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

信学技報
IEICE Technical Report
SS2005-47 (2005-10)

類似プロジェクトを用いた工数予測方法の性能比較

角田　雅照† 　柿元　健† 　大杉　直樹† 　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 けいはんな学研都市

E-mail: †{masate-t, takesi-k, naoki-o, akito-m, matumoto}@is.aist-nara.ac.jp

あらまし 近年，ソフトウェア開発組織や開発プロセスは多様化しており，COCOMO や重回帰分析等の単一のモデル式を用いた工数予測手法では十分な予測精度が得られないことがある．そこで，プロジェクトの個別性をより強く反映できる，プロジェクト間の類似度に基づく予測方法(事例ベース推論(CBR)，協調フィルタリング(CF))が注目されている．ただし，CBR や CF は適用実績が少なく，その優劣も明らかにされていない．そこで本論文では，実プロジェクトのデータを用いて CBR と CF の予測性能を比較する実証実験を行った．その結果，ＣＦのほうがＣＢＲよりも予測性能が高いことが確認できた．

キーワード 欠損値，協調フィルタリング，事例ベース推論，コスト見積もり，プロジェクト管理

A Comparison of Effort Prediction Methods Using Similar Projects

Masateru TSUNODA† 　Takeshi KAKIMOTO† 　Naoki OHSUGI† 　Akito MONDEN†

and 　Ken'ichi MATSUMOTO†

†Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630-0192 Japan

E-mail: †{masate-t, takesi-k, naoki-o, akito-m, matumoto}@is.aist-nara.ac.jp

Abstract Due to heterogeneity of recent software development organizations and software processes, a single effort prediction model, such as COCOMO and regression models, often does not perform well to predict an individual project. On the other hand, effort prediction methods based on project similarity (Case-Based Reasoning (CBR) and Collaborative Filtering (CF)) have become an attractive alternatives in recent years because they consider individuality of each project in prediction. However, there are no empirical evaluation conducted to compare CBR and CF. Therefore, we focus on these methods and compared their prediction performance using real project data. The result showed that CF showed better prediction performance than CBR.

Keyword missing value, collaborative filtering, case-based reasoning, cost estimation, project management

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，工数(コスト)予測は欠かせない要素であり，人員などの資源配置やスケジュール管理の基礎となる[14]．ただし，工数予測の手段は複数存在するため，状況に応じて適切な方法を採用する必要がある．

工数を予測する１つの手段は，COCOMO[2]や SLIM[15]などの汎用的な（予め定義された）工数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）の見積もり値から開発工数，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，見積もり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特性値を与えることで，より正確な工数予測が行える．しかし，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，これらの予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[3]．

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[19]などを用い，自社の実績データセットに特化した工数予測モデルを構築することである．実績データセットとは，完了したプロジェクトごとに工数や開発規模，検出バグ数などの特性値を記録，蓄積したものである．実績データセットにより多くのプロジェクトを含むほど，また，より多種類の特性値を含むほど，より高い予測精度が得られることが期待される[12]．しかし，これらの予測モデルでは，多様化するプロジェクトの特性値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 などの予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)[18]，および，協調フィルタリング(CF)[22]が提案されている．これらの手法は，重回帰分析と同様，過去の実績データセットを必要とするが，データセット全体に対してただ１つの予測式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式を構築するため，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，経験に頼った見積もり方法として，過去に手がけた類

	特性値 1	特性値 2	…	特性値 j	…	特性値 l
p_1	m_{11}	m_{12}	…	m_{1j}	…	m_{1l}
p_2	m_{21}	m_{22}	…	m_{2j}	…	m_{2l}
…	…	…		…		…
p_i	m_{i1}	m_{i2}	…	m_{ij}	…	m_{il}
…	…	…		…		…
p_k	m_{k1}	m_{k2}	…	m_{kj}	…	m_{kl}

表 1 予測に用いるデータセット

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見積もる場合がある（「類推見積もり」などとも呼ばれている）が，CBR や CF は，このような経験に頼った見積もりを，系統的に行う方法である．

近年，CBR については，適用事例が徐々に増えており，重回帰分析に対する優位性がいくつかの論文で確認されている0[23]．ただし，未計測の値（欠損値）が予測性能に与える影響は明らかにされていない．一方，CF は欠損値が予測性能に与える影響が小さいことが報告されているが[8]，適用事例そのものが少なく，評価は必ずしも定まっていない．また，CBR と CF の予測性能を比較した事例も従来報告されていない．

そこで，本論文では，近年有望視されている CBR と CF について，より現実的なデータセットを用いて予測性能を比較する実証実験を行うことを目的とする．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実績データセットには欠損値が含まれるため，本論文では，データの欠損率を様々に変えたデータセットを用意し，それぞれのデータセットについて CBR と CF を用いた工数予測を行い，その精度を比較する．本論文で用いたデータセットは，Desharnais[5]によって収集されたカナダのソフトウェア開発企業におけるものであり，77 件のプロジェクトのデータを含む[10]．

以降，2 章では CF について，3 章では CBR について説明する．4 章では実験方法および結果を述べたあと，結果に対する考察を行う．最後に 5 章でまとめを述べる．

2. 協調フィルタリング

協調フィルタリング(CF)は，多量に存在するアイテム(記事，ウェブページ，書籍，楽曲，映画作品など)の中からユーザの好みに合うと予測されるアイテム情報を選出して推薦するシステムの基盤技術として用いられている．CF に基づく推薦システムでは，「あるアイテムに対して同様の評価をするユーザ同士は，他のアイテムに関しても同様の評価をするであろう」と考え，自分と類似した評価をしているユーザが高い評価をしたアイテムの推薦を行う．

我々の研究グループでは，CF に基づく推薦システムの考え方をソフトウェアプロジェクトの工数予測に適用し，「類似するプロジェクト同士は，工数も類似しているであろう」と考え，プロジェクト間の類似度を計算し工数予測を行う方法を提案した[22]．

CF では，欠損値を非常に多く含むデータを用いて，ユーザの嗜好を予測することが前提となっている．Amazon.com などの一般的な推薦システムでは，存在するアイテム（書籍）の量が莫大であるため，ほぼ全てのユーザは全アイテムの 1%未満しか評価していないためである．我々の提案方法でも，欠損値を非常に多く含むデータセットを用いた予測が可能となっている．

CF による工数予測では，表 1に示す $k \times l$ 行列で表されるデータセットを入力として用いる．図中，p_i は i 番目のプロジェクトを表し，m_{ij} はプロジェクト p_i の j 番目の特性値を表す．すなわち，行がプロジェクト，列が特性値を表している．ここで p_a を予測対象のプロジェクト，$\hat{m}_{ab}$ を m_{ab} の予測値とする．CF による工数予測のアルゴリズムはいくつか考えられるが，以降では，実験の結果高い予測性能を示したもののみを述べる．CF による工数予測は，以下の手順に従って行われる

1. 互いに異なる各特性値の値域を統一する(特性値の正規化)．本論文では，特性値 m_{ij} の正規化された値 m'_{ij} を計算する際，以下の 2 つのアルゴリズムを用いる．

   Normalize: 以下の式を用いて，特性値の値域を[0, 1]に揃える．

   $$m'_{ij} = \frac{m_{ij} - min(m_j)}{max(m_j) - min(m_j)} \quad (1)$$

   ここで，$max(m_j)$ と $min(m_j)$はそれぞれ j 番目の特性値の最大値，最小値を表す．この計算方法は，値域を変換する際によく用いられるものの 1 つである[20]．

   Order: 特性値の順位付けを行い，その順位を特性値の代わりとして用いて正規化を行う．例えば，ある特性値が{1, 2, 2, 10}であるとき，順位付けを行うと{1, 2.5, 2.5, 4}となる．同順位の場合は平均順位を用いる．特性値 m_{ij} の順位を $odr(m_{ij})$とすると，次式によって $odr(m_{ij})$の値域を[0,1]に揃える．

   $$m'_{ij} = \frac{ord(m_{ij}) - min(ord(m_j))}{max(ord(m_j)) - min(ord(m_j))} \quad (2)$$

2. 予測対象プロジェクト p_a と他のプロジェクト p_i との類似度 $sim(p_a, p_i)$を求める．p_a と p_i が持つ特性値を要素とする 2 つのベクトルを作成し，ベクトルのなす角のコサインを用いて類似度を計算する．予測対象プロジェクト p_a と他のプロジェクト p_i との類似度 $sim(p_a, p_i)$を次式によって計算する．

   $$sim(p_a, p_i) = \frac{\sum_{j \in M_a \cap M_i} (m'_{aj} - avg(m'_j))(m'_{ij} - avg(m'_j))}{\sqrt{\sum_{j \in M_a \cap M_i} (m'_{aj} - avg(m'_j))} \sqrt{\sum_{j \in M_a \cap M_i} (m'_{ij} - avg(m'_j))}} \quad (3)$$

   ここで M_a と M_i はそれぞれプロジェクト p_a と p_i で計測された（未欠損の）特性値の集合を表し，$avg(m'_j)$は j 番目の特性値の順位の平均値を表す．$sim(p_a, p_i)$の値域は[-1, 1]である．

   従来の CF による工数予測[8][22]では，正規化した特性値をそのまま用いているが，本論文では

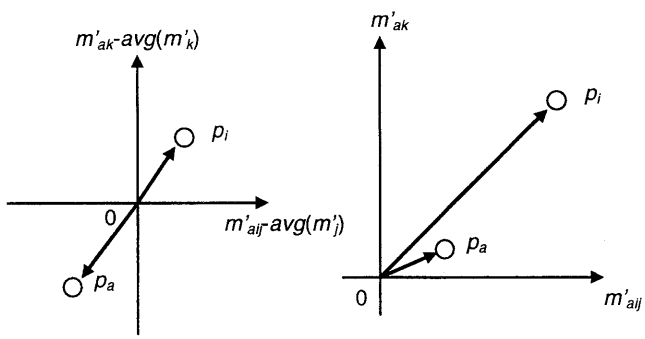

図 1　CF による類似度計算例

m'_{ij}から $avg(m'_j)$を減算し，m'_{ij}の重心(平均値)を 0 に移した点が異なる．これにより，平均順位よりも大きい特性値は正の値をとり，小さい特性値は負の値をとるため，

3. 図 1の例で示すように，大きく特性値が離れたプロジェクトは類似度が低くなる．同様の計算方法が[17]で提案されている．
類似度計算では，未欠損の値のみを用いるため，欠損値を補完する必要がない．なお，類似度計算は，他のアルゴリズム（相関係数を用いるものなど[16]）にも交換可能である．
4. 前の手順で求めた類似度 $sim(p_a, p_i)$を用いて，プロジェクト p_a の特性値の予測値 $\hat{m}_{ab}$を計算する．本論文では以下の 2 つのアルゴリズムを用いる．
Weighted Sum: 類似度 $sim(p_a, p_i)$を重みとして，プロジェクト p_a と類似したプロジェクトの特性値 m_{ib} の加重平均を行う．これは，従来の協調フィルタリングのアルゴリズム[17]と同様のものである．予測値 $\hat{m}_{ab}$を次式により計算する．

$$\hat{m}_{ab} = \frac{\sum_{i \in k-nearestProjects} (m_{ib} \times sim(p_a, p_i))}{\sum_{i \in k-nearestProjects} sim(p_a, p_i)} \quad (4)$$

ここで *k-nearestProjects* は，プロジェクト p_aとの類似度が高い k 個のプロジェクトの集合を表す(以下類似プロジェクトと呼ぶ)．類似プロジェクトは，特性値 m_{ib} が測定されているもののみとする．一般に，この類似プロジェクト数は予測精度に影響を与える．
Amplified Weighted Sum: 予測値計算時に，プロジェクトの規模を補正する $amp(p_a, p_i)$を乗じた値で加重平均を行う[22]．ただし，本論文では $amp(p_a, p_i)$の計算方法を変更している．

$$\hat{m}_{ab} = \frac{\sum_{i \in k-nearestProjects} (m_{ib} \times amp(p_a, p_i) \times sim(p_a, p_i))}{\sum_{i \in k-nearestProjects} sim(p_a, p_i)} \quad (5)$$

$$amp(p_a, p_i) = \frac{\sum_{j \in M_a \cap M_i} \left(\frac{m_{aj}}{m_{ij}} \times cor(m_b, m_j) \right)}{\sum_{j \in M_a \cap M_i} cor(m_b, m_j)} \quad (6)$$

ここで，$cor(m_b,m_j)$は予測対象である m_b とそれ以外の m_j との相関係数を表す．ただし，$cor(m_b,m_j)$が負の値になる場合は $cor(m_b,m_j)$=0.0 として計算した．

3. 事例ベース推論

事例ベース推論(CBR)は，人工知能の分野で研究されてきた問題解決のための方法である．CBR では，蓄積された過去の事例の中から，問題を解決したい現在の事例と類似したもの抽出し，その解決方法を適用する．CBR の基礎となる考え方は「類似した問題は類似した解決方法を採っている」というものである．

CBR の特長は 2 つある[6]．1 つは COCOMO のように予め定義されたモデル式を用いないため，環境の変化が激しく，作成したモデル式が陳腐化しやすい状況においても，最新のデータの特徴を反映した予測を行うことができることである．もう 1 つの特長は，重回帰分析のように 1 つのモデル式ですべての事例に適合させようとしないため，個別の事例に対しても比較的適合しやすいことである．なお，これらの特長は CF にも当てはまることである．

Shepperd ら[18]は CBR をソフトウェアプロジェクトの工数予測に適用することを提案した．CBR による工数予測においても，表 1に示す $k \times l$ 行列で表されるデータセットを入力として用いる．CBR による工数予測は，以下の手順に従って行われる0.

1. データセット中のプロジェクトにおける，各特性値の範囲を [0,1]に正規化し，特性値ごとの単位の違いが予測に影響を与えないようにする．これには次式を用いる．

$$std(m_{ij}) = \frac{m_{ij} - min(m_j)}{max(m_j) - min(m_j)} \quad (7)$$

2. 予測対象のプロジェクトと類似した，工数が既知の完了プロジェクトを見つけるため，2 つのプロジェクト間の「距離」を尺度として用いる．いくつかのアルゴリズムが提案されているが，プロジェクトを l 個の特性値で表現し，l 次元空間におけるユークリッド距離を求めることが一般的である[11]．この距離が短いほど 2 つのプロジェクトは類似しているとする．この距離は次式により求める．

$$dist(p_a, p_i) = \sqrt{\sum_j (std(m_{aj}) - std(m_{ij}))^2} \quad (8)$$

3. 類似プロジェクトの工数から，予測対象のプロジェクトの工数を予測する．データセットの大きさにもよるが，2 または 3 個の類似プロジェクトを用いて予測することが多い[11]．予測値計算には主に以下の 3 つのアルゴリズムのいずれかを用いる．
単純平均: 類似プロジェクトの工数を単純平均する．CBR で最も一般的に用いられる方法である[11]．k を類似プロジェクト数を表すとするとき，次式により計算する．

$$\hat{m}_{ab} = \frac{\sum_{i \in k-nearestProjects} m_{ib}}{k} \quad (9)$$

距離の逆数を用いた加重平均: 類似プロジェクト

特性値	平均値	中央値
開発期間	11.3	10
開発チームの経験年数	2.3	2
プロジェクトマネージャーの経験年数	2.6	3
トランザクション数	177.5	134
未調整ファンクションポイント	298.0	258
調整要素	27.5	28
調整済みファンクションポイント	282.4	247
開発環境(カテゴリー変数)	-	-
開発完了年(西暦)	85.8	86
エンティティ数	120.5	96
開発総工数	4833.9	3542

表 2 実験に用いたデータセット

間の距離の逆数を重みとして加重平均を行う[7]. 次式により予測値を計算する.

$$\hat{m}_{ab} = \frac{\sum_{i \in k-nearestProjects} \left(m_{ib} \times \frac{1}{dist(p_a, p_i)} \right)}{\sum_{i \in k-nearestProjects} \frac{1}{dist(p_a, p_i)}} \quad (10)$$

順位の逆数を用いた加重平均: p_a と距離が近い順に，類似プロジェクトを 1, 2, 3...と順位付けし，順位の逆数を重みとして加重平均を行う[7]. $rank(dist(p_a,p_i))$を距離が近い順に付けられた順位とするとき，次式により予測値を計算する.

$$\hat{m}_{ab} = \frac{\sum_{i \in k-nearestProjects} \left(m_{ib} \times \frac{1}{rank(dist(p_a, p_i))} \right)}{\sum_{i \in k-nearestProjects} \frac{1}{rank(dist(p_a, p_i))}} \quad (11)$$

CBR と CF の最も大きな違いは，プロジェクト間の類似度（距離）の計算方法にある．CF では 2 つのベクトルのなす角を用いるのに対し，CBR ではベクトル間のユークリッド距離を用いる点が異なる．この差は，特にデータセットに欠損値が存在する場合に類似度（距離）の計算に大きな影響を与える．

CBR では，欠損値を多く含むほど，ベクトルの次元が下がり，ベクトル間のユークリッド距離は小さくなる傾向にある．そのため，事前に欠損値処理を行う必要がある．欠損値処理とは，データセットに予測方法を適用する前に，あらかじめ欠損値の存在をなくす処理である．欠損値処理法としては以下の 2 つが広く用いられている[9].

リストワイズ除去法 ：欠損値を一つでも含んでいるプロジェクトをデータセットから除去し，欠損値を全く含まないプロジェクトのみでモデル作成を行う．

平均値挿入法 ：特性値から欠損値を除いて平均値を計算し，計算した平均値を欠損値部分に挿入する．これによりデータセットに欠損値が含まれていない状態になる．欠損値部分に値を挿入する方法は複数提案されているが，これはその中で最も単純かつ一般的な方法である[13].

4. 実験

4.1. 実験用データ

実験に用いたデータセットは，Desharnais[5]によって収集されたカナダのソフトウェア開発企業における 80 年代のデータであり，77 件のプロジェクトが含まれている[10]. データセットには特性値が 10 種類記録されており，欠損値は含まれていない．データの詳細を表 2に示す．表中の開発期間からエンティティまでの特性値を説明変数とし，開発総工数を目的変数として予測を行った.

予測を行う工程として，開発対象ソフトウェアのファンクションポイントが計測が終了した時点，すなわち，概要設計もしくは基本設計の完了時を想定している．また，開発期間（納期）は，開発初期に予め決定されていることを想定している.

このデータセットを用いて，以下の手順により，一定割合で欠損値を含むデータセットを複数作成した.

1. データセットを無作為に半数ずつに分け，一方をフィットデータ，もう一方をテストデータとし，これらのペアを 10 組作成した．テストデータのプロジェクトは工数が未知と仮定され，工数予測の対象となる．フィットデータのプロジェクトは，テストデータのプロジェクトの工数を予測するために用いられる.
2. フィットデータの特性値を一定の割合でランダムに欠損させた．10 個のフィットデータそれぞれについて，欠損率を 10%～90%まで 10%刻みで変化させ，フィットデータを欠損率ごとに 10 個ずつ新たに作成した.

4.2. 評価尺度

予測性能の比較に用いた評価尺度を説明する．これらは，予測方法の評価尺度として広く用いられているものである[11].

***mean*(|*r*|)** : 絶対誤差(|*r*|)を，実績工数を z，予測工数を y として以下のように定義したときの|*r*|の平均値．*mean*(|*r*|)が小さいほど予測性能が高いことを示す.

$$|r| = |z - y| \quad (12)$$

MMRE : 相対誤差(MRE) [4]を，実績工数を z，予測工数を y として以下のように定義したときの，MRE の平均値．MMRE が小さいほど予測性能が高いことを示す.

$$\mathrm{MRE} = 100 \times \left| \frac{z - y}{z} \right| \quad (13)$$

MdMRE : MRE の中央値．MdMRE が小さいほど予測性能が高いことを示す.

PRED(25) : 予測したプロジェクト件数を n とし，そのうち MRE が 25%以下となった件数を m とすると，PRED(25)は以下のように定義される[4]. PRED(25)が大きいほど予測性能が高いことを示す.

$$\mathrm{PRED}(25) = 100 \times \frac{m}{n} \quad (14)$$

4.3. 実験結果

まず，CBR における最適な欠損値処理法と予測値計算アルゴリズムを調べるために，欠損値を含むデータセットに，リストワイズ除去法，平均値挿入法をそれぞれ適用後，それらのデータセットに対し CBR の 3 種類の予測値計算アルゴリズムをそれぞれ適用した．また，(予測値計算に用いる)類似プロジェクトの個数

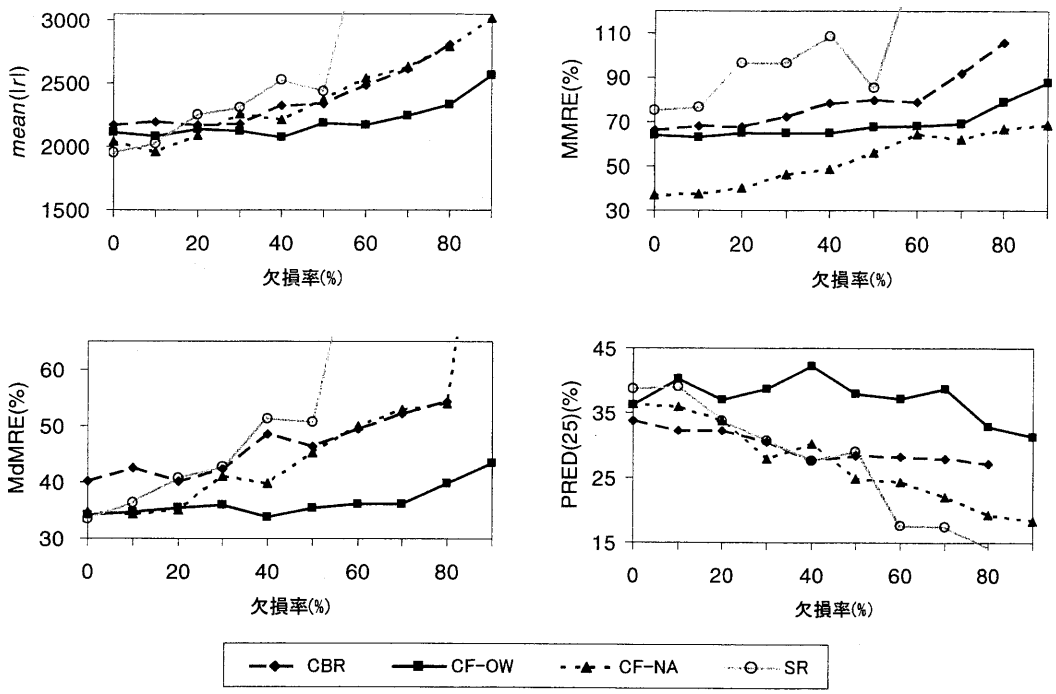

図 2　CF と CBR の予測性能比較

を決定するために，欠損値が含まれていないデータセットを用いて予測を行い，MdMRE が最小となった類似プロジェクト数を採用した．類似プロジェクト数はそれぞれ単純平均アルゴリズムの場合は 3，距離の逆数を用いた加重平均アルゴリズム，および，順位の逆数を用いた加重平均アルゴリズムの場合は 4 とした．

CBR の予測実験を行った結果，リストワイズ除去を行った場合，欠損率が 30%以上のフィットデータでは，すべてのプロジェクトが除去されてしまったデータセットが存在したため，欠損率 20%までの予測しか行っていない．また，平均値挿入を行った場合，欠損率が 90%のフィットデータでは，特性値がすべて欠損値のため平均値が計算できないデータセットが存在しており，欠損率 80%までの予測しか行っていない．

次に，CF における性能の高いアルゴリズムの組み合わせを決定した，予備実験によって，Order と Weighted Sum の組み合わせ(以下 CF-OW)，および，Normalize と Amplified Weighted Sum の組み合わせ(以下 CF-NA)の 2 つを採用した．類似プロジェクトの個数は CBR と同様の方法により，CF-OW の場合は 13，CF-NA の場合は 3 と決定した．予測結果は，それぞれの欠損率ごとに最も高い性能を示したアルゴリズムのものを採用した．

また，実験に用いたデータセットにおける，一般的な工数予測方法による精度を確認するために，ステップワイズ重回帰分析[21]による予測もあわせて行った．ステップワイズ重回帰分析は工数予測に広く用いられている方法であり，目的変数と説明変数の関係を一次式で表したモデルを作成して予測を行う．ステップワイズ変数選択法は説明変数を選択して最良の回帰モデルを探索する手法の一つである．ステップワイズ重回帰分析(以下 SR)においても欠損値処理法を適用する必要があるため，リストワイズ除去法と平均値挿入法を用いた．予測結果は，それぞれの欠損率ごとに最も高い性能を示した欠損値処理法のものを採用した．

CF と CBR の予測性能の比較結果を図 2に示す．比較対象とした CBR の評価尺度は，それぞれの欠損率で最も高い予測性能を示したアルゴリズムのものを用いた．CF-OW の評価尺度を見ると，すべての欠損率において，CBR よりも高い予測性能を示した．また，PRED(25)を除き，欠損率が高くなるほど，CF-OW と CBR の予測性能の差が大きくなる傾向が見られた．CF-NA は，欠損率が低いときは最も高い予測性能を示しており，特に MMRE は他の予測手法の約半分であったが，欠損率が高くなると予測性能の低下が大きくなる傾向があった．

ステップワイズ重回帰分析(SR)を見ると，欠損値が存在しない場合，MMRE を除く評価尺度において最も高い性能を示した．しかし，欠損率が 10%以上の場合，CF の評価尺度を下回っており，欠損率が 20%以上の場合，PRED(25)を除き，CBR の評価尺度を下回っていた．欠損率が 50%を超えると，ステップワイズ重回帰分析の評価尺度は極端に悪化した．

5. 考察

CF の予測値計算アルゴリズムの 1 つである Amplified Weighted Sum は，データセットの欠損率に影響を受けやすい可能性がある．欠損率が高い場合，CF-NA の予測性能は低くなっており，また欠損率が増加するとともに，性能が急激に低下している．対照的に，CBR や Order と Weighted Sum アルゴリズムを組み合わせた CF では，予測値計算時に通常の平均を用いており，比較的欠損率に影響を受けていない．

実験結果より，データがランダムに欠損している場

合，欠損値が存在しない場合はステップワイズ重回帰分析を用い，欠損率が30%未満の場合はNormalizeとAmplified Weighted Sumアルゴリズムを組み合わせたCFを用い，欠損率が30%以上の場合はOrderとWeighted Sumアルゴリズムを組み合わせたCFを用いるのが適していると考えられる．

6. まとめ

本論文では，プロジェクト間の類似性に基づく工数予測方法である，協調フィルタリングと事例ベース推論の性能比較を行った．データセットの特性値を一定割合でランダムに欠損させ，工数を予測する実験をした結果，OrderとWeighted Sumアルゴリズムを組み合わせた協調フィルタリングは，すべての欠損率において，CBRよりも高い予測性能を示した．さらに，欠損率が30%未満の場合は，NormalizeとAmplified Weighted Sumアルゴリズムを組み合わせた協調フィルタリングの予測性能が最も高くなった．

今後の課題は，より多様な欠損メカニズムをデータセットに適用して実験を行ったり，CBRの別の予測アルゴリズム[23]とCFの予測性能を比較することである．

文　献

[1] L. Angelis, and I. Stamelos, "A Simulation Tool for Efficient Analogy Based Cost Estimation," Empirical Software Engineering, Vol.5, Issue 1, pp.35-68, 2000.

[2] B. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[3] S. Chulani, B. Boehm, and B. Steece, "Bayesian Analysis of Empirical Software Engineering Cost Models," IEEE Trans. on Software Eng., Vol.25, No.4, pp.573-583, 1999.

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

[5] J. M. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function," Masters Thesis, University of Montreal, 1989.

[6] C. Hayes, P. Cunningham, and B. Smyth, "A Case-Based Reasoning View of Automated Collaborative Filtering," Proc. 4th Int'l Conf. on Case-Based Reasoning, pp.243-248, 2001.

[7] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd, "Experiences Using Case-Based Reasoning to Predict Software Project Effort," Proc. 4th Int'l Conf. on Empirical Assessment & Evaluation in Software Eng., pp.1-34, 2000.

[8] 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.

[9] R. Little, and D. Rubin, Statistical Analysis with Missing Data, 2nd ed., 408pp., John Wiley & Sons, 2002.

[10] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster, "An Investigation of Machine Learning Based Prediction Systems," J. Systems and Software, Vol. 53, Issue 1, pp.23-29, 2000.

[11] 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, Issue 2, pp.163-196, 2003.

[12] I. Myrtveit, and E. Stensrud, "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models," IEEE Trans. on Software Eng., Vol.25, No.4, pp.510-525, 1999.

[13] 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 Trans. on Software Eng., Vol.27, No.11, pp.999-1013, 2001.

[14] Project Management Institute, A Guide To The Project Management Body Of Knowledge (PMBOK Guides), Project Management Institute, 2004.

[15] L. H. Putnam, "A General Empirical Solution to the Macro Sizing and Estimating Problem," IEEE Trans. on Software Eng., Vol.4, No.4, pp.345-361, 1971.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," Proc. ACM Conf. on Computer Supported Cooperative Work, pp.175-186, 1994.

[17] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," Proc. 10th International World Wide Web Conference, pp.285-295, 2001.

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

[19] 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.

[20] K. Strike, K. El Eman, and N. Madhavji, "Software Cost Estimation with Incomplete Data," IEEE Trans. on Software Eng., Vol.27, No.10, pp.890-908, 2001.

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

[22] 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.1155-1164, 2005.

[23] F. Walkerden, and R. Jeffery, "An Empirical Study of Analogy-based Software Effort Estimation," Empirical Software Engineering, Vol. 4, Issue 4, pp.135-158, 1999.