ソフトウェア工学 125－6
（2000．1．14）

認知構造の違いによる生産性の変化を考慮したソフトウェア開発シミュレーションモデル

森川昌平† 花川典子† 松本健一† 井上克郎†* 鳥居宏次†
†奈良先端科学技術大学院大学情報科学研究科
*大阪大学大学院基礎工学研究科
〒630-0101 奈良県生駒市高山町 8916-5
E-mail{shohei-m,noriko-h,matumoto,k-inoue,torii}@is. aist-nara. ac. jp

あらまし 本稿では習熟による認知構造の変化が生産性に及ぼす影響を明らかにするために，既に提案したモデルと認知マップを組み合せた新しいソフトウェア開発シミュレーションモデルを提案する．本モデルでは，作業実施に要求される知識構造を，要素知識と要素知識間の関係（前提知識）を示す有向グラフで構成される認知マップで表す．認知マップの要素知識に開発者の持つ知識量を割当てることで開発者の知識の偏りを表現し，さらに，認知マップの要素知識に作業量を分配することで作業実施に要求される知識の偏りを表現する．又，認知マップ上で表された前提知識は作業実施中の知識獲得効率の計算に影響する．すなわち，前提知識が十分である要素知識獲得は効率良く実施できるが，前提知識が不十分な要素知識は獲得効率が悪い状況を表すことができる．本改良モデルにて，認知構造の異なる２人の開発者が同一作業を実施した場合の開発期間をシミュレーションした結果，改良前に表現できなかった２人の開発者の進捗の違いを明らかにすることができた．

キーワード シミュレーションモデル，習熟，認知構造，認知マップ，前提知識

Software Development Simulation Model based on Variation of Productivity by Difference of Cognitive Structure

Shohei Morikawa† Noriko Hanakawa† Ken-ichi Matumoto† katsuro Inoue†* Koji Torii†
†Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, Japan
*Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
E-mail{shohei-m,noriko-h,matumoto,k-inoue,torii}@is. aist-nara. ac. jp

Abstract This paper proposes a new simulation model which is customized from the "Learning Curve based Simulation Model for Software Development". Cognitive Structures as a cognitive map are incorporated into the model. The cognitive map consists of primitive knowledge and relationship among the primitive knowledge as arrows. The arrows mean relationship of premise knowledge. If developers' knowledge is assigned to the primitive knowledge of the cognitive map, the map shows a difference of developer's cognitive structure. If quantity of activity is assigned to the primitive knowledge, the map shows a difference of activity's cognitive structure. The premise knowledge of the map influences quantity of gain to knowledge to execute an activity. That is, if a developer has sufficient premise knowledge, the developer's progress will be good. Using the customized model, the difference between two developers; one has required knowledge to execute an activity, the other has different knowledge from required knowledge to execute an activity.

key words Simulation model , Learning curve, Cognitive structure, Cognitive Map, Premise knowledg

1. はじめに

近年,顧客が求めるソフトウェアの大規模,多様化に伴い,ソフトウェアの高品質,高生産性を保つのが困難になってきた.ソフトウェア開発者は高品質,高生産性を保つための新技術を次々に学習する必要がある一方,ソフトウェア開発にかける時間も取らなければならない.そのため,十分な学習ができていない技術を使用してソフトウェア開発を行わなければならないことがあり,その場合学習とソフトウェア開発が同時に進行する[1][9].

学習とソフトウェア開発が同時に進行するプロジェクトにおける,開発者の生産性は新技術学習の進み具合によって変化する.プロジェクトにおいてソフトウェア開発と使用する技術の学習を同時に行う場合,プロジェクト開始当初は十分に学習していない技術を使用してソフトウェア開発を行うため進捗が遅い.一方プロジェクト終了間際では,それまで行った作業により新技術の経験を積み,学習が進んでいるためプロジェクト開始当初に比べ進捗が速い.そのため,学習による生産性の変化を表現できないと,学習とソフトウェア開発が同時に進行するプロジェクトの進捗管理を適切に行うことはできない.

そこで,花川らは「習熟を考慮したソフトウェア開発シミュレーションモデル」を提案した[2].このモデルでは作業実施によって新技術の知識を獲得し,それに伴う生産性の変化を計算して進捗と開発期間を予測する.しかし,花川らのモデルでは新技術獲得,すなわち開発者の習熟を知識の量だけで議論した.つまり,作業実行に要求される知識と開発者の持っている知識の差を量であらわし,要求される知識量の方が多い時,開発者にとって作業が難しく,反対の場合は作業が容易であると仮定し,生産性を予測した.従って,知識の内容の差による難しさは表現できなかった.例えば,開発者Aはリスト操作のCプログラムを作成し,開発者Bは文字コード自動判別のCプログラムを作成した.両者とも作成したプログラムの規模は1KLOCで開発期間は3ヶ月だとする.次に両者がリストによる線形探索プログラムをC言語での作成期間を予測する場合,花川らのモデルでは,両者は同じ開発期間と進捗を示す.なぜならば,本モデルは過去の開発期間や規模から求めた知識量だけを議論の対象にしていたので,開発者AとBの知識内容の差を明らかにできなかったからである.当然予測される結果である「開発者Aの方がBよりも早く作成できる.」というシミュレーション結果には導けなかった.

そこで,本研究では知識の内容の差による生産性の違いを表現するために,ソフトウェア開発者の認知構造を考慮して生産性を変化させるソフトウェア開発シミュレーションモデルを提案する.認知構造の表現には認知マップ[8]を使用する.一般にいう認知構造とは,(人が)物事を理解するシステムの構造を指すが[5],本研究においては物事を理解することによって得た「知識」と「知識間の関係」を指す.本来はエキスパートと初心者が築く認知構造は異なることもある.つまり,初心者の勘違い等で,認知構造,すなわち,認知マップに差異が生じる.しかし,本研究ではエキスパートも初心者も同じ認知マップ上で議論する.つまり,エキスパートと初心者の差は,認知マップ上に示されるそれぞれの「知識」をどれだけ習得しているかの違いとする.

本稿ではまず,先行モデルである習熟を考慮したソフトウェア開発シミュレーションモデルについて説明する.次に提案モデルの改良を,認知マップ,認知マップの導入によるサブモデルの改良の順に説明する.そして開発者の知識の偏りにより進捗に差異が生じるシミュレーション例を示し,改良されたモデルの有効性について考察する.関連研究を示し,まとめを述べる.

2. 習熟を考慮したソフトウェア開発シミュレーションモデル

花川らのモデルでは,開発者の持っている知識と作業実施に要求される知識の差から生産性と知識獲得量を求める.つまり,作業を実施することによって知識獲得し,獲得した知識が生産性を向上させる仕組みを表す.知識の差では単純に量,すなわち知識が多いか少ないかだけが議論され,知識の内容には立ち入っていない.作業実施に要求される知識量とその作業量を示す作業モデルと,開発者の知識獲得効率を示す知識モデルと,開発者の知識と要求される知識の差によって変化する開発者の生産性を表す生産性モデルから構成される.

2.1. 作業モデル

作業モデルは作業実施に要求される知識レベル,すなわち難しさと,その作業量の関係を示す．例えば,Cプログラム開発の場合多くの関数を作成するが,それぞれの関数の難しさは異なる．非常に簡単な宣言だけの関数もあれば,複雑な処理の関数もある．非常に容易もしくは困難な関数が少なく,中程度ほどの関数が多いと仮定すると,作業モデルは関数の難しさの分布が正規分布の式となる．(図1参照)

$$W_j(\theta) = w_j \frac{1}{\sqrt{2\pi}\sigma} e^{\left[\frac{(\theta-\mu)^2}{2\sigma^2}\right]}$$

θ:作業実行に要求される知識レベル

$r\,g\,W_j(\theta)$:作業jに含まれる全作業のうち要求される知識レベルがθの作業量

w_j:作業jの総作業量

μ:知識レベルの平均

σ:知識レベルの分散

2.2. 知識モデル

知識モデルは作業実施による開発者の知識獲得の様子を示す．例えば,容易な作業を行う場合,開発者は新しく学習する内容はなく知識は増えない．反対に難しい作業を行う場合,開発者は未知の知識を要求され学習により知識が増加する．ただし,開発者にとって難しすぎる作業の場合は理解困難となり知識獲得量は少ない．開発者の知識獲得の様子を表す知識モデルを以下に示す（図2参照）．

$$L_{ij}(\theta) = \begin{cases} (\theta - b_{ij}) K_{ij} \times e^{-E_{ij}(\theta - b_{ij})} & (b_{ij} \le \theta) \\ 0 & (b_{ij} > \theta) \end{cases}$$

θ:作業実行に要求される知識レベル

$L_{ij}(\theta)$: 開発者iが作業jのうち要求される知識レベルがθの作業を行ったときの知識獲得量

b_{ij}:開発者iの作業jに関する知識レベル

E_{ij}:開発者iの作業jに関する知識獲得効率

K_{ij}:開発者iが作業jを行うときの単位時間あたりの最大知識獲得量

2.3. 生産性モデル

生産性モデルは開発者の知識レベルと作業実行に要求される知識レベルの関係から開発者の生産性の変化を示す．例えば,開発者にとって容易な作業を行う場合は生産性が高く,難しい作業を行う場合は生産性が低い．前述した知識モデルの知識獲得量と反対になっている．生産性モデルをオージブモデル（累積正規分布）[7]を用いて次のように示す（図3参照）．

$$P_{ij}(\theta) = C_{ij} \int_{-\infty}^{a_j(b_{ij}-\theta)} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$$

θ:作業実行に要求される知識レベル

$P_{ij}(\theta)$: 開発者iが作業jのうち要求される知識レベルがθの作業を行ったときの生産性

C_{ij}: 開発者iの作業jを行ったときの最大の生産性

a_j:作業jの実行に要求される作業の厳密さ

b_{ij}: 開発者iの作業jに関する知識レベル

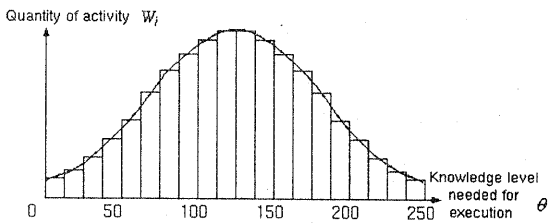

図1 作業モデル

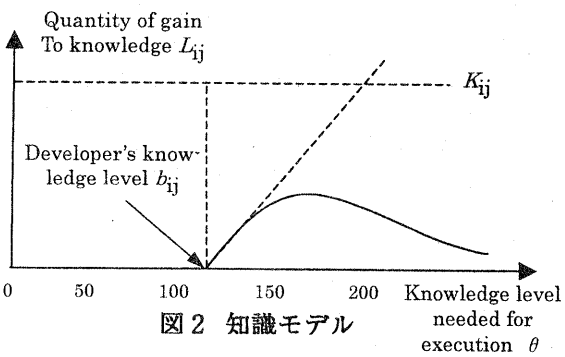

図2 知識モデル

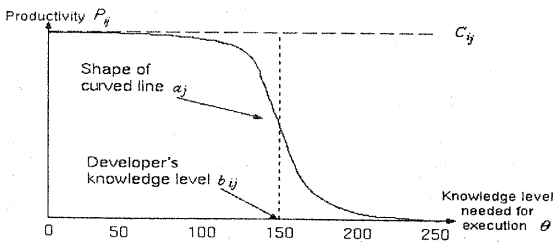

図3 生産性モデル

2.4. シミュレーション手順

作業モデル,知識モデル,及び生産性モデルを用いて以下の手順でシミュレーションし,作業の進捗状況や開発者の習熟の様子を明らかにする.

手順1：各モデルの初期化と時間 t の初期化

手順2：時間 t に実施する作業の選択(θ の決定)

手順3：$P_{ij}(\theta)$の決定

手順4：$L_{ij}(\theta)$の決定

手順5：開発者の知識レベルb_{ij}の更新

$b_{ij}=b_{ij}+L_{ij}(\theta)$

手順6：作業モデルの更新

$W_j(\theta)=W_j(\theta)-P_{ij}(\theta)$

手順7：知識モデルと生産性モデルを更新

手順8：終了判定

3. 提案モデルの改良

3.1. 認知構造の導入

3.1.1. 認知マップ

認知マップとは認知構造をグラフによって表したもので,「要素知識」をノード,「要素知識間の関係」を有向線分によって表したものである.要素知識とは,知識を構成する最小単位の知識とする.例えば図4はC言語の認知構造の一部,特にリスト操作プログラミングに必要と思われる認知マップを表したものである.本研究では,この「要素知識間の関係」は,矢印の元の「要素知識」は矢印の先の「要素知識」の前提知識であることを示すものとする.例えば,図4において動的メモリ割当ての前提知識はポインタの知識となる.

3.1.2. 認知マップにおける作業モデル

認知マップを取り入れると,作業量の配分をより正確に割当てることができる.つまり,提案モデルは,全体の作業量のみを対象としている.図1で示すように,w_j(作業jの総作業量)を θ(作業実行に要求される知識レベル知識レベル)の分散にしたがって配分しただけである.認知マップを導入すると,『全作業量10KLOCのうち,「再帰処理」要素知識を要求される作業は0.5KLOCであり,「動的メモリ割当て」要素知識を要求される作業は1KLOCである.』と全作業量を各要素知識に分配することができる.

さらに,配分されたそれぞれの作業量を用いて各要素知識に作業モデルが定義される(図5参照).すなわち,認知マップ上の要素知識の数だけ再配分された作業量に基づく作業モデルが構築される.

3.1.3. 認知マップにおける生産性モデルと知識モデル

認知マップを導入することによって,各要素知識に知識モデル,生産性モデルを定義できる.つまり,各要素知識に対する開発者の知識量b_{ijg}を設定されるので,開発者のC知識の偏りを表現できる.例えば,図5に示すように,「動的メモリ割当て」要素知識にはb_{ij}(開発者iの作業jに関する知識レベル)が20%,「再帰処理」要素知識にはb_{ij}(開発者iの作

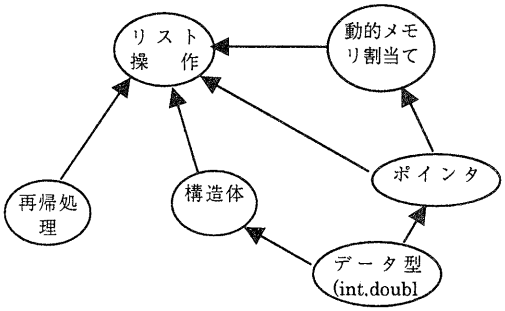

図4 認知マップ

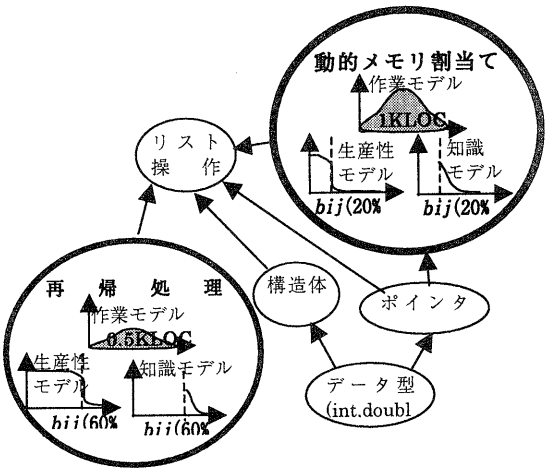

図5 要素知識へ割当てられるの提案モデル

業jに関する知識レベル)が60%と割当てられている．つまり，開発者iは「動的メモリ割当ての知識は少ないが,再帰処理の知識は多い.」という意識の偏りを表現することができる．さらに要素知識ごとに割当てられたb_{ij}（開発者iの作業jに関する知識レベル）を基に知識モデルと生産性モデルを定義する,これによって，要素知識ごとに生産性の変化や知識獲得量の変化を表現することができる．

3.2. 提案モデルの改良

3.2.1. 作業モデル

前節3．1で述べたように，作業モデルは認知マップ上の要素知識ごとに定義される.従って，「作業j」のサフィックスjが「全作業の内,要素知識pを要求される作業」のサフィックスpへ置き換えられる．このサフィックスjからpへの変更は知識モデル，生産性モデルも同様である．

$$W_p(\theta) = w_p \frac{1}{\sqrt{2\pi\sigma}} e^{\left[-\frac{(\theta-\mu)^2}{2\sigma^2}\right]}$$

θ:作業pの実行に要求される知識レベル
$W_p(\theta)$:作業pを必要とし,さらにその知識レベルがθの作業量
w_p:作業pの総作業量
μ:知識レベルの平均
σ:知識レベルの分散

3.2.2. 知識モデル

知識獲得は,認知マップの導入によって「前提知識」の概念が取り入れられる．つまり，ある作業p(全作業のうち,要素知識pを要求される作業)を実施して獲得する知識量L_{ip}は，認知マップ上のリンク（有向線分）であらわされている前提知識の有無も影響する．つまり，これまでの知識モデルでは，作業実行に要求される知識量と開発者の知識量のみで獲得する知識量が計算されていたが，これに加えて,開発者の持っている前提知識の量によって獲得知識量を大きく変化させる．

K'_{ip}が前提知識の有無による影響を表すパラメータである．提案モデルのK_{ij}がK'_{ip}へ置き換えられ，不足する前提知識によってK'_{ip}の値が減少する．つまり，認知マップで示される前提知識のb_{ip}の値が100%であるならば，K_{ip}=K'_{ip}となり，提案モデルと同様の計算結果となる．しかし，不足する前提知識がある場合，K'_{ip}の値は不足の割合に従って減少する．つまり．前提知識が少なすぎる（すなわち作業が難しい）と知識獲得できず，前提知識が十分である（すなわち簡単な作業）と効率よく知識獲得できる．

$$L_{ip}(\theta) = \begin{cases} (\theta - b_{ip}) K'_{ip} \times e^{-E_{ip}(\theta - bip)} & (b_{ip} \le \theta) \\ 0 & (b_{ip} > \theta) \end{cases}$$

$$K'_{ip} = K_{ip} \times \frac{\sum_{\text{前提知識数}} b_{ip}}{100 \times \text{前提知識数}} \times 1/(H_{ip}+1)$$

θ:作業p実行に要求される知識レベル
$L_{ip}(\theta)$: 開発者iが作業pの内，要求される知識レベルがθの作業を行ったときの知識獲得量
b_{ip}:開発者iの作業pに関する知識レベル
E_{ip}:開発者iの作業pに関する知識獲得効率
K_{ip}:開発者iが作業pを行うときの単位時間あたりの最大知識獲得量
H_{ip}:開発者iが作業pに習熟していない前提
要素知識数

3.3. 生産性モデル

生産性モデルも作業モデルと同様にサフィックスjをサフィックスpへ変更する．

$$P_{ip}(\theta) = C_{ip} \int_{-\infty}^{a_p(b_{ip}-\theta)} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$$

θ:作業p実行に要求される知識レベル
$P_{ip}(\theta)$: 開発者iが作業pの内，要求される知識レベルがθの作業を行ったときの生産性
C_{ip}: 開発者iの作業pを行ったときの最大の生産性
a_p:作業pの実行に要求される作業の厳密さ
b_{ip}: 開発者iの作業pに関する知識レベル

3.4. シミュレーション手順の改良

開発者iが作業jを実施した場合の手順を示す．

手順1：作業jの認知マップ作成
手順2：認知マップの各要素知識の作業pに全作業量を配分する．

手順3：認知マップの各要素知識に開発者iの知識レベルを決定する．
手順4：認知マップの各要素知識の作業モデル，知識モデル，生産性モデルの初期化と時間 t の初期化．
手順5：時間 t に実施する作業pを認知マップの要素知識からランダムに選択．
　手順5-1：作業pの中から時間 t に実施する知識レベル θ の選択
　手順5-2：作業pの$L_{ip}(\theta)$の決定
　手順5-3:作業pのb_{ip}の更新 $b_{ip}=b_{ip}+L_{ip}(\theta)$
　手順5-4：作業pの$P_{ip}(\theta)$の決定
　手順5-5：作業モデルの更新
　　$W_p(\theta)=W_p(\theta)-P_{ip}(\theta)$
　手順5-6:更新されたb_{ip}で，作業pの知識モデルと生産性モデルの更新
　手順5-7：作業pの前提知識の知識獲得
　　手順5-7-1：前提知識の$L_{ip}(\theta)$の計算
　　手順5-7-2：前提知識のb_{ip}の更新
　　$b_{ip}=b_{ip}+L_{ip}(\theta)$
　　手順5-7-3：前提知識の知識モデル更新
　　手順5-7-4：前提知識の生産性モデル更新
　　手順5-7-5：前提知識獲得の終了判定，作業pのすべての前提知識の更新終了ならば手順6へ．それ以外は手順5-7-1へ．
手順6：時間 t =時間 t +1
手順7:シミュレーション終了判定，作業jの残作業量（$\Sigma_{要素知識}w_p$）=0ならば，終了．それ以外が手順5へ．

4. シミュレーション例

認知マップを導入し改良したモデルを用いてシミュレーションを行う．想定された状況は，リスト操作プログラムの経験がある開発者Aと文字列操作プログラムの経験がある開発者Bが，リストによる線形探索プログラムを作成した時のシミュレーションを行う．図6がシミュレーションの入力となるC言語の認知マップの一部，特にリスト操作と文字列操作をプログラミングに必要な知識である．シミュレーション手順2,3に従って認知マップの要素知識に開発者の知識レベルと作業量を割当てる．表2には，リスト操作プログラムの経験がある開発者Aと文字列操作プログラムの経験がある開発者Bのそれぞれの割当てた知識量と，新しい作業（リストによる線形探索プログラム）の作業量をそれぞれ要素知識に割当てた値を示す．

図7に開発者Aのシミュレーション途中の認知マップイメージを示す．要素知識の円の中に灰色に塗りつぶされた面積がすでに開発者Aが知っている知識である．開発者Aはリスト操作プログラム経験があるので，「ポインタ」や「動的メモリ割当て」の知識は十分にある．その開発者が「文字列操作」の知識を要求する作業を実施（図7では太線の円で示す）した場合，「文字列操作」の前提知識である「動的メモリ割当」，「文字列」，「文字コード」，「配列」，「データ型」の知識がそれぞれ増え，しかも「文字列操作」の知識も増える（図7では，斜線面積で表す）．増える知識量は前提知識がどれだけ備わっているかに依存して知識モデル中で計算される．

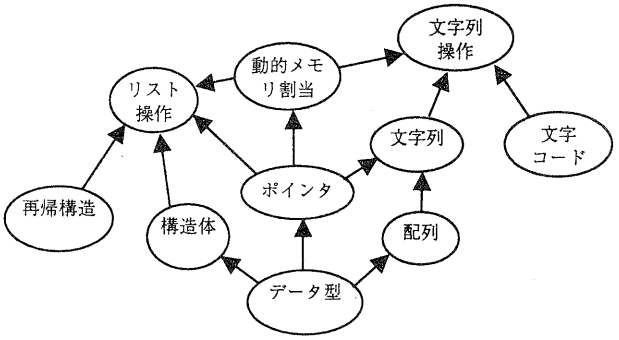

図6 シミュレーション例の認知マップ

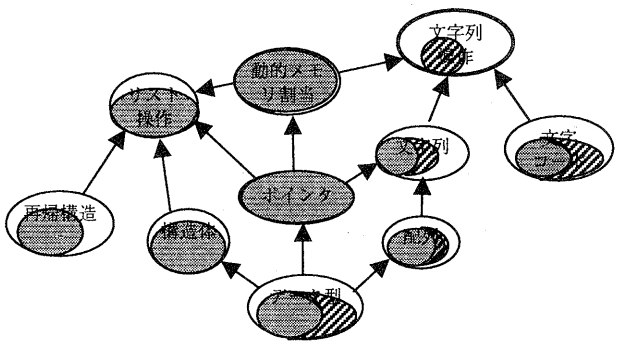

図7 シミュレーション途中の認知マップイメージ

シミュレーション結果を表1に示す．従来のシミュレーションでは開発者の知識の偏りによる進捗の差異や開発期間の違いを表現できず，開発者A，B共に同じ開発期間を予測した．しかし，認知マップを導入することによって，開発者Aが開発者Bより早く作業が完了することを表現できた．

5. 考察

本稿では，習熟を考慮したソフトウェア開発シミュレーションモデルに認知マップを導入し，開発者の知識の偏りによる進捗の違いを表現した．認知マップでの重要な概念は「前提知識」であり，前提知識の有無で知識獲得の効率を大きく影響を受けるとした．

通常，多くの前提知識を必要とする難解な知識は，前提知識がある程度以下であると全く知識獲得できないことは容易に想像できる．そこで，本モデルに「前提知識20%以下であるときは，全く知識獲得できない．」という条件を追加することで，開発者の学習の行き詰まり，そして作業の行き詰まりを表現することができる．この行き詰まりとは「いくら時間をかけても進行（学習）しない」ことであり，他からのイベント（他の人の援助や作業の再割当てなど）を必要とすることを意味する．

したがって，本モデルの計算結果はプロジェクト管理，特に計画作成にて有用である．すなわち，開発者ごとに作業を割当てた場合，本モデルを用いることで，いつ頃，何に行き詰まるかが予測できる．その予測に従って，技術レビューを計画に組み込むことができる．技術レビューの実施時期が本モデルにおいて計算された「開発者の行き詰まる時期」であり，技術レビューの参加メンバーが本モデルで計算された「知識獲得に行き詰まった要素知識」を持つ技術者となる．本モデルを用いることによって，より効率的な技術レビューの計画を組み込むことが可能と考えられる．

また，「行き詰まった開発者」の作業を再配分する場合にも，本モデルの計算結果を用いることができる．開発者が作業が進まない原因となる要素知識（すなわち，要素知識の獲得に行き詰まっている）が本シミュレーションにて明らかになる，したがって，再配分する場合には，原因となる要素知識を要求する作業を他者へ配分することにより，より効率的な作業の再配分が可能となる．

表 1 シミュレーション結果

	開発者A	開発者B
認知マップを用いたシミュレーション結果	305	512
従来のシミュレーション結果	445	445

表 2 認知マップに割り当てた知識量と作業量

要素知識	開発者Aの知識量	開発者Bの知識量	割当作業量
データ型	50%	50%	10
構造体	75%	0%	20
ポインタ	75%	60%	20
配列	0%	60%	0
再帰構造	0%	0%	0
リスト操作	75%	0%	20
動的メモリ割当	75%	0%	20
文字列	0%	60%	5
文字列操作	0%	60%	5
文字コード	0%	60%	0

6. 関連研究

6.1. 提案モデルの関連研究

ソフトウエア開発プロセスのモデル化と評価に対しいくつかの方法が提案されている．Kellnerは，機能，動作，構成の3つの側面から開発プロセスをモデル化し，そのうち主に動作面のモデルに基づいてプロセスシミュレーションを行っている[4]．シミュレーションには，通信ソフトウェア設計用のCASEツールの一つであるSTATEMATEが用いられ，開発工数の見積もりや作業計画の作成が可能となっている．STATEMATEは状態遷移モデルに基づくツールである．従って，状態遷移確率等によって作業効率や作業者の習熟度を表現することは原理的に可能であるが，特に言及はされていない．

Kusumotoらは，一般化確率ペトリネットで作業進捗をモデル化している[6]．ソフトウェア開発を構成する各作業に資源(作業者等)を割当てた上で

ペトリネットの発火機構を利用すると,開発期間,開発工数,及び,作成されるソフトウェアの品質を推定することが可能である．パラメータとして「作業者の経験レベル」がモデルに組み込まれている．「作業者の経験レベル」は,ペトリネットの発火レートや作成されるソフトウェアへのフォールトの混入率を決定する重要なパラメータの一つとされている．但し,その値は作業者毎に与えられる定数であり,シミュレーションにおいて変化することはない．

Iidaらは,多人数での並行作業による開発を記述するためのプロセスモデル,及び,並行作業間の影響を記述するための作業進捗モデルを提案している[3]．彼らのモデルを用いれば,並行作業による開発時の期間と工数の関係を予測することが可能である．作業者の習熟度を表すパラメータとして「スタッフの能力」がモデルに組み込まれている．「スタッフの能力」は,作業効率(単位時間当りの作業進捗)を決定する重要なパラメータの一つとされている．但し,Kusumotoらのモデル同様,その値は作業者毎に与えられる定数であり,シミュレーションにおいて変化することはない．

7. まとめ

本稿では，花川らが提案した習熟を考慮したソフトウェア開発シミュレーションモデルを改良し，認知構造の違いによる生産性の変化を考慮したシミュレーションモデルを提案した．本シミュレーションモデルは認知マップで開発者の知識の獲得状状態（認知構造）を表現することができる．さらに，その認知構造の違いがソフトウェア開発期間に及ぼす影響をシミュレーション例で示した．

今後は実験により本シミュレーションモデルの評価を行う．ただし，実際にソフトウェア開発プロジェクトを行ってみて,シミュレーション結果の見積もりと比較することは困難なので，学習，作業，ともに粒度の小さい問題について実験を行い，その結果に基づいて評価を行う予定である．

文　献

[1]B．Bochenski, "Implementing production -quality client/server systems," John Wiley & Sons,1994.

[2] N．Hanakawa, S．Morisaki, K．Matumoto, "A learning curve based simulation model for software development, " Proceedings of 20th International Conference on Software Engineering, pp. 350-359,1998.

[3] H．Iida, J．Eijima, S．Yabe , K．Matsumoto, K. Torii, "Simulation model of overlapping development process based on progress of activities, " Proceedings of 1996 Asia-Pacific Software Engineering Conference, pp. 131-138,1996.

[4] M. Kellner, "Software process modeling support for management planning and control, " Proceedings of 1st International Conference on Software Process, pp. 8-28,1991.

[5] R. J. Shavelson, "Some aspects of the correspondence between content structure and cognitive structure in physics instruction, " Journal of Educational Psychology, Vol . 33, pp225-234,1972.

[6] S. Kusumoto, O. Mizuno, T. Kikuno, Y. Hirayama, Y. Takagi, K. Sakamoto, "A new software project simulator based on generalized stochastic," Proceedings of 19th International Conference on Software Engineering, pp. 293-302,1997.

[7]芝祐順, 渡部洋, 石塚智一,"統計用語辞典", 新曜社,1980.

[8]竹谷誠,佐々木整, "学習者描画の認知マップによる理解度評価法", 電子情報通信学会論文誌, D-II, Vol. 80, No. 1, pp336-347, 1997.

[9] E. Yourdon, "Object-oriented systems design," Prentice Hall,1994.