ソフトウェアタグデータ収集システムの試作

西川 倫道　鵜久森 将隆　山田 慎也　楠本 真二
大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻
E-mail: tm-niskw@ist.osaka-u.ac.jp

概要

近年，発注者の視点に立った開発プロジェクトデータの活用が求められている．そこで，発注者に製品品質や開発状況を把握可能にするソフトウェアトレーサビリティの概念が提案され，これを実現する技術としてソフトウェアタグの研究開発が行われている．本研究ではソフトウェアタグ実現の一環として，ソフトウェア開発中に作成される種々のデータからメトリクスを収集し，ソフトウェアタグデータを得るシステムを提案する．さらに，システムに基づいて開発したプロトタイプを実際のプロジェクトデータに適用してソフトウェアタグを作成し，ソフトウェアタグの有用性を確認した．

1. まえがき

近年，ソフトウェアの大規模化と社会的影響の拡大によりソフトウェアの重要性が高まると共に，様々な問題が発生している [2]．例えば，ソフトウェアに障害が発生することで，金融システムや交通システムなどの重大な社会インフラが停止したり，航空管制システムや自動車安全システムなどが人命に関わる危険を引き起こす．また，それに伴いユーザ・ベンダに莫大な経済的損失を与える．他にも，開発期間の短縮のためにコストの低減や生産性の向上が要求されたり，情報システム開発に関わる訴訟が頻繁に発生している [20]．

これらの問題を解決するアプローチの一つとして，ソフトウェアトレーサビリティの概念が提案されている．トレーサビリティとは本来，野菜や食肉等の物品の生産・流通履歴を確認可能であることを意味するが，これをソフトウェアに対し適用することでソフトウェアユーザによるソフトウェアの製品品質・開発状況の把握や，ユーザ・ベンダ間の法的係争発生時の処理の短期化を可能とすることが期待されている．

ソフトウェアトレーサビリティを実現するためにソフトウェアタグの研究開発が行われている [18]．ソフトウェアタグとはソフトウェア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であり，ユーザはソフトウェアタグを見ることでソフトウェアがいつ，どこで，誰に，どのように開発されたか把握可能になる．

ソフトウェアタグによるソフトウェアトレーサビリティの実現には様々な課題があり，その一つにソフトウェアタグとなるデータを収集するツールの開発がある．そこで本研究ではソフトウェアトレーサビリティ実現の一環として，タグデータ収集システムの提案を行う．具体的には，システムへの要求に応える設計方針を検討し，システム構成を設計した．また，プロトタイプを開発し実際のソフトウェア開発プロジェクトに適用してソフトウェアタグを作成した．そして作成したソフトウェアタグからユーザがどのようにソフトウェアの品質や開発状況を把握するのか検討した．

2. ソフトウェアトレーサビリティの実現

2.1. ソフトウェア受発注における問題

ソフトウェア開発においてユーザは要件定義などの上流工程からベンダ任せにするケースが従来多かった [8]．しかし，そういった開発では，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，納期遅れなどをもたらす．また，障害発生時には，ユーザによる原因や

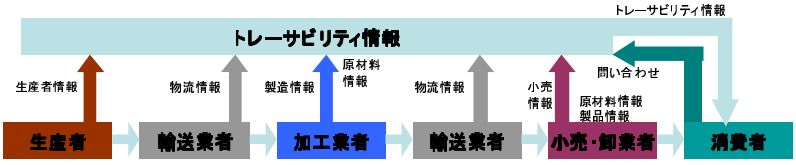

図 1. 食品のトレーサビリティ [18]

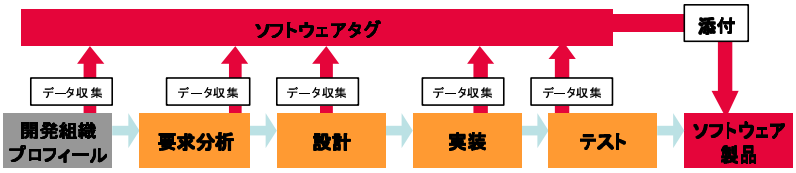

図 2. ソフトウェアトレーサビリティ [18]

責任所在の追究が困難で，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，解決まで発注者・開発者双方の被害が増大する．

以上から近年，ソフトウェア開発プロジェクトにおいてユーザの参画が重要視されてきている．しかし，これまでソフトウェア開発におけるデータ収集・分析はベンダ側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，ユーザの視点に立って行われてこなかった．そこで，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いる．

2.2. ソフトウェアトレーサビリティ

2.1 節の問題の解決策としてソフトウェアトレーサビリティの概念が提案されている．トレーサビリティとは，物品の生産・流通履歴を確認可能であることを意味し，「追跡可能性」とも言われている．近年，食の安全上の問題から食品に対するトレーサビリティへの取り組みが活発に行われており，生産・処理・加工・流通・販売等の各段階で，情報の追跡・遡及が可能になるよう推し進められている (図 1).

トレーサビリティの概念をソフトウェア開発に適用したソフトウェアトレーサビリティでは，ソフトウェア開発過程で作成される実証データからソフトウェアタグを作成し，ソフトウェア製品に添付して流通させる (図 2). 製品に添付されたソフトウェアタグからユーザはソフトウェアの製品品質や開発状況を把握する．

2.3. StagE プロジェクト

StagE(Software Traceability and Accountability of Global Software Engineering) プロジェクト [18] とは，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：ソフトウェア構築状況の可視化技術の開発と普及」の略称として，平成 19 年度から奈良先端科学技術大学院大学と大阪大学が産学官連携で進めているプロジェクトである．ソフトウェアトレーサビリティの実現を目的とし，ソフトウェアタグの研究開発を行っている．

2.4. ソフトウェアトレーサビリティ実現上の課題

ソフトウェアタグによるソフトウェアトレーサビリティの実現には以下の課題がある．

- ソフトウェアタグの規格化
- タグデータ収集ツールの開発
- タグデータの可視化・評価手法の開発
- 実証実験の実施
- 法的諸問題の検討

表 1. タグ項目構成要素

要素名	説明
項番	タグ項目に割り当てられる通し番号
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・タグデータのインデックス
説明	各タグ項目の意味や活用方法など
具体化例	タグデータの例
実証データ例	タグデータを抽出する元となるプロジェクト情報やドキュメント・生データ
予定・実績の要否	予定・実績値の差分による管理を行うデータ
工程別の要否	工程別に管理を行うデータ
備考	備考欄

表 2. ソフトウェアタグ

プロジェクト情報(12項目)

分類	項番	タグ項目
基本情報	1	プロジェクト名
	2	開発組織の情報
	3	開発プロジェクト情報
	4	顧客情報
システム情報	5	システム構成
	6	システム規模
開発情報	7	開発手法
	8	開発体制
	9	プロジェクト期間
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

進捗情報(29項目)

分類	項番	タグ項目
要件定義	13	ユーザヒアリング情報
	14	規模[推移]
	15	変更[推移]
設計	16	規模[推移]
	17	変更[推移]
	18	要件の網羅率
プログラミング	19	規模[推移]
	20	変更[推移]
	21	複雑度
テスト	22	規模[推移]
	23	変更[推移]
	24	密度
	25	消化
品質	26	レビュー状況
	27	レビュー作業密度
	28	レビュー指摘率[推移]
	29	欠陥件数[推移]
	30	欠陥対応件数
	31	欠陥密度
	32	欠陥指摘率
	33	静的チェックの結果
工数	34	作業工数
	35	生産性
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
その他成果物	40	規模[推移]
	41	変更[推移]

このうち，本研究ではタグデータ収集ツールの開発を目標としている．

3. ソフトウェアタグ

3.1. 概要

現在，StagE プロジェクトではソフトウェアタグ規格 第 1.0 版が策定されている [17]．この規格においてソフトウェアタグは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．タグ項目はユーザ・ベンダ間で共有する候補として定義してあり，ユーザ・ベンダは用途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．

タグ項目は全 41 項目あり，プロジェクト情報と進捗情報に大別される．また，各タグ項目は分類，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，備考を要素としてもつ．各要素の説明を表 1 に示す．なお，予定・実績の要否とあるが，成果物やプロセス，目標についてベースラインを作成することはソフトウェア開発の改善において重要であるため [3]，いくつかのタグ項目に対し予定と実績による管理を要求している．また，簡単のためにソフトウェアタグを分類，項番，項目名のみ表 2 に示す．

3.2. プロジェクト情報

ソフトウェアタグの内，プロジェクト情報とは開発プロジェクトおよびシステムの基本的な情報を指す．プロジェクト情報はプロジェクトの途中で変更がなくプロジェクト全体を通して不変である．プロジェクト情報内でタグ項目は 12 個あり，基本情報，システム情報，開発情報，プロジェクトの階層構造情報，その他の 5 つに分類される．

3.3. 進捗情報

ソフトウェアタグの内，進捗情報とはソフトウェア開発によって得られた作業の進捗状況や，成果物やプロセスの品質を表す情報を指す．進捗情報はプロジェクト情報とは異なりプロジェクト途中で変化する．進捗情報内でタグ項目は 29 項目あり，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

の 8 つに分類される．なお，タグ項目名の後ろに [推移] という語句が付くものは，時系列もしくは一定間隔で連続するデータが想定されるタグ項目である．

3.4. タグ利用のシナリオ

ソフトウェアタグ定義をユーザ・ベンダがどのように利用するのか一例を述べる．ソフトウェアタグの内，19 番プログラミングの規模としてソースコードからコード行数を，20 番プログラミングの変更としてソースコード行数の差分量を選択したとする．これにより，ユーザは実装時にコード行数がどのように推移して作成されたか，また，日々のコード行数の推移がどれくらいの変更・削除・追加によって行われたかを把握可能になる．他に，30 番の欠陥対応件数として不具合消化数を選択すれば，不具合を修正するためにどの程度ソースコードの修正を行ったかも分かる．

4. ソフトウェアタグデータ収集システム

4.1. 設計方針

ソフトウェアタグ収集システムに対する要求をどのように実現するか述べる．

まず，多くのソフトウェアタグデータに対応し収集可能であることが要求される．そこで，基本的なメトリクスの収集機能は収集システムが提供するとし，もし提供するメトリクスで対応出来ない場合は，システム利用者側でメトリクス収集機能を用意すれば，ツールに組み込み可能にする．

また，低コストでタグデータが収集可能であることも要求される．これには，極力タグデータはシステムが自動収集することで対応する．収集システムを用いるだけでソフトウェアタグが作成でき，余計な作業を必要とさせない．

タグデータ収集後に得られたソフトウェアタグが分析しやすいものであることも要求される．これは，タグデータ収集後に，ユーザが収集結果を理解しやすいよう視覚化等の分析が行われるためである．そこで，タグデータ分析ツールが利用しやすい形式でタグデータを出力することで対応する．

4.2. システム構成

4.1 節の方針に基づき設計したソフトウェアタグデータ収集システムの構成について述べる．

収集システムへの入力は，ソフトウェア開発時に発生する日々の開発履歴データと収集設定とする．日々の開発履歴データとは例えば，日々一定間隔でバックアップをとったプロジェクト開発フォルダや，構成管理ツールで管理されたデータなどである．また，入力として開発履歴データを得ただけでは，どのタグ項目を何から収集するのか判別できないため，収集対象と収集方法の特定を行う必要がある．そこで，これらの情報を特定するためにシステム利用者に収集設定の入力を求める．具体的には以下の項目を選択してもらう．

- 収集したいタグ項目は何か
- どの実証データから収集を行うか
- 収集対象はどのような開発支援ツールを介して得られたデータか
- 収集対象のデータ形式は何か

システムの出力はソフトウェアタグデータである．出力形式はタグデータ分析ツールの利用に適したものとする．

収集ツールではシステム利用者が使用したいメトリクスを提供していない場合が 2 通り考えられる．一つは，プログラムを作成すればメトリクス計測可能だが，収集システムでは提供していない場合である．もう一つは，メトリクスの計測が手作業でしか出来ない場合である．例えば，定型フォーマットの存在しない文章ファイルからメトリクスを計測する場合が挙げられる．前者に対してはメトリクス計測プログラムをシステム利用者側で作成してもらうことで対応する．プログラム作成には収集システムが API を提供し，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定することでプラグインとして導入可能にする．後者に対しては，利用者側でメトリクス計測後に特定のファイル形式で計測データを作成してもらう．そして，収集方法の設定時に入力データとして指定することで対応する．例えば，csv 形式や xml 形式で作成された計測データを指定し，収集したいタグデータとどのように対応をとるか設定する．

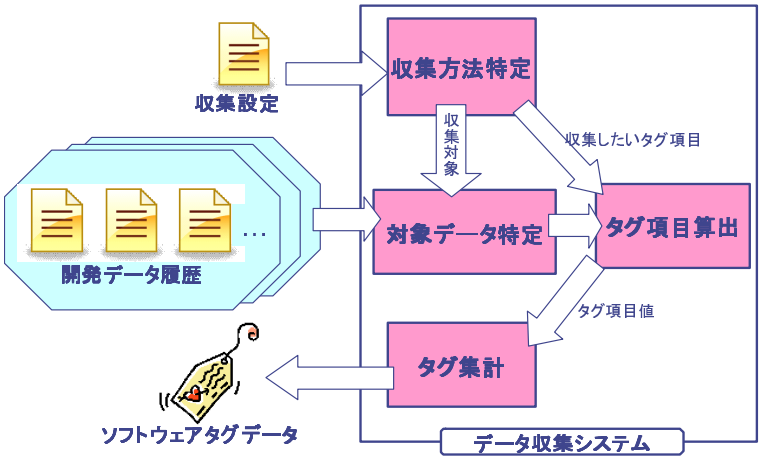

図 3. データ収集システム概要

図 3 に設計したソフトウェアタグデータ収集システムの概要図を示す．開発データ履歴を入力とし，また，システム利用者から収集方法を設定してもらうことでソフトウェアタグデータを出力する．データ収集システム内部は収集方法特定，対象データ特定，タグ項目算出，タグ集計の 4 つのサブシステムで構成する．各サブシステムについて説明する．

4.2.1 収集方法特定サブシステム

収集設定をシステムユーザに入力してもらい収集方法を特定する．特定した情報のうち，タグ項目に関する情報はタグ項目算出サブシステムへ，収集対象となる実証データの情報は対象データ特定サブシステムへと送る．

4.2.2 対象データ特定サブシステム

収集方法特定サブシステムから送られた収集対象の情報を用いて，開発データ履歴から収集対象を特定する．例えば，収集対象となる実証データがどのような構成で履歴内に存在するのか，構成管理ツールで管理されているのかなどを判別し，各状況に即した特定を行う．特定した実証データはタグ項目算出サブシステムへと送る．

4.2.3 タグ項目算出サブシステム

タグ項目ごとに実証データからメトリクスを計測する．収集方法特定サブシステムから送られたタグ項目情報をもとに使用するメトリクスを決定し，対象データ特定サブシステムで得られた実証データから計測を行う．計測結果はタグ項目ごとにタグ集計サブシステムへと送る．

4.2.4 タグ集計サブシステム

タグ項目算出サブシステムから得られた各メトリクス値を集計し，ソフトウェアタグデータとして出力する．この際，タグデータ分析ツールが利用しやすい形式にデータ変換を行う．

4.3. プロトタイプシステムの開発

4.3.1 対象とするタグ項目

ソフトウェアタグ定義ではタグ項目の各メトリクス値として実際に何をとるかはユーザ・ベンダ間の協議で決めるものとし，具体化例を挙げるに留まっている．そこで，まずどのタグ項目に対し何をタグデータとして収集するのかを決定し，そのタグデータをどのような実証データから収集するかを決定する．今回開発したプロトタイプでは進捗情報の 10 項目を対象とし，各項目で何をタグデータとするかは表 3 のように選択した．

表 3. プロトタイプが収集するタグデータ

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

4.3.2 タグデータ収集の実現方法

特定のプロジェクトに特化せず，多くのプロジェクトに対応可能なデータ収集を行うことを目指した．そこで，まず開発現場でよく使用される開発支援ツールを調査し [7]，どうすればそれらツールから包括的にタグデータを収集できるか検討した．ソフトウェアタグの分類ごとにタグデータ収集方法を述べる．

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．一般的にユースケース図の作成時には UML モデリングツールが使用される．実際の開発現場では UML モデリングツールとして，JUDE[13]，Rational Rose[11]，Microsoft office Visio[15] などが主に利用されている．これら UML モデリングツールは作成したモデルを XMI 形式 [21] で出力するという共通した機能をもつ．XMI(XML Metadata Interchange) は，抽象的なモデルをツール間で交換可能にするために OMG[16] が策定した標準規格である．そこで，この機能に着目し，XMI 出力されたファイルからユースケース図の情報を抽出する．

設計では，要件定義同様に UML 図作成のために UML モデリングツールが使用されるので，UML モデリングツールから XMI 出力されたファイルから UML 図の数を抽出する．

プログラミングでは，一般的に構成管理ツールが使用される．構成管理ツールとはソースコードやドキュメントの日々の進捗を管理するツールであり，Subversion[19] や CVS[6],[14] が知られている．これら構成管理ツールはプロジェクトデータを作業リポジトリに登録して用いられる．そこで，この作業リポジトリから日々登録されたソースコードを取得し，解析を行う．

品質では，バグ管理ツールが使用される．EPM[4] はソフトウェア開発で発生するデータからプロジェクトの情報可視化を行うツールであるが，データ収集対象となるバグ管理ツールとして bugzilla[10] や影舞 [9] を挙げている．これらバグ管理ツールは登録されたバグを様々な条件で検索し，結果を CSV 形式で出力する機能がある．そこで，この機能に着目し csv 出力されたバグ検索結果から不具合数，不具合消化数といった品質情報を求める．

5. 適用事例

5.1. 適用の目的

ユーザがソフトウェアタグからソフトウェアの品質やプロジェクトの開発状況が把握可能であるかを調べることが適用の目的である．そこで，プロトタイプでは収集しない他のタグ項目についても手作業でメトリクスを収集し，ソフトウェアタグの効用をはかる．

5.2. 対象プロジェクト

ITSpiral[12] で作成された実プロジェクト教材に対してプロトタイプを適用する．これは大学の履修登録プログラムを開発した際に得られた開発データであり，要件定義，設計，実装，テスト等の開発プロセスでさまざまな実証データが存在する．開発期間は 5ヶ月で，総データ量は 450MB になる．実プロジェクト教材で収集されているデータを表 4 に示す．

5.3. 実プロジェクトデータのタグ

プロトタイプでは進捗情報 29 項目の内，10 項目に対応している．実プロジェクトデータから手作業で他のタグ

表 4. 適用対象プロジェクトで収集されているデータ

プロセス名	収集データ
要件定義	要求仕様書，業務フロー図
要求	ユースケース図，アクター定義書，品質要求書
分析・設計	クラス図，コラボレーション図，シーケンス図
	アーキテクチャ仕様書，エンティティクラス設計書
	コントロールクラス設計書，バウンダリクラス設計書
実装	ソースコード，API ドキュメント，
	静的チェック記録（checkstyle,FindBugs,PMD,CPD,Jdepend）
テスト	テスト計画書・仕様書
プロジェクト管理	内部・外部レビュー記録，プロジェクト管理記録
	変更経緯説明書

表 5. 手作業により得られたソフトウェアタグデータ

分類	項番	タグ項目	タグデータ
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
テスト	22	規模 [推移]	テストケース数
	23	変更 [推移]	変更されたテストケース数
	24	密度	テストケース数/全体行数
	25	消化	消化テスト数
品質	26	レビュー状況	レビュー回数
	27	レビュー作業密度	レビュー時間/全体時間
	28	レビュー指摘率 [推移]	レビュー指摘数
	32	欠陥指摘率	欠陥件数/消化テスト数
	33	静的チェックの結果	FindBugs 指摘件数
工数	34	作業工数	作業時間
	35	生産性	行数/作業時間

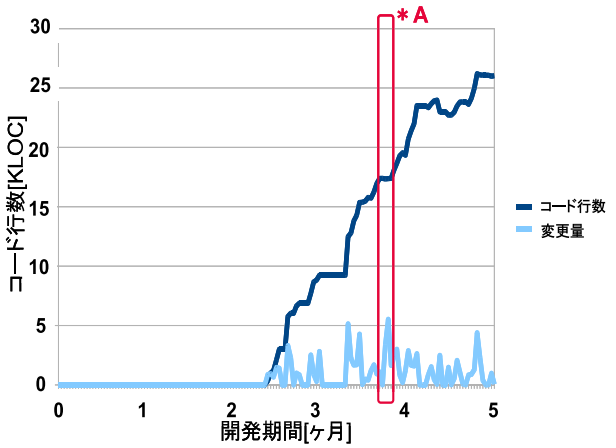

図 4. ソースコード行数と変更量の推移

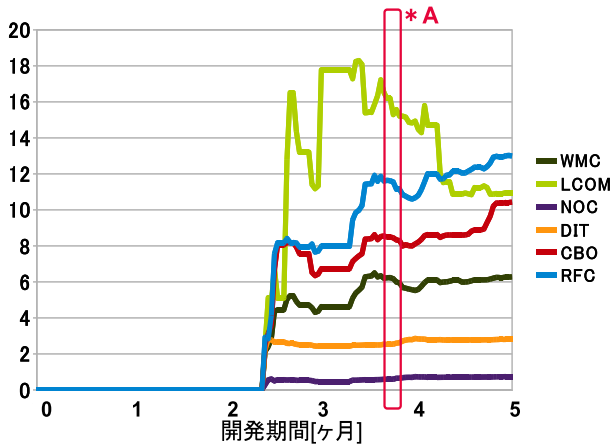

図 5. CK メトリクスの推移

項目についてメトリクスを収集した結果，進捗情報のうち表 5 の 12 項目のタグデータを新たに得た．計測したメトリクス値を一部省略して表 6 に示す．

5.4. タグデータによる開発状況の把握

計測結果から，いくつかのタグデータをグラフ化したものを示す．図 4 はソースコード全体のコード行数と変更量の推移を表し，図 5 は CK メトリクス [1] の推移を表したものである．

*A の期間において，コード行数はほとんど変化していないが，変更量が大きい．そこで，同期間において CK メトリクスの推移を見ると，減少していることが分かる．CK メトリクスはプログラムの複雑度を表すメトリクスである．そこで，この期間では新たに追加や削除が行われたのではなく，リファクタリングが行われたのではないかと推測できる．実際に構成管理ツールからソースコードの変更内容を調べたところ，リファクタリングの

表 6. 進捗情報計測結果 (一部省略)

分類	項番	タグ項目	メトリクス		開始日									終了日
					3/27	12/9	12/10	12/11	12/12	12/13	12/14	2/25	2/26	2/27
要件定義	13	ユーザヒアリング情報	ユーザヒアリング回数											3
	14	規模[推移]	ユースケース図数		11	12	12	12	12	12	12	12	12	12
	15	変更[推移]	ユースケース図数	追加	11	0	0	0	0	0	0	0	0	0
				変更	0	0	0	0	0	0	0	0	0	0
				削除	0	0	0	0	0	0	0	0	0	0
設計	16	規模[推移]	UML図数		2	114	114	114	114	114	114	116	116	116
	17	変更[推移]	UML図数	追加	2	0	0	0	0	0	0	0	0	0
				変更	0	0	0	0	0	0	0	0	0	0
				削除	0	0	0	0	0	0	0	0	0	0
	18	要件の網羅率												
プログラミング	19	規模[推移]	行数(全体)		0	0	354	994	1207	2093	2996	26068	25988	26033
	20	変更[推移]	変更量(追加＋削除行数)		0	0	866	946	664	1471	1400	21	974	58
	21	複雑度	WMC		0	0	2.14	2.4	2.86	4.35	4.45	6.27	6.27	6.28
			LCOM		0	0	2.5	5.12	4.86	6.74	5.1	10.96	10.96	10.96
			NOC		0	0	0.29	0.56	0.61	0.5	0.57	0.74	0.74	0.74
			DIT		0	0	2.21	2.8	2.79	2.65	2.67	2.81	2.81	2.81
			CBO		0	0	2.5	3.16	3.57	6.68	8.05	10.41	10.41	10.43
			RFC		0	0	2.93	2.84	4.14	7.56	8.17	13.01	13.02	13
テスト	22	規模[推移]	テストケース数	予定	0	76	76	76	76	187	187	587	613	597
	23	変更[推移]	テストケース数	予定	0	0	0	0	0	111	0	33	26	−16
				実績	0	0	0	0	0	0	0	174	140	68
	24	密度	テストケース数/全体行数		0.0000	0.0000	0.2147	0.0765	0.0630	0.0893	0.0624	0.0225	0.0236	0.0229
	25	消化	消化テスト数	実績	0	0	0	0	0	0	0	377	517	585
品質	26	レビュー状況	レビュー回数		0	17	17	17	17	17	17	21	21	21
	27	レビュー作業密度	レビュー時間/全体時間		0	0.92	0.92	0.92	0.92	0.92	0.92	1	1	1
	28	レビュー指摘率[推移]	レビュー指摘数		0	599	599	599	599	599	599	649	649	649
	29	欠陥件数[推移]	全体欠陥件数		0	0	0	0	0	0	0	19	19	19
	30	欠陥対応件数	全体欠陥対応件数		0	0	0	0	0	0	0	19	19	19
	31	欠陥密度	全体欠陥件数/行数		0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000729	0.000731	0.000730
	32	欠陥指摘率	全体欠陥件数/消化テスト数		0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0504	0.0368	0.0325
	33	静的チェックの結果	FIndBugs指摘件数		0	0	0	0	0	0	6	52	52	52
工数	34	作業工数	作業時間(日)											152
	35	生産性	行数/作業時間											171.3
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]												

実施を確認した．このように，ソフトウェアタグを用いることでユーザは適用対象プロジェクトの特徴を把握，分析可能である．

6. 考察

プロトタイプの開発からソフトウェアタグの作成を通して，どの程度の粒度でタグデータを収集すべきかという問題があった．これは，例えばプログラミングの規模としてコード行数を収集するとした場合に，サブシステム毎に計測するのか，またはファイルやクラス毎に計測するのかという問題である [5]．ソフトウェアタグ定義において収集するデータの粒度はユーザ・ベンダ間の協議により決定するとし，特に規定していない．しかし，ユーザ・ベンダの取り決めるあらゆる粒度に応えるタグデータ収集ツールを実現することは困難である．そこで，ソフトウェアタグ規格において，タグ項目の定義に加えて，収集するデータの粒度についても規定するほうが，タグデータの収集やその後の視覚化，分析において対応が取りやくなると考える．

7. あとがき

本論文では，ソフトウェアトレーサビリティの実現を目的としてソフトウェアタグ収集システムを設計した．さらに，設計したシステムに基づいて開発したプロトタイプを実際の開発プロジェクトデータへ適用し，ソフトウェアタグの有用性を確認した．

今後の課題としては，以下があげられる．

- 対応外メトリクスへの対応
  ソフトウェアタグ収集システムでは対応しないメトリクスを，ユーザが組み込める機能を実装する．
- 他開発プロジェクトへの適用
  他のプロジェクトデータに適用実験を行い，新たな知見や問題点を得ることで，ソフトウェアタグ収集システムを改良する．

謝辞

本研究は，文部科学省「次世代 IT 基盤構築のための研究開発」(研究開発領域名：ソフトウェア構築状況の可視

化記述の開発と普及）の委託に基づいて行われている．また，文部科学省科学研究費補助金基盤研究 (C)(課題番号:20500033) の助成を受けている．

参考文献

[1] S. R. Chidamber, C. F. Kemerer, “A Metrics Suite for Object-Oriented Design”, IEEE Trans. Software Eng., vol.20, pp.476-493, 1994.

[2] 井上 克郎, 松本 健一, 鶴保 征城, 鳥居 宏次, “実証的ソフトウェア工学研究への取り組み”, 情報処理, Vol.45, No.7, pp.722-728, 2004

[3] 松本 健一, “エンピリカルソフトウェア工学の現状と展望：SEL が残した 13 の教訓 ”, SEC journal, No2, pp.6-13, 2005

[4] 大平 雅雄, 横森 励士, 阪井 誠, 岩村 聡, 小野 英治, 新海 平, 横川 智教, “ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム ”, 電子情報通信学会論文誌, Vol.J88-D-I, No.2, pp.228-239, 2005

[5] Qiang Tu, Michael W. Godfrey, “An Itegrated Approach for Studying Architectural Evolution ”, Proceedings of the 10th International Workshop on Program Comprehension, pp.127-136, 2002

[6] Thomas Zimmermann, Peter Weisgerber, “Preprocessiong CVS Data for Fine-grained analysis ”, 1st International Workshop on Mining Software Repositories, pp.2-6, 2004

[7] “特集 1：開発支援ツールの利用実態”, 日経 SYSTEMS, 2008 年 6 月号, pp.11-39, 2008

[8] “特集 1：ユーザーの言い分ベンダーの言い分”, 日経コンピュータ, 2008 年 10 月 1 日号, pp.40-55, 2008

[9] バグトラッキングシステム影舞
http://www.daifukuya.com/kagemai/

[10] Home::Bugzilla::bugzilla.org ホームページ
http://www.bugzilla.org/

[11] IBM Rational ソフトウェア
http://www-06.ibm.com/jp/software/rational/

[12] IT スパイラル
http://it-spiral.ist.osaka-u.ac.jp/index.html

[13] JUDE:UML, ED, CRUC, DFD, Flowchart and Mind Map:Design & Modeling Tool
http://jude.change-vision.com/jude-web/index.html

[14] Main Page - Ximbiot - CVS Wiki
http://ximbiot.com/cvs/wiki/

[15] Microsoft Office Visio ホームページ
http://office.microsoft.com/ja-jp/visio/default.aspx

[16] Object Managemant Group
http://www.omg.org/

[17] ソフトウェアタグ規格 第 1.0 版
http://www.stage-project.jp/kanri/data/dl/20081029151602.pdf

[18] StagE プロジェクト
http://www.stage-project.jp/

[19] subversion.tigris.org
http://subversion.tigris.org/

[20] 「動かないコンピュータ」の解決に裁判は役立つか
http://itpro.nikkeibp.co.jp/free/ITPro/OPINION/20040609/145573/

[21] XML Metadeta Interchange
http://www.omg.org/technology/documents/formal/xmi.htm