2008－SE－161 (3)
2008／9／26

ソフトウェア開発プロジェクト，開発組織のアジリティ計測

居駒幹夫† 大島真幸† 谷田耕救† 大場みち子† 酒井三四郎††

†(株)日立製作所 ソフトウェア事業部 ††静岡大学 情報学部

ソフトウェア開発プロジェクト及びソフトウェア開発組織のアジリティを計測するメトリクスとその適用方法を提案する．ソフトウェア開発プロセスを validation の観点でモデル化し，中間成果物が生成されてから validate されるまでの期間を最小化することをアジリティと定義し，メトリクス化した．このメトリクスは，特定のアジャイル開発手法に依存せず，あらゆるプロセスモデルに独立である．また多様なプロセスモデルに従ったソフトウェア開発プロジェクトを多数持つようなソフトウェア開発組織でも適用可能である．このメトリクスの実用性は大規模ソフトウェア開発組織に約 10 年，7,000 以上のソフトウェア開発プロジェクトで計測した事例により検証されている．

Proposal of an Agility Metric for Software Development Projects and Organizations

Mikio Ikoma† Masayuki Ooshima† Takahiro Tanida† Michiko Oba†
Sanshiro Sakai††

†Software Division, Hitachi, Ltd. ††Dept. of Computer Science, Shizuoka University

This paper provides a metric for evaluating agility of software development projects and organizations. The model is based on a validation model of software development process and defines agility as minimizing life of intermediate materials to be validated. Its major distinguish feature is that it is independent of specific software development process models including waterfall model, agile software development methods and etc. Therefore, the metric can be used for software development organizations that have many software development projects with various development processes. The metric is verified to have a practical use in large software development organization through the application of more than 7,000 projects for about 10 years.

1. はじめに

本論文は，ソフトウェア開発プロジェクト及びソフトウェア開発組織のアジリティを計測する実用的なメトリクスを提案する．

2 章ではソフトウェア開発における効率の考え方，特にアジリティがソフトウェア開発プロジェクト及びソフトウェア開発組織で重要であることを示す．3 章では，すでに提案されているアプローチを紹介し，その課題を述べる．4 章では，開発プロジェクトごとに異なる要件，多様なソフトウェア開発プロセスといった昨今のソフトウェア開発組織特有の環境に適合するソフトウェア開発のモデルを示す．続いて，このモデルにしたがってどのように，ソフトウェア開発プロジェクト，ソフトウェア開発組織がその目標に合ったアジリティを計測できるかを示す．5 章では，このモデル，メトリクスを使用した大規模ソフトウェア開発組織のアジリティを改善した事例を紹介しその結果を報告，評価する．

2. ソフトウェア開発におけるアジリティの重要性と課題

本章では，背景とソフトウェア開発においてもアジリティの計測が必要な理由を明確にする．

2.1. 背景

同じ機能，同じ品質の商業ソフトウェアを開発するとき，ソフトウェア産業という観点では，より低コスト，より短期間に完成するソフトウェア開発組織のほうが優位である．したがって，ソフトウェアの開発組織においても，その生産性と開発期間を定量的に計測

し，改善していくことは他の工業製品事業者と同様重要である．

1980 年代，日本の主要なソフトウェアベンダは画一的な開発プロセスを採用したソフトウェア工場 1)のアプローチで，ソフトウェアの生産性を定量化した．しかし，ソフトウェア開発に関する競争が激しく，またプロジェクトに求められる環境要件が多様な昨今では，次の 2 点がソフトウェア開発組織の課題となっている．

- ソフトウェアの開発の生産性向上だけでなく，開発期間を短縮，変化する要件に対するすばやい対応といったアジリティの確立
- 特定の開発プロセスモデルに依存しない，多様な開発手法に適合した，アジリティの計測及びその推進

2.2. アジリティ計測の必要性

ソフトウェアに限らず，すべての産業のすべての製品で製品開発，製品製造での生産性の向上と期間の短縮は共に重要である．

生産性は，「生産量／コスト」で計測される．ソフトウェア開発においても，同じ機能要件，同じ機能外要件のソフトウェアを開発するときには，少ないコストで開発したほうが高効率といえる．したがって，生産性はソフトウェア開発においても重要な指標である．

しかし，生産性の指標だけではソフトウェア開発の開発期間の短縮や，複雑な要件への迅速な対応といった，ソフトウェア開発のアジリティを計測することはできない．例えば，図 1 のガントチャートの例での場合，プロジェクト C のほうがアジリティの観点でプロジェクトA，Bより優位であるが，生産性すなわち成果量／コストで計測すると，各プロジェクトの生産性は同一になってしまう．

Project A = Project B = Project C

次にコストの代わりに開発期間を用い，成果量／開発期間でサブプロジェクト単位に測定すると，図 1 の効率は

Project A = Project C > Project B

という関係になる．しかし，ソフトウェア開発プロジェクトの経営的な視点ではプロジェクトCはプロジェクトAよりも優位で，プロジェクトAとプロジェクトBのパターンはプロジェクトの単位が異なるだけで，ビジネスの実態はほとんど同様である．したがって，成果量／開発期間というメトリクスは使用できず，図 1 の例で

Project C > Project A = Project B

のようになる指標がソフトウェア開発のアジリティを計測する場合に必要である．

3. 既存のアプローチとその課題

本章では，ソフトウェア開発における既存のアジリティを計測，改善していく手法を紹介し，その課題を示す．

3.1. アジャイル開発手法向けメトリクスの課題

図 1 のプロジェクトCのようなソフトウェア開発プロセスは反復型プロセスモデルとしてよく知られたプロセスである．

XP，Scrum といった主要なアジャイル開発手法は反復型の開発プロセスモデルを採用しており，これらのアジャイル開発手法で適用可能なメトリクスを使用してソフトウェア開発組織のアジリティを間接的に測定する方法がある．

例えば，Qumer ら 3)は 6 種類のアジャイル手法がどの程度アジャイルマニフェスト 4)に準拠しているかという測定を行っている．また，Williams ら 5)は，XP の各プラクティスへの適合度を評価するフレームワーク XP-EF を提唱している．これらの手法と同様の評価をあるソフトウェア開発プロジェクトのソフトウェア開発プロセスに当てはめることによりアジャイルマニュフェストや XP への適合度を測定することが可能で

プロジェクト	見積り KLOC	開発人員	200x	200y											200z				
			Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May
プロジェクトA	150	12																		
プロジェクトB	150	12																		
サブプロジェクトB-1	50	4																		
サブプロジェクトB-2	50	4																		
サブプロジェクトB-3	50	4																		
プロジェクトC	150	12																		
サブプロジェクトC-1	50	12																		
サブプロジェクトC-2	50	12																		
サブプロジェクトC-3	50	12																		

図1 同じ規模のソフトウェア開発プロジェクトのガントチャート例

あり，間接的にソフトウェア開発プロジェクトのアジリティを測定できる．

このアプローチは，単一のプロジェクト又は，すべてのプロジェクトが特定のアジャイル開発手法（例えば XP）を採用しているような組織のアジリティを計測するのには使用できる．しかし，大きなソフトウェア開発組織という観点では，Boehm7) 8)が述べているように，多くのプロジェクトは純粋なアジャイル開発手法，純粋なウォーターフォールモデルということはない．したがって，アジャイル開発手法のみを採用するというのは現実的でないし，固定的な要件に基づくようなソフトウェア開発では，依然ウォーターフォールモデルに近い重量プロセスのソフトウェア開発プロセスを適用する機会も多い．

したがって，アジャイル開発の特定手法の各プラクティスへの適合度を計測するメトリクスを組織全体のアジリティ計測するために使用することは困難であり，多様なプロセスモデルに独立なアジリティを表すメトリクスが必要と考える．

3.2. 財務／ハードウェア生産手法適用時の課題

前節で述べた特定のアジャイル開発手法に特化する方法の課題は，財務指標における回転率又はハードウェア量産品の生産手法では広く適用されているサイクルタイムのメトリクスをソフトウェア開発組織にも適用することが有効な解になりうる．

すなわち，開発中のソフトウェア項目を財務指標で言えば資産，ハードウェア生産手法でいう在庫と位置付け，その開発中の中間成果物の量と完成製品の量からソフトウェア開発の回転率（又はその逆数のサイクルタイム）を求めることができる．

図 1 の例の場合，200y 年のプロジェクト A，プロジェクト B は平均の仕掛かりは 150KLOC で，アウトプットも 150KLOC なので，回転率は，1 回／年となる．プロジェクト C の仕掛かりは 50KLOC で，アウトプットは 150KLOC なので，回転率は，3 回／年となる．したがって，アジリティの要件であった

Project C ＞ Project A ＝ Project B

を満足する．

上記の例は同じ開発量の場合の比較であるが，開発量が違う複数のプロジェクトで比較することができる．すなわち，ある期間での成果物の総和を V，その期間内のある時点で中間成果物の平均を U としたとき，そのソフトウェア開発のアジリティは次の式で表すことができる．

ソフトウェア開発プロジェクトや開発組織のアジリティ ＝ V ／ U

この値の単位は，ある期間での回転数であり，ある期間でどの程度そのプロジェクトが回転しているか（すなわち，サイクルタイムの逆数）を表す．

(株)日立製作所ソフトウェア事業部では回転率による組織レベルのアジリティの計測を 1999 年から継続している(この結果は 5 章参照)．同様にハードウェア生産手法で広く用いられているサイクルタイムをソフトウェア開発にも適用するアイデアは，Poppendieck10)（リーン生産手法をソフトウェアに適用），Anderson11)（制約条件の理論 TOC をソフトウェアに適用）も提案している．

回転率により，ソフトウェア開発組織のアジリティは測定可能であるが，現実にこの手法をソフトウェア開発組織で使用するためには次の課題がある．

- どのようなモデルに従ってソフトウェア開発の回転を測っているのかが不明確．ソフトウェア開発における在庫とは何なのか？何を以って成果物としているか？等
- ソフトウェア開発プロジェクトや組織の目標に合ったメトリクスなのかどうかが不明確．単に計測できるから計測するでは意味が無い．

上記のような課題が解決しないと，ソフトウェア開発組織がそのビジネスの目標としてソフトウェア開発のアジリティを評価改善することはできない．

4. Validation モデルによるアジリティメトリクス

本章では，3 章で示した課題を解決する，Validation モデルおよびメトリクスを 4.1 節で紹介し，ソフトウェア開発組織でどのように適用するかを 4.2 節で説明する．適用時の考慮事項を 4.3 節に述べる．

4.1. ソフトウェアの Validation 確保モデル

本節では，ソフトウェアにおける品質確保を考えるときに重要な概念である V&V すなわち verification と validation を使うことにより，ソフトウェア開発のアジリティをモデ

ル化できることを示す.

IEEE 用語集 15)によれば, verification は

"*The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.*"

であり, validation は

"*The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.*"

と定義されている.

シンプルな定義は Boehm 6)によって与えられている. すなわち, verification は,「正しく製品を作っているか」であり, validation は「(結果として) 正しい製品を作っているか」を表す.

本論文では, このうち validation に焦点を当てる. なぜなら, 最終的なソフトウェア品質は validation でのみ確認できるからである. この考え方に基づき, 最終的に validate される任意のソフトウェア項目の状態遷移モデル, Validation モデルを提案する(図 2).

このモデルでは, すべての最終的に validate される項目は, 計画時に「識別された計画項目の状態」になり, 生成時に「validate されていない仕掛かりの状態」になり, validation により,「validate された成果物の状態」と状態遷移する. ただ, ソフトウェア開発の場合は, validation の対象となる要求に変更が発生する場合も多い. したがって, 要求変更時には, そのソフトウェア項目が破棄される, 又は仕掛かり状態に戻るという状態遷移も考慮している.

このモデルの特徴は「(作り始めたが) まだ validate されていない中間成果物」, すなわち仕掛かり (又はハードウェアにおける在庫) という状態を定義したことである.

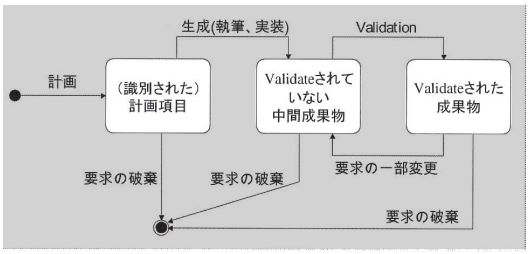

図 2　ソフトウェア項目ごとの状態チャート図

Figure 2. State Chart Diagram of a Software Item

本論文では, ソフトウェア開発におけるアジリティを, Validation モデルにおける「validate されていない仕掛かりの状態の期間を最小限にすること」と定義する.

このモデルで, 3.2 で示したアジリティを示す式を解釈する. 中間成果物を「(作り始めたが) まだ validate されていない中間成果物」状態にあるソフトウェア項目の総量を U´とし, 成果物を「validate された成果物の状態」にあるソフトウェア項目の総量を V´とすることにより, ソフトウェア開発においてのアジリティを示す回転率を

ソフトウェア開発プロジェクトや開発組織のアジリティ　＝　V´／U´

とすることができる.

本モデルに基づくメトリクスを直感的に理解できるように, 簡単な例を示す. 同じソースコード規模のソフトウェアをウォーターフォールモデル, アジャイル開発手法で開発した場合を仮定する. このときの典型的な, verification, validation の適用パターンは図 3 のようになる.

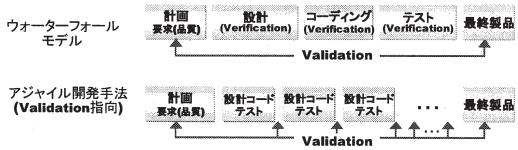

図 3　V&V の適用パターン

Figure 3. Typical Application Pattern of V&V

ウォーターフォールモデルの場合, その中間工程でのソースコードは, verification が唯一の品質確保方法で, そのソフトウェア開発プロジェクトの必要条件(requirements)を満足しているかどうかの validation は不可能である. 一方, アジャイル開発手法は, そのマニフェスト 4)が示すように, 中間工程を軽視しており, ソースコードも動作する形式で, もともとの要求に合っているかどうかを繰り返し validate できることをその価値としている.

次に, ウォーターフォールモデル, アジャイル開発手法でどのようにソフトウェア項目が図 2 に示した状態遷移するかを図 4, 図 5 に示す.

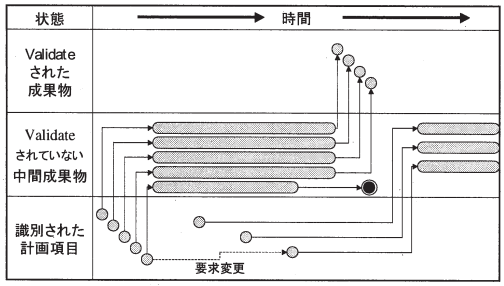

図4　ウォーターフォールモデルでのソフトウェア項目の状態遷移例

Figure 4. Typical State Transition of Software Items Using Waterfall Model

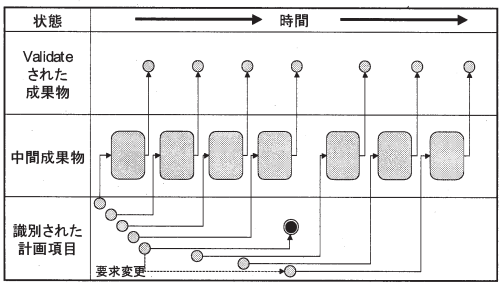

図5　アジャイル開発手法でのソフトウェア項目の状態遷移例

Figure 5. Typical State Transition of Software Items Using Agile Development Methods

これらの図から明らかなようにウォーターフォールモデルでは，各ソフトウェア項目の中間成果物状態の期間が長く，アジャイル開発手法の場合では短くなることが分かる（ここでの比較の対象は期間の長短であり，コストはアジャイル開発手法のほうが大きい可能性もあることに注意）．

開発期間中の validate されていない中間成果物，例えばソースコードの量の変化をグラフ化すると図6のようになる．

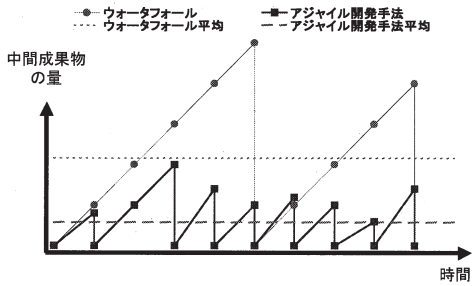

図6　プロセスモデルによる中間成果物量の推移

Figure 6. Transition of Inventory of process models

ウォーターフォールモデルの場合，validate されていないソースコードなどが開発の最後まで蓄積され，工程の最後に validate されて，中間成果物は一掃されて最終成果物となる．したがって，開発プロセス全体での中間成果物の平均量は多くなる．アジャイル開発手法の場合は，適時 validation があるため，中間成果物の平均量は少ない．したがって，同じ成果量のソフトウェア開発においては，その期間中の中間成果物の平均量が少ないプロジェクトのアジリティのほうが高いといえる．

本モデルで強調したいのは次の点である．

- validation に焦点を当てたことにより，プロジェクトの全てのステークホルダにとって意味のあるメトリクスが導かれる．
- このモデルは特定のプロセスモデル，開発手法にも依存していない．
- 例に示したソースコードの規模だけでなく，最終的に validate されるすべてのソフトウェア項目で適用が可能である
- このモデルから得られたメトリクスは，さまざまな開発プロセスモデルを使ったプロジェクトの結果であっても比較や結合が可能である
- 多数のソフトウェア開発プロジェクトを並行に走らせているような大組織でも組織のアジリティが容易に計測可能である

4.2. Validation モデルの適用プロセス

前節で示したメトリクスを複数のソフトウェア開発プロジェクトを持つ組織にどのように適用するかを Basili の GQM (Goal Question Metric) パラダイム 14)を用いて説明する．

Goal：

ソフトウェア開発組織でアジリティに関連したビジネスの目標を明確にする．適用する組織の特徴により組織レベルのアジリティの目標は変わる．例えば，次のような目標が考えられる．

- 単に開発プロジェクトの回転を早くする（4.1 節でのソフトウェア項目を組織のもつ開発プロジェクトそのものとする）
- 開発する予定の機能要件，機能外要件を早く決定する
- コーディングから出荷までの開発期間を短くする
- 障害が起きてから，対応するフォールトをみつけ，修正し，確認するまでの期間を短くする

Question :

現状の組織の実態を評価したうえで，ビジネスのアジリティ目標に直接関係するソフトウェア項目を選定する．選定するソフトウェア項目は組織内のソフトウェア開発プロジェクトで共通的にデータ採取できるものである必要がある．次に，それらの項目が，4.1 節で示したモデルに従って，どのように目標が達成できるか質問を生成する．

Metric:

4.1 節で示した回転率で，選定されたソフトウェア項目のアジリティを少数プロジェクトで予備的に測定する．得られた結果を解析し，ビジネスのアジリティ目標が達成できるかどうかを判断し，目標との整合性が検証されれば組織全体でメトリクス取得を展開する．

4.3. Validation モデル適用時の考慮事項

（1）プロジェクトレベルでの適用

組織共通のアジリティ目標に向けたメトリクスを使うだけでなく，プロジェクト固有の目標に応じたメトリクスを設定することも可能である．

（2）ソフトウェア項目選定時の考慮事項

ソフトウェア項目を選ぶ場合，重要なのは，本モデルで対象となるソフトウェア項目は，「validate された成果物」という状態を取りうる，すなわち validation の対象となるものでなければならない．validation の対象とは，対応する要求のステークホルダが，その項目を正しいか否か validate できることである．したがって，例えば，動作可能なソフトウェアのテストであっても，テスト対象の実装が，外部機能を持たないソフトウェア内部に組み込まれたライブラリのような場合は本モデルの対象外となる．一方，定義された要件のように動作可能なソフトウェアはなくても，それが書かれたドキュメントは対応する要望をだした顧客が validate できる場合がある． したがって，このような項目は本モデルの対象になる．

また，選択されたソフトウェア項目は構成管理されている必要がある．構成管理されていない場合，その項目の識別，状態の変更を構成管理できるようにする必要がある．

5. 適用実例の紹介

本章では，前章で説明した Validation モデルにしたがったアジリティ改善を大規模ソフトウェア開発組織に適用した事例を紹介する．

5.1. 概要

(株)日立製作所ソフトウェア事業部は，約 3000 人のソフトウェア技術者を擁し，パッケージ型のソフトウェアを開発する組織である．1969 年に設立され，1970 年代からソフトウェア工場アプローチにより，ソフトウェアの生産性，ソフトウェアの信頼性の両面で定量的な目標を決めて管理推進している．

1990 年代までは，期間や工程を意識しない，「成果量／コスト」，「フォールト数／成果量」といったメトリクスを活用していた．しかし，1990 年代以降，市場での競争激化に従い，従来の「効率向上」「信頼性向上」に加えて「開発スピードアップ」も組織目標としている．この目標に対応して，1990 年後半以降，生産管理，品質管理の両面で開発期間や工程を意識した定量的な目標を立てて推進している(表 1)．

本論文では，このうち，生産管理へ 4 章で説明したアジリティのメトリクスを適用した事例を紹介する．

表 1　生産管理，品質管理の重点項目
Table1　Priority Targets of Production Management and Reliability Management

	生産管理	品質管理
1970 年代後半～	生産性，納期遵守，etc.	最終工程でのフォールト摘出数，フィールドでの障害数，etc.
1990 年代後半～	(上記に加え) アジリティ，多様な開発形態に対する生産性の正規化	(上記に加え) 各フォールトの摘出すべき上流工程の特定による定量的評価

5.2. Validation モデルの適用結果

ソフトウェア事業部では，事業部自身及び全ソフトウェア開発プロジェクトを対象にアジリティを計測中である．1999 年 10 月より，現在，7000 以上のプロジェクトでのデータが蓄積されている．使っているメトリクスは，4 章で述べたモデルに従っており，仕掛かりとして，開発中の見積りソース行数，最終成果量として完了時のソース行数を用いている．

本章では，アジリティの改善を目標とした施策を推進したグループと，アジリティ施策を推進しなかった他のグループを比較した結果を報告する．

表 2 にグループの概要と特徴をまとめた．計測期間は 4 年間．対象プロジェクト数は両グループともに 1000 プロジェクト以上である．グループＢの組織目標の一つにアジリティの改善がある．このグループに属する製品の多くの開発プロジェクトは，反復型開発の適用，プロトタイピング，コンカレントＱＡ（品質保証部署が行う validation を開発工程の最終段階だけでなく，開発の中途段階で実施する施策）といったアジリティ向上施策を採用している．

表２　計測したグループとその特徴
Table2　Outline of Groups

	グループＡ	グループＢ
計測区間でのプロジェクト数	1,649	1,185
メトリクス(共通)	ソフトウェア項目：ソース行数 メトリクス：6 ヶ月の回転数	
計測期間	4 年間（同時期）	
主な開発プロセスモデル	ウォータフォール	ウォータフォール，反復型開発
対象市場	成熟市場	成長市場
生産技術関連の主な施策	テスト自動化 母体解析 形式的検証試行	プロトタイピング コンカレントＱＡ 部品活用

5.3.　適用のまとめ

アジリティの 6 ヶ月ごとの比較結果を図 7 に示す．同時期の「成果量／コスト」の生産性の比較結果を図 8 に示す．

図 7 の結果により，アジリティ向上施策を実施した製品グループに対しては 4 年間で約 2 倍のアジリティを達成したが，アジリティ向上施策を実施しなかった製品グループはほぼ変化が見られなかった．

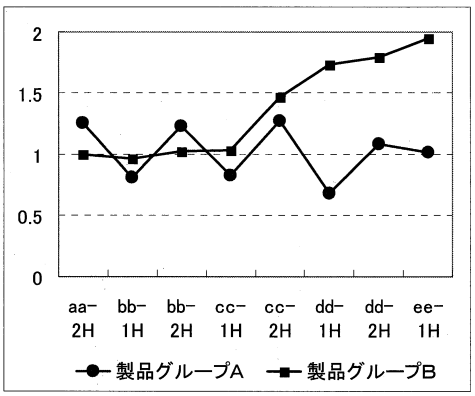

図 7　アジリティの推移比較
Figure 7. Transition of Agilities of Two Groups

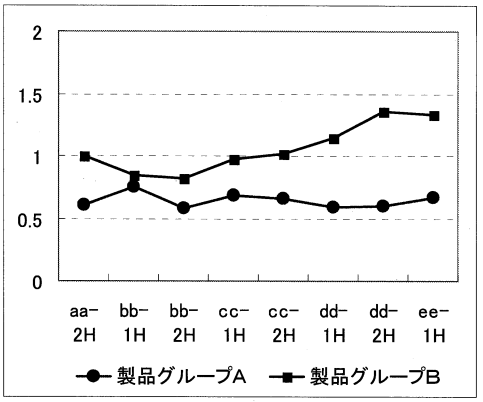

図 8　生産性の推移比較
Figure 8. Transition of Productivities of Two Groups

図 8 の生産性の結果と比較すると，アジリティの推移と生産性の推移は，必ずしも同じ傾向を示さないことが分かる．生産性が大きく低下した bb 年でも，アジリティはほぼ横ばいであった．また，4 年の計測期間で生産性は３０％の向上に対して，アジリティはほぼ倍増した．本適用事例においては，組織としてのアジリティ向上施策が指標として的確に表現できたと考える．

6.　結論及び今後の課題

本論文で示したソフトウェア開発におけるアジリティメトリクスの有効性及び課題をまとめる．

メトリクスの有効性：

- ソフトウェア開発におけるアジリティを，中間成果物が生成されてから validate されるまでの期間を最小化することと定義し，財務の回転率やハードウェア生産手法でのサイクルタイムを，ソフトウェア開発の特性に合致するようにした．
- 提案したメトリクスは，プロセスモデルに独立で，純粋なアジャイル開発手法や既存のソフトウェア開発プロセスモデルに従っていないようなプロジェクトでも計測可能であることを示した．
- 提案したメトリクスは，従来からの「成果量／コスト」といった生産性と，併用可能であり，プロジェクトのアジリティを計測可能である．
- 提案したメトリクスは，多様なプロセスモデルを適用し，多くのプロジェクトを並行して実行している大規模ソフトウェア組織にも適用可能である．

メトリクスの課題：

- 提案したメトリクスを使用するためには，開発プロセスの成熟度が必要である．具体的に，要件管理，品質管理，成果物管理を支えるような構成管理環境により，validationを定量的に行う必要がある．
- 仕掛かり量や成果量の測定が難しいタイプのソフトウェア開発プロジェクト（例えば，他社への外注，部品ベースでの開発等）においては，仕掛かり量，成果量を正規化する必要がある．

今後は，提案したモデル，メトリクスを継続して適用，評価することにより，改善していく．また，本提案で示したモデルを使い，ソフトウェア開発プロセスを最適化する手法を検討する予定である．

参考文献

1) Cusumano, M. A.: Japan's Software Factories: A Challenge to U.S. Management: Oxford Univ Pr on Demand(1991)
クスマノ，マイケル：日本のソフトウェア戦略—アメリカ式経営への挑戦：三田出版会（1993）
2) Matsumoto, Y.: Japanese Perspectives in Software Engineering, pp303 Addison-Wesley(1989)
3) Qumer, A. et al: An evaluation of the degree of agility in six agile methods and its applicability for method engineering, Information and Software Technology 50 (2008) pp280-295
4) Agile Manifesto. Manifesto for Agile Software Development, http://www.agilemanifesto.org/
5) Williams, L. et al: Extreme Programming Evaluation Framework for Object-Oriented Languages Version 1.4, ftp://ftp.ncsu.edu/pub/unity/lockers/ftp/csc_anon/tech/2004/TR-2004-18.pdf , 2004
6) Boehm, B. W.: Verifying and Validating Software Requirements and Design Specifications, IEEE Software, Volume 1 , Issue 1 (January 1984)
7) Boehm, B. W.: Get Ready for Agile Methods, with Care, IEEE Computer, January 2002 (Vol. 35, No. 1) pp. 64-69
8) Boehm, B. W. et al: Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley(2004)
9) 大野 耐一：トヨタ生産方式—脱規模の経営をめざして，ダイヤモンド社（1978）
Ohno, T.: Toyota Production System: Beyond Large-Scale Production, Productivity Pr (1988)
10) Poppendieck, M. et al: Lean Software Development: An Agile Toolkit, Addison-Wesley(2003)
ポッペンディーク，メアリー他：リーンソフトウエア開発~アジャイル開発を実践する22の方法~,日経BP社(2004)
11) Anderson, D. J.: Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results, Prentice Hall PTR (2003)
アンダーソン，アジャイルソフトウェアマネジメント，日刊工業新聞社（2006）
12) Marciniak, J. J. et al: Encyclopedia of Software Engineering, John Wiley & Sons Inc (1994)
片山 卓也 et al 監訳：ソフトウェア工学大事典，朝倉書店（1998）
13) Dunn, R. N.: KPIs for Agile Teams, http://www.agilejournal.com/content/view/786/111
14) Basili, V. et al: Using Measurement to Build Core Competencies in Software, http://www.cs.umd.edu/~basili/publications/technical/T87.pdf , (2005).
15) IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990