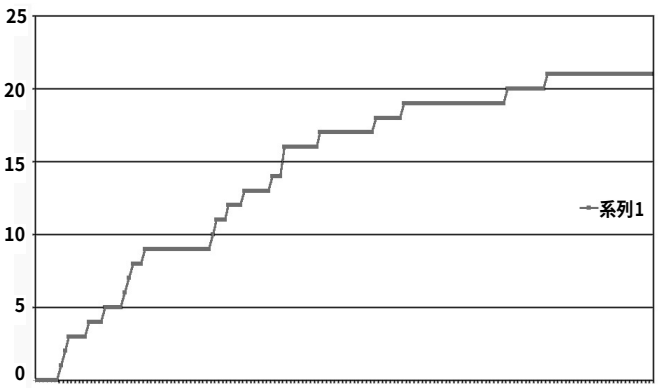

図1　レビュー回数（項番26）のデータの推移

※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

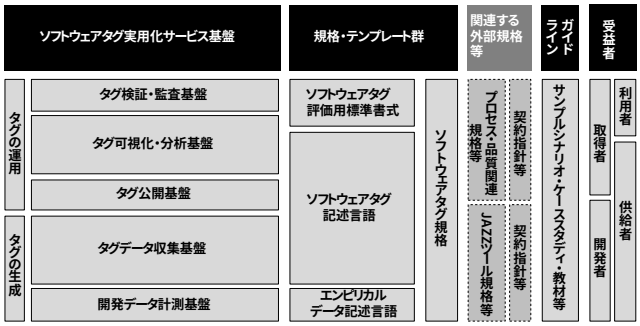

図2　ソフトウェアタグ実用化技術俯瞰図（簡略版）

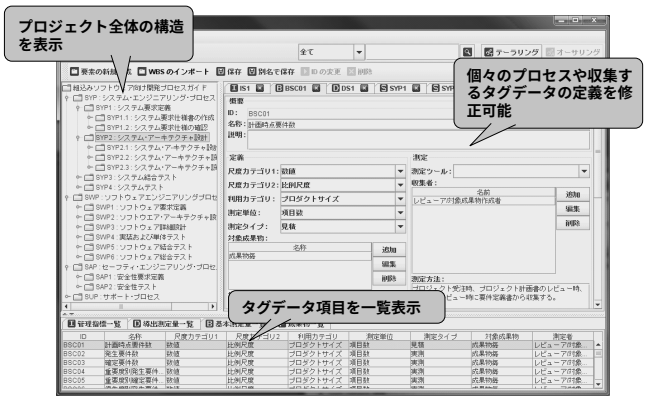

図3　タグデータ定義画面例

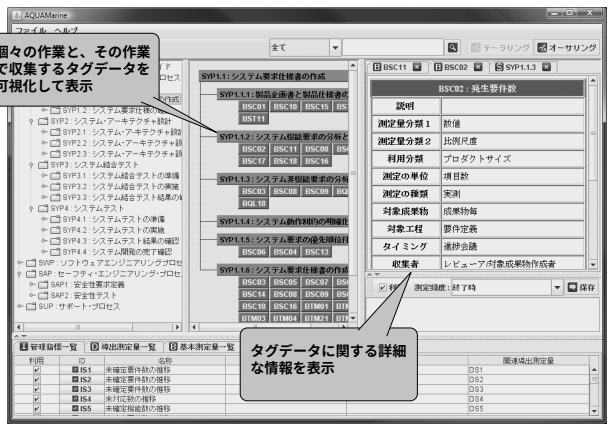

図4　策定したプロジェクト計画の閲覧画面例

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）

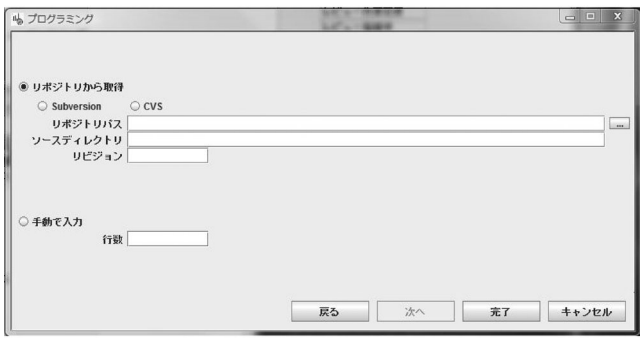

図7　画面例3（規模の自動収集・手動入力設定）

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

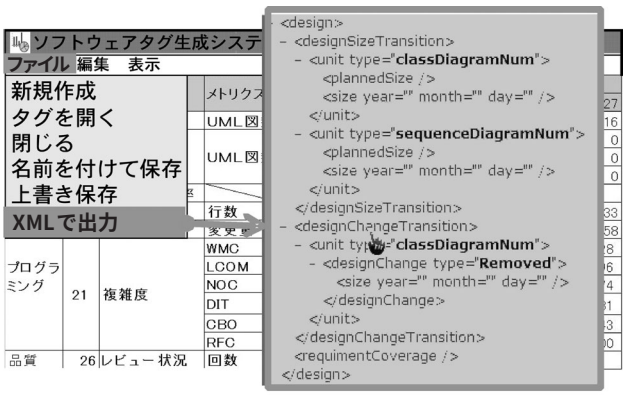

図8　XML形式への出力イメージ

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、

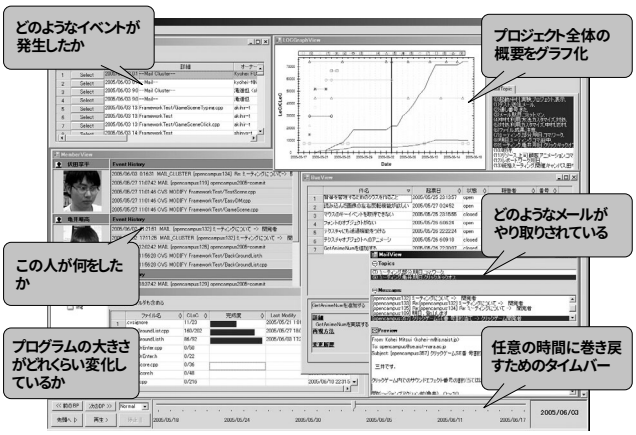

図9　タグ・リプレイヤーのメイン画面

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