第６分科会 品質研究ゼミ

品質研究ゼミ

Research Project for Quality Software Development

主査	：	奈良	隆正（ＮＡＲＡコンサルティング）
副主査	：	中井	一人（Bowline Human & Technology）
アドバイザ	：	堀田	文明（北陸先端科学技術大学院大学）
書記	：	河野	哲也（電気通信大学大学院）
メンバ	：	岡田	文和（アドバンソフト開発（株））
		豊田	拓也（(株)アドバンテスト）
		小渕	一幸（セイコーエプソン(株)）

研究概要

ソフトウェア開発の状況をより良くしようと、多くの組織で、品質向上、生産性向上などを目的として、改善活動が行われている。このような活動を通して、個人、あるいは、ある組織の中でいろいろなアイデアを考え、実践している方々も多い。しかしながら、このような取り組みでは、いろいろな障壁にぶつかることもよくある。その際に、外部の方からアドバイスをもらったり、同じ悩みを持った人々で議論することは、解決のためのヒントを得たり、モチベーションを上げたりするうえでとても有効である。

前述のような取り組みを実践するために、本分科会を昨年度立ち上げた。2 年目となる本年度は、昨年から継続した「要求の評価方法の研究」、新たな視点での取り組みである「形式仕様記述言語による派生開発でのモデル検証」、「プロジェクト管理における統合モデルの提案」に関する論文という 3 つの成果が得られた。また、メンバ間で問題や課題を一緒に議論する中から自身のテーマに対する解決のヒントばかりでなく、これまでにない新たな気付きを得られるといった効果があった。

Abstract

In order to improve the software development, improvement activities, aiming for the quality improvement, the productivity enhancement, and etc., are conducted in many organizations. Through these activities, individuals or members in organization think up many ideas and participate directly in the implementation. However, there emerged many obstacles during these processes. In such cases, it is better to obtain advices from outside experts and discuss with others that have similar problems. This is very effective to acquire hints for possible solutions and raise the motivation.

To practice the approach discussed above, this committee was set up last year. Three research results are obtained this year. One is the paper “A Study of the Evaluation Method of Requirement”, which is a continued research from last year. The other two are the paper “Model Verification in the Derived Development by the Formal Specification language” and the paper “The Approach of Integrated Model in Project Management”, both of which take the new perspective. Moreover, through discussing problems and issues with others, not only the hints, which help resolve problems, but also the good results, which many new findings are yielded, are obtained.

１　目的

近年、ソフトウェア技術者が持つべき基礎知識として、ソフトウェアエンジニアリングやマネジメントの技術が注目を集めている。ソフトウェアが今後の産業を支える柱のひとつであることを考えると、ソフトウェアエンジニアリングやマネジメントの技術の実践の場で適用・評価・考察するという作業が必要である。

本分科会では、このような背景を踏まえ、個々のメンバが抱えている問題を分析し、解決のためのアプローチを検討し、適用･評価･考察するというサイクルをタイミングよく回すことを目的とした。最終的な成果物としては、1 年間をとおして実践した結果を整理し、報告書、論文という形式にまとめることを目標のひとつとして活動した。

２　活動経過

本分科会は、合計 8 回の分科会を開催した。最初の 2 回の分科会（4～6 月）では、各メンバの業務の内容と、課題の洗い出し・絞り込みに時間を割いた。合宿形式で行った 7 月の分科会で抽出できた課題を解決する方法を議論し、各自の研究テーマを決めた。合宿までの 3 回の分科会をとおして、各メンバとも、自分たちの問題をある程度客観的に捉えることができ､本当に解決すべき問題･課題が明確になってきた。

合宿終了後から 12 月までの分科会では、各メンバがテーマに沿った検討を行い、分科会で発表し、他のメンバと討議し、内容を具体的にすることを行った。具体的になった項目は、自分の業務に持ち帰って実践し、その結果を次の分科会にフィードバックするというサイクルを回した。この活動をとおして、各メンバとも、「各自で仮説を確立」→「全員で分科会で検討」→「各自の業務での仮説の検証」という良いサイクルで一年間を過ごすことができた。1 月の分科会では、4 月から実践してきた内容を各自整理し、論文という形式でまとめた。

また、毎回の分科会の中で各自の発表の参考になる事例や論文、技術的なヒントなどを適時、主査、副主査、アドバイザ、書記、メンバから提供した。

３　研究成果

今年度も 3 名のメンバが参加して、分科会メンバ全員でいろいろな議論を交わしながら、各自のテーマを深掘りした。その結果、以下の 2 つの提案論文と 1 つの事例論文が成果としてまとまった。

(1) テーマ：要求の評価方法の研究
報告者：岡田 文和（アドバンソフト開発（株））
内容：
要求分析は、そのアウトプットにより、開発されるシステムの方向性と顧客満足度は大きく左右されてしまう。昨年度の研究で、要求分析の方法を提案し、要求の導出はできるようになったので、今年度の研究では、要求の具体的な検証方法を提案する事とした。

(2) テーマ：形式仕様記述言語による派生開発でのモデル検証
報告者：豊田　拓也（(株)アドバンテスト）
内容：
本研究では、組み込みソフトウェアの派生開発において、スペックアウトした仕様を形式仕様記述言語を用いて整理した。成果として、プラットホーム独立モデルと、論理数学による検証性が得られた。

(3) テーマ：プロジェクト管理における統合モデルの提案

報告者：小渕 一幸（セイコーエプソン（株））

内容：

プロジェクト評価およびフィードバックを行うにあたっては、プロジェクトの特性やPMの能力に依存することが多い。そこで、様々なプロジェクトに対して統一的な概念で評価およびフィードバックを行うためのプロジェクト管理フレームワークを提案する。

４　分科会活動の総括

3K（きつい、きつい、きつい）と言われるソフトウェア開発の状況をより良くしようと、多くの組織で、品質向上、生産性向上などを目的として、改善活動が行われている。

このような状況を打破しようと、個人、あるいは、ある組織の中でいろいろなアイデアを考え、実践している方々も多い。しかしながら、このような取り組みでは、いろいろな障壁にぶつかることもよくある。その際に、外部の方からアドバイスをもらったり、同じ悩みを持った人々で議論することは、解決のためのヒントを得たり、モチベーションを上げたりするうえでとても有効である。

昨年度、前述のような取り組みを実践するために、本分科会を立ち上げた。結果として、昨年から継続した取り組みである論文として１つ、新たな視点での取り組みである論文として２つをまとめることができた。各メンバの業務に直接フィードバックできることは当然として、それ以外にも以下のような効果があった。

- 他のメンバの問題意識やアイデアを聞き、議論を重ねることで、自分自身の活動のヒントを得る
- 他のメンバの活動状況を見て、自分も頑張らねばという気持ちになる（モチベーション向上）
- 会社は違ってもソフトウェア開発において同様な悩みや問題を抱えていることが把握できる

結論として、本年度、本分科会を継続した目的は達成できたと考えている。来年度も本分科会を継続し、より良い進め方を検討するとともに、良い成果が出ることを期待している。

要求の評価方法の研究

岡田 文和（アドバンソフト開発株式会社）

１　概要

要求分析は、システムの開発工程の最初に行う重要な工程であり、そのアウトプットにより、開発されるシステムの方向性と顧客満足度は大きく左右されてしまう。一方で、この工程の精度が低かったとしても、システムを開発する事は可能である。つまり、要求分析のやり方が大きな問題を引き起こす危険性を秘めていると言える。実際に、筆者の近傍で、要求分析が原因となって発生した問題が散見されている。

昨年度の研究で、要求分析の方法を提案した。これにより、要求の導出はできるようになったが、導出された要求の検証方法については、単に「レビューによって確認する」と提案するに留まった。

今年度の研究では、要求の具体的な検証方法を提案する事とした。

２　昨年度の研究成果

昨年度の研究では、システム要求が発生する源となるアプリケーション(適用)・ドメインでの要求を明らかにする事に主眼を置いた。これは、顧客満足度を高める為には、システムがアプリケーション・ドメインでどのように貢献するかが重要であり、その為には、システム要求を明らかにする前のステップとして、アプリケーション・ドメインでの要求を明らかにする事が必要と考えたからである。

その方法論として、次の事を提案した。

・「4 アイテム要求モデル」を使用して要求を表現する
・要求をドメイン毎に階層毎に分離する

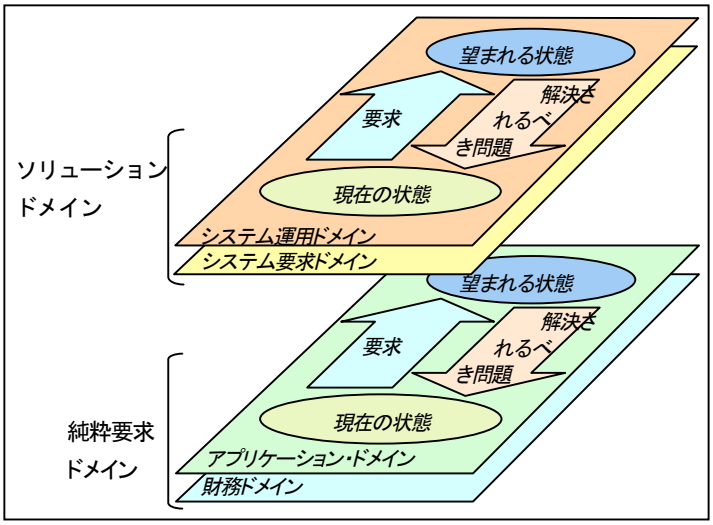


図１　4 アイテム要求モデルとドメインの階層化の例

「4 アイテム要求モデル」では、次の 4 アイテムを明らかにする事によって、要求とその関連情報を表現した。

・現在の状態
・解決されるべき問題
・望まれる状態
・要求

「要求をドメイン毎に階層化」では、4 アイテムの情報が、どのドメインに属するのかを分類するようにした。 特に、システムというソリューションを提供する立場から、システム要求以降のドメインを「ソリューションドメイン」、ソリューションとしての情報を取り除いた、要求の源泉となるドメインを「純粋要求ドメイン」として分離した。

これらの方法論のねらいは、次の事であった。

・解決されるべきアプリケーション領域の問題を明確にする
・要求にとって、ノイズであるソリューションを排除する
・ドメイン間の要求の関連を整理できるようににする。

３　本年度の研究目標

要求の導出方法については、昨年度の研究を引き出すまでもなく、様々な方法論が提案されている。どのようなプロセスにより導出された要求であれ、検証され、不足が見つかれば補完され、プロセスにフィードバックしていく事が必要である。では、導出された要求はどのように検証したら良いのだろうか?多くの文献では、要求はレビューで検証する事を提案しているが、具体的には、どのような方法をとれば良いのだろうか?

このような問題意識から、本年度の研究では、「要求の具体的な検証方法を提案する事」を目標とした。

４　研究成果

4.1.　IEEE の要求特性

IEEE830 では、要求特性として、次の項目を定めている。

・妥当性
・非曖昧性
・完全性
・無矛盾性
・重要度と安定性のランク付け
・検証可能性
・変更可能性
・追跡可能性

この研究では、これらの要求特性に沿って、要求を評価する事とした。
以下に、それぞれの要求特性の説明と検証方法の概要を示す。

4.1.1. 妥当性

意味: 開発されるソフトウェアが満たすべきものである事

検証方法: 下記を評価、チェックする。

その要求が目的・目標の達成にどの程度、貢献するか?

貢献しない要求が排除されているか?

具体的には、要求の貢献度評価(詳細後述)を実施する。

4.1.2. 非曖昧性

意味: 要求が一意に解釈できる事。　違った意味に解釈されない事。

検証方法: 言語検証を実施する。

補足: 要求が曖昧でない事を検証する以前に、形式言語等を利用して、要求記述から曖昧性を排除する等の方法が望まれる。

4.1.3. 完全性

意味: 漏れがない事。　冗長でない事

検証方法: 最上位の要求(目的)については、下記をチェックする。

解決されるべき問題が明らかである事

解決されるべき問題に対して、要求が的確に応えている事

適用範囲が明確である事

目標値が明確である事

上記に漏れがない事

下位の要求については、上位要求の目標値に対する貢献度を評価する

4.1.4. 無矛盾性

意味: 要求の中で一貫性が保たれている事

検証方法: 要求項目間で一貫性が保たれ、矛盾が無い事をチェックする。

具体的には、要求項目間の関係をマトリクスにより、総当たりチェックする。

4.1.5. 重要度と安定性のランク付け

意味: 要求の重要度と安定性のランクが付与されている事

検証方法: 下記をチェックする。

重要度(又は必要性)のランクが定義されている事。

安定性(変更されやすいかどうか)について、ランク付けがされている事。

重要度及び安定性にそのランクが付与された理由が明らかな事。

4.1.6. 検証可能性

意味: ソフトウェア製品が要求を満たしているかどうかを検証できる事

評価の視点: 要求が具体的に定量化されてに表現されているか?

要求に定性的な表現がされていない事。

検証方法: 下記をチェックする。

要求が具体的に定量化されてに表現されている事。

要求に定性的な表現がされていない事。

補足: 形式言語等を利用して、要求記述から定性的な表現を排除する等の方法が望まれる。

4.1.7. 変更可能性

意味: 変更しやすさ
評価の視点: 目次・索引がある事。冗長でない事。個々の要求が分離されている事
検証方法: 下記をチェックする。

目次・索引がある事
冗長でない事
個々の要求が分離されている事

4.1.8. 追跡可能性

意味: 追跡可能性が確保されている事。
追跡可能性には、前方追跡可能性、後方追跡可能性がある。
検証方法: 前方追跡可能性については、下記をチェックする。

要求の発生した原因が参照可能であるかどうか。

前方追跡可能性については、下記をチェックする。

要求を基に作成されたもの(機能仕様、コード等)から、要求が参照可能であるか

4.2. 評価方法の詳細

ここでは、8 要求特性の中で重要と思われる妥当性と完全性を評価する為の方法として、貢献度評価について説明する。

要求分析によって導出された各要求項目を GQM の手法を用いる事によって、貢献度を評価し、この値を利用して、妥当性と完全性の評価を行う。

4.2.1. 貢献度評価の入力情報

貢献度評価の入力情報には、次のような条件が求められる。

要求項目間の関連が明らかである事
要求項目それぞれの目標が明らかである事
当該案件の目的が明らかである事
目的についての目標が明らかである事

この条件について、4 アイテム要求モデルでは、次のようになる。

(1) 要求項目間の関連が明らかである事

関連とは、包含、主従、And、Or 等の事である。4 アイテム要求モデルでは、これらの事を明らかにするように求めている。

(2) 要求項目それぞれの目標が明らかである事

目標とは、その要求項目の達成基準である。4 アイテム要求モデルにおいて、これは、「望まれる状態」として表現される。

(3) 当該案件の目的が明らかである事

4 アイテム要求モデルでは、純粋要求ドメインの主たる要求を、当該案件の目的と定義している。(1)の条件により、要求項目間の関連は明らかにする事になっているので、どれが主たる要求か、つまりどの要求が「目的」かという事についても明らかになっている。

ここで主たる要求は、包含関係又は主従関係のトップに位置するものとして、取り出す事ができる。

(4) 目的についての目標が明らかである事
(2)及び(3)の条件により、「主たる要求」に対応する「望まれる状態」を「目的についての目標」とする。

図 2 に、貢献度評価の入力情報の簡単な例を示す。

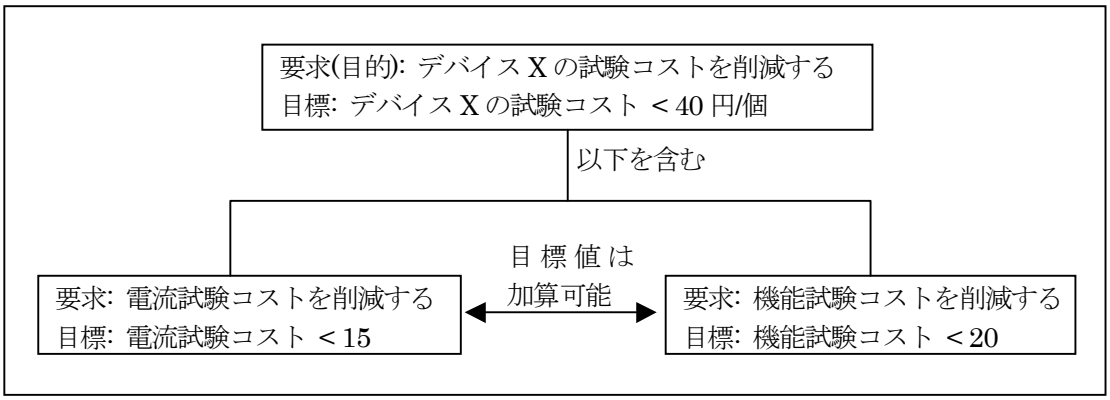


図 2　デバイス試験コストについての要求の例

4.2.2. 貢献度評価手順

一般的な GQM(Goal Question Metrics)の手法を用いる。
GQM については、既に多くの文献が出版されているので、ここでは詳説しない。
次の手順により実施する。

1. 目標に対する貢献度を計る為の質問を用意する。
2. 各要求項目毎に、目標に対する貢献度を要求者に質問する。
3. 各要求項目毎の貢献度の総和が目標に達しているかを要求者に確認する。
4. 誰に質問したか?　等、記録内容を明示する

図 2 の例では、次のような質問が用意される。

A)「電流試験コストを削減する」という要求の実現は、「デバイス X の試験コスト ＜40 円/個」という目標に対し、どれだけ貢献しますか?
B)「機能試験コストを削減する」という要求の実現は、「デバイス X の試験コスト ＜40 円/個」という目標に対し、どれだけ貢献しますか?
C)「電流試験コストを削減する」、「機能試験コストを削減する」という要求の実現により、デバイス X の試験コスト ＜40 円/個」という目標は達成できますか?

これらの質問で得られた貢献度の値により、妥当性と完全性を評価する。
評価結果の一例として、C)の質問に対して、「電流試験及び機能試験以外の試験コストへの言及が不足している。」等の完全性に対する問題が検出される事が期待される。

５　まとめ

この研究では、

- 要求の具体的な検証方法を提案する事

を目標とした。その結果、「貢献度評価による、妥当性及び完全性の評価」を提案した。しかし、手法の評価を実施する事はできなかった。その原因としては、一般に手法の有効性を評価するという事は非常に難しく、この研究で提案した「要求の具体的な検証方法」の評価も例外ではないからである。これについては、実務に適用し、顧客満足度向上という実績を積み上げて、初めてその有効性が認められると考える。

そこで、今後の課題を次のようにする。

- この手法を実務に適用し、考察を深める

＜参考文献＞

1. 大西淳、郷健太郎著、「要求工学」、共立出版（2002）

形式仕様記述言語による派生開発でのモデル検証

豊田　拓也 (株式会社アドバンテスト)

１　背景と目標

当社の業務においても、組み込みソフトウェアの派生開発を行っている。そこでの問題点として「仕様の把握と検証の工数増大」が挙げられた。そこで以下の対応策を考えた。

1. 仕様書を整備する。
2. 設計段階で仕様の正しさを検証できるようにする。

また、派生開発である事から以下の制約も与えられた。

1. 既存仕様書を補完するように、プレーンテキスト形式で適用すること。
2. 記述量や導入コストは極力抑えること。

２　解決へのアプローチ

本研究では、形式仕様記述言語を用いた仕様の整理を考えた。理由は以下の通り。

1. 自然言語による曖昧さを減らしたい。厳密に書きたい
2. （実装からのスペックアウト作業が主な為）詳細な仕様表現に向いている。
3. 設計段階で論理数学による検証が可能。
4. プレーンテキストで表現できる。既存の仕様書に追加する形で適用可能。

また、形式仕様記述言語としてはモデル指向言語の VDM を採用した。理由は以下の通り。

1. 形式仕様記述言語として、比較的表現の自由度が高い。
2. 開発対象が主に逐次処理型

他にも可能性のある技術としては、性質指向言語の OBJ も挙げられる。本研究では上記の理由で採用されなかったが、分散型のシステムに対しては有効であろう。

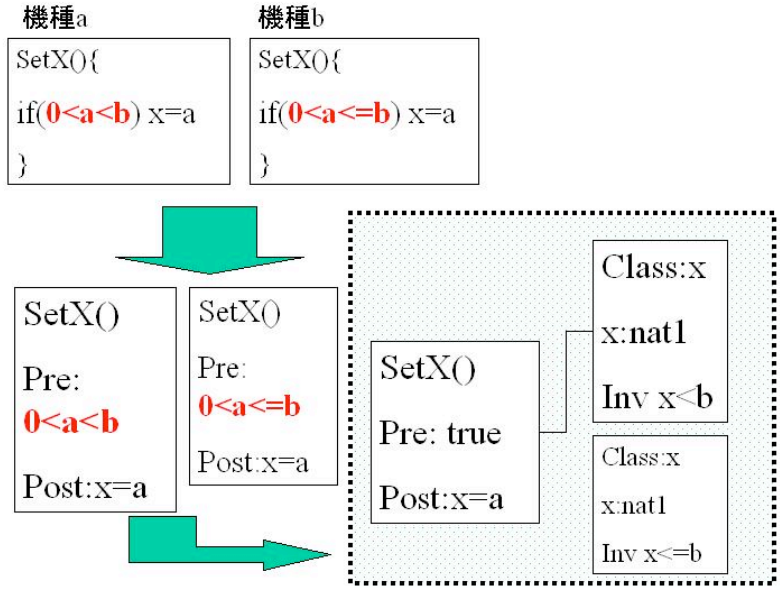
３　導入の工夫

最初に行われるのは、システムに登場する物や操作のモデリングである。モデリングと検証と言えば、モデル駆動型アーキテクチャ(MDA) が有名である。MDA では、まず実装とは独立したプラットフォーム独立モデル(PIM) が作成され、プラットフォーム定義モデル(PDM)と組み合わせて、プラットフォーム特化モデル（PSM）へと変換することができる。

本研究でも、同様のモデルに因んだモデリングの程度をあらかじめ決めておく事で、出力のバラつきを抑える事とした。具体的な作業は既にある実装からのスペックアウトからのモデリングとなるので、PSM 相当のモデルが先ずは見える。だがモデリングのゴールは、さらに PIM 相当とした。理由は以下の通り。

1. システム全体像の理解を助けるモデルとして適当であり、そのモデルを必要としている。
2. MDA のように、生成系や設計検証等へのモデル活用にも期待している。

以下は機種毎の仕様を分離していくモデリングの例である。



また「記述量や導入コストは極力抑えること」に対しては、汎用の VDM をそのまま適用せず、特化した書式として簡単化した。以下はその例である。

型情報 †

CP

--コンパレータ種

型

```
map of nat --TesterPin
to VRadioButton of {UT_TH_CPTYPE_IO,UT_TH_CPTYPE_SP}
```

スコープ

{RW}

不変条件

```
dom this.CP = {p|p ∈ TesterPin } -- 全TesterPinについて存在する。
∀ d1,d2∈ dom this.CP･
VtcSocket.Cond(d1).TesterPinType =
VtcSocket.Cond(d2).TesterPinType ⇒
 CP(d1)=CP(d2) --同一ピン種ではCP typeは一緒。
```

初期値

{* |-> UT_TH_CPTYPE_IO}

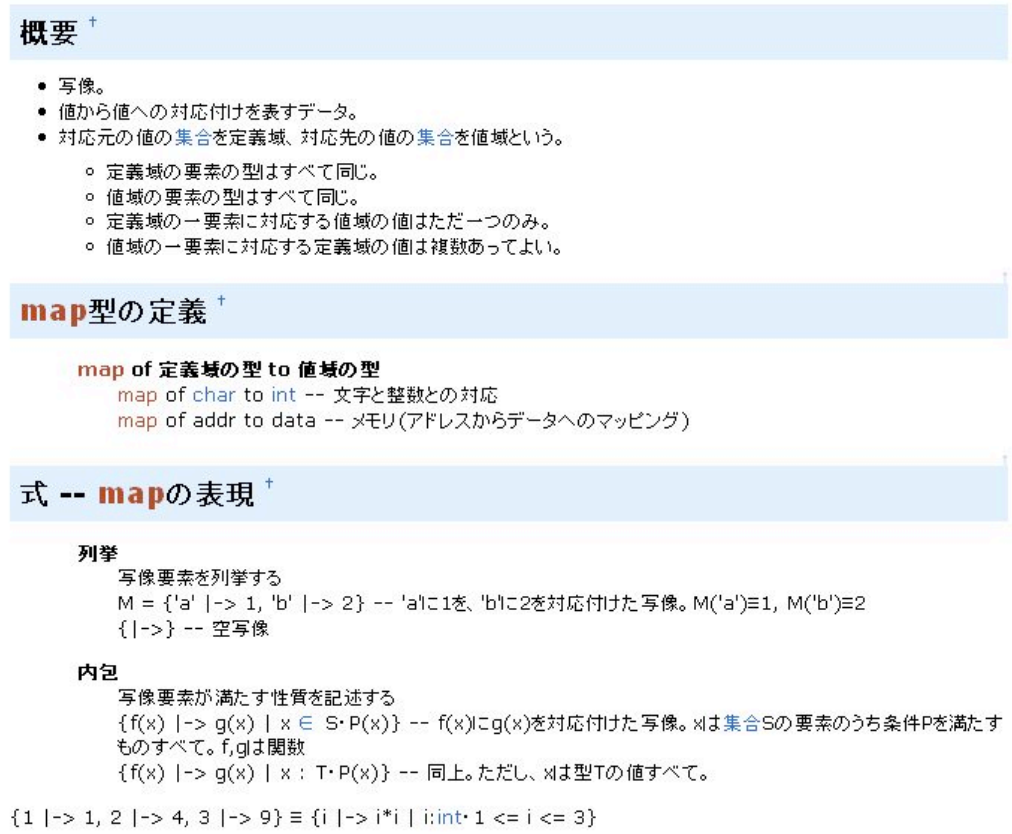
存在単位

クラス

状態遷移

VtcThPinCtrl_Cp.sce

最後に、論理数学の基本知識や表記法をまとめたリファレンスを作成した。これらは全てを wiki 上に構築され、快適なブラウズ環境を心がけた。以下は map(写像)のリファレンス例である。



概要

- 写像。
- 値から値への対応付けを表すデータ。
- 対応元の値の集合を定義域、対応先の値の集合を値域という。
  - 定義域の要素の型はすべて同じ。
  - 値域の要素の型はすべて同じ。
  - 定義域の一要素に対応する値域の値はただ一つのみ。
  - 値域の一要素に対応する定義域の値は複数あってよい。

map型の定義

map of 定義域の型 to 値域の型

```
map of char to int -- 文字と整数との対応
map of addr to data -- メモリ(アドレスからデータへのマッピング)
```

式 -- mapの表現

列挙

写像要素を列挙する

```
M = {'a' |-> 1, 'b' |-> 2} -- 'a'に1を、'b'に2を対応付けた写像。M('a')≡1, M('b')≡2
{|->} -- 空写像
```

内包

写像要素が満たす性質を記述する

```
{f(x) |-> g(x) | x ∈ S･P(x)} -- f(x)にg(x)を対応付けた写像。xは集合Sの要素のうち条件Pを満たすものすべて。f,gは関数
{f(x) |-> g(x) | x : T･P(x)} -- 同上。ただし、xは型Tの値すべて。
```

```
{1 |-> 1, 2 |-> 4, 3 |-> 9} ≡ {i |-> i*i | i:int･1 <= i <= 3}
```

４　効果

以下の効果が得られた。

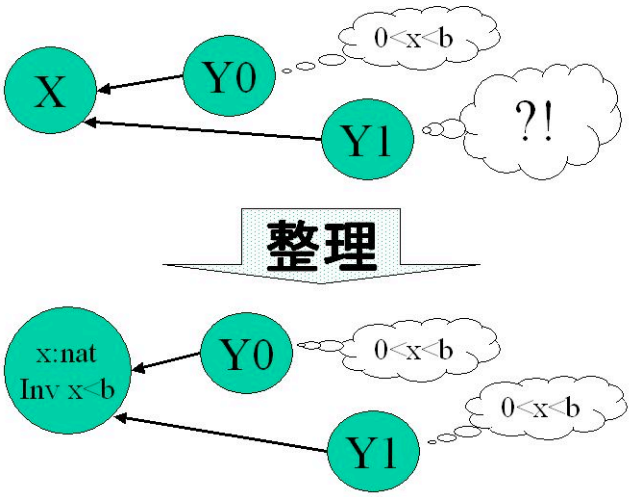
1. 実装に依存しない、シンプルなモデルが得られた。
2. 実装由来の隠れ仕様が明確にされた。（例：属性名は「～リスト」だが本質は集合(set)だった）
3. 人手によるモデル検証が可能となった。
4. 述語論理的な視点により、自然言語（日本語）説明も洗練された。

今後は、これらのモデルを基に改善が期待される。

５　考察

結果的には「MDA を PSM 方向から導くために、形式仕様記述言語を用いてモデリング」という作業となった。今後の派生開発において、実装に依存しないシンプルなモデルが得られたメリットは強力である。

形式仕様記述言語導入による大きなメリットとしては、物が備えている性質をそのクラスの不変条件で定義できる事が挙げられる。 次の図のように、x に関する仕様は x にて定義されるので、x を使う側は仕様を漏らさなくなり、x の仕様を調べるのにも関連クラスすべてを調べる必要も無くなる。最初から望ましく整理されていれば、このような問題は無いだろうが、我々のケースでは効果的であった。



今後は、今回作成されたモデルを用いた各種変換系にも期待しているが、その用途に応じて PDM 相当を用意する事が課題である。特にモデルチェック技術への展開には期待している。一般的には Promela による SPIN が有名であるが、そういった技術との連携についても今後は考えてみたい。

また、UML と形式仕様記述を組み合わせる方法も考えられる。VDMtools というツールは UML を入出力可能であり、コードの生成も可能である。UML との連携においては強力なツールとなるであろう。

６　最後に

第６分科会では、自分の興味ある事について自由に研究する事ができるので、本研究に関してはとても有意義な一年となりました。また、形式仕様記述などという特殊な分野であるにも関わらず、豊富な情報や的確なアドバイスを頂く事ができました。多方面でご活躍されている素晴らしいメンバーと、忌憚なく幅広い意見を交えることが出来る事こそ、第６分科会の素晴らしさです。ありがとうございました。

７　参考文献

- 荒木啓二郎、張　漢明　著、「プログラム仕様記述論」、株式会社　オーム社（2002）
- 佐藤和夫　著、「無故障ソフトウエアを開発する為の…「クリーンルーム手法」紹介」、情報処理（1994）
- 清水吉男　著、「要求を仕様化する技術表現する技術」、技術評論社（2005）
- Stephen J.Mellor/Marc J.Balcer 著、「Executable UML」、翔泳社（2003）
- Martin Fowler/KendallScott　著、「UML モデリングのエッセンス」、アジソン・ウェスレイ・パブリッシャーズ・ジャパン株式会社（1998）

プロジェクト管理における統合モデルの提案

小渕 一幸（セイコーエプソン株式会社）

1　背景

1.1　プロジェクトの評価

日々の作業の中で PM（Project Manager：プロジェクトマネージャ）は、現在のプロジェクト状態を評価し、プロジェクトをコントロールしている。また、組織ではプロジェクトの評価モデルを定義し、全てのプロジェクト状態を監視している場合もある。しかしながら、これらのプロジェクトを評価し、コントロールする活動がプロジェクトの成功に結びつけられて、その効果を実感していることは少ない。

1.2　従来の方法

プロジェクトを評価する方法は、その発想方法により大きく 2 つある。

1 つ目は、その組織で最適と思われる評価モデルを仮定して、それに基づいて評価する方法である。春日ら[1]は、式 1 のようにプロジェクト状況を 8 つの要因別に数値化して総合指標を算出し、プロジェクトの成否判断を行うモデルを提案している。この方法を演繹的予測にもとづくプロジェクト評価（演繹的評価）と呼ぶ。

プロジェクト定量的評価値=∑{(残課題重要度ファクタ×課題対策期限ファクタ)
+プロジェクト進捗状況ファクタ+コスト状況ファクタ}×監視フェーズファクタ
+(試験結果状況ファクタ|検証結果状況ファクタ)+レビュー活動状況ファクタ … (1)

2 つ目は、経験的に何らかの因果関係を発見し、それに基づいて評価する方法である。プロジェクト反省会等での結果分析から特徴的な失敗要因（例えば、残業時間の増加率など）として見出される場合が多い。安部ら[2]は、この方法による理論的なアプローチとして、ベイズ識別器の学習によるモデルを提案している。この方法を帰納的予測にもとづくプロジェクト評価（帰納的評価）と呼ぶ。

1.3　従来の方法の課題

演繹的評価では、そのモデルの精度はモデル作成者の能力に依存することが多い。また、新たな製品領域への適用では、既存の評価モデルを流用することが難しい。さらに、作成したモデルでは説明できない要因が出現しモデルを改良する場合、作成するモデルにさらに説明変数を追加することになり、モデルが複雑になりやすい。

帰納的評価では、要因からプロジェクト結果へ至る因果関係を説明するモデルがないため、評価結果をフィードバックすることが難しい。また、ある程度信頼できる法則を導くためには、多くのデータを収集し相関があるかを確認する必要がある。

2　目的・目標

演繹的評価モデルに対して、帰納的評価で見出される因果関係を取り込み、モデルの改良を行えれば、論理的かつ事実にもとづくフィードバックが行えるようになる。さらに、このようなモデル作成の体系的な枠組みが構築できれば、様々なプロジェクトに対して柔軟にモデルを適用でき、また、評価結果に対するフィードバックも同様の考え方を用いて実施できるようになる。

2.1　目的

本研究では、様々なプロジェクトを効率的に成功に導くために、普遍的なプロジェクト管理フレームワークを提案することを目的とする。また、この方法が実際に活用できるように、典型的なソフトウェア開

発おける具体例を示す。

プロジェクト管理フレームワークは、大きく以下の 2 つから成る。

- プロジェクトの成功を発想の起点として演繹的評価モデルを統一的な方法で作成する枠組み
- 評価モデルから得られる評価値から、プロジェクトを成功に導くためのフィードバックモデル

これにより、様々な組織、プロジェクトにあった評価・フィードバック方法を同一の枠組みで定義できるようになる。さらに、様々なプロジェクト管理に対して同一の枠組みを使用できることから、個々に定義した評価・フィードバックモデルを、同様の手順で改良していくことが可能になる。

3　研究成果

3.1　コンセプト

3.1.1 プロジェクトの評価

プロジェクト評価は「要求が実現できた」、「QCD（Quality, Cost, Delivery : 品質, コスト, 納期）が守れた」など、それぞれの評価要素の達成度で測ることによって行う。評価にあたっては、納期が重要なプロジェクトもあれば、コストが重要なプロジェクトもある。図 1 はプロジェクト評価を視覚的に表したもので、バブルの大きさは評価要素の重要度を表す。

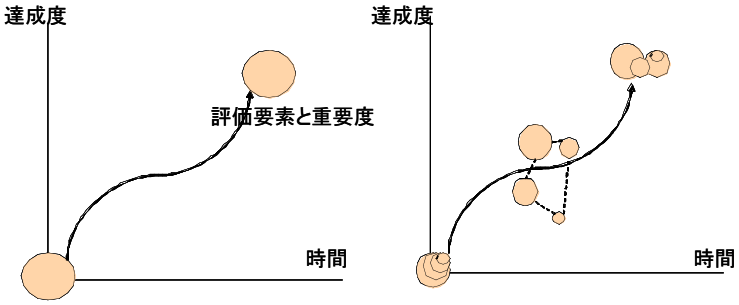


図 1 プロジェクトの評価イメージ

このことからプロジェクト評価値は重要度を w、達成度を a とし、式 2 のように表現することができる。

$$\text{プロジェクト評価値}=\sum(w_i \times a_i) \quad \ldots(2)$$

またプロジェクト管理の別の側面としてリスク管理が重要と言われる。リスクを定量化したリスク値はプロジェクト開始当初は大きいが、徐々に小さくなりプロジェクト終結時には 0 になる性質がある。リスクが問題として発現した場合、その後の評価値に変動を与える。したがって、リスクは評価値に対するばらつきとして捉えることができる。図 2 の点線はプロジェクト開始時のリスクをばらつきとして表現している。

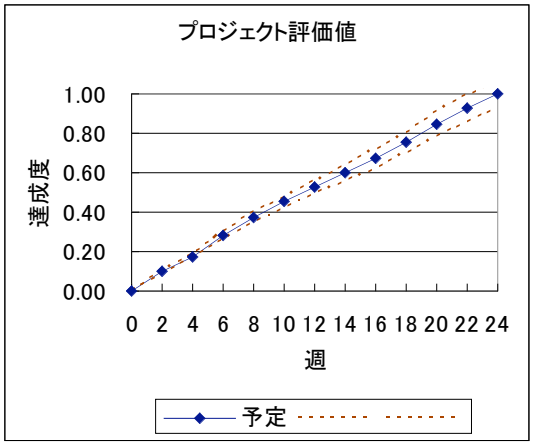


図 2 リスクとばらつき

3.1.2 評価結果のフィードバック

PM は、評価結果から、いくつかのアクション計画を検討し、プロジェクトを成功に導くと思われるアクションを選択することになる。このことは、「アクションが実施されたと仮定して予定評価値を再計算し、評価値が以前よりプロジェクトが成功に近づいているかを判断すること」と捉えることができる。

また、実際の評価では評価要素間の相互作用によるトレードオフを比較検討している。したがって、フィードバックを行うためには、評価要素間の影響度を含めて、再評価する必要がある。評価要素の依存関係は図 3 に示すようにグラフで表現することができる。

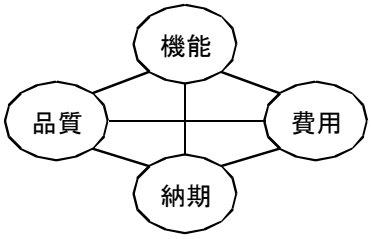


図 3 評価要素の依存関係

3.2 評価モデル

3.2.1 評価モデルの作成

評価要素はプロジェクトの目的、達成基準から導出する。典型的には、要求機能の実現とその品質(欠陥)、成果物を実現するために投入した費用、および納期が考えられる。式 2 を機能 f、品質 q、費用 c、納期 d の評価要素で展開すると式 3 になる。

$$\text{プロジェクト評価値}=\sum(w_i \times a_i)$$

$$=w_f \times a_f + w_q \times a_q + w_c \times a_c + w_d \times a_d \ldots (3)$$

重要度は、評価要素間を同じ次元で定量化する必要がある。しかし、通常、評価要素の尺度は同じではない。例えば、費用は人月であったり、納期は日であったりする。ここでいう重要度は評価要素間の重みを表す。したがって、重要度の算出には、様々な要素間の重みを導出することに適している一対比較法を用いる。

達成度は、評価要素の達成度合いを適切に示す関数とする必要がある。そのため、評価要素の達成基準を明確にし、適切なメトリクスを選択しなければならない。したがって、メトリクスの決定にはゴールを段階的に展開することによりメトリクスを導出することができる GQM 手法[3]を用いる。なお、複数のメトリクスを用いて達成度を算出する場合は、GQM 階層に沿ってメトリクスを合算する時の変換レートであるメトリクス・ウエイトを決定する。また、メトリクスの精度（単位）を決定するには、計測費用と関連する評価要素の重要度を考慮する必要がある。

最後に、プロジェクト評価値は、様々なプロジェクトで普遍的に扱うために正規化する必要がある。つまり、重要度はその合計が 1、達成度はプロジェクト開始時に 0、終結時に 1 となる関数として考える。

3.2.2 評価モデルに対するリスク表現

リスクは、評価要素の評価値に影響を与えるものとして表現される。つまり、リスク値は評価値に対するばらつきとして表現される。また、通常、リスク値は発生確率と影響度として表現されるが、この値はプロジェクトの状態により変化するものと捉える。

3.2.3 評価モデルの具体例

典型的なソフトウェア開発を想定して、評価モデルの作成例を示す。想定するプロジェクトの概要を表 1、スケジュールを表 2 に示す。

表 1 プロジェクトの概要

プロジェクト名	ABCバージョンアップ開発
目標	定例マイナーバージョンアップを前提として請負うため、少ない工数で必要最小限の機能追加および変更を速やかに達成する。 品質の高いソースコード品質を維持しているので、バグを作り込まないように留意する。
工程	設計～評価
メンバ	3人(固定)
期間	24週

表 2 プロジェクトスケジュール

週	2	4	6	8	10	12	14	16	18	20	22	24
作業1	要求	分析	設計	実装								
作業2			要求	分析	設計	実装						
作業3			要求	分析		設計	実装					
作業4								評価	評価	評価	評価	評価

プロジェクト目標から抽出した評価要素として、要求機能の実現(F)、欠陥検出(Q)、費用(C)、納期(D)を考える。評価要素の重要度を算出するための一対比較法の質問票を表 3 に、算出した重要度を図 4 に示す。

表 3 一対比較の質問票

	A	Aが絶対的に重要である	Aがかなり重要である	Aが重要である	Aがやや重要である	AとBは同程度重要である	Bがやや重要である	Bが重要である	Bがかなり重要である	Bが絶対的に重要である	B
1	全ての要求機能を実現する										欠陥(バグ)を残さない
2	全ての要求機能を実現する										工数(要員・残業)を追加しない
3	全ての要求機能を実現する										納期を守る
4	欠陥(バグ)を残さない										工数(要員・残業)を追加しない
5	欠陥(バグ)を残さない										納期を守る
6	工数(要員・残業)を追加しない										納期を守る

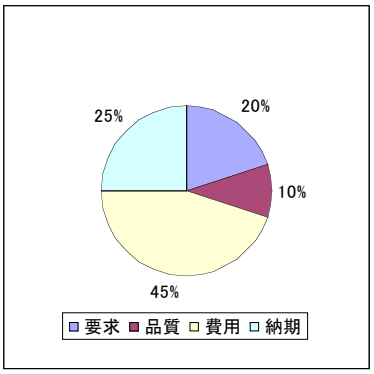


図 4 重要度

GQM 手法を用いて、機能達成度をメトリクスにまで展開する。これにより導出したメトリクスとそのウエイトを表 4 に示す。

表 4 機能における GQM 展開とメトリクス・ウエイト

A1.機能達成基準: (1)機能要求を明確にしたユースケースを作成されている (2)分析、設計、実装を行い既存コードに機能が組み込まれている	配分		メトリクス ウエイト
G1.バージョンアップ予定の全ての機能が動作する			
Q1.1実現機能は明確か?	0.30		
Q1.1.1バージョンアップ予定機能のユースケース記述は作成したか?		1.00	
M1.1.1作成ユースケース数/予定スユースケース数(完了したら1)			0.30
Q1.2実現機能は完成したか	0.70		
Q1.2.1既存機能に関連するクラスとその依存関係は明確か?		0.30	
M1.2.1影響度調査済みクラス数/修正予定クラス数(完了したら1)			0.21
Q1.2.2 設計は完了したか?		0.40	
M1.2.2 …			0.28
Q1.2.3 実装は完了したか?		0.30	
M1.2.3 …			0.21

表 2 および表 4 をもとに作成した達成度関数を式 4 および式 5 に示す。

機能達成度 a_f=∑作業達成度 $_i$

=作業達成度 $_1$+作業達成度 $_2$+作業達成度 $_3$…(4)

作業達成度=∑(メトリクス計測値 $_j$×メトリクスウエイト $_j$)

=要求機能定義率×0.3+影響度分析完了率×0.21+設計完了率×0.28+実装完了率×0.21…(5)

各メトリクスの予定値を表 5 に示す。これに達成度関数を適用し、プロジェクトの開始時の機能における予定達成度を算出した結果を表 6 と図 5 に示す。

表 5 機能における予定メトリクス

分類	要求	分析	設計	実装
予定数	M1.1.1	M1.2.1	M1.2.2	M1.2.3
作業1	3	20	60	60
作業2	1	10	20	20
作業3	1	10	20	20
合計	5	40	100	100

表 6 機能における予定達成度

週	2	4	6	8	10	12	14
作業1	0.18	0.11	0.17	0.13			
作業2			0.06	0.05	0.06	0.04	
作業3			0.06	0.05		0.06	0.04
合計	0.18	0.29	0.57	0.80	0.86	0.96	1.00

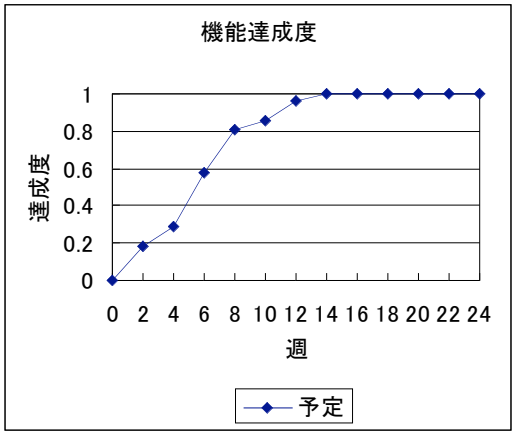


図 5 機能達成曲線(予定)

欠陥検出、費用、納期においても同様に行い評価モデルを作成する。重要度を加味してプロジェクトの予定評価値を算出した結果を表 7 と図 6 に示す。図 5 と図 6 にあまり関連が見られないのは、機能の達成がプロジェクトの成功に大きく寄与しないためである。

表 7 プロジェクト評価値の算出

週	0	2	4	6	8	10	12	14	16	18	20	22	24	ウエイト
機能	0.00	0.18	0.29	0.57	0.80	0.86	0.96	1.00	1.00	1.00	1.00	1.00	1.00	0.20
品質	0.00	0.09	0.14	0.29	0.40	0.43	0.48	0.50	0.55	0.65	0.90	0.98	1.00	0.10
費用	0.00	0.07	0.13	0.20	0.27	0.36	0.45	0.54	0.63	0.73	0.82	0.91	1.00	0.45
納期	0.00	0.09	0.17	0.20	0.22	0.30	0.34	0.41	0.53	0.65	0.77	0.88	1.00	0.25
全体	0.00	0.10	0.17	0.28	0.38	0.45	0.53	0.60	0.67	0.75	0.85	0.93	1.00	1.00

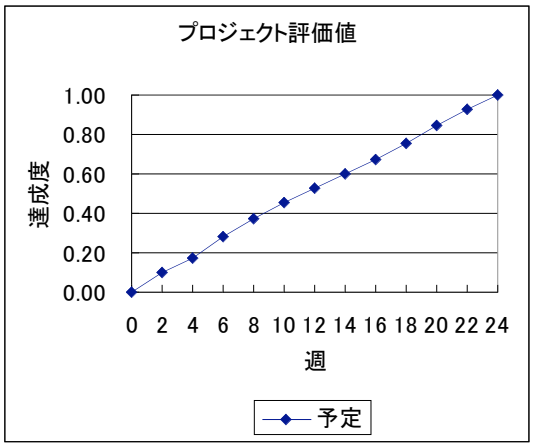


図 6 プロジェクト評価値(予定)

3.2.4 評価モデルに対するリスク表現の具体例

一般的なリスク管理表に時系列でのリスク変化を付加したリスク管理表を表８に示す。発生確率はリスク項目毎に決定する。一方、影響度は評価要素毎に決定する。影響度はそのリスクの大きさを表すので、評価要素毎に適切な尺度で表現する。

表８ リスク管理表

リスク項目	分類	週	2	4	6	8	10	12	14	16	18	20	22	24
評価中に… が起こる		発生確率	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.20	0.20	0.20	0.20
	機能	影響度	…											
		リスク値												
	…		…											
	費用	影響度								−90.00	−90.00	−90.00	−90.00	−90.00
		リスク値	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−36.00	−18.00	−18.00	−18.00	−18.00
Cさんが… に参加できない		発生確率	0.60	0.50	0.40	0.30	0.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00
	…													
バッファを消費しない		発生確率	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	…		…											67.50
	費用	影響度	49.50	49.50	13.50	13.50	45.00	22.50	45.00	67.50	67.50	67.50	67.50	33.75
	費用	リスク値	24.75	24.75	6.75	6.75	22.50	11.25	22.50	33.75	33.75	33.75	33.75	

次に、評価要素毎に＋のリスクとーのリスクを合算後、評価値からの予想される変位としてリスク影響度に変換する。費用リスク影響度を表９に示す。

表９ 費用リスク影響度

	週	2	4	6	8	10	12	14	16	18	20	22	24
費用リスク(＋)	リスク値	24.75	24.75	6.75	6.75	22.50	11.25	22.50	33.75	33.75	33.75	33.75	33.75
	工数	198.00	198.00	198.00	198.00	270.00	270.00	270.00	270.00	270.00	270.00	270.00	270.00
	リスク影響度	0.13	0.13	0.03	0.03	0.08	0.04	0.08	0.13	0.13	0.13	0.13	0.13
費用リスク(－)	リスク値	−10.80	−9.00	−7.20	−10.40	−11.60	−9.80	−6.00	−42.00	−24.00	−24.00	−24.00	−24.00
	工数	198.00	198.00	198.00	198.00	270.00	270.00	270.00	270.00	270.00	270.00	270.00	270.00
	リスク影響度	−0.05	−0.05	−0.04	−0.05	−0.04	−0.04	−0.02	−0.16	−0.09	−0.09	−0.09	−0.09

全ての評価要素のリスク影響度を算出した後、プロジェクト評価値とリスクによるばらつき求める。この結果を図７に示す。

また、６週目まで進んだ状態で、実績値とリスクの見直しを行ったプロジェクト評価値を図８に示す。なお、図８では顧客からの変更要求がない場合として、プロジェクトの予定評価値の見直しは行っていない。上振れリスクを見込んでも当初の予定評価値に達成することは厳しく、プロジェクトの成功は難しいことが認識できる。

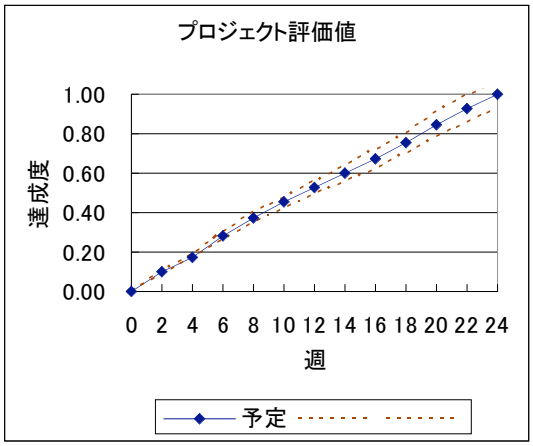


図７ 評価値とリスク

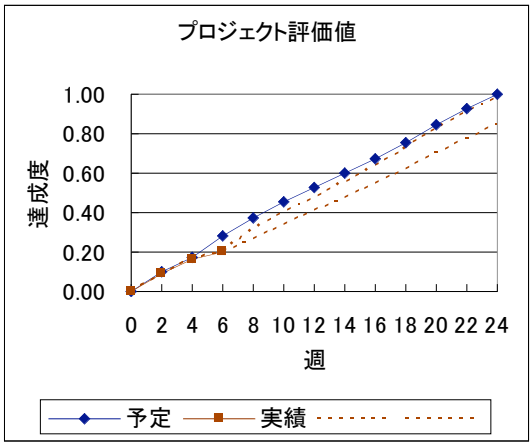


図８ 評価値とリスクの見直し

3.3 フィードバックモデル

以下にフィードバックの考え方を示す。

3.3.1 評価要素間の依存関係の作成

フィードバックにあたっては評価要素間の依存関係が重要となる。このことを視覚的に表現するために効果図式[4]を用いる。図 9 はプロジェクト全体の効果図式の中で工数追加のフィードバックアクション（アクション）を仮定した場合に関連する依存関係を示している。

自然な正の効果、自然な負の効果、PM のアクションにより正にも負にもなる効果を直感的に理解できる。このような効果図式を用いることにより、複数の仮定したアクションを適切に取捨選択できるようになる。

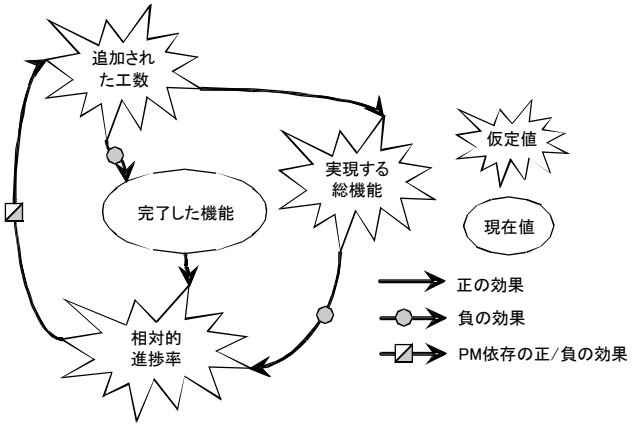


図 9 評価値とリスクの見直し

3.3.2 評価式の再評価

効果図式の辺には評価要素に対する影響度を定義する。PM が仮定したアクションは頂点の追加（または変更）を意味し、ここを起点に影響度を積和することによりアクションが全体に与える影響を求めることができる。影響が無限に伝播しないように閾値を設けて、十分に小さくなれば計算を打ち切る。

効果図式に影響度の伝播の考え方を導入することは、通常 PM が行っているトレードオフの評価を定量化していると考えることができる。例えば、「要件追加で人を追加すると実現機能数が増えるが開発スピードが落ちる」ということを効果図式で見える化しつつ定量的に評価できるようになる。

この影響度を評価要素毎に合算して、いくつかの仮定したアクションに対する評価値の変化としてシミュレーションする。

3.4 モデルの改善

評価要素や達成度関数、重要度、効果図式等は当初は仮定にもとづき作成される。実際の値との相関により徐々に精度を高めて行く必要がある。

3.5 プロジェクト形態によるモデル利用の変化

本枠組みは、プロジェクト形態により適用の視点が変化する。以下に特有の視点の例を示す。

3.5.1 ウォーターフォール

ウォーターフォールでは顧客が最初に凍結した要求が最大価値と認識される。基本的に予定評価値は変化せず、リスクを再評価しながら、変更を最小化するようにコントロールする。

3.5.2 インクリメンタル

インクリメンタルでは評価モデル式が機能ごとに階層化される。インクリメンタル毎に成功基準（顧客価値）を確認し、評価式とリスクを見直す。

3.5.3 アジャイル

アジャイルでは、顧客の価値が最大化されるように振舞う。つまり、リスクのばらつきが顧客の許容範囲に収まるならば、常に変化を受け入れるようにコントロールを行う。プロジェクトの途中で変更要求を受け入れた結果、縮小したリスクのばらつきが再拡大してもよい。

４　考察

本枠組みにより様々なプロジェクトに対してプロジェクト全体を俯瞰しながら管理できる見通しを得た。プロジェクト成功に必要な要素を重点志向により管理することになるため、PM に対して、枝葉末節にこだわらない大局観をもたらすと期待できる。

特に、リスクをばらつきと捉えることにより、プロジェクト全体での許容範囲を認識し、柔軟性のあるプロジェクト管理が行える。

また、フィードバックにおいては、変動の全体に与える影響を定性的/定量的に検討しながらアクションの選択が行える見通しを得た。

本枠組みの適用にあたっては、組織レベルで固定/可変にする部分、プロジェクトレベルで固定/可変にする部分を決めることにより、導入が促進されると思われる。

５　今後

本年度は、フィードバックモデルについては概念的な説明にとどまり、具体例の提示が十分に行えなかった。フィードバックモデルは、グラフを行列表現にすることにより、定量的評価が行いやすくなると考えられる。

また、実適用を促進するためには、モデルの改善方法についても今後検討する必要がある。

これらは、実適用による検証を行いながら確認していく必要がある。

６　参考文献

[1] 春日君夫, 福島利彦, 山田茂: "実践的なソフトウェアプロセス監視活動の取り組み", 第 25 回ソフトウェア品質シンポジウム発表報文集, pp.319-326, 2006.

[2] 安部誠也, 濱崎考成, 水野修, 菊野亨: "ソフトウェアプロジェクト混乱に対するベイズ識別器による予測の試み", ソフトウェアシンポジウム 2004 論文集, pp.137-144, 2004.

[3] V.Basili, G.Caldiera, and H.D.Rombach: "Goal Question Metric Approach: Encyclopedia of Software Engineering", pp. 528-532, John Wiley & Sons, Inc., 1994.

[4] G.M.Weinberg: "ワインバーグのシステム思考法", 共立出版, 1994.