ソフトウェア開発企業における開発タスクの自動計測

Automatic Measurement of Software Development Tasks in Software Companies

**門田 暁人\***[[1]](#footnote-1)**上野 秀剛\*2　荒木 健史\*3　山田 欣吾\*4　松本 健一\*5**

**あらまし** 本稿では，ソフトウェア開発作業の自動計測の効果を明らかにし，その知見を蓄積・公開することを目的として，開発行動記録システムTaskPitを2つの開発組織に適用した結果について報告する．1つめの組織では，12名の開発者を6日間計測した結果，(a)リーダが開発作業に従事しすぎている，(b) メールによるコミュニケーションが少なすぎる人がいる，(c)短時間に顧客あての長文メールを何本も出していると推定される人がおり，メールの書き方に改善の余地がある可能性がある，といった改善の糸口を発見できた．2つめの組織では，13営業日にわたって1名の開発者の作業の計画値，実測値，申告値を記録した．その結果，(d)半分以上の日に計画外の作業が入った，(e)日数が経つにつれて申告値が正確になった，などが分かり，TaskPitを継続的に使うことで，開発者自らの仕事の管理に役立つことが示唆された．

1. はじめに

著者らはこれまでに，「計測に基づくプロセス改善」を実践し，世の中に広めていく一手段として，ソフトウェア開発行動記録システムTaskPit[8][11]を2008年に開発し，普及に努めてきた．TaskPitは，開発者もしくは開発チームが，日々の開発タスクに従事した時間や作業量を計測するツールであり，EclipseやVisual Studioを使った「プログラミング」，Wordを使った「文書作成」，ブラウザ上のWebメールを使った「メールの読み書き」等の，アプリケーションやウィンドウにひも付けられたタスクを自動計測できる．TaskPitをとりまくコミュニティは次第に広がり，有志による可視化ツールTaskVieweRの開発，クイックマニュアルの作成が行われ，2013年6月現在のTaskPit 1.0.0～1.0.3の総ダウンロード回数は1200を超えるに至った[11]．今後は，TaskPitユーザ向けに，実開発におけるTaskPitの適用事例およびその知見の蓄積が求められている．

本稿では，TaskPitを，2つのソフトウェア開発組織に適用した結果について報告する．1つめの組織では，9日間（6営業日）にわたって12名の開発者を計測した．12名の内訳は，開発メンバー7名，リーダ3名，顧客窓口2名である．計測データの分析は，調査対象部署の作業概要を把握している他部署のベテラン社員1名が行った．2つめの組織では，17日間（13営業日）にわたって1名の開発者を計測した．この計測では，TaskPitによるタスクの自動計測に加えて，1日ごとの各開発タスクの従事時間の計画値と申告値も記録してもらい，実測値との違いを明らかにすることで，タスクの自動計測の効果を分析した．

以降では，まず，2章において，改良したTaskPitの目的，機能，システム構成などを紹介し，3，4章で2つの組織における計測結果と分析を述べる．5章はまとめである．

1. TaskPit
   1. 開発の動機と特徴

Tom DeMarcoの「計測できない物は制御できない」という格言に代表されるように，開発プロセスの制御や改善には計測が必須である[1]．そのために，従来，様々なプロダクト・プロセスメトリクスが提案され，開発現場に適用されてきた[3]．多くの現場では，ソフトウェア規模，開発工数，バグ数をはじめとする多数のメトリクスが計測され，プロジェクト管理や品質保証に用いられてきた[4][9]．

一方，開発現場で生じる問題の多くは人的要因に起因する[2]ものであるから，プロダクトやプロセスを計測するよりも，開発の主体である人間やその作業を計測し，改善につなげることが自然であると考えられる．その一つの手段であるPersonal Software Process (PSP) / Team Software Process (TSP)は，開発者および開発チームが日々のタスクに関する情報を記録し，自らのプロセス改善を行う方法として広く知られている[5][6]．ただし，PSP/TSPのデータ計測は人手で行う必要があることから，導入の敷居は高く，広く普及するには至っていない．

開発作業を自動計測するツールとしては，Ginger 2[12]，HackyStat[7]，PROM[10]などが提案されている．これらのツールでは，開発者のキー入力やマウスの動作に加えて，Eclipse，Microsoft Office，Emacs等のアプリケーションの動作履歴を自動計測できる．ただし，計測対象が特定のバージョンのOS，アプリケーションに限られるため，多くの開発現場に導入することは難しい．また，計測の粒度が細かすぎるために解釈が容易でなく，プロセス改善につなげるまでの距離があった．

そこで，著者らは，「開発現場ですぐに使え，計測結果も解釈しやすい」ことを実現する自動計測ツールとして，TaskPitを開発し，改良を行ってきた[11]．TaskPitは，(1)どのタスクにどの程度の時間を費やしているか，(2)どの程度の作業量を各タスクに費やしているか，(3)各タスクの成果物の量，を個人またはチーム単位で自動計測できる．

計測にあたっては，(1)各タスクは，それぞれ異なるアプリケーションやウィンドウ上での作業であると捉える．各アプリケーションやウィンドウ上での作業時間（ユーザが何らかの入力を与えていた期間）を記録することで，各タスクに費やした時間を計測する．また，(2)各タスクの作業量は，各アプリケーションやウィンドウに対するキーストロークやマウスの操作量（回数）として記録する．さらに，(3)各タスクの成果物は，特定のディレクトリ下にファイルとして出力されると捉える．ファイルサイズの増減を一定時間ごとに調べることで，各タスクの成果物の量の推移が計測できる．

従来の自動計測システムと異なり，TaskPitでは，個々のアプリケーションやウィンドウに対する入力（コマンドやボタンの選択など）やアプリケーションの動作（画面出力やモードの切り替わりなど）を記録しない．このことは，タスクの内部に立ち入った計測を行わないことを意味するが，PSP/TSPでは「どのタスクを実施しているか」が重要であり，タスクの中身は必ずしも重要でない．個々のアプリケーションの内部動作に立ち入らないことで，特定のアプリケーションに限定せず，履歴の計測が可能となった．

* 1. タスクと成果物の定義と計測

異なるタスクであっても，同一のアプリケーションを用いる場合があるため，TaskPitでは，アプリケーション実行時のウィンドウ名（に含まれる文字列）によってタスクを区別する．拡張BNFによるタスクと成果物の定義は下記の通りである．

〈タスク〉 ::= 〈アプリケーション〉｛ |〈アプリケーション〉｝  
〈アプリケーション〉 ::= 〈実行ファイル名〉［〈ウィンドウ名〉］  
〈成果物〉::= 〈ディレクトリ〉〈拡張子〉｛〈拡張子〉｝

タスクは，実行中の1つ以上のアプリケーションの組として定義され，アプリケーションは，実行ファイル名とウィンドウ名の組として定義される．また，成果物は，1つのディレクトリと1つ以上のファイル拡張子の組として定義される．

TaskPitでは，計算機のユーザが，各アプリケーションの使用を開始／終了した時刻を記録する．ここで「アプリケーションを使用している」とは，ウィンドウにフォーカスが当たっている状態を指す．ただし，フォーカスされている場合であっても，一定時間（例えば3分間）計算機に対する入力（マウス，キーボード）が行われなかった時点で，いずれのアプリケーションも使用していないとみなす．

* 1. システム構成

図1に示すように，TaskPitは計測部とそのバックエンドとなるデータベース，可視化部，設定ファイル，ログファイル，作業日報ファイルから構成される．設定ファイルでは，タスクと成果物の定義，ログファイルの出力時間間隔などの指定を行う．また，計測部は，図2に示すように，「タスク」タブでは，各タスクの累積の実行時間，打鍵数，クリック数が表示される．「成果物」タブでは，各成果物のファイルサイズとファイル数の増減が表示される．計測結果は，一定時間（例えば10分間隔）でログファイルに出力される．ログファイルの出力先を共有フォルダとすることで，多数の開発者の計測結果を容易に集計できる．可視化部は，設定ファイルとログファイルを入力とし，指定された期間の計測結果を様々な側面からグラフ化する．また，1日毎の計測結果の概要を作業日報として（Excelの.xls形式で）出力できる．



図1. TaskPitのシステム構成



図2. 計測部のGUI

表1　ケーススタディ1における計測対象の開発者

|  |  |  |
| --- | --- | --- |
| 役割 | メンバー | 備考 |
| リーダ | L1, L2, L3 |  |
| プロジェクトAの開発作業（開発A） | A1, A2, A3 | A1は4,5年目．A2，A3は1年目 |
| プロジェクトBの開発作業（開発B） | B1, B2, B3, B4 | B1は他部署から異動してきてから1ヶ月 |
| 顧客窓口 | CS1, CS2 | 二人とも時短勤務 |

* 1. 実装

TaskPit 1.0.xは，Windows XP/Vista/7の.Net Framework上で動作し，データベースにはSQLiteを用いている．タスクの特定は，Windows APIを用いて行っている．具体的には，GetForegroundWindowでアクティブなウィンドウのハンドルを取得し，GetWindowTextによりウィンドウタイトル取得している．また，System.Diagnostics.ProcessクラスのGetProcessByIdメソッドによってプロセスを取得し，実行ファイル名を得ている．

同様に，キーストロークとマウスクリックの計測は，SetWindowsHookExにおいてキーボードイベントとマウスイベントをフックするメソッドを登録して計測している．また，成果物の計測は，一定時間ごとにSystem.IO.DirectoryクラスのGetFilesメソッドにより，指定されたディレクトリ内のファイルを走査し，ファイル数とファイルサイズを計測している．

1. ケースタディ１：PMOの立場からの計測・分析
   1. 計測の目的

１つめの組織におけるケーススタディでは，ソフトウェア開発部署における問題の糸口を発見し，改善につなげることを想定している．計測結果の分析を行うのは，プロジェクトマネジメントオフィス（PMO）としての機能を持つ開発管理・支援部門のベテラン社員1名であり，全社的なプロジェクト管理手法の標準化，品質管理，人材育成の観点から開発支援またはコンサルティングを行うことを想定している．分析者は，各作業者の役割を把握しているが，作業内容までは把握していない．

* 1. 計測対象と期間

計測対象は，ソフトウェア開発部署の12名である．表1に示す通り，部署における各人の役割は，リーダ3名，開発（Aプロジェクト）3名，開発（Bプロジェクト）4名，顧客窓口2名である．Aプロジェクトは納期が近いため多忙であり，Bプロジェクトは納期が近くない．メンバーのうちB1は，他部署から1か月前に移動してきたため，必ずしも業務に慣れていない．また，顧客窓口の2名（CS1, CS2）は時短勤務である．

この部署ではプロジェクト管理ツールとしてブラウザ上でTracを使っており，Trac上での資料管理とコミュニケーションを行うことが奨励されている．

ブラウザ = iexplore.exe|firefox.exe|Safari.exe|chrome.exe

メール = thunderbird.exe|Outlook.exe|iexplore.exe:gmail

ファイル操作 = explorer.exe

エディタ　= sakura.exe|noediter2.exe

ＤＢ操作　= SqlWb.exe

エクセル = EXCEL.EXE|soffice.exe

ワード・パワポ = WINWORD.EXE|POWERPNT.EXE

文書閲覧　= AcroRd32.exe

テスト　= mstsc.exe|Beyond32.exe|DF.exe|reverse.exe|reverseserver.exe|perfmon.exe|….

プログラミング・デバッグ = eclipse.exe|devenv.exe|VPC.exe|VMWindows.exe|hh.exe

図3. 設定ファイル（抜粋）

データ計測においては，開発作業に外乱を与えないために，GUIを表示しないように改造したサイレント版TaskPit 1.0.1を用いた．また，計測に先立って，この部署で使っているアプリケーションについてヒヤリングを行い，タスクとアプリケーションのひも付けを行った．計測は9日間（6営業日）にわたって行った．

* 1. 設定ファイル

計測に用いた設定ファイルの一部を図3に示す．図に示されるように，1つのタスクに対して複数のアプリケーションが割り当てられている．

* 1. 計測結果と分析

図4は，各タスクの従事時間，全タスクの合計時間，勤務時間をそれぞれ1日あたりの平均値で示したものである．勤務時間は，TaskPitの起動・終了時刻から，1日の勤務時間を推定したものである．得られた主な結果は次の通りである．

* 図4に示されるように，いずれの開発者においても登録外の作業があった．特に，A3, B2, B4の3名については，登録外のタスクが100分を超えている．その原因の一つとして，この部署では，予め決められたアプリケーションのみPCにインストールできることとなっていたが，指定外のアプリケーション（ブラウザソフト）をインストールしていた場合があった．また，この部署では，勤怠管理システムを用いていたが，設定ファイルに登録されていなかった．また，TaskPit実装上の問題として，仮想OS上でのテストやプログラミング・デバッグについて，計測できていないことが明らかとなった．（TaskPit 1.0.3では，設定ファイルに登録していないアプリケーションについても計測できるよう拡張を行った．）
* 納期を間近に控えたA1, A2, A3の勤務時間，および， PC上での作業時間は長くなっている．一方，B1, B2, B3, B4は仕事が順調に進捗しており、勤務時間はプロジェックトAに比べ短くなっている．TaskPitにより，プロジェクトの忙しさをある程度把握できるといえる．
* 勤務時間・PC上の作業時間ともに，最も多かったのはA1である．従事時間が多い順にプログラミング・デバッグ（184分），エディタ（106分），テスト（94分）であり，コーディングを中心に作業を行っていることが伺える．一方，A2は，同じプロジェクトに属するが，プログラミング・デバッグは0分であり（登録外を入れても76分），テスト（156分）を担当していることが伺える．TaskPitにより，各開発者の作業内容を把握できるといえる．



図4　各タスクの平均従事時間（1日あたり）



図5. 勤務時間とPC上の作業時間の関係

* 勤務時間とPC上の作業時間との関係を図5に示す．図5より，リーダ3名は，開発者や窓口の者よりも勤務時間に対するPC上の作業時間が少ないことが分かる．これは，作業指示，指導，管理等の業務を行っているためである．ただし，リーダL2は他のリーダと比べて，勤務時間に対するPC上の作業時間がやや大きくなっている．タスクの内訳をみると，リーダL2はテストに平均71分従事し，また，テストケース管理に関すると思われるエクセル操作に103分も従事している．リーダとしては開発作業に従事しすぎているといえる．



図6　メールとプログラミングの従事時間と打鍵数（1日あたり）

* 顧客窓口の2名（CS1, CS2）は，他者よりもメールの従事時間が多く，ブラウザの使用時間も多い．メールによって顧客とのやり取りを行っていることと，ブラウザ上でTracによる資料管理とコミュニケーションを行っていることが伺える．（今回の計測では，ブラウザ上のタスクの違いを設定ファイルに反映できていないという問題も明らかとなった．）
* メールタスクに着目すると，L2，A3，B1, B2, B3, B4はいずれも15分未満であった．この部署では，コミュニケーション手段として電話よりもメールを使用することが奨励されている（記録を残すため）．このことから，特にリーダL2は，メールの使用時間が少なすぎるといえる．その後の調査で，L2は電話を多用していることが分かった．
* メールタスク，プログラミング・デバッグタスクの従事時間と打鍵数を図6に示す．メールタスクについて1分あたりの打鍵数を比較すると，CS1, CS2の値が高い．この2名は打鍵数自体も多く，メールの本文が長い傾向があることが伺える．顧客とのメールのやりとりでは，なるべく簡潔に要点を押さえて書くことが望ましく，また，記述内容をよく吟味・推敲することが求められるため，メールの書き方に改善の余地がある可能性がある．
* 図6において，プログラミング・デバッグのタスクと打鍵数に着目すると，A1は，打鍵数，1分あたりの打鍵数ともに多く，品質を別にすれば生産性の高い開発者であるといえる．

1. ケーススタディ２：開発者自らの改善活動を想定した計測
   1. 計測の目的と方法

2つめの組織では，開発者自らが計測・分析を行い，状況把握，問題の発見，改善につなげることを想定する．本ケーススタディでは，1日ごとの各開発タスクの従事時間の計画値，実測値，申告値を計測し，それらの違いを明らかにする．計画値は1日の初めに開発者が設定する値である．実測値はTaskPitによる得られる値である．また，申告値は，1日の終わりに，開発者が（TaskPitによる計測結果を見る前に）各タスクに従事した時間を自己申告する値である．

仮に，計画値，実測値，申告値が一致していれば，TaskPitを使うまでもなく，開発者は自らの計画に沿ってタスクを実行できており，かつ，状況把握もできていることになる．しかし，実測値と申告値に大きな差がある場合，タスクの実施時間を把握できていないことになる．特に，計画値と実測値に大きな差がある場合，状況把握できていないことは大きな問題となる．

* 1. 計測対象と期間

計測対象は，ケーススタディ1とは異なる組織に属する，Web系プログラミングに従事する開発者1名である．計測期間は17日間（13営業日）である．計測に先立って，TaskPitの設定ファイルの調整と予備計測を3日間行った．また，毎日の仕事の終わりに，計画値，実測値，申告値を参照してもらい，コメント（感想）を記述してもらった．

* 1. 計測結果と分析

計画値の推移を図7に，実測値の推移を図8に示す．各グラフの横軸は日数，縦軸は時間数である．11日目を除いて実測値は計画値よりも小さくなりがちであり，計画通りの作業時間を費やせていないことが分かった．開発者のコメントより，この原因として，パソコンを用いない作業（電話対応）に時間を取られたケースが4日，その他の想定外の作業が入ったケースが3日あった．このように計画外の作業は入ることはむしろ一般的であるため，全作業の5～10%程度を予備工数として計画に含めておくべきであることが，TaskPitにより明らかになった．なお，11日目に実測値が計画値より上回った原因は，計画ミス（本来行うべき作業があったのを忘れていた）であった．

図9に，実測値から申告値を引いた値の推移を示す．図に示されるように，TaskPitを使い始めた当初は過大見積もりしていたが，次第に実測値に近づくようになった．このことから，このケーススタディでは，TaskPitを継続して用いることで，開発者は自らの作業量を正しく把握できるようになった，その裏付けとして，計測5日目に「作業時に実行中タスクを意識するようになり，計画時間および申告時間は実測時間にかなり近づいてきた」という開発者のコメントがあった．また，7日目に「差し込みの要件がない場合は，計画時間，申告時間，実測時間の開きが少なくなってきた」というコメントがあった．このことから，TaskPitを用いることで，開発者が計画時間と実測時間を意識して開発を行えるようになり，自らの仕事の管理に役立つことが期待される．

1. まとめ

本稿では，ソフトウェア開発作業の自動計測の効果を明らかにし，その知見を蓄積・公開することを目的として，開発行動記録システムTaskPitを2つの開発組織に適用した．1つめの組織では，6営業日にわたって12名の開発者を計測した．その結果，PMOの立場から，プロジェクトの忙しさや各開発者の作業内容を把握できることが分かった．また，(a)リーダが開発作業に従事しすぎている，(b) メールによるコミュニケーションが少なすぎる人がいる，(c)短時間に顧客あての長文メールを何本も出していると推定される人がおり，メールの書き方に改善の余地がある可能性がある，といった改善の糸口を発見できた．



図7　各タスクの計画値

2つめの組織では，13営業日にわたって1名の開発者の作業の計画値，実測値，申告値を記録した．その結果，(d)半分以上の日に計画外の作業が入った，(e)日数が経つにつれて申告値が正確になった，などが分かり，TaskPitを継続的に使うことで，開発者自らの仕事の管理に役立つことが示唆された．



図8　各タスクの実測値

以上の適用事例および知見は，TaskPitによるプロセス計測・改善を検討する開発組織に役立つと期待される．著者らは，今後も引き続き，TaskPitの適用事例を増やし，その知見を公開していく予定である．



図9　申告値と実測値の差の推移

参考文献

1. T. DeMarco, "Controlling Software Projects: Management, Measurement & Estimation," Yourdon Press, New York, USA, 1982.
2. 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター, "ITプロジェクトの「見える化」中流工程編," 日経BP, 2008.
3. R. B. Grady, "Practical software metrics for project management and process improvement," Prentice Hall, 1992.
4. 誉田 直美, "ソフトウェア品質会計," 日科技連, 2010.
5. W. S. Humphrey, "A discipline for software engineering," Addison-Wesley, 1995.
6. W. S. Humphrey, "パーソナルソフトウェアプロセス入門," 共立出版, May 2001.
7. P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore and J. Miglani, S. Zhen and W. E. J. Doane,"Beyond the Personal Software Process,“metrics collection and analysis for the differently disciplined," Proc. 25th International Conference on Software Engineering, pp.641-646, 2003.
8. 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XV, 日本ソフトウェア科学会 FOSE2008, pp.123-128, Nov. 2008.
9. L. H. Putnum and W. Myers, "Five Core Metrics – The Intelligence Behind Successful Software Management," Dorset House, 2003.
10. A. Sillitti, A. Janes, G. Succi and T. Vernazza,"Collecting, Integrating and analyzing software metrics and personal software process data," Proc. 29th EUROMICRO Conference, pp.336-342, 2003.
11. ソフトウェア開発行動記録システムTaskPit, <http://taskpit.jpn.org/>
12. K. Torii, K. Matsumoto, K. Nakakoji, Y. Takada, S. Takada, K. Shima, "Ginger2: an environment for CAESE (computer-aided empirical software engineering)," IEEE Transactions on Software Engineering, Vol. 25, No. 4, pp. 474-492, Aug. 1999.

1. \*1 Akito Monden, 奈良先端科学技術大学院大学

   \*2 Hidetake Uwano, 奈良工業高等専門学校

   \*3 Kenji Araki, 株式会社アクセス

   \*4 Kingo Yamada, 株式会社ファインバス

   \*5 Kenichi Matsumoto, 奈良先端科学技術大学院大学 [↑](#footnote-ref-1)