2015 年 3 月 1 日発行
第 10 巻 第 6 号（通巻 43 号）
ISSN 1349-8622

SEC journal 40

巻頭言

荻原 紀男　一般社団法人コンピュータソフトウェア協会（CSAJ）会長
株式会社豆蔵ホールディングス 代表取締役社長

所長対談

システムズエンジニアリングの最前線－技術・人・社会を含めて考えるシステム開発と運用－

ジェイムズ・マーティン　The Aerospace Corporation Principal Engineer

論文

UML による組込みソフトウェア設計の検証支援環境の開発

横川 智教　岡山県立大学 情報工学部／天嵜 聡介　岡山県立大学 情報工学部／佐藤 洋一郎　岡山県立大学 情報工学部
有本 和民　岡山県立大学 情報工学部／宮崎 仁　川崎医療福祉大学 医療技術学部

概念段階におけるハザード・脅威の識別手法

伊藤 昌夫　株式会社ニルソフトウェア／有限会社 VCAD ソリューションズ

次世代ソフトウェア信頼性評価技術の開発に向けて

土肥 正　広島大学／岡村 寛之　広島大学

報告

第 12 回クリティカルソフトウェアワークショップ（12thWOCS[2]）開催報告

SECjournal 論文賞

SECjournal 論文賞 受賞論文発表

連載

情報システムの障害状況 2014 年後半データ

松田 晃一　IPA 顧問／八嶋 俊介　SEC システムグループ 主任

SWEBOK V3.0 日本語訳版の連続紹介ー３の１

新谷 勝利　SC7/WG20 エキスパート・新谷 IT コンサルティング

組織の活動紹介

YRP の概要と活動について

森下 浩行　YRP 研究開発推進協会 事務局長

Column

IoT 時代のグローバル競争

IPA 独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

SEC journal No.40
Vol.10 No.6 Mar. 2015

巻頭言 ・・・・・*1*

輝けるIT 立国に向けて
荻原 紀男　一般社団法人コンピュータソフトウェア協会（CSAJ）会長／株式会社豆蔵ホールディングス 代表取締役社長

所長対談 ・・・・・*2*

システムズエンジニアリングの最前線
― 技術・人・社会を含めて考えるシステム開発と運用 ―
ジェイムズ・マーティン　The Aerospace Corporation Principal Engineer

論文 ・・・・・*10*

UMLによる組込みソフトウェア設計の検証支援環境の開発
横川 智教、天嵜 聡介、佐藤 洋一郎、有本 和民、宮崎 仁

概念段階におけるハザード・脅威の識別手法
伊藤 昌夫

次世代ソフトウェア信頼性評価技術の開発に向けて
土肥 正、岡村 寛之

報告 ・・・・・*36*

第12回クリティカルソフトウェアワークショップ
（12thWOCS^2）開催報告
大久保 梨思子　独立行政法人 宇宙航空研究開発機構（JAXA）研究開発本部 情報・計算工学センター（JEDI）
荒川 明夫　独立行政法人 情報処理推進機構（IPA）技術本部 ソフトウェア高信頼化センター（SEC）

SECjournal 論文賞 ・・・・・*40*

SECjournal論文賞 受賞論文発表

連載 ・・・・・*44*

情報システムの障害状況2014年後半データ
松田 晃一　IPA 顧問、八嶋 俊介　SEC システムグループ 主任

SWEBOK V3.0日本語訳版の連続紹介 ー3の1
新谷 勝利　SC7/WG20 エキスパート　新谷ITコンサルティング

組織の活動紹介 ・・・・・*52*

YRPの概要と活動について
森下 浩行　YRP研究開発推進協会 事務局長

Column・・・・・*56*

IoT時代のグローバル競争
鶴保 征城　IPA 顧問　学校法人・専門学校HAL東京 校長

書籍紹介 ・・・・・*57*

編集後記 ・・・・・*58*

SECjournal 論文募集 /IT パスポート試験（i パス）のご案内

輝ける IT 立国に向けて

一般社団法人コンピュータソフトウェア協会（CSAJ）会長
株式会社豆蔵ホールディングス 代表取締役社長
荻原　紀男

昨今、クラウドやスマートデバイスの本格的普及、IoT（モノのインターネット）の進展、ビッグデータ・オープンデータの活用拡大、スマートシティの実現など、ITの活躍する分野はますます拡大しております。

そのような時代の潮流の中、新会長として三つの大きな目標を立てました。

まず、情報発信基地としての役割を担うために、シンクタンク化を目指します。新しい技術、新しい情報をいち早く会員に届け、次のビジネス展開のための種としていただきたく思います。

次に IT 製品・サービスの輸出を拡大し、グローバル化を推進します。我が国は IT 製品について圧倒的な輸入超過の状態が続いております。日本のソフトウェアを輸出するためにはまず人、物、金の国際化を推進し、環境づくりをしていかなければなりません。単に英語化するだけではなく、切磋琢磨して海外の環境に適応したソフトウェアを作らなければならないと思います。

最後にビジネスチャンスの拡大です。首都圏と地方の情報格差をなくし、地方でも先進的な開発が可能な環境を構築していきたいと思います。それにより首都圏と地方の技術格差・情報格差が解消され、地方でも起業が容易になり、また企業の地方移転も含めた地方創生の足がかりとなりたいと思います。

更にこれらとは別に、次代を担うエンジニアを発掘するために、昨年から経済産業省に代わり当協会が事務局となって「U-22 プログラミング・コンテスト」を始めました。小学生から大学生まで幅広く募集した結果、ユニークなアプリや創造性の高いゲーム、そしてかなり高いレベルのシステムまで、称賛に値する作品が数多くあつまり、日本の輝ける未来を確信いたしました。

またそれに付加して優秀な人材をどんどん輩出して日本を IT 立国にするためにアクセラレーターを創設し、その力で多くのベンチャー企業を、資金面だけではなく経営面からも有意義な支援をしていきたいと思います。

次に、来る 2020 年の東京五輪開催に向けて最重要課題であるサイバーセキュリティです。IT は全ての重要インフラにとって必要不可欠な存在であり、サイバーアタックによって五輪会場はおろか首都圏の機能がマヒする危険性を秘めております。東京五輪ではサイバーアタックに対処するために最大８万人のセキュリティエンジニアが必要とされており、一刻も早く人材育成とその組織化に着手する必要があります。当協会はセキュリティ委員会を設置し、IPA セキュリティセンターなどのセキュリティ関連組織と協力しつつ、官民を挙げてのサイバーディフェンスリーグ構築のため、積極的に活動していきたいと思います。

また、情報システムの安全性・信頼性を高めるためにはソフトウェアの品質を向上させる必要があります。当協会は、ISO/IEC25051 に基づくパッケージソフトの品質認証制度「PSQ 認証制度」を通じて、ソフトウェアの品質向上に貢献していきます。

最後に IT 教育です。IT 先進国になるためには早くから授業に IT 教育をとり入れて IT リテラシーを向上させる必要があります。子供たちの IT リテラシーが向上すれば現場の教師、親にも良い影響を与えます。簡単なプログラミングを含めた IT 教育を推進していくことは将来の国づくりにとても重要です。日本には既に一線を退かれた優秀なエンジニアの方が沢山いらっしゃいます。そのような方たちに教師をサポートする立場から支援頂ければ現場もスムーズに動くと思います。また、子供たちに幼いころから IT の正しい利用方法を教えることで、逆にサイバーテロを起こさない、しっかりとした道徳観を植え付ける事になると思います。

当協会は、日本のソフトウェア産業の更なる発展と日本経済の成長のために、時代に即した役割を担っていきたいと思います。

システムズエンジニアリングの最前線
―技術・人・社会を含めて考えるシステム開発と運用―

The Aerospace Corporation
Principal Engineer
ジェイムズ・マーティン

×

SEC 所長
松本　隆明

大規模で複雑なシステムを、ソフトウェアやハードウェアといった従来の視点ではなく、システムの利用者や社会構造などの全体から捉え、高い信頼性をもったシステムを開発・運用するというアプローチがシステムズエンジニアリングの考え方である。その第一人者として活躍するジェイムズ・マーティン氏にお話を伺った。

INCOSE とはどのような組織か

松本：本日は、システムズエンジニアリングに関する国際組織である INCOSE の中心メンバ、ジェイムズ･マーティンさんにおいでいただき、システムズエンジニアリングについてお話を伺いたいと思います。IPA/SEC は、ソフトウェア・リライアビリティ・エンハンスメント・センターの英語名称の通り、基本はソフトウェアを中心にシステムの安全性を考えていくということに取り組んでいます。ところが最近は、システムがどんどん複雑化・大規模化し、更に色々なシステムが複雑に連携する形になっており、これはソフトウェア、これはハードウェアという形で単純に区分けをしてシステムの安全性を考えることはできなくなってきています。

ジェイムズ・マーティン

エアロスペース社所属のエンタープライズ･アーキテクト、システム･エンジニア。INCOSE（The International Council on Systems Engineering）のフェローであり、標準技術委員会のリーダーとしても活動している。SEBoK（Systems Engineering Body of Knowledge）の編集においては BKCASE プロジェクトの主要執筆者として参加し、エンタープライズ・システムズエンジニアリングについての見識を広めた。また、システム工学のプロセスを定義した米国の標準 ANSI/EIA 632 の制定を担うワーキンググループを率いる。過去には AT&T のベル研究所にて、無線通信製品や海底の光通信ケーブル製品に従事した経験を持つ。著作に「Systems Engineering Guidebook」（CRC Press）などがある。

これからは、よりシステム的な見方や考え方、あるいはシステム的なアプローチに注目し、ソフトウェアも含めて、システム全体で捉えていくシステムズエンジニアリングの重要性が増してくると考えています。
さっそくですが、まず INCOSE という組織についてお伺いしたいと思います。

ジェイムズ・マーティン（以下、マーティン）：INCOSE はインターナショナル・カウンシル・オン・システムズエンジニアリングを正式名称とする組織で、1990 年に設立されました。システムズエンジニアリングを行っている様々な企業が、複雑なシステムの設計を単独で行っていくことは難しいということから、協力してそれを解決していくために設立された組織です。従って、当初は、とくに複雑なシステムを扱う可能性の高い航空、宇宙、防衛に関連する企業が参加しました。

松本：会員は一般企業だけでしょうか。それとも、政府あるいは大学の機関・関係者も入っているのですか。

マーティン：二種類のメンバが参加しています。まず個人メンバ。それからコーポレートメンバです。コーポレートメンバには、企業、大学、政府官公庁といった様々な組織が含まれます。しかし、もともと INCOSE は個人メンバのために設立されたものであり、システムズエンジニアリング学会のような、システムズエンジニアリングのプロが集うという趣旨でスタートしています。

INCOSE の活動内容

松本：具体的にはどういう活動をしているのですか。

マーティン：主な活動の 1 つは国際シンポジウムです。これは個人または企業の方から、新しいコンセプト、新しい方法・手法、新しいツール、または事例などを発表していただくものです。それ以外にも、国際的なワークショップを行っています。

また、INCOSE の中には 30 ほどのワーキンググループがあり、それぞれのワーキンググループで、独自の問題に取り組んでいます。例えば、要求、アーキテクチャ、リスク、デザインなどです。

松本：ワーキンググループでは、方法論であるとか、そうした開発を通して標準化につなげていくという活動をされているのでしょうか。

マーティン：現行のやり方を文書化する活動は行っていますが、国際的な標準化団体のように標準作りをするということはしていません。現行のプロセス、手法、ツールなどを記述し、ガイドブックやワークペーパーに編さんしていくという作業を行っています。また、そういうツールがどう実装されているのか、それを文書化し、ワークショップを行う時に、それぞれの領域で行われた作業結果を比較しています。

松本：そのガイドブックは、会員企業以外の人でも見たり使ったりすることはできるのでしょうか。

マーティン：「ハンドブック」と呼んでいますが、有償で購入していただくことはできます。

松本：「ハンドブック」を作って広げていくということには、システムズエンジニアリングの人材育成という意味もあるのですか。

マーティン：「ハンドブック」は、一つの目的のためだけに作られているものではありませんが、INCOSE が行っているシステムエンジニアの認定を受けるための試験は、このハンドブックに基づいています。また、この「ハンドブック」は国際標準 ISO/IEC 15288 に基づいたものです。もちろん、これを個人が仕事のために使うこともできます。

松本：その認定は、INCOSE が認める認定の仕組みですか。

マーティン：システムエンジニアの認定は、INCOSE として行っています。

松本：つまり、ISO の認定ということではなく、それをガイドブックの形で少しわかりやすくしたものの認定を INCOSE が独自に行っていると理解して良いのでしょうか。

マーティン：ISO ではシステムエンジニアが何をやるのか、という記述がありますが「ハンドブック」は、それをより膨らませ、そのプロセスをどのように行っていくのかというところまで含めた内容になっています。先ほど申し上げた認定試験というのは「ハンドブック」に記述されている内容の知識を測ることによって認定していくものです。ただし、認定は試験に受かれば良いだけではありません。知識レベルは試験で調べますが、加えて経験、更に他の人からの推薦が必要です。他の人というのは、ほかの認定されているシステムエンジニアやその他の経験豊富な人からの推薦という意味です。従って知識の部分は、経験と推薦を加えた 3 つの要件のうちの 1 つということになります。

システムズエンジニアリングの拡大

松本：INCOSE における現在の一番のトピックは何ですか。

マーティン：トピックとなっているのは、モデルを使ったシステムズエンジニアリングです。更にもう一つ、よく議論になる重要なトピックは、従来の領域を超えたところに考え方を広げていくということでしょう。例えば、商用のプロダクトエンジニアリングや医療のエンジニアリング、自動車または交通といった領域です。もともとシステムズエンジニアリングが始まったのは、防衛や軍事の世界ですが、そこから更に他のドメインに広げていくということがホットトピックになっています。

松本：モデルベース・システムズエンジニアリング（MBSE）については、SysML などのツールもあり、やり方としてはある程度ポピュラーになりつつあるのではないかと思いますが、MBSE についてはどういう

松本 隆明（まつもと　たかあき）

1978 年東京工業大学大学院修士課程修了。同年日本電信電話公社（現 NTT）に入社、オペレーティング・システムの研究開発、大規模公共システムへの導入 SE、キャリア共通調達仕様の開発・標準化、情報セキュリティ技術の研究開発に従事。2002 年に株式会社 NTT データに移り、2003 年より技術開発本部本部長。2007 年 NTT データ先端技術株式会社常務取締役。2012 年 7 月より独立行政法人情報処理推進機構（IPA）技術本部ソフトウェア高信頼化センター（SEC）所長。博士（工学）。

議論がされていますか。

マーティン：INCOSE の中で話をしているのは、例えば、色々なモデリングの手法と統合していくというようなことです。SysML というのは、ソフトウェアの世界における UML の拡張版と言われています。ソフトウェアの世界からシステムの世界へ広げていくということで生まれてきたのが SysML ですが、それを様々なツール、様々なモデルと統合していくという話をしているのです。というのも、既存のシステム、または従来型のシステムが様々あり、分析のモデル、パフォーマンスモデル、更にはシステムシミュレーションなどといったものを、SysML と統合していくということです。

システムというのは、単なるハードウェアやソフトウェアというより、もっと大きなものです。システムの中には人も含みます。その人の部分をどうモデリングするのか、ということについては、SysML にはできない。その人の部分のモデリングのために、どういうモデリング言語を使うのか、あるいは分析ツールを使うのか、というところも大きな問題になってきます。システムの部分、またはセキュリティというところにも人はかかわってくる。その人の部分をどうモデリングするのか、というのが大きな問題なのです。

松本：それは重要なポイントだと思います。ヒューマン・ファクター（人的要因）というのは、システム全体を考えた時に、非常に大きな比重を占めるものですね。

私たちも今、色々なシステムの障害がなぜ起きているのか、ということを調べているのですが、やはり人の要因によるところがかなりあります。人を具体的にどのようにモデルの中に取り込んでいくのかについて、INCOSE では何か具体的なアイデアを出されているのでしょうか。

マーティン：残念ながら INCOSE でその部分を考えていくのは困難であると思っています。というのは、INCOSE のメンバは、そのバックグラウンドとして、ハードウェアまたはソフトウェアの世界の人間です。従って、人の側面、またはソーシャルエンジニアリングや社会学の側面を一番の専門としているメンバではないのです。人の部分に対して、最も適切に包括的に考え方を作り出していく適切なメンバではないのですね。ただ、人の部分の本当に一部ではありますが、そういった部分を扱うワーキンググループはあります。ヒューマンセイフティのワーキンググループやエンタープライズ・アーキテクチャのワーキンググループです。

システムエンジニアは何をするのか

松本：INCOSE の会員の数は、どんどん増えているのでしょうか。

マーティン：現在のメンバ数は 8,000 程ですが、毎年少しずつ増えています。

松本：日本では 3 年ほど前に JCOSE という支部のようなものができていますが、まだ日本の国内では、システムズエンジニアリングの技術者はそれほど多くないというのが実状ではないかと思います。日本とアメリカあるいは海外との違いということで、何か感じられるところはありますか。

マーティン：今 8,000 と申し上げた INCOSE のメンバのうち、半分の 4,000 は、米国以外のメンバです。もともと、米国のみのメンバで始めたものが、かなり米国以外で増えているという状況です。

現在は、先ほどもお話したように、防衛だけでなく商用の領域にも広げていこう、としているわけですが、その際の難しい点が、商用の領域になった時に、システムエンジニアがシステムエンジニアとは呼ばれていないというところです。例えば、プロダクトエンジニアまたはマニファクチャリングエンジニアと呼ばれているので、システムエンジニアであると認識されていない。それが一つの障壁になり、そういう人達にアクセスしにくいという状況が生まれています。

松本：そもそもどういうことをやる人がシステムエンジニアなのか、そこが明確に理解されていないということがありますね。

マーティン：おっしゃる通りです。そもそもエンジニアが何をやるのか、ということがわかりにくい。エンジニアが行うのは、抽象的なものであったり数学的なものであったりします。それがシステムエンジニアになると、更に抽象的、更に数学的で、具体的なものではない。結果が目に見える有形のものではないということによって、システムエンジニアが何をやる存在なのかという理解が難しくなっているのだと思います。

松本：私が伺ったマーティンさんのプレゼンテーションの中で、PICARD 理論のお話がありました。システムというのは、人によって見方が全然違ってくる。システムというのは、単にパーツの組み合せだけではなくて、色々な見方によって変わってくるものだと。それをエンジニアリングしていくというのが、システムズエンジニアリングの仕事になってくるということでしょうか。

マーティン：エンジニアリングはパーツにフォーカスしたものだと思いますが、システムズエンジニアリングと言う時には、パーツ間の関係、これをエンジニアリングするものだと思います。従ってパーツよりも、そのパーツ間の関係や、パーツ間の相互作用、つまりインターアクションの重要性が高くなります。
実は私の PICARD 理論ですが、これは日本で作った理論なのです。ある電機メーカに、システムのコンセプトを教える時、自分の母国ではない国で、違う言語で、大変複雑なコンセプトを教えなければならない。そのため理解していただきやすい方法が必要であるということから策定したのが PICARD 理論でした。
松本：同じパーツを組み合わせても、使われるコンテクスト、状況に応じてシステムは変わってくるし、何を目的にパーツを組み合わせ、システムとして提供するかによっても、見方は違ってくる。つまりシステムには色々な属性がある、と考えればいいのでしょうか。
マーティン：システムズエンジニアリングは、そうした様々な属性を調べて、システムがきちんと正しい方法で振る舞い、正しい成果を出せるようにしていくということを行っていきます。その際は、正しいコンテクストで正しい成果が出せるように考えていくと共に、未知の環境、想定されていなかったような状況の中でも堅牢性が持てるようにしていかなければなりません。システムが使われる環境というのがすべて知り尽くされているとは限りません。従って、未知のコンテクストに対しても、きちんとした振る舞いができるようなシステムにしていくということも、考慮に入れる必要があります。
松本：未知の環境というのは、未知なので結局はわからないわけですね。例えば、どのように世の中が変化していくか、ビジネスが変わっていくか。技術が進歩していくか。それを予測するのは、非常に難しいと思うのですが、その点はどう考えていらっしゃいますか。
マーティン：一つの方法は、システムを学習型にしていくということです。システム自体が新しい環境を理解し、それに対して適応ができるようにしていくということ。そうすれば、複雑なシステム、適応型のシステムになり、設計自体は難しくはなりますが、未知の環境に対して対応ができるようになると共に、システムを未知の脅威、未知の条件に対して対応できる堅牢なものとすることもできます。また、様々な多数のシナリオを使ったテストを行っていくことも考えられます。更に、進化型のプログラムにしていく、ということも考えられます。システム自体が学習し進化していく。将来が一足飛びに予測できなくても、暫時出てくる条件に合わせて進化していくという考え方です。

「ユーザ要件」だけでは不十分

松本：先ほどのお話にあった、システムを色々な属性で見ていくという時に、誰がそのシステムを見るかによっても変わってくるのではないかと思います。つまり、ビジネスパーソンがシステムを見る時の捉え方と、技術者や開発者がシステムを捉える見方と、今度は運用する人が、オペレーターも含めて、システムとして捉える見方というのは、それぞれ変わってくると思うのですが、どういう視点でシステムを捉えるかということについて、一つの考え方はあるのでしょうか。
マーティン：以前は、システムズエンジニアリングということを行っていく際、ユーザ要件だけが考えられていました。しかし、それでは十分ではないということで、今ではオペレーター、マネージャ、オーナー、構成管理をしていくビルダー、検証をするテスター、更に政府、スポンサー、資金調達をする銀行、こういったところすべてを含めて、いわゆる“ステークホルダ”、利害関係者と呼ぶ人を考慮に入れる必要があります。一番最初から、フロントエンドでシステムのエンジニアリングをする時から、ユーザ要件だけを考えるのではなく、こういう様々なステークホルダの要件を考えていく、というように考え方を膨らませているのです。
松本：そうすると、システムズエンジニアに求められるスキルが、ビジネスのこともわからなければいけない、ユーザビリティもわからなければいけない、開発者として、実際の開発をどうするか、ということもわからなければいけない、というように本当に幅広い知識が必要になってくると思うのですが、そういう人を育てるのは、かなり難しいのではないかという気がします。
マーティン：おっしゃる通りです。一つのシステムエンジニアのチームに、すべてのそうした部分のエキスパートをそろえるということはできません。システムエンジニアが、そのチームの中で、いわゆるファシリテーター（進行役）の役割を担い、様々なステークホルダを関与させていく、そういったことが必要だと思います。外部の人たちにアクセスし、そこから情報を吸い上げ、それを理解し、問題に関しての全体的な理解が取れるようにしていくということが必要になると思います。

標準フレームワークとしての「7SAMURAI」

松本：マーティンさんが提案されているものに「7SAMURAI」があります。システム構築の方法を、7つの視点から考えていくという意味で“七人の侍”という比喩が使われているのだと思いますが、それについてご説明いただけますか。

マーティン：基本的には、まずエンジニアリングで考えていきます。ソリューションというのは、システムの中の一つを考えていくわけですが、それと共に、システムが存在する全体的なコンテクスト、このコンテクストたるシステムというものも考えて行かなければなりません。そして、開発されたソリューションですが、例えば、それを開発する開発組織というものも別の一つのシステムとして考えていく必要があります。更にまた、開発されたソリューションが実際に効果を発揮できるかどうかというところにかかわる実現のためのシステム、これが、どれぐらいのシステムなのかによって制約を受けることになるので、その実現システムに合わせたソリューションにしていくという調整も必要になります。

松本：一つの問題をどう解決するか、システムによってどう解決するか、ということを考えるのが最初のターゲットだと思うのですが、それをどんどん広げていくということをしていくと、検討の範囲が広がり、実際に開発すべきシステム範囲が、なかなか確定できなくなるのではないかという心配があるのですが、そのあたりは、どのように絞っていけばいいとお考えですか。

マーティン：この「7SAMURAI」というのは、標準のフレームワークを確立するものだと考えています。問題に対して、フレームワーク、枠組みを調整していくことができる。良い枠組みがあれば、様々な情報が入って来ても、その情報を整理して考える手助けになっていきます。情報同士がどうインターアクションをしていくのか、というところも考えていかなければなりません。7つのシステムというのは、標準のやり方でインターアクションをしていく形になります。そのフレームワークを持つことによって、色々な情報、考え方を整理することができますし、何かを見過ごしてしまう、忘れてしまうということを回避することにもつながります。

例えば、何らかのソリューションにフォーカスをしてしまうと、問題の解決だけはできたけれども、それが持続できない、持続するためのシステムに制約があるために、その解決が長続きしないということが起こります。そういった見過ごされてしまうものを、無くしていくということにも使えるフレームワークなのです。

松本：実際にシステムの設計で使われた例は、あるのでしょうか。

マーティン：まだありません。リサーチの早い段階にあるものなので、実践的な方法論として確立されるところまではまだ至っていません。

松本：実際に、先ほどの会員企業などで、こうした考え方に共鳴されているという方もあるのでしょうか。

マーティン：企業ではまだありません。使われているのは学術の世界です。システムズエンジニアリングを教える際に使われています。システム的な思考ということで、カリキュラムに入れている大学が幾つかあります。

松本：実際に産業界に広げていくために、どういったところが重要になってくるのでしょうか。たとえ研究的なものでも、実際に産業界で使われないと効果が出てこないということがあると思います。どうやって実際の現場に適応していくか、お考えがありますか。

マーティン：まずは学術界でこの考え方に沿った方法論を策定する必要があると思います。または SysML を、この枠組みに合わせて広げていくということを行っていくことが、実際に産業界で採用されていくためには必要でしょう。SysML を「7SAMURAI」に適応させていくためにどうすればいいのか。こういったところには、リサーチグループや研究のプログラムが必要になると思います。従って、まず方法論やツールを策定していく。現在私が考えている方法論は、一般的に使うためのものではありません。企業なら企業が、自社で使われているシステムなどに適合させていく、ということが求められると思います。

松本：「7SAMURAI」の考え方は、非常にすぐれたものだという印象を受けています。7つという視点が適切かどうかは、私にはよくわからないのですが、それ以外にも、既存のシステムとの協業も考えていかなければいけないと思います。いずれにしても、単に一つのソリューションを考えるのではなくて、色々な視点を持ってシステムを考えていかないと、きちんとしたシステムができないというのは、非常に良く整備されたアイデアだと思います。

一方で産業界に実際に広めていくためには、ツールや仕掛けがないと、なかなか難しいのかも知れないですね。単にコンセプトだけで、これでやりなさい、と言っても

産業界ではすぐに対応できない。具体的に SysML で書けるとか、色々なツールがそろってくると、徐々にそういう考え方が浸透していくのかも知れないと思います。

マーティン：おっしゃる通り、ある意味では科学的理論のようなものだと思います。エンジニアは、科学的理論を直接適用するということはしません。それをいったん、方法論や手法、またはツールという形に変換し、変換した上で採用する、というものになると思います。今の段階は、システムサイエンスという概念であり、それをシステムズエンジニアリングのツール、またはメソッドに変換して、初めて採用という形になっていくのだと思います。

松本：ところで一つお聞きしたかったのは、なぜ「七人の侍」という名前を付けられたのか。その動機はどの辺にあるのでしょうか。

マーティン：最初に、そのシステム単体だけを考えていては足りない、ということを思いました。他のシステムを全体的に見ていかなければならない、と考えを進めていったところ、７つのシステムを考えなければならないという考えに至り、７という数字が出て来ました。私の好きな映画監督が黒澤明だったので映画の「七人の侍」が浮かんで来ました。そのストーリーが、ちょうど合うと思ったわけです。

ストーリーはご存知だと思いますが、個人個人の侍は、もちろん強いわけですが、それだけではなく、彼らが協力し力を合わせることで、更に力を増していく、というものです。システム同士のシナジーという考え方、またシステムの全体というのは、システムを構成するパーツの総和よりも大きなものになるのだ、という考え方にちょうど合う映画だと思いましたので、映画タイトルを使いました。一人一人では問題が解決できない。しかしながら、それが手を携え、協力をすることによって、全体として問題が解決できる、コラボレーションをする、インターアクションをする。それによって、適切な振る舞いになっていくのだという、全体的なシステムの考え方に合うということです。

エンタープライズ・システムズエンジニアリングとは

松本：マーティンさんが主張されているもうひとつの新しい考え方で、エンタープライズ・システムズエンジニアリングという概念があります。ここで言われているエンタープライズというのは、基本的にはビジネスだと思えばいいのでしょうか。

マーティン：エンタープライズと言う時には、大きな、複雑なアクティビティを行うもの、という考え方をしています。そのためビジネスをイメージされることが多いと思いますが、ではすべてのビジネスがエンタープライズかといえば、そうではない。例えば、長きにわたって事業を行っているような企業、ルーティンで業務をこなし、構造はきちんとできている、色々なものがすべて予測できるという場合には、エンタープライズという考えではないですね。エンタープライズと言う場合には、色々な挑戦的で難しいことがあり、不確定要素や困難が多々あるということです。従って、例えば複数の企業が、一社ではできない困難なことを行うために協力をするという時に、その複数の全体を指してエンタープライズという言い方ができるかも知れません。

松本：そうすると、エンタープライズ・システムズエンジニアリングという考え方は、エンタープライズをいかにシステムとして構築するかと捉えればいいのでしょうか。例えば、最近で言うと、スマートコミュニティとかスマートシティのような幅広いコンセプトや概念を、いかにシステムとして構築するかということが、エンタープライズ・システムズエンジニアリングの一つだというように。

マーティン：２つあると思います。１つは大きなエンタープライズのアクティビティの中で使われる大きな技術的なシステムやソーシャルの部分の中にある技術的な部分。そして、もう１つは大きな複雑なエンタープライズが、技術とどういうインターアクションを持つのかということです。そういったところをエンジニアリングはすべて一緒に考えて行かなければなりません。

松本：そういうソーシャルな部分とかテクニカルな部分とか、色々な局面の視点から、システムを幅広く捉える、というのが、エンタープライズ・システムズエンジニアリングだと思えばいいわけですね。

マーティン：そうですね。社会の動き、ソーシャルのダイナミクスであるとか、人の動き、人間の力学であるとかダイナミクス、こういったものも全体の大きなアクティビティの一環となっていきます。個人個人が、どうシステムの一環として動いていくのか、というところだけではなく、システム間の連携など、大きな目標のためにコーディネーションを取っていかなければならないのだ、という考え方です。

松本：そうすると、やはりヒューマンの問題というのが、

一番大きなファクターになってくるような気がするのですが、先ほど人をどうやってモデル化するのかというのが非常に難しいというお話がありました。エンタープライズ・システムズエンジニアリングの中でも、人の要素をどのように位置付けていくかという点が、同じように難しい問題になるような気がします。

マーティン：そうです。更には、人を個人としてモデリングするだけでも十分ではない。人をグループとして見ていく。そのグループとの相互作用、グループ対グループという場合には、例えば協力をしたり、または競い合ったり、というような動き、力学が発生するという場合もあるので、そういうモデリングも必要になってきます。人の側面を考えるという場合には、ソーシャルサイエンスもそうですし、ソーシャルなダイナミクス、それから行動学、それからビジネス。こういったすべてのモデルが必要になります。

松本：そういう意味で言うと、もう一つのキーワードはダイナミズムということになるでしょうか。時々刻々変化していくものを、どう捉えていくか。つまり、今までのシステムズエンジニアリングは、どちらかと言うと静的なモデリングのようなイメージが強い気がするのです。そこにもう少しダイナミックなものを入れているというところが、もう一つのポイントと言えるでしょうか。

マーティン：人は動的なものです。予測しづらい。そして、人はコントロールされることを嫌います。ソフトウェアやハードウェアを考える際には、すべてコントロールをするということを考えます。ソフトウェアにしてもハードウェアにしても、その挙動、振る舞いに対してきちんとタイトにコントロールしていきますけれども、人はなかなかそれができるものではないですね。コントロールされることを好む反面、自由を欲しがりますし、個人の選択ができるということを求めます。そこで重要になるのが、経済学であり心理学であり、または社会生理学であるということだと思います。電子や複合化合物や、エネルギー変換といったことだけを考えればいいのではなく、ソーシャルダイナミズムまで考えて行かなければならない、ということです。

IT ソリューションを部分として含むもの

松本：似たような考え方として、だいぶ前に、ザックマン・フレームワークをベースにしたエンタープライズ・アーキテクチャの考え方というのが出てきてたのですが、あれは、どちらかと言うと、エンタープライズと言っても、もう少し企業に閉じたような狭い範囲で、マーティンさんが言われているエンタープライズ・システムズエンジニアリングのように幅広くはないと思いますが、考え方としては、それに近いでしょうか。

マーティン：似ていると思います。ただ、ザックマン・フレームワークに関しては、コンピューティングシステムを考えているだけのものであったと言えると思います。企業全体の IT ソリューションのアーキテクチャであるということですね。

しかし私が言っているのは、IT にとどまらず、より高いレベル、技術だけではなくソーシャルまたは組織、または人、こういったところを含めた全体のものです。もちろん、その中でエンタープライズレベルの IT を使うこともありますけれども、それらの全体を指しているということです。

松本：そういう意味では、これを IT にカスタマイズすることもできると考えられるのでしょうか。

マーティン：このエンタープライズ・システムズエンジニアリングとエンタープライズ・アーキテクチャを、一緒に IT のために使っていくということは可能だと思います。ただし、このエンタープライズ・システムズエンジニアリングは、IT だけではない、というところは注意しておかなければならないと思います。

エンタープライズ・システムズエンジニアリングという言葉を聞くと、どうしても IT だけに使うものと考えがちになってしまうんです。

松本：この考え方は、具体的にどこかに適用していこうという話があるのでしょうか。

マーティン：INCOSE の中に、交通運輸を扱うトランスポテーション・ワーキンググループというものがあります。インテリジェント・トランスポテーションシステムという部分に、幅広いエンタープライズ・システムズエンジニアリングの考え方を適用していこうという考え方があります。

松本：それは例えば、最近注目されている自動運転というようなものも含まれるのでしょうか。ああいうものは人間的な要素も非常に重要な部分を占めるので、社会活動全体で捉えていかないと、単に自動でブレーキをかけたり、運転したりという形だけでは非常に狭い範囲のソリューションにしかならないと指摘されています。そういうものも視野に入っているということでしょうか。

マーティン：インテリジェント・トランスポテーションシステムの中に、自動車は一部として含まれますが、そ

れだけではなく、自動車が全体的な交通システムと、どう一緒に機能するのか、全体的な社会インフラとどう連動するのか。例えば、都市における金融システムとどう関係するのかというところまで含めて、統一したやり方で全体的な大きな目標を実現していくという考え方。これがインテリジェント・トランスポテーションシステムです。

目標の一つとしては、例えば、事故を減らす、全体的な交通システムの安全性を上げていく。またはスループットを上げていく。遅延を減らしていくということを、グローバルな都市全体として考えていくというものになります。

システムズエンジニアリングにおける安全性

松本：安全性のお話が出てきましたけれども、私たちもシステムの安全性をどうやって確保していくかということについて力を入れようとしています。エンタープライズ・システムズエンジニアリングで考える時に、全体の安全性というのは、どのように捉えればいいのですか。

マーティン：エンタープライズにおける目標というものを、考えていかなければならないと思います。もちろん、エンタープライズにとって安全というのは、一つの関心事であるわけですが、例えば、成功の確率とバランスを取っていくということも必要になります。宇宙プログラムで、例えば宇宙船を火星に飛ばすというプロジェクトがあった場合、様々な安全の問題、課題というのが出てきます。ただ、それを本当に完璧にすべて、ということになると、その目的自体の実現可能性が無くなってしまう。それでは目標が果たせないということになるので、そこのバランスを取っていくということも必要になります。宇宙プログラムであった場合には、それは有人なのか無人なのか、ということによっても安全の捉え方というのは、変わってくると思います。また、原子力などに関しても、安全というものを考えていかなければなりません。

松本：安全性、そしてコスト面も当然重要な問題だと思いますし、利用者の立場から見ると、ユーザビリティや、色々な視点があると思います。そういった視点も、先ほどのエンタープライズ・システムズエンジニアリングの中に入ってくるのでしょうか。

マーティン：先ほどステークホルダの分析という話をしました。ユーザがステークホルダだと考えれば、ユーザは安全を求める。しかし、別のステークホルダであるオーナーは、利益追求をしたいということからコスト効率を高めることを求める。安全をきちんと実現することと相反してしまうかも知れない、そうしたこととの間でバランスを取っていくことが必要になります。

幅広いステークホルダがいれば、安全、ユーザの使い勝手、経済性、セキュリティ、という様々な目標があります。これらのバランスを取って、エンタープライズの中で両立させていくことが求められるということだと思います。

松本：そのバランスをどう取ればいいのか。色々なステークホルダがいる時に、誰を重要視するかというのは、ケースバイケースで違ってきますね。

マーティン：そのベストのバランスを見極めていくのは、アーキテクチャを決めていくところで行っていきます。アーキテクチャ作りの作業の一環の中で、誰がステークホルダなのかを理解し、どういう関心事、どういうプライオリティがステークホルダにあるのか、ということを理解していくわけです。そして、それに基づいたアーキテクチャを作っていくわけですけれども、そこでは様々な理論を借用して使っていくことができます。オペレーションの分析、ミッションの分析、またはビジネスの分析。こういったものが活用できると思います。

松本：なるほど。私たちも、安全性を考える上で、もっとシステム的に見ていかなければいけないと考えてきましたが、これまで私たちが考えていたシステムというのは、いわゆるITシステムといった少し狭い範囲であったかも知れません。マーティンさんがおっしゃるような、もっと幅広い社会システム的なものまで含めて捉えていかないと、全体が見えてこないということをあらためて感じました。

本日は、幅広いお話をありがとうございました。

UMLによる組込みソフトウェア設計の検証支援環境の開発





横川　智教[†]　天嵜　聡介[†]　佐藤　洋一郎[†]　有本　和民[†]　宮崎　仁[‡]

情報システムは年々複雑化しており，その信頼性を保証することが大きな課題となっている．上流工程で網羅的に設計を検証できるモデル検査の優位性が認識されているが，専門知識やノウハウの習得の難しさが問題となっている．本論文ではモデル検査による設計検証を支援するツールを提案する．本ツールでは開発者が普段記述している設計文書から自動でモデル検査に必要な文書を生成し設計を検証できる．協力企業で実際作成された設計文書へ本ツールを適用した結果，その有用性が確認できた．

Tool Support for Verification Environment of Embedded Software Design with UML

Tomoyuki Yokogawa [†], Sousuke Amasaki [†], Yoichiro Sato [†], Kazutami Arimoto [†] and Hisashi Miyazaki [‡]

Information systems get more complicated recently, and assuring their reliability has been an urgent problem. Model checking is known as one of advantageous technology because of its nature: exhaustive and automatic verification available in early development phase. However, it is also known for difficulty in learning special knowledge. This paper proposed a tool supporting model checking on software design artifacts. This tool can perform automatic verification on software design artifacts written in a well-known notation by translating them into specialized format files suitable for a model checking software and performing verification on the files. In cooperation with a software development company, we confirmed that this tool can help software reliability assurance.

1. はじめに

情報システムは年々複雑化しており，どのように信頼性を保証するかが大きな課題となっている．クリティカルな分野のソフトウェアでは1件の不具合が社会に及ぼす影響が甚大であるため，特に品質保証が重要視される．ソフトウェア開発における修正のコストは開発工程の後半に進むほど大きくなるため，設計検証に基づいた上流工程での品質保証が求められる．

設計検証による品質保証のために有効な手段と考えられるのがモデル検査 [Clarke1999] である．モデル検査では，ソフトウェアの振る舞いを有向グラフによってモデル化し，グラフの網羅的探索により求める特性が満たされるか否かを自動的に検証することが可能である．モ

【脚注】
† 岡山県立大学 情報工学部
‡ 川崎医療福祉大学 医療技術学部

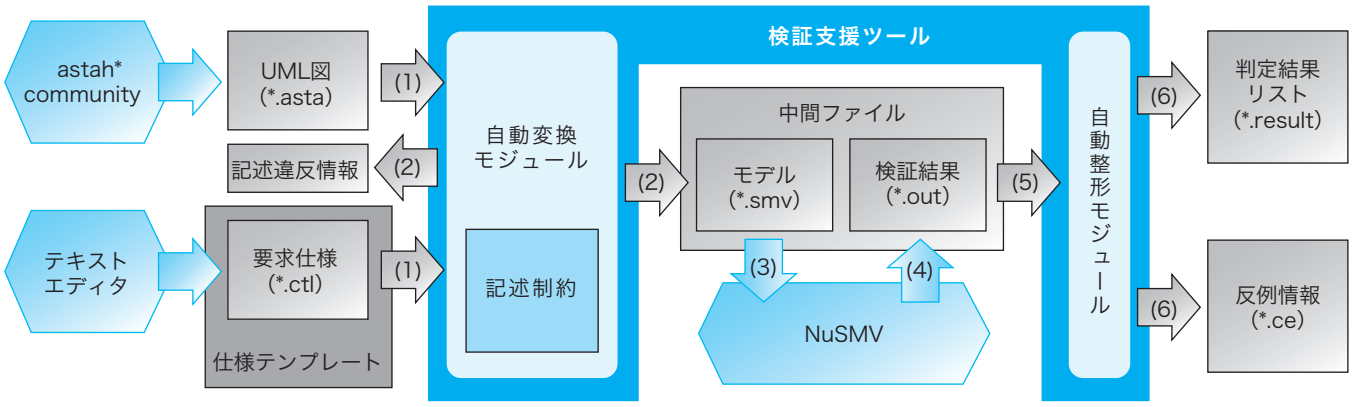


図 1　本ツールを用いた設計検証の枠組み

デル検査を用いることで設計の矛盾や仕様との不整合を自動でかつもれなく検出することが可能となる．

モデル検査を実施するためのツールは SMV[McMillan 1993] や SPIN[Holzmann1997] をはじめとして数多くのものが公開されている．しかしながら，モデル検査を設計検証に導入するためには，ソフトウェアの設計文書をツール固有のモデル化言語で記述しなければならない．モデル化言語の記法は，ソフトウェア開発者が通常使用している設計文書とは大きな隔たりがあり，設計文書の記述規則は実務的な利便性に重きが置かれる一方で，モデル化言語ではシステムの振る舞いを厳密に記述するための意味論が定められている．したがって，ソフトウェアの設計文書をもとにモデル化言語による記述（以下，モデルという）の作成を行うには専門的な知識やノウハウが必要となる．

そこで本論文では，モデル検査技術を組込みソフトウェアの設計検証に導入する上でのコスト削減という課題の解決を目的として，設計文書からモデルを自動生成する検証支援ツールを開発する．本ツールでは，開発現場において広く用いられている設計記法の一つである UML(Unified Modeling Language)[UML] を対象として，モデルの自動生成を行う．また，モデル検査ツールとして，記号モデル検査アルゴリズム [Burch1990] に基づく高速な検証が可能である NuSMV[NuSMV] を用いる．本研究の主な貢献点は以下の 3 つである．

(1)　UML 描画ツール astah* community[astah] との連携
(2)　NuSMV の入力モデルへの自動変換
(3)　ツールの公開 [Tool]

本ツールにより，UML 描画ツールで作成された設計文書をモデル検査ツールの入力モデルへと自動変換を行うことが可能となり，モデル検査を設計検証へと導入する上でのコストが大幅に削減できる．これにより，上記の課題が解決される．

本論文では，協力企業から提供を受けたソフトウェア設計文書に対して本ツールを適用し，検証を行っている．適用実験を通して，本ツールの導入により，新たなコストを要することなくモデル検査による設計検証が実現できることを確認している．さらに，NuSMV によって検出された反例が設計誤りの修正に有効であることを併せて示している．

2.　検証支援ツール

2.1. 概要

本ツールは，astah* シリーズのソフトウェアの 1 つで無償利用が可能である UML モデリングツールである astah* community(以下，単に astah* という) で記述された UML によるソフトウェア設計文書とソフトウェアが満たすべき仕様を入力とし，NuSMV に対応するモデルに変換し出力する．仕様の記述はあらかじめ定められたテンプレートを用いて行われる．また，本ツールは NuSMV による検証結果を整形し出力する機能を有する．

UML はソフトウェアの異なる側面を記述する 13 種類の図 (UML 図) から構成されるが，本ツールでは，組込みソフトウェアの振る舞いを記述するために利用され，利用頻度の高い状態マシン図を取り扱う．本ツールはコンソールアプリケーションとして Java を用いて実装されており，コマンドラインで起動する．

2.2. 設計検証の枠組み

本ツールを用いた設計検証の枠組みを図 1 に示す．設計検証は以下の手順で実施される．

ステップ 1.　ユーザは astah* で作成した状態マシン図 (*.asta) と，仕様テンプレートに基づいて記述した要求仕様 (*.ctl) を作成し，本ツールに入力する．この ctl ファイルはテキスト形式で作成する．

ステップ 2.　本ツールにより NuSMV の入力となるモデル (*.smv) が生成される．このとき，asta ファイル

として入力された状態マシン図が記述制約を満たさなかった場合はエラーとなり，記述違反箇所の情報が出力される．

ステップ 3. ユーザは smv ファイルを NuSMV へ入力して検証を行う．

ステップ 4. NuSMV によって状態マシン図が要求仕様を満たすか否かについて検証した結果 (*.out) が出力される．

ステップ 5. ユーザは out ファイルを本ツールに入力する．

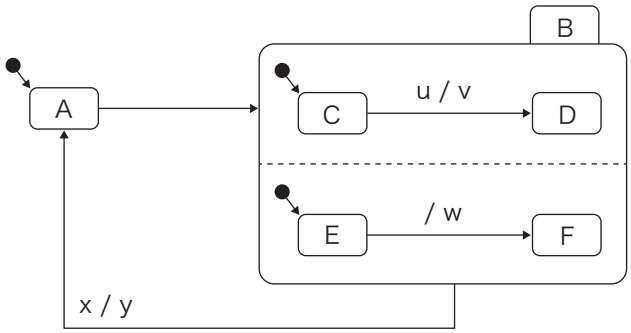


図 2　合成状態をもつ状態マシン図

表 1　状態マシン図に関する制約

	記法	可否
状態関連	単純状態	○
	合成状態	×
	直交状態	×
	擬似状態	
	初期擬似状態	○
	履歴	×
	ジョイン・フォーク	×
	接合点	○
	選択擬似状態	○
	入場点，退場点	×
	終了状態	○
	入退場アクション	×
	状態内アクション	×
遷移関連	変数	△
	アクション	
	メッセージ送信	○
	変数更新	△
	イベント	○
	ガード条件	
	活性状態に関する条件	○
	変数に関する条件	△

○： 記述できる
△： 一部記述できる
×： 記述できない

ステップ 6. 本ツールにより検証結果が整形され，要求仕様が満たされるか否かに関する判定結果リスト (*.result) が生成される．ここで，ステップ 4 において NuSMV による検証の結果，要求仕様が満たされなかった場合は，NuSMV が生成した反例に関する情報 (*.ce) を出力する．result ファイルおよび ce ファイルはテキスト形式で保存される．判定結果リストからは ctl ファイルとして入力された要求仕様のどれが満たされ，どれが満たされなかったについての情報が得られる．また，反例からは，要求仕様を満たさないような動作系列についての情報を得ることができる．

2.3. UML の記述制約

UML は実務的な利便性に重きが置かれており，記述上の意味論について解釈が分かれる部分が多い．例えば，合成状態をもつ状態マシン図では，遷移の交互実行の解釈は設計者の意図と必ずしも一致しない．例として，図 2 に示すような合成状態をもつ状態マシン図において，状態 A から合成状態 B へと遷移した際は状態 C と状態 E がアクティブとなる．その後イベント u を受信すると状態 C から D へ遷移するが，その際に並行してイベントをもたない状態 E から F への遷移が行われるか否かの解釈は設計者によって異なる．そこで，本ツールでは対象とする UML の記法に制約を与え，意味論が一意に定まる記述のみを自動変換の対象とする．

状態マシン図に対する記述制約を表 1 に示す．ここで可否が△となっているものは，以下のように部分的に制約が加えられていることを示す．まず，変数は整数型およびブール型に限るものとし，アクションにおける変数の更新については，右辺に変数および定数の四則演算をもつ代入文によってのみ行われる．そして，ガード条件上での変数に関する条件は，変数と整数値に関する線形制約に限るものとする．線形制約とは，「mx ～ n」の形 (x は変数，m,n は整数，～は＜ , ＞ , ＝のいずれか) をもつ式と，否定，論理和および論理積からなる論理式のことを指す．

表 2　仕様テンプレートと対応する CTL 式

No	仕様テンプレート	CTL 式
1	safe(変数 = 状態)	AG !(変数 = 状態)
2	safe(メッセージ)	AG !(メッセージ =TRUE)
3	safe(変数，値)	AG !(変数 = 値)
4	live(変数 = 状態)	AF (変数 = 状態)
5	live(メッセージ)	AF (メッセージ =TRUE)
6	live(変数，値)	AF (変数 = 値)
7	reachable(変数 = 状態)	EF (変数 = 状態)
8	reachable(メッセージ)	EF (メッセージ =TRUE)
9	reachable(変数，値)	EF (変数 = 値)

2.4. 仕様テンプレート

モデル検査を用いて設計検証を行うためには，対象となるシステムが満たすべき仕様を時相論理 [Pnueli1977] を用いて記述する必要がある．NuSMV では時相論理の一種である CTL(Computation Tree Logic)[Clarke1981] を用いる．ここで，時相論理とは「いつかある状態に到達する」や「常にある性質が満たされる」といったシステムの性質を，状態の遷移や時間の経過の観点から記述するための論理体系である．しかしながら，ソフトウェアが満たすべき仕様を，時相論理式として記述するためには，論理学や離散数学などの専門的な知識を要する．そこで，ソフトウェアが満たすべき仕様を入力するためのテンプレートを提供する．これにより，専門的な知識を有しないユーザであっても記述が容易となる．

本ツールで用いる仕様テンプレートを表 2 に示す．仕様テンプレートは，状態マシン図の重要な要素である状態，メッセージ，そして変数に関する特性を記述することが可能であり，特性として利用頻度が高い安全性 (safe)，活性 (live)，そして到達可能性 (reachable) の 3 つを記述することが可能である．

2.5. モデルへの変換

図 3 に NuSMV の入力言語によるモデルの基本構成を示す．モデルは変数宣言部，状態遷移系記述部および検査式記述部からなる．変数宣言部では検査対象となるモデルの要素を変数として宣言する．状態遷移系記述部では，モデルの各要素が，他の要素の値に基づいて定義される遷移条件に依存して，どのように変化するかを記述する．そして検査式記述部では，満たすべき性質を CTL による検査式で記述する．本ツールでは，変数はブール型と列挙型の二通りのみを用いる．以下に，本ツールにおける状態マシン図から状態遷移記述への変換および仕様テンプレートから検査式記述への変換について述べる．

状態マシン図から状態遷移記述への変換

状態遷移記述部では，状態マシン図の遷移の発火による状態，メッセージおよび変数の値の変化を case 文で記述する．状態の変化は以下のルール 1 ～ 3 に従って記述し，メッセージおよび変数の変化はルール 4 に従って記述する．

ルール 1. 実行可能な遷移が 1 つのみならば，その遷移先の状態へと変化する．

ルール 2. 実行可能な遷移が 2 つ以上存在するならば，いずれかの遷移先の状態へと変化する．

ルール 3. 実行可能な遷移が存在しなければ，状態は変化しない．

ルール 4. メッセージおよび変数の値を変化させる遷移のいずれかが実行されたときかつそのときのみ，それに従って値を変化させる．

仕様テンプレートから検査式記述への変換

検査式記述部では，テンプレートを用いて記述された仕様を CTL による検査式として記述する．仕様テンプレートから得られる CTL 式を表 2 に併せて示す．まず，安全性については CTL の AG 演算子によって記述する．AG 演算子は，全ての実行系列において常に成り立つような性質を表す．よって，「AG !(式)」により，「与えられた式が決して成り立たない」という安全性に関する特性を表すことができる．ここで「!」は論理否定を表す演算子である．次に，活性については AF 演算子によって記述する．AF 演算子は，全ての実行系列においていつか成り立つような性質を表す．よって，「AF (式)」により，「与えられた式がいつか必ず成り立つ」という活性に関する特性を表すことができる．最後に到達可能性に

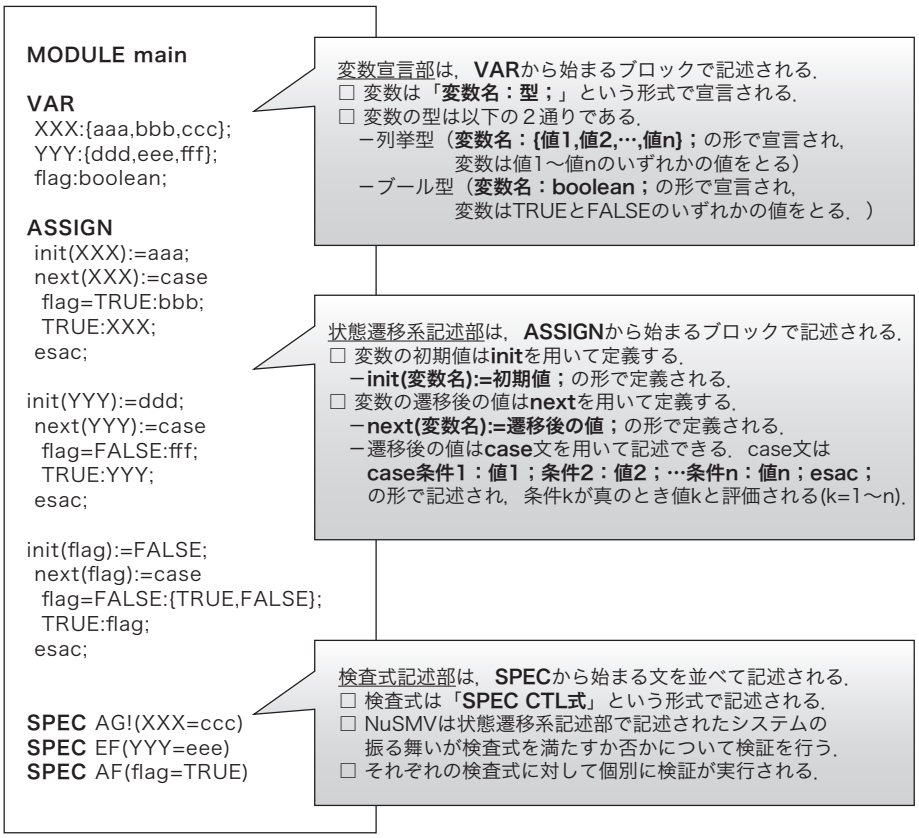


図３　モデルの基本構成

ついては EF 演算子によって記述する．EF 演算子は，いずれかの実行系列においていつか成り立つような性質を表す．よって，「EF (式)」により，「与えられた式がいつか成り立つ可能性がある」という到達可能性に関する特性を表すことができる．

3. 適用事例

3.1. 概要

協力企業から提供された状態マシン図に対して本ツールを適用し NuSMV による検証を行った．この状態マシン図は，実際の開発で用いられた状態遷移表などをもとに作成されたものである．適用対象のシステムは，店舗従業員向けの商品供給指示システム（R-CDS）である．R-CDS の概要は以下のとおりである．

- 売場従業員 (以下，従業員) と商品管理者 (以下，管理者) との通話システムである．
- 管理者は従業員に什器への商品の補充指示を発令し，従業員は補充の結果や現場の状況報告等を行う．従

表３　R-CDS の状態遷移表

イベント	状態			
	wait	send	receive	com
report	send			
accept			com MD_A=MD_A+1	
stop_report		wait		
reject			MD_A=0: wait MD_A=1: com	
finish_A				MD_A=1 : wait MD_A=MD_A-1
order				MD_A=1 : receive
response		com MD_A=1		
stop_order			MD_A=0 : wait MD_A=1 : com	
non_ response		wait		
finish_Ctl				MD_A=1 : wait MD_A=MD_A-1

表４　仕様テンプレートで記述した検査特性

検査項目	テンプレートで記述した仕様
状態の到達可能性	reachable(Terminal_A = send)
	reachable(Terminal_A = receive)
	reachable(Terminal_A = com)
	reachable(Terminal_B = send)
	reachable(Terminal_B = receive)
	reachable(Terminal_B = com)
通信量の到達可能性	reachable(MD_A, 1)
	reachable(MD_A, 2)
	reachable(MD_B, 1)
	reachable(MD_B, 2)
通信量の安全性	safe(MD_A, 3)
	safe(MD_A, -1)
	safe(MD_B, 3)
	safe(MD_B, -1)
管理者側端末の表示の到達可能性	reachable(Ctl_Monitor = surplus)
	reachable(Ctl_Monitor = sufficient)
	reachable(Ctl_Monitor = optimal)
	reachable(Ctl_Monitor = few)
	reachable(Ctl_Monitor = short)

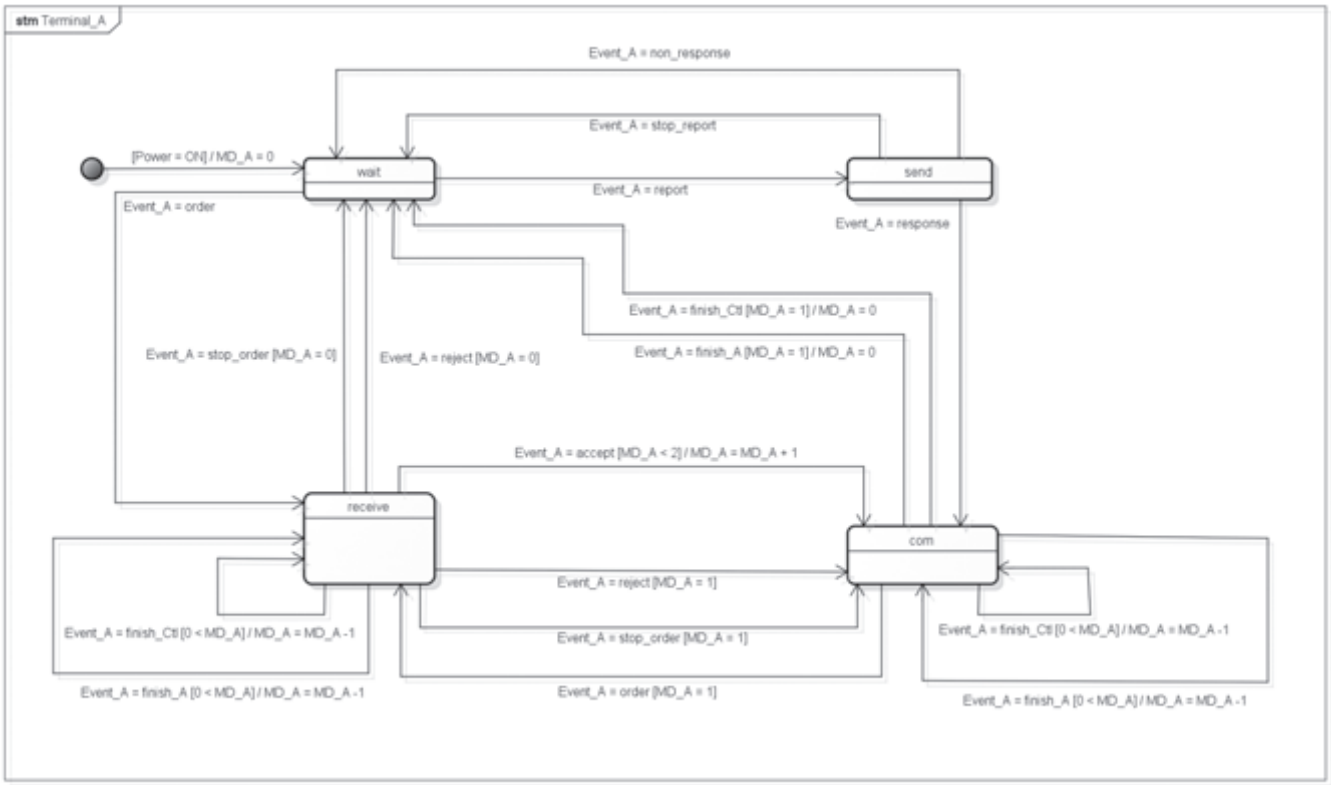


図４　端末 A の状態マシン図

業員が持つ端末は同時に 2 通話が可能である．
・管理者側の端末には，通信回線の利用数に応じて，従業員の配置数を評価する指標が過剰 (0) ～不足 (4) の 5 段階で表示される．

3.2. 例題システム

R-CDS の状態遷移表を表 3 に示す．ここでは，従業員は 2 名としそれぞれの持つ端末を端末 A および端末 B とする．表 3 の行はイベントを，列は端末 A の状態を表す. 各セルの上段は遷移の条件と遷移先を [条件 : 遷移先] の形で記述する．下段は遷移の際に実行されるアクションを表している． ここで MD_A は端末の通信モードを表しており，MD_A =0 の場合は通信しておらず，MD_A =1 および 2 の場合は 1 回線および 2 回線で通信していることを表している．端末 B の動作は，表 3 の変数 MD_A を MD_B で置き換えて得られる状態遷移表によって記述される．端末 A の状態遷移表から作成した状態マシン図を図 4 に示す．端末 B についても同様に状態マシン図を作成する．

R-CDS では，変数 MD_A および MD_B の値に応じて管理者側の端末に従業員の配置数を評価する指数が 5 段階で表示される．通信回線の利用数が多いことは，売り場への商品補充指示が多いことを表しており，従業員が不足することを意味する．逆に，通信回線の利用数が少ないことは，従業員が過剰であることを意味する．MD_A および MD_B の値に応じた管理者側の端末への表示の変化を表す状態マシン図を図 5 に示す．

本実験では，まず (1) 状態遷移表の各状態への到達可能性，(2) 各端末の通信量に関する到達可能性，(3) 各端末の通信量に関する安全性，(4) 管理者側の端末の表示に関する到達可能性，の 4 つの特性について検証を行った．これらの特性は，仕様テンプレートを用いて表 4 に示すように記述できる．

次に，事例提供元から要望があった 3 つの特性について検証を行った．検証した特性は，(1) 端末が待機状態 (wait) にもかかわらず通信中である状態への到達可能性，(2) 端末が 2 回線使用中にもかかわらず着信中 (receive) となる状態への到達可能性，(3) 端末が通話中 (com) にもかかわらず通話なしである状態への到達可能性，である．これらの特性は仕様テンプレートを用いず直接 CTL を用いて記述した．検証に用いた CTL 式を表 5 に示す．

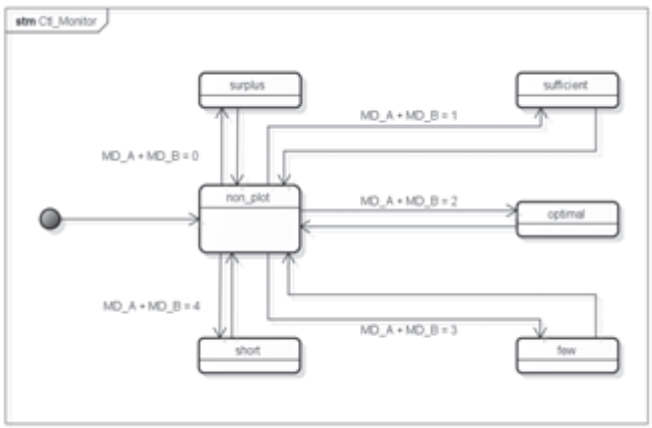


図 5　管理者端末への表示の状態マシン図

3.3. 結果

NuSMV による検証結果から得られたファイルを図 6 に示す．図 6(a) に示す通り，表 4 に示す特性はすべて TRUE となり，誤りがないことが確認できた．次に，図 6(b) に示す通り，表 5 の特性については，3-A および 3-B が FALSE となり，特性を満たさないという結果が得られた．図 6(c) の 3-A に対する反例を解析したところ，Terminal_A=com，すなわち端末 A が通信中となったにもかかわらず，MD_A の値が 0，すなわち通話なしの状態のままとなる系列が存在していた．Terminal_A の値が send から com となる遷移を確認したところ，表 3 の状態遷移表で記述されていた MD_A=1 のアクションが，図 4 の状態マシン図では記述されていなかった．そのため，Terminal_A=com となっても MD_A の値が 0 から変わらず，3-A の CTL 式に違反する状態へと到達していた．以上から，この誤りはユーザによる状態マシン図作成時のミスであることが判明した．

3.4. 評価

適用実験の結果をうけて，事例提供元からは，今回は転記ミスが原因ではあったものの，記述の漏れや間違いを発見できる技術として，モデル検査は非常に有用であるという評価を得た．さらに，違反が検出された際に，反例を用いて原因を特定できる点についても高く評価された．一方で，モデル検査の導入を容易にするという面から，本ツールの有効性についても一定の評価を得た．また，ツールの機能面に関して，状態マシン図の全ての記法に対応して欲しいとの要望をうけるともに，GUI の実装による利便性の向上についての要望があった．

表 5　事例提供元の要望に基づく検査特性

No	検査項目	CTL 式
1-A	端末が待機状態にもかかわらず通信中である状態への到達可能性	!EF (Terminal_A = wait & !(MD_A = 0))
1-B		!EF (Terminal_B = wait & !(MD_B = 0))
2-A	端末が 2 回線使用中にもかかわらず着信中となる状態への到達可能性	!EF (Terminal_A = receive & MD_A = 2)
2-B		!EF (Terminal_B = receive & MD_B = 2)
3-A	端末が通話中にもかかわらず通話なしである状態への到達可能性	!EF (Terminal_A = com & MD_A = 0)
3-B		!EF (Terminal_B = com & MD_B = 0)

4. 考察

4.1. 既存の技術との比較

UML の状態マシン図を対象としてモデル検査による設計検証を行うための研究開発は，これまでにも盛んに行われている [Bhaduri2004]．中でも，SPIN を用いたもの [Latella1999] と，本ツールと同様に SMV を用いたもの [Chan1998][Clarke2000] が著名である．

Latella ら [Latella1999] の提案した手法は，状態マシン図の構造を階層化オートマトンで表現した上で，それらを PROMELA 言語によるモデルへと変換し，モデル検査器 SPIN を用いて検証を行うものである．

一方で，Chan ら [Chan1998] の提案した手法では，状態マシン図と同様の構造および意味論をもつ設計記法である RSML を対象として，SMV による検証を行っている．モデル化においては，遷移による状態や変数の値の変化を本ツールと同じく case 文によって記述する．そして，Clarke ら [Clarke2000] の提案した手法では，状態マシン図の各遷移による動作を論理式として書き下し，結合することによって状態マシン図全体の振る舞いをモデル化している．

Latella らの手法では，複数の状態マシン図を用いて記述される振る舞いは考慮していない．本論文の事例のように，複数のコンポーネントが相互に通信を行うようなソフトウェアの設計を対象とする場合は，手法の拡張が必要となる．Chan らの手法では決定的動作のみを考慮している．すなわち，複数の遷移が同時に実行可能となり，非決定的にどちらかが選ばれるような振る舞いを

```
(001) EF Terminal_A = send  is true
(002) EF Terminal_A = receive  is true
(003) EF Terminal_A = com  is true
(004) EF Terminal_B = send  is true
(005) EF Terminal_B = receive  is true
(006) EF Terminal_B = com  is true
(007) EF MD_A = 1  is true
(008) EF MD_A = 2  is true
(009) EF MD_B = 1  is true
(010) EF MD_B = 2  is true
(011) AG !(MD_A = 3)  is true
(012) AG !(MD_A = -1)  is true
(013) AG !(MD_B = 3)  is true
(014) AG !(MD_B = -1)  is true
(015) EF Ctl_Monitor = surplus  is true
(016) EF Ctl_Monitor = sufficient  is true
(017) EF Ctl_Monitor = optimal  is true
(018) EF Ctl_Monitor = few  is true
(019) EF Ctl_Monitor = short  is true
```

(a) 表 4 の特性に対する検証結果
(result ファイル)

```
(001) !(EF (Terminal_A = wait & !(MD_A = 0)))
is true
(002) !(EF (Terminal_A = receive & MD_A = 2))
is true
(003) !(EF (Terminal_A = com & MD_A = 0))  is
false
(004) !(EF (Terminal_B = wait & !(MD_B = 0)))
is true
(005) !(EF (Terminal_B = receive & MD_B = 2))
is true
(006) !(EF (Terminal_B = com & MD_B = 0))  is
false
```

(b) 表 5 の特性に対する検証結果
(result ファイル)

```
(001) !(EF (Terminal_A = wait & !(MD_A = 0)))  is true
(002) !(EF (Terminal_A = receive & MD_A = 2))  is true
(003) !(EF (Terminal_A = com & MD_A = 0))  is false
-> State: 1.1 <-
  Terminal_A = initial
  MD_A = 0
  Power = start
  Event_A = emp
-> State: 1.2 <-
  Power = ON
  Event_A = report
-> State: 1.3 <-
  Terminal_A = wait
-> State: 1.4 <-
  Terminal_A = send
  Event_A = response
-> State: 1.5 <-
  Terminal_A = com
  Event_A = report
(004) !(EF (Terminal_B = wait & !(MD_B = 0)))  is true
(005) !(EF (Terminal_B = receive & MD_B = 2))  is true
(006) !(EF (Terminal_B = com & MD_B = 0))  is false
-> State: 2.1 <-
  Terminal_B = initial
  MD_B = 0
  Power = start
  Event_B = emp
-> State: 2.2 <-
  Power = ON
  Event_B = report
-> State: 2.3 <-
  Terminal_B = wait
-> State: 2.4 <-
  Terminal_B = send
  Event_B = response
-> State: 2.5 <-
  Terminal_B = com
  Event_B = report
```

(c) 表 5 の特性に対する反例
(ce ファイル)

図 6　NuSMV による検証結果から得られたファイル

もつ状態マシン図は扱うことができない．Clarke らの手法では，遷移に基づく振る舞いを全て論理式として表現している．そのため，case 文によるモデル化と比して探索対象となるモデルが複雑となり，検証コストが大きくなる可能性がある．また，このモデルでは COI(Cone of Influence) オプション [Berizin1998] による高速化の効果も得られ難い．

また，これらの手法では，検査対象となる仕様の入力支援についてはサポートしておらず，ユーザは時相論理を用いて直接仕様を記述する必要がある．そのため，専門知識をもたないユーザがこれらの手法を導入して設計検証を行うことは困難である．

このほかにも UML を対象とした設計検証手法は数多く報告されているが，その大部分は個別の例題への適用事例であり，汎用的に利用できるものではない．また，モデル化の手続きを一般化したものについても，実装したツールを現場で即座に利用できる形式で公開しているものは筆者の知る限りでは存在していない．

4.2. 産業界への展開

モデル検査技術はソフトウェアの設計を網羅的に検証し，設計の無矛盾性や仕様との整合性を確認するために有効である．また，モデル検査技術によるソフトウェア設計の検査について産業界からの期待も大きい．このような状況にも関わらず，組み込みソフトウェアの設計検証にモデル検査技術を導入した事例はまだまだ少ない．一部企業での導入事例は存在するが，産業界からの期待の大きさに比してモデル検査による設計検証を導入している企業の数は少ない．モデル検査を設計検証に導入している事例が少ない原因として，モデル作成の困難さが指摘されている．

この問題を解決するために，本ツールは，モデル検査による設計検証を行う上でのモデル作成を支援として，UML 図で記述されたソフトウェア製品の設計図を入力とし，モデル検査ツールの入力形式に自動で変換・出力できる．しかしながら，本ツールでは自動変換の効率化のために，UML 図の記法の多くの部分に制約をかけており，そのトレードオフとして利便性はある程度低下してしまう．適用事例における事例提供元との意見交換においても，状態マシン図のすべての記法に対応してほしいと要望を受けている．

また，本ツールの成果として，安全性・活性・到達可能性という登場頻度の極めて高い検査特性をテンプレートによって記述することが可能である．また，状態マシン図において重要な要素である状態・メッセージ・変数について扱うことが可能であることから，本研究で開発した仕様テンプレートは，ソフトウェアの代表的な特性を記述するにあたって十分な表現能力があると考えられる．しかしながら，より幅広い層のユーザへの普及のためにも，仕様パターンによる検査特性の記述への対応は取り組むべき課題であると考えられる．

最後に，本ツールでは現在のところコマンドラインでの実行のみが可能であり，GUI はサポートされていない．利便性を向上し，普及を進める上では GUI の実装が必要不可欠であると考えられる．

5. まとめ

本論文ではモデル検査によるソフトウェアの設計検証を支援するツールを開発した．協力企業で実際作成された設計文書へ本ツールを適用した結果，その有用性が確認できた．検証できる UML 図や性質の拡張およびツールのインタフェースの改良が今後の課題である．

謝辞

本研究は，独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（SEC: Software Reliability Enhancement Center）が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けたものです．

【参考文献】

[Clarke1999] E.M. Clarke, O. Grumberg and D. Peled: Model Checking, MIT Press (1999).

[McMillan1993] K. McMillan: Symbolic Model Checking, Kluwer Academic (1993).

[Holzmann1997] G. Holzmann: The Spin model Checker, IEEE Trans. Soft. Eng., 23(5), pp.279-295(1997).

[UML] O. M. Group: Unified Modeling Language Specification, Object Management Group (2001). http://www.uml.org.

[Burch1990] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang: Symbolic model checking: 10^{20} states and beyond, Proc. of the Fifth Annual IEEE Symp. on Logic in Computer Science (1990).

[NuSMV] NuSMV. http://nusmv.fbk.eu/.

[astah] astah* community http://astah.change- vision.com/.

[Tool] http://circuit.cse.oka-pu.ac.jp/tool.html/.

[Pnueli1977] A. Pnueli: The temporal logic of programs, Proc. of 18th IEEE Symp. Foundations of Computer Science (FOCS'77), pp.46-57 (1977).

[Clarke1981] E.M. Clarke and E.A. Emerson: Design and synthesis of synchronization skeletons using branching time temporal logic, Proc. of Logics of Programs Workshop, vol. 131 of Lecture Notes in Computer Science, pp.52-71 (1981).

[Bhaduri2004] P. Bhaduri and S. Ramesh: Model Checking of Statechart Models: Survey and Research Directions, ArXiv Computer Science e-prints (2004).

[Latella1999] D. Latella, I. Majzik, and M. Massink: Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker, Formal Aspects of Computing, 11(6), pp.637–664, (1999).

[Chan1998] W. Chan, R. J. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin and J. D. Reese: Model checking large software specifications, IEEE Trans. Software Eng., 24, 7, pp. 498–520 (1998).

[Clarke2000] E. M. Clarke and W. Heinle: Modular translation of statecharts to SMV, Technical report, Carnegie-Mellon University School of Computer Science (2000).

[Berizin1998] S. Berezin, S.V.A. Campos, and E.M. Clarke: Compositional reasoning in model checking, Proc. of Int'l Symp. on Compositionality: The Significant Difference (COMPOS'97), pp.81–102, (1998).

概念段階におけるハザード・脅威の識別手法

伊藤　昌夫[†1, †2]

本論文では，概念段階におけるハザードおよび脅威の識別手法を示している．システムの安全性およびセキュリティを確保するためである．ネットワーク接続された組み込み機器において，ソフトウェアの重要性が高まるとともに，安全性およびセキュリティ確保の重要性も増している．概念段階とは，システム要求仕様を記述するための開発初期段階を指す．この概念段階に適用可能なハザードおよび脅威の識別手法は，これまで存在しない．ハードウェア中心のシステムの場合，漸次的な変更が主となるため，その必要性が低かったためと考える．しかし，新しいソフトウェア中心のシステムが増え，その重要性が増すと共に，概念段階におけるハザードおよび脅威の識別手法が求められている．本論では，アイテムスケッチとゴールモデルを用いた要求の整理と，これらの記述を利用した，ハザードおよび脅威を見つけるための手順を示す．乗用車のための機能安全規格との対応も示した．手法の適用例としては，先進運転支援システム（ADAS，例えば [1]）に関係するシステムを用いた．但し，本論での議論は，広い範囲で適用可能と考えている．

Finding Hazards and Threats in Concept Phase

Masao Ito[†1, †2]

Abstract

In this paper, we propose an approach for finding hazards and threats. Those are relating to safety and security respectively, that become important in especially the software-intensive embedded system such as ADAS (Advanced Driver Assistance Systems). The definition of the concept phase is the process in which we consolidate the requirements and create the specification. There is no appropriate method that we can apply in this phase to find both hazards and threats. Because the hardware-centered system usually evolves gradually, but recent new software-intensive systems born out of scratch and need the method to analyze hazards and threats. In this paper, we mainly focus on finding hazards and threats to assure the system safety and security. We use the item sketch and goal model and apply the guide words. We use the standards and example of the automobile for the explanation purpose, but we believe that we can apply this method to various domains.

1. はじめに

ソフトウェアが重要な位置を占めるシステムが，ますます我々の日常生活と密接に関わるようになっている．また，IoT(Internet of Things) [2] という言葉に代表されるように，これらシステムは，ネットワークを介して相互に接続され

【脚注】
†1　株式会社ニルソフトウェア
†2　有限会社 VCAD ソリューションズ

つつある．従って，システムの安全性およびセキュリティが，これまで以上に重要になってきている．

特に，近年進歩が著しい先進運転支援システム（ADAS, Advanced Driver Assistance Systems）は，その代表である．ADAS は，駐車支援・衝突緩和ブレーキ・先行車追従などのシステムを包含した名称である（本論では，ADAS を手法の例として用いている）．ADAS は，運転者の支援や代行を行う．システムの役割は大きく，安全性に大きな影響を及ぼす可能性がある．また，効果的な ADAS を実現するために，車両同士，あるいは道路上のインフラシステムとネットワーク結合されつつあるため [3]，セキュリティへの対応も重要な課題となっている．

本論では，概念段階においてハザード・脅威を識別するための手法を示す．安全性・セキュリティを確保するためには，ハザード・脅威を識別することが開始点となる．安全性の確保とは，網羅的にハザードを識別し，それに対する処置を決めることである．セキュリティに関しても同様に脅威を識別し，その脅威に対する対処を決定することが必要である．

なお，本論の概念段階は，乗用車の機能安全規格である ISO 26262[4] の第三部に対応する．もちろん，車両に限らず全てのシステム開発において，概念段階は存在する．ISO 26262 規格は，現段階で，もっともよく概念段階を整理していると考えており，本論ではこの規格との対応についても考える．

信頼性解析のための手法として，FTA（Fault Tree Analysis）や FMEA（Failure Mode and Effect Analysis）などの既存の手法（ハザード分析全般に関し整理した書籍としては，例えば，[5]）がある．これら手法を，概念段階において使用することはできない．既存手法は，システムを対象とし，そのシステムが，明確に分解されていることを前提としているためである．

また，よく知られているように，信頼性と安全性・セキュリティは，異なる概念である．システムが高い信頼性を持っていたとしても，事故の時に安全側に移行しなければ，安全性が高いとはいえない．もちろん，概念段階を除けば，共通する部分も多い．我々の方法においても設計段階では，FTA や FMEA を用いる．

後述するように概念段階では，システムそのものではなく，その抽象概念である「アイテム」を対象とする．「アイテム」という新しい概念を用いる理由として，ソフトウェアとハードウェアの進化の速度が影響していると考えている．機械に代表されるハードウェアの場合，大きな構造の変化は少ないため，漸次的に良くしていくことになる．逆に，ソフトウェアの比重が高いシステムにおいては，新しいシステムを相対的に作りやすい．従って，ADAS に代表されるソフトウェアの比重が高いシステムが増えている今，概念段階における新しい手法が求められている，と考えている．

本論で示す手法は，2 つの良い特徴を持つ．

- システム開発における概念段階で利用可能である
- ハザードおよび脅威の識別を同時に行うことができる

全体の構成は，次の通りである．2 章で，解決すべき課題を整理する．3 章では，最初に手法の概要を説明する．次に，例を用いながら，具体的なハザード・脅威の識別手順について説明する．4 章では，既存の研究との比較を行う．最後に，5 章でまとめを行う．

2. 課題

本章では，概念段階におけるハザード・脅威識別を行うときの課題整理を行う．最初に，概念段階を定義し，用語「アイテム」について考える．次に，安全性とセキュリティの関係について考える．これら課題整理は，先に挙げた本手法の 2 つの特徴と対応している．

2.1 概念段階と「アイテム」

最初に，概念段階とは何かについて整理する．本論でいう概念段階は，ISO 26262 の第三部に示されている範囲に含まれる．特にハザードと脅威の識別に関しては，第三部の 5.4.1 から 7.4. までが，対応する箇所である（表 2 参照）[1]．

概念段階は，アイテムの定義から始まる．規格の中で，用語「アイテム」は，通常とは異なった意味を持っていることに注意が必要である．アイテムとは，システムの抽象表現である [2]．例えば，衝突緩和ブレーキシステムを取り上げる．アイテムは，特定の車に搭載される具体的なシステムではない．衝突緩和ブレーキシステムとは何かを考えて，機能や非機能を整理し，アイテム境界を定めた，衝突緩和ブレーキの本質である．

アイテムではなく，具体的なシステムからスタートすると，そのシステムの枠内で，安全性やセキュリティを考えるしかない．従って，対応する範囲も限定的となる可能性がある．アイテムを対象にするのは，開発の初期段階において，本質的な安全性やセキュリティを持つシステムを目指すためである，と考えることができる．

ADAS に代表される車両の情報システムは，これまでにない新しいシステムである．従って，システムの抽象概念であるアイテムを用い，概念段階で，本質的な安全性確保を図る．そこから段階的に詳細化していくことで，新規の場合であっても，安全なシステムとすることを狙いとしている．

以上より，概念段階での課題は，新しい用語であるシステムの抽象表現としての「アイテム」を，如何に表現するか，ということになる．

【脚注】

1 ISO 2626 は 10 部から構成されている．パートと項番を組み合わせて示す場合がある．例えば，第三部の 7.4.2 を 3-7.4.2 と記述する．

2 アイテムの定義は規格中に 2 種類存在する．第一部の用語定義のものと，第十部の定義である．ここでは，最後に発行された第十部の定義を用いる．「（具体的なシステム）は抽象表現であるアイテムから生成（現実化）したものである」．第一部の用語定義とは異なっているが，第十部の定義の方が，他の箇所との整合性は高い．

2.2 安全性とセキュリティ

安全性・セキュリティを，ともに「危害から＜対象＞を守ること」と考えると，両者を同一の枠組みで扱うことができる．＜対象＞が人の場合，安全とは，人を危害から守ることである．＜対象＞がアセット（資産）の場合，セキュリティとは，アセットを危害から守ることである[3]．

一方で，違いもある．安全性の場合，その関心は，設計・実装上の検討不足（信頼性）や構成品の故障である．それに対して，安全機構を組み込み，問題が生じたときにどのようにふるまうかを検討する．一方，セキュリティの場合は，悪意ある第三者を想定した上で，設計・実装上の検討不足（脆弱性）を除去することに，関心がある．安全性・セキュリティを脅かすハザード・脅威の識別において，この違いを考慮する必要がある．

どういう結果になれば問題と考えるか，ということに関しても違いがある．例えば，アセットに対する侵害は，セキュリティの場合，直ちに問題となる．安全性においては，アセットの十全性が破られ，人に危害が生じる可能性があった時，はじめて安全性の問題となる．

従って，ここでの課題は，ハザードと脅威の類似性と差異を考慮しつつ，両者を適切に識別するための方法を見つけることである．

なお，本論で扱うセキュリティは，情報空間上に限定している．セキュリティという言葉は，情報空間以外でも用いられる．自動車に関して例を挙げると，物理的に車両に侵入し，ECU（Electronic Control Unit）ボードをすり替える，或いは，OBD（On-Board Diagnostics）から情報を抜き出すといったこともセキュリティ上の問題である．しかし，本論では，物理的なセキュリティ上の侵犯を扱わない．あくまで情報空間に限定している．ちなみに，情報空間上でセキュリティが問題になった事例として次がある．キーレスエントリーシステムにおいて，キーの保持者の近傍から車両まで情報をリレーすることにより，キーを保持していないにもかかわらず車両を操作可能となったケースである[6]．ここでは，物理的なセキュリティ上の問題は生じていない．純粋な情報空間上の問題である．

3. プロセス

本論で示す手法には，次の 4 つのステップがある．

- アイテムスケッチを作成する (3.1)
- アイテムのゴールモデルを作成する．同時にアイテムスケッチも詳細化する (3.2)
- 各ゴール記述文にガイドワードを適用する (3.3)
- アイテムスケッチを利用してハザードおよび脅威を識別する (3.4)

アイテムスケッチでは，静的および動的なアイテムの定義を行う．ゴールモデルでは，アイテムが持つゴール（トップゴール）の詳細化を行う．

ゴールの詳細化に合わせて，アイテムの静的・動的表現も詳細化する．具体的には，ゴール木の各階層（抽象度レベル）に応じて，アイテムスケッチを用意する．なお，ゴール木とは，後述するゴールモデル中の木構造のことである．厳密には，非循環有向グラフとなるが，説明のために，本論では「ゴール木」という表現を用いる．

次に，各ゴールの記述文にガイドワードを適用する．

ガイドワード適用文とアイテムスケッチを利用して，最終的なハザードおよび脅威の識別が可能となる．

注意すべきは，ゴールおよびアイテムスケッチの抽象度に応じた記述を，それぞれ最後まで維持することである．即ち，抽象レベルに応じたゴールがあり，それに対応するアイテムスケッチが存在する．詳細（低い抽象度）のみをメンテナンスすることは，アイテムに対する理解や，再利用の観点から不利である．

3.1 アイテムスケッチ

アイテムスケッチは，アイテムの構造・ふるまい，およびアイテム境界を明確にするために用いる．アイテムの機能・非機能要求はゴールモデルが受け持つが，ゴールモデル中の各ゴール記述文を利用することで，ゴールとアイテムスケッチは，対応関係を，ムリなく維持することができる．

3.1.1 ゴール記述文とアイテムスケッチ表現

アイテムスケッチには，静的および動的表現がある．静的表現は，図式（例えば UML のクラス図，SysML[7] の内部ブロック図，或いは CATALYSIS アプローチ [8] の仕様型表現）により与えることができる．本論の例では UML を用いている．

また，以降では，CACC（Cooperative Adaptive Cruise Control）[9] を，ゴールモデルの題材としている．CACC とは，車間制御クルーズコントロールシステム (ACC，Adaptive Cruise Control system [10]) に，車車間通信を付加したものである．CACC では，ACC が持つイメージ等から得た情報に加えて，先行車両と通信することによって，先行車両の情報を取得する．先行車を運転している人のアクション（例えばブレーキ踏下情報）が，通信を介して直ちに伝わる．レーダ等による測距結果のみを用いる場合と比べ，より素早く後続車は反応できる．他車と通信するため，計算機や携帯電話同様に，セキュリティ上の懸念があり，本論では，例に用いている．

例えば，次の要求文を考える．

(S1)（自車は）先行車を認識する.

S1 のアイテムスケッチは，図 1 の (b) である．後述する

【脚注】

3 もともと，古いラテン語の securus は，両者の意味を持っている．現代においても例えば，ロマンス語系に属するフランス語では，sécurité は，安全もセキュリティも表す．
しかし，本論では，安全を脅かす危害の発生可能源のことを「ハザード」，セキュリティの場合は，アセットに対する危害の発生可能源を「脅威」として，区別している．

ゴールモデル中のゴール（図 1 の (a)）から，(b) の静的表現を得ることができる．ここでは，自車，先行車をクラスとして含むパッケージ「認識」を示している．

なお，この図において，「自車」というクラスを追加している．要求では，主語に相当する部分を省略することが多いため，必要に応じて補う必要がある．また，パッケージ名として，「認識」を用いている．本来，機能に相当する部分（ゴール記述文の述部）は，アクションとしての記載が望ましい．CATALYSIS であれば，仕様型中で，アクションとして記述することができ，より素直な表現とすることができる．

アイテムスケッチの動的表現は，有限状態機械図によって与えることができる．次の要求文を考える．

(S2) 先行車を認識したならば，追従中状態に移行する．見失った（ロスト）場合は，待機中状態に移行する（図 2 左上 , FSM_A).

(S3) スイッチを ON にすることで，CACC は ON（モード）になる．OFF にすることで，OFF（モード）になる (図 2 左下 , FSM_B).

状態と遷移のトリガが要求中に記載されていれば，容易に要求と状態遷移図の間で対応付けを行うことができる．

アイテムスケッチの静的・動的表現は，同一対象の 2 つの側面を示している．例えば，静的表現における先行車と自車の追従関係（図 1 の (b)）は，動的表現（図 2）の FSM_A における「追従中」状態での関係である．また，FSM_B におけるスイッチは，静的表現の一部として，表現することになる（図 4 参照）

3.1.2 開発カテゴリとアイテムスケッチ

ところで，概念段階を開始する時に，アイテムスケッチのどの抽象度レベルから始めるかは，開発カテゴリに従って決まる．開発カテゴリとは，対象のアイテムが，新規のアイテムか，それとも，すでに解析済みのアイテムかということである（[4] の 3-6.4.1 あるいは，表 2 参照）．

あるシステムを作成するときに，既存のアイテム記述が存在し，その一部を変更する必要がある場合，解析済みのアイテムスケッチを利用することができる．このとき，影響が生じるレベルから解析をスタートする．もちろんゴールモデルに対しても必要な変更を加える．

3.1.3 概念段階終了時のアイテムスケッチ

本論のスコープ外であるが，概念段階は，要求仕様書の完成により終了する（安全性に限れば，並行して作成する機能安全要求仕様書の完成である）．アイテムスケッチは，このとき，抽象ハードウェア・ソフトウェアアーキテクチャのベースとなる．

3.2 ゴールモデルとゴールの詳細化

ゴールモデルは，安全性およびセキュリティ確保に限らず，アイテムの機能・非機能要求を整理するために用いる．本論では，KAOS アプローチ [11] のゴールモデルを利用する．KAOS は，代表的なゴール指向要求分析手法の一つである．

ゴールモデルを用いて，最上位の要求を詳細化し，最終的に要求仕様を得ることになる． ここで用いる記法を，簡単に説明する．主たるノードは，ゴール・ソフトゴール・障害・解決・仕様である．ソフトゴールは，ゴールの達成基準が明確に示されないゴールである．障害は，ゴールの達成を妨げるノードであり（3.2.3 項），その解決は，解決ノードを用いて示す．仕様は，最終的に確定した要求であり，ゴールモデルの葉の部分に相当する．

上位のゴールと下位ゴールは，AND 洗練あるいは OR 洗練によって結合する．前者（AND 洗練）は，ゴールの分解である．逆に言えば，全ての下位ゴールを合わせたものが

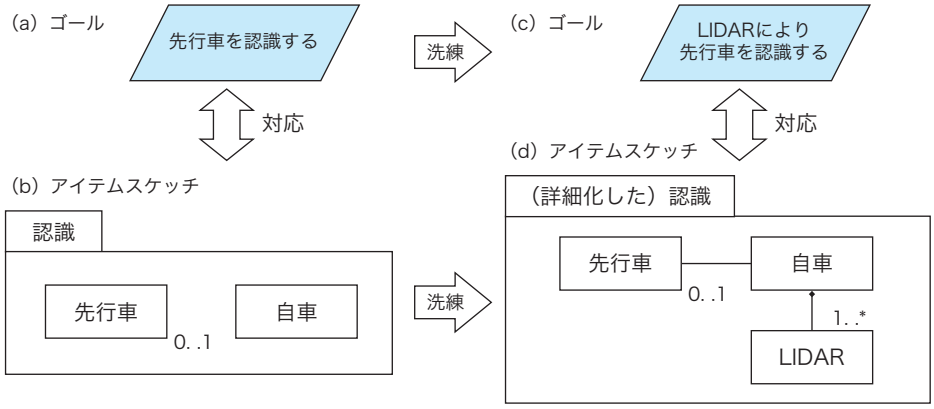


図 1　ゴールと対応するアイテムスケッチ（静的表現）

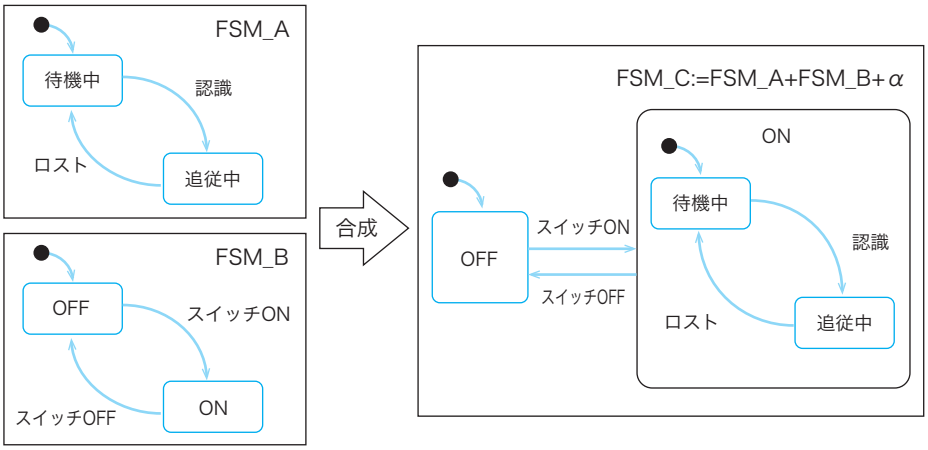


図 2　アイテムスケッチ（動的表現）とその合成

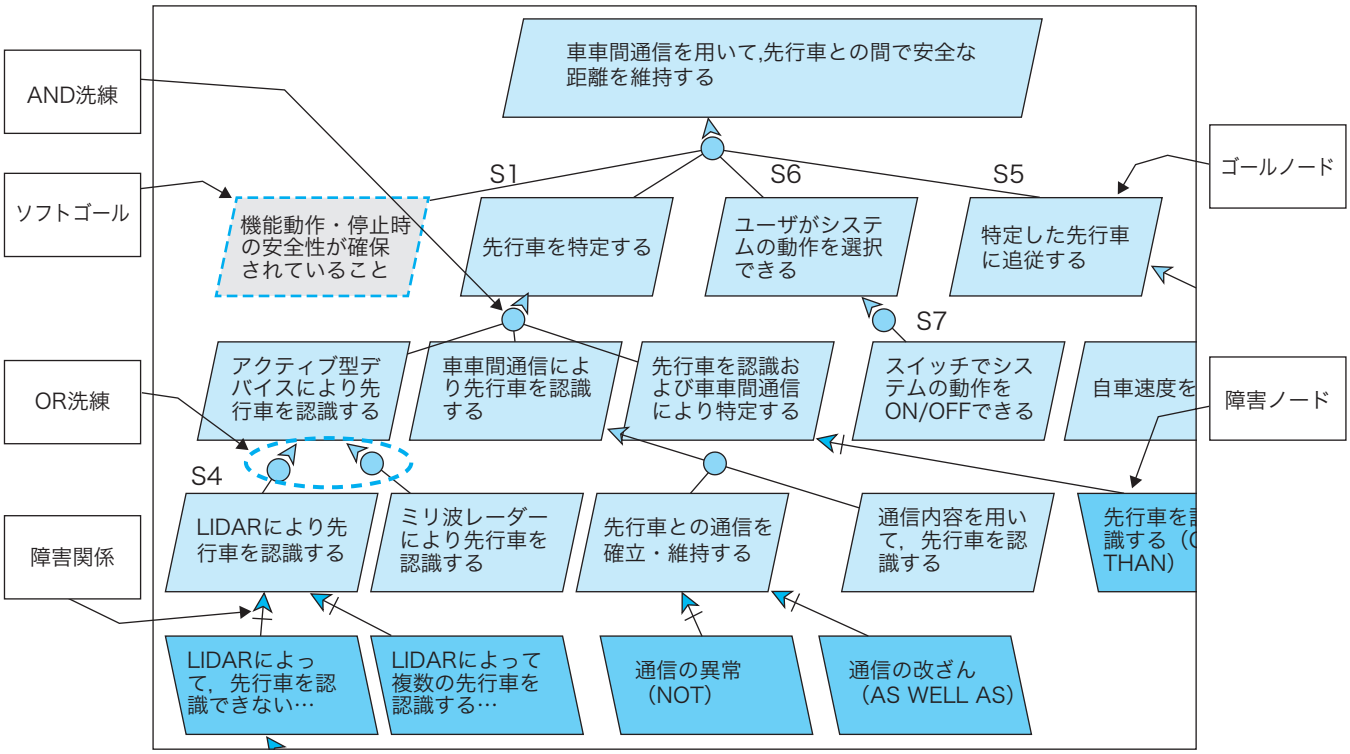


図 3　CACC のゴールモデルの例（本論に必要な範囲で単純化している，S1，S4，S5，S6，S7 は本文中のゴール記述文に対応している．但し，S1 は，一部変更している）

上位ゴールとなる．後者（OR 洗練）は，下位ゴールが上位ゴールの選択肢であることを示している．

CACC におけるゴールモデルの例を図 3 に示す．

次項からは，ゴールの詳細化に伴って，アイテムスケッチを組み合わせる二つの方法を示す．一つは，「洗練」であり（3.2.1 項），もう一つが「合成」である（3.2.2 項）．これらは，組み合わせから生じるハザード・脅威を識別するために用いる．

3.2.1 洗練

上位のゴールを，AND 洗練を利用し，複数のゴールに分割する．ゴールモデルをグラフとして見た場合，これは，下方向に枝を延ばすことに相当する．この時，ゴールと紐付いているアイテムスケッチをどう記述するが，本項の主題である．

次の例を考える．

(S4) LIDAR により，先行車を認識する

ここでは，アクティブ型デバイス，LIDAR[4] で先行車を認識することを示している．このアイテムスケッチは，図 1 右 (d) のモデルとなる．

図 3 に示すように，S1 と S4 の関係は，洗練関係にあり，基本的な構造を変えることなくアイテムスケッチを作成できる [5]．

3.2.2 合成

ここでの「合成」とは，直接の洗練関係にはないゴール同士を組み合わせることである．

アイテムスケッチの動的表現における例を示す．

次のゴール記述文について考える．

(S5) 特定した先行車に追従する

(S6) ユーザがシステムの動作を選択できる

(S7) スイッチで，システムの動作を ON/OFF できる

S5 は図 2 の左上（FSM_A）に対応し，S7 は図 2 の左下（FSM_B）に対応している．S6 は S7 の上位のゴールであるが，ここでは S5 と S7 で考える．なお，ゴールモデル上のそれぞれの位置は，図 3 参照のこと．ここまでで，2 つのアイテムスケッチの動的表現（FSM_A および FSM_B）を得た．この動的表現に対して合成を行う．前述のように「合成」とは，直接の洗練関係にはないゴール同士（即ち，異なる枝にいるゴール）を組み合わせることである．

なお，この場合，待機中・追従中は，あくまで ON 状態における状態遷移である．従って，ON 状態中の状態遷移と

【脚注】

4　LIDAR とは, Laser Imaging Detection and Ranging の略. 光を用いて，対象検知と測距を行う．前方の車両の認識技術としては，この他にミリ波レーダ・（赤外線）カメラなど様々なものがある．本論の例では，LIDAR をアクティブ型（認識用）デバイスとしている．認識技術が異なる場合は，バリアントとして，ゴールモデルを含む記述の再利用が可能である．例えば, 図 3 では, ミリ波レーダを OR 結合し, LIDAR とミリ波レーダは，代替の関係であることを示している．

5　図 3 のゴール木上では，S1 に相当するゴール記述は，「先行車を特定する」と述部の名称が異なっている．CACC では LIDAR 以外に通信を使用するためである．S1 を読み替えて頂ければ幸いである．

して合成する．先の「洗練」とは異なり，「合成」の場合は，（その記載が要求に含まれない場合）意味を考え組み合わせることになる．

3.2.3 障害ノード

障害ノードは，前述のようにゴールモデル中で用いるノードの一つである．ゴール達成の障害となる要素を表現する．例えば，「高い加速性能を有する」というゴールに対して，「運転者によっては，大きな加速度に不快感を持つ」は，障害ノードとなる．

この障害ノードを，本論では，ハザード・脅威の識別のために使用する．ハザード・脅威もまた，ゴールの達成を妨げるからである．次節では，ガイドワードを用いて，この障害ノードを如何に見つけるかを記述する．

3.3 ガイドワードの適用

各ゴールの記述文に対して HAZOP（Hazard And Operability Study）[12-15] のガイドワードを適用する．ハザードと脅威を見つけるための準備である．HAZOP ガイドワードは，2 つのタイプに分かれる．一つは，空間に関するもの．もう一つは，時間に関するものである（表 1 の次元欄を参照）．

表 1　HAZOP ガイドワード

ガイドワード	意味	次元
NO or NOT	否定	空間
MORE	量的な増加	
LESS	量的な減少	
AS WELL AS	質的な変化／増加	
PART OF	質的な変化／減少	
REVERSE	論理的に逆	
OTHER THAN	完全な代替	
EARLY	（時間的に）早い	時間
LATE	（時間的に）遅い	
BEFORE	（順番の）前	
AFTER	（順番の）後	

ガイドワードを用いた分析により，What-if（仮定）分析を，系統立てて行うことができる [16]．他の開発初期に用いられる手法とは異なり，チェックリストに依らず，対象を時空間上で網羅的に調べることができる．

今，ガイドワードを用いて，先のゴール S4 の変形を行う（日本語は，必要に応じて助詞等を変えている）．

（S4-NOT） LIDAR によって，先行車を認識できない

上記は，NOT ガイドワードを適用した場合である．ガイドワードによって，様々な吟味すべき文を作成することができる．いくつか例を挙げる．

（S4-MORE） LIDAR により，複数の先行車を認識する

（S4-ASWELLAS）LIDAR によって，誤って先行車を認識する

（S4-LATE） LIDAR による，先行車を認識したとの通知が遅れる

（S4-NOT_LATE）LIDAR による，先行車の認識不能との通知が遅れる

最後は，2 つのガイドワードを組み合わせた例である．

このようにゴール記述（S）に対して，ガイドワードを利用して，新しい文（S-*）を得ることができる．アスタリスクには，各ガイドワードが含まれる．即ち，一つの記述文から，複数の S-* を得ることができる．

次に，このガイドワードから生成した文を用いて，主題となるハザードおよび脅威の識別を行う．

3.4 ハザードおよび脅威の識別

ガイドワードにより生成した文（S-*）からハザードおよび脅威を識別する手順について説明する．

利用するのは，ここまでに既に作成した以下の記述である．

- アイテムスケッチ（3.1 項）
- ゴール記述に対して，ガイドワードを適用した文（S-*）（3.3 項）

「アイテムスケッチ上で操作を行うことで，ガイドワード適用文（S-*）を解釈する」というのが基本的なアイデアである．詳細は，次項で述べるが，アイテム定義上の記述や値を変更することで，何が生じるかを考えるという思考実験を行う．これは，ソフトウェアテストにおいてテストベクタ（入力の組）を見つけるために，境界値や境界値外に着目することに類似している．

最初にハザード識別に関して説明する．次に，脅威の識別に関して示す．

3.4.1 ハザード

単純なハザード識別としては，構成品の非正常動作をハザード候補として挙げる．アイテムを対象としているため，故障モードを単純に定めることは難しい，という点を除けば，FMEA に類似した方法である．例えば，S4-NOT:「LIDAR によって，（認識すべき）先行車を認識できない」では，アイテムスケッチの静的表現中の構成要素の故障モード（例えば，永続的な故障・一時的な故障，天候による検知不能）によりハザード候補を見つけることができる．

より複雑なハザードは，アイテムスケッチ中の情報（例えば，多重度）を操作することによって，見つけることができる．

例を挙げる．先行車は自車から見た場合，ゼロないしは 1 の多重度を持つ（図 1(b)）．いま，2 台以上の近接し同一速度で走行する先行車があったときに，（アイテムスケッチ上の記述ではゼロあるいは 1 なので）先行車を認識できない場合を想定できる．このように，存在する記述を削除する，或いは数を増やしてみることで，何が生じるかの思考

実験を行うことを，ここでは「アイテムスケッチを操作する」と呼んでいる．

別の解釈もできる．前述のケースで，システムは，（先行車はゼロ或いは 1 なので）誤って一台と見なすかもしれない．この場合は，S4-ASWELLAS（LIDAR によって，誤って先行車を認識する）と関係する．

ともに，静的なアイテムスケッチの多重度を「もし，先行車の多重度が 2 だったら？」と考え，各ガイドワード適用文（S-*）と関連づけることで，ハザード候補を得ることができる．

ゴールにガイドワードを適用した S-* から，どういう場合に問題が生じるかを，直ちに推定できるわけではない．上記に示したように，アイテムスケッチ上の情報の操作を通じて，その推定を行うことになる．

時間次元のガイドワードの例としては，S4-NOT_LATE がある．時間次元ガイドワードは動的表現を利用する．ここでは，図 2 を，参照のこと．先行車を見失った（ロスト）場合は，待機中に遷移する必要がある．しかし，先行車のロストを遅れて通知された場合，待機中に遷移すべきであるにも拘わらず，追従中のふるまいを続けることになる．先行車が急減速している場合は，危険な状態となる．この例では，ガイドワード適用文に相当する動的アイテムスケッチの該当箇所を探し，思考実験を行う．ガイドワード適用文のみを利用するより，より明確に危険な状況を理解することができる．

なお，LATE を適用したもう一つのガイドワード適用文 S4-LATE は，（使用性に影響を及ぼす可能性はあるが）安全性には影響しない．全てのガイドワード適用文が，ハザードに結びつくわけではないことに，注意が必要である．それでも，網羅性の観点から，問題がないという記録は必要である．

3.4.2 脅威

基本的な方法は，ハザードの場合と同等である．ゴール記述に対して，ガイドワードを適用することで，脅威の候補を見つけることができる．

但し，次の点で違いがある．ハザードは，アイテム（最終的にはシステム）の不十分な設計や，アイテムの要素であるエレメントの故障に由来する．一方で，脅威は，悪意ある攻撃者によるアセット（資産）への攻撃である．従って，ガイドワードを適用した場合の解釈を，変更する必要がある．例えば，S4-NOT は，「システムは第三者の攻撃によって先行車を認識できない」と解釈する．

ガイドワードを含んだ文（S-*）から脅威を見つけ出すためには，2 つの場所に着目する必要がある．最初は，脅威から保護されるべきアセットの場所であり，2 つ目は，他のアイテムとの相互コミュニケーションの場所である．

図 4 は，より詳細化したアイテムスケッチの版である．自車に付属する 3 つのクラスは，アセットを示している．これは，前述の着目場所のうち，前者（脅威から保護されるべきアセット）である．後者の相互コミュニケーションについては，通信デバイスを挙げることができる．これは車車間通信を行うもので，アイテム外（即ち先行車）との通信を行う．

ちなみに，更に詳細化を行うことで，異なる候補も現れる．アイテム内部のコミュニケーションに対する脅威である．例えば，CACC のメインの処理が動作する ECU と異なる ECU で，LIDAR が動作する時，通信路による内部コミュニケーションが存在する．概念段階の後半部分では，アイテムスケッチの静的表現を進化させた抽象ハードウェアアーキテクチャの記述を行う．内部コミュニケーションもこの段階では，考慮が必要になる．但し，本論のスコープ外となる．

さて，アセットのうち，ここでは，履歴データに着目する．履歴データの役割は，LIDAR と通信による先行車認識において，それぞれ時間軸上のデータ管理を行うこととする．先行車の安定的な認識を行うための仕組みである．即ち，通信対象の先行車が突然過去の値と（物理的にあり得ない）大きく異なる値を返したときは，異常とみなすために使用しているとする．履歴データが改ざんされると，存在しない先行車を存在すると見なす，或いは，逆に存在している先行車を存在しないと判断するかもしれない．従って，このアセットへの攻撃は，セキュリティばかりか安全性にも影響を及ぼす．

脅威の識別に関してまとめる．基本的な識別法は，ハザードと同様にゴールに対してガイドワードを適用することにより可能である．加えて，脅威とはアセットへの攻撃であり，アセットを識別することが重要である．これには，アイテムスケッチを利用する．アセットを意識することで，適切にガイドワードを含んだ文（S-*）から脅威を識別することができる．

3.5 SSM

ここでは，環境条件を定義するために考案した SSM（Situation-Scenario Mapping）について簡単に示す．車両のような移動体で，かつ車両レベルでハザード・セキュリティを考える場合，どのような環境の組み合わせがあり得るかを考えることは重要である．そのため SSM を記述する．（SSM はアイテムと相互作用するが，アイテムの一部ではないため，図 4 では，独立したクラスとして示している．具体的な環境からアイテムが受ける影響の例としては，LIDAR に対する天候・視程の影響がある）．

一般に，組み込みシステムの制御を考える場合，制御器（コントローラ）と制御対象（プラント）を用いてモデル化することが多い．しかし，環境と相互作用するシステムでは，環境もまた考慮すべき対象である．例えば，定速走行したいが，道路に勾配がある，或いは，天候によって路面の摩擦抵抗が変化しているときは，制御も影響を受ける．ここでは，環境に対して，状況を示す要素を選択的に与え，そ

の組み合わせをシナリオとした表を用いる．例えば，「＜晴天＞で＜高速道路＞を＜一定速度＞で走行する」というシナリオの場合，＜＞で示されるのが状況要素になる．

参考として，Appendix に，SSM の例を示す．

ISO 26262 が要求するリスク評定において，アイテムに ASIL を付与するときにも，シナリオは必要な情報であり，車両レベルで考える必要がある．ASIL は，ある環境中の車両に，故障が発生する発生する度合い・ドライバの対応可能性・事故が発生したときのドライバへの身体的影響から算出する．そのためには，具体的なシナリオが必要である．シナリオ毎に ASIL を計算し，想定可能な複数のシナリオからもっとも厳しい ASIL を付加することになる [17].

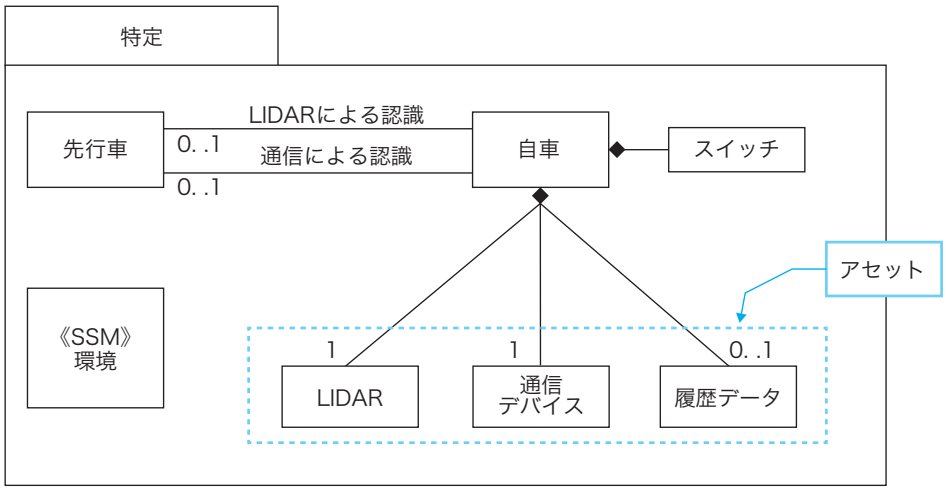


図４　詳細化したパッケージ「特定」とアセット

従って，ハザード識別に限らず，SSM は有効に利用できる．

システムは様々な環境で能動的に動作するため，最初の SSM 作成の負荷は大きい．しかし，一度作成すれば，同種のアイテムにおいては，再利用可能である．例えば，CACC 用に作成した SSM は ACC でも使用することができる．

なお，環境モデルである SSM および，今後ドライバ支援システムの増加とともに，必要性が増しているドライバモデルの作成手法については，別途詳細に記述したいと考えている．

3.6 規格との対比

ISO 26262 との対比を，表 2 に示す．本論で対象とするのは，アイテム定義，安全ライフサイクルの開始，ハザード分析とリスクアセスメントの一部である（3-7.4.2 まで）．表 2 では，この範囲に限定して「要求と推奨」の内容と本手法の関係を示している．

セキュリティに関して，同様の規格は，現在存在していない．しかし，いくつかの提案がなされている．例えば，[18] は，ISO 26262 に対応した脅威分析とリスクアセスメントを提案している．Severity（重大さ）や Controllability（制御可能性）については新たな定義を行っている．プロセスは，ISO 26262 と同一にできるとしている．

4.　関連研究

概念段階において適用可能なハザードおよび脅威の識別手法は，我々の知る限り存在しない．従って，ここでは，範囲を広げ，関連する研究および手法について，記述する．

初期段階の安全性解析手法として，PHA（Process Hazard Analysis），What-If 分析，HAZOP が知られている [16]．HAZOP 以外は，基本的にチェックリスト方式を用いる方法である．チェックリストにより網羅性を担保する．従って，如何にチェックリストを作るかが重要であるが，新しいシステムにおいて，網羅性を担保したチェックリストを作ることは困難である．本論のアプローチは，HAZOP のガイドワードを用い，更に，アイテムスケッチを組み合わせることで，チェックリストに頼ることなく，網羅性を確保している．

STAMP に基づく STPA のアプローチは，コントローラの相互作用に注目する解析手法である [19]．しかし，開始点として機能制御図等を使用するため，概念段階には使用することができない（[19] p. 213 “STPA uses a functional control diagram and the requirements, system hazards, and the safety constraints and safety requirements …”）．[20] では，STPA-sec としてセキュリティ拡張を行ったと主張している．STPA では，環境を陽に扱わないために，自動車のような動作環境が一定ではなく，かつ操作者も多様な場合には適用しづらい．我々の提案手法では，SSM によって環境を定義することができる．

セキュリティに関する総合的な手法としては，CORAS アプローチ [21, 22] がある．CORAS アプローチは，脅威図を中心として，解析を進める．しかし，概念段階における詳細な手順を持たない．

自動車分野に特化した方法としては，欧州の EVITA（E-safety Vehicle Intrusion proTected Applications）がある [23, 24]．EVITA は，特に車両内のコミュニケーションに特化したリスク分析を行う．この手法では，ユースケースからダークサイドシナリオを用いて，アセットを導出する．アセットの候補としては，性能・安全性・プライバシー・アセットがある．このアプローチは，概念段階終了後に，我々の手法と接続して使用することができる．

オリジナルの KAOS 手法を，セキュリティ問題に適用する方法についても報告がある [25]．ここでは，セキュリティゴールメタクラス（秘密性，保全性，可用性等）を特殊化することによって，網羅性を確保している．特殊化するときのパターンは，チェックリストのゴール表現と考えるこ

とができ，本手法と共に（検証のために）用いることも可能である．

5. 結論

本論では，概念段階におけるハザードおよび脅威の識別法について示した．アイテムスケッチとゴールモデルをベースとし，ガイドワードを利用することで，網羅的にハザードおよび脅威を識別することができる．具体的な例とともに手順を説明した．

また，本論に示した方法は，ISO 26262 の概念段階の要求事項と対応づけることができることも示した．

本論は，あくまでハザード・脅威の識別までがスコープである．しかし，本論での記述内容は，以降のフェーズでも利用することができる．例えば，ASIL 決定に必要なシナリオは，すでに SSM を用いて記述してある．また，解決ノードを障害ノードに対置することで，最終的に必要となる安全ゴールを定めることができる．

最後に，次の点を強調したい．本手法は，安全性・セキュリティ確保を，通常の要求分析プロセスと並行して行うことができるということである．本手法は，ゴール指向要求分析とともに用いている．トップゴールは，あくまでアイテムに対する要求である．それはゴールモデルの中で，詳細化され最終的には，要求仕様書になる．その過程で，各ゴールの評価を行うことで，安全性・セキュリティに関して担保できる．また，安全性とセキュリティが，必ずしも独立しているわけではないことは，すでに記した通りである．つまり，車両システムへの悪意ある侵入によって，安全性が脅かされる場合も考えられる．このとき，安全性とセキュリティを別々に解析していると見逃す可能性もある．

従って，要求分析を実施しながら，安全性およびセキュリティ確保のためのハザード・脅威の識別を同時に実施できることは，本手法の特徴であり，かつ最大の貢献と考えている．

謝辞

本論文および本論文の内容を口頭発表した第 11 回クリティカルソフトウェアワークショップでの匿名の査読者の方々および，欧州の ECQA（European Certification and Qualification Association）の機能安全コースにおいて，有意義なコメントをくださった各国の参加者のみなさんに感謝致します．

表 2　ISO 26262 との対比（概念段階のうち 3-7.4.2 までが本論文の対象）

ISO 26262（要求と推奨）				本論でのアプローチ
3-5	アイテム定義	3-5.4.1	アイテムの機能および非機能要求．アイテムの依存関係とその環境を含むこと	ゴールモデルを用いて，アイテムの機能および非機能要求を示す．また，アイテムスケッチは，このゴールモデルを理解するための静的・動的表現である．SSM は，アイテムが動作する環境を示す
		3-5.4.2	アイテム境界，アイテムインターフェイス，他のアイテムとの相互作用に関する仮定を定義すること	アイテムスケッチの静的表現により，アイテム境界やアイテムインターフェイス・他アイテムとの相互作用を定義することができる
3-6	安全ライフサイクルの開始	3-6.4.1	開発カテゴリを決定すること：新規開発か，既存アイテムの変更あるいは，動作する環境の変更か	開発カテゴリが，「既存アイテム」変更の場合は，変更が必要な抽象度レベルから開始することができる（本文 3.1.2 項「開発カテゴリとアイテムスケッチ」参照）
		3-6.4.2	開発カテゴリが修正の場合，影響分析および可能な修正ライフサイクルを決定すること	ゴールモデルでは，洗練関係が示されているため，容易に影響範囲を見つけることができる．即ち，変更の必要があるゴールに対して AND 或いは OR 洗練関係にある下位ゴールは，全て影響の有無を確認すべき候補である
3-7	ハザード分析とリスクアセスメント	3-7.4.1	アイテム定義に基づき，ハザード分析とリスクアセスメントを開始すること	アイテムスケッチとゴールモデルは，開始のために必要な材料となる
		3-7.4.2	状況分析とハザード識別を行う．動作状況とハザードの組み合わせから，ハザードイベントを決定すること	本論の主題である．状況分析については，本文 3.5 節が対応する

APPENDIX

SSM の例を示す．本文で例に使用した CACC における SSM となる．車載のシステム，例えば ACC は，この SSM を使用できる（状況のうち，「電波」が不要になる）．一方で，同じ車載のシステムである駐車支援システムでは，考慮すべき他の状況要素がある（例えば，有料駐車場におけるロック板の有無）

全く異なるシステム，例えば家庭内で移動可能なロボットにおいては，状況要素を改めて検討する必要がある．

しかし，それでもなお，枠組みとしては共通にできる．

[A] SSM

分類 / 時刻 (HH:MI:S)	道路				道路構造物		近傍車輌		道路混雑状況			気象・可視性			非自動車アクタ				法規			電波	タスク
	道路種*	道路状態*	車線数	曲率 (m)	照明	ガードレール他	前方車距離(m)	後方車距離(m)	進行方向	対向	交差*	天候*	気温 (摂氏)	視程 (度)	オートバイ	自転車	歩行者	障害物	区間内信号	速度制限 (KMH)	その他	到達可能距離 (m)	
1010:00	RT_SB	GR(0), GG(0), MU(0.8)	2	-	有	有	30	20	10	8	-	CM_CS	28	8	0	2	0	0	直進・青	M60	なし	200	T_FL
1012:00	↑	↑	↑	-	↑	↑	30	20	↑	↑	↑	↑	↑	↑	↑	1	↑	0	↑	↑	↑	↑	↑
…																							
1030:00	RR_CL	GR(0), GG(0), MU(0.6)	1	-	なし	なし	150	200	1	3	-	CM_CS	28	8	0	0	0	0	-	M40	なし	200	T_DR

*: 別表あり

区間あたりの台数

区間あたりの数

[B] 道路種

道路種	市街地	高速道路	自動車専用道路	幹線道路	バイパス	郊外道路	田舎道
RT_ID	DT	FW	LH	UA	BP	SB	CL

[C] 道路状態

道路状態	勾配	うねり	μ
RS_ID	GR(v)	GG(p)	MU(v)
	v:勾配%	p:パターン e.g. 波状路	v:摩擦係数

[D] 交差

交差する道路	交差	食い違い交差点	環状交差点	立体交差	合流
CF_ID	CF(iv)	SJ(iv)	RA(iv)	GS	JC
	iv:交差数	iv:交差数	iv:交差数		

[E] 天候

天候	晴れ	曇り	雨	雪	霧	霧	霧
CM_ID	CS	CL	RN(v)	SN(v)	HL	FG	HZ
			v: 降水量	v: 降雪量			

[F] タスク

タスク	タスク
T_SR	発車
T_DR	一定速走行
T_FL	先行車追従
T_CL	レーン変更
T_PS	追い越し
T_TL	左折する（交差点）
T_CL	左に曲がる
T_TR	右折する（交差点）
T_CR	右に曲がる
T_UG	急停止する
T_ST	（一時的な）停止する
T_PO	路肩にとめる
T_PK	駐車する
T_OB	障害物を避ける
・・・	・・・

図 5　CACC における SSM の例

[A] が SSM の本体になる. 他の表([B]～[F])は, [A] を記載するために必要な定義を行っている表である. 各カラムは環境を構成する要素(状況要素)になる. 一つのレコードが，ある時刻における状況を示している．時刻は，秒単位で記載している．この状況の組の時系列変化は，一つのシナリオとなる．即ち，一つの SSM の表が，一つのシナリオとなる.

【参考文献】

[1] Thalen, J., ADAS for the Car of the Future. 2006.

[2] Gubbi, J., et al., Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 2013. 29(7): p. 1645-1660.

[3] Delgrossi, L. and T. Zhang, Vehicle safety communications : protocols, security, and privacy. Information and communication technology series. 2012: Wiley. xxvi, 372 pages.

[4] ISO, ISO 26262. Road vehicles - Functional safety -, 2011, ISO.

[5] Ericson II, C.A., Hazard analysis Techniques for System Safety. 2005: John Wiley & Sons, Inc.

[6] Francillon, A., et al. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. in NDSS. 2011.

[7] OMG, OMG Systems Modeling Language (OMG SysML) V1.1, formal/2008-11-01, 2008, OMG.

[8] D'Souza, D.F. and A.C. Wills, Objects, Components, and Frameworks with UML: The Catalysis Approach. 1998: Addison-Wesley Professional.

[9] Naus, G., et al. Cooperative adaptive cruise control. in IEEE automotive engineering symposium Eindhoven, The Netherlands. 2009.

[10] SAEInternational, Adaptive Cruise Control (ACC) Operating Characteristics and User Interface (J2399), 2003, SAEInternational.

[11] van Lamsweerde, A., Requirements engineering: from system goals to UML models to software specifications. 2009: John Wiley & Sons Ltd.

[12] CEI/IEC, Hazard and operability studies (HAZOP studies) - Application guide, CEI/IEC 61882:2001, 2001, IEC.

[13] Fenelon, P. and B. Hebbron. Applying HAZOP to software engineering models. 1994. Citeseer.

[14] Nolan, D.P. and Knovel (Firm). Application of HAZOP and What-If safety reviews to the petroleum, petrochemical and chemical industries. 1994; viii, 128 p.].

[15] Redmill, F., M. Chudleigh, and J. Catmur, System Safety: HAZOP and Software HAZOP. 1999: John Wiley & Sons, Inc.

[16] Nolan, D.P., Safety and security review for the process industries : application of HAZOP, PHA, What-If and SVA reviews. 3rd ed. 2011, Oxford: Elsevier/GPP.

[17] Czerny, B.J., R. Suchala, and M. Runyon, A Scenario-Based Approach to Assess Exposure for ASIL Determination, 2014, SAE Technical Paper.

[18] Ward, D., I. Ibarra, and A. Ruddle, Threat Analysis and Risk Assessment in Automotive Cyber Security. SAE International Journal of Passenger Cars-Electronic and Electrical Systems, 2013. 6(2): p. 507-513.

[19] Leveson, N., Engineering a safer world: Systems thinking applied to safety. 2011: MIT Press.

[20] Young, W. and N.G. Leveson, An integrated approach to safety and security based on systems theory. Commun. ACM, 2014. 57(2): p. 31-35.

[21] Brændeland, G., et al., Using dependent CORAS diagrams to analyse mutual dependency, in Critical Information Infrastructures Security. 2008, Springer. p. 135-148.

[22] Lund, M.S., B. Solhaug, and K. Stølen, Model-driven risk analysis: the CORAS approach. 2011: Springer.

[23] Henniger, O., et al. Securing vehicular on-board it systems: The evita project. in VDI/VW Automotive Security Conference. 2009.

[24] Ruddle, A., et al., Security requirements for automotive on-board networks based on dark-side scenarios. EVITA Deliverable D2.3, EVITA project, 2009.

[25] van Lamsweerde, A. Elaborating security requirements by construction of intentional anti-models. in Proceedings of the 26th International Conference on Software Engineering. 2004. IEEE Computer Society.

次世代ソフトウェア信頼性評価技術の開発に向けて

土肥　正[†]　　岡村　寛之[†]

本研究では次世代のソフトウェア信頼性評価技術として，ソフトウェアメトリクスを利用した評価モデルの構築を行う．具体的には，従来のソフトウェア信頼度成長モデルに対して，メトリクスを説明変数とした回帰構造を取り入れる．また，信頼性尺度に強く寄与するメトリクスの分析ツールの開発を行い，数値実験を通じてその有効性を検証する．

Towards development of the next-generation software reliability assessment technology

Tadashi Dohi[†], Hiroyuki Okamura[†]

This paper proposes a next-generation framework on software reliability assessment by incorporating software metrics as explanatory variables. More specifically, we apply the regression-based approaches to the existing software reliability growth models, and develop software reliability assessment tools to evaluate the software reliability quantitatively and to identify the significant metrics which strongly affect some reliability measures. It is shown through numerical experiments that our methods are quite effective and useful.

1. はじめに

ソフトウェアの高信頼化は近年の大きな課題である．一般的に，ソフトウェアの信頼性確保には，(i) 開発プロセスの管理技術の向上，(ii) 開発技術自体の向上，が大きく起因している．それと同時に，プロセスと製品の信頼性を定量的に評価し，プロジェクト管理技術および開発手法にフィードバックする取組は，多様化するソフトウェアに対して高信頼性を保証するための重要な技術である．70 年代初頭から研究が開始されたソフトウェア信頼度成長モデル（あるいはバグモデルとも呼ばれる）は，ソフトウェアテストで発見されるフォールトの計数過程を統計的に推論することで，発見フォールト数の飽和状態を監視し，ソフトウェア信頼度などの定量的評価尺度を算出するために用いられてきた．このようなフォールトの計数データだけを用いたソフトウェア信頼度成長モデルは，取扱いが非常に簡便であるため，現在においても尚実務において利用されている．しかしながら，従来のモデルでは，テスト労力やソフトウェア種別などの情報を明示的に利用せず，フォールト計数データのみに依拠しているため，それらの要因と定量化された信頼性の因果関係がブラックボックス化している．そのため「得られた数値の妥当性検証が難しい」ことや「管

【脚注】
† 広島大学　Hiroshima University

理技術や開発手法への明示的なフィードバックが難しい」ことが欠陥として指摘されている．

従来のソフトウェア信頼性技術の啓蒙活動が必ずしも産業界において成功していない要因の一つは，「信頼度を左右する要因を明らかにしたい」と言う開発現場における要求を念頭においたモデル開発が行われていなかった点にある．これは，モデルの抽象度が非常に高いため，信頼度を左右する要因を根本的に特定することが不可能である点に起因する．ソフトウェア工学分野では，モデルのあてはめによるソフトウェア信頼性評価技術は役に立たない技術として捉えられている向きがあり，現在主流となっている各種メトリクス計測に基づいた実証的ソフトウェア工学と独立した領域と見なされることが多い．

本研究では，従来のソフトウェア信頼度成長モデルの利点を活かしながら，より詳細なプロジェクトデータが扱えるモデルへの拡張を試みる．開発現場で解決すべき課題の一つは「どのような設計・テストをしたらフォールトのないソフトウェアが開発できるか」であり，その第一歩として，設計段階やテスト段階において計測されるメトリクスを有効に活用できるソフトウェア信頼度成長モデルを開発し，モデルから得られるソフトウェアの信頼性尺度が統計的にどのようなメトリクスに起因するかについて分析する．なお，本研究は独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（SEC: Software Reliability Enhancement Center）が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援のもと行われた「次世代信頼性評価技術の開発とその実装」[1] における成果の一部をまとめたものである．

2. ソフトウェア信頼度成長モデル

本研究では，従来のモデルにおいてどのようにしてメトリクス情報を取り扱うかを議論し，その統計的な分析手法を提示する．そのため，ここでは基礎となるソフトウェア信頼度成長モデルの数理的な構造について述べる．ソフトウェア信頼度成長モデルはこれまでに様々なものが提案されているが，ここでは非同次ポアソン過程（NHPP: Non-Homogeneous Poisson Process）に基づいたモデルを包括的に扱う枠組みを紹介する．NHPP によるソフトウェアフォールト検出過程のモデリングの起源は Goel and Okumoto(1979)[6] であり，以来，過去 35 年間に渡って数多くのモデルが提案されてきた．NHPP モデルでは，時刻 t までに発見されたバグ数の累積数を $N(t)$ とすると，$N(t)$ を以下の確率関数で定義する．

$$P(N(t)=k)=\frac{H(t)^k}{k!}e^{-H(t)}. \tag{1}$$

上述のポアソン型の確率関数で表現される確率過程は一般にポアソン過程と呼ばれ，さらに $H(t)$ が時刻に依存する（非定常である）ため，非同次ポアソン過程（NHPP）と呼ばれる．ここで $H(t)$ は時刻 t までに発見される累積バグ数の「期待値」を表し，平均値関数と呼ばれる．

これまでに数多くの NHPP によるソフトウェア信頼度成長モデルが提案されているが，本質的な違いは平均値関数 $H(t)$ にどのような関数を仮定するかという点に集約される．一方，Langberg and Singpurwalla(1985)[8] は，以下の仮定に基づいたモデル化により，ほとんどすべての NHPP モデルが包括的に表現できることを示した．

仮定 A: テスト実施前にプログラム中に含まれる総バグ数は有限であり，平均 a のポアソン分布に従う．

仮定 B: それぞれのバグは互いに独立に発見される．

仮定 C: 各バグの発見時刻は確率分布関数 $F(t)$ をもつ非負の確率変数によって表される．

このとき，時刻 t までに発見される累積バグ数の確率関数は以下のようになる．

$$P(N(t)=k)=\frac{(aF(t))^k}{k!}e^{-aF(t)}. \tag{2}$$

これは平均値関数が $\mathrm{E}[N(t)]=aF(t)$ の形をもつ NHPP に一致する．$F(t)$ に 0 から 1 まで単調に増加する確率分布関数を代入することで，上述の枠組みは初期残存バグ数が有限個の NHPP モデルを任意に表現することができる．さらに，テスト時間が $(0,t_1],(t_1,t_2],\ldots,(t_{K-1},t_K]$ のように離散的に与えられる場合であっても，あるバグが $n-1$ 番目のテスト日までに発見されない条件のもとで n 番目のテストで発見される条件付き確率（バグ発見確率）を $p_n=(F(t_n)-F(t_{n-1}))/(1-F(t_{n-1}))$ とすると，

$$P(N_n=k)=\frac{(aq_n)^k}{k!}e^{-aq_n} \tag{3}$$

のように表現することもできる．ここで $q_n=1-\Pi_{k=1}^{n}(1-p_k)$ である．

上述の仮定に従えば，ソフトウェア信頼度成長モデルは次の二つの要因によって決定されることが自明である．

(1) ソフトウェアバグ総数の平均値 (a)

(2) 単一のバグ発見時刻に関する確率分布 ($F(t)$)

上述の仮定 (1) と (2) は，それぞれ，テスト前にプログラム全体に残存する総バグ数とバグの見つかり具合を表現するパラメータが存在すると言い換えることができ，事実「バグの見つかり具合」の違いで様々な NHPP モデルがこれまでに提案されてきた．表 1 に代表的な NHPP モデルと仮定されるバグ発見確率の対応関係を示す．

表 1　代表的な NHPP モデルとバグ発見確率の対応.

モデル	バグ発見確率分布	文献（既存モデル名）
Exponential	指数分布	指数形 SRGM，Goel and Okumoto model [6]
Gamma	ガンマ分布	遅延 S 字形 SRGM [13,14]
Pareto	第二種パレート分布	修正 Duane model [2,9]
Truncated Normal	切断正規分布	[12]
Log-Normal	対数正規分布	[3]
Truncated Logistic	切断ロジスティック分布	習熟 S 字形 SRGM [10]
Log-Logistic	対数ロジスティック分布	Log-Logistic model [7]
Truncated Extreme-Value Max	切断 Gumbel 分布	修正 Gompertz model [11, 15]
Log Extreme-Value Max	Frechet 型極値分布	[11]
Truncated Extreme-Value Min	Gompertz 分布（最小値）	[11]
Log Extreme-Value Min	Weibull 分布	Goel model [5,11]

3. メトリクス情報を利用したソフトウェア信頼度成長モデル

3.1. 一般化線形ソフトウェア信頼度成長モデル

先に示したように，従来のソフトウェア信頼度成長モデルは，総バグ数とバグ発見確率の二つの要因から構成され，これらはソフトウェアテストにおいて観測された発見バグ数から統計的に直接推定される．しかしながら，総バグ数やバグ発見確率はソフトウェアの規模や種類などソフトウェア固有の特徴量に強く関連しており，そのような情報を含んだメトリクス情報を活用することで，信頼度推定の精度向上や，バグ総数などに寄与する要因分析を行うことが可能になる．

ソフトウェアメトリクスは，コード行数やソースの複雑度といった開発するソフトウェアに関するプロダクトメトリクスと，開発プロセスに関連したプロセスメトリクスに大きく分類することができる．ソフトウェア信頼性評価においては，ソフトウェアそのものの特徴量を表すプロダクトメトリクスと，ソフトウェアテストにおいて観測され得るプロセスメトリクスが特に重要であると考えられる．一般に，これら二つのメトリクスはそれぞれバグに与える影響が異なるため，その特徴を十分に考慮した分析手法を提案する必要がある．

いま，j 個のモジュールからなるソフトウェアを考え，以下の仮定を設ける．

- ソフトウェアテストは逐次的かつ離散時刻上で実施され，各テストで発見されたバグは次のテストまでに修正される．
- モジュール i において，一つのバグが k 番目のテストで発見されるバグ発見確率 $p_{i,k}(0 \le p_{i,k} \le 1)$ を，モジュール i の k 番目のテストに関連したプロセスメトリクス $\boldsymbol{x}_{i,k}$ を用いて

$$\mathrm{logit}(p_{i,k}) = a_{0,i} + \boldsymbol{a}_i \boldsymbol{x}_{i,k} \tag{4}$$

のように表す．ここで logit はロジット関数であり，

$$\mathrm{logit}(p) = \log \frac{p}{1-p} \tag{5}$$

で与えられる．また，$a_{0,i}$ および $\boldsymbol{a}_i$ は回帰係数と呼ばれ，プロセスメトリクスがバグ発見確率に与える影響度を表す．

- 各モジュール $i = 1,\ldots,j$ に含まれる総バグ数 $M_1,\ldots,M_j$ はモジュール固有のプロダクトメトリクス $\boldsymbol{s}_i$ に依存した平均 v_i のポアソン分布に従い，その平均値は以下で与えられる．

$$\log(v_i) = b_0 + \boldsymbol{b}\boldsymbol{s}_i. \tag{6}$$

ここで，b_0, $\boldsymbol{b}$ はプロダクトメトリクスが総バグ数に与える影響度を表す回帰係数である．

上記の仮定のもとで，モジュール i の総バグ数が平均 $\exp(v_i)$ のポアソン分布に従うため，モジュール i の k 番目のテストまでに発見されるバグ数 $Y_{i,k}$ は以下の確率関数で与えられる．

$$\begin{aligned} &P(Y_{i,k} = y_{i,k}) \\ &= \sum_{m=y_{i,k}}^{\infty} P(Y_{i,k} = y_{i,k} | M_i = m) P(M_i = m) \\ &= \exp(-v_i q_{i,k}) \frac{(v_i q_{i,k})^{y_{i,k}}}{y_{i,k}!}. \end{aligned} \tag{7}$$

ここで，$q_{i,n} = 1 - \Pi_{k=1}^{n}(1 - p_{i,k})$ である．このモデルでは，従来のソフトウェア信頼度成長モデルに対して，総バグ数に対するポアソン回帰と k 番目のテストにおけるバグ発見確率に対するロジスティック回帰を同時に適用していることに注目すべきである．本稿では上述のモデルを一般化線形ソフトウェア信頼度成長モデルと呼ぶ．この

モデルでは各モジュールの各テスト期間で発見されたバグ数，プロセスメトリクス，プロダクトメトリクスに関するデータから回帰係数 $a_{0,i}$,$\boldsymbol{a}_i$,b_0,$\boldsymbol{b}$ の推定を行う．また，上の仮定におけるロジット関数や対数関数はメトリクスからの影響度（線形の関数で仮定されている）から，実際の総バグ数やバグ発見確率への変換を規定しており，リンク関数と呼ばれる．これらは変更することが可能であり，プロビット関数や恒等関数を適用することもできる．

3.2. カーネル法の適用

一般化線形モデルに基づいたモデル化は，基本的にメトリクスとバグ数あるいはバグ発見確率に与える影響度が線形であることを仮定している．しかしながら，実際問題として，これらが線形の関係にあるかどうかを事前に判断するのは難しい．一方，適切なメトリクスを選択する必要があるため，任意の非線形構造を仮定することは有効ではない．これらの問題を解決する試みとして，ここではカーネル法の適用を考える．カーネル法とは，パターン認識などでよく用いられる手法の一つであり，線形回帰やサポートベクターマシンなどと組み合わせることで，非線形で表される因果関係を記述することができる．原理的には，データを高次元の特徴空間へ写像し，その高次元の特徴空間でデータの分析を行うものである．もとの空間において非線形な関係も，高次元空間では線形の関係で表せることが多く，線形回帰などの線形モデルを適用してより効率よく分析を行うことができる．また，高次元空間での座標を決める代わりにデータ間の内積にカーネル関数を用いることで，高次元化に伴う計算量の増大を抑えることができる．このため，文字列やグラフなどの構造データに対して，カーネル関数による内積を定義することで，様々な特徴量を分析できることも大きな利点となっている．

一般にカーネル法では，もとの線形モデルにカーネル関数を直接適用する．いま，一般化線形ソフトウェア信頼度成長モデルに対してカーネル法を適用すると，先に示した関係式を次のような関数で置き換えることができる

$$\mathrm{logit}(p_{i,k}) = \sum_{u=1}^{k} c_u \mathcal{K}(\boldsymbol{x}_{i,u}, \boldsymbol{x}_{i,k}) \tag{8}$$

$$\log(v_i) = \sum_{m=1}^{j} h_m \mathcal{K}(\boldsymbol{s}_i, \boldsymbol{s}_m). \tag{9}$$

ここで，c_u, h_m は回帰係数，$\mathcal{K}(x, y)$ はカーネル関数であり，x と y の類似度（内積）を表す関数である（付録参照）．カーネル関数は類似度の性質を満たしていればどのような関数でも適用することができるため，通常，数値ベクトルに対してはガウスカーネルや多項式カーネルがよく用いられる．また，文字列やグラフなどの構造データを扱えるカーネル関数を導入することで，ソースコードやその他の設計ドキュメントからメトリクスを抽出することなく，モデルを構築することも可能である．例えば，二つの文字列 x, y の何らかの距離 $D(x, y)$ が定義できる場合，これを用いたカーネルとして以下のようなものが構成できる．

$$\mathcal{K}(x, y) = \exp\left(-\frac{D(x, y)}{\sigma}\right). \tag{10}$$

本論文における数値例ではカーネル法の適用を行っていないが，具体的に，文字列スペクトルカーネル [16] とここで記述したモデルを用いてプログラムソースコードから直接，総バグ数を推定する試みを行っている．詳細は [1] を参照のこと．

4. 実験による検証

4.1. 検証ツール

一般化線形ソフトウェア信頼度成長モデルを利用する標準的な手順は次のようになる．

(1) データの収集：テスト中盤以降において，これまでのバグ発見数，テストに関するプロセスメトリクス（工数，カバレッジなど），各モジュール単位のプロダクトメトリクス（コード行数, 複雑度など）を収集する．

(2) モデルパラメータの推定：収集したバグ数に関するデータとメトリクスデータを用いて，一般化線形ソフトウェア信頼度成長モデルのパラメータ推定を行う．推定手法には最尤法あるいは罰則付き最尤法を用いる．また，モデルの適合性尺度をもとにしたメトリクスの取捨選択を行う．

(3) 信頼性評価：推定したモデルパラメータを用いることで，モジュール単位の期待残存バグ数やフォールトフリー確率などの信頼性尺度を算出する．

最終的には，定量的な信頼性尺度をもとにして，開発中のソフトウェアに対してリリースの判定やテスト戦略の変更などのアクションを行う．また，メトリクスと定量的な信頼性尺度の関連が明確になるため，この分析結果を次期以降の類似プロジェクトに反映させることもできる．

本研究では，上記の (2) と (3) を支援するツールの開発を行った[1]．開発したツールは，推定に関する統計的な計算を行う DLL(Dynamic Link Library) をコアとし

【脚注】

1 http://www.rel.hiroshima-u.ac.jp/msrat/

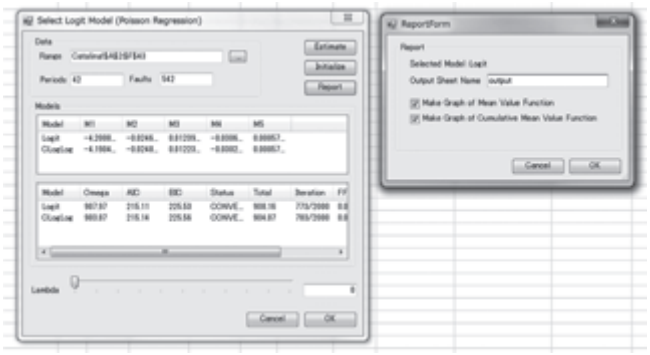


図 1　信頼性評価ツールの画面

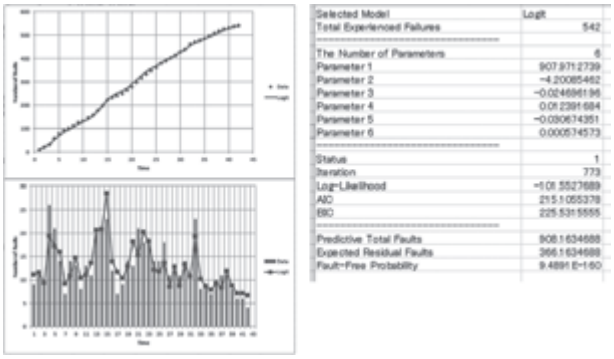


図 2　出力例

て，Microsoft Excel AddIn によるインタフェースおよび統計ツール R からの利用が可能となっている．図 1 に Microsoft Excel AddIn インタフェースの画面を示す．Microsoft Excel ではシート上で管理しているバグ発見数およびメトリクスデータを選択することで容易に推定が行える．また，結果として各種信頼性尺度や推定したモデルによる予測グラフなどの出力を行うことができる（図 2 参照）

4.2. 一般化線形ソフトウェア信頼度成長モデルの検証

オープンソースプロジェクトにおけるデータを対象としたモデルの有効性分析を行う．ここでは，一般化線形ソフトウェア信頼度成長モデルの適合性評価を行う．特に，従来の手法であるメトリクスデータを用いない単純な NHPP モデルに対して，プロダクトメトリクスだけを利用する場合，プロセスメトリクスだけを利用する場合，両方のメトリクスを利用する場合の比較を行う．AIC(Akaike Information Criterion)[4] を用いた適合性評価を行うことで，単純なデータとの誤差比較ではなく，背後にある（真の）モデルを適切に表現できるかどうかの検証を行う．AIC は以下のように定義されるモデル選択基準であり，値が小さいほどデータをよく説明しているモデルであることを示している．

$$\mathrm{AIC} = -2(\text{最大対数尤度} - \text{モデルのパラメータ数}). \tag{11}$$

検証に用いるデータは Apache Software Foundation (ASF) で管理されている Tomcat プロジェクトを対象とした．Tomcat のバージョンは 5, 6, 7 であり，メジャーバージョンアップについては別アプリケーション（別プロジェクト）と見なして分析している．それぞれのバージョンに対してバグトラッキングシステム（ASF Bugzilla）からバグレポートを収集し，バグ数の計測を行った．データの収集期間は Tomcat5 が 2004/5 から 2009/1，Tomcat6 が 2006/1 から 2013/11，Tomcat7 が 2010/6 から 2013/11 となっている．また，モジュール構成は次の通りである．

- webapps (manager): 管理用ウェブアプリケーション
- catalina: Servlet container のコア
- connectors: Tomcat と Apache の連携
- jasper: JSP ページコンパイラとランタイムエンジン
- servlets: Servlet プログラム群

プロダクトメトリクスは上記のモジュールに対応するソースコード群から直接計測した．計測には Source Monitor[2] を用いて，対応するソースコードが格納されているディレクトリ全体に対して以下のメトリクスを算出した．

- Files (Fl): ファイル数
- Lines (Ln): コード行数
- Statements (St): ステートメント数
- % Branches (Br): 分岐の出現率
- Calls (Ca): メソッドの呼び出し回数
- % Comments (Cm): コメントの出現率
- Classes (Cl): クラス数
- Methods/Class (Me/Cl): 単一クラスあたりのメソッド数
- Avg Stmts/Method (St/Me): 単一メソッドあたりのステートメント数
- Max Complexity (MCx): 最大複雑度
- Max Depth (MDp): 最大のネストの深さ
- Avg Depth (ADp): 平均のネストの深さ
- Avg Complexity (ACx): 平均複雑度

一方，プロセスメトリクスに関してはプロジェクトのアクティビティを表す指標として考えられる以下のメトリクスをそれぞれのモジュールに対して計測した．

- time: 日数
- ctime: 累積日数
- comments: Bugzilla 上での該当モジュールに対するコメント数
- votes: Bugzilla 上での該当モジュールに対する投票回数
- wokers: コメントを投稿した人の数（Tomcat5 のみ）

【脚注】

2　http://www.campwoodsw.com/sourcemonitor.html

実験では，まずモジュール毎で独立に (1)NHPP モデルの推定，(2) プロセスメトリクスのみを考慮したモデルの推定を行った．(1) では表 1 に示した代表的な 11 種類のモデルすべてに対してパラメータを推定し，最小の AIC を示すモデルを選択した．また，プロセスメトリクスのみを用いたモデルでは，変数増減法による変数選択を行って最小の AIC を示すメトリクスの組み合わせを求めた．次に，モジュールのプロダクトメトリクスを用いて，(3) プロダクトメトリクスのみを考慮したモデルの推定と (4) 一般化線形ソフトウェア信頼度成長モデルの推定を行った．(3) の推定では，事前に各モジュールに適合した NHPP モデルを選ぶ必要がある．これには，(1) の手順で選ばれたモデルと同じモデルを採用した．同様に (4) では，各モジュールに適用するプロセスメトリクスを選ぶ必要がある．これには (2) で選択されたメトリクスを用いることとした．さらに，(3) と (4) ではプロダクトメトリクスの選択を行う必要がある．これも，変数増減法により最小の AIC を示す組み合わせを選んだ．

表 2 から表 4 は NHPP モデルのみ（NHPP），プロセスメトリクスのみ（Logit），プロダクトメトリクスのみ（Poi），一般化線形ソフトウェア信頼度成長モデル（GLSRM）それぞれを適用したときの AIC の値を示している．表から，いずれの場合でも，AIC の大小関係において

$$\mathrm{NHPP} \geq \mathrm{Poi} \geq \mathrm{Logit} \geq \mathrm{GLSRM} \qquad (12)$$

の関係が得られた．つまり，適合性の観点からは一般化線形ソフトウェア信頼度成長モデルが最も高い適合性を示すことがわかった．また，プロダクトメトリクスのみの場合とプロセスメトリクスのみの場合を比較すると，プロセスメトリクスを利用する方がモデルの適合性が高いことがわかった．これは初期の総バグ数に影響する情報よりも，テスト中に観測された発見バグ数から得られる情報の方が信頼性評価により寄与することを示唆している．この一つの理由として，今回の実験ではモジュール数が比較的少ないため，プロダクトメトリクスから得られる情報量が相対的に少なかったことが考えられる．よりモジュール数が多い状況では，プロダクトメトリクスからの情報も信頼性評価に大きく影響を与えるものと予想される．

また表 5 から表 7 は，選択された NHPP モデル，プロセスメトリクス，プロダクトメトリクスの種類を表している．選択されたメトリクスをみると，プロダクトメトリクスでは最大複雑度（MCx）が比較的多く選ばれており，プロセスメトリクスではコメント数が多く選択される結果となった．これは，プロダクトメトリクスとして従来から重要視されてきたコード行数や複雑度などの因子が信頼性評価にも大きく影響することを意味している．また，コメント数などプロジェクトのアクティビティを直接表す指標の利用は，バグ発見確率をより正確に同定するために大きく寄与することも確認できた．

表 2　Tomcat5 に対する AIC.

Module	NHPP	Logit	Poi	GLSRM
catalina	268.96	271.04	-	-
connectors	145.10	128.20	-	-
jasper	159.96	147.70	-	-
servlets	116.99	102.08	-	-
webapps	150.51	137.69	-	-
Total (project)	841.52	786.71	838.87	784.58

表 3　Tomcat6 に対する AIC.

Module	NHPP	Logit	Poi	GLSRM
catalina	489.30	388.66	-	-
connectors	277.74	201.74	-	-
jasper	293.51	247.27	-	-
manager	163.15	125.58	-	-
servlets	181.54	181.58	-	-
Total (project)	1405.24	1144.83	1403.82	1144.84

表 4　Tomcat7 に対する AIC.

Module	NHPP	Logit	Poi	GLSRM
catalina	247.91	210.87	-	-
connectors	168.82	129.44	-	-
jasper	167.34	137.57	-	-
manager	117.07	90.49	-	-
servlets	148.33	110.57	-	-
Total (project)	849.47	678.94	847.40	675.86

表 5　Tomcat5 において選択されたモデルとメトリクス.

Module	NHPP	Logit	Poi	GLSRM
catalina	Truncated Extreme-Value Max SRGM	comments, ctime	St, MCx	St, MCx
connectors	Log Extreme-Value Max SRGM	worker, ctime		
jasper	Log-Logistic SRGM	worker, ctime		
servlets	Gamma SRGM	comments, ctime		
webapps	Truncated Logistic SRGM	comments, ctime		

5. まとめと今後の課題

実験の結果から，本研究で提案した一般化線形ソフトウェア信頼度成長モデルが従来モデルと比較してかなり高い適合性能力を有することが示された．AIC などの情報量規準はモデルの自由度を考慮したモデル選択基準であり，値が小さいものが最良モデルとなる．よって，扱う統計情報の量が多くなることでモデルの自由度が増えたとしても，メトリクス情報の計測は適合性評価結果に貢献できていることが分かる．また，プロダクトメトリクスとプロセスメトリクスの効果を比較した場合，明らかに後者の方が適合性に関する貢献度が高いことも分かった．コード行数，ファイル数，クラス数などのソフトウェアの規模に関するメトリクスやソースコードの複雑性は，ソフトウェアの信頼性評価に全く影響を与えないわけではないが，少なくともテストの進捗状況に応じて変化するバグ発見の挙動に直接的に効果を与えることは稀である．このことは実証ソフトウェア工学でこれまで提案されてきたプロダクトメトリクス計測よりも，テスト時間やバグフィックスのために費やされた労力（コメント数，メール交換数など）の方がむしろ信頼度成長現象を説明するための要因となっていることを意味している．

一般化線形ソフトウェア信頼度成長モデルの実務的利点として，複数のモジュールを有するシステムのバグ情報並びにメトリクス情報を直接扱うことで，各モジュール単位で信頼性評価を行うことが可能であることが挙げられる．従来まで，モジュールテストにおける信頼性評価では，各々のモジュールに対して対応するバグデータに信頼性評価モデルを独立に適用する方法がとられてきた．このような方法では，回帰構造を導入したとしてもプロダクトメトリクスの情報を矛盾なく信頼性評価に組込むことが困難であった．本研究で提案した一般化線形ソフトウェア信頼度成長モデルは，オープンソースに代表されるようなリポジトリデータベースで開発管理されるソフトウェアに対しても，現実的かつ有効な評価スキームを提供している．

設計情報や開発管理情報がメトリクスの形で集約されているような場合，すなわち，定量的なソフトウェアの開発管理が徹底している現場においては，提案ツールがそのまま利用できる．計測可能でかつ入手可能なメトリクス情報があれば，とりあえずそれを全て利用することでより正確な信頼性評価が行える．しかしながら，メトリクス情報の計測と管理にはコストがかかることも事実である．どのようなメトリクスを収集すべきかについての実証的知見がなければ，本研究で開発された信頼性評価技術を有効に活用することは難しい．その意味で，これまで実証ソフトウェア工学において積み重ねられた研究成果や企業で培われた独自のノウハウをメトリクス選択に生かすことが肝要であると考えられる．一方で，考えられ得る種類のメトリクス情報が獲得できたとしても，適切な選択を行うためには膨大な計算量が必要となる．例えば三つのモジュールそれぞれに対して 2 種類のプロダクトメトリクスと 3 種類のプロセスメトリクスが観測されている場合でも，適切な要因を選択するためには 21952 通りの組み合わせを試す必要がある．このような問題に対処するため，変数選択手法の強化が課題となる．一つの可能性としては，ここで紹介したカーネル法と現在のデータマイニング技術を融合することで，高いスケーラビリティを持ったアルゴリズムを開発することである．一方，信頼性評価技術を業務の一環として恒

表 6　Tomcat6 において選択されたモデルとメトリクス．

Module	NHPP	Logit	Poi	GLSRM
catalina	Truncated Extreme-Value	comments, votes, ctime	Fl,Cl,MCx	Fl,Ln,Cm,Cl
connectors	Log Extreme-Value Max	comments, votes, ctime		
jasper	Truncated Extreme-Value Max	time, comments		
manager	Log Extreme-Value Min	comments, ctime		
servlets	Log Extreme-Value Max	comments, ctime		

表 7　Tomcat7 において選択されたモデルとメトリクス．

Module	NHPP	Logit	Poi	GLSRM
catalina	Truncated Extreme-Value Min	comments, votes	Fl, Ca, Cm	St, Cm
connectors	Log-Normal	comments, votes		
jasper	Truncated Extreme-Value Min	comments		
manager	Gamma	comments, votes		
servlets	Exponential	comments, votes, ctime		

常的に利用するためには，バグトラッキングシステムやリポジトリデータベースと連動するツールが必要不可欠である．バグトラッキングなどのデータ形式は標準化されていない場合が多く，このようなデータの標準化が，データマイニング技術と融合した信頼性評価実務において今後益々必要になると考えられる．

謝辞

本研究は，独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（SEC: Software Reliability Enhancement Center）が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援のもと行われたものである．

A カーネル法

カーネル法の基本的なアイデアは，非線形の関係を線形で表すために，データに対して非線形変換を事前に適用することから始まる．例えば，真の関係として $Y = X^2$ で表される入力値 X と出力値 Y のデータ $\{(x_1, y_1),\ldots,(x_n, y_n)\}$ に対する分析を考える．この場合，このデータに対して $Y = aX + b$ の関係を仮定した線形回帰では明らかに真の関係を捉えることができない．しかしながら，元のデータに対して非線形変換を行った新たなデータ $\{(x_1^2, y_1),\ldots,(x_n^2, y_n)\}$ を考えると，もとの関係が非線形であるにもかかわらず，$Y = aX + b$ の関係式を仮定した線形モデルによる分析で真のモデル構造を同定することができる．

このように，得られたデータに対する非線形変換を行うことは，非線形な関係を分析するのに有効な手段である．一方で，どのような非線形変換が適切かという問題がある．これを組み合わせにより解決しようと試みると組み合わせ爆発が起こるため，計算上の困難性が生じる．カーネル法はこの計算上の問題に対する一つの有効な手段を与える．

いま，Φ を任意のデータが存在する空間から（高次元の）あるベクトル空間への非線形写像とする．このとき，先の議論は Φ によって元のデータを非線形変換し，適当なベクトル空間において線形モデルで分析することに対応する．一般に線形モデルを用いた分析では，二つのデータに対する内積が定義されている必要がある．ここで，$<x, y>$ をデータ x と y の内積として表記すると，あるベクトル空間においては $<\Phi(x), \Phi(y)>$ が計測できれば，線形モデルによる分析が可能であることを意味している．カーネル法におけるアイデアの一つは，非線形写像 Φ を規定することなく，内積を表わすカーネル関数

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = <\Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})> \qquad (13)$$

を直接定義することにある．これはカーネルトリックと呼ばれ，これにより高次元ベクトル空間における内積の計算をすることなく，より少ない計算量で線形モデルによる分析が可能となる．一般に，実数値関数を考えるとき，$\mathcal{K}$ は以下を満たすような関数（正定値カーネルと呼ばれる）であればどのようなものを選んでも良い．

・対称性：

$$\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i) \quad \text{for } i, j = 1, \ldots, n. \qquad (14)$$

・正値性：対称性のもと，以下の行列 $\boldsymbol{G}$ が半正定値

$$\boldsymbol{G} = \begin{pmatrix} \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}. \qquad (15)$$

【参考文献】

[1] 2013 年度ソフトウェア工学分野の先導的研究支援事業「次世代ソフトウェア信頼性技術の開発とその実装」成果報告書 . 独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（http://www.ipa.go.jp/files/000038549.pdf）.

[2] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood. Evaluation of competing software reliability predictions. IEEE Transactions on Software Engineering, SE-12:950-967, 1986.

[3] J. A. Achcar, D. K. Dey, and M. Niverthi. A Bayesian approach using nonhomogeneous Poisson processes for software reliability models. In A. P. Basu, K. S. Basu, and S. Mukhopadhyay, editors, Frontiers in Reliability, pages 1-18. World Scientic, Singapore, 1998.

[4] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, Proc. 2nd Int'l Sympo. on Information Theory, pages 267-281. Akademiai Kiado, 1973.

[5] A. L. Goel. Software reliability models: Assumptions, limitations and applicability. IEEE Transactions on Software Engineering, SE-11:1411-1423, 1985.

[6] A. L. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. IEEE Transactions on Reliability, R-28:206-211, 1979.

[7] S. S. Gokhale and K. S. Trivedi. Log-logistic software reliability growth model. In Proc. 3rd IEEE Int'l. High-Assurance Systems Eng. Symp. (HASE-1998), pages 34-41. IEEE CS Press, 1998.

[8] N. Langberg and N. D. Singpurwalla. Unication of some software reliability models. SIAM Journal on Scientific Computing, 6(3):781-790, 1985.

[9] B. Littlewood. Rationale for a modied Duane model. IEEE Transactions on Reliability, R-33(2):157-159, 1984.

[10] M. Ohba. Inflection S-shaped software reliability growth model. In S. Osaki and Y. Hatoyama, editors, Stochastic Models in Reliability Theory, pages 144-165. Springer-Verlag, Berlin, 1984.

[11] K. Ohishi, H. Okamura, and T. Dohi. Gompertz software reliability model: estimation algorithm and empirical validation. Journal of Systems and Software, 82:535-543, 2009.

[12] H. Okamura, T. Dohi, and S. Osaki. Software reliability growth models with normal failure time distributions. Reliability Engineering and System Safety, 116:135-141, 2013.

[13] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. IEEE Transactions on Reliability, R-32:475-478, 1983.

[14] M. Zhao and M. Xie. On maximum likelihood estimation for a general non-homogeneous Poisson process. Scandinavian Journal of Statistics, 23:597-607, 1996.

[15] 山田 . ゴンペルツ曲線を用いた確率的ソフトウェア信頼度成長モデル . 情処学論 , 33(7):964-969, 1992.

[16] 福水 . カーネル法入門 . 多変量データの統計科学 8. 朝倉書店 , 2010.

第12回クリティカルソフトウェアワークショップ（12thWOCS^2）開催報告

独立行政法人 宇宙航空研究開発機構（JAXA）
研究開発本部 情報・計算工学センター（JEDI）
大久保　梨思子

独立行政法人 情報処理推進機構（IPA）
技術本部 ソフトウェア高信頼化センター（SEC）
荒川　明夫

1．開催概要

クリティカルソフトウェアワークショップ（WOCS^2）は、独立行政法人 宇宙航空研究開発機構（JAXA）と独立行政法人 情報処理推進機構（IPA）が共催する、宇宙・航空、自動車などのミッションクリティカルなソフトウェアの開発・運用・保守に関する技術やプロセスに焦点を当てたワークショップである。WOCS^2 は 2002 年から開催しており、2009 年からは IPA との共催で産、学、官の枠をも超え、ソフトウェアシステムの信頼性と安全性についての議論の場を提供している。

第 12 回を迎えた今回の WOCS^2 では、「Sociotechnical Science and Systems Engineering」をメインテーマとして掲げた。Sociotechnical Science とは、大規模複雑なシステムを技術的な側面だけでなく、システム利用者や社会構造も含めて安全性や信頼性などを考えるアプローチである。また、主テーマを実現する重要な技術領域である「信頼性と検証・妥当性確認（Reliability and V&V）」「安全性とセキュリティ（Safety and Security）」「プロセスと計測指標（Process and Metrics）」をサブテーマとして掲げ、2015 年 1 月 20 日（火）から 1 月 22 日（木）

表1　1月 20 日（第 1 日目）プログラム

10：00 ～ 12：00	**IV&V コンテスト**
13：00 ～ 13：40	**専門セミナー** **アシュアランスケース入門** 独立行政法人 情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センター（SEC） 研究員 鈴木 基史
13：40 ～ 14：30	**専門セミナー** **セーフティとセキュリティ規格の同時認証方法論について** 独立行政法人 産業技術総合研究所 セキュアシステム研究部門 システムライフサイクル研究グループ 招聘研究員 田口 研治 氏
14：40 ～ 16：00	**専門セミナー** **ソフトウェア IV&V の基礎** 独立行政法人 宇宙航空研究開発機構（JAXA） 研究開発本部 情報・計算工学センター（JEDI） 開発員 川口 真司

表2　1月 21 日（第 2 日目）プログラム

9：30 ～ 9：45	**開会挨拶** 独立行政法人 宇宙航空研究開発機構（JAXA） 技術参与 本間 正修
9：45 ～ 11：05	**基調講演** **Holistic Approach to Finding the Whole Solution: Using Systems Principles and Concepts** James N. Martin 氏 Principal Engineer, The Aerospace Corporation
11：10 ～ 12：00	**招待講演** **Assuring NASA's Safety and Mission Critical Software** Wesley W. Deadrick 氏 IV&V Office Lead, NASA IV&V Program
13：00 ～ 16：10	**一般講演（査読有）11 件**
15：40 ～ 16：55	**SECjournal 論文賞表彰式** **SECjournal 論文賞受賞者講演**
16：55 ～ 17：05	**WOCS^2 賞表彰式**

の３日間、御茶ノ水ソラシティカンファレンスセンターにて開催した。

2. プログラム概要

今回の WOCS[2] では、第１日目に技術習得を目的とした専門セミナー、第２日目に基調／招待／一般講演及び SECjournal 論文賞表彰式、第３日目に基調／招待講演を行った。プログラムを表１、表２、表３に示す。おのおの、第１日目は 46 名、第２日目は 161 名、第３日目は 143 名の方にご参加いただいた。

3. IV&V コンテスト

IV&V コンテストでは、ソフトウェア検証を実施している方、自社で行う検証に課題をお持ちの方を対象に、ソフトウェア IV&V（Independent Verification & Validation: 独立検証・妥当性確認）をコンテスト形式で体験していただいた。ソフトウェア IV&V とは、ソフトウェアの Verification（検証）と Validation（妥当性確認）を Independent（独立）に実施することであり、JAXA では 1990 年代から実施してきた。2013 年には IPA により「製品・システムにおけるソフトウェアの信頼性・安全性等に関する品質説明力強化のための制度構築ガイドライン（通称：ソフトウェア品質説明のための制度ガイドライン）」が整備され、製品やシステムの品質説明力強化のために、供給者から独立性を確保した第三者による認証活動が注目されている。今回は電子レンジの仕様書を題材として、コンテスト参加者に IV&V ケース（IV&V の検証戦略を可視化した図で、アシュアランスケースの考え方を取り入れている）を作成していただいた。作成いただいた IV&V ケースの検証戦略は、以下の４つの指標を用いて対話形式で審査された。

(a) ステークホルダ（利害関係者）に与えた安心度合い
(b) V&V に対する独自性
(c) 評価作業者に対する作業の容易性
(d) 検証戦略の有効性

様々な立場の人に対して、どのような考えに基づいて検証を行うのかなどの説明責任を果たすためのツールとして、IV&V ケースを用いることの有効性を体験していただいた。

（「製品・システムにおけるソフトウェアの信頼性・安全性等に関する品質説明力強化のための制度構築ガイドライン（通称：ソフトウェア品質説明のための制度ガイドライン）」については、下記 Web サイトを参照 http://www.ipa.go.jp/sec/reports/20130612.html）

表３　１月 22 日（第３日目）プログラム

9：50～ 10：00	**開会挨拶** 独立行政法人 宇宙航空研究開発機構（JAXA） 情報・計算工学センター（JEDI） 参与 井上 弘
10：00～ 11：30	**基調講演** **社会とテクノロジーの統合はどうすればデザインできるか？** 慶應義塾大学大学院 システムデザイン・マネジメント研究科（SDM） 准教授 白坂 成功 氏
12：30～ 13：20	**招待講演** **YRP や IIOT に於ける移動通信技術に関する動向** YRP 研究開発推進協会 会長 一般社団法人 IIOT 代表理事 甕 昭男 氏
13：30～ 14：30	**招待講演** **ソフトウェア品質リスクと品質向上技術戦略** 早稲田大学 理工学術院 名誉教授 東 基衛 氏
14：40～ 15：40	**招待講演** **つながるクルマのセーフティ＆セキュリティ** 株式会社デンソー 電子基盤技術統括部 DP- 情報セキュリティ開発室 室長 早川 浩史 氏
15：40～ 16：40	**招待講演** **JAXA のロケット開発におけるシステムズエンジニアリングの取り組み** 独立行政法人 宇宙航空研究開発機構（JAXA） 新型基幹ロケットプリプロジェクトチーム チーム長 岡田 匡史
16：45～ 17：00	**閉会挨拶** 独立行政法人 情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センター（SEC） 所長 松本 隆明

4. 専門セミナー

IV&V コンテストの後、専門セミナーとしてソフトウェアの安全性と信頼性を確保するための技術習得を目的としたセミナーを開催した。専門セミナーの開催は昨年から引き続き２回目となり、今回はアシュアランスケースに焦点を置いた。アシュアランスケースとは、与えられた運用環境において、システムに信頼性があること／安全であることの論拠である。

セミナーではまず、IPA/SEC 鈴木基史によるアシュアランスケース、セーフティケースとは何かという切り口から、アシュアランスケースをこれから作成する初心者向けの講義を行った。

続いて、独立行政法人 産業技術総合研究所 田口研治

氏から、安全とセキュリティ規格を同時に認証する際の問題点、課題と方法論についての講義が行われた。

次に、ソフトウェア IV&V の基礎と、前述の IV&V ケースの事例について JAXA 川口真司による講義を行った。IV&V ケースの事例では、IV&V 活動においてアシュアランスケースの考え方をどのように取り入れて活用しているのかを紹介した。

写真１　James N. Martin 氏

写真２　白坂成功氏

写真３　Wesley W. Deadrick 氏

（講演資料は下記 Web サイトにて公開中
http://www.ipa.go.jp/sec/events/20150120.html
JAXA 講義資料及び IV&V ガイドブックの入手については、IVV_INFO@jaxa.jp まで）

5. 基調講演

基調講演では、The Aerospace Corporation Principal Engineer の James N. Martin 氏、慶應義塾大学大学院システムデザイン・マネジメント研究科（SDM）准教授の白坂成功氏にご登壇いただき、両名から今回のテーマである Sociotechnical Science と Systems Engineering についてご講演いただいた（写真１、２）。システムズエンジニアリングという言葉は昨今よく耳にするが具体的にはどのような概念であるのか、どのように活用できるのかについてご説明いただいた。

Martin 氏の講演は、現状認識として、「複雑化が進む世界において、その課題も複雑化をしており、これらの課題を解決するためには、俯瞰的に考えて解決策をとる必要がある」ということを示した上で、そのためのアプローチを「PICARD 理論」や「システムズエンジニアリングの７人の侍」などを使って、わかりやすく伝えるものであった。

白坂氏の講演では、市場に新たな価値を提供するものを生み出すために、技術と社会の融合をデザインするための方法論とその実施例についての紹介を中心にご説明いただいた。システムズエンジニアリングから発展したエンタープライズ・システムズエンジニアリングやマサチューセッツ工科大学（MIT）が進めているエンジニアリングシステムズをもとに、慶應 SDM がおこなった新規ビジネスや街のデザインの事例を含み、今後の技術・社会融合デザインの方向性をかいま見ることができるものであった。

（基調講演資料は、下記 Web サイトにて公開中
http://www.ipa.go.jp/sec/events/20150120.html）

6. 招待講演

招待講演では、産業界、学術界、官公庁から５名の方にご登壇いただき、高信頼性ソフトウェアシステム実現のための取り組みついてご紹介いただいた。

米国航空宇宙局（NASA）IV&V Facility IV&V Office リーダ Wesley Deadrick 氏からは、近年の NASA IV&V プログラムでの取り組みや今後の課題であるセキュリティに関する検証についてご紹介いただいた（写真３）。

YRP 研究開発推進協会会長及び一般社団法人 IIOT 代

表理事 甕昭男氏からは、高速化、大容量化が進む移動体無線技術の動向を中心にご講演いただいた。

早稲田大学理工学術院名誉教授 東基衛氏からは、ソフトウェア品質向上のための技術戦略を ISO/IEC 25000 シリーズ（SQuaRE）の品質モデルなどの例を挙げながら紹介いただいた。

株式会社デンソー電子基盤技術統括部 DP- 情報セキュリティ開発室室長 早川浩史氏からは、車載システムのセーフティとセキュリティの動向と、それに対するデンソーの取り組みについてご講演いただいた。

JAXA 新型基幹ロケットプリプロジェクトチームチーム長 岡田匡史は、JAXA のロケット開発におけるシステムズエンジニアリング適用及び適用における課題について紹介した。

（講演資料は下記 Web サイトで公開中
http://www.ipa.go.jp/sec/events/20150120.html）

7. 一般講演

一般講演では、様々な産業分野の企業、大学、研究所に所属する方々から全 11 件のご講演をいただいた。一般講演のうち、とくに優れていた発表について、最優秀賞と優秀賞を授与した。受賞者を表 4 に示す。

最優秀賞を受賞した 2 名のうち、松並氏はシステム全体のセキュリティ設計がどのようになされているのかを分析、可視化する手法を、小林氏は開発成果物のレビュー過程や結果を D-Case（利害関係者同士が合意を得るために用いる構造化されたドキュメント）を用いて可視化する手法についての発表であった。両名とも、ソフトウェアやシステムの品質がどこまで保証されているのか、全体を俯瞰しながら可視化することを目的としており、今回の WOCS^2 の主旨と合致していた。

8. SECjournal 論文賞表彰式

1 月 21 日の一般講演の後、WOCS^2 会場内にて 2014 年 SECjournal 論文賞の受賞論文の発表を行った。受賞論文については表 5 の通り。優秀賞を受賞した中村氏の「ソフトウェアプロダクトラインのエンタープライズ・システムへの適用と評価」についての論文は、10 年間の開発実績を通じ、組織全体での開発ソースコード量、開発コストの削減という成果をあげた点が高く評価された。

（SEC journal については、下記 Web サイトを参照
http://www.ipa.go.jp/sec/secjournal/index.html）

表 4　12thWOCS^2 賞 受賞者と講演テーマ

受賞講演	
最優秀賞	**「ソニーの電子お薬手帳システムに適用したセキュリティ設計分析手法」** 松並 勝（ソニーデジタルネットワークアプリケーションズ株式会社）
	「D-Case を用いたレビューを見える化する方法の導入事例」 小林 展英（株式会社デンソークリエイト）
優秀賞	**「CAN メッセージにおける振る舞い検知手法に関する考察」** 氏家 良浩（パナソニック株式会社）
	「要求逸脱に基づく例外試験項目の作成実験」 山本 修一郎（名古屋大学）

表 5　SECjournal 論文賞 受賞者と論文テーマ

受賞論文	
優秀賞	「ソフトウェアプロダクトラインのエンタープライズ・システムへの適用と評価」 中村 伸裕（大阪大学 / 住友電気工業株式会社）
SEC 所長賞	「ピアレビュー有効時間比率計測によるピアレビュー会議の改善と品質改善の効果」 久野 倫義（三菱電機株式会社 設計システム技術センター）
	「プラットフォーム依存種検索によるソースコードからのプラットフォーム依存部抽出手法 」 岡本 周之（株式会社日立製作所 横浜研究所 / 大阪大学 大学院情報科学研究科）

9. 今後の WOCS^2 について

今回の WOCS^2 では、ソフトウェアやシステムを技術的な側面だけではなく、利用者なども含めた全体として捉えることをテーマとして開催し、参加者からも好評であった。今回もアンケートを通して多くのご意見をいただいたので、次回 WOCS^2 の基調・招待講演を含む全体構成に反映していく予定である。また、専門セミナーについては同一内容の開催やより詳しい研修の要望が多く、新たなセミナーを企画していきたい。

10. 謝辞

WOCS^2 プログラム委員の皆様、後援団体の皆様にはワークショップ成功のためにご支援いただきました。ここに深謝いたします。

SECjournal 論文賞 受賞論文発表

SEC は、我が国ソフトウェア産業発展のための様々な取り組みを実施しておりますが、その取り組みの一つとして、ソフトウェア工学に関する論文に賞を設け表彰を行っております。

今年の SECjournal 論文賞は、2013 年 8 月から 2014 年 7 月までに投稿された合計 13 編のうち、査読者により採録された 7 編の論文を候補とし、選考委員会と表彰委員会による厳正な審査の結果、 3 編を選出いたしました。

各賞の発表と表彰式は 2015 年 1 月 21 日に第 12 回 WOCS[2] と併催で実施されました。本年は最優秀賞は該当なし、優秀賞 1 編、所長賞 2 編が表彰されました。片山表彰委員長による審査報告は本号 42 ページに掲載されています。

SECjournal 論文賞表彰委員会 委員

委員長	片山　卓也	北陸先端科学技術大学院大学 名誉教授
委員（50 音順）	有賀　貞一	AIT コンサルティング株式会社 代表取締役
	岩野　和生	三菱商事株式会社 ビジネスサービス部門 顧問
	大原　茂之	一般社団法人 スキルマネージメント協会 理事長
	土井　美和子	独立行政法人 情報通信研究機構 監事
	松田　晃一	独立行政法人 情報処理推進機構 顧問
	松本　隆明	独立行政法人 情報処理推進機構 技術本部 ソフトウェア高信頼化センター 所長
	横塚　裕志	一般社団法人 情報サービス産業協会 副会長

SECjournal 論文賞選考委員会 委員

委員（50 音順）	飯泉　紀子	株式会社日立ハイテクノロジーズ 医用システム設計開発本部 医用ソフトウェア設計部 主管技師
	大島　啓二	一般財団法人 日本科学技術連盟
	兼本　茂	会津大学 コンピュータ理工学部 教授
	神庭　弘年	神庭 PM 研究所 所長
	楠本　真二	大阪大学 大学院 情報科学研究科 教授
	紫合　治	東京電機大学 情報環境学部 情報環境学科 教授
	新谷　勝利	新谷 IT コンサルティング 代表
	寺中　勝美	NTT ソフトウェア株式会社 常勤監査役
	古山　恒夫	東海大学 理学部 非常勤教授
	松本　健一	奈良先端科学技術大学院大学 情報科学研究科 教授
	水野　修	京都工芸繊維大学 大学院 工芸科学研究科 情報工学部門 准教授
	神谷　芳樹	みたに先端研合同会社 代表社員
	峯　恒憲	九州大学 大学院 システム情報科学研究院 情報知能工学部門 准教授
	森崎　修司	名古屋大学 大学院 情報科学研究科 准教授
	山城　明宏	東芝ソリューション株式会社 ソリューションセンター ソリューション品質保証部 ソリューション品質企画担当 主幹
	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
	山本　雅基	名古屋大学 大学院 情報科学研究科 附属組込みシステム研究センター 特任教授
	鷲崎　弘宜	早稲田大学 基幹理工学部 情報理工学科 准教授

選考委員会では、全委員の査読結果を含め、審査を行った。
ただし、委員が著者の論文や委員の関係者の論文については、該当委員は審査を行っていない。

優秀賞

ソフトウェアプロダクトラインの エンタープライズ・システムへの適用と評価

中村　伸裕、谷本　收、楠本　真二
（SEC journal 35 号掲載）

所長賞

ピアレビュー有効時間比率計測による ピアレビュー会議の改善と品質改善の効果

久野　倫義、中島　毅、松下　誠、井上　克郎
（SEC journal 36 号掲載）

所長賞

プラットフォーム依存種検索によるソースコードからのプラットフォーム依存部抽出手法

岡本　周之、藤原　貴之、楠本　真二、岡野　浩三
（SEC journal 39 号掲載）

SECjournal 論文賞 2014

上段左より、松本　隆明、土井　美和子、大原　茂之、横塚　裕志、　（敬称略）
片山　卓也、久野　倫義、中村　伸裕、岡本　周之、立石　譲二※

※ IPA 理事。理事長藤江の出張に伴い代行出席。

SECjournal 論文賞　表彰委員会審査報告

SECjournal 論文賞
表彰委員会委員長
北陸先端科学技術大学院大学
名誉教授
片山　卓也

SEC journal は、ソフトウェア開発現場での先端技術の実践や開発の報告、論文の掲載などを通して我が国のソフトウェア産業、IT 産業の技術力向上に貢献してきました。そして、そのような論文の中からとくに優れたものを毎年選び表彰を行ってきました。今回は、2013 年 8 月からの 1 年間に掲載された論文を対象に論文賞選考委員会、表彰委員会で審査を行い、以下の 3 論文を優秀論文として表彰することを決定いたしました。いずれも優れた内容のものであると同時に、実際の開発現場における有効性などを評価の主な観点と致しました。

「ソフトウェアプロダクトラインのエンタープライズ・システムへの適用と評価」

ソフトウェア資産再利用技術として組込みシステム開発ではその有効性が示されているソフトウェアプロダクトライン技術を、著者の属する組織で使われる社内情報システム開発に適用して、開発ソースコード量やコストの削減を達成した事例を紹介したものである。このようなシステムの特徴として、データ入出力に関連した画面や画面遷移の再利用が有効であることに着目して、プロダクトライン技術の適用を行ったもので、10 年間の開発実績を通してその有効性が示されていることが高く評価された。

「ピアレビュー有効時間比率計測によるピアレビュー会議の改善と品質改善の効果」

ソフトウェア開発における品質向上を目的としたピアレビュー会議の実施を適切に行いその効果を高めるために、会議の目的や会議プロセスを明確に定めると同時に、時間の有効な使われ方の計測法を定めることにより、ピアレビュー会議の有効性が向上することを実証的、計量的に示した報告である。ややもすると漫然とした会議運営になりがちなピアレビュー会議において、欠陥抽出を有効に行うための様々な工夫が示され、それが社内教育により組織の中で実践されていることが評価された。

「プラットフォーム依存種検索によるソースコードからのプラットフォーム依存部抽出手法」

組込みシステムのプラットフォーム（CPU、OS）変更に伴う移植作業に関して、ソースコードに含まれるプラットフォーム依存部分の抽出方法とそのためのツール構築について述べたものである。過去の移植事例の調査などをもとに定めた 39 個の依存種を対象にして、ソースコードからそれらを抽出するためのプログラム解析ツールを作製し、それを実際の移植作業に適用し、方法論とツールの有効性を評価したものである。ソースコード中に依存部分が分散しがちな組込みシステムへの有用性が評価された。

― 受賞者コメント ―

ソフトウェアプロダクトラインのエンタープライズ・システムへの適用と評価

中村　伸裕（大阪大学／住友電気工業株式会社）


中村　伸裕


谷本　收


楠本　真二

ソフトウェア再利用技術は品質、開発コスト、納期の改善策として古くから期待されている。ソフトウェアプロダクトライン（以下、SPL）はアドホックな再利用ではなく、計画的な再利用を実現する手法である。組込み系ソフトウェアなどで成果が報告されているものの、企業内で利用するエンタープライズ・システムでの事例報告は少なく、定量的な効果が示されていない状況であった。本論文は住友電工グループでの 10 年間にわたる取り組みの内容と成果をまとめたものである。課題の 1 つは効果が継続的に出せるソフトウェア資産の構築であった。広く認識されているアプローチは業種ごとに共通部品を開発し、ビジネスロジックを再利用するというものであるが、住友電工では逆にビジネスロジック以外の画面や画面遷移といった処理をソフトウェア資産として開発した。その結果、開発するソースコード量は約 92％削減できた。開発生産性は IPA 発行の『ソフトウェア開発データ白書』のデータと比較して 3 ～ 5 倍となった。また、顧客満足度調査の結果、操作性に関する設問に 60％の利用者が ” 非常によい ” または ” よい ” と答えており、ソフトウェア資産の再利用により利用者の満足度も改善できることがわかった。

ピアレビュー有効時間比率計測によるピアレビュー会議の改善と品質改善の効果

久野　倫義（三菱電機株式会社　設計システム技術センター）


久野　倫義


中島　毅


松下　誠


井上　克郎

ソフトウェア開発において、要求定義や設計段階の欠陥をピアレビューにより除去することは、品質的にもコスト的にも有効です。しかし、これまで、その中心的活動であるピアレビュー会議が実際にどのように行われているかを定量的に把握しないまま、その有効性の向上のため技法の導入や有識者の参加ばかりに目が行きがちでした。

本研究では、まず開発現場で行われているピアレビュー会議を作業者と作業項目ごとに時間計測し、その実態の把握を行いました。その結果、現場においてはピアレビュー会議が、説明会、設計会議、欠陥検出などの様々な活動タイプに分かれていることがわかりました。この結果に基づき、改善の方向として、ピアレビュー会議が欠陥検出活動になるようにプロセスを再定義し教育することで、欠陥抽出の効率が向上し最終的な品質も改善することができました。

今後も、こうした現場に密着した改善活動を続け、更なる高品質なモノづくりを進めていきたいと思います。

プラットフォーム依存種検索によるソースコードからのプラットフォーム依存部抽出手法

岡本　周之（株式会社日立製作所 横浜研究所
／大阪大学大学院情報科学研究科）


岡本　周之


藤原　貴之


楠本　真二


岡野　浩三

組込みシステム開発は理想的に進まないことが多い。厳しいコスト制約のため限られたリソースの中でソフトウェア処理の高速化が求められることもあれば、出荷間際で開発期間が不足する中で，不具合に対するソフトウェア修正が求められることもある。これらの要求に応えるためには、ソフトウェア階層構造を崩してでも、実行処理が高速な実装や短時間修正が可能な実装が優先される。更に、開発期間の不足でソースコードのみを修正することがあり、これにより仕様書やコメントの情報とソースコードの整合性が崩れる。

このような状況が多発する実際の開発において、プラットフォーム（PF）変更に伴うソフトウェア移植作業効率をいかに向上するかを研究し、まとめたのが本論文である。

提案した PF 依存部抽出手法は、社内の過去の移植事例を中心に集めた知見を基にしており、関係各位のご協力がなくては本論文は完成し得なかった。この場を借りて御礼申し上げる。

情報システムの事故データ

情報システムの障害状況 2014年後半データ

IPA 顧問
松田　晃一

SEC システムグループ 主任
八嶋　俊介

2014年7月から12月までに報道された情報システムの障害状況を報告する。この間に報道された情報システムの障害は合計11件であった。これらの事故事例の中から、システムの処理能力を大きく超える突発的なトラフィックの集中によって発生した事故及び保守作業に伴って発生した事故を取り上げて若干の考察を加え、今後の同種事故防止の参考としたい。

1．2014年後半の概況

2014年7月から12月までの半年間に報道された情報システムの障害は合計11件であり、その概要は表1に示す通りである。月平均を見ると1.8件／月となり平均的な値であった。しかし、2014年を通算すると合計36件､月平均3.0件という比較的高い値となった（図1）。これは、2014年4月に実施された消費税の8％への改定に伴うシステム更新に関連するトラブルが集中的に起こったことが原因である [松田 2014]。今期の11件の事故の内、原因が判明しているものについて見ると、2件（表1事例1432、1433）※1がシステムの処理能力を超える大きなトラフィックが突発的に発生したことを契機とする事故である。また、3件（表1事例1426、1428、1434）は保守作業に伴って発生している。次節以降ではこの2つの原因による事故について取り上げる。

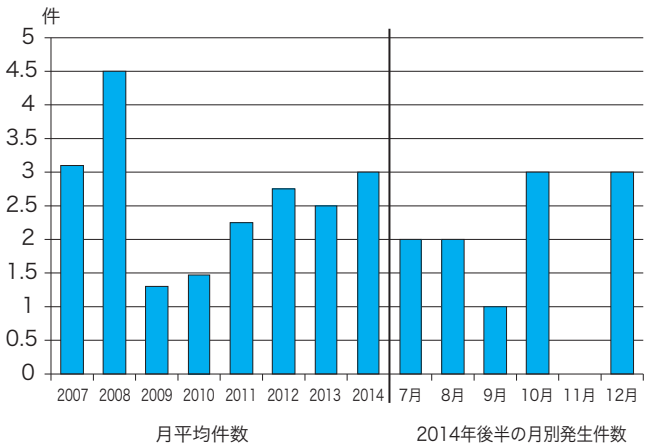


図1　情報システムの障害発生件数の推移

2．突発的な大量トラフィックによる事故

システムに対する処理要求は時間的に変動する。システムの容量設計においては、時間変動の平均値ではなくピーク時のトラフィックをカバーできるように設計することが必要である。一日の内にも時刻によって変動し、更に曜日によっても、あるいは給料日や決算日、月末などの特異日の要因によってその集中度合いは異なるため、設計条件としてどの値を取るかはよく吟味する必要がある。このようなトラフィックの特性は、これまでの運用実績などからある程度予測が可能であり、それらに基づいて一定の余裕度を見て設計が行われる。ただし、まれに発生するトラフィックまですべて想定すると、システムの経済性を損なうことになるため、一定のトラフィック値を上限として設計される。しかし、実際にはこの値を超えるトラフィックが発生することも想定し、それでもシステムが完全に停止するなどの事象が起こらないように負荷を調整する機構を同時に組み込むのが通例である。すなわち、溢れそうな処理要求は、処理を保留あるいは受付けずに、ピークを時間的に平滑化するような負荷制御である。高速道路で渋滞が発生すると、

【脚注】

※1　事例に付与されている番号は、前半2桁は事例発生した西暦年の下2桁、後半2桁はその年に発生した事例の通し番号であり、本連載を通して一意の番号となっている。

一部の入口を一定時間閉鎖して、流入する車の量を制御することによって既に道路上に進入している車をスムーズに流し、渋滞をできるだけ早く解消することと同じである。

しかし、定型的に変動するトラフィックとは異なり、突発的な事象によって突然発生する大量トラフィックへの対策は一層難しい。大きなイベントが行われて狭いエリアに集中した多くの参加者が一斉に携帯電話を使う例や、チケットの予約サイトに予約開始日時に一斉にアクセスが集中するなど多くの事例を挙げることができる。このようなイベントの場合は、ある程度事前に予想ができ準備ができる可能性はある。例えば、クラウドコ

表1　2014年後半の情報システム障害データ（報道に基づきSECが整理）

No.	システム名	発生日時（上段） 回復日時（下段）				影響	現象と原因	直接原因	情報源
		年	月	日	時				
1426	ハローワーク求人情報検索システム	2014	7	22	9時20分	全国1,000ヵ所、21,000台設置されている求人情報検索用端末について、通常は2秒程度で結果が表示されるが、数十秒かかったり、そのままエラーになって検索結果が表示できなくなった。	7月19日からの三連休を使って求人情報管理用のサーバをリプレースしたところ、22日になって不具合が表面化した。（端末や検索画面は変更しておらず、ハードウェアだけのリプレース）原因はサーバの不具合とネットワーク機器の不具合。テスト時には問題なかったが、本番用の回線に切り替えた際に障害が発生した。サーバの不具合はメモリを交換することで解消されたが、ネットワーク機器の不具合は10月2日時点でまだ解消しておらず、まれに遅延が発生している。	ハードウェア障害	・日本経済新聞電子版（2014.7.22） ・日本経済新聞電子版（2014.7.23） ・日経コンピュータ（2014.10.16号）
		2014	7	25					
1427	JR西日本予約システム	2014	7	25	4時30分	JR西日本の新幹線や在来線特急のインターネット予約システム「e5489」と電話予約のシステムに不具合が生じた。予約の受付や券売機での切符の受け渡しができなかった。影響件数は約2,200件。	予約管理のサーバに不具合があった。エクスプレス予約には影響がなかった。	ハードウェア障害	・朝日新聞電子版（2014.7.25）
		2014	7	25	7時50分				
1428	世田谷区役所基幹システム	2014	8	4	8時30分	システム障害が発生し、190台の窓口端末で一日業務が停止した。住民票の転出入の入力や国民健康保険の加入、印鑑の新規登録などができなかった。	ベンダのデータセンター内にある負荷分散装置で障害が発生して通信ができなくなった。原因は負荷分散装置のバグで、連続運転を続けるとメモリ不足の状態に陥るようになっていた。待機系に切り替えたが、待機系も同様の状態に陥っており、正常に機能しなかった。 暫定措置…負荷分散装置のログを監視する仕組みを整備し、同じトラブルが発生してもすぐに対応できるようにした。（8月11日） 本格対処…バグの修正パッチを適用した。（9月15日）	ソフトウェア障害	・読売新聞朝刊（2014.8.5） ・ITpro（2014.8.5） ・日経コンピュータ（2014.10.30号）
		2014	8	4	17時00分				
1429	大阪市営地下鉄御堂筋線	2014	8	19	9時40分	梅田駅構内で、なかもず方面に向かう下り線の信号機が、赤信号のまま変わらなくなった。上下線計22本が最大50分遅れ、約1万人に影響した。	信号装置のトラブル。原因調査中。	不明	・朝日新聞電子版（2014.8.19） ・産経新聞電子版（2014.8.19）
		2014	8	19	13時30分				
1430	埼玉県加須市「ゲリラ攻撃情報」緊急メール	2014	9	11	16時30分	住民に緊急情報を流すメールで、「ゲリラ・特殊部隊攻撃情報」を誤って配信した。利用登録者約6,000人のうち、4,250人がメールを受け、市に問い合わせの電話が相次いだ。	全国瞬時警報システム（Jアラート）の整備に伴い、防災行政無線のソフトウェア改修中に、誤って配信された。市はホームページにおわびを掲載したり、誤配信を知らせるメールを送ったりした。	保守作業ミス	・朝日新聞朝刊（2014.9.12）

No.	システム名	発生日時（上段） 回復日時（下段）				影響	現象と原因	直接原因	情報源
		年	月	日	時				
1431	JR 東日本非常ブレーキ	2014	10	6		JR 東日本は山手線や京浜東北線などで走る 1,548 の車両（保有車両の 1/3）について、非常ブレーキのシステムに不具合があったと発表した。※ JR 各社及び民鉄各社からも同様の発表があった。	運転士の居眠りや急病に備え、システムでは、電車の走行中に 60 秒間、運転士がブレーキや汽笛などを動かさないと警告ブザーが鳴る。更に 5 秒間放置すると、「運転士に異常発生」と判断。非常ブレーキがかかる仕組みとなっている。しかし現状は、自動列車制御装置 (ATC) などにより速度が抑えられても、運転士が操作したと認識されてしまうため、運転士にトラブルが起きていることが分からない状態になっていた。原因は制御ソフトウェアのミス。JR 東日本は、緊急ブレーキがきかなくても衝突の危険はないとし、1 年かけてソフトウェアを改修する予定。	ソフトウェア障害	・JR 東日本プレスリリース (2014.10.6) ・朝日新聞朝刊 (2014.10.7) ・日本経済新聞朝刊 (2014.10.7) ・JR 西日本プレスリリース (2014.10.8) ・朝日新聞朝刊 [大阪版] (2014.10.9)
1432	横浜市 Web サイト	2014	10	13	20 時頃	台風 19 号の詳細情報が掲載されている横浜市の Web サイトがダウンし、アクセスができなかった。	台風 19 号の接近に伴い、約 370 万人が住む市内のほぼ全域の携帯電話に、「緊急速報メール」が配信された。土砂災害の恐れがあるという内容だったが、システムの文字数制限が 200 文字のため、対象の地区は横浜市の Web サイトが参照された。結果、Web サイトにアクセスが集中し、サーバがダウンした。暫定対処として、Web サーバから容量の大きい地図データを削除し、文字で危険箇所を示すようにした。	アクセス集中	・日経コンピュータ (2014.11.27 号)
		2014	10	13	深夜				
1433	NTT ドコモ	2014	10	20	5 時 45 分	北海道にて契約の一部顧客にて、音声通話とデータ通信が利用しづらい状況が発生した。影響は最大で約 70 万人。	LTE 回線「Xi（クロッシィ）」と、第 3 世代携帯電話サービス「FOMA」を利用する一部の端末が通信障害の影響を受け、音声通話とデータ通信がしづらい状況となった。障害の原因は、利用者の契約情報と通信状況をひもづけて制御する、同地域のネットワーク設備の輻輳（ふくそう）だった。復旧対策は次の二つを実施した。 ・北海道地域における通信を一部制限して輻輳を緩和 ・障害の原因となったネットワーク設備の利用者情報を再設定	ネットワーク設備の輻輳	・NTT ドコモプレスリリース (2014.10.21) ・日経コンピュータ電子版 (2014.10.21)
		2014	10	21	4 時 25 分				
1434	JR 東日本えきねっとモバイル Suica	2014	12	7	0 時 20 分 5 時 30 分	「えきねっと」、「モバイル Suica」システムが計約 4 時間半にわたって停止し、以下の影響があった。 ・えきねっとを利用して切符の受け取りができなかった件数が約 1,100 件 ・モバイル Suica のエラー件数が約 3,000 件	システムを管理する関係会社間で、メンテナンスを実施する情報が共有されていなかったのが原因。利用者へ通告がないままサービスが停止した。事前に購入していた切符を受け取れなかった利用者には後日、払い戻しを行う。	情報の共有ミス	・朝日新聞電子版 (2014.12.7) ・毎日新聞電子版 (2014.12.7) ・日本経済新聞朝刊 (2014.12.8)
		2014	12	7	1 時 40 分 8 時 47 分				
1435	JR 北海道自動列車停止装置 (ATS)	2014	12	11	10 時 20 分	自動列車停止装置（ATS）の誤作動で非常ブレーキがかかった。列車は乗客約 280 人を乗せたまま、約 4 時間半にわたって停車した。	自動列車停止装置（ATS）の誤作動で非常ブレーキがかかった。運転士が手動でブレーキを解除し運転を再開したが、約 25 分後に再びブレーキが誤作動し、緊急停車した。	不明	・日本経済新聞夕刊 (2014.12.12)
1436	国土交通省航空局管制システム	2014	12	18	9 時 45 分	管制システムのデータ通信回線に障害が発生した。日本航空の 6 便が欠航し、全日空でも遅れが生じた。	東京航空交通管制部と羽田空港を結ぶ回線で通信ができなくなり、飛行計画書のやりとりができなくなった。その結果、羽田空港のレーダーに航空機の情報が表示されなくなった。障害発生中は、情報を手動で入力したり、管制官同士で電話でやりとりして対応した。	通信回線障害	・時事通信 (2014.12.18) ・読売新聞電子版 (2014.12.18)
		2014	12	18	11 時 45 分				

ンピューティングを利用してサービスを提供している場合ならば、スケールアウトの機能を使って短時間の間に処理能力を拡張できるため、このような事故を軽減できる可能性がある。しかし、事件、事故、災害などの場合は、このような事前の準備はほとんど不可能とも言え、なお対処は難しい。事例 1432 はその典型である。すなわち、災害の発生を警告するメールが一斉に送信され、詳細な情報を見るために誘導された WEB サイトに大量のアクセスが集中しダウンしてしまった事故である。先に述べた適正な容量設計、負荷制御機構、スケールアウト機構などは、もちろんこれらに対する対策とはなり得るが、十分とは言えない。また、このような事故は発生の初期段階では通常の運用監視の仕組みからは検出し難く、放置されたまま大規模な障害につながりやすい。いわゆる「サイレント障害」と呼ばれるもので、適切なサービス監視によってこのような障害をできるだけ早く検出し対応する必要がある。SEC が公開した障害教訓集 [SEC 2015] においても [教訓 T11]「サイレント障害を検知するには、適切なサービス監視が重要」として取り上げているので参考にされたい。

これまでも、この連載において高負荷を契機とした障害を取り上げ、容量設計や負荷制御の重要性を何度か指摘してきた [松田 2012]、[松田 2013]。それだけ、このような高トラフィックを契機とした事故が後を絶たないことを示している。対策が望まれるところである。

3．保守作業にかかわる事故

今期 11 件の事例の内、3 件は保守作業に関連した事故である。

事例 1426 はハードウェアの保守作業によって更新したハードウェアに不具合があったと報じられている。前々号でも保守を行う前には事前の手順確認が必要と述べたが [松田 2014]、もしこの手順が実施されていたら新しいハードウェアの不具合は事前に見つけることができていたのではないか、と思われる。

また、事例 1428 はソフトウェアのバグが原因であったが、そのバグを解消するパッチが事故以前に既に出ていたにもかかわらずその情報が把握されず、パッチが未適用であったためにバグが顕在化し、事故が発生してしまった。もともとの原因はソフトウェアのバグであるが、保守作業が適時適切に行われていれば発生しなかった事故であろう。前出の障害教訓集 [SEC 2015] においては、[教訓 T16] として取り上げている。近年は IT システムに対するセキュリティ攻撃が頻発しており、このような脅威から守るためにも常に最新のパッチを適用することは必須である。

事例 1434 は保守作業を実施する情報が運用管理している部署に正しく伝わらないまま保守が行われて発生した事故である。単純なミスであり、関係部門間での情報の共有という基本的な動作が行われてさえいれば避けられた事故であるだけに残念である。システムが大規模、複雑になるにつれ作業分担、分業が進み、かかわる組織が多数になってきており、この種の問題が発生する可能性が一層高くなっている。日常作業において情報共有の手順を定着させておくことが必要である。

4．むすび

2014 年後半半年間の情報システムの障害について、報道などをもとに整理し報告した。今期の事故事例の中からもこれからの開発・運用・保守にあたって参考にすべき教訓を汲み取ることができる。今後とも、これらの経験を社会の共通の財産として共有し、少しでも事故を防ぎ、安心・安全な IT 社会に向けて地道な努力を続けていく必要がある。

SEC では様々な事故の原因や対策について多方面から考察を行い、業界横断的に利用可能な要素を抽出し「見える化」する活動を行い、その成果を教訓集として公表した [SEC1 2014]、[SEC2 2014]。2014 年度においてもその活動を継続し、その成果を 2015 年 3 月に「情報処理システム高信頼化教訓集（2014 年度版）」[SEC 2015] として公開した。今後ともこの活動を継続発展させ、新たな教訓を更に追加すると共に、得られた教訓を広く共有し、活用を促す活動を推進していく予定である。システム障害の再発や影響拡大を防ぐために、経験者や関連事業者の方々に、この事業への積極的な参画と協力をぜひお願いしたい。

【参考文献】

[松田 2012] 松田晃一・大高浩：情報システムの障害状況 2012 年前半データ , SEC journal No.30, Vol.8 No.3, PP139-141, 2012 年 9 月

[松田 2013] 松田晃一・鈴木三紀夫他：情報システムの障害状況 2013 年前半データ , SEC journal No.34, Vol.9 No.3, PP142-146, 2013 年 9 月

[松田 2014] 松田晃一・八嶋俊介：情報システムの障害状況 2014 年前半データ , SEC journal No38, Vol.10, No3, pp.42-47, 2014 年 9 月

[SEC1 2014] 独立行政法人 情報処理推進機構 SEC：情報処理システム高信頼化教訓集（IT サービス編）, 2014 年 5 月

[SEC2 2014] 独立行政法人 情報処理推進機構 SEC：情報処理システム高信頼化教訓集（製品・制御システム編）, 2014 年 5 月

[SEC 2015] 独立行政法人 情報処理推進機構 SEC：情報処理システム高信頼化教訓集（2014 年度版）（IT サービス編）, 2015 年 3 月

SWEBOK V3.0 日本語訳版[※1]の連続紹介 −3の1

SC7/WG20[※2] エキスパート、新谷 IT コンサルティング
新谷　勝利

1．はじめに

SWEBOK は、Software Engineering Body of Knowledge の省略形で、それが広範に用いられることを目的とし、そのオリジナルの英語版はすべて無料で脚注に示す URL からダウンロード可能になっている。2001 年に SWEBOK Trial Version[※3]、2004 年に SWEBOK 2004[※4]、2013 年に SWEBOK V3.0[※5] がそれぞれ発行されている。

SWEBOK V3.0 を紹介するにあたり、連続紹介の第一回の今回は、そもそもソフトウェア・エンジニアリングはどのように世界に紹介されたか、その後今回の SWEBOK V3.0 が発行されるまでの歴史を振り返って見ることにする。対象とするのは、1）1968 年の NATO 会議の Software Engineering[※6] というタイトルの報告書、2）2001 年 5 月発行の SWEBOK Trial Version、3）2005 年 6 月発行のソフトウェア・エンジニアリング基礎知識体系―SWEBOK 2004―日本語訳版であり、それぞれを抄訳にて紹介する。第二回及び第三回は、SWEBOK V3.0 の知識領域を 2 回に分けて説明する。

なお、今回用いる技術用語は、SWEBOK V3.0 日本語版の訳者により「SWEBOK V3.0 原語対訳表」が追加されているので、基本的にそれに従うものとする。場合により、ISO/IEC/IEEE Software and Systems Engineering Vocabulary（SEVOCAB[※7]）を参照する。

2．1968 年の NATO 会議の Software Engineering というタイトルの報告書

オリジナル版の報告書は 1969 年 1 月に発行されており、その後報告書をスキャンし OCR 技術を用いてテキスト版が作成され、編集されたものが、2001 年にニューキャッスル大学のサーバーから広く入手できるようになっている。本紹介は、ダウンロード版に依り、抄訳は前半のソフトウェア・エンジニアリング全般に関する議論の部分である。ぜひ報告書をダウンロードの上、前半部分のみならず後半部分も読まれることをお勧めする。今日に通じるものが既に 1969 年初めには広く報告されていることが分かる。後続の SWEBOK の進化の紹介でも明らかになるが、学問的基礎及び実践的な規律の上に成立するエンジニアリングが「身に付く」ためには 50 年という年数では足らないのではないかと思わざるを得ない。

2.1 NATO 会議の背景

1967 年秋に、NATO 科学委員会はコンピューター科学に関する作業部会を発足させている。その年の暮れには、ソフトウェア開発者がその開発にあたり、他のエンジニアリング分野と同様学問的基礎に立脚すると共に実践的な規律に基づくことが必要であり、それを表す用語として、「ソフトウェア・エンジニアリング」という新しい用語を生み出している。この作業部会は、ソフトウェ

【脚注】

※ 1　松本吉弘訳、ソフトウェアエンジニアリング基礎知識体系－SWEBOK V3.0 －、オーム社、2014 年 11 月 25 日、ISBN978-4-274-50521-8

※ 2　SC7/WG20 は、ソフトウェア及びシステムの知識体系とプロフェッショナルの形成にかかわる国際標準を審議する委員会。SWEBOK を審議。

※ 3　以下から入手可能、http://cisas.unipd.it/didactics/STS_school/Software_development/Trial_Version1_00_SWEBOK_Guide.pdf

※ 4　以下から入手可能、http://www.computer.org/portal/web/swebok/2004guide;jsessionid=dba3ca44da150499851e19bd752a

※ 5　以下から入手可能、http://www.computer.org/portal/web/swebok/swebokv3

※ 6　Software Engineering, Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th Octo., 1968、本報告書は以下から入手可能、http://homepages.cs.ncl.ac.uk/brian.randell/NATO/

※ 7　IEEE と ISO/IEC/SC7 の共同プロジェクトで以下から用語が検索可能、http://pascal.computer.org/sev_display/index.action

アの設計、ソフトウェアの生産、及びソフトウェアのサービスの観点で議論を進めることを決めた。

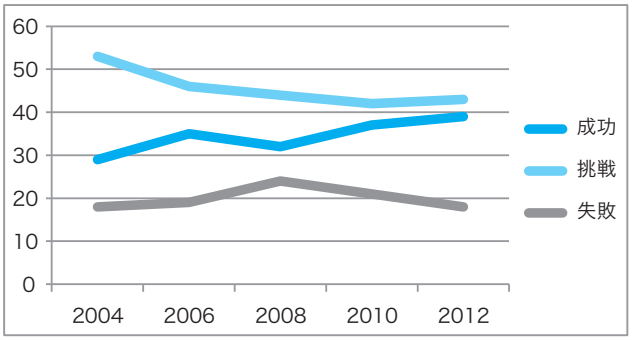
2.2 ソフトウェア・エンジニアリングと社会

前述作業部会は、コンファレンスを組織化し、後日発行される報告書を幅広い読み手に読んでもらうために技術論のみならず、幅広い観点での議論を進めた。この議論の中には以下のようなものがあった。

- コンピューターの導入は年率 25 ～ 50％で増えており、これらの増加に対応するソフトウェアのために 25 万人にのぼる分析者及びプログラマに何らかの影響を与える。
- ほとんどのプログラムは正常に動くであろうが、そうでないプログラムも出てくる。
- 危機とまでは言えないかも知れないが、とくに大規模システムにおいては、危惧がある。
- 大規模システムにおいては、ソフトウェアの不具合の発生をなくすことはできない。
- 他のエンジニアリング分野と比べ、ソフトウェア・エンジニアリングはまだその初期段階にある。
- プログラミングのコスト、スケジュール管理は、依然として低い評判のままである。
- ソフトウェア開発の工程管理の難しさは、進捗をどう測定するのがよいのかわかっていないことにある。
- ソフトウェア不具合は指数関数的に増加している。
- ソフトウェア開発への要望は現場の能力を超えてなされている。

これらは、驚くことに 2015 年の今における発言と異なるものはほとんどない。NATO の会議でソフトウェア・エンジニアリングの基礎及び方向性が見定められたのだとすると、この間の進歩は何であったのであろう？ベンチマーキングを数千のプロジェクトデータで長年実施し Chaos Manifesto 2013 [※8] として報告しているものに、次のデータがある。ここでは、「成功」とは、品質、コスト及び納期が予定した通り及び凌駕したものを示し、「挑戦」とは 3 項目の内どれかが予定に収まらなかったもの、「失敗」とは出荷出来なかったものを示す。

残念ながら 1969 年以降 2003 年に至るデータはないが、最近の 8 年間のデータで見ると、その間、「成功」と「挑戦」において約 10％の改善は見られるものの、ほとんどフラットと言ってもよいであろう。成功が



50% に満たないエンジニアリング成果は他の分野では見られないものではないか。これは、主として欧米のデータであり、日本はこれらより良いと言われている。IPA/SEC の調査ではこれほどの期間のものはないが、スナップショットで、成功が大凡 80% となっている。ただ、日本のデータは母集団が小さい上にプロジェクトの大きさが Chaos Manifesto では小規模のものに相当し、その場合約 70% の成功になっている。NATO 会議においても、ソフトウェアが大規模化していることが問題の出発点と指摘している。

2.3 ソフトウェア・エンジニアリング

コンファレンスを通し、ソフトウェア・エンジニアリングの様々な側面が議論された。

2.3.1 ソフトウェア・エンジニアリングの性質

- ソフトウェア開発はフィードバックループであり、それを通して改善がなされる。
- 設計と実装のペアが繰り返され、再構築を経て最終の成果物になる。
- 設計の段階でユーザが使用時のオプションに関して関与できない。
- どの段階においても、外部仕様はユーザに入手可能なものを示し、内部仕様は外部仕様を実現するプログラム構造を示す。
- プロジェクトの初期の段階から最終段階まで外部仕様にかかわるものと内部仕様にかかわるものの間にはフィードバック系ができていなければならない。
- ソフトウェアシステムの構造として、一番上にアプリケーション、次いでミドルウェア、 そして、サービスルーチン及び制御プログラムとなっている。よ

【脚注】

※ 8　The Standish Group International, Chaos Manifesto 2013

り下位のものを変えることはより上位のものに影響を与えるので、維持に経費がかかる。

2.3.2 ソフトウェア・エンジニアリングの管理と方法論

- ソフトウェア設計の方法論は、プログラムとは何かの理解の下に、それが開発される手法、手順及び技術から構成されている。このことは、設計の方法論はソフトウェア・エンジニアリングの管理と密接な関係にあることを示す。
- プログラミングには依然としてアートの部分があるが故に、技術の側面を更に教育し、それを実践するように動機付けなければならない。
- ハードウェア及びオペレーティングシステムのリリースから独立したパッケージというものは存在し得ない。これは、システムエンジニアリングの範疇に入る。（既に、この時点でシステムズエンジニアリングではなく、システムエンジニアリングという用語が用いられている。）
- プログラム設計及び生産の管理のために、以下の項目を考察する必要がある。
  - 設計プロセス
  - 設計プロセスを実施するための組織
  - プログラムの文書化
  - 汎用コンピューター用のツールの開発

2.3.3 ソフトウェア・エンジニアリングにおける設計と実装

- 設計と実装が分けられているのは、担当する人が異なることによる。パフォーマンスは実装の影響が大きく、設計がパフォーマンスに責任を持つのであれば、実装者が何も判断せずに実装できるような設計がなされている必要がある。
- 製品の品質は設計による。
- 設計プロセスは繰り返えされるものであり、とくに大規模システムの場合、他の繰り返しで問題が分かったとしても自分の仕様を変更するのは締め切りの関係で困難な場合があり、0版、1版、N版というものが作られてしまうことになる。
- 一般的に実践されているものとして、先ず仕様が決められ、それに基づいて設計がなされ、更に実装、と進む。設計者が仕様の対象に関して無知の場合、何が起こる？

3．2001年5月発行のSWEBOK Trial Version

1993年5月にIEEEコンピューターソサイエティは、ソフトウェア・エンジニアリングを専門分野として確立するための実行委員会を設立することを決め、1993年8月にACMも同様な決定をした。同年9月から12月まで両者は合同会議を開き、1994年1月に両者は合同実行委員会を開くことで合意し、以下の分野において適切な基準を作ることにした。

- 要求される基礎知識体系と推奨されるプラクティスを定義する。
- 倫理と専門職として充足すべき基準を定義する。
- 学部、大学院及び生涯教育のためのカリキュラムを定義する。

1994年から1996年にかけて、上記の最初の項目のためのタスクフォースが設置された。1996年までに、タスクフォースは基礎知識体系を定義するにはとてつもない経費が必要とされることを認識し、過去40年にわたり開発され、展開されている基礎知識体系の一覧及びガイドとすることを決めた。この基礎知識体系そのものは静的なものではなく動的なものである。1993年から2000年にかけて、IEEEとACMは、Software Engineering Coordinating Committee（SWECC）を通して共同してソフトウェア・エンジニアリングを専門職分野とすることに協働してきた。

SWEBOK Trial Versionは、以下の目次にてカバーされている。

第1章　ガイドへの序説
第2章　ソフトウェア要求
第3章　ソフトウェア設計
第4章　ソフトウェア構築
第5章　ソフトウェアテスティング
第6章　ソフトウェア保守
第7章　ソフトウェア構成管理
第8章　ソフトウェアエンジニアリング・マネジメント
第9章　ソフトウェアエンジニアリングプロセス
第10章　ソフトウェアエンジニアリングのためのツールおよび方法
第11章　ソフトウェア品質

付録 A　ソフトウェアエンジニアリング基礎知識体系トライアル版のための知識領域記述のための仕様
付録 B　ソフトウェアエンジニアリング基礎知識体系トライアル版のための関連する規律のリスト
付録 C　ブルームの教育評価分類法によるトピックスのクラス分け
付録 D　コンポーネント統合知識分野のための構造の提案

第 1 章において、ソフトウェア・エンジニアリングの定義として IEEE コンピューターソサイエティ用語集[※9]から以下を採用している。

1）ソフトウェアの開発、運用及び保守に対して、システマチックでよく訓練された定量化可能なアプローチを適用すること、すなわち、エンジニアリングをソフトウェアに適用すること。
2）上記アプローチに関する研究。

本 Trial Version は、以下の大規模な査読及びコメントの産物である。

― 3 回の査読工程
― 42 カ国 500 人の査読者
― 9,000 件のコメント

4. 2005 年 6 月発行のソフトウェアエンジニアリング基礎知識体系－ SWEBOK 2004 －日本語訳版

本バージョンでは、以下の査読及びコメントが寄与した。

― 21 カ国から 124 人の査読者
― 1,024 件のコメント

以上は個人の Web 登録を経由するものであるが、加えて以下の関連する組織における査読及びコメントの提出があった。

― IEEE CS/ACM Computing Curricula Software Engineering
― IEEE CS Certified Software Development Professional プロジェクト
― IEEE Software Engineering Standards Committee
― American Society for Quality Software Division
― Canadian Council of Professional Engineers

結果として、Trial Version と 2004 年版は、内容において改善されたものはあるものの第 1 章から第 11 章までは同じ知識領域をカバーし、以降以下のように修正されている。

第12章　ソフトウェアエンジニアリングに関連するディシプリン
付録 A　SWEBOK の Ironman バージョンにおける知識領域記述のための仕様
付録 B　SWEBOK へのガイドの進化
付録 C　IEEE および ISO ソフトウェアエンジニアリング標準の SWEBOK 知識領域への割りつけ
付録 D　ブルームの教育評価分類法によるトピックスのクラス分け

5. 終わりに

インダストリー 4.0、Internet of Things（IoT）、System of Systems（SoS）、Cyber Physical System（CPS）という用語が広く語られるようになっている。これらのベースはソフトウェアにあり、その社会における重要性は非常に高いものになっている。今回 SWEBOK V3.0 を紹介する機会が与えられ、ソフトウェア・エンジニアリングという用語そのものがいかに世の中に出現したのか、そのオリジンにかかわる報告書を調査すると共に、SWEBOK そのものの進化もトレースした。1968 年にコンピューターにかかわった人々の問題意識は今日現在と同様なことに驚くと共に、ソフトウェア・エンジニアリングという専門分野を担当する専門職として切磋琢磨する必要性を感じている。SWEBOK の策定及びその維持プロジェクトにおいては、実践への考慮をしている。学問的基礎の下の実践が幅広く身につくためには 50 年という年数は短いのかも知れないが、ソフトウェアの社会への影響を考えた時にまだまだ時間が必要とするのではなく、積極的に日々のプロジェクトにソフトウェア・エンジニアリングの新しいプラクティスを導入するようにしたいものである。次回から、SWEBOK V3.0 の具体的な内容を紹介してゆく。

【脚注】
※ 9　IEEE Standard Glossary and Software Engineering Terminology, IEEE, 1990

YRPの概要と活動について

YRP 研究開発推進協会 事務局長
森下　浩行

1 YRP とその歴史

横須賀リサーチパークは、Yokosuka Research Park と表記した際の頭文字から通称「YRP」と呼ばれている。地理的には、横須賀市の中心部から南側の郊外にあり、東京湾を見下ろす丘の上に位置している。電波技術など ICT（情報通信技術）に特化したリサーチパークであることが特徴で、約 60ha の敷地に 50 余りの企業や機関が入居しており、昼間人口は約５千人となっている。

図１の地図中、①は株式会社横須賀テレコムリサーチパークが所有する「YRP センター１番館」であり、②～⑤はこの地域のディベロッパーである京浜急行電鉄株式会社が所有するテナントビルで、「YRP センター２番館」、「YRP ３番館」、「YRP ベンチャー棟（４番館）」及び「YRP ５番館」となっている。⑥～⑯の建物は、京浜急行電鉄株式会社から進出企業が敷地を買い取り自社でビルを建てたところで「独立研究棟」と呼ばれている。

発足に至る経緯としては、1986 年、土地を所有している京浜急行電鉄株式会社がこの地域の開発を検討し、横須賀市に相談したことが構想の発端となった。郵政省関東電気通信監理局（現総務省関東総合通信局）を中心とした連絡会が設けられ、1988 年、「横須賀リサーチパー

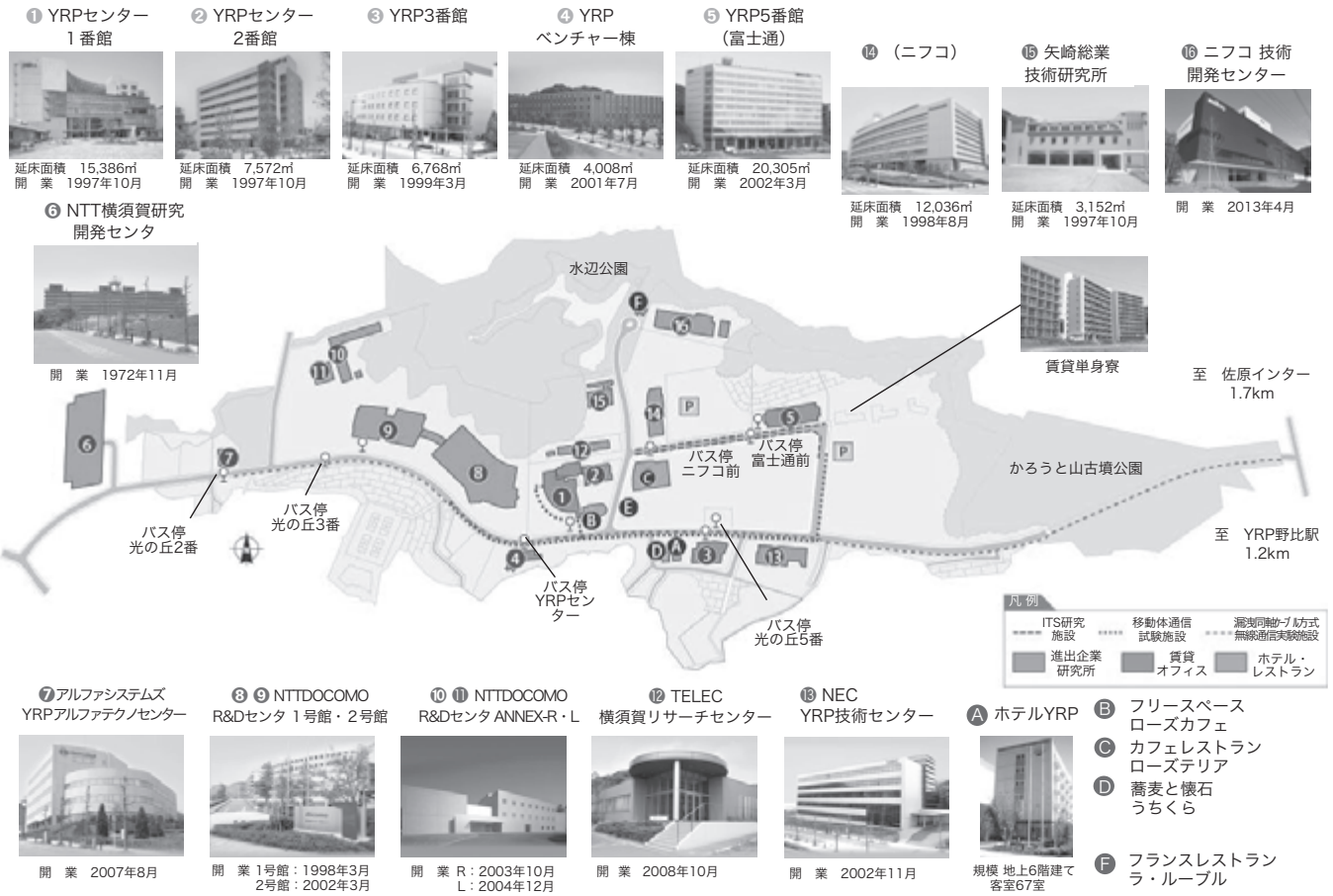


図１　YRP の全体図

ク構想」が発表された。その後、電波・移動体通信技術が中心テーマとして選定され、1997 年 10 月にオープンした。当初 YRP で最大規模の研究開発が行われたのは第三世代（3G）携帯電話の開発であり、その成果として、2001 年に株式会社 NTT ドコモが FOMA という名称で 3G サービスを開始している。

2 YRP の運営主体と役割分担

① YRP 研究開発推進協会
② 株式会社横須賀テレコムリサーチパーク
③ 京浜急行電鉄株式会社
④ NICT（情報通信研究機構）
⑤ 総務省と横須賀市

YRP では関連する企業が連携・協力して研究開発活動を行っており、こうした活動を支援するために組織されたのが①の YRP 研究開発推進協会（以下、「協会」という。）である。協会は約 160 の会員から成る非営利の任意団体で、会員の年会費により運営されている。②の株式会社横須賀テレコムリサーチパークは、横須賀市、株式会社日本政策投資銀行（旧日本開発銀行）、京浜急行電鉄株式会社、神奈川県などの出資を受けた第三セクターの会社である。同社は、YRP センター 1 番館の所有者として、テナントスペースの貸出、会議室など共用施設の貸出のほか、テストベッドの運営や各種研修も行っている。③の京浜急行電鉄株式会社は、2 番館～5 番館を所有して入居企業に貸し出しているほか、レストランやホテルなどの運営を行っている。④は独立行政法人である NICT（情報通信研究機構）で、YRP における研究開発の中心的な役割を担っている。⑤が政策的支援・財政支援を行っている総務省と横須賀市である。

3 YRP が目指す方向性

YRP では、1997 年の設立から 5 年毎に長期ビジョンを策定し、活動の指針としている。第 4 期ビジョンは

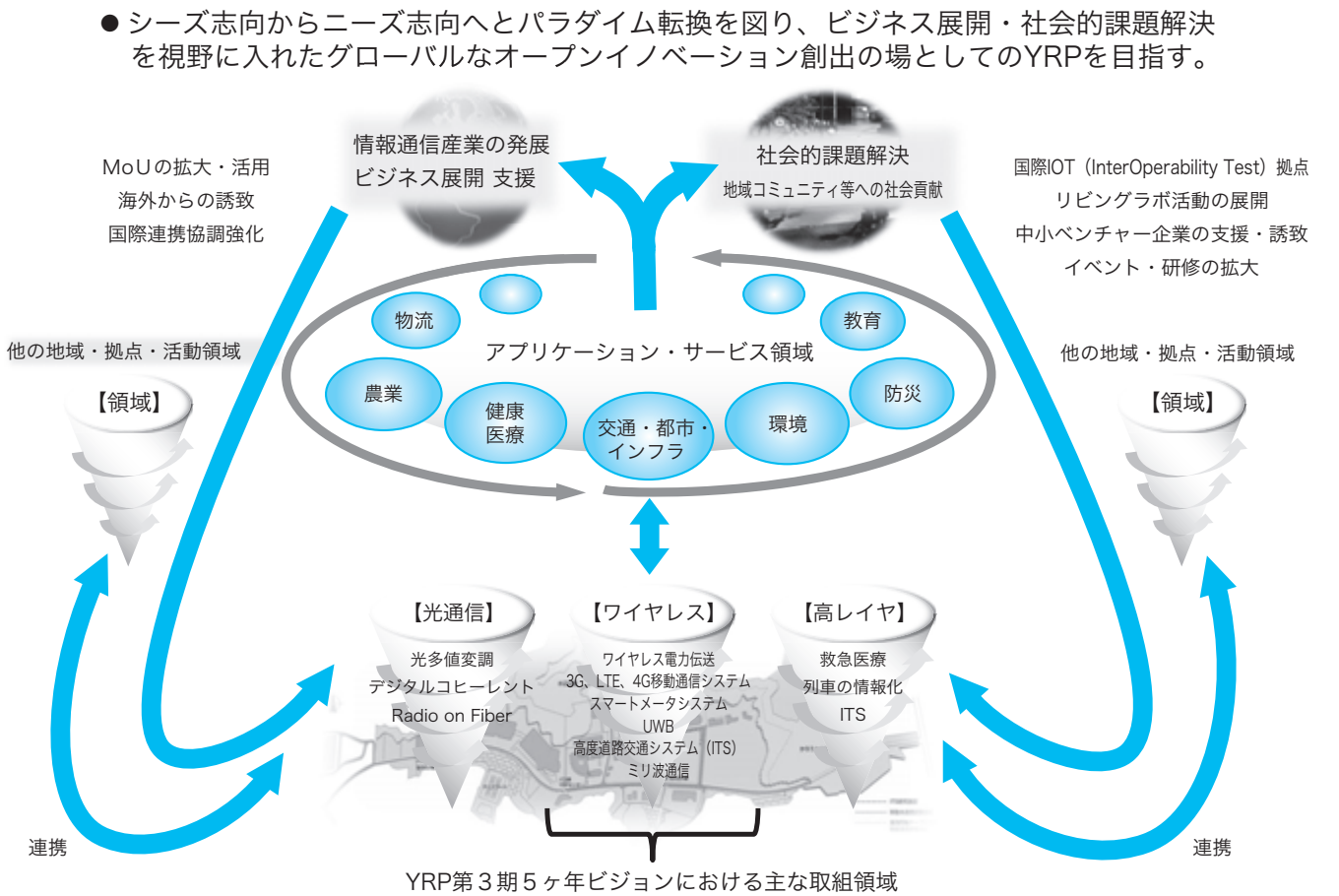


図２　第４期ビジョン

2012 ～ 2017 年までの期間を対象としている。

第 4 期ビジョンの特徴は次の通りである。まず、従来から取り組んでいるワイヤレス（無線）分野のみならず、隣接する NTT 横須賀研究開発センターが実施している光通信分野との融合をはじめ、YRP に結集する幅広い分野の機関と連携しながら研究開発を推進すること、２点目は、研究開発の成果の製品やサービスへの具現化という観点を重視し、社会的ニーズやユーザニーズの視点からの研究開発を進めること、３点目は、プラットフォーム分野のみならず、アプリケーションやコンテンツ分野を得意とする機関との積極的な連携を進めることである。

4 YRP の具体的活動

(a) テストベッド

YRP がそのオープン以来、重視して取り組んできたことのひとつがテストベッドの充実と活用である。費用面や運用面から企業が一社単独では所有することが難しい実験施設や最先端の研究施設を YRP が整備し、その利用を促してきている。

(b) 議論の場

第２点目として、研究開発に関する議論を行うための場作りである。現在、最も活発に実施している活動分野は、「ワイヤレス電力伝送」と「Wi-SUN（ワイヤレス・スマート・ユーティリティ・ネットワーク）」である。

ワイヤレス電力伝送は、2009 年に協会内に設立された BWF（ブロードバンド・ワイヤレス・フォーラム）の場で議論されている。ワイヤレス電力伝送は、無線を使って情報通信機器、家電製品や電気自動車（EV）などに電力を供給するものである。BWF では、総務省における国内制度化や標準化に寄与しているほか、国際標準化活動へ貢献するため、CJK（中国、日本及び韓国）の IT 標準化会議（CJK IT Standards Meeting）、アジア・太平洋電気通信共同体（Asia-Pacific Telecommunity）の AWG（APT Wireless Group）、国際電気通信連合無線通信部門（ITU-R）

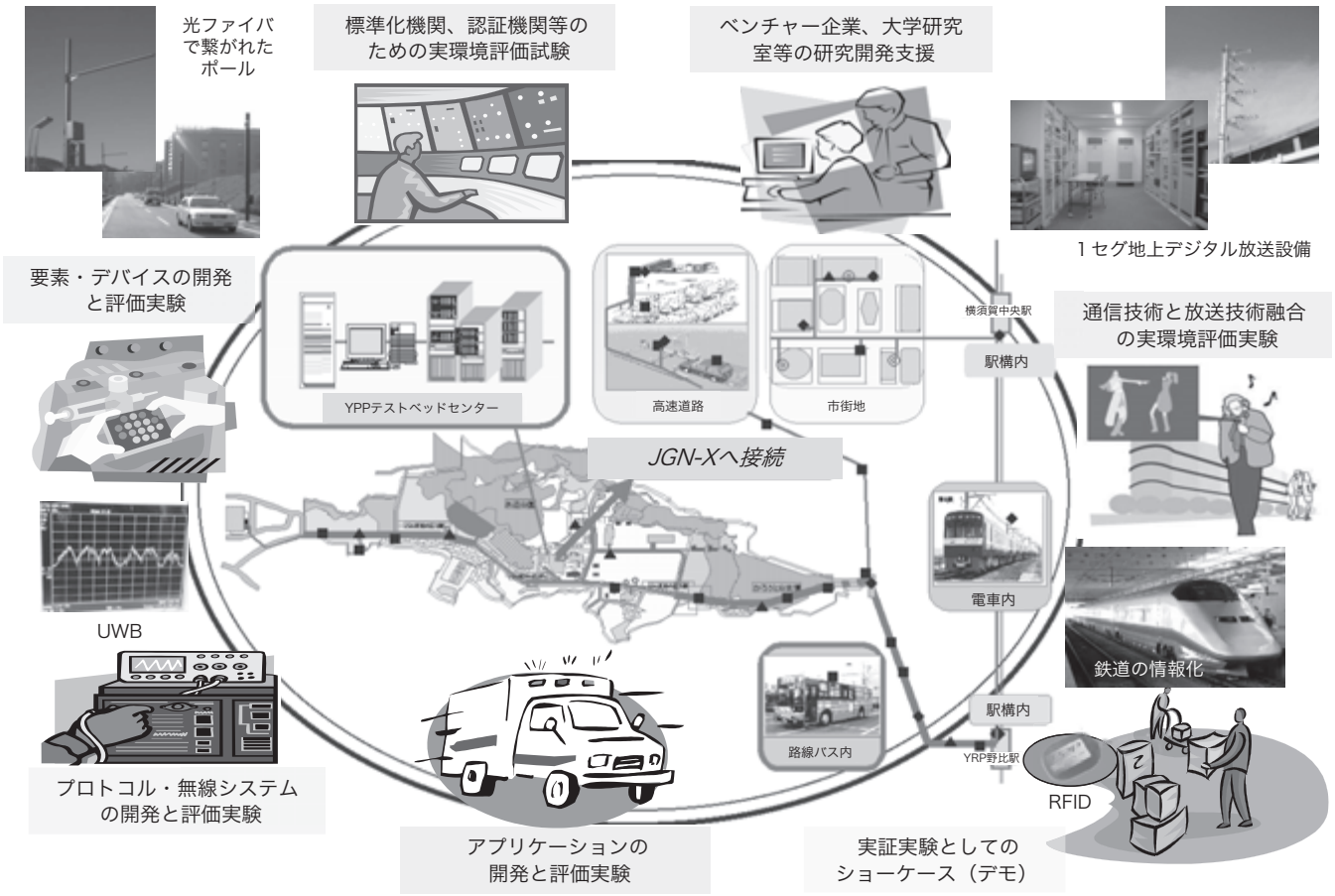


図3　テストベッド

などへの提案・調整を行っている。

Wi-SUN については、昨年５月、「ワイヤレススマートユーティリティネットワーク利用促進協議会」が協会内に設立された。この協議会は、NICT が開発した Wi-SUN の利用を推進するために設立されたものである。Wi-SUN は、スマートメータによる電力管理などエネルギーマネジメントのほか、交通インフラ、農業、防災などの幅広い分野で、センサと機器を結ぶ用途での利用が期待されている。

(c) 国際展開

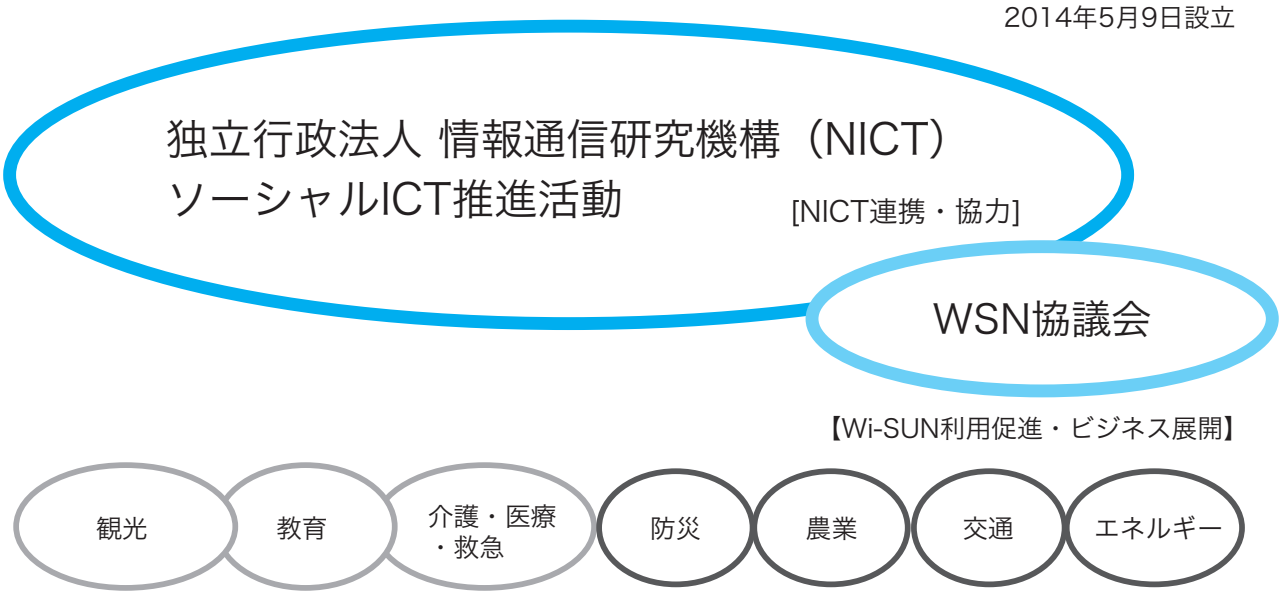
第３点目として、国際的なネットワーク作りが挙げられる。YRP のオープン当初から、国内のサイエンスパークなどに加え、欧州やアジアの研究機関などと研究交流を進めるための覚書（MOU）を締結し、イベントの実施や研修生の受け入れ、共同実験の推進などを図ってきた。

また、アジア諸国との協力を進めるため、YRP は、2000 年にアジア太平洋電気通信共同体（APT）へ賛助加盟員として参加し、その活動の一環として APT 加盟国の主管庁、通信事業者などを対象とした移動体通信研修を毎年、日本で実施している。

更に、中国に関しても 2001 年に日・中移動体通信技術フォーラムを設け、当時交流の無かった政府機関、地方政府、中国の通信事業者やメーカとの橋渡しを行い、2004 年には、これを日･中 ICT 技術フォーラムと改称し、今日に至っている。

5 おわりに

YRP の開設から 17 年が経過し、携帯電話やスマホの普及に代表されるように、当時では全く想像ができなかった電波利用や移動体通信が広がっており、その進歩は今なお続いている。このような社会の要請に応えるため、YRP においても企業や研究機関などによる研究開発活動が続けられている。



ワイヤレススマートユーティリティ利用促進協議会（WSN協議会）は、Wi-SUN技術の利用を促進し、
我が国のICT産業の強化と国際的な研究開発連携の推進を図ることを目的に設立し、次のような活動を行います。

① NICTのソーシャルICT推進活動と連携した活動を行うとともに、会員のビジネス推進の活動を行う。
② Wi-SUNの利用促進分野は、NICTが構築した農業、防災、エネルギー、交通の実証環境に加え、他の社会・産業分野にも拡大する。
③ YRPをWi-SUN利用促進活動の拠点として形成していくための活動を行う。

図４　WSN 協議会

Column

IoT 時代のグローバル競争

IPA 顧問、学校法人・専門学校 HAL 東京 校長
鶴保　征城

NTT ドコモが、様々な機器・装置に通信モジュールを組み込み、データ通信を行う M2M（マシン to マシン）、いわゆるマシンコミュニケーションを発表したのは 2000 年代前半だ。この考え方は、「物にもペットにも無線がつく時代」を先取りしたものであり、今や、M2M 回線は世界で 2.5 億回線、日本で 1,500 万回線に達している。

家電業界で世界最大の見本市は、毎年米国ラスベガスで開催される CES だ。この数年はサムスン電子などのアジア勢の躍進が目立っていたが、今年は「IoT（Internet of Things）」と呼ばれるネット技術が注目され、欧米の自動車メーカ、半導体メーカ、ベンチャー企業が展示の主流になっていたようだ。

CES で注目される基調講演でも、独ダイムラーのディーター・ツェッチェ、米フォード・モーターのマーク・フィールズ、米インテルのブライアン・クルザニッチの各社会長、CEO が登壇している。

これは正に、家電の潮流が従来型の家電製品から IoT にシフトしていることを示している。IoT は、M2M とほぼ同義語で、様々なモノや装置がインターネットにつながり、新しいネット端末になっていくということだ。自動車も例外ではない。

日本勢はどうかというと、ソニーやパナソニックが例年通り大きなブースを構えていたが、展示内容は薄型テレビや携帯端末、カメラといった従来型の家電製品が中心で、IoT を意識した商品はあまり見られなかったとのことだ。

更に、米国は IoT の流れを加速するために、政府が 2012 年、「Big Data R&D Initiative」を発表し、総額 2 億ドル以上の拠出を決めた。民間では 2014 年、GE、IBM、シスコ、インテル等が「Industrial Internet Initiative」を設立し、既に 70 社近くの企業が参加している。Industrial Internet は IoT と同義語だが、産業界の工場や現場の意識が強い。

とくに、GE はこれに社運をかけていると言っても過言ではないほどの力の入れ方だ。同社の主力製品である発電機、ジェットエンジン、掘削機械、機関車などをインターネット接続し、機械が生み出す膨大なデータを取得・分析することによって、発電所、航空、鉄道などの効率と信頼性を向上させている。

恩恵を受ける業界は、石油・ガス、電力、鉄道、ヘルスケア、航空など多岐にわたるが、石油・ガス業界は燃料効率などが 1 %向上しただけで、数百億米ドルの効果がもたらされると言われている。

一方、欧州に目を転じると、ドイツが 2011 年に、開発・製造・流通プロセスを IoT により全体最適化する「Industrie 4.0」を開始している。こちらはより明確に、第 4 次産業革命を意図している。ちなみに、第 1 次（18 世紀後半）は蒸気機関による機械化、第 2 次（20 世紀初頭）は電力の活用、第 3 次（1980 年代以降）はコンピュータによる自動化である。

Industrie 4.0 の狙いは、デジタル化とネットワーク化を駆使して、スマート工場を実現することと、工場間や企業間に横串を通すことである。

スマート工場は従来の工場を、決められた工程を経て生産するライン生産から、生産単位を自在に組み変えるセル生産に変貌させるものである。また、工場・企業間に横串を通すのは、輸出総額の 30%（日本は 8 %）を占める独立独歩の優秀な中小企業を連携させて、今後必須となる全体最適をやりやすくするためである。

米国の Industrial Internet がクラウドの活用を前提とし、進化した AI の集中を想定しているのに対して、ドイツの Industrie 4.0 は既存の企業や工場を自在につなぐことで自律分散型のシステムを狙っている。このあたりに今後の製造業の覇権争いが垣間見えるように思う。

米国、ドイツ共に国運を賭けて、インターネットと IT を融合させた新しい産業の創出に全力を挙げているが、日本も当然遅れてはならない。2014 年、国会議員有志によって組込みイノベーション議員連盟（会長 河村建夫衆議院議員）が、また民間では一般社団法人組込みイノベーション協議会（理事長 筆者）が設立された。

プラットフォーム競争やモバイル機器での敗退、変革に対応できない組織、イノベーション創出の貧困さ、専門人材不足等々の諸問題はあるが、これまでのモノ作り大国の蓄積を活かして、グローバル競争に打ち勝たなければならない。

サービスサイエンスによる顧客共創型 IT ビジネス

諏訪良武、山本政樹　著

ISBN: 978-4798141411
翔泳社刊
四六版・218 頁
定価 2,000 円（税抜）
2015 年 1 月 27 日刊

システムは計画どおり稼働したが、次も同じベンダに頼むかどうかはわからないね

世間で一般的に用いられているプロジェクトの成功の定義は、計画した Q（uality）、C（ost）、D（elivery）の値を、Q に関しては出荷後のあらかじめ決めた期間におけるバグの数、C は出荷時までにかかった経費、D は出荷日時をあらかじめ開発ベンダとユーザ企業で合意した上で、それらを全て満足していることとしている。これら値をクリアしているにもかかわらず冒頭の発言がユーザ企業から出たとするとプロジェクト成功への今までの定義を考え直さなければならないであろう。

本書第 3 章は、「システム開発の顧客満足の鍵を握るプロセス品質とは」であり、「システム開発サービスは、サービス業の特徴を全て満たしている」としている。とすれば、プロジェクト評価に当たり、「サービスとは何か」をシステム開発に携わる者は十分理解しておく必要があろう。第 1 章では「実践されているサービスサイエンスを理解する」を説明している。ここで、「サービスとは、人や構造物が発揮する機能でユーザーの事前期待に適合する」としている。このことは、サービスを考えるに当たりユーザ（お客様）を理解できていなければならないことを示し、第 4 章は、「システム開発のお客様を明確に定義する」を説明している。

本書は「サービス」に関する新しい知見を上述のように提供してくれるのみならず、第 5 章にて、「プロセス品質を高めるために」として、具体的な方策を提示してくれる。最終章においては、開発ベンダとユーザ企業（お客様）との間で「顧客共創型サービスモデルを実現するために」として提案をしている。

（新谷　勝利）

一般システム思考入門

ジェラルド・M・ワインバーグ　著
増田伸爾　訳

ISBN: 978-4314002547
紀伊国屋書店刊
四六版・342 頁
定価 3,592 円（税抜）
1979 年 6 月 20 日刊

システムとは物の見方である

今年の 1 月に IPA と JAXA 共催で 12thWOCS[2]（The 12th Workshop on Critical Software Systems）が開かれ、基調講演はエアロスペース社の James N. Martin 氏による「総体的な解決策を見つけるための全体的アプローチ：システム原則と概念の活用」と題するものであった。

マーティン氏は、システムというものがしばしば間違って「システム＝部分（Parts）の集合」と認識されているが、システムは PICARD（Parts、Interactions、Context、Actions、Relations,Destinations）から成ることを分かりやすく説明された。

この PICARD によるシステム思考では、「システムとは世界をどう見るかということである」と看破され、この考え方は本書から会得したとのことであった。もう何十年も前に読んだ記憶があり、帰宅後探したが見つからなかったが、インターネットで発注すると翌日には配達され、早速読み始めた。便利な世の中になったものである。

ワインバーグ博士は、この本は、「私のあらゆる“システム”問題に対する、最初の考え方、アイディア、ヒントの寄せ集めであり、それらに取り組む際の、読者の最初の考え方に、何らかの助けになろうとするものである」と“はじめ”に述べられている。書評者は、最初の数ページを読み、次いで「第 3 章 システムと錯覚」を読み、本書評のタイトルにした「システムとは物の見方である」を読み、マーティン氏が WOCS[2] で言及されたことを確認した。次いで「第 4 章 観測結果の解釈」、「第 5 章 観測結果の分解」とつながる。この本は気の向いたところから読み直したいものである。

（新谷　勝利）

編集後記

SEC journal 40 号をお届けします。まずは、前号にて速報でお知らせいたしました 2014 年 SECjournal 論文賞の授賞式が、去る 1 月 21 日に WOCS[2] の会場にて行われました。あいにくの雪にもかかわらず多くの方のご来場を受け、晴れやかな式となりました。受賞者の皆様、あらためておめでとうございます。その WOCS[2] の講演で来日された INCOSE のジェイムズ・マーティン氏との所長対談は、システムズエンジニアリングという難しいテーマにもかかわらず興味深いお話を十分に伺え、予想外の長編となりました。今正に各国で IoT 環境の利活用に取り組んでいる中で、ますます重要性を増すシステムズエンジニアリングの話は、絶好のタイミングだと思います。コラムでも述べられているように、世界のスピードに負けない日本の技術力の発揮が期待されます。

（編集長）

編集部より

次世代のソフトウェア・エンジニアリングに関して等、忌憚のないご意見をお待ちしております。下記の FAX またはメールにてお気軽にお寄せください。

SEC journal 編集部　　FAX：03-5978-7517　　e-mail：sec-journal_customer@ipa.go.jp

SEC journal　編集委員会

春のきざし　　（撮影：K.Hasegawa）

SEC journal　第 10 巻第 6 号（通算 43 号）　2015 年 3 月 1 日発行

編集兼発行人　独立行政法人情報処理推進機構
技術本部　ソフトウェア高信頼化センター
所長　松本隆明
〒 113-6591　東京都文京区本駒込 2-28-8　文京グリーンコート センターオフィス 16 階
Tel：03-5978-7543　Fax：03-5978-7517
URL：http://www.ipa.go.jp/sec/
e-mail：sec-journal_customer@ipa.go.jp

SEC journal 論文募集

独立行政法人情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センターでは、下記の内容で論文を募集しています。

論文テーマ

・ソフトウェア開発現場のソフトウェア・エンジニアリングをメインテーマとした実証論文または先導的な論文
・ソフトウェアが経済社会にもたらす革新的効果に関する実証論文

論文分野

品質向上・高品質化技術、レビュー・インスペクション手法、コーディング手法、テスト／検証技術、要求獲得・分析技術、ユーザビリティ技術、プロジェクト・マネジメント技術、設計手法・設計言語、支援ツール・開発環境、技術者スキル標準、キャリア開発、技術者教育、人材育成、組織経営、イノベーション

応募要項

締切り　：1月・4月・7月・11月　各月末日
査読結果：締切り後、約1カ月で通知。「採録」と判定された論文は SEC journal に掲載されます。
応募方法：投稿は随時受付けております。応募様式など詳しくは HP をご覧ください。
http://www.ipa.go.jp/sec/secjournal/papers.html

SEC journal 論文賞

毎年「採録」された論文を対象に審査し、優秀論文には SECjournal 論文賞として最優秀賞、優秀賞、所長賞を副賞と併せて贈呈します。

---

iパス ITパスポート試験 のご案内

ー ビジネスにＩＴを活用する すべての社会人のための「国家試験」ー

- ビジネスにＩＴを利活用するためには、情報システム部門に限らず、利用する側の社員一人ひとりにも“ＩＴ力”が求められています。
- ｉパス（ＩＴパスポート試験）は、セキュリティ、ネットワーク等のＩＴに関する基礎知識をはじめ、企業活動、経営戦略、会計や法務、プロジェクトマネジメントなど、幅広い総合的知識を測る国家試験です。
- ｉパスを通じて、社員一人ひとりに“ＩＴ力”が備わることにより、組織全体の“ＩＴ力”が向上し、様々なメリットが期待されます。

ｉパスのメリット

ＩＴを活用した業務効率化とビジネス拡大に！

ｉパスを通じて習得したＩＴの基礎知識を活かすことで、業務にＩＴを積極的に活用し、業務効率化につながります。また、ＩＴに関する基礎知識は、社内の情報システム部門等との円滑なコミュニケーションにも役立ちます。営業職であれば、顧客に対して製品やサービスを具体的にわかりやすく説明できるようになり、顧客のニーズをより深く把握できるようになり、ビジネスチャンスの拡大にもつながります。

情報セキュリティ対策・コンプライアンス強化に！

社員一人ひとりが、情報セキュリティやモラルに関する正しい知識を身につけ、意識することで、情報セキュリティに関する被害を未然に防ぐことができ、「情報漏えい」などのリスク軽減、企業内のコンプライアンス向上・法令順守に貢献します。

経営全般に関する知識など幅広い知識がバランスよく習得できる！

ｉパスは、ＩＴに関する知識にとどまらず、企業活動、経営戦略、会計や法令など、ＩＴを活用する上で前提となる幅広い知識がバランスよく習得できます。そうした知識が身につくことにより、業務の課題把握と、ＩＴを活用した課題解決力が備わり、組織全体の業務改善につながります。

詳しくは、ｉパス Web サイトをご覧ください。https://www3.jitec.ipa.go.jp/JitesCbt/index.html
※企業の活用事例、企業の声、合格者の声など魅力的なコンテンツがご覧になれます。

IPA Better Life with IT

SEC journal No.40
第 10 巻第 6 号（通巻 43 号）
2015 年 3 月 1 日発行

ISSN 1349-8622

VEGETABLE OIL INK

R70 古紙パルプ配合率70％再生紙を使用

リサイクル適性Ⓐ この印刷物は、印刷用の紙へリサイクルできます。

独立行政法人情報処理推進機構