NAIST-IS-DD0561010

博士論文

ソフトウェア開発管理における予測手法の利用的側面からの評価

柿元 健

2008年2月7日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

柿元 健

審査委員：

松本 健一 教授	（主指導教員）
関 浩之 教授	（指導教員）
飯田 元 教授	（指導教員）
門田 暁人 准教授	（指導教員）

ソフトウェア開発管理における予測手法の利用的側面からの評価*

柿元 健

内容梗概

ソフトウェア開発管理を支援するために，開発工数と信頼性に関して数多くの予測手法が提案されているにも関わらず，その多くは開発現場で利用されていない．本論文では，予測手法を利用的側面から評価することで，その適用の基準や得られる効果を明確にし，開発現場での採用を促進することを目的とする．

まず，開発工数の予測において，予測に必要なデータセットに求められる基準を明らかにするために，データセットに含まれる欠損値，プロジェクトの件数，メトリクス数について，類似性に基づく工数予測手法とステップワイズ重回帰分析の 2 種類の工数予測手法を評価した．欠損値について，欠損値が生じる 3 つのメカニズムを想定し，それぞれについて欠損率を変化させたデータセットを多数作成し，各データセットを用いて工数予測を行うことで実験的に評価した．評価実験の結果，類似性に基づく工数予測手法が，ステップワイズ重回帰分析よりもロバスト性が高い，すなわち，欠損のメカニズムに関わらず，欠損率が増大しても予測精度が大きく低下しないことが示された．また，プロジェクト件数とメトリクス数について，それぞれを変化させたデータセットを作成し，各データセットを用いて工数予測を行うことで実験的に評価した．評価実験の結果，プロジェクト件数が 50 件以上の場合，類似性に基づく工数予測手法がステップワイズ重回帰分析よりも高い精度で予測でき，メトリクス数の増加に伴って予測精度が向上することが示された．

* 奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DD0561010, 2008 年 2 月 7 日.

次に，信頼性予測の一手法である fault-prone モジュール判別において，予測により得られる効果を明らかにするために，各モジュールの fault の有無，割り当てられるテスト工数，およびソフトウェアの信頼性（fault 発見率）の関係をモデル化した．費やされたテスト工数に基づいて fault 発見率を示すモデルを構築した．構築した fault 発見率モデルを用いて，評価指標の 1 つである F1 値，fault モジュール含有率，fault-prone モジュール率をパラメータとしてシミュレーションを行った．シミュレーションの結果，プロジェクト全体の信頼性（fault 発見率）は，F1 値，fault モジュール含有率，fault-prone と判別されたモジュールの割合によって決定されることが示された．

キーワード

工数見積もり，fault-prone モジュール判別，欠損値，プロジェクト件数，メトリクス数，fault 発見率

User-side Evaluation of Estimation Methods in Software Development Management*

Takeshi Kakimoto

Abstract

Recently, various methods to estimate software development effort or software reliability have been proposed to assist software development management; however, most of them are not actually used in software development companies. To encourage companies to use estimation methods, this thesis evaluates existing methods from the perspective of their users, and clarifies their application requirements and benefits.

Firstly, to clarify the requirements for a project dataset, which is used for building effort estimation models, this thesis experimentally analyzes how the missing data, the number of projects and the number of metrics in the dataset affect the estimation accuracy of analogy-based method and stepwise multiple regression model. In the experiment, a set of datasets containing different numbers and types of missing values were built as an experimental testbed based on three data missing mechanisms. The result showed that the analogy-based method was more robust than conventional stepwise regression models for all types of datasets, i.e. estimation accuracy of the analogy-based method was kept high regardless of increase of data missing ratio (up to 60%) and missing mechanisms. Then, a set of datasets containing different numbers of projects and metrics were built as another testbed. The result showed that the analogy-based method showed

* Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561010, February 7, 2008.

higher estimation accuracy than conventional stepwise regression models when the number of projects became greater than fifty, and the estimation accuracy of the analogy-based method kept improving as the number of metrics increased.

Secondly, to clarify the benefts of the fault-prone module prediction model, which is one of the reliability estimation methods, this thesis proposes a model that represents the relationship among the accuracy of fault-prone module prediction, the test effort assigned to fault-prone/not-fault-prone modules based on the prediction, and discoverable faults estimated by an exponential-type software reliability growth model. The result of an simulation based on the proposed model showed that the percentage of discoverable faults was affected by the F1 value of fault-prone module prediction, percentage of (actual) faulty modules in software, and percentage of modules predicted as faulty.

Keywords:

Effort Estimation, Fault-prone Module Detection, Missing Value, Number of Projects, Number of Metrics, Fault Detection Rate

関連発表論文

第 2 章に関連する発表論文

学術論文誌

1. 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本 健一, “協調フィルタリングに基づく工数見積もり手法のデータの欠損に対するロバスト性の評価,” 電子情報通信学会論文誌 D, Vol.J89-D, No.12, pp.2602-2611, December 2006.

国内会議（査読あり）

1. 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本 健一, “協調フィルタリングに基づく工数見積もりのロバスト性評価,” 野呂昌満, 山本晋一郎 (編), ソフトウェア工学の基礎 XI, 日本ソフトウェア科学会 FOSE2004, pp.73-84, November 2004.

第 3 章に関連する発表論文

国内会議（査読あり）

1. 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本 健一, “協調フィルタリングによる工数見積もり手法におけるデータ数と見積もり精度の関係の分析,” 権藤 克彦, 小林 隆志 (編), ソフトウェア工学の基礎 XII, 日本ソフトウェア科学会 FOSE2005, pp.77-86, November 2005.

第 4 章に関連する発表論文

国内会議（査読あり）

1. 柿元 健, 門田 暁人, 亀井 靖高, 柗本 真佑, 松本 健一, “Fault-Prone モジュール判別における F1 値とソフトウェア信頼性の関係,” ソフトウェア工学の基礎 XIV, 日本ソフトウェア科学会 FOSE2007, pp.75-83, November 2007.

その他の発表論文

学術論文誌

1. 亀井 靖高, 柗本 真佑, 柿元 健, 門田 暁人, 松本 健一, “Fault-prone モジュール判別におけるサンプリング法適用の効果,” 情報処理学会論文誌, vol.48, no.8, pp.2651-2662, August 2007.

国際会議

1. Yasutaka Kamei, Akito Monden, Shinsuke Matsumoto, Takeshi Kakimoto, and Ken-ichi Matsumoto, “The Effects of Over and under Sampling on Fault-Prone Module Detection,” In Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement, pp.196-204, September 2007.

2. Naoki Ohsugi, Akito Monden, Nahomi Kikuchi, Michael Barker, Masateru Tsunoda, Takeshi Kakimoto, and Ken-ichi Matsumoto, “Is This Cost Estimate Reliable? – the Relationship between Homogeneity of Analogues and Estimation Reliability,” In Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement, pp.384-392, September 2007.

3. Takeshi Kakimoto, Yasutaka Kamei, Masao Ohira, and Ken-ichi Matsumoto, “Social Network Analysis on Communications for Knowledge Collaboration in Oss Communities,” In Proceedings of 2nd International Workshop on Supporting Knowledge Collaboration in Software Development, pp.35-41, September 2006.

4. Masateru Tsunoda, Akito Monden, Takeshi Kakimoto, Yasutaka Kamei, and Ken-ichi Matsumoto, “Analyzing Oss Developers’ Working Time Using Mailing Lists Archives,” In Proceedings of International Workshop on Mining Software Repositories Mining Challenge 2006, pp.181-182, May 2006.

5. Takeshi Kakimoto, Akito Monden, Yasutaka Kamei, Haruaki Tamada, Masateru Tsunoda, and Ken-ichi Matsumoto, “Using Software Birthmarks to Identify Similar Classes and Major Functionalities,” In Proceedings of 3rd International Workshop on Mining Software Repositories Mining Challenge 2006, pp.171-172, May 2006.

6. Masao Ohira, Tetsuya Ohoka, Takeshi Kakimoto, Naoki Ohsugi, and Ken-ichi Matsumoto, “Supporting Knowledge Collaboration Using Social Networks in A Large-Scale Online Community of Software Development Projects,” In Proceedings of 1st International Workshop on Supporting Knowledge Collaboration in Software Development, pp.835-840, December 2005.

7. Tomohiro Akinaga, Naoki Ohsugi, Masateru Tsunoda, Takeshi Kakimoto, Akito Monden, and Ken-ichi Matsumoto, “Recommendation of Software Technologies based on Collaborative Filtering,” In 12th Asia-Pacific Software Engineering Conference, pp.209-214, December 2005.

8. Masateru Tsunoda, Takeshi Kakimoto, Naoki Ohsugi, Akito Monden, and Ken-ichi Matsumoto, “Javawock: A Java Class Recommender System based on Collaborative Filtering,” In Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering, pp.491-497, July 2005.

国内会議（査読あり）

1. 前島弘敬, 柗本真佑, 亀井靖高, 柿元健, 大西洋司, 大平雅雄, 松本健一, “コーディネータのコミュニティ媒介性の評価指標の提案,” グループウェアとネットワークサービスワークショップ 2007, pp.71-76, November 2007.

2. 田村 晃一, 柿元 健, 戸田 航史, 角田 雅照, 門田 暁人, 松本 健一, “工数予測における類似性に基づく欠損値補完の効果,” ソフトウェア工学の基礎 XIV, 日本ソフトウェア科学会 FOSE2007, pp.85-93, November 2007.

国内会議（査読なし）

1. 渡邊 瑞穂, 柿元 健, 戸田 航史, 門田 暁人, 松本 健一, “ソフトウェア開発工数予測における特異プロジェクト除去の効果,” 日本ソフトウェア科学会第 24 回大会論文集 CD-ROM(講演番号 7B-2), September 2007.

2. 田村 晃一, 柿元 健, 戸田 航史, 角田 雅照, 門田 暁人, 松本 健一, “プロジェクト間の類似性に基づくソフトウェアメトリクスの欠損値の補完,” ソフトウェア信頼性研究会 第 4 回ワークショップ, pp.17-23, June 2007.

3. 瀧 進也, 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本健一, “プロジェクト類似性に基づく工数見積もりのための変数選択,” 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, no.SS2006-49, pp.1-6, October 2006.

4. 亀井 靖高, 角田 雅照, 柿元 健, 大杉 直樹, 門田 暁人, 松本 健一, “進行中のプロジェクトに有用なソフトウェアコンポーネントの推薦方法,” 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, no.SS2006-12, pp.25-30, April 2006.

5. 柿元 健, 大岡 徹也, 大平 雅雄, 大杉 直樹, 松本 健一, “オープンソースソフトウェア開発コミュニティにおける知識協創支援へ向けたプロトタイプシステムの開発〜大規模サイバーワールドにおけるスケールフリーネットワークの問題点と対策〜,” 電子情報通信学会サイバーワールド研究会, 第 2 回研究会, pp.33-38, January 2006.

6. 角田 雅照, 柿元 健, 大杉 直樹, 門田 暁人, 松本 健一, “類似プロジェクトを用いた工数予測方法の性能比較,” 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, no.SS2005-47, pp.45-50, October 2005.

7. 本村 拓也, 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本 健一, “協調フィルタリングを用いたプロジェクトコスト超過の予測,” 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, no.SS2005-39, pp.35-40, August 2005.

8. 秋永 知宏, 大杉 直樹, 柿元 健, 角田 雅照, 門田 暁人, 松本 健一, “協調フィルタリングに基づくソフトウェア開発技術の推薦,” 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, no.SS2005-13, pp.7-13, June 2005.

9. 柿元 健, 角田 雅照, 大杉 直樹, 門田 暁人, 松本 健一, “協調フィルタリングに基づく Java クラスファイル推薦システム,” 電子情報通信学会技術研究報告, ディペンダブルコンピューティング研究会, vol.DC2004-26, pp.29-34, October 2004.

目 次

第 1 章　はじめに **1**

1. 研究の背景と目的 1
2. 論文構成 4

第 2 章　工数予測手法の欠損値に対するロバスト性の評価 **6**

1. まえがき 6
2. 欠損値を考慮した工数予測手法 8
   - 2.1 類似性に基づく工数予測手法 8
   - 2.2 MDT とステップワイズ重回帰分析の併用 11
3. ロバスト性の評価方法 12
   - 3.1 欠損メカニズム 13
   - 3.2 実験データセット作成手順 13
   - 3.3 ロバスト性の評価尺度 16
4. 評価実験 17
   - 4.1 基礎データセット 17
   - 4.2 評価基準 17
   - 4.3 実験手順 19
5. 結果と考察 20
   - 5.1 欠損率に関するロバスト性 20
   - 5.2 欠損メカニズムの違いに関するロバスト性 24
6. 関連研究 26
7. まとめ 27

第 3 章　工数予測手法におけるデータ数と予測精度の関係の分析　29
1. はじめに 29
2. 実験 31
2.1 実験で利用したデータ 31
2.2 評価基準 31
2.3 実験手順 32
3. 実験の結果と考察 33
3.1 プロジェクト件数の変化 33
3.2 メトリクス数の変化 35
3.3 考察 38
4. 関連研究 40
5. まとめ 41

第 4 章　Fault-Prone モジュール判別における F1 値とソフトウェア信頼性の関係　42
1. はじめに 42
2. Fault 発見率モデルの構築 43
2.1 テスト工数の割り当てに関する仮定 43
2.2 1 個のモジュールに対する fault 発見率モデル 44
2.3 全モジュールに対する fault 発見率モデル 45
3. シミュレーション 48
3.1 シミュレーションの設定 48
3.2 シミュレーション 1：fault-prone モジュールに割り当てるテスト工数の倍率 r を変化させた場合 49
3.3 シミュレーション 2：F1 値を固定して他の条件を変動させた場合 50
3.4 シミュレーション 3：条件を固定して F1 値を変化させた場合 52
4. 考察 52
5. 関連研究 55
6. まとめ 56

第 5 章　おわりに **57**

謝辞 **59**

参考文献 **61**

図 目 次

2.1 MAR と NM の欠損値生成手順 15
2.2 実験におけるデータ加工 21
2.3 各欠損率における予測精度 (MCAR) 23
2.4 各欠損率における予測精度 (MAR) 23
2.5 各欠損率における予測精度 (NM) 24
2.6 各欠損メカニズムの予測精度 26

3.1 プロジェクト件数の変化の実験におけるデータの加工手順 34
3.2 メトリクス数の変化の実験におけるデータの加工手順 34
3.3 各プロジェクト件数における予測精度 36
3.4 各メトリクス数における予測精度 37
3.5 設計工数未追加時との比較 39

4.1 モジュールの関係 45
4.2 Fault-prone モジュールに割り当てるテスト工数の倍率の変化 . . . 50
4.3 F1 値固定時の fault 発見率 51
4.4 F1 値を変化させたときの fault-prone モジュール率ごとの fault 発見率 53

表 目 次

2.1 実験データセットの各メトリクスの統計量 18
2.2 変数選択率 21
2.3 予測誤差の中央値 25

第1章　はじめに

1．研究の背景と目的

ソフトウェア開発プロジェクトにおいて，開発計画の立案や管理を円滑に進めるために，当該プロジェクトに関して様々な予測が行われる．特に重要となるのは，開発工数（人月または人時）と信頼性（fault 数または fault の有無）の予測である．開発工数の予測により，開発に必要な期間や要員数が決定され，要員の調達や要員割り当てなどの決定も可能となる．また，信頼性の予測では，ソフトウェア全体または個々のモジュールの残存 fault 数や fault の有無を予測することで，試験工数の割り当てや試験終了の時期の決定が可能となり，信頼性の向上や無駄な試験工数の削減が期待される．このように，適切な予測に基づいて開発計画を立案することで，コスト超過や納期遅れなどのリスクを抑えつつ，要求される信頼性を確保できることが見込まれる．

これらの予測を行うために，過去の開発プロジェクトで計測，収集されたデータを用いる手法が多数提案されている [1][3][7][8][21][30][37][38]．開発工数の予測は，プロジェクトの特性値（メトリクスと呼ばれる．例えば，プロジェクトの規模，開発ソフトウェアの対象ドメインやアーキテクチャ等）を入力として行われる．また，信頼性予測は，モジュールのメトリクス（例えば，コード行数，分岐数，サイクロマティック数など）を入力として行われる．いずれの手法も，開発者の勘や主観に左右されることなく，過去のデータを入力として数学的に予測結果を得ることができる．

しかし，予測手法が数多く提案され，実データを用いた予測精度の評価も行われているにも関わらず，その多くは開発現場において採用されないという問題がある．開発現場において採用が見送られる原因として，次の問題が挙げられる．

問題 1 (予測に必要なデータセットの基準の欠如) 開発工数予測では，各開発組織における過去プロジェクトを計測したデータセットが（開発組織ごとに）必要となる．組織が違えば，データに表れない開発の詳細（開発体制や開発プロセスなど）が異なるため，他の組織で計測されたデータを用いたとしても，十分な予測精度が得られない．そのため，開発組織ごとにデータ収集計画を立てて，十分なプロジェクト数，及び，メトリクス数のデータセットを収集する必要がある．

しかし，必要とされるデータセットの質や量に関する基準がないのが現状であり，開発現場において開発工数予測を行う際の障害となっている．特に，ソフトウェア開発データに多く含まれているデータ欠損の許容範囲や，必要なデータセットのサイズ（プロジェクトの件数，メトリクスの数）を明らかにすることが求められる．

問題 2 (予測後のアクションの決定および評価方法の欠如) 信頼性予測，特に，モジュールの fault の有無の予測においては，予測しただけでは信頼性の確保，テスト工数の削減ともに達成できず，予測後のアクションの決定が重要となる．具体的には，予測結果に基づいて，各モジュールに対するテスト工数を適切に割り当てることが重要となる．その基本方針としては，fault ありと予測されたモジュールにはより多くのテスト工数を割り当て，fault なしと予測されたモジュールにはより少ないテスト工数を割り当てることである．ところが，一般に，予測はある程度の誤りを含むため，工数の割り当てには慎重を要する．仮に，fault なしと予測されたモジュールに一切テスト工数を割り当てなかった場合，予測が完全に正しければ問題ないが，予測誤りを含む場合には，信頼性の低下を招くことになる．

従来，信頼性予測においては，相対誤差平均値や相対誤差中央値，Pred(25) 等の指標により，予測精度そのものの評価が行われてきたが，予測後に取るべきアクションや，アクションの結果としてどの程度の信頼性が確保できるのか，また，テスト工数の削減が見込めるのかについては，明らかでなかった．そのため，信頼性予測手法は数多く提案されているにも関わらず，その多くは開発現場において採用されていないのが現状である．

本論文の目的は，ソフトウェア開発管理における予測手法の採用において障害となっている，**問題 1** および**問題 2** を解決するため，予測手法を精度だけでなく

利用的側面から評価し，予測に必要なデータセットの基準や，見込まれる効果の度合いを明らかにする事である．これにより，開発現場における予測手法の採用の促進が期待される．

まず，**問題 1** の解決のために，本論文では，開発工数予測における，データセットに含まれる欠損値，プロジェクトの件数，メトリクスの数に着目する．開発工数予測においては，予測の失敗がそのまま開発プロジェクトの失敗に結びついてしまうため，予測において十分な精度を得ることが必須である．過去プロジェクトのデータセットの特徴として欠損値が多く含まれていることが挙げられる．欠損値は，プロジェクトの実施時期やプロジェクトの規模によって計測メトリクスに差異が生じていることなどを原因としてデータセットに含まれており，予測精度に大きな影響を与える [19]．また，データセットに含まれているプロジェクトの件数やメトリクスの数も予測精度に大きな影響を与える．そこで，十分な予測精度が得られる，データ欠損の割合の許容範囲や，必要なプロジェクト件数やメトリクス数を明らかにする必要がある．その結果，開発現場において予測を行う際に，手持ちのデータセットが予測に十分であるか，もしくは，何が不足しているかを知ることができる．

本論文では，特に，データセットに含まれる欠損値，プロジェクトの件数，メトリクスの数について，近年，工数予測手法として注目を浴びており，データセットの欠損値に強いと言われている類似性に基づく工数予測手法と従来一般的に用いられてきた予測手法である重回帰分析による工数予測手法において有効となる基準を明らかにする．

次に，**問題 2** の解決のために，本論文では，信頼性予測の 1 つである fault を含みやすいモジュールの判別（fault-prone モジュール判別）に着目する．開発コスト（テスト工数）とソフトウェアの信頼性はトレードオフの関係にあるが，最小のテスト工数で最大の信頼性を得ることを目的として，fault-prone モジュール判別が行われる．Fault-prone モジュール判別において fault を含みやすいモジュール（fault-prone モジュール）を判別し，fault-prone モジュールに多くのテスト工数を割り当てることで，より低コストでの高信頼性を目指す．そのため，fault-prone モジュール判別の結果に基づいてテスト工数を割り当てた場合，予測結果に基づ

かない場合と比べて，テスト工数がどれだけ変化し，それに伴って，開発コストがどの程度削減できるのか，あるいは増加してしまうのか．もしくは，発見される fault がどれだけ変化し，ソフトウェアの信頼性がどの程度向上するのか，あるいは低下するのか．といったことが求められる．従って，fault-prone モジュール判別を行うことで，どれだけの低コストでどれだけの高信頼性が得られるかを明らかにする必要がある．また，予測結果に基づくテスト工数の割り当て方についても従来議論されていないため，高い信頼性を得るためのテスト工数の割り当て方についても明らかにする必要がある．その結果，開発現場において新たに予測を行う際に，予測結果を開発プロジェクトに反映することが，開発プロジェクトにとって有効であるかを知ることができる．

本論文では，各モジュールの fault の有無，割り当てられるテスト工数，およびソフトウェアの信頼性（fault 発見率）の関係をモデル化し，様々な条件のもとでシミュレーションを行うことで判別精度とソフトウェアの信頼性の関係を明らかにする．

2. 論文構成

本論文の主要部分は大きく 3 つの章から構成される．第 2 章，第 3 章では，**問題 1** に対して，十分な予測精度を得るためのデータセットの条件について述べ，第 4 章では，**問題 2** に対して，予測手法導入の効果について述べる．

工数予測において十分な予測精度を得るためのデータセットの条件として，第 2 章では，データセットに含まれる欠損値に着目する．欠損値の割合と欠損メカニズムの 2 つの観点において予測精度との関係を明らかすることで，欠損値について十分な予測精度を得るために必要なデータセットの条件について明らかにする．また，第 3 章では，データセットの大きさに着目する．プロジェクト件数，および，メトリクス数について予測精度との関係を明らかにすることで，データセットの大きさについて十分な予測精度を得るために必要なデータセットの条件について明らかにする．

第 4 章では，fault-prone モジュール判別において判別結果を開発プロジェクト

に適用した際の予測手法導入の効果について述べる．各モジュールの fault の有無，割り当てられる工数，ソフトウェアの信頼性の関係をモデル化を行い，fault-prone モジュール判別導入の効果を明らかにする．

そして，本論文の最後の章である第 5 章で本論文全体のまとめを述べる．

第2章　工数予測手法の欠損値に対するロバスト性の評価

1. まえがき

ソフトウェア開発プロジェクトにおいて，工数の予測は，適切な資源の配置，および，スケジュール管理を行う上で非常に重要であり，多くの工数予測手法が提案されている [1][3][7][37][38]．これらの工数予測手法では，プロジェクトのメトリクス（規模，開発期間など）を説明変数，工数を目的変数とする予測モデルを過去プロジェクトの実績データに基づいて構築し，現行プロジェクトで計測したメトリクスをモデルに代入することで予測値を算出する．

予測モデルの構築は，多くの場合，欠損値（未記録の値）を含んだ実績データを用いて行うことが要求される．一般に，多数のプロジェクトから多種類のメトリクスを計測して得られるデータセットには，欠損値が数多く含まれるためである．例えば，情報処理推進機構ソフトウェア・エンジニアリング・センターが 15 社から収集した 1,009 件のプロジェクト，約 400 種類のメトリクスの実績データでは，全体の 87.7%が欠損値であり，記入を必須とした項目に限っても 36.9%が欠損値である [9]．また，１つの企業内で収集された 1,081 件のプロジェクト，14 種類のメトリクスの実績データにおいても，60%の欠損値を含んでいたことが報告されている [41]．海外においても，International Software Benchmarking Standards Group(ISBSG) において収集された，20ヶ国，3024 件のプロジェクト，99 種類のメトリクスの実績データは，57.6%の欠損値を含んでいる [13]．データ欠損の原因は様々であり，時間的制約や不注意による記録漏れ，組織内で統一的な取り決めがないためにプロジェクトごとに計測したメトリクスの種類が異なる，などが挙げられる．

欠損値を含むデータセットを用いて予測モデルを構築する１つの方法は，欠損値処理（MDT:Missing Data Techniques）を併用することである [19][27][39]．MDT とは，与えられたデータセットから欠損値を含むプロジェクトを除外したり，欠損値を何らかの値で補完したりすることによって，欠損値を含まないデータセットを作成する方法のことである．典型的な例として，欠損値を含むデータセットに対し，MDT を適用後，ステップワイズ重回帰分析により（回帰モデルを構築して）工数を予測することが行われている [39]．MDT は，データ全体に対する欠損値の割合（欠損率）が 30%以下で，かつ，欠損値が均一に生じている場合に有効である [19]．しかし，欠損率が 30%を超える場合，および，欠損値が均一に生じていない場合には予測精度が著しく低下することが報告されている [19]．現実には，データの欠損率が 30%を超えることも多く，欠損のメカニズムも多様であるため，MDT の適用範囲は限られる．

もう１つの方法は，欠損していない値のみを用いて予測を行うことであり，その具体的な手法として，協調フィルタリングに基づく工数予測手法 [31][41]（以降，類似性に基づく工数予測手法と呼ぶ）や Optimized Set Reduction (OSR) 法 [4] が提案されている．類似性に基づく工数予測手法は，約 60%の欠損値を含むデータセットを用いた事例において，MDT 適用後にステップワイズ重回帰分析を行った場合と比較して，予測値の相対誤差が，22.11 から 0.79 に改善されたことが示されている [41]．このことから，データの欠損率が 30%を超える場合の予測手法として有望であるといえる．しかし，適用事例はこの１つのみであり，欠損率や欠損値の分布が異なるデータセットに対しても高い予測精度が得られるかどうかは不明である．類似性に基づく工数予測手法を開発現場で採用するためには，欠損値の生じるメカニズムの違い，および，欠損率の変化が予測精度に与える影響（ロバスト性）を明らかにすることが望ましい．

本章では，欠損値が生じる 3 つのメカニズムを想定し，それぞれについて欠損率の異なるデータセット（欠損率 10%，20%，30%，$\cdots$）を用意し，各データセットを用いて工数予測を行うことで，類似性に基づく工数予測手法のロバスト性を実験的に評価する．評価にあたっては，MDT 適用後にステップワイズ重回帰分析を行った場合と比較する．これは，データ欠損を含むデータセットに対して一

般的に用いられ，予測手法の評価においても，比較対象としてしばしば用いられる手法である [25]．3 つの欠損メカニズムは，ソフトウェア工学分野において従来想定されている Missing Completely At Random (MCAR)，Missing At Random (MAR)，Nonignorable Missingness (NM) を用いる [22][39]（3.1 参照）．ただし，これらの欠損のメカニズムにそれぞれ合致し，かつ，欠損率 10%，20%，30%，… を満たすような評価用のデータセット群を，現実のプロジェクトから直接得ることは難しい．そこで，本章では，従来から使用されている欠損値を与える方法 [39] を用いて，それぞれの欠損メカニズム（MCAR，MAR，NM）に合うように意図的にデータを欠損させることで，評価用のデータセット群を作成することにした．ソフトウェア開発データでは，これらの欠損メカニズム以外に，バースト的に欠損値が生じていることが多いが，バースト的な欠損メカニズムは考案されていない．しかし，これらの欠損メカニズムは，ソフトウェア開発データで生じている欠損値の発生メカニズムの一部を表していると言える．本章の評価の方法は，欠損値を与える方法以外も Strike らの方法 [39] を踏襲している．ただし，Strike らの評価対象は本章とは異なり，MDT の各手法の性能である．Strike らは，欠損メカニズムと欠損率が異なるデータセット群を作成し，それらを用いて MDT を併用した重回帰分析を行い，MDT の各手法の性能比較をしている．

以降，2 で欠損値を考慮した工数予測手法を説明し，3 でロバスト性の評価方法について述べる．4 では実験の方法と手順について説明し，5 で実験結果と結果に対する考察を述べる．最後に 7 で本章の結論について述べる．

2. 欠損値を考慮した工数予測手法

2.1 類似性に基づく工数予測手法

協調フィルタリングは，多くの欠損値を含むデータセットを用いることを前提として，ユーザ間の類似性に基づいて各ユーザにアイテムの推薦を行うための基盤技術である [35][36]．類似性に基づく工数予測手法は，協調フィルタリングをソフトウェア開発分野での予測に応用し，プロジェクト間の類似性に基づいて工数予測を行う手法である [31][41]．類似したプロジェクトを用いて予測を行う手法

として，類似性に基づく工数予測手法の他に事例ベース推論（CBR）[37] があるが，欠損を含まないデータセットの使用を前提としているため，本章では評価の対象外とする．

類似性に基づく工数予測手法は 4 つの手順（ダミー変数化，標準化，類似度計算，予測値計算）から構成され，各手順で用いるアルゴリズムは複数の中から選択可能である．本章では，NCFE[29] を用いた予備実験で最も高い予測精度が得られたアルゴリズムを採用した．以降では，各手順の詳細と採用したアルゴリズムについて述べる（他のアルゴリズムについては [29] を参照されたい）．

手順 1．ダミー変数化 データセットに名義尺度のメトリクスが含まれる場合，カテゴリごとにダミー変数に置き換える．プロジェクト p_i のメトリクス m_j のカテゴリ k のダミー変数 $d_{ij}(k)$ は式 (2.1) で定義される．

$$d_{ij}(k) = \begin{cases} 1 & \cdots \quad \text{カテゴリ } k \text{ に属する} \\ 0 & \cdots \quad \text{カテゴリ } k \text{ に属さない} \end{cases} \tag{2.1}$$

手順 2．メトリクスの標準化 各メトリクスは値域に大きなばらつきがあるため，値域をそろえるための標準化を行う．本章では，標準化のアルゴリズムとしては Z-score[39] を用いた．Z-score では平均値が 0，分散が 1 となるように標準化され，データの分布が正規分布あるいは正規分布に近い場合 [-2,2] にデータの約 95%が含まれるように標準化する．本章の評価実験で用いたデータセット（基礎データセット（4.1 参照））では 97.9%が [-2,2] に含まれた．プロジェクト p_i のメトリクス m_j の値 v_{ij} を標準化した値 v'_{ij} は式 (2.2) で定義される．

$$v'_{ij} = \frac{v_{ij} - \mu_j}{\sigma_j} \tag{2.2}$$

ここで，μ_j はメトリクス m_j の平均値，σ_j はメトリクス m_j の標準偏差を表す．

手順 3．プロジェクト間の類似度計算 予測対象のプロジェクトと類似した他のプロジェクトを見つけるため，プロジェクト間の類似度を算出する．類似度

計算のアルゴリズムとしては，Cosine Similarity[36] を用いた．予測対象のプロジェクト p_a と他の各プロジェクト p_i との類似度 $sim(p_a, p_i)$ は式 (2.3) で定義される．

$$sim_{p_a,p_i} = \frac{\sum_{j \in M_a \cap M_i} v'_{aj} \times v'_{ij}}{\sqrt{\sum_{j \in M_a \cap M_i} {v'_{aj}}^2} \sqrt{\sum_{j \in M_a \cap M_i} {v'_{ij}}^2}} \tag{2.3}$$

ここで，M_a と M_i はそれぞれプロジェクト p_a と p_i に関して記録されている（欠損していない）メトリクスの集合を表す．

類似性に基づく工数予測手法では，類似度を求める 2 つのプロジェクトに共に記録されているメトリクスの集合を用いて類似度を算出するため，欠損値を含むデータセットに対しても適用できる．

手順 4. 類似度に基づく予測値の算出 類似したプロジェクトの実測値を用いて，予測対象のプロジェクトの目的変数（工数）を算出する．予測値算出のアルゴリズムとしては，Weighted Sum[36] を用いた．プロジェクト p_a の目的変数 m_b の予測値 $\hat{v}_{ab}$ は式 (2.4) で定義される．

$$\hat{v}_{ab} = \frac{\sum_{i \in k \text{ - } nearestProjects} (v_{ib} \times sim(p_a, p_i))}{\sum_{i \in k \text{ - } nearestProjects} sim(p_a, p_i)} \tag{2.4}$$

ここで，k-$nearestProjects$ は，メトリクス m_b が欠損しておらず，かつ，プロジェクト p_a と類似度の高い上位 k 個のプロジェクトの集合を表す．k の値は実験的に別途求める必要がある．

2.2 MDT とステップワイズ重回帰分析の併用

ステップワイズ重回帰分析

重回帰分析は多変量解析の一手法であり，ソフトウェア開発に要する工数を予測するために広く用いられており，予測手法の評価において，比較対照としてしばしば用いられる [25]．本章の評価実験においても，比較対象として重回帰分析の一手法であるステップワイズ重回帰分析を用いた．

重回帰分析では，予測対象の変数（目的変数） と，目的変数に影響を与える複数の変数（説明変数）との関係を表した一次式（回帰式）を作成する．回帰式中の各係数と定数は，予測値の絶対誤差（残差）の 2 乗和が最小になるように決定される．作成された回帰式に，現行プロジェクトで計測した説明変数を与えることで，目的変数を予測することが可能となる．

重回帰分析では，予測精度を向上させるために，多数の説明変数候補の中から，予測精度の向上に寄与すると予測される変数を選択して回帰式を作成する方法がとられる．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採用する変数を決定し，重回帰分析を行う手法である．ステップワイズ変数選択は次の手順で行われる．

手順 1. 変数を全く含まないモデルを初期モデルとして作成する．

手順 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（本章の評価実験では，偏 F 値の有意水準を $p_{in} = 0.05$, $p_{out} = 0.1$ とした）で棄却されない場合に変数を採択する．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 (VIF) が一定値（本章の評価実験では 10 とした）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 VIF が一定値以上となる場合，その変数は採択しない．

手順 3. 検定により適切な変数が選択されたと判断されるまで手順 2 を繰り返す．

欠損値処理（MDT:Missing Data Techniques）

欠損値を含むデータセットに対してステップワイズ重回帰分析を適用する場合には，MDT を併用する必要がある．MDT とは，多変量解析を可能とするために，与えられたデータセットから欠損値を含むプロジェクトを除外したり，欠損値を何らかの値で補完する，といった前処理を行う方法である．重回帰分析に対しては，次の 3 種類の MDT の手法が広く用いられる [19][39].

リストワイズ除去法　欠損値を 1 つでも含むプロジェクトを全て除去する．

ペアワイズ除去法　重回帰分析に特化した手法で，重回帰分析の過程においてメトリクス間の相関を求める際に，相関を求めるメトリクスのいずれかが欠損しているプロジェクトを除外して相関を求める [39].

平均値挿入法　欠損値に対して，当該メトリクスの平均値を挿入することで，欠損値を補完する．

これら 3 手法のうち，リストワイズ除去法が最も予測精度を低下させない手法である [39]．ただし，リストワイズ除去法は，欠損率が高い場合には全てのプロジェクトが除去され予測不可能となる．そこで，本章の評価実験においては，上記の 3 手法全てを用いてステップワイズ重回帰分析を行い，予測可能な手法の中で最も高い精度を示した手法を採用し，ステップワイズ重回帰分析の結果として評価に用いた．

3. ロバスト性の評価方法

本章では，Little ら [22] が定義した 3 種類（MCAR，MAR，NM）の欠損メカニズムを仮定し，それぞれについて欠損率の異なるデータセット（欠損率 10%, 20%, 30%, $\cdots$）を用意し，各データセットを用いて工数予測を行うことで，類似性に基づく工数予測手法のロバスト性を実験的に評価する．欠損値を含んだデータセットの構築は，Strike ら [39] の方法を踏襲する．

3.1 欠損メカニズム

3種類の欠損メカニズムの概要とデータ欠損の一例を次に示す.

Missing Completely At Random（MCAR）ある値が欠損値となる確率は，データ中の他の値に依存しない．すなわち，ランダムに欠損値が生じる．例えば，時間的制約や不注意によってメトリクスの記録漏れが生じ欠損値となったとする．この場合，データ中の他の値に依存しないで欠損しているため，欠損メカニズムはMCARである.

Missing At Random（MAR）ある値が欠損値となる確率は，ある変数（メトリクス）の値の大きさに依存する．例えば，ソフトウェアの規模をあらわす尺度（ファンクションポイントなど）とコードレビューにおける発見バグ数が変数として含まれるデータを考える．規模が小さなプロジェクトほどコードレビューが省略されやすく発見バグ数が欠損値となりやすいとする．この場合，コードレビューにおける発見バグ数が欠損値となる確率は規模の大きさに依存しているため，欠損メカニズムはMARである.

Non-ignorable Missingness（NM）ある値が欠損値となる確率は，その値自体の大きさに依存する．例えば，ソフトウェアの規模を表す尺度が変数として含まれるデータを考える．規模が小さいプロジェクトほど，人員的余裕がないため規模自体が記録されにくく，欠損値となりやすいとする．この場合，規模が欠損値となる確率は規模自体の大きさに依存しているため，欠損メカニズムはNMである.

これらの欠損メカニズムは，元来，統計の分野で考案されたものであるが，ソフトウェア開発データの欠損を表す手段としても用いられている [39].

3.2 実験データセット作成手順

指定された欠損率に基づいて，実験データセットを作成する手順（基礎データセットに欠損を与える手順）を，欠損メカニズムごとに示す.

MCAR

手順 1 全プロジェクトから，欠損値を与えるプロジェクトをランダムに選択する．

手順 2 手順 1 で選択したプロジェクトで欠損していないメトリクスの中から，1 つをランダムに選択する．ただし，予測対象のメトリクス（目的変数）は選択対象としない．

手順 3 手順 1 で選択したプロジェクトの，手順 2 で選択したメトリクスを欠損させる．

手順 4 指定された欠損率に達するまで手順 1〜3 を繰り返す．

MAR

手順 1 依存変数の値が小さい順にプロジェクトをソートし，ソートしたプロジェクトを先頭から 5 つのグループに均等に分ける（図 2.1）．本章では Strike ら [39] と同様に依存変数として開発規模を用いた．すなわち，開発規模が小さなプロジェクトほど，データ収集の必要性が低かったり，データ収集に多くの時間を割くことができずデータに欠損が生じやすいというモデルを仮定している．

手順 2 上位 4 グループについて，各グループ内の欠損値の量が，上位のグループから順に，総欠損値数（指定した欠損率に達するのに必要な欠損値の個数）の 40%，30%，20%，10%になるまで手順 3〜5 を繰り返す（図 2.1）．

手順 3 グループ内のプロジェクトから，欠損値を与えるプロジェクトをランダムで選択する．

手順 4 手順 3 で選択したプロジェクトで欠損していないメトリクスの中から，1 つをランダムで選択する．ただし，目的変数と依存変数は選択対象としない．

手順 5 手順 3 で選択したプロジェクトの，手順 4 で選択したメトリクスを欠損させる．

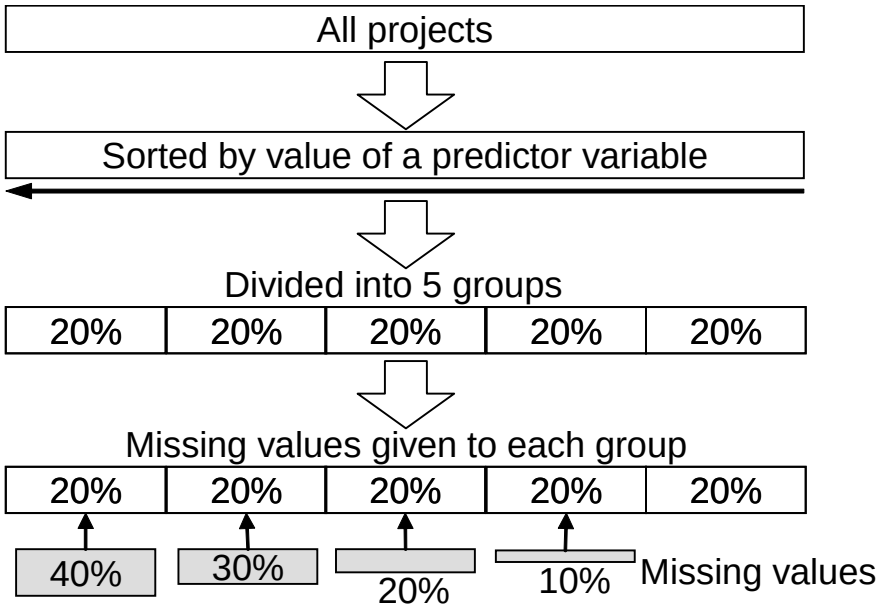

図 2.1 MAR と NM の欠損値生成手順

NM

手順 1 目的変数以外の全変数の中から 1 つを選択する.

手順 2 選択された変数の値が小さい順にプロジェクトをソートし，ソートしたプロジェクトを先頭から 5 つのグループに均等に分ける（図 2.1）.

手順 3 上位 4 グループについて，各グループ内の当該メトリクスについての欠損値の量が，上位のグループから順に，総欠損値数の 40%，30%，20%，10%になるまで手順 4，手順 5 を繰り返す.

手順 4 グループ内のプロジェクトから，欠損値を与えるプロジェクトをランダムに選択する.

手順 5 手順 4 で選択したプロジェクトの当該メトリクスを欠損させる.

手順 6 全変数について手順 1〜5 により欠損値を与える.

上記手順の MAR および NM において，全プロジェクトの 20%に対して総欠損値数の 40%の欠損値を与える必要があるため，欠損率 50%以下でしか欠損値を与えることはできない.

どの変数がどの欠損メカニズムに従うかは，データ計測体制やソフトウェア開発方法論に依存すると考えられるため，一意に決定することは難しい．そこで，本章では，欠損メカニズムごとにデータセットを作成することとし，各データセットにおいて，目的変数と依存変数以外の全変数が当該メカニズムに従うと仮定した [39].

3.3 ロバスト性の評価尺度

本章では，データの欠損率，及び，欠損メカニズムの違い，のそれぞれに関して，類似性に基づく工数予測手法とステップワイズ重回帰分析のロバスト性を比較する．

ある予測手法が「欠損率に関してロバストである」とは，予測モデルを作成するためのデータ（以降フィットデータと呼ぶ）の欠損率を増大させた場合に，予測精度を評価するためのデータ（以降テストデータと呼ぶ）の予測精度が大きく低下しないことと考える．本章では，欠損率に関するロバスト性の評価尺度として，フィットデータの欠損率を 0%から 50%へと増大させたときの予測精度（絶対誤差，相対誤差）の変化率を用いる．変化率が小さいほど，欠損率に関してロバスト性が高いとみなす．欠損率を 50%とした理由は，欠損メカニズムのうち MAR および NM では，欠損率 50%以下でしか欠損値を与えることができないためである．

また，ある予測手法が「欠損メカニズムの違いに関してロバストである」とは，異なる欠損メカニズムを用いて作成されたフィットデータが複数存在する場合に，どのフィットデータを用いてもテストデータの予測精度に大差がないと考える．本章では，欠損メカニズムの違いに関するロバスト性の評価尺度として，欠損率 50%のフィットデータを 3 つの欠損メカニズムをそれぞれ用いて作成し，それらのフィットデータを使って予測を行ったときの，最良の予測精度と最低の予測精度の差を用いる．この差が小さいほど，欠損メカニズムの違いに関してロバスト性が高いとみなす．

4. 評価実験

4.1 基礎データセット

実験で利用した基礎データセットは，プロジェクト件数 140 件，メトリクス数 10 個の，欠損値を含まないデータセットである．10 個のメトリクスの名称，および，各メトリクスの統計量を表 2.1 に示す．これらのメトリクスのうち試験工数を目的変数とし，試験工数以外のメトリクスを説明変数の候補とした．基礎データセットは，企業のソフトウェア開発で得られたプロジェクトのデータセットから欠損値を含まない部分を取り出したデータセットである．基礎データに含まれるプロジェクトは，新規開発，機能拡張，パッケージのカスタマイズなど様々な開発形態のプロジェクトが含まれており，実施期間が数週間から数年までのプロジェクトが含まれている．また，守秘義務契約により，メトリクスごとに何らかの定数であらかじめ除算された値が企業から提供されている．

4.2 評価基準

評価基準として，絶対誤差，相対誤差のそれぞれの中央値を用いて評価を行った．各評価基準は値が小さいほど予測精度が高いことを表す．

それぞれの評価基準は次の式 (2.5)〜(2.6) で計算される．ここで，M 件のプロジェクトがあるとする．また，実測値と予測値をそれぞれ X_i，$\hat{X}_i (i = 1 \sim M)$ とし，$A_i = \left|\hat{X}_i - X_i\right|$, $R_i = \frac{\left|\hat{X}_i - X_i\right|}{X_i}$ とおく．

絶対誤差中央値

表 2.1 実験データセットの各メトリクスの統計量

	設計工数（総計）	設計工数（自社）	設計工数（SES)	設計工数（請負）
平均値	1.15	0.13	0.7	0.01
中央値	0.09	0.02	0.04	0
分散	47.5	0.61	25.8	0.002
最大値	67.5	8.72	58.7	0.42
最小値	0.001	0	0	0
	製造工数（総計）	製造工数（自社）	製造工数（SES)	製造工数（請負）
平均値	2.06	0.03	0.12	1.51
中央値	0.13	0.003	0.03	0
分散	175	0.01	0.07	152
最大値	144	0.98	1.59	144
最小値	0.001	0	0	0
	開発規模	試験工数		
平均値	233	1.45		
中央値	43.9	0.07		
分散	1×10^6	101		
最大値	1×10^4	112		
最小値	0.4	0.001		

値はあらかじめメトリクスごとに何らかの定数で除算されている

$$
\text{絶対誤差中央値} = \begin{cases} A_n & M = \text{奇数}\,(2n-1) \\ & (A_1 \leq A_2 \leq \cdots \leq A_n \leq \cdots \leq A_{2n-1}) \\ \dfrac{A_n + A_{n+1}}{2} & M = \text{偶数}\,(2n) \\ & (A_1 \leq \cdots \leq A_n \leq A_{n+1} \leq \cdots \leq A_{2n}) \end{cases} \tag{2.5}
$$

相対誤差中央値

$$
\text{相対誤差中央値} = \begin{cases} R_n & M = \text{奇数}\,(2n-1) \\ & (R_1 \leq R_2 \leq \cdots \leq R_n \leq \cdots \leq R_{2n-1}) \\ \dfrac{R_n + R_{n+1}}{2} & M = \text{偶数}\,(2n) \\ & (R_1 \leq \cdots \leq R_n \leq R_{n+1} \leq \cdots \leq R_{2n}) \end{cases} \tag{2.6}
$$

4.3 実験手順

実験手順は次の通りである（図 2.2）.

1. 4.1 で述べた基礎データセットを 10 回無作為に 2 等分し，予測モデルを作成するためのデータ（フィットデータ）と予測精度を評価するためのデータ（テストデータ）の組を 10 組作成した.

2. 作成した 10 個のフィットデータに対して 3.1 で述べた方法を用いて欠損値を与えた．MCAR で欠損率 0〜90%，MAR と NM で欠損率 0〜50%の範囲

で，10%刻みで欠損値を与え，テストデータ 1 個に対して 22 個のフィットデータ，計 220 個のフィットデータを作成した．

3. 類似性に基づく工数予測手法のパラメータである類似プロジェクト件数 (式 (2.4) の $k - nearrestProjects$) の最適な値を決定した．類似プロジェクト件数 k は，予備実験で，最も高い予測精度が得られた $k = 14$ を採用した．

4. （3）で決定した類似プロジェクト件数を使い，(2）で作成したフィットデータとテストデータの組に対して 2.1 で述べた類似性に基づく工数予測手法により予測を行い，予測値の絶対誤差と相対誤差を求めた．

5. （2）で作成したフィットデータとテストデータの組に対して，2.2 で述べた MDT を併用したステップワイズ重回帰分析により予測を行い，予測値の絶対誤差と相対誤差を求めた．3 種類の MDT の結果のうち，最も予測誤差が小さい結果をステップワイズ重回帰分析の結果として採用した．採用した結果において，ステップワイズ変数選択で各メトリクスが選択された割合を表 2.2 に示す．

6. （4）, (5）で得られた結果を比較し，類似性に基づく工数予測手法のロバスト性を確認した．

5. 結果と考察

5.1 欠損率に関するロバスト性

工数予測結果として，データ欠損率が 0%と 50%の時の絶対誤差，相対誤差の中央値を表 2.3 に示す．また，表 2.3 の右端の欄には，欠損率が 0%から 50%へと増大したときのそれら誤差の中央値の変化率を示す．ここで誤差の評価に平均値でなく中央値を用いた理由は，誤差の大きな少数のプロジェクトが存在し，誤差の分布が正規分布となっていなかったためである．データ欠損のない場合の予測精度の評価は本章の対象外ではあるが，表 2.3 に示されるように，データ欠損率

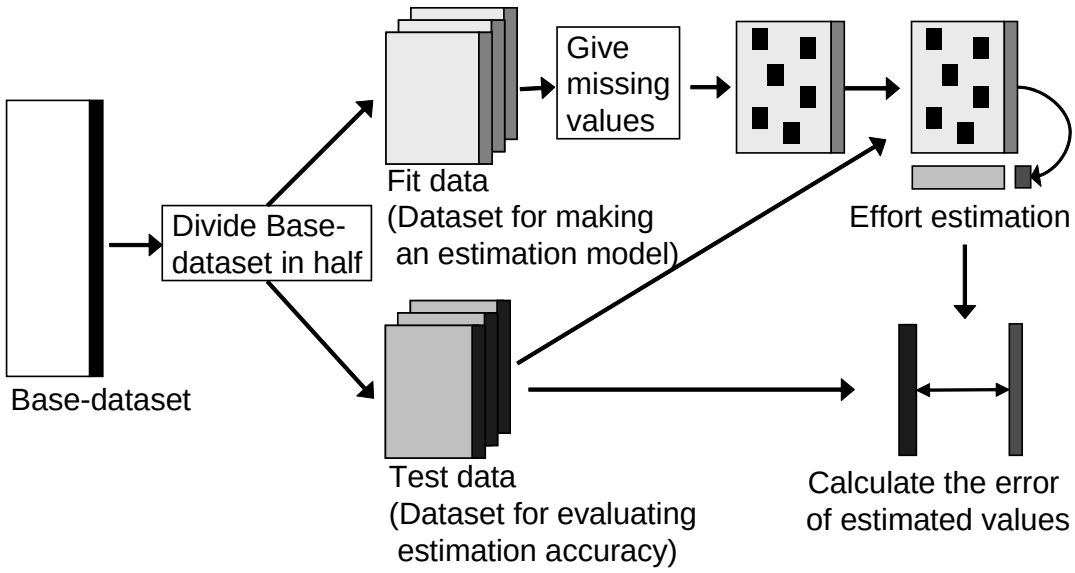

図 2.2 実験におけるデータ加工

表 2.2 変数選択率

メトリクス		絶対誤差			相対誤差		
		MCAR	MAR	NM	MCAR	MAR	NM
設計工数	(総計)	7.00%	49.50%	48.70%	6.50%	49.50%	48.70%
	(自社)	13.30%	42.00%	38.20%	12.80%	42.00%	38.20%
	(SES)	12.30%	41.90%	39.70%	11.50%	41.90%	39.70%
	(請負)	7.00%	0%	0%	6.70%	0%	0%
製造工数	(総計)	7.90%	62.10%	38.30%	7.90%	62.10%	38.30%
	(自社)	0%	35.70%	23.50%	0%	35.70%	23.50%
	(SES)	52.40%	45.30%	43.00%	50.40%	45.30%	43.00%
	(請負)	0%	36.30%	23.20%	0%	36.30%	23.20%
開発規模		5.00%	1.50%	39.30%	5.00%	1.50%	39.30%

0%では，類似性に基づく工数予測手法の方がステップワイズ重回帰分析よりも絶対誤差，相対誤差ともに小さかった．

表 2.3 において，データ欠損率を 0%から 50%へ増大させたときの絶対誤差の変化率に着目すると，類似性に基づく工数予測手法では変化率が 7.8%〜28.1%であるのに対し，ステップワイズ重回帰分析では 107.5%〜905.7%となっており，14〜48 倍大きな値を取っている．このことから，予測値の絶対誤差について，類似性に基づく工数予測手法は欠損率の変化に関してよりロバストであるといえる．

相対誤差についても，同様に，データ欠損率を 50%へ増大させたときの誤差の変化率に着目すると，類似性に基づく工数予測手法では変化率が 1.0%〜6.8%であるのに対し，ステップワイズ重回帰分析では 43.0%〜743.8%となっており，33〜109 倍大きな値を取っている．このことから，相対誤差についても，類似性に基づく工数予測手法は欠損率の変化に関してよりロバストであるといえる．

それぞれの欠損メカニズムについて，欠損率を 10%刻みで変化させたときの誤差の分布を示す箱ひげ図を図 2.3〜2.5 に示す．グラフの縦軸は予測誤差を，横軸は欠損率を示し，箱の下端は第 1 四分位，上端は第 3 四分位，箱中の横線は中央値，線分（ひげ）の下端の横線は最小値を表す．グラフの上方を省略しているため最大値は示されていない．

図 2.3〜2.5 において，ステップワイズ重回帰分析の結果は，MCAR では欠損率 30%以上で予測精度が低下しており，MAR では欠損率 40%以上で，NM では欠損率 50%で予測精度が低下している．一方，類似性に基づく工数予測手法では，欠損率 60%以下では予測精度は大きく低下していない．ステップワイズ重回帰分析と比較すると，高い精度で予測可能な欠損率の範囲が大きく，許容される欠損率の範囲という観点からも，ロバスト性は高いといえる．ただし，類似性に基づく工数予測手法も，欠損率が 70%以上では予測精度が大きく低下しており，予測を行うべきではないといえる．

また，図 2.3〜2.5 の箱の大きさ（第 1 四分位と第 3 四分位の差）に着目すると，類似性に基づく工数予測手法の方が小さく，ばらつきが小さいといえる．ステップワイズ重回帰分析は，特に MCAR において，欠損率の増大に伴って誤差のばらつきが著しく増大した．

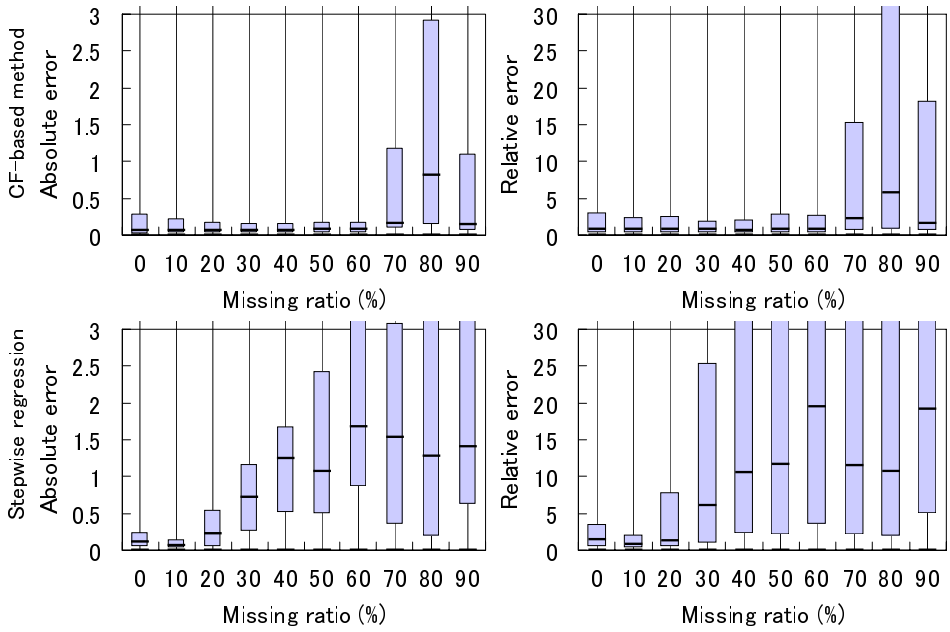

図 2.3 各欠損率における予測精度 (MCAR)

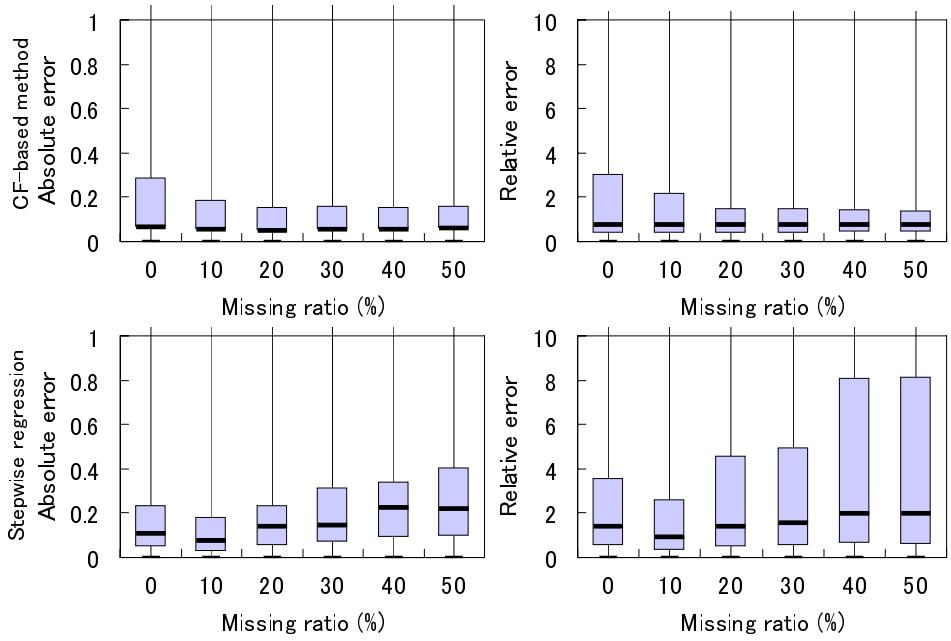

図 2.4 各欠損率における予測精度 (MAR)

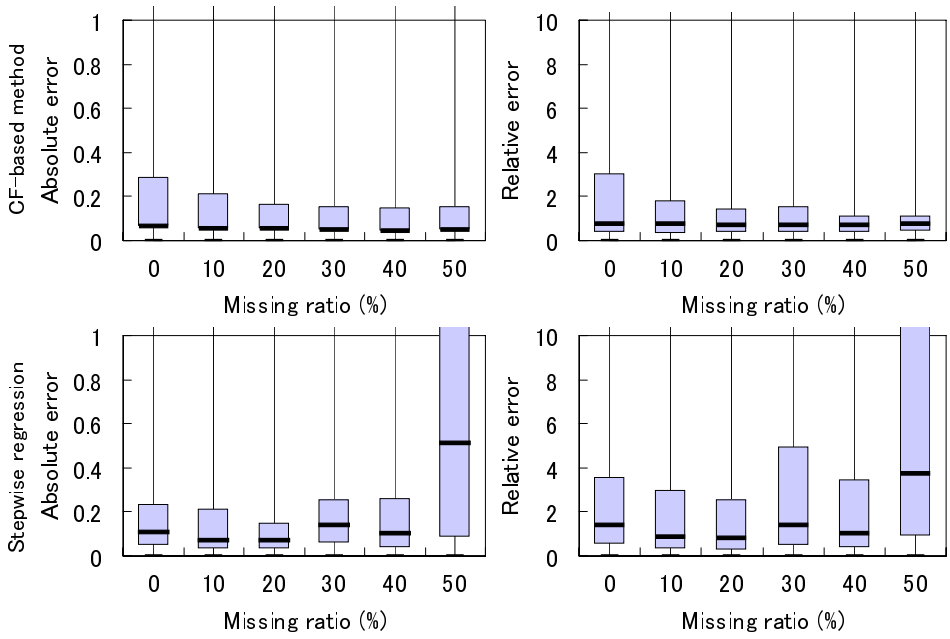

図 2.5 各欠損率における予測精度 (NM)

5.2 欠損メカニズムの違いに関するロバスト性

表 2.3 において，欠損率 50%の絶対誤差に着目すると，類似性に基づく工数予測手法では，絶対誤差中央値の最大が MCAR の 0.076，最小が NM の 0.048 であり，その差は 0.028 であったのに対し，ステップワイズ重回帰分析では最大が MCAR の 1.066，最小が MCAR の 0.220 であり，その差は 0.846 となり，これは類似性に基づく工数予測手法と比べて約 31 倍大きい．このことから，予測値の絶対誤差について，類似性に基づく工数予測手法は欠損メカニズムの違いに関してよりロバストであるといえる．

相対誤差についても，同様に，類似性に基づく工数予測手法では，最大が MAR の 0.762，最小が MCAR の 0.718 であり，その差は 0.044 であったのに対し，ステップワイズ重回帰分析では最大が MCAR の 11.67，最小が MAR の 1.977 であり，その差は 9.7 となり，これは類似性に基づく工数予測手法と比べて約 220 倍大きい．このことから，相対誤差についても，類似性に基づく工数予測手法は欠損メカニズムの違いに関してよりロバストであるといえる．

表 2.3 予測誤差の中央値

欠損率			0%	50%	変化率 (%)
絶対誤差	類似性に基づく手法	MCAR	0.064	0.076	18.8
		MAR		0.059	7.8
		NM		0.048	28.1
	SW 重回帰分析	MCAR	0.106	1.066	905.7
		MAR		0.220	107.5
		NM		0.510	381.1
相対誤差	類似性に基づく手法	MCAR	0.770	0.718	6.8
		MAR		0.762	1.0
		NM		0.757	5.1
	SW 重回帰分析	MCAR	1.383	11.67	743.8
		MAR		1.977	43.0
		NM		3.731	169.8

$$\text{変化率} = \frac{|50\% - 0\%|}{0\%}$$

また，欠損メカニズム，欠損率，及び，予測誤差の関係を一つの図で表したものを図 2.6 に示す．グラフの縦軸は予測誤差を，横軸は欠損メカニズムと欠損率を示す．図 2.6 の箱ひげ図の中央値に着目すると，類似性に基づく工数予測手法は欠損値の大小に関わらず，欠損メカニズムの違いによる誤差のばらつきは小さい．一方，重回帰分析では，欠損率が 20%を超えると欠損メカニズムの違いによる誤差のばらつきが大きくなり，特に，欠損値が均一に生じている MCAR と均一に生じていない MAR，NM の間で差が大きくなった．このことからも，類似性に基づく工数予測手法は，より広い欠損率の範囲において，欠損メカニズムの違いに関してよりロバストであるといえる．

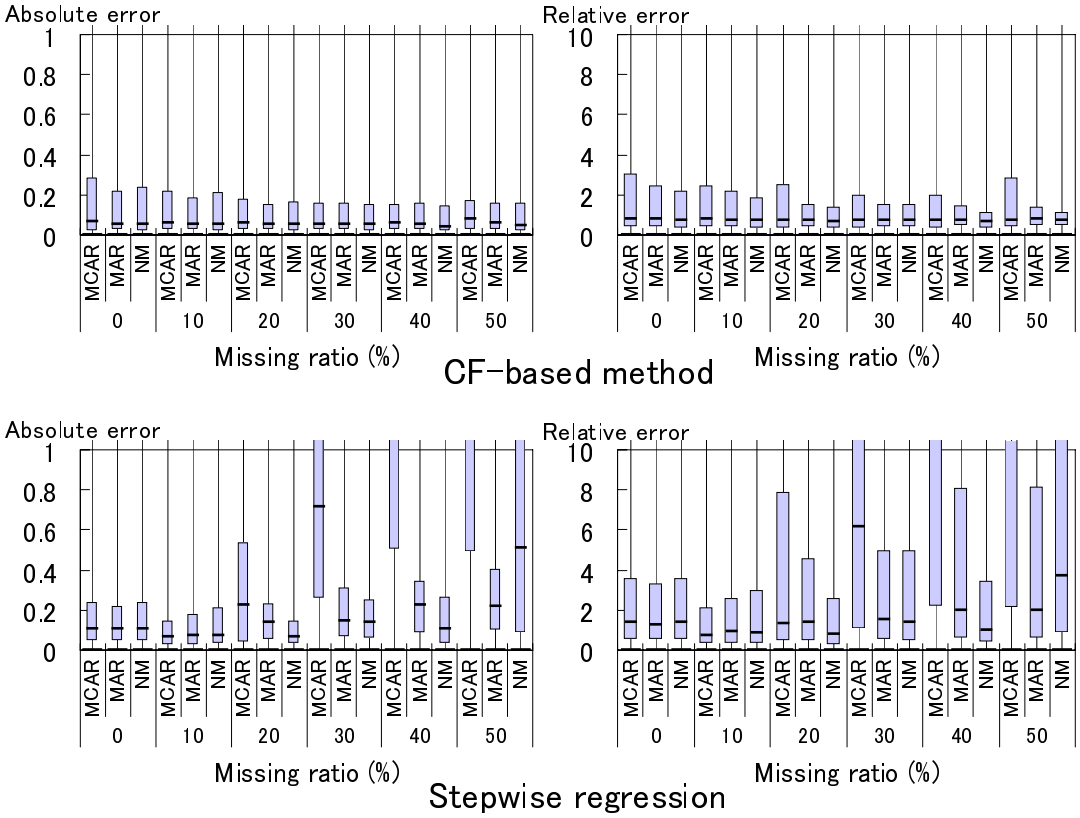

図 2.6 各欠損メカニズムの予測精度

6. 関連研究

工数予測手法の精度と欠損値の関係を調査した研究としては，Kromrey ら [19]，Li ら [20] の研究がある．Kromrey ら [19] は，重回帰分析の予測精度について，欠損率，及び，欠損値の均一さとの関係を明らかにし，欠損率 30%以下かつ，欠損値が均一（本章の評価実験では MCAR が該当する）に生じている場合に十分な予測精度が得られると結論付けている．Kromrey らはソフトウェア開発プロジェクトで計測，収集されたデータセットを用いてはいないが，本章のステップワイズ重回帰分析の結果においてもほぼ同様の結果が得られており，Kromrey らの結果をソフトウェア開発データにおいて補強するものである．また，Li ら [20] は，工数予測手法 AQUA について，複数のデータセットにおいて予測精度と欠損率の関係を明らかにし，AQUA では，欠損率 40%以下において十分な予測精度が得

られると結論付けている．しかし，Li らはデータセットに欠損値を与える際にランダム（本章の MCAR に該当）にしか与えておらず，欠損メカニズムについては考慮していない．そのため，本章で用いた MAR や NM のような欠損メカニズムにおいて AQUA がどの程度の欠損率において十分な予測精度が得られるかは明らかではない．

工数予測手法以外で予測精度と欠損値の関係を調査した研究としては，Strike ら [39] らの研究がある．Strike ら [39] は，本章の評価実験も用いたリストワイズ除去法，ペアワイズ除去法，平均値挿入法などの欠損値処理について，十分な予測精度が得られる欠損値の割合と欠損メカニズムを明らかにしている．対象が欠損値処理であり，本論文で対象とする工数予測手法とは異なるが，十分な予測精度を得るために必要なデータセットの基準を明らかにするという本論文の目的と合致している．本章の評価実験では，最も高い精度が得られた欠損値処理をステップワイズ重回帰分析の結果として採用したが，欠損値処理に関して，Strike らの結果を採用することも 1 つの手段である．

7. まとめ

本章では，協調フィルタリングに基づく工数予測手法（類似性に基づく工数予測手法）のデータの欠損に対するロバスト性を評価した．評価においては，欠損率に関するロバスト性と欠損メカニズム（MCAR，MAR，NM）の違いに関するロバスト性の 2 つの観点から評価を行った．140 件のプロジェクト，10 種類のメトリクスの実績値データを用いた評価実験の結果，類似性に基づく工数予測手法は MDT を併用したステップワイズ重回帰分析よりも，欠損率に関しても，欠損メカニズムの違いに関してもロバストであった．

ただし，類似性に基づく工数予測手法でも，欠損率 70%以上では予測精度が大きく低下した．従って，類似性に基づく工数予測手法を用いて十分な予測精度を得るためには，欠損率 70%未満のデータセットを用意すべきであると言える．また，本章では，全変数が同一の欠損メカニズムに従うと仮定してロバスト性の評価実験を行ったが，実際には，変数ごとに従う欠損メカニズムは異なると考えら

れる．そこで，３種類の欠損メカニズムを組み合わせてデータ欠損させたデータセットによるロバスト性の評価が今後の課題として挙げられる．

第3章　工数予測手法におけるデータ数と予測精度の関係の分析

1．はじめに

ソフトウェア開発プロジェクトにおいて，工数を予測するための手法は数多く提案されている．工数の予測は，過去のプロジェクトで計測されたプロジェクトのメトリクスを用いて数学的な予測モデルを作成し，現行のプロジェクトで計測されたメトリクスを作成されたモデルに代入することで予測値を算出することで行われる．モデルを作成するために用いるプロジェクトのメトリクスは，ソフトウェアメトリクス (または，単にメトリクス) と呼ばれる尺度のもとで計測，数値化される．

工数を予測する 1 つの手段は，COCOMO[3] や SLIM[33] などの汎用的な（あらかじめ定義された）予測モデルを利用することである．例えば，COCOMO では，ソフトウェアの規模（ソースコード行数）の予測値から開発工数，期間，要員数，生産性の予測値をそれぞれ算出することができる．規模を与えるだけで容易に予測が行えるので，従来，COCOMO は多くの企業で用いられてきた．さらに，COCOMO を拡張した COCOMO II では，予測規模としてファンクションポイントを採用し，コストドライバと呼ばれるプロジェクトのメトリクスを与えることで，より正確な工数予測が行える．ただし，ソフトウェアの開発プロセスや開発手法は急速に多様化しており，これらのあらかじめ定義されたモデルでは開発組織やプロジェクトの個別性を十分に反映できず，高精度な予測を行うことが困難になってきている [6].

工数を予測するもう 1 つの手段は，重回帰分析やニューラルネット [38] などを用い，自社の過去プロジェクトのデータに特化した工数予測モデルを構築するこ

とである．過去プロジェクトのデータには，完了したプロジェクトごとに工数や開発規模，検出バグ数などのメトリクスが記録，蓄積されている．過去プロジェクトにより多くのプロジェクトを含むほど，また，より多種類のメトリクスを含むほど，より高い予測精度が得られることが期待される [27]．ただし，これらの予測モデルでは，多様化するプロジェクトのメトリクスと工数との関係をただ１つのモデル式により表現するという点では COCOMO などのあらかじめ定義されたモデルと同じであり，個別性の高いプロジェクトでは十分な予測精度を得ることが難しい．

近年，第三の手段として，プロジェクト間の類似性に基づく予測方法が注目されており，事例ベース推論 (CBR)[37]，および，協調フィルタリングに基づく手法 [31][41] が提案されている．これらの手法は，重回帰分析と同様に，過去プロジェクトのデータを必要とするが，データ全体に対してただ１つの予測モデルを構築するのではなく，予測対象プロジェクトごとに個別に予測モデルを構築するため，プロジェクトの個別性をより強く反映した予測が行える．一般に，ソフトウェア開発現場では，経験に頼った予測方法として，過去に手がけた類似の案件の実績データを元に，新規プロジェクトの工数を予測場合がある（「類推見積もり」などとも呼ばれている）が，CBR や協調フィルタリングに基づく手法は，このような経験に頼った予測を，系統的に行う方法である．

類似性に基づく工数予測手法では，予測対象プロジェクトと類似する過去プロジェクトが存在する場合に高い精度で予測が行える．そのため，過去プロジェクトの数が多いほど類似プロジェクトが見つかる可能性が高くなり，予測精度が向上することが推測される．また，より多数のメトリクスが含まれるほど，類似プロジェクトであるか否かの判別が正確に行えることとなり，予測精度が向上すると期待できる．ただし，どの程度の数のプロジェクトやメトリクスを用いて予測を行えば高い精度で予測が行えるかは従来明らかとされていない．

本章では，類似性に基づく工数予測手法のプロジェクト件数，及びメトリクス数と予測精度の関係を実験的に評価し明らかにする．実験は，含まれるプロジェクト件数とメトリクス数が異なる過去プロジェクトのデータを用いて，実測値が既知である複数のプロジェクトに対して予測を行った．そして，得られた予測値

と実測値の誤差から予測精度の関係を調べた.

以降，2 で実験について説明した後，3 で実験の結果とその考察について述べる．最後に 5 でまとめを述べる.

2. 実験

2.1 実験で利用したデータ

実験では 2 章と同様に 2.1 節のデータを用いた.

2.2 評価基準

評価基準として，絶対誤差，相対誤差のそれぞれの平均値，中央値の四種類の誤差を用いて評価を行った．各評価基準は値が小さいほど予測精度が高いことを表す.

平均値に関する評価基準は次の式 (3.1)〜(3.2) で計算される．(中央値に関する評価基準については 2 章 4.2 参照のこと．) ここで，M 件のプロジェクトがあるとする．また，実測値と予測値をそれぞれ X_i，$\hat{X}_i (i = 1〜M)$ とし，$A_i = \left|\hat{X}_i - X_i\right|$, $R_i = \frac{\left|\hat{X}_i - X_i\right|}{X_i}$ とおく.

絶対誤差平均値

$$\text{絶対誤差平均値} = \frac{\sum_{i=1}^{M} A_i}{M} \tag{3.1}$$

相対誤差平均値

$$\text{相対誤差平均値} = \frac{\sum_{i=1}^{M} R_i}{M} \tag{3.2}$$

2.3 実験手順

実験は，プロジェクト件数を変化させた実験と，メトリクス数を変化させた実験の二種類の実験を行った．それぞれの実験は次の手順で行った．

- プロジェクト件数の変化

  1. 2.1 で述べた基礎データを無作為に 70 件ずつに二等分し，予測モデルを作成するデータ（以降フィットデータと呼ぶ）とフィットデータを用いて実際に予測を行うデータ（以降テストデータと呼ぶ）とした．
  2. プロジェクト件数が異なるデータを作成するために，フィットデータから無作為に 10 回 n 個のプロジェクトを抽出し新たなフィットデータを作成した．n は 2〜69 まで変化させた．
  3. 全てのフィットデータを使って一つのテストデータに対して類似性に基づく工数予測手法により予測を行い，各評価基準を求めた．予測対象のメトリクスを試験工数とし，テストデータの試験工数は未知数として予測を行った．予測時に式 (2.4) の k - $nearestProjects$ は 14 としているが，プロジェクト件数が 14 以下のフィットデータの場合には，k - $nearestProjects$ の値はプロジェクト件数として予測を行った．
  4. ステップワイズ重回帰分析でも同様に予測を行った．

- メトリクス数の変化

  1. 2.1 で述べた基礎データを，時系列を考慮し，開発規模，設計工数，設計工数 (自社)，設計工数 (SES)，設計工数 (請負)，製造工数，製造工

数 (自社)，製造工数 (SES)，製造工数 (請負) の順でメトリクスを増加させ，メトリクス数が異なるデータを作成した．

2. 作成したメトリクス数が異なるデータを無作為に均等に 70 件ずつに二等分しフィットデータとテストデータとし，新たなフィットデータとテストデータの組を作成した．
3. 全てのフィットデータとテストデータの組に対して類似性に基づく工数予測手法により予測を行い，各評価基準を求めた．予測対象のメトリクスを試験工数とし，テストデータの試験工数は未知数として予測を行った．
4. ステップワイズ重回帰分析でも同様に予測を行った．

実験におけるデータ加工の手順を示した図を図 3.1 と図 3.2 に示す．

3. 実験の結果と考察

3.1 プロジェクト件数の変化

プロジェクト件数を変化させた時の類似性に基づく工数予測手法とステップワイズ重回帰分析の各評価基準の予測精度の変化を図 3.3 に示す．各グラフの横軸はプロジェクト件数を，縦軸は評価基準の値を示す．

各評価基準で共通して，類似性に基づく工数予測手法では，プロジェクト件数の増加に伴って，徐々に予測精度が向上し，また，精度の変動幅も小さくなっている．ステップワイズ重回帰分析では，あるプロジェクト件数で予測精度が急激に向上し，しばらく向上が続くが，その後，徐々に予測精度が低下している．また，絶対誤差平均値を除いて，プロジェクト件数が約 50 件を超えるとステップワイズ重回帰分析よりも高い予測精度が得られている．

この結果より，類似性に基づく工数予測手法では，プロジェクト件数が少ない時には似ているプロジェクトも少ないために予測精度が低いが，プロジェクト件数の増加に伴って似ているプロジェクトも多くなるため予測精度が向上すること

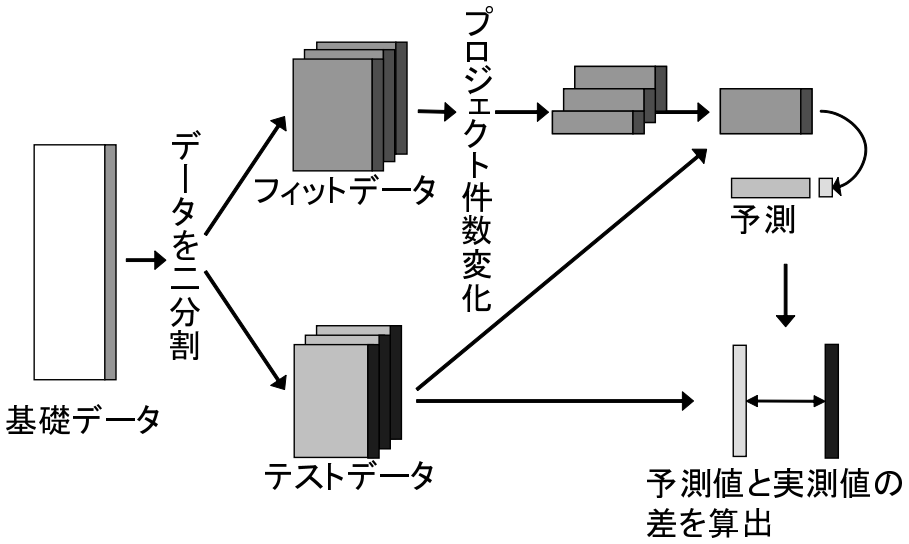

図 3.1 プロジェクト件数の変化の実験におけるデータの加工手順

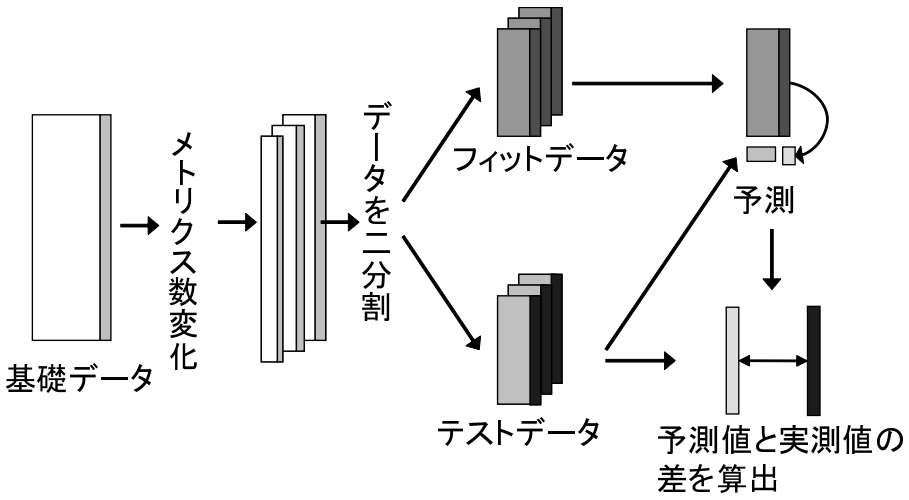

図 3.2 メトリクス数の変化の実験におけるデータの加工手順

を示している．ステップワイズ重回帰分析では，予測精度の変化が向上から低下へと変わった時点で最適な予測モデルが作成されているが，その後は，プロジェクトが増加することで最適な予測モデルから外れていくことがわかる．

絶対誤差平均値において，類似性に基づく工数予測手法の予測精度がステップワイズ重回帰分析の予測精度よりも悪い結果となっている．これは，ステップワイズ重回帰分析が絶対誤差平均値が最小となるように予測モデルを作成し，また，大きな値のデータに大きく影響を受けるのに対して，類似性に基づく工数予測手法では誤差を最小にするようなモデルを作成するわけではないため，大きな値のデータを予測する場合には絶対誤差が必然的に大きくなっているためだと考えられる．これは，絶対誤差平均値以外では，プロジェクト件数が多い場合にはステップワイズ重回帰分析よりも高い予測精度を示していることからも推測される．

絶対誤差，相対誤差ともに，平均値と中央値を比較すると中央値の予測結果の方が高い予測精度が得られている．これは，予測結果中に予測精度が著しく低い結果が一部含まれているために，平均値が中央値よりも悪くなっていると考えられる．

3.2 メトリクス数の変化

メトリクス数を変化させた時の類似性に基づく工数予測手法とステップワイズ重回帰分析の予測精度の変化を図 3.4 に示す．各グラフの横軸はメトリクス数を，縦軸は評価基準を示す．

絶対誤差平均値を除いて，類似性に基づく工数予測手法では，メトリクス数の増加に伴って多くの場合に予測精度が向上しているのに対して，ステップワイズ重回帰分析では一部のメトリクスが追加された場合において予測精度が著しく向上し，それ以外のメトリクスが追加された場合には予測精度の変動は小さい．この結果より，類似性に基づく工数予測手法では，メトリクスを増加させても予測精度が大きく低下することはないが，ステップワイズ重回帰分析では，メトリクスの増加によって予測精度が大きく低下する可能性があることがわかる．

また，プロジェクト件数の変化と同様に，絶対誤差平均値では，他の評価基準と全く異なる傾向を示し，ステップワイズ重回帰分析の予測精度が高い結果が得

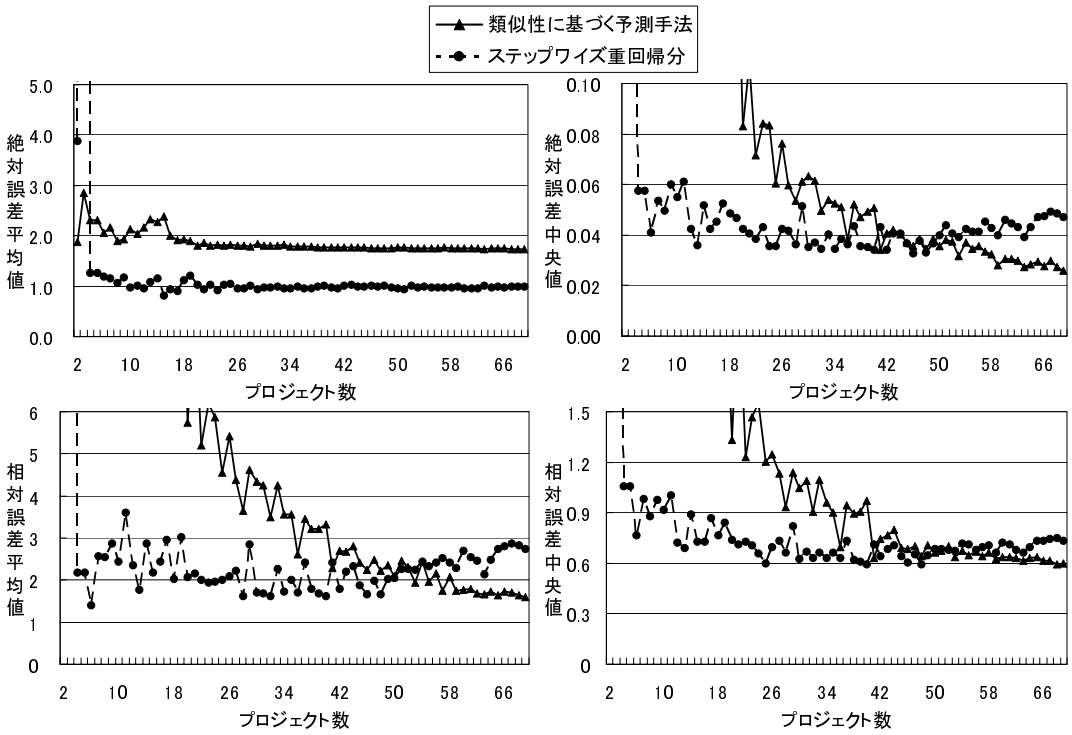

図 3.3 各プロジェクト件数における予測精度

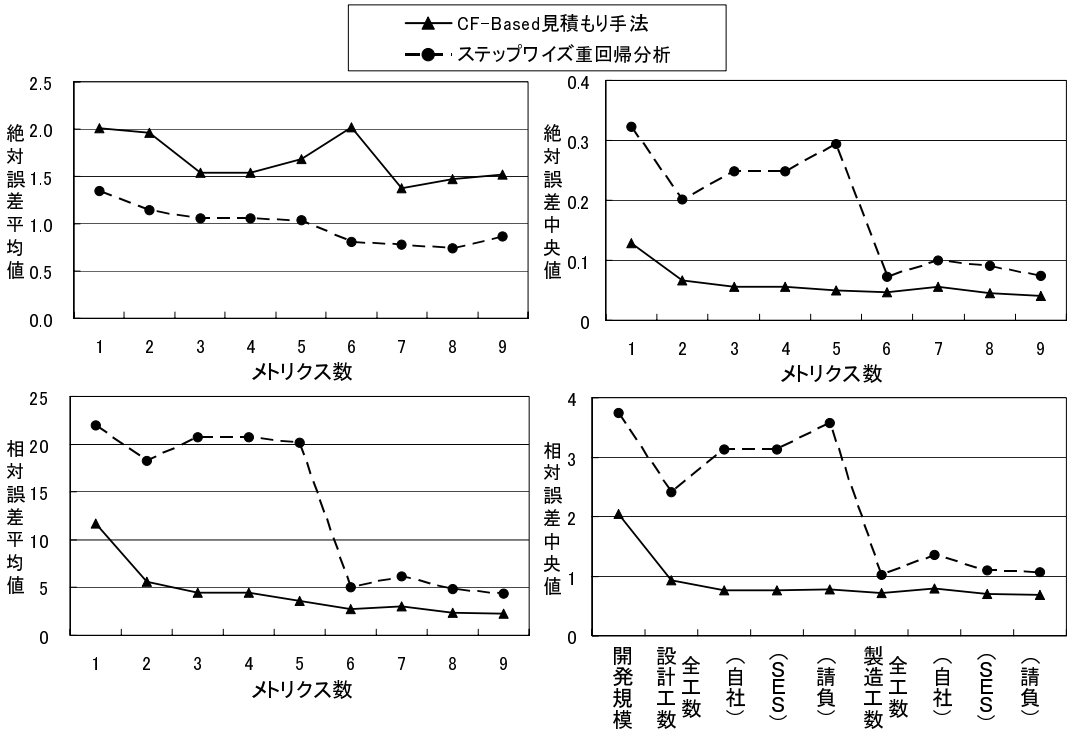

図 3.4 各メトリクス数における予測精度

られている．また，絶対誤差，相対誤差ともに，平均値よりも中央値で高い予測精度が得られている．これらの原因は，プロジェクト件数の変化と同様であると考えられる．

3.3 考察

実験の結果より，類似性に基づく工数予測手法は，プロジェクトデータの数の増加に伴なって予測精度は向上している．ただし，プロジェクト件数が少ない場合には，ステップワイズ重回帰分析の方が高い予測精度を示している．また，プロジェクト件数が著しく少ない場合には，類似性に基づく工数予測手法，ステップワイズ重回帰分析ともに高い予測精度は得られない．そこで，予測に用いるプロジェクト件数によって使用する予測手法を変えることで，より高い予測精度で予測を行うことが可能となると考えられる．そのためには，より多くのデータセットを用いた実験が必要であると考えられる．

メトリクス数の変化においては，類似性に基づく工数予測手法では，設計工数に関するメトリクスの追加時に予測精度が大きく向上している．一方，ステップワイズ重回帰分析では，設計工数に関するメトリクスの追加時には予測精度はあまり向上せず，製造工数に関するメトリクスの追加時に予測精度が大きく向上している．

設計工数に関するメトリクスを追加せず，開発規模の次に製造工数に関するメトリクスを追加した場合の結果を図 3.5 に示す．図 3.5 より，設計工数に関するメトリクスを追加しない場合でも，設計工数に関するメトリクスを追加した場合と同程度の予測精度が得られている．従って，設計工数に関するメトリクスは，ステップワイズ重回帰分析による試験工数の予測において，予測精度の向上にあまり貢献していないと言える．一方，類似性に基づく工数予測手法は，ステップワイズ重回帰分析では予測精度にあまり貢献していない設計工数を用いても高い予測精度が得られている．

実験で使用した実験データに含まれるメトリクスは，規模と工数に関するメトリクスであるため，それぞれのメトリクス間には少なからず依存関係があると考えられる．また，設計工数と製造工数では，総計とは別に自社，SES，請負の個別

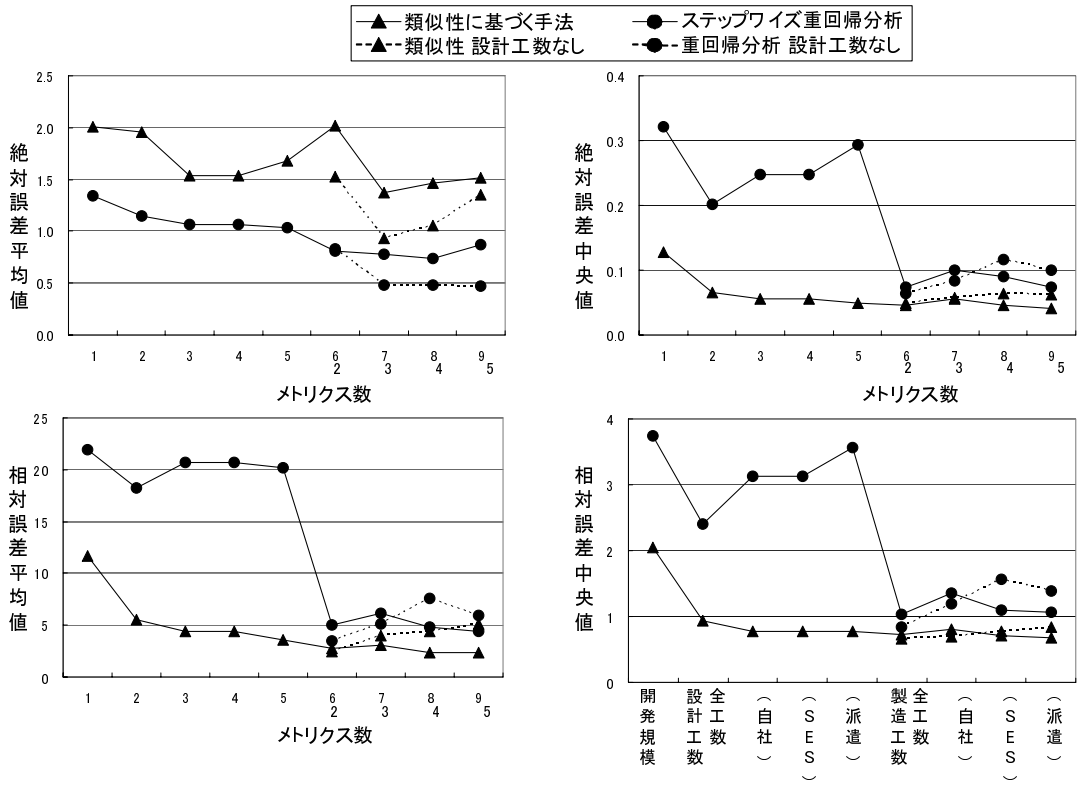

図 3.5 設計工数未追加時との比較

の工数がメトリクスとして含まれており，これらの間の依存関係は特に高いと考えられる．ステップワイズ重回帰分析では一般的に依存関係のある変数を用いて予測した場合には多重共線性を考慮していても予測精度が低下しており，それぞれ依存関係があるメトリクスが含まれている場合が多い工数予測で使用するデータでは，ステップワイズ重回帰分析よりも類似性に基づく工数予測手法が有効であると考えられる．

両方の実験において，絶対誤差平均値の結果が他の評価基準の結果と大きく異なっている．これは，実験の結果でも述べたように，ステップワイズ重回帰分析が絶対誤差平均値が最小となるようにモデルを作成するためだと考えられる．他の評価基準での結果ではグラフはほぼ同一の形を示しているため，予測精度を確かめる際に絶対誤差平均値だけを用いることは適切ではないと言える．ただし，絶対誤差平均においてもステップワイズ重回帰分析を超える予測精度が得られることが望ましい．

4. 関連研究

プロジェクト件数と予測精度の関係を調査した研究としては，天嵜 [2] の研究がある．天嵜は，複数の変数選択手法について，プロジェクト件数が 77,44,23,10 件の 4 種類のデータセットにおいて，各手法で変数選択を行い，変数選択されたデータセットにおいて，類似性に基づく工数予測手法の 1 手法である ABR(Analogy-Based Reasoning) によって工数予測を行った際の予測精度を比較している．比較の結果，プロジェクト件数が少ない（10 件）の場合を除き，ステップワイズ変数選択手法が最も高い予測精度が得られ，プロジェクト件数が少ない場合には，相関係数を用いた変数選択法が最も高い精度が得られると結論付けている．対象が工数予測の前処理である変数選択手法であり，本論文で対象とする工数予測手法とは異なるが，十分な予測精度を得るために必要なデータセットの基準を明らかにするという本論文の目的と合致している．本章の評価実験において，重回帰分析においてステップワイズ変数選択を採用しているが，変数選択手法のプロジェクト件数による精度の変化を考慮していないため，変数選択手法の影響を考慮し

た比較実験を行うことが課題として挙げられる．

メトリクス数と予測精度の関係を調査した研究としては，van Koten ら [42] の研究がある．van Koten らは，ベイジアンネットワークに基づいた工数予測手法について，メトリスク数が 2 個および 5 個のデータセットにおいて予測精度を重回帰分析と比較している．比較の結果，ベイジアンネットワークに基づいた工数予測手法と 重回帰分析がほぼ同じ精度を示すと結論付けている．ただし，メトリクス数が 2 個の場合の方が高い予測精度が得られており，変数選択によって工数予測に不必要なメトリクスを除去したに等しく，メトリクス数の変化と精度の関係を知り得るには至っていない．

5. まとめ

本章では，類似性に基づく工数予測手法のプロジェクト件数，及びメトリクス数と予測精度の関係を実験的に評価し明らかとした．実験の結果，類似性に基づく工数予測手法は，プロジェクトの数の増加に伴って予測精度が向上し，プロジェクト件数が約 50 件を超えるとステップワイズ重回帰分析よりも予測精度が高くなり，また，メトリクス数の変化では，設計工程のメトリクスの追加時に予測精度が向上し，製造工程のメトリクス追加時に予測精度が向上するステップワイズ重回帰分析よりも早期に高い予測精度が得られるという結果が得られた．従って，プロジェクト件数が 50 件以下の場合には，ステップワイズ重回帰分析を，プロジェクト件数が 50 件以上の場合や，メトリクス数が少ない場合には，類似性に基づく工数予測手法を用いるべきであると言える．また，プロジェクト件数，メトリクス数ともに，データセットに多数含まれるほど予測精度は向上するため，プロジェクト件数，メトリクス数を増やす努力を継続することが重要である．より多くのデータセットを用いて実験を行うことで，実験結果に一般性を持たせることが今後の課題である．

第4章　Fault-Prone モジュール判別におけるF1値とソフトウェア信頼性の関係

1．はじめに

ソフトウェアモジュールから計測したメトリクス値（SLOC，サイクロマティック数，継承の深さなど）に基づいて，欠陥（fault）の有無を推定するモデル（fault-prone モジュール判別モデル）が従来盛んに研究されてきた [21][30]．モデル化の方法としては，線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木などがよく用いられている [11][18][26][32]．さらに，モデルの性能を向上させるために，オーバー/アンダーサンプリング [16] や外れ値除去 [23] が併用されたり，SVM(Support Vector Machine) などの新しいモデルも提案されている [15]．

その一方で，これらの fault-prone モジュール判別モデルが，開発現場にどの程度役立つのかについての議論はほとんどなされていない．開発現場での想定される利用方法は，fault を含むと判別されたモジュールにより多くのテスト工数を割り当てる（含まないと判別されたモジュールには少しのテスト工数しか割り当てない）ことで，(1) 信頼性の確保と (2) 無駄なテスト工数の削減の両立を目指すものである．ABB Inc. では実際にこのような利用を試み [21]，効果のあることを定性的に示している．ただし，fault の有無を判別すること，テスト工数を割り当てること，及び，信頼性を確保することの定量的な関係は従来明らかにされていない．そのため，たとえ精度のよい判別モデルが得られたとしても，その有用性の程度は不明であり，判別モデルが現場で採用されにくい一因となっている．

本章では，fault-prone モジュール判別モデルの評価指標として広く採用されて

いるF1値（適合率と再現率の調和平均）[12][26]に着目し，F1値の変化がテスト効率や信頼性の向上にどのような影響を及ぼすかを明らかにすることを目的とする．そのために，fault-proneモジュールに対するテスト工数の割り当てに関する仮定を整理し，faultの有無，テスト工数，及び，fault発見率の関係をモデル化する（fault発見率モデルと呼ぶ）．また，F1値と信頼性（ソフトウェア全体でのfault発見率）の関係を明らかにするために，様々な条件の下でシミュレーションを行う．シミュレーションでは，(1)fault-proneと判別したモジュールに割り当てるテスト工数の倍率を変化させた場合，(2)F1値を固定して各種条件を変動させた場合，(3) 各種条件を固定してF1値を変化させた場合，の3つのケースを実施した．

以降，2では，構築したfault発見率モデルについて説明する．3では，構築したモデルを用いたシミュレーションとその結果について述べ，4ではF1値とテストの効率化や信頼性向上の関係について議論する．最後に6で本章のまとめと今後の課題について述べる．

2. Fault発見率モデルの構築

2.1 テスト工数の割り当てに関する仮定

Fault-proneモジュールの判別結果に対し，開発現場では，次のポリシーに基づいてテスト工数の割り当てを行うと仮定する．

- Faultを含むと判別されたモジュール（fault-proneモジュール）により多くのテスト工数を割り当てる．逆に，faultを含まないと判別されたモジュール(non-fault-proneモジュール)には少しのテスト工数しか割り当てない．
- Fault-proneモジュールには，non-fault-proneモジュールのr倍のテスト工数が割り当てられる．ここで，rは1以上の実数である．

この仮定の下では，判別精度が低いほど，実際にはfaultの無いモジュールにより多くのテスト工数を割り当てたり，faultが存在するモジュールに少ししか工数を割り当てないこととなり，信頼性の低下や工数の増大を招く．

現実には，fault を含む確率（期待値）に応じて段階的に工数を割り当てることも考えられる．しかし，実際には，fault を含む確率に応じて段階的に工数を割り当てることは開発現場ではほとんど行われていない．議論が複雑になるため，本章では，fault の有無は 2 値判別されるものとし，$1 : r$ の比でテスト工数を割り当てることとした．

2.2 1 個のモジュールに対する fault 発見率モデル

2.1 で決定されたテスト工数に基づいて，ある確率で fault が発見されることになる．一般に，モジュールに fault が含まれている場合，より多くのテスト工数を費やすほど fault の発見率は高まる．ただし，fault が含まれていない場合は，費やしたテスト工数に関わらず fault は発見されない．ここでは，モジュール中の fault の有無，テスト工数，fault 発見率の関係をモデル化する（fault 発見率モデルと呼ぶ）．

本章では，1 個のモジュールごとの fault 発見率モデルとして，比較的単純なモデルである指数関数モデルを採用する．指数関数モデルは，fault 数と発見率，時間と工数といった違いはあるものの，ソフトウェア信頼度成長モデル（SRGM: Software Reliability Growth Model）[10][34] でも採用されている．あるモジュールに含まれる fault のうち発見される fault の割合を式 (4.1) で定義する．

$$d(E) = 1 - e^{-ZE} \tag{4.1}$$

ここで，$d(E)$ はテスト工数 E を割り当てたときの fault 発見率．Z は単位工数あたりの fault 発見率である．この指数関数モデルでは，fault は費やしたコスト（テスト工数）に対して一様の確率で発見されることとなる．

この 1 個のモジュールごとの fault 発見率モデルは，プロジェクト全体で時間 t までに発見される期待 fault 数を求める SRGM と基本的な考え方は同じである．求める値は，SRGM の期待 fault 数に対して，期待 fault 発見率であるが，fault 発見率は，全 fault 数に対する発見 fault 数の割合であり，fault 発見率は fault 数とほぼ同義である．また，SRGM では，時間 t だけでなく工数 E を用いることもあ

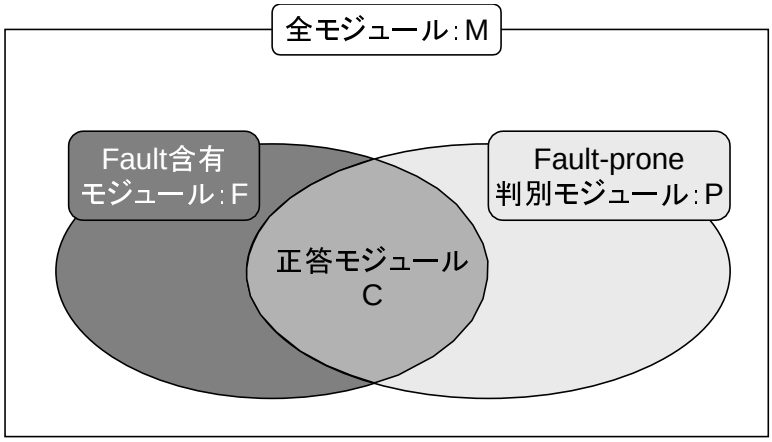

図 4.1 モジュールの関係

る．従って，この 1 個のモジュールごとの fault 発見率モデルは，SRGM と同程度の信頼性はあると考えられる．

2.3 全モジュールに対する fault 発見率モデル

式 (4.1) のモデルは，1 個のモジュールにおける fault 発見率を表したモデルである．このモデルを，全モジュールの fault 発見率のモデルに拡張する．

本章では，fault 数ではなく fault の有無を扱うため，実際に fault が含まれるモジュール（fault 含有モジュール）の数がモデルで重要な要素となる．そのため，複数モジュールにまたがる fault の場合，fault に影響する各モジュールを全て fault 含有モジュールとする．

全モジュール数を M，fault 含有モジュールの数を F，fault-prone モジュールと判別したモジュール（fault-prone モジュール）の数を P，fault-prone モジュールと判別し実際に fault を含むモジュールであったモジュール（正答モジュール）の数を C とする．それぞれを集合と考えると，それぞれの関係は図 4.1 のようになる．

Fault-prone モジュール 1 個あたりにテスト工数 E_F を費やし，non-fault-prone モジュール 1 つあたりにテスト工数 E_{NF} を費やすとすると，fault が発見される確

率は，fault 含有モジュールの内，fault-prone モジュール（図 4.1 の C）では $d(E_F)$，fault 含有モジュールの内，non-fault-prone モジュールと判別したモジュール（図 4.1 の $F-C$）では $d(E_{NF})$，実際に fault を含まないモジュール（図 4.1 の $M-F$）では 0 となる．

従って，fault 含有モジュールに含まれる全ての fault に対して，

- fault 含有モジュールの内，fault-prone モジュールと判別した割合 (Recall) のモジュールでは，$d(E_F)$ の割合で fault が発見される．
- fault 含有モジュールの内，non-fault-prone モジュールと判別した割合 (1-Recall) のモジュールでは，$d(E_{NF})$ の割合で fault が発見される．

そこで，全モジュールの fault 発見率 D は式（4.2）で表される．

$$\begin{aligned} D &= d(E_F) \times Recall + d(E_{NF}) \times (1 - Recall) \\ &= \left(1 - e^{-Z \times E_F}\right) \times Recall + \left(1 - e^{-Z \times E_{NF}}\right) \times (1 - Recall) \end{aligned} \tag{4.2}$$

式 (4.2) の E_F，E_{NF}，Recall について，詳細な式を算出する．

全テスト工数を E_{all}，fault-prone モジュールには non-fault-prone モジュールの r 倍のテスト工数が重点的に割り当てられるとすると，式 (4.3) と式 (4.4) が成り立つ．

$$E_{all} = E_F \times P + E_{NF} \times (M - P) \tag{4.3}$$

$$E_F = E_{NF} \times r \tag{4.4}$$

E_F，E_{NF} を求めるため，式 (4.3) と式 (4.4) の連立方程式を E_F と E_{NF} について解くと

$$E_F = \frac{E_{all} \times r}{P(r-1) + M} \tag{4.5}$$

$$E_{NF} = \frac{E_{all}}{P(r-1)+M} \tag{4.6}$$

が得られる．

また，Recall は式 (4.7) で表される．

$$Recall = \frac{C}{F} \tag{4.7}$$

式 (4.2) に，式 (4.5)，式 (4.6) および，式 (4.7) を代入すると，式 (4.8) が得られる．

$$D = \left(1 - e^{-\frac{E_{all} \times r}{P(r-1)+M} Z}\right) \times \frac{C}{F} + \left(1 - e^{-\frac{E_{all}}{P(r-1)+M} Z}\right) \times \frac{F-C}{F} \tag{4.8}$$

また，F1 値は図 4.1 の記号を用いると式 (4.9) で表される．

$$F1 = \frac{2C}{F+P} \tag{4.9}$$

従って，式 (4.8) を F1 値を用いて表すと式 (4.10) のようになる．

$$D = \left(1 - e^{-\frac{E_{all} \times r}{P(r-1)+M} Z}\right) \times \frac{C \times F1}{2C - F1 \times P} + \left(1 - e^{-\frac{E_{all}}{P(r-1)+M} Z}\right) \times \left(1 - \frac{C \times F1}{2C - F1 \times P}\right) \tag{4.10}$$

本章では，式 (4.8)，および，式 (4.10) の fault 発見率モデルを使用してシミュレーションを行う．

構築した fault 発見率モデルでは，全テスト工数 E_{all}，もしくは，単位工数あたりの fault を発見率 Z が増加すると fault 発見率 D は $1 - e^{-x}$（x は E_{all} または Z）で増加する．また，全モジュール数 M，もしくは，判別モジュール数 P が増加すると fault 発見率 D は $1 - e^{-1/x}$（x は M または P）で減少する．それ以外のパラメータは他のパラメータの値によって複合的に fault 発見率は変化する．

3. シミュレーション

3.1 シミュレーションの設定

本章のシミュレーションは，fault-prone モジュール判別モデルの評価指標の 1 つである F1 値の変化が，fault 発見率にどのような影響を及ぼすかを明らかにするために行う．シミュレーションは式 (4.8) のモデルに入力する値を変化させながら fault 発見率を算出することで行う．

Fault-prone モジュール判別モデルの代表的な評価指標としては，適合率（Precision），再現率（Recall），F1 値が挙げられる [12]．これらの評価指標のうち，適合率と再現率はトレードオフの関係である．そこで，本章では，それらの調和平均の値である F1 値を採用することとした．

モデルに入力する値は，現実のプロジェクトの状況を反映したものであることが望ましい．本章では，NASA/WVU で公開されているモジュールに関するデータ [28] や，情報処理推進機構 (IPA) ソフトウェア・エンジニアリング・センターで収集されているテスト工数に関するデータ [9] を参考に，次のように値を決めた．（なお，本章のシミュレーションでは，特に明記しない場合にはこれらの値を入力として用いる．）

- 全モジュール数：M=1000 (個)
- 全テスト工数：E_{all}=1500 (人時)
- 単位工数あたりの fault 発見率：Z=0.5
- Fault-prone モジュールに割り当てるテスト工数の倍率：r=3

全モジュール数 M は値を大きくするとソフトウェア全体での fault 発見率は低下し，全テスト工数 E_{all}，および，fault 発見率 Z は値を大きくすると fault 発見率は向上する．これらの値の変化は，シミュレーション結果のグラフにおいて縦軸の値が変化するだけで，グラフの形状は変化しない．従って，全モジュール数 M，全テスト工数 E_{all}，fault 発見率 Z については今回のシミュレーションでは定数とした．r については，次のシミュレーション 1 で述べる．

3.2 シミュレーション 1：fault-prone モジュールに割り当てるテスト工数の倍率 r を変化させた場合

概要

シミュレーション 1 では，fault-prone モジュールに重点的にテスト工数を割り当てたときに，ソフトウェアの信頼性，すなわちソフトウェア全体の fault 発見率が，F1 値によってどのように変化するかを明らかにする．そのために，F1 値ごとに non-fault-prone モジュールに対する fault-prone モジュールに割り当てるテスト工数の倍率 r を変化させて，fault 発見率の変化を確かめた．

シミュレーションでは，fault モジュール含有率（Fault 含有モジュール数 / 全モジュール数）を 0.2，fault-prone モジュール率（Fault-prone モジュール数 / Fault 含有モジュール数）を 1 とし，正答モジュール数は F1 値によって変化させた．

結果

シミュレーションの結果を図 4.2 に示す．シミュレーションの結果，F1 値が高いときには，fault-prone モジュールにテスト工数を多く割り当てることで，発見される fault は増加するが，F1 値が低いときには，fault-prone モジュールにテスト工数を多く割り当てると，発見される fault が減少している．また，境目となる F1 値付近（シミュレーションの結果では 0.5 付近）では，fault-prone モジュールに対して，ある程度重点的にテスト工数を割り当てることで発見される fault 数は増加するが，割り当てすぎると発見される fault 数が減少している．これは，fault-prone モジュールに対してテスト工数を割り当てすぎることで，non-fault-prone モジュールに割り当てるテスト工数が減少し，fault 含有モジュールのうち non-fault-prone と判別されたモジュールに含まれる fault の発見率の低下が，正答モジュールに含まれる fault の発見率の上昇を上回ったためである．

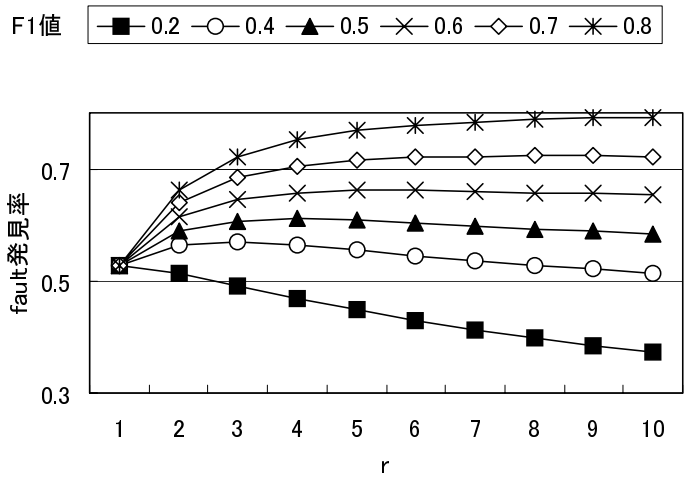

図 4.2 Fault-prone モジュールに割り当てるテスト工数の倍率の変化

3.3 シミュレーション 2：F1 値を固定して他の条件を変動させた場合

概要

シミュレーション 2 では，F1 値を固定し，fault 含有モジュール数や fault-prone モジュール数を変化させることで fault 発見率がどのように変化するかを明らかにする．そのために，F1 値を同じ値に保ったまま，fault-prone モジュール数の fault 含有モジュールに対する割合（fault-prone モジュール率），および，fault 含有モジュールの全モジュールに対する割合（fault モジュール含有率率）を変化させて，fault 発見率の変化を確かめた．

このシミュレーションは，ロジスティック回帰分析などの fault-prone モジュール判別モデルによって，各モジュールが fault を含む確率（期待値）を求め，一定の閾値で fault あり／なしを判別するケースを想定している．このケースでは，閾値を変化させることで，fault-prone と判定するモジュールの割合（fault-prone モジュール率）を変化させることができる．

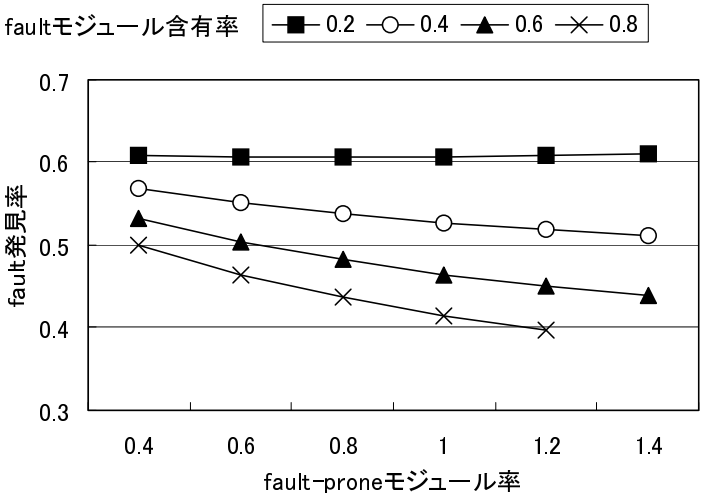

図 4.3 F1 値固定時の fault 発見率

シミュレーションでは，F1 値を 0.5 に固定した．そして，F1 値が 0.5 となるように，fault 含有モジュール数，fault-prone モジュール数，正答モジュール数を変化させた．

結果

シミュレーションの結果を図 4.3 に示す．グラフ中のプロットが存在しない箇所は，正答モジュール数が fault モジュール含有率や fault-prone モジュール数を超える場合である．シミュレーションの結果，同じ F1 値をとる場合でも，fault-prone モジュール率や，fault モジュール含有率が異なると fault 発見率は大きく異なっている．

Fault モジュール含有率が高い場合には，fault-prone モジュール率を高くするほど同じ F1 値でも発見される fault は減少している．しかし，fault モジュール含有率が低い場合には，発見される fault に大きな差は見られない．また，fault モジュール含有率が低い場合には，fault-prone モジュール率を高くすると，最初は発見される fault は減少するが，fault-prone モジュール率をより高くすることで

発見される fault が増加している．これは，fault-prone モジュール率を高くすることで，適合率は低下するが再現率が向上し，より多くの fault 含有モジュールに対して fault-prone モジュールに費やされるテスト工数が割り当てられることになるためである．

3.4 シミュレーション 3：条件を固定して F1 値を変化させた場合

概要

シミュレーション 3 では，fault 含有モジュール数や fault-prone モジュール数を固定し，F1 値を変化させることで fault 発見率がどのように変化するかを明らかにする．そのために，fault-prone モジュール率，および，fault モジュール含有率ごとに F1 値の値を変化させて fault 発見率の変化を確かめた．

シミュレーションでは，正答モジュール数を F1 値ごとに変化させた．

結果

シミュレーションの結果を図 4.4 に示す．グラフ中のプロットが存在しない箇所は，正答モジュール数が fault モジュール含有率や fault-prone モジュール数を超える場合である．シミュレーションの結果，fault-prone モジュール率を高くするほど，F1 値の変化に伴う fault 発見率の変化は大きく，fault-prone モジュール率が低いほど，F1 値の変化に伴う fault 発見率の変化は小さい．また，F1 値が小さいときには fault-prone モジュール率が低い方が fault 発見率は小さいが，F1 値が増加するほど fault-prone モジュール率による差は小さくなり，ある F1 値を境目に，fault-prone モジュール率が高い方が fault 発見率が高くなる．境目となる F1 値の値は，fault モジュール含有率の上昇に伴って大きくなる．

4. 考察

シミュレーション 1 の結果から，fault-prone モジュール判別モデルの精度（F1 値）が低いと予想される場合には，fault-prone モジュールと non-fault-prone モ

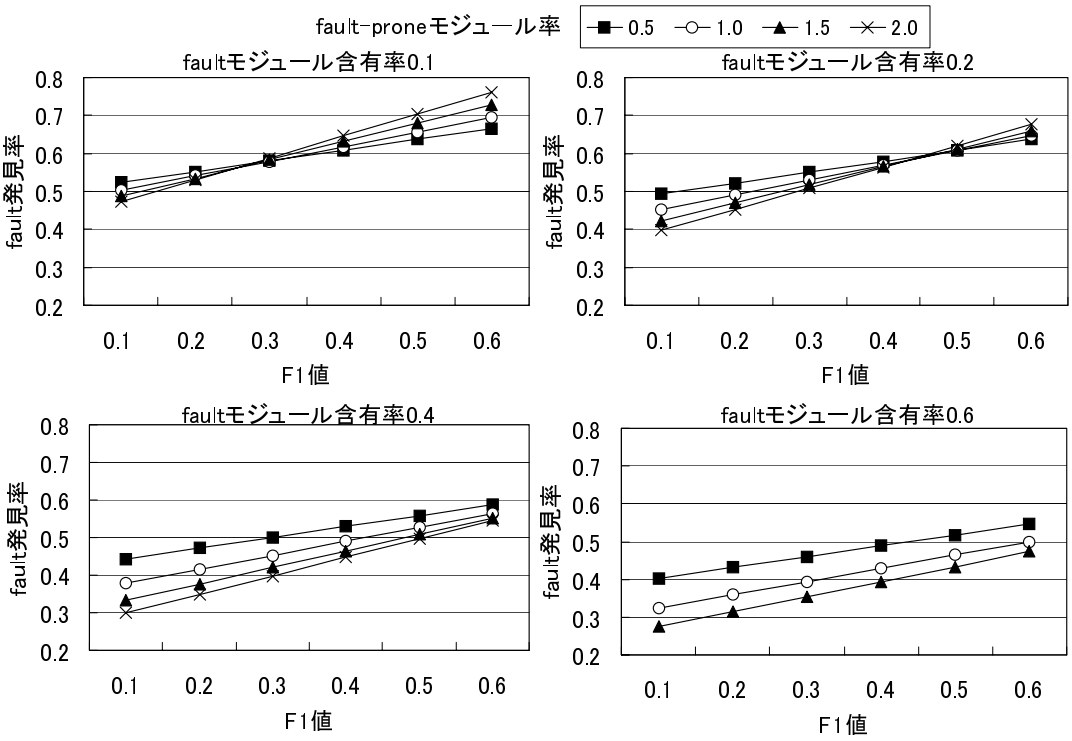

図 4.4 F1 値を変化させたときの fault-prone モジュール率ごとの fault 発見率

ジュールに均等にテスト工数を割り当てることで（ソフトウェア全体における）fault 発見率を高くでき（すなわち，fault-prone モジュール判別を行わない方が良い），予想される F1 値が高くなるにつれて，fault-prone モジュールにテスト工数を重点的に割り当てることで fault 発見率が高くできると言える．

シミュレーション 2 の結果から，fault モジュール含有率が低いと予想される場合には fault-prone モジュール率を高くする（fault-prone と判別するモジュール数を多くする）ことで fault 発見率を高くでき，予想される fault モジュール含有率が増加するにつれて fault-prone モジュール率を低くする（fault-prone と判別するモジュール数を少なくする）ことで fault 発見率を高くできると言える．

シミュレーション 3 の結果から，F1 値が低いと予想される場合には，再現率を重視して fault-prone モジュール率を低くすることで，fault 発見率を高くできると言え，予想される F1 値が増加するにつれて，適合率を重視して fault-prone モジュール率を高くすることで，fault 発見率を高くできると言える．また，fault モジュール含有率に関してはシミュレーション 2 と同様の結果が得られている．

これらのシミュレーションの結果から，プロジェクト全体の信頼性（fault 発見率）は，F1 値や，fault モジュール含有率，fault-prone モジュール率によって決定されると言える．従って，テストの効率化や信頼性の向上のためには，fault モジュール含有率，fault-prone モジュール率を考慮したうえで F1 値を上昇させる必要があると考えられる．開発現場においては，手持ちのデータセットを用いて予測を行うことでおおよその F1 値を推測可能である．手持ちのデータセットを用いて予測を行う手法として，手持ちのデータセットを n 分割し，そのうちの 1 個を見積もり対象のデータ（テストデータ），残りの $n-1$ 個を見積もりモデルを作成するデータ（フィットデータ）として予測を行う n-cross validation や，1 つのプロジェクトを抜き出し，そのプロジェクト以外のプロジェクトを用いて予測することを全てのプロジェクトで行う leave one out がある．

ただし，fault-prone モジュール判別モデルの比較などにおいては，fault-prone モジュール率を一定とすることは難しく，また，fault-prone モジュール率の変化に伴って判別精度も変化する可能性が高い．そのため，fault-prone モジュール率を固定することも，fault-prone モジュール判別モデルの性能を制限する可能性が

ある．fault モジュール含有率や fault-prone モジュール率が判明していれば，本章で用いたような何らかのモデル式等によって fault 発見率等の性能が得られる．従って，fault-prone モジュール判別モデルの評価などにおいては，F1 値だけでなく，fault モジュール含有率や fault-prone モジュール率を F1 値と併記することが望ましいと言える．

また，本章のシミュレーションで得られた結果に基づいたテストの効率化や信頼性の向上が行えると考えられる．予想される fault モジュール含有率や F1 値に基づいて，fault-prone モジュールに割り当てるテスト工数の割合を変化させたり，fault-prone モジュール判別モデルをチューニングすることで fault-prone モジュール率を変化させることで，fault 発見率の増加が見込めると考えられる．

5. 関連研究

信頼性評価を開発管理に結びつけた研究として，Kanoun ら [17]，Matsuodani ら [24] の研究がある．Kanoun ら [17] は，記述分析，傾向分析，信頼性モデルを組み合わせる事で，テストの管理，ソフトウェア信頼性の評価，保守工程の計画を行う方法を提案し，実際のプロジェクトに適用する事で有用性を示している．ただし，開発プロジェクトのテスト工程で発見された fault に基づいて記述分析，傾向分析を行い，信頼性モデルに適用するため，テスト工程開始後にしか利用することができない点が，テスト工程開始前にテスト計画の立案が可能な本章で提案した信頼性モデルとは異なる．

また，Matsuodani ら [24] は，テストを探索行為ととらえ，探索の重複の度合いから算出される探索の効率をテストの効率とみなして評価する方法を提案し，過去の論文で用いられたデータ，および，実際のテストで収集したデータに適用している．ただし，この方法は，テスト工程終了後に，発見された fault を基にテスト工程の評価を行うため，テスト工程の管理に用いることはできない．

6. まとめ

本章では，fault-prone モジュール判別において F1 値の変化がテスト効率や信頼性の向上にどのような影響を及ぼすかを明らかにするために，fault の有無，テスト工数，fault 発見率をモデル化し，F1 値と信頼性の関係をシミュレーションで明らかにした．構築した fault 発見率モデルは，ソフトウェア信頼度成長モデル（SRGM）と同程度の信頼性があると考えられる．シミュレーションの結果，プロジェクト全体の信頼性（fault 発見率）は，F1 値や，fault モジュール含有率，fault-prone モジュール率によって決定されることが判明した．

本章では，比較的簡単なモデルである指数関数モデルを基に fault 発見率モデルを構築した．指数関数以外のモデルの構築や F1 値以外の評価指標との関係についても検討を行うことが今後の課題である．

第5章　おわりに

本論文では，ソフトウェア開発管理における予測手法の開発現場での採用を促進することを目的として，予測手法の利用的側面からの評価によって，その適用の基準や得られる効果を明確にした．

まず，開発工数の予測において，予測に必要なデータセットに求められる基準を明らかにするために，データセットに含まれる欠損値，プロジェクトの件数，メトリクス数について，類似性に基づく工数予測手法とステップワイズ重回帰分析の 2 種類の工数予測手法についてコンテキストを明らかにするための評価を行った．欠損値について，欠損値が生じる 3 つのメカニズムを想定し，それぞれについて欠損率を変化させたデータセットを多数作成し，各データセットを用いて工数予測を行うことで実験的に評価した．評価実験の結果，類似性に基づく工数予測手法が，ステップワイズ重回帰分析よりもロバスト性が高い，すなわち，欠損のメカニズムに関わらず，欠損率が増大しても予測精度が大きく低下しないことが示された．また，プロジェクト件数とメトリクス数について，それぞれを変化させたデータセットを作成し，各データセットを用いて工数予測を行うことで実験的に評価した．評価実験の結果，プロジェクト件数が 50 件以上の場合，類似性に基づく工数予測手法がステップワイズ重回帰分析よりも高い精度で予測でき，メトリクス数の増加に伴って予測精度が向上することが示された．予測に用いる過去プロジェクトのデータセットのコンテキストを明らかにするための評価を行うことで，類似性に基づく工数予測手法では，メトリクス数が数件，データ欠損が 60%以下であれば，十分な予測精度が得られる，ステップワイズ重回帰分析では，プロジェクト件数が数件あれば十分な予測精度が得られる，といったことが明らかとなるため，開発現場における予測手法の採用を促進できると考えられる．

次に，信頼性予測の一手法である fault-prone モジュール判別において，各モ

ジュールの fault の有無，割り当てられるテスト工数，およびソフトウェアの信頼性（fault 発見率）の関係をモデル化することで，予測により得られる効果を明らかにした．構築した fault 発見率モデルを用いて，評価指標の 1 つである F1 値，fault モジュール含有率，fault-prone モジュール率についてシミュレーションを行った．シミュレーションの結果，プロジェクト全体の信頼性（fault 発見率）は，F1 値，fault モジュール含有率，fault-prone モジュール率によって決定されることが示された．構築した fault 発見率モデルを用いることで，fault-prone モジュール判別手法の予測後のアクションおよび評価が得られるため，開発現場における予測手法の採用を促進できると考えられる．

謝辞

本研究を進めるにあたり，研究方法などに関する多くのアドバイスと共に，研究に対する姿勢など数多くのご指導を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 松本 健一 教授に心から深く感謝します.

本研究を進めるにあたり，本研究に対し，本研究を確固たるものとする的確で有益な多くのご助言を頂きました 奈良先端科学技術大学院大学 情報科学研究科 情報基礎学講座 関 浩之 教授に心から深く感謝します.

本研究を進めるにあたり，本研究に対し，有益かつ的確な多くのご助言を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計学講座 飯田 元 教授に心から深く感謝します.

本研究を進めるにあたり，研究方法などに関する多くのアドバイスに加え，細部にわたる熱心なご指導を頂きました，奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 門田 暁人 准教授に心から深く感謝します.

本研究を進めるにあたり，多くのご助言と丁寧なご指導を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 大平 雅雄 助教に，深く感謝します.

本研究を進めるにあたり，有益なご助言と丁寧なご指導を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 森崎 修司 助教に深く感謝します.

本研究を進めるにあたり，多くの技術的なアドバイスを頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 玉田 春昭 特任助教に深く感謝します.

本研究を進めるにあたり，有益な多くのご助言と技術的なアドバイスを頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 角田 雅照 特任助教に深く感謝します.

本研究を進めるにあたり，多くのご助言を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 黒崎 章 特任助教に深く感謝します.

本研究を進めるにあたり，多くご助言を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 松村 知子 研究員に深く感謝します.

本研究を進めるにあたり，有益な多くのご助言と技術的なアドバイスを頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計講座 川口 真司 助教に感謝します.

本研究を進めるにあたり，多くのご助言を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計学講座 名倉 正剛 特任助教に深く感謝します.

本研究を進めるにあたり，有益な多くのご助言と丁寧なご指導を頂きました 神戸大学 大学院工学研究科 情報知能学専攻 中村 匡秀 准教授に深く感謝します.

本研究を進めるにあたり，有益なご助言と技術的なアドバイスを頂きました 阪南大学 経営情報学部 花川 典子 准教授に深く感謝します.

本研究を進めるにあたり，有益な多くのご助言と技術的なアドバイスを頂きました 株式会社 NTT データ 大杉 直樹 氏に深く感謝します.

本研究を進めるにあたり，奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 山内 寛己氏，栗山 進氏には多くのご助言やご協力を頂きました．ここに記して謝意を表します.

本研究を進めるにあたり，様々なご助力を頂きました 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 秘書 乾 純子氏に深く感謝します.

本研究を進めるにあたり，技術的なご助言やご協力を頂きました　奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座 定量的ソフトウェア管理グループの皆様に深く感謝します.

本研究を進めるにあたり，発表資料準備などにおいて，ご助言やご協力を頂いた 奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座，ソフトウェア設計学講座の皆様に感謝します.

最後に，日頃より私を励まし，支えてくれた家族や友人らに心より感謝します.

参考文献

[1] A. Albrecht, and J. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction,” IEEE Transactions on Software Engineering, vol.9, no.6, pp.83-92, 1979.

[2] 天嵜 聡介, “工数見積もり手法における変数選択手法の比較,” 情報処理学会研究報告, 2006-SE-154, vol.2006, no.125, pp.17-24, 2006.

[3] B.W. Boehm, Software Engineering Economics, Prentice Hall, New Jersey, 1981.

[4] L. Briand, V. Basili, and C. Hetmanski, “Developing Interpretable Models with Optimized Set Reduction for Identifying High-risk Software Components,” IEEE Transactions on Software Engineering, vol.19, no.11, pp.1028-1024, 1993.

[5] L. Briand, T. Langley, and I. Wieczorrek, “A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques,” In Proceedings of 22nd International Conference on Software Engineering, Limerick, Ireland, pp.377-386, 2000.

[6] S. Chulani, B. Boehm, and B. Steece, “Bayesian Analysis of Empirical Software Engineering Cost Models,” IEEE Transactions on Software Engineering, vol.25, no.4, pp.573-583, 1999.

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

[8] 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, ソフトウェア開発見積りガイドブック〜IT ユーザとベンダにおける定量的見積りの実現〜, 株式会社オーム社, 東京, 2006.

[9] 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, ソフトウェア開発データ白書２００６ ～IT 企業 1400 プロジェクトの定量データで示す開発の実態～, 日経 BP 社, 東京, 2006.

[10] A.L. Goel, and K. Okumoto, “Time-dependent Error detection Rate Model for Software Reliability and Other Performance Measures,” IEEE Transactions on Reliability, vol.28, no.3, pp.206-211, 1979.

[11] A.R. Gray, and S.G. MacDonell, “Software Metrics Data Analysis – Exploring the Relative Performance of Some Commonly Used Modeling Techniques,” Empirical Software Engineering, vol.4, no.4, pp.297-316, 1999.

[12] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, “Evaluating Collaborative Filtering Recommender Systems,” ACM Transactions on Information Systems, vol.22, no.1, pp.5-53, 2004.

[13] “ISBSG Estimating, Benchmarking, and Research Suite Release 9,” International Software Benchmarking Standards Group, 2004, http://www.isbsg.org/

[14] 柿元健, 角田雅照, 大杉直樹, 門田暁人, 松本健一, “協調フィルタリングに基づく工数見積もりのロバスト性評価,” ソフトウェア工学の基礎 XI, 日本ソフトウェア科学会 FOSE2004, pp.73-84, 2004.

[15] Y. Kamei, A. Monden, and K. Matsumoto, “Empirical Evaluation of Svm-based Software Reliability Model,” In Proceedings of the 5th International Symposium on Empirical Software Engineering, vol.2, pp.39-41, Rio de Janeiro, Brasil, 2006.

[16] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K. Matsumoto, “The Effects of Over and Under Sampling on Fault-Prone Module Detection,” In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, pp.196-204, Madrid, Spain, 2007.

[17] K. Kanoun, M. Kaaniche, and J.C. Laprie, "Qualitative and Quantitative Reliability Assessment," IEEE Software, vol.14, no.2, pp.77-86, 1997.

[18] T.M. Khoshgoftaar, and E.B Allen, "Modeling Software Quality with Classification Trees," Recent Advances in Reliability and Quality Engineering, World Scientific, pp.247-270, Singapore, 1999.

[19] J. Kromrey, and C. Hines, "Nonrandomly Missing Data in Multiple Regression: An Empirical Comparison of Common Missing-data Treatments," Educational and Psychological Measurement, vol.54, no.3, pp.573-593, 1994.

[20] J. Li, A. Al-Emran, and G. Ruhe, "Impact Analysis of Missing Values on the Prediction Accuracy of Analogy-based Software Effort Estimation Method AQUA," In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, pp.126-135, Madrid, Spain, 2007.

[21] P.L. Li, J. Herbsleb, M. Shaw, and B. Robinson, "Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc," In Proceedings of 28th International Conference on Software Engineering, pp.413-422, Shanghai, China, 2006.

[22] R.J.A Little, and D.B. Rubin, Statistical Analysis with Missing Data, 2nd edition, John Wiley and Sons, New York, 2002.

[23] S. Matsumoto, Y. Kamei, A. Monden, and K. Matsumoto, "Comparison of Outlier Detection Methods on Fault-proneness Models," In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, pp.461-463, Madrid, Spain, 2007.

[24] Tohru Matsuodani, and Kazuhiko Tsuda, "Evaluation of Debug-testing Efficiency by Duplication of the Detected Fault and Delay Time of Repair," Information Sciences, vol.166, no.1-4, pp.83-103, 2004.

[25] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A Comparative Study of Cost Estimation Models for Web Hypermedia Applications," Empirical Software Engineering, vol.8, no.2, pp.163-196, 2003.

[26] J.C. Munson, and T.M. Khoshgoftaar, "The Detection of Fault-prone Programs," IEEE Transactions on Software Engineering, vol.18, no.5, pp.423-433, 1992.

[27] I. Myrtveit, E. Stensrud, and U.H. Olsson, "Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-based Methods," IEEE Transactions on Software Engineering, vol.27, no.11, pp.999-1013, 2001.

[28] NASA IV&V Facility, Metrics Data Program, http://mdp.ivv.nasa.gov/

[29] "Naist Collaborative Filtering Engines," http://sourceforge.jp/projects/ncfe/

[30] N. Ohlsson, and H. Alberg, "Predicting Fault-prone Software Modules in Telephone Switches," IEEE Transactions on Software Engineering, vol.22, no.12, pp.886-894, 1996.

[31] N. Ohsugi, M. Tsunoda, A. Monden, and K. Matsumoto, "Applying Collaborative Filtering for Effort Estimation with Process Metrics," In Proceedings of 5th International Conference on Product Focused Software Process Improvement, Lecture Notes in Computer Science, vol.3009, pp.274-286, Kyoto, Japan, 2004.

[32] M. Pighin, and R. Zamolo, "A Predictive Metric Based on Discriminant Statistical Analysis," In Proceedings of 19th International Conference on Software Engineering, pp.262-270, Boston, 1997.

[33] L. H. Putnam, "A General Empirical Solution to the Macro Sizing and Estimating Problem," IEEE Transactions on Software Engineering, vol.4, no.4,

pp.345-361, 1971.

[34] C.V. Ramamoorthy, and F.B. Bastani, “Software Reliability-status and Perspective,” IEEE Transactions on Software Engineering, vol.8, no.4, pp.354-371, 1982.

[35] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” In Proceedings of the Conference on Computer Supported Cooperative Work, pp.175-186, Chapel Hill, North Carolina, 1994.

[36] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl, “Item-based Collaborative Filtering Recommendation Algorithms,” In Proceedings of 10th International World Wide Web Conference, pp.285-295, Hong Kong, 2001.

[37] M. Shepperd, and C. Schofield, “Estimating Software Project Effort Using Analogies,” IEEE Transactions on Software Engineering, vol.23, no.12, pp.736-743, 1997.

[38] K. Srinivasan, and D. Fisher, “Machine Learning Approaches to Estimating Software Development Effort,” IEEE Transactions on Software Engineering, vol.21, no.2, pp.126-137, 1995.

[39] K. Strike, K. El Emam, and N. Madhavji, “Software Cost Estimation with Incomplete Data,” IEEE Transactions on Software Engineering, vol.27, no.10, pp.890-908, 2001.

[40] 田中豊，垂水共之 (編), “Windows 版 統計解析ハンドブック 多変量解析,” p.240，共立出版，東京，1995.

[41] 角田雅照, 大杉直樹, 門田暁人, 松本健一, 佐藤慎一, “協調フィルタリングを用いたソフトウェア開発工数予測方法,” 情報処理学会論文誌, vol.46, no.5, pp.1156-1164, 2005.

[42] C. van Koten, and A.R. Gray, "Bayesian Statistical Effort Prediction Models for Data-centred 4gl Software Development," Information and Software Technology, vol.48, no.11, pp.1056-1067, 2006.

[43] C. Walston, and C. Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, vol.1, pp.54-73, 1977.