インクリメンタル・イテラティブ開発プロセスにおける潜在設計欠陥を考慮したソフトウェア提供期間予測モデル

野中 誠
東洋大学経営学部経営学科
nonaka-m@toyonet.toyo.ac.jp

東 基衞
早稲田大学理工学部経営システム工学科
azuma@azuma.mgmt.waseda.ac.jp

Abstract

本論文では, 反復の並行性を認めたインクリメンタル・イテラティブ開発プロセスを対象に, 設計レビュー以降に発見された設計欠陥が後続反復へ与える影響を考慮した, 最終ソフトウェア製品の提供期間予測モデルを提案する. まず, 予測モデル構築にあたっての前提条件を整理し, これに基づいて予測モデルを定式化した. 次に, 仮想プロジェクトの開発計画を設定し, これに本予測モデルを適用して最終ソフトウェア製品の提供期間を算出した. さらに, 設計工程の品質, 設計レビュー工程の品質および反復の重複率を変動させて感度分析を行い, 仮想プロジェクトの提供期間の変動を分析した. 本予測モデルを用いることで, 設計工程および設計レビュー工程の品質に応じて, 提供期間を最短とする反復の並行化方法を求めることができる.

1 はじめに

近年, スパイラルモデル [1] を源流とするインクリメンタル・イテラティブ開発プロセスが, 多くのソフトウェア開発プロジェクトに適用されている. インクリメンタル・イテラティブ開発プロセスは, イテレーション（以降, 反復と呼ぶ）間での統合作業が発生するため, 全体の作業工数はウォーターフォール型開発プロセスより増加するといった欠点がある [2][3]. しかし, 開発に伴う潜在リスクを実行コードにより順次確認できる [4], 反復の並行化によって最終ソフトウェア製品の提供期間を短縮できる [5], プロジェクト全体の作業負荷を平準化できる [5], 仕様変更に対応しやすいなど多くの利点がある.

インクリメンタル・イテラティブ開発プロセスに適用できる工数見積りおよび提供期間予測モデルとして, COCOMO (Constructive Cost Model) II[6] がある. COCOMO II を適用する際には, 反復間の統合による追加工数を考慮したうえで全体の工数を見積もる. この見積り工数を, 各反復の計画規模に応じて配分することで, 工数および最終ソフトウェア製品の提供期間を予測する. しかしこれは, 次に述べる欠陥修正および仕様変更の影響を十分に考慮したモデルにはなっていない.

後続反復の開始時刻以降に先行反復で欠陥が発見され, その欠陥が後続反復に影響を及ぼすものであった場合, その欠陥に起因する追加の変更作業が後続反復において必要となる. また, 先行反復終了後のソフトウェア試用時に仕様変更が生じた場合にも, 同様の変更作業が後続反復において必要となる. 一般に, 進捗が進んでいるほど手戻りが多くなるため, 変更作業の工数は反復の進捗度に応じて増大する. その結果, 後続反復の開発期間が長くなり, その影響が連鎖的にさらに後続の反復へと影響する. この影響が反復ごとに積み重なり, 最終ソフトウェア製品の提供期間は当初の計画よりも長くなる.

本研究の目的は, 以上に述べた変更作業によるリスク要因を考慮した最終ソフトウェア製品の提供期間を予測することにより, 提供期間を最短とする反復の並行化方法を求めることである. 本論文では, 反復の並行性を認めたインクリメンタル・イテラティブ開発プロセスにおいて, 設計工程で作り込んだ欠陥（設計欠陥と呼ぶ）のうち設計レビューで摘出されなかった設計欠陥（潜在設計欠陥と呼ぶ）が後続反復に与える影響を考慮した, 最終ソフトウェア製品の提供期間予測モデルを提案する. ここで設計欠陥とは, 例えばアーキテクチャの変更や状態空間の設計漏れなど, 設計工程が終了した後に行われる設計上の変更を意味する.

以降では, まず, モデル構築にあたっての前提条件を整理する. この前提条件に基づき, ソフトウェア提供期間の予測モデルを定式化する. 本予測モデルを仮想プロジェクトに適用し, 最終ソフトウェア製品の提供期間を算出する. また, 予測モデルのパラメータを変動させて感度分析を行い, パラメータの各組合せに応じて提供期間を最短とする反復の並行化方法を求める.

2 準備

ここでは, 予測モデルを定式化するうえでの準備として, 本研究が対象としている開発プロセス, 予測モデルの前

提条件および記号の定義について述べる.

2.1 インクリメンタル・イテラティブ開発プロセス

インクリメンタル開発とは, システム全体の一部分を少しずつ追加しながら開発することを意味する. 一方, イテラティブ開発とは, 分析から実装およびテストにいたる一連のプロセス（イテレーション, 反復）を繰り返し実施しながら開発することを意味する. 両者の特徴を併せ持った開発プロセスを, 一般に, インクリメンタル・イテラティブ開発プロセスと呼ぶ. とくに本研究では, 複数の反復を並行して実施できる開発プロセスを対象としている.

本研究が対象とするインクリメンタル・イテラティブ開発プロセスの概略を図 1 に示す. 以下に, その概要を説明する.

図 1 において, 反復 1 の開始前に, 最終ソフトウェア製品に対するすべての要求事項を, どの反復で, 何回の反復で開発するかの開発戦略を立案する. 反復 1 では, 最終ソフトウェア製品に対する要求に基づいて基本アーキテクチャを設計するとともに, 反復 1 に対する要求事項に基づく設計・実装・テストを行う. 各反復の成果物と先行反復までの成果物を統合したものを, 本論文では中間ソフトウェアと呼ぶ. 反復 2 以降では, 依存関係にある先行反復の設計仕様または中間ソフトウェア, ならびに当該反復に対する要求事項に基づく設計・実装・テストを行う. さらに, 先行反復までに作成された中間ソフトウェアに対して, 当該反復の増分を統合する.

各反復は, ウォーターフォール型開発プロセスによって実施されるものとする. 各反復は, 2.2 節で述べる前提条件を満たしていれば, 並行して実施できる. また, 動作可能な中間ソフトウェアであれば, 顧客によって試用される場合がある. このような複数の反復を経ることによって, 最終ソフトウェア製品が出荷される.

MBASE (Model-Based Architecting and Software Engineering)[7] や RUP (Rational Unified Process)[8] は, 本研究が対象としている開発プロセスであるといえる. 軽量開発プロセスと呼ばれるような, 各イテレーション内に明確な工程を持たない開発プロセスは対象としていない.

2.2 予測モデルの前提条件

ソフトウェア提供期間予測モデルを作成するにあたっての, 前提条件を以下に示す.

1. 反復間には, 設計仕様に関して, 先行反復から後続反復への一方向の依存関係がある. ただし, 依存関係のない反復が存在してもよい. また, 依存関係には強度があり, これを本論文では反復間の依存度と呼ぶ.
2. 後続反復を開始できるのは, 依存関係にある先行反復のマイルストーン CDR (Critical Design Review) 終了後である. 反復間に依存関係がない場合は, それらは同時刻に開始してもよい.
3. 潜在設計欠陥は, それが作り込まれた反復内で発見される.
4. ある反復内で発見された潜在設計欠陥は, その反復内において修正される. すなわち, 後続の反復において修正されることはない. またその工数は, あらかじめ反復の開発期間として見積もられているものとする.
5. 先行反復において潜在設計欠陥が発見された場合, その反復と依存関係にある後続反復において変更作業が生じる.
6. ある反復内で発見された潜在設計欠陥は, 修正されると同時に, 依存関係にある後続反復において変更作業が直ちに開始される.
7. 変更作業量は, 依存度および反復の進捗度に応じて増大する.
8. 変更作業によって, 新たな設計欠陥は作り込まれない.
9. 後続反復は, 先行反復よりも後に終了する.
10. 並行して実施される反復には, 異なる資源（要員など）の割り当てが可能である.
11. ある先行反復と依存関係にある後続反復において, 依存度が最大 (100%) と仮定した場合に, 後続反復の進捗度が最大 (100%) の時点で先行反復の潜在設計欠陥が発見されたとする. これに起因する変更作業量を, 本論文では最大変更作業量と呼ぶ. 予測モデルでは, 最大変更作業量の平均値をパラメータとして用いる.

前提条件 11 に関して補足する. 例えばオブジェクト指向ソフトウェア開発において, 先行反復で設計されたクラスの潜在設計欠陥が発見された場合を考える. このとき, その欠陥に対する修正を後続反復の終了直前に行う場合に, 継承関係にある依存度の強いクラスを修正する場合と, 集約関係にある依存度の弱いクラスを修正する場合では, 最大変更作業量は大きく異なる. したがって, 最大変更作業量を確率変数とみなし, 何らかの分布に従っていると仮定することができる. しかし本論文では, 予測モデルの複雑化を回避するために, 最大変更作業量の平均値を予測モデルのパラメータとしている.

2.3 記号の定義

以下に, 本論文で使用する主な記号を定義する.

- N: 計画された反復数.
- $Iter_i$: 第 i 番目の反復.$(1 \leq i \leq N)$
- Dur_i: $Iter_i$ の開発期間の計画値（月）.
- TS_i: $Iter_i$ の開始時刻の計画値（月）.

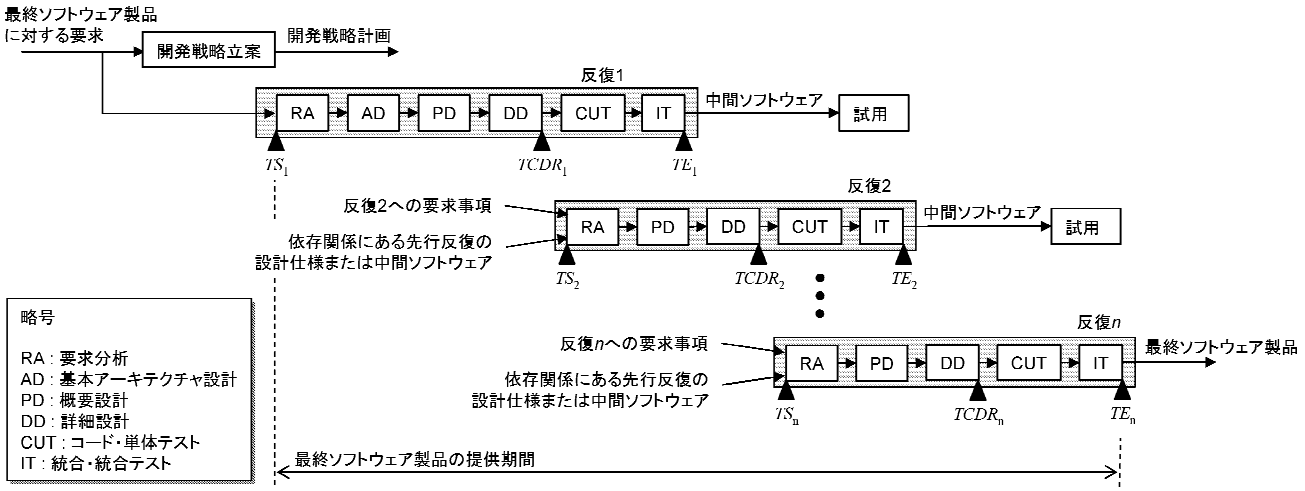

図 1. 対象とするインクリメンタル・イテラティブ開発プロセスの概略

- TE_i: $Iter_i$ **の終了時刻の計画値（月）**.
- $TCDR_i$: $Iter_i$ **の** CDR **終了時刻の計画値（月）**.
- $Size_i$: $Iter_i$ **で実装するソフトウェアの見積り規模**（KSLOC: Kilo Source Lines of Code）.
- DDD_i (Design Defects Density): $Iter_i$ **において** $TCDR_i$ **までに作り込まれる設計欠陥の密度（設計欠陥数/**KSLOC）.
- DDY_i (Design Defects Yield): $Iter_i$ **における** $TCDR_i$ **時点の設計欠陥摘出率（**CDR **までに除去した設計欠陥数/全設計欠陥数）**.
- UDD_i (Unremoved Design Defects): $Iter_i$ **における** $TCDR_i$ **以降の潜在設計欠陥数の推定値**.
- D_{ik}: $Iter_i$ **において** $TCDR_i$ **以降に発見された第** k **番目の潜在設計欠陥**.
- TD_{ik}: D_{ik} **の発生時刻（月）**.
- IDN (Iteration Dependency Network): **イテレーションの依存関係を表す有向ネットワーク**.
- w_{ij}: $Iter_i$ **の** $Iter_j$ **に対する依存度** $(0 \leq w_{ij} \leq 1)$.
- $C_i(t)$: **時刻** t **における** $Iter_i$ **の進捗度** $(0 \leq C_i(t) \leq 1)$.
- $\overline{TW}_{\max\ ij.}$: $D_{ik}(1 \leq k \leq UDD_i)$ **の** $Iter_j$ **に対する最大変更作業量の平均値（月）**.
- ΔTW_{ijk}: D_{ik} **によって生じる** $Iter_j$ **における変更作業量（月）**.

3 提供期間予測モデル

2.2 **節の前提条件に基づいて, ソフトウェアの提供期間予測モデルを以下に定式化する**.

3.1 反復の開発期間

Dur_i **を見積もる手法は任意でよいが, ここでは,** COCOMO II **ポストアーキテクチャモデル** [6] **を用いる方法を述べる.** COCOMO II **では, 式** (1) **を用いて** Dur_i **を見積もる.**

$$Dur_i = 3.67 \times PM^{(0.28+0.2\times(E-0.91))} \quad (1)$$

$$PM_i = 2.94 \times {Size_i}^E \times \prod_{m=1}^{17} EM_m \quad (2)$$

$$E = 0.91 + 0.01 \times \sum_{n=1}^{5} SF_n \quad (3)$$

ここで, PM_i (Person Month) **は** $Iter_i$ **の見積り工数を表しており, 単位は人月である.** EM_m **と** SF_n **は, それぞれ** COCOMO II **に定められた工数乗数と規模要因である**.

TS_i **に関して,** $TS_1 = 0$ **とする.** $TS_i(2 \leq i \leq N)$ **の初期値は, プロジェクト計画者が設定する.** TE_i **の初期値は** $TE_i = TS_i + Dur_i$ **によって与えられる**.

$TCDR_i$ **は, プロジェクト計画者が** Dur_i **に基づいて任意に設定するか, または** Dur_i **に工程間の時間比率を乗**

じて設定する．なお，前提条件 2 より，$w_{ij} < 0$ のとき $TCDR_i \leq TS_j (i < j)$ を満たしていなければならない．

3.7 節で述べる予測モデルにより，Dur_i, TS_i, TE_i および $TCDR_i$ の初期値は，潜在設計欠陥の影響によって順次更新される．更新後の値をそれぞれ Dur'_i, TS'_i, TE'_i および $TCDR'_i$ で表す．

3.2 潜在設計欠陥数

UDD_i は，ソフトウェアの見積り規模，設計欠陥密度および CDR での欠陥未摘出率の積，すなわち，式 (4) により与えられる．

$$UDD_i = Size_i \times DDD_i \times (1 - DDY_i) \qquad (4)$$

DDY_i について，作り込まれた設計欠陥を CDR においてすべて除去することは一般に困難である．TSP (Team Software Process) のような欠陥の早期除去を指向したプロセスの場合でも，コンパイル前の欠陥摘出率[1] の目標値を 75%以上としている [9]．また DDD_i に関連して，TSP では，詳細設計における欠陥作込み率[2] の目標値を 2.0 個/時以下としている [9]．これらの値を参考に，あるいは組織の実績データに基づいて，DDY_i および DDD_i の値を設定し，UDD_i を求める．

3.3 潜在設計欠陥の発見時刻

前提条件 3 および 4 より，$Iter_i$ の潜在設計欠陥は，$TCDR_i$ から TE_i の間にすべて発見・修正される．また，前提条件 6 および 7 より，変更作業量はある反復の進捗度による影響を受ける．進捗度は時刻によって異なるため，ある反復における変更作業量を算出するためには，依存関係にある先行反復における UDD_i 個の潜在設計欠陥について，それぞれが発見される時刻を何らかの仮定に基づいて与える必要がある．

本論文では，$TCDR_i$ 以降の時刻 t における潜在設計欠陥の発見率を，形状パラメータ $m > 1$ のワイブル分布に従うものとする．その理由として，ソフトウェア信頼度成長モデルの研究 [10] から明らかなように，テスト工程の終盤では欠陥の発見が困難になるためである．また，ワイブル分布は，形状パラメータ m の値により様々な分布に適用できることと，実際のデータからパラメータを推定することが容易であるため，実用的な観点から適用しやすい [11]．

$TCDR_i$ 以降の時刻 t における潜在設計欠陥の発見率 $F_i(t)$ を，ワイブル分布の累積分布関数を用いて式 (5) に，そのグラフを図 2 に示す．

[1] コンパイル前に作り込んだ欠陥数に対するコンパイル前に除去した欠陥数の比率．

[2] 1 時間あたりに詳細設計工程で作り込んだ設計欠陥数．

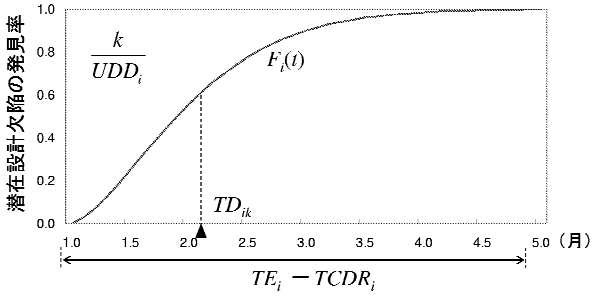

図 2. 潜在設計欠陥の発見時刻の与え方

$$F_i(t) = 1 - \exp\Big(-\frac{(t - \gamma_i)^{m_i}}{t_{0 \cdot i}}\Big) \qquad (5)$$

ここで，m_i は形状パラメータ，$t_{0 \cdot i}$ は尺度パラメータ，γ_i は位置パラメータである．

$Iter_i$ における潜在設計欠陥の発見時刻 TD_{ik} $(1 \leq k \leq UDD_i)$ は，図 2 に示したように，$TCDR_i$ から TE_i までを区間とした式 (5) の累積分布グラフにおいて，累積確率が k/UDD_i と等しい値をとる時刻とする．すなわち，

$$F_i(TD_{ik}) = \frac{k}{UDD_i} \qquad (6)$$

となる TD_{ik} を求める．ただし，$\lim_{t \to \infty} F_i(t) = 1$ であるため，例えば $F_i(TE_i - TCDR_i) = 0.995$ となるように区間を与える必要がある．

3.4 反復間の依存関係

前提条件 1 に基づき，反復間の依存関係を有向ネットワークを用いて表現できる．すなわち，反復の集合 $V = \{Iter_i \,|\, 1 \leq i \leq N\}$ と，反復間の依存関係を表す枝 e_{ij} の集合 $E = \{e_{ij} \,|\, 1 \leq (i, j) \leq N, i < j\}$ について，e_{ij} に重み w_{ij} を対応させる実数値関数 $w : E \mapsto R$ で表す．この場合，反復間の依存関係は有向ネットワーク

$$IDN = (V, E, w) \qquad (7)$$

で表される．

3.5 反復の進捗度

一般に，プロジェクトの進捗状況を追跡する際には，EVMS (Earned Value Management System)[12] が用いられる．EVMS では，計画された個々の作業単位（ワー

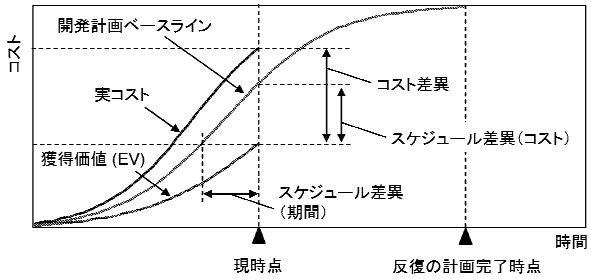

図 3. 開発計画ベースラインと獲得価値

クパッケージ）に対して計画価値 (PV: Planned Value) をあらかじめ与えておき, 各ワークパッケージの完了時刻を計画することで, 開発計画ベースラインを立案する. プロジェクト開始後, ある時刻までに完了したワークパッケージの PV を累積して, 獲得価値 (EV: Earned Value) を算出する. 同時に, 実コストを追跡することで進捗を管理する. 図 3 に, 開発計画ベースライン, EV および実コストとの関係を示す.

開発計画ベースラインおよび EV のグラフが描く形状は, PV および EV の評価方法によって異なるが, 一般に, S 字曲線を描くといわれている. 文献 [13] では, 機能性の実現度に着目し進捗を追跡した結果, EV グラフが S 字曲線を描いた事例が報告されている. このことから, 本研究では, 開発計画ベースラインのグラフをロジスティック曲線で表現し, 反復の進捗度の算出に用いる.

時刻 t における $Iter_i$ の進捗度 $C_i(t)$ は, ロジスティック曲線を用いると式 (8) によって与えられる.

$$C_i(t) = \frac{1}{1 + c_i \cdot \exp(-b_i t)} \tag{8}$$

ここで, b_i, c_i は開発計画ベースラインのグラフを特徴づけるパラメータである.

3.6 後続反復における変更作業量

前提条件 7 および 11 に基づき, 後続反復における変更作業量を, 反復間の依存度, 潜在設計欠陥の発見時における反復の進捗度および $Iter_j$ に対する最大変更作業量の平均値の積で与える. すなわち,

$$\Delta TW_{ijk} = \overline{TW}_{\max\ ij.} \cdot C_j(TD_{ik}) \cdot w_{ij} \cdot \epsilon_{ijk} \tag{9}$$

ここで,

$$\epsilon_{ijk} = \begin{cases} 1 & \text{if} \quad TS_j < TD_{ik} < TE_j \\ 0 & \text{otherwise} \end{cases}$$

である.

3.7 提供期間予測モデル

以上に基づき, ソフトウェア提供期間予測モデルを定式化する. 任意の $n\,(2 \leq n \leq N)$ について, 更新後の提供期間 TE'_n を式 (10) から式 (13) により与える.

$$\begin{aligned} TE'_n &= TS'_n + Dur'_n && (10) \\ TS'_n &= TCDR'_{n-1} \\ &\quad +(TS_n - TCDR_{n-1}) && (11) \\ TCDR'_n &= TS'_n + (TCDR_n - TS_n) \\ &\quad + \sum_{i=1}^{n-1} \sum_{k=1}^{UDD_i} \Delta TW_{ijk} \cdot \delta_{ink} && (12) \\ Dur'_n &= Dur_n + \sum_{i=1}^{n-1} \sum_{k=1}^{UDD_i} \Delta TW_{ijk} && (13) \end{aligned}$$

ここで,

$$\delta_{ink} = \begin{cases} 1 & \text{if} \quad TS_n < TD_{ik} < TCDR_n \\ 0 & \text{otherwise} \end{cases}$$

である. また, $TCDR'_1 = TCDR_1$ である.

式 (11) において, TS'_n に $TS_n - TCDR_{n-1}$ を加算しているのは, 潜在設計欠陥の影響を考慮する前に計画された TS_n と $TCDR_{n-1}$ との差異を, 更新後の TS'_n に反映させるためである. ただし, この値はプロジェクト計画者が任意に設定し直してもよい.

3.8 提供期間の算出手順

最終ソフトウェア製品の提供期間の算出手順を以下に示す.

step 1 開発戦略を立案する.
プロジェクト計画者が, N, $Size_i$ を設定する. その後, COCOMO II などを用いて Dur_i, TS_i, TE_i, $TCDR_i$ を計画する. すなわち w_{ij} を与える. また, 反復間の依存関係を考慮して IDN を設定する. なお, 提供期間の算出時の都合上, $TS_i < TS_j (i < j)$ となるよう反復を並べ替える.

表 1. 仮想プロジェクトの開発計画

反復	$Size$	TS	TE	$TCDR$	Dur
1	43.5	0	16.90	13.40	16.90
2	31.7	13.40	22.40	19.20	9.00
3	36.9	19.20	28.60	25.00	9.40

step 2 パラメータを設定する.
プロジェクト計画者が, 設計欠陥に関するパラメータ DDD_i および DDY_i を設定し, UDD_i を求める. また, 式 (5) における m, t_0, γ, 式 (8) における b_i, c_i および式 (9) における $\overline{TW}_{\max\ ij.}$ を設定する.

step 3 変更作業量を算出する.
各反復において潜在設計欠陥を式 (5) に基づいて擬似的に発生させる. 式 (6) を満たす TD_{ik} を求め, これを各欠陥の発生時刻とする. その後, 式 (9) により ΔTW_{ijk} を求める ($1 \leq (i,j) \leq N$, $i < j$, $1 \leq k \leq UDD_i$).

step 4 ソフトウェア提供期間を算出する.
式 (10) から式 (13) を, n を 2 から N まで変化させて順に算出し, 最終ソフトウェア製品の提供期間を式 (10) により算出する.

4 提供期間の予測例

提案した予測モデルを用いて, ある仮想プロジェクトの最終ソフトウェア製品の提供期間を予測する. また, 感度分析を行い, パラメータの組合せに応じて提供期間を最短とする反復の並行化方法を求める.

4.1 仮想プロジェクトの設定

表 1 は, MBASE の開発プロセスを対象に, COCOMO II を用いて開発計画を立案した例を, 文献 [6] から引用したものである. この例では反復数が 3 となっているが, 本予測モデルではとくに反復数に制約を設けていない. 表 1 のうち, $TCDR_1$ と $TCDR_2$ については, 著者らがそれぞれ TS_2 と TS_3 と等しい値を設定した. また, $TCDR_3$ は著者らが任意の値を設定したが, これは最終反復の値であるため, 提供期間を算出するうえで影響はない.

潜在設計欠陥数, 潜在設計欠陥の発見時刻および反復の進捗度に関するパラメータとして, 表 2 の値を与えた. ここで $Iter_n$ における $t_{0\cdot n}$ の値は, 先行反復で発見された設計欠陥に起因する変更作業量を加算したうえで, 式 5 において $F_n(TE_n - TCDR_n) = 0.995$ となるようにパラメータ推定した値である. また, w_{ij} は 0.8, $\overline{TW}_{\max\ ij.}$ は 0.1 に一律に設定した ($1 \leq (i,\ j) \leq 3$, $i < j$).

表 2. 仮想プロジェクトのパラメータ

	設計欠陥		欠陥発見時刻			進捗度	
反復	DDD	DDY	m	t_0	γ	b	c
1	3.0	0.8	1.5	1.24	1.0	0.6	40
2	3.0	0.8	1.5	1.08	1.0	1.0	40
3	3.0	0.8	1.5	1.29	1.0	1.1	40

表 3. 提供期間の算出結果

反復	TS	TE	$TCDR$	Dur	UDD
1	0	16.90	13.40	16.90	25
2	13.40	24.39	21.19	10.99	18
3	21.19	32.03	28.43	10.84	21

4.2 提供期間の算出結果

本仮想プロジェクトを対象に, 最終ソフトウェア製品の提供期間を算出した. その結果を表 3 に示す. 表 1 の反復 3 における TE の値は 28.60ヶ月であったが, 本予測モデルによって算出された提供期間は 32.03ヶ月であり, 3.4ヶ月の遅れが生じると予測している.

ここで, 表 2 に示した初期条件のうち, 各反復の設計欠陥に関するパラメータを, 一例として $(DDD, DDY) = (5.0, 0.6)$ へと変更する. この条件における提供期間の算出結果を表 4 に示す. この条件下では提供期間が 40.39ヶ月となり, 表 1 の当初計画に比べて 11.8ヶ月, 表 3 に比べて 8.4ヶ月の遅れが生じると予測している. これは, 設計工程の品質 (DDD) および設計レビュー工程の品質 (DDY) が低下して潜在設計欠陥数が多くなることで, 最終ソフトウェア製品の提供期間が大幅に遅くなることを表している.

4.3 設計欠陥に関する感度分析

4.2 節で一例を示したように, 最終ソフトウェア製品の提供期間は, 予測モデルのパラメータの値によって大きく異なる. 以降では, 設計欠陥密度, 設計欠陥摘出率および反復間の重複率について, それぞれ値を変化させた場合の提供期間の変化を示す.

表 4. 提供期間の算出結果（設計欠陥の条件を変更）

反復	TS	TE	$TCDR$	Dur	UDD
1	0	16.90	13.40	16.90	86
2	13.40	29.23	26.03	15.83	62
3	26.03	40.39	36.79	14.36	72

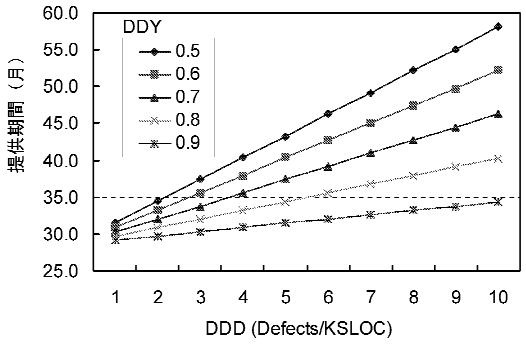

図 4. 設計欠陥に関する感度分析

図 4 に, 設計欠陥密度と設計欠陥摘出率に関する感度分析の結果を示す. 図 4 より, DDD が増加するにつれて提供期間は線形に増加しており, DDY が低いほどその増加率が大きいことがわかる. また, 設計工程の品質と設計レビュー工程の品質がいずれも高い水準であれば, 開発期間の遅延は比較的小さく抑えられることを示している. 例えば, 図 4 において $(DDD, DDY) = (1.0, 0.9)$ の提供期間は 29.2ヶ月であり, 表 1 の初期計画と比べて 0.6ヶ月の遅延に抑えられている.

4.4 反復の重複率に関する感度分析

次に, 反復の重複率を変化させた場合の感度分析を行う. 反復 n の反復 $n-1$ に対する重複率 $OL_n(OverLap)$ を, 式 (14) で与える.

$$OL_n = \frac{TE_{n-1} - TS_n}{TE_{n-1} - TCDR_{n-1}} \tag{14}$$

ただし, $w_{n \cdot n-1} > 0$, $2 \leq n \leq N$ とする.

$OL_n = 1$ は, $TCDR_{n-1}$ と同時刻に $Iter_n$ が開始されることを表している. また, $OL_n = 0$ は, TE_{n-1} と同時刻に $Iter_n$ が開始されることを表している. 例えば, 表 1 の開発計画では, $Iter_2$ および $Iter_3$ の重複率はいずれも 1 である.

図 5 と図 6 に, DDD がそれぞれ 3.0 と 10.0 の場合の, 反復の重複率に関する感度分析の結果を示す. ここで, 図中の反復率は $OL_2 = OL_3$ としている. DDD および DDY の値の組合せによって描かれる各グラフにおいて, 提供期間を最短とする重複率が, 求める反復の並行化方法である. この重複率に基づいて反復を並行化すれば, 最短で最終ソフトウェア製品を提供できると予想される.

これらの図は, 本仮想プロジェクトにおいて, DDY が低い場合に反復の重複率を高めると初期計画よりもかえって提供期間が遅延する場合があるが, DDY が高い場合

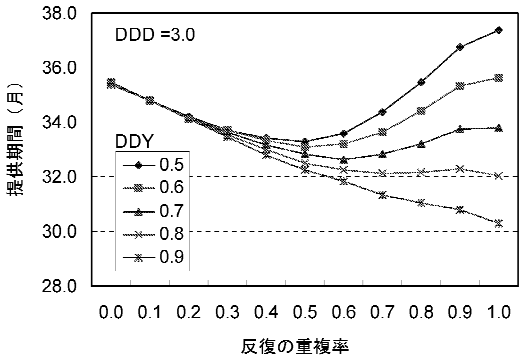

図 5. 反復の重複率に関する感度分析 ($DDD = 3.0$)

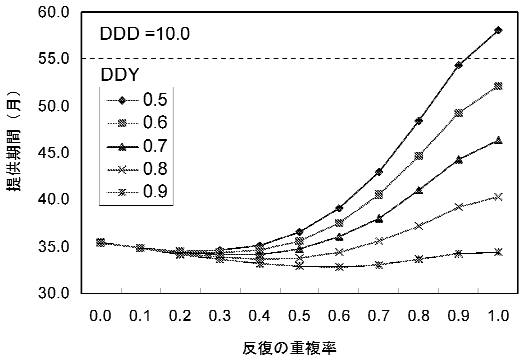

図 6. 反復の重複率に関する感度分析 ($DDD = 10.0$)

には, 反復の重複率を高めることで提供期間が短縮できることを示している. しかし, DDD が高い場合には, 重複率が高いからといって提供期間が最短にならないことも示している. また, 重複率が低い (0.2 以下) 場合は, 後続反復が開始される前にほとんどの潜在設計欠陥が先行反復で取り除かれているため, DDY の値による差異がほとんど現れていない.

5 考察

一般に, プロセスを改善すれば, プロダクトの品質が向上するとともに, 提供期間およびコストについても改善できると考えられている. 本論文では, プロセス改善により設計工程の品質および設計レビュー工程の品質を向上させれば, 提供期間が短縮できることを数理モデルにより示すことができた. また, 反復を並行化させることによって, 提供期間をさらに短縮化できる可能性があることを示すことができた.

設計工程の品質および設計レビュー工程の品質が低い場合に, 提供期間が遅延することは一般に推測できることである. 本予測モデルを用いることで, これら工程の品質がどの程度の場合に, どの程度の遅延が生じるのかを求めることができる. これは, プロジェクトの開発計画を立案する際に有益な情報になると考えられる.

本予測モデルで用いるパラメータは, 設計欠陥, 反復間の依存度および反復の進捗度の 3 つに分類できる. これらのうち, 設計欠陥に関しては, さらに, 欠陥密度, 設計レビュー時の欠陥摘出率, 設計レビュー以降の欠陥発見時刻および変更作業量の 4 つのメトリクスに分類できる. これらの測定値を実プロジェクトで収集することは, メトリクスプログラムを導入している組織であれば比較的容易であると考えられる. 実プロジェクトのデータを用いてパラメータ推定することにより, 本予測モデルを実プロジェクトの開発計画時に活用できると考えられる.

設計工程の品質および設計レビュー工程の品質を改善するためには, 一般に, それぞれの工程において追加の工数を伴う. 本予測モデルでは, この追加工数の増加を考慮したモデルとなっていない. したがって, これらの工程の品質を改善した場合に, 本論文で示した感度分析の結果よりも提供期間が長くなる可能性がある. しかし, 生産性の低下を伴わずに品質を向上させることも不可能ではない [14]. そのようなプロセス改善を行うことができれば, 本予測モデルによる感度分析の結果に近い提供期間が実現できると考えられる.

本予測モデルの前提条件には, 実プロジェクトに即していないものがある. 例えば前提条件 10 について, 実プロジェクトでは投入できる要員数が大きな制約となる. また, 本予測モデルでは各反復に遅延を生じさせるモデルとなっているが, 一般にタイムボックス型と呼ばれるような, 各反復での遅延を認めない開発プロセスも存在する. 今後, これらの条件を本予測モデルに取り込み, より実情に合った予測モデルへと発展させる必要がある.

6 おわりに

本論文では, 反復の並行性を認めたインクリメンタル・イテラティブ開発プロセスを対象に, 設計レビュー以降に発見された設計欠陥が後続反復へ与える影響を考慮した, 最終ソフトウェア製品の提供期間予測モデルを提案した. 本モデルを仮想プロジェクトに適用して提供期間を予測し, 感度分析を行った. その結果, 設計工程の品質および設計レビュー工程の品質に応じて, 提供期間を最短とする反復の並行化方法を求められることを示した.

本予測モデルを構成する分布モデルおよびパラメータは, いずれも仮説に基づいたものである. 今後, 実プロジェクトのデータを収集するなどして, これらの推定を行う必要がある. また, 対象とする開発プロセスおよび前提条件を緩和させることで, 様々な開発プロセスモデルに対する提供期間予測へと発展させたい.

参考文献

[1] Boehm, B. W., “A Spiral Model of Software Development and Enhancement”, *IEEE Computer*, Vol. 21, No. 5, pp.61–72, 1988.

[2] Aoyama, M., “Agile Software Process and Its Experience”, *Proceedings of 20th ICSE*, pp.3–12, 1998.

[3] Pittman, M., “Lessons Learned in Managing Object-Oriented Development”, *IEEE Software*, Vol. 10, No. 1, pp.43–53, 1993.

[4] **藤井拓**, **上林弥彦**, “**反復型開発プロセスにおいて不明点に起因する生産性低下を抑制する戦略**”, **情報処理学会** OO 2002 **シンポジウム**, pp.113–116, **近代科学社**, 2002.

[5] Aoyama, M., “Concurrent-Development Process Model”, *IEEE Software*, Vol. 10, No. 4, pp.46–55, 1993.

[6] Boehm, B. W. et. al., *Software Cost Estimation with COCOMO II*, Prentice Hall PTR, 2000.

[7] Boehm, B., Port, D., Egyed, A. and Abi-Antoun, M., “The MBASE Life Cycle Architecture Milestone Package: No Architecture Is An Island,” *1st Working International Conference on Software Architecture*, 1999.

[8] Kruchten, P., *The Rational Unified Process: An Introduction*, 2nd Ed., Addison-Wesley, 2000 (**藤井拓監訳**, **ラショナル統一プロセス入門**, **第 2 版**, **ピアソンエデュケーション**, 2001).

[9] Humphrey, W. S., *Introduction to the Team Software Process*, Addison-Wesley, 1999.

[10] **山田茂**, **ソフトウェア信頼性モデル**, **日科技連**, 1994.

[11] **佐々木正文**, **基礎システム工学**, **共立出版**, 1972.

[12] Fleming, Q. W. and Koppelman, J. M., *Earned Value Project Management*, 2nd Ed, Project Management Institute, 2000.

[13] Ze’evi, M., “Using Earned Value to Track Software Projects”, *Proceedings of 12th ICSQ*, American Society for Quality, 2002.

[14] Hayes, W. and Over, J. W., “The Personal Software Process(PSP): An Empirical Study of the Impact of PSP on Individual Engineers”, CMU/SEI-97-TR-001, 1997.