# 卒　業　研　究　報　告　書

平成25年度

研究題目

## ソフトウェア開発における<br>作業履歴の非改ざん性保証と共有手法


**指導教員**　　上　野　秀　剛　　助教

**氏　　名**　　池　田　祥　平


平成26年1月7日　提出


**奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科**

# ソフトウェア開発における<br>作業履歴の非改ざん性保証と共有手法

上野研究室　池田祥平

ソフトウェア品質を確保するための取り組みの1つであるソフトウェアIV&Vでは，開発組織から技術面・組織面・資金面で独立した第三者組織が，開発プロセスおよび開発製品の正当性を客観的に評価することで，ソフトウェア品質を保証し，確保を目指す. しかし，ソフトウェアの分野においては品質管理の問題は開発組織に任されており, ソフトウェアの品質評価は開発組織が組織内に独立した部門を設置することで行われている. 独立した第三者組織による客観的な評価が行えておらず, ソフトウェアの品質を保証するには弱い. 本研究では開発組織が正しい開発プロセスで開発を行っていることを作業履歴の第三者評価から保証することを目的とし，独立した第三者組織が改ざんのない作業履歴を評価可能な手法を提案する. 提案手法ではハッシュ関数を利用したサーバ・クライアントシステムにより作業計測システムと作業履歴ファイルの改ざんを検出し，開発組織による改ざんを防ぐことで改ざんのない作業履歴を評価可能にする. 提案手法の有効性を検証する実験では，システムを運用するうえでの作業計測システムと作業履歴ファイルの改ざん検出は可能であることが分かった.

# 目 次

# 1 はじめに

近年の情報化社会の発展に伴い，製品やサービスの高機能化・多様化は急速に進み，その開発や構築においてソフトウェアの果たす役割が飛躍的に拡大している．そのため要求通り作られているか，誤作動がないかといったソフトウェア品質の確保は重要な課題となっている[1][2][3].

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：独立検証及び妥当性確認）がある[4]. ソフトウェアIV&Vとはソフトウェアを開発する組織（以下，開発組織という）から技術面・組織面・資金面で独立した組織（以下，認証組織という）が，分析・設計・開発・製造などの各開発プロセス正しく行われているか（検証），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（妥当性確認）を評価することである．これにより，ソフトウェア品質を保証し，確保を目指す. ソフトウェアの品質を保証する場合，開発組織自身がいくらテストやレビューを行ったとしても客観的な証拠にはならず，保証するには弱い．ソフトウェアIV&Vでは開発組織（第一者），製品利用者（第二者）の両者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を行うことで，客観的に品質を保証している．ソフトウェアIV&Vは信頼性の高い，安全なシステムを求めてNASAやJAXAで実施されている[5][6][7]. しかし現状，NASAやJAXAでは各自の系列の会社に委託して評価を行っているため，完全に独立した組織による評価とはいえず，結果として客観的な証拠による品質の保証ができているとはいえない．品質の保証には完全に独立した認証組織による評価と保証が必要になる．

完全に独立した認証組織による評価と保証の仕組みとして，近年パッケージソフトウェア品質認証制度（PSQ認証制度）が一般社団法人コンピュータソフトウェア協会によって制定された[8]. 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，開発組織が提供するソフトウェアの機能が製品説明（カタログなど）と利用者用文書（マニュアルなど）と一致しているかを評価し保証することで，間接的にソフトウェア品質を保証する．PSQ認証制度により，完全に独立した認証組織によるソフトウェア品質の保証が可能である．しかしPSQ認証制度では，ソフトウェアにおける機能の有無やある程度の品質は保証可能であるものの，開発作業の良否に起因した不具合の可能性を評価していない．また開発・管理における工数・作業の妥当性が評価できないといった問題があり，品質保証には弱い面がある．開発作業の良否に起因した不具合の可能性の評価や開発・管理における工数・作業の妥当性の評価を可能にするには，開発組織がソフトウェアの開発のために行った開発プロセスを評価する必要がある．

一方，ソフトウェア品質の向上を目的とした手法としてPSP/TSP(Personal Software Process /Team Software Process)がWatts Humphreyによって提唱されている[9]. PSP/TSP

は開発組織が自身の作業履歴を記録し，コーディングやデバッグ，設計，会議といった個々の開発作業にどれだけの時間を費やしているかを計測することで，改善すべき作業の発見や開発プロセスの効率化を行い，ソフトウェア品質を向上させる手法である．作業履歴の計測および記録は開発組織にとっては負担であるため，これを支援するシステムが複数開発されている[10][11][12][13].

完全に独立した認証組織による開発プロセスの評価を行うことを考えたとき，必要となるのは開発プロセスの内容である．本研究では開発プロセスの内容のうち開発組織が行った作業履歴の評価を考える．既存のPSP/TSP支援システムによって作業履歴は記録可能であるため，認証組織はその作業履歴から開発組織が正しい開発プロセスで開発を行っているかを評価し，保証することが可能である．ソフトウェアを構成する要素の一つである開発プロセスを保証することで，間接的にソフトウェア品質を保証する．しかし既存のPSP/TSP支援システムは開発組織自身の開発プロセスの分析を補助するためのものであり，作業履歴は開発組織によって自由に閲覧・編集が可能である．編集されたことを検出する機能もないため，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，虚偽の作業履歴を報告しても，認証組織はそれを検出できず，誤った保証をしてしまう恐れがある．そこで，本研究では開発組織による作業履歴の改ざんを検出可能とし，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提案する．

# 2 関連研究

## 2.1 作業計測システム

Wattsによって提唱されているPSP/TSPは，開発組織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[9]. 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（1996年）は，作業履歴の計測は紙を用いた手作業によって行われており，計測の際に多大なコストがかかるとされてきた．そのため計測支援システムとしてTaskPit[13]が提案されている．

TaskPitは開発組織の作業を自動的に計測し，作業履歴として記録することが可能である．認証組織による評価を行う際にこのシステムを利用することで，認証組織による保証だけでなく，記録した作業履歴から開発組織自身がPSP/TSPにより開発プロセスの改善を行うことが可能になる．この場合に問題となるのは記録した作業履歴が改変されることを検出できないことである．開発組織が自身で作業履歴の分析を行う場合には改変が検出できなくとも問題はないが，第三者による評価においては問題である．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，虚偽の作業履歴を報告しても，認証組織はそれを検出できず，誤った保証をしてしまう恐れがある．作業計測支援システムにはTaskPitの他にProcess Dashboard[10]，Task Coach[11]，Slim Timer[12]が提案されているが，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．

そこで本研究ではTaskPitを拡張して，開発組織による作業履歴の改ざんを検出可能とし，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提案する．またこの際，開発組織自身による作業履歴を利用したプロセス改善を妨げない．

## 2.2 電子文書の保護

電子文書は電子的に記録された記録のことで，会議資料や顧客名簿といった重要性の高いデータの保存手段として利用されている．このような，改ざんされたときの被害が大きいデータを記録した電子文書は，改ざんを防ぐ保護が必要である．そこで，改ざんを防止する仕組みの1つとして電子署名がある[14]. 電子署名は現実世界での印鑑の押印に相当するもので，電子文書の作成者を証明が可能な仕組みであるが，改ざん防止にも利用できる．

電子署名では，まず元の文書のハッシュ値をハッシュ関数により計算し，公開鍵暗号方式における「秘密鍵」でハッシュ値を暗号化する．暗号化されたハッシュ値は電子文書とともに保管される．そして電子文書を利用する際,「公開鍵」によりハッシュ値を複合化しする．同時にともに保管されていた電子文書のハッシュ値を

計算する．この2つを比較することで改ざんの検出が可能となる．ハッシュ関数は同じ入力に対して常に同じハッシュ値を生成する一方，少しでも入力が異なるとまったく異なるハッシュ値を生成するため，利用時の電子文書のハッシュ値と元の文書のハッシュ値が異なれば，改ざんされていることになる．電子文書の保護について原田らは，ライトワンス文書管理システムを提案している[15]．ライトワンス文書とは追記によってのみ更新が許される電子文書のことで，このシステムでは電子署名を2重に利用することで改ざん検出能力を強化している．

本研究ではハッシュ関数を利用することで，改ざんを検出可能にし，作業履歴を保護する．

# 3 第三者評価

## 3.1 第三者評価組織の役割

当事者（開発組織および利用者）以外の公正・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. 本論文では第三者評価を行う組織を, 開発組織に対応して認証組織と呼ぶ.

ソフトウェアの品質を利用者に保証することを考えたとき, 保証するに足る証拠が必要になる. しかし開発組織自身がいくらテストやレビューといった評価を行ったとしても客観的な証拠にはならず, 保証するには弱い. たとえば評価の結果重大な欠陥が発見されたとき, その欠陥を修正するコストを恐れて, 欠陥がなかったというように評価結果を利用者に対して偽造する可能性などが考えられるためである. これに対し, 認証組織は当事者から独立しているため, 評価結果を開発組織に有利にまたは利用者に有利になるように偽造することで得られる利点はない. そのため, 認証組織による評価結果は客観的な証拠としてソフトウェアの品質を保証するのに役立つ.

ソフトウェアの品質が第三者の認証組織によって保証されていることは, 利用者に品質の情報を与え, 結果として安心して利用できるようにする. また開発者は保証を得ようとソフトウェア品質の改善を図るため, ソフトウェア市場の競争力強化, ひいてはソフトウェアを利用する国内産業の育成にもつながる.

## 3.2 ソフトウェア品質評価の現状

近年の情報化社会の発展に伴い, 製品やサービスの高機能化・多様化は急速に進み, その開発や構築においてソフトウェアの果たす役割が飛躍的に拡大している. そのため要求通り作られているか, 誤作動がないかといったソフトウェア品質の確保は重要な課題となっている[1][2][3].

品質の確保が重要なのはソフトウェアに限った話ではなく, 医療[16]や保育園[17]などのサービスの分野でも品質向上を目的として, 独立した組織による第三者評価といった取り組みが実施され, 成果が得られている.

ソフトウェアの分野においてもソフトウェア品質を確保するための取り組みとして, 独立した認証組織による第三者評価を行うソフトウェアIV&Vがある[4]. ソフトウェアIV&Vでは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・設計・開発・製造などの各開発プロセス正しく行われているか（検証）, 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（妥当性確認）を客観的に評価することで, ソフトウェアの品質を保証し, 確保を目指す. ソフトウェアIV&Vは, 信頼性の高い安全なシステムを求めてNASAやJAXAで実施されている[5][6][7].

しかし, ソフトウェアの分野においては品質管理の問題は開発組織に任されており, 完全に独立した第三者による品質保証が可能な仕組みが十分に整備されて

いないのが現状である．そのため，NASAやJAXAは各自の系列の会社に委託してソフトウェアIV&Vを行っている．各自の系列の会社による評価では客観的であるとはいいづらく，ソフトウェア品質の保証には弱い．

そこで，完全に独立した認証組織が必要となる．ここで完全に独立しているとは技術面・組織面・資金面において，開発組織および利用者から独立しており，上記で示したように開発組織の系列の組織ではないとする．

近年，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，開発組織が提供するソフトウェアの機能が製品説明（カタログなど）と利用者用文書（マニュアルなど）と一致しているかを評価し，機能の品質を保証することで間接的にソフトウェアの品質を保証する，パッケージソフトウェア品質認証制度（PSQ認証制度）を制定した[8]．PSQ認証制度により，完全に独立した認証組織によるソフトウェア品質の保証が可能となった．しかしPSQ認証制度では，ソフトウェアにおける機能の有無やある程度の品質は保証可能であるが，開発作業の良否に起因した不具合の可能性を評価していない，また開発・管理における工数・作業の妥当性が評価できないといった問題があり，品質保証には弱い面もある．

本研究ではソフトウェアの開発プロセスに着目し，評価することで開発作業の良否に起因した不具合の可能性の評価や開発・管理における工数・作業の妥当性の評価を可能にする．よって本研究における認証組織は，客観的な立場で開発組織の行った作業（作業履歴）を評価し，開発組織が正しい開発プロセスで開発を行っていることを保証する組織を想定する．ソフトウェアを構成する要素の一つである開発プロセスの品質を保証することにより，間接的にソフトウェア品質を保証する．

認証組織の具体的な業務としてはまず開発組織から開発作業を保証する依頼を受ける．認証組織は評価するために作業計測のためのシステムを提供し，そのシステムが計測した結果の作業履歴を評価する．作業履歴の評価を行い，正しい開発プロセスで開発を行っているかを保証し，その内容をわかりやすく利用者に伝える．認証組織は複数の開発組織の保証を請け負う．

本研究では認証組織による開発組織の作業履歴の評価を行うため，作業履歴の改ざんを防ぐシステムを提案する．

# 4　提案手法

本章では，既存の作業計測システムを利用した履歴共有手法の問題点を明確にした上で，本研究が提案する作業履歴の共有手法について述べる．認証組織は複数の開発組織と作業履歴を共有する必要があるが，本研究では基礎的な手法であると位置づけ，認証組織と開発組織が1対1で共有を行う手法を提案する．

## 4.1　作業履歴共有システム

開発作業を計測した結果である作業履歴は，開発組織においては開発プロセス改善のために，認証組織においては開発組織が正しい開発プロセスで開発していることを保証するために，それぞれ利用される．そのため，作業履歴は両者の組織間で共有される必要がある．

2章で示したような作業計測システムを利用すると，計測した作業履歴は開発組織内にファイルとして記録される（以下，作業履歴ファイルという）ので，開発組織はそれを分析することでプロセス改善を図れる．また，認証組織は評価を行うタイミングで，開発組織内の作業履歴ファイルを受け取ることで，評価が可能になる．図1に共有システムのモデルを示す.

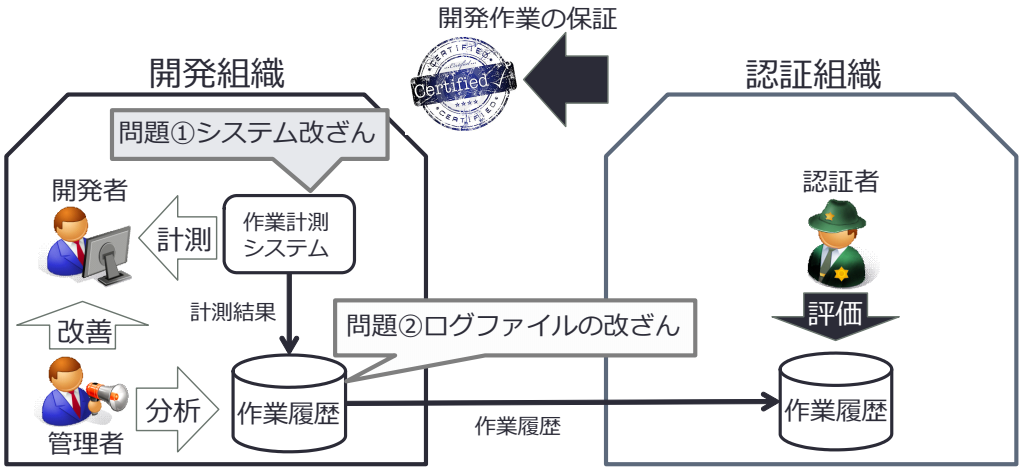



図1 単純な共有システム

図中の円柱はデータを記録したファイル，矢印はデータの流れを意味する．モデルでは開発組織内に開発者と管理者，認証組織内に認証者が存在する．開発者がソフトウェアを開発する間，作業計測システムが開発作業を計測し，作業履歴として開発組織内の作業履歴ファイルに記録する．管理者は作業履歴ファイルを閲覧・分析することで改善点を見つけ，開発者の作業方法や開発プロセスを改善する．認証者は例えば年1回のような評価を行うタイミングで，開発組織から作

業履歴ログを受け取り, 評価することで開発組織が正しい開発プロセスで開発していることを保証する.

## 4.2 問題点

図1のモデルには, 第三者評価の実施において問題点がある. 第三者評価を行うには, 認証者が開発者の作業履歴を正しく知る必要がある. 図1のモデルでは, 認証者が閲覧する作業履歴の正当性がこのシステムでは保証されていない. つまり, 開発組織が自身の評価を高めるために作業履歴ログを改ざんし, 虚偽の作業履歴を報告しても, 認証組織はそれを検出できず, 誤った保証をしてしまう恐れがある. 図1中に示している通り, 具体的に作業履歴を改ざんする方法として以下の2つがある.

問題1 システムの改ざん

開発作業を計測する作業計測システム本体を対象に改ざんを行い, 誤った作業履歴を作業履歴ファイルに記録させる.

問題2 ログファイルの改ざん

作業計測システムが計測した, 作業履歴を記録した開発組織内の作業履歴ファイルを対象に改ざんを行い, 誤った作業履歴に改変する.

## 4.3 サーバ・クライアントシステム

前節の問題を解決するため, 本研究ではハッシュ関数による検証を利用したサーバ・クライアントシステムを提案する. ハッシュ関数は同じ入力に対して常に同じハッシュ値を生成する一方, 少しでも入力が異なるとまったく異なるハッシュ値を生成する. 2つのファイルのハッシュ値をそれぞれ計算し, 比較することで, 同一のファイルであるか, または異なるファイルであるかを確認可能になる. 図2にサーバ・クライアントシステムのモデルを示す.

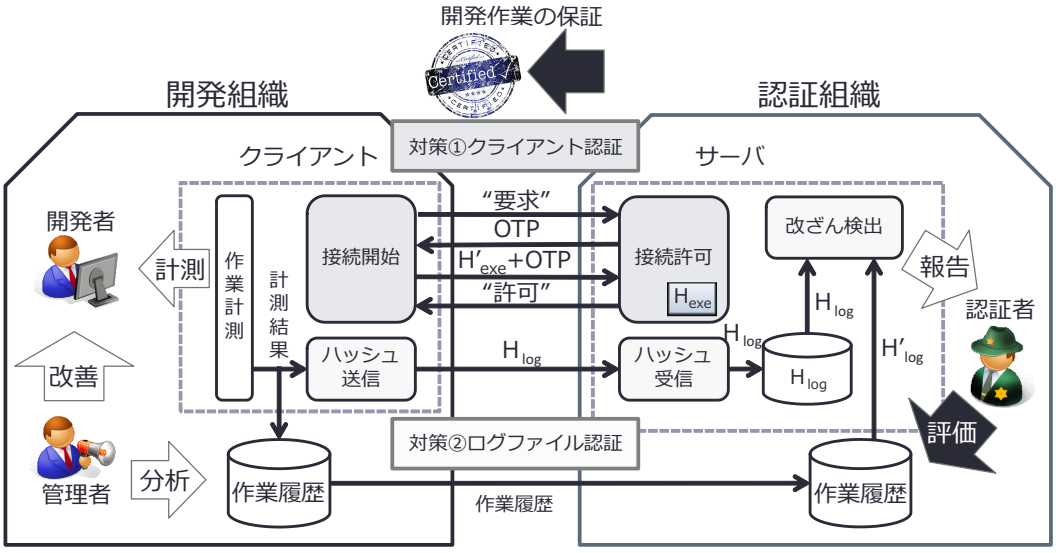



図 2 サーバ・クライアントシステム

提案システムは，開発組織がクライアントシステム（以下，クライアントと略記）を利用し，認証組織がサーバシステム（以下，サーバと略記）を利用する．クライアントは作業計測システムを拡張したもので，元の機能である作業計測および記録機能に加え，サーバに対する接続要求とハッシュ送信機能を持つ．サーバシステムはクライアントの接続許可とハッシュ受信・記録，改ざん検出機能を持つ．クライアントは認証組織から開発組織へ提供される．

提案システムは前節の問題に対してそれぞれ以下の対策をしている．

対策 1 クライアント認証

クライアントは作業計測を始める前に，サーバからクライアント認証を受ける．認証でサーバとの接続が許可されればクライアントは作業計測を始め，拒否されればクライアントは作業計測を行わずに終了する．

クライアント認証の手順を説明する．認証組織は開発組織へクライアントを提供する前に，あらかじめ改ざんされていないクライアントの実行ファイルのハッシュ値 $H_{exe}$ を計算し，記録しておく．提供されたクライアントは作業計測を始める前にサーバとの接続を開始し，自身の実行ファイルのハッシュ値 $H'_{exe}$ を計算してサーバへ送信する．サーバは記録していた $H_{exe}$ と受信した $H'_{exe}$ を比較する．2つのハッシュ値が一致していればクライアントは改ざんされていないと判断できるため接続を許可し，異なっていれば改ざんがあると判断できるため拒否する．クライアントの実行ファイルのハッシュ値を比較することで，認証者はクライアントの改ざんを検出できる．

一方クライアント認証では，毎回同じデータ（クライアントのハッシュ値）を作業計測開始前に送信する．通信パケットをのぞき見るなどして毎回同じ

データを送信していることが改ざん者に知られると，改ざん者は改ざんしたクライアントに同様のデータを送信させるだけでサーバから許可を得られることになるため，クライアントの改ざんが容易になる．毎回同じデータを送信する事態を防ぐため，ワンタイムパスワードによる暗号化を導入した．クライアントがサーバへ接続を開始したとき，サーバはワンタイムパスワード$OTP$を生成し，クライアントへ送信する．$OTP$はクライアント認証のたびにサーバがランダムに生成する．クライアントは受信した$OTP$を$H'_{exe}$に加算して暗号化して，サーバへ送信する．サーバは$H_{exe}$に対して同様に$OTP$を加算し，加算結果を暗号化した$H'_{exe}$と比較することでクライアントの改ざん検出を行う．

対策2 ログファイル認証

サーバは認証者が作業履歴ファイルから作業履歴を評価する前に，ログファイル認証を行う．ログファイル認証により，作業履歴ファイルに改ざんがあるか否かが認証者に知らされる．

ログファイル認証の手順を説明する．クライアントは作業計測時に，計測した作業履歴を作業履歴ファイルに記録すると同時に，作業履歴のハッシュ値$H_{log}$を計算しサーバへ送信する．サーバは受信した$H_{log}$をハッシュファイルに記録する．認証組織が評価のタイミングで作業履歴ファイルを開発組織から受け取ったら，作業履歴ファイルに記録された作業履歴のハッシュ値$H'_{log}$を計算する．ハッシュファイルに記録された$H_{log}$と$H'_{log}$を比較することで作業履歴ファイルが改ざんされているかを検出する．一致していれば改ざんなし，異なれば改ざんありと認証者に報告して，認証者は作業履歴ファイルの改ざんを検出できる．

作業履歴の改ざんを防ぐ方法として，作業計測時に作業履歴をそのままサーバへ送信する方法も考えられるが，多数の開発組織を認証する場合に通信の負担になる．ハッシュ関数であれば作業履歴のデータに比べ容量が少ないため，ハッシュ関数による認証を採用した．

# 5 実装

前章で示した提案手法を実装した. 本章では, 実装したプログラムの仕様と具体的な実装方法について述べる.

## 5.1 概要

クライアントは既存の作業計測システムであるTaskPit[13]を拡張して実装した. TaskPitはC#言語で作成されているため, 作業履歴のハッシュを計算・送信するクラスHashSenderを作成し, 既存のシステムに追加した. HashSenderは200行のプログラムコードで, 9つのメソッドを持つ. HashSenderクラスは設定されたIPアドレス, ポートを用いてサーバと接続し, 与えられたデータのハッシュ値を計算し送信する. 以下にHashSenderの主要なメソッドの仕様を示す.

- サーバ接続開始メソッド

  クライアント認証を行い, 作業履歴のハッシュ値を送信するためにサーバとの接続を開始する.

  クライアント認証の手順を説明する. HashSenderクラスのコンストラクタで設定されたIPアドレス, ポートのサーバとの接続を開始し, ワンタイムパスワード$OTP$をサーバから受信する. 自身の実行ファイルのハッシュ値$H'_{exe}$を計算し, $OTP$と加算した値をサーバへ送信する. クライアント認証を受けた結果を受け取り, 許可されていれば接続を続け, 拒否されていれば接続を切断し, クライアントを終了する.

  サーバへの接続にはSystem.Net.Socketsライブラリ内のTcpClientクラスを, データの送受信には同ライブラリ内のNetworkStreamクラスを利用する. System.Net.Socketsライブラリはネットワークへのアクセスを厳密に制御する必要がある場合にWindowsソケット(Winsock)インターフェイスのマネージ実装を提供する.

- ハッシュ送信メソッド

  作業履歴のデータを引数として受け取り, ハッシュ値$H_{log}$を計算してサーバに送信する.

  サーバへの接続およびデータの送受信にはサーバ接続開始メソッドと同じく, TcpClientクラスおよびNetworkStreamクラスを利用する.

HashSenderをTaskPitに追加することで, 3章で説明した問題点に対応する事ができる. 次節以降では改良したTaskPitが2つの問題に対してどのように対応しているか説明する.

## 5.2 対策1　クライアント認証の実装

対策1のログファイル認証の実装方法について述べる．図3に対策1の処理の流れを示す.

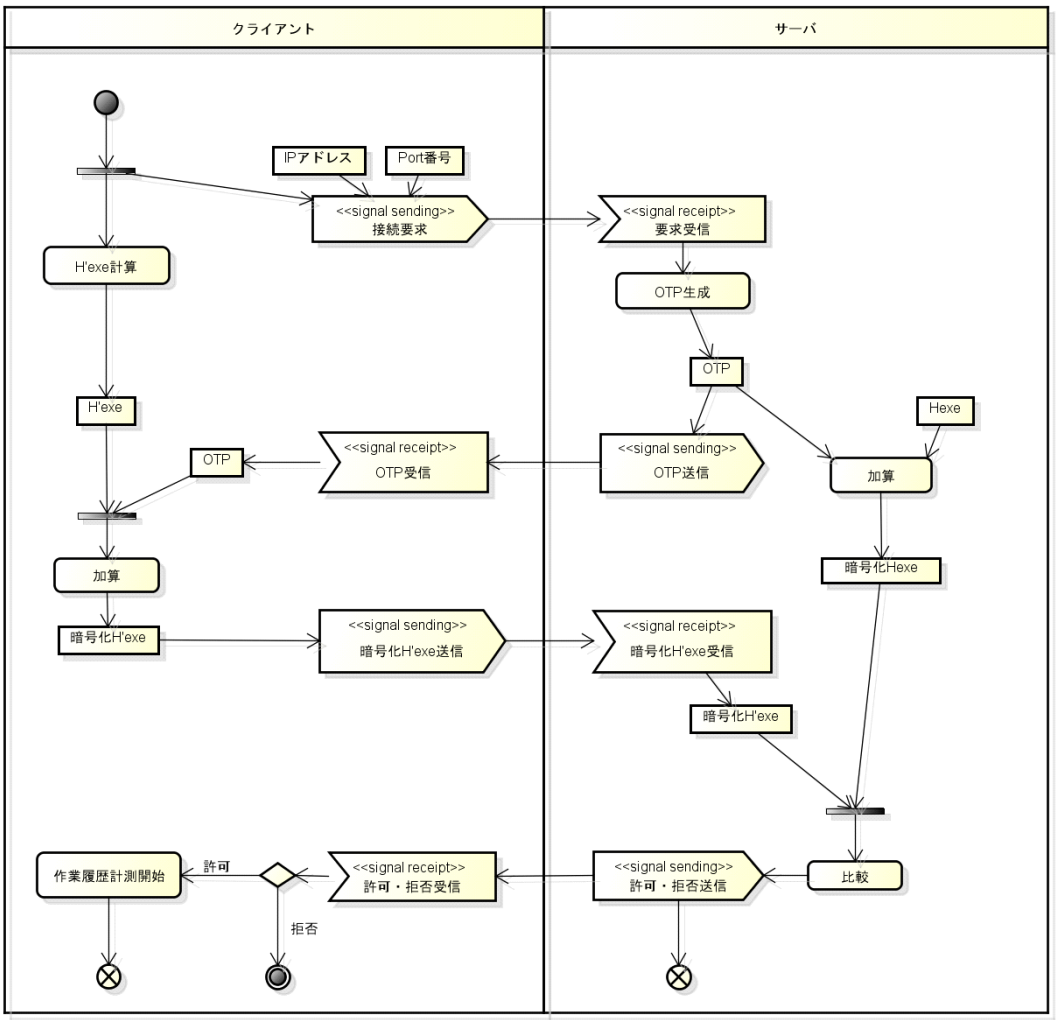



図3対策1の処理の流れ

問題1はクライアントが開発組織によって改ざんされてしまうことである．開発組織にあるクライアントの実行ファイルのハッシュ値$H'_{exe}$をサーバに記録されている改ざんされていないクライアントの実行ファイルのハッシュ値$H_{exe}$と比較することで，クライアントの改ざんを検出する．

比較するまでの手順を説明する．クライアントはコンストラクタで設定されたIPアドレス，ポートのサーバへ接続要求を送信すると同時に，自身の実行ファイルのハッシュ値$H'_{exe}$を計算する．接続要求を受信したサーバはランダムな数値列であるワンタイムパスワード$OTP$を生成し，クライアントへ送信すると同時に，記録している改ざんされていないクライアントの実行ファイルのハッシュ値$H_{exe}$に$OTP$を加算して暗号化$H_{exe}$を計算する．$OTP$は送信データが同一になるのを防

ぐために利用される．クライアントは$OTP$を受信したら$H'_{exe}$と加算し，計算した暗号化$H'_{exe}$をサーバへ送信する．暗号化$H'_{exe}$を受信したサーバは暗号化$H_{exe}$と比較する．

比較した結果，一致していれば改ざんなし，異なれば改ざんありとサーバは判断できる．サーバは判断結果に応じて接続許可または拒否をクライアントに送信する．クライアントは許可を受信すれば，作業履歴の計測を開始し，拒否を受信すればクライアントを終了する．

## 5.3 対策2　ログファイル認証の実装

対策2のクライアント認証の実装方法について述べる．図4に対策2の処理の流れを示す.

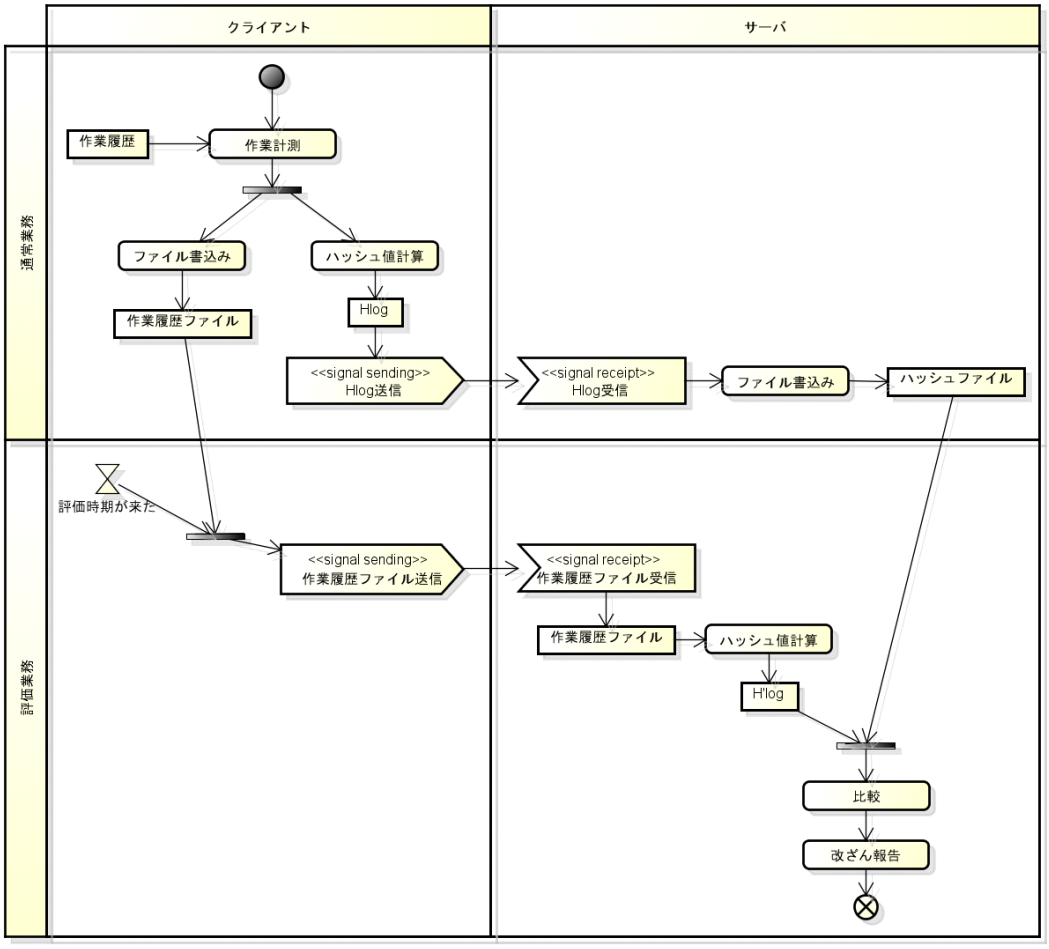



図 4 対策2の処理の流れ

問題2は作業履歴ファイルが開発組織によって改ざんされてしまうことである．評価時に認証組織が受け取った作業履歴ファイルに記録された作業履歴のハッシュ値$H'_{exe}$を作業計測直後の作業履歴のハッシュ値$H_{exe}$と比較することで，作業履歴

ファイルの改ざんを検出する.

比較するまでの手順を説明する.クライアントは作業履歴の計測後,作業履歴を作業履歴ファイルに記録すると同時に,ハッシュ値$H_{log}$を計算し,サーバへ送信する.サーバは$H_{log}$をハッシュファイルに記録する.作業履歴ファイルは認証組織による評価時期が来た時にサーバへ送信する.サーバは受信した作業履歴ファイルに記録された作業履歴のハッシュ値$H'_{log}$を計算し,ハッシュファイルに記録された$H_{exe}$と比較する.

比較した結果,一致していれば改ざんなし,異なれば改ざんありとサーバは判断できる.サーバは判断結果を認証者に報告する.

# 6 検証実験

本章では前章で実装したシステムが, 要求通り改ざんを検出可能かを検証する方法と結果について述べる.

## 6.1 概要

3章で説明した問題点は作業計測システムと作業履歴ファイルが改ざんされることであった. 4章で実装したシステムが, 2つを対象とした改ざんを検出可能であるかを検証する. 開発組織は認証組織から提供されたクライアントを対象に改ざんする. なお, クライアントと作業履歴ファイルに対してどんな作業履歴に改ざんするか, どのような方法で改ざんするかは議論しない.

検証方法の説明にあたり, 簡単にクライアントの動作について説明する. クライアントは実行中のソフトウェアを作業として計測し, 設定された時間間隔で作業履歴ファイルに作業履歴を記録する. また作業履歴ファイルに作業履歴を記録するタイミングで各作業履歴のハッシュ値を計算し, サーバへ送信する. 作業履歴ファイルに記録する時間間隔は1分とした.

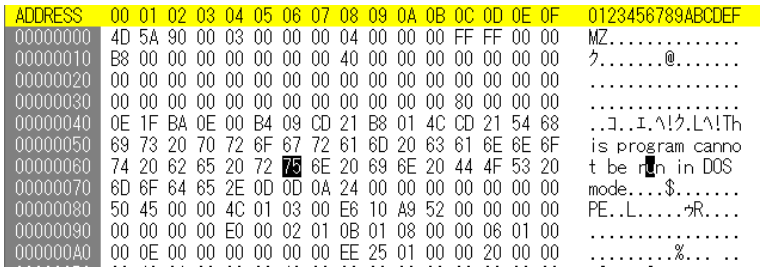
次節以降では作業計測システムと作業履歴ファイルの改ざんをそれぞれ行い, 改ざん検出が可能かを検証する.

## 6.2 対策1の検証

### 6.2.1 検証方法

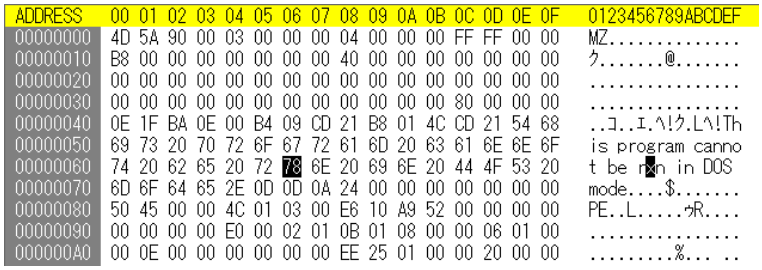
以下に示す, 手順1~5を順に行い, 改ざんされていないクライアントはサーバに許可されること, 改ざんされているクライアントはサーバに拒否されることをそれぞれ検証する.

1. 改ざんされていないクライアントとサーバをそれぞれ同じローカルサーバ内の別々のパソコンに導入する.
2. 導入直後のクライアントを実行しクライアント認証を行い, 改ざんされていないクライアントはサーバに許可されることを確認する.
3. クライアントを終了し, サーバとの接続を終了する.
4. クライアントの実行ファイルをバイナリエディタを用いて図5の“ run ”を図6の“ rxn ”に改ざんする.
5. 改ざんしたクライアントを実行しクライアント認証を行い, 改ざんしたクライアントはサーバに拒否されることを確認する.



| ADDRESS | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 0123456789ABCDEF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 4D | 5A | 90 | 00 | 03 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | FF | FF | 00 | 00 | MZ.............. |
| 00000010 | B8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ｸ.......@....... |
| 00000020 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ................ |
| 00000030 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 00 | 00 | 00 | ................ |
| 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.....ｩR.... |
| 00000090 | 00 | 00 | 00 | 00 | E0 | 00 | 02 | 01 | 0B | 01 | 08 | 00 | 00 | 06 | 01 | 00 | ................ |
| 000000A0 | 00 | 0E | 00 | 00 | 00 | 00 | 00 | 00 | EE | 25 | 01 | 00 | 00 | 20 | 00 | 00 | ........%... .. |

図 5 クライアント改ざん前



| ADDRESS | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 0123456789ABCDEF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 4D | 5A | 90 | 00 | 03 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | FF | FF | 00 | 00 | MZ.............. |
| 00000010 | B8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ｸ.......@....... |
| 00000020 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ................ |
| 00000030 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 00 | 00 | 00 | ................ |
| 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.....ｩR.... |
| 00000090 | 00 | 00 | 00 | 00 | E0 | 00 | 02 | 01 | 0B | 01 | 08 | 00 | 00 | 06 | 01 | 00 | ................ |
| 000000A0 | 00 | 0E | 00 | 00 | 00 | 00 | 00 | 00 | EE | 25 | 01 | 00 | 00 | 20 | 00 | 00 | ........%... .. |

図 6 クライアント改ざん後

### 6.2.2 検証結果

手順2では，サーバは接続を許可してクライアントは作業計測を開始し，手順5では，サーバは接続を拒否してクライアントは作業計測を開始せずに終了した．

サーバが許可または拒否をしたことは認証者に報告される．接続が許可されている様子を図7に，拒否されている様子を図8に示す．

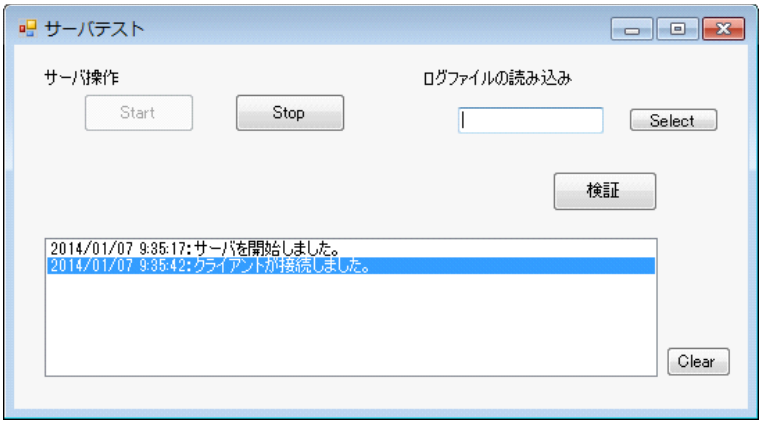



図 7 接続許可の報告

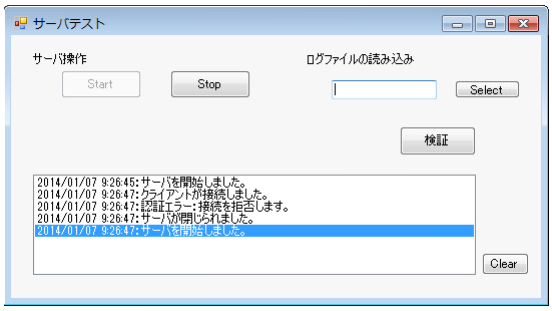



図 8 接続拒否の報告

サーバはクライアント認証の結果，クライアントに改ざんが検出されなければ接続を許可し，検出されれば接続を拒否する．改ざんされていないクライアントの接続を許可し，改ざんされているクライアントの接続を拒否しているため，本検証で行った改ざんは検出可能であるといえる．

本検証で行った改ざんは，クライアントの実行ファイルの内容を一部だけ変更するものであり，あらゆる改ざん内容を検証したわけではない．しかし，ハッシュ値は少しでも内容が異なれば値が変わるため，検証により改ざんした箇所以外を改ざんしても同様に検出可能であると考えられる．

## 6.3 対策2の検証

### 6.3.1 検証方法

以下に示す，手順1~5を順に行い，改ざんされていない作業履歴ファイルでは改ざんなしと報告されること，改ざんされている作業履歴ファイルでは改ざんありと報告されることをそれぞれ検証する．

1. 改ざんされていないクライアントとサーバをそれぞれ同じローカルサーバ内の別々のパソコンに導入する．
2. 導入直後のクライアントを実行しクライアント認証を行い，作業計測を開始する．
3. 作業を行い，図9の内容の作業履歴ファイルを用意する．
4. 用意した作業履歴ファイルを改ざんせずにそのままサーバへ送信し，改ざん検出を実施して，改ざんなしと報告されることを確認する．
5. 用意した作業履歴ファイルの14行目を図9の“テキスト閲覧・編集”から図10の“ファイル操作”に改ざんし，サーバによる改ざん検出を実施して，改ざんありと報告されることを確認する．

```
"テキスト閲覧・編集","2013年12月11日20時29分25秒","2013年12月11日20時29分47秒","3","0","0","C:\Windows\explorer.exe: hash"
```

図 9 作業履歴ファイル改ざん前

```
"ファイル操作","2013年12月11日20時29分25秒","2013年12月11日20時29分47秒","3","0","0","C:\Windows\explorer.exe: hash"
```

図 10 作業履歴ファイル改ざん後

### 6.3.2 検証結果

サーバは一つの作業（作業履歴の一行分）ごとに改ざんの有無を検証する．手順2では，サーバは14行目を含みすべての内容は改ざんなしと報告し，手順5では，サーバは14行目の内容のみ改ざんありと報告した．14行目の内容が改ざんなしと報告されている様子を図11に，改ざんありと報告されている様子を図12に示す．

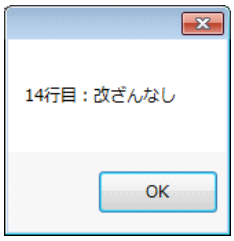



図 11 14行目の改ざんなし報告

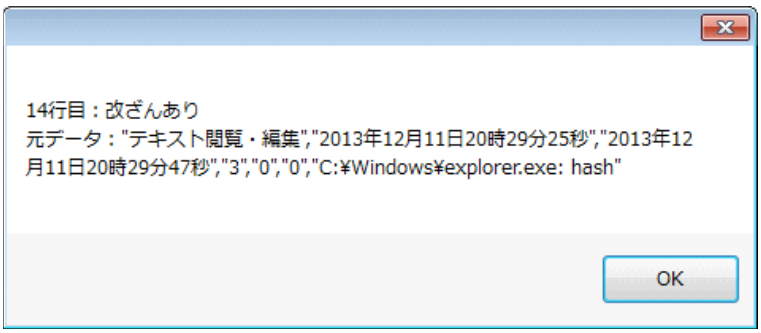



図 12 14行目の改ざんあり報告

サーバはログファイル認証の結果，作業履歴ファイルに改ざんが検出されなければ改ざんなし，検出されれば改ざんありと改ざん部分とともに報告する．改ざんされていない作業履歴ファイルでは改ざんなし，改ざんされている作業履歴

ファイルでは改ざんされている部分が14行目であることとともに改ざんありと報告しており，本検証で行った改ざんは検出可能であるといえる．

本検証で行った改ざんは，作業履歴ファイルの内容を一部だけ変更するものであり，あらゆる改ざん内容を検証したわけではない．しかし，ハッシュ値は少しでも内容が異なれば値が変わるため，検証により改ざんした箇所以外を改ざんしても同様に検出可能であると考えられる．

## 6.4 考察

実装したシステムによって，クライアントと作業履歴ファイルの内容の変化は検出可能であった．すなわち，要求通り作業履歴の改ざんは検出可能であるといえる．検証ではあらゆる改ざん内容を検証したわけではないが，ハッシュ値は少しでも内容が異なれば値が変わるため，検証により改ざんした箇所以外を改ざんしても同様に検出可能であると考えられる．

一方で，改ざんの方法によっては検出不可能になる問題点がある．理論上，クライアントの実行ファイルを解析して実行ファイルのハッシュ値を送信して認証を行っていることを知られると，改ざんしたクライアントに改ざん前クライアントのハッシュ値を送信させることでサーバに改ざんを検出されずに認証させることができる．

しかし実行ファイルの解析は非常に困難であり，クライアントがハッシュ値を送信して認証していることを解析できる可能性は低く，またワンタイムパスワードによる暗号化により送信データの内容は知られにくいため，システムを運用するうえでの改ざん防止能力はあると考えられる．

また，今回利用したTaskPitは動作のための別の実行ファイルや出力ファイルを持っており，同様に改ざん対象になりうるが，別の実行ファイルには対策1，出力ファイルには対策2の手法をそれぞれ適応することで，改ざんへの対策が可能になる．同様の理由で提案手法は実装先をTaskPitのみに縛らず，その他の作業計測システムにも実装可能である．

# 7 おわりに

本研究では開発組織が正しい開発プロセスで開発を行っていることを作業履歴の第三者評価から保証することを目的とし，改ざんのない作業履歴を評価可能な手法を提案した．提案手法ではハッシュ関数を利用したサーバ･クライアントシステムにより作業計測システムと作業履歴ファイルの改ざんを検出し，開発組織による改ざんを防ぐことで改ざんのない作業履歴を評価可能にする．提案手法の有効性を検証する実験では，システムを運用するうえでの作業計測システムと作業履歴ファイルの改ざん検出は可能であることが分かった．

今後の展望としては複数の開発組織の保証を目指し，一対多の通信が可能なサーバ･クライアントシステムの設計や提案手法が実際にシステムを運用するうえでの改ざん防止能力があるかの検証が挙げられる．改ざん防止能力がない場合，更なる改ざん対策の導入が必要になる．

# 謝辞

本論文の執筆をおよび研究をすすめるに当たり, 多くの方々に協力して頂きました. この場を借りてお礼を申し上げます. ありがとうございました. 指導教員である上野秀剛助教には, この1年間を通して研究に関する知識のご教授から, アドバイス, 論文のチェックまで多くの面でご指導頂きました. また, 私的な事柄についてもご相談を受けて下さり, アドバイスをいただくなど, 一年間支えて頂きました. ここに深謝の意を表明させていただきます. ありがとうございました.

査読教員である内田眞司准教授からは, 査読コメント, 卒研発表会の両方で, 鋭い質問, 今後の研究に対するアドバイスをいただきました. ここに深謝の意を表明させていただきます. ありがとうございました.

同じ上野研究室であった皆様には, 私の研究に対するアドバイスや発表での至らぬ点の指摘など, 多くの面で支えて頂きました. ここに深謝の意を表明させていただきます. ありがとうございました.

# 参考文献


[1] 情報処理推進機構:“ソフトウェアの品質説明力強化のための制度フレームワークに関する提案（中間報告）”, (2013).

[2] 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).

[3] 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).

[4] T. Matsumoto, S. Matsumoto, H. Uwano, A. Monden, Y. Miyamoto, S. Ujihara, N. Kohtake, and M. Katahira:“ Cost Effective IV&V Planning Activity Derived from Experiences on Jaxa's Spacecraft Projects, ” ESA Workshop on Avionics Data, Control and Software Systems (ADCSS) (2008).

[5] National Aeronautics and Space Administration :“ NASA Independent Verification and Validation Program, ”(2009).

[6] 宇宙航空研究開発機構:“ IV & Vガイドブック ”, 2013.

[7] 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).

[8] 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).

[9] 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).

[10] Process Dashboard： http://processdash.sourceforge.net/pspdash.html

[11] Task Coach Your friendly task manager：http://members.chello.nl/f.niessink/

[12] SlimTimer Time Tracking without the Timesheet：http://www.slimtimer.com/

[13] 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).

[14] 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.

[15] 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).

[16] 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).

[17] 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).