社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE

工数見積もりモデルで予測できないソフトウェアプロジェクトの特徴分析

戸田 航史† 角田 雅照† 門田 暁人† 松本 健一†
†奈良先端科学技術大学院大学情報科学研究科 〒630-0192 奈良県生駒市高山町 8916-5
E-mail: †{koji-to, masate-t, akito-m, matumoto}@is.naist.jp

あらまし ソフトウェア開発プロジェクトの計画立案のためには，開発工数の予測が必須となる．そのために，重回帰モデル，ニューラルネットなどの見積もりモデルが多数提案されている．しかし，それらのモデルによる工数の予測精度は，プロジェクトごとに大きなばらつきがある．本研究では予測が外れやすい，またはばらつきやすいプロジェクトの特徴を明らかにするための実験を行った．その結果，予測が外れやすいプロジェクトの特徴のいくつかを特定した．

キーワード プロジェクト管理，工数予測，重回帰分析，相対誤差

Characterizing Software Projects Unpredictable by Effort Estimation Models

Koji TODA† Masateru TSUNODA† Akito MONDEN† and Ken-ichi MATSUMOTO†
†Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192 Japan
E-mail: †{ koji-to, masate-t, akito-m, matumoto }@is.naist.jp

Abstract Effort prediction is necessary for project manegers to make software development plan. To predict development effort, various estimation moedels such as multivariate regression models or neural network model have been proposed. However, there is much difference in effort prediction accuracy among predicted projects. In this study, we experimentally analyzed the project charactaristics whose prediction accuracy tend to become worse. As a result, we identified some characteristics.

Keyword Project Management, Effort Prediction, Multivariate Regression Analysis, Magunitude of Relative Error

1. まえがき

ソフトウェア開発プロジェクトを成功に導くためには，開発工数の予測を定期的に行い，作業済み工数（実工数）との乖離を随時把握することが重要となる．そのために，従来，重回帰モデル，ニューラルネット[9]，事例ベース推論(CBR)[8]などの数多くの工数見積もりモデルが提案されてきた．工数見積もりモデルとは，プロジェクトの特性値（開発規模，開発者数など）を説明変数とし，目的変数である工数との関係を数学的に表すものである．過去のプロジェクトで計測された特性値に基づいてモデルの構築を行い，現行プロジェクトで計測した特性値をモデルに代入することで工数の予測値を算出できる．

ただし，それらモデルによる工数の予測精度は，プロジェクトごとに大きなばらつきがある．一つの実例として，ステップワイズ対数重回帰モデルにより 68 件のプロジェクトの試験工数の予測を試みた結果を図 1に示す（予測方法の詳細は4章で述べる）．図 1は，工数予測値の相対誤差の分布を示している．この例では，相対誤差が 30%以内の「予測がほぼ的中した」プロジェクトが数多く存在する一方で，100%を超える「予測に失敗した」プロジェクトも少なからず存在する．このように，予測が大きく外れる可能性がある以上，たとえ平均的には誤差が小さくとも，予測値を信頼してプロジェクト管理を行うことは危険である．

工数見積もりモデルを現場で用いるためには，予測が外れやすい（もしくは予測精度がばらつきやすい）プロジェクトの特徴を明らかにしておくことが必要である．一つの典型的な例として，開発中に何らかのトラブルが発生して予定よりも工数が大きく超過したプロジェクトでは，当然のことながら予測は大きく外れることになる．トラブルを未然に防ぐために，様々なリスク評価手法が従来数多く提案されている[7]．ただし，トラブルの発生以外にも，予測が外れる要因は数多く存在する．例えば，外注率が高くて正確なデータの収集が難しい場合や，データ収集のための人員が不足している場合は，収集したデータ（特性値）の信頼性が低くなり，正確な予測が行えない可能性がある．また，開発手法や開発体

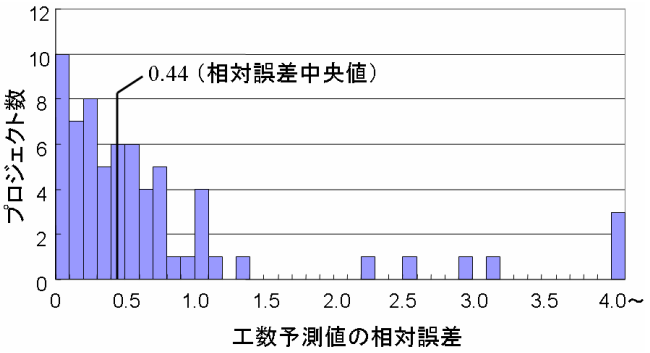

図 1. 工数予測値の相対誤差の分布の例

制によっては予測精度に大きなばらつきが出る可能性がある. 例えば, Java 言語を用いた開発では, (COBOL などと比較して)コードの自動生成を伴うことが多いため, ソースコード行数よりソフトウェア規模を計測すると予測精度が大きくばらつく可能性がある.

本論文では, 予測が外れやすい, もしくは, ばらつきやすいプロジェクトの特徴を実験的に明らかにすることを目的とする. 実験では, まず, 重回帰分析により多数のプロジェクトの工数予測を行い, 各プロジェクトの予測誤差を求める. 次に, 特性値ごとに閾値を設けてプロジェクトを 2 つ以上のグループに分割し, グループ間で予測誤差の平均値やばらつき(標準偏差)に有意な差があるかどうかを検定する. この検定を全ての特性値について行う. 最後に, 有意差が見られた特性値, 及び, その閾値について, その原因を考察する. ただし, 本論文では, 紙面の都合上, 予測誤差の平均値の有意差の分析結果のみを報告する.

以降, 2章では本研究で用いた見積もり方法であるステップワイズ重回帰分析について述べる. 3章において, 本研究で用いたデータの詳細について説明する. 4章で実験方法, および結果を述べたあと, 5章で結果に対する考察を行う. 6章で関連研究について説明し, 最後に7章でまとめを述べる.

2. ステップワイズ重回帰分析

本研究では, 工数見積もり方法として, 多変量解析の手法の一つであるステップワイズ重回帰分析を用いた. 工数見積もり方法として, 重回帰分析以外にも, ニューラルネット[9] , 事例ベース推論(CBR)[8]などの数多くの工数見積もりモデルが提案されている. ただし, 重回帰分析はソフトウェア開発工数見積もりモデルの作成において最も一般的に用いられている[1][11]ため, これを採用した.

重回帰分析は, ある変数(目的変数)と, それに対して影響すると考えられる複数の変数(説明変数)の間の関係を一次式で表した見積もりモデルを作成し, 作成された見積もりモデルに基づいて説明変数から目的変数を予測する手法である. 工数予測においては, 工数が目的変数であり, それ以外の特性値が説明変数の候補となる. 目的変数と説明変数の関係を表す一次式は, 絶対誤差の 2 乗和が最小となる係数が与えられる.

ステップワイズ重回帰分析は, ステップワイズ変数選択法により説明変数を決定し重回帰分析を行う手法である. ステップワイズ変数選択法は, 目的変数に対して強く影響する変数を説明変数として選択する手法の一つである. ステップワイズ変数選択法による変数選択は以下の手順で行われる.

1. 初期モデルを作成する.
2. 作成されたモデルに対して, 各説明変数の係数が 0 でないかの検定を行い, 指定した有意水準で棄却されない場合に変数を採択し, 棄却される場合には変数を除去する(ただし, 多重共線性を回避するために, 採択する変数の分散拡大要因が 10 以上の場合, またはその変数を採択することによって, 他の変数の分散拡大要因が 10 以上となる場合, その変数は採択しないという操作が行われる[10]).
3. 手順 2 ができなくなるまで繰り返す.

ステップワイズ変数選択法には, 変数増減法と変数減増法がある. 変数増減法では, 変数を全く含まないモデルを初期モデルとし, 変数減増法では, 全ての変数を含むモデルを初期モデルとして変数選択を行う. 本研究では変数増減法を用いた.

3. ISBSG データセット

実験に用いたデータセットは International Software Benchmarking Standards Group (ISBSG)が収集した, 20 ヶ国のソフトウェア開発企業の実績データである[1]. データセットには, 1989 年から 2004 年までの 3026 件のプロジェクトについて, それぞれ 99 種類の特性値が記録されている. ただし, このデータセットには数多くの欠損値が含まれている.

本実験では, 欠損を多く含むプロジェクト, および, 特性値を除外し, 欠損を含まない 134 件のプロジェクト, 11 種類の特性値を含むデータセットを作成した. これは欠損値の多いプロジェクトでは予測精度が下がり, 実験結果に悪影響を与えると考えられるためである.

さらに, 11 種類のうち Effort Plan, Effort Specify, Effort Build の 3 種類の特性値を用いて, まずこれらの工数の合計を算出, 次に合計に対して 3 つの工数のおのおのが占める割合を算出し, 特性値としてデータセットに追加した(Effort Plan Ratio, Effort Specify Ratio, Effort Build Ratio). これらを新たに定義した理由は, 設計工程や製造工程などの各工程の比率の違いが, 予測に影響する可能性を考慮したためである.

実験に用いた特性値を表 1に示す. 特に説明のないものはカテゴリ変数である. 表中の Development Type から Effort Build Ratio までの 13 種類の特性値を説明変数とし, Effort Test を目的変数とした. ただし, 量的変数は正規分布していなかったため, あらかじめ値を対数変換した. カテゴリ変数は, それぞれ 1 つ以上のダミー変数(0 または 1 の値を取る変数)に変換したものを用いた. また, Resource Level 3 については, これを含むプロジェクトが存在しなかった.

ここで Recording Method のカテゴリ変数である Productive Time Only と Stuff Hours, Resource Level のカテゴリ変数である Resource Level 1〜4 について詳しく説明する. Recording Method と Resource Level は, どちらも工数の測定方法に関係するカテゴリ変数である. 工数の単位は人時であり, その算出には人的要素と時間的要素が必要

表 1. 実験に用いた特性値

特性値名	説明
Development Type	New development or Enhancement
Adjusted Function Points	数量データ
Count Approach	COSMICFFP, IFPUG or MESMA
Architecture	Client Server, Stand Alone or Multi-tier
Primary Programming Language	COBOL, Java, PL/I, TELON or Others
Recording Method	Productive Time Only, Stuff Hours or Others
Resource Level	工数として記録する作業範囲. Level 1～4
Effort Plan (E_1)	数量データ
Effort Plan Ratio	$E_1 / (E_1 + E_2 + E_3)$
Effort Specify (E_2)	数量データ
Effort Specify Ratio	$E_2 / (E_1 + E_2 + E_3)$
Effort Build (E_3)	数量データ
Effort Build Ratio	$E_3 / (E_1 + E_2 + E_3)$
Effort Test	数量データ

である. このうち人的要素に関わるのが Resource Level であり, 時間的要素に関わるのが Recording Method である.

Resource Level とは, プロジェクトとしての作業内容の定義であり, どのような業務を行っていた「人員」をプロジェクトの開発に関わっていると認識し, 報告したかを表している. これに対して Recording Method は作業時間の記録方法の定義であり, どのような業務に費やした「時間」をプロジェクトの開発として認め, 報告したかを表している.

具体例を以下に示す. まず Resource Level 1 とは, 開発チーム(プロジェクトチーム, プロジェクトマネジメント, プロジェクト管理)の業務を作業時間として認めるものであり, 以下, Resource Level 2 では開発チームのサポート(データベース管理, データ管理, 品質保証, データセキュリティ, 標準化サポート等), Resource Level 3 ではコンピュータオペレーションに関連するもの(ソフトウェアのサポート, ハードウェアのサポート, ネットワーク管理等), Resource Level 4 ではエンドユーザやクライアントに関連するもの(ユーザを訓練する時間, アプリケーションユーザやクライアント等)までを含めて, 工数として報告している. 次に Recording Method の Productive Time Only とは, プロジェクト開発に費やした時間のみを工数として記録する方法であり, 教育など, 直接にはプロジェクト開発に関わらない時間などは一切含まれない. これに対して Stuff Hours とは, プロジェクトに関係する作業であれば, 教育などの直接関係がない作業であっても, それに費やされた時間を工数として毎日報告・記録する方法である.

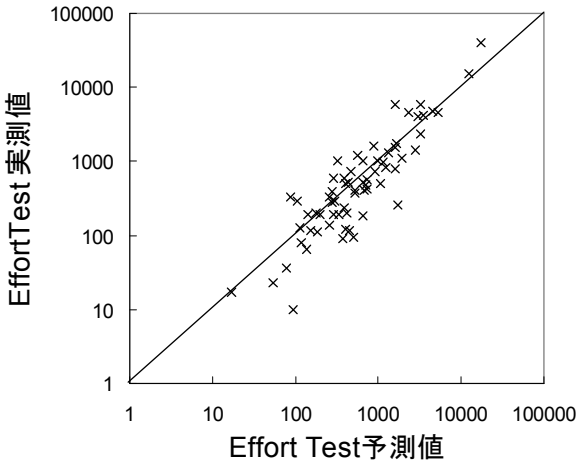

図 2. Effort Test の予測値と実測値の関係

4. 実験

4.1. 実験手順

実験の手順は次の通りである.

[手順 1] 134 件のプロジェクトを, 目的変数(Effort Test)の分布がほぼ等しくなるように 2 等分し, 一方をモデル構築用のフィットデータ, もう一方を予測値算出用のテストデータとした. 具体的には, 134 件のプロジェクトを, Effort Test の大きな順に並べたものを $P_1, P_2, P_3, \ldots, P_{134}$ としたとき, 奇数番号のプロジェクトをフィットデータとし, 偶数番号のプロジェクトをテストデータとした.

[手順 2] フィットデータを用いて重回帰モデルを構築した. モデル構築にあっては, ステップワイズ変数選択を行った.

[手順 3] 構築したモデルにテストデータを適用し, 各プロジェクトの Effort Test の予測値を算出した.

[手順 4] 各プロジェクトについて予測値の相対誤差(MRE: Magnitude of Relative Error)を算出した. MRE は実績工数を E, 予測工数を $\hat{E}$ とするとき, 次の式で定義される[2].

$$MRE = \frac{\left|E - \hat{E}\right|}{E} \quad (1)$$

[手順 5] 特性値ごとに閾値を設けてプロジェクトを 2 つ以上の群に分割し, 群間で予測誤差に有意な差があるかどうかを検定した. 具体的には, 量的変数については中央値を閾値として大小の 2 群に分けた. そして, 片方の群に属するプロジェクトの相対誤差と, もう一方の群に属するプロジェクトの相対誤差について平均値の差の検定(t 検定)を行った. カテゴリ変数については, カテゴリ数が 3 以下の場合, 各カテゴリを群とみなして全ての 2 群の間で t 検定を行った. また, カテゴリ数が 4 以上の場合, 全カテゴリについて, 1 つのカテゴリに属する群とそれ以外の

群の 2 群に分けて t 検定を行った.

4.2. 実験結果

実験において作成されたモデルを式(2)に示す. ここで, *ln*(*n*)は *n* の自然対数を示し, Development Type is Enhancement と Recording Method is Productive Time only はダミー変数を示す.

ln(Effort Test)
＝5.51＋1.07(*ln*(E_2))－17.25(*ln*(Effort Specify Ratio)
＋0.47(Development Type is Enhancement)
－0.68(Recording Method is Productive Time only) (2)

Effort Test の予測値を横軸にとり, Effort Test の実測値を縦軸にとった散布図を図 2に示す. 散布図の対角線から遠いプロジェクトは, 予測が大きく外れたプロジェクトである. 相対誤差の分布は図 1のようになった.

相対誤差の平均値の差の検定を行った結果, 有意水準 10%で有意差があったものは, 以下の 4 つであった.

- Primary Programming Language の COBOL と Others(それ以外の言語) (*P*=0.068)
- Recording Method のうち Productive Time Only と Others(それ以外の時間測定法) (*P* =0.091)
- Stuff Hours と Others(それ以外の時間測定法) (*P* =0.064)
- Resource Level のうち Resource Level 2 と Resource Level4 (*P* =0.038)

有意差が見られた 4 つにはそれぞれ 2 群が含まれているが, 片方の群に属するプロジェクトの相対誤差と, もう一方の群に属するプロジェクトの相対誤差の分布をグラフで示したのが図 3である. ここでグラフは特性値ごとに分けたが, Productive Time Only と Others, Stuff Hours と Others は同じカテゴリ変数に属しているので, これらについては一つのグラフにまとめて示した.

この結果から分かるように, まず COBOL と Others のグラフでは, あきらかに Others のばらつきが大きいことが分かる. 次に Productive Time Only, Stuff Hours, Others のグラフでは, 全体としてばらついてはいるものの, Productive Time Only と Stuff Hours はその多くが相対誤差 1.0 以内に固まっているのに対して, Others は相対誤差の値に関係なくばらついていると言える. 最後に Resource Level 1,2,4 については, Level 4 ではほとんどが相対誤差 1.0 以内に収まっているのに対して Level 2 は相対誤差のばらつきが非常に大きいことが分かる. また Resource Level 1 については多くが相対誤差 1.0 以内にあるものの, それを大きく外れた値もあることが分かる.

さらに, 有意差が見られた 4 つについて, 箱ひげ図で示し

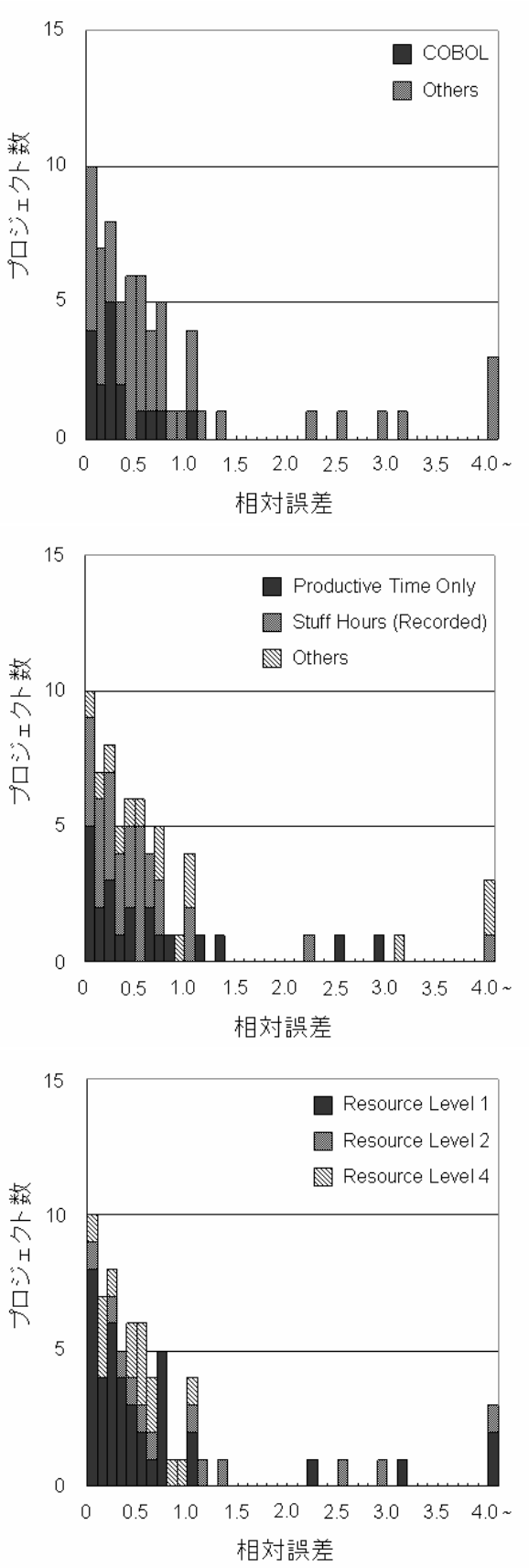

図 3. 特性値ごとの相対誤差のヒストグラム

たものが図 4である. この図では図 3における誤差のばらつきがより明確に示され, 特にCOBOLとOthersの間の差がはっきりと分かる. また, Resource Level 1 については中央値から大きく外れた値が多いことも一見して分かる.

5. 考察

有意差のあった4つの場合について, その理由を考察する. Primary Programming Language の COBOL と Others については, 1章でも述べたが, COBOL はコードの自動生成を行うことがほとんどなく, このためにソースコードの規模が開発規模に比例するし, その結果として相対誤差が小さくなりやすいと考えられる.

また Others には C や Visual Basic など様々な言語が含まれており, これによって相対誤差のばらつきが大きくなり, 有意差がみられたと考えられる.

本研究では言語をひとくくりにしてモデルを作成したが, 言語によってその仕様が大きく異なることから, モデルを作成する上では言語ごとに別のモデルを作成する必要があるとも考えられる.

Recording Method の Productive Time Only と Others, Stuff Hours と Others については, Primary Programming Language と同じく, Others に様々な手法が含まれていることを考えれば, Others のばらつきが大きくなることは当然であり, 有意差がみられたのは妥当な結果だと考えられる. また, Others に含まれているプロジェクト数は少なく, Recording Method は工数に直接関わってくることもあり, モデル構築に最初から含めないほうが, 正確な予測が行えるのではないかと考えられる.

Resource Level のうち Resource Level 2 と Resource Level 4 については, Resource Level 2 のばらつきが非常に大きく, このために有意差が現れたと考えられる. このばらつきの原因としては, Level 1 ではほぼ開発チームのみ, Level 4 では少しでもプロジェクトに関わった全ての開発者を「開発に携わった」をデータの収集対象にする, という非常に分かりやすい定義づけがされている. これに対して, Level 2 や Level 3 では「開発チームのサポート」や「ソフトウェア, ハードウェアのサポート」というように定義が非常に曖昧で, この結果, 報告された Level で定義されている業務内容と実際に行っている業務内容が異なっているという状況が起きたためではないかと考えられる.

6. 関連研究

予測が外れる要因を分析した研究がいくつか行われている[5]. これらの多くはインタビュー結果に基づいた分析であるのに対し, 本研究は特性値に基づく分析を行っている点が異なる.

Mizuno ら[6]は, コストの実測値が予測値よりも超過したプロジェクトについて分析を行っている. Mizuno らの研究

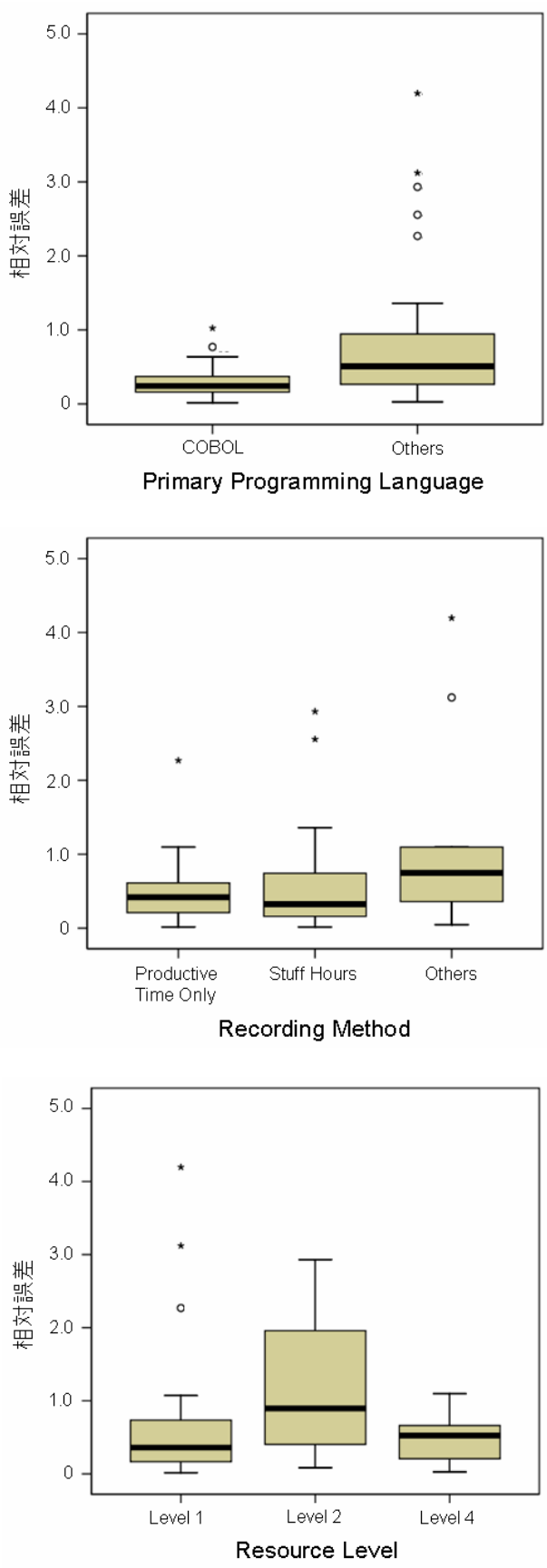

図 4. 特性値ごとの相対誤差の箱ひげ図

では，|実測コスト－予測コスト|÷予測コストを誤差と定義している．そして，完了したプロジェクトについて，プロジェクトの計画(WBS，PERT 図など)や執行(ソフトウェアレビューなど)が，どの程度厳格に行われたかを，プロジェクトマネージャに得点付けしてもらい，その得点の合計と，誤差との相関を調べている．

Jorgensenら[5]は，予測工数に誤差が発生した理由に対して，影響を及ぼしている要因について分析を行っている．あるソフトウェア開発企業において，見積もりに責任を持ち，それぞれの立場が異なる複数の従業員にインタビューを行い，68 個のプロジェクトの特徴と誤差との関係について分析を行っている．その結果，相対誤差が起きた理由に影響を及ぼしているのは，回答者の立場，データの収集方法，データの分析方法にあるとしている．この研究は，インタビューの結果に基づいた分析を行う際の問題点について分析したものである．

特性値に基づき，予測が外れる要因を分析した研究も少数ながら存在する．Gray ら[3]は，モジュールのタイプ，および特徴から，工数予測が過少予測，適正予測，過大予測のいずれになりやすいかを分析している．ただし，この研究は，エキスパートによる見積もりと実績値との差を分析している点が，本研究と異なる．

7. むすび

本論文では，予測が外れやすい，または，ばらつきやすいプロジェクトの特徴を明らかにするための実験を行った．まず工数の予測を行い，その結果をもとに特性値ごとにプロジェクトを2つ以上のグループに分割した．それを用いて，グループ間での予測誤差の平均値に有意な差があるかどうかを検証した結果，いくつかの特性値では有意差がみられた．

今後，相対誤差の平均値だけでなく，そのばらつきについても検定を行うこと，ニューラルネット等，他の予測手法を行った場合の結果との比較を行うこと，特性値の値，例えば言語ごとにモデルを構築した場合の結果と本実験の結果との比較を予定している．

謝　辞

本研究の一部は，文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．

文　献

[1] L. Briand, T. Langley, and I. Wieczorrek, “A replicated assessment and comparison of common software cost modeling techniques,” In Proc. 22nd IEEE International Conf. on Software Eng., Limerick, pp.377-386, 2000.

[2] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1986.

[3] A. R. Gray, S. G. MacDonell, and M. J. Shepperd, “Factors Systematically Associated with Errors in Subjective Estimates of Software Development Effort: The Stability of Expert Judgment,” In Proc. 6th International Software Metrics Symposium (METRICS‘99), pp.216-227, Boca Raton, FL, Nov, 1999.

[4] International Software Benchmarking Standards Group (ISBSG),”ISBSG Estimating, Benchmarking and Research Suite Release 9,” ISBSG, 2004.

[5] M. Jorgensen, and K. Molokken-Ostvold, “Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method,” IEEE Trans. on Software Eng., Vol.30, No.12, pp.993-1007, 2004.

[6] O. Mizuno, T. Kikuno, K. Inagaki, Y. Takagi, and K. Sakamoto, “Statistical analysis of deviation of actual cost from estimated cost using actual project data,” Information and Software Technology, Vol.42, No.7, pp.445-515, 2000.

[7] Project Management Institute (PMP), A Guide to the Project Management Body of Knowledge 2000 Edition, PMP, 2000.

[8] M. Shepperd, and C. Schofield, “Estimating Software Project Effort Using Analogies,” IEEE Trans. on Software Eng., Vol.23, No.12, pp.736-743, 1997.

[9] K. Srinivasan, and D. Fisher, “Machine Learning Approaches to Estimating Software Development Effort,” IEEE Trans. on Software Eng., Vol.21, No.2, pp.126-137, 1995.

[10] 田中豊，垂水共之，統計解析ハンドブック 多変量解析，共立出版，東京，1998.

[11] C. Walston, and C. Felix, “A method of programming measurement and estimation,” IBM Systems Journal, Vol.1, pp.54-73, 1977.