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　：　小笠原　秀人（(株）東芝）
副主査　：　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））
木村　初夫（アンリツエンジニアリング（株））
後藤　玄次（(株）富士通北陸システムズ）

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、このような活動においては、壁にぶつかることもよくある。このような時に、外部の方からアドバイスをもらったり、同じ悩みを持った方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。

今年度、上述したような活動を実践するために、本分科会を立ち上げた。結果として、「リスクを低減するプロジェクト計画」、「ソフトウェア開発プロセスの点検」、「4 アイテム要求モデルによる要求分析」に関する報告書、論文という形式で 3 つの成果が得られた。また、他のメンバの問題や課題を一緒に議論する中からヒントを得られる、といった効果があった。

Abstract

For the effective development of software products in a organization, it is necessary to introduce software engineering techniques and to promote software process improvement activities. Many people have been trying to improve their software development process for high quality. But, those activities do not often work. It is very important for practitioner to share ideas and practices of people with same situation. For the purpose, we started the research project for quality software development.

As a result, project members completed their research activities relate to “project planning”, “verification of software development process”, and “requirement analysis” as paper and report respectively.

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を実践の場で適用・評価・考察するという作業が必須である。

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ひとつのゴールとして本分科会の活動をスタートした。

2. 活動経過

本分科会は、臨時分科会 1 回を含め、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。また、問題分析手法や IDEAL（Initiating – Diagnosing – Establishing – Acting – Learning）サイクルを活用した改善活動の進め方を、主査、副主査から説明した。最初の 2 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。

合宿を含めた 7 月から 12 月までの分科会では、各メンバが解決のためのアイデアを持ち寄り、それをメンバ全員で検討して、アイデアを実践に移すための具体的な項目まで落とし込むことに注力した。落とし込まれた項目は、各自、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、「アイデア出し」→「分科会の場で検討」→「自分の業務の中で実践」という良いサイクルが確立できてきた。

臨時分科会を含めた最後の 2 回の分科会では、4 月から実践してきた内容を各自整理し、報告書、論文という形式でまとめた。

3. 研究成果

今年度は 3 名のメンバが参加して、上述したとおり、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 つの報告書と 1 つの論文が成果としてまとまった。

＜報告書＞

(1) テーマ：リスクを低減するプロジェクト計画についての一考察
報告者：木村 初夫（アンリツエンジニアリング（株））
参　照：168 ページ
目　的：

1) プロジェクトの失敗予測に関しベイズ識別器を使ったデータマイニング手法が提案されているが、予測の元となるアンケートはプロジェクトが完了した後、プロジェクトメンバーに対しアンケートを実施している。そこで本研究ではプロジェクト開始時点での計画レビュー時に記録された評価票をもとにベイズ識別器による成功/失敗プロジェクトクラス分けを行い、適用事例を検証する。
2) エンタープライズ系ソフトウェアを開発するプロジェクトは、工期あたりの規模の大きな場合にコスト超過を引起す割合が高いことが報告されている。本研究では組込み系ソフトウェア開発のプロセスデータ分析によりプロジェクト遂行に影響を与える定量的な要因を把握し、PM(Project Manager)、PL(Project Leader)が前記予測と指標を効果的に活用する方法を提案する。

(2) テーマ：『ソフトウェア開発プロセス点検』の実践と考察
報告者：後藤 玄次（(株）富士通北陸システムズ）
参　照：171 ページ
目　的：

ソフトウェア開発では、プロジェクトにおけるＱＣＤ（品質、コスト、納期）のバランスを定義し、それらを管理し、予定期間内、予定コスト内で期待通りの品質を実現することが求められる。しかし、技術の高度化、要求の多様化のなかで、短期開発を要求される今日のソフトウェア開発において、開発部門のプロジェクト管理者の努力のみで開発プロセスの質を維持し、最終プロダクトの品質を確保することは、困難になりつつある。そのため、品質検証部などの

外部部門による品質支援活動の展開が必要となる。

当社品質検証部では、事業部製品に対する製品検査業務を行う一方で、ＳＩ部門開発ソリューションに対する第三者検証を行い、ソフトウェア成果物に対する検証を行っている。しかし、ＳＩ部門開発においては、納期、コストへ意識が傾き、開発プロセスの十分性が不透明なまま工程を進めているプロジェクトが散見され、本来、未然に防止できるはずの品質問題が、最終成果物に対する第三者検証の段階で顕在化することが増えてきた。

そこで、開発プロセスの十分性を定量的に把握し、いかにして、品質を予測しながら開発していくかという課題が、改めて持ち上がった。

本稿では、課題に対する取組の途中経過とその考察について述べる。

＜論文＞

(1) テーマ：4 アイテム要求モデルによる要求分析

報告者：岡田 文和（アドバンソフト開発（株））

参　照：173 ページ

目　的：

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も散見されている。

このような状況を鑑みて、要求分析にまつわる要求として、次のようなものが挙がった。

1) 如何に精度の高い要求を少ない工数の中で見つける事ができるにしたい
2) 上記スキルを持ったエンジニアを短時間に育成したい

本年度の研究では、上記要求に応えるため、以下のことを目標とした。

① 簡単に導入できる要求分析手法を提案する
② 提案された手法を評価する

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、このような活動においては、壁にぶつかることもよくある。このような時に、外部の方からアドバイスをもらったり、同じ悩みを持った方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。

今年度、上述したような活動を実践するために、本分科会を立ち上げた。結果として、論文として 1 つ、報告書として 2 つ、まとめることができた。自分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、同じよう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、本分科会を立ち上げ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。

リスクを低減するプロジェクト計画についての一考察

木村 初夫（アンリツエンジニアリング（株））

研究の目標

1）プロジェクトの失敗予測に関しベイズ識別器を使ったデータマイニング手法が提案さているが、予測の元となるアンケートはプロジェクトが完了した後、プロジェクトメンバーに対しアンケートを実施している。そこで本研究ではプロジェクト開始時点での計画レビュー時に記録された評価票をもとにベイズ識別器による成功/失敗プロジェクトクラス分けを行い、適用事例を検証する。

2）エンタープライズ系ソフトウェアを開発するプロジェクトは、工期あたりの規模の大きな場合にコスト超過を引起す割合が高いことが報告されている。本研究では組込み系ソフトウェア開発のプロセスデータ分析によりプロジェクト遂行に影響を与える定量的な要因を把握し、PM(Project Manager)、PL(Project Leader)が前記予測と指標を効果的に活用する方法を提案する。

活動内容

予測モデル

ソフトウェア開発モデル(Wモデル)における各工程、デザインレビュー(DR-A～DR-C)、失敗予測、プロセスDBの関係を図2-1に示す。本モデルは要求分析工程のDR-A等の評価結果から成功/失敗を予測するとともに当該工程における是正処置(対応するテスト工程を含む)及び次工程以降への申し送り事項、プロセスDBへのプロセスデータ登録を表わしている。図下方のプロセスDBから失敗予測システムへ向かう矢印は、プロジェクト完了によるプロセスDB更新後、プロセスDB分析結果からプロジェクト遂行に影響を与える要因を抽出して予測システムの予測精度やリスク検出能力を向上させることを表現している。本研究はDR-A時点での失敗予測①とプロジェクト遂行に影響を与える要因の抽出②を実施する。

分析結果

プロジェクト失敗予測

DR-Aの評価結果が現存する7プロジェクトから表3-1に示すデータ一覧を作成した。プロジェクトd1～d3が成功プロジェクト、d4～d7が失敗プロジェクトである。プロジェクトの定性的な情報を確認後、完了結果の納期・コスト・品質どれかひとつでも3の場合、失敗プロジェクトと判断した。

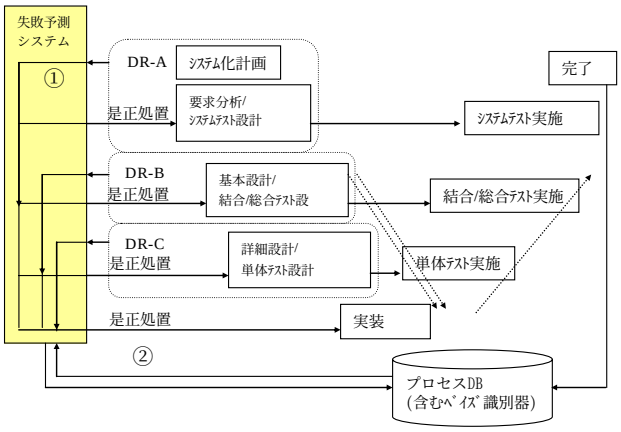

図2-1 プロジェクト失敗予測モデル

表 3-1　DR-A での評価結果データ

Projects	DR-A 評価項目							判定	完了時結果		
	市場分析	ｽｹｼﾞｭｰﾙ	ｺｽﾄ	ﾘｽｸ	品質	体制	ｺﾐｭﾆｹｰｼｮﾝ		納期	ｺｽﾄ	品質
D1	3	3	2	3	3	3	3	成功	1	0	1
D2	2	2	2	2	2	3	2	成功	0	0	1
D3	3	3	3	3	2	3	3	成功	0	1	1
d4	2	2	2	2	1	2	2	失敗	3	3	3
d5	3	3	2	3	3	3	3	失敗	3	0	1
d6	3	3	3	3	3	3	2	失敗	0	0	3
d7	3	2	2	3	2	4	4	失敗	0	3	0
d8	3	3	3	3	3	3	3	成功	0	0	0

DR-A 評価結果の各項目(市場分析～ｺﾐｭﾆｹｰｼｮﾝ)は5：秀、4:優、3:良、2:可、1:不可の5段階評価
完了時結果は0から3の4段階で評価、0は予定遵守、1～2:予定未達成、3は予定大幅乖離、失敗。

【結果】 単純ベイズ法による分類検証

前記データを用い単純ベイズ法を使って完了プロジェクト d8 が成功失敗どちらかに入るかを検証する。d8 は成功プロジェクトであった。P(d8|成功)は d1～d7 の成功確率(3/7)と d8 の各項目に対する成功プロジェクトの値の数え上げから、

P(d8|成功)=3/7×2/3×2/3×2/3×2/3×1/3×3/3×2/3=0.009406　となり、失敗は

P(d8|失敗)=4/7×3/4×2/4×1/4×3/4×2/4×2/4×1/4=0.002511　となる。

P(d8|成功)＞P(d8|失敗)なので成功に属す。同様にしてd1～d7 のプロジェクトに対しても失敗予測した。データ数の制約から予測不可のプロジェクトもあったが、正答率は0.375 であった。ここでは割愛するがベイズの定理及び計算方法の詳細は水野らの研究[1)] を参照されたい。

表 3-2　失敗予測結果

実際の結果	予測結果	
	成功	失敗
成功	3	予測不可
失敗	予測不可	予測不可

8PJ で 3PJ 正答　　正答率：0.375

プロジェクト失敗要因

コスト超過または納期超過を起こした失敗プロジェクトグループとそれ以外のプロジェクトグループ間で開発規模等に有意差があるかを調べた。

1）コスト超過

検定した全項目でコスト超過ありグループとなしグループで有意な差異が見られなかった(表3-3)。

2）納期遅延

納期遅延グループとそれ以外のグループ間で有意な差が現れた項目は工期予実差と工期あたりの規模予実差であった(表3-4)。

表3-3 コスト超過-検定結果

検定項目	検定手法	
	t 検定	Mann-Whitney の U 検定
規模予実差(KLOC)	有意差なし	有意差なし
工数予実差(人月)	有意差なし	有意差なし
工期予実差(月)	有意差なし	有意差なし
工期あたりの規模予実差 (KLOC/月)	有意差なし	有意差なし
バグ件数予実差	有意差なし	有意差なし

表3-4 納期遅延-検定結果

検定項目	検定手法	
	t 検定	Mann-Whitney の U 検定
規模予実差(KLOC)	有意差なし	有意差なし
工数予実差(人月)	有意差なし	有意差なし
工期予実差(月)	[両側検定] 1%有意差あり [片側検定] 1%有意差あり	両側検定：1%有意差あり
工期あたりの規模予実差 (KLOC/月)	[両側検定]有意水準5%有意差なし [片側検定]有意水準5%有意差あり	両側検定：有意水準5%有意差あり、1%有意差あり
バグ件数予実差	有意差なし	有意差なし

まとめ

1）プロジェクト失敗予測モデルを提案し、計画レビュー時、組織既存の評価項目(表3-1)を使って失敗予測が可能である一例を示した。

2）定量的データの影響要因について、当初計画の工期あたりの規模（KLOC/月）の維持が失敗回避に重要であることを検定等から特定した。

3）今後の課題

① 技術的要因(経験、技術力等)項目追加やDR-A後の施策展開結果をDR-Bの評価項目に加えるなど各工程における評価項目(表3-1)の選定手順を明確にして、失敗予測モデルを完成する。

② プロジェクトの属性(規模等)に合わせたベイズ識別器の評価方法や評価項目(表3-1)と関連付けたプロセスデータ分析方法を考案する。

参考文献

1）水野修、安部誠也、菊野亨、「プロジェクト混乱予測システムのベイズ識別器を利用した開発」、SEC journal 創刊記念論文、Oct.2005

『ソフトウェア開発プロセス点検』の実践と考察

後藤 玄次（(株）富士通北陸システムズ）

1. 背景／目的

ソフトウェア開発では、プロジェクトにおけるＱＣＤ（品質、コスト、納期）のバランスを定義し、それらを管理し、予定期間内、予定コスト内で期待通りの品質を実現することが求められる。しかし、技術の高度化、要求の多様化のなかで、短期開発を要求される今日のソフトウェア開発において、開発部門のプロジェクト管理者の努力のみで開発プロセスの質を維持し、最終プロダクトの品質を確保することは、困難になりつつある。そのため、品質検証部などの外部部門による品質支援活動の展開が必要となる。

当社品質検証部では、事業部製品に対する製品検査業務を行う一方で、ＳＩ部門開発ソリューションに対する第三者検証を行い、ソフトウェア成果物に対する検証を行っている。しかし、ＳＩ部門開発においては、納期、コストへ意識が傾き、開発プロセスの十分性が不透明なまま工程を進めているプロジェクトが散見され、本来、未然に防止できるはずの品質問題が、最終成果物に対する第三者検証の段階で顕在化することが増えてきた。

そこで、開発プロセスの十分性を定量的に把握し、いかにして、品質を予測しながら開発していくかという課題が、改めて持ち上がった。

本稿では、課題に対する取組の途中経過とその考察について述べる。

2. 解決のためのアプローチ

上記課題を踏まえ、第三者である品質検証部員が開発プロセスの十分性をチェックし、徹底を促し、潜在する品質問題を早期発見して対策へのトリガーとすることを目指し、『ソフトウェア開発プロセス点検活動』に取り組むこととした。

－点検実施タイミング

開発の立上げ時、及び各工程の完了時に点検を実施する（場合によっては、工程会議毎)。

・開発の立上げ時

開発・品質計画書を点検し、見積開発量、スケジュール、体制、品質計画等の検証を行う。

・各工程の完了時

進捗資料及び開発データ（開発量、レビュー時間、テスト項目数、検出バグ数　他）を点検し、プロセスの充分性を検証する。点検対象工程は、以下の通り。

設計工程：外部仕様設計、内部仕様設計、詳細仕様設計

製造工程：コーディング、単体テスト、結合テスト、機能テスト、総合テスト

－点検観点の統一

点検観点が人に依存し、担当者による点検結果のバラツキの発生を防ぎ、点検を均質なものとするため、チェックリストを作成した。各チェック項目についても、結果判定に曖昧さが入る余地を極力排除するよう、充分吟味した。

－点検結果の記録

各項目のチェック結果は、以下の３パターンで表現する。

○：問題なしと判定

×：問題ありと判定　または　記載事項やデータがなく判定できない

－：判定対象外

必要情報が報告されていない場合、“判定対象外”とはせず“問題あり”と判定する。これは、開発状況の不透明化に繋がる問題へのアラームを意図するためである。

－点検結果の定量表現による可視化

点検の実施結果から開発プロセスの良し悪しを表す定量的表現として、「開発プロセス良好率」を設定した。

「開発プロセス良好率」：チェック結果○の数÷(チェック結果○の数＋チェック結果×の数)

－点検結果のフィードバックと報告

点検結果は、「開発プロセス点検結果連絡票」としてまとめ、開発元へ送付する。その際、チェックリスト内容を全て開発元へ開示する。これは、開発者が最初からチェック項目の内容を意識したプロセスの実施を心掛けることを期待するからである。

点検の結果問題ありと判断したプロジェクトに対しては、対面ヒアリングの場を設定し、開発リーダ及び必要に応じて幹部社員への参加も要請する。

また、事業部門長への報告会も定期的（1 回／月）に設定し、開発プロセスの状況をダイジェストとして報告した。

3. 適用結果

以上、述べてきた『ソフトウェア開発プロセス点検』への取り組みは、まだ、緒についたばかりであるが、途中経過ながら、一定の効果が見えてきた。

・開発部門のプロセス品質状況の改善（開始時点の良好率：４５%、現時点の良好率：７０%）
・プロセス品質チェック観点を開示することにより、開発者に対し、開発プロセス良くするための具体的指針を与える効果（現チェック項目：４４項目）
・定量表現の採用により、開発プロセスの状況を、各層が同じモノサシで共有

一方、本取組みで実現したプロセス品質の定量化モデルは、まだまだ改善の余地を残すが、今後継続し、さらに成熟させていくことで、次の効果へ繋げていくことが期待できる。

・開発プロセス良好率と成果物品質との相関関係明確化
・開発プロセス良好率と問題兆候と具体的対策事例へのリンク
・開発プロセス標準のテーラリング指針

4. まとめ

本分科会では、『ソフトウェア開発プロセス点検』の立上から実践過程を考察する中で、種々の課題を議論してきたが、「品質を作りこむ主体は人間である」ことをしっかり抑えておくことの重要性を、強く認識するに至った。つまり、技術論や職業倫理（高品質を実現するのは仕事なんだからきちんとやりなさい）を幾ら振りかざしたところで、生身の人間である開発者の質の高いモチベーション維持への配慮なくして、高品質の実現はなしえないということである。

奇しくも、『ソフトウェア開発プロセス点検』の実践においても、点検結果を○／×で表現した「開発プロセス点検結果連絡票」を開発者へ送付したが、開発者との会話を通して、“×”という表現が必要以上に嫌悪感を生んでおり、素直に指摘を受け入れられなくなる開発者のメンタリティへの配慮不足に気づくことになった。

今後は、できていない部分を早期に指摘するのは当然であるが、開発者の努力の結果、できている部分にもフォーカスし可視化する仕組みや、気持ちが前に進むような表現への配慮を盛り込むことにより、さらに、効果の高い活動になるという当分科会での指摘を踏まえ、活動を磨いてゆきたい。

4アイテム要求モデルによる要求分析

The Requirements analysis method by 4 Items Requirements Model

岡田 文和（アドバンソフト開発株式会社）

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も散見されている。

このような状況を鑑みて、要求分析にまつわる要求として、次のようなものが挙がった。

1) 如何に精度の高い要求を少ない工数の中で見つける事ができるようにしたい
2) 上記スキルを持ったエンジニアを短時間に育成したい

この研究では、上記要求に応えたい。

1. 本年度の研究目標

本年度の研究では、

① 簡単に導入できる要求分析手法を提案する
② 提案され手法を評価する

これらを目標とした。

要求分析の結果、抽出される要求に求められる事として、次の事が挙げられる。これらに沿って、要求を評価する事とする。<参考文献2>

・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

要求分析手法に求められる事として次の事が挙げられる。手法についても、これらに沿って評価する事とする。

・定義された要求の精度が高い事
・方法論が明確でわかりやすい事
・簡単に導入・実施できる事

2. 研究成果

2.1. 要求ハンドリングにまつわる失敗事例

【顧客に言われた通りに作ったケース】

顧客担当 SE が「計測器から取得したデータを表計算ソフトで統計処理するソフトウェアを作って欲しい」との顧客要求を取得した。入力データのフォーマット及び統計処理の仕様を明確にし、「実現には表計算ソフト A を使用する事」と明記された仕様を開発部署に回した。開発で、その仕様と表計算ソフト A の調査を実施したが、表計算ソフト A ではその仕様に提示された統計処理の実現が難しいという事がわかった。そこで、開発担当は、表計算ソフト A で実現する事の必然性について、顧客に確認してもらうよう顧客担当 SE に要請した。しかし、顧客担当 SE は、実現方法の大幅な変更が納期を大きく遅らせる事と、表計算ソフトで実現する事は顧客要求であり絶対であるという事を理由にこれを聞き入れなかった。顧客の確認がとれないまま、開発は当初の要求の通り、表計算ソフトを使用して統計処理ソフトウェアを開発したが、そのソフトウェアは使い物にならず、破棄される事となった。

2.1.1. 素朴な疑問

上記のような失敗から、次のような疑問があがった。これが、この研究のスタートポイントである。

- ユーザが言うところの要求は正しいのか?
- 要求はどのようにハンドリングすればよいのか?
- 誰が本当の要求を知っているのか?
- 本当の要求を知る為の正しいアプローチは?
- そもそも、要求とは何なのか?

2.1.2. 失敗事例の分析

この失敗事例は、「間違った要求通りに作ってしまう」というものである。その発生原因として、次のものが挙げられた。

① 顧客が何故、その統計処理を必要としているのか？ どのような状態で使用するのか？ 他のシステムとの関連等が明らかにされていなかった。
② 「顧客の言った事」の正当性・重要性を検証せず、盲信していた。
③ 機能仕様についての重要性は認識されながらも、ソフトウェアの利用のされ方についての重要性が認識されていなかった。

つまり、「顧客が本当に必要としているものを明らかにする事が重要である」という事が再認識された。

2.1.3. 要求分析の誤りが引き起こす問題の特性

ソフトウェアの開発工程において、何らかの誤りが発生した場合、その発見が遅れるほどリカバリーのコストは大きくなると言われている。Boem は、リカバリーコストを次のように示した。(ここでは、「要件」は「要求」と同等のものと解釈している。)

表1 誤りを訂正する為の相対費用<参考文献1より>

誤りを見つけたフェーズ	相対費用
要件	1
設計	3-6
コード化	10
開発試験	15-40
受入試験	30-70
稼動	40-1000

要求分析はソフトウェアの最初の開発工程であるため、その誤りが大きなリカバリーコストを発生させてしまう危険性を内包していると考えられる。

また、要求仕様については、開発工程で作成される他ドキュメントと比較して、その誤りの検出が難しいという事が懸念される（未検証の仮説）。誤りの検出には、リファレンスとなるドキュメントとの整合性をチェックするという方法がよく使われる。例えば機能仕様の正当性は、要求仕様をリファレンスとしてチェックする事が可能であるが、要求仕様はソフトウェアの開発工程で作成される最初のドキュメントである為、その正当性を確認するためのリファレンスが整理された形態では存在しない事に起因する。

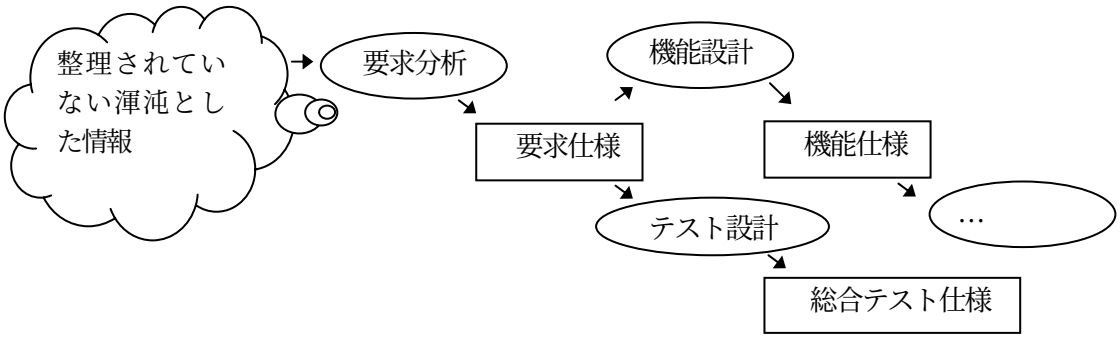

図１　工程とその成果物

2.2.　要求とは?

2.2.1.　純粋要求とシステム要求

要求はソフトウェアの開発において、スタートポイントとなる重要な情報である。それは何なのかを確認しておきたい。IEEE では、Requirements を次のように定義している。<参考文献 2 より>

(1) 問題を解決したり、目標を達成するために、利用者にとって必要な条件や能力
(2) 契約書、標準規格書、仕様書、その他の正式な文書を満たすために、システムやシステムの構成要素が満たすべき、あるいは持つべき条件や能力
(3) ソフトウェアやソフトウェアの構成要素を開発していくための基礎となる全ての要求の集合
(4) ソフトウェア要求仕様書(Software requirements specification)という用語の短縮形

(1)は、利用者の着目しているアプリケーション領域における最も基本的な「要求」を表わしていると考えられる。ここには「システム」や「ソフトウェア」等の言葉は使われていない事から、ソリューションを排除した本質的な要求を指していると解釈し、この研究では、「純粋要求」と呼ぶ事とする。

(2)は、(1)をシステムで実現する為に必要とされる事項に変換されたものであり、ISO9126 の品質特性で表現可能な領域であると考えられる。この研究では、(2)を「システム要求」と呼ぶ。

2.2.2.　要求を明らかにするステップ

要求開発<参考文献 3>では、システムを開発するステップを図 2 のように定義している。「ビジネス分析」や「ビジネス要求」等、「ビジネス」という言葉が使われているのは、ビジネス分野のシステム開発との親和性を高める為と思われる。ビジネス以外の分野を取り扱う場合は、この「ビジネス」という言葉をそれぞれのアプリケーション領域を表わす言葉に読み替える事が可能と思われる。また、「ビジネス要求」は前述の IEEE の Requirements の「(1)問題を解決したり、目標を達成するために、利用者にとって必要な条件や能力」を、「システム要求」は同様に「(2) 契約書、標準規格書、仕様書、その他の正式な文書を満たすために、システムやシステムの構成要素が満たすべき、あるいは持つべき条件や能力」と同等のものを指していると考えられる。

この中で、「要求開発」で定義する工程は、「システム要求開発」の部分であり、ビジネス分析の工程に

ついては、言及していない。従って、ビジネス要求を定義する事は、要求開発の入力を定義する事に等しい。

表2に、本研究で扱う「要求」と他の文献との関連を示す。

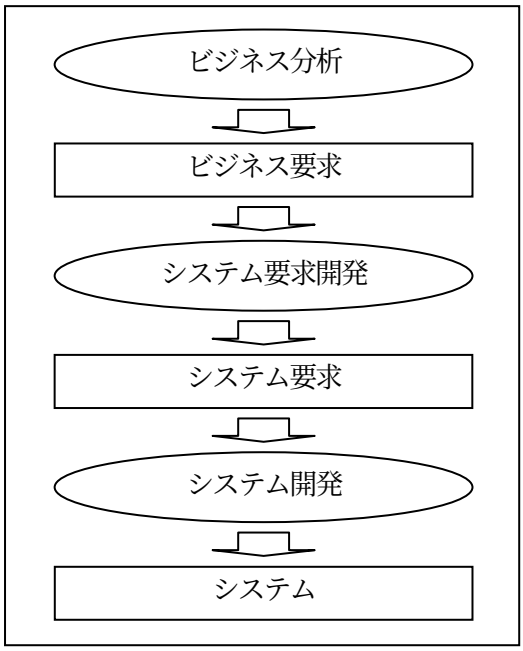

図2　要求開発の定義する開発ステップ

表2　本研究で扱う「要求」と他の文献との関連

IEEE の Requirements	要求開発	本研究
(1) 問題を解決したり、目標を達成するために、利用者にとって必要な条件や能力	ビジネス要求	純粋要求
(2) 契約書、標準規格書、仕様書、その他の正式な文書を満たすために、システムやシステムの構成要素が満たすべき、あるいは持つべき条件や能力	ビジネス要求	システム要求

2.2.3. 純粋要求とシステム要求の関連

純粋要求とシステム要求は、「目的」と「実現方法」の関係にあると考えられる。従って、どちらが重要であるかは自明である。「目的」に合わせて最適な「実現方法」を選択すればよい。前述の失敗事例では、顧客の言った不確かな「実現方法」にこだわるあまりに、「目的」がおろそかにされたと考えられる。しかし、要求者の満足度は、「自分の言った事に対し、忠実に対応したか?」ではなく、「的確に目的を遂行したか?」ないしは「的確に問題に対処できたか?」によって決まる筈である。もし、「目的」又は純粋要求が明らかにされていたならば、失敗のリスクは大きく低減されていたと推測される。

ここで、純粋要求を明らかにする事の意義として次の事が期待される。

- 要求者の本質的に求めるものを明らかにし、これに対応する事で顧客満足度を高める。
- 精度の高いシステム要求を導き出せるようにする。
- システム要求又はシステム仕様を検討する上での制限事項を最小化、対応コスト、対応工期を低減できるようにする

このような期待から、この研究では、「純粋要求」を導き出す事に主眼を置く事とする。

2.3. 要求の構造

2.3.1. 要求モデル

要求の発生する過程を次のように考えることができる。

「現在の状態」を観察する
↓
「現在の状態」と「望まれる状態」との乖離を発見する
↓
その乖離が「解決されるべき問題」として認識される
↓
「解決されるべき問題」を解決するために「要求」が発生する

上記の過程から、この研究では、要求とその周辺を次の 4 つのアイテムでモデル化する。
この研究では、これをこの「4 アイテム要求モデル」と呼ぶ事とする。それぞれのアイテムは次の事を表す。

(1) 現在の状態
問題領域における「解決されるべき問題」を含んだ状態とその周辺環境の状態を表わす。
(2) 望まれる状態
「現在の状態」から「解決されるべき問題」を取り除いた状態であり、ステークホルダの望む状態。
(3) 解決されるべき問題
解決されるべき問題 = 現在の状態 - 望まれる状態、で表わされる。ある問題領域に属する「問題」。
(4) 要求
要求 = 望まれる状態 - 現在の状態、で表わされる。

図 3 は、4 アイテム要求モデルの全体像と、それぞれのアイテムの関係を示したものである。

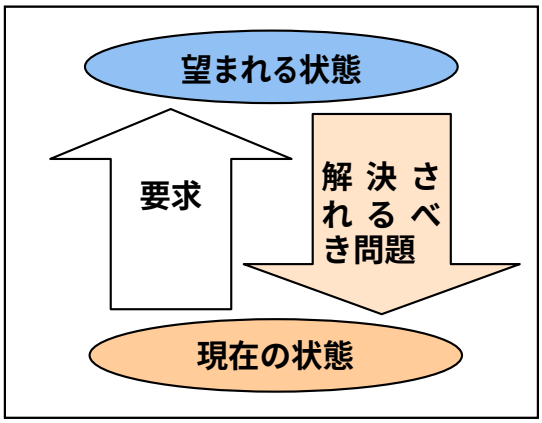

図 3　4 アイテム要求モデル

2.3.2. ドメイン

要求事項を整理する場合には、その要求がどのドメイン（問題領域）に属するのかを意識する必要がある。企業の業務とそのシステム化について議論する場合には、業務ドメインがアプリケーション・ドメインである。更にその周辺のドメインとして考えられるドメインの一例を示す。

財務ドメイン
企業戦略ドメイン
業務ドメイン(アプリケーション・ドメイン)

システム要求ドメイン
システム仕様ドメイン
実装ドメイン

一方、この研究では、純粋要求を明らかにする事を主眼に置き、ソリューションと分離するアプローチを取っているので、それぞれ、「純粋要求ドメイン」と「ソリューション・ドメイン」をおく事とする。システムというソリューションを開発する立場からみれば、財務、企業戦略、業務の各ドメインは純粋要求ドメインに、システム要求、システム仕様、実装の各ドメインはソリューション・ドメインにそれぞれ属している。

純粋要求を明らかにするという事は、アプリケーション・ドメインでの要求を明らかにする事であり、その発生原因は同じアプリケーション・ドメインか又はその上位のドメイン、例では財務又は企業戦略ドメインに求めることができる。

2.3.3. 表層要求

要求収集活動の初期においてステークホルダが「要求」として上げた事を指す。表層要求は、対応者がどのように対応すれば良いかを考えるための重要な手がかりである。

2.3.1 では、要求の発生する過程を説明したが、要求者サイドにおいて、それぞれのステップが漏れなく正確に行われる事はまれである。従って、表層要求はステークホルダの「要求」を正しくあらわしてない事が多く、次のような問題を含んでいる事に、対応者は注意する必要がある。

- 整理されていない
- ノイズを含んでいる
- 漏れがある
- 複数のステークホルダ間の「要求」の違いが調整されていない
- 「要求」がソリューション・ドメインの情報で表現される

最後の『「要求」がソリューション・ドメインの情報で表現される』について、考察しておきたい。前述の失敗事例においても、「表計算ソフトで実現する」というソリューションが表層要求として対応者に伝達されていた。問題を見つけた場合には、その解決方法（ソリューション）を具体的に考えるのも自然なことであり、また、要求者は対応者にソリューションを発注するという状況を想定すれば、これを直接に表現すれば、ソリューションの形態をとるのが自然であると考えられる。しかし、注意しなければならないのは、前述の通り、顧客満足度は、ソリューションを忠実に具現化したかではなく、問題を的確に解決したかによって決まる筈である。従って対応者は、表層要求を手がかりに「解決されるべき問題」あるいは「純粋要求」を明らかにすることが求められる。

2.4. 要求分析手順

(1) ステークホルダの識別

ステークホルダを検出した場合には、下記の事を記録する。

- 氏名
- どのような立場か?
- 当該要求とどのようなかかわりがあるか?

他のステップを実施している時にも、ステークホルダ検出される事がある。 その場合には、このステップに戻って、情報を収集する。

(2) 表層要求を収集する

各ステークホルダから、表層要求を収拾する。収集した情報については、次の事を記録しておく。

- 表層要求の内容
- 誰から入手した情報か？

(3) 4 アイテムを明らかにする

表層要求を手がかりに、ステークホルダに次のようなインタビューする。

- 解決したい問題は何ですか？
- その目的は何ですか？
- 実現したい事はどんな事ですか？
- 現状は、どのようになっていますか？

回答がどのドメインに属する情報なのかを区別して記録する。純粋要求ドメインの情報が得られるまで、掘り下げる。必要に応じて、ソリューションドメインでの 4 アイテムを明らかにしておく。「現在の状態」や「望まれる状態」については、シナリオ、データフロー、ワークフロー等で表現する。

4 アイテムの全てを必ずしも要求者から引き出す必要はない。あるアイテムが決まれば、簡単な変換により、他を導き出すことができるものもある。例えば、「解決されるべき問題」を反転させれば「要求」が導き出せる。

図 4 は、純粋要求ドメインとソリューションドメインとの関係を表したものである。

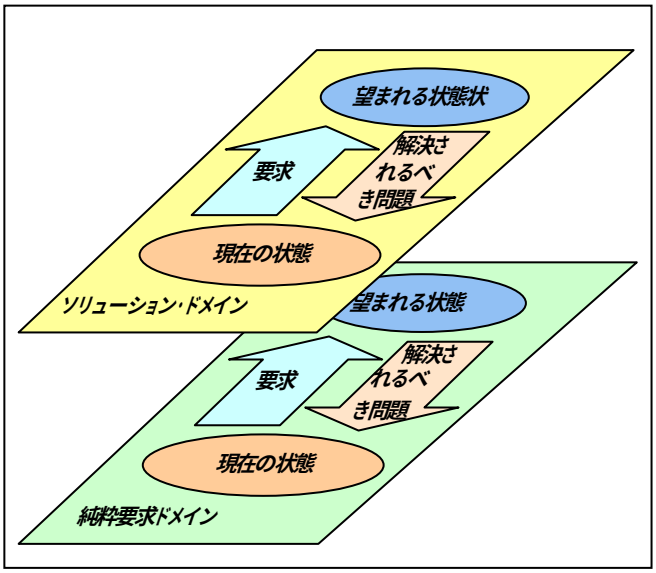

図 4 純粋要求ドメインとソリューションドメインの関係

(4) 複数の要求の間に矛盾があれば、調整する

通常、一つのシステムは複数の要求により開発され、更にステークホルダは複数である。矛盾が発生するのは当然の事である。　矛盾の発生原因を調べて適切な対処をすべきである。

例えば、「ステークホルダ間の情報伝達の不徹底」が原因なら、正確な情報を提供すればよいし、「ステークホルダ間の利害の不一致」が原因なら、折衷案の提示やプライオリティ付け等が考えられる。

(5) 導き出された 4 アイテムをレビューにより検証する

レビューには、ステークホルダに参加してもらい、4 アイテムの妥当性を確認する。

2.5. 手法のトライアル

手法の有効性を確認するために、要求分析のトライアルを行った。

このトライアルでは、次の二点を目的とした。

- 手法の有効性を検証する
- 純粋要求を抽出することで、分析結果にどのような変化が現れるかを確認する

対象は、XEnv という名称のバーチャルプロジェクトである。

同じ分析対象に対して、二種類の分析を行う事とする。基本的な分析手順は、2.4 に示した通りである。二種類の分析の差異は、分析をソリューションと純粋要求のいずれのドメインを基盤に行うかである。

前提事項

XEnv:	計測器を使用する為の総合オペレーション環境ソフトウェア。これを使用する時には、高価な計測器も同時に使用する事になる。
XEnv 環境ファイル:	XEnv からどのような計測データをロギングするかを設定するためのテキストファイル。
XEditor:	XEnv 環境を設定する為の GUI ソフトウェア。 XEnv を起動した状態で使用する。 XEnv 環境ファイルをメニュー形式で表示・編集する機能がある。但し、メニュー形式で表示・編集する為の情報については、計測クラス DLL に依存している。
計測アプリ:	計測クラス DLL を組み合わせて作成される
計測クラス DLL:	計測方法をサポートするクラスライブラリ

(1) ステークホルダの識別

- 氏名　Fred Olson
- 立場　デバイスメーカ A 社の計測技術取りまとめ役
- 要求とのかかわり

  実際の XEnv ユーザーであり、直接に XEnv の要求を発している。
- その他

  A 社には、他にもステークホルダが存在すると思われるが、Fred 氏が担当窓口である為、氏以外と接触することはできなかった。

  （要求対処側のステークホルダについては、割愛）

(2) 表層要求を収集する

各ステークホルダから、表層要求を収拾する。収集した情報については、次の事を記録しておく。

- 表層要求の内容

  XEditor を Standalone（XEnv を立上げる事なく）で使えるようにして欲しい
- 誰から入手した情報か?

  Mr. Fred Olson

(3) 4 アイテムを明らかにする

【現在の状態（ソリューションドメイン）】

利用シナリオで表現した。

0) あらかじめ、XEnv 環境ファイルの記述方法は学習しておく

1) 計測に先立って、XEnv 環境ファイルのうち、基本的な設定（計測データとその処理方法の関連付け等）をテキストエディターで記述し、保存する。これを記述する為には、計測クラス DLL が、どのような計測データやデータ項目を持つのかを知る必要があり、その為には、ドキュメントを参照するか又は記憶に頼る

2) XEnv を立上げる。(立上げに一分程の時間を要する)

3) 計測アプリをローディングする

4) XEnv 環境ファイルを XEnv に設定する。XEnv 環境ファイルに記述ミスがあれば、エラーが標示される。これを修正して、再設定する。

5) XEditor を立上げる

6) 計測アプリを実行し、出力された計測結果を確認しながら、XEditor で XEnv の設定を調整する。

7) XEditor で、XEnv 環境ファイルを保存する

8) 1)～7)を繰り返す

【解決されるべき問題（ソリューションドメイン）】

0) あらかじめ、XEnv 環境ファイルの記述方法を学習しておくのは煩わしい

1) XEnv 環境ファイルをテキストエディターで編集して XEnv 環境ファイルを作るのは煩わしい。

1-1) 計測クラスがどんな計測データやデータ項目を持っているのか、ドキュメントを見て調べなければならない。

1-2) 記述ミス、タイプミス等が発生してしまう。

2)

2-1) XEditor を使用するには XEnv を起動する必要がある。

2-2) XEnv の起動に 1 分を要する。

【望まれる状態（ソリューションドメイン）】

システムの利用シナリオで表現した。

0) XEnv 環境ファイルの記述方法を学習する必要はない。

1-1) Standalone XEditor を立上げる

1-2) Standalone XEditor で XEnv の設定を調整する。この時、計測クラスのドキュメントを参照する必要はない

1-3) Standalone XEditor で、XEnv 環境ファイルを保存する

2) XEnv を立上げる。(立上げに一分程の時間を要する)

3) 計測アプリをローディングする

4) XEnv 環境ファイルを XEnv に設定する。

5) XEditor を立上げる

6) 計測アプリを実行しながら、XEditor で XEnv の設定を調整する。

7) XEditor で、XEnv 環境ファイルを保存する

8) 1)～7)を繰り返す

注 [現在の状態] の 1)が 1-1)から 1-3)に置き換えられた

シナリオから、状態だけを抽出。

a) XEnv 環境ファイルを簡単に編集できる。

b) XEnv 環境ファイルを編集するのに要する時間が最小化されている。

【要求（ソリューションドメイン）】

a) XEnv 環境ファイルを簡単に編集できる

XEnv 環境ファイルの編集操作がわかりやすい。

手入力しなければならない情報は最小化されている。

計測クラスのドキュメントを参照しなくても、XEnv 環境ファイルを編集できる。

XEnv 環境ファイルのシンタックスを学修する必要がない。

タイプミスが発生しない

b) XEnv 環境ファイルを編集するに要する時間が最小化されている

XEnv 環境ファイルの編集ツールの、操作に必要な時間が短い。

XEnv 環境ファイルの編集ツールのオーバーヘッド時間（起動時間、終了時間、Apply 処理時間）が短い。

【要求（ソリューションドメイン）から導き出されたソリューション】（参考）

1) XEnv を起動することなく、XEditor を起動できる（Standalone Xeditor）。

Standalon Xeditor は NonStandalone XEditor と同等の機能を提供する。

1-2) 計測クラス DLL から、計測データとそのデータ項目を取得できる

2) XEnv の起動に要する時間を短縮する

【現在の状態（純粋要求ドメイン）】

シナリオで表現した。

0) あらかじめ、計測データの処理方法の設定の仕方は学習しておく

1) 取得する計測データとその処理方法を設定する。

設定ミスによりエラーが発生する事がある。

設定方法が煩雑で煩わしい。

2) 計測対象を計測し、取得された計測結果を確認しながら、処理方法を調整する

3) 処理方法を保存する。

4) 1)～3)を繰り返す

【解決されるべき問題（純粋要求ドメイン）】

0) あらかじめ、計測データの処理方法の設定の仕方を学習しておく必要がある

1) 計測方法を設定するのが煩わしい

1-1) 設定方法が煩雑で煩わしい。

1-2) 設定ミスによりエラーが発生する事がある。

2) 計測方法設定の準備に時間を要する。

【望まれる状態（純粋要求ドメイン）】

シナリオで表現した。

0) 計測データの処理方法の設定の仕方は学習する必要はない。

1) 計測データとその処理方法を記述し、保存する。(簡単に、短時間に)

2) 計測対象を計測し、取得された計測結果を確認しながら、処理方法を調整する（簡単に、短時間に）。

3) 処理方法を保存する。

4) 1)～3)を繰り返す

シナリオから、状態だけを抽出。

a) 計測データの処理方法を簡単に設定できる。

b) 計測データの処理方法を設定するのに要する時間が最小化されている

【要求（純粋要求ドメイン)】
a) 計測データの処理方法を簡単に設定できる。
b) 計測データの処理方法を設定するのに要する時間が最小化されている

【要求（純粋要求ドメイン）から導き出されれたソリューション】(参考)
ソリューションとして、二つの方法が提示された。いずれの方法が要求者の状況にマッチするかを検討する必要がある。

1) Standalone XEditor を開発して提供する
2) この顧客用にカスタマイズされた計測クラス DLL を提供し、計測方法の設定を不要とする。

(4) 複数の要求の間に矛盾があれば、調整する
このバーチャルプロジェクトでは、課題が単純であったためか、矛盾は発生しなかった。

(5) 導き出された 4 アイテムをレビューにより検証する
このバーチャルプロジェクトでは、課題が単純であったためか、特に指摘は無かった。

2.6. 手法の評価

論文の冒頭において、「定義される要求に求められる事」を紹介した。4 アイテム要求モデルによる要求分析結果をそれに沿って評価する。

【要求分析結果の評価】
① 妥当性
分析のフェーズで「解決されるべき問題点」が明らかになっている為、導き出されたソリューションの妥当性をチェックしやすくなった。
ソリューションドメインを基盤にした分析では、現在の状態や要求を具体的に検討することができる一方で、要求の一般化が足りないために、導き出されたソリューションの幅も狭くなってしまった。
純粋要求ドメインを基盤にした分析では、今ひとつ具体性に欠けるものの、「純粋要求」が明らかになっていたため、ソリューションに与える自由度が大きくなった。

② 非曖昧性
「現在の状態」と「望まれる状態」をデータフローやシナリオで表現する事により、排除できるように考慮した。

③ 追跡可能性
情報源は必ず記録する事により、追跡可能性は確保されるように考慮した。

これ以外の項目については、残念ながら考慮されなかった。
・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

④　その他

- 純粋要求ドメインでのシナリオ記述は、具体性を欠いているので、問題の本質を捉えづらい面がある。純粋要求ドメインでの記述の工夫が求められる。
- 「現在の状態」は「解決されるべき問題」を「望まれる状態」は「要求」をそれぞれ含んでいる。この為、4 アイテムの記述に冗長な部分が生じている。これについても、工夫の余地がある。
- 分析のステップの内、「複数の要求の間に矛盾があれば、調整する」、「導き出された 4 アイテムをレビューにより検証する」の二つについては、評価できなかった。

また、「要求分析手法に求められる事」についても、同様に紹介した。

①　定義された要求の精度が高い事
　これについては、前述の「要求分析結果の評価」に任せる。
②　方法論が明確でわかりやすい事
③　簡単に実施できる事

単純で明快な要求モデルを提示し、それにそった手順を紹介した。本質的な評価は、この手法を筆者以外のエンジニアにレクチャーし、実践してもらうことで評価したい。

3. まとめ

この研究では、

①　要求分析手法を提案する
②　提案され手法を評価する

を目標として研究した。

その結果、①については、4 アイテムモデルによる分析方法を提案することができた。しかし、手法の評価については、十分に実施できなかった。その原因としては、

・サンプル数が少ない事
・導き出された要求の評価方法が無い

という事が挙げられる。

サンプル数については、今後、増やしていけばよいが、導き出された要求について、次の点を満たしているかどうかをどのように評価するのか、その方法が必要である。

・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

手法の向上だけでなく、要求の評価方法についても、今後の課題としたい。

＜参考文献＞

1. D.C.ゴーズ、G.M.ワインバーグ著、「要求仕様の探検学」、共立出版（1993）
2. 大西淳、郷健太郎著、「要求工学」、共立出版（2002）
3. 山岸耕二、安井正男、萩本順三、河野正幸、野田伊佐夫、平鍋健児、細川努、依田智夫著、「要求開発」、日経 BP（2006）