CRDS-FY2014-WR-08

システム科学検討会ワークショップ報告書

システム構築方法論
ー現在の到達点と今後の課題ー

平成26年6月23日（月）開催

主催: 独立行政法人科学技術振興機構研究開発戦略センターシステム科学ユニット
協力: 独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター

独立行政法人科学技術振興機構 研究開発戦略センター
Center for Research and Development Strategy Japan Science and Technology Agency

エグゼクティブサマリー

独立行政法人科学技術振興機構（JST）/研究開発戦略センター（CRDS）は、JST の研究開発戦略を立案するとともに、我が国の研究開発の推進に資する基礎データおよび知見の収集とそれに基づく戦略的研究分野の提言を行っている。その中で、システム科学ユニットは、「『システム構築は難しい』それゆえに非効率で非経済的な活動が、社会の中で科学技術や経済界の発展の足かせになっている。この難しさを解明し、システム構築の方法論を示すことが一つの課題となっている。」との認識の下に、独立行政法人情報処理推進機構（IPA）/技術本部ソフトウェア高信頼化センター（SEC）の全面的な協力を得て、平成 25 年 11 月に有識者委員 8 名からなる「システム科学検討会」（主査：鈴木久敏特任フェロー（当時））を立ち上げた。この検討会では、JST-CRDS と IPA-SEC が互いに持つ研究情報を共有し、IPA-SEC が長年蓄えてきたソフトウェア開発技術の課題や難しさを学ぶことを通して、ソフトウェア開発技術を含むシステム技術の研究課題を探り、システム構築の方法論を探索することを目的としている。約 8 ヶ月、都合 8 回の検討会活動であったが、最後の第 8 回検討会は主としてソフトウェア開発技術を専門とし、かつシステム構築にも造詣が深い実務家や研究者である外部講師 5 名を招聘し、有識者委員と討論するワークショップ形式で、システム構築方法論の現在の到達点と今後の課題を探ることにした。

この報告書は、今後のシステム構築方法論の研究開発の方向性を確認するため、本ワークショップの記録をまとめたものである。5 名の講師から、それぞれの貴重な経験に裏付けられた報告を受けた後、総合討論を行い、主な結論として、「システム開発を巡る環境変化」、「システム構築方法論の世界の潮流」、「システム構築方法論を含む日本のシステム科学技術が抱える課題と対策」など、以下のような知見を得た。

(1) システム開発を巡る環境変化
- システムはますます大規模化、複雑化している。
- システム開発は、単に技術の問題だけでなく、価値創造と捉える必要がある。

(2) システム構築方法論を巡る世界の潮流
- ISO15288 など、システムズエンジニアリングの国際標準の策定が進められている。
- SEBoK のようなシステムズエンジニアリングの標準知識の体系化が進められている。

(3) 日本企業の課題
- 日本企業は技術の自前主義が強く、技術のガラパゴス化が進み、現在の国際標準と乖離してしまっている。
- これがグローバル市場での日本企業の産業競争力の低下を招いている。

(4) 日本社会の課題、学術界の課題
- システムとソフトウェアを同一視し、システム科学技術が、ハードウェア系、ソフトウェア系、さらにはマネジメント系を含めた総合力であるとの視点に欠けている。
- システム関連部門が機械工学、情報処理などそれぞれの個別領域毎の学会別に分属し

ている。

・システム科学技術という固有の領域の存在、及びその重要性に対する認識が低い。

(5) 日本の人材育成面の課題

・国内大学の工学教育が学問のための学問を志向し、産業競争力を高める人材の育成という視点に欠け、産業界において即戦力となる人材を育てていない。

・システム工学科等においても SEBoK のような世界標準を前提として教育が行われていない。

(6) 日本のファンディング政策の課題

・学問的深さを追求するあまり、顧客価値を生み産業化に繋がる領域にファンディングできる仕組みがない。このためイノベーションが起こらない。

・解析と異なり、設計という活動は解空間が広過ぎて再現性が出ない。そのような特性を考慮し、設計分野にもファンディングできる施策が必要である。

・システム分野がファンディングの対象となるには、システム分野の研究者コミュニティもシステムが世の中をこう変えるというビジョンを示すことで、システムの重要性について社会的認知を得る努力が必要である。

(7) システム構築方法論の課題

・単にシステム構築のプロセスや技術についてのメソッドエンジニアリングに留まらず、開発プロセスの管理やシステムのライフサイクル全体を見据えた管理等のマネジメント系、さらには上流工程である価値創造やビジネス展開を含むシステム全体のストラクチャを決定するマネジメント系まで含めた概念へ拡張されなければならない。

(8) 日本が抱える諸課題に対する対策

・国内のシステム関連の研究者・実務家が集う研究集会の開催、システム関連学会の設立、システム科学技術先進諸国との国際連携の推進が必要であり、ファンディングを含め官がそれを支援する必要がある。

・産業界は国際標準である ISO15288 への早期転換が不可欠。日本の独自性を追求することを止め、まだ国際標準が確立してない、より上流の価値創造領域で、日本発の国際標準策定に向けた活動を強化するべきである。そのために産学官並びに異分野を糾合する研究活動とその政策的支援が必要である。

目　　次

1．ワークショップの概要

1.1 開催趣旨

システム科学検討会は、独立行政法人科学技術振興機構研究開発センター（以下、JST-CRDS）システム科学ユニットが独立行政法人情報処理推進機構ソフトウェア高信頼化センター（以下、IPA-SEC）の協力を得て発足した。検討会では、両者の共通の関心事である「システム構築」という難しい課題を共同して解決していく方策を探るため、有識者委員8名を集め、およそ1カ月に1回の頻度で有識者委員各自の成果・経験を持ち寄り、また必要に応じて委員以外の報告者を外部に求め、過去7回にわたり報告と討論を中心に検討を重ねてきた。

今回の会合はその最後8回目の検討会ということで、従来とは少し形態を変えワークショップ形式で、検討会の外部の方でシステム構築に関連の深い研究や実践を進められている講師の方を5人お招きし、報告をお願いした。報告を踏まえ講師の方を含めた有識者委員との討議を通じて、システム構築方法論の現在の到達点と今後の課題を明らかにし、今後のわが国の科学技術政策に反映させることが目的である。将来の検討の材料として活用するため、ワークショップでの議論と成果を取りまとめたものが本報告書である。今回のワークショップでシステム科学検討会としては一つの区切りを迎えるが、検討会やワークショップでの議論の成果を今後どのように展開していくべきか、JST内部でもこれから議論していきたいと考えている。

本ワークショップ開催の背景としては以下のような認識がある。世の中では一般に、「ソフトウェア」と「システム」が混同されていることが多い。特にわが国ではソフトウェアエンジニアをシステムエンジニアと呼び習わす慣習があるため、その誤謬が蔓延している。本来、ソフトウェアはシステムの目的を達成する一つの要素に過ぎず、システムの機能を発揮し、付加価値を実現する目的でソフトウェアが作られている。今日、あらゆるシステムや製品の機能の多くがソフトウェアで実現されていることが多いため、混用する傾向に一段と拍車が掛かっている。歴史的には、システムの概念の方が世の中に先に現れたのに、その概念の捉え難さから世間一般にはその重要性や有効性について十分に認識されないまま、後発のソフトウェアの方が急速に浮上・拡大し、進歩し、世間にその存在が認識されている。故に、我々システム側の関係者はもう一度ソフトウェアに学ぶ、特にソフトウェア開発分野で発達した技法に学ぶ必要がある。

こうした認識を背景に、本ワークショップではソフトウェア開発の難しさを学ぶことを通して、システム技術の研究課題を探索したい。それに加えて、ソフトウェア開発の中で標準化された諸技法、本ワークショップで具体的に採り上げることができたものは、形式仕様、Agile 開発、モデリング、要求工学、妥当性確認であるが、それらを通してシステム構築に求められる研究要素を抽出することを狙っている。

外部講師としてお招きしたソニー株式会社栗田太郎氏には、日頃、我々が電子マネーとし

て良く使う FeliCa の開発で用いられた形式仕様について、株式会社アッズーリ代表取締役濵勝巳氏には、環境変化に迅速に対応して機能拡張を図ることが可能な軽量なソフトウェア開発手法である Agile 開発について、それぞれ実務家としての経験を踏まえてご講演をしていただく。後半は、慶應義塾大学西村秀和教授にシステム構築をモデルベースで進めるシステムズエンジニアリング MBSE と SysML 言語との関係を、NTT データから名古屋大学に移られた山本修一郎教授にはシステム構築の上流工程で社会のニーズを的確に捉えていく要求工学を､最後に神奈川大学木下佳樹教授には Verification と Validation の違いを踏まえて、構築したシステムが社会のニーズにきちんと合致していることを確認する妥当性確認技術について、最新の動向をご講演いただく予定である。

総合討論では、ご講演内容への質疑を通して、現在、システムが直面する課題と解決策について議論したい。さらに、ソフトウェア開発とシステム開発の 2 つの間の同質性、異質性について、またソフトウェア開発技法の中でシステム開発に使えるものと似ているが少し違い使えないものについても議論を重ねたい。その結果、システム構築方法論として、現時点でどこまで到達しているのか、あるいは今後の課題は何かということを探ることが、本ワークショップ開催の趣旨である。

1.2 ワークショップ開催概要

【開催場所・日時等】

場　所：(独）科学技術振興機構東京本部別館 4 階会議室 E
日　時：平成 26 年 6 月 23 日（月）13：00～17：30
主　催：独立行政法人 科学技術振興機構 研究開発戦略センター システム科学ユニット
協　力：独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター
統一テーマ：システム構築方法論 －現在の到達点と今後の課題－

【プログラム】

13:00-13:05　開会挨拶
木村 英紀　（独）科学技術振興機構研究開発戦略センター上席フェロー
13:05-13:10　ワークショップの趣旨説明
鈴木 久敏　（独）科学技術振興機構研究開発戦略センターフェロー
13:10-17:25　講演
（13:10-13:50）「モバイル FeliCa の開発における形式仕様記述手法 VDM の適用
～システム開発への形式手法の適用による品質の確保～」
栗田 太郎 氏　フェリカネットワークス株式会社開発部 2 課
Q&A

（13:50-14:30）「Agile 開発コンセプトとその動向」
濵　勝巳　氏　（株）アッズーリ代表取締役

Q&A

（14:30-15:10） 「モデルに基づくシステムズエンジニアリング―MBSE と SysML―」
西村 秀和 氏　慶應義塾大学大学院システムデザイン・マネジメント研究科
教授

Q&A

（15:10-15:25） 休憩

（15:25-16:05） 「社会ニーズ把握のための技術－要求工学の動向－」
山本修一郎 氏　名古屋大学情報戦略室教授

Q&A

（16:05-16:45） 「妥当性確認とアシュランスケース」
木下 佳樹 氏　神奈川大学理学部情報科学科教授

Q&A

16:45-17:25 総合討論

17:25-17:30 閉会挨拶
松本 隆明　（独）情報処理推進機構技術本部
ソフトウェア高信頼化センター 所長

18:00～20:30 懇親会


富川 弓子
久保 忠洋


藤井 新一郎

1.3 ワークショップ参加者一覧

【外部講師】

氏名	所属・役職	備考
栗田太郎	ソニー(株)FeliCa 事業部モバイル設計部 2 課 統括課長 フェリカネットワークス株式会社開発部 2 課	
濵　勝巳	(株)アッズーリ 代表取締役	
西村秀和	慶應義塾大学大学院システムデザイン・マネジメント研究科 教授	
山本修一郎	名古屋大学情報戦略室 教授	
木下佳樹	神奈川大学理学部情報科学科 教授	

【委員及びオブザーバ等】

氏名	所属・役職	備考
鈴木久敏	JST-CRDS システム科学ユニット 特任フェロー(～2014.3) JST-CRDS システム科学ユニット フェロー(2014.4～)	主査/事務局
大島啓二	日本科学技術連盟 嘱託	委員
田名部元成	横浜国立大学大学院国際社会科学研究院 教授	委員
白坂成功	慶應義塾大学大学院システムデザイン・マネジメント研究科 准教授	委員
松本隆明	情報処理推進機構技術本部ソフトウェア高信頼化センター 所長	委員
木村英紀	JST-CRDS システム科学ユニット 上席フェロー	委員
久保忠伴	情報処理推進機構技術本部ソフトウェア高信頼化センター 調査役	オブザーバ
新谷勝利	情報処理推進機構技術本部ソフトウェア高信頼化センター 連携委員	オブザーバ
茅 明子	JST 社会技術研究開発センター アソシエイトフェロー	オブザーバ
岡山純子	JST-CRDS 海外動向ユニット フェロー	オブザーバ
己斐裕一	JST-CRDS 政策ユニット フェロー	オブザーバ
藤井新一郎	JST-CRDS システム科学ユニット フェロー	オブザーバ
シン・ジャワ	JST-CRDS システム科学ユニット フェロー	オブザーバ
舩橋誠壽	JST-CRDS システム科学ユニット 特任フェロー	オブザーバ
富川弓子	JST-CRDS システム科学ユニット フェロー	事務局
小野里実	JST-CRDS システム科学ユニット フェロー(2014.5～6)	事務局

2．講演概要

2.0 ワークショップの趣旨

鈴木久敏 （JST-CRDS フェロー）

それではワークショップを開始したいと思います。最初に私の方から本日のワークショップの趣旨説明をさせていただきます。

まずワークショップの前に、この検討会の設置の趣旨ですが、これまで JST-CRDS と IPA-SEC が、両機関が所有する研究情報を共有しましょうとの趣旨で協力して、システム科学検討会を 7 回ほど開催して参りました。ソフトウェア開発技術や社会の中のソフトウェア構築について、その課題解決に向けて一緒に議論して行きましょうという狙いです。さらにその中で新たな課題の発掘や今後の研究開発戦略に役立てれば好都合、ということで検討を始めました。

JST-CRDS 側の立場から言うと、ソフトウェア開発そのものよりも、ソフトウェア開発の難しさを学ぶことを通して、システム技術の研究課題とシステム構築方法論を探索したいという期待があります。さらにシステム構築方法論の標準化の中でモデリング技術や可視化技術など、アカデミアに求められる研究要素を抽出したい。こういうものを抽出して行き、これから JST として振興していくべき分野を探索していきたいということでございます。

今日は第 8 回になりますけれども、これまではお示ししたスライドのような形で議論をして参りました。

次に本ワークショップの趣旨です。背景ですが、我々としては、「ソフトウェア」と「システム」というのがどうも混同されていると考えています。違うと言うと語弊がありますけれども、本来、ソフトウェアはシステムの目的を達成するための一つの要素であり、あるシステムの機能を発揮し、付加価値を実現するためにソフトウェアを使っている。ただ、システムの方が歴史的には先に世の中に現れたのに、後発のソフトウェアの方がずっと急速に拡大し、進歩している。だから我々システム側はもう一度ソフトウェア開発分野に学ぶ必要があるだろうと考えています。

ということで、本ワークショップの狙いは、ソフトウェア開発の難しさを学ぶことを通して、システム技術の研究課題を探索したい、ということになります。それに加えて、ソフトウェア開発の中で標準化された諸技法、本日の場合には形式仕様、Agile 開発、モデリング、要求工学、妥当性確認などを通して、システム構築に求められる研究要素を抽出したいということでございます。

最初に 5 人の講師の先生方にお話ししていただきますけれども、その後の総合討論では、次のことを主たるテーマに議論をさせていただければと思います。「ソフトウェア開発とシステム開発との関係」と書きましたが、その 2 つの間の同質性、異質性はどのようなところにあるのだろうかと。さらに、ソフトウェア開発に使われる技法の中で、システム開発に使えるものと使えないもの、と言うと語弊がありますけれども、ここは少し違うのではないかと

いうところを明らかにしていきたいと思っております。その結果、システム構築方法論として、現時点でどこまで到達しているのか、あるいは今後の課題は何かということを探ることが、このシンポジウムの趣旨でございます。

2.1 モバイル FeliCa の開発における形式仕様記述手法 VDM の適用
～システム開発への形式手法の適用による品質の確保～

**栗田太郎 （ソニー株式会社 FeliCa 事業部モバイル設計部/
フェリカネットワークス株式会社開発部２課統括課長）**

モバイル FeliCa のソフトウェア開発における形式仕様記述手法適用事例を紹介する前に、「システムの品質確保をしていく上でソフトウェアはその中の一要素にすぎない」ということを最初に申し上げておきます。つまり形式仕様記述手法を用いて難しいソフトウェアを作るのではなく、ごくシンプルな仕様の部分を形式仕様記述手法で簡単に済ませ、余った労力をシステム開発の方に向けるという話になります。ここでは、大人数のプロジェクトが一連の流れにおいて品質を維持し続けていく中で仕様はどのような役割を果たすのか、おサイフケータイのソフトウェア開発における形式仕様記述手法適用事例を基に見ていきます。

フェリカネットワークス（株）はおサイフケータイを実現・運用するために、NTT ドコモと JR 東日本がソニーと共に出資して生まれた会社です。おサイフケータイのさまざまな用途を実現するセキュリティ LSI の開発は非常に社会的責任が重く、システムの信頼性・安全性に対する関心が高まる一方、リーマンショック以降の人員削減や納期短縮の影響から、より効率の良い開発が求められましたが、ソフトウェア開発の現場は曖昧な仕様に起因するトラブルの多発を何とか解決したいという状況でしたので、形式仕様記述手法導入の提案に反対する人はいませんでした。

数ある形式手法の中から仕様策定段階から動作が可能でモデル化から動作確認まで適用できるモデル規範型の VDM を選定し、外部仕様書と呼ばれる部分について VDM++言語で記述、VDMTools で構文チェック・課題生成・デバッグ支援等を行いました。これにより厳密な仕様の策定と記述の継続、仕様を生かしたイテレーティブな開発プロセスの確立、仕様の多方面からの精査、仕様のテスト、コミュニケーションの促進が可能になりました。

そして「仕様策定」→「設計・実装」→「テスト」を繰り返す開発プロセスを組み立てて回した結果、バグ分析では仕様関連の不具合を 2%に抑え、仕様は非常に良く書けたという評価になりました。一方で仕様を読みこなせないことによる不具合も見受けられ、「読ませる仕様の記述」が課題となっています。

このように形式仕様記述手法は上流工程、特に仕様策定工程における記述と検証、成果物の品質確保に直接的な効果があります。また、自分たちが開発するドメインに対する理解が深まり、コミュニケーションの活性化、汎用的な技能向上という間接的な効果も生まれます。

＜質疑応答＞

○鈴木　ありがとうございます。質疑の時間が 4 分ほどありますので、ご質問をお受けしたいと思います。ご意見はこの後の総合討論でお話ししていただいて、今は簡単に栗田さんに答えていただける質問のみをお願いします。

○栗田　ちなみにこれが最新版の仕様書です。これを仕様書と呼ぶべきか分からないですけれども、わが社では少なくともこの情報をすべての活動の基準としていて、日本語や英語で書かれた仕様書はこれ以外にありません。ですからこれを読まない限り、仕様については一切分からない状態になっています。

○白坂　あくまでも仕様ということは、中身の設計構造には関わらないような情報しか書いてないということですね。

○栗田　そうです。抽象度を保って、自分たちが決めた仕様はこうですよということで書いており、その実現方法はどうでも良いということになっています。実際に FeliCa を作るときは法則化をしなければいけないので、この仕様書には当然含まれていない新たなことを考えて、それを実践的に加えていく必要があります。

○大島　ここで仰っている仕様は、どのように作るかという仕様のように思えます。普通はまず使う側の要求仕様があり、その視点で要求を正しく満たしているかという問題と、それをどう作るかという問題に分けて考えます。後者の方はこれで良く分かるのですが、前者（要求の把握）における悩みはあまりないのですか。

○栗田　その話をすると結構苦しいのは、私どもはメーカなので、お客さんは自分たちなのですという話があって、想定するお客さんはいらっしゃるわけですし、例えば JR さんとかドコモさんともいろいろな折衝や調整はしなければいけないのですが、基本的には要求仕様がお客さん側から結構細かく出てきて、それを実現するための仕様を作り込むというよりも、自分たちがこういうものを世の中に出したいという願望や希望や構想があって、それを現実化するという類いのプロジェクトなので、何とかできていますという話です。それは世の中に行くととてもたくさんある質問です。例えばうちの会社はシステムを受注して開発していて、ある会社の社員管理システムを造ることになったとき、「これを適用した場合はどうしたら良いですか？」と聞かれるので、困ってしまうこともあります。

ただ、それでも分かっていることは、自分たちが造るための仕様を厳密に書いていくことによって、お客さんはこれを読まないかもしれないですけれども、自分たちが分かったことを日本語に置き換えて伝えることによって、それはお客さんにとっても相当価値があるのではという話になります。

○岡山　この仕様は 5 日間の研修を経れば理解可能なものなのでしょうか。

○栗田　理解できない人は退場してもらうことになりますが、これまでの数百人で本当に駄目な人は 1 人だけでしたので、それほど難しくないです。

○舩橋　　この上位に概要仕様書があるのですね。
○栗田　　はい。
○舩橋　　それは UML で書いたのですか。
○栗田　　そうです。あとは書き散らしたパワーポイントとかいろいろなものでできていますが、私も含めて誰もメンテナンスをしていないので、情報としてただあるというだけです。
○久保　　客先とのコミュニケーションはこれでやるのですか。
○栗田　　手元には一応持っていますが、お客さんから「これはバグですか？」と聞かれた時に、従来ですと日本語の仕様書を読んで書いてなければプログラムを見て、確かにその動きを確認するのですが、それがそうなってしまったのか、そうしたのかが分からないので困るわけです。しかしこのケースですと、ここに書いてあれば「仕様です」、書いてなければ「自分たちの実装誤りです」と明確に言えるので、お客さんとコミュニケートする時にも、仕様書とプログラムは両方厳密に書かれているので、その 2 つの情報があることは確かに役に立ちます。

しかしそのことと、これをお客さんが読むかどうかはまた別です。ただ、非常にマニアなお客さんもいらっしゃるので、そういう方にこれを見せると「私だけにこんな情報を開示してくれるのですか」と喜ばれて、好評なときもあります。
○松本　　品質が向上したという説明があったのですが、生産性の面ではどうですか。
○栗田　　まったく同じプロジェクトをやっているわけではないので、その比較は難しいです。
○松本　　FeliCaで形式手法を使わずに以前開発したということはなかったのですか？
○栗田　　ありますが、仕様の規模もメンバーも違います。仕様の生産性に関して言うと、うちは 10 年間で 3 回ぐらいプロジェクトをやったのですが、管理工数とかミーティングとか顧客との仕様調整をすべて含めて、1,990～2,100 行／人月ぐらいで仕様書を生産しています。実装もだいたい同じような感じで 2,200～2,300 行／人月ぐらいになりますが、これが高いのか低いのか分からないです。ある企業の例では全社でやると 60 行／人月ぐらいとのことです。
○岡山　　1,990 のところが 60 ということですか。
○栗田　　そうです、社長まで含めて全部で割ると。
○岡山　　なぜそこまで･･･。
○栗田　　それだけミーティングとか、調整とか、パワポを作るとかが、多いということです。そのことが正しいかどうかは分からないですけれども、コードで割るとそういうことで、生産性とは何ですかという話ですが。
○鈴木　　どうもありがとうございました。総合討論でもいろいろ質問が出ると思いますが、どうぞよろしくお願いいたします。
○栗田　　ありがとうございました。

（拍手）

2.2 Agile 開発コンセプトとその動向

濵 勝巳 （株式会社アッズーリ代表取締役）

2001 年に提唱されたアジャイルソフトウェア開発宣言は大きなインパクを与え、これを中心にいろいろな人たちが日本でも活動しました。従来のシステムは、設計図を作り、部品を造り、結合して全体を造って、出荷するというものですが、アジャイルプロセスシステムは、種から双葉が出て、そこから葉が茂り、最後に花が咲き、出荷するというイメージです。お花屋さんではいろいろなタイミングで商品が売られているように、アジャイルプロセスのベースは、「育てる」という考え方で、「常に全体を意識」して、「常に出荷可能」であるという、この 3 つにあり、最初の標準化どおり進めるのではなく、状況に合わせて常に考えていくことが重要なのです。アジャイルプロセス導入のコンサルティングをするとき、多くの方がメトリックスを取りたがりますが、アジャイルはその都度考えながらプラクティスを変えていくものなので、過去のデータは何の意味を持たない。それがアジャイルプロセスというものなのです。

プロセスがいくつかのプラクティスを持ち、このいろいろなプラクティスを行うことで一つのプロセスになっているのですが、プラクティスというものは少なければ少ないほど良いのです。最初は見本のような形でいろいろなことをやっていきますが、だんだんやっていることを意識させなくなり、実は何のプラクティスもない、標準化もない、でも品質の良い生産性の高いソフトウェアが作れるプロジェクトチームだった、というのが一番理想的な形です。しかしそれは難しいので、状況を見ながら適切なプラクティスを導入し、チームをうまく作っていくことが大事になります。

アジャイルの普及啓発を目的としてアジャイルプロセス協議会が設立されてから 10 年以上になりますが、アジャイルに興味を持つ人たちの意識が、開発者の視点で考える狭義のアジャイルプロセスから、社会全体の視点で考える広義のアジャイルプロセスへと変わっていることが、活動を通して見てとれます。この流れの中で、「新ソフトウェア宣言」というものが出されて、もう一度ソフトウェアを見直そうという動きが出てきています。これまでどうしても工業的な部分をベースに考えられてきましたが、ソフトウェアによって知を働かせ、人々の社会的な活動を生んでいくことを、そろそろ考え始めなければいけないと感じています。

最後に、アジャイルプロセスに対するいくつかの誤解の中から大事なポイントを一つ、それはゴールに対する誤解です。アジャイルは課題解決のためのプロセスなので、ソフトウェア開発がゴールではありません。従ってアジャイルプロセスはイノベーションを生むものでは決してないということです。

＜質疑応答＞

○鈴木　　では、質問をお願いします。実は最初に言っておくべきでしたが、そちら側に並んでいる方々は皆さんソフトウェア専門家の方々ですけれども、こちら側は必ずしもソフトウェアを良く分かっていない方々もいますので、アジャイル

とはどういうものか、ひとことで言っていただけますか。

○濵　常に考え続けるということです。

○鈴木　日ごろ開発しているときに、全体の機能を持たせるようなものを少し造って、またそれを見直して造り直すというイメージで良いですか。

○濵　ものづくりであると考えない方が良いかもしれません。ものづくりは一つの結果であって、どのような価値効果を生むのかということが前提で、その中でソフトウェアを作るときは、それに必要なソフトウェアを作るということで、別に八百屋さんから買ってきたものに価値があれば、八百屋さんから買ってくることをやれば良いわけです。ですから広義のアジャイルと言った意味は、ソフトウェアだけに拘らずにやっていきましょうということです。

○鈴木　そこのところは分かるのですけれども、アジャイルというものの考え方がいまひとつピンと来ないのです。いつでも出荷できるという視点は、普通のものづくりやソフトウェア開発ではあまりないということは良く分かるのですが。

○濵　少しずつやっていくというと、上の部分の葉っぱを1枚作るところも少し作るということですが、葉っぱを1枚作っても花としては完成されないので、それでは駄目で、やはり少し作るのであれば双葉が開くようにして、それで一つ完成されているものにしなければならないということです。

○大島　開発プロセスが waterfall であれば、仕様を最初にきちんと決めて、計画もしっかりやって、エンジニアリングのサポートもそれなりに用意して、あとは「決められたとおり造ろう！」というやり方になります。アジャイルはその対極にあって、造る過程でも、新たなニーズなどを柔軟に取り入れながら目的に向かっていくようなイメージを持っているのですが、その考え方で良いでしょうか。

○濵　何を造るかを決めて造れというものではないのです。

○大島　アジャイルは！

○濵　アジャイルも最初に決めなければいけないのです。まずはどういった花にするのか、鉢をどのぐらいの大きなものにしておかなければいけないのかということは、変えられれば良いのですが、変えられない可能性もあるわけです。であれば、そういうものをあらかじめデザインしなければいけないし、毎日水をやるのであれば水をやるというルールは決めておかなければいけません。ですから、そういうプロセスに関してきちんと計画を立てることは必要なのですが、何をどのように造るかを計画する必要はないということです。ですから、計画は必要です。そのためにシェフという人がきちんとプランしなければならないということです。

○舩橋　最後に造るものは考えるのですか？でき上がった花はイメージしなければいけない？

○濵　方向性のイメージですね。

○舩橋　そのときに、いつ花が咲く予定かという時間的な概念はどう見ていますか？

○濵　　フィードバックで戻ってきてしまうので､次の状況が変わってしまうのです。ですから最初に設定したゴール設定も、何かを市場に出した瞬間に売れないとなったらやめるという話にもなります。そうすると、最初に継続的に 2 年続けようと思っていたものが、1 カ月ぐらいで駄目になりましたと。でもこれでやったら意外にこういうところにお客さんが食いついたからというと、そこを基にまたフィードバックしてどんどん状況が変わってきてしまいます。

○舩橋　　一見、無計画のように、

○濵　　はい、そう言われると無計画です。(一同笑)

○鈴木　　方向性はあったけれども、途中でその方向を自由自在に変えていくのだという。

○濵　　はい。先輩方に言うのもあれですけれども、人生設計と同じで、どうなるか分からないものを･･･。

○木下　　陽子崩壊の実験をニュートリノの観測に切り換えたのも、途中で変える例ですね。小柴昌俊教授のお仕事のことです。陽子崩壊を観測するための実験をしていたのだけれど、観測できないでいたところ、同じ実験施設でニュートリノの観測ができたので、目的を切り替えたということがありました。

○濵　　あえて言うならばそうですね。ですから、アジャイルで良く最初に語られたことは、変える勇気を持ちなさいということで、決められたことが変わったことに関して､「ああ、変わったのだ、間違えたのだ」ということを認める勇気が大事だと言われていました。

○山本　　でも、顧客も入ってくるのですよね。

○濵　　顧客が入っています。今までのアジャイル開発プロセスは顧客を入れているようであまり入れてないのです。ですので、顧客とか、例えばベンダと発注側だと、さらに発注側がやっているビジネスのお客さんもいるので、そういうところも巻き込んで、どんどん変えていかなければいけないということです。

○栗田　　会社ではさぼっていると思われて､｢計画はダイナミックに立てていますから」と言っても「栗田さん、最初からもっと全力尽くして考えなさいよ」と言われて、さぼりの技術と思われてしまうのが、つらいところですね。

○濵　　だから「技術者の幸せ」とか言うのは、止めて欲しいですよね。

○栗田　　そうなのです。(一同笑)

○久保　　鉄道システムとか社会インフラとか、時間をかけてモノを造っていかなければいけないところは、アジャイルの考え方で成立するのですか？

○濵　　僕は成立しているのではないかと思います。それを 1 年や 2 年のプロジェクトという目で見るとアジャイルになっていないかもしれないですけれども、100 年という単位で見ると（アジャイルになっている)、「今の Suica は最初に計画されていたものですか？」と言いたいですね。

○久保　　時間軸をもう少し広く取って見れば、それもアジャイルだと。

○濵　　広く見ればそれが当たり前なのだから、小さくやるときもそれを当たり前で

やっても良いのではないですか、ということがベースにあります。

○栗田　ソニーの例で見ると、テレビ開発は waterfall ですが、中の部品は全部アジャイルで、その中で試行錯誤しているのです。

○久保　先ほど waterfall もアジャイルのスペシャルケースだという話もありましたので、そういう組み合わせも。

○栗田　Suica は waterfall ですけれども、私の開発はアジャイルに近い試行錯誤があるということだと思います。

○大島　栗田さんの今のお考えを、「新しいことに挑戦するときはアジャイルで、完成されたところは waterfall でやるのが良い」と単純に受け取ってしまうと危険ですか？

○栗田　危ないです。例えば PASMO だと 800 社の人たちが集まって仕事するそうですが、そういうところでアジャイルは難しいので全体的な計画が必要です。いつからいつまでテストするのだけれども、その時までにこれがないとテストは対象外になりますというような全体計画は、大きな公共事業では必要な気がします。

○久保　こういうことをいろいろなステークホルダを巻き込んでやろうとするときに、契約をしてお金のやりとりが発生しますが、コロッと変わった時はどうなるのですか。契約はまた破棄するのですか。

○濵　組織自体も、ヒエラルキーを作って皆でやりましょうというものではなく、それぞれ細かく分散して独立した組織がうまく、セル生産ではないですが、何らかの目的に向かって関係を持って連動していくという仕組みにしないと、いろいろなステークホルダが居過ぎて多分一枚岩ではいかないと思うのです。ですから一枚岩でやることを止めて、どちらかというと、独立分散した組織をどのように組み上げてゴールを達成するかを考えるべきなのかなと。その答えは何ですかと言われると、私は分かりません。

○久保　最後にもう一点。さきほど仰られたアジャイルと仏教思想はどうつながるのですか？

○濵　諸行無常です。常に変わり得るものを受け入れるということです。

○久保　それは仏教思想だけではなく、東洋思想全体がそうですよね。

○濵　そうですね。なので、何かスタティックな完成されたものを目指すことは止めようということです。

○久保　大前提はそうですね。そうすると、アジャイルの元の意味は俊敏ですけれども、そこから発生して、変化に対応するという話ですか。

○濵　そうですね。ですから、速くも安くもないです。

○久保　ちょうど良いという。

○濵　はい、いい塩梅で。

○鈴木　どうもありがとうございました。（拍手）

2.3 モデルに基づくシステムズエンジニアリング ―MBSE と SysML―
西村秀和 （慶應義塾大学大学院システムデザイン・マネジメント研究科教授）

INCOSE（International Council on Systems Engineering）のシステムズエンジニアリングに対する定義とは、「さまざまなドメインが絡む複雑なシステム開発を成功裏に実現するためのアプローチおよび手段である」ということです。初期の段階で顧客ニーズを明確化し、機能要求を定義し、関連する問題をすべて考慮しながら、設計のための総合とシステムの妥当性確認を進めるという意味では、ソフトウェア開発で言うアジャイルにはなり得ない分野を扱うのがシステムズエンジニアリングだと思っています。

1990 年代後半から 2000 年にかけてシステムズエンジニアリングの標準化が活発に進み、ISO15288 や IEEE1220 などが標準の中心的な存在になっています。

システムズエンジニアリングプロセスを構成する上で重要なアクティビティは、「システム設計」、「インテグレーション（統合）」、「評価・解析」、「システムズエンジニアリングマネジメント（全体のマネジメント）」の 4 つであるとされています。

ドメインを跨ぎ協働作業で行うシステム開発においては共通言語となるものが必要となり、良いコミュニケーションを取るためには、仕様書よりも図的に表現したモデルの方が効果的です。また、抽象度の高いモデルによってイノベーションを導くことも可能となります。こうしたモデルベース・システムズエンジニアリングの重要性は、Boeing や NASA などの航空宇宙関係では当たり前のように語られ、価値を生むシステムズエンジニアリングを行おうということを強調しています。

モデルの記述には SysML を利用します。これは UML から派生したもので、「要求」、「振る舞い」、「構造」、「パラメトリック制約」の観点で図的に表現できることが大きな特徴です。ただ、複雑過ぎて膨大な資料が読み切れなくなっている点が課題となっています。

SysML で表現したシステムモデルを利用して、ソフトウェアチームとハードウェアチームのコミュニケーションを取ることが大きな目的となります。システムレベルでしっかりと要求を検討し、システムアーキテクチャを決定し、仕様書として機械、電気などのハードウェアチームやソフトウェアチームへ投げます。ここでハードウェアチームとソフトウェアチームの開発がスタートできるわけです。そして、その後の設計変更や統合についても、このモデルを中心に検討していきます。

最近ではマニュファクチャリングに関わる PDM（Product Data Management）とも関係性を持たせようという流れが進んでいますので、あっという間にまた標準ができるのではないかといわれています。日本もそれに乗り遅れないようにしなければなりません。

＜講演中の質疑応答＞

○木村　　この場合のシステムは何を想定していますか。手順ですか、それともでき上がったものですか？

○西村　　これから開発しようとしているシステムで、これを System-of-Interest と呼んでいます。

○木村　　これはソフトウェアではないのですね。
○西村　　違います。例えば自動車を開発したいという人が、･･･。
○木村　　開発するシステムですか？
○木下　　プラントとか、自動車とか、飛行機とか、開発する対象です。
○木村　　開発のプロセスですね。
○西村　　プロセスも入りますけれども、成果物としては対象そのものです。自動車を開発するのであれば、自動車がシステムになります。
○木下　　ISO15288 はその中のプロセスを規定しています。作業手順は規定していないです。
○木村　　どのような手順かは別として、手順ですね。
○木下　　アウトカムです。
○木村　　ライフサイクルというのは、自動車ができ上がってから廃棄されるまでというものではないのですね。
○木下　　廃棄のプロセスもここです。
○山本　　製造プロセスだけではなく、企画も全部入っています。
○木村　　そこまで入れると何が何だか分からなくなります。何も言わないに等しいのでは？
○山本　　それを包含したものがシステムです。
○木下　　いわゆる技術的と言われるプロセス以外にも、マネジメントなども入っています。
○木村　　こういうことを JST の中で言うと必ずそういう反論が出るのですが、それが通るということがすごいなと。
○山本　　それが世界標準なのであって、日本の中で言われている、JST の中で考えられているシステムはものすごく狭くなっているのです。
○木村　　いえ、広すぎるのです。システムというのはわけの分からないまったく無限定なものになってしまっている感があります。そこでこういうきちんとした言語として乗せようとするときに、必ずギャップが生ずるのです。「いったいシステムとは何ですか？」と。
○山本　　「当店のシステムはこうなっています」という、要求システムのことを言われるみたいな感じで。
○木村　　システムを限定しないで、こういう記述ができるということが、･･･。
○西村　　システムはしっかり限定しています。このスタンダードの中でシステムとは何かを定義しています。
○木村　　それが出ていないので。
○西村　　システムとは何かということを言う必要性をこの場では不必要かと思って。
○木村　　あまりにもいろいろな人がいろいろな定義を持っているので、そのことを言っていただかないと。途中で申し訳ありません。

＜講演後の質疑応答＞

○鈴木　ありがとうございました。質疑の時間を少し取りたいと思いますけれども、何かご質問はございますか？

○木村　飛ばされた 25 ページ以降のシーケンス図、ブロック定義図、アクティビティ図、状態機械図とありますが、状態遷移図ではなく、状態機械図ですか？

○西村　英語が State Machine Diagram なので私はそのまま訳したのですが、日本では状態遷移図とか State Machine Diagram としてそのまま使われています。

○木村　こういうものはすべてモデルと考えておられるわけですね。単にシミュレーションができる、できないではなく、こういうものが言語になると。

○西村　システムモデルの一部です。

○木村　分かりました。

○久保　これは物理システムと、そのほか金融系、バンキング系のシステムなどに対しても適用できますか。

○西村　IT システムそのものは、使えるとは思うのですが。ここではシステムというのはもう少し上流のシステムで･･･。

○久保　これはダイナミクスを持ったものだけですか？

○西村　だけではなく、別に全部を使い切る必要はないと思いますので。

○山本　データモデルとかできないですから、バンキングではない。

○白坂　元々が複数のモデリングのツールとか、ドメインごとに持っている今までのアプローチがあるので､それらをつなぐための統合のための言語として SysML は生まれています。ソフトウェア系はソフトウェアで UML があります。熱は熱で特有のアプローチがあります。構造は構造であります。制御は制御であります。でもそれではつながらないので SysML でつなげましょうというのが、もともとの概念です。

○久保　もともとの発端はそこにあったわけですね。

○白坂　そこがスタートです。

○久保　コミュニケーションツールとしてやりましょうという。

○白坂　それらをつないで、システムとしてモデル化することを目指しましょうというのが、バージョン 0.2 の時の話です。

○久保　OMG（Object Management Group）の BMM というのがありましたよね。

○木下　BMM は Business Motivation Model で、ビジネスはシステムより上位ですけど、これはシステムから下なので、OMG が分けてやっていることには意味があるのです。

○西村　最新の情報に近いのですが､InterCAX という会社が､SysML の外側に CAD のモデルとか、最適化ソフトとか、シミュレーションモデルとか、それぞれモデルをつなげようとしています。さらにプロジェクトマネジメントの要求ツールとか、そういったものも全部統合して、プロダクトライフサイクルマネジメント PLM （Product Lifecycle Management）とかサプライチェーンマネジメ

ントもしっかりやろうということを、INCOSE 全体で言っています。SysML というのはシステムモデルが中心にあって、このやりとりがいろいろ行われる、その間に彼らは SLIM（Software Lifecycle Management）というソフトウェアを提供しようとしているわけです。こういうことを目指そうということで、Lockheed Martin 社でも、Boeing でも、Ford でも進めようとしているというのが今の状況です。Siemens は PLM のアプリケーションとして TeamCenter を提供していて、Siemens もこれに近いことを最近は言っています。

○木村　SysML というのは誰がやっているのですか？

○西村　管理は OMG（オブジェクトマネジメントグループ）が中心にやっています。

○木村　INCOSE の？

○西村　INCOSE ではなく、OMG というグループが単独にやっていて、本拠地はボストンです。

○木下　企業がお金を出し合ってやっているような、フォーラム標準という標準化のための団体です。

○久保　キーパーソンはどこかの大学の先生がやっているのですか？

○西村　もともとは MIT だと聞いています。今、代表は Soley さんという方です。

○松本　先ほどモデルは実行可能でなくても良いという話をされましたが、実行可能でないと、結局システムとして動かないものができてしまうことにはならないのですか？

○西村　それだけで造り上げると言っているのではなく、実行可能なモデルではないものもモデルと言って、このような記述をしていく。そして最終的に、例えばシステム解析をしたいと思ったら、それは実行可能なモデルを使わなければシミュレーションも何もできませんので、それは当然そのモデルも含みます。けれども、システムモデルといったときに別にそれがシミュレーションでガツガツ動かなくても、それはモデルです。

○山本　間接的には実行可能だとおっしゃっている。直接には実行できないが。

○西村　実行可能なモデルがなくてシステム開発ができるわけがないです。でもそれを実行可能なモデルでないとモデルとは言わないと言い切る人がいるのです。これは戦いが起こるぐらいです。

○山本　30 年ぐらい前に「データフロー図が実行可能じゃないんじゃないの？」と言ったら、「指で実行できるだろ」と。（一同笑）

○西村　脳内のシミュレーション機能が実行可能と言っていただけるのでしたら、シーケンス図で線を引いて、ここで何か起きて、こう戻る、これはシミュレーションを脳内でやっているのです。でもソフトウェアで何かガチャガチャとやって計算して結果を出すというのを実行可能なモデルと普通は言うので、そればかりがモデルではないですよ、ということを申し上げているのです。

○鈴木　ではちょうど良い時間になりましたので、西村先生への質問はこれで終わりにして、あと総合討論の中で続けたいと思います。どうもありがとうございま

した。

（拍手）

2.4 社会ニーズ把握のための技術－要求工学の動向－

山本修一郎（名古屋大学情報戦略室教授）

システムの実現には、社会があり、要求があり、アーキテクチャが必要だということが認識され始めています。最近の情報システム構築では、システム要求と社会ニーズをつなぐことがどうしても必要となります。システム要求が「問題化」され、社会ニーズとして合意が必要となり、「合意化」されたシステム要求が解決され、「解決化」したシステムが使われることにより発展し、「発展化」したシステム要求がまた「問題化」されていくというプロセスを継続していくということです。

要求工学では、要求を「抽出」し、「分析」し、「仕様化」し、「確認」することが重要なプロセスとなります。要求は「機能要求」と「非機能要求」に大きく分けられ、そこには「成果物に対する要求」「プロセスに対する要求」「外部環境に対する要求」が含まれます。ISO/IEEE29148 では、ステークホルダの要求を作る観点から、「ステークホルダ要求工学」「システム要求工学」「ソフトウェア要求工学」の 3 つがあるとされています。

要求分析手法としては、あらゆる視点における分析の必要性から、実体関連図、IDEF、i*フレームワークなど、さまざまなモデルが提案されています。

エンタープライズアーキテクチャのメソッドとしては、Software Architecture、Twin Peaks Model、ACE、AUTOSAR、MEFSA などが提唱されています。最近の TOGAF（The Open Group Architecture Framework）のメソッドでは、「ビジネスアーキテクチャ」「情報システムアーキテクチャ」「テクノロジーアーキテクチャ」ということで、アーキテクチャが階層化されており、ターゲットアーキテクチャを重要視したあらゆるリソースの再利用体系が既に規格化されています。これは 10 種類のアーキテクチャ開発工程から構成されており、一番重要なプロセスは「要求管理」であるとして、全開発工程において要求管理するということが TOGAF の考え方となっています。記述言語としては ArchiMate が標準化されており、TOGAF の全開発工程をサポートしています。

システム論としては、System of Systems を分析するためのシステミグラム（Systemigram）、生物をモデルにした VSM（Value Stream Map）などが提案されています。そして STAMP（Systems Theoretic Accident Modeling and Processes）という階層的制御構造が、「制御ソフト」「アクチュエータとセンサー」「制御対象」の上に「制御担当者」を置いて制御ソフトをさらに監視・コントロールするということで、新しいシステム方法論であるといわれています。

しかし「従来の安全性分析」「ディペンダビリティ分析」「システム論的安全性分析」という階層の横にいろいろなものを並べていくという整理もこれから必要になるのではないかと思っています。

＜質疑応答＞

○鈴木　　どうもありがとうございました。最初にお断りすればよかったのですが、山本先生からいただいた資料は 140 ページ近い資料なものですから、さすがにそ

れを全部印刷してお配りするのはJSTとして費用もかかりますので、山本先生に40ページに減らしてくださいとお願いして、皆さんに配布したものは40ページ分相当の資料です。山本先生にもし許していただけるのであれば、140ページ近いファイルを皆さんに後でお送りすることは可能かと思います。山本先生に直接お願いする形が良いですか、それとも私のところに。

○山本　面倒くさいので、配布してください。

○鈴木　では、ご希望を聞かずに今日いらっしゃる方にお送りしてよろしいですか。

○山本　はい。

○鈴木　先ほど配布された資料にプレゼンで使われている図がいくつか抜けているというのは、こういう事情でございます。それではさっそく質疑に入りたいと思いますけれども、いかがでしょうか。

○田名部　モデルの表現形態として、今までのVDMのようなやり方もあれば、SysMLの図というのもありますが、今回のビジネスレイヤーの話ですと、むしろ文章的なものの方が、理解が共有されやすいというような、図よりも文章ということを少し言われていたようなのですが、そういう理解でよろしいですか？ビジネスのレイヤーですと、図だとコンテクストの情報をどのように共有するかというところが問題になっていて、システム開発が実際に進んでいったときにコンテクストが抜けてしまうのです。ですからコンテクストをなるべく正しい理解のままで伝えて行って、運用の時点までコンテクストの情報がずっと伝わることが良いと思うのですが。

○山本　スコープで何がコンテクストなのかということを、例えばアシュアランスケースの手法ではきちんと明記させる。どういう原則を対象にしているのですかということは整理しなければいけないのですけれども、こういったことは、文章に書いてあっても良いのですが、コンテクストが何かというのが明記されていれば良いわけです。それから、データフロー図とかだと、コンテクストダイアグラムということで、そもそもこのシステムのコンテクストはこういうものがあるということが用語でリストアップされています。ですからコンテクストのリストがあれば良いのではないかと思います。それを図で書けないわけではないです。

○田名部　コンテクストを持続的に、開発なり運用のフェイズでも顕在化させるための良いやり方は何かないですか？

○山本　コンテクストという記述項目を用意するとか、こういうふうにして明記させるとか。つまり識別されなければ管理されない。管理できなければ変更できない。そういうことです。普通、コンテクストを書かないことが多いので、要求工学ではコンテクストを書いてくださいというのが主張です。コンテクストだけではなく、コンテクストバウンダリーも書いてくださいと。これはコンテクストを変えるなと言っているのではないです。

○田名部　認知したことをまず書いておいてくれと。

○山本　　そうです、識別されたということを。識別しておかないと変更できないので。

○鈴木　　他にいかがでしょうか。

○藤井　　先ほど、システム科学の分類で組織化されている・されていない、単純・複雑性というところをもう一度見せていただけますか。そのリファレンスは、・・・。

○山本　　Rebovich さんの本も見てください。

○藤井　　分かりました、ありがとうございます。

○田名部　　これは古くから、George J. Klir とか、あの時代からいわれているシステム科学の基本的な考え方ですよね。

○山本　　そうだと思います。でも、右上の領域がこれから研究しなければいけない領域として識別されています、ということだけです。

○大島　　今、説明していただいた中で、人というのはシステムの中にどのように位置づければ良いのだろうかとずっと考えていました。「システムの一要素として入っている」と言われたらそれまでなのですが、「システム」を計画するときに、最初は人の要素を抜きにして考え、運用局面において人を考慮するというやり方もあります。

安全性やセキュリティなど、人に係わる要件は多々あります。人というものを、システムを構成する他の要素とは一段次元の異なるものとして捉え、それとの関係性でシステムを把握するという考えがあると、もう一つ理解しやすいと思いながらお聞きしました。

○山本　　説明してないですけれども、ここで言っていたのは、システムがディペンダブルでセキュアだというときに、まずシステムの前提条件を付けます。そしたら、その前提条件に基づいてシステムが動作している間は安全ですということを言っています。これは 3 つに分けていて。前提条件から逸脱することがあります。そのときに例外条件として例外処理を定義していれば、例外処理の枠内でシステムの安定動作ができるということでもあります。でも例外処理の範囲も超えることがあります。そのときは人間が対応しますということを決めておく。人は運用プロセスで関与します。

例えばどういう例があるかというと、エスカレーションルールで、ある範囲に問題が止まっていれば現場で対応できます。それは手順書があります。でも現場で対応できないことが発生したらどうするかというと、エスカレーションしてもらいます。エスカレーションしてくると、上位の対応委員会か何かが動いて、それが用意されているのです。エスカレーションルールがないケースがほとんどなので、現場でぐずぐずしていて、結局、大変なことになるというケースが起きてきます。そういった形で、例えばシステムの運用ということについては、こういうモデルを作ることで確認することができます。

○大島　　それは、人間というか、運用みたいなものが補完的な存在としてあるというモデルですよね。

○山本　　そうですね。

○大島　　システムの外縁にまず「使う人」がいて、というとらえ方を言っています。私が言いたいのは、人間は補完的な存在ではないと言う意味でもあります。

○山本　　それはステークホルダ要求仕様の話にいきます。

○大島　　なるほど「システムの中の人」と言うとらえ方ですね。

○山本　　運用者から見たらこうなる。経営者もやはりこういう例外が起きたときに、「どうなっている？」のということで入ってきます。ですからステークホルダ要求はシステム要求のさらに外側にあります。そこでは人の活動とか組織も含めて定義しています。

○大島　　運用者としての人もそうですか？

○山本　　はい、なので大変です。

○山本　　本当にやろうと思ったら大変ですけど、外部的に整理すれば簡単です。こんな面倒くさいことは、実際にはやっていられないです。

○栗田　　やっている人はいるのですか？

○山本　　そこで重要なのが優先順位です。なぜそうなっているのかという証拠が要るのです。「経営を優先します。安全はちょっと置いときました。なので、脱線転覆事故が起きました」「どうなっているの？」ということです。ですので、こういうものをちゃんと残しましょうと。

○濵　　優先順位の付け方はないのですか？

○山本　　あります。それは皆さん、やっておられるでしょう。

○濵　　でも全部は難しいです。

○山本　　一般化して言えば簡単です。評価項目をリストアップします。それから評価項目のウエイティングを決めます。そして総合点を出して、上から順番に選んでいくぐらいの手法しかないです。

○栗田　　それで裏切られることはないですか。

○山本　　裏切られます。（一同笑）

○栗田　　だから人はどう学んでいきますか、という話だと思うのですが。

○山本　　そのときの論点は何かというと、その優先順位の決め方が間違えている可能性があれば変更しなければいけないですけれども、定義しなければ変更できないのです。だから定義してマネジメントしろということになるのですが、立派な経営者は直感的に決めています。だから「こんな手法を使うような経営者は駄目だ」と言う人はいるかもしれません。（一同笑）

○栗田　　いつもの話で、結局、立派な人がやれば良いという話になります。

○山本　　成功した人が立派な人だったということです。

○鈴木　　一方で、経営者がすべて見通せるわけではないという･･･。

○山本　　そうではない普通の人は、こういう手法を使ってきちんとマネジメントしましょうということです。

○栗田　　だからやり切れないという話になって、また優先順位を付けましょうという、ルールが堂々巡りになります。

○山本　　でも説明責任がどういう場面で出てくるかといえば法廷です。裁判の時の「論理的に考えて行動した結果ですか？証拠はどこにありますか？」ということなので、「きちんと定義してみんなで合意した結果、こうなっています」ということを言えないと、「お前が悪い」となるわけです。

○栗田　　それはそうです。でも、それが達成したいことではないので。裁判に負けないことではなく、社会に貢献するとか、そのようなことなので。

○山本　　どういう場合であっても、我々は適切に行動しているということをモニタリングしておかないと改善できないので、改善のための手法だと思っていただけると良いのではないでしょうか。

○鈴木　　どうもありがとうございました。（拍手）

2.5 妥当性確認とアシュランスケース

木下佳樹 （神奈川大学理学部情報科学科教授）

システム（現実）と記述（仕様）とはどういうものか。「群盲、象を撫でる」ということばがありますが、この場合は、象がシステム（現実）、群盲一人一人の感じたものが記述（仕様）です。システムと記述には必ず「ずれ」が生じます。記述がシステムに照らして正しくなかったり、記述がシステムのすべてを記しているわけではなかったりすることを指します。

このような「ずれ」を調べるのが「検証」や「妥当性確認」と呼ばれるプロセスです。「検証」とは、正し「く」システム構築しているかどうかということです。「妥当性確認」とは、正し「い」システムを構築しているかどうかということです。

これらのプロセスのアウトプットとして最近注目を集めているのが、アシュランスケースとよばれる文書です。システムが要求を満たすことを、一連の主張、証拠、それらを結ぶ明示的議論によって、理由付き・監査可能な形で示す文書一式がアシュランスケースと呼ばれているものです。

アシュランスケースは一般に複雑・膨大な文書ですが、プログラムと似ており、ワンフレーズの誤りが全体を無意味にしてしまうものになりかねないため、正確な記述とチェックが必要です。アシュランスケースを図式で表記し、人が理解しやすくするようにする試みが普及しつつあります。さらに、アシュランスケースを形式化、つまり機械で処理できる形で記述して、機械が検査できることは機械に検査させようという試みもあります。アシュランスケースの形式化のためには、用語とその基本的な性質、つまりオントロジーを機械でも処理できる形で定義し、それに基づく議論を記して、チェックさせます。コンピュータによる定理証明技術をこのチェックのために用いることができます。

システム（現実）と記述（仕様）を区別することが重要で、その二つを比べたり確かめたりする妥当性確認や検証というプロセスがあり、これらのプロセスのアウトプットがアシュランスケースである、というお話をしました。アシュランスケースの図式表現に関しては、現在、産業展開がかなり進み、普及してきています。形式アシュランスケースについては研究段階といって良いでしょう。

＜質疑応答＞

○鈴木　　どうもありがとうございました。時間どおりに終えていただきまして、ありがとうございます。5 分ほど時間の余裕がありますので、質疑をしたいと思います。

○木村　　学問的なベースはどこにありますか？

○木下　　数理論理学、プログラム理論です。

○木村　　数学基礎論のあたりですか？

○木下　　はい。いわゆる formal logic ですが、最近では informal logic という言葉で formal logic と非形式的な議論の間をつなぐ議論を指すことも始まっており、アシュランスケースの根拠のひとつはこれです。Stephen E. Toulmin の考えを

基に、トゥールミン・ダイアグラム（Toulmin Diagram）というものが作られていますが、アシュランスケースの図式表現は、それに近いものです。いずれにしろ、基本的にこの分野の研究活動が拠って立つ学術的基盤は論理学といって良いでしょう。

○木村　人工知能をやったような人たちがやっているのですか？

○木下　必ずしもそうではありません。人工知能は情報科学のあらゆる分野の母体ということができるように思いますが、プログラム、あるいはもっと一般的に情報処理過程の数学的モデルに関する研究が 1970 年代半ばあたりから、人工知能から分化してきており、現在では programming semantics 等と呼ばれています。さらに software engineering（SE）と呼ばれる分野も、同じ頃から形成されてきました。今回お話した話題は、この二つの分野に跨るものです。コンピュータによる定理証明は 60 年代から人工知能の分野で研究されてきましたが、80 年代にインタラクティブな定理証明技術がはじまりました。対話的な証明システムはプルーフ・アシスタント（proof assistant）と呼ばれています。定理の証明を形式的に構造エディターで書くためのシステムです。

○木村　数学的な定理の代わりに、このシステムは、「これができますよ」という言明を証明できると。

○木下　そうです。数学の理論を相手にするのではなくシステムごとに、例えば自動車とか JST の組織とか、そういうものに一つの理論ができるわけです。そういうものです。

○山本　あと、アーギュメントセオリー（Argumentation Theory）という議論理論があります。

○木下　Toulmin の本で『The Uses of Argument』というのがあります。

○山本　そこでの議論は、クレームを書くのですが、「そもそもそのクレームに対する確信度はどれぐらいあるのですか」という話ばかりです。100%などないと。証拠があると言っても、その証拠もどれぐらいのものなかという、どんどんわけの分からない話になっていくのですが、そういうことについての理論的な学問もあります。

○木下　先ほど栗田さんが「我々の仕様はこういうものです」と言いましたが、あのような感じでアシュランスケースを書くわけです。Proof assistant を使って、栗田さんの仕様のようなものをチェックすることも考えられます。

○藤井　数理論理学に基づくということですが、これは形式手法の一つということですか？

○木下　そういって良いと思います。ちなみに、形式手法とよばれるものは、多角的なアプローチをまとめたものです。形式手法というと一つの手法のように聞こえますけど、そうではないのです。そういう意味では、これも形式手法の一つで良いのですけれども、栗田さんの話でも、モデル検査もあれば VDM もあるという話が出ました。他にもいくらでも挙げられるわけですけれども、これら

に共通するのは、全部、数理論理学に基盤を置くところです。飛行機を造ったり、速く運転したりするのは、全部、微分方程式、微分積分でやりますが、「物理学が解析学を使ったように、情報科学は数理論理学を使っていくのではないか」と、昔、竹内外史教授も仰っていました。

○木村　結局は自然言語の限界を、不完全さをカバーするというところにあるわけですか？

○木下　そうだと思います。自然言語で書くことの一部分は機械でもチェックできるという立場です。

○木村　栗田さんは自然言語で書いてないですよね。

○木下　今回、見せてもらったのは形式仕様ですが、形式仕様の上に必ず自然言語による説明があるはずです。

○山本　事前条件と事後条件を VDM で書くわけです。でも事前条件自体が間違えている可能性もあるわけです。でも一応、事前条件と事後条件に基づいて、この仕様は妥当であるということはチェックしています。

○木村　事前条件の妥当性は問わないということですね？

○山本　一応、そこは議論して、「これで良いね」というレビューは要るわけです。

○木村　要るけれども、別問題になるわけですね。

○山本　そうです。

○木下　もう一つ、システムの定義について。先週、新谷さんも出席していた ISO/IEC の SC7（Software and Systems Engineering）の席上で、システム「ズ」エンジニアリングだろうと、システムエンジニアリングではないのだという議論がありました。複数の“s”はどういう意味なのか、と訊いてみました。「世の中にはすでにシステムエンジニアリングというものがあって、そこではどちらかというとエレクトロニクスや機械工学におけるテクニカルなシステムが扱われる。我々が対象とするものは、マネジメントやヒューマンファクターも含めたものなので、ここは絶対に“s”が要るのだ」ということでした。

○西村　ですから日本語でもぜひ「ズ」を付けて欲しいのです。

○木下　そうですね。ですから ISO 15288 の策定でも、皆、そうして既存の概念と戦っていますので、我々もやらなければいけないのではないかと思っています。

○木村　日本語ではその次に工学という漢字が来てしまうから、「ズ」と言いにくい。

○西村　工学を僕は反対していまして、システム工学という表現をこれには当てはめない方が良いと思います。システムズエンジニアリングという日本語を片仮名で並べないと誤解されると思います。

○山本　確かに。

○木下　そうすると、名前がますます長くなりますね。

○鈴木　Verification と validation の違いについての説明で、私の解釈と何か違っている感じがしたのですけれども。私は、verification というのは、仕様と実際にできたものの間にきちんと consistent があるかどうかを確認するのが

verification だと思っていたのですが、先ほどそのように聞こえなかったのですけれども。

○木下　そういうつもりです。

○鈴木　Consistent の方は、validation の方に入っていましたよね。

○木下　それは説明が悪かったです。ここですね。

○鈴木　「正しくシステムを」、それは理解できるのですけれども、そのときに仕様とシステムの間がきちんと 1：1 という感覚が verification（検証）の方なのかなと。

○木下　そうです、これが検証です。

○鈴木　そうすると、右側の図と、…。

○木下　システムが仕様を model しているという意味です。

○鈴木　なぜ consistent というキーワードが上に入るのですか。

○木下　Consistent というのは、これだけを見て、辻褄が合っているかどうかなのです。

○鈴木　それが妥当性ですか。

○木下　はい。

○山本　妥当性は、要望？　要求？

○鈴木　要求との関係ですよね。

○木下　ここで言いたいことは、仕様だけを見てこれが良いものかどうかというのが妥当性だと。そして関係を見ているのが検証だと。それを言いたかったのです。

○木村　それは要求に対して妥当かどうかという意味ですか。

○山本　そうです。普通はそれが validation です。

○鈴木　それで良いのですね。

○田名部　ずれていますね。

○山本　だから validation は検証できないです。つまりお客さんが見て、望んだものかどうかを確認してもらうのが validation です。

○木下　でも、ここに要求があるのなら、それは検証と変わらないところがあるというのが、我々のあれです。

○山本　だから、ちょっとずれていますね。

○木下　ずれていました。

○田名部　そうすると、システムが仕様のモデルになっているのですか。

○木下　そういう理解です。

○田名部　ということは、仕様の世界だけはシンタックスの世界で考えていて、こっちの世界を考えようと思ったらセマンティクスと考えなければいけないということですね。いろいろなシステムがあり得て。

○木下　セマンティクスと言った途端に、セマンティクスがまた基準となるので、話がややこしくなりますけど。

○田名部　このシステムはどちらが最初にあるのですか？仕様というのは形式言語で何

でも書けますけれども、仕様のモデルになっているシステムは一杯あり得ますよね。

○木下　　あります。そこがポイントだと思います。

○田名部　　システムのコンテクストとかを考え始めると、そこのシステム･･･。

○木下　　コンテクストと言った途端に、それは基準だと思うのです。システムというのはぐちゃぐちゃと動いているものであって。

○田名部　　後でもう一回議論させてください。

○鈴木　　木下先生、どうもありがとうございました。（拍手）

3．総合討論

○鈴木　　引き続き総合討論に入りたいと思います。あらためてお見せする必要もないですが、最初にご説明したワークショップの趣旨のところに書きましたような視点で総合討論をしたいと思います。まず講師の先生方、あるいはシステム科学検討会の有識者とメンバーの方々で、今日の5件の発表に関して、ここだけはもう少しコメントしたいというところはありますか？

○田名部　　あまりにも一杯あったので･･･。

○鈴木　　では、最終的に私としてはシステム構築方法論というのは何だろうというところに関心を持っていますので、その趣旨に沿うご質問から。現時点での到達点というのは、これから俯瞰報告書をまとめていく中で出てくるかと思いますけれども、システム構築方法論というものを創っていこうとしたときに、ソフトウェア開発のいろいろな技法、今日ご紹介いただいたものの大部分がそういうものですけれども、結構そのままシステム開発に使えそうなものは多いのだとか、逆にシステム開発の世界で必ずしも形式手法や何かは今までなされてこなかったとか。西村先生がまさにそこを創ろうとされているということが、今日のお話だったような気がするのですけれども。

さて、1番目、まずソフトウェア開発とシステム開発は同じなのですか、あるいはどこが同じでどこが違うのですか、という議論をしたいと思いますが、どうでしょう、今日のお話を聞いて。

○西村　　そもそも今までの歴史的な経緯をしっかり見なければいけないような気がするのです。いわゆるソフトウェアの開発というのが先行していたのは事実で、システム開発という言葉は、恐らくソフトウェア開発の次のステップで出てきた言葉ではないかと思います。システムとソフトウェアをちゃんと分けて言っているのですけど、ハードウェアも含めたソフトウェアが含まれているようなシステムの開発というのは、ソフトウェア開発が先にあって、もちろんシステム開発をしている人はいたのですが、ソフトウェア開発というプロセスがしっかりできてきたところで、「やはりシステムとしても創らなければね」と言ってきたはずです。そういう歴史的な経緯からすると、当然、同質なものもあるとは思うのですが、それをソフトウェアの方が使っているものを、そのままシステム開発に使えば良いのですよね、という話にはなり得ないところです。

○鈴木　　どこがそうならないか。単にソフトウェアだけではないというだけでは、･･･。

○西村　　ソフトウェアシステムというものと、ソフトウェアのみならずのシステムの振る舞い方とか、構造とか、そういったものが違うからです。ソフトウェアだけで動くものと、ハードウェアも含んで動くシステムとは、明らかに違いがあるので、その違いがあるのに同じ方法論でできるわけがありません。ただし共通点はあるので、それを見出そうということであれば、もちろんそれはそれで

良いのですが。でもそれがほとんどスタンダード（国際標準）に書いてある可能性があって、･･･。

○木下　どうでしょう、みんな手探りだと思います。

○大島　今、「ソフトが先にあって」と言われましたけど、そこからして「えっ、それはどういう意味でしょうか？」という感覚があります。前提をある程度明らかにしてから、ソフトとかシステムを議論する必要があると感じました。「システム」や「ソフトウェア」という言葉を聞いて描くイメージが、人によってさまざまだと思うからです。

○西村　まずシステムと言ったときには、ソフトウェアのみならずのシステムを言っているというのがあります。

○大島　それはそのとおりです。ただ、そこで「ソフトウェアが先に」となったと言った瞬間に「えっ、そうなの？」と感じた訳です

○木下　対立的に採られるのが抵抗ありますね。

○西村　ソフトウェア開発の方法論の方が先にしっかり形づくられたということを申し上げたかったのです。

○山本　もしそうでなければ、なぜ今システム構築方法論の議論があるのかということです。情報システム構築方法論という本はたくさんあります。

○木村　システムは何もソフトウェアよりもずっと昔にありました。

○山本　言っているのは、system development method とか system construction methodology がなぜなかったのか。既にあったのだったら、今、改めて問う必要はないです。

○大島　そこが大事なところですね。なぜ今になって「システム開発方法論」が表舞台に出てきたかというと、一つはシステムが大規模複雑化してきたからではないかと思うのです。昔はシステムの規模も小さくそれほど複雑ではなかったように思えます。

○山本　そうではなく、システム開発方法論も昔からあります。では、もしそうなら、大規模複雑システム開発方法論というタイトルにした方が良いのではないですか。

○木村　現実にはそうです。

○松本　でも、変わってきているのは事実だと思います。システムとソフトウェアという区別がだんだんなくなってきた、その境目がなくなってきた今は、ソフトウェアなしのシステムはあり得ないですし。

○山本　それが一番重要なポイントで、システムの機能が昔はソフトウェアで作られてなかったけれども、今、ソフトウェア抜きのシステムがほとんどなくなってきたというのが重要なのです。例えば自動車の 90%以上の機能はソフトウェア化されています。

○木下　それも怪しくて、例えば昔の帝国陸軍では、あの巨大システムを扱うマニュアルをもっていました。そういう意味では、コンピュータソフトウェアではな

いかもしれないけれども、ソフトウェアに相当するものは昔からあると思います。ソフトウェアに相当するものなしにシステムを動かすようなものはないと思います。

○山本　開発方法論はないです。

○木村　方法論として意識したものはなかったけれども、方法論はあったのです。例えばORなどはまさにその最たるもので、これはソフトウェアよりもずっと前です。第二次戦争のときに始まったけれども、あれは非常にシステマチックなシステム解析なのです。予測も入っていますし。

○山本　もしそうだとすると、ここで問われているものはORのネクストバージョンですかということで。

○木村　もちろん違います。

○山本　そうですよね。ですから、その次のステップが要るわけです、今、あらためて再構築か、構築しなければならないものの実態が本当に過去にあったのか、なかったのか、という問いです。そのベースで、例えばソフトウェア開発方法論や情報システム開発方法論は、それなりにたくさん精緻なものが用意されてはいます。ということです。膨大なものがありますから。

○大島　そのときに、ソフトやシステムというものが何なのかというイメージを合せておかないと、システムの中にソフトがあるみたいな形で、ソフトというのはシステムの要素であると見るのか、利用技術という意味でシステムを括るものであるのかということを、どちらでも良いのですけれども、規定しないと。

○山本　Checklandのソフトシステム方法論というのは、もっとすごい概念で、全体を括るソフトな方法論という、そういうのも1980年代からあるものです。

○木下　システム構築方法論という題ですけれども、ISO15288的に考えると、構築というのは先ほどのリストの右端だけのわけです。あの左のところはマネジメントとか。

○鈴木　「そういうものをこれから作っていかなければいけないね」というのが私の立場なのですけれども。

○木下　ライフサイクル全体を考えなければいけないねという感じですね。

○山本　業界標準でもITIL（Information Technology Infrastructure Library）などは全ライフサイクルです。開発だけではなく、運用も含んでいますし、さらにその後の廃棄も、先頭のビジネスの戦略のところまで含めたものが既にあるので、そういったものを参照されたら良いのではないですか。逆に言えば、そのものとの違いを明確にして、

○木下　だいぶ標準化までされているので、研究する必要はないということなので。

○木村　標準化にはかなり驚いた。

○山本　だから標準もあるし、標準化されていて、まだそうなっていないものも当然あるので、その残ったところとか、今後の課題を明らかにした方が良いと思います。そうしないと重複してしまいます。

○木村　　つまり日本では、一般人が「システムズエンジニア、SE」というとソフトウェアなのです。それに我々は非常に苦しめられているわけです。つまりシステム科学技術というものはITの一部であるという発想は非常に根深いですよね。

○大島　　そうなのですか？

○木村　　実際そうです。ファンディングということになると完全に一部に入っているのです。

○木下　　システムズエンジニアリングは単なるシステムエンジニアリングではないという了解を共有した上で、しかしそのシステムズエンジニアリングは、実はやはり IT なのだと、僕は思っています。だから帝国陸軍のマニュアルのようなものがあるとしたら、それはITですし。

○（数人）　それは･･･。

○木村　　それだとなかなかうまくいかないです。

○木下　　私のいう IT というのは、コンピュータのプログラミングという意味ではないのです。

○西村　　情報系という意味で、情報的な流れとしては、それはそうですけど。

○木下　　そうなのです。インフォメーション・テクノロジーですから。

○木村　　システムズエンジニアリングがその中に入ってしまうと、我々のユニットは存在意義がないのです。

○木下　　それは政治的な話ですね。

○木村　　政治的ではなく、私は全然違うと思ったのです。両者は多分資質が全然違います。システム構築に必要な能力あるいは才能と言いますか、そういうものに向いた資質というのは、ITの人たちがこうだと言っているような資質とは全然違います。

○木下　　そこでITとおっしゃるのは、プログラマー？

○木村　　プログラマーではないです。広い意味で情報。

○西村　　木下先生、システムの定義をちゃんとしませんか？　混乱していると思います。

○鈴木　　一応我々としては、このように定義しているのですが（ワークショップ趣旨説明時のスライド 4 ページ目を提示)。これが良いかどうかは別の議論ですけれども。

○久保　　鈴木先生、前、これを以前のシステム検討会で俯瞰図として作りましたけれども、それを木下先生たちにも見てもらった方が良いのではないかという感じがするのですが。

○木下　　僕はあのシステムの定義に立った上で、システムズエンジニアリングというのはITだろうと思っています。だから単に言葉の問題ではありません。

○木村　　では、最適化とかモデリングとか予測とかネットワークとか、こういうのはITの中に入っていますか？

○鈴木　　今度は IT の定義がはっきりしないと。多分、木下先生は情報を扱うものは

すべて IT だという感覚で、その中に入るだろうと仰っています。

○木下　そういうところです。

○西村　相当大きな捉え方をして言っているのだったら同意するのですが、IT と言った瞬間に、IT 系のいわゆる今、情報システムでやっている人たちがドンと入ってきて、その方たちをシステムズエンジニアリングの人ですねと言われたら、それは違うでしょうと。

○木下　プログラミングがどうのこうのと言われたら、そうだと思います。

○木村　（システムは）IT と物理学のどちらに近いですかと言われたら、それは IT に近いです。だけど、INCOSE のように独自のプリンシプルを持って独自のコミュニティがある訳です。

○西村　INCOSE がはっきり IT と違うと言っているので、それは間違いないことです。

○木下　INCOSE の人と、標準化の会議などで話をしますけど、共通の基盤に立って話はできています。

○山本　例えば OR とか、メソッドレベルで使えるものと、全体をストラクチャリングするようなマネジメント系とか、プロセスを構築していくようなテクノロジーは違うので、そこはレイヤーがあって、そういうのをメソッドエンジニアリングと言っていたりする訳です。

○木下　上の方ですか。

○山本　全体の構成です。メソッドアーキテクチャとか、そういう議論もどんどん進んでいるので。

○木村　私は進んでいるとは思うし、悪戦苦闘していると、今日を見ていて良く分かるのです。対象は難しいものであって。でも、こう言うと非常に失礼ですけれども、残念ながら深さがないのです。ファンディングの対象までいかないのです。

○山本　それはそうかもしれないです。

○木村　それだと困るわけです。そこをファンディングできるぐらい深くしなければいけないのです。それが IT の人ではできないと思います。

○木下　それはちょっと･･･。

○木村　IT 的な方法では違うのです。

○山本　ファンディングの方針とか、ファンディングできる条件が明確になっていれば、それに合わせて depth は作れると思っています。ただ、ファンディングの方針が分からないので、そっちですよね。

○大島　ソフトにファンディングというのは世界では当り前です。その点では日本が特異なので、日本が変わらないと駄目なのでしょうね。

○木村　我々の競争相手はナノテクとかライフとか情報もそうなのです。

○大島　産業界はやはり困っていると思います。このままではソフトウェアの開発現場は疲弊していく一方なので、そこにファンディングという光を当てて欲しい

のです。

○木村　ただ、IT の人たちは非常にアグレッシブです。何でもかんでも抱え込もうとします。

○西村　最近は IT と言わず Internet of Things（IoT）というので掴もうとしているのです。それは絶対逃れられないのです、どう考えても。マニュファクチャリングの人たちは、それをどのようにうまく結び付けるかということで、この間の OMG の会議の中では、GE と Intel と IBM といくつかの会社が一緒になってコンソーシアムを始めているのです。そこに日本の名立たる大手メーカの人がいるのですが、様子を見に来ただけで、本気で掴もうとしないのです。

○木村　（IoT など最近の流れに対して）僕は正直なところ、かつてのユビキタスとどう違うのだという気がしてしょうがないのです。日本の方が先発で、向こうは追いかけてきているだけではないかという。僕は深く知らないので、間違えているかもしれません。

○西村　そのように見る日本人も多いのですが、残念ながら乗り遅れるのです。世の中がそこでも進んで行くのに、いつまで経っても日本は追いついて行かないというのが、今までの実態です。

○木村　だからユビキタスがなぜ日本でポシャったかを反省しなければいけない。

○山本　それは世界と一緒にやらないからです。

○西村　そうです、標準化をやろうとしない。

○山本　EU のプロジェクトとして IoT とかをやる訳です。例えばクラウドの新しいアーキテクチャについてファンディングして IoT と一緒にやる訳です。そういうことにお金がどうも付いている。EU だと国の数とか、そこにアメリカも入ってくるので､圧倒的に全体のプラットフォームを支配されてしまうわけです。日本でいくらユビキタスと言っても、後から来たものを遠吠えで何とかと言っているだけだと、本当に負けてしまう訳です。

○木村　それは問題ですね。

○山本　むしろ一緒にやった方が良いです。

○松本　あと、山本先生が言われたように、超巨大システム、ウルトララージシステムというのは、やはり米国でもかなり熱心に研究を始めています。それだけ大規模になると、本当にソフトでどこまでコントロールできるのかというところが大きなテーマになりつつあるので、そういう観点もひょっとしたら出てくるかもしれないです。ファンディングの観点とすれば。

○山本　もう一つ難しいと思うのは、隠蔽主義です。これが安全だとか何とかと言いたいときに、日本はできているということを前提にするから、そもそもそういう技術を開発する人が･･･、というようになってしまって、なかなか難しい面があります。そこの文化的な要素は確かにあると思います。

○木村　ただ、僕は深さという点では、先ほどのアシュランスのお話は新しいテーマで、深くなる可能性が非常に強いという気はしました。

○鈴木　木村先生が深さと言う時には、やはり何らかの学問的基盤が欲しい、あるいは学問的基盤の上に成り立っているというところが欲しいということなのですね。

○木村　基盤というよりも、論理構造が深くて面白いなとか、エレガンスがあるなとか、言葉だけが連なっているのではなく。

○木下　バズワードをできるだけ作らずに、･･･。

○木村　そう、バズワードがなくて。私、制御から来たので、多分、久保さんも。制御は非常に見事な理論体系ができている訳です。それが制御は多分特殊だと思います。

○鈴木　検証とか妥当性の確認技術、あるいは仕様書を書く形式手法的なところは、皆、根底は数学の定理証明の技術ですね。逆に山本先生がいろいろ紹介してくださった手法は山のようにある訳です。それの基盤になるものは何ですか？

○山本　それはアーキテクチャではないですか？

○鈴木　ではアーキテクチャの理論は何ですか？フレームワーク的なことはたくさん提案されるのですが。

○山本　基盤はやはり数学になると思います。

○鈴木　そこまでどうすれば落とせますか？

○山本　基本的には構成要素を特定して、構成要素間の関係を明確に定義した後、そのアーキテクチャによって実現されるはずのプロパティをそのアーキテクチャが満たしているということの充足性の確認技術を作れば良いのですけれども、そこは証明できる範囲で､実は顧客価値を生むところはそこではないわけです。そもそも論として、そうして造り上げられたシステムが本当に使われるのか？使われるという環境の中で安全なのかというところの議論構築の方が、あるいはその反復的なプロセスをモニタリングして変更して改善していくというところの方が実は重要で、そこが現実的な価値を生むわけです。だからそこに価値を見出してファンディングできないと、非常に狭い範囲だけで限定されてしまう可能性があると思います。

そういうところで言えば、形式手法はずっとやっている訳です。でも、形式手法がなぜ現場に本当に展開されていかないのか、栗田さんみたいなところだけが頑張っているのかといえば、技術としては精緻化されていたとしても、それだけでは産業化されないからなのです。そこをきちんと認識できるかどうかというのは、結構大きいと思います。

○鈴木　イノベーションが重要だと言いながら、実はイノベーションを起すようなところには、お金はなかなか行かないということですね。

○山本　訳の分からないイノベーションにはファンディングできないので。

○木村　経産省のファンディングはいいのです。ここ JST は文科省なのです、残念ながら。そこに大きな違いがあるのです。

○白坂　基本的にアナリティカルなものは確かにバックグラウンドを付けて再現性を

出して証明できるかもしれないですけど、シンセティカルなデザイン問題とかは解空間が広過ぎて再現性も出ないですし。一回デザインしてしまえば、それをベースに評価軸を決めればいくらでもアナリティカルできるし、一回決めれば行けますけど。

○木下　シンセシスにも理論の貢献というのはあり得る関係です。ただ、数学にだとカテゴリーセオリーというのがあって、この理論には乗るのですが、どんな公理を立てるべきかという指針を与えるという側面があります。理論がある訳です。オントロジーです。ロジックというのは、ディダクションとオントロジーの 2 つからなっていると哲学の人は言うのですが。

○白坂　分かります。そのときに、私は 0 か 1 かと言っている訳でないのですけれども、そこに限定するのかと言っているのです。

○木下　別に限定しなくて良いと思いますけど。

○白坂　山本先生が先ほどおっしゃったように、そこにしかファンドが創れないとなると、それは厳しくないかなという話です。

○山本　そうであれば（文科省がそういう考えなら JST は）、日本の中だけで小ぢんまりとやっていれば良いのではないかと。世の中は世界に出てどんどん発展していくので、拡大している領域の中で、「本当に何をすべきか」という議論をしないと、もっと個別化されてしまいます。

○木下　それを経産省がやっているような産業育成の立場から世界でやるのか、文科省がやっているような学術的な研究をどうするかということでやるのかで、だいぶやることが変わると思うのです。

○山本　産業の成長エンジンとしての文科省がやるべき科学を提案することは、できなくはないと思います。

○木村　そこを是非やって欲しいのです。

○山本　そういう視点を入れれば良いのではないですか。

○木村　それが今、我々が悪戦苦闘しているところで、学問の世界でこの分野の市民権を得たい訳です。

○山本　それはアジャイルに提案して潰され、提案しては潰されて行って、徐々にやっていくしかないのではないですか。

○木村　ある程度認識され始めているのです。先ほど松本先生がおっしゃったように、これだけ企業が悩んでいる訳です。

○山本　企業の悩みはもっと陳腐なものだと思いますけど。

○木村　そうでもないです。

○山本　悩むぐらいであれば、先ほど言われましたけど、グローバルスタンダードの世界でどんどんやっていけば良いのです。

○木村　そこで負けている訳です。

○山本　だから出て行っていないのです。仮に行ったとしても見物しているだけですから。

○松本　　今は出て行けないのです、本当に。エンジニアが不足して、2015 年にはソフトのエンジニアがいなくなると。いまだに手で作っているからエンジニアが不足するわけです。今の現場には工学的な要素がないのです。

○山本　　作ってない会社の方が多いと思います。作っているのはベトナムの人とかに作らせているからじゃないですか？

○木下　　イギリスはインドの人に作らせる訳ですよね。

○松本　　日本でもオフショアでやりますけど、それでもエンジニアが足りなくなります。上流の要件定義といったところは内部のエンジニアがやらなければできないです。

○山本　　そこを教育しなければいけないのに、日本の中だけでやっているからコストが掛かり過ぎるわけです。

○木下　　日本が不足しているなら、イギリスはもっとエンジニアが不足しているはずなのに、彼らは別にそんなこと言ってないです。どこが違うのでしょうか？

○山本　　人は一杯居るような気がするのですけど、かけ方が違っているのではないですか？

○松本　　いやいや。

○鈴木　　もう一つ今の話の中で、山本先生の話では国際標準の話が出て来ました。日本からは個人ベースで、あるいは経産省をバックには出ているけれども、アカデミアとしてまとまって出ているとか、どこかのアカデミアがきちんと議論しているという場がないのではないですか？

○木下　　情報関係は全部、情報処理学会の情報規格調査会の中で･･･。

○鈴木　　ですから狭い意味での IT というところではやられているのだけれども、システムという視点でやられていません。やはり専門家のネットワークが日本の中にきちんとできていないことが、日本の弱い原因でもあるのです。独立した分野として認知されていないという原因でもあるのです。

○白坂　　INCOSE でも言われましたし、SC7 の WG でも言われましたけれども、やはりソフトウェア屋さんだけではなく、システム屋さんを出してくれとか、翻訳のときには必ずソフトウェア屋さんではない人を入れてくれとか、それは必ず言われます。私はどちらかというとシステム屋として ISO の方には入っていたので、「珍しい」と言われたのですけど。日本は情報処理学会なので、基本的にハードの学会とかの人は居ないのです。それは常に言われます。

○木下　　そうですね。先ほどの ISO15288 のシステムライフサイクルと、ISO12207 のソフトウェアライフサイクルでいうと、ソフトウェアライフサイクルの方が先にあったのです。それでシステムが後で出てきて、それが今回の改定では、システムライフサイクルの方が上位にあって、ソフトウェアというのはシステムの一例なのだから、それに合わせるようにしましょうと。だからシステム屋さんが乗っ取ったのです。国際的にはそういう状況です。

○鈴木　　今、新谷さんが探してくださいましたけれども。

○新谷　　これは先週の ISO の会議で出た一番新しい ISO 15288 の定義です。
○木下　　DIS（Draft International Standard）ですね。
○新谷　　はい。そこでシステムの定義、それからシステムエレメント、システムインタレスト、システムズエンジニアリングが定義されています。
○木下　　さっきの鈴木先生の定義と、だいたい一緒ですね。
○新谷　　これが今日現在、もっとも新しいものです。
○山本　　Combination of interactive elements はまさにアーキテクチャのことで、achieve some purpose となっているのが特性の訳です。あそこから原理的にいろいろな学問的なアイデアが生まれる可能性はあると思います。
○西村　　ここのシステムは、実態としてのシステムでしょうね。アーキテクチャは？
○山本　　もうちょっとそれを一般化すれば、人間の組織も入れて、･･･。
○新谷　　ここはサービスも入っていますから。サービスでもあるし、モノでもあります。それから先ほど極めて重要な発言があったわけですけれども、実は日本では ISO15288 というプロセスはほとんど知られていないです。ところが世界は逆に ISO12207 が知られていないのです。ISO15288 というシステムの方が知られているのです。
○木下　　ISO12207 を一番良く使っているのは日本です。
○新谷　　だから今すごい問題が起こっているわけです。結局、日本では、ハードウェア系の人は、モノを造れるということに関しては、自分たちに自信があったのだと思うのです。
○木下　　ISO15288 はだいたいアメリカ軍と NATO がやったのです。そこに日本の自衛隊が居なかったのです。
○白坂　　私は電機メーカ出身ですけど、当時メーカ系の会社がどこをリファーしているか調べたのです。そしたらほとんど MIL-STD-499A の古いままです。だから 69 年ぐらいの規格を入れたまま、その後、アップデートしてないのです。なので、相当古いままで来ています。勤務先の電機メーカもそうでした。私が入って改定しようとしたのがちょうど 2004 年ぐらいでした。
○木下　　古い規格を使っていて大丈夫ですか？
○白坂　　なのでソフトが合わなくなってきて、中のプロセスが合わなくなってきて、それで入って、これを合せましょうという話になって、やはり ISO15288 にしなければという話をして、変えたのです。
○西村　　ほとんどの企業が古いままでやっているので、本当に困っているのです。
○木下　　ガラパゴス化しているわけですね。
○西村　　そうです。本当に外注で何かやろうとしたときに話が合わなかったり、そういうことでトラブルが起きてリコールが生じたりと、いろいろな問題を起しているわけです。
○新谷　　今のは標準の世界の話ですけれども､もう一つは､これは情報規格調査会で、情報処理学会の下にある訳ですけれども、どなたかが仰ったように、ハードウ

ェア系の人がほとんどいないのです。これは WG7 というワーキンググループでやっているのですけれども、ソフトの人ばかりです。

○木下　ソフトのインダストリーの人が圧倒的に多く、インダストリー以外は僕をいれて３名だけです。

○新谷　ハードウェアのインダストリーの人が入ってないのです。

○木下　皆、あれで一グループなのです。

○鈴木　私はやはり人とか組織、マネジメントのところまで本当は入らなければいけないと思うのですが。

○木下　マネジメントを入れなければいけないという意識は、日本の WG7 にも大いにあります。そいうのは、ソフトウェアでもそういうことを感じているからなのですけれども、組み込みシステムとかプラントとか、そういうことを知っている人がいないのです。

○新谷　なので、それは今後変わっていくという一つの要素かもしれません。

○木下　ぜひここからも WG7 にも入っていただいて。

○山本　でも、ここの話ではないような気がしますけど。

○鈴木　ここの話ではなくて、どこの話ですか？

○山本　それは学問ですかと言われるのです。

○木村　あまりそれを振りかざすつもりはないのですけど、ファンディング、いつもナノテクの人たちとやり合っていますから。

○山本　ナノテクの方がお金を出しやすいですよね。（システムのように）見えないものにはお金を出せないという感じですよね。

○木下　ナノテクの対象も見えないですよ。（一同笑）

○木村　見えないものにお金を出すようにならないと駄目だと思います。そこが一番大事です。

○山本　だとしたら、やはりビジョンではないでしょうか。それで世の中がこう変わるというような、あるいは世の中がこれからこのようにシステムで変わっていくという。

○木村　だからそれを出して欲しいわけです。

○西村　そう言えば OMG の中のテクニカルミーティングで、ヘルスケアについてこれからモデルベースでやっていこうという話があったのです。その中のオバマ大統領直轄の委員会のコメントに「システムズエンジニアリング」という言葉が入っていたという話が最初に出てきました。オバマ大統領がどこまで理解しているか分かりませんけれども、それぐらいシステムズエンジニアリングが大事だということを認識していらっしゃる訳です。ところが日本はそれがまったくないので。

○木村　まったくというか、最近は多少、例えば総合科学技術・イノベーション会議から科学技術イノベーション総合戦略というのが毎年出るのですが、そこには 3 つの戦略視点という形で、一つがグローバル化、もう一つがスマート化、も

う一つがシステム化ですが、3 本のうちの 1 つにシステム化が入っている。そのぐらいになっているのです。

○西村　そのシステム化という言葉も、多少誤解されやすい面もあって、やはりシステムズエンジニアリングという、最後までやり切るというようなイメージがないのです。ですから、モノになって世の中に出て行かないで、何十億円も注ぎ込んだのに「はい、終わり」となるわけです。そういうことをしっかり提言するのもここの役割で、文部科学省かもしれませんけれども。これは実は教育にも関係する話で、要するにそういうシステムを造り上げる能力を大学にも学生たちにも持たせないといけない訳です。それがまったくないですよね。何とかシステム工学科というのがあっても､そういう教育はまったくやってないです。

○木下　大学がシステムとして機能していないと。(一同笑)

○西村　もちろんそれもテーマにすべきかもしれませんが、まったくそのとおりですね。それを言ったら文部科学省もそうかもしれませんけれども、まったく機能してないということをもっとしっかり言うべきで、ファンドを取るとか取らないとかは、その次で良いような気がするのです。

○木村　ファンドを取るか取らないかがその分かれ目になっていて、役所と先生と研究者の結び付くところはファンディングなのです。

○西村　内閣府の SIP で 10 個ぐらいテーマを挙げて、内燃機関の方が大きなのを取っていますよね。あれは、実は白坂さんはすごく動いていて、システムズエンジニアリングの観点でしっかり構築していかなければいけないということを彼らに言って、いろいろ調べてみると、ドイツはまさに大学に資金を投入して、企業も一緒になって開発をするということを 10 年 20 年やってきたことが出てきました。それに対して日本も何かやらなければいけないということで、あの予算が付いているのです。そういういろいろな分野にこれから必要性があるということが、こういったところでしっかり言っていかないと、いろいろなことが良くなって行かないです｡あれはたまたま内燃機関が良くなるかもしれない、ならないかもしれないですが。

○木村　内燃機関は、(研究開発の) ポイントはディーゼルです。

○西村　ディーゼルですけど、あまりにも遅れてしまったということを反省して、やっているわけです。

○木村　そう、大学でもああいう研究はやらなくなってしまったから。

○西村　そうです。ですからマニュファクチャリングもそうです。大学の中に本当に生産ということで一生懸命やっている人が居るのかとか、あるいは設計ということで語っている人がどれだけ日本の大学に居るかというと、ほとんど居ないと思います。

○山本　それは大学の先生の問題なのではないですか。

○西村　そうです｡組織として大学の先生がそういう人たちをどんどん排除してきた。

○山本　排除ではないのでしょうけど、どうなのですかね。

○西村　　そうですよ。ロボティクスはやるけれども、設計はもう良いですと。

○木村　　東大の IRT（Information and Robot Technology）という拠点があったのですが、かなり大々的にやっていまして、企業もたくさん入っていました。ところが変な問題が起こったのです。機械工学科のある建物のあるフロア 2 つぐらいを借り切ってやろうとしたのですが、そこに ID カードでなければ入れないようにしたのです。企業が入ってきますから、いろいろな人が出入りしてもらっては困る訳です。そしたら猛反対が起こって、「大学というのは開かれた場所である。一部の人だけが入るような場所があってはいけない」ということで、結局押し切ったのですけれども、大変だったらしいです。日本の大学はそういうところがあります。ヨーロッパの場合でもあることはあって、企業が出入りするのは別の建物になっていました。

○西村　　比較的そういう議論をオープンにやろうとしているのが、標準といわれるものだと思うのです。ベースラインはオープンにしておいて、そこから先は皆で競い合いましょうという。

○木村　　だからプレ・コンペティティブの部分に限っているわけです。アメリカでも多分限っています。コンペティティブになると絶対駄目です。

○西村　　もちろんそうです。ですから、その辺の仕切り方も含めて、日本というのはうまくできてないです。文句ばかり言って申し訳ないのですけれども、そういういろいろなことがあるのが現状なのです。だからこれをどうしたら良いかということをどこかで考えなければ、本当にまずいです。

○鈴木　　そこの部分は西村先生にも入っていただいて、もう一つの別な会議でこれから議論していこうということになっています。

○久保　　西村先生が今言われた話は、システム構築方法論の中で非競争領域と競争領域というグレードをある程度分類して、非競争領域のところに何か学問的な基礎付けをきちんとした方が良いという意味で言われたのですよね。

○西村　　大きくはそういう意味です。ですから大学も一緒になってそこに入り込んで欲しいと思うのですが。

○久保　　最終的には産業の競争力につながらなければ、工学という意味がないですものね。

○西村　　意味ないです。

○久保　　学問のための学問ではないわけですから。

○西村　　今はそれに行きがちな感じがしてしょうがないのです。

○木村　　行きがちといえば、今、大学でやっているのは、実は応用研究のみだと私は思っています。基礎研究と言いながら、本当の意味での基礎研究は多分かなり枯れ果てているのではないかなと。だけど応用に結び付かない応用研究をやっているのです。（一同笑）

○西村　　そうです。それもファンドの問題だと思っていて、本当に基礎研究というところにお金を注ぎ込もうとしてないということも、一方で問題かと思っていま

す。

○木村　そうですね。ですからそちらは一方で本当にやらなければいけない。応用の方も、実用まで貫徹するような形式を創らなければいけないです。それは非常に仰っているとおりです。

○新谷　SEC は海外のいろいろな研究機関とも交流していますが、世界で珍しく産官学が集まっている場所でもあります。産官学連携の手法として、SEC の中で今まで WG を積極的に活用してきています。産官学の連携を通した議論の中で、これは更に深堀り、あるいは、基礎的研究をしなければならないという項目を特定することも可能と考えます。ただ、SEC は SEC で持っている事業計画の達成という問題もあるので、ひょっとしたらしんどいかも知れません。ファンディングの原資が、SEC は METI で、JST は文科省ですから。

○木村　だからこそ協働してやる意味が非常にあると思うのです。

○久保　だからもともとは文科省と経産省の横からの独法で、木村先生とやりましょうと。

○新谷　そうすると一番可能性があるのは、SEC という場所で産官学の人が集まっているわけですから、そこの中で「これはファンドを付けてもっとやろう」というものをいくつかピックアップすることが非常に具体的ですよね。

○鈴木　SEC の S がソフトウェアからシステムズに変わると。

○久保　昔、S&S と言っていたのです。System and Software と。少なくとも前職の会社ではそう言っていました。電力系統の System and Software。

○木村　SEC さんにファンディングの項目があるのです。

○松本　ほとんどないです。文科省さんに比べたら微々たるものです。

○木村　タイトルがソフトウェアだけだったのに、最近は Software and Systems も付けてくださいましたね。非常にありがたいと思っております。

○松本　一応システムズエンジニアリングも対象にして提案をいただくということに。そういう意味で白坂先生が今回応募していただいて。

○久保　西村先生の応募もあります。ですから、それはここでやるファンディングのつなぎとしてやっているような位置付けなので、もう何年かするとお金がなくなりますから、できるだけ早くこれを片付けなければ。

○松本　そういう意味でシステムズエンジニアリングといっても、結局組み込みソフトは対象になるわけですよね。

○西村　もちろんそれも入ります。

○松本　そうすると、それは基本的にソフトウェアですよね。

○西村　ただ、システム全体を見渡して欲しいと。先ほどのお話の中でも、ソフトウェアがシステムの一部であっても、コンテクストが大事だということが山本先生のお話であったと思いますが、やはりそこからしっかり明確にして、そして組み込みソフトウェアとして何をするべきなのかということを定義しなければいけないのです。

○松本　　そうなると、別に組み込みであろうとエンタープライズであろうと関係なくて、エンタープライズでもコンテクストが重要。

○山本　　そう思います。エンタープライズ系もデバイスを持っていますから。

○木下　　もう組み込みソフトウェアという風に特別扱いするのはおかしいですね。エンタープライズでも、例えば株式市場制御のソフトウェアも組み込みソフトウェアと同じで･･･。

○松本　　そうですね、リアルタイムでやらなければいけないので。

○山本　　ArchiMate でも、デバイスとして何をしますかということをきちんと書いているので、あれを見れば、組み込みも ArchiMate で書けるということだと思います。でもやはりハードがある、そのハードにソフトウェアをこのように割り当てるという割り当て関係の妥当性とか、それもきちんと検討しろということです。機能としての入力と出力の間の関係性だけ検証して終わりではなく、きちんとデバイスに割り当てたときに、所望の性能を達成できるのですかということも議論しなければいけないです。ですから全体論的なことをちゃんと考えましょうという話なのですけど、それはなかなか難しいと思われがちです。

○松本　　個人的には SysML などでエンタープライズ系は書けるのではないかと思うのですが。

○山本　　ビジネスを書けないですから、無理です。

○西村　　そのために BPMN（Business Process Modeling Notation）というのを別に作られて。

○松本　　ビジネスモデルはそうですけど。

○山本　　ビジネスプロセスも書かなければいけないので、そこは少し違います。

○木下　　そういうライブラリをビジネス用に作ればできるという話ですね。

○久保　　BPMN というのは、OMG が強く活動しているのですよね。

○山本　　割と BPMN は､日本の中ではユーザキーを使っているところもあるのです。

○松本　　BPMN はすごくシンプルで分かりやすいですね。

○鈴木　　そういう日本の中で提案していたアイデアが、海外から入ってくるという実情が問題なのです。

○山本　　逆に言えば｢あるものは再発明するな｣ということだと思います。ですから、その上に乗せる新たなアイデアを日本から出せば良いだけだと思います。

○大島　　今の議論から、ソフトウェアに対して、日本が本質的に変わらなければいけないところが浮き彫りになってきているように思えます。結局何がネックかというと､｢学｣もそれなり考えているし､｢産｣の方も一生懸命やっているのに､｢産学連携は悪だ｣みたいな前世紀の遺物がまだチラチラしているような感じでしょうか。旧世代の経営者はともかく、若い人はだいぶ変わってきているので、そういった若い人が発言力をもち、行動していけば世の中は大きく変わると期待しています。

○山本　　でも日本はまだ余裕があると思います。

○大島　そうですね。確かに余裕があると思います。各々の企業が実力を持っていて、それなりにできてしまうので、企業間をまたいだ日本連合軍で勝とうなどということをしなくてもよかったのです。でもこれからの時代は違います。企業連合、産学連携とか日本トータルの総合力を発揮できれば、もっと世界をリードできると私は思っています。

こういうことを我々はもっと啓蒙して行くべきでしょうね。大学での勉強に対して企業があまり期待していないという現実も変えていく必要があります。併せて企業の人材採用のあり方も見直さなければなりません。

○久保　それは企業で社内教育の制度がしっかりしていて、こっちで鍛え直すという発想があったからです。

○大島　ありますね、どちらが先かは分かりませんけれども。

○木下　昔の話ですね。

○久保　僕らが入社したころの話で、今はそうではないかもしれません。

○木下　今は即戦力を求められているので。

○山本　海外の連中は UML とかを大学で習っていてすぐ使えるけれども、日本の人は大学を出ても知らない人ばかり。あるいは企業ごとに固有のメソッドがあって、それを使いましょうといっても、海外の人はそれを知らないですから、海外に持っていくためにさらにコストを掛けるわけです。そのようなことは早く止めた方が良いです。

○大島　日本はまだ余裕があるのですね。

○山本　アズイズ（As-Is）でできるものは可能な限りスタンダードを使って、さらにその上の新たな価値領域で日本の企業や大学が勝負できるような枠組みを創っていただきたいということです。

○木村　まったくそのとおりです。

○大島　日本の企業がそのようになっていくためにはどうしたら良いかです。企業の経営者が自分の企業の枠の中で人を育てるということは必要なのだけれども、企業ごとにやらなければいけないものと、そうでないものの切り分けをする必要があります。「いい人を採って育てよう、そして競争に勝とう」ではもはや世界では戦えないでしょう。

○山本　例えばこのシステムズエンジニアリングのシラバスか何かを作って提示すれば良いのではないですか？そのシラバス作りに企業の人も入れる。そして皆で共通にこれをやっていくという。形として見せることが必要だと思います。

○大島　企業も参加して、自分たちが作ったという風にしていけると良いですね。

○新谷　でもシステムよりもソフトウェアの方がはるかに具体的ですよね。ただ、SWEBOK（Software Engineering Body of Knowledge）というのが SEBoK（Systems Engineering Body of Knowledge）よりもはるかに具体的なのですが、SWEBOK の Ver. 3 というのも最近出ましたけれども、日本でどのぐらい知られているか、日本の大学ではどこで教えているか。多分誰も教えていない

です。SWEBOK でもそうなのです。

○山本　それのエントリー版か何かを作れば良いのではないですか？

○新谷　そう。ましてや SEBoK というのは項目も多くて大変なものですから。

○白坂　SEBoK では大変なので、GRCSE （Graduate Reference Curriculum in Systems Engineering）で一緒にやっていて、一応 Graduate School のカリキュラムのベースのフレームワークを作って､そこの CorBoK というのがあって、コアとしてこれだけは教えなければいけないということが一応識別されているので、少なくともそれを教えるぐらいはやっておかないと本当はまずいです。

○新谷　そうですね。研究も何も、今ある技術でさえも、認知されていないという事実の方が怖いです。

○山本　そういうものを簡単なブックレットとして作って配れば良いのでは？私が作ったようなものを。「これです」と言えるものを形だけ見せれば。

○白坂　あと凄く難しいのは、大学で言うと論文に出していくことを考えるときに、学会として情報処理とかソフトウェア関係があります。じゃあシステムズエンジニアリングを日本でどうするというときに、システムズエンジニアリング学会というのはなくて、基本的には機械学会の中のシステムだとか、宇宙工学の中のシステムだとか、ばらばらなのです。ここは多分、大学の先生が入り込むというモチベーションとしては結構辛いところで、論文どうすれば良いのという。

○山本　学会を創れば良いのではないですか。

○木村　横幹（横断型基幹科学技術研究団体連合）ね。

○木下　インターナショナルなレベルだと･･･。

○白坂　インターナショナルだと、INCOSE があったり IEEE があったり、いくつかありますけれども、やはり数が少ないです。IEEE にはシステムがありますので。

○木村　システムというか、あれはまだソサイアティではないですね。ソサイアティになれば良いのだけれども。

○白坂　はい。インターナショナルで見てもあまりないので、そうすると大学の先生に「どこに論文出すのですか」と必ず最初に聞かれるのですけど。我々が出すときは、ドメインに合わせてドメイン側に出すとか、切りとったところだけをどこかに出すとか、そういうアプローチなので。

○木村　西村先生にお願いしたいことは、例えば電気工学科の先生たちの連合体がありますが、システムでそれを創ったらどうですか？小さくても制御工学はあるのです。文科省からお金が出て、年に 1 回研究集会をやるのです。ですからシステム、先生のところとか、和歌山大学とか、秋田県立大学とか、学科の連合体。それを創ると、少なくとも文科省は認知の方向に一歩進みます。研究集会をやれば文科省の人間が来ますから。

○西村　研究集会という名前でやるのですか。

○木村　　そうです。
○久保　　それは SICE（計測自動制御学会）とは別にということです。
○木村　　SICE はやってないです。
○西村　　SICE は制御の研究集会がありますけど、あれのシステム版。
○木村　　制御は古いですけど、毎年やります。メンバーは 300 人ぐらいで、毎回 100 人ぐらい、全国を回り持ちで。そういうこともぜひ先生のところがリーダーシップを執って。
○西村　　検討させていただきます。

4．まとめ

本日の総合討論ではさまざまな視点から深い議論が行われ、システム開発を巡る環境変化、システム構築方法論の世界の潮流、システム構築方法論を含む日本のシステム科学技術が抱える課題と対策が明らかになった。それを項目ごとに纏めると以下のようになる。

(1) システム開発を巡る環境変化

・システムはますます大規模化、複雑化している。
・システムは、ハードウェア、ソフトウェアの複合体である。
・システムの機能の大部分がハードウェアに代わりソフトウェアで実現されるようになったが、システムの実現にはハードウェアが必須なものも多い。
・システム開発は、単に技術の問題だけでなく、価値創造と捉える必要がある。そのため、システム開発の上流部である要件定義やシステムライフサイクル全体を通してマネジメントの要素がより重要になって来ている。必然的に人間・組織・社会の問題と切り離せなくなって来ている。このためシステム開発は、これまで以上に異分野の知の統合が求められており、既存ディスプリンの価値観からの評価は危険である。
・この点で、従来のシステムエンジニアリング（システム工学）ではなく、システムズエンジニアリングと呼称するようになって来ている。

(2) システム構築方法論を巡る世界の潮流

・ISO15288 など、システムズエンジニアリングの国際標準の策定が進められている。
・INCOSE など、ハードウェア系、ソフトウェア系を含めてシステムに関連する異分野の研究者・実務家が産学官の枠を超えてネットワーク化が進められている。
・SEBoK などシステムズエンジニアリングの標準知識の体系が進められている。また大学教育で用いる SEBoK の簡易版である CorBok の作成も進められている。

(3) 日本企業の課題

・日本企業は技術の自前主義が強く、各社独自の規格を作って差別化を図ってきた。
・このため技術のガラパゴス化が進み、現在のシステムズエンジニアリングの国際標準と乖離してしまっている。また、国際標準策定の枠組みからはじき出されている。
・日本企業の中には、依然として旧式の国際規格をそのまま利用している企業もある。
・このため、海外のグローバル市場で販路を確保するために、わざわざコストを掛けて国際標準規格に合わせる必要があり、産業競争力の低下を招いている。

(4) 日本社会の課題、学術界の課題

・システムとソフトウェアを同一視し、システムをソフトウェアの一部としてしか認識していない傾向が強い。

・システム科学技術が、ハードウェア系、ソフトウェア系、さらにはマネジメント系を含めた総合力であるとの視点に欠けている。
・システム関連部門が機械工学、情報処理などそれぞれの個別領域毎の学会別に分属している。研究者がシステム関連の学術論文を発表するにも個別領域に合わせて切り出す必要がある。
・システムズエンジニアリングの国際標準の窓口が情報処理学会となっており、日本からの代表はソフトウェア関係者ばかりで占められている。
・システム科学技術という固有の領域の存在、及びその重要性に対する認識が低い。

(5) 日本の人材育成面の課題
・国内大学の工学教育が、学問のための学問を志向し、産業競争力を高める人材の育成という視点に欠けている。
・一方で日本企業も自前主義で、社内教育でガラパゴス化した独自技術を教育しようとしている。
・システム工学科等においても、SEBoK や CorBok のような世界標準を前提としてシステムズエンジニアリング教育が行われていない。
・結果として産業界において即戦力となる人材が育たず、産業競争力の低下に結びついている。

(6) 日本のファンディング政策の課題
・システム構築方法論には、検証、妥当性確認、仕様書策定など数学に裏付けられた基盤技術もあり、学問的深さを追及できる領域もあるが、学問的深さが必ずしもシステムの顧客価値を生み出す訳ではない。技術として精緻化されることと産業化されることは別である。イノベーションを叫ぶなら、顧客価値を生み、産業化に繋がる部分にファンディングする仕組みが必要である。
・解析領域は再現性を証明しやすく、学問的に裏付けされた深さを出せるが、設計領域は解空間が広過ぎて再現性を出せない。そのような領域にもファンディングできる施策が必要である。
・産業育成・振興は経産省の役割だとしても、文科省は産業の成長エンジンとしての科学技術の振興の立場からファンディング政策を見直す必要がある。
・本当に必要な基礎研究にファンディングをしていない。応用も、実用化まで貫徹するような仕組みを創らなければいけない。
・システムは姿・形が見え難いのでファンディングを得難い。システム分野の研究者コミュニティも、そういう特性を踏まえて、システムが世の中をこう変えるというビジョンを示すことで、システムの重要性について社会的認知を得る努力が必要である。

(7) システム構築方法論の課題
・概念の点ではシステムはソフトウェアより遥か前から出現したが、開発方法論の点で

はシステム分野はソフトウェア分野に大きな遅れを取っている。
・システム構築方法論は未だ学問としての深さが足りないと見られている。
・システムとソフトウェアの違いは意識しながらも、ソフトウェア開発方法論の中でシステム構築に有効な手法群は取り入れる必要がある。
・システム構築方法論は、単にシステム構築のプロセスや技術についてのメソッドエンジニアリングに留まらず、開発プロセスの管理やシステムのライフサイクル全体を見据えた管理等のマネジメント系、さらには上流工程である価値創造やビジネス展開を含むシステム全体のストラクチャを決定するマネジメント系まで含めた概念へ拡張されなければならない。

(8) 日本が抱える諸課題に対する対策
・国内のシステム関連の研究者・実務家が集う研究集会の開催と官による支援が必要である。
・INCOSE に相当する国内のシステム関連学会を設立し、システム先進諸国との国際連携を推進する必要がある。
・ハードウェア系、ソフトウェア系、マネジメント系を包含したシステム構築方法論の確立に向けた研究活動とそれを支援するファンディング政策が必要である。同活動を通して、システムとソフトウェアの違いについて社会的認知を得る必要がある。
・ISO15288 など、最新のシステムズエンジニアリングに関する国際標準の国内普及と早期転換が不可欠である。国際標準が確立した領域で日本の独自性を追求することを止め、まだ国際標準が確立してない、より上流の価値創造領域で、日本発の国際標準の策定に向けた活動を行い、産業競争力の強化に貢献すべきである。そのために産学官並びに異分野を糾合する研究活動とその政策的支援が不可欠である。
・工学部等における世界標準に基づくシステム科学技術関連の教育を推進する必要がある。

最後に、IPA-SEC 松本所長は、「技術面ではシステム開発以上に進歩しているはずのソフトウェア開発においてすら、ソフトウェア開発は難しいものである。IPA-SEC はソフトウェア工学を何とか現場に広めていこうと活動して来たが、やはりなかなかうまくいかない。現場はなかなか開発のやり方が変わっていかない。これはひょっとするとソフトウェアの持っている本質的な開発の難しさというものがあるのではないかと思う」という趣旨の認識を閉会挨拶の中で述べられた。本ワークショップが目指したものは、システム開発における困難さの壁の打破であるので、さらにその壁の高さ、強度は大きいことになる。国内外においてシステム構築方法論への関心が高まり、欧米では米国を中心に複雑なシステムを構築するシステムズエンジニアリングに関する研究集会や国際標準制定の動きが活発化している。翻って、わが国においては未だこの分野に関する意識が低く、そのような動きから取り残される危険があることが、本ワークショップの総合討論の中で明らかになった。このような危機意識の中で、今後わが国としても、既に欧米で先行している部分については、新たにわが国独

自の標準の策定を目指すのではなく、むしろ新しい価値創造の面で、他国が手を付けてない分野で、システム化の標準を勝ち取る戦略を練る必要性が高いことが指摘された。

以上のように、システム構築方法論が抱える日本の各種課題と対策まで議論が行き着いたが、本日の講演で明らかにされた視点を参考に、システム構築方法論の研究開発が一層進められることを期待したい。

5．ワークショップ講演記録

5.0 ワークショップ趣旨スライド

JST-CRDSシステム科学ユニット主催、IPA-SEC協賛
第8回システム科学検討会

ワークショップ趣旨説明

JST-CRDSフェロー
鈴木久敏

システム科学検討会の趣旨

JST-CRDS(科学技術振興機構研究開発戦略センター)システム科学ユニットとIPA-SEC(情報処理推進機構技術本部ソフトウェア高信頼化センター)が協力して開催

- 両機関が所有する研究情報を共有
- ソフトウェア開発技術や社会の中のソフトウェア構築について、課題解決に向けて議論
- 新たな課題の発掘、今後の研究開発戦略に役立てる
- ソフトウェア開発の難しさを学ぶことを通して、システム技術の研究課題とシステム構築方法論を探求
- ソフトウェア開発方法の標準化の中でモデリング技術や可視化技術など、アカデミアに求められる研究要素を抽出

システム科学検討会の記録

- **第1回 (2013. 11. 25)**
  - ソフトウェアシステム構築の困難性［松本］
  - システム構築型イノベーションの重要性とその実現に向けて［木村］
- **第2回 (2013. 12. 27)**
  - システムズエンジニアリングの最新動向と慶應SDMの取組み［白坂］
- **第3回 (2014. 1. 23)**
  - 大規模開発を成功に導くアプローチ —あるシステムの開発—［大島］
- **第4回 (2014. 2. 19)**
  - 非機能要求グレード ～システム基盤に対する非機能要求の合意形成～［山下］
- **第5回 (2014. 3. 26)**
  - 情報システムにおけるデザイン科学研究［田名部］
  - ミニワークショップ システム関連キーワードの抽出とマッピング
- **第6回 (2014. 4. 25)**
  - ソフトウェアと競争戦略：競争戦略論の視点から［立本］
- **第7回 (2014. 5. 9)**
  - SECを通した技術移転 —形式手法の場合—［新谷］
- **第8回 (2014. 6. 19)**
  - ワークショップ「システム構築方法論 —現在の到達点と今後の課題— 」

本日のWSの趣旨

【背景】

- 世の中でソフトウェアとシステムが混同されている
- システムは、ある目的を達成するために機能要素を適切に結び付けた複合体(人・組織、制度、ハード、ソフトを含む)
- システムの機能や付加価値実現にソフトウェアを利用
- システムが先に世の中に現れたが、後発のソフトウェアが急速に拡大・進歩。ソフトウェアに学ぶべきところは多い。

【狙い】

- ソフトウェア開発の難しさを学ぶことを通して、システム技術の研究課題とシステム構築方法論を探求
- ソフトウェア開発方法の中で標準化された諸技法(形式仕様、Agile開発、モデリング、要求工学、妥当性確認など)を通して、システム構築に求められる研究要素を抽出

総合討論

ソフトウェア開発とシステム開発との関係

(1) ソフトウェア開発とシステム開発の同質性、異質性

(2) ソフトウェア開発に使われる技法で、システム開発にも使えるもの、使えないもの

システム構築方法論

(1) 現在の到達点

(2) 今後の課題

本日のプログラム

13:00-13:05 開会挨拶 木村 英紀(独)科学技術振興機構研究開発戦略センター上席フェロー
13:05-13:10 ワークショップの趣旨説明
13:10-13:50「FeliCa開発における形式仕様記述VDM++の適用について」
栗田 太郎 氏 ソニー(株) FeliCa事業部モバイル設計部
13:50-14:30「Agile開発コンセプトとその動向」
濱 勝巳 氏 (株)アッズーリ 代表取締役
14:30-15:10「モデルに基づくシステムズエンジニアリング(MBSE)とSysML」
西村 秀和 氏 慶應義塾大学システムデザイン・マネジメント研究科教授
(15:10-15:25) 休憩
15:25-16:05「社会ニーズ把握のための技術－要求工学の動向－」
山本修一郎 氏 名古屋大学情報戦略室教授
16:05-16:45「妥当性確認とアシュランスケース」
木下 佳樹 氏 神奈川大学理学部情報科学科教授
16:45-17:25 総合討論
17:25-17:30 閉会挨拶 木村 英紀(独)科学技術振興機構研究開発戦略センター上席フェロー

5.1 栗田太郎氏ご講演スライド

FeliCa Networks

科学技術振興機構 研究開発戦略センターさま
【第 8 回システム科学検討会】

モバイル FeliCa の開発における形式仕様記述手法 VDM の適用
～システム開発への形式手法の適用による品質の確保～

フェリカネットワークス株式会社 開発部 2 課
栗田 太郎
2014 年 6 月 23 日

概要

少人数ではないチームにおいて，設計や実装，テストを正確に行い品質を確保するために，また，運用や保守，派生開発において品質を維持し続けていくために，仕様や文書はどのような役割を果たすのでしょうか．「おサイフケータイ」の開発プロジェクトにおける，形式仕様記述手法の適用により品質を確保した事例とともに，文書の記述力やチームのコミュニケーション力を向上させるための取組みとその結果を紹介しながら，そのことについて考えていきます．

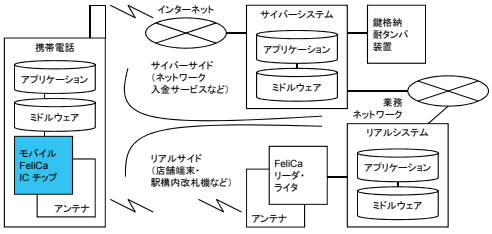
 FeliCa Networks

Agenda

1. 自己紹介・会社紹介 ～モバイル FeliCa とプロジェクトの紹介～
2. ソフトウェア開発現場の現実と課題 ～記述とコミュニケーションの問題～
   - 2.1. 私たちの現実 1 仕様書?
   - 2.2. 私たちの現実 2 研修その 1
   - 2.3. 私たちの現実 3 研修その 2
3. 形式手法の現在 ～形式仕様記述とモデル検査の概説～
4. 組込みソフトウェア開発における形式仕様記述手法の適用事例 ～目的・範囲・方法・結果・考察～
   - 4.1. 目的・適用範囲・方法
   - 4.2. 結果 1 仕様の記述実績と不具合原因の割合
   - 4.3. 結果 2 仕様のテストと実装のテスト
   - 4.4. 結果 3 仕様とコミュニケーション
   - 4.5. 考察・まとめ
5. 業務におけるチームでの効率的な「検算」～「ストレスフリー指向開発」に向けて～

 FeliCa Networks

モバイル FeliCa システム 1



 FeliCa Networks

「おサイフケータイ」の主な用途とユーザ 1

【主な用途】
- 電子マネー
- 公共交通機関の乗車券・定期券
- クレジットカード
- キャッシュカード

【ユーザ】
- 一般ユーザ
- コンテンツプロバイダ (サービス事業者)
- 公共交通サービス事業者
- 携帯電話機メーカ
- チップメーカ
- 移動体通信事業者 (キャリア)

株主 = 株式会社エヌ・ティ・ティ・ドコモ
+ 東日本旅客鉄道株式会社
+ ソニー株式会社

 FeliCa Networks

モバイル FeliCa IC チップファームウェアの開発 1

- 電子マネーや公共交通機関の乗車券，定期券などに応用される，社会インフラを構成するセキュリティソフトウェアである
- 日本中の携帯電話に組み込まれる

- 社会的責任が重い
- システムやサービスの根幹に関わる重大なトラブルが発生することで，フェリカネットワークス自身のみならず，一般ユーザの生活や，サービス事業者，携帯電話メーカ，移動体通信事業者のビジネスに影響が及ばないよう，品質の確保に当たらなければならない

 FeliCa Networks

私たちの現実 2

- ソフトウェアの信頼性や安全性に対する関心が高まってきた
- これまで以上に効率良く開発することが求められてきている

- 普通の現場で開発に携わっているのは普通のチームとメンバ
- 現場にプレッシャーをかけても品質は高くならずストレスばかりがかかる

 FeliCa Networks

問題のひとつ 3

- ソフトウェアの開発現場では，曖昧な仕様に起因するトラブルが多い
- 下流工程の修正コストは，上流工程の 10^n 倍になる
- 正しくないプログラムの欠陥修正は非常に難しい，または不可能である

- 自然言語により曖昧かつ不完全な仕様を記述している
- 仕様記述と称して，手続的日本語プログラミングを行っている
- 業務を知らないコーダーが手続的にプログラミングを行っている
- 結果として，重大な欠陥を含み，保守性・再利用性のないソフトウェアが大量に生産されている

 FeliCa Networks

正しい (?) 仕様の記述 3

- 正しい仕様を自然言語で書くのは困難である (形式仕様記述言語を用いても困難である)
- 曖昧さの排除が困難である
- ツールによるチェックができない
- 仕様を書きながら「擬似コード」を考えなければならないことが多くあるが，文法の検討に時間がかかる
- 図表のレイアウト検討や記述にも時間がかかる
- 変更管理が困難である
- 改善方法やその基準が分からない… (仕様書がメンテナンスできない，されない…)
- 仕様っぽいものが誰にでも書けてしまう…

- UML なども適用が困難である?
  - 仕様記述言語がなく (あるいはまだ整備中で)，自然言語で詳細を記述するため，結局自然言語仕様と同様の問題が発生する
  - ツールによるチェックがほとんどできない
  - 記法に曖昧さが多いため，形式手法より学習が困難である
  - 再利用のための仕掛けがない

 FeliCa Networks

プログラムは「数学系」 3

- C/C++ 言語や Java 言語で書かれたプログラムは，検証が困難な数学系 (数理モデル) である
- 誰かが，どこかで，数学系に変換しなければならないのだが，現場では業務を知らないコーダーが数学系に変換している
- 誰かが，どこかで，数学系に変換しなければならないのであれば，最初から数学系で書いてもいいだろう

「いつか最後に世界を見ようと思ってたんですが，それなら最初でもいいだろう」(平田オリザ)

 FeliCa Networks

形式手法とは 3

- 形式手法とは，システム開発，特にソフトウェア開発の手法で，数理論理学に基づく科学的な裏付けを持つ仕様記述言語を用いて設計対象を表現することにより，ある側面の仕様を厳密に記述し，開発の各工程で利用する手段の総称である
- 形式記述を行うことで曖昧さが取り除かれ，機械処理が可能になり，様々な可能性が開ける
- 形式手法の一分野であるモデル規範型の手法と，形式仕様記述言語，仕様の記述や検証のためのツールの活用により，厳密な仕様の記述や多面的な検証が可能となる上，手法の導入は，システム開発におけるステークホルダ間のコミュニケーションの活性化に寄与する

 FeliCa Networks

形式手法の適用レベル 3

【レベル 0】

形式的に仕様を記述する．数理論理学に基づく記法を用いて厳密に仕様を記述する．この記述を元にプログラムを開発する．

【レベル 1】

形式的に開発と検証を行う．段階的詳細化により仕様からプログラムを作成したり，仕様やプログラムの性質を証明したりする．

【レベル 2】

機械支援による証明を行う．定理証明器や証明支援器を用いて，仕様やプログラムの性質を証明する．

- 私たちの適用はレベル 0

 FeliCa Networks

形式手法の一部の分類 3

モデル規範: VDM・Z 記法・B メソッド等と，ツールを利用

- 集合論や命題論理，述語論理が基礎となる
- 不変条件，事前条件，事後条件を記述する
- 情報の構造や，ある状態から他の状態への変化をモデル化する
- 専用言語の利用によるコンパクトな仕様記述，曖昧さの除去，仕様の「実行」，「テスト」，「回帰テスト」，定理証明等の可能性が広がる

モデル検査: PROMELA，LTS 等の言語と SPIN (PROMELA)，LTSA (LTS) 等のツールを利用

- 振る舞い仕様を，状態遷移モデルと，モデルが満たす条件として記述することで，全状態空間を生成し，自動検査する
- 従来の「テスト」では見つからない潜在的な障害を発見できる

 FeliCa Networks

モバイル FeliCa における仕様の記述 3

型: モデル規範型

仕様記述言語: VDM++

ツール: VDMTools

 FeliCa Networks

VDM++ ＋ VDMTools 3

- VDM の選定理由
  - 仕様策定段階から動作可能
  - モデル化から動作確認まで広い範囲での適用が可能である
- VDM の特徴
  - VDM++ = VDM-SL (ISO で標準化) + OO
  - VDMTools
    - 仕様の構文チェック
    - 仕様の型チェック
    - 証明課題の生成
    - 実行可能仕様の逐次実行とデバッグ支援
    - 実行可能仕様のコードカバレッジ計測
    - 実行可能仕様から C++ 言語や Java 言語への変換
    - Java 言語から VDM++ 言語への変換
    - 各種 CASE ツールとの連動
    - 仕様の清書支援

 FeliCa Networks

制御・改善可能な開発プロセスへ向けて 4

検証可能なモデルを

プロジェクトにおいて終始メンテナンス対象とし

仕様，モデルを議論の出発点に

仕様，モデルをコミュニケーションのフィードバック先として

「仕様 – 設計 – 実装 – テスト」をイテレーティブに成長させていく

プロセス，プロジェクトの事例を紹介します．

 FeliCa Networks

プロジェクトの概要 4

【ゴール】
フェリカネットワークス発の次世代モバイル FeliCa サービスの実現
【具体的な目標】
モバイル FeliCa IC チップファームウェアおよび周辺ツールの開発
【期間】
約 3 年 3 ヶ月
【メンバ数】
約 55 名
【平均年齢】
約 30 歳

FeliCa Networks

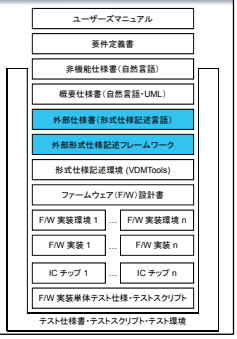
目的 4.1

- 厳密な仕様の策定と記述の継続
  - 第三者テストが可能になる
  - 運用・保守が可能になる
  - 再利用性が向上する
- 仕様を活用したイテレーティブな開発プロセスの確立
- 仕様の多方面からの精査
- 仕様のテスト
  - 仕様のテスト
  - 「テスト仕様」のテスト
- コミュニケーションの促進

FeliCa Networks

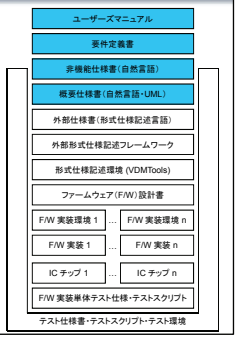
適用範囲 – モバイル FeliCa OS 4.1

- 実行可能な外部仕様を形式的に記述，テスト（形式手法のレベル 0）
- 仕様記述フレームワーク上に仕様を展開
  - VDM++ 汎用ライブラリ
  - ドメイン固有の用語や型や述語の定義
  - ドメイン固有のデータ構造
  - 仕様記述フレームワーク
  - 仕様記述テンプレート
  - テスティングフレームワーク
- フレームワークと仕様の境界，すなわち動作する仕様と仕様の明確化が困難であった



適用範囲 – モバイル FeliCa OS 4.1

- 以下は自然言語で記述した
  - ユーザーズマニュアル
  - 要件定義書
  - 非機能仕様書
  - 概要仕様書
- 形式言語と自然言語それぞれによる記述の整合性確保，自然言語で書いた文書の品質確保に課題が残る

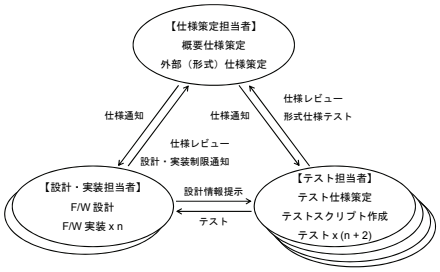


チームへの形式手法の導入 4.1

- 当初，形式手法や形式仕様記述手法に通じているメンバはいなかった
- 専門家による 5 日間の教育を受けた
- VDM++/VDMTools は，ソフトウェア開発技術者にとって馴染みやすいものである

FeliCa Networks

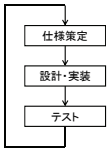
チーム構成 4.1



FeliCa Networks

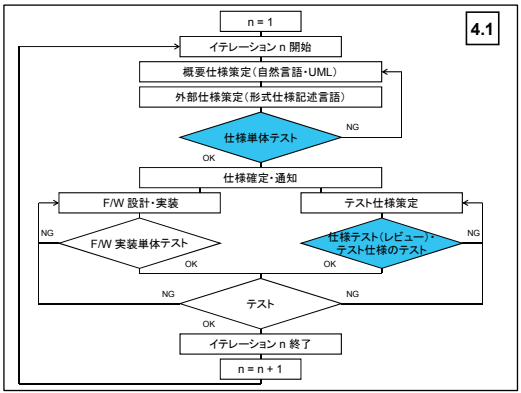
開発プロセス 4.1

イテレーティブな開発プロセスを組み立てた．



FeliCa Networks

4.1

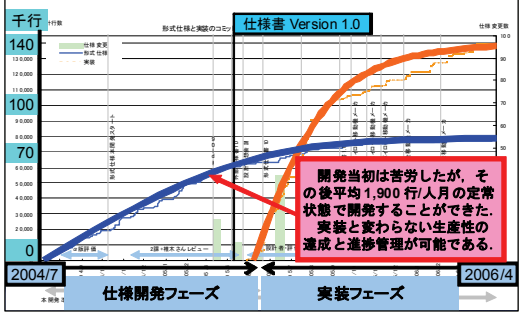


形式仕様記述手法の適用成果 4.2

- 仕様記述言語 VDM++ と仕様開発環境 VDMTools を用いて，外部機能仕様を，動作する形式仕様として表した
- 作成した仕様書は以下の通り:
  - 自然言語による 383 ページのプロトコル仕様書
  - 形式仕様記述言語 VDM++ による 677 ページの外部仕様書
- 仕様書のコード量はテストコードを含め，約 10 万行
- C++ 言語によるファームウェアのソースコードは一種類のチップにつき約 11 万行
- 開発時における仕様関連のトラブルは少なかった
- IC チップの出荷後，ファームウェアの品質に関連するトラブルはない

 FeliCa Networks

仕様開発と実装の成長曲線 4.2



仕様開発フェーズにおけるデバッグ 4.2

表 1 仕様開発フェーズで修正した誤りの件数

誤りの発見工程	件数
仕様の記述	162
仕様の実行と単体テスト	116
仕様のレビュー	93
設計実装担当者・テスト担当者とのコミュニケーション	69
合計	440

「デバッグ密度」= 440 / 40,000 = 約 11 エラー / 千行

? 形式手法は，開発の初期段階における誤りの発見に有効である

 FeliCa Networks

ファームウェア実装フェーズにおける不具合原因の割合 4.2

表 2 不具合原因の割合

不具合原因	割合
仕様記述もれ	0.2%
仕様記述誤り	0%
仕様不明確	1.8%
仕様見落とし	5.6%
仕様理解不足	10.7%
仕様確認不足	0%
仕様変更通知不徹底	0.2%
その他仕様関連外	81.5%

? 仕様は書けている

 FeliCa Networks

ファームウェア実装フェーズにおける不具合原因の割合 4.2

表 2 不具合原因の割合

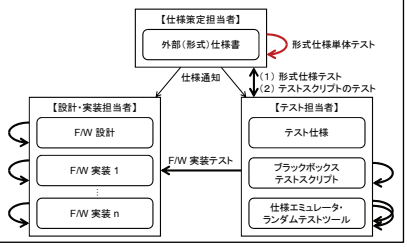
不具合原因	割合
仕様記述もれ	0.2%
仕様記述誤り	0%
仕様不明確	1.8%
仕様見落とし	5.6%
仕様理解不足	10.7%
仕様確認不足	0%
仕様変更通知不徹底	0.2%
その他仕様関連外	81.5%

? 「読ませる仕様」の記述が課題である

? DSL (仕様記述フレームワーク) の検討と構築が重要

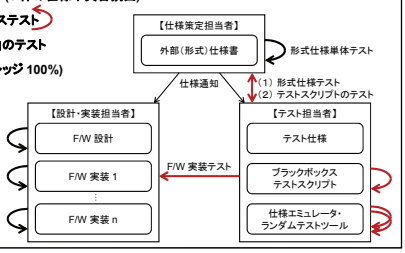
結果・考察 – 仕様のテストと実装のテスト (1/4) 4.3

- 仕様単体テスト
- カバレッジ: 82%
- 結果的に品質にとくに問題はなかった



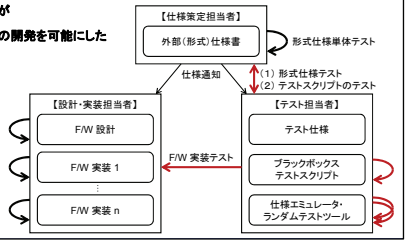
結果・考察 – 仕様のテストと実装のテスト (2/4) 4.3

- ブラックボックステスト x 20,000 項目 + ランダムテスト x 数億項目
- 仕様のテスト (6 件の仕様不具合摘出)
  - クロステスト
- 「テスト仕様」のテスト
  - (カバレッジ 100%)



結果・考察 – 仕様のテストと実装のテスト (3/4) 4.3

- ブラックボックステスト x 20,000 項目 + ランダムテスト x 数億項目
- 仕様エミュレータの開発に活用
- 厳密な仕様が
  エミュレータの開発を可能にした



結果・考察 – 仕様のテストと実装のテスト (4/4) 4.3

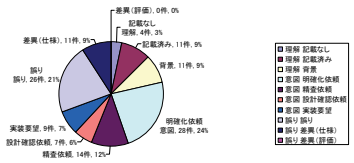
- 実装だけに有益なのではない
- 厳密な仕様はテストにも活用することができる
- 仕様 = テスト仕様・テスト環境 = 実装 が確認できる
- テスト担当者は緊張するとともに安心もできる
- 自然言語による仕様のみの場合，テスト担当者はよりどころになるものがない

FeliCa Networks

結果・考察 – 仕様とコミュニケーション (1/3) 4.4

- 自然言語によるマニュアル

→ 明確化依頼が多い



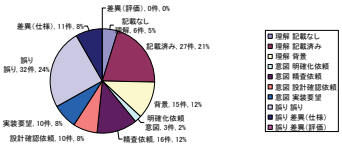
FeliCa Networks

結果・考察 – 仕様とコミュニケーション (2/3) 4.4

- 形式仕様記述言語による外部仕様書

→「理解」に関する質問が多い

→ 仕様策定背景・経緯はコメントに記述する



FeliCa Networks

結果・考察 – 仕様とコミュニケーション (3/3) 4.4

- 自然言語・形式仕様記述言語の違い

→「理解」と「明確化依頼」以外は似通っている

- 自然言語

→ まずよくわからない…「明確化依頼」=「わかった」でとどまってしまう

- 形式仕様記述言語

→ 中途半端に「理解」できない．背景までつきつめて「理解」したい，納得したい

- 日本語での議論はつらい?

→ 記述から人格を消して「問題対私たち」とする

FeliCa Networks

まとめ 4.5

- 上流工程，とくに仕様策定工程における成果物の品質向上に効果
- 仕様策定・実装・テストのイテレーションを多数回しても，ノイズが増幅されず，仕様を洗練することができる
- プロジェクト内のコミュニケーションの活性化に寄与する
- 他の工夫との組み合わせにおいて効果を発揮する
- 開発ドメインに応じて，他の手法を組み合わせる

FeliCa Networks

形式手法の活用の効果 4.5

- 直接的な効果
  - 記述と検証 → 書くだけでも効果があるが，ゆくゆくは証明や段階的詳細化へ?
  - 品質の確保

→ チームによる記述や検算の可能性を開く

- 間接的な効果
  - ドメインに対する認識・理解
  - コミュニケーションの活性化
  - 技能の向上 (ドメインエンジニアリング，エンジニアリング，コミュニケーション)

→ ストレスの軽減や生活の改善につながる

FeliCa Networks

5.2 濵勝巳氏ご講演スライド

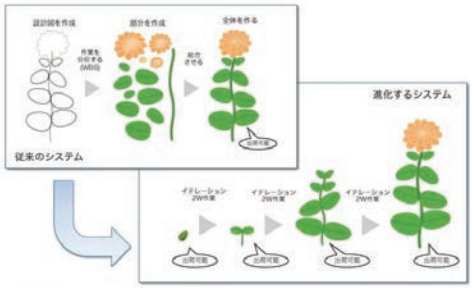
第8回システム科学検討会 2014.6.23
Agile開発コンセプトとその動向
濱 勝巳 ～想いを色に～ 株式会社アッズーリ

Manifesto for Agile Software Development, 2001.2

- Individuals and interactions
  over processes and tools
  細かいプロセスやツール整備に走るより、個人の能力発揮と人と人 との相互作用やコミュニケーションが大切だよ
- Working software
  over comprehensive documentation
  読まれもしないドキュメンテーション作成より、動くソフトウェアの方が大切だよ
- Customer collaboration
  over contract negotiation
  契約や交渉ごとより、顧客との協調の方が大切だよ
- Responding to change
  over following a plan
  計画遂行より、変化への対応の方が大切だよ

それなりにインパクトがありました・・・

「作る」から「育てる」へ



新しいパラダイム(New Paradigm)

Making things
Conversation

母を横に乗せ、ハンドルを握った時のことです。最初はうまく行ったように見えました、でも、ちょっと油断した瞬間に車は道路を外れて砂利に突っ込んでしまいました。母は車を道に戻すと、初めて運転とは何かを教えてくれました。

「運転で大切なのは、車を正しい方向に進めることじゃないのよ。大切なのは、常に注意を払って細かく左右に方向修正していくことなのよ。」

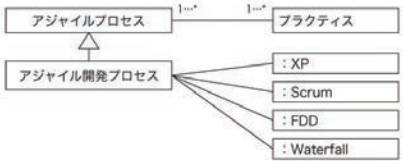
2001年4月『XPとアジャイル開発セミナー』ケントベック氏講演より

間違ったアジャイルプロセス

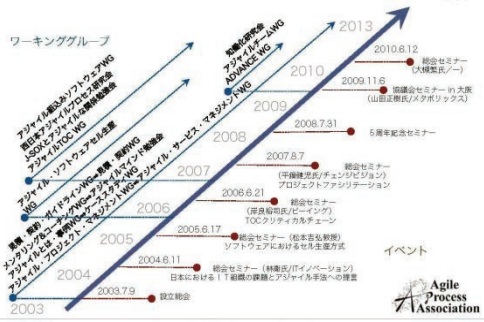
- プロセスの誤解
  - プロトタイプ開発ではありません
  - インクリメンタル開発やフェーズ開発ではありません
  - 万能ツールではない。マルチプルパスは必須
  - 時間の要素を含むため簡単ではない
- ドキュメントの誤解
  - 中間成果物は作成しない。残すドキュメントは作成しなければならない
- 対象者の誤解
  - ベンダー側ではなく利用者のためのプロセスです
  - エンジニアの幸せのためではない
- ゴールの誤解
  - ソフトウェア開発がゴールではない。機能分割、WBSでは導入できません
  - イノベーションは起こさない。収束する

様々なアジャイルプロセス

- プラクティスの集合
- ソフトウェアプロダクトに依存しない



アジャイルプロセス協議会の歩み

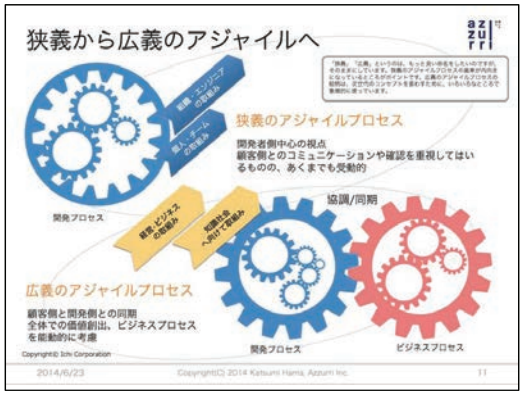


協議会のWG活動

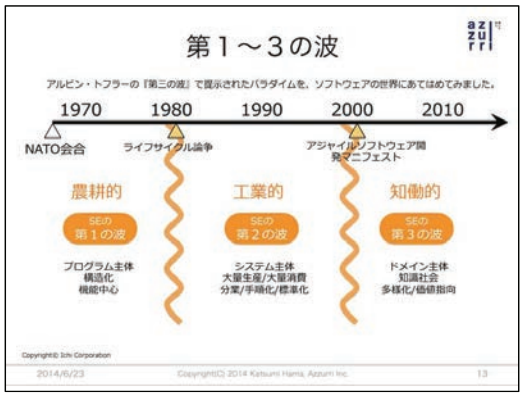
- 見積・契約・ガイドラインWG⇒見積・契約WG (東京：2003-現在)
- メンタリング&コーチングWG⇒アジャイルマインド勉強会 (東京：2003-現在)
- アジャイルとは・事例WG ⇒ケーススタディWG (東京：2003-2006)
- ジャイル・プロジェクト・マネジメントWG (東京：2003-2010) ⇒アジャイル・サービス・マネジメントWG (東京：2010-現在)
- アジャイル・ソフトウェアセル生産WG (東京：2005-2009)
- アジャイル組込みソフトウェアWG (東京：2005-2009)
- 西日本アジャイルプロセス研究会(大阪：2008-2012)
- J-SOXとアジャイルな関係勉強会(大阪：2007-2009)
- アジャイルTOC WG (東京：2006)
- 知識化研究会 (東京：2009-現在)
- アジャイルチームWG (東京：2009-現在)
- ADVANCE WG (東京：2009-現在)
- アジャイルテストWG (東京：2012-現在)
- アジル経営WG (東京：2012-現在)

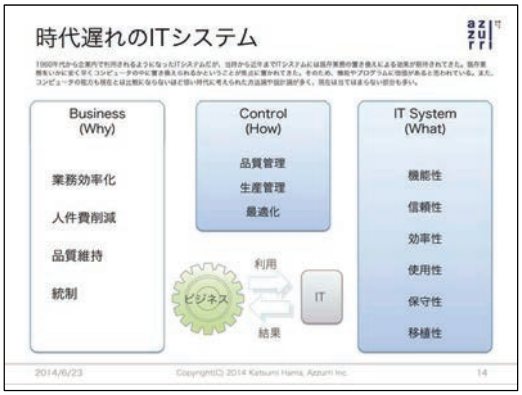
Agile Process Association

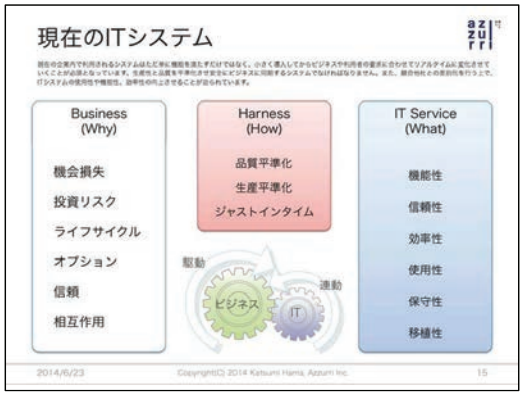

協議会の活動の結果
❖ 啓蒙の時代：2003～2005年
– 設立2003年2月、設立総会2003.7.9
– セミナー、メディアによる啓蒙活動
❖ 活動の時代：2005～2009年
– 事例の発表。見える化の流行
– ワーキンググループ活動最盛期
– 地方活動の立ち上げ
– 会報誌発行 2009年から
❖ 多様の時代：2010～2014年
– 大阪支部が一般財団法人 西日本アジャイルプロセス協議会を設立し独立
– 島根支部が Agile Shimane として独り立ち
– 知のフリマに60名の参加（2012年）
Agile Process Association
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
10

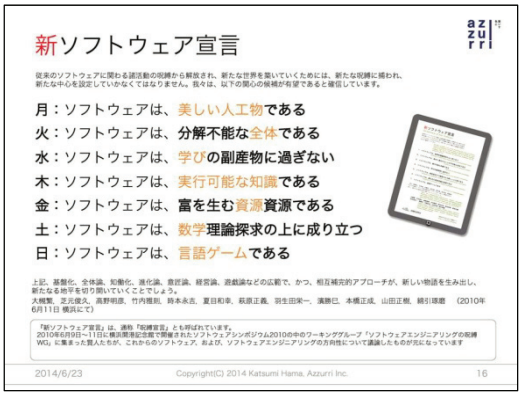

狭義から広義のアジャイルへ
狭義のアジャイルプロセス
開発者側中心の視点
顧客側とのコミュニケーションや確認を重視してはいるものの、あくまでも受動的
協調/同期
開発プロセス
広義のアジャイルプロセス
顧客側と開発側との同期
全体での価値創出、ビジネスプロセスを能動的に考慮
開発プロセス
ビジネスプロセス
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
11

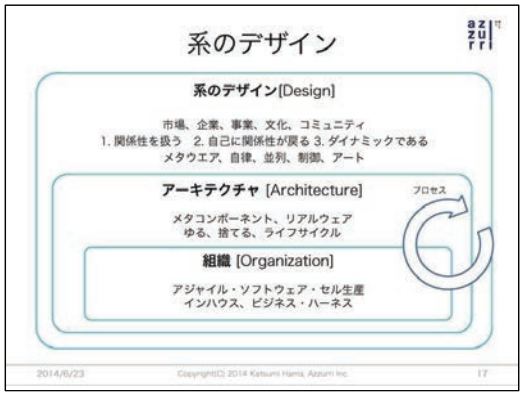

人働説から知働説へ
人月の神話
人働説
天動説
価値
技術
価値
知働説
知働化研究会
http://www.evokt-lab.org/
研究室：ソフトウェア哲学研究所（濱 崇巳）
知働化研究会
✓ソフトウェアとは《実行可能な知識》である (Executable Knowledge)
✓ソフトウェアとは実行可能な知識を紡いだ《様相》である (Texture)
✓ソフトウェアを《作ること》と《使うこと》とは本質的に同じ
✓中心は《機能》から《様相》へ
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
12


第１～３の波
アルビン・トフラーの『第三の波』で提示されたパラダイムを、ソフトウェアの世界にあてはめてみました。
1970
1980
1990
2000
2010
NATO会合
ライフサイクル論争
アジャイルソフトウェア開発マニフェスト
農耕的
SEの第１の波
プログラム主体
構造化
機能中心
工業的
SEの第２の波
システム主体
大量生産/大量消費
分業/手順化/標準化
知働的
SEの第３の波
ドメイン主体
知識社会
多様化/価値指向
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
13


時代遅れのITシステム
Business (Why)
業務効率化
人件費削減
品質維持
統制
Control (How)
品質管理
生産管理
最適化
IT System (What)
機能性
信頼性
効率性
使用性
保守性
移植性
利用
ビジネス
IT
結果
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
14


現在のITシステム
Business (Why)
機会損失
投資リスク
ライフサイクル
オプション
信頼
相互作用
Harness (How)
品質平準化
生産平準化
ジャストインタイム
IT Service (What)
機能性
信頼性
効率性
使用性
保守性
移植性
駆動
連動
ビジネス
IT
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
15


新ソフトウェア宣言
月：ソフトウェアは、美しい人工物である
火：ソフトウェアは、分解不能な全体である
水：ソフトウェアは、学びの副産物に過ぎない
木：ソフトウェアは、実行可能な知識である
金：ソフトウェアは、富を生む資源資源である
土：ソフトウェアは、数学理論探求の上に成り立つ
日：ソフトウェアは、言語ゲームである
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
16


系のデザイン
系のデザイン[Design]
市場、企業、事業、文化、コミュニティ
1. 関係性を扱う 2. 自己に関係性が戻る 3. ダイナミックである
メタウエア、自律、並列、制御、アート
アーキテクチャ [Architecture]
プロセス
メタコンポーネント、リアルウェア
ゆる、捨てる、ライフサイクル
組織 [Organization]
アジャイル・ソフトウェア・セル生産
インハウス、ビジネス・ハーネス
2014/6/23
Copyright(C) 2014 Katsumi Hama, Azzurri Inc.
17

5.3 西村秀和先生ご講演スライド

第8回システム科学検討会WS
2014年6月23日(月)

モデルに基づくシステムズエンジニアリング
MBSEとSysML

慶應義塾大学 大学院 システムデザイン・マネジメント研究科
教授 西村 秀和 http: lab.sdm.keio.ac.jp/nismlab/

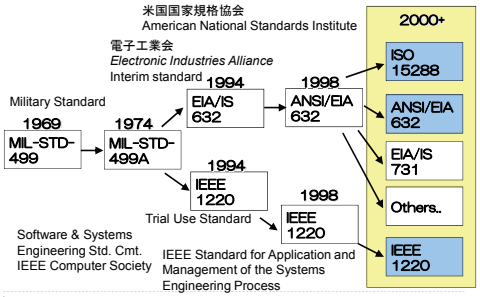
1

システムズエンジニアリングとは何か？

- システムズエンジニアリングの定義
  - システムを成功裏に実現するための複数の分野にまたがるアプローチおよび手段
- システムズエンジニアリングでは、開発のライフサイクルの初期段階で顧客のニーズを明確化し、機能要求を定義し、関連する問題をすべて考慮しながら設計のための総合とシステムの妥当性確認を進める。
- システムズエンジニアリングは、ビジネスとすべての顧客の技術的要求の両者を考慮し、ユーザーニーズに合致した品質の製品/サービスを供給することを目的とする。
  - INCOSE: International Council on Systems Engineering

2

システムズエンジニアリング標準の歴史



出典: INCOSE Systems Engineering Handbook 3

システムズエンジニアリングプロセスの概要

システムズエンジニアリングを構成する4つのアクティビティ

1. システム設計
   - 要求分析、アーキテクチャ設計を実施し、サブシステムへの要求を導出する.
2. インテグレーション(統合)
   - 検証済みのサブシステムを統合する.
3. 評価・解析
   - エンジニアリング活動における解析および検証(verification)・妥当性確認(validation)等を実施する.
4. システムズエンジニアリングマネジメント
   - QCDを満たすように、各種活動の計画・実施・評価を行う.

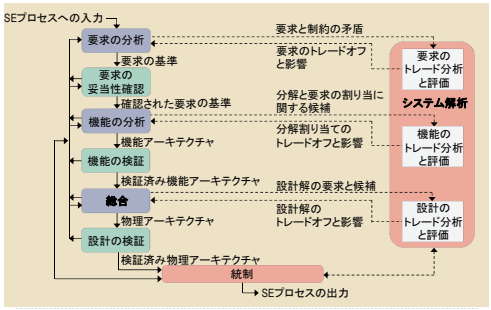
4

ISO/IEC 12207と15288との関係

- **ISO/IEC 12207:2008**
  Systems and software engineering -- Software life cycle processes
- ISO 12207は以下のISO 15288のプロセスに関しては，ソフトウェアライフサイクルに特化している.
  - Agreement Processes
  - Organizational Project-Enabling Processes
  - Project Processes
- ISO 12207の"Technical Processes"では，Verification,Transition,Validationのプロセスで，ISO 15288の対応するプロセスの結果の達成に寄与する.他のプロセスではソフトウェアライフサイクルに特化している.

5

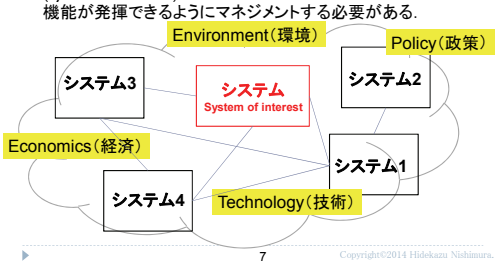
IEEE 1220 systems engineering process



6

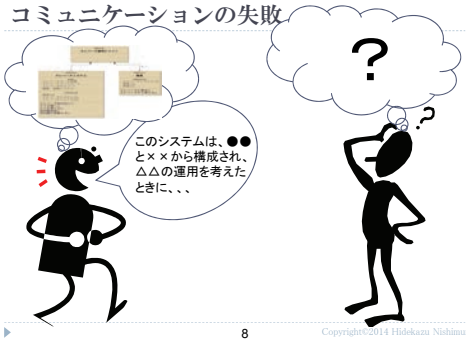
System of Systems Architecture

- 様々なシステムが複雑に関連する中で、対象とするシステム(System of interest)を設計することは極めて難しいが，所望の機能が発揮できるようにマネジメントする必要がある.



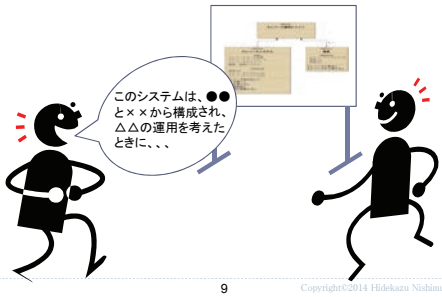
7

コミュニケーションの失敗



8

図を用いたコミュニケーション



9 Copyright©2014 Hidekazu Nishimura.

“モデルベースでシステムを考える”とは？

- **モデル*に基づくシステム開発**
  - 仕様書など文書だけではすぐに理解できないことが、**図的に表現する**ことで理解が容易になる。
  - **協働してシステム開発をする**には、共通言語が必要であり、それをサポートするには**図的な言語**が有効である。
  - **モデルを再利用**することにより開発の効率化が期待できる。
  - モデルを用いて**抽象度を上げる**ことにより**革新**に導く。

***注：実行可能ではないモデルを含む．**
もちろん，実行可能なモデルも含む．

10 Copyright©2014 Hidekazu Nishimura.

INCOSE IW 2014 -MBSE WS-

- Digital systems engineeringの必要性 例：Industry 4.0 （ドイツ）
  - システムズエンジニアリングが，プロジェクトマネジメントや政策決定者，意思決定者に対して，意思決定のための分析の支援を行う．
- Model-based architectures 正しいシステムモデルを得る！（Boeing）
  - Requirement Architecture 正しい要求
  - Functional Architecture 正しい機能，インタフェース
  - Logical Architecture 正しいコンポーネント
- モデリングのスキル （Boeing）
  - 問題に応じた詳細度で何をモデル化するべきかを知る．
  - 解析するべきデータは何か，それをどのようにモデルで解析するかを知る．
- 価値の最大化，Value-driven Systems Engineering （Georgia Institute of Tech.）
  - SoSでアーキテクチャを考える．モデルベースで考える．

11 Copyright©2014 Hidekazu Nishimura.

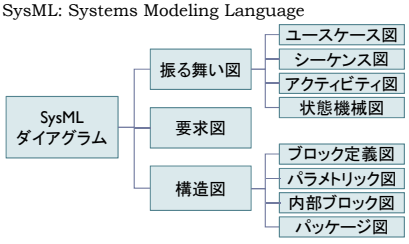
システムモデルの記述

- **システムモデル表記法**：SysML（Systems Modeling Language）
  - システムを**構造**，**振る舞い**，**要求**，**パラメトリック制約**の観点で図的に表現することができる。
  - 図的表現により、**開発者の思考**を支援できる。
  - 複数のドメインにまたがる開発、分業化された開発環境で、**共通言語**として利用できる。
  - システム開発プロセスの中で**要求のトレーサビリティ**が確保される。
  - **構成管理**、**変更管理**が容易になる。ーあるサブシステムやコンポーネントの要求の変更や設計の変更が生じた際に、他のサブシステムやコンポーネントにどのような影響が及ぶかを判断できる。

12 Copyright©2014 Hidekazu Nishimura.

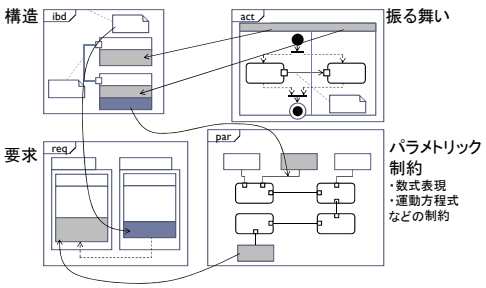
SysMLダイアグラムの分類

SysML: Systems Modeling Language



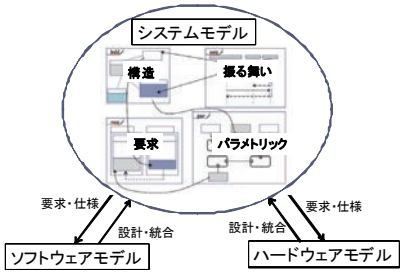
13 Copyright©2014 Hidekazu Nishimura.

SysMLのダイアグラムは，互いに関連している。
→ 設計変更があった場合にもその影響を容易に把握できる．



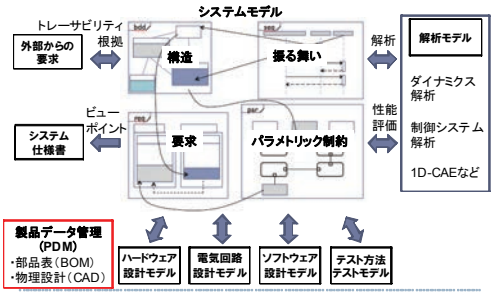
14 Copyright©2014 Hidekazu Nishimura.

SysMLを用いた協働作業



15 Copyright© 2014 Hidekazu Nishimura.

コンカレントデザインを促進するフレームワーク



16 Copyright© 2014 Hidekazu Nishimura.

SysMLによるMBSEのサポート

システム仕様の決定 ← 以下の活動の反復

- ブラックボックスのシステム要求の取得と分析（コンテキストレベル）
  - 要求管理ツールでの文書ベースでの要求の獲得
  - SysMLモデリングツールへの要求のインポート
  - システムのユースケースからシステムレベルでの機能の特定
  - ユースケースと要求間のトレーサビリティの獲得
  - ユースケースシナリオの実現：アクティビティ図，シーケンス図，状態機械図
  - システムコンテキスト図の創出
  - システム検証をサポートするシステムのテストケースの特定
- 要求を満たすシステムアーキテクチャ候補の開発
  - ブロック定義図を用いたシステムの分解
  - アクティビティ図またはシーケンス図を用いたパート間の相互作用の定義
  - 内部ブロック図を用いたパート間の相互接続の定義

17

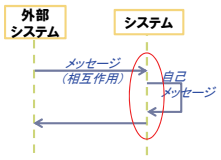
SysMLによるMBSEのサポート（続き）

- 所望のアーキテクチャの評価と選択のためのエンジニアリング解析とトレードオフ分析の実行
  - 性能，信頼性，コスト，その他の重要なプロパティの分析をサポートするためのパラメトリック図を用いたシステムプロパティの制約の獲得
  - システムプロパティの予算を決定するためのエンジニアリング解析の実行（通常，別のエンジニアリング解析ツールで行われる）
- コンポーネント要求の規定とシステム要求に対するコンポーネント要求のトレーサビリティの規定
  - アーキテクチャにおける，各コンポーネント（ブロック）のための機能要求，インタフェース要求，性能要求の獲得
  - システム要求へのコンポーネント要求のトレース
- システムレベルでのテストケースの実行→システム設計に関する要求の充足の検証

18

機能の明確化とインタフェース（1）

- コンテキストレベルでの外部システムとのインタフェース
- システムのユースケース（使われ方、動作）を考える。
- シーケンス図による記述　パート間の相互作用



機能を記述するユースケースをシーケンス図で記述し、システムの機能の抽出を行う。

↓

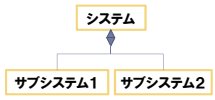
・システムは、外部システムからメッセージを受けるという機能1をもつ。
・システムは、自己メッセージを処理するという機能2をもつ。

次に、抽出された各機能を実現するシーケンスを考える。

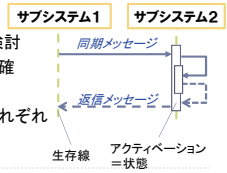
19

機能の明確化とインタフェース（2）

- システムの分解を検討する。
- ブロック定義図



・サブシステム1はサブシステム2から同期メッセージを受け、サブシステム2に返信する。
・サブシステム2は自己メッセージを処理する。

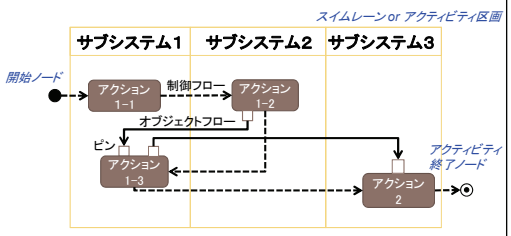


- 機能1を実現するシーケンスを検討
- サブシステム間の相互作用を明確にする。
- 各サブシステムの生命線上にそれぞれの機能が抽出される

20

機能のコンポーネントへの割り当て

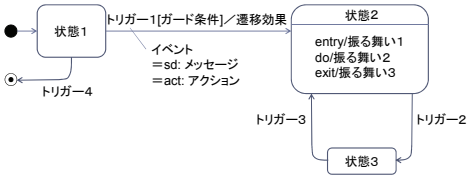
- アクティビティ図
- コンポーネントとインタフェースの明確化



21

状態機械図（ステートマシン図）

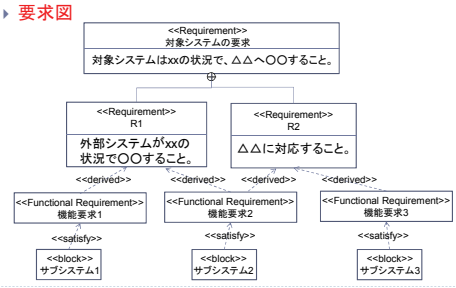
- 状態機械図
- イベントで生じる状態の遷移



22

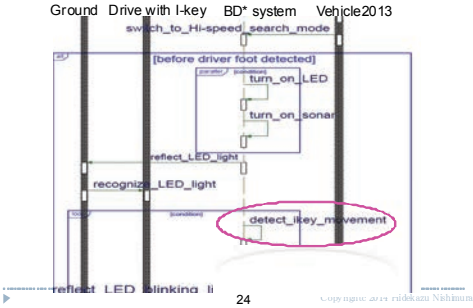
要求の詳細化とトレーサビリティ

- 要求図



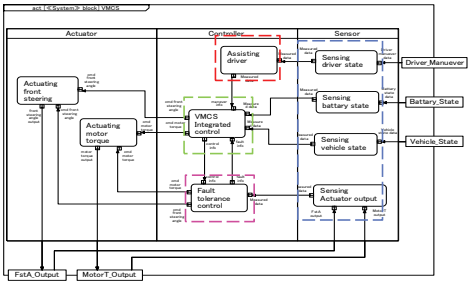
23

事例：自動車のバックドアシステム開発



24

事例：車両運動制御システム開発



25 Copyright©2014 Hidekazu Nishimura.

4．まとめ

- システムズエンジニアリング標準(ISO/IEC 15288)をもとに，そのライフサイクルプロセス，ステージを示し，ソフトウェアエンジニアリング（ISO/IEC 12207）との関係性を述べた．
- モデルで考えてシステムズエンジニアリングプロセスを進めるモデルベースシステムズエンジニアリング（MBSE）の重要性と，システムモデルを構造／振る舞い／要求／パラメトリック制約の4つの柱で表現するSysMLを紹介した．
- 企業との共同研究事例から見えてきたこと：
  - 対象とするシステムを開発する初期の段階から，システムレベルで，かつ，モデルベースで考えることが明らかに重要である．

26 Copyright©2014 Hidekazu Nishimura.

参考文献

- Systems Engineering Handbook Ver.3.2, INCOSE, 2010
- Visualizing Project Management, Third Edition
  - Kevin Forsberg, Hal Mooz, Howard Cotterman, John Wiley & Sons, Inc.
- INTERNATIONAL STANDARD, ISO/IEC 15288, IEEE Std 15288-2008, Second edition, 2008-02-01
- Adoption of ISO/IEC 15288:2002, Systems Engineering—System Life Cycle Processes
- INTERNATIONAL STANDARD, ISO/IEC 26702, IEEE Std 1220-2005, First edition, 2007-07-15
- INTERNATIONAL STANDARD, ISO/IEC/ IEEE 42010, First edition, 2011-12-01
- システムズモデリング言語 SysML（A Practical Guide to SysMLの翻訳本）
  - 西村 秀和（監訳），白坂成功，成川輝真，長谷川堯一，中島裕生，翁志強（翻訳），
  - 東京電機大学出版局，2012

27 Copyright©2014 Hidekazu Nishimura.

5.4 山本修一郎先生ご講演スライド

名古屋大学 第8回システム科学検討会

社会ニーズ把握のための技術 ー要求工学の動向ー

- 情報システムと社会ニーズ
- 要求工学概論
- エンタープライズアーキテクチャ
- システム理論

名古屋大学 情報連携統括本部 情報戦略室
教授 山本修一郎

Copyright Prof. Dr. Shuichiro Yamamoto 2014 1

名古屋大学

配布資料

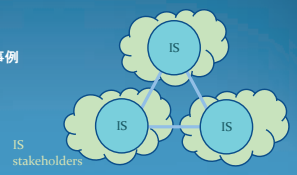
- 要求工学の基礎知識
- ゴール指向要求工学と推論
- 実践的セキュリティ要求工学
- Goal Oriented Requirements Engineering–Trends and issues
- 持続的情報連携サービス分析方法論
- アーキテクチャ論
- 高信頼性エンタープライズ・アーキテクチャの取り組み
- ディペンダビリティケース適用のフロンティア2014
- ソフトウェア再利用の潮流

Copyright Prof. Dr. Shuichiro Yamamoto 2014 2

名古屋大学

情報システムのニーズ分析

- PASEモデル
- 情報連携仮説
- 情報連携コミュニティ分析事例

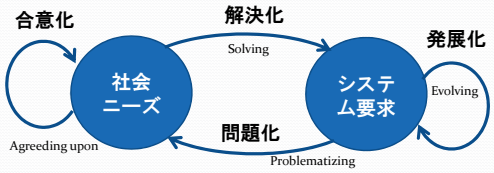


Copyright Prof. Dr. Shuichiro Yamamoto 2014 3

名古屋大学

PASEモデル

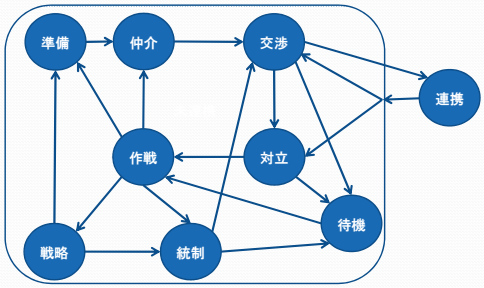
システム要求が問題化されると、社会ニーズとして、合意化される
合意化されたシステム要求が、システム構築により解決化される
システムの構築過程や社会への浸透過程で要求が成長し発展化する
この結果、また新たなシステム要求が問題化する



Copyright Prof. Dr. Shuichiro Yamamoto 2014 4

名古屋大学

情報連携の対話構造



Copyright Prof. Dr. Shuichiro Yamamoto 2014 5

名古屋大学

情報連携事例

	概要	情報連携対話プロセス
事例1	仲介者が情報システムを担当関係者との交渉で情報連携	戦略→準備→仲介→交渉→連携
事例2	組織内の別の戦略部門が管轄仲介者を通じた交渉で多数の対立点が表面化	戦略→準備→仲介→交渉→対立→作戦→(統制｜仲介｜戦略｜準備｜待機)
事例3	仲介者を通じた交渉で，情報連携に対して合意システム更改のため時期があわないこと，部門ごとに情報形式が異なるので情報連携を待機	戦略→準備→仲介→交渉→対立→(待機｜(作戦→準備))

Copyright Prof. Dr. Shuichiro Yamamoto 2014 6

名古屋大学

要求工学の基本概念

- 要求工学の動向
- IREBシラバス
- 要求開発工程
- システム・コンテクスト
- 要求の種類
- 要求の構成
- 要求変化

Copyright Prof. Dr. Shuichiro Yamamoto 2014 7

名古屋大学

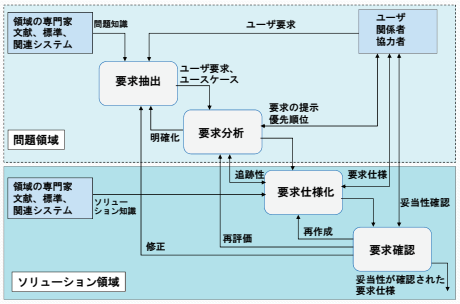
要求工学の動向

年代	要求工学の特徴	主要な成果
1970	要求分析の時代	外部環境から概念を獲得する標準的なモデル
1980	要求仕様の時代	曖昧性，矛盾，冗長性を解消する標準仕様のテンプレート
1990	要求プロセスの時代	要求プロセスの定義とプロセス改善手法
2000	高水準要求の時代	ゴール記述言語とプロセス
2010	要求整合化の時代	顧客価値創造のための要求工学研究と実証評価

参考） Davis, A., Keynote speech, RE2010

Copyright Prof. Dr. Shuichiro Yamamoto 2014 8

要求工学の基本的なプロセスモデル



Copyright Prof. Dr. Shuichiro Yamamoto 2014 9

IREB 基礎レベル 講義項目

分類	内容	時間
導入と基礎	要求の基本概念	1.25
システムとコンテクスト	①システム、システム・コンテクスト、システム境界 ②システムとコンテクスト境界の決定	1.25
要求抽出	①要求源、②要求分類、③背景モデルによる要求分類、④抽出技法	1.5
要求の文書化	①文書設計、②文書種別、③文書構造、④要求文書の利用、⑤要求文書の品質基準、⑥要求品質基準、⑦用語	2
自然言語による要求文書	①言語効果、②テンプレートによる要求構成	1
モデルによる要求の文書化	①モデル、②ゴールモデル、③ユースケース、④要求の3つの見方、⑤データ要求モデル、⑥機能要求モデル、⑦挙動要求モデル	5
要求の妥当性評価と合意形成	①要求妥当性確認の基礎、②要求合意形成の基礎、③要求の品質面、④要求妥当性確認の原則、⑤要求妥当性確認の技法、⑥要求合意形成	2.5
要求管理	①要求の属性割当、②要求ビュウ、③要求の優先順位付け、④要求追跡性、⑤要求の版管理、⑥要求変更管理	2.5
ツール支援	①ツール種別、②ツール導入、③ツール評価	1
9分類	38項目	18

参考
[1] The home of Requirements Engineering, http://www.ireb.org/
[2] IREB Certified Professinal for Requirements Engineering – Foundation Level-Version 2.1 Dec.20th 2012

Copyright Prof. Dr. Shuichiro Yamamoto 2014 10

BABOKとCPRE FLの技法比較

	BABOK	IREB
共通点	データディクショナリと用語集 データフロー図 データモデリング シナリオとユースケース コンテクスト図 状態遷移図 ベースライン化 要求の文書化テンプレート	用語集 データフロー図 実体関連図 ユースケース図 コンテクスト図 状態図 ベースライン要求 要求テンプレート
差異点	組織モデリング シーケンス図 ユーザーストーリー プロセスモデリング RACIマトリクス ステークホルダマップ カバレッジマトリクス ベンダー選定要求依頼書 ビジョンステートメント	クラス図 ゴールモデル アクティビティ図 ユースケース仕様 ステートチャート 標準文書構造(IEEE std.830) 要求文書品質基準 要求品質基準

Copyright Prof. Dr. Shuichiro Yamamoto 2014 11

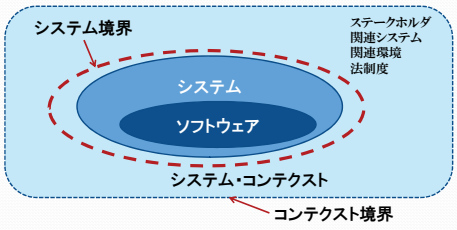
SEMAT カーネル・アルファの状態

層	アルファ	状態
カスタマ Customer	機会	識別, 必要解, 価値形成, 実現可能, 対処, 利益獲得
	ステークホルダ	認識, 表現, 参画, 合意中, 展開充足, 使用充足
ソリューション Solution	要求	着想, 限定, 首尾一貫, 受容可能, 対処, 充足
	ソフトウェア・システム	構成選択, 例証, 利用可能, 用意, 運用可能, 廃棄
エンデバー Endeavor	作業	開始, 準備, 開始, 制御中, 完結, 終了
	作業方法	原則確立, 基礎確立, 使用中, 環境整備, 有効, 廃棄
	チーム	手配, 形成, 協調, 実演, 解散

SEMAT : Software Engineering Method and Theory
アルファ(alpha, Aspiration Led Progress and Health Attribute)

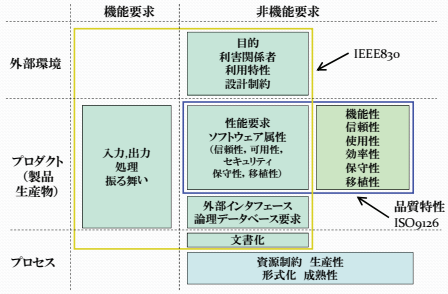
Copyright Prof. Dr. Shuichiro Yamamoto 2014 12

システム・コンテクスト、システム境界、コンテクスト境界とソフトウェア・システム



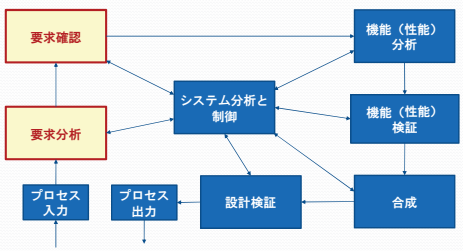
Copyright Prof. Dr. Shuichiro Yamamoto 2014 13

機能要求と非機能要求



Copyright Prof. Dr. Shuichiro Yamamoto 2014 14

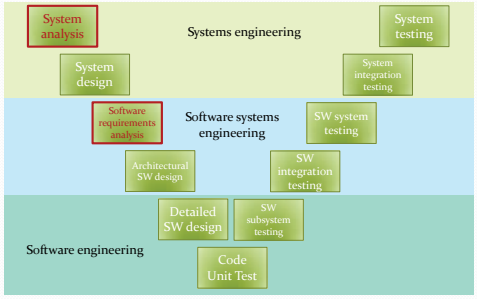
IEEE 1220 systems engineering process



参考) IEEE 1220 systems engineering process

Copyright Prof. Dr. Shuichiro Yamamoto 2014 15

IEEE software systems engineering process

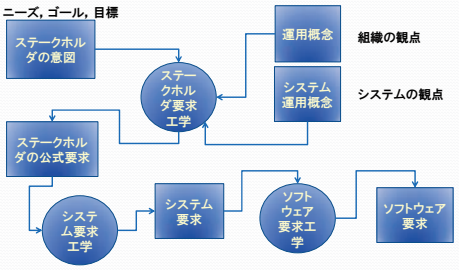


Copyright Prof. Dr. Shuichiro Yamamoto 2014 16

名古屋大学

ISO/IEC/IEEE 29148:2011

ニーズ，ゴール，目標



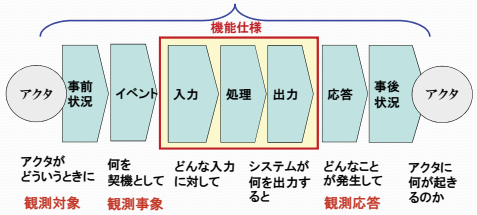
参考) ISO/IEC/IEEE 29148:2011, Systems and software engineering —Life cycle processes — Requirements engineering, 5.2.3

 17

名古屋大学

要求記述の構成要素

曖昧性：範囲，内容，関係の多義性・不明性



アクタがどういうときに 観測対象 | 何を契機として 観測事象 | どんな入力に対して | システムが何を出力すると | どんなことが発生して 観測応答 | アクタに何が起きるのか

参考)要求の曖昧さ，http://www.bcm.co.jp/site/youkyu/youkyu38.html

 18

名古屋大学

要求管理の必要性

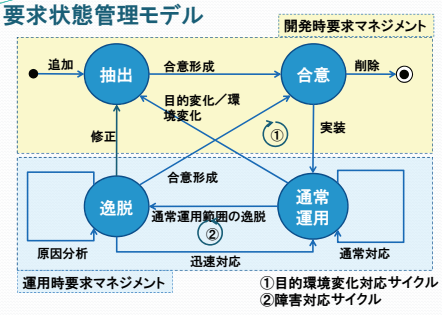
- 要求が静的であることはまれである
- 開発管理の観点から，要求を確定することが望ましいけれども,それが可能になるのはまれである.

参考) ISO/IEC/IEEE 29148:2011, Systems and software engineering —Life cycle processes — Requirements engineering, 6.5

 19

名古屋大学

要求状態管理モデル

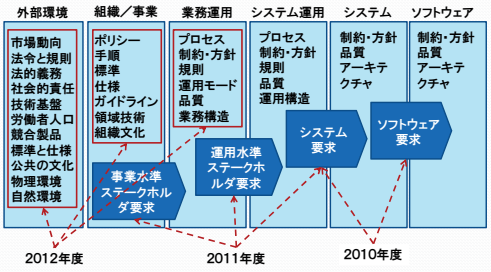


①目的環境変化対応サイクル
②障害対応サイクル

 20

名古屋大学

IPA要求発展型WGのスコープの発展



参考)ISO/IEC/IEEE29148

 21

名古屋大学

要求モデルの具体例

(1)課題分析表
(2)ゴールモデル
(3)BMC
(4)BSC(戦略マップ)
(5)BPMN
(6)ユースケース
(7)実体関連図
(8)クラス図
(9)データフロー図
(10)IDEF
(11)アクティビティ図
(12)状態図
(13)シーケンス図
(14)i*フレームワーク

 22

名古屋大学

要求モデル手法

側面	手法
ビジネス側面	■課題分析表 ■ゴールモデル(要求図) ■i*フレームワーク ■BMC ■BSC（戦略マップ） ■BPMN
機能側面	■データフロー図 ■IDEF ■ユースケース
データ側面	■実体関連図 ■クラス図
振舞側面	■アクティビティ図 ■状態図 ■シーケンス図

 23

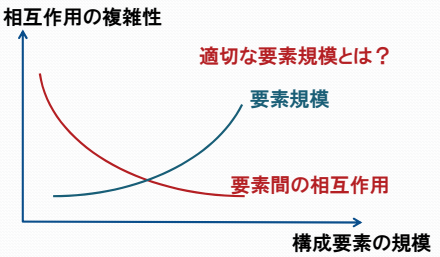
名古屋大学

高信頼性エンタープライズアーキテクチャ

- Software Architecture
- Twin Peaks Model
- ACE
- MEFSA
- Enterprise Architecture
- TOGAF
- ArchiMate
- O-DA

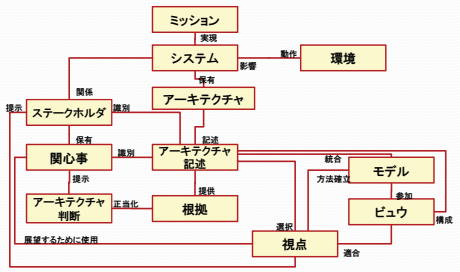
 24

要素の規模と相互作用の複雑性



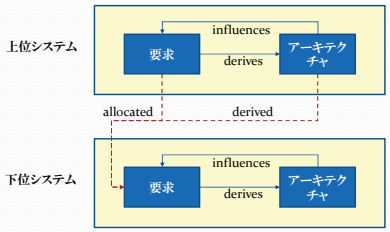
Copyright Prof. Dr. Shuichiro Yamamoto 2014 25

アーキテクチャメタモデル



Copyright Prof. Dr. Shuichiro Yamamoto 2014 26

要求とアーキテクチャ



参考) D. Firesmith et.al., The Method Framework for Engineering System Architectures, CRC Press, 2009

Copyright Prof. Dr. Shuichiro Yamamoto 2014 27

ArchiMateの構成

分類	ADMフェーズ	
ArchiMate コア	B C D	ビジネス・アーキテクチャ 情報システム・アーキテクチャ 技術アーキテクチャ
動機づけ拡張	– A H –	準備 アーキテクチャ・ビジョン アーキテクチャ変更管理 要求管理
実装・移行拡張	E F G	機会とソリューション 移行計画 実装・ガバナンス

ArchiMate 2.0、アーキテクチャモデリング言語、オープングループ、2012年1月

Copyright Prof. Dr. Shuichiro Yamamoto 2014 30

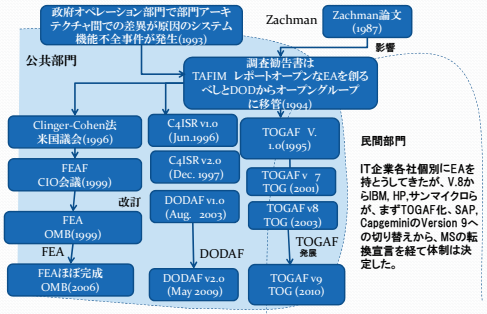
TOGAFアーキテクチャ開発工程

工程	説明
準備	TOGAFアーキテクチャプロジェクトの準備活動
A.アーキテクチャビジョン	スコープ，制約，期待，ステークホルダを定義，事業環境を確認
B.ビジネスアーキテクチャ	ビジネスの現行と目標アーキテクチャを定義，差異を分析
C.情報システムアーキテクチャ	情報システムの現行と目標アーキテクチャを定義，差異を分析
D.技術アーキテクチャ	技術の現行と目標アーキテクチャを定義，差異を分析
E.ソリューション	実施計画，展開手段，要素を定義し移行アーキテクチャを構築
F.移行計画	費用対効果分析，リスク分析に基づき移行実施計画を詳細化
G.実装監督	アーキテクチャ移行計画を管理，実装結果を確認
H.アーキテクチャ変更管理	アーキテクチャの事業目標適合性を継続的監視，変更管理
要求管理	全工程で要求確認を実施

参考) TOGAF V.9 A Pocket Guide, THE Open GROUP, 2008

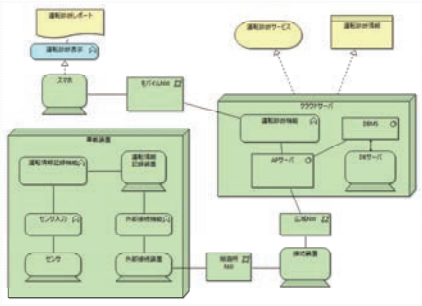
Copyright Prof. Dr. Shuichiro Yamamoto 2014 29

主なEAの歴史と関係



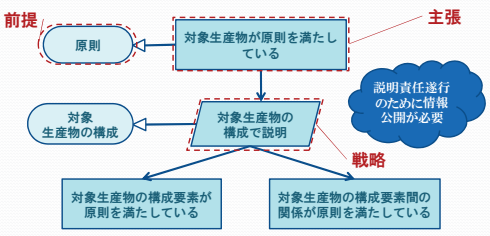
Copyright Prof. Dr. Shuichiro Yamamoto 2014 28

運転診断サービスの技術層の例



Copyright Prof. Dr. Shuichiro Yamamoto 2014 31

第三者への説明責任の遂行



Copyright Prof. Dr. Shuichiro Yamamoto 2014 32

高信頼ADMの概要

工程	AADM
準備	①アーキテクチャリポジトリでの証拠文書と保証ケースの格納 ②ディペンダビリティ委員会での主張間の優先順位の合意
A.アーキテクチャビジョン	①ディペンダビリティスコープ定義 ②主張の定量評価尺度定義 ③保証ケース能力評価 ④ディペンダビリティパラメタ定義
B.ビジネスアーキテクチャ	①ディペンダビリティ原則定義 ②BA保証ケース作成 ③BA保証ケースレビュ
C.情報システムアーキテクチャ	①IA保証ケース作成 ②IA保証ケースレビュ
D.技術アーキテクチャ	①TA保証ケース作成 ②TA保証ケースレビュ
E.ソリューション	①BA, IA, TA保証ケースを統合 ②一貫性を確認
F.移行計画	①運用管理の保証ケース作成 ②ディペンダビリティパラメタ価値分析
G.実装監督	①保証ケースの証拠を作成 ②プロセス保証ケースの証拠を作成 ③網羅的に主張と証拠の関係を確認 ④運用に対する保証ケースをレビュ
H.アーキテクチャ変更管理	①運用保証ケースの証拠を管理 ②主張の不成立への対応策の確認③保証ケースによるリスク管理 ④保証ケースによる障害分析
要求管理	保証ケースの主張と要求の追跡管理

Copyright Prof. Dr. Shuichiro Yamamoto 2014 33

ArchiMateのためのD-Case方法論

ArchiMate		D-DA/ D-Case	ArchiMate /D-Case パターン
動機拡張	ゴールモデル	ゴールD-Case	
	要求モデル	要求D-Case	
ビジネス層	ビジネスプロセス	BP CAE D-Case	
AP層	データモデル、APモデル	データ・AP D-Case	
技術層	配置モデル	技術層 D-Case	
実装移行拡張	作業モデル、成果物モデル	作業・成果 D-Case	

ArchiMateモデルに対して開発中

Copyright Prof. Dr. Shuichiro Yamamoto 2014 34

システム論

- SSM
- BSSM
- VSM
- ESE

Copyright Prof. Dr. Shuichiro Yamamoto 2014 35

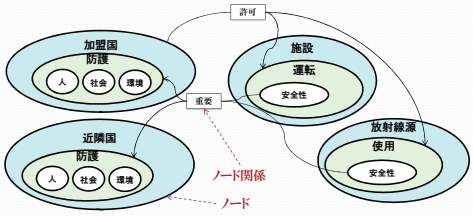
BSSMとSSM

段階	視点	SSM	BSSM
1	状況の抽出	問題状況を分析	同左
2	状況の表現	■適切な選択を可能とするように状況を表現 ■構造とプロセスの相互関係「状況の風潮」	同左
3	基本定義	■特定の視点「世界観」によるシステムの命名 ■問題状況を改善する仮説群	構造化文章により システム要素を定義
4	概念モデル	■入力ー変換ー出力 ■構成要素＝動詞 ■形式化できるシステムとそれを取り巻く環境 ■階層化 ■終了条件	システムミグラムによって入力を出力に変換するシステムの行為を記述する
5	比較	■概念モデルによる問題状況の評価 ■反復的な評価	系統的な質問 概念モデルと現実を比較
6	改革案	■構造改革■プロセス改革■行動改革	同左
7	改革の実行	改革による問題状況が解消されることを監視 新たな問題が発生する場合，段階1に戻る	同左

Copyright Prof. Dr. Shuichiro Yamamoto 2014 36

システミグラムによる安全基準の記述例

施設の運転及び放射線源の使用における安全はその運転及び使用を許可した加盟国における、また、その他の加盟国、特に近隣国における人（個人及び集団）、社会及び環境の防護にとって非常に重要である。（出典）IAEA 原子力安全基準、導入、背景



Copyright Prof. Dr. Shuichiro Yamamoto 2014 37

VSM：Viable System Model

システム	役割	説明
S1	インプリメンテーション	対象システムの基本活動 環境とのインタフェースを持つ
S2	コーディネーション	S1及びシステム間を調整するための情報チャネル
S3	コントロール	他のS1を制御するだけでなく、S2を監督 システムの資源配置を維持 責任体制の構築と制御
S4	インテリジェンス	S4は外部環境の変化に対してS3と相互作用することによりインテリジェントに適応 戦略的意思決定と危機管理
S5	ポリシー	S5は様々な部分のバランスをとり、組織全体方針の決定と舵取りにより，現在と未来をバランス

参考)Stafford Beer, Viable System Model

Copyright Prof. Dr. Shuichiro Yamamoto 2014 38

システムの階層

階層	説明
Enterprise	Anything that consists of people, processes, technology, systems, and other resources across organizations and locations interacting with each other and their environment to achieve a common mission or goal.
System of Systems	A set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities. (DoD)
System	A set of interrelated components working together toward some common objective

参考) A. Kossiakoff, W. Sweet, S. Seymour, S. Biemer, Systems Engineering Principles and Practice, second edition, Wiley, 2011

Copyright Prof. Dr. Shuichiro Yamamoto 2014 39

まとめ

- 要求工学の対象は、ソフトウェアだけでなく、システムやサービス、ビジネスにまで拡大している
- 要求工学は、多様な分野の実践的な手法を取り込んで発展している
- 変化を前提にした要求工学が必要である
- 要求工学は、多様なシステム開発技法の中核である
- システム要求プラクティスの統合が必要である

Copyright Prof. Dr. Shuichiro Yamamoto 2014 40

5.5 木下佳樹先生ご講演スライド

妥当性確認とアシュランスケース

〜第8回システム科学検討会〜

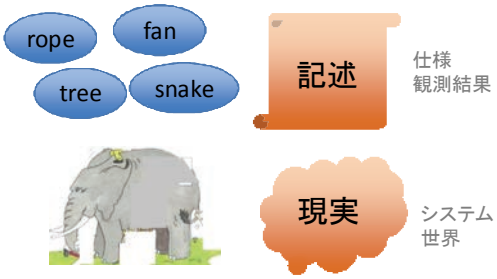
2014-06-23

神奈川大学理学部情報科学科

木下佳樹

群盲、象を撫でる

記述 vs 現実



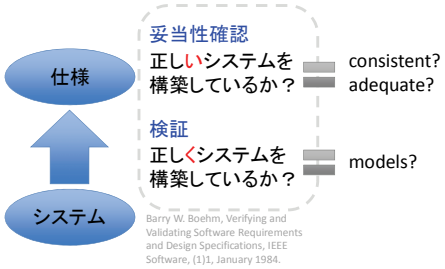
世界: 現実 vs 記述

- 「システム自身」(現実)とその「記述」は別！
  - システムは 機械、ソフトウェア、社会、etc。
  - 記述は 言葉、文書。
- いろいろな現れ方
  - 実装 vs 仕様
  - 認証対象 vs 提出文書
  - モデル vs 形式理論 (math. logic)
- 現実と記述のずれ？
  - 記述は現実において正しいか？
  - 記述は現実をすべて記しているか？
    不完全な記述は多数並び立ち得る。(群盲の記述. . .)

主張: 導出可能 vs 正当

- 導出可能 (derivable):
  - **記述**のつじつまが合っている。
  - 公理から推論して導出が可能である。
- 正当 (valid):
  - **現実**において成り立っている、正しい。

検証 vs 妥当性確認



アシュランスケース (AC)

システムが要求を満たすことを
一連の主張、証拠、それらを結ぶ**明示的議論**によって
理由付き・監査可能な形で示す**文書**一式を
アシュランスケースという。

- V&V過程の出力
- 歴史的にはsafety caseが起源。
  欧州安全規制行政が90年代にprescriptiveな安全管理からgoal-basedなものにシフトするに従って重要になった。
- 規制や標準において義務化。石油リグ、原発、防衛、航空、鉄道、自動車、医療機器、その他の業界。
- Assurance caseは safety case, reliability case, security case, dependability case などの総称。

アシュランスケースの問題点

ACは、複雑かつ膨大になりがち。

なのに、たった一句の誤りが全体を無意味にしてしまうかもしれない。

プログラムと似ている。

アシュランスケースを、

隅々まで正確に構築、

整合性検査(レビュー)すること

が難しい。

アシュランスケースの問題点

ACは、複雑かつ膨大になりがち。

なのに、たった一句の誤りが全体を無意味にしてしまうかもしれない。

プログラムと似ている。

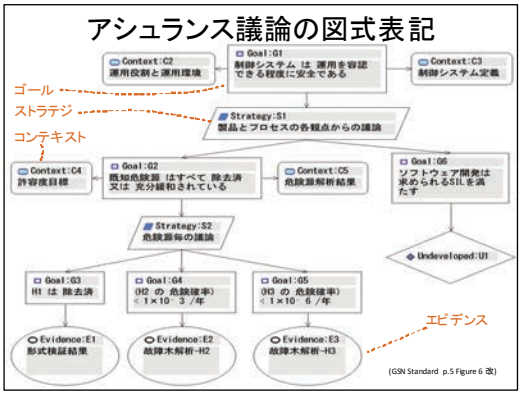
アシュランスケースを、

隅々まで正確に構築、

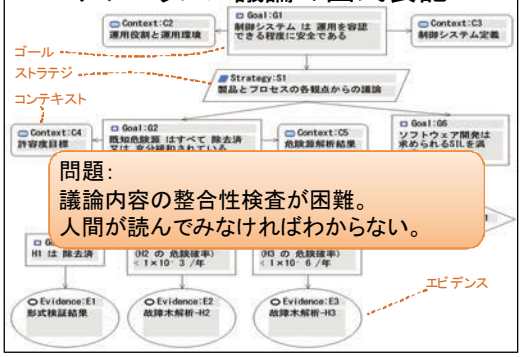
整合性検査(レビュー)すること

が難しい。

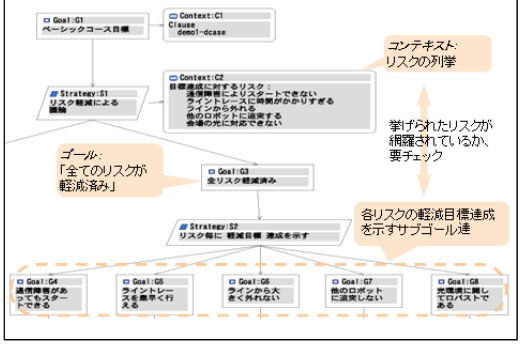
アシュランス議論の図式表記



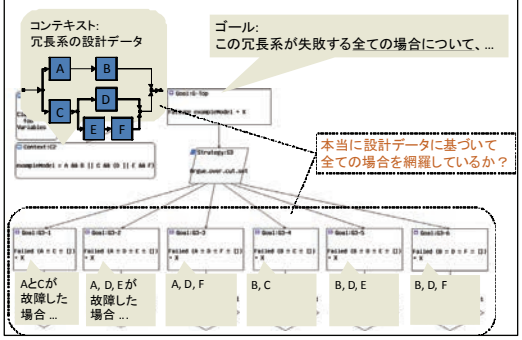
アシュランス議論の図式表記



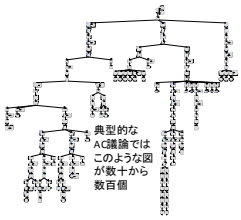
単純な整合性検査



専門的判断を要しない検査



課題



- 専門的判断を要しない検査も人力レビュー頼り。
- 不整合は局所的とは限らない
  - 遠く離れた枝間
  - 何百ファイルにも分かれた議論間
  - 議論と外部参照データとの間
- 各議論、データは多数の手で刻々変更される

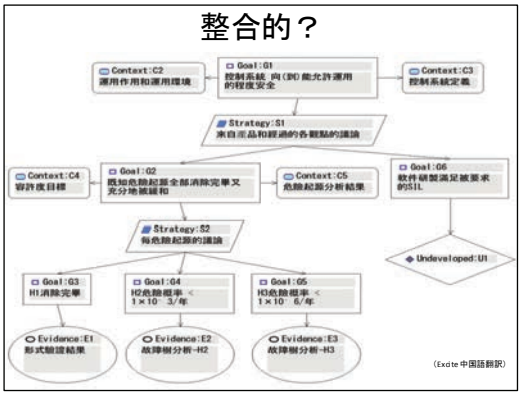
- 機械に検査できることは機械で検査
- レビュワーは専門的判断を下すことに集中できるように

アプローチ: 形式アシュランスケース

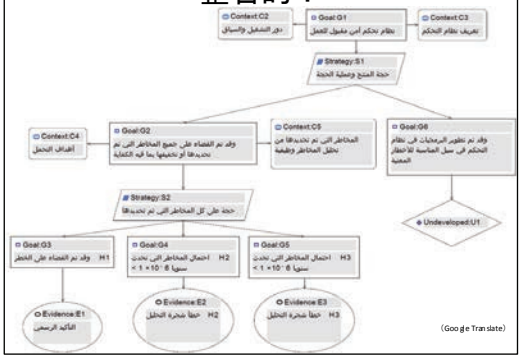
- 人力レビューでは、人間がゴール記述他を読んで、
  - *システムや環境を構成する「もの」*
  - *システムに要求される性質、環境の制約、仮定などの考慮すべき「こと」*

  にどんなもの・ことがあるか(**オントロジー**)を**文法・言葉の意味から理解**した上で、議論整合性を判断。
- **形式AC ＝ 機械にも解るオントロジーの定義 & それに基づく議論**
- 議論の整合性検査を「議論がきちんと定義に基づいているかどうか」の機械的検査に帰着

整合的？



整合的？



アプローチ: 形式アシュランスケース

- 人力レビューでは、人間がゴール記述他を読んで、
  - *システムや環境を構成する「もの」*
  - *システムに要求される性質、環境の制約、仮定などの考慮すべき「こと」*

  にどんなもの・ことがあるか(**オントロジー**)を**文法・言葉の意味から理解**した上で、議論整合性を判断。
- **形式AC ＝ 機械にも解るオントロジーの定義 & それに基づく議論**
- 議論の整合性検査を
  「この議論は定義に正しく基づいているか？」の
  機械的検査に帰着　（定義は適切か？とは別）

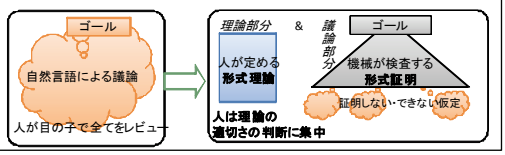
形式アシュランスケース

- **形式アシュランスケース**
  ＝機械にも解るオントロジーの定義 & それに基づく議論
  ＝形式 理論の定義 & その理論の中での形式証明

形式論理での 語彙定義, 公理の宣言（理論部分）

語彙, 公理からの推論 の正しい組み合わせ（議論部分）

- 議論の整合性検査をすべて
  「これは形式証明になっているか？」の機械的検査に帰着。



おわりに

- システム自身とその記述の区別
- 妥当性確認と検証 (V&V)
- アシュランスケース(AC)： V&V過程の出力
- AC検査への二つのアプローチ
  - 図式表現
  - 形式アシュランスケース

付録

付録１．システム科学検討会の設置趣旨及び実施方法

１．設置の趣旨

「システム構築は難しい」それゆえに非効率で非経済的な活動が、社会の中で科学技術や経済界の発展の足かせになっている。システム科学ユニットでは、この難しさを解明し、システム構築の方法論を示すことが一つの課題となっている。

独立行政法人情報処理推進機構（以下、IPA と記載）の技術本部ソフトウェア高信頼化センター（以下、SEC と記載）では、IT 分野における安全性や信頼性の向上が重要命題の一つとなっており、産業界のさまざまな事例を収集し、ソフトウェア開発のガイドラインの作成や品質評価指標などをまとめ上げてきている。この活動の中でソフトウェア開発技術の課題や難しさが蓄積されている。

ソフトウェア開発はシステム構築の一つの例であり、その難しさを学ぶことを通して、システム技術の研究課題を探ることができ、システム構築の方法論を探索することができると考える。また、ソフトウェア開発における方法の標準化の中でモデリング技術や可視化技術など、アカデミアに求められる研究要素が抽出できる可能性もある。

本検討会では、IPA-SEC にご協力いただき、JST-CRDS とお互いの持つ研究情報を共有し、ソフトウェア開発技術や社会の中のソフトウェアをそれぞれの課題解決に向けて議論を行い、新たな課題の発掘を行うことを目的とする。

２．実施方法：

IT ソフトウェア技術、社会システム科学、システム科学分野に精通した有識者に参加いただき、ミニワークショップ形式でクローズドな議論を行う。IT ソフトウェア技術やソフトウェア構築などの有識者は、IPA-SEC にご紹介いただく。必要に応じて、講師を招聘する。

事務局は、JST-CRDS システム科学ユニットが行う。約 10 名のメンバーを招聘。開催場所は JST-CRDS 内の会議室を予定。

最終開催はワークショップを開催し、その報告書をまとめる。

付録２．システム科学検討会委員及びオブザーバー覧

氏名	所属・役職	備考
鈴木久敏	JST-CRDS システム科学ユニット 特任フェロー(～2014.3) JST-CRDS システム科学ユニット フェロー(2014.4～)	主査/事務局
大島啓二	日本科学技術連盟 嘱託	委員
倉橋節也	筑波大学大学院ビジネス・サイエンス系 准教授	委員
田名部元成	横浜国立大学大学院国際社会科学研究院 教授	委員
寺野隆雄	東京工業大学総合理工学研究科 教授 JST-CRDS システム科学ユニット フェロー(～2014.3) JST-CRDS システム科学ユニット 特任フェロー(2014.4～)	委員
白坂成功	慶應義塾大学大学院システムデザイン・マネジメント研究科 准教授	委員
松本隆明	情報処理推進機構技術本部ソフトウェア高信頼化センター 所長	委員
木村英紀	JST-CRDS システム科学ユニット 上席フェロー	委員
久保忠伴	情報処理推進機構技術本部ソフトウェア高信頼化センター 調査役	オブザーバ
豊内順一	JST-CRDS システム科学ユニット フェロー(～2014.3) JST-CRDS システム科学ユニット 特任フェロー(2014.4～)	オブザーバ
藤井新一郎	JST-CRDS システム科学ユニット フェロー	オブザーバ
シン・ジャワ	JST-CRDS システム科学ユニット フェロー	オブザーバ
舩橋誠壽	JST-CRDS システム科学ユニット 特任フェロー	オブザーバ
富川弓子	JST-CRDS システム科学ユニット フェロー	事務局
小野里実	JST-CRDS システム科学ユニット フェロー(2014.5～6)	事務局

付録３．システム科学検討会開催記録

回数	年月日 時　間	内容	参加 人数
第 1 回	2013 年 11 月 25 日 13:00-15:40	報告１：ソフトウェアシステム構築の困難性［松本］ 報告２：システム構築型イノベーションの重要性とその実現に向けて［木村］	12 名
第 2 回	2013 年 12 月 27 日 15:00-17:15	報告：システムズエンジニアリングの最新動向と慶應 SDM の取組み［白坂］	12 名
第 3 回	2014 年 1 月 23 日 15:00-18:00	報告１：大規模開発を成功に導くアプローチ ―あるシステムの開発―［大島］ 報告２：システム構築俯瞰図の考え方［鈴木］	11 名
第 4 回	2014 年 2 月 19 日 15:00-17:00	講演：非機能要求グレード　～システム基盤に対する非機能要求の合意形成～［情報処理推進機構技術本部ソフトウェア高信頼化センター グループリーダー 山下博之］ 報告：システム科学技術のあゆみ［鈴木］	13 名
第 5 回	2014 年 3 月 26 日 13:00-17:00	報告：情報システムにおけるデザイン科学研究［田名部］ ミニワークショップ：システム関連キーワードの抽出とマッピング［鈴木］	12 名
第 6 回	2014 年 4 月 25 日 15:00-17:30	講演：ソフトウェアと競争戦略：競争戦略論の視点から［筑波大学ビジネス・サイエンス系准教授 立本博文］ 報告：システム関連キーワードの抽出とマッピングの作業結果［鈴木］	13 名
第 7 回	2014 年 5 月 9 日 15:00-17:35	講演：SEC を通した技術移転　―形式手法の場合―［情報処理推進機構技術本部ソフトウェア高信頼化センター連携委員 新谷勝利］ 報告：システム構築方法論俯瞰図について［鈴木］	12 名
第 8 回	2014 年 6 月 23 日 13:00-17:45	ワークショップ：システム構築方法論 ―現在の到達点と今後の課題―	20 名

付録４．第１回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 25 年 11 月 25 日（月）　13:00～15:40

場所：JST 東京本部別館　4 階会議室 A

［プログラム］

１）はじめに

・配布資料確認

・開会挨拶

・自己紹介

・JST-CRDS の紹介と戦略スコープについて

・趣意説明

２）「システム構築学」の創成に向けて

２－１：システム構築の困難性

・ソフトウェアシステム構築の困難性

松本隆明（独）情報処理推進機構ソフトウェア高信頼化センター所長

・システム構築型イノベーションの重要性とその実現に向けて

木村英紀（独）科学技術振興機構研究開発戦略センター上席フェロー

２－２：ディスカッション A

・「ソフトウェアシステム構築の困難性」と「社会システム構築の困難性」との間で類似性、異質性を整理した上で、社会システム構築する上で解決すべき課題を抽出する。その上で、学術的解決策を探求できそうな領域を炙り出す。

２－３：ディスカッション B

・人材、国内外の現状について等

３）おわりに

・今後の流れと閉会挨拶

２．資料

別紙１　ソフトウェア開発の難しさについて

別紙２　プログレスレポート　システム構築型イノベーションの重要性とその実現に向けて

３．議事メモ

・開会挨拶概要（木村上席）

我々JST-CRDS システム科学ユニットでは、システム構築の重要性を主張している。また、システム構築に難しさがあり、それゆえに付加価値があると見ている。システム構築の先輩であるソフトウェアの構築の難しさと長年対峙し克服する道を模索している IPA-SEC さんと協力して、省庁の枠を取り払って検討しようということになった。皆様、是非、ご知見を集約して実りある会にしていただきたい。

・趣旨説明概要（鈴木主査）

質疑、議論

○活動方針のキーワードは、「ビジネス」と「教育」と受け止めた。システム科学は日本の競争力に寄与するという前提でその価値を考えるべき。日本の中に閉じていたものを形式化し、論理的根拠を与えることによって共有財産になる。

→（主査）初等教育からこういった考え方を身に付けさせるべきであると思うが、そのための教科書もない。この会で教科書を作ろうとまで言っているのではなく、盛り込むべき内容を検討できれば良いと思っている。また、企業でも使えるものであるべき。今は経験で行使されている方法論について、知恵の出しあいの場としたい。

○議論の広さを確認したい。ソフトウェアにこだわらずシステムを対象にするのか。

→（主査）ソフトだけでなくハード的なものを含んで広くとらえる。もっというと情報だけを切り出して検討するつもりはない。

○ハードウェアに比べて確かにソフトウェアの構築は認識されにくい部分がある。またソフトはそれだけを単独で向上させれば良いというものではない。

○現行私が研究している内容とかなりかぶると思った。先行しているソフト構築の類似性や異質性を見ていきたい。

・ソフトウェアシステム構築の困難性（松本氏）

質疑、議論

○開発の工期のグラフで、まったく駄目になってしまったものはどこに含まれるのか？

→（松本氏）このグラフのなかには入っていない。失敗して完了しなかったものもあるが、数値の把握はしていない。

○米国、英国の実務家団体を中心に活動している INCOSE（The International Council on Systems Engineering）からシステムズエンジニアリングハンドブックが出ていて、その中で、システムの初期計画にお金をかけると良いものができるが、そこにかけられる実数は 10～15%という数字が出ている。日本では、分割されていてそういった数字も出せない。

○日本にそういう組織が欲しい。

○「単なる冗長化による信頼度向上策はとれない」とあるが、バックアップではないものとは？

→（松本氏）ソフトの場合には、バックアップ系でも同じソフトを使っていれば同じように不具合が発生するため単なる冗長化では高信頼化にはつながらない。研究レベルでは、自己適応型システムという、システム自身が環境や状況をモニタリングして自身も変化するといったシステムもある。

○自動車の中には、状況に応じたシステム制御が組み込まれている。

○宇宙の分野では、冗長が主流であるが、グレースフルデグラデーションといって、機能を落としても死なないシステムを構成することもある。衛星などは 15 年間オペレ

ーションでメンテナンスができないために必要としている。ヨーロッパでは複雑にシステムを組み合わせる傾向がある。自動車も自動運転に向けて組み込みは進化しているところであろうと思う。

○化学プラントの世界でもヨーロッパを中心にシステム化が進んでいる。企業安全性規格の整備ができている。

○「こうのとり」では、ミッションが 1 故障耐性、安全性が 2 故障耐性を持っている。

○システムにどの程度の冗長性をもたせるのが良いかは、価値をどこに置くかによって変わる。鉄道の運行管理システムにおける計算機構成は、信頼性の面で 2 重系、3 重系になっている。また、駅単独の層、安定運行の層、全体の層と 3 階層で構成することによって、縮退運転を可能としている。安全を最優先として、おかしくなったらまず止めるという「フェールセーフ」機能も取り入れている。

→(松本氏)ディペンダビリティの確保が重要だと思う。自動車のような場合には、故障が発生しても単純に止めるのではかえって危険なケースもあり、最低限の機能でも走り続けるというディベンダビリティの確保が求められる。

○一つは、ソフトから得られる情報を組織としてどう生かすかだと思う。社会システムの中では、要素技術をどう組み合わせるかが重要で目標とデザインが重要になる。それぞれの制度をどう共有するかがポイント。ここの欠如が、システムが機能を生かし切れない理由だと思う。二つ目は冗長性について、実は切り替え時の障害が最も多いことが分かっている。オペレーションミス。人間のノウハウの伝達技術がここでは大きなポイントとなる。

○次回はこういったことを踏まえたモデリングの話を白坂先生にお願いしたい。

○ビジネスの世界では、丁寧に設計する方向と逆の方向に進んでいる。「まずは造ってしまえ」の発想。要求定義されていないものをコントローラブルにする方法やアジャイルといった手法がとられる傾向がある。

○開発プロセスもウォータフォール～アジャイルといろいろあるが、システムの特性も考えてどの方法を採用するのが良いのかを見極める必要がある。

○オープンデータ、ビックデータの広がりの中でそのデータの信頼性と計算処理の信頼性が重要になってくる。

○マッシュアップ、アジャイル、ブラッシュアップなど、個人的には問題だと思いつつダメだとは言い切れない。

・システム構築型イノベーションの重要性とその実現に向けて（木村氏）

質疑、議論

○2009 年に SoS（System of Systems）の考え方が発生し、2010 年から 2011 年にエンタープライズシステムズエンジニアリングスが出て来た。当初は、システムの考慮範囲が狭かった。MIT でもエンジニアリングシステムズという言い方で、単なる技術システムだけでなく、それに関連する外部環境や法律も合わせてデザインすることが必要と言い出している。難しいがやらなくてはならないところであり、先端のところで

ある。

○システムは日々成長していくものである。また、日常生活で実際に多数の利用者に使用されてもいる。社会という「動いている＋使われている」システムを前提に、新たな仕組みを構築することに立ち向かわなくてはならない。ゴールを見据えて、影響を最小限にしながら改修や拡張を行っていくことになる。その意味ではアーキテクチャとしての「自律分散」も鍵となる仕組みの一つである。

○年金制度などはその例だと思う。

→（木村氏）制御の世界では、古い制御で動いている工場の中で、パラランといって少しずつ動かしながら革新して最終的に新しい制御に組み替えることはやってきている。不可能ではないと思う。ただ、社会には、ある程度政治力も必要であることは否めない。

○情報システムでもやってきている。

○プラットフォームのビジネスモデルとしては、Apple Store と Google が挙げられる。彼らはプラットフォームを作り広く普及させることでその世界の標準化を握った。国が指導するシステムも標準化できれば良いが、インターネットのような広がりのある世界で、民主化、市民化は上からのコントロールができない状況。プラットフォームを作った人の勝ちという世界。その中でも EMS（Energy Management System）などは国が標準化できる余地のある分野だと思う。

→（木村氏）国主導でやらなくてはならないものも多い。エネルギーだけではなく、農業、食糧といった分野。

○日本の民族性や教育制度などを考えると、Apple や Google のようなものが日本で生まれるかについては懐疑的だ。逆に日本ならではの「勝てるシステム」を模索したい。

→（木村氏）日本の要素技術のレベルの高さの上にシステムを構築することがそうであろうと思う。

○うまく融合できるかがポイントとなる。

○失敗ができるプラットフォームが必要だと思う。

○日本の教育は失敗するなというもの。得られた知識を使っていろいろ試そうという素地がない。

○プラットフォームやガイドラインを国で作成しようとすると、業界から大反対がある。それらは、マーケットが決めるという言い分。かつ、品質は第三者に委ねられている状況。

○エコシステムを考慮することだと思うが、利用者側、参画者側、双方にメリットがないといけない。

○多様化する中で失敗しながらデザインする。これらの教育が必須。間違っていない方向。日本もそういった教育が必要。

○早く追い付こうという目標では追い付くことしかできない。高度成長時代はそうだった。日本が追い越せることがあるのだとしたら、日本の特性に根ざした標準を作ることだと思う。

○システムやソフトウェアの品質そのものを評価するプロダクト標準ではなく、開発のプロセスを評価するプロセス標準ができてきた。しかし、プロセス標準はプロセスさえ守れば良く、最終成果物の品質に差が出てこない。最終成果物の品質そのものを評価するプロダクト標準のほうが、最終成果物の品質で勝っている日本の強みを生かせる。
○海外から見ると、日本は「得体のしれない強さ、集合知」があると見られている。
○アメリカなどは、これを目指してデザイン思考を取り入れたといわれている。
○この「得体のしれない強さ」を世界に説明できるようにすることが重要であると思う。
○94 年以降に出てきたシステムズエンジニアリング標準は、日本の強みの分析が活かされている。
○強み、弱みを見極める必要がある。
→（木村氏）科学技術の中身が違ってきているのではないかと思っている。その変化に日本が付いていけなかった。グループはあるが、チームはない。日本の強みである品質への拘りがある。
○要求水準より高いレベルを追求する傾向がある。
→（木村氏）そこが、波に乗り切れなかったポイントではないかと考えている。
○国内で生まれたものを国内で高く評価されない傾向がある。それぞれ使えるものであるのに、使うのは海外企業であったりする。
○日本の中だけで戦っていた「井戸の中の蛙」を「世界という土俵」に引っ張りださなくてはならない。
○90 年代はハード中心であったが、現在はソフトが 90%のウェイトを占めるようになった。しかもプログラム作成は市民化され、できもしない人がソフトを作るようになった。「できる人にやらせる」から「普通の人が普通に作れるもの」になり、そのプラットフォームが整備されている。いまやソフトは中国やベトナム製。楽天や Yahoo は「リクルート」。アジャイルでどんどん市場に出す傾向。
○アジャイルなどではできたものをお客さんに叩いてもらう。マイクロソフトのやり方がまさにそう。「検証」は市場に出して、クレームがきたら直す。
○すべてを同じ方法論で開発してしまうことに問題がある。システムの特徴にあわせて、ここはアジャイルでやる、そこは違う方法という感じで方法論を選べれば良い。
○「複合的」というのも一つのキーワードだと思うが、異分野同士を扱う場合、それぞれ違った言葉を使う。そこの共通化は必要。
○文理融合の観点は重要。ポリシーメーキング／制度など、それぞれを考慮してものをデザイニングできるように共通化する必要がある。このプラットフォームは重要。
○異なった世界の人が理解し合える、異なるシステムが融合できると言う意味での「共通言語基盤」が必要。
○言語だけでなく、扱う単位も違っていたりする。工数を比較する際にも、片やキロで片やコスト（円）であったりする。

以　上

付録５．第２回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 25 年 12 月 27 日（金）　15:00～17:15

場所：JST 東京本部別館　2 階会議室 E

［プログラム］

１）はじめに

・配布資料確認、前回議事録確認

２）「システム構築学」の創成に向けて

２－１：システム構築の困難性

・システムズエンジニアリングの最新動向と慶應 SDM の取組み

白坂成功　慶應義塾大学システムデザイン・マネジメント研究科准教授

２－２：ディスカッション

・人材、国内外の現状について等

３）おわりに

・今後についてと次回の確認

２．資料

・前回議事録

・資料１　システムズエンジニアリング最新動向と慶應 SDM の取組み

３．議事メモ

・システムズエンジニアリングの最新動向と慶應 SDM の取組み（白坂氏）

質疑、議論

○白坂さんの発表で、アーキテクトとエンジニアはどのように区別していますか？

→（白坂氏）システムエンジニアのアクティビティの一部がアーキテクトのアクティビティと位置付けている。(われわれのところでは) アーキテクトだけを育てているわけではない。ライフサイクル全体を教えている。その中には、インテグレーション、バリデーション等、種々の活動がある。世の中でアーキテクトと呼ばれている肩書きがあるので､アーキテクトについて講義･講演の依頼を受ければ、そこだけ切り出して話すことはできるが、はっきり定義するのは難しい。

○私は、個別の解を考えるのがエンジニアリングであり、メタレベルの普遍解を見いだすのがアーキテクティングと考えていますが、いかがでしょうか？

→（白坂氏） おっしゃることは分かるが、我々が扱う範疇ではそれでは定義しきれないポイントがある。システムはビルディングという階層構造を持っている。例えば「こうのとり」という宇宙船を考えると、宇宙船本体がシステムとしてあるが、その本体は宇宙ステーション全体の一部であり、また、地上システムともつながっている。そう考えると､「こうのとり」のアーキテクティングというと、ど

こが普遍解でどこが個別の解であるかというのは容易ではない。

○ビューポイントについて伺いたい。ステークホルダごとのコンサーンがあって、ビューポイント同士の整合性はどのようにとるのか？橋渡しや調整が難しいところだと思うが。

→（白坂氏）世の中的にはシンセシスの中にすべて入っている。すごく難しい。

○さまざまなビューポイントがあって相反する利害があるところで、結局、対話などによって調整していくのだろうが。

→（白坂氏）確かにコ・クリエーション、対話型で相互理解（アコモデーション）をやるというアプローチです。

○コ・クリエーションのスタンダードやデファクトは？

→（白坂氏）被災地での例やメーカでの実例などはあるだろうが、スタンダードやデファクトな方法はない。アコモデーションとしては、ワールドカフェなどが良く用いられている。

○そこのところを、モデルを使ってバーチャルに見えるようにすると、コミュニケーションをとるには有効になると思う。まさにモデルベース。

○多様性について、社会心理学の論文で、「一人じゃないとイノベーションは生まれない」といわれているが。

→（白坂氏）論文でも二転三転している状況。一つのことを深めるイノベーション創出は一人が良い。探索範囲を広げて生まれるイノベーション創出は多様性が高い方が良いといわれている。

○整合を高めないとイノベーションはできないと思うが、授業などで実際にどのくらいのグループでやったときにイノベーションが生まれるか？

→（白坂氏）イノベーションというと結果を示すのは難しいが、どのくらいのグループでやったときに面白い結果がでるかというと、授業では、学生には 4～7 人でグループを作ることを指示しているが、一番良い結果を出すのは、感覚的に 5,6 人である。

○システム思考とデザイン思考の大きな違いと、双方くっつけないとうまくいかないというのはどの辺か？

→（白坂氏）デザイン思考では、個人が感じることを尊重しながら、多様性と集合知を活用して、ブレインストーミングなどを通じてアイデア／インサイトを生み出し、それをプロトタイプで共感していく。しかし、このプロセスの過程において、目的に到達するために、全体を見ること、またどのタイミングでといった観点を考慮し参加者の多様性や能力を活かすかを考えて、流れを組み立てるところなどはシステム思考が必要となる。可視化、構造化もシステム的にやる場合の重要なポイント。実はデザイン思考の文脈では、欧米でも明示的にはいわれていない。

○大規模なシステム開発、ソフトウェア開発に使えそう。

→（白坂氏）使えると思う。更に、新規事業やもののコンセプトづくりにも生きる。

○IT の世界でいうと、EA（エンタープライズアーキテクチャ）は、一時期盛り上がったが、現在では下火になっている。その原因は部門間の壁を越えられないこと。最近は実態を反映してゴール思考。そのゴールを部門ごとのゴールに分けて落とし込み、アーキテクチャで実現するといった方向。

→（白坂氏）保険金の事例はそのやり方でやった。フレームワークが良くなくて、アナリシスはできるがデザインはうまくいかない。

○運用とビジネスはそもそも観点が違う。一緒のところで同じビューポイントを目指しても、いくらコミュニケーションしてもできないと思う。

→（白坂氏）同じ組織の中でもその役割に応じた観点、ビューポイントがあるので、それぞれ違ってくる。

○インテグレーションは難しいという傾向は、先祖がえりのような気がするが、ERP などは？

○ツールとしてはあるが、現在やられているのは、何を実現したいか、何のためにしたいかをまず決めて、といったゴール思考。

○ゴールを決めるのはトップ？部門？それとも別にインテグレートする役割のものがいるのか？

○ビジネスの価値を生むためには、どうビューポイントを合わせていくかだと思うが、デザインプロセス自体がコミュニケーションの役割を担うのではないか。システム同士のカップリングにおいて、どのように言葉や概念の異なる個々のシステムに関わる者同士のコミュニケーションをとれば良いのかについては、局所座標系を張り合わせた多様体と同じように、システムに局所座標系のようなものを導入すれば良いのではといった話もある。

→（白坂氏）先ほどの質問への回答は、半トップダウンなどの「分割」とコ・クリエーションタイプのボトムアップの両方あると思う。どちらを採用するのかは会社のカラーによりいろいろ。町みたいに大きくなるとコ・クリエーションを全員ですることはできないので、トップダウンでやらざるを得ないと思う。

○いわゆる SI（システムインテグレータ）は何をやっているのか？

→（白坂氏）SI の定義も微妙。インテグレータをそのまま訳すと「寄せ集めてくっつける」。どう分けるか切り分けを決めた後くっつける。本来は SI がすべてをするが、最近、前半「アーキテクト」と呼ばれるようになったので、先の質問にあった話になるが、切り分けが難しい。

○SI（システムインテグレーション）という表現にはかっこ良いイメージがあるけれど、実際にビジネスの現場でやっていることは、コンポーネントやサブシステムの寄せ集めということが多いですね。本当は、システム化する際に付加価値がどう付けられたかが重要です。

→（白坂氏）経験豊富な方は過去の経験知からデザインを頭の中でやれてしまう。ただ、いままでやったことのないことに立ち向かうには方法論をもって当たらないとできない。

○アーキテクチャはさまざまあるということだが、その間の比較は？

→（白坂氏）ISO でアーキテクチャ評価の標準を作ろうとしているが、解空間が広くてどうやって選ぶかが定められない。たとえば、防衛システムで、「予算 2 兆ある。敵機の撃墜率を 2%上げろ」という命題があったとして、やり方は種々考えられる。戦闘機を開発するのか、レーダーの性能を上げるのか、･･･といったこれらの実現可能性をどう比較したら良いのか･･･。世界中で難しさにぶち当たって、検討されていることで最先端のことだと思う。

○難しい。授業ではコンセプトメーキングまでは行くけど、オペレーションまではいかない。

→（白坂氏）授業では時間も限られているので、コンセプトメーキングで終わってしまう。ただ、企業とやるときはオペレーションまで当然ながらやる。このとき、開発当初の思いが最後まで引き継がれないとオペレーションまではいけない。例えば、米国のベンチャー企業では、最初に考えたビジネスでオペレーションまでいくのは２割程度。多くの場合は、最終的には最初に考えたのとは違う形となっている。しかし、形は変わってしまっても思いが引き継がれている場合は、バリューも引き継がれてうまくいく。もし仮にコンセプト作りまで外部に委託した場合は、この思いを引き継ぐことができないので、形をうまく変えることができず、うまくいかない。また、実際に我々が企業とやっていてうまくいくのは、相手先に卒業生がいることが多い。プロジェクトのオーナシップが保たれていることが重要。

○ゼネコンなど実際の建築現場ではラフスケッチをして、顧客の要求分析を十分している。

→（白坂氏）要求分析で使われているのはデザイン思考であると思う。ビジネスエスノグラフィーと言って、人間中心のこの人のために良いものを出すといった考え方で設計をシンプルにすることも採用されている。うちでもアーティスト系の方が入って一緒にやっている。

○アートは教えられる？

→（白坂氏）教えられます。「ガリガリくん」を使った授業などおもしろいですよ。

○デザイン思考は、ものが分かりやすいものはモデルなりシミュレーションしたりして、やりやすい。ただ見えないソフトにおいて、デザインを共有するのは難しい。

→（白坂氏）ソフトで実現されるバリューを見せることでやりやすくなる。IDEO（著名な米国のデザイン・コンサルティング会社）でスマホのアプリの開発のために作ったものは、実現イメージを可視化することで分かりやすくした例。

○デザイン思考は、スタンフォード大学に行って感じたのだが、「スタンフォードの学生だからできるのでは？」と思った。

→（白坂氏）d.school は、いすゞの社員が寄り添うように議論をしながら開発をしているところを見て、それをなんとか真似たいと考えた。しかし、インディビアリストのアメリカ人には、そのように近い距離感で一緒に作業をすることが容易

にはできない。このため､そういったアメリカ人が自然にできるようにするため、デザイン思考という考え方を体系化したと言っている。ただし、確かにおっしゃるとおり、デザイン思考を教え込んでいる相手は、機械工学科の学生たち。システム的にモノを造ることに長けていて、基礎的知識のある人達にデザイン思考の教育をして相乗効果による成果をあげている。日本でデザイン思考だけ教育をしてどうなのだろうと思う。やはり、システム構築思考とデザインのセットで教育すべきであると思う。

○学生に教える場合、小粒の課題と社会や制度などに問題がある大きな課題は違うと思う。持っている可能性を組み合わせて解決することを教授する上で、その課題設定の使い分けはどのようにしているか？

→（白坂氏）企業からテーマを与えてもらう。例えば、今年度は政策投資銀行からはイノベーションを起こすための場として大手町に創られた「イノベーションハブのあり方」、広島にある造船業を基盤としたツネイシからは「地域活性化」、TOSHIBA からは「美と健康」のビジネス、NTT データからは「次世代コールセンタの姿」、折箱産業のヤサカからは「箱産業の未来」、UR からは「虎ノ門再開発のコンセプト」などといった課題をもらった。

○スポンサーを探すのは大変では？

→（白坂氏）今のところ、企業側の研修の一環としてやっているので、0 円。このため、企業側は学生の対応をしなければいけないという負担はあるが、進んで受けていただいている。

○使い方が広い、解が広いということで方法論にならないことを危惧する。どこが科学技術なのだという意見にどう答えるか？

○今年度発行した俯瞰報告書では、この方法論は深堀ができなかった。2015 年版では鈴木先生を統括にして、是非、深堀したいと思っている。

→（白坂氏）わざとメタにしてどうやるかの考え方の考え方を提示することはできるかもしれない。ものを造っている人たちは無意識にできている。そこの体系化、明文化は重要だと思う。ただ、ドメインスペシフィックな部分もある。ある程度分野ごとに重要視するところのポイントが違ってくる。少し踏み出したときには、技術の棚卸が必要になってくる。その分野の特性で考察のポイントの深さがそれぞれある。方法論は少しメタで。また、システムズエンジニアリングまで落としてしまうと足りないところがある。その間を埋めたいと思う。方法論を少しスペシフィックにしてすることも考えているが、その途端に分野ごとの特性にぶつかり、「ここでは使えるけど、こっちでは使えない」ということが起こる。

○まずはメタから始めないと考えにくいかもしれません。メタのレベルで全体を押さえた上で、例を示しながらインスタンスレベルに落としていくと言うのが現実的な方法かと思います。「インスタンスレベルから抽象化していく」という逆のアプローチを併用すると、より確実なものになります。

○物理や数学で説明できない？

→（白坂氏）多分、そう簡単にはいかない。

○意思決定などは、分野化されてきている。そのことで、共通化でき、深堀も進み、学問の進化も望めるようになってきている。

○ユーティリティ関数などが分かっている前提。

○方法論においてもそのような前提がいくつかあって良い。ナノテクや各分野の人たちと議論する上で思考の深さが欲しい。

○その通りだと思います。システムやソフトウェアにおいては、深さを追求といってもプログラムレベルの話ではありません。システムやソフトの世界でやってきたものの考え方を方法論として表現することは、難しいけれど、とても重要だと思います。

○ソフトの世界でも、綺麗なソフト、汚いソフトがあるが、それを説明するのは難しい。

○そうですね。とりあえず動けば使えるし、20 年経って、改造しようとして初めて、その「作りの」悪さが分かる。「後の祭り」であることが多いですね。

○世界観や潜在的哲学的仮定を、学問分野で共有あるいは認識することが重要だと思う。例えば、震災時に議論になった SPEEDY のようなシミュレーションのとらえ方自体は十分に浸透していない。依って立つ哲学的基盤によって、シミュレーションの考え方自体は異なるはず。自然科学的世界観とは違って、社会システムを対象とする分野において、知識に対する誤謬を認めるかどうは、世界観の問題である。深さという意味では、そこも追及すべき。

○方法論を持ってシステムを開発するのは大変重要だと考えます。だが、自身の経験を顧みると、「きちんとした方法論をベースにソフトウェア開発をしてきたか」と問われると、内心忸怩たるものがあります。それでも、まがりなりに開発はできたような気もします。方法論以外に何がシステムを造る力になったかを考えたいですね。２点あると思う。一つ目は、組織知（経験）として蓄えられた「考える力」や「世界をとらえる力」。ある価値判断に基づき一種の直感で開発を進め完成させてきたが、方法論として教えられたことはなかったような気がします。個人レベルの判断でもありません。何によって培われたかが重要です。二つ目は、大きなシステムを造ろうとするときのチーミングです。目的（goal）を持った人（ユーザ）とシステムを開発する人（how を考えられる人＝メーカ）の思いが一つになって初めて、できあがることだと思います。

○デザインについて、情報処理学会誌１月号は興味深い。機能要求と非機能要求について書かれている。日本の某電機メーカの例だと iPhone の一番重要な機能は要求されていなかったが、触ったときのぬるぬる感であったという。要求分析とともにビューポイントとして非機能品質に思い至れるかどうかもポイントだと思う。また、先に出ているものがデファクトになってしまって、後から優れたものを出しても売れず、機能を落として先行に合わせたら売れたという例もある。

○時間軸もある。iPhone のような例だと、先に出した方が世界を握ってしまったら追従するしかない。

○ニーズより早過ぎても駄目ですし。

○やや古いが認知科学の「誰のためのデザイン？」という本の中で、外界にある知識を取り扱っている。知識は頭の中にあるものだけではなく、外界にもあるというものである。知識は、それを使う人と外界（環境）との組で考えなくてはならない。ユーザも分かっていないユーザの要求をどのように掘り起こすか。アンケートという調査法では、作り手側の聞きたいことしか聞いてないから、本当の要求が見つからない。情報システム学会でも議論してきたが、システム成果物に対して何を視点にして評価するか。システムを設計、構築し、利用する主体の目的に照らして、システム成果物の評価視点が決められるべきで、ここの部分は、学問的に知識化すべきところがあると思う。

○大変重要で大変難しいポイント。

○お客は明確なゴールを持っていないことが多い。その中で付加価値をいかに創るかということと、要求要件を満たすということは別物。でも、メタは一緒。アーキテクチャはもしからしたら定量的にいけるかもと思う。

→（白坂氏）アーキテクチャの評価の観点として複雑度というものがある。ソフトウェアの複雑度はどのように計測するかが決まっており、評価をすることができる。しかし、システムの複雑度を評価する標準的な手法はまだない。どのレベルでシステムを評価するか。全体がシンプルでうまくみえるシステムでも、そのしわよせがサブシステムに負荷をかけていたら評価はどうするかといった問題もある。

○そのような例はありますか？

→（白坂氏）自動車などがそうです。部分ネットワークが一杯組み合わせてあるが、すごくシンプルなネットワーク構成だが、それぞれの ECU（電子制御ユニット）の処理がとても複雑なものと、ネットワークはすごく複雑な構成であるが、ぞれぞれの ECU の処理がとてもシンプルなものがあった場合、どちらが良いのかを判断するのは難しい。

○企業だと使い勝手の良いものを組み合わせて造り上げることが多い。どうしてもまだないもの、新しいものが要求される場合、博打だなと思いつつ開発するといった具合。

→（白坂氏）現場の人は分かっているが、定量的評価はない。その指標は世界的にもケースバイケースで。でも必要とのことで、評価軸などが検討されている。

○システムのレジリエンスを考えると、「効率性を最大化しないほうが良い」ということも起こる。

○対象が違うと比較できないと思いますが。

→（白坂氏）評価する対象は同じものです。

○フォールトトレラントコンピューティング限定だと研究されている。

○ドメインによると思う。CREST の DEOS（実用化をめざした組込みシステム用ディペンダブル・オペレーティングシステム、所先生研究統括）で研究されているものだが、アーキテクチャは最初から綺麗なものはできない。運用とフィードバックを経て開発が二重ループになっていて、実際に使える思想だと思う。

○議論をあえて止めませんでしたが、議題２－２「人材、国内外の現状について等」が今回も議論できませんでした。次回、この辺りを話せる方はいらっしゃいませんか。
※提示いただいたご講演者候補リスト別途参照
○本日のご講演や議論のまとめとして白坂先生お願いします。
→（白坂氏）システム構築は論理だけではできなくて、選び出さなくてはならないところの補助としてデザイン思考が必要。このデザイン思考は、考え方、理解、発想力を助ける。

・おわりに
○2 月 17 日（月）に慶應大（日吉）にて、MIT エンジニアリングシステムズ（航空）のオリバー・ディベック（de Weck）氏（スイス）に基調講演いただく予定。お時間があれば、是非ご参加ください。

以　上

付録６．第３回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 26 年 1 月 23 日（金）　15:00～18:00

場所：JST 東京本部別館　4 階会議室 H

［プログラム］

１）配布資料確認、前回議事録確認

２）「システム構築学」の創成に向けて

２－１：システム構築の困難性

・大規模開発を成功に導くアプローチ　－あるシステムの開発－

大島啓二　一般財団法人日本科学技術連盟嘱託

２－２：システム構築俯瞰図の考え方

２－３：分野横断を推進する仕組みを入れたことで成功した事例について

３）おわりに

・今後についてと次回の確認

２．資料

前回議事録

資料１　大規模開発を成功に導くアプローチ

資料２　ATOS 開発メモ

資料３　JRATOS（日立評論）

資料４　ソフトウェア品質の多様性（品質誌）

資料５　アーキテクチャとアーキテクト

資料６　プロマネの要諦

３．議事メモ

・大規模開発を成功に導くアプローチ　－あるシステムの開発－（大島氏）

質疑、議論

○広域光ネットが切れるとすべて切断されてしまうのか？

→（大島氏）縮退して部分的に生きられるように、自律分散型になっている。

○自動車会社で生産ラインの受注側になったことがある。会議参加者はふつう５名くらいだが、その会社は、部署が違う 30 人くらいが常に会議に出席して、自分たちをそっちのけでお客様同士が大ゲンカを始める。１時間くらいたって、なんらか決まったらしく「それではよろしく」といわれ、何を？と思ったことがある。このように一つのシステムを請け負っても人のルール、会社の規則、他のシステムとの統合など、そのシステムの仕様だけを見て開発できるものではないというのが、現実。JR においても他のシステムとの関係、外部との関係と軋轢など、どう一体化したのか興味がある。

→（大島氏）JR では、組織の形やおのおののミッションが自動車会社とは異なるような気がする。取りまとめ部門がしっかりしていたこともあって、システム開発に入ってからはご指摘のようなことはなかった。裏話として、JR 内部ではそこまで自動化して良いのか等、信号機メーカや運行管理関係の現場といった異なるレイヤーからの異論や問題提起があったと伺っている。

○会社のカルチャーによって違うのですね。

→（大島氏）実際にものを生産する自動車と運行管理を中心に行う JR とでは違うと思います。本来なら、システムを組み上げる際に、現場の機器もリニューアルしてより近代的にするのでしょうが、その時は、現場の機器はそのままでシステム化を行うことになりました。変数が少ないと言う意味ではとても楽でした。ただ、今後システムと機器を合わせて輸出するとなると、もう一度全体を考えなくてはならないので、大変かもしれない。

○製鉄所のシステムとは違いはあるか？

→（大島氏）どの視点で違うかによるが、構築方法やアーキテクチャは似ていると思う。

○まとめあげるところ、デザインするところなどはどうですか？

→（大島氏）鉄鋼は工場ごとに分散する傾向。鉄鋼は会社や時代によって目的関数が異なり、極端に言えば製鉄所ごとに開発の進め方が違うこともあった。JR は最初に全体を考えなければならない。ただ、製鉄会社でも、緩やかではあるが本社を中心にした「標準のガイド」があったようだ。どうしても、製品の違いを含めて現場固有の問題は残るので、システムとしては製鉄所ごとの対応が必要だった。

○日本メーカの英国への鉄道販売などに見られるように日本は競争力を持っていると思うが、どこが日本のシステムの評価点なのでしょうか。

→（大島氏）外から見た時の理解のしにくさ、言い換えれば「得体の知れなさ」でしょうか。技術的には日本も欧米ももはやそれほど変わらないのではと思います。日本のメーカは、お客様のニーズに合わせて最後まで造り上げてしまう。メーカとユーザが一体となって価値観を共有してシステムを造り上げる。しかしそのやり方は、欧米流の契約、商習慣では通用しないかもしれない。技術の面での優劣についてはもう少し考えてみたいと思う。

○システムに関して日本が強いというのであれば、我々のユニットは存在しない。それとは別に、JR のシステム化でサイエンス的な要素（例えば、最適化、OR、モデリングなど）やシーズはどのくらい使われましたか？それともまったく使われませんでしたか？

→（大島氏）アーキテクチャ面、たとえばネットワークアーキテクチャ、自律分散などでしょうか。

○サイエンティフィックなベースのところで大学からの知恵を使ったということはありましたか？

→（大島氏）大学とは関わりはありませんでした。自社の研究所に相談しました。例えば安全性の確率（不安全な状態に陥らない確率）をテン・ナイン（99.99999999%）に設定するなど。（注：テン・ナインは正確ではないかもしれない）

○テン・ナインを計算する仕組みのところにサイエンティフィックな裏付けがあったはずですよね。

○そこにいたる設計解を選び出す作業がどういう形で行われたのか、という部分に興味がある。そこがサイエンティフィックになればと期待する。

→（大島氏）残念ながら学問的な裏付けを考えるのは後回しにしてしまった。身もふたもないけれど、「勘と経験」でしょうか。ではもう一度やるとなったら科学的裏付けを取りながら開発できるかと言われると、これまた自信がない。新しいシステムや製品に対しては、必ず試作開発を行う（サイエンティフィックな裏付けをとる）ハードウェアと、開発そのものが試作であるソフトウェアとの違いがあるかもしれない。

○最適化とか意思決定とか、サイエンティフィックなベースを使ってもらいたい。

→（大島氏）確かにお客様もそのような裏付けを求めていたような気がする。

○発想するところを Science にするというより、発想することを Science して一般化することにより、現場から出てくる、そこから生まれるものがあると思う。

○同じ意見であるが、システム開発をやられていた時には、勘や経験でやられていたかもしれない。しかし、後から見直して、整理して体系化（理論化といって良いか分からないが）してみた時に、使えるものにできたものがあれば、教えてもらいたい。

→（大島氏）事例でしか語れないというところに「知の伝承」という意味での課題が残っていると感じますね。

○制御は裏付けとなる制御理論がある。私も携わったことがあるが、プラントの場合は主導原理があった。そこまでやらなくて良いといった場合にも、やらなくて良いということへの根拠を示せる。手でやっていたことを自動化すること以上のことは？

→（大島氏）JR 東京圏運行管理システム（略称 ATOS）は、現場での運用を基にして（そのまま活かして）システム化（機械化・自動化）した。次期システムでは列車運行（現場の運用）そのものを見直してシステムを組むべきなのかもしれないが、今のシステムに手を入れるのは相当の困難が伴うし、コストに見合う価値があるかどうかは不明。

○事故からの復旧などは？現在は我々乗客に大きな混乱をもたらしているが、そこら辺から何か新しいシステムが要求されるのではないか？

→（大島氏）私が仙台出張に行ったときに常磐線内の倒木事故で何時間も電車が復旧しないことがあった。駅員に聞くと、「実はシステム化が進んだので駅に人がいなくなってしまい、･･･」と言われたことがある。「定常」を前提としたシステム化は、何かあったとき（異常時）には弱い。「定常（稼働率）重視」と「異常時の対応重視」はある意味トレードオフの関係にあると思う。

○何かあったときには、先ほど見せていただいたダイヤグラムの線をずらしたり、折り返したりといったことがあると思うが、それは自動化されている？
→（大島氏）そこまでの自動化も視野に入っていたと思うが、実運用では、「サポートのレベル」にあえて止めたと思われる。ガイダンスを示し、予測を出して、最終判断は人が見て決定する。そこまで自動にすると、人が判断できなくなる（安全性を担保できなくなる）といった考え方もある。人間系を含めたシステム化の問題ですね。
○これ全体を理論化しようとするといろいろ出てきてしまうけれども、ここの局面は理論化できるといったレベルでも良いかも。
○現場では勘と経験とコストで決めてきているが、ここまでは計算できるなど、どのくらいの粒度で整理するかは、私たちの研究対象だと感じた。
→（大島氏）工場サイド（システム開発担当）の力量（スキル）を一つの学問レベルで受け止めるということなのでしょうが、考えろと言われても、役割が違うのでむずかしいのでは。「構想する人」、「作る人」、「意味づけする人」といった役割分担が重要。少し視点を変えて、「こう整理したけどどうですか？」とみせていただけると、「あぁ、なるほど」ということになり、一つ上のレベルに止揚できる気もする。
○以前にも出たが、コ・クリエーションのやり方。どう議論を持ち込んだかというよりどういう議論を想定していたか、結果からこうだったのではないかというアブダクション（推論）も得られるし、そこを理論化、共有化することだと思う。それが生まれるのは議論から。得られた理論は単体で評価することは不能。どの条件下で有効か、サイエンスの側からするとなるべくそのような前提をなくしたうえで整備したい。それが理想であるが、綺麗な理論は現場にフィットしない。制御理論も否定するわけではないが、断片的であると言わざるを得ない。
○まさに我々が今やろうとしているところ。鈴木先生主幹でその分野を俯瞰したい。確かに難しいと思うが是非協力していただきたい。
○システムを考えるときにシステムの境界をどの辺に置くか、90 年代には運行の管理はシステム全体の一部であったと思うが。
→（大島氏）人しかできないことは人。機械でできることは機械に。安全やサービスなどの本質的にかかわるところは「人」が責任を持つべきだと言う考えを、お客様は根底に持っていたと思う。
○アクターネットワークセオリーといって、人工物も人も同じ構成要素として分けないで構成要素間の連携を考えるといった考え方もある。
○DOD の Architecture Framework（DoDAF）のオペレーショナルビュー（Operational View）でも、システムのアクティビティをアロケーションするときに人でも良いし、モノでも良いという概念。役割は後で決める。
○今、世界的な情報システム界ではそういう考え方。
○設計とは、アブダクション的思考とアナリティカルな思考の融合。そこを俯瞰してい

るところはない。やらなくてはならないところであるし、徐々に考え始められているところ。難しいがやりがいがある。

○ただ、やっていることはシステム的でも融合と思っていない人たちも多い。特にドメインスペシフィックの人たちは、これまでの経験に基づけるところが多いため、気付きにくい。言われて、「あぁ、自分たちのやっていることだ」と気づいたりする。

○そういう人たちはシステム化を示されて喜んでくれますか？

○人によると思います。

○今できている人たちは、自分たちはすでにやっているとしか思わなくても、次の世代につなげようとするときに生きるのではないかと思う。

○確かに、経験できていた人たちが、次世代を育成するときに「システム化」が果たすべき役割は大きい。

→（大島氏）「経験」は正しく伝えることができれば、「次の世代の育成」や「技術輸出」（欧米に持っていこうとするとき）に生きる。経験は強みとすべきネタ（材料）なのだけれども、強みにする「仕組み」がなくては、単なる懐古（懐旧）に陥ってしまうので危うい。共通化、体系化したものを、事例とセットにしてもっているとものすごく強い。

○日本では経験や勘が個人で蓄積されて、その経験は場を共有することで経験から経験へ引き継がれてきた。相撲部屋や研究室など。ただ、若手と文化を共有できなくなってきた。言葉やメンタリティが違う人たちに伝えることが難しくなってきた。「ちゃんと教えてくれればやるのに、教えてくれないからやらない」といったこと。これでは、今までの強いところが途絶えてしまう。

○我々の主張も負けた原因はそこにあるのではないかと。いわゆる暗黙知。

○理論化する行為が伝承の媒体になると考える。アメリカでは多様なバックグラウンドの人たちに伝達しようとしたとき、理論化の必要性があった。逆に日本では、D スクールみたいなものは必要ないと言われるが、そうではない。

○その通りだと思う。だからこそやらなくてはと思う。経験のベースがあることはとても強い。それは、経験があるので言われたときに違和感を持つ人が少ない。全部理論化できないかもしれないが、少しでも理論化でき、見えるようにすると、引き継ぎやすくなるし、渡しやすくなるし、説明しやすくなる。どうやったらコ・クリエーションできるか我々もやろうとしている。

○システムの大規模障害を見るとこんな対策もしていなかったのかとか出てくる。システムをレジリエントにしようとするときにそれに必要な知見は一杯ある。ただ、全部を言葉で並べ立てられないし、そうしたとしても必ず漏れる。現代のシステム知は認知限界を超えている。これは現代のシステム科学の課題であり、30～40 年かけてきた経験を全部伝えることは不可能であるし、システムはどんどん高度化して非常にスペシフィックになってきている。これらに対処する仕組み作りが必要。

→（大島氏） 日本では企業間（特にメーカー間）の交流が少ない。交流を求めて学会などに参加したりすると、「彼は余裕がある」と見られたり。これからの日本に

は、企業の枠を超えたつながりが以前にも増して重要であるように思う。何か欲しいと思う（「組み込み領域」は草の根レベルで場ができ始めている）。ただ、日本の経営者が 10 年先を見通して布石を打つことが少ないことが課題である。目の前の課題に精一杯と言えば言い過ぎか。成功している同業他社がやっていることは真似する傾向があるが。

○OR についても 1940 年後半から盛んになり、50 年代なると吸収できたとなれば、下火になり。人工知能など、陽が当たっているところに飛び付き、消化したら去るといった（消化したかどうかも疑問だが）、そこで育てていくといった発想はない。

○海外から見ると日本の研究者は吸収するだけで、発言や発信をしなくて不気味といわれている。それでも優秀で日本に持って帰ってそれを有効に活用したりする。「それだけ優秀なら、発信すれば良いのに」と非常に気持ち悪がられている。

○日本の強みという部分で、科学技術においては、技術そのものが変わってきてしまった。20 年くらい前までは通用したが。

→（大島氏）いままでの延長線上では負ける。

○ガラパゴスという考え方があって、日本は人口からいったら世界 10 番目でガラパゴスにしたら、大きい。以前は家電を買っていた人は先進国の 7 億人であった。現在は 35 億人といわれている。日本には 35 億人に供給する力はない。韓国は人口が半分。（韓国は国内市場規模の点から）直接世界に出ていかなくてはならない。これから日本はどういう方向でものづくりを考えなくてはならないか、中小企業の経営者が話をしていた。

→（大島氏）日本が、これから世界に対してどういう価値を提供していくかが重要だと考えている。

○日本の市場が縮小する中で、先進国相手だけでも中国には勝てない。暗黙知が全部分かった時点でコピーされる。

○日本は分析されて真似される。大部屋方式というのも欧米でやられている。宇宙の分野では、欧米で注目されている新興企業などが、「安くするには 7 割を内製化」といったりしている。これは、日本がグループ企業で阿吽の呼吸でやっていたことをやり始めるイメージである。一方で、JAXA の契約では気心のしれた発注先に発注する場合でも、各種審査などの多くの手続きと書類を必要とする形態をとる。すごく欧米的な進め方をする。まるで逆転している。

○モジュール化は、世界の流れだと思う。これは、外部発注が基本方針だと思うが。

→（大島氏）取り替えのきくものは、確かにモジュール化だと思う。

○バリエーションの多い仕様と大量発注の仕様は違う。

○常に揺り戻しがある気がする。ある方式が取れられてそれが成熟化すると昔のものに戻る。日本の場合は、欧米の後追いが多い。ある方式を国内で開発していても独自に評価してその方式を採用するという形ではなく、欧米の評価を得てから国内に入るという欧米の一周遅れになっているのではないか。

○システム化だけはどうしても真似ができていないと思う。日本メーカも頑張っておら

れるが、だけどまだまだ。

→（大島氏）得意不得意がある。得手のある部分は強いが、Google など発想が「飛んだ」ものには、からきし弱い。

○自動車メーカも無人自動車を研究しているが、大切なところを決められない。このようなことをして良いのか等、懐疑心が生じて本気になってシステム化を進めることができない。

○マネジメントの価値の源泉が変わってきているのではないか。すでにお分かりのようにもはや部品には価値がない。大きなユーザは何年も発注できるので、ノウハウを引き継ぐことができる。ラピットプログラムは深さがないものが対象で、逆に社会インフラシステムなどは、日本が劣っているとは思えない。価値の源泉として相手が買い叩ける立場に落とそうとする傾向がある。日本は標準作りが弱い。

→（大島氏）社会インフラシステムのような大規模なものと、コンシューマプロダクトなど、日本が不得意なものと分けて考えた方が良い。社会インフラは果たして欧米流が良いのか？ユーザとメーカが一緒になってやっていくと言う日本流のやり方は、品質や効率面で最適ではないかと思う。ビジネスの仕組みそのものも考えて輸出すると言う視点。

○海外エンジニアと話をしていると、基本的アプローチは似ている気がする。日本よりも人材は流動的で会社が変わっても鉄道屋は鉄道屋みたいなところはあるが。海外のやり方が日本メーカと違っているのだったら、見習うべきことは見習ってと思うがそのような魔法のようなことはないのではないか。

○仕様書の書き方は反省しきりではあるが。後から見ても分からないような。

○仕様書の書き方について、その目的が違うからではないかと思った。納品物として形態を整えるためだけの目的で、実際の仕様は頭の中にあるとか。後々つつかれないようにわざと分かり難くしているとか。記述能力がないとは思えない。

○日本メーカは最後までやってくれるから、そのような仕様書でも通るとか。

→（大島氏）15 年スパンくらいで帳尻が合うというくらい（の懐の深さ）で。鉄道も鉄道会社と一緒だと海外に出て行けるが、メーカだけだと出て行けるかは疑問。

○CREST で水システム（持続可能な水利用を実現する革新的な技術とシステム）の依田副研究総括の話では、海外にも出しているけど儲かっているのは商社だけという話。

○宇宙の分野では、仕様書を意図的に詳しく書くことがある。設計書レベルのことまで仕様書に書いて、変更があったら仕様変更として追加の費用や時間をもらう。これは意図的にやっているので良いのだが、若い人はそれを見ていて、仕様書はこう書くのだと思ってしまったりする。そうすると、そういった仕様を海外に持っていったりすると失敗する。

・システム構築俯瞰図の考え方（システム構築学の俯瞰図案、軸設定、ビューポイント）

主査：システム構築学の俯瞰をしたい。ハンドブック的なものをまとめようとしているが、どのような見出し項目があるのか。まずは、システムのビューポイント、ライフサ

イクル、ビルディング階層といった軸をとり、既存の技術のマッピングを考えたい。抜けている軸や視点、フェイズ等はないでしょうか？

○何があるとシステム分析で、何があるとシステム設計なのかその右左の軸は？

→（主査）ライフサイクルは時間軸。ビューポイントは時間軸ではない。どういう側面で評価すべきか。

○３次元にすると難しいと思う。

○軸というよりマトリックスなのでは？

○ビューポイントとして、「システムの価値から見た性質を表す」とみて良いのですね。

○評価の側面なのですね。整理が難しいのは良く分かります。

○システムの前提条件を明記しておく必要がある。読者はそれぞれ「システム」と聞いていろいろなことを思い、期待する。製造システムなのか、社会システムみたいなものか。組み込みだとすると、軸が足りない気もするし。

→（主査）特定分野に依存しない形のものを想定している。

○それは大変です。

○一応、我々のユニットで使うシステムは、定義があるが。

○組み込みに対する課題は、いままで議論していない？

○システムの分類学が必要と思っていた。システムのような外来語は往々にしてそうなってしまうが、改めて定義を聞かれると明確でないことが多い。パターンとかシナリオとか。俯瞰とはまた別にあると良いと思う。

→（主査）応用分野はまずは外して基本的なところを整理したい。

○気持ちは分かりますが、結果として応用分野を外すというのは良いが、一旦視野に入れておいて、その上でこれは外すということをしないと、クラウドシステムと聞かれても説明できなくて困りませんか？

→（主査）こういう基準で外しました（採択しました）と説明できなくては駄目ですね。

○一応、我々のシステムは、人工システムを扱うとなっています。それくらいでしょうか。

→（主査）時間もないので切り上げますが、ハンドブックみたいなものを作るのだったらこういったフレームが必要という意見をお寄せいただければありがたい。また、次回も少し議論させていただければと思う。

○INCOSE の SEBOK（Systems Engineering Body of Knowledge）の目次と見比べても非常に近い観点だと思う。ただ、システムズエンジニアリングに足りないところ、アブダクションとか発想とかいう観点がこの目次には入っていないので、これらを一緒に語っているところが必要であり、良いところだと思う。

→（主査）そのハンドブックの目次を後程、送ってください。

・分野横断を推進する仕組みを入れたことで成功した事例について

○CSTP（内閣府総合科学技術会議）に SIP（戦略的イノベーション創造プログラム）

などでプログラムを横串で管理する仕組みが必要と提案した際、「そのように横串を通したことで成功した事例はないか？」と聞かれた。そのような事例を知っていたら教えて欲しい。

○ヨーロッパのフィンランドなどで政策意思決定機関として採り入れられているフューチャーセンターは事例ではないか。

○国内で成功事例はあるわけない。欧米ではあるのだろうか。

○企業が社外取締役を採用することによって業績が著しく良くなった例などは成功事例かもしれないと思った。

○仕組みを入れたことだけでうまくいくとは思えない。お互いの価値を増幅することに喜びを見出す人が上に立って指揮することが重要だと思う。単に情報をまとめて共有するだけでは駄目。

○人の問題になってしまう。

○COI シーズニーズ創出事業で、多様性を生かして新しい研究アイテムや事業アイテムを創出するための方法論づくりの委託を受けている。全国 30 大学ほど。

○我々のユニットでもシステム化を説く際に実例が欲しいと良く言われる。どちらが先かの問題でまずはどこかで突破口を開かなくてはならないと思う。

・次回について

○商社がシステム化を実施しているところと見ているが。

○商社の方は新たな What を見つけ出すことは得意だが、そこからデザインすることは不得意。既存のものを安易に持ってきてしまって、せっかく良いアイデアなのにモノに落とすことができない、と商社の人に言われた。

○商社のアクティビティについて聞きたい。

○損切は早い。

○確かにサプライチェーンなどは動的に変わるシステムであるし、契約関係が変わるだけで最適値が変わると思う。

○アマゾンとか新しい取組みを実施しているところの話は聴いてみたいですね。

○産業プロデュースの方に少し打診してみます。

以　上

付録７．第４回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 26 年 2 月 19 日（水）　15:00～17:00

場所：JST 東京本部別館　2 階会議室 E

［プログラム］

１）配布資料確認、前回議事録確認

２）「システム構築学」の創成に向けて

２－１：システム構築の困難性

・非機能要求グレード　～システム基盤に対する非機能要求の合意形成～

山下博之　（独）情報処理推進機構技術本部ソフトウェア高信頼化センター
グループリーダー

２－２：システム科学技術の歩み

３）おわりに

・今後についてと次回の確認

２．資料

前回議事録

資料１　非機能要求グレード　～システム基盤に対する非機能要求の合意形成～

参考１：・(INCOSE）SYSTEMS　ENGINEERING　HANDBOOK　目次
・SEBoK　目次

参考２：システム科学技術の歩み

３．議事メモ

・非機能要求グレード　～システム基盤に対する非機能要求の合意形成～（山下氏）

質疑、議論

○エンジニアリングシステムズの中でも指摘されているが、reliability、dependability などの「イリティ」と呼ばれるものがある。これらが非機能要求の全体を指しているわけではないが、システムでは注目しなくてはならないポイントとして挙げられている。

○コストの問題は、非機能要求の項目に入っていないのですか？コストを出さないとユーザは選べないのでは？

→（山下氏）ベンダとユーザとの調整の中で見積もりをすることになる。

○コストは一般化できませんか。

○（明示された）機能と非機能をもう一段下げて、ベンダが置き換える作業が必要なのですね。

○量産品の受託は一般化できるが、個別の事情があるものは、一般化できないのではないか。

○これは、「メトリックスがないので決めましょう」ということで指標を決めたもの。
○一般化を狙っているものですか？それとも全体の規格化が目標ですか？
→（山下氏）それぞれ事例ごとに異なる。多くの選択肢の間で検討するときに使用するものです。
○ユーザとベンダ間で合意形成するときのモデルであると思う。どうしても実現しなくてはならないものというよりは、参照モデルであり、方法論の一つであろうと思った。
○双方で納得する材料として非常に強力なものと感じた。お客さまの要求をある程度明確にできるので、これらによって見積もることができる。まったく新しいものの場合は難しいと思うが。
○性能くらいまでであれば機能については議論ができる。後は使ってみてから。という感じであるが、その時点でのリスクが減る可能性はある。
○スライドの背景（4）で書かれている部分。
○大項目、中項目くらいは共通化できそうな気がするが、小項目になるとお客さんによって違ってくると思う。都度カスタマイズして使うのか？
→（山下氏）項目は同じマトリックスを使う前提になっている。レベルの値がお客さんによって違う。ただ、網羅的に作成されているため 236 個ある。お客さんによっては必要ないものもあるので取捨選択して使用してもらう。
○初めて使う方にはその使用方法が明示されている必要があると思うが、各小項目レベルで機能要求を仕様書に書き起こす方法とそこから変更するといったメトリックスはあるか？
→（山下氏）その部分は各ベンダのノウハウによっている。
○実際にまだベンダは各自のリストを持っていて、各ベンダのメトリックスによってなされているのが現状。
○システムの運用モデルとして、事前にそのモデルを想定するわけだが、災害が起こった場合等、非常時などを想定して、段階的なマルチシナリオになるのではないか。
→（山下氏）平時と災害時は分類を変えて想定する場合があると思うが、そこまでは議論しにくい。
○その辺になると、業務が入ってくる。
○「想定の範囲外」というのでは実際に使用する場合は駄目だと思う。いかなるフェイズでも動くものを想定すべき。
○p.14 に冗長化の例が記載されているが、もう少し広げると縮退運転とか入れられるようになると思う。
→（山下氏）どの機能を落とすかは一般化して規定できない。
○BCP 中でどう活用するかもポイントだと思う。
○この中ですべてやろうとすると大変。分かり易さが重要だと思う。
○ベンダの差別化が必要だと思う。何に重点を置くかで違ってくるし、すべてのケースでこれを綺麗に書き切れるわけではないと思う。だからこそ、応用例として地方版（ドメイン版）として簡易版を作成したのだと思うがそうですか。

→（山下氏）本当に必要なものを選んで示して提供したもの。

○p.10 に揚げられている中で、外から見たときの機能要件的なものは比較的分かりやすいが、実際はなかなか目が届かない内部的なものの造りが重要だと思っていて、拡張性、保守性、移行性などはすぐには分からない。ソフトウェア構造やアーキテクチャの綺麗さまで書き込むなり、示さないと分からないし、保障できない。ユーザがそこまで求めていないというか、表現できていないのかもしれないが、ハードと違うソフト特有の機能要求としての視点が必要だと思う。

○項目としては入っている。性能や拡張性などは担保されているということ。

○それをやるにはどうすれば良いかまで保障しないと。それをするにはソフトウェアの構造まで踏み込まないとならない。

○さっきの意見に近いが、私も綺麗に整理されていて素晴らしいと思ったが、はたと返って、自分が担当する場合、ここから先どうするのかと思った。こういうことを決めることがソフトウェアの発展に寄与するかというとそれは第一歩だと思う。この先どうすれば良いのか。99.9%成功したシステムの例としてどういうベストプラクティスがあるのかを解析するために仕組みの成功例を集めるのだろうか。そして、これを追求していくとソフトだけでなく、センサーや通信、全部包括して考えなければならない。ハード、人の運用、ソフトウェア･･･。かなりのものが一緒にならないと解は出てこないのではないか。一つ一つはものすごく大きなこと。この先は、ベンダの仕事？それともベストプラクティスを集めて我々が考えていくこと？

○そこまで行くと、ソフトウェア人材の教育まで踏み込まないと駄目だと思う。日本の価値が認められるためには欧米の追いかけでは駄目で、そういうところで人の価値みたいなものが出てくると良いなと思う。

○実現の方法論までいくと業務を考えなくてはならないし、個別の話になってしまう。

○ソフトウェアの品質に視点を置いて次に行きたいですね。ハードウェアでは解決することが、ソフトウェアでは解決できない闇がある。ここまで整理してくださったことに感謝して、その上で次のステップを一緒に考えていきたい。

○お話いただいたことがベンダとユーザのネゴシエーションのツールになることは良く分かった。次のステップとして、インフラや防災などのシステム構築においても同様の困難さがある。この場合は、ステークホルダが多様でベンダとユーザというわけにはいかず、切り分けることも難しいと思う。この同様の困難さを整理して、どういう風に引き受けて、システム構築の困難さに位置づけて一般化して学問にして行くかということを考えたい。機能か非機能かという区別も多分ないと思う。

→（山下氏）このツールは産業界の事例から抽出したものであるが、アーキテクチャなどはまだベンダの暗黙知の中にある。一方、構造化や方法論も出てきているが、実務とのリンクがとれていない。ここのリンクがとれればもう少し先に進む気がしている。

○リンクについて、考えていないのか、難しいのかどちらでしょう。

○両方でしょう。考えていないし、難しい。

○多くの場合、システムは開発者などの自分の好みで造ってしまう。
○一般に「私はこれをやっているのだ」ということが表出化されていない。非機能性要求という今日の整理を出されると、「私がやっていることはこれです」といえると思う。これを表出化のツールに使って足りない分を追加するなり修正していくという点で、取っ掛かりとして非常に良いと思う。何か見える形で出さないと。
○システム構築をするメーカの立場から言うと、困難さはメーカ内で暗黙知されている。方言といって良いのかもしれないが。知識と経験が事例ベースで集積され整理される。そういうものが次にあるなとは思った。
○障害事例を分析することも我々はしている。似たような障害が、さまざまなところで起こっていることが分かる。ただ、その原因を追究すると多種様々。アーキテクチャの問題であったり、体制、マネジメント、体質（企業文化）、オペレーション等であったりする。全部考えないとベストの解は出てこない。
○確かに、いきなりできるとは思えない。
○後の研究の目的と重なるが、できると思う。システム構造を考えるときはトータルで考えなくてはならない。可用性の研究もあるし、移行性の研究もあり、システムの研究も･･･。全部考える必要がある。そして、実際と構造とのギャップもある。そこのギャップを埋めていくことは難しいが、みなさんとやって行かなくてはならない部分だと思う。
○道は遠いが、可用性、拡張性、運用、環境を考えたうえで、アーキテクチャをきちんとすることは難しいことだができること。そこさえあれば、何があっても耐えうる基盤ではあると思う。
○モジュール化も水平、垂直いろいろある。どう切るかとういうのはやっぱり全ライフサイクルを考えてやることになる。モジュール化も汎用ではないと思う。
○製品によって、モジュール化のやり方はパターン化できる気がする。コンシューマ、組み込み、社会インフラシステムなど、ジャンルを区切ればやり方はある気がする。
○同じ組み込みでも、個別業務によってモジュール化の粒度は千差万別。綺麗にモジュール化をすることは難しいと思う。
○完璧を求めるのではなくて、いずれにしてもモノは造っている。それを開発者の好き勝手に造らせるのではなくて、少なくてもこういう枠組み、こういう考え方というメタなレベルでの構造化はできると思う。
○アーキテクチャがいくつかのモデルになったとき、それらの間の評価をするときに、どんな評価軸があるのか。これには合うがこれには合わないということが、きちんと分かっていることが重要。そこは学問としてやっておかないと後の人につなげられない。あえて、間違いを選ぶこともあると思う。その選ぶ理由があるなら、それは分かった上での選択であることを明確にしておく必要がある。
○システムは非常に良く使われる単語とされているが、そのシステムをある意味で分類するということか。それは誰もやれていないと思う。ある軸を取って分類すると見えてくるかもしれない。やってください。

○情報システムは、ソフト、ハードだけではなく、人間が要素となって動作するもの。ソフトウェアだけの話ではない。場合によっては、機械でなくて人を置いた方が良いという場合もある。このフレームワークとして 236 あるが、これはソフトやハードで実現しない（人で実現）というオプションは選べる？
→（山下氏）運用とか人の話は対象外。
○経営判断として意思決定する際は、必ず、ソフト、人を要素とした系全体としてとらえる。もう少し人を要素としてシステムに組み込むことが必要だと思う。
○ますます複雑に。
○IPA が発行している非機能性要求の冊子を読ませてもらった印象では、人の問題も含めて非機能要求という枠組みでユーザとデベロッパ間の合意と理解した。本日の話の中でもメトリックスを作るところで人間が処理することもできそうだが。
○本来、ビジネスの目標、戦略に対して、機能の取捨選択をする際にはどの部分を人でどの部分を機械でということを必ず行なう。上位の目標との整合性をどのタイミングでとっていくかは、BCP のフレームの中で考えることになる。情報リスクマネジメントという枠組みの中で、情報のレイヤーと目標のレイヤーがうまく結び付くかなという印象を受けた。
○システム開発をするとき、多くの場合、ユーザが意識するのは平常時のこと。この枠組みの中で障害時もユーザに注意を喚起させるという点では意味がある。
○ユーザ側で経営の目標があって、それに対してどう実現するか、戦略を判断することは、それは別にある気がする。
○おそらく、それを最初に決められれば綺麗だが、並行してやっていくことになると思う。行ったり来たりしながらやっていくのが実際。マネジメントのツールはあって、システムのツールがこのようにあるとしたら､そういう意味ではうまく併用するとか。ただマネジメントのレイヤーとシステムのレイヤーをつなぐ仕組みがない。
○ここでおっしゃるシステムとは情報システムのことだと思う。マネジメントもシステムの一つであるが、強調したいのは、システムとソフトウェアシステムを混同されては困る。IT やソフトに長けた人とシステムに長けた人は違う。IT が発達する前からシステムは存在していた。IT システム＝システムではない。混同しているがゆえにシステム化といった場合、IT 化すれば良いといった誤解が生じる。
○素人には難しい。
○こういう話を学問でどう結び付けるか。コンペティションはどうだろう。データマイニングの手法や、ロボカップのように、あるコンペティションの仕組みを作ってあげて、システムアーキテクチャとは何かの理解を楽しくやれたら若い人も学べるのではないか。
○ET（Embedded Technology）ロボットコンテスト（組込みシステム技術協会主催）というのがある。ハードやレゴブロックが与えられていて、その中のソフトをいかにきちっと設計できるかを競う。評価はモデルの設計の綺麗さと実走。
○そういったものでアーキテクチャを競うものがあると良い。

○cloud 上の実験でユーザを巻き込んでソーシャル実験するというものもある。ハッキングやセキュリティの分野で進んでいる。
○多賀城市の制御システムセキュリティセンターでは、インフラのシミュレーションができていて、これは攻撃を受けた時の訓練にもなっている。
○企業でもなにか障害を起きた時の対応をシミュレートする場として、現実を切り取って小さなモデルを作成しソーシャル実験ができると良い。
○JAIST の篠田先生のところでは、ある町をシミュレートして EMS の効果的なマネジメントアルゴリズムを作成している。人の入ったシステムをどう効率的に造るかどうオペレーションをするかは、なかなか扱い切れない。ただ、うまく切り出せば箱庭を造ってできるかもしれない。
○国主導でできれば良いですね。

・「システム科学技術の歩み」について
2/21 の国際シンポジウムで配布予定の「システム科学技術の歩み」について、各分野の専門家の目で足りない部分等のご指摘をいただいた。

以　上

付録８．第５回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 26 年 3 月 26 日（水）　13:00～17:00

場所：JST 東京本部別館　2 階会議室 A-2

［プログラム］

１）配布資料確認、前回議事録確認］

２）「システム構築学」の創成に向けて

２－１：システム構築の困難性

・情報システムにおけるデザイン科学研究

田名部元成　横浜国立大学国際社会科学研究院教授

２－２：ディスカッション

３）ミニワークショップ「システム関連 KW のマッピング」

４）おわりに

・今後について

２．資料

前回議事録

資料１　情報システムにおけるデザイン科学研究

資料２　ワークショップの進め方

３．議事メモ

・情報システムにおけるデザイン科学研究（田名部）

質疑、議論

○INCOSE（The International Council on Systems Engineering）とシステムエンジニアリングとの交流はあるのですか。

○重なる部分もあるが、狙っているところが少しずれている。INCOSE は、設計する方にターゲットをおいている。ただ、最近、システムサイエンスなどいろんな方向に広げなくてはならないと始まっているので、一部重なっている人たちはいると思う。エンジニアリングはもともとクリティカルなシステムをいかに効率良く造るかが主眼である。防衛・宇宙から始まって、発展してきた。もともとのスタート基盤が違う。

→（田名部氏）米の ABET（The Accreditation Board for Engineering and Technology）では、大学の情報系５分野を次のように分けている。CS（コンピュータサイエンス）、CE（コンピュータエンジニアリング）、SE（ソフトウェアエンジニアリング）、IS（インフォメーションシステム）、そして、ここ最近できた IT（インフォメーション・テクノロジー）。このうち、CS と IS はコンピューティングとして、エンジニアリングと分けて、プロフェッショナルのプログラムの国際基準を作ろうという動きがある。

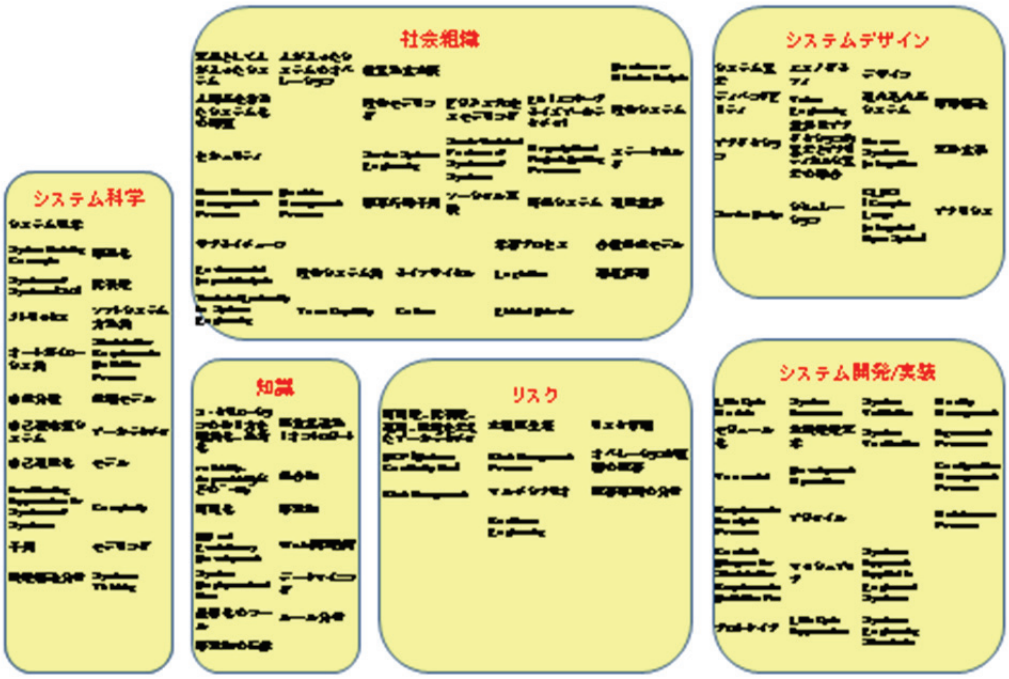
○実際の開発の現場では、提示されたような枠組みの中で考えて開発はしていない。そこをどうやるかは難しい。実際に製品を造りながら学ぶなどはやっている暇がない。
○日本にはこのようなコミュニティはない。
○開発だけでなくて運用も含めて全体システムをとらえて行かなくてはならない。いろんな要素を考えなくてはならない。オペレータの行動パターンやユーザの使い方を含めてシステムとしてとらえ、考えなくてはならない。幅が広がりつつあるので、このようなモデルができて、きちんと整理しないと本当はいけないはずという気はする。
○インフォメーションシステムというコミュニティは「システム」という方向で統合されていくべきと思う。
→（田名部氏）基本的にシステムという考え方、発想は必ずある。システム思考だけではなく、システム科学、システムセオリーは、必ずどこの教科書でも第１章に、章を割いて書かれている。
○システムズエンジニアリングもサイエンスの方に目を向けて行くべきだと思うが。
○ISSS（The International Society for the Systems Sciences）では、ものづくりや実際に使う側への問題や意識を持ってきていて、また、INCOSE ではシステムサイエンスという方向に向かい、共同でワーキングを創ってシンポジウムなどを開催している。
○日本の弱いところが二つ浮き彫りになると思う。①システムを造る人は社会で使われる運用まで含めた一気通貫のものの考え方ができていない。開発者中心の考え。②プロセス等を抽象化して理論化することができていない。本当のシステムにならない。
→（田名部氏）そこを何とか explicit しようということがないのが不思議。
○かつて強かったということもある。
○技から学ぶとか見て学ぶという伝承方法。
○オランダの天然ガスのプラント会社では、毎回入札をしていたが駄目で、５社に限定して 10 年くらい一緒にやると非常にコストが下がり、エラーも少なくなって、とても良くなったと主張していた。お互いの良いところを採ろうという方向に力が掛かるのだと思う。日本でかつてしていたことではないか。
→（田名部氏）欧米から日本は研究され、学ばれている。日本はというと「阿吽」でやってきた。知識化されたものが逆輸入されるのではないかと思う。

・ミニワークショップ

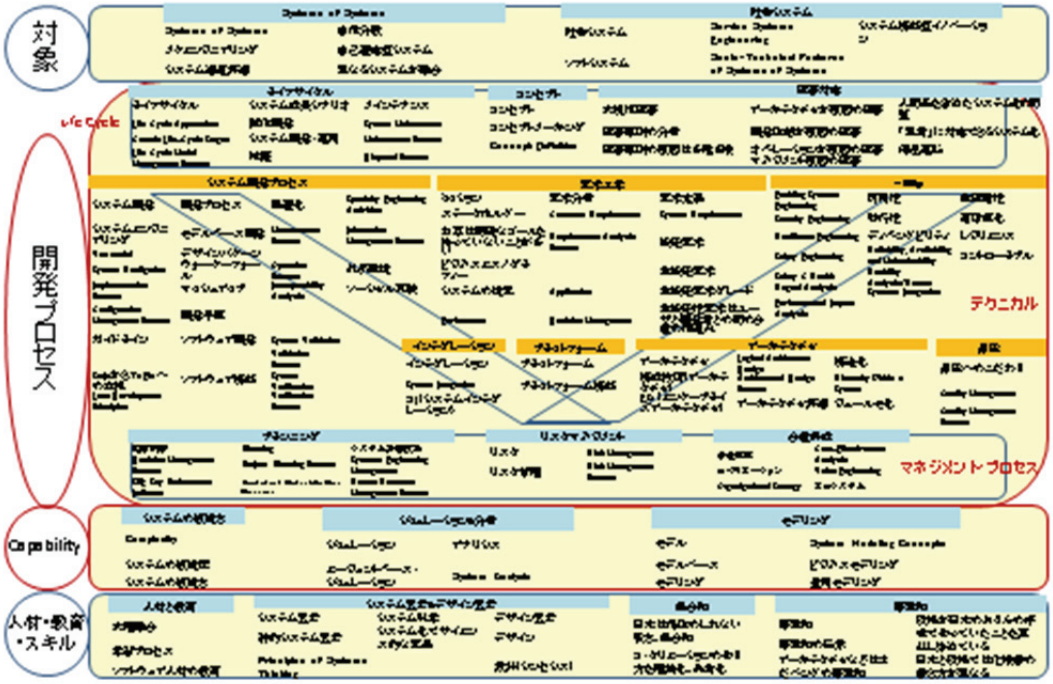
参加者が２チームに分かれ、これまでの４回の検討会での議論に現れた KW（キーワード）及び関連書籍等から抜き出したシステム関連の KW 群から重要な KW の選出とマッピングを行い、現状のシステム科学技術分野の構造を俯瞰した。

ミニワークショップの結果、各チームの抜き出した KW とマッピング結果が以下の俯瞰図である。

Aチーム俯瞰図



Bチーム俯瞰図



以　上

付録９．第６回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 26 年 4 月 25 日（金）　15:00～17:30

場所：JST 東京本部別館　2 階会議室 E

［プログラム］

１）配布資料確認、前回議事録確認

２）講演

・ソフトウェアと競争戦略：競争戦略論の視点から
　立本博文　筑波大学ビジネス・サイエンス系准教授

３）システム KW マッピング作業 WS 結果の報告

・チーム別俯瞰図素案

・俯瞰図素案の精緻化

４）おわりに

・今後についてと次回の確認

２．資料

・ソフトウェアと経営戦略：競争優位の観点から（立本氏）

・戦略的標準化　ー国際標準化の戦略的活用―　（立本氏）

・オープン・イノベーションとビジネス・エコシステム：新しい企業共同誕生の影響について（立本氏）

・A チーム俯瞰図／B チーム俯瞰図（主査）

３．議事メモ

・ソフトウェアと競争戦略：競争戦略論の視点から（立本氏）

質疑、議論

○Intel のプラットフォームを戦略の有効性としてお話されていたが、そもそも、Intel だけにその技術があったので、部品を売るために必然的にプラットフォームを作らざるをえなかったのではないか。他ではできなかったのでは。

→（立本氏）技術があったことはもちろん前提条件ではある。Intel の方に言わせると CPU バススピードが足りなくなるから共同的にやりましょうといってコンソーシアムを創っている。ただ、元々パソコンを造っていた IBM やコンパックは反対した。チップセットはもともとパソコンメーカが差別化していた部分。何で Intel がそこをやるのか。これによって PC メーカは差別化ができなくなった。マルチメディア用の周辺機器まで囲い込む作戦がとれなくなった。後の歴史の話だと技術からという面からしか見えなくなるが、本当はいろんな選択肢があってその中でその方向へもっていたことが成功につながったことは間違いがない。同じことが自動車メーカと Google の間で行われている。外部情報をつなげる必要が

あることは分かるが、なぜ Google なのか。技術的なことだけでなく、（産業構造や各社の技術レベルなど）全体を見ないとならないと思う。

○最終的には技術を持っていたところができることで、その他に理由はあるのか。

→（立本氏）例えば、ディーラもメーカもインターフェイスを作るのは止めた方がよかった。工場投資ができない日本メーカは技術と経営の判断をつなげて考えるべきであった。次に、このような問題を、会社の中でどこと話すのかという問題がある。技術統括部であろうか、経営企画であろうか。どちらだけでも駄目で、二つ合わさって決めて行かなくてはならない。しかし、残念ながら現在の典型的な日本の企業組織は、このような（システム全体を俯瞰するような）問題を扱うようにはなっていない。

○このプラットフォームは、意図して発生したのか、自然発生なのか？経営としてマネジメントできるものなのか？また、アメリカばかり勝って、日本ばかり負けているのか？それはなぜか？民族性か？

→（立本氏）プラットフォームは、アメリカしかないわけではない。ヨーロッパでは、ボッシュが成功させている。ここは特殊な組織構造をもっている。また、日本でもエアコンメーカのダイキンはできている。ただ、圧倒的に、日本企業が（プラットフォームビジネスが）下手なのはその通り。なぜかというと、事業部制をとっていて、部品部署と製品部署どちらもがんばって利益を上げようとする。プラットフォーム事業においては、損する部署があっても良いという考え方に立たないと、成功できない。日本企業はプラットフォームができそうなネタはたくさん持っている。だが、全部自社でやろうとしてしまうため、こういう戦略には不得意。次にプラットフォームを作るにはどうやって相手に言うことをきかせるかということがある。標準を作るときは交渉が発生する。この核には特許があるが、日本では、特許を扱う知財部署は先端技術を扱う中央研究所の下などにあり、本部と離れている。特許を使って経営戦略を立てようとするときには、本社の近いところに無くてはならない。（最近）その傾向になってきているが、その方向にシフトするのに遅かった感がある。日本はアメリカ、ヨーロッパとこの部分で違っていると思う。

○視点が少し違うかもしれないが「複雑化」は必然ですか？大きく見たときに、多少快適になったと思われるが、誰が複雑化しているのか？そして誰が得をしているのだろう。複雑化させないという動きがあっても良いと思い始めて来た。ユーザが求めているものは、複雑化ではなく、サービス向上とか快適性だと思う。

→（立本氏）確かに複雑化は手段であって、それを求めているものではない。

○サービスはつながっていくと必然的に複雑化する。

○サービスはリニア。複雑化はそれをこえていませんか？

○サービスは組み合わせで２のｎ乗になるのだろう。

○戦略論アプローチについて伺いたい。ある意味大きなケースに合致するような企業ケースを持ってきて説明するといったことが多いが、そこに理念はあるのか。エンジニ

ア的な発想からするとどうすればそれを利用できるのであろう。例えば、クローズなところで利益を得るにはといった解を導けるのだろうか？我々がデザインしてシステムを造っていくとき、安全などを考えるが、それを含んだ戦略として示唆はあるか？

→（立本氏）典型としてベストプラクティスの会社を見つけ、そのメカニズムとして成功にいたるファクターを拾う。また、あまりうまくいっていない会社を持ってきて比較する。判っている人はファクターの中で予測はできるけれども動かせるところと動かせないところを分ける。

たとえば、こういうケース研究の積み重ねから、プラットフォームビジネスでは、オープン標準の作り方が重要であると言うことが分かってきた。プラットフォーム企業にとっても、周りの企業にとっても、どのオープン標準の作り方で、オープン標準を作るかの整理が必要。コンソーシアムでオープン標準を作る場合、独禁法のしばりがあり、後から参加企業が来るところを拒むことができない。標準化のコントロールができるのは、（コンソを）設置する直前と直後。この時の出自の産業を見ればだいたい（方向性の）予測ができる。発起人になることが重要で、特許が企画に含まれる処理の仕方が重要。

○合意形成の部分。戦略とオペレーションがつながるかつながらないかを知りたい。

→（立本氏）10 年くらい前はミンツバーグという有名な先生が、創発的な戦略について強調していた。彼は「日本企業はほとんどすべて創発戦略だ」と言っていた。しかし、最近私は、それではいけないのではないか、と思っている。すべてを計画するのは難しいが、必ずいつかはやってやろうと戦略意思を持っていて、それに備えている。そして、機会が来たら、その戦略意思をできるときにやるものなのでは無いかと思う。これは、ミンツバーグがいっている創発戦略とはずいぶんと違うのではないかと思う。

○両方あると思う。イノベーションもリソースとケーパビリティの両方があるからこそ強く出られるから戦略としてやれるという部分がある。

→（立本氏）上と下でデシジョンの行き来がある。良い現場を持っていたら良いものができる、というのは無責任。良い経営陣がいないと無理ではないか。

○戦略に従って動かないという現実もある。

○Intel はプラットフォームビジネスに持っていきたいと意図してやったのか？たまたまか？

→（立本氏）前者です。明確に上層部の戦略的意思決定があった。組織も、この戦略に沿って再編した。Intel のチップセット部門と CPU 部門は、元々は別組織であった（CPU は半導体部門、チップセットは周辺機器部門）。プラットフォーム戦略後は、チップセット部門と CPU 部門を統合した。

○それも、プラットフォームビジネスを意図してのことか？

→（立本氏）そのとおり。たまたまではない。

○ARM も（Intel と同じような事を）やっていた。それは、戦略としてとらえて良いと思う。

→（立本氏）戦略的という点で、もう一つ重要な点がある。Intel は重要な拠点を台湾に持ってきた。それは、売り先が台湾であることを見越したことで、やりながら見えた話と思うが、チップセットまで付けてあげないと台湾では売れないという判断によるもの。

○それも、技術のある意味必然性と思う。チップセットにいち早く目を付けて売れるようにした。戦略もあったと思うが、日本の実力が弱かった。

→（立本氏）90 年代のチップセットは、パソコン業者とベンチャー企業で作っていた。

○スピードが高くなって必要になってきたし､それに対応するのは難しいものであった。日本は、技術は強かったがビジネスモデルで負けた、という言い方はひっかかる。技術力が負けた（のだ)。

→（立本氏）その話で代表的に出てくるのが、半導体産業、とくに DRAM 事業である。時代と一緒に見て行かなくてはならないと思うが、DRAM でいうと 98～99 年頃は、日本はプロセス技術では勝っていたが、その後、設備投資しなかったため製造技術で競争優位を維持できなくなった。

○良い物を作る技術はあったかもしれないが、安く作る技術が無かった。そこは難しいところでもある。

→（立本氏）そういうことは、逸話的に良く言われているが、冷静に科学的に検討した方が良い。DRAM 事業で日本企業が凋落したプロセスについて、私はいろいろ企業にインタビューしたし、不完全ながらデータも集めた。そこから総合的に考えると、そういう逸話的な話は違うのではないか、と思っている。

○安いものを作るのが好きじゃなかった。

○メインフレームメーカと半導体メーカが同じでメインフレーム用に技術を伸ばしたということもある。

→（立本氏）そういう話もあるかもしれないけれども、それらはすべて日本企業の内部事情の話だ。そういう風に日本企業の駄目なところを指摘するというのは、良くやられている。しかし、それは、競争相手に負けたという話とはかなりギャップがある。完璧な企業というのは、どこにも存在しない。駄目なところを見つけようと思えば、どんな企業にも見つかるものだ。しかし、そのような欠点が、競争相手に負けた原因であるかどうかは、別の話だ。たとえば、サムスン電子が安く作れる技術の開発に成功したから、日本半導体産業に打ち勝ったという主張は良く聞く。日本は高いものを作る技術しかなかった､という風に言われている。しかし、サムスン電子の半導体技術の基盤は日本と同じだ。（当時）韓国国内に半導体製造装置産業はなかった。日本と米国から大量に製造装置を輸入していた。日本からエンジニアも大量に韓国に行った。

サムスン電子が DRAM を安価に供給できたのは、むしろ、設備投資の規模の巨大さが貢献していると思っている。規模の経済である。日本の半導体企業の失敗は、適切なタイミングで大規模投資ができなかったことであり、これは技術の

問題ではなく、投資戦略の問題だと思う。もちろん、この問題を、戦略的に安価なものを作るという技術経営の問題としてとらえることもできる。しかし、そのときのベンチマーク相手はサムスン電子ではない。先進国企業で、DRAM を安く作る技術を開発して生き残った会社がある。それはアメリカのマイクロンだ。マイクロンはメインフレーム向けではなく、パソコン向けに特化した DRAM 設計をしていた。マイクロンは、日本の最後の DRAM 企業のエルピーダを買収した会社だ。もし本当に、安く作れる技術の技術開発戦略を知りたいならば、マイクロンを調べるべきだ。また、エルピーダ設立前と後もはっきりと分けて考えるべきだと思う。私はエルピーダになってから投資戦略や技術経営の問題はずいぶんと改善したという風に思っている。日本の半導体産業をすべて同じように語るのは間違っていると思う。

いずれにしても、日本の半導体産業の凋落については、さまざまな人がさまざまなことを言っているが、それらが科学的な根拠に基づいて主張されているのか、もっと注意を払う必要がある。

○システム的な観点から言うと、ものづくり、製品にこだわってきたといえる。（日本企業の問題は）ビジネスシステムというものを描けるかということ。ドコモはエコシステムを国内で作っていた。囲い込み戦略に走り、オープン戦略に行けなかったため Apple に負けた。ビジネスモデルが作れなかった。

→（立本氏）i モードの話は、私も調査した。国内でエコシステムができていたが、世界で失敗したのはなぜか。2002 年、ヨーロッパに持っていこうとして、２兆円損失を出している。何でだろう？ブランドもできていてベストなコンディションであったのに。インタビューから浮かび上がったのは、事業部の壁の弊害であった。国内の事業部の形は、海外にプラットフォームを広げるときには、必ずしも適しているとは限らない。変えた方が良かったかもしれない。

○何でそこに気が付かなかった？

→（立本氏）プラットフォームという発想が無かったということと、組織だと思う。

○確かに事業部制だと独立採算。

→（立本氏）出資した２兆円は、すべてマイナー出資であった（注：メジャー出資でないと、結局、通信オペレーション事業を、自分の意思で行うことはできない）。メジャー出資ではないと、オペレータ事業のキラーコンテンツとして、i モードをやる意味が無いと思う。当然、メジャー出資だけが正解ではなく、さまざまな形のビジネスがあり得た。そのような根本的な青写真が必要だったと思う。

○ビジネスモデルを持たなかった。

○日本のビジネスをグローバルに展ばすことは、そのまま持っていけば良いというものではない。日本の場合は既にインフラが整っている等、持っていくところによって違ってくる。

→（立本氏）国際的な人材を持っていなかったということが敗因であるともいわれている。Apple のやっていることは、ドコモと電機メーカの技術を合わせたもの。

だから、ドコモのビジネスモデルは世界で通用するものだった。ドコモはオペレーティング、端末業者を買ってもよかった。

○電機メーカはハードを売りたかった。そこの転換が日本企業ではできないという見方もある。IBM もハードを売っていたがソフトに転換した。Intel も CPU 事業としてはワイマックスに投資している。

→（立本氏）確かにビジネスモデルの価値転換ができるかできないかがポイント。

○同様に技術転換ともいえる。

→（立本氏）デジタルだからできる話。どこでもインターフェイスが切れる。物理的なものだと切れない。

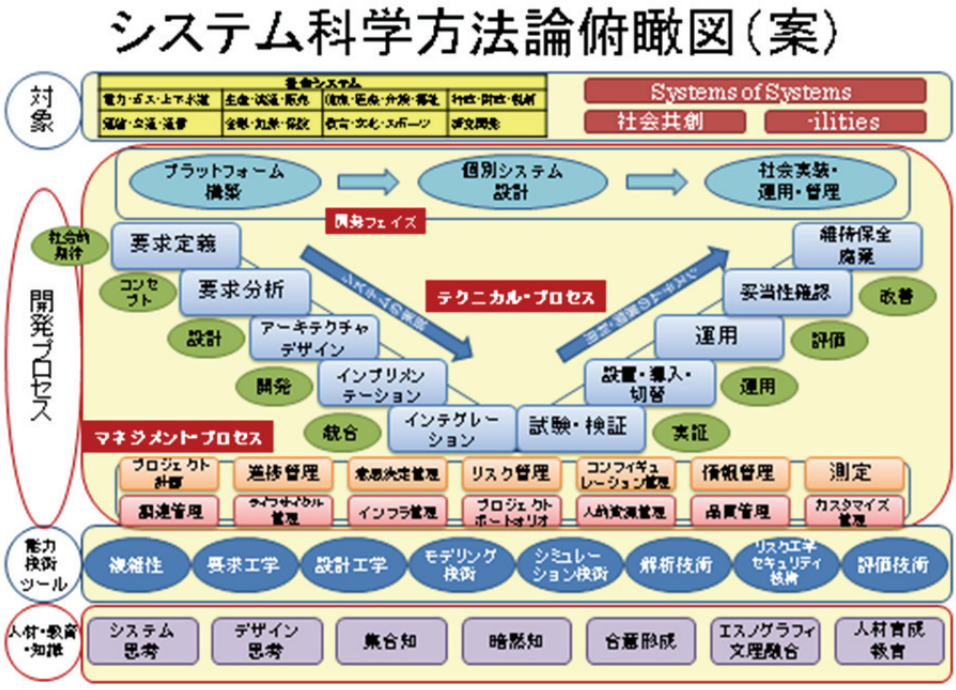
○（資料中の表で）2003 年以降に興味がある。新たなレイヤーやステークホルダが出てきているのではないか。

→（立本氏）クワルコムとか出てきている。レイヤーが違うがスタイルは同じ。クアルコムは利益２兆円の内、半分の１兆円がロイヤリティ、特許収入である。技術ではなくて、ビジネス構造の特許。

○難しい議論だと思うのは、通話料やコンテンツで儲けるといったときにどこに共通の源泉があるのか、売っていくものはどこなのか。Apple はプラットフォームを作って回線業者を土管屋さんにしてしまった。中は何を通しても文句は言わせないといった。ドコモは必然として負けることだったのか。分社化するしかなかったのか。

・システム KW マッピング作業 WS 結果の報告

前回マッピング作業の結果をもとに議論を行った。最終的には、「システム構築方法論」の領域俯瞰図としてまとめ、その中からさらに深堀する領域を抽出した。



資料 10. 第７回システム科学検討会議事メモ

１．開催日時・場所・プログラム

日時：平成 26 年 5 月 9 日（金）　15:00～17:35

場所：JST 東京本部別館　4 階会議室 A

［プログラム］

１）配布資料、前回議事メモ確認

２）講演

・SEC を通した技術移転－形式手法の場合－

新谷勝利　新谷 IT コンサルティング（IPA-SEC 連携委員）

３）システム構築方法論俯瞰図について

・システム構築方法論俯瞰図（案）

・研究開発領域候補、執筆協力者候補

４）おわりに

・今後についてと次回の確認

２．資料

前回議事録

資料１　SEC を通した技術移転－形式手法の場合－

○ラーセン教授を迎えて（新谷氏）

○形式手法の実践に対して良く尋ねられる質問とその回答（栗田氏）

○モデル規範型形式手法 VDM と仕様記述言語 VDM++

－高信頼性システムの開発に向けて－　（栗田氏、荒木氏）

資料２　システム構築方法論俯瞰図（案）

３．議事メモ

・SEC を通した技術移転－形式手法の場合－（新谷氏）

質疑、議論

○（シート 16 の）折れ線グラフの縦軸は何を意味しているのですか？

→（新谷氏）人月です。

○形式手法で仕様書内の無矛盾性をチェックできることは分かりました。例えば、それぞれのところでは正しい言明が書かれていたとして、同じ仕様書の中に何ら関係がない二つの言明が書かれていることまで分かるのですか？

→（新谷氏）それは形式手法でも答えが出るものではない。

○それを発見するには、仕様書を形式言語にしたものを図式化して見ると分かると思うが、可視化のツールはありますか？

→（新谷氏）自然言語と VDM のツールとしてパブリックにはクラス図を出すことができる。

○逆に最初から図を作って形式言語化するとか、両方向あると良いと思う。
→（新谷氏）荒木先生の初期の研究にそういうのはある。正しいか正しくないかは比較において言えるものであり、もう一つ上の記述というか抽象度を上げることになると思う。
○冗長性は分かると思う。２つ定義があって、片方が使われていないということは、それは無駄な定義であるということ。
→（新谷氏）大学では研究に使っておられますか？
○形式手法を（システム開発に）使うにはハードルが高くて実用上はまだ。小さい範囲で試すことはしているが。記述がある決まりに基づいて書かなくてはならない。決まりを知らなくてはならないし、それに基づいて書きづらいところがある。形式手法は検証できる範囲とできない範囲が明確にある。それで書かなくてはならないとなると、書く人も見る人も分からなくてはならない。どう書くか、書いたものがそもそも正しいか、実用上すぐにはできない。FeliCa のように分かる人が何人も集まっていれば良いが。もう少し自由度の高い言語でさえ、使えていないのが現状。
→（新谷氏）結局、形式手法は語彙を作ることが基本になる。（スライド 32）事例が集まったのは、交通機関を専門にする会社があり、そこで溜めたドメインに関するノウハウが使えるようになって行ったからだと思う。
○形式手法は数学的な証明可能な手法なのか。それとも単に基本を決めておいて、動くものということなのか？
→（新谷氏）後者である。証明にはかなりの数学的能力が求められるが、現実ではそれはほとんどされていない。FeliCa でも証明はしていない。SONY と協力会社でチームを作り、その中で実際の仕様を書くのは数名。シナリオを想定しながら書くことが重要。残りの人は読めれば良い。ハードルが高いというのは、最初からすべての人が書けなくてはならないと思われるからだと思う。実際には数人が書いて、書き方を決めて読み方のフレームワークを作って、そのタームを定義して、記述スタイルの統一を行う。このフレームワーク作ったのは一人。この人は九大で形式手法のドクターを取った。コーディングレベルでなくて、もう少し上の設計者レベル、仕様書のレベルで、会社の中のコミュニケーションが取れることが重要。誤解無く作れるものとして同じ語彙で開発できるようにするためのもの。今の日本の多重開発階層では難しいかもしれないが、特定の開発会社と長年やっていくようなときには有効。
○後者であれば、ハードルの高さはまったく違う。
→（新谷氏）前者を狙うというのは、アカデミックな興味としてはおもしろいかもしれないが、産業では求められない。
○VDM やモデルチェッカーにつなげていける。SysML ではフリーで書かせるといけないので、そのままは使えない。使うにはルール化する必要がある。そういった意味では使いやすいものだと思う。
→（新谷氏）形式手法という言葉は使うのを止めようかという議論もある。

○形式手法というのは、上流工程のどの辺に相当するのか？UML より上流でしょうか、それとも同じレベルでしょうか？

→（新谷氏）UML とレベルは並行。

○UML とは何が違うのか？

→（新谷氏）UML で書いたボックスを見ると、まさしく形式手法的に書かれている。逆に形式手法で書いたものを UML の BOX 内で書けるということもある。複数の異なる手法とリンクができるということが重要。

○実際にプログラムを開発した経験から、要求仕様は最終的に取り扱い説明書になった。ユーザが使うし設計者もそれを見て作る。日本語とアルゴリズム、数式で書かれている。これを形式手法に置き換える必要がありますか？

→（新谷氏）FeliCa のチームでは、VDM で書いたものを日本語に直した。日本語から日本語へより形式手法から日本語に直す方が、誤解が少ない。

○UML より擬似コードに近いと思う。理工系であれば、論文は判断できる。

→（新谷氏）一行の要求仕様を数行のコードに変換するときにいかに誤変換を少なくするかが重要。

○論文は再現性が重要なので、それを見れば再現できる。それを形式手法で書けるのだとしたら素晴らしい。一つの言語では書けないので、アルゴリズムを説明する擬似コードや、SFC などのフローチャート形式であったり、シーケンス処理を記述するラダー言語であったり、組み合わせないとすべてを書き表すことができない。

→（新谷氏）VDM は日本語もサポートしている。

○「形式手法は万能ではない」という説明が少ないところにいつも疑問を感じています。何ができて何ができないのかという話をきちんとしてくれると納得性が増すと思います。

→（新谷氏）栗田さんの FAQ に明記されている。お時間のあるときに目を通して頂ければと思う。

○多くの場合、形式手法の世界（研究のレベル）で閉じているようで、現実と折り合いを付ける努力が少ないように思えます。難しいとは思いますが、フォーマル（形式）と現実との間の距離を縮める活動がもっとあっても良いのではないでしょうか。使いやすさを改善する工夫も必要です。

→(新谷氏)C 言語とか JAVA で何でもできますかと聞かれたときにどう答えるか。コンピュータは機械が理解できるように書かれていないと何もできない。VDM は中間言語。抽象度が高い。ということは、どこかで詳細さを落としている。組み込みの人が良くいうのは、タイミングの問題。

○「ここは形式手法の出番だ」とか「タイムクリティカルなことやイベントオリエンティッドのような部分は形式手法がまだ十分にアプローチできていない」というような説明があると良いと思います。

○確かにそういう議論があって、形式手法は非同期のシステムの記述には向いていないと言われている。不確実な事象を書くには向いていない。ではどんな言語が向いてい

るかといわれても難しいが。

→（新谷氏）いかに対象をモデル化するかが重要な点である。残念ながら「これはできる、できない」と言えるほど経験は溜まっていない。

○UML との違いは、前提条件と事後条件を必ず書きなさいという風に決めているところだと思う。

○形式手法は好きなのですが、限界を感じていて、今、情報システム分野に身をおいています。ソフトウェアを数式で書くことをやっている。抽象化したときの意味付けとういことをもっと柔軟にできる。直接、手法をどうとらえてマップするかというとき、数学的手法を使うことによって別の意味が見える。システムの同型性を見ることができる。UML でも同定できるが、数学より強くないと感じている。マークアップ言語（ML）は証明できない。限定して証明しないとチェックできない。厳密性を求めると絶対に成り立つメカニズムを証明できるが（例えば軍のシステムやセキュリティシステム）、一方でそこまで厳密でないときの強みは抽象理論に持ち込んで柔軟性があることだと思っている。

→（新谷氏）まさに SEC で研究しているが、上流の問題は未解決のところもある。ただ、関係者間のコミュニケーションに強みがあると思っている。

○上流工程でのコミュニケーションの問題は、情報システムでも見ていて、仕様をまず書くというところから始めても駄目で、ある程度まず造って、これではないということを確認して･･･、と進めることが良いという風に考え方が変わった。どうインタラクションが生まれ･･･、といったことは、デザインしながら共有しながら確認していくしかないと思う。仕様といった命題的なものから入るとその場その場になる。ナラティブを集めると、要求仕様に入らない文脈が見えてくる。

○話が少し飛びます。組み込みの世界では「アジャイル開発プロセス」を活用することが多いのですが、それが可能なのは比較的規模が小さいからなのではないかと思っています。規模が大きいものだと最初からすべてアジャイルでというのは難しいので、それに代わるものとして「プロトタイプ」を造るといったことをしています。ナラティブで仕様を確認するといったところは、それと似ているところがあると思いました。

○認知科学的にストーリー展開の方が認知しやすいとされている。ただ、ソフトウェアは仕様に落とさないといけない。

→（新谷氏）FeliCa の方々に今度聞いてもらうと良いと思うが、彼らも開発において、仕様の成長をさせている。イテラティブに繰り返しを行なっている。VDM は、記述言語なのでこういったことが可能になり、また仕様が実行できる。実際に試さなくても仕様書の段階である程度分かる。

○Haskell などの関数型言語が研究されるべきと思う。

→（新谷氏）そういった研究は名古屋を中心として進められている。SONY の場合でも三世代やって学習してきた。企業の社風にも依ると思うが、これによりドクターを３名も出している。そういうソフトウェアはあまりない。ドクターを作らないと形式手法は使えないという誤解を与えてしまうのは不本意だが、エンジニ

アも形式手法を導入していく上で勉強しなくてはならない。

○ソフトウェア的に形式手法が良いなと思うのは、最初から仕様ですべて決めることは当然無理で、その中で、決められる範囲はどこなのかということを決めることができるところだと思う。

○他の言語でもしていると思うが、形式手法ではその範囲を保証できるというところ。

→（新谷氏）ツール化といったところがあるが、自然語を使わざるを得ないところでどこまで誤解を少なくできるかだと思う。EU の FP7 でも検討されている。

○人材育成は必要。人が思っていることと仕様を一致させることはできないが、その差を埋めていくもの。

→（新谷氏）結局、産業として成り立つためには専門家がいる。私が形式手法を学び始めたのは 1984 年。当時 IBM では、ハードからソフトウェアへの転換がなされ、開発エンジニアはソフトウェアエンジニアリングに基づき仕様を書けなくてはならないということになった。企業は人材育成をしなくてはならないと思う。

○ありがとうございました。ソフトウェアの使うレベルと使われるレベル、妥当性のレベルとあり、これらのケースで使われるモデルベースのようなものと理解しました。

・システム構築方法論俯瞰図（案）について（主査）

○アジャイルの位置づけは？

→（主査）「システム開発技法、形式手法」に含めているつもり。

○アジャイル自体は手法ではなくもっと大きい。SS（ソフトシステム）とも近い。

○テクノロジーで分けるのかアプローチの対象で分けるのか。世界の見え方がそれぞれ違うような気がします。

→（主査）共通に使える技術という感じで書いてもらいたい。特定の対象というよりはできるだけ抽象化したい。

○抽象化した世界と具体の世界の両方の記述が必要で、それを踏まえた上で両者のつながりを明らかにすると良いのではないでしょうか。

○コンセプトを作るところと実現をするところを明確にすべきと思う。コンセプトエンジニアリングといって、その中でモデルベースなどを検討する試みがオーストラリアを中心に発達してきている。この図でこれはどこに置けるだろうと思うと難しい。われわれが研究しているところはニーズから入ってピボットを行い、社会ニーズを把握するという流れ。システムは複合的であるからこそ難しいと改めて思う。

○開発プロセス（フェイズ）とは異なる軸が必要かも知れません。例えばレビュープロセスはこの中では語れないような気がします。

→（主査）要求定義の小さなループになるのだろうか。

○領域はこんなに多いのか？

→（主査）実際には領域の重要なところを中心に 7 つほどを書いてもらう予定。

○V モデルではなくて W モデルに近いのでは。

○クオリティーマネジメントという一つの軸が貫いているイメージ。その他、セキュリ

ティや･･･、と考えるとｎ次元になってしまう。

→（主査）ここではシステムのライフサイクルと社会実装から基礎技術の軸の２次元を考えている。

○この俯瞰は産業で使えるというよりはアカデミックな俯瞰である。ディスプリンにまではなっていないが研究領域があって、論文はあるだろうというイメージ。難しくて前回の俯瞰では深堀ができなかったところ。ただ、システム科学技術分野の中でこれが一番重要な位置を占めている。

→（主査）（俯瞰図の）下から２段目くらいは書きやすいと思うが、ここの部分だけではあまり意味がない。ライフサイクルの部分も掘り下げたい。

○デザイン絡みのところで俯瞰ができるかどうか。

○難しい。

→（主査）社会からシステムに落とすところ、できたシステムが社会に理解してもらえる部分、Ｖモデルのところ。この辺を埋めたい。

○大きな見方はいろいろある。人工物システムをどう造るのかという風に考えるのではなくて、造っている人たちもシステムの一部だという認識で人々が関与しているシステムとして捉えなくてはならない。その観点から、人材育成や経営を変えていくことが派生すると思う。エンジニアリングデザインモデル、ユーザモデル、･･･等。求められているのは、世の中のシステムをどうメンテナンスしていくかではないか。大きい話だが。

→（主査）コンセプトレベルだけの議論になってしまっても困る。

○どの側面から見ているかが見えない。ただ、３次元以上になると人は認識できなくなるというから、難しいところ。

○われわれが扱うシステムは人工物であるということは明確にしている。

→（主査）ぐるぐる回るイメージでしょうか。

○平凡なものを狙いたい。専門家以外の方にも納得してもらえるような。この俯瞰を参照する対象者は、専門家も一部含まれるかもしれないが、専門家でない方々。

→（主査）領域俯瞰をしてみて、この辺にファンドをした方が良いという見方に使う。

○やはり今のままで詳細記述をするのはかなり難しいですね。社会システムやインフラシステムを構築してきた経験からいうと、世界のとらえ方が重要で、そこが不十分なままで部分を切り出して詳述しても全体の整合性が取れないように思います。見てもらう相手に何を伝えるかが重要なのかも知れません。

→（主査）研究開発領域を見つけたい。最終的に学問になりそうなところを出して行きたい。

○他のユニットのナノテクなどは皆が納得できるような上手いものが書けている。システム科学技術についても理解してもらえるものを目指したい。

○「ソフトウェア工学は工学ですか？」ということを最近考え始めています。機械工学、制御工学、電子工学など、基盤となる科学原理があり、物理現象があります。ソフト

ウェア工学にはそういうものが無いところがほかの工学と異なるところではないかと思っています。

○自然とは別にちゃんとした「岩盤」があると思わなくてはならない。日本人の悪いところだと思うが、手触りの感覚が無いものは無いのだということはない。

○そうですね。現状無いから駄目だといっているのではなくて、それに代わる「岩盤」を見つけましょうということです。その努力をせずに「工学」という言葉を安直に使うことに警鐘を鳴らしたいと考えていることも確かです。

○無いことはないと思う。でも、工学（エンジニアリング）だと思う。ただ、再現性のあるサイエンスは無いかもしれない。

○科学的なこうやれば上手くいくというものはないが、経験上、こうすれば上手くいくかもといったことは蓄積できる。

→（主査）この図について、社会と開発プロセスを中心に抜けているところやまたこの先生がいるという情報を寄せてもらいたい。さらに、この先生との間を取り持てるといった情報を。ただ、すべての先生に原稿をお願いする訳ではない。まずヒアリングをさせてもらうところからお願いしたい。

以　上

■ワークショップ報告書作成メンバー■

木村　英紀	上席フェロー	（システム科学ユニット）
鈴木　久敏	フェロー	（システム科学ユニット）
富川　弓子	フェロー	（システム科学ユニット）

※お問い合せ等は下記ユニットまでお願いします。

CRDS-FY2014-WR-08

ワークショップ報告書

システム構築方法論
ー現在の到達点と今後の課題ー

平成 26 年 10 月

ISBN978-4-88890-412-4

独立行政法人科学技術振興機構　研究開発戦略センター　システム科学ユニット
Systems Science Unit, Center for Research and Development Strategy
Japan Science and Technology Agency

〒102-0076　東京都千代田区五番町 7　K's 五番町
電　　話　03-5214-7481（代表）
ファックス　03-5214-7385
http://www.jst.go.jp/crds