ISSN 1884-0760

先端ソフトウェア工学に関するGrace 国際シンポジウム

形式手法の産業応用ワークショップ 2010

WIAFM2010: Workshop on Industrial Applications of Formal Methods

予稿集

2010 年 3 月 15 日　国立情報学研究所

Grace-TR-2010-03

CENTER FOR GLOBAL RESEARCH IN
ADVANCED SOFTWARE SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF INFORMATICS
2-1-2 HITOTSUBASHI, CHIYODA-KU, TOKYO, JAPAN

WWW page: http://grace-center.jp/

The GRACE technical reports are published as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

はじめに

ソフトウェアが社会の様々な製品やサービスに不可欠なものとなっている現在、ソフトウェアの不具合に起因するシステム障害は時として社会に広範囲かつ深刻な影響を与えています。こうした背景からソフトウェアの信頼性向上への要求がさらに求められつつあります。一方、海外においては、数学的基盤に基づいて論理的に検証しながらソフトウェアを開発する形式手法についての研究開発が長年にわたり実施されており、実問題への適用が本格的に始まっています。日本においては、開発現場への普及が始まっているとはまだいえない状況にありますが、一部の開発現場においては適用が試みられ、人材育成の取り組みも始められています。

本ワークショップは、形式手法に関する研究開発、適用実践、人材育成に取り組んでいる国内の研究者や技術者が一堂に会し、相互の情報交流や連携の機会の場を提供することを目的としています。

今回のワークショップでは形式手法の適用実践や人材育成を先駆的に取り組まれている有識者をゲストに迎え、各活動の現状や今後の構想について語っていただきます。また、形式手法の実践・教育に係る具体事例について、専門家にご紹介いただくとともに、形式手法の教育についてパネルディスカッションを開催し、企業における形式手法の教育の現状と課題について議論をいたします。

事前の参加登録者数は当初の予想を大幅に超え、100 名近くの登録がありました。形式手法への関心と期待が高まっていることの証左といえます。この機運をさらに盛り上げるため、来年以降もワークショップを開催する予定です。

実行委員長: 粂野文洋 （三菱総合研究所/国立情報学研究所）
実行委員:來間啓伸（日立製作所/国立情報学研究所）、石川冬樹（国立情報学研究所）、田口研治（国立情報学研究所）

形式手法の産業応用ワークショップ 2010　プログラム

日時、場所:2010 年 3 月 15 日　国立情報学研究所

Session 1 (10:20- 11:50) 形式手法の展望

IPA/SEC における形式手法に関する活動状況と課題

山本　修一郎、名古屋大学 情報連携統括本部 情報戦略室　教授

システム検証研究センターの七年間

木下佳樹、産業技術総合研究所 システム検証研究センター

形式手法の知識体系 FMBoK (Formal Methods Body of Knowledge) 構築に向けて

田口研治、国立情報学研究所　GRACE センター

Session 2 (14:20-15.50) 形式手法の実践・教育

モデル検査知識体系 MCBOK2008 について

西原秀明、産業技術総合研究所 組込みシステム技術連携研究体

IC カードセキュリティ開発における形式手法 SPIN の適用（Smartcard security development using formal method tool SPIN）

市原尚久、(株) NTT データ　技術開発本部 SI アーキテクチャ開発センタ

ソフトウェア開発における形式手法導入に関する課題と解決アプローチ

石黒 正揮、(株)三菱総合研究所　情報技術研究センター　情報セキュリティ研究グループ

Session 3 (16:10-17:40) パネル　形式手法教育の現状と課題

企業内で形式手法教育を実践されている方々や研修セミナー・教育プログラム等で社会人を対象とした形式手法教育を実践されている方々をパネラとしてお招きし、各々の形式手法教育の現状についてご紹介いただくとともに、形式手法が開発現場に普及するための課題について人材育成の観点で議論していただきます。

パネラ：小川清（名古屋市工業研究所)、劉少英（法政大)、青木利晃（ＪＡＩＳＴ)、來間啓伸（ＮＩＩ/日立)、石川冬樹（ＮＩＩ）

IPA/SEC における形式手法に関する活動状況と課題

山本修一郎†

アブストラクト
IPA/SEC では，2007 年に高信頼性システム技術に関する調査検討会を発足させ，これまで 3 年にわたって形式手法の産業界での活用について産学官での議論を続けてきた．本稿では形式手法に関するこれまでの活動状況を紹介するとともに今後の課題について述べる．

Current status and future issues on Formal Method activities of IPA SEC

Shuichiro Yamamoto†

Abstract
IPA/SEC established an investigation group on the dependable system development technology in 2007. We have continuously discussed how to promote industrial application of Formal Method these 3 years in collaboration with members from Japanese industry, academia and SEC. In this article, I will introduce an overview of the activity and future issues to deploy Formal Method into Japanese industries.

1. はじめに

高信頼性システム技術調査検討会(DSTWG, Dependable System Technology Working Group)は IPA ソフトウェアエンジニアリングセンター（SEC）の活動グループの一つとして 2007 年に発足した．この DSTWG では社会に浸透して重要インフラになった情報システムが引き起こすシステム障害が社会的に大きな影響を与えるため，重要インフラシステムを高信頼化するための技術を調査するために 2007 年度は海外ならびに国内における形式手法の導入状況を調査した．

この結果，米国ならびに欧州では形式手法の産業応用を積極的に支援するために，多数の研究機関やユーザ企業が参加する多くの国家プロジェクトを進めていることが明らかになってきた．このため，DSTWG では 2008 年から 2009 年にかけて，我が国の情報システム開発に形式手法を導入するためにはどのような課題があり，それをどのようにして解決していくべきかについて議論を継続してきた．このような議論を受けて 2009 年末には DSTWG を発展させることで, 形式手法導入プロセス実証評価 WG が発足することになった．

以下では DSTWG における活動状況と主な調査結果をまず紹介する．次いで高信頼性システム技術としての形式手法の産業応用についての課題認識について述べる．最後に形式手法の導入プロセスの実証の必要性と，その進め方について提言する．

†名古屋大学 情報連携統括本部 情報戦略室
†Information and Communication Planning Office, Headquarter of Information and Communication Services at Nagoya University

2. 活動状況

2.1 体制とイベント

2007 年度 は産 7 名，学 5 名，官 6 名の委員が参加して調査検討会を 7 回開催した． また，高信頼性システム開発手法フォーラム を開催するとともに，「高信頼ソフトウェア構築技術に関する動向調査報告書」を公開した．

2008 年度は産 13 名，学 5 名，官 5 名の委員が参加して，6 回の高信頼性システム技術 WG を開催した．また，日中高信頼性システム検討会ということで中国の形式手法の研究者らと意見交換した．また JAXA と IPA-SEC 共催で，第 7 回 WOCS-Workshop Of Critical Software を実施した．

2009 年度も継続して 6 回の高信頼性システム技術 WG を産 16 名，学 7 名，官 8 名の委員により開催した．当初 7 名の参加であった産からの委員が倍増していることからも産側の期待が大きいことが分かる．また 3 回のプロジェクトチーム会議を組織して形式手法の導入プロセスの実証に向けた準備検討を実施した．さらに「高信頼システム開発技術の動向」について書籍化を予定している． また，ソフトウェアジャパン 2010 では，「ソフトウェア開発のパラダイム・チェンジ in マインド ーIPA/SEC における新しい取組みー 」の一つとして形式手法の取り組みを紹介している．

2.2 主な調査結果

形式手法の産業界への導入事例についての主な結果

を表 1 に示す[1]．黒字で示したのは海外事例，赤字で示したのは国内事例である．この表から多様な形式手法が多方面で導入されていることと，その導入形態も一様ではなく多様であるということが分かる．

表 1　主な形式手法の導入事例

	VDM, Z	B, Event-B	SPIN	SMV
航空			テスト	設計，テスト
宇宙		探査衛星仕様	要求，設計	
原子力		設計		
鉄道		仕様，設計，実装		状態設計
船舶			設計，テスト	
公共		設計，実装		
医療	設計仕様記述			
社内システム		SOA仕様	HMIイベント	HMIタイミング
セキュリティ	状態設計検証	運用設定設計		
OS		カーネルテスト		
ICカード			AP設計	
携帯電話	ICチップファームウェア仕様	プラットフォーム設計		
電力				設計，テスト
組込機器		仕様	テスト	設計

3.　課題

3.1　大規模化の影響

JUAS の調査[2]によれば，システム規模が増大するとともに，開発予算が超過し品質が不十分になる．たとえば500 人月以上のプロジェクトでは50%以上が予算超過し約 40%が品質不十分だとされている．これに対して 100 人月未満のプロジェクトでは，品質不十分は 20%以下，予算超過も 10%程度だとされる．この結果が示唆することは，大規模システムの分解の必要性である．システムを分解するためにはアーキテクチャ設計が重要になる．

3.2　重要インフラ障害の原因

経済産業省の報告書[3]によれば，重要インフラ情報システム障害の原因は，開発 21%，保守 39%，運用 40%となっている． つまり，ソフトウェア開発時のバグをいくら削減しても重要インフラ障害の 80%はなくならないということである．したがってソフトウェア開発時に運用と保守を考慮することが必要になる．すなわち運用時や保守時に運用者や保守者によってソフトウェアがどのように扱われるかを考慮した仕様や設計が求められる．

3.3　ディペンダビリティ

JIS　Z　8115 では「アベイラビリティ性能及びこれに影響を与える要因，すなわち信頼性性能，保全性性能及び保全支援能力を記述するために用いられる包括的な用語」としてディペンダビリティが定義されている．また，アベイラビリティは，「要求された外部資源が用意されたと仮定したとき，アイテムが与えられた条件で与えられた時点，または期間中，要求機能を実行できる状態にある能力」として定義されている．したがって，外部資源のもとでソフトウェアに対して要求される機能を実現する性質としてディペンダビリティを記述することが求められる．

4.　形式手法の導入プロセス

4.1　必要性

DSTWG のこれまでの議論から明らかになったことは次のようである．実際のソフトウェア開発に形式手法を導入するためには，現場に導入するための実践的なプロセスを構築する必要があり，この導入プロセスは，形式手法を導入しようとする対象プロジェクトの置かれた状況, つまり開発プロセス, 担当者のスキル, 開発環境, プロジェクトリーダの意識などに依存する．優れた形式手法やその自動化ツールができたからそれを現場ですぐに導入すればいいということにはならない．形式手法には長所もあるが限界もあり，それをどのように個々の開発現場のプロセスに調整・調和させていくかということが「導入プロセス」に込められた意味合いだと理解している．

高信頼性システム技術 WG では 2007 年の発足当時から, 形式手法を産業界に導入するためには, 技術面, 普及面，教育面の取り組みが必要であり，それぞれ次のように考えていた．

技術面では，高信頼性開発技術フレームワークの策定と開発ライフサイクルに対する統合的開発手法の構築が必要になる．普及面では，実証研究に基づく成功事例蓄積と知識の体系化, 国内研究機関との連携 ，国際連携の取り組みが必要になる．教育面では，教材開発とスキル認定についての検討が必要になる．

今回，形式手法の導入プロセス実証評価 WG とともに形式手法の教育普及 WG が発足したのは，実証評価と教育普及が両輪で相補的に活動していくことが大切だと考えられたからであり，この方針は当初から変わっていない．

4.2　導入プロセスの構成要素

前述したことから，形式手法の導入プロセスでは次のような方法を具体的に明らかにする必要がある．

①ソフトウェアをコンポーネントに分解して仕様を記述する方法

②運用環境や保守環境でソフトウェアが外部とどのような相互作用をするのかを記述する方法

③仕様が満たすべき性質を記述する方法

④形式的に仕様を記述する方法

⑤仕様が性質を満たすことを模擬的に確認できる方法

⑥仕様が性質を満たすことを数学的に保証できること
⑦仕様をコードが実現する追跡性を確認できること
⑧上記の方法を開発プロセスに統合すること

これらの項目のうち③④⑤⑥は従来の形式手法でもよく知られている．①は大規模化するソフトウェアに対してアーキテクチャ記述と親和性の高い形式手法を確立するために必要である．②は重要インフラ障害を防ぐために必要である．⑦は仕様とコードの一貫性を保証するために必要になる．⑧は形式手法を実際の開発プロセスに組込むために必要になる．たとえば，次のような早期テスト駆動型 V 字モデル[4]として開発プロセスを確立する必要がある．

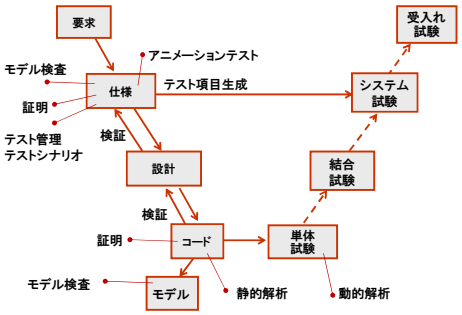


図１　早期テスト駆動型 V 字モデル

4.3　エンジニアリングケース

調査結果で示したように，形式手法の導入は多様な分野で急速に拡大しつつある．このため前述した導入プロセスを適用対象システムごとに異なる方法で記録したのでは，多様な導入事例が蓄積できたとしても，円滑な活用が困難になることが予想される．このため次のような形式手法をなぜ，どのように導入したのかという判断根拠を記述するための共通パターンを提案している．

①情勢変化

要求モデル，アーキテクチャ記述言語，形式手法を採用するに至った開発・技術の情勢を記述

②IT 変容の期待

要求モデル，アーキテクチャ記述言語，形式手法が対象とした IT システム変容への期待を記述

③技術課題

要求モデル，アーキテクチャ記述言語，形式手法が達成しようとしたシステム高信頼化への貢献と解決した技術課題を記述

④記述内容

構成要素，構成要素間関係，評価特性を記述

⑤記述手順

要求モデル，アーキテクチャ記述言語，形式手法の作成プロセスを記述

⑥実施組織

要求モデル，アーキテクチャ記述言語，形式手法を用いた開発組織，適用組織などを記述

⑦取り組み

要求モデル，アーキテクチャ記述言語，形式手法を記述，適用，教育するためのプロジェクト管理面での具体的な取り組みを記述

この①から⑦からなる共通パターンの記述をエンジニアリングケースと呼ぶ．エンジニアリングケースでは各プロジェクトの置かれた状況で，形式手法の導入に当たってどのような課題があり，それをどのように解決したかを記録するため，具体的であるから容易に理解できる．しかし，一般化されていないから，同じような状況にならなければ使えないという問題がある．また，一般的な手法を具体的な現場の状況に合わせてカスタマイズすることは非常に困難な問題でもある．具体的な導入パターンの蓄積とその一般化についての研究を期待したい．

5.　まとめ

本稿では，IPA/SEC における形式手法に関する取り組みについて述べた．また産業界から期待される形式手法の導入プロセスの実証的な取り組みを進める上での課題と，そのために必要となる導入プロセスの共通的なパターン言語としてエンジニアリングケースの構成を提案した．

参考文献

1) 高信頼ソフトウェア構築技術に関する動向調査報告書, http://sec.ipa.go.jp/reports/20080606.html
2) JUAS，企業 IT 動向調査 2007
3) 経済産業省，重要インフラ情報システム信頼性研究会　報告書，2009, IPA
4) ROBERT M. HIERONS, et.al., **Using Formal Specifications to Support Testing,** ACM Computing Surveys, Vol. 41, No. 2, pp. 9-76, 2009

システム検証研究センターの七年間†

産業技術総合研究所　システム検証研究センター長
木下佳樹

アブストラクト
産業技術総合研究所（産総研）システム検証研究センター（CVS）の前身システム検証研究ラボ(LVS)が設置されたのは 2003 年 4 月でした。一年後にシステム検証研究センターに改組し、以後六年間、研究センターとして、システム検証に関する研究活動と技術移転活動を行ってまいりました。産総研では、各研究センター・研究部門の長は毎年の仕事の内容について、理事長と「契約」を交わし、その内容を「方針声明(policy statement)」として公にすることとなっております。ここでは、その中の主な項目を拾い、CVS の七年間にわたる活動をふりかえってみたいと存じます。

1.　目標

CVS 活動の目標は、

> システム検証の数理的技法が情報処理システム開発における生産性および信頼性の向上に有効であることを産業界に説得し、数理的技法を普通の検証法とすること

です。平たくいうと「システム検証の数理的技法」の技術移転を根本の目標としたのですが、研究所として当然のこととして、実用化研究のみならず応用研究や基礎研究にわたる研究活動も CVS で行なってきました。しかし、CVS では全体としては、技術移転を通して産業界から一般社会と相互に作用する活動を行うということを、この目標を掲げることによって宣言したのです。

CVS を組織するにあたっては、情報システムそのものに関する科学技術分野の研究、技術開発、教育などが立ち遅れている、との観測があったことを申し上げておきたいと思います。研究センター発足の後、２００５年には日本経済団体連合会の意見書「産学官連携による高度な情報通信人材の育成強化に向けて」が出され、文部科学省の先導的 IT スペシャリスト育成プログラムが開始されるなど、この立ち遅れが社会的に広く認知されるに至りました。２００8 年には国立情報学研究所に先端ソフトウェア工学･国際研究センターが設置されています。しかし、CVS が発足した時は、情報システムのトラブルが 2010 年の今日ほど顕在化していませんでした。ソフトウェアトラブルに起因する携帯電話の回収が大きくクローズアップされるなど、まだまだ、情報システムトラブルは珍しい現象として扱われた時代でした。CVS の設置は、時代を若干先取りした、時宜に合ったものだったと自負しております。

†本稿は、システム検証研究センター最終報告会（2010.3.18）の予稿として執筆したものに体裁上の修正を加えたものである。

2.　数理的技法

数理的技法という研究対象についても説明しておこうと思います。近年ようやく、形式技法（形式手法、formal methods）が開発現場で本格的に考慮されるようになってきています。いっぽう、システムの仕様などを機械処理可能な形式言語で記述する「完全な」(full) 形式技法だけでなく、背景に数理科学的なモデルを持ちつつも、仕様などの記述自体は英語や日本語で書き、記述の機械処理を必ずしも可能とはしない、という半形式技法（semi-formal methods）、あるいはモデル指向アプローチ（model based approach）もあります。我々は、数理的技法という言葉を形式技法とモデル指向アプローチを包含するものとして用いています。そこで、数理的技法は情報システム設計に関する唯一の科学技術であるといってよいでしょう。逆にいうと、情報システム設計の科学技術であれば、それはきっと何らかの意味で数学を使うのでしょうから、数理的技法と呼べるものです。形式技法という言葉は、何か特定の技術であるかのような印象を与え、事実、昨今ではそのような誤解が広まっているように感じられるのですが、実はそうではありません。

さて、ここで科学技術がどうこうと云々するのは、情報システム開発において、科学技術と呼べるほどのもの、客観的事実と推論の積み重ねによって伝えられる技術体系が、未だ殆どないからです。大規模な情報システムの構築法としては、単なるルールの集積として先輩から後輩に見よう見まねで伝えられる職人芸的なノウハウがいくらかあるだけではないでしょうか。その結果、技術の伝承も口伝でしかおこなえず、技術者の急増が必要な現実に対応できないでいるように思われます。

組込みシステムや業務システムをはじめとする情報システムの開発現場では、大規模システム全体を見通すことが困難であるために深刻な問題が多数発生しています。これは、周知のとおりです。この状況は1970年代にいわれたsoftware crisisと本質的に同じものですから、これらの問題が一朝一夕に解決するとは思われません。数理科学を用いて、職人芸としてではなく、科学技術として情報システムの開発技術を確立することによって、この本質的問題に立ち向かおう、というのが我々の立場です。

3. 科学研究とフィールドワーク

CVS はシステム検証の数理的技法を産業界に移転することを目標として組織されましたが、この分野の研究が進んでいないので、科学研究も誰かが行わなければなりません。数理的技法は、まだまだこれから研究が必要な分野でありまして、いわゆる「枯れた分野」、十分に研究がされつくした、といった分野ではありません。一方、技術移転に優秀な研究者が携わるのは大いに望ましいことですが、研究者の意欲をかきたてる環境を提供するためには、技術移転だけを行なう組織にするわけにはいきません。そこでCVSでは科学研究と技術移転を両方行なうことを目指しました。

いわゆるwaterfall式の考えで研究活動をみると、科学研究と技術移転の間には随分いろいろな活動が必要です。科学研究の成果が実用化されるとしてもそれには数十年かかるのが普通ですから、CVS設置期間の六年間の間に技術移転まで辿り着くのは不可能です。それでは科学研究に従事するものは技術移転と無縁でいなければならないのでしょうか?

科学研究に従事するものが、同時に技術移転に携わることにしよう、と我々は考えました。これは一つの挑戦でした。しかし大学では既に似たことが起こっています。大学では第一線の研究者である教授が学生に学術を「移転」しています。

ここでよく考えてみると、大学では、研究の対象をそのまま学生に教えるとは限りません。カリキュラムを作って、学部の学生には初歩からより進んだ知識へと順序立てて学術を伝授することになっています。これと同じように科学研究の対象とは別に、技術移転の対象を考え、それを順序だてて移転していくことにすればよいのではないかと考えました。つまり、われわれは

自らの科学研究の対象をそのまま技術移転することにはこだわらない

ことにしたのです。自分の研究対象に限らず、自分の研究分野の周辺の研究成果であっても、その技術を移転することが適当なのであればその技術の移転を行うことにしました。我々は自分の研究分野の周辺のことをも勉強してよく知っているわけでして、これは研究者がもつ強みであるはずです。

考えてみればこれは当然の話です。科学研究は、科学の世界における価値観にしたがって行なわれるのですが、その価値観は産業界や一般社会の価値観と同じだとは限りません。従って、科学研究の結果を移転できるような技術に仕上げるには、そのために特別の仕事が必要です。これはCVS設立前後の産総研草創期に当時の吉川弘之理事長を中心にして行なわれた「本格研究」の議論におきまして、吉川氏らが「第2種の基礎研究」という考えで主張されたことでもありました。いろいろな世界がそこに固有の価値観を持つのだという考えは、民族学や大乗仏教などにも影響されているかもしれません。吉川氏も、第2種の基礎研究の説明に、民族学者のLévi-Straussによる“bricoleur（便利屋、日曜大工）”の考えを引用しておられたのは興味深いことです。

このように、技術移転のためには、技術の移転先の世界での価値観をよく知るべきですが、そのこと自身がまた、研究に近い活動です。技術の移転先の世界をわれわれのフィールドだと考えれば、この活動こそフィールドワークと呼ぶのがふさわしいでしょう。図らずも、ここでも民族学から用語を借用することになりました。

科学研究だけ、技術移転だけでも大変なのに、両方やるのは不可能ではないか、との疑問もありました。しかし、大学で起こっていることを見ると、これにはすぐに肯定的な答えが出ました。産総研には学生がいないのだから、代わりに技術者を対象にした技術の移転をすることも不可能ではないはずではないでしょうか。大学では、教育に関する仕事をよく、educational dutyといっています。そこで、いくつかの大学での講演で

industrial duty instead of educational duty

などといってみたら、教授連から喝采、若手研究者からはブーイングを得たこともあります。

ともあれ、CVSでは科学研究とフィールドワークを分業によって行なうのではなく、各研究員に両方に携わることを求めようとした。これには二つの意図がありました。一つは、最新の研究成果あるいは成果そのものでなくとも、研究の第一線の考え方を持ち込むことによって、単なる技術移転を超えて、より効果的な技術を生む可能性を求めました。もう一つは、逆に、

技術移転の活動の中から、新たな科学研究の対象を見出す可能性を求めました。つまり

科学研究とフィールドワークの相互作用

を求めたのです。研究成果を産業や社会に適用することはよくいわれるが、逆の作用はあまり強調されません。

このことについては、筆者が 1990 年代初頭にエディンバラ大学滞在中に聴いた Robin Milner 教授の lab lunch における談話が大きく影響しています。dti (department of trade and industry)を訪ねた機会に、「実用化の議論ではしばしば、産業界の価値観が科学の価値観よりも優先されることが暗黙のうちに仮定されるが、逆に考えることできるよ」という話をしてきた、というのがその内容でした。産業界の価値観と学界の価値観は、確かに違うものです。しかし、どちらかが絶対的に優先されるべきものというわけではありません。対等の価値観が二つ並んでいると考えたらどうか、というわけです。簡単な話ですが、蒙を開かれる思いがしたものです。

各研究員に科学研究とフィールドワークの両方に携わってもらうという当初の考えは、しかし、実務的理由から簡単に崩れてしまいました。プロジェクト費用から研究員の人件費を出している場合もあり、その場合には当然ながらプロジェクトに専念することが求められるからです。そこで、そのような場合は例外としましたが、その場合でも、例えばフィールドワークのプロジェクトに携わっている研究員に対しても、科学研究をおこなうことをできるだけ援助することにしました。

両方に携わるに当たって、具体的にどのようにすべきかの指針を与えることはできませんでした。しかし両者への労力配分は、

四分六の原則

つまり、フィールドワークに四分、科学研究に六分くらいの労力配分でやってほしいという原則を作りました。これにはいろいろな反応がありました。産総研の本拠地つくばで有名な四六の蝦蟇にちなんだわけでもないのですが、英語でこれを講演したときに、4:6 と書いてみたら、それは約分して 2:3 とすべきだろう、という意見が出てびっくりしたこともあります。四分六という言葉の語呂が英語では消えてしまうわけで、わざわざそのことを説明する羽目になりました。フィールドワークに四分の労力しか割かないのは少なすぎるのではないかいう管理側からの意見もありました。しかし、労力が４：６でも成果は７：３か 8:２くらいになるのです。それくらいに科学研究にかけなければならない労力が大きいのだ、ということがなかなかわかってもらえないのは残念でした。４：６なのか３：７なのかどのようにして測るのか、という生真面目な質問に対しては、それはどうでもよくて、科学研究に割く労力をちょっぴり多めにしてくれというメッセージだ、というような説明をしたものです。

4. コミュニケーション

コミュニケーションを重視する、と方針声明の中で謳いました。研究活動において、同僚研究者との議論は大きな役割を果たします。正確なコミュニケーションのためには読み書きによるのがよいでしょうが、考えを交換し、自分の考えをさらに進めていくためには、会話によるコミュニケーションがより便利です。研究に関する会話が研究センター内で日常的に行なわれるのは当たり前に聞こえるかもしれませんが、実際には、各研究者が個室に閉じこもってしまって全く会話のない研究所はあちこちにあります。一人で研究が進められるのなら研究所は要らないかもしれません。会話によって研究を進めるような研究集団を作ることを我々は望みました。そのために研究室の配置を工夫したり、一人の研究員が複数のプロジェクトに参加することを奨励したりしました。

5. 共通知識基盤

情報システム検証や開発の数理的技法という分野は、わが国では殆ど講座がないばかりか、その研究活動に必要な数理論理学、抽象代数学、プログラミング言語などにまたがった知識がわが国の大学ではあまり教えられていません。その結果、CVS には論理学、数学、ソフトウェア工学など、異なる背景知識をもつ研究者が集まることとなりました。背景の違う研究者が会話によるコミュニケーションによって研究を進めるには、考えを表現する基盤として共通のものが必要になるはずです。そのために

- 数学的構造によって複雑な概念を定式化するのに有用な圏論
- 論理と計算の記述、あるいは仕様、プログラム、証明の記述に有効な構成的型理論
- プログラミング言語 Haskell

の三つを共通知識基盤として、これらの枠組を使って考えを交換することを奨励してみました。しかし、結果をみると、これは部分的にしかうまくいっていないようです。

研究センター設置後まもなく共同開発を開始した、構成的型理論に基づく証明支援系 Agda のおかげで、

構成的型理論の考えは急速に研究員に広がりました。

しかし、圏論は、CVS 設置当初はいくらか広まったものの、七年目の現在は、研究員の間で殆ど使われていません。これは研究センターの研究活動が、当初想定したよりも大幅にフィールドワークに寄ってしまったために、圏論の有効性が明らかになるような複雑な数学的議論があまり必要でなくなったことに原因があると思われます。科学研究を強化することになれば、圏論の有効性がまた明らかになるだろうと見ています。

Haskell もあまり浸透していません。この理由は二つ考えられます。一つは Agda 言語から Haskell へのコンパイラができてしまった上、Agda 言語にも随分改良が加えられて書きやすくなったので、函数型言語としてはむしろ、依存型まで使える Agda 言語を使うようになったこと、つまり当初の予想に反して Haskell が不要になってしまったことです。一方、実行時の高い性能が求められる場合には Java や C++が用いられることになったことが、もう一つの原因としてあげられます。

このように具体的に掲げた知識基盤は研究員メンバーの間に広がったものもあれば広がらなかったものもありますが、議論の土台になる枠組を共有しようという試みは、研究組織として、今後も常に続けていくべきものと考えています。

6. 班とチーム

研究センターは、全体として目標達成のために働かなければなりません。そのためにいろいろな仕事を手分けして行なうのですが、一般に、企画、計画、実行、後始末、などの段階からなる一まとまりの仕事の手分けするやり方に二通り考えられます。ひとつの分け方は、仕事を、いくつかの、より小さな一まとまりの仕事にわけ、それを手分けすることです。おのおののより小さな一まとまりの仕事は、そのレベルで再び企画、計画、実行、後始末などの段階を持ちます。もう一つは、一まとまりの仕事を、企画だけ、実行だけ、後始末だけなどの段階別に分けるやり方です。後者の極端な場合は Chaplin の Modern times に風刺されたような流れ作業で、ここまでいってしまうと作業に携わるもの自身が仕事へのやりがいを感じなくなりかねないことは、よく言われるとおりです。

研究員も支援要員も含めた各メンバーがやりがいを感じながら仕事を進めるためにも、できるだけ前者のやり方で仕事を手分けし、流れ作業に陥らないようにして、各メンバーがやりがいを感じながら仕事をできる場所にしたいものだと考えました。そのために一まとまりの仕事、つまりプロジェクト毎に、一つの班を作ってそれが見える形にしました。そうして、各メンバーが、それぞれの立場や能力に応じて、企画から計画、実行、後始末の段階を全部経験しながらやりがいを感じて仕事をしてほしいと期待しました。

班を作ったおかげで、誰がどの仕事に従事しているのかが明らかになりましたが、メンバーから見ると、自分がどの仕事に関わっているのかを明確にすることにもなりました。これはどちらかというと、いろいろなことを裁断していく父性原理によるやり方で、なんでも包みこもうとする母性原理による進め方をとりがちなわが国では理解されにくいことだったかもしれません。また、やりがいなんかどうでもよい、何をしたら給料をくれるのかをはっきりさせてほしい、とまあ、ここまで直接に言うものは現れませんでしたが、要するにそういう風に考えているものもいて、組織管理の難しさを痛感させられたこともあります。

産総研になって導入された短期評価のためには、班の制度によって仕事を明確にしておくことは、評価者にとっても被評価者にとっても、好都合なことでした。なお、班のようなプロジェクト指向の組織は、筆者が産総研設立前に所属していた電子技術総合研究所の最後の何年間かで既にラボという名前で導入されていたもので、その考えを借用してできたものです。

各班には班長を設けてリーダを明確にしました。しかし、班長は班員に比べて人間的に偉いというのではなく、リーダとしての役割を負うだけなのだということを強調しました。一般に研究センターのメンバーは複数の班に所属するのですが、ある班で班長と班員だった関係が別の班では逆転して班員と班長の関係になる、ということがあちこちで起こりました。しかし、我々はそのことを特に問題にはしていません。この点については、産総研中央の管理側から心配する意見をいただいたことがありますが、仕事を進める上で問題を感じたことはありません。

研究センター内部の組織として、当初は、仕事別につくる班を作るだけですすめようとしたのですが、すぐに、メンバー別の組織の必要が感じられました。中長期的なキャリアに関する世話、つまり進路相談や細かな日常業務に関する管理、また、途中から産総研で本格的に導入された勤務管理などを行なうためには、各メンバーを階層的に所属させる、別の組織を設けるが望ましいことがわかりました。そこで、産総研全体で研究センターの下部組織として定めている研究チームを、階層的な組織として用いることにしました。その結果、各メンバーは、一般に複数の（あるいはゼロ個の）班に属し、また、ちょうど一つの研究チームに

属する、という形で組織を管理することとなりました。班と研究チームのような二通りの組織を作って管理するやり方はマトリックス管理と呼ばれていて、既にいろいろな組織で行なわれていることを、後に、当時人事担当の理事を務めておられた小林直人氏からご教示いただきました。

この班と研究チームによる管理は、大体のところうまく機能したと感じています。一年毎の短期評価にもうまくマッチしたし、誰がどの仕事に従事しているのかが、新たに加入したメンバーにもすぐにわかるようになっているように思います。

7. 企業との共同研究

産総研に改組後四年目に CVS が設置されたのですが、そのメンバーは科学研究よりの研究経歴を持つものばかりで、産総研改組前に企業との共同研究の経験を持つものは皆無でした。そればかりでなく、どちらかというと、産業と関係して活動すること嫌悪する傾向があったことは否めません。しかし、フィールドワークという考えによって企業との共同研究を開始したところ、これはこれで大変面白く、やりがいを感じながら進めることができたように思います。先輩から、それほど産業との協業を強調して、若手の研究員が納得するのか？との心配をいただいたこともありますが、若手研究員の連中のほうが産業との協業を面白がった場合もあります。尤も、徹底的に産業との協業を嫌うものがいなかったわけではありません。フィールドワークを面白いと感じるものが最後まで残っているというだけのことかもしれません。別の先輩は、「研究者にはあまり実用化研究をさせないほうがいいという説がある。なぜなら、それを始めたものは皆、面白くてやめられなくなり、基礎研究に携わるものがいなくなるから」などという話をしてくれました。若干怪しげな説ではありますが、実用化研究が面白いものだ、というところは今の我々にはよくわかるように思います。

数理的技法は、情報システム検証の手法です。手法研究を企業との共同で行うというのは、つまるところ、その企業において、できればそこの開発現場で手法の適用を実験するということにつきます。我々がフィールドワークと称して行ってきた企業との共同研究は、すべてそのようなものでした。このような共同研究を行うにあたってのノウハウも

縁側から奥座敷へ

をはじめとして、LVS/CVS 活動七年間をとおして、いくつか蓄積しました。

これについては、

木下佳樹、高井利憲： フォーマルメソッドのフィールドワーク、「情報処理」、第 49 巻第 5 号、第 9-15 頁

にまとめてあります。

8. 連携検証施設「さつき」と組込みソフト産業推進会議

2007 年秋の首都圏鉄道改札システムのトラブルなどを契機に、経済産業省は組み込みシステムの信頼性向上技術のための産学官連携施設を CVS などの産総研における活動を支援して拡充することとし、検証向けに設計されたクラスタコンピュータが関西センターに設置されることとなりました。産総研内にはこのための連携検証施設「さつき」（CFV）を設置しました。

これと相次いで関経連が組み込みソフト産業推進会議（推進会議）の活動を三年間の予定で開始し、二つの活動が相互に作用して、大規模な連携活動を生み出しつつあります。クラスタコンピュータなどを使った検証サービスと、組込み適塾と名付けた研修コース提供の、二つを核にした事業が開始されており、推進会議の設置が 2010 年春に終了した後も、関経連側でも後継組織をつくって活動を継続することになっています。

「さつき」では学術利用も行っており、jopshop scheduling problem の未解決問題解決などの成果が得られています。

9. 研修コース開発と組込み適塾

手法の適用実験を企業の技術者が行うには、まず彼ら技術者たちが手法を身につける必要のあることが、フィールドワークを開始した初期のうちから明確になりました。そこで、適用実験に用いる技術を技術者に伝えるための研修コースを本格的に開発することとしました。これまで、モデル検査のコース、対話型定理証明のコース、などを試行コースとして無料で提供し、延べ百名以上の技術者に対して研修を提供しました。これらをカリキュラム化することも開始しており、CVS 教程のシリーズとしてモデル検査初級、上級などの教科書をナノオプトメディア社から刊行しています。

以上の CVS による研修活動に加え、2008 年からは推進会議と関西センターの共催で組込み適塾を開催しています。約 30 名の受講者に対し、3 ヶ月にわたって 30 前後の演習付講義を開講する基本コースと、リバースエンジニアリングやモデル検査などのトピックに関する数日の研修コースを開催する実践演習を設けています。組込み適塾共催にあたって産総研側で核となっ

ているのはこれまでの研修コース開発の活動です。

研修は、技術移転の中核として位置づけられる活動です。今後も最も重要な活動の一つとして続けていくべきだと考えております。

10. 法定計量のソフトウェア認証

産総研の重要な任務の一つに National Measurement Institute としてわが国の法定計量の検定・型式承認の管理にあたることがあり、計量標準総合センターがその任にあたっていますが、最近の計量器は殆ど皆計算機制御になっており、その中のソフトウェアをどのように型式承認するかが国際的な問題になっています。産総研設立の直前に、たまたま田中充氏（計測標準分野担当研究コーディネータ、当時計測標準研究部門副部門長）が関西センターを訪ねられたときにその話をされたことがきっかけで、CVS がソフトウェア認証に関してお手伝いすることとなりました。当時は非自動はかりの組込みソフトウェア認証に関するヨーロッパの規格があるだけでしたが、現在、一般の計測器に関してソフトウェア認証を導入するためのガイドラインが OIML の公式文書 D31 として定められており、前後してタクシーメータ、非自動はかりなどには既にソフトウェア認証が開始されています。これらの活動のために CVS から二名の研究員が計測標準研究部門に移籍し、任にあたっています。これは当時計測標準研究部門長であった小野晃氏（現副理事長）もソフトウェア認証を重要案件として積極的に支持された結果でもあると思っております。

法定計量のソフトウェア認証に関係したのをきっかけに、機能安全やディペンダビリティなど、ほかの分野のソフトウェア認証の仕事も開始しています。あとでご紹介する利用者指向ディペンダビリティの研究もこの方向の仕事ですが、これらの仕事は、計算機システムから社会システムへ留めを向けるきっかけをも与えてくれたように思います。

11. 今後—検証を不要にするために

2010 年 3 月末に CVS の六年間の設置期限が終了した後も、殆どのメンバーは CFV(Collaborative Facilities for Verification—連携検証施設)において活動を続けます。

これまでは、システムの検証に焦点を当てて活動してきましたが、実はそもそも、検証という作業は行わないで済むのが理想でしょう。誤りのない情報システムしかできないような開発の方法があって、C by C—correctness by construction という言葉が使われたりしていますが、このような方法があるのならそれに越したことはありません。従って、今後はシステムの検証から、システムの開発全般にわたる技術に、研究の対象を拡大していくべきでしょう。それはソフトウェア工学あるいはソフトウェア科学といわれているものに近いのですけれども、我々が取っている数理科学に基づくアプローチは、対象をソフトウェアに限定する必要はありません。情報を処理するシステムであればハードウェアでもかまわないし、企業や法人のような社会組織を対象にすることもできます。最近進展しつつある法令工学といわれる分野もその一部です。このような分野をなんと呼ぶべきか、算譜科学、記述の科学など、いろいろに考えているのですが、そのことをも今後の課題の一部だということにしておきたいと思います。

形式手法の知識体系 FMBoK (Formal Methods Body of Knowledge) 構築に向けて

田口研治†
国立情報学研究所 GRACE センター

形式手法（Formal Methods）は情報システムの信頼性・安全性を保証する開発技術として注目されつつある．しかし、形式手法には多くの異なる特徴を持つ言語・表記法、検証技術があり、それらを使いこなすための実践的な知識が十分に確立されているとは言えない状態である．また、そのための教育カリキュラムの標準化のための知識体系が確立されていない状態である．本論文においては、形式手法の知識体系 FMBoK (Formal Methods Body of Knowledge) 構築のための試みについて、その構想（目標、体制）について述べるものである．

Towards Building a Body of Knowledge on Formal Methods (FMBoK)

Kenji Taguchi
National Institute of Informatics, Grace Center†

Formal Methods have been gaining attention as a development method for assuring reliability and safety of information systems. However formal methods comprise of a wide variety of formal languages/notations and verification methods and their best practices have not been established yet. In addition to this, the curricula have not been standardized due to lack of a body of knowledge. This paper will give an overview on an initiative to build a body of knowledge on formal methods (FMBoK) and touches on its vision including its goal and working team.-

1. はじめに

現在の情報社会において、情報システムの安全性は社会全体に大きな影響を及ぼす要因として、非常に大きなウエイトを占めている．電気・電子関連機器の国際安全規格である ISO／IEC 61508 [1]，セキュリティ保証に関する国際標準 Common Criteria (ISO/IEC 15408) [2] などにおいて，形式手法（Formal Methods）の利用の推奨や, 利用の義務化等, システムの信頼性・安全性を検証する技術として形式手法への関心が高まりつつある．さらに，産業界においていくつかの本格的な活用が報告されている．しかし，形式手法には多くの異なる特徴を持つ言語･表記法, 検証技術があり，またこれらを使いこなすための実践的な知識もまだ十分に確立されているとはいえないのが現状である．本論文においては，このような状況を打破するための方策として，形式手法（Formal Methods）における知識体系構築の重要性を論じることを目的としている．

ソフトウェア工学においては SWEBOK(Software Engineering Body of Knowledge) [3] や SE 2004 [5] が知られている．これらを用いて形式手法の教育を行えば充分である，という考え方もある．しかし，現在，形式手法は非常に多岐に渡り研究開発が進んでおり，そのほとんどは SWEBOK において網羅されていないのが現状である．

形式手法において知識体系が最初に論じられたのは J. Oliveira [8] においてである．これは， FME (Formal Methods Europe) における教育を担当とするサブグループ FME-SoE (Subgroup on Education) において，実施された教育調査を元に，知識の分類を行ったものである．残念ながら，体系的なまとまりに欠けているものであった．我々はモデル検査に限った知識体系（MCBOK）の構築を実施しており[9,12,13]，その体系を基に， FMBoK の構築を考えている．

本論文においては，次章において工学・科学分野における知識体系について概観し， 3 章において FMBoK の構想について，その目標，体制について述べることで示す．そして，最後に， 4 章において結論を述べる．

2. BoK とは

BoK とはある工学･科学分野における知識を表現する重要な概念，技術の構造的な集まりと，それに対する共通理解を意味する． BoK は技術的用語と理論的概念および推奨される現場の知識を含む．様々

な工学・科学分野において知識体系 BoK (Body of Knowledge) が構築されている. ソフトウェア工学の分野においては，既に述べた IEEE Computer Society と ACM が共同で構築した SWEBOK や SE 2004 が知られている．またプロジェクト管理分野においては PMBOK (Project Management BOK) [4] が Project Management Institute (PMI) により構築されている．これらのような広い領域を網羅する BOK 以外にも非常に小さな技術領域を対象とするものを存在する．PSP (Personal Software Process) に関する BOK [6] はその例である．

BoK は知識を木構造により表現する．ただし，木構造の深さに対して一定の制限をすることで，理解度と可読性を保持している. BoK の主目的は知識の分類を行うことであるが，その詳細な説明や関連文献などについては別の文書「ガイド」により与えられる．例えば，SWEBOK に対しては，実際に流布しているのは「ガイド」である．

BoK の目的は教育におけるカリキュラムの標準化と同時に，その分野における実務者に対する資格認定が大きな役割である．例えば， PMI は 5 つの資格認定を行っている．この二つの目的は， BoK の構築において利害の衝突を生じさせる場合がある．現在，ソフトウェア工学における三つのイニシアティブ，SWEBOK，SE2004，Certified Software Development Professional (CSDP) を改訂する動きがある．そこにおいては，大学教育における知識体系（SWEBOK, SE 2004）と，資格認定（CSDP）における知識体系をマージさせるというものである．

3. FMBoK

本章においては，FMBoK が目指す目標, 知識体系, イニシアティブとしての体制の構想に関して述べる．

3.1 目標

FMBok の役割としては，通常の BoK としての二つの役割，教育カリキュラムの標準化と資格認定の両方の役割を目指すものである．特に，アプローチとしては，当初から資格認定を重視するものである．この理由としては，2 章で述べたソフトウェア工学における 3 つのイニシアティブにおいて現在, マージする努力が行われているのは，大学における教育カリキュラムの標準化とソフトウェア開発者の資格認定のためのカリキュラムを別々に開発してしまったために，その整合性を取る努力が後で必要になったことに起因するからである．

知識体系としては，形式手法全体を網羅的にカバーするものである．モデル検査分野に関する知識体系 MCBOK (Model Checking Body of Knowledge) [9,12,13] のドラフト版については既に発表を行っており，その際の知見を FMBoK 構築に活かすことを考えている．

3.2 知識体系

知識体系を構築するためには，主な知識領域（Knowledge Area, KA）を何にするかを決定する必要がある．例えば， SWEBOK においては，主に，開発プロセスに沿った知識領域を設定している．形式手法は開発プロセス全体において用いられている，というよりは要求仕様から始まり，設計，構築（construction）にかけて重点的に利用されている．

このような考えに基づき，知識体系としては，開発プロセスに沿って構築するのでは無く，各技術・理論分野毎に構築する方法を取ることを構想中である．つまり，モデル検査の知識体系，形式仕様記述の知識体系，定理証明の知識体系といった知識領域を設定し，開発プロセス毎の特徴については，各知識領域内で取り扱うという方法である．このような方法を採用する他の理由としては，上記の分野毎に専門家が分化している，という現状もある．ただし，形式仕様記述言語に対するモデル検査や定理証明などが要素技術として存在するので，そのような場合には，各知識体系間での相互参照を行うことにより，重複を避けることを考えている．

3.3 活動状況と今後の体制

これまで著者は形式手法の知識体系の構築に向けて, 様々な活動を行ってきた．2008 年には FMET (Formal Methods Education and Training) 国際ワークショップ [10] を M. Hinchey (Lero, U. Limerick), J. Davies, J. Gibbons (U. Oxford) らと組織した．本ワークショップの論文は後に ACM SIGCSE から刊行された [11]．さらに 2009 年においては，FME 主催の TFM (Teaching Formal Methods) 国際会議において FMBoK についてパネル討議をすることで, FMBoK のアイデアの深化, 協力者を募ってきた（パネルのチェアは M. Hinchey. パネリストは著者，P. Larsen, J. Oliveira, R. Johnson ）．今後も FMET, TFM を中心に本活動を組織することを計画している．

FMBoK は当初から国際化を目指して構築するのを目指しており，その母体として FME-SoE と共に行動することが合意されている．さらに，作業メンバーとしては，産学官における研究者，実際に利用している開発者の両方を含む，バランスの取れたチーム構成を目指すものである．国内においては，MCBOK 構築の

主メンバーである，西原（産業技術総合研究所システム検証センター），青木（JAIST），篠崎（関電），早水（メルコパワーズ），粂野（MRI，NII），石川（NII）を中心に，より広くメンバーを募集する予定である．

4. 終わりに

本論文においては, 形式手法の知識体系 FMBoK をどのように構築するかについて構想を述べた．本知識体系の特徴としては, 当初から大学の形式手法教育と，開発者の資格認定の両者を行うことを想定して構築が行われる，という特徴があり，組織作りにおいてもその考えを考慮している．

どのような工学分野においても BoK が存在するといっても過言ではないが，必ずしも BoK が技術，学術教育の解決方法で無い．例えば， D. Notkin らは [7] において SWEBOK に対する批判をしている．本イニシアティブはこれらの様々な先見者により得られた知見を反省材料にして, 構築を行うことを目指している．

参考文献

1) IEC61598: Functional safety of electrical/electronic/programmable electronic safety-related systems. Bureau Central de la Commission Electrotechnique International, 2000.

2) ISO/IEC 15408: Information technology – Security techniques –Evaluation criteria for IT security – Part1, Part2 and Part3, 2005.

3) A. Abran, J. W. Moore, P. Bourque, and R. Dupuis: Guide to the Software Engineering Body of Knowledge, 2004 version SWEBOK, IEEE, 2004.

4) Project Management Institute: Guide to the Project Management Body of Knowledge (PMBOK Guide), Forth Edition, PMI, 2008.

5) The Joint Task Force on Computing Curricula: Software Engineering 2004, IEEE and ACM, 2004.

6) M. Pomeroy-Huff, R. Cannon, T. A. Chick, J. Mullaney, and W. Nichols: The Personal Software Process (PSP) Body of Knowledge, Version 2.0, CMU/SEI-2009-SR-018, 2009.

7) D. Notkin, M. Gorlick and M. Shaw: An Assessment of the Software Engineering Body of Knowledge. A Report to the ACM Council, 2000.

8) J. N. Oliveira: A Survey of Formal Methods Courses in European Higher Education. In Teaching Formal Methods (TFM) ’04 Conference Proceedings, pp 235-248. LNCS 3295, Springer, 2004.

9) 青木，粂野，木下，篠崎，高木，高村，田口，中原，西原，早水，本位田，渡邊: モデル検査の教育プログラム構築に向けて”，算譜科学研究速報，（独）産業技術総合研究所,システム検証センター, AIST-PS-2008-012. 2008.

10) J. Davies, J. Gibbons, M. Hinchey, K. Taguchi (editors): Proceedings of the first international workshop on Formal Methods Education and Training, GRACE-TR-2008-03, GRACE Center, National Institute of Informatics, October 2008

11) J. Davies, J. Gibbons, M. Hinchey, K. Taguchi: A preface on Special section on Formal Methods Education and Training, ACM SIGCSE Bulletin inroads, vol. 41, issue2, June, pp14-16, 2009.

12) H. Nishihara, K. Shinozaki, K. Hayamizu, T. Aoki, K. Taguchi, F. Kumeno: Model Checking Education for Software Engineers in Japan, ACM SIGCSE Bulletin inroads, vol. 41, issue 2, June, pp45-50 2009.

13) 西原，青木，粂野，篠崎，田口，早水: MCBOK2008:ソフトウェア開発のためのモデル検査知識体系，組み込みシステムシンポジウム(ESS ‘09) , 2009.

モデル検査知識体系 MCBOK2008 について

西原秀明†

モデル検査教育の質の向上と技術の普及を目指し，大学・産業界に属する組織と共同で教育体系の基礎付けを進めている．著者らが行ってきた教育活動の概要と結果について述べた後，モデル検査分野の知識を体系づけてまとめた MCBOK 2008 についてポイントや利用法を説明する．

On Model Checking Body Of Knowledge 2008

Hideaki NISHIHARA†

Activities on modelchecking training for engineers are discussed. The Activities are in collaboration with NII, JAIST, KEPCO, MPS, and CVS/AIST. Training courses managed by authors are reviewd and evaluated. Next, a foundamental document of model checking called Model Checking Body Of Knowledge 2008 is introduced.

1. はじめに

本稿の内容は国立情報学研究所 (NII)，北陸先端科学技術大学院大学 (JAIST)，関西電力 (KEPCO)，メルコ・パワー・システムズ (MPS) との共同研究[4),11),12)]をもとにしている．

近年，情報処理システムの普遍化，複雑化によって，システムに要求される信頼性はますます高くなってきており，形式手法が注目されている．特にモデル検査は人間が見落としがちな想定外の振る舞いを全数探索によって発見することができ，強く期待されている．この数年では産業界一般における認知度も高まり，国内でもシステム開発における本格的な導入が進み，多方面の分野における適用事例 や企業により開発されたツール が発表されるなど注目されている．さらに，モデル検査を科目として持つ大学のカリキュラムや人材育成プログラムも増えてきているし，モデル検査を特色としてあげるコンサルテーション企業も存在する．

著者らは産，学，官それぞれの立場から技術者向けモデル検査の教育・普及を進めてきたが，更に普及を推進することを目的に連携し，より一般的な見地から活動している．

まず，お互いの研修活動やカリキュラムについて情報を交換し，技術者向けモデル検査教育のあるべき姿を検討した[4),12)]．その結果は第 2 節，第 3 節で述べる．

その検討を受けて，モデル検査技術の普及を推進するためには，統一的な基礎付けが必要だという認識に至った．つまり，モデル検査に関する知識を体系化し，新たなカリキュラムの開発や既存カリキュラムの比較の際の基礎資料とすべきである．

関連する分野で，上記のように教育の基礎となるものとしてはソフトウェア工学の教育に関する SWEBOK[1)]，SE2004[6)]，J07[10)] などがある．しかしその中には，形式手法がトピック名として現れている程度で，モデル検査とその知識については殆ど記載がない．

そこで，著者ら自身でモデル検査の知識を体系化した[11)]．その成果である知識体系を「ソフトウェア開発のためのモデル検査知識体系 (MCBOK 2008)」と呼ぶことにする．

MCBOK 2008 は「モデル検査のプロセス」,「モデル検査の理論」,「モデル検査ツール」,「モデル検査の適用対象」の四つの領域を持ち，モデル検査の知識が存在する広い範囲から知識項目を収集して体系化している．各領域は段階的に副領域に分割され，その中に「有限オートマトン」「内部モデル/外部モデルの設計」「NuSMV」といった知識項目が分類されている．

MCBOK 2008 はモデル検査カリキュラムの基礎的な資料として機能する．第 5 節で述べるように，著者らがこれまで進めてきた教育活動のカリキュラムとの対応づけを考えることで，カリキュラムの扱う内容を顕にして特徴付ける．新たにカリキュラムを策定す

† 産業技術総合研究所 組込みシステム技術連携研究体
Collaborative Facilities for Verification, National Institute of Advanced Industrial Science and Technology(AIST)

る際にも，到達目標に合った教育する知識の選択を支援する．その他，技術認定の枠組み構築や，コンサルテーションなど，モデル検査の関係する幅広い活動の基礎資料として機能することを期待している．

2. カリキュラムの比較

著者たちが行ってきた，技術者向けモデル検査研修について述べる．トップエスイー (NII)，JAIST 組込み大学院，MPS による研修テキスト，CVS/AIST 研修コースそれぞれのカリキュラムを比較し，受講者からの評価を紹介する[4](但し 2007 年時のデータである)．

開催形態は以下の通り．

NII スーパーアーキテクト育成プロジェクト「トップエスイー」の中でモデル検査に関係する科目 7 科目を持つ．各科目につき，週一回の講義 12 回．受講者対象は SW 科学専攻の修士修了程度の知識を持つ社会人．

JAIST 組込み大学院科目「ソフトウェア検証手法」(週一回の講義 15 回)，日科技連セミナー「ソフトウェアモデル検査入門」(3 日間集中)．社会人対象．

MPS 社内セミナー基礎編 (12 時間)・応用編 (40 時間)・実践編 (80–160 時間)．受講者対象は社内のプログラマ，設計者．

AIST モデル検査研修コース初級編 (4 日間集中)・中級編 (3 日間集中)・上級編 (3 日間集中)．初級編受講者対象は，プログラミングの経験を持つ学生・開発者．

表 1 は各カリキュラムの共通点と相違点についてまとめたものである．各カリキュラムにおいて，ツール，論理体系，システムの性質，抽象化，検証プロセス，基礎理論，と理論から実践までバランスよく項目が扱われている．モデル検査は従来の開発手法・検証手法と異なり，理論的基盤である数学や論理について一定の知識が必要であること，受講者の背景知識や経験が多様であることから，カリキュラムで扱うべきことは少なくない．実際，研修期間をもっと長く確保したかったという意見も出ている．また，各カリキュラムでは実践・現場への適用を念頭においており，検査ツールの機能，操作や検証プロセスについても扱っている．教材で扱われる例題や演習も，近年モデル検査の適用領域として注目されている組込みシステム関連から選ばれていることが多い．

一方，カリキュラム間の相違点も少なくない．AIST のカリキュラムでは理論に属する内容を多く扱っているほか，TopSE では複数のツールを用いてモデル検査ツール一般の知識を伝えようとしている．JAIST のコースではシステムプログラムを題材にしたモデルを百余り準備し，実際の適用の様子を想起させている．また，MPS の研修では，社内教育の利点を活かし，実際の開発対象を題材にして開発業務の中でモデル検査の経験を積ませるという手法をとっている．

モデル検査という技術は多様な分野と関連を持っていて，個別の対象領域・問題に合ったツールや知識の選択が課題の一つになることを示している．新たにカリキュラムを策定する際にはその理念と到達目標を確認した上で内容を決めていくことが必要である．

表 1 モデル検査教育内容の比較
Table 1 Common/Peculiar topics in courses

共通項目	非共通項目
ツール · Spin · SMV 並行システム 論理体系 · LTL · CTL システムの性質 · 安全性 · 活性 · 公平性 抽象化 · 抽象化写像 · 構造の保存定理 · データ抽象化 · 述語抽象化 検証プロセス 基礎理論 · 有限/Buchi オートマトン · CTL モデル検査	AIST: · LTL と CTL の比較 · 遷移系の合成 · Kripke 構造への変換 · 空語判定問題 · 有界モデル検査 TopSE: · 時間オートマトン · LTSA MPS · 実システムの検証

表 2 アンケート結果
Table 2 Questionnaire Results
アンケートの設問に対して，好評価 (平均点以上) の回答数を集計

	受講者数	理解度	有用さ	実施可能性
AIST (初級)	75	59(79%)	70 (93%)	N/A
AIST (中級)	24	7 (29%)	24 (100%)	18 (75%)
TOPSE (基礎)	36	22 (61%)	N/A	26 (72%)
TOPSE (応用)	7	5 (71%)	N/A	5 (71%)
JAIST	61	58 (95%)	54 (89%)	43 (70%)
MPS	13	10 (77%)	N/A	10 (77%)
合計	216	161/216 (74%)	148/160 (93%)	102/141 (72%)

3. 評　　価

それぞれのカリキュラムで研修終了後に受講者アンケートを実施した結果について述べる．総合すると，

受講者からの評価は良好であった (表 2).

内容の理解度，モデル検査の有用性と実施可能性それぞれについて，各カリキュラムで肯定的な回答が多かった．このことは受講者のモデル検査に対する期待を示していると同時に，技術者向けモデル検査カリキュラムとして適切なものであったことを示している．

自由形式によるアンケート回答から以下のような意見が得られた．

- 多くの受講者がモデル検査の分析力を評価しており，そのことは表の「有用性」に対する回答が非常に良かったことにも現れている．特に並行動作システム，非決定的に振る舞うシステムのように，人力で検査できない対象に対する検査能力には驚いており，学習への動機付けとして効果が高い．
- より実践的な知識についての要望が強い．具体的なシステムのモデル化，検査式の作りかたや，従来の開発プロセスにモデル検査プロセスをどのように当てはめるか，といった事柄に対しての質問や要望が多かった．
- 論理記号や抽象概念に対する苦手意識が強い．また理論を学ぶモチベーションは低い．

これらの意見は教材改善のポイントを示唆している．実践的な知識への要望については，実際のところ個別の適用領域や個別の開発手法に対してモデル検査の適用方法を整理し教材を準備することは現実的でない．しかし，典型的な成功事例を題材として扱うことで適用の指針を伝えることは可能である．記号や概念については，その意義や具体例との関連を充分に示すことや，導入の工夫により改善できる．適切なモデル化，適切な検査式の作成，抽象化など，モデル検査では理論を効果的に用いて検査の質を高められる場合が少なくない．具体例により意義と効果を示しつつ理論的知識を伝えるなどの工夫が必要である．

4. MCBOK

前節のカリキュラムに関する議論を受けて，モデル検査の知識を体系づけてまとめた[11]．これを MCBOK 2008 と呼ぶ (本稿末尾に掲載).

ある分野における知識を体系化してまとめたものを知識体系 (Body Of Knowledge) と呼び，BOK と略記する．知識には用語・概念のような理論に属するものも含まれるし，実践の際にノウハウとして適用されるようなものも含まれる．関連分野ではソフトウェア工学知識体系 SWEBOK[1]，プロジェクトマネジメント知識体系 PMBOK[5] やソフトウェア品質知識体系 SQuBOK[13] などが発表されている．また「情報専門学科カリキュラム標準 J07」[10] には領域ごとに定められた BOK が同梱されている．

多くの BOK では対象領域に関する知識を三段階から五段階程度の樹形図の上に配置することで構造を可視化している．BOK に求められるものは，本質的にこの樹形図だけであって，そこに現れる知識の解説や参照文献などは「ガイド」として別に扱うことが多い．

4.1 モデル検査のプロセス (KA 1)

以下の節で MCBOK 2008 の領域ごとに簡単な説明を加える．

KA 1. ではモデル検査をソフトウェア開発に適用する際の知識があげられている．

KA 1.1 では目的と工程によってモデル検査の適用に関する知識を整理している．一般にモデル検査は設計検証の技法と言われているが，最近ではソースコードの検査事例や開発ツールに統合されているものもあり多様化している．

KA 1.2 「モデル検査の適用手順」にはモデル検査を適用するというプロセス自体についての知識を整理している．検査範囲の特定，モデルの抽象度決定，外部環境の構築，モデル検査で検査できる性質，ツールの実行などが含まれている．

モデル検査を開発に適用するに当たり，そのコストや実施体制は大きな問題となる．KA 1.3 「モデル検査のコスト」，KA 1.4 「モデル検査の実施体制」でそれらの知識を整理している．

4.2 モデル検査の理論 (KA 2)

KA 2 は理論的基礎に関する知識を体系化した領域である．

KA 2.1 「計算モデル」にはモデルを記述するための枠組みを整理した．多くのモデル検査ツールでは有限オートマトン，またはそれに類する各種の構造を使ってモデルを記述する．個々の計算モデルについて，それを基盤とする検査ツールがあり，ソフトウェア開発における適用例があるものを選んで知識項目として採用した．

KA 2.2 「時相論理」には検査式を記述する論理体系について整理した．現在モデル検査で使われている論理体系は LTL と CTL を基にしている．それぞれの意味論と決定可能性，表現力をここで扱う．

KA 2.3 「モデルの縮小手法」は状態爆発問題の回避方法に関する知識を整理した．モデル化の対象の分析によってモデル化の範囲を縮めることで縮小する方法もあり，一旦モデル化したものを機械的に縮小する方法もある．縮小の前後を比較し正当性を示す手段も含む．

KA 2.4 「モデル検査の原理」では，モデル上での検査式の妥当性を検査するアルゴリズムの知識について整理した．

4.3 モデル検査ツール (KA 3)

KA 3 はモデル検査ツールに関する知識を体系化した領域である．ソフトウェア開発における適用例があり，動作原理についての文献が存在しているものを採用した．

KA 3.1 はモデル検査アルゴリズムを実行するツールを整理した領域である．モデル検査ツールは個々に基盤となる理論，入出力の形式，検査規模の限界，検査できる性質が異なり，6 個の領域に分かれている．

KA 3.2 はモデル検査アルゴリズムを実行するものではないが，検査作業を支援するツールを整理している．

4.4 モデル検査の適用対象 (KA 4)

「モデル検査の適用対象」は適用事例を分類したものである．モデル検査は理論的な適用範囲は広く，殆ど全てのシステムを適用対象とすることができるが，適用効果が高い分野とそうでない分野がある．また，未だモデル検査が日常的に使われているとは言いがたい現状にあって，適用例を整理してまとめておくことには MCBOK の目的からも価値がある．

5. MCBOK の利用法

MCBOK 2008 の目的はモデル検査における知識の構造を可視化し，産業界 (主にソフトウェア開発分野) へのモデル検査の普及を推進することである．主目的は教育の際の基礎資料であるが，他にも幾つかの利用法が考えられる．

5.1 カリキュラムの基礎付け

第 2 節で述べたように，モデル検査を学ぶには様々な背景と目標が考えられ，教育内容や教育の順序その他も多彩なものが考えられる．現在，技術を身につけたい開発者も，開発プロセスへの適用方法を知りたい管理者も，同じカリキュラムで学ばざるを得ないが，実はそれぞれ目的を満足する教授内容は異なる．それぞれの目的に応じて MCBOK の各知識領域から必要な知識項目を選択し，カリキュラムを構成することで効率のよい知識の習得が可能になる．

逆に既存のカリキュラムに対して，扱われる内容を MCBOK 2008 の各知識とを対応づけることによってそのカリキュラムを評価できる．**表 3** は産総研のモデル検査初級のカリキュラム[12] について，各項目の MCBOK 2008 の対応する知識領域を表したものである．このカリキュラムには「モデル検査のプロセス (KA 1)」の知識が多く扱われており，モデル検査の適用のしかたに興味を持つ学習者に適していることがわかる．

表 3 産総研モデル検査初級カリキュラムから MCBOK 2008 へのマッピング
Table 3 Mapping from AIST's curriculum to MCBOK 2008

カリキュラム項目	対応する知識領域
モデル検査とは	
状態, 遷移, 次の瞬間, 非決定的/決定的	1.2(4),2.1(1)
安全性, 活性	1.2(2)(6)
モデル検査の基礎	
状態と遷移を定めて状態遷移系を書く	1.2(1)(4)
検査する性質:正しい性質/反例のある性質	1.2(2)
遷移系を目でみて検査	1.2(7)
ツールによるモデル検査	1.2(5)(6)(7)(8)
検査式の記述	
直列回路, 並列回路, And/Or の導入	1.2(6)
Until operator	1.2(6)
並行システムと排他制御	
並行システム/セマフォ/Critical Section	1.2(4)(5), 2.1(1), 4.2
検査:同時に CS に入らない	1.2(2)
検査:各プロセスとも CS に入ることが可能	1.2(2)
公平性の問題	2.1(1)
LTL	
パス, パス上の真偽	2.2(1)
様相記号	2.2(1)
意味論	2.2(1)
LTL 式を書く演習	1.2(2)(6),2,2(1)
ソースコード検査	
簡単な C プログラムのモデル化	1.2(4)(5), 4.1
テスト技法とモデル検査の違い	-
演習:自動販売機	1.2(4)(5)(6)(7)(8), 4.1

5.2 保有する知識の可視化

個人や組織が，自身が持つ技術の程度を知るツールとしても MCBOK 2008 は有効に働く．モデル検査に関して幾らかの知識を持つ技術者が，MCBOK 2008 に挙げられている個々の知識について，

- モデル検査の適用時に，その知識を意識しているか，
- その知識を自分なりに知識として整理して使っているか，

を確認することで，その技術者が保持する知識や適用時に意識していることを明確にできる．同様に不足している知識，その時点では必要でない知識も明確になり，将来の学習計画を立てる際にも有用である．

5.3 モデル検査の適用の指針

システム開発プロセスへのモデル検査導入を検討する際のツールとして MCBOK 2008 は機能する．開発プロジェクトにおいて，開発対象の種類や特徴に

よって，また使用する他の開発技術によって，モデル検査をどの部分にどの程度適用するか違いが出てくる．MCBOK 2008 の「モデル検査のプロセス (KA 1)」を参照することで，モデル検査の適用工程に関する知識が明らかになり，開発工程の適切な部分でモデル検査による検証を行えるようになる．また,「モデル検査ツールの選定 (KA 1.2(3))」と「モデル検査のツール (KA 3)」を併せて参照することで，問題に即したツールを選択し検証を進めるための判断材料が得られる．

6. まとめ・今後の課題

技術者向けモデル検査教育を実施していた産・学・官の組織で協働し，カリキュラムの基礎付けとしてモデル検査の知識を MCBOK 2008 に体系づけてまとめた．

しかし，モデル検査は普及途上であるという事情から，MCBOK 2008 には検討が不十分な知識項目が残っている．特に KA1 の項目については文献も多くなく，改善の余地が大きい．今後，MCBOK の内容を見直し，質を高めていくことが必要である．

また，MCBOK 2008 自身は知識項目の樹形図であり，そのままでは利用に適さない．今後，モデル検査適用時のツールとして MCBOK が機能するように，知識項目間の関連，意義，文献などを整理した手引書を準備するなどの活動が必要である．

参 考 文 献

1) Abran, A., Moore, J. W., Bourque, P. and Dupuis, R.: *Guide to the Software Engineering Body of Knowledge 2004 Version SWEBOK*, IEEE (2004).
2) Edmund M. Clarke, Orna Grumberg, D. A. P.: *Model Checking*, The MIT Press (2000).
3) Holzmann, G. J.: *The SPIN model checker: Primer and reference manual*, Addison Wesley (2004).
4) Nishihara, H., Shinozaki, K., Hayamizu, K., Aoki, T., Taguchi, K. and Kumeno, F.: Model Checking Education for Software Engineers in Japan, *SIGCSE bulletin inroads*, Vol. 41(2), pp. 45–50 (2009).
5) Project-Management-Institute: *Guide to the Project Management Body of Knowledge*, PMI (2000).
6) The Joint-Task-Force-on Computing-Curricula, IEEE-Computer-Society, A.-f.-C.-M.: Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (2004).
7) 産業技術総合研究所システム検証研究センター: モデル検査初級編, ナノオプト・メディア (2006).
8) 産業技術総合研究所システム検証研究センター: システム検証の事例報告集 2008 年度版, 算譜科学研究速報 aist-ps-2009-004, 産業技術総合研究所 (2009).
9) 吉岡信和, 青木利晃, 田原康之: SPIN による設計モデル検証–モデル検査の実践ソフトウェア検証, 近代科学社 (2008).
10) 情報処理学会情報処理教育委員会, J07 プロジェクト連絡委員会: 情報専門学科におけるカリキュラム標準 J07.
11) 西原秀明, 青木利晃, 粂野文洋, 篠崎孝一, 田口研治, 早水公二: MCBOK2008:ソフトウェア開発のためのモデル検査知識体系, 組込みシステムシンポジウム論文集, pp. 49–57 (2009).
12) 青木利晃, 粂野文洋, 木下佳樹, 篠崎孝一, 高木理, 高村博紀, 田口研治, 中原早生, 西原秀明, 早水公二, 本位田真一, 渡邊宏: モデル検査の教育プログラム構築に向けて, 算譜科学研究速報 aist-ps-2008-012, 産業技術総合研究所 (2008).
13) 日本科学技術連盟, 日本品質管理学会: ソフトウェア品質知識体系ガイド.

KA1

- モデル検査のプロセス
  - 1.1 モデル検査の適用方法
    - 設計検証プロセス
    - 開発ツール一体型適用プロセス
    - ソースコード検証適用プロセス
    - デバッグ適用プロセス
    - アドホックな適用
  - 1.2 モデル検査の適用手順
    - 検査対象の選定
      - 適用工程／詳細化レベルによる選定（仕様／設計／コード）
      - システムの特徴による選定（状態遷移系／並行動作／通信）
    - 検査項目の設定
      - 検査できる性質（到達可能性／活性）
      - 検査の観点（仕様／固有知識）の獲得
      - 検査項目の具体化
      - モデル動作確認のための検査項目
    - モデル検査ツールの選定
      - モデル／検査内容による選定
      - モデルの複雑さによる選定
    - モデル設計
      - モデル化範囲／レベルの決定
      - 内部モデル／外部モデル
      - 非決定性・公平性
      - 検査項目との整合性
    - モデル実装
      - 専用プログラム記述
      - GUI入力
      - 開発用言語による入力
    - 検査項目の記述
      - 時相論理式による記述
      - プロパティパターン
      - アサーション記述
      - 観測変数のモデル化
    - モデル検査ツールの実行
      - 実行順序（モデル確認から重要な検査へ）
      - 状態爆発への対応（打ち切り、モデル変更）
      - 繰り返し実行
    - 反例解析
      - トレース分析（停止／ループ）
      - シミュレーション実行
      - 検査項目／モデルの不具合修正
      - 検査項目／モデルの変更
      - 対策要否の検討
  - 1.3 モデル検査のコスト
    - 導入（教育）コスト
    - 作業コスト
    - 効果
  - 1.4 モデル検査の実施体制
    - 専門チーム
    - 設計者／テスト技術者

KA2

- モデル検査の理論
  - 2.1 計算モデル
    - オートマトン
      - 有限オートマトン
      - 無限オートマトン
      - 時間オートマトン
      - ωオートマトン
      - 確率オートマトン
      - プッシュダウンオートマトン
      - ハイブリッドオートマトン
      - 非決定性
      - 公平性
    - プロセス代数
      - CCS
      - CSP
      - π計算
    - 項書き換え系
      - 項書き換え系
  - 2.2 時相論理
    - LTL
      - LTLの Kripke 意味論
      - 決定可能性と決定手続き
    - CTL
      - CTLの意味論
      - 決定可能性と決定手続き
    - 時相論理の表現力
      - 実時間論理
      - CTL＊
      - 様相μ計算
  - 2.3 モデルの縮小手法
    - モデル間の関係
      - 抽象解釈
      - Cone of Influence Reduction
      - Partial Order Reduction
    - モデルの最適化
      - 抽象化
      - スライシング
  - 2.4 モデル検査の原理
    - 記号モデル検査
    - On the Fly モデル検査
    - 有界モデル検査
    - 無限有界モデル検査

KA3

- モデル検査ツール
  - 3.1 モデル検査ツールの種類
    - 基本モデル検査ツール
      - Spin
      - Cadence SMV
      - NuSMV
    - プロセス代数のモデル検査ツール
      - LTSA
      - FDR
      - Mobility Workbench
    - 無限状態モデル検査ツール
      - SAL
      - Maude
    - 実時間モデル検査ツール
      - UPPAAL
      - KRONOS
    - 確率モデル検査ツール
      - PRISM
    - ハイブリッドシステムモデル検査ツール
      - HYTECH
    - モダンモデル検査ツール
      - BLAST
      - Java Path Finder
      - Bandera
      - VARVEL
      - SDV(SLAM)
  - 3.2 モデル検査支援ツール
    - 設計検査支援ツール
      - HUGO
      - vUML
      - charmy
      - UMLチェッカー
    - モデル検査ツールの使用支援ツール
      - Xspin
      - Lambda Trace
    - プログラム検査支援ツール
      - FeaVar(MODEX)

KA4

- モデル検査の適用対象
  - 4.1 アプリケーションシステム
    - web アプリケーション
    - グループウェア
    - 交通管理システム
  - 4.2 ミドルウェア
    - 例外処理
  - 4.3 ネットワーク
    - 遠隔操作
    - 機器間ネットワーク
  - 4.4 組込みシステム
    - メーター制御
    - 航空機機器制御システム
    - 船舶通信システム
    - センサー入力処理

IC カードセキュリティ開発における形式手法 SPIN の適用
Smartcard security development using formal method tool SPIN

（株）NTT データ　技術開発本部 SI アーキテクチャ開発センタ
市原尚久

アブストラクト
近年の IC 及び IC カード製品は、高度なセキュリティを実装している事が要求されており、関連企業は、Common Criteria（ISO/IEC15408）の認証を取得した IC カード製品を市場に提供する事で、これらの市場要求に対応している。日本においても、公共系 IC カードや電子パスポートなどが ISO/IEC15408 の認証取得を調達条件としている。当社においては、通常の評価クラス以外にも、形式的手法によるセキュリティの安全性証明が要求される ADV_SPM.3 を追加した評価にチャレンジし、アジア系企業として初めてこれに合格する事に成功した。本講演では、当社製 IC カード「Xaica-α64K」の ISO/IEC15408 認証取得において採用した形式手法ツール SPIN による、セキュリティの安全性証明の技術的アプローチ、及び、実際の IC カード開発プロジェクトへの形式手法を導入する際に考慮すべきポイントについて述べる。

Smartcard security development using formal method tool SPIN

15th of March, 2010

Naohisa ICHIHARA
NTTDATA Corporation

Copyright© 2010 NTT DATA Corporation

Contents

1. Background
2. Key to succeed
3. Modeling
4. Modeling framework
5. Example
6. Summary

Copyright© 2010 NTT DATA Corporation

1. Background

- CC evaluation experience (for smartcard)
  - (1) Smartcard "Xaica-alpha" for JUKI-card, EAL4+, DCSSI (France), CC v2.1, 2005
  - (2) Smartcard "**Xaica-alpha 64K**" for e-Passport, EAL4+ (includes **ADV_SPM.3**), CC v2.3, DCSSI (France), 2007
- Why Formal methods?
  - Technical challenge as R&D
  - Establish high quality security development

Copyright© 2010 NTT DATA Corporation

2. Key to succeed

- Preparation:
  - Understanding of ADV_SPM.3, AIS 34
  - Study of Formal methods
  - Technical Approach
    - Scope of modeling
    - Choice of tool (to modelize and prove)
    - Step toward the goal
  - Draw up a project
    - Milestone scheduling
    - Team formation
  - Negotiation with Certification Body, ITSEF
    - Tool choice, Approach, Modeling scope, Interpretation of CC requirements, ... etc

Copyright© 2010 NTT DATA Corporation

2. Key to succeed

- Technical approach
  - Scope of modeling
  - What we verify?
  - How we prove?
  - Step toward the formal modeling and verification

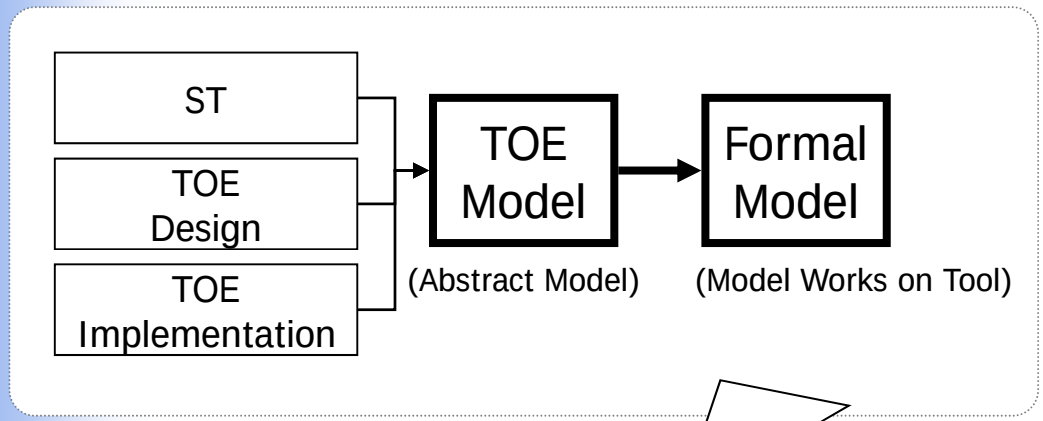
- Project management
  - Team formation (Developer, ST author, Formal modeler)
    Especially, “Bridge person” is required.

  - Formal Method Education
  - Internal Review
  - External Meeting/Review with ITSEF, CB

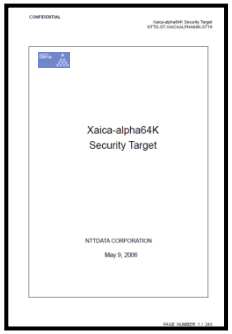
Copyright© 2010 NTT DATA Corporation

2. Key to succeed

- Internal Review (in detail)



*Model is correct, from **ST , TSP and CC requirement** points of view*



ST author

Developer

Formal modeler

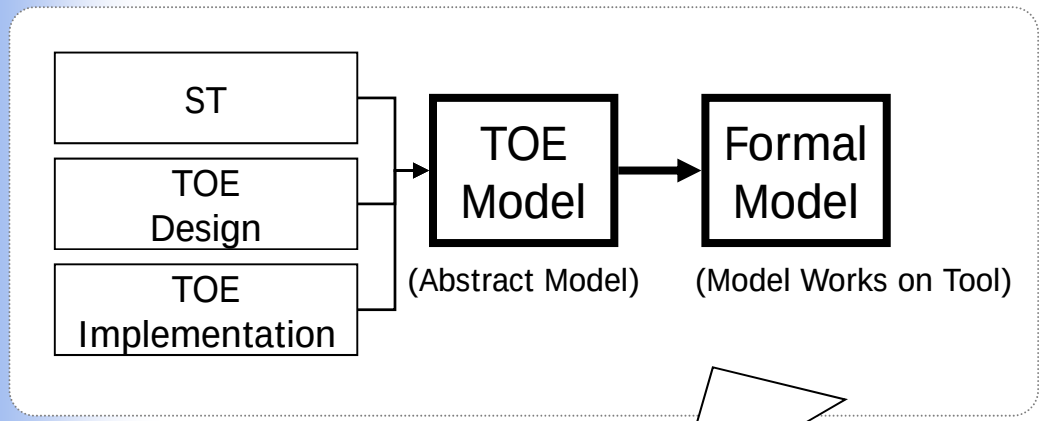
*Model is correct, from **TOE design and implementation** points of view*

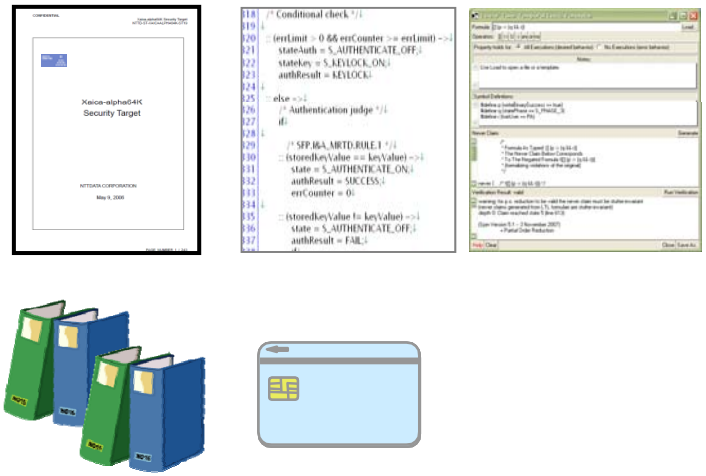
*Model is correct, from **formal and logical** points of view*

Copyright© 2010 NTT DATA Corporation

2. Key to succeed

- External Review (in detail)





Model satisfies requirements of ADV_SPM.3 as well as AIS34

Sponsor

ITSEF

CB

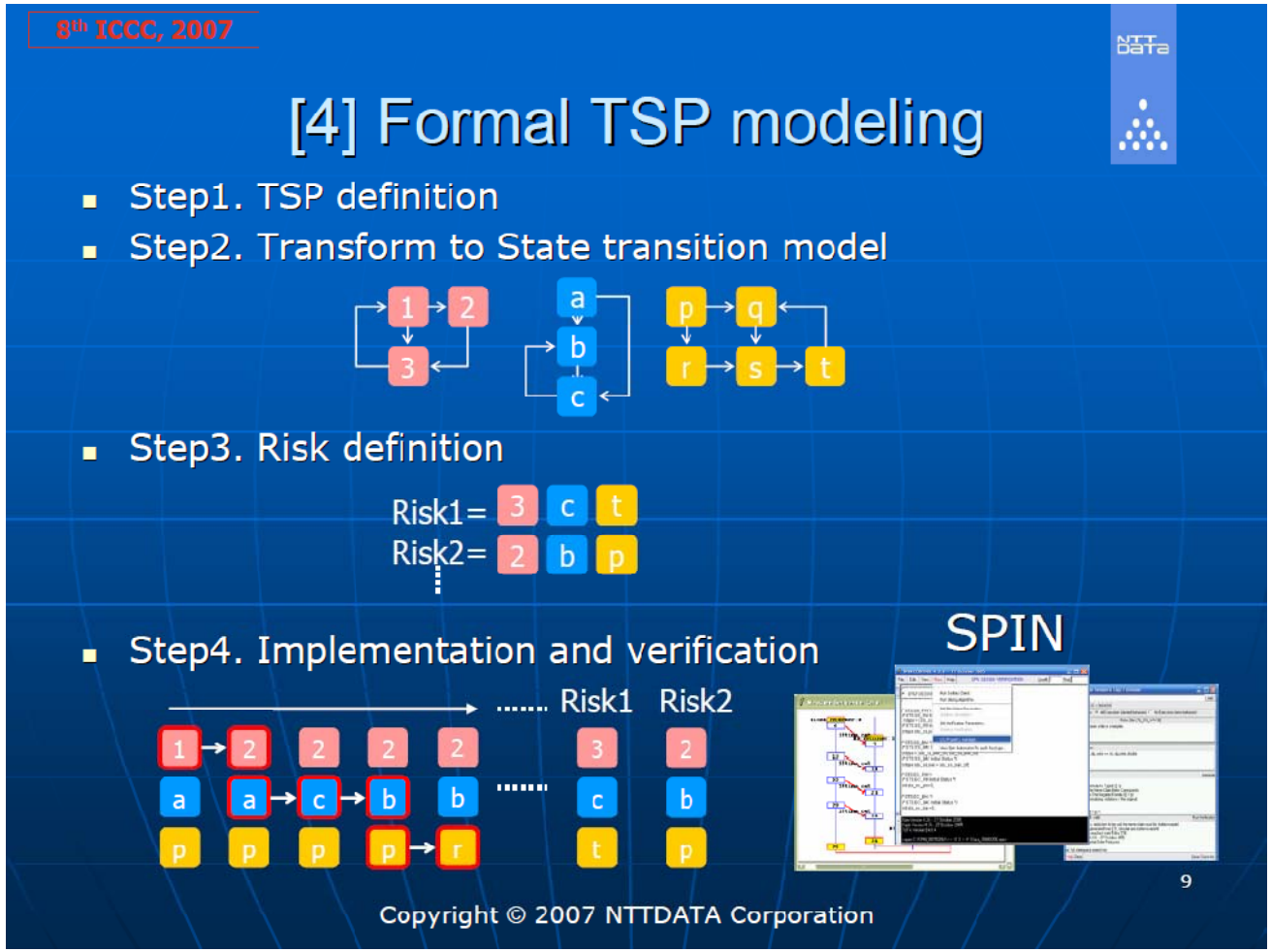
Recommends;
- Apply AIS34
- Understand the Formal method tool, approaches and models

- AIS34 as interpretation
- Formal method tool could be applied

Copyright© 2010 NTT DATA Corporation

3. Modeling

- Approach overview (and why SPIN/Promela?)



8th ICCC Presentation, Naohisa Ichihara, NTTDATA (2007)

Copyright© 2010 NTT DATA Corporation

3. Modeling

Copyright© 2010 NTT DATA Corporation

3. Modeling

Security Target (ST) → **TOE Model** → **Formal Model**

- TOE Security Policies
- Intended Usage
- Security Problem definition
- Security Objectives
- Security Functional Requirements
- Design Documents
- TOE implementation

Invariant conditions as for security

e.g. "Data1 is only accessible by User1", "Admin authentication is available only in the INITIAL_MODE", "Once Key3 is used for successful authentication, it is locked", ...

Business scenario, assumptions and threat

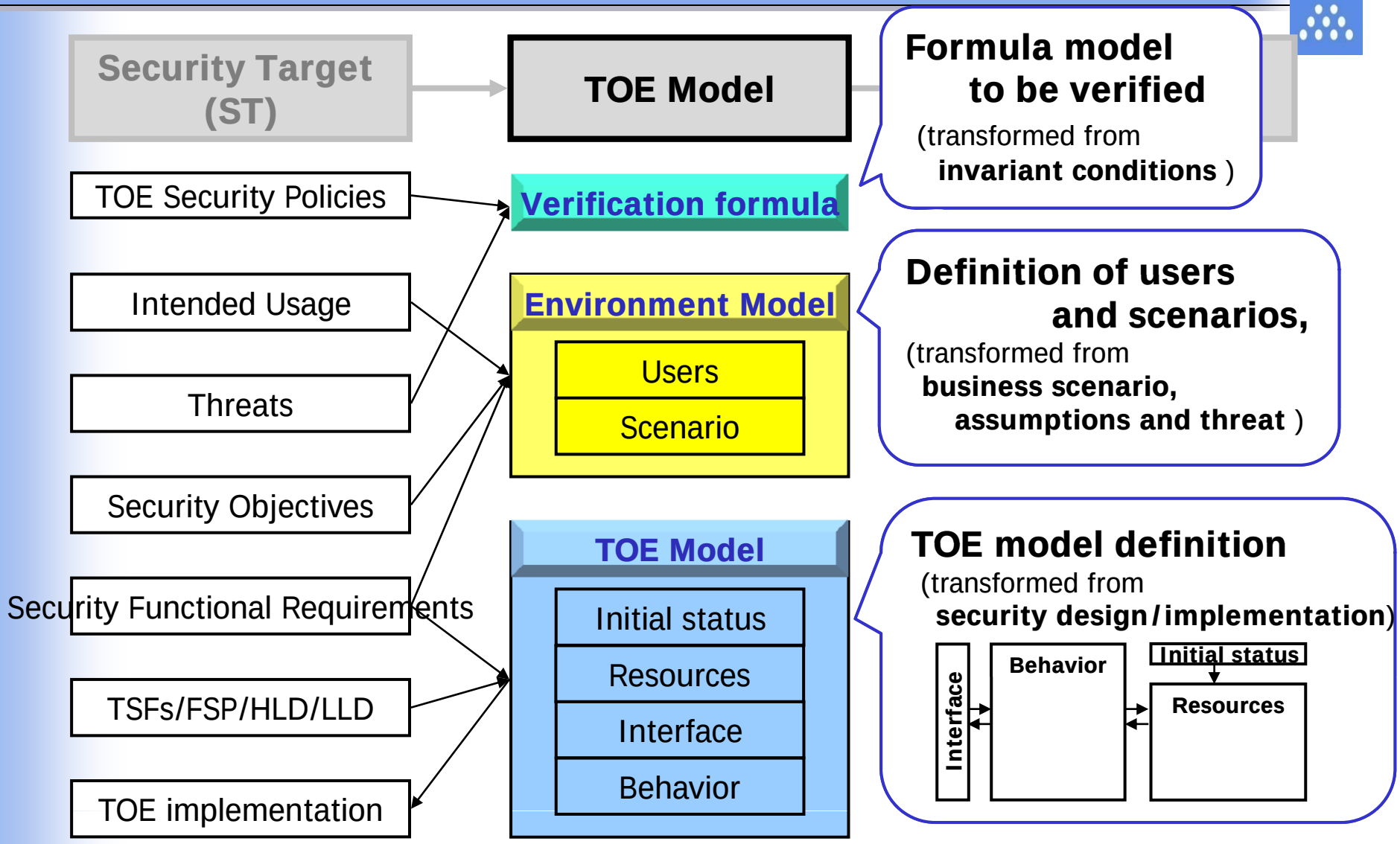
e.g. "the User can change his profile Data (Data2) after PIN verification" , "As a threat, an attacker may try to get Data1", ...

Product behavior definition (REQ, design) and implementation,

addressing above Security problems & policies

Copyright© 2010 NTT DATA Corporation

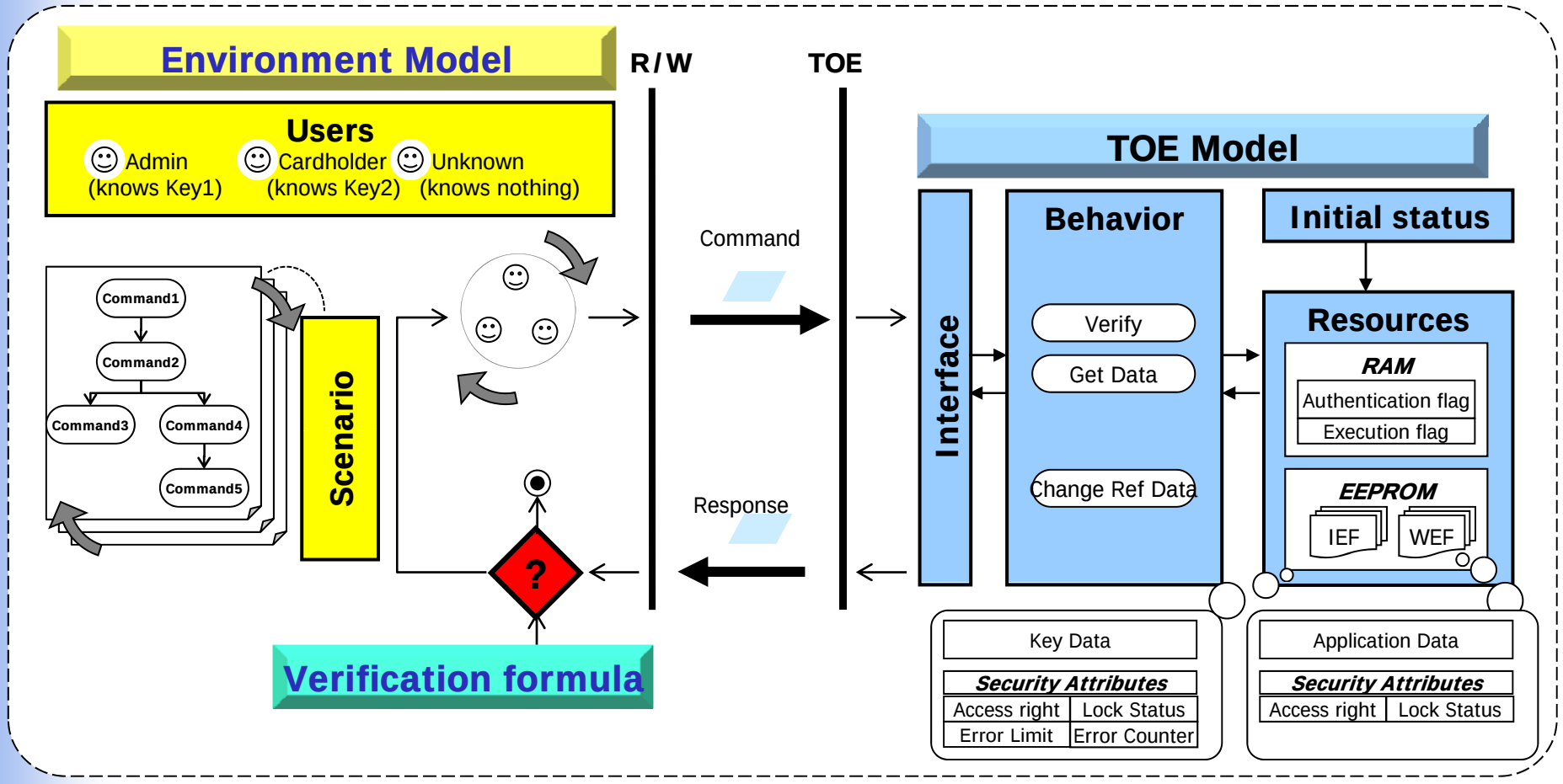
3. Modeling



Copyright© 2010 NTT DATA Corporation

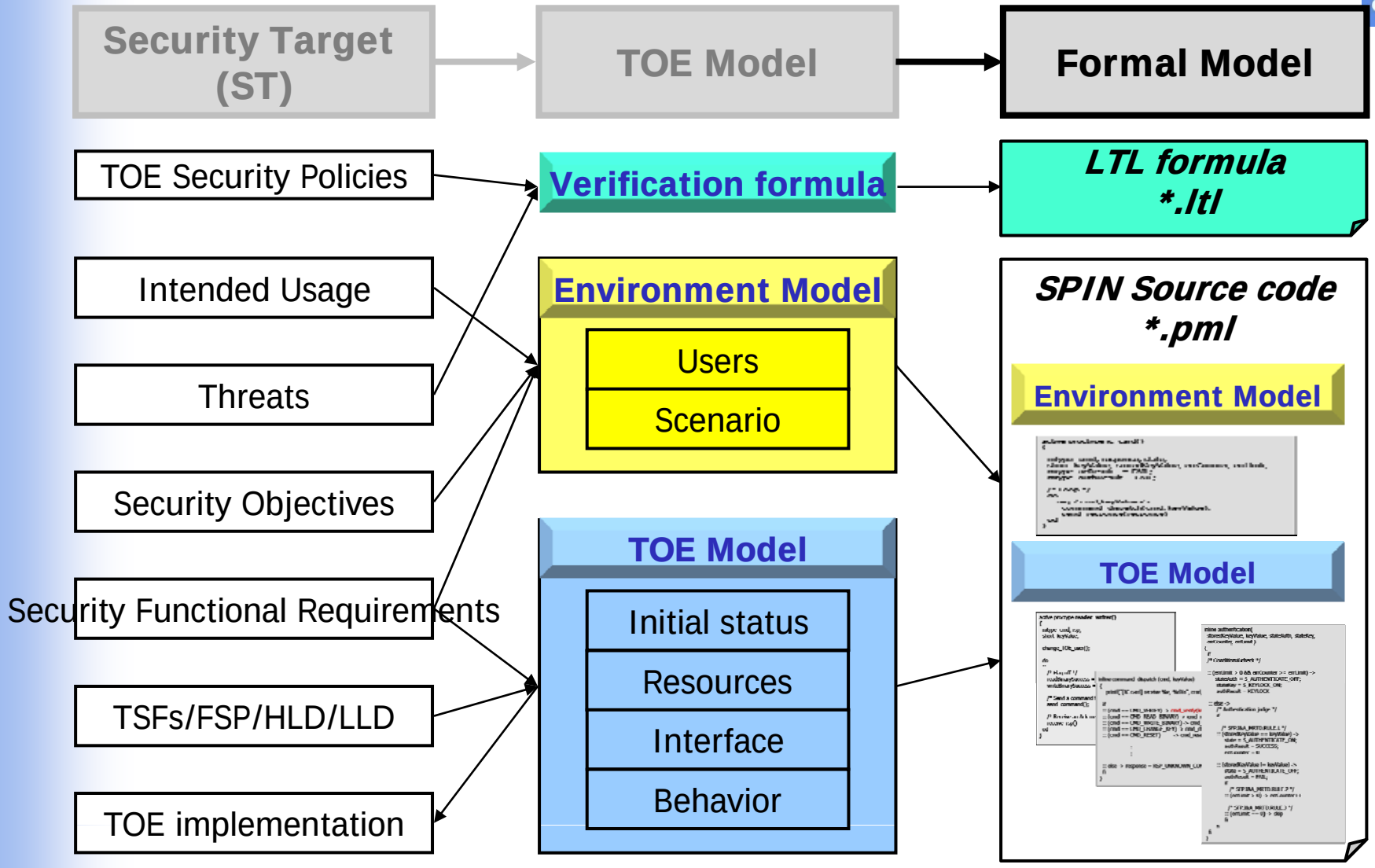
3. Modeling

Security Target (ST) → **TOE Model** → Formal Model



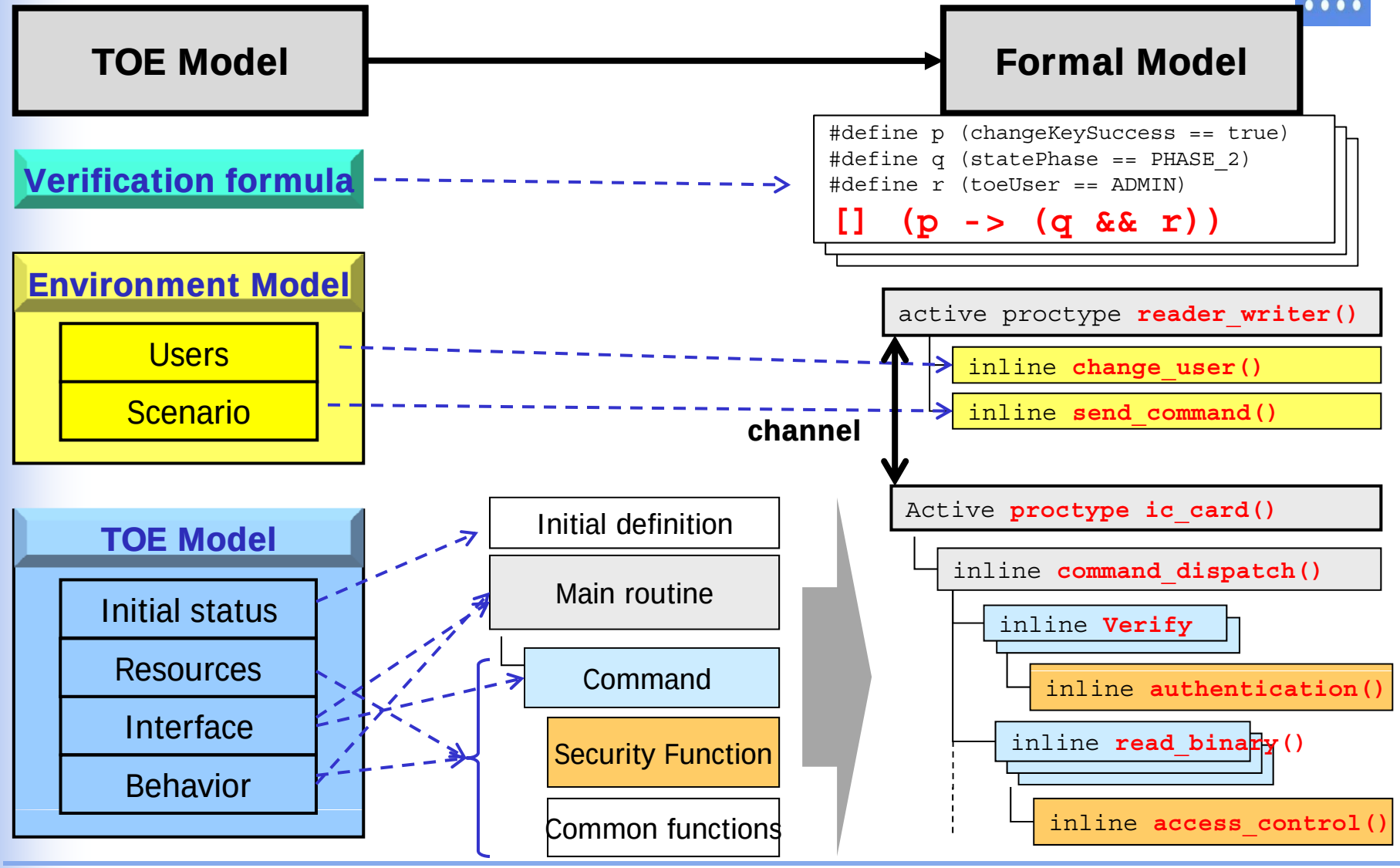
Copyright© 2010 NTT DATA Corporation

3. Modeling



Copyright© 2010 NTT DATA Corporation

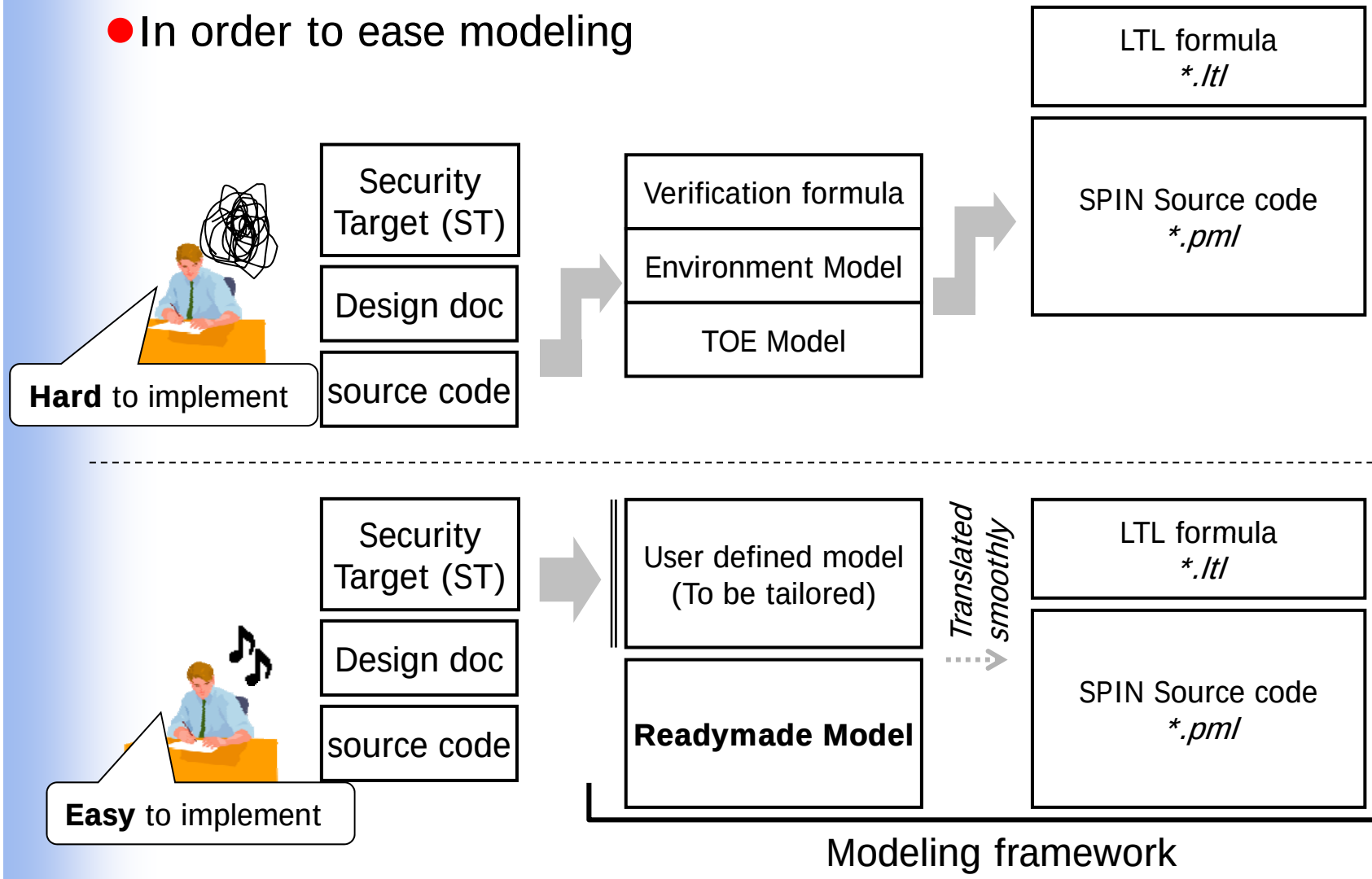
3. Modeling



Copyright© 2010 NTT DATA Corporation

4. Modeling framework

- In order to ease modeling



Copyright© 2010 NTT DATA Corporation

4. Modeling framework

	Common	R/W		Smartcard	
		Readymade	User defined	Readymade	User defined
Verification formula	pattern	-	-	-	-
Behavior	Channel	Main routine Send command Receive response User change	Scenario	Main routine Receive command Dispatch Send out response Commands Authentication Access Control File access	Behavior of (additional) user defined commands
Users	-	Unknown user	(Intended) users Key known by user	-	-
Interface	-	Command with or without value Response with or without value	(additional) user defined commands with or without value	Command with or without value Response with or without value	(additional) user defined commands with or without value
Resources	-	-	Environmental status	Keys, Files with Value, Error Limit	Value, Error Limit
Initial Status	-	-	Initial user Initial environmental status	Error Counter, Lifecycle status	Error Counter, Lifecycle status

Copyright© 2010 NTT DATA Corporation

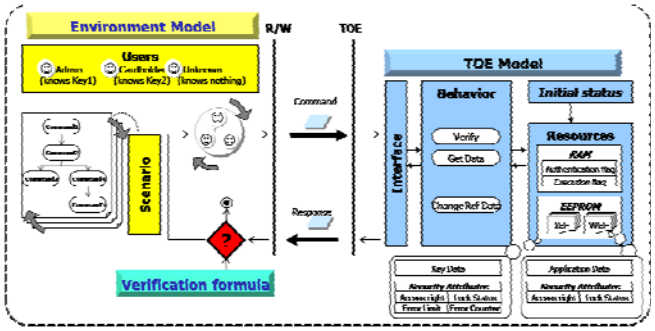
5. Example (SPIN)

Confidential
(to be presented)

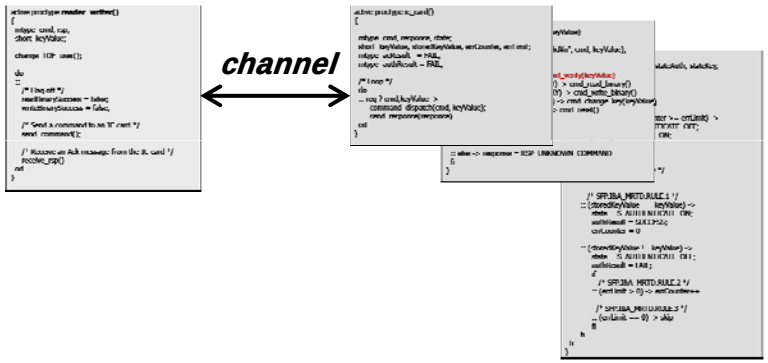
Copyright© 2010 NTT DATA Corporation

6. Summary

- **As a Result**
  - 3 members, 2 years (including technical training)
  - 0.5K steps by SPIN
- **Technical approach**
  - **Why SPIN?** Easy for developer, although there is limitation…
  - **What we prove?** TSP (= invariant conditions as for security concerns)
  - **How we model and implement?**




```
#define p (changeKeySuccess == true)
#define q (statePhase == PHASE_2)
#define r (toeUser == ADMIN)
[] (p -> (q && r))
```



Copyright© 2010 NTT DATA Corporation

6. Summary

- Project Management in development
  - Short training for “Bridge person” (as described above)
  - Team formation, especially assigning;
    - “**Bridge person**” (from development team), who understand
      - the product design / implementation (deeply)
      - the security problems and security policies (so well)
      - the concept of “modeling” and how to modelize
      - the concept of “formal method”
    - “**Formal method modeler**” who is an formal method expert, and understands
      - the product specification (overall)
      - how to transfer the model to formal method implementation
    - Either must determine and identify the security policies to be verified
      - Needs to identify the critical condition of the product
      - Needs to define the scope of verification in order to avoid meaningless verification

Copyright© 2010 NTT DATA Corporation

7. Current works and issues

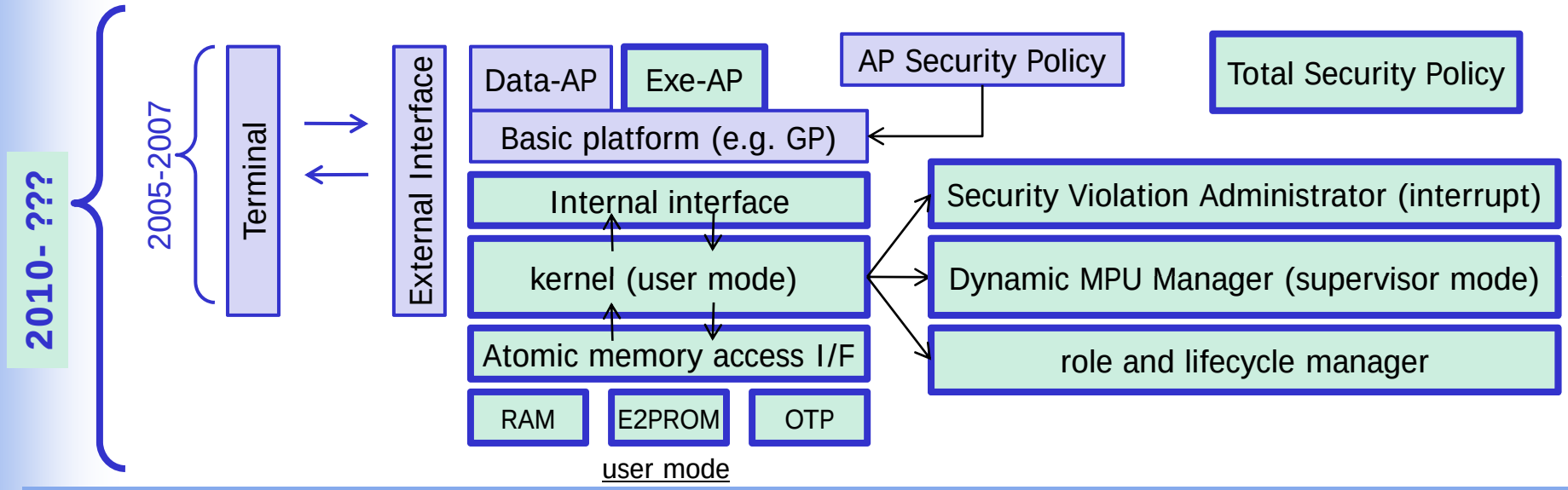
●Smart card security

– Attack methods for IC and Smart Card are **still evolving** as well as IC and SC technologies

e.g. combination of glitch with laser beams, perturbation attacks, EMA rather than SPA/DPA, …

– IC & SC vendors, evaluators **needs to follow** "state-of-the-art" attacks and countermeasures

– Important to determine if SC security design or implementation has critical vulnerability or not, **in early stage** of development



Copyright© 2010 NTT DATA Corporation

Questions?

ichiharan@nttdata.co.jp

Copyright© 2010 NTT DATA Corporation

ソフトウェア開発における形式手法導入に関する課題と解決アプローチ

石　黒　正　揮 †

ソフトウェアの信頼性等に対する要求は高まる一方で、開発現場では、従来のテストケースに基づく方法では解決できない多くの問題を抱えている。このような問題の一部に対する解決手段として、形式手法は従来のテスト法の弱みを補う有効な特徴を備えている。欧米においては形式手法の産業応用が進んでいるが、国内の産業界では、形式手法の導入が進んでいない。本稿では、日本の開発現場で形式手法の導入が進まない原因とその解決アプローチについて議論し、形式手法の普及促進を目的として作成中の「フォーマルメソッド導入ガイダンス」の特徴について紹介する。

Approaches to Employing Formal Methods in Software Development

Masaki Ishiguro †

While the demand of improving dependability of software is increasing in society, managers and engineers in software development vendors are faced with difficult problems which cannot be sovled by traditional test methods alone. Formal methods have effective features which can cover up weakness of traditional testing. Formal methods are increasingly employed in industry in Europe and US, it is not the case in Japan. This article aims at discussing difficulties and their possible solutions in employing formal methods in industry. It also introduces "A Guidance of Empolying Formal Methods in Sofware Development" for promoting formal methods in industry in Japan.

1. はじめに

情報化が進む社会においてソフトウェアの信頼性および安全性に対する要求は高まる一方であるが、開発現場では、従来のテストケースに基づく方法では解決できない多くの問題を抱えている。このような問題に対して、形式手法が全ての解決策を与えられる訳ではないが[1)]、従来のテスト法の弱みを補う有効な特徴を備えている[2)]。従来のテスト法と形式手法の併用を前提として、形式手法の適性を理解して、ソフトウェア開発全体のうちの適した部分に適用[3)] することで経済的で、かつ高い効果が得られると期待できる。

現在、形式手法は欧米において産業応用が進みつつあるが、国内では普及が進んでいない。このような背景のもと、経済産業省の新世代情報セキュリティ研究開発事業において、開発現場へ形式手法の普及を図ることを目的として「フォーマルメソッド導入ガイダンス」の開発を進めている。本稿では、形式手法の導入に関わる障害について議論し、その解決策としてのガイダンスの主要な部分について紹介する。

† 三菱総合研究所, Email： masa @m ri.co.jp
Mitsubishi Research Institute, Inc

以下、第 2 節では、ソフトウェア開発に係わる課題、第 3 節では、形式手法の意義と効果、第 4 節では、形式手法導入に係わる障害と解決策、第 5 節では解決策としてのガイダンスの主要な内容の紹介、第 6 節では、ケーススタディの概要紹介、第 7 節では、今後の展望について述べる。

2. 安全なソフトウェアの開発に係わる課題

ソフトウェアの信頼性および安全性☆に関して、開発現場において、容易には解決できない課題が多く残されている。　それらの課題は多岐にわたるが、開発現場の声やソフトウェアの安全に関する研究分野において挙げられる課題[4)] のうち主なものを挙げると以下のようになる：

☆ モデル検査等の形式手法にける技術用語として「安全性 (Safety)」という用語が用いられているが、ここでは「安全性」を一般的な広い意味で、「事故や損害が発生しないこと」を意味するものとして用いる。信頼性は、「期待された機能を提供すること」を意味するものとして用いる。安全性は、信頼性を前提に議論することが重要である。たとえば、「飛ばない飛行機は安全である」という例を考えると、飛行機は「飛ぶ」という本来の機能が提供され信頼性が満たされて、初めて「安全」であるかどうか議論することに意味を持つことが分かる。

開発現場において、もっとも多く挙げられる問題は、従来のテストケースに基づく方法☆では、ソフトウェアの正しさを保証することができないという点である。特に分散並行システムにおける並列処理のタイミングに係わる不具合は、いくらテストを繰り返しても全ての場合を再現する保証が得られないためである。また、システムが厳密にモデル化されていなければ、系統的に網羅的なテストケースを生成することも困難である。さらに、無限ループを含むシステムの振舞いを検証性質としてテストケースにより表現することは困難である。また、従来のテスト法では、検査結果に対してどの程度の安全性が確保されるのか根拠が得られないという問題がある。そのため、製品に対する安全性に確信が持てないまま出荷される例も少なくない。

検証すべき性質が予め想定される場合においてさえ、上記のような問題を抱えているが、不具合の可能性さえも気が付かない場合も多い[5)]。航空・宇宙分野、自動車などのクリティカルシステムや金融システムなどにおいては、このような想定外の不具合を如何に防ぐかということが喫緊の課題となっている[4)]。

一方、情報システムに係わる多くの事故は、ソフトウェアが仕様通りに正しく動作している時に発生している事が指摘されている[4)]。これはシステムを使う人にとって、仕様自体が妥当でない場合などに発生する。一例として、緊急事態に、システムのオペレータに大量の情報が提示されると、人は何が起きているか理解するまで何も行動できなことなどを挙げることができる。仕様の妥当性を確認するためには、開発上流工程において、要求仕様を明確に表現し、システムに係わる利害関係者の間で、仕様をレビューすることが不可欠である。発注者と開発者の間での合意形成のための手段としても要求仕様の明確化が重要である[6)]。

以上のような課題は、例に過ぎないが、開発現場におけるこれらの深刻な問題に対する解決策が強く求められている。

3. 形式手法の意義と効果

第 2 節に挙げた課題を解決する手段として形式手法が有効な特徴を備えている。形式手法は、対象システムと検証すべき性質を厳密に (数学的に) 定義することで、性質に関して網羅的な検査や、論理的な証明を行うことが可能になり、検証性質が満たされることを論理的に保証するものである。従来のテストケースではいくらテスト回数を繰り返しても、安全性は保証するには至らない。テストケースを用いたプログラムの実行を伴う検査では、並行プロセスのタイミングに係わる再現性に依存する性質が満たされることが保証ができないためである。従来のテスト法は、バグを発見することはあっても、ソフトウェアの安全性などを保証する点においては無力といえる。従来のテスト☆☆のような動的分析に対して、形式手法は仕様を静的に分析する点が本質的に異なる。

一方で、形式手法の適用においては、検証までは行わ無いまでも、システムを厳密に記述する過程で、設計書の曖昧な部分や矛盾する部分を発見したりするなどの効果も得られる。

形式手法は、このような特徴を持った強力な手段を提供するが、一つの手法で全ての問題を解決できる万能な手法ではない。形式手法を用いてシステムを厳密に記述することは、プログラムを書くことと同じぐらいエラーを起こしやすい[4)]。いくら形式手法により厳密な検証ができるといっても、記述したモデルが正しくなければ意味が無い。形式手法は、ベースとする数学理論に応じて様々な手法が開発されており、それぞれ特徴と適性が異なっている。形式手法の導入においては、対象システムを厳密にモデリングするため、上流工程において工数が増加することは避けられない。ソフトウェア開発全体に形式手法を適用するよりも、ソフトウェアに求められる検証ニーズとコストのバランスを考慮し、検証目的に適した形式手法を選択し、適切な部分に適用することで高い効果が得られている[7)]。

4. 形式手法導入に係わる障害と解決策

形式手法は、欧米を中心に産業応用が進みつつあるが、国内ではまだ普及が進んでいない。形式手法の普及に係わる障害とそれを解決するために本研究で作成中の「フォーマルメソッド適用ガイダンス」による解決策および既存の関連文献の関係をまとめたものが表 1 である。

普及の障害は、大まかに「管理面」と「技術面」に分けられる。管理面については、形式手法の導入意思決定者に対するコストと効果に関する情報が不足している点が挙げられる。ガイダンスでは、効果の規模感を示す費用対効果や開発者が実感できる具体的な効果を示すことで、管理者レベルから形式手法の導入を検討するための情報を提供する。これに関しては既存の文献はほとんど存在しない。次に、形式手法に対する認

☆ ホワイトボックステストやブラックボックステストなどを指す。

☆☆ ブラックボックステストの場合

表 1　形式手法の普及の障害と解決策

普及の障害		ガイダンスの解決策	既存文献の例
管理面	導入意思決定に必要な情報の不足	●費用対効果分析の提示 ●効果影響分析の提示 ●具体的な効果を例示 （効果の見える化）	（なし）
	形式手法に対する認識と実際の適性とのズレ	●形式手法の導入における留意点の提示 （意識のギャップを解消）	●Bowen: Ten Commandment Revisited ●NASA Guidebook
	形式手法選択の情報の入手の困難	●手法選択のための解説文献のガイド （既存の文献を活用して、適切な手法の選択を支援）	●ソフトウェア科学 解説論文 ●日経BP組込みソフトウェア形式手法特集
技術面	モデリングに対する理解の困難性	●モデリング･アプローチの枠組みを整理 ●形式検証プロセスの具体的な手順化 （既存の文献を補強）	（SPINによる設計モデル検証）
	状態爆発等の技術的障害（モデル検査の場合）	●検証性質を意識した抽象モデリングの手順 ●状態爆発の発生原因と解決策の例の提示 （既存の文献を補強）	(Ben Ari, Principles of SPIN Model Checker)

識と実際の適性のズレが上げられる。形式手法は、100以上の異なる手法の総称であり、手法により適性がことなる。適した手法を開発全体の一部に適用することで高い効果が得られている[7]。あらゆる目的で他の形式手法より優れている唯一の手法は存在しない。限られてはいても達成可能な範囲から検証を進めることが重要である。

このような形式手法に関する認識について、海外の文献を中心に重要な点をまとめている文献を日本の開発者にも示すことで、効果的な形式手法導入の方法を示す。

次の障害は、多数の形式手法の中から自身の目的に適した手法を選択するための情報が入手しにくいことが上げられる。現在、世の中に分散している形式手法の解説文献をカタログ化して示すことにより、限られた時間で、幅広い手法の中から自分に適した手法を選択するための情報を提供する。一方で、技術面の障害には、モデリングの理解の困難さが挙げられる。ガイダンスでは、モデリングのアプローチを整理し、形式検証プロセスを要素化して具体的な手順を示すことにより、初学者の理解を促進する。モデリングの理解は、大学などでの形式手法の科目の有無によるところが大きい。若いころ経験したことがない概念を大人になってから学ぶのは厳しい。柔軟な吸収欲をもつ学生時代に形式手法を学ぶことは効果が大きい。

もう一つの技術面の障害として、モデル検査の状態爆発などの障害が上げられる。ガイダンスでは、検証性質を意識したモデリング手順を示すことで、モデルの抽象化を進めたり、状態爆発の原因と解決策の例を示すことで、障害を回避する手段を示す。

5. 形式手法の導入におけるポイント

現在開発中の「フォーマルメソッド導入ガイダンス」☆のうち、主なポイントを紹介する。

5.1 モデルおよび形式検証の概念の整理

形式手法の導入において重要な事の一つにモデルの概念を理解する事が挙げられる。ガイダンスでは、この点について一般の技術者向けに分かりやすく説明する。

「モデル」という用語は、UML のクラス図など様々な場面で用いられるが、形式手法における「モデル」の意味は、UML のモデルと関連性はあるが、必ずしも一致しない。形式手法の「モデル」は、対象システムの振舞いや性質を数学的に厳密に定義するもので、要求はモデルによって満たされるべきものという関係として理解される[8]。システム検証の視点で言えば、要求は、システムと環境によって満たされるものであり[9]、システムと環境を合わせたものを数学的に定義したものがモデルである (図 1)。

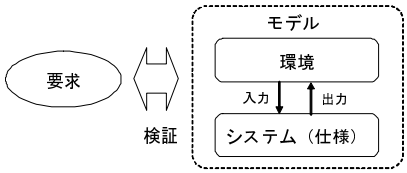


図 1　要求とモデルの関係

形式手法における検証方法は大まかにモデル検査と定理証明に分けられる。それぞれのモデルは関連するが、直感的に定義すると以下のようになる☆☆：

☆ ガイダンスの構成は、第 A.1 節に示す。
☆☆ 1),2) 等の定義を参考に直感的な表現に修正。

モデルとは?

(1) システムの振る舞いを計算するための数学的な表現で、状態遷移システム（オートマトン）などにより表現される。

(2) 性質の集合を論理式などによって数学的に表現するにより、それらを満たすシステムとして定義される概念で、代数などを用いて表わされる。

モデルを定義することにより、対象システムに関する様々な性質（論理式）の真偽が決まる。「検証」とは、モデルが要求を満たすことを示すことである。以下では主に、主にモデル検査を想定して留意点をまとめる。

5.2 形式検証の上流工程で意識すべき事

ソフトウェアの検証を行うためには、検証対象と検証基準の 2 者を明確に定義しなければならない。当然のことのように聞こえるかも知れないが、ソフトウェアの開発現場では両者が明確化されておらず、曖昧なままにテストが行われている場合が多い。検証対象の仕様は、システムの設計仕様に相当し、モデリングにより厳密に定義されるものである。検証基準は、要求仕様を形式的に記述することにより定義される。要求仕様は、「何を実現するか (What)」を定義するもので、設計仕様は、「どのように実現するか (How)」を定義するものである。この両者を明確に区別して、検証を行わなければ、一体何を検証しているかさえ判然としなくなる (図 2)。

要求仕様の記述は、実装に依存しない発注者のニーズに限定すべきであり、発注者と開発者の間で合意を形成するための重要な成果物となるものである[6)]。発注者やドメインの専門家によって妥当性を確認されていない仕様は有用ではない。仕様の欠陥がソフトウェアの欠陥のもっとも大きな要因となっているためである[10)]。このことからも、要求仕様は、開発者以外のステークホルダーによって理解できるレベルで記述することが重要である。形式手法を導入することにより、要求仕様と設計仕様の区別を明確化することが強いられる点も効果の一つと考えられる。

5.3 検証プロセスの要素化と手順化

設計検証プロセスは、入出力となる成果物に着目することによりモジュール化することができ、各サブプロセスを手順化することで習得しやすくなる。設計検証プロセスの途中で生成される中間成果物を保存管理することは、要求仕様は、最終的なソフトウェアによって満たされているか確認するという要求仕様のトレーサビリティを確保する上で有用な情報となるため、開発過程で中間成果物を保管管理することが重要である。ガイダンスでは、図 3 のように設計検証プロセスにおいて作成される成果物を入出力と見なして、設計検証プロセスを要素化して、各要素プロセスに対して具体的な手順を示すことで、形式手法の導入を支援する。

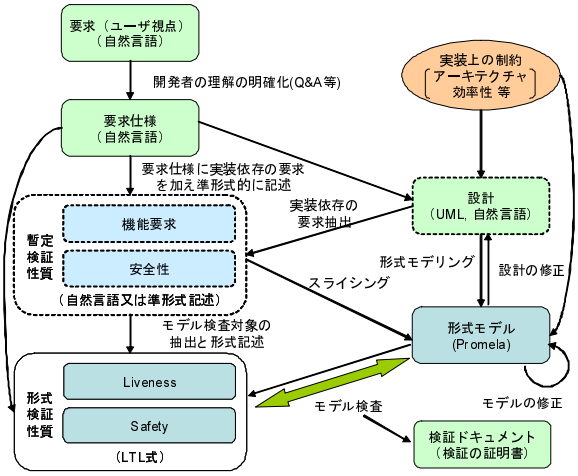


図 3 設計検証プロセスの要素と入出力 (中間成果物)

5.4 費用対効果分析

形式手法導入に係わる費用対効果全体を定量的に評価することは困難である。実際、定量的な費用対効果に関する文献はほとんど発表されていない。本研究では、形式手法の効果の全体像を把握するために、考えられる効果を可能な限り洗出し、その関係を整理するとともに、いくつかの効果についてはその規模感を把握するための定量的な評価を行った。ここで、定量評価を行う目的は、経済的な効果を正確に測ることが第一義では無く、これまで効果の規模さえ見えていない状況で、組織管理者に対して効果の大まかな目安を示すことである。このような情報が無いことが、組織として形式手法への取組み決定するための障害となっており、結果として個人レベルの形式手法の取組むなど、個人に負担を強いる状況となっていた。

形式手法のツールはフリーソフトウェアとして公開されているものが多く、導入コストは、主に形式手法の学習と具体的な対象システムのモデリングや形式検証のための工数として見積もられる。

形式手法の効果の全体像をまとめたものが図 4 である。この全体像は、現時点で必ずしも効果の全体を網羅しているとは限らないが、主な効果を一通り示すことを目指したものである。

効果は、大まかに「直接効果」、「ユーザ・開発現場の

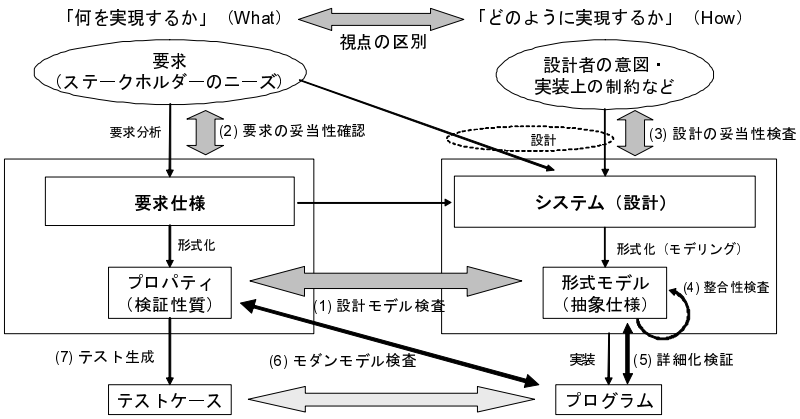


図 2　要求仕様と設計仕様および形式検証の関係

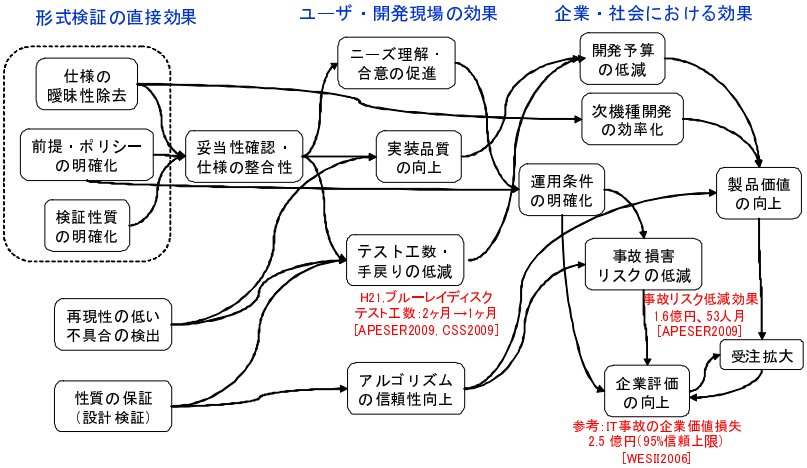


図 4　形式手法の効果の全体像 (案)

効果」、「企業・社会における効果」に分けられる。このうち、定量的評価に関しては、本研究では、ブルーレイディスクのミドルウェアに対する形式検証により、テスト工数が 2 人月から 1 人月に半減させられる可能性について評価した。また、製品の出荷後に不具合が発見されることによるリスクを低減させることによる効果を、1.6 億円、53 人月と評価した[5)]。

さらに、著者らの研究[11)] では、IT システムに係わる事故により企業に対する信頼の毀損、顧客離れにより企業のブランド価値を含む企業価値全体が減少することが統計的に有意であること (有意水準 5%) で示し、国際的な情報セキュリティ経済学の分野で検証を経ている。たとえば、いくつかの前提が必要であるが、企業の市場価値が 500 億円程度の企業が IT 業務障害で事故を起こした場合、2.5 億円の企業価値が損なわれることを示している。これらの評価結果からは、形式手法導入の効果は、テスト工数が削減されること以上に、ソフトウェアの信頼性が向上することで、事故リスクが低減されたり、企業価値の損失が回避されることの方が大きい可能性があることが示された[5)]。

6. ケーススタディの紹介

ガイダンスにおいては、実システムに対する形式検証のケーススタディを例にして、形式検証の方法を説明する。ここでは、実際に行ったケーススタディ概要を紹介し、そのうちの IC カード認証ベース入退室管理システムについて特徴を示す。

6.1　ケーススタディの概要

本研究では、実システムに関するケーススタディを 2 例実施した。その概要は、表 2 の通りである。

6.2　対象システムの概要

入退室管理システムに関して説明する。対象システ

表 2　ケーススタディの概要

事例	検証観点	検証性質 (例)	検証結果・不具合発見等	プロセス数
ブルーレイディスク	ディスク操作 API の制御アルゴリズムの安全性	常に要求された API は実行される。常に API が実行された後待機状態に戻る。	API 要求と割込みの同時発生に関わるデッドロックの検出。再生 API 要求の不実行の発見	6
入退室管理システム	仕様の妥当性確認、運用ポリシーの明確化	待合室に入った人はいづれは閲覧室に案内される。閲覧室が空室の時以外決して案内は変更されない。	入室待ちリストのバッファ溢れの発見。	6

ムは、図 5 の通り、前室、閲覧室 A、閲覧室 B、金庫室から構成される入退室管理システムである。利用者は、IC カードによる登録が行われ、各部屋を IC 認証を経て移動することができる。前室に入った利用者は、案内サービスにより、閲覧室 A または B に呼出されることで、閲覧室に入室可能となり、閲覧室内の IC カード認証を経て金庫にアクセスできる。各部屋には、人感センサーが設置され、人の有無は検知できるが人の人数は検知できない。このようなシステムにおいて、案内システムは、安全に人の案内を行い金庫の安全性を確保することが求められる。

6.3　ケーススタディの狙い

本ケーススタディの特徴は、利用者の位置とシステムが認識できる利用者の位置にズレが生じるという点である。部屋の移動は、扉の開閉ごとに同時に複数人の出入りが可能であり、人感センサーは人の人数を検知できないため、システムが認識できる利用者の位置は必ずしも正確とは限らない。このような状況の下で、金庫の安全性という要求を満たすためには、システムの仕様だけでは十分でないことが予想された。本ケーススタディでは、人の移動に関して運用ポリシーとして定め、移動を制限することにより金庫の安全性を示す。このようにシステムの仕様に加えて、利用者に対する運用ポリシーを形式仕様により明確化すれば、発注者が運用時に守るべき条件が示され、また、運用ポリシーがサービスとして妥当であるかどうかの検討も可能になるなどの効果がある。

6.4　モデルの概要

ケーススタディにおけるモデルの構成は図 6 の通りである。ここでは、モデル検査ツール SPIN による並行プロセス、チャネル、グローバル変数の関係を示している。

各プロセスには、それぞれプロセスの振舞いを定義する状態遷移システムを定義している。

6.5　検証性質と検証結果

本ケーススタディに関して、システムに求められる要求仕様、検証性質などを示したものが表 3 である。

要求仕様は、発注者の視点で理解できるシステムに対する要求を示している。検証性質は、SPIN によってモデリングしたシステムを前提に、時相論理式で表現するための性質を自然言語で書いたものである。これらのうちケーススタディで実際に検証した性質は、右端の列に丸で示している。本ケーススタディでは、7 プロセスから構成されるシステムに対して最大で 12 億状態のモデル検査を行った。

7.　おわりに：今後の展望

形式手法は、従来のテストでは実現できない効果をもたらすが、従来のテストを丸ごと置き換えるものではない。数多くの形式手法には、それぞれ適性があり、コストを考慮しつつ適性を見極めてソフトウェア開発全体の一部に適用することで、高い効果が得られる場合が多い。

システムがある要求を満たすことを論理的に保証するためには、システムを厳密にモデリングすることは避けて通れない。形式手法を適用する際の本質的な難しさは、形式手法ではなく、問題そのものの難しさによると言える。

形式手法には、難しい理論もあるが、理論を知ることと実際の検証を行う能力は別である。検証の実践では、ツールの機能を理解して、モデリングに関する実用的なノウハウを身につけることが重要である。

謝辞　本研究は、経済産業省新世代情報セキュリティ研究開発事業において実施した成果をまとめたものです。本稿作成に共同研究者である日立ソフトウェア 田中一之氏、NTT データ 梅村晃広氏、早稲田大学 岸知二教授から貴重なコメントを頂きましたことに感謝いたします。また、研究の推進にあたり「フォーマルメソッド適用ガイダンス検討委員会」の座長荒木啓二郎教授および委員の方から貴重な助言を頂きましたことに御礼申し上げます。

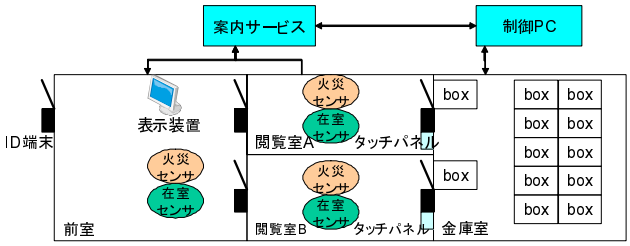


図 5　対象システムの概要

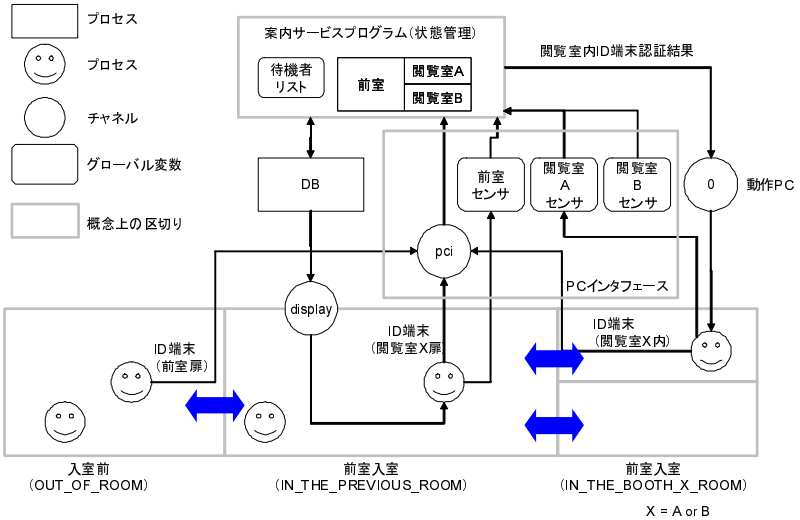


図 6　モデルの構成

表 3　要求仕様と検証性質

分類	要求仕様	No.	検証性質	備考	検証対象
ライブネス	IDカードを登録している人は、前室に入ることができる			簡略版モデルでは、全ての閲覧者は登録済み(前提1-5)	×
	前室に入った人は、必ず案内される	L01	前室に入室した人は、いつか必ず案内される	−	○
	閲覧室の扉認証は、閲覧室内が空室なら誰でも成功する			センサを閲覧者のモデルに組み込んでいるため	×
	案内された人は、一定時間、閲覧室の金庫ボックス取り出しの認証を自分のIDカードでパスできる。	L02	閲覧室に案内された人だけが、タッチパネル脇ID端末で、いつか必ず認証OKとなる	簡略版モデルでは、時間を取り扱っていない(前提1-1)	○
		L03	閲覧室から退出した閲覧者は、いつか必ず待機者リストから削除される	システム構築ベンダ殿からの要請	○
安全性	閲覧室に人が案内されると、その人以外は、一定時間、閲覧室の金庫ボックスの認証に成功しない。(最終的に保証すべき安全性)	S01	閲覧室に案内されていない人が、(タッチパネル脇)ID端末で認証を行っても、常に認証をパスしない	簡略版モデルでは、時間を取り扱っていない(前提1-1)	◎
	閲覧室が空室の時以外、閲覧室に人を案内しない。	S02	閲覧室が空室のとき以外、決して案内は変更されない	−	◎
	利用者は、どの部屋にも閉じ込められない。(常に、退出することが出来る。)			簡略版モデルでは閲覧室の扉を取り扱っていない(前提1-3)	×
		S03	待機者リストはオーバーフローしない	コードデバック中にindexエラーを見つけたため	◎

参　考　文　献

1) JohnC, K.: Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Volume I: Planning and Technology Insertion, Technical ReportNASA/TP-98-208193, NASA, JPL (1998).

2) JohnC, K.: Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems, Volume II: A Practitioner's Companion, Technical Report NASA-GB-001-97, NASA, JPL (1998).
3) Bowen, J. P. and Hinchey, M. G.: Ten commandments of formal methods ...ten years later, *IEEE Computer*, Vol.39, p.2006.
4) Leveson, N.G.: *Safeware: System Safety and Computers*, ACM, New York, NY, USA (1995).
5) Ishiguro, M., Tanaka, K., Nakajima, S., Umemura, A. and Kishi, T.: A Guidance and Methodology for Employing Model-Checking in Software Development, APESER: Asia-Pacific Embedded Systems Education and Research Conference (2009).
6) Press, I.: IEEE Std. 830-1998, Recommended Practice for Software Requirements Specifications, Piscataway (1998).
7) Bowen, J.P.: Ten Commandments Revisited: A Ten-Year Perspective on the Industrial Application of Formal Methods, *In Proc. 10 th Workshop on Formal Methods for Industrial Critical Systems (FMICS 2005*, ACM Press, pp.8–16 (2005).
8) Jackson, M.: The Meaning of Requirements, *Ann. Software Eng.*, Vol.3, pp.5–21 (1997).
9) Jackson, M. and Zave, P.: Deriving Specifications from Requirements: An Example, *ICSE*, pp.15–24 (1995).
10) Jones, C.: *Applied software measurement (2nd ed.): assuring productivity and quality*, McGraw-Hill, Inc., Hightstown, NJ, USA (1997).
11) Ishiguro, M., Tanaka, H. and Matsuura, K.: The Effect of Information Security Incidents on Corporate Values in the Japanese Stock Market, *The Workshop on the Economics of Securing the Information Infrastructure (WESII)* (2006).
12) Ishiguro, M. and Nakagawa, A.: Proof Assistance for Algebraic Specifications Based on Proof Obligations in CafeOBJ, *Cafe: An Industrial-Strength Algebraic Formal Method*, Elsevier Science Ltd, pp.79–96 (2000).
13) Craigen, D., Craigen, D., Canada, O., Canada, O., Gerhart, S., Gerhart, S., Ralston, T. and Brown, R.H.: An International Survey of Industrial Applications of Formal Methods: Volume 1 Purpose, Approach, Analysis, and Conclusions (1993).
14) Larsen, P.G.: A Survey of Industrial Applications of Formal Methods, International Symposium on Formal Methods (2008). Engineering College of Aarhus.
15) Cusmano, M.: Software Development Worldwide: The State of the Practice, *IEEE Software*, pp.2–8 (2003).

付　　録

A.1 「フォーマルメソッド 適用ガイダンス」の構成

「フォーマルメソッド 適用ガイダンス」の構成は図 7 の通りである。

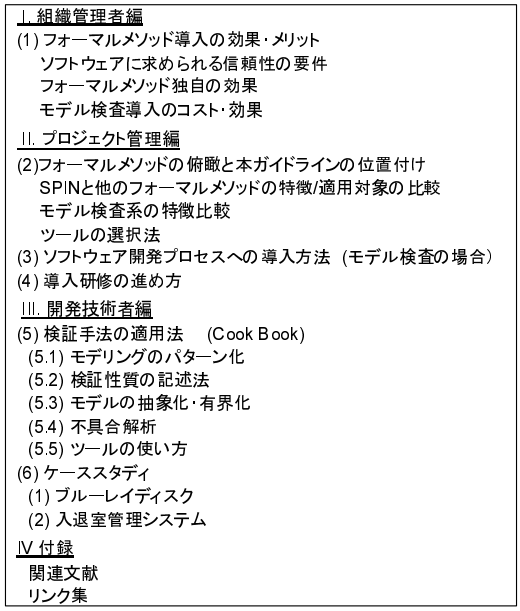


図 7　「フォーマルメソッド 適用ガイダンス」の構成

石黒 正揮

博士 (情報科学). 1992 年 東京大学理学部情報科学科卒業. 1994 年 東京大学大学院理学系研究科情報科学専攻修士課程修了. 同年株式会社三菱総合研究所入社. 1994 年より、形式仕様検証システムに関する研究, 米欧アジアの IT 政策調査、情報セキュリティ経済分析に関する研究, インターネット脅威分析システムの研究, 医用画像診断支援システムに関する研究などに従事する.

形式手法教育の課題
- HAZOP, UML, モデル検証

小　川　清　†

本論文は，形式手法を利用するための教育について，具体的な経験に基づいて知見を整理する。論理設計における，HAZOP のような安全分析，UML を使った設計，および SPIN, UPPAAL, VDM, Z, B method を使ったモデル検証について検討してきた。論理設計における設計指針が重要であることがわかった。

Education on formal method
- HAZOP, UML and model verification

Kiyoshi Ogawa †

This paper is a experience report of training and education on formal method for programmers. It is included safety analysis such as HAZOP, digital/logical design using UML and model checking like SPIN, UPPAAL, VDM, Z, and B method. Digital/logical design principles are studied.

1. はじめに

名古屋市工業研究所では，セーフウェア[1] の原書，IEC61508 など安全系のシステムに関する JAXA 様からの情報提供[2] に基づき，研究に取り組んできた。

経済産業省及び独立行政法人中小企業基盤整備機構の委託にて実施した戦略的基盤技術高度化支援事業 (通称サポイン事業) において，機能安全対応の自動車制御用ソフトウェアプラットフォームの開発を目的に研究開発事業（機能安全サポイン）を行っていたプロジェクトにおいて教育を実施した。安全なシステムを設計するために，ソフトウェア技術者はどのような技能または指針があるとよいかが課題となった。安全なシステムを設計するには，HAZOP をはじめとする安全分析，UML をはじめとするモデルの記述と，模擬試験，モデル検証などの人工言語の教育・訓練を行った。これらの技能の評価と，教育・訓練についての現在の考察を示す[3)4)]。

第２章で，技術者教育の実施状況を示す。第３章で，言語の分類を考慮し，システム設計者がどのような人工言語に関する技術を習得するかについて示す。第４章で，安全分析について HAZOP を例に，UML の図の利用と試験事例の作成を示す。第５章で，システム設計に必要となる設計指針について示す。第６章で，教育で得た知見を示す。最後にまとめと今後の課題を記述する。

2. 技術者教育

コンピュータシステムは，自然現象，社会現象を観察，制御，伝達することが目的である。自然現象，社会現象を観測，制御，伝達するためのシステムの設計には，言語を用いる。

名古屋市工業研究所は，中小企業の技術者向けに，技術研修，技術相談，試験，共同研究を行う公的試験研究機関である。技術研修の受講者には,「自分以外がお客様」であることを説明している。受講者の目的には，先行技術の習得を目的とする場合と，製品開発のための具体的な技能習得の場合がある[5)]。

技術者は，自分に技術があれば，自分以外はお客様であることを容易に理解できる。技術習得時から「お客様」にならないことは，一つの教育方法である。公的機関であるため民業圧迫にならないように，継続的同じ教育をするのではなく，毎回新しい試みを実施している。受講する技術者いは，社内外での講師になれることを目標設定していただくようお願いしている。

† 名古屋市工業研究所
NAGOYA Municipal Industrial Research Institute

☆ 著者

コンピュータシステムに関連する技術者教育で４日以上の日程の場合は，課題設定をお願いしている。課題を設定または想定し，その解決を技術研修の目標とする。

2.1 設計環境

コンピュータによる設計環境の基礎は OS である。設計者は，一つのコンピュータによる設計環境になじんでいるが，複数の設計環境が使いこなせる人と，一つの開発環境に長く，複数の開発環境の利用が困難な場合がある。例えば，二つの設計環境の使用頻度が同じ程度であれば，二つの開発環境を使いこなせるようになるのは容易であるが，利用頻度に一桁以上の違いがあると，容易には習得できない。あまり利用頻度の少ない設計環境の利用に慣れるように努力すると，普段利用している設計環境での操作ミスが増えるかもしれない。

文字編集，図形編集のエディタは，単一の操作でないと，効率が悪いことはよく知られている。そのため，いくつかの設計環境では，エディタを使い慣れたものに置きなおす工夫がされている。

2.2 導　　入

新たな設計環境への試用が，導入の不具合への対応で止まっている場合がある。初めて実習を企画するときは，設計環境の導入を実習に入れ，導入時にどのようなエラー，初期操作時にどのような不具合が発生するか，それをどう解決するかを課題の一つに設定することがある。

設計環境の導入で，導入時のエラー，初期操作時のエラーを収集し確認することにより，設計環境への対応を体系化し，エラーが生じた場合に原因が設計環境にあるのか，自分が書いた記述にあるのかを切り分けられるようにする。２回目以降は，過去の知見を資料としてつけ，導入作業は行わない場合がある。

SPIN のセミナなどで利用した「４日で学ぶモデル検証」[6] は，それだけで動作する KNOPPIX の CD を同梱していたことが利点である。HDD にソフトウェアを入れなくても，操作できるのは，道具の試験の視点ではよい。Windows から XSPIN を起動する設定を作成して受講した技術者もいた。

商用の設計環境では，ネットワークライセンスや，ライブラリなどで，他のソフトウェアとの共存がやっかいなものもある。

2.3 進捗確認

数日間の実習の場合には，初日の午前に，自己紹介を兼ねた課題の説明を実施している。また，実際の設計環境で動作確認も初日の午前にしている。初日の食事までの間に，やる気と緊張感を増幅させることが，全体の進捗に影響を与えるからである。設計環境の不具合があると，昼食時に対策を立てることができる。

実際に動作させることによって理論が理解できるようになる場合もある。教育の進捗の確認は，日報として，その日に分かったこと，分からなかったこと，よかったこと，明日以降に実施することの４点を報告する。

課題解決型の教育では，課題を解決するための技術を講師陣が理解していない場合もある。ネットワークを利用した調査，関係者への問合せを実施している。そのため，期間内に終了しないこともある。そういう場合には，解決の方向性を確認し，システム設計者がするとよいことを整理するようにしている。

システム設計者は，システムの目的と制約条件を明確にしながら，モデルを記述し，模擬試験または仕様の記述を検証するために人工言語を利用する。どの言語，との設計環境を利用するとよいかは，課題による。フリーのものはメールで導入報告をし，ダウンロードして利用するようにしている。

3. 言　　語

言語を，自然言語と人工言語に分類する。コンピュータシステムで利用する人工言語には，プログラミング言語，HDL，図記号，仕様記述などがある。

仕様記述などの形式手法も，人工言語の一種として，UML，HDL，プログラミング言語と同様の教育方法を行っている。また，道具を使うだけでなく，道具を作る技術者を視野に入れている。

3.1 言語教育

言語教育には，母語方式と文法方式とがある[7]。母語方式とは，母親から繰り返し同じ単語を聞くことによって覚えるように，反復学習方式である。文法方式とは，言語の文法を覚えることによって，理論的に言語を覚えるようにする方式である。言語の種類によって，どちらの方式がよいかが異なる可能性があるほか，学習者の性格，態度，期間などによっても，どちらの方式がよいかが異なる場合がある。そのため，言語学習では，どちらの方式でも選択可能にしてあるとよい。ある段階までは母語方式で，途中から文法を選択してもよいし，最初文法から初めて，途中から反復学習になってもよい。

3.2 自然言語

自然現象，社会現象には，自然言語で記述している情報が大量にある。これらの自然言語を組み立てて，システムを設計することを，自然言語による設計とす

る。自然言語による設計においては，システムにはどういう入力があり，システムがどういう状態を取り得て，出力がどうなっているとよいかを記述するものである。また，システムにはどういう入力があった場合は，どういう出力があるとよいか，どういう状態を取り得るかなどを検討する。条約，法令は，システムに対して制約条件である。制約は，顧客の要求事項と区分するとよい。

システム設計の際に利用する特許情報，学術情報は，膨大であるためデータベースになっている。これらのデータベースでは，索引を階層構造的に作成し，用語の包含関係を木構造の類語辞書 (thesaurus) として定義している。安全工学においても，木構造の分析が重要である[8)]。

名古屋市工業研究所では，自然言語に関しては，シソーラスの利用について，企業の知財部門を対象にした研修を実施してきた。文書作成は，LaTeX で記述した書籍[9)] をもとに，自然言語を構造的に書く LaTeX の利用を，印刷業界を対象に実施してきた[10)]。また，自然言語の機械翻訳にあたって，英語の構文の分析を行い[11)]，自然言語間の変換についても検討してきた。英文は，単語表現であるため機械処理が容易で，仕様を英文化することによって，明確にすることも試みている。

自然言語の課題は，利害関係者の視点によって意味が異なったり，概念の包含関係が異なることである。システムの設計において，自然言語の用語の木を作成する[12)] ことは，人工言語での集合関係の矛盾を少なくする上で重要である。

3.2.1 工業標準

国際規格などの工業標準は，特許，論文とともに，自然言語の文書では，形式が定まった技術文書である。工業標準は，製品の仕様，試験方法の使用を記述している。

新しい技術を工業標準にして普及する場合もあるため，目的と制約を認識して利用するとよい。[13)] ただし，国際規格間の間に矛盾あるかどうかを，自動的に検証する仕組みは提供されていない。

ISO9000 をはじめとする国際規格を，設計の際に参照関係を調査し，網羅的に対応を見ていくと役立つこともある。例えば，人の作業についての規定は，人の能力，やる気に応じていないと効果がないことが分かってくる。

3.3 人工言語

自然言語と人工言語の違いが明確に存在している訳ではなく，人工言語の一部は，自然言語の中に取り込まれことがある。自然言語自体も，遺跡に存在する壁画などから，人工言語から発展してきたものであると推測することもできる。ヒエログリフや，甲骨文字のように，自然言語の起源の図記号が，人工言語であるとすれば，自然言語と呼ばれているものも，人工言語の一部であると考えることができるかもしれない。自然言語にしても，人工言語にしても，図表記のものと，文字表記のものとがある。

そのため，図表記も文字表記も，人工言語の一種として扱う。ただし，図表記と文字表記とが，１対１対応できる関係を前提とする。

図と言語の変換は，UPPAAL のように，状態遷移図と XML とが相互変換可能な道具があるように，言語に変換可能な図のみを用いることもある。

- 自然言語
- 人工言語
  - 図 (例：UML)
  - 仕様・モデル記述言語
    - 静的仕様記述言語 (例：XML)
    - 動的仕様記述言語
      - 振る舞い記述言語（例：VHDL）
      - プログラミング言語

C 言語教育などでは，C コンパイラのソースコードを読むなど，複数の人工言語を理解する上で，言語処理系などを作ることが理解の一つの方法であることを確認している。プログラミング言語であるか，HDL であるか，制約・仕様記述の形式手法であるかの違いよりも，開発環境の操作性の違いと,「, ; == :=」などの記号，書式の違いの違和感が大きい場合がある。

3.4 プログラミング言語

自動車などの制御プログラムでは，アセンブラでの開発が行われてきたが，近年では C 言語による開発が主流である。名古屋市工業研究所の C 言語教育においては，C 言語の課題を明確に理解してもらうために The C Puzzle Book[14)] を利用してきた。The C Puzzle Book では，CPU によって値が違ったり，処理系によって振る舞いが違う処理が記述されている。そのため，どれだけ C 言語のことを理解していても，基本的なプログラミングの原則[15)] に従っていても CPU と処理系を理解していないと，振る舞いは確定しない。

小さな CPU の情報は開示されている[16)] が，大きな CPU の状態遷移や電気的な変動に対する振る舞いが，公開されていなかったり，一意に決まっていない場合もある。メモリの取得が成功したり，キャッシュがヒットするのは確率論的である。そのため，ソフトウェアの振る舞いは，決定論的には表現できず，確率

論的にしか取り扱うことはできない。CPU の状態遷移がすべて公開されていても，並列処理で，どの結果を最終的に利用するかが分からない処理もある[17)18)]。

MISRA-C[19)20)] では，メモリの取得の処理は行わない静的な記述であり，空間的な制約を明確にしている。MISRA-C に基づいたプログラミングでも，複数の割り込みが同時に発生した場合の振る舞いなどは，決定できないこともある。そのため，CPU の振る舞いを確率論的に理解していないと，ソフトウェアの振る舞いを網羅的に補足することは難しい。また，同じ系列の CPU であっても，入出力の周辺回路によって，構成管理を徹底する必要がある[21)]。

3.5 HDL

CPU をはじめとする論理回路では，HDL(hardware description language) による記述が可能である。HDL でも，あいまいな記述が可能であるため，RTL 設計スタイルガイド[22)23)] が，HDL 言語記述の課題を明確にしている。RTL 設計スタイルガイドは，MISRA-C に比べると，時間的な振る舞いに対する制約を明確にしている[24)]。RTL 設計スタイルガイドに沿った設計をしても，信号の揺らぎ（jitter）によって，論理回路を決定論的に設計することは難しいかもしれない。

3.6 UML

論理回路のような状態機械を表現する方法に状態遷移図がある。状態遷移図 (state chart) では，時間的な経緯を表現し切れないので，時系列図 (sequence chart) で表現することがある。また，より細かい時刻ごとの表現として時刻図（timing chart) によって表現することがある。これらの状態遷移図，時系列図，時刻図は，UML として標準化されている。

名古屋市工業研究所ではそのため，UML をモデル記述の道具[25)] として利用する教育を行ってきた。UML 教育では，状態遷移図，時系列図，時刻図以外にも，利用事例 (use case) 図のように，利用者側視点での図も作成するようにしている。

3.6.1 状態遷移図

電子計算機は，状態機械である。離散（デジタル）処理では，０という状態と１という状態の間を行き来する。１を真，０を偽とすると，論理計算である。実在する CPU，ソフトウェアの状態遷移のすべてを，後から図にしたり，検証することは困難な場合がある。そのため，主たる機能を状態遷移に記述したり，不具合のあった部位を状態遷移で記述するようにしている。

状態遷移の網羅性を確認するには，表形式が有効であり，UML でも振る舞い図の表形式について言及している。

3.6.2 時 系 列 図

時系列図は，通信プロトコルでよく用いる。状態遷移を時間に沿って記述することにより，どのような試験事例 (test case) を作るとよいかが明確になる。また，利用者の視点の利用事例図との両立性を確かめることにより，利用者視点の操作に耐えうることが確認できる。

3.6.3 刻 時 図

電子回路，論理回路，コンピュータの割り込みなど，順番だけでなく，時刻が関係する場合に適している。

3.6.4 利用事例図

利用事例図は，一つの利用の台本 (senario) を示すもので，一連の処理を確認するのに有効である。

3.7 制約・仕様記述

制約には，条約や法律で規定されている社会的な事項と，回路の振る舞いのように物理的な事項とがある。制約事項と特定の顧客の要求とが明確に分離されていないと，システムの設計，試験事例の作成が不十分になることがある。

OS の仕様など，API の振る舞いを確かめることも重要である[26)27)] が，割り込み，例外などの CPU の仕様を記述することが大事である。プログラミング言語における MISA-C，HDL における RTL 設計スタイルガイドのような空間と時間に関する設計指針があるとよい。

3.7.1 SPIN

産業総合研究所が作成した教科書[6)] で，４日の自習をした。システムを操作していくことによって，内容を理解していくことができた。そこで，SPIN を利用した教育を数度実施してきた。記述事例としては，通信プロトコル[28)]，論理回路[29)] を用いてきた。改訂した教科書では[30)] は，CD を同梱していない。

状態遷移の複雑なシステムでは，自動生成によって不具合の検出に利用している事例もある[31)]。

3.7.2 UPPAAL

デジタル回路，プログラムなどの論理設計においては，状態遷移が基本である。その基本を記述して，模擬試験することができる。UPPAAL の利点は，状態遷移図を XML 出力することと，時間指定ができることである。UPPAAL の課題は，大きな状態に対応できないため，他のシステムで階層化した部分との情報交換を容易にすることによって，部分の設計の検証に利用することである。

3.7.3 Z

Z は，国際規格になっている。OS の仕様の記述について試みている。ALLOY などの検証系も調査中で

表 1 Basic guide words and their generic meanings

Guide word	Meaning
no or not	complete negation of the design intent
MORE	quantitative increase
LESS	quantitative decrease
as well as	qualitative modification/increase
PART OF	qualitative modification/decrease
REVERSE	logical opposite of the design intent
other than	complete substitution

ある。

3.7.4 VDM

VDM は，国際規格になっている。仕様の記述を実施したほか，使用事例についても調査した[32]。

3.7.5 B method

SPIN, UPPAAL, Z, VDM, で記述したものを B でも記述することによって，各手法の利点，制約を明確にしようとしてきた[33]。

4. 安全分析

安全分析は，安全設計の基本的な技術である[1]。安全工学で紹介されている多くの分析手法の中で，HAZOP[34]，FTA[35]，FMEA[36] は IEC の国際規格になっている。FTA と FMEA は，ISO TS16949 で参照されていることと，日本においても国際規格になる前から広く品質，信頼性の確保に利用されてきた。FTA がシステム全体の故障から分析を始めるのに対して，FMEA は，部品の故障モードから分析を開始することができる。HAZOP は，システム全体の抽象的な概念での分析もできれば，部品の入出力に着目した分析もできる手法である。

4.1 HAZOP

HAZOP は，Hazard Analysis and Operability Study の略で,「潜在的危険・運用分析」と呼ばれている。HAZOP は，日本語では潜在的危険・操作性検討の原則及び手順ということができる。

HAZOP は，システムにおける運用時の物の流れに注目し，システムがどのような影響を受けるかを誘導語 (guide word) を使って分析する。誘導語は二種類あり，基本的なものと，時間に関するものがある。時間には，順序に関する誘導語と，時刻に関する誘導語に分かれていることにより，試験・検証仕様を明確にできる利点がある。

当初，化学プラントのような装置産業において，管を流れる液状を，弁で制御するシステムで用いられてきた。物の流れを，電子の流れや，情報の流れに置換えると，電子回路，通信，論理回路・ソフトウェアの分析に用いることができる。

表 2 Guide words relating to clock time and order or sequence

Guide word	Meaning
EARLY	Relative to the clock time
LATE	Relative to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

HAZOP で，ありとあらゆる物の流れに着目して分析しようとすると，どこまで分析をしないといけないか分からなくなる。これは，検証をどこまでやればよいかと対応する。ほとんど起こりそうにないことを検討していると，作業が永久に終わらない可能性がある。そのため，事前に，何に着目するか，どれくらいの時間をかけるか，自動検証するとよいことが何があるかを設定するとよい。

分析手法が生まれたプロセス産業では，設計者が操作することはない。設計者と操作者は，別の人である。しかし，自動車やコンピュータは，設計者が対象を操作することもある。そのため，設計者が利用者としての分析も可能になる。ただし，利用者の参加が分析に必要だという要請は大切である。

複雑なシステムでは相互作用があるため HAZOP でが有効でないとの指摘がある[37]。相互作用を，論理回路またはソフトウェアで監視または制御している場合には，そのシステムの入出力で HAZOP を実施することにより，HAZOP の対象とすることができる。HAZOP には原因究明の仕組みはないので，なぜなぜ分析などでの知見を生かすとよい[38][39]。

分析は，試験事例の作成が到達点で，実施可能な時間での試験事例が目標である。誘導語の適用にあたっては，状態遷移図，時系列図，刻時図などを書いて，利用者視点での指摘をしやすいようにする。分析は，概念段階，設計段階，試作段階など，多段階で行うとよく，その事点での試験事例の作成を目指すとよい。

HAZOP は，利用者の分析への参加を要請していることと，具体的な分析の手がかりとして誘導語を設定しているため FMEA，FTA で分析し切れていないことを発見することができる。

4.2 技能標準

安全診断には，高度な専門性が必要であり，技能標準が提案されている[40]。安全関連系の設計においては，技能に応じた作業が必要である。組込み系では ETSS が能力に応じた作業をするための指針を目指している[41]。

形式手法に関しては，論理演算が理解できて，真理表が作成できれば，母語方式の教育が可能であることを前提として，ETSS の枠組みを利用して技能標準を，北海道立工業試験場の指導のもと作成してきた。証明する式の作成，モデルの設計方法は，対象とするシステムの制約条件と，設計指針によって求めるとよい[42)43)] ことが分かった。

5. システム設計

システムの設計は，入力，状態，出力を明確にし，想定範囲内の正常入力に対して，特定の時間以内に特定の出力が出るようにすることである。また，想定範囲外の異常入力に対しては，システムの安全性を損なわないような状態，出力になるように組み上げることである。

土木工学，建築学などにおいても，設計は，部品の性質が明確であることが設計工学の基本となっている。設計工学の成立条件は，次の事項が考えられる。

- 要素技術 (部品) は確立しているか
- 設計･試作の道具は完備しているか，自作できるか
- 模擬試験・検証の道具は完備しているか，自作できるか

コンピュータシステムの最大の部品は CPU である。しかし，CPU は，一部の仕様が公開されていない場合がある。また，CPU の試験結果が公開されていない場合もある。CPU の検証結果なしに，その上で動くソフトウェアの検証を完全と言うことは困難である。また，１ビットごとの状態を考えると，32 ビット CPU のレジスタ１つで，4294967296 状態の遷移になる。対象の分析によって重要な状態遷移を少なくする設計にすることが課題となる。

システムの設計の要素として，機械設計，電気設計と，論理回路の設計とソフトウェアの設計を考慮する。設計は，ある意味モデルを記述することである。モックアップのように動かない静的モデルであっても，風洞試験という動的試験も可能である。モデルを記述するためには，設計指針が明確であるとよい。

5.1 設 計 指 針

設計指針には，単純化などが例示されている。相似，対称などは，単純化の切り口である。部品は，相似，対称など，単純化したものについて成立する。単純化には，等価，相似，対称性[44)] などが有効な指針である。

ソフトウェアはシステムの中で機能する頭脳部分だということができる。電子計算機は，電脳とも表記されることがあるように，頭脳である。ソフトウェアは，電脳のために記述した算譜である。

システムの動きを理解しないと，ソフトウェアの動きだけを理解しても，現実との境界が明確にならない。

5.2 部 品 化

設計方法として，部品を組み立てて設計する場合と，手書きのように任意の物質をから作成する場合とがある。設計工学では，部品を前提にした理論が基本となっている。

ソフトウェア分野でも，様々な部品化が進行しており，近年では，TOPPERS コンポーネント WG が，組込みシステムの部品化の設計指針を検討している[45)]。ソフトウェア部品に対する分析[46)] 例もある。

5.3 検証・模擬試験 (シミュレーション)

モデルの検証は数値的な模擬試験により一つの結果に到達することを示す模擬試験を行うことと，論理式の反例を探すことによって，論理的な検証をする方法とがある。

模擬試験は，実際の利用手順を想定して，その際，システムがどのように振舞うかを確認する方法である。逐次実行処理だけで構成したシステムでは，ある入力の系列または利用手順に対して，１通りの振る舞いの確認でよいが，ある入力の系列に対して，複数の可能な状態の遷移がある場合には，すべての状態遷移を辿る試験を実施するかどうかはシステムの達成したい目的による。

CAD/CAE という分野では，Matlab/Simlink のようにモデル設計から模擬試験まで一連の設計環境を揃えるようになっている。

6. 教育で得た知見

6.1 設計戦略と設計指標

モデルの設計のための指標が明確でないと，どういうモデルを記述するとよいか，どこまで詳細に書くとよいかが明確にならない。実際に記述しながら，どこまで必要になるかを考えることも有用である。Safeware[1)] における安全設計の考え方や置換，単純などの設計指針，ユニバーサルデザイン，かたちの設計規範など既存の設計規範をうまく利用していくのがよい。設計においては，全体の抽象的な概念から記述していく方法と，OS やネットワークプロトコルのように，部品から記述していく方法とがある。両方の方法を取ることにより，モデルに必要な部品かどうかの判定ができる。

HAZOP で，状態遷移図，時系列図などを用いることと，時刻に関する誘導語が有用であることがわかった。

6.2 分析と試験・検証

モデル作成は，分析，設計，試験・検証と一環して実施するものである。モデルで何を検証するかは，分析作業において，具体的に議論するとよいことが分かった。分析作業は，試験・検証するのに十分な知見が得ることを目標にするとよいことがわかった。HAZOP における順序と時刻を分けて分析する手法が，モデル設計，試験・検証の事例作成に有用である。また，論理回路に関する試験可能性[47] などを確実にする必要性を感じた。

7. ま と め

本論文では，システムの設計の道具として UML, HDL，プログラミング言語，仕様記述言語の教育を人工言語教育の一貫に位置づけた。,

仕様記述をどの程度，どうやって記述するかは，対象システムの特性と，設計指針による。対象システムの特性を，制約条件として記述する事項と，満足するとよい要求事項と，意図した利用について整理するとよい。図で記述することにより，入出力の関係に集中することができたり，時間的な経過に集中することができる。表で記述することにより，網羅性に集中することができる。言語で記述することにより，論理展開に集中することができる。従来の道具が，図，表，言語に対して部分的であったので，一環した（One Stop Service）道具作りが今後の課題である。

謝辞 松原友夫様，JAXA の片平真史様に Safeware についてご教示いただいた。名古屋大学結縁祥治先生，産業総合研究所の水口大地様，北海道立工業試験場の堀武司様に，論文の背景となる教育にあたって，ご指導いただいた。アイシン AW 株式会社の久保孝行様，株式会社東海理化の足立秀和様，株式会社サニー技研の米田真之様，中本加那様，東海ソフト株式会社の岡村真吾様，服部智幸様，古田英美様，株式会社ヴィッツの鵜飼敬幸様に形式手法に関するシステムの調査，記述，検討に御協力いただいた。名古屋大学の高田広章先生，株式会社ヴィッツの服部博行様，アイシン精機株式会社の鈴村延保様，トヨタ自動車株式会社様，株式会社ルネサステクノロジー様ならびに機能安全プロジェクトに開発者，アドバイザ，オブザーバとして参加いただいた皆様にご協力いただいた。(株) 日本システム安全研究所 の吉岡律夫様に機能安全プロジェクトでご指導いただいた。名古屋市工業研究所でのセミナは，斉藤直希，渡部謹二との分担で行った。富士ゼロックス株式会社の服部彰宏様，山本訓稔様，フェリカネットワークス株式会社の栗田太郎様，株式会社キャッツの穴田啓樹様から情報提供いただいた。NPO 法人トップエスイー教育センター，国立情報学研究所 GRACE センターでチュートリアルを受講させていただいた。中部アイティ協同組合様，中部エレクトロニクス振興会様に名古屋市工業研究所での UML, spin, uppaal のセミナ開催にご協力いただいた。組込み中核人材プロジェクト[48)49)] において，株式会社名古屋ソフトウェアセンター様，国立大学法人名古屋工業大学様，国立大学法人岐阜大学様に関連する講座を開講していただいた。皆様に謹んで感謝の意を表します.

参 考 文 献

1) ナンシー・G・レブソン:セーフウェア 安全・安心なシステムとソフトウェアを目指して, 翔泳社, (2009)
2) プロセス改善ナビゲーションガイド　ベストプラクティス編, IPA/SEC, (2008)
3) 小川 清, 斉藤直希, 渡部謹二, 森川聡久, 服部博行:デジタル設計工学の基礎の検討, ディペンダブルシンポジウム,(2009)
4) 堀武司, 小川 清, 斉藤直希, 渡部謹二, 森川聡久, 服部博行：安全分析, 状態記述と形式手法に着目した安全教育とスキル, 安全工学シンポジウム,(2009)
5) 産業技術連携推進会議情報・電子部会組込み技術研究会：組込みシステム開発事例集, 工業調査会,(2006)
6) 産業技術総合研究所システム検証研究センター: 4 日で学ぶモデル検査 (初級編) CVS 教程 (1), エヌ・ティー・エス, (2006)
7) 小川清, プログラミング言語教育の自然言語教育方式の応用, 名古屋市工業研究所研究報告, No84, pp19-23, (1999)
8) 鈴木喜久：安全性に関する樹木図解析, 東京都立科学技術大学研究報告, (1989)
9) 小川清, 安藤彰敏：データベース検索技術者試験傾向と対策, 紀伊國屋書店, (1993)
10) 奥村晴彦：LaTeX2 ε 美文書作成入門 , 技術評論社, (2006)
11) 加藤輝政，小川清，佐良木昌：英語複文の構文解析と編集，その論理と方法, 言語理解とコミュニケーション研究会 , 電子情報通信学会,(1997)
12) 小川清, 斉藤直希：リアルタイム組込みソフトウェアの用語の木, 実時間ワークショップ, 電子情報通信学会,(2003)
13) 水野智仁, 森川聡久, 小川 清, 斉藤直希, 渡部謹二, 堀 武司：安全に貢献するソフトウェア関連国際規格, 安全工学シンポジウム,(2009)
14) Alan R. Feuer: The C Puzzle Book, アスキー出版局, (1985)
15) スティーブ マコネル: Code Complete 第 2 版, 日経 BP ソフトプレス, (2005)
16) 渡波郁: CPU の創り方, 毎日コミュニケーショ

ンズ, (2003)
17) ヘネシー, パターソン:コンピュータアーキテクチャ-定量的アプローチ第 4 版, 翔泳社 (2008)
18) パターソン, ヘネシー:コンピュータの構成と設計-ハードウウェアとソフトウェアのインタフェース第 3 版, 翔泳社, (2006)
19) MISRA-C 研究会: 組込み開発者におくるMISRA-C―組込みプログラミングの高信頼化ガイド, 日本規格協会, (2004)
20) MISRA-C 研究会: 組込み開発者におくるMISRA‐C:2004―C 言語利用の高信頼化ガイド, 日本規格協会, (2006)
21) 斉藤直希, 渡部謹二, 小川清, 大澤史郁, 大槻直哉, 片岡歩, 服部博行: 類似の CPU への組込み OS の移植のための訓練内容と構成管理表, 高速信号処理応用技術学会誌, 第 10 巻 第 1 号. (2007)
22) STARC: RTL 設計スタイルガイド VHDL 編,STARC, (2003)
23) STARC: RTL 設計スタイルガイド Verilog-HDL 編 第 2 版,STARC,(2006)
24) 長谷川誠, 渡会勝彦, 米永裕司, 渡部謹二, 小川清: Verilog HDL スタイルガイドの利用方法, 情報処理学会組み込み研究会, (2007)
25) 児玉公信: UML モデリング入門 本質をとらえるシステム思考とモデリング心理学, 日経 BP, (2008)
26) 斎藤靖, 高田広章, 坂村健: 形式システム仕様記述を用いた組み込み用リアルタイム OS ITRON の記述とテスト,ITRON 技術研究会 Vol.4, No.3, pp63-74 , 1992
27) 斉藤直希, 堀武司, 小川 清: 安全関連システムのための OS の検討, 安全工学シンポジウム,(2009)
28) Chris J. Myers: 非同期式回路の設計, 共立出版, (2003)
29) Gerard J. Holzmann: コンピュータプロトコルの設計法, カットシステム,(1994)
30) 産業技術総合研究所システム検証研究センター: モデル検査 初級編―基礎から実践まで 4 日で学べる, ナノオプトメディア,(2009)
31) 村石理恵, 服部彰宏, 野村秀樹, 山本訓稔：Model Checking を適用した実践的非同期制御検証, ソフトウェアテストシンポジウム東京 ,(2007)
32) 栗田太郎, 荒木啓二郎：モデル規範型形式手法 VDM と仕様記述言語 VDM++：高信頼性システムの開発に向けて (情報システムの信頼性・安全性), 日本信頼性学会誌 , 31(6) pp.394-403,(2009)
33) 堀武司, 岡村真吾, 服部智幸, 斉藤直希, 小川清: 機能安全対応組込みソフトウェア開発における B メソッド導入の試み, システム検証ワークショップ, (2008)
34) IEC 61882 Hazard and operability studies (HAZOP studies) - Application guide , (2001)
35) IEC 61025 Fault tree analysis (FTA), IEC, (2006)
36) IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), IEC, (2006)
37) Trevor A. Kletz: Hazop and Hazan 4th, The Institution of Chemical Engineers, (2006)
38) 佐藤洋介:「実践！なぜなぜ分析」 ,SWEST9, (2007)
39) 間瀬順一:「実践！なぜなぜ分析 パート 2」, SWEST10, (2008)
40) IEE: Safety, Competency & Commitment,Competency guidelines for safety-related system practitioners, IEE, (1999)
41) 組込みソフトウェア管理者技術者育成研究会:組込みソフトウェア開発のための ETSS 標準ガイドブック―組込みスキル標準に基づく技術体系と詳細スキル, 日経 BP 社, (2006)
42) 小川清, 斉藤直希, 堀武司, 水口大知, 吉岡律夫, 森川聡久, 服部博行:ソフトウェア開発における安全分析, 形式手法, 工業標準に焦点をあてた安全関連スキルと教育訓練, システム検証ワークショップ, (2008)
43) 小川清, 渡部謹二, 斉藤直希, 堀武司, 奥田篤, 水口大知, 吉岡律夫, 渡辺登: ETSS を利用した機能安全対応スキル判定と教育訓練, 安全工学シンポジウム,(2008)
44) 小川清, 二木厚吉: OBJ の試用経験,SEA ソフトウェアシンポジウム,(1992)
45) 安積卓也, 山本将也, 小南 靖雄，高木信尚, 鵜飼敬幸，大山博司, 高田広章: 組込みシステムに適したコンポーネントシステムの実現と評価，コンピュータソフトウェア, Vol.26,No.4, pp. 39-55, Nov (2009).
46) 斉藤直希, 小川清, 水口大知, 菊池達也, 大西秀一, 長谷部浩二, 堀武司: コンポーネントソフトウェアに対するハザード分析手法の検討, システム検証ワークショップ, (2008)
47) Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, Digital systems testing and testable design, Jaico Publishing House, 1993
48) 名古屋工業大学: CIET 中部組み込みソフトウェア技術者養成講座プロダクトマネージャ育成, (2008)
49) 岐阜大学: CIET 中部組み込みソフトウェア技術者養成講座ドメインスペシャリスト育成, (2008)

大学と企業での形式手法教育の経験と課題

劉少英
法政大学・情報科学部・情報科学研究科
Email: sliu@hosei.ac.jp

大学と企業での１５年間の形式手法ＶＤＭ（Vienna Development Method）及び形式工学手法ＳＯＦＬ（Structured Object-Oriented Formal Language）の教育経験によって、形式手法の役割、教育方法、および様々な教育についての課題を指摘する。

Experience and Challenges in Teaching Formal Methods at University and Industry

Shaoying Liu
Faculty of Computer and Information Sciences, Hosei University
Email: sliu@hosei.ac.jp

Several issues including the role of formal methods, teaching methods, and various challenges in teaching formal methods are discussed based on the author's experience of teaching VDM and SOFL over the last fifteen years.

1. はじめに

ソフトウェア工学と形式手法の分野での２８年間の研究経験に加え、大学と企業での１５年間の形式手法ＶＤＭ（Vienna Development Method）「1」及び形式工学手法ＳＯＦＬ（Structured Object-Oriented Formal Language）「2」の教育活動を行い、形式手法の役割、教育技術、および様々な教育についての課題を体験してきた。いくつかの大学で、形式仕様記述言語とその技術を学部三・四年生中心に教え、形式工学手法 (Formal Engineering Method)、形式詳細化手法および形式検証技術を大学院生に授業をしてきた。また、日本企業にもＶＤＭとＳＯＦＬを教えている。これらの経験を通じて、ソフトウェア開発人材を育成するには、形式手法の教育が大事だと考えてきた。

2. 教育経験

具体的には、形式手法教育における重要なポイントを、以下に指摘したい。

１．形式手法は、実際の開発プロジェクトに適用することよりも、ソフトウェアエンジニアを育てるための力が大きい。形式手法は、企業であまり普及していないのが現実であり、その理由が既に明らかになっている。但し、企業でよく使われているソフトウェア開発手法は、形式手法のように厳密性と科学性のどちらをも持っている訳ではない。つまり、形式手法は、「理想的」なソフトウェア開発プロセスを示し、プログラムの正確性を保証するために必要な条件とアプローチを提供している。理想的な開発プロセスを理解していないソフトウェアエンジニアは、高信頼ソフトウェア製品を作るのが難しい。このため、ソフトウェアエンジニアの教育には、形式手法が適切だと考えている。

２．形式手法の教育を成功させるには、教育方法が重要である。ソフトウェア工学におけるどのような問題を解決できるのか、どのように解決できるのか、などの課題を常に考慮した上で、形式手法の教育を行なわないと、学生らに興味を持たせることは困難である。また、簡単な事例を使うことや、沢山の練習をさせることや、小さなプロジェクトで実践させる、などの教育技術も大事である。

３．形式手法は、全ての学生またはソフトウェアエンジニアが勉強できる訳ではなく、数学的な考え方またはプログラミングが強い学生またはエンジニアだけが、楽しんで勉強できる。

４．形式手法の教育には、支援ツールを使うことが有効ではない。むしろ、効果が悪い。但し、形式手法を本当に把握した上で、具体的なプロジェクトに適用する場合は、支援ツールを使うことが望ましい。理由としては、形式手法教育に一番求められていることは、概念を理解することである。自

分の手で必要な概念や仕様などを練習することが、勉強に一番有効な方法だと分かって来ている。

3. 課題

形式手法教育においては、次のような課題に直面している。

1．形式手法の教育は、人気がない。主な原因は、数学を好きな学生が少ない、数学的なプロセスが難しいと感じる学生が多いからである。そして、Computer Graphics やインタネットなどの科目より学んで楽しくないことである。
2．企業においては、同僚でも理解していない人が多い。同じような理由で、大学または企業の同僚からも、形式手法の有用性への理解を得ることが難しい。
3．企業は形式手法を良く勉強した学生を優先採用していないし、給料も普通の学生より高く設定していない。

4. 改善策

これらの課題を克服するために、以下の改善策を提案をする。

1．形式手法の授業を必ず必修科目として設定する。
2．形式手法の教育技術を改善する。
3．企業や標準機関などは、形式手法の重要性を強調し、形式手法を勉強した学生を就職時に優先採用する。
4．形式手法にこだわるコンサルタント会社を育ていく。そのような会社を通じて、形式手法の応用を企業で普及させる。

参考文献

1) C. B. Jones, ``Systematic Software Development using VDM'', 2nd edition, Prentice Hall International, 1990.

2) S. Liu, ``Formal Engineering for Industrial Software Development using the SOFL Method'', Springer-Verlag, 2004.

形式手法教育の取り組みについて

青　木　利　晃†

本稿では，これまでの形式手法教育に関する取り組み，および，それらから得られた知見について紹介する．

On Education of Formal Methods

Toshiaki Aoki†

In this paper, I introduce my education activities on education of formal methods, and their experineces.

1. はじめに

形式手法は古くから研究がされている分野であるが，日本の産業界においては，それほど注目される分野ではなかった．しかしながら，今日では，システムの信頼性低下が問題になっており，形式手法に対する期待が急激に高まっている．このような急激な期待の高まりは，情報不足も相まって，形式手法に関する様々な情報の氾濫が生じつつあった．そこで，正しい情報を提供するためには，実際にその一部に触れてもらうことが最善であると考え，セミナーなどによりその普及活動を行うようになった．また，共同研究のために，企業の内部に，同じ興味や問題意識を持つ人を増やすことも目的である．当初は，古典的なプログラム検証や定理証明システムに関するセミナーを，試行として開催していた．しかしながら，理解するためには様々な前提知識が必要となり，短時間で現実的な問題に適用できる知識を身につけるのは困難であった．そこで，比較的理解しやすく使いやすいモデル検査について，セミナーを行うようになった．

2. 形式手法教育内容

これまでに，大学での授業と産業界向けのセミナーを行っている．まず，これら２つの活動について紹介する．

2.1 大学における授業

北陸先端科学技術大学院大学では，従来からの博士前期課程や博士後期課程で勉学に専念する学生さんの他に，会社で働きながら学位を目指す，いわゆる，社会人学生も対象に教育を行っている．なお，社会人学生向けの授業は，東京のキャンパスで行っている．前者の学生さんを対象とした形式手法に関する授業には，基礎理論から最先端の研究に関するものまで幅広くある．後者の学生さんを対象とした授業には，基礎理論の他にモデル検査に関するもの，形式仕様に関するものがある．

2.2 産業界向けセミナー

試行として自主セミナーを数回行った後，日本科学技術連盟による有料セミナー，QUBE, TopSE などの人材養成プログラムに協力してきた．QUBE は 2 日間，日本科学技術連盟は 3 日間のセミナーである．日本科学技術連盟による有料セミナーに関しては，大学の社会貢献の一環として行っており，セミナー開催にかかる経費（資料代，日本科学技術連盟に関係する費用）以外は徴収していない（例えば，講師代）．これらのセミナーでは，主に，モデル検査ツール Spin の使い方についてセミナーを行っている．知識を効率的に身につけるためには，具体的なツールの使い方を覚えるのが良いと考えたからである．また，Spin を使っている理由は，シンタクスが C 言語などの手続き型言語のものに似ており，かつ，並行プロセスに関する様々な問題を簡潔に記述可能であったからである．

3. アンケートなどから

日本科学技術連盟で行ったセミナーに関しては，日本科学技術連盟が開催するセミナーすべてに共通のアンケートを実施している．その内容の一部を図 1 に紹

† 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

アンケート数	理解度	有効性	実践できるか
81	74 (91%)	76 (91%)	71 (88 %)

図 1 アンケート結果

介する．特に前提知識に関する条件を提示していないにも関わらず，理解度が高い．モデル検査ツールは，使うには難しく無いことがわかる．並行プロセスにまつわる問題は，比較的小規模な振る舞いでも発見するのが困難である．それらについて，モデル検査ツールで自動的に発見できる例を示せたことが，有効性や実践に関する評価も高くなった理由であろう．

このように期待が高い反面，理解はできたが，各人が直面している実際の問題にどのように適用すれば良いかわからない，という意見も多かった．モデル検査は単純な技術であるが，それゆえ，直面している問題を，どのようにして，その単純な技術に落とし込むかが難しいのである．このような応用法に関しては，一般論はなく，企業や各人においてノウハウとして蓄積していくしかないと考えている．このノウハウの習得を教育として実現するのは難しいが，他の関連技術を総合的に用いた問題解決に関する演習，すなわち，PBL(Problem Based Learning) などの教育手法が有効に働くかもしれない．

また，このセミナーを受講した後，数ヶ月も原因不明だった製品の不具合が，数日で発見できた，という報告もあった．どのように適用したかは不明であるが，いずれにしても嬉しい報告であった．モデル検査は万能な手法ではなく (そもそも万能な手法自体が存在しないが...)，それが有効に働くであろう対象と問題がある．もちろん，セミナーでは，そのような個々の製品の問題に関しては紹介できない．セミナーで紹介できるのは，典型的な問題と，それらのモデル検査ツールによる取扱いくらいである．これらは個々の製品の問題ではないが，受講者の応用力を働かせるきっかけにはなると考えている．

4. さいごに

形式手法教育と一言で表現しても，それについて議論すべき事は非常に多い．まず，形式手法には様々なものがあるため，教育する技術自体にバリエーションがある．さらに，教育対象者の背景，すなわち，研究者を目指す人なのか，企業人を目指す人なのか，さらには，すでに企業で問題に直面している人なのかによっても議論は異なるであろう．本稿で紹介した意見は，主に，モデル検査を企業ですでに問題に直面している人向けのセミナーに関してである．

最近，研究会やシンポジウムで形式手法に関するセッションに参加すると，十数年前とかなり状況が異なることに驚く．十数年前の同様のセッションでは，聴衆が少なく，そのセッションの発表関係者しか聴衆が居ないという状況は少なくはなかった．現在では，満席という状況も少なくは無い．技術への注目には波があると考えているが，できれば，現在の状況を継続させたい．最悪，波の底が来たとしても，十数年前よりは良い状況にしたい．そのためには，形式手法が有効に働いた事例を増やすことが重要であり，その鍵は形式手法ユーザーを増やすこと，すなわち，形式手法教育であろう．

形式手法教育への取り組み

来間　啓伸 †

産業応用の視点からすると，形式手法の研究開発は基本技術の整備の段階から，基本技術をソフトウェア開発でどのように有効に利用するかの実適用技術の開発の段階へと，移行しているようである．このような状況の下，形式手法教育を前提として，さらに産学が協同して問題解決にあたるコラボレーションが必要であると考える．

1.　はじめに

近年，信頼性の高いソフトウェアを開発するための手法として，形式手法が産業界からも関心を持たれている．形式手法の基礎は 1970 年代には固まっており，産業界から注目されるのも今回が初めてではない．過去の経験から，形式手法に懐疑的なあるいは否定的な意見を持つ技術者も，産業界には多い．

最近の動きを過去のそれと比較すると，ソフトウェアの信頼性に対する顧客の関心が高まりビジネス上の重要性が増したこと，支援ツールの性能が飛躍的に向上し仕様の検証が現実的になったこと，形式手法教育を行う機関が増え企業技術者が形式手法に触れる機会が増したこと，が相違点として挙げられる．

2.　形式手法の技術要素

ここでは，形式手法の基本要素として次を考える．

- 形式仕様記述言語
- 検証手法
- 作業過程

形式仕様記述言語は文法規則が（数学的基盤のもとに）厳密に定められた仕様記述言語であり，検証手法は仕様が整合していることを文法規則にしたがって（数学的に）確認するための方法を提供する．作業過程は，ソフトウェア開発過程における手法の位置付けを定め，形式仕様記述言語を使って仕様をどのように記述し検証するかについての手順を提供する．形式手法には様々な手法があり，これらの基本要素の内容も手法によって多様である．

形式手法の研究開発は現在も進められているが，実適用の側面からは，既に基本技術は十分整備されているように思われる．課題は，それらの基本技術をソフトウェア開発に有効に利用するための，実適用技術の開発に移っているようである．ここで，有効性は一般に開発現場によって異なり，利用方法もノウハウのような定式化し難い要素を含むことから，産業界と協同した技術開発が必要になる．

† 国立情報学研究所 National Institute of Informatics／日立製作所 Hitachi, Ltd.

3.　トップエスイープロジェクト

トップエスイープロジェクトは社会人向けのソフトウェア工学教育プロジェクトであり，形式手法に関してもモデル検査手法や VDM，B メソッドなどの講座がある．B メソッドの場合，基礎編と応用編に分けて合計 10 回の講義と 4 回の事例演習を行う．講義では仕様記述言語の文法や検証の基本概念を学ぶほか，支援ツールを使った実習を通じて作業過程を体験的に学ぶ．事例研究では，より複雑な例題をもとに，グループ討論を通じて受講生が各々の視点で議論する．さらに，修了制作では受講生自身が設定した問題について形式手法を適用し，その有効性を考える．

トップエスイーは教育プロジェクトであり，受講することによって一定のスキルを身につけることができる．しかし，例えば B メソッドについては入門講座であり，修了すれば直ちに B メソッドを開発現場に導入できるノウハウが身につくわけではない．前述のように，各開発現場の特性に合わせた問題の解決が必要であり，実用化のための技術開発が求められる．このような視点からすれば，教育は前提であるが，教育の次のステップとして，共に問題解決に向かうコラボレーションが必要であると考える．

4.　おわりに

形式手法の実適用のためには様々な課題を解決する必要があり，短期間で目覚しい効果を挙げることは難しい．一方，産業界での適用成功事例の報告も増加しており，有効に使うための方法を考える段階に来ていると思われる．まず教育があり，次の段階としてコラボレーションへと向かうべきと考える．

参考文献

1) T.-S. Hoang, H. Kuruma, D. Basin, J.-R. Abrial: Developing Topology Discovery in Event-B, Science of Compute Programming, 74, 11/12, pp.879-899, 2009.
2) 来間: B メソッドによる形式仕様記述，近代科学社，2007.

形式手法教育の取り組み：「新人」の視点から

石　川　冬　樹†

発表者は，社会人向けのソフトウェア工学教育コースであるトップエスイープログラム[?]において，2007 年より形式仕様記述，特に VDM に関する講義や修了制作指導を担当してきた[?]．また，同プログラムにおける形式手法教育全体に関する統計調査，海外での報告も行っている[?]．その他，モデル検査も含めた形式手法全般についても，入門的な講演・チュートリアル等を担当してきた．

本パネルセッションにおいては，発表者固有の視点として特に，以下の話題を提供する．

- トップエスイープログラムにおいては，実際に形式手法を学んだ開発者が，修了制作において「自分の課題」を設定し修了制作に取り組んでいる．修了制作の課題設定等に関する統計[?]等を紹介し，その示唆することについて議論したい．
- 発表者は,「経験浅い若手教育者」でもあり,「他分野を主たる専門としてきた（教育を行うために知識獲得・調査等の準備を行った）教育者」でもある．この視点から,「教育者の教育」，教育者の異なる役割等について議論したい．

On Educational Activities of Formal Methods: from Viewpoints of a Newcomer

Fuyuki Ishikawa[?]

Since 2007, the presenter has been working as a lecturer on formal specifications (especially VDM) and supervisor on graduation studies, in the Top SE program where developers from the industry learn software engineering approaches. Meanwhile, the presenter has been working on statistical surveys and reporting activities in international contexts, regarding the whole picture of formal methods education in Top SE. The presenter has also given some general, introductory talks or tutorials on formal methods.

In the panel session, the presenter specifically provides specific topics as follows.

- In the Top SE program, developers who have leant formal methods identify and taclke their own problems, in their graduation studies. The statitics such as the ones on the problem settings will be potential sources of interesting discussion.
- The presenter has been a young, inexperienced educator as well as an educator who has been working on different areas (and thus prepared for the education by obtaining knowledge and surveying). Such experiences will be potential sources of interesting discussion on education of educators, different roles of educators, and so on.

[1] 【TOPSE】国立情報学研究所 トップエスイー プロジェクト, `http://topse.jp/`.

[2] Ishikawa, F., Taguchi, K. and Honiden, S.: How Top-Level Engineers Learn and Investigate VDM: Experiences in the Top SE Project, *The 7th VDM-Overture Workshop at FM 2009* (2009).

[3] Ishikawa, F., Taguchi, K., Yoshioka, N. and Honiden, S.: What Top-Level Software Engineers Tackles after Learning Formal Methods - Experiences from the Top SE Project, *The 2nd International FME Conference on Teaching Formal Methods (TFM 2009)*, pp. 57–71 (2009).

† f-ishikawa@nii.ac.jp
国立情報学研究所
National Institute of Informatics