電子情報工学専攻

Advanced Electronic and Information Engineering Course

平成27年度 専攻科特別研究論文

ソフトウェア品質の第三者評価のための作業履歴計測手法

Automatic Activity-record Method for Third-party Evaluation of Software Quality

指導教員名　上野 秀剛，内田 眞司

論文提出者名　池田 祥平

独立行政法人 国立高等専門学校機構

奈良工業高等専門学校 専攻科

National Institute of Technology, Nara College

Faculty of Advanced Engineering

ソフトウェア品質の第三者評価のための作業履歴計測手法

Automatic Activity-record Method for Third-party Evaluation of Software Quality

池田 祥平
Ikeda Shohei

独立行政法人 国立高等専門学校機構
奈良工業高等専門学校 専攻科 電子情報工学専攻
大和郡山市矢田町 22 番地（〒 639-1080）
National Institute of Technology, Nara College, Faculty of Advanced Engineering
22 Yata-cho, Yamatokoriyama, Nara 639-1080, Japan

Abstract— This paper proposes third-party evaluation of software development process using activity-record. The third-party evaluation can evaluate the process based on real activities and while being developed. This paper makes an automatic activity-record method for the third-party evaluation. The method can (1) Easily record many developer's activity-logs, (2) Prevent tampering of activity-logs, (3) Match activities and products.

Keywords— Third-party Evaluation, Software Quality Assurance, Automatic Activity-record, Time Tracking, Tampering Detection

関連業績リスト

1. 池田 祥平, 上野 秀剛, “ソフトウェア開発における作業履歴の非改ざん性保証と共有手法,” 第 19 回電子情報通信学会関西支部学生会, pp.85, February 2014.

2. 池田 祥平, 上野 秀剛, “第三者によるソフトウェア開発作業評価のための作業記録の保護手法,” 情報処理学会研究報告　ソフトウェア工学研究会, Vol.2014-SE-185, No.2, pp.1-8, July 2014.

目次

1. はじめに

ソフトウェアの開発組織は，発注者に高品質なソフトウェアを提供することが求められる．ソフトウェアの品質を担保する方法として，第三者機関による開発プロセスの評価（以後，第三者評価）がある．第三者評価は，中立な第三者である認証組織が専門的かつ客観的に評価対象の品質を評価・保証するものである [1, 2, 3]．ソフトウェア開発組織の開発プロセスが第三者によって保証されていれば，発注者はその保証を指標に発注先を選択できる．

Personal Software Process (PSP)[4] に代表されるようなプロセス改善手法では，改善提案の根拠として開発に関する作業履歴データの収集・分析が利用されている．作業履歴は，PSP のように，開発組織のプロセス管理能力や工数見積もりの正確性把握などプロセスの分析に利用されていることから，開発プロセスの品質を保証する根拠としても有用であると考えられる．

作業履歴の計測手法としては，第三者の人手による計測や，PSP 支援システム [5, 6] による自動計測が挙げられる．しかし，人手による計測は開発作業の妨げになりうる上に，作業量が多いため，計測漏れも起こりうる．加えて，既存の計測システムは，自己改善のための作業履歴を用意するもので，第三者評価を想定していないため，作業履歴の改ざんが容易である．第三者評価のためには，計測漏れがなく，改ざんがないことを証明できる作業履歴の計測を小さな負担でできる必要がある．

本研究の目的は，開発プロセスの第三者評価に利用可能な作業履歴を計測・収集する手法を構築することである．計測する作業履歴は開発組織によるプロセス改善にも利用できる．提案する計測手法によって，計測漏れがなく，改ざんのない作業履歴に基づいた開発プロセスの品質評価が可能になる．また開発組織自身によるプロセス改善にも利用することで，発注者により高品質なソフトウェアを提供できる．

2. 作業履歴に基づいた第三者評価

2.1 ソフトウェアの第三者評価

第三者評価とは，業務の実施者，および成果物の利用者以外の公正で中立な第三者が，専門的かつ客観的な立場から製品やサービスを評価するものである．第三者評価はソフトウェア開発の分野において，品質の向上や外部への品質証明を目的に実施される．例えば，NASA や JAXA では開発する宇宙機の品質を高めるために，ソフトウェア IV&V(Software Independent Verification and Validation) を実施している [2]．ソフトウェア IV&V は，ソフトウェアの開発組織から技術面・組織面・資金面で独立した認証組織が，開発の各プロセスで作成される要求仕様書や設計書，ソースコードなど成果物間の整合性に着目して，分析や設計などの各開発プロセスが正しく行われているか評価することで，ソフトウェアの品質を保証する．開発組織は独立した第三者から品質に対する評価を受けることで，内部で品質評価をする場合と比べてより厳密な評価が期待できる．

パッケージソフトウェア品質認証制度（PSQ 認証制度）は，一般社団法人コンピュータソフトウェア協会が独立した認証組織としてカタログなどの製品説明と実装されたソフトウェアを比較し，機能の有無やその内容を評価する [3]．開発組織は第三者に評価してもらうことで，より信頼を得やすい形で製品の品質を証明できる．

本稿では，従来とは異なるアプローチとして，作業履歴に基づく開発プロセスの第三者評価を提案する．

2.2 作業履歴に基づく第三者評価の利点

図 2.1 に本研究で想定する作業履歴の例を示す．1 行が 1 つの履歴で，作業に利用したアプリケーションと作業の実施日時，アプリケーションに対する操作の数といったような作業の量が含まれる．図 2.1 に示した作業履歴から開発者が設計書と UML 図の双方を同

```
アプリケーション名,                作業実施日時,    作業量
文書編集ソフト,      05/02 10:12:29 - 05/02 10:16:40,     192
UML 図編集ソフト,    05/02 10:16:40 - 05/02 10:17:23,       3
文書編集ソフト,      05/02 10:17:23 - 05/02 10:17:58,     368
UML 図編集ソフト,    05/02 10:17:58 - 05/02 10:20:30,      16
```

図 2.1　プロセス評価に必要な作業履歴

時に表示し，主に設計書を編集していることがわかる．

例えば，図 2.1 に示した作業履歴が設計工程で記録された場合，開発者が工程に従って設計作業をしていることが履歴から確認できる．一方で，同じ作業履歴が実装工程で記録された場合，設計上の手戻りやプロセスに反した作業手順を使用している可能性を示す．また，開発チーム内の複数開発者を対象に作業履歴を計測することで，特定の開発者に作業が偏っていないか，各作業者が役割に応じた作業をしているか理解することができる．作業履歴はプロジェクトのスケジュール情報やチーム構成情報とともに分析することで開発組織のプロセス管理能力やチーム管理能力，工数見積もりの正確性把握に有用となる．

従来の第三者評価と比較して作業履歴に基づいた第三者評価の利点は以下のとおりである．

(1)　作業内容に基づいたプロセス評価

実作業を記録した作業履歴を分析することで，手順書などに書かれた情報からではなく，実際に開発者がした作業内容に基づいたプロセス評価が可能になる．

例えば，スケジュールでは 1 ヶ月テストすることになっていたにもかかわらず，実際には 3 日しかテストに費やしていなかったという実態を把握できる．

作業履歴に基づくことで，厳密なプロセス評価が可能になる．

(2)　開発途中における評価

作業履歴は開発中でも計測できるため，開発の終了を待たずともそれまでのプロセスを第三者評価可能である．

開発組織は第三者評価の結果を受けて，ソフトウェアやプロセスの問題を把握する．従来の第三者評価では開発終了後にしかソフトウェアを評価できないため，開発組織は開発

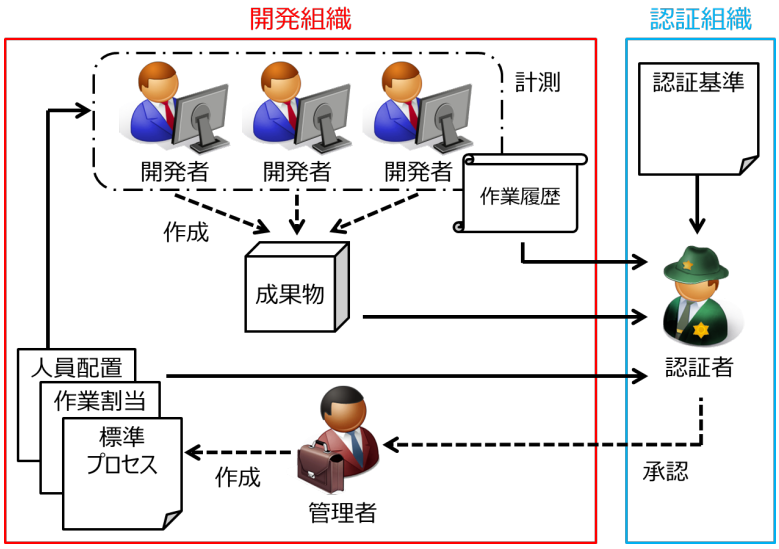


図 2.2　評価までの流れ

が終了しないと第三者評価で見つかった問題に対処できない．開発終了後の対処は負担が大きい．開発プロセスも同様で，開発終了後に評価を受けて改善するだけでは，次の開発以降にしか改善したプロセスを反映できない．品質の悪いプロセスのまま開発が進んでしまい，最終工程になってバグが見つかるといった事態になり対処に追われる恐れがある．

開発途中で評価結果を受け取ることができれば，開発組織は開発工程の早期におけるプロセス上の問題の把握が可能になり，以降のプロセスを改善することができる．これにより，開発プロセスの第三者評価は開発中のソフトウェアの品質向上を図れ，開発終了後の対処による負担を軽減できる．

2.3　履歴による作業状況の把握

作業履歴に基づく開発プロセスの第三者評価を実施した際の，評価するまでのデータの流れを図 2.2 に示す．

関係する組織としてソフトウェアを開発する開発組織と，開発プロセスを認証する認証組織がある．開発組織には複数の開発者と，開発者がプロセス通りに働いているか管理する管理者がいる．認証組織には開発プロセスを客観的かつ専門的な立場から評価して，品質を満たしていることを保証する認証者がいる．ソフトウェアの開発プロジェクトが立ち上がるたび，管理者は人員割当や作業割当，スケジューリングを行い，それらに従って作

業するよう開発者を管理する．開発者は定められたプロセスに従って開発する．このとき，開発者がプロジェクトのためにした作業履歴を計測・記録する．

管理者は記録された作業履歴を確認して，開発プロセスに問題がないか分析する．問題を発見した場合，原因を特定して対策を講じる．例えば，コーディングが予定通りに進んでいないという問題が発見されたとする．その場合，作業履歴を分析し，開発担当者が報告書作成に時間を取られすぎているということが原因ならば，報告書の項目数を見直し時間がかかり過ぎないようにするといった対策を講じる．もしくはネットワークに不調が発生し，それが原因となって作業が中断されているならば，ルータを買い直すといった対策を講じる．

認証者は計測された作業履歴に加え，人員配置や作業割当，開発スケジュールといったプロジェクトの概要と，成果物を受け取る．これらに基づいて，認証者は開発作業が適切に実施されているか評価する．プロジェクトの概要と成果物は，ある作業が予定通りに実施できたか，ある作業が成果に結びついているかなどを判断するために有用である．例えば開発スケジュールがあれば，設計書を編集する作業が設計工程に予定通り実施されたのか，手戻りなどが起こり実装工程に実施されたのか判断することができる．評価の結果，評価基準を満たしていれば認証し，開発組織が適切な開発プロセスに基づいて作業していることを外部に保証する．認証基準の例としては，上流工程において，テストを十分手順に沿って行っており，バグを上流で減らそうとしているかといったことが考えられる．他にも，プロジェクトが終わると，開発プロセスを見直して，ソフトウェアの品質向上を目指して，自己改善を忘れていないかということもある．本稿では具体的な評価基準は策定しないが，開発者の作業履歴に基づいた分析手法はこれまでに多数研究されており [4, 7, 8, 9, 10, 11, 12]，これらに基づくことで評価基準は定められると考えられる．分析手法の研究については，5 章にて詳しく述べる．なお，本研究で提案する第三者評価は，プロジェクト開始時に作成される作業割当や開発スケジュールなどの情報を基に評価するため，作業割当や開発スケジュールなどが適宜変化するアジャイル型のような開発プロセスよりも，これらが大きく変化することのないウォーターフォール型の開発プロセスの評価に適している．

作業履歴に基づく開発プロセスの第三者評価を実施するには，評価する前に作業履歴を計測・収集する必要がある．本研究では計測・収集の工程に着目し，開発プロセスの第三者評価に利用可能な作業履歴を計測・収集する手法を構築する．

3. 作業履歴計測手法

本章では作業履歴に基づいた第三者評価を実現するために必要となる，作業履歴計測手法が満たすべき要件と要件を満たす機能について述べる．要件は大きく分けて，すべての作業履歴を計測すること，改ざんされないこと，作業と成果の対応が取れることの 3 つである．

3.1 作業履歴の計測

第三者評価では，評価に漏れがないよう，全開発者の作業を漏れなく計測しなければならない．このとき，開発者の作業を妨げずに全開発者を評価するため，容易に多人数を計測できることも重要である．また開発途中でも評価できるように，認証機関はリアルタイムに作業履歴を収集する．収集の際，作業履歴にはプロジェクト名や開発のノウハウといった機密情報が含まれるため，認証機関以外の外部組織に漏洩させない．

計測手法は開発者の端末で動作するクライアントシステムと，認証機関の端末で動作するサーバシステムで構成される．クライアントは開発者が端末上で行った作業を自動で計測する．作業として計測するのは，アクティブになっているアプリケーションの名称と，利用時間，加えて作業量としてアプリケーションに対する打鍵数とクリック数である．計測した作業は，作業履歴として開発者の端末上に記録すると同時に，暗号化しサーバに送信する．サーバは作業履歴を受信し．復号化してから，認証機関の端末上に記録する．

作業履歴をクライアントによって自動的に計測することで開発者に負担をかけず，また，入力忘れによる抜けや記入ミスによる誤りを含まない，正確な履歴を取得することが可能である．また，複数の端末にクライアントを導入することで，多人数の開発者に対しても容易に計測が可能となる．記録時に端末上に出力された作業履歴は，管理者が開発プロセスの改善のために利用できる．記録と同時にサーバに送信するため，認証機関はリアルタイムに作業履歴を把握できる．送信データは暗号化しており，外部から傍受されても

作業履歴の内容は漏洩しない．

計測手法では，資料作成やコーディングといった端末上で実施される作業のみを計測する．開発者のする作業は，会議のような端末上以外で実施される作業も含まれる．第三者評価するには，こういった作業も知る必要がある．しかし，会議用の資料作成や会議後の実施報告書作成は端末上で実施されるため，会議の内容を把握することは十分可能である．よって端末上のみの作業履歴であっても，開発プロセスを第三者評価できると考えられる．また端末上に絞ることにより，計測手法の導入，運用が容易になるという利点もある．

3.2　作業履歴の改ざん防止

第三者評価の信頼性が損なわれないよう，作業履歴に対する改ざんを防がなくてはならない．改ざん方法として以下の 2 点がある．

- 出力される作業履歴の内容を書き換える．
- クライアントを事実と異なる作業履歴を出力する動作に書き換える．

各改ざんに対する対策について述べる．

3.2.1　作業履歴の内容改ざん防止

3.1 節で述べたように，クライアントは作業履歴を計測後，即時に暗号化して送信する．

計測後，即送信するため，端末上に出力される作業履歴を書き換えても，作業履歴の内容を書き換えることはできない．暗号化して送信するため，送信データの傍受しても，同様に書き換えられない．

3.2.2　クライアントの改ざん検出

クライアントに対する改ざんについては，サーバにその改ざんを検出可能とすることで，防止を図る．

クライアントは認証機関から開発企業へと提供され，開発企業の各開発者の端末上に導入される．クライアントはサーバからの認証を受けなければ作業履歴を計測できない．認証機関に導入されるサーバは，提供される前の改ざんのないクライアントの実行ファイル

からハッシュ値を計算し，記憶する．提供されたクライアントは，起動して計測を開始する前に，サーバから認証を受けるため，クライアントの実行ファイルからハッシュ値を計算し，サーバへ送信する．このとき送信するハッシュ値は，ワンタイムパスワードで暗号化する．サーバは認証を要求してきたクライアントからハッシュ値を受け取り，あらかじめ記録しておいた改ざんのないクライアントの実行ファイルと比較する．

サーバによるクライアント認証は次の手順で行われる．

1) クライアントはサーバと通信し，接続要求 req をサーバに送信する．
2) サーバはワンタイムパスワード $pass$ をランダムに生成し，クライアントに送信する．
3) クライアントは自身の実行ファイルのハッシュ値 $hash'_{client}$ を計算し，受信した $pass$ で暗号化した $encrypt(hash'_{client})$ をサーバに送り返す．
4) サーバは $encrypt(hash'_{client})$ を復号化し，得られた $hash'_{client}$ とあらかじめ記録していた $hash_{client}$ と比較する．
5) $hash'_{client}$ と $hash_{client}$ が一致していればクライアントに起動許可 ack を，一致していなければ起動不許可 $nack$ を送信する．
6) クライアントは ack を受け取れば作業履歴の計測を開始し，$nack$ を受け取れば実行を終了する．

ハッシュ値は計算元になるデータが変更されると異なる値になるため，サーバはあらかじめ記録しておいたクライアントのハッシュ値 $hash_{client}$ と接続を要求してきたクライアントのハッシュ値 $hash'_{client}$ を比較することで，クライアントの改ざんを検出できる．ただしクライアントのハッシュ値 $hash'_{client}$ は，改ざんがない限り常に同じ値となるため，送信データの盗聴によって，改ざんしたクライアントによるなりすましが容易に行えてしまう．そこで，生成のたびに値が変わるワンタイムパスワード $pass$ で暗号化することで，$hash'_{client}$ が送信される際の送信データが一様になることを防いでいる．

3.3 作業履歴に対応した成果物の把握

ある作業が適切に実施されたかを評価するためには，作業の成果も評価できる必要がある．

クライアントは，利用ソフトウェアなどの作業に関する情報に加えて，作業によって編集された成果物のサイズや名前の移り変わりを計測する．

成果物の変遷を記録することで，作業履歴と成果を関連付けて分析できる．

3.4 実装

3.1 節から 3.3 節までの機能を持つ手法を実装した．C#言語で実装したサーバ・クライアント型のシステムである．サーバは，634 行で，2 個のクラスを持つ．クライアントは，既存の作業計測システムである TaskPit[6] に新たに一つクラスを追加することで実装しており，追加したクラスは 444 行，クライアント全体は 4985 行で 31 個のクラスを持つ．

サーバはインストーラ形式で認証機関の端末上に導入する．サーバは，クライアントをクライアントに設定されたクライアント名で識別し，クライアントごとに作業履歴を記録する．クライアントは同時に 63 台まで接続可能である．サーバは，クライアントが認証を接続要求してきた結果や，各クライアントから最後に受信した時間を表示する．認証機関は改ざんのあるクライアントが接続してきていないか，どのクライアントが動作しているか確認できる．

クライアントもインストーラ形式で開発者の端末上に導入し，バックグラウンドで動作する．導入時にスタートアップに登録され，以後端末の起動と同時に自動で計測を開始する．クライアントは導入された端末上において，アクティブになっているアプリケーションの名称とウィンドウタイトル，利用時間，作業量としてアプリケーションに対する打鍵数とクリック数を作業として記録する．作業履歴の一例を図 3.1 に示す．図 3.1 に示す作業履歴から，開発者は設計書を確認しながら，プログラミングしていることがわかる．クライアントが送信する作業履歴の暗号化には，RSA 暗号の公開鍵暗号方式を用いている．クライアントのハッシュ値を計算するために利用するハッシュ関数は MD5 である．

クライアントは作業と成果物を対応付けるために，成果物であるファイルへのアクセス履歴を記録する．アクセス履歴には，アクセス時間，ファイルへの操作の種類，アクセスした後のファイル名とアクセスする前のファイル名，アクセス後のファイルサイズが含まれる．ファイルへの操作の種類は，新規ファイル作成時の「作成」，既存ファイル編集時の「変更」，ファイル名変更時の「ファイル名変更」の 3 つがある．ファイル名が変更された場合は，アクセスする前のファイル名を記録し，ファイルサイズは記録しない．アクセスした時間を参考に作業履歴を確認することで，各ファイルアクセスがどの作業によって実施されたものかわかる．実装したアクセス履歴の一例を図 3.2 に示す．

行,	タスク名,	開始日時,	終了日時,	左 click,	右 click,	打鍵,	exe 名
12,	プログラミング,	05/13 16:08:46,	05/13 16:13:49,	2,	0,	306,	eclipse.exe: Java
13,	文書編集・閲覧,	05/13 16:13:49,	05/13 16:13:52,	2,	0,	0,	WINWORD.EXE: 設計書.docx
14,	プログラミング,	05/13 16:13:52,	05/13 16:16:19,	7,	1,	92,	eclipse.exe: Java

図 3.1　実装した作業履歴の例

行,	アクセス時間,	種類,	ファイル,	元ファイル,	サイズ
1,	5/13 20:14:55,	作成,	C:\projectA\ 設計書.docx,	,	0
2,	5/13 20:15:06,	変更,	C:\projectA\ 設計書.docx,	,	223856
3,	5/13 20:15:09,	ファイル名変更,	C:\projectA\ 設計書_v2.0.docx,	C:\projectA\ 設計書.docx,	-

図 3.2　実装したアクセス履歴の例

4. 評価

4.1 作業履歴の計測

4.1.1 計測漏れなし

実装システムが，全作業を漏れなく計測できるか，実験して確かめた．

実験環境

実験では，クライアントを Windows 7 の Core i7 というスペックを持つラップトップ型 PC（Client）に導入し，サーバを Windows 7 の Core i7 というスペックを持つデスクトップ型 PC（Server）に導入した．デスクトップ型は有線 LAN で，ラップトップ型は無線 LAN でローカルネットワークに接続した．

実験方法

クライアントを導入した端末上で，ランダムに用意した 100 作業を実施し，クライアントとサーバが実施した作業をそれぞれ漏れなく計測・収集できるか確認する．用意した作業は，アクティブにするアプリケーション，そのアプリケーション上での打鍵数，クリック数を指定している．アプリケーションは，Word，Explorer，Excel の中からランダムで選ばれ，同じアプリケーションを連続でアクティブにすることはない．打鍵数，クリック数はそれぞれ 0〜9 の間でランダムに選択される．作業の一部を図 4.1 に示す．この場合，最初に Word をアクティブにして，9 回打鍵する．次に Explorer をアクティブにして，5 回左クリック，8 回右クリックする．

実験の手順としては以下の通りである．

[手順 1] Server 上で計測システムのサーバを起動する．

[手順 2] Client を起動する．

順番,	アクティブにするアプリケーション,	打鍵数,	左クリック数,	右クリック数
1,	Word,	9,	0,	0
2,	Explorer,	0,	5,	8
3,	Word,	5,	4,	3
4,	Excel,	6,	1,	1

図 4.1　作業の一部

[手順 3]　用意した 100 作業を実施する．

[手順 4]　Client と Server に記録された作業履歴から，計測漏れがないか確認する．

実験結果

Client に記録された作業履歴から，100 作業中 100 作業が漏れなく記録されていることを確かめた．Server に記録された作業履歴から，100 作業中 100 作業が漏れなく記録されていることを確かめた．

クライアントは全作業を漏れなく計測し，サーバも漏れなく収集できたため，実装システムは，すべての作業を漏れなく計測・収集できると言える．

4.1.2　多人数を容易にリアルタイムでセキュアに計測

実装システムが，多人数を容易に計測でき，リアルタイムかつ認証機関外部への漏洩なしで収集可能かを評価する．多人数を容易に計測可能かは導入と運用の面から評価する．

評価では，(a) 導入容易性，(b) 運用容易性，(c) リアルタイム収集，(d) セキュアな収集の 4 つの指標を用いて，実装システムを従来の計測システムと比較する．従来の計測システムである Process Dashboard[*1]，Task Coach[*2]，Slim Timer[*3]，TaskPit[6] と実装システムの比較結果を表 4.1 に示す．

Process Dashboard，Task Coach，TaskPit，実装システムは，インストーラ形式で端末に導入するため，負担は大きくない．SlimTimer Time は，Web アプリケーションで

[*1] www.processdash.com
[*2] members.chello.nl/f.niessink
[*3] www.slimtimer.com

表 4.1　従来システムとの比較

	(a) 導入容易性	(b) 運用容易性	(c) リアルタイム収集	(d) セキュアな収集
Process Dashboard	o	x	x	x
Task Coach	o	x	x	x
Slim Timer	o	x	o	o
TaskPit	o	x	x	x
実装システム	o	o	o	o

インストール不要なため，同様に導入の負担は大きくない．

Process Dashboard，Task Coach，SlimTimer Time，TaskPit は，手動で操作しないと作業計測を開始しない．開発者の作業を妨げることから，これらの計測システムは運用の負担が大きい．実装システムは，導入後スタートアップに登録され，以後端末起動時に自動で作業計測を開始するため，手動での操作が必要ない．開発者の作業を妨げないため，実装システムは運用の負担が小さい．

Process Dashboard，Task Coach，TaskPit は，計測した作業履歴を端末上のみに出力するため，認証機関が作業履歴を収集するには一手間かかる．SlimTimer Time には共同作業者と報告相手に計測した作業履歴を共有する機能があり，実装システムは計測した作業履歴を計測時にサーバに送信するため，リアルタイムに収集可能である．

Process Dashboard，Task Coach，TaskPit が端末上に出力した作業履歴は，暗号化されているわけではなく，外部にもれないことが保証されてるわけではない．SlimTimer Time はアカウントで管理されており，パスワードが知られなければ作業履歴が漏洩することはない．実装システムは，作業履歴を公開鍵で暗号化しているため，外部に漏洩することはない．

比較の結果，実装システムは作業履歴の多人数計測が導入と運用の二面で容易であり，リアルタイムでセキュアな収集が可能である．よって実装システムは，開発プロセスの第三者評価に利用する計測システムとして適していると言える．

4.2　作業履歴の内容改ざん防止

4.2.1　計測システムの改ざん防止

実装システムが，出力する作業履歴に対する改ざんを防止できるか評価する．

クライアントは計測後，即時に作業履歴をサーバに送信するため，開発者の端末上に出力される作業履歴を書き換えての改ざんは防止できている．また送信データは，RSA 暗号の公開鍵暗号方式で暗号化している．RSA 暗号の公開鍵で暗号化されたデータを秘密鍵無しで復号化することは現実的な時間では不可能であるため，送信データを傍受しての改ざんは防止できている．

よって実装システムは，計測した作業履歴に対する改ざんを防止できると言える．

4.2.2 クライアントの改ざん検出

実装システムが，クライアントに対する改ざんを検出できるか，実験して確かめた．

実験環境

実験では計測漏れなしの実験時と同様，クライアントを Windows 7 の Core i7 というスペックを持つラップトップ型 PC（Client）に導入し，サーバを Windows 7 の Core i7 というスペックを持つデスクトップ型 PC（Server）に導入した．デスクトップ型は有線 LAN で，ラップトップ型は無線 LAN でローカルネットワークに接続した．

実験方法

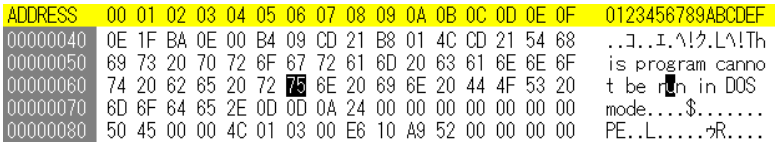
サーバがクライアントの改ざんを検出可能か確かめるために，クライアントを改ざんしたときにサーバから認証されずに作業の計測ができないことを確認する．クライアントに対する改ざん方法としてソースコードの改ざんは想定せず，バイナリの書き換えによる改ざんを想定する．図 4.2 にバイナリエディタによるクライアント実行ファイルの改ざんの様子を示す．改ざん箇所には，クライアントの動作に影響しない表示文字列の一部を選択した．実験の手順を以下に示す．

[手順 1] Client に改ざんされていないクライアントを，Server に改ざんされていないサーバをインストールする．

[手順 2] サーバを起動し，クライアントの受付待機状態にする．

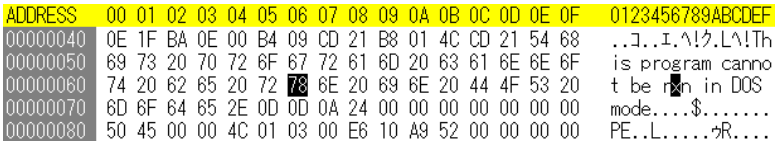
[手順 3] 改ざんされていないクライアントを起動し，正常に作業履歴の計測が始まることを確認する．

[手順 4] Client のクライアントをアンインストールし，改ざんされたクライアントをインストールする．



ADDRESS	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..ｺ..ｴ.ﾍ!ｸ.Lﾍ!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....$.......
00000080	50 45 00 00 4C 01 03 00 E6 10 A9 52 00 00 00 00	PE..L.....ｦR....

(a) 改ざん前



ADDRESS	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..ｺ..ｴ.ﾍ!ｸ.Lﾍ!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 78 6E 20 69 6E 20 44 4F 53 20	t be rxn in DOS
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....$.......
00000080	50 45 00 00 4C 01 03 00 E6 10 A9 52 00 00 00 00	PE..L.....ｦR....

(b) 改ざん後

図 4.2　クライアントのバイナリ書き換え

[手順 5]　改ざんされたクライアントを起動し，サーバに認証されずクライアントが終了することを確認する．

実験結果

図 4.3 に改ざんされていないクライアントを起動したときのサーバの処理結果を示す．改ざんされていないクライアントに対して，サーバが認証を成功し，クライアントの起動を許可していることがわかる．

図 4.4 に改ざんされたクライアントを起動したときのサーバの処理結果を示す．改ざんされたクライアントに対してサーバが認証できず，クライアントの起動を拒否していることがわかる．またクライアントは処理を終了し，作業履歴が計測されていないことが確認できた．

検証の結果，改ざんのないクライアントは作業履歴を計測でき，改ざんされたクライアントは作業履歴を計測できないことが確認できた．しがたって実装システムは，クライアントに対する改ざんが検出できると言える．

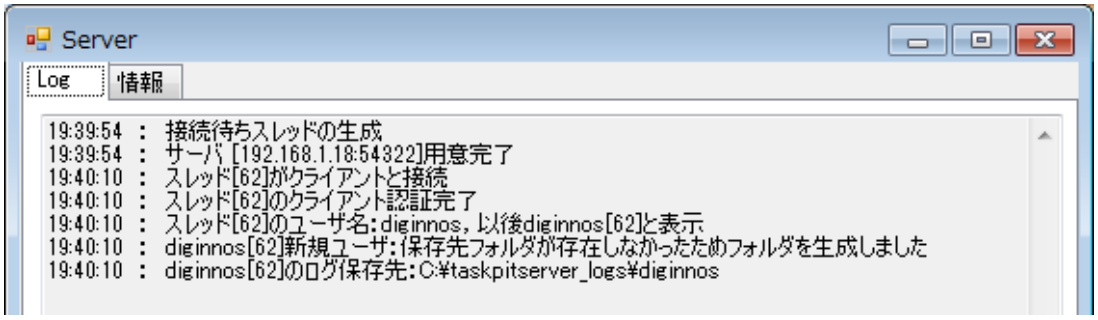


図 4.3　改ざんのないクライアントへの起動許可

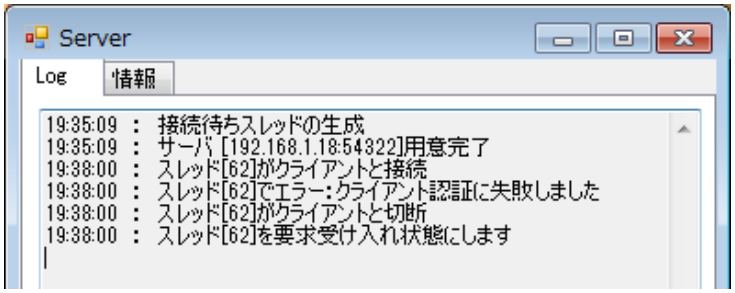


図 4.4　改ざんされたクライアントへの起動拒否

4.2.3　他の改ざんに関する考察

改ざん方法は多様にあり，計測手法はそのすべてに対応しているわけではない．計測手法では，容易に行える改ざんを防止することで，第三者評価への信頼性を高めた．対策していない改ざんへは必要に応じて適宜追加や検証が必要であるものの，他の改ざんは困難であるため，運用上の防止は十分可能であると考えられる．例えば，クライアントが出力する前に，クライアントのメモリ上のデータを書き換える改ざんも考えられるが，メモリ上のデータの書き換えはクライアントの動作を理解していなければ難しいため，運用上の防止は十分できていると言える．

また 4.2.2 節の実験では，バイナリエディタを用いて実行ファイル中の文字列を書き換えたが，作業履歴の出力機能に対する書き換えについて検証していない．だが，実行ファイルのハッシュ値は計算元の内容が少しでも変化すれば値が変わるため，履歴出力機能に対する書き換えや他の方法による改ざんも検出が可能と考えられる．一方で，クライアン

ト認証機能を書き換えて改ざんのないクライアントのハッシュ値をサーバに送信することで，誤認証してしまう恐れがある．しかし，この改ざんにはクライアント認証にクライアントのハッシュ値を用いていることを知る必要がある．一般に実行ファイルからシステムの動作を解析することは容易ではなく，そこからクライアント認証に用いているハッシュ値を知ることは難しい．また実行ファイルのハッシュ値は送信時に同一のデータとならないよう暗号化しているため，通信内容から知られることも防止できる．よって，クライアント認証機能を書き換える改ざんには十分対応できていると言える．

4.3 作業履歴に対応した成果物の把握

実装システムが，作業と成果物の対応を記録できるか，実験して確かめた．

実験環境

実験では，クライアントを Windows 7 の Core i7 というスペックを持つラップトップ型 PC（Client）に導入した．サーバは利用しないため，導入しない．

実験方法

Word と Explorer を用いてファイルの編集作業を実施したときにクライアントが記録したアクセス履歴から，作業と成果物の対応が取れるか確認する．実験の手順を以下に示す．

[手順 1] Client を起動する．

[手順 2] Word を起動し，「test.docx」という名前のファイルを新規に作成する．

[手順 3] Word を起動しなおし，手順 1 で作成した「test.docx」を開いて，文字列を入力して上書き保存する．

[手順 4] Word を終了し，その後 Explorer を起動して手順 1 で作成した「test.docx」の名前を「change.docx」に変更する．

実験結果

実験時に記録したアクセス履歴を図 4.5 に，同時に記録した作業履歴を図 4.6 に示す．ただし，Word はファイル編集時にバックアップファイルを自動で保存するが，図 4.5 にはそのバックアップファイルの変遷は示していない．

```
2/16 13:57:16,  作成,                C:\projectA\test.docx,                ,       0
2/16 13:57:40,  変更,                C:\projectA\test.docx,                ,   12908
2/16 13:58:02,  ファイル名変更,  C:\projectA\change.docx,  C:\projectA\test.docx,       -
```

図 4.5　アクセス履歴（成果物対応付け実験）

```
タスク名 ,                開始日時,          終了日時 ,  左 click,  右 click,  打鍵 ,  exe 名
テキスト閲覧・編集,  2/16 13:57:01,  2/16 13:57:08,          2,          0,       0,  WINWORD.EXE: 文書 1
テキスト閲覧・編集,  2/16 13:57:08,  2/16 13:57:16,          2,          0,       5,  WINWORD.EXE: 名前を付けて保存
テキスト閲覧・編集,  2/16 13:57:16,  2/16 13:57:20,          1,          0,       0,  WINWORD.EXE: test.docx
デスクトップ ,            2/16 13:57:20,  2/16 13:57:22,          1,          0,       0,  explorer.exe: desktop
テキスト閲覧・編集,  2/16 13:57:22,  2/16 13:57:30,          3,          0,       0,  WINWORD.EXE: 文書 2
テキスト閲覧・編集,  2/16 13:57:30,  2/16 13:57:34,          2,          0,       0,  WINWORD.EXE: ファイルを開く
テキスト閲覧・編集,  2/16 13:57:34,  2/16 13:57:44,          1,          0,      35,  WINWORD.EXE: test.docx
ファイル操作 ,            2/16 13:57:44,  2/16 13:58:08,          3,          0,       7,  explorer.exe: projectA
```

図 4.6　作業履歴（成果物対応付け実験）

図 4.5 から，「test.docx」が新たに作成され（手順 2），その後「test.docx」のサイズが増加し（手順 3），最後に「test.docx」が「change.docx」にリネームされたこと（手順 4）がわかる．

また，アクセス時間を参考に作業履歴を確認することで，各アクセスがそれぞれ Word や Explorer によって実施されたことがわかる．例えば，「test.docx」が作成された時間の「2/16 13:57:16 」には，「WINWORD.EXE」が利用されている．

ある作業がどの成果物を編集したかがわかるため，実装システムは作業と成果物の対応を記録できていると言える．

5. 関連研究

Watts によって提唱された PSP/TSP は，開発者や開発チームが自身の作業に関するデータを収集・分析することで，開発プロセスを改善する手法である [4]．PSP/TSP では，各作業の見積もり時間と実測値の差異を求め，その原因を分析することでプロセスの問題点を明らかにする．PSP/TSP が提唱された当初は，作業履歴の計測は紙を用いた手作業だったため，PSP/TSP を支援するために複数の研究 [5] [13] や実装[*1][*2][*3]が提案されている．TaskPit は開発者の PC 上での作業を自動的に計測することで，作業履歴の分析に基づいた開発プロセスの分析と改善を支援する [6]．PSP/TSP 支援システムは開発者の作業履歴計測を支援する点で本研究と関連が深い．本研究で提案する計測手法は，異なる開発組織にいる複数の開発者を対象とした計測が容易であり，また，第三者が作業内容を評価できるように改ざんを検出する仕組みを有している点で新規性がある．

統合開発環境上における開発者の行動を計測・分析することで，ユーザの意図推定やプログラムに対する変更の理解を支援する手法が提案されている [7, 8]．Gu は Eclipse 上で動作するフレームワーク IDE++ を開発した [7]．IDE++ は開発者が Eclipse 上で行う操作を監視し，フレームワーク上のプラグインが操作履歴に基づいて作業効率の向上に寄与するホットキーを推薦する．

これらの研究は開発者の操作履歴を詳細に計測・分析している点で本研究と関連しているが，計測の対象が統合開発環境に限定されており，本研究と適用対象が異なる．本研究の提案手法は開発者が PC 上で行う全ての作業を対象にアプリケーション名や打鍵数などを計測することで，仕様書作成や設計といった，主に IDE 以外のツールを利用するプロセスの分析も可能である．

ソフトウェア開発における開発履歴に着目した研究として，リポジトリマイニングが

[*1] www.processdash.com
[*2] members.chello.nl/f.niessink
[*3] www.slimtimer.com

ある．リポジトリに蓄積された開発履歴に対してデータマイニングの手法を適用するリポジトリマイニングはソフトウェア工学の幅広いテーマで行われている [9, 10]．例えば畑らはコードクローンに関わる開発履歴のプロセスや人的属性に関する情報についてのリポジトリマイニングが可能なコードクローン版管理システムを提案している [10]．リポジトリマイニングに関する研究はオープンソースソフトウェア (OSS) のプロジェクトが利用する Subversion や Git などの版管理システムに記録されたソースコードの変更履歴や，Redmine のようなチケット管理システムに蓄積されたタスクや不具合の履歴を対象に，ソフトウェア進化やプロセスに関する多数の知見を報告している．

本稿が提案する作業履歴の計測手法は，これまでにリポジトリマイニングの対象とはされてこなかった，開発者のソフトウェア使用履歴を計測・蓄積することで開発プロセスの分析・評価に利用可能な新たなリポジトリの作成に寄与すると考えられる．ソフトウェアの使用履歴を対象とした分析はヒューマンコンピュータインタラクションの分野で広く行われており [11, 12]，ユーザの意図推定や作業支援に利用されている．ソフトウェア開発者を対象とした分析においても，作業履歴に基づいた開発プロセスの遷移やプロセスごとの作業時間の把握に有用であると考えられる．

6. おわりに

本稿では，作業履歴に基づく開発プロセスの第三者評価を提案した．また開発プロセスの第三者評価に利用可能な作業履歴を計測・収集する手法を構築した．計測手法は，開発者の端末上で動作するクライアントによって，計測を自動化することで，全開発者の作業履歴を漏れなく容易に計測し，暗号化して認証機関の端末上で動作するサーバに送信することで，リアルタイムかつセキュアに収集可能である．計測前にクライアントのハッシュ値を，作業履歴は計測後，即時に暗号化して，サーバに送信することで，作業履歴に対する改ざんを防止できる．また作業履歴に対応付けて成果物のファイルサイズの増減を計測することで，作業と成果を結びつけて分析できる．計測手法をサーバ・クライアント型システムとして実装し，全開発者の作業履歴を成果物と対応付けて漏れなく容易に計測し，リアルタイムかつセキュアに収集できることを確かめた．提案手法で計測した作業履歴は第三者による開発プロセスの品質保証と，開発組織自身による改善の双方に使用可能であり，開発組織がより高品質なソフトウェアを第三者による保証の元で提供することが可能となる．

今後の展望として，作業履歴に基づく開発プロセスの第三者評価を実現するために，評価基準を決めることが挙げられる．また評価のために計測する必要がある作業履歴の項目を定める．

謝辞

本研究を進めるにあたり，多くの方々のご助力をいただきました．この場を借りて，お礼を申し上げます．指導教員である上野秀剛講師には，研究にあたって助言をいただくばかりでなく，進路やインターンなどの面でも助けて下さいました．ありがとうございました．また本論文の作成にあたっては，内田眞司准教授，大谷真弘准教授のお二方に査読していただきました．ありがとうございました．同じ上野研究室の皆様，学友の皆様にも，発表練習やセミナーを通して，的確なアドバイスや鋭い指摘をいただきました．ありがとうございました．

本研究は，独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（SEC: Software Reliability Enhancement Center）が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けたものです．

参考文献

[1] 情報処理推進機構, “「ソフトウェア品質説明力強化の普及・推進のための調査」報告書”, 2013.

[2] National Aeronautics and Space Administration, “NASA Independent Verification and Validation Program, ” 2009.

[3] 一般社団法人コンピュータソフトウェア協会, “パッケージソフトウェア品質認証制度 申請者ガイドブック, ” 2013.

[4] W.S. Humphrey, “パーソナルソフトウェアプロセス入門, ”共立出版, 2001.

[5] 門田暁人, 亀井靖高, 上野秀剛, 松本健一, “プロセス改善のためのソフトウェア開発タスク計測システム, ”ソフトウェア工学の基礎ワークショップ (FOSE2008), pp.123-128, 2008.

[6] 門田 暁人, 上野 秀剛, 荒木 健史, 山田 欣吾, 松本 健一, “ソフトウェア開発企業における開発タスクの自動計測, ” ソフトウェア工学の基礎 XIV, ソフトウェア工学の基礎ワークショップ (FOSE2013), pp.257-262, 2013.

[7] Zhongxian Gu., “Capturing and exploiting fine-grained IDE interactions,” In Proceedings of the 34th International Conference on Software Engineering (ICSE ’12), pp.1630-1631, 2012.

[8] 大森 隆行, 丸山 勝久, “開発者による編集操作に基づくソースコード変更抽出”, 情報処理学会論文誌, Vol.49, No.7, pp. 2349-2359, 2008.

[9] Hadi Hemmati, Sarah Nadi, Olga Baysal, Oleksii Kononenko, Wei Wang, Reid Holmes, and Michael W. Godfrey. “The MSR cookbook: mining a decade of research,” In Proceedings of the 10th Working Conference on Mining Software Repositories (MSR ’13), pp.343-352, 2013.

[10] 畑 秀明, 肥後 芳樹, 楠本 真二, “リポジトリマイニング可能なコードクローン版管理

システムの提案”, 情報処理学会論文誌, Vol.54, No.2, pp.894-902, 2013.

[11] Michael Granitzer, Andreas. S. Rath, Mark Kroll, Christin Seifert, Doris Ipsmiller, Didier Devaurs, Nicolas Weber, and Stefanie Lindstaedt, “Machine Learning based Work Task Classification,” Journal of Digital Information Management, pp.306-313, 2009.

[12] Michael Bernstein, Jeff Shrager, and Terry Winograd,“Taskpose: Exploring Fluid Boundaries in an Associative Window Visualization,” In Proc. the 21st Annual ACM Symposium on User Interface Software and Technology (UIST), pp.231-234, 2008.

[13] Koji Torii, Ken-ichi Matsumoto, Kumiyo Nakakoji, Yoshihiro Takada, Shingo Takada, Kazuyuki Shima, “Ginger2: An Environment for Computer-Aided Empirical Software Engineering, ” Transactions on Software Engineering, Vol.25, No.4, pp.474-492, 1999.