ソフトウェア社会の手法基盤と人材基盤

実践的ソフトウェア教育コンソーシアム

WG1 活動報告書 (2011 年 11 月～ 2013 年 3 月)

WG1

2013 March

2013.12.8 発行

序

日本のソフトウェア開発技術者は 100 万人を越えると言われているが、80％近くが IT 企業に属し、ユーザー企業には 20 数万人しか属していない。一方、米国は技術者総数も日本の 3 倍以上だが、その 70％がユーザー企業に属している。かつてのように、開発内容があらかじめ決められる場合は、多くの開発技術者を擁する IT 企業に開発業務の大半を委託しても、何とかシステム開発が進められた。

ところが現在はビジネス環境が激変するから、できるならばユーザー企業自らが開発したいのだが、開発技術者が社内にいないというジレンマに直面している。ソフトウェア開発の人材構造が、ユーザー企業の競争力を蝕みつつある。

とは言っても、この問題は一朝一夕には解決しない。問題は問題と認識しつつ何か打つ手はないかと考えたのが､WG のテーマである「要求分析を重視した分析･設計方法論」である。つまり、ユーザー企業の数少ない開発技術者が業務担当者と連繋して、自分たちのシステムを自らの手で開発できないか、そのための手法やツールにどのようなものがあるのか、である。

WG は、WG1（これまでの方法論や方法の深耕に関する勉強)、WG2（より広い対象分野で適用できる新たな方法論の開発研究、未着手)、WG3（大学等での要求分析講座の開発、P-sec の HP 参照）から構成される。今回は WG1 の報告である。WG1 は野木秀子さん、大槻繁さんのリードで進められたが、多くの有識者の協力で予想以上の成果が得られたと思う。ぜひ詳細をお読みいただきたい。

実は、ソフトウェア開発の諸問題への有力な対策の一つとして考えているのが、様々な技術や経験を有する方々が集う「場」を設定することだ。いつもと同じメンバーが議論しても大して新しいアイデアは出てこない。筆者の前職場であった IPA/SEC も産官学の有識者が集う「場」であったが、今回の WG1 も大いなる創発の「場」であったと思う。

2013 年 5 月吉日

実践的ソフトウェア教育コンソーシアム 会長

鶴保征城

本作品集は、実践的ソフトウェア教育コンソーシアム（P-sec）の WG1（通称：手法基盤 WG）の活動報告書です。各作品の執筆者は、以下の通りです。本作品集の著作権は、各執筆担当者に帰属しています。本報告書の一部、あるいは全てを参照･引用する場合には､「ソフトウェア社会の手法基盤と人材基盤:実践的ソフトウェア教育コンソーシアム WG1 活動報告書 (2011 年 11 月～ 2013 年 3 月)」の該当箇所（章・節番号等）を特定する形式で示してください。その他の利用については、該当箇所の著者にお問い合わせください。

目次

第 1 章　はじめに

1.1　背景と目的

今、低迷する日本に欠けているのはプラグマティズムです [Fujii2012]。学問、技術、経営、あらゆる領域において、自らの活動の社会的な意義や価値を問うことを忘れ、専門性や組織の殻に閉じこもっていては、富を生み出し豊かな社会は実現できません。ソフトウェアの世界も例外ではありません。ソフトウェアの開発者は、自らが開発するソフトウェアがどのように社会で活用されるかを、意識しなくてはなりません。理論、技法、ツールの開発と利用とは一体なのです。個別のものではなく、周囲との関係や全体として捉える視点が必要です。

実践的ソフトウェア教育コンソーシアム（P-sec）の WG1（手法基盤 WG）では、2011 年 11 月から、特定のツールによらないモデリング手法、設計手法、開発方法論や適用事例について、調査・分析を進めてきました。そのために、現在、市場に出ている各種ツールについて、ツール開発会社の担当者からの発表も聞き、多くの事例を得ることができました。

本報告書は、その活動経緯やそれを通して得られた知見をまとめたものです。

2 か月に 1 回の頻度での定例会合に加え、2012 年 11 月に OpenForum という一日をかけた大きなイベントを開催し、広範で多様な情報を入手しました。本報告書は、これ等を再度見直し、編集を加えたもので、報告書作成に当たっては、以下の目標を設定しました。

目標１：手法やツールに関する潮流や全体像を示す

目標２：将来の検討や探求のためのベースになる情報を蓄積・整理する

目標３：報告書作成活動を通じ関係者の方々自身が学び、新たな知見を得る

本報告書の想定読者の第 1 は、ソフトウェアに関わる研究、教育、技術開発、経営に携わっている方々です。個別の技術のみならず、目標 1 に掲げているように全体を俯瞰し、動向や潮流を見極め、メッセージ性のある知見を提示するように心がけています。

想定読者の第 2 は、この WG 活動に参加されたメンバ、そして、今後の活動に参加される予定の方々です。WG 活動というのは、それなりの時間とリソースを割いたもので、一歩一歩蓄積を図りながら次に進めていくものです。目標 2 に掲げているように得られた情報を次のステップのために散逸しないように一つにまとめておくことが求められていると考えています。さらに、WG 活動や報告書作成に知力を割いてくださる方々にとっての一番の意義は、目標 3 に掲げているように、貢献してくださる方々ご自身にとって、それぞれの世界で気付きや新たな知見が得られることだと考えています。

本報告書の構成は、以下の通りです。

第 1 章　はじめに
- 1.1　背景と目的
  - WG 活動、および、報告書作成の背景や目的、位置づけについて説明しています。
- 1.2　WG 活動経緯
  - WG 定例会合、および、イベントの開催経緯をまとめています。

第 2 章　手法の知識体系
- 2.1　手法の分類と系統樹
  - ツールや方法論を整理する方法として、ある時点でいくつかの軸で分類することと、時系列や因果関係で系統樹として整理することというものの見方そのものを解説します。

2.2　ケーススタディ

複数の発表やトピックスを俯瞰して、全体の動向や潮流をいくつかの観点でまとめています。分析するための方法として 2.1 節で述べた方法も活用しています。

第 3 章　開発手法・技法・ツールの各論

WG 定例会合や OpenForum の講演、パネラーの個別の内容と討議状況、見解を個別にまとめています。資料の分量の都合から、本章では概要を示し、詳細を附録 3 に掲載しています。

第 4 章　おわりに

4.1　結論

WG 活動を総括し、得られた知見を分かりやすい形でメッセージとしてまとめています。

4.2　今後の展開

WG の成果をふまえて、次にどういった方向へ進めていくのかを述べています。

編集後記

報告書作成や編集にご協力いただいた方々への謝辞、参加した委員の方々のお名前をまとめています。

参考文献

[Fujii2012] 藤井聡 ,「プラグマティズムの作法：閉塞感を打ち破る思考の習慣」, 技術評論社 , 2012.5.25

OpenForum では開催趣旨の基底となるメッセージとして「プラグマティズム」を掲げました。もともとはチャールス・サンダース・パースが提唱した思想で、開発から利用へ、システムからコンテクストへといった、より全体観があり目的指向の考え方の重要性を説いています。ウィトゲンシュタインや言語ゲームの哲学との関連も分かりやすく解説しています。ソフトウェア、あるいは、ソフトウェア社会の諸活動に関わる哲学を検討していく際の必読書の一つです。

〔野木 , 大槻 著〕

1.2　WG 活動経緯

P-sec が 2010 年 8 月に開催した「要求分析を重視した設計手法と分析・方法論説明会」で、以下に示す WG による検討方針が示されました。

WG1: これまでの方法論や方法の深掘に関する勉強

WG2: より広い対象分野で適用できる新たな方法論の開発研究

WG3: 大学等での要求分析講座の開発

このうち、WG3 は 2010 年 10 月から検討を着手し、2010 年度、2011 年度の検討成果の発表会を、それぞれ、2011 年 7 月、2012 年 4 月に開催しました。

WG1 は、2012 年度から、「ソフトウェア産業に関わる人々が、ソフトウェアの要求分析・設計などの手法の基本を押さえられるように、要求分析手法、設計手法、各種モデリング手法、開発方法論、あるいは、これ等を支援するツールについて概観し、知見をまとめ、世の中に提供（公開）」することを目的に、特定のツールによらないモデリング手法、設計手法、開発方法論について、基本に立ち返って勉強し、さらに、要求分析を重視したツール、実績をあげているツールおよび方法論を中心に適用事例について、議論を進めてきました。

なお、検討に先立って、広く、P-sec 会員から参加者を募り、約 40 名の方から、参加の希望が寄せられ、WG1 登録メンバーとしました。

会合は、WG 1 登録メンバーに案内を送付し、都合のつく方が参加する形で、2012 年 11 月から検討を開始しました。

全体の会合スケジュールを図 1.2-1、各会合の概要を表 1.2-1 に、それぞれ示します。

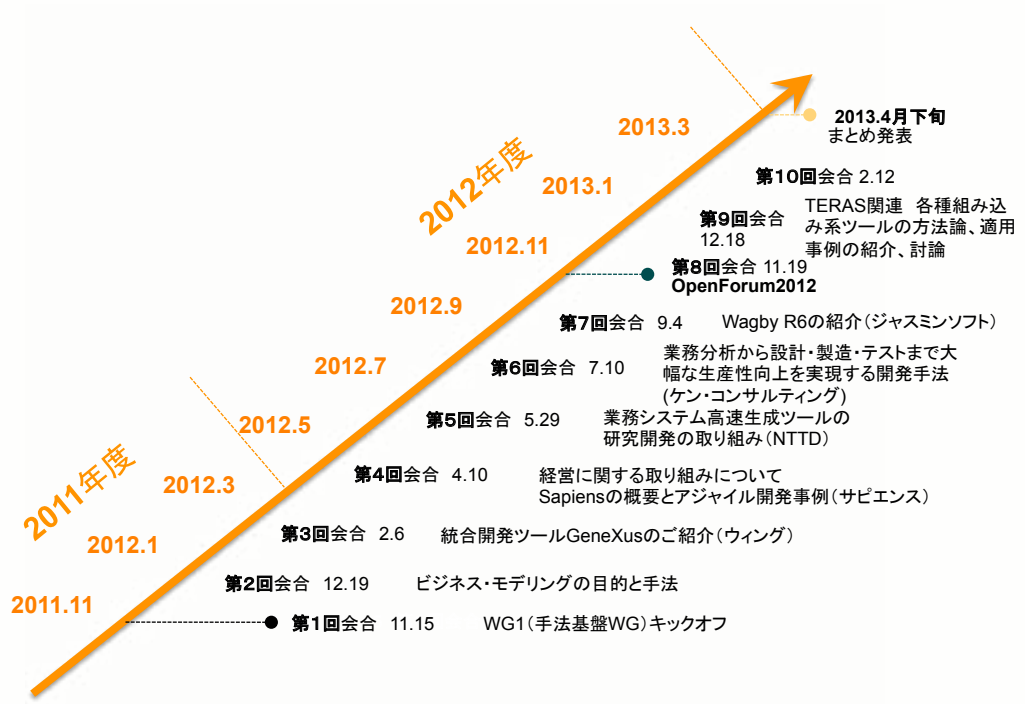


図 1.2-1　WG1 の検討スケジュール

表 1.2-1　各会合の概要

回	開催日	主な内容 標題：話題提供者（所属）	場所	出席者数	備考
1	2011.11.15	全体計画と進め方：大槻　繁（一（いち））	サイバー創研 （渋谷）	17	
2	2011.12.19	ビジネス・モデリングの目的とその手法：大島正善（MBC） 開発プロセスSTRAP＆GeneXus の概要： 樋山証一（ウイング）	サイバー創研 （渋谷）	17	
3	2012.2.6	BMAPROS：福島祐子（日本ユニシス） 統合開発ツールGeneXusのご紹介：山内啓悦（ウイング）	サイバー創研 （渋谷）	22	
4	2012.4.10	ビジネス貢献を含んだソフトウェア開発： 黒川利明（金沢工業大学） Sapiensの概要とアジャイル開発事例： 大北達彦（サピエンス・ジャパン）	日本ユニシス （新木場）	19	
5	2012.5.29	業務システム高速生成ツールの研究開発の取り組み： 斉藤忍（NTTデータ）	王子物流 （銀座）	23	
6	2012.7.10	ルール定義でシステムが自動的にできるBRMS： 片貝孝夫（片貝システム研究所） 業務分析から設計・製造・テストまで大幅な生産性向上を 実現する開発手法：高橋俊夫（ケン・コンサルティング）	王子物流 （銀座）	17	
7	2012.9.4	Wagby R6の紹介：贊良則（ジャスミンソフト） Agile can be really agile?：宮崎比呂志（富士通）	富士通 （新橋）	15	
8	2012.11.19	基調講演1 要求工学知識体系 顧客価値創造からの展望： 山本修一郎（名古屋大学）） パネル1 ツールのための開発方法論： モデレータ　黒川利明（金沢工業大学） 基調講演2 人間学アプローチからみた要求方法論： 羽生田栄一（豆蔵） パネル2 一生役立つ情報システム技術の叡智： モデレータ　大島正善（MBC）	専修大学 （神保町）	82	P-sec Open Forum 2012
9	2012.12.18	オープントレーサビリティプラットフォームTERAS： 渡辺政彦（CATS）	CIJ （横浜）	13	
10	2013.2.12	WG1成果報告書の執筆進捗状況：大槻　繁（一（いち）） ワークショップ（情報システム化第二期、キャリア）： 三輪一郎（プライド））	サイバー創研 （渋谷）	15	

第 1 回（2011.11.15）は、鶴保 P-sec 会長から、P-sec における WG1 発足の経緯の紹介のあと、出席メンバーの自己紹介を行い、野木リーダ、大槻サブリーダから、WG1 での全体計画と進め方（スケジュールについては図 1.2-1 参照）の説明がありました。

第 2 回（2011.12.19）から第 7 回（2012.9.4）までは、表 1.2-1 の「主な内容」欄に示すような表題で、上記 WG1 の目的に沿う内容の発表と質疑を行いました。

第 8 回（2012.11.19）は、検討会の中間成果の報告会も兼ねて、P-sec Open Forum 2012 を開催しました。この Open Forum 開催のためには、WG1 メンバーから有志を募り、タスクフォースを組織し、プログラムの検討から、当日の会場設営まで、献身的に実行し、Open Forum を成功裏に終えることができました。

第 8 回までは、主にエンタープライズ系の内容でしたので、第 9 回は組み込み系の話題を取り上げました。

第 10 回（2013.2.12）は会合としての最終回で、2013 年 3 月末までにまとめる WG1 の成果報告書の進捗について議論したほか、次年度以降の活動に対する課題提起を兼ねて、情報システム化第二期（IT の利用を前提に業務が進化する時代：IT と業務の進化が拮抗し、同時に進化していく時代）での課題とお勧めキャリアについて、ブレーンストーミング形式によるワークショップを行いました。

会合場所については、渋谷のサイバー創研のほか、メンバーの中から、新木場の日本ユニシス、銀座の王子物流、新橋の富士通、横浜の CIJ、それぞれの会議室を利用させていただきました。また、第 8 回の Open Forum では、専修大学・神田キャンパスの大きな教室を利用させていただきました。

〔田島 著〕

第 2 章　手法の知識体系

2.1　手法の分類と系統樹

（1）分類と系統樹

複数の手法を俯瞰し、全体を見通すためには、何らかの分類をするのが効果的です。手法を利用する実践的な場面では、複数の手法から適切なものを選択しなくてはなりませんが、その折に、分類による整理があると選択基準として一つの拠り所になるでしょう。分類の軸の定義や、その上への位置づけは、主観的で恣意的なこともありますが、その時々の認識をマップとして切り出していると言えます。

一方、手法は日進月歩で進化、あるいは、変貌を遂げていきます。どのような手法も、他の手法を参考にしたり発展させたりしています。系統樹的な観点を導入すれば、手法そのものの理解が深まりますし、未来の方向性も見極めることがやりやすくなります。系統樹的な観点の中には、家系図のような素朴で直接的な関係がありますが、技術進化の観点からはもう少々複雑な因果関係で繋がっていることもあるでしょう。

上記の「分類」と「系統樹」を相補的に組み合わせることにより、より強力な分析ができます。これを「マンダラ分析法」と呼ぶことにします。「マンダラ（曼荼羅）」が仏教において諸行無常な因縁の世界を表していることに依っています。ちなみに、かつて EA（Enterprise Architecture）策定の方法として Mandara Matrix というのがありましたが、全く関係ありません。

もともとこの分類と系統樹の考え方は、近年の系統情報学として発展してきています [Minaka2010]。ダーウィンの進化論、生物の博物学に源流をたどることができます。ここでは、生物進化ではなく、人工物（アーティファクト）や手法・ツールの体系化に適用してみようという試みです。

（2）分類

分類するということは、簡単に言えば、対象物を要素として、それをいくつかの集合に属すように分けるということです。このものの見方は、以下のような学問や考え方として発展してきました。

分類科学（classificatory sciences）

> 体系動物学、体系植物学、比較解剖学などの対象物の類似度（degrees of likeness）を研究する学問

分類思考

> ある時空平面で歴史を切断した時に見える「断面」の分類パターン認知

メタファー（隠喩：metaphor）

> 類似（similarite）による体系化。目に見える表面的な類似性によるグループ分け。類型学（typology）と密接な関係がある。

認知カテゴリー化

> 何らかの基準で互いに似ている対象物を一つのグループにまとめる心的能力

分類を、ある集合の要素か否かという二値ではなく、ファジー（曖昧）集合にしたり、測度（程度）を導入すると分かりやすい体系を示すことができることもあります。ソフトウェアの開発手法を整理する場合の例として、以下のような軸の候補が考えられます。

静的 (Static) ←→動的 (Dynamic)

閉じている (Closed) ←→開いている (Open)

形式的 (Formal) ←→非形式的 (Informal)

技術的 (Technological) ←→人文的 (Humanistic)

個別領域的 (Domain Specific) ←→汎用的 (Generic)

データ指向 (Data Oriented) ←→プロセス指向 (Process Oriented)

２軸でマップを描くと、より視覚的に全体像を俯瞰することができます。要求工学の技術を描いたものは、以下のようになります [Ohnishi2009]。

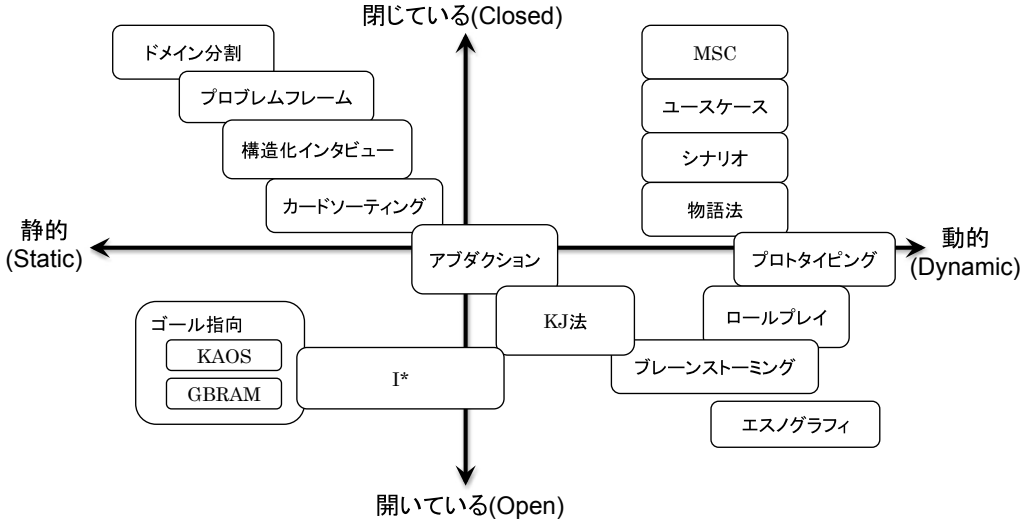


図 2.1-1　要求工学技術の分類

（3）系統樹

系統樹分析の基本は、時系列や因果関係で対象を見るということです。このものの見方は、以下のような学問や考え方として発展してきました。

古因科学（palaetiological sciences）

動植物の地理的分布、地質学、語源学、民俗学などの歴史的因果（historical causation）を研究する学問。現在得られる証拠に基づいて過去の状態に遡る研究

系統樹思考

「断面」で得られたデータに基づいて系統関係を推定する作業

メトニミー（換喩：metonymy）

血縁（genealogie）による体系化。部分を示すことで全体（隠された根源的なもの）を表現する文彩（修辞法：レトリック）

比較法（アブダクション：abduction）

断片的知見を集積して一貫した全体的ストーリーを復元する思考様式をメトニミー的思考といい、この全体的ストーリーをアブダクションによる最良の仮説の推論と呼ぶ。

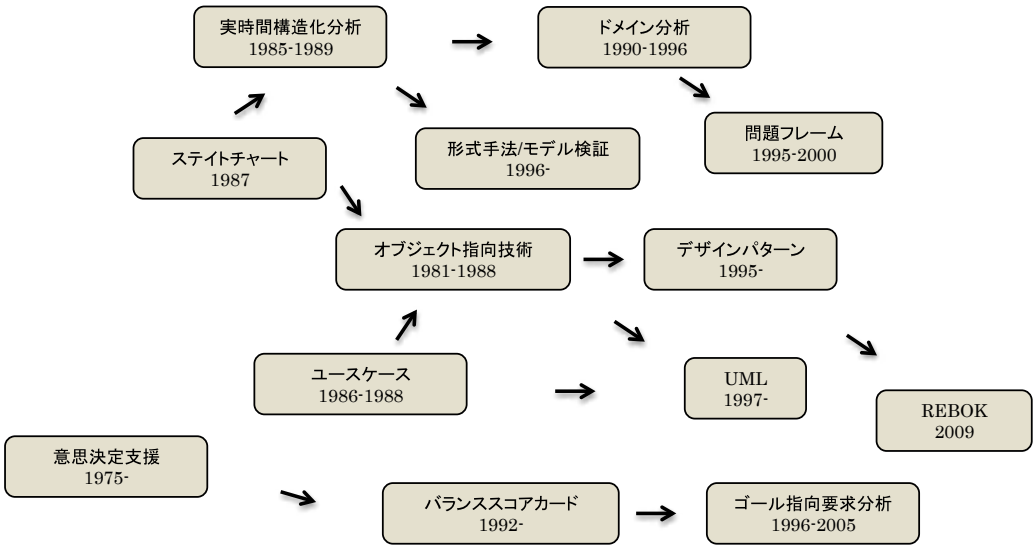


図 2.1-2　手法の系統樹

最も簡便な時系列記述は年表です。ある手法や技術がいつごろ開発されたか、あるいは、発表・公開されたかといった観点で年表を作ると、それだけで一つの全体像が浮かび上がってきます。個別の対象は、何らかの因果関係で他の対象と結びつけることができます。手法やツールといった複合的な対象の場合には、因果関係を見出すのは難しいこともあるでしょう。間接的な影響、発展・進化、拡張、複数の手法の統合、標準化、自動化などいろいろなストーリーを見出すことができれば、全体像に説得力が増すことになるでしょう。

ソフトウェアエンジニアリングの一部の手法を系統樹として描いた事例を図 2.1-2 に示します。

（4）マンダラ分析法

上記で解説した「分類」と「系統樹」によって、主観的かもしれないけれども世界観をまとめておくのは、自ら判断し、選択し、技術開発をしていくために価値があることです。ここでは、簡便な分かりやすいマンダラ分析法を作業仮説として提示しておくことにします。

Step1　関心テーマを設定する

要求工学、ソフトウェア開発手法、自動化ツールといったそれぞれの分析者の関心事をいくつかのキーワードを掲げて設定します。

例えば、ここでは、要求工学の分野で近年周辺の考慮事項として注目を浴びてきている「ビジネスアナリシス (BA:Business Analysis)」を採り上げてみましょう。ロナルド・ロスの著作『ビジネス・アナリシス』[Ross2011] を読むと、その内容からシステム要求へ接続するビジネス知識体系（BABOK）のビジネス側を中心にしたものであることがわかります。また、BA 自身の多くがザックマン・アーキテクチャを下敷きにしていることも読み取ることができます。

Step2　年表を作成する

大まかに年の時間軸を設定して年表を作成します。厳密には、ある手法が水面下で開発された時、検討を開始した時、公開された時などいくつかの設定がありますが、粗いレンジで捉える程度でよいでしょう。この年表を作成する理由は、関心テーマの周辺のトピックを洗い出したり、時代の趨勢を大まかに把握するためです。次ステップでトピック間の因果関係を見極める時に、未来は過去に影響を与えないという制約条件を意識することにも有効です。

Step3　因果関係を把握する

Step2 で得られた年表上のトピックス間の因果関係を把握し、その関係を矢印等（複数の種類を設定してもよい）で示します。因果関係の分析には、トピックの内容を調べて、共通的な因子を発見したり、開発者の人物の背景や経歴などを参考にします。

BA の場合には、その手法の中にザックマン・アーキテクチャを一部取り入れているところがあります。また、システム要求に展開していく場合に、BABOK の思想的背景として活用されているのも明らかです。周辺の事情含めて図 2.1-2 と融合させて、系統樹を描くと図 2.1-3 のようになります。

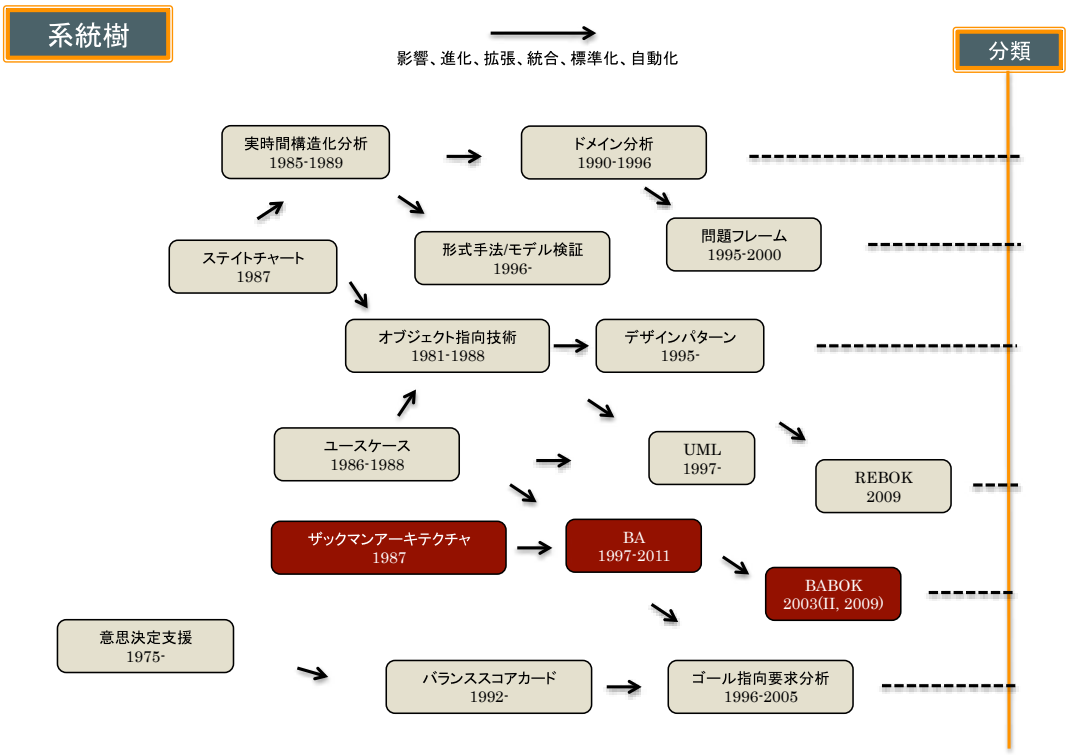


図 2.1-3　BA を関心テーマとした系統樹

Step4　分類対象トピックを抽出する

Step3 で得られた系統樹から、分類の対象とするトピックを抽出します。分類は、系統樹を時系列と見なした場合の現在で利用されているもののリストに相当します。

図 2.1-3 の図中で右側に破線で伸びている項目が分類の対象となる第一候補です。その他、過去のものでも現在も利用されているものは分類対象としてもかまいません。

Step5　分類マップを作成する

Step4 で得られた分類対象をカテゴリー化し分類します。分類の観点は、まさに直感による世界かもしれません。代表的な分類軸は、以前にも掲げた以下の観点も候補の一つです。

・静的 (Static) ←→動的 (Dynamic)

・閉じている (Closed) ←→開いている (Open)

・形式的 (Formal) ←→非形式的 (Informal)

・技術的 (Technological) ←→人文的 (Humanistic)

・個別領域的 (Domain Specific) ←→汎用的 (Generic)

・データ指向 (Data Oriented) ←→プロセス指向 (Process Oriented)

その他、その時々のこのマンダラ分析を行う目的に沿ったものを設定することもあるでしょう。

既に、分類した体系がある場合には、その軸上に抽出された分類対象を重ねてプロットしてもよいかもしれません。図 2.1-1 の分類例の上に BA 関連のものを重ねてみると図 2.1-4 のようになります。

分類

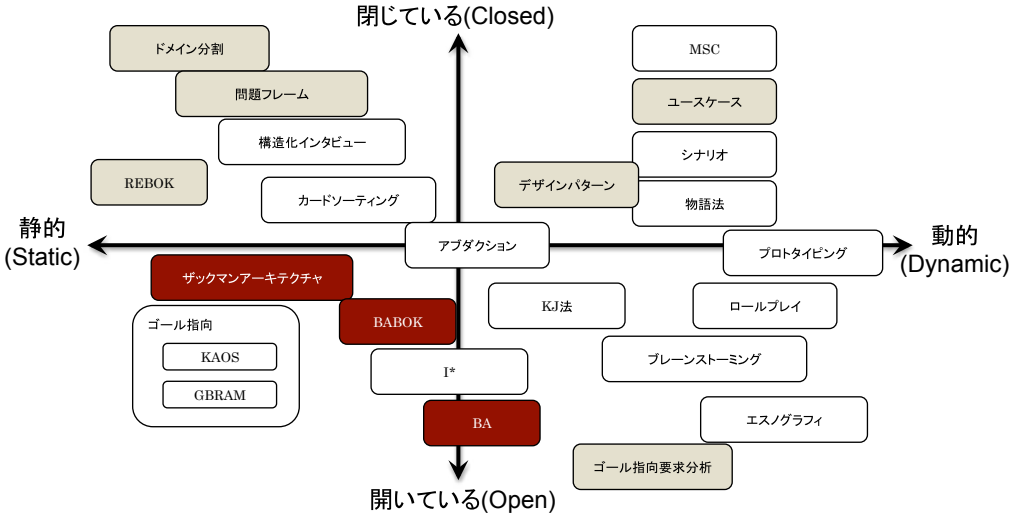


図 2.1-4　BA を関心テーマとした分類

上記の Step1 ～ 5 は、逐次的な手法として提示しています。実際には、何度も行き来して完成度を上げていくことになります。そして、分類や系統樹といった具体的な表現にすることによって、多くの人々の間でコミュニケーションを行うことができ、主観的なものから集合知へと成熟していくことでしょう。

参考文献

[Minaka2009] 三中信宏 ,「進化思考の世界」, NHK Books, 2010.9.25

著者の三中教授は、進化生物学や進化統計学が専門で啓発的著作も数多く著している人です。「進化思考」というのは、分類や系統分析の方法を提供する体系学です。生物、言語、音楽などの世にたくさんあふれているものを体系的に整理して示す方法と思えばよいでしょう。

[Ohnishi2009] 妻木俊彦 , 白銀純子（大西淳監修）,「要求工学概論：要求工学の基本概念から応用まで」, 近代科学社 トップエスイー基礎講座 2, 近代科学社 , 2009.9.30

要求工学の３つのアプローチ：ビジネス指向、コミュニケーション指向、ドメイン指向を観点として設定して調査を進めています。こういった要求や手法の全体像をつかむのに好適な書籍です。

[Ross2011] Ronald G. Ross and Gladys S.W. Lam, “Building Business Solutions: Business Analysis with Business Rules”, Business Rule Solutions Inc, October 15, 2011

（「IT エンジニアのためのビジネスアナリシス」, 日経 BP, 2011.11.26）

ビジネスアナリシスの分野を牽引してきた著者が、ビジネスルールやビジネスアナリシスの基本概念を丁寧に解説しています。システムとは切り離した領域でビジネスについて分析する方法として有効だと言われています。

〔大槻 著〕

2.2　ケーススタディ

2.2.1　ツール

（1）開発ツールの歴史と系統樹

ツールの歴史は、それを使う人の環境、それを実現する技術などといったエコシステムと切り離せません。ソフトウェア開発における顕著な特徴は、コンピュータ（すなわち、ソフトウェアそのもの）が、ソフトウェア開発におけるツールとして用いられてきたということです。これは、コンピュータのハードウェア開発においても同様であり、ツールと成果物そのものが、再帰的に関わるところが、ソフトウェア開発の重要なところです。それは、ソフトウェア開発ツールの歴史が、ソフトウェア・システムの歴史と一体になっているということにほかなりません。

概要の図を下に示しますが、ソフトウェア開発の黎明期は、コンピュータ開発の黎明期に重なりますが、最初のツール

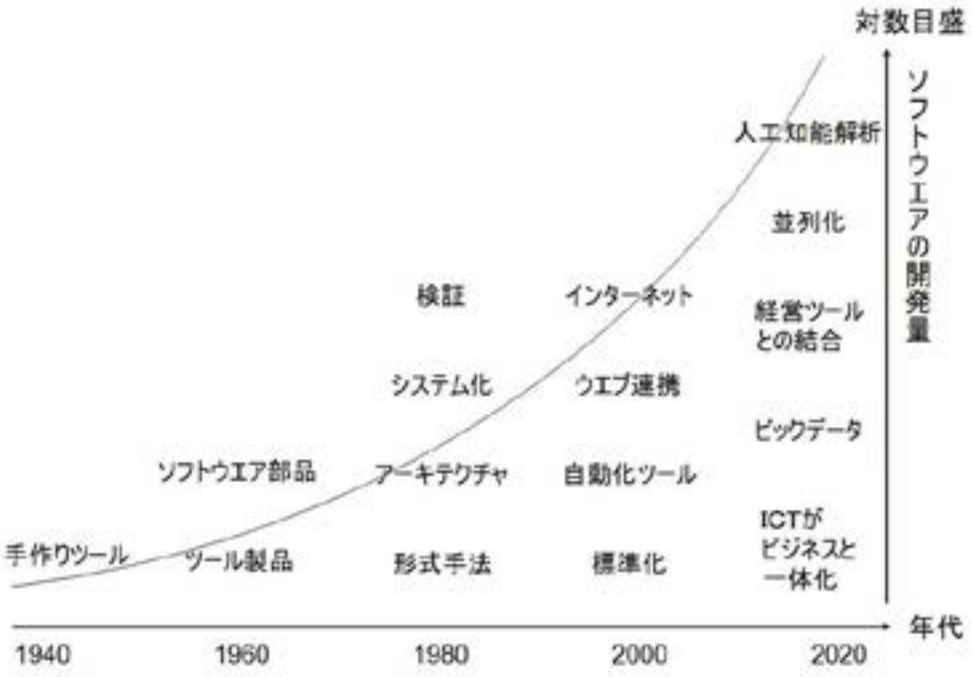


図 2.2.1-1 開発ツールの歴史

イニシャルローダー、インタープリター、コンパイラなどといった基本的なツール（現在では、コンパイラをソフトウェア開発ツールなどとは、思わないかもしれませんが）、が整備されると同時に、シンボリック・デバッガなどのツールが 1960 年代には、登場し、初期のソフトウェア・ツールが製品化され、流通するようになりました。

この時代の個別のソフトウェア開発の時代は、やがて、システムとしてのソフトウェア開発の時代に取って代わられます。70 年台から始まったこの時代のキーワードの一つは、アーキテクチャでした。個別のソフトウェア開発だけでは、全体システムの開発管理はできません。そこで、ツールにおいても、アーキテクチャが重要になりました。この時代のそういった動向の典型例は、IBM が当時推進した AD/Cycle に見られます。

この 80 年代のシステム化の時代には、コンピュータサイエンスの成果を、ソフトウェア開発に取り入れる活動も盛んになり、オブジェクト指向が、ユーザインタフェースに絡んで、研究室から開発現場へと導入され、数学に基づいた形式手法も盛んになりました。

2000 年代は、コンピュータ・システムが、メインフレーム、ワークステーション、PC といったハードウェア基盤から、ネット上へと変化し始めた時代である。Web 連携が、ソフトウェア開発での主要事項となり、ソフトウェ

ア開発そのものも、ネット上への移行が始まりました。

ソフトウェア開発においては、その前のシステム化時代に、研究開発されていた自動化ツールが、実用化されるようになりました。インターオペラビリティーが重要な要素となり、オブジェクト指向言語だけでなく、設計用の言語である UML の国際標準化が行われるようになりました。

現在は、情報通信システムが企業経営と密接に関わるようになった、ICT-Convergence の時代に突入しており、経営現場とコンピュータ・システムとが一体となって運営することが求められるようになっています。ソフトウェア開発ツールも、経営ツールと一体となって、開発運用に寄与することが求められる時代となってきています。

技術的には、昨今話題のビッグデータを活用するデータ・サイエンス、さらに、音声理解など、人工知能技術の活用が目前まで来ていますが、ソフトウェア開発の量と質は、今後共飛躍的に増大しますので、開発ツールについても、より一層の飛躍が求められるところです。

（2）開発ツールの活用前提

開発ツールとは、広義にはソフトウェア開発を支援するソフトウェアであるが、ここではできるだけコードを書かず、ソフトウェアを構築していく高速開発ツールを対象に論じています。

開発ツールを活用し、大きなメリットを生むために重要な前提条件を考えてみます。

1) 開発ツールに適した開発方法論の採用

開発ツールによるソースコード自動生成や繰り返し開発が可能となるため、アジャイル開発やプロトタイピング開発などにより、システム要求を迅速に具現化、確認して変更要望をとりこむことが可能となります。従来のウォータフォール型開発方法における製造工程だけの開発ツール適用では、システム全体の多大な生産性向上や品質確保を図に結び付けることは難しいです。

2) ユーザ要求とツール機能を考慮できる設計能力

システムの要求とツールの標準機能を結びつける設計能力及びユーザへの提案能力を持ったシステムエンジニアがプロジェクトの初期段階から関わっていることが重要です。

システム要求を ①ツールの標準機能で実現できること ②カスタマイズによって実現できること ③ツールベンダーに相談することの判別ができます。また、予算や工期、工数についての見通しを立てられるとともにツールの能力を活かしやすくなります。

3) 開発ツールのサポート体制整備

自動開発ツールゆえに意図したプログラムの動きにならない現象が発生した時にツールメーカーもしくは代理店による迅速なサポート対応が欠かせません。開発ノウハウや技術が蓄積されるまで、外部サポートに支援していただくことが効果的になります。

また、開発ツールの基本的な特徴や機能、操作方法をトレーニングしたうえで活用することが望ましいです。

4) 複数人の開発基盤とセキュリティ

大規模なシステム開発にも、複数人による同時開発に対応できることや生成されたソースコードなどのセキュリティなどについても活用の前提条件として調べておく必要があります。

（3）開発ツールの限界

高生産性をはかる開発ツールにおいての限界と対策を考えます。

1) ツールの標準機能でユーザ要望を実現する限界

ユーザインタフェースや操作性などを自社要求に合わせたいときにツールの標準機能では開発できないことが生じます。その限界を超えるための方法について述べます。

この限界を乗り越えるための方法はいくつかあります。一つはツールベンダーを巻き込み、要求に合わせてツールそのものを改修できるような体制を敷くことです。この方法はツールベンダーを自社システム開発の「チーム」の一員に組み込むことになるため、ツールベンダー側がそのような協力に応じられるかどうかが鍵になります。もう一つは、複数のツールを組み合わせることです。画面設計に特化したツール、帳票設計に特化したツール、バッチ処理に特化したツール、業務ロジック作成に特化したツール、そしてこれらを統合できるツールを採用します。一つの完璧なツールを求めるのではなく、それぞれが得意とする領域の組み合わせでツールの限界を乗り越えるというアイデアです。

また、開発ツールによっては、細かいレベルで記述できるものもありますが、目的とする高生産性が減じる（とはいえスクラッチ開発よりは高生産性です）ことになりますので考慮が必要です。

2) 将来の技術への対応

開発されたシステムもしくはシステムを生成する入力情報が将来にわたり主要なプラットホームやデータベースに対応できるかどうかが問題です。開発ツールの稼働環境が無くなっても生成されたシステムが修正できる方法があれば望ましいです。

（4）開発ツールと開発手法

開発対象のシステムを「構造」と「振る舞い」という視点で整理します。その中心となるのは「DOA (Data Oriented Approach) 」であり、プログラムとデータは独立するという概念です。データの「構造」を整理し、これを「モデル」として表現します。次に、モデルの生成を時間軸で整理します。例えば「見積書」から「売上伝票」がつくられ、さらに「請求書」がつくられる、という業務であれば、自ずと各モデルの生成に順序があることがわかります。その生成過程において見つかるルールを「振る舞い」として抽出します。例えば「見積書」の「見積番号」を「売上伝票」に転記するというようなものが振る舞いの一種です。このように抽出された設計情報をツールに入力することで、システムの雛形がつくられます。従来、「振る舞い」はプログラムによって表現されるものでしたが、パターン化された振る舞いはツールによって自動生成されるようになっているため、開発が不要となっています。このように、ツールを使った開発では設計の比重が高まります。

開発手法ありきで、開発手法を実現するためのツールであるべきであり、ツールを有効活用するためにはそのベースとなっている開発手法を理解することが必須です。

業務アプリケーションの開発の場合は、DOA の開発手法をベースとすべきであり、データモデルはシステムの土台や骨格となるので、業務モデルを的確に表現できるし、ビジネスが変化してもデータは大きな影響を受けることは少なくなります。

（5）スクラッチ開発との違い

開発ツールを利用する場合は「なるべくソースコードを書かないように心がける」という意識を持ちます。ソースコード開発に代わって、設計工程に時間をかけます。そのためプロジェクトに、ツールの使い方に精通したメンバーを加えることが求められています。これにより、システムへの要求仕様をツールのどの機能で実装するかを判断するという過程を（開発プロセス中に）組み込みます。その意図は、「できるだけ手組みしない」方針で臨むことです。可能な限りツールの標準機能を使い、手組み（従来型のプログラム開発）のボリュームを減らすことで、高速開発を目指します。

ツールを使った場合でも、ウォータフォールで開発するという考え方もあります。ただ、工期は圧縮するように意識します。例えば従来方式で 1 年と見積もられたプロジェクトを、4 か月で達成させる、というような目標を立てます。このときのお薦めは、これを 3 回繰り返すというものです。4 か月の 3 サイクルだと結局 1 年ということになるのですが、2 回のブラッシュアップの機会がある、というのがポイントです。1 回目のサイクルを通して、はじめて気付いた改善点というのが必ずあります。ツールの利点は、仕様の再設計が行いやすいということです。プログラムの多くは自動生成されたものであるため､捨てることにためらいが生じません｡そのためウォータフォール型開発であるにもかかわらず､「変化を受け入れる」というアジャイル型開発の考え方を導入できるようになります。

（6）運用保守のメリット

クラウドが注目されていますが、それと同時に、もしクラウド環境が停止した場合の業務継続性も重視されています。つまり、システムはクラウドでも自社環境でも、どちらでも動作するように構築することがベストです。両方をサポートするツールを採用することで、開発者はこの負担から解放されます。すなわち運用保守の可用性と安定性も、ツールが担保する領域になっています。

運用保守の工程でも開発ツールのメリットが大きいと言えます。運用保守の生産性向上による保守費用の削減や経営戦略の変更を実行するまでの期間短縮がはかられることになります。

（7）経営に与える影響

外部環境の変化が激しく、それに対応するために経営戦略を練り直し、業務のやり方が日々、変化するスピードが企業の業績に大きな影響を与えます。よって、それを支えるシステム基盤の変更容易性がスピード経営に欠かせません。プログラムを書かない開発という、新しい体制に移行することで、このスピード感を達成しようというのが、これらのツールが登場した背景にあります。

従来のスクラッチ開発に比して、システム開発時の期間短縮、繰り返し開発によるユーザの満足性向上、システム運用保守の迅速性はスピード経営を支え、ビジネスの変化に柔軟に対応できるため、ビジネスチャンスを有効に活用できることにつながります。

これからもシステム開発には、開発ツールの活用が広がり、もっと高生産性や容易に使えるツールが出現してくると思われます。

〔樋山 著〕
〔西住 図制作〕

2.2.2　情報システム技術

（1）情報システム技術が担う背景

情報システムが生まれて 60 年が過ぎましたが、人類の歴史の長さに比べれば、60 年はほんの一瞬の出来事に過ぎません。しかし情報システムは社会に大変な勢いで浸透し、私たちはいまや情報システムと共に生活を過ごしていると言っても過言ではありません。これほどまでに爆発的に浸透した情報システムは産業構造に革命をもたらしました。その発展の行方には予想もつかない展開が待ち受けていることと思われます。

新しい仕組みが次々と生まれるこの技術には、次の社会を構築するためのヒントが隠されている予感を私達に与えています。情報システム技術は、経済社会の効率化要求を満たすだけに留まらず、私達の感性に直接訴えかけるエモーショナルな部分でも重要な役割を担っています。ネットワークが発達し、コミュニケーションが容易になることで、人との関係性を構築する社会基盤としても必要不可欠なものとなっています。

（2）情報システム技術の歴史

今後、登場する技術を推測するためにも、これまでに登場した情報システム技術を列挙します。

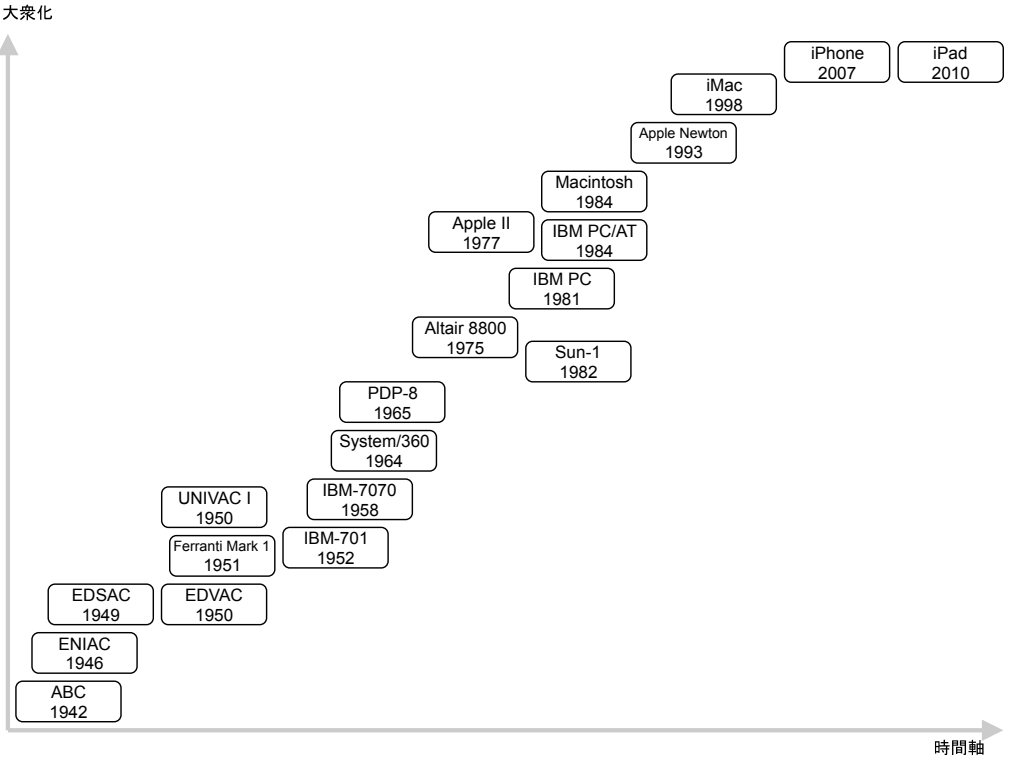


図 2.2.2-1　情報システム（ハードウェア）の歴史

真空管を使って弾道を計算する機械。これが最初のコンピュータです。コンピュータの技術の多くは、軍事技術から転用されているものが多く、人類の争いの歴史の中から生み出された副産物といっても良いかもしれません。

その後、小型化、高速化が進められ、大量のデータを扱えるように進化し、ネットワーク上で高速且つ正確に情報を共有する役割を担った時点で、社会インフラとして急成長を遂げることとなります。

当初は原始的な命令しか認識できなかったものの、近年は高度な命令を認識できるように変化し、命令言語もより人間社会を抽象化したオブジェクトとして認識できるように発達しています。より多くの人が関われるように、分かりやすさを求めた改良が続けられています。

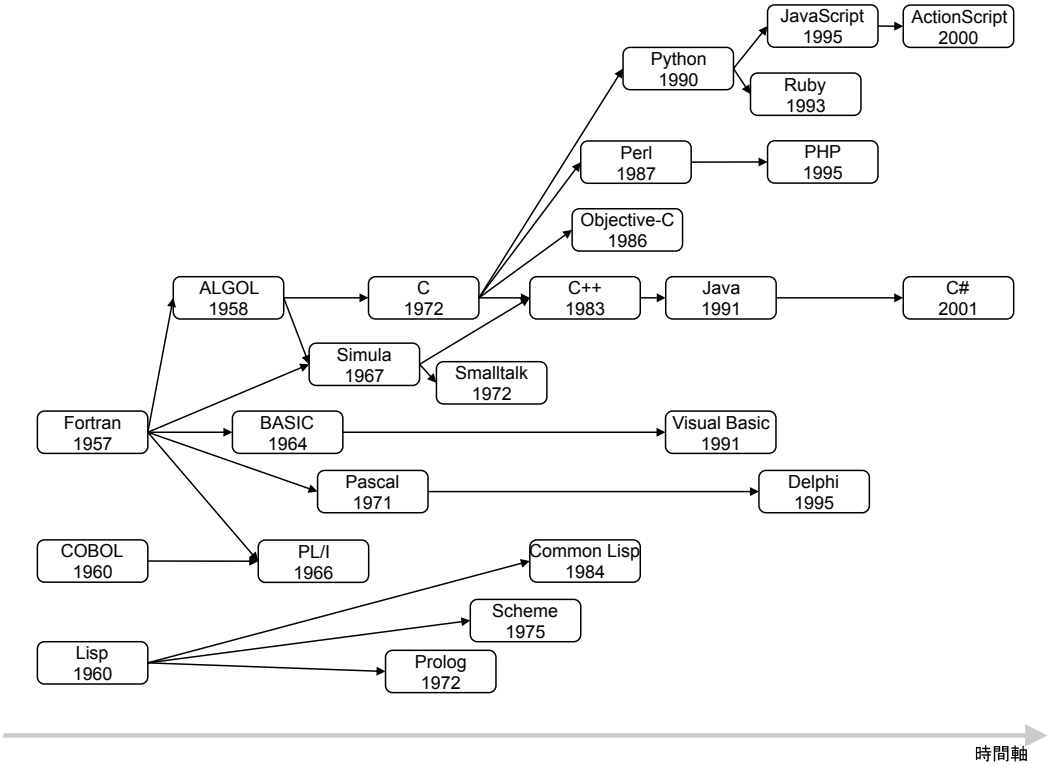


図 2.2.2- 2　プログラム言語の推移

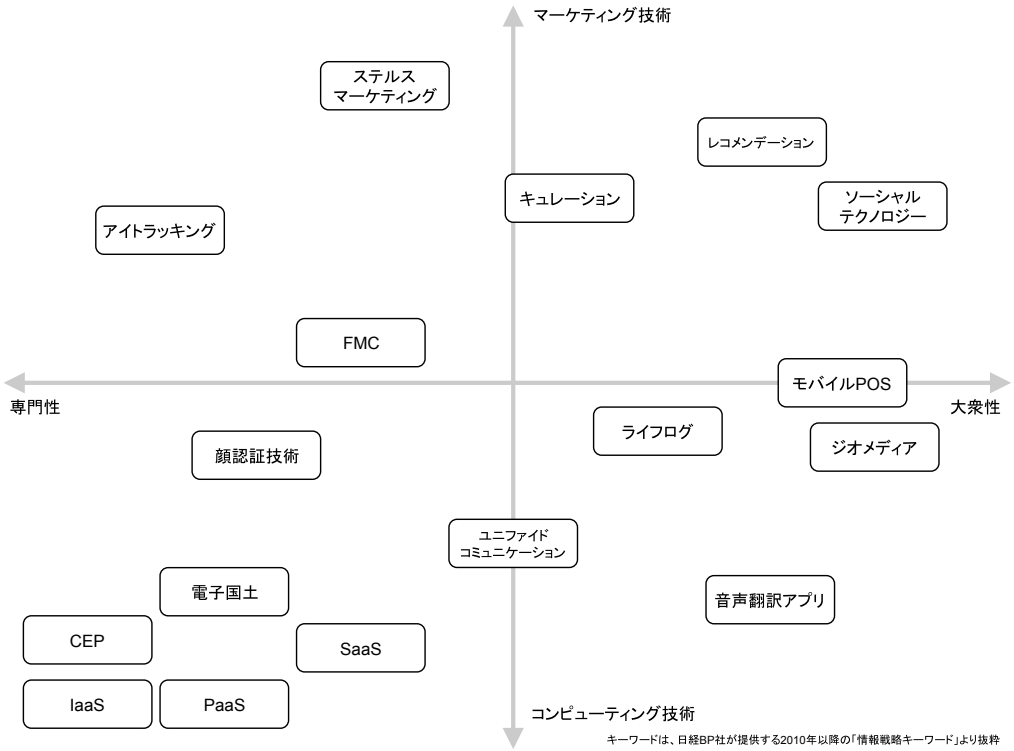


図 2.2.2-3　情報戦略キーワードの分布

コンピュータはいずれ、自己組織化して自立進化できるソフトウェアとして、仮想空間上で発達する可能性すらあります。生命の進化の仕組みをデジタル世界の中で再現し、爆発的に進化していく可能性をも秘めています。

（3）教育の充実化に向けて

システム開発は人が関わる知的な作業です。この知的作業に高い生産性を求めるのであれば、効率的な管理を考えるよりも、やりたいと思える環境をどのようにして作り出すかを考えるほうが、より根源的な課題に向き合っていると言えます。

情報システム技術は多様化かつ複雑化し、それを扱う人々の技量の向上が求められています。近年のシステム開発では、大規模な開発は徐々に減り、小規模な開発が増えていますが、このような時代、組織による効率的なシステム開発の管理手法よりも、個々人の持つ能力を尊重するアジャイル的なシステム開発の手法が生産性を高める鍵となります。個々人の持つ能力を適切に発揮するためにも、情報システム技術の教育がますます重要となります。

人は「勉強していて楽しい」と思える教育環境に入らなければ、主体的に学習意欲が伸びることはありません。基礎教育とその応用により得られる感動を繰返し、知的活動に対する好奇心のスパイラルを築くことで、生産性も高まりイノベーションが生み出され新たな社会雇用も生まれる。そのような循環を築くことが教育の目的・使命ではないかと思うのです。

情報システム技術は時代と共に変化してきました。これからも社会のニーズに合わせて変化していくものと思われます。これらの変化に対応するためには、そもそも業界に興味を持ち、変化を楽しめる素養を持つ人材を多く育成しなければなりません。指導型の教育カリキュラムだけでは変化に対応する柔軟性を身につけることは困難です。

豊かな社会へと活性化させるためにも、人の行動を科学的に分析して、

① 社会的・文化的，技術的ツールを相互作用的に活用する力

② 多様な社会グループにおける人間関係形成能力

③ 自立的に行動する能力

これらの力を獲得できる環境づくり、枠組みづくりが教育には求められています。

（4）これからの情報システム技術

情報システム技術は、人と人を繋いで社会を再構成する力を持つ技術だと言えます。

個人が社会の一員として、より自然な形で再構成され、旧来の組織構造とは異なる「新しい意識体を持った社会」として再出発できる時代になったのだと思います。

参考文献

OECD、『The Definition and Selection of KEY COMPETENCIES』2003 年
安富、『生きるための論語』、2012 年 4 月
平田、『欧米で普及するアジャイル開発』、IT ソリューションフロンティア 2010 年 11 月

（上原 著）

2.2.3　手法知識体系

本項では、P-sec オープンフォーラムにおける名古屋大学教授　山本修一郎先生のご講演「要求工学知識体系」で紹介された要求工学手法を、マンダラ手法により整理しました。なお、ご講演の内容は 3.10 節に記載していますので、合わせてご覧ください。

（1）山本先生ご講演から

要求工学の歴史を振り返ると、1990 年代にはプロセス、2000 年代にはシステムを作る目的が重視されてきました。最近では顧客価値創造のための「ビジネスと IT の整合化」、要求を正しく理解するための「運用時オペレーションの理解」、要求は完全には定義できないことを前提とした「不確実性への対応」、「製品品質の説明責任」が求められてきています。その背景としては、外部・内部の環境の変化やリスクの露呈などにより要求が変化する、また、変化のスピードが一昔前と比較するとより速くなっていることが挙げられます。

こうしたことから、要求工学を取り巻く手法にも以下のような変化が見られます。

① 要求－開発－運用の統合化
② ビジネスアナリシスへの重点シフト
③ 品質要求の満足度を証明するための手法研究

（2）手法体系の整理

要求工学手法を、まずはマンダラ分析法の「系統樹」で整理し、各手法がどのように発展してきたのかを時系列に概観します。次に時系列に表した手法を「分類」で整理し、それぞれの手法の位置づけを明確にすることにより傾向を見極め、今後の方向性の予想を試みることにします。

1) 系統樹の整理

主要な手法としては ISO/IEC/IEEE の標準規格、xxBOK の知識体系、TOGAF 等の EA 系があります（図 2.2.3-1 参照）。

標準規格では、ライフサイクルにフォーカスした ISO/IEC 12207（Software life cycle processes）、ISO/IEC 15288（System life cycle processes）における要求工学のプロセスと要求仕様の標準規格である IEEE 830（Software Requirements Specifications）[1]、IEEE 1233（System Requirements Specifications）が、要求工学の標準規格である ISO/IEC /IEEE29148[2] に引き継がれています。

EA 系 [3] を見ていくと、ザックマンフレームワークに端を発し、米国政府内では FEAF（The Federal Enterprise Architecture Framework）、DoDaF（Department of Defense Architecture Framework）に分かれ、民間では TOGAF（The Open Group Architecture Framework）に引き継がれています。なお TOGAF はビジネス分析の知識体系である BABOK[4] に BA（Business Analysis）の観点から影響を与えており、さらに BABOK は JISA が策定した要求工学知識体系である REBOK[5] に影響を与えています。一方で、FEAF は、各国政府の EA に影響を与えており、経産省 EA もその一つです。

運用管理や品質ケースについては、現在は他の系列と強い繋がりはまだ見られません。しかし、ITIL V3[6] ではサービス運用に「要求実現」という内容が追加されており、運用管理の視点からも要求が重視されてきています。反対に要求工学的にも開発時と運用時における要求マネジメントサイクルを回す要求状態管理モデルが研究されている、TOGAF でも要求管理が重視されるのに加えモニタリングが重要とされているなど、さまざまな手法で要求－＞開発－＞運用－＞要求というサイクルを統合的に捉え、運用時の変化に迅速に対応しようとしています。またシステムの安全性を確認するための保証ケースが ISO/IEC15026 で提唱され、保証ケースをシステムや製品の開発プロセスに組み込もうとしています。JST（科学技術振興機構）の研究プロジェクトでは、保証ケースを拡張したディペンダビリティケース、さらにディペンダビリティケースを組み入れた開発プロセスである DEOS プロセス [7] の研究を進めています。

今後は要求変化への迅速な対応、製品品質の説明責任遂行のため、「運用管理」、「品質ケース」の知識を他の手法に組み入れる方向も考えられるのではないでしょうか。

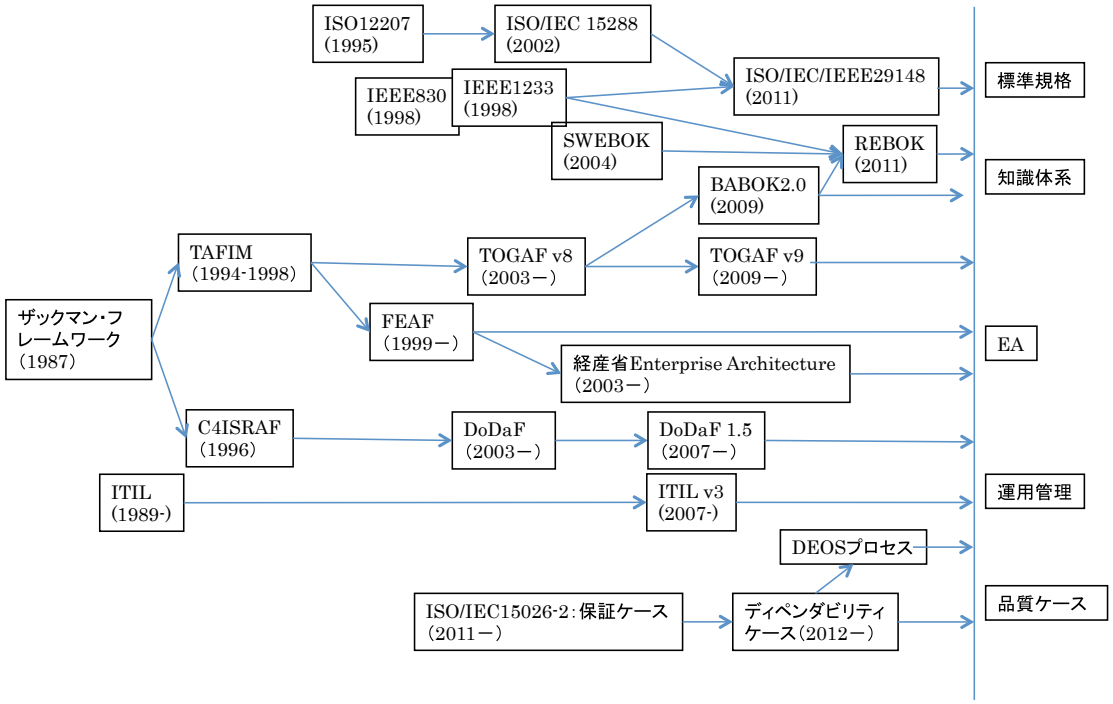


図 2.2.3- 1　要求知識体系の系統図

なお上記に挙げた手法につきまして、「表 2.2.3-1　要求工学手法一覧」に簡単な説明を記載します。

表 2.2.3- 1　要求工学手法一覧

手法	説明
標準規格	国際標準化機構(ISO)、国際電気標準会議(IEC)あるいは IEEE によって制定された国際的な標準規格
ISO12207	ISO によるソフトウェアエンジニアリングのライフサイクル全般についての標準
ISO/IEC 15288	ISO/IEC によるシステムエンジニアリングのライフサイクル全般についての標準
IEEE830	IEEE によるソフトウェア要求記述(SRS)の内容と品質についての標準
IEEE1233	IEEE によるシステム要求記述(SyRS)についての標準
ISO/IEC/IEEE29148	ISO/IEC/IEEE によるシステム・ソフトウェアの製品／サービスを対象とした要求工学におけるプロセスあるいは成果物についての標準
知識体系	関連する専門組織によって策定される専門領域の知識を定義するドキュメント記述
SWEBOK(ソフトウエア工学知識体系)	IEEE Computer Society スポンサーにより定義されたソフトウエア工学分野の知識体系
BABOK(事業分析知識体系)	NPO 法人 International Institute of Business Analysis (IIBA) が発行したビジネスアナリシス(BA) のベストプラクティスを体系化した書籍
REBOK(要求工学知識体系)	情報サービス産業協会(JISA)が発行した要求定義を組織的に実践するためのガイドライン

表 2.2.3- 1　要求工学手法一覧（続き）

EA	企業全体のシステム最適化を目的としたフレームワーク
ザックマン・フレームワーク	ジョン・ザックマンにより制定された、情報システムのアーキテクチャ・フレームワーク
TAFIM	米国国防省によるアーキテクチャ開発の方法論を含むように拡張されたフレームワーク
C4ISRAF	米国国防省により作成された TAFIM の後継のフレームワーク
FEAF	連邦 CIO 協議会による米国連邦機関向けの EA 構築指針とガイド
TOGAF	民間標準化団体オープン・グループ（The Open Group）が作成したビジネス向けのアーキテクチャ・フレームワーク
DoDaF	米国国防省による軍事及びビジネス両方の運用向けのアーキテクチャ・フレームワーク
運用管理	システム・ソフトウェア運用管理のための知識体系
ITIL	イギリス政府の CCTA（中央コンピュータ電気通信局）により公表された IT サービスマネジメントにおける成功事例をまとめた書籍
品質ケース	製品品質を証明するための手法
ISO/IEC15026-2	システム/ソフトウェアの品質を保証するための保証ケースの構造と内容

2) 分類の整理

手法の将来像を予想するために、以前と比較して知識体系がどの領域に広がりを見せているか、また、どのような知識構造に進化しているかを確認していきます。そこで、実世界「経営／ビジネス」から IT システム「システム／ソフトウェア」に至る「要求の領域」を横軸にとり、領域の広がりを把握します。そして縦軸はプロセスから詳細な技法に至る「知識構造」[8] とし、プロセス知識を上位に、生産物に対する知識を表したプロダクト知識を中間に、テクニックを表す技法知識を下位に位置づけます。上位の知識は下位の知識領域によって階層的に知識が分解されるという構造になります。（図 2.2.3-2 参照）

「系統樹」に記載した手法を「分類」で整理するのですが、手法の移り変わりを明確にしたいので、2005 年以降に策定された（あるいはバージョンアップされた）手法を実線、それ以前の手法は破線で囲み、区別しました。

まず標準規格ですが、ISO12207 はソフトウェアライフサイクルの規格、ISO/IEC15288 はシステムライフサイクルの規格ですので、図の右上に配置しています。IEEE830（ソフトウェア要求仕様）と IEEE1233（システム要求仕様）は要求仕様書の目次例、要求分の制限構文といったプロダクト知識を提示しています。ISO/IEC/IEEE29148 はこれらの標準規格を統合化し、さらにステークホルダ要求まで対象を拡張しています。

次に「xxBOK」という名称の知識体系ですが、BABOK はビジネス分析知識、REBOK は要求工学知識、SWEBOK はソフトウェアソフトウェアエンジニアリング基礎知識を体系化したものです。中でも SWEBOK では、必要な知識は参考文献の参照を推奨しており知識そのものは詳細に記述していないので、「プロセス知識」寄りに配置しています。また、BABOK がビジネス分析に焦点を当てているのに対し、REBOK はプロジェクト内での要求分析に注目していることから、BABOK を「ビジネス」寄りに配置しています。

このように整理すると、標準規格、知識体系はソフトウェア／システム要求の領域から運用／ビジネス要求の領域に拡張してきていることが分かります。

EA フレームワークはもともと技術アーキテクチャのためのフレームワークとして始まっているので、IT には強いのですが、ビジネスや組織全体にかかわる部分は相対的にあまり記述されていません。しかし TOGAF は最近のバージョンアップにおいて、企業全体を扱うための拡張に力を入れてきている［9］ので、FEAF、DODAF よりもビジネス側に配置しています。知識構造としては、FEAF は比較的大まかな枠組みだけを規定するにとどまっている、DODAF は成果物およびその記述法を詳細に定義している、TOGAF は成果物の定義にとどまらず詳細な EA 構築手法も提供しているといった違いがあります。

ITIL は運用管理に関するガイドラインですが、作業の詳細な記述が薄いので「プロセス」寄り、品質ケースの手法は製品品質を保証する具体的な方法を研究していることから「技法」寄りとしています。ITIL におけるサービスの優先順位を決めるためのポートフォリオマネジメントや品質ケースにおけるアーキテクチャ選定の正当性を保証するという考え方は最近要求工学で話題となっていますが、これらは TOGAF の中でも整理されています。

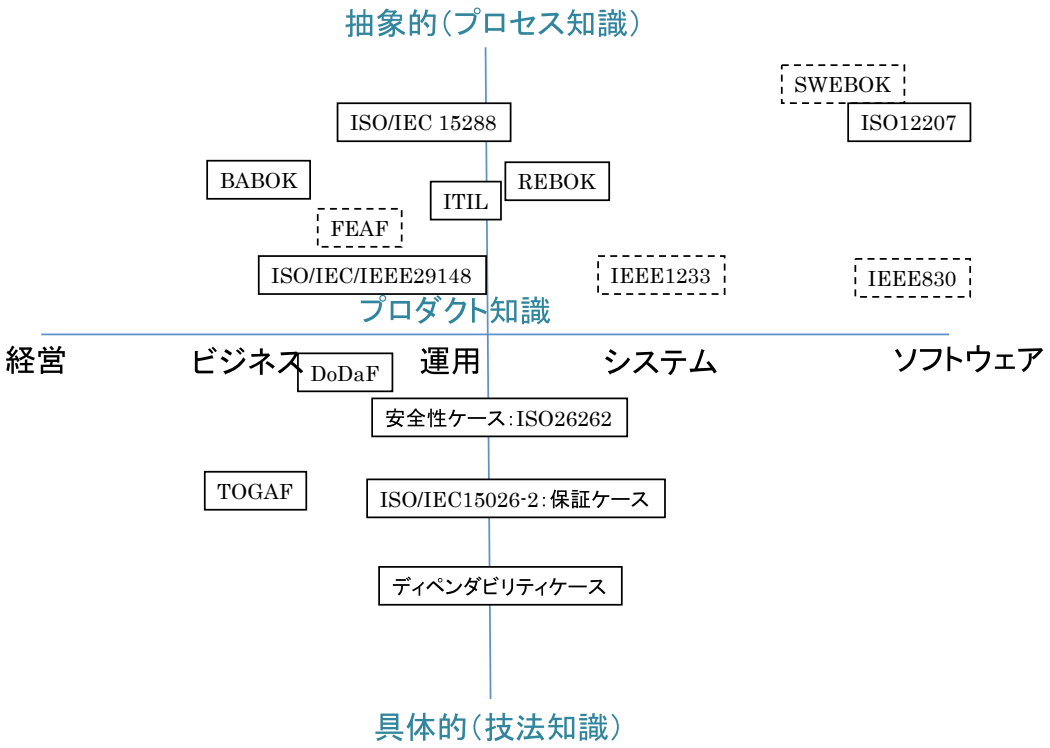


図 2.2.3- 2　要求知識体系の分類

今回の「分類」の整理では「要求の領域」と「知識構造」を軸としましたが、今後の方向性をみていく上では以下のような軸を設けても面白いかもしれません。

オープン性（他の手法との組み合わせを推奨しているかどうか）
プロセスとアーキテクチャ
静的と動的
全体的と個別的

（3）今後の方向性の予想

マンダラ分析法により整理すると、要求の源泉でもあるビジネスや運用の領域への拡張・統合化、全体最適化の考えを提唱している EA の進化、製品品質を証明するための技法（品質ケース）の登場などが最近の傾向として見えてきます。その背景としては、他社との競争がますます激しくなってきているため IT をビジネス価値の最大化／効率化に迅速に活用していく必要性が増していること、システムの複雑化・高度化に伴いバグ発生の完全防止はもはや不可能であり、発生した際の経済的損失・社会的影響が少なくないケースが増えていることなどが考えられます。

また「知識構造」としては、BABOK、REBOK における全体の枠組みの中での技法の使いどころの提示、EA における知識構造のモジュール化など、他の優れた手法と組み合わせていく方向性が分かります。

このような傾向から今後もビジネス領域や運用領域、さらに経営の領域まで含めた統合化が進むことが予想されます。それも 1 つの知識だけではなく、ビジネスやシステムの多様性や複雑度に合わせて複数の手法をうまく組み合わせることが不可欠になっていくと思われます。一方で、外部・内部の環境変化はさらに速くなり、バグの顕在化、リスクの重なりにより多大な影響を与えるケースが増えると考えられます。したがって、今まで以上に環境変化を見越したリスク分析の研究が進み、そして要求工学の手法からも参照・統合化されていくのではないでしょうか。

参考文献

[1] IEEE Std 830-1998, IEEE Computer Society, IEEE Recommended Practice for Software Requirements SpeciÞcations,1998.

[2] ISO/IEC/IEEE29148, Systems and software engineering — Life cycle processes — Requirements engineering,2011.

[3] みずほ情報総研，経営とシステムをつなぐ EA フレームワーク，http://www.mizuho-ir.co.jp/publication/contribution/2004/ea0404.html

[4] IIBA 日本支部 , ビジネスアナリシス知識体系ガイド , 2009

[5] 情報サービス産業協会 REBOK 企画 WG 編，要求工学知識体系 REBOK(Requirements Engineering Body Of Knowledge), 近代科学社，2011

[6] ITIL, itSMF japan http://www.itsmf-japan.org/itil

[7] DEOS, http://www.dependable-os.net/osddeos/index.html

[8] 山本修一郎 , ソフトウェア知識体系のメタモデルについての一考察 , http://www4.atpages.jp/sigksn/conf12/SIG-KSN-012-02.pdf

[9] みずほ情報総研，TOGAF9 ビジネスと IT をつなぐ EA のためのフレームワークに関する研究，http://www.mizuho-ir.co.jp/publication/contribution/2010/itarchitecture0904_03.html

（福島 著）

第３章　開発手法・技法・ツールの各論

3.1　ビジネス・モデリング

（1）　ビジネス・モデリングの必要性

3.1 節は､「ビジネス･モデリング」と題して､システム開発の早い段階で実施する「要件定義」という､要件の識別､整理・分析・定義する作業の中で、モデルとして作るべき対象（つまりビジネス活動）をどのように理解したらよいのかということを中心に記述します。なお、ここでは紙面の都合上概要を記載するにとどめています。

要件定義の作業には多くの方法があります。SLCP や UML ベースの開発方法論、あるいはベンダーが提唱する方法論がその代表です。しかしながら、そういう開発方法論を読んでも、ハウツー（料理で言えば、包丁のようなもの）についての記述はあっても、料理する対象についての記述は一切ありません。要件が発生する対象であるビジネス活動というものを、分析者がイメージを持たずに料理をすることはとても危険です。包丁を持って出かけたら熊に出会うようなことが我々の仕事には多くあります。

この資料では、「ビジネス・モデリング」という言葉を「ビジネス活動における主要な要素を識別し、要素間の関連構造（静的構造と動的構造）と意味（セマンティクス）を抽象化して（モデル化して）表現する作業」という意味で使います。

あえて付け加えることがあるとすれば、「その結果が、情報システムの開発につながること」という条件をつけることができるでしょう。つまり、「情報システムの開発を前提に、システム化を行う上での分析対象であるビジネス活動に関わる主要要素を識別しその全体像を枠組みとして表現すること」ということです。その結果が、ビジネス・モデルということです。

図 3.1-1　ビジネス・モデリング

要件定義という作業の中で、ビジネス・モデルを可視化するということはなぜ必要なのでしょうか？　それを考えるには、現在の情報システムが抱える課題が、以前とは質的に大きく変化してきたことが理由としてあげられます。ポイントは二つあります。

① 情報システムが経営環境の変化に迅速に対応することが困難である

② 情報システムの開発や維持・改善・運用を、責任を持って主体的に行える企業が少なくなっている（ビジネスの仕組みが見えなくなってきている）

これらの問題は決して新しいものではなく、10 年ほど前から言われています。しかし、改善したという話はあまり聞きません。システムの変更作業は単純ではなく複雑で多岐にわたるということもありますが、まだ広くは認識されていませんが、次のようなことも大きな理由なのではないかと考えています。

・ビジネスの構造とシステムの構造とのマッピングができていない

・企業の情報システム部門の中に、ビジネスを理解している人材が少なくなった

そのことを念頭に、以下にビジネス活動をどのように理解したらよいのかという点について記述します。

（2）　ビジネス活動とそのモデル

ビジネス・モデリングの対象であるビジネス活動には、非常に多くの要素があります。情報システムの機能を定義するという観点から、IT 技術者は従来から“機能”というものが主要要素だと考えてきました。要件定義で「機

能一覧表」という成果物を作成していることから判断するとそのように思えます。また、「データモデル」や、「業務フローという名のもとに作成している“仕事の流れ”」もビジネス活動の主要要素ととらえていると判断してさしつかえないでしょう。

一方、経営戦略や経営マネジメントに関する本には、「顧客」や「事業（提供する製商品やサービス）」、あるいは「市場」こそが主要要素と考えられています。また、サプライチェーンの観点からは、「事業を行う拠点・場所」、「関連会社」、あるいは、「取引先」などもビジネス活動を行う上での主要要素と考えられます。

また、担当者レベルの仕事を考えてみれば、顧客と商談をしたり、打ち合わせや提案資料を作成したり、情報を検索するなどの仕事が、実務としての仕事の大部分を占めているのも確かです。

こういった中で、ビジネス活動の主要要素とは何でしょうか？　詳細な説明は、ここでは割愛しますが、ビジネス活動における主要要素は多くあり、また複雑な関係を持っています。その中でビジネス・プロセスと業務機能は、目標や戦略を実現する上での戦術であって、そういったものが要件としてでてくる背景、－　つまり、目標や戦略との関係　—を押さえておかないと、それこそ“経営に役立つ情報システム”を開発し、ビジネス環境の変化に迅速に対応することは困難です。

従来の要件定義という作業は、個別に挙がってくる要求事項をなんとなく整理しながら文書化（Office ツールで!!）しているのですが、そういった個別の要求事項は、ビジネス活動の目標や戦略との関連づけがなされていないものが多いようです。また、Office ツールで作成しても多くの変更が発生するのがあたりまえですから不整合が発生するのは当然です。また、要求事項の整理の仕方についての基準はなく、単に“合意できたから適切だ”と考えて進めているように見えます。要件定義書が不整合と網羅性に課題が残り信頼感に欠けるものとなりがちなのも頷けます。

事業目標や戦略があり、それが戦術として展開されるのがビジネス・プロセスや業務機能であり、それらの全体がビジネスのモデルです。そういった事柄のすべてが関連付けされたモデルとして表現されない要件定義書には、ビジネスへの貢献、あるいは、合目的性という観点から問題があるといえます。また、システムの稼働後もその関係をもとに維持改善していくためには、その関係を単に Office 文書で作成するのではなく、ビジネス・モデルの要素間の関連を保持した何らかのリポジトリーの構築が必須となります。

（3）　ビジネス機能とシステム機能の関係

ビジネス・モデルの主要要素の構造をリポジトリーで管理したとしても、開発するソフトウェア・システムのモデルとの追跡可能性を維持しなければ、ビジネスの変化がシステムの要素（特にソフトウェア）にどのような影響をあたえるのかを把握することは困難です。そんなことは当たり前で、できていると考えている方がいたら少しお待ちください。本当に適切な管理はできているでしょうか？　図 3.1-2 をご覧ください。

この図は、いわゆるシステム機能体系図ですが、この図には大きな”勘違い”が認められます。詳細な説明はここでは省きますが、上位に「業務機能」を配置し、それを展開して「システム機能」を識別しているのですが、「業務機能」と「システム機能」の関係は、本来「多対多」なので、このような単純な階層構造で示すことはできません。そこが、“勘違い”という点です。

もちろん、システム機能それ自体も上位の機能と下位の機能との関係は「多対多」です。そのことは、言われてみればすぐにわかることですが、実際に機能体系をそのように管理してシステム開発プロジェクトを進めているところはまずないのではないかと思います。

業務機能とシステム機能の関係、そして、展開した上位と下位のシステム機能間の関係がいづれも「多対多」になるということは、実は、システム開発プロジェクトの体制および進捗や品質の管理のメカニズムに非常に大きなインパクトを与えます。従来から行われているシステム開発プロジェクト組織は、基本的に縦割りです。本来は、マトリックス型の体制を構築してプロジェクトを運営しなければならないことを意味します。

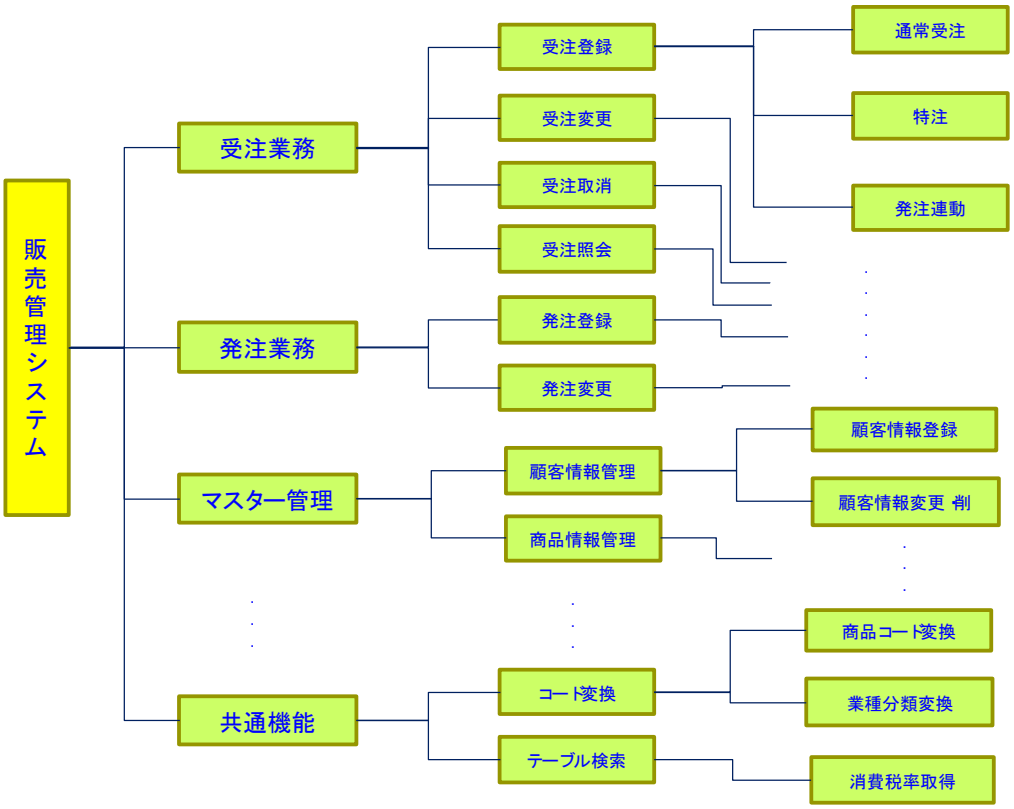


図 3.1-2　システム機能体系図の例

（4）　今までの開発パラダイムに決定的に欠けている視点

これまで述べてきたような、ビジネス・モデルの構造をメタモデルで管理し、情報システム・モデルの構造とのトレーサビリティを管理している企業はおそらく皆無ではないでしょうか？　経営の変化に迅速に対応できるためには、「ビジネス・モデルと情報システムのモデルのトレーサビリティを管理すること」は必要条件だと考えます。これができていないので、「経営環境の変化に迅速に対応できない情報システム」となるのは当然です。ですから、この課題は 10 年以上たっても解決できないのです。

もちろん、ビジネスの変化に迅速に対応できるようにするためには、トレーサビリティの管理は必要条件ですが十分条件ではありません。トレーサビリティが管理できるので影響を受けるシステム要素の特定は容易になりますが、要件の分析、設計、開発、テスト、文書化、マニュアル改訂などの作業は必要です。これらの工数と期間を短縮する現時点で可能な方法は、設計情報をもとにコードを自動生成する方法でしょう。テストもある程度は自動化できます。

情報システムは、ますます業務そのものに深く入り込んでいきます。エンタープライズ系の情報システムは、将来、BPMS (Business Process Management System) や BRMS (Business Rule Management System) が業務に組み込まれ、それらのツールに登録されているビジネス・プロセス、ビジネス・ルール、あるいは、組織、戦略、目標、方針などを変更することで、プログラムが自動生成される世の中になると確信しています。

そして、企業や行政組織において、情報システムの価値が組織の価値になる時代となるでしょう。業務それ自体も縦割りでは顧客の要求に耐えられなくなっていることは明白であり、また、ソフトウェアの世界だけをとってもマトリックス組織を確立し、ビジネス・モデルと情報システム・モデルのトレーサビリティを管理できる仕組みを構築しながら、開発・保守（維持、改善）を行うようにしていくことが今後必須になると確信しています。そういう仕組みを持っている組織とそうでない組織では、ビジネスの競争力に圧倒的な差が生まれているのは間違いありません。そのことを十分認識されてはいかがでしょうか。

〔大島 著〕

3.2　BMAPROS

日本ユニシスの製造物流部門において開発された業務分析手法 BMAPROS を紹介します。

（http://www.unisys.co.jp/tec_info/tr105/10507.pdf）

（1）開発の背景

昨今では、多くの業務領域に対するシステム化が一巡し、新たな追加要求に対応するためのシステムの再構築が増えています。しかし、再構築にあたり実際に業務分析を実施すると、システムの仕様を把握している担当者は顧客を含めて存在せず、また機能の追加・改修作業を積み重ねてきたことによりプログラムが複雑化しており、中に埋め込まれたビジネスルールを容易に理解できない事が多いのです。このような機能の変更や例外処理のパターンが多いシステム開発向けに、“ 様々な場面により変化する業務を可視化する業務分析手法 ” として開発されました。特に外的要因から “ 作業手順の違い ” や “ 例外的な振る舞い ” が多く発生する場合、要件定義の早い段階での抜け・漏れを防止するのに有効な手法です。

（2）概要

BMAPROS 手法では業務を３次元の箱で表現しており、一つの業務は業務軸に沿って並んだ平面を流れる連続した作業（業務フロー）として表現します。業務はいろいろな場面（状態、条件、タイミング等）により変化します。そこで、場面ごとの振る舞い（業務ルール）を定義し、さらに実際に発生するデータ（駆動データ）を定義することにより業務をモデル化します。

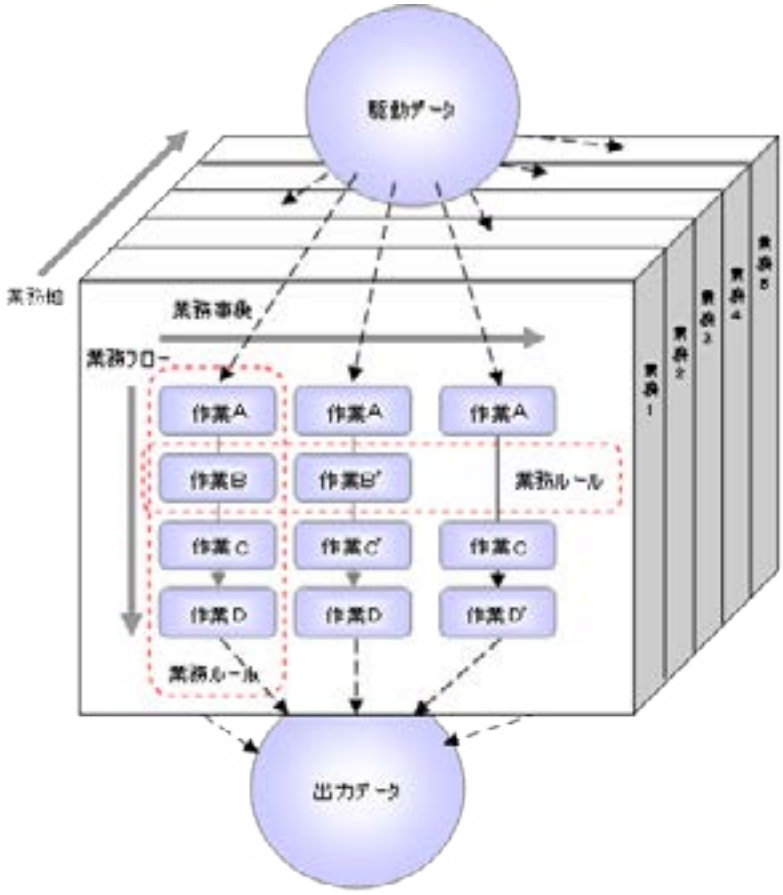


図 3.2-1　BMAPROS 全体像

（3）業務分析の進め方

BMAPROS による業務分析は、1）業務軸を定義する、2）業務フローを定義する、3）業務事象を定義する、4）駆動データを定義する、 5）業務ルールを定義する、という手順で進めていきます。

1）業務軸を定義する

対象業務全体で業務フローがどれだけあるかを定義し、全体像を把握します。例外がない標準の業務の始点から終点までの業務フローを一つの業務軸として捉えます。

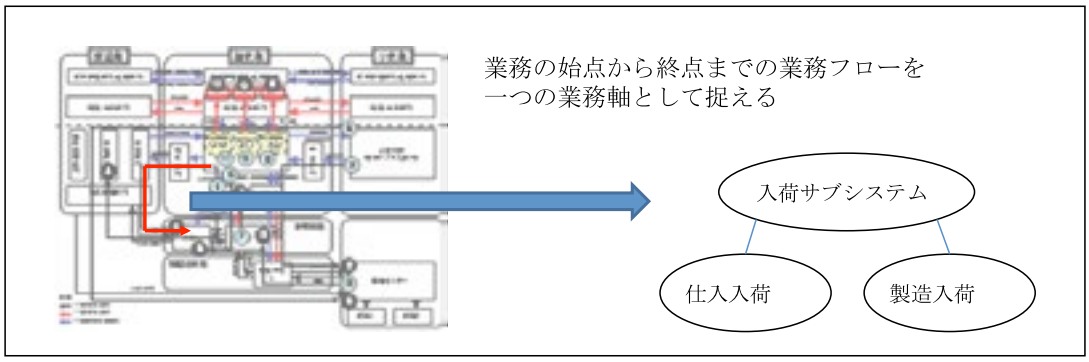


図 3.2-2　業務軸（例）

2）業務フローを定義する

業務軸単位に「作業・処理」を分解し，作業の流れを定義します。

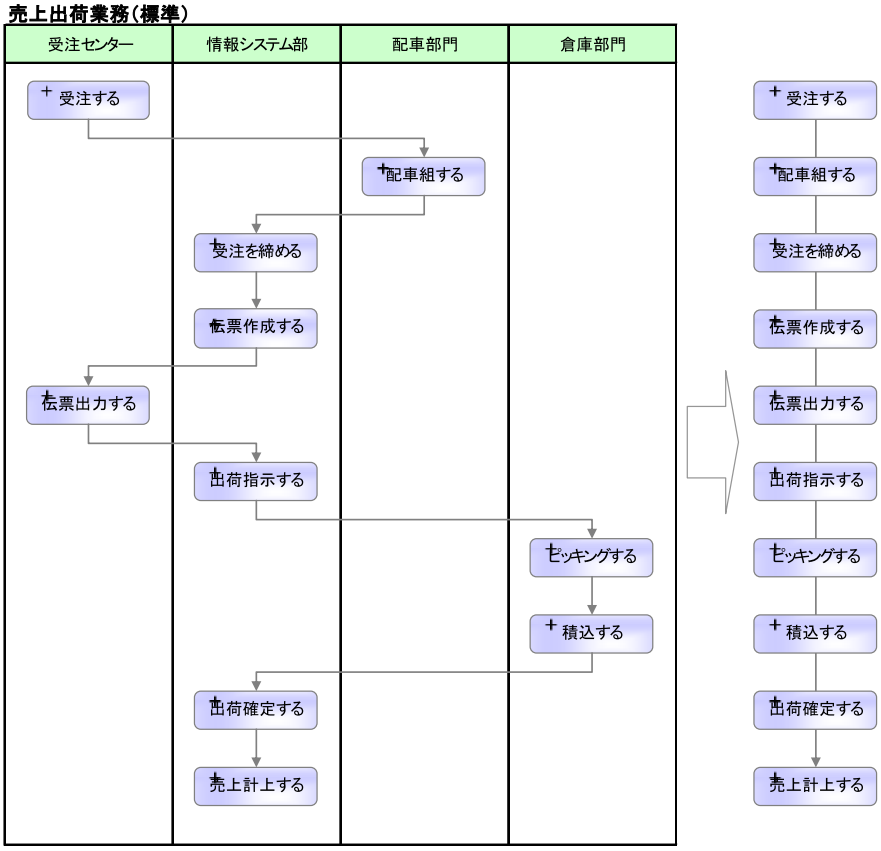


図 3.2-3　業務フロー（例）

3）業務事象を定義する

業務事象（業務に影響を与える外的・内的イベント）を抽出し定義します。この業務事象をどれだけ多く定義できるかが重要となります。

表 3.2-1　業務事象（例）

業務事象

一般的事象	顧客個別事象
・データの追加・修正・削除 ・品切れ・欠品の解消 ・返品、欠損・破損の発生 ・遅延（製造遅延・入荷遅延・出荷遅延等） ・作業手順省略 ・作業手順入れ替え ・締め時間を越えたデータの登録・修正・削除 ・確定処理を越えたデータの登録・修正・削除	・特定場所における個別作業 ・特定顧客に対する個別作業 ・特定商品に対する個別作業 ・特定の条件組み合わせで発生する個別作業 ・法規制による特殊処理 ・通例・慣例による特殊処理

4）駆動データを定義する

「作業の中における識別」に必要な属性を抽出します。マスタ、トランザクション等の各属性から抽出します。

表 3.2-2　駆動データ（例）

駆動データ

マスタ属性	データ属性	トランザクション属性
・商品属性とその組み合わせ ・場所属性とその組み合わせ ・顧客属性とその組み合わせ ・契約属性とその組み合わせ ・上記同士の組み合わせ	・情報分析系KPI指標 ・会計仕訳ロード ・特別課税項目	・作業状態・結果 1）受注済み 2）在庫引当済み 3）出荷済み 4）I/F済み 5）確定済み

5）業務ルールを定義する

業務と業務事象・駆動データを掛け合わせて業務分析マトリクスを作成します。作業・処理が発生しないのか、発生するとしたら標準ルールが適用されるのか、標準で無い場合にはどのような業務ルールが適用されるのかをチェックしていきます。

図 3.2-4　業務分析（業務 × 事象）マトリクス（例）

（4）効果

一般的に設計以降ではシステムをサブシステムと呼ばれる開発単位に分割し並行開発していくので、全体の機能実装の整合性を確保する事が難しくなります。しかし BMAPROS では、対象業務の想定されるあらゆる場面の振る舞いをあらかじめ定義しておくことにより、全体の機能実装を縦串・横串で俯瞰し分析できます。

業務分析マトリクスにより設計以降の検証項目が定量化されるため、作業完了クライテリアの閾値として活用でき、また、見積制度の向上も期待できます。テストシナリオ作成にも活用することも可能です。

（5）WG1 への展開に関する見解

本手法は製造物流のプロジェクトにおいて開発されたものですが､業種に特化したものではありません。本質は、業務において業務事象と駆動データを掛け合わせたときのルールの相違を明確にするところにあります。したがって他業種でも、ビジネスルールが複雑で、例外処理が多い、業務の複雑度が不明であるといった場合には、本手法を参考にすることにより、後工程での仕様膨張のリスクを下げることができます。同種のプロジェクトにおいて効果が期待できることから、今後 WG1 において手法基盤検討する際の候補に加えたいと思います。

また、本手法を適用して作成する業務分析マトリクスで定義する業務や事象の抽出は、分析者の経験から洗い出される業務知財そのものです。これにより、同様の業務の後続プロジェクトでは、マトリクスを再利用することにより効率的に業務分析が可能となります。それとともに、次世代の開発者への業務知識の継承にも役立つと考えています。

〔福島 著〕

3.3　ビジネス中心ソフトウェアエンジニアリング

注）本稿は、2012 年 4 月 10 日の第 4 回 WG1 で発表した、「ビジネス貢献を含んだソフトウェア開発」という発表に基づいたものです。

（1）日本のソフトウェア開発が抱える問題

日本のソフトウェア開発の現在の最大課題は、システム開発需要が顕在化していないことです。すなわち、日本全体の IT 投資が、金額ベースで減少傾向にあります。重要なことは、世界的に見れば、米国を筆頭にシステム開発需要は旺盛で、IT 技術者の給与は平均的には上昇傾向にあるし、日本においても、労働量としてのシステム開発需要が減少しているわけではないという点です。

顧客の現場に行けば、システムの保守改良を含めた開発需要は、アチラコチラに転がっています。営業現場では、価格を顧客が望む以上に下げることができさえすれば、受注案件は限りなくあるはずです。

適正な利益を確保できる水準での価格で、システム開発を請け負えないことが問題であり、これは、顧客が支払える価格帯で製品を提供できないために苦境に陥っている日本の TV メーカの苦境に重なります。赤字でも、稼働しないよりはマシだという受注活動は、自分たちの首を締める結果にならざるをえません。

この問題に対して、ソフトウェア・エンジニアリング、当 WG のソフトウェア開発論は、どのような解決策を提案できるのかが問われます。筆者の答えは、本節のタイトルである、「ビジネス中心ソフトウェアエンジニアリング」にあります。

簡潔に述べれば、従来のソフトウェアエンジニアリングは、納期、品質、資源、コストしか考慮してきませんでした。ビジネス中心ソフトウェアエンジニアリングでは、ビジネスにどれだけ貢献するかという経営視点でソフトウェアを開発運用します。

これは、これからは、「IT よりも BT」というスローガンに集約される顧客の意識に合わせて、ビジネスに直結する ICT の開発運用を志すということです。

（2）新たな動き

これに呼応した新たな動きが始まっています。旧来のウォータフォール型開発から、アジャイル型開発への動きは、ビジネス面からも説明できます。ウォータフォール型は、経営環境が一切変化しない安定したビジネス状況で、インタフェースの完全性、無謬性を前提とした分割統治法式で、開発コストを回収するものでした。今や、経営環境は、一寸先が闇という不確定な状況にあり、ウォータフォール型のような固定した開発費を計上する訳にはいきません。ビジネスの環境変化に合わせて。開発を変更できるアジャイル型のソフトウェア開発が必然的に求められます。

要求定義や要求開発への興味も、同様に、変化するビジネス環境において、本当に有効なシステム開発が望まれているからです。USDM (Universal Specification Describing Manner) では、ソフトウェアの仕様に対して、それがなぜ必要かの理由を書きます。もはや、システムが稼働するビジネスの状況を知らずして、仕様を記述することなどできません。

また、先進企業においては、情報システム部門を解体して、開発営業部門にシステム開発運用を任せる取り組みも進んでいます。これからのビジネスには ICT が不可欠どころか、ICT 技術が支える情報処理そのものが本質的な事が多いのです。政府機関などは、その典型でしょう。まさに、ICT ＝ Business Technology なのです。

（3）ビジネス中心ソフトウェアエンジニアリングに向けて必要なことは

それでは、ビジネス中心ソフトウェアエンジニアリングに向けて、今、必要なことは何かをまとめておきます。

1) ソフトウェア開発側からのビジネス提案の姿勢

まず大事なのは、ビジネスに関わるという積極的な姿勢。ビジネスのことはアナタの仕事。私は、ソフトウェアを担当しているだけですという守りの姿勢では、貢献はおろか、参加もままなりません。

2) ビジネス貢献とシステム機能との紐付け

USDM のなぜかという理由、トレーサビリティの確保。「あったらいいな」から一歩踏み込んだ、「これだけ役立つはず」というシステム構築。

3) ビジネス貢献の定量化

これは、今期の経営計画ですらできていない事が多いから難しいことだけど、取り組みが大事。やっていけば見当がつくし、原価積み上げでなくて、成果共有型の価格形成につながります。そうすれば、ブランド戦略も見えてくるし、グローバル対応も可能なはずです。

4) ソフトウェア評価とビジネス評価の連携

ビジネス評価がソフトウェア評価に繋がる仕組み。ビジネスの効率と価値を高めるために、ソフトウェアを進化させるという取り組み。

5) ビジネス貢献が見えるソフトウェアエンジニアリングの手法・システム

大学の教育も、大学関係者のための教育ではなくて、学生のための教育にかわらなくてはいけません。

（4）今後の展望について

「ビジネス中心ソフトウェアエンジニアリング」に関係しそうな話題や動きを展望しておきます。

1) ソフトウェア＋ハードウェア＋デザイン＋データの総合力が、これからの製品とサービスの競争力の中心になる。どれ一つが欠けてもうまくいきません。

2) Big Data をいかすソフトウェア開発がこれから出てくるでしょう。ビッグデータのリアルタイム性がどのように生かされるか興味深い。

3) 世界的には、数年前からの IT（ICT）融合が一段と進み、情報関係の科学技術はさらに応用分野との融合が進む。ソフトウェア・エンジニアリングも、アプリケーション・エンジニアリングとの一体化、融合化が進みます。システム開発者は、一般的な問題の発見と解決を期待されるようになるでしょう。

参考文献

・Sommerville,　Software Engineering 9th Ed. 2010
・清水、要求を仕様化する技術・表現する技術 - 入門＋実践 仕様が書けていますか？技術評論社、2005
・黒川、情報システム構築の品質・信頼性向上のために――上流工程の ” ビシネスルール ” と要求工学を検討する――、科学技術動向 No.32、文部科学省、2003 年 11 月
・黒川・品川、ソフトウェア・テストの技術動向と課題、科学技術動向、No.85、2008 年 4 月

〔黒川 著〕

3.4　GeneXus

（1）地球の裏側発の開発ツール

この数年は知名度が拡大している GeneXus ですが、５、６年前までは説明をすると IT の知識をよく知っていると自負している人ほど信じられない！今の話は本当なのか？と言われたものです。

四半世紀より前に、ウルグアイの大学教授と天才的な助手が IBM の勧めで IBM ホスト上のプロトタイプを AS/400 に移植したのが GeneXus の始まりです。

GeneXus のメーカであるアルテッチ社がある南米ウルグアイの首都モンテビデオに毎年「GeneXus　World Meeting」が開催され、世界 30 か国から３千人以上の関係者が集まります。そして１日 30 くらいのセッションが３日間連続してあり、多くの GeneXus を愛する人たちの交流があります。ウルグアイの人口 90％以上がヨーロッパからの移民ですので、モンテビデオの街並みもヨーロッパの古い街に似ています。

数年前にアルテッチ社を訪問した人の話を聞くと GeneXus を開発するチーム 20 名ほどガラスで囲まれていた部屋で作業をしていたが、あちこちのガラス（壁）に 2、3 名が集まり、ガラスをホワイトボード代わりにものを書きながらミーティングしていたということです。

（2）GeneXus とは

業務仕様（業務内容の記述：What）から、様々な実装環境に合わせた実現方法（ロジック：How）を推論し、自動的にシステムを生成するという DOA（データ中心アプローチ）による統合開発ツールです。データ分析結果と業務ルールを入力することにより、「プログラム言語」、「データベース」、「実行環境」を意識することなく、システムを自動生成します。 自動生成されたプログラムはその後技術者が手を加えなくても問題なく動きます。

GeneXus は単にシステムを構築するだけではなく、プラットフォームが変わってもシステム資産を永続的に継承可能とすることができ、将来の技術革新に対応します。エンドユーザに好まれますが、従来のスクラッチ開発でゴリゴリプログラム作成をする IT ベンダーには嫌われる理由です。

GeneXus は 20 年以上の歴史を持ち、戦略的パートナーとして IBM,Microsoft,Oracle.HP,Intel 各社の技術と連携して開発されており、その時代の主要なプラットフォーム、データベース、プログラム言語に対応してきましたが、将来においてもメジャーな技術に対応し、ジェネレータを提供するとしています。一回、GeneXus に INPUT すると、対象ごとにシステム開発が必要なくなります。

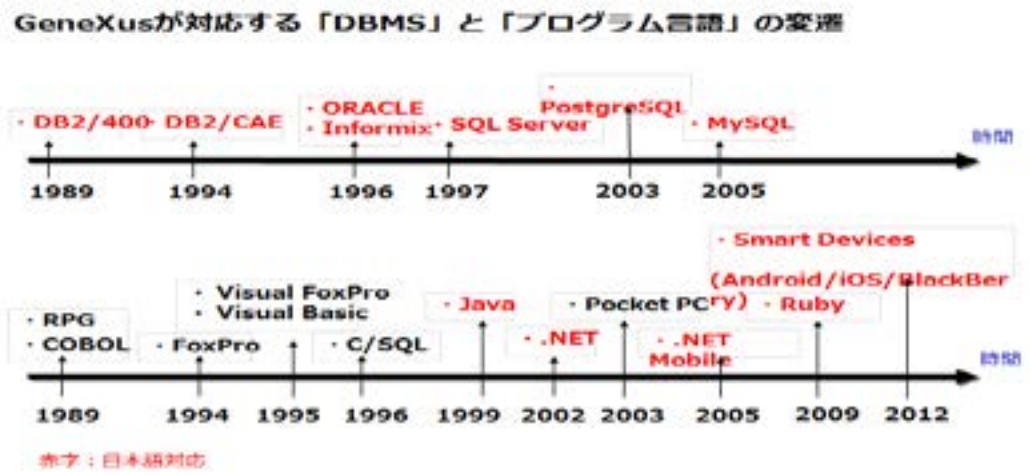
（3）GeneXus の機能

一般にプログラムは　何をするか（What）が約 20 ～ 30％、残りの 70 ～ 80％がどのようにして実現するか（How）を記述されるといわれています。

GeneXus は What にあたる業務ルール、データ構造項目を INPUT すると、Prolog として中間言語になります。これを仕様解析して、指定されたアプリケーションプログラムとデータベース物理テーブルを自動で生成することができます。

この中間言語を保持することで、他のプログラミング言語やデータベースにもジェネレータを通して生成することが可能となり、マルチプラットフォームにも対応することができます。将来の技術に対してもジェネレータを提供していくことで自動生成しながら　システム資産の継承が実現できるようになります。

以下に　GeneXus が対応してきた技術の歴史を示します。



一つの中間言語から、Java、.NET や最近では Android、ios、Blackberry などの SmartDevices 対応のプログラムまで生成してしまいます。

（4）GeneXus の開発方法論

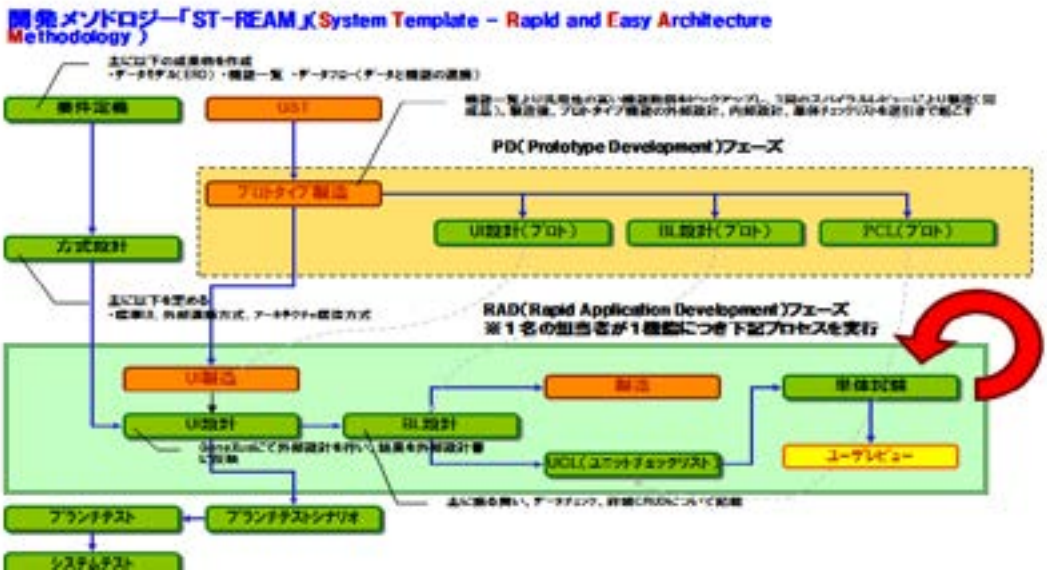
GeneXus による業務システム開発で、適していると思われる開発方法の一例を示します。（開発メソドロジー「ST-REAM」参照）

小規模開発であればアジャイル開発方式でも有効ですが、業務システムの開発の場合、ユーザインタフェースや操作面における統一性やデータベース構造を確定することが有効であるため、基本的に要件定義、設計、製造、テストという工程単位で確認していく方法がまだ日本企業には求められます。

設計工程でそのシステムにおける主要機能からいくつかを選び出し、GeneXus でプロトタイプを製造します。このプロトタイプは見せるだけのものではなく、本番で使えるようなU／I、操作、外部連携、セキュリティ面などが装備されたもので、ユーザがこれから開発するシステムを容易にイメージできるようになります。また、共通機能要件、非機能要件（ログ出力、権限管理等）も実装を行うことで早期の「システムの見える化」を実現することが可能となります。

ここで何回かユーザに使っていただき、修正要望を聞きながら、修正を迅速に行い精度を高めていきます。ユーザの了解が得られた段階で、各機能の設計、製造の工程に入りますがプロトタイプを標準としながら、製造工程でも数回ユーザに確認、修正要望をとりこんでいく方法をとります。

結果として、テスト段階になっての前工程への戻り作業が発生せず、ユーザが確認しながら進められることで納得度が高いシステム開発ができるようになります。



GeneXus に最初から INPUT することは習熟していない技術者にとって、慣れるまで（通常システム開発経験者であれば 3 か月で GeneXus を使えるようになります）多くの作業時間を費やすことと品質低下を招くこともあるため、GeneXus の開発フレームワークとして汎用業務モジュール集（GeneXusSystemTemplate）の活用が効果的です。一般的なシステムで必要となるログイン画面、メニューや明細入力、マスタ業務などが備わっていますので、これを開発するシステムに合わせてカスタマイズすることで、さらに短期間のシステム開発が可能となります。

（5）GeneXus の活用前提

GeneXus は DB 物理テーブルを自動生成するのですがその前提として考えておくことがあります。

DOA(Data Oriented Approach；データ中心アプローチ）の知識ノウハウが重要となってきます。データやデータベースを中心に考えるというデータ処理アーキテクチャの考え方です。

また、凝ったユーザインタフェースやデザイン性が高い画面の要求には GeneXus 本来の高生産性が発揮できないことも前提として考えておく必要があります。

GeneXus ひとつの業務システム開発に活用しても工数削減の効果がありますが、マルチプラットフォームですので、異なる OS で使うパッケージや SaaS ソフトウェアとして活用できます。横展開するビジネスに最適といえます。

（6）GeneXus への期待

現在は設計工程の一部からプログラム製造までの工程を GeneXus の活用範囲としていますが、要件定義上流工程のツールで整理し、その情報データを GeneXus に Input することで、さらなる工数短縮や要件の誤理解や要件欠落することを防ぎ、品質向上に結び付けることが考えられています。また、GeneXus で生成されたソースプログラムからテストシナリオの自動生成及びテストの自動化を行うことでシステム開発全体の生産性と品質向上を検討されています。

これらができますと、システム開発が力作業や残業過多である仕事ではなく、創造的に考えるという技術者が必要になってくると思われます。

GeneXus 開発環境もクラウド環境で開発できるようになると、開発するコスト感も安くなり、GeneXus 技術人口は世界中に拡大していくようになります。

企業の経営戦略策定から事業開始までの期間が短縮でき、日本企業のスピード経営の応援をする武器になることが期待されます。

参考文献 『はじめての GeneXus』

〔樋山、山下 著〕

3.5　SAPIENS

SAPIENS は 1980 年代後半、イスラエルの「ワイツマン研究所」で産声をあげ、研究者たちは、次の２つの目標

「プログラムレス」

「環境からの独立」

を実現し、アプリケーション開発に革命をもたらしました。

（1）プログラムレスでアプリケーションを作る

あらゆるアプリケーションは、プログラム無には動きません。プログラミングは、開発工程の中で下流工程と位置づけられ、人力で行う極めて生産性の低い作業ですが、プログラミング作業を行うためには、アプリケーションの要件、データベースの構造や項目など上流工程を完全に用意しないと始めることはできません。いわゆる Water Fall 型のアプローチが必須となります。

実際は、ビジネスアプリケーションは作って使って初めて機能不足、実際の業務との乖離など様々な問題が見つかってくるのが当たり前です。そもそも、この問題の解決には Water Fall 型のアプローチは無理なのです。Water Fall 型のアプローチはプログラミングが前提ですから、脱却するにはプログラムをなくすことしか有りません。

また、作られたアプリケーションは経済状況の変化などで絶えず変更、改良を続けなければなりませんが、プログラムはスパゲッテイ状態になっていて、簡単には修正できません。SAPIENS はプログラムレスでアプリケーションを作ることを実現させました。

（2）環境から独立してアプリケーションを作る

これまでのアプリケーションは、環境を意識して作られました。Hardware メーカーの選定、プラットフォームの選定からスタートし、提供されている OS,DB, プログラム言語、ユーティリティを意識せざるを得ません。その為に、環境を変えることは容易ではありませんでした。

SAPIENS は、環境を意識せずに開発することができ、プラトフォーム間でのアプリケーションのポータビリティが保障されています。SAPIENS は環境を意識しないでアプリケーション開発を実現させました。

（3）SAPIENS の開発方法

SAPIENS は開発しようとするアプリケーションで把握すべきデータ（データモデル）を描き、ビジネスルール（ルールモデル）をデータにアタッチすることで、アプリケーションを動かすことが出来ます。ソースプログラムも作りません。

SAPIENS は実行環境も提供しており、推進エンジンを持つなど従来のアプリケーション開発の常識を打ち破るものです。

（4）サピエンス・ジャパンは、国内で 20 年に渡る SAPIENS の製品ベンダーです

本田技研､TDK, 日本総研､キリンビール､アルプス電気（敬称略）など 160 社で SAPIENS を導入頂いております。

〔大北 著〕

3.6　業務システム高速生成ツールの開発と適用

（1）はじめに

近年、企業や組織の業務システム（エンタプライズシステム）の開発のアプローチとして、アジャイル開発に代表される繰り返し型開発の実プロジェクトへの適用が増加しています。繰り返し型開発の特徴として、初期の開発サイクルでは主要機能に特化したシンプルな業務システムを開発・導入し、利用したユーザからのフィードバック（追加 / 変更要件）を反映して機能拡充のサイクルを繰り返す点が挙げられます。

この繰り返し型開発の効果は、短期間にフィードバックループを回すことにより、ユーザの要件を的確に把握し、満足度の高いシステムを開発できることです。動くシステムを早期にリリースすることを優先し、実システムを利用して要件の確認を行いながら進めることで要件の把握が容易になるといえます。

また、繰り返し型開発では優先度の高い機能から徐々に開発していき、各開発サイクルのタイミングで実装する機能の計画を見直しその評価を行うことが一般的です。そのため、仕様の欠陥を早期に発見できるほか、作りすぎや要件の盛り込みすぎといったムダが省けるため、開発コストが削減できる効果もあります。

一方で、構築する業務システムが、ユーザからの継続的なフィードバックに対応するには要件の追加や変更に迅速に対応することが求められます。変化に柔軟で短期で改修が可能なシステムでなければ、開発サイクルを短期間で繰り返せず、納期やコストが計画より大幅に超過してしまうというリスクを背負うことになります。

そこで本節では、繰り返し型開発において要件の追加や変更に迅速に対応する手段として、要件定義工程での成果物であるビジネスプロセス、ビジネスルールと画面定義をインプットとした業務システム開発ツールを報告します。さらに、企業の業務システムの開発における本ツールの適用事例を紹介して、有効性や今後の課題を考察していきます。

（2）業務システム高速生成ツールの概要

提案する業務システム高速生成ツールは、ビジネスプロセス実行エンジン（PEXA Engine：http://www.atrris.com/）とビジネスルール実行エンジン（JBoss Drools：http://www.jboss.org/drools）の 2 つの実行エンジンを組み合わせて動作する業務システムを生成します。本ツールでは、DBMS や Web コンテナなど業務システム（Web アプリケーション）を動作させる環境も自動配備する仕組みを導入しています。

本ツールのインプットは、ビジネスプロセス、ビジネスルール、および画面定義になります。これらのインプットに基づき、業務システムを自動的に生成します。そのため、何度も資材の修正とシステムの試作を繰り返すことが可能です。

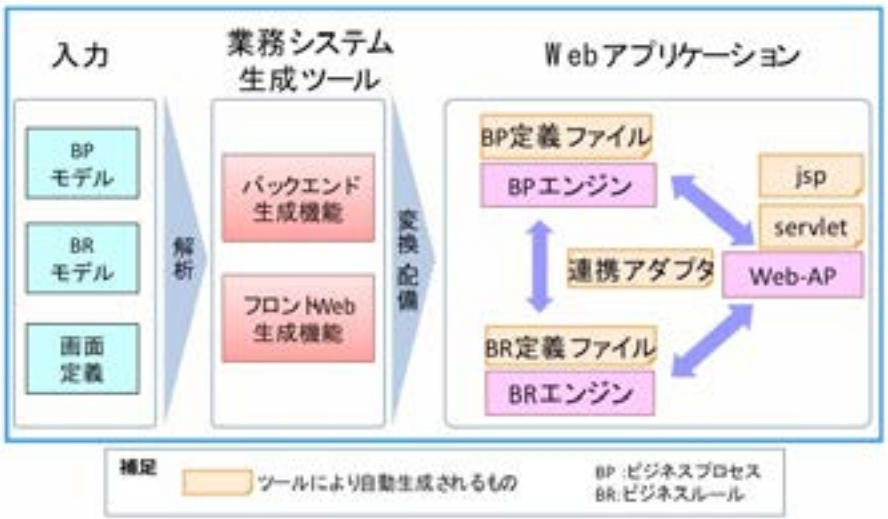


図3.6- 1　ツールの概要図

ツールのフロント Web 生成機能では、ビジネスプロセスから、Servlet を、画面定義から JSP を生成します。バックエンド生成機能では、ビジネスプロセス、ビジネスルールモデルを入力として、エンジンが解釈可能な定義体への変換、ビジネスプロセスごとに、処理の委譲先を振り分けるアダプターの生成、実行エンジン固有の値への型変換を行うラッパーを生成します。さらに、フロント Web 生成機能、バックエンド生成機能により生成された定義体等、実行エンジン、DBMS、Web サーバ、を配備していきます。

1）ツールへのインプット

(a) ビジネスプロセス

ビジネスプロセスを表現する方法としては UML などの図を用いたものや、表形式で表現する方法が存在します。本ツールでは、ビジネスプロセスの定義に表形式を用いる要求分析手法 [Ouchi2011] の記法を参考にカスタマイズをしたモデルを採用しています。ビジネスプロセスは表計算ソフトで記述します。

シートID :40

番号	プロセス名	操作	対象テーブル	対象カラム	ビジネスルール	プロセス遷移先
1	予約を検索する	検索	予約	予約番号		
				申込者名		
				登録従業員名		
				予約日		
		出力	-	予約番号		
				申込者名		
				登録従業員名		
				予約日		[FW]40_2
2	入金ステータスを検索する	検索	入金ステータス	販売額合計	rule_001	
				入金額合計		
		出力	-	入金ステータス		
3	予約を選択する	選択	予約	-		[FW]41_1

図3.6- 2　ビジネスプロセスの記載内容

ビジネスプロセスで記載する要素はプロセスの番号、プロセス名、プロセスでの操作、プロセスで操作するテーブル名およびテーブル内のカラム、ビジネスルールを利用する場合のビジネスルール ID になります。また、プロセスを実行し終了した際に、次に実行されるプロセスの番号を記載しています。例では、 1 番目のプロセスでは、予約テーブルを対象に、検索処理を行うプロセスとして定義されています。「検索」と書かれている行の項目が検索条件になり、「出力」と書かれている行の項目が検索結果の出力項目となります。ビジネスルール列にルール ID が記載されていないため、この処理はビジネスプロセス実行エンジンにて処理されます。また 2 番目のプロセスは、入金ステータスを算出するプロセスが定義されています。「検索」と書かれている行の項目が算出ロジックに渡される引数となります。「出力」と書かれている行の項目が算出結果の出力項目です。ビジネスルール列にルール ID が記載されているため、この処理はビジネスルール実行エンジンにて処理されます。本ツールの入力として、上記のビジネスプロセスを画面単位ごとに作成していきます。

(b) ビジネスルール

ビジネスルールは、本ツールで採用するビジネスルール実行エンジン（JBoss Drools：http://www.jboss.org/drools）の記法に準拠しています。ビジネスプロセスと同様に、ビジネスルールも表計算ソフトで記述します。ビジネスルールはデシジョンテーブル形式となっており、条件と、条件に応じた結果を定義します。提示した例では、条件として、“ 販売額合計 ” と “ 入金額合計 ” 比較を行い、結果に対応するステータスを返却するビジネスルールです。

ルールID:rule_001

入金ステータス	条件	結果
	販売額合計と入金額合計の比較	入金ステータス
	入金額合計=0	未入金
	販売額合計 > 入金額合計 !=0	一部入金
	販売額合計 = 入金額合計	入金済
	販売額合計 < 入金額合計	入金超過

図 3.6- 3　ビジネスルールの記載内容

(c) 画面

本ツールの入力となる画面定義は HTML で作成されたファイルです。HTML ファイルは基本的にビジネスプロセスと対応する形で作成します。ファイル中の画面の項目には、ビジネスプロセスとの入出力を紐付ける記述がされています。

図 3.6- 4　画面定義の例

2）生成されるシステムのアーキテクチャ

業務システムを生成するためには、前述のビジネプロセス、ビジネスルール、画面定義をツールで指定する所定のフォルダに格納し、生成を実行します。

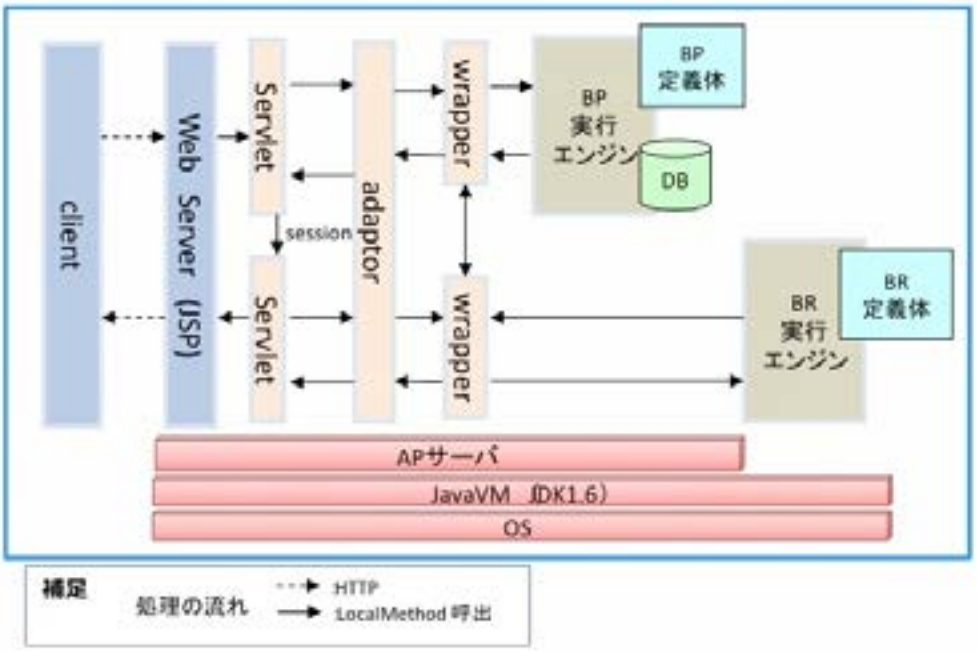


図 3.6- 5　システムのアーキテクチャ

本ツールは２つの実行エンジンのサーバへの配備についてもツールが実行します。なお、本ツールを用いた開発の特徴として、試験モデルからシステムの生成を自動で行うため、単体試験・結合試験は不要になることが挙げられます。そのため、総合テストのみを実施すればよい点があげられます。

（3）適用事例

開発したツールの適用事例を述べていきます。２つの業務システム開発案件（以下、“A 案件 ”、“B 案件 ”）に適用を行いました。ここで取り上げる２つの事例はいずれも新規開発案件であり、繰り返し開発における 1 回目のサイクルです。また､開発者はいずれの案件においても本ツールを用いた開発は初めてでした。ビジネスプロセス、ビジネスルールおよび画面の作成方法、ツールの使用方法について事前に 3 時間程度のレクチャーを実施した後に開発に着手しました。

	A 案件	B 案件
画面数	36 画面	19 画面
機能数	48 機能	12 機能
FP 数	196 FP	87 FP

図３.6- ６　開発規模

１）事例概要

A 案件で開発したシステムは、あるサービス提供型商品の予約を管理するための Web システムです。システムを利用するアクタは、システム管理者、接客担当者、経理担当者の 3 種類です。システム管理者はデータベースのマスタ管理など、システム全体のメンテナンスに関わる業務を行います。接客担当者はサービス受付窓口でシステムを操作し、顧客の好みに合った商品を検索して予約情報を登録する。経理担当者は、接客担当者が登録した予約を顧客ごとにとりまとめ、会計処理などを行います。なお、A 案件では提案ツールを利用せずに、個別に開発を実施した部分もあります。

B 案件では開発したシステムは、社内組織の備品を管理するための Web システムです。システムを利用するアクタは、システム管理者、備品管理担当者の 2 種類です。システム管理者はデータベースのマスタ管理など、システム全体のメンテナンスに関わる業務を行います。備品管理担当者はシステムを操作し、組織内担当の備品情報を登録、削除、更新します。なお、B 案件では個別に開発した機能は存在しません。

２）結果

提案するツールを活用することで、A 案件では 196FP と見積もられたシステムを約 1.5 人月、B 案件では 87FP と見積もられたシステムを約 0.7 人月で開発を完了することができました。これらは、一般的な開発の生産性と比較しても高い生産性であると言えます。

		A 案件	B 案件
工数（人月）		12.6	5.6
資材作成（H）	ビジネスプロセス	99.5	32
	ビジネスルール	8.5	4.5
	画面定義	20	48.5
個別開発（H）		21	—
環境構築（H）		2	7.5
試験工数（H）		105	20
合計（H）		256	112.5
人月換算（人月）		1.51	0.66

図３.6- ７　２案件の実績データ

次に各工程の全体の作業時間における割合を以下に示します。A 案件、B 案件ともに資材作成の時間が 5 割以上を超える結果となっています。また、資材作成の内訳として、A 案件ではビジネスプロセスの作成に時間がかかり、B 案件では画面定義作成に時間かかかっているという結果になっています。

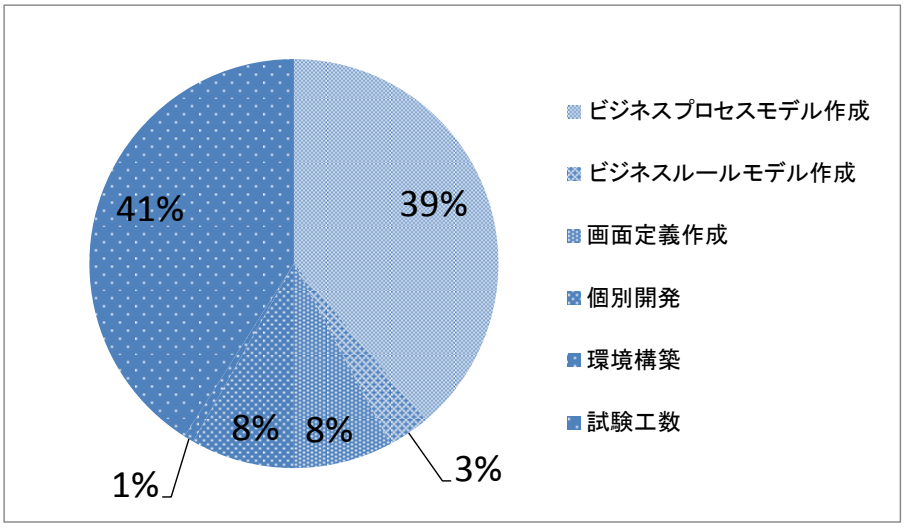


図 3.6- 8　A 案件の各工程の時間割合

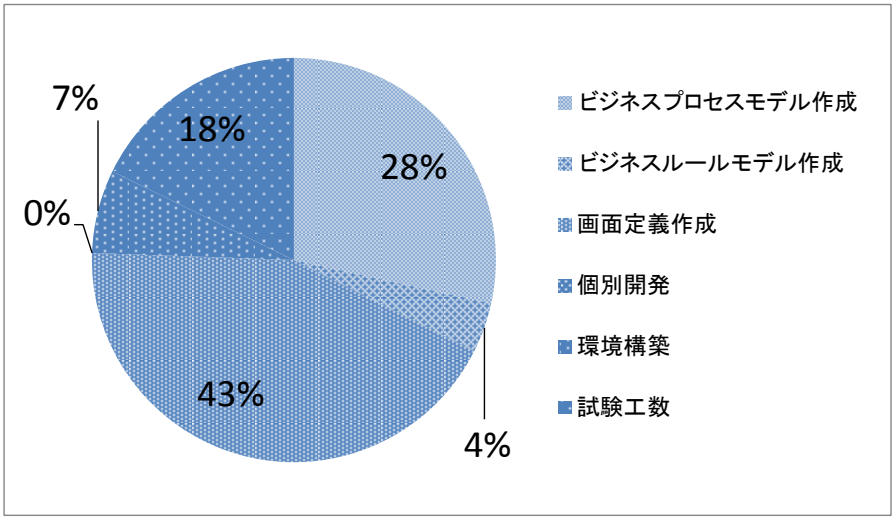


図 3.6- 9　B 案件各工程の時間割合

（4）考察

本ツールを用いた 2 つの案件ではどちらも比較的高い生産性でシステム開発を行うことができました。この理由の 1 つとして、本ツールの入力情報として、モデル（ビジネスプロセス・ビジネスルール）を活用したシステムへの自動変換が寄与していることが挙げられます。モデルからシステムへの自動変換が実現できたことで、可能コーディング工数を削減できることはもちろんですが、単体試験から結合試験までの試験工数が削減することができました。そのため、高い生産性でシステムを構築することができました。

本ツールを用いた開発ではビジネスプロセス、ビジネスルール、そして画面定義の 3 つの入力が必然的に残ります。そのため、開発サイクルの後期に新しいメンバが加わった際にも、明確な業務システムの要件の引き継ぎができました。併せて、ユーザからの要件変更や要件追加のフィードバックに対しても、ビジネスプロセスやビジネスルールの変更、画面の修正で対応ができました。修正した資材を用いて再度ツールを実行することで迅速に対応することができました。

一方で、本ツールにより全ての業務システムが開発できるわけではありません。実際に適用事例においても、ツー

ルは利用できず個別開発（実装）に委ねる部分は存在しました。このように、本ツールが適用できる機能の見極めは非常に重要であると考えられます。また、入力となるビジネスプロセス・ビジネスルール・画面の作成では、ツールの入力として用いるため、厳密な記述が必要になります。そのため、担当者には開発の早期から正確な要件の抽出と定義をすることが求められました。結果として、本ツールはユーザからの要件の確認ツールとしての役割も果たすことができたと言えます。

（5）まとめと今後の課題

本稿では、業務システム高速生成ツールを提案しました。提案ツールの適用事例を紹介し、ツールの有効性と課題について考察しました。一方で、提案したツールはシステム開発の現場での適用実績が少なく、適用の正式な手順なども十分に検討しきれていません。そのため、現状はツールを用いた開発の生産性は開発担当者のスキルや経験に依存することが多いといえます。今後検討を重ねて、本ツールにおける開発手順や、ツールで作成した業務システムへのユーザからのフィードバックの反映方法などの中期以降のサイクルでの活用方法を検討していく予定であります。

本ツールでは、入力となるビジネスプロセス、ビジネスルール、画面定義の準備に未だ手間がかかるという課題があります。要件変更や要件追加が発生した場合に、モデルや画面定義の改修を簡易化するための支援ツールの検討も必要となります。また、本ツールで作成する業務システムのアーキテクチャは固定となります。このため、開発サイクルの後期では性能要件や可溶性など非機能要件にそぐわなくなる可能性もあります。その場合、ツールで作成したシステムに改修を加えていくことが考えられますが、その手順や拡張対応範囲などは改めて検討する必要もあります。

参考文献

[Ouchi2011] 大内隆信、森下大介、丹郁夫、寺町康昌、鶴保征城 “ 帳票の取り回しによる業務システムに最適化したユースケースとそれに適合した業務フロー図による要求分析手法 ” 情報処理学会研究報告 IPSJ SIG Technical Report Vol.2010-SE-171 No.31 2011/3/15

〔斎藤忍、萩原淳 著〕

3.7　ルール定義型システム自動生成ツールと Biz/Browser 連携について

ノンプログラミングでシステムを作る試みは、30 年ほど前からさまざまに試みられてきましたが、なかなか成功には至りませんでした。その大きな要因は次の 2 つだと思います。

① ユーザ企業に自らのシステムは自ら作るという機運があった。

② 自動生成ツールの UI が稚拙で利用部門に受け入れられなかった。

しかしながら昨今、BRMS と呼ばれるルール定義型システム自動生成ツールが急速に普及し始め、ノンプログラミングでシステムが作れる時代となりました。その要因は次の 2 つと思います。

① 企業の要求が、安く・早くということが重要視されるようになった。

② Web の時代となり、HTML ベースでは以前のように画面操作性にこだわることができなくなり、操作性が低下しても仕方がないと考える人が増えた。

こういったことから BRMS がもてはやされるようになってきました。しかしながら、そうは言ってもシビアな操作性やレスポンスを、たとえ Web システムであったとしても、それとは関係なしに求められる業務は多々あります。そういった場合はどうするかです。つまり以下のような要求があった場合です。

① 早く・安く上げたい。

② 画面操作性は自由に指定したい。

③ レスポンスは 1 秒程度。

④ 電波の届かないところでも仕事ができないと困る。

④ 開発後社内要員でメンテナンスをしたい。

こういったことに応えられる技術がリッチクライアントとか RIA と呼ばれる技術です。その中で日本で最も歴史がありユーザ数も多い Biz/Browser をとりあげて、BRMS との連携方法を WG1 の会合では論じました。

（片貝 ,「ルールとして定義することでシステムが自動的にできる BRMS と、個々人の使い勝手との乖離をどう乗り越えるか」, WG1 会合内部資料：WG1-6-1.pdf, 2012.7.10）

〔片貝 著〕

3.8　XupperII（クロスアッパーⅡ）

（1）上流工程の重要性

システム開発プロジェクトの 70％近くが失敗しています。その最大の理由の 1 つは要件定義などの上流工程でユーザ部門と開発担当者との間で意思の疎通を欠いているからと言われます。上流工程の不具合が下流工程やテスト工程で発見されるため大きな手戻りが発生し、いわゆる手戻り工数が開発工数全体の 50％近くを占めるとの報告もあります（日経コンピュータ 1998 年 9 月 28 日号より）。

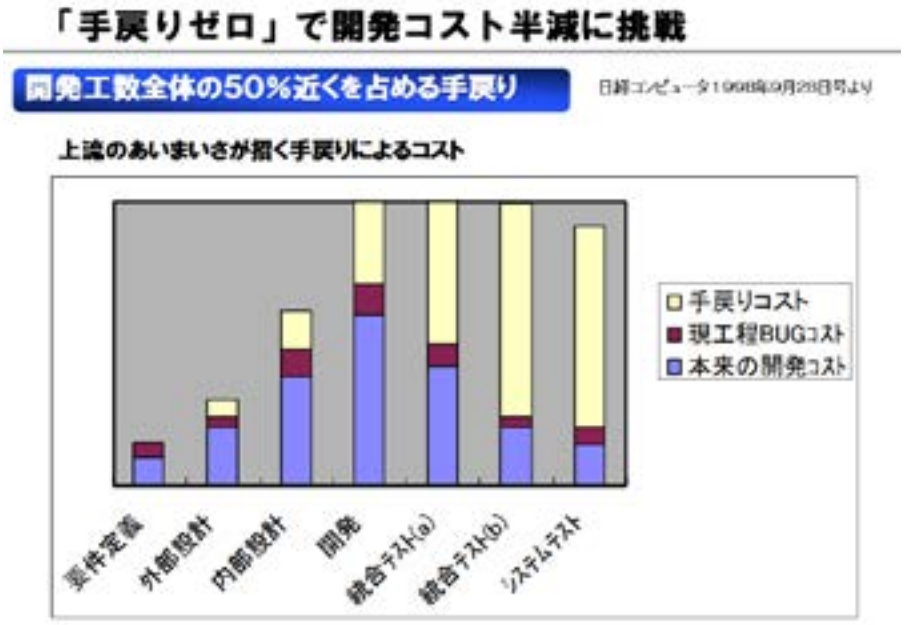


図3.8- 1　手戻りゼロでコスト半減

逆に言うと、システム開発の成功のポイントは

①上流工程で業務要件を、早期に確立する。また、業務要件の変更に対してすばやく対応する仕組みをつくる（設計情報の一元管理）

②エンドユーザに分かりやすいドキュメントを使用して、システム開発者とエンドユーザのコミュニケーションギャップを無くす

③データモデルを正しく設計して、データ統制を行う

④設計情報を下流工程へ正しく伝える（下流工程への連携性）

にあります。つまり、ユーザが参画する開発体制を確立することが不可欠で、現実の業務とユーザの意図を正しく設計に反映させることがカギになります。とりわけ上流の要件定義で、ユーザがきちんと理解できるビジネスルール定義、業務フロー図、ユーザ・インタフェース（入出力設計）を作成することが重要となります。

上流工程の不整合をその工程で発見すればすぐに修正できるところが、不整合を見過ごしてしまうと、設計工程、製造工程、試験工程で発見するためには、それぞれ、5 倍、10 倍、50 倍の工数がかかると言われています（IEEE,1996 資料より）。 上流工程の不整合をいかに早く発見し、その工程内で解決し下流工程への上流設計 バグ混入を未然に防止することが重要です。

（2）データモデルの重要性

先に述べましたように、ユーザ要求を正しく反映して要件定義・基本設計の上流工程を行うことが重要ですが、その結果を正しく設計図として表現することが更に大切です。

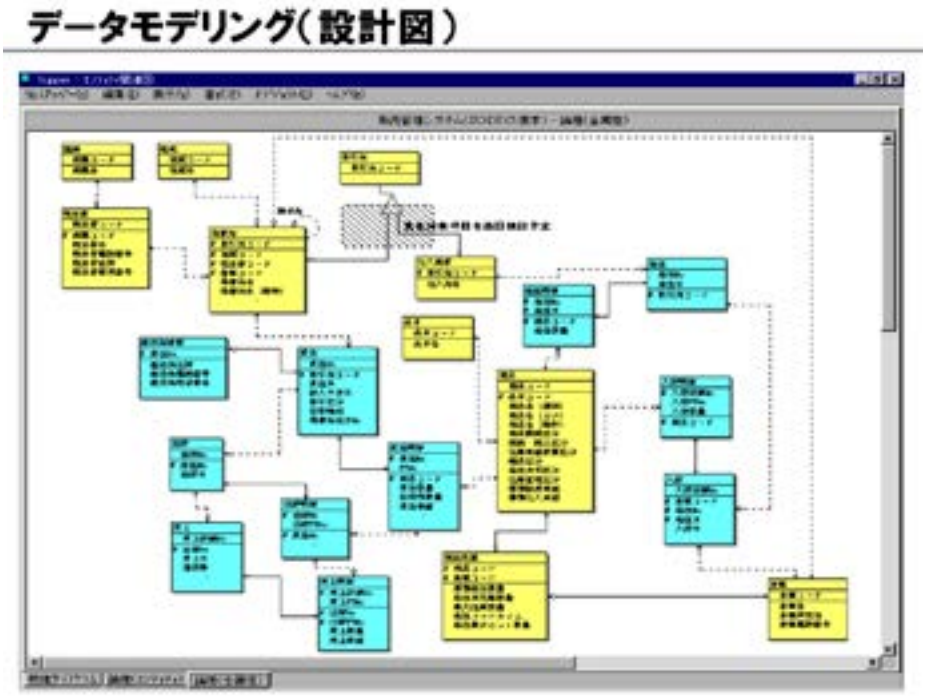


図3.8- 2　データモデリング

情報システムの 2 大構成要素はプロセス（処理）とデータですが、このうちのデータを把握することで業務ルールの本質を知ることができます。プロセスはユーザ要求に伴って変化していきますが、データはビジネス形態が変わらない限り、大きく変わりません。

業務要件からデータを正しく 抽出し、データとデータの関連、どこでどのようなデータが管理 されているかを端的に表現できるものがデータモデルです。

データモデルは建築でたとえれば土台や骨格にあたるもので､これがしっかりしていれば､少々の仕様変更があってもびくともしません。ビジネスを正しく反映した データモデルが作成できれば、60%以上は成功したと言っても過言ではありません。

以下に、データモデルの主な効果を記します。

・プロセスよりもデータの方が客観的に業務ルールを把握できる

・プロセスよりもデータの方が変化に強い

・分析手法は人によって異なっていても、最終のアウトプットは同じとなり人による違いが出にくい

（3）リポジトリの重要性

さらに重要なことは、ユーザからの要求変更や要件変更に対して、素早く対応をとれる仕組みを取り入れることです。ユーザからの変更依頼に対して、システムのどこにどのような影響を与えるかの分析が簡単に行えることが重要です。

そのためには、開発の全工程における設計情報をリポジトリで一元管理することが必須であり、その結果、仕様変更に対する影響分析を正しく行い、素早く対応することが可能となります。

リポジトリによる一元管理を行うことにより、データ統制が可能となり、要件定義から基本設計・詳細設計さらには製造工程に至るまで、データ項目の整合性を保つことが可能となります。

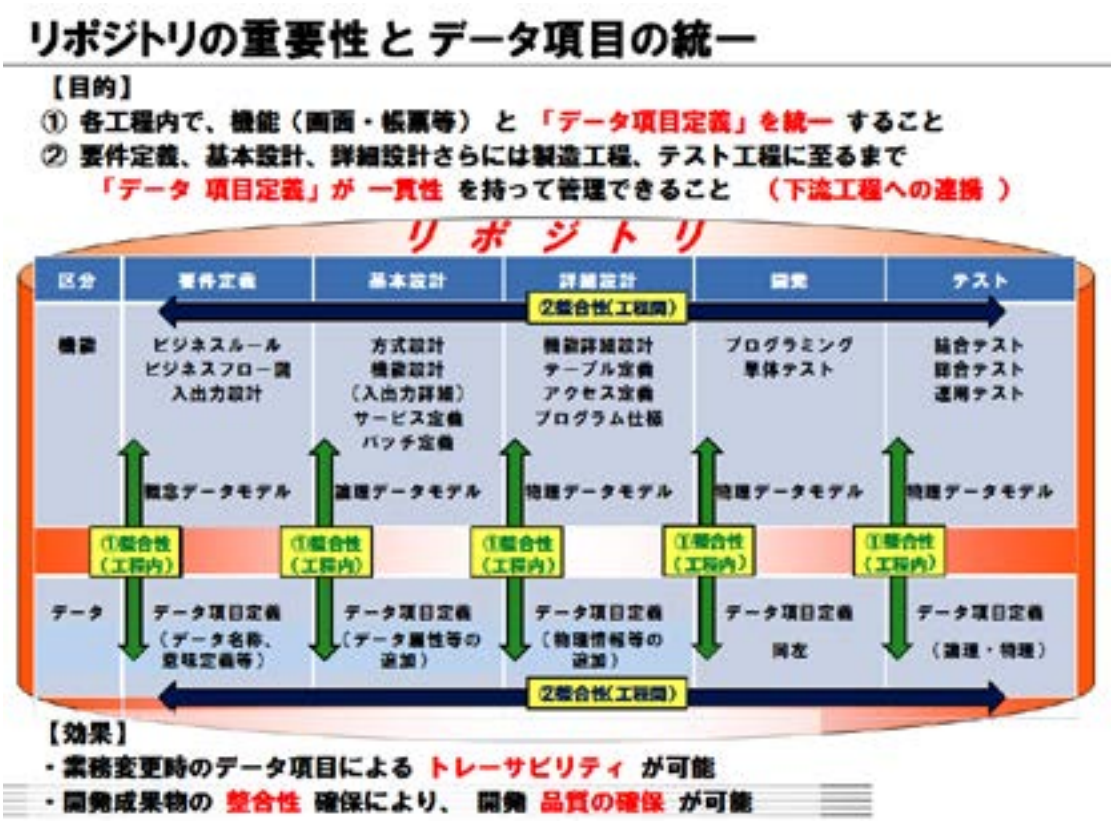


図3.8- 3　リポジトリの重要性

（4）XupperII による解決

上流工程のポイントは、

①ユーザ要求を正しく反映したデータモデルを作成することであり、そのためには

②ユーザとシステム開発者のコミュニケーションを正しく取る、

③ビジネスを正しく反映したデータモデルを設計する、そして、

④これらの設計情報をリポジトリで一元管理する

ことです。

XupperII は、ユーザ部門とシステム開発部門との間に位置しており、業務を可視化することで経営者やユーザ部門にも分かりやすく、システム部門とのコミュニケーションをきちんと取ることができ、結果として現実のビジネスを正しく反映したデータモデル（設計図）を作成することができます。 そして、その最大の特徴は、設計情報が「リポジトリ」としてデータベースで一元管理されているため、仕様変更による各種影響分析が可能となることです。

さらに、MDFrame/X と連携することで、Java、.NET（C#）、Biz/Browser のスクリプト等の自動生成も可能となり、また、テスト連携オプションにより自動テストツールとの連携も可能となります。 このように上流工程の設計情報から下流の製造工程やテスト工程までの設計情報の整合性を取り一元管理をすることが可能です。

これらの機能により、仕様変更が発生した場合も、上流工程だけでなく、下流工程やテスト工程に至るまでの影響分析が可能となり、システム開発のトータルな生産性向上と品質向上に大きく寄与することが可能になります。複数の開発者で同時に使用することも可能なので、ソフトウェアの開発規模が大きくなればなるほど、さらに大きな効果を発揮することができます。

（5）XupperII の機能概要

XupperII の主な機能は以下の通りです。

・ビジネスフロー図（業務フロー図）

・ビジネスルール

・入出力設計（外部設計）

・エンティティ設計

・ディクショナリ

・マトリクス分析

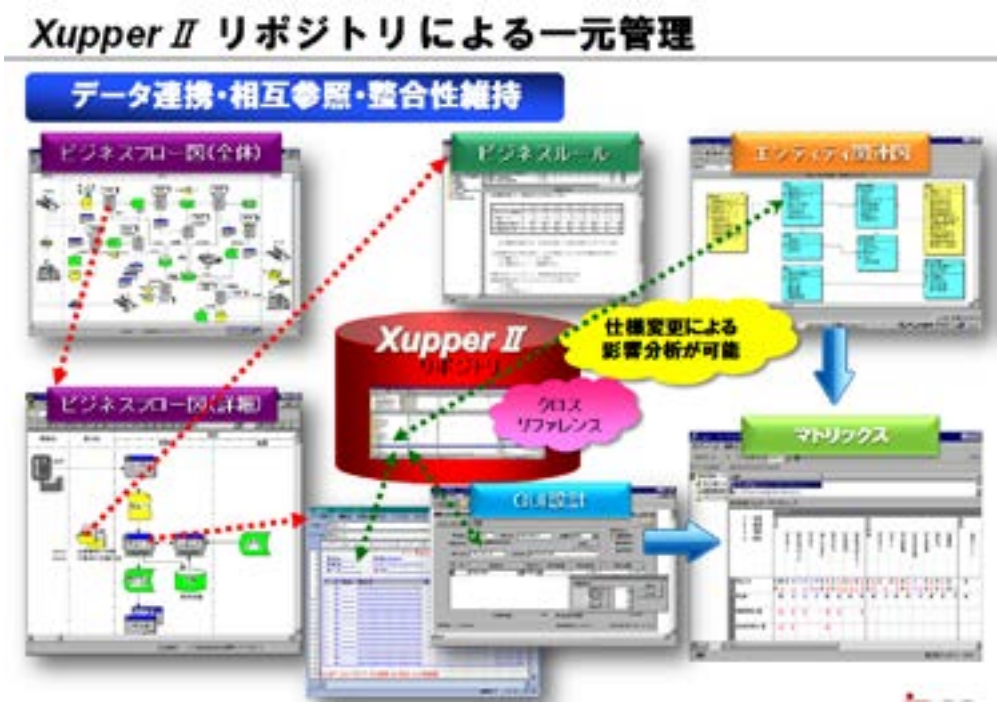


図３.8- ４　XupperII リポジトリによる一元管理

また、これら全ての設計情報がリポジトリで一元管理されており各種クロスリファレンスや影響分析機能も充実しています。

それではそれぞれの機能の主な使い方をご説明します。

1) ビジネスフロー図、ビジネスルール

システム開発の上流工程で重要なのは、顧客企業のニーズをいかに正しくくみ取るかということです。これを実現するための XupperII の大きな機能が「ビジネスフロー図」と「ビジネスルール」です。

「ビジネスフロー図」は、業務を遂行する上で必要となる工程と、その目的、実施手順、実施要領など、ビジネス全体の運用を表わすものです。新システムの稼動時に、人とシステムがスムーズに連携し、イメージどおりに運用できるかどうか、エンドユーザと共に事前検証していくことができます。

「ビジネスルール」は、必要に応じて XupperII から WORD,EXCEL 等の各種 ドキュメントを利用するための機能です。ユーザはビジネス上の取り決め事項を、SE はシステム全体の共通認識事項を整理します。さらに、XupperII の各種機能から ビジネスルールを共通ルール部品として参照することも可能です。

2) 入出力設計（外部設計）

こうして固めた業務仕様をもとに、画面や帳票などの外部設計に入ります。ビジネスフロー図で洗い出された機能（プロセス）から直接外部設計の画面に飛ぶことができ入出力項目の設計を行います。ここで大事なことは、外部設計で使用するデータ項目の名称や属性等を明確にすることです。そして、これらのデータ項目はディクショナリ機能で一元管理されます。

3) エンティティ関連図

一連の設計作業を通じて、固まってくるのが、情報システム上で企業が持つデータのあり方です。XupperII を用いて要件定義や外部設計を行っていくと、システム上で管理すべきデータが抽出されます。画面設計を行いながらデータを整理することで、正しいデータモデリングが可能となります。「エンティティ関連図」機能により、論理的なデータの関連図と、物理的なデータ関連図の双方ともに把握することができます。

4) マトリクス分析

各業務プロセスとエンティティとの相互作用を分析設計する CRUD 図を作成し、データのライフサイクル（発生、参照、変更、削除）を定義することができます。

5) ディクショナリ

入出力設計で登録した画面／帳票のデータ項目やエンティティ関連図において登録したデータ項目などは、すべて「ディクショナリ」で一元管理され、整合性が取れています。このため、ある項目がどこで利用されているかをクロスリファレンスとして検索でき、同様のデータ項目の重複を排除できます。またデータ項目についての変更要件が発生した場合にも、その項目に関連する箇所を正確かつ速やかに洗い出すことができ、修正漏れなどを確実に防止できます。

6) リポジトリ

こうしたことができるのは、XupperII 自体がシステム開発に必要な設計情報を「リポジトリ」（データベース）で一元管理しているためです。このためシステム設計の各機能が有機的に連動することができます。

一般的に、上流工程での要件分析・要件定義のドキュメントは、オフィス系のドキュメントツールで作成しているのがほとんどですが、これらのツールで作成していると仕様変更が発生した場合にドキュメント間の整合性をとることが不可能です。

XupperII では、全ての設計情報が「リポジトリ」で一元管理されているため、仕様変更が発生しても、どこに影響があるかのインパクト分析が非常に簡単にできます。さらに、変更対象の設計情報を修正すれば自動的に関連する部分がすべて更新され、設計情報の整合性をとることができます。

（6）XupperII による開発手法

XupperII を利用することでユーザに分かりやすく、SE に使いやすい方法論で要件定義や設計を実施することができます。

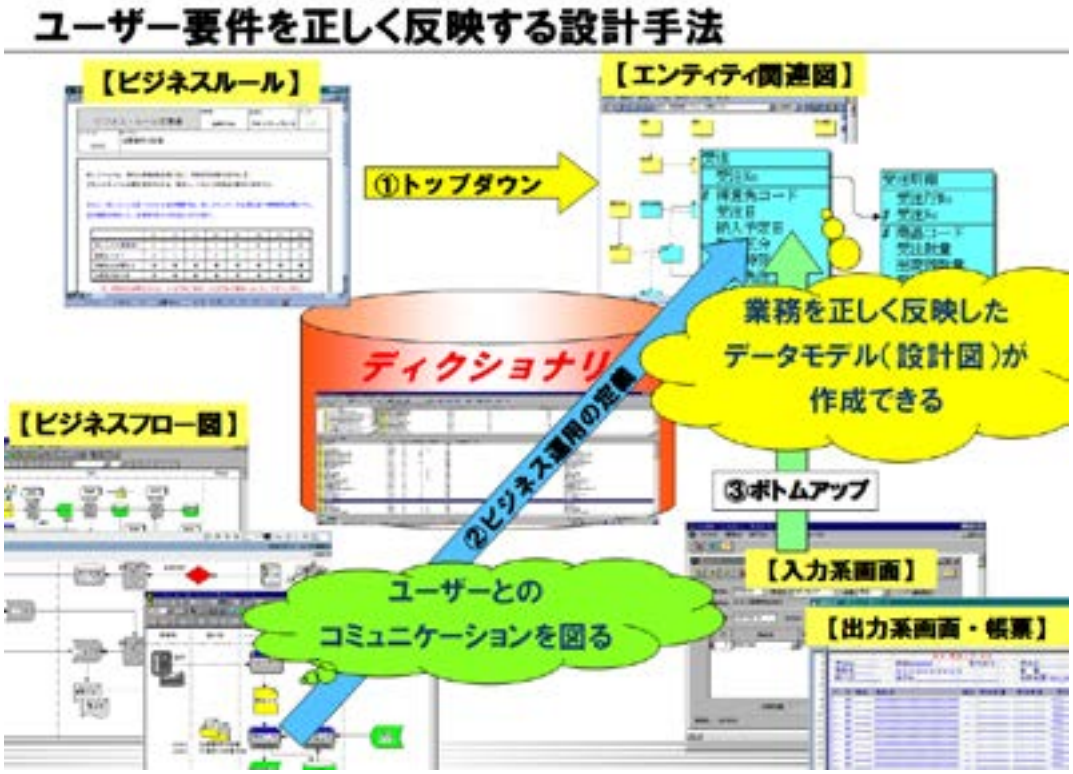


図3.8- 5　XupperII による開発手法

①トップダウン分析

・ビジネスルールをベースにユーザヒアリングを行い、リソース系エンティティを抽出する

②ビジネス運用の定義

・業務フローをベースにユーザ部門とのコミュニケーションを図り、イベント系エンティティを抽出する

③ボトムアップ分析

・入力系の機能を分析して、入力画面設計を行い、データ項目を抽出して整理する

・抽出されたデータ項目をエンティティにマッピングすることで、データモデルを完成に近づける

・帳票系のデータ分析を行うことで、データモデルの検証を行う

上記①～③を繰り返すことにより、業務を正しく反映したデータモデルを完成させることができます。

（7）MDFrame/X の機能紹介

MDFrame/X では、XupperII で登録されたディクショナリ、エンティティ、GUI の設計情報を流用し、業務仕様および画面の振舞いに対する物理設計や SQL の定義を行うのみでアプリケーションプログラムを自動生成することが可能です。XupperII のリポジトリを下流領域まで拡張し、上流から下流までの設計情報を一元管理します。

XupperII の設計情報と MDFrame/X の定義情報から、Java 、.NET（C#）および BizBrowser のプログラムを自動生成します。これにより、設計情報とソースプログラムの乖離がなくなり、生産性の向上、品質向上さらには保守工数の削減に大きく貢献することができます。

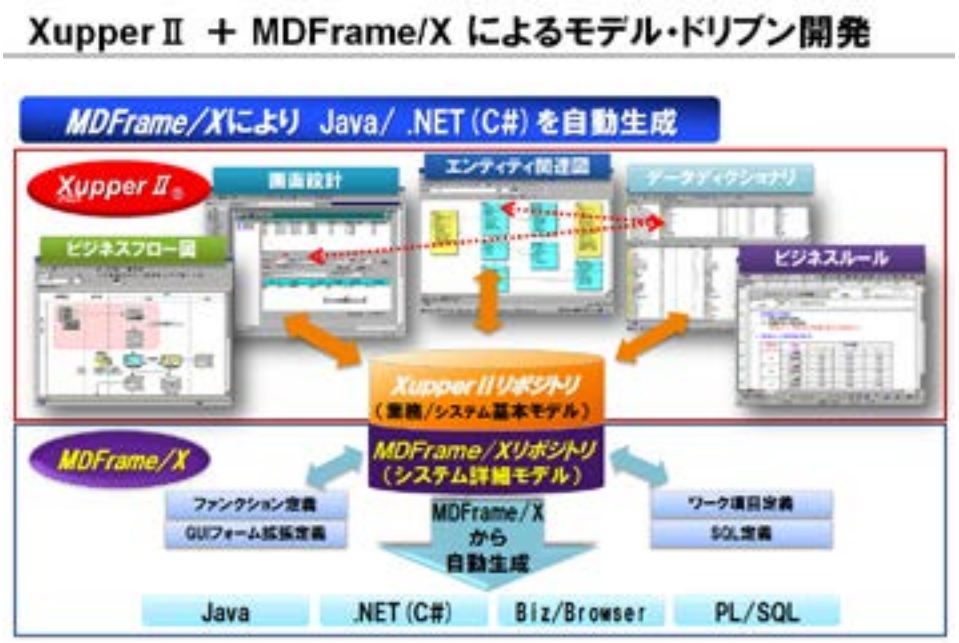


図３.8- 6　XupperII ＋ MDFrame/X による開発

（8）XupperII の導入効果

XupperII を導入することにより、リポジトリによる設計情報の一元管理が可能となり、各種影響分析を行うことができ、システム開発の生産性向上、品質向上に寄与することができます。主な導入効果を以下に記載します。

・エンドユーザが理解できる表現で分析・設計を進めていくことによる手戻りの減少

・共通業務モデル（リポジトリコントロール）による整合性の維持が可能となり、マトリックス機能やクロスリファレンス機能により変更・訂正による反映漏れによる手戻りが減少

XupperII はシステム開発フェーズだけでなく、保守フェーズに入ってからも威力を発揮します。システム稼働後にもビジネスの変化に応じてシステム改変が発生しますが、この時威力を発揮するのがクロスリファレンスやシステム差分検査機能です。任意のバージョンの設計リポジトリ同士の差分をマッチングして検出する機能もあり、バージョン管理にも非常に役立ちます。

今までの導入実績から、XupperII で設計すれば、開発コストを 20 ～ 40%は確実に減らすことができ、保守フェーズに至っては 50%以上の工数削減に貢献します。

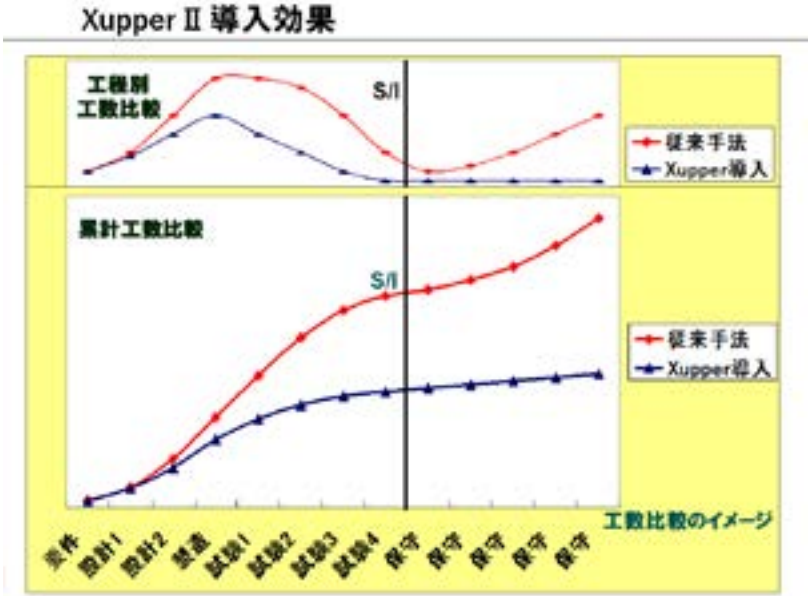


図３.8- 7　XupperII の導入効果

Xupper の導入ユーザとしては大手 SI ベンダー、大企業の情報システム子会社さらにはパッケージメーカーも多く含まれ、最近では、金融機関からの引き合いが非常に多くなっています。特に社会的な影響が大きく品質を重視するお客様のシステム開発に最適なツールです。

（9）まとめ

世界の経済情勢の激変により、企業を取り巻くビジネス環境も大きく変化しています。ビジネスの変化に柔軟に対応するためには、業務プロセスを可視化し、PDCA サイクルを継続させる体制を定着させる必要があり、そのためにも、IT システムを迅速に対応させることが必要です。

ビジネスモデルの変化に IT システムを迅速に対応させるためには、ユーザ要求とシステム化要件・システム設計情報とを正しく対応付け、ユーザ要求によるシステムへの影響分析をきちんと行い、システム開発に反映させることが重要です。

業務プロセスを正しく反映したデータモデルを設計し、その設計情報をリポジトリで一元管理し、下流工程へ連携することによる全工程でのデータ統制をきちんと行うことが重要です。その結果、システム開発および保守における品質・生産性を大幅に向上させることが可能となります。

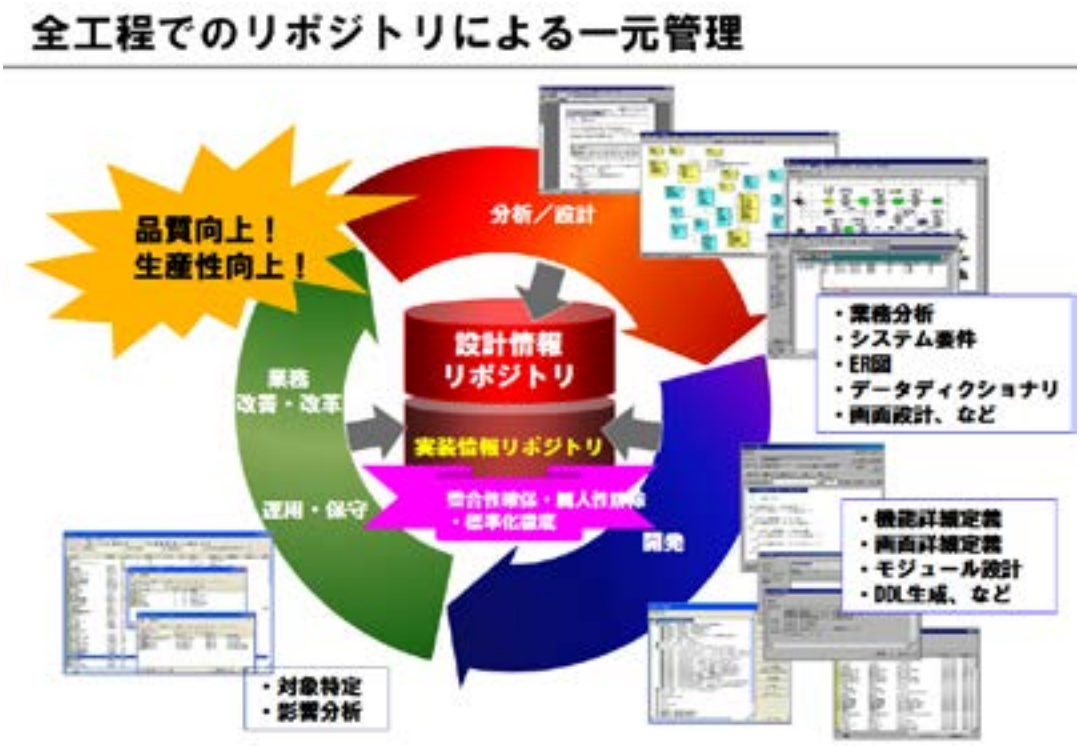


図３.8- ８　全工程でのリポジトリによる一元管理

〔高橋 著〕

3.9　Wagby

2012 年 9 月 4 日の第 7 回 WG1 定例会議、2012 年 11 月 19 日のオープンフォーラムにおける株式会社ジャスミンソフトの贄良則氏のご講演資料のまとめとなります。

（1）Wagby（ワグビィ）とは

外部設計書相当の情報を記述した Excel ファイル（以下、Wagby 定義ファイル）を入力として、Web アプリケーションを実現する自動生成ツールです。

Wagby 定義ファイルは、データオリエテッドアプローチでテーブルを定義して、その定義にロジックやユーザインタフェース、権限設定を付加することで作成します。

自動生成により、Java、JavaScript、HTML といった Web アプリケーションを構成する必要なソースコードを生成します。

プログラム言語を知らなくても開発ができることを目的としています。

2006 年から市場導入して、現在のユーザは 200、ライセンスは 400 個程度となります。

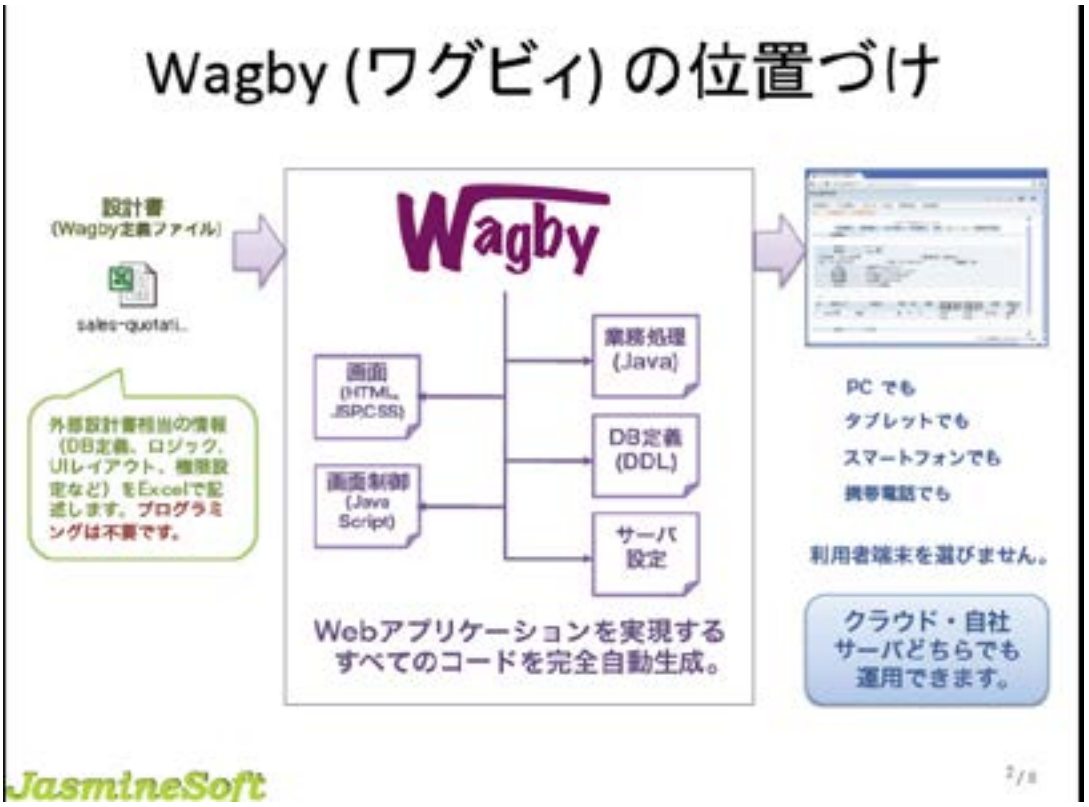


図 3.9-1　Wagby とは

（2）Wagby の歴史

2001 年 ジャスミンソフト設立
2004 年 Web アプリケーション自動生成ツールとして「JasmineSoftHarvest」を発表（後の Wagby の原型）
2005 年「第 17 回中小企業優秀新技術・新製品賞」においてソフトウェア部門の優良賞を受賞
2006 年 JasmineSoftHarvest の後継製品である「Wagby (ワグビィ)」を発表
2010 年 スマートフォン対応業務アプリケーションを自動生成し、クラウドコンピュータ上で運用する機能を備えた Wagby R6.6 を発表
2011 年 沖縄県 IT ビジネスアワードを受賞

（3）対象ユーザ

エンドユーザ、SIer が主たる利用者です。最近は情報システム部門も使うようになってきています。

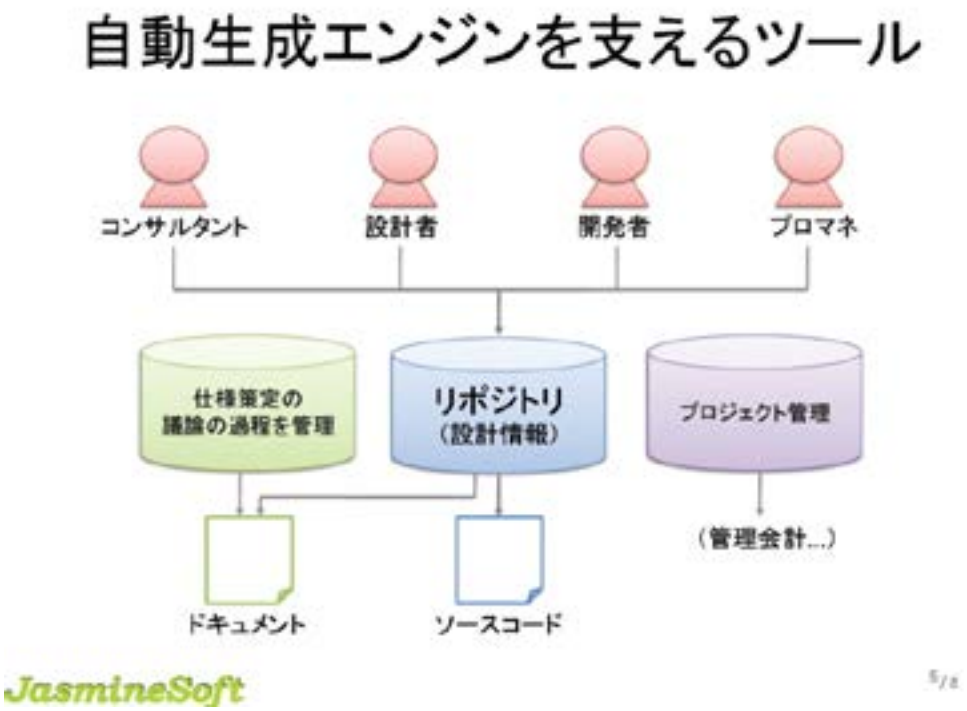


図 3.9-2　対象ユーザ

（4）開発方法論

ユーザ主導による仕様書作成と、プログラムの自動生成技術を組み合わせた開発手法です。「超高速ウォータフォール」として、設計～製造～試験のサイクルを短くして全体を少しずつブラッシュアップする漆塗り工法を意識した方法です。開発生産性 10 倍とうたっています。

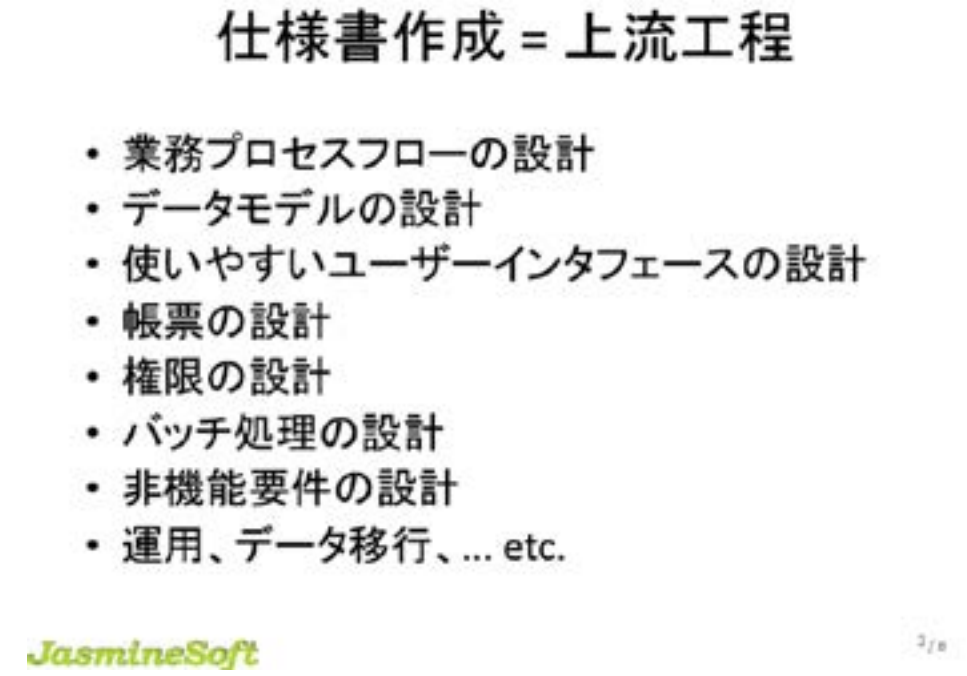


図 3.9-3　開発方法

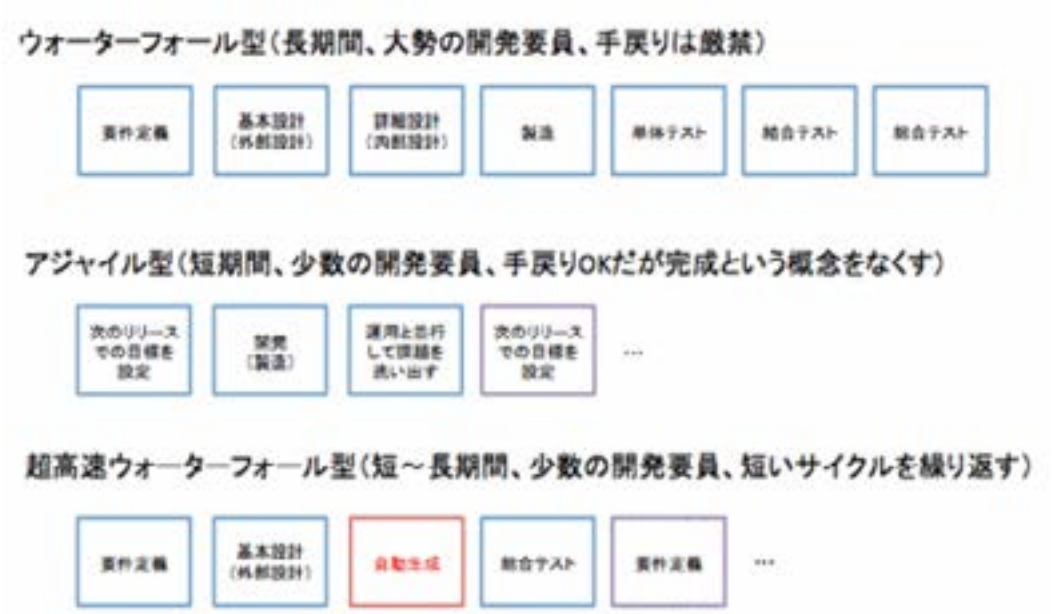


図 3.9-4　超高速ウォータフォール型

プロジェクトの大半は「設計」業務になるとして、仕様書の確定に時間をかけ、製造工程は簡単で、単体・結合テスト工程が無く、総合テストで確認をする方法を推奨しています。

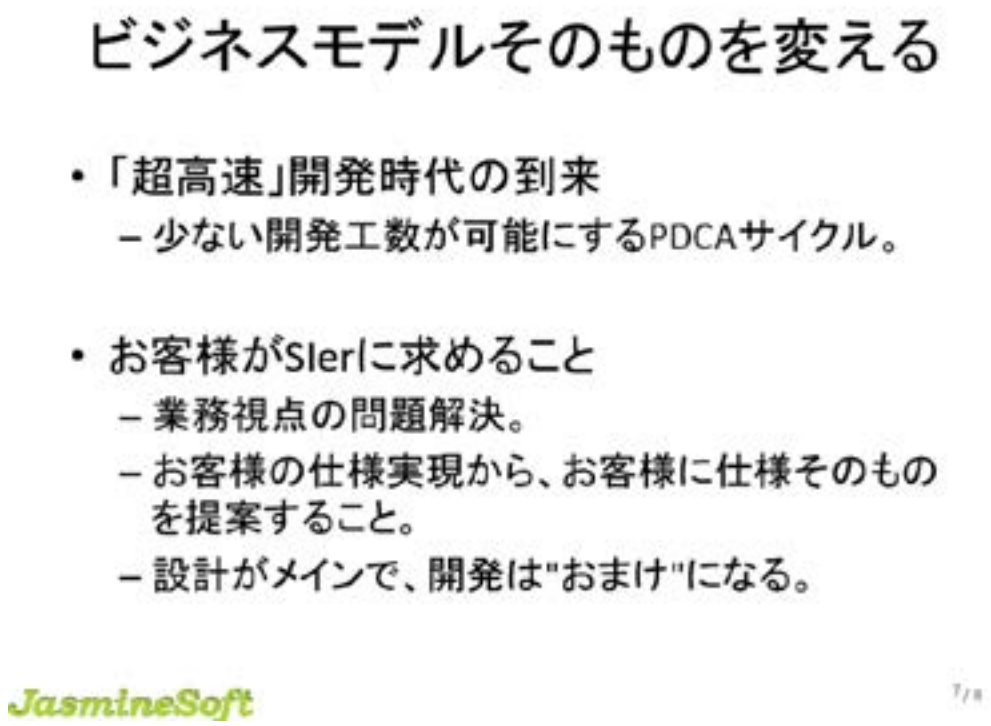


図 3.9-5　Wagby の考え方

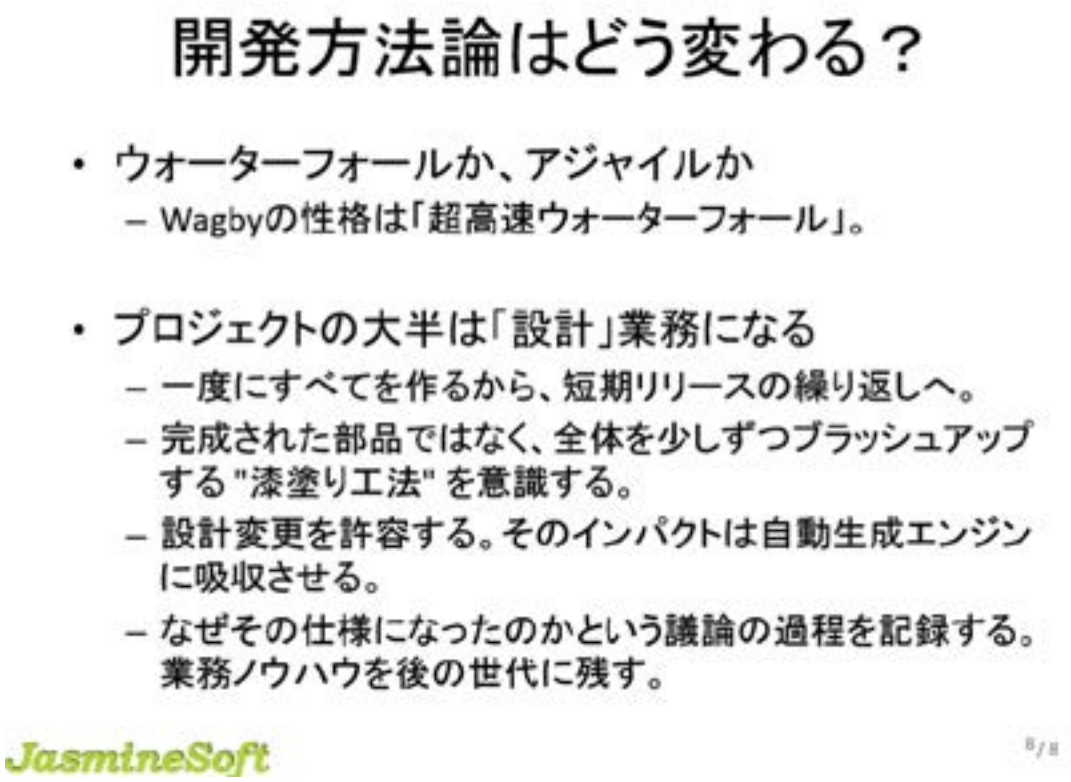


図 3.9-6　Wagby の開発方法論

（5）得手不得手

Web をベースとした業務アプリケーション開発用であり、組込系分野やバッチ処理は適用できません。過去に Wagby でも販売管理ビジネス用のテンプレートを考えましたが、日本ではお客様によってすべてやり方が違うので、この種のパッケージソフトの横展開は難しいとの結論となりました。

むしろ、Wagby のようなツールがあれば、お客様はテンプレートのカスタマイズではなく、好んでゼロから完全に自社システムを設計する方を選択されることが多い傾向にあります。

Wagby でテストシナリオの自動生成は難しいです。これは業務内容がわからないとシナリオ作成はできないと考えるためです。Wagby を使った開発では、この部分はお客様が行うことが良いとしています。

上流（要件定義）ができていれば、データモデルを作りテーブルの構造を決めていきますが、お客様の要件を Wagby でこのように書くという翻訳作業を行うための WagbySE が必要です。WagbySE は、ER 図などで処理内容を具体化する作業も行います。

Wagby は分散開発も可能ですが、これまでの実績では多くて 10 名程度です。

Wagby は新規案件開発用であり、既存 AP との連携は簡単にはいきません。既存テーブル定義を Wagby 定義ファイルで表現できるかどうかがカギになります。

〔中山 著〕

3.10　要求工学知識体系

～ OpenForum2012　名古屋大学 山本教授ご講演より～

本節では、OpenForum2012 における名古屋大学山本修一郎教授の基調講演「要求工学知識体系－顧客価値創造からの展望－」の内容のサマリを【ご講演内容】、参加者とのディスカッション内容を【参加者からの質問･意見】、WG1 メンバーによる所感を【所感】に記述します。

【ご講演内容】

要求工学国際会議においても話題となっていますが、最近は顧客価値創造のために要求を新たに創り出すことが重要になってきています。また、要求は確実には定義できないということ、システム運用を知らないと要求を正しく定義できないことが重視されてきています。これを踏まえて、最新の要求工学知識体系を紹介します。

（1）要求工学の動向

要求工学の歴史を振り返ると、1990 年代には要求工学もプロセスで考えるという時代でした。しかし 2000 年代に入り、システムを作る目的が重視され、最近では顧客価値を提供するためにビジネスと IT を整合させることが重要になってきています。そのため、要求工学もイノベーション、要求を新たに創造することが重要になってきました。

2012 年の要求工学国際会議におけるキーワードとして「不確実性」が重視されており、要求仕様は完全に定義できることはないことを前提として要求工学の研究を進めることが求められています。さらに重要なのは「実行時環境」であり、要求を正しく定義するにはシステム運用時のオペレーションを深く知る必要があります。

（2）ISO/IEC/IEEE 29148:2011

国際標準 ISO/IEC/IEEE 29148:2011 では、新しいシステムソフトウェアエンジニアリングのライフサイクルプロセスを定義しており、ステークホルダ、システムという外部の制約条件の結果としてソフトウェアに対する要求が決まるという考え方を示しています。ステークホルダ要求には、顧客がシステム、サービスをどのように使用しているかを理解するための運用要求が含まれており、運用品質（SLA）を明確にし、運用品質を達成できる証拠を提供する必要があると述べています。

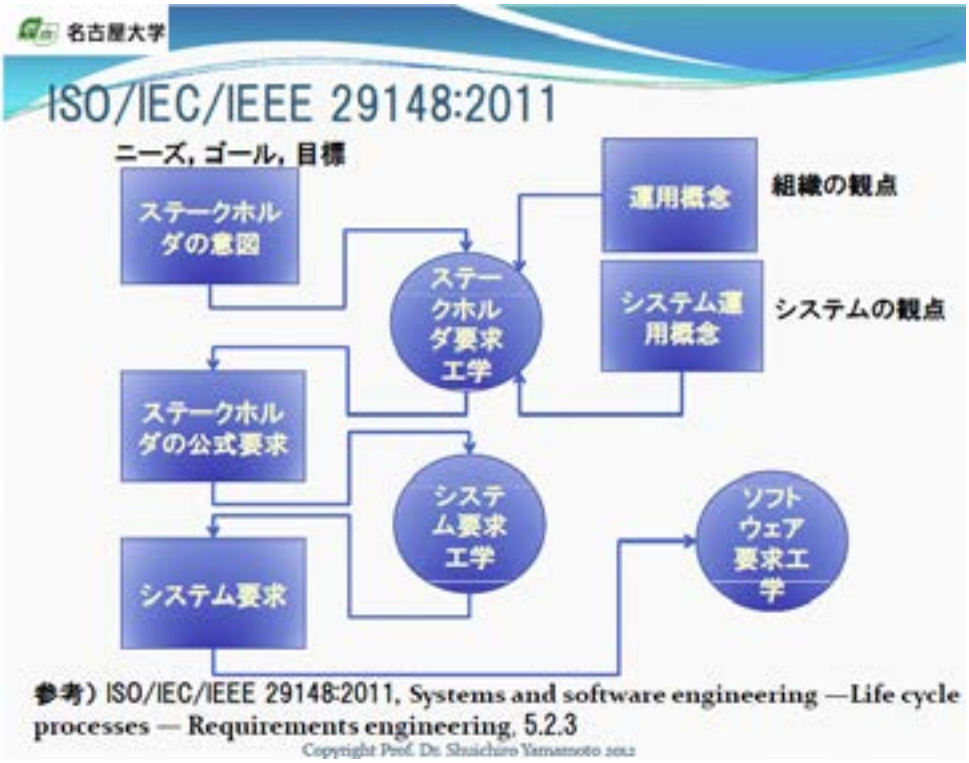


図 3.10-1　ISO/IEC/IEEE 29148:2011（山本教授ご講演資料抜粋）

（3）BABOK と REBOK

BABOK はカナダの IIBA が発行したビジネスアナリストのための知識体系であり、知識領域／タスク／タスクに対する技法が定義されています。北米では、ビジネスアナリストの概念が確立しているが、日本ではまだ役割が確立されていないという現状があります。

一方、REBOK は日本の情報サービス産業協会（JISA）が発行した、要求開発に関与するすべてのステークホルダ（要求アナリスト、エンドユーザ、経営者、など）が習得すべき範囲と水準を整理した知識体系です。

BABOK と REBOK を比較すると、BABOK の方がソリューション評価やテスト専門家を明確に定義しており、システムの妥当性確認を重視しています。また、BABOK におけるビジネスアナリストは PM ではなく、PM と顧客をつなぐコミュニケータです。それに対して、REBOK の要求エンジニアは、プロジェクトの中で活動するという違いがあります。

（4）EA（エンタープライズアーキテクチャ）–TOGAF

EA（エンタープライズアーキテクチャ）は、企業システム全体を一つのアーキテクチャで構成するという考え方であり、代表的なものとして TOGAF があります。TOGAF は EA を開発するための方法論として ADM（アーキテクチャ開発方法）を持っているが、この中で要件管理は重要とされています。要件に対してアーキテクチャのコンポーネントが妥当であるかということを、プロセスの最初から最後まで常に評価します。また、システム運用開始後のモニタリングで、システムが目標を達成してビジネス価値を生み出しているか評価することが重要視されています。要求仕様には、アーキテクチャ選定の判断をしたときに、その正当性の保証をどのようにするかを含んでいます。要求工学で最近話題となっている多くのことが、TOGAF の中で整理されています。

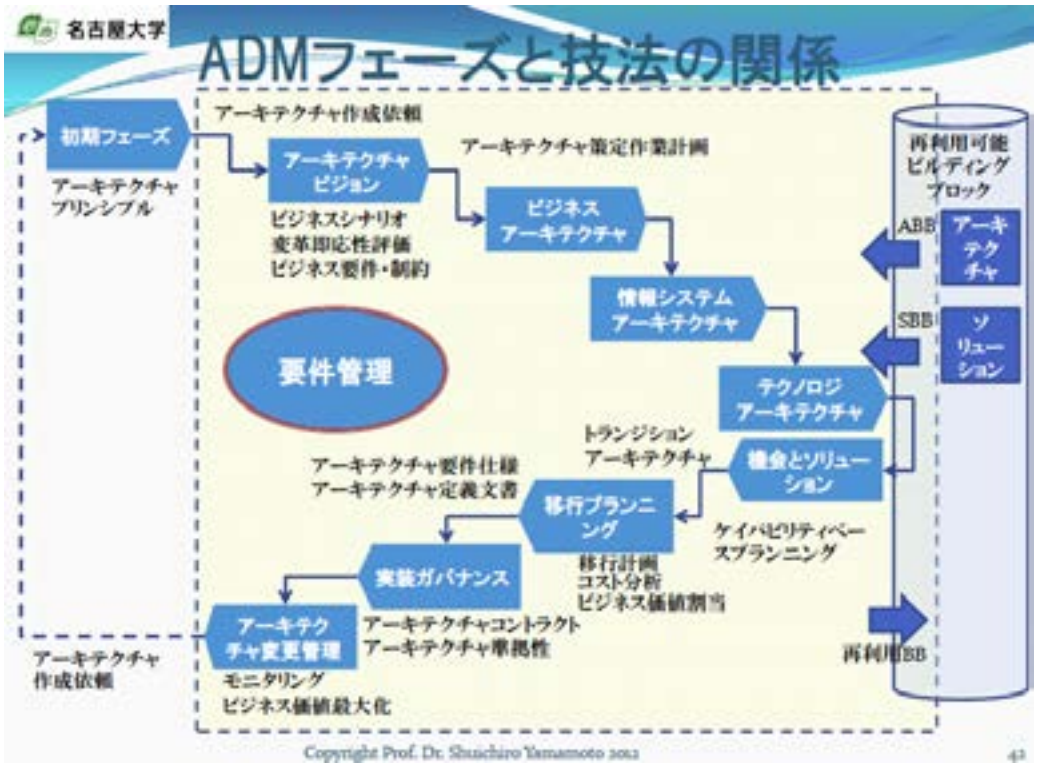


図 3.10-2　ADM フェーズと技法の関係（山本教授ご講演資料抜粋）

（5）ITIL

ITIL（Information Technology Infrastructure Library）は、運用プロセスに対する標準知識であるが、運用に入る前のサービスの戦略、設計、移行、継続的改善の知識も整理されています。ITIL の最新版では、サービス運用に「要

求実現」という内容が追加されており、サービスがあらかじめ定義された要求を満たすことを含めています。運用の標準においても要求が重視されてきています。

（6）品質ケース

品質特性が保証されていることを示す方法として、米国カーネギーメロン大学ソフトウェア工学研究所（CMU/SEI）が品質ケースという考え方を提唱しています。品質を保証するための主張、議論、証拠をドキュメントとして作成します。判断に対する理由と結果を示すことにより､品質特性が保証されていることを示すことができます。

（7）今後の展望

今後は、サービス開発とサービス運用を統合した要求状態管理モデルの実現、組み込み系で重視されてきている機能安全規格（ISO26262）で提唱されている安全性ケース、およびディペンダビリティケースが重視されると考えられます。

安全性ケースは、品質ケースを包含する仕組みであり、プロダクトだけでなく開発および運用プロセスが機能安全であることを確認するものです。

ディペンダビリティケースは、システムの高信頼性を保証するための仕組みです。開発時のプロダクトだけではなく、運用プロセスも含めてディペンダビリティを保証する仕組みが必要と考えており、現在山本教授（講演者）の研究室において研究を進めています。

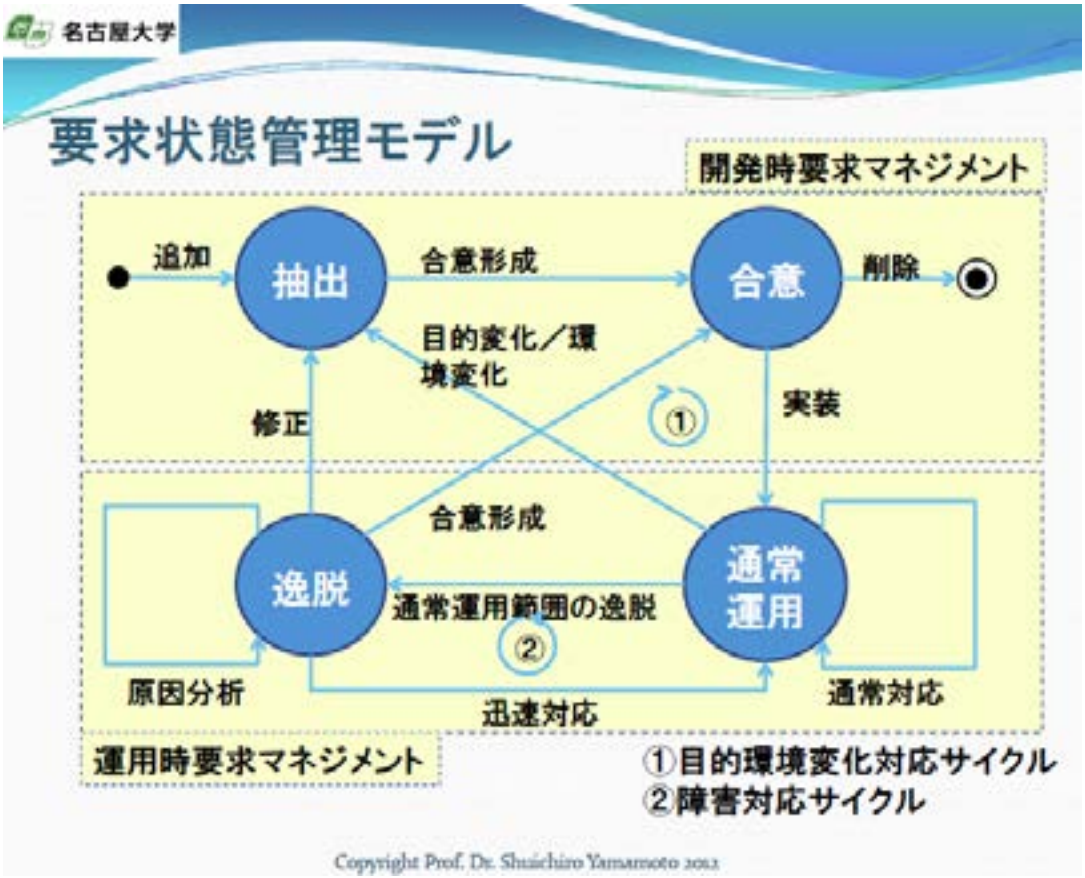


図 3.10-3　要求状態管理モデル（山本教授ご講演資料抜粋）

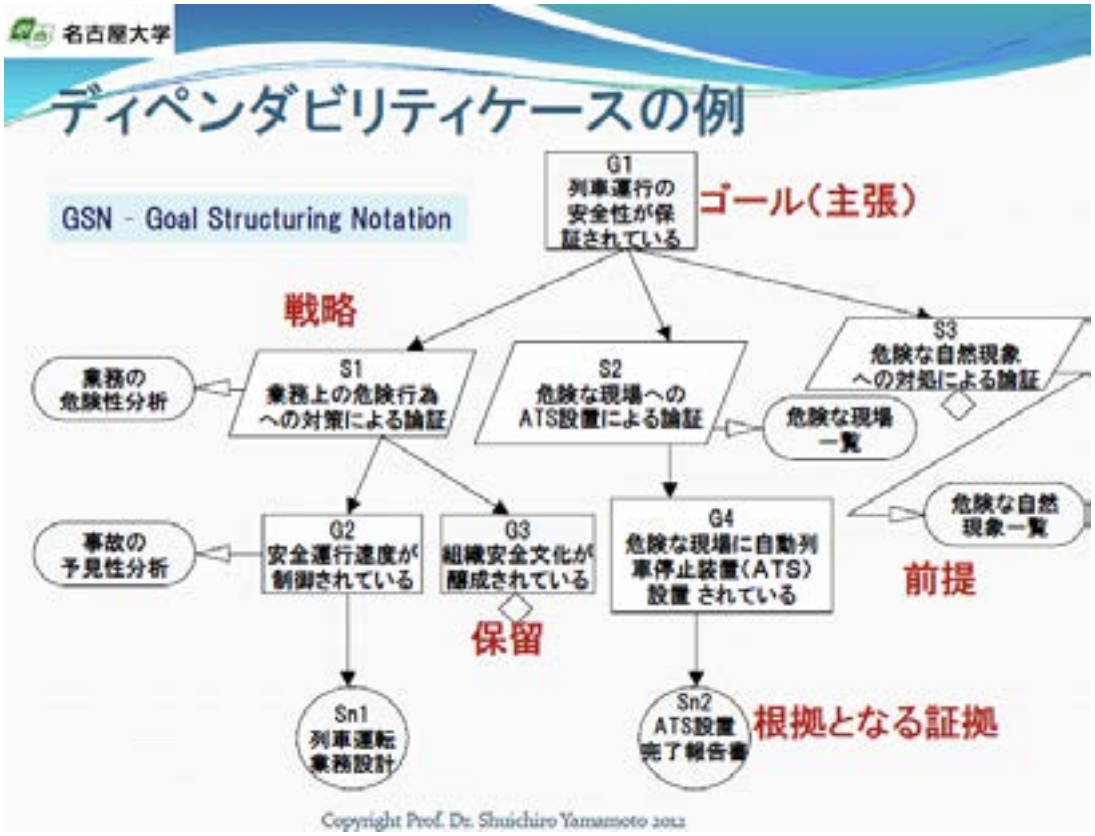


図 3.10-4　ディペンダビリティケースの例（山本教授ご講演資料抜粋）

【参加者からの質問・意見】

＜質問＞

要求定義において、担当者自身はリスクに気付いていてもドキュメント上には記述されず、リスクの存在が伝わらないことがよくある。ディペンダビリティを追及していくということは、そのようなリスクを表面化し解決していくということでしょうか？

＜回答＞

そのとおりです。ディペンダビリティケースは、最初にシステムのリスクを定義することにより、そのリスクに対する証拠、対策を記述できます。ドキュメントにすると、マネジメントレベルにも認識される。高信頼性のあるものを作るにはどうしたらよいかという観点にシフトしないと、多発する問題を解決することは難しいでしょう。

＜意見＞

大学では、基本的な知識を細かく身に着けさせるよりも、顧客価値創造はどうすればよいのかという一番肝心なことを本気で考えさせないといけないと思います。

＜回答＞

山本教授（講演者）は、TOGAF や PMBOK といった特定の知識だけを教えるということはしていません。学生が社会に出たら、世の中にあるたくさんの知識を咀嚼して、かつ最適な形で組み合わせる必要があるということを事前に教えています。総合的な観点と、社会におけるシステム開発とは何かという視点で授業を行っているので、学生が社会に出たら知っていてよかったと思うのではないでしょうか。

＜質問＞

海外の企業は信頼性を高くするところと捨てるところを割り切っています。国内の企業は過剰品質にする傾向があり、コストが上がります。そういうことをどう考えればよいのでしょうか？

＜回答＞

欧米と国内では､社会が規定する望ましい信頼性がちがいます。外から国内を見ると過剰品質かもしれないが、社会が規定することに対して適正であることが必要でしょう。

＜質問＞

SW の信頼性はどこかで割り切らないといけないが、受注、発注の関係ではそれができません。信頼性を高レベルでコミットしたベンダーが受注することになります。そのような意識を改革しないといけないと思いますが、どう考えればよいでしょう？

＜回答＞

何％コミットできるのかということを顧客への納品時に保証するような仕組みが必要です。例えば TOGAF だと、残留リスクを定義しています。リスクは必ず残るので、これだけのリスクには対処したが、先のリスクには対処できていないということを説明する必要があります。

【所感】

「顧客価値」は、システム運用が開始されて初めて生まれるものです。ベンダー側はシステム開発の終了がゴールと考えがちであるが、それでは本当の意味で顧客志向とは呼べないのではないでしょうか。本講演は、要求の段階から顧客価値を考えるというのはどのようなことか、知識を得るよい機会となりました。一つの知識体系だけではなく、ビジネスから運用までのライフサイクル全体を見る ISO/IEC/IEEE 29148、ビジネスアナリシスを重視した BABOK、企業アーキテクチャという全体最適化の視点を持つ TOGAF、品質特性を保証する品質ケースなど、さまざまな知識を概観することにより、それぞれの知識体系の特徴や強みを組み合わせる必要性を感じました。

教授は大学においても、特定の知識だけではなくいろいろな知識を教えており、学生には社会に出たときにそれを最適な形で組み合わせて使って欲しいと思われているとのこと。実際の開発現場においては一つの手法で開発されているわけではなく、顧客、業種・業務、メンバースキルなどさまざまな特性により適合する方法が変わってきます。今後は、そのような時々の状況に柔軟に対応し、知識を組み合わせて対応できる人材が求められてくると考えられます。

〔福島 著〕

3.11　人間学アプローチ

人間学アプローチというのは、「人間とは何か」「人間社会はどのように成り立っているのか」「知識やコミュニケーションはどれほど重要か」「学ぶことと社会とはどう関係するか」といった根源的な問いかけに答えることが結果としてより良い、意味のある社会に近づくために IT システムが貢献できるのではないかという考え方です。

従来のソフトウェア工学としての正当な要求定義への取り組みではなく、そこから漏れてしまう新しい動き、既存のアプローチが見逃してきた本来的な課題、社会自体が大きく変化し従来のアプローチじたいが有効でなくなってきている兆し、といった点に注目して、少し大風呂敷を広げて新しい視点を持てるような解説をしたいと思います。

図 3.11- 1 Project ICHIGAN 宣言

現代の日本において、IT やソフトウェアや要求定義を語るという場合にも東日本大震災のことを抜きにして考えることはできないと思います。それは社会というものの不安定さが露呈し、既存のビジネスや社会システムを単純に継続すればよいとか、何はともあれビジネス価値をなにごとにも優先してものごとを考える、といった安直な姿勢に待ったが掛けられたからです。自分達の存在意義や IT が社会にどう貢献できるのか、IT システムや IT エンジニアの社会的な価値は何なのか、ということを根本的に見直すところから始めないわけにはいかなくなっているというのが日本社会の現状だという認識が、多くの人に共有され始めているのではないでしょうか。

図 3.11- 2 Hack For Japan の活動

それでひとつ大きなヒントとなるのが、Hack For Japan の活動です。IT エンジニアが被災した現地に飛び込んで行って、そこで現場の実際のニーズを聞き出し、実現可能性・効果と価値・コスト・IT で、できることできないこと等を話し合って優先順位付けし、できるだけ効果のある実用的なところから優先的に 2 泊 3 日等で集中的にデザイン・実装・使ってみてのフィードバックといった PDCA サイクルを高速に回し、実際に使えるシステムを立ち上げていくという試みです。

これはまさに今、日本の IT エンジニアや IT 産業が日本社会や他産業から求められている役回りのモデルであり、このスタンスを被災地だけでなく、日常の IT の利活用業務全般に応用・展開できれば、日本の IT 産業の再生・復興も可能性があるのではないかと考えます。

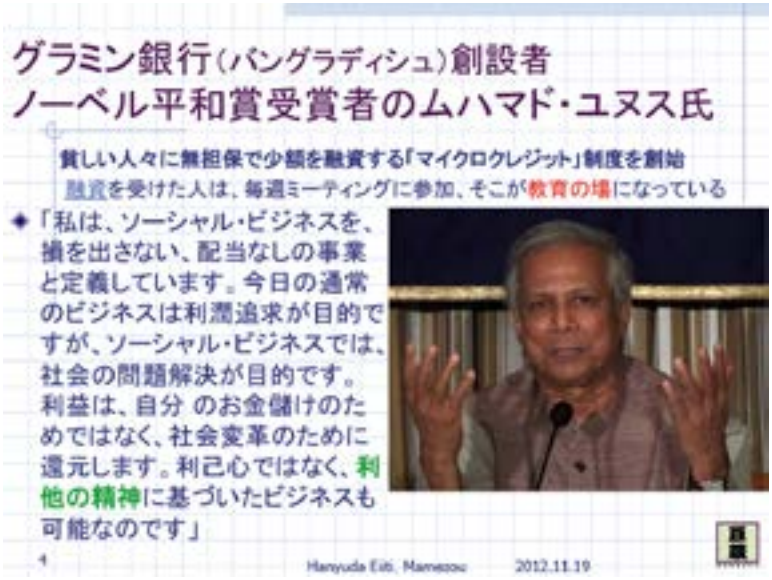


図 3.11- 3 マイクロクレジット制度の創始

もう 1 つのヒントは、ノーベツ平和賞を受賞したグラミン銀行の事例です。マイクロクレジットという貧しい市民向けの融資制度の確立も凄いことですが、そこにモバイル端末を無償で配るということを通じて、社会インフラの整備や融資したお金の有効活用をするためモバイル端末を通じて情報交換・コミュニケーションツールとして生かし、さらには毎週、融資先の市民たちにアドバイスをし、お互いの情報交換するミーティングの場を提供し、社会意識・ビジネスマインド・具体的なビジネスアイデアといった教育も行っている。そのような直接・間接的な形で IT が市民教育や社会整備の役に立っています。

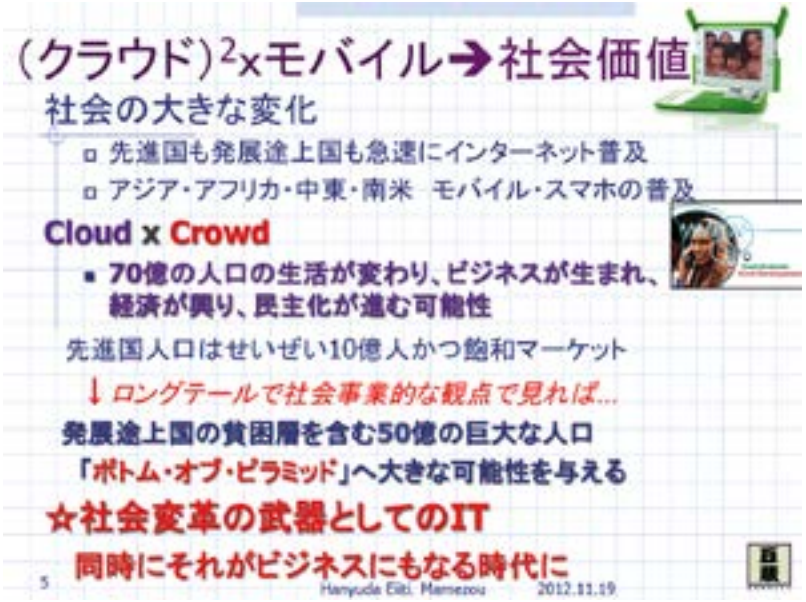


図 3.11- 4 クラウドによる社会の変化

Cloud computing としてのクラウドと、膨大な単なる群衆という Crowd がソーシャルネットワークやモバイルや IT の技術を利用して自立できる可能性をもった集団になろうとしています。つまり、現代社会において、２つのクラウドの力がモバイルやユビキタス技術とうまく結びつけることで大きなパワーを発揮できるようにもっていける可能性が開かれているということです。クラウド・コンピューティングの膨大な計算力とデータを社会階層・地理的分散・会社家庭・公共民間の区別を問わずにモバイル技術が切り崩して平等に提供することができるという可能性です。

図 3.11- 5 テクノロジードリブンの視点

いままで、IT システムの要求定義は、まず組織の目的ありきで、その目的のために適切な情報システムを設計しましょうというスタンスでした。それはそれで全く正しいのですが、一方で、蒸気機関による産業革命に次ぐ大きなテクノロジー上の大変革がいままさに起こりつつあることを直視し、そのテクノロジー上の意味を正しく認識したうえで、社会システムやビジネス戦略、制度設計、テクノロジーの応用を考えていかなければいけないグローバル社会になってしまった、というテクノロジー（というよりはテクノロジーと社会の融合）ドリブンの視点も忘れてはならないということです。

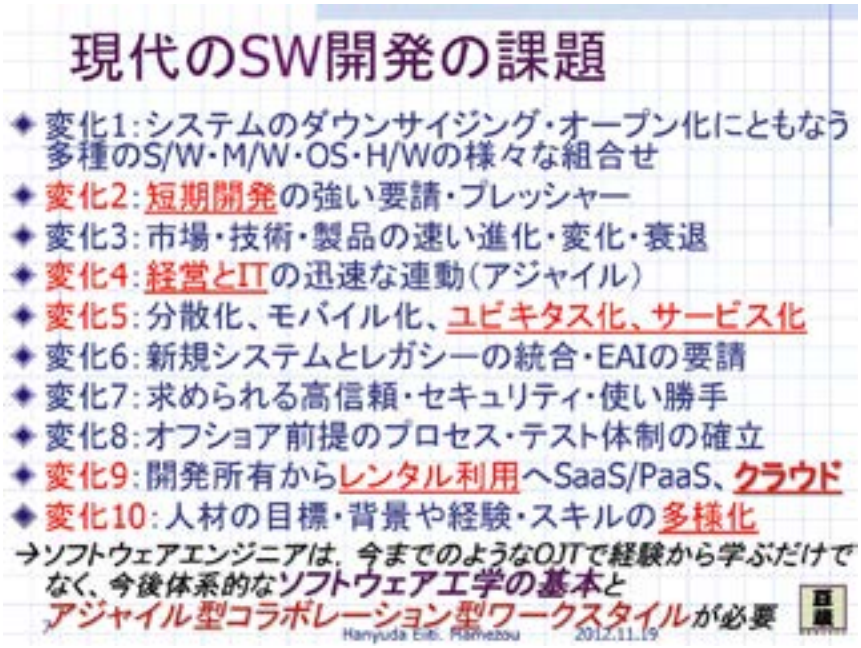


図 3.11- 6 現代のソフトウェア開発の課題

そのような視点で、ここ 20 年ほどに起こってきた IT 関連の変化を見ると、大きなテクノロジー・社会融合視点でのトレンドの中で影響を受けていることがまざまざとわかります。そして、いまやらねばならいないことは、クラウドやモバイル、サービス化を前提として受け入れた上で、いかに自分たちの社会を作りなおしていくか、そのためにいかに社会のメンバーが自主的・自律的にチームワークで協力しながら社会システムやビジネスの仕組みを作っていけるか、自分達が仕方なくではなく、納得できる仕事や生き方のスタイルとセットで確立していくことではないでしょうか。IT エンジニアはその先鞭をつけ、社会にお手本を示す立場にあるのではないでしょうか。

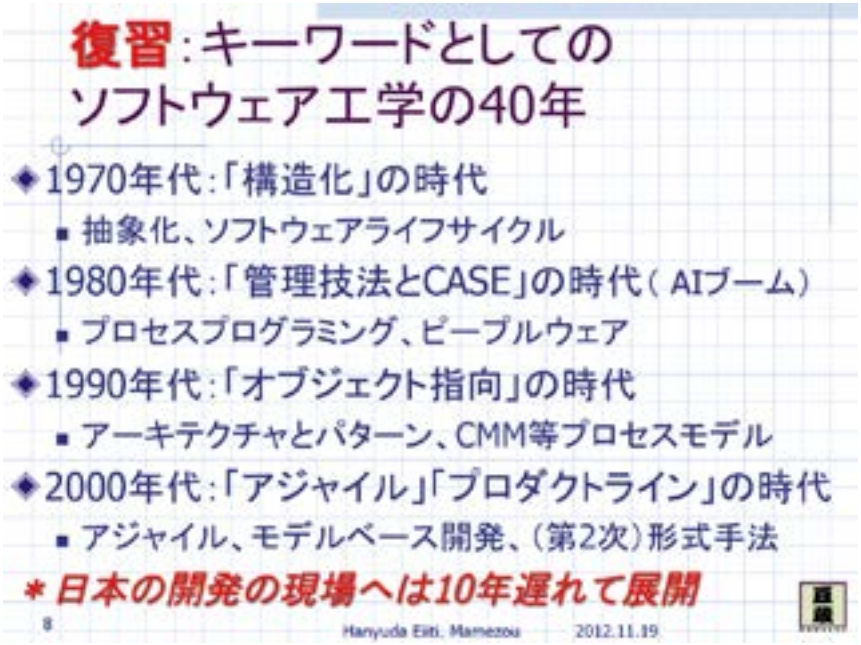


図 3.11- 7 ソフトウェア工学の 40 年

ソフトウェア工学の観点から IT 技術の 40 年間を振り返ると、社会の変化から影響を受けているとも読み取れます。

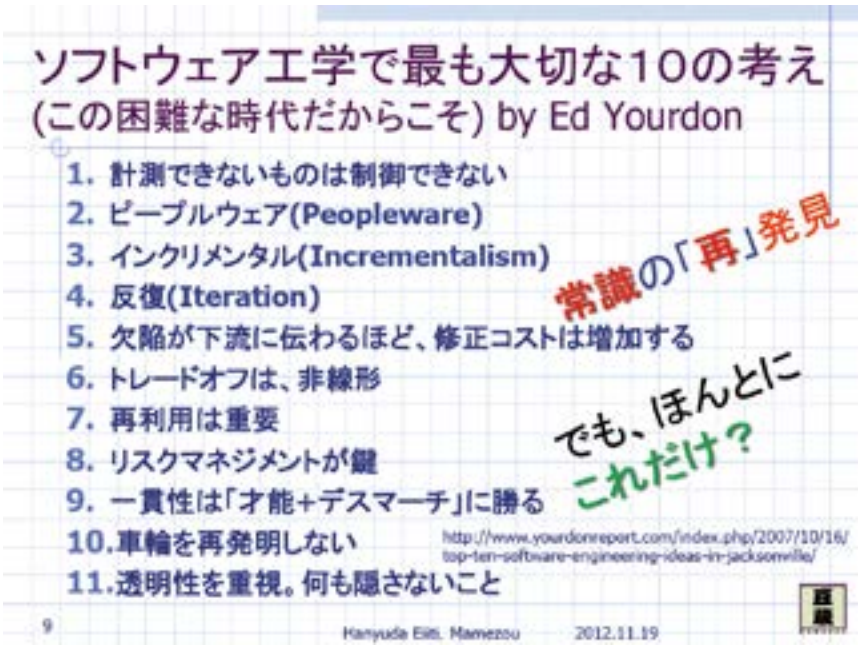


図 3.11- 8 ソフトウェア工学で最も大切な 10 の考え

そして、IT 分野でのご意見番、あの『デスマーチ』の著者のエド・ヨードンは時代の変革を感じ取りつつも、従来のソフトウェア工学の教えは十分に通用するという折衷的な立場に立っています。従来のソフトウェア工学の教えをきちんと守れば今後も IT はやっていけますよ、という路線です。しかし、本当にそれだけで済むのでしょうか。

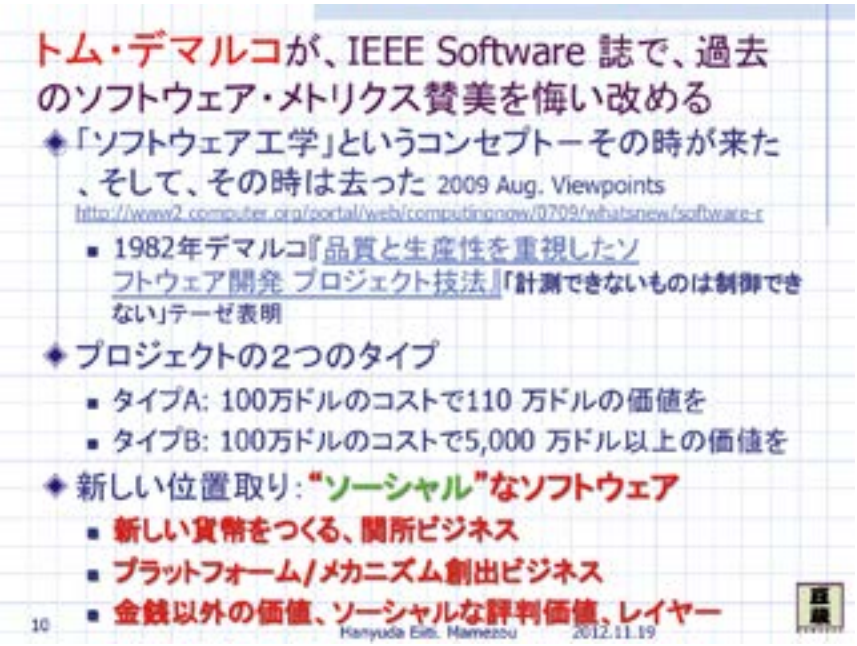


図 3.11- 9 トム・デマルコの考え

もう 1 人のソフトウェア工学のパイオニアであり、ソフトウェアメトリクスやピープルウェアを提唱したトム・デマルコは数年前の IEEE 会報誌のエッセイの中で衝撃的なことを告白しました。もう従来のソフトウェア工学というコンセプトが中心の時代は IT 分野において終わったというのです。デマルコはメトリクスの重要性を IT に持ち込み推進した張本人ですが、彼が今の時代に「計測できないものは制御できない」というテーゼは役に立たないというのです。革新的なソフトウェアやソーシャルな価値をもつソフトウェアというのはメトリクスで測ることに意味はない、もっと新しい価値や意味づけを通して評価しなければいけない、自分は間違っていたとしみじみ語ったのです。

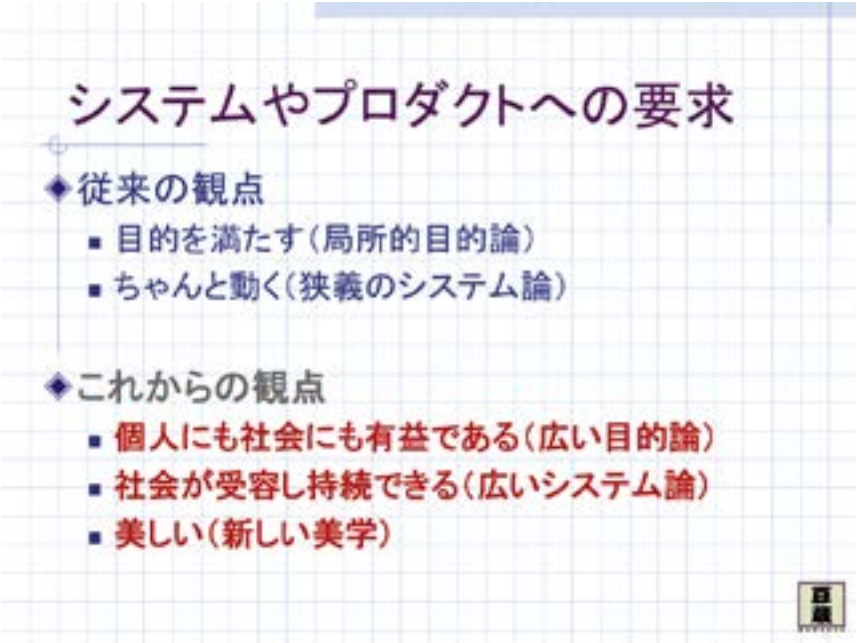


図 3.11- 10 システムやプロダクトへの要求

経済や社会が大きく変わり、システムやプロダクトに対する意味づけ、評価というものが揺らいでいることは間違いありません。従来のように、ある目的を満たしてちゃんと動く、という目的論的な世界観ではうまく説明できない事象が増えてきている。これからのシステムやプロダクトは、単に企業のビジネス目的を満たすという以上に個人にも社会にも有益であるというより広義の目的論を伴うものになるでしょう。しかも、そのシステムやプロダクトを社会が受容し持続できる広義のシステム論的な視点やそのシステムやプロダクトが美しい美学を伴っているということも評価の基準として常識となっていくことでしょう。

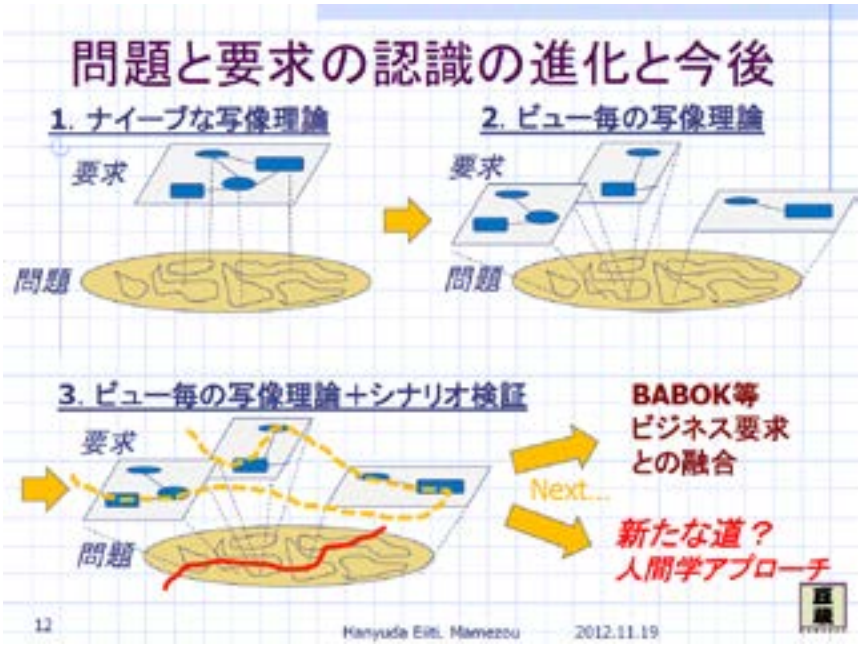


図 3.11- 11 問題の要求の認識の進化と今後

従来、情報システムを定義するという場合、クライアントやユーザ自身がもっている明確な問題やニーズが存在し、それを要求という形で写し取りさえすれば、それが要求定義になるという「ナイーブな写像理論」的な態度がありました。

しかしながら、おなじクライアントといっても立場や考え方が違う人が混じっているからその人たちごとの視点で要求を整理する必要がある、ということに気づき、「ビュー毎の写像理論」にもとづいて要求を整理すればうまくいくというアプローチに進化しました。

それでもやってみると、確かにステークホルダー毎の要求は整理されたけれども、それらの要求の間は整合性が取れているのかどうか怪しいのではないか、ということに気づき、ビュー毎の要求どうしの連関や整合性を担保するために、それらに横断的なシナリオを複数用意して「シナリオ検証」を行うとよいという方法論が提唱され、それの有効性が確認されてきました。

そして現在、社会が大きく変化しつつあり、テクノロジーの大トレンドもあり、そもそも要求や問題をクライアントが持っているのを引き出せばよいという「写像理論」自体への疑問が広がりつつあります。それに対する対応策としては大きく２つのアプローチが考えられます。１つは、情報システム要求よりも上流のビジネス要求にまで広げて分析・モデル化を行うことでシステム要求もぶれなくなるだろうという、分析というスタンスはあくまで踏襲する考え方。問題というものは対象を分析することで抽出され解決されるというスタンスは変わらない。主にBABOK 等の導入がその路線を志向しており、このアプローチ自体それなりに成果を出しつつあります。一方、もう１つの路線は、そもそも「問題と要求」という２つのカテゴリを分離してシステムを考えること自体が誤っていたという気付きに基づく考え方。顧客も IT システムにコミットするし、開発者も現場にコミットする。問題が存在するのではなく、現場や組織が機能不全を起こしている状況に IT やソフトウェアや開発者として一緒に関わりながら、状況自体を自己組織化させ問題を解消するだけでなく新たな社会システムに変革していく。対象世界（問題、顧客）と解決世界（要求、デザイン、開発者）という２つのレイヤーを分離することなく、経営と情報、経営とシステム、社会とシステム、現場と開発を１つのものとして捉えましょう、モデル化しましょう、一体としてフィードバックサイクルを回しましょう、というアプローチです。ここではそれを『人間学アプローチ』と呼んでいます。そもそも人間や社会とは何かを哲学的に問い直すことから始めなければいけないという根源的なスタンスにもとづくものだからです。

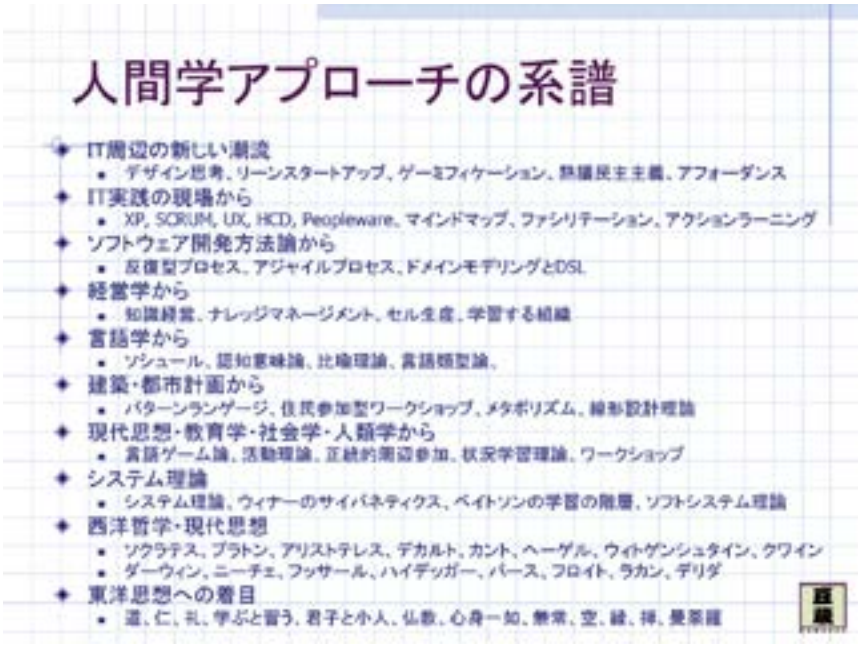


図 3.11- 12 人間学アプローチの系譜

人間学アプローチといわれるものに特定の定型があるわけではありませんが、それでも、IT や経営学といった従来の情報システム学に比べても、言語学、建築、社会学、人類学、哲学、とくに東洋思想といった非常に幅広い分野からの影響のもとに形作られつつあるアプローチだということがいえるでしょう。

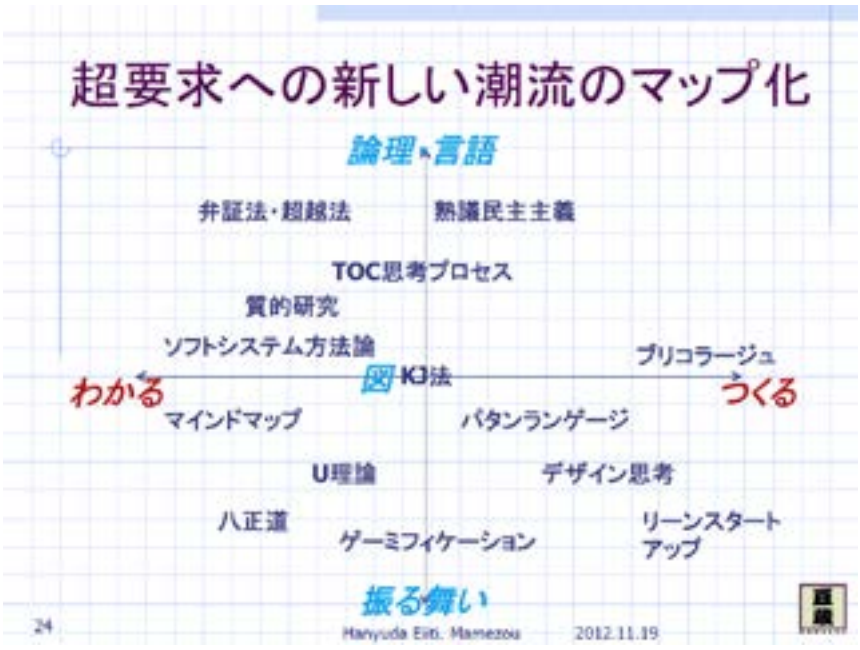


図 3.11- 13 超要求への新しい潮流のマップ化

人間学アプローチの具体例として超要求への新しい潮流といっているのは、既存の要求定義方法論の路線に対するオルタナティブな手法という意味で使っています。

それらをここでは 2 軸で整理しています。横軸に「わかる〜つくる」ということで理解すること中心なのか、実際に作りだすことまで含むのか、を示します。縦軸は表現手段での分類で、真ん中をビジュアル表現として上に行くと論理や言語という「言葉」への信頼を軸足にした手法、下に行くと身体性や実際の行為に軸足を置いた手法という位置づけです。

私の分類では、少なくとも日本の問題解決手法の原点であり世界的に見ても尤も早い段階での提示という意味で敬意をこめて KJ 法を中心に置いています。左辺のマインドマップや質的研究などはまず素直な理解をしましょう

という「わかる」こと重視の視点であり、右辺にあるブリコラージュ、デザイン思考、リーンスタートアップなどはとりあえず試しに作ってみて考えよう、という「つくる」発想が基本です。上方にあるマイケル・サンデルで有名になった熟議やその元になる弁証法は西洋思想の基本であり「ダイアローグ」「対話」を通しての理解や意思決定ということになります。一方、下方に位置づけられる U 理論やゲーム化による組織動機づけなどは行為を重視したものです。

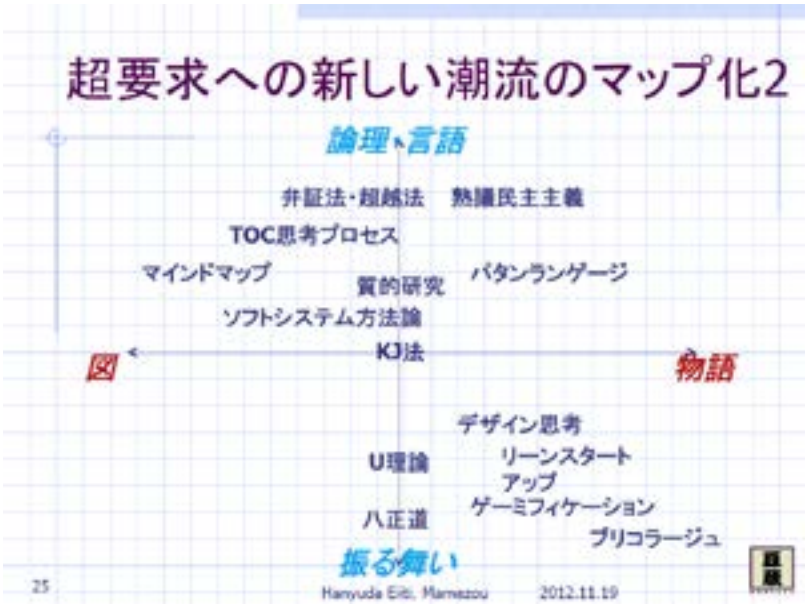


図 3.11- 14 超要求への新しい潮流のマップ化２

もう一つの視点は、表現方法・手段だけに着目して、「言語～振る舞い」という縦軸と「図～物語」という横軸で問題解決手法を整理したマップです。やはり KJ 法がそのすべての点でバランスが取れた中心という認識で、物語性をどのくらいもっているかがこの図から読み取るべきポイントとなります。

パターンランゲージもデザイン思考も物語性をそれなりに取り込んでいますが、言語を重視するかモノコト作りの実践を重視するかで鏡像関係にあります。興味深いのは、振る舞いを追求するとそれは自ずと物語性の追求にもなっていくという点でしょう。ゲーミフィケーションは業務や人生の一部の物語化ともいえるものであり、この点は今後さらに追及していく必要があります。リーンスタートアップなどもゲーミフィケーションほどではありませんが、ゲーム感覚を経営や起業に対して持ち込んでいるといえるものです。

図 3.11- 15 W プロセス 2.0

本提案では、さまざまな問題解決手法の原点を KJ 法に置いています。理由としては、日本標準であるだけでなく、世界的に見ても最も早期の総合的・体系的な方法論といえるからです。その KJ 法の背後にあるのが、川喜多二郎氏の提唱したこの W プロセスです。

なお、川喜多二郎氏が提唱された段階では IT という観点が未発達であり、ネットワーク上でのコラボレーションという発想はなかったため、それを踏まえた改訂を行いました。①野外調査を中心に置いていたため「現象学的態度」としてより一般化するとともに、②プロトタイピングといったモノコトづくりの視点が弱かったのを「身体知的態度」として提示しています。

また、物語化はそれらが累積しネットワーク上で誰もがアクセスできるフォークロアやパターンランゲージとしてデータベース化されていくと思われますが、その部分の機能を「集合知的態度」として旧来の民俗学と現代の SNS を総合させていただきました。それらをひっくるめて今回は補強改訂し、W プロセス 2.0 として再提案しています。つまり、質的研究 -> 物語化 -> プロトタイピング -> フィードバックと知識化 / パターンランゲージ化のループが繰り返し行われることが、人間の知的社会活動の理想形だということです。

なお、今回、経験レベル・思考レベルに加えて、物語化が集積し知識化・パターンランゲージ化・共同幻想化するレイヤーとして、無意識レベルを追加しています。

これからのシステムやプロダクトへの要求は、特定の企業団体の対応や支援するものを超え、社会も良くしていかなければならないという視点が必要になります。また、使い捨てではなく、再利用したりや保守を行い持続できることを目指さなくてはいけません。単なる機能の善し悪しだけではなく、機能性から解放された遊戯性や物語性をも兼ね備えたものがあることなどが新しい要求につながるのではないでしょうか。

そのためには、問題を抱えている人々と解決しようとする人たちが相対するということではなく、交わり一緒になって進めていく必要があるのではないでしょうか。

〔樋山 著〕

3.12　情報システム技術の叡智

（1）この資料の位置づけ

3.12 節は、2012 年 11 月 19 日（月）に開催した Openforum2012 でのパネル 2（“ 一生役立つ情報システム技術の叡智 ”）の内容をまとめたものです。

（2）パネルの趣旨

情報技術や情報システムの設計と開発・維持（計画立案、要件定義、設計、開発、テスト、運用、改善）あるいは、ソフトウェアが組み込まれたシステムを活用しそれをマネジメントするのに関わる “ 幅広い領域 ” のスキルは、単に IT ベンダーだけでなく、企業や行政組織やそれ以外のあらゆる組織になくてはならないものとなっています。これから数年後の将来を見据えても、企業や行政組織において情報を活用し管理する情報システムの役割は、ますます重みを増すことはあっても減ることはありません。

そういった中で、情報の意味と情報システムの役割を理解した若年層を育成することは、世代交代が進む中、IT ベンダーは別として、一般企業や行政組織の効率化や変革を推進する上で急務の課題となっています。IT があらゆる業務に関わることにより、そういったスキルの習得は、理系・文系を問わずに必須になっているのではないかという問題意識がベースにありパネルを設定しました。

パネルでは、情報システムが社会のあらゆる仕組みにかかわっていく中で、その重要性、役割の変化、そして、それに関わることの面白さをパネラーの方々にお話ししていただき、大学教育や社会人教育における情報システム分野の教育の充実のヒントになればと期待してテーマとして設定しました。なお、タイトルの意味は、「情報システムに関わるスキルを身につけることは、これから一生役に立つ叡智ですよ」ということを意味しています。特定の技術分野を叡智と語っているのではないことをお断りしておきます。

（3）パネルでの意見

パネルは次の 4 名の方に参加していただきました。

① 加藤　修明　様　大成建設株式会社　社長室　情報企画部　企画室　主事

② 中川　憲一郎　様　ＮＫコンサルティング　代表，中小企業診断士　事業再生士補、ISO 9000/14001 審査員補・産業カウンセラー，かわさき新産業創造センター　販路開拓アドバイザー

③ 大熊　眞次郎　様　カシオ計算機株式会社　業務開発部　部長

④ 桑原　里恵　様　　札幌スパークル株式会社　システム　コーディネーター

加藤様と大熊様は大企業で情報システムの企画・開発・運用にかかわっている方。中川様は、中小企業診断士として、中小企業の企業経営にかかわっている方、桑原様は、システム・コーディネーターとして、IT の先端技術の普及に長らく関わってこられた方です。

以下にパネルでの意見交換で語られたキーワードを３つのカテゴリーに分けて並べてみます。

A. 業務と情報システムとの関わり方

① 仕事の品質を IT で強化できる

② 会社全体の業務を俯瞰する力が求められる

③ 企業経営を知らない技術者が増えている

④ 試作開発、販売における IT の活用が課題

⑤ IT がわかる業務メンバが必要

⑥ 業務がわかる IT メンバも必要

⑦ その融合で仕事をする

⑧ 中小企業は業務の見通しがつきやすく、IT 部門と融合しやすい環境がある

⑨ 部門の技術者と一緒に現場に入り、理論、背景を理解してシステム化する

⑩ 業務と IT 分野の技術者からなる立体的なチームを作る

⑪ 立体的なチーム（業務、IT、日本、海外、社内、社外）を組み合わせるのが今後のテーマ

⑫ 若いうちにビジネスモデルを知っておいたほうが得だと思う

⑬ 費用対効果を新卒のころから考えてほしい

⑭ ベンチャーや中小企業に学生は関わってほしい

⑮ 技術の組み合わせの時代となり、ユーザー企業側の醍醐味が大きくなっている

⑯ 基幹系のシステム再構築は 10 年に 1 回程度で、経験者が不足していることに危惧される

⑰ どのような会社・組織に入っても、情報技術は一生役立つ

B. 情報技術の方向性と活用

① 業務プロセスを複数のシステムを組み合わせて使う技術 (SOA) が大きな変化

② 手段を組み合わせてシステム・業務を実現する

③ 事業視点で選択してカスタマイズ (組み合わせの時代がきた事が大きな変化)

④ システム単位での要件定義は時代遅れ

⑤ プロトタイプを市場に投げて均衡を見て分析する時代

⑥ 速やかに実現する力が求められている

⑦ IT 化のデザインに関わっていくことが大事

⑧ 日本は新バージョンの採用に保守的

⑨ バーチャルとリアルが一体化する時代になる

⑩ IT の位置づけが変わり、求める姿と条件が変わって、使う技術が進化すれば、方法論も違ってくるべき。そうすれば当然、求める人材とチームの形も変化する

⑪ 事業と IT の一体化が加速（事業の商品サービスに組み込まれる IT）

⑫ “ 発想が大事 ” なサービスにも開発の初期から IT 系スキルが必須（–立体化するチーム、そしてデザイン思考とシステムデザイン）

⑬ 事業の強みを活かし、個性に応えるにも、IT の可能性を発揮するにもデザインの力が大事

C. 人材育成

① これからは、知の構造化が必要

② 新しい仕組みや価値を創造する力が必要

③ 体系的な学習をする力が必要

④ IT 時代の位置づけが変わり、システム感が変わり、メソトロジーも変わる

⑤ 人材も変わるチームも変わる必要のあることを感じる

⑥ ソフトウェアスキルは極めて重要

⑦ 人材は不足している

⑧ 高いエンジニアリング・スキルを持つ技術者を育成しなければならない

⑨ 不確実なことに挑戦したいメンタリティに不足している

⑩ 海外の技術者はモチベーションが高い

⑪ 不確実は技術で乗り越えるという意欲が不足している

⑫ どこでも IT 人材は不足している

⑬ 創業者は中国・東南アジアが多い

⑭ 彼らには、不確実に挑戦し、走りながらプロトタイプをつくり実現していく勢いがある

⑮ デザインをイメージ、発想できる人を増やしていかなければならない

⑯ 戦略的ゼネラリスト、スペシャリストの育成が課題

⑰ 決められたフォーマットで教育されている

⑱ 新しい価値を生み出す教育になっていない

⑲ 考えて何かを生み出す訓練がなされていないので、そこからではないか

70 分というテーマに対して短い時間でしたが、企業における情報技術の役割、価値、そして人材育成の重要性について思いを語っていただき、情報技術がビジネス価値そのものを生み出しているということをお伝えできたのではないかと思います。

（4）高等教育に望むこと

情報システムが企業や行政組織あるいは社会に完全に浸透しているという認識は、20 年以上前に慶応義塾大学の教授でいらした浦昭二氏（故人）がすでに指摘しています (*1)。平成 4 年に書かれたその著述の中に次のような記述を見ることができます。

『コンピューターを中心とした情報技術の社会への定着が進む中、情報システムの企画・開発または運営に当たる人材の不足が現在指摘されており、将来も要求され続けるものと思われる。情報システムのための人材は、情報およびコミュニケーションの観点でシステムを考察できるとともに、関係する領域について問題点を洞察できるだけの知識と観察力を持ち、さまざまな情報技術を活用できる能力を備えていなくてならない』（下線は筆者）

さらに、次のような指摘もされています。

『情報システムの重要性､必要性が社会に意識されるようになっているにもかかわらず､現状は「コンピューター・システム」と「情報システム」との混同が見られ、必ずしも情報システムとしても視点が正しく認識されているとはいいがたい』

その報告書では、情報システムを「単なるコンピューター・システムではなく、人間系を含めた組織体の活動に深くかかわるもの」ととらえています。情報を収集、活用することは、人が組織の中あるいは一般社会生活を行う上で必然の行為と考えており、そのような観点から社会生活を行え上で情報システムに関わる人材の育成が必要だと主張されています。

まさに炯眼であります。ただ、現在もこの状況が改善されていないところに大きな問題があります。

現在でも情報分野の大学教育は、情報システム・サイエンスや情報工学を中心に、従来から理系中心に行われてきています。それは、コンピューター・システムがまだプログラミングの時代であった名残で、IT ベンダーに就職する学生やコンピューター技術の研究者としての専門家の育成のみを意識した教育になっているように思えます。

20 年前の報告書を顧みるまでもなく時代は大きく変わっており、今では、情報システムを開発し維持・改善していく仕事は、まさに企業や行政の業務の仕組みや顧客サービスの仕組みを改善することに直接つながる仕事となっています。また、情報システムは、すでに社会生活の基盤となっているといっても差し支えないでしょう。情報技術の詳細の仕組みは知らなくても、インターネットでの情報検索やショッピング、あるいは意見投稿などは日常的な生活基盤となりました。

グローバルなのは経済だけでなく社会そのものです。政治にも“インテリジェンス”の必要性が叫ばれていますが、まさに世界情勢の情報をどのように分析し、どのように対応すべきかが問われる世界です。そのような社会で政治が情報に敏感に対応する、ないし情報を発信する、あるいは、企業が競争し国民が生活する中で、情報や情報技術を活用し、そして生活する手段としてソフトウェアを作ることが当たり前の世の中になるということを見据えて、先の報告書に示された教育体系をも含めて新しい視点から“情報、および情報システムに関する教育”を考えていくことが求められています。

企業や行政などの組織における情報システムにかかわる人は、情報システム部門や IT ベンダーの技術者だけではなく、これからは、業務部門に所属する社員の多くが該当するようになるでしょう。その観点から、次のような観点からの人材育成としての教育を推進してほしいと願っています。

(ア) 社会､企業､あるいは人と“情報”とのかかわり(人の活動が“情報活動”であるという視点を考えさせる)

(イ) 社会と“情報システム（情報を扱う、管理するシステム）”とのかかわりの今と将来を考えさせる教育

(ウ) 企業や行政における経営マネジメント技術と情報システムとのかかわりを考えさせる教育

(エ) 現実の問題を問題としてとらえ、解決策を導き出す方法論に関する教育

(オ) 現実を抽象化して、モデル化するスキルを身につけさせる教育

(カ) ソフトウェアがあらゆる“モノ”に入り込むことを前提とした、抽象化とソフトウェアの設計技術

(キ) ソフトウェア開発の自動化がもたらす新しい設計パラダイムを考えさせる教育

(ク) 社会生活を行う上でのインフラとしてのインフォメーション・テクノロジーについて考えさせる教育

(ケ) インフラとしてのインフォメーション・テクノロジーのリスクについて考えさせる教育

(コ) インフォメーション・テクノロジーの進化のスピードの観点からの将来の社会システムの姿を考えさせる教育

いずれにしろ、情報技術の進化は非常に早く、過去の技術を大学４年間で教えるだけでは足りません。また、実務経験がない 20 歳前後の学生が深い理解をするのは困難という現実もあります。したがって、大学における情報技術や情報システムに関する教育は、“ 教える ” ことよりも “ 自ら考え実践する ” という観点を充実させるのがよいと思います。そして、最初にも書いたように、情報技術をどう使っていくべきかという考え方は、理系・文系を問わずすべての若い人が身につけるべきリテラシです。それが、日本の企業や行政、そして政治活動のイノベーションの源泉になるとの理念を持って、高等教育の場における情報教育の見直しを推進していただきたいと思います。

なお、このパネルの企画にあたっては、P-Sec 会員でもあります神沼靖子氏に事前に相談にのっていただきました。末尾ながらお礼申し上げます。

〔大島 著〕

(*1) 情報システムの教育体系の確立に関する総合的研究　平成４年３月発行
　　平成３－４年度科学研究費補助金（総合研究 A）　研究成果報告書
　研究代表者　浦　昭二（慶応義塾大学理工学部　管理工学科）
　（なお、神沼靖子氏も著者の一人です。）

3.13　TERAS

本節は、2012 年 12 月に開催された WG1 会合における TERAS 理事& CATS 取締役副社長の渡辺政彦氏の講演に基づき、内容をまとめたものです。

一般社団法人名の TERAS とは、”Tool Environment for Reliable and Accountable Software” の略称です。TERAS では ”Open Traceability Tool Platform” と称して、トレーサビリティを中心とする技術的・社会的な仕組みを提供し、製品の説明責任や訴訟のための証跡機能の技術開発や普及・啓発を行っています。2011 年 10 月に設立され、以降、ツールプラットフォームの開発も版を重ねてきています。

（1）背景と目的

製品・システム・サービスは無くてはならないものであると同時に、原発事故、証券システムトラブル、銀行システム障害、自動車リコール、個人情報流出、国家機密漏洩などに現れているように、提供者の社会的な責任を問われるケースが増加してきています。どの程度頼りになるかという性質は、信頼性、可用性、安全性、セキュリティなどを総称して「ディペンダビリティ (dependability)」と呼ばれています。

世界標準の世界では、IEC 61508(JIS C 0508)：電気・電子・プログラマブル電子安全間連係の機能安全の標準が制定されており、さらに、この自動車分野向けの ISO/IS 26262（2011 年 11 月発行）が検討されています。これ等の標準は、欧米への輸出において関税障壁の観点から考慮せざるを得ない状況にあります。

その一方で、技術革新が激しい先端技術には潜在リスクがある中で、高機能化・大規模化・複雑化・多様化してきており、これ等を経済的な観点からグローバルな競争にさらされており、低コスト化も図っていかなくてはなりません。

日本では、独特のすり合わせ型の開発があり、ドキュメント化が不足していたり、暗黙知が多い状況にあり、ディペンダビリティに関する説明責任を果たすために必要なドキュメントのトレーサビリティを確保することが難しいのが実情です。

このような状況を鑑み、TERAS では、日本のものづくりを強くするツールとそのプラットフォームの実現を目指しています。

（2）トレーサビリティとは

トレーサビリティの考え方は、ISO などでは、「作業内容の記録を残し、源、処理の履歴、所在を追跡できること」と述べられています。食品の安全性を確保する目的で、牛の個体識別の管理が義務づけられたり、食品においても事業者間の産地情報の伝達、消費者に対する産地情報の提示が一般に行われるようになってきています。

トレーサビリティの目的は、設計変更時の影響を分析すること、設計・検証に漏れがないことを示すこと、さらには、第三者による検証や妥当性確認、認証を得る場合に必要になります。追跡する対象は、要求 - 設計 - プログラムの間の関係や、要求・設計・プログラムに対するレビューやテストなどの検証項目との間の関係です。

さらに、製品の開発プロセスに沿った形で、トレーサビリティを確保していく方法を仕組みとして確立していく必要があります。日本では、要求から実装へ向けての一方向だけでなく、サプライヤからのフィードバックや、協調作業によって製品の開発が進められるいわゆる「擦り合わせ型開発」に対しても有効な方法を確立していくことが望まれています（図 3.13-1）。

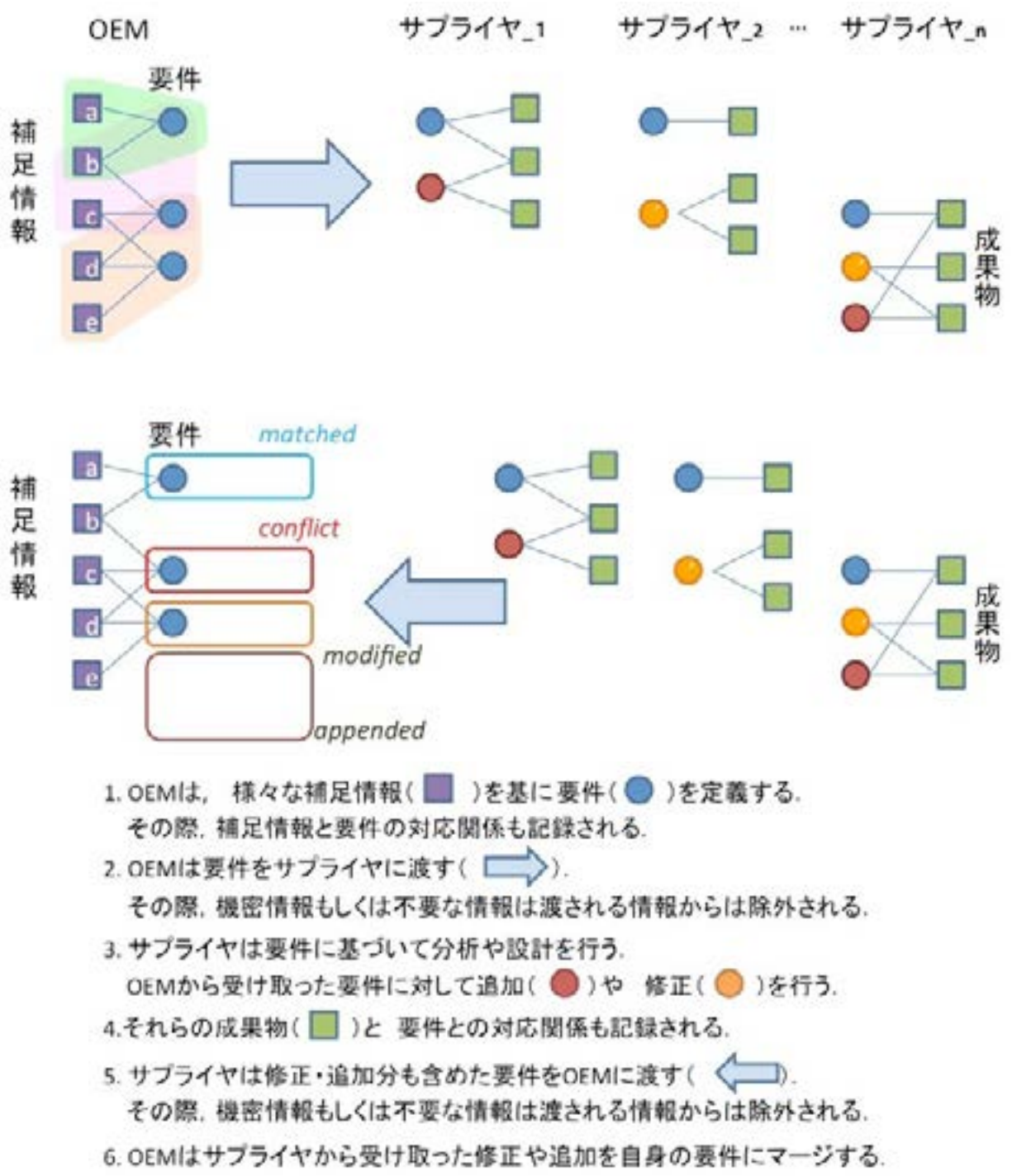


図 3.13-1　モノ作りプロセスを支援する

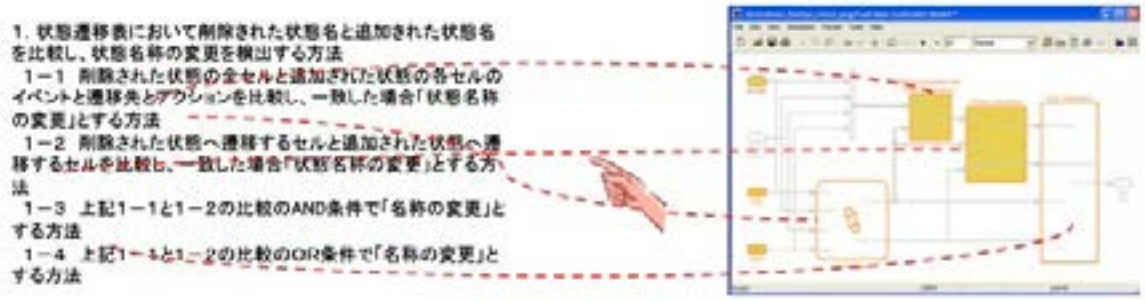
（3）ツールプラットフォーム

トレーサビリティ支援のためのツール（プラットフォーム）に対する要件は、以下のようにまとめられます。

・製品ライフサイクル全体を対象としていること：機械・電気・電子設計、ソフトウェア設計などすべての開発プロセス、出荷後の保守・運用（クレーム処理）などの広範囲にわたる ALM(Application Lifecycle Management) が必要。

・すべての種類、複数の関与組織にまたがるドキュメント間の関係を扱えること。すなわち、オープンであること。

・すべてのドキュメント（モデル記述、プログラムなどを含む）に対してバージョン、バリエーションを扱えること。



変換ルール	UID	UID	変換ルール
1-1	R0123	D011	BLOCK1
		D013	BLOCK2
1-2	R0124	D021	BLOCK3
1-3	R0125	D022	BLOCK4
1-4	R0126		

図 3.13-2　タグ・変換ルールを活用したトレースの方式

ドキュメント間のトレーサビリティを確保するためのリンクを貼っていく方法には、明示的にユーザ管理のタグ ID を付していく「タグ方式」と、トレースのルールを設定してツール管理によるタグ ID を使用していく「タグレス方式」とがあります（図 3.13-2）。

TERAS ツールでは、Microsoft Word の章立て、Excel の行 / 列 / シートや図形、Power Point のテキストボックスの各行や図形のテキスト、PDF の正規表現による抽出項目の粒度で成果物をアーカイブし、リンクを貼ることができます。

また、オープン化という観点から、開発時に使用されるさまざまなツールで作成されるドキュメントや成果を自然に扱えるようにツールとの連携の仕組みを、Plug-in や Web 標準技術（REST）の標準インタフェースを提供しています。具体的には、MATLAB/Simulink、ZIPC の記述単位、Trac、Subversion でタスクやファイル単位の粒度で対応関係を管理することができます。

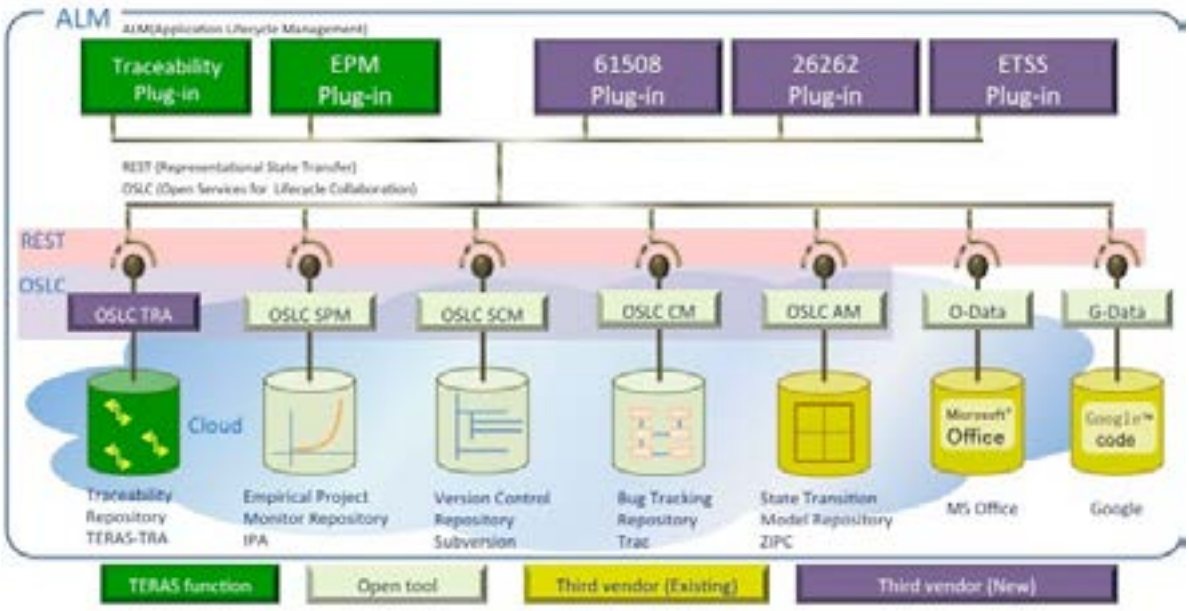


図 3.13-3　オープントレーサビリティ・ツールプラットフォーム

（4）アーキテクチャ

TERAS のアーキテクチャのコンセプトをまとめると以下のようになります（図 3.13-4）。ここでポイントになるのが OSLC(Open Services for Lifecycle Collaboration) です。

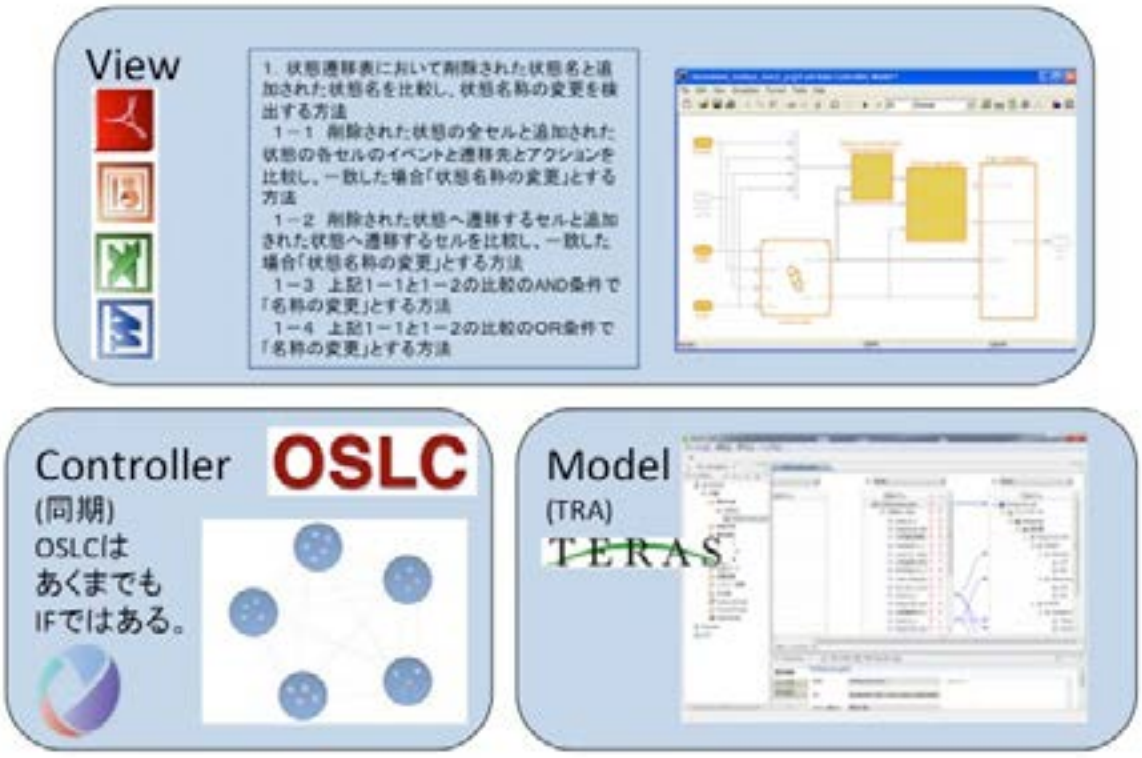


図 3.13-4　TERAS アーキテクチャ

OSLC は、2008 年に IBM から提案されたソフトウェアおよび製品の企画・開発から使用・破棄にいたるライフサイクルを支援するツールを統合するための仕様（インタフェース）を規定しています（http://open-services.net）。

データ版の World Wide Web である Linked Data を活用します。すなわち、ものの名前として URI(Uniform Resource Identifier) を使い、名前の検索のために HTTP URI を使い、RDF(Resource Description Framework) や SPARQL(SPARQL Protocol and RDF Query Language) などの標準技術を使って情報提供を行い、データの中に他の URI リンクを含めて検索効率をあげます。

設計原理は、分散型ハイパーメディアシステムのための REST (Representational State Transfer) をベースとしており、以下の 4 点になります。

- **ステートレス**：クライアント側もサーバ側もメッセージ間のセッションの状態を記憶しておく必要がない。
- **よく定義された操作**：操作（メソッド）の小さなセットが定義されている。例えば、GET, POST, PUT, DELETE のセット。
- **リソースを識別する仕組み**：すべてのリソースは URI によって一意的に識別できるアドレスを持つ。
- **リソース間の関係性の記述**：ある REST リソースから他の REST リソースを参照するリンクを含んでいる。

（5）TERAS の持つインパクト

TERAS の活動は、2011 年度から開始され、まず Version 1 がリリースされ、成果物間リンク、カバレッジ確認、影響範囲分析などの基本機能が提供されました。2012 年度末は、Version 2 が開発され、Version 1 の機能強化が図られ、REST/OSLC 対応、Subversion 連携、認証・ユーザ管理等のプラットフォーム基盤機能、トレーサビリティ管理機能等が提供されました。2013 年度以降は、よりユーザビリティを向上させ、Trac や Redmine 連携、バリエーション管理機能、クラウド環境評価等が推進されている模様です。

このような社会基盤整備活動は、時間がかかり、人々の意識改革を促していく地道な取り組みです。コミュニティや団体として立ち上げた後に、活動が継続し、進化していかなくては意味がありません。ライフサイクルの観点から言うと、製品、部品、ソフトウェア技術、そして、開発に携わる企業よりも、トレーサビリティ情報は長く、維持・管理されなくてはなりません。

一方、TERAS が対象としている製品分野のみならず、エンタプライズシステム、一般のソフトウェア、そして、音楽・映像や著作などの電子コンテンツなども、複数の関与者が絡み、版管理などでも、こういった仕組みは必要になります。TERAS で得られた知見、社会的な仕組み、アーキテクチャなどを、より広い分野で展開していけるとよいでしょう。その際に考慮すべき点は、以下のように考えています。

- TERAS は、どちらかと言うと作り手側の論理です。製品とりまとめ企業から部品提供企業、下請けなどを辿ることを中心にしています。これを使い手側、利用のコンテクストの世界まで広げる必要があります。
- ドキュメントのトレースをする機能も重要ですが、製品やサービスそのものをトレースする機能も有効です。発注・受注の関係だけでなく、あらゆる意味での取引や組織間のやりとりを辿ることができると、強力な社会基盤になると考えられます。
- 社会的な責任の連鎖を辿ることは、反対の見方をすると経済的な配分を辿ることにも繋がります。契約や取引の方法は多様です。金銭や貨幣の考え方も社会情勢によって変化していくことも考えられます。このようにトレース情報は、後から出現してくる新しい取引や経済的な仕組みに対応できるような広義のアーキテクチャであることが要請されます。

〔大槻 , 渡辺 著〕

3.14　変化する企業 IT、多様化する役割

――事業と IT の一体化を担う “ ワクワク感 ”

（1）企業 IT を巡る変化の波

クラウド、モバイル、ビッグデータ、M2M……。企業 IT を巡る技術は著しい速さで進化し、「Consumerization（個人向け技術を企業 IT に適用）」の波は費用や期間、アプローチに対する旧来の常識を覆し、IT 投資全体に新しい価値観をもたらしている。

企業 IT のスコープも、事業基盤となる基幹系プロセスから企業の枠組みを越えたコラボレーション、顧客接点、商品サービスそのものまで、広く深く拡大している。それに呼応して、「CDO（Chief Digital Officer）」という IT を事業の最前線で活かす役割が生まれ、遂には、「CMO（Chief Marketing Officer）など非 CIO による IT 投資」が、IT ベンダーの重点戦略として注目されるようになった。

企業 IT の世界は、複数の側面から大きく様変わりをしている。しかし、そのめまぐるしい変化の中に、ひとつ、大きな変化のうねりを見ることができる。あらゆる動きに共通する根幹となる変化。それが、システム視点から利用視点への変化だ。

システムの側から何を使い、何を作るのかを考えるのではなく、利用の側から「どういう状態を実現したいのか」を考え、そこに必要なシステムを組み合わせる。個々のシステムを構築することは、利用から見れば、手段であり一過程でしかない。クラウド、ビッグデータといった先進技術もそれ単体として使うよりも、全体像に位置付くことで「これまでなし得なかった」状態を実現する。イノベーションへの期待が大きい。

（2）システム像と技術が変われば役割も変化する

システム視点と利用視点。この顕著な違いは、実現の単位にある。個々のシステム単位ではなく、利用から見た全体像として考えること。利用視点の典型が、事業活動の単位であり、事業が志す顧客体験の単位である。事業の視点で求める状態を描き、具体的な業務プロセスを考えれば、そこには複数のシステムが関わる。

複数のシステムを組み合わせて、求める状態をつくりだすこと。クラウドではこれを、「Choice and Customize（選んで自社に最適な状態をつくること）」の言葉で説く。パブリック・クラウドの SaaS を使ったからといって、「アプリケーションに業務を合わせる」わけではない。同じクラウド・サービスを使っても、企業によって異なる姿ができあがる。求める姿を描き、必要なサービスや自社開発のシステムを組み合わせる。時には、独自性の高いサービスが全体像を刺激する。サービスを介して、専門知識や高度な判断を得ることもある。クラウドはすでに、所有 / 非所有の価値を超えて、利用視点の IT 化に欠かせない策となっている。

「事業と IT の一体化」はスローガンではない。事業から見た IT の重要性が高まり、事業と同じスピードと投資感覚が求められるようになる。事業の視点で実現像を描き、稼働後の業務適用や成長を含めたライフサイクル全体で取り組みを考える。

求める姿と価値観が変わり、それに応える技術が進化すれば、方法論も変化する。方法論が変化すれば、自ずと求める役割とスキルは変わってくる。実現すべき What が変われば、それに応えて How も変わる。事業と IT の一体化、そして、背景にある利用視点への変化は、新しい役割と人材像をもたらしている。逆に言えば、事業と IT の一体化が切望され、技術が進化する中で、それに応える人材とスキームが強く求められている。

まず、漠然としたニーズや課題から求める姿を描き、実現可能なプランへと具体化するグランド・デザインの仕事がある。求められる姿を求められる条件で実現するには、「実現の解」となる工夫が必要だ。多くの場合、「これだけは譲れない」という機能や品質があり、その実現が全体のプランさえ左右する。費用に合わせて実現内容を削ることも、そう単純にはできない。システムが動くために必要な機能と、事業が競争力として求める機能とは合致

しない。トレードオフ・バランスを打破する具体策が要る。

システム側に自然な“できるなり”では、事業の期待に応えられない。実現像や機能の持ち方、システム構成と技術の選択、方法論とアプローチ。あらゆる側面から知恵を絞り、技術を駆使する。不確実を乗り越え、広く人材を求める必要もある。どのような姿を、いかにして実現するのか。デザインとプロデュース力が問われる。

実際に全体像を実現する工程は、大きく３つの仕事に整理できる。ひとつは個々のシステムを構築、あるいは用意する仕事。もうひとつは、事業の視点から業務プロセスを描き、システム機能を組み合わせて、利用者ごとに応じた画面や機能へと実現する仕事。そして３つめが、システムの稼働準備と平行して、業務への適用を進める仕事である。利用環境や前提とする状態を整えることもまた、重要な役割になる（図 3.14-1 参照）。

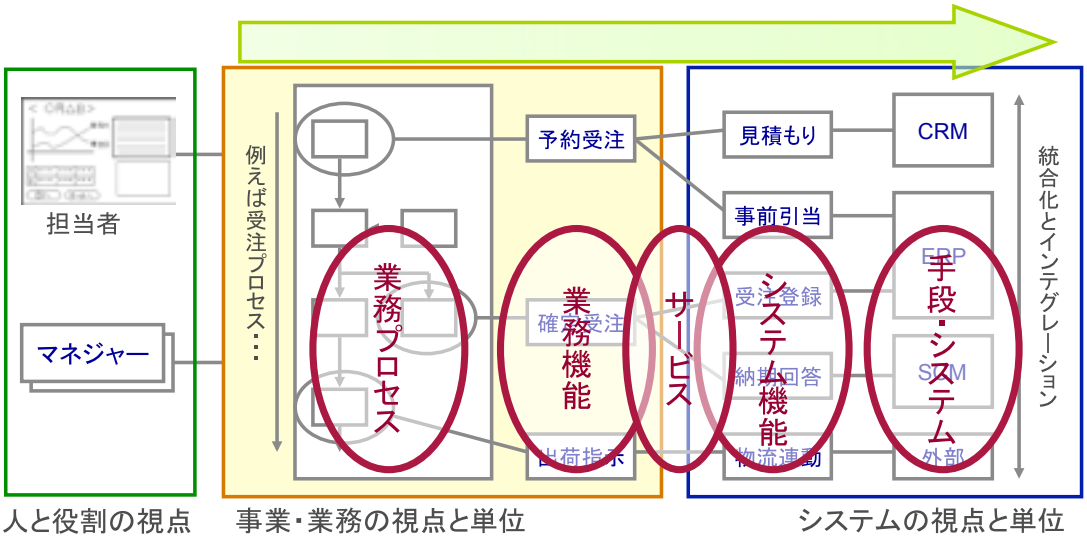


図 3.14-1 「事業視点」を実現するシステム像（SOA の構図）

従来の IT 化は「システムをつくる」部分を中心にしてきた。このため、２点目にあるような、業務プロセスを描き、システム要素を組み合わせてひとつのアプリケーションに実装する役割を担う人材が圧倒的に足りない。BPM への取り組みはここを担うものだが、システム要素を組み合わせ、ひとつの画面や機能へと実装（マッシュアップ）する役割を満たすこと、持続的な成長と改善に応え続けていくのに合った生産性と体制が求められることなど、新しい役割への期待と課題がある。

（3）事業と IT の一体化を担う“ワクワク感”

「What の変化」から多くの新しい役割が見えてくる。そのひとつに、データの専門家がある。求める姿を構成する要素にデータの存在がある。高度な分析を担うデータ・サイエンティストの存在。データ駆動のプロセスに使うパターンや閾値を判断するプロファイルの分析。マスターやデータの状態を介して、システムと業務プロセスの健全性を測るデータ・マネジメントの役割もある。

これまであった仕事でも、まったく異なる重要度と役割へと見直すべきものがある。そのひとつが、クラウドやモバイル、M2M などあらゆるものがインターネットにつながった状態を支える、ネットワークとセキュリティに関わる仕事だ。同様に、複数のシステムを組み合わせてひとつのプロセスや機能が動く、その複雑かつ高密度に統合された状態を運用する役割がある。

事業と IT の一体化も、クラウドやモバイルの利用も一様に、運用の難度を上げている。無策では事業継続性を弱める。運用技術の高度化と担う役割の重点化は、これからの正否を握る最重要課題である。統合こそ難しい。それは、移行やテストの技術にも共通する。

一方、マーケティングや商品サービスに関わる IT 化では、速やかにプロトタイプをつくり、市場でテストをし、さらなる策へ結びつけていくアプローチがとられる。つくったアプリケーションやサービスをソーシャル・メディアで広め、フィードバックをビッグデータとして分析する（図 3.14-2 参照）。

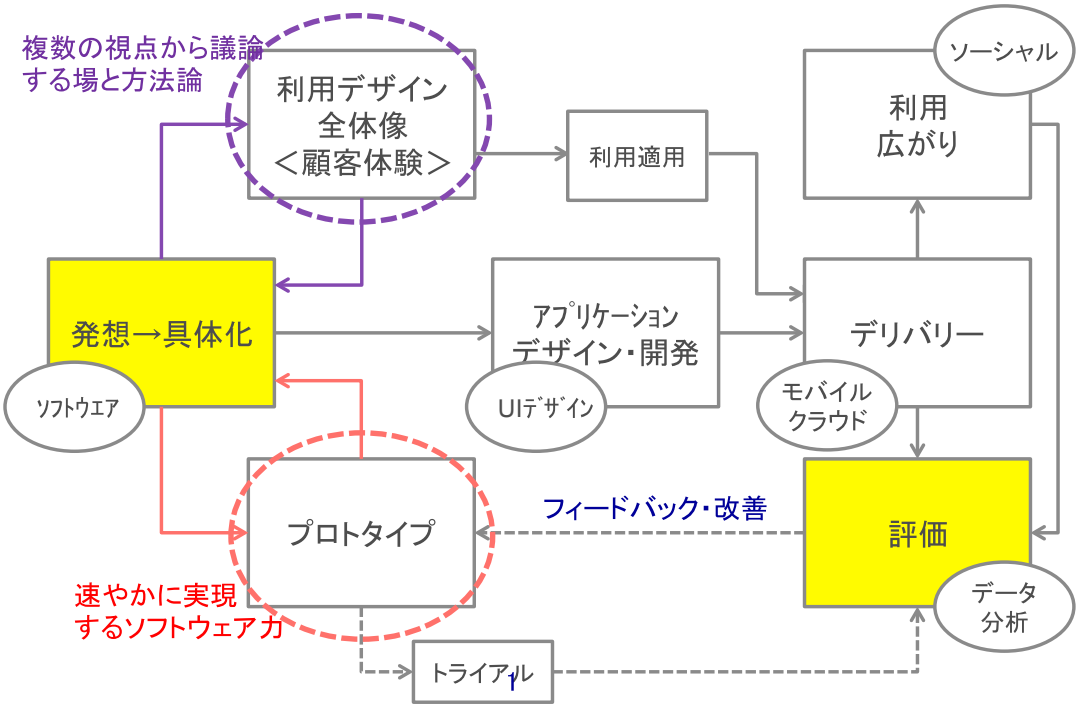


図 3.14-2 「事業視点」を実現するシステム像（SOA の構図）

アイデア段階から IT の観点を入れ、あらゆる段階で手段として IT を使う。アイデア段階には、一人の優れたプログラマーが大きく正否に寄与する。クラウドにある多種のツールを使って、利用者に広げ、評価をする知恵も要る。ここにも、新しい IT の役割がある。

企業 IT の変化は、役割の多様化をもたらしている。それは、個々の役割に高い専門性を求めるということでもある。多様な立場の専門家が集まって、立体型のチームをつくる。事業と IT の一体化は、IT に携わるあらゆる役割と事業との距離を縮め、実現の手応えを大きくしている。IT の力がそのまま、事業の競争力へとつながる。

クラウドやソーシャル・メディアによって、IT 化の手段とそれに関する情報が容易に手に入るようになった。Consumerization では、まず個人で使ってみて、その使用体験を企業へと持っていくアプローチがとれる。クラウドの多くは、試験利用を用意している。オープン・ソースや企業を越えたコラボレーションもある。オープン・データもある。

求める姿を求める条件で実現するために、知恵を絞り技術を駆使する。事業との一体化は、IT 本来の醍醐味を強調する。不確実を乗り越え、実現するおもしろさ。専門性を発揮し、チームでひとつのものをつくる充実感。もちろん、企業やプロジェクトによって、おかれている状況に違いがある。しかし、たとえ小さな一部分であっても、「What」から考えることで、実現像もそれに携わる仕事も違ってくる。その変化とワクワク感を体験し、共有していきたいと切に思う。

〔札幌スパークル株式会社　システム・コーディネータ　桑原 里恵 著〕

3.15　可能性を信じて、継続できるか

Open Forum2012 で私は概ね次のことをお話させていただいた。「自分の基本的価値観や姿勢が培われた環境」、「プログラマーからスタートして経営品質や人材育成にたどり着いた過程と、その間に携わった情報システムやその変遷」、最後に、問題提起として、自己の人生をどのように生きようとしているのかが「可能性と継続性に関係している」ということにわずかばかり触れた。

この Forum を終えてセッションのリーダであられた大島氏から「情報システムの構築・運用に 直接・間接に関わる分野で、是非若いうちに身につけておいて欲しいスキルや学業分野、関心を持って欲しいテーマなど」についてまとめるようにとのご要望が届けられた。

人は素質や気質というものがあるが、それを封じ込めて別の素質や気質を身に着けることが可能なのだ。封じ込めずにそのままで生きていくことができれば楽に違い無い。封じ込めなければならないか、封じ込めなくてもいいのか、それを左右するのは自分自身が何に価値をおいて、どのような生き方をしたいのかということにかかわってくる。

そしてどのようなことに価値を置いた場合でも、どのような生き方を望んだとしても、他人や社会に害悪を与えることはあってはいけない。この無限の宇宙の中でつながれてきた一つ一つの命の有難さを大切に思い、そう思うことですべてに包まれた全体としての自分の存在と、全体をいたわることで得られる心の安らぎや可能性への自信を感じることができる。

まず、身に着けておいて欲しいこと、それは自然とともに、友達や大人とともに、のびのびと楽しく遊ぶこと、体を動かし、大きな声をだし、語り・笑いあうこと。差別や上下や偏見が生まれる前の、人としての善と可能性が芽生えるビッグバンの季節を思う存分に過ごすことです。それは 137 億年の宇宙の歴史の中の一瞬のように、わずかな間でしかないけれども、あなたの基本 OS が育つ瞬間です。

そういう時期が無かった方は、今からでも多くのこだわりを捨て去るか封印して、功利的な生活から離れてみることがビッグバンにつながります。何者にでもなれる可能性を持っていた頃に戻れるということ、私は Open Forum2012 の最後のスライドの最後の行に、「ips 化する」と書きました。人はいつからでも可能性を秘めた始まりとしての存在なのです。

次に感動すべきことは、人類の歴史です。人類が成してきたこと、そして今の知識体系や文化・風土・社会システムとして機能していることへのシームレスな変化を空気のように体に吸収し受け止めることです。

私たちは多くの人の命の継続の結果の上に存在していることは言うまでもありません。地球誕生からと考えれば、４５億年の命の流れの先端に存在しているのが私たちです。人類の歴史の中で注目すべきことは人間の心と人間関係に関すること、言語に関すること、技術革新に関すること、社会・経済などの制度やシステムのことなど多岐にわたります。

そして、このような明日を実現できたらということを Nobi-Nobi と Waku-Waku として抱くことです。それを実現する対象の範囲は広ければ広いほど、実現の可能性が見えた時の歓喜予感が高まりますし、多くの人に共感と勇気と幸福を与えるものになります。このような「温故知新」あるいはイノベーションを実践できるには、人類が築いてきた知識体系とその意味、それを成し遂げた人々の生き様に感動し共感する自分というものを育み続け

なければなりません。継続は力です。

では、感動や共感、「温故知新」、新たな明日の実現ができるようになるには何をどうすればいいのだろうか。

私は微分を学習し始めたころに、微分係数の正負とその出現領域の個数と位置が関数の次数と形状に関係していることを何気なく見つけて使っていました。数学の授業のあるとき先生が傍にたってそれを見て言いました。「それは自分で考えたのか」と。「そうです。」と答えるのを聴くと教師はまた教壇に向かっていきそのまま授業を続けました。私は「自分で考えたのです。」と伝えることができた、ただそれだけのことが大変嬉しかったことを覚えています。キーワードは「自分で考えて、歴史上の解にたどり着く」なのではないかと思います。

このような学習や能力開発が行われることが人類の最先端の社会や経済に到達した日本人に必要なのだと思います。自律的に発見し、その快感を体感することが、新しいことを創造しようとする意欲を醸成していくはずです。知識は体系的に教える必要がありますが、教師による知識（結果）の一方的な提供ではそのような人材が育つ可能性は小さいと言えます。

高度成長、大量生産・大量消費、人口増加の時代の教育モデルは破綻しています。

未知なる世界に向かって必要となる人材は、「人々の幸福のために」を価値の中心におき、自律的・発見的・創発的に考え行動する人です。そういう人が現れると、それに感動し共鳴する人材が出現します。誰かがブレークスルーしなければならないのです。

知識はネットの中に充満しています。教師に教えてもらわなくても手に入れることは容易です。自分の能力や知識を補完してくれる人々もネットの中に存在しています。

ネットの中に存在しえないもの、それは実感です。動画や音声がいくら高解像度になったとしても、生身の人間が発する呼吸や温度を感じることはできません。従って、ネットを最大限活用するとともに、「一緒に風呂に入り同じ釜の飯を食う」ことで真に価値のある知識体系や実践につながっていくのです。

自然とともにあり、偏見やこだわりが生まれる前の原体験 / 原感覚を忘れてはいけません。有機的な存在として在り続けなければならないのです。

何のために働くのか、究極の目的は何か、何によってあなたは幸福感を得られるのか。

さて、IT を目指す学生や教師にとって心しなければならないことは、IT の分野は「技術変化が激しいうえに社会の変化の影響も受けやすい」「今後も複雑化・高度化傾向にある」「IT 以外に、人類が築いてきたあらゆる知識や社会システムについての理解が必要となる」「特別な専門知識・スキルを持たないと知的肉体労働者で頑張ることになる」「日常生活での役立ち度は低い、ビジネスとして役立てるには多くの人や組織との連携を必要とする」ということです。

「技術変化・社会変化」には、「消えゆくもの」としての変化と「消えないものとしての変化」に大きく分類されます。たとえば自然科学的な知識の変化の多くは「消えないもの」であり、社会・経済の変化に対応しているものの多くは「消えゆくもの」といえます。

「分類力」「汎化力」「類推力」「結合力」「システム思考力」「説明力」「計算力」「向上力」などのような基礎的な能力やスキルは表層的な変化に左右されない「消えないものとしての“変化”」といえるものですから、物事の本質を見失わないために常に意識して使い続けて欲しい「力」です。

桜の花びらが散るのを観て物のあわれを感じる「力」は上記の中の力のどれかに該当します。「一を聞いて百を知る」も同様の力です。「1 と＋（加算）の意味を知れば（決めれば）」無限に数を生成できるだろうことを、私たちは類推できます。その数を簡潔かつ限られたルールで表記したいという意欲（向上心）が、「0（ゼロ）」の発見によっ

て、位取り記法につながる。計算をするアルゴリズムができれば経済の急速な発展によって必要となった大勢の人による計算作業を自動化できますが、その原理を見事に教えてくれた「チューリング・マシン」は 1936 年に「変化（新しい理論）」として出現し、そしてコンピュータの基本原理として今日まで存続し、消えないものとしての“変化”であったことを証明している。

身近な力である「分類力」は物の体系図や、オブジェクト指向図、論理ピラミッド、企業の組織体系などあらゆるところに現れる、いわば人間が持つ「本能」ともいえる。

汎化力は人間の受容体としての感覚器官を通して得た感覚をことばで表現する。「冷たい、温かい」「固い、やわらかい」など、すべては「共通のことば」（共通であることが社会生活を通し実現できている音声と文字）に汎化されている。個別の言葉は組み合わされて新しい言葉を生み出す。組み合わせは物事による組み合わせが発端であることも、頭の中での思索や思考が発端であることもある。結果的には両者は融合する。

「風が吹けば、桶屋が儲かる」はシステム思考と言えなくもない。因果関係の複雑なループなのか、相互に影響を及ぼす関係なのか、そこにあたかもシステム原型があるかのように、現象を観察し分析し、そして再度組み合わせるといった力を使います。

そのシステムは「開放系」なのか「閉鎖系」なのか、視野（定義範囲）によって決まります。いわゆる中期経営計画は狭い時間軸上での計画です。今日の空腹を満たすことはさらに短い時間軸に生じ、しかも食欲がある限り（命ある限り）続くものです。

IT の基本原理と社会での応用モデルの構造を理解し感動し、未来創造に向けて情熱と力を継続することができるか、それを支援する指導力も問われています。

〔大成建設株式会社　社長室　情報企画部　企画室　主事　加藤修明　著〕

第４章　おわりに

4.1　結論

ソフトウェアエンジニアリングという領域が提唱されて 45 年経ち、ソフトウェアというものが、我々の生活、世界を支える無くてはならないものになっているにも関わらず、ソフトウェアづくりに携わっている人々からは、不条理、混迷、辛いといった悲痛とも言える言葉を耳にすることが多いのはなぜなのでしょう。どのような産業でも、日々の活動や業務ノウハウが蓄積され、成熟していきます。そしてそのことが、豊かな社会に繋がっていくことが自然な流れです。

ソフトウェアづくりにおいて重要な観点は、「抽象化」「自動化」「モジュール化」に整理することができます。ソフトウェアエンジニアリングの世界に「銀の弾丸」は存在しませんが、地道にこれ等３つの観点で方法論、ツール、アーキテクチャ、設計、仕様化等の技術開発を進めていくしか途はありません。図 4.1-1 は、筆者がソフトウェアに関わるエンジニア、研究者、経営者にヒアリングをして、ソフトウェアエンジニアリングの技術がどういった価値や効用をもたらすかをまとめたものです。詳しくは、拙作『ソフトウェア開発はなぜ難しいのか』（技術評論社 SE 選書 , 2009.11.25）に詳説しています。

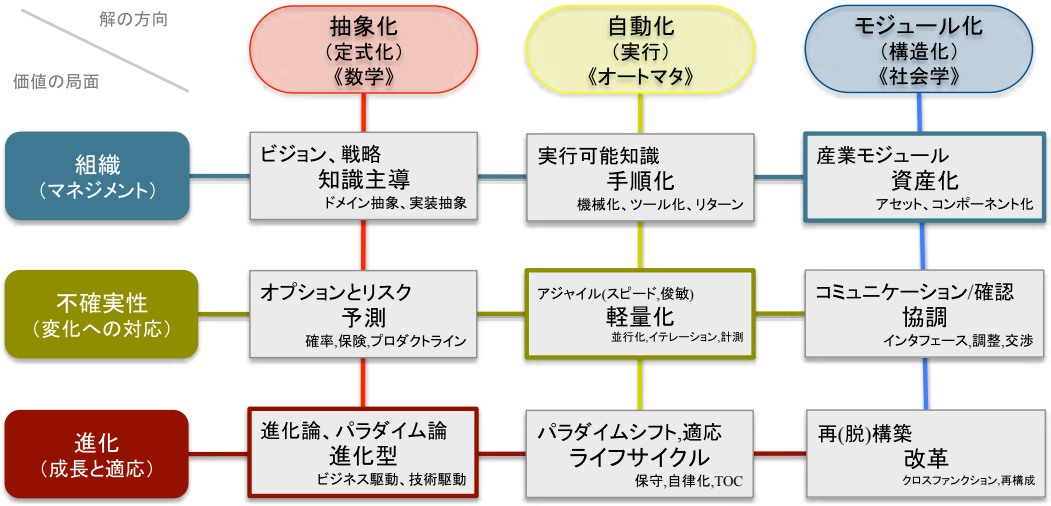


図 4.1-1　ソフトウェアエンジニアリングの価値と解の関係

抽象化（Abstraction）とは、問題を解決するために不要な情報を捨象（捨て去る）ことです。プログラミング言語の高級化も、抽象化の観点から見ると、ハードウェアやオペレティングシステムの個別のことを知らなくても、計算や操作の指示ができるように抽象化を図っているとみなすことができます。実世界側でも抽象化は重要です。経営者が経営上の意思決定をする場合には、個別の取引きや従業員一人一人の細かい業務情報は目に入れずに、経営問題に抽象化しているはずです。

自動化（Automation）とは、計算機や機械、さらには、人手によって網羅的・機械的に作業や業務を実行できるようにすることです。コンパイラによる言語処理や変換、ツールによる処理は典型的な自動化です。定型的な業務遂行も人手による自動化と考えてよいでしょう。システムに故障があった場合にこれを自動的に修復して実行し続けることができるようにする「オートノミック（Autonomic）」というコンセプトも提唱されたりもしています。究極的には、問題そのものを発見して自動的に解いてくれるような機械があればよいかもしれませんが、おそらくこれは永遠に SF の世界の話であり続けるでしょう。

モジュール化（Modularization）とは、ソフトウェアの構成要素を独立に管理できるものに分割することから、組織において分担が可能なように役割やルールを設定することです。プログラミングにおいて、サブルーチンやクラスを切り出すのもモジュール化ですし、会社の組織やチーム構成を定義したり、企業間のパートナーシップを決めるのもモジュール化による意思決定といえます。複数の人、チーム、組織が協調して仕事を進めるためには、役割分担をし、インタフェースとしての約束（コミットメント）や契約をすることになりますが、どういった組織構

成や役割分担がよいかは組織論の中心課題です。生産現場でのセル生産方式や、組織構成の系列の仕組みでの擦り合わせといった「産業アーキテクチャモジュール」の研究も進められており、ソフトウェアエンジニアリングの領域への適用も可能な面もあると考えられています。

本報告の第 3 章で述べた項目を、図 4.1-1 上にマッピングしたものが図 4.1-2 です。

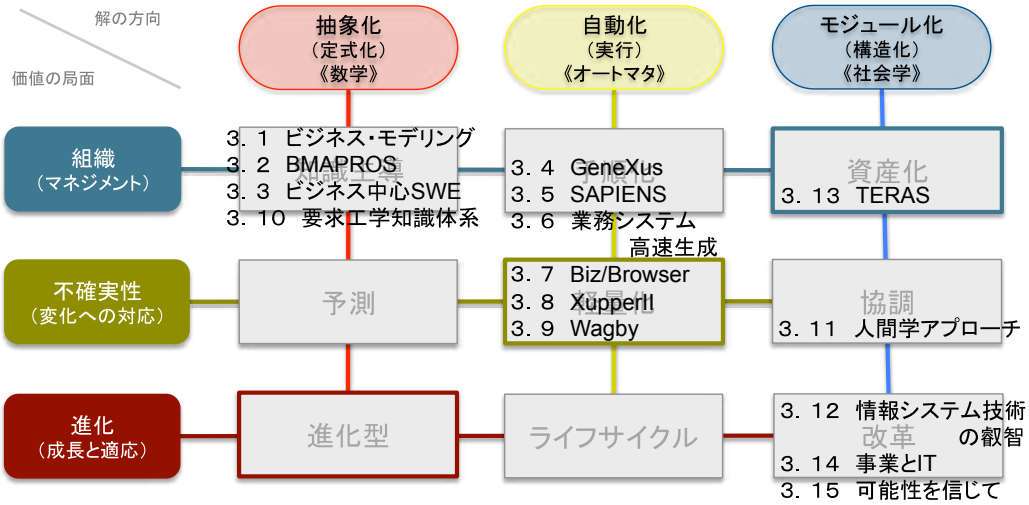


図 4.1-2　個別項目の価値 × 解へのマッピング

無論、本報告で採り上げているものは、ソフトウェアエンジニアリング領域全体からするとごく一部に過ぎませんが、ツールについては、「自動化」の技術として位置づけられます。対象領域を限定し、機械化・手順化できることはやり尽くされているように見えます（3.4, 3.5, 3.6, 3.7, 3.8, 3.9）。ここのところ急速に普及が進んできているアジャイルプロセスとの相性もよく、モデル駆動型の開発スタイルに近い様相になっています。すなわち、開発の効率化をツールによって補完することによって、以前より不確実性に対処し、システムに対する要求変化に俊敏に対処できるようになってきています。

一方で、非定形で人手に頼らなくてはならないのが、システムを取り巻く利用局面やビジネス面です。このあたりの定式化や知識体系の整備も進んできています（3.1, 3.2, 3.3, 3.10）。本来、ソフトウェアは人間の知を組み込んでいくものですから、そこに人間の日常活動、社会的行動、思考方法などが中心になっていくということも、いくつかの局面から見て取ることができました（3.11, 3.12, 3.13, 3.14, 3.15）。

本報告での調査対象は、一部を除き（3.13）、多くはエンタプライズ系の開発支援ツール、一般的な要求工学と標準知識体系、および、これ等の実践事例です。限られた調査対象ですが、大局的な観点で得られた知見は以下のようにまとめることができます。

一般的で統一的な技法や方法論は無く、多様化してきている

知識体系はいくつかの領域が統合・全体化してきている

ビジネスとの連動要求が増大してきている

このような徴候は、ソフトウェアの領域が成熟してきている証かもしれません。自動化ツールについても、前提としている方法論が何かを探りましたが、データ中心設計といった実装中心のものは対応させることができても、要求やドメインの分析については、個別具体で、実践領域や現場に密着した知識が中心になっていると思われます。システムを取り巻く状況も多様化しており、M&A による組織統合に伴うもの、既存のシステムを前提とした事業継続が中心となっているもの、クラウド活用を目指すものといった、従来の意味でのスクラッチ、初期構築型の開

発が少ないことも影響しているのではないかと推定されます。

こうして状況が多様化してくるということは、ソフトウェアやシステムに関わる人の役割も多様化して、複数の知識体系を横断的に活用するといった事態になってきています。開発と保守・運用プロセスを統合したり、ビジネス領域に拡大していくアプローチが一つの潮流になっています。

2010 年 8 月の時点での計画では、WG2 の目標として「より広い対象分野で適用できる新たな方法論の開発研究」が設定されており、これに向けての調査を行うのが WG1 の活動目標でしたが、上記状況を踏まえ、WG1 の結論としては、以下のような認識に至りました。

広範囲での新方法論開発という意味での WG2 活動は非常に困難、あるいは、ほとんど不可能である。

つまり、実践的ソフトウェア教育コンソーシアムの目指す活動として、3 年前に設定した WG 活動の計画のままでは限界があり、WG1 の活動を通じて得た認識に基づいて今後の活動アプローチについても再検討が必要ということです。

〔野木 , 大槻 著〕

4.2　今後の展開

本報告書のまとめをもって WG1 の調査活動は一区切りとなります。P-sec の WG 活動の全貌は、WG1 が調査、WG2 が新方法論の開発、WG3 が教育ということになっています。無論、調査を受けて方法論開発、その結果を教育にという逐次的な関係ではなく、それぞれ並行に推進していくのが自然です。従って、下図に示すような関係になります。ただし、前節に述べたように、従来の意味での WG2 活動は難しいという認識に至ったので、それを踏まえた上で WG2 の活動を展開していくことにしています。

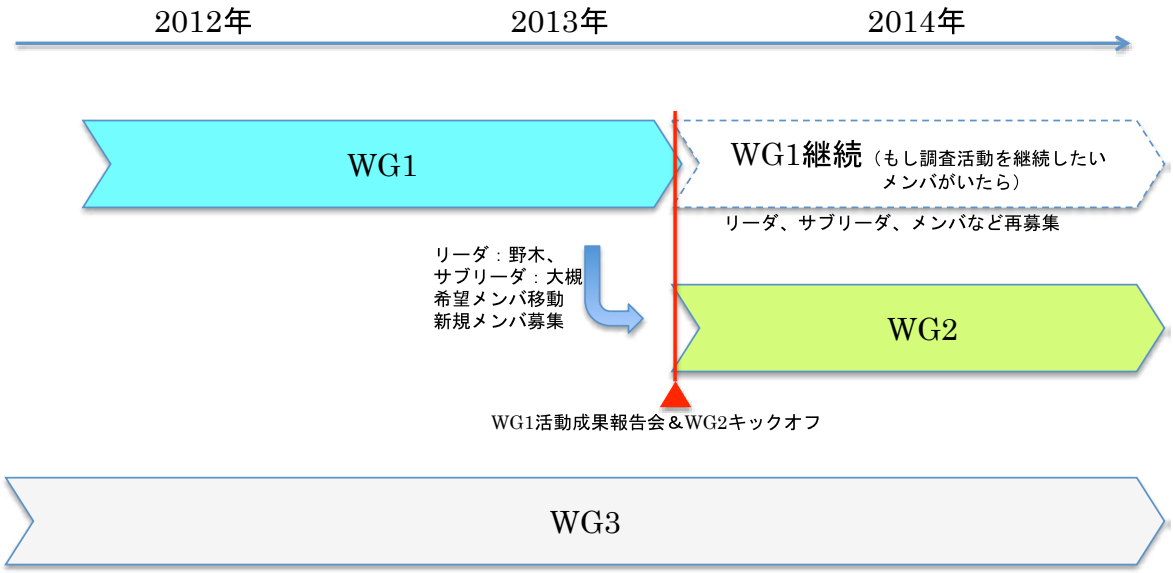


図 4.2-1　WG 間の関係

今後の活動の方向性を探るために WG1 の 2012 年度最終回（第 10 回）会合では、ブレーンストーミングを行いました。詳細は第 10 回会合の議事録（WG1-10-Meeting.doc）に記載されています。この会合の前半では、課題認識と称してさまざまなキーワードを出してもらいました。

- 超高効率化、デザインシンキング、コミュニケーション支援、着実な現実、断捨離、年を重ねるが楽しくなることなど
- エンタプライズ系、組み込み系といった企業内（組織内）システムの枠組みを超える期待感に基づく、幅広い課題認識など

これを集約して一つのメッセージとしてまとめると、以下のようになります。

> 今後の IT が扱うべきは、企業内の数値データや一方通行の情報配信だけではなくなる。また、システムの開発は専門の業者から利用者の手に戻さなくてはならない。
>
> IT が取り組むべき課題は、特定の組織業務を自動化し、利益や経営に貢献することから、社会と人を支え、生活の質を高めることへと進展してゆく。

さらに、会合後半では、キャリアについて討議して、以下のような状況でした。

・技術⇒安心、再生産伝承、開発から活用へ、などに集約され、最終的には「愛」に行き着く

・IT 系スペシャリストは今後の社会でも重要なキャリアであるが、人間工学スペシャリストのように、「人」が中心になる

これを総括すると、以下のようにまとめられます。

今後の IT 人材には、社会的コンセンサス作りをリードできるコーディネータ的或いはプロデューサ的な素養が強く求められるようになる。「モノ」を作ることから「人と社会」を作り上げていくことに変わっていく。

専門特化した人材が互いにプロとして分担・協力する環境で能力を発揮していく必要がある。自らが最新の技術を確立するだけでなく、その継承と人材の再生産も重要な役割となる。

上記状況を踏まえ、WG2 の活動概要を以下の通りに計画しています。

WG2 名称：

ソフトウェアの方法論に関する実践知と社会基盤

活動概要：

ソフトウェアの方法論・ツールに関する実践知を体系化し、それを蓄積・活用・伝承できる社会的な仕組み（知のプラットフォーム）を検討します。

活動内容

知のプラットフォームの検討：

ソフトウェアに関する実践知は、特定の技法、ツールの利用法などがあり、かつ、それ等は状況に応じ変化し、技術革新によって進化していくものです。このダイナミックな知を、蓄積・活用・伝承するための、社会的な仕組みを検討します。

これからの日本の置かれているビジネス環境、社会的な情勢、高齢化やシルバー活用、女性の活用、ワークシフト、人材育成、技術伝承、国の施策、新領域開拓、グローバル人材の輩出、これからの大学の役割、イノベーション、スタートアップといった変化に対応した方法論・ツールに関する知のプラットフォームの構築を目指します。

調査活動（WG1）との関係：

WG1 で収集した情報（報告書）をベースにして、これを活用しながら新しい取り組みをします。

ツールや方法論の調査を主眼とした活動は行いません。

教育（WG3）との関係：

今までの教育は、講師から受講者へ一方通行の伝承が中心で、限界があります。

普遍的な方法論構築を目指すより、社会的な仕組みや、知の共有の場を設定する方が現実的です。

〔野木、大槻 著〕

謝辞

2011 年 11 月から約 1 年半に及ぶ P-sec　WG1 の報告書が出来上がりました。

これだけ変化が激しく、またソフトウェアへの需要が非常に多様化している世の中において、特定のツールによらないモデリング手法、設計手法、開発方法論や、適用事例について、この 1 年半、調査分析を進めてきました。

特に規模が大きかったのは、2012 年 11 月に開催した　P-sec　OPEN　Forum でした。Forum の内容はこの報告書に大いに反映されています。御登壇いただいた、山本先生、羽生田先生をはじめ、パネルを引き受けていただいた　贊良則氏（ジャスミンソフト）、山内 啓悦氏（ウイング）、加藤 修明氏（大成建設）、中川 憲一郎氏（NK コンサルティング）、大熊 眞次郎氏（カシオ計算機）、桑原 里恵氏（札幌スパークル）に、この場をお借りして、感謝の意を表します。

報告書執筆メンバには、ツールベンダの方、ソフトハウスで、現場の開発に従事されている現役バリバリの方、大企業でソフトウェアエンジニアリングに携わる方、大企業卒業後に、コンサルや、大学にて教鞭をとる方、ユーザ企業において、適用を検討する方々など、本当に、さまざまなメンバが、それぞれの感覚で、それぞれの考えで、この報告書の執筆にあたりました。副委員長の大槻さんの、本当に、多大なる指導の下、アクテイブな皆さん、多忙な合間を縫って、このワーキングに参加いただき、執筆をしていただき、本当にありがとうございました。心から謝意を表します。

この報告書は、WG1 の活動のあかしとして、そして、WG2 という形になる今後の活動のベースとして、生かしていただけたらと願います。

＜編集委員 and/or 執筆者＞（敬称略・五十音順：所属は 2013 年 3 月時点のもの）

上原 誠（株式会社アイピーエル）
大北 龍彦（サピエンス・ジャパン株式会社）
大島 正善（MBC :Method Based Consulting）
大槻 繁（株式会社一（いち）：副委員長）
片貝 孝夫（片貝システム研究所（アクシスソフト　エバンジェリスト））
加藤 修明（大成建設株式会社）
亀田 弘之（東京工科大学）
黒川 利明（個人）
桑原 里恵（札幌スパークル株式会社）
斉藤 忍（NTT データ）
高橋 俊夫（ケンシステムコンサルテング）
田口 眞美（個人）
田島 孝（サイバー創研　：事務局）
佃 均（個人）
鶴保 征城（学校法人・専門学校 HAL 東京）
中山 浩一（株式会社 CIJ）
西住 浩樹（王子物流株式会社）
野木 秀子（株式会社 CIJ：委員長）
樋山 証一（株式会社ウイング）
福島 祐子 (日本ユニシス)
萩原 淳（NTT データ）
藤川 恵一（個人）
三輪 一郎（株式会社プライド）
渡辺 政彦（キャッツ株式会社）

2013 年 5 月吉日　WG1 リーダ　野木秀子


Pse
c
W
g
1
2013
March