20-J007

要求工学知識体系(REBOK)と
ユーザ指向要求工学の調査研究

―ユーザとベンダのかけ橋となる要求工学の実践―

平成 21 年 3 月

社団法人　情報サービス産業協会

概要

平成 20 年度　要求工学知識体系(REBOK)とユーザ指向要求工学の調査研究
—ユーザとベンダのかけ橋となる要求工学の実践—

1. 調査目的

本調査研究は、情報システム開発における最大の課題である要求開発、管理の現場で要求工学を活用するための知識とその人材育成の指針となる要求工学知識体系(REBOK: Requirements Engineering Body Of Knowledge)[リーボックと呼ぶ]の検討、ならびに、ユーザとベンダが協調した要求開発の実践を目指し、事例の収集、整理を行った。

要求工学に関連する知識体系は幾つか提案されているが、要求工学を現場で活用し、経営戦略や製品戦略を実現する情報システムを開発するためには、要求開発に参画するアナリストだけでなく、ユーザや経営者を含む関係者が要求工学について一定の理解が必要である。REBOK はこのような観点から検討を行い、ユーザとベンダが共通にもつべき知識の枠組み、知識水準を整理し、体系化した。あわせて、要求工学に関連する知識体系の位置づけを明らかにした。さらに、これまでの調査研究で収集した要求工学ベストプラクティスや事例集を REBOK に統合し、包括的で実践的な要求工学知識体系を構築する基盤とした。要求開発、管理の実践、さらには、要求開発を主導するアナリストやプロジェクトマネジャーなどの人材育成の指針として頂きたい。

2. 要求開発・管理に関わる知識体系の課題

要求工学とは、情報システムに対する要求を組織的、体系的に扱う技術体系である。我が国のソフトウェア開発の現場では、要求定義が最大の課題となっている。図 1 に示す調査は JISA 会員企業を対象としたものであるが、要求定義は、ユーザとベンダの共通の課題である。このような要求開発プロセスの重要性の認識は、近年、産業全体に広がりつつある。しかし、現場で要求工学を実践するためには、解決すべき課題が多い。この中で、本調査研究では、現場で要求開発を主導できる人材育成のための知識の整理、体系化を課題とした。

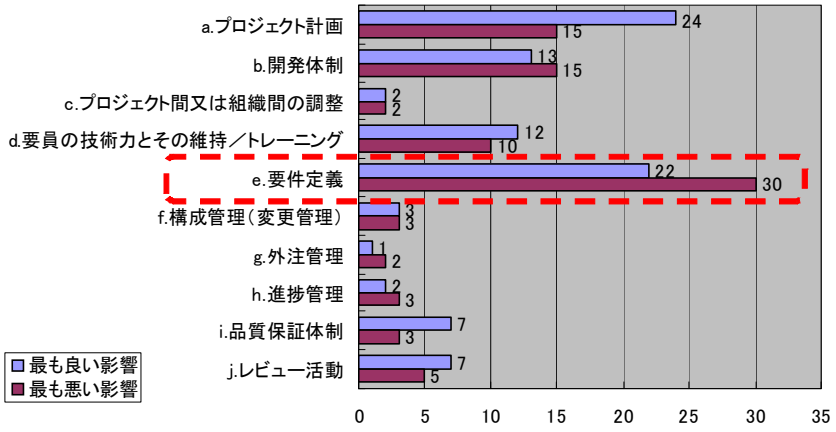


図 1. 情報システム開発における課題

要求工学に関連する主な知識体系として、表 1 に示すものが提唱されている。

(1) ソフトウェア工学知識体系(SWEBOK: Software Engineering Body of Knowledge)とソフトウェア開発専門職の認定として CSDP(Certified Software Development

Professional)と CSDA(Certified Software Development Associate)

(2) ビジネス分析知識体系(BABOK: Business Analysis Body Of Knowledge)と高度な能力を持つビジネスアナリストの認定 CBAP(Certified Business Analysis Professional)

(3) IT スキル標準(ITSS)、 情報システムユーザスキル標準(UTSS)、 組込みスキル標準(ETSS)

(4) 要求工学プロフェッショナルのための教育プログラムと要求工学の初心者レベルの認定試験 CPRE (Certified Professional for Requirements Engineering)

表 1 要求工学に関連する知識体系とその特徴

知識体系	認定	対象者	内容	認定レベルと前提条件
SWEBOK	CSDP, CSDA	ソフトウェア開発者	ソフトウェア工学中のソフトウェア要求工学	なし
BABOK	CBAP	ビジネス分析者	ビジネス分析	高度専門家(7,500 時間(5年)以上のビジネス分析実務経験など)
ITSS	―	コンサルタント, IT アーキテクト	情報技術とスキル	中～高度技術者(スキルレベルを設定)
IREB カリキュラム	CPRE	要求分析者	要求工学	初心者

3. 要求工学知識体系 REBOK とは

要求開発は、対象のスコープに応じて、エンドユーザやユーザの経営者など、多様なステークホルダが関与することに特徴がある。そのため、要求開発を適切に行うためには、要求分析の主導者がエンドユーザや経営者などのステークホルダと協調できる必要がある。このことから、要求アナリストだけでなく、エンドユーザや経営者も要求工学の基礎知識を持つことや理解できることが望まれる。このような観点から、要求工学知識体系(REBOK)への要求を次のように定めた。

(1) ベンダ、ユーザを問わない共通の知識体系

(2) 要求アナリストに加えて、エンドユーザ、経営者など、要求開発に関与するステークホルダが、必要に応じて習得すべき範囲と水準を整理した知識体系

(3) ビジネス要求、システム要求、ソフトウェア要求の 3 つのスコープに応じた知識体系。

(4) エンタープライズシステム、組込みシステムの要求開発に共通する技術の知識体系。ただし、ドメイン固有の知識は個別に定義されるものとする。

これをまとめて要求工学知識体系のフレームワークとして、図2に示す。

この知識体系は、ソフトウェア工学知識体系(SWEBOK)の中の要求工学知識体系に基づき、ビジネス要求とシステム要求のスコープへ対応できるように拡張した。REBOK は、図 3 に示す 8 つの主要な知識要素から成る。

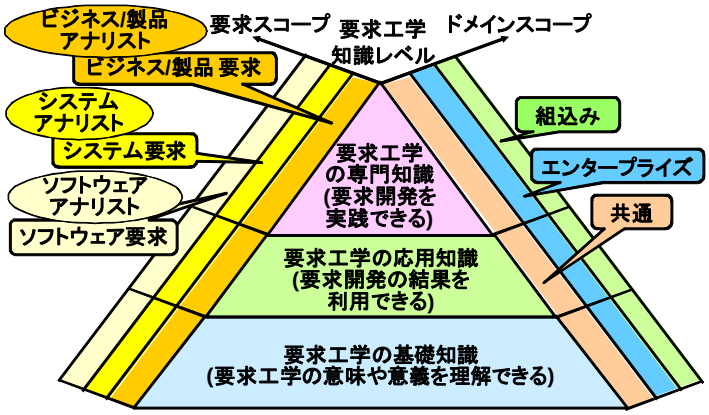


図 2 要求工学知識体系のフレームワーク

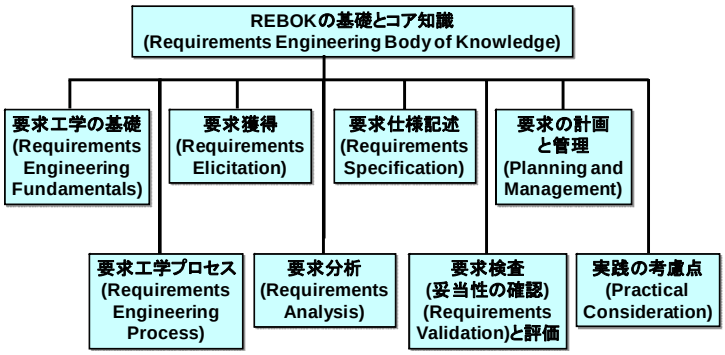


図 3 要求工学知識体系 REBOK

REBOK では、要求工学に関与するステークホルダを、図 4 に示す、次の 3 つの役割モデルで定義する。なお、この役割モデルは、ユーザ、ベンダ共通であることに注意して頂きたい。

(1) 要求アナリスト(Requirements Analyst)

要求開発、管理の実行者。対象のスコープに応じて、さらに、ビジネス要求アナリスト、システム要求アナリスト、ソフトウェア要求アナリストに分けることができる。また、アナリストの専門性のレベルとして要求される知識水準は、初級、中級、上級の 3 つのレベルを想定する。

(2) 要求の利用者(Requirements User)

要求アナリストが要求を獲得するための協力者となりえるエンドユーザや要求開発の成果物を理解し、利用するユーザ、ソフトウェア開発者、プロジェクトマネジャーなど、要求開発の基礎的な知識を持つことを想定する。

(3) 要求工学の理解者(Requirements Engineering Supporter)

要求工学をユーザやベンダ内で、組織的取組みを推進する、CIO や経営者など、要求工学の意義を理解することを想定する。

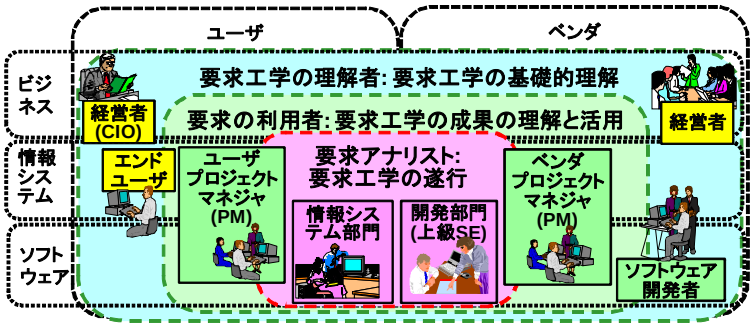


図 4 要求開発における 3 つの役割と期待される知識水準

REBOK の知識要素の間の関係を図 5 に示す。

基礎知識として、要求工学の基礎と要求プロセスを位置付けている。要求プロセスは、要求開発、管理の全体の枠組みを規定する。要求開発の知識である、要求獲得、要求分析、要求仕様化、検査(妥当性の確認)と評価は要求プロセスに基づいて実行され、相互に関係する。要求の計画と管理は、要求開発の 4 つの知識を適用した要求開発を計画、管理する。

実践知識として位置付けている実践の考慮点は、要求開発、管理を実践する上で知っておくべき知識やノウハウである。ユーザ, ベンダにおける実践経験から抽出したノウハウを整理し、要求工学ベストプラクティスとして組み入れている。あわせて、事例を事例集としてまとめている。

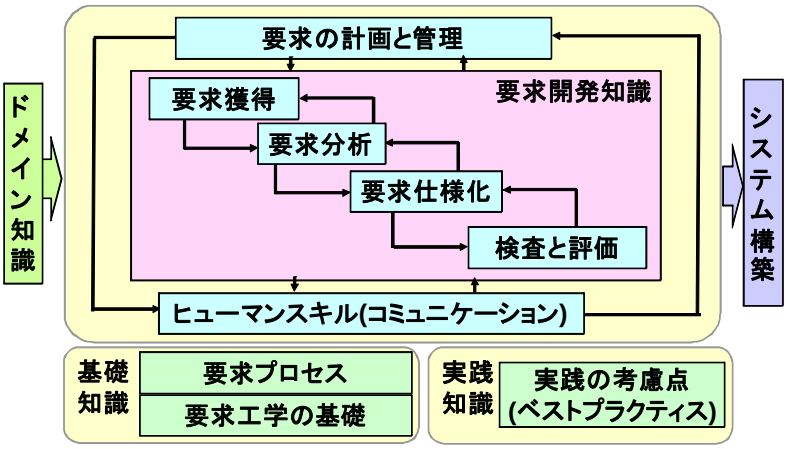


図 5 要求工学知識要素間の関係

4. まとめ

本調査報告者は、要求開発に関与するステークホルダが活用できる要求工学知識体系 REBOK[リーボック]の枠組みを示した。さらに、REBOK に基づく実践を支援する要求工学のノウハウとして、要求工学ベストプラクティスと実践事例集を収集し、まとめた。本報告書が要求開発、管理の実践、ならびに、人材育成において活用されることを期待する。

なお、要求工学知識体系は引き続き検討を行う予定である。

はじめに

本報告書は、現在、情報システム開発において最も重要な課題となっている要求開発・管理を現場で実践するための知識を整理、体系化した「要求工学知識体系(REBOK: Requirements Engineering Body Of Knowledge)[リーボックと呼ぶ]」の枠組みを報告するものです。要求工学を実践する関係者すべてが持つべき知識とその水準の指針とし、あるいは、要求開発を主導する人材育成の指針として、活用して頂きたい。

従来から、情報システム開発の最上流工程である要求分析は、システム開発の成否を握る鍵として、最も重要な工程として認識されてきました。しかし、要求分析はこれまで個人のスキルによるところが大きく、要求工学を現場で組織的に活用することが重要かつ緊急の課題です。本報告書は、現場の視点から要求工学の知識を整理体系化し、要求分析者のみならず、ユーザ、プロジェクトマネジャー、経営者、など、ユーザとベンダが共通に持つべき知識を明らかにし、要求開発に関与するすべてのステークホルダの指針とするものです。

本報告書は、平成 18 年度の調査研究報告書「要求開発ベストプラクティスが示す成功パターンの調査研究」、平成 19 年度の「要求開発・管理ベストプラクティスとその体系化の調査研究」に続くものです。本調査研究では、要求開発、管理に関わる知識全体の包括的な枠組みを規定し、これまでの調査研究の成果をその中に組込んでいます。

要求工学に関係する幾つかの知識体系が提示されています。REBOK の意義は、これらの知識体系を包括し、なおかつ、専門家のみならず要求開発の関与するすべてのステークホルダの知識と水準を整理した点にあります。従って、エンタープライズシステム、組込みシステムを問わず、広範な現場で利用可能となっています。さらに、関連する知識体系の位置づけを明確にしました。また、REBOK に基づく実践を支援する要求工学のノウハウとして、要求工学ベストプラクティスと実践事例集を収集し、まとめています。これによって、要求工学に関連する知識体系の理解、現場での適用、と人材育成の指針になります。本報告書が要求開発、管理の実践、ならびに、人材育成において活用されることを期待します。

本調査研究が JISA 会員企業各位の開発の指針の一助になれば幸いです。

最後に、本調査研究を実施するに当たり、ご指導、ご高配いただきました経済産業省に厚く御礼申し上げます。また、本報告書をまとめるにあたりご尽力いただいた委員、講師各位に深謝いたします。

平成 21 年 3 月

(社)情報サービス産業協会
技術委員会　ソフトウェアエンジニアリング部会　要求工学 WG
座長　青山 幹雄

平成 20 年度 ソフトウェアエンジニアリング研究会 要求工学 WG 委員名簿

委員長	青山 幹雄	南山大学 情報理工学部 ソフトウェア工学科 教授
委員	矢野 達也	アイエックス・ナレッジ(株) 品質管理部 PMO 担当部長
	菊島 靖弘	(株)アイネス 損保システム本部 フェロー
	尾木 巧永	伊藤忠テクノソリューションズ(株) ビジネス&システムコンサルティング本部本部長
	桶谷 貴弘	(株)インテック 技術部 主事
	渋谷 照夫	NECインフロンティア(株) 国内営業事業本部 / システム開発事業部シニアエキスパート
	加藤 久美子	NECソフト(株) 生産技術部リーダー
	小池 輝明	NECネクサソリューションズ(株) 技術開発事業部 マネージャー
	斎藤 忍	(株)エヌ・ティ・ティ・データ 技術開発本部 ソフトウェア工学推進センタシニアエキスパート
	後藤 卓史	(株)構造計画研究所 ソフト工学センター
	田中 等	(株)シーエーシー ビジネス支援本部 アプリケーション・アーキテクトグループ
	安田 哲平	(株)CSKシステムズ プロジェクト管理部 品質技術課 主事
	小浜 耕己	スミセイ情報システム(株) PMO 部 統括マネージャ
	千秋 徹	(株)大和総研 人事部人材開発・採用課 次長
	鈴木 三紀夫	TIS(株) 基盤技術センター エキスパート
	杉岡 克也	ティーディーシーソフトウェアエンジニアリング(株) 社会情報システム本部 第1システム統括部 第2システム部 担当課長
	大石 貴之	(株)DTS ビジネス開発本部 IT コンサルティング部
	村田 尚彦	東芝ソリューション(株) IT 技術研究所 研究開発部 IT 品質ラボラトリー 主任研究員

委員	新保　康夫	日本コンピューター・システム(株) 管理本部　IT 調査研究室　室長
	森永　善宏	日本電子計算(株) 生産管理本部　生産技術部
	大久保　明彦	日本電子計算(株) 生産管理本部　生産技術部
	荒生　知之	(株)野村総合研究所 システムデザインコンサルティング部　上級システムコンサルタント
	斉藤　直樹	(株)野村総合研究所 事業統括本部　事業推進部長
	村山　武士	日立公共システムエンジニアリング(株) ソリューション第4事業部　研究・開発センタ　センタ長
	元村　憲一	(株)福音館書店 総務部　電算課
	大里　章	三菱スペース・ソフトウエア(株) 技術推進部・次長
	眞木　康裕	三菱総研DCS(株) 金融カード本部　金融カード営業部システム開発室
	袴田　典之	(株)菱友システムズ ビジネスソリューション事業部　事業推進室PMOグループ
オブザーバ	塚本　英昭	(独)情報処理推進機構 ソフトウェア・エンジニアリング・センター　企画グループ
	小林　健児	(独)情報処理推進機構 ソフトウェア・エンジニアリング・センターエンタプライズ系プロジェクト
	中谷　多哉子	筑波大学大学院 ビジネス科学研究科　准教授　博士(学術)
	白銀　純子	東京女子大学 現代文化学部　准教授
	鎌田　真由美	日本アイ・ビー・エム(株) 東京基礎研究所
	堀　昭三	安川情報システム(株) 品質保証本部　品質保証部　部長
事務局	鈴木　律郎	(社)情報サービス産業協会 企画調査部　技術課　課長
事務局	大原　道雄	(社)情報サービス産業協会 企画調査部　技術課

要求工学知識体系(REBOK)とユーザ指向要求工学の調査研究・目次

概要
委員名簿
はじめに
1. 要求開発を担う人材のための要求校が知識体系(REBOK)とベストプラクティス………… 3
1. 1. はじめに……………………………………………………………… 3
1. 2. 要求工学に関連する知識体系……………………………………………… 3
1. 3. 要求工学に関連する知識体系の課題と知識体系化への要求……………………… 4
1. 4. 要求工学知識体系の枠組み………………………………………………… 7
1. 5. 要求開発のステークホルダに応じた知識水準の設定…………………………… 8
1. 6. ドメイン特性とドメイン知識の扱い………………………………………… 9
1. 7. 要求工学知識要素間の関係………………………………………………… 10
1. 8. 実践の考慮点：要求工学ベストプラクティスと事例集…………………………… 11
1. 9. 要求工学に関連する知識体系との比較……………………………………… 12
1. 10. 今後の課題…………………………………………………………… 13
1. 11. まとめ……………………………………………………………… 13
付録 1. 要求工学知識体系の知識要素…………………………………………… 15
付録 2. SWEBOK の要求工学知識体系とその構造………………………………… 18
付録 3. BABOK・Version2.0 の知識体系とその構造……………………………… 18
付録 4. 要求工学の主要概念……………………………………………………… 19
付録 5. 要求工学ベストプラクティス……………………………………………… 22
2. ユーザー企業における共通フレーム 2007 の実践的活用～要件定義を中心としたシステム構築の改善～……………………………………………………………………27
2. 1. シミズの IT 戦略の全体像………………………………………………… 27
2. 2. システムの変遷……………………………………………………… 28
2. 3. 要件定義の見直し……………………………………………………… 29
2. 4. 超上流を中心にした標準の見直し…………………………………………… 31
2. 5. システム開発標準の見直しのポイント……………………………………… 34
2. 6. 要件定義行程の作業関連………………………………………………… 37
2. 7. プロジェクトレビュー標準の見直し………………………………………… 38
2. 8. その他の新たに規定した標準……………………………………………… 39
2. 9. 最後に……………………………………………………………… 40
3. 東京海上日動の抜本改革システム開発における要件定義の進め方……………………… 45
3. 1. はじめに……………………………………………………………… 45
3. 2. 東京海上日動システムズ株式会社の組織体制………………………………… 45

3. 3. 抜本改革プロジェクトの概要 46
3. 4. アプリオーナー制度 48
3. 5. 抜本改革システム開発における要件定義の進め方 50
3. 5. 1. 基本要件定義工程 51
3. 5. 2. 詳細要件検討工程 52
3. 5. 3. プログラム開発工程 53
3. 5. 4. システムテスト工程 54
3. 5. 5. 稼働後の評価の工程 56
4. 知恵と勇気の要件獲得？プロセス 61
4. 1. はじめに 61
4. 2. 要件とは何か 61
4. 2. 1. 犯罪の構成要件とは 61
4. 2. 2. 犯罪の構成要件を情報システムに当てはめると 62
4. 3. 要件定義はなぜ難しいのか 63
4. 3. 1. 意志が見えない 63
4. 3. 2. 知識・経験の不足 63
4. 3. 3. 錯綜する意図 63
4. 3. 4. 要件定義とは 64
4. 4. 要件定義に必要なもの 64
4. 4. 1. 道具の例：WBS と DFD 64
4. 4. 2. 論理モデルによる抽象化 66
4. 5. 顧客との共有・合意形成 67
4. 5. 1. 建設的な役割分担 68
4. 5. 2. 進むべき道は？ 69
4. 5. 3. ユーザー参画に向けてできること 69
4. 5. 4. スミセイ情報システムの方針 70
5. 日本における IT ガバナンスの取り組み 73
5. 1. はじめに 73
5. 2. 民間(損害保険会社)の IT ガバナンスの歴史 74
5. 3. 政府(経済産業省)IT ガバナンス政策事例 75
5. 4. ユーザー（一般事業会社）における IT ガバナンス 76
5. 5. 終わりに 84
6. 要件定義フェーズを取り巻くコンテキスト分析 87
6. 1. コミュニケーションとコンテキスト 87
6. 1. 1. ソフトウェア開発におけるコミュニケーション 87
6. 1. 2. コミュニケーションモデル 87

6. 1. 3. 2 つのコミュニケーションモデル・・ 88
6. 1. 4. コミュニケーションを支えるコンテキスト・・ 89
6. 1. 5. 意味ネットワーク・・ 90
6. 2. コンテキスト可視化の実験・・・ 91
6. 2. 1. コンテキスト発見ワークショップ・・・ 91
6. 2. 2. 役割の違いによるコミュニケーションの問題・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 92
6. 2. 3. ワークショップ実施結果について・・ 93
6. 3. コンテキストの可視化・・・ 93
6. 3. 1. ソフトウェア技術者の意味ネットワーク・・ 93
6. 3. 2. 開発者の意味ネットワーク・・ 93
6. 3. 3. 品質担当者の意味ネットワーク・・・ 94
6. 3. 4. テスト担当者の意味ネットワーク・・・ 95
6. 4. 要件定義フェーズを取り巻くコンテキスト分析・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 96
6. 4. 1. 要件定義フェーズを取り巻く意味ネットワーク・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 96
6. 4. 2.「要件定義フェーズ」内の連想単語分析・・・ 97
6. 4. 3. 経験年数別にみる「要件定義フェーズ」内の連想単語・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 98
6. 4. 4. コンテキストの視点から要件定義フェーズへの提言・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 99
6. 5. コンテキストの発見を「気づき」に・・ 100
7. 経営と IT の連携に向けた可視化の取り組みについて・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 105
7. 1. 背景、目的・・・ 105
7. 2. ビジネスプランのメタモデル・・・ 106
7. 3. 要求分析技法・・・ 109
7. 4. ケーススタディ・・・ 110
7. 5. まとめと今後の課題・・ 112

1. 要求開発を担う人材のための要求工学知識体系(REBOK)とベストプラクティス

南山大学　青山 幹雄

1. 1. はじめに

本報告書は、要求工学(要求の開発、管理)を現場で実践するために必要な知識を体系化した要求工学知識体系 REBOK(Requirements Engineering Body Of Knowledge)[リーボックと呼ぶ]の枠組みをまとめたものである。

情報サービス産業協会(JISA)の要求工学調査研究委員会では、2006 年度から要求工学の実践を推進するための調査研究を行ってきた。

2006 年度は、要求工学の組織的な実践のための組織体制とその事例について調査を行い、これまでのベンダ主体の要求分析から、要求分析をより円滑に進めるための組織として、ユーザとベンダの仲介役を果たす要求アナリストの役割モデルを提案した[7]。

2007 年度は、要求工学の実践ノウハウの収集と整理体系化を行い、ベストプラクティスとしてまとめた。114 件のプラクティスを収集し、その中から、35 件をベストプラクティスとして整理した。あわせて、実践事例を収集した[8]。

2008 年度は、これまでの成果を包括するために、要求工学の知識を体系化し、要求開発、管理を実践する人材育成の指針となる要求工学知識体系(REBOK)をまとめた。あわせて、ユーザの視点からの要求工学の実践と要求工学の人材育成の実践事例を調査した。

1. 2. 要求工学に関連する知識体系

要求工学を理解し、実践するための知識体系(REBOK)に関連する知識体系、人材育成モデルとして、これまで、次のようなものが提唱されている。

(1) ソフトウェア工学知識体系(SWEBOK: Software Engineering Body of Knowledge)

米国 IEEE Computer Society が策定したソフトウェア工学全般の知識体系である[1]。この中に、要求工学の知識体系も含まれている。現在は、2004 年に策定された SWEBOK 2004 である。ソフトウェア開発の専門職としての CSDP(Certified Software Development Professional)と、その前段階としての CSDA(Certified Software Development Associate)の２つのレベルの認定試験を実施している。

(2) ビジネス分析知識体系(BABOK: Business Analysis Body Of Knowledge)

カナダのトロントに 2003 年に設立された IIBA(International Institute of Business Analysis)が策定した、高度な能力を持つビジネスアナリストのための知識体系である。2009 年に大幅に改訂され、第 2 版として公開された[2]。BABOK に基づき、2006 年から、CBAP(Certified Business Analysis Professional)と呼ぶ認定試験が実施されている[3, 9]。IIBA は 2008 年 12 月に日本支部が設立された。

(3) IT スキル標準(ITSS)、 情報システムユーザスキル標準(UTSS)、 組込みスキル標準(ETSS)

経済産業省によるソフトウェア技術者のスキル標準として、情報処理推進機構(IPA)の IT スキル標準センターにより推進されているスキル標準群である[5]。

(4) 要求工学プロフェッショナルのための教育プログラム

要求工学の研究者が中心となって、2006 年にドイツで設立された IREB (International Requirements Engineering Board)が要求工学の普及のために初心者レベルの CPRE(Certified Professional for Requirements Engineering)と呼ぶ認定試験を実施している。この認定試験の受験者のための教育カリキュラムを公開している[4]。

これらの知識体系の特徴を表 1-1 にまとめて示す。

表 1-1 要求工学に関連する知識体系の特徴

知識体系	認定	対象者	内容	認定レベルと前提条件
SWEBOK	CSDP, CSDA	ソフトウェア開発者	ソフトウェア工学の中のソフトウェア要求工学	なし
BABOK	CBAP	ビジネス分析者	ビジネス分析	高度専門家(7,500 時間(5 年)以上のビジネス分析実務経験など)
ITSS	－	コンサルタント, IT アーキテクト	情報技術とスキル	中～高度技術者(スキルレベルを設定)
IREB カリキュラム	CPRE	要求分析者	要求工学	初心者

1. 3. 要求工学に関連する知識体系の課題と知識体系化への要求

要求工学とは、情報システムに対する要求定義を組織的、体系的に行うための技術体系である[10, 11, 12]。

我が国のソフトウェア開発の現場では、要求定義が最大の課題となっている[6]。図 1.1 に示す調査は JISA 会員企業を対象としたものであるが、要求定義は、ユーザとベンダの共通の課題である。

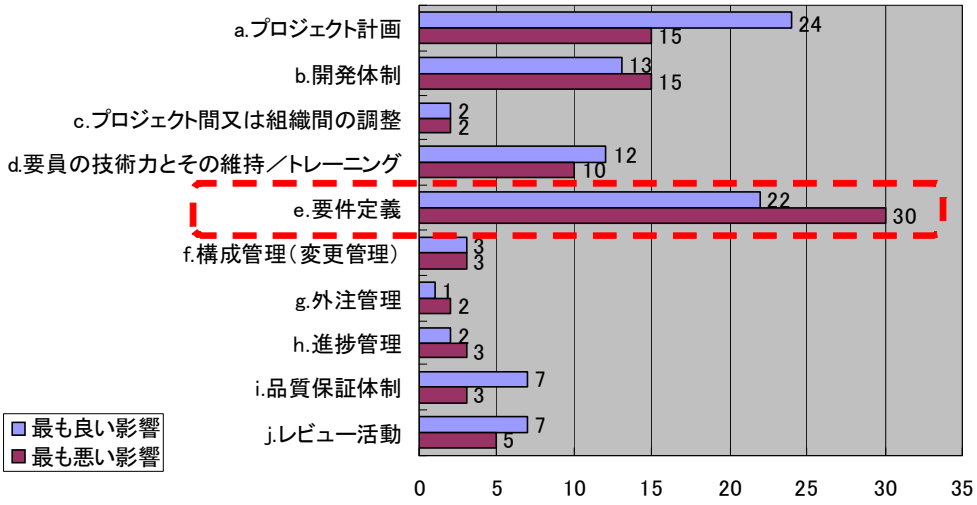


図 1.1 情報システム開発における課題

要求開発の対象は、図 1-2 に示すように、次のような、ビジネス/製品、情報システム、ソフトウェアの３層のスコープで捉えることができる。

(1) **ビジネス要求(製品要求)**：ビジネスゴールを達成するために、業務の現状と達成すべき姿や製品を定義する。業務に従事する人やユーザを含めて、ビジネスルールやコスト、資産などの制約の下で、ビジネスプロセスやその実行形態を定義する。一方、組込みシステムやパッケージソフトウェアでは、製品要求として捉えられる。

(2) **システム要求**：ビジネス要求を満たすために情報システムのハードウェアとソフトウェアが果たすべき要求を定義する。

(3) **ソフトウェア要求**：情報システムへの要求の中でソフトウェアに対する要求を定義する。

ここで、ビジネス要求のアナリストをビジネスアナリストと呼ぶ。同様に、情報システム、ソフトウェアシステムのアナリストを、それぞれ、システムアナリスト、ソフトウェアアナリストと呼ぶこととする。これらのアナリストを総称して、要求アナリストと呼ぶことにする。

ただし、これらの名称はスコープに対応した知識範囲を役割として表したものであり、同一人が複数のスコープを扱うこともあるので，複数の役割を担うこともある。また、組込みシステムでは、ビジネス要求に相当するものは、製品(Product)要求と考えられるが、特に、断らない限り、本報告書では、ビジネス要求に製品要求の意味も含めるものとする。

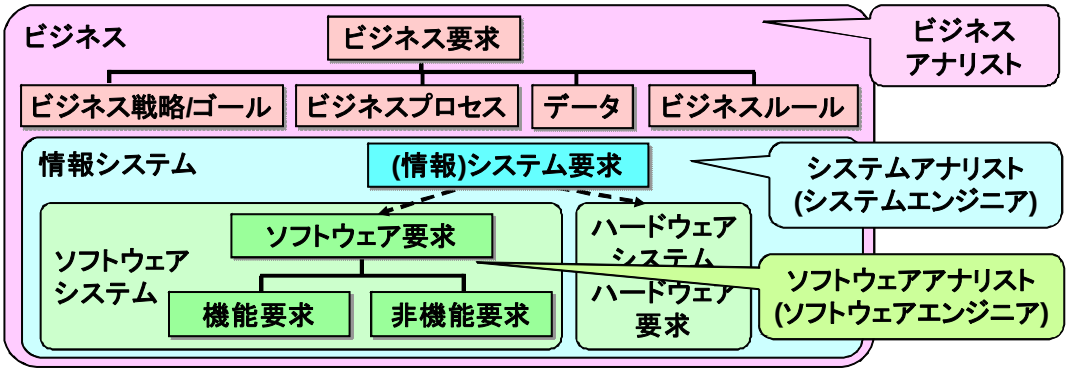


図 1.2 要求開発のスコープと要求アナリストの役割

要求開発は、対象のスコープに応じて、エンドユーザやユーザの経営者など、多様なステークホルダが関与することに特徴がある。そのため、分析を適切に行うためには、要求アナリストがエンドユーザやユーザの経営者などのステークホルダと協調する必要がある。このことから、要求アナリストだけでなく、エンドユーザや経営者も要求工学の基礎知識を持つことや理解できることが望まれる。

このような観点から見ると、表 1.1 に示した、現在提示されている要求工学に関連する知識体系は高度な専門化を対象とする内容や初心者に限定された内容となっている。このため、要求開発で広く実践する上では制約や課題がある。また、ビジネスシステムに限定されているなど、組込みシステムの要求分析へ適用する上で制限がある。

以上のような観点から、要求工学知識体系(REBOK)に対する要求として、次の 4 項目を挙げることができる。

要求工学知識体系(REBOK)への要求

(1) ベンダ、ユーザを問わない共通の知識体系
(2) 要求アナリストに加えて、エンドユーザ、経営者など、要求開発に関与するステークホルダが、必要に応じて習得すべき範囲と水準を整理した知識体系
(3) ビジネス要求、システム要求、ソフトウェア要求の 3 つのスコープに応じた知識体系。
(4) エンタープライズシステム、組込みシステムの要求開発に共通する技術の知識体系。ただし、ドメイン固有の知識は個別に定義されるものとする。

1.4 要求工学知識体系の枠組み

前節で示した要求工学知識体系(REBOK)が満たすべき要求に基づき、要求工学知識体系の枠組みを図 1.3 に示す。この知識体系は、ソフトウェア工学知識体系(SWEBOK)の中の要求工学知識体系に基づき、ビジネス要求とシステム要求のスコープへ対応できるように拡張した。図 1.3 に示すように、REBOK は 8 つの主要な知識要素から成る。

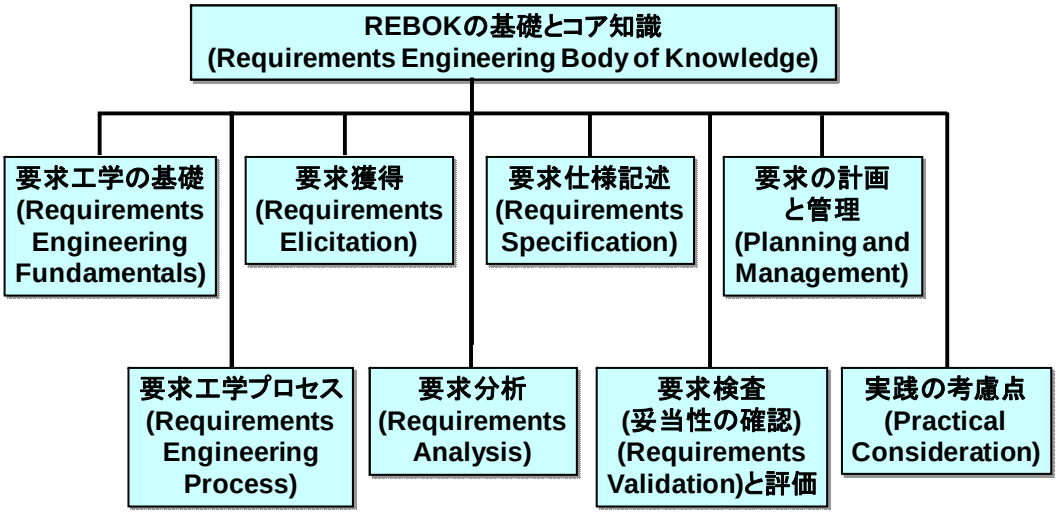


図 1.3 要求工学知識体系(REBOK)の枠組み

表 1.2 要求工学知識体系の知識分類(主要要素)

知識分類	内　容
要求工学の基礎	要求とそのスコープや性質などの基礎的事項。機能要求、非機能要求も含む。
要求工学プロセス	要求開発、管理のプロセスと主要なアクティビティなどに関する知識。
要求獲得	顧客を含むステークホルダを明らかにし、会議やインタビューなどを通して要求を引出す技術に関する知識。
要求分析	要求項目を整理し、その間の関係を付け、さらに、優先順位付けなどを行い、実現すべき要求を明らかにして絞り込む、などに関する知識。
要求仕様化	分析された要求を規定の書式や表記法で記述する技術に関する知識。
要求検査(妥当性の確認)と評価	要求の間の矛盾がないことや、必要な顧客の要求項目を満たしていることの確認、あるいは、その達成の度合いを評価する技術などに関する知識。
要求の計画と管理	要求開発を計画し、遂行や成果物の管理を行う技術に関する知識。
実践の考慮点	要求工学を実践する上で知っておくべき知識やノウハウ。

ここで、要求工学知識体系(REBOK)の基礎としてソフトウェア工学知識体系(SWEBOK)を採用したのは、次の理由による。

(1) IEEE Computer Society による、ソフトウェア工学の広範で標準知識体系を基礎とし、技術的根拠や標準としての独立性が期待できる

(2) 現在の知識体系の内容は基礎的かつ一般的であり、高度な専門性を前提としていないことから、本知識体系が目指している、初心者から専門家までの異なる知識水準を設定可能である。

1.5 要求開発のステークホルダに応じた知識水準の設定

要求工学が広く活用されるためには、要求開発、管理に関与する多様なステークホルダが要求工学を一定の水準で理解し、実践を推進することが望ましい。REBOK では、要求工学に関与するステークホルダを次の 3 つの役割モデルで定義する。なお、図 1.4 に示すように、以下の役割モデルは、ユーザ、ベンダによらないことに注意して頂きたい。

(1) 要求アナリスト(Requirements Analyst)

要求開発、管理の実行者。対象のスコープに応じて、ビジネスアナリスト、システムアナリスト、ソフトウェアアナリストに細分化できる。また、アナリストの専門性の高さとして要求される知識水準は、初級、中級、上級の 3 つのレベルを想定する。

(2) 要求の利用者(Requirements User)

要求アナリストが要求を獲得するための協力者となりえるエンドユーザや要求開発の成果物を理解し、利用するユーザ、ソフトウェア開発者、プロジェクトマネジャーなどがある。要求開発の基礎的な知識を持つことを想定する。

(3) 要求工学の理解者(Requirements Engineering Supporter)

要求工学をユーザやベンダ内で、組織的の取り組むことを推進する、CIO や経営者など、要求工学の意義を理解することを想定する。

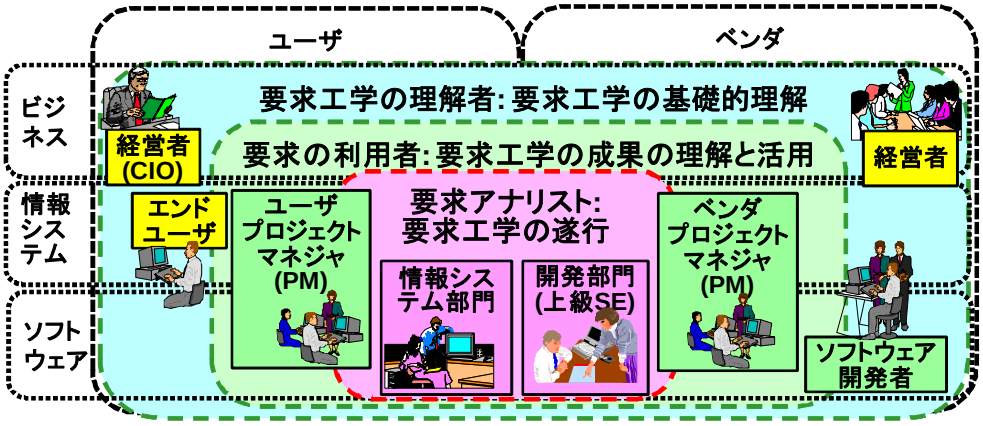


図 1.4 要求開発の 3 つの役割と期待される知識水準

表 1.2 要求開発ステークホルダと期待される知識要素の水準

知識要素　　レベル	要求工学の理解者	要求の利用者		要求アナリス		
	1	2	3	3	4	5
要求工学の基礎	1	2	3	3	4	5
要求工学プロセス	1	2	3	3	4	5
要求獲得	－	2	3	3	4	5
要求分析	－	2	3	3	4	5
要求仕様化	－	2	3	3	4	5
要求検査と評価	－	2	3	3	4	5
要求の計画と管理	1	2	3	3	4	5

ここで、知識要素水準の意味は、次のように定義する。

1: 意味を知っている

2: 内容を理解している

3:内容を他者に説明できる、利用できる

4: それを用いて、分析などを行える

5: 複雑な問題に対して適切な方法の選択と遂行、あるいは、指導ができる

1.6 ドメイン特性とドメイン知識の扱い

要求工学の実践にあたっては、対象ドメインの特性に依存し、ドメイン知識が不可欠である。しかし、ドメインの知識をすべて体系化することは事実上不可能である。一方、特定のドメイン、例えば、ビジネスアプリケーションのみを対象とすることは、適用範囲に大きな制約を加えることになる。

このため、REBOK では、ドメインの知識を、図 1.5 に示すように、次のように扱うこととする。

(1) ドメイン知識は、REBOK の知識とは独立に整理、蓄積されるものとする。

(2) REBOK は特定ドメインのみに限定しない。これは、例えば、BABOK のようにエンタープライズ(ビジネス)システムに限定することによる適用範囲の制約を加えないことにする。

(3) ドメインに強く影響される要求獲得では、エンタープライズシステムなどのプロセスを中心とする要求獲得と組込みシステムなどのプロダクト(製品)を中心とする要求獲得がある。これら両方の基本的な要求獲得の知識を使い分けることができることとする。

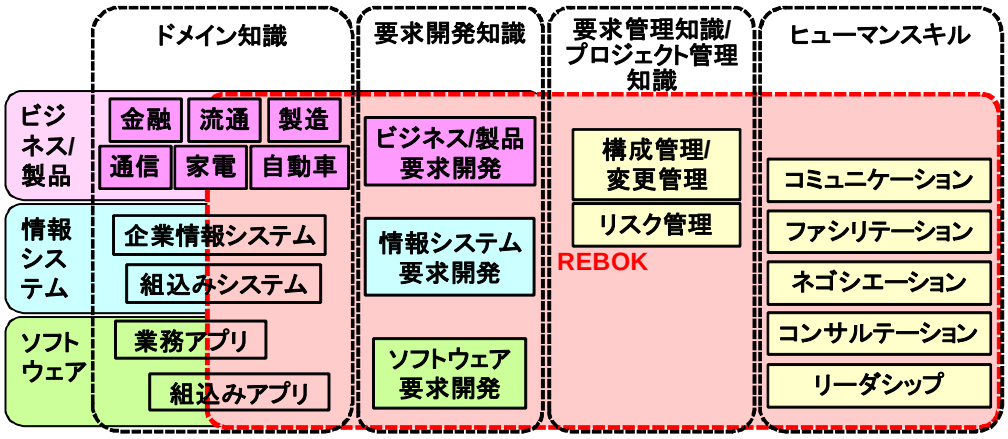


図 1.5 ドメイン知識の扱い

1.7 要求工学知識要素間の関係

REBOK の知識要素の間の関係を図 1.6 に示す。

基礎知識として、要求工学の基礎と要求プロセスを位置付けている。要求プロセスは、要求開発、管理の全体の枠組みを規定する。要求開発の知識である、要求獲得、要求分析、要求仕様化、検査(妥当性の確認)と評価は要求プロセスに基づいて実行され、相互に関係する。要求の計画と管理は、要求開発の 4 つの知識を適用した要求開発を管理する。

実践知識として位置付けている実践の考慮点は、要求開発、管理を実践する上で知っておくべき知識やノウハウである。実践経験から抽出した事例，ノウハウをベストプラクティスとして組み入れる。

要求開発、管理ではコミュニケーションなどのヒューマンスキルも重要である。ヒューマンスキルの扱いについては、今後の課題である。

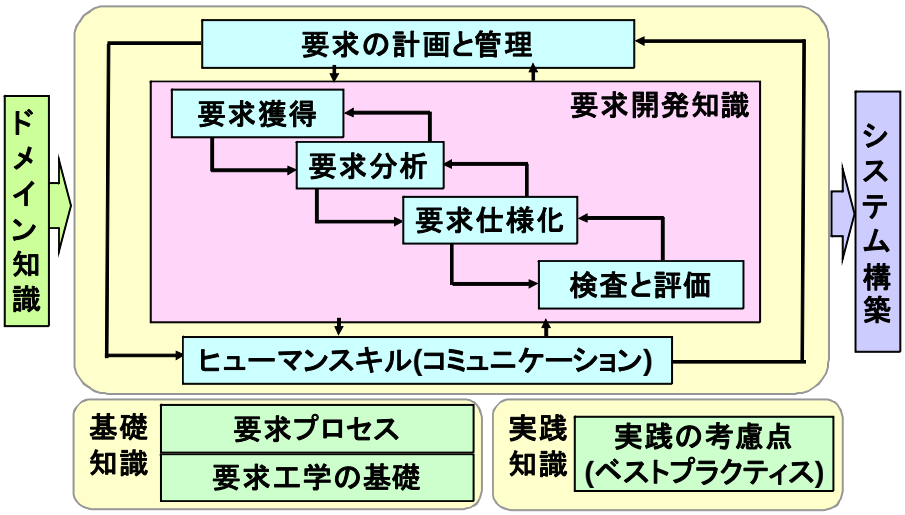


図 1.6 知識要素間の関係

1.8 実践の考慮点: 要求工学ベストプラクティスと事例集

要求工学の実践においては、ユーザの業務やそれを取巻く環境などが多様であることから、要求工学知識体系を環境に応じて適切に適用したり、カスタマイズする必要がある。従って、要求工学を実践するためには、このような業務や環境に応じて、適切な技術の選択や調整が必要である。このため、実践における様々な考慮点を共有することが望ましい。

現場における要求工学の実践のノウハウの中から適切と思われるノウハウをプラクティスとして収集し、レビュー後、共通の形式で文書化したものを「要求工学ベストプラクティス」としてまとめている。本調査委員会で収集した、要求工学プラクティスとベストプラクティスを表 1.3 に示す。

さらに、これらのベストプラクティスを含む要求工学の事例をまとめて事例集を作成している。このような、実践に基づくベストプラクティスや事例を収集している点が REBOK の活用を支援する。

表 1. 3　プラクティスとベストプラクティス

プロセス	問題分類	プラクティス	ベストプラクティス
要求獲得	要求獲得	7	3
	ステークホルダ分析	9	3
	業務知識	6	2
	真の要求(ユーザとベンダの相互理解)	7	0
	目的	6	3
	要求発散	18	6
	非機能要求	2	1
	ファシリテータ	1	1
	小計	56	19
要求分析	分析・評価	9	9
	整合性・矛盾の解消	7	3
	実現可能性	1	0
	方法論	1	0
	小計	18	12
仕様記述	仕様記述	9	2
	文書化	3	1
	文書・用語	3	0
	小計	15	3
要求検査	検証・レビュー	7	1

要求管理	プロセス	1	0
	プロジェクト管理	2	0
	要求定義の長期化	2	0
	要求変更	2	0
	スキル・人材	5	0
	ユーザの人事異動	3	0
	取引慣行	2	0
	経営層の支援	1	0
	小計	18	0
合 計		114	35

1.9 要求工学に関連する知識体系との比較

要求工学知識体系 REBOK と関連する知識体系を知識要素毎に比較したものを表 1.3 に示す。この表から、各知識体系は、他と比較して、次のような特徴があることが分かる。

(1) REBOK

知識要素の広がりと構造化の点で、バランスが良いといえる。また、前述したように、要求開発に関与するステークホルダの役割モデルと持つべき水準に応じた知識獲得を促すことができる。

(2) SWEBOK

ソフトウェア工学の知識体系として幅広い枠組みを提供するが、ソフトウェアに限定されるため、ビジネス分析、システム分析の明示的な枠組みや知識水準の定義がない。

(3) BABOK

ビジネス分析の高度なアナリストがもつべき知識体系として豊富な内容を提供するが、対象がビジネス分析に限定される. さらに、高度なアナリストに限定さえることで、対象者が限定される。また、BABOK は、REBOK において、エンタープライズドメインのビジネス分析に対象に限定し、かつ、レベル 4～5 の水準のビジネスアナリストの知識体系に相当していると考えられる。

(4) CPRE

初心者向けの簡易なカリキュラムに留まり、知識体系として整理されていない。

表 1.3 REBOK と関連知識体系との比較

分類	知識要素	REBOK	SWEBOK	BABOK Ver. 2.0	CPRE
技術	基礎	○	○	○	○
	プロセス	○	○	×	△
	要求獲得	○	○	○	○
	要求分析	○	○	○	○
	要求仕様化	○	○	×(分析に含まれる)	○
	検証・評価	○	○	○(アセスメントと検査)	△
	ツール	×	×	×	○
ドメイン	エンタープライズ	個別知識とする	×	エンタープライズ分析	×
	組込み	個別知識とする	×	×(対象外)	×
要求管理		○(計画と管理)	×	ビジネス分析, 計画とモニタリング	○
コミュニケーション		△(今後検討)	×	要求管理とコミュニケーション	×
実践の考慮点		○	△(管理)	×	×
知識水準のレベル		役割と水準(1~5)	×	高度専門家	初心者

1.10 今後の課題

本章で報告した要求工学知識体系(REBOK)は、枠組みを議論した段階である。今後、知識要素の内容と必要とされる水準を明確にする必要がある。

あわせて、REBOK に基づく人材育成方法についても検討を行う。

1.11 まとめ

要求開発はユーザ、ベンダを問わず、情報システム、組込みシステム開発における最大の課題である。要求開発を組織的に行うための技術体系である要求工学はユーザ、ベンダ双方に有用な技術体系を提供する。要求工学が広く実践されるためには、要求分析の専門家の育成はもとより、エンドユーザ、経営者など要求開発に関与するすべてのステークホルダが要求工学に一定の理解をもち、かつ、活用できる必要がある。

このため、本調査研究委員会では、多数のステークホルダが活用できる要求工学知識体系として、 REBOK[リーボック]の枠組みを開発した。さらに、REBOK に基づく実践を支援する要求工学のノウハウとして、要求工学ベストプラクティスと実践事例集を収集し、まとめた。

本報告書の成果が、要求開発、管理の実践、ならびに、人材育成、実践の推進において多くの方々に活用されることを期待する。

なお、要求工学知識体系は引き続き検討を行う予定である。

参考文献

[1] A. Abran, et al. (eds.), Guide to the Software Engineering Body of Knowledge, 2004 Version, IEEE Computer Society, http://www.swebok.org/.

[2] IIBA, A Guide to the Business Analysis Body of Knowledge (BABOK Guide), Version 2.0, 2009.

[3] IIBA 日本支部, CBAP ハンドブック, http://www.iiba-japan.org/.

[4] IREB, Syllabus: IREB Certified Professional for Requirements Engineering Foundation Level, Version 1.3, Jan 2008, http://certified-re.de/en/.

[5] IPA IT スキル標準センター, IT スキル標準 V3 2008, http://www.ipa.go.jp/jinzai/itss/download_V3_2008.html.

[6] 情報サービス産業協会, 平成 17 年度情報サービス産業におけるソフトウェア開発の実態アンケート調査結果(概要), 2007.

[7] 情報サービス産業協会 要求工学調査委員会, 要求開発ベストプラクティスが示す成功パターンの調査研究, 情報サービス産業協会, No. 18-J008, 2007 年 3 月.

[8] 情報サービス産業協会 要求工学調査委員会, 要求開発・管理ベストプラクティスとその体系化の調査研究, No. 19-J004, 2008 年 3 月.

[9] J. Phillips, CBAP Certified Business Analysis Professional All-in-One Exam Guide, McGraw Hill, 2009.

[10] K.E.Wiegers (渡部 洋子(訳)), ソフトウェア要求：顧客が望むシステムとは, 日経 BP ソフトプレス, 2003.

[11] 山本 修一郎, 要求を可視化するための要求定義・要求仕様書の作り方, ソフト・リサーチ・センター, 2006.

[12] R.R.Young, The Requirements Engineering Handbook, Artech House, 2003.

付録 1: 要求工学知識体系の知識要素

知識分類	知識要素	内 容
要求の基礎	要求の定義(Definition of Requirements)	要求とは実世界の課題を解決するために明らかにすべき特性(Property)
要求の基礎	プロダクトとプロセスの要求(Product and Process Requirements)	プロダクト要求は開発すべきソフトウェアへの要求, プロセス要求は開発の制約条件.
要求の基礎	機能要求と非機能要求(Functional and Non-Functional Requirements)	機能要求はシステムやソフトウェアが提供すべき機能. 非機能要求は機能要求の制約, あるいは, 品質要求.
要求の基礎	創発的特性(Emergent Properties)	システムやソフトウェアの要素間の相互作用によって発現する特性に関する要求
要求の基礎	定量的要求(Quantifiable Requirements)	定量的な定義が必要な要求. スループットや障害発生確率など.
要求の基礎	ビジネス要求, システム要求, ソフトウェア要求(Business Requirements, System Requirements and Software Requirements)	ビジネス要求は, 対象のビジネスが達成すべき要求. システム要求はハードウェアとソフトウェアを含むシステム全体の要求. ソフトウェア要求はソフトウェアのみに対する要求.

知識分類	知識要素	内 容
要求工学プロセス	プロセスモデル(Process Models)	ソフトウェア要求の獲得から検査, 変更へ至るプロセス
要求工学プロセス	プロセスアクタ(Process Actors)	要求工学プロセスの各アクティビティに参画する人
要求工学プロセス	プロセスの支援と管理(Process Support and Management)	要求工学プロセスを実行するために必要な資源とその管理
要求工学プロセス	プロセスの品質と改善(Process Quality and Improvement)	要求工学プロセスの品質を評価し, 生産性や品質などを向上

知識分類	知識要素	内 容
要求獲得	要求源泉(Requirements Sources)	ゴール, ドメイン知識, ステークホルダ, 運用環境, 組織環境(ビジネス環境)
要求獲得	抽出技法(Elicitation Techniques)	インタビュー, シナリオ, プロトタイプ, ファシリテーション(ブレーンストーミング, など), 観察

知識分類	知識要素	内　容
要求分析	要求分類(Requirements Classification)	要求内容の分類(機能要求と非機能要求, など), 優先度, 有効範囲(Scope), 変更可能性(Volatility)
要求分析	概念モデル化 (Conceptual Modeling)	対象のモデル化の方法と表記法(UMLなど)
要求分析	アーキテクチャ設計と要求割当(Architecture Design and Requirements Allocation)	ソリューション(ソフトウェア)アーキテクチャの設計とコンポーネントへの要求を実現する責務の割当て
要求分析	要求調整(Requirements Negotiation)	要求間の競合などを解消し，ステークホルダのゴールを一定の水準で満たすよう要求を調整

知識分類	知識要素	内　容
要求仕様化	システム定義の文書化(System Definition Documents)	システム定義はユーザ要求とも呼ぶ. ユーザ要求は, 目的, 運用環境, 制約, 前提条件, 非機能要求を含む．要求定義は，概念モデル, システムを取巻くコンテキスト, 利用シナリオ, 主要ドメインエンティティ, データ/情報, ワークフロー, などを含む.
要求仕様化	システム要求の仕様化(Systems Requirements Specification)	ソフトウェアとソフトウェア以外のハードウェアなどのコンポーネントから成るシステムの要求を仕様化．ソフトウェア要求はシステム要求から導出.
要求仕様化	ソフトウェア要求の仕様化 (Software Requirements Specification)	ソフトウェア要求の仕様は，ソフトウェアが行うべきこと，と，行ってはならないことに関して，ユーザとベンダの間で合意するための基礎となる．また．開発のコスト，リスク，スケジュールを見積る基礎となる. 仕様の記述法: 自然言語と形式的，あるいは，半形式的表記法を併用．仕様記述の品質評価の尺度として，規模，分かりやすさ，深さ，文書構造などがある.

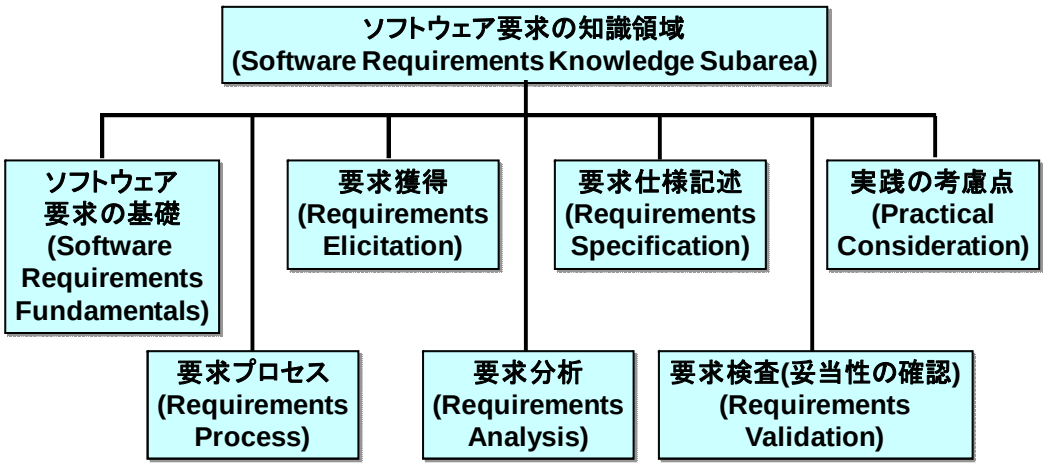
知識分類	知識要素	内　容
要求検査	要求レビュー (Requirements Reviews)	要求仕様をレビューし，誤り，前提条件の誤解，不明な点，標準やガイドラインからの逸脱などを発見する．チェックリストなどの実施の指針を準備．場合によっては，レビューヤにユーザ代表を含める．
要求検査	プロトタイプ (Prototyping)	ソフトウェア要求に対する開発者の解釈の検証や新しい要求の発見のために行う．適切なプロトタイプ開発の技術や利用方法．
要求検査	モデル検査(Model Validation)	要求分析で作成したモデルの正した検査する．静的解析や形式仕様の場合，形式的検証の適用．
要求検査	受入試験(Acceptance Tests)	開発したソフトウェアが機能要求，非機能要求を満たすか試験．試験計画の立案が重要．

知識分類	知識要素	内　容
要求の計画と管理	要求開発の繰返しプロセス (Iterative Nature of Requirements Process)	要求開発における繰返しや要求変更を行うプロセス技術．要求トレース，影響解析(Impact Analysis)，構成管理/影響管理など．
要求の計画と管理	変更管理(Change Management)	要求の変更管理．構成管理と関連．
要求の計画と管理	要求属性(Requirements Attributes)	要求を識別，管理するための情報．識別子の設定など．
要求の計画と管理	要求トレース (Requirements Trace)	要求は，前方と後方の両方にトレースできる必要がある．後方では，ステークホルダなど，要求の起源や動機．前方へは，要求を実現する要求や設計など．トレースは影響解析の基礎となる．
要求の計画と管理	要求測定(Measuring Requirements)	要求の規模などを測定する尺度と測定方法．ファンクションポイントなどの機能規模尺度(FSM: Functional Size Measurement)など．

知識分類	知識要素	内　容
実践の考慮点	ベストプラクティス(Best Practice)	要求工学の実践における成功のノウハウ．
実践の考慮点	事例集(Case Study)	要求工学の実践事例集．

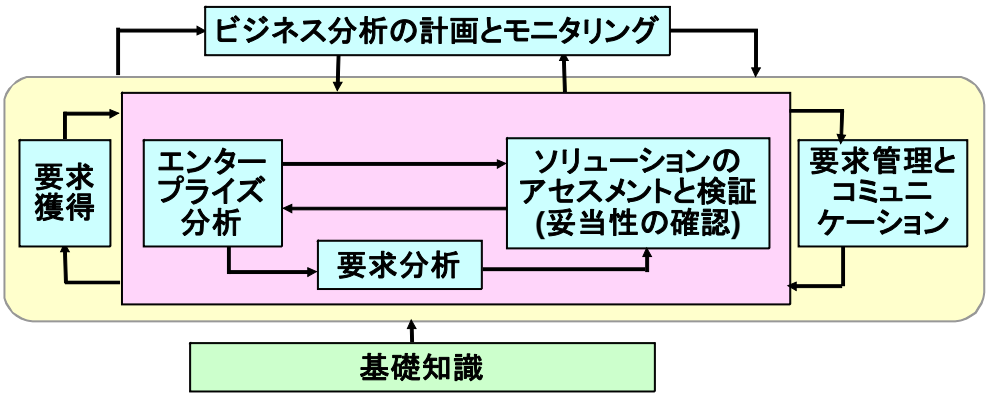
付録 2

SWEBOK の要求工学知識体系とその構造



付録 3

BABOK Version 2.0 の知識体系とその構造



付録 4
要求工学の主要概念

A4.1 要求開発プロセス

要求開発プロセスでは、図 A.1 に示す要求開発の 4 つのプロセスが段階的に実行される。注意すべきは、この 4 つのプロセスは、1 回実行すれば要求が確定するということではない。必要に応じて、これらのプロセスが繰返し実行される。

要求管理は、要求仕様書をはじめとする要求開発の成果物の変更管理、進捗管理、要求検査による品質確保など、要求開発プロセスの管理に関する諸活動を総称したものである。

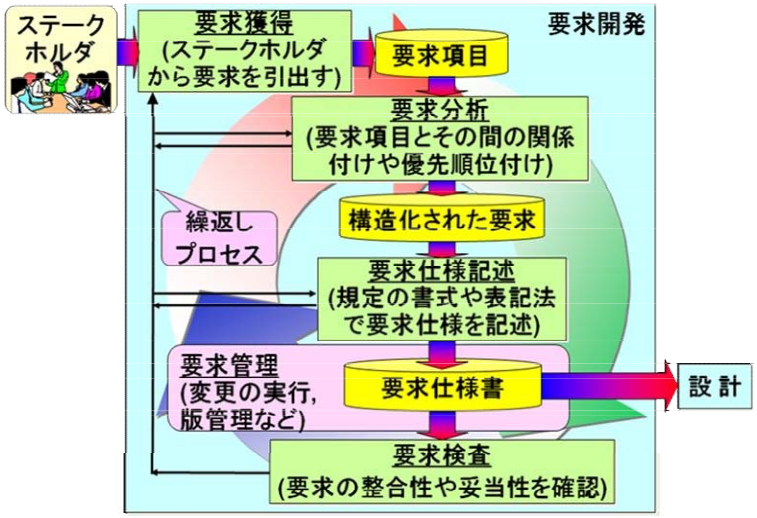


図 A. 1 要求開発プロセス

A4.2 要求獲得

要求獲得では、獲得する要求の源泉と獲得方法の 2 つが鍵となる。要求の源泉としては、次の要素を考慮する必要がある。

(1) **ビジネスゴール**：情報システムやソフトウェアがビジネスプロセスの効率化など達成すべきビジネス上の目標。
(2) **ステークホルダ**：要求に関与する人や組織。ユーザの組織における経営者、情報システム部門、エンドユーザ、ベンダの組織、あるいは、監督官庁などを含む。
(3) **ビジネスルール**：要求の前提となるビジネスルールであり、ビジネス上の制約、ユーザ内や業界内の諸標準など。
(4) **ドメインの知識**：ユーザの業務やユーザの業界に関する知識、製品に関する知識など、ユーザやその情報システムに関する知識。

(5) **実行環境**：情報システムやソフトウェアが実行するハードウェアやソフトウェアの環境。

要求獲得の方として、次のような方法が知られている。

(1) **インタビューとその関連技術**：エンドユーザや情報システム部門などステークホルダへのインタビューやブレインストーミング、アンケート調査を行う。

(2) **シナリオ(ユースケース含む)**：システムとは、ユーザやステークホルダとシステムとの関わりを時系列で記述したものである。ユースケースは、図的な表現で情報システムの機能とその関係を表し、機能毎にシナリオを用いて機能の内容を記述する方法である。

(3) **プロトタイプ**：ユーザインタフェースなど、人の感性に訴える要求に関しては、実際に使ってみないと要求を明確にすることが難しい場合がある。プロトタイプは、システムの一部を簡易に作成し、ユーザが試用して要求の抽出や確認を行う。

(4) **ファシリテーション**：ユーザとベンダの間に立って、第三者の立場から要求の抽出や分析などを仲介し、要求開発を円滑に行う組織的な方法である。

(5) **ユーザ観察**：ユーザの作業やシステムの利用方法を観察することにより、システムの問題点や要求を洗い出す方法である。

A4.3 要求分析

要求分析では、システムとソフトウェア要求の範囲を決定し、 要求間の関係を明らかにし、 矛盾や競合のない 妥当な要求を抽出する。

(1) **要求の分類**

(2) **概念モデリング**：要求の特性に応じたモデル化を行う。

(3) **アーキテクチャの検討と要求のサブシステムやコンポーネントへの割当**

要求を実現する情報システムやソフトウェアシステムは、必ずしも自由に実現選べるとは限らない。実行環境や抽出した要求とのバランスの中で、実現可能なアーキテクチャを検討し、要求仕様の実現可能性を検討する。また、必要に応じて、性能や信頼性などの非機能要求の実現可能性や、実現するための要求仕様上の課題を検討する。必要に応じて、サブシステムやコンポーネントの検討も行う。

(4) **要求のネゴシエーション**：要求仕様の実現可能性や製薬条件などをユーザと調整する。

A4.4 要求仕様記述

分析した要求を公式な要求仕様書としてまとめる。要求に対する観点によって、図 A.2 に示す、次のような表記法がある。

(1) **機能の観点に基づく仕様を記述**：データの流れに沿って機能を洗い出す：データフロー図(DFD: Data Flow Diagram)など。

(2) **データの視点に基づく仕様を記述**：データとその間の関係を整理する実体関連図(ER: Entity Relationship Diagram)など。

(3) **オブジェクト指向に基づく仕様記述**：機能、データ、ならびに、システムの動的特性に関する表記法などの複数の表記法を統合した UML が標準となっている。

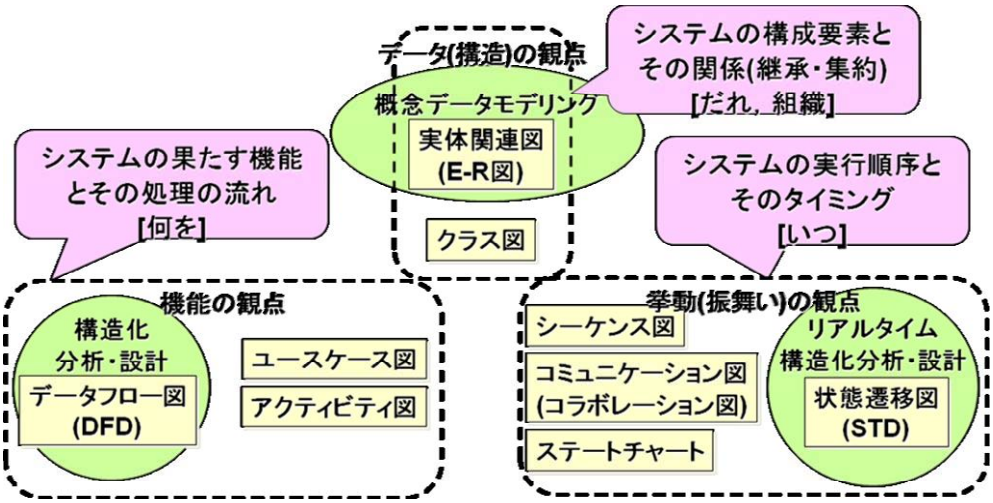


図 A.2　三つの視点と主な要求仕様記述の方法

A4.5 要求検査(妥当性の確認)

要求検査は、次の二つの側面がある

(1) 要求それ自体に矛盾がないことや冗長でないことなどの確認：レビューによる確認や、ツールを使った確認

(2) 顧客の要求と要求仕様とが一致することの確認：顧客とのレビューやプロトタイプによる確認、ネゴシエーションを通した確認など。

付録 5
要求工学ベストプラクティス

A5.1 要求工学ベストプラクティスの記述：ベストプラクティスパターン

ベストプラクティスとして統一的に理解し、参照できるようにするため、ベストプラクティスの記述形式を定めた。

この記述形式は、ソフトウェアの良い設計方法を記述する形式として広く利用されているデザインパターンの記述形式を参考に、要求工学のベストプラクティスの記述に適するようにした。これを、「要求工学ベストプラクティスパターン」と呼んでいる。

特に、要求開発、管理では、対象の業務や組織によって類似の問題であっても異なる解決方法があることから、図 A.3 に例示するように、問題と解決方法を対にして一つのパターンに記述している。さらに、問題の背景によって解決方法も異なる。問題はその背景と問題とを記述することとし、あわせて、できるだけ具体的事例を盛り込んで記述するようにした。

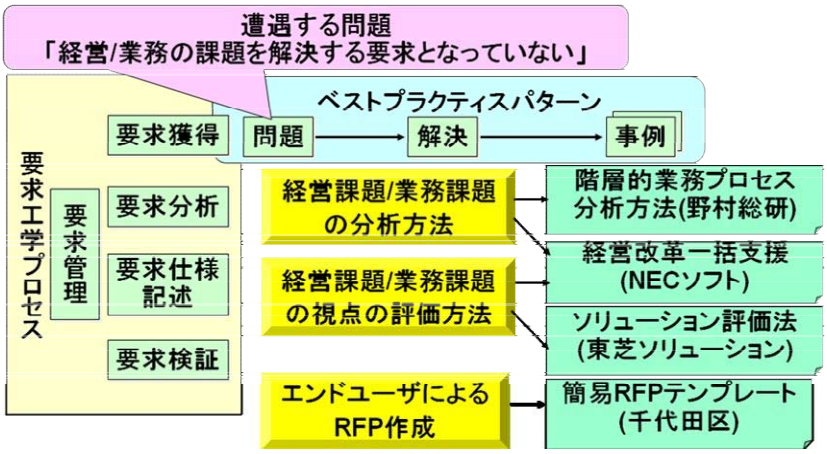


図 A.3　要求工学ベストプラクティスパターン

A5.2. 要求工学ベストプラクティスの整理と体系化

収集したプラクティスとその中から選ばれたベストプラクティスは、要求工学プロセスの要素に対応づけて、次の 5 分類に分けて整理し、番号を付した。

(1) 要求獲得(No.A01~)
(2) 要求分析(No.N01~)
(3) 要求仕様記述(No.D01~)
(4) 要求検査(No.V01~)
(5) 要求管理(No.M01~)

さらに、特定のプロセスに限定されず、要求開発・管理全体を通して適用できるベストプラクティスも見いだした。これを「要求工学戦略」として、次のように分類した。

(6) 要求工学戦略(No.S01~)

A5.3. 要求工学ベストプラクティスの記述形式

要求工学ベストプラクティスは、図 A.4に示す。ベストプラクティスが解決した課題と解決したノウハウを対としてまとめている。

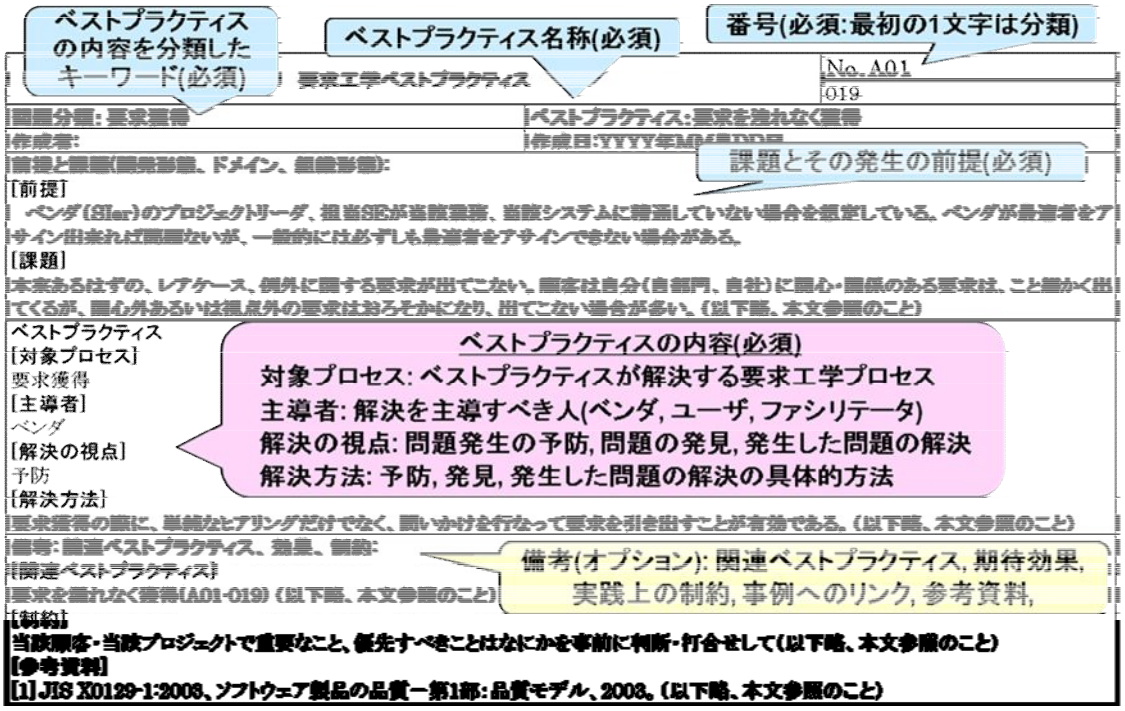


図 A.4 要求工学ベストプラクティスの記述方法

2. ユーザー企業における共通フレーム 2007 の実践的活用 ～要件定義を中心としたシステム構築の改善～

清水建設株式会社 寺田尚弘

2. 1. シミズの IT 戦略の全体像

清水建設は 1804 年に創業した老舗の建設会社です。東京芝浦に本社を構え、従業委員 1 万 1000 人、事業内容は建築および土木の総合請負業として全国に 1500 の建設現場があります。その建設現場で、大体半年から長い工期ですと 2 年、3 年という長いスパンでものづくりをしています。

情報システム部は約 50 人です。特徴的なのは、清水建設が情報子会社を持っていないことです。企画から開発、運用、保守、情報管理といった部分を役割分担しながら、さまざまなベンダやパートナーと連携し、役割分担を明確にして効率よく 50 人で仕事をこなしているという状況です。

清水建設がどのような領域に IT を使っているのかその全体像は次の図の通りです（図 2-1）。

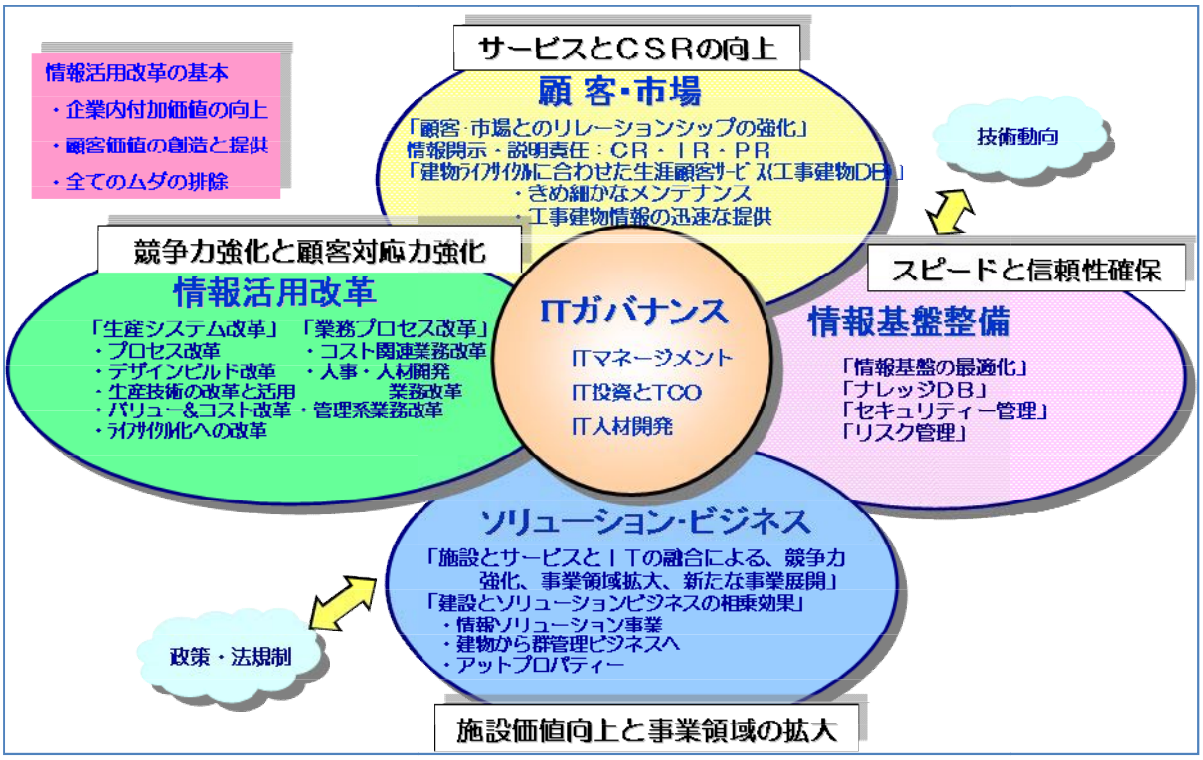


図 2-1. シミズの IT 戦略の全体像

中心に IT ガバナンス、つまり IT のマネジメントや投資把握があります。そこから大きく 4 つの領域があります。

「顧客・市場」は、本報告の中心となる、業務の領域、会社経営の中でどう情報を活用していくのかという情報活用の改革の領域です。それを支える情報基盤の整備ということで、インフラからセキュリティが含まれます。また、建設会社は建物をつくるだけではなくて、その建物の中に情報を付加価値としてお客様に提供するというビジネスも行っています。例えば事務所ビ

ルをつくったときに、地下に駐車場があります。その駐車場の料金精算から管理や、ビル管理のための空調や電気をつかさどるシステムを導入しているのですが、それと会議室やさまざまな予約システムとの連動といった形で、建物の中にソフトウェアやシステムを付加価値として提供するソリューションビジネスも行っています。

昨今はインハウスだけではなくて、お客様や協力会社へのさまざまな情報公開や、情報をやりとりしながらビジネスを回すということで、顧客や市場への情報提供も行っています。

2. 2. システムの変遷

弊社では1970年代にホストコンピュータを導入したときには、PL/1という言語でバッチ中心の開発していました。当然この時代は、経理部なら本社経理部、人事部なら本社人事部だけをユーザとして、このデータをバッチで入力し、帳票を出力するというシステムの開発が中心で、ほぼ100％内部で開発していました。

80 年代になってホストコンピュータ上にデータベースが出現すると、今までのバッチ中心からオンラインで対話型のシステム開発が始まりました。弊社は12支店の下に60～70の県ごとに営業所があります。その営業所の下に建設現場が随時生まれては竣工して、また次のプロジェクトに移るという形態をとっています。オンラインになったときにやっと本支店間に専用線が結ばれて、各支店の経理、各部門の経理、人事から直接入力できる仕組みができました。この時代になって、自分たち独自ではなく外部のプログラマにプログラムを作成してもらうというフェーズに移りました。

90年代でクライアント/サーバの時代になると、Visual Basicなどの言語が出てきます。そうすると、分散でデータベースを置くことが可能になりました。今までパソコンを計算機にしか使ってなかった営業所や全国の作業所に、パソコンだけでなくサーバが置かれ、現場ごとにデータをデータベースに蓄積して、管理するクライアント/サーバのシステムができました。そうすると作業所もユーザになります。このころにはインタフェースの凝ったシステムの開発が行われるようになっており、外部設計フェーズくらいから外部に依頼する比率が1:1くらいになってきました。

2000年になってインターネットの時代になると、Java や.NET、Webという世界が到来しました。建設会社も、自社内だけでなくお客様や協力会社と同じシステムを使い、情報交換しながらビジネスを進める時代になってきました。

このシステムの移り変わりを開発体制で見ると、バッチ中心の時代はすべて内部開発でした。それがオンラインになり、クライアント/サーバから、Webにとなっていく中で、ユーザ部門の要求も、ただ、ものをつくることよりも、どうやって業務改革をして業務を流していくかという要求へ変わっていきました。情報システム部隊の役割も、ものをつくることよりもシステムをどう設計し、システムをどう業務の中に活用していくかという領域になっていきました。

最初は本社の所管部署でだけシステム開発をしていればよかったのが 12 支店に広がり、現場にまで広がり、そして今はお客様まで相手にしてシステム開発をしなくてはいけない。成果に対しても、最初は事務処理の効率化、手でやっていたものをどう効率よく帳票を出すか、効率よく入力するか、事後処理の省力化を考えていればよかった。それが、今までコミュニケ

ーションも含めてうまく情報が回らなかったものを、どう社内で業務を変えていくか。コンピュータやネットワークを使いながらどう処理や業務を効率化していくかという領域。それが顧客や協力会社と連携することで売上や利益にどう貢献していくかというところまでも求められるようになってきています。

今はマルチベンダ、マルチプラットフォームは当たり前のように存在します。ほとんどのコンピュータベンダが、弊社から見るとお客様になります。お客様からお仕事をいただくと同時に、お客様のコンピュータも使わせていただくようになっています。

そして、イントラネットがインターネットに広がることでシステムにセキュリティというニーズが出てきました。今まではホストコンピュータのログイン ID だけ管理していればよかったのが、別の切り口のニーズが発生してきています。

さらに、今年度は多くの企業が内部統制という観点で、今までよりも厳しくシステムの開発や保守や運用のプロセスを監査されています。

2．3．要件定義の見直し

弊社では、2004 年下期からホストコンピュータのダウンサイジングを検討することになりました。弊社のホストコンピュータは 4 年もしくは 5 年に 1 度の、リースの更新の度に更新するか、オープン系にするか検討をしていましたが、2007 年 12 月にリースアップを迎えるホストコンピュータを本格的にオープン系にすることが経営決裁されました。

当時、財務会計、営業、調達、総務のシステムの再構築が課題になっていました。既にオープン系のものとホストが混在していましたから、かなり複雑な処理系になっておりました。オープン系に統一することによってトータルコスト(TCO)の削減と過去 30 年にわたって作り続けたシステムを再構築し、使い勝手と保守性を向上させることが狙いでした。

その中でも財務会計系のシステムは、30 年ホストコンピュータ上で作っていましたので、システム間連携がかなり複雑でした。データベースは、途中で財務会計システムを再構築したため、財務会計のメインは IBM の DB2、その他周辺の会計系システムが Adabas で残っていたため、夜間に必ず両データベース間のバッチ処理を行って、それを翌朝全部確認しなくてはなりませんでした。どちらかにマスタがあれば 1 日遅れでないとデータの反映ができず、同じデータを両方で重複して持つことも問題でした。

システムの開発についても、開発文書が欠落していたり、レベル間が不統一だったりしました。新規に開発したものはドキュメントが整備されていて問題ないけれども古いシステムはソースコードを見ないとわかない、という状態で、どのようにして再構築するかが課題になりました。

すべてをオープン系にする上で注目したのが EA（Enterprise Architecture: エンタープライズアーキテクチャ）という考え方です。EA のフレームワークを導入することで、ばらつきのあるシステムをトータルで最適化し、同時に要件定義工程でドキュメンテーションも含めた標準化を図ることになり、業務要件定義の仕組みづくりへの EA 適用に挑戦しました。

EA 導入の最大の目的は、要件定義を標準化し、業務部門も含めた役割分担の明確化を図ることでした。機能要件のプロセスと成果物を標準化し、ステークホルダーの役割分担を明

確にして、最小限の成果物で機能要件を表現する取り組みを行いました。

2004 年下期から要件定義を開始する予定でしたが、その前段階で 3 カ月をかけて弊社独自の EA フレームワークを構築するために標準化を行いました。

一般的な EA は、ビジネスアーキテクチャからテクノロジーアーキテクチャまでの 4 階層からなります(図 2-2)。

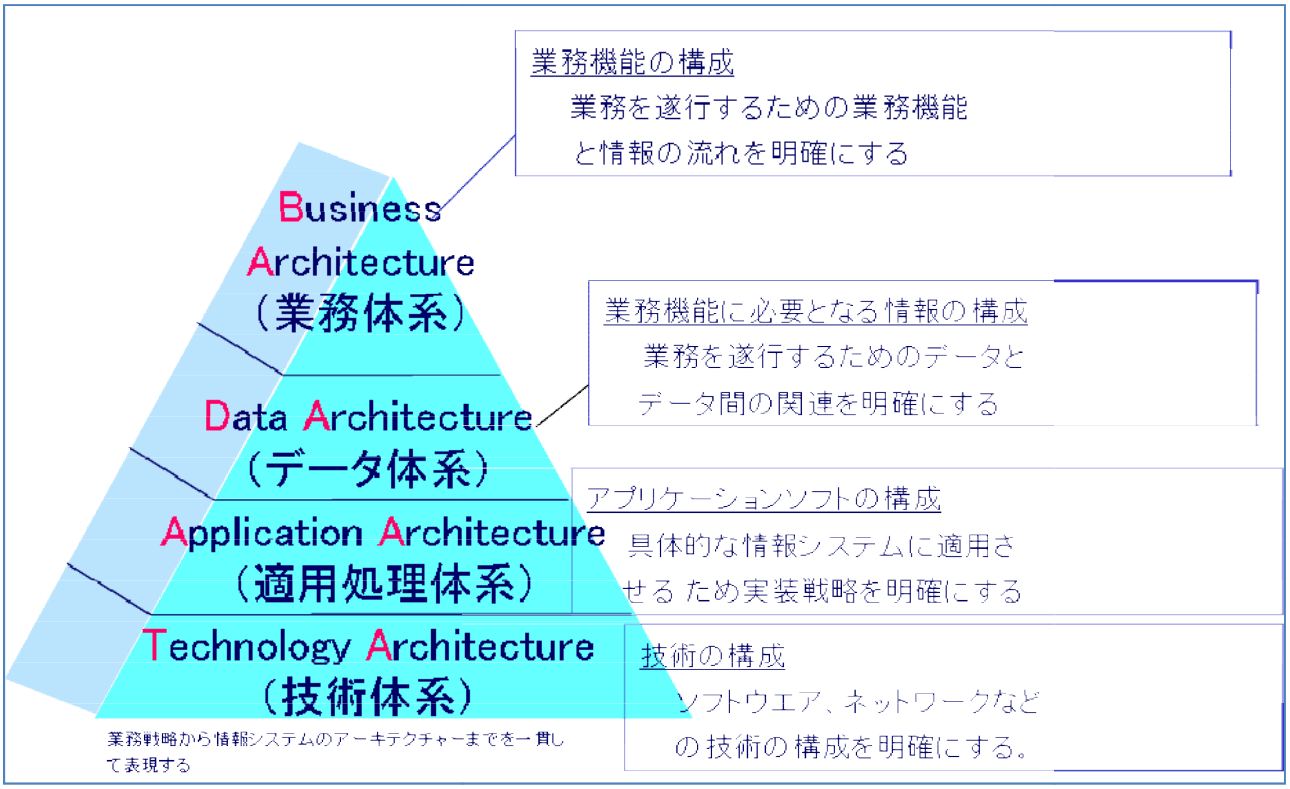


図 2-2. 一般的な EA の体系

これに対し、弊社ではビジネスを実現するアプリケーションの領域とデータの領域を横並びに配置しています(図 2-3)。

中心となるドキュメントは業務フロー図とシステム機能定義書、業務とシステムです。この 2 つを作成し、整理して、すべての会計系システム、サブシステムに対して作り直しを行いました。

その際に役割分担として、経理部門にもシステムを担当するグループがありましたので、経理部のシステム担当と情報システム部の間で、ビジネスアーキテクチャ(図 2-3、BA)、アプリケーションアーキテクチャ(図 2-3、AA)、それぞれの最低限必要なドキュメントに対して、だれがそのドキュメントの責任を持つかを明確にして作業を行いました。

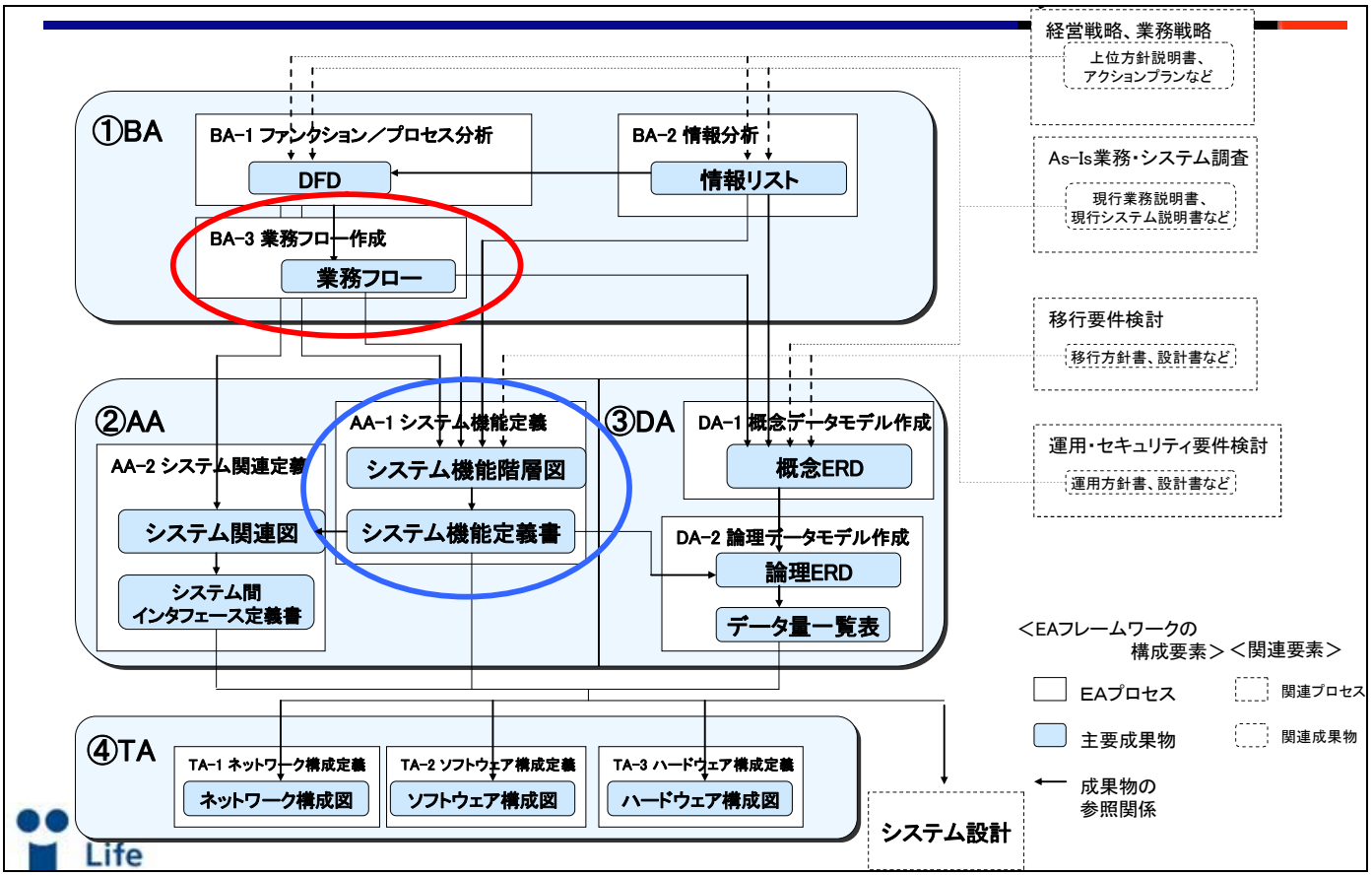


図 2-3. シミズの EA フレームワーク

この結果、2004 年下期、10 月から 3 月までの要件定義工程を経て、2005 年 3 月から実際の基本設計に着手して、2007 年 5 月のゴールデンウイークに会計系も含めてすべてのホストコンピュータアプリケーションがオープン系で稼働を開始しました。稼働後 1 年半を経過しましたが、要件定義工程をしっかり回した関係でかなりトラブルも少なく、期間も当初の予定どおり、品質も要求どおりで、ホストコンピュータのダウンサイジングに成功しました。

2. 4. 超上流を中心にした標準の見直し

2004 年のダウンサイジングが始まる年と同じ時期に、SEC（Software Engineering Center：独立行政法人情報処理推進機構（IPA）のソフトウェア・エンジニアリング・センター）の開発プロセス共有化部会に参加する機会がありました。1 年目には「経営者が参画する要求品質の確保」[1]の一部に採用させていただきました。

超上流の 17 カ条については情報システム部全員に配付しています。私自身も、できる限り開発プロジェクトのさまざまな場面でこの 17 カ条の考え方を話すようにしています。こうした冊子を社内に流通させることで、今何が産業界の常識になっているのか、日常の開発や保守の中で苦しんでいるリーダや開発担当者に理解してもらう活動をしてまいりました。

弊社では業務を整理する上で、業務フロー図を中心のドキュメントに置いています。そのときの原理原則として、業務フロー図にはステークホルダを全部きちんと書いて、そのステークホルダの合意を取りつけなければ次の工程に入っていけないようにしています。そのために業務

フロー図を精査して、皆がレビューで確認します。

業務フロー図を書くことによって、要件定義が使える業務システムになっているかを業務部門や利用者の目線でチェックしてもらうようにしています。

会計系システムについては EA を使うことによって、要件定義内容が整理できました。ただ、それはまだ全社の標準になっていませんし、機能要件は整理できたものの非機能要件はどうなるのかといった超上流の問題を抱えておりました。

そうした中で、超上流も含めた見直しをするきっかけになったのが、システム開発、インフラの導入を含めたさまざまなプロジェクトに関連した標準の見直しでした。

最近まで社内には平成 10 年につくった基本設計工程を中心にしたクライアント/サーバアプリケーションのための開発標準がありました。これは、現在のオープン系や Web のシステム開発、特に要件定義工程の運用には不十分でした。実際に、各開発グループは、オープン系や Web のシステム開発でステークホルダが増加する、サーバも増加する、セキュリティやバックアップも考えなくてはならない、Web の負荷分散はどうするか、試行錯誤しながらシステム開発やプロジェクトを担当していました。

当然、彼らは超上流の工程を何とかしたいという認識を持っていました。しかも、2008 年度から内部統制に対応するために開発標準だけではなくて、レビューや承認する仕組みをつくって実施する、開発だけでなく保守についてもルールに基づいてアプリケーションの開発と運用に関して組織も含めた分離とその体制づくりが必要になってきました。

このような背景から、3 つの目的で、標準の見直しを行うことになりました。

1 つ目は、開発・導入・構築のプロジェクトにおけるプロセスの標準を見直して、プロジェクトが「見える化」されるようにすることです。

2 つ目が、上記を通じて QCD（品質：Quality、コスト：Cost、納期：Delivery の略、ものづくり現場の重要な目標）の管理レベルを 1 ランクも 2 ランクも向上させることです。

3 つ目が、アプリケーションに関して、開発と保守をきちんと分離させることです。保守プロセスについて、保守担当者が自分の力量やスケジュールだけで判断せず、組織の中できちんと承認・保守する仕組みが必要になりましたが、保守や運用に関しての質を向上させれば内部統制における IT 全般統制、業務統制に対応できる仕組みになります。

これら 3 つを目的に 2007 年 5 月に各種標準の見直しプロジェクトを部内で立ち上げました。

インフラの導入・整備に適用し、アプリケーションの保守、運用業務についても標準をつくる。システム開発とレビューについては、既にある標準を見直す。変更管理については、各担当者、グループ任せだった部分を、会社全体としての保守の標準で規定する。アプリケーションの運用管理についての標準も規定する、というアプローチです。

社内の材料としては、現行の開発標準、会計系についてノウハウのたまった EA での機能要件定義、この 10 年間各グループの担当者が独自に工夫してきたプロジェクトの実例です。先ほどご紹介した「経営者が参画する要求品質の確保」[1]で規定される上流工程での作業や成果物と「共通フレーム 2007」[2]を参照し、何が足りなくて、どう強化していくべきかを検討

しました。

内部統制を実施する上では、昨年 5 月の時点では監査法人とのコンサルは始まっていたのですが、まだきちんとした内部統制に対する指針や統制のポイントが出ていませんでした。それを待って開発標準を決めるのでは間に合わないので、経済産業省から出されている「システム管理基準 追補版」[3]を精査して、その中の必要な部分を開発の標準や変更管理の標準の中に組み入れることで標準の見直しにかかりました（図 2-4）。

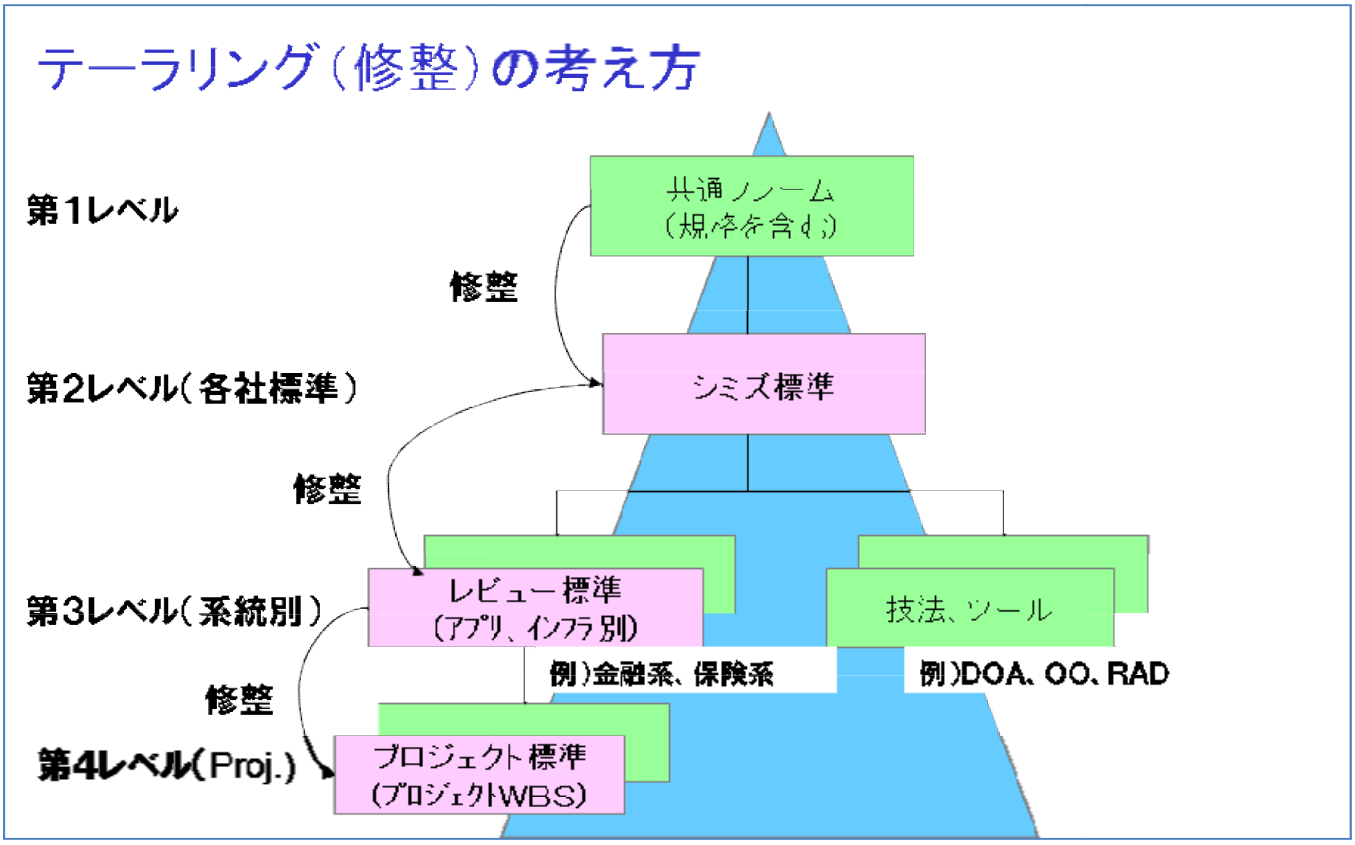


図 2-4. 共通フレーム 2007 の活用

共通フレームをメインに置いて、テーラリングの第 1 レベルを修整して弊社の開発標準を考え、これを第 2 レベルとしました。さらにこれを修整した第 3 レベルは系統別になります。レビュー標準は、アプリケーションのレビューとインフラのレビューは分けようとしています。アプリケーション開発には情報システム部が関与せず部門独自で開発されるもののレビューと、情報システム部が開発に積極的に参画するプロジェクトでアプリケーションのレビュー標準は分けられています。

さらに、この下の各プロジェクトには標準はなく、開発標準のベースになる WBS（Work Breakdown Structure：プロジェクトの作業を細分化し、階層構造で示す方法）があります。この WBS に各システムのシステム機能から出てきたサブシステムを割り当てて、それをいつまでに実施するかというスケジュール管理も含めた工夫が、各プロジェクトでなされています。

次に工程の見直しを行いました。工程 1 をシステム計画、2 を要件定義、3 を基本設計、4 を詳細設計、5 をプログラミングから結合テストまで、6 を基本設計に相対する工程としてシステム

テスト工程、7 に要件定義を検証し、妥当性を確認する運用テスト、に分けました(図 2-5)。

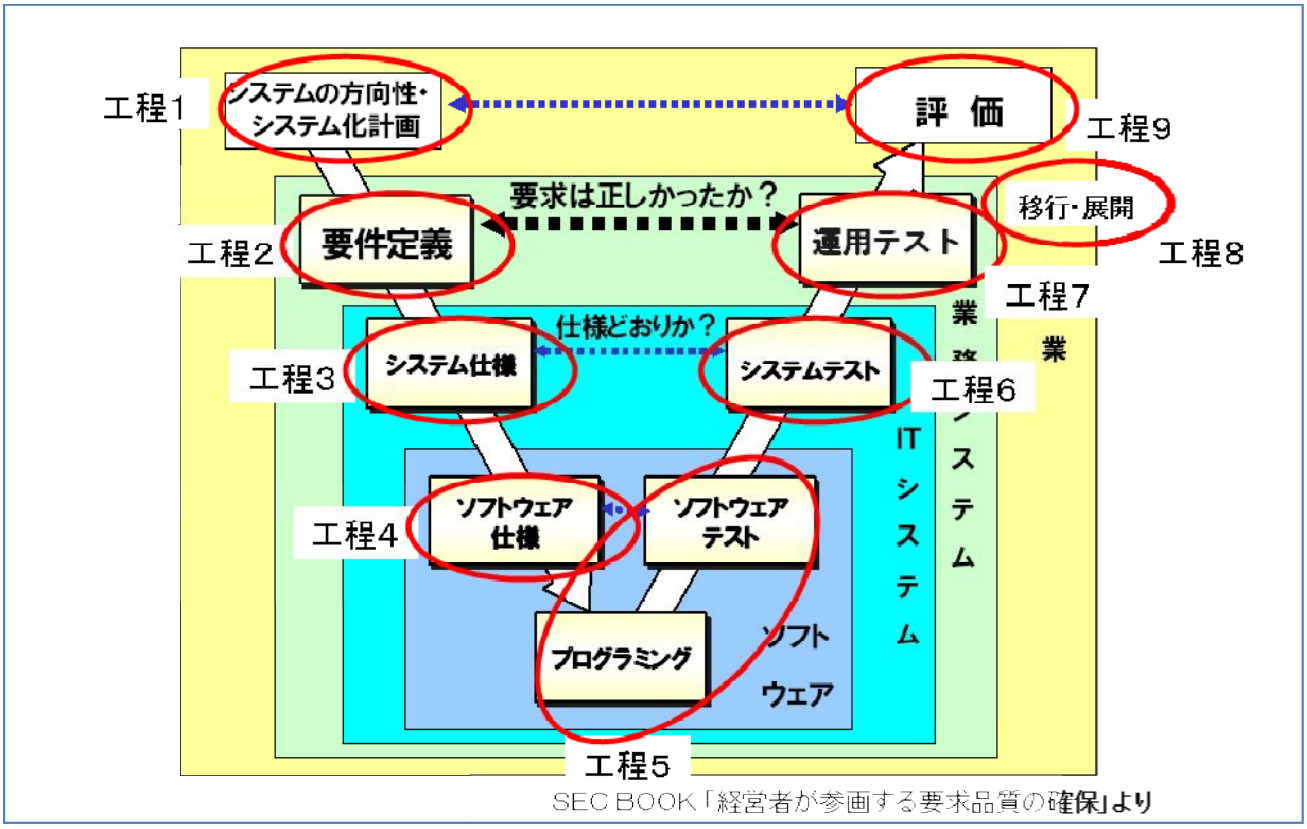


図 2-5. 工程見直しでの上流・下流の位置づけ

今回の中心は要件定義をどう強化するかに焦点を置いています。基本設計、詳細設計、プログラムは、どちらかといえばベンダにお願いするものづくりのフェーズですが、ユーザ側でもその間に、運用設計やシステムテスト、計画の立案を行います。基本設計までで必要なアーキテクチャが全部決まり、基本機能が決まっていますから、どう運用するかというアウトプットはもう出ているのです。これをもとに何が運用で必要になってきて、運用の項目は何なのかを考えて、運用設計はいつまでに何を決めてシステムテストや運用テストを準備するという計画づくりをこの工程でやろうということです。

ベンダがプログラムをつくっている間に、当方では運用設計の実施とそれに基づいた運用テスト計画の立案に着手します。ものづくりのフェーズとは言え、ユーザ側もやるべきことが工程 3 から工程 6 の間には山積みです。

この辺が今までの標準では弱かったので、今回は運用を見据えて、要件定義が終わり、ベンダが開発を開始するときには、こちら側で運用のことを考えながら設計工程に入っていくという標準に変えています。

2. 5. システム開発標準見直しのポイント

要件定義工程の機能強化と、非機能要件についても大幅な追加を行いました。システムテストと運用テストという役割を明確に分けましたので、それに向けてさまざまな準備だとか、移行

や展開も含めた内容の強化を行っています。

もう一つ、この標準により、工程 1 のシステム化計画で概算の費用が出るようにすることです。

プロジェクトオーナたち、システム化計画の段階で稟議を通そうとします。例えば、8 カ月かかるシステムで、ソフトウェアの開発に 3000 万、インフラの構築で 1000 万、合計 4000 万、この金額ですぐに稟議を通すことはかなり危険です。ただ、待つことができない場合があり、多くのプロジェクトで要件定義が明確でないままGOをかけて、外部設計まで進み、後はつくるしかない段階になってから、いくらかかると稟議にかけていました。

システム標準によって、要件定義工程の終了の段階で投資の決裁の稟議を受け、この稟議におけるプロジェクト総費用と決済金額の差を 5%以内にすることが目標です。要件定義で発注者であるユーザ企業が要件をどこまで検討するかにポイントを置いています。

見直し前の要件定義工程は、ほとんど機能要件についてだけで、非機能要件については標準としては定めていませんでした。全くやっていなかったわけではありません。すべて担当者任せであったり、その時点で何を決めるべきか、明確な指針や指標はありませんでした。そこで、さまざまなアクティビティを確認しながら、共通フレームの要件定義プロセスに必要なものを全部埋め込みました。

下流工程では、運用テストに向けて上流工程から何を準備すべきか、運用の準備も含めたプロセスの抽出を、共通フレームの運用プロセスを参照することによって抜き出しています。

例えば、業務詳細については、業務フロー図と一つ一つの業務の箱に対する説明書が一覧で整理された段階で、すべての工程について何をするのかがまとまります。

同じように、システムの機能要件については、システム機能一覧と定義書と、画面、帳票等のイメージを必ずつけます。そして、機能側では、データの機能要件と、見落としがちなシステム間インタフェースに関する他システムとのインタフェースの定義までを機能要件としています。非機能要件として、品質、セキュリティ、技術のほかに、プロジェクトの要件として扱われる移行や展開、教育といったことも弊社の標準の中では非機能要件という位置づけです。

役割分担もそれぞれ所管部署なのか、情報システム部の開発担当なのか、運用担当なのかを決めています。すべての工程のドキュメントは保守フェーズに入ったときに保守対象ドキュメントなのかどうかもここで規定しています。工程内の各作業は、作業間の依存関係についてフローで、このアウトプットはどのインプットになるのかを図で表現しています。

何を書きなさいと言っただけではなかなか仕事が進まないので、開発標準については標準成果物をどうつくるのかという作成要領と、サンプルを用意しました。今回は工程 1 から工程 8 までの間に 250 のサンプルを用意しました。実プロジェクトのドキュメントを対象に工程 1 からテストまで、一貫したストーリで流れるサンプルを、パートナーの協力を受けて 3 カ月間かけてつくりました。

作成要領は、成果物についてその記述内容を規定しており、この書き方ではいけないということを規定しているわけではありません。

成果物作成要領にはそれぞれの成果物を何のために書くのか、どんなことを書くのか、注

意点として何があって、その成果物のインプットやアウトプットが何になるのかを規定しています（図 2-6）。

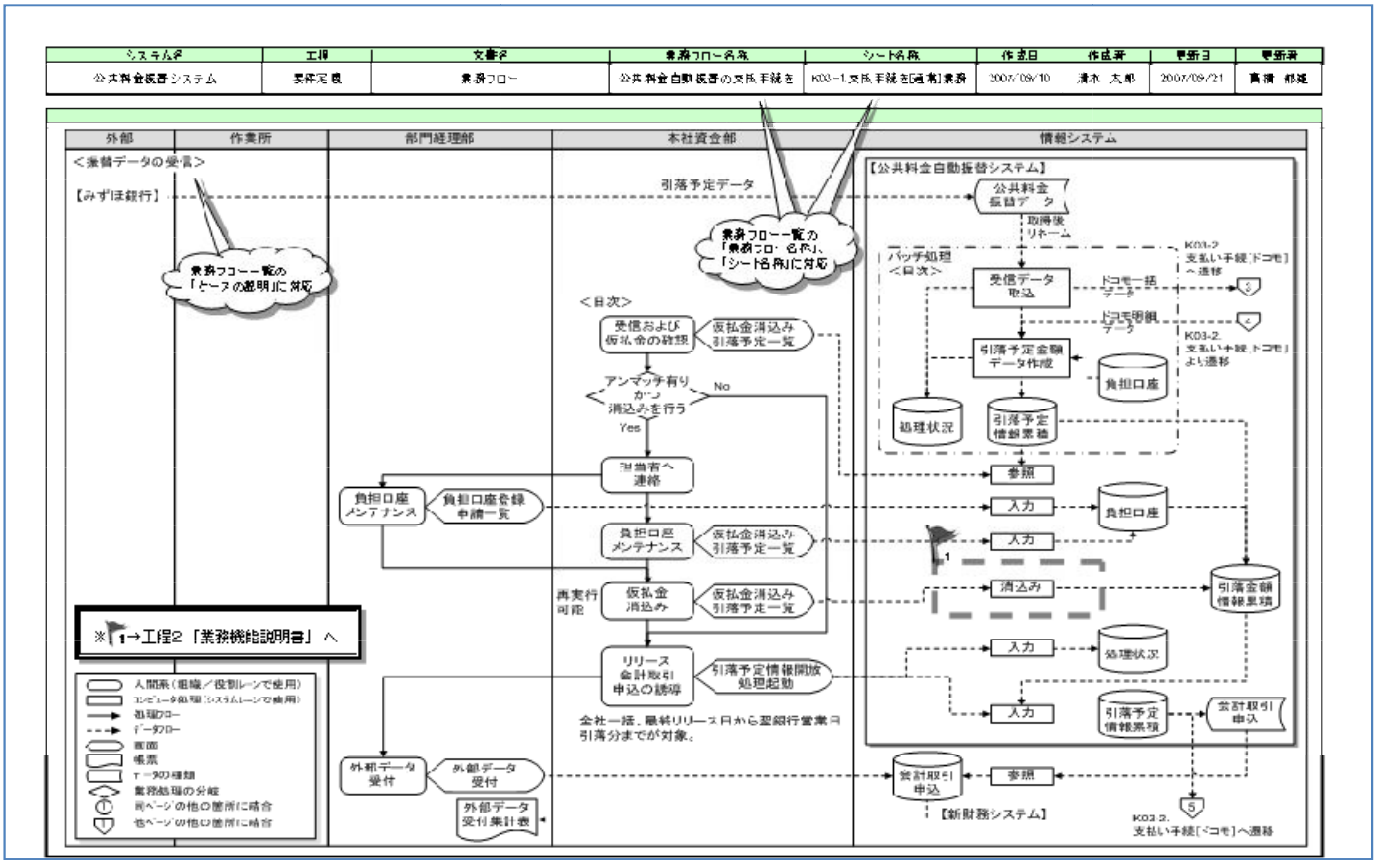


図 2-6. 成果物サンプル（例）

そしてすべてに実帳票のサンプルをつけ、その中で記述上のポイントを記載しています。図 2-6 の中で旗が立っていますが、別のドキュメントに業務機能説明の詳細があるという印です。これを見るとストーリがわかるようになっています。

もう 1 つ、業務フロー図の一番右側に必ず情報システムというレーンを設けています。ここに登場するのはステークホルダです。業務フロー図では情報システムもステークホルダです。人間系で処理をするのか、コンピュータで処理をするのか、コンピュータと人間の、あるいはコンピュータとのインタフェースがあるのかがここから読み取れます。例えばここで、受領および仮払金の確認という業務は、本社経理部が画面の前で処理をするということで、ここは新しいシステムができる。しかし、担当者への連絡については電話か紙が送られているのでコンピュータの線がありません。そして、システムとシステムの間に線が結ばれているところは、バッチ系の処理が必要になってくる。こうすることによって、システム間インタフェースを持っているところはここから抜き出せば、業務がきちんと流れるかシステム機能階層図が書けます。システム間インタフェースも抽出できますので、この業務フロー図を標準にしています。

画面サンプルには、画面レイアウトではなく画面イメージをつけるようにしています。この仮払金の確認は画面で行いますと説明するだけではイメージがわきません。本社の経理担当者にどのような画面を見せればこの業務ができるかを、このイメージを見せることによって確認することが

できます。インタフェースについては、帳票や画面を可能な限り書いて、入力できるか、業務が回るかを、システム面からも検証できます。このレベルで画面をつけておけば、ベンダに要件定義書を渡したときに、ソフトウェアの要件については精度のかなり高い見積もりが出てきます。

2. 6. 要件定義工程の作業関連

要件定義工程について整理します。業務の詳細については業務フロー図と業務機能説明書を書きます。システムの機能については、この業務フロー図から抜き出されたシステム機能の一覧に対して、そのシステムの機能説明書と入出力の定義ということで、定義というより入出力イラストをつけるようにしています。これらをもとにシステム間インタフェースを検討し、データについては概念データモデルとエンティティの定義まで書きます。これが機能要件になります。

工程 1 での SLA や機能要件や業務フローなどから非機能要件と呼ばれるものが出てきます(図 2-7)。

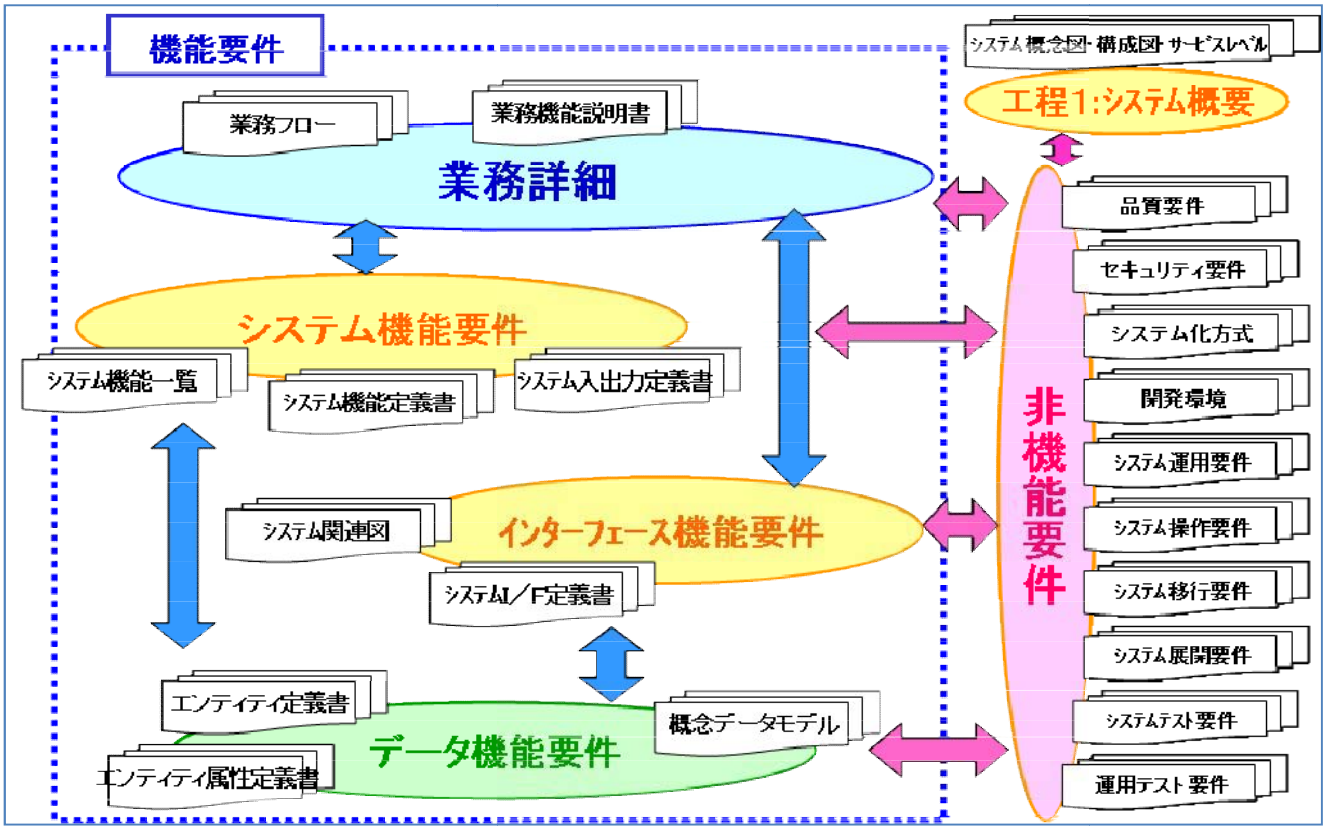


図 2-7. 要件定義工程の作業関連

オープン系ではセキュリティが厳しく追及され、セキュリティ要件についても記述することが定義されています。

さらに、業務フロー図を書くことによってどの局面でだれが何をするかがはっきりわかります。ある時点でシステムに求める機能のアクセスコントロールついて、こういう登場人物がいて、どの機能を使うのか。利用するだけなのか、本当に更新が入るのかといったことが整理できます。

この情報をセキュリティ要件として渡すことによって、アプリケーションで実現するか、ミドルウ

エアで実現するかベンダが工夫をして提案してくれます。

このように、標準の中では、ほかにシステム化の方式についての話、共通のデータベースを使うことを前提にするか、データベースを独自に持った方がいいのか、そうした提案もお願いすることにしています。

2. 7. プロジェクトレビュー標準の見直し

レビューについての標準の見直しは、品質を上げる上で、また内部統制上重要な位置づけを持っています。

今回はシステムのライフサイクル全般にわたって、単にものをつくるだけではなく、移行・展開・運用・保守といった観点から、各フェーズでのレビューを行うのにあたり、それまでプロダクトを中心としたデザインレビューという位置づけから、プロジェクトを見詰めていくことしています。名前も DR から PR に変更しました。

特にシステムの開発計画に対するレビューとカットオーバーに関するレビューについては、業務所管部署長と情報システム部長、両方の承認を必須にしました。この見直しにあたっては、共通フレームの支援ライフサイクルプロセスにおける品質をどう見るか、検証と妥当性確認をした上で、だれが何のためにこのレビューを実施するのか、だれとだれが何を合意するのかを確認するために共通フレームを参照しています。

開発工程別にレビューを規定していますが、特筆すべきは、要件定義の中に 2 つのレビューがある点です。社内では PR1、PR2-1、PR2-2 と呼んでいます（図 2-8）。

工程		PR名称	略称	目的	承認者（確認者）					
					業務所管部門・部署		情報システム部			
					部門・部署長	開発責任者	部長	開発責任者	運用責任者	保守責任者
1	システム化計画	システム化計画レビュー	PR1	・システム化計画の確認（了承）＊1	○	○	○	○		
2	要件定義	要件定義レビュー	PR2－1	・要件定義内容（RFP）の承認		◎		◎	◎	◎
		システム開発計画レビュー	PR2－2	・システム開発計画の承認	◎		◎		◎	◎
3	基本設計	基本設計レビュー	PR3	・基本設計内容の承認		◎		◎		
4	詳細設計									
5	プログラム作成 単体・結合テスト	システムテスト計画レビュー	PR5	・運用テスト計画の確認 ・システムテスト計画の確認		○		○	○	
6	システムテスト	運用テスト計画レビュー	PR6	・システムテスト結果の承認 ・運用テスト計画の承認		◎		◎	◎	
7	運用テスト	カットオーバーレビュー	PR7	・運用テスト結果、移行・展開・教育計画の承認 ・カットオーバーの承認	◎		◎		◎	◎
8	移行・展開			・プロジェクトの評価						
9	運用・保守			・運用の評価 ・効果の把握						

＊1：インフラ導入・整備については、情報システム部長及び関係者の承認が必要

Life Cycle Valuation

FUJITSUファミリ会関東支部 第3回トレンドセミナー20081023

(C) Copyright 2008 Shimizu Corporation. All rights reserved.

SHIMZ 40

図 2-8. プロジェクトレビュー一覧

PR2-1 は、要件定義の内容について承認を得るための要件定義書、PR2-2 は開発計画に対する承認を得るための要件定義書です。後者は、要件定義書をさまざまなベンダに渡して戻ってきた提案書を検討し、ベンダ選定と開発計画の承認を得るためのものです。

例えばレスポンスに関してはどういう状況でどの処理を行うと何秒までのレスポンスが必要になるか、バックアップはどのタイミングで実行して、どこまで戻れなくてはいけないか、そうしたさまざまな要件まで全部検討し、業務が回るための要件の妥当性の確認を行うレビューと、費用と期間と品質と体制と保守・運用まで含めたライフサイクル上の計画でよいかという承認を得るということで、2 つに分けています。

詳細設計から結合テストまではほぼ請負で行われて、システムの性格にもよりますが、ここまでは、ベンダの責任です。工程 6 のシステムテスト結果のレビューでは、運用テストの計画についてレビューをする。システムテストはここまでできましたので、運用テストが終わったら、この基準でカットオーバーすることを、工程 6 の PR6 で承認を得て、その結果を受けて、運用テストが終わった後に運用テストの結果と移行・展開・教育の計画の承認、併せてカットオーバー基準を満たしているかを承認してカットオーバーに移ります。

工程 2 における PR2-2、PR7 がプロジェクトのキーになります。開発標準ができ上がって以降の部内では、今日 PR2-1 だとか、PR2-2 が通ったというように、PR というと開発担当者の間で今何をしているのかが互いにわかるようになりました。そして、業務部門の責任者が出てきて承認するということは、その内容を OK としたということですから、要件定義に今まで以上に業務部門の担当者が積極的にかかわるようになってきました。

プロジェクトレビューについても同じように開催要領を標準の中で規定しています。目的、開催方法、何を承認するのかといったことを規定しています。

PR について整理すると、工程 2 の中には要件定義の内容を検討し、RFP（Request for Proposal：要求仕様書）ができた段階で要件定義書のレビューをする。この RFP に基づいてベンダ提案を受けて、そこからシステム開発計画を精査し、その計画書に基づいて計画レビューを行う。これを受けてシステムテストが終わった段階の運用テスト計画レビューで、カットオーバー基準を確認し、それに基づいて運用テストの結果でカットオーバーする。これが PR の流れです。

2．8．その他の新たに規定した標準

変更管理についても標準を規定しました。

業務部門から依頼書として来る変更要求の内容を PMO（Project Management Office：プロジェクトマネジメントオフィス）で確認して、どこの工程から開始するかをシステム開発の標準にのっとって規程をします。これは要件定義に関係するから工程 2 から、これはシステムの画面の機能変更だから工程 3 からというように開始工程を規定するのです。こうしてその工程以降、プロジェクトはそのまま開発標準とレビュー標準に基づき、そこから移行の作業を始めることになります。

変更管理については、要件定義工程、基本設計工程、詳細設計からの入り口に関する開始規程が規定されています。例えば業務フロー図に変更がかかる。業務フロー図に変更がか

かるということは業務部門のステークホルダの運用が変わるということです。もしくは、技術要件や運用要件が大幅に変更になって運用テストが必要になるようなインフラの要件があった場合は要件定義から。要件定義から入った場合は必ず工程 7 まで実施。基本設計の 3 から入った場合は 6 まで実施。詳細設計から入った場合は 5 まで実施ということで、入り口と出口の規定をするようにしています。要件定義書が回ってくると担当者が来て業務依頼書がきましたが、これは工程 3 からでいいですねといった相談が来ます。よく内容を見ると、運用要件が変わるからこれは工程 2 から回して運用テストまでしてから、カットオーバーしようというように、了解を取りつけて開始規程を決めています。

アプリケーションの運用管理についても標準を規定しています。これは開発と運用の分離をする上で、開発者が運用に引き継ぐために運用手順書をどうつくるべきかを規定しています。

主に業務運用としてバッチのスケジュール登録、バッチ処理結果の確認、バックアップの確認、リリースに関して運用者が実施するようにしています。運用プロセスには運用の評価というアクティビティがありますが、アプリケーション運用引き継ぎしたものについては月に一度、運用グループが主体になって開発者と運用者が月間アプリケーション運用報告という会議を設けています。そこでは、今月どのような運用がなされていて、その中で例えば障害がどう発生して、その障害がインフラの障害なのかアプリケーションの障害なのか、それがどう解決され、運用状況がどうなのかということを、運用者から開発者と業務部門に説明し了解するというフェーズを設けています。

2．9．最後に

2007 年 5 月から 10 月まで標準の見直しを行い、11 月から約 1 年間、新しいプロセスで実際に業務を回しているのですが、共通フレームを適用することにより、個人の経験や仕事の進め方でばらついていた開発作業が標準という背骨を通すことによって、何をよりどころに開発作業を決めるのかが判断できるメリットがあると思います。

ものづくりだけではなく、要件定義の上流から運用テスト、保守といったシステムのライフサイクル全般にわたって何が必要になるのかという点で、共通フレームの考え方やフレームにおけるさまざまなプロセス、アクティビティがとても参考になります。

さらに、開発者は新しい標準や新しい成果物が追加されることにものすごく抵抗があるのですが、これが国際規格、日本の常識、つまり業界の共通認識だと説得できます。

要件定義工程は発注者側の責任です。業務の整備については業務部門の人間が主体になって作業を進めない限り、情報システム部門だけではなかなかできません。要件定義プロセスにおいて、使う人が要件定義者なのだと業務部門担当者の役割を明確にする上でも、共通フレームを参照することは意味があります。

内部統制を実施する上で開発標準をどうやってつくったのか質問を受けることがありますが、弊社の開発標準は共通フレームを参照しましたと、これだけで監査法人は「そうですか」と納得します。どこまで監査法人がそれを読んでいるかわかりませんが、監査法人は共通フレームを知っています。

開発者からは、各工程における作業や成果物の目的や必要性、それが次工程でどう使わ

れるかも含めて標準を規定していますので、品質が向上したといった声が聞かれます。

参考文献

[1] 情報処理推進機構ソフトウェアエンジニアリングセンター（編集), 経営者が参画する要求品質の確保―超上流から攻めるIT化の勘どころ, 第2版, オーム社, 2006.

[2] 情報処理推進機構ソフトウェアエンジニアリングセンター（編集), 共通フレーム2007―経営者、業務部門が参画するシステム開発および取引のために, オーム社, 2007.

[3] 経済産業省, システム管理基準 追補版, 2007年3月
http://www.meti.go.jp/press/20070330002/systemkijun-tsuiho.pdf.

3. 東京海上日動の抜本改革システム開発における要件定義の進め方

東京海上日動システムズ　西村　光司

3．1．はじめに

2005 年に開発着手し 2008 年 5 月にサービスインした、東京海上日動の抜本改革システムにおける要件定義の進め方について、様々な局面で実施した「効率的かつ網羅的に要件を定義するための対応策（工夫したこと、苦労したこと）」を紹介しますので、参考にしていただければ幸いです。

本レポートは、

1．東京海上日動システムズ株式会社（以下、システムズ）の組織体制

2．抜本改革プロジェクトの概要（背景、内容、システム開発のスケジュール）

3．アプリケーション・オーナー制度（以下、アプリオーナー制度）

4．抜本改革システム開発における要件定義の進め方（工夫したこと、苦労したこと）

で構成されまる。

上記 3．にある「アプリケーション・オーナー制度」とは、東京海上日動火災保険株式会社（以下、東京海上日動）からシステムズが受託している全システムを対象に、その一つ一つのシステムを所管する東京海上日動の部門、たとえば自動車保険の契約システムなら自動車保険の商品開発部門、損害サービス（保険金支払）システムなら損害サービス部門、経理システムなら経理部門、等をアプリケーションオーナーと定め、システム開発におけるアプリケーションオーナーとシステムズの役割・責任を明確化し、それに則ってシステム開発を進めていく制度のことです。

3．2．東京海上日動システムズ株式会社の組織体制

図 3-1 が、システムズの組織体制の概要です。

システム開発部門、IT サービス（システム運用）部門、コーポレート部門に分かれており、全体で 1,250 名の人員がおります。

システム開発部門では、東京海上日動のアプリオーナーからの依頼を受けて、保険商品システムや損害サービス（保険金支払）システムを開発しています。

IT サービス部門は、性能、キャパシティ面を含め、全システムを安定的に稼動させる役割を担っています。

SLA(Service Level Agreement)に関しては、各システムの目標稼働率を定め発注者である東京海上日動と合意し、その達成に向けて様々な取り組みをしています。

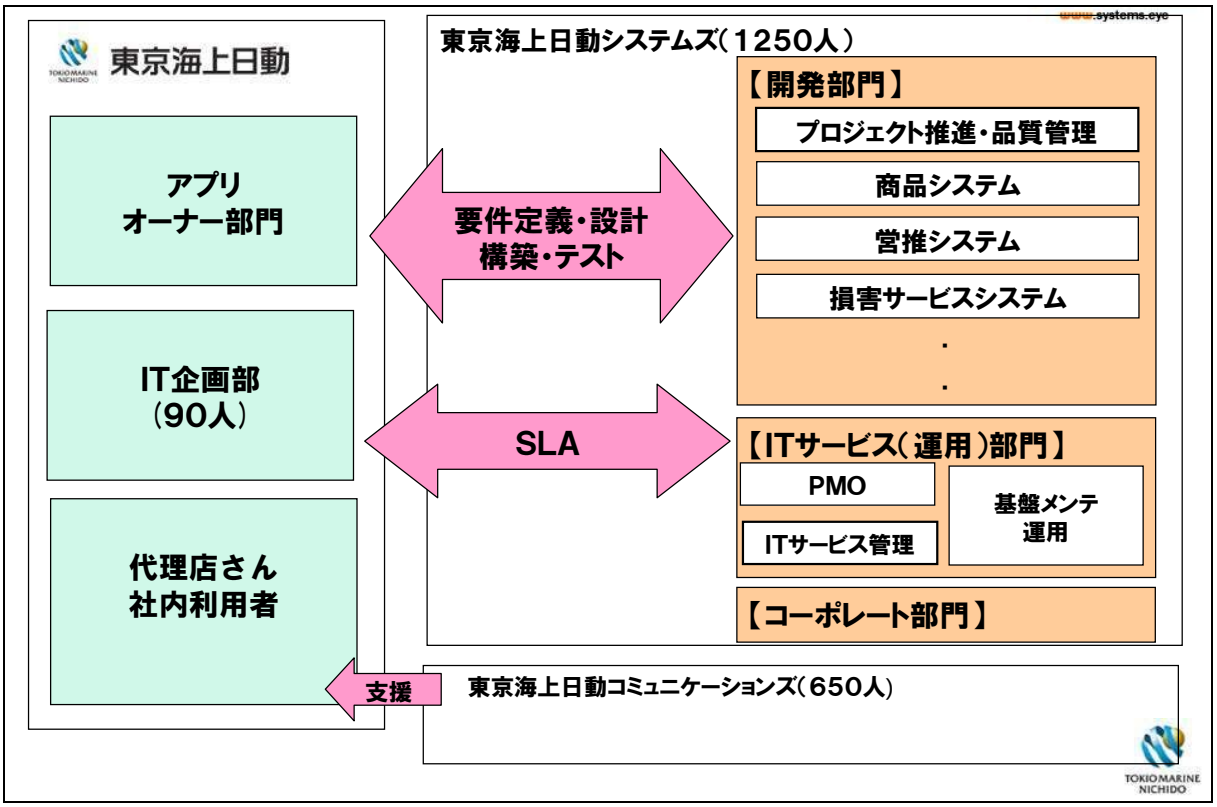


図 3-1. 東京海上日動システムズの組織・体制

3. 3. 抜本改革プロジェクトの概要

図 3-2 が、抜本改革プロジェクトとそのシステム開発のスケジュールです。

抜本改革は、5 年前の 2003 年 12 月の東京海上日動の役員合宿で、商品、事務、システムに関する課題や検討の方向性について論議されたことから、スタートしました。

代理店さんへのヒアリングや社内への説明を始め、様々な議論を経て、2004 年 10 月に方向性を基本計画としてまとめました。2005 年 3 月に開発スケジュールや実施内容をまとめた基本計画が策定され、2005 年 4 月からシステム開発に着手しました。

システム開発規模は、8,000K ステップ、20,000 人月を見込みました。下記の開発日程を経て、2008 年 5 月にシステム基盤と自動車保険がサービスインしました。(火災保険、傷害保険、超保険等については、2009 年度以降のサービスインを目指して、現在開発を進めているところです。)

2005 年 4 月から基本要件

2005 年 10 月から詳細要件

2006 年 4 月からプログラム開発

2006 年 7 月からシステムテスト(機能テスト)

2007 年 2 月からアプリオーナーテスト
2007 年 6 月から性能やバッチ運用等の全体テスト
そして 2008 年 2 月からのリハーサル等のリリース準備

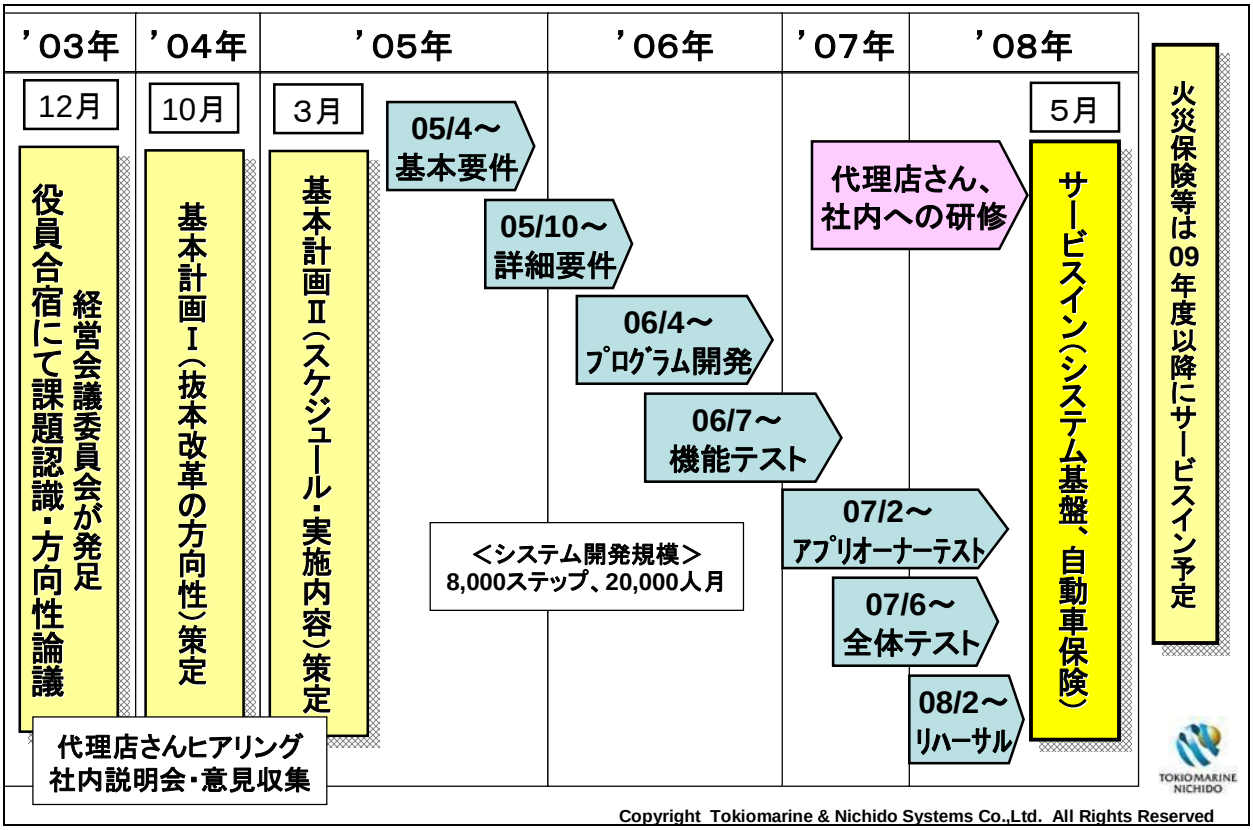


図 3-2. 抜本改革プロジェクトのスケジュール

2003 年 12 月の役員合宿で論議された内容は、次のとおりです。
まず、(当時の)現状課題として、次の点がということが挙げられました。
(1)事務処理の多さ・複雑さ
(2)複雑でわかりにくい商品内容
(3)保険商品毎に異なるシステムによって「複雑で多様な事務処理に追われ息がつまりそう」といった空気が代理店さんや社内に広がっている
(4)お客様も「保険はややこしくて手続きが面倒」との感想を持っていらっしゃる

一例を挙げますと、お客様にお支払いいただく保険料の払込方法の複雑化です。一括払、年払、月払といった払込回数や現金払、口座振替、コンビニ払といった決済方法によって約 50 種類の保険料払込方法が存在し、自動車や火災などの保険商品毎に適用可能な払込方法が少しずつ違っている、という状況でした。

システム面でも、1982 年に開発された基幹システムに対し、保険料払込方法の複雑化、90 年代後半の規定緩和による保険商品の多様化、IT 技術の進歩や代理店さんの機械化、という変化を次々と反映していきました。様々なニーズに応えてきたものではありますが、代理店さんや社員が逐一マニュアル等で確認する必要がある、システム開発面でも工数の増大化、開発期間の長期化を招いていました。

このような状況を打開するために、次のような方向性・コンセプトで抜本改革プロジェクトを進めていこうということが役員合宿で決議されました。

抜本改革プロジェクトの方向性とコンセプト

保険商品、事務プロセス、システムを見直して再構築し、業務プロセス、つまり仕事のやり方を根本から改革し、活力ある会社にしていこう。お客様への対応についても、商品内容をきちんとご説明し十分にご理解いただく、正しい保険証券をスピーディーにお届けする、正しい保険料を頂戴し正しい保険金を迅速にお支払いすることを実践し、お客様にご満足いただける会社を目指す。

3．4．アプリオーナー制度

アプリオーナー制度は、当該システムを所管する東京海上日動の部門、たとえば自動車保険の契約システムでしたら自動車保険の商品開発部門、損害サービス（保険金支払）システムでしたら損害サービス部門、経理システムでしたら経理部門、をアプリオーナーと定め、システム開発におけるアプリオーナーとシステムズの役割・責任を明確化し、システム開発を円滑に進めシステム品質を高めていくための制度です。

2000 年にアプリオーナー制度を創設しましたが、それ以前は、要件定義が不十分でアプリオーナーとシステムズの要件認識に齟齬が生じる、テストが不十分で要件とシステム仕様が合致していない、といったケースがあり、それらを要因としたトラブルが頻発していました。

それらを改善しシステム品質を高めるためには、要件定義やテストの精度を高める必要があり、アプリオーナーとシステムズの役割を整理・明確化した上で、それらがうまく機能していくためのスキームを構築しました。

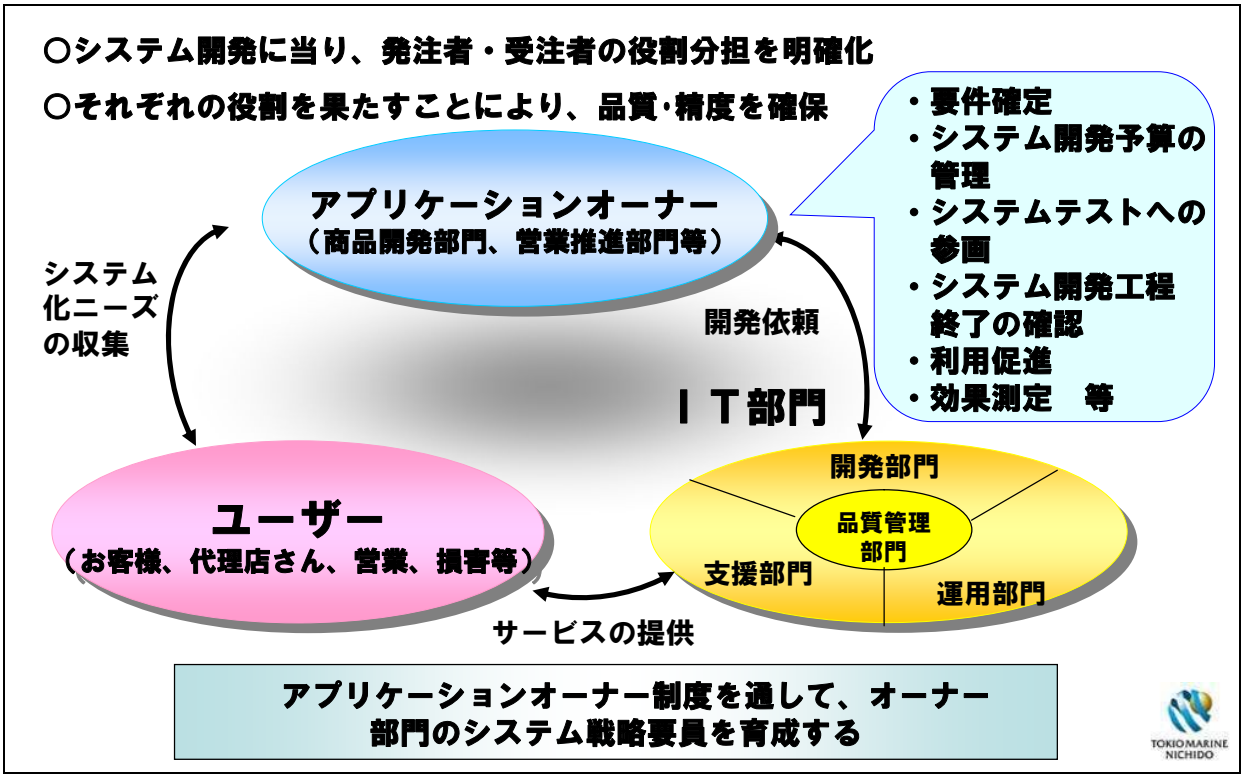


図 3-3. アプリケーションオーナー制度

表 3-1 に示すように、システム開発の工程毎に、アプリオーナーとシステムズの役割を整理しました。

表 3-1. アプリオーナーとシステムズの役割

システム開発工程	アプリオーナーの役割	システムズの役割
基本計画	・現行の事務・システムの問題点やユーザーのニーズの把握 ・システム化する業務プロセスや機能の決定 ・スケジュールの策定とプロジェクト体制の構築	・システム化実現性の検討 ・開発規模、コストの見積もり ・リスクの明確化
要件定義	・業務フローなどの業務要件固め ・具体的な画面や帳票を設計	・業務要件に関しシステム化の実現性を検討 ・データベースやファイルの設計 ・性能面などの非機能要件の検討

プログラム開発	・業務要件に基づいたテスト計画の策定 ・事務処理マニュアルの作成 ・ユーザー教育計画の策定	・システム内部構造の設計 ・プログラムの開発 ・システム仕様に基づいたテスト計画の策定
システムテスト	・業務要件に基づいたテストの実施 ・マニュアル提供、研修実施などユーザーへのリリース準備	・システム仕様に基づいたテストの実施 ・データ移行やプログラム配備などのリリース準備
稼動後の評価	・ユーザーに活用を情宣 ・利用状況の把握と効果の測定	・開発規模やコストの実績確認 ・システムの安定運用

そして、アプリオーナー、システムズ双方の役割がきちんと果たされていることを確認するために、次表のとおり、各工程の終了時にレビューを行う制度を設けました。（全システム開発案件を対象に実施しています。）

表 3-2. システム開発の工程別レビュー

システム開発の工程	レビューの種類
基本計画	プロジェクト計画レビュー
要件定義	要件確定レビュー
プログラム開発	テスト計画レビュー
システムテスト	サービスインレビュー

レビューを実施することによって、決まったことや未決事項を組織間でオーソライズし共有認識することが重要です。具体的には、アプリオーナー、システムズ、管理部門など複数の視点で内容を吟味する、リスクを共有する、プロジェクトの責任を組織で持つ、役割分担を明確にする、スケジュールを共有する、といったことがレビューの目的・観点です。

3．5．抜本改革システム開発における要件定義の進め方

次に、本レポートの本題である「抜本改革システム開発における要件定義の進め方」について紹介します。システム開発工程毎に、どのような状況が発生し、それに対してどのような対策を講じていったか、といった形式で紹介します。

「発生した状況」については、ある程度予期していたもの、予期していなかったもの、様々ですが、ほとんどは抜本改革が「大規模」で「長期間」に亘るプロジェクトであることに

起因しています。決めなければならない要件が膨大かつ細部に亘っており通常のシステム開発と同様のスキームでは決めきれなかった、開発期間中に様々な変動要素が発生した、ということです。

3．5．1．基本要件定義工程

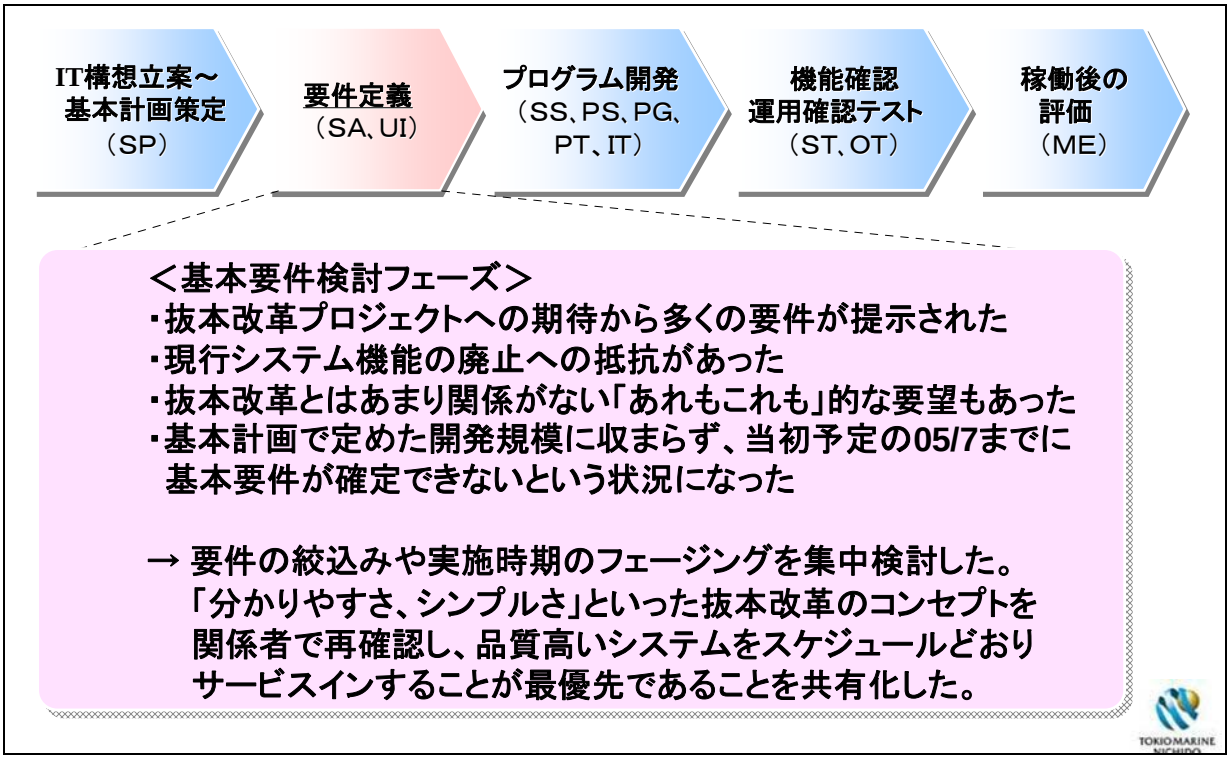


図 3-4．要件の適正化に向けた取り組み－「要件定義」工程－1

2005 年 3 月に完成した基本計画に基づいて、2005 年 4 月から基本要件の検討を開始しました。抜本改革プロジェクトの概要でご説明したとおり、「複雑で個別最適化している商品、事務、システムを刷新し、分かりやすくシンプルなものにしていこう」というコンセプトを定めたわけですが、これに対する社内の期待は大きく、いざ基本要件の検討に入ると、多くの要望が殺到しました。また、現行システムの機能を廃止することへの抵抗もありました。さらには、この機会にと、抜本改革とはあまり関係のない要望もあがってきました。

その結果、

(1)それらの全てをきちんと検討していくと時間がいくらあっても足りない

(2) 開発規模も、基本計画で定めた計画を大きく超えそうだ

といった状況になり、予定の 2005 年 7 月に基本要件検討が確定しない、といった事態になりました。

そこで、大きな要件の塊り毎に、要件を絞り込むためのポイントを整理する（例：.対象と

なる保険契約件数や利用頻度などの観点で優先順位を付ける)とともに、一部機能のサービスイン時期をずらすことによって開発要員の山崩しができないか、といったことをシステムズ内で検討し、アプリオーナーと折衝に入ったわけです。

アプリオーナーとしても、実現したい、存続したいという思いが強い案件ばかりであり、厳しい調整になりましたが、「分かりやすくシンプルなものにしていこう」という抜本改革のコンセプトを何度もリマインドするとともに、品質高いシステムをスケジュールどおりにサービスインすることが最優先であるという認識を共有化し、基本要件を適正な範囲に収めていきました。

3. 5. 2. 詳細要件検討工程

2005年10月から詳細要件検討を開始しましたが、まず実施したのが、アプリオーナーの多摩ごもりです。

通常、アプリオーナーの本拠地は丸の内で、システムズは多摩センターです。普段のシステム開発では、互いに丸の内や多摩に行き来して打ち合わせを行っています。(電話、メール、テレビ会議も利用しますが、要件をしっかり決めるためには打ち合わせが必要だと思っています。)

しかしながら、抜本改革の詳細要件は大量・多岐にわたり相当な時間を要します。普段のやり方では、日常業務に時間を取られ、まとまった検討時間が確保できません。それでは到底終わらないという危機感から、集中検討できる環境を設けることとし、アプリオーナーに多摩センターにこもってもらうことにしました。

その中で、複数の部署がアプリオーナーであるシステムについて、アプリオーナー間の役割責任が不明確なケースがいくつか発生しました。いわゆる「お見合い、ポテーンヒット」です。最終責任を持つ部署を決めたり、画面や項目単位に守備範囲を決めたり、様々な調整を行いました。

2～3 ヶ月間のアプリオーナーの多摩ごもりによる集中検討によって詳細要件が固まってきましたので、2006 年 1～2 月に詳細要件ウォークスルーを実施しました。テーマ別に検討していたチームが事務処理フローや画面、帳票を持ち寄り、一同に会して縦横斜めから整合性を確認し共有化することが目的です。ヘルプデスクのメンバーにも参加してもらい、日常寄せられているユーザーの意見がうまく反映されているか、確認してもらいました。

ウォークスルーの場で新たな要望が出され、折角固まりつつあった詳細要件が再検討になる、といったこともありましたが、ウォークスルーは要件の整合性確認、共有化という点で有意義でした。

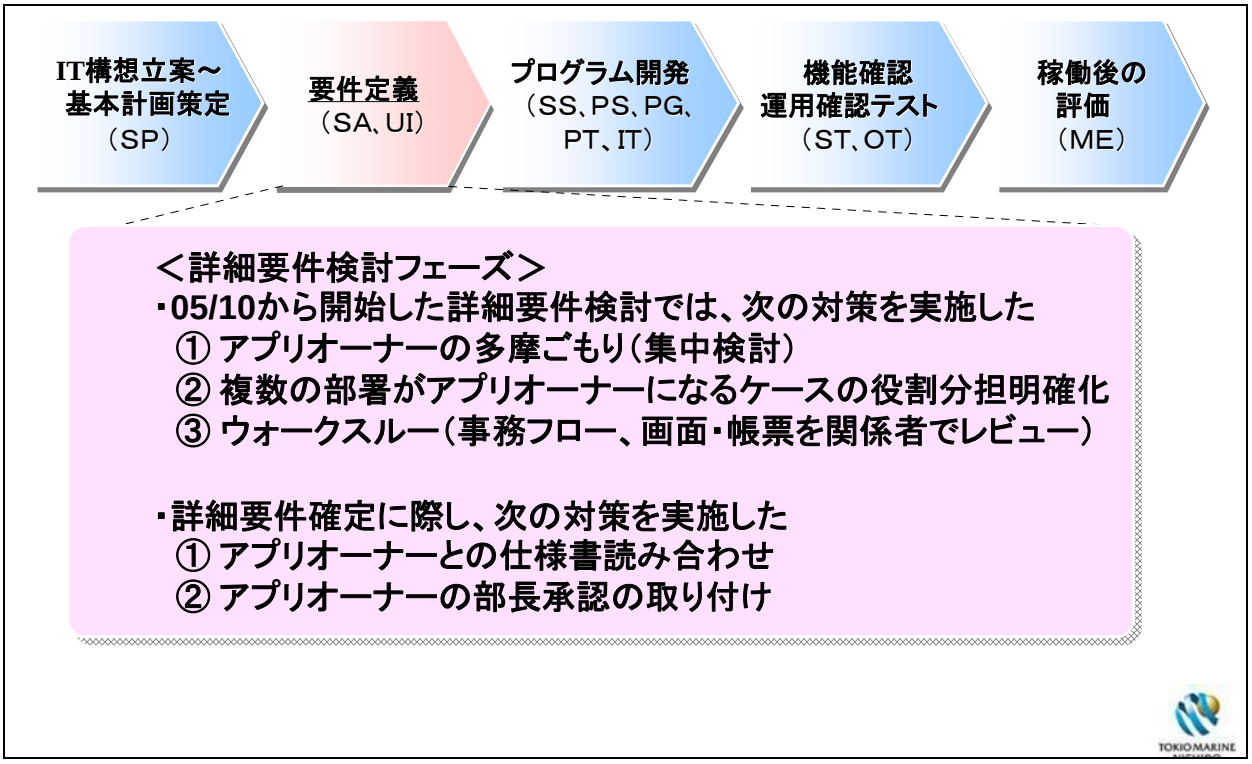


図 3-5. 要件の適正化に向けた取り組み－「要件定義」工程－2

そして、詳細要件確定に際しては、漏れ、認識齟齬や後戻りを防止するために、アプリオーナーと詳細要件仕様書を読み合わせた上で（結構時間を要しました）、組織間でしっかり合意するためにアプリオーナーの部長承認取り付けを行いました。

3．5．3．プログラム開発工程

このようにして詳細要件が確定しましたので、2006年4月からプログラム開発工程に突入しました。

通常であれば、一気に開発を進めていく段階ですが、この工程でも、アプリオーナーから多くの要件追加・変更がありました。

その代表的なものが、保険料率改定です。保険料率は 1 つの商品（例えば自動車保険）について数年毎に改定されますが、通常は 1 つのシステム開発案件としてノミネートされます。しかし、今回の抜本改革は長期に亘るプロジェクトであり、その期間中に、保険料率改定の必要性が生じてしまった、というわけです。適正な保険料水準を維持するという観点で、対応しない、あるいは時期をずらすという選択肢はありません。

そして、プログラム開発工程における要件追加・変更のもう1つの要素は、アプリオーナーによる要件細部の確認です。オーナーが多摩にこもって要件を詰め、ウォークスルーや仕様書の読みあわせをして要件を確定させたわけですが、オーナーが保険商品の各種規定や事務処理のイレギュラーケースを考える段になって、一旦決めた要件では不都合

な部分が発見されました。商品や事務を一から作り直す大改定であり、要件定義の工程だけで決め切れなかった、ということです。

保険料率改定の必要性は「長期プロジェクトである」が故に、アプリオーナーによる要件細部の確認は「商品、事務を再構築する」が故に、プログラム開発工程にも拘らず要件追加・変更が発生することになりました。

この要件追加・変更を対応するために、「追いかけ開発」を行うことにしました。当初予定していなかった本開発との二重開発であり、要員・ロードの問題、プログラムの世代管理の問題などがありましたが、アプリオーナーテストが開始するまでに本開発に追いつくという目標の下、鋭意開発を進めていきました。

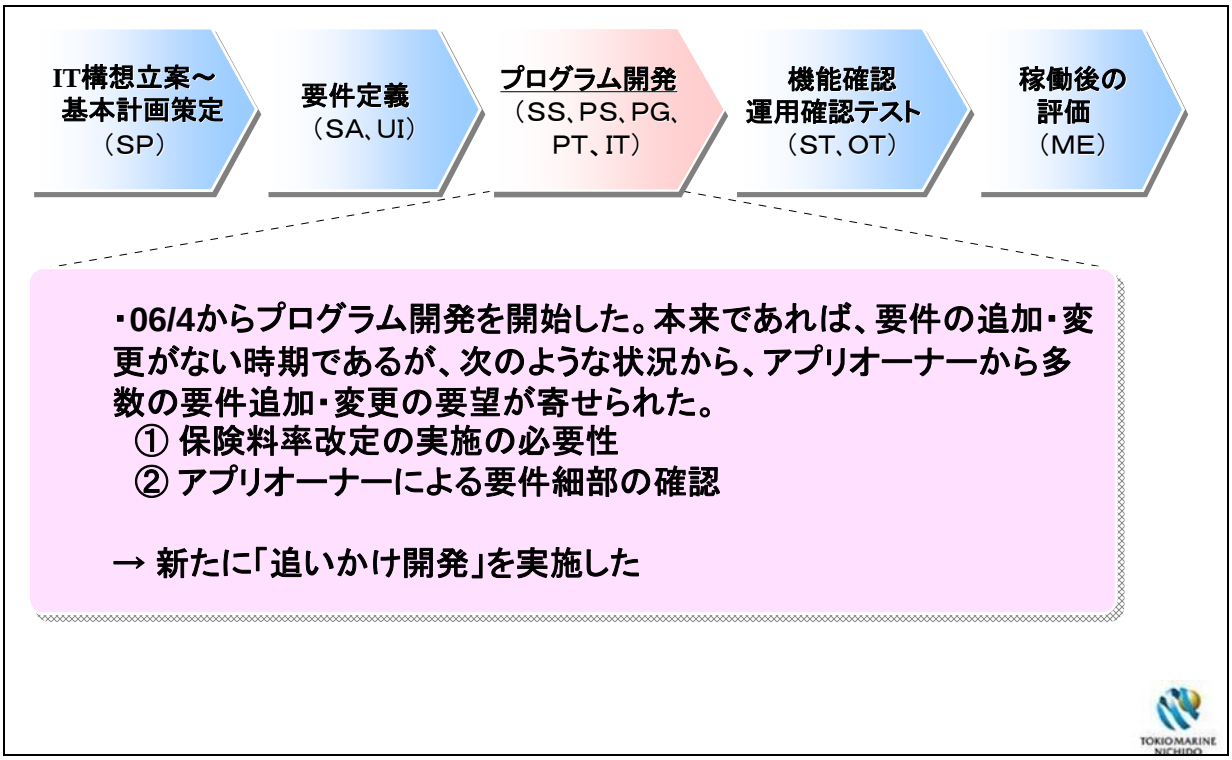


図 3-6. 要件の適正化に向けた取り組み－「プログラム開発」工程

3. 5. 4. システムテスト工程

2006 年 7 月から機能確認のためのシステムテストを開始し、それと並行して実施した追いかけ開発も無事終了し、本開発と追いかけ開発を合流させて、2007 年 2 月からアプリオーナーテストを開始しました。

テスト工程において、一定のバグが発見され要件追加・変更も少しは発生すると想定していましたが、想定以上の要件追加・変更が寄せられました。その要因は、アプリオー

ナーとシステムズの要件認識の齟齬です。アプリオーナーテストを実施してみると、「そういう認識ではなかった」といったたぐいの話が散見されました。その要因の1つが、要件検討時に「現行システムと同じ」と定義したことです。商品や事務のベースが変りシステムも一新しているわけですから、本来「現行システムと同じ」はあり得ないはずです。膨大な要件検討の中で安易に決めたツケが回ってきた格好になりました。

また、テスト工程で要件追加・変更が多く発生したもう1つの要因は、サービスインが近づくに連れて、アプリオーナーが保険商品のガイドブックや事務処理マニュアルを作成していく段階で、一旦決めた要件では不都合な部分を発見した、というものです。

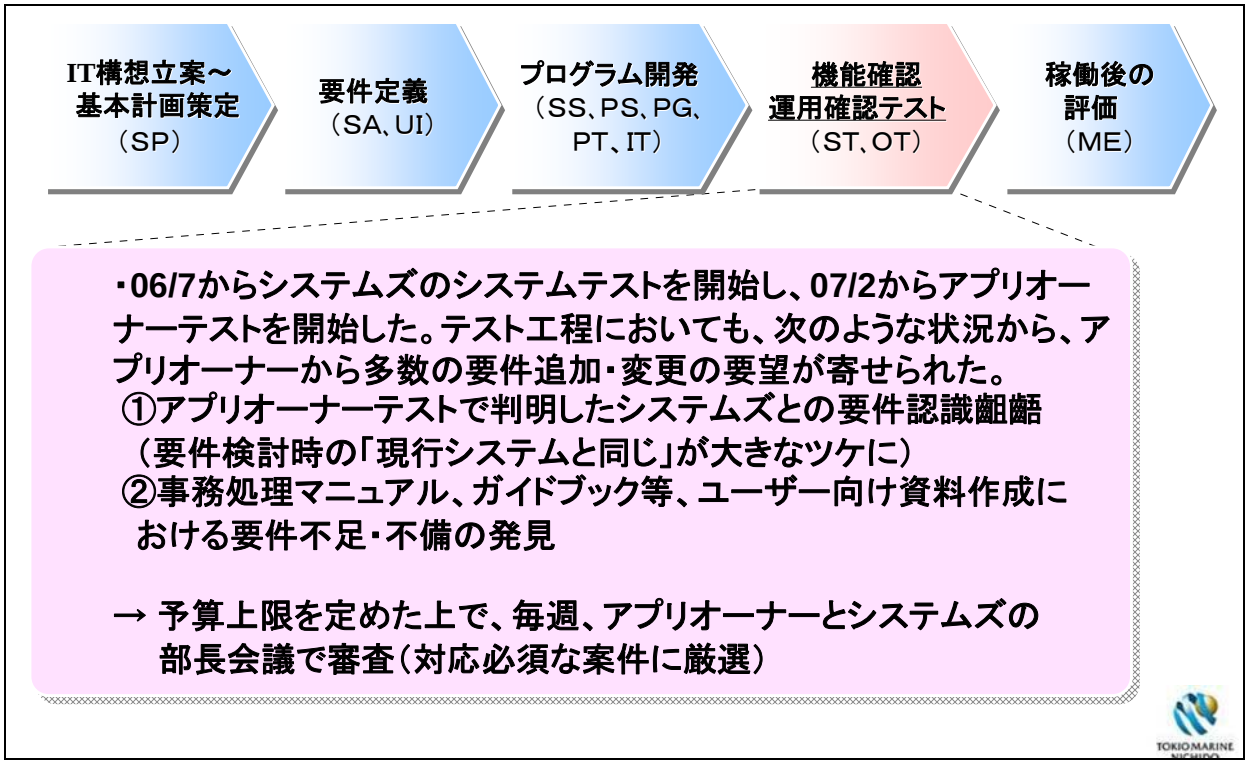


図 3-7. 要件の適正化に向けた取り組み－「テスト」工程

もう、追いかけ開発ができる時期は過ぎていました。1つ1つの要件を精査し、必須なものに厳選して対応していくことにしました。そして、この「要件を精査し、必須なものに厳選する」ことを実践するために、次の対策を講じました。

1 つは、追加予算の額を決めたことです。予算管理＆要件審査の機関を設け、案件内容やコストの審査を受けることにしました。2 つ目は、この要件審査に上げる前にアプリオーナーとシステムの部長会議で協議したことです。「実施しなければ営業活動や事務が混乱し代理店さんや社内に説明レターを出すレベル」の案件に限って対応する、といった基準をこの部長会議で設けました。そして、画面や帳票の文言 1 つの変更に至るまで、全ての要件追加・変更案件を、毎週の部長会議で協議しました。（各部長は、相当大変

だったと思います。）

この 2 つのスキームによって、遡上に上がってくる案件が絞られてきましたし、協議・審査を経て実施する案件が精査されました。サービスインを間近に控えシステムを凍結していく時期であり、円滑なサービスインに向けて効果的なスキームだったと思います。

3．5．5．稼動後の評価の工程

最後に、「稼動後の評価」の工程での取り組みについて、ご説明します。

これまでご説明した経緯を経て、2008 年 5 月に抜本改革の基盤システムと自動車保険システムがサービスインしました。

サービスイン後も、実際にシステムを利用している代理店さんや社内から様々な改善要望が寄せられています。従来ですと、各アプリオーナーがそれらを取りまとめてシステム開発案件化するわけですが、今回は（2008 年度の開発計画では）、抜本・自動車保険システムの改善要望に一定の開発工数枠を設け、アプリオーナー単位ではなく全社単位で優先順位を付け、開発工数枠の範囲内で開発を行うようにしました。
また、自動車保険システムの開発での反省を活かし、火災保険システムなど抜本改革の次のプロジェクトにおいて、次の 5 つの取り組みを実践しています。

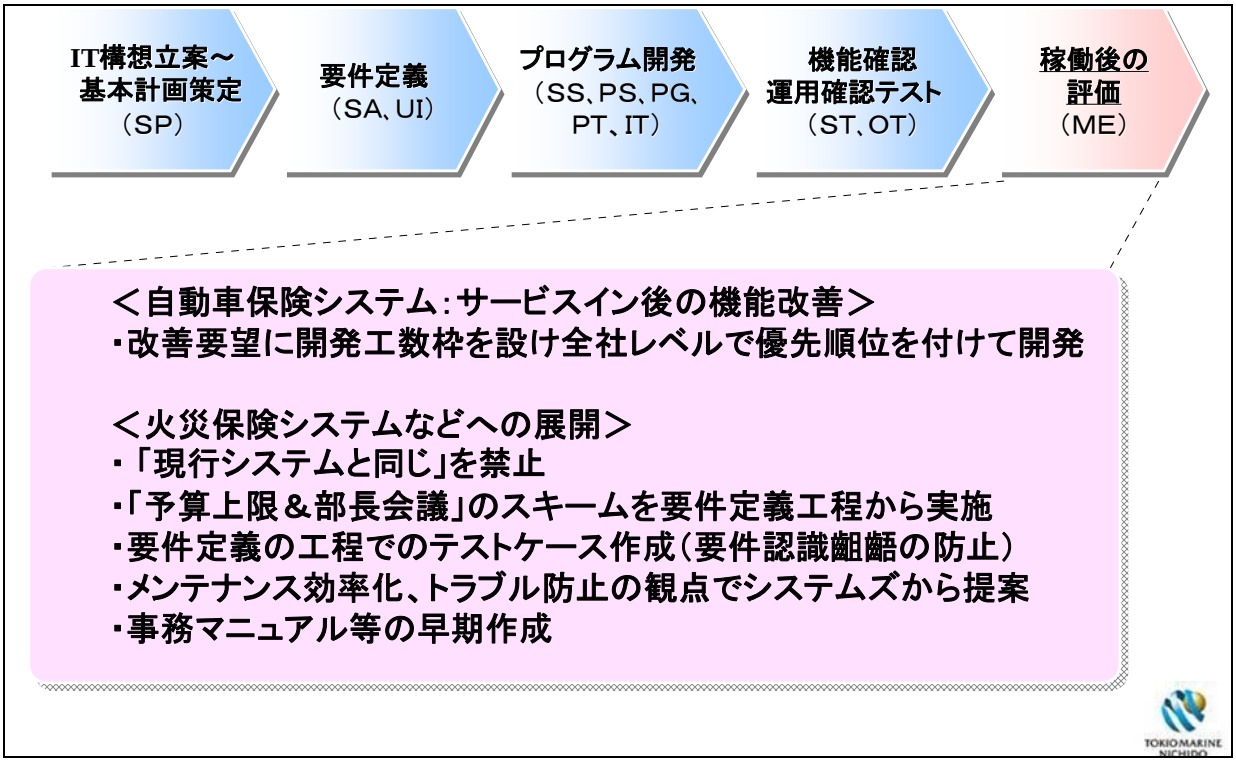


図 3-8．要件の適正化に向けた取り組み－「稼働後」工程

まず、1 つ目は、要件検討時の「現行システムと同じ」の禁止です。膨大な要件検討に

おいて大変ではありますが、1つ1つの要件を具体的に定義することで、アプリオーナーとシステムズの要件認識齟齬防止を図っています。

2つ目は、自動車保険システムのテスト工程で実施した「追加予算上限を定めた上でアプリオーナーとシステムズの部長会議で要件を精査する」スキームを要件定義工程から実施し、要件の肥大化防止を図っていることです。

3つ目は、要件定義工程で、アプリオーナー、システムズ双方がテストケースを作成することです。双方のテストケースを突き合わせることで、要件の認識齟齬の早期発見を図っています。(プログラム開発工程以降の手戻りが減少しテスト工程がスムースになる、という効果も期待できます。)

4つ目は、自動車保険システムのサービスイン後に発生したトラブルを教訓にして、メンテナンスを複雑化する要件、トラブルを誘発するような要件を洗い出し、それらの要件のシンプル化を図っているということです。自動車保険システムの改定時に対応していくこと、火災保険システム等を開発する際の要件定義に反映していくことに努めています。

5つ目は、保険商品ガイドブックや事務マニュアルの早期作成です。細部やイレギュラーケースの要件を早期に固めるために、アプリオーナーに早期作成を要請しています。

以上が、抜本改革プロジェクにけるシステム要件定義において、発生した状況やそれらに応じて実施した対応策です。途中、何回か触れましたが、今回発生した状況やそれらに対する対策は、抜本改革プロジェクトの特徴である

・巨大システムの再構築であり、膨大で、かつ細部に亘る要件定義が必要であること
・システム開発期間が長く、外的要素も含め、期中で様々な要件追加・変更の要素が生じること

に起因しています。

当たり前のことではありますが、それらに関するリスクを早期に洗い出し、関係者で共有し、適切な対応策をしっかり講じていくことが重要です。また、次回のプロジェクトに活かしていくことも肝要だと思います。

抜本改革システム開発における要件定義の進め方については以上ですが、関連することとして、最後に、システムズが考える「SEの本分」について、ご説明しておきたいと思います。「SEの役割はシステムだけを構築するだけではない」ということです。

(1)システムはプロセスを実現する手段であり、システムを作るのではなくプロセスを作るのだ
(2)現場・お客様のニーズを捉える感性やプロセスを分析できる力が必要だ
(3)システムを知らないと「いいプロセス」は作れない。だからシステムスキルも重要だ

と考えています。

プロセスをよく知っているシステムの専門家こそが、新ビジネスモデルを構想できるのだと思います。システムエンジニアではなくビジネスエンジニア。システムズでは、こういう人財を育成していくことを目指しています。

4. 知恵と勇気の要件獲得？プロセス

スミセイ情報システム株式会社　小浜　耕己

4．1．はじめに

要件定義は難しいとよく言われる。ここに失敗することが、プロジェクト破綻の第一の要因であり、成功例よりも失敗例のほうが多く見受けられるとまで言われることもある。だからと言って、失敗してもかまわないとは誰一人思っていないはずである。ITC プロジェクト成功のためには、顧客企業のビジネス実現のためには、即ち ITC 業界が今後健全な発展を遂げるためには、要件定義工程を成功裡に終わらせるしかない。

ではどうすればうまく行くのか。結局のところ、基本通りのことを愚直に、ねばり強くやり抜くしかないというのが結論であるが、われわれ情報システムに携わる者は、難しいからと投げ出すことなく、知恵を絞り、勇気を持ってそのことに取り組む必要がある。これは、避けて通れない道であると言える。

そこで本稿では、そもそもなぜ要件定義工程が難しいと言われるのか、どうすればうまく行くのかを、若干の事例を交えながら論考することとしたい。

4．2．要件とは何か

まずは、基本に立ち返って、要件とは何かを考えてみる。

われわれ IT 屋は、業界内での習慣や思考法に囚われて、視野が狭くなりがちだと言われる。しばし日常業務の枠から離れ、別の世界の用語から、「要件」という言葉を考えることから始めてみよう。

4．2．1．犯罪の構成要件とは

法律学・犯罪学の世界では、以下のように犯罪を定義する。

犯罪とは、<u>構成要件</u>に該当し、<u>違法</u>・<u>有責</u>な行為である。

ここで、問題の「要件」という語が用いられている。簡単に言えば、法律の世界で言う「要件」とは、あらかじめ成文によって定められた法律、つまり刑法などの条文のことを示す。わが国をはじめとする多くの近代国家では、「罪刑法定主義」の立場を取っており、刑法典に定められていない行為は、いかなる悪事であろうとも、そもそも犯罪の要件を満たさない。

例えば、有印私文書の偽造は、法典に定められた犯罪である。一方、いわゆるコンピュ

ータウィルスの作成・配布は犯罪ではない。現在施行されている刑法が、そのような「悪事」の存在を予定しておらず、従って法文の定めがないためである。必要性が叫ばれて久しい「ウィルス作成罪」の法制化が成るまでは、悪意を持ったウィルスの作成・配布という行為は犯罪ではなく、著作権に関わる法律や業務妨害に関わる法律などで間接的に裁くしかないのが現状である。

法律の世界では、「法典に書かれたもの」が要件と言うことになる。その法典をひもとくと、多くの「犯罪」では、その行為の意図が問題になっていることがわかる。一例として「詐欺罪」を引くと、いわゆる「二段の故意」が要件となっている。煩瑣な法律用語を避け、日常用語に翻訳すれば、「人を騙そうという故意（積極的な意思）」と、「その行為によって利益を得ようとする故意（積極的な意思）」が、犯罪の成り立ちのために必要なのである。

しかしながら、犯罪の成立のためには、要件充足は充分条件ではない。

さらに、「違法性」「有責性」が問題となる。逆の言い方をすれば、違法性、有責性を損なう事由（阻却事由）のないことが、犯罪成立の条件になる。例えば、自身のまたは他人の身を守るための「正当防衛」や、社会的必要性に基づいた「正当行為」が、違法性阻却事由とみなされる。襲いかかって来る暴漢を投げ飛ばした結果、あるいは外科医が医療行為として手術を実施した結果、傷を負わせたとしても傷害罪にはならない。また、責任能力のない未成年者や障害者が犯した行為については、有責性が阻却され、犯罪にならないことがある。

4．2．2．犯罪の構成要件を情報システムに当てはめると

さて、本題に戻り、ICTの世界について考えてみる。法律の用語に倣って定義しなおすと、よい情報システムは、以下のようなものと言える。

<u>要件</u>を充足し、<u>妥当</u>・<u>有効</u>な仕組み

法律上の「要件」では、「二段の故意」を要した。情報システムでも同様である。「要件」には、「こういう仕組み（システム）を作ろうという意思」と、「その仕組みにより、ビジネス上の利益を得ようとする意思」が二つながら組み込まれていなければならない。

さらに、犯罪の定義に倣って言えば、「よいシステム」たり得るには、「要件」として明示されていないかも知れない「適法性」や「有効性」が実現されている必要がある。「適法性」とは、社会的要請や法律、社内規範、標準化といったものに求められるコンプライアンスであり、「有効性」とは、例えば投資対効果、設計妥当性などである。法典ならぬ「要件定義書」に明示されていなかったにせよ、これらは、広義の「システム要件」に含まれると考えら

れる。

4．3．要件定義はなぜ難しいのか

思うに、要件定義の難しさは、ここに起因する。

4．3．1．意思が見えない

第一に、あるはずの「意思」が、なかなか明示されない。特に、二段目の意思、即ち、「ある仕掛けを使って、どのような果実を、いつ、どれほど得たいのか」という意思がない、あるいは隠されているように見えることが多い。

RFP はあっても、この「二段目の意思」は、必ずしも明示されておらず、明示されていたとしても、「業務効率の向上」「使い勝手の向上」「情報のリアルタイムかつ一元的な管理」といった抽象表現に留まるケースが多い。何かを実現するためには、何かをあきらめる必要があることも多いが、この後段部分は、制約事項として明記されないことも多い。

そして、明示されなかった「意思」を、「要件定義工程」の中でヒアリングしようとしても、なかなかうまく行かない。

ひとつには、「聞く相手を間違っている」ということもある。ユーザ部門であっても、事務処理担当者は、「自分の仕事を楽に、円滑に進めたい」「慣れたやり方を変えたくない」と考えていることが多く、ビジネス上の目標達成には興味が薄い。そのような相手と「要件」を詰めようとしてヒアリングをかけても「お悩み相談室」になるのがオチである。

4．3．2．知識・経験の不足

第二に、ユーザの知識や経験の不足がある。要件を必要機能に分解していくに際しても、「二段の意思」の確認は重要である。何のための業務であるのか、どうすれば改善できるのか、また、コンプラ上問題はないのか、改善効果は投資に見合うものであるのかを確認するには、業務全般に対する深い理解、コスト感覚、法制を含む広範な知識、システムの運用やリスクに関する理解、情報活用の経験や感覚などが必要となる。必然的に発生し、システム構築上大きなインパクトを持つ例外事象への目配りや、大局と細部のバランス感覚、構想をまとめる論理的思考力も重要である。

日常的にBPRを検討する機会などないのが普通であるから、一般的なユーザに、それを期待するのはいささか無理であって、情報活用の専門家の助力が必要となる部分である。

4．3．3．錯綜する意図

第三に、ユーザ・発注者と言えども一枚岩ではない。情報システムに限らず、住宅を一

軒建築する場合を考えても、ひとくくりにユーザと位置づけられる舅・姑、その息子、嫁、子供のそれぞれに思惑があるであろう。スポンサーである舅は、極力建築コストを抑えたいと考えているかも知れない。姑は夫亡き後の同居生活の快適さを思うはずだ。息子は自分の趣味を、嫁は姑の束縛からの独立を、子供は親の干渉の排除を考える。即ち同床異夢のユーザを想定する必要がある。

情報システムもこれと同じである。ビジネス実現を考える経営層や企画部門、コスト削減をミッションとする購買部門、内部統制を担当する監査部門、不整合を嫌う経理部門、変化を好まない事務部門など、ユーザといえども同床異夢の多様な要求を出しがちで、要求相互にコンフリクトが発生する。

4．3．4．要件定義とは

となると、「要件定義」とは、一部が隠されていたり、曖昧だったり、誤っていたり、相互に矛盾をはらんだ状態にある「要求」を、実現性のある整合した「要件」に定義し直す工程ということになる。

4．4．要件定義に必要なもの

前節の定義に従えば、要求には抜けや嘘、矛盾が内包されており、真意が明示されていないことも多いということになる。であれば、必然的に、要件定義工程では抜け、嘘、矛盾、不明、曖昧さを検知し、校正する行為が必要になってくる。そして、一定規模以上のシステムでは、その全てを誰かの勘や経験に頼るわけにはいかない。

となると、抜け・矛盾を検知し、校正するための道具や方法論が求められることになる。

4．4．1．道具の例：WBS と DFD

スミセイ情報システムでは、若手中心に、要件定義工程の標準 WBS(Work Breakdown Structure)を策定した。要件定義工程のスケジュールを策定する場合、よく陥る過ちに、「ヒアリングのスケジュールしか引けない」というものがある。業務要件を確認するために、ユーザにヒアリングするスケジュールだけを決めてこと足れりとするものだ。

そこで、「とりあえずヒアリングをすればよいという徒手空拳の取り組みではうまく行かない」という問題意識が生じた。この問題意識に基づき、例えばヒアリング対象者と順序の選定、ヒアリングに当たって用意すべきドキュメントとヒアリング結果の成果物、レビュープロセスやユーザとの合意形成のプロセス、コンフリクト解消のための仕掛けなどを、言語や開発ツールに依存しない形で定義しようとしたものが、「要件定義 WBS」である。

言語や開発ツールに依存しない形とは言え、WBS として機能させようとすれば、相当程度の具体性が必要となる。若手 SE が、抜け・矛盾の検知と合意形成のために担ぎ出して来た「道具」に、筆者は一驚した。それが、過去に置き放され、既にいい加減埃を被っているものだと思っていた「DFD」－－－データフローダイアグラムだったからである。

しかしながら、彼らと議論しているうちに記憶が蘇り、「なるほど」の念を強くした。確かに、厄介な「要件定義」工程で、わがままで矛盾に満ちた「要求事項」の交通整理をしようとすれば、こういったツールが有効になるのである。

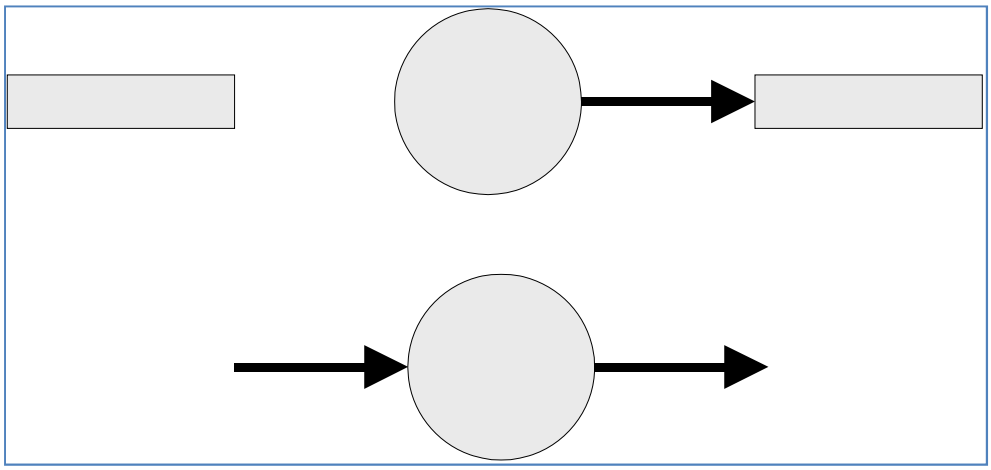


図 4-1. DFD の例

図 4-1 は、提起された要求を記述する DFD の例である。この記述法では、長方形データストアを、円がプロセスを、矢印が情報の流れを示している。この例では、ダイヤグラムで記述することによって、以下の矛盾が露呈している。

(1)参照されないデータストア
(2)情報なしに実施されるプロセス
(3)情報をスルーするだけのプロセス

溜め込まれるだけで参照されないデータストアは、要求の不完全性を意味する。ヒアリングした相手が、例えば伝票起票には熟知しているが、勘定元帳を参照する業務は経験していないような場合、このような記述になる。

情報なしに実施されるプロセスは通常あり得ないため、何らかのデータストアが未だに隠されていると考えることができる。仕入れ担当者が発注先を決定するプロセスで、既存システムを参照せず、自分の頭の中の記憶や、デスクの抽斗にしまい込んだ「業者別仕切値実績ノート」といった隠された情報に基づいて業務を行っている場合などに、こういう

見え方をすることが多い。当然ながら、正しいヒアリングは、「業者別仕切値実績ノート」をあぶり出し、既存システムの限界を明らかにするものでなければならない。

スルーするだけのプロセスは、プロセスが廃止可能か、必要情報が隠されていることを意味する。物理的制約から、「伝票を梱包し直してから発送する」などの業務がこう表現されることが多い。だとすれば、伝票を電子化し、物理的搬送をなくすことで、不要となるはずのプロセスである。

4．4．2．論理モデルによる抽象化

DFD は、論理モデルとして抽象化するための記述である。抽象化することによって、物理的で複雑な業務を整理し、「情報の流れと利用」に特化した図形として表記する。抽象化することで、論理矛盾を発見しやすくなる。また、物理的な制約を廃した実現策を検討することができるようになるのである。

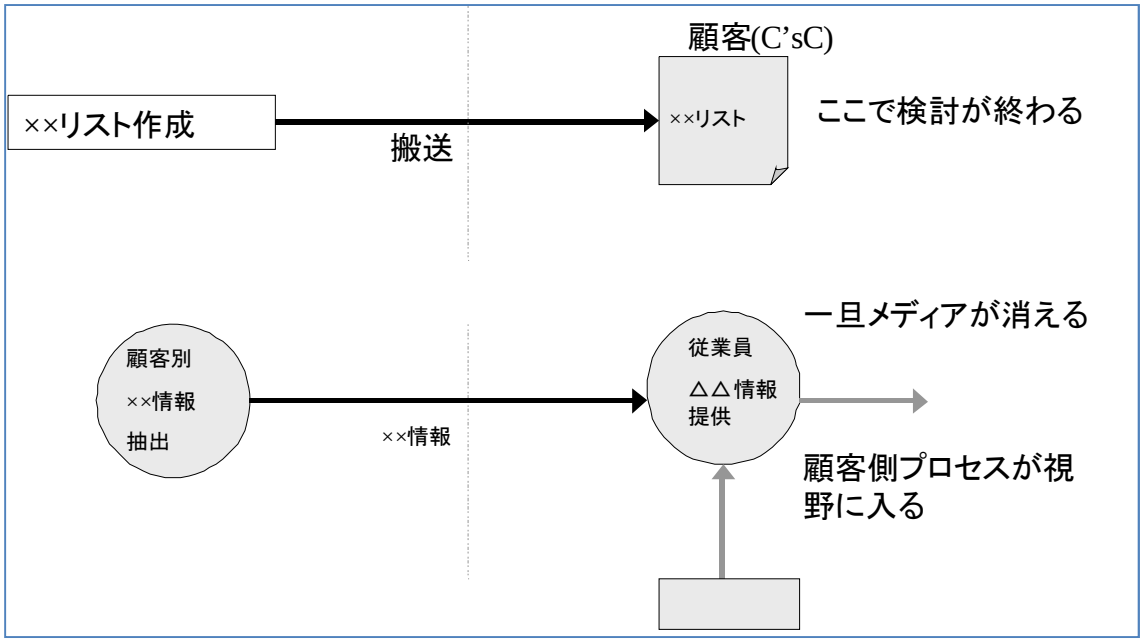


図 4-2．抽象化の実例

また、抽象化することで、解決策を検討する上での制約を外しやすくすることもできる。図 4-2 は、過去の実例である。

リストを作成し、搬送するという業務を、物理的に捉えてしまうと、その効率化策にはおのずと制約が科される。リスト作成サイクルの見直し、搬送速度の改善といったあたりで、思考停止に陥ってしまう。（図 4-2 の上側の流れ）

物理的なメディアを一度追い払って、同じ業務を抽象化すると、ある情報があるプロセスに引き渡されるというシンプルな姿になる。そのプロセスのインプット、アウトプットを表記

しようとすれば、真に重要な改善対象が見えて来る。（図 4-2　下の流れ）

実例では、当初、リスト作成サイクルを見直そうとしたが、論理モデルから、ファクシミリ自動送信などのアイディアが出て、最終的には顧客事業所への情報検索ターミナル設置という実現方法で決着した。物理モデルで ASIS を見ていてはなかなか見いだせない「真のニーズ把握」と、「解決策」である。

4．5．顧客との共有・合意形成

若手 SE の選択から教えられた通り、要件定義工程で使用するツールは、抜けや矛盾の検知を助け、より高次の合意形成に導くことのできるようなものでなければならない。しかしながら、ここで一つ「壁」があると言われる。次のような「常識」が、設計者の頭の中に抜きがたく存在するのだ。

＜「ベンダー」が、(つい)言いたくなる泣き言は---＞

・顧客がプロジェクト目標を明確にしていない
・顧客が要求事項を知らない
・顧客が要求事項をまとめようとしない
・顧客要求事項は大雑把すぎて見積りができない
・顧客の要求事項はひっくり返る
・顧客の要求は際限なく拡大する
・顧客の要求事項は矛盾に満ちている
・顧客がスケジュールを守ってくれない
・顧客が金額や工数の提示を要求し、それで縛ろうとする

正しい要件定義やユーザとの合意形成などきれい事ーーーそんなぼやきを聞くことは少なくない。しかしながら、こういった「あきらめの姿勢」も、ユーザから見れば別の「不満」として見えるはずだ。

＜「ベンダー」の問題として捉えると---＞

・顧客のプロジェクト目標や利害得失に無関心
・自社のプロジェクト目標や利害得失に無関心
・要求事項がはっきりしていないとわかっているくせにそのまま受注したがる
・顧客が要求事項を引き出す・まとめることに有効な協力ができていない
・要求事項変動や仕様拡大、スケジュール遅延に対する策を持たず、看過している
・顧客のわがままを許している

・要望があれば闇雲に作りたがる
・プロジェクトの QCD(品質、コスト、納期)に対する執着・厳しさが足りない

4．5．1．建設的な役割分担

お互いに不満をぶつけあっても建設的でないことは自明である。効果的に要件定義を進める上で、何かが足りないのは明らかなのだから、お互いの役割を見直してみるとよい。

例えば、「データベース設計」において、顧客はどんな役割を担うべきだろうか。

以下は、あるプロジェクト立ち上げ初期の、開発者とユーザの問答である。

開発者	「管理したいのは何ですか」
ユーザ	「お客様です」
開発者	「お客様とは誰/何のことですか? 既顧客ですか見込客ですか、両方ですか?」
ユーザ	「見込客も含めて管理したいのです」
開発者	「では、コンタクトが薄く、情報に欠落がある段階からの管理ですね。受注や精算と紐づけをしたいのですか」
ユーザ	「そうです」
開発者	「お客様は個人ですか、法人ですか、世帯ですか?」
ユーザ	「実は、世帯として見たいのです」

このような問答があるということは、「ERD」を誰が書くかという問題はともかくとして、「ユーザ」が、データベース設計に参画しているということに他ならない。とかく曖昧になりがちだと言われる「非機能要求」について、ユーザは以下のように提示する。

『この～なんというか～画面はこの、キビキビと、パッパッパッパッって感じで出て来て欲しいんだよね』 ―――某社某部長による「非機能要求提示」

「要件定義」は、ここから始まる。曖昧・不正確な「要求」を嘆くのではなく、ここから、実装に持ち込めるところまでもって行けばいい。既存システムの「キビキビしていない」画面はどういうものなのか、それによって、業務がどうなっているのか、どういうユーザが、どういう局面で、どういう理由で、『パッパッパッパッ』を求めているのか――。

多くの場合、ユーザは、「関わりたがらない」のではない。何を提示すればよいのかわか

らなかったり、提示するための「言葉」を知らないだけなのだ。

4．5．2．進むべき道は？

進むべき道は、ある意味明らかだ。

要求に不備・不足があるから難しいというのなら、それを明らかにするような方策を講じる。

ユーザに知識、経験が不足しているというのなら、（実装技術の専門家としてではなく）情報活用の専門家としてそれをサポートする。また、それができる人材を育成する。

要求相互にコンフリクトが発生するから難しいのなら、コンフリクトを解消するように仕掛ける。例えば、最近設置されることの多い舵取り委員会（ステアリング・コミッティ）は、有効なツールに成りうる。

4．5．3．ユーザ参画に向けてできること

自戒を込めて問うが、われわれIT屋は、情報システムの専門家と自負しながら、やるべきことをやっているのだろうか。「要件定義なんて」とあきらめる前に手を尽くしているだろうか。まだまだ、やり残したことがあるように思われるのだが。

例えば、最近、ユーザとブレインストーミング合宿をしたことがあるというSEが、どれほどいるだろう。KJ法、KT(Kepner-Tregoe)法などを駆使して、CPSを仕切ったことのあるSEが、部下の中にどれだけいるのか。改めてそう考えてみると、忸怩たる思いを禁じ得ない。

かつて、埃を被らせる前に、筆者もDFDを書いていた。それをレビューしていたのはユーザの事務設計者だった。考えてみれば、もともと抽象化が得意なのは、実装屋上がりのSEではなく、むしろ事務企画者としてのユーザなのかも知れない。

われわれは、要件を整理する過程でありがちな抜けや漏れ、不整合、無駄やムラについて、情報活用の専門家として、どこまでユーザをガイドできているだろうか。

また、「ベンダー側」のPMは、ユーザ側に必要なカウンターパートナーや、コンフリクト解消の仕組みを常に要求しているだろうか。その必要性を説明できているだろうか。

少なくとも筆者は、「全てできている」と言い切る自信がない。

であれば――。

要件定義が「難しい」理由がいろいろあったとしても、情報やツールをユーザと共有しての要件定義ができないはずはない。そういう信念を持つことから再度、始めるべきではあるまいか。

「ユーザは投資する気になっている。目的・目標に立ちかえって議論しようとすれば、失注するだけだ」
「この案件は、下流(実装さえできればよい)のものだ。BPR は関係ない」
「そもそもユーザが目標定義できないのだから、われわれがどう言ったところで---」
「納期があるのだから、とっとと作ってしまうしかない」

そんな愚痴は、そろそろやめにして、基本に戻る道を選びたい。「この素晴らしい仕事」を後輩に引き継ぐ責任を、われわれは負っているのだから。

4．6．スミセイ情報システムの方針

まだ不十分なところはあるが、筆者の所属するスミセイ情報システムでは、以下の方針で臨もうとしている。

(1)要件定義工程作業の明確化

例に挙げた WBS の例にもあったように、顧客の役割を含めた要件定義工程の作業とその目的を明らかにする。

(2)要員育成力の取り戻し

要員を現場から教育担当部門に異動させ、実務を経験させながら計画的に育成する制度を昨年から始めた。まずは PM 人材からスタートして、今後は SE 等への展開を考える。

(3)プロジェクト計画へのこだわり

プロジェクト計画、わけてもプロジェクトの目的・目標にこだわり、それを管理できる仕組み作りに執着する。

なるほど要件定義は難しく、その困難さに起因する失敗も多い。しかし、本質と基本に立ち返って、姿勢を正す余地は、まだわれわれには多く残されているはずだ。
まだあきらめる気はない。愚直に、本質にこだわりながら、進んで行きたいと思う。

5. 日本におけるITガバナンスの取り組み

株式会社アイネス　菊島　靖弘

5．1．はじめに

要求工学の研究対象である要求は、その多くが、システムを必要とする一般事業会社のニーズから発生します。今回はこの一般事業会社のシステム活用に係るITガバナンスの歴史と、様々な事業分野でシステムの利活用が進む中で行われた国のITガバナンス政策について、経産省の事例を報告したいと思います。

さて、システム開発はシステムを作ることを目的として行うものですが、本当の目的はシステムを使って実現する業務処理・事務処理、商品またサービスそのものにあります。つまり要求は開発するシステムを使って実現する業務処理、事務処理、商品、サービス及び出来上がったシステムの運用・利活用に係るライフサイクルから生まれてくるわけです。

ITガバナンスは、システムの開発・利活用に係るライフサイクル全てにわたった健全活用力強化策ですから、まさに要求の内容と質に大きな影響を与えるものと言ってよいでしょう。

ところで、ITガバナンスは、SOX法(日本ではJ-SOX法)以来、高度にシステムによって支えられた企業においてのコンプライアンスを中心とした、ITコントロールの範囲で語られることが多いのですが、コンピュータを持たない会社において「コンピュータを使ったシステムを利用して発展しよう」と言う企画をすることが、ITガバナンスそのものであることで分かるように、コンピュータがない時代から行われてきました。

企業はコンピュータのない明治時代には、紙とそろばんと電話と郵便で、20世紀初頭には、機械式会計機を使って情報を処理し、今日コンピュータを用いたシステムで実現している業務の効率化、革新に取り組んでいたのです。

要求が発生するビジネスの場から歴史的にITガバナンスを見ると、システム開発に係る要求の実像、性格の一端が見えてきます。早速、ITガバナンスの視点から要求の発生場所を見ていくことにしましょう。

＜一言コラム＞

SOX法で要求されている多くの事項は、本質的には全て人間系で処理されている場合でも守らなければならないことです。業務の可視化は人が処理を担っていても、コンピュータで行っていても必要なことです。

5. 2. 民間（損害保険会社）の IT ガバナンスの歴史

次の年表は、1879 年、日本で最初に設立された損害保険会社である、東京海上日動火災保険会社（以下、東京海上日動）の IT 化の歴史です。

1930 年頃についての記録を見ますと、当時米国支店において IBM 社製機械式会計機を活用していたことが分かります。

東京海上日動は、機械式会計機を用いて、今日と同じように、損害保険がいつ、どこで売れたか、いくらの保険料収入があったかを集計していました。本質的にはこの範囲の会計業務に関しては、今日でも基本は同じです。最新式の POS システムも、何が、いつ、どこで、どれだけ売れたかをつかみ、事業の効率化に活用している訳ですが、この範囲に限ってならコンビニビジネスにおけるニーズも本質的には同じです。もっとも、ニーズは同じと言っても、要求となると、その時代によって変わってきます。要求はその時代の最新技術に対応して進化するものなのです。新しい技術は、新しいビジネスの革新を実現しますが、革新を実現するための要求は、技術に加え、実現可能な範囲で経営方針や、経営リソース、ステークホルダの立場等の制約を受けながら決まってくるのです。

話を歴史に戻しましょう。

その後、年表にあるように、1961 年に IBM1401 コンピュータが導入され本格的コンピュータを活用したシステム利用が始まりました。

1. 損害保険会社(1879年創業)の利用事例

機械化の黎明期からネットワークの時代へ

- 1930年頃：米国支店、米国子会社（ｽﾀﾝﾀﾞｰﾄﾞ）にてIBM会計機を活用
- 1953年：統計調査部発足
- 1961年：IBM1401導入（メモリー4K）
- 1962年：事務機械化総合計画
- 1968年：IBM360導入
- 1974年：自動車保険オンラインスタート
- 1996年：ひとり1台
- 現在端末30万台を超えるネットワーク

図 5-1. 機械化の黎明からネットワークの時代へ

次の写真は、1930 年当時の「損害保険契約計上システム」です。

パンチカードに、コード化された、保険種類、販売支店名、販売年月日、保険料等が打ち込まれ、それをソータと言う機械で保険種類別や、販売場所別といった分類に従ってカードを並べなおし、会計機で一枚、一枚読みながら、機械的に保険料を（おそらくカタカタと言う音を出しながら）カウントし、コードがブレイクしたところで集計した合計金額を、カウンタから打ち出すというものであったわけです。

現在、同じことはコボルなどのプログラムによって、コンピュータの中で電気的に行われています（バッチシステムでは「コントロールカットによる集計表作成」といった名前のロジックですね）。

図 5-2. 1930 年頃使われた保険契約計上システム

5. 3. 政府（経済産業省）ITガバナンス政策事例

経済産業省が IT ガバナンスに係るスタンダードを出したのは、1985 年にさかのぼります。このころ、大銀行は第 3 次オンライン、ファームバンキングと言ったシステム開発を進めており、大手損害保険会社も全ての業務をオンラインで処理する総合機械化システムの構築を進めていました。まさに時代のトップランナーはシステム技術を活用した大規模な BPR を進めていたのです。

コンピュータシステムの高度活用は、システムの開発・利活用に係るレベル、品質が企業経営に大きく影響を与えるという状況を生み出し、そのような中で、その内容を監査し

て企業経営の健全化を確保するための監査基準が制定されたわけです。

その後、多くの大企業において不祥事が発生し、米国における SOX 法の制定、わが国でも 2006 年に「金融商品取引法」が制定され、それを受けて IT ガバナンスに係るスタンダードも主に内部統制の強化という観点から改定・強化され今日に至っています。

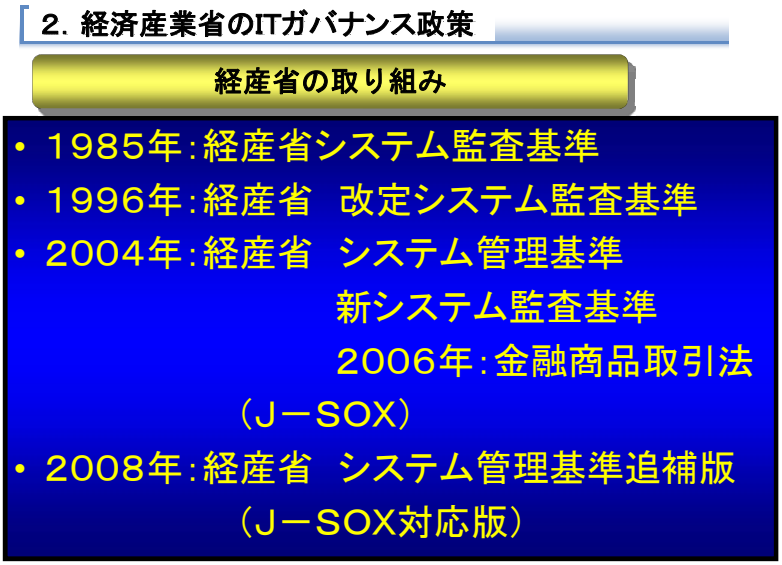


図 5-3. 経産省の取り組み

これらの管理基準、監査基準は経産省の Web[1, 2]サイトから入手できます。

本稿の最後に付録として概要をまとめておきましたが、一度、原本をお読みになることをお奨めします。

また、金融庁や、FISC(金融情報システムセンター)[3]といったところからも優れた IT ガバナンス、IT コントロールの資料が提供されています(一部有料)。

5. 4. ユーザ(一般事業会社)におけるITガバナンス

それでは、ユーザ企業における IT ガバナンスの実際を、東京海上日動の例を参考にしながら見ていくことにしましょう。

(1) 3 つのガバナンスレベル

IT ガバナンス、IT コントロールについては多くの書籍が出版されており、大変、分かりやすくその意味が解説されています。「IT ガバナンスとはなにか？」という議論は、そういった専門書にお任せするとして、ここでは損害保険事業という一般事業会社における取り組みを以下の図の通り階層構造的に 3 つに分け、それぞれのレベルでどういった施策が打たれているか見ることにします。

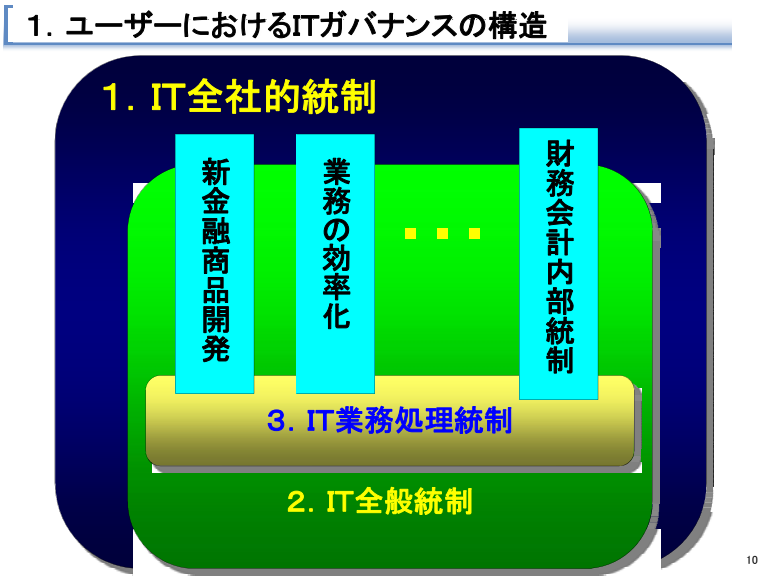


図 5-4. ユーザにおける IT ガバナンスの構造

(2) IT 全社的統制を実現する事例

企業には、システムを活用して業務を行っているところもあれば、人手で行っているところもあります。既に高度にシステムの活用をしている会社でも全ての業務処理がシステムに乗って動いているのはまれで、例えばシステム化の進んだ会社の営業などからも、「システム化して時間を作って客先を訪問できるようにしよう」などと言う企画が出てきます。こういったシステム化企画構想は当然 IT ガバナンスの対象になります。

しかし、IT ガバナンスの視点から見ますと、今時はネットワークに営業の機能を持たせ営業担当者を不要にするシステム化もあります。「会社をどのようにシステムを使って変えて行くか？」と言ったレベルの話については、担当レベルを超えて、経営レベルで行うことが不可欠です。

IT 全社的統制のレベル感は、こういった経営視点でのシステム利活用政策と言えます。

システム利活用にかかる原理原則や、システム開発体制の拡充施策、経営から見た開発対象システムの選択と集中実現の仕組などが挙げられるでしょう。

(a)1961 年策定、東京海上事務機械化 9 原則

ここに、紹介するのは、今から約半世紀前、当時の東京海上（現：東京海上日動）が初めてのコンピュータ導入に当たって策定した「事務機械化 9 原則」です。21 世紀の今日でも用語を現代風に変えれば立派に通用する原理原則といえます。

この原則は今では社内で大きく掲げられることはありませんが、現在の東京海上日動における、アプリケーションオーナー制度や、プロジェクトレビューと言ったマネジメント施策に受け継がれて時代に合った進化をしています。

東京海上1961年事務機械化9原則

- 原則1：
  当社の事務機械化は経営管理の高度化に資することを主たる目的とする
  （事務処理能力の増加、経費の節減にとどまらず、経営管理に役立つ統計資料を迅速に作成し、経営の全般方針および部門方針の迅速な樹立、修正を可能にすることを目的とする。）

- 原則2：
  長期の機械化計画にもとづいて総合的に実施する
  （機械化は将来の業務の発展を十分考慮しなければならないので長期計画を樹てて、段階的に実施する。
  また、変化に対応する弾力性のある計画とする。）

- 原則3：
  機械化の効果は、短期的な採算をみるにとどまらず長期的な採算をも十分考慮する
  （経営の合理化は、機械化なしには考えられない。目先の効果のみにとらわれず、将来の発展という立場から機械を中心とした合理的な経営組織を作り上げる。）

- 原則4：
  機械化に適する業務はすべて機械化する
  （機械化しやすい業務から機械化するのではなく、機械化する業務の選定調整を、全社的にできるだけ統一的に行なう。）

- 原則5：
  機械の購入には、実験費ないし研究開発費的支出を認める
  （機種の選定が不適切という結果をおそれて消極的にならない、実験費的な余分の支出が必要と認める。
  また、機械化当初は事務処理が若干不円滑となることもあるが、これをおそれてはならない）

- 原則6：
  部門ごとに機械化担当のスタッフを組織上明確にする
  （各部門では、統計調査部と連絡を保って、機械化を担当するスタッフを組織上明確にし、その機能を十分果たしうる体制を整えなければならない）

- 原則7：
  機械化の効果を高めるために、事務組織および手続きを根本的に改める
  （手作業を機械におきかえるのではなく、機械を中心とした合理的な事務組織を確立するために、部門意識を排して組織を根本的に改める）

- 原則8：
  機械の処理能力を増強する
  （作業日程、故障、臨時作業等について問題を発生させないため、稼働率を高めるよりも、機械の能力にある程度の余裕を持たせ、事務作業の迅速化と労力の軽減を主とする）

図 5-5. 東京海上 1961 年事務機械化 9 原則

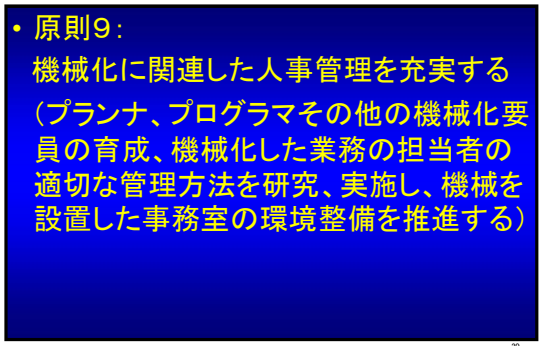


図 5-5. 東京海上 1961 年事務機械化 9 原則(続き)

(b)システム開発体制の拡充施策

保険会社が金融保険専業のシステム開発会社を持つ。

「システム開発はお金もかかるし、重要だという事は分かるけれど、うちの本業ではないよね。とにかくコストセンターであることを意識してやってもらいたい」

という考えの経営層がいまだ多く存在するのが、一般事業会社の現状ではないかと思います。そのような中で、損害保険会社では本業に係る全ての商品、多くの事業計画がシステム開発を伴って遂行されるため、その損害保険事業という本業の QCD が大きくシステム開発力に依存している産業です。

そのため、東京海上日動では、本体のシステム開発部門を分離し、徹底的にシステム開発強化を狙える組織構造を実現しました。

自社のグループに専業の戦略的システム会社を作る。実現業務も、実現システム技術にも秀でた、IT アーキテクト集団を実現しています。

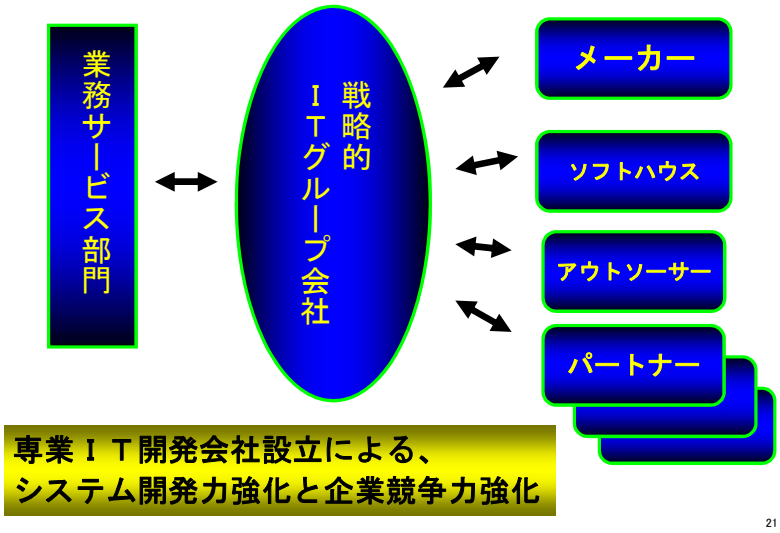


図 5-5. 専業 IT 開発会社

(c)開発対象システムの選択と集中実現

先に、損害保険会社では、本業に係る全ての商品、多くの事業計画がシステム開発を伴って遂行されると書きましたが、結果、どのシステムを開発するかは、どの商品を先にマーケットに出すか、どの事業をいち早く実現するかという経営戦略に密接に関連することになります。

つまり、システム開発案件の選択と集中は、企業競争力を高め、よりマーケットに歓迎される商品、サービスを選択することそのものとなっている訳です。

こうした中で、東京海上日動は情報化委員会と言う、経営的視点からシステム開発案件を選び、システム投資を決定していく仕組みを構築し、運用しています。

良いと思われる、必要とされるシステム開発案件は星の数ほど出てきます。その中から、真に優先順位の高い案件、必要性の高い案件を適切に選ぶと言うことはまさに経営そのものと言ってよいでしょう。

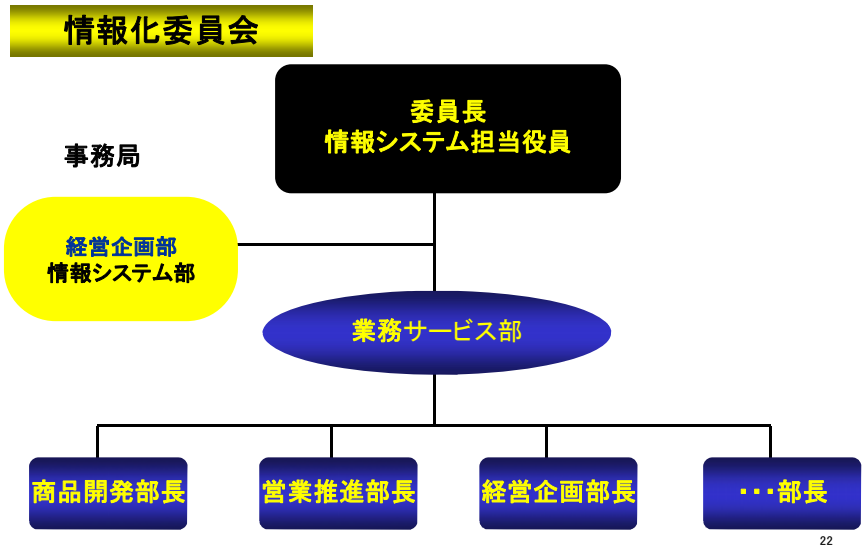


図 5-6. 情報化委員会

(3) IT 全般統制を維持する仕組み

次に、2 層目の IT 全般統制の事例を紹介します。

これは、システム開発が正しくうまく行く、運用が適切に行われると言ったことのために用意される IT ガバナンス施策とも言えます。

(a)アプリケーションオーナー制度

システム開発の参加者はたくさんいますが、最も重要なのは、そのシステムが自分の企てのために欲しい人ということになります。

突き詰めれば社長さんと言うことになるのでしょうが、実際のところ、ビジネスの装置としてのシステムということである場合、そのシステムを使って商売をしよう、商売をうまくやろうと組みたてる業務部門の人たちです。

東京海上日動では、この業務部門の人たちにアプリケーションに係るオーナーシップを持ってもらい、システム開発から運用に至るまで、システムライフサイクルの一翼を担う制度を作っています。

アプリケーションオーナーはシステムを商売上必要とする人たちです。どういう仕組をシステムで実現するか、システム開発に係る要求を具体化するのもこの人たちですから、この制度はシステムの要求品質を高め、システム開発の QCD を高めることに繋がっています。

図 5-7. アプリケーションオーナー制度

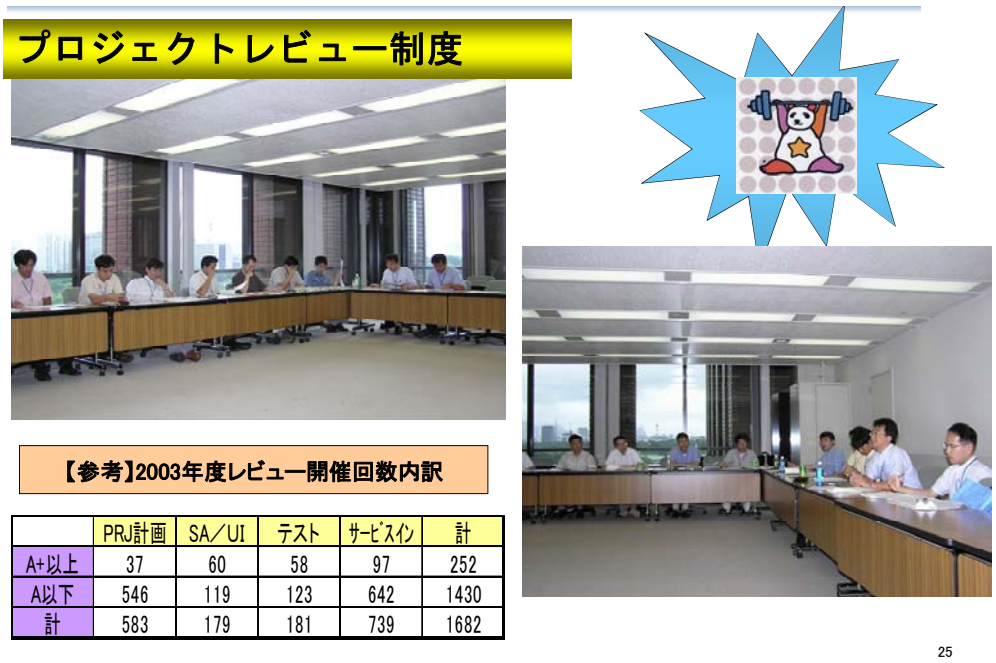
(b)プロジェクトレビュー制度

次にご紹介するのはプロジェクトレビュー制度です。アプリケーションオーナーとシステム開発担当者、加えて必要に応じて、各利用部門等のステークホルダが参加するプロジェクト管理のプラットフォームです。

システム開発はそれ自体が目的ではありませんので、システム開発を行うことで実現しようとしているものの QCD まで含めプロジェクト管理をする必要があります。システムがうまく作れても活用できなければ、作った意味はありませんし、開発が破綻しては描いたビジネスの絵が、それこそ絵に描いた餅になってしまいます。多い

時には数十億の費用がかかるシステム開発ですから、システム開発のマネージがその主要課題であることは言うまでもありません。

次の写真は、少し前のものですが、アプリケーションオーナーである業務部門と、システム部門がプロジェクト計画レビューを行っているところです。



	PRJ計画	SA／UI	テスト	ｻｰﾋﾞｽｲﾝ	計
A+以上	37	60	58	97	252
A以下	546	119	123	642	1430
計	583	179	181	739	1682

図 5-8. プロジェクトレビュー

(c)障害対策

運用に入ってからの IT ガバナンス、IT 全般統制事例をひとつご紹介します。

東京海上日動では、場合によっては経営レベルで大きな影響が出る、システムに係る障害対策に力を入れています。

管理職が主体となって、障害が発生の原因を作り込まないシステム開発の仕組作り、万一発生した時の被害最小化の施策、さらに再発防止策について組織レベルで対応力を増す取り組みを行っています。

システム開発に係る要求も、様々なものがここから発生しています。

それは、高信頼性を実現するシステム作りの原動力となっていると言ってもよろしいでしょう。

図 5-9. 障害対策

(d)その他施策

以上、3 例をご紹介しましたが、このほかにも大切な IT ガバナンス施策があります。

次にあげるものは、何れも IT 全般統制レベルで行うことが必要なものでしょう。

1) 社内システム監査

2) 開発と運用の分離

3) プロセス改善

4) 開発・運用子会社との SLA

5) エスカレーションスキーム

6) セキュリティ整備

(4) IT 業務統制実現の仕組

最後に、3 層目の IT 業務統制です。

ここでは、各アプリケーションで実現しなければならない仕組が多く出てきます。

アプリケーションにより、その実現の仕方が違ってくることもあります。要求はより、個別、具体的になります。

このレベルで発生する要求は、登場するステークホルダの、役割、職務権限、責任、目論見、都合といったことから、強く影響を受けます。IT ガバナンスレベルできちんとした考え方、一貫性の確保をしないと、掲げられた課題は処理したが、開発毎にいろいろな意味でバラバラと言うことになりかねません。

そのあたりを強く意識して取り組まなければなりません。
いくつかあげてみましょう。

1) 入力情報の完全性、正確性、正当性保証の仕組構築
2) 例外処理(各種異常数値等)発見修正の仕組
3) マスターデータの維持管理の仕組
4) アクセス、処理権限等管理の仕組
5) 誤処理防止の仕組:誤契約、誤払い、未払い等防止
6) データ処理証跡管理の仕組

5. 5. 終わりに

ITガバナンスという取り組みを見ると、要求の発生場所のいくつかが明らかになってきます。またその取り組みが品質の向上に役立つこともお分かりいただけたと思います。

要求はシステム開発の目的となったところから多く発生することは言うまでもありません。それは要求の保守本流と言ってよいでしょう、しかし、保守本流の目論見を正しく実現するためには、他に必須となる要求が多く存在する、と言うことなのだと思います。そして、様々な場所で発生するニーズを要求としてきちんと形にしていくためには、IT ガバナンスで要求の発生場所の施策品質、業務品質を上げ、要求品質を向上させることが必要なのです。

参考文献

[1] 経済産業省, IT ガバナンス, http://www.meti.go.jp/policy/it_policy/it_keiei/action/keyword/governance/index.html.
[2] 経済産業省, 新「システム監査基準」、「システム管理基準」の公表について, 2004, http://www.meti.go.jp/policy/netsecurity/new_systemauditG.html.
[3] 金融情報システムセンター, http://www.fisc.or.jp/.

6. 要件定義フェーズを取り巻くコンテキスト分析

NEC ネクサソリューションズ株式会社　小池　輝明

6．1．コミュニケーションとコンテキスト

6．1．1．ソフトウェア開発におけるコミュニケーション

コミュニケーションに関する研究は、多くの学者、研究者が広範囲に行っており、諸説の解説には膨大な紙面を要する。ソフトウェア開発では、顧客の要求を品質の確保されたコードにするまでに気の遠くなるようなコミュニケーションを行わなければならないことは周知の事実であり、名著「人月の神話」[4]でもコミュニケーションの問題を取り上げられている。

ソフトウェア開発は、顧客の要件に基づき開発が行われているが、元々要件を定義する作業が困難なために曖昧さが残り、そこから様々な解釈や思いこみが発生する。要件定義作業の困難さの要因の一つに顧客と開発側のコミュニケーションの問題がある。コミュニケーションには曖昧さがあり、これを排除するための技術や技法はいくつもあるが、結局完全なコミュニケーションは存在しない。開発が始まれば開発チーム内で無数のコミュニケーションが行われる。業務要件は何か、アーキテクチャで留意すべきこととは何か、品質面では何が要求されていて、どのようにそれを確かめるのか。各担当やチーム間でこれらが話し合われ開発計画が立案されるが、この計画にもコミュニケーションの曖昧さが暗い影を落とす。そんな要件は聞いていないよ、それは君の担当の仕事じゃなかったのか、など。コミュニケーションの問題がバグとなることも少なくない。しかし、納期は確実に迫り出荷のプレッシャーがメンバーにのしかかり始める。ソフトウェア技術者はそれらの困難に立ち向かわなくてはならず、ゴールの見えない作業を続けているかの錯覚に陥る。コミュニケーションの問題が回避できれば、もっと開発作業に専念できるのではないだろうか。

要件定義フェーズにおいて、コミュニケーションの視点からどのようにこの困難に取り組めばよいのか、本稿で考えて行きたい。

「組織の目的は、必要になるコミュニケーションと調整作業の量を減らすことだから、組織は上記のようなコミュニケーション問題に対する根本的な取り組みが必要なのだ。」(人月の神話[4]、第7章より)

6．1．2．コミュニケーションモデル

コミュニケーションモデルではシャノンとウィーバーの通信モデルが知られている。メッセージは情報源から送信機によって信号に変換され通信チャンネルを通して受信でメッセ

ージに変換されるというものである。ここでは、このモデルに近いコミュニケーションモデルとして「記号論への招待」[11]に記載されている記号論のモデルを取り上げたい。このモデルで取り上げるメッセージはただ相手に知覚されれば良いものではなく、何らかの意味を持っていなければならないと述べられている。何かを意味するとは記号で表現されることであり、さらに意味がなんらかの作用を引き起こすことである。相手に伝わるためにはメッセージを記号に変換する手順が決まっていなければならず、これをコードと呼ぶ。コードが厳密に決まっていれば良いが、人間のコミュニケーションはそうはいかない。いちいちコードに従って全てのメッセージを伝えるにはあまりに多くの労力を要し、相手にも相応の労力を求めることになる。短時間で効率よく伝えられるように言語には冗長性が備えられており、柔軟にコミュニケーションが行われるようになっている。しかし、冗長性が高ければ品質のよい伝達は難しくなる。冗長性を補間する仕組みとして、コードにはコンテキストを前提とした仕組みが備わっているのである。

少々乱暴に言えば、「コンテキストが共有できていなければ、コミュニケーションは失敗する」ということになる。

6．1．3．2 つのコミュニケーションモデル

コミュニケーションの構造をモデルで説明する。「記号論への招待」の「Ⅱ　伝えるコミュニケーションと読み取るコミュニケーション」の記載内容から図式化した「コミュニケーションモデルの派生」では、二つのモデルを表している(図 6-1)。

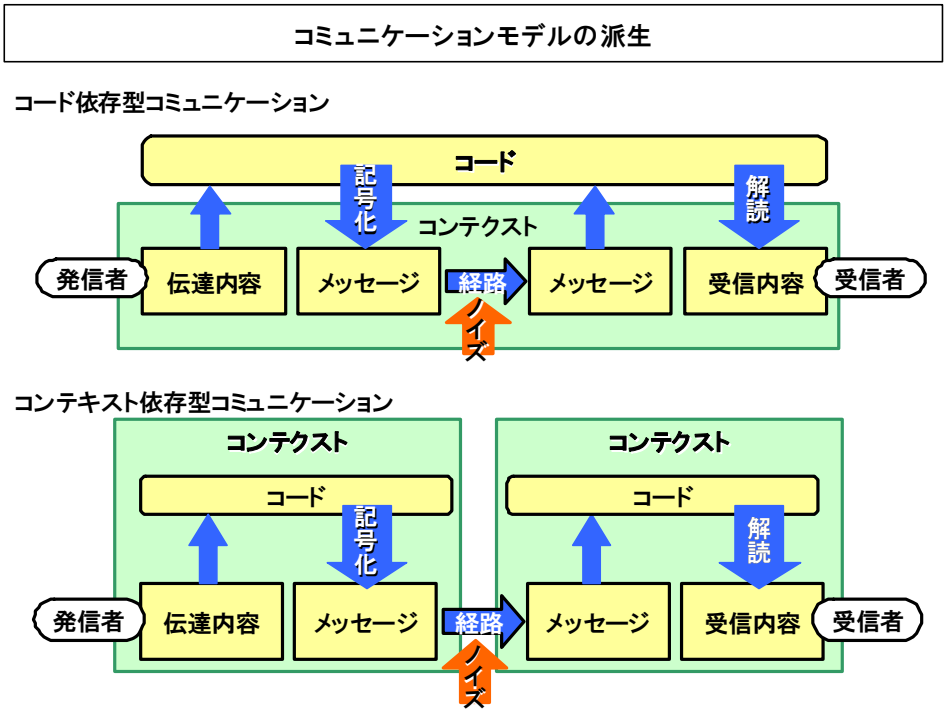


図 6-1．コミュニケーションモデルの派生

コード依存型コミュニケーションは、コードが厳密に決められていて、コンテキストの影響を殆ど受けない。そのため、発信者と受信者の間で正確なコミュニケーションが可能になる一方で、膨大な記号を駆使しなければならない。

一方、コンテキスト依存型コミュニケーションモデルは、コードがコンテキストの影響を受けることを意味している。つまりコンテキストを前提としたコードによりメッセージを変換し解読する。

コンテキスト依存型コミュニケーションは、発信者と受信者がそれぞれのコンテキストに依存したコード体系を持っているため、冗長性が高く効率が良くない。しかしコンテキストが共通化されれば、少ない記号で効率的なコミュニケーションが可能になる。

6．1．4．コミュニケーションを支えるコンテキスト

コミュニケーションで重要な役割を果たすコンテキストについてコミュニケーションモデルで説明する。コンテキストとは、文脈や背景という意味である。図 6-2 に示す、開発者とテスト担当者のコミュニケーションを想定したコミュニケーションについて着目してみる。開発者とテスト担当者は役割が異なるためにコンテキストも異なると考えられるが、全くコミュニケーションがとれないわけではない。同じソフトウェア技術者であれば技術的コンテキストもある程度共通だということが言えるが、同じ開発チームであれば、対象システムの内容に関しても共通なコンテキストがあるはずである。

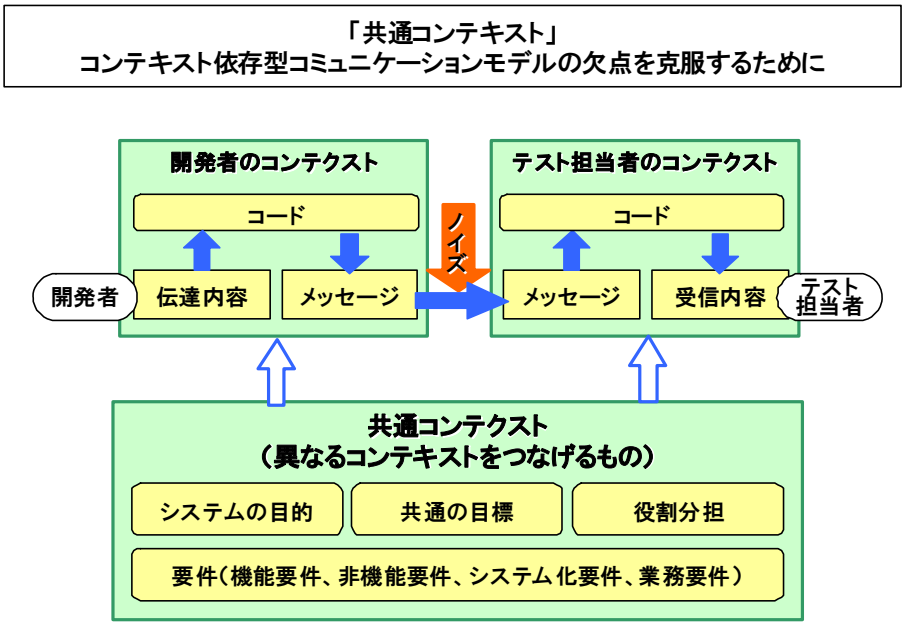


図 6-2．共通コンテキスト

コンテキストに依存したコミュニケーションであることを前提にすれば、なんらかの共通な

コンテキストがあれば、コミュニケーションを補間できるのではないだろうか。お互いが、どのようなコンテキストでいるか、どのようなコンテキストを採用しようとしているかがわかれば、コミュニケーションの問題の改善に役立つ。

そこで、ソフトウェア開発ではソフトウェア開発関連知識が共通のコンテキストになっていると仮定し、コンテキストの共有がどの程度計測できるのか検証を試みた。何らかの方法でコンテキストを可視化し、定量的評価を加えることができれば、改善の方法も見つかる筈である。ソフトウェア技術者のコンテキストは立場によってどのように異なり、それはソフトウェア開発にどう役立てることができるのだろうか。

6．1．5．意味ネットワーク

コンテキストの可視化のヒントになった意味ネットワークについて述べる。心理学ではコリンズとキリアンによる意味ネットワークモデルが有名であるが、ここで取り上げた意味ネットワークとは厳密には異なるが図示表現として参考にしている。

人の知識は宣言的知識と手続き型知識に分けられ、宣言的知識は概念を代表する概念ノードがネットワーク上につながった意味ネットワークモデルとして仮定されている。また、宣言的知識は言葉の概念や概念同士の関係についての意味記憶と特定の場所や時間などの文脈情報を含むエピソード記憶で構成されている。

宣言的知識は概念をノードとしたネットワーク構造をしていると言われている。この意味ネットワークの図示化のヒントになったのは、「進化しすぎた脳」[13]に掲載されている図である（図 6-3）。

この意味ネットワークは、英語圏の学生 1000 人を対象にした調査結果である。この図は、「指」という単語から体のどの部位を連想するかを表している。指(Finger)からは 52%の確率で手(Hand)を連想し、手(Hand)から 27%の確率で足指(Tue)を連想することを示している。この図からわかることは、連想の調査から意味ネットワークは可視化できるということと、使用している言語により意味ネットワークは異なるということである。この実験を日本人に対して行ったら違う結果になると考えられる。それは、日本人なら大抵は「足」と「脚」を明確には使い分けないが、英語の場合にはくるぶしの上と下で明確に分けているから「足」と「脚」を使い分けるのである。日本語なら「太もも」「すね」「ふくらはぎ」という連想が出てくるだろう。

このことから、携わっている仕事や状況によってこのような意味ネットワークが異なるのではないかと考えた。ソフトウェア開発ではいろいろな立場の人間が関係しているため、各位が異なった価値観を持っているのは想像に難くない。プロジェクトマネージャー、アーキテクト、テスト担当者、PMO、品質保証の担当。それぞれの役割により考えていることも異な

り、担当責務によって意味ネットワークが異なると考えるのが自然であろう。

コミュニケーションモデルと意味ネットワークという二つの知見から、ソフトウェア開発現場で働く人のコンテキストを明確にできればチームビルディングや開発中のコミュニケーションが効率化できるはずであると考えた。

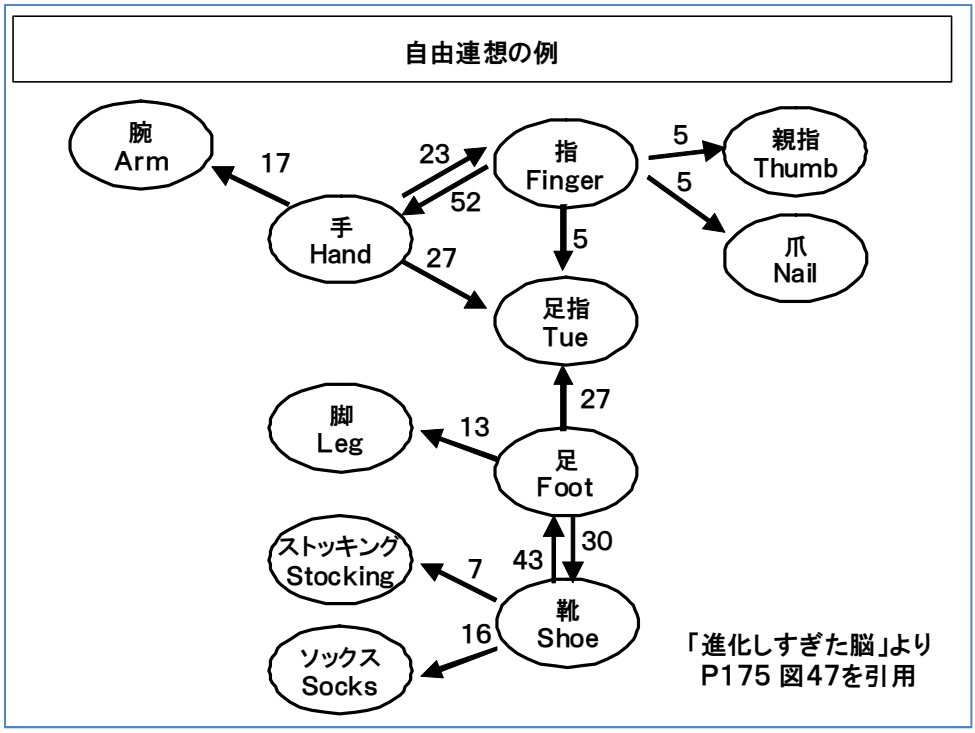


図 6-3. 自由連想の例

6. 2. コンテキスト可視化の実験

6. 2. 1. コンテキスト発見ワークショップ

意味ネットワークを図示化するためには自由連想データが必要になる。本事例では、この自由連想データを集めるためにコンテキスト可視化ワークショップを行った。このワークショップでは、ポジション、スタートワードと連想ドメイン、連想方法を設定して行われる。

最初にポジションの設定を行う。ポジションの設定とは、ソフトウェア開発関係者なのか、品質担当者だけなのか、ソフトウェア開発関連企業のスタッフなのかなど、どのような共通点が求められているかを明確にすることである。コンテキストを計測するのであれば、参加者のポジションをできるだけあわせておかなければならず、ワークショップ時には、設定されているポジションは何で、どのような人達のコンテキストを明確にしようとしているかを参加者に説明しておかなければならない。

次に、スタートワードと連想ドメインの設定を行う。スタートワードは連想を開始するワード(単語)である。スタートワードはワークショップ参加者が理解できるものでなければならず、

参加者によって捉え方が異なっていてもいけない。連想ドメインは、開発工程全体を指すのか、特定のフェーズに焦点をあわせるのかなど、連想範囲が発散しないようにするために設定する。

最後に、連想方法を決定する。連想方法は、連想ドメインの中で自由に連想させるが、連想の結果をどのように記録するのかを決めることが重要である。

今回のワークショップでは、ソフトウェア開発関係者に対して、スタートキーワードを「RFP」、連想ドメインを「ソフトウェア開発全般」と設定し、連想時間を 10 分で実施した。連想された単語は付箋紙に書き出し、書き出された単語から、再度連想される単語を書き出す。連想される単語は重複して出されたときもそのまま続ける。(この方法は認知アセスメントに使用する自由再生法にヒントを得た方法である。)

また、実施時に参加者のプロフィール情報として主な役割について、PMO、プロジェクトマネージャー、開発者、品質保証担当者、テスト担当者、技術スタッフからどれに属するのかを記録した。立場の違いでのコンテキストの差を分析するためにこうしたプロフィールは重要な情報である。

書き出された単語はソフトウェア開発のプロセスで分析するために共通フレーム[2]を参考に集計した。例えば、RFP→要件定義→設計、という連想であれば、「契約プロセス」、「企画・要件定義」、「開発プロセス」の順に連想されたとみなす。

6．2．2．役割の違いによるコミュニケーションの問題

現在の IT システムは多機能化が進み、そのためいくつかのチームが編成され、専門性も高まりつつある。専門性が高まるということは独立性も高まっているということになる。独立性が高まれば、各チーム間、チーム内のコミュニケーションが重要になり、コミュニケーションの質が品質に影響することも想像に難くない。コミュニケーションが上手くとれている協調性のあるチームは品質も良いことが予想される。

実際には、大抵のコミュニケーションは自分の役割を全うすることを優先し、他の役割のメンバーの立場を考えたコミュニケーションを最初からとられていることは少ないのではないだろうか。チームビルド時には適切なコミュニケーションを可能にする基盤が作られなくてはならないが、初めて顔をあわせるメンバー間でどのようにコミュニケーションの質を高めて行けばよいのだろうか。

簡単な例として、私たちが初対面の人と話をするときのことを思い出してみよう。当たり障りのない天気の話から初めて、次第に景気や時事の話題に移行していくはずである。これは、会話の中でお互いの共通話題を探し、相手の反応が良い話題に関しては段々とその内容を深め、コミュニケーションにおけるコンテキストを模索している行動なのである。

しかし、今日のソフトウェア開発プロジェクトには、天気の話題から始めるように時間をか

けてゆっくりとコンテキストを共有している余裕はない。そこで、コンテキストを見えるようにすれば、お互いの役割の理解に役立つはずと考えた。

6．2．3．ワークショップ実施結果について

今回の調査では、ソフトウェア開発関係者 58 人の方にご協力いただき、連想単語は全部で 2100 にもなった。その中から、今回は、開発者、品質管理担当、ソフトウェアテスト担当の 43 人の方々のデータに絞って分析を行った。

分析した三者は次のような担当を想定している。開発者は要件定義から実装・テストまでの開発全般、もしくは設計/実装などの一部分を担当する技術者。品質担当者は、品質を維持する活動に従事する技術者で品質管理/品質保証部門など独立部門だけでなく、開発プロジェクト内で品質維持業務を専任または専任に近い形で従事している技術者を含む。テスト担当者は、ソフトウェアテストに従事する技術者で、テスト計画/テスト設計から実施までのテスト関係全般、もしくはその一部を担当する技術者を想定している。

6．3．コンテキストの可視化

6．3．1．ソフトウェア技術者の意味ネットワーク

ソフトウェア技術者がソフトウェア開発でどんなことを意識しているかについて自由連想でソフトウェア開発関連の単語を書き出すワークショップでデータを収集し、そのデータを共通フレームのプロセスで集計し意味ネットワークを作成した。分析は共通フレームのプロセス名を使用して分析を行い、開発者、品質担当者、テスト担当者の各意味ネットワークからコンテキストの違いを分析する。

ここで「開発者の意味ネットワーク分析」を例にこの図の見方を説明する。「契約合意の視点」の領域に相当する単語からは 29％の割合で「企画要件定義の視点」に関連する単語を連想している。同様に「企画要件定義の視点」から 25%の割合で「開発プロセス」に関連する単語を連想していることになる。また、同じプロセス内での回帰的連想もある点に注意が必要である。（但し、10%未満のものは図の簡素化のために省略している。）

意味ネットワークの構成要素は次の通りである。プロセス群を形成している円を「ノード」、ノードとノートの間の線を「リンク」、リンクが集中しているノードを「ハブノード」と呼ぶことにする。「ハブノード」は、コンテキストの中心的位置づけである。

6．3．2．開発者の意味ネットワーク

開発者の意味ネットワークで、リンクの多いハブを探してみると「開発プロセス」が最も多い五つのリンクを持っていることがわかる。

開発者のパスをなぞってみると、「契約合意の視点」→「企画要件定義の視点」→「開発プロセス」→「運用プロセス」と、ウォータフォールのようなパスになっていることがわかる。開発者の意味ネットワークで、「運用プロセス」と「保守プロセス」のノードとリンクがあるのがこの後説明する品質担当者とテスト担当者と大きく違う点である。「文書化プロセス」が「品質管理の視点」と「組織に関するライフサイクル」の二つのノードがあることから、開発者にとっての文書は、開発のアウトプットという位置づけよりも管理のために記録するものという位置づけの側面からの意識が強いのかも知れない。

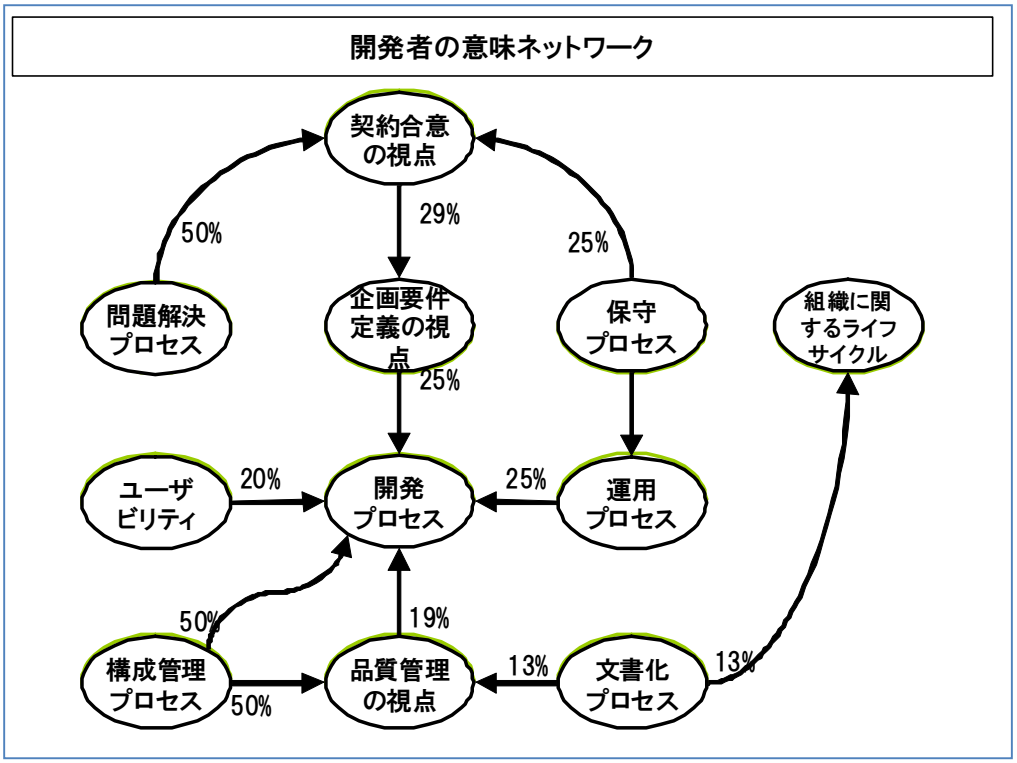


図 6-4. 開発者の意味ネットワーク

6. 3. 3. 品質担当者の意味ネットワーク

次に品質担当者の意味ネットワークについて解説する。開発者の意味ネットワークと見比べるとリンクがやや複雑化していることがわかる。これは「組織に関するライフサイクル」と「開発プロセス」の二つがハブノードになっているからである。そのために開発者の意味ネットワークとパスも異なっている。「文書化プロセス」は、「組織に関するライフサイクル」と「品質管理の視点」とリンクがあるが、「開発プロセス」とのリンクが強いことがわかる。これは品質担当者が開発者と違い、文書を開発プロセスのアウトプットとして位置づけられているのだと予想される。開発者の意味ネットワークと比べると、「運用プロセス」と「保守プロセス」がないことがわかる。

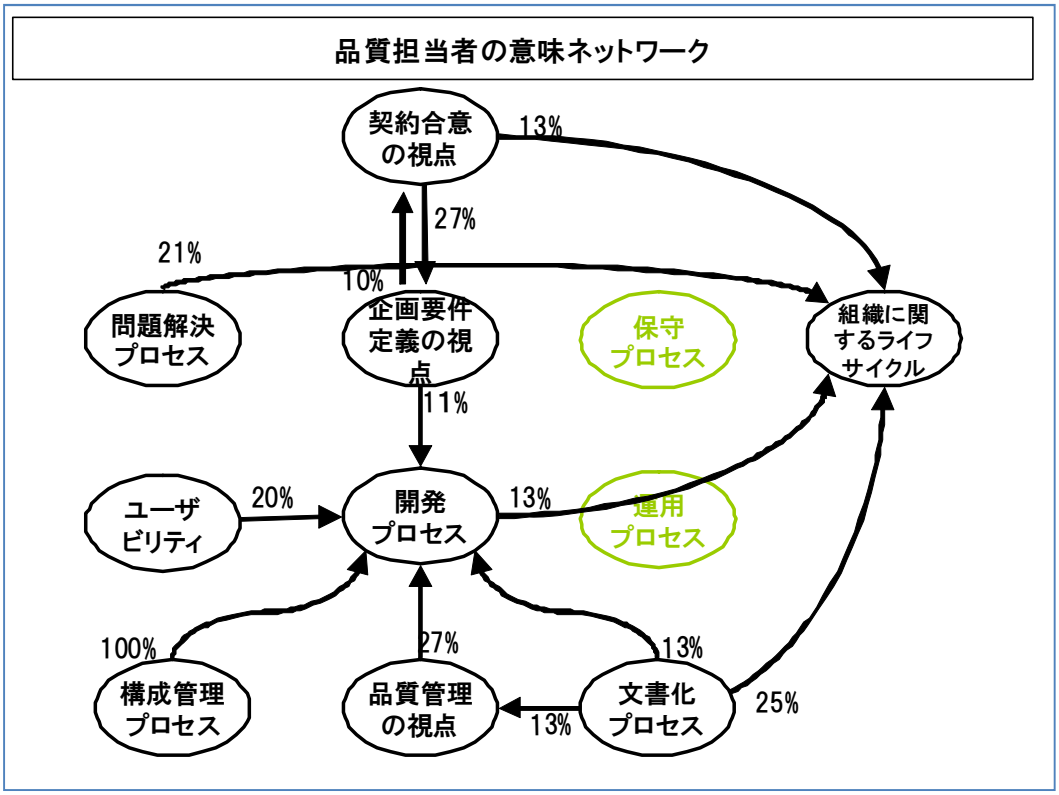


図 6-5. 品質担当者の意味ネットワーク

6. 3. 4. テスト担当者の意味ネットワーク

テスト担当者の意味ネットワークは開発者と品質担当者と比べてどうなっているのだろうか。テスト担当者の意味ネットワークではハブノードが 6 個存在する。さらに、ノードの数は三者の中で最も少ない 7 個にも関わらず各ノードのリンクの数は 26 個にもなっている。「文書化プロセス」に着目してみると、「契約合意の視点」、「企画要件定義の視点」、「品質管理の視点」、「開発プロセス」と 4 つのノードとリンクがある。このことからテスト担当者は、各フェーズにはアウトプットされたドキュメントがあることを前提にテストの取り組みをしているのではないだろうか。また品質担当者と同じく「運用プロセス」と「保守プロセス」に対するリンクがないことがわかる。

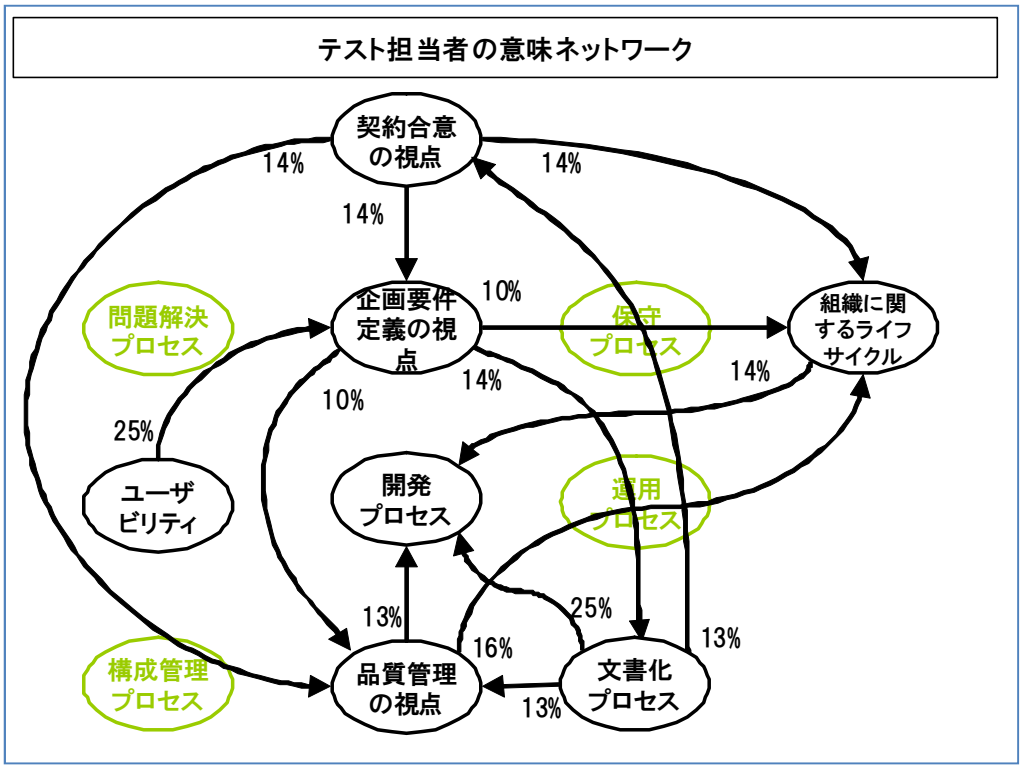


図 6-6. テスト担当者の意味ネットワーク

6. 4 . 要件定義フェーズを取り巻くコンテキスト分析

6. 4. 1. 要件定義フェーズを取り巻く意味ネットワーク

意味ネットワークの分析を通して、三者の違いについて分析をしたが、さらに「要件定義フェーズ」に着目して分析をしてみる。開発者と品質担当者の意味ネットワークで、「企画要件定義の視点」に関連するノードについて分析した結果を示す。

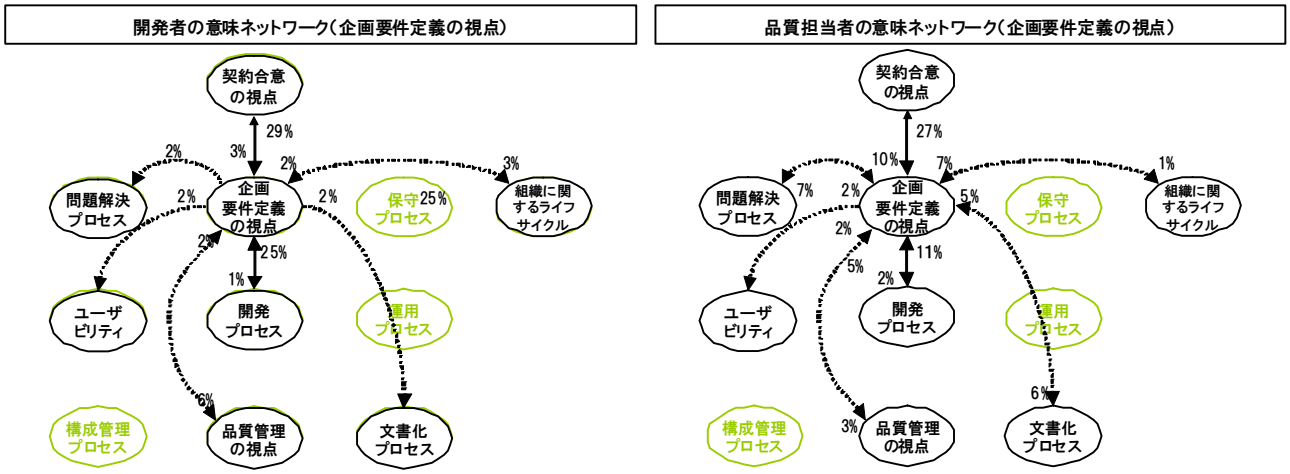


図 6-7. 企画要件定義の視点

連想割合が 10%以上のリンクは実線で、10%未満のリンクは破線で表示している。実線のリンクに着目してみると、開発者と品質担当者は、ウォータフォールのように「契約プロセス→企画要件定義→開発プロセス」という流れになっており、意味ネットワークは非常に似た形状をしているが、細部に留意してみると違いを発見できる。

開発担当者は「企画要件定義の視点」から他のプロセスへの連想は 7 つあり、これは品質担当者と同じである。しかし、開発担当者が他のプロセスから「企画要件定義の視点」への連想が 4 つなのに対して、品質担当者は 6 つあることがわかる。

要件定義を基点とする開発担当者と品質担当者のコミュニケーションでは、開発担当者は他のプロセスに向けて発散する可能性があるが、品質担当者は他のプロセスから戻りがあるので、開発者の思考の発散に歯止めをかけることができるかもしれない。そのためプロジェクト計画のレビューなど、開発全体を見渡す必要がある場面では、二者のペアが有効になる。

6. 4. 2.「要件定義フェーズ」内の連想単語分析

次に、開発担当者と品質担当者の「要件定義フェーズ」内の詳細アクティビティ別の連想単語数の比較を分析する(図 6-8)。

要件定義フェーズの詳細アクティビティ分析(単語数分析)

	開発担当		品質担当	
	単語数	割合	単語数	割合
1.5.1.1 要件定義作業の定義	7	12%	13	29%
1.5.1.2 必要な支援プロセスの実施	3	5%	7	16%
1.5.1.3 要件定義環境の準備	3	5%	2	4%
1.5.2.1 利害関係者のニーズの識別と制約事項の定義	26	45%	13	29%
1.5.2.2 業務要件の定義	13	22%	6	13%
1.5.2.3 新組織及び業務環境要件の具体化	1	2%	0	0%
1.5.2.4 機能要件の定義	3	5%	1	2%
1.5.2.5 非機能要件の定義	2	3%	3	7%
	58		45	

図 6-8. 要件定義フェーズの詳細アクティビティ分析(単語数)

図 6-8 から、開発担当者も品質担当者もほぼ同じアクティビティの連想が多いことが分かる。共通で多いアクティビティとして、「利害関係者のニーズの識別と制約事項の定義」、

「業務要件の定義」があげられる。「利害関係者のニーズの識別と制約事項の定義」はステークホルダ間での要件の調整が含まれ、「業務要件の定義」は業務としてのシステムで実装する要件は何かを定義する。この二つのアクティビティは要件定義作業全般を通して最も苦労が多く、しかし、要件定義の醍醐味とも言える作業である。そのため、連想数が多いのだと予想される。

逆に、「新組織及び業務環境要件の具体化」、「機能要件の定義」、「非機能要件の定義」は全体的に連想が少ない傾向である。「新組織及び業務環境要件の具体化」では、構築するシステムと組織・環境の整合をとる作業だが、一般には顧客に依存する内容なので連想が少ないのだと予想できる。「機能要件の定義」と「非機能要件の定義」の連想数が少ないのには意外な感じだが、「業務要件の定義」が終了した時点で要件定義フェーズは終わりで、以降は設計作業と捉えているのかも知れない。

これらのアクティビティは要件定義フェーズでは「要件定義→業務要件→機能要件」というステップで作業が進むことを考えると後半の作業と言える。一般に、要件定義フェーズの完了と設計フェーズの開始の定義が曖昧であることも影響しているのだろう。また、機能要件/非機能要件の連想が少ないのは、設計以降に問題を先送りしている可能性もあり、次工程に問題が混入するリスクも高くなる。

6.4.3. 経験年数別にみる「要件定義フェーズ」内の連想単語

役割の分析をさらに詳細化し、10 年目までの経験年数で分析してみる(図 6-9)。

要件定義フェーズの詳細アクティビティ分析(経験年数別分析)

	開発者			品質管理		
	A群	B群		A群	B群	
1.5.1.1 要件定義作業の定義	5		△	5	4	△
1.5.1.2 必要な支援プロセスの実施		2	○	1	2	○
1.5.1.3 要件定義環境の準備		1	○	1		△
1.5.2.1 利害関係者のニーズの識別と制約事項の定義	5	10	○	5	6	○
1.5.2.2 業務要件の定義	6	3	△	3	2	△
1.5.2.3 新組織及び業務環境要件の具体化			×			×
1.5.2.4 機能要件の定義		2	○	1		△
1.5.2.5 非機能要件の定義		2	○		2	○

凡例:A群(～5年)、B群(6年～10年)

図 6-9. 要件定義フェーズの詳細アクティビティ分析(経験年数別)

このように比較してみると、経験年数が増えることで連想数が増える傾向と、経験年数が増えると連想が減るという二つの傾向が見える。

経験年数が増えることで連想数が増える傾向があるのは、「必要な支援プロセスの実施」、「利害関係者のニーズの識別を製薬事項の定義」、「非機能要件」であり、経験年数が増えると連想が減る傾向にあるアクティビティは「要件定義作業の定義」、「業務要件の定義」があげられる。

経験年数が短いA群で連想が多いアクティビティは、要件定義フェーズの比較的初期に遭遇する事象に密着しているアクティビティだからであろう。この A 群で連想数が多いアクティビティが B 群では下がっている傾向があり、逆に A 群で連想の少ないアクティビティは B 群では多くなっており、この点について考察してみたい。

B 群で連想が少ないアクティビティは実際には経験が長くなるにつれ作業量が増えているはずだ。人間はある作業をする場合、作業に不慣れだと作業で失敗しないように注意深く作業しているが、慣れてくるとあまり頭では考えずに作業を行うことができる。これは作業が無意識でも行えるように学習されているからである。こうなるといちいち作業手順を確認することなく効率的に作業を行うことができる。今回のワークショップでは宣言的知識が現れることを前提にしているので、今回の結果のように B 群で連想が少ないのは、作業経験が豊富である意味で潜在意識に移っていると言える。

6. 4. 4. コンテキストの視点から要件定義フェーズへの提言

以上の分析結果を要件定義フェーズでどのように活かすために、要求開発、スキルアップ、チーム編成の三つの視点で考えたい。

要求開発では、「獲得(引き出し)」、「分析」、「仕様作成」、「妥当性確認」のプロセスがある。「獲得(引き出し)」では顧客と、「分析」と「仕様作成」では開発側チーム内の、「妥当性確認」では開発者と品質担当者とのコミュニケーションが欠かせない。本稿では、顧客とのコミュニケーションについては省略するが、開発を受けた側でのコミュニケーションについてはこれまでの考察が適用できる。例えば、開発チーム内でも経験年数により連想プロセスが違い、開発者と品質担当者ではプロセスの連想が異なり補間関係にあることがわかった。そこで、要求開発の各プロセスでコンテキスト発見ワークショップによりコミュニケーション状態を確認することが有効である。

経験年数別の内容からスキルアップについて考えてみる。経験年数が少なければ連想されるプロセスが少なく、経験年数が長くなったときに連想されるプロセスが減少することを考慮すると何らかの方法で各プロセスへの意識付けを顕在化する必要があるのではないか。例えば、要件定義を進める上で OJT を組み込み、既に経験・習得済みのプロセスに

ついて言葉で伝えることを通して、形式知化するとともに自分の経験を振り返ることができる。そして、暗黙知化によるチーム力の低下を防げるのではないだろうか。

また、開発者と品質担当者、さらに経験年数の分析を振り返ると、チーム編成では経験が長いメンバーだけ集めれば効率よく作業が進むということは言えないことがわかる。そのため、チーム力を高めるためには、経験の少ないメンバーを敢えて投入したり、品質管理を担当するメンバーを積極的にレビューに参加させたりすることが重要なのである。

6. 5. コンテキストの発見を「気づき」に

コンテキストの可視化の効果は、コンテキストを意識したコミュニケーションにより、お互いの立場や役割の違いについて「気づき」の機会を創出できることである。この「気づき」こそ、コミュニケーションの問題解決のスタートだ。そして、コミュニケーションの問題の解決法は正確なコミュニケーションをするより他にない。コンテキストが見えてくるとお互いの立場の違いが明確になり、立場の相互理解がコミュニケーションを効率化するはずである。

コミュニケーションの問題を乗り越え、ステークホルダの間でコンテキストが共有されることにより、顧客要件を正しく理解できるようになる。顧客要件を正しく理解できれば、システムで実装すべき要件が明確になり、顧客にとって「動いて、使えるシステム」になるはずである。

また、コンテキストは変化するということを忘れてはならない。変化するコンテキストを正確に把握し、コンテキストの共有を継続することも大切である。ワークショップの実施により、コンテキストの変化にも気づくことができる。

今回はコンテキストを例にお互いの立場の違いに気づくことを提案した。

重要なのは、顧客を含めて開発に携わるメンバーがどのようなコンテキストを持ち、それによりどんな価値観を抱いているかを把握しようとすることである。こうしたコンテキストへの「気づき」がコミュニケーションの効率化の大切なステップなのである。

参考文献

[1] SQuBOK 策定部会, ソフトウェア品質知識体系ガイド, オーム社, 2007.
[2] 情報処理推進機構ソフトウェアエンジニアリングセンター(編), 共通フレーム 2007, オーム社, 2007.
[3] JSTQB, テスト技術者資格認定シラバス, http://www.jstqb.jp/.
[4] F. P. Brooks, Jr., 人月の神話 新装版, ピアソンエデュケーション, 2002.
[5] D. C. Gause, G. M. Winberg, 要求仕様の探検学, 共立出版, 1993.
[6] D. C. Gause, G. M. Winberg, ライト、ついてますか, 共立出版, 1987.
[7] K. E. Wigers, ソフトウェア要求, 日経 BP, 2003.

[8] ソフトウェア・テスト Press, Vol.8, 技術評論社.
[9] E. M. Rogers, コミュニケーションの科学, 共立出版, 1992.
[10] 辻 幸夫(編), ことばの認知科学事典, 大修館書店, 2001.
[11] 池上 嘉彦, 記号論への招待, 岩波新書, 1984.
[12] 池谷 裕二, 脳はなにかと言い訳する, 祥伝社, 2006.
[13] 池谷 裕二, 進化しすぎた脳, 朝日出版社, 2004.
[14] 中島義明, ほか, 新・心理学の基礎知識, 有斐閣ブックス, 2005.
[15] 山本七平, 空気の研究, 文春文庫, 1993.

7. 経営とITの連携に向けた可視化の取り組みについて

株式会社 NTT データ 技術開発本部 ソフトウェア工学推進センター 斎藤 忍

7.1. 背景、目的

システム開発のモデルのひとつとして V 字モデルがあります(図 7-1)。V 字モデルは、左側にシステムの品質を作り込むプロセス、右側に作り込んだ品質を検証するプロセスを配置し、それぞれの対応を明確化するモデルです。

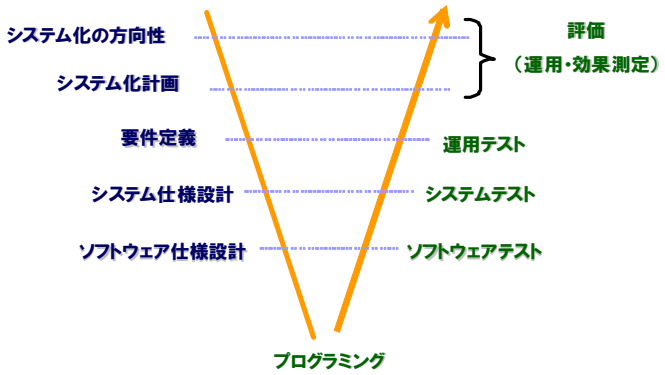


図 7-1. V 字モデル

近年のシステム開発では、事業や業務の検討から要件定義までの開発プロセスの初期段階(超上流プロセス=システム化の方向性、システム化計画、要件定義)の重要性が指摘されています[1]。超上流プロセスの特徴としては以下のようなことがあげられます。

(1)開発に入る前の要求仕様をはっきりさせる段階。

(2)IT ベンダとユーザの双方が参画することが重要。

(3)現場の「やりたいこと」ではなく、経営戦略の視点から「やるべきこと」を定義する。

(4)経営者の参画が求められる。

超上流プロセスにおける 3 つの工程の実施内容と主な成果物を表 7-1 に示します。

表 7-1. 超上流プロセスの実施内容と成果物

工程	実施する内容	主な成果物
システム化の方向性	開発するシステムは何のために必要とされるのかを明らかにする。 (=システムの目的)	ビジネスプラン (システムの目的と手段に関する計画)
システム化計画	システムの目的の達成のためにはどのようなことを実現しなければならないのかを明らかにする。 (=システムの手段)	
要件定義	ビジネスプランに基づきシステム実現のために必要な事項をまとめる。	要件定義書

本稿ではシステム化の方向性とシステム化計画で作成されるビジネスプランに着目します。ビジネスプランの策定、運用の課題としては以下のようなことがあげられます。

(a)ステークホルダ増加に伴うシステム要求の多様化に伴い、経営から業務・システムのレベルまで整合性のあるビジネスプランを策定しなければならない。
(b)ステークホルダを取り巻く環境変化のスピードの加速化に伴い、環境変化に即座に対応するためのビジネスプランの迅速な見直しをしなければならない。

そこで、以上の課題を達成するためのアプローチとして、本稿ではモデルを用いたビジネスプランの要求分析技法の提案、および、提案技法の事例への適用と考察を述べます[2]。

7. 2. ビジネスプランのメタモデル

提案技法で用いるビジネスプランのメタモデルを概説します。OMG (Object Management Group)で策定されているビジネスプランのメタモデルである Business Motivation Model(以降、BMM と略記)を取り上げます[3]。

BMM ではビジネスプランの基本部分は「目的」と「手段」の 2 つで構成されます。そして目的と手段に影響を及ぼす(作用する)ものが「影響要因」になります。図 7-2 に BMM の主要な要素を示します。

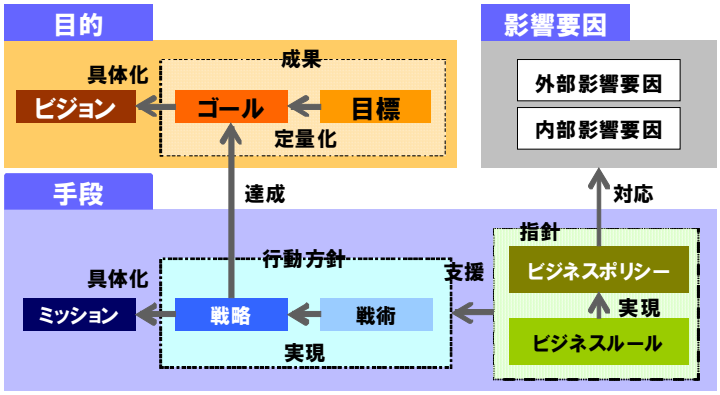


図 7-2. ビジネスプランのメタモデル

BMM と超上流プロセスとの対応関係を図 7-3 に示します。BMM のモデルでは成果(ゴール、目標)には対象とするシステムは何のために必要とされるのかというシステムの目的が表現されます。これは開発プロセスの中では最上流のシステム化の方向性で定義される内容に相当します。一方で、行動方針(戦略、戦術)と指針(ビジネスポリシー、ビジ

ネスルール）にはシステムの目的を達成するためにどのようなことを実現するべきかというシステムの手段が表現されます。これはシステム化計画で定義される内容に相当します。このようにBMMのモデルは、超上流プロセスの2つの工程（システム化の方向性、システム化計画）と対応付けができます。また、BMM のビジネスルールに記述された内容が自動化対象（IT システムで実現する内容）であるとき、その内容は次の工程である要件定義工程への入力情報となります。

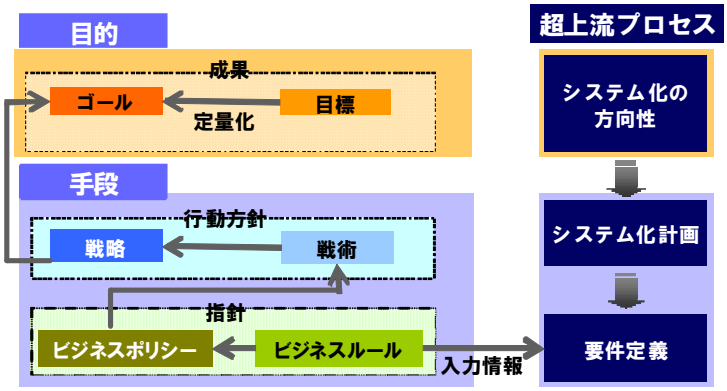


図 7-3. BMM と超上流プロセスの関係

以降では BMM の構成要素の説明をしていきます。

目的は、ビジョン、ゴール、目標の 3 つの要素に分かれて表現されます。図 7-4 に各要素の定義とサンプルを記します。以降のサンプルではレンタカー会社のビジネスプランになります。

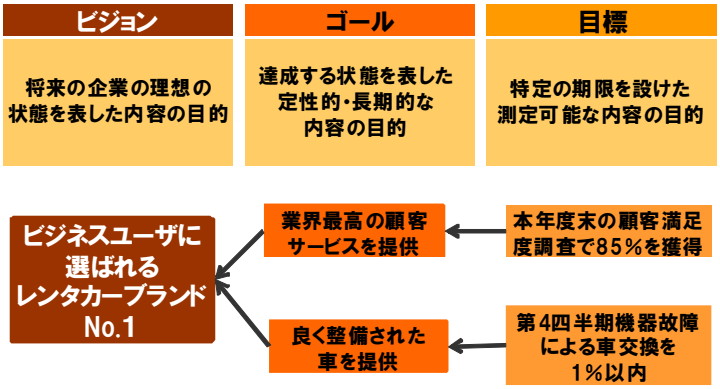


図 7-4. 目的のモデル

手段の基本部分は、ミッション、戦略、戦術の 3 つに分けて表現されます（図 7-5）。

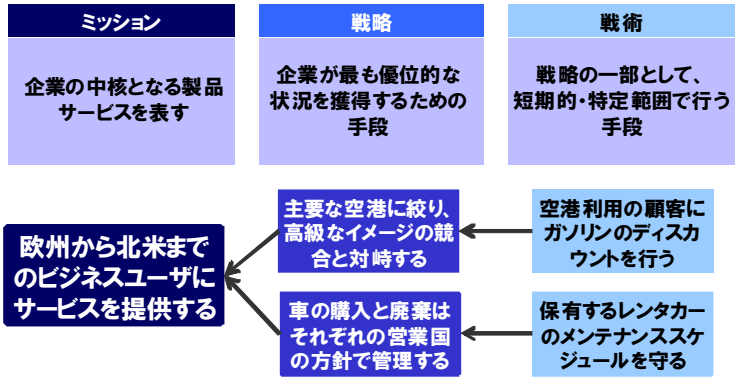


図 7-5. 手段のモデル(1/2)

手段のもう一つのモデルとして指針があります。指針は戦略、戦術を支援する内容が記述され、ビジネスポリシーとビジネスルールの 2 つに分けて表現されます(図 7-6)。

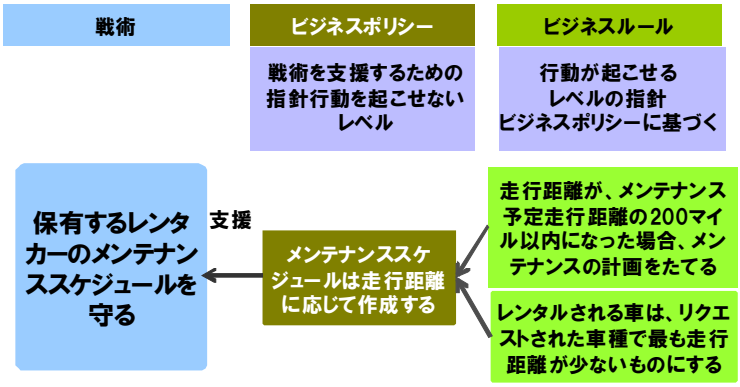


図 7-6. 手段のモデル(2/2)

また BMM では、目的や手段の内容に作用する要素として、影響要因を考慮しています(図 7-7)。BMM では影響要因の標準的なカタログを提供しています。影響要因の内容を特定しその内容を理解することはビジネスプランの強化につながります。

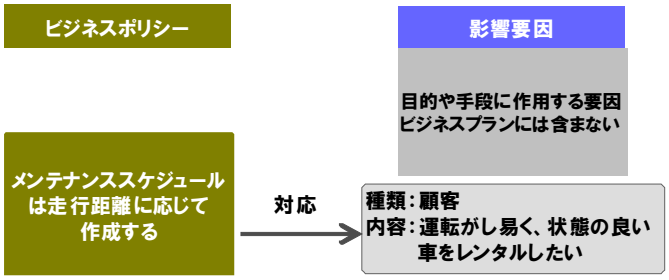


図 7-7. 影響要因のモデル

7. 3. 要求分析技法

技法 1:モデルチェックルール

BMM を活用してビジネスプランの内容をモデル化する際にモデルが守るべきルールを定義します。BMM を拡張しメタモデルの各要素間の多重度を厳密に定義することでルールを導出しました。モデルチェックルールは、このルールを適用することによりビジネスプランのモデルを構造的な側面からチェックを行う技法です。表 7-2 はモデルチェックルールの抜粋になります。

表 7-2. モデルチェックルール(抜粋)

分類	モデルチェックルールの内容
成果	ゴールは少なくとも1つ以上の目標と関連をもたなければいけない。
	ゴールは少なくとも1つ以上の戦略と関連をもたなければいけない。
	:
行動方針	戦略は少なくとも1つ以上の目標と関連をもたなければいけない。
	:
指針	ビジネスポリシーは少なくとも1つ以上のビジネスルールと関連をもたなければいけない。
	:
影響要因	影響要因は1つ以上のビジネスポリシーと関連をもたなければいけない。

モデルチェックルールを適用することで、経営のレベルから業務、システムのレベルまでビジネスプランの構成要素が途切れることなく連鎖しているかを確認することができます。これによりビジネスプランの内容の整合性の検証を支援できます。

技法 2:アセスメントパターン

経営環境は時々刻々と変化をするため、ビジネスプランに作用する影響要因の内容にも変化が起きることが考えられます。この場合、一度作成したビジネスプランにも見直しが必要になります。アセスメントパターンは影響要因の変化に伴うビジネスプランへのインパクトを評価して、その評価結果に基づき現状のビジネスプランの見直し対象を提示する技法です。

はじめの影響要因のインパクト評価では、影響要因の変化がもたらすビジネスプランへのインパクトの大きさを現在と将来の 2 つの観点から判定します。表 7-3 にインパクト評価の例を示します。インパクトはビジネスプランに対して良い作用(ポジティブ)と悪い作用(ネガティブ)の双方から評価を行うようにします。双方の得点を足し合わせて、それぞれの影響要因の変化に対する現在と将来のインパクト得点を算出します。

次に算出したインパクト得点の組み合わせより、影響要因の変化を 4 つのパターンに分類します。この 4 つのパターンをアセスメントパターンと呼びます。

表 7-3. インパクト評価表

◎:5点、○:3点　△:1点

影響要因		現在のインパクト			将来のインパクト		
種類	内容	ポジティブ	ネガティブ	得点	ポジティブ	ネガティブ	得点
競合	企業合併による競合の大規模化	ー	△	1点	ー	◎	5点
顧客	少子高齢化の促進	△	△	2点	△	ー	1点
規制	コンプライアンスの強化	ー	○	3点	△	○	4点
	……						

図 7-8 にインパクト得点とアセスメントパターンの関係を示します。アセスメントパターンは影響要因の変化に対するビジネスプランの見直し対象の方針を示しています。例えばパターン 3 では見直し対象に指針(ビジネスポリシー、ビジネスルール)を提示しています。これは,現在のインパクトは大きいが将来のインパクトが小さいため、成果(ゴール、目標)や行動方針(戦略、戦術)を見直しする必要はないが、指針(ビジネスポリシー、ビジネスルール)の見直しは必要であるという意味になります。4 つのパターンで示す見直し対象は見直しをする上での基点になります。例えば、パターン 1 の提示する見直し対象は成果(ゴール、目標)ですが、当該成果の見直しを受けて行動方針(戦略、戦術)、そして指針(ビジネスポリシー、ビジネスルール)まで変更されることもあると考えられます。

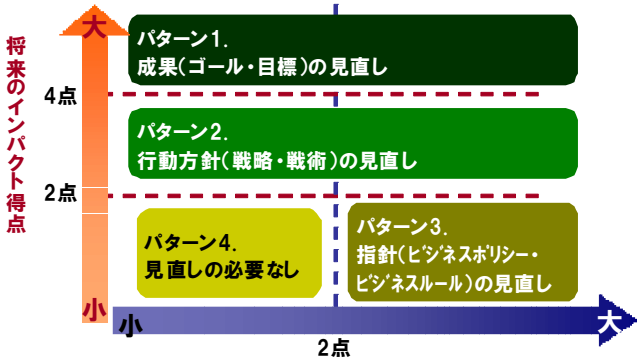


図 7-8. インパクト得点とアセスメントパターンの関係

7. 4. ケーススタディ

企業の商品開発・調達を支援する業務システム(IT システムも含む)の計画(ビジネスプラン)を対象に提案技法の適用を実施しました。具体的な実施内容は以下の 2 つになります。

1.文章で記述された計画(ビジネスプラン)のモデル化を行い、提案技法 1 を適用

2.計画策定時点には考慮していなかった影響要因を取り上げて提案技法 2 を適用

[技法 1 の適用結果]

モデルチェックルールの適用により、記述されたモデルのなかでルールが守られていない箇所(構造上の不備)を指摘することができました。その上で、これらの不備に対してビジネスプランの担当者より妥当性の評価を実施しました。評価結果を表 7-4 に示します。指摘内容の 87.5%(=7/8)が妥当との評価が得られました。担当者は指摘されたビジネスプランの構造上の不備を見直すことで、ビジネスプランの内容の不整合や検討漏れを改善できます。このように、提案技法はビジネスプランの内容の品質の向上に寄与できると考えられます。

表 7-4. 指摘内容の評価結果

ルールの分類	指摘内容	評価
成果	ゴールに関連付けされていない目標が存在	○
	目標に関連付けされていないゴールが存在	○
行動方針	ミッションに関連付けされていない戦略が存在	○
	ゴールに関連付けされていない戦略が存在	○
	関連するビジネスポリシーのない戦術が存在(1)	○
	関連するビジネスポリシーがない戦術が存在(2)	○
	関連するビジネスポリシーがない戦術が存在(3)	−
	関連するビジネスポリシーがない戦術が存在(4)	○

○:指摘内容は妥当(修正の必要有)と判断

[技法 2 の適用結果]

図 7-9 に技法 2 の適用によりビジネスプランの見直しを実施した例を示します。外部影響要因の「顧客」の変化として「商品の偽装表示への不信感高騰」を取り上げています。本例ではこの影響要因の変化をパターン 2 と評価しました。その上で、行動方針の見直し(戦略の追加、戦術の追加)と指針の見直し(ビジネスポリシーの追加)を実施しました。

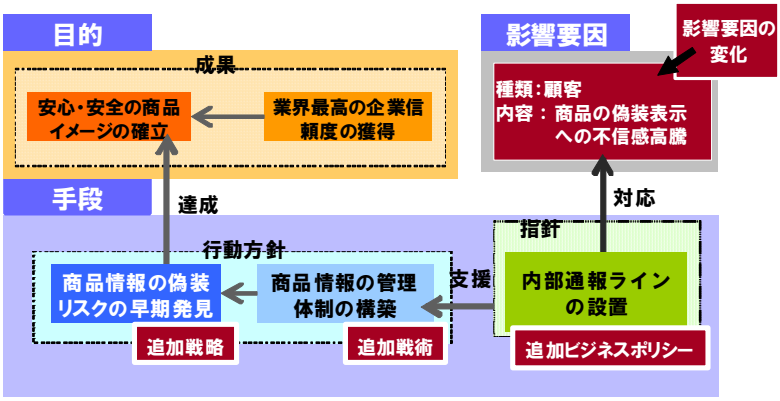


図 7-9. ビジネスプランの見直しの例

以上のような適用結果に対してビジネスプランの担当者からは以下のような評価コメントが得られました。

(1)アセスメントパターンは影響要因の変化への対応を考える際に、ビジネスプランのどの部分から考えれば良いかのヒントや手がかりを提供してくれる。
(2)ビジネスプラン策定の当事者であれば提案技法と同じようなことを実施しているはずである。但し、提案技法のような手順化は通常なされていない。
(3)ビジネスプラン策定の経緯を知らない人が運用担当者になった場合に、影響要因の変化に対してビジネスプランのどの部分を見直すべきかの判断は難しい。そのような状況では提案技法は更に有用になる。

7. 5. まとめと今後の課題

本稿ではシステムの開発プロセスの初期段階(システム化の方向性、システム化計画)で定義されるビジネスプランの要求分析技法として、モデルチェックルールとアセスメントパターンの 2 つを提案しました。また、提案技法を企業の商品開発・調達の業務システムの計画策定に適用したケーススタディを紹介しました。

一方で提案技法には課題も数多く残っています。提案技法が対象とする開発プロセスの初期段階では対象システムに対する財務計画やプロジェクト基本計画など、BMM で既定する内容以外にも多くの内容を定義しなければいけません。そのため、メタモデルおよび提案技法を拡張する必要があるといえます。また、提案技法がビジネスプランの見直しを支援することについては担当者の評価コメントに基づく定性的な効果は把握できましたが、定量的な効果の測定はできていません。技法適用に伴うビジネスプランの策定・運用業務の効率化やコスト削減等を今後検証することも必要です。

参考文献

[1] 情報処理推進機構ソフトウェアエンジニアリングセンター (編集), 経営者が参画する要求品質の確保, 第 2 版, オーム社, 2006
[2] 斎藤忍、小橋哲郎, メタモデルを活用した要求分析技法の適用と考察、SEC Journal, No.15, 2008, pp.15-22.
[3] Business Modeling & Integration Domain Task Force, Business Motivation Model, Version 1.1, 2008, http://bmi.omg.org/

— 禁 無 断 転 載 —

要求工学知識体系（REBOK）と
ユーザ指向要求工学の調査研究
ーユーザとベンダのかけ橋となる要求工学の実践ー

平成21年10月発行
発行所：社団法人　情報サービス産業協会
〒104-0028　東京都中央区八重洲2-8-1　日東紡ビル9F
TEL（03）6214-1121（代表）FAX（03）6214-1123
© Copyright, 2009 ; JISA, All Rights Reserved