JEITA

IS－15－情シー２

平成26年度 ソフトウェアに関する調査報告書 Ⅱ

組込み系ソフトウェア開発の課題分析と提言

～モデリングはなぜ失敗するのか～

２０１５年３月

一般社団法人 電子情報技術産業協会
ソフトウェア事業委員会

序

IoT（Internet of Things, モノのインターネット）の進展やネットワーク化された多様なサービスの展開、クラウドコンピューティングの到来により、企業の情報システムから多種多様な組込み機器に至る広範な分野でソフトウェアの重要性は益々高まってきており、我が国のあらゆる産業を支える中核技術として位置付けられるまでになっている。当協会の統計によれば、国内主要ベンダーによるシステム開発、ミドルウェア、アウトソーシングなどのソフトウェア及びソリューションサービスの国内市場規模は、2013 年度で 5 兆 6,145 億円[1] になっており、これは前年度から 1 割増加している。

このような状況で産業として成長し日本企業のシェアを維持・拡大するために、現代社会及び産業を支える中核技術として位置付けられるソフトウェアの果す役割への期待が高まっている。この中で、開発規模の増大や複雑化への対応、開発コストの低減、品質の確保や産業を支える高度人材の育成など、ソフトウェア業界が抱える共通の課題も依然として多い。このような状況の中で、当協会のソフトウェア事業委員会では、いくつかの専門委員会を設け､我が国ソフトウェア産業の国際競争力強化やソフトウェア事業基盤の強化、開発者の育成などの業界共通の課題を抽出し、必要な施策を提言するなど、ソフトウェア業界の更なる活性化に向けた活動を積極的に推進している。本専門委員会では、この中で我が国の強みの源泉であり価値創出のキーであると言われている「組込み系ソフトウェア分野」に焦点を当て、この分野でのソフトウェア開発力の現状把握及び基盤強化を図るための取組みを 2005 年度から行ってきた。2005 年度から 2007 年度までの 3 年間は、ものづくりの根本である「品質確保」の視点から、組込み系ソフトウェア開発の実態調査と課題分析、提言活動を行ってきた。続く 2008 年度から 2010 年度の 3 年間は、日本の力をアクティブに発揮できるものとして、「開発スピードアップ」を視点に据え活動を行ってきた。そして 2011 年度から 2013 年度はこれまで活動してきた中で大きな影響を与えていたアーキテクチャ設計を作るアーキテクトをテーマにして活動してきた。

2013 年度までテーマに活動をしてきたアーキテクトであるが、このアーキテクトの仕事の一つでもあるモデリングが問題になっていることが分かった。そこで 2014 年度から数年間に渡る活動として､組込み系ソフトウェア開発におけるモデリングをテーマに決定した。このテーマでワークショップを開催し、また有識者を招いてブレーンストーミングを行った。活動内容として 2014 年度は開発現場におけるモデリングの現状や事例などの調査を行い、モデリングの課題やモデリングが失敗する要因やその対策などをまとめたので、報告する。

本報告書は、上記のワークショップの議論やアンケートによる調査、ブレーンストーミングの分析結果と当専門委員会の 2014 年度の活動内容を取りまとめたものである。この報告書が各方面に広く利用され、我が国の組込み系ソフトウェア分野の発展に寄与できることを祈念する次第である。

2015 年 3 月　　一般社団法人　電子情報技術産業協会

---

[1] 2013 年度ソフトウェアおよびソリューションサービス市場規模調査結果について(JEITA), http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=706&ca=1

ソフトウェア事業委員会名簿

（敬称略・順不同）

委員長	嶋津和行	日本電気㈱
委員	谷本宏昭	沖電気工業㈱
〃	五味　弘	沖電気工業㈱
〃	深谷哲司	㈱東芝
〃	苗村健二郎	東芝ソリューション㈱
〃	篠原郁二	日本電気㈱
〃	服部　徹	日本電気㈱
〃	白井克昌	㈱日立製作所
〃	戸井哲也	富士ゼロックス㈱
〃	市川俊樹	富士ゼロックス㈱
〃	松本　学	富士通㈱
〃	前川隆昭	三菱電機㈱
〃	河内浩明	三菱電機㈱
事務局	稲垣　宏	(一社)電子情報技術産業協会
〃	内田光則	(一社)電子情報技術産業協会
〃	石川　淳	(一社)電子情報技術産業協会

ソフトウェア事業基盤専門委員会名簿

（敬称略・順不同）

委員長	五味　弘	沖電気工業㈱
委員	金子　博	㈱東芝
〃	三友雄司	東芝ソリューション㈱
〃	奥山嘉昭	日本電気㈱
〃	大橋浩幸	㈱日立製作所
〃	市川俊樹	富士ゼロックス㈱
〃	竹田和正	富士通㈱
〃	小林英将	富士通㈱
〃	三橋　崇	富士通㈱
〃	田村直樹	三菱電機㈱
オブザーバ	春名修介	大阪大学
〃	谷口真也	セイコーエプソン㈱
〃	牧　隆史	㈱リコー
〃	山田大介	ビースラッシュ㈱
〃	福嶋慎一	㈱日本データコントロール
〃	石井正悟	情報処理推進機構（IPA）
事務局	内田光則	(一社)電子情報技術産業協会
〃	石川　淳	(一社)電子情報技術産業協会

目　　次

1. 目的 1
2. 本報告書の想定読者 3
3. 2013 年度までの活動成果と 2014 年度の活動 3
3.1 2012 年度までの活動 3
3.2 2014 年度の活動 6
3.2.1 活動の狙い 6
3.2.2 活動内容 6
4. モデリング技術の位置付け 8
4.1 ソフトウェア・システムのモデリング 8
4.2 モデルを利用したソフトウェア・開発の技法と位置付け 12
4.3 アーキテクティングにおけるモデリング技術の適用 19
5. 組込みソフトウェア開発現場でのモデリング適用上の課題 23
5.1 制約や品質要求がモデル化されていない 24
5.2 モデリングスコープに対応したモデル評価基準が設定されていない 25
5.3 仕様の検証や妥当性確認にモデルを活用しにくい 36
5.4 新規開発技術導入の目的を明確化していない 37
5.5 ソフトウェア資産再利用や共通プラットフォーム化のモデル化技術が未成熟 39
6. モデリング事例に見る課題の考察 41
6.1 ワークショップ事前アンケート 41
6.1.1 モデリングの定義について 43
6.1.2 モデリングに必要なスキルについて 43
6.1.3 モデリングの効果について 43
6.1.4 良い／悪いモデルについて 43
6.1.5 モデリングの失敗／成功要因について 44
6.1.6 モデリングの課題について 44
6.1.7 モデリングへの期待について 44
6.2 モデリングの失敗事例 45
6.2.1 モデリング導入目的の誤りについて 46
6.2.2 モデリングという道具（ツール・技法）の理解不足について 46
6.2.3 モデリングの道具（手法・ツール・環境）の能力不足について 47
6.2.4 モデラーの能力不足について 48
6.3 モデリングの事例 50
6.3.1 事例 1) Excel と Enterprise Architect を駆使したモデル設計現場適用 50
6.3.2 事例 2) MFP の複合機のコントローラと板金機械制御におけるモデル化開発 51
6.4 モデリング課題の考察 54
6.4.1 モデリング技術導入目的の曖昧さ 54
6.4.2 開発フェーズの目的と異なるモデルの作成 54
6.4.3 仕様の検証／妥当性確認にモデルを活用しにくい 55
6.4.4 新規開発技術導入目的の不明確さ 55
6.4.5 SW 資産再利用とモデルの扱い（基準・標準） 56
7. モデリングの適用に対する課題解決の考察 57
7.1 ワークショップ参加者アンケート 57
7.1.1 チーム・組織力向上について 58

7.1.2 設計力向上について 58
7.1.3 非機能要件・制約条件のモデル化について 59
7.1.4 再利用性の向上について 59
7.1.5 ワークショップ事後アンケートまとめ 59
7.2 モデリングの課題解決（複合的要因） 61
7.2.1 問題カテゴリのバランス 61
7.2.2 モデリングの良し悪し 63
7.2.3 アーキテクチャとモデリング 64
7.3 まとめ 64
8. まとめ 65
8.1 モデリング技術の現状 65
8.2 モデリングの課題認識 65
8.3 モデリング事例に見るモデリングの課題の考察 65
8.4 モデリングの適用に対する課題解決の考察 65
8.5 まとめ 65
9. 今後に向けて 67
謝辞 68
参考文献 69
付録 1. ワークショップ 2014 事前アンケートまとめ 75
付録 2. ワークショップ 2014 事前アンケート回答詳細 78
付録 3. ワークショップ 2014 当日アンケート回答詳細 86
報告 1. 組込み系ソフトウェア・ワークショップ 2014 91
報告 2. CEATEC JAPAN 2014 講演 98
報告 3. モデリング・ブレーンストーミング 99
講演資料 1. CEATEC Japan 2014 講演資料 101
講演資料 2. ワークショップ講演資料 1（五味氏） 113
講演資料 3. ワークショップ講演資料 2（春名教授） 116
講演資料 4. ワークショップ講演資料 3（奥村氏） 124
講演資料 5. ワークショップ講演資料 4（石田氏） 128
講演資料 6. ワークショップ講演資料 5（久我氏） 130
講演資料 7. ワークショップ公開問題（スマートガレージ） 138

報 告 書 要 旨

報告書要旨

1. 活動の概要

ソフトウェア事業基盤専門委員会では、2014 年度から「モデリング」という新たな視点から組込み系ソフトウェア開発の課題解決に取組んでいる。2013 年度までの過去 3 年間のアーキテクト調査からアーキテクチャ設計などの上流工程におけるモデリングを中心に、要求モデリング、ビジネスモデリング、設計モデリングなどでの課題やその解決策を探っていく。

2014 年度は 10 月初めに CEATEC JAPAN 2014 において昨年度までのテーマである組込み系アーキテクトに関する当専門委員会の取組みについて講演を行った。続く 11 月上旬には「組込み系ソフトウェア・ワークショップ 2014～モデリングはなぜ失敗するのか～」を開催した。このワークショップでは参加申し込み時に参加者に対してモデリングの実態調査や課題、対策などの事前アンケートを実施し、ワークショップ当日はモデリングの役割や課題、解決策に関する基調講演と事例講演を実施し、参加者の意見交換の場を設けた。また今年度は、モデリングに関して有識者を招いて、委員とブレーンストーミングを行った。これらの活動と委員による議論から得た情報を分析して報告書としてまとめる。

2. モデリング技術の位置付け

4 章ではモデリング技術の位置付けと最近の技術状況を紹介する。モデリング技術の位置付けとして、モデルに期待される特性や Zachmann Framework によるモデル化の視点を紹介する。次にモデリング技術として、図表を用いたモデリングや仕様証明を目的とするモデリング、シミュレーションを目的とするモデリングに分類し、最近のモデリング技術やソフトウェア・システム開発での利用方法を紹介する。またモデルと開発プロセスと関係を説明する。最後に、特にアーキテクティングにおけるモデリング技術を整理し、モデルと開発プロセスの関係を説明する。

3. 組込みソフトウェア開発現場でのモデリング適用上の課題

5 章では、実際に開発現場でモデリングを適用したときに現れる課題を提示する。組込み系の開発現場でのモデリングの課題は、制約条件や品質要求がモデリングされていないこと、モデルの評価基準が設定されていないこと、仕様の検証や妥当性確認にモデルを活用しにくいこと、技術導入の目的を明確化していないこと、また、再利用を想定したモデル化技術が未成熟であることを挙げる。これらの課題とその背景についての考察を説明する。

4. モデリング事例に見る課題の考察

6 章では各種のモデリング事例からモデリングにおける課題を分類し考察する。考察したモデリング事例としては､ワークショップの事前アンケートや講演でのモデリング事例、委員会で収集した事例がある。これらのモデリング事例からモデリングの定義や必要なスキル、モデルへの期待、失敗／成功要因、課題などのカテゴリで個別の事例を抽出し、それを分類してまとめている。これからモデリングにおける課題の傾向性として、モデリング導入目的の曖昧さや目的と異なるモデルの作成、アーキテクチャ設計の評価などを考察したので紹介する。

5. モデリングの適用に対する課題解決の考察

7 章ではモデリングを適用するときに現れる課題に対する解決方法について述べる。最

初にワークショップのアンケート事例からの課題を分類し、その中で性能などの非機能要件や制約条件のモデリングや再利用性の課題をピックアップして、その課題解決について考察している。モデリングの複合的要因による課題については、設計力や開発ツール、開発プロセスの三位一体による向上が課題解決の鍵になりそうだという考察を提示する。この考察を元に来年度はより具体的に解決への道を探っていく。

6. 付録

付録として、ワークショップの講演資料やワークショップのアンケート結果、CEATEC Japan の講演資料を付録として掲載する。

組込み系ソフトウェア開発の課題分析と提言

～モデリングはなぜ失敗するのか～

1. 目的

当専門委員会は 2005 年 4 月から活動を開始し、以下のような問題意識を持って、組込み系ソフトウェア開発の課題の分析とその解決に向けた提言などの取組みを行ってきた。

- 組込み分野は日本の強みの源泉であり価値創出のキーであると言われているが、組込み対象となるハードウェア機器は強いとしても、そのソフトウェア開発力が国際的に見ても本当に強いのであろうか。
- 「擦り合わせ」なるものが日本の開発力の強みと言われているが、急激に増大している開発規模や短納期化の状況の中で、現在でもそれが強みになっているのであろうか。
- 何を強くすれば、日本の組込み系ソフトウェア開発力の国際競争力を強化し、真に「日本の強みの源泉」たるものにできるのであろうか。

2005 年度から 2007 年度までの 3 年間は、「品質確保」の視点から、組込み系ソフトウェア開発の実態調査、問題・課題分析、提言活動を行ってきた。特に、2007 年度は 3 年間の活動のまとめとして、組込み系ソフトウェア開発に関する課題解決に向けての各社での具体的な取組み、特に「ハード部門などとの連携」「自動化」「上流工程重視」「多機種開発」への取組み内容についてアンケート調査を実施、この調査結果を基に取組み内容の分析と今後の方向性の検討を行い、総括的提言をまとめて公表した。

また、ドイツ Fraunhofer IESE 研究所（Institute for Experimental Software Engineering）と共同で 2007 年 7 月に IESE／JEITA 共同ワークショップ[1)]も開催し、当協会としては初の組込み系ソフトウェア関係者の意見交換の場を提供した。

このように、2007 年度までは、組込み系ソフトウェア開発を取り巻く大規模化・複雑化・短納期化・多機種化への対処施策について、社会的要請とも言える「品質確保」の視点からアプローチしてきたが、2008 年度からは、視点を変えて、開発側目線とも言える「開発スピードアップ」の新視点からアプローチして、2007 年度の総括的提言の検証と新視点からの深堀と提言の具体化を行うことにした。

2008 年度は開発スピードアップの最初の年度として、開発スピードアップを阻害する要因を事例とともに収集した。開発スピードアップの阻害要因について、ワークショップを開催し、パネラーや参加した人たちとディスカッションを行った。またこれを基にしてアンケート調査を行い、分析結果を報告書としてまとめた。続く 2009 年度は 2008 年度の調査結果で重要であると分かった要求定義とアーキテクチャ設計に絞り込んで、その阻害要因を深堀することにした。ワークショップとアンケート調査を実施し、これらを基に委員で議論を重ねた。組込み系ソフトウェア開発のスピードアップを阻害している要因について、様々な観点からアンケート調査を実施し、その分析を通じて阻害要因の実態把握を行った。2010 年度は阻害要因を 2008 年度の調査結果で重要であることが分かった要因の中で 2009 年度に深堀しなかったプロジェクトマネジメントに絞り込んで、その阻害要因を深堀することにした。また 2009 年度の要求定義とアーキテクチャ設計については深く検討を行った。この結果を 2010 年度の報告書で公表し、組込み系ソフトウェア関係者が情報共有できるようにした。

[1)] 2007 IESE/JEITA 共同ワークショップ：資料室
組込み系ソフトウェア開発の課題分析と提言 ～大規模化、短納期化、多機種開発にどのように立ち向かうべきか～http://home.jeita.or.jp/is/committee/software/070906/index.html

2011 年度からは、過去 6 年間の活動の中で議論をしてきた品質問題や開発スピードアップにおいて、重要な要因であることが分かった「アーキテクチャ設計」に注目した。日本の組込み系ソフトウェア開発の強みは、擦り合わせによる高品質開発にあると言われているが、大規模化・複雑化・短納期化・多機種化の潮流の中で、国際競争力を今後維持していくことが難しくなってきており、このような状況への対処として、欧米流のアーキテクチャ設計の導入・強化が注目されている。このアーキテクチャ設計を行なうためにはアーキテクトが必要であるが、欧米流のアーキテクトの手法を模倣するだけでは却って競争力が損なわれる危険性もあり、今こそ日本の強みを生かすアーキテクト、日本の擦り合わせ型開発に合ったアーキテクトを見出していく必要がある。このような背景から、2011 年度からの 3 年間のテーマとして、アーキテクトを取り上げることにした。最初の年度である 2011 年度は開発現場におけるアーキテクトの現状、その役割や求められるスキルなどを調査し分析した。続く 2012 年度は 2011 年度の調査をベースにより深く分析を行い、また海外のアーキテクト調査も行い、日本のアーキテクトの定義や役割などについてまとめた。2013 年度はこれまでの結果をベースに、アーキテクトヒアリングとワークショップ開催を行い、それぞれのアーキテクトが示すアーキテクトの定義や役割、そして特に育成について意見を聞き、また当委員会で分析した。

2013 年度まで活動してきたテーマの「アーキテクト」の活動から、そのアーキテクトが行うアーキテクチャモデリングを始めとする上流工程におけるモデリングが不十分であることが分かった。その結果、IoT の進化によるクラウドやスマート化、セキュリティなどの技術融合による複雑さの増大や、コスト面から汎用製品の活用と国内外の開発会社の活用の増大などが原因となる課題が表出してきている。そこでビジネスモデリングや要求モデリング、アーキテクチャモデリングなどの上流工程におけるモデリングの重大さが増していることから、2014 年度以降のテーマを「モデリング」にした。

モデリングはいつ、誰が、何を対象に、何を目的に、何を成果として、どのように実施し、どのようにそれを効率化し、どのように開発現場で受け入れられるようにするのか、そのためにどのようにモデラーを育成していくのかなど、このモデリングを効率良く正しく行うための課題やその解決を探ることを目的として、今後、活動していく予定である。最初の年度である 2014 年度ではモデリングの現状を中心に、その課題や対策などを探るべく、ワークショップを開催し、また有識者を招いてのブレーンストーミングを行った。

2. 本報告書の想定読者

情報家電機器、携帯機器、自動車、産業機器などの機器やシステムを開発するプロジェクトにおいて、下記のような方を含め、その機器やシステムに搭載される組込み系ソフトウェアの開発に関与されている多くの方に回答を依頼し、開発現場からみた「開発スピードアップの阻害要因」の実態について調査することを試みた。

- 開発プロジェクトの責任者あるいはプロジェクトマネージャ
- 開発プロジェクトの技術面をまとめている方
- 開発プロジェクトの技術者、開発プロジェクトのアーキテクト
- 生産技術部門など開発プロジェクト全般を技術面で統括/支援する部門の方
- 生産管理部門など開発プロジェクト全般を管理面で統括/支援する部門の方
- 組込み機器やシステムの事業企画や製品企画などの企画系部門の方

その調査結果を分析し、結果の取りまとめにあたっては、「開発プロジェクトのプロジェクトマネージャ層から経営層」を本報告書の主な読者として想定した。

本報告書は、「開発現場の組込み系アーキテクト」という切り口から、実際の開発現場で実施されている取組みの問題点や課題をまとめている。本報告書の内容から何か触発されるものを見つけて、読者の関与されている組込み系ソフトウェアの開発プロジェクトの今後の改善の一助になれば幸いである。

3. 2013 年度までの活動成果と 2014 年度の活動

以下に、2012 年度までに活動成果の要点を総括し、その後で、2013 年度の活動の狙いと活動内容について述べる。

3.1　2012 年度までの活動

(1) 2005 年度～2007 年度

2005 年度、2006 年度の 2 年間の活動を通じて、組込み系ソフトウェア開発における問題の根源が「大規模化」、「複雑化」、「短納期化」、「多機種化」の 4 つの波にあることを明確にした。そして 2007 年度の活動では、組込み系ソフトウェアにとって最重要かつ社会的要請と言える「品質確保」の問題を取り上げて、品質確保のための「品質施策」と「品質プロセス」に焦点を絞って、4 つの波への対処の観点から開発現場における取組みの現状と今後の取組みの方向性を分析して、今後の取組むべき方向について提言をまとめた。

2007 年度にまとめた提言では、組込み系ソフトウェア開発における現在の主要課題である大規模化、複雑化、短納期化、多機種化という 4 つの波に打ち勝つために、

- ハードウェア部門などとの連携
- 自動化
- 上流工程重視
- 多機種開発

などの必要性を提案した。

(2) 2008 年度～2010 年度

2008 年度からは、2007 年度までに行ってきた品質確保というテーマの視点を変えて、開発側目線とも言える「開発スピードアップ」について検討することにした。その最初の年度である 2008 年度の活動は、図 3.1-1 に示すように、開発スピードアップを阻害する要因の事例収集とその分析を行った。

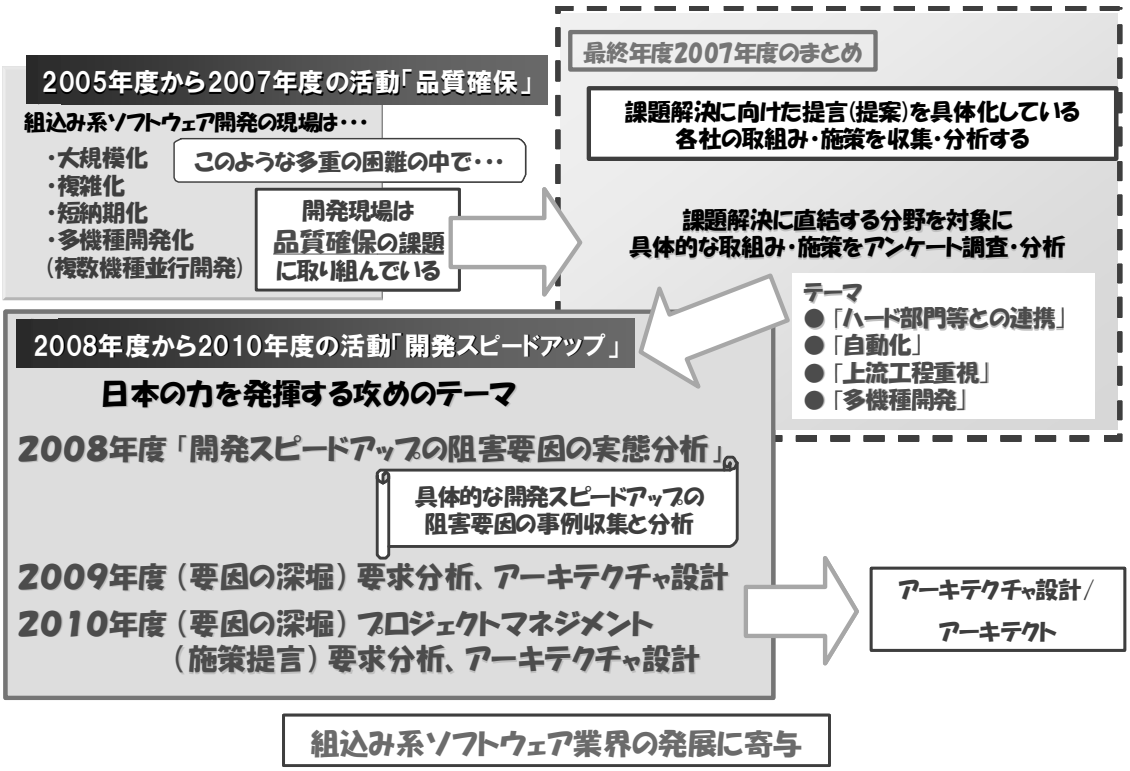


図 3.1-1　2010 年度までの活動

開発スピードアップの阻害要因の事例を収集するためにワークショップを開催し、またアンケート調査を実施した。これらを基に委員で議論を行い、報告書をまとめた[1)]。そこでは 30 個以上の開発スピードアップ阻害要因の事例を発見し、特に要求定義に重要な阻害要因があることが知れた。

続く 2009 年度の活動では、2008 年度で開発スピードアップを阻害する要因で重要であると知れた要求定義とアーキテクチャ設計に絞り込んで、要因の深堀を行った。ここでは、やるべきことがやられていない事が多いことが分かった[2)]。

最後の年である 2010 年度の活動では、2009 年度で深堀しなかったプロジェクトマネジメントを深堀し、プロジェクトマネジメントによる開発のスピードアップに取り組んだ。さらに 3 年間の活動の総まとめとして、2008 年度と 2009 年度で実施した要求分析とアーキテクチャについての提言を行った[4)]。

(3) 2011 年度～2013 年度

2010 年度までの活動では、組込み系ソフトウェアで開発スピードアップを阻害する要因で重要であるものとして、アーキテクチャ設計があった。また 2007 年度までの品質確保の活動においてもアーキテクチャ設計が重要であることを得ている。

[2)] 平成 20 年度ソフトウェアに関する調査報告書 II（IS-09-情シ-2）
http://home.jeita.or.jp/is/publica/2009/is-09-jyousi-2.html
[3)] 平成 21 年度ソフトウェアに関する調査報告書 II（IS-10-情シ-2）
http://home.jeita.or.jp/is/publica/2010/is-10-jyousi-2.html
[4)] 平成 22 年度ソフトウェアに関する調査報告書 II（IS-11-情シ-2）
http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=423&cateid=6

そこで 2011 年度から始まる 3 年間のテーマとして、アーキテクトそのものを取り上げるようにした。最初の年度である 2011 年度ではまず、開発現場におけるアーキテクトの実情を調査するようにした。例えば、各開発現場でアーキテクトはいるのかいないのかから始まり、その役割はどのようなものがあるか、アーキテクトが必要としている、または求められるスキルは何であるかなどを調査し、分析を行った[5)]。続く 2012 年度は 2011 年度の結果を元に委員の間で議論を行い、アーキテクトを役割やステークホルダなどから分類し、それに基づいたアーキテクトの配置や事例などをまとめた[6)]。

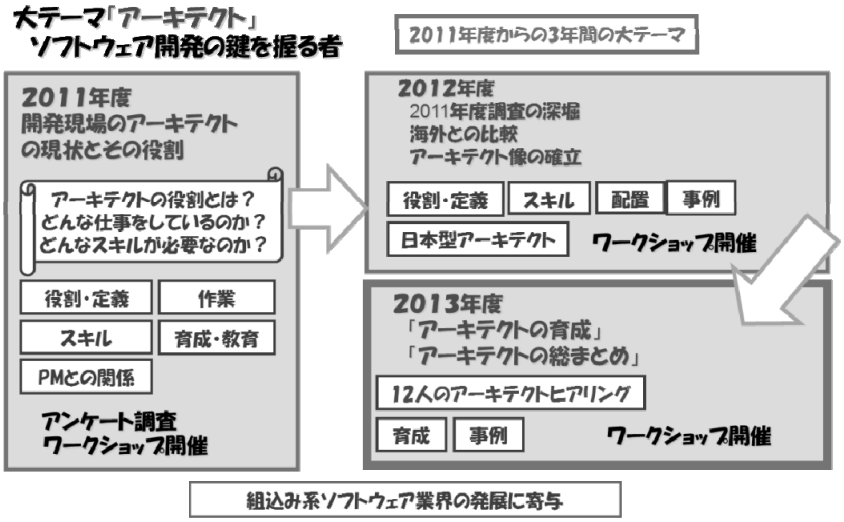


図 3.2-1　2013 年度までの活動

最後の年度である 2013 年度では、2012 年度までにまとめたアーキテクトに関する報告書を元に、開発現場の 12 名のアーキテクトにヒアリングを実施した。アーキテクトの成功事例や失敗事例、アーキテクトに必要なスキルなど多岐に渡るヒアリングを行った。またアーキテクトワークショップも開催し、各講演者や会場から意見を元に報告書にまとめた[7)]。

---

5) 平成 23 年度ソフトウェアに関する調査報告書 II（IS-12-情シ-2）
http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=461&cateid=6
6) 平成 24 年度ソフトウェアに関する調査報告書 II（IS-13-情シ-2）
http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=512&cateid=6
7) 平成 25 年度ソフトウェアに関する調査報告書 II（IS-14-情シ-2）
http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=556&cateid=6

3.2　2014 年度の活動

3.2.1　活動の狙い

2013 年度まで活動してきた「アーキテクト」の活動から、そのアーキテクトが行うアーキテクチャモデリングが主要な作業の一つにあるにも関わらず、それが十分に行われていないことが分かり、それがアーキテクチャ設計の課題になっていることが分かった。さらにアーキテクチャモデリングだけでなく、上流工程におけるモデリングが十分でないことが分かり、その結果、IoT の進化による各種の技術融合による複雑さの増大に対応できていない。例えば、クラウドやスマート化、ビッグデータ、セキュリティなどの技術融合に対応できるモデリングが行われていない。またコスト面から汎用製品の活用が必要になり、人材面からは海外や国内の開発会社の活用が必要になり、そのコミュニケーションや仕様伝達でモデリングがますます必要になってきているが、それに対応できていない。そこで 2014 年度以降のテーマをビジネスモデリングや要求モデリング、アーキテクチャモデリングなどの上流工程を中心とした「モデリング」にした。

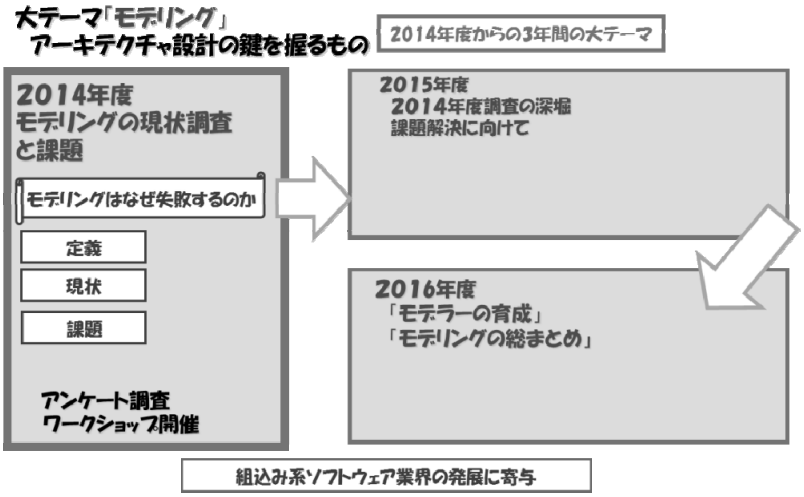


図 3.2-1　2014 年度の活動

モデリングを大テーマにした最初の年度である 2014 年度は、モデリングの現状を中心に調査・分析を行う。モデリングの定義から始め、どのような場面でどのようなモデリングが実施され、それがどのような目的で行われ、その成果はどうだったのかなどを調査する。このときに見つかった課題を分析しまとめる。また課題解決に向けて方策を検討し、来年度の活動に繋げる。

3.2.2　活動内容

1 年間の活動期間を概ね 3 期に区切って、以下のような活動を行った。

(1) 第 1 期（4 月～11 月）: ワークショップ 2014 の開催とモデリングの調査

最初に 2014 年度から新たに取組むことにした「モデリング」に関する活動の最初の年度になる 2014 年度の活動方針を決めた。この方針に従い、モデリングの調査を開始し、それと並行して、ワークショップの講師選定やワークショップのテーマを検討した。

具体的には 2006 年度から毎年開催しているワークショップを、2014 年 11 月に「～モデリングはなぜ失敗するのか～」と題した組込み系ワークショップ 2014 を開催した。このワークショップの参加申し込み時に参加者に対してモデリングに関する事前アンケートを実

施した。モデリングの現状調査として、どの工程でどのようなモデリングを行い、そのような手法やツールを使っているか、モデリングにおける課題やその対策をどのようにしているかをアンケートで聞いている。このワークショップ当日は、大阪大学の春名修介教授の基調講演を皮切りに、富士通の奥村一幸氏と富士通コンピュータテクノロジーズの石田晴幸氏による事例講演、そしてアマダの久我　雅人氏の事例講演を行い、最後に“全員参加 106 人ワークショップ：徹底議論！モデリングの成功のヒントを探る”と銘打った討論セッションで、参加者全員に対して、実際にモデリングをしてもらい、それを元に意見や質問を募り、活発な議論が行なわれた。

なお 10 月には CEATEC JAPAN 2014 において、当専門委員会での 2013 年度のアーキテクトに関する活動について当専門委員会委員が講演を行った。

(2) 第 2 期（11 月～3 月）: モデリングのブレーンストーミングとワークショップの分析、2014 年度の活動報告の作成

組込み系ワークショップ 2014 で、参加申込み者全員を対象としたモデリングに関する事前・当日アンケート調査の結果と 106 人ワークショップでの意見やコメントも参考に、委員の議論を通じて、モデリングの現状を分析した。

モデリングの有識者を招いて､委員とモデリングのブレーンストーミングを 2 回行った。このブレストではモデリングの失敗事例の分析を中心に、課題を分析し、そのいくつかで解決策を探った。

これらの活動成果を踏まえて、組込み系ソフトウェア開発においてモデリングの現状と課題分析を行い、実態としての主要な要因の深堀を試み、その結果を調査報告書として取りまとめた。

本報告書の主な内容は、4 章でモデリング技術の位置付けと現状の技術概要、5 章でモデリングの課題認識、6 章で課題分析、7 章で課題解決を述べ、8 章でまとめ、9 章で今後について記述している。また付録として、ワークショップの事前アンケート・当日アンケートや講演資料、CEATEC の講演資料などを掲載している。

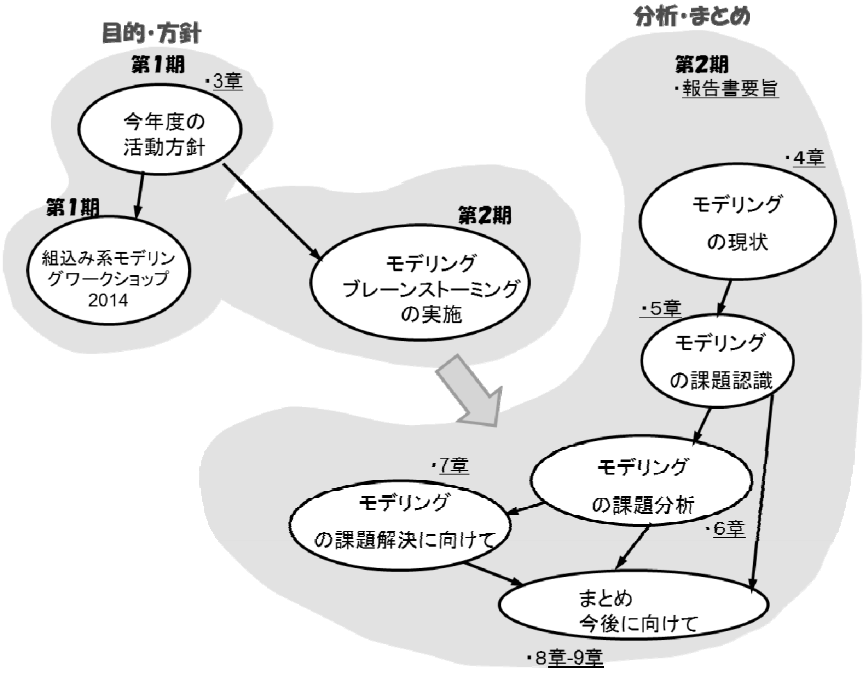


図 3.2-2　2014 年度の活動内容と本報告書の内容構成

4. モデリング技術の位置付け

本章では、モデリング技術の位置付けと、その現時点の技術状況を概説する。

4.1　ソフトウェア・システムのモデリング

ソフトウェア・システム開発を進めるにあたり、これから開発するソフトウェアやシステム（ソフトウェアを中心としたシステム）に対する要望や期待、ならびに実現する際の制約や決定済の項目を、当該ソフトウェア・システムの様々なステークホルダで共有するために、ソフトウェア・システム開発では設計フェーズを定義し、設計フェーズの成果として「仕様書」を作成する。ステークホルダには、開発するソフトウェア・システムのユーザなど、必ずしもソフトウェア開発の手法に精通していない人も含まれるため、仕様書を作成する際の共通言語として「自然言語」を利用することが多い。

今日、大規模化・複雑化するソフトウェア・システムを確実に開発するには、一般に、仕様書は、表 4.1-1 に示す特性を持つことが期待されている[1]。しかし、ソフトウェア・システムが大規模化・複雑化するにつれて、設計過程で作成すべき仕様書の量も増え、また、仕様書作成の分業化も進んでいる。この結果、表 4.1-1 に示した特性を網羅する仕様の作成、およびレビューによる仕様の欠陥排除は困難な状況にある。

表 4.1-1　仕様記述に期待される特性[IEE98]

仕様記述の特性	特性に対する定義
正確性（Correct）	顧客やユーザの最終的に達成したい目的に合致する仕様であること
完全性（Complete）	全ての達成すべき仕様が記述されており、抜けがないこと
一貫性（Consistent）	仕様記述中に、相互に矛盾する記述がないこと
非曖昧性（Unambiguous）	誰が読んでも“同じ解釈”が成立するものであること
検証性（Verifiable）	仕様で定義した内容が実現されたことを、テストアンドの有限の工数で確認できること
追跡性（Traceable）	仕様の源や仕様から展開された成果物が追跡可能であること
変更容易性（Modifiable）	上位仕様の変更に追随して容易に変更できること
優先順位付け（Ranked）	仕様実現の優先順位付けがされていること

これに対する仕様記述技術側からの改善策のひとつが、開発対象ソフトウェア・システムのモデル化になる。ここでモデルは、「開発対象のソフトウェア・システムの仕様を、ある特定の視点（Viewpoint）から抽象化し、その視点から見た特性が際立つように記述したもの」と定義される。また、このモデルを作成する作業を、モデリングと呼ぶ。

ここで、視点（Viewpoint）は、開発対象ソフトウェア・システムのステークホルダ毎に異なる。例えば、ソフトウェア要求仕様を策定する要求エンジニアは、外部から見たソフトウェアの振る舞い（External Functional Behavior）の記述内容の一貫性と網羅性を志向する。ソフトウェアのアーキテクチャを定義するアーキテクトは、ソフトウェアを構成する構成品目（例えば、タスクやプロセス、あるいはパッケージ／モジュールなどのソフトウェア構造を表す構成要素）と、構成品目間の相互作用とインタフェース仕様を志向する。具体化ユーザや顧客は、現在の業務フローとの整合性や、利用手順、使いやすさなどを志

[1] IEEE-Std-830-1998[IEE98]は、ソフトウェア要求仕様書を対象とした IEEE 標準であるが、ここに示した特性は設計仕様書においても同様に期待される項目と考えられる。

向する。各々の興味の対象が異なるため、開発対象システムの特性を抽象化する視点も変わる。

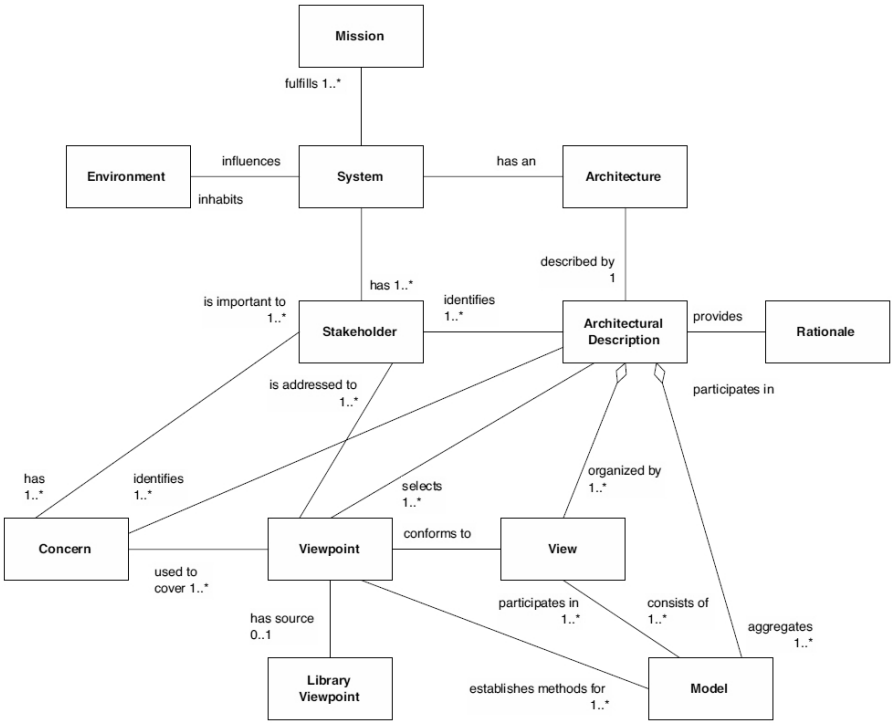


図 4.1-1　IEEE Std.1471-2000 に示される View Point のモデル

このような状況を整理し、複数の異なる観点から記述されたモデル間の関係付けを整理するため、IEEE Std. 1471-2000[1] [IEE00]では、モデル記述のメタモデル（図 4.1-1）を提唱している。また、この Viewpoint のモデルを具体化したものの代表例が、Enterprise Architecture などのアーキテクチャフレームワークと位置付けられる。例えば Zachman Framework[ZAC87]（図 4.1-2）は、この Enterprise Architecture の一例と捉えて良い。

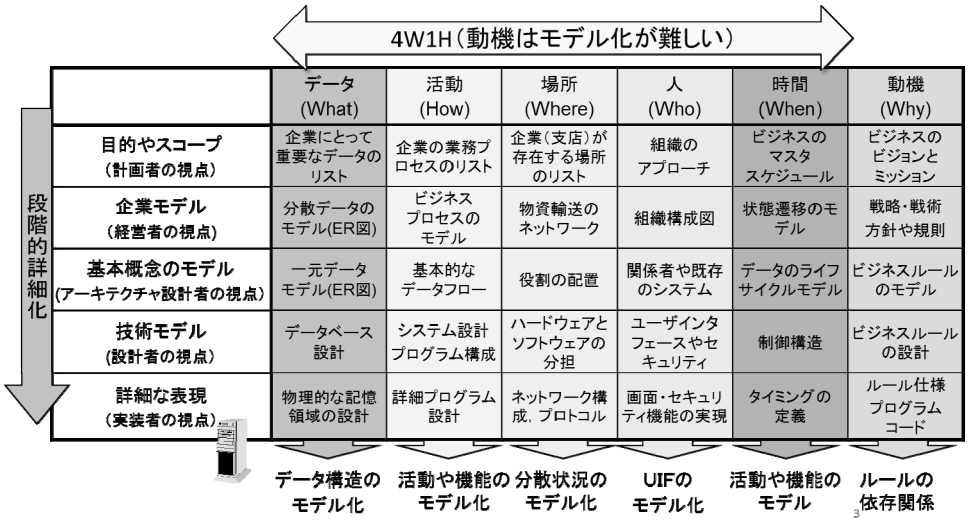


図 4.1-2　Zachman Framework

[1] IEEE-Std-1471-2000 は、現在 ISO/IEC/IEEE 42010 として ISO 化がされている[ISO11]。

Zachman Framework では、縦軸にはモデリングに対する視点を置く。経営者が捉える視点は、情報システム導入に対する計画や利用範囲（システムのスコープ）となる。一方、最も詳細な視点では、実装を強く意識した視点を採用する。縦軸にある視点は、上位視点でのモデルを受けて次の視点でのモデルが作成される、という、段階的な視点の詳細化を構成している。また、横軸には、興味の対象を 5W1H の切り口で並べる（実際には、Why を形式的に仕様化することは難しいため、実際のモデル化は 4W1H）。そして、図 4.1-2 に示した各マス目をひとつの視点（Viewpoint）として、開発対象システムをモデル化するという捉え方をする。

また、UML[OMG09]も同様に、複数の視点（Viewpoint）から、開発対象ソフトウェア・システムを捉えられるよう、複数のモデリング言語を提供している。

こうした複数の視点から繰り返しモデリングを行なうことを前提に、今日、様々なモデリング手法や言語が揃ってきている。これらのモデリング技術は大きく、次の 3 種類に分類できる。

・一定の記述文法を持った図や表を用いた仕様のモデリング
・形式的仕様記述言語を用いたモデリング
・物理的な数式を解いて観測対象の動作をシミュレーションする目的のモデリング

各モデル化の手法について以下に簡単に示す。

(1) 一定の記述文法を持った図や表を用いたモデリング

表 4.1-1 に示した「仕様に求められる特性」を確保するひとつの方策として、一定の記述文法に則って、開発対象ソフトウェア・システムの仕様をモデリングする図式言語がこれまでに多数提唱され、またソフトウェア・システム開発の現場でも利用されつつある。代表的なモデリング言語を表 4.1-2 に示す。

これらのモデリング言語は、ソフトウェア・システムの新しい分析・設計手法の進展とともに発展して来た経緯がある。例えば、ソフトウェア工学分野では、70 年代に提唱された構造化分析・設計手法に併せて、データフロー図やストラクチャーチャート[DEM79]、あるいは IDEF0[MAR05]などが提唱されてきた。80 年代後期からはオブジェクト指向分析・設計手法が数多く提唱され、これらの手法が各々独自のモデリング言語を導入してきている。これらの分析・設計手法では、単一のモデリング言語を利用するのではなく、機能、動作、データなど、個々の視点から対象をモデリングすることを想定しており、これらの視点毎にモデリング言語を用意するアプローチが主流である。

1997 年ごろからは、これらのモデリング言語を統一化しようとする機運が高まり、Object Management Group による UML[OMG09]の標準化に繋がっている。2004 年には UML2.0 が発表され、13 種類の図式モデリング言語と、stereotype を利用した言語拡張の考え方が標準化されている。また、システムエンジニアリングへの適用も議論され、UML の拡張として SysML[FEI09]が定義されている。

また、分析・設計以降の詳細なコード設計レベルでのモデリング言語として、表 4.1-2 に示すように、

・コードロジックのモデル化の手段となるフローチャートなどのモデリング言語
・FA 機器でのプログラム設計に適したプログラムのモデル化手段となるファンクションブロック図（FBD）やラダー図など[JIS12]のモデリング言語

も利用されている。また、並列動作を表現するペトリネット[PET81]とそのバリエーション[1]、ITU の Specification and Description Language（SDL）[ITU11]など、特定分野で利用されているモデリング言語も数多く存在する。

[1] 例えば、カラードペトリネット[TOM97]など。

表 4.1-2　代表的な図や表を用いたモデリング言語

対応する分析・設計手法		代表的なモデリング言語
構造化分析・設計手法（SA/SD）		
	データ定義	Entity-Relationship 図（E-R 図）[CHE76]
	機能定義	データフロー図（DFD[DEM79]、IDEF 0 [MAR05] 等
	動作仕様定義	状態遷移表、状態遷移図、Statecharts[HAR87] 等
	モジュール構造定義	Structure Chart[DEM79] 等
オブジェクト指向分析・設計		
	オブジェクト指向分析・設計手法	Shlare-Mellor[SHA88], Coad-Yordon[COA90], OMT[RAM91], UseCase[JAC92], BOOCH[BOO93], ROOM[SEL94], Use Case Map[BUH95]等の各図法
	統合化	UML[OMG09] , SysML[FEI09]
その他		
	プログラミング	フローチャート、PAD、HCP 等 ファンクションブロック図[JIS12]、ラダー図[YOS94]等
	並行処理	ペトリネット[PET81]

(2) 仕様証明を狙ったモデリング

前項に示した図式モデリング言語は、図や表を表現手段として用いることで、記述容易性や読みやすさを確保したモデリング言語と言える。仕様記述にあたって一定の文法（記述ルール）を持つものの、各言語の文法定義の厳密さ、言語利用者の理解度の差異により一定の曖昧さが残ることも多い。

これに対して、開発しようとするソフトウェア・システムの仕様を、集合論や論理式などを用いて抽象的かつ厳密に記述し、数学的に検証可能とすることで、仕様策定時から記述された仕様の不整合を検出可能とし、実装やテストでの仕様誤りの発覚やそれに伴う手戻り（多大なコストを要する場合が多い）を防ぐことを狙った“形式的仕様記述言語”が数多く提案されてきている。これらの形式的な仕様記述言語も、現実の世界を抽象化し、仕様として記述するという観点で、モデリングの一手法と位置付けられる。

形式的仕様記述言語には、いくつかの類型がある。例えば、[LAM00]では、以下のように分類している。

- 履歴ベースの仕様記述：記述対象のシステムを、時間軸に沿って取りうる履歴（振る舞い）によって記述しようとするアプローチ。時間軸に沿ってシステムの特性を記述するため、実時間時制論理が必要になる。
- 状態ベースの仕様記述：記述対象のシステムを、ある時点で取り得るシステムの状態のスナップショットとして記述するアプローチ。このアプローチの代表的なモデリング言語には、Z、VDM、B などがある。
- 遷移ベースの仕様記述：記述対象のシステムを、状態から状態への遷移に必要な状態遷移によって特徴付けようとするアプローチ。状態遷移に紐付けられた遷移関数によって、システムが何をするかを記述する。Statecharts、PROMELA、RSML などが代表的なモデリング言語になる。
- 機能仕様の記述：記述対象のシステムを、数学的な機能の構造化された集合体として記述するアプローチ。2 つのアプローチに分割される。ひとつは、抽象データ型などを用いて代数関数を構造化して機能を記述するアプローチで、ASL や OBJ などの形式仕様記述言語がある。もうひと

つは、高次関数を用いて論理式を記述する仕様記述言語で、HOL や PVS がこのグループに属する。

・動作仕様を利用した仕様記述：記述対象システムを、何らかの抽象マシンで実行可能なプロセスの構造として記述するアプローチ。例えば、PAISLey、GIST やペトリネットなどはこのグループになる。

形式的仕様記述言語によるモデル化は、モデル化にあたって文書化されていない項目も含めて詳細な記述を要求されること、一方でモデル化作業やその妥当性の確認が困難なこと、形式的に開発対象ソフトウェア・システムの仕様をモデル化するには何度も修正が必要なこと、などから古くから提唱されているモデリング手法にもかかわらず、中々開発現場に適用されてはいない。ただ、近年では、航空管制システム[HAL96]、SoC などの LSI 設計[TAD10]など、仕様の曖昧さや一貫性の欠如が、事業者や社会に大きな損失を招く分野での利用が広まりつつある[IPA07a]。

(3) 観測対象の動作をシミュレーションする目的のモデリング

モデル化のもうひとつの手法は、仕様として書き下す対象の動作を物理学などで用いられる「仕様化対象の動作や特性を微分方程式などの数式で表現する」手法である。特に、物理的な法則に則った動作をする「系」（システム）を表現し、ある特定の条件下でどのような動作が見られるかをシミュレーションによって予測・検証するケースに利用される。

組込みシステム開発の中では、例えば、電力場、電磁気場、空力特性、熱特性など、ソフトウェアで制御対象とする自然現象の振る舞いを模擬し、機器に与える影響を予測するなどの目的で利用される。また、ネットワークやサービス要求の輻輳状況を模擬する手段として、待ち行列を利用したモデル化を行なうことも良く行われる[KAS06]。

これらのシミュレーションは、仕様空間上で制御対象の特性を反映した数式モデルを製作し様々な仮想実験を行なうことで、

・実際の動作を確認することが困難・危険な状況での系の動作を模擬し「系」の特性を知る、振る舞いを予測する

例えば、大規模震災時の津波の振る舞いのシミュレーション[IMA11]や交通渋滞対策に用いられる交通シミュレーション[DOB98]、あるいは人の振る舞い[MET09]など、現実環境での実験が困難な事象をシミュレーションするケースで利用する。

・機器を製作して実験をするよりも安価に、システムに対する制御を実験する

例えば、車載機器搭載用制御ソフトウェア開発では、ハードウェアに組み込んでのソフトウェアの検証・妥当性確認に先立って、シミュレータと組合せてソフトウェア検証を進める取り組みが進んでいる[TON10] [FUK13]。こうした取り組みは、機器の開発以前に仮想環境を構築して実施することで、適切な制御アルゴリズムを安価に、また早く開発することを狙っている。また、ソフトウェアや一部の制御機器を先行して妥当性を確認する目的で、Software In the Loop Simulator（SILs）／Hardware In the Loop Simulator（HILs）[IPA12]の利用も進んでいる。

ことを目的とする。

4.2　モデルを利用したソフトウェア・開発の技法と位置付け

4.1 に示した様々なモデリング技術については、作成するモデルの利用目的に応じて、どのようなモデリング技術を利用するかが変わる。ソフトウェア・システム開発では、仕様化の作業は、仕様化、作成された仕様の検証、そして作成された仕様の妥当性確認のステップを踏んで進められる。

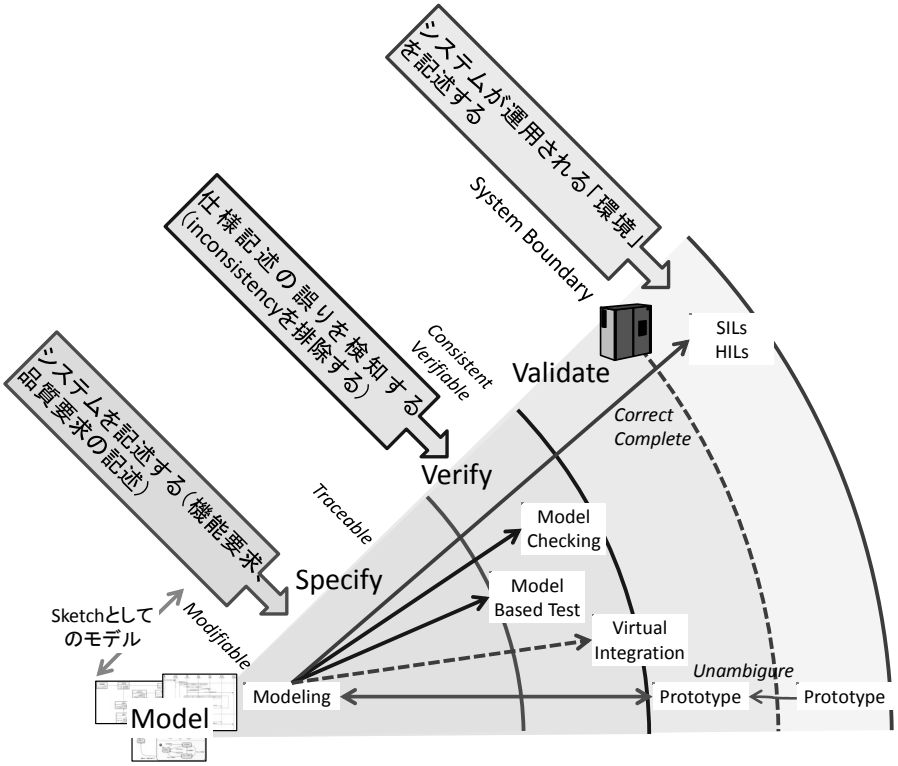


図 4.2-1　モデルを活用したソフトウェア・システム開発の技法とその位置付け

・ 仕様化（specification）：曖昧性を排除した仕様を作成し、関係者の間で共有することを目的とする
・ 仕様検証（verification）：記述された仕様の一貫性や完全性を検証することを目的とする
・ 仕様の妥当性確認（validation）：仕様の正確性（合目的性）を確認することを目的とする

これらの目的に照らして、近年提案されているモデルを利用したソフトウェア・システム開発技術を対応付け、模式的に記述すると、図 4.2-1 のようにまとめることができる。以下、仕様作成作業の各ステップに対応して、モデルを活用したソフトウェア・システム開発技術の位置付けを述べる。

(1) 仕様化（specification）

モデリングの基本は、開発対象のソフトウェア・システムを仕様化する（Specify）する活動である。開発対象のソフトウェア・システムを、様々な観点から抽象化し、特定の表記方法でモデルとしてまとめる。

また、仕様化の目的は、ユーザや顧客、あるいは自身が開発対象のシステム／ソフトウェアに対する共通の理解を得ることである。一定の文法規則に従って記述された仕様を利用してコミュニケーションを取ることで、仕様に対する異なった解釈、異なる期待を誘発する仕様の曖昧さ（ambiguity）を排除できる。一方で、今日の代表的なモデリング言語である UML では、全ての品質要求[JIS03][JIS13]を記載する枠組みは与えられておらず、これに対応するには、UML の言語仕様に定義されているステレオタイプを利用して、仕様記述モデルを独自に拡張するなどの対応が必要になる[1]。

このように肥大化するモデリング言語の言語仕様に対して、Fowler はモデルの完全性を求めるよりも、コミュニケーションを加速する目的でモデリング言語を利用することが重

[1] OMG では、こうした特定分野向けのモデル拡張も標準化を進めており、例えば UML MARTE profile[OMG11]はリアルタイム組込みシステム向けのモデル拡張の標準の例である。

要と主張している。[FOW03]では、UML に対して、“人によって UML に期待するものが違う”として、この捉え方は以下の 3 種類に分類されるとしている。この中で、③をまず実践することが重要というのが、Fowler の主張になる。

① 設計図としての UML（UML as blueprint）：プログラム設計者に対して正確に仕様を伝えるため、網羅的かつ完璧なモデルを作成するアプローチとして、この設計図としての UML を位置づけている。

② プログラミング言語としての UML（UML as programming language）：UML の広範な言語仕様は、ソフトウェアの作成に必要な全ての記述ができる高級プログラミング言語として UML を捉える人も居るとしている。UML の図的な表現を用いてプログラムを記述し、ツールを用いてコード生成すれば、通常のプログラミング言語よりも高い生産性でコード開発ができるという主張である。なお、Fowler 自身は、このアプローチで生産性は高くならないと表明している。

③ スケッチとしての UML（UML as sketch）：開発者は UML をシステムの「ある側面」を伝え、相談する目的で UML を利用する立場を、Fowler はこう名付けている。このアプローチでは、モデリングは同僚に説明したいと思う重要な個所だけ行い、UML の文法も全て利用する必要は無く、取捨選択して利用するもの、としている。

(2) 検証（verification）

検証（Verification）は、「正しくソフトウェア開発が進められていること」を確認する活動[BOE84]である。通常のソフトウェア開発では、各種仕様書やコードに対するレビュー、あるいは実行可能となったコードに対するテストとして実施され[IPA07b]、開発対象ソフトウェア・システムの非曖昧性（unambiguity）や一貫性（consistency）を確認する。

その際、開発対象ソフトウェア・システムの特性を網羅的にモデル化出来れば、記述された仕様の検証は容易に進められる。また、モデル記述の一貫性を確保することで、未定義の仕様を早期に検知し、将来の欠陥混入を予防することも可能になる。これは、Fowler の「①設計図としての UML」を意図したモデリングになる。

この検証を目的にモデルを利用する開発技術について、最近の動向として、以下のものを概説する。

・ Model Driven Architecture によるモデルマッピング技術
・ モデルベーステスト（図 4.2-1 の Model Based Test）
・ モデル検査（図 4.2-1 の Model Checking）
・ Virtual Integration（図 4.2-1 の Virtual Integration）

① Model Driven Architecture によるモデルマッピング技術

作成されたモデルを検証する直接的なアプローチではないが、OMG が提唱するモデル駆動アーキテクチャ（Model Driven Architecture : 以下 MDA）[FRA03a]では、Meta-Object Facility（以下 MOF）を利用した「モデル間のマッピング機構」の利用を提唱している（図 4.2-2）。MOF は、分析用のモデル、設計用のモデル、あるいは実装用のモデルなど、開発フェーズの各々に対応したモデリングのモデリング構成要素に対する「意味」を定義する道具となる。

MDA では、これらの MOF 間の対応関係をマッピングするルールを用意することで、計算独立モデル（Computation Independent Model : CIM と称す）、プラットフォーム独立モデル（Platform Independent Model: PIM と称す）、プラットフォーム依存モデル（Platform Specific Model : PSM と呼ぶ）、各々の間のモデル変換を行なう手段を提供することを狙いとする[FRA03b]。最終的には、ソフトウェアの実装に充分な情報を PSM モデルとしてまとめることが可能となり、PSM から実装コードへの変換も可能とする。従って、PSM は Fowler の言う “②プログラミング言語としての UML”と位置付けられ

る。
MDA のモデル変換の枠組みである MOF、MOF 間のマッピングルールを利用することで、他のモデリング言語とのモデル変換も可能になる。例えば、多くのソフトウェア開発者にはモデリング言語として活用しきれていない形式的仕様記述言語を、MOF とマッピングルールを利用して UML の各モデルと対応付けることも可能となる。実際、AADL[FEI06]や OpenEmBeDD[MAH08]などでは、MOF とマッピングルールの定義を利用して、形式的仕様記述言語と UML や SysML などの図的なモデリング言語を相互に変換を提供している。図 4.2-3 に OpenEmBeDD でのツール連携の様子を示す。

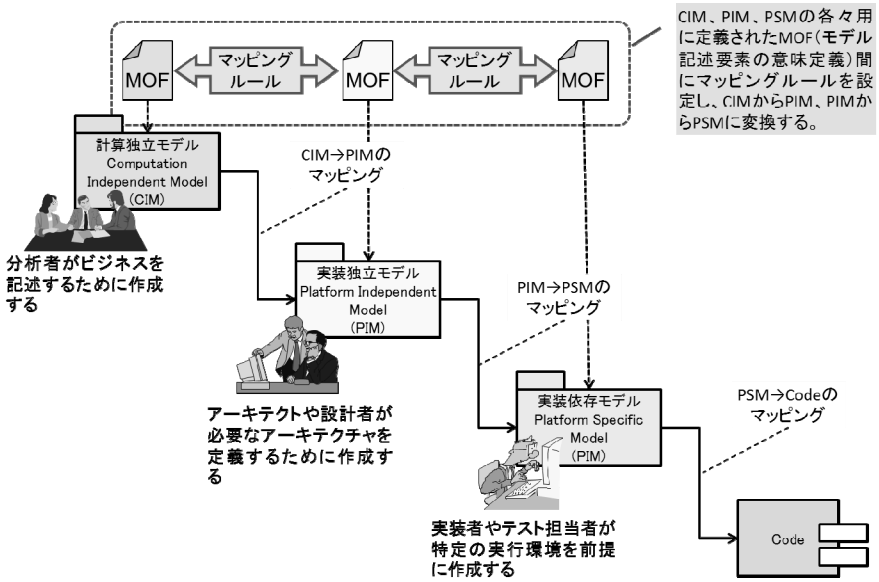


図 4.2-2　Model Driven Architecture のアプローチ

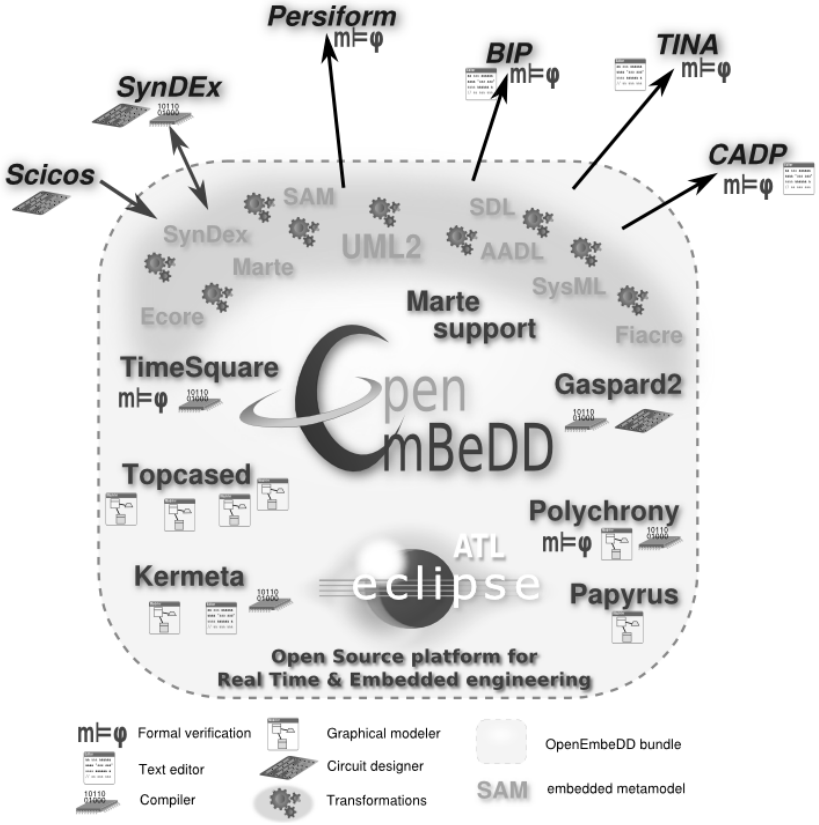


図 4.2-3　OpenEmBeDD で連携するツール群

http://openembedd.org/public/img/OpenEmbeDD_main_scheme.png

② モデルベーステスト

モデルベーステストは、テスト対象のシステムの振る舞い（仕様）をモデル化し、そのモデルに基づいてテストケース、テストスイート、テストスクリプトなどを生成する技法である。モデル化の方法として、有限状態マシン、状態遷移図、デシジョンテーブル、文法などが用いられる[SQU14]。開発対象ソフトウェア・システムの設計時に作成したモデルからテストケースを直接導出することで、テストケース設計時に漏れのないテストを実現することを狙っている[ZAN13]。

モデルベーステストには、次のものが含まれる[BOU14]。

a) デシジョンテーブル

デシジョンテーブルは、条件（大雑把に言えば入力）とアクション（大雑把に言えば出力）の論理関係を示す。テストケースは、その結果によって生じる、対応するアクションの可能な全ての組合せを考えることで、体系的に導出される。

b) 有限状態マシン

テスト対象のソフトウェア・システムを有限状態マシンとしてモデル化することで、テストは状態と状態間の遷移を網羅するように選定される[ISO07] [YAM98]。

c) 形式的仕様

形式的仕様記述言語を用いて仕様を表記することによって、機能的テストケースの自動生成が可能となる。それと同時に、テスト結果を検査するのに役立つ原理（オラクル）が提供される。

d) ワークフローモデル

ワークフローモデルは、人やソフトウェアによって実施されるアクティビティの

繋がりを仕様化する手段であり、通常、図的な表現方法を持って書き下される。個々のアクティビティシーケンスが、ひとつのシナリオを形成する。ワークフローに沿って、典型的なシナリオ、条件が異なった時に実行される代替シナリオの全てがテスト対象となる。特に、ビジネス・プロセスをテストする場合は、ワークフローモデルの中でも役割(role)に注目したテストを行なうことになる[TAN13]。

③ モデル検査

モデルベーステストが、モデルを元に開発されたソフトウェア・システムに対する検証の手段を提供するのに対して、作成されたモデルそのものを検証対称にするのが、モデル検査のアプローチである。

モデル検査は、仕様の形式的検証方法のひとつである[BOU14]。テストや人手でのレビューでは難しい、特定の状況でのみ顕在化する障害の自動検出や、対称範囲において障害が発生しないことを保証する目的で実施される[SQU14]。自然言語や図表などで記述された仕様を、文法や意味論が厳格に定義された形式仕様記述言語を用いてモデル化して、検証する。モデル検査での検証の考え方は、様々な条件に対して網羅的に探索する状態空間探索を行い、到達可能性を分析するというものである。一般に、以下の項目を検査する用途で利用される。

- アサーション:ある時点で、成り立つべきソフトウェア・システムの特性、性質
- 不変条件、事前条件、事後条件:システムが状態として保持するデータについて、常に成り立つ条件(不変条件)、各関数・操作の実行前に前提として成立すべき条件(事前条件)、及び実行後に成立しているべき条件(事後条件)
- 安全性や活性:システムの状態遷移の進み方に関し、ある条件下で特定の状況が発生してはならないという性質(安全性)や、ある状況が発生しなければならないという性質(活性)

これらの性質が成り立たない場合は、モデル検査ツールが反例として、この事象を出力する。モデル検査ツールとしては、SPIN/Promela や VDM++などがある。

一般に、これらのモデル検査ツールは、モデルを定義した開発フェーズで実施するため、分析・設計フェーズでの開発コストが増加する傾向にある。また、検査対象とする状態遷移モデルの複雑さに応じて探索する状態空間が膨大に膨らみ、モデル検査を行なう計算機環境によっては実用的な時間、メモリ容量の範囲で検査を終えられなくなることがある。このような場合は、モデルを見なおして、検査したい性質に関連した状態遷移のみを抽象化、抽出するなどの必要もある。

こうした難しさがあるものの、特に安全性や高信頼性が要求される社会インフラを構成する機器用の組込み制御ソフトウェア開発で、モデル検査が使われる事例も増えている[TAD10][HAT12]。また、近年では、UML や状態遷移表などの図的表現によるモデルから、形式的仕様記述言語によるモデルを生成するなどの取り組みも進められている[YAS14]。OpenEmBeDD や AADL も、こうした「モデル変換器」を通すことで、形式仕様記述言語を用いた仕様検証を実現している。

④ Virtual Integration

Virtual Integration は、CMU SEI が提唱する、実際にソフトウェア構成品やハードウェア構成品が出来上がる前に、仮想空間上でモデル化された仕様記述を組み上げ、アーキテクチャ設計の問題を早期に検出、手当しようというアプローチである[FEI09]。

Virtual Integration の目的のひとつは、アーキテクチャ設計の結果として定義される、ソフトウェアやシステムの構成要素間のインタフェース仕様を早期に検証することである。アーキテクチャ設計の結果を、AADL を用いてモデル化し、これを仮想空間上

で組み上げて、様々なケースで実行、検証を繰り返し、必要な安全性を達成しているか、性能などの品質要求を満たしているかを確認することを狙っている[FEI12]。個々のシステム構成品、ソフトウェア構成品の開発に着手する前に、ユーザや顧客の要求を満たす適切なアーキテクチャを開発したか、またアーキテクチャを定義する仕様が一貫性を持っているかを確認することを想定している。Virtual Integration の過程で発見した仕様の不整合は、実際に個々のソフトウェア構成品、システム構成品の開発の前に修正され、見直されるため、製品のリリース間際に行われるテストでの不具合検出に伴う大きな手戻りは減ると期待される。

AADL を利用したシステムのモデル化や Virtual Integration は、高い安全性や信頼性が求められる、大規模な航空機制御システム／ソフトウェア、車載機器搭載ソフトウェアなどでの適用が進められている。

(3) 妥当性確認（Validation）

Verification が「正しくソフトウェア・システムを開発している」ことを確認する作業であるのに対して、Validation は（顧客やユーザが求める）「正しいソフトウェア・システムを開発している」ことを確認する作業と定義される[BOE84]。表 4.1-1 に示した仕様記述に求められる品質の中で、完全性（Complete）や正確性（Correct）を確認する活動と位置づける。

Validation は、通常のソフトウェア・システム開発ではいくつかの方法で実践されている。代表的なものは、レビューやテストによる確認である。特に、顧客やユーザを交えたレビューや、仕様によって定められた利用条件を逸脱した環境を模擬したテスト（高負荷やエラーに相当する入力）として、Validation の活動が行われる。この他、開発の早い段階で提示された仕様に基づいて作成するプロトタイプの作成と仕様評価も、Validation のひとつと捉えられる。

この Validation 目的でのモデルの利用は、大きく次の 2 つを考えることができる。

・モデルベースのプロトタイプ

・モデルを利用して開発した対向装置を利用してのテスト：SILs、HILs などのアプローチ

① モデルベース・プロトタイプ

プロトタイピングは、これから開発する製品を動作させるモックアップを作成する開発手法である。プロトタイプは、その目的に応じていくつかのアプローチが提唱されている。

- 顧客要求の抽出や開発者側の顧客要求に対する理解の妥当性を確認する場合、作成したプロトタイプを評価後に放棄する「使い捨て型プロトタイピング」
- アーキテクチャが顧客要件を満たしているか、実現性があるかを確認する場合は、作成したプロトタイプを製品に成長させる「成長型プロトタイピング」

特に前者の目的で作成するプロトタイプは、開発者と顧客・ユーザの間で、これから開発するソフトウェア・システムの最終的なイメージを共有することで、ユーザや顧客が提示しきれていない要求を抽出し、あるいはユーザや顧客の要求を理解して要求仕様を策定しているかを確認することを目的に開発される。この目的でのプロトタイピングでは、顧客・ユーザの発言を受けて簡単にプロトタイプを作り替え、ゴールの摺合せを行なうことが重要になる。このため、GUI を紙芝居として作成するプロトタイピングから、Quick & Dirty なコードを準備するプロトタイピング迄、いくつかのアプローチがある。重要な点は、このプロトタイプの開発では、顧客やユーザの声を反映して、プロトタイプの改変を素早く繰り返すことにある。同時に、評価対象とする要求事項の確認がとれたら、その時点で理解した内容を文書化し、プロトタイプそ

のものは放棄する必要もある。
この使い捨て型プロトタイピングのアプローチに、モデルを利用したプロトタイプを利用する方策は、古くから提案されている。例えば、[TAM97]では、OMT 法に基づいて作成した仕様を直接実行し、制御対象機器の動作を可視化する。設計モデルをもとに性能評価を行なう手法も、[ICH08][SAK10]などで提案されている。[ATA07]では、モデルを利用したプロトタイプの生産性が良いことも示されている。
構築物の企画から設計の段階で、モデルを利用して最終製品の機能や性能を可視化、評価するアプローチは、一般に Computer Aided Engineering（CAE）と呼ばれている。なお、これらの構造物の解析目的でのモデリング手法は、有限要素法（Finite Element Method：以下 FEM）を用いることが多い。
これらのモデルベース・プロトタイプで重要な点は、個々の製品開発において、実機開発の前に、設計の妥当性を確認することにある。従って、ソフトウェア・システム開発の中でこれらの手法を適用する際は、何を評価するためにプロトタイプをするのか、またどのように評価するのかを事前に計画しておくことが求められる。実際にモデルベース・プロトタイプを行なう場合はこの評価基準に照らして、どこまでのモデル化が求められるかが決まると考えると良い。

② SILs, HILs などのアプローチ
4.1 (3)で紹介した SILs、HILs の取り組み[TON10][IPA12]はシミュレータを用いて、開発対象のソフトウェア・システムの「環境」をモデル化し、これを利用してテスト用対向装置をモデルベースで開発するアプローチと捉えられる。特に、
・ 実際のテスト用環境を用意するのに時間やコストがかかる場合
・ 実環境でのテストが困難な場合
などに、開発対象とするソフトウェア・システムの「環境要因」をモデル化し、これを活用して仮想的なテストを行なうことが可能となる。ソフトウェアや、ソフトウェアを組み込んだサブシステム単体でのテスト（特に検証目的でのテスト）を、システム全体を組み上げる前に行なうことができることから、組上げ作業前に欠陥を検出・排除し、システム全体の組上げ作業を効率化することが可能となる。

4.3 アーキテクティングにおけるモデリング技術の適用

当委員会では、2011 年度より 3 年間、組込みソフトウェア開発におけるアーキテクチャ設計の重要性を切り口に、アーキテクチャ設計の推進者であるアーキテクトの役割と期待されるスキルについて議論を重ねて来た。本節では、この観点から、4.2 節で紹介したモデリング技術の位置付けを整理しなおす。

まず、組込みソフトウェア開発を議論するに際し、機器制御ソフトウェアを組み込んだ製品全体の開発プロセスは、[IPA07b] [IPA07c] [JIS04]などを統合することで、図 4.3-1 のようにまとめることができる。

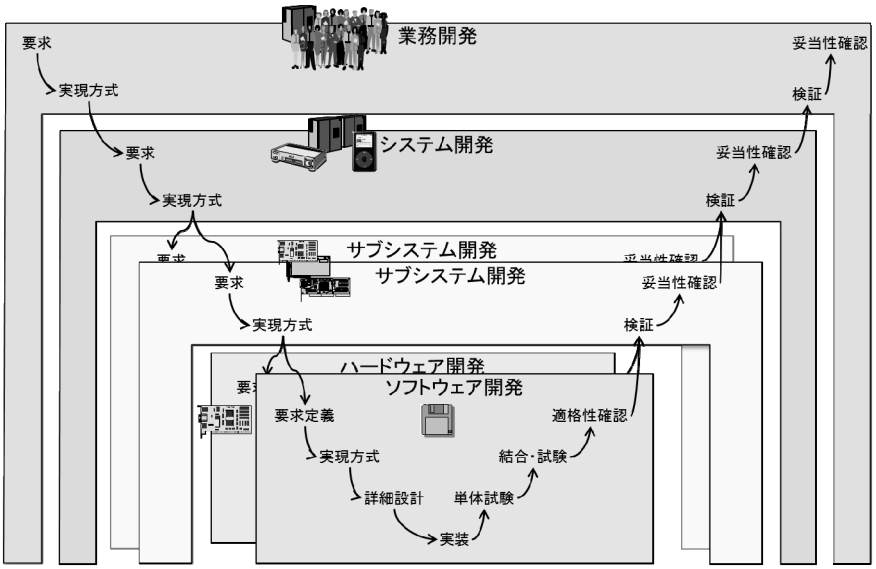


図 4.3-1　製品の開発ライフサイクルプロセス

ここでは、もっとも外側に業務やユーザのビジネスに対する要求を具体化し、その要求を達成するために、ビジネスの実現方式（ビジネス・アーキテクチャ）を定義する活動と位置付けられる[IPA07b]。ここでビジネス活動の一部は、情報システム化により対応することを決定されると、システム開発に対する要求が具体化され、システム開発のプロセスが実行される[JIS04]。システム開発の中でも、まずシステムに対する要求が具体化され、その要求を実現する実現方式（システム・アーキテクチャ）が定義される。ここでは、システムを構成するサブシステム群が定義され、システムに対する要求はこれらのサブシステム群に割り当てることになる。さらにサブシステムの中でも、サブシステムに対する要求に基づいて、その実現方式が定義される。一連の詳細化を通じて、最終的に、システムの最小構成要素となるソフトウェアやハードウェアに対する要求が定義される。ここからは、ソフトウェア開発プロセス[IPA07b]やハードウェア開発プロセスが実行される。ここでも、同様に、ソフトウェアに割り当てられた要求を実現するための、ソフトウェア実現方式（ソフトウェア・アーキテクチャ）が定義され、それに基づいて実装作業が行われる。出来上がったソフトウェアやハードウェアは、各々、定義した仕様通りのものが出来たかどうかを確認する検証の活動、与えられた要求の狙いを満たすかを確認する妥当性確認の活動を経て、上位の開発プロセスに返される。同様に、上位の開発プロセスでは、下位スコープの成果物を集約・統合[1]し、そのスコープでの（例えば、サブシステムならサブシステムの範囲、システムならシステムの範囲での）検証と妥当性確認の活動をしながら、システム全体を組み上げていく。

この開発プロセスのスコープの段階的詳細化と、成果物の段階的な組み上げは、図 4.1-2 に示した Zachman Framework の示すものと同じことに注意されたい。すなわち、Zachman Framework の詳細化の各階層にアーキテクティングの活動があり、各階層のスコープでモデル化を行なうことになる。

アーキテクトの活動は相似形で、その活動は図 4.3-2 のようにまとめることができる。

[1] 一般には「結合テスト」と呼ばれるが、本来の作業の目的は、CMMI[CMM02]の key process のひとつである Product Integration（統合）という捉え方が適切である。従って、ここでは、この活動を「統合」と呼ぶこととする。

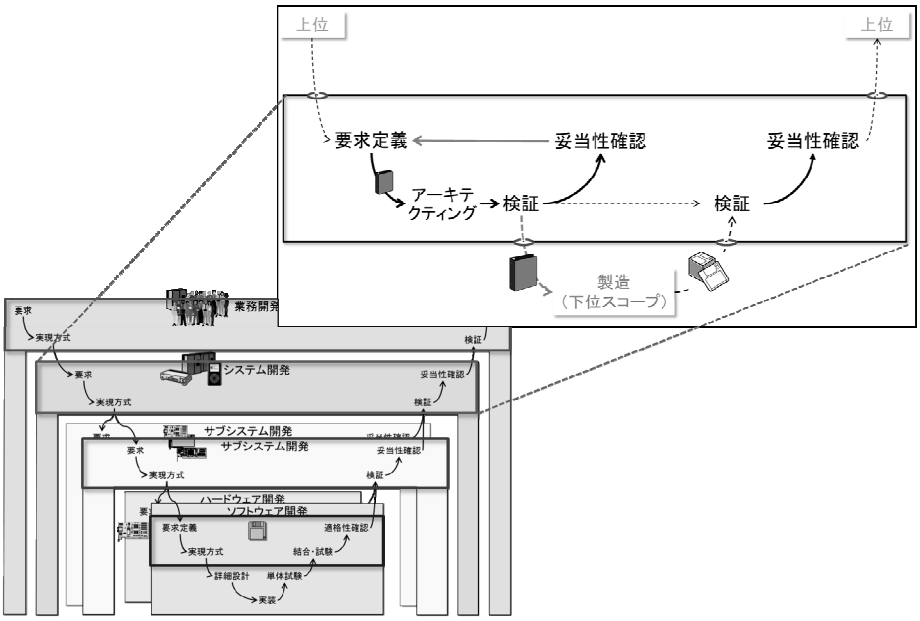


図 4.3-2　開発ライフサイクルプロセスの各スコープにある作業の概要

各階層のすなわち、各開発スコープの範囲で、アーキテクトは上位のスコープから与えられた要求を整理し、実現方法を定義して下位スコープの開発者に仕様を渡す。同時に、下位スコープから上がって来た成果物を集積・統合し、最初に決めた仕様通りに「正しく開発されたか」を確認する検証を行い、その上で与えられた要求に応える「正しい構成品を開発したか」を妥当性確認によって確認して、上位スコープの開発者に引き渡す。現実には､検証や妥当性確認を下位の成果物ができる迄待って行なうのは非現実的であるため、各スコープの中で要求の定義や実現方式の定義を行った段階で、検証や妥当性確認の活動を行い､「欠陥を充分に排除した」仕様を下位のスコープに引き渡すよう努力する。

この各開発スコープでのアーキテクトの活動に､4.2 のモデルを利用したソフトウェア・システムの開発技術をマッピングすると、図 4.2-6 のようにまとめることができる。すなわち、上位レイヤから「充分に欠陥が除去されたと期待される仕様」が与えられ、これを元に要求定義を行なう。この段階では、スケッチとしてのモデリングを行なうなどのモデリングが行われる。また、要求が曖昧な場合は、要求抽出を行なう目的でのプロトタイプをモデルベースで行なうのはひとつの策である。さらに、こうして定義された要求に対して実現方式を具体化する（アーキテクティングを行なう）。ここでは、品質要求の実現性を確認する目的でのプロトタイピングを行なうなどの策を講じることも可能である｡例えば、性能要求を満たすかを確認するモデルベースのプロトタイプは、この段階で適用可能である。要求を実現する実現方式が具体化されたら、この段階で策定された仕様の一貫性を確認するために、検証や妥当性確認を行う。より厳密な検証が必要であれば、レビューによる検証／妥当性確認だけでなく、4.2 で示したようにモデル検査などの技術を用いることも考えられる。このように上流フェーズで充分に仕様の不具合を排除出来たことを確認した仕様を、下位スコープの開発者に引き渡す。

一方、下位のスコープでの開発成果物が出来たら、これを集積し、まず定義した仕様通りのものが出来たか（i.e. 下位スコープの開発が正しく行われたか）を確認する検証の活動が行われる。仕様のモデル化が出来ていれば、上流フェーズで定義したモデルへの適合性を確認するために、モデルベーステストを行なうことも可能である。仕様との整合性との観点では、下位スコープに提供した仕様通りの成果物となっているか、また組み上げた時に当該スコープでの設計が正しかったかを確認するため、同じスコープでのアーキテク

チャ定義への確認を取ることが重要となる。最後に、効率性や信頼性などの品質要求を中心に当該開発スコープに提示された要求の「目的」を満たしているかを確認するため、妥当性確認のテストを行う。この妥当性確認では、開発した成果物のおかれる環境要因をモデル化し、その環境要因下で確実に所要の動作を満たすか、また環境要因が崩れた時どこまでソフトウェア・システムが耐えられるかを確認することも必要になる。このような確認の道具として、環境要因をモデル化しシミュレーションして開発したソフトウェア・システムを仮想的にテストする SILs、HILs などのツールを利用するのもひとつの策である。これにより、例えば、周囲の他のシステム構成品の作成が完了する迄の待ち時間を短縮し、あるいは統合時に明らかになる欠陥の要因切り分け工数を削減する、あるいは再現性の低い環境要因を仮想的に作り出すと行ったテストが可能になり、作業の効率化が図れる。

このように、一連のアーキテクトの活動の中でモデルを利用することにより、欠陥を下位スコープに出さない開発、欠陥を上位スコープに伝播させない開発のプロセス構築が可能になり、組込みシステム開発の効率化が期待できる。

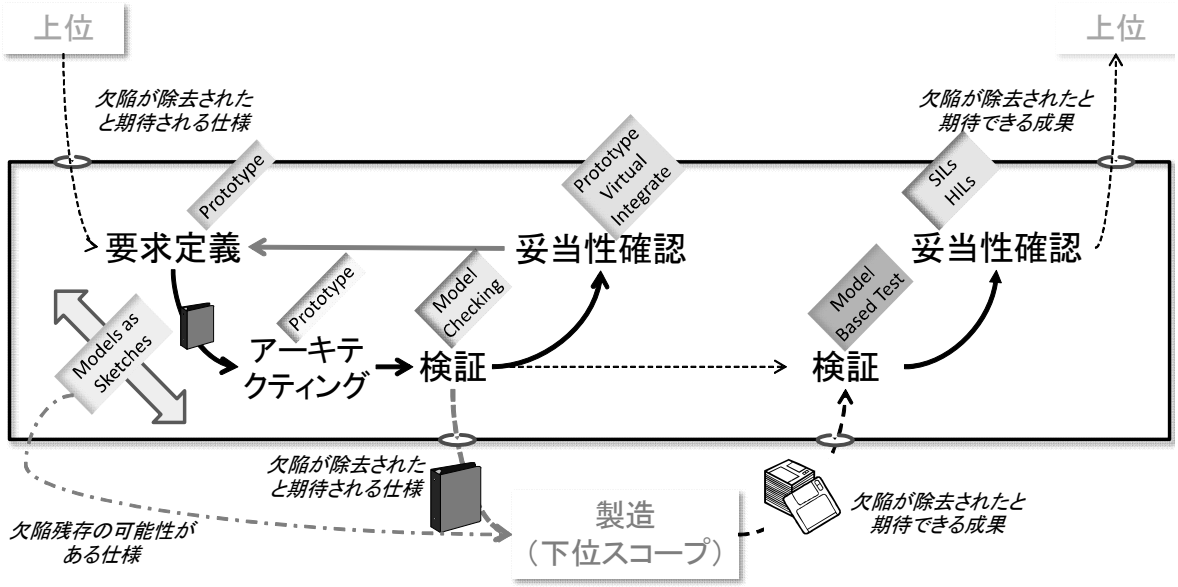


図 4.3-3　アーキテクティングとモデル化技術の対応

5. 組込みソフトウェア開発現場でのモデリング適用上の課題

今日、ソフトウェアを組み込んだ制御機器の開発では、単純にソフトウェアで実現する個々の機器の機能高度化、複雑化に留まらず、IoT（Internet of Things）に見られるように開発された機器を大規模に組み合わせたシステム開発、さらに、既存の制御システムを横断的に連携させた System of Systems の実現など、機器開発に対する要求の高度化と複雑化が急速に進んでいる。この結果、開発対象システムの構成品目数の増加と、個々の機器に搭載するソフトウェアの大規模化が同時に進行している状況にある。さらに、機能・性能の継続的な強化に対応するため、個々の製品のプロダクトライン化（系列製品化）も睨んだ組込み制御システム開発を推進する必要に迫られている。

これに対して、組込みソフトウェア開発の現場では、開発要員の大規模な増援や、既存ソフトウェア資産の活用、共通プラットフォームの導入で対応を図っている状況にある。特に開発要員の大規模な増援には開発のアウトソース化で対応することも増えている。実際、[MET04]では外部委託率が 8 割を超えるのが実状で、この数値は米国の 50%、欧州の 35%と比較しても大きいとの実状が示されている。既に、組込みソフトウェア開発現場は、従来の小規模な開発チームによる開発から、発注・受注の枠組みを前提にした協業型のソフトウェア開発への移行が強く求められる。現実にはこうした協業型のソフトウェア開発への移行は進まず、むしろアウトソース化が進む中でアーキテクチャ設計時の欠陥混入は識別されにくく、アウトソースから上がったソフトウェア成果物を統合する際にはじめて仕様理解齟齬やアーキテクチャ設計上の欠陥が明らかになり、リリース間際に大規模な手戻り作業に発展する状況も見られるようになっている。

同時に、現実の開発現場では、設計の複雑化に対応する取り組みも遅れている。4 章に示したように、ソフトウェア・システム開発の各段階で開発するソフトウェア・システムの適切なモデル化がなされれば、仕様理解の齟齬を抑制することも期待される。また、上流設計からモデルを利用した発注仕様の検証や妥当性確認も可能となる。しかし、こうしたモデルを活用した最新のソフトウェア・システム開発技術の導入はごく一部の開発現場に限られ、一部の経験豊富なエンジニアに依存した開発が旧態然として行われている現場も多い。それ以前に、組込みソフトウェア開発で重要な要素である制約事項や品質要求を具体化出来ていないなど、ソフトウェア開発者に開示する仕様が不足していることが多いのも実状である。

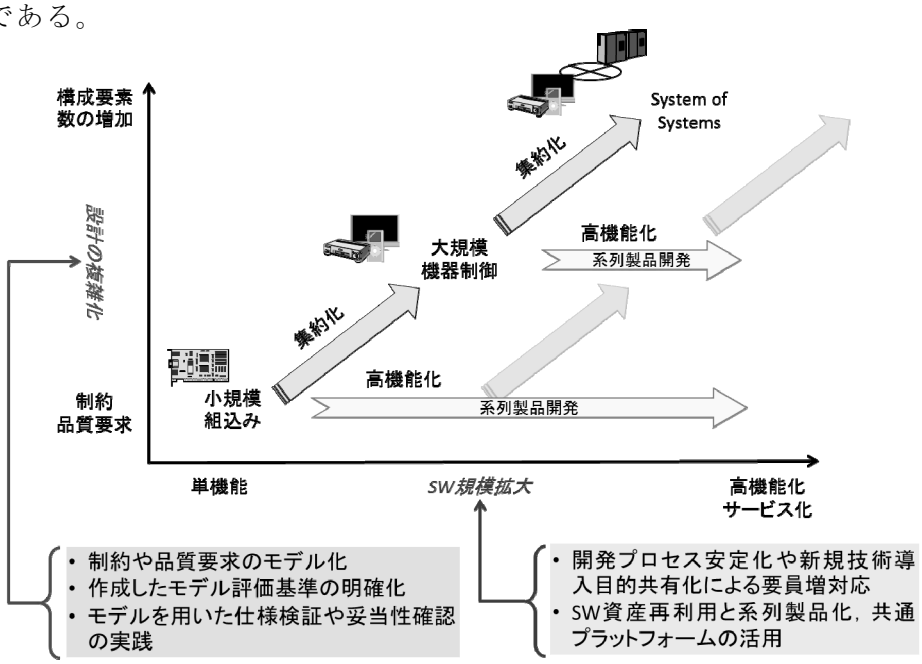


図 5.1-1　ソフトウェア組込み制御機器の市場動向

2014 年度は、こうした実状を踏まえて、組込みソフトウェア開発現場のアーキテクトの活動をどうすれば改善できるのかを、我々のテーマと置いた。まず委員会での議論から、図 5.1-1 に示した 5 つの課題があるという仮説を立て、これらの仮説の妥当性を確認するために、委員会内でのブレーンストーミングやワークショップ参加者からのアンケート分析を行った。

初期の段階で、我々がおいた仮説は以下の 5 点である。

・ 組込みソフトウェア開発で重要なインプットとなる制約や品質要求がモデル化されていない
・ モデル開発のスコープに応じたモデル評価基準が設定されていない
・ 仕様の検証や妥当性確認にモデルを活用していない
・ 開発要員増や開発のアウトソース化が進む中で、技術導入の目的が明確化されていない
・ 開発現場で急激に広がっているソフトウェア資産再利用や共通プラットフォーム化を想定したモデル化技術が未成熟である

以下、各仮説の詳細を示す。

5.1 制約や品質要求がモデル化されていない

ソフトウェア組込み制御機器は、制御対象となる外部機器やハードウェアがソフトウェア設計に与える制約も意識した開発が求められる。例えば、筐体などの固有振動数を考慮した制御を行うなどの制御対象ハードウェアの特性は、ソフトウェア開発部門に漏れ無く伝えられ、これを踏まえた機器制御方式が具体化されることが求められる。

また、これらのソフトウェア組込み制御機器の開発では、電源を入れたら常に同じように動作する、制御対象に対して適切なタイミングで制御を行う、有限の計算機リソースの中で充分な反応速度を持って動作するなど、単純に「要求された機能」を実現すれば良いソフトウェアではない。顧客がソフトウェア組込み制御機器に期待する「所期の導入目的」を満たすよう、機器開発を担当するシステム部門は、機器に組み込まれるソフトウェアの開発部門に、これらの要求を品質要求[JIS03][JIS13]（図 5.1-2）に展開し具体化して伝えることも求められる。ソフトウェアやシステムの開発者は、これらの品質要求や制約に応えるよう、システムアーキテクチャ、ソフトウェアアーキテクチャを選定する。組込み制御ソフトウェアの開発では、一般に、時間効率性や資源効率性、障害許容性、回復性などの品質要求が特に具体化されるべき品質要求であり、開発現場ではこれらの品質要求の実現のために多大な労力をかけている状況にある。

ソフトウェア開発では、これらの品質要求や制約に対する解決策はアーキテクチャ設計での実現方式検討でほぼ決定される。しかし、現実には、これらの品質要求や制約がソフトウェア要求として、ソフトウェア開発部門に明示されないことも多い。この結果、ハードウェアとソフトウェア各々が揃った後の統合の段階でシステムとしての仕様の問題が明らかとなり、不具合改修のための大きな手戻りを招くことになる。同時に、これらの欠陥の原因は「ソフトウェア部門の仕様不理解」で片付けられることすらある。

一方で、現在主流のモデリング言語である UML 2.0 の範囲では、品質要求や制約の記述を受け入れる枠組みが充分に示されていないも事実である。例えば、UML 2.0 では、入出力イベントの時間間隔や時間制約は記述できる[OMG09]ものの、これ以外の品質要求を記述する場合は UML2.0 から導入されたステレオタイプを利用して UML の言語体系を拡張することが期待されている。組込みシステム開発向けのステレオタイプとして UML MARTE profile[OMG11]が標準化されてはいるが、そこで定義されているステレオタイプの数は膨大で、逆に開発現場でこの profile を使いこなすことも難しいのが実状である。また、既存の市販 UML ツールで MARTE profile に対応したモデル作成や、この profile を利用して作成したモデルのチェックを支援するものはなく、このためモデル作成者が自ら MARTE profile で定義するステレオタイプを個別に入力し、明示的に品質要求を書き込むことが求められ

ている。多忙なソフトウェア開発現場でここまでの取り組みを行って厳格に品質要求を書き下すのに時間がかかることや品質要求の一貫性や無矛盾性を抽出する良い仕様検証の手法もないことなど、ツールの面からも品質要求や制約を網羅的に矛盾無くモデル化することを支援する環境がない状況にあるのも事実である。

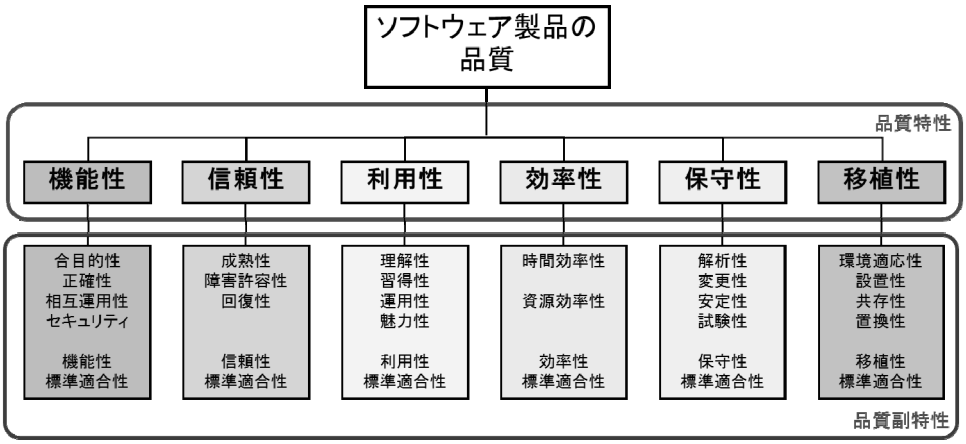


図 5.1-1　JIS-X0129/ISO 9126 品質特性モデル

5.2　モデリングスコープに対応したモデル評価基準が設定されていない

4 章で紹介した Zachman Framework[ZAC87]などのアーキテクチャフレームワークの考え方に則れば、モデリング作業は本来、各開発フェーズの目的に沿って数回繰り返されるものとなる。ここで開発フェーズの目的に基づいて、例えばシステム導入の目的、企業の業務、システム全体、ソフトウェア全体、そしてソフトウェア・モジュールなど、モデリングする範囲（スコープ、以下これをモデリングスコープと呼ぶ）が設定される。このモデリングスコープを切り替えながら、モデルの記述は段階的に詳細化され、最終的に実装コードに相当するモデル迄具体化される。また、各モデリングスコープでは、What, Where, Who, How, When といった複数の観点からモデルが作成され、各モデリングスコープ内で複数のモデルの記述内容の一貫性が確認されることが求められる。この Zachman Framework のモデリングスコープと SLCP に定義された開発フェーズの対応付けを取ると、図 5.2-1 のように整理できる[1]。

[1] なお、図 5.2-1 では 4.3 での議論を踏まえて、開発フェーズ名は ISO 12207 SLCP のそれ[IPA07b]とは揃えていない点に注意すること。業務開発、システム開発、ソフトウェア開発の相似性を意識して開発フェーズ名を改変した。

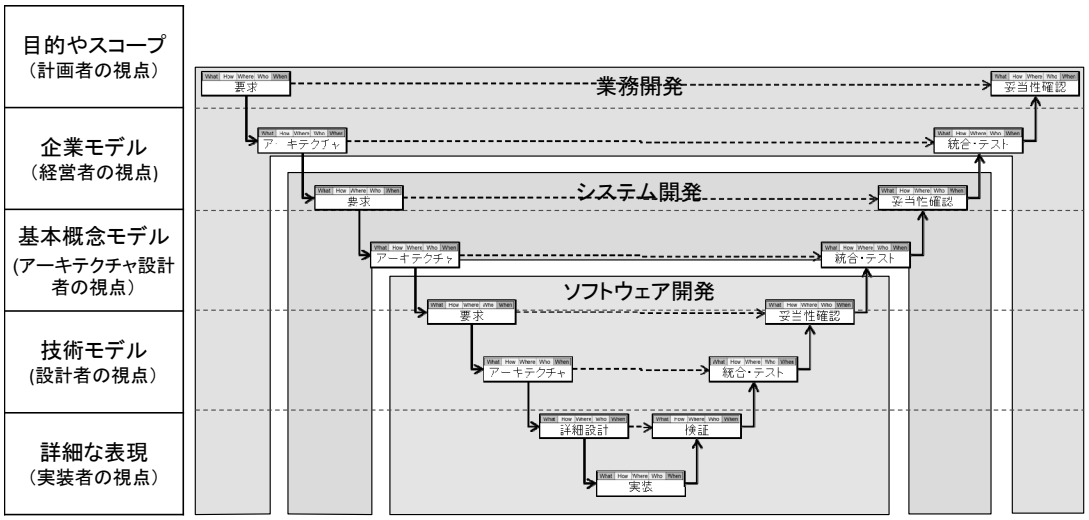


図 5.2-1　Zachman Framework と SLCP 開発フェーズの対応付けの例

組込みソフトウェアの開発現場でも、C++や Java などのオブジェクト指向プログラミング言語の利用が広まる中で、UML を利用して実装コードの構成をモデル化するようになっている。しかし、その上位のモデリングスコープでは、このようなモデルの段階的な詳細化が曖昧にされやすい。この結果、例えば、

- 基本概念を具体的に示す目的で What をモデル化する際に作成したデータモデル上に、実装上の制約に起因する属性の型情報を記述する

  データモデル上の属性に早い段階から実装上の型情報を与える行為は、それが（例えば、過去の既存システムとの互換性を確保するなどの）既定の制約事項として要求されるならともかく、通常は他の要求の実現方法などとのバランスから設定されるべき項目であり、少なくともビジネスアーキテクチャを具体化するときに定義される必然性は低い。

- システムのアーキテクチャを明らかにする目的で How をモデル化するために作成したイベントフロー図の量が膨大になり、一方でエラーケースの動作定義に見落としが発生する

  イベントフロー図は、本来インスタンス間のイベントの相互作用を、ある条件の元で動作させたスナップショットの記述であり、本来は、いくつかのイベントフロー図を元にクラスの動作仕様として状態図の形で整理しなおし、動作仕様としてエラー状態を含めて一定の網羅性を確保する必要がある。

など、「将来不具合を招き込みやすい」モデリングが行われる状況が散見される。これらは、モデリング言語の言語仕様上は「正しい表記」であっても、各開発フェーズの本来のモデリングスコープとは異なるモデルを作成している可能性が高い。すなわち、モデル作成者が作業の目的（Process Purpose）を正しく意識せずにモデリングを進めていると捉えられる。

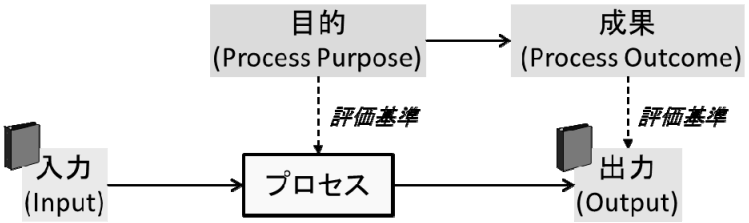


図 5.2-2　開発プロセス記述の構成要素と関係

ここで、作業の目的（Process Purpose）は、Automotive SPiCE の PRM[AUT10]などの記述が参考になる。図 5.2-1 に、[AUT10]や[IPA07c]の記述様式を参考に、開発プロセスの記述要素をまとめたものを示す。Automotive SPiCE では、各作業フェーズに対して、作業の目的（Process Purpose）と、作業の成果（Process Outcome）を明示的に示している。これらは、作業そのものや、プロセスのアウトプットに対する評価基準として位置づけることができる。Automotive SPiCE[AUT10]では、各開発プロセスの作業の目的と成果を表 5.2-1 に示すように定義している。

表 5.2-1　Automotive SPiCE 2.5 PRM での各プロセスの目的と出力

プロセス	作業の目的	成果
要件抽出 (ENG1)	必要な作業成果物を定めるための基盤としての役目を果たす要件ベースラインを確立させるために、発展する顧客ニーズ及び要件を、製品、及び／又は、サービスの生涯を通じて収集、加工、及び追跡すること	・顧客との継続的なコミュニケーションが確立される ・合意された顧客要件が、定義され、かつベースライン化される ・変化する顧客ニーズに基づき、顧客要件に対する変更を評価し、ベースライン化された要件へと変更を組み込むための、変更のメカニズムが確立される ・顧客ニーズを継続的に監視するためのメカニズムが確立される ・顧客が彼らの依頼の状況及び最終処理を容易に判断できることを保証するためのメカニズムが確立される ・技術の変化及び顧客ニーズから生じる変更が識別され、関連するリスクが評価され、またその影響が管理される
システム 要件分析 (ENG2)	定義された顧客要件を、システム設計へ導くであろう、要望されたシステム技術要件の集合へと変換すること	・定義されたシステム要件の集合が確立される ・システム要件が、正当性及びテスト可能性について、分類かつ分析される ・運用環境におけるシステム要件の影響が評価される ・システム要件の優先順位が定義される ・システム要件が必要に応じて、承認かつ更新される ・顧客要件及びシステム要件間での一貫性及び相互のトレーサビリティが確立される ・顧客要件ベースラインへの変更が、費用、スケジュール及び技術的影響について評価される ・システム要件が、全ての影響を受ける関係者に連絡され、かつベースライン化される
システム アーキテクチャ 設計	システムのどの要素に対して、どのシステム要件が配置されるかを識別すること	・システムの要素を識別し、定義されたシステム要件を満たすシステムアーキテクチャ設計が定義される

(ENG3)		・システム要件がシステム要素に割り当てられる ・各システム要素の内部及び外部インタフェースが定義される ・システム要件とシステムアーキテクチャ設計間の検証が実施される ・システム要件及びシステムアーキテクチャ設計間での一貫性及び相互のトレーサビリティが確立される ・システム要件、システムアーキテクチャ設計、及びそれらの関係がベースライン化され、全ての影響をうける関係者に連絡される
ソフトウェア要件分析 (ENG4)	システムのソフトウェア要素のソフトウェア要件を確立すること	・システムのソフトウェア要素及びそのインタフェースに対して割り当てられたソフトウェア要件が定義される ・ソフトウェア要件が、正当性及びテスト可能性について、分類かつ分析される ・運用環境におけるソフトウェア要件の影響が評価される ・ソフトウェア要件の優先順位が定義される ・ソフトウェア要件が必要に応じて、承認かつ更新される ・システム要件及びソフトウェア要件間での一貫性及び相互のトレーサビリティが確立される ・システムアーキテクチャ設計及びソフトウェア要件間での一貫性及び相互のトレーサビリティが確立される ・ソフトウェア要件への変更が、費用、スケジュール及び技術的影響が評価される ・ソフトウェア要件が、ベースライン化され、かつ全ての影響を受ける関係者に連絡される
ソフトウェア設計 (ENG5)	ソフトウェア要求事項を実装し、かつそれに対して検証されるソフトウェアについての設計を提供すること	・ソフトウェア構成要素を識別し、定義されたソフトウェア要件を満たすソフトウェアアーキテクチャ設計が定義される ・ソフトウェア要件が、ソフトウェア要素に割り当てられる ・各ソフトウェア構成要素の内部及び外部インタフェースが定義される ・ソフトウェア構成要素の動的動作及び資源消費目標が定義される ・実装及び試験可能なソフトウェアユニットを記述した詳細設計が策定される

		・ソフトウェア及びソフトウェアアーキテクチャ設計間での一貫性及び相互のトレーサビリティが確立される ・ソフトウェアアーキテクチャ設計及び詳細設計間での一貫性及び相互のトレーサビリティが確立される
ソフトウェア構築 (ENG6)	ソフトウェア設計を適切に反映した、検証されたソフトウェアユニットを生成すること	・ユニット検証戦略が定義される ・設計にて定義されたソフトウェアユニットが作成される ・詳細設計及びソフトウェアユニット間での一貫性及びに相互のトレーサビリティが確立される ・ユニット検証戦略に従い、ソフトウェアユニットの検証が実施される ・ユニット検証の結果が記録される
ソフトウェア統合テスト (ENG7)	ソフトウェア設計と整合性のとれた統合ソフトウェアを生成しつつ、ソフトウェアユニットをより大きな構成へと統合し、かつソフトウェア品目間の相互作用をテストすること	・ソフトウェア設計と整合性の取れたソフトウェア品目のために、ソフトウェア要件の優先度及び分類に基づいたソフトウェア統合及び統合テスト戦略が策定される ・各ソフトウェア品目に割り当てられたソフトウェアアーキテクチャ設計、ソフトウェア詳細設計との適応を確認するために、ソフトウェア統合テスト仕様が策定される ・統合戦略の定義に従い、ソフトウェアユニット及びソフトウェア品目が統合される ・統合されたソフトウェア品目が、テストケースを用いて検証される ・ソフトウェア統合テストのテスト結果が記録される ・ソフトウェアアーキテクチャ設計、詳細設計、及びテストケースを含むソフトウェア統合テストのテスト仕様間での一貫性及び相互のトレーサビリティが確立される ・ソフトウェア品目(関連要件、設計及びコードを含む)に変更が生じた場合に、ソフトウェア品目の再統合及び再検証のために、リグレッションテスト戦略が、策定され適用される
ソフトウェアテスト (ENG8)	統合ソフトウェアが、定義されたソフトウェア要件を満たすことを確認すること	・統合ソフトウェアをテストするために、ソフトウェア要件の優先度及び分類に基づいた戦略が策定される ・ソフトウェア要件との適応を示す統合ソフトウェアのためのソフトウェアテスト仕様が策定される ・統合ソフトウェアが検証される

		・テスト結果が記録される ・ソフトウェア要件及びテストケースを含むソフトウェアテスト仕様間での一貫性及び相互のトレーサビリティが確立される ・ソフトウェア品目に変更が生じた場合に、統合ソフトウェアの再検証のために、リグレッションテスト戦略が、策定され適用される
システム統合テスト (ENG9)	システムアーキテクチャ設計及びシステム要件にて表明されている顧客期待を満足する統合システムを生成するために、システム要素を統合すること	・システムアーキテクチャ設計と整合性の取れたシステム品目のために、システム要件の優先度及び分類に基づいたシステム統合及びシステム統合テスト戦略が策定される ・システム要素間のインタフェースを含む、システムアーキテクチャ設計との適応を検証するために、システム統合テスト仕様書が策定される ・統合戦略の定義に従い、統合システムが統合される ・統合システムが、テストケースを用いて検証される ・システム統合テストのテスト結果が記録される ・システムアーキテクチャ設計及びテストケースを含むシステム統合テスト仕様間での一貫性及び相互のトレーサビリティが確立される ・変更が生じた場合に、システム要素の再検証のために、リグレッションテスト戦略が策定され、適用される
システムテスト (ENG10)	各システム要件の実装の適合性がテストされており、システムは出荷できる状態にあることを保証すること	・システムをテストするたに、システム要件の優先度及び分類に基づいた戦略が策定される ・システム要件との適応性を証明する統合システムのためのシステム統合テスト仕様が策定される ・統合システムが、テストケースを用いて検証される ・テスト結果が記録される ・システム要件及びテストケースを含むシステムテスト仕様間での一貫性及び相互のトレーサビリティが確立される ・システム要素に変更が生じた場合に、統合システムの再検証のために、リグレッションテスト戦略が、策定され適用される

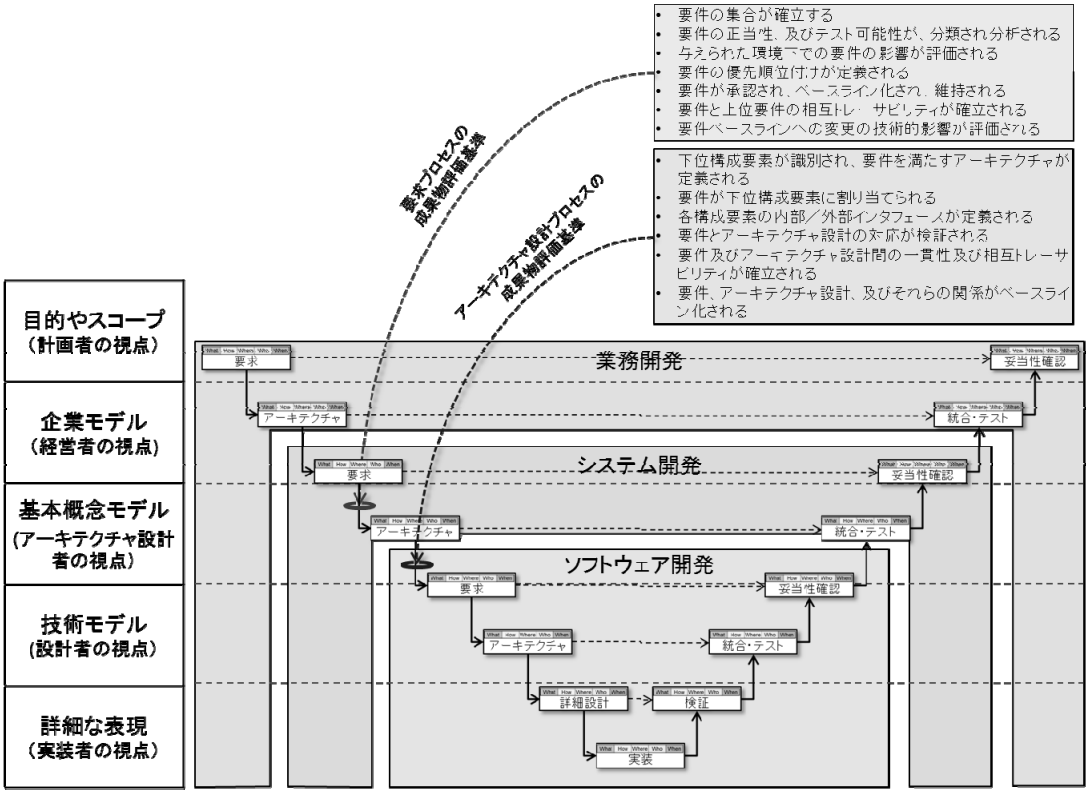


図 5.2-3　Zachman Framework、SLCP と成果物評価基準の対応関係

従って、各開発プロセスの成果物に対する評価基準（Automotive SPiCE のプロセス記述にある成果）は、例えば図 5.2-3 のように整理できる。ここに示した成果物の評価基準を満たすようなモデリングが開発現場で行われているかが議論のポイントとなる。なお、図 5.2-3 では、成果（Process Outcome）の記述は、開発プロセスのスコープでの相似性を考慮して、スコープに関する記述を敢えて抽象化している。従って、実際に利用する場合は、Zachman Framework のスコープの記述を加えて、モデリングスコープを意識した「モデルの評価基準」として利用することを想定している。

現実の開発現場ではこの評価基準の記述を元に、さらに開発するソフトウェア・システム毎に、その製品に期待される品質特性や要請される制約を勘案して、各開発フェーズの成果物となるモデルの評価基準を具体化することが求められる。各モデリングスコープでの、アーキテクトのモデリング作業とそれに対応した評価基準の具体化の一例を、表 5 に示す。ここでは、アーキテクティングの活動を抽象化して記述をしていることもあり、まだ開発現場でそのまま利用できるような「モデルに対する評価基準」とはなっていない点は注意すること。　実際には、個別のシステム毎の品質特性を反映して、当該モデリングスコープでどこまでモデルを作成するか、どこまで具体化していれば良いか、逆にどこまで具体化してはいけないか、という基準を、プロジェクト毎に設定しておくことが望ましい。

表 5.2-2　あるモデリングスコープの中でのモデルの評価基準の例

	Automotive SPiCE の成果	モデルの評価基準(例)
要求	定義された要件の集合の確立	・開発対象構成品目の利用目的に照らして、必要とされる要求(機能要求、品質要求)を全てモデル化していること 現実に、どの品質要求が重要になるか

		は、製品毎に異なる。顧客や最終ユーザのソフトウェア・システムの導入目的に応えるように、必要な要求が何かを最初に具体化する必要はある。これを基準に、要求事項を網羅するようにモデルを作成する。品質要求の抜けを防止するため、ISO 9126[JIS03]などの品質特性モデルを基準に、要求の有無を確認するのもひとつの方策である。なお、機能毎に期待される品質要求は異なるため、機能対応で品質要求を具体化してモデルに記載して置くことが望ましい。
	要求の正当性及びテスト可能性の分類と分析	・要求の正当性やテスト可能性が具体的にモデルに対応付けて記載されていること 要求の正当性は、顧客やユーザを巻き込んで、例えばプロトタイプやレビューなどの設計段階での妥当性確認を通して実施する。これらの活動に利用できるモデルを用意する。あるいは、上位の要求との整合性を見るため、クロスリファレンスの利用、SysML の要求図のような要求事項の根拠をモデルに記載する方法などもある。 要求のテスト可能性は、有限時間で所期の要求が実現したかを確認できること[IEE98]が重要になる。モデル中に、要求の実現を評価する基準を明記する。特に、品質要求は、要求が実現されたと判断するための、計測可能な基準値を設定し、モデルに明記しておくことが推奨される。
	運用環境における要求の影響評価	・周囲の運用環境と開発対象構成品目との相互作用をモデル化していること 必要に応じて、開発対象構成品の周囲の運用環境そのものに対する仮定や期待事項を含めてモデル化し、これと開発対象構成品のモデルとの相互作用を評価する（シミュレーションする）。ここで、仮定や期待事項は、しばしば裏切られることもあるので、要求の正当性の分析の中で確認を取ることも重要になる。
	要求に対する優先順位付け	・モデル記述の中に、個々の要求に対す

		る優先順位付けを記載していること 要求事項の優先順位付けは、UML のステレオタイプなどで明記するアプローチや、別に要求の優先順位を整理した一覧表を用意する方法が考えられる。
	一貫性及び相互のトレーサビリティの確立	・モデルの記述内容の一貫性を確保すると同時に、モデル記述の中に上位仕様とのトレーサビリティ情報を記載していること 仕様の一貫性を確認するには、モデル化された仕様記述中の矛盾や齟齬を排除する活動をする。一般にはレビューで、可能であればモデル検査などの仕様検証の技法を活用して、モデル記述の一貫性が確保されていることを確認する。ここで、一貫性は、Zachman Framework で示したような What, Where, Who, When, How などのモデル化観点内での一貫性、及びこれらのモデル化観点間での一貫性を確認することが求められる。なお、一貫性の確認対象は、機能要求、品質要求全てを包含する点に注意すること。 一方、トレーサビリティは、特に段階的に詳細化されたモデル間でのトレーサビリティの確保を行う。重要なのは、モデリングスコープを越えた仕様の一貫性を確認し、同時に段階的詳細化の過程で詳細化の漏れが起きていないかを確認することにある。
アーキテクチャ設計	構成要素の識別と上位から渡された要求を満たすアーキテクチャ設計の定義	・モデリングスコープでの構成要素をモデル上に記述し、これらの関係付けを具体化する 何を構成要素として切り出すかは、対象とする製品によって異なる。組込みソフトウェア開発では、候補としては、筐体やボードというハードウェア、タスクやプロセスなどの実行単位、そしてソフトウェアそのもののモジュールなどがこの構成要素となる。各々で、モデリングスコープが段階的に詳細化されている点は注意が必要である。

		モデル化に際しては、個々の構成要素として何があり、どの構成要素と連携動作するか、その関係を具体化したモデルを、各々のモデリングスコープで明らかにすることになる。
	上位要求の構成要素への割り当て	・同じモデリングスコープの要求モデルで現れた要求を、上に示した構成要素へ割り当て、構成要素の役割、責任範囲を具体化する ここで要求として、機能要求と品質要求の双方を分割、詳細化し、当該モデリングスコープで洗いだした構成要素へ全て割り当てる。この結果、個々の構成要素の役割と責任が具体化されることになる。 これらの構成要素の動作の並列性を考慮するのもこの時点での作業になる。これに併せて、共有する計算機リソースへのアクセス方法も方針として決めて置く必要はある。
	構成要素の内部及び外部インタフェースの定義	・当該モデリングスコープで設定した構成要素と構成要素の役割と責任に基づいて、構成要素間の相互作用をモデル化、この結果を元に構成要素間の内部インタフェース、及び周囲の構成要素とのやり取りをする外部インタフェースを定義する。 当該モデリングスコープの範囲で、これ以前に設定した構成要素と構成要素に割り当てた役割と責任に基づいて、具体的に構成要素間の相互作用をモデル化する。ここでは、情報交換やイベントのやり取りなどの、構成要素同士の相互作用の順序とやり取りする必要があるデータを具体化する。これらの相互作用を構成要素が外部へ公開するインタフェースとして仕様にまとめる。この際、モデリング対象としているソフトウェア・システムの置かれる環境とのインタフェース(外部インタフェース)も具体化する。 ここで、相互作用の途中でエラーが発生した時の振る舞いなども具体化し、インタフェース仕様に反映しておく。さらに、情報やイベントの交換に際して守るべき品質仕様もモデルに含めて置く。

		最終的に、構成要素の動作は、構成要素の取るべき(外部から伺える)状態遷移仕様としてモデル化される。
	上位要求とアーキテクチャ設計間の検証	・モデルを利用して、上位要求を満たしたアーキテクチャが実装されていることを確認する 当該モデリングスコープの範囲で、モデル化された仕様の一貫性を確認する。また、上位のアーキテクチャ設計で決定した事項と、本モデリングスコープで決定した事項が矛盾を起こしていないかを確認する。手段としては、レビューやモデルを利用した仕様検証などの手段を利用する。
	上位から渡された要求とアーキテクチャ設計間での一貫性及び相互のトレーサビリティの確立	・モデルを利用して、要求との一貫性、アーキテクチャ設計内での一貫性を確認する。また、上位要求との対応関係をトレーサビリティ情報としてまとめ、上位要求が全てアーキテクチャによって実現されていることを確認する。 一貫性の確認は、レビューやモデルを利用した仕様検証によって行う。 一方、トレーサビリティは、モデリングスコープを越えた複数のモデル間で対応関係が取れて、必要な事項が当該モデリングスコープで適正に扱われ、モデルに反映されていることを確認する。

組込みソフトウェアやソフトウェア組込み機器の複雑さが増大し、同時にソフトウェア開発要員が膨張する経営環境下にある時は、本来、開発組織毎、あるいはプロジェクト毎に、ここに示したような議論を踏まえて、モデル（仕様）記述の詳細化のステップの設定、各々の詳細化ステップでのモデルで決定されるべき内容、逆にそれ以降の詳細化されたモデリングスコープで決定すべき内容などの「作成されたモデルに対する記述詳細化の評価基準」を決めて置くことが、チームで作成するモデルの均質性を保ち、仕様決定漏れを抑止する観点からは望ましい。このような基準を設定することは、開発の一部アウトソース化の際も、モデルの記述レベル安定化に繋がるはずである。さらに、プロジェクト管理の観点からは、例えばモデルの詳細化が進まない部分が見つかれば、それは未決事項が残っている可能性を考慮し、リスク管理の対象とするなどの対策も取れるようにもなる。

現実には、ここまでの検討を踏まえて、各モデリングスコープに対応したモデルに対する評価基準を設けていること、外部委託先とも共有していることはまれと予想される。その結果、モデルを作成するエンジニア、アーキテクト毎に、作成されるモデルの記述詳細度が揺らぐことになる。また、モデリング基準が不明確なため、プロジェクト管理活動との連携も意識しないままアーキテクトがモデル化を進めている現状も多く、この結果、モ

デル化は行ったものの、作成されたモデルそのものに曖昧さが内包され、ソフトウェアの欠陥に繋がるような事態もあると想定される。結果的に、現場エンジニアが手間をかけて正確なモデルを作製するだけ時間の無駄と考えるようになり、開発対象ソフトウェア・システムに対するモデル化作業が定着しない可能性も考えられる。

5.3 仕様の検証や妥当性確認にモデルを活用しにくい

開発するソフトウェア･システムの複雑化が進み同時にアウトソース化が進む環境では、図 5.2-3 に示したモデリングスコープの各々の段階で、作成した仕様（モデル）が一貫性を持ち欠陥を充分に排除されたことを確認して、下位のモデリングスコープの開発者にモデルを引き渡すことが重要となる。これは、下位スコープの成果物を統合して、大きなシステムとして組上げていく際の欠陥検出と上流へ戻っての手戻りを減らすための条件となる（4.3 参照）。

現実の開発現場では、事前にこうした取り組みの必要性が強く意識されていないことは多い。この結果、下位スコープの成果物が出来上がった後、しばしばリリース間際に大きな欠陥が発見され、上位設計からのやり直しや、それに伴う開発要員の大規模な追加投入など、プロジェクトに混乱を齎す施策で乗り切っているのが実状である。上流設計での不具合排除を強く意識していたとしても、その手段は成果物のレビューに頼ることが多く、複雑化するソフトウェア・システムの仕様の「量」に耐えられなくなり、欠陥の見落しに繋がることも多いのが実状である。

この課題に対しては、モデル検査やモデルベースのプロトタイプなど、モデルを利用した開発技術での対応も検討されている。検証作業の自動化や、妥当性確認作業での「モデルを利用した仕様の見える化」を推進することで、仕様に対する理解齟齬やアーキテクティングで混入する欠陥の排除を効率化することが期待できる。しかし、導入しようとする CASE ツールやモデリング言語の制約と、モデルを利用した新規開発技術の導入目的が齟齬を起こしており、意図した通りに新しい開発技術を適用出来ないという問題もある。これは、例えば以下のような現象として観測される。

① 仕様検証の道具として期待されるモデル検査技術は、ツールの制約により、状態爆発がおきないようにモデルを再構成する必要がある

モデル検査は Felica の開発[TAD10]や自動改札機の開発[HAT12]など、特にソフトウェアの欠陥が大きな問題となる LSI 開発や機器組込み制御ソフトウェア開発での適用事例が報告されており、徐々に実システム開発での適用が広まりつつある。

モデル検査ツールは、様々な条件に対して網羅的に状態空間探索を行い、特定の状態に至るかの到達可能性を分析することで仕様の正しさを確認する。従って、4.2 (2)③でも触れたように、状態空間の定義によっては状態組合せの数が爆発し、モデル検査ツールを動かす計算機環境によっては、実用的な時間で仕様を検証出来なくなることが多い。このような場合には、モデル検査ツール用にモデルを修正する必要があるとされている。この結果、ひとつのソフトウェア・システム製品に対して

- 実際の製品の仕様を反映したモデル
- モデル検査用に製品仕様を一部簡略化したモデル

と、2 種類のモデルを準備する必要が出てくる。開発現場では、仕様変更もしばしば起きるため、これに対応して複数のモデルを常に維持する必要も出てくる。複数のモデルに対する改変作業は、しばしば同期しなくなる可能性もあり、この結果、ひとつの製品に対するモデルの一意性が損なわれる懸念も出てくる。

② コード自動生成機能を持ったツールは、クリティカルな動作に対してどのような保証をするのかが不明な点がある

モデルからコードを自動生成する MDA ツールの多くは、状態遷移モデルをもとに実

行コードを生成する。状態遷移そのものは、ツールが提供する状態遷移エンジン（ライブラリ）に隠蔽され、ユーザ作成のコードを呼び出す構成を取ることが多い。

一方、組込み制御ソフトウェアの場合、しばしば、並行動作を実装する必要が出てくる。この場合、それに伴う共有資源へのアクセス競合設計が性能問題として現れることは多い。このような並行動作に関わる詳細な設計は、コード生成の元とするモデルに詳細に定義しておかない限り、ツールが提供する状態遷移エンジンによって決定されることになる。

これは、クリティカルな状況で確実にソフトウェアを動作させる必要がある組込みソフトウェア開発では問題になることがある。結果的に、モデル生成のために、コード設計と同等レベルのモデルを作成するなどの対策が必要になり、モデルが肥大化、レビューやモデル検査によるチェックも困難になる可能性が高い。

これらはいずれも、導入する CASE ツールやモデリング言語の制約によって、開発現場で作成されるモデルが歪められるという課題を引き起こす。各々のツール上で、モデルを「動かすため」のモデルの改変が行われ、その結果

・ 各開発プロセスの本来の目的を逸脱したモデルを準備しなければならなくなる
・ ひとつの作業に対して、複数のモデルを多重に用意し、また維持する必要性に迫られる

といった問題を引き起こす。「開発対象ソフトウェア・システムを特定の視点から抽象化して整理する」というモデル化の本来の目的を損ない、同時にソフトウェアに対する欠陥を招きこむ可能性も高くなる。

5.4　新規開発技術導入の目的を明確化していない

規模の拡大に対するもっとも単純な対策は、開発要員の増援によるチームとしての開発スループットの向上にある。こうした開発要員の増加している環境下で、同時に開発対象のソフトウェア・システムの増大する複雑さに対応するため、モデリング言語やモデルを利用した新規の開発技術・ツールの導入を進めることは良く見られる現象である。

一方で、これらの新しい開発技術の導入にあたって、技術導入の目的を曖昧にしたままである状況も散見される。大規模な開発プロジェクトになるほど、開発メンバ全員が導入する新しい開発技術に併せた作業成果物を作成するため、新規開発技術の導入に際して、解決したい課題を明らかにし、それに応じてどの開発フェーズでどのような手法やツールを導入するかを明確にすることが必要となる。しかし、これらの解決したい課題を明らかにし、これに導入しようとする新規開発技術がどのように効果をもたらすかを具体化しないまま、新しい開発技術を漫然と導入すると、本来解決したいソフトウェア・システム開発の課題も解決されないままで終わることになる。

例えば、ソフトウェアの規模拡大に対して、生産性改善を狙ってモデル駆動型開発ツールによるコード自動生成を導入しようとする事例はこれにあたる。COCOMO の見積りモデル[BOE81][BOE00]などを用いてシミュレーションすると、開発規模が膨らめば人月辺りの生産性は下がるとされる。これは、開発メンバ間での仕様調整などの作業コストが膨らむこと、仕様齟齬に伴う手戻りが増えることなどが要因と考えられる。従って、開発現場では、こうした状況が観測されたら、解決すべき真の課題（生産性低下の具体的な要因）を具体化し、それに応じた施策をすべきである。例えば、設計不備などの「手戻り作業」に起因するものか、あるいは仕様調整の非効率性に起因するものかで、その改善の施策は異なって来る。また、同じ設計不備に関しても、ユーザや顧客との要求仕様調整不備に起因するものか、あるいは開発チーム内の設計で埋め込んだ欠陥が起因なのかでも、対応施策は異なる。これらを、一律、コード自動生成を行えば解決するという判断で、新しいツールや開発技術を導入しても、5.2 で示したように、本来の開発フェーズの目的と異なるモデルを作成する状況を誘発する。

例えば、“スケッチとしてのモデル”を日常的に実施している開発現場に、同じモデリング言語を使っているという理由だけで、モデル駆動開発ツールを導入する状況はこれにあたる。“スケッチとしてのモデル”は、開発者がアイデアを整理するため、特に関心を持った範囲をモデル化する。作成されたモデルは、仕様としての一貫性、完全性を満たす必要はない。一方、実機上で動作するコードをツールから自動生成する場合、結果的には「ハンドコーディングと同等の詳細で一貫性を持ったモデル」の作成が求められる。例えば、並行状態遷移をどのような実装コードに展開するか、リソース競合を回避するためにどのような機構を持ち込むかなどの実装上の配慮もモデル化の対象となる。この結果、図 5.2-3 に示した「段階的な詳細化」のステップを踏まずに詳細な実装レベルのモデルを作成することになる。動くモデルを作成する迄に大きな工数がかかると同時に、出来上がったモデルには欠陥が混入される可能性も高い。この結果、生産性向上を狙ったツール導入により、開発工数が膨張する可能性も出てくる。同時に、こうして作られた実装レベルのモデルは、そのままでは、モデルベースの検証や、要求の妥当性確認のためのモデルベースのプロトタイプに用いることは困難になることも多い。前者に対しては、詳細な実装レベルのモデルを作成してしまったためにモデル検査での状態爆発を引き起こす。後者では、既に実装レベルの意思決定をモデルに反映してしまったために、特に要求に対する理解を確認するという目的で期待される

・ ターンアラウンドを短縮して素早く動作するプロトタイプを作り直すこと
・ その後の「できるだけ安価な実装方式」を選択する自由度を確保すること

が困難なモデルになる可能性も高い。

本来なら、モデルを利用した開発技術を導入するにあたり、開発現場の抱える課題と新技術導入の狙いに応じて、適切な開発手法を導入するべきである。例えば、モデルを活用した開発技術を導入するには、例えば図 5.4-1 に示すような開発技術の位置付け、メリットやデメリット、動向などを整理した上で、

・ 何を目的としての開発技術の導入か
・ どのモデリングスコープで利用するのか
・ そのために、どのようなモデリングをする必要があるか、
・ どのように導入した開発技術を、日常のソフトウェア・システム開発プロセスに盛り込むか

を事前に整理し、また、必要なモデリングスキルを向上させるなどの策を取った上で、新規の開発技術導入を進める必要がある。なお、新しい開発技術の導入に対する事前評価の進め方の例には、[IPA13]がある。

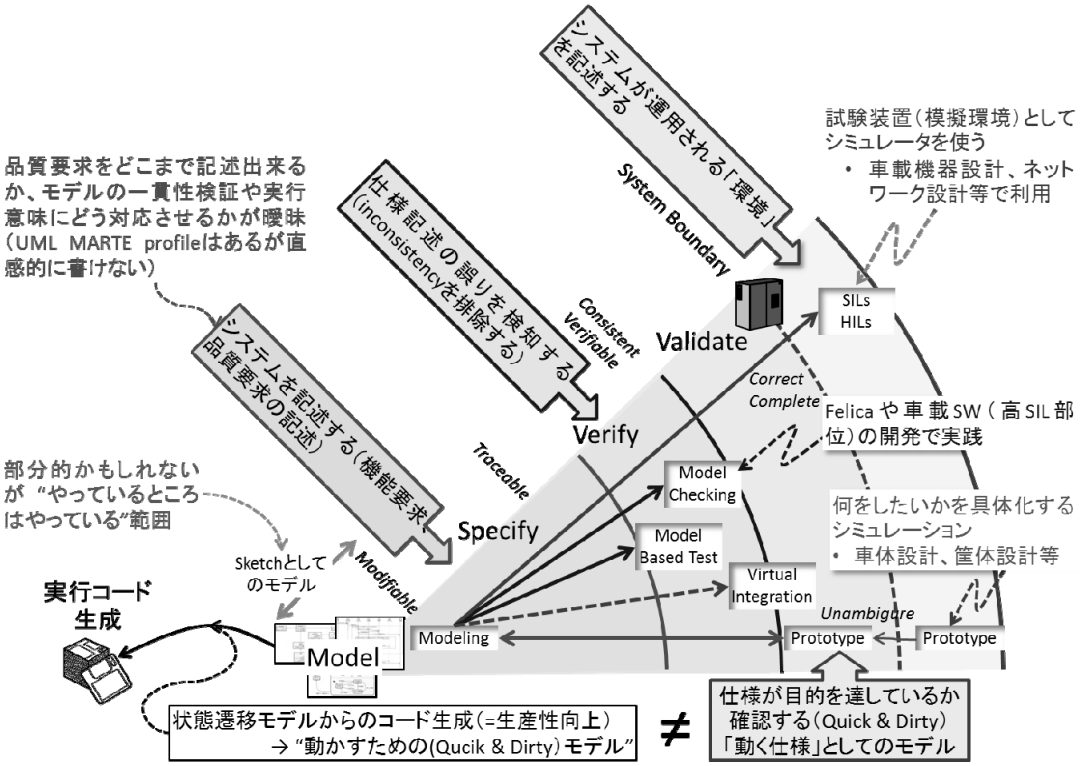


図 5.4-1　モデルを利用した開発技術と利用の目的

5.5　ソフトウェア資産再利用や共通プラットフォーム化のモデル化技術が未成熟

開発するソフトウェア・システムの規模拡大に対応するアプローチのひとつは、開発組織が抱えている既存のソフトウェア資産の活用、あるいは Android OS などの共通プラットフォームの導入による「新規開発コード量の抑制」である。実際、既に、組込みソフトウェア開発の現場でも、スクラッチからのソフトウェア開発をすることは稀で、新しい製品向けのソフトウェア開発であっても、開発チームが抱える既存のソフトウェアを流用・改修して利用するケースが増えている。実際、組込みソフトウェア開発の実態調査[IPA13b]では、開発されたソフトウェアコードのうち、新規開発コードは 33%程度にとどまっており、残り 60%前後が既存ソフトウェア資産の再利用や改良によるもの、6%程度がオープンソースソフトウェアや市販ソフトウェアの導入で占めているとの報告がある。新規開発コード比率は、一般的な情報システム系の開発よりも高い[JUA10]ものの、組込みソフトウェア開発現場でもソフトウェア資産再利用は無視することは出来ない状況と言える。

しかし、現状のモデリングツールは、このような状況に充分対応できる状況にあるとは言えない。ソフトウェア組込み制御機器の開発では、前述の通り、与えられた品質要求や制約に応える実現方式（アーキテクチャ）を採用することが重要となる。これは、これらの製品の開発で既存のソフトウェア資産を流用・改修する場合でも同様である。既存のソフトウェア資産が内包する現行アーキテクチャが、新製品に対する品質要求や開発にあたっての制約条件を充分に満たすことを確認することが求められる。しかし、既存のソフトウェア資産の流用・改修の繰り返しに頼った開発を数年続けると、開発現場には「信頼できるアーキテクチャ仕様書」が消失していることが多い。これは、流用型開発を進めるにあたり、開発ライフサイクルプロセス全般にわたる成果物の更新、アーキテクチャ設計まで遡っての設計の妥当性確認を行っておらず、既存のソースコードの改修だけを行っていることも一因である。この結果、機能の追加などの改修に伴って、本来満たすべき品質要

求が未達となるなどして、大規模な欠陥改修作業が求められることになる。さらに、この状況を繰り返すうちに、手戻りコストが増大し、一方でスクラッチからの再開発費用も捻出出来ない状況に陥って、結果的に、製品開発そのものが立ち行かなくなる事態に陥る可能性もある。このような状況にある開発現場では、モデリングツールに対する期待も高いものとなる。例えば、既存のソフトウェア資産としてソースコードを与えると、そこからアーキテクチャを復元する機能などを期待することになる。組込みソフトウェアの開発では、与えられた品質要求や制約条件に応えるアーキテクチャを利用しているかの妥当性確認が重要であり、従って再利用型のソフトウェア開発でも、より上位のソフトウェアやシステムのアーキテクチャを既存のソフトウェア資産から自動的に復元することが期待される。これに対して、現状ではコード解析ツールを利用したソースコード構造解析は行える[IPA12b]ものの、その上位のソフトウェアやシステムのアーキテクチャ復元に至る機能は技術的にも困難で、現状では、人手でのアーキテクチャ設計情報の復元に頼るほか有効な手段はない。同様の問題は、外部からの共通プラットフォームの導入でも発生する。

一方、新規開発する組込みソフトウェアやソフトウェア組込み機器の開発時に、今後の長期的なソフトウェア資産の流用・再利用に対する準備として、システムアーキテクチャやソフトウェアアーキテクチャのモデルなど、上位設計の成果物を適切に維持する施策は充分に講じられていない現実もある。今日では、オープンソースソフトウェアや市販パッケージソフトウェアとして仕様変更などのチケット管理ツールや、モデルとドキュメントをコードと同時に扱える構成管理ツールも数多く出現しており、先進的な開発組織ではこれらのツール群を組合せて上位設計のモデルから実装コード迄のソフトウェア資産を一括して管理する開発環境を構築、運用している事例もあるものの、多くの開発現場ではこうした開発環境の整備、運用に至っていないのが現状である。さらに、こうした開発環境下で [AUT10]や[CMM02]で要求されるソフトウェア資産のトレーサビリティ管理迄取り込んで実践している例は極めて少ないのが実状である。モデルを利用したソフトウェア開発の実践では、図 5.2-3 に示したように、開発フェーズ毎にその開発フェーズに期待されるモデルを作成することが期待される。これに伴って、開発フェーズ毎に作成されるモデル間のトレーサビリティ管理の重要性も高まると想定されるが、モデル作成ツールもトレーサビリティ管理を機能としてサポートするものが少ない状況にある。

結果的に、共通プラットフォームの外部からの導入や、組織が抱えている既存のソフトウェア資産の流用開発と、導入しようとするモデル技術の間に齟齬が生まれ、期待通りにモデルを使ってアーキテクチャの妥当性を確認する、という継続した取り組みが出来なくなっている可能性がある。

6. モデリング事例に見る課題の考察

広義のモデリングについては、第 4 章に記述してある通りで、過去古くから現在に至るまで、様々な分析・設計手法と、記法、ツールなどが用いられながら実施されてきている。

また、IPA/SEC でも組込みソフトウェア開発における品質向上の勧め［設計モデリング編］[IPA14]や、モデルベースエンジニアリング導入の手引き[IPA15]その利用の促進と、レベルの向上に関しての活動が見られる。

本委員会では、2014 年 10 月に開催した、恒例のワークショップのテーマとして「モデリング」を取り上げた（ワークショップの詳細報告は報告「1. 組込み系ソフトウェア・ワークショップ 2014」参照）。ワークショップに関しては、参加者の方々に参加前と参加後にアンケート調査を実施している(アンケート結果の詳細は､事前アンケートが付録 2 に、当日アンケートが付録 3 に記載されている)。

アンケート結果から､モデリングに於ける課題が浮き彫りになっていることが判明した。また、本委員会の委員及びその所属組織に於いて、実際に起きたモデリング開発時の課題や成功・失敗事例、ワークショップで発表して頂いた事例講演の内容を体系的にまとめたので紹介する。

対象とする課題は以下の通りである。

① ワークショップ事前アンケート
② 委員会で収集した課題
③ モデリングの事例（ワークショップセミナー事例講演）

以降の節で、それぞれについて、紹介する。また、6.4 節に、これらの事例を分析した結果のまとめを記載している。

6.1 ワークショップ事前アンケート

前述の通り、ワークショップの開催に際して、参加者の方々に事前アンケートの記入をお願いしている。自由記述形式アンケート項目の中で、モデリングに関するコメントを多数頂戴した。その全体像をまとめたものが図 6.1-1 である。

コメントの全体を以下のカテゴリに分類した。

a. モデリングの定義
b. モデリングに必要なスキル
c. モデリングへの期待／効果
d. 良い／悪いモデル
e. モデリングの失敗／成功要因
f. モデリングの課題
g. セミナーへの期待

以降では、上記の 7 つのカテゴリについて、そのコメントの傾向や背景の推察などを述べる。

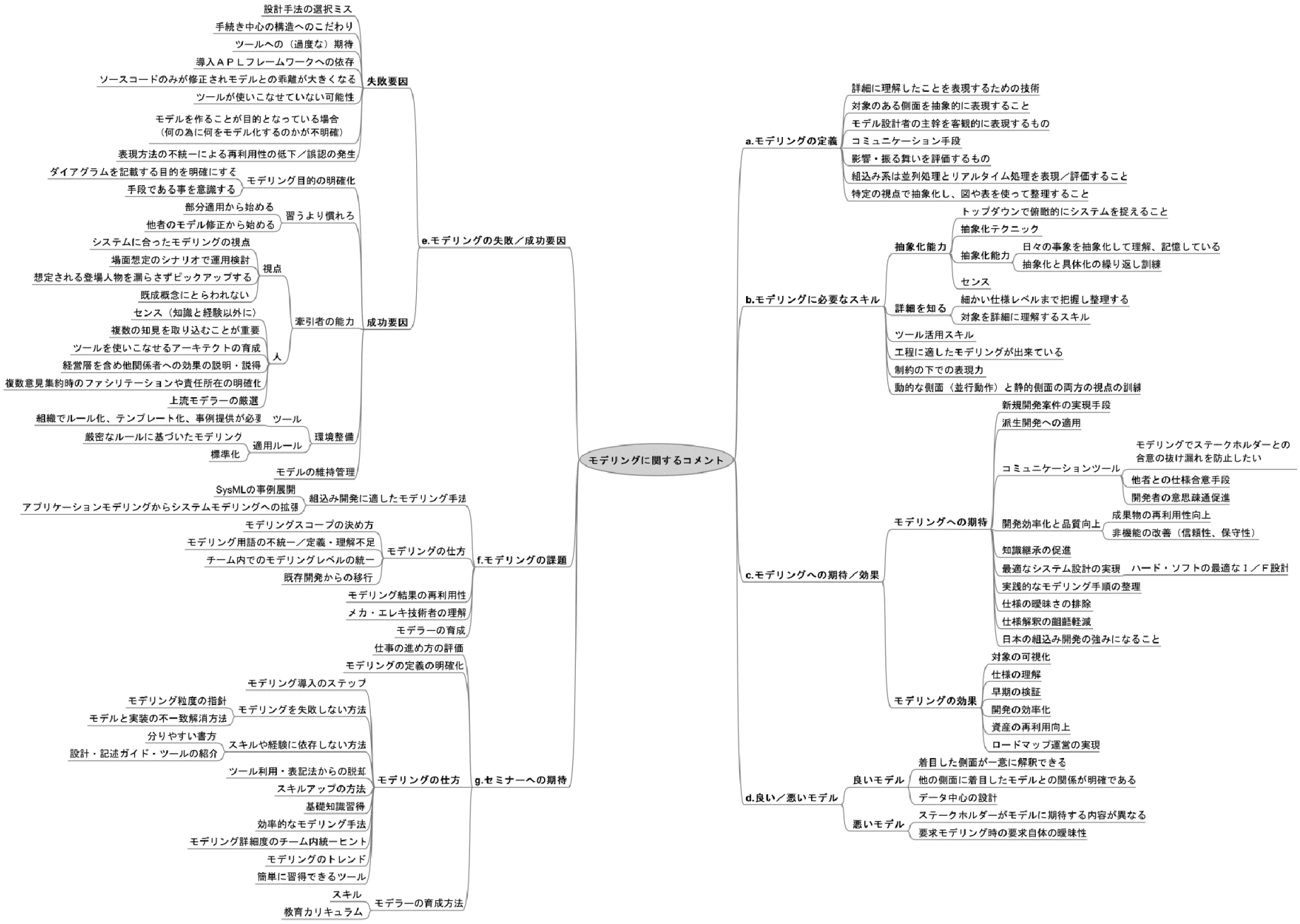


図 6.1-1　セミナー事前アンケート（モデリングに関するコメント）

6.1.1　モデリングの定義について

「a.モデリングの定義」は、受講者の考える「モデリング」について述べて頂いたものである。「モデリング→モデル化→抽象化」を実施し、複雑な事象、対象物を扱い易いものにすることであり、結果として、関係者の共通認識が容易になり、問題の混入、仕様や要件などの抜け・漏れを防ぐことが可能となる。その意味で、モデリングの目的が、ここに列挙されたもの、或いは、ここに列挙されたことが実現し、それによりもたらされた二次、三次効果として得られるものと考えられている。

そもそも、現在の製品開発に於いて、システム化する規模が大きくなり、また、社会インフラシステムなどでは、継続的に改良が加えられ、差分開発が常態化しているのが、組込みシステムの現状であるため、対象となるシステムを理解しやすく分析することが求められている。そのため、コミュニケーションの手段や共通認識を深める手段、複雑なシステムの理解を助け深める手段として期待する状況がわかる。

6.1.2　モデリングに必要なスキルについて

モデリングを導入しても期待する効果を得ることが出来ない場合がある。その事例を分析して、対策をすることで、同様の失敗を防ぐことが可能である。また、成功事例の理由を分析することでも、期待する効果を得るための参考となる。

「b.モデリングに必要なスキル」は、その一要因として、モデラーとして求められるスキルを、実体験から述べて頂いているものである。

その必要なスキルとしては、抽象化能力という回答が多かった。モデリングとはそもそも、現実の世界の事象をコンピュータシステムで制御、管理するため、複雑な仕組みを抽象化する事で、扱いやすく、或いは扱える範囲を設計することである。そのため、抽象化能力は必須であると言える。とはいえ、毎回新しい考え方、仕組みなどで捉えるのではなく、既知のメカニズムなどへの等価できることが多く、それを如何に効率的に、且つそのメカニズムで解決可能かどうかを判断できるかということになるのではないだろうか。

6.1.3　モデリングの効果について

「モデリングの効果」と「モデリングへの期待」はほぼ同一の目的となっている。「モデリングへの期待」に関する“思い”は、それぞれ「モデリングにより期待する効果」である。

モデリングされた対象の可視化が適切であれば、コミュニケーションロス・ミスなどは減り、仕様の誤りなども早期に発見ができる。共通認識のモデルで検討できれば、再利用性も上がり、生産性の向上が期待できる。期待の視点で特徴的なことは、日本の強みとして生かしたいという思いがあること。本委員会でも、日本の強みになる技術施策を提言してきており、思いは同じである。今後日本の強みにできるモデリングとは何かを具体的に検討していく必要がある。

6.1.4　良い／悪いモデルについて

d.の「良いモデル」、「悪いモデル」とは、それぞれ、「期待する効果」が「得られた」、「得られなかった」事を示している。但し、アンケートの内容が、直接的にモデル評価事例を収集していないため、具体的なものは余りない。プロジェクトの目指すゴールに対して、適しているか否かが評価された結果である。正確に判断するためには、その基準を明確にし、それを満たしているか否かで決定することになる。

6.1.5　モデリングの失敗／成功要因について

e.は、「モデリングの失敗／成功要因」で、セミナーのタイトル「なぜモデリングは失敗するのか？」とも関係するため、多くの記述があった。

成功要因については、牽引者やモデラーの能力に関する意見が大半を占める結果となった。

一方で、失敗要因については、ツールや手法の使い方、開発環境の問題指摘が多く見られる。アンケート自体が、数量分析を目的としていないため、意見のバリエーションで分析するのは適切ではないが、成功の要因の方が様々な要因が挙がった。単に推測となってしまうが、失敗の要因自体が十分に分析された結果を示していない可能性も考えられる。成功要因も同様の考え方ができるが、「牽引者の能力」に分類されたものが多いのは、成功の陰には頼れる人が居て、その効果が大きく感じ取れたのだと推測する。

6.1.6　モデリングの課題について

f.の「モデリングの課題」では、組込みシステムとしての適用の仕方に、多くの課題が挙がった。その中でも、アプリケーションモデリングからシステムモデリングへの拡張・対応が課題となっている。モデリングの失敗・成功要因にもあるが、モデリング手法の適用環境の整備が求められている。当然ながら、これらの手法を上手く適用して、開発を推進できるモデラーの育成も必要な課題となっている。

6.1.7　モデリングへの期待について

g.の「セミナーへの期待」では、モデリングを成功させるためのツールや技法、技術習得方法などへの要望が多かった。モデリングに限らず、ソフトウェア開発では、開発者の力量が多くの成功要因となる。そのスキル向上には、様々な教育・セミナーなどが提供されてはいるものの、誰にでも習得できるものとは限らない。プロセスやツール、ルールなどを整備することで、ある程度の底上げは期待できる。そのための、手段を期待しているものと推察する。これまで、本事業委員会で取り上げてきたテーマでは、最後は「育成」がカギとなることが多かった。本テーマでも、成功のためのカギになることは容易に想像される。また、企業活動としては、必ず取り組まなければならないテーマでもある。

6.2　モデリングの失敗事例

モデリング失敗例に関して、本委員会各員及び委員の所属する組織の有識者から収集し、まとめたものを図 6.2-1 に示す。

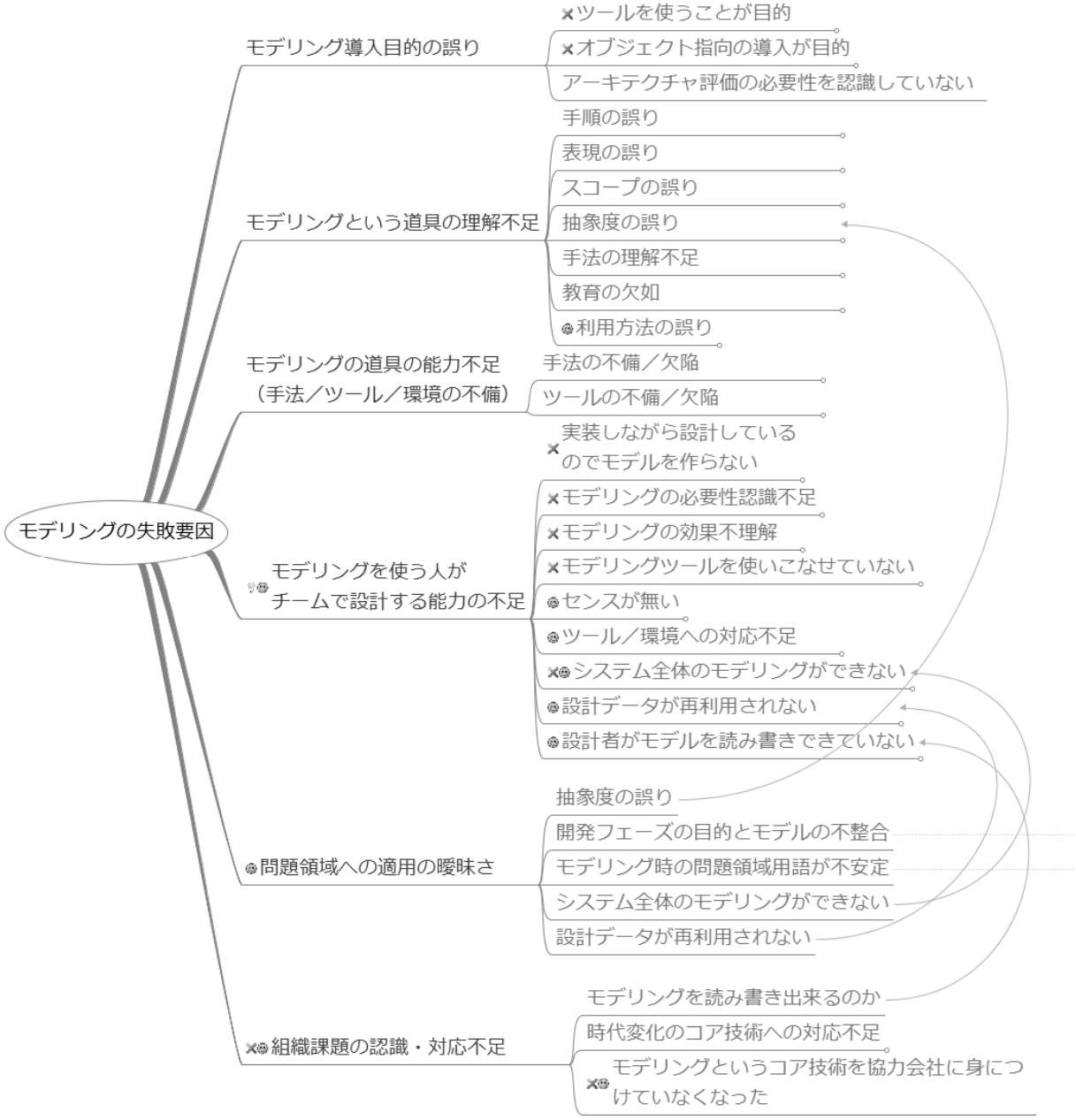


図 6.2-1　モデリングの失敗要因収集結果

この中で、「モデリングの失敗」とは、「正しいモデル化が出来なかった」というのが、失敗の定義になっている。

そこで、正しいモデル化とは何か？であるが、本来あるべき「抽象化した成果物が得られていること」であり、それが得られない原因、あるいは得られなかったことで生じた事象などが記載されている。大別すると、以下の通りとなった。

① モデリング導入目的の誤り
② モデリングという道具（ツール・技法）の理解不足
③ モデリングの道具（手法・ツール・環境）の能力不足

④ モデラーの能力不足

4 章に述べたようにモデリングのスコープは様々あり、それぞれのスコープでのモデリング技法を駆使する必要がある。その観点で、現在世の中にあるツールや技法などが必ずしもサポートしきれておらず、その道具を使える能力が不十分なまま適用したために、得られるべき結果が得られなかったということになる。

6.2.1 モデリング導入目的の誤りについて

「モデリング導入目的の誤り」とは、モデリングツールを使うことが目的となってしまい、形式的にツールを使わせているだけになって、設計ルールに縛られ、既存モデルを表面的に真似するだけとなっている。また、コード自動生成のためにモデリングツールを使うといったことが考えられる。

さらに、オブジェクト指向を導入することが目的となってしまい、手段と目的の区別がつかなくなってしまっているということもある。

また、アーキテクチャ評価の必要性を認識せずに、おざなりにモデリングを行ってしまい、モデリングが失敗するということもある。

何のためにモデリングを行うのかという、モデリングの目的を明確にして、その目的を達成させるためにモデリングを行う必要がある。モデリングのゴールは何か、何ができたらモデリングが完了したといえるのかを明確にしてく必要があると思われる。

6.2.2 モデリングという道具（ツール・技法）の理解不足について

「モデリングという道具（ツール・技法）の理解不足」とは、モデリングというものを正しく理解していなかったことにより、モデリングが失敗したというものであり、いくつかの要因に分類できる。

- そもそも、どうやってモデル化していくのかというやり方やなど、モデリングの手順が間違っている場合や、モデリングのアプトプットが、次工程のインプットになっておらず、各プロセスのゴールが繋がっていないなど、手順の誤りによるもの。
- 設計意図が表現できていない、複雑すぎるモデルや名前付けが悪く、他人が理解できないモデル化をしてしまうという、表現の誤りによるもの。
- 開発対象のドメインや開発プロセスに対応していないモデル化や解決できない問題をモデル化しようとするなど、スコープの誤りによるもの。
- スコープの誤りとも関連するが、適切な抽象度になっていないケースや、モデラー毎に記述レベルが異なるケースや実装の意識が拭えないといった、抽象度の誤りによるもの。
- モデリングの検証をしない、（別な観点での？）再モデリングを実施しない、静的モデリングだけで動的なモデリングを行わないなど、モデリングの目的を達成させるためには必要なことが分かっていない・できていないという、手法の理解不足によるもの。
- そして、モデリングという道具（ツール・技法）について、教える人材・教材がない、適用するための標準プロセスがないなど、教育の欠如によるもの。

これらの理解不足というのは、やはり導入時の教育の欠如によるところが大きいと考えられる。教育の中では、技法、ツールの使い方はもちろんのこと、モデリングの目的についてもしっかりと教えることが必要と考える。但し、教える人材・教材がないといったジレンマで苦しんでいるのではないかと思われる。

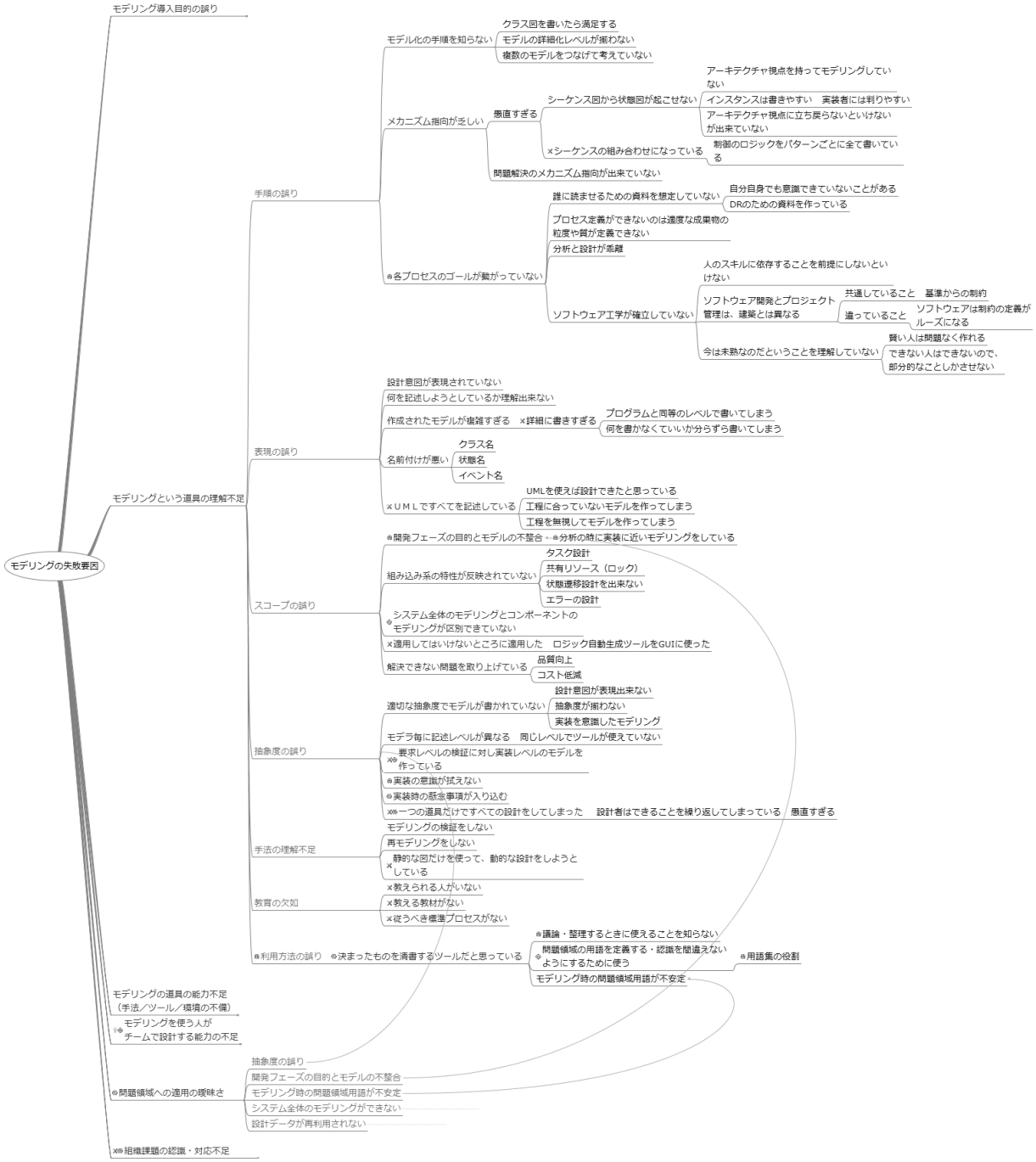


図 6.2-2　モデリングの失敗要因（ツールの理解不足）

6.2.3　モデリングの道具（手法・ツール・環境）の能力不足について

- 「モデリングの道具（手法・ツール・環境）の能力不足」とは、品質要求（非機能要求）が考慮しきれないことや、離散系モデリングと連続系モデリングが整理されていないこと、また、UML の表現能力では足りない部分があるなど、手法の不備や欠陥に起因するもの。
- 自動生成機能の不具合やツール間のデータ互換性がなく、過去の資産が流用できない、流用・派生開発への対応が不十分、仕様検証ができないなど、モデリングの効果を最大限発揮させようとした場合に、ツールの不備や欠陥によるもの。

これらに対しては、開発対象のドメインに応じた手法やツールを適切に選択し、手法・ツールの適用効果を最大限に引き出すために必要なこと、やるべきことを明確にして適用することが必要であると思われる。

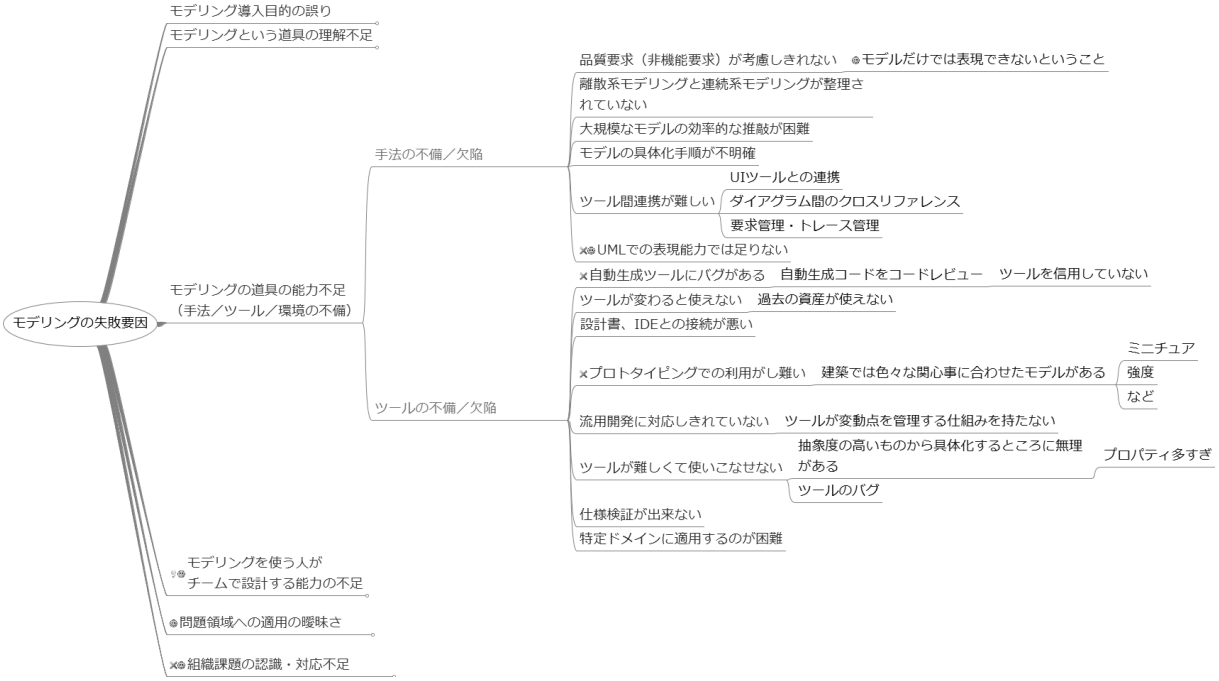


図 6.2-3　モデリングの失敗要因（道具の能力不足）

6.2.4　モデラーの能力不足について

「モデラーの能力不足」とは、モデリング行う個人やチームとして設計する能力が不足しているために起こる問題である。個人としては、実装しながら設計をしているためモデルを作らない、モデリングの必要性認識不足や効果不理解およびツールを使いこなせないなど、個人のスキル不足によるもの。また、モデリングはある程度センスが要求されるものなので、センスが不足していることによるものもある。

チームとしては、道具（手法・ツール・環境）を使った開発プロセスを定義できないといった、チームでの対応力が不足している場合、システム全体のモデリングや、レビューなどにより他人のモデルを読んで理解する人材に欠けているという問題もある。

これらは、6.2.2 の道具（手法・ツール・環境）の理解不足とも関連するが、個人だけでなくチームとしても道具（手法・ツール・環境）について、教育する必要があるということが考えられる。

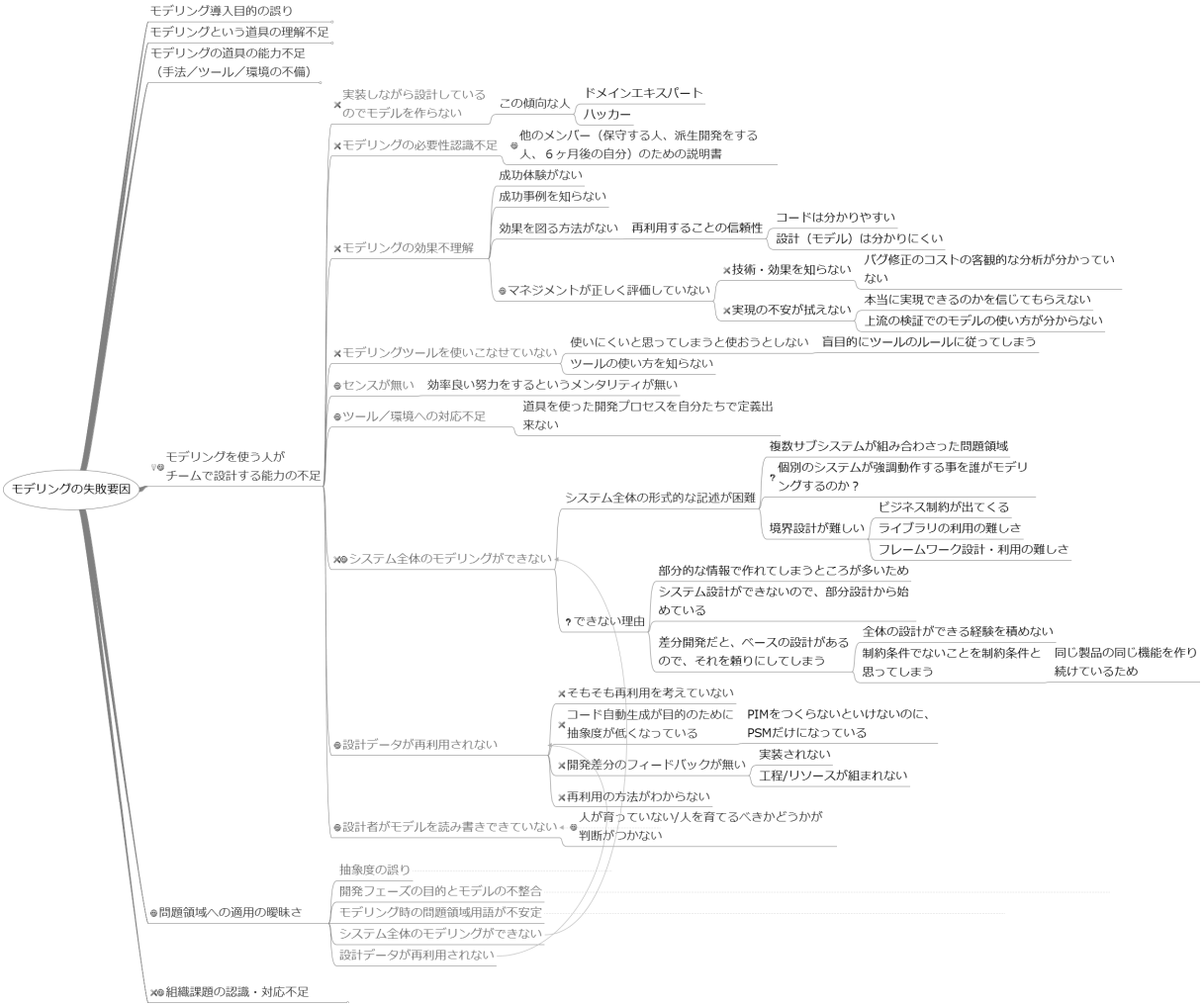


図 6.2-4　モデリングの失敗要因（モデラーの能力不足）

6.3　モデリングの事例

本節では、2014 年度ワークショップでご講演頂いた事例について、その要約を示す。モデリングの課題や対策について、その背景を推察するのに参考にして頂きたい。

6.3.1　事例 1）Excel と Enterprise Architect を駆使したモデル設計現場適用

（2014 年度ワークショップ 富士通株式会社　奥村 一幸氏、株式会社富士通コンピュータテクノロジーズ　石田 晴幸氏　事例講演）

[背景・目的]

モデリング手法の開発現場適用に対し、以下の三大要因により踏み切れずにいた。

- 従来の開発プロセス変更に対する懸念
- 予算上の問題（ツール購入／メンバ教育費）
- モデル言語を使用するメリットが把握できない

一方、大規模なソフトウェア開発の現場では、以下の要因により品質確保が困難な状況が続いていた。

- 大勢の開発担当者により機能毎に担当を割り当てた並行開発を進めるため、担当個々に異なる処理フローを作成し、機能間を含めた全体像の把握が困難となる。
- 機能間のタイミング問題に対する適切な試験ケースの抽出が行えず、競合試験での抜け・漏れが多発する。

これらの問題を解決するためにモデリング手法の適用に踏み切った。

[適用手法]

UML モデリングツール Enterprise Architect（EA）

[解決したい具体的な課題]

テーブルアクセスにおける機能間の競合洗い出し。

[課題解決への施策]

Excel を使って開発担当者が自由記述で作成していた ER 図や処理フローを、記述書式を定義して型決めし、ツールにより EA に取り込んで EA モデル化。

EA モデル化したものをツールにより機能・テーブルマトリックスとして出力し、競合試験項目抽出に使用。

[効果]

ツールを利用した手順の確立にて、開発工数の肥大化を防ぎ、テーブルアクセス競合問題の洗い出しを実現。

[考察]

既存のソフトウェア開発プロジェクトに対するモデリング手法の組み込みについては、少なからず新たな手番の追加が発生するため、開発工数の増加は避けられないが、ツール化による増加工数の抑止を実現し、その上で課題となっていた競合問題の洗い出しを実現した。

本モデリング適用事例成功のポイントは以下 2 つと考える。

① 適用に伴う工数増加をツール多様により回避

② Enterprise Architect は、API・アドインによる拡張が容易なため、独自生産物⇔EA モデル化の変換が可能で、適用プロジェクトに合った生産物を出力

図 6.3-1 に本適用事例（事例講演）のまとめを示す。

図 6.3-1　事例１まとめ

6.3.2　事例 2）MFP の複合機のコントローラと板金機械制御におけるモデル化開発

（2014 年度ワークショップ 株式会社アマダ　久我雅人　氏事例講演）

講演者の経験時系列に沿って主に 2 つの製品領域におけるモデリング事例が紹介された。

１．事例 2-1：MFP 制御ソフトのオブジェクト指向によるモデル化

[背景・目的]

下記の課題に対応するためにモデル化を行った。

・従来型の全体シーケンス制御方式では模模の増大に伴って構造が複雑になる。

・従来のソフトウェア構造では小規模な変更にも全体に影響が及ぶ。

[適用手法]

オブジェクト指向設計によるモデル化

[解決したい具体的な課題]

・ソースコードの再利用ではむしろ悪くなる品質を救いたい

・仕様変更による二次障害仕様変更の工数を低減したい

[課題解決への施策]

・アーキテクチャの変更

シーケンス制御を中心とする構成から、具体的な処理単位である「紙搬送」と「画像形成」を下位層として処理を委譲するような形で全体のアーキテクチャを改めた。

・反復開発（イテレーション開発プロセス）の導入

・ソースコード中心の開発からモデル中心の開発へ

[効果]

小規模な変更は局所的な変更で対応できるようになった。また、ソフトウェア全体の再利用性も向上した。

[考察]

組み込み製品開発におけるモデル化とは、単純に一般化することでも、対象を抽象化な表現に改めることでもない。制御対象の動作についてドメインの特徴を素直にモデ

リングするのが良い。単純な一般化を行ってしまうと、どのドメインでも条件、制御、センサ、負荷などからなる同じようなモデルになってしまう。また抽象的な表現でまとめるよりは、紙搬送であれば紙の動作といったドメインの特徴に着目することで、制御対象に相応しいモデルを作成できる。

図 6.3-2 に本適用事例（事例講演）のまとめを示す。

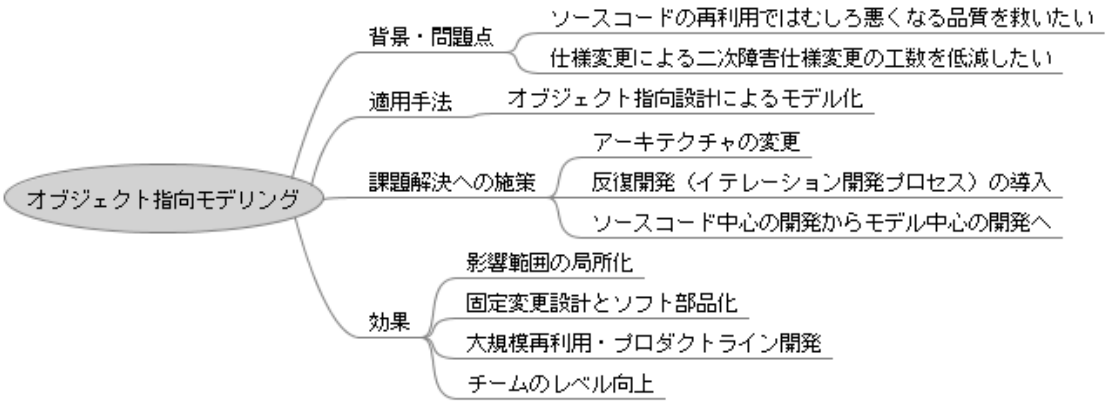


図 6.3-2　事例２－１まとめ

1．事例 2-2：板金曲げコントローラソフトのモデル化

[背景・目的]

下記の課題に対応するためにモデル化を行った。

・プログラムの見通しが悪く、開発・テストに時間がかかるようになっていた。

・一箇所の変更が多くの機能に影響を与えるようになっていた。

[適用手法]

・制御対象の特徴を抽象化して記述する

[解決したい具体的な課題]

・納期遅れや低品質が常態化していた。

[課題解決への施策]

・対象となる板金が素の板の状態から何回曲げられて製品に仕上がるかの様子をモデルで表現する。

・朝会とデイリーレビューを徹底する。

[効果]

静的指標によるソフトウェア品質が向上した。

日程の見積もりが正確にできるようになった。

メンバがソフト全体の構造を理解することで、設計についての意見交換が可能になった。

[考察]

ソフトウェアの構造が目に見えるようになることである。これによりアーキテクチャを崩さずに機能追加が可能となり、品質も安定する。さらに、ソフト構造モデルをメンバが共有し、変更点と影響範囲を全員が特定可能となったことで、適切なテスト計画が立てられるようになり、その結果として品質の向上につながるという良い方向の

スパイラルが形成されるようになった。

図 6.3-3 に本適用事例（事例講演）のまとめを示す。

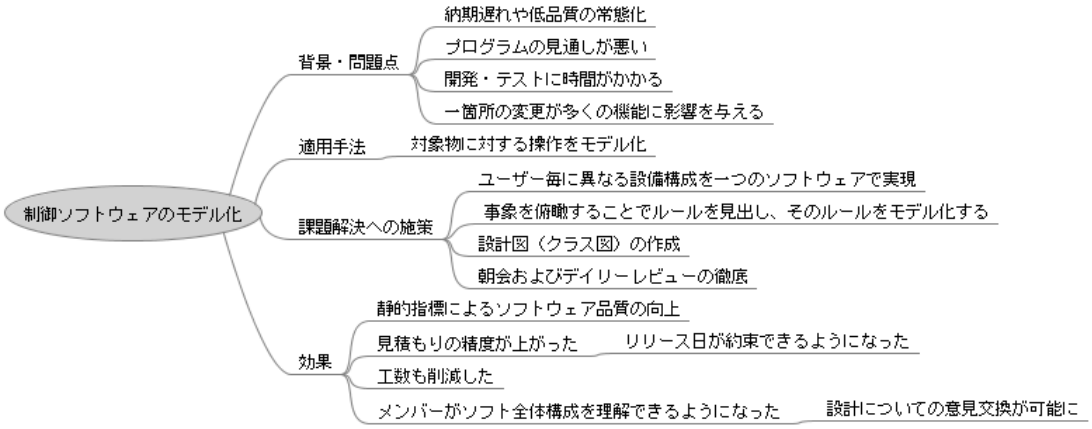


図 6.3-3　事例２－２まとめ

6.4　モデリング課題の考察

「6.1 ワークショップ事前アンケート」、「6.2 委員会で収集した課題」「6.3 モデリングの事例」により、モデリングへの期待や実態の一端が見えてきた。「5.組込みソフトウェア開発現場でのモデリング適用上の課題」で述べているように、本委員会では、以下の課題があるものと想定しており、本章内で紹介している事例との関係を分析した。

① モデリング技術導入目的の曖昧さ
② 開発フェーズの目的と異なるモデルの作成
③ 仕様の検証／妥当性確認にモデルを活用しにくい
④ 新規開発技術導入目的の不明確さ
⑤ SW 資産再利用とモデルの扱い（基準・標準）

その結果、組込みシステム産業にとっても、モデリング技術の活用は必須要件となりつつあり、如何に上手く適用できるかが重要事項となってきていることが分った。そのために、組込みシステム特有な範囲との切り分けと、エンタープライズ系技術の範囲での効果的なモデリング設計の推進と、組込み部分におけるモデリング適用人材の育成が必要であり、そのための先行投資できるかが、次の時代へ生き残るポイントになると思われる。

6.4.1　モデリング技術導入目的の曖昧さ

導入目的が曖昧な場合、期待する効果が得られないばかりか、弊害を生じてしまう場合が多く見られる。コミュニケーションの促進を目的とするには、対策自体の実施コストが大きくなってしまっている場合や、本来のコミュニケーションロスがモデリングでは改善できるスコープではない場合もある。モデリングが有効な解決策である、という根拠に至った正当な理由が必要である。流行りであるからや成功事例があるからというだけでチャレンジすると、手法を適用することに主眼が置かれてしまう危険性がある。

アンケート結果などからは、ツールや環境の整備に関する要望が多く見受けられるが、手法をうまく使うことだけに目的がある場合には、注意が必要である。またツールや環境、整ったプロセスがあれば、モデリングの効果が出せると考えてしまうと、失敗する可能性が高い。

5 章の導入部にも記したように、組込み分野においても、急速にエンタープライズ系の技術導入が進んできている。この様なソフトウェア・システムでは従来の規模視点で IT システムなどに見られる設計技術を導入してもうまくいかない。組込みシステムには、H/W システムも含めたプラットフォームやフレームワークを設計しなければいけないからだ。厄介なのは、概ね強い制約（メモリ使用量や動作時間、低スペック CPU など）が存在するため、他システムとの共通化などが困難なケースが多いことである。過去に組込み分野での H/W、S/W の共通プラットフォームの推進が行われたケースもあるが、現在はあまり聞かれなくなってしまった。システムを俯瞰して、組込みとの切り分けを行い、適材適所のモデリングを適用する複合モデリングとでも言える技術が必要になってきているのではないだろうか。

6.4.2　開発フェーズの目的と異なるモデルの作成

事例講演の内容は、成功事例である。モデリングを成功するには、上記 5 つの失敗要因を持たないことが望ましいが、実際には完全な計画・環境で実施できるのは希である。事例講演に於ける成功要因を、論理的な因果関係に拘らず、ソフトスキル面も含めた印象面などから以下の 3 つのポイントが導き出された。

① 複雑化したシステムの効果的代替設計の発想力
現状の課題を的確に捉え、効果的な解決メカニズム設計が発想できる。
② 開発者のスキル面のアンバランスをカバーするプロセス
開発者が必ずしも均一でかつ高いスキルを持っているとは限らず、必要最低限のレベルへ引き上げるためのツール、プロセスなどを整備する事ができる。
③ 新方式への抵抗感を払拭する牽引者の熱意と努力（労働力）
それぞれの置かれた立場での課題は様々であり、教科書通りであったり、事例通りであったりすることはほぼない。その中で、新しい設計や開発手法を導入するには、周囲の説得や、時には走り始めて納得させるなどの、努力が必要で、これを実践する熱意がある。

成功事例をケースバイケースと捉えず、一般プロセス化するには、幾つか課題も見えてきた。
今回の範囲を、一般ケースに拡張（汎化）して考えると、以下のことに気付く。

① モデリング適用効果（コスト、開発者負担）の示し方（現状のロスの定量化とモデリング導入効果後の差分表現）
② モデリング導入を成功に導く能力の高い牽引者の養成
③ 新しい開発方法に対するスポンサー（承認者など）の理解

組込みシステムにおいても、アジャイルなどの開発手法の適用事例が聞かれ始めているものの、決して多くはない。その様な中で、企業方針として取組み始めている事例もある。しかし、トップダウンで実施されるが故の問題も聞こえてきている。有識者とのブレーンストーミングでも、エンタープライズアジャイルなども有効な手段であるとのコメントを頂いており、次年度以降の検討課題として取り上げていきたい。

6.4.3 仕様の検証／妥当性確認にモデルを活用しにくい

アンケート結果から、モデリングの失敗／成功要因として関連が見られるものの、他の分類や、事例講演などでは明確な課題提議にはなっていない。
アンケート結果では、ツールの整備やツールの使い方、プロセスの整備などの要望として挙がっている背景として、本来目的としているモデリング手法やツールが適切ではなかったために、期待通りの評価結果が得られなかったということが考えられる。現時点では、検証／確認に適したモデリングが整備されているとは言い難い面も有り、環境整備と運用標準化などが整ってくると、問題が解決するものと期待される。
5 章でも触れている通り、外部委託（特に海外）が増えており、その活用目的も上流設計を任せることも増えてくると考えられる。そのときに、検証手法の明確な基準・標準を持つことは、単に品質を評価するに留まらず、その後のプロダクトライン開発や保守コストの面で大きな意味を持つことになる。

6.4.4 新規開発技術導入目的の不明確さ

当初の想定課題として難解であり、この視点での課題分類は難しい。6.4.3 と同様に、モデリングの失敗／成功要因として関連が連想されるが、直接的な課題としては分類が困難である。開発フェーズと検証手段・対象が適切に選択されることで、問題を回避できるものであるが、「先ずはモデリング」という段階では、この種の問題に直面することになる。前述の通り、6.4.3 と同様に環境整備と運用標準化などが整ってくると、問題が解決するものと期待される。

6.4.5　SW 資産再利用とモデルの扱い（基準・標準）

アンケート結果でも、ソフトウェア資産の再利用性への問題を述べているものがある。開発フェーズ間での可逆性の完全を求めてしまったり、設計評価のタイミングを間違えてしまったりすると、下位設計・実装時の変更を反映しきれなくなり、派生開発時にモデルの劣化を発生させる原因にもなる。

組込みシステムにも押し寄せてきている標準資産（COTS など）の活用でコストダウンを狙うと、H/W 制約などから新規に作る程の改良が必要になってしまったり、折角モデル化したアプリケーションが動かせないプラットフォームに変化してしまったりすることも起きうる。徐々に変化するプロダクトライン設計が可能な場合には、あまり問題にはならないかも知れない。

7. モデリングの適用に対する課題解決の考察

開発現場において、モデリングの適用にはしばしば課題がある。本章では、課題解決のヒントとなりそうな考え方および課題解決に向けて、本年度のワークショップ及び当委員会におけるブレーンストーミングの結果からみられた、解決の糸口となりそうな考察を紹介する。

7.1　ワークショップ参加者アンケート

エンタープライズの分野ではアーキテクチャ構築やシステム設計などで多様にモデルを活用することで、作業の効率化が行われている。一方、組込みソフトウェア分野では、現状モデリングを活用したソフトウェア開発技術が大きく普及していない。近年、組込みソフトウェア分野では、従来の小規模、職人芸的な気質での開発だけでなく、大規模開発が必要とされるケースも増加している。これは、様々な機器に組込みソフトウェアが搭載されるようになり、かつ機器間が連携することでシステムとして要求される開発規模、複雑度も年々増大しているのが原因である。これら増大した開発規模、複雑度を扱うためには、モデリングによる抽象化が必要不可欠である。

一方、生産効率向上のためにエンタープライズ分野で培われたモデリング技法を、組込みソフトウェア分野に適応しようという試みは様々な試行錯誤の中で行われてきた。しかしながら、組込みソフトウェア開発特有の部分が原因であるためか、開発現場に定着するのは難しく、次第に利用されなくなるという現実がある。

この問題を明らかにするため「様々な適応例があるエンタープライズ系のモデリング技法と、組込みソフトウェアで適応すべきモデリング技法の根本的な違いは何か?」という議論が本専門委員会で行われた。本専門委員会では、この疑問を解決すべく、組込みソフトウェア開発者に、現状モデリングを成功させる要因を確認するためにアンケートを実施した。アンケートは、本専門委員会が開催するワークショップに参加された組込みソフトウェア分野の開発者に対して行った。

アンケート方法は、ワークショップ開催後参加者の方々に「組込み系ソフトウェア開発において、モデリングをどのようにすればいいとお考えでしょうか。」という質問を投げかける形で行った。開発現場の生の声をできるだけ正確に把握する必要があるため、こちらで定型的な回答は用意せず自由形式での記入をお願いした。参加者が考える組込みソフトウェア開発でモデリングを成功させる要因として、コメント数が多い物から分類したのが以下の項目である。

a. モデリングのルールを決めモデルの共有を行う	11 名
b. 対象を適切に抽象化したモデリングを行う	8 名
c. モデリングへの理解を教育・実践を通して行う	8 名
d. モデリングの意義を明確化しモデリングを行う	7 名
e. モデルの検証を行う	6 名
f. 修正/再利用性を高めるためモデルとコードを対応づける	5 名
g. モデルに非機能要件/制約事項を盛り込む	5 名
h. モデルに要求を盛り込む	5 名
i. 静的構造を可視化する	4 名
j. 動的構造を可視化する	2 名

本アンケート結果を分析すると、各項目は、「チーム・組織力向上」、「設計力向上」、「非機能要件・制約条件のモデル化」、「再利用性の向上」という 4 つのカテゴリに大きく分け

ることができる。カテゴリ分類を行ったものを図 7-1.1 に示す。

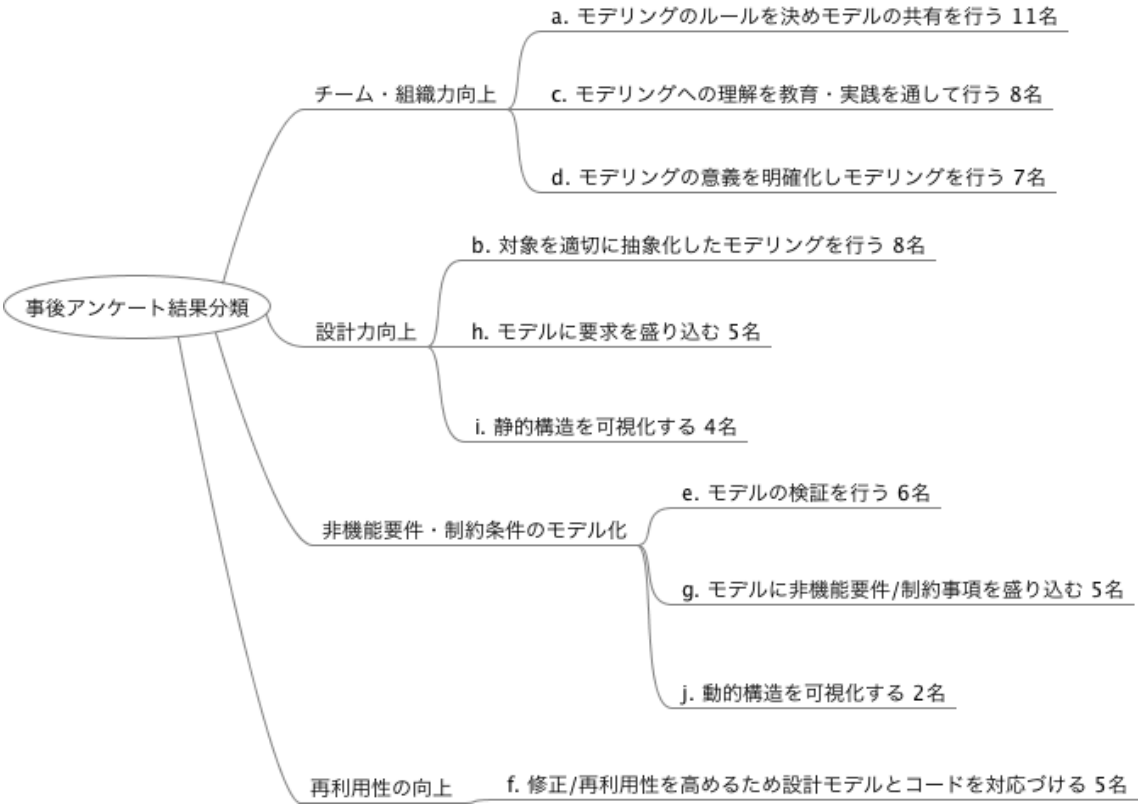


図 7-1.1　事後アンケート結果分類

7.1.1　チーム・組織力向上について

「チーム・組織力向上」は、モデリングを実開発に活用するためには、組織的な取り組みが必要であることについて述べていただいたものである。モデリングは、少ないコストでアーキテクチャやシステム設計などをプロジェクトに関わるメンバ同士で、相互に正しく理解できるようにするために行うものであり、モデルはチームで設計を行うためのメンバ内の共通の言語である。したがって、モデリングを実開発に活用する組織は、チームメンバ間で抽象化の粒度のずれや、モデルとして共有すべきポイントのずれが起きないようにモデルを作成する時のルール／意義を明確化する必要がある。また、顧客や開発者以外のステークホルダに説明し、ドメインに対しての理解を深めながら整理していく必要があるので、チームメンバ全員が、OJT や教育を通じてモデリングへの理解を深めておく必要がある。

7.1.2　設計力向上について

「設計力向上」は、良いモデルを作るためには、開発メンバの設計力を向上させる必要があることを述べていただいたものである。システム開発の対象になるドメインは一般的に複雑であり、人間が一度に対処できる限界を越えている。開発対象のドメインを適切に把握するために、人間は不要な側面を切り捨てて必要な部分にフォーカスできるように抽象化を行う。しかし、モデル化により必要以上に抽象化し細部をそぎ落としてしまうと、本来システムで表現すべき項目まで失うことになる。適度な粒度で抽象化するためにはモデリングの豊富な知識・経験が必要になってくるため、チームメンバの設計力を向上させる必要がある。

7.1.3　非機能要件・制約条件のモデル化について

「非機能要件・制約条件のモデル化」は、組込みソフトウェアにモデリング技術を導入する際は、特に非機能要件や、制約条件など動的構造を重視する必要があることを述べて頂いたものである。組込みソフトウェアは、組込み機器上で動作するソフトウェアなので、ハードウェアと協調してシステムとして動作しなければならない。したがって、エンタープライズ向けのソフトウェアに比べて、タイミング制約、割り込み処理など、実時間に対しての制約や、CPU、メモリ資源などリソース面の制約を持っているものが多い。また、組込みソフトウェアが実装されるシステムはハードウェア上に実装されるので、製造されてからの変更が難しい。システムが完成してから性能上の問題などが発生すれば、製造したハードウェアも含めた見直しが必要になり、手戻りが大きくなる。したがって、非機能要件である性能要件や制約要件をあらかじめモデルに組込み設計をする必要がある。手戻りを減らすため制約条件が満たされていることを製造工程より前に検出できた方がよいので、作成したモデルによって非機能要件を検証できることが望ましい。

7.1.4　再利用性の向上について

「再利用性の向上」は、生産性の向上には再利用が効果的であることを述べていただいたものである。従来の組込みソフトウェア開発では、小規模、職人芸的な開発の名残で、類似するシステム開発を行う際には、以前の開発したソフトウェアをコピーし必要な部分を追加・修正されることが往々にしてある。このような従来型の組込みソフトウェア開発では、ソースコードはコピーアンドペーストにより流用されているが、設計より上流の工程が再利用されていないことが少なくない。設計が再利用されていないソフトウェアは機能拡張のための修正箇所特定のコード解析が必要で、修正の影響範囲を考慮せず場当たり的な修正になる。このようなプロセスで製造されたソフトウェアはテストを通じて品質をフォローしなければならず、テスト漏れによる品質懸念、テスト過剰によるコスト増大などの問題が生じる。低コスト・高品質のソフトウェアを組込み開発で行うためには、従来のボトムアップ型のコード中心の開発から脱却し、アーキテクトの育成、モデリングベースの設計開発を行うことで、トップダウン型の戦略的なアーキテクチャ主導開発に発展させる必要がある。

7.1.5　ワークショップ事後アンケートまとめ

ワークショップ開催後参加者の方々に、組込みソフトウェアにおけるモデリングの実施状況についてアンケートを行い、そのアンケート結果を元にモデリングの成功要因について分析を行った。「チーム・組織力の向上」、「設計力の向上」、「非機能要件・制約条件のモデル化」、「再利用性の向上」が組込みソフトウェアのモデリングにおける成功要因として重要であるとアンケート結果から参加者は考えていることがわかった。

「チーム・組織力の向上」、「設計力の向上」については、組込みソフトウェア開発特有のものではなく、モデリングを行う際には基本的に行うべき手段として一般的にあげられることが多い。これらの項目が必要であると多くの参加者が考えていることから、本専門委員会では組込みソフトウェア開発分野でモデリングが十分浸透していないと考える。

また、前記 2 項目に比べると少数にはなるが、「非機能要件・制約条件のモデル化」「再利用性の向上」など、組込みソフトウェア開発でモデリングを行う際に特徴的な項目がアンケート結果として得ることができた。特に「非機能要件・制約条件のモデル化」については、本専門委員会では組込みソフトウェア分野の特徴であると考える。組込みソフトウェア分野は、ハードウェアも含めた開発になるため非機能要件・制約条件を加味しなければ、手戻りのコストが大きいハードウェアの修正が起こる。このような手戻りを防ぐためには、設計段階からの非機能要件・制約条件を十分に検討する必要がある。現場でこのよ

うな問題が多く発生しており、モデリング時に解決することを望まれていると考える。

「再利用性の向上」については、従来型の組込みソフトウェア開発では、コード中心の開発になることが多く、これらの開発にモデリングベースの設計開発を導入する難しさがわかった。

アンケート結果から、組込みソフトウェア開発で、モデリングに失敗しないためには、「非機能要件・制約条件のモデル化」と、モデリングベースの設計開発の導入方法について検討する必要があると考えた。本専門委員会では、次年度以降これらの項目について更に深く掘り下げる。

7.2　モデリングの課題解決（複合的要因）

7.2.1　問題カテゴリのバランス

上記 5 章で挙げた課題の仮説は、大きく分けて、開発ツールの問題、設計力の問題、運用（プロセス）の問題というカテゴリに分けて見ることができる。これらの関係を図 XX1 に示す。たとえばモデリング技術目的の曖昧さやアーキテクチャ設計の評価基準の曖昧さといった課題は設計力の問題に帰するだろう。また、開発フェーズの目的と異なるモデルの作成については、設計力も問題であると同時に、開発フェーズの規定のありかた、すなわち開発プロセスの問題でもあると言える。このように、それぞれの課題は一つまたは複数のカテゴリに属するという見方ができる。今後の調査によって新たな課題が明らかになった場合においても、図 7.2-1 上にマッピングできるものと考えている。

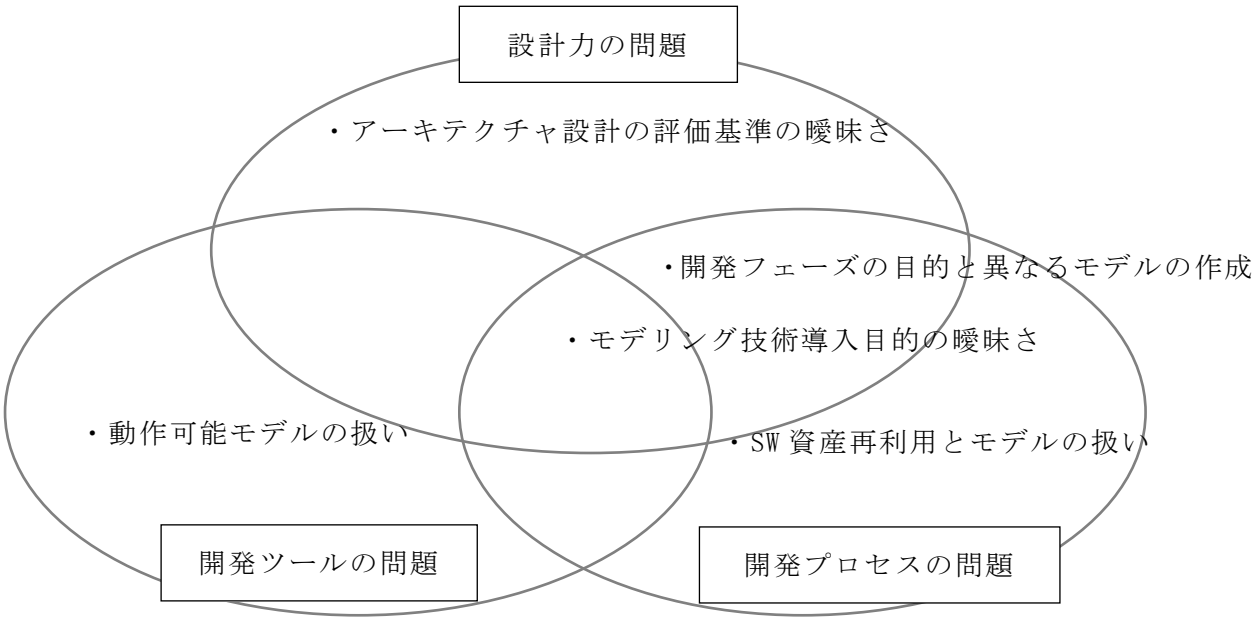


図 7.2-1　課題の仮説と問題カテゴリの関係

モデリングを成功させるためには、これらのいずれかが突出して優れていることよりむしろ、それぞれがバランスよく調和している必要があるだろう。たとえば、設計開発力が非常に優れた組織であっても､適切なツールなしには導入が困難なことは想像に難くない。また、設計開発力と適切なツールに恵まれていても、モデリングの成果を組織の中で効果的に活用するためには、運用ルールやプロセスが正しく機能しることも必要であろう。逆に、現場の設計開発力が十分でないところに過大なツールやプロセスを導入してもモデリングを成功につなげるのは難しい。売り文句につられて必要以上に高度なツールを導入するも本来の目的が達成できずに頓挫してしまう、などといったケースはありがちな事例である。

モデリング成功のためには、設計開発力のレベルが一定以上であることに加えて、目的に見合った開発ツール、適切に運用するためのプロセスの調和がとれていることが重要ではないだろうか。設計力に見合ったツールの機能、ツールや設計力に見合ったプロセスのありかたというものを判断するための指標を設けることができそうである。モデリング成功の鍵とみられるこれらカテゴリ間のバランスについて、開発現場で活用できるようなより具体的な指標があることが望ましい。

一旦これらのバランスがとれた状態でモデリングが成功すると、そこで新たな課題も見つかり、それを解決することで各カテゴリも少しずつレベルが上がっていくだろう。言う

なれば設計力・ツール・プロセスの三位一体による成功のスパイラルアップである。この様子を図 7.2-2 に示す。成功している組織ではこのようなプラスのスパイラルが形成されることで、人的にも組織の仕組みとしても成長する仕組みが形成されているものと考えられる。ここでのレベル課題解決によるレベルアップの程度についても、おそらく成功への鍵となる指標が存在するだろう。成功事例、失敗事例をより詳細に分析するなどして、こうした指標についても探っていきたい。

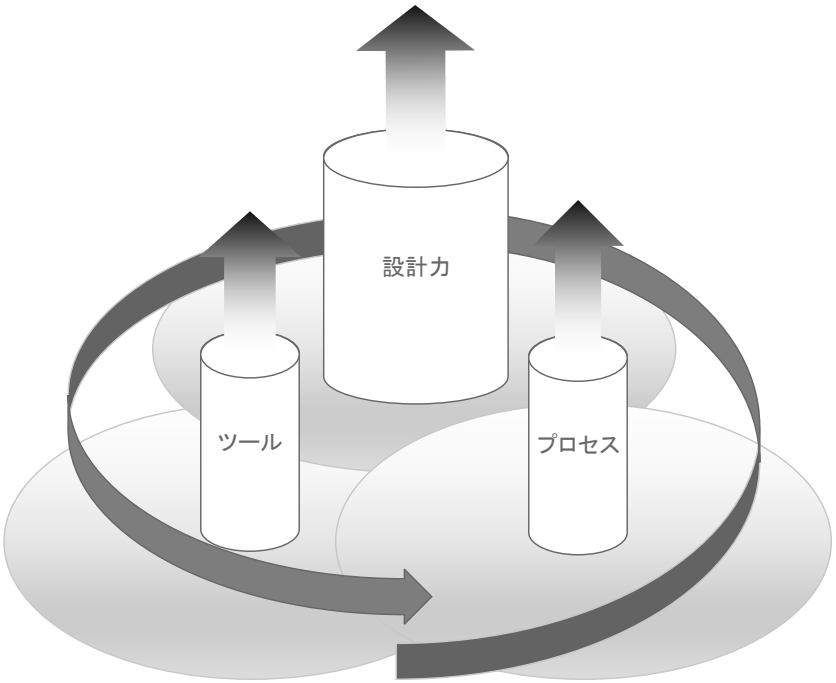


図 7.2-2　設計力・ツール・プロセスの三位一体による成功のスパイラル

7.2.2　モデリングの良し悪し

ここでは、チェンジビジョン平鍋氏を交えて行った、モデリングの良し悪しに関するディスカッションの概要について報告する。

モデリングの失敗とは何かを考えるにあたって、まずビジネスにとってのモデリングへの期待を確認する必要があろう。現代のビジネスゴールには、IoT、SoS、グローバル調達など、開発を複雑にする要因が多分に存在する。こうした環境においてビジネスゴールを達成するために、アジャイル開発、モデリング、品質確保のための活動など、現代的に有効なツールが活用される事例が珍しくなくなってきている。これらの関係を図 7.2-3 に示す。組み込み開発においては従来、モデリングのスコープとして、ビジネスゴールやビジネスアーキテクチャまでを視野に入れる必要があまり高くないケースが多かったかもしれない。それは、これまでは製品に求められる機能や性能が製品ごとに完結しており、外的要因の影響があまり大きくなかったからだといってもよいだろう。しかしながら近年の組み込み開発においては通信機能が標準的に搭載されるなど、外部システムとの関係を意識せざるを得なくなってきている。

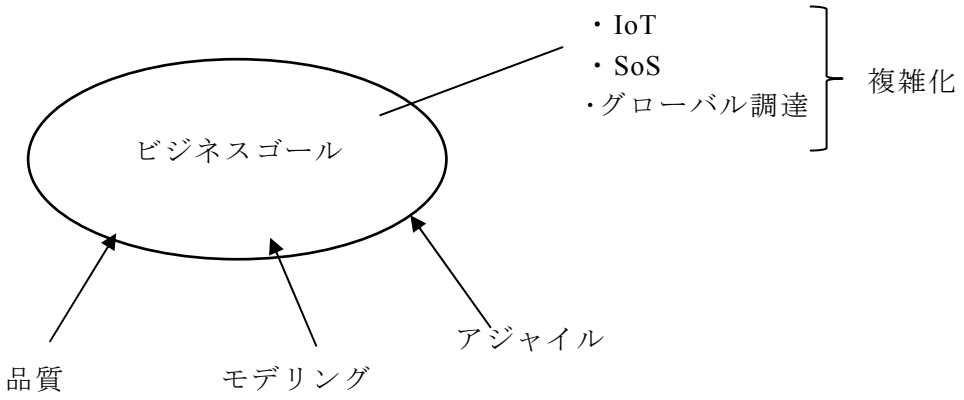


図 7.2-3　ビジネスゴールとモデリング

このうちモデリングに最も期待されているのは、開発現場およびステークホルダ間のコミュニケーションを円滑にすることだと言ってもよいだろう。コミュニケーションにも様々あり、誰に何を伝えるのかという目的によって用いる手段も異なる。また、開発の各段階（V 字モデルでの）においても適したモデリング手法は異なるだろう。たとえばビジネスモデリング、要求モデリング、アーキテクチャモデリング、システムモデリングといったように、段階に合ったモデリング手法が活用できると良い。さらに、グローバル調達も含めた外部委託の増加に供ない､そこに適したモデリングの必要性も高まってきている。

こうしたモデリングのニーズの高まりの中では、むしろどの段階でどのようなモデリングをするのか、取捨選択も求められる。経験的に費用対効果の高いモデリングのセットをミニマムセットとして扱う「これだけモデリング」はその一つの指標となるだろう。また、モデリングの結果として何を残して何を捨てるのか、コミュニケーションに最低限必要なセットを抽出することもドキュメントの爆発を防止する意味できわめて重要である。システムモデリングにおいては、例えば、静的なモデルのみを残し、ドキュメント量が多くなりがちな動的なモデルについては敢えて残さない、なども一つの方法である。

個々のモデリング技術向上が不可欠なのは勿論であるが、今後はこれらモデリングの成果を活用する技術も重要になってくるものと思われる。これら活用技術についても今後掘り下げていきたい。

7.2.3　アーキテクチャとモデリング

ソフトウェアアーキテクチャは一般に、そのシステムの構成要素となる各部分の役割を主に規定する静的な構造と、各構成要素及び構成要素間のふるまいを規定する動的な構造と、さらに設計の考え方を示す設計規則の類のものを含む。これらはいずれもシステム開発の秩序と一貫性を保つために、ステークホルダ間で正しく認識が共有されている必要がある。モデリング技術には、こうしたアーキテクチャの理解や伝達のための表現手段としての側面もある。

アーキテクチャのモデリングにとって重要な視点は、個別の構成要素を表現することよりも、全体構造をいかに的確に伝えられるかという点であろう。中国の諺「盲人模象」で、部分的な理解だけに基づいて全体像を正しく把握するのが困難であることが示唆されているように、システムの構成要素となる個別の「部分」を詳細に理解していても、それだけでは全体のアーキテクチャを正しく理解・表現することにはつながらない。全体構造の的確な表現・伝達のためには、それに適した表現手段を用いるべきである。（[CLE02],[KIS05]）こうしたアーキテクチャモデリングの世界においても、ミニマムセットとしての「これだけモデリング」が提案できれば、実際のプロジェクトで有用なビューの指針となるだろう。

7.3　まとめ

本章では、本年度の活動の中からモデリングの成功につながりそうな考察をいくつか紹介した。この中で、組み込み開発においては機能用件だけでなく非機能用件をモデルとして表現していくこと、開発の現場で利用価値の高いビューをミニマムセットとして抽出した「これだけモデリング」、さらに、設計力・ツール・プロセスが三位一体となってバランスがとれていることが重要なポイントとして浮かび上がってきた。現時点では仮説の段階だが、次年度以降の活動ではこれらの妥当性について検証していくと共に、より具体化した方策や指標についても検討していきたい。

8. まとめ

本章では 4 章～7 章までのモデリングに関する報告をまとめることにより、この組込み系モデリング報告書の全体のまとめとする。

8.1　モデリング技術の現状

4 章ではモデリング技術の位置付けと最近の技術状況を紹介した。モデリング技術の位置付けとして、モデルに期待される特性や Zachmann Framework によるモデル化の視点を紹介し、次にモデリング技術として、図表を用いたモデリングや仕様証明を目的とするモデリング、シミュレーションを目的とするモデリングに分類し、最近のモデリング技術やソフトウェア・システム開発での利用方法を紹介した。またモデルと開発プロセスと関係を説明し、最後に、特にアーキテクティングにおけるモデリング技術を整理し、モデルと開発プロセスの関係を説明した。

8.2　モデリングの課題認識

5 章では、実際に開発現場でモデリングを適用したときに現れる課題を提示した。組込み系の開発現場でのモデリングの課題は、制約条件や品質要求がモデリングされていないこと、モデルの評価基準が設定されていないこと、仕様の検証や妥当性確認にモデルを活用しにくいこと、技術導入の目的を明確化していないこと、また、再利用を想定したモデル化技術が未成熟であることを挙げた。これらの課題とその背景についての考察を説明した。

8.3　モデリング事例に見るモデリングの課題の考察

6 章では各種のモデリング事例からモデリングにおける課題を分類し考察した。考察対象にしたモデリング事例としては、ワークショップの事前アンケートや講演でのモデリング事例、委員会で収集した事例がある。これらのモデリング事例からモデリングの定義や必要なスキル、モデルへの期待、失敗／成功要因、課題などのカテゴリで個別の事例を抽出し、それを分類してまとめた。これからモデリングにおける課題の傾向性として、モデリング導入目的の曖昧さや目的と異なるモデルの作成、アーキテクチャ設計の評価などを考察し、それを紹介した。

8.4　モデリングの適用に対する課題解決の考察

7 章ではモデリングを適用するときに現れる課題に対する解決方法について述べた。最初にワークショップのアンケート事例からの課題を分類し、その中で性能などの非機能要件や制約条件のモデリングや再利用性の課題をピックアップして、その課題解決について考察した。モデリングの複合的要因による課題については、設計力や開発ツール、開発プロセスの三位一体による向上が課題解決の鍵になりそうだという考察を提示した。この考察を元に来年度はより具体的に解決への道を探っていく予定である。

8.5　まとめ

最後にモデリングをテーマにした初年度の活動をまとめる。モデリングをテーマにした初年度は、モデリングの現状分析から課題認識、課題分析そして課題解決に向けての活動を行ってきた。その結果、現状としては、モデリングそのものを実施していないところもあり、また目的を明確化せずに形式的に実施しているところもあり、課題は多くあることが分かった。その課題解決のために施策を打っているところもあるが、課題認識もされずにモデリングをしているところもあるなど、モデリングの実施はまだ弱いところがある。

このモデリングの現状から、当委員会ではモデリングにおける課題を分析し、なぜモデリングが失敗するのか、その逆にモデリングを成功させるにはどのようにすればいいかをこの報告書にまとめた。

9. 今後に向けて

2014 年度から 3 年間に渡る大きなテーマとして「モデリング」を選定し、2014 年度の当委員会の活動として、モデリングの現状と課題分析を行った。その活動の中でワークショップを開催し、有識者とのブレーンストーミングを行ってきた。

2015 年度の活動としては、2014 年度に行ったモデリングの現状と課題分析について、さらに深堀するところから始め、組込み系ソフトウェア開発におけるモデリングの課題解決に向けての施策を提言していきたい。

また広くモデリングに対する課題や施策の考えを集めるために、2014 年度と同様にワークショップを開催し、またブレーンストーミングやヒアリングなどを実施していく予定にしている。

最後に、この報告書で取りまとめた組込み系の課題状況やその分析、提言が読者の一助となることを望むものである。また、当専門委員会のアンケート調査にご協力いただいた方々、さらには組込み系ワークショップ 2014 に参加して貴重なご意見をいただいた皆様、また快くご協力頂いた講師の方々に、改めて深く謝意を表する次第である。

謝辞

本報告書の作成やワークショップ講演や参加、そしてモデリング・ブレーンストーミングにご協力いただいた方々に深く感謝いたします。

以下にご協力いただいた組織・団体、個人の方のお名前を紹介させていただきます。
（順不同）

・大阪大学　春名 修介教授
・富士通株式会社　奥村 一幸氏
・株式会社富士通コンピュータテクノロジーズ　石田 晴幸氏
・株式会社アマダ　久我 雅人氏
・ビースラッシュ株式会社　山田 大介氏
・NPO 法人組込みソフトウェア管理者・技術者育成研究会　渡辺 登氏
・株式会社チェンジビジョン　平鍋 健児氏

・NPO 法人組込みソフトウェア管理者・技術者育成研究会（SESSAME）
・一般社団法人 組込みシステム技術協会（JASA）

参考文献

[ATA07] 新 吉高,他,“組込みシステム向け HMI ツールのモデルベース開発について,” 日本データベース学会 Letters, Vol.6, No.2, pp.1-4, 日本データベース学会 (2007-09).

[AUT10] Automotive SIG, Automotive SPICE® プロセス参照モデル(PRM) v4.5J、 The SPICE User Group (2010-05).

[BUH95] Buhr, R.J.A., and Casselman, R.S., Use Case Maps for Object-Oriented Systems, Prentice Hall College Div (1995).

[BOE81] Boehm, B.W., Software Engineering Economics, Prentice Hall (1981).

[BOE84] Boehm, B.W., “Verifying and Validating Software Requirements and Design Specifications,” IEEE Software, Vol.1, No.1, pp.75-88, IEEE (1984-1).

[BOE88] Boehm, B.W., “A Spiral Model of Software Development and Enhancement,” IEEE Computer, Vol.21, No. 5, pp.61-72, IEEE Computer Society (1988-05).

[BOE00] Boehm, B.W.,et.a., Software Cost Estimation with Cocomo II, Prentice Hall (2000).

[BOO93] Booch,G., Object-Oriented Analysis and Design with Applications (2nd Edition), Addison-Wesley Professional (1993-03).

[BRA14] Bourque,P., and Fairley,R.E., Ed., Guide to the Software Engineering Body of Knowledge (SWEBOK) Version 3.0, IEEE Computer Society (2014).
松本 吉弘 訳, ソフトウェアエンジニアリング基礎知識体型－SWEBOK V3.0－, オーム社 (2014).

[CHE76] Chen,P.P., “The Entity-Relationship Model: Toward a Unified View of Data,” ACM Transactions on Database Systems, Vol.1, pp. 9-36, ACM (1976).

[CLE02] Clements.P., Bachmann,F., Bass,L., Garlan,D., Ivers,J., Little, R., Nord,R. and Stafford,J. Documenting Software Architectures: Views and Beyond, Addison Wesley (2002).
前田 卓雄 訳, ソフトウェア アーキテクチャ ドキュメント, 日刊工業新聞社 (2004).

[CMM02] CMMI Product Team, CMMI(SM) for Systems Engineering / Software Engineering, Version 1.1, Staged Representation (CMMI-SE/SW, V1.1, Staged), CMU-SEI Technical Report, CMU/SEI-2002-TR-002, CMU SEI (2002-01).

[COA90] Coad,P., and Yourdon,E., Object Oriented Analysis (2nd Edition) (Yourdon Press Computing Series), Prentice Hall PTR (1990).

[DEM79] DeMarco,T., Structured Analysis and System Specification, Prentice-Hall Inc. (1979).
高梨 智弘、高田 純一郎 監訳, 構造化分析とシステム仕様:目指すシステムを明確にするモデル化技法, 日経 BP 出版センタ (1994).

[DOB98] 土木学会 土木計画学研究委員会 「交通ネットワーク」出版小委員会編, 交通ネットワークの均衡分析―最新の理論と解法, 土木学会 (1998).

[FEI06] Feiler, P.H., et.al., The Architecture Analysis & Design Language (AADL): An Introduction, CMU SEI Technical Note, CMU/SEI-2006-TN-011, CMU SEI (2006-02).

[FEI09] Feiler, P.H., et.al., System Architecture Virtual Integration: An Industrial Case Study, CMU SEI Technical Report, CMU/SEI-2009-TR-017, CMU SEI (2009-11).

[FEI12] Feiler,P.H., and Gluch, D.P., Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language, Addison-Wesley Professional (2012).

[FOW03] Fowler,M., UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition), Addison-Wesley Professional (2003).
羽生田 栄一訳, UML モデリングのエッセンス 第 3 版 (Object Oriented SELECTION), 翔泳社(2005).

[FRA03a] Frankel,D.S., Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley (2003).
日本アイ･ビー･エム TEC-J MDA 分科会訳, MDA:モデル駆動アーキテクチャ, エスアイビーアクセス (2003-11).

[FRA03b] Frankel,D.S., et.al., “The Zachman Framework and the OMG’s Model Driven Architecture,” Business Process Trends White Paper, Business Process Trends, http://www.bptrends.com/ (2003-09).

[FRI09] Friedenthal,S. et.al., A Practical Guide to SysML: The Systems Modeling Language (The MK/OMG Press), Morgan Kaufmann (2009).
西村 秀和 訳, システムズモデリング言語 SysML,東京電機大学出版局 (2012).

[FUK13] 深野 善信, 他, “複雑化する車載制御ソフトウェア開発を支える最新電子プラットフォーム技術,” 日立評論, Vol.95 No.11 (2013 年 11 月号), pp.72-75, 日立製作所 (2013-11).

[GOT12] 後藤 芳顯, “構造設計における FEM 解析の高度化と信頼性の向上,” 第 15 回 鋼構造と橋に関するシンポジウム論文報告集, 橋梁設計における最新FEM解析技術の適用と課題, pp.1-13, 土木学会鋼構造委員会 (2012).

[HAL96] Hall.,A., “Using Formal Methods to Develop an ATC Information System,” IEEE Software, Vol.13, No.2, pp. 66-76, IEEE Computer Society (1996-03).

[HAR87] Harel,D., “Statecharts: A Visual Formalism for Complex Systems,” Science of Computer Programming, Vol.8, Issue 3, pp.231–274, North-Holland (1987-06).

[HAT12] 端山 五郎, “自動改札機ソフトウェアの品質向上の取組み―厳密な仕様、もらさないテストを目指して―,” ソフトウェア品質シンポジウム 2012 C2, 日本科学技術連盟 (2012).

[HAY03] Hay,D.C., Requirements Analysis: From Business Views to Architecture, Prentice Hall PTR (2003).

[ICH08] 市原 利浩, 他, “モデル駆動開発によるシステム性能検証手法の提案,” 情報処理学会第 70 回全国大会, 2B-1,pp.1-231-1-232, 情報処理学会 (2008).

[IEE98] Software Engineering Standards Committee of the IEEE Computer Society, IEEE Std. 830-1998: IEEE Recommended Practice for Software Requirements Specifications, IEEE (1998-06).

[IEE00] IEEE Computer Society, IEEE Std. 1472-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Computer Society (2000).

[IPA07a] 独立行政法人 情報処理推進機構 ソフトウェア･エンジニアリング･センター, 高信頼ソフトウェア構築技術に関する動向調査 調査報告書, https://www.ipa.go.jp/sec/softwareengineering/reports/20080606.html (2007-06).

[IPA07b] 独立行政法人 情報処理推進機構 ソフトウェア･エンジニアリング･センター, 共通フレームワーク 2007 第 2 版～経営者,業務部門が参画するシステム開発および取引のために～, オーム社 (2007).

[IPA07c] 独立行政法人 情報処理推進機構 ソフトウェア･エンジニアリング･センター ,改訂版 組込みソフトウェア向け開発プロセスガイド (SEC BOOKS) , 翔泳社 (2007-11).

[IPA12] 独立行政法人情報処理推進機構 技術本部 ソフトウェア･エンジニアリング･センター,「組込みシステムの先端的モデルベース開発実態調査」調査報告書, http://www.ipa.go.jp/files/000004608.pdf, 独立行政法人 情報処理推進機構 (2012-03).

[IPA12b] 独立行政法人情報処理推進機構 技術本部 ソフトウェア･エンジニアリング･センター, 組込みソフトウェア開発における品質向上の勧め 〔テスト編～事例集～〕, https://www.ipa.go.jp/files/000005149.pdf, 独立行政法人 情報処理推進機構 (2012-11).

[IPA13] 独立行政法人情報処理推進機構 技術本部 ソフトウェア・エンジニアリング・センター,モデルベース開発ツールを活用した際のコストの効果検証実施報告書, http://www.ipa.go.jp/files/000026867.pdf, 独立行政法人 情報処理推進機構 (2013-02).
[IPA13b] 独立行政法人情報処理推進機構 技術本部 ソフトウェア・エンジニアリング・センター, 2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書, http://www.ipa.go.jp/files/000026799.pdf, 独立行政法人 情報処理推進機構 (2013-04).
[IPA14] 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, モデルベースシステムズエンジニアリング導入の手引き, 2013-08
[IPA15] 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター, 組込みソフトウェア開発における品質向上の勧め[設計モデリング編], (SEC BOOKS), 2006-06
[ITU11] Telecommunication Standardization Sector OF ITU, Z.101: Specification and Description Language – Basic SDL-2010, Recommendation ITU-T Z.101, ITU (2011).
[IMA11] 今村 文彦, “東日本大震災での津波被害と数値シミュレーションへの期待,” 地球シミュレータ／HPCI 戦略プログラム(分野 3)合同シンポジウム ～防災・減災に資する地球変動予測「京コンピュータ」との連携～, 独立行政法人 海洋研究開発機構 (2011-09).
[ISO07] 磯田 誠, 他, “モデル駆動によるテスト仕様の生成手法の提案,” 情報処理学会研究報告. ソフトウェア工学研究会報告 2007(33), pp.89-96, 情報処理学会 (2007-03).
[ISO11] International Standard ISO/IEC/IEEE42010, Systems and software engineering — Architecture description, ISO (2011).
[JAC92] Jacobson, I., Object Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley Professional (1992).
[JIS03] 日本規格協会, JIS-X-0129-1:2003(ISO/IEC 9126-1:2001) ソフトウェア製品の品質—第1 部:品質モデル, 日本標準工業調査会 (2003).
[JIS04] 日本規格協会, JIS-X-0170:2004 (ISO/IEC 15288:2003) システムライフサイクルプロセス, 日本標準工業調査会 (2004).
[JIS12a] 日本規格協会, JIS-B-3503:2012 プログラマブルコントローラープログラム言語, 日本標準工業調査会 (2012).
[JIS12b] 日本規格協会, JIS-X-0160:2012 (ISO/IEC 12207) ソフトウェアライフサイクルプロセス, 日本標準工業調査会 (2012).
[JIS13] 日本規格協会, JIS-X-25010:2013 (ISO/IEC 25010:2011) システム及びソフトウェア製品の品質要求及び評価(SQuaRE)−システム及びソフトウェア品質モデル, 日本標準工業調査会 (2013).
[JUA10] 日本情報システム・ユーザ協会編, ユーザ企業ソフトウェアメトリクス調査【調査報告書】2010年版, http://www.juas.or.jp/servey/library/pdf/10swm.pdf,日本情報システム・ユーザ協会 (2010-09).
[KAS06] 笠原 正治, “トラヒック理論のインターネット設計への応用,” 電子情報通信学会・通信ソサイエティ 電子情報通信学会 通信ソサイエティ 2006 年ソサイエティ大会 チュートリアル, BT-2-4, 電子情報通信学会 (2006-09).
[KAW14] 川勝 則孝, 川田 秀次, “モデルベース開発に向けたソースコードからの状態遷移モデル抽出技術,” 東芝レビュー, Vol.69, No.8, 東芝 (2014-08).
[KIS05] 岸 知二, 野田夏子, 深澤良彰, ソフトウェアアーキテクチャ, 共立出版 (2005).
[KOJ00] 小島 芳生, “車両開発における構造・機構の CAE,” 豊田中央研究所 R&D レビュー, Vol.35, No.4, pp.1-8, 豊田中央研究所 (2000-08).
[LAM00] Lamsweerde, A.V., “Formal Specification: a Roadmap,” ICSE '00 Proceedings of the Conference on The Future of Software Engineering, pp.147-159, ACM (2000).

[MAH08] Mahé,V., “OpenEmbeDD as an MDE platform: A powerful set of tools to improve your DSL work,” 4ième Journées sur l'Ingénierie Dirigée par les Modèles (IDM2008), Centre national de la recherche scientifique(CNRS) (2008).

[MAR05] Marca,D.A., and McGowan,C.L., IDEF0 and SADT: A Modeler's Guide, OpenProcess, Inc (2005).

[MET04] 組込みソフトウェア開発力強化推進委員会, 2004 年版組込みソフトウェア産業実態調査報告書, 経済産業省(2004).

[MET09] 横断型基幹科学技術研究団体連合, 分野横断型科学技術アカデミック・ロードマップ報告書, 経済産業省 平成 20 年度技術戦略マップローリング事業（アカデミック・ロードマップ）, 経済産業省 (2009-03).

[NAG09] 長坂 康史, 金只 圭司, “アドホックネットワークにおけるノード負荷を考慮した高効率ルーティング手法に関する研究,” 広島工業大学紀要研究編, Vol.43, pp.305-310, 広島工業大学 (2009).

[OHO09] 於保 茂, 他, “エンジン制御モデルベース開発の先進技術,” 日立評論, Vol.91, No.10, pp.792-793, 日立製作所 (2009-10).

[OMG09] Object Management Group, OMG Unified Modeling Language™ (OMG UML), Superstructure, Version 2.2, formal/2009-02-02, OMG (2009).

[OMG11] Object Management Group, UML Profile For MARTE: Modeling And Analysis Of Real-Time Embedded Systems, formal/2011-06-02, OMG (2011).

[PET81] Peterson, J.L., Petri Net Theory and the Modeling of Systems, Prentice Hall (1981).
市川 惇信, 小林 重信訳, ペトリネット入門—情報システムのモデル化, 共立出版 (1984).

[RAM91] Rambaugh, J., et.al., Object-Oriented Modeling and Design, Prentice-Hall, Schenectady, NY (1991).

[SAK10] 坂本 佳史他, “組込みシステムのアーキテクチャ設計と性能評価のためのモデルベース手法,” ProVISION, No.64, pp.89-95, 日本 IBM (2010).

[SEL94] Selic,B., Gullekson,G., Ward,P.T., Real-Time Object-Oriented Modeling, Wiley (1994).

[SHL88] Shlaer,S., and Mellor, S.J., Object Oriented Systems Analysis: Modeling the World in Data, Prentice Hall (1988).

[SQU13] SQuBOK 策定部会編, ソフトウェア品質知識体系ガイド（第 2 版）－SQuBOK Guide V2－, オーム社 (2014).

[TAD10] 只野 賢二, 栗田 太郎, “モバイル FeliCa IC チップ開発における SPIN を用いたモデル検査による品質確保,” Technical report of IEICE, SS 110(169), 29-34, 電子情報通信学会 (2010-07).

[TAM97] Tamura, N., “ROAD/EE: a prototyping environment for object-oriented specifications,” Proceedings of Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23., pp. 176-189, IEEE Computer Society (1997-08).

[TAN13] 丹野 治門他, “ソフトウェアの品質確保と開発コスト削減を目指したテスト自動化技術,” NTT 技術ジャーナル, Vol.25, No.10, pp. 19–22, NTT (2013-10).

[TOM97] 富井 規雄, “カラード ペトリネットによる列車運行モデルとその列車ダイヤ作成システムへの応用,” 電気学会産業システム情報化研究会資料 IIS-97-25, pp.39-44, 電気学会 (1997).

[TON10] 斗納 宏敏, “仮想環境による自動車制御ソフトウェアの評価手法,” ISIT 第 6 回カーエレクトロニクス研究会, 公益財団法人 九州先端科学技術研究所 (2010-05).

[YAM98] 山中 弘, 他, “マイコン組み込み S/W の開発を支援する CASE ツール ：testCASE(2)：状態遷移表とテスト手順生成,” 情報処理学会 全国大会講演論文集 第 57 回平成 10 年後期(1), pp.248-249, 情報処理学会 (1998-10).

[YAM11] 山崎 光悦, “CAE と設計の基礎,” 日本機械学会設計工学・システム部門 No. 11-117 講習会「革新的ものづくりのための最適設計法入門」, (1), 日本機械学会 (2011-11).
[YAS14] 安岡 大知, “ZIPC 状態遷移表のモデル検査適用への取り組み,” 第 19 回 ZIPC ユーザーズカンファレンス, キャッツ株式会社 (2014-10).
[YOS94] 吉本 久泰, PC シーケンス制御—入門から活用へ, 東京電機大学出版局 (1996-03).
[ZAC87] Zachman,J.A., “A Framework for Information Systems Architecture,” IBM Systems Journal, Vol.26, No.3, IBM (1987).
[ZAN13] 張暁晶 他, “設計書に基づくテスト項目抽出手法の提案および既存開発プロセスへの適用,” 電子情報通信学会技術研究報告 SS, ソフトウェアサイエンス, Vol. 113, No.269, pp.167–172, 電子情報通信学会 (2013-10).

付　録

付録 1.　ワークショップ 2014 事前アンケートまとめ

付録 2.　ワークショップ 2014 事前アンケート回答詳細

付録 3.　ワークショップ 2014 当日アンケート回答詳細

報告 1.　組込み系ソフトウェア・ワークショップ 2014

報告 2.　CEATEC JAPAN 2014 講演

報告 3.　モデリング・ブレーンストーミング

付録 1. ワークショップ 2014 事前アンケートまとめ

当専門委員会が 2011 年度～2014 年度に実施したワークショップでの「事前アンケート調査」に対して回答をして頂いた回答者の概要を以下に示す。

1. 担当/関与した仕事の内容

1.1 経験のある開発対象

経験のある開発対象の回答結果を図 1.1-1 に示す。

「コンピュータ用周辺機器・OA 機器」が最も多く、次が「AV 機器」であった。2011 年度のアンケートでは、「AV 機器」これに続いて「通信端末」、「通信設備機器」、「工業制御／FA 機器／産業機器」や「エンタープライズ系」との回答割合が多い。

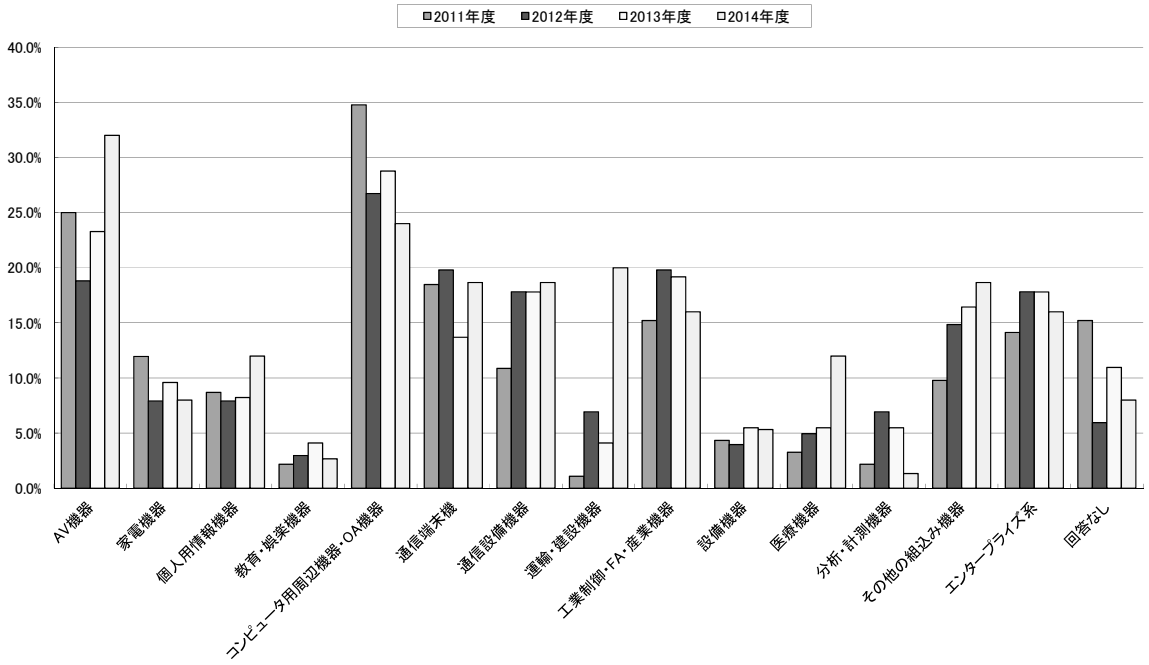


図 1.1-1 貴方が、過去を含めて、担当/関与されたお仕事の内容（組込み系、エンタープライズ系等々）について、差し障りのない範囲でご記入願います。（複数選択可）

1.2 発生している問題

発生している問題の回答結果を図 1.1-1 に示。

「短納期化」、「複雑化」、「大規模化」に起因する問題が同様の割合で、半数近くの方が発生しているとの回答している。

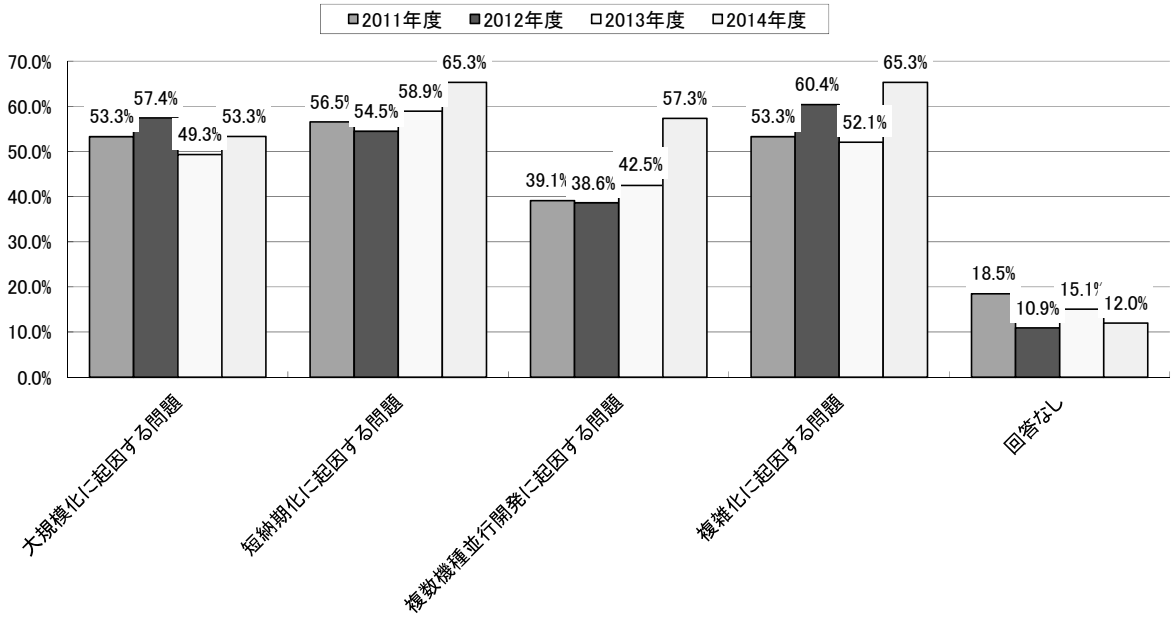


図 1.2-1 発生している問題についてチェックしてください。（複数回答可）

1.3 経験のある開発形態

経験のある開発形態の回答結果を図 1.3-1 に示す。

擦り合わせ型の開発が若干多い傾向を示している。

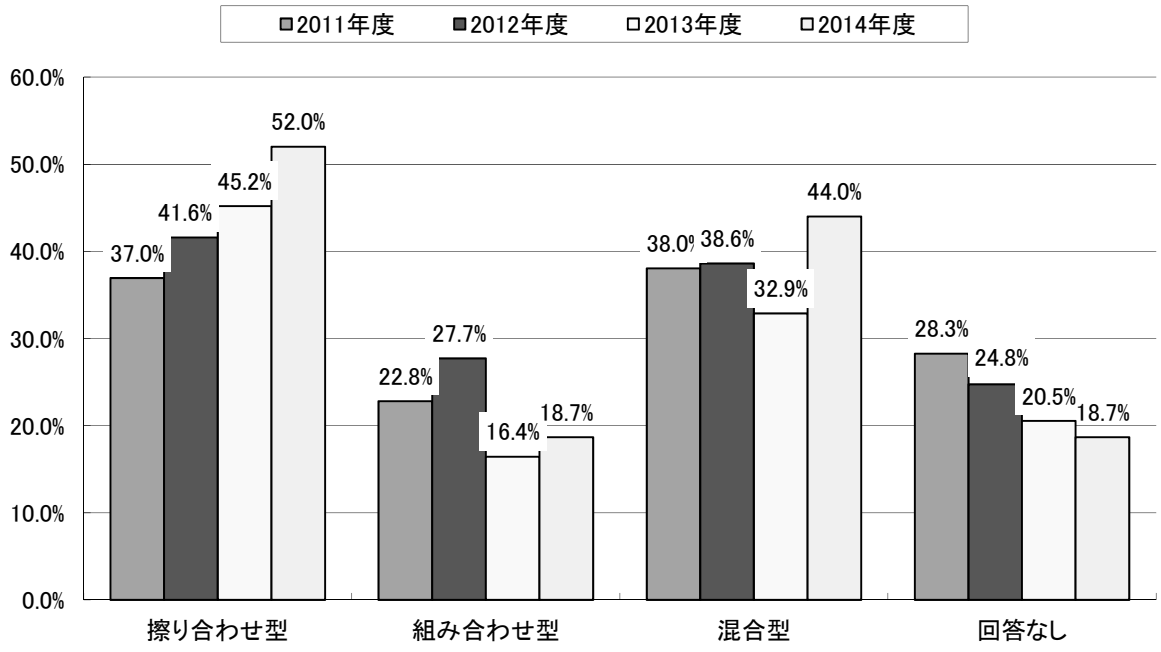


図 1.3-1 経験のある開発形態についてチェックしてください。（複数回答可）

付録 2. ワークショップ 2014 事前アンケート回答詳細

当専門委員会が実施したワークショップ 2014 での「事前アンケート調査」に対する 75 名からの回答の集計結果データを以下に示す。

【問１－１】貴方が、過去を含めて、担当/関与されたお仕事の内容（組込み系、エンタープライズ系等々）について、差し障りのない範囲でご記入願います。	
24	AV 機器（TV、DVD、ﾃﾞｼﾞﾀﾙｶﾒﾗ、ｾｯﾄﾄｯﾌﾟﾎﾞｯｸｽ、ｵｰﾃﾞｨｵ機器、等）
6	家電機器（電子ﾚﾝｼﾞ、炊飯器、ｴｱｺﾝ、洗濯機、冷蔵庫、等）
9	個人用情報機器（個人情報端末、電子手帳、GPS、ｶｰﾅﾋﾞ、等）
2	教育機器、娯楽機器（ｹﾞｰﾑ機、電子楽器、電子辞書、玩具ロボット、等）
18	コンピュータ周辺機器／OA 機器（ﾌﾟﾘﾝﾀ、ｽｷｬﾅ、ｽﾄﾚｰｼﾞ機器、複写機／複合機、ﾌｧｸｼﾐﾘ、等）
14	通信端末機器（民生用端末機器）（固定電話機、携帯電話端末、スマートフォン、PHS 端末、等）
14	通信設備機器（ﾙｰﾀ、通信網用ｽｲｯﾁ、放送機器、無線機器、等）
15	運輸機器／建設機器（自動車、船舶、飛行機、ｵｰﾄﾊﾞｲ、ﾌﾞﾙﾄﾞｰｻﾞ／ｼｮﾍﾞﾙ機、等）
12	工業制御／FA 機器／産業機器（ﾌﾟﾗﾝﾄ制御、工業用ロボット、印刷機器、縫製機械、等）
4	設備機器（ｴﾚﾍﾞｰﾀ／ｴｽｶﾚｰﾀ、照明機器、空調機器、等）
9	医療機器（診断・検査装置、個人用検査機器、福祉機器、等）
1	分析機器・計測機器　（分光光度計、ﾛｼﾞｯｸｱﾅﾗｲｻﾞ、質量分析機器、電子顕微鏡、等）
14	その他の組込み機器
12	エンタープライズ系
6	記載なし
【問１－２】以下のものから発生している問題についてチェックしてください。（複数回答可）	
40	大規模化に起因する問題
49	短納期化に起因する問題
43	複数機種並行開発に起因する問題
49	複雑化に起因する問題
9	記載なし
【問１－３】経験のある開発形態についてチェックしてください。（複数回答可）	
39	擦り合わせ(インテグラル)開発
14	組み合わせ(モジュラー)開発
33	擦り合わせ(インテグラル)開発と組み合わせ(モジュラー)開発の混合型
14	記載なし
【問２】モデリングの目的は何ですか？（複数回答可）	
35	仕様の妥当性確認（Validation）
5	コードの検証
13	コードの自動生成
38	仕様の整理
17	仕様の調整
12	シミュレーション（顧客、技術）
32	仕様の検証(Verification)
35	品質改善
28	再利用
15	工期短縮
31	コミュニケーション
6	リスク対策
5	その他
その他の回答	
12	既存ソースコードの解析
16	機能安全規格への対応
23	実現可能性の検証，ネックポイントの発見・対策検討
44	ハードソフトインタフェースのすりあわせ
74	開発手法としてどのような効果があるか全般的な興味

【問３】どのようなモデリングをしていますか？
その対象や詳細度はどの程度でしょうか。
その現状・効果と手法・図はどのようなものですか？（複数回答可）
現状・効果は、以下でお答えください
1-検討しておらず実施していない、2-検討はしたが実施していない、3-導入したが運用に至っていない、4-現場で導入しているが課題が多く活用しているとは言えない、5-現場で導入し、活用している

ビジネス（プロセス）モデリング（ビジネスの方針やプロセスを決定するモデル）をしていますか？	
現状	
41	未実施
14	検討のみ
1	未運用
5	未活用
7	活用
手法、図、ツール	
7	EA
2	BPMN
1	IDEF
15	UML
5	SysML
10	フローチャート
14	状態遷移図
7	ワークフロー図
12	Excel(表)
7	その他
その他	
7	PFD
23	DFD
25	CSP モデル
33	CVCA、Pict 図、WCA、Business Model Canvas、9 セルメソッド
46	・TOC（制約理論）の思考プロセスや S&T（Strategy&Tactic）Tree も有効と考える。 ・BABOK でリストアップされているようにビジネスアナリシスには様々な手法があり、それの応じて多様なモデルが使われている。
59	PFD
73	マインドマップ

要求モデリングをしていますか？	
現状	
41	未実施
14	検討のみ
1	未運用
5	未活用
7	活用
手法、図、ツール	
7	EA
2	BPMN
1	IDEF
15	UML
5	SysML
10	フローチャート
14	状態遷移図
7	ワークフロー図
12	Excel(表)
7	その他
その他	
8	XDDP
25	Z/Object-Z/B
46	・USDM は有効な方法の一つと考えるが、外部とのインタフェース仕様の記述はユースケースモデルの方が明快と考える。 ・ユースケースモデルだけで要求を記述するのは望ましくない。RUP でも要求モデリングは要求管理ツールによる要求マトリクスと組み合わせてモデリングするように示してある。

	・旧来から製造業で使われていた QFD は有効な方法の一つと考える。（要求管理の原型であり、1 枚の図で要求から設計までを表現する手法が特徴的であり、今でも有効な手法と考える）"
59	ユースケース図、要求図、USDM
64	Raquest
65	ユースケース図
73	マインドマップ
アーキテクチャモデリングをしていますか？	
現状	
16	未実施
14	検討のみ
1	未運用
8	未活用
29	活用
対象	
16	システム
37	ソフトウェア
4	ハードウェア
手法、図、ツール	
12	EA
35	UML
10	SysML
4	その他
その他	
23	DFD，プロセス構造図，状態遷移図
25	CSP モデル
73	マインドマップ
データモデリングをしていますか？	
現状	
33	未実施
10	検討のみ
1	未運用
5	未活用
14	活用
対象	
9	概念モデル
15	仕様モデル
10	実装モデル
手法、図、ツール	
3	EA
13	UML
4	SysML
7	E-R 図
11	DFD
4	その他
その他	
4	何を指しているのかよくわかりませんでした
7	CRUD
23	データ相互関連図
38	Excel
46	・オブジェクト指向的な設計をするならば、データモデルが アーキテクチャモデルのベースにもなるのでデータモデリングは重要と考える。 ・個人的には、E-R 図よりクラス図によるデータモデリングの方が関連をキーで示さなくてよいので描きやすい。
設計モデリングをしていますか？	
現状	
14	未実施
11	検討のみ
2	未運用
7	未活用

31	活用
対象	
22	概念モデル
30	仕様モデル
29	実装モデル
手法、図、ツール	
35	UML
5	SysML
18	フローチャート
9	その他
その他	
3	EA
23	状態遷移図，状態遷移表
30	Simulink, Stateflow
38	Excel
41	Matlab/Simulink
46	・組込み系では状態とシーケンス両方の情報を入れられるタイミング図の利用が有効と考えます。
65	ユースケース図
73	状態遷移図、マインドマップ

【問４】モデリングに対する色々なお考えをお聞きします。
例えば、モデルの定義や役割などから、モデリングをするときの課題、モデリングを効率よく実施するコツ、またモデリングは誰でもできるのか、モデリングのためにはどのようなスキルが必要なのかなど、モデリングに対する考えや思いをお書きください。
方針的なものから具体的なものまで何でも結構です。

2	モデリング対象の詳細な部分まで突き詰めて理解しなければ、また理解できなければ、モデリングすることはできないと思う。別の言い方をすると、モデリング対象を詳細な部分まで突き詰めて理解したことを表現するための技術がモデリング技術と考える。 つまり、だれでもコツさえつかめれば、モデリングできるものではなく、要求事項を実現するために、モデリング対象を隅々まで詳細に理解することができるスキルまたは、理解しようとするスキルがある人だけが行う作業。
4	モデリングのコツは、トップダウンで俯瞰的にシステムを捉えることと、細かい仕様レベルまで把握し、整理することの、両方から攻めること。特に、複数機種に展開する場合は、そのどちらが欠けても「使える」モデルにならない。
7	モデルとは，対象のある側面にだけ着目した抽象的な記述.
	良いモデルとは，モデル作成者の主観が客観的に表現されているもの．特徴としては，着目した側面が，一意に解釈でき，他の側面に着目したモデルとの関係が明確である.
9	顧客、マネージメント、現場設計者、テスターなどステークホルダがモデル化に期待する内容が異なると問題になると思います。
11	階層化することで概念はわかりやすくなりますが、実装に近づくにつれて自然言語が増えていってしまうという課題に陥ってしまう傾向にある。適した工程で、適したモデリングを行うことができれば、上記課題が削減できるのではないかと思っている。
	今回は実践的なモデリングを通じて、お客様および委託先に対して過不足のない情報を提供できるようになりたいと思っている。
12	「モデリングはなぜ失敗するのか？」という WS の副題に違和感があります．モデリングの「失敗」とは何でしょうか？何をもってモデル化が上手くいっていないと判断するかにも依りますが，もし，組込み系のソフト開発で開発対象がモデル化できない，或いはモデル化しにくいことが問題だとするのであれば，組込み系の設計手法がモデルと親和性が低いためではないかと思っています. 現在開発から離れていますが，振返ると設計時に手続き中心の構造，制御構造に着目し過ぎ，裏返せばデータを軽視し過ぎていたように思います．データ中心設計へのシフトがオブジェクト指向設計，モデルへの親和性の向上に繋がるのではないでしょうか
	また，モデリングに最も必要な能力は，抽象化能力であり，次は，制約の下での表現力ではないかと思っています.
13	モデリングツールを導入しても結局使う側のスキルがなければ意味のないものになってしまうので、如何にスキルを高めて使うかが課題
	車載系の組込分野ではモデルベース開発が浸透してきている状況化で、どういった分野に適用されていくのか、ニーズの広がりをつかみたい。
15	「モデリング」とは何か、「モデリング」をどのように利用していけば、開発効率化・品質向上に役立てられるかを、勉強したいと考えております。
	「モデリング」を運用するにあたって、全員の「モデリング」に対する知識が必要だと

	思いますが、全員が理解してもらうためにどのような課題があるのかを知りたいと考えております。
16	派生開発におけるモデリングの活用。変更指示と言う関心事を表現できる図が UML に存在しない。現状の枠組みで変更指示を表現しようとすると、構造と振る舞いの図を Before と After で並べて示すことになる。この場合、間違いさがしと同じで、人間の認知能力に多大な負荷をかけるため、適切にレビューできない。
19	これまで民生品、産業用途の組み込み開発に携わってきましたが、すでに出来上がった製品をバージョンアップ、改修するなどメンテナンス型がほとんどで、1 から組み上げる経験がそれほどありません。知識不足だとは思いますが、よろしくお願い致します。
	モデリングを実践することによって期待することは、チーム内の開発効率の改善と、品質の向上です。コミュニケーションツールとして活用できればと考えております。
22	これまで所属する開発チームでは、モデリングを活用していません。モデリングをチームに導入するためのステップや事例など興味があります。
23	モデリングの目的を明らかにしておくことが大切と思う． 例えば，コード自動生成が目的のモデリングは，モデリングではなく，高位言語によるプログラミングと捉えるべきと思う．おまけとして「自動生成する/できる」なら問題はないのですが,. . .
	複数の視点からのモデリングが求められる．対象システムによって，重要となるモデリングの視点・視点の組合せが異なる（はず）．システム毎に，どの視点が重要かの整理ができると，モデリングの有効な利用が拡大していくと思う．
	モデリングの主目的は，他人とのコミュニケーション（各種縮小モデル，ファッションモデル等），影響・ふるまいの評価（空洞実験用のモデル，GUI のプロトタイムモデル，各種シミュレーションモデル等）だろうと考えています．もっと違う目的に気付くことができるワークショップとなることを期待しています．
24	自分自身が行っている仕様整理がモデリングに相当するのか、判断出来ていない
	モデリングにより、他人との仕様整合がスムーズになるようにしたい
25	組み込み系で必要となるモデルは、並行処理とリアルタイム処理です。並行処理が複雑になると、UML/DFD などのモデリングのツールを使用して、仕様の振る舞いを表現しても、証明された事にはならない。欧米ではモデリングのツールから CSP モデルに変換し、FDR2/PAT3/ProB などのモデル検査ツールを使用して、検証する方法が盛んに行われています。更に、付け加えるならば、プログラミングが安全に動作するためには、証明されたものでなくてはなりません。C 言語は本当に証明された言語なのでしょうか？マルチスレッドの競合の問題は放置されたままです。マルチコア、メニーコアの出現は並列処理プログラミングのスキルが必要となります。どうするのでしょうか？マイクロプロセッサも同じことが言えます。日本では、こういった議論がない。ヨーロッパは既に、20 年前にこの問題を解決しています。この一線を超えない限り、日本の組み込みシステムは、世界のレベルには届きません。何時までたっても根性論の議論しか見えません。2013 年に ACM/IEEE はコンピュータ・サイエンスのガイドラインを提示し、米国の大学の学部の教育に並列処理のコースが必須となりました。UK の大学の多くが CSP 理論を学部の教育でやっています。シンガポール大学、中国のトップも CSP 理論の教育に力を入れています。日本の産業界、学界、大学はこういった事を世界を知らない様に思えます。如何でしょうか？
26	どの工程においても明確にモデリング手法を導入しておらず実際に開発が進行している状況で、モデリングを導入する際、どのフェーズから導入すると効率的か、それとも同時で問題ないか等、効率的なモデリング導入について教授願いたい。
27	想定される登場人物を、漏らさずピックアップする事。
	シナリオを用いて、場面を想定して運用を検討する。
	既成概念にとらわれない。
28	コツ：ダイアグラムを記載する目的を明確にすること
29	モデリングの定義がはっきりとわかっていない。仕様書（設計書）を作るときに UML を使って記述していればそれもモデリングなのか？など。
30	モデルベース開発に取り掛かったばかりで試行錯誤しています．モデル作成のレベル，粒度について指針が必要だと思います．
31	モデルと実装の不一致が発生し始めたとき、乖離をどのようにリカバリーすればよいか？を解決できる手段があれば知りたい。
33	モデリングについては、組織で、表記法、ルール、粒度など典型的な事例を用いで普及、運用促進する仕組みが重要と思います。
	またモデリングを推進している人が、経営に与える効用をよく理解し、経営層、ミドル、現場の技術者に丁寧に、粘り強く説明し、展開していく必要があります。モデリングを実施することで、対象を可視化することができ、仕様の理解、早期の検証が可能となると同時に、設計、実装、V&V の開発の効率化、資産再利用、事業ロードマップ策定など、様々な側面で効果が期待できると考えています。

35	excel でのモデリングは誰もが見ることができ便利なのですが､新しいツールの導入を考えております。簡単に習得でき、使いやすいツールの模索中です。
38	スキルや経験に依存せず、書くべきモデルを分かり易くすばやく書くコツやガイド・ツールがあればよいと思います。
40	毎回参加させて頂いておりますが、自分ではソフトの開発を行っている訳では有りません。コンサルの端くれとしてソフト開発等を行う中小企業等と付き合うに際して勉強になる機会として参加させて貰っております。宜しくお願いします。
41	「ツールを使う」あるいは「表記法として使う」にとどまり、本質的なモデリングの検討が十分なされていない状況を克服するには？
42	成果物の再利用性が高まることを期待しています。
42	サブシステム間の関係性が整理されることで、システムの信頼性や保守性が向上することを期待しています。
	モデルを記述することを通じて、開発スタッフの意思疎通や知識継承が促進されることを期待しています。
43	他の人に説明する為にモデリングを行いたい。でも、どこからどこまで、何をどのようにモデリングしたら良いのか分からない。 書籍を読んだり、簡単に試してみているが、これで良いのか？何が正解なのか分からないし、自信も持てない。 諦めずに Try しているが、仕事に適用できない。
44	非常にハードウェアに近い位置で、ハード・ソフトの最適なインタフェースを決定するためにモデリングを適用中。失敗しないようなノウハウを吸収したい。
45	数年前からソフトウェア開発にモデリングを取り込んでいますが、まだ十分活用できていません。特にシステム稼動後に不具合や仕様変更があっても、ソースコードのみが修正されてしまい、モデリングと実装がどんどん乖離していく現状があります。
46	モデリングとは、物事を特定の視点で抽象化し、図や表を使って理解しやすく整理することだと考えます。 モデリングは目的ではなく、手段なので何のためにモデリングをしているかを意識して作業することが重要だと考えます。
	厳密なルールのもとで、モデリングしないと、モデルが安定せず、モデルに対する議論も発散してしまいます。
	すぐれたモデリングを効率よく実施するためには、コミュニケーションをとって複数の知見をとりこむことが重要と考えますが、意見を整理するファシリテータや、モデルに対して責任を持つ者を明確化しないと、議論が発散してかえって時間がかかる可能性もあります。
47	モデリングツールはあるが、それを使いこなせているかは疑問。例えば要求モデリングにおいて、成果物としてモデルは出来ているが要求自体が曖昧で不明な部分はブラックボックス化されていたりするため、後工程のモデル作成に影響する事がある。
	色々なモデルはあるが、それがどういう目的のために、使われるものであり、何を表現しなくてはいけないかをモデル作成者が分からないと、モデルを作る事そのものが目的となってしまう。特に組み込みだと、様々なシステムがある為、何を表現しなくてはいけないかもシステムによって異なる。
	ツールなどで標準化しないと、作成者によって表現方法がまちまちになり、モデルの標準化による再利用性が上がらず、かつ、読み手の誤解が生じるリスクが増える。一方で、標準化すると上記の様に作り手のスキルに依存し、開発プロセスとしては一見問題無く進んでいるように見えても、成果物の内容が本来の目的を満たせていない可能性がある。
	モデリングを効率よく行うには環境（ツール、適用ルール）に加えて、ツールを使いこなせるアーキテクトの育成がセットで必要だと思う。
49	アプリケーションフレームワークを導入すると、与えられる設計に依存しすぎて、モデリングがおろそかになりがちである。
50	まずは部分的にでも使ってみることが大切。 人が作ったモデルの修正から始めると入りやすい。 習うより慣れよの精神で入っていくのがよいと思う。
51	モデリングは抽象化能力そのもの。そこに必要なものとして、抽象化能力と抽象化テクニックがあり、抽象化テクニックは教えることはできるが、抽象化能力は教えることは難しい。抽象化能力は自分でテクニックとその応用事例から学び、実践して補うしかない。が、物事の本質を掴む能力が伸びるわけではない。抽象化能力の高い人は、日々の身の回りの事柄自体も抽象化して理解、記憶している特徴があると私は考えている。
52	既存システムからの新 Platform 作成に伴い、アーキテクト検討時にモデリングを実施しました。モデリングとともにテスト用簡易 Platform を作成し、モデリングした内容から簡易プログラムを作成することによりモデリングと実装の乖離を防ぐようしましたが、効率的なモデリングについて整理できていません。

54	モデリングの手法に関する手引きは、広く公開されており、誰でも学習することができる。 但し、要求、仕様をどのように各手法を用いてモデリングを行っていくかは、経験に基づいており、誰でも行える状態にないと感じる。 あいまいさをどのように排除し、間違いを減らしていくのか、またモデリングによりそれらがどのように解決されていくのかを学習する機会があれば、モノづくりの品質を飛躍的に向上できると思われる。
55	最終的にはモデル作成→シミュレーション→コード自動生成 といった、MDD 開発の実践を行いたいと思っていますが、現状としては、モデリングによって、仕様そのものの曖昧さや仕様解釈の齟齬が軽減されることを一番に期待しています。
57	モデリングは、頭の中で要求とそれを実現するモジュールの抽象化と具現化をいったりきたりする練習とか。　ソフトウェアの動作の動的な側面（並行動作等）を両面から考える練習が必要なのかなと思います。
58	ＣＤＭを検討中です
59	モデリングのポリシー ・A3 1 枚で全体を表現する粒度とする ・「役割」が明確となるようコンポーネントへ分割する ・コンポーネント間の主要な関連(I/F)を表現する ・垂直階層構造表現にはこだわらない ・実行コンテキストを表現する ※モデリングは誰にでもできるわけではない。上級モデラーになるためには、モデリングに対するスキル（知識と経験）だけではなく、センスを要すると考える。
60	モデリングの基礎知識をつけるという目的で参加致します。
61	組込み系開発でモデリングを行う際の課題として，適切なモデリング手法見つけにくいことがあると考える．情報システム系で広く使われるモデリング手法が組込み系では使いにくいことは多々ある．近年 SysML が注目を浴びているが，現状は手探り状態で導入事例はほとんど聞かれない．
	またソフトウェア技術者にとってモデリング手法の導入は精神的障壁は低いが，エレキ設計者やメカ設計者は従来の設計手法からの脱却が難しく，この点も組込みシステムへのモデリング手法の導入の課題となっている．
	モデリングによりドキュメントベースの開発を行うことは，日本がこれまで行ってきたすり合わせ型の開発と反する可能性があるが，日本の良さを踏襲しつつも近年の開発環境の変化に順応し，日本の組込み系開発の競争力に対する強味になることを期待する．
63	モデリング対象のスコープを決めるのが難しい
	上流のモデリングする人は限定すべきだと考える
	一旦モデリングしたもののメンテナンスを維持することに課題を感じている
	モデリングしたものを有効的に活用できているか、疑問に感じている
64	「モデリング」といったときに、指すものが、ひとによってバラバラすぎ。上記のアンケートに回答するにも、微妙に言葉がちがうものの、EA/UML/SysML といった選択肢しかないように、上流設計であることには変わらないように感じられる。
	現在、弊社のモデリングはソフトウェア内部の要求→要件→仕様化→Simulink ロジック化、という流れは曲がりなりにもできていますが、ハードウェアも含めたシステムのモデリングまでは手が付いていない状態です。
	設計段階で詳細な検討がされていないため、上記のプロセスをすっ飛ばしてしまった影響は、実機検証の直前に現れます。全体を網羅した設計ができていないがゆえに、部分最適化のかたまりになってしまい、その接続にすべてのしわ寄せが行く現状があります（たいていはラピッドプロトタイピング実装で吸収する）。
	このような課題を解決するためのヒントを、当日吸収できればうれしく思います。
65	モデリングの詳細度をプロジェクトメンバで合わせるための具体的な事例があれば伺いたい。
	初めてモデリングに取り組むのに適した手法やツールがあれば伺いたい
71	モデリングを実施しましたが，その開発者のスキル，経験によって記載レベルが異なることが分かりました。 ある基準を決めて提示したとしても，類似のモジュールは書けても，応用は記載できないなど課題に直面しています。
	モデリングの粒度についても，開発者によっては詳細設計レベルのユースケースを記載するなどもあり， どのように開発者を教育していいのか課題に感じております。
	また，運用ルールが製品個々でバラバラであり，社内でもきちんとしたルールがないのが現状です。既存の製品開発を実施しつつ，モデリングなどの新しい概念を取り入れていることにも大きな課題を感じています。すんなり浸透できればよいのですが。
73	モデリング手法に期待するものとしては、漠然とした要求を分析し、それに適したソリ

	ューションの模索のための１つの分析ツールとして有効活用したいと考えています。現在の手法のトレンド、モデリングの活用手法について基本的なレベルから学びたいと考えています。本セミナーがそのヒントになることを期待しております。
74	モデリングのためにはどういうスキルが必要なのか。モデリングができる人材を育成するためにはどのような教育が必要なのか?
75	モデリングはセンスが大事。

付録 3. ワークショップ 2014 当日アンケート回答詳細

当専門委員会が実施したワークショップ 2014 での「当日アンケート調査」に対する回答を以下に示す。

【1】ワークショップ全体の開催方法について、該当するものに○印をご記入下さい

①テーマ数

多い	やや多い	丁度良い	やや少ない	少ない
0	4	52	1	0

②講演時間

長い	やや長い	丁度良い	やや短い	短い
0	0	48	8	1

【3】今回のワークショップ 2014 での講演や全員参加 106 人ワークショップで議論に参加されてみて、改めて、組込み系ソフトウェア開発において、アーキテクトをどのように育成するのかとお考えでしょうか。キーワードや短文で結構ですので３つ程度ご記入下さい。

1	・モデリング対象の明確化 ・静的と動的のそれぞれの構造を分けたモデリング ・『抽象化』の意味の正しい理解
2	・チームで共有する ・同じ方法（図）で
3	春名先生、久我さんのような話を広めると、親しみやすいセミナーが出来て、ムズカシイイメージを和らげられる。
4	モデル（モデリング）＝コミュニケーションツール
5	・社内のモデリングツール、ルールの統一 ・まずモデリングをちゃんとしよう！いう布教
6	・データをどうモデル化できるか ・機能構造と実装構造のマッピングをどうするか ・モデルを元にしたディスカッションが重要。議論すれば何かが
8	現在のＵＭＬには派生開発で主な関心事となる変更指示を表現できる View と Model がない。そのため派生開発においてモデリングの紅葉があまり出ずに活用が進まない。この点なんとかする必要がある。
9	モデルは、必要な部分のみを残した（いらないところを省いた）というところか。 改めて印象に残った。
10	・適切な抽象化（クラス、イベント、状態）モデルの作成 ・適切なふるまいモデルの作成 ・問題発見（モデリングの対象は何か、実現課題は何か？）
11	・ＳｙｓＭＬ ・要求モデリング ・非機能要件
12	・⑤にて、どの企業の方もエンジニア教育、設計指針の継承にご苦労されているのが理解できた ・モデルについても４種４様のモデルになっており、考え方の統合というのが出来ないという事が分かった。 →（先生方の絵も異なっていたため）
13	ステートマシン図が重要であることを再認識した。 静的な構造について講演内容が理解できなかった。
14	目的に合わせて適切な手法を用いてモデリングを実施する事の重要性を理解できた。 また、他社の皆さんの考え方、取り組みを知る事ができ、大変参考になりました。
15	モデル、ビュー、静的構造
16	・今現在、社内でモデルを使った開発に取り組んでいるが、モデリングと言っても色々な考えがあり、今後の活動のヒントも得られたので良かったと思います。 ・SysML に関しては取り組みが難しく、諦めていた所がありますが、再度チャレンジしていきたいと思いました。
17	ソフト技術者はプログラム作成ではなく、モデリングから学習した方が良いのではないかと思います。 モデルはシステムの構造・振る舞いを理解するには不可欠です。
18	要求のモデリングが意外と難しい。
19	・簡単なキーワードで表現することが重要 ・レビューが重要 ・論理的に説明することが重要

20	高速に動作させる方法 開発工数低減
21	フリーでモデリングを行うと、人によって注目する点が違うので、割と違ったモデルが出来上がってくるので、複数人での設計なら、事前に方針を決めないとダメかなと思いました。 逆に、視点の異なるコメントも重要で、ぜひ複数人のレビューがほしいなと思いました。
22	オブジェクトとその状態（データ）に注目する。
23	何がしたいのか（何をするのか） 何が必要なのかを掘り下げる。 静的、動的。
24	・いかなる視点のモデルかを明らかにしておくこと。 ・複数の視点でのモデルを示す事
25	・モデリングを進めるのは有効ではないかと思った。 ・もっと自分自身勉強する必要性を感じた。 ・会社でも推進したい
26	ソフトウェア設計図 コミュニケーション 完全な正解はない？
28	・自分だけがやっても客がやってないとコミュニケーションが取れない。業界で足並みをそろえることも重要。 ・自社内でもセクションによってやり方は色々。広めるのは大変。進め方は同上。 ・教育も重要
29	非機能案件、バリエーションなどをいかに考えるか？ ゴール指向分析で案件とアーキをつなげるのが１つの解かも知れない。
30	モデルを理解して使いこなせる人の育成。 組込みならではのモデリングパターンの確立
31	・普段の生活の中で、物を見た時に構成を考えてみる ・形式にとらわれない ・最低でも自分以外の人が一度は見ること
32	・抽象化 ・「制御対象」に着目
33	・モデリングの教育 ・共有（フレセレサス？エレセレサス？） ・わかりやすさ
34	派生開発、トレーサビリティ　などなど考えても、モデリングの有効性はあきらかだと思っています。
35	まだ解を持っていません。 なぜ必要かを考え中です。
36	アーキテクチャの範囲のとらえ方が難しいと思います。 Simnlink などの動作させるモデリングも取り上げてほしいです。
37	・まずは描いてみることが大切 ・得意な図で作る
38	モデルを広めるために、 　　モデルと品質向上 　　モデルと効率、生産性向上 が、分かりやすいモノが出てくると更に広がると思う。
39	・実践・実行すること ・俯瞰的に考える ・プロダクトライン
40	モデリングを実施し、自分のクセや他者のご意見が聞けたので、とても良かったです。 ビジネス戦略の視点をモデリングに組み込む方法を今後考えていきたいです。
41	・メンバのモデリング教育の推進 ・実務への効果
42	モデル検証の必要性の活がない
43	４人の中に自動車の組込みソフトをやっている人が居られたので非常に参考になりました。
44	・抽象化 ・要求モデリング→システムモデリングで保守性が高い構造にする
45	システムレベルからのモデリング モデルから実装までの一気通貫
46	まずはモデリングをする文化を醸成したいと思います。
47	システムのアーキテクチャを考えるだけでなく、ビジネス面でのモデリングが重要だと感じた。
50	モデリングを考える上で、何か抽象度を決めるコツのようなものがありそうなので、それが明確化されるとよいように思う。非機能の表現方法や制約条件（ハード性能、コスト等）折り込み表

	現する方法が知りたい。
51	初期導入が難しい→アドオン 一般化を行っていきたい
53	・汎用 ・モスク型（目的からのブレークダウン） ・要求モデリング
54	モデリングの階層を統一する 複数の視点でモデリングする 複数の人との情報共有する
55	・オブジェクト指向のイミが分かった気がする（処理が主役でなくモノが主役） ・制御の処理の視点ではなく、ドメインの動作の視点で書く
56	・静的／動的の量方モデリングが質を高める。 ・少なくとも要求→分析→本日行った部分位は生かしたい。 ・あとはモデリングを推進する気持ちが大切。
57	完全な正解はないので、とにかくトライすべき
58	まずは、モデリングの目的と、どのモデルでどこまで抽象化するか、を関係者で意識共有、合意すること
59	全体を認識しやすくするという意味で有用と思えるが、モデル図から実際のコードにまで、誰が見てもつながる様になるのかどうかは疑問
【4】今回のワークショップ 2014 に参加されたことによって、何か気づかれたことやお考えが変わったことがありましたらご記入願います。	
1	今まで実装に向けて実現性しか考慮していなかったが、ビジネス性を含め付加価値を高めることに気づけた。
2	同じモデリングというテーマで、訂正しても個人で色々な表現方法があるので、チームで統一することが重要だと再認識した。
4	若い方の課題認識パターンがある程度理解できた気がする
6	・構造図の中にどう責務を表現するか。構造代設計手法だと書き易いかと。 ・その場合、手続きに引きずられるのをどう防ぐか ・データをどう考えるかが課題
9	他の企業の方と交流できてよかった
10	サービスでビジネス（石田氏）の視点が忘れがちなので、教訓になりました。
11	ＳｙｓＭＬとＵＭＬの違い
12	課題が共有できたこと、どこも同じ問題で苦労されているのが理解できたが、特に新たな技術を得られたという訳ではなかったのが残念
13	モデリングを必須としている所が多い
14	モデリングに本格的に取り組まれている方々が予想以上に居られたことが発見でした。 これを機に勉強し、取り組んでいきたいと思う。
16	気づいた事としては、社内にモデリングを導入することに、皆さん、苦労されていて、自分達だけではないことが分かりました。今後、モデリングが社内に浸透するように努力していきたいと思います。
17	ＵＭＬ　以外のモデリング方法があることを知った。
18	より広い分野にモデリングを摘要してみたいと思った。
19	簡単な事でもｆｕｎｃｔｉｏｎを定義することが重要。特に日本人の若手には。
21	要求モデリングをいざやるぞ、と思っても、なかなか手が進まないことに気づきました。 トレーニング不足・・・。
22	モデルをコミュニケーションに使う重要性
23	モデリングの考え方
24	分の弱点、視野の狭さに気付いた。
25	みなさんの状態が分かって、良かった。
26	みなさん、モデリングに苦労されている様子で、業界全体で取り組むべきと感じた。
27	あらためて、モデリングは柔軟に、かつシンプルに（素直に）行うべきだと感じた。
28	ワークショップで他社の内情が分かってよかった
29	他社においても同じ悩みがある。 日本の受発注文化なども影響があるとディスカッションから知れた。
30	皆さん、同じようなことに苦労している
31	他社の人と話すのは重要
32	人によって抽象化レベルが異なるので、レベルをどこにもって行くのか考える必要がある。
33	・ソフトウェア設計だけではなく、上流のモデリングも重要 ・ビジネスモデルとソフトウェアモデルのシームレスな連携
34	モデリングの答えは１つではないということは大きな気づきでした。
35	社外の方と交流して非常に刺激になりました。

37	日頃やっていることを少し改善すれば良いということが分かった
38	マネージャーも「モデ力」を上げなければならない
39	小さい会社なので、自分だけ分かれば良いかと思っていましたが、設計資料としてモデリング図等を残した方が良いと思いました。
40	性能という観点で動的構造を考えられるようなモデルがあれば、より組込み業界に受け入れられると気づきました。
41	同じ問題を各社かかえている
43	去２～３回の出席に比べて、若い方、プログラマーの方の感じの人の出席が増え、質問も実践のものと思われる。これは大いに結構なこと。
44	要求モデリングを実施していないので、これからは実施したいと思った。
46	参加者の方とモデリングの結果をもとにディスカッションできて楽しかったです。 来年も期待しています。
47	モデリングの目的などを理解していなかったので、本日の講演は非常に参考になった。
49	ディスカッションにおいて、他社の取組みが聞けた事が参考になった。
50	モデリングする場合、静的表現と動的表現を分けて考える必要があること。汎化しすぎずにドメインの機能（要求）を表現しなければいけないことに気づけた。
51	どこもモデリングの大切さは分かっていても、実現は難しいんだと思いました。
53	今はスタッフ部門なので、久しぶりにモデリングをしてみて大事さがわかった
54	モデリングに対する重要性を認識できた
55	モデリングというと従来はSimnｌｉｎｋによるコントローラモデルやＭｏｄｅｌｉcaのようなプラントモデリングしか浮かばなかったが、システムを記述するモデリングの必要性が理解できた。
56	106人ワークショップでは他社の方と接する事ができ、体験談をほんの多少聞く事が出来たのが刺激になった。普段は、自社内部の人間としか話さない自分にとっては有意義な時間でした。
57	最初から完全を求めすぎていたように思う。試行錯誤すれば、案外うまくいくのでは、と感じた。
58	抽象化（どこまで抽象化するか）は難しい 抽象化スキル＝設計スキル
59	
【5】その他、ワークショップに関するご意見、ご感想、ご要望事項などがありましたらご記入下さい。	
1	特になし
2	
3	
4	おもしろい試みだと思う。
5	
6	
7	
8	
9	
10	楽しいＷＳなので、今後も継続でお願いします。
11	このセミナー形式を毎年やり続けてください。
12	モデリングの成功事例、失敗事例を集めて頂きたい（発表して頂きたい）
13	開発の上での必須のモデリング対象について講演して欲しい
14	今年を１年目として、来年以降のモデリングの本セミナーにも参加したいと思います。
15	大変、勉強になりました。
16	社外の方とモデルに関して話合うことがなかったので、とても良い機会を与えてくれたことに感謝いたします。今後も参加したいと思いますので、よろしくお願いいたします。
17	モデリングのトレーニングをするに、どうすべいか教授してほしい。ＪＥＩＴＡでトレーニング教材を開 発して欲しい。
18	
19	ＷＳが活発で参考になった。大阪でも開催できないか、ご相談させてください。
20	
21	
22	次回もぜひ参加したい
23	
24	
25	楽しいワークショップで、次回も是非参加したい。
26	
27	
28	大変ためになった。今後も参加したい。
29	海外のトレンドと日本とのギャップ、その問題など。
30	

31	おつかれさまでした。
32	
33	
34	ワークショップの正答例？なども配布いただければと思います。
35	次のテーマが何になるのか楽しみです。
36	
37	講演の時間を長くしてほしい
38	
39	
40	
41	
42	電機系だけでなく、自動車、ロボット系も入れるとよい
43	１．我が国産業の競争力はソフトウェア作成が関ると思い、過去何回か出させてもらっています。 ２．帰するところ、日本人の抽象化能力がかかってくると思いますが、プログラム作成に直接かかわらない小生としては、今日のテーマは難しかったです。専門家の皆様の議論のように感じました。 ３．毎回感じますが、特に久我氏が機械設計部隊と機械設計の修正まで議論されたのかどうかを聞きたいところです。ソフトウェア部隊の中の議論では次の飛躍はないと思います。 ４．異業種の組み込みソフトの経験者を講師に呼ばれたらどうでしょうか。
44	色々な人の意見を聞く事ができて参考になった。
45	
46	運営事務局の方、準備、運営ありがとうございました。
47	
48	
49	
50	講師の方も参加するワークショップは非常に参考になりました。
51	少し寒いです。 ③のお話、もっと詳しく聞きたかったです。
52	
53	
54	
55	・演習も含む内容を無料で開催いただけるなんてすばらしいです。 ・ＵＭＬのセミナーというと、ツールの使い方のセミナーが大勢である中、ＵＭＬ的な考え方が重要、という当方の問題意識にマッチした大変意義のあるものでした。ありがとうございました。
56	
57	
58	
59	全体を通して非常に参考になるものでした。ありがとうございました。

報告 1. 組込み系ソフトウェア・ワークショップ 2014

ソフトウェア事業委員会/ソフトウェア事業基盤専門委員会主催で、「組込み系ソフトウェア・ワークショップ 2014」を 2014 年 11 月 7 日に、大手センタービルで開催した[1)]。

1. ワークショップ 2014 の目的・概要

JEITA 組込み系ソフトウェア開発に関するワークショップも好評のうちに今年で 8 年目を迎えた。日本の組込み系ソフトウェア開発の強みは、擦り合わせによる高品質開発にあると言われているが、大規模化・複雑化・短納期化・多機種化の潮流の中で、国際競争力を今後も維持していくことが難しくなってきている。

このような状況への対処として、私たちは品質の強化、開発スピードアップ、アーキテクトの育成というテーマで、今までワークショップを開催してきた。その中でアーキテクチャ設計の重要性を議論し、発信してきた。

そこで、今回のワークショップでは、アーキテクチャ設計でも重要であり、組込み系ソフトウェア開発のキーにもなる「モデリング」をテーマにワークショップを開催することになった。モデリングをテーマにした最初のワークショップとして、モデリングとはどんなものなのかから始め、何を何の目的で、どのようにモデリングしていくべきなのか、モデリングはなぜ失敗するのかなどを議論した。

このワークショップでは、大阪大学の春名修介教授による、これまでのモデリング経験を基にした基調講演を皮切りに、富士通の奥村一幸氏と石田晴幸氏、アマダの久我雅人氏によるモデリングの事例講演をした。最後に“全員参加 106 人ワークショップ：徹底議論！モデリングの成功のヒントを探る”と銘打った討論セッションで、参加者全員の方にモデリングを作成して、それを元に自由に意見や質問をして、本音の議論をしてきた。

90 名に近い出席者全員が参加して一つのテーマを議論する今年の「全員参加 106 人ワークショップ」では、参加者がモデリングを実際に行い、それを元に司会からの設問に対して全員が自分の意見を 3 色の色紙を挙げて表明し、近くの机に居合わせた人達がチームとなって一斉に会場全体で議論した。この様子は、今年も参加者の意欲と意識の高さを感じるワークショップそのものであった。当日は、90 名近い方々の参加となり、組込みソフトウェア関係者の意見交換・情報交換の場として継続していく重要性を感じつつ、今回のワークショップを終えた。

[1)]組込み系ソフトウェア・ワークショップ 2014：資料室 1
http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=743&ca=1

2. プログラム

ワークショップのプログラムを以下に示す。

時間	テーマおよび講師
13：00 ～ 13：15	開会の挨拶： ソフトウェア事業基盤専門委員会の活動について ワークショップ 2014 の狙いと課題認識 JEITA ソフトウェア事業基盤専門委員会 委員長 沖電気工業（株） 五味 弘 氏
13：15 ～ 14：15	基調講演： 実践的モデリング論～難しいことを考えずにモデリングを実践するには～ 大阪大学 大学院 情報科学研究科 春名 修介 教授
14：15 ～ 14：45	事例講演 1： Excel と EA を駆使したモデル設計現場適用例と MBSE 展開に向けての課題 PDF-- Excel と EA を駆使したモデル設計現場的用例 PDF -- 富士通株式会社共通開発本部 ネットワーク管理ソフトウェア開発統括部 プロフェッショナルエンジニア 奥村 一幸氏 氏 株式会社富士通コンピュータテクノロジーズ 組込みシステム技術統括部 アーキテクチャ部 アーキテクト 石田 晴幸氏
14：55 ～ 15：25	事例講演 2： 組込ソフトの開発現場におけるモデリング事例 株式会社アマダ 板金機械開発本部 ソフト開発部門 第一ソフト開発部 久我 雅人氏

15：25 ～ 17：30	全員参加 106 人ワークショップ： 徹底議論！モデリングの成功のヒントを探る 司会： ビースラッシュ(株)　代表取締役 山田　大介 氏 アドバイザ： 大阪大学　春名　修介　教授 富士通株式会社 奥村　一幸 氏 株式会社富士通コンピュータテクノロジーズ 石田　晴幸 氏 株式会社アマダ　久我　雅人 氏 NPO 法人組込みソフトウェア管理者・技術者育成研究会 理事　渡辺 登 氏

＜総合司会：ソフトウェア事業基盤専門委員会委員　富士通株式会社 竹田 和正 氏＞

3. 講演の概要

3.1 1. 開会の挨拶：

ソフトウェア事業基盤専門委員会の活動について､ワークショップ 2014 の狙いと課題認識

ソフトウェア事業基盤専門員会の活動の出発点となった課題認識は、『擦り合わせなるものが日本の開発力の強みと言われているが、急激に増大している開発規模や短納期化の中で、「擦り合わせ」が強みになっているのであろうか』であった。

この課題認識に基づいた 2005 年度から 2011 年度までの活動内容の要点と今年度(2014 年度)の活動について報告した。

2005 年度からの 3 年間は「品質」をメインテーマに、 2008 年度からの 3 年間は「開発スピードアップ」をテーマにしてきた。 その活動を通して重要であったアーキテクトについて 2011 年度から 3 年間活動をしてきた。

2013 年度までの活動での課題やそれを解決するための提言では､「アーキテクチャ設計」が弱く、さらにそのための「アーキテクト」が必要であることをまとめてきた。

そこで2014年度から始まる3年間の活動では､このアーキテクチャ設計でも重要になる｢モデリング」をテーマにしている。 このテーマで 2014 年度から活動を開始し、最初の年度では、モデリングに関するワークショップを初めて開催した。 サブテーマを「モデリングはなぜ失敗するのか？」と題して、モデリングの現状や問題点、そしてその解決法を探ることにした。

以上からこのワークショップをスタートし、開発者視点に立脚して、組込み系ソフトウェア開発におけるモデリングについて、参加された方々との議論の中から明らかにしたい旨が当専門委員会委員長 五味氏より紹介された。

3.2 基調講演：実践的モデリング論～難しいことを考えずにモデリングを実践するには～

(1) モデルとは
　モデルの例

(2) アーキテクチャとは -目的の一例として-
　ソフトウェアアーキテクチャの定義
　ソフトウェアアーキテクチャの私なりの解釈

(3) 組込みソフトウェアアーキテクチャ設計におけるモデル
　視点(View)とは
　アーキテクチャ設計の目的
　事例 1: アーキテクト A 氏の憂鬱
　組込みソフトウェアの特徴：釈迦に説法ですが
　構造設計の 3 つの視点（View）

(4) 表記法の事例 - 対象・目的に合った表現とは
　代表的な構造の表記方法
　C&C モデルの現実的な意味
　その他の構造表記方法
　事例 2：蜘蛛の巣クラス図
　アーキテクチャ記述としての UML 見直しの動き
　一番のポイント：詳細化構造の表現
　表記法の候補,選択した表記法,視点（View）間の整合
　事例 3：本人にしか分からないモデル
　事例 4：放射線状のクラス図
　事例 5：MDD に向けての注意

組込みソフトウェアの基本構造
(5) まとめ：モデリング成功のポイント

3.3 事例講演 : Excel と EA を駆使したモデル設計現場適用例と MBSE 展開に向けての課題
(1) モデル設計に踏み切れない三大要因
(2) プロジェクト実践例紹介
大規模ソフト開発の悩み
EA 取り込みまでの作業フロー
(3) MBSE 適用に向けての課題
草の根活動の紹介
競合試験の環境構築
性能等の見積もり精度
設計から適切な試験の抽出
活動ロードマップ
(4) モデル設計現場の課題
テーブルアクセスにおける機能間の競合
EA のデータ構造
施策: Excel の ER 図と Activity 図を読んで EA に入力するツール
施策: EA モデルから機能・テーブルマトリックスを出力するツール
評価

3.4 事例講演：組込ソフトの開発現場におけるモデリング事例
(1) 私とモデル化技術のかかわり
T 社時代の開発のやり方
ソフトウェア設計図との出会い
R 社でモデル化技術と出会う
モデル化と反復開発の導入
モデル化技術導入の背景
反復開発が時代の要請だった
モデリングの工程
【事例 1】 カラーMFP の制御ソフトの全体像
オブジェクト指向によるモデル化でどこが改善されたか
【事例 2】 カラーMFP の紙搬送制御部分のモデル
モデリングは誤解されている
【事例 3】 カラーMFP の紙搬送制御部分の固定変動設計とソフト部品化
【事例 4】 固定変動技術を利用した 大規模再利用の基本的な考え方
モデル化のレベルと開発者の組織
【事例 5】ソフトウェアプロダクトライン開発の導入
モデル化によるチームの変化
異質なモデル化に遭遇したことも
(2) モデル化･･･新たな挑戦
【事例 1】板金曲げマシンのコントローラソフトのモデル化
【事例 1】モデル化と反復開発の適用効果
【事例 2】 自動倉庫システムの課題
【事例 2】 自動倉庫システムのモデル化、課題解決
モデル化を契機にチームが成長

ソフトウェア資産を活用した戦略的な開発へ

4. 全員参加 106 人ワークショップ概要

全員参加 106 人ワークショップ「徹底議論！モデリングの成功のヒントを探る」の概要を示す。このテーマを中心に司会の山田氏により、講師陣をアドバイザとして参加し、またアドバイザの渡辺氏を加えて、様々な視点から議論が展開され、当日は 90 名近い参加者で活発な議論が行われた。

このワークショップでは、参加者全員に 3 色のカードが配られ、それを持って挙手することで意思表示を行った。カード提示で表明された意思や、参加者やアドバイザの意見がリアルタイムにスクリーンに表示した。

またワークショップ参加申し込み時に、モデリングの現状や課題についての事前アンケートを実施し、80 名近くのアンケートの回答が得られた。ワークショップでは、これらを元に、モデリングについての議論が進められた。

用意していたモデリング問題「スマートガレージ」について、参加者が要求モデリングとアーキテクチャモデリングを行い、このモデリング結果を元に議論を行った。また参加者の事前アンケートの回答結果を元にした質問や、参加者からその場で表明された項目について司会の山田氏から質問がなされた。参加者は、カードで意思表示をするときに、近くの参加者達で意見を交換することにより、より鮮明な意思表示が行われた。その結果のいくつかを以下に紹介する。

(1) 最初に、ワークショップ参加者に事前に用意した「スマートガレージシステム」について、要求モデリングを短時間で作成した。これを元に参加者が 4 名から 6 名ほどのグループになり、議論をした。その後、この要求モデリングを元に全体で議論を行った。

(2) アドバイザもモデリングを実施し、発表をし、会場と色々とディスカッションを行った。

(3) この議論のときに、参加者はカードで意思表示をして、全体の状況を調べた。以下にその概要を示す。

(a) どのようなベースで記述したのか？（何で書いた？）

自社標準・・・8 名

講演資料ベース・・・8 名

適当・・・その他

(b) どのようなモデル図を描いたのか？

ユースケース、ブロック図、機能ツリー

ステートマシン図、システムコンテキスト図

要求図

形式手法、USDM

(c) ビジネス戦略と制約のつなぎ方は？

つながった・・・3 割（目的から機能ツリー抽出、要求図から作成、SysML）

つながらなかった・・・7 割

(d) 何を売るつもりなのか？

サービス・・・1 割

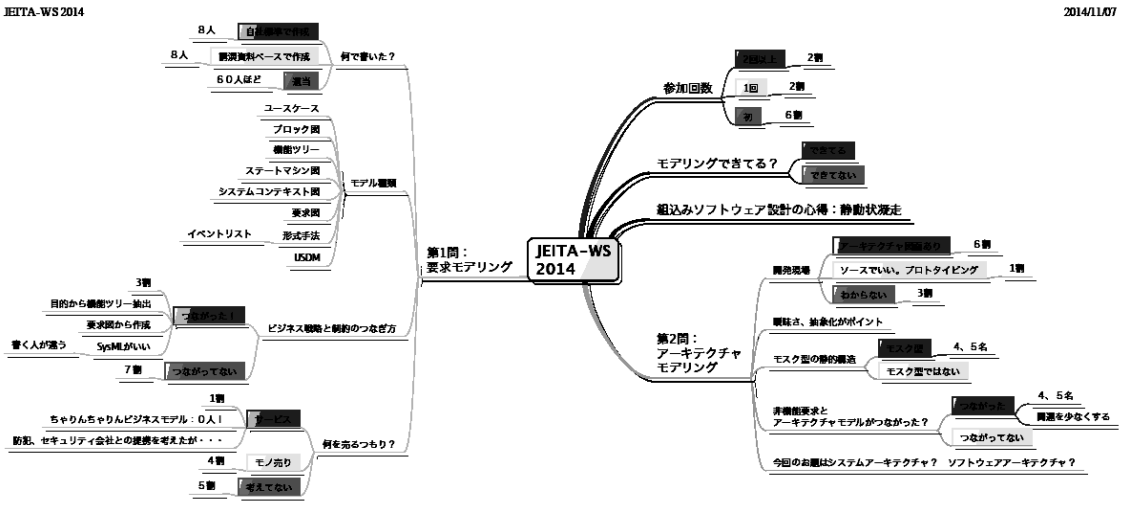
モノ売り・・・4 割

考えていない・・・5 割

(4) 次に、「スマートガレージシステム」のアーキテクチャモデリングを作成してもらい、これを元にグループでさらに議論をした。その後、このアーキテクチャモデリングを元に全体で議論を行ったので、以下にその概要を示す。

(a) 開発現場でアーキテクチャモデリングは？
　アーキテクチャ図面あり・・・6 割
　ソース、プロトタイピング・・・1 割
　分からない・・・3 割
(b) モスク型の静的構造になったか？
　モスク型・・・4,5 名
　モスク型でない・・・その他
(c) 非機能要求とアーキテクチャモデルがつながったか？
　つながった・・・4,5 名（関連を少なくする）
　つながっていない・・・その他
(5)このようにしてワークショップは、時間一杯まで、活発な意見交換が行われた。

カードで意思表明



議論の概要

報告 2. CEATEC JAPAN 2014 講演

ソフトウェア事業委員会/ソフトウェア事業基盤専門委員会は CEATEC JAPAN 2014 のインダストリアルシステムトラックで活動報告の講演を 2014 年 10 月 8 日(水)に行った。

1. 講演の目的・概要

ソフトウェア事業委員会では、ソフトウェア産業の地位向上と国際競争力強化に向けた施策提言、ソフトウェア開発の基盤強化へ向けた取組み、ソフトウェアリソースに関する調査分析など、ソフトウェア産業共通課題の解決に向けた調査分析・提言の活動を行っている。

当事業委員会の下に設置されているソフトウェア事業基盤専門委員会では、我が国の強みの源泉であり価値創出のキーであると言われている「組込み系ソフトウェア」に焦点を当て、この分野での開発力向上等の基盤強化のための活動に取り組んでいる。

組込みソフトウェア開発では大規模化や多機種化の流れの中で、これまで以上にアーキテクトの重要性が増している。JEITA ソフトウェア事業基盤専門委員会では、アーキテクト育成の課題をアーキテクトヒアリングや委員による討論、ワークショップでの討論を通じて、分析してきた。これらの分析結果に基づき、日本の組込み開発現場を成功に導くアーキテクトの育成やその課題についてまとめた。

今回の CEATEC JAPAN 2014 での講演は、昨年度までの結果を中心に、アーキテクトの定義や配置、事例、育成などについて報告した。

2. 講演テーマ・講演者

講演テーマ：
組込み系ソフトウェア開発の課題分析と提言～アーキテクトをどのように育てるのか～（JEITA 調査結果報告）

講演者：
五味 弘氏（OKI）

講演概要：
組込みソフトウェア開発では大規模化や多機種化の流れの中で、これまで以上にアーキテクトの重要性が増している。JEITA ソフトウェア事業基盤専門委員会では、アーキテクト育成の課題をアーキテクトヒアリングや委員による討論、ワークショップでの討論を通じて、分析してきた。これらの分析結果に基づき、日本の組込み開発現場を成功に導くアーキテクトの育成やその課題についてまとめたので報告する。

報告 3. モデリング・ブレーンストーミング

有識者を招いてのモデリングに関するブレーンストーミングを以下の要領で行った。

(1) 第 1 回モデリング・ブレーンストーミング
日時: 2015 年 1 月 9 日 13:30-17:00
参加有識者: 春名修介教授（大阪大学）
　春名教授と当委員会の委員やオブザーバが一緒にモデリングに関する事例をベースに失敗要因などをブレーンストーミングした。この結果はマインドマップなどにまとめ、本報告書本文で紹介している。

(2) 第 2 回モデリング・ブレーンストーミング
日時: 2015 年 1 月 28 日 13:30-15:30
参加有識者: 平鍋健児氏（チェンジビジョン）
　平鍋氏を招いて当委員会の委員やオブザーバでモデリングに関する事例をブレーンストーミングした。平鍋氏による事例紹介やモデリングのコツなどを中心に議論を行った。この結果は本報告書本文で紹介している。
　平鍋氏より紹介のあった資料を以下に掲載する。
(i)「アジャイル時代のモデリング」KEEPS/TEMPS
http://www.infoq.com/jp/articles/kenji-modeling-agile
https://www.slideshare.net/hiranabe/modeling-in-the-agile-age-jp
(ii) これだけモデリング
http://qiita.com/kenjihiranabe/items/3006a43c58b0bdd1a38a
http://www.slideshare.net/ChangeVision/koredake-modeling
(iii) GSN
http://blogs.itmedia.co.jp/hiranabe/2013/11/goal-structuring-network.html
http://www.slideshare.net/hiranabe/omg-astah-gsn-demonstration
(iv) 認証工学
http://www.sice.or.jp/org/ce-wg/
(v) SCDL(Safety Concept Description Language)
http://scn-sg.com/
(vi) SafeML
http://swest.toppers.jp/SWEST16/data/s2c_proceeding.pdf
(vii) DEOS/D-Case
http://deos.or.jp/technology/dcase-j.html
(viii) エンタープライズアジャイル
http://www.slideshare.net/hiranabe/what-is-enterprise-agile

講演資料

1. CEATEC Japan 2014 講演資料
2. ワークショップ講演資料 1（五味氏）
3. ワークショップ講演資料 2（春名教授）
4. ワークショップ講演資料 3（奥村氏）
5. ワークショップ講演資料 4（石田氏）
6. ワークショップ講演資料 5（久我氏）
7. ワークショップ公開問題（スマートガレージ）

JEITA調査結果報告

組込みソフトウェア開発の課題分析と提言

～アーキテクトをどのように育てるのか～

2014年10月8日

一般社団法人 電子情報技術産業協会
ソフトウェア事業委員会
ソフトウェア事業基盤専門委員会

委員長 五味 弘(OKI)

目次

1. JEITA ソフトウェア事業基盤専門委員会の紹介
2. 背景 － 日本の組込みソフトウェア
3. アーキテクトの必要性とその役割、スキル
4. アーキテクトの育成
5. アーキテクトヒアリング
6. アーキテクトワークショップから
7. まとめ

予告「モデリングワークショップ2014(11/7)」
参考文献

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

1. JEITA ソフトウェア事業基盤専門委員会の紹介

＜JEITAとは＞ HPより抜粋

一般社団法人 電子情報技術産業協会(JEITA: Japan Electronics and Information Technology Industries Association)は、電子機器、電子部品の健全な生産、貿易及び消費の増進を図ることにより、電子情報技術産業の総合的な発展に資し、我が国経済の発展と文化の興隆に寄与することを目的とした業界団体です。

（中略）

JEITAは、まさに21世紀のデジタル・ネットワーク時代を切り拓いていくことを使命としており、電子情報技術の発展によって、人々が夢を実現し、豊かな生活を享受できるようになることを願っています。

このため、政策提言や技術開発の支援、新分野の製品普及等の各種事業を精力的に展開するとともに、地球温暖化防止等の環境対策にも積極的に取り組んでいます。

395社・団体が参加中

- ソフトウェア事業委員会
  - ソフトウェア事業戦略専門委員会
  - ソフトウェア事業基盤専門委員会
  - スマート社会ソフトウェア専門委員会

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ソフトウェア事業基盤専門委員会の活動概要

- 目的:「組込み系ソフトウェア分野」でのソフトウェア開発力の現状把握および基盤強化を図るための取組み
- 活動概要
  1. 毎年のテーマに沿った委員による議論(月1回～2回のペース)
  2. テーマに応じたアンケート調査(JEITA参加企業を中心に実施)
  3. ワークショップの開催(毎年1回)
  4. CEATECでの講演
  5. 関係団体との交流、意見交換
     (財団法人にいがた産業創造機構(NICO)、組込みソフトウェア管理者・技術者育成研究会(SESSAME)、組込みシステム産業振興機構(ESIP)、組込みシステム技術協会(JASA)、 ほか)
  6. 各種調査(海外調査も含む)
  7. 「ソフトウェアに関する調査報告書」の発行(年1回)

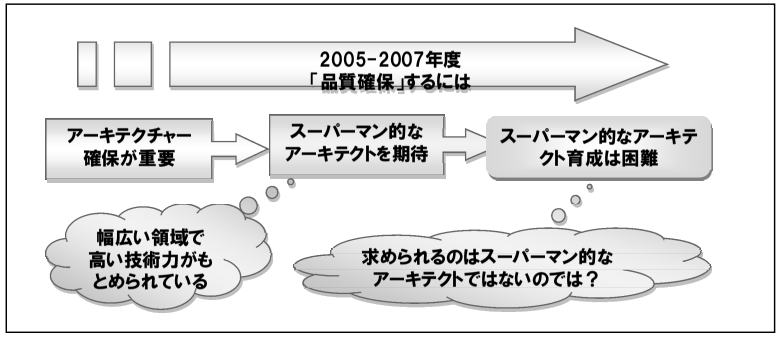
本専門委員会参加企業(2014年度)

沖電気工業、東芝、東芝ソリューション、日本電気、日立製作所、富士ゼロックス、富士通、三菱電機

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

3. アーキテクトの必要性とその役割、スキル
過去の活動からのアーキテクトの必要性

■ 品質確保の活動からアーキテクトに関する課題を抽出



集まった意見の例
- ・アーキテクトは座学など学校教育での育成は困難
- ・一企業内の活動での育成は難しい
- ・業界を通じてアーキテクトを育成するような取組みが必要
- ・産官学一体となっての活動が望まれる
- ・変動性を実現していける技術を持つアーキテクトの育成が必要。
- ・経営や戦略の視点を持つアーキテクトの育成が必要

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

2. 背景 － 日本の組込みソフトウェア
組込みソフトウェア開発に関する問題意識

■ 「組込みソフトウェアは日本の強みの源泉であり価値創出のキー」と言われているが、
組込み対象となるハードウェア機器は強いとしても、
ソフトウェア開発力が国際的に見ても本当に強いのであろうか?

■ 「擦り合わせ」の開発方法が日本の強みと言われているが、
急激に増大している開発規模や短納期化、複雑化、並行開発の中で、
現在でも「擦り合わせ」が強みになっているのであろうか?

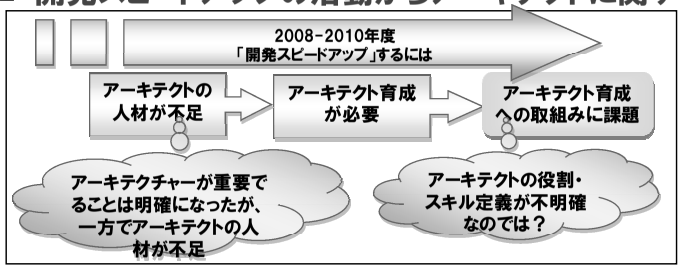
■ 何を強くすれば、
日本の組込みソフトウェア開発の国際競争力を強化し、
真に「日本の強みの源泉」たりうるものにできるのであろうか?

■ 組込みソフトウェア開発を取り巻く状況：4つの大きな波を迎え撃つには
- 大規模化
- 短納期化
- 複雑化
- 複数機種並行開発

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

過去の活動からのアーキテクトの必要性（続き）

■ 開発スピードアップの活動からアーキテクトに関する課題を抽出



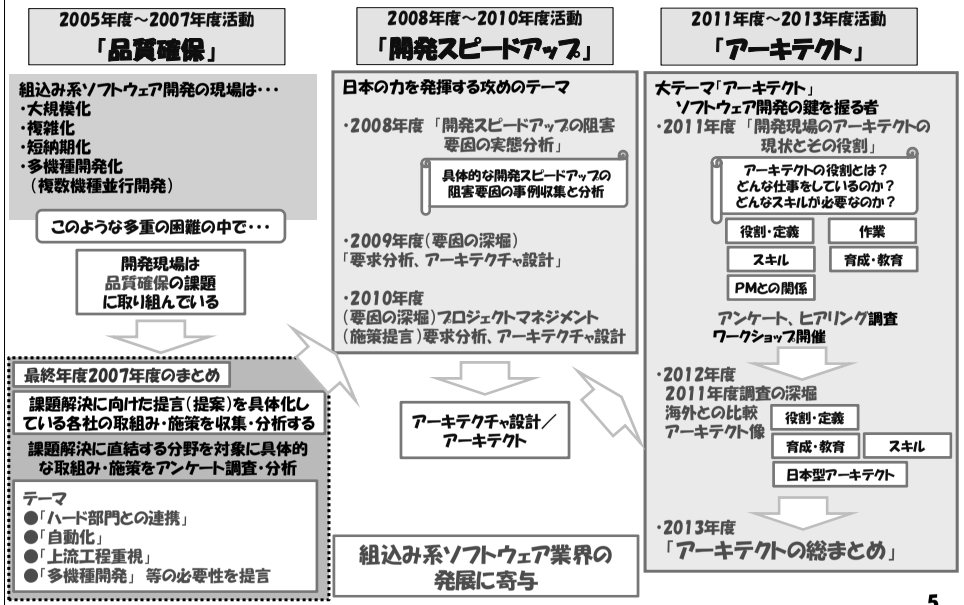
集まった意見の例
- ・「アーキテクトが不足している」という意見は毎年の調査で報告される
- ・アーキテクチャー設計の位置付けとタスクを明確化
- ・アーキテクトのスキル不足により、アーキテクチャーの問題が発生

■ アーキテクトの育成とその定義の必要性

- □ 育成には、アーキテクトの役割を明確する必要性
- □ 必要なスキル、必要な環境（権限、責任、処遇）
- □ アーキテクトの定義
- □ アーキテクト像の確立へ

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

問題解決に向けての JEITA 委員会活動

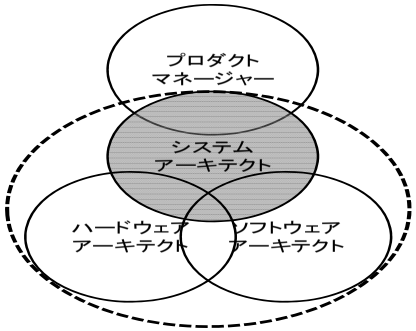


JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

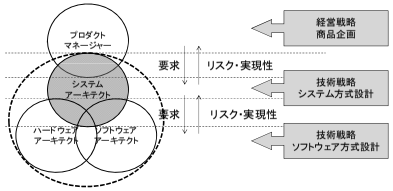
アーキテクトに必要なスキルと知識

1. アーキテクトに必要なスキルと知識（2012年度まとめ）
   1. アーキテクチャー設計プロセスと必要なスキル
   2. アーキテクトの活動
      1. 活動領域
      2. 活動範囲
      3. アクティビティ
      4. ステークホルダー
      5. 配置
   3. 2011年度アーキテクトアンケートの考察
2. アーキテクトの育成（2012年度まとめ）
3. 海外のアーキテクトの育成（2012年度まとめ）

アーキテクトの活動領域 –定義



組込みアーキテクトの範囲

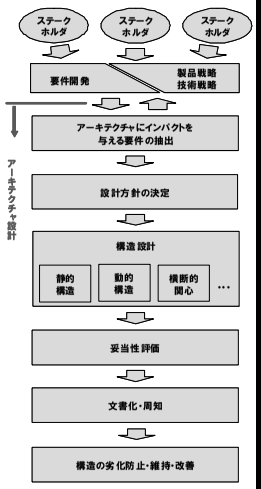


組込みアーキテクトの範囲

開発に関わる職種	役割
プロダクトマネージャー	経営的観点のもとに、製品の企画・開発・製造・保守などにわたる製品ライフサイクルを統括する責任者。
システムアーキテクト	システムの利用・開発等の要求を満すようにソフトウェア・ハードウェアを含めたシステム構造ならびに開発プロセスを設計する技術者。
ハードウェアアーキテクト	システムアーキテクトがハードウェアに割り振った要件を解決する実現方法を具体化して技術リスクを抽出しシステムアーキテクトやソフトウェアアーキテクトと調整を図る技術者。
ソフトウェアアーキテクト	システムアーキテクトがソフトウェアに割り振った要件を解決する実現方法を具体化して技術リスクを抽出しシステムアーキテクトやハードウェアアーキテクトと調整を図る技術者
プロジェクトマネージャー	製品開発プロジェクトの構築ならびに遂行にあたり、プロジェクトを計画・指揮・監督する責任者。
ソフトウェアエンジニア	ソフトウェアの各開発工程において開発・実装・テスト作業を担当する技術者。
テストエンジニア	テスト設計、テスト実行等のテスト作業の実施を担当する技術者。
QAスペシャリスト	プロジェクトの全工程において品質確保・維持・向上の推進を担当する専門技術者。

アーキテクチャー設計プロセスと必要なスキル

アーキテクチャー設計プロセス



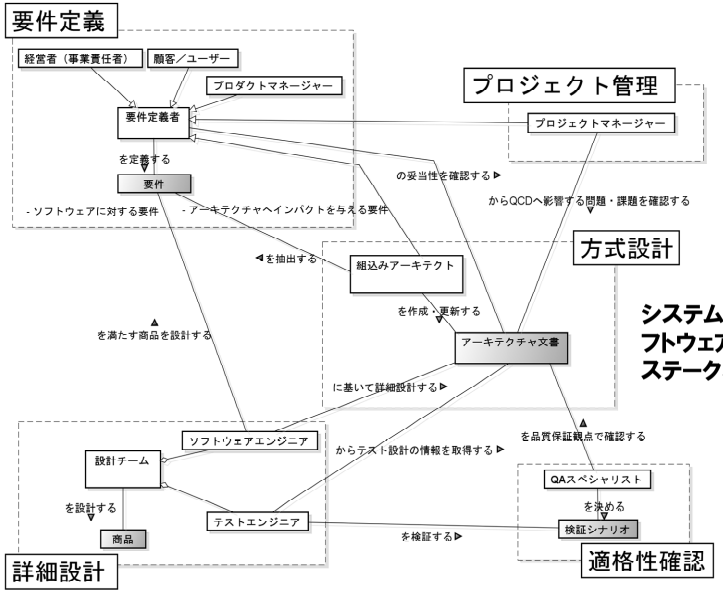
アーキテクチャー設計に必要なスキル例

アーキテクチャー設計の各アクティビティ		知識・スキルの例
アーキテクチャーにインパクトを与える要件		・要求モデリング、ビジネスモデリング ・問題解決、視野の広さ（素養・経験）
設計方針の策定		・プラットフォーム設計、アーキテクチャパターン／フレームワーク、スコーピング ・過去の設計事例（素養・経験）
構造設計	静的構造	・構造化設計 ・オブジェクト指向設計、コンポーネント指向設計、UML,SysML ・モデル駆動開発手法、モデルベース開発手法　など
	動的構造	・リアルタイムシステム設計（ハード/ソフトリアルタイム） ・リアクティブシステム設計 ・状態遷移設計 ・イベント駆動、時間駆動システム設計　など
	横断的関心	・セキュリティ設計、低消費電力設計
妥当性評価		・トレーサビリティ、アーキテクチャー評価 ・プロトタイピング、シミュレーション ・インタビュー／レビュー技法
文書化・周知		・アーキテクチャドキュメンテーション　など
構造の劣化防止・維持・改善		・構造解析、メトリクス分析　・リスク管理 ・技術・製品・システムの進化予測　・リーダーシップ

アーキテクトの活動領域 –主業務と関連業務

開発に関わる職種	経営戦略	技術戦略	商品企画	システム方式設計	SW方式設計	SW実装	SW検証評価	システム検証評価
プロダクトマネージャー	◎	○	◎	○	–	–	–	–
システムアーキテクト	○	◎	○	◎	–	–	–	○
ハードウェアアーキテクト	–	◎	–	○				○
ソフトウェアアーキテクト	–	◎	–	○	◎	○	○	○
プロジェクトマネージャー	–	–	○	○	○	○	○	○
ソフトウェアエンジニア	–	–	–	–	○	◎	◎	○
テストエンジニア	–	–	–	○	○	–	◎	◎
QAスペシャリスト	–	–	–	○	○	○	○	○

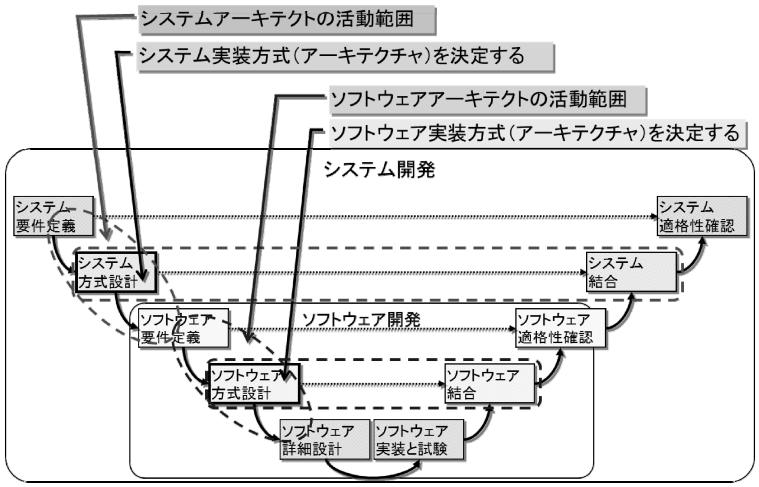
アーキテクトの活動 –組込みアーキテクトのステークホルダー



システムアーキテクトとソフトウェアアーキテクトのステークホルダーは後述

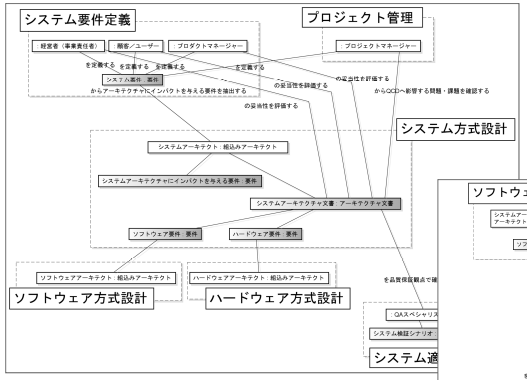
JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトの活動範囲

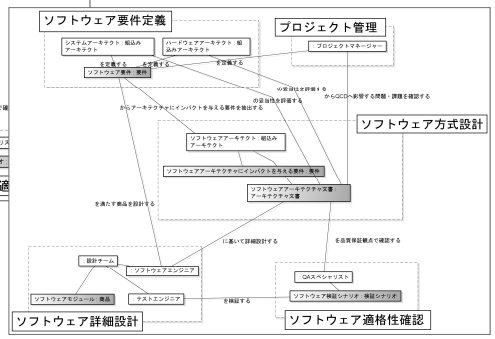


JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトの活動 –各アーキテクトのステークホルダー



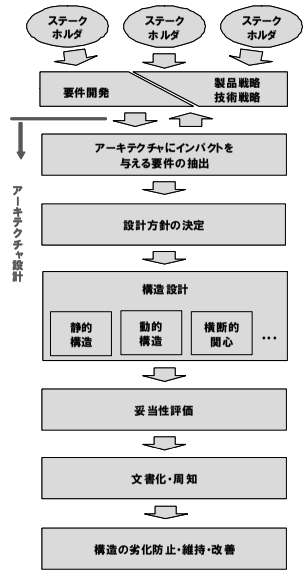
システムアーキテクトのステークホルダー



ソフトウェアアーキテクトのステークホルダー

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトの活動 –アクティビティ

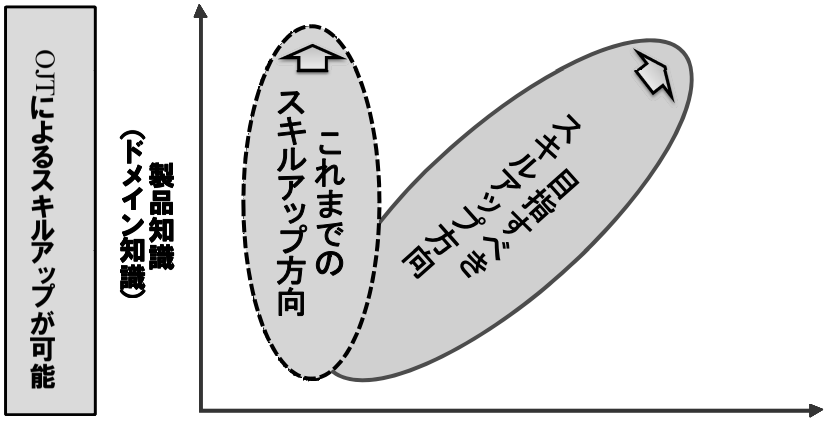


	アクティビティ	作業内容
	アーキテクチャーにインパクトを与える要件の抽出	設計のインプットを的確に把握し、重要な設計項目を漏れなく設計に反映する。設計の手戻りを極小化する。
	設計方針の策定	システム全体の構造を設計するための設計方針を決める。
	構造設計	システム全体の構造を設計する。 構造の設計は、複数の視点から行われる。機能（責務）分割の視点、動作の視点、横断的関心、データの視点
	妥当性評価	設計された構造が要求に合致しているか、適切な設計になっているかを検証する。
	文書化・周知	設計した構造を様々なステークホルダーに周知する。アーキテクチャドキュメンテーションを通じて、設計内容を的確に伝える。
	構造の劣化防止・維持・改善	仕様変更、機種開発、機能追加などのソフトウェアの変更時に構造劣化を起こさないように修正の方針を立て、構造の維持を図る。

これらの詳細は報告書を参照

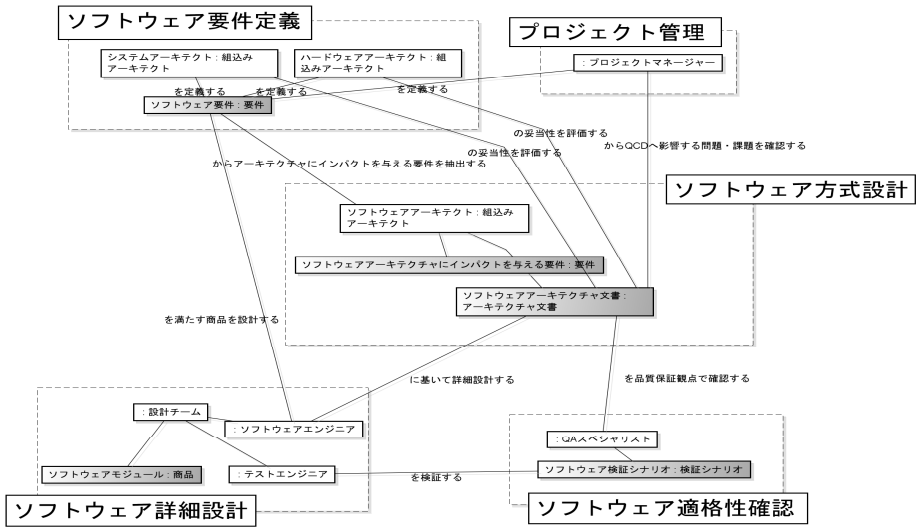
JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

4. アーキテクトの育成 －アーキテクトとドメイン知識

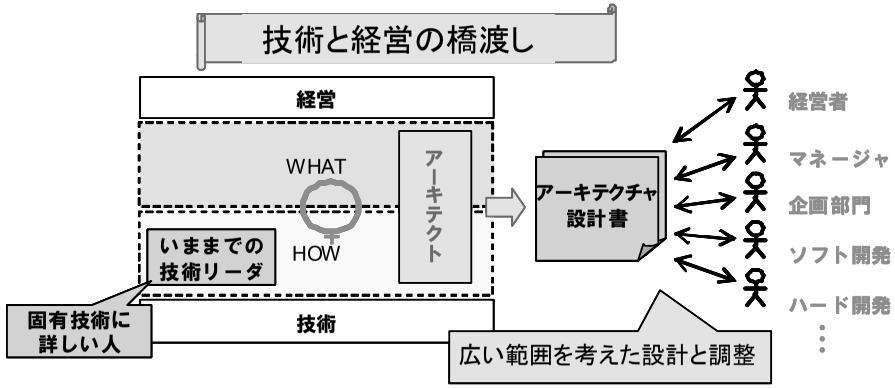


意識的な育成が必要

アーキテクトの活動 －アーキテクチャ文書とステークホルダー



アーキテクトの育成 －アーキテクトの人材像



アーキテクトの配置

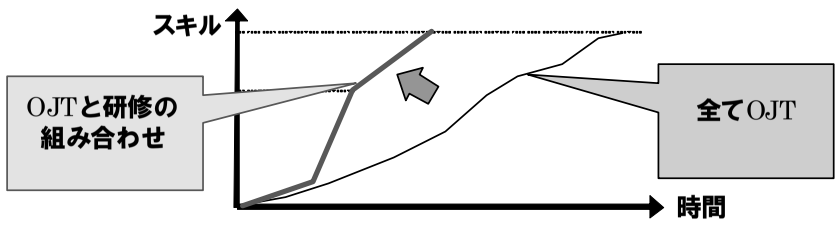
配置パターン	システムアーキテクト	ハードウェアアーキテクト	ソフトウェアアーキテクト	メリット	リスク
1	同一			・開発スピード ・全体最適化 ・迅速な意思決定	・アーキテクチャーの妥当性 ・ドキュメンテーション
2-1	兼任		別	・製造原価低減視点	・ソフトウェア開発への配慮
2-2	別	兼任		・開発スピード ・並行開発、非同期開発	・機能と構成品の整合
2-3	兼任	別	兼任	・ユーザー要求への即時対応	・ハードウェア開発への配慮
3	別	別	別	・設計多視点化 ・技術の相互研鑽	・開発期間 ・コスト

海外のアーキテクト育成の調査　海外調査サマリーから

	米国	欧州
産学連携	INCOSEというシステム工学の専門的団体があり、1990年頃から産学が連携している。特に人材育成ではINCOSEに認定資格制度があり、そのための教育プログラムが大学にある。	ARTEMISという産学連携プロジェクトを推進している。 2011年にSRA(Strategic Research Agenda)が発行されている。
官の関わり	INCOSEのCAB(Corporate Advisory Board)という組織会員52のうち、約1/4が政府系の重要な組織（軍関係、航空宇宙関係、研究所）である。	ドイツでは、半官半民の組織であるFraunhofer研究所が全国に７０の活動を行っている。ソフトウェア工学に特化した研究所も２カ所ある。
組込みシステムの基盤産業分野	航空中、軍事のソフトウェア工学技術をもとにメソドロジ化を行っている。	航空宇宙、原子力の重厚なソフトウェア工学を、自動車などの民間で使えるエンジニアリングに適用している。
これからの取組み分野	組込みシステムを、CPS(Cyber Physical System)としてとらえ、クラウドなどのITシステムの分野とは区別している。	ARTEMISでは、スマートビルディング、グリーン交通システム、ヘルスケアと健康増進を研究エリアとしている。
職業として	ソフトウェアアーキテクトを代表とする職種は、これからの成長が期待できる人気職種としてその地位を確立している。	エンジニアやプロフェッサーに対する職業階級としての経緯が感じられる。
スキルアップの仕組み	企業に就職したあとに、大学に入って学ぶというような流動性がある。MBAやロースクールのような仕組みがあり、意欲の高い人が通っている。	アーキテクトという職種があり、産学連携の組織で、その技術を習得する場がある。

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトの育成 –研修とOJTの組み合わせ



技術的なスキル以外のマネージメントスキルも必要
総合的なスキルが要望

しかし、… **育成は困難**

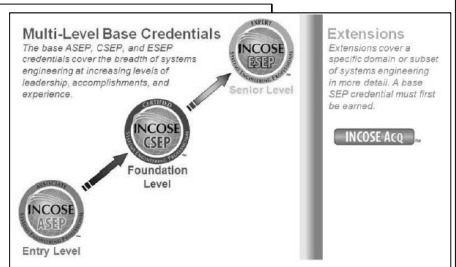
技術的なスキルはある程度は研修でも可
OJTも含めてアーキテクトとしての経験を積ませる2段階の育成方法が効果的

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

海外のアーキテクト育成の調査 –米国の例

システムエンジニアに必要なスキルや資質 INCOSE（**米国**）

高い数学の能力
強力な時間管理のスキル
システム思考の適正
良識
組織と効率性への強い願望
優れたコミュニケーション／セールスマンシップ
創造的問題解決
定量化スキル
技術能力
改善のための継続的なドライブ
機知
リスニングスキル
交渉スキル
外交忍耐
多様な個人のグループと、多くの環境に適応する準備が有り、対話する能力
探求心
学ぶための連続的な願望
リーダーシップスキル
倫理

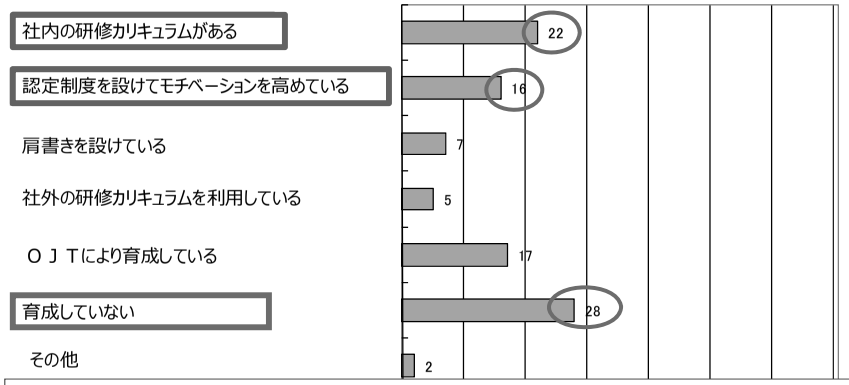


INCOSE システムエンジニア認定制度

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトの育成（アンケート結果から）

■ 育成施策（現実）　(2011年度アンケートから)

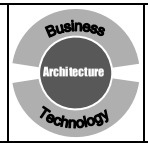
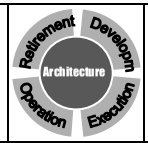
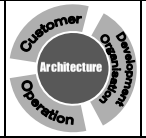
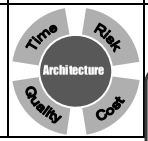


育成施策は
「社内研修カリキュラムがある」「OJT」「認定制度」
「育成していない（できない）」も比較的多い

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

海外のアーキテクト育成の調査 –欧州の例

Fraunhofer IESE **でのアーキテクチャ中心開発とそのセミナーカリキュラム**

1		あらゆるビジネス要求と技術的解決策をつなぐ	5		保守、開発、運用、実行というソフトウェアライフサイクルをカバーする
2		ソフトウェアシステムに関わる異なる組織間の要求を調整する	6		要求工学、コンポーネント工学、実装、統合、テスト、そして品質保証というソフトウェア開発の全ライフサイクルをスコープとする
3		様々なステークホルダーの要求とコミュニケーションの促進のバランスを取る			
4		コスト、時間、品質とリスクのバランスを取る			

日程	コンテンツ
第一期	ソフトウェア工学の導入 組込みシステムのソフトウェア開発 プロジェクトマネジメント
第二期	ソフトウェア品質保証 要求工学 ソフトウェアプロダクトライン工学
第三期	コンポーネントに基づくソフトウェア開発 モデルに基づくコンポーネント工学 リアルタイムシステム 信頼性工学
第四期	実習

アーキテクチャをソフトウェア工学のハブと見なし、アーキテクチャ中心の設計・開発を提唱している。

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトヒアリングの概要

目的:
- アーキテクトの育成などに対する事例
- アーキテクトやその育成に対する開発現場での意見
- 具体的なアーキテクト育成施策

などを収集・分析

方法:
- 12名のアーキテクトを当委員会で選出
- 1人または複数の委員が直接、面接ヒアリング
- 時間は1回のヒアリングで2時間から半日程度
- 事前に質問項目などは統一
- 事後に追加ヒアリングやメールでのやりとり

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

5. アーキテクトヒアリング

1. アーキテクトヒアリングの概要
2. アーキテクトヒアリングの結果（12名）
3. アーキテクトヒアリングのまとめ

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトヒアリングの質問

質問A. アーキテクトのスコープや立ち位置、経験
(1) ヒアリング対象者のプロフィール
(2) ヒアリング対象者が考えるアーキテクト像(役割や定義など)
(3) 重視するスキル(ETSS や IESE のモデルをベース)
(4) ヒアリング対象者がアーキテクトになるまでの経験
　アーキテクトとしての活動事例

質問B.アーキテクトの育成に関する項目
(1) ヒアリング対象者の育成経験・事例
(2) ヒアリング対象者が考えるアーキテクトの育成方針
(3) ヒアリング対象者が考えるアーキテクトの育成施策
(4) 育成の事例やその評価

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ヒアリング例 A氏(通信設備機器)

- アーキテクトのやるべき仕事
  - 「なぜそのアーキテクチャを発想しているか」を自分の言葉で伝える
  - 新しい問いを自分で立て解決していく
  - 大局を見て考える軍師
  - 製品の完成まで関わる
- アーキテクトのスキル
  - 一つの製品の構造だけでなく製品ラインの全体を貫く設計思想が必要
    - ETSS のスキルの個々ではない
  - ビジネス要求と技術的解決策をつなぐ
- アーキテクトの育成
  - 意図的に育てるものではない
  - 職場の文化により勝手に育つが、それを活性化させるために組織としての取り組みが必要
  - 短時間で育つために、キャリアパスに基づく教育は必要
  - 次にアーキテクトとして切磋琢磨するコミュニティが必要

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトヒアリングの質問ー重視するスキル

[1]技術要素	[1.1]技術、方式を評価・選択してシステムアーキテクチャを構築できるだけの技術理解が必要
	[1.2]プロダクト実現に必要な各種技術スキルとシステムの基盤となるプラットフォーム技術
	[1.3]専門技術
[2]開発技術	[2.1]システム要求分析
	[2.2]システム方式設計
	[2.3]ソフトウェア要求分析
	[2.4]ソフトウェア方式設計
	[2.5]システム結合
	[2.6]システム適格性確認テスト
[3]管理技術	[3.1]統合マネジメント
	[3.2]スコープマネジメント
	[3.3]タイムマネジメント
	[3.4]コストマネジメント
	[3.5]品質マネジメント
	[3.6]組織マネジメント
	[3.7]コミュニケーションマネジメント
	[3.8]リスクマネジメント
	[3.9]調達マネジメント
	[3.10]開発プロセス設定
	[3.11]知財マネジメント
	[3.12]開発環境マネジメント
	[3.13]構成管理・変更管理
[4]パーソナルスキル	[4.1]コミュニケーション
	[4.2]リーダーシップ
	[4.3]ネゴシエーション
	[4.4]問題解決
[5]ビジネススキル	[5.1]マーケティング

ETSS スキル一覧

このスキル一覧を元に優先順位などをヒアリング

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ヒアリング例 D氏(AV機器)

- アーキテクトのやるべき仕事
  - システムの全体構造を決める
  - 構造を維持する
- アーキテクトのスキル
  - 具体的な経験に裏付けられての抽象的に考える力
  - 開発技術が最重要
  - 要素技術、パーソナルスキル、管理技術が重要
  - ビジネス要求と技術的解決策をつなぐ
- アーキテクトの育成
  - 育成というよりは素養を持っている人がなる
  - アーキテクトは経験が必要である
  - 技術レベルの高い人と一緒に仕事をする
  - 世の中の技術を積極的に吸収する
  - デザインパターンのように網羅的なものは意義を感じない

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトヒアリングの対象者12名

A氏(通信設備機器)
B氏(通信設備機器)
C氏(通信設備機器)
D氏(AV機器)
E氏(医療機器)
F氏(エンタープライズ)
G氏(エンタープライズ)
H氏(コンピュータ周辺機器／OA機器)
I氏(コンピュータ周辺機器／OA機器)
J氏(コンピュータ周辺機器／OA機器)
K氏(コンピュータ周辺機器／OA機器)
L氏(コンピュータ周辺機器／OA機器)

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ヒアリング例 J氏(コンピュータ周辺機器／OA機器)

- アーキテクトのやるべき仕事
  - 定義が人によって異なり、期待される仕事も様々である
  - 設計方針を決め、ドキュメントを作成するのが主な仕事
  - 全体の枠組みから、いざとなればコードレベルまで見る
  - 重要なテクノロジーについては個別に理解する
  - 予算策定はプロジェクトマネージャーとともに関わる
- アーキテクトのスキル
  - 技術要素と開発技術を最重要
  - パーソナルスキル、ビジネススキル、管理技術が重要
  - 誰が見ても分かるドキュメントを書く力、わかりやすく説明する力
- アーキテクトの育成
  - 基本はドキュメント作成だが一定量のコードを書いた経験も必要
  - ドキュメント作成を座学だけでなく実際の課題を演習で実施
  - OJT では実開発で上司が理解を持つ
  - 部門を越えたコミュニケーション（例. 飲み会）

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ヒアリング例 G氏(エンタープライズ)

- アーキテクトのやるべき仕事
  - システムのコンセプトを提案、設計思想を決定し、実現に向けてガバナンス
    - なぜそう決めたかという設計の背景、理由、思想が重要
  - アーキテクチャにより顧客価値を高める
- アーキテクトのスキル
  - 各ステークホルダを説得するためのプレゼン力やリーダーシップが必要
  - プラットフォーム技術や技術理解などの技術要素が最重要
  - パーソナルスキルやマーケティング（ビジネススキル）が重要
  - コンセプトを創出するセンス・コンセプトを実現可能なレベルに体系化する分析力と総合力
  - 変化の本質を感じるセンス
- アーキテクトの育成
  - OJTを通してコンセプトデザインからプロトタイピングしていく
  - この機会がない人は Off-JTで体系的に計画的にスキルを高める
    - アーキテクチャー設計の疑似体験、社内外技術者との交流、メンタリング

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

ヒアリング例 L氏(コンピュータ周辺機器／OA機器)

- アーキテクトのやるべき仕事
  - 不具合を根絶する(バグを生み出す仕組みを根絶する)
  - みんながやろうとしないことを解決する
  - 危ないところなどの気配として嫌なにおいに気づく(リスク対策)
- アーキテクトのスキル
  - アーキテクトに向いている人は、まずはしがらみに囚われない素直な人
  - バグを取るトラブルを解決する能力
  - バグを出さないようにするトラブルを未然に防ぐ力
  - 誰も気づきにくい問題を気づくなど、誰もやらないことをする力
- アーキテクトの育成
  - 学校と違って自分で答えを見つける
  - 上司から仕事のやり方をおそわったら、自分の言葉で表現しなおす
  - ソースコードレビュー、モデルレビュー

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

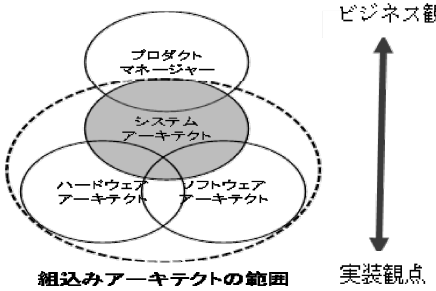
ヒアリング例 H氏(コンピュータ周辺機器／OA機器)

- アーキテクトのやるべき仕事
  - コンポーネント単位への分割と構造化した設計図を用いて、全体像を把握
  - アーキテクトは仕様があいまいな箇所などのアーキテクチャーが際どい箇所を押さえる
  - メンバーが自分自身の成長を実感できる機会(成長点)を仕事の随所に埋め込み、モチベーションを与える
- アーキテクトのスキル
  - スーパー過ぎないバランスの取れた技術力
  - みんなで解決していく力
  - リスク対応のマネジメント能力
  - ビジネス視点で語れる能力
- アーキテクトの育成
  - 色々な経験（海外経験や仕事のローテーション）
  - ピアレビューによる設計レビュー
  - 同じ分野の本を10冊読め

これはヒアリングの一部であり、詳細は報告書を参照

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトヒアリングの結果のまとめ
アーキテクトの分類、重要視するスキル



ヒアリングしたアーキテクトは、ビジネス観点重視から実装観点重視までバラエティに富んでいた

各アーキテクトが重要視するスキル

カテゴリ	A	B	C	D	E	F	G	H	I	J	K	L
[1]技術要素	–		1	2			1	1	2	1	2	2
[2]開発技術	–		3	1			4	4	1	2	3	1
[3]管理技術	–		5	4			5	5	4	5	4	3
[4]パーソナルスキル	–		4	3			2	3	3	3	1	
[5]ビジネススキル	–		2	5			3	2	5	4	4	

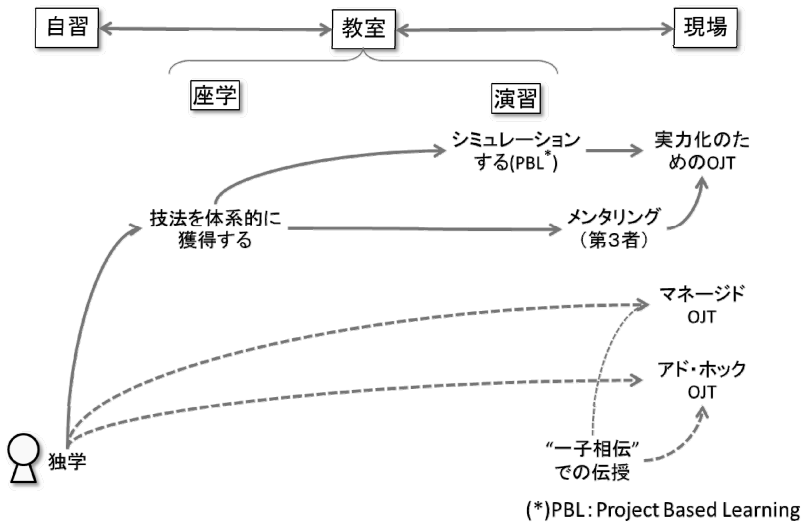
JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

重要視するスキル（小分類）

カテゴリ	細目	A	B	C	D	E	F	G	H	I	J	K	L
[1]技術要素	[1.1]技術、方式を評価・選択してシステムアーキテクチャを構築できるだけの技術理解が必要	-		1-1	2-1			1-1	1-1	2-1	1-1	2-1	2-2
	[1.2]プロダクト実現に必要な各種技術スキルとシステムの基盤となるプラットフォーム技術	-		1-2	2-2			1-2	1-2	2-2	1-2	2-2	2-1
	[1.3]専門技術	-							1-3	2-3	1-3		
[2]開発技術	[2.1]システム要求分析	-		3-1	1-1			4-1	4-3	1-1	2-1		1-1
	[2.2]システム方式設計	-		3-2	1-2			4-2	4-1	1-3	2-2	3-1	1-2
	[2.3]ソフトウェア要求分析	-							4-4		2-3		1-3
	[2.4]ソフトウェア方式設計	-							4-2			3-3	1-4
	[2.5]システム結合	-		3-3				4-3					
	[2.6]システム適格性確認テスト	-							4-5	1-2			
[3]管理技術	[3.1]統合マネジメント	-			4-1				5-1				
	[3.2]スコープマネジメント	-											
	[3.3]タイムマネジメント	-		5-3				5-3					
	[3.4]コストマネジメント	-		5-1				5-1			5-1		
	[3.5]品質マネジメント	-		5-2				5-2			5-2		3-3
	[3.6]組織マネジメント	-								3-3			
	[3.7]コミュニケーションマネジメント	-							5-3			4	
	[3.8]リスクマネジメント	-							5-2				
	[3.9]調達マネジメント	-											
	[3.10]開発プロセス設定	-			4-2					3-4	5-3		
	[3.11]知財マネジメント	-											3-1
	[3.12]開発環境マネジメント	-											
	[3.13]構成管理・変更管理	-								3-5			3-2
[4]パーソナルスキル	[4.1]コミュニケーション	-		4-1	3-2				3-3		3-2	1-1	
	[4.2]リーダーシップ	-			3-1			2-1	3-1	3-2	3-1		
	[4.3]ネゴシエーション	-		4-2					3-2		3-3	1-2	
	[4.4]問題解決	-						2-2		3-1			
[5]ビジネススキル	[5.1]マーケティング	-		2	5			3	2	5	4	5	

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクト育成のパス



JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクト育成の手法

OJT (On the Job Training)
- アド・ホックOJT
- マネージドOJT
- 実力化のためのOJT

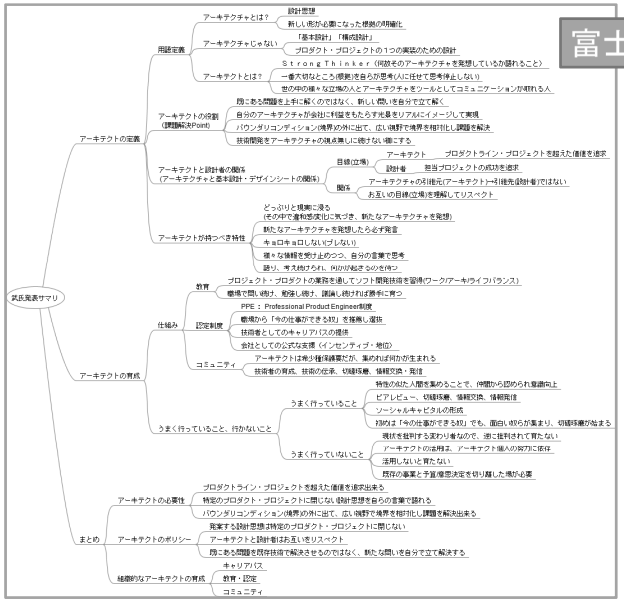
シミュレーション(PBL)

メンタリング
- OJT環境でのメンタリング
- 実業務外環境でのメンタリング

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトワークショップから

富士通研究所　武氏



JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクト育成に対するヒアリングまとめ

アーキテクトに必要な素養

- 俯瞰の視点、大局観、自分達だけでなく全体の構造への洞察力
- 重要事項を見極める力
- 抽象的に考える力
- 技術力を維持する姿勢、最新の技術に触れる積極性、自ら勉強し続ける姿勢
- バランスの取れた技術力
- コンセプトを実現可能なレベルまで体系化する分析力
- 何故そのアーキテクチャーを発想したかを自らの言葉で語る
- 関係者と積極的にコミュニケーションを取る姿勢

アーキテクトとしての心構え

- 座学や演習形式のポイントは、様々な視点があることを自覚
- ソースコードレビューやモデルのレビューがアーキテクト育成に役立つ
- 担当システムのアーキテクチャーの変遷を整理
- アーキテクチャーを考える仕事・活動に関わり続け、アーキテクト同士がアーキテクチャーについて意見をぶつけ合う
- 定期的なジョブローテーションは、アーキテクトとして重要な経験になる
- 同じ分野の本を10冊読めば、その技術が掴める
- 斬新な製品やサービスが出てきたときには、内部がどうなっているか、自分なりに考えてみるという習慣を持つ

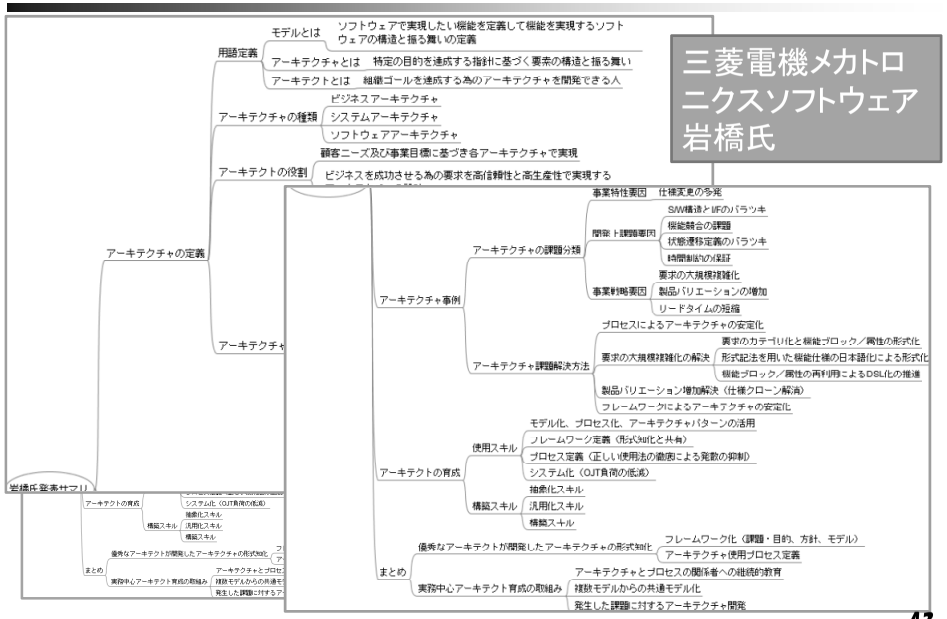
他多数

アーキテクト
標語集・名言集

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

アーキテクトワークショップから

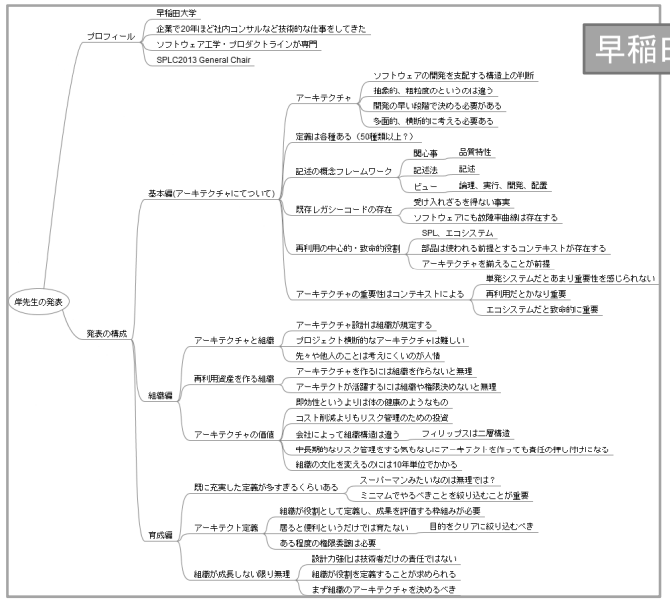
三菱電機メカトロニクスソフトウェア
岩橋氏



JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

6. アーキテクトワークショップから

早稲田大学　岸教授



JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

付録: JEITA活動報告の参考文献 1

- ワークショップ
  1. 2007 IESE/JEITA共同ワークショップ(2007年7月3日)
     http://home.jeita.or.jp/is/committee/software/070906/
  2. 組込み系開発スピードアップワークショップ2008 (2008年8月27日) 組込み系ソフトウェア開発をスピードアップ！
     http://home.jeita.or.jp/is/committee/software/080827/
  3. 組込み系開発スピードアップワークショップ2009 (2009年10月20日) 組込み開発に影響を及ぼす多様な特性とは？
     http://home.jeita.or.jp/is/committee/software/091020/
  4. 組込み系開発スピードアップワークショップ2010 (2010年10月29日) 日本型組込み開発における強みと弱み
     http://home.jeita.or.jp/is/committee/software/101029/
  5. 組込み系アーキテクトワークショップ2011 (2011年10月18日) 開発現場に求められるアーキテクトとは
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=244&ca=1
  6. 組込み系アーキテクトワークショップ2012 (2012年11月7日) アーキテクトは何を解決するか
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=488&ca=1
  7. 組込み系アーキテクトワークショップ2013 (2013年11月1日) アーキテクトをどのように育てるのか
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=644&ca=1
- CEATEC
  1. CEATEC JAPAN 2007 インダストリアルシステムトラック講演(2007年10月2日)
     http://home.jeita.or.jp/is/committee/software/071002/
  2. CEATEC JAPAN 2008 インダストリアルシステムトラック講演(2008年10月2日)
     http://home.jeita.or.jp/is/committee/software/081002/
  3. CEATEC JAPAN 2009 インダストリアルシステムトラック講演(2009年10月9日)
     http://home.jeita.or.jp/is/committee/software/091009/
  4. CEATEC JAPAN 2010 インダストリアルシステムトラック講演(2010年10月8日)
     http://home.jeita.or.jp/is/committee/software/101008/
  5. CEATEC JAPAN 2011 インダストリアルシステムトラック講演(2011年10月6日)
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=249&ca=1
  6. CEATEC JAPAN 2012 インダストリアルシステムトラック講演 (2012年10月3日)
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=493&ca=1
  7. CEATEC JAPAN 2013 インダストリアルシステムトラック講演 (2013年10月2日)

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

7 まとめ

まとめ
- 日本の組込みソフトウェア開発
- アーキテクトの必要性
- アーキテクトの役割と期待されるスキル
- アーキテクトの育成ーアーキテクトヒアリング
- ワークショップから

今後に向けて
- アーキテクトの育成のフィードバック
- モデリング観点からのアーキテクチャ設計

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

JEITA活動報告の参考文献 2

- JEITA報告書 他の専門委員会の報告書とセットで販売(*)
  1. 平成18年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-06-情シ-1、2、3) 要旨のみ閲覧可能
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=251&cateid=6
  2. 平成19年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-07-情シ-1、2、3) 以下は無料でダウンロード可能(**)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=299&cateid=6
  3. 平成20年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-08-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=350&cateid=6
  4. 平成21年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-09-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=389&cateid=6
  5. 平成22年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-10-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=423&cateid=6
  6. 平成23年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-11-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=461&cateid=6
  7. 平成24年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-12-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=512&cateid=6
  8. 平成25年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-12-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=556&cateid=6

(*) 3委員会分3冊セットで会員 5,250円、非会員10,500円
(**) 但し印刷不可能、クリッカブル不可能

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

予告　モデリング・ワークショップ

当委員会の今年度からの活動テーマ「モデリング」

- JEITA 組込み系ワークショップ2014
- テーマ「組込み系開発の実践的モデリング
  ～モデリングはなぜ失敗するのか？～」
  - 基調講演： 春名修介教授（大阪大学）
  - 事例講演： 2件を予定
  - ワークショップ: 全員参加型のワークショップ形式で実施
- 日時: 2014年11月7日　13:00-17:30
- 場所: 東京・大手町 JEITA 会議室（定員150名）
- 入場無料（但し参加申し込みが必要です）
  - 参加申し込み: JEITA ウェブサイトからお願いします
  - http://home.jeita.or.jp/cgi-bin/about/detail.cgi?ca=1&ca2=727

JEITA Copyright (C) 2014Japan Electronics and Information Technology Industries Association

JEITA組込み系ソフトウェア・ワークショップ2014

組込み系開発の実践的モデリング
～モデリングはなぜ失敗するのか？～
ソフトウェア事業基盤専門委員会の活動について
ワークショップ2014の狙いと課題認識

2014年11月7日

一般社団法人 電子情報技術産業協会
ソフトウェア事業委員会
ソフトウェア事業基盤専門委員会
委員長 五味 弘（OKI）

目次

1. JEITA ソフトウェア事業基盤専門委員会の紹介
2. 背景 － 日本の組込みソフトウェア
3. 問題解決に向けての JEITA 委員会活動
4. モデリング
5. ワークショップ2014 に向けて
参考文献

JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

1. JEITA ソフトウェア事業基盤専門委員会の紹介

＜JEITAとは＞ HPより抜粋
一般社団法人 電子情報技術産業協会（JEITA: Japan Electronics and Information Technology Industries Association）は、電子機器、電子部品の健全な生産、貿易及び消費の増進を図ることにより、電子情報技術産業の総合的な発展に資し、我が国経済の発展と文化の興隆に寄与することを目的とした業界団体です。
（中略）
JEITAは、まさに21世紀のデジタル・ネットワーク時代を切り拓いていくことを使命としており、電子情報技術の発展によって、人々が夢を実現し、豊かな生活を享受できるようになることを願っています。
このため、政策提言や技術開発の支援、新分野の製品普及等の各種事業を精力的に展開するとともに、地球温暖化防止等の環境対策にも積極的に取り組んでいます。

395社・団体が参加中

- ソフトウェア事業委員会
  - ソフトウェア事業戦略専門委員会
  - ソフトウェア事業基盤専門委員会
  - スマート社会ソフトウェア専門委員会

JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

ソフトウェア事業基盤専門委員会の活動概要

- 目的：「組込み系ソフトウェア分野」でのソフトウェア開発力の現状把握および基盤強化を図るための取組み
- 活動概要
  1. 毎年のテーマに沿った委員による議論（月１回～2回のペース）
  2. テーマに応じたアンケート調査（JEITA参加企業を中心に実施）
  3. ワークショップの開催（毎年1回）
  4. CEATECでの講演
  5. 関係団体との交流、意見交換
     （財団法人にいがた産業創造機構（NICO）、組込みソフトウェア管理者・技術者育成研究会（SESSAME）、組込みシステム産業振興機構（ESIP）、組込みシステム技術協会（JASA）、 ほか）
  6. 各種調査（海外調査も含む）
  7. 「ソフトウェアに関する調査報告書」の発行（年1回）

本専門委員会参加企業（2014年度）

沖電気工業、東芝、東芝ソリューション、日本電気、日立製作所、富士ゼロックス、富士通、三菱電機

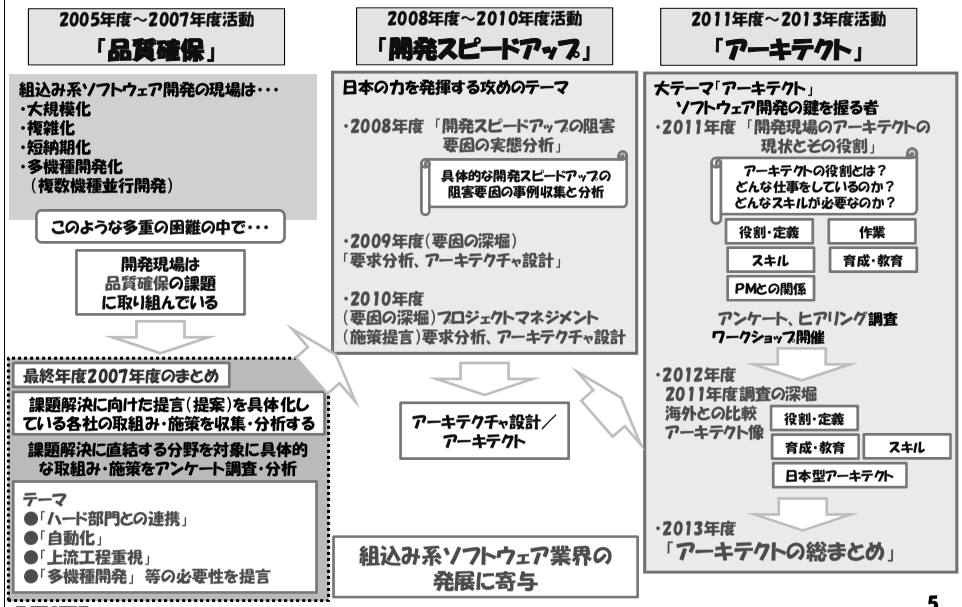
JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

2. 背景 – 日本の組込みソフトウェア 組込みソフトウェア開発に関する問題意識

- 「組込みソフトウェアは日本の強みの源泉であり価値創出のキー」と言われているが、
  組込み対象となるハードウェア機器は強いとしても、
  ソフトウェア開発力が国際的に見ても本当に強いのであろうか?
- 「擦り合わせ」の開発方法が日本の強みと言われているが、
  急激に増大している開発規模や短納期化、複雑化、並行開発の中で、
  現在でも「擦り合わせ」が強みになっているのであろうか?
- 何を強くすれば、
  日本の組込みソフトウェア開発の国際競争力を強化し、
  真に「日本の強みの源泉」たりうるものにできるのであろうか?
- 組込みソフトウェア開発を取り巻く状況:4つの大きな波を迎え撃つには
  - 大規模化
  - 短納期化
  - 複雑化
  - 複数機種並行開発

JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

3. 問題解決に向けての JEITA 委員会活動



JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

4. モデリング – 背景、課題

- なぜモデリングなのか？
  - 2010年度から2013年度までアーキテクトをテーマに活動
    アーキテクトが作るアーキテクチャ設計でもキーになる技術
  - 品質確保、開発スピードアップにも繋がるキーとなる技術
  - 課題
    - 上流工程でのモデリングが弱いのではないか
      ビジネスモデリングや要求モデリングなどが弱いのではないか
    - 実装モデリングに偏りすぎているのではないか
      UML や SysML で実装モデリングばかりしているのではないか
    - モデリングの抽象化ができていないのではないか
      捨象できずに、すべての情報を埋め込もうとしているのではないか
    - モデリングの目的を見失い、儀式的に作成しているのではないか
      何をモデリングしたいのかを明確にせずに、単に描いているだけ
      モデリングがゴールになっていて、それを利用していないのでは
    - そもそもモデルの定義があやふやでないのか
      モデルの目的や役割を決めずにモデリングツールを使っているだけ
  - そしてモデリングは失敗している

JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

4. モデリング – 定義、現状、課題、コツ、育成

モデリングの成功のヒントを探る

そのために・・・

- モデルの定義や分類
  - 目的、役割、作成工程、作成者など
- モデリングの現状や事例
  - 図、ツール、手法、対象、詳細度など
- モデリングの課題
- モデリングのコツ
- モデラーのスキルやその育成

これらをサブテーマにして活動する

JEITA Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

5. ワークショップ2014に向けて

組込み系開発の実践的モデリング
～モデリングはなぜ失敗するのか？～

基調講演
実践的モデリング論～難しいことを考えずにモデリングを実践するには～
春名　修介 教授（大阪大学）

事例講演
ExcelとEAを駆使したモデル設計現場適用例とMBSE展開に向けての課題
奥村　一幸氏（富士通）、石田　晴幸氏（富士通コンピュータテクノロジーズ）

事例講演
組込ソフトの開発現場におけるモデリング事例
久我　雅人氏(アマダ)

全員参加106人ワークショップ
徹底議論！モデリングの成功のヒントを探る
司会: 山田 大介 氏(ビースラッシュ)、アドバイザ:渡辺 登 氏（SESSAME）、講師陣
参加者の皆様の活発な議論・意見をお願いします！

 Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

付録: JEITA活動報告の参考文献 1

- ワークショップ
  1. 2007 IESE/JEITA共同ワークショップ(2007年7月3日)
     http://home.jeita.or.jp/is/committee/software/070906/
  2. 組込み系開発スピードアップワークショップ2008 (2008年8月27日) 組込み系ソフトウェア開発をスピードアップ！
     http://home.jeita.or.jp/is/committee/software/080827/
  3. 組込み系開発スピードアップワークショップ2009 (2009年10月20日) 組込み開発に影響を及ぼす多様な特性とは？
     http://home.jeita.or.jp/is/committee/software/091020/
  4. 組込み系開発スピードアップワークショップ2010 (2010年10月29日) 日本型組込み開発における強みと弱み
     http://home.jeita.or.jp/is/committee/software/101029/
  5. 組込み系アーキテクトワークショップ2011 (2011年10月18日) 開発現場に求められるアーキテクトとは
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=244&ca=1
  6. 組込み系アーキテクトワークショップ2012 (2012年11月7日) アーキテクトは何を解決するか
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=488&ca=1
  7. 組込み系アーキテクトワークショップ2013 (2013年11月1日) アーキテクトをどのように育てるのか
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=644&ca=1
- CEATEC
  1. CEATEC JAPAN 2007 インダストリアルシステムトラック講演(2007年10月2日)
     http://home.jeita.or.jp/is/committee/software/071002/
  2. CEATEC JAPAN 2008 インダストリアルシステムトラック講演(2008年10月2日)
     http://home.jeita.or.jp/is/committee/software/081002/
  3. CEATEC JAPAN 2009 インダストリアルシステムトラック講演(2009年10月9日)
     http://home.jeita.or.jp/is/committee/software/091009/
  4. CEATEC JAPAN 2010 インダストリアルシステムトラック講演(2010年10月8日)
     http://home.jeita.or.jp/is/committee/software/101008/
  5. CEATEC JAPAN 2011 インダストリアルシステムトラック講演(2011年10月6日)
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=249&ca=1
  6. CEATEC JAPAN 2012 インダストリアルシステムトラック講演（2012年10月3日）
     http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=493&ca=1
  7. CEATEC JAPAN 2013 インダストリアルシステムトラック講演（2013年10月2日）

 Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

JEITA活動報告の参考文献 2

- JEITA報告書 他の専門委員会の報告書とセットで販売(*)
  1. 平成18年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-06-情シ-1、2、3) 要旨のみ閲覧可能
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=251&cateid=6
  2. 平成19年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-07-情シ-1、2、3) 以下は無料でダウンロード可能(**)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=299&cateid=6
  3. 平成20年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-08-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=350&cateid=6
  4. 平成21年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-09-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=389&cateid=6
  5. 平成22年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-10-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=423&cateid=6
  6. 平成23年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-11-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=461&cateid=6
  7. 平成24年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-12-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=512&cateid=6
  8. 平成25年度 ソフトウェアに関する調査報告書Ⅰ、Ⅱ、Ⅲ(IS-12-情シ-1、2、3)
     http://www.jeita.or.jp/cgi-bin/public/detail.cgi?id=556&cateid=6

(*) 3委員会分3冊セットで会員 5,250円、非会員10,500円
(**) 但し印刷不可能、クリッカブル不可能

 Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

引き続き、講演が始まります

活発なご質問、ご議論をお願いします

 Copyright (C) 2014 Japan Electronics and Information Technology Industries Association

実践的モデリング論

- 難しいことを考えずにモデリングを実践するには -

2014年11月7日

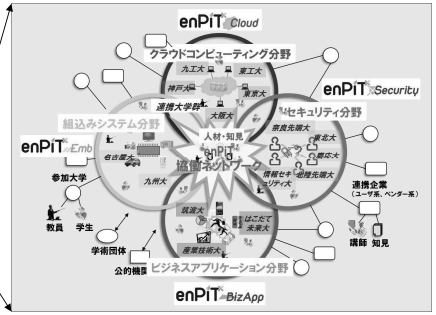
大阪大学大学院情報科学研究科
特任教授　春名　修介

自己紹介

- 1977年松下電器産業株式会社入社
  - 4ビットμプロセッサのクロス環境の開発から組込み業界へ
- 研究開発部門において，プログラミング言語処理系，ソフトウェア開発環境，リアルタイムOS，プロセッサアーキテクチャの研究開発及び製品開発に従事
  - アセンブリ言語からC言語への移行を推進．全国行脚
    - 一時期，ハード設計の経験も
  - C++，Javaのデジタル家電応用を推進．組込みソフトの大規模化の中へ
- 2001年よりエンジニアリング部門において，ソフトウェア開発力向上のための全社施策の企画に従事
  - 特に，組込みソフトウェア向けのアーキテクチャ設計方法，モデル駆動開発手法の普及び社内教育を推進
- 対外活動
  - 組込みシステム産業振興機構 システムアーキテクト育成カリキュラム「組込み適塾」の企画・講師
  - JEITA ソフトウェア事業基盤専門委員会委員（2005～2013.8.31）
- 2013年9月より大阪大学にて，情報教育ネットワークenPiT参画

enPiTとは

- 全国15大学が連携した，PBLを主体にした実践的情報教育プロジェクト（文部科学省事業）．



大学院生向けに1年間実施．単位認定
2013年度修了者　308名
2016年度の修了者目標：1200名程度

本日の内容

- モデルとは
  - 目的が重要 -
- アーキテクチャとは
  - 目的の一例として -
- 組込みソフトウェア・アーキテクチャ設計におけるモデル
  - 何を表現しなければならないか -
- 表記法の事例
  - 対象・目的に合った表現とは -
- まとめ：モデリング成功のポイント

アーキテクチャとは
- 目的の一例として -

モデルとは

- ある目的のため必要な部分のみを残した抽象的な表現
  - ソフトウェアの場合，コードより抽象度が高い
  - 目的外の内容は，省略されている
- モデルに共通した考え方（詳細を省略した後，残るものは？）

と 間の関係を表現

と表現方法は，目的により異なる

他者に全体像を伝えるための手段．関係者間での分かり易さも重要な観点

- 例
  - 構造モデル ⇒ 機能（責務）の利用関係を記述
  - 要求モデル ⇒ 要求間の関係を記述
  - フィーチャモデル ⇒ 構成機能の固定・変動の関係を記述
  - 状態遷移モデル ⇒ 状態間の遷移関係を記述
  - ビジネスモデル ⇒ ビジネスの諸要素（人・もの・金）の関係を記述

ソフトウェア・アーキテクチャの定義

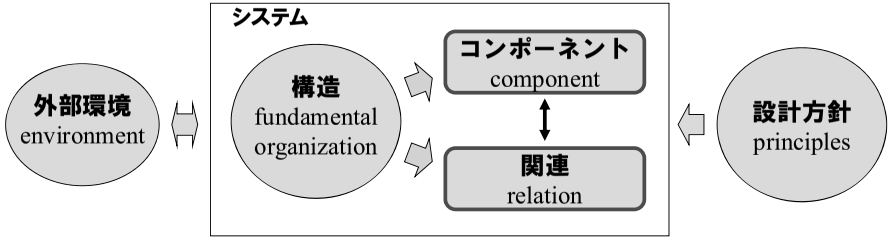
- 国際規格：※IEEE STD 1471-2000に見るアーキテチャの定義

※IEEE std 1471-2000:
IEEE Recommended Practice for Architectural Description of Software-Intensive Syste

説明も抽象的？

システム内のコンポーネントとそれらの関連や外部環境との関連で表現される基本構造及び設計とその進化の指針となる方針

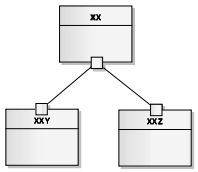
3.5 architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.



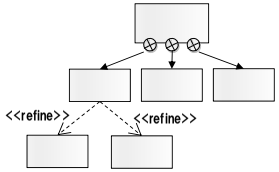
モデルの例

- 色々なモデル表現

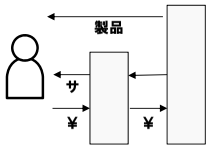
【構造図】
SysML内部ブロック図
コンポーネント間の関係を表現



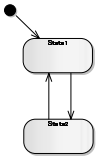
【要求図】
SysML要求図
要求間の関係を表現



【ビジネスフロー図】
ピクト図
¥，人，ものの流れを表現



【状態遷移図】
SysML状態遷移図
状態間の遷移関係を表現

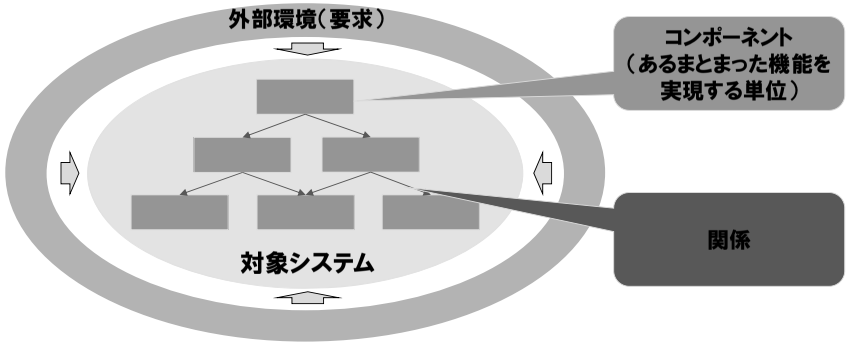


皆さんも記法に関わらず普段から行われていること
（何も難しいことはないハズ？）

ソフトウェア・アーキテクチャの私なりの解釈

- ある要求を実現するソフトウェアを設計するに当たり，
- システムをコンポーネントに分割，
- コンポーネントとその関係を規定し，
- 分割の考え方・要求との対応などと共に表現されたもの

分割の視点は色々（View）



視点（View）とは

- 全ての関係を一度に表現することは不可能
- 関心事毎にモデルを用いて表現
- 機能分担の視点：静的構造
- 動作の視点　　：動的構造
  - 状態遷移もこの範疇
- ネットワーク，分散
- 横断的関心

⋮

代表的な視点はあるが固定化されたものではない．設計するシステムを表現するために必要な視点を決めればよい．

よくできた概念モデルです

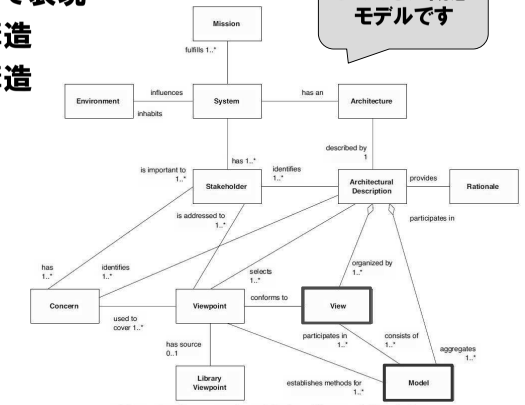


Figure 1—Conceptual model of architectural description

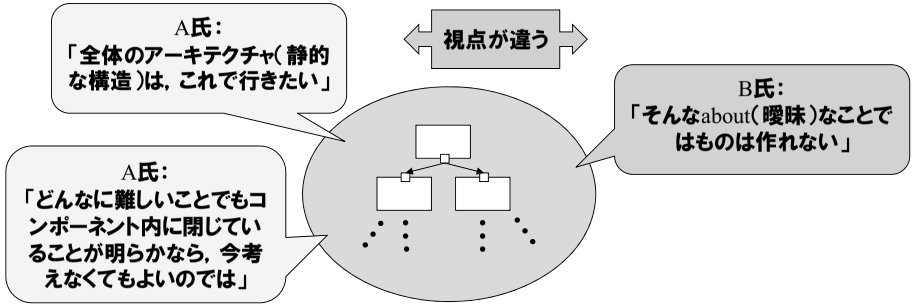
IEEE std 1471-2000より引用

アーキテクチャ設計の目的

- 目的
  - 全体構造を可視化
    - 》システムをどのように分割し，分割単位：コンポーネント間の関係を規定するか
    - 》コンポーネント内に閉じた詳細には言及しない ⇒ モデル化
  - ステークホルダ間で開発するシステムの全体像を共有する
- 効果（もっと色々あります）
  - 設計制約を与えることにより，担当者間の擦り合わせを減らす
  - 設計漏れによる大きな手戻りを減らす
  - 再利用など，設計時に考慮するべき非機能用件への対応が明確になる
  - 修正時に，修正の影響範囲が明らかになる
  - 揉め事が起こったとき，論理的に正しい方向に議論を進めることができる
  - 次の機能進化の議論のベースとなる　などなど

事例①：アーキテクトA氏の憂鬱

- 抽象化と曖昧さ



抽象化することと，曖昧になることとは違う

コンポーネント内の詳細は欠落するが，コンポーネント間の関係は厳密
細かいことが解決しないと先に進めないタイプの人は，アーキテクトには向いていない？

組込みソフトウェア・アーキテクチャ設計における モデル

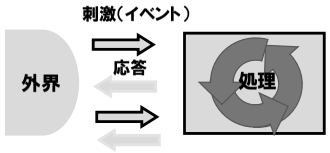
・組込みソフトウェアのアーキテクチャ設計では
何を表現しなければならないか
・どのようなViewがあるのか

組込みソフトウェアの特徴：釈迦に説法ですが

① 入力出力型

・入力と出力に時間関係／制約はない
・処理は，逐次的に実行
・処理が完了するまで，次の入力はない
・一回の入力毎に処理が完結
例：数値計算など
　アルゴリズムの教科書に出てくる処理

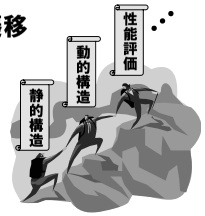
② イベント駆動型 ← 組込みソフト

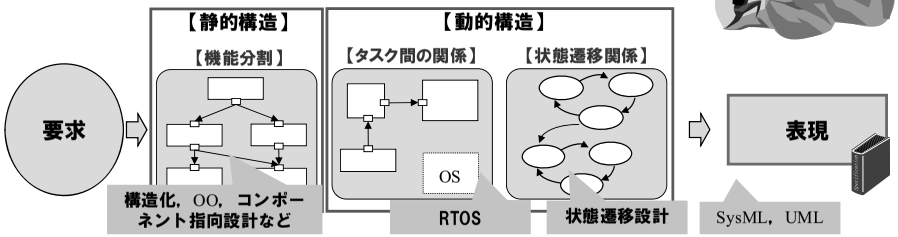


・入力と出力に時間関係／制約がある
・処理は，並行的に実行
・1つイベントの処理中にも，
　別のイベントの処理をする必要がある
・中断，処理の変更は例外ではなく設計項目
例：ハード制御，通信制御，・・・・

構造設計の3つの視点（View）

- 静的視点と動的視点の両面を表現する必要がある
  ＜組込みソフトウェアは，静的な構造設計で，ひと山越えるのみ＞
  - 静的構造：①機能分割（責務分割）
  - 動的構造：②並行動作（タスク間の関係），③状態遷移
- 表現することも同時に重要
  - 設計力と表現力は表裏一体
  - 関係者が理解し易い表記法の選択も重要

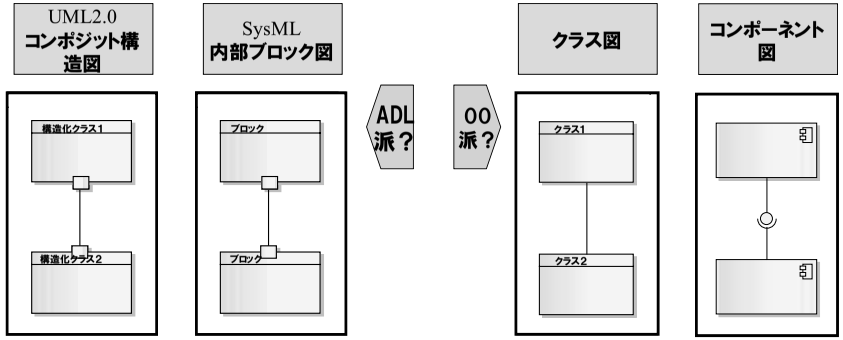




表記法の事例
- 対象・目的に合った表現とは -

その他の構造表記方法

- コンポーネントとその関係の表現方法は色々あります
  - モデル駆動開発を視野に入れると異なりますが，宗教論争になります
  - それぞれ，利害得失がありますので，
  - 使用ツールや開発実態に合わせて決めればよいと思います



代表的な構造の表記方法

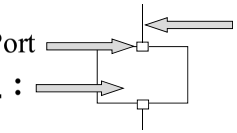
- 国際規格：IEEE STD 1471-2000に見る静的構造の表現要素例

C.1 The structural viewpoint in software architecture 付録Appendix C.1

Viewpoint language

The structural viewpoint depends on the following entities:

- *Components* (individual units of computation)
- Components have *ports* (interfaces)
- *Connectors* (represent interconnections between components for communication and coordination)
- Connectors have *roles* (where they attach to components)



I/F：Port

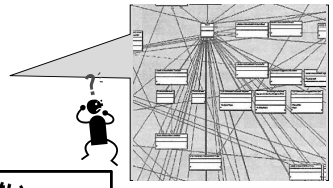
固有の処理：
Component
（独立性の高い）

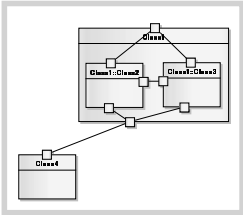
関係：Connector

C&C（Componet＆Connector）モデル
ADL（Architecture Description Language）の
図的表現として著名
通信業界のSDL，SysMLの内部ブロック図，
UML2.0のコンポジット構造図もこの流れ

事例②：蜘蛛の巣クラス図

- クラス図の苦手な点
  - クラス図は階層の表現が苦手
  - 少し大きい図になると蜘蛛の巣状態に

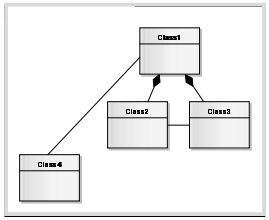




SysML内部ブロック図
UML2.0 コンポジット構造図

クラス図では

階層構造がわかり難い



オブジェクト指向言語の仕様を反映
実装に近い
今ほど，大規模ではない頃に生まれた

クラス図の ♦ ◊ ▲ △ は，
要注意
クラス自体の定義と
I/Fとしての関連とは
区別して考える

SysMLでは
ブロック自体の定義と
ブロック間の関係の記述を
分離

C&Cモデルの現実的な意味

- 特別なものではなく，あるモジュールの持つ基本要素を表現したもの

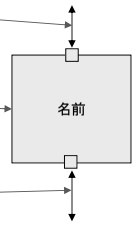
名前（モジュール名）

C&Cモデル
ADL
SysML内部ブロック図 など

提供API：
外部に提供する機能のためのインタフェース

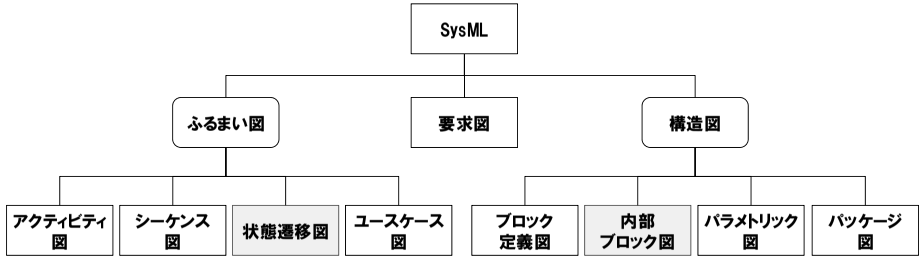
実現する機能（責務）

参照API：
下位に委譲する機能のためのインタフェース



表記法の候補

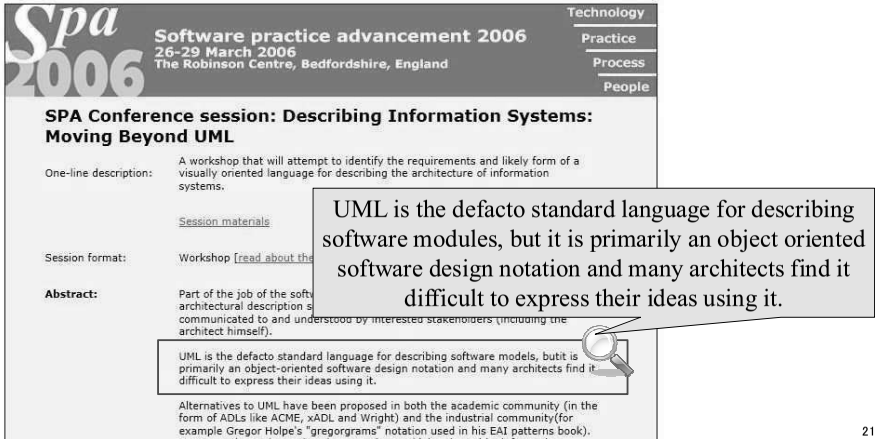
- SysML
  - UML2.0からの7種類の図に要求図とパラメトリック図を加えた9種類の図
  - 階層構造の記述が可能な内部ブロック図に着目



アーキテクチャ記述としてのUML見直しの動き

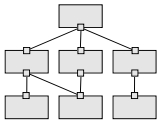
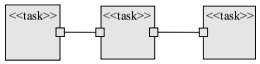
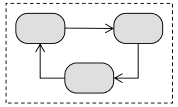
- 2000年位から，欧米ではアーキテクチャ記述の観点からUMLの見直しが始まる → SysML，AutoSarへの動き

http://www.spaconference.org/spa2006/sessions/session22.html



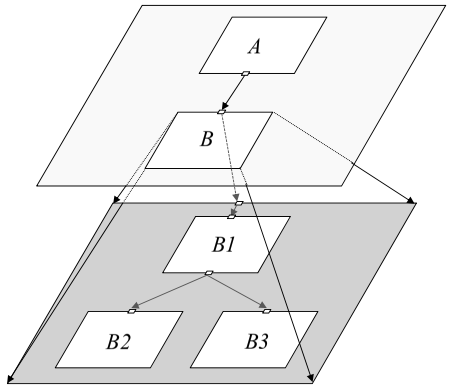
選択した表記法

- 何でもよいので，開発プロジェクトの中で統一することが重要
  - プロジェクトメンバーが親しんでいる・ツールのサポートがある・・・
  - 組込みソフトウェアの場合，最低限，静的構造・動的構造・状態遷移は決めておく

静的構造	動的構造	
静的構造図	タスク関連図	状態遷移図
SysMLの内部ブロック図 または コンポジット構造図(UML2.0) 	SysMLの内部ブロック図 または コンポジット構造図(UML2.0) 	状態遷移図(SysML，UML2.0) または 状態遷移表 
機能ブロック(コンポーネント)とその利用関係を記述	メッセージ，割込み，共有メモリ，などの関係(I/F)を記述	静的構造の機能ブロック単位で記述

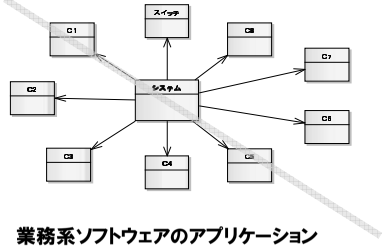
一番のポイント：詳細化構造の表現

- 規模が大きくなると，コンポーネントの粒度も大きくなり，各コンポーネントの内部構造を示す必要が生じる
- 構造の表現には階層構造(詳細化構造)の表記が不可欠

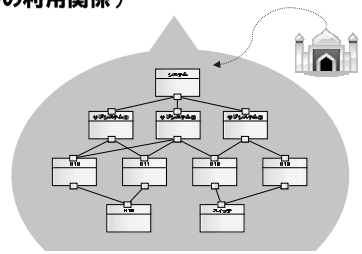


事例④：放射線状のクラス図

- システム階層の変化が表現されないと，設計意図が伝わらない
- 特に，システム階層の全てが開発対象の組込みシステムでは，
  - 一般常識
    - 》上がユーザに近い（抽象度が高い），下がハードウェアに近い（具体的）
    - 》上が下に処理を依頼する（機能分割時の利用関係）



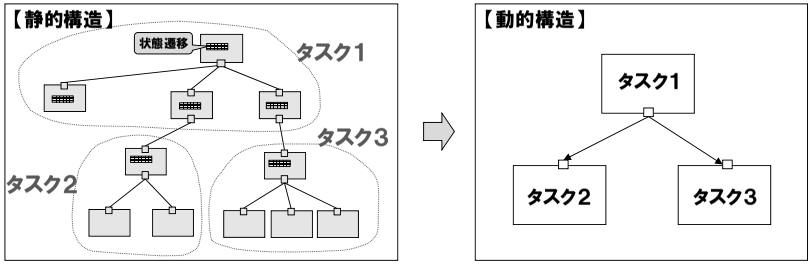
業務系ソフトウェアのアプリケーション層の中では出てくることもあるかも？



古来からの定石：モスク型は理解し易い（その他：STS分割，DFDの活用など）

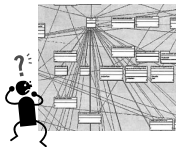
視点（View）間の整合

- 各視点毎のモデルに一貫性がなければ，全体として設計の意図が理解できない
  - View間の整合の方法を考えておく必要がある
- 定石
  - 並行に動作しない静的構造の各コンポーネントをタスクにまとめる
    - 》逆を言えば，並行に動作するコンポーネントは，タスクに分ける
  - 状態遷移は静的構造の単位で考え，構造と対応を明確にする



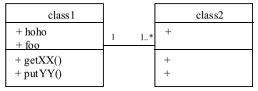
事例⑤：MDDに向けての注意

- 「全体を俯瞰するモデル」と「コード生成可能なモデル」は，観点が異なることがある
  - 「コード生成可能なモデル」の発想で，全体を考えて行くと蜘蛛の巣状態に



どうしても
コンポーネントの粒度
が小さくなる

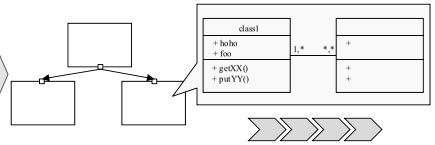
【コード生成可能なモデル】



コードがモデルに変わっただけスパゲッティモデル？

- 全体構造を表現するモデルから詳細化してコンポーネント内のコード生成可能なモデルとの整合を取る

アーキテクチャ
全体に影響を与える重要な関係を押さえる
（アーキテクトの仕事）

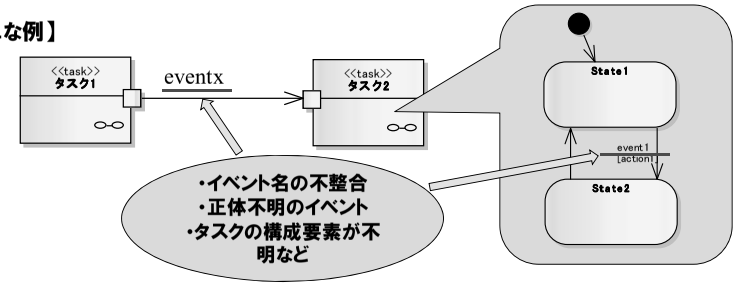


コンポーネント内
コード生成可能なモデルを作成
（モデラの仕事？）

事例③：本人にしか分からないモデル

- View毎のモデル記述では，各View間の整合が取れていないと，読み手は理解できない
  - また，実現可能性が？（絵に描いた餅）の可能性が大

【簡単な例】



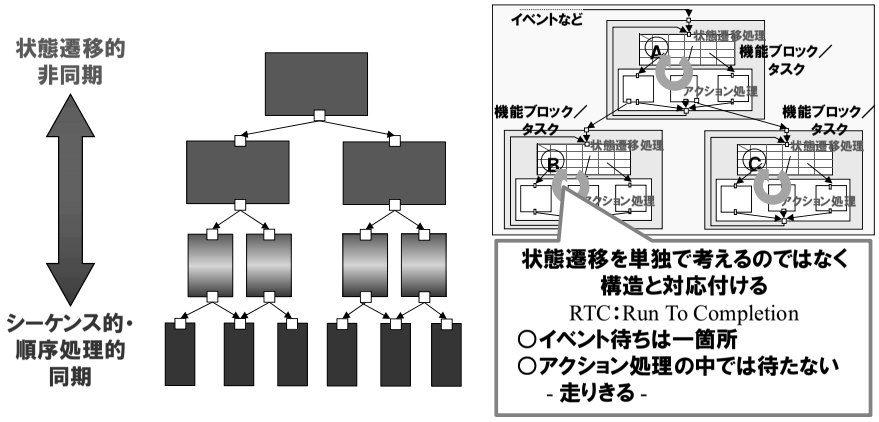
- 一般的な傾向
  - 自分が苦労したことを書きたがり，当たり前と思っていることは，書かない
  - しかし，当たり前と思っていることこそが第三者から見ると重要なことが多い

ご清聴ありがとうございました

106人ワークショップでの積極的な
ご議論をお願いいたします

組込みソフトウェアの基本構造

- 3視点(静的構造，動的構造，状態遷移)の融合設計
  - RTOS・状態遷移を組み合わせて，非同期処理を小さな同期処理(順序処理)の集合体に分解する



まとめ：モデリング成功のポイント

- 対象・目的を明らかにする
- 対象・目的に適したモデルを選択する
  - 一つのモデルに複数視点を盛り込まない．View毎にモデルを記述．
  - View間の整合を取ることが設計力．
  - 表記法の細かい文法に囚われると本末転倒．
- 昔からあるソフトウェアの基本を忘れず
  - 静的構造：構造化設計，DFD(Data Flow Diagram)の活用 など
  - 動的構造：RTOS，状態遷移，走りきり(RTC) など
- 抽象度をコントロールする
  - コードの延長ではなく，全体を俯瞰できる抽象度の発見が成功のポイント

細かすぎず，一般化しすぎず　一番難しい

モデル設計に踏み切れない三大要因

FUJITSU

■従来の開発プロセスを変更したくない

■予算上の問題
（ツール購入や開発メンバーへの教育費）

■モデル言語で書くメリットがよくわからない

Copyright 2014 FUJITSU LIMITED

FUJITSU
shaping tomorrow with you

JEITA組込み系ソフトウェアワークショップ2014

ExcelとEAを駆使したモデル設計現場適用例とMBSE展開に向けての課題

2014年11月7日
富士通株式会社
共通開発本部
ネットワーク管理ソフトウェア開発統括部
奥村 一幸

Copyright 2014 FUJITSU LIMITED

FUJITSU

1.プロジェクト実践例紹介

■ 1．プロジェクト実践例紹介
■ 2．MBSE適用に向けての課題

Copyright 2014 FUJITSU LIMITED

自己紹介

FUJITSU

■ 氏名：奥村一幸
■ 経歴
入社当時ファームウェア開発、95年頃より伝送機器運用管理業務アプリケーション開発に従事
■ 1991～1995
・富士通北陸通信システム株式会社入社　伝送機器ファームウェア開発
■ 1995～
・伝送機器運用管理業務アプリケーションソフトウェア開発
■ 2005～
・富士通株式会社へ統合
■ 2010
・社内プロフェッショナル制度　認定
■ 2012～
・モデル設計要素を排他設計プロセスに試行
■ 2014～
・MBSE設計プロセス適用展開活動

Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

1.プロジェクト実践例紹介

FUJITSU

・設計書執筆担当者全てにモデリングツールである
EntepriseArchitectを使用させる？
・要件定義書や処理フロー、ER図を全担当者にEAに写像させる作業を実施してもらう？
⇒　開発工数肥大化や作業効率低下

作成する設計書から必要情報のみを抽出しEAに取り込めないか？
従来の設計書内にはモデル要素が散らばっている
【ツールの検討】（詳細は石田氏発表で）
・既存設計書の記載ルールを定義。
・ツールにて設計書 ⇒ EAへ写像させる
これによって、
設計者は、ほぼ今まで通りの作業を実施可能に。

Copyright 2014 FUJITSU LIMITED

1.プロジェクト実践例紹介

FUJITSU

【大規模ソフト開発の悩み】

■膨大なユースケースにおける機能間資源の関連性が不明確
⇒　複数の開発担当者が異なる機能の処理フローを個々に作成。
全体像把握が困難

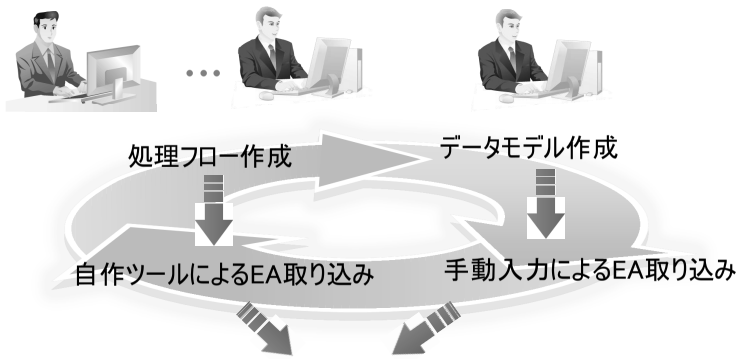
■システム試験の組み合わせケースは人がランダムに抽出
⇒　タイミング問題を検出する適切な試験ケースが抽出出来ていない。
競合試験での抜け漏れ

Copyright 2014 FUJITSU LIMITED

1.プロジェクト実践例紹介

FUJITSU

【EA取り込みまでの作業フロー】



全要件×リソース競合マトリクス生成

【設計工程】
・排他方式設計へのインプット
・機能間レビュー時のインプット

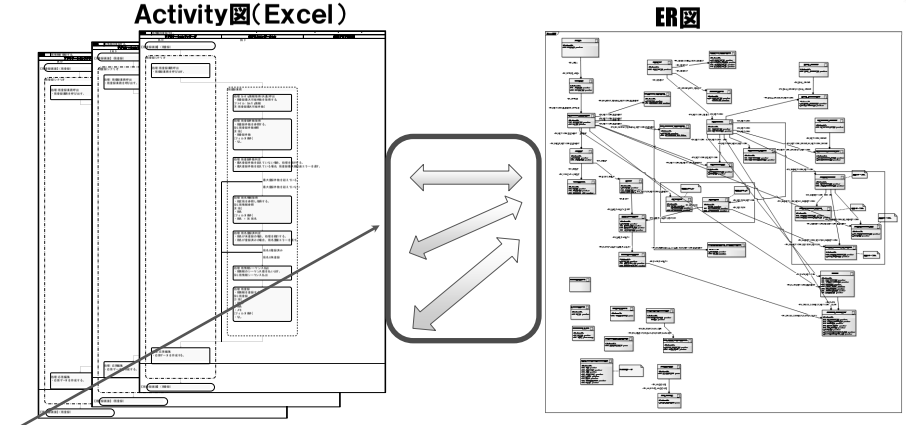
【試験工程】
・競合試験項目抽出インプット

Copyright 2014 FUJITSU LIMITED

1.プロジェクト実践例紹介

FUJITSU

Activity図（Excel）　　ER図



【着目点】
要件（ユースケース）毎に対象ブロックを紐付けることで要件毎及び要件間で関係するブロックが明確になることでソフト開発の悩みを解決出来るのでは？
・全ユースケース間のリソース利用関係の把握
・機能間競合システム試験項目の最適抽出

Copyright 2014 FUJITSU LIMITED

2.MBSE適用に向けての課題

FUJITSU

■競合試験の環境構築が大変なので効率化したい
課題：設計書から競合の自動検出

■性能等の見積もり精度を向上したい
課題：設計書を使ってシミュレーション

■設計から適切な試験を抽出したい
課題：設計書からの試験項目自動抽出

Copyright 2014 FUJITSU LIMITED

FUJITSU

2.MBSE適用に向けての課題

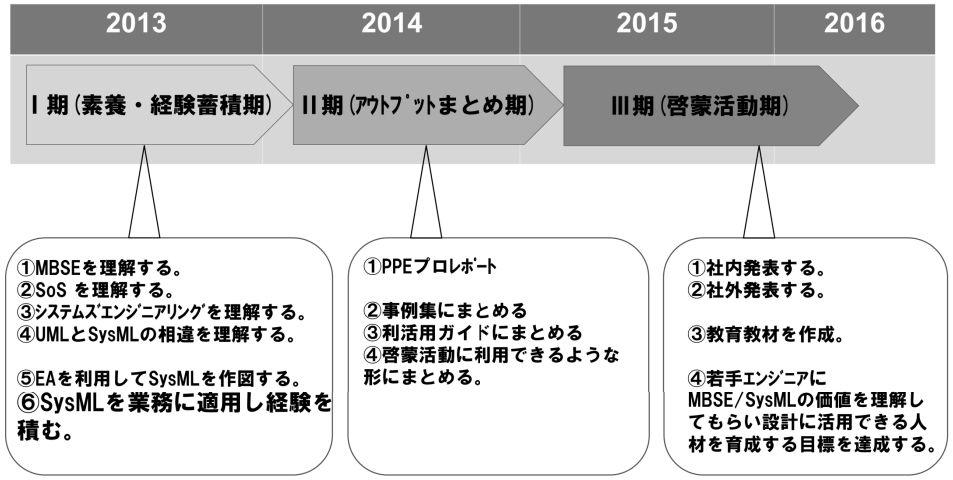
- 1. プロジェクト実践例紹介
- 2. MBSE適用に向けての課題

Copyright 2014 FUJITSU LIMITED

2.MBSE適用に向けての課題

FUJITSU

■活動ロードマップ



■プロセスを変化させるにはメリットもあるがデメリット（リスク）も付き物。根気よく継続していくことが重要。

Copyright 2014 FUJITSU LIMITED

2.MBSE適用に向けての課題

FUJITSU

■草の根活動の紹介

富士通プロフェッショナル・プロダクト・エンジニアによる、

システム開発におけるMBSE(*1)の適用を最終目標とし、個々のプロジェクトが抱える課題解決アプローチとしてSysMLを用いた事例蓄積と設計プロセス導入への牽引活動を行っている

背景

世の製品やサービスはもはや単体では成立しない。
システムは複数のシステムで構成され、大規模化、複雑化している。
システム開発当初からシステム全体を捉え、表現することが必要。

(*1)
MBSE(Model Based System Engineering)
背景に記載した課題解決することを狙いとしたプロセスを定義

Copyright 2014 FUJITSU LIMITED

FUJITSU

shaping tomorrow with you

ExcelとEAを駆使した モデル設計現場適用例

FUJITSU
shaping tomorrow with you

2014年11月7日
富士通コンピュータテクノロジーズ
組込みシステム技術統括部アーキテクチャ部
石田 晴幸

PROFESSIONAL PRODUCT ENGINEER

1292karch01

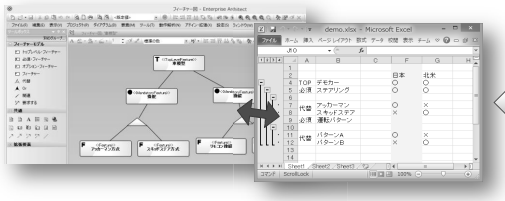
Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

背景と自己紹介

FUJITSU

■ 背景
- ■ 大規模な監視制御システム
- ■ 開発チームは、過去にデータベース・テーブルのアクセス競合で苦労したことがある

■ 自己紹介
- ■ モデル駆動開発ツール BricRobo の開発に関わる
  http://www.fujitsu.com/jp/group/fct/products/bricrobo/index.html
- ■ 最近は、UMLモデリングツール Enterprise Architect（EA）のアドインを開発



これは、EAでフィーチャー図を描くアドインです。
組込み総合技術展（ET2014）の富士通ブースで展示します。

記載されている製品名などの固有名詞は、各社の商標または登録商標です。

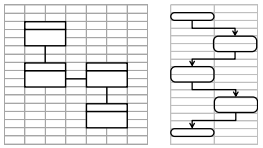
Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

課題

FUJITSU

■ 目的
テーブルアクセスにおける機能間の競合を洗い出すこと

■ 現状
- ■ テーブル構造を表したER図(Excel)
- ■ 機能ごとに振舞いを記したActivity図(Excel)
  - ・どのテーブルをどうアクセスするかが書いてある(Create,Read,Update,Delete)
  - ・大勢で書いていて全体像が掴めない



ER図　Activity図

■ 課題
ER図とActivity図を組み合わせて
機能vs.テーブルのマトリックスを
生成する

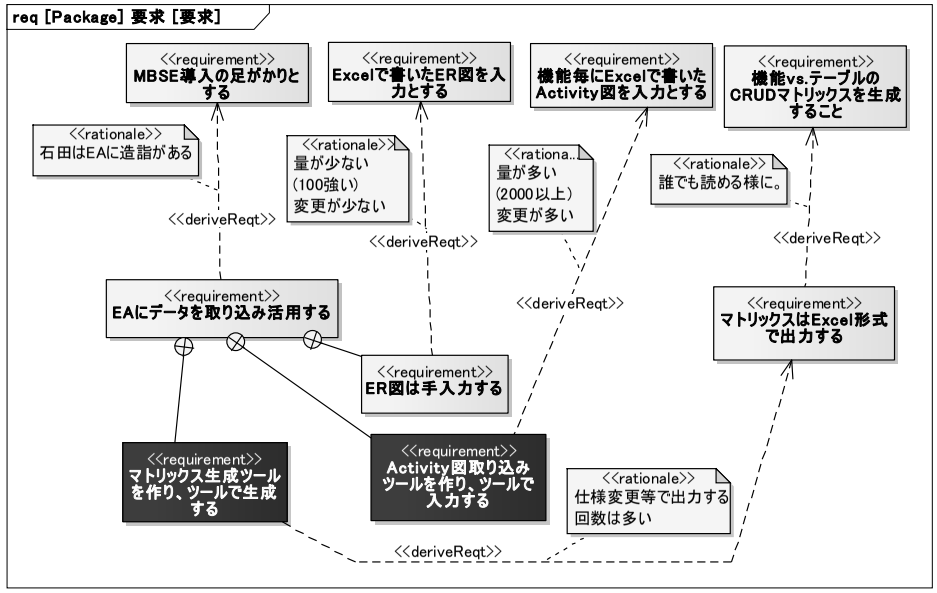
競合

	テーブルA				テーブルB				テーブルC			
機能1	C	R	U	D								
機能2					C	R	U	D				
機能3						R						
機能4									C	R	U	D
機能5												
機能6		R	U									
機能7												
…												

Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

考え方

FUJITSU



Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

評価

FUJITSU

■ マトリックスを見て、競合の多さ（1300件）に設計変更
→テーブルアクセスをシリアライズするキューを配置

マトリックスを見ながら競合試験項目を抽出してもらう目論見だったが、設計変更により**テーブルアクセス競合の問題を解決**

■ 名前のブレ
- ■ Activity図に記述されたテーブル名がER図のテーブル名と一致しない
  - ・プログラム的に吸収可能なブレ、プログラム的に吸収不可能なブレ
  - **・仕様書の曖昧さを炙り出す**

■ モデリングツールのカスタマイズの必要性
- ■ 入出力ファイル形式への対応
- ■ 独自のモデル変換・コード生成への対応
  （Model Driven Architecture や Domain Specific Modeling）
- ■ 独自のモデル編集効率向上施策（アドインによる編集時のサポート）

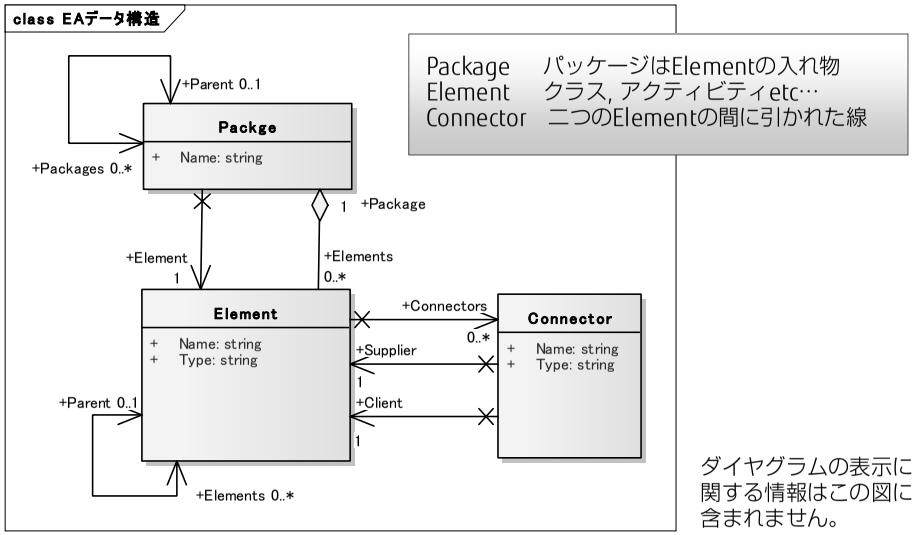
記載されている製品名などの固有名詞は、各社の商標または登録商標です。

Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

EAのデータ構造

FUJITSU

■ このくらいわかれば、EAのデータを色々いじれます。



ダイヤグラムの表示に関する情報はこの図に含まれません。

Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

FUJITSU

shaping tomorrow with you

施策

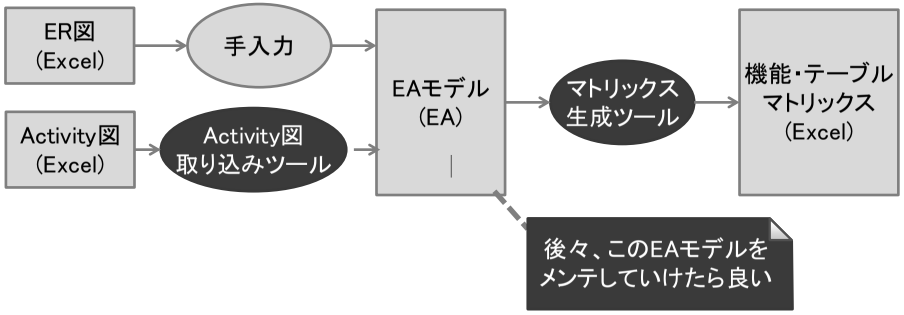
FUJITSU

以下の二つのツールを開発・運用する

■ ExcelのER図とActivity図を読んでEAに入力するツール

■ EAモデルから機能・テーブルマトリックスを出力するツール

いずれもC#で作成。下記DLLによりEAとExcelを操作
EA　Interop.EA.dll
Excel　microsoft.office.interop.excel.dll



Copyright 2014 FUJITSU COMPUTER TECHNOLOGIES LIMITED

筑波大学 University of Tsukuba

JEITA 組込み系ソフトウェア・ワークショップ 2014
組込み系開発の実践的モデリング

組込ソフトの開発現場における モデリング事例

～モデル化と反復プロセスを組織的に推進する～

久我 雅人

1

私とモデル化技術のかかわり

- T社時代
  - 前期：設計図等の技術を知らなかった時代
  - 後期：設計図面を使い始めた時代
- R社時代
  - モデル化技術によって高生産性を発揮した時代
- D社時代
  - 異質なモデル化技術に遭遇した時代
- そして現在
  - 再び、モデル化技術で品質確保する

2

T社時代の開発のやり方（前期）
★T社初の本格的MFPのコントローラ・ソフト開発

- 手法･･･といっても、特にない。
  - 専任のアーキテクトが決めたアーキテクチャにしたがって設計していました。
  - そのころは、まだ「いかに機能を盛り込むか」が重点課題でした。
  - 品質についての意識は低かったと思います。
  - はじめは要求分析の責任者、途中からUIアプリケーションの設計リーダーを兼ねました。
- 苦労した点
  - 品質をどのように確保してよいかわからず、作りこみレベルが不十分なまま総合試験を始めてしまったり･･･
  - しかも私の管理が甘く、基本設計から詳細設計へうつる段階で、内容をキチンとレビューできていなかったり･･･
  - そのため、UIだけで500件以上のバグを抱えていました。

3

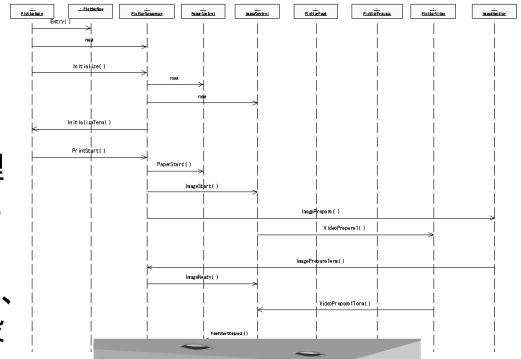
T社時代の開発のやり方（後期）
★T社初の本格的MFPのコントローラ・ソフト開発

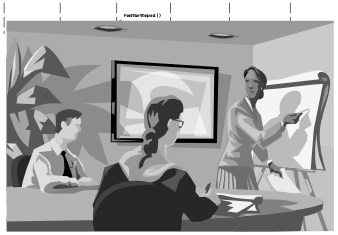
- 何が問題だったか？
  - アーキテクチャを意識していない
    - アーキテクチャとソースコードが結びついていない
  - 問題が起きると、何が問題か？をつきとめるまでが大変
    - つきとめるための道具も持っていなかった
    - その結果、必ず担当者に頼り、担当者が手一杯だと人を投入して解決しようとしていた
  - アーキテクチャに問題があるのでは？と気づいたときが転機だった
    - 機能開発＋バグ対策と並行してアーキテクチャの再構築を進めた
    - 一気に不具合が収束した
- アーキテクチャ（とそれを表現すること）の重要性を知った！

4

ソフトウェア設計図との出会い

- いまにして思えば・・・
  - タスク間のメッセージのやりとりと、各タスクのメッセージ受信時の処理を図にすることが契機に
  - 毎朝、ホワイトボードでチームメンバーが集まり、タスク間のやりとり（＝設計の動的側面）を図にしながら議論・整合
  - モデル化へのいわば≪胎動≫の時期？？





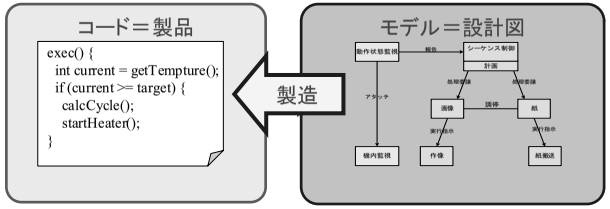
R社でモデル化技術と出会う

★オブジェクト指向技術の本格導入

- オブジェクト指向開発手法をとりいれたモデル開発
  - 抱えていた問題点
    - 仕様変更による2次障害仕様変更の工数を低減したい
    - ソースコードの再利用ではむしろ悪くなる品質を救いたい
  - この課題解決のため、当時の上司が提案
    - でも、ほんとうは上司がとにかく新しいことをやってみたかっただけなんじゃ…？
- 自分の経験から改善を意図していたことは？
  - しっかりした構造（アーキテクチャ）を作ること
  - 管理者の目に品質が見えるようにすること
  - 設計仕様書が目に見える形になればよい
  - ・・・というよりも、理解したことを形に表すことが成功につながった・・・そして、「議論する」土壌を培った

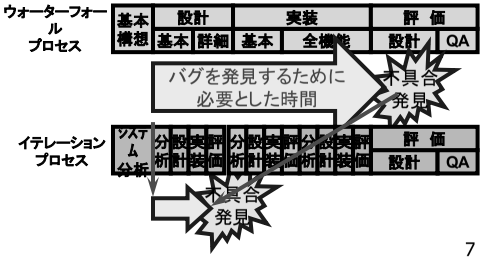
モデル化と反復開発の導入

オブジェクト指向ベースのモデリング ＋反復開発（イテレーション開発プロセス）



- 問題点を早期に発見できる＝リスク管理が容易

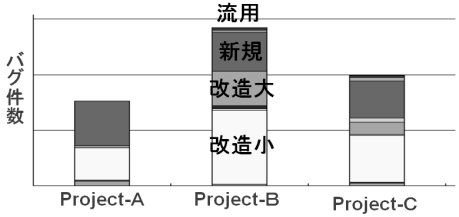
- 設計図（モデル）とコード（製品）が一致
  - 上流での品質確保がやりやすくなる
- ソースコード中心の開発（＝職人による手作り）から、モデル中心の開発（＝誰でも設計図から作れる）へ



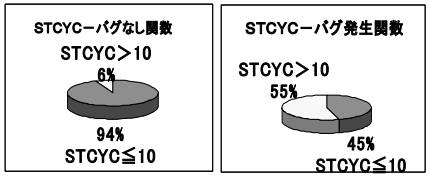
モデル化技術導入の背景

- 抱えていた問題点
  - 仕様変更による2次障害仕様変更の工数
    - 担当者を変更できない
  - ソースコードの再利用ではむしろ品質が悪くなる
    - 手軽で簡単だが全体を知らないと副作用を発生しやすい
    - 技術が開発者を通じてしか伝わらない
      - ＝属人性が高い

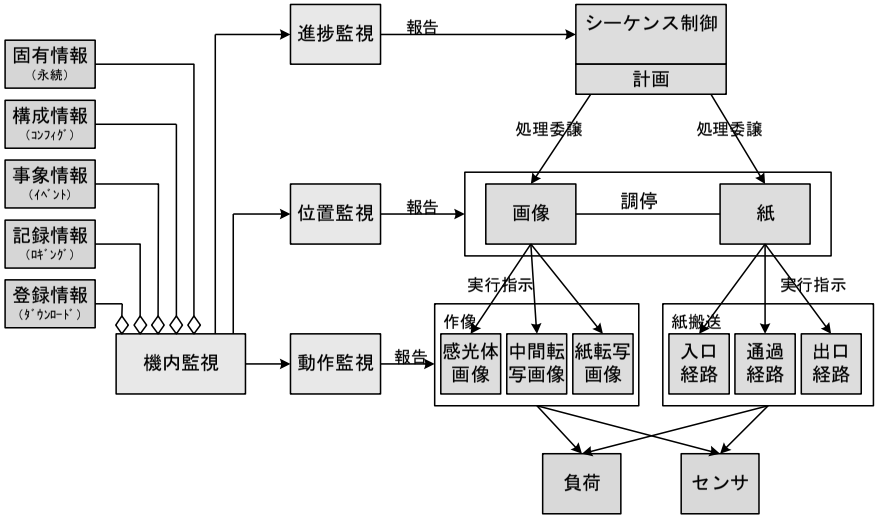
- ソースコードの流用では再利用を繰り返すほど品質が低下する



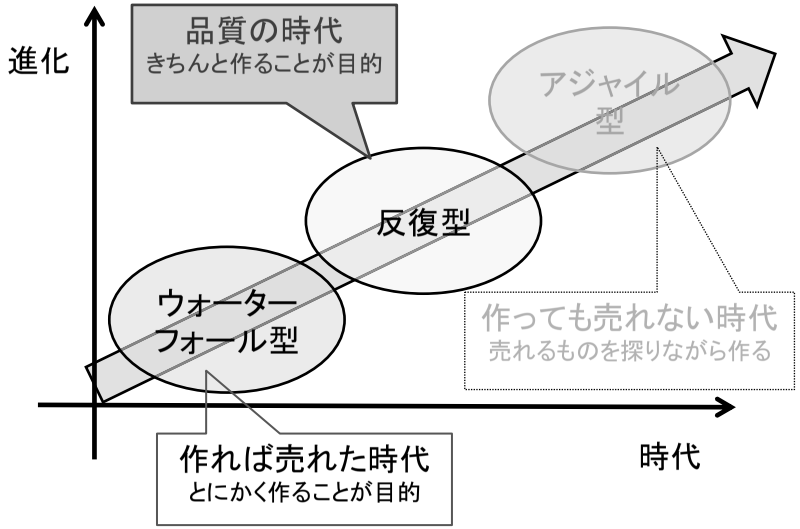
- ソフトウェアの複雑度が増加すると不具合が増加する傾向がある



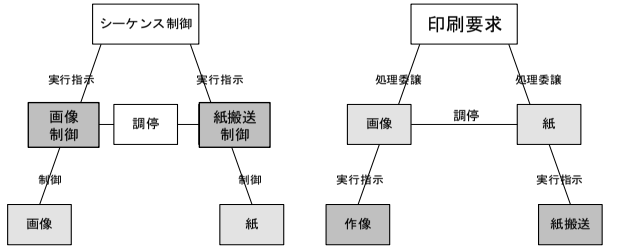
【事例1】カラーMFPの制御ソフトの全体像（初期の分析モデル）



反復開発が時代の要請だった



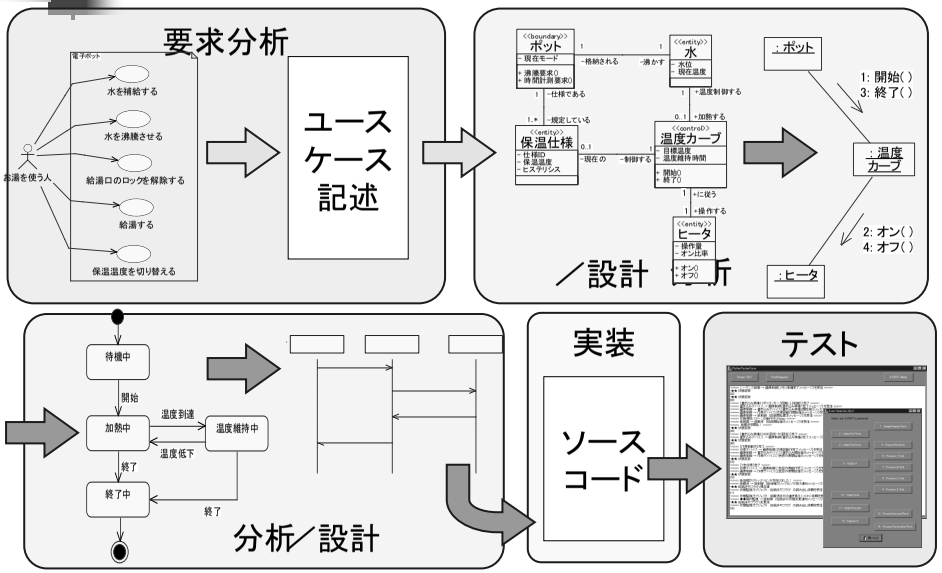
オブジェクト指向によるモデル化でどこが改善されたか（1）



従来手法によるソフトウェアを図式化したもの　　オブジェクト指向アプローチによるモデル開発

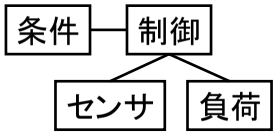
- 左側のような形従来モデルでは，「シーケンス制御」で，「作像」と「紙搬送」へ順次実行処理を発行し，その二つのタイミングを調停するような存在も必要になるので，複雑な構成となる．
- また，このモデルでは，「画像」や「紙」は実行処理の結果として動作するだけなので，モデルとしての意味（責務）は持ち合わせないものとなる．つまり，「画像」や「紙」は制御されるだけの存在となる．
- 従来モデルの問題点は，制御タイミングの処理部分がモデルの中心部分である「作像」と「紙搬送」部分に実装されていることである．すなわち，この部分が変動部となり，変更による修正範囲が広くなってしまうため，品質が劣化するだけでなく，異なる機種への展開が困難になる．つまり再利用性が低い．

モデリングの工程



モデリングは誤解されている

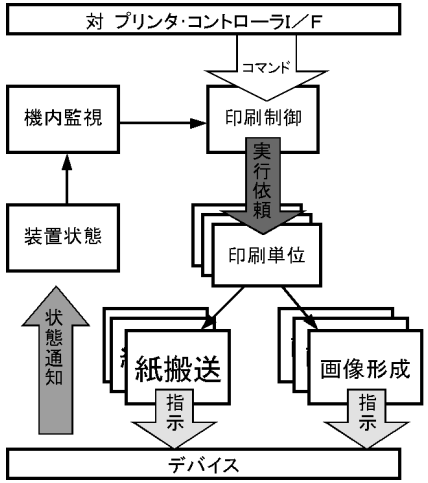
- モデル化でよくある誤解
  - 一般化しようとする
    - 一般化は、モデル化に結果見えてくるもの
      - モデリング時に一般化しようとすると、ドメインの特徴に関係なくどれも同じモデルになる



  - 抽象化の誤解
    - 抽象的な表現でまとめようとする
    - 組込では「制御対象」に着目し、その様子をモデル化するとうまく行く
    - ドメインの特徴を素直にモデリングしたほうがよい

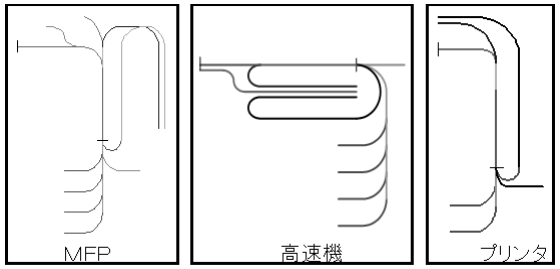
オブジェクト指向によるモデル化でどこが改善されたか（2）

- これに対し，オブジェクト指向的アプローチで作成したモデル（前ページの右側）は，「印刷要求」が「画像」と「紙」に処理を委譲する構造となり，「画像」と「紙」は自分自身の状態を遷移させるために，作像や紙搬送へ実行指示を発行する．「作像」や「紙搬送」は，指示された動作を単純に実行するだけ，という役割分担となる．
- すなわち，前頁右側のモデルは，まずモデルの中心部分が抽象的な概念同士が連携して動作する，という汎用性の高い構造になっており，ちょっとしたタイミングの変更は，「作像」や「紙搬送」というモデルの下位層（知的情報を持ち合わせない）で対応できるため，局所的な修正で済む，という大きな特長がある．これにより，複数の機種にも適用可能なモデル化ができる．つまり再利用性が高い．



【事例3】カラーMFPの紙搬送制御部分の固定変動設計とソフト部品化

- メカの構成、組み合わせ、寸法などのレイアウトは千差万別
  - メカ構成をそのままモデリングするのはNG
  - 物理構造ではなく、論理構造でとらえるべき
- メカ構成の影響を受けるもの
  - 紙詰まり検知の間隔
  - センサの数や位置
  - 両面／後処理加工の種類や機構

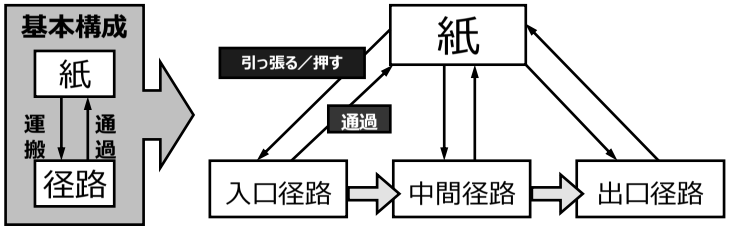


- これらは全て変動部か？
  - やることは毎機種同じ
  - やり方も経験上同じにできる
- → ならばルール化しよう！
  ･･･固定部としてモデリングする

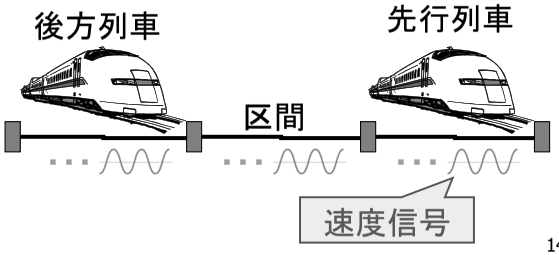
この部分は当日会場でお見せします

紙間制御方式をルール化し、構造的に捉える。

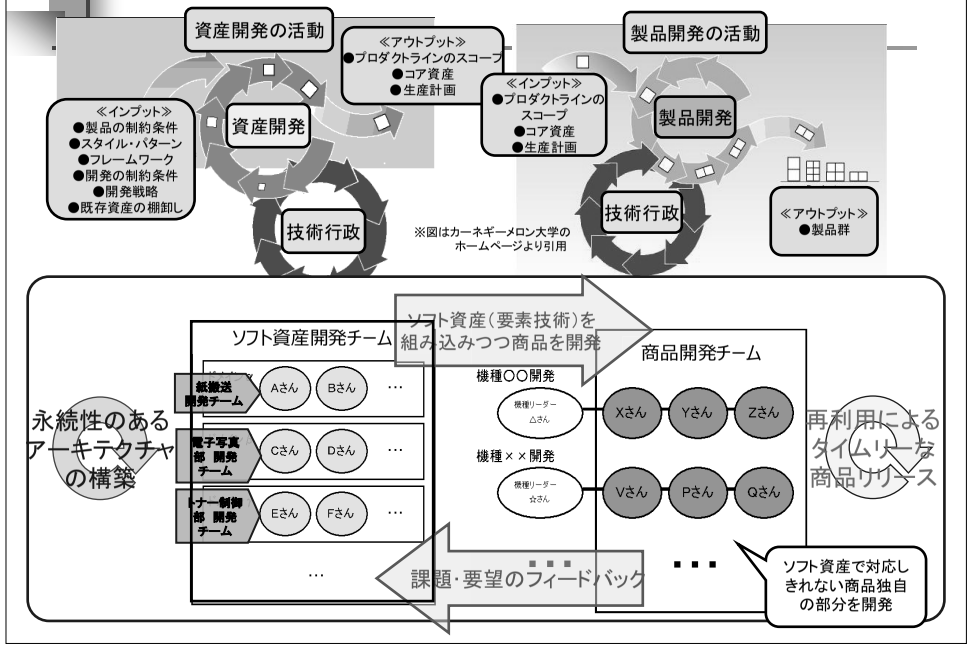
【事例2】カラーMFPの紙搬送制御部分のモデル（初期の分析モデル）



- 概念上の「紙」は「論理的な経路」を運ばれていく過程で自身の状態を変化させる
- 概念上の「紙」は「論理的な経路」１つに対し１枚しか入らない
- １区間に必ず１編成しか入れない新幹線のATCに似ている



【事例6】ソフトウェアプロダクトライン開発の導入



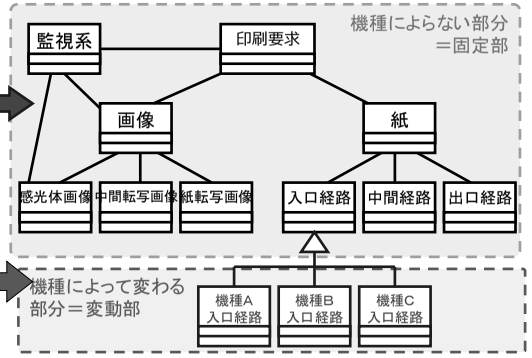
【事例4】固定変動技術を利用した大規模再利用の基本的な考え方

- ユースケースをもとにまず固定部をモデル化し、
- 備考部分もモデル化して機種変動部とする

まず系列部分(「基本系列」「代替系列」「例外系列」)のみでモデル化

UC名	消耗品切れを判定する
概 要	印刷ごとに、××センサーと、補給部の固有情報から、消耗品切れを判定する。
アクター	1. センサー1 2. 操作部 3. 作像部
事前条件	・印刷動作が正常に行われていること
事後条件	・消耗品切れ判定が終了していること
基本系列	1. ***** 2. *****
代替系列	
例外系列	
備 考	

つづいて備考部分を加えてモデル化



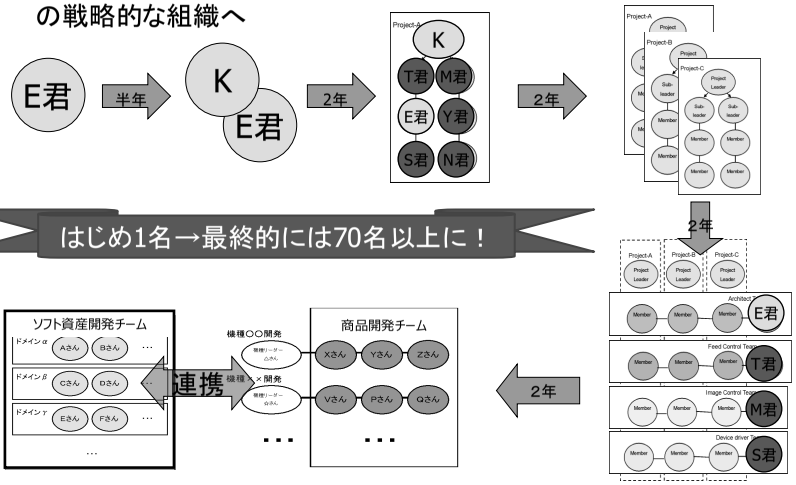
モデル化によるチームの変化

- モデル化によって、設計図面を中心に議論ができる
  - モデル化によって何が変わったか？
    - 誰でも設計内容を理解できるようになった
    - 誰でも議論に参加できるようになった
- チームのレベルが向上した！
  - 個人のスキルに頼らず、チームとしての生産性に自然と目を向けるようになった
  - 高度な平凡性を有する質の高いチームになった

スポーツに例えれば、選手が監督の指示を待ってプレイするようなことが減り・・・いわば選手一人一人がそれぞれの戦術眼を有し、ある程度自分で判断して次のプレイを選択して実行するようになった

モデル化のレベルと開発者の組織化

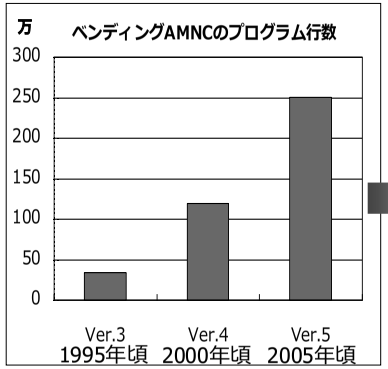
- フレームワークの開発を進めるため、機種開発中心のいわゆる縦割り組織から機能中心の横割り組織へ・・・最終的にはSPLEを実行するための戦略的な組織へ

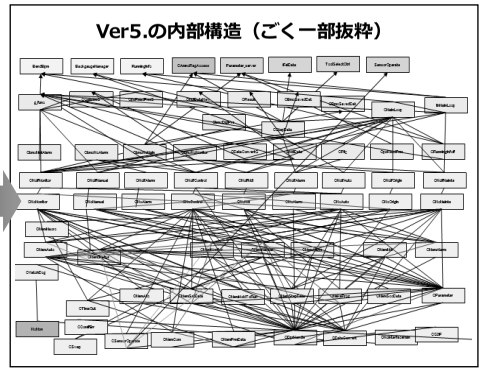


【事例1】板金曲げマシンのコントローラーソフトのモデル化

筑波大学 University of Tsukuba

従来ソフトの問題・・・高機能化・複雑化していくソフトウエア





- プログラムの見通しが悪く、開発・テストに時間がかかるようになっていた
  - １ヶ所の変更が多くの機能に影響を与えるようになっていた

従来の手法では品質が確保できなくなってきた。

異質なモデル化に遭遇したことも

筑波大学 University of Tsukuba

- あるソフト開発チームでの体験
  - せっかくモデル化を行い、プロダクトライン開発の取り組みまで進めていたが・・・
  - 資産となるべきモデルをどう再利用して良いかわからず、商品開発の生産性が大幅に悪化

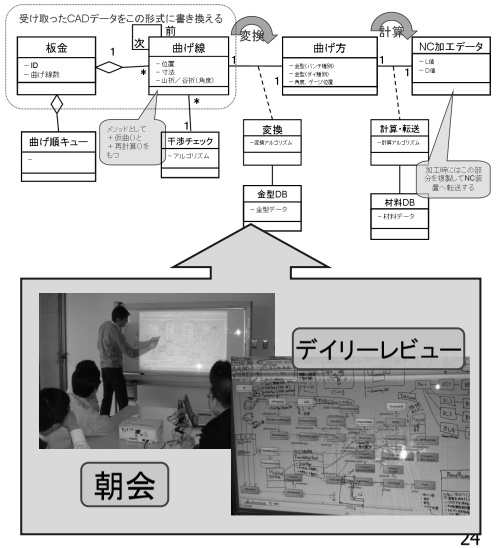
この部分は当日会場でお見せします

間違ったモデル化の事例
- 構造が毎回変わってしまう
  - メンバーが設計図で何を表現すべきかを良くわかっていなかった
- モデル化によって何を達成したいかが不明になってしまった
  - メンバーもモデリングによって成長することができなかった

【事例1】板金曲げマシンのコントローラーソフトをモデル化

筑波大学 University of Tsukuba

- やはり組込系では、制御対象の特徴をいかに記述できる（＝抽象化）か がカギ
  - 従来は曲げ機械コントローラの機能や処理に着目して設計していた
  - モデル化にあたり、対象となるもの＝板金が素の板の状態から、何回か曲げられて製品に仕上がるまでの様子を表現することにした
    - いずれも、「金型と角度を決めて加圧する」ことの繰り返しであることを「究明」
    - これをモデル化することで一気に「設計内容」が見えるようになった



モデル化・・・新たな挑戦

筑波大学 University of Tsukuba

工作機械メーカーのソフト開発部門にてモデル化を導入

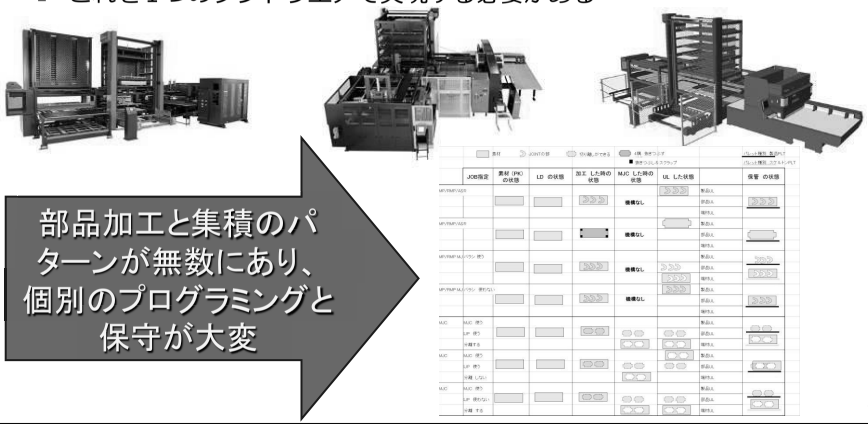
- つねに納期遅れ
- つねに低品質
  - 何か技術や手法を導入しても効果が出ない

- モデル化技術を『道具』としてきちんと設計する風土を作る
  - 一緒にモデリングをしながら、チームの組織化を進める
- 5つのプロジェクトへ技術と手法を順次導入

【事例2】自動倉庫システムの課題

筑波大学 University of Tsukuba

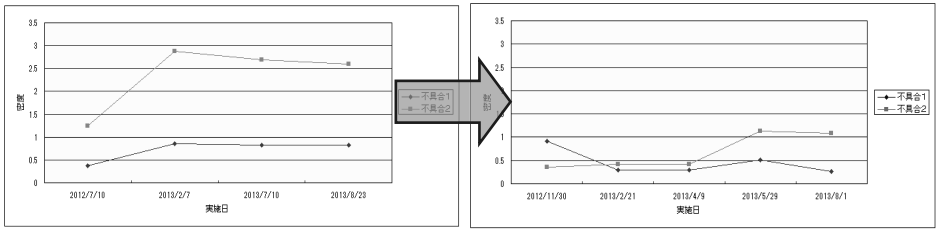
- 板金を倉庫から取り出し、加工を施した後、部品の切り離しを行って集積、再び倉庫へ戻すシステム
- ユーザー先ごとに設備構成が異なる
  - 設備構成が異なると各工程で実施する内容が少しずつ異なる
  - これを１つのソフトウエアで実現する必要がある



【事例1】モデル化と反復開発の適用効果（品質面）

筑波大学 University of Tsukuba

- ソフトウェアの品質が向上
  - 静的指標の変化
    - 適用前のソフト よりどの程度改善されたか、を見ると･･･



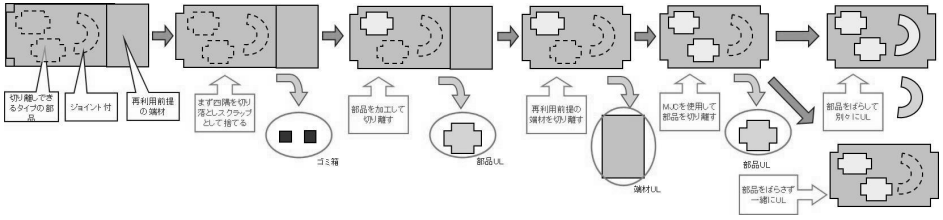
モデル化適用前　　モデル化適用後

品質が大幅に改善されていることがわかる

【事例2】自動倉庫システムのモデル化

筑波大学 University of Tsukuba

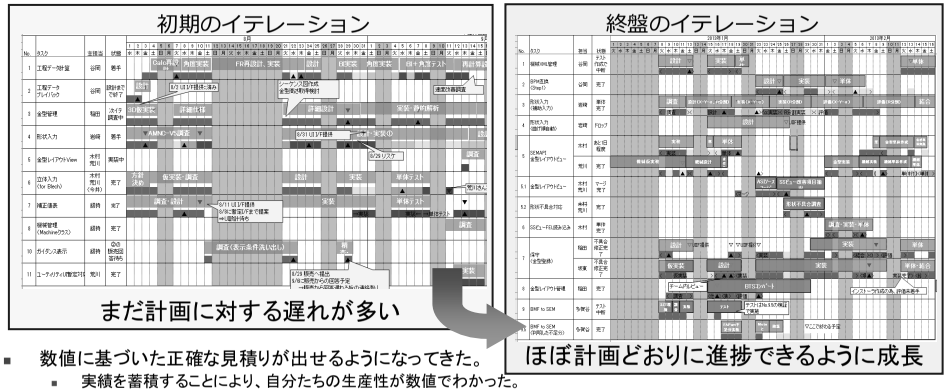
- 膨大な数のパターンに惑わされて、個別の事象にこだわっても特徴は見えない
  - すべての事象を俯瞰するとそこにルールが見えてくる
  - ルールをモデル化する



【事例1】モデル化と反復開発の適用効果（工数面／納期面）

筑波大学 University of Tsukuba

- 見積りの**精度が上がり、リリース日が約束できるようになってきた**



- 数値に基づいた正確な見積りが出せるようになってきた。
  - 実績を蓄積することにより、自分たちの生産性が数値でわかった。
- 突発的な先行リリース要求に対しても、適切な対応ができるようになった。
  - 見積りを根拠に、「代わりに機能A、Bを落とすことで対応可能」などの回答ができるようになってきた。

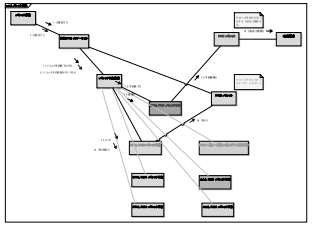
納期遅延が解消　　工数も削減

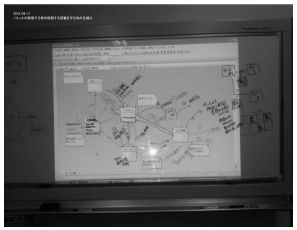
モデル化を契機にチームが成長

本人たちの感想から

筑波大学 University of Tsukuba

- モデル化の効果
  - 要求内容から仕様を早く理解できるようになった
    - 「～がしたい」という内容から、システムが行う内容「～が必要」を考え理解するようになった
  - ソフトウェアの構造を図面化することで、メンバーがソフトウェアの全体を理解できるようになり、検討した内容で意見交換が行えるようになった
    - 図面を見ることで、確認すべき場所や影響範囲がすぐにわかるようになり、スムーズに設計が行えるようになった



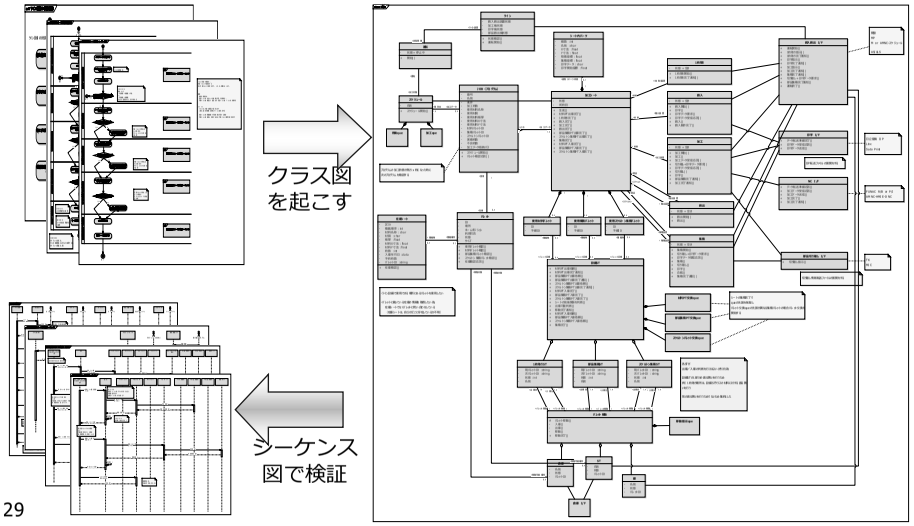


31

【事例2】自動倉庫システムのモデル化（本人たちの報告文書より）

筑波大学 University of Tsukuba

- 試行錯誤しながら作り始めた**設計図**（クラス図 等 ）



29

ソフトウェア資産を活用した戦略的な開発へ

筑波大学 University of Tsukuba

- モデリング（図面化）することによるメリット
  - ソフトウェアの構造が目に見える
    - アーキテクチャを崩さずに機能追加できる
    - アーキテクチャの再利用で品質が安定する
  - 設計図（＝モデル図）とソースコードのひも付けができる
    - ソフトウェアを図面をもとにつくれる
    - 図面からソースコードを（半自動的に）生成できる
      - 上流の検討に工数を割くことができる
  - モデルの共有効果
    - モデル図で変更点と影響範囲を全員が特定可能
    - いままでフルテストまたはノーテストだったものが、適切なテスト計画を策定して実施できる
      - 商品設計の工数が75分の1になった例もあった

32

【事例2】モデル化による自動倉庫システムの課題解決

筑波大学 University of Tsukuba

この部分は当日会場でお見せします

30

JEITA OPEN PROBLEM スマートガレージシステム

【概要】

- 自動で開閉するガレージ（基本機能）
- 便利機能や安全機能、防犯機能などの各種の機能（スマート機能）を持つスマートなガレージ

【要求や背景】

- 雨天時にクルマから降りてガレージドアを開閉したくはない
- ガレージドアを開けたまま外出するのは、防犯上、好ましいことでは無い
- ・・・

【機能の例】

- 自動開閉
  - クルマが:
    - ガレージに近づいたらドアが開く
    - ガレージから遠ざかったらドアが閉じる
    - ガレージから出庫しようとしたらドアが開く
    - ガレージに入庫したらドアが閉じる
  - 人がいないところでのドアの開閉
    - 安全に対する配慮

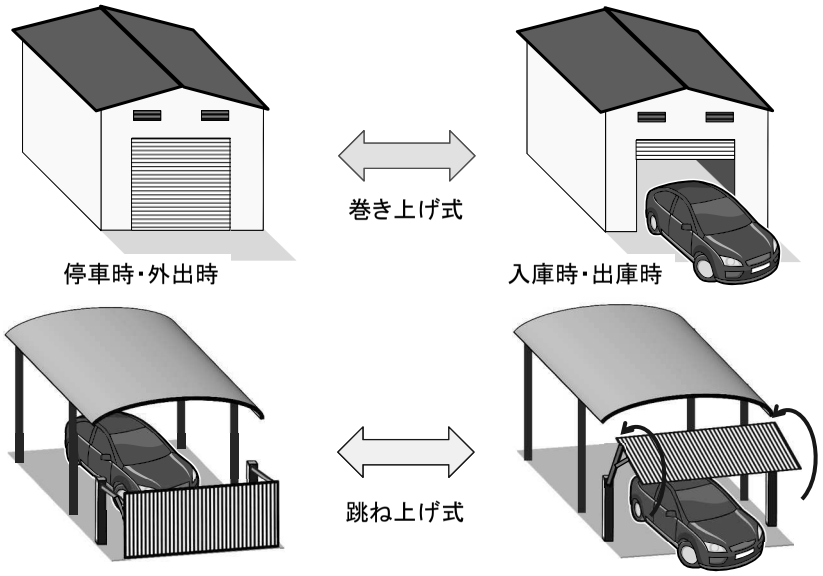
様々な検知手段を用意、B.T.O.対応が可能

JEITA EMBEDDED SOFTWARE WORKSHOP 2014
MODELING WORKSHOP
JEITA 組込み系ソフトウェア・ワークショップ 2014
モデリングワークショップ

JEITA OPEN PROBLEM “SMART GARAGE”

JEITA 公開問題「スマートガレージ」

ガレージの方式別のイメージ



JEITA OPEN PROBLEM スマートガレージシステム

【ビジネス戦略】

自社の主力製品である、ガレージ用シャッターを用いたシステムを開発・販売することになった。そこで、既存製品を生かし、新たな顧客価値を付加できるシステムを目指し、ガレージシステムの中核となるコントローラーを開発する。

【ビジネス上の制約】

ガレージとコントローラー以外は調達品を用いて構成することとし、出来るだけ汎用品を用いる。長期利用が一般的と考えられ、保守性、保守部品の確保などの観点からも、インデント開発はせず、カスタマイズが容易な1製品を開発する。この1製品で出来るだけ多くの顧客ニーズに対応し、将来の拡張にも対応できるものを目指す。

スマートガレージコントローラー モデリングのヒント

- 車が接近した時の検知方法はいくつかの手段がある
- 誰でもガレージを使えるワケでは無い
- 安全が脅かされる危険にはどんなものがあるか
- 各種のB.T.O. に対応するにはどうするのか
- 何をモデリングするのか明確にする(ビジネス、要求、アーキテクチャ、・・・)
- 図の形式は自由。但し図の意図が伝わってこそのモデリング
- ・・・

禁 無 断 転 載

平成26年度 ソフトウェアに関する調査報告書 Ⅱ

組込み系ソフトウェア開発の課題分析と提言
～モデリングはなぜ失敗するのか～

発　行　日　　平成27年３月
編集・発行　　一般社団法人　　電子情報技術産業協会
ソフトウェア事業委員会
〒100-0004　東京都千代田区大手町1－1－3
大手センタービル
TEL　(03)5218-1057
印　　刷　　三 協 印 刷 株 式 会 社