超上流工程の要求定義を変革する
環境変化への取組みガイド

～要求発展型開発 WG 2012 年度活動報告書～

2013 年 3 月 27 日

独立行政法人 情報処理推進機構
技術本部 ソフトウェア・エンジニアリング・センター

はじめに

IPA/SEC では、日々変化する社会環境やビジネスニーズ等に柔軟に対応するための情報システム構築技術の適用に関する指針作りに向けた調査を実施し、環境の変化に対応する情報システムの構築に関する技術を整理し、体系化を行い、その結果をガイド（報告書）にまとめました。

また、環境の変化に対応するために用いられる現在の情報システム構築技術において、解決されない課題として認識されるテーマについて整理し、その結果をガイド（報告書）にまとめました。

本活動は、「2012 年度システムエンジニアリング実践拠点事業」として、調査については IPA/SEC が実施し、検討及び整理についてはエンタプライズ系ソフトウェア開発力強化推進委員会要求発展型開発ワーキンググループが実施しました。

超上流工程の要求定義を変革する環境変化への取組みガイド

～ 要求発展型開発 WG 活動報告書 ～

独立行政法人情報処理推進機構

目次

1. 背景と目的

社会環境、経済環境、IT 環境が相互に関連し不確実性を増しながら大きく激しく変化している。それも先の見えない不連続な変化である。社会インフラとなり得る企業の情報システムも規模が大きくなり、それに伴い多くのステークホルダが関与することになり、複雑さを増している。複雑さが増すにつれ、システム障害が発生する危険性も高まっている。

このような状況の中で、情報システムは信頼性を保ちつつ環境の変化に適応し続けなければ競争の足手まといになり、企業の競争力が低下してしまう。そのため、情報システムを環境変化に適応するように、各企業は既存の技術で様々な工夫を行っている。

これらの工夫を整理しモデル等に可視化することで、環境変化に対応する信頼性の高い情報システムの構築の一助となることを目的とする。なお、本報告書では、市場環境などの外部環境の変化、組織変更などの内部環境の変化、テクノロジの変化を対象とする。

■想定読者

超上流工程に関与するユーザ系及びベンダ系担当者

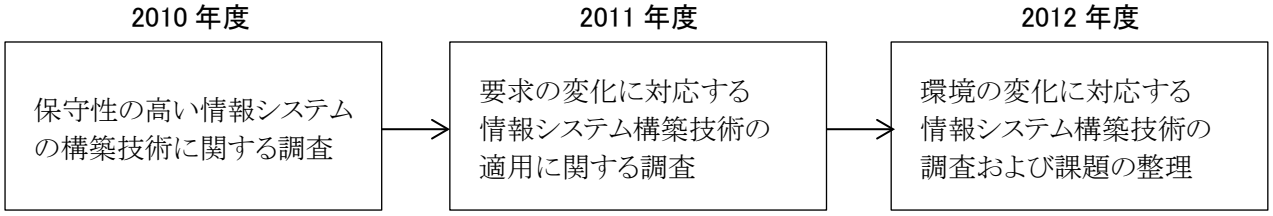
■目的

環境変化に対応する情報システムの構築の一助となるため

■狙い、または想定する効果

・ビジネス戦略と情報システム構築の乖離をできるだけ少なくする
・変化に対応すべき領域を見極め、環境変化に柔軟に対応する
・予測できる環境変化を考慮することで、運用や保守の行いやすい情報システム開発を行う
・環境変化を把握することで既存システムの改修箇所の妥当性を判断する
・ビジネス環境の変化を考慮したユーザ受け入れテストを設計する

2. WG 活動概要



2.1. 2010 年度、2011 年度の取組み

2010～2011 年度にかけて有識者に保守性の高い情報システム構築技術についてヒアリング調査を実施した結果、次の技術が必要であることが分かった。

2010 年度	2011 年度
開発における技術課題	開発における技術課題
要求管理	要求管理
要求の明確化	要求の明確化
スコープ決め	スコープ決め
トレーサビリティ管理	トレーサビリティ管理
―	ベースライン管理
	コントロールケースの活用
部品化・連携	システム構成関連技術（部品化・連携）
構造の明確化	部品の構造化
部品化	部品化
連携	―
	クラウドの活用
ソフトウェアのアシュアランス・品質保証	安全と情報セキュリティ
テスト網羅性の保証	テスト網羅性の保証
共通基盤としての保証	共通基盤としての保証
他社に求める保証	他社に求められる保証
―	不測の管理・仮定の管理
	運用性、可用性、アシュアランスの評価
	上流工程における横断的連携とシステム検証技術
	上流工程開発力強化
	ロバストネスの確保
	非ウォーターフォール型開発技術
	要求の明確化
	構成管理
	短期間での開発

					反復による開発
					品質管理
―			ビジネス IT オペレーションにおける技術課題		
					超上流工程分析・評価（システムズエンジニアリング）
			利用・運用における技術課題		
					運用技術

2.2. 2012 年度 WG 活動

2012 年 6 月より、WG 会合を開催し、環境の変化に対応した要求発展型開発に関する検討を行った。

第 1 回開催: 2012 年 6 月 18 日
第 2 回開催: 2012 年 7 月 30 日
第 3 回開催: 2012 年 8 月 9 日
第 4 回開催: 2012 年 9 月 11 日
第 5 回開催: 2012 年 10 月 30 日
第 6 回開催: 2012 年 11 月 19 日
第 7 回開催: 2012 年 12 月 6 日
第 8 回開催: 2013 年 1 月 21 日
第 9 回開催: 2013 年 2 月 4 日
第 10 回開催: 2013 年 3 月 7 日

【要求発展型開発 WG 委員リスト】 敬称略

	氏名	所属
主査	山本 修一郎	国立大学法人名古屋大学
委員	秋山 浩一	富士ゼロックス株式会社
委員	小林 茂憲	日本電気株式会社
委員	斎藤 忍	株式会社エヌ・ティ・ティ・データ
委員	白坂 成功	慶應義塾大学 大学院
委員	成瀬 泰生	富士通株式会社

委員	古川 正伸	株式会社東京証券取引所
委員	山本 久好	日本アイ･ビー･エム株式会社
委員	鷲崎 弘宜	学校法人早稲田大学
オブザーバ	青木 保壽	富士通株式会社
オブザーバ	宮本 祐子	独立行政法人宇宙航空研究開発機構

2.3. 2012 年度の取組みと要求のスコープ

本年度は図 2-1 で示すようにステークホルダ要求に影響を与える環境の変化に着目しその対応について議論を進めてきた。本報告書における環境の変化とは企業における外部環境と内部環境の変化のことである。

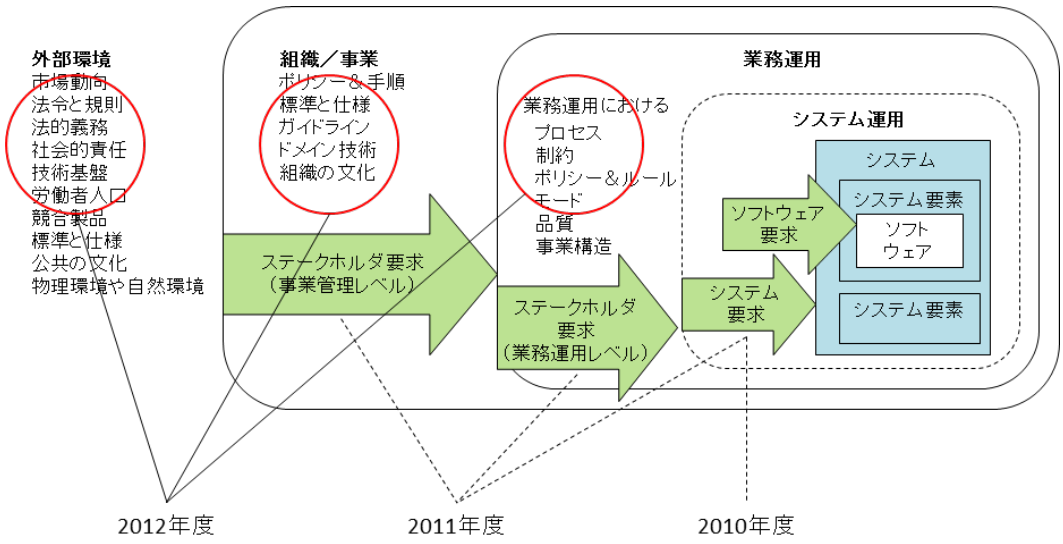


図 2-1 WG実施年度とビジネスコンテキスト[1]

表 2-1 表 WG 実施年度と要求のスコープ

	2012 年度	2011 年度	2010 年度
要求のスコープ	環境変化に伴う ステークホルダ要求	ステークホルダ要求 システム要求	システム要求

ステークホルダ要求（事業管理レベル）、ステークホルダ要求（業務運用レベル）を整理する[2]と図 2-2 のとおりになる。

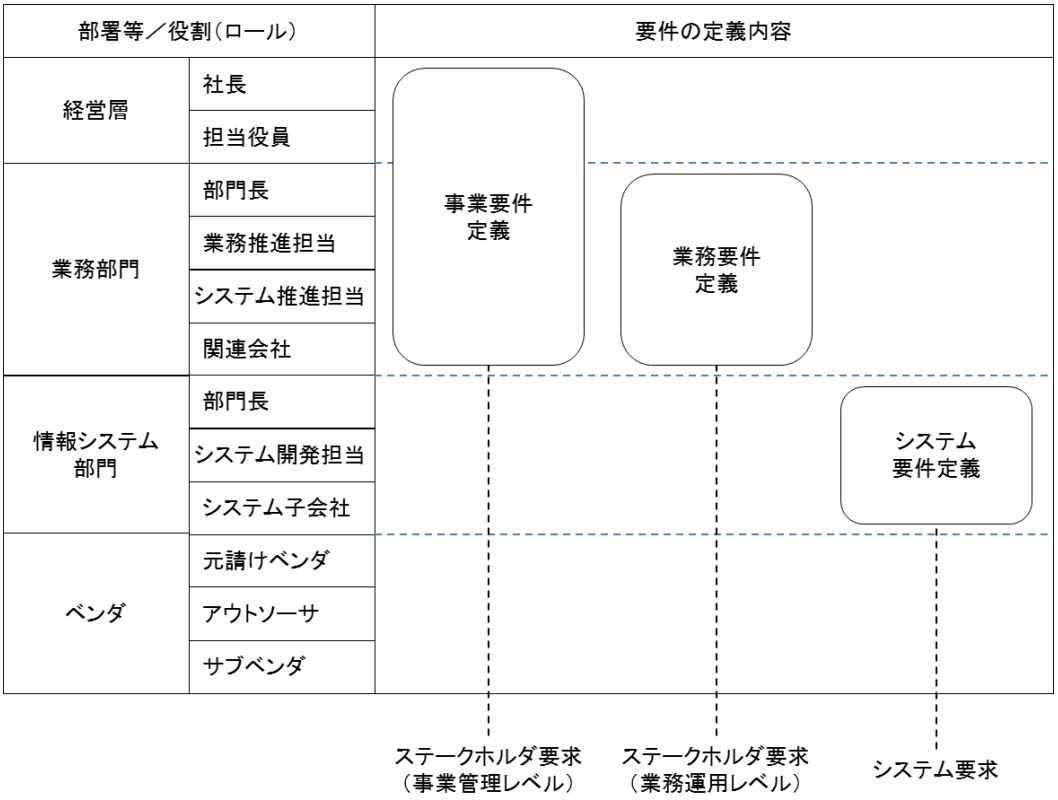


図 2-2 要件の定義と要求のスコープ[2]

事業要件定義、業務要件定義、システム要件定義の内容は次のとおりである。

表 2-2 要件の定義[2]

名称	項目	内容
事業要件定義	ビジネスモデルの検討 等	新規事業／社外連携／組織改編／部門間業務分掌変更／現行業務踏襲／セキュリティなど
業務要件定義	業務モデルの検討 等	業務内容（手順、責任・権限など）、業務形態（ピークなど）、業務品質、性能目標、運用、移行条件、セキュリティなど
システム要件定義	システムモデルの検討 等	システム構成、業務アプリケーション（構造、DB・ファイル構造など）、運用、移行要件、セキュリティ、機密情報保護対策など

本報告書では、ステークホルダ要求（事業管理レベル）を視野に入れつつ、主にステークホルダ要求（業務運用レベル）とシステム要求を対象とする。事業管理レベルと業務運用レベルを明確に区別しない時は、ステークホルダ要求と記述する。なお、『共通フレーム 2007 第 2 版』と同様に要求と要件を区別をせず、特別な理由がなければ「要求」を用いる。

[1] ISO/IEC/IEEE29148, Systems and software engineering – Life cycle processes – Requirements engineering Figure 4 – Example of requirements scope in a business context より作成

[2] SEC BOOKS 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～ 独立行政法人情報処理推進機構 ソフトウェア・エンジニアリング・センター, オーム社, 2006, p.42, 図 4.1 要件の定義と役割より作成

2.4. 不確実性の 4 レベルと WG の取組み

2.4.1. 不確実性の 4 レベルと戦略姿勢

企業や組織は環境変化に対応しなければ競争力が低下してしまうため、環境変化を明確に識別しようとする。環境の変化を読み取り、その変化に応じてステークホルダ要求が変化する。

本報告書では、環境変化からステークホルダ要求が変化する過程、つまり、企業や組織がどのように環境変化に対応するのかをコートニー等[1]の不確実性レベルと戦略姿勢に基づいて整理する。不確実性は四つのレベルに分けられ(表 2-3)、それぞれのレベルに対して戦略姿勢(Strategic posture)(表 2-4)をとることができる。

表 2-3 不確実性の 4 レベル

レベル	名称	内容
レベル 1	確実に見通せる未来 (A Clear-Enough Future)	ある程度の正確さで未来を予測できるレベル
レベル 2	他の可能性もある未来 (Alternate Future)	異なる複数のシナリオとして未来を描き出すことができるレベル。 方向性が見えるレベル。
レベル 3	可能性の範囲が見えている未来 (A Range of Future)	起こりうる未来を一定範囲内で特定できるレベル。
レベル 4	全く読めない未来 (True Ambiguity)	様々な不確実性が相互に作用し予測できないレベル。 レベル 4 の状況は稀であり、時間が経過するにつれ、他のレベルに移行する。

これらの不確実性のレベルに対して取り得る姿勢は三つある(表 2-4)。

表 2-4 戦略姿勢

名称		内容
留保	プレー権を確保する (Reserve the Right to Play)	ゲームを継続するための投資を続け、チャンスを待つ。 「形成」または「適応」の意思決定を留保する。
適応	未来に適応する (Adapt to the Future)	市場の変化に素早く、柔軟に対応することを目的とする。
形成	未来を形づくる (Shape the Future)	自らが構想する新たな業界構造を実現することを目的とする。

[1] [新訳]不確実性時代の戦略思考, ヒュー・コートニー, ジェーン・カークランド, パトリック・ビゲリー, DIAMOND ハーバード・ビジネス・レビュー, July 2009, pp.64-81

2.4.2. 不確実性の 4 レベルと WG テーマ

2010 年度はシステム要求の保守に関する技術課題を整理し、2011 年度はシステム要求の変化の契機となるステークホルダ要求の明確化と要求変化に関する技術課題を整理した。2012 年度はステークホルダ要求の変化の契機となる環境変化の明確化とそれに対応する手法やモデル及び技術課題を整理しテーマごとにまとめた(表 2-5)。

表 2-5 2012 年度のテーマ

章	テーマ
3	様々な環境適応モデル
4	情報システム開発における変化適応モデル
5	アシュアランスケース
6	サービスデザイン
7	環境変化に対応するシステムに関する課題

本年度の取組みを不確実性の4レベルと戦略姿勢に対応させると表 2-6のとおりになる。前掲論文によると、レベル1の「留保」とレベル2と3の「形成」はないとのことなので網掛けにしている。なお、レベル 2 と 3 の「留保」に関しては、「形成」または「適応」の意思決定を留保することから、WG では取り上げていない。

表 2-6 不確実性レベル、戦略姿勢と WG 実施テーマ

<table>
<tr><th>レベル</th><th>留保</th><th>適応</th><th>形成</th></tr>
<tr><td>レベル 1</td><td rowspan="2"></td><td>・アシュアランスケース
（5 章で取り上げる）</td><td>・サービスデザイン
（6 章で取り上げる）</td></tr>
<tr><td>レベル 2</td><td rowspan="2">―</td><td rowspan="2">・変化適応モデル
（3～4 章で取り上げる）</td><td></td></tr>
<tr><td>レベル 3</td><td></td></tr>
<tr><td>レベル 4</td><td colspan="3">・環境変化に対応するシステム開発に関する課題
（7 章で取り上げる）</td></tr>
</table>

凡例：－：取り上げていない

3. 様々な環境適応モデル

分野を問わず様々な環境変化に適応するモデルが提唱されている。本章では6種類のモデルを取り上げ比較検討する。

3.1. 生存可能システムモデル(Viable System Model:VSM)

生存可能システムモデル[1](以下、VSMという)は、変化する環境の中で生き残るための組織のシステムの一つである。VSMは相互作用を持つ 5 つのサブシステムから構成される。VSMをシンプルなモデルで表現すると図 3-1 になる。

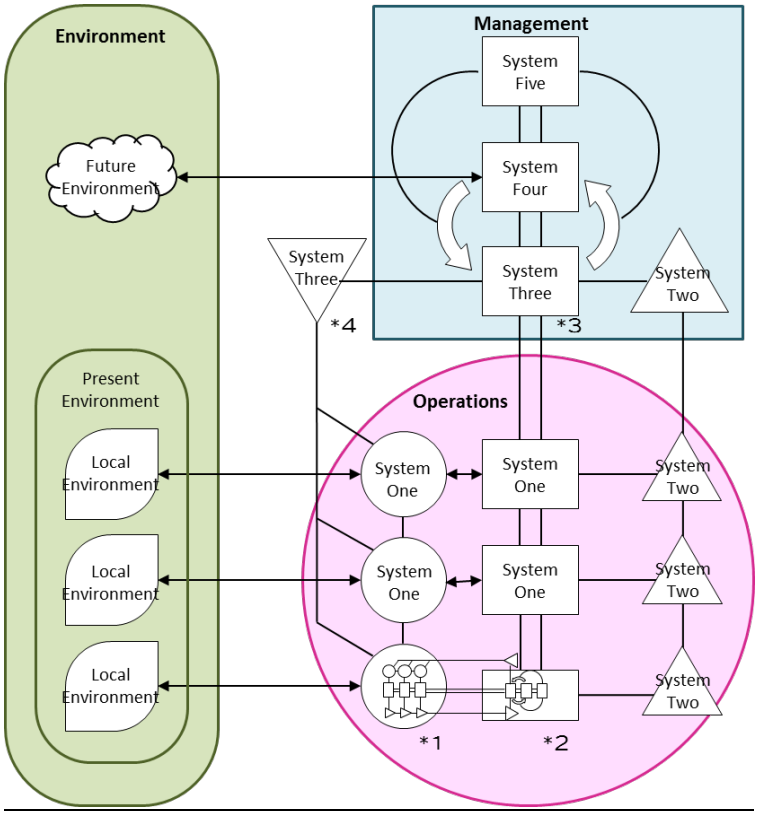


図 3-1 生存可能システムモデル[2]

■システム 1(S1) インプリメンテーション

対象システムの基本活動。運用を担当するシステムであり、直接的に環境とのインタフェースを持つ。複数の運用単位で構成される。S1 は実行機能を担当する実行システム(図 3-1 中の *1)と、実行機能を制御する規制システム(図 3-1 中の *2)からなる。

■システム 2(S2) コーディネーション

S1 及びシステム間を調整するためのシステム。運用単位間で相反する利害を持ったり、競合が発生した時に調整する。S1 がバラバラに動かないようにする。

■システム 3(S3)　コントロール

S1 と S2 を制御し監査するシステム。恒常性を維持し全体最適を考慮して資源獲得や資源配分を行う。統制を実行する制御システム(図 3-1 中の *3)と、状況をモニタリングする監査システム(図 3-1 中の *4)からなる。

■システム 4(S4)　インテリジェンス

外部環境の変化や内部環境の状況をモニタリングし、情報に対してインテリジェントに適応するシステム。S4 は S3 と多くの相互作用がある。戦略的意思決定と危機管理を行う。

■システム 5(S5)　ポリシー

様々な部分のバランスをとり、組織全体の方針を示すシステム。戦略と事業の方向性を提供する。全体方針の決定と舵取り。現在と未来のバランスをとる。

VSM は図 3-2 のように S1 の中に下位レベルの S1～5 が含まれる再帰システムである。

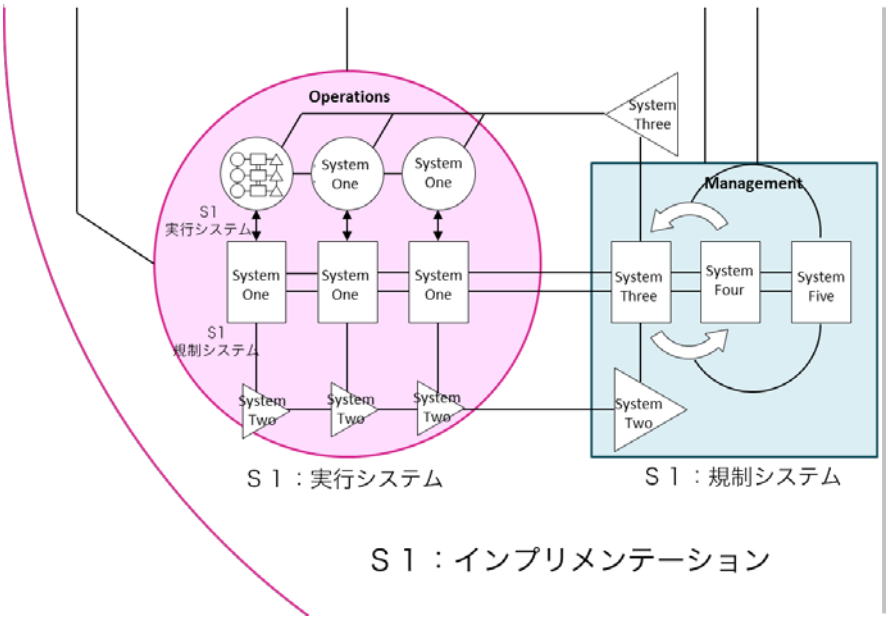


図 3-2 VSM と再帰

[1] Stafford Beer's Viable System Model, http://pdfsb.com/readonline/59466c4666673538585852364333566d56413d3d-5

[2] 企業組織のシステム判断，スタッフォード・ビーア，杉山書店, 1994, p.136 を参考に作成

3.2. ISO 9004:2009(JIS Q 9004:2010)

ISO 9004 は組織が自主的にQMSを構築し、改善するための指針である。2009 年の改正では、環境変化への対応を主たるテーマとしている。組織が持続的な成功を得るために、品質マネジメントアプローチによって環境変化に対応するための指針として改正された。持続的成功とは、「どのような経営環境の変化にあっても、持続的な事業の成功を成し遂げる」[1]ことである。

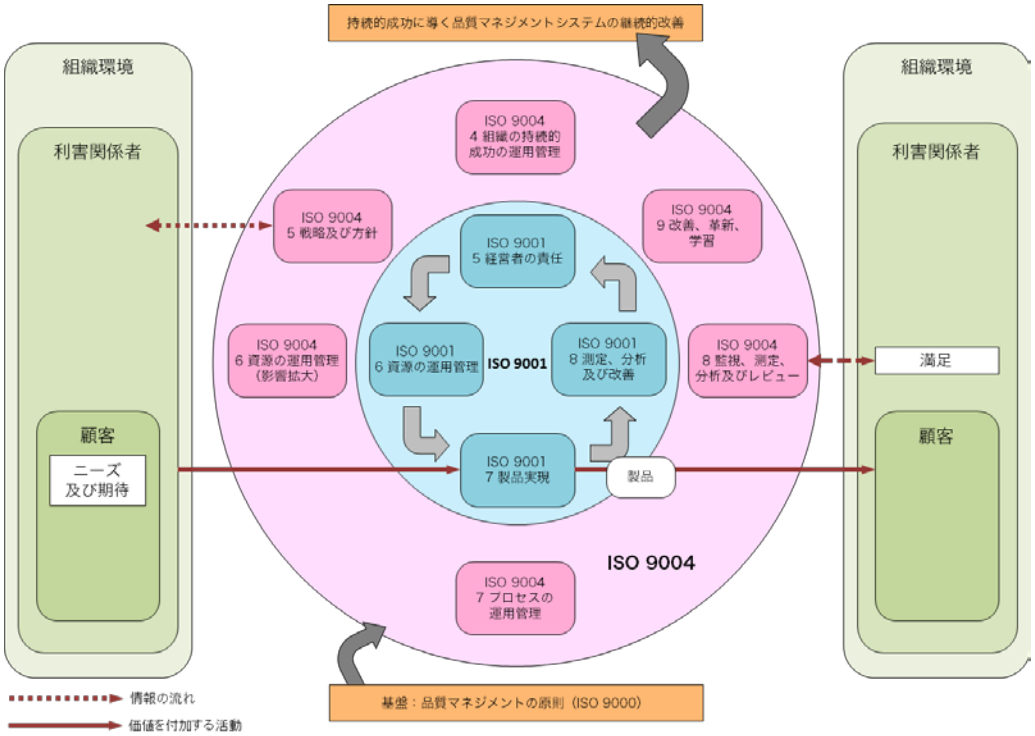


図 3-3 品質マネジメントシステムの拡大モデル[2]

VSM モデルと比較し、環境変化に ISO 9004:2009(JIS Q 9004:2010) がどのように対応できるかについて述べる。

■組織の持続的成功のための運営管理

組織にとって持続的成功にどのような価値があるのか、持続的成功を得るための基本的な概念が記述されている。これらのことから VSM のインテリジェンス(S4)と関係がある。

■戦略及び方針

ミッション、ビジョン、価値基準から戦略及び方針の策定、展開まで記述されている。これらのことから、VSM のポリシー(S5)と関係がある。

■資源の運用管理

外部及び内部の資源の種類及び資源の運用管理(計画、獲得、維持、改善)について記述されている。これらのことから、VSM のコントロール(S3)と関係がある。

■プロセスの運営管理

個々のプロセスのインプット、アウトプット及び必要な資源を明らかにし、測定方法を決め、プロセスの監視やプロセスを実施する責任や権限などのプロセスの運用管理について記述されている。これらのことから、VSM のコーディネーション(S2)に関係があり、一部はインプリメンテーション(S1)と関係がある。

■監視、測定、分析及びレビュー

ISO 9004 では組織の戦略レベルの監視、測定、分析、レビューについて記述されている。これらのことから、VSM のコントロール(S3)と関係がある。

■改善、革新及び学習

上記事柄を実践することで知識や知見を得る。組織として活用できるように学習し、得られた能力を組織の改善、革新に活かすことが記述されている。

上述したことをまとめると、表 3-1 になる。

表 3-1 VSM と ISO 9004:2009 の比較

VSM	説明	ISO 9004:2009
インプリメンテーション (S1)	対象システムの基本活動。 環境とのインタフェースを持つ。	7.プロセスの運用管理
コーディネーション (S2)	S1 及びシステム間を調整するための情報チャネル。	7.プロセスの運用管理
コントロール (S3)	S1 を制御するだけでなく、S2 を監査する。 システムの資源配置を維持する。 責任体制の構築と制御を行う。	6.資源の運用管理 8.監視、測定、分析及びレビュー
インテリジェンス (S4)	S4 は外部環境の変化に対してインテリジェントな適応を行う。 S4 は S3 と多くの相互作用がある。 戦略的意思決定と危機管理。	4.組織の持続的成功のための運営管理
ポリシー (S5)	S5 は様々な部分のバランスをとり、組織全体の方針を示す。 全体方針の決定と舵取り。 現在と未来のバランス。	5.戦略及び方針

[1] ISO 9004:2009 解説と活用ガイド，飯塚悦功 他，日本規格協会, 2011, p.44

[2] ISO 9004:2009 解説と活用ガイド，飯塚悦功 他，日本規格協会, 2011 p.74 図 1 プロセスを基盤とした品質マネジメントシステムの拡大モデルを引用

3.3. センス＆レスポンドの SIDA サイクル

環境の変化が頻繁に発生する状況では、事前に立案した計画どおりに進めることは難しい。計画よりも変化を感知して、その変化に対応することが大切であるという考えがスティーブ・ヘッケルのセンス&レスポンドである。SIDAサイクル[1]はセンス&レスポンドのコアをなすものである。

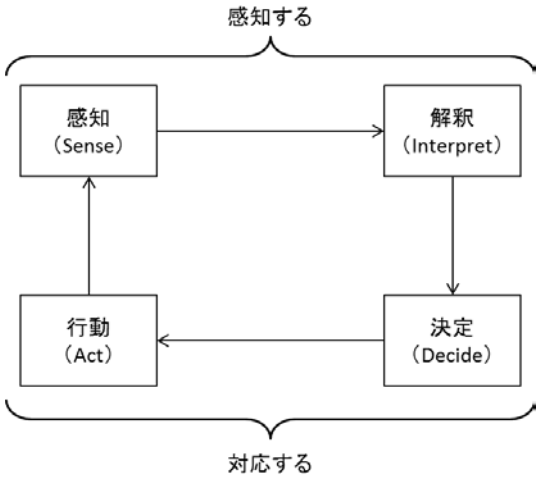


図 3-4 SIDA サイクル

SIDA サイクル（図 3-4）とは、外部環境や内部環境の変化を感知（Sense）し、従来の経験に基づいて解釈（Interpret）し、変化に対してどのように対応するかを決定（Decide）し、行動（Act）するという一連のサイクルを繰り返すモデルである。変化を予測するのではなく、変化の兆候を捉え、変化に対応することを重視する。

VSM モデルと比較し SIDA サイクルがどのように対応できるかについて述べる。

■感知：Sense

外部環境や内部環境の状況をモニタリングし、閾値を超えた出来事の発生を感知し、認識する。ノイズとシグナルを区別するのが大切である。これらのことから VSM のインプリメンテーション(S1)と関係がある。

■解釈：Interpret

これまでの経験に基づいてイベントやシグナルまたは状況変化を解釈する。感知した状況の意味を把握し、生データを抽象度の高いパターンに変えることで無用な情報を切り捨て、意思決定に必要な分析を行う。システム間を調整するための情報チャネルを考慮していると考えられるため、VSM のコーディネーション(S2)と関係がある。

■決定：Decide

解釈に基づいて、どのように対応するかを決定する。戦略的な意思決定である。結論に達するだけでなく、資源の割り当ても行う。これらのことから VSM のコントロール(S3)及びインテリジェンス(S4)と関係がある。

■行動：Act

必要に応じて新たな能力を調達し、対応策を実行する。決定されたことを実行に移す前に、企業方針と一致しているかどうかを確認する。対応策を実施することにより変化した状況に対して再び情報を収集する。これらのことから VSM のインプリメンテーション(S1)と関係があり、一部に VSM のポリシー(S5)と関係がある。

上述したことをまとめると、表 3-2 になる。

表 3-2 VSM と SIDA サイクルの比較

VSM	説明	SIDA サイクル
インプリメンテーション (S1)	対象システムの基本活動。 環境とのインタフェースを持つ。	感知（Sense） 行動（Act）
コーディネーション (S2)	S1 及びシステム間を調整するための情報チャネル。	解釈（Interpret）
コントロール (S3)	S1 を制御するだけでなく、S2 を監査する。 システムの資源配置を維持する。 責任体制の構築と制御を行う。	決定（Decide）
インテリジェンス (S4)	S4 は外部環境の変化に対してインテリジェントな適応を行う。 S4 は S3 と多くの相互作用がある。 戦略的意思決定と危機管理。	決定（Decide）
ポリシー (S5)	S5 は様々な部分のバランスをとり、組織全体の方針を示す。 全体方針の決定と舵取り。 現在と未来のバランス。	行動（Act）の一部

[1] 適応力のマネジメント，スティーブ・ヘッケル，ダイヤモンド社, 2001, p.113 図表 4-1 SIDA サイクルより作成

3.4. BCG アダプティブ戦略の学習プロセス

ボストン コンサルティング グループ（BCG）の戦略スタイルの一つにアダプティブ戦略[1]がある。アダプティブ戦略とは、環境を予測することが難しく環境に働きかけて変化させることも難しい場合に採用する戦略であり、環境の急激な変化に適応するために用いる。アダプティブ戦略の特徴は四つのステップからなる学習プロセスにある。

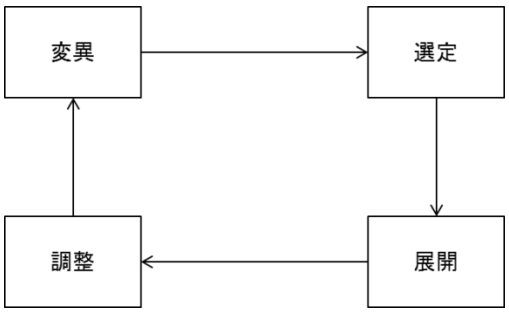


図 3-5 BCG アダプティブ戦略 学習プロセス

VSM モデルと比較し環境変化に BCG アダプティブ戦略の学習プロセスがどのように対応できるかについて述べる。

■変異

斬新なものを創造することにより、変化する環境に対処する。これらのことから、VSM のポリシー(S5)と関係がある。

■選定

変異により生じたものの中から将来的に最も有望なものを選択する。これらのことから、VSM のコントロール(S3)と関係がある。

■展開

選定されたものを拡大展開し、最適化する。必要に応じて、それを組織に根付かせる。これらのことから、VSM のインプリメンテーション(S1)と関係がある。

■調整

環境や企業目標の変化に合わせて、学習プロセス全体を調整する。これらのことから、VSM のコーディネーション(S2)と関係がある。

上述したことをまとめると、表 3-3 になる。

表 3-3 VSM と BCG アダプティブ戦略の学習プロセスの比較

VSM	説明	学習プロセス
インプリメンテーション (S1)	対象システムの基本活動。 環境とのインタフェースを持つ。	展開

コーディネーション (S2)	S1 及びシステム間を調整するための情報チャネル。	調整
コントロール (S3)	S1 を制御するだけでなく、S2 を監査する。 システムの資源配置を維持する。 責任体制の構築と制御を行う。	選定
インテリジェンス (S4)	S4 は外部環境の変化に対してインテリジェントな適応を行う。 S4 は S3 と多くの相互作用がある。 戦略的意思決定と危機管理。	ー
ポリシー (S5)	S5 は様々な部分のバランスをとり、組織全体の方針を示す。 全体方針の決定と舵取り。 現在と未来のバランス。	変異

[1] ボストン コンサルティング グループ 展望 176:なぜ戦略に戦略が必要なのか, http://www.bcg.co.jp/documents/file107198.pdf

3.5. DEOS プロセス 目的・環境変化対応サイクル

DEOS プロセスを図 3-6 に示す。

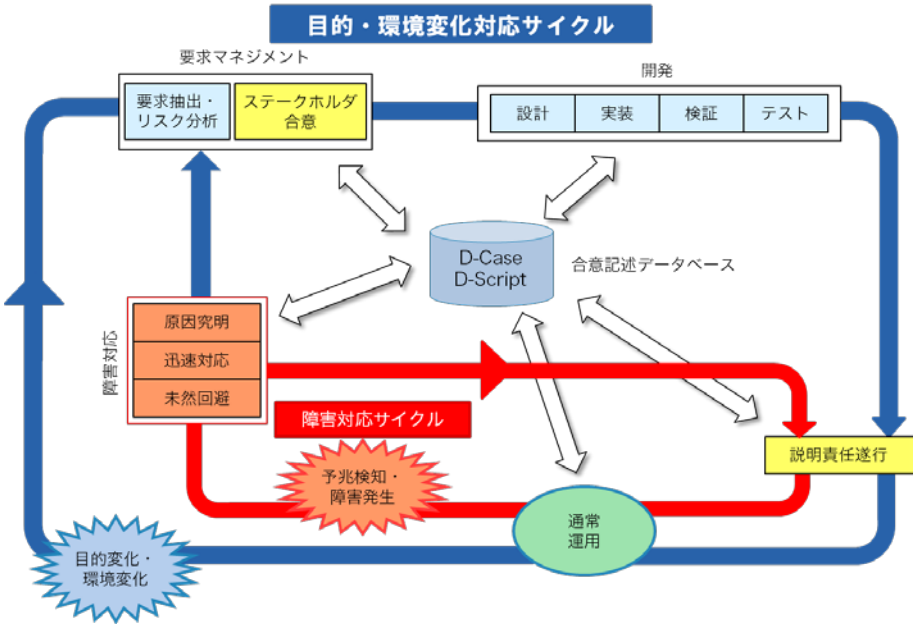


図 3-6 DEOSプロセス[1]

DEOS プロセスには、目的・環境変化対応サイクルと障害対応サイクルがある。DEOS プロセスの変化対応サイクルでは、目的変化・環境変化に対する合意形成を可能とする要求マネジメントプロセスと変化に対応するための開発プロセスを用意している。また、DEOS プロセスの障害対応サイクルでは、障害の予兆検知と障害発生に対する障害対応プロセスを用意している。これにより、システムを維持するための責任体制を構築して制御していると考えることができる。

以下では、VSM モデルと比較することによって環境変化にどのようにこの DEOS プロセスが対応できるかについて述べる。

■要求マネジメントフェーズ

DEOS プロセスの要求マネジメントでは、D-Case を用いて、システムが持つ現在の問題を解決するための開発内容に関する要求を抽出してステークホルダと合意形成することができる。これにより、システムの現在と未来をバランスさせる全体方針を決定することから、システム発展のための統制能力がある。

■説明責任遂行フェーズ

DEOS プロセスの説明責任遂行と障害対応によって、システムの危機管理に対応できる。システム発展のためのインテリジェンス能力がある。

■開発・通常運用フェーズ

DEOS プロセスには、開発と通常運用が含まれているので、対象システムの基本活動が考慮されている。

■変化対応・障害対応のための認識フェーズ

DEOS プロセスでは、目的変化・環境変化並びに予兆検知・障害発生を認識できることから、システム変化を調整するための情報チャネルを考慮している。

上述したことをまとめると、表 3-4 になる。

表 3-4 VSM と DEOS プロセスの比較

VSM	説明	DEOS プロセス
インプリメンテーション (S1)	対象システムの基本活動。 環境とのインタフェースを持つ。	開発・通常運用
コーディネーション (S2)	S1 及びシステム間を調整するための情報チャネル。	目的変化・環境変化 予兆検知・障害発生
コントロール (S3)	S1 を制御するだけでなく、S2 を監査する。 システムの資源配置を維持する。 責任体制の構築と制御を行う。	変化対応サイクル 障害対応サイクル
インテリジェンス (S4)	S4 は外部環境の変化に対してインテリジェントな適応を行う。 S4 は S3 と多くの相互作用がある。 戦略的意思決定と危機管理。	説明責任遂行 障害対応
ポリシー (S5)	S5 は様々な部分のバランスをとり、組織全体の方針を示す。 全体方針の決定と舵取り。 現在と未来のバランス。	要求マネジメント 合意記述データベース

[1] オープンシステムディペンダビリティの実現, DEOS プロジェクト White Paper Version 3.0, 科学技術振興機構, 2011

3.6. ITIL®の ITSM モニタ・コントロール・ループ

環境変化対応と目的変化対応の両方を視野に置くモデルとして、ITIL®の ITSM モニタ・コントロール・ループ・モデルを示す。ITSM とは IT サービスマネジメントの略であり、ITSM モニタ・コントロール・ループ・モデルは、日々の IT サービスマネジメント活動における基本的なコントロール活

動が、どのようにして、より大きなサービスマネジメント・ライフサイクルとダイナミックに連携するのかを表したものである。IT サービスマネジメントの目的は、事業のニーズとそれを支えるサービスとしてのITを常に整合させることであり、ITSMモニタ・コントロール・ループは、IT サービス提供者が組織として、それをどのように実現するのかをモデル化したものである。

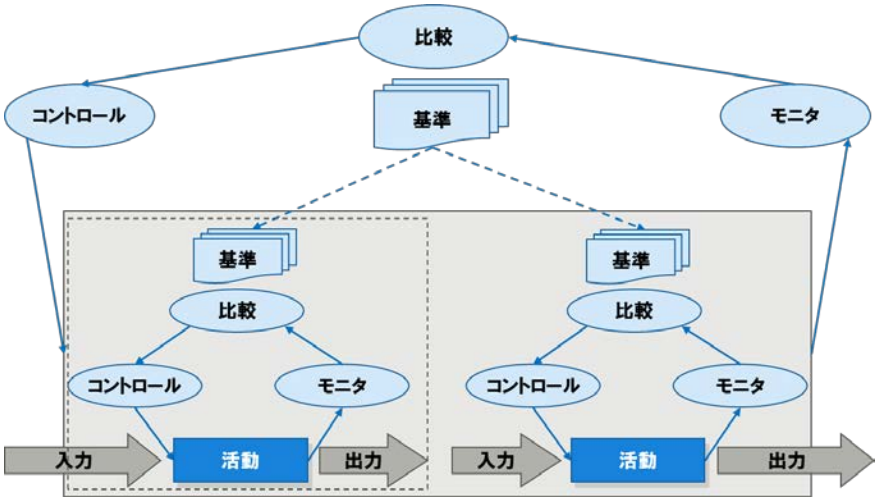


図 3-7 モニタ・コントロール・ループの基本構造

図 3-7 にモニタ・コントロール・ループの基本構造を示す。左下の点線四角の中に、モニタ・コントロール・ループの基本形が示されている。対象となる活動の「アウトプットをモニタリングし、そのアウトプットを事前に定義された基準と比較し、その比較に基づいて適切な処置を行う」(『ITIL2011 用語集』 モニタ・コントロール・ループの定義)というモデルである。

ポイントは、この単純な基本形が横に連結し、さらに、それらを入れ子構造として縦に繋げることで、組織体における IT サービスマネジメントのダイナミクスを、より現実的に表そうとしている点にある。横に連結することにより、個々の活動はプロセスとなる。プロセスは、より上位のレベルから示される基準を介して、より上位のモニタ・コントロール・ループのコントロールを受ける。ITIL®では、マネジメントのレベルを、戦略、戦術、運用の三つに分けており、ここでのレベルは、例えば、それらに該当すると考えてもよい。

この構造のキーとなるものが、基準である。現実には、基準は、方針、標準、目的、目標、KPIなど、様々な形をとる。ITIL®では、その基準の階層を意識し、それらの繋がりを組織として明確にすることが重要であるとしている。図 3-8 に基準の階層構造を示す。

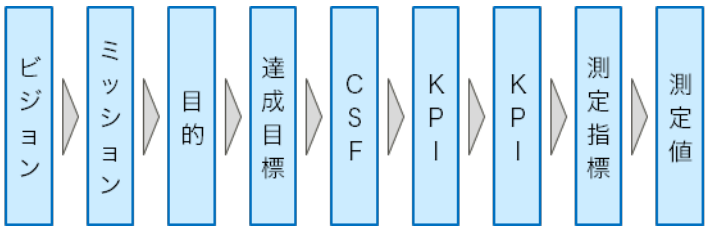


図 3-8 基準の階層構造

図 3-9 は、モニタ・コントロール・ループを使って、サービスマネジメント・ライフサイクルを表現したものである。これを、本 WG の「環境変化への対応」「目的変化への対応」という二つの観点で解釈すると以下のようになる。

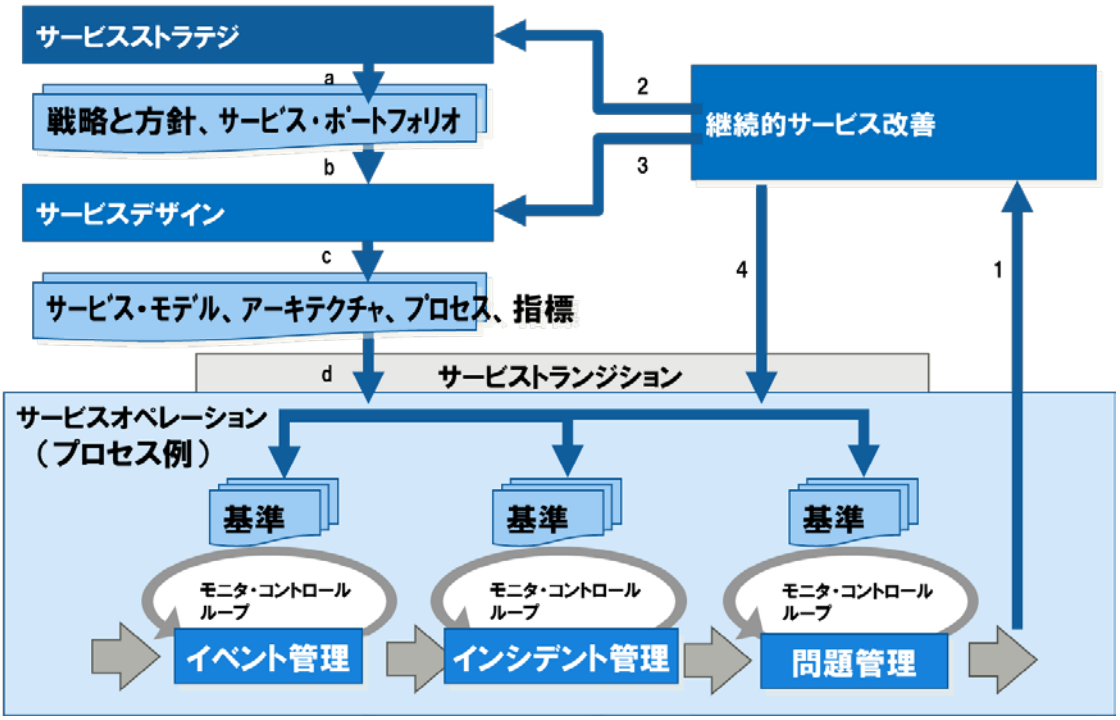


図 3-9 ITSM モニタ・コントロール・ループ

「環境変化への対応」は、図の右側の部分に表される。日々のモニタ・コントロール・ループによる活動のアウトプットから「継続的サービス改善」に環境変化の情報が集約される（矢印 1）。さらに、それらは、戦略レベルの検討を必要とするものは「サービスストラテジ」へ（矢印 2）、設計レベルの検討を必要とするものは「サービスデザイン」へ（矢印 3）、日々の活動レベルで対応すべきものは、活動の「基準」（矢印 4）へフィードバックされ、適切な環境変化への対応が実行される。

一方、「目的変化への対応」は、図の左側の部分に表される。まず、「サービスストラテジ」において、目的変化への対応戦略や方針が検討され（矢印a）、サービス・ポートフォリオ（IT 投資計画など）を通じて（矢印 b）、「サービスデザイン」に通知される。「サービスデザイン」では、それに対応して、サービス・モデルの追加や変更を検討し、アーキテクチャやプロセスを設計し、パフォーマンス指標を変更する（矢印 c）。さらに、日々の活動の基準が変更され（矢印 d）、「サービスオペレーション」における日々の活動レベルにまで変化が伝播することになる。

このように、「継続的サービス改善」と「基準の階層構造」を潤滑油として、ITSM・モニタ・コントロール・ループが機能することによって、IT サービスマネジメントの「変化への対応」がダイナミックに実現される。

上述のアクティビティと VSM を比較すると、表 3-5 になる。

表 3-5 VSM と ITSM モニタ・コントロール・ループの比較

VSM	説明	モニタ・コントロール・ループ
インプリメンテーション (S1)	対象システムの基本活動。 環境とのインタフェースを持つ。	活動
コーディネーション (S2)	S1 及びシステム間を調整するための情報チャネル。	改善

コントロール (S3)	S1 を制御するだけでなく、S2 を監査する。 システムの資源配置を維持する。 責任体制の構築と制御を行う。	コントロール
インテリジェンス (S4)	S4 は外部環境の変化に対してインテリジェントな適応を行う。 S4 は S3 と多くの相互作用がある。 戦略的意思決定と危機管理。	モニタ
ポリシー (S5)	S5 は様々な部分のバランスをとり、組織全体の方針を示す。 全体方針の決定と舵取り。 現在と未来のバランス。	基準

3.7. 各環境適応モデルの評価

様々な環境適応モデルを分析すると、環境の変化にボトムアップで対応しているか、目的変化にトップダウンで対応しているかによってモデルを特徴づけられる。また、モデルの構造として再帰性（入れ子構造）を持っているかどうかによっても分類することができる。

各環境適応モデルを評価すると表 3-6 になる。

表 3-6 各環境適応モデルの評価

環境適応モデル	環境変化対応 （ボトムアップ）	目的変化対応 （トップダウン）	再帰性 （入れ子構造）
VSM	○	○	○
ISO 9004		○	
SIDA サイクル	○		
学習プロセス	○		
DEOS プロセス	○	○	
ITSM モニタ・コントロール・ループ	○	○	○

これらの評価を踏まえ、環境変化に対応し目的変化にも対応し再帰性がある情報システム開発における変化適応モデルを次章で述べる。

4. 情報システム開発における変化適応モデル

4.1. 情報システム開発における変化適応モデル

文献・Web 調査及び WG 内の意見を反映して、以下に述べる情報システム開発に適用した変化適応モデルをとりまとめた。この整理はあくまで今回の調査や議論の中で、テーマに比較的合致したものを挙げており、網羅的にまとめていない。また、記述内容は仮説レベルのものであり、実証実験がなされていないことに留意して欲しい。

2011 年度はシステム要求の変化の一因と考えられるステークホルダ要求に着目し、ステークホルダの種類を整理した。2012 年度はステークホルダ要求の変化に影響を与えるビジネス環境に着目し検討を進めた。3 章で検討を加えた様々な環境適応モデルを参考に情報システム開発に適用した変化適応モデルを作成した(図 4-1)。

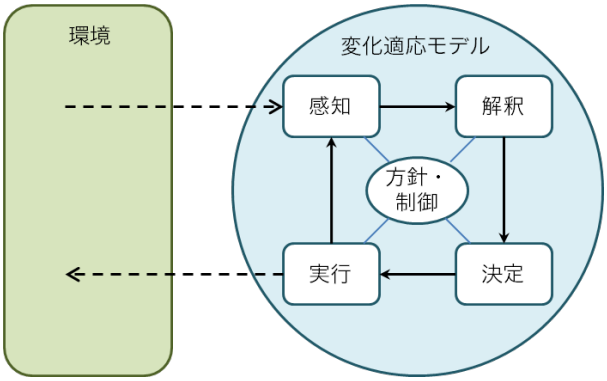


図 4-1 変化適応モデル

このモデルは再帰的であり、事業管理レベル、業務運用レベル、システム運用レベルで同型のサイクルが回る。図 4-2 に示すのは、この変化適応モデルと業務開発及び情報システム開発の関係である。環境変化に伴い企業のポリシーに基づき変化に対して感知・解釈・決定・実行を行う。実行の中には業務プロセスの構築・変更、情報システムの構築・変更が含まれる。

本章では主に業務運用レベルのステークホルダ要求を対象とする。この要求を扱うプロセスは、『共通フレーム 2007 第 2 版』ではステークホルダ要求を明らかにする要件定義プロセスに該当し、ISO/IEC/IEEE 29148 では業務運用レベルの要求プロセスが該当する。

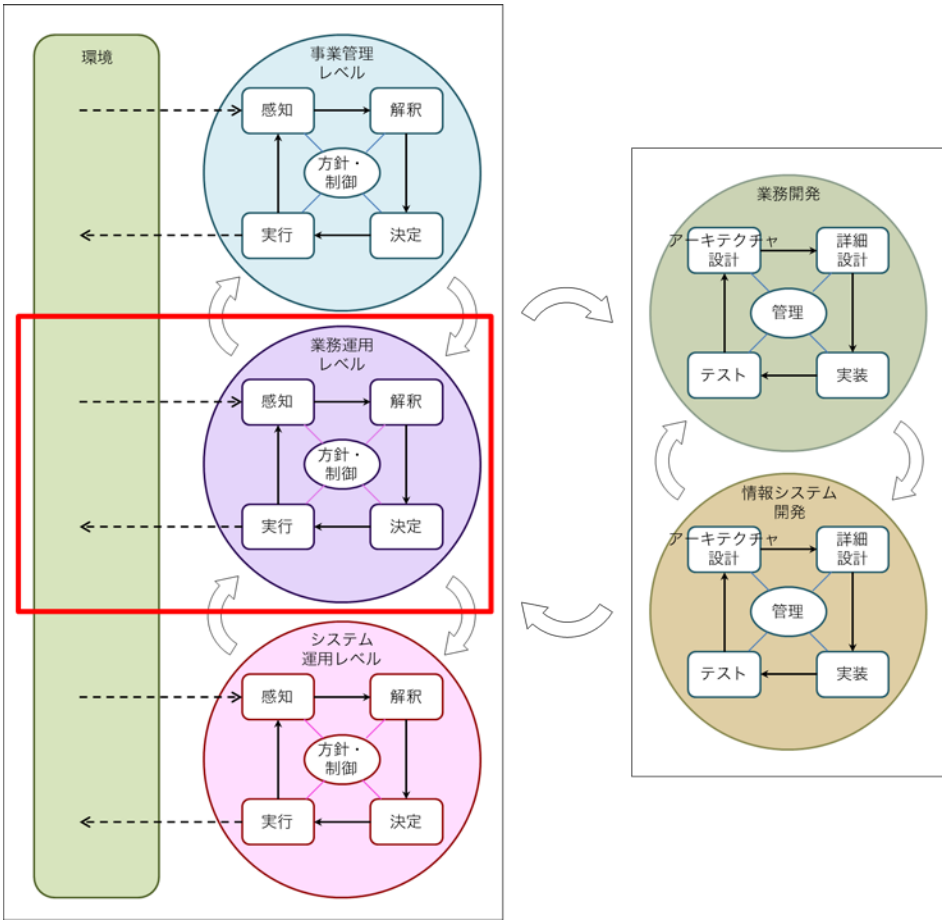


図 4-2 ビジネスコンテキストに合わせた変化適応モデルと情報システム開発

この業務運用レベルを詳細にすると図 4-3 になる。

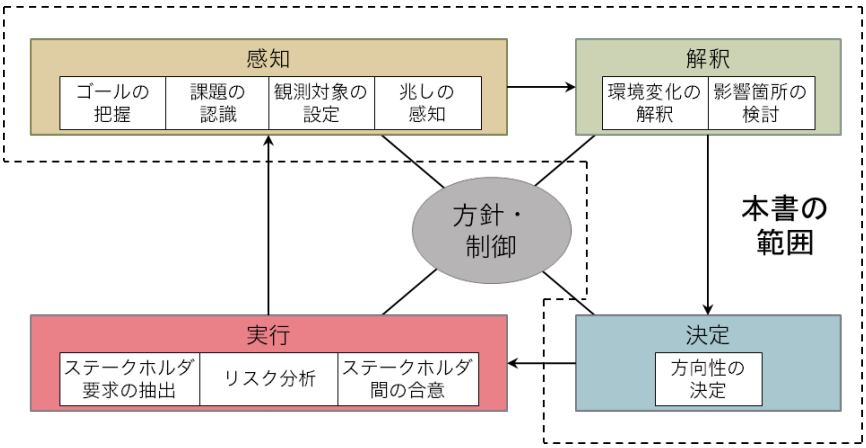


図 4-3 情報システム開発に適用した変化適応モデル

なお、「実行」に関しては 2011 年度に実施、検討を行ったために、本章では**「感知」「解釈」「決定」**を主に取り上げる。「方針・制御」に関しては今後の課題とする。

4.2. 感知

4.2.1. ゴールの把握

変化の兆しを感知するには課題の認識が必要である。しかし、その課題を認識できない場合が少なからずある。課題は組織のゴールや目標と現状のギャップにより識別されるため、課題の認識の前に、組織のゴールや目標を把握しておく必要がある。

4.2.2. 課題の認識

ユーザは様々な課題認識を持っており、それらの領域を整理すると、図 4-4 になる。

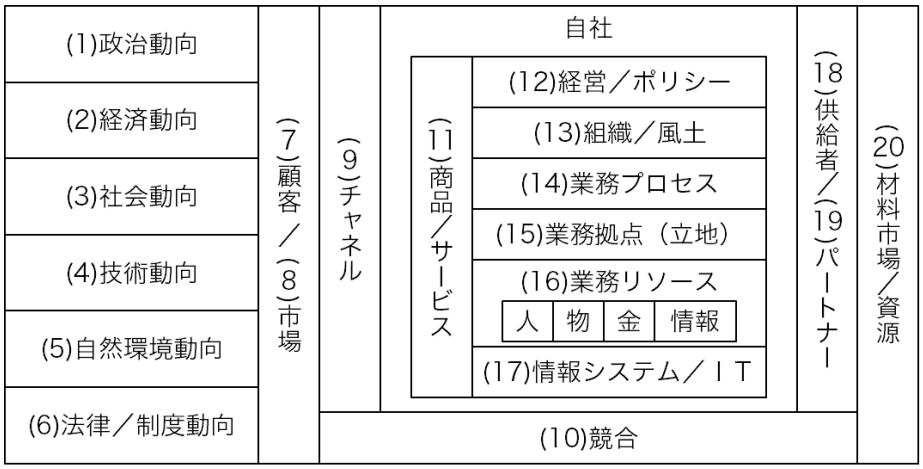


図 4-4 課題領域の分類

上記課題領域の分類に従って課題の例を挙げる(表 4-1)。

表 4-1 課題の例

	課題領域分類	課題の例
1	政治動向	政治的安定さ、金融政策や財政政策の動向
2	経済動向	経済成長率、金利、インフレ、景気の動向
3	社会動向	公共文化の変化、企業の社会的責任(CSR 向上)、労働者人口の変化、業界ルールの整備、商慣行の変化、産業構造の変化(基幹産業の交代、成長分野の変化など)
4	技術動向	技術基盤の変化、新技術体系への転換、技術移転の割合、同業他社の自動化レベル、標準化の動向、標準仕様への準拠
5	自然環境動向	気候変動(地球温暖化など)、リサイクル、環境保護のための規制、物理環境の変化
6	法律／制度動向	法整備、規制の緩和または強化、法令遵守、法的義務、消費者保護
7	顧客	顧客ニーズの多様化、顧客との関係性の強化または改善、新規顧客の獲得及び既存顧客の維持、顧客生涯価値(LTV)の重視、ターゲット顧客の変動
8	市場	市場シェアの拡大、国内市場の成長鈍化、新興国の発展、原材料やエネルギーの価格の変動及び獲得競争の変化、業界構造の変化、リーディング企業の盛衰、新たな合従連衡、事業領域の転換、グローバル化に伴う再編
9	チャネル	販売チャネルの選別や統廃合、インターネットチャネルの割合増加、ソーシャルネットワークや口コミの影響、営業情報共有による営業効率の向上

10	競合	競合製品、代替品
11	商品、サービス	魅力的な新商品の開発、市場やニーズに応じた商品やサービスの多様化、商品・サービスを市場に投入するまでの期間の短縮、付加価値の高い商品・サービスへのシフト、商品に付帯するサービスや関連商品やアフターサービスの充実、製品ライフサイクルの変化への対応、商品を含むエコシステムの構築、商品やサービスの品質向上、価格競争力の向上
12	経営／ポリシー	ポリシーの確立、経営方針の整備、全社的なリスクの管理、事業継続計画（BCP）の見直しまたは強化、経営の透明性の確保（内部統制、システム監査）、情報セキュリティの強化、経営スピードの向上、経営情報の一元化、迅速な業績把握や情報把握（リアルタイム経営、月次から日次把握へ）、実績情報の即時フィードバック、意思決定の迅速化、個別最適から全体最適へ、コアコンピタンスの見直し、事業再編、事業領域の転換
13	組織／風土	組織の文化、組織改革、社内コミュニケーション
14	業務プロセス	業務標準化、ガイドラインの整備、ビジネス・アーキテクチャの構築、業務プロセスの効率化（省力化、業務コスト削減）、業務プロセスのスピードアップ（リードタイム短縮など）、業務プロセスの質・精度の向上（ミス、欠品削減策）、業務ポリシー＆ルール、生産方式の改善による生産の効率化、シェアードサービス化、アウトソーシング化、ペーパーレス化、新商品の開発期間の短縮、在庫圧縮、物流効率化、調達コスト削減、社内コミュニケーションの強化
15	業務拠点（立地）	地域特性への柔軟な対応、グローバル対応、在宅勤務
16	業務リソース	社員の生産性向上、経営情報の共有化によるやる気向上、従業員削減、生産技術の社内空洞化防止
17	情報システム／IT	IT インフラ変更への対応
18	供給者	原材料の部品の仕入れ及び調達先の変更や範囲の拡大、調達・生産・物流におけるグループ外企業との連携
19	パートナー	外部経営資源との連携・活用、技術や特許を保有する企業の買収
20	材料市場／資源	原油価格、レアメタル調達先、電気代

4.2.3. 観測対象の設定

変化の兆しを感知するためには、課題領域を漠然と眺めるだけではなく、変化が生じると思われる箇所、つまり観測対象を設定する。観測対象の例を次に挙げる（表 4-2）。

表 4-2 観測対象[1]

分類		観測対象
顧客	顧客	・用途分析 ・顧客分析
チャネル	顧客とのインタフェース	・インタフェースの数（トップ、管理者、営業、技術者） ・頻度 ・カバー率 ・営業情報の共有化率 ・サービスの差別化
業務プロセス	社内処理	・情報伝達リードタイム
商品	設計（施策）	・設計リードタイム ・部品共通化率 ・新製品開発率
業務プロセス	計画 ↓ 購買 ↓ 加工	・生産リードタイム（工程別） ・機械稼働率 ・部品在庫回転率 ・ロスタイム ・製造原価

	↓ 組立 ↓ 出荷	・生産性 ・計画／変更のタイミング ・自動発注率 ・見積もり妥当性 ・外注先評価 ・購買貢献度
業務プロセス 業務拠点	デポ （小型の配送拠点）	・製品在庫回転率 ・物流費
顧客	顧客	・月次決算スピード ・売上高（地域、顧客、商品別） ・売上原価等 ・販管費等 ・営業利益率（粗利益、貢献利益） ・顧客満足度 ・納期遵守率 ・クレーム率 ・売掛金回収状況

また、観測対象はセンサデバイス及びセンサネットワークで収集したデータ、情報システムのアクセスログ、システムログ、システム運用時の障害情報なども観測対象になる。

[1] 企業経営情報システム構築の手ほどき，沢村淑朗，井上正和，同友館，1995，p.43，図表1-10 オンタイム度管理項目より作成

4.2.4. 兆しの感知

観測対象を設定した後、様々な手段を通じて情報を集める（表 4-3）。例えば、商品別顧客セグメント別売上高を観測ポイントに設定している時、「ある商品の 30 代女性の売上高が落ち始めている」「テレビや雑誌に競合製品（代替品）の広告が出稿されている」「口コミサイトで競合製品（代替品）の評判がよい」「SNS で競合製品（代替品）の商品名が取り上げられることが多い」「タレントブログで競合製品（代替品）が取り上げられた」などから、変化の兆しを感知することができる。

表 4-3 兆しを感知する手段の例

変化の兆しを感知する手段の例
・顧客の声が載っているアンケート結果
・第一線の営業マンの声
・お客様サービスセンターに寄せられた苦情内容
・商品の故障情報や修理情報
・自社で保有しているデータ（情報システムから出力される帳票類など）
・法規制の動向
・同業他社の動向や事例
・海外事例
・他業界の事例
・白書や業界団体がまとめた文献
・インターネットにある情報
・各種オンライン・データベース・サービス

・セミナーや展示会

また、システム運用におけるアクセス数、アクセスログやシステムログの変化、または、システム障害やインシデントの発生状況から変化の兆しを感知したり、システム開発におけるバグや問題票の起票状況、起票内容から変化の兆しを感知したりすることもある。例えば、アクセスログやシステムログから情報セキュリティの脅威の兆しを感知することもある。

4.3. 解釈

4.3.1. 環境変化の解釈

変化の内容を解釈するとともに、感知した変化の兆しを吟味し、変化の特性に応じた解釈を行う。変化の特性とは変化のスパンや頻度などである（表 4-4）。例えば、変化の頻度またはペースが速いユーザインタフェースに関連するものと、比較的遅いデータモデルに関連するものとでは、影響箇所が異なるため、変化の特性を分析し解釈する必要がある。

表 4-4 変化の特性

変化の特性	変化の解釈
変化のスパン／変化の持続性	・変化の幅がどの程度か。 ・変化が持続的に続くのか。 ・長期に亘って変化が継続するか。
変化の頻度／変化のペース	・変化の頻度はどのくらいか。 ・変化のペースはどの程度か。
変化のサイクル	・変化は周期的か。
変化の連続性	・変化は連続して発生するか、不連続で発生するか。
変化の規則性（規則的、不規則的）	・規則的な変化をするか、不規則的な変化をするか。
変化の確実性	・変化は確実に発生するか。
変化の広がり	・変化は波及するか、変化は局所的にとどまるか。

4.3.2. 影響箇所の検討

変化の特性を考慮しつつ変化の内容を見極めて影響箇所を検討する。影響箇所はビジネス領域及び情報システム領域である。

表 4-5 影響箇所の検討

変化の対象	影響箇所
ビジネス（経営、業務）	経営者、従業員、ビジョン、目的、企業戦略、組織構成、社内制度、業務プロセス、ビジネス・アーキテクチャ、商品サービス、財務／会計、立地ロケーション
情報システム	業務アプリケーション、アプリケーション・アーキテクチャ、データ・アーキテクチャ、テクノロジ・アーキテクチャ、施設（データセンター）

4.4. 決定

4.4.1. 方向性の決定

方針を決定する前に、次の観点(表 4-6)から変化対応の手間や難易度を判断する。

表 4-6 対応する際に考慮する点

変化に対応する際に考慮する点	内容
変化の対応レベル	・変化は経営レベルに影響を及ぼすか。 ・変化は事業レベルに影響を及ぼすか。 ・変化は業務レベルに影響を及ぼすか。
想定する影響箇所	・変化対応の規模 ［大きいー小さい］ ・変化対応の範囲 ［広いー狭い］ ・変化対応、導入のスピード ［速いー遅い］ ・変化対応の進め方 ［一気にー徐々に］ ・変化対応時の試行錯誤の回数 ［多いー少ない］ ・変化を予想した変化対象の余裕(スラック)の程度 ［多いー少ない］
想定する影響箇所がシステムの場合	・過去にあった機能変更の頻度 ［多いー少ない］ ・機能変更の頻度の予想 ［多いー少ない］ ・システムの寿命 ［長いー短い］ ・データの結合度 ［高いー低い］ ・変更による信頼性の変化 ［ないーある］
想定する影響箇所に関与するステークホルダ	・変化対応時の利用部門のレスポンスの速さ ［速いー遅い］ ・変化対応時の利用部門の協調の程度 ［高いー低い］ ・変化対応時の管理部門のレスポンスの速さ ［速いー遅い］ ・変化対応時の管理部門の協調の程度 ［高いー低い］ ・変化対応時のベンダ管理の程度 ［容易ー困難］
想定する影響箇所を運用した場合	・利用部門への技術サポートの程度 ［多いー少ない］ ・利用部門への業務サポートの程度 ［多いー少ない］ ・ベンダサービスの使用頻度 ［多いー少ないー無い］

対応の方向性として不確実性の 4 レベルのうち、戦略姿勢「適応」に対応するレベル 1～3 を用いる(表 4-7)。

表 4-7 不確実性レベル(再掲)

レベル	名称	内容
レベル 1	確実に見通せる未来 (A Clear-Enough Future)	ある程度の正確さで未来を予測できるレベル。
レベル 2	他の可能性もある未来 (Alternate Future)	異なる複数のシナリオとして未来を描き出すことができるレベル。 方向性が見えるレベル。
レベル 3	可能性の範囲が見えている未来 (A Range of Future)	起こりうる未来を一定範囲内で特定できるレベル。

不確実性レベルに対応した例を表 4-8 に挙げる。

表 4-8 変化対応の方向性

不確実性レベル	例
レベル 1 (確実に見通せる未来に対応する)	・変化が予測できるものを予めシステム上に実装しておき、必要な時期が来たらすぐに使えるようにする。 ・変化の頻度やペースが異なることが予測できるユーザインタフェース、アプリケーションロジック、データモデルを分けた 3 層アーキテクチャを採用

	する。 ・技術的な進化により、使用するデバイスが頻繁に変更することを予測して、デバイス対応モジュールを切り出して管理する。 ・ほぼ定期的に起きる法令改正に対応するために、スクラッチ開発ではなく法令対応パッチが提供されるパッケージソフトウェアを導入する。
レベル 2 （他の可能性もある未来に対応する）	・想定外の事象に対応するため、全てシステムで対応するのではなく、運用で対応できる余地、または人が関与できる余地を残しておく。 ・業務パターンを事前に定義し用意しておくことで、新たな業務に対する適応性を高める。 ・適応性が求められる箇所を内製化し、環境変化に対して迅速に行うことで適応性を高める。 ・設定値を変えるだけでアプリケーションの機能を変更できるようにする。 ・将来リリースする製品バリエーションに円滑に対応するために、ソフトウェアプロダクトラインを採用する。 ・開発環境、テスト環境、運用環境をスムーズに移行できるようにする。
レベル 3 （可能性の範囲が見えている未来に対応する）	・環境の変化を予測し、拡張が容易になるようなアーキテクチャを構築する。 ・汎用品またはデファクトスタンダードの部品や製品を採用することで、拡張時の調達が容易になり、拡張性が高まる。 ・自社で機材を調達するのではなくクラウドサービスを活用することで、IT リソースの利用を容易にし、環境の変化に応じた拡張を行う。 ・商品バリエーションの拡大に対応させるために、商品コードなどのコード体系を拡張可能なものに変更する。

4.5. 環境変化適応の事例

4.5.1. 情報システムに求められる柔軟性

情報システムに求められる重要な要件の一つに、ビジネス環境の変化に合わせた機能変更の「柔軟性」がある。変更への柔軟性が必要かどうかは、設計後の変更の多さにも大きく関係している。稼働以降、全く手を入れることなくリプレースを迎えるシステムなどほとんどなく、稼働前には既に次のリリースが計画されているケースなどもある。

また、変更にかかるコストも当初構築時にかかった費用と同程度の費用がその後の追加開発の費用としてかかっているケースもある。実際、恒常的に機能を変更して使い続けている現実があるので、変更の手間を軽減する価値は非常に大きい。システムを柔軟に変更できなければ、システムが事業の足を引っ張る結果となってしまうが、近年このようにシステムが事業を進めていく上で足を引っ張る原因となってしまい、システムが悪者になってしまうというケースが起きている。

4.5.1.1. これまでの対応

情報システムを構築する際に「柔軟性確保」という表現で「処理対象範囲、処理手法の変更に、ソフトウェアを中心とした大規模なシステム改造を行うことなく、対応可能な構築を行う」という要件は必ず入れている。では実際に十年以上前ではこの柔軟性要件に対する実装をどのようにしてきたかというと、事前にテーブルを用意し、パラメータを変更するだけで新たなサービスが提供できる仕組みを開発当初から組み込むということを行った。

また、当時、プログラミングの手法として保守性を重視したプログラム構造を目指したと言われて

いる「構造化プログラミング」が主流となっていた。保守性を高めるということは、システム変更に対して柔軟性を提供するということでもあり、この手法も取り入れシステムの柔軟性を確保したシステム構築を行った。

しかし、その効果はどうであったかというと、事前にテーブルを用意することで対応できる機能は、開発当初に実装されている既存機能の組み合わせの中でのバリエーションであり、あくまで既存機能の範囲内、既存機能の延長線上でしかないという限界があった。このために実際にこの仕組みを使い、新たな機能を提供することができたケースは、実際に用意したテーブル数の数十分の一、またはそれ以下という状況であった。ほとんどの場合、事前に用意していた仕組みが、その後発現したビジネス環境の変化には対応できずに何らかのシステム改造を伴わなければ対応できないケースであった。

また、「構造化プログラミング」は業務処理ごとの独立性が保たれている構造になっていることから、修正を行おうとした場合に修正箇所が限定的となるなど、業務処理の一部を変更する場合には非常に有効な手法として機能している。しかし、反面、業務処理ごとに処理を配置する手法をとり、同じような処理が各業務に点在している構造となるため、組織変更などの業務を超えた変更を行おうとした場合には、修正箇所の調査に多くの時間がかかってしまうという状況であった。

4.5.1.2. 現状の対応

近年では多くの IT ツールベンダが提唱している SOA という手法が拡大しつつある。予め様々な機能をサービスとして用意し、それらを利用して情報システムを組み立てるという SOA は、柔軟性の高いシステムを実現するために非常に優れた仕組みであると言われている。しかし、SOA によって実現できる柔軟性とは、予め整理したサービスの単位での機能変更であれば柔軟に対応できるといえるが、設計の初期段階から変更となるサービス単位を考慮して設計しないと柔軟性を高められない。やはり変更への柔軟性を高めるには、開発当初からどこが変更となるのか意識して設計する必要がある。前述した予め変更を予想しテーブル用意する方法にも似ている部分でもある。

また、SOA では複数の機能の中から共通のサービスを統合する構成をとることになるので複数業務を処理する共通機能の修正を行う場合などは、その共通処理を利用している全ての業務機能について影響調査やその後のテストが必要になり、結果として開発工数が膨らんでしまう。

SOA によって効果が期待できるのは、企業の合併や部門横断的な BPR（業務改革）などと言われており、業務レベルでの機能変更にはあまり有効でないとも言われている。一方で、集計機能の追加や表示方法の変更など、情報システムを日々利用している担当者の要望を柔軟に反映していくための手法としては、例えば、「オブジェクト指向」による機能の再利用や、Ruby などのオブジェクト指向スクリプト言語を用いた Agile 型開発や BI ツールの利用といった手法の方が直接的に有効なケースも存在する。

4.5.1.3. これからの対応

これまで説明した手法を含め、処理対象範囲、処理手法の変更に、ソフトウェアを中心とした大規模なシステム改造を行うことなく、対応可能な構築を行うという「柔軟性の確保」のためのアプローチについては、様々な方法論が出ているが、全てに効果のある万能な方法はない。実際には、情報システムを企画する時に将来の変化について適切な予測を行い、そこで求められる柔軟性

の内容を分析するとともに具体化し、それに合わせた手法の選択を行えばよいということになるが、変化のスピードが速くなっている現在、予測そのものが非常に困難になっている。

一方、予測できない変化への対応としては、システム改造を行う際に多くの工数を割くことになる修正箇所の調査と回帰テストに対して対策を打つことである。

ある業務を変更する場合に、システムのどこを変更すればよいのかを調査するのに多くの工数がかかる。この対応としては、要件定義書～基本設計書～詳細設計書～プログラム設計書までの間をトレースできる仕組みを用意しておくことで、業務の変更に対して容易に修正箇所が発見でき、かつドキュメントのどこを修正したらよいかも容易に把握することができ、調査工数を削減するのには有効な手段となる。また、この仕組みは新規にシステムを構築すると同時に行うことになるので、その際には、要件を基本設計以降の工程で要件をトレースすることと同じ活動になり、上流工程で要件の設計への展開漏れ、誤りを発見し、改修することができ次工程に持ち越される不備を減少させることが可能となり、開発品質の向上や、特に製造工程以降に発生する手戻りを大幅に削減する効果もある。

また、プログラムを変更する際に、それによって新たな不具合が起きていないかを検証する回帰テストへの対応としては、初期開発時に使用したテストプログラムとデータをセットにしてシステム運用に入ってからも常に使用可能な状態にメンテナンスすることである。プログラム変更時に改めて準備することなく利用できるようにすることは、システム改造時の工数削減には有効な手段となる。

回帰テストを行う際には、通常、全てのテストケースを実施するわけではなく、修正したプログラムコードの影響範囲を考えて行うべきテストケースを絞り込む作業を行うことになるが、前述の要件定義書～基本設計書～詳細設計書～プログラム設計書までの間をトレースできる仕組みが用意されていれば、影響範囲を特定しテストケースを絞り込む作業を効率的に行うことができる。

予測できない変化への対応については、設計書のトレース状態を維持する等、初期開発時はもとより、システム運用に移って以降についても、相応のメンテナンスのためのコストが必要になる。概してこのようなコストは削減の対象となりがちなので、企画段階で「柔軟性」を確保するためには、このようなコストが必要であるということについて、社内でコンセンサスを得ていくことが重要になる。

4.5.2. 要求進化（Requirements Evolution）

ソフトウェア進化（Software evolution）とは、一旦出荷されたソフトウェアに対する変更を受け入れる仕組みや活動と言われている[1]。新規に開発されて出荷後にユーザに提供され利用された後に、ソフトウェアのユーザを含むステークホルダの要求の変化や、ステークホルダを取り巻く環境の変化により、ソフトウェアは変更される必要がある。従来、このための活動はソフトウェア保守（Software maintenance）の一部と位置付けられてきた。

1980 年代になるとLehman[2]に代表される研究者達は、このような活動を、ステークホルダからの要求をより積極的に受け入れることでソフトウェアに変化を起こすこと、即ち「ソフトウェア進化」と捉えるようになり、ソフトウェア保守と区別されるようになった。

ソフトウェア進化の研究は現在では様々な拡がりを見せている。ソフトウェア進化の一つの構成

概念として、要求工学の領域では、要求進化（Requirements Evolution）の研究分野が挙げられる。要求進化の研究課題としては、ステークホルダからの変更要求を時系列に捉えて、要求変更時の影響分析や要求トレーサビリティの管理等が挙げられる[3]。

4.5.2.1. 進化的変化タイプ

要件定義を進める過程で、ステークホルダの新たな要求（追加要求・変更要求）のリクエストに伴い、要件定義の成果物は分割・統合・置換等の様々な形態に変化をする。Janeらの研究[4]では、上述の要求進化の変化の形態について、要件定義の成果物の連続的な変化に着目して、以下に示すような七つのタイプ（Evolutionary Change Type：進化的変化タイプ）を定義している。

図 4-5 要求進化のタイプ[3]

これらの七つの要求進化の内容は以下のようになる。Create（追加）では、新しい要求を作成する。Inactive（削除）では、既に存在する要求を削除する。Modify（修正）では、要求の属性（値や記載）の内容を変更する。例えば、要求の削除に関する属性を真（true）に変更するという修正により、当該要求が削除されたとみなす、というケースも存在する。そのため、 Inactive は Modify の特殊なケースであるともみなせる。Merge（統合）では、二つ以上の要求を一つに統合する。拡充（Refine）は、オリジナルな要求に新しい要求を追加する。分解（Decompose）では、一つの要求を二つ以上の要求に分解する。Merge と Decompose は非常に似ている変化（要求進化）である。Merge がオリジナルな要求が引き続き存在する一方で、Decompose はオリジナルな要求はこの変化後は存在しないことが両者の違いである。Replace（置換）では、既に存在していた要求が新しい要求に置き換わる。なお、これらの要求進化は排他的なものではない。

4.5.2.2. 要求進化の経緯と根拠

要求進化の経緯（History of evolution）は、最新の要件定義の成果物セットを調査しても把握することは難しい。例えば、前述の七つのタイプの Decompose や Replace の要求進化によって新しい成果物が作成されたケースを考える。ここで要求進化のタイプが Decompose や Replace なので、その元となる成果物（オリジナル成果物）は存在しない。もし、ステークホルダがオリジナル成果物の内容に関連した要求変更をリクエストしてきた場合、本来であればオリジナル成果物に基づき作成された成果物を調査する必要がある。しかしながら、開発者がこの要求進化の経緯を知

らないと、対象となる成果物が見つけられない、もしくは工数を要するリスクがある。

また、要求進化の経緯と併せて、なぜそのような変更をしたのかという意思決定（要求進化の実施）の根拠（Rationale）も重要なノウハウというべき情報である。こういった進化の根拠を記録しておくことで、将来のソフトウェア開発に活用することは、設計根拠（Design Rationale）の研究分野では古くより言及されている[5]。

このように、要求進化の過程を体系的に扱うためには、進化の経緯（History of evolution）と進化の根拠（Rationale of evolution）の二つを正しく把握することが重要となる。特に新規開発よりも保守開発が増えていく中、アーキテクチャやプログラムと同様に、要求も進化する。これらの進化を前提として、要求を管理していくことが重要となってくる。しかしながら、要求進化の情報は従来開発者の頭の中だけに存在し記録はされていないのが現状である。開発者の入れ替えで失われると、要求進化の情報の再現や再利用が困難になってしまう。

4.5.2.3. 研究動向

IPAの要求工学の動向調査[6]では、“要求の進化”は「要求管理（Requirements management）」の研究分野に位置付けられている。この動向調査では、要求管理の研究分野はその重要性は以前より認識されているが、研究者・実務者とも取組みは不十分であると報告されている。取組みが不十分な大きな理由として、この研究分野は実務者との共同研究や、実務上の協業が必須であるため、要求工学の研究者がなかなか踏み込めない領域であるとされている。

今後、要求進化、及びその履歴や根拠を記録し、再現・再利用するための理論的・実証的な研究が実施されることを期待したい。

---

[1] 大森隆行, 丸山勝久, 林晋平, 沢田篤史, ソフトウェア進化研究の分類と動向, コンピュータソフトウェア, Vol.29, No.3, Aug.2012, pp.3-28.

[2] Lehman, M.M. “Programs, Life Cycles, and Laws of Software Evolution”, Proceedings of the IEEE, vol.68, No.9, 1980.

[3] N. A. Ernst, J. Mylopoulos, Y. Yu, and T. Nguyen, Supporting requirements model evolution throughout the system life-cycle, Proceeding of 16th IEEE International Requirements Engineering Conference, 2008, pp.321-322.

[4] J. Cleland-Huang, C. K. Chang, and M. Christensen,“Event-Based Traceability for Managing Evolutionary Change”, IEEE Transactions on Software Engineering, vol.29, No.9, 2003, pp.796-810.

[5] C. Potts and G. Bruns, Recording the Reasons for Design Decisions, Proceeding of 10th international Conference of Software Engineering, pp.418-427, IEEE, 1988.

[6] 鎌田真由美, 要求工学:1.要求工学の現状と課題, 情報処理, vol.49, No.4, 2008, pp.347-356

4.5.3. 要求変化とアーキテクチャ

非機能要求（以下、NFRという）とアーキテクチャ分析WG報告書[1]では、事業体としての企業や公的機関(以下、エンタプライズという)が価値創造や組織活動の支援に利用するITシステムを対象として、WG独自のメタモデル（以下、「NFRとアーキテクチャ分析のメタモデル」という）を作成して、NFRとアーキテクチャの因果関係を整理した。本節では、上記メタモデルの対象をITシステムからSocio-Technical Systemsに広げて要求の変化とアーキテクチャの関係を考察する。

4.5.3.1. IT システムと Socio-Technical Systems

本WG 2011 年度 活動報告書[2]のステークホルダ整理の一つとして、企業におけるステークホルダと要求、ゴールの関係を示すKikuma Ringを紹介した(図 4-6)。Kikuma Ringでは、企業のITシステムは、その企業のビジネスゴールの達成、ビジネスリスクを軽減する目的で開発し、ITシステムのユーザの要求だけではなく、エンタプライズ・レベルの要求やそのエンタプライズの顧客やパートナー企業からの要求、法令やCSR(企業の社会的な責任)に関連する要求を意識する必要があることを説明している。

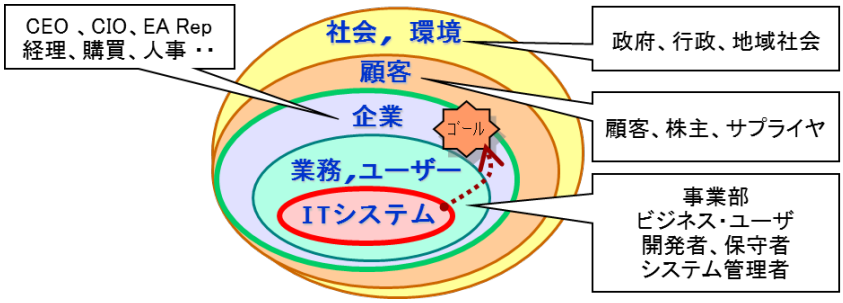


図 4-6 企業におけるステークホルダと要求、ゴールの関係[2](一部加筆)

一方、Socio-Technical Systems(以下、STSという)[3]は、複数の企業の他、コミュニティなどが協調して、社会的なゴールを満たす(あるいは社会的なリスクを低減する)ことが目的である(図 4-7 参考[4])。

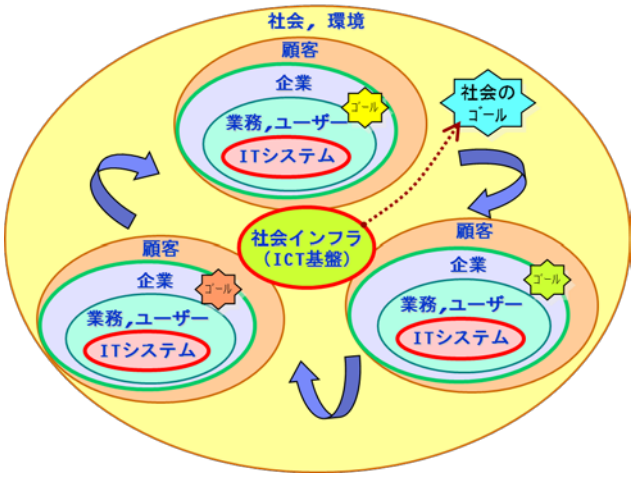


図 4-7 Socio-Technical Systems でのステークホルダと要求、ゴールの関係[4]

関連する企業、コミュニティなどが社会における法令、コンプライアンスや各企業内の規則やビジネスルールを遵守しつつ、IT システムだけでなく、ビジネスプロセスや組織、人も含んだビジネスシステム全体で社会的な目的を果たすというものである。各々のエンタプライズの IT システムは、インターネットやブロードバンド、あるいは携帯電話網といった ICT 基盤(社会インフラ)を介して、協調することで社会的なゴールの達成に寄与している。

このSTSは、まさにSystem of Systems(以下、SoSという)の構造である[5]。SoSは、表 4-9 に示した四つのタイプに分類されることが知られている[6]。Virtual SoSは、SoSを構成するシステムが、別々の目的の達成のために存在していたところに、情報通信技術(ICT)により、システムが接続さ

れることで新たな目的を仮想的に実現している。Collaborate SoSでは、各システムのユーザやシステム・オーナーが相互依存性を理解し、システムの代表者が、相互利益のために自発的に協力している。Acknowledged SoSでは、承認された目的、指定管理者やリソースが存在する。また、構成要素（システム）は、独立したオーナーシップ、目的、資金調達、それに、開発・維持のアプローチを保持している。システム変更は、SoSとシステムの間での協調に基づいている。Directed SoSは、特定の目的の遂行のために構築、運営されているSoSである。そのSoSの構成要素（システム）は、独立して稼働する能力があるが、通常時の稼働モードでは、中央の管理された目的に従って稼働する。

表 4-9 SoS の種類[6]

種類	解説（「システムズスクール」講義資料[7]）
Virtual	・仮想的 ・目的の認識がなく、創発的に目的が達成
Collaborate	・協調的 ・共通の目的のために自発的に協力
Acknowledged	・承認された ・それぞれが目的を持つが、方針は共通
Directed	・中央集権的 ・各要素の独立性はあるが、一つの目的

4.5.3.2. 「NFR とアーキテクチャ分析のメタモデル」の拡張

「NFRとアーキテクチャ分析のメタモデル」は、BMM（Business Motivation Model ）[8]、「非機能要求記述ガイド」[9]で記述したコントロールケース[10]、文献[11]で唱えているミッションからアーキテクチャの関連性、ITアーキテクチャ・メタモデルセマンティクス解説[12]などを参考に整理されたものである。今回のテーマである「要求の変化」を検討する上では、組織（Organization Unit）やビジネスプロセス（Business Process）、アクター（Actor）の 3 要素も「要求変化に影響を及ぼすもの」と考えられるが、上記のメタモデル中では、対象として取り込まれていなかった。そこで、BMMを参照し、「NFRとアーキテクチャ分析のメタモデル」に追加し、拡張メタモデル（図 4-8）とした。これにより、Socio-Technical Systemsに含まれ、フォーカスすべき、Mission、Visionやビジネス目標、人、プロセス、ビジネスルールなどのビジネス制約、ITシステムなどの関係が整理できた。

表 4-9 で示した SoS の種類を拡張メタモデルと照らし合わせて、整理する。

Virtual SoSの例として、KOMTRAX[13]（コムトラックス：Komatsu Machine Tracking System：機械 稼働管理システム）がある。製品（建設機械）に通信アンテナを搭載し、GPSや通信衛星、携帯電話、インターネットを組み合わせることで、製品の盗難防止や異常管理、需要予測などを可能にしている。このSoSを構成しているGPSや通信衛星、携帯電話、インターネットは、建設機械とは、全く別次元のビジネスシステムとして存在していて、ビジネスレベル、ITレベル共、対象範囲となっていない。

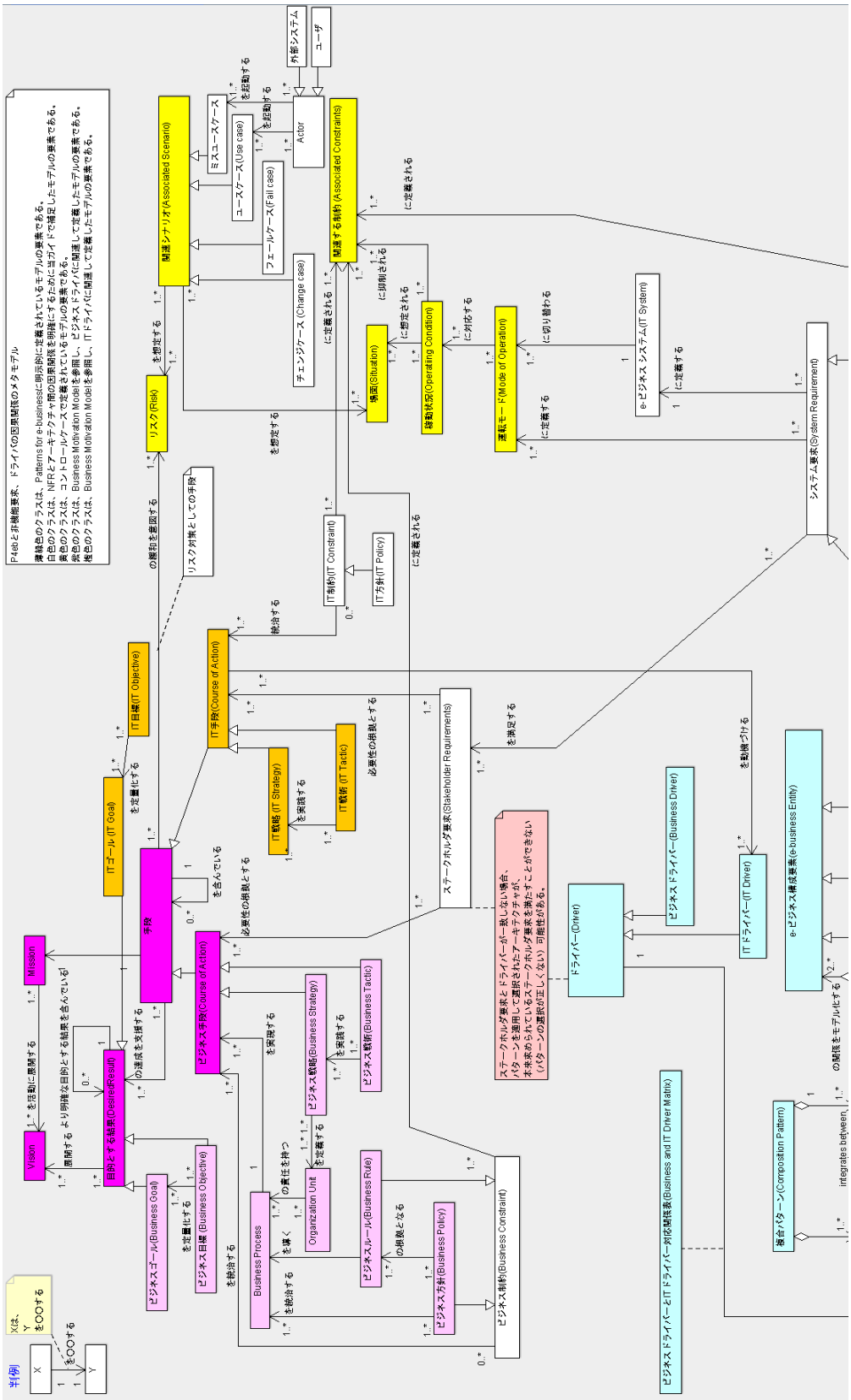


図 4-8 拡張したメタモデル(拡張部分を抜粋)

Collaborate SoSとして、OSLC（Open Services for Lifecycle Collaboration）[14]の試みがある。OSLCとは、開発ツールの相互運用性を向上するという共通の目標を持つ企業・組織・個人からなるオープンコミュニティである[15]。異なるツール間であってもデータ連係を可能にすることで製品やシステムの開発品質、及び効率（コスト）の向上、リードタイムの短縮を図るという共通目的のために、参加ベンダが協力している。拡張メタモデルで考えると、目的とする結果（Desired Result）は、共通であり、ビジネス手段の一部として、他システムが定められた仕様に準拠し、協調する。他システムのビジネス手段からIT手段への関連やITシステムへの関連は、各ベンダ独自の考えに基づくものとなり、ブラックボックスといえる。SuD（System Under Discussion）のITシステムとの関連を考えると、この他システムは、外部システム（アクター）として取り扱われる。

Acknowledged SoS の例としては、サプライ・チェーン・マネジメント（SCM）やビジネスプロセス・アウトソーシング（BPO）がある。これらは、各システム（サプライ・チェーンを構成する企業や BPO 先の企業）が独自の目的や資金調達、開発・維持のアプローチを持っているが、契約で承諾した SoS の目的達成のために協調する。したがって、契約範囲外の各システムの目的やそれに紐付く手段やビジネスプロセスなどは、対象範囲外となる。

Directed SoS としては、一企業内あるいは、企業グループの SoS が挙げられる。この SoS では、個々のシステムは、個別に稼働するが、その目的は、その企業の目的達成のためであり、CIOにより中央集権的に管理される。つまり、ビジネスレベルでも IT レベルでも統一して扱われるので、SuD の IT システムも SuD 以外の IT システムもサブシステムとして取り扱うことができるといえる。

以上のように、いずれの SoS の種類も、拡張メタモデルの対象となっている要素（Element）で表現できる（表 4-10）。即ち、拡張メタモデルは、単一のエンタプライズのシステムを対象としたビジネス要求、IT 要求からアーキテクチャの関係を扱えるだけではなく、SoS も扱うことができる。

表 4-10 SoS のタイプの拡張メタモデル上での取り扱い

System of Systemsの種類		メタモデル上での他システム（SoSを構成するが、SuDでないシステム）の取り扱い（当WGでの検討）		
Type	解説	ビジネスレベル	ITシステムレベル	説明
Virtual	・仮想的 ・目的の認識がなく、創発的に目的が達成	×	×	ビジネス系の要素（Element）も含め、全く別のシステム（メタモデル上の関係がない）
Collaborate	・協調的 ・共通の目的のために自発的に協力	△	△	ビジネス系の要素（Element）のうち目的とする結果（Disired Result）は、共通だが、ビジネス手段の一部として他システムが協調。 他システムにおけるビジネス手段からIT手段、ITシステムへの関連は、ブラックボックス。 他システム内のITシステムは、SuDのITシステムの外部システムとして扱われる。
Acknowledged	・承認された ・SoS全体で承認された目的、指定された管理者やリソースが存在 ・それぞれが目的を持つが、方針は共通	○	○	ビジネス系の要素（Element）は、同一だが、IT手段（Course of Action）としてのステークホルダ要求、システム要求は、別 （ITシステムの要求構造が同列で並ぶ）。 他システム独自の目的（契約対象範囲外の目的）に紐付くものは、対象外である

Directed	・中央集権的 ・各要素の独立性はあるが、 一つの目的	◎	◎	ビジネス系の要素(Element)は、同一で、IT手段も統一して扱う。 SuDのITシステムもSuD以外のITシステムもサブシステムとして扱える。

SuD: System under Discussion

ビジネスレベルの取り扱い
◎:SuD も他システムも統一して扱える
○:ビジネスシステムとして統一して扱うが、一部対象範囲外
△:SuD のビジネス手段の一部が統一
×:対象範囲となっていない

IT システムレベル
◎:サブシステムとして扱う
○:同列の他 IT システムとして扱う
△:緩い協調をする他システム(外部システム)として扱う
×:対象範囲となっていない

4.5.3.3. 拡張メタモデルでの要求の変化の検討

上位のビジネス要求(ビジネス目的、目標や手段)や IT 手段、IT 制約、あるいは、リスクの緩和策に対して、NFR やアーキテクチャが関連している。メタモデル要素に影響を及ぼす要求の変化(要求の変化要因)を Kikuma Ring の層と紐付け、表 4-11 にまとめた。例えば、Mission(メタモデル要素# B2)は、会社の合併、統合によって、影響を受ける(変化する可能性がある)。社会、環境(政府、行政、地域社会)からの法規制・ガイドラインの変化や顧客・株主・サプライヤとの契約・商慣習の変化がビジネス・ルール(メタモデル要素# B8.1)の変化に繋がる。また、そのビジネスルールの変化は、ビジネスプロセス(メタモデル要素# B6)の変化の要因となり得る。IT システムの関連シナリオ(ユースケース、チェンジケース、フェールケース、ミスユースケース)の変化は、リスク(メタモデル要素# IT5)に影響を及ぼすことが分かる。

要求抽出の段階で、これらの要求の変化要因を予め、整理・検討し、どの要求の変化に対応するか決めておくことで、要求変化に対応可能なシステムが構築できる。特に、フェールケース(メタモデル要素# IT9.2)の変化要因(災害・事故・障害)や場面(Situation)(メタモデル要素# IT6)、稼働条件(Operating Condition)(メタモデル要素# IT7)の変化要因を考慮することで、Resilience な IT システムの構築が進められる。

また、非機能要求とアーキテクチャ分析 WG 報告書[1]では、アーキテクチャ決定に関する正当性根拠(Justification)と理論的根拠(Rationale)の重要性を説いたが、理論的根拠(Rationale)として、予測している要求の変化に対してもアーキテクチャの仕掛け(Mechanism) とその原理(Principle)を明らかにする必要がある。その他にアーキテクチャ検討の考慮事項として、変化を予測している要求が期待値の範囲内か否かのモニタリングする仕掛けや、期待値を超えた場合の IT システムの挙動シナリオも明確にする必要がある。その挙動シナリオの実現手段を IT システムに取り込むのか、運用作業で実現するのかも判断し、文書化や適切な対処をすべきである。

表 4-11 メタモデル要素に影響を及ぼす要求の変化

			メタモデル要素に影響を及ぼす要求の変化					
			Kikuma Ring の層(Ring の主なステークホルダ)					
分類	要素#	影響を受ける メタモデル要素(Element)	社会、環境 (政府、行政、地域社会)	顧客・ 株主・サプライヤ	企業 (CxO・経理・ 購買・人事)	事業部 ビジネス・ユーザ システム部門 システム開発者 システム保守者 システム管理者	IT システムベンダ (IT システム)	ソフトウェアベンダ (ソフトウェア)
ビジネス関連	B1	Vision	ビジネス環境の変化					
	B2	Mission			企業の合併、統合			
	B3	ビジネスゴール	ビジネス環境の変化					
	B4	ビジネス目標	ビジネス環境の変化					

	ビジネス手段							
	B5.1	ビジネス戦略	ビジネス環境の変化					
	B5.2	ビジネス戦術	ビジネス環境の変化					
	B6	ビジネスプロセス	ビジネスルールの変化	ビジネス方針の変化 ビジネスルールの変化				
	B7	組織			ビジネス戦略の変化			
	ビジネス制約							
	B8.1	ビジネスルール	法規制、ガイドライン	契約、商慣習	社内規定、社内標準	ビジネスルール		
	B8.2	ビジネス方針			ビジネス手段の変化 要員の変化			
IT関連	IT1	IT ゴール			ビジネス手段の変化 IT 環境の変化 リスクの変化			
	IT2	IT 目標			ビジネス手段の変化 IT 環境の変化 リスクの変化			
	IT 手段							
	IT3.1	IT 戦略			IT 環境の変化 IT 制約の変化			
	IT3.2	IT 戦術			IT 環境の変化 IT 制約の変化			
	IT4	IT 制約				既存 IT 環境の変化	テクノロジの変化 ハードウェア性能の向上 既存 IT 環境の変化	テクノロジの変化 新機能追加による前提 ハードウェア・リソース要求の変化
			IT 国際標準や IT 業界標準の変化		社内 IT 標準、 IT 規約	システム開発者のスキル	物理的制約（フロアスペースや電力など）の変化	
	IT4.1	IT 方針			IT 環境の変化			
	IT5	リスク			リスク評価方法		関連シナリオ	
					手段（ビジネス/IT）の変化			
	IT6	場面（Situation）			IT システムのライフサイクルプロセスの 様々な局面で起きるビジネスの変化 （新規出店、事業統廃合、海外展開など）			
	IT7	稼働条件(Operating Condition)				業務処理要求件数の増加		
						業務処理要求の時間的な集中		
	IT8	運転モード(Mode Of Operation)					想定した場面と稼働状況に対応した運転モードの追加	
	関連シナリオ							
	IT9.1	ユースケース					IT システムの使い方 提供機能の変化	
	IT9.2	フェールケース	災害・事故				劣化 故障の発生	バグ
	IT9.3	ミスユースケース	想定外の使い方 ハッキング手法の変化					
	アクター							
	IT10.1	ユーザ	ユーザの規模、範囲の変化 スキルレベルの変化 利用 IT 環境、テクノロジの変化					
	IT10.2	外部システム	入力値・出力値の変化 非機能要求の変化 テクノロジの変化					

4.5.3.4. まとめ

要求変化に対応可能なシステム（Resilienceを考慮したシステム）を構築するためのヒントとして、STSを対象とした「NFRとアーキテクチャ分析のメタモデル」の拡張メタモデルを紹介した。そのメタモデル要素と要求の変化を引き起こす要因（要求）との関係をKikuma Ringの各層で整理し、要求変化とアーキテクチャの関連を示した。要求変化に対する要求の記述方法については、[2] [16]で紹介しているコントロールケースが利用可能と考えている。コントロールケースのスキーマ中のchangeCaseのfromAssociateSituationとtoAssociateSituationを利用し、場面や対象範囲とともに、想定される要求の変化が及ぼすリスクを記述することで、要求変化への対応要求が記述できると

考えられるが、今後、さらに検討すべき課題である。

今後、益々、IT システムの SoS 化が進み、些細な要求の変化や直接接続していない IT システムの停止や挙動の変化で、システム全体に影響が出る可能性もある。「要求の変化・発展」を適切に捉え、記述する方法、要求変化のモニタリングや対応アーキテクチャなどの研究が進むことを期待する。

---

[1] 非機能要求とアーキテクチャ分析 WG 報告書, IPA/SEC 非機能要求とアーキテクチャ分析 WG, 2011.4.27

[2] 要求発展型開発 WG 2011 年度 活動報告書, IPA/SEC 要求発展型開発 WG, 2012.4.26

[3] Large-Scale Complex IT Systems, Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid, and Richard Paige, Communications of the ACM, JULY 2012, vol. 55, no. 7, pp.71-77

[4] 6 次産業化を推進する ICT 基盤の開発―水産業や農業の 6 次産業で地域経済活性化をサポートするクラウド・アーキテクチャー, 末次 信治, 山本 久好, 2011 年 10 月 IBM PROVISION 71 号「特集テーマ:Smarter Cities―スマートな都市の実現」 http://www-06.ibm.com/ibm/jp/provision/no71/pdf/71_article3.pdf

[5] INCOSE Systems Engineering Handbook Version 3.2, SE Handbook Working Group, International Council on Systems Engineering (INCOSE), January 2010

[6] Systems Engineering Guide for Systems of Systems Version 1.0, Director, Systems and Software Engineering Deputy Under Secretary of Defense (Acquisition and Technology) , Office of the Under Secretary of Defense (Acquisition, Technology and Logistics) , August 2008

[7] 慶應義塾大学システムデザイン・マネジメント研究科, 「システムズスクール」システムエンジニアリング・ワークショップ 講義資料, 2013 年 2 月

[8] The Business Motivation Model, The Business Rules Group, 2007

[9] 非機能要求記述ガイド, 経済産業省 ソフトウェア開発力強化推進タスクフォース 要求工学・設計開発技術研究部会 非機能要求とアーキテクチャ WG, 2008 年 7 月

[10] Joe Zou and Christopher J. Pavlovski: “Modeling Architectural Non Functional Requirements: From Use Case to Control Case”, Proceedings of IEEE International Conference on e-Business Engineering 2006(ICEBE’06) (2006).

[11] IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE, 2000

[12] IT アーキテクチャ・メタモデルセマンティクス解説, ITスキル標準 プロフェッショナル・コミュニティ ITアーキテクト委員会, 2007.6.28

[13] 産業構造審議会産業競争力部会(第 4 回)-配付資料 4(2) 主要産業・戦略分野(エレクトロニクス・IT関係)～情報通信コストの劇的低減を前提とした複合新産業の創出と社会システム構造の改革～, 経済産業省, 2010 年 4 月 23 日, p.12

[14] Open Services for Lifcycle Collaboration, http://open-services.net/

[15] developerWorks OSLC Japan Wiki, https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/OSLC%20Japan%20Wiki/page/1.OSLC%20%E3%81%A8%E3%81%AF?lang=en

[16] 要求工学・設計開発技術研究部会 非機能要求とアーキテクチャ WG 2006 年度活動報告書, 経済産業省 ソフトウェア開発力強化推進タスクフォース 要求工学・設計開発技術研究部会 非機能要求とアーキテクチャ WG, 2007 年 8 月

4.5.4. アジャイル開発

アジャイル開発(Agile development)とは、反復漸進及び顧客参加のソフトウェア開発により高い顧客満足度を得るソフトウェア開発手法の総称である。代表的なものとして、Scrum[1]や

eXtreme Programming (XP)[2]がある。以降においては、アジャイル開発の基本的な考え方とプラクティスを説明した上で、その特徴に基づく要求変化・発展との親和性及び不確実性との関係を説明する。

4.5.4.1. 基本的な考え方

アジャイル開発は、計画に基づく開発(Plan-based development)に対する言わばアンチテーゼであり、根底にある考え方(マインドセット)は以下のアジャイル宣言[3]にまとめられている。

- プロセスやツールよりも、人間と人間関係を重視する
- ドキュメントよりも、動くソフトウェアを重視する
- 契約交渉よりも、顧客との協力を重視する
- 計画に従うよりも、変化に対応することを重視する

名称の候補として他に"ビジネス・アラインド(Business Aligned)"が挙げられていた[4][5]ことからも分かるように、顧客と対立するのではなく手を携えて、顧客のビジネスに寄り添い一体化されて変化に付いていき価値をともに高める考え方が根底にある。

4.5.4.2. プラクティス

具体化されたアジャイル開発手法である Scrum や XP においては、これらの考え方に基づいて日々実践することが推奨されるプラクティスがまとめられている。例えば、Scrum において「30 日イテレーション」や「自律的組織化チーム」など、XP において「1 週間サイクル」や「全員同席」「チーム全体」といったものである。

ただしプラクティスはあくまでも一般に推奨される事柄であり、常に有効であるとは限らず、組織やプロジェクトが抱える問題や状況に応じた取捨選択や適切なパラメータの設定(例えばイテレーション期間[6])が重要である。

4.5.4.3. 要求変化の扱い

アジャイル開発手法によって重きを置く点やプラクティスの違いはあるものの、一度に全ての範囲を扱うのではなく、反復漸進的に進めることで顧客のフィードバックを多く得て着実にかつ変化に対応可能とすること、及び、顧客の積極的な参加を得て顧客にとって当該時点で最も価値のある事柄に取り組む点は、共通している。

種々のアジャイル開発手法に概ね共通する開発の流れを図 4-9 に示す。アジャイル開発では、小刻みな反復(イテレーション)を繰り返す中で頻繁に顧客のフィードバックを得て、必要な決定を、その決定に必要な情報が揃ったタイミングで実施する。これは経済学のリアルオプション分析の考え方に照らすと、リスクに基づいて戦略的に決定を遅らせることで全体としての価値を高めるものである[7][8]。

計画に基づく開発は、要求の変化や発展を「やっかいなもの」と捉えて、要求や環境が最初の計画時に全て見渡せてそのとおりに(計画どおりに)開発することを目指す。これに対して、アジャイル開発はその根底において要求の変化や発展をむしろ「好機」と捉えて味方に付けようとする。

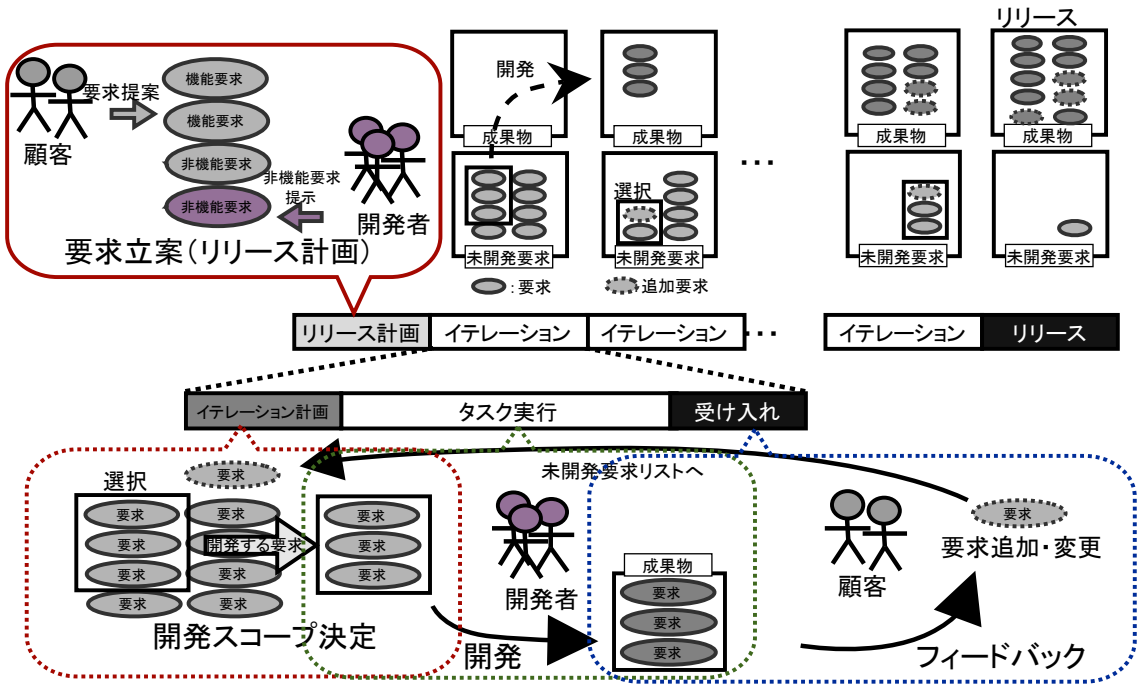


図 4-9 アジャイル開発に概ね共通する流れ([6]より)

4.5.4.4. 不確実性との関係

アジャイル開発は上述の考え方やプラクティスに基づき、ビジネスや環境が完全に予測できる場合よりもむしろ、複数の可能性があり方向性が見える状況(ヒュー・コートニーの不確実性レベル 2)におけるビジネスや環境のそれぞれの可能性に適応した開発や、可能性の範囲が見えている状況(レベル 3)における要求の探索的な開発に適している。

ただし、アジャイル開発の導入が直ちに、不確実性の克服を保証するものではない。IPA/SECの過去の調査[9]は、アジャイル開発を代表とする非ウォーターフォール型開発を対象に、不確実性の属性として市場の不確実性や技術の不確実性、依存関係・スコープの柔軟性などを挙げた上、その程度に応じた留意すべき点をまとめている。

---

[1] Ken Schwaber, アジャイルソフトウェア開発スクラム, ピアソンエデュケーション, 2003.

[2] Kent Beck, XP エクストリーム・プログラミング入門 – 変化を受け入れる, ピアソン・エデュケーション, 2005

[3] Manifesto for Agile Software Development (アジャイルソフトウェア開発宣言), http://agilemanifesto.org/

[4] 平鍋健児, 「対話」をする場面を「早めに」「多く」持つ, Agile Japan リレーコラム, 2009, http://www.agilejapan.org/2009/09/07150127.html

[5] Junichi Niino, アジャイル開発の現在・過去・未来, 2010, http://www.publickey1.jp/blog/10/post_121.html

[6] Ryushi Shiohama, Hironori Washizaki, Shin Kuboaki, Kazunori Sakamoto and Yoshiaki Fukazawa, Estimate of the appropriate iteration length in agile development by conducting simulation, Agile Conference (AGILE), IEEE Computer Society, 2012.

[7] Hakan Erdogmus and John Favaro, Keep Your Options Open: Extreme Programming and the Economics of Flexibility, in Extreme Programming Perspectives, Addison Wesley, 2002

[8] David F. Rico, Hasan H. Sayani, Jeffrey V. Sutherland, Saya Sone, The Business Value of Agile Software

Methods: Maximizing ROI with Just-In-Time Processes and Documentation, J. Ross Publishing, 2009.

[9] 独立行政法人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, 非ウォーターフォール型開発に関する調査, 2010, http://sec.ipa.go.jp/reports/20100330a.html

4.5.5. 環境の変化とソフトウェアテスト

ソフトウェアテストは、ソフトウェア開発全体の 40%程度を占める重要なアクティビティである。ソフトウェアテストに合格し、リリース直後は順調に稼働していたシステムが、ある日突然動かなくなることがある。また、取り扱うデータ量の増加によって著しくパフォーマンスダウンが起こり、社会インフラの停止を引き起こした例もある。本節では、環境の変化とソフトウェアテストについて考察する。

4.5.5.1. 二つのソフトウェアタイプ

ソフトウェアテストを行う時に、M. M. Lehman が 1984 年に定義した、三つのソフトウェアタイプ、即ち、S-type (Specifiable type) 、P-type(Problem-solving type)、E-type (Embedded type)について考えることが役に立つ。

S-type ソフトウェアは、例えば、sin(x)の計算結果を返すようなものである。S-type ソフトウェアは、仕様を厳密に確定できて時間の経過には左右されない。P-type ソフトウェアは、仕様は実世界記述するには不完全だけれども、作成したシステムを改善することを前提としてなんとかプログラムを作ることができるタイプである。そして、E-type ソフトウェアは、現実世界の作業をコンピュータに肩代わりさせるために開発するもので、業務に組み込まれるタイプと名付けられた。

ソフトウェアテストを実施する際には、テスト対象ソフトウェアを仕様が厳密に定義されている S-type として取り扱うことが多い。そして、仕様に対してテストを実施してバグが検出されなければリリースしてしまうということが行われてきた。

しかしながら、世の中のほとんどのソフトウェアは P-type もしくは E-type である。業務に組み込んで使用するのであるから、「環境の変化への対応」について十分考慮したソフトウェアテストを実施する必要がある。

4.5.5.2. 環境の変化がソフトウェアテストへ及ぼす影響

(1) バージョンアップ問題

ソフトウェアを取り巻く環境が変化するため、ソフトウェアは頻繁にバージョンアップを繰り返す。バージョンアップにおいて、プログラマは新規に開発する部分に注力して開発を行えばよいが、テスト担当者は、ソフトウェア全体(既存＋新規)に対して新規機能が仕様どおりに動作することはもちろんのこと、既存機能が問題なく働くことも確認する必要がある。

しかし、一般的にバージョンアップにおいて、新規開発時のようにソフトウェア全体を十分にテストできる工数を使えることは稀である。したがって、いかにして少ない工数で全体をテストするかが鍵となる。

(2) 環境の変化に起因するバグの発生問題

たとえ、1 行もソースコードに手を加えていなかったとしても、外部環境の変化によってソフトウェ

アの不具合が発生することがある。

例えば、動作していたサーバのグレードが上がり、CPU が 4 個から 8 個に変わることで並行処理のタイミングが変化してこれまで問題がなかったモジュールのバグが顕在化することがある。また、ソフトウェアへの入力として渡されるデータの複雑度が増してスタックを食いつぶすようになることや、当初想定していなかった負荷を持つ環境で使われること等々によって、いわゆる仕様想定外バグが発生する。他にも、Windows Update によって行われる OS のパッチ当てが予期しない問題の引き金になるケース等々が発生している。

現実のソフトウェアでは、突発的な高負荷が加わった場合などに、仕様書に書かれている条件や制約を超えて動作する場合がある。したがって、ソフトウェアテストでは、環境の変化を予知・予測してそれに対するロバストネスを評価する必要がある。

（3）運用の理解不足による問題

従来のソフトウェア開発では、実現する機能と品質特性に着目して要求・仕様・設計が実施されてきた。しかしながら、環境の変化は運用の場面で発生する。したがって、システム運用に対しても実現する機能同様に、要件定義段階から要求担当者、設計担当者、テスト担当者そして運用担当者が協業して知恵を集める必要がある。

4.5.5.3. 環境の変化に対するソフトウェアテストの対応

（1）頻繁なバージョンアップへの対応

頻繁なバージョンアップにおける課題は、少ない工数で全体を網羅的にテストすることである。有効な対策は二つある。

- **テストケースの管理と選択**

テストケースの管理と選択では、これまで実施してきた全てのテストケースを一元管理し、そこからバージョンアップごとに実施するテストケースを層別サンプリングしていく。この時に、前回のバージョンアップ時に実施したテストと重ならないように、できるだけ違うテストケースをサンプリングすることが重要である。

こうすることで、バージョンアップごとに少数の同じテストを繰り返すのではなく、数回のバージョンアップの中で全てのテストを実行するようになる。また、テストケースに重要度を付けて重要なテストケースは全てのバージョンアップ時にテストするということが行われている。

ALM（Application lifecycle management）ツールの中には、このようなテストケース管理ソフトウェアを含むものがある。

- **テストの自動化**

人手に頼りきったテスト工程から脱却するためのテストの自動化が効果的である。追加・修正したソフトウェアに対して、その周辺まで含めてこれまで実施したテストを再実行することを、リグレッションテスト（Regression testing）と呼ぶ。リグレッションテストを自動化するためにはプログラミングスキルを必要とし工数がかかる。しかし、リグレッションテストを 3～4 回実施するうちには、自動化への投資を回収することができる。また、人手によるテストよりも実行速度が速いためテスト工程の短縮にも繋がる。

2002 年に Kent Beck が執筆した“Test-Driven Development”という書籍を契機とし、TDD という「プログラミングする前にテストコードを書くこと」が行われるようになってきている。日本では、「テスト駆動開発」と呼ばれている。

現在、CI（Continuous Integration）ツールを用いて、TDD で作成したテストコードをシステム構築の度に実行する「継続的インテグレーション」が盛んになってきている。TDD と CI を導入することで、バグの混入にすぐに気づくことができるとともに、いつでもリリースできる体制が実現できる。

テストの自動化を推進することは、工数削減及び工期短縮に繋がる。そして、保守開発における新機能の追加が既存機能に悪影響を及ぼす問題を自動検知することで品質が向上するとともに、リリースが早くなるというメリットがある。

（2）環境の変化に起因するバグへの対応

外部要因が変わることに対する機能性の評価が求められている。つまり、外部要因（これをノイズと呼ぶ）の変化に対して機能が正しく働くことのテストが必要である。ノイズには様々なものがあるから、その全てをテストすることは不可能である。したがって、意図的に強いノイズを与えて、加速度テストを行うことで評価しようという対策が生まれた。これには、大きく三つのタイプがある。

- **負荷テスト**

従来の負荷テストは仕様で保証されている正常動作範囲を確認するものだった。強いノイズを与えるとは、この正常動作範囲を超えてシステムがダウンするまで実施することで、負荷テストと分けて限界テストという場合もある。

これは、正常動作範囲以外で障害が起こった場合でも、システムの破壊、課金情報など重要データの不整合が起こらないことを確認するテストとなる。ロバストネスなソフトウェアであれば、システムダウンに至る前に、システム保守要員に対して適切な警告メッセージを通知することだろう。最悪の場合でも重要データ不整合を発生することは起こさないことが期待される。

どこまで、ロバストネスを実現するかはシステムのプロファイルに依存する。システム定義の時に十分考慮しておく必要がある。

- **機能から目的機能へ**

仕様書に書かれている条件を超えて、環境の変化を予知・予測することが大切である。これは、仕様書に書かれている機能を、テストするという従来の方法から、その機能が将来に亘って果たすべき目的を予測し機能を目的機能に書き換えて、目的機能に対して評価を実施することへのシフトである。

目的機能とは、「システムの目的を得るための働き」と品質工学で定義されている。機能を単なる「働き」として捉えるのではなく、システム的な観点で市場導入後の信頼性を含めてその機能が果たすべき目的を考えてテストを設計することが必要である。目的機能は、システムが果たすべき目的をシステム導入時の目標だけではなく、システムがターミネートするまでのことを予知・予測して定義する。

- **ノイズの統計的分析**

直交表を用いてソフトウェア全体に対してノイズも因子に組み込んだ組合せテストを実施してノイズの影響を統計処理し、どのノイズ要因の影響が深刻かをテストする。ソフトウェア全体に対して組合せテストを実施することで想定外のバグを検出することができる。また、ノイズの影響を調べる

ことで、仮に耐性の弱いノイズ要因が見つかったとしてもその情報が明らかであれば、システム全体として（保守要員含めて）対策をとることが可能である。

（3） 運用の理解不足に対する対策

市場導入後の不具合について、テスト担当者が現象を再現テストで確認している場合が多いものである。この時に、ユーザと対話し不具合の周辺にある不満足の種を拾っておく。そして、それを要件定義段階のレビューに使用することで運用時のユーザの声をフィードバックすることができる。

また、コールセンターに入った VOC（Voice Of Customer）を集め、整理・分析し要求定義に活用する。これについても、要求担当者だけでなくテスト担当者も実施することがある。技法としては、コンジョイント分析（conjoint analysis）を使用する。

ソフトウェア開発の上流工程にテストの知見を適用することをWモデルと呼ぶ。今後、テスト工程のみならず全体の最適化についての議論が盛んになっていくと思われる。

4.5.5.4. まとめ

現在は、環境の変化を予知・予測してソフトウェアテストを行うことの大切さと、ソフトウェアテストという技術を要求定義という上流工程へ適用することに対していくつかの芽が生まれてきた段階と考えている。今後に向けてさらに要求自体を変化するものと捉えて、それを前提としたテストが重要になると考えられる。

4.5.6. 環境の変化とアシュアランスケース

4.5.6.1. アシュアランスケースを用いた解決方法

要求変化のマネジメントを有効に行うための手法としてアシュアランスケースがある。アシュアランスケースを用いたサービス提供判断方法の提案[1]では、サービス提供を例に、要求変化のマネジメントに有効な手法としてアシュアランスケースを取り上げ、サービス提供に至るプロセスにおいて普段から作成している内部文書である品質ドキュメントや各種成果物を、有機的に関連付けて整理することで、要求変化が発生した時の対応がスムーズに行えるとしている。

上記提案[1]では、サービスの提供にあたり、提供可否の判断条件を明確にして、十分なリスク対策を施し、サービス開始後に発生する様々な問題への対応に備えておくことの必要性を主張している。その実現手段の一つとしてディペンダビリティを確認するためのアシュアランスケースとしてディペンダビリティケース（D-Case）を用いたサービス開始判断方法を提案している。

ITIL のサービスマネジメント・プロセス（サービスストラテジ、サービスデザイン、サービストランジション、サービスオペレーション、継続的サービス改善）に対応付け、サービス提供に至るまでのプロセスを企画、開発、保守、運用、変更管理のフェーズとし、各フェーズを構成するカテゴリごとに D-Case を用いて、それぞれ保証条件を明記し、エビデンスを残すことが重要であるとしている。

4.5.6.2. アシュアランスケースを用いるメリット

サービス提供にあたりこのような D-Case を作成しておくと、以下の二つのメリットがある。

1)運用において発生した事象が、当初の想定内なのか想定外なのか、判断を素早く行うことができ、それに伴い素早くアクションに移すことができる

2)想定外の事象が発生した場合は、なぜサービス提供前に想定できなかったのか、分析を行うための材料が揃っているため、再発防止に向けたプロセスの改善に活かすことができる

例えば、一つの事例として、サービス提供後にサービス停止を伴う障害が発生し、その原因がプログラムのバグであった場合を考える。

サービス提供前の開発工程でテストが実施され、当該現象を把握していれば、問題のプログラムを修正し解決しておくか、運用時の制限としてサービスレベル定義に反映するか、対応をとるためそれらのエビデンスが残っている。制限に該当すると判明した場合は、その際に作成した運用マニュアルに従って回避策などの対応作業を粛々と進めることができる。

D-Case 及びそれに含まれるエビデンスを参照した結果、そのような準備ができていなかったと判明した場合は、なぜ検討が漏れていたのか、どのフェーズで、どのカテゴリで、どの保証条件が抜けていて、どんな作業を行い、どんなエビデンスを残しておく必要があったのかを分析し、再発防止のためプロセスを改善することができる。具体的には D-Case に新規追加・修正などの改善を実施する。

別の事例として、予めサービスの提供レベルを維持するために、モニタリング対象のデータが、閾値を超える場合を考える。

予め、閾値を超えた時の状況確認手順や対応手順を運用マニュアルに記載している場合は、粛々と作業を進めることができる。

そのような準備ができていなかった場合は、前述の事例と同様に D-Case でなぜ検討が漏れていたのかを分析し、再発防止のためプロセスを改善することができる。

いずれの場合も、サービス自体の継続、撤退、変更（追加・削除）を行うか否かを決定することがあることも考慮に入れておく必要があり、環境の変化が発生した時に判断する際のよりどころとすることができる。

その際に決断するためには、提供サービスの本来の目的や、成功と判断する基準、変更する際の決定者等を予め決めることが必要である。これがないと運用中に適切な判断ができなくなることから、これらの情報の整理に D-Case は有効な手法であるとしている。

なおこのような考え方はサービスに限らず、ソフトウェア製品の開発や企業内情報システムの構築についても同様に有効である。

表 4-12 サービス提供におけるアシュアランスケースの例（企画）[1]

L1	L2	L3	エビデンス
企画			
	目的	サービスを提供する目的を定義している	サービス企画書（目的）
	投資計画	投資計画を策定している	サービス企画書（投資計画）

		設備（ハードウェア、ネットワーク、ソフトウェア）調達計画を策定している	サービス企画書（投資計画）
	回収計画	回収モデルが定義されている	サービス企画書（回収計画）
		（回収計画を実現するための）サービスの開発計画を策定している	サービス企画書（回収計画）
		（回収計画を実現するための）サービスの展開計画を策定している	サービス企画書（回収計画）
	変更計画	目的を確認するプロセスを定義している（ヒアリング、アンケート、調査）	サービス企画書（変更計画）
		サービスの追加・変更に関して、変更管理プロセスを定義している	サービス企画書（変更計画）
		障害修正の本番環境への反映に関して、変更管理プロセスが定義されている	サービス企画書（変更計画）
		サービス変更を管理する体制を構築している	サービス企画書（変更計画）
	運用	回収計画をモニタリングする対象を定義している（利用者数、利用頻度、在庫、需要）	サービス企画書（運用）
		サービスレベルを定義している	サービス企画書（運用）
		サービスレベルを維持するために必要な監視対象を定義している	サービス企画書（運用）
		モニタリング対象の監視方法を定義している	サービス企画書（運用）
	セキュリティ対策	セキュリティ問題を未然に防ぐ計画を策定している	サービス企画書（セキュリティ対策）
	リスク対策	災害時の問題を未然に防ぐ計画を策定している	サービス企画書（リスク対策）
	コンプライアンス遵守	法令（輸出管理、労働法）上違反がないことを確認している	サービス企画書（コンプライアンス遵守）
		利用するソフトウェアにライセンス違反がないことを確認している	サービス企画書（コンプライアンス遵守）
	保守	保守プロセスを定義しているか	サービス企画書（保守）
		問合せ対応プロセスを定義している	サービス企画書（保守）
		要望・要件管理プロセスを定義している	サービス企画書（保守）
	承認	企画書を承認している	企画審査会議事録
		サービス開始基準を定義している	企画プロセス標準

表 4-13 サービス提供におけるアシュアランスケースの例（開発）[1]

L1	L2	L3	エビデンス
開発			
	開発計画	投入が投入計画の範囲内であることを確認している	開発計画書
	機能開発	機能要件を実装している（ツールが利用できる、サンプルが実行できる）	リリース判定報告書（機能開発）
		非機能要件を実装している（性能、メモリ、ディスク、ネットワーク）	リリース判定報告書（機能開発）
	サービス開発	サービスレベルを定義している	開発計画書
		サービスレベルを実現するための手順を定義している	開発計画書
		サービスレベルを実現するための手順のリハーサルを実行している	リリース判定報告書（サービス開発）
	運用	運用プロセス（フロー、ロール、着手条件、タスク、終了条件、成果物）を定義している	開発計画書（運用）
		運用手順を定義している	運用マニュアル
		運用テストを実行している	実施報告書（運用）
		監視対象を定義している	開発計画書（運用）
		監視手順を定義している	運用マニュアル
		監視のリハーサルを実行している	実施報告書（運用）
	セキュリティ対策	セキュリティ問題発生時の対応方法を実装している	リリース判定報告書（セキュリティ対策）
	リスク対策	データセンターを複数、異なる地点に設置している	環境構築報告書
		一方の拠点が被災してもデータが保存されるよう実装している	環境構築報告書
		緊急連絡体制を定義している	緊急時対応マニュアル

		緊急時の対応が行えるよう実装している	実施報告書(リスク対策)
	コンプライアンス遵守	ユーザ認証を行うよう実装している	リリース判定報告書(コンプライアンス遵守)
		データ漏えいのリスク対策を実装している	リリース判定報告書(コンプライアンス遵守)
		ユーザごとにアクセス制御の仕組みを実装している	リリース判定報告書(コンプライアンス遵守)
		利用するソフトウェアにライセンス違反がないことを確認している	リリース判定報告書(コンプライアンス遵守)
		海外からの利用について、輸出管理問題が解決していることを確認している	リリース判定報告書(コンプライアンス遵守)
	保守	保守プロセスを実装している	リリース判定報告書(保守)
		保守プロセス実施手順を定義している	保守マニュアル
		保守プロセス実施手順のリハーサルを実行している	実施報告書(保守)
	設備調達	本番環境に必要な設備(ハードウェア、ソフトウェア、ライセンス)を調達している	環境構築報告書
	出荷判定	基準に従ってサービス開始を承認している	リリース判定会議議事録
		サービス提供機能の品質が確保されていることを承認している	リリース判定会議議事録
		投資総額が計画内であることを確認している	リリース判定会議議事録

表 4-14 サービス提供におけるアシュアランスケースの例(保守)[1]

L1	L2	L3	エビデンス
保守(次への移行)			
	要件管理	利用者からの改善要望を要件管理している	要件管理台帳
	修正	本番環境で起きた事象を検証できる環境を構築している	環境構築報告書
		本番環境へ変更を反省する前にテストを実施する評価環境を構築している	環境構築報告書
		本番環境にリリースするモジュールの品質基準を定義している	リリース標準定義書
	問合せ	問合せを受け付け、回答を行う体制を構築している	実施報告書(問合せ)
		問題が起きた時に解決できるメンバで体制を構築している	実施報告書(問合せ)
	遂行	保守手順を実行している	保守作業管理台帳
		障害発生時にプロセス(問題登録、修正、リリース、再発防止策検討)に従って実行している	保守作業管理台帳
		保守作業を記録している	保守作業管理台帳

表 4-15 サービス提供におけるアシュアランスケースの例(運用)[1]

L1	L2	L3	エビデンス
運用			
	監視	監視対象の監視を実行している	運用報告書(監視)
		監視対象のログ取得を実行している	運用報告書(監視)
	サービス提供	サービスレベルを満足するサービスを提供している	週間作業実施報告書(サービス提供)
		サービスレベルを満足する問合せへの回答を提供している	週間作業実施報告書(問合せ対応)
	セキュリティ対策	セキュリティ問題発生時の対応方法を確認している	運用報告書(セキュリティ対策点検)
	リスク対策	データセンターが複数、異なる時点での運用を実行している	運用報告書(リスク対策)
		一方の拠点が被災してもデータ保存を実行している	運用報告書(リスク対策)
		緊急連絡体制を構築している	運用報告書(リスク対策)
		緊急時の対応を実行している	運用報告書(リスク対策)

	コンプライアンス遵守	ユーザ認証を実行している	運用報告書(コンプライアンス遵守点検)
		データ漏えいのリスク対策を実行している	運用報告書(コンプライアンス遵守点検)
		ユーザごとにアクセス制御を実行している	運用報告書(コンプライアンス遵守点検)
		利用するソフトウェアにライセンス違反がないことを確認している	運用報告書(コンプライアンス遵守点検)
		海外からの利用について、輸出管理問題は解決されていることを確認している	運用報告書(コンプライアンス遵守点検)
	コストマネジメント	計画に基づき、利用状況調査を実施している	利用状況調査報告書
	情報公開	サービスの情報公開を実施している	サービス紹介資料
	利用者教育	利用者教育を実施している	運用報告書(教育)

表 4-16 サービス提供におけるアシュアランスケースの例(変更管理) [1]

L1	L2	L3	エビデンス
変更管理			
	モニタリング	本サービスの投資に対する定量効果(利用者数、効果金額など)のモニタリングを実行している	利用状況報告書(モニタリング結果)
		本サービスの投資に対する定性効果(目的など)のモニタリングを実行している	利用状況報告書(アンケート結果)
		要件・要望を管理している	要件管理台帳
	サービス変更	サービスの追加・変更が必要か否か判断している	委員会会議議事録
		障害修正の本番環境への反映が必要か否か判断している	委員会会議議事録
		設備の増減が必要か否か判断している	委員会会議議事録
	プロセス変更	問題に対する再発防止策を企画・開発・保守・運用・変更管理プロセスに反映している	実施報告書(プロセス変更)
	承認	サービスの変更を管理する委員会において承認している	委員会会議議事録

補足) L1 はフェーズ、L2 はカテゴリ、L3 は保証条件を表しており、それぞれにエビデンスが存在する。

[1] 小林 茂憲, 山本 修一郎, 保証ケースを用いたサービス提供判断方法の提案, 信学技報, 電子情報通信学会 知能ソフトウェア工学研究会, 2012

4.5.7. IT サービスマネジメントにおける環境変化への対応

4.5.7.1. ITIL®の進化とサービス・ライフサイクル

ITサービスマネジメントのベストプラクティスをまとめてフレームワーク化したITIL®[1]は、1980 年代後半に英国政府のOGC[2](当時はCCTA[3])が作成した情報システムの運用管理基準であり、ITIL®のプロセスを適用したマネジメントシステムを実施する国際規格としてISO 20000 がある。

ITIL®は、IT サービスデリバリを中心とした初版(31 冊)、包括的なフレームワークとして統合された V2(7 冊)、サービス・ライフサイクル視点で強化・統合された V3(5 冊)、V3 を大幅に増補改訂した 2011 Edition がある。

V2 は世界的に受け入れられ、サービスサポートとサービスデリバリを中心とした現場プロセス改善が図られた。この時点では、各々のプロセス内での PDCA サイクルが改善の中心であった。

V3 では、「サービスストラテジ(SS)」での事業要件の最初の定義、「サービスデザイン(SD)」で

の分析と設計から、「サービストランジション（ST）」での稼働環境への移行を経て、「サービスオペレーション（SO）」での実運用、「継続的サービス改善（CSI）」での改善へと進むことで、サービス・ライフサイクル全体を網羅している（図 4-10）。

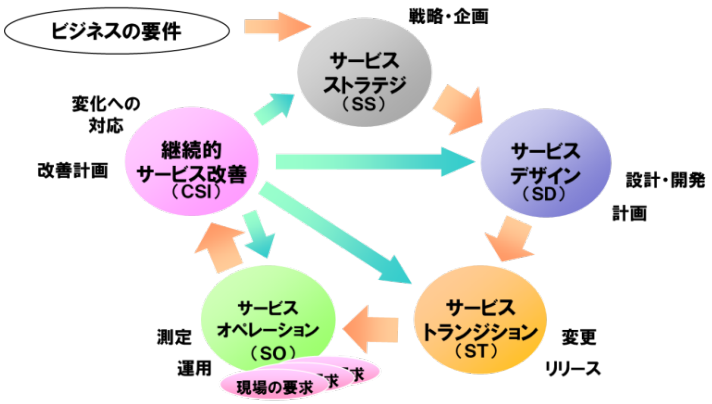


図 4-10 ITIL® V3 の全体像[4]

4.5.7.2. サービス・ライフサイクルと継続的サービス改善

サービス・ライフサイクルには、事業の要件（Requirement）を起点とするものと、現場で発生する要求（Request）の変化を起点とするものがある。事業の要件の変更はサービスストラテジで定義された一連の事業成果を実現するサービス・ポートフォリオ及び個々のサービスレベルパッケージ（SLP）として特定・合意される（図 4-11）。

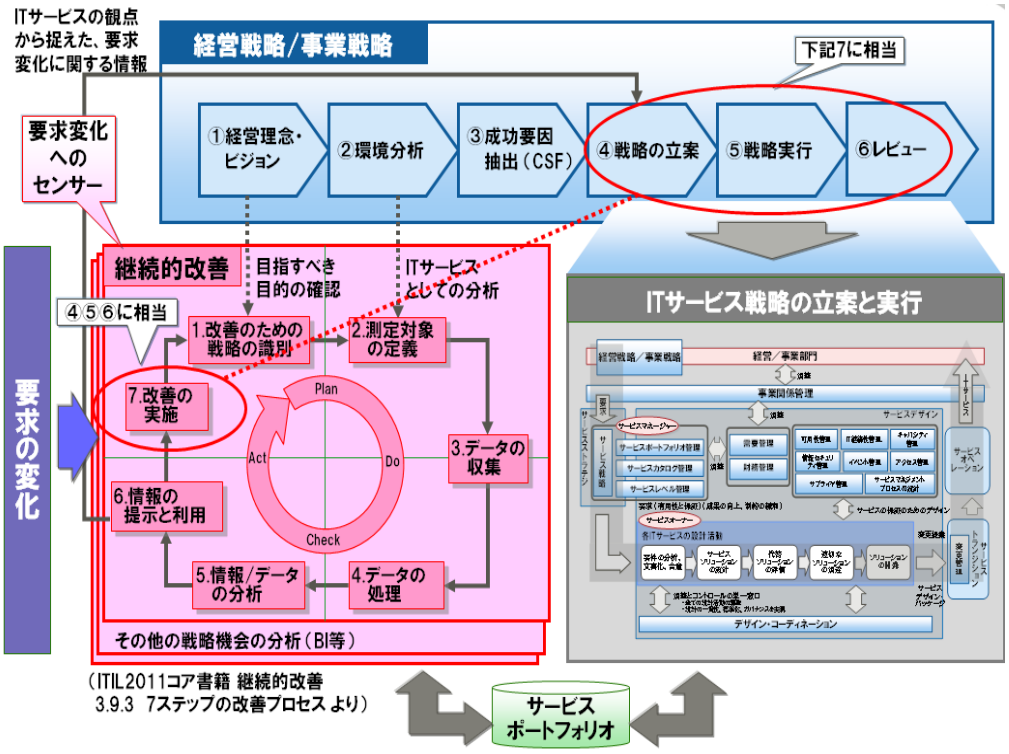


図 4-11 継続的サービス改善を中心とした要求変化への対応[4]

継続的サービス改善はライフサイクルのあらゆる段階で、弱みや障害の改善の機会を可能な限り識別する。これを、適切な段階にフィードバックすることで環境変化への対応が行われる。

2011 Edition では、経営戦略と IT サービス戦略が明確に分離され、サービスストラテジに事業関係管理(ステークホルダ関係管理)プロセスが追加されたことで、事業の要件を起点としたサービス要求での対応プロセスが明確になった。また、サービスデザインにデザイン・コーディネーション・プロセスが追加されたことで、より全体最適視点での変化への対応プロセスが明確になった。

4.5.7.3. システム・ライフサイクルとサービス・ライフサイクル

本報告書では、システム開発のV字に代表される工程の考え方に対して、V3 以降の ITIL®ではマネジメントを階層化した段階(フェーズ)という考え方をとっている(図 4-12)。

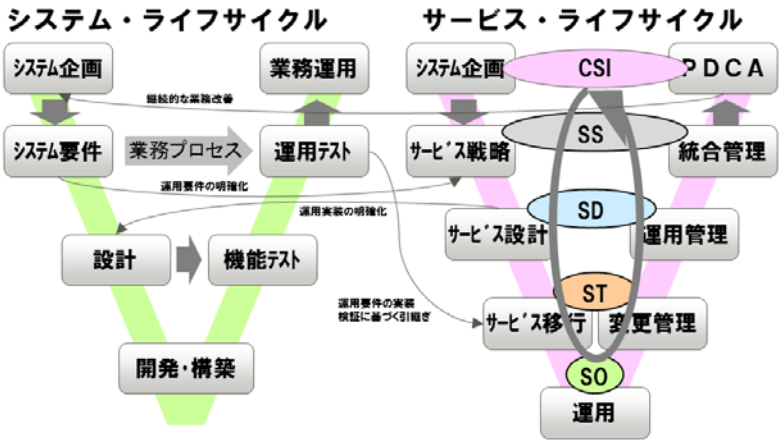


図 4-12 サービス・ライフサイクルと段階[4]

また、「サービストランジション(ST)」を独立させることにより、変化を前提としたプロセス体系となっており、日々の変更からシステム移行までを含んでいる。これにより、「継続的サービス改善(CSI)」に集まった変化に関する情報を基に、適切なマネジメント段階に遡って対応するプロセスを実現している。変化への対応をマネジメントし、ビジネス要求に整合したサービスの効率性と品質を追求していくことが、サービス・ライフサイクルだと言っても過言ではない(図 4-13)。

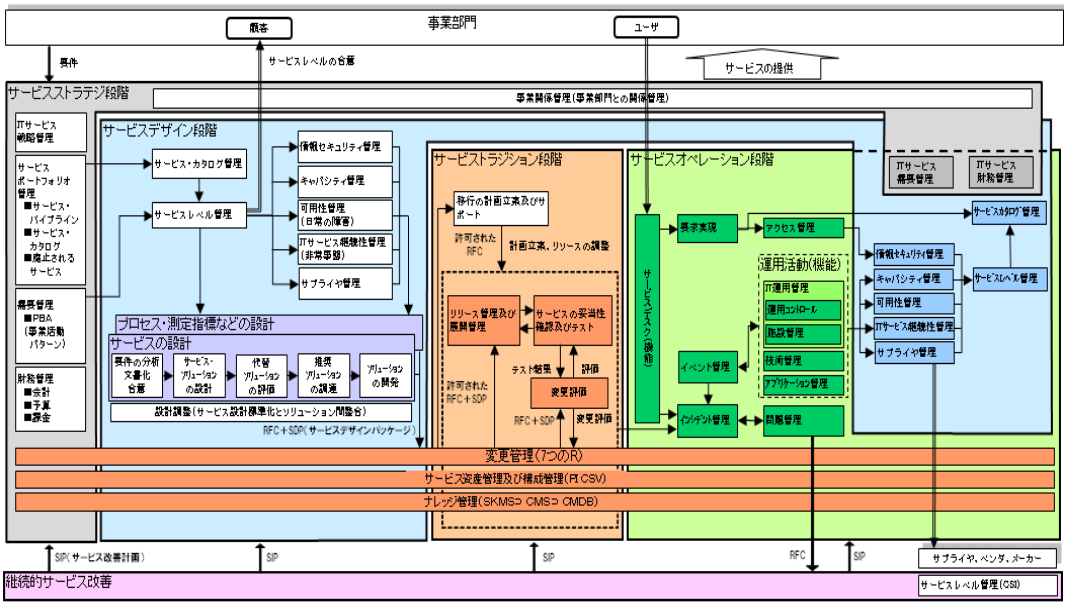


図 4-13 サービス・ライフサイクルの最適化を図るプロセス体系[4]

4.5.7.4. 現場の要求を起点としたサービス・ライフサイクル

ITIL®では、事業の要件の変更だけでなく、現場で発生する要求の変化にも着目している。現場で発生する要求には「対応手順が明確に定義されているもの」「サービスの維持や復旧を目指すもの」「サービス内容変更を必要とするもの」がある(図 4-14)。

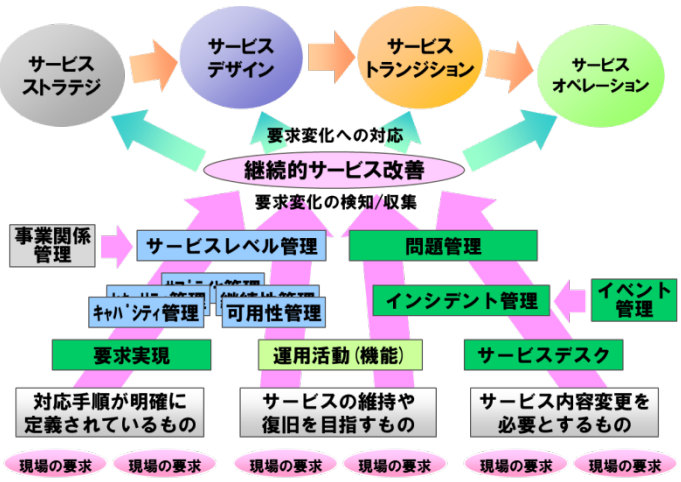


図 4-14 現場で発生する要求への対応[4]

「対応手順が明確に定義されているもの」とは、日常発生する特定の処理依頼に基づくものであり、サービスオペレーション(SO)の要求実現プロセスで処理されるが、処理頻度や処理時間の変化等はサービスレベル管理から継続的サービス改善に通知される。

「サービスの維持や復旧を目指すもの」は、サービスオペレーション(SO)の運用活動(機能)で処理され、サービスデザイン(SD)の可用性管理などを通じてサービスレベル管理に反映される。また、不具合についてはインシデント管理に紐付く問題管理から継続的サービス改善に通知され、

サービス改善計画(SIP)としてサービストランジション(ST)で変更が実施される。

「サービス内容変更を必要とするもの」は、その影響度により継続的サービス改善でフィードバック先が判断される。具体的には、サービストランジション(ST)で処理される障害修正レベルのもの、サービスデザイン(SD)で処理されるサービス仕様の変更を伴うもの、サービスストラテジ(SS)で処理されるリソース配分から見直すもの等である。

4.5.7.5. サービスの可視化とサービス基盤化、オープン化

以上、ITIL®における環境変化対応の仕組みを述べたが、サービスの提供プロセスを設計あるいは可視化する手法としては「サービス・ブループリンティング」が知られている。

サービス・ブループリンティングでは、横軸に時間軸、縦軸にサービス提供主体・役割・分業単位等を配置した 2 次元の平面図でプロセス全体を表現する。一般に、縦軸は「サービス環境を形成する物的要素」「顧客が典型的にたどるプロセス」「顧客接点を持つサービス・スタッフの役割・活動」「顧客接点を持たないサービス・スタッフの役割・プロセス」「サービスをサポートするプロセス」で構成することが多い。「顧客接点を持つサービス・スタッフの役割・活動」は顧客プロセスに応じて多様化するが、「顧客接点を持たないサービス・スタッフの役割・プロセス」はサービス基盤として効率性や信頼性に重点が置かれる。また、情報システムは「サービスをサポートするプロセス」の一つとしてサービス基盤を支える。

ITIL®では、顧客に対するサービスの単位が「コアサービス」である。コアサービスを実現するためには、顧客に直接的価値を提供する「実現サービス」群と、付加価値を提供する「強化サービス」群が必要となり、これらをまとめたものがサービスパッケージである。また、サービスを実現する構成要素として 4P(People、Process、Product、Partner)を提唱しており、情報システムを中心としたサービスでは顧客プロセス以外の大部分が Product としてシステム化される場合もある。

本報告書では、環境変化に強い IT サービスマネジメントとは、顧客接点における多様性への対応と、多様性に対応することから生まれるノウハウを蓄積し汎化したサービス基盤であり、この識別はサービスデザイン(SD)のデザイン・コーディネーション・プロセスで考慮すべき重要なポイントである。ノウハウが蓄積されたサービス基盤は、企業の枠を超えてサービスの在り方を変える(自ら環境変化を起こす)可能性を秘めており、現状の Amazon や Apple のマーケットプレイスなどが該当すると考えている。

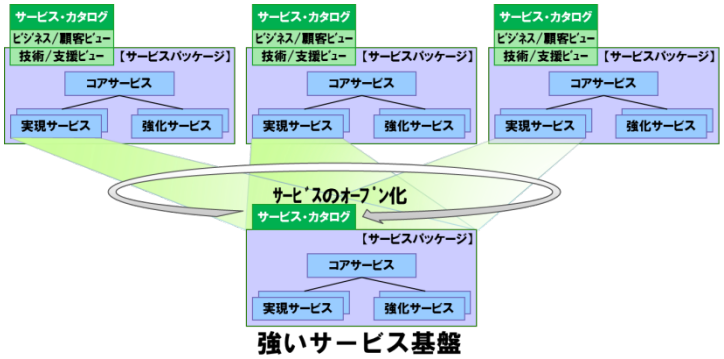


図 4-15 強いサービス基盤のオープン化[4]

ITIL®では、IT サービスマネジメントの目的を「IT サービスをビジネスの価値向上の目的に合致させること(Utility)」と「IT サービスのパフォーマンスを保証すること(Warranty)」としている。言い換

えれば、企画/開発/運用に携わる関係者が、ビジネスに貢献する IT サービスを一連のプロセスに則って構築/提供/改善していくことが ITIL®の目指すところである。旧来の「開発者の仕事は新機能を追加すること」「運用者の仕事は既存のサービスを安定稼働させること」という概念から、「IT サービスはビジネスの目的を実現すること」という考え方に変えていく必要がある。IT サービス提供責任を持つ運用部門が本番環境へのリリース権限を持っている企業や、開発者＝運用者の企業が増加しており、注目されている「DevOps」についてもサービス起点で考えることが重要である。Ops は Operations ではなく IT サービス提供でなければならない。

[1] "ITIL® is a registered trade mark of the Cabinet Office"
[2] OGC(Office of Government Commerce)
[3] CCTA(Central Computer & Telecommunications Agency)
[4] 富士通作成資料に基づく

4.5.8. システムエンジニアリングにおける要求変化への対応

システムエンジニアリングでは、要求が変わることはある程度許容はするものの、基本的には要求を決めて、それに基づいて開発をするという考え方が中心として存在していた。もちろん、要求が変化するリスクに対応するために、要求から設計、製造、試験、運用までのトレーサビリティをとるということを実施している。しかしながら、現実にはトレーサビリティで影響範囲を判断し、そこを変えるだけでは対応できないようなレベルの要求変化が発生してきている。このため、このような要求変化へのシステムエンジニアリングにおける取組みとして、ここでは二つの考え方を紹介する。

まず一つ目は、2004 年の INCOSE(International Council on Systems Engineering)の国際シンポジウムでベストペーパーの一つに選ばれた Aerospace 社の James Martine 氏による「"The Seven Samurai of Systems Engineering Dealing with the Complexity of 7 Interrelated Systems"」である。この論文によると、伝統的なシステムエンジニアリングでは、問題に対するソリューションとしてのシステムを考えているが、本来的には次に示す七つのシステムを考慮しなければシステムの要求は明確には規定できないという主張である。

- S1:コンテクストシステム
- S2:開発システム（例えば、ボーイング 747 など）
- S3:現実化されたシステム（例えば、ANA のボーイング 747 など）
- S4:展開されたシステム（例えば、ANA のボーイング 747-100 シリアルナンバー1）
- S5:共同システム（例えば、空港ターミナル）
- S6:維持システム（例えば、部品調達システム、機内食運搬システムなど）
- S7:競合システム（例えば、エアバス A380）

これらの七つのシステムを表すと図 4-16 のようになる。この時、七つのシステムの間に 15 の相互作用が存在する。このうち、2 番目が通常システム要求と呼ばれるもので 11 番目がインタフェース要求と呼ばれるものである。しかしながら、七つのシステムがあると認識した場合には、15 種類全てのインタラクションについて要求として考慮する必要がでてくることが分かる。以下に 15 個の

インタラクションを示す。

1. 問題(P1)はコンテクストシステム(S1)に含まれる
2. 開発システム(S2)は、問題(P1)を解決する
3. 実現化されたシステム(S3)は、開発システム(S2)を現実化したものである
4. 開発システム(S2)は、実現化されたシステム(S3)の現実化されたものの一つである
5. 実現化されたシステム(S3)は、コンテクストシステム(S1)を理解する必要がある
6. 実現化されたシステム(S3)は、S3 を実際に利用することによって変更されるコンテクストシステム(S1')を理解する必要がある
7. 実現化されたシステム(S3)は、維持システム(S6)を開発あるいは変更する必要があることがある
8. 開発システム(S2)は、展開されたシステム(S4)になる
9. コンテクストシステム(S1)は、変更されたコンテクストシステム(S1')になる
10. 展開されたシステム(S4)は、変更されたコンテクストシステム(S1')に含まれる(そして、相互に作用する)
11. 展開されたシステム(S4)は、一つかそれ以上の共同システム(S5)と共同する
12. 展開されたシステム(S4)は、維持システム(S6)によって維持される。
13. 展開されたシステム(S4)は、新たな問題(P2)を引き起こすかもしれない
14. 競合システム(S7)は、オリジナルの問題(P1)に対処するかもしれない
15. 競合システム(S7)は、展開されたシステム(S4)とリソースとユーザやオペレータの注意について競合する

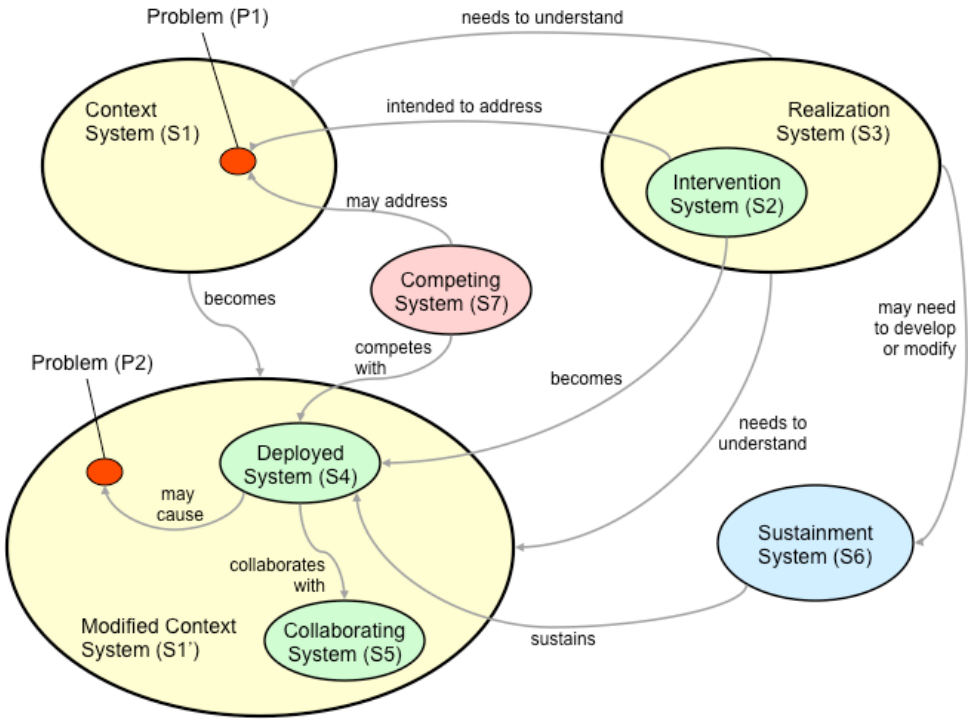


図 4-16 七つのシステム[1]

この The Seven samurai におけるポイントにおいて、要求変化の観点でも最も重要なものは変更されたコンテクストシステム（S1’）である。つまり、ある問題があり、それに対する対策としてソリューションを実現すると、そのソリューションによってコンテクストシステムが変更されてしまい、結果として新たな問題を引き起こすということである。例えば、自動車は、馬車では高速に移動できないという問題を解決したが、その結果として、交通事故や公害などの新たな問題を引き起こした。

このような問題に対処する学問として、米国 MIT（Massachusetts Institute of Technology）は、“Engineering Systems”という学問を提唱している。この Engineering Systems という学問では、これまでのシステムエンジニアリングにおいて、システムの外界あるいは外的制約として扱われていたものをシステムの範囲内として扱うことを主張している。例えば、図 4-17 において、システム範囲 1 では、電気自動車単体をシステムの範囲として扱っている。これは、伝統的なシステムエンジニアリングにおけるシステム境界の設定であり、技術的・工学的なアプローチのみで対応が可能となる。システム範囲 2 では、発電所やリチャージステーションをシステムの一部として捉えることで、利用も含めた全体の設計が可能となる。この場合は、リチャージステーションと電気自動車とのインタフェースを標準化することにより、System of Systemsとして進化的な開発が可能なシステムとして捉えることができる。伝統的なシステムエンジニアリングよりは広い範囲での対応となり、標準的なインタフェースを利用した協調的 System of Systems Engineering の範疇とはなるが、システム範囲 1 と同様に技術・工学的アプローチのみで対応が可能となる点に代わりはないといえる。最後に、システム範囲3では、エネルギー政策、インフラストラクチャ政策、税制などの政策がシステムの範囲に入っている。つまり、システム範囲 1 及びシステム範囲 2 と大きく異なるのは、これまでは技術・工学的な対象とされていなかった政策設計についてもシステムの内部として取り扱っていく必要があるとみなされているところである。これらを統合的に扱わなければ、政策などの理由により、要求が変わり、それによりシステムの設計が影響を受けるということが長い時間を考えると発生し得るのである。まだ、これらを統合的に扱う学術体系は確立していないが、世界的にこれらの動きを進めるために、Council of Engineering Systems Universities を 2004 年に立ち上げ、体系化に向けて世界で協力して研究を進めている。

現状ではまだ確立されてないが、システムの範囲及び非技術的な対象も含めた開発の方法論がポイントとなっている。

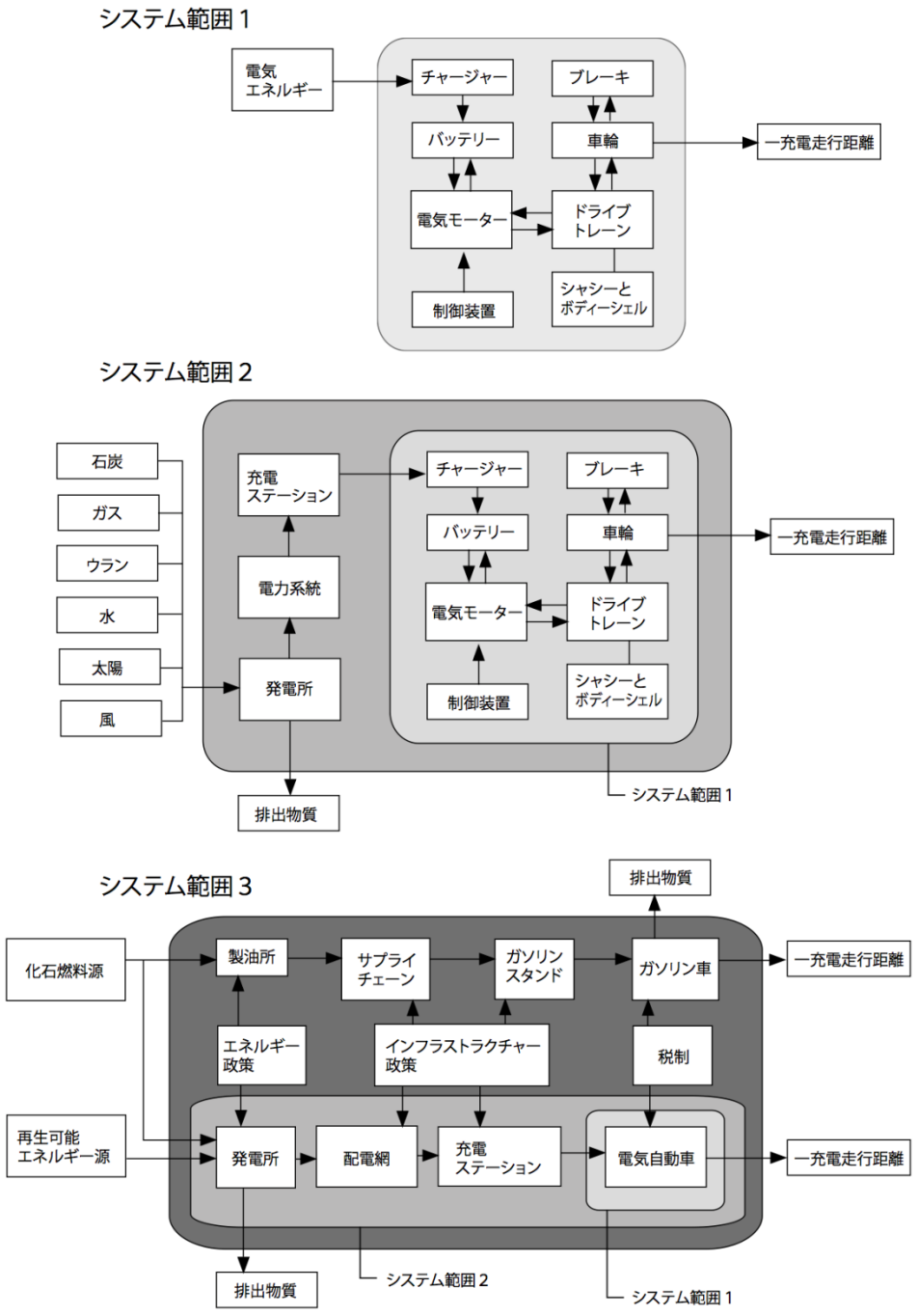


図　4-17 Engineering Systemsにおけるシステムの範囲[2]

[1] The Seven Samurai of Systems Engineering Dealing with the Complexity of 7 Interrelated Systems, James N Martine, 14th International Symposium INCOSE, 2004

[2] Engineering Systems: Meeting Human Needs in a Complex Technological World, Olivier L. de Weck, The MIT Press, 2011

4.5.9. 要求変化対応プロセスの事例

ISO/IEC/IEEE 29148 では、「開発管理の観点から、 要求を確定することが望ましいけれども、それが可能になるのは稀である」と指摘している[1]。以下では、要求発展型プロセスの事例として、要求変化対応プロセスの事例を紹介する。

4.5.9.1. ISO/IEC/IEEE 29148 の要求変更管理プロセス

要求変更の必然性を認識することと変更による影響範囲を緩和することが重要であることから、ISO/IEC/IEEE 29148 では、要求追跡と要求構成管理を用いて、提案された要求変更に対する影響評価、レビュー、承認プロセスを確立する必要があるとしている[1]。

要求変更管理では、プロジェクトにおける構成管理手続きに従って、要求変更レベルに応じて必要となる承認権限者と要求のベースラインを定義する。要求変更管理の対象となるベースラインとして機能（functional）、配置(allocated)、 開発(developmental)、 製品(product) がある。これらのベースラインに対する要求は、機能要求、設計に配置された要求、開発された要求、製品化された要求となるから、要求変更管理プロセスは上流だけでなくライフサイクル全体を対象とすべきであることが分かる。ここで、運用中の要求は、製品化された要求に対応している。

要求追跡表(Requirements Traceability Matrix、 RTM)を用いることによって変更要求に対する影響評価を容易化するとともに、要求変更が承認された場合 RTM を適切に更新しておく必要がある。

4.5.9.2. 要求状態管理プロセス

システム要求には、図 4-18 に示すような四つの基本的な状態がある[2]。まずシステム要求が抽出されると、ステークホルダによって合意される。合意された要求が実装されると通常運用される。合意されない要求は削除される。

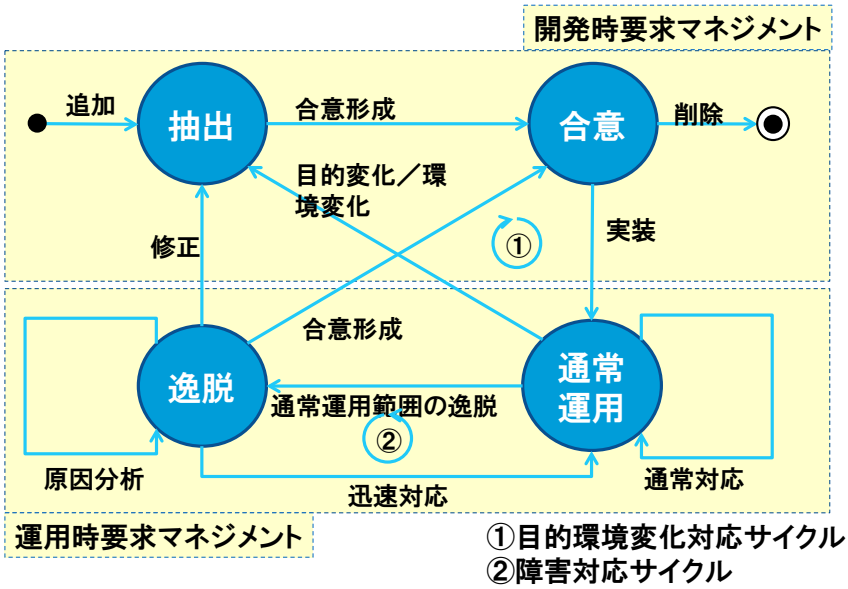


図 4-18 要求状態管理プロセス

要求が通常運用されると、目的変化・環境変化によって新たな要求が抽出されるか、当初想定した要求が通常運用範囲を逸脱する。通常運用範囲を逸脱した要求は原因分析される。逸脱した要求が迅速対応されると、通常運用に戻る。一方、通常運用に復帰できない要求には、要求に対する実装が誤っているかどうかによって二つの場合がある。もし要求に対する実装が誤っていれば、要求の合意状態に遷移して、要求を再実装する。もし要求に対する実装が誤っていなくて逸脱したのだとすると、要求を修正することにより新たな要求として抽出する。

この要求状態管理プロセスには、開発時と運用時の要求マネジメントがある。要求の抽出状態と合意状態は開発時の要求マネジメントの対象である。これに対して、通常運用と逸脱状態の要求は、運用時の要求マネジメントの対象である。

この要求状態管理モデルには、図示したように、DEOS プロセスにおける①目的環境変化対応サイクルと、②障害対応サイクルが対応している。

従来の要求変更管理プロセスでは、開発時の要求マネジメントプロセスしか考慮していなかった。したがって、要求状態管理プロセスが示すような運用時を含めた要求マネジメントが必要である。

4.5.9.3. MAPE-K による DSPL プロセス

ユーザ要求とシステム環境の動的は変化が頻発化するようになってきたことから、システムを自己適応的に、これらの変化に対応できるようにすることが重要になっている。このため、動的ソフトウェアプロダクトライン(Dynamic Software Product Lines、 DSPLs)が研究されている[3][4]。DSPL では予め設計要素の代替候補を用意しておき、環境変化に応じた適切な候補を動的に選択して製品構成を変更できる。

Bencomo らは、図 4-19 に示すような MAPE-K モデルを用いた 2 階層の適応型 DSPL を提案している。

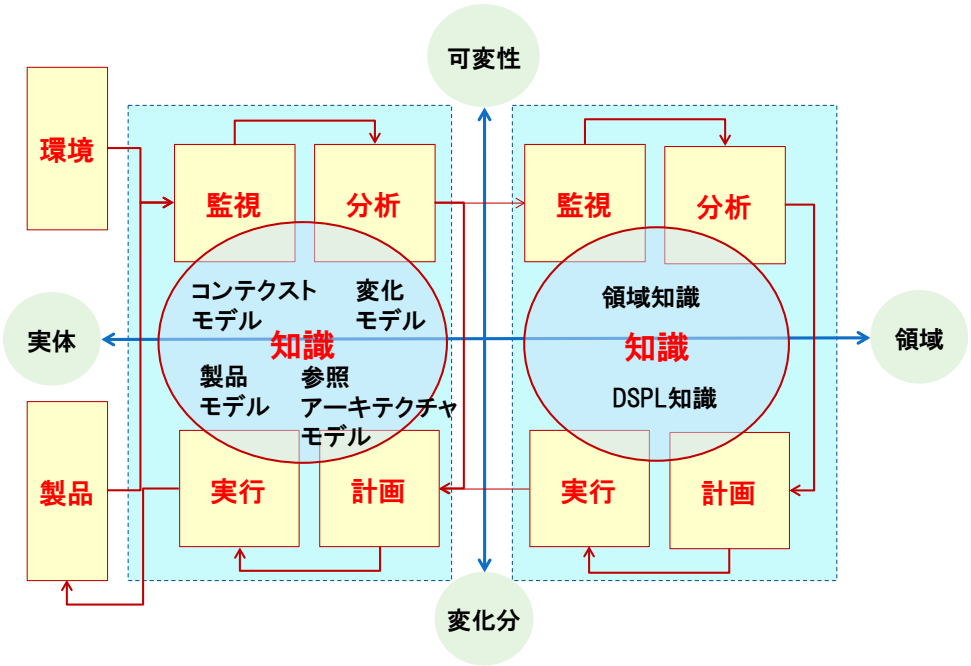


図 4-19 2 階層適応型 MAPE-K プロセス

MAPE-Kモデルは、監視（Monitoring）、分析(Analysis)、計画(Planning)、実行(Executing)、知識（Knowledge）から構成される。MAPE-KモデルはIBMが提唱したオートのミック・コンピューティングのための適応制御方式である[5]。

2 階層適応型 MAPE－K プロセスでは、領域レベルと実体レベルの 2 段階で MAPE-K を用いている。実体レベルの知識には、コンテクストモデル、変化モデル、参照アーキテクチャモデル、製品モデルがある。領域レベルの知識には、領域知識と DSPL 知識がある。実体レベルで環境と製品を監視した結果が分析されると、領域レベルの MAPE-K の監視並びに実体レベルの計画へフィードバックされる。

このようにして実体レベルでの環境変化対応の結果に基づいて製品要求が発展すると同時に、領域レベルの知識も発展する。

4.5.9.4. 発展と変革プロセス

Hitchinsは、図 4-20 のような問題に対してシステムが発展するプロセスを提示している[6]。

まず、現実世界の既存システムに対する問題が認識される。この問題を解決するために、理想世界における解としてのシステムが開発される。この解システムと現実システムが比較されて代替可能であれば、既存システムの変更計画が策定され、解システムが既存システムを代替する。システム環境の変化によって代替システムに対して新たな問題が発生すると、このプロセスが反復的に繰り返される。

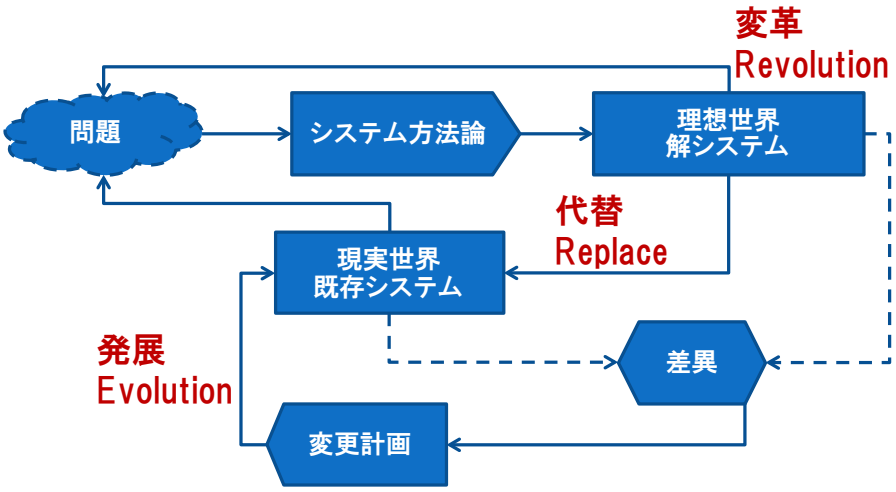


図 4-20 発展と変革プロセス

4.5.9.5. TOGAF の ADM プロセス

TOGAFアーキテクチャ開発手法ADMの工程を図 1 に示す[7][8]。TOGAFアーキテクチャ開発工程には、①準備、②A.アーキテクチャ・ビジョン、③B.ビジネス・アーキテクチャ、④C.情報システム・アーキテクチャ、⑤D.技術アーキテクチャ、⑥E.ソリューション、⑦F.移行計画、⑧G.実装監督、⑨H.アーキテクチャ変更管理、⑩要求管理という 10 個の工程がある。図 4-21 では、フェーズ名の側にADMの各フェーズで用いられる主な技法を明記している。

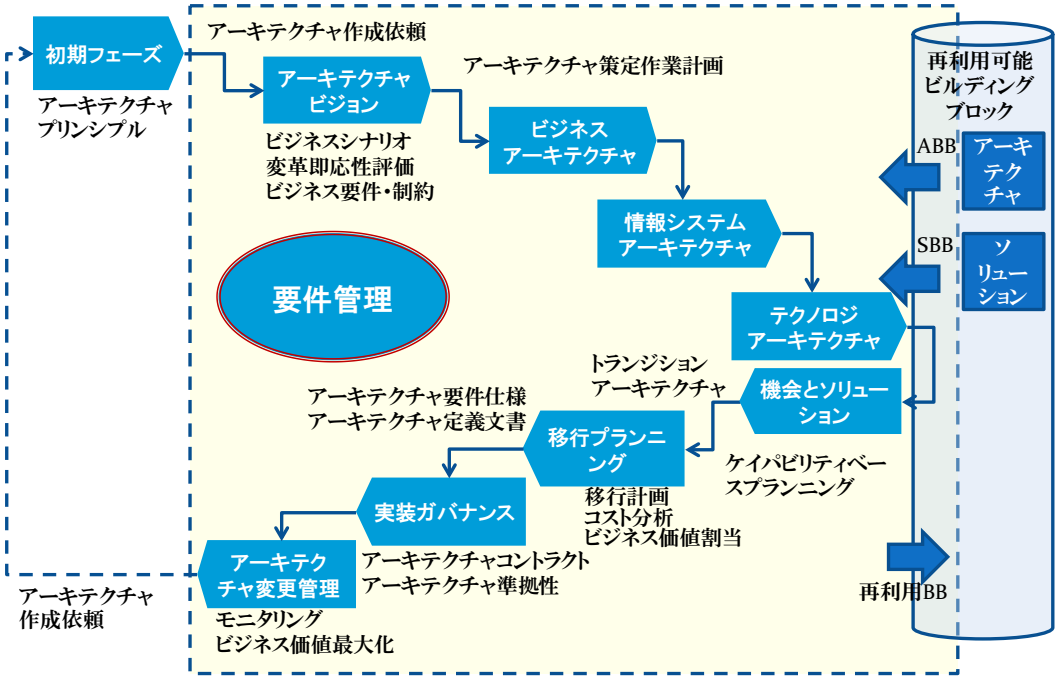


図 4-21 TOGAF のアーキテクチャ開発法（Architecture Development Method）

アーキテクチャフェーズでは、スポンサからのアーキテクチャ作成依頼に基づいてアーキテクチャ・プリンシプルを定義する。アーキテクチャ・ビジョンでは、ビジネスシナリオを作成することにより、変革即応性評価とともにビジネス要件とビジネス制約に基づいてアーキテクチャ策定作業計画書を作成する。アーキテクチャ策定作業計画に基づいて、ビジネス・アーキテクチャ、情報システム・アーキテクチャ、テクノロジ・アーキテクチャでは、アーキテクチャ定義とアーキテクチャ要件を作成する。この過程で、再利用可能ビルディングブロックを参照してビジネス・アーキテクチャと情報システム・アーキテクチャでは、アーキテクチャビルディングブロックを活用する。またテクノロジ・アーキテクチャではソリューションビルディングブロックを活用する。機会とソリューションではトランジションアーキテクチャを作成する。移行プランニングでは、ベースラインアーキテクチャからターゲットアーキテクチャへのアーキテクチャの移行計画を作成する。また、アーキテクチャ策定で得た経験を教訓として文書化する。

実装ガバナンスではアーキテクチャ・コントラクトを作成するとともに、プロジェクトのアーキテクチャ準拠性を確認する。アーキテクチャ変更管理では、プロジェクトをモニタリングすることにより、ビジネス価値が最大化できているかどうかを確認する。もしビジネス価値が最大化できていなければ、アーキテクチャ作成依頼を作成することにより、初期フェーズに立ち戻り ADM サイクルを反復する。

4.5.9.6. 要求構文を用いた要求変更管理

ISO/IEC/IEEE 29148 では、表 4-17 に示すような 3 種類の要求構文が例示されている。このように要求構文を規定することにより、要求が変化する要素を、条件、主体、対象、制約、活動、値、値条件に限定できる。したがって、これらの要求構成要素が変化した場合、対応する要素を含む要求構文と、その要求構文に含まれる他の要求構成要素を特定することにより、要求変更の

影響範囲を分析できる。このように考えると、要求構文を規定することが、実は、このような要求変更管理プロセスの可能性を示唆していることが分かる。

表 4-17 要求構文の例

要求構文の種類	例
[条件]、[主体] が[対象]を[制約] [活動]する必要がある。	信号 x を受信した時、システムが 信号 x の受信ビットを 2 秒以内で 設定する必要がある。
[条件]、[活動 または 制約] は[値]である必要がある。	海の状態が 1 の時、レーダーシステムが目標を検知すべき範囲は 100 nautical マイル である必要がある。
[主体]が [値条件]で、[活動] する必要がある。	請求書システムが 昇順 で、支払いが保留中の顧客請求書を表示する必要がある。

ただし、ISO/IEC/IEEE 29148 には、ここで指摘したような要求変更管理については言及されていないことを注意しておく。しかし、単純な RTM を用いるだけでは、ここで述べたような精密な要求追跡ができないことは明らかである。

4.5.9.7. まとめ

上述した要求変化対応プロセスの事例をまとめると表 4-18 のようになる。変化対象の粒度に応じて多様な対応プロセスが構築されていることが分かる。今後、変化対象の構造を一般化することができれば、これらの変更管理プロセスが統合できることが期待される。

表 4-18 変更管理プロセスの比較

変更管理プロセス	要求変化	粒度	プロセス
要求構文による変更管理	要求記述要素の変化	要求構成要素	変化要求追跡
要求状態管理	要求の逸脱	個別要求	要求状態遷移
ISO/IEC/IEEE 29148	要求ベースラインの変化	要求集合	要求構成管理
2 階層適応型 MAPE－K	環境と製品の変化	システム要素	MAPE-K
発展と変革	現状と将来のシステムの差異	システム	差異分析
TOGAF の ADM	現状と将来のエンタプライズアーキテクチャの差異	複合システム	ADM サイクル

[1] ISO/IEC/IEEE 29148:2011, Systems and software engineering ―Life cycle processes ― Requirements engineering

[2] MarioTokoro, Eds., Open Systems Dependability, Dependability Engineering for Ever-Changing Systems, CRC Pres, 2012

[3] S.O. Hallsteinsen et al., “Dynamic Software Product Lines,”Computer, Apr. 2008, pp. 93-95.

[4] Bencomo, N.; Hallsteinsen, S.; Santana de Almeida, E., A View of the Dynamic Software Product Line Landscape, Volume: 45 , Issue: 10, Computer,pp.36-41, 2012

[5] IBM. 2003. An architectural blueprint for autonomic computing. Tech. rep., IBM

[6] Hitchins, Derek, K., Systems Engineering- A 21st Century Systems Methodology, John Wiley & Sons, Ltd., 2007.

[7] The Open Group, TOGAF Version 9, 2009

[8] ReGIS Inc., TOGAF Version 9, 日本語訳版, 2010

5. アシュアランスケース

5.1. アシュアランスケースとは

ISO/IEC 29148-2011 で指摘されているように、現代システムの要求仕様を静的に確定することは困難である。例えば、システム環境の変化や、連携する他システムの変化によって、相互作用するシステムの要求が影響を受ける。したがって、システムが正常に動作する前提条件を明確にしておき、システム環境の変化がシステムに与える影響を確認できるようにしておく必要がある。

アシュアランスケースでは、システムが持つべき性質を主張として定義することにより、システムが置かれている状況を示す前提と、システムに対して確認できた証拠を用いて、証拠と前提から主張が成立することを論理的に説明することができる。

アシュアランスケースによってシステムの要求を主張として定義しておくことができれば要求変化に対するシステムの挙動の影響を客観的に確認することができる。

5.2. DEOS プロセス

システムの目的と環境変化によって、システム境界を確定できないことから、DEOS プロセスではステークホルダ合意に基づく変化対応サイクルを提案している。DEOS プロセスでは、ディペンダビリティを確認するためのアシュアランスケースとしてディペンダビリティケース(D-Case)を用いる。

システムの目的環境変化によって初期要求が逸脱すると、初期要求が逸脱したことが証拠として記録される。この初期要求逸脱の証拠によって D-Case によって合意された初期要求についての主張が成立しないことが次のようにして確認される。即ち関連するステークホルダに、客観的な証拠に基づく主張の逸脱に対する論理的な説明によって提示される。ステークホルダは D-Case に基づいて逸脱した初期要求を改訂して再合意することができる。

5.3. D-Case による要求のマネジメント

システム要求の逸脱が発生する条件を整理すると、表 5-1 のようになる。システムには必ず、正常動作するための前提条件がある。この前提条件が成立する限り、システムを正常に運用できる。しかし、この前提条件から逸脱する事象が発生すると、システムは正常に運用できなくなる。

この時、二つの場合がある。前提条件からの逸脱を想定できる場合と、そうでない場合である。想定できる逸脱を例外条件として明確に定義できれば、例外対策を用意することによって、システムの機能を継続して提供できる。しかし、この場合でも、提供できる機能は限定される可能性がある。

一方、想定できない逸脱に対しては、事前に対策を用意することができないので、事後対策手順を定義する必要がある。多くのシステムでは、事後対策手順が明確になっていないために後手の連鎖が生じることとなって事態が悪化することになる。

表 5-1 要求逸脱条件の分析

分類	場合	状況
1	前提条件が成立する	システムを正常運用できる
2	想定した例外条件が発生する	用意された例外対策を実施することにより、システムの機能を継続して提供することができる
3	前提条件に対する想定外の逸脱が発生する	想定外事象が発生することに備えた、運用プロセスにより、システム機能回復活動を実施する必要がある

表 5-1 に基づいて作成した D-Case を示すと、図 5-1 のようになる。

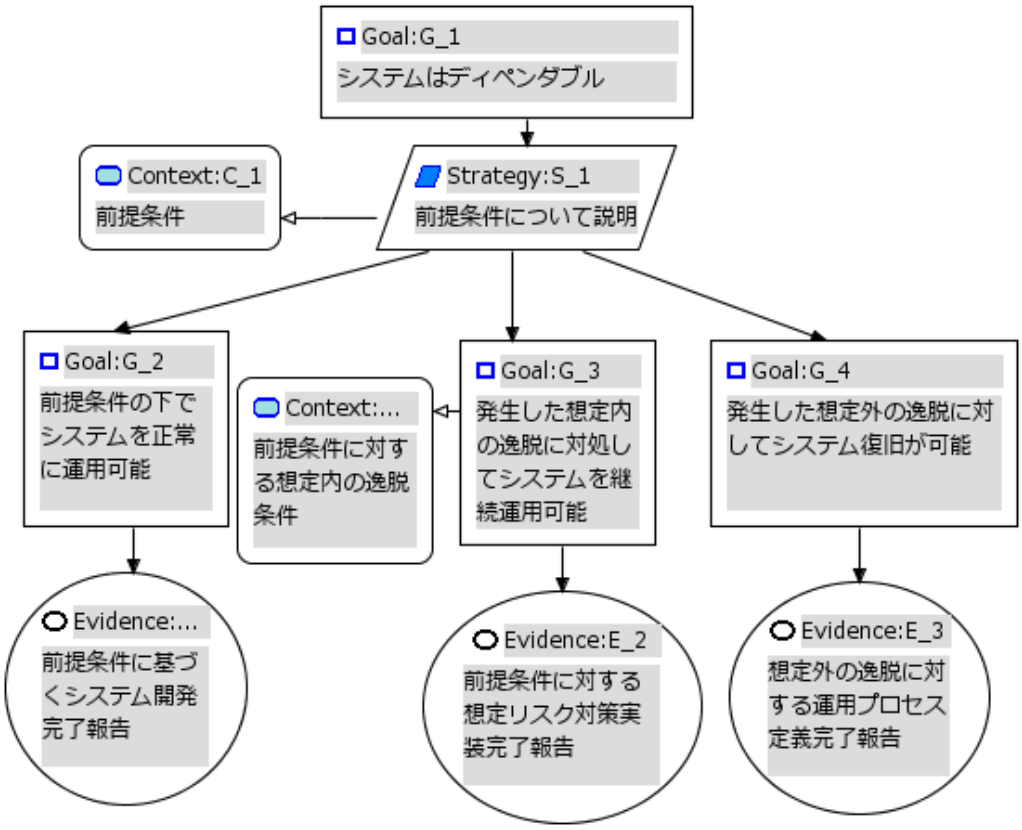


図 5-1 要求逸脱への対応を確認する D-Case

この D-Case は要求逸脱に対する一般的な対応を表現する参照モデルを示しているので、実際には、個別要求に対してこの参照モデルとしての D-Case を具体化することになる。

5.4. ゴール指向要求工学とアシュアランスケース

ゴール指向要求工学では、木構造を用いてゴール要求を下位要求に分解することによってシステムがゴール要求を満たすことを確認できる。この場合、最下位のノードにはシステムが提供する機能要求やコンポーネントが対応する。

図 5-1 で示したアシュアランスケース(D-Case)でも、木構造を用いてシステムが満たすべき主張が最下位の証拠と前提によって成立することを確認する。

このように、ゴール指向要求工学とアシュアランスケースは木構造を用いてシステムの特性を扱うことから、両者が同じなのか違うのか分からなくなる場合があるので、表 5-2 で比較した結果を示す。なお、実際には、ゴール指向要求工学には多数の手法があるので、具体的な手法を特定しないと、厳密な比較ができないことを断っておく。

表 5-2 ゴール指向要求工学とアシュアランスケースの比較

項目	ゴール指向要求工学	アシュアランスケース
用途	システム要求の獲得	システム要求の保証
分解	目的手段展開	前提と証拠への主張の説明分解
関係の方向	下位から上位	上位から下位
分解の種類	AND 分解と OR 分解	なし
分解の属性	肯定的と否定的	なし
関係の理由	分解関係に対する非機能要求	説明分解

ゴール指向要求工学とアシュアランスケースの比較として、ここでは分解の種類に絞り説明する。

ゴール指向要求工学手法では、ゴール分解関係として AND 分解と OR 分解がある。アシュアランスケースではこのような AND 分解や OR 分解を明示的に識別する記法は用意されていない。OR 分解がアシュアランスケースで必要とならない本質的な理由は、アシュアランスケースの目的が対象システムの妥当性を保証することであって、対象システムの動作を表現することではないからである。つまり、結論を導く全ての下位の主張について、個別に妥当性を確認する必要があるからである。したがって、対象システムの OR 条件ごとに、妥当であることを確認するためには、これらの条件ごとに説明する必要があることになる。

このようにゴール指向要求工学手法とアシュアランスケース手法は異なるので、逆に言えば、補完的に組み合わせることができるということでもある。今後、ゴール指向要求工学とアシュアランスケースを統合するための研究が望まれる。

6. サービスデザイン

成功したシステムがどのように環境変化に対応していったかを調べると、情報システム構築に焦点を当てているのではなく、サービスに焦点を当てていることに気づく。情報システムが環境変化に対応するためには、システム中心に見ているだけでは不十分なのである。

ビジネスが環境変化に対応する時に、どのようにサービスを設計しているのかという見方を知っておくことで、変化に強いシステムを設計できる可能性がある。特にサービスがオープン化に向かっている流れの中で、オープン化を踏まえたサービスデザインの考え方を取り入れることが大切である。

6.1. サービスの構成要素

日常、「サービス」という言葉は、「無料」や「奉仕」という意味で使われることが多い。本章ではこのような意味では使用しない。サービスを、「人間や組織体に何らかの効用をもたらす活動であり、様々な要素を組み合わせて顧客に価値を提供する活動」[1][2]と定義する。

サービスをデザインするには、サービスの構成要素[3][4]を知っておく必要がある。サービスはサービス項目とサービス活動に分解することができる。

表 6-1 サービスの構成要素

構成要素		内容
サービス項目	コアサービス	顧客が主としてその内容のサービスを利用するために、料金を支払っているサービス サービスメニューに載っているサービス
	付帯サービス	コアサービスに付随する副次的なサービス サービスメニューに載っているサービス
	臨機応変サービス	顧客の突発的な要求や火災や事故など定常業務を妨げる要因が発生した時に、その処理を行うサービス
サービス活動	サービス提供プロセス	顧客にサービスを提供する一連の活動、提供過程
	サービス時の態度	サービスの提供過程における提供側の態度
	ツール類	サービス提供時に用いられるツールやシステム

[1] サービスマネジメント入門 第 3 版, 近藤隆雄, 生産性出版, 2007

[2] ITIL®入門 IT サービスマネジメントの仕組みと活用, 野村総合研究所システムコンサルティング事業本部, ソーテック社, 2008

[3] サービスマネジメント入門 第 3 版, 近藤隆雄, 生産性出版, 2007 を参考に作成

[4] 顧客はサービスを買っている, 諏訪良武, ダイヤモンド社, 2009 を参考に作成

6.2. サービス提供プロセスの可視化

業務フローなどを用いてプロセスを可視化することは良く行われており、サービス提供プロセスを可視化する時も業務フローが用いられることがある。しかし、サービスにおいて重要な顧客の体験ポイントが表現されることが少なく、サービスデザインに活かせないことがあった。

この体験ポイントを明示的に表現する方法が、ビトナーらのサービス・ブループリント[1]である。サービス・ブループリントの構成要素と基本構造は次のようになる。

表 6-2 サービス・ブループリントの構成要素

サービス・ブループリントの構成要素
サービス環境を構成する物理的要素 （行動のトリガーとなるもの）
顧客の活動
表舞台での従業員の行動 （顧客接点を持つ従業員の対応）
舞台裏での従業員の行動 （顧客接点を持たない従業員の対応）
プロセスをサポートするツールやシステム

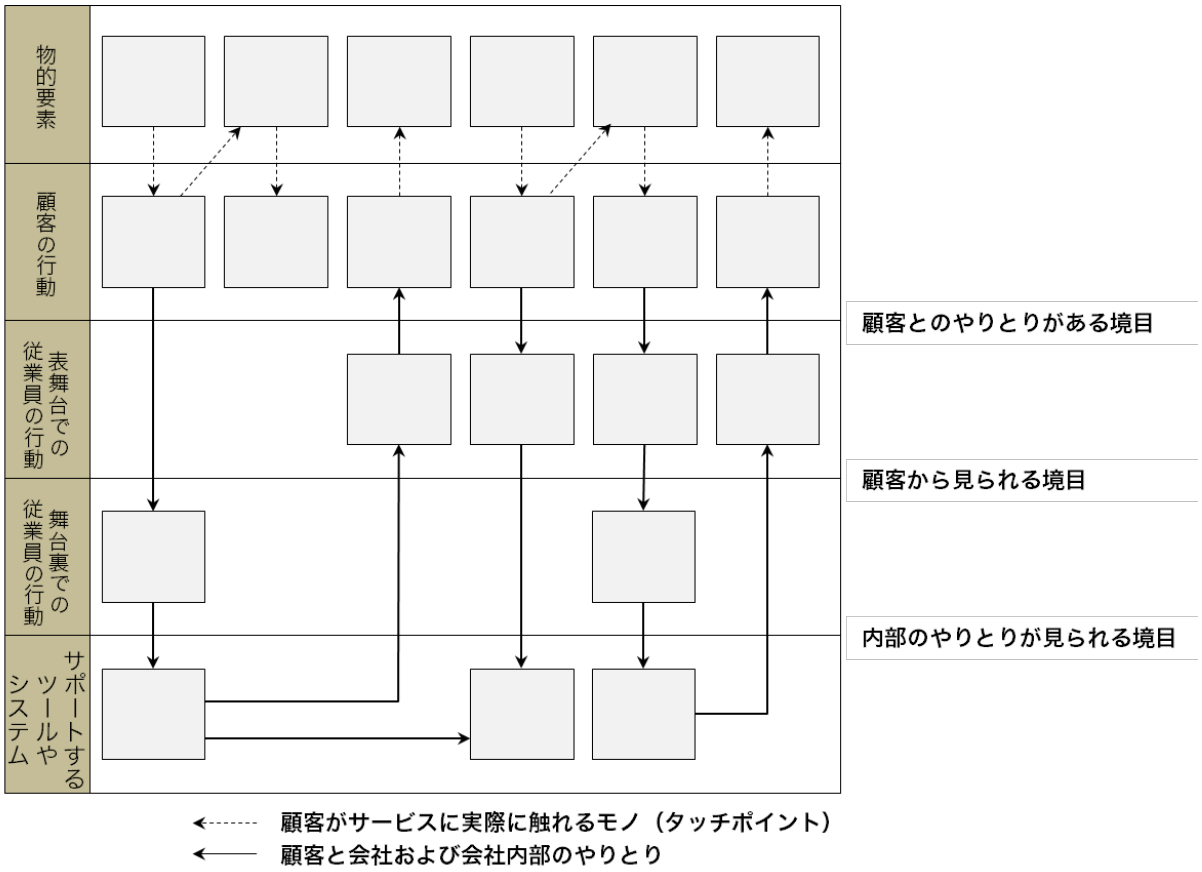


図 6-1 サービス・ブループリントの基本構造

サービス・ブループリントと従来使われている業務フローとの大きな違いは、サービスを受ける側である顧客を起点にして表現している[2]ところにある。サービス・ブループリントは、従業員の業務フローから書き始めるのではなく、顧客のサービス利用に関するプロセス（顧客の行動）から書き始める。従業員の業務フローから記述すると、必要以上に詳細に描き込み過ぎる傾向にあるが、サービス・ブループリントは、顧客を軸にプロセスを構成するため、ある程度の粒度が揃い記述しやすいという効果がある。

---

[1] Service Blueprinting, M.J.Bitner, California Management Review, 2008

[2] サービスサイエンス，亀岡秋男（監修），北陸先端科学技術大学院大学 MOT コース編集委員会 サービスサイエンス・イノベーション LLP, NTS, 2007

6.3. オープン・サービス・イノベーション

サービスにもオープン化の波が到来している[1]。自社だけのクローズしたプロセスの中でサービスを設計し、そのプロセスを守っていたとしても、競合他社がそのプロセスをオープンにし、新たなサービスを提供し始めると、結果として自社のシステムを競合他社のサービスに接続しなければならないという事態が発生する。

今のサービスプロセスを可視化することで、オープンにすべきポイントが見えてくる。サービス・ブループリントはサービスを立ち上げる時に使ったり、既存サービスを改善するために使うことができるが、サービスのオープン化を検討するために活用することもできる。また、顧客の体験ポイントに集中し顧客とともにサービスを作り出すこと（共創）ができないかどうかを検討することも、サービスのオープン化では必要なことである。

他社に対して自社のサービスを提供すると、サービス規模が拡大するにつれサービス運用知識が蓄積されてくるので、競合他社に対する競争力が付きさらに強くなっていく。ただし、単純に運用しているだけでは、専門知識は獲得できない。蓄積された情報を知識化する仕組みが必要である。

---

[1] オープン・サービス・イノベーション，ヘンリー・チェスブロウ，阪急コミュニケーションズ，2009

7. 環境変化に対応するシステムに関する課題

この章では、環境変化に対応するシステムの課題として次の七つの課題を取り上げる。

- ソシオーテクニカルの観点
- システム連携の観点
- 要素還元主義の問題
- システム運用の問題
- モニタリング負荷の問題
- 既存システムのテストの問題
- ユーザインタフェースの観点

7.1. ソシオーテクニカルの観点

社会生活の中に IT が浸透しているため、人間の行動によってシステムが影響を受けるケースが増えている。決められた人や限られた人がシステムを使うのではなく、一般の人が IT システムを使うケースが増えてきているからである。しかし、従来のシステム開発において、この使い方なら安全で、それ以外の使い方をされると何が行われるか、このような議論はあまりなされていない。そのため、ソシオーテクニカルな観点が必要になっている。

電話網システムのようにソシオーテクニカルな観点を用いたシステムは存在している。大規模災害時や年末年始、チケット予約などで輻輳が起きそうな時には発信規制が行われている。これらは、経験によって人間の行動を予測しながら対策を立てていた。しかし、最近は人と電話網システムの間に位置するスマートフォンのアプリケーションが、アルゴリズムによって予測不能な振る舞いをすることがあるので、想定外の事象が起こりやすくなっている。その結果、社会インフラのシステムに影響を与え、社会に対しても影響が生じる可能性がある。

社会への影響を最小限にするには、変化に対応できるプロセスを作り込み、安全性を十分に配慮していると言えるような説明責任を果たす必要がある。

7.2. システム連携の観点

従来のシステム開発では、システムの所有者が自分でシステム境界を設定し、その範囲で要件定義を行ってきた。しかし、現在起きている環境変化の問題は、システム境界が曖昧になってきたということに起因している。所有者自らが仕様を定義しても、環境の変化によりその仕様（システム境界）が所有者の意図とは関係なく変わっていく。要件定義時、仕様策定時に考えていたスコープの中では妥当だと思っていても、システム境界が変わりスコープが変われば正しいとされたものが正しくなくなる。

境界が定義できなくなってきている事例[1] である Flash Crashは、2010 年 5 月 6 日の午後に発生した米国株式市場における異常な株価変動のことである。原因はアルゴリズム取引とされたが、アルゴリズムを実装している売買システムにはバグはなく、超短期的に売り買いする売買システム

間の連携によって、株式市場全体に重大な影響を引き起こした。個々の売買システムと直接繋ぐところは問題がなく正常な繋ぎ方をしているにも関わらず、大量取引が発生してしまったことにより停止してしまった。

従来、接続先のシステムがどのように振る舞うかについてまでは考慮することは少なかったが、これからは接続先の振る舞い次第によって、自分のシステムが停止する可能性について検討しなくてはならなくなった。これは、システムの所有者が自分のシステムをコントロールできなくなっているということである。個々の売買システムには所有者がいるし、証券取引システムの基盤システムにも所有者がいるが、各売買システムが連携したトータルなシステムには所有者はいない。

システム全体の所有者がいないため、環境の変化に応じてシステムの境界が頻繁に変わる度に、ステークホルダ間の利害を調整し、定常的にステークホルダ間の再合意が必要になる。ステークホルダとの合意にはポリティカルな要素が含まれるため、ポリティカルな意見を考慮するようなプロセス、変化に対応するためのプロセスを作成する必要がある。

7.3. 要素還元主義の問題

ソフトウェアエンジニアリングの基本原理は、複雑なシステムを設計するために、ハンドリング可能なコンポーネントに分けてそれを繋げるというものである。しかし、新しいタイプの連携されたシステムでは、バラバラに分解した後に全体として繋げると、意図したとおりのシステムになるかどうかが分からない。コンポーネントの繋ぎ方を想定しているが、新しいタイプでは、想定どおりの繋ぎ方をしているかどうかが分からない。

この問題はシステム・アーキテクチャの必要性と類似しているが、個別のシステム・アーキテクチャではなく、個々のシステムを繋いだ全体の話であるので、今までのシステム・アーキテクチャで用いている方法が通用しないことがある。オープンシステムでは、他と協調するシステムの問題というのは従来からあり、そのためにアーキテクチャ設計を行う。アーキテクチャ設計では、特定のシステムをどのように設計するのか、全体を考えて設計する。例えば、NFR（非機能要件）である安全性やセキュリティは個々のコンポーネントで保証しても駄目で、システム全体で考えないといけないというものである。

これからの問題は、特定のシステムのアーキテクチャ設計ではなく、所有者がいない、つまり、全体を管理するものがいないという中で、全体をどのように設計するかという新しい方法が見えていないのである。

7.4. システム運用の問題

システム運用のオペレーションにおいて、システム化できるところは徹底してシステム化してしまうと、想定外の障害が発生した時オペレーションがブラックボックスになってしまうため、原因が分からなくなるリスクがある。

システムは正確ではあるが柔軟ではない。一方、人は柔軟ではあるが正確ではない。人の不正確さに着目し、システム障害撲滅のためにシステム化を推進する場合、行き過ぎると変化への柔軟な対応が難しくなることもある。

環境変化に対応してシステムを変更していくと、当初シンプルだったシステムも複雑になってい

き、変化対応のスピードが劣ってくる。それをシステムの老化とみなして、つまり予めシステムの寿命を決めて運用するということを検討する必要がある。

7.5. モニタリング負荷の問題

環境の変化に気づくためにはモニタリングする箇所を増やせばよいが、モニタリングする項目を増やせば増やすほど負荷が重くなる。モニタリング負荷を軽減するために、モニタリング項目を減らすと漏れてしまう。この矛盾に対してどのように対応すべきか検討する必要がある。

7.6. 既存システムのテストの問題

環境の変化に対してシステムが追随できない理由の一つが、既存システムの仕様が分からないために多くのテストを実施しなければならないということである。結果としてユーザが期待している時期にリリースできないこともある。

既存システムのテストケースを減らせないため、テスト実行部分を自動化することにより期間を短縮するアプローチを検討する。しかし、テストスクリプトのメンテナンスコストも無視できないため、テストの自動化は最も効果の高い箇所に部分的に採用されるにとどまっている。

次のアプローチとして最大リスクを低減させるようなテストを考えるが、客観的な方法はなく、最後には人間の判断で行われる。どのようなアプローチがよいのか検討する必要がある。

7.7. ユーザインタフェースの観点

現場は忙しく、余裕がないという前提で、環境変化への対応を考えるべきである。特にユーザインタフェースの学習コストを考えないといけない。例えば、ある病院の情報共有システムでは、普段使っている Facebook 風なインタフェースを採用した。その理由は、新たに使い方を学ぶことなく、すぐにシステムを使えるからである。

---

[1] Large-Scale Complex IT Systems, Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid, and Richard Paige, Communications of the ACM, JULY 2012, vol. 55, no. 7, pp.71-77

8. 環境変化への取組み事例

本章では、企業が環境変化に対応するためにどのような取組みをしてきたかを紹介する。なお、日本企業 5 社にヒアリングした内容に基づいて記す。

8.1. A 社の事例(製造業)

ポイント 1:早く安く作るための手法を組み合わせる

ビジネスのグローバル展開を踏まえた顧客マスタを作成する時、各リージョンでの要望の調整に手間取り、統制がとれなくなる。そのため、より早くより安くまず走り始めることが重要であると考えている。特に統計データはいずれにせよばらつきが生じるので精度が必要なオペレーションとはアプローチを変えて、パッケージ導入、ラピッドプロトタイピング、アジャイル開発などを検討している。どのアプローチを採用しても、開発当初にトータルコストで投資対効果を確認できることが大切であると考えている。

ポイント 2:変化の影響を見極めて対応を講じる

シンクライアント、タブレット、スマートフォンなど業務で使用する端末の候補がいろいろ登場しているが、業務で必要な要件を見極め一時的な動向に左右されないことが大切である。また、変化するフロント部分と変化が少ないバックオフィス部分を区別して扱うようにしている。顧客接点であるフロント部分と、プロフィットや課金管理などのバックオフィス部分とで、IT 化対応部門を変えている。

社外向けサービスについては、どんどん作ってみて駄目なものは捨てなくてはならないものもあり、パーフェクトを求めがちな旧来の IT 部門の人には不向きなケースがある。そのため、スクラップ&ビルドに慣れている R&D 部門の人間を IT 部門に異動させ対応したことがある。

ポイント 3:予測できる変化は予め対処する

予測できる変化は既に対策を講じており、例えば、新商品を追加しても、システム変更はあまり発生していない。

ポイント 4:ビジネスと IT の連携をとり、IT 投資の成果評価を行う

ビジネスと切り離した IT はあり得ないと考え、ユーザ部門と密に連携をとってビジョンを共有できる会議体を設定している。IT システム開発を計画する際には必ず必要な成果刈り取りの評価を行うこととしており、単なるシステムの入れ替えでは成果とみなさないので、そのような再構築は行わない。また、IT システム再構築を伴わないビジネス上での BPR 達成についても検討するようにしている。

8.2. B 社の事例（卸売業）

ポイント 1：変化の種類に応じて、標準化とカスタマイズを切り分ける

過去、グループ各社のシステムに対して、パッケージ導入やマスタ整備を推進した結果、業務の標準化を実現したが、パッケージのカスタマイズ比率が 70%もあった。コストがかかったことばかりでなく密結合構成であったため、一箇所止まると全てが止まってしまう問題を抱えていた。

これら課題に対応するため、会計業務など普遍的な業務については既存の仕組みを継続して適用するが、変化の激しい業務についてはパッケージ製品をやめて部品化を進め、密結合を疎結合にしてどこか一つが使えなくなってもバイパスする仕組みを作ることとし、SOA を基盤にしたアプリケーションに移行することとした。狙いはよかったが、ウォーターフォール型開発で実施したため多大な時間とコストを要した。

この課題を受けて、現在はプレゼン層／ロジック層／データ層の中で変化の激しいプレゼン層に対してアジャイル開発を適用している。ワークスタイルの変革に対応するために、基幹システムで提供している機能をマルチデバイス対応に改修しているが、その開発をアジャイルで行っている。

アジャイル開発ではユーザと意識合わせをやりながら開発を行う。具体的なイメージが開発者に伝わり齟齬がなくなったため、従来のウォーターフォール型開発と比較して約 6～7 割の期間で開発ができている。

また、全てを開発するのではなく、CRM など営業活動に関わるものは、早い時期からクラウドサービスを採用している。

変化の種類に応じて対応策を検討することが大切だと考えている。

ポイント 2：ユーザ部門に IT 部門から積極的に提案する

かつてのシステム開発は基幹業務のシステム化がメインであったため、IT 部門からユーザ部門へは積極的に提案は行っていなかった。今はビッグデータなどマーケティングのために現場のビジネスにいかに貢献できるのかへと領域が広がっており、IT 主導でいろいろ積極的な提案ができている。

IT 部門からの提案を積極的に行える仕組み作りとして、IT 部門の約 20%の人数を BPR 部隊として編制した。社長をはじめとする経営層と IT 部門長が一体となって各部門へ BPR 部隊の人員を配置してどこを IT 化した方がよいかを確認して提案を行うなど、現在では IT 部門から仕掛けることができている。

ポイント 3：競合他社との共創を検討し始めている

営業部門は競合他社がライバルであるが、IT 部門は同じ悩みを抱えるなど協力してできることがある。実際にバックオフィスに関わる基盤は共通する部分が多いため、共同開発の検討を始めている。データ層、ロジック層は業界標準で、プレゼン層は各社 API で作るというのも一つの有様と考える。

8.3. C 社の事例（運輸業）

ポイント 1：要求変更による手戻りを減らすために、アジャイル開発を採用する

ウォーターフォール型開発では、開発規模が大きく開発期間が長期（3～4 年間）に亘るケースが多く、ビジネスのスピード感に追い付いていない。特に、開発途中に当初の要求が変更になり、手戻りが発生することが多い。

要求の変化に対応しつつ、手戻りを減らすことができるアジャイル開発に取り組んでいる。ただし、全てのシステム開発をアジャイル開発で行っているわけではなく、業務の種類によって使い分けている。Web関連のシステム開発や、パッケージソフトを使用するシステム開発にはアジャイル開発を選択し、バックエンドでいろいろなシステム連携が必要なシステム開発の場合はウォーターフォール型開発を適用している。

ポイント 2：予測できる変化は予め対処する

変化の影響が予測できる場合、パラメータの変更等で対応できるように予め対応するようにしている。また、プログラムの部品化も進めている。

ポイント 3：ベンダ依存からの脱却のためにクラウドサービスを利用する

4～5 年で保守切れ・更新対応が必要など、ビジネス環境の変化とは関係ないベンダの理屈で対応せざるを得ないケースが発生していた。ベンダ依存型の投資スタイルを変える一つの手法としてクラウドサービスの利用を進めている。

8.4. D 社の事例（金融業）

ポイント 1：早く作るために様々な手法を組み合わせる

ビジネス環境の変化に適応するために、システム開発のスピードを上げなくてはならない。スピードを上げるために、パッケージソフトの導入やアジャイル開発を採用している。

パッケージソフトの場合、データモデル・プロセスモデル等基礎となるものが用意されており、パッケージにない部分（インタフェース等）を外側で作ることに専念することで、開発スピードを上げている。

アジャイル開発では業務部門の協力が不可欠であるため、業務部門の担当者や責任者が、拠点の離れたシステム開発現場に週 2～3 日通ってもらい、対面で要件を詰める等を工夫した。アジャイル開発のノウハウを習得後、スマートフォン系システム開発をアジャイル開発で行った。

ポイント 2：変化に柔軟に対応するため、自主性を尊重する

海外子会社でのシステム開発では、アプリケーションオーナーの考え方や重要案件の進捗管理方法など社内のプリンシプルとなる部分は共有するが、原則買収した現地の会社の自主性を重視して、意思決定を行っている。

ポイント 3：工期短縮、工数削減のためにツールを活用する

テスト自動化ツールを用いてバリエーションのあるテストを用意するなど、工期短縮、工数削減に取り組んでいる。

8.5. E 社の事例（サービス業）

ポイント 1：変化の種類に応じて、開発手法を組み合わせる

基幹系のコアとなる勘定系業務システムは変化の影響をあまり受けないので、ウォーターフォール型で開発を行っており、計画・管理型で対応している。

一方のフロント系システムは、新しい技術を使わないと競合他社に後れをとることになる。そのため、部品化、標準化、モジュールの共有化はほとんど行わず、アジャイル開発で対応している。

ポイント 2：競合他社の変化に応じて、組織を頻繁に改編する

インターネットビジネス環境では、競合他社の変化が速く、他の産業の比ではない。50 以上のビジネス領域を持ち、常時 300～400 プロジェクトを抱えているが、原則プロジェクトが期をまたがることはなく、3 カ月以内で対応するようにしている。また、各プロジェクトの規模は 10 人～40 人規模である。毎年、組織を大改編しており、途中で組織を打ち切り、新しい組織を作ることもある。

ポイント 3：状況の変化に応じて、外部サービスを利用する

原則外部ベンダを使わず内部要員で開発した自社システムを活用しているが、トランザクション件数が跳ね上がった時にはパブリッククラウドサービスを使うような仕組み作りをしている。

ポイント 4：技術動向を常にウォッチし、必要な人材を確保する

新規技術の研究コミュニティには積極的に社員が参加しており、アカデミックとの距離は非常に近くなってきている。コンピュータサイエンスの博士号を持った人を積極的に採用している。現在、データサイエンティストの育成に取り組んでいる。

8.6. 考察

ヒアリングした各社では、それぞれの事業の実情及び経験に基づいて、ビジネス環境の変化に対応するために独自の工夫を行っている。全ての企業が、要求の変化に俊敏に対応するために、何らかの形でアジャイル開発手法を導入している。また、ほとんどの企業（C 社以外）では、変化に柔軟に対応可能な組織体制をとっている。特に、各社のコアとなる事業においては、変化対応のノウハウが蓄積されており、IT システム等にその対策が予め組み込まれている（A 社、C 社）。さらに、アーキテクチャ上の工夫（A 社、C 社）やテスト自動化ツールの採用（D 社）など、本報告書の 4.5 節 環境変化適応の事例で紹介した取組みを行っている企業も見られる。

しかしながら、いずれの企業も、本報告書で示した環境適応モデルや事例ヒアリングのベースとした「柔軟化」のような、体系的な取組みの整備には至っていない。例えば、環境変化の予兆を

いち早く捉えるための環境監視の仕組みについて、特別の取組みを行っている企業は見当たらない。あるいは、個々の事業における対策はとられているものの、変化の対象や特性等に応じてどのような対応をとるか、といったことを整理している企業も見当たらない。

このような状況について、CMMI（能力成熟度モデル統合）で規定されている 5 つの成熟度レベル（レベル 1：初期、レベル 2：管理された、レベル 3：定義された、レベル 4：定量的に管理された、レベル 5：最適化している）にたとえるならば、レベル 1～レベル 2（一部）といったところであろうか。即ち、各社とも、環境変化対応を陽に意識した取組みがなされていないということがいえる。

今後、ビジネス環境の変化は、そのスピードと範囲の面のみならず質の面でも、激しさを増していくものと思われる。そのような状況において、各企業等には、特にそのコアとなるサービスや事業について、利用者に安心してサービスを使い続けてもらうために、あるいは競争優位を保ちつつ事業を継続するために、環境変化対応のより成熟した取組みが求められる。

9. まとめ

第 1 章では、本 WG が発足した背景と目的について述べた。

第 2 章では、2010 年度から 2012 年度までに実施した活動をまとめた。2010 年度は情報システムの保守を容易化する技術を調査した。2011 年度は要求の変化に対応する情報システム構築技術の適用に関する調査を実施した。本年度は、環境の変化に対応する情報システム構築技術を調査するとともに、環境変化対応における基本的な課題を整理した。

第 3 章様々な環境適応モデルでは、代表的な環境適応モデルとして①生存可能システムモデル（Viable System Model:VSM）②ISO 9004:2009（JIS Q 9004:2010）③センス＆レスポンドの SIDA サイクル④BCG アダプティブ戦略の学習プロセス⑤DEOS プロセス 目的・環境変化対応サイクル⑥ITIL®の ITSM モニタ・コントロール・ループを取り上げて比較評価した。

これにより、環境適応モデルでは、環境の監視、環境変化の識別、環境変化対応計画の策定、環境変化対応策の実施、並びに、これらの活動を支える知識が必須であることが共通していることを明らかにした。また、本報告書で取り上げた環境適応モデルでは、これらの活動が再帰的な繰り返し構造が認められることについても明らかにした。

第 4 章情報システム開発における変化適応モデルでは、感知、解釈、決定の観点から、変化適応モデルで必要となる検討項目を具体的に例示した。

また、4. 5 節環境変化適応の事例では、本 WG の委員の皆さんに要求発展についての多様な視点から寄稿していただいた。タイトルを列挙すると、情報システムに求められる柔軟性について（古川 正伸委員）、要求進化（Requirements Evolution）について（斎藤 忍委員）、要求変化とアーキテクチャについて（山本 久好委員）、アジャイル開発について（鷲崎 弘宜委員）、環境の変化とソフトウェアテストについて（秋山 浩一委員）、環境の変化とアシュアランスケースについて（小林 茂憲委員）、IT サービスマネジメントにおける環境変化への対応について（成瀬 泰生委員）、システムエンジニアリングにおける要求変化への対応について（白坂 成功委員）、要求変化対応プロセスの事例について（山本 修一郎主査）となっている。要求発展について、まとめて考察した資料はおそらく初めてであろう。簡潔にまとめられているので、ぜひ参考にしていただきたい。委員によるこの事例紹介は本報告書の大きな特徴である。

第 5 章では、アシュアランスケースによってシステムの要求を主張として定義しておくことにより、要求変化に対するシステムの挙動の影響を客観的に確認する手法について解説している。

第 6 章サービスデザインでは、オープン・サービス・イノベーションのために、サービス構成要素に基づいて、サービス提供プロセスを可視化する方法を紹介している。

第 7 章では、環境変化に対応するシステムに関する課題について、検討する上で重要になる観点として、ソシオーテクニカル、システム連携並びにユーザインタフェースについて述べた。ま

た、環境変化に伴う問題として、要素還元主義、システム運用、モニタリング負荷、既存システムのテストについて考察した。

第 8 章では環境変化への取組みについて、国内の先進企業へのヒアリングを通じて、環境変化への取組み事例を調査した。この結果、環境変化に対応するための実践的な知識を明らかにすることができた。

ここで、本報告書の冒頭で示した WG の狙いについて振り返っておこう。

① ビジネス戦略と情報システム構築の乖離をできるだけ少なくする
② 変化に対応すべき領域を見極め、環境変化に柔軟に対応する
③ 予測できる環境変化を考慮することで、運用や保守の行いやすい情報システム開発を行う
④ 環境変化を把握することで既存システムの改修箇所の妥当性を判断する
⑤ ビジネス環境の変化を考慮したユーザ受け入れテストを設計する

まず、①については、ISO29148 の StRS、SyRS、SRS に基づいてステークホルダ要求からソフトウェア要求への段階的な追跡性を考慮して要求仕様書を構成することができる。②から④については、VSM や SIDA などによって、環境変化を監視、分析、計画、実行、管理を反復的に継続することができる。⑤については運用を考慮して StRS を明確化しておき、要求記述項目を定義することで、ユーザによる受け入れテストを客観的に設計できるようになる。

これらのことから、本報告書で明らかにした技術を活用することで、WG の狙いが達成できることは明らかである。

以上述べたように、本報告書は、情報システムが環境変化に対応する上での課題と対策について包括的かつ具体的に系統立てて説明している。現時点で環境変化対応に対して考え得る最良の知識を収集できたと考える。

この成果を我が国のソフトウェア産業並びにユーザ企業に展開普及するとともに、本検討で明らかになった課題に対する取組みを推進していく必要がある。