Vol.2014-SE-185 No.2
2014/7/9

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

池田 祥平[1]　上野 秀剛[1]

概要：ソフトウェア IV&V に代表されるソフトウェア開発プロセスの第三者組織による評価は，ソフトウェアの品質を保証するための重要な方法である．PSP 支援ソフトのような作業履歴を記録するシステムを用いることで，客観的な証拠に基づいた品質保証が可能になる．一方で，既存のシステムは第三者による評価に用いることを想定していないため，作業履歴が開発組織に改ざんされ，正当な評価ができない可能性がある．本研究では第三者組織による評価を可能にするために，作業履歴に対する改ざんを防ぐ手法を提案する．提案手法では作業履歴と作業計測システムのハッシュ値を随時送信することで，開発組織による作業履歴および作業計測システムの改ざんを検出する．

1. はじめに

様々な製品やサービスの高機能化・多機能化においてソフトウェアの果たす役割が増大するとともに，要求通りの機能があるか，誤動作がないかといったソフトウェア品質を確保することが重要な課題となっている [1], [2]．ソフトウェア品質を確保するための取り組みとしてソフトウェア IV&V (Independent Verification and Validation) がある [3]．ソフトウェア IV&V では，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，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検証し，開発プロセス全体を通して正しい製品が作られているか評価する．ソフトウェアの開発組織と利用者の双方から独立した中立的な立場の第三者である認証組織が評価することで，ソフトウェア品質が確保されていることを客観的に保証する．ソフトウェア IV&V は，信頼性が高い安全なシステムが求められる宇宙機開発やインフラ系システムの開発において実施されている [3], [4], [5]．今後，家電製品やパッケージソフトウェアなど他分野においても，高機能化・多機能化したソフトウェアの品質を客観的に保証するためにソフトウェア IV&V の重要性が高まると考えられる．

開発組織における開発プロセスの評価には，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に着目する方法と，各プロセス中に実施される作業を記録した作業履歴に着目する方法がある．作業履歴に基づいた作業の正当性評価は医療分野においても行われており，作業の正しさを証明するためには作業履歴を実施した順序通りに漏れなく記録するとともに，記録された作業履歴が改ざんされていないことが求められる [6]．ソフトウェア IV&V における作業履歴の記録は，PSP (Personal Software Process) 支援システムのような既存のシステムが利用可能である．一方で，これらの既存システムは記録した作業履歴を第三者評価に利用することを想定していないため，開発者や開発組織が自身の評価を高めるために作業履歴を改ざんしても，認証組織が検出できない．また，開発者や開発組織によって改ざんが行われなかった場合においても，改ざんがないことを認証組織やそれ以外の外部組織に示すことができない．

本稿では，認証組織がソフトウェア品質を評価するために必要な作業履歴が，開発者や開発組織によって改ざんされていないことを客観的に示す手法を提案する．提案手法は，既存の作業計測システムが記録した作業履歴のハッシュ値を計算して認証組織に随時送信することで，作業履歴が変更されたことを検出する．これにより，開発者や開発組織による作業履歴の改ざんを防ぐと共に，認証組織やそれ以外の外部組織に対して改ざんが無いことを客観的に示すことができる．認証組織は，改ざんされていない作業履歴に基づいて開発プロセスを評価することでソフトウェア品質が確保されていることを客観的に保証できる．本稿では，提案手法に基づいて既存の作業計測システムを拡張し，作業履歴の改ざんを検出可能か確認する．

[1] 奈良工業高等専門学校
National Institute of Technology, Nara College

ⓒ 2014 Information Processing Society of Japan

2. 関連研究

2.1 第三者評価

業務の実施者，および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対象を評価することを第三者評価という．実施者や利用者から独立した第三者が，監査や作業履歴に基づいて評価・認証をすることで客観的な評価が期待できる．第三者評価はサービスや製品の品質向上を目的として実施され，医療 [7] や保育園 [8] などのサービス分野においても実施されている．

ソフトウェアの分野においても，宇宙機開発などで実施されているソフトウェア IV&V や，一般社団法人コンピュータソフトウェア協会が実施するパッケージソフトウェア品質認証制度（PSQ 認証制度）が存在する．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，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検証することで，ソフトウェアの品質を客観的に保証する．ソフトウェア IV&V では，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評価する．PSQ 認証制度[*1]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ソフトウェアの機能が一致しているか評価する．PSQ 認証制度では，ソフトウェアにおける機能の有無やその内容に着目することでソフトウェアの品質を評価する．

これらの取り組みでは開発作業の良否に起因した不具合の可能性や，各開発プロセスに要した工数や作業の妥当性については評価の対象としていない．本研究では，ソフトウェア開発組織に所属する開発者の作業履歴に着目した評価をするための手法を提案する．開発工程ごとにかけた時間数や各工程で用いているツールなどの利用履歴から開発作業の良否に起因した不具合の可能性や工数の妥当性を評価できるよう，利用履歴が改ざんされていないことを保証する．

2.2 作業計測システム

Watts によって提唱されている 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し，ソフトウェア品質を向上させる手法である [10]．PSP/TSP では，各作業に必要な時間を事前に見積もり，計測したデータから見積もりとの差異，および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，ソフトウェア開発者の作業履歴計測は紙を用いた手作業による計測のためコストが大きかったが，これまでに，多数の研究において計測支援システムが提案されている [11], [12].

TaskPit は開発者の PC 上での作業を自動的に計測し，作業履歴として記録するシステムである [11]．TaskPit が計測した作業履歴を用いることで開発プロセスの分析と改善が可能であることに加え，作業履歴を認証組織に渡すことで開発作業履歴に基づいた第三者評価が可能である．しかし，TaskPit は開発者や開発組織が自身の作業履歴を記録・分析することを目的に作成されているため，開発者や管理者が作業履歴を自由に閲覧・変更できる．したがって，認証組織に作業履歴を渡す際に，開発組織が自身の評価を高めるために作業履歴を改ざんして虚偽報告する可能性がある．また，作業履歴が改ざんされなかった場合でも，改ざんがないことを認証組織やそれ以外の外部組織に示すことができないため，第三者評価の信頼性が損なわれる恐れがある．作業計測支援システムには他にも Process Dashboard[*2], Task Coach[*3], Slim Timer[*4], Manic Time[*5]などが提案されているが，これらのシステムも TaskPit と同様に改ざんを検出できず，第三者評価に有用なシステムが存在しないのが現状である．

本稿では開発組織による作業履歴の改ざんを検出可能な手法を提案する．また，TaskPit を拡張して提案手法を実装し，認証組織が改ざんのない作業履歴を評価可能なシステムを構築する．この際，TaskPit の本来の目的である，開発組織自身による作業履歴を利用した開発プロセスの改善を妨げない．

3. 作業履歴による開発プロセスの第三者評価

3.1 評価方法

本稿では開発者の作業履歴に基づいて開発プロセスを分析し，その妥当性を認証組織が評価できるように，作業履歴に対する改ざんを防ぐ手法を提案する．認証組織は作業履歴を元に，開発組織がテストに十分な時間をかけているか，実装工程で要求書の修正に多くの時間をかけていないかなど，開発プロセスの正しさを評価する．このとき，認証組織は開発者の作業履歴に記録された各作業がどんな目的で行われたか，その作業をある時点において実施するが妥当なのか判断する必要がある．しかし，認証組織は外部組織であるため，作業の目的や妥当性について作業履歴のみから判断するのは難しい．作業の妥当性を判断するためには，開発者がそれぞれの時点で関わっていたプロジェクトやプロジェクト内での役割，開発組織内で定められたプロセスについて，開発組織から情報を得る必要がある．

作業履歴に基づいた開発プロセスの第三者評価モデルを図 1 に示す．図は，開発組織で計測された作業履歴を元に，認証組織が開発組織を評価する様子を示している．開発組

[*1] www.csaj.jp/psq

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

ⓒ 2014 Information Processing Society of Japan

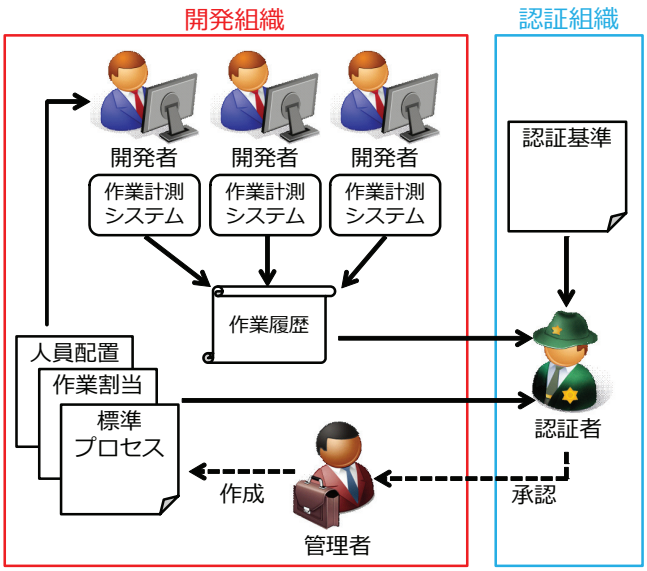


図 1 開発プロセスの第三者評価

織には開発者とその管理者が所属する．管理者はプロジェクトや部門ごとに配属され，複数の開発者について，配置や作業日程，業務内容を管理する．また，開発組織やプロジェクトごとに定められた開発プロセスにしたがって開発者が作業するよう管理する．開発者は定められた開発プロセスにしたがって，割り当てられた作業を実施する．このときの作業履歴は作業計測システムを利用して記録される．管理者は，管理を担当する開発者全員の作業履歴を収集し，プロジェクトや工程ごとに実施された作業の総工数や，各作業が開発プロセスにしたがっているか，割り当てた業務が問題なく実施しているかを把握する．管理者はこれらの情報を元に，プロジェクトの進捗状況を把握すると共に，開発プロセスの改善を行う．

認証組織には認証者が所属する．認証者は開発組織から作業履歴を受け取り，認証基準に基づいて開発プロセスを評価する．評価に当たっては，開発組織の定める開発プロセスにしたがって開発作業が行われているか，また設計や実装やテストなど各工程で，異なる工程でするべき作業をしていないかなど，様々な情報に基づいた評価が必要である．本稿では，具体的な評価方法は特定しない．認証者は評価結果に応じて開発組織のプロジェクトやソフトウェアの品質が確保されていることを保証する．

3.2 問題点

前節で説明した通り，開発組織によって収集された作業履歴は第三者評価における品質保証の根拠となる．そのため作業履歴には，実施した順序通りに漏れなく記録されていることと，開発者や管理者によって改ざんされていないことが求められる．既存の作業計測システムを用いることで，作業履歴を実施した順序通りに漏れなく記録することは可能だが，既存システムは作業履歴を第三者評価に利用することを想定していないため，作業履歴の改ざんを検出することができない．すなわち，1) 開発者や開発組織が作業履歴を改ざんしたことを認証組織が識別できない，2) 開発者や開発組織が改ざんをしなかった場合でも認証組織やそれ以外の外部組織に証明することができない．

また，作業履歴を記録するための作業計測システムにも改ざんの可能性がある．作業計測システムは作業履歴を記録するために開発組織内で利用される．そのため，3) 実際の作業とは異なる作業履歴を出力するよう作業計測システムを改ざんすることが可能である．

これらの問題は，いずれも作業履歴を用いた開発プロセスの第三者評価の正当性を失わせてしまう．本稿ではこれらの問題を解決するため，作業履歴および作業計測システムが開発者や開発組織に改ざんされていないことを客観的に示す手法を提案する．

4. 提案手法

前章で述べた問題を解決するため，本稿では作業計測システムと作業履歴それぞれに対する改ざんを検出する手法を提案する．提案手法のモデルを図 2 に示す．提案手法はクライアント・サーバ方式によるクライアントの認証と，ハッシュ値を利用した作業履歴の認証によって，開発組織による改ざんを検出する．次節以降では，提案手法による改ざんの検出方法を説明する．

4.1 クライアント認証

提案手法では，開発者の端末上で動作する作業計測システムをクライアントとして，認証組織に設置されたサーバに対してクライアントの実行ファイルを元に計算したハッシュ値を送信することで，クライアントを認証する．作業計測システムは認証組織から開発組織に提供されるものとし，サーバによる認証を受けなければ作業履歴を計測できない．サーバは認証を要求してきたクライアントからクライアントの実行ファイルを元に計算した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ハッシュ値と比較する．クライアントのハッシュ値を送信する際には盗聴によるなりすましを防止するためにワンタイムパスワードを用いて暗号化する．

サーバによるクライアント認証は次の手順で行われる（図 2）．

(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

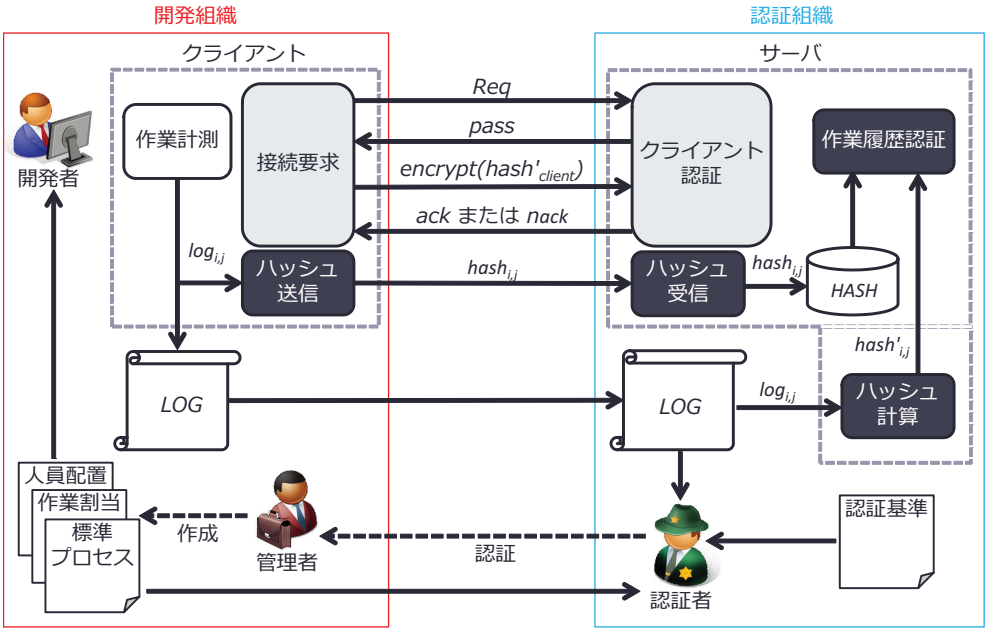


図 2 提案手法

$encrypt(hash'_{client})$ 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，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．また，クライアントは接続を要求するたびに $hash'_{client}$ を送信するため，パケットを盗聴することで，改ざんのないクライアントのハッシュ値 $hash'_{client}$ を知られる恐れがある．$hash'_{client}$ を知られた場合，改ざんした作業計測システムに $hash'_{client}$ を送信させることで，誤認証してしまう恐れがある．そのため，提案手法ではワンタイムパスワードを用いた暗号化を行っている．安全性をより高めるためには公開鍵暗号方式などを利用することが考えられるが，今後の課題とする．

4.2 作業履歴認証

本手法は作業履歴を作業計測システム（クライアント）が計測する際にハッシュ値を計算し，サーバに送信することで作業履歴に対する改ざんを検出可能にする．作業計測システムは開発者が利用する端末上で動作し，開発中のすべての作業を記録し，端末上に保存する．このとき，クライアントは履歴が更新されるたびに，差分からハッシュ値を計算しサーバに送信・蓄積する．認証組織は開発組織を認証評価する際に，開発組織が収集した作業履歴を受け取り，サーバに蓄積されたハッシュ値から改ざんの有無を検出する．

作業履歴認証は，履歴のハッシュ収集フェーズと認証作業フェーズの 2 つに分かれる（図 2）．履歴のハッシュ収集手順を以下に示す．

(1) クライアントは開発者 $dev_i(i = [1, 2, \ldots, n])$ の作業計測を開始する．

(2) dev_i の作業 $ope_j(j = [1, 2, \ldots, m])$ ごとに作業履歴 $log_{i,j}$ が記録され，開発組織内で全開発者の作業履歴 LOG として蓄積される．

(3) クライアントは $log_{i,j}$ からハッシュ値 $hash_{i,j}$ を計算し，サーバに送信する．

(4) サーバは $hash_{i,j}$ を受信時刻とともに $HASH$ として記録する．

作業履歴の認証手順を以下に示す．

(1) 開発組織は組織内で記録された LOG を認証組織に送る．

(2) 認証者は LOG から，$log_{i,j}$ を取り出してハッシュ値 $hash'_{i,j}$ を計算し，保存しておいた $hash_{i,j}$ と比較する．

(3) 認証者はすべての $hash'_{i,j}$ と $hash_{i,j}$ が一致していれ

ⓒ 2014 Information Processing Society of Japan

タスク名,	開始日時,	終了日時,	左 click,	右 click,	打鍵,	exe 名
プログラミング,	05/02 10:12:29,	05/02 10:16:40,	27,	3,	192,	devenv.exe: TaskPit
デバッグ,	05/02 10:16:40,	05/02 10:17:23,	3,	1,	0,	devenv.exe(実行中): TaskPit
文書編集・閲覧,	05/02 10:17:23,	05/02 10:17:58,	2,	0,	0,	sakura.exe: tasklog.csv
データ編集・閲覧,	05/02 10:17:58,	05/02 10:20:30,	5,	2,	66,	EXCEL.EXE: tasklog.csv

図 3 TaskPit の出力例

```
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 クライアントのバイナリ書き換え

ば開発組織内での改ざんなしとして，*LOG* に基づいた作業評価を実施する．

(4) 認証者は $hash'_{i,j}$ と $hash_{i,j}$ が 1 つでも一致しなければ開発組織内での改ざんありとして，作業履歴を評価しないなどの対処をする．

改ざんされていない作業履歴に基づいて作業評価する方法として，作業履歴 $log_{i,j}$ を直接サーバへ送信する方法も考えられる．しかし，複数の開発組織に存在する多数の開発者の作業履歴を随時送信することは開発組織のネットワークや認証組織のサーバにとって高い負荷になる恐れがある．また，作業履歴には開発組織が秘匿したい開発プロセスや情報が含まれている恐れがあるためネットワーク上に平文のまま送信することはセキュリティ上好ましくない．各種暗号化方式を用いることで作業履歴に含まれる情報を秘匿できるが，作業計測システムの処理が重くなるため，開発者の作業に影響する恐れがある．そのため，提案手法ではクライアントの負荷が小さく，作業履歴のネットワーク上での漏洩の恐れがないハッシュを用いた方法を採用した．

4.3 実装

提案手法に基づき，作業履歴の計測と認証者へのハッシュ送信が可能なサーバ・クライアントシステムを実装した．実装したシステムはサーバ，クライアントともに Windows で動作する GUI アプリケーションとして実装した．実装には Microsoft .Net Framework 2.0 を利用した．

サーバは C#言語を用いて作成した 458 行のプログラムで，複数クライアントからの認証要求や，作業履歴ハッシュを受信する機能を持つ．サーバはクライアントから認証要求があるごとにその結果を表示する．また，クライアントから受信した作業履歴のハッシュと，作業履歴から改ざんの有無を検証する機能も実装した．作業履歴の改ざんが検出された場合，改ざんが検出された作業履歴の行と内容を表示する．

クライアントは既存の作業計測システム TaskPit[11] を拡張して作成した．TaskPit は C#言語で作成されたプログラムで，端末上で利用されたアプリケーションに対するクリック数や打鍵数，およびアプリケーションの切り替えを記録できる．図 3 に TaskPit が出力する作業履歴の例を示す．本実装では，TaskPit に自身のハッシュ値と作業履歴のハッシュ値を計算・送信する HashSender クラスを追加した．HashSender は 258 行，9 つのメソッドを持つクラスで，サーバへのクライアント認証機能および作業履歴のハッシュ送信機能を担う．サーバによるクライアント認証に失敗した場合，クライアントは作業を計測せず，改ざんが検出された旨を利用者に表示してプログラムを終了する．

5. 実験

提案手法に基づいて実装したシステムが作業履歴と作業計測システムに対する改ざんを検出できるか実験で検証する．実験では 1) クライアントに対する改ざんを検出できるか，2) 作業履歴に対する改ざんを検出できるかを確かめる．

5.1 クライアント認証の検証

5.1.1 検証方法

クライアントを正しく認証できるか確かめるために，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 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．検証の手順を以下に示す．

手順 1 改ざんされていないサーバとクライアント，改ざんされたクライアントをそれぞれ異なる端末にインス

ⓒ 2014 Information Processing Society of Japan

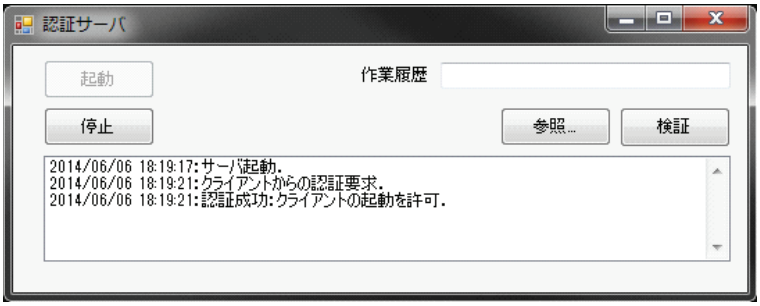
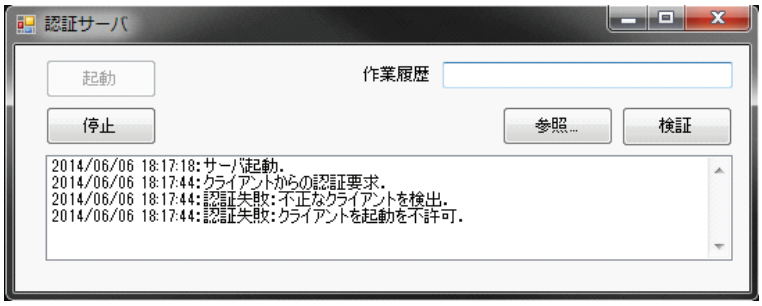
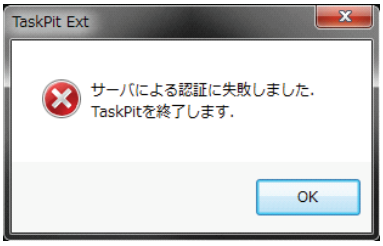


図 5 改ざんのないクライアントへの起動許可



(a) サーバの出力結果



(b) クライアントシステムの表示メッセージ

図 6 改ざんされたクライアントへの起動拒否

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 改ざんされたクライアントとを起動し，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.

5.1.2 結果

図 5 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．また，起動後のクライアントは TaskPit の設定に従って，一定周期ごとに作業履歴を端末上に出力していることが確認できた.

図 6 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 (a) とクライアントの表示 (b) を示す．図 6 (a) から，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，図 6 (b) に示した表示の後，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，提案手法に従って実装されたシステムにより，クライアントに対する改ざんが検出できるといえる.

5.2 作業履歴認証の検証

5.2.1 検証方法

作業履歴を正しく認証できるか確認するために，クライアントを用いて記録した作業履歴を改ざんし，作業履歴の認証時に改ざんが検出されることを確認する．図 7 (a) にクライアントを用いて計測した作業履歴の一部を，図 7 (b) に改ざんした作業履歴を示す．作業履歴を収集した開発者は，配置されたプロジェクトにおいて実装工程におり，Eclipse を用いて Java のソースコードを作成している．開発者は必要に応じて要求仕様書 (Req.pdf) や設計書（設計書.docx）を閲覧しながら開発を進める．社内のプロセスとして，開発工程では設計書や仕様書の編集は許可されておらず，プロジェクトチーム内でのレビューを経てのみ設計や仕様の変更が可能であるとする．ここで，開発者は実装中に見つけた設計上の不具合をプロセスに反して直接修正をしている．図 7 (a) の 14 行目に記録されたこの作業は，打鍵数から設計書を修正したことが予測でき，そのため，開発者は図 7 (b) に示すとおり実装作業として改ざんした．この開発履歴に対する検証の手順を以下に示す.

手順 1 改ざんされていないサーバとクライアントをそれぞれ異なる端末にインストールする.

手順 2 サーバとクライアントを起動し，クライアントによる作業計測を開始する.

手順 3 作業を実施し，図 7 (a) の内容を含む作業履歴を出力する.

手順 4 手順 3 で作成された作業履歴を改ざんせずにサーバが動作している端末にコピーし，サーバの履歴検証機能で改ざんが検出されないことを確認する.

ⓒ 2014 Information Processing Society of Japan

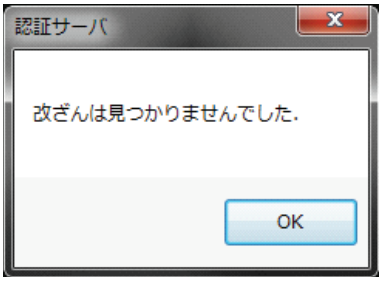
行,	タスク名,	開始日時,	終了日時,	左 click,	右 click,	打鍵,	exe 名
12,	プログラミング,	05/13 16:08:46,	05/13 16:13:49,	2,	0,	306,	eclipse.exe: Java
13,	PDF 閲覧,	05/13 16:13:49,	05/13 16:13:52,	2,	0,	0,	Acrobat.exe: Req.pdf
14,	文書編集・閲覧,	05/13 16:13:52,	05/13 16:16:19,	7,	1,	92,	WINWORD.EXE: 設計書.docx

(a) 改ざん前

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,	PDF 閲覧,	05/13 16:13:49,	05/13 16:13:52,	2,	0,	0,	Acrobat.exe: Req.pdf
14,	プログラミング,	05/13 16:13:52,	05/13 16:16:19,	7,	1,	92,	eclipse.exe: Java

(b) 改ざん後

図 7 作業履歴の改ざん



(a) 改ざん無し

(b) 改ざん有り

図 8 作業履歴の改ざん検出

手順 5 手順 3 で作成された作業履歴を図 7 (b) の通り改ざんし，サーバの履歴検証機能で改ざんが検出されることを確認する．

5.2.2 結果

図 8(a) に改ざんされていない作業履歴の検証結果を示す．図は改ざんされていない作業履歴に対して，サーバが正しく検証できたことを示している．図 8(b) に改ざんされた作業履歴の検証結果を示す．図は，作業履歴の 14 行目が改ざんされたことをサーバが正しく検出できたことを示している．また，改ざんされていない他の行に対して，改ざんは報告されなかった．

検証の結果，作業履歴が改ざんされたときに，どの行が改ざんされたか検出できることが確認できた．しがたって，提案手法に従って実装されたシステムにより，作業履歴に対する改ざんが検出できるといえる．

5.3 考察

実験の結果，提案手法を実装したシステムはクライアントと作業履歴に対する改ざんを検出できることが確認された．クライアントに対する改ざんでは，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．しかし，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異な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．一般に実行ファイルの改ざんは容易ではないため，従来の作業計測システムと比べると改ざんを防止する能力は向上したと言えるが，実行ファイルの暗号化や通信内容の秘匿を適用することで，改ざんを困難にすることは今後の課題である．

作業履歴に対する改ざんでは，作業履歴中の 1 行を直接書き換えた．作業履歴の認証にもハッシュ値を用いたため，クライアントの認証と同様に他の行に対する改ざんであっても検出が可能と考えられる．一方で，評価者が開発組織を評価するために必要な開発組織の標準プロセスやプロジェクトの人員配置，作業割当については改ざんされる可能性がある．例えば，作業履歴を改ざんする代わりに作業割当や標準プロセスを変更することで，本来であれば不適当な作業をしていた場合であっても評価者は検出できない．今後，作業履歴以外の情報についても定期的にハッシュ値を送信するなどの方法で改ざんを防ぐ事が考えられる．

本稿ではクライアントを実装するために既存システムで

ⓒ 2014 Information Processing Society of Japan

ある TaskPit を用いた．TaskPit は実行ファイルの他にも複数の DLL を利用しており，実行ファイルと同様に改ざんの対象になる恐れがある．しかし，DLL のハッシュ値などを用いて起動時に検証することで実行ファイルと同様に改ざんを検出できると考えられる．また，TaskPit 以外の作業計測システムであっても，実行ファイルや DLL など動作に必要なファイルを起動時に検証することで改ざんを検出できると考えられる．

6. おわりに

本稿ではソフトウェアの開発作業履歴を用いた開発組織の第三者評価のために，作業履歴に対する改ざんを防ぐ手法を提案した．提案手法は作業計測システムが出力する作業履歴と計測システム自身のハッシュ値を評価組織に随時送信することで，開発組織による作業履歴および作業計測システムの改ざんを検出する．提案手法を実装したシステムに対して，検証実験を行った結果，作業計測システムと作業履歴に対する改ざんを検出できることが確認された，

今後，提案手法を用いた第三者評価に対して，開発組織による改ざんの恐れがある箇所や，外部組織から第三者評価の正当性を疑われる恐れのある箇所を攻撃者モデルとして整理し，提案手法によって改ざんが防止できるか評価する．また，第三者評価組織に設置されるサーバには多数の開発者からアクセスがあるため，ネットワークやサーバ機に高い負荷がかかる可能性がある．作業計測システムについても提案手法を適用することでシステムの処理が重くなり，開発者の作業に支障が出る恐れがある．今後，提案システムによるサーバやクライアント，ネットワークへの負荷を計測し，第三者評価手法として有用であるか確認することも重要である．

謝辞

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] 情報処理推進機構：“ソフトウェア品質説明力強化に向けた実験報告書，” (2013).
[3] National Aeronautics and Space Administration：“NASA Independent Verification and Validation Program，” (2009).
[4] 宇宙航空研究開発機構：“IV & V ガイドブック”，(2013).
[5] 宇宙航空研究開発機構：“ベストプラクティス調査報告 – 究極の高品質ソフトウェア開発プロセスをめざして，” (2007).
[6] 秋山 昌範: “クラウドコンピューティング時代に必要なデジタル・フォレンジック，”日本セキュリティ・マネジメント学会誌，Vol.23，No.1, pp.61–67, (2009).
[7] 株式会社 明治安田生活福祉研究所（厚生労働省医政局委託）：“医療施設経営安定化推進事業，” (2013).
[8] 杉並区役所：“平成 24 年度 保育園サービス 第三者評価事業報告書，” (2012).
[9] 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).
[10] 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).
[11] 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).
[12] 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).
[13] 総務省，法務省，経済産業省：“電子署名及び認証業務に関する法律に基づく特定認証業務の認定に係る指針，” (2002).
[14] 原田篤史，西垣正勝，曽我正和，田窪昭夫，中村逸一：“ライトワンス文書管理システム，”情報処理学会論文誌，Vol.44，No.8, (2003).

ⓒ 2014 Information Processing Society of Japan