Vol.2015-SE-190 No.8
2015/12/15

ソフトウェア開発プロジェクトデータセットからの典型的なプロジェクトの抽出

川下正宏[†1]　門田暁人[†1]　畑秀明[†2]

概要：過去のソフトウェア開発経験を将来の開発に生かすことを目的として，多数の開発プロジェクトの実績データを含むデータセットから，ごく少数の典型的な開発プロジェクトのデータのみを抽出する方法を提案する．提案方法では，開発工数の見積もり（予測）時に過去の典型的なプロジェクトを参照する場面を想定し，予測に必要な最小限のプロジェクトのデータを抽出する．

キーワード：ソフトウェア開発工数予測，データ抽出，データマイニング

Extracting Typical Projects in a Software Project Data Set

Masahiro KAWASHIMO[†1]　Akito MONDEN[†1]
Hideaki HATA[†2]

Abstract: To learn from past software development projects, this paper proposes a method to extract a small set of typical software projects from a given (possibly large) software project data set. The method assumes a situation where a project manager wants to refer to a minimum set of past projects for the purpose of software development effort estimation.

Keywords: Software development effort estimation, data extraction, data mining

1. はじめに

ソフトウェアやその開発プロセスから得られる定量データ（ソフトウェア開発プロジェクトデータ）は，プロジェクト管理，品質保証，開発工数予測，生産性の向上などのために広く活用されている 4)16)．例えば，日本の受託ソフトウェア開発を対象とした調査において，開発工数予測の基礎データとなるファンクションポイント，ソースコード行数等の規模尺度を計測している企業は 85%に達すると報告されている 15)．

ただし，一般に，計測・蓄積されたプロジェクトデータには，ノイズとなるデータ（外れ値，異常値，矛盾のある値など）が多数含まれており，そのためにデータが活用されないままとなるケースも少なくない．例えば，Promise Data Repository 12)には開発規模や工数などを記録した多数のプロジェクトデータが公開されているが，その一つである China データセットは，プロジェクト件数は多い（499 件）ものの，データ品質が低く，精度の良い工数予測モデルの構築には役立たないことが知られている 7)．また，比較的データ品質が高いとされている Kemerer データセット 6)7)においても，互いに矛盾するプロジェクトの組が少なからず存在することが知られている 13)．表 1 は Kemerer データセットからの抜粋であるが，9 番目と 11 番目のプロジェクトは，プロジェクトの特性（開発規模，開発期間，言語，ハードウェア）が極めて類似しているにも関わらず，開発工数が倍以上も異なっている．

表 1. ソフトウェア開発プロジェクトデータセットの例
（Kemerer データセット 6)より抜粋）

ID	Language	Hardware	Duration	KSLOC	AdjFP	EffortMM
1	1	1	17	253.6	1217.1	287.0
2	1	2	7	40.5	507.3	82.5
3	1	3	15	450.0	2306.8	1107.3
4	1	1	18	214.4	788.5	86.9
5	1	2	13	449.9	1337.6	336.3
6	1	4	5	50.0	421.3	84.0
7	2	4	5	43.0	99.9	23.2
8	1	2	11	200.0	993.0	130.3
9	1	1	14	289.0	1592.9	116.0
10	1	1	5	39.0	240.0	72.0
11	1	1	13	254.2	1611.0	258.7

Very similar　　Very different

データセット中のこのようなノイズは，プロジェクト固有の事情（プロジェクトの失敗など），計測誤り，計測者に依存する計測のばらつきなどが考えられ，避けることが一般にできない．そのため，ノイズの存在を前提として，データを現在進行中もしくは将来の開発に役立てることが大きな課題となっている 13)．

本稿では，このような玉石混交のデータから「玉」となるデータだけを取り出す方法について検討する．つまり，多数の開発プロジェクトの実績データを含むデータセット

†1 岡山大学
Okayama University
†2 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

から，ごく少数の参考にすべき（典型的な）プロジェクトのデータのみを抽出することを考える．具体的には，開発工数の見積もり（予測）時に過去の典型的なプロジェクトを参照する場面を想定し，予測に必要な最小限のプロジェクトのデータを抽出する方法を提案する．

また，提案方法の評価実験として，71 件のプロジェクトのデータを含む Desharnais データセット 4)を対象として，典型的なプロジェクトの抽出を行い，得られたプロジェクトについての分析を行う．

以降，2 章では関連研究について述べ，3 章では，典型的なプロジェクトの抽出方法を提案する．4 章では，提案方法の評価実験について述べる．5 章は，まとめと今後の課題である．

2. 関連研究

2.1 データ品質の重要性

ソフトウェア開発における計画立案，プロジェクト管理，品質保証，プロセス改善等の活動において，その根拠となるソフトウェア開発データの品質は極めて重要である．データに不備がある場合，有益な情報が得られない，もしくは，誤った結論を得る恐れがあるためである．このために，従来，質の高いソフトウェア開発データを計測・蓄積するためのノウハウやガイドラインが提案されてきた 1)．一般に，品質の高いデータを得るためには，データ計測から活用に至るまでのライフサイクル全体を通した品質向上活動が必要であり，そのために，計測値の精度，再現性，完全性，一貫性などに着目して改善を続けていくことが奨励されている 2)5)．

一方，ソフトウェア工学の研究者にとっても，研究対象となるデータの品質は重要である．問題のあるデータから導かれた結論は，信頼できないためである．ところが，Liebchen と Shepperd 9)が実証的ソフトウェア工学研究の systematic review を行った結果，論文中で扱われているデータの品質について明確な評価や考慮を行っている論文は（何百もの論文のうちの）わずか 23 件であったことを示している．Liebchen と Shepperd は，研究者はデータ品質にもっと着目すべきであり，データ品質を評価・改善に関する研究が必要であると結論付けている．

これらのデータ品質を高めようというアプローチに対し，本稿は，たとえ品質の低いデータセットであっても，利用価値のある少数のケースを取り出すことで活用を目指そうとしている点が異なっている．

2.2 外れ値除去手法

本稿とは逆に，データセット中のノイズ（外れ値や異常値）を除去することで，その有効活用を目指す取り組みが行われてきた 10)．

一般に，外れ値の定義は様々である．最も単純な方法は，正規分布を取る値を対象として，平均値から±2σ，または±3σを超える値を外れ値とみなす方法である．ただし，ソフトウェア開発データにおいては，外れ値が必ずしも（削除すべき）異常値であるとは限らない．例えば，規模が大きなプロジェクトは数が少ないことが多いが，だからといって大規模プロジェクトを取り除いてしまうと，大規模プロジェクトに対するモデル化が困難となる．

そこで，ソフトウェア開発プロジェクトデータに適した外れ値除去方法が提案されてきた．例えば，Cook の距離を用いる方法 11)，Least Trimmed Squares (LTS)を用いる方法 14)，k-means 法を用いる方法 14)，Mantel 相関を用いる方法 7)などである．これらの方法では，外れ値を除去した後に残ったデータを用いてソフトウェア開発工数予測モデルを構築することで，より精度の高いモデルが構築されることが示されている．

これらの方法は，少数の外れ値を除去するための方法であるが，外れ値を極限まで除去すれば典型的なプロジェクトを残すことができる可能性がある．ただし，筆者らは，これらの方法は，典型的なプロジェクトのみを残すという用途には必ずしも向いていないと考える．例えば，Cook の距離を用いる方法は，重回帰モデルを対象とした方法であり，重回帰モデルの偏回帰係数に大きな影響を与えるプロジェクトを外れ値として除去できるが，外れ値を除去しすぎると，どのプロジェクトを除去しても偏回帰係数に大きな影響を与えることとなり，それ以上プロジェクト数を削除できなくなる．

一方，提案方法は，外れ値を除去していくという考え方ではなく，工数予測に必要な最小限のプロジェクトのデータを抽出する，目的に即した方法である．

3. 提案方法

本稿では，与えられたプロジェクトデータセットから，予測に必要な最小限のプロジェクトのデータを抽出する 2 つの方法を提案する．一つは，予測対象のプロジェクトデータセットが存在することを仮定した方法（以下 fit-test 法）であり，もう一つは，予測対象のプロジェクトデータセットの存在を仮定しない方法（以下 fit-fit 法）である．いずれの方法においても，ソフトウェア開発工数予測の方法として，線形重回帰モデルを用いる．

Fit-test 法では，予測モデル構築に用いるデータセット（fit dataset）からプロジェクト 1 件を取り除く全ての組み合わせで重回帰分析モデルを作成し，予測モデル評価用のデータセット（test dataset）に対する予測を行い，除去時に実際の目的変数との相対誤差平均が最も下がったプロジェクトを削除する．これを残りのプロジェクト数が 1 件になるまで繰り返し，十分な予測精度が得られている時点のプロジ

ェクト集合を採用する．例えば，プロジェクト件数が残り 5 件より少なくなると極端に予測精度が低下する場合, 5 件の時点におけるプロジェクト集合を，予測に必要な最小限のプロジェクト集合とする．ただし，この方法では，最適解に到達できない可能性がある．そこで，プロジェクト集合が 3 件の場合と，4 件の場合に限り，全探索により最も相対誤差が小さくなる組み合わせを求め，先に求めた最小限のプロジェクト集合における相対誤差よりも小さいならば，こちらを採用する．なお．5 件以上については，全探索における計算量が爆発するため，本稿では実施しない．

また，fit-fit 法では，test dataset の代わりに fit dataset を用いる．つまり，dataset からプロジェクト 1 件を取り除く全てのすべての組み合わせで重回帰分析モデルを作成し，fit dataset を予測し，除去時に相対誤差平均が最も下がったプロジェクトを削除する.これを繰り返し，十分な予測精度が得られている時点のプロジェクト集合を採用する．本方法においても，プロジェクト集合が 3 件の場合と，4 件の場合に限り，全探索により最も相対誤差が小さくなる組み合わせを求める．

4. 評価実験

4.1 実験の題材

Promise Data Repository 12)に収録されているプロジェクトデータセットのうち，もっともよく使われている Desharnais データセット 3)を用いた．Desharnais データセットは，カナダのあるソフトウェア企業の開発実績データであり，無償で一般公開されているため，追実験が可能である．

過去の実績データを用いて将来のプロジェクトの工数予測を行うことを想定し, モデル構築用の fit dataset として 1986 年以前のプロジェクト 60 件を用い，モデル評価用の test dataset として 1987 年以降の 21 件を用いた．

目的変数は開発工数である．説明変数は，本データセットにおいて開発工数に大きく寄与することが知られているファンクションポイント（未調整），及び，開発言語を用いた．開発言語については，Desharnais データセットにおいてカテゴリが 3 であった場合に 1 を取り，そうでない場合に 0 を取る 2 値変数として用いた．

4.2 結果

図 1 に，fit-test 法の結果を示す．図中，横軸は除去したプロジェクト件数であり，縦軸は test dataset に対して開発工数予測を行った場合の相対誤差平均である．全プロジェクトのデータを用いた場合の相対誤差平均は 0.295 であり，そこからプロジェクト件数を減らしていくと，誤差はなだらかに下がり，削除件数が 10 件以降ではほぼ横ばいになった．残り 12 件から 11 件に減らす時点で相対誤差平均が

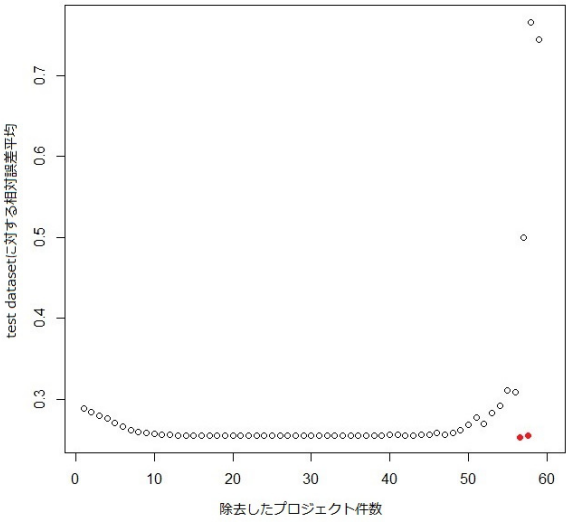

図 1. Fit-test 法における相対誤差平均の推移

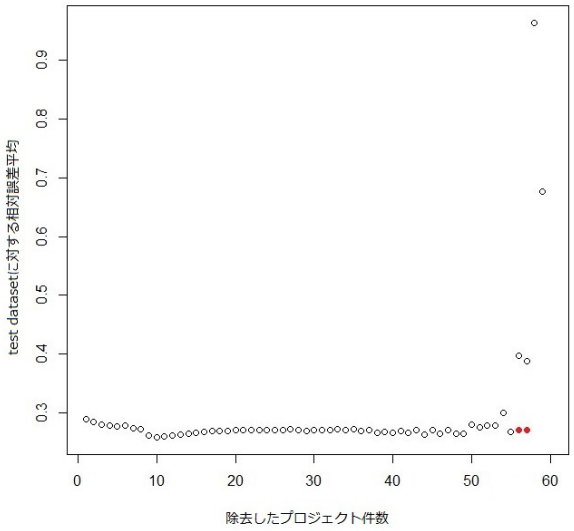

図 2. Fit-fit 法における相対誤差平均の推移

大きくなるため，この残り 12 件が，典型的なプロジェクトの集合の一つといえる．また，残り 5 件から 4 件に減らす時点で相対誤差が大きく増えるため，典型的なプロジェクトはこの 5 件であるという捉え方もできる．一方，全探索した場合の相対誤差は 3 件で 0.250, 4 点で 0.249 となった．図中，全探索の結果を●で示している．結果を見ると，全探索における 3 件と 4 件では相対誤差平均にほとんど差がなく，また，全探索における 3 件の方が 5 件の時点よりも相対誤差平均が小さくなっているため，典型的なプロジェクト集合としては，全探索における 3 件を採用する．

同様に，図 2 に，fit-fit 法の結果を示す．縦軸は，fit dataset に対する相対誤差平均ではなく, 現実の工数予測を想定し，test dataset 予測時の相対誤差平均を示している（ただし，

プロジェクト削除の過程では fit dataset に対する予測を行っている点に注意されたい). 全プロジェクトのデータを用いた場合の相対誤差平均は 0.295 であり，そこからプロジェクト件数を減らしていくと，誤差はなだらかに下がり，削除件数が 10 件以降でほぼ横ばいになった．残り 11 件から 10 件に減らす時点で相対誤差が増えるので,この残り 11 件が, 典型的なプロジェクトの集合の一つといえる. また，残り 5 件から 4 件に減らす時点で相対誤差平均が大きく増えるため，典型的なプロジェクトはこの 5 件であるという捉え方もできる．一方，全探索した場合の相対誤差は 3 件で 0.270，4 点で 0.271 となった結果を見ると，全探索における 3 件と 4 件では相対誤差平均にほとんど差がなく，また，全探索における 3 件の方が 5 件の時点よりも相対誤差平均が小さくなっているため，典型的なプロジェクト集合としては，全探索における 3 件を採用する．

4.3 考察

表 2 に，プロジェクト件数を 1 件から 11 件まで削除していった際の fit-test 法の相対誤差平均と fit-fit 法の相対誤差の比較結果を示す．表中，最左カラムは削除後に残ったプロジェクト件数である. 両社の比較においては, fit-test 法の相対誤差のほうが全体を通してやや低くなったが，その差はわずかである．また，4.2 節で述べたように，全探索における 3 件の相対誤差平均を比較すると，fit-test 法では 0.250 であったのに対し，fit-fit 法では 0.270 であった．Fit-fit 法がやや劣るものの，予測対象のプロジェクトデータセットの存在を想定しない fit-fit 法であっても，予測に有用な典型的なプロジェクトを選定できているといえる．

表 3 と表 4 に, fit-test 法と fit-fit 法のそれぞれにおいて，プロジェクト件数を 1 件削除した時点から，10 件削除された時点に至るまでの，削除されたプロジェクトの内訳を示す(表の上ほどより初期に削除されたプロジェクトを示す). 表中，ProjectNumber はプロジェクト ID，FunctionPoints は未調整ファンクションポイント，Lang3 は開発言語がカテゴリ 3 であることを示す 2 値変数，Effort は開発工数を示す．両方法で共通に除去されたのは，プロジェクト ID が 45，29，38，31，24 の 5 件であった．ある程度共通のプロジェクトが除去されているといえる．また，いずれの方法においても，初期に削除されたプロジェクトに共通している性質として，開発工数が大きいものが多い（開発工数平均値は 5233 であった）ことが分かった．

表 5 に，fit-test 法の全探索により抽出された 3 件プロジェクトの内訳を示す．また，それら 3 件をレーダーチャートで表したものを図 3 に示す．レーダーチャートでは，各変数値の最小値を結んだ線と，最大値を結んだ線をリファレンス用に示している．表と図から分かるように，3 つのプロジェクトは特徴が大きく異なっている．ただし，ファンクションポイントと開発工数については，3 つのプロジェクトのいずれについても最大値から離れている点は興味深い．このことは，規模や開発工数が大きなプロジェクトは，特異点となりやすく，将来の開発の参考になりにくいことがうかがえる．また，表 5 より，プロジェクト ID が 15 と 30 とでは, FunctionPoints の値が類似している一方で，Lang3 が 0 と 1 という違いがあり，そのため Effort に 3 倍

表 2. fit-fit と fit-test の相対誤差の差

	fitfit	fittest	fiffit−fittest
59	0.288	0.288	0.000
58	0.284	0.284	0.000
57	0.280	0.280	0.000
56	0.278	0.276	0.002
55	0.277	0.271	0.006
54	0.278	0.267	0.011
53	0.274	0.262	0.013
52	0.272	0.260	0.012
51	0.261	0.258	0.003
50	0.258	0.257	0.001
49	0.259	0.256	0.003

表 3. 削除されたプロジェクト(fit-test 法)

Project Number	Funtion Points	Lang3	Effort
45	514	0	19894
29	342	0	14434
38	587	0	14973
31	1127	0	23940
24	311	0	12824
58	655	0	13860
11	100	0	805
48	432	0	8232
59	588	0	9520
37	92	0	840

表 4. 削除されたプロジェクト(fit-fit 法)

Project Number	Funtion Points	Lang3	Effort
45	514	0	19894
29	342	0	14434
24	311	0	12824
44	256	0	9135
5	258	0	9051
13	260	0	7854
38	587	0	14973
60	645	1	5880
8	286	1	595
31	1127	0	23940

表 5. Fit-test 法において全探索により選択された 3 件のプロジェクト

Project Number	Funtion Points	Lang3	Effort
15	344	0	4977
30	398	1	1400
57	709	0	9100

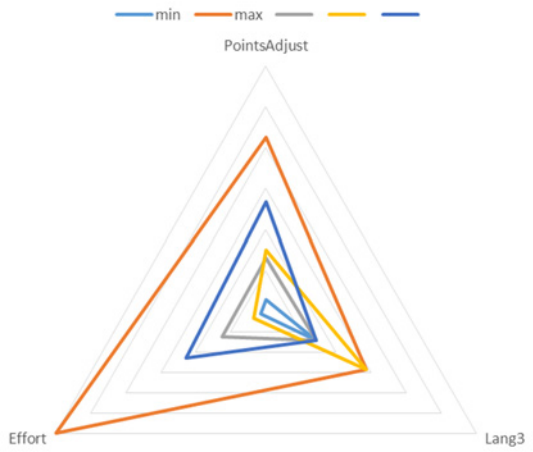

図 3.　Fit-test 法における 3 件のプロジェクトのレーダーチャート

表 6. Fit-test 法において全探索により選択された 3 件のプロジェクト

Project Number	Funtion Points	Lang3	Effort
30	398	1	1400
40	289	0	3983
57	709	0	9100

以上の違いが（4977 と 1400）が生じていることが読み取れる．このことは，Lang3 が 1 の開発においては生産性が極めて高く，開発工数が大幅に少なくて済むことを示しており，プロジェクト管理者が開発工数見積もりをするにあたって極めて重要な性質を示すことができていると考える．また，プロジェクト ID＝57 は，プロジェクト ID＝15 と比べて，Lang3 がどちらも 0 で，FunctionPoints が 2 倍近い（2.06 倍）値を取っており，そのために Effort も 2 倍近い（1.83 倍）値となっていることが読み取れる．このことは，FunctionPoints が Effort に大きく寄与しており，比例関係にあることをうかがわせる．やはり，プロジェクト管理者に極めて重要な性質を示すことができていると考える．

表 7.作成された開発工数予測モデル

全件	(Intercept)	FunctionPoints	Lang3
	2.263	1.099	−2.081
fit−test	(Intercept)	FunctionPoints	Lang3
	3.628	0.836	−2.004
fit−fit	(Intercept)	FunctionPoints	Lang3
	3.0589	0.9226	−1.9325

同様に，表 6 に，fit-fit 法の全探索により抽出された 3 件プロジェクトの内訳を示す．また，それら 3 件をレーダーチャートで表したものを図 4 に示す．表 5 の fit-test 法と比較すると，プロジェクト ID=15 とプロジェクト ID=40 が入れ替わっているが，これらは似た特徴を持っており，fit-test 法と同様，プロジェクト管理者に有用となるプロジェクトを抽出できているといえる．

表 7 に，fit dataset に含まれる全てのプロジェクト（60 件）から作成した工数予測モデルと，fit-test 法，fit-test 法のそれぞれで選択された 3 プロジェクトから作成した工数予測モデルを示す．表中，Intercept は重回帰モデルの定数項を示し，FunctionPoints と Lang3 はそれぞれの偏回帰係数を示す．3 つのモデルを比較すると，互いに類似しているといえる．このことからも，開発工数予測に必要な最小限のプロジェクトを選定できていると考えられる．

5. おわりに

本稿では，多数の開発プロジェクトの実績データを含むデータセットから，開発工数の見積もり（予測）時に過去の典型的なプロジェクトを参照する場面を想定し，予測に必要な最小限のプロジェクトのデータを抽出する 2 つの方法（fit-test 法，fit-fit 法）を提案した．いずれの方法においても，3 件のプロジェクトを抽出でき，それらが工数予測に役立つことと，プロジェクト管理者に必要な情報を与えることを確認できた．

今後の課題としては，原型分析など，他のアプローチを用いた方法についても検討していくことが考えらえる．

謝辞　本研究の一部は，JSPS 科研費基盤研究（C）課題番号 26330086 の補助を受けた．

参考文献

1) Basili, V. R. and Weiss, D. M.: A Methodology for Collecting Valid Software Engineering Data, IEEE Trans. Software Engineering, Vol. SE-10, No. 6, pp. 728-738 (1984).
2) Chapman, A. D.: Principles of Data Quality, version 1.0, Report for the Global Biodiversity Information Facility, Copenhagen (2005).
3) Desharnais, J. M. : Analyse Statistique de la Productivité des Projets de Développement en Informatique à Partir de la Technique des Points de Fonction in Program de maîtrise en informatique de gestion, Université du Québec à Montréal (1988).
4) 独立行政法人情報処理推進機構ソフトウェア高信頼化センター: ソフトウェア開発データ白書 2014-2015 (2014).
5) Howles, T.: Data, Data Quality, and Ethical Use, Software Quality Professional, Vol. 16, Issue 2, pp. 4-12 (2014).
6) Kemerer, C. F.: An Empirical Validation of Software Cost Estimation Models, Communications of the ACM, Vol. 30, No. 5, pp. 416-429 (1987).
7) Keung, J., Kitchenham,B., and Jeffery, R.: Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation, IEEE Trans. on Software Engineering, Vol. 34, No. 4, pp.471-484 (2008).
8) Keung, J., Kocaguneli, E. and Menzies, T.: Finding Conclusion Stability for Selecting the Best Effort Predictor in Software Effort Estimation, Automated Software Engineering, Vol. 20, No.4, pp. 543-567, pp. 543-567 (2013).
9) Liebchen, G. A. and Shepperd, M.: Data Sets and Data Quality in Software Engineering, Proc. 4th International Workshop on Predictor Models in Software Engineering (PROMISE'08), pp.39-44 (2008).
10) Liebchen, G. A., Twala, B., Shepperd, M. and Cartwright, M.: Assessing the Quality and Cleaning of a Software Project Data Set: An Experience Report, Proc. 10th International Conference on Evaluation and Assessment in Software Engineering (EASE2006) (2006).
11) Mendes, E., Martino, S., Ferrucci, F., Gravino, C.: Cross-company vs. single-company web effort models using the Tukutuku database: An extended study, Journal of Systems and Software, Vol.81, No.5, pp.673-690 (2008).
12) Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., Peters, F., Turhan, B.: The PROMISE Repository of Empirical Software Engineering Data (2012).
13) Phannachitta, P., Monden, A., Keung, J., Matsumoto, K.: Case consistency: A necessary data quality property for software engineering data sets, Proc. 19th International Conference on Evaluation and Assessment in Software Engineering (EASE2015), Article No. 19, (2015).
14) Seo, Y., Yoon, K., and Bae, D.: An Empirical Analysis of Software Effort Estimation with Outlier Elimination," Proc. International Workshop on Predictor Models in Software Engineering (PROMISE), pp.25-32 (2008).
15) 社団法人情報サービス産業協会: 情報サービス産業における受注ソフトウェア開発の技術的課題に関するアンケート調査 (2004).
16) Tsunoda, M., Monden, A., Yadohisa, H., Kikuchi, N. and Matsumoto, K.: Software Development Productivity of Japanese Enterprise Applications, Information Technology and Management, Vol. 10, No. 4, pp. 193-205 (2009).