ソフトウェア工学 116－3
（1997．9．11）

ソフトウェアプロセスのツール化状況を計測するメトリクスとその利用方法

池田 邦彦　西山 哲人　丹羽 徹　仲島 晶

オムロン株式会社 技術本部 IT研究所

〒617 京都府長岡京市下海印寺

E-mail: Kunihiko_Ikeda@omron.co.jp

ソフトウェアプロセスのツール化により，ソフトウェアの品質と生産性の向上が期待できる．しかし，開発部門における適切なツール化は容易ではない．その理由として，1)ツールが前提とする利用手順と既存のソフトウェアプロセスとの不整合，2)ソフトウェアプロセスのツール化による効果を計測することの難しさ，といった問題がある．本稿では，ソフトウェアプロセスの構成単位として，個々の成果物と1:1対応する単位プロセスを考え，成果物の記載項目に対してツールの機能を対応させることによるソフトウェアプロセスのツール化手順を定義する．更に，ソフトウェアプロセスのツール化状況を計測するメトリクスとその利用方法を提案する．本方法により，ソフトウェアプロセスのツール化状況を計測しながら，段階的なツール化を制御できる．

A Method of Measuring and Controlling the Introduction of Tools to the Software Process

Kunihiko Ikeda, Tetsuto Nishiyama, Toru Niwa and Akira Nakajima

OMRON Corporation
Information Technology Research Center

Shimokaiinji, Nagaokakyo-City, Kyoto, 617 Japan

The introduction of tools to the software process is expected to improve the quality and productivity of software development. However, this is not easily accomplished. The reasons are as follows: 1)Procedures for using tools does not necessarily follow the current software process. 2)It is difficult to measure the total amount of the software process that has been automated. In this paper, we define process units which have a one-to-one relation with the various software products and we propose metrics to measure the status of the introduction of tools to the software process by relating tool functions with corresponding product items. By using this method, we can measure and control the gradual introduction of tools to the software process.

1 はじめに

ソフトウェアプロセスへのツールの導入(ツール化)により，ソフトウェアの品質と生産性の向上が期待できる．しかし，実際に開発部門において適切なツール化を行うことは容易ではない．開発部門に導入するツールの選定は通常，SEPG(Software Engineering Process Group)[1]，プロジェクトリーダーらにより行われる．ツール選定においては，ツールの機能，性能，適用範囲，使いやすさ，費用，保守契約内容等が考慮される．こうして採用されたツールについても，ツールが前提とする利用手順と組織で使用しているソフトウェアプロセスとの不整合が起こる可能性がある．この不整合により，ツールが使用されない，あるいは，ツールの無理な使用により開発部門が混乱する，といった結果を招く．このような事態を避けるために，ツール利用手順とソフトウェアプロセスとの整合化を行う必要がある．整合化のためにソフトウェアプロセスを変更する場合，組織または作業者を混乱させないために，ソフトウェアプロセスの変更を徐々に行うことが望ましい．

また，ソフトウェアプロセスのツール化において，ツール化による効果を計測するためのメトリクスが無ければ，1)データに基づくツール化の状況把握と制御ができない，2)品質や生産性についての定量的な議論ができない，3)従って，ツール化に関する経営層の支持を得にくい，といった結果を招く．よって，ツール化による効果を計測するメトリクスとその利用方法を定義する必要がある．

文献[1]では，ソフトウェアプロセスのツール化について，主に管理的側面から方針の概要が述べられている．しかし，ソフトウェアプロセスのツール化を行うための具体的な手順までは明らかにしていない．

文献[2]では，世の中で行われているソフトウェア開発環境に関する主要な研究を体系的に紹介している．従来の研究は，環境のフレームワーク(器)，または，その中で管理される対象(中身)の形式に関する研究が中心であった．しかし，実際に環境を構築していくには，現状の環境への新たなツールの組み込み等により，環境を徐々に進化させていくことになる．実務においてソフトウェアプロセスのツール化を行うためには，この「環境の進化」という動的な観点からの考察は必須である．

ツール化に関するメトリクスの研究としては，文献[3]においてツール化率をツール数とプロセス数に基づいて定義している．しかし，ある1つのプロセスにおいても部分的なツール化が行われることから，ツール化率について，成果物の記載項目までの詳細化を検討する必要がある．

本稿では，ソフトウェアプロセスの構成単位として，個々の成果物と1:1対応する単位プロセスを考え，成果物の記載項目に対してツールの機能を対応させることによるソフトウェアプロセスのツール化手順を定義する．更に，ソフトウェアプロセスのツール化状況を計測するためのメトリクスとその利用方法を提案する(ツール化の効果を計測するメトリクスについては，本稿で提案するメトリクスとソフトウェアの品質と生産性との関係を明らかにすることにより，考案する予定である)．本方法により，1)ツールが前提とする利用手順と既存のソフトウェアプロセスとの不整合がある場合にも適切なツール化が可能になり，2)ツール化による効果に関する議論に必要となるツール化状況の計測が可能になった．

以降，2.では，ソフトウェア開発環境モデルについて述べる．3.では，ソフトウェアプロセスにツールを組込む手順について述べる．4.では，ソフトウェアプロセスのツール化状況を計測するためのメトリクスについて述べる．5.では，提案する方法の適用事例を述べる．6.では提案する方法について考察する．7.では，まとめと今後の課題について述べる．

2 ソフトウェア開発環境モデル

ソフトウェアプロセスのツール化について議論するための前提として，ソフトウェア開発環境を，ソフトウェアプロセス，成果物，ツールの関係に着目してモデル化する(文献[4]では，本章のモデルを更に詳細化している)．

図1は，ソフトウェア開発環境モデルをOMT法[5]を用いて記述したものである．図1において，ソフトウェアプロセスは，複数のプロセスにより，構造化される．プロセス自身もまた，複数のプロセスにより，構造化することができる．このように，ソフトウェアプロセスは木構造になる．木構造の各終端(葉)を単位プロセスと呼ぶ．単位プロセスには，成果物が1:1対応している．

成果物とは，計画書，仕様書，ソースコード，オブジェクトコード，品質管理帳票等，ソフトウェア開発活動において作成される文書類の総称である．成果物は複数個の記載項目から構成される．成果物には，そ

れを作成する際に利用するツールを対応づけることができる．ツールは複数個の細粒度の機能により構成される．

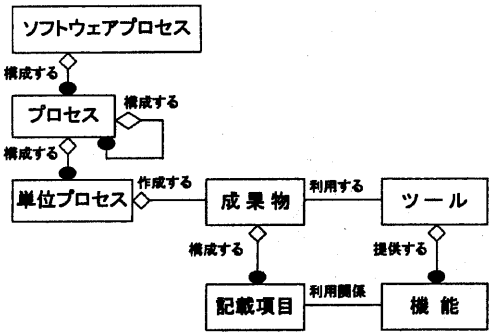

図 1：ソフトウェア開発環境モデル

3 ソフトウェアプロセスのツール化

ソフトウェアプロセスの変更による開発者の負担を軽減するために，成果物の記載項目単位で，ソフトウェアプロセスのツール化を行う手順を提案する．

3.1 ツール化の作業フロー

提案するツール化手順では，図 2 に示したツール化の基本サイクルを繰り返しながら行われる．

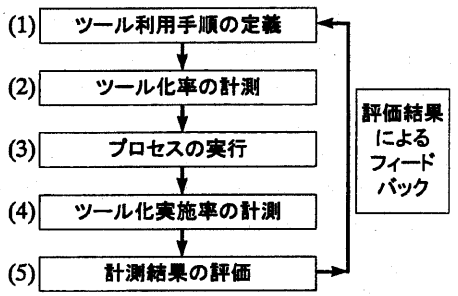

図 2：ツール化の基本サイクル

(1)ツール利用手順の定義では，ツールを用いて作成される成果物への適用範囲，ツールを組込むプラットフォームの特徴，技術移転の手段，支援体制，成果物の品質や作業効率を計測するメトリクスの定義を行う．

(2)ツール化率の計測では，ソフトウェアプロセスがどの程度ツール化されているか(ツール化率)を計測する．(ツール化率の詳細は次章で述べる．)

(3)プロセスの実行では，ツール化されたソフトウェアプロセスを開発部門において実行する．

(4)ツール化実施率の計測では，プロセスの実行後に実際にツールがどの程度利用されたか(ツール化実施率)を計測する．(ツール化実施率の詳細は次章で述べる．)

(5)計測結果の評価では，ツール化率とツール化実施率の計測値から，ツール化が適切であったかどうか，また，今後の作業方針について検討する．評価結果は次サイクルのツール利用手順の定義に反映される．(評価方法の詳細は次章で述べる．)

3.2 ツール化の形式的定義

1つの成果物に対して部分的なツール化を繰り返し行うために，ツール化の形式的な定義を行う．

作成すべき成果物 P_n の記載項目の集合を

$$I_n = \{i_1, i_2, i_3, ..., i_m\}$$

とする．ここで，I_n の要素の粒度は，成果物 P_n の章，節，副節，副々節等の文書構造の深さから決まる．通常，I_n の要素名は，章，節，副節，副々節等の見出しになる．

次に，導入しようとするツール T_j の提供する機能の集合を

$$F_j = \{f_1, f_2, f_3, ..., f_k\}$$

とする．ここで，F_j の要素の粒度は，ツール T_j により作成できる成果物により決まる．

ツール T_j が成果物 P_n の作成に利用できるツールの候補であるかどうかは，成果物作成の目的とツールの機能概要との比較により判断する．

ある記載項目 i の作成に機能 f を利用できるとき，i と f の関係を(i, f)と表記する．

すると，I_n と F_j とのツール利用関係 R_{nj} が次のように定義できる．

$R_{nj} = \{(i, f) \mid i \in I_n$ の作成に利用できる $f \in F_j$ が存在する $\}$

ツール利用関係 R_{nj} の定義域 dom R_{nj} と値域 ran R_{nj} はそれぞれ次のように定義できる[6]．

dom $R_{nj} = \{i \in I_n \mid (i, f) \in R_{nj}$ をみたす $f \in F_j$ が存在する $\}$

ran $R_{nj} = \{f \in F_j \mid (i, f) \in R_{nj}$ をみたす $i \in$

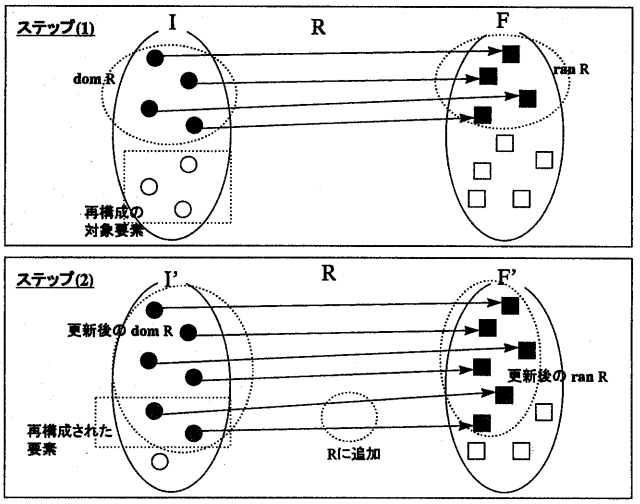

図 3：成果物の記載項目集合(I)とツールの機能集合(F)との関係付け

I_n が存在する }

以上より，「ツール化とは I_n と F_j とのツール利用関係 $R_{n,j}$ を定義する行為である．」と定義する．

3.3 成果物とツールとの関係付け

成果物の記載項目集合(I)とツールの機能集合(F)との関係付けの手順を説明する．図3は本節で述べる手順の概念図である．

(1) I_n の部分集合である dom $R_{n,j}$ の作成にツール T_j の機能 ran $R_{n,j}$ を利用する．

(2) ツール化しなかった部分 I_n − dom $R_{n,j}$ に対して，ツール利用関係 $R_{n,j}$ の要素が少なくとも1個存在するように記述形式を再構成する．

(3) I'_n 及び $R'_{n,j}$ に対して(1), (2)を繰り返し行う．

(4) I_n − dom $R_{n,j}$ = ϕ となるか，または，ツール利用関係 $R_{n,j}$ の要素が存在するように記述形式を再構成できなくなれば，終了する．

4 メトリクス

ソフトウェアプロセスがどの程度ツール化されているか，また，ツール化が適切であるかの評価に使用する2種類のメトリクスを定義する．

4.1 ツール化率

単位プロセスにはある成果物が 1:1 対応しているので，成果物 P_n の全記載項目 I_n に対する，ツール T_j を利用して作成される項目 dom $R_{n,j}$ の(重み付けされた)割合を「単位プロセスツール化率(X)」と定義できる．

$$X = \frac{\sum_{k=1}^{N} a_k \cdot i_k}{\sum_{k=1}^{N} a_k}$$

ただし，

$$i_k = \begin{cases} 1 & (i_k \in \mathrm{dom}R_{n,j}) \\ 0 & (i_k \notin \mathrm{dom}R_{n,j}) \end{cases}$$

重み a_k は，記載項目の記述量，難易度に比例して決まる定数である．重み a_k を用いることにより，1個の記載項目のツール化においても，記述量または難易度が大きい項目のツール化率の値が大きくなる．また，N は各成果物の記載項目の総数である．

同一プロセスに含まれる単位プロセスの単位プロセスツール化率の(重み付けされた)平均値を「プロセスツール化率(Y)」と定義する．

$$Y = \frac{\sum_{k=1}^{N'} b_k \cdot X_k}{\sum_{k=1}^{N'} b_k}$$

重み b_k は，成果物の記述量に比例して決まる定数である．また，N'は同一プロセスに含まれる単位プロセスの総数である．

ソフトウェアプロセスを構成するすべてのプロセス

のプロセスツール化率の(重み付けされた)平均値を「ソフトウェアプロセスツール化率(Z)」と定義する.

$$Z = \frac{\sum_{k=1}^{N''} c_k \cdot Y_k}{\sum_{k=1}^{N''} c_k}$$

重み c_k は，プロセスの工数比率に応じて決まる定数である．また，N"はソフトウェアプロセスを構成するプロセスの総数である.

ソフトウェアプロセスのツール化の達成目標として，ツール化率に組織ごとの具体的な値を設定することができる.

4.2 ツール化実施率

ツール化されたプロセスにおいて，実際に開発者がどの程度ツールを使用しているかを評価するためのメトリクスを定義する.

単位プロセスに 1:1 対応する成果物が実際にどれだけツールを用いて作成されたかの割合を「単位プロセスツール化実施率(α)」とする.

$$\alpha = \frac{\sum_{k=1}^{N} a_k \cdot i_k}{\sum_{k=1}^{N} a_k}$$

ただし,

$$a_k = \begin{cases} \text{定数}(\neq 0) & (i_k \in \mathrm{dom}R_{n,j}) \\ 0 & (i_k \notin \mathrm{dom}R_{n,j}) \end{cases}$$

$i_k \in \mathrm{dom}\, R_{n,j}$ については,

$$i_k = \begin{cases} 1 & (i_k \text{の作成にツールを使用}) \\ 0 & (i_k \text{の作成にツールを未使用}) \end{cases}$$

同一プロセスに含まれる単位プロセスの単位プロセスツール化実施率の(重み付けされた)平均値を「プロセスツール化実施率(β)」と定義する.

$$\beta = \frac{\sum_{k=1}^{N'} b_k \cdot \alpha_k}{\sum_{k=1}^{N'} b_k}$$

ただし,

$$b_k = \begin{cases} \text{定数}(\neq 0) & (\alpha_k \neq 0) \\ 0 & (\alpha_k = 0) \end{cases}$$

ソフトウェアプロセスを構成するすべてのプロセスのプロセスツール化実施率の(重み付けされた)平均値を「ソフトウェアプロセスツール化実施率(γ)」と定義する.

$$\gamma = \frac{\sum_{k=1}^{N''} c_k \cdot \beta_k}{\sum_{k=1}^{N''} c_k}$$

ただし,

$$c_k = \begin{cases} \text{定数}(\neq 0) & (\beta_k \neq 0) \\ 0 & (\beta_k = 0) \end{cases}$$

4.3 メトリクスの評価方法

計測されたツール化率とツール化実施率の評価方法を述べる.

ツール化率及びツール化実施率の基準値をそれぞれ，p, q とする(図 4 において，太い実線が基準値を表す).基準値の具体的な値は，提案した方法を用いて継続的なデータ収集と分析を行うことにより，適切な値を求める.

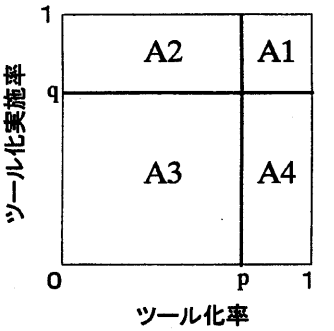

図 4：メトリクスの評価基準

ツール化率及びツール化実施率の計測値をそれぞれ s, t とするとき，領域 A1, A2, A3, A4 をそれぞれ以下のように定義する(図 4 参照).

$A1 = \{(s,t) \mid p \leqq s \leqq 1, q \leqq t \leqq 1\}$

$A2 = \{(s,t) \mid 0 \leqq s < p, q \leqq t \leqq 1\}$

$A3 = \{(s,t) \mid 0 \leqq s < p, 0 \leqq t < q\}$

$A4 = \{(s,t) \mid p \leqq s \leqq 1, 0 \leqq t < q\}$

以下では，計測結果 (X, α) の評価方法について説明する((Y, β), (Z, γ) についても，同様の評価方法を適用できる).

(X, α) ∈ A1 ならば，ツール化は成功している．

(X, α) ∈ A2 ならば，ツール化は順調であり，次段階のツール化では，更にツール化率を高くすることを検討すべき．

(X, α) ∈ A3 ならば，ツール化率が低かったにも関わらず，ツール化に失敗した．ツールが不適当であった可能性や他の要因について再確認すべき．

(X, α) ∈ A4 ならば，ツール化率が高いために，ツール化されたソフトウェアプロセスが開発部門に浸透し切れなかった．今後は，ツール化率を低くしてツール化をやり直すことを検討すべき．

5 適用事例

提案する方法の適用事例を述べる．

市販のオブジェクト指向分析・設計ツール Rational Rose/C++(以下では対象ツールと記す) [7], [8]をソフトウェアプロセスに導入する．

筆者らは，これまで使用してきたウォーターフォール型開発モデルとの整合性を保ちつつ，オブジェクト指向開発方法論[9], [10]によるソフトウェアプロセスを定義した．具体的には，要求分析，概念設計(システムの基本構成，ハードウェアとソフトウェアの切り分けを定義)，機能設計(ソフトウェア構成，機能の詳細定義)，構造設計(実装方法の詳細定義)，プログラミング，単体デバッグ(構造設計に対応したテスト)，機能デバッグ(機能設計に対応したテスト)，システムデバッグ(概念設計に対応したテスト)，テスト(出荷テスト)の順である(各プロセスの名称は，当社において独自に定義したものである)．

対象ツールの適用可能な成果物は，機能設計において作成される「ソフトウェア仕様書」と構造設計において作成される「プログラム構造仕様書」である．(ソフトウェア仕様書は，開発対象システムの内部構造を定義するために作成される．プログラム構造仕様書は，ソフトウェア仕様書で定義した内部構造に基づき，具体的な実装方法を定義するために作成される．)

ソフトウェア仕様書及びプログラム構造仕様書の記載項目は，各仕様書の副々節(3階層目)の見出しを参考にして決定した(表2，表3)．

5.1 対象ツールの機能集合

対象ツールの機能集合は，文献[7], [8]の目次を参考にして，対象ツールが作成できる成果物を列挙することにより求めた(表1)．

表1：対象ツールの機能集合

機能番号	機能名
1	クラス図
2	クラス仕様
3	クラス属性仕様
4	操作仕様
5	引数仕様
6	集約／保有関係仕様
7	使用関係(依存関係)仕様
8	継承関係仕様
9	関連仕様
10	クラスカテゴリ仕様
11	状態図
12	状態仕様
13	遷移仕様
14	状態アクション仕様
15	オブジェクトメッセージ図
16	メッセージトレース図
17	オブジェクト仕様
18	クラスインスタンス仕様
19	リンク仕様
20	メッセージ仕様
21	モジュール図
22	モジュール仕様
23	サブシステム仕様
24	プロセス図
25	プロセッサ仕様
26	プロセス仕様
27	デバイス仕様
28	接続仕様
29	モデル管理ユニット
30	C++コード生成

5.2 ツール化率の計測

ソフトウェア仕様書の全記載項目と各記載項目の重み a の値，及び，対応関係にある対象ツールの機能項目の番号(無い場合は×)を表2に示す．表2において，各記載項目の重み a の値は，記述量と難易度により3段階評価して，大きい順に3，2，1とした．

成果物「ソフトウェア仕様書」の単位プロセスツール化率は，

16／41 ＝ 0.39

となる．

次に，ソフトウェア仕様書の記載項目「機能仕様」を「シナリオ」とし，Booch 法[10]の「メッセージトレース図」を作成するように，記載項目を再構成する．

すると，対象ツールの機能項目「メッセージトレー

ス図」との対応関係を追加できる.

再構成後のソフトウェア仕様書の単位プロセスツール化率は,

19／41 ＝ 0.46

となる.

このようにして，ソフトウェア仕様書の一部分に対するツール化を行うことができる.

表 2：ソフトウェア仕様書の記載項目

記載項目番号	記載項目名	重み a	ツール機能番号
1	目的	1	×
2	設計方針	1	×
3	設計指標	1	×
4	ハードウェアとのI／F	2	×
5	外部I／F	2	×
6	ソフトウェア構成要素の構成図	3	1
7	機能仕様	3	×
8	GUI 仕様	3	×
9	クラスカテゴリ概要	2	10
10	クラス概要	2	2
11	属性仕様	3	3
12	操作仕様	3	4
13	ウインドウ仕様	3	×
14	状態遷移図	3	11
15	データ仕様	2	×
16	性能目標	1	×
17	HMI 仕様	2	×
18	障害対策仕様	2	×
19	構造設計以降の設計方針	1	×
20	保留項目	1	×

プログラム構造仕様書の記載項目は，Booch 法にほぼ完全に従うように定義されている.

プログラム構造仕様書の全記載項目と各記載項目の重み a の値，及び，対応関係にある対象ツールの機能項目の番号(無い場合は×)を表 3 に示す．重み a の値の決定方法はソフトウェア仕様書の場合と同様である.

成果物「プログラム構造仕様書」の単位プロセスツール化率は,

31 ／ 31 ＝ 1

となる.

プログラム構造仕様書の場合は，部分的なツール化ではなく，導入初期から仕様書の全記載項目に対してツール化を実施した.

表 3：プログラム構造仕様書の記載項目

記載項目番号	記載項目名	重み a	ツール機能番号
1	クラスカテゴリ図	2	1
2	クラスカテゴリ仕様	2	10
3	クラス図	3	1
4	Class 仕様	3	2
5	属性保有関係	2	3
6	操作仕様	3	4
7	継承関係	1	8
8	使用関係	1	7
9	関連	1	9
10	状態図	3	11
11	相互作用図	3	16
12	オブジェクト図	3	15
13	プロセス図	2	24
14	モデル管理ユニット	2	29

5.3 ツール化実施率の計測

あるプロジェクトＡにおいて対象ツールを使用した際のソフトウェア仕様書の単位プロセスツール化実施率は，「状態遷移図」の作成に対象ツールが使用されなかったので,

16 ／ 19 ＝ 0.84

となる.

また，同じプロジェクトＡにおけるプログラム構造仕様書の単位プロセスツール化実施率は，「状態図」，「オブジェクト図」，「プロセス図」，「モデル管理ユニット」の作成に対象ツールが使用されなかったので,

21 ／ 31 ＝ 0.68

となる.

このように，プログラム構造仕様書の単位プロセスツール化実施率がソフトウェア仕様書のそれよりも低くなった.

5.4 計測結果の評価

「ソフトウェア仕様書」のようにツール化率が低い場合，従来の手順の部分が多いので，開発者には受け入れられ易い．結果的に，ツール化実施率が高くなる.

今後は，更にツール化率を高めたツール化を行えばよい．

「プログラム構造仕様書」では，ツール化率が高く，開発者の適応能力を越えた導入であったおそれがある．ツール化実施率が低い場合はその傾向の現れと見ることができる．プロジェクトAの場合は，プログラム構造仕様書作成においても，提案する方法によるツール化を行うべきである．

6 考察

提案した方法の前提条件と有効性について考察する．

6.1 提案する方法を適用する組織

提案する方法を適用可能な組織は，既にソフトウェアプロセスが定義されている必要がある．つまり，作業手順の詳細化が行われ，作成すべき成果物とその記載要領書が文書化されている必要がある．

6.2 メトリクスの理解容易性

ツール化率とツール化実施率はともに簡単な数式により与えられ，それが示す意味も理解しやすい．よって，開発部門の開発者から経営層に至るまで，メトリクスに基づいた議論が行える．

6.3 メトリクスの経済性

ツール化率は，開発部門で使用している各成果物の記載要領書やツール利用手順書から簡単に計測できる．ツール化実施率についても，成果物の確認，または，開発者へのインタビューにより計測できる．基本的には，これらの作業をSEPGが行うので，開発部門における開発作業を妨げることがない．従って，提案するメトリクスは開発部門に受け入れられやすい．

7 おわりに

本稿では，ソフトウェアプロセスのツール化状況を計測するためのメトリクスとその利用方法を提案した．本方法では，既存のソフトウェアプロセスとツールが前提とする利用手順との不整合に対して，成果物の記載項目とツールの機能の対応関係に基づきツール化手順を定義することにより，単位プロセスに対応付けられた成果物の記載項目単位でのツール化を可能にした．また，ソフトウェアプロセスのツール化による効果を計測することの難しさに対して，ツール化状況を計測するメトリクスとして，ツール化率とツール化実施率を定義することにより，ソフトウェアプロセスのツール化による効果を議論するために必要なデータの計測を可能にした．

今後の課題としては，提案したメトリクスとソフトウェアの品質と生産性との関係を明らかにすることが重要である．ソフトウェアプロセス全体で上記関係を議論すると，多数の外乱に悩まされることになる．そこで，まず，単位プロセスのツール化率について上記関係の分析を行う予定である．

参考文献

[1] Humphrey, W. S.: Managing the Software Process, Addison-Wesley (1989).

[2] 鯵坂 恒夫：プロセス支援環境，情報処理，Vol. 36, No. 5, pp. 431-437 (1995).

[3] 伊藤 昌夫：統合環境に於けるツールの在り方と結合のためのツールの分析及び設計法に関する考察，ソフトウェアシンポジウム '94 論文集，pp. 42-52 (1994).

[4] 池田 邦彦，西山 哲人，丹羽 徹，仲島 晶：成果物とツールの対応関係に着目したソフトウェアプロセスのツール化手法，ソフトウェアシンポジウム '97 論文集，pp. 186-194 (1997).

[5] Rumbaugh, J., et al.: Object-Oriented Modeling and Design, Prentice Hall (1991).

[6] Potter, B., et al.: An Introduction to Formal Specification and Z, Prentice Hall (1991).

[7] Rational Rose 3.0 ユーザーガイド: (株)オージス総研[販売元], RATIONAL Software Corporation[開発元] (1995).

[8] Rational Rose/C++ 3.0 によるラウンドトリップエンジニアリング: (株)オージス総研[販売元], RATIONAL Software Corporation[開発元] (1995).

[9] Jacobson, I., et al.: Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley (1992).

[10] Booch, G.: Object-Oriented Analysis and Design with Applications, 2nd edition, Addison-Wesley (1994).