2017年7月1日発行
第13巻 第1号(通巻52号)
ISSN 1349-8622

SEC journal 49

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

SEC journal No.49
Vol.13 No.1 Jul. 2017

巻頭言

国際規格SQuaREシリーズの活用促進

藤井 洋一
一般社団法人 IT検証産業協会（IVIA） 会長

IVIAについて －IVIA10年の歩みと成果を振り返る－

一般社団法人 IT検証産業協会（以下IVIA）は、「IT検証事業の産業化」を目指して、2005年に設立された団体です。ソフトウェア開発工程の一つであるテスト（中でもシステムテスト／総合テストと呼ばれることの多い、開発内の最終段階のテスト）・検証を専門分野として請け負うことで、開発成果物の第三者視点による検証を通じてソフトウェアの品質確保に貢献してきました。

当時、ソフトウェアは高機能化（複雑化）・大規模化、短納期化の波にさらされていました。第三者検証が事業として成り立つのもそれが大きい要因だったと言えるでしょう。IVIAに求められる検証としては、機能の検証（機能テスト）が主でした。我々は業界団体としてテスト技術やテスト手法を研究して『IT検証標準工法ガイド』を作成するなど、テスト技術の標準化に取り組み、一定の評価を得てきました。

ソフトウェアとテストをとりまく現状と、IVIAの課題

この10年の間に、ソフトウェアを取り巻く状況は更に激しい変化を見せてきています。ソフトウェアの提供形態はオンプレミスからWebアプリケーションやクラウド化にシフトしてきました。他のベンダのサービスを呼び出すこともごく普通に行われるようになってきています。また、インターネットを介してあらゆるものをつなげようという“IoT”が加速度を増しています。

「他のシステム、他の製品とつながる」ことが当然の前提となっている世界では、どの製品とつながるのか／つながらないのか、想定外の製品／システムと接続した場合に問題は生じないか、ということが十分に確認されなければなりません。デバイスやハードウェアが故障や誤動作を起こしたとき、接続先のシステムに影響を及ぼさないかも重要です。また、システムとして悪意のある第三者の脅威に十分備えていることも確認が必要です。

このような、ソフトウェアや製品／システムが提供する機能以外の「非機能要求」に対してはそれを確認する非機能テストが必要ですが、その重要性はこれまで以上に増すと考えられます。こうした“非機能のテスト”のあり方について提言していくことがIVIAの当面の課題です。

非機能要求のテストを考える際のよりどころになるのがISO/IEC 25000シリーズ（通称SQuaRE）です。ソフトウェア品質を体系的に考える枠組みであるソフトウェア品質モデルを規定したISO/IEC 25010を中核として、品質要求の考え方（2503n）、品質の測定（2502n）・評価（2504n）などと、ソフトウェア品質の要求から評価までを包括した規格となっています。こうした国際規格を手がかりに、非機能のテストのあり方を研究・実践していき、成果を『IT検証標準工法ガイド』に取り込んでいけたらと考えています。

IVIAの今後とIPAへの期待

ソフトウェアの品質を考える場合、「仕様通りに振る舞う」というだけではだめで、「そもそも要求されていることに適っている、要求を満たす」ということを確認することが求められます。品質の良しあしの感覚にとっては実際に（動かしたときに）、顧客や利用者の期待や要求を満たすかどうかが重要だからです。前者を“検証（verification）”と言うのに対し、後者を“妥当性確認（validation）”と言い、両者を併せて行うことがソフトウェアや製品の品質保証にとって大切なことです。妥当性確認には、SQuaREにおける品質の測定（2502n）や評価（2503n）が役に立つでしょう。

IVIAの“V”はVerificationのVですが、ValidationのVでも顧客に貢献し、ソフトウェア品質保証のお手伝いをできるようになっていきたいと考えています。

IPA/SECが編纂した『つながる世界のソフトウェア品質ガイド』は、SQuaREの普及促進にとって大きな意義があったと思います。高品質ソフトウェア製品の開発のために、今後も研究・実践両面でご支援をお願いしたいと考えています。

IoT時代に求められるレジリエンスエンジニアリングの考え方

南デンマーク大学教授
リンショピン大学（スウェーデン）名誉教授
エリック・ホルナゲル

×

IPA/SEC所長
松本 隆明

IoT時代においては、開発時には想定もできなかったモノが相互につながって高度なサービスやシステムを提供するケースが増える。こうした不確定で複雑なシステムの安全性を確保するためには、不測の事態を考慮し、運用も含めた柔軟な発想でシステムの設計を行うことが求められる。その方法論として注目を集めているのが、レジリエンスエンジニアリングとそれに基づく安全の概念であるSafety-Ⅱという考え方だ。提唱者であるエリック・ホルナゲル氏にお話を伺った。

システムの構造ではなく機能に着目する

松本　IoT時代と言われる今、様々なものがダイナミックにつながって、システムやサービスを構成するようになってきました。こうした不確定で複雑なシステムの安全性を設計するためには、画一的な方法では対応しきれず、より柔軟な設計や運用が必要になってきています。そうした意味で、レジリエンスエンジニアリングという考え方は、現在に非常にマッチした考え方と言えるのではないかと思います。

最初に、レジリエンスエンジニアリングの考え方を簡単にご紹介いただけますでしょうか。

ホルナゲル　短く申し上げるのは難しいのですが、あえて申し上げれば、安全に対するほかのアプローチとの違い、ということでお話をするのが良いかと思います。

レジリエンスエンジニアリングは、システムの機能に対してその設計を考えていくものです。システムの構造と機能を区別して考え、構造よりも機能に主眼を置いていきます。そして、システムがうまく機能した場合、どのように機能するのか、それを理解していくというものです。

長きにわたり伝統的にうまくいってきたやり方、これを仮に従来型と呼びますが、この従来型の考え方との対比で言えば、Safety-Ⅰ、そしてSafety-Ⅱという考え方を取っています。安全に対する従来型の考え方は、何か問題が起こらないように、それを予防していくというものが一般的でした。

一方、レジリエンスエンジニアリングにおいては、システムをどううまく機能させ続けていくのか、というところに注目します。

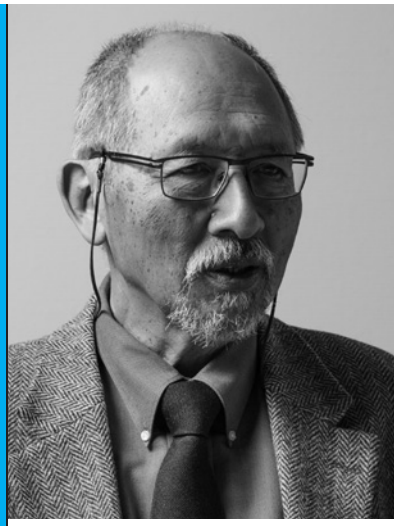
もう一つ、このレジリエンスエンジニアリングで注目している点は、複雑さを増したシステムがどんどん増えてきているこの世界の中では、それぞれのコンポーネント、構成要素がそれぞれ機能するところを見ていくだけでは、システム全体の安全性が担保できない、ということです。システムが全体としてどのように機能していくのか、これを見ていかなければなりません。複雑性が高まるシステムの中では、それぞれのパーツ、パーツがどのように相互作用をするのか、というところを理解していく必要があります。

松本　システム全体で考える場合には、人間もシステムの一つのコンポーネントとして考えるのでしょうか。

ホルナゲル　従来型ではそうであったと言えると思います。人をコンポーネントとして見ていました。

それはなぜかと言えば、やはりヒューマンマシンとかヒューマンコンピューターのシステムというものが出てくる随分前から、安全に対しての懸念が部分部分を見ていくということであったからです。

現在は電車であるとか発電所であるとかが、システムとして存在します。以前は、それぞれの構成要素単位で信頼性があって、



エリック・ホルナゲル

Ph.D. 南デンマーク大学教授、リンショピン大学（スウェーデン）名誉教授。産業安全、レジリエンスエンジニアリング、患者安全、事故調査、大規模社会技術システムを専門領域とし、大学、研究所、産業界において、原子力発電、宇宙・航空、ソフトウェア工学、地上交通、ヘルスケアなどにおける課題に取り組んできた。著者、編者として20冊の書籍を出版し、そのうち4冊がレジリエンスエンジニアリングに関するものである。

うまく機能していれば、全体的に安全を考えることができました。

従来型のやり方について少し変遷をお話すると、かつては、あるシステムのそれぞれの構成要素が機能するというところから、システム全体の理解を図るということが成り立っていました。システムが何か障害を起こしたということは、そのシステムを構成するコンポーネント、構成要素のどこかが障害を起こしている、という考え方ができたわけです。

それでかつてはうまくいっていた。しかし、1970年代に入って初めてコンポーネントだけでは原因が分からないような事例が出てきました。かつての、構成要素だけを考えるやり方では、事故や状況の原因が説明できなかったのです。

そこで、システムにはもう一つ中心的なコンポーネントである、人という要素があるということに気づいたのです。それがかつての考え方であるコンポーネント、構成要素に加えて、考えなければならない側面であるということが分かってきました。

そしてその後、更なるコンポーネント、構成要素が出てきました。それが、ソフトウェアです。最初は、ソフトウェアを構成要素として考える際に、あたかも機械的なシステムのように捉えて考えようとしました。最初の頃のソフトウェアはシンプルであり、構造もきちんとしている。そして、論理的な接続しか持たないというものでした。しかし、そのすぐ後に、機械的なシステムと同様に考えるのではあまりにも複雑性が高く理解ができない、ということが分かってきたのです。そこは、SECにおいても大きな関心事になっているところだと思います。

レジリエンスエンジニアリングは何のためのものか

松本　正に私たちも、今までの考え方では安全性の確保は難しくなってきていると感じています。とくに、ソフトウェアもそうですが、それ以外に本当にダイナミックに変わっていく時代になってくると、単一的なアーキテクチャベースの安全性設計ではもう対応できなくなってきた、と考えています。

その意味では、レジリエンスエンジニアリングで先生が提唱されている考え方というのは、開発の方法、システムの設計の方法論なのでしょうか、それとも、安全性解析のやり方というように捉えたほうが良いのでしょうか。あるいは、もっと幅広い概念と受け止めるべきでしょうか。

ホルナゲル　レジリエンスエンジニアリングには、幾つかの考え方が含まれています。

ある意味では、これは哲学であり、考え方やものの見方だと言えると思います。どのようにシステムが機能するのかということの捉え方、というふうにも言えるでしょう。それが、Safety-Ⅰ、Safety-Ⅱという言葉で表現されています。

そして、システムがどのように不具合を起こすのか、障害を起こすのか、という見方ではなく、どのようにシステムがうまく機能しているのか、というところを理解しようとするものです。

もう一方で、このレジリエンスエンジニアリングというのは、障害が起こった際の、何が起こったのかという分析のアプローチでもあります。

これまでの歴史の中で、どういう形でレジリエンスエンジニアリングが始まってきたのかと言いますと、これまでは安全にかかわる現場の仕事をしている人たちが、何らかの事故なり問題なりが起こった場合にそれを分析してきました。しかし、分析の手法には満足がいっていなかった、ということがあります。従来型のやり方は、事故または問題の原因だけを見るというやり方だったのです。そして随分前から多くの人たちが、その分析手法は生産性が悪いということに気づいてきました。そこで、違った分析の考え方が必要だと分かり、異なる分析のし方、分析の視点として、レジリエンスエンジニアリングが提唱されるに至りました。

レジリエンスエンジニアリングは、ものの見方、捉え方であり、ある意味では、哲学、考え方でもあります。そして同時に、問題が起こった際の分析のアプローチであるとも言えます。

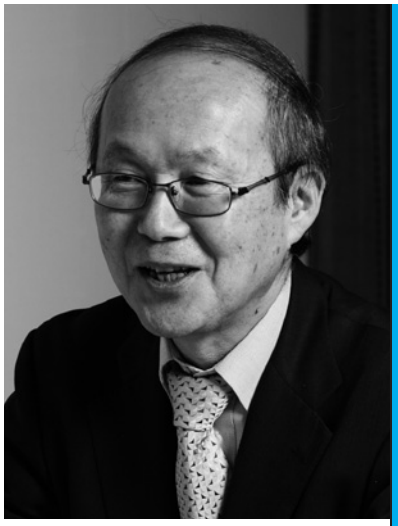
そして、それと共にデザインの考え方だとも言えると思います。どのように機能すべきかを考える、その考え方でもあります。現在では、このレジリエンスエンジニアリングが、システムのデザインの評価に使われるという例も出てきています。期待通りにシステムが機能するのか、そのデザインの部分を評価するためのものです。

失敗ではなくうまく機能していることを見る

松本　レジリエンスエンジニアリングにはウェイ・オブ・シンキングに基づくデザインと、それから分析と、二つ大きなポイントがあるということですね。とくに後者の安全性分析ということに関しては、実は私たちも非常に苦労しているところがあります。

何かと言うと、私たちも、実際に起きた障害を分析し、何が原因だったのかを解析し、それを教訓の形にまとめ、再発防止に役立てようという取り組みをしています。

ところが、実際に色々な企業から事故の事例を出してもらおうとしても、なかなか出してくれません。事故というのは、その企業にとってネガティブな面になってしまうので、積極的



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）。

にその情報を出そうとは思わないわけです。そのため、障害の事例に基づいて安全性を解析しようというのはなかなか難しい。むしろ先生が提唱されているような、うまくいっていることに着目するほうが、企業にとっても情報が出しやすいのではないかという気がするのです。

ホルナゲル　おっしゃる通りです。私どもも似たようなことを経験しています。やはり、マイナスのもの、悪いと思われるものを人は避けたい。こちらで取り組まれていることが、主にソフトウェアまたはITシステムだったとしても、どのようなシステムでも人が全く関与しないものは存在しません。どこかの段階で、必ず人は入ってきます。通常であれば、オペレーションの段階で人がかかわるという形になるでしょう。何か問題があった部分に注目されてしまうと、それにかかわった人たちが不快になったり、不安を感じたりしてしまいます。それに関する情報は明らかにしたくない、明らかにすることによって自分が罰せられるのではないか、という感覚さえ出てきてしまいます。

もう一つ、うまく機能しているところに焦点を当てるべき理由があります。それは、不具合が起きた際に、その理由が、何かが突然壊れたとか何かが突然異なってしまった、ということに限らないからです。ちょっとだけ通常とは違う小さなことによって、うまくいかなくなる、ということがよく起こるわけです。全体的な状態としては、うまくいっていたときとほぼ同じであるという中で障害が発生していることになります。そうであれば、うまくいっている状態を理解することが問題解決の基本になる、ということです。

更に三つ目のポイントとして言えることですが、障害、とくに重大な障害というものはめったに起こらない、まれだということです。めったに起こらないということは、そこから教訓を導き出し、学ぶことができる可能性が少ないということでもあります。一方、うまく稼働しているシステムは、常に稼働しているのですから、学べる可能性が大きいということになります。

単なる原因分析では起きていることが分からない

松本　レジリエンスエンジニアリングは、フィロソフィー、考え方だというお話が先ほどありました。それを支える具体的な方法論として、先生はRAGとFRAMというものを提唱されていますが、RAGというのは、どういうものでしょうか。

ホルナゲル　RAGというのは、Resilience Assessment Gridの略で、レジリエンス分析評価グリッドを示します。組織の中で、システムがどのように機能しているのかを測定し、モニタリング、監視をするための方法論です。

組織を管理したければ、必ず知らなければならない重要なことがあります。それは、その組織がどれくらいうまく機能しているのか、稼働しているのか、ということです。マネジメントをするためにはその状況を知らずにはできません。

これが機械であれば、センサやメジャーメントポイントを使うことによって、様々な測定ができます。自動車でも飛行機でも電車でも金融システムでも、そのような測定が可能です。

しかし、組織の中ではどうかと言いますと、そういう自然なメジャーメントポイントがありません。RAGとは、組織がレジリエンスという側面でどれだけうまく機能しているのかを測定する方法の提案です。組織の測定をすると言いましたが、ほかの組織と比較するための測定ではなく、また何らかの規格や参照するものと比較するための測定でもありません。その組織がどう変遷していくのか、進展していくのか、ということを比較する、自組織を相手として比較するための測定ということです。時間が経つにつれて、どう変わっていったのかを測定することによって、その組織が正しい方向に進んでいるかどうかを知り、それに基づいたマネジメントができるようになります。

他方、FRAMというのは、Functional Resonance Analysis Method、つまり機能共鳴分析法と言われるものです。4つの単語から成る略語ですが、中でも重要な言葉が、Functional Resonance、機能共鳴です。先程おっしゃっていた、世界がどんどん複雑性を増し、相互依存性、相互作用性が増しているというところに関係するものです。

相互依存性が高まり、相互作用が高まっている複雑な世界の中では、因果関係の分析や原因分析だけに依存することはできません。そこで重要になるのが、機能共鳴という考え方です。これは、因果関係の分析、原因分析を補完するという役目を持っています。これまでと違った形で起こる可能性がある場合、これまでの物理現象の中でも共鳴分析は行われてきましたが、ここでは想定外のことが突然起こり、そこに明確な原因がない、という場合にも分析できるものになります。

松本　FRAMの考え方は、分野に依存しないのでしょうか。FRAMのように、ダイナミックな世界のインタラクションに着目すると、分野に依存するような、例えば鉄道分野、あるいは電力分野、あるいは自動車というふうになってくると思うので、なかなか普遍的な方法にするのが難しいような気がするのですが。

ホルナゲル　FRAMの目的は、システムがどのように機能するのか、またはどのように活動が発生していくのか、という記述・説明を作っていくというものです。必要な機能がどのように相互に依存しているのか、どのように必要な機能がカバーされているのか、ということの記述・説明ということになりますから、その記述・説明、その方法論は、どのような活動にも適用することが可能です。

FRAMが使われた実際の使用例でご説明すると、病院における輸血で使われたことがあります。また、飛行機の事故やソフトウェアの会社がどのようにサーバーベースのサービスからクラウドベースのサービスに移行したのか、ということを記述することにも使われました。また、金融機関で資産運用のファンド・マネージャが、日常の取引でどのように投資をしたのかを記述することにも使われました。最近の例としては、ニュージーランドで森林における狩猟のために使われたというケースもありました。

松本　物理構成というか、アーキテクチャに依存しないで機能に注目するから、色々な分野に適用可能だと言えるでしょうか。

ホルナゲル　そうです。機能というところで何が起こっているかを説明しようとして着目するため、そのような適用が可能になります。機能がどのようにお互いを中継していくのか、というところにフォーカスをするから、とも言えるでしょう。

IPA/SECでは、とくに焦点を当てられるのがソフトウェアだと思います。ということであれば、オブジェクトベースのプログラミングに適用することにもつながると思います。

定量的な評価の前提には定性的な評価がある

松本　先ほどのRAGのお話にあった組織の評価という意味では、どちらかと言うと定性的な評価になるような気がします。つまり、今までのSafety-Ⅰの考え方であれば、障害に着目するので障害の件数がこれだけ減りました、という形で定量的な評価ができると思うのですが、RAGでは定性的になってしまうような気がします。

ホルナゲル　興味深いご質問です。簡単に言えば答えはイエスです。定量的と言うより定性的な評価になるということは言えるでしょう。ただ、科学者として一言添えれば、定性的な評価と定量的な評価の間に違いはないのです。定量的な評価も、その基本には、定性的な区別、違いがあると考えています。

松本　結果として定量的に表れるのであって、定性的なところをきちんと評価しないと、定量的な評価にもつながらない、ということですか。

ホルナゲル　おっしゃる通りです。まずは、何を記述、測定しようとするのか、何を評価しようとするのか、というところの記述(ディスクリプション)を理解することが必要です。例えば、測定対象が私の前にあるテーブルの幅だったとしましょう。それはセンチメートルで測定できたとしても、まず考え方として、幅の持つ意味、または測定のための評価尺度というものがあります。そういった定性的な側面というのは必ずあるわけです。

何かを測定して、どのように組織を経営し管理していくのか、という意思決定をすることがあるかもしれません。他方、何が起こっているのかということを説明したディスクリプションをベースに意思決定する場合もあります。数字よりも、ディスクリプションのほうが、より意味ある情報ということになるので、ディスクリプションによって意思決定をするほうが好まれるという状態にあります。

松本　適切な比喩ではないかもしれませんが、テーブルの幅といったときにも、そのテーブルを何に使うのか、オフィスのテーブルになるのか、それともセミナーの聴衆のテーブルになるのかによって、幅の持つ意味が違ってくる。幅が広いのか狭いのかということは、テーブルが何に使われるか、何の機能を目的として使われるのかによって違ってくる、という解釈で良いでしょうか。

ホルナゲル　その通りです。今私が前にしているテーブルの幅がいくつなのか、分かりませんが、これが80cmだとしましょう。80cmという数字が出てくると、はっきりとした測定値というふうには聞こえますが、それが何のためのものかということが分からなければ、80cmという具体的な測定値は意味を持たないということになります。何に関連してのものなのか。ホールの聴衆のためのものであれば、80cmは幅が広過ぎるでしょうし、私の机であれば、ものがあふれていますから80cmの幅は狭すぎるということになるでしょう(笑)。単なる測定値ということは、その文脈(コンテキスト)がなければ意味を持たないのです。どういう状況に対するものなのかが分かって、初めて意味が生まれます。

レジリエンスはコストではなく投資

松本　冒頭に申し上げたように、正に今はIoT時代になり、どんどん不確実性の時代になりつつあると私たちは見ています。そうした時代では、システムを設計したり開発したりするときに、どれだけコストをかければ良いかということが、経営者にとって非常に大きな問題になってきます。とくに、例えばセキュリティの世界で、セキュリティを完璧にして障害をなくすという前提でシステムを作ろうとすると、膨大なコストが発生してしまう。その意味では、正にレジリエンスエンジニアリングの考え方で、悪いことが起きないようにするというよりはむしろ、うまくいっていることをどうやって保証するか、という方向に考えていくほうが、経営者には受け入れやすいような気がします。いかがでしょうか。

ホルナゲル　システムのレジリエンスの能力に対する投資は、生産性を高めるための投資にもなると言えます。多くの場合、システムがうまく機能することになるからです。

一方、従来型の安全という意味でのプロテクションのための投資は、何かを生み出すという投資ではありません。従って、コストと見られます。レジリエンスは、プロテクションのためではなく、生産性のためのものです。

松本　システムの安全性はコストではなく投資だというのは、とても重要な考え方だと思います。私たちもそういう考え方を産業界に広げていきたいと思っているのですが、経営者層から見ると、どうしてもコストに見られがちで、なかなか投資と見てくれません。その考え方を変えていくには、どうしたら良いか。何かアイデアをお持ちですか。

ホルナゲル　残念ながら、簡単な答えはありません。考え方を変えていくというのは時間がかかるプロセスです。レジリエンスエンジニアリングに関する最初のミーティングが行われたのが2004年です。今ようやく、これは有効な考え方である、うまく機能させ続けることを担保するものとしてアドバンテージがあると、多くの方からかなり急速に受け入れられてきていると思いますが、それでもこのマインドセットの変化は、これまで13年間にわたってやってきたことが背景にあるからです。これからも長く時間がかかると思います。

マニュアル通りか否かという視点ではない

松本　レジリエンスエンジニアリングが、既にヨーロッパで適用されているというお話がありました。どのように実践されているか、少しお聞きしたいと思います。

まず、どういった産業分野で効果を上げているというような、分野ごとの違いはあるのでしょうか。

ホルナゲル　医療、ヘルスケアが最も成功している分野の一つです。私のもともとのバックグラウンドは産業界で、ヘルスケア畑の人間ではありませんが、過去6年間にわたり、ヘルスケアにかかわっています。仲間と共にレジリエンスヘルスケアを作り、2012年から毎年、会合を開いています。3冊の書籍が発行され、多くの論文も発表しました。1冊目の書籍は中島和江先生によって日本語にも翻訳されています。

このレジリエンスヘルスケアは大変幅広く採用され、日本においても活用されています。ほかにも、オーストラリア、アメリカ、ヨーロッパ、ブラジルなどで使われています。

松本　具体的にはどのように実践されているのでしょうか。

ホルナゲル　現場の人たちと色々なやり取りをし、ディスカッションをして、どういう問題を解決しあるいは改善したいと考えているのか、というところをまず話し合いの中から見い出していきます。そこで、このレジリエンスエンジニアリングの考え方、そしてレジリエンスエンジニアリングの方法論というものを説明し、その理解を図っていく。それによって、改善ができる。理解しようと思っていたことが分かるようになっていく、ということになります。そして、それが改善できれば、更にほかのところにも適用できるという形で広げていきます。

松本　日本ではとくにそうなのですが、決められたマニュアルがあって、それに従ってやりなさい、それに従ってやれば大体うまくいくというやり方が、普通に行われています。何か不測の事態が起きたときに、たまたまマニュアルを無視してやったらうまくいったというケースがあったとき、それはうまくいったケースなんだから、マニュアルは場合によっては無視しても良いのだということを、やり方として組織の中に広げようと考える人がいるかもしれません。しかし、それはレジリエンスエンジニアリングの考え方、ということではないのですよね。マニュアルあるいはマニュアルに従う、ということに関連してどうお考えですか。

ホルナゲル　もちろん、レジリエンスエンジニアリングは、マニュアルを無視して良い、というような考え方ではありません。そういう提唱ではありません。

マニュアルのことは大変興味深い点ですので、たとえを使ってお話しします。例えば、コンピューターのプログラムでプログラミングをするときには、本当にすべて細かいところまで、一番小さな詳細な点まで説明し、記述をしていくことが必要です。コンピューター自体は頭が悪いと言いますか、すべて何もかも記述されたことに従うだけだからです。

だいぶ前のことになりますが、ノルバート・ウイナーという人は、コンピューターの問題というのは、私たちがコンピューターに与えたと思っている指示に対して動くのではなく、本当に与えられた指示の通りに動くことだ、と言っています。

マニュアルとルール、ガイドラインというのは、知性のある人が使うものとして作られています。そのためにすべての細かいレベルにわたっての記述はされていません。それは人が理解をしてくれるものだというベースに従っているからです。書かれていない部分に関しては、ユーザの理解によって埋めてもらえるということに依存しています。書いてあることをやれば、すべてうまくいくというのは、幻想にすぎません。

確かに手順の紹介は、ユーザに対する一助にはなります。やるべきことを覚えておかなければならない、やるべき順序を覚えておかなければならない、ということがなくなり、標準的な状況の中で、ユーザがその手順に従っていくということを期待することはできます。しかしながら、標準ではない状態において、その通りにやってうまくいくということは期待できません。それは生産的ではないのです。

通常の仕事がどのように行われているのかを知ること

松本　書いてあることをやっていればうまくいくという考えは幻想だとおっしゃった点ですが、実は、私どもの、実際に起きた障害の分析や教訓化と再発防止の作業の中で面白い事例がありました。

日本のある金融サービスのシステムが止まった事故です。非常に重要な金融サービスシステムが、半日くらい止まってしまいました。原因は、その日の朝、システムを立ち上げるときにハードウェアの障害が起き、それで待機系のシステムに切替わったはずだったのですが、実際は切替わっていませんでした。そのとき、実はオペレーターは、何かおかしいということに気がついていたのです。しかし彼は、それを報告しなかった。なぜかというと、マニュアルに「何かおかしいと思ったら報告しなさい」ということが書かれていなかったからなんです。

それで大きな障害につながってしまった。もう少し柔軟な運用や組織構成、設計などを考えていく必要があるのだ、ということをあらためて考えさせられる出来事でした。

ホルナゲル　オペレーターが、マニュアルに書かれていなかったから報告しなかったということこそが、マニュアルに従えと主張し過ぎることのリスクであるわけです。マニュアル自体が不完全であるとき、それだけに固執し過ぎることは大きなリスクです。おそらくマニュアルの作成者は、そんなことは書かなくても、何かおかしなことがあれば報告するであろうと想定し、わざわざそのことを記述しなかったのだと思います。その発想は当然でしょう。

レジリエンスエンジニアリングの視点から言っても、通常の作業がどのように行われているか調査したり、日ごろから現場の人たちと話をしてそのやり方を知っていれば、現場での「こういう事象は重大だから報告が必要」「こういうものはそれほど

重大性がないので報告は必要ない」という判断の基準が理解できていたでしょう。そうした日常的なやり方を理解していれば、どういう場合に問題になり得るのかということも見えてきたのではないかと思います。報告すべきか、すべきではないのかという判断を、通常の仕事のやり方の中から導き出し、理解していくことができる。通常の仕事のやり方というものを理解せずに、その機能の改善につなげるということはできないと思います。

松本　日本の場合は、文化的にマニュアル厳守の風潮が非常に根強いので、まず文化から変えていかないといけないのかな、と思いますね。

ホルナゲル　そうですね。文化自体を変えることが必要なのかもしれません。そして、それと共に、その文化の中で毎日行われていることが、実践にどういう影響を持ち得るのか、ということを理解する必要があると思います。

レジリエンスエンジニアリングを実践するためのスキルとは

松本　レジリエンスエンジニアリングを実践するための人材のスキルについて、お話を伺いたいと思います。システムを相互依存性や相互作用性で見る、そして機能で見るという場合、どういうスキルが求められるでしょうか。

ホルナゲル　最初にやらなければならないのは、マインドセットを変えるということです。ものの見方を変えていくということです。これまでの私たちの経験から言うと、始めは実践が容易ではありません。これまでに慣れたやり方と違うやり方になってくるからです。これまでと違うものの見方をする、起こっていることの見方を変えていく、という習慣が必要になると思いますし、それを見るスキルが必要になってきます。

この点に関して二つの言葉を使っています。work as imaginedとwork as doneです。

考えられている仕事のし方が、work as imagined、実際に行われている仕事のやり方が、work as doneです。work as imaginedのほうが、いわゆる手順やマニュアルの中に規定されているような、頭の中でこういうことが起こるであろう、こういう風にやらなければならない、と思っている仕事のやり方です。

頭の中で考えている仕事のやり方と、実際に行われている仕事のやり方について、必要な考え方（マインドセット）に違いがあるということを理解し、その違いを受け入れることが必要になります。

そして、スキルとして必要なのは、実際に行われている仕事のやり方を見ることができる力です。従来は、そこには注意が払われてきませんでした。しかし、見ることができる力を培うことは可能だと思います。実際に起こっていることを見る力です。私の記憶に間違いがなければ、宮本武蔵が「見えざるものを見る」と言った。それが、実際にやられているやり方を見る、というものです。

松本　それは、どうやって身に付けられるのでしょうか。実際に現場を見るということを繰り返してやる、それがスキルを磨く一つの方法になるのでしょうか。

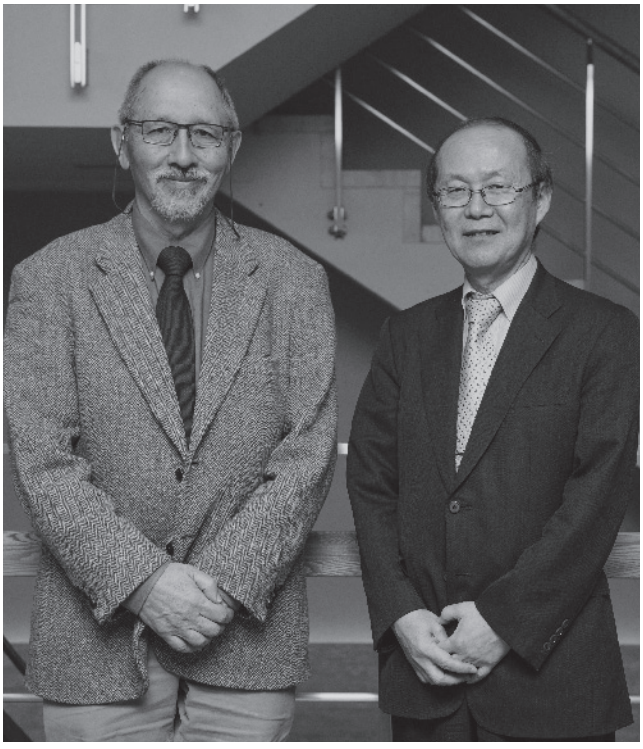
ホルナゲル　このスキルは、比較的容易に身に付けることができると思います。現場の人たちと話をし、その人たちが観察をし始めることによって、自分たちから気づき始めると言えると思います。実際に、日常の業務というのは、注意を払わずに自動的に行われているものが多いのですが、そこから実際に見え始めてきます。自分たち自身が、どう言っているのかということを考えるようにもなるでしょう。時間はかかるかもしれませんが、それがスキルとなり、習慣となっていくと思います。ほかの分野の専門家にも当てはまりますが、専門家はそれ以外の人たちに見えないことを、その状況の中で見ていくことができる。それと同様だと思います。

レジリエンスエンジニアリング すなわち「弾」

松本　最後に一つだけお伺いしたいのですが、先生の著書には、漢字の「弾」というキャラクターが表紙に使われています。これはどういう理由でしょうか。

ホルナゲル　2004年にレジリエンスエンジニアリングの最初の会合が行われたときに、日本人の古くからの友人であるドクター・フジタを招待し、参加していただきました。西洋の人間にとって漢字というのは、とてもエレガントで美しいものですので、ドクター・フジタに、レジリエンス、またはこの考え方を表す漢字はどんなものですか、と聞いてみました。そして教えていただいたのがこの字です。

松本　正にこの「弾」という字は、弾力性の弾であり、レジリエンスという意味だと思います。今日は大変貴重なお話をいただき、ありがとうございました。



CMMI成熟度レベル別に見たソフトウェア品質の良否にかかわる要因の複合的分析



柳田 礼子[※1]　野中 誠[※2]　誉田 直美[※1]

ソフトウェア品質の良否に影響する要因は，通常単一ではない．要因の一つとしてCMMIで示される開発組織の能力があるが，アセスメント結果のみからキーとなる要因を探り当てることは難しい．ソフトウェア品質を効率的に向上させるためには，各成熟度レベルにおいて，実績データを基に分析した結果を含めて考察する必要がある．本論文では，商用ソフトウェア開発プロジェクト522件のデータを対象に，成熟度レベル別に分類木を構築し，有意差検定と相関分析を組み合わせてソフトウェア品質の良否に影響する要因を複合的に分析した．その結果，成熟度レベル1では開発規模，上工程バグ数/KL，及び上工程バグ摘出率が良否を分ける要因と示された．成熟度レベル2では開発規模とテスト工程バグ数/KLが要因と示され，開発規模は成熟度レベル1の約4倍の境界値となった．成熟度レベル3では有意差検定の結果からテスト工程バグ数/KLが主たる要因と示された．成熟度レベル3では，テスト後半にバグが残存している場合にテスト工数が十分にかけられずに出荷後の品質が悪くなる傾向が確認された．

A Compound Factor Analysis by CMMI Maturity Levels related to Success and Failure on Software Quality

Reiko Yanagida[※1], Makoto Nonaka[※2], Naomi Honda[※1]

Organizational capability measured with CMMI is a promising factor to deliver high-quality software. To improve the maturity level of organizational capability, we should identify factors and their relationships affecting software quality by maturity levels. In this paper, the authors analyzed 522 commercial software development projects by using a classification tree method, statistical significance tests and correlational analysis. The result showed that development size, defect detection density in reviews, and defect detection rate before testing were identified as major factors in maturity level 1. In maturity level 2, development size and defect detection density in testing were identified as major factors, but the boundary of development size is four times larger than that in maturity level 1. In maturity level 3, defect detection density in testing was identified with significance tests. In this maturity level, the failed projects had more bugs remained in the latter stages of testing because of lack of testing effort.

※1 日本電気株式会社　※2 東洋大学

1　はじめに

ソフトウェア品質の良否に影響する要因の一つとしてCMMI(Capability Maturity Model Integration)[1]で示される開発組織の能力がある．Jones[2]は成熟度レベルが高いほどリリース後のソフトウェア欠陥密度が低くなることを示している．また，成熟度レベルがソフトウェア品質の良否に影響することが先行研究により示されている[3][4][5]．

CMMIは段階的なプロセス改善を促すモデルだが，成熟度レベルを上げるのには一般に長い期間を要する[6]．そのため，成熟度の状況に応じたプロセス改善を効率的に進めるには，それぞれの成熟度レベルにおいてソフトウェア品質の良否に影響する要因を把握することが求められる．

そのような要因について成熟度レベル別に分析した研究が幾つかあるが，それらには課題がある．Rainerら[7]は成熟度レベル別にプロセス改善の成功要因を示している．しかし，それらは定量的なプロジェクト管理に結び付く要因を示していない．Yanagidaら[8]は開発規模や上工程バグ摘出率などのメトリクスを用いて成熟度レベル別に品質改善の要因を導出している．しかし，メトリクスは「値が大きいほど良い」のように一元的に判断できないことが多く，複合的に分析する必要がある．また，Yanagidaら[8]は成熟度レベル別に相関分析をしているが，セグメントを細分化した分析はしていない．

本論文では，筆者らの所属部門が管轄する組織から得た商用ソフトウェア開発プロジェクト522件のデータを，成熟度レベル別に見たソフトウェア品質の良否にかかわる要因を複合的に分析した結果を報告する．本論文では，ソフトウェア品質会計[9]と呼ばれる定量的管理の仕組みを通して収集されたメトリクスのデータを分析対象とし，CARTアルゴリズムに基づいて分類木を構築し，その上でほかのメトリクスも含めた有意差検定及び相関分析を用いる．

2　分析対象データ

2.1　対象組織の背景

筆者らの所属企業ではSW-CMM[10]及びCMMIに基づくプロセス改善に継続的に取り組んでおり，成熟度レベル1から5までの多様な開発組織がある．また，ソフトウェア品質会計[9]と呼ばれる定量的管理の仕組みを30年以上にわたって展開しており，成熟度レベルの低い組織であっても一定の定量的管理が行われている．

開発プロセスはV字モデルに基づいており，要件定義からコードレビューまでを上工程，単体テストからシステムテストまでをテスト工程と呼んでいる．

2.2　対象組織とプロジェクト数

本論文では，筆者らの所属企業内において開発対象や背景などが似ている，システムインテグレーション事業にかかわる組織を分析対象とする．これらの組織は，成熟度レベル1から3の組織がある．ソフトウェア品質の良否は，リリース後バグ密度[b]の実績値をそれぞれの組織があらかじめ定めた品質基準と比較して，「達成」または「未達」に分類した．

本論文では，分析対象の組織において2014年度に開発を完了した522件の商用ソフトウェア開発プロジェクトを対象とした．成熟度レベル別のプロジェクト件数の内訳を表1に示す．表1では，「達成」と「未達」のプロジェクト件数内訳を都合により非掲載としているが，成熟度レベル1の達成率(「達成」件数をプロジェクト合計件数で割った値)を基準としたポイントの差の値を示している．成熟度レベルが上がるにつれてその比率が向上していることが分かる．なお，表1は外れ値を除去した後の件数である．

表1　分析対象のデータ(外れ値除去後)

レベル	プロジェクト数	レベル1との達成率の差
1	317	−
2	138	+13.3pt.
3	67	+21.8pt.

2.3　メトリクス

メトリクスの一覧を表2に示す．これらを選んだ理由は，ソフトウェア品質会計[9]に基づく品質管理において着目することの多い主要なものであるためと，欠損率が比較的低いためである．欠損率については2.4項で詳しく述べる．

なお，本論文で分析結果を示すときには，実測値ではなく相対値を用いる．相対値は，それぞれの値を対応するメトリクスの全体の平均値で割った値とする．

表2　分析に用いたメトリクス

記号	メトリクス	単位	意味
SIZE	開発規模	KL	新規作成または変更した論理ソースコード行数
UpBD%	上工程バグ摘出率	%	全バグ数に対する上工程摘出バグ数の比率
UpEff	上工程工数/KL	人時/KL	上工程工数を開発規模で割った値
TstEff	テスト工程工数/KL	人時/KL	テスト工程工数を開発規模で割った値
UpRvEff	上工程レビュー工数/KL	人時/KL	上工程レビュー工数を開発規模で割った値
TstItem	テスト項目数/KL	件/KL	すべてのテスト項目数を開発規模で割った値
UpBug	上工程バグ数/KL	件/KL	上工程で摘出したバグ数を開発規模で割った値
TstBug	テスト工程バグ数/KL	件/KL	テスト工程で摘出したバグ数を開発規模で割った値

※KLはKilo Lines of Codeの略

2.4　成熟度レベル別のメトリクスの値

成熟度レベル別のメトリクスの中央値を，「達成」と「未達」に分けて表3に示す．開発規模とテスト工程バグ数/KLは「達成」のほうが一貫して小さい．一方，上工程バグ摘出率は「達成」のほうが一貫して大きい．

成熟度レベル別の各メトリクスの欠損率を表4に示す．成熟度レベル1の欠損率の高さが目立つが，2.1節で述べたようにソフトウェア品質会計の効果もあってバグ数にかかわるメトリクスの欠損率は比較的低く抑えられている．

表3　成熟度レベル別のメトリクスの中央値

メトリクス	レベル1		レベル2		レベル3	
	達成	未達	達成	未達	達成	未達
SIZE	.309	.662	.184	.661	.144	.217
UpBD%	1.071	.996	.986	.950	.983	.942
UpEff	.708	.537	.773	.776	.835	.834
TstEff	.446	.344	.778	.968	.650	.845
UpRvEff	.757	.620	.647	.737	.631	.477
TstItem	.626	.537	.534	.620	.951	.634
UpBug	1.084	.822	.595	.787	.672	1.220
TstBug	.595	1.142	.773	1.142	.744	1.542

表4　成熟度レベル別の各メトリクスの欠損率

メトリクス	レベル1	レベル2	レベル3
SIZE	0.0%	0.0%	0.0%
UpBD%	14.2%	2.9%	1.5%
UpEff	63.1%	0.0%	5.9%
TstEff	65.0%	0.0%	4.4%
UpRvEff	67.8%	0.7%	5.9%
TstItem	9.7%	0.0%	0.0%
UpBug	9.8%	0.0%	1.5%
TstBug	10.1%	0.0%	0.0%

3　分類木による分析結果

3.1　成熟度レベル1

成熟度レベル1の分類木を図1に，各ノードの境界値と達成率を図2に示す．分類木の葉ノードの縦軸に示した値は「達成」と「未達」を示しており，それぞれ1と0である．図2の項番は図1のノード番号と対応しており，＊印は葉ノードであることを示している．

最初のノードは開発規模であり，その境界値は平均値の0.101すなわち約1/10となった．図2のノード1と7にある通り，開発規模が全体の平均値の1/10未満だと達成率が91.0%と高いが，これ以上の開発規模では達成率が61.6%に低下している．成熟度レベル1では，小規模開発であることが「達成」に寄与する第一の要因と言える．

続いて，上工程バグ数/KLが平均値の0.349すなわち約1/3未満だと達成率が90.2%だが，これ以上になると達成率が56.0%へと低下している．上工程での摘出バグ数が少ないことは，上工程での混入バグ数が少ない，または，上工程でのバグの摘出が不十分であることが考えられる．しかし，更に上工程バグ数が平均値の0.349以上のノードにおいては，上工程バグ摘出率が平均値の0.922以上だと達成率が62.0%だが，これ未満になると達成率が32.6%へと大きく低下している．これらより，開発規模が小規模でないプロジェクトにおいて，上工程でのバグ摘出が不十分とは考えにくい．上工程でのバグ混入数を低い水準に抑えること，すなわち設計品質の良さが「達成」に寄与する要因と言える．

また，開発規模が小規模でなく，設計品質が悪いプロジェクトでは，上工程バグ摘出率を平均値に近い値以上に維持することが「達成」に寄与すると言える．

以上の分析結果はいずれもソフトウェア品質管理における筆者らの経験則と整合しており，合理的な分析結果と言える．

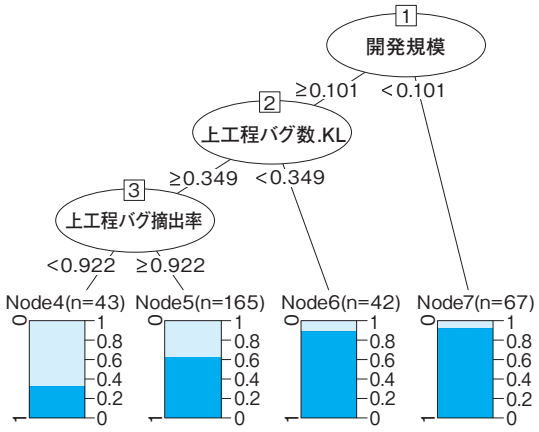


図1　成熟度レベル1：分類木

[1]開発規模>=0.101（61.6%）
　[2]上工程バグ数/KL>=0.349（56.0%）
　　[4]上工程バグ摘出率<0.922（32.6%）*
　　[5]上工程バグ摘出率>=0.922（62.0%）*
　[6]上工程バグ数/KL<0.349（90.2%）*
[7]開発規模<0.101（91.0%）*

図2　成熟度レベル1：分類木の各ノードの境界値と達成率

3.2　成熟度レベル2

成熟度レベル2の分類木を図3に，各ノードの境界値と達成率を図4に示す．　成熟度レベル1と同様，最初のノードに開発規模が示されたが，その境界値は平均値の0.463であり，レベル1の約4倍の値となった．図4のノード1と7にある通り，開発規模がこの境界値未満だと達成率が92.5%と高いが，これ以上になると達成率が57.8%に低下している．成熟度レベル2では，レベル1よりは大きな規模まで対応できるものの，中規模までの開発であることが「達成」に寄与する第一の要因と言える．

続いて，テスト工程バグ数/KLが平均値の0.869未満だと達成率が76.0%だが，これ以上だと達成率が35.0%へと低下している．テストでの摘出バグ数が少ないことは，上工程で品質が確保されてテスト開始時点で残存しているバグ数が少ない，または，テストでのバグの摘出が不十分であることが考えられる．しかし，これらのノードにおけるテスト工数の中央値を比較すると，テスト工程バグ数/KLが平均値の0.869未満のノードの値が大きく，有意差がある．この結果は4.2節で詳細に述べるが，このことから，開発規模が中規模より大きいプロジェクトにおいて，テスト工程でのバグ摘出が不十分とは考えにくい．テストでの摘出バグ数を平均値のやや下という水準に抑えられていること，すなわち，上工程で一定の品質が確保されていることが「達成」に寄与する要因と言える．

ここまでの分析結果は経験則と整合しており，とくにレベル1の4倍という開発規模の境界値は興味深い．しかしその次には，上工程レビュー工数/KLが平均値の0.403未満だと達成率が62.5%だが，これ以上になると達成率が16.7%へと大きく低下するという，経験則とは逆の関係を示す結果が得られた．ただし，ノード3はデータ件数が20件と少ないため，より多くのデータでの検証が必要である．

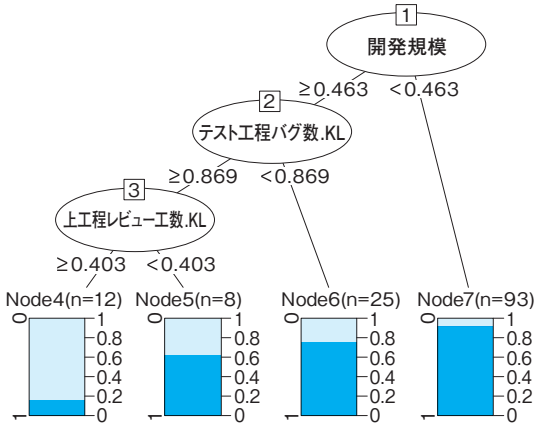


図3　成熟度レベル2：分類木

[1]開発規模>=0.463（57.8%）
　[2]テスト工程バグ数／KL>=0.869（35.0%）
　　[4]上工程レビュー工数／KL>=0.403（16.7%）*
　　[5]上工程レビュー工数／KL<0.403（62.5%）*
　[6]テスト工程バグ数／KL<0.869（76.0%）*
[7]開発規模<0.463（92.5%）*

図4　成熟度レベル2：分類木の各ノードの境界値と達成率

3.3　成熟度レベル3

成熟度レベル3では分類木が構築されなかった．これは，達成率に偏りのあるデータであることと，プロジェクト数が67件とほかに比べて少ないこと，成熟度レベル3であるためプロセスが標準化されていてメトリクス間に大きな差がなかったことによると考えられる．4節において各メトリクスの有意差検定により「達成」の寄与要因を分析する．

4　ノード別の相関分析

3節で得られた分類木の葉ノードのそれぞれについて，メトリクス間の関係を相関分析により詳しく分析する．

4.1　成熟度レベル1

分類木に現れたメトリクスは開発規模，上工程バグ数/KL，及び上工程バグ摘出率であった．ここで，小規模プロジェクトの成功要因を探ることは実務ニーズとして優先度が低いため，図1のノードの2以降に着目する．そして，ノード3と6を分ける要因と，ノード4と5を分ける要因についてそれぞれ分析する．分類木に現れなかったほかのメトリクスは，ウィルコクソンの順位和検定を用いて中央値についての有意差検定を行う．なお，いずれも帰無仮説を「達成」と「未達」の間で各々のメトリクスの中央値に差がないものとし，対立仮説ではそれらに差があるものとしている．

(1)ノード3と6

ノード間の有意差検定の結果と各ノードにおける中央値を表5に示す．両側検定でp値が5%未満のものと，達成率と中央値の変化が妥当と考えられるものを太字で示している．該当するメトリクスとしてテスト工程工数/KL，テスト項目数/KL，及びテスト工程バグ数/KLが得られた．以降，これらのメトリクスを加えて相関分析を行う．

表6と表7に，ノード3と6の「達成」と「未達」に分けたメトリクス間の相関分析の結果をそれぞれ示す．なお，ノード6の「未達」はデータ数が3件と少ないため分析の対象外とした．表6と表7を比べると，ノード6の「達成」ではテスト項目数/KLとテスト工数/KLにやや強い正の相関があるが，ノード3の「達成」では相関がない．3.1節で述べた通り，ノード6は小規模ではないが設計品質が良く，テスト工数がテスト項目数に比例するという安定したテストが実施できていると考えられる．また，ノード6では上工程バグ数/KLとテスト工程バグ数/KLに相関がないが，ノード3では「達成」と「未達」のいずれにも正の相関がある．すなわち，ノード3では「上工程で多くのバグが摘出されるとテストでも多くのバグが摘出される」という状況だが，ノード6ではそうではない．これはノード6における設計品質の良さの現れと考えられ，3.1節の分析結果と整合する．

表5　成熟度レベル1：ノード3と6の中央値と有意差検定

メトリクス	p値	ノード	
		3	6
達成率	—	56.0%	90.2%
UpEff	.1282	.596	.412
TstEff	**.0463**	**.409**	**.243**
UpRvEff	.0841	.691	.424
TstItem	**.0232**	**.569**	**.475**
TstBug	**3.29e-13**	**1.101**	**.068**

表6　成熟度レベル1：ノード3の相関分析

メトリクス	成否	TstEff	TstItem	UpBug	TstBug
UpBD%	達成	-.049	.129	.131	**-.723****
	未達	.022	.095	.126	**-.732****
TstEff	達成		.168	**-.337***	-.220
	未達		.066	-.172	-.111
TstItem	達成			.084	-.122
	未達			.132	.010
UpBug	達成				**.422****
	未達				**.502****

**p < 0.01　*p < 0.05

表7　成熟度レベル1：ノード6の相関分析（達成）

メトリクス	TstEff	TstItem	UpBug	TstBug
UpBD%	-.229	**-.564****	.189	**-.538****
TstEff		**.692****	.016	.136
TstItem			-.262	.130
UpBug				.246

**p < 0.01　*p < 0.05

(2)ノード4と5

続いて，ノード4と5を分ける要因について分析する．

ノード間の有意差検定の結果と各ノードにおける中央値を表8に示す．有意差があり，達成率と中央値の変化が妥当と考えられるメトリクスとして上工程レビュー工数/KL，テスト項目数/KL，及びテスト工程バグ数/KLが得られた．以降これらのメトリクスを加えて相関分析を行う．

表9と表10に，ノード4と5について「達成」と「未達」に分けたメトリクス間の相関分析の結果をそれぞれ示す．ノード4の「達成」と「未達」の両方において上工程バグ数/KLと上工程バグ摘出率の間に正の相関があるが，ノード5では相関がない．また，ノード4では上工程バグ摘出率とテスト工程バグ数/KLの間に相関がないが，ノード5では強い負の相関があった．ノード4では，設計品質が悪く，レビュー

での摘出が容易なバグを上工程で数多く混入しており，上工程バグ摘出率もそれに応じて一定水準までは高まっていく．しかし，上工程バグ摘出率の割にはテスト工程でバグを十分に摘出できていない状況が考えられる．

ノード５では，上工程バグ数/KLと上工程バグ摘出率の間には相関がなく，「達成」と「未達」の両方において，上工程バグ数/KLとテスト工程バグ数/KLの間に正の相関がある．設計品質が悪い状況において，レビュー及びテストで一定水準まで摘出できていることが，ノード４と比較すると達成率が高くなっている要因と考えられる．

表８　成熟度レベル１：ノード４と５の中央値と有意差検定

メトリクス	p値	ノード	
		4	5
達成率	−	32.6%	62.0%
UpEff	.2018	.437	.599
TstEff	.6778	.393	.430
UpRvEff	**.0070**	**.504**	**.803**
TstItem	**.0376**	**.486**	**.642**
TstBug	**2.53e-12**	**1.942**	**.721**

表９　成熟度レベル１：ノード４の相関分析

メトリクス	成否	UpRvEff	TstItem	UpBug	TstBug
UpBD%	達成	-.125	-.096	**.768****	-.160
	未達	-.060	-.269	**.459***	-.337
UpRvEff	達成		**.887***	.089	.131
	未達		.260	.449	.416
TstItem	達成			-.307	-.432
	未達			.353	**.635****
UpBug	達成				.483
	未達				**.642****

**p < 0.01　*p < 0.05

表１０　成熟度レベル１：ノード５の相関分析

メトリクス	成否	UpRvEff	TstItem	UpBug	TstBug
UpBD%	達成	.065	.111	-.014	**-.735****
	未達	-.170	-.085	-.185	**-.757****
UpRvEff	達成		.281	-.037	-.031
	未達		.398	.408	.408
TstItem	達成			.084	-.083
	未達			.071	.075
UpBug	達成				**.532****
	未達				**.721****

**p < 0.01　*p < 0.05

4.2　成熟度レベル２

分類木に現れたメトリクスは開発規模，テスト工程バグ数/KL，及び上工程レビュー工数/KLであった．ここで，中規模以下のプロジェクトの成功要因を探るのではなく図３のノード２以降に着目する．ただし，ノード４と５はデータの件数が少ないため，ノード３と６に着目して分析する．分類木に現れなかったほかのメトリクスについて，中央値についての有意差検定を行った．ノード間の有意差検定の結果と各ノードにおける中央値を表11に示す．

有意差があり，達成率と中央値の変化が妥当と考えられるメトリクスとして上工程バグ摘出率，テスト工程工数/KL，及び上工程バグ数/KLが得られた．以降，これらのメトリクスを加えて相関分析を行う．

表12及び表13に，ノード3と6について，「達成」と「未達」に分けたメトリクス間の相関分析の結果をそれぞれ示す．相関係数の求め方は4.1節で述べた通りである．各セグメントのデータ数は少ないことに留意が必要である．

ノード３の「達成」と「未達」の両方において，上工程バグ数/KLと上工程バグ摘出率の間に正の相関があるが，ノード６では相関がないことが読み取れる．また，ノード３では上工程バグ摘出率とテスト工程バグ数/KLの間に相関がないが，ノード６の「達成」では負の相関があった．これは成熟度レベル１のノード４と５で読み取れたことと同じであり，同様の状況が生じていると考えられる．

また，ノード６の「達成」において，上工程レビュー工数と上工程バグ摘出率との間に正の相関があり，テスト工程バグ数/KLとの間に負の相関がある．これは，一定量のバグが含まれる成果物に対しては，上工程レビューに十分な工数を投入することによってテストで検出されるバグ数が減少する，という品質管理の観点からすれば期待通りの結果が示されている．別の見方をすれば，このような期待通りの傾向が現れるのは，成熟度レベルが２の中でも一部のセグメントのみと言える．

表１１　成熟度レベル２：ノード３と６の中央値と有意差検定

メトリクス	p値	ノード	
		3	6
達成率	−	35.0%	76.0%
UpBD%	**.000**	**67.2**	**77.4**
UpEff	.148	.541	.964
TstEff	**.013**	**.379**	**1.604**
TstItem	.293	.601	.508
UpBug	**1.37e-06**	**1.081**	**.252**

表12　成熟度レベル2：ノード3の相関分析

メトリクス	ノード	TstEff	UpRvEff	UpBug	TstBug
UpBD%	達成	.388	.415	**.797***	-.116
	未達	.291	.267	**.747****	-.297
TstEff	達成		**.987****	.700	.288
	未達		.291	.090	-.046
UpRvEff	達成			.144	-.260
	未達			-.358	-.108
UpBug	達成				**.761****
	未達				**.840****

**p < 0.01　*p < 0.05

表13　成熟度レベル2：ノード6の相関分析

メトリクス	ノード	TstEff	UpRvEff	UpBug	TstBug
UpBD%	達成	**.685****	**.515***	.034	**-.482****
	未達	.246	.481	.255	-.453
TstEff	達成		**.536***	-.323	**-.611****
	未達		-.138	-.453	-.501
UpRvEff	達成			-.358	**-.581***
	未達			-.082	-.438
UpBug	達成				**.840****
	未達				.739

**p < 0.01　*p < 0.05

4.3　成熟度レベル3

表3に示したうち，「達成」と「未達」について有意水準5%で有意差のあるメトリクスはなかった．最もp値が小さかったのはテスト工程バグ数/KLであり，0.087であった．

テスト工程の状況を詳細に分析するため，表14に示すメトリクスを用いて，工程別の値を比較する．各メトリクスの「達成」と「未達」の中央値についての有意差検定を行った結果を表15に示す．

工程別テスト工数/KLについて，「未達」は工程が進むにつれて値が一様に増加しているが，「達成」は減少している．一方，工程別テスト項目数/KLについては，全工程にわたり「未達」のほうが「達成」よりも値が低い．とくに「未達」のIT工程テスト項目数/KLは，有意差があるとは言えないが低い値となっている．工程別テスト工程バグ数/KLは，全工程にわたって「達成」よりも高い値である．とくにST工程テストバグ数/KLは有意差があることが示された．

表14　工程別分析に用いたメトリクス

メトリクス	単位	意味
工程別テスト工数/KL	人時/KL	UT, IT, ST工程ごとの，各テスト工程工数を開発規模で割った値
工程別テスト項目数/KL	件/KL	UT, IT, ST工程ごとの，各テスト工程のテスト項目数を開発規模で割った値
工程別テストバグ数/KL	件/KL	UT, IT, ST工程ごとの，各テスト工程で摘出したバグ数を開発規模で割った値

※UT：単体テスト　IT：結合テスト　ST：システムテスト

表15　工程別の中央値と有意差検定

メトリクス	p値	達成	未達
UT工程テスト工数/KL	.945	.677	.532
IT工程テスト工数/KL	.812	.488	.587
ST工程テスト工数/KL	.338	.456	1.271
UT工程テスト項目数/KL	.702	.584	.392
IT工程テスト項目数/KL	.621	.420	.278
ST工程テスト項目数/KL	.888	.400	.372
UT工程テストバグ数/KL	.134	.578	1.160
IT工程テストバグ数/KL	.494	.583	1.014
ST工程テストバグ数/KL	**.006**	**.000**	**2.357**

5　考察及び関連研究

成熟度レベル1について総じて言えることは，基本的なプロジェクト管理に課題があり，小規模でない開発においてはソフトウェアの品質が悪化する傾向があるということである．小規模でない開発で，設計品質が悪く，上工程バグ摘出率が低い場合には，設計工程での品質状況に応じた十分なレビュー及びバグ摘出が実施されていない．個人の力量によって結果的に品質基準を満たすプロジェクトも存在するが，プロジェクトの結果は予測不能である成熟度レベル1の概念及びアセスメントの結果と合致する．まずは，基本的な開発管理技術の教育，適用が必要である．

成熟度レベル2においては，レベル1と比較すると約4倍の開発規模のプロジェクトまでは品質を制御することが可能である．したがって，基本的なマネジメントが機能していると考えられるが，やはり規模が大きくなるとソフトウェア品質が悪化する傾向が観察されている．ただし，上工程でレビューに十分な工数を投入し，テストでの検出バグを低く抑え，結果的に「達成」となるプロジェクトもある．品質確保を確実なものにするためには，品質の良し悪しにかかわらず上工程でレビューに一定量の工数をかける必要があるという我々の経験則から，レビューに十分な工数を投入することにより一定の品質を確保していると

考えられる．このような傾向を示すプロジェクトのプロセスを標準化して展開し，組織内のプロジェクトの品質底上げを図ることが求められる．ベストプラクティスの展開が不十分，改善成果の組織資産への組み込みが不十分であるといったアセスメントの所見とも整合する．

成熟度レベル３の組織においては開発プロセスの標準化がある程度進んでいる．分類木が構築されなかった理由は，標準化の浸透のために「達成」と「未達」において有意差のあるメトリクスはなかったためと考えられる．一方で，「未達」は，テスト工程全般にわたって「達成」よりもテスト項目数が少ない割にテスト工程バグ数は多く，終盤のST工程においてのバグ数の差が最も大きい．テスト工数については，「未達」は工程が進むにつれて一様に増加しており，徐々に積み残されたバグが終盤のテスト工程で検出され，工数を費やしても検出し切れずに最終的には工数不足となって十分に品質を確保できないまま出荷していると考えられる．すなわち，プロジェクト遂行中のきめ細やかな管理に課題がある場合に「未達」になる傾向がある．プロジェクトの実績をより高い頻度で把握し，実績に応じたタイムリーなマネジメントが重要である．CMMIの成熟度レベル４で求められる，定量的，統計的なプロジェクト管理に関する課題と言える．

CMMまたはCMMI成熟度レベル５の組織に焦点を当てた品質良否の要因分析はこれまでに報告されている．Honda[11]はCMMI成熟度レベル５に該当する２つの組織について分析しており，同じ成熟度レベル５でもメトリクスの現れ方には大きな違いがあることを示している．また，Agrawalら[12]はCMM成熟度レベル５に該当する組織について分析した結果，品質に寄与する統計的有意な要因は開発規模のみであったとしている．成熟度レベル別に分析した研究ではRainerら[7]の研究があるが，１節でも述べたようにこの研究では組織的プロセスの施策にかかわる要因を挙げており，個別のプロジェクト管理において有用な要因を挙げていない．また，Yanagidaら[8]の研究では，成熟度レベル別に品質の良否にかかわる要因を分析しているが，メトリクスを複合的に組み合わせた分析という観点において十分でない．本研究はソフトウェア品質の良否にかかわる要因の複合的な分析を成熟度レベル別に行った点が特徴である．

6　おわりに

本論文では，522件のプロジェクトデータからCMMIの成熟度レベル別に分類木を構築した上で，有意差検定及び相関分析を組み合わせてソフトウェア品質の良否に影響する要因を複合的に分析した．その結果，成熟度レベル１では開発規模，上工程バグ数/KL，及び上工程バグ摘出率が良否を分ける要因であると示された．成熟度レベル２では開発規模とテスト工程バグ数/KLが要因であると示され，このうち開発規模は成熟度レベル１の約４倍の境界値となった．成熟度レベル３では，有意差検定の結果からテスト工程バグ数/KLが主たる要因と示された．

今後の課題は，データ数の少ないセグメントの対処，より上位の成熟度レベルでの分析などである．

【参考文献】

[1] CMMI Product Team, CMMI for Development, Version 1.3, Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2010-TR-033 (2010).
[2] Jones. C.: Software Quality in 2013: A Survey of the State of the Art, http://namcookanalytics.com/software-quality-survey-state-art/（参照2015-04-15）.
[3] Harter, D.E., Krishnan, M.S. and Slaughter, S.A.; Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development, Management Science, Vol. 46, Issue 4, pp.451-466 (2000).
[4] Harter, D.E. and Slaughter, S.A.,: Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis, Management Science, Vol.49, Issue 6, pp.784-800 (2003).
[5] Gibson, D. L., Goldenson, D R. and Kost, K.: Performance Results of CMMI-Based Process Improvement, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2006-TR-004 (2006).
[6] Process Maturity Profile, http://cmmiinstitute.com/process-maturity-profile（参照 2015-04-15）.
[7] Rainer, A. and Hall, T.: Key Success Factors for Implementing Software Process Improvement: A Maturity-based Analysis, Journal of Systems and Software, Vol. 62, No. 2, pp. 71 – 84 (2002).
[8] Yanagida, R., Honda, N. and Komiyama, T.: What are the Key Factors of Quality Improvement regarding CMMI Maturity Levels?, Proceedings of 6th World Congress on Software Quality, London, England, pp.1-12 (2014).
[9] 誉田直美：ソフトウェア品質会計，日科技連出版社（2010）.
[10] Key Practices of the Capability Maturity Model, Version 1.1, Carnegie Mellon University, Software Engineering Institute, CMU/SEI-93-TR-025 (1993).
[11] Honda, N.: Success Factors to Achieve Excellent Quality – CMMI Level 5 Organizations Research Report, Proceedings of 5th World Congress on Software Quality, Shanghai, China, pp.1-8 (2011).
[12] Agrawal, M. and Chari, K.: Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects, IEEE Transactions on Software Engineering, Vol.33, No.3, pp.145-156 (2007).

要求仕様の一貫性検証支援ツールの提案と適用評価

位野木 万里※　　近藤 公久※

要求仕様の品質特性である「一貫性」に着目し，ベテラン技術者が経験的に得た検証知識を形式知化し，それら知識に基づき，要求仕様の一貫性検証支援ツールを実現した．実システムの要求仕様書を用いて本ツールの適用評価を行い，本ツールは技術者の効率的な仕様検証を支援する点において一定の効果があることを明らかにした．

Proposal of a Requirements Consistency Verification Support Tool and Its Evaluation

Mari Inoki※, Tadahisa Kondo※

In this paper, focusing on the consistency of the requirements quality, we propose a requirements consistency verification support tool. It helps a requirements analyst verify the requirements specification and improve the specification from the viewpoint of the requirements consistency. We have extracted tacit knowledge for verifying requirements and made it explicit as the verification knowledge which were incorporated into the tool. We have evaluated the tool by utilizing the actual requirements specifications. According to the evaluation, it was clarified that the tool is effective for an analyst to verify and improve the requirements specifications efficiently.

1　はじめに

近年，国内外において要求定義に関する標準化が盛んであり［ISOIECIEEE2011］［JISA2011］，真の顧客要求を反映させた魅力あるソフトウェアの要求を定義することは，産業界が重視する重要な課題である．要求定義に関する標準や知識体系が策定され，各企業はそのような標準や知識体系に基づき要求定義を実践しつつある．しかし，実際の要求定義は開発対象となる領域，組織が直面する課題，利用する技術などの条件に応じて工夫が必要となり，標準が示す一般化されたやり方のみでは対応が困難である．例えば，要求定義工程で定義した要求仕様の品質に関しては，要求工学知識体系REBOK［JISA2011］によれば，完全性，トレーサビリティ，一貫性などの品質特性が定義されているものの，現状では，各検証は組織のベテラン技術者が，各自の属人的な方法により実施している．初級の技術者が要求定義を実施することは失敗のリスクが高く，要求定義はベテランの技術者のみが従事することとなり，効率的な要求定義の実施は困難な状況にある［JISA2014］．

要求仕様書の検証のしやすさを考慮して，ダイアグラムや形式言語を用いて仕様を厳密に記述する取り組みも行われている．しかし，ソフトウェア要求は，構築してみなければ明らかにならない仕様が多いことや，要求仕様の分析者や要求の源泉となるステークホルダはそのようなダイアグラムや形式言語の専門家でないことから，自然言語により仕様を記述するという状況は増加傾向にある．例えば，アジャイル開発などの開発スタイルの需要が高まっているが［Hiranabe2013］［Rasmusson2011］，アジャイル開発ではユースケースシナリオ［Cockburn2000］や，ユーザストーリー［Cohn2004］［Wang2014］などの自然言語による仕様により要求仕様を記述している．したがって，自然言語で記述された仕様を対象に，仕様の品質を一定に保つための

※　工学院大学

取り組みは極めて重要である．

木村らは，自然言語で記述された要求仕様の検証支援ツールを開発し，同ツールを用いた要求仕様の高品質化への取り組み事例を報告している［Kimura2014］．［Kimura2014］による検証技術は，設計要素の一つであるアクターにフォーカスし，ベテラン技術者による，アクターの表記ゆれを防止するための検証ノウハウをツール化した点が特徴である．しかし，［Kimura2014］の成果は，アクター定義の観点で一貫した要求仕様を作成することに限定した，特定企業内での成功例であり，ノウハウやツールの一般化には至っていない．

本研究では，要求仕様の一貫性の検証の範囲に，機能への入出力となる「データ」，アクターとシステムとのインターフェースとなる「画面」，システムの働きに相当する「振る舞い」の設計要素を追加する．設計要素の定義漏れや表記ゆれの検証ノウハウは，対象ドメインや組織の条件により様々と考えられるため，組織がカスタマイズ可能なように柔軟な形式知化を図る．形式知化した検証知識を検証支援ツールに組み込み，検証を自動化し，初級技術者が効率的かつ適切に要求仕様を検証することを目指す．

以下，本稿は次のように構成する．2節において，自然言語で記述された要求仕様の検証技術の現状を整理する．3節では，2節で示す現状の技術動向を踏まえ，自然言語で記述された要求仕様の検証技術に対する本研究のアプローチを示す．4節では，要求仕様の一貫性検証知識の形式知化について説明する．5節では，開発した要求仕様の一貫性検証支援ツールについて説明する．6節では，実際の仕様書に対して，当該ツールを適用した結果を示す．7節では，ツールの有効性や妥当性について考察し今後の課題を整理する．8節で本稿をまとめる．

2　自然言語で記述された要求仕様の検証技術に関する関連研究

自然言語で記述された要求仕様を記述するために，Pohlらは機能要求仕様を記述するためのテンプレートを提案した［Pohl2015］．テンプレートでは，英語で記述された一つの機能要求文は，条件，主語（システム名），助動詞，動詞，目的語，目的語の詳細により構成するとしている．日本語で記述された要求に対して，Pohlらの提案をそのまま適用することは困難であるが，テンプレートが示す構成要素を満たすように要求を記述することは，要求仕様の品質向上に貢献すると考えられる．Pohlらのテンプレートにおいて，目的語には，データや画面が相当するが，それら用語自体に表記ゆれがあれば，要求文自体のあいまい性は解消されない．よって，要求文を構成する設計要素に焦点を当てた，使用する用語の一貫性の検証を支援することが必要である．

Lucassenらは，アジャイル開発においてユーザ要求の記述によく使われるユーザストーリーの品質を安定させるための品質モデルと支援ツールAQUSAの開発及び評価を報告している［Lucassen2015］．Lucassenらはユーザストーリーの品質評価のフレームワークとして，Syntactic, Semantic, Pragmaticの視点に分類される14個のメトリクスを定義した．AQUSAは，Atomic, Minimal, Explicit dependencies, Uniform, Uniqueの5つのメトリクスによる検証を自動化しているが，Semanticsの観点のUnambiguousの検証はスコープ外である．自然言語で記述された要求仕様の品質向上には，Semanticsの観点からのUnambiguousの検証，すなわち，表記ゆれなどの用語の一貫性検証が不可欠と考えられる．

日本語の自然言語で記述された仕様書の検証やレビューについても研究が行われている［Kouno2010］［Kuno2012］．河野らは，ドキュメント内において，あいまいさや不備につながりやすいキーワードの洗い出しと，そのようなキーワードのチェックツールを考案した．キーワードとしては，「～場合」などの条件を示す用語，「あれ」「これ」などの指示代名詞などが含まれる［Kouno2010］．

久野らは，同じ語句でも，使われ方によりあいまいの度合いが異なるとして，あいまい性の高い語句を選択的に検出することを目的に，語句のあいまい性を判定するツールを提案した［Kuno2012］．久野らが提案するツールでは，あいまい性のレベルを，曖昧語，準曖昧語，非曖昧語に分けて検出する．提案されたツールを実際の仕様書にて評価したところ，複数の解釈が可能な曖昧語を高いレベルで網羅的に検出でき，仕様書の修正に対して有効であることを確認している．

先行研究［Kouno2010］［Kuno2012］が着目した「あいまいさにつながりやすいキーワード」は重要なものの，要求仕様の品質向上には，対象ドキュメント内で定義された設計要素のあいまいさの解消が更に重要である．例えば，ある設計要素を示す用語が，あいまいさにつながりやすいキーワードには非該当でも，その設計要素が，別の語句で定義されたり，明確な定義なく機能要求などに使用されていれば，その機能要求の内容は，開発者や分析者など複数の読み手により解釈が異なる内容につながるリスクが高い．自然言語で記述された要求仕様に対しての検証支援のためには，特定のキーワードに加えて，要求仕様の本質的な定義内容に踏み込んで，設計要素のあいまいさや不整合を指摘する検証を行うことが必要である．

前述したように，木村らは，要求仕様の一貫性をツールにより検証することで要求仕様の安定化を図ることに取り組んだ事例を報告した［Kimura2014］．これは，設計要素の一つであるアクターにフォーカスし，ベテラン技術者による，アクターの表記ゆれを防止するための検証ノウハウを「用語不一致検証ルール」と「定義漏れ検証ルール」の機能を備えたツールとして実現した点が特徴である．例えば，表1のアクター定義と，図1のシナリオ記述が要求仕様書内に記述されているとする．シナリオ記述に出現するアクターは，ユーザ，レジ係，顧客，店員，顧客（会員）である．これらは，表1のアクター定義には未定義である．

表1　要求仕様書内のアクター定義例

No.	アクター定義
1	ユーザ（管理者）
2	ユーザ（担当者）
3	オペレータ
4	マスタ管理者
5	運用管理者

要求仕様の検証では，これらが現行のアクター定義の表記ゆれなのか，アクター定義からの定義漏れなのかを確認する必要がある．仮に，現行のアクター定義表のまま，システムへのアクセスコントロールを実装したとする．実装後，「レジ係」というアクターが別途必要になった場合，「レジ係」のアクセスコントロール機能の拡張のため，作業の手戻りが考えられる．

［Kimura2014］では，要求仕様書内のアクター定義とシナリオ記述を対象に，アクター定義表からの定義漏れを指摘するアクターの定義漏れ検証と，「ユーザ」と「ユーザ（担当者）」などの表記ゆれを指摘する用語不一致検証を自動化し，検証レポートを生成した．しかし，［Kimura2014］における要求仕様の検証のノウハウは，アクター定義の一貫性にフォーカスした，特定企業の分析結果に基づいて定義された内容（図2）であり，様々な組織での活用を想定した一般化には至っていない．一貫性検証の対象となる，要求仕様を構成する設計要素の種類を拡張することや，様々な組織での活用を想定した柔軟性のあるツールのアーキテクチャの提供が必要と考えられる．

（シナリオ1）

1. ユーザは，レジに購入したい商品を持参し，ポイントカードを提示する．
2. レジ係は，商品情報およびポイントカード情報を端末から入力し，ポイント残高，残金を顧客に提示する．
3. ユーザは，提示された金額（残金）を支払う．
4. 店員は，支払われた金額（残金）を入金する．
5. システムは販売管理システムと連動してレシート（購入商品，金額，利用ポイント，残高ポイント）を出力する．ポイント付与ルールに基づいて残高ポイントがアップデートされる．
6. レジ係は，商品とレシートを顧客（会員）に渡す．

図1　要求仕様書内のシナリオ記述例

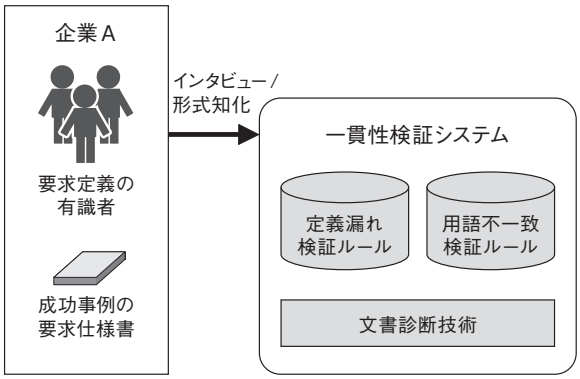


図2　一貫性検証支援ツールの構成

3　研究のアプローチ

2節の内容を踏まえ，自然言語で記述された要求仕様に対して，要求仕様の品質特性である「一貫性」に着目した検証手法のノウハウの形式知化とツールサポートに関する研究課題を以下に整理する．

課題1：要求仕様の一貫性検証知識の形式知化
課題2：要求仕様の一貫性検証の支援ツールの開発
課題3：要求仕様の一貫性検証の支援ツールの評価

以降，各課題の解決策のアプローチを整理する．

3.1　課題1：要求仕様の一貫性検証知識の形式知化

先行事例［Kimura2014］では，自然言語で記述された仕様説明内における「アクター」を対象とし，「用語不一致検証ルール」と「定義漏れ検証ルール」の2種類の対象企業に固有の検証知識の形式知化と支援ツールによる自動検証にとどまっていた．本研究では，研究成果を広く展開するため，検証知識の抽出範囲を複数企業に拡張する．また，検証の対象を要求仕様書全体に一般化した上で，検証の観点を，アクターに加えて「データ」「画面」「振る舞い」に拡張する．そして，複数企業の有識者インタビューを実施し，設計要素の一貫性に関する検証知識を抽出し，それらを共通部分と可変部分に仕分け，検証ルール及び辞書として形式知化する．

3.2　課題2：要求仕様の一貫性検証の支援ツールの開発

課題1で定義した検証ルール及び辞書を組み込んだ，要求仕様の一貫性検証を支援するツールを開発する．開発するツールは，要求仕様書内で言及されている，「アクター」「データ」「画面」「振る舞い」の記述が，要求仕様書内の記述と整合していることを検証する．検証の種類としても，用語定義完全一致，NGワード検証，NGワードを妥当な用語に置換するシナリオ自動生成の機能を追加する．開発するツールの考え方と構成を図3に示す．

［Kimura2014］では，技術や環境の変化に伴う検証ノウハウの拡張の方式については言及されていない．本研究では，検証ルール並びに辞書の共通部分を共有すると共に，各社が容易にそれらの維持管理と拡張を実施可能にするため，検証のエンジン，検証機能，検証ルール，辞書を独立させたアーキテクチャとする．更に，各社が独自に検証エンジンの拡張を可能とするために，形態素解析エンジンはOSSを用いる．

3.3　課題3：要求仕様の一貫性検証の支援ツールの評価

実システム開発で用いられた要求仕様書に対して適用評価を行い，有効性を検証する．

課題1で開発した検証知識を組み込み，課題2で開発したツールを，課題1でインタビューした有識者及びその関係者に提示し，適用評価し，検証のノウハウが妥当に具現化されていることを確認する．評価は，インタビューを実施した複数企業に対して，複数の視点から確認することとする．一貫性の検証にあたっては企業において実際に開発に用いられた仕様書を適用し評価する．

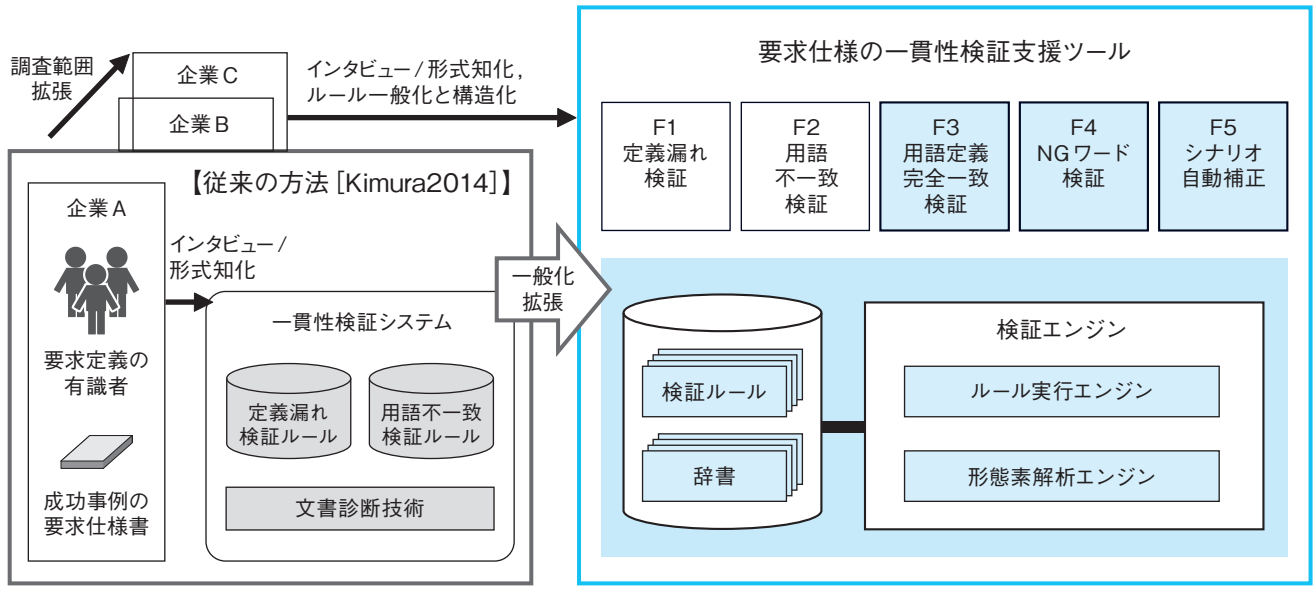


図3　要求仕様の一貫性検証支援ツール

4　要求仕様の一貫性検証知識の形式知化

先行研究[Kimura2014]の成果を踏まえ，既存の仕様書の分析，及び4社5部門の有識者にインタビューを行い，「アクター」「データ」「画面」「振る舞い」の一貫性に関する要求仕様の検証ノウハウを分析し整理した．分析・整理の結果得られた検証ルールと辞書の構造を図4に示す．

要求仕様の一貫性検証ルールは，定義漏れ，用語不一致，用語完全一致，NGワードの4つの検証ルールから構成した．これらのルールに基づき，要求仕様書を検証するには，自然言語で記述された要求仕様から設計要素を特定する必要がある．「アクター」「データ」「画面」「振る舞い」の識別ルールは，要求仕様に出現する名詞句や動詞句からそれぞれの用語を特定するルールである．各設計要素の識別ルールの実行には，各設計要素を特徴付ける用語を定義した識別辞書を用いる．また，同一意味で表現の異なる表記ゆれについても識別辞書と識別ルールを定義した．各ルールの定義を表2に示す．

例えば，アクター識別ルールには，「アクターとみなした用語の後ろに括弧で囲まれた文字列があれば，それらを連結してアクターとしてみなす」などが定義されている．要求仕様書内には，「ユーザ」「ユーザ(会員)」「ユーザ(非会員)」などのアクター名が記述されるが，本ルールに基づけば，これらはアクターとして特定することができる．要求仕様書内に，「ユーザ」しか定義されていなければ，用語不一致検証ルールによって「ユーザ(会員)」「ユーザ(非会員)」はユーザの表記ゆれ候補として指摘することができる．また，表記ゆれ辞書に，「ユーザ」と「利用者」が異音同義語として定義されていれば，「ユーザ」は「利用者」の表記ゆれとして指摘できる．

設計要素別の識別辞書の定義例を以下に示す．各設計要素では，「○○者」「○○ユーザ」などの表記や，辞書に定義された用語の複合名詞を当該設計要素とみなす．例えば，シナリオ中に，「顧客(会員)」「顧客(非会員)」などと記述されていれば，アクターを記述する辞書の「顧客」と「会員」の複合名詞として，これらアクターをアクター用語とみなす．

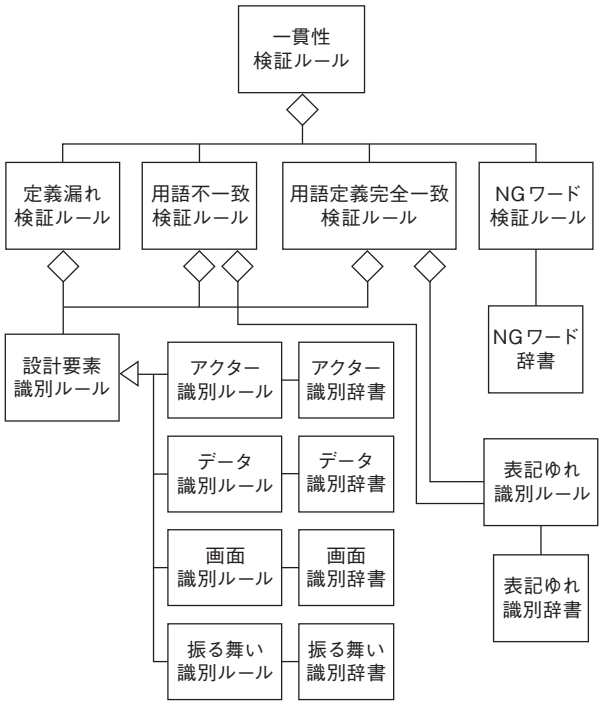


図4　検証ルールの構造

表2　検証ルールの説明

No.	ルール	ルール説明
1	定義漏れ検証	要求仕様書中に定義されていないにもかかわらず，本文中で利用されている設計要素を検証する．
2	用語不一致検証	要求仕様書では表記ゆれした形式で定義されている設計要素の用語不一致（表記ゆれ）を検証する．
3	用語定義完全一致検証	要求仕様書にて利用されている設計要素は，要求仕様書中に定義されており，かつ，定義された設計要素は，要求仕様書内で1回以上利用されていることを検証する．
4	NGワード検証	要求仕様書に出現するNGワードを検証する．NGワードはNGワード辞書にて定義する．
5	アクター識別	要求仕様書からアクター用語を識別する．アクター用語として特徴付ける単語はアクター識別辞書にて定義する．
6	データ識別	要求仕様書からデータ用語を識別する．データ用語として特徴付ける単語はデータ識別辞書にて定義する．
7	画面識別	要求仕様書から画面用語を識別する．画面用語として特徴付ける単語は画面識別辞書にて定義する．
8	振る舞い識別	要求仕様書から振る舞い用語を識別する．振る舞い用語として特徴付ける単語は振る舞い識別辞書にて定義する．
9	表記ゆれ識別	要求仕様書の複数の用語から表記ゆれ関係を識別する．表記ゆれ関係の用語は，表記ゆれ識別辞書にて定義する．

●**アクター識別辞書**：者，部，部門，会社，局，課，グループ，チーム，組織，ユーザ，会員，顧客，客，お客様，社員，従業員，員，委員，メンバ，オペレータ，運用者，管理者，監督者，人，市，市民，市職員，受託業者，オフィス，国，都道府県，市町村，業者，署，係，ユーザー，メンバー，オペレーター，市，町，村，都，道，府，県

●**データ識別辞書**：情報，データ，オブジェクト，帳票，書，ドキュメント，票，ファイル，調書，リスト，ID，コンテンツ，結果，パスワード

●**画面識別辞書**：画面，スクリーン，ディスプレイ，ページ，ウェブサイト，ホームページ，メッセージ

●**振る舞い識別辞書**：する，実施，実行，管理

●**NGワード**：全て，あらゆる，既存と同じ

●**表記ゆれ辞書**：利用者，ユーザ，ユーザー

5　要求仕様の一貫性検証支援ツールの開発

課題1を通して得られた，検証ルールと辞書に基づき，開発したツールの機能を表3に示す．本ツールは，F1～F4までの検証機能とF5のシナリオ自動補正から構成した．検証ルールと辞書は，組織や対象ドメインにより異なると考えられるため，拡張可能性を考慮し，検証ルール，辞書，検証エンジンを分離したアーキテクチャとした．更に辞書はテキストファイルで提供した．検証ルール及び検証エンジンの構成を図5に示す．検証エンジンは，ルールを実行するエンジンと，自然言語で記述された要求仕様を解釈する形態素解析エンジンで構成する．検証実行では，TemplateMethodパターンを用いて，設定ファイルに基づき，文書処理，用語抽出，検証，検証結果出力の一連の動作を実行する．各組織によって，用語抽出のエンジンを独自のものに変更するような場合には，検証実行クラスを別クラスに派生させて定義することを想定している．

本ツールは，Windows® 8.1 OS環境の下で動作可能とし，形態素解析ソフトウェアMeCab［MeCab］を用いた．開発言語はRuby［Ruby］であり，実装したクラス数は63，ステップ数は約4.4Kとなった．開発期間は6カ月，開発工数は約8人月，このうち，検証ルールと辞書の開発工数は約2人月である．

利用者が設定ファイルで書き分けることで，機能F1～F5を使い分ける．ツールは，記述された設定ファイルに基づいて，必要な検証ルールを選択・適用し検証を行う．検証結果は，検証レポート及び改善シナリオとして出力される．なお，入力シナリオ，設計要素定義とあるのは，要求仕様書に記述されたテキストファイルである．出力する検証レポート及び改善シナリオもテキストファイルである．

アクター用語の用語不一致検証のみの検証を行う条件を設定ファイルにおいて定義した場合の，一連の処理の流れを図6に示す．

表3　要求仕様の一貫性検証支援ツール機能一覧

ID	機能および概要
F1	定義漏れ検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，それぞれ，対応する定義表に定義されていることを確認し，未定義であれば指摘する．
F2	用語不一致検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，対応する定義表と別の表現（表記ゆれ）で記述されている場合に，その表記ゆれを指摘する．
F3	用語定義完全一致検証
	検証対象の仕様（シナリオ）及び，「アクター」「データ」「画面」「振る舞い」などの定義表を入力として，シナリオに出現する，「アクター」「データ」「画面」「振る舞い」などが，対応する定義表に定義されていること，かつ，「アクター」「データ」「画面」「振る舞い」などの定義表に定義された各要素が，シナリオ中に1回以上出現していることを確認し，未定義または出現しない場合にそれを指摘する．
F4	NGワード検証
	検証対象の仕様（シナリオ）及び，NGワード定義表を入力として，NGワードの定義表に定義された用語がシナリオ中に出現していれば，それを指摘する．
F5	シナリオ自動補正
	検証対象の仕様（シナリオ）及び，NGワード定義表を入力として，NGワードの定義表に定義された用語がシナリオ中に出現していれば，同じくNGワード定義表に定義された置換候補用語でシナリオを置換し，改善シナリオを生成する．

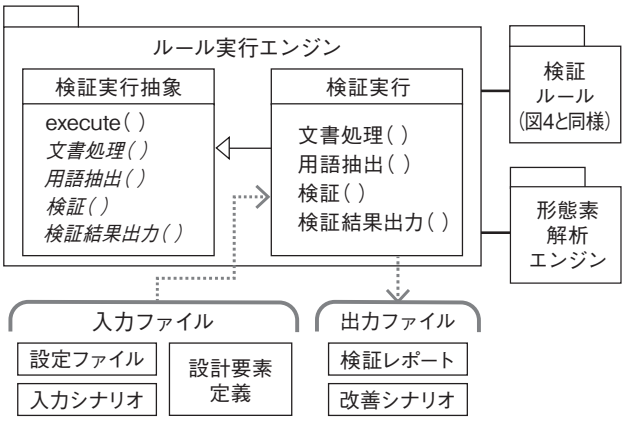


図5　検証ルール及び検証エンジンの構成

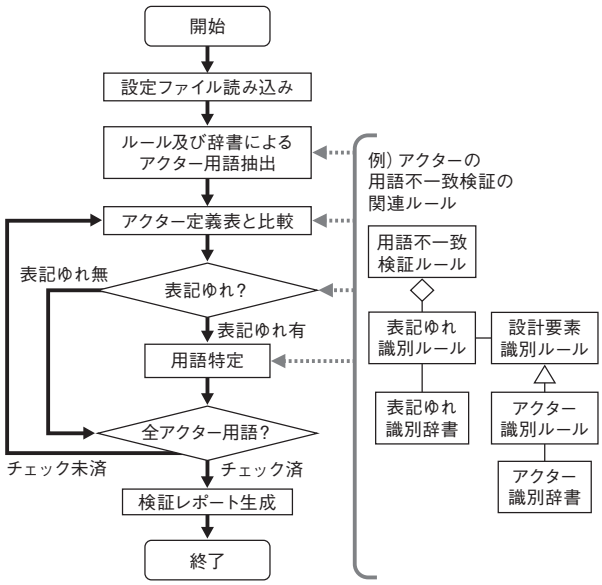


図6　アクターの用語不一致検証のフロー

6　要求仕様の一貫性検証支援ツールの評価

実システム開発で用いられた以下2件の要求仕様書に対して，要求仕様の一貫性検証支援ツールを用いて検証を実施した．これらは日本語で記述されたドキュメントである．それぞれの文字数は(a) 17,635，(b) 66,048である．

(a)某ウェブサイトリニューアル事業要求仕様書
(b)某制度システム構築・運用等業務調達仕様書

ツール評価は主に2つのステップで実施した．

ステップ1） ツールが組み込んだ検証ルールの意図に沿って，適切に指摘を行っていることを確認した．そのために，上記仕様書に対して，設計要素：アクター，データ，画面，振る舞い及びNGワードを対象に，定義漏れ検証，用語不一致検証，用語完全一致検証，NGワード検証，シナリオ自動補正を著者らの人手により実施し，その結果と，ツールによる検証結果を突き合わせて確認した．

ステップ2） 研究課題1の際にインタビューした有識者及び関係者に対して，ステップ1)の結果を提示して再インタビューを実施した．インタビューにおいて，検証のノウハウが妥当に具現化されていること，ツールによる検証結果の妥当性，ツールの有効性，操作性，適用可能性について評価した．

6.1　ステップ1)の結果

図7は仕様書(b)を本ツールで検証した結果の抜粋である．図7において，仕様書内の「交付申請情報」はデータ定義に定義されていないため，不一致として検出できることを示している．なお，「交付申請情報」が「交付金申請・決定情報」と同義語である(つまり表記ゆれである)ことまでは，現状のツールでは検出できる状態にはない．これは，文字列が部分一致するか，表記ゆれ辞書に記述するかによって，表記ゆれ候補を抽出しているためである．

表4にツールの定義漏れ，用語不一致，用語完全一致に関する再現率，適合率を示す．表記ゆれの特定と，統一化のための指針の抽出は100%には至らぬものの，平均して，再現率90%以上，適合率82%以上，F値0.84～1.00の結果を得た．NGワード検証は，NGワード辞書で定義した用語は100%特定でき，あらかじめ用意したNGワードに対する置換用語を用いてシナリオ自動生成が行えた．

仕様書(a)のアクターの定義漏れ，表記ゆれ検証では，ツールによる誤指摘が10件あるものの，53件の定義漏れ／表記ゆれの指摘が行えた．ツールによる適切な指摘としては，「管理者」「閲覧者」「利用者」など，アクターとして定義されていない，あいまいなアクター名が仕様書内に記載されていたことが含まれ

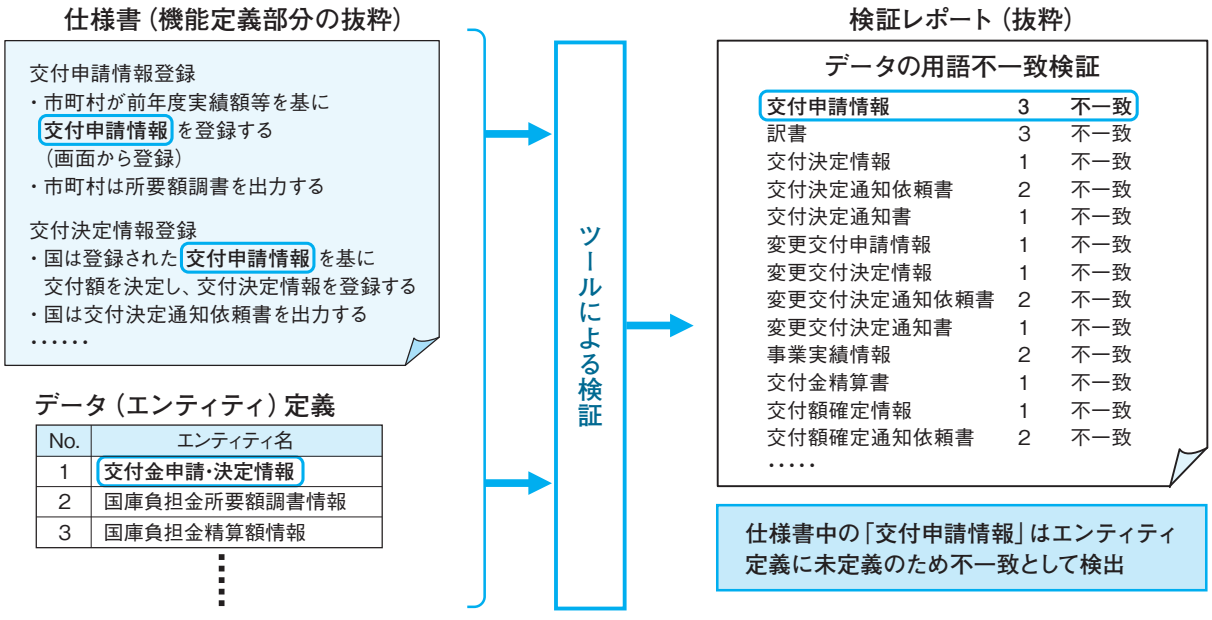


図7　仕様書(b)の検証結果例の抜粋

表4　検証結果抜粋

No.	仕様書	検証名	検証対象	人手の指摘数	ツールの指摘数	指摘漏れ数	誤指摘数	一致指摘数	再現率(%)	適合率(%)	F値
1	(a)	定義漏れ検証	アクター	62	63	9	10	53	85	84	0.84
2			データ	63	64	0	1	63	100	98	0.99
3			画面	30	31	2	3	28	93	90	0.91
4			振舞い	550	659	8	117	542	99	82	0.90
5		表記ゆれ検証	アクター	62	63	9	10	53	85	84	0.84
6			データ	63	64	0	1	63	100	98	0.99
7			画面	30	31	2	3	28	93	90	0.91
8			振舞い	550	659	8	117	542	99	82	0.90
9		完全一致検証	アクター	62	65	9	12	53	85	82	0.83
10			データ	63	64	0	1	63	100	98	0.99
11			画面	30	31	2	3	28	93	90	0.91
12			振舞い	550	659	8	117	542	99	82	0.90
13	(b)	定義漏れ検証	アクター	139	147	13	21	126	91	86	0.88
14			データ	209	210	12	13	197	94	94	0.94
15			画面	15	15	0	0	15	100	100	1.00
16			振舞い	1,370	1,644	13	287	1,357	99	83	0.90
17		表記ゆれ検証	アクター	139	147	13	21	126	91	86	0.88
18			データ	209	210	12	13	197	94	94	0.94
19			画面	15	15	0	0	15	100	100	1.00
20			振舞い	1,370	1,644	13	287	1,357	99	83	0.90
21		完全一致検証	アクター	139	147	13	21	126	91	86	0.88
22			データ	253	254	12	13	241	95	95	0.95
23			画面	70	70	0	0	70	100	100	1.00
24			振舞い	1,370	1,644	13	287	1,357	99	83	0.90

ている．なお，アクターとして，「高齢者・障がいのある人」「作成したページについての変更権限を持たない課」「専門知知識を有しない者」「コンテンツの所有部署」「コンテンツの担当(作成)部署」「担当部署」「所有部署」に対して，ツールは「アクター」用語として特定できず，定義漏れや表記ゆれの検証で指摘がすり抜ける結果となった．「部署」をアクター用語の識別辞書に登録し，再現率を向上させる対策が考えられる．

データに関しては，形態素解析エンジンMeCabの特性により，「内訳書」が「内」と「訳書」に分割して認識されたため，「所要額市町村別内訳書」「変更所要額市町村別内訳書」「精算額市町村別内訳書」は，データ用語として特定できなかった．これらはMeCabの特性に起因し，現状ではツール側での改善は困難なため，ユーザマニュアルにて識別困難であることを注意喚起する対策が考えられる．

6.2　ステップ2）の結果

本ツールによる仕様書の検証結果について，4社5部門の有識者にインタビューを行った．インタビューによれば，2つの実事例による検証結果は一定のレベルに達している，という評価を得た．すなわち，検証レポートの内容は有意義な指摘が提示されており，数千～数万文字からなる仕様書において，数十件以上の未定義やあいまいな表現の指摘を人手で実施することは困難であるが，本ツールを用いることで，初級の技術者でも検証レポートの指摘事項に従って仕様書を改善していくことが現実的に実施可能であると考えられる．今回は，既に開発が完了している案件に対しての要求仕様書への検証レポートである．したがって，定義漏れや表記ゆれの指摘が存在していたとしても，実際の開発において悪影響がどの程度あったかどうかまでは分析できていない．しかし，検証レポートに出力される抽出用語一覧を確認することで，出現頻度の高い用語や，用語の使われ方のパターンが把握でき，仕様書に対する理解が深まるという点で有用であるというコメントを得た．

また，検証レポートは数十ページにわたるボリュームになることや，テキストデータの羅列により直観的かつ効率的に指摘事項を特定して改善に取りかかることが困難であることから，検証レポートの見せ方の工夫が重要との指摘を受けた．また，検証レポートに基づき仕様書を改善する過程で，利用者が指摘結果を採用したかどうか，それに基づき修正を実施したかどうかを学習し，指摘事項を効率的に対処する仕組みが今後の課題であることを確認した．

本研究の成果は，主にはエンタープライズ系システムの企画立案，要求定義工程において，要求仕様書，提案書，調達仕様書，操作仕様書，製品取扱い説明書，業務マニュアルなどを対象に，初級技術者がセルフチェックするほか，次のシーンでの活用が考えられることを確認した．

- **担当者が作成した仕様書の管理者によるチェック**
- **検証ルールや辞書を用いた知識継承や人材育成**
- **抽出したアクターを用いたステークホルダ分析**
- **要求仕様書と基本設計書の検証レポートの比較による設計要素**

の不統一個所や設計漏れのチェック

- **プロダクトライン型開発の各モデル間の比較による，共通部分と可変部分の切り分けチェック**

7　考察

本研究では，要求仕様の一貫性検証のために，NGワードの指摘に加えて，システム開発で重要な設計要素となる「アクター」「データ」「画面」「振る舞い」の定義漏れと表記ゆれの指摘に着目した．実仕様書によるツールの適用評価や有識者インタビューにより，仕様書から設計要素を抽出すること，それらの定義漏れや表記ゆれを指摘することは，要求仕様の内容確認や誤った理解防止に効果的と考えられる．

本ツールでは，あらかじめ基本的な検証ルールと辞書を提供した．要求仕様の検証の経験のない組織でも，スムーズに要求仕様の検証に取り組むことが期待できる．様々な分野の要求仕様書に対して本ツールによる検証を実施するには，識別辞書や表記ゆれ辞書をカスタマイズする必要がある．本ツールはこれらの辞書をテキストファイルで提供している．辞書の拡張は，用語が確定していれば，1人H程度で対応可能と考えられる．なお，提供した検証ルールでは不足で，検証ルールを追加するには，検証ルールクラスの派生が必要なため，記述量にもよるが拡張コストは0.25人月程度必要と考えられる．

組織別に定義した設計要素の識別辞書は要求仕様の一貫性検証に関する専門知識であり，組織の資産として知識継承に活用可能である．また，本ツールはプロダクトライン型開発でのモデル間の仕様書の整合性の検証など，様々な活用が期待できる．

本ツールの評価者は，検証ルール及び辞書の構築の際にインタビューをした有識者である．したがって，本ツールにより初級技術者でも的確な一貫性検証が可能であることの十分な確認には至っていない．今後，評価者及び評価案件の対象を拡大し，ツールの妥当性の評価を継続し，評価過程で明らかにした課題の解決に取り組んでいく．

要求仕様の一貫性検証支援ツールを構築する際に利用した形態素解析エンジンの制約や，入力する仕様書の記述内容や対象組織での業務ルールや商習慣により，ツールが出力する検証結果には，誤指摘や当該組織にとり不要な指摘が15%程度含まれる場合がある．本ツールの適合率の向上には，形態素解析エンジンの条件や対象組織に固有の特性を考慮し，指摘対象からは除外すべき用語情報の学習と，ツールによる検証を対象組織用に進化させる技術開発に取り組むことが重要と考えられる．また，現状の検証レポートは数十ページに及ぶ場合がある．管理者や分析者の仕様書の改善作業を効率化するため，検証結果を図式化，定量化するなどが必要と考えられる．

8　まとめ

本研究では，要求仕様の品質特性である「一貫性」に着目し，「アクター」「データ」「画面」「振る舞い」の設計要素が，要求仕様書で一貫した定義で記述されていることを確認する検証ノウハウを，ルールと辞書として形式知化し，それら知識に基づく要求仕様の一貫性検証支援ツールを実現した．

本研究において，実システム開発で用いられた仕様書に対して適用評価を行い，有効性を検証した．有識者に対してインタビューを行い，当システムの有効性，妥当性，適用可能性の観点から考察し，システム開発の企画や計画段階での設計要素候補の抽出や，プロダクトライン型開発でのモデル間の仕様書の整合性の検証への活用が期待できることを明らかにした．

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「2015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ある．

【参考文献】

[Cohn2004] M. Cohn, User Stories Applied: for Agile Software Development. Redwood City, CA, USA: Addison Wesley, 2004.
[Cockburn2000] Alistair Cockburn, Writing Effective Use Case, Addison Wesley, 2000.
[Hiranabe2013] 平鍋健児，野中郁次郎，アジャイル開発とスクラム～顧客・技術・経営をつなぐ協調的ソフトウェア開発マネジメント，翔泳社，2013.
[ISOIECIEEE2011] ISO/IEC/IEEE 29148:2011, Systems and software engineering — Life cycle processes — Requirements engineering, 2011.
[JISA2011] 情報サービス産業協会，REBOK企画WG，要求工学知識体系，近代科学社，2011.
[JISA2014] 情報サービス産業協会 REBOK企画WG，要求工学実践ガイド，近代科学社，2014.
[Kimura2014] 木村隼人，北川貴之，位野木万里，要求仕様書の品質向上に向けた活動報告 ～一貫性検証の形式知化および自動化～，情報サービス産業協会 SPES2014 経験報告 要求工学S4a，2014.
[Kouno2010] 河野哲也，猪塚修，藤森麻紀子，本間周二，茂中義典，キーワードベースドレビュー，ードキュメントのあいまいさや不備に着目したレビュー手法ー，ソフトウェアテストシンポジウムJaSST 2010，http://jasst.jp/archives/jasst10e/pdf/C2-3.pdf，2010.
[Kuno2012] 久野綾子，平尾英司，五藤智久，仕様書の曖昧性を検出するツールの試作と評価，電子情報通信学会，電子情報通信学会総合大会講演論文集 2012年_情報・システム（1），27，2012-03-06，2012.
[Lucassen2015] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper, Forging High-Quality User Stories: Towards a Discipline for Agile Requirements, Proc. of IEEE 23rd International Requirements Engineering Conference, pp. 126-135, 2015.
[MeCab] MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://taku910.github.io/mecab/#download
[Pohl2015] Klaus Pohl and Chris Rupp, Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant, 2nd Edition, Rocky Nook, 2015.
[Rasmusson2011] Jonathan Rasmusson（著），西村直人（監訳），角谷信太郎（監訳），近藤修平（訳），角掛拓未（訳），アジャイルザムライ―達人開発者への道，オーム社，2011.
[Ruby] Ruby: https://www.ruby-lang.org/ja
[Wang2014] X. Wang, L. Zhao, Y. Wang, and J. Sun, The Role of Requirements Engineering Practices in Agile Development: An Empirical Study, Proc. of the Asia Pacific Requirements Engineering Symposium, ser.CCIS., vol.432, pp.195–209, Springer, 2014.

実践的保証ケース作成方式



山本 修一郎※1　森崎 修司※1　渥美 紀寿※2

システムの安全性や高信頼性を保証するために必要となる保証ケースの作成を実践的に支援できる保証ケース作成方式が必要である．このため，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成手法，客観的保証ケースレビュー法からなる実践的保証ケース作成方式を提案し，その有効性を実験的に確認する．

A Practical Approach to develop assurance case

Shuichiro Yamamoto※1, Shuji Morisaki※1, Noritoshi Atsumi※2

Practical assurance case development methods are necessary to validate safety and dependability of critical systems. In this paper, a model based unified assurance case creation method, code based assurance case development method and objective assurance case review method are proposed and experimentally evaluated.

1　はじめに

航空宇宙，自動車や医療機器など高い安全性が要求される複雑なシステムを実現するために，保証ケース（assurance case）の作成が必要とされるようになっている．例えば，自動車分野で導入が必要とされるISO26262機能安全規格などで，安全性に対する保証ケースである安全性ケースの作成が開発プロダクトだけでなく開発プロセスに対しても義務付けられている．このため，多様なモデルに対する統一的な保証ケースの作成法の研究，既存コードに対する保証ケースの作成法の研究，保証ケースの客観的なレビュー法の研究が必要である．これらの研究の必要性を以下に述べる．

(1) 多様なモデルに対する統一的な保証ケース作成法の必要性

システムの安全性を保証するためには，システムを構成するソフトウェアのモデルだけでなく，システムの利用モデルや構造モデルを明らかにすることにより，これらのモデルの安全性を保証ケースで確認する必要がある．これらのモデルを記述する言語には，BPMN（Business Process Modeling Notation），UML（Unified Modeling Language）やSysML（Systems Modeling Language）などがある．BPMNでは業務プロセスをモデル化できるのでシステムの利用・運用プロセスを定義できる．現代のソフトウェア開発で広く普及しているUMLで記述できるモデルには，ユースケース図やクラス図，シーケンス図，状態図，アクティビティ図などが含まれる．要求図などによりUMLを拡張したSysMLは組込みシステム向けに普及してきている．これらの多様なモデルに対して保証ケース作成法を用意できないと，産業界への保証ケースの適用は進展しない．

保証ケースの作成を容易化するために，アーキテクチャやプロセス構造に基づく保証ケースパターンが報告されている．しかし，多様なモデルに対する統一的な保証ケースの作成法については明らかになっていない．

これまで，著者らはネットワーク装置監視システムの保証ケース作成へのアーキテクチャパターン適用評価について研究を進めてきた[Yamamoto2013a]．しかし，一般的なアーキテクチャに対する保証ケース作成法については明らかになっていなかった．また，これまで，多様な工程生産物に対するモデルに基づく50以上の保証ケースパターンを具体化してきた[Yamamoto 2013b]．しかし，このような個別的な方法では新たなモデルが登場するごとに新たな保証ケース作成法が必要になるという問題がある．

このため，本論文では，任意のモデルがモデルを構成する概念要素とその関係に基づいていることに着目して，「統一的保証ケース作成法」を提案する．もし，統一的保証ケース作成法がなければ，モデルごとに保証ケースを作成する手法を開発する必要があり，冗長であるだけでなく，非効率であるという問題がある．このため，任意のモデルに共通する概念要素とその関係に基づいて保証ケースを階層的に分解することによって，「統一的保証ケース作成法」を提案する．ここで，任意のモデルから保証ケースを作成できるという点でモデルごとに個別的に保証

※1　名古屋大学　　※2　京都大学

ケースを作成する手法ではないということから「統一的」という用語を使用している．

(2) 既存コードに対する保証ケース作成法の必要性

ソフトウェアコードを再利用することで，ソフトウェア開発を効率化するだけでなく，開発されたソフトウェアの品質を向上できる可能性がある．この場合，再利用対象コードの安全性を保証する必要があるため，コードに対する保証ケースの作成法が必要である．

既存コードの安全性を保証するには，上述したようなモデルに対する保証ケースだけでなく，モデルが作成されていないような既存コードに対する保証ケースが必要である．しかし，現状では，コードに対する保証ケース作成法は確立されていない．

(3) 保証ケースの客観的なレビュー法の必要性

既に作成された保証ケースもあることから，保証ケースを適切にレビューする方法が必要である．保証ケースの具体的なレビュー法については，保証ケースが正しい構成規則を用いて記述されていることや成果物と保証ケースとの追跡性並びに網羅性を確認する手法が報告されている．また，主張，証拠などの不完全性や依存性についての定義がある．しかし，保証ケースを構成する主張を記述する用語の適切性や一貫性，主張を下位の主張に分解する際の網羅性，主張に対する証拠の十分性などの観点から，保証ケースの妥当性を内容に踏み込んで客観的に確認するためのレビュー法は確立されていない．

なお，広辞苑[Shinmura1991]によれば「実際に行動するさま」が「実践的」であると定義されていることから，本論文では，手法を提案するだけでなく，実際の適用例を示すことで，提案した手法が実践的であることを明らかにする．

2 関連研究

システムの安全性を確認するために，安全性ケース(Safety case)，保証ケース(Assurance case，アシュアランスケース)やディペンダビリティケース(Dependability case)が注目されている．例えば，重要システムの実行中に優先順位の高い要求を満足することを確認するために，ディペンダビリティケースが必要とされている[Jackson2008]．このため，GSN (Goal Structuring Notation)[Kelly1997][Kelly1998][Yamamoto 2009]を用いてこれらを記述する方法が提案されている．

保証ケースでは，ゴール(主張)，戦略，前提(コンテクスト)，根拠(証跡，ソリューション)によって，システムのディペンダビリティに関する議論を構造化して確認することができる．しかし，実際のシステム開発プロジェクトの担当者による保証ケースの作成を容易化するためには，表記法だけでなく，システム開発プロセスやシステム開発文書に即した具体的な保証ケース作成手順を提供する必要がある．

このため，保証ケースの作成を支援するために，開発手順，開発文書に基づく作成法，議論分解手法，ツールについて研究されている．

保証ケースの開発手順の研究には，Kellyによる6段階開発手法[Kelly1997][Kelly1998]とEuropean Organisation for the Safety of Air Navigationが制定している安全性ケース開発マニュアル[European Organisation2006]がある．

6段階手法では，保証ケースを作成するために，①ゴールを識別する，②ゴールを記述するための基礎を定義する，③ゴールを満足させるための戦略を識別する，④戦略を記述するための基礎を定義する，⑤戦略を吟味する，⑥基本的な証跡を識別するという6段階の手法をKellyが提案している[Kelly1997][Kelly1998]．安全性ケース開発マニュアルでは，安全性ケースをレビューするためのチェックリストを提案している．しかし，これらの開発手順の研究では，具体的な開発文書を対象としていないという問題がある．

開発文書に基づく保証ケース作成法の研究には，複数のシステムから構成されるシステム(System of Systems, SoS)に対する手法[Despotou2007]と，複合的な開発運用文書に基づく手法[Denney2012]が提案されている．前者の手法では，SoSの開発過程で，システム分析，ゴール要求抽出，代替設計案の識別，矛盾点の解消に基づいて保証ケースを作成できる．後者の方法では，概念文書，設計書，運用手順書，ハザードリストに基づいて安全性ケースを作成できる．しかし，これらの開発文書に基づく保証ケースの作成法では，特定の開発運用文書を対象としており，任意のモデルに対して適用できないという問題がある．

保証ケースを作成する際に必要となる議論分解の観点として，システム構成や機能構成，属性構成などが整理されている[Bloomfield 2010][Yamamoto2013a]．しかし，議論分解の代表的な例が提示されているだけであり，具体的な開発運用工程生産物に基づく保証ケースの作成法は提示されてはいないという問題があった．

DEOSプロジェクト[DEOS2013]では保証ケースを編集できるエディタを開発している．しかし，開発運用工程生産物に対して保証ケースを手作業で編集して作成する必要があるという問題があった．

このように，従来研究では，対象となる任意のシステム構造に基づいて，保証ケースを作成する具体的な手法が明確ではなかった．

3 保証ケース作成支援方式

以下では，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成法，保証ケースの客観的なレビュー法からなる実践的保証ケース作成方式(図1)について，目的，課題，手法の概要について述べる．

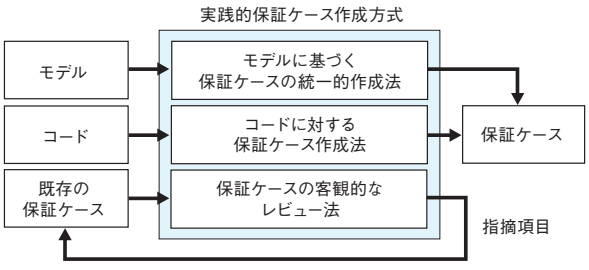


図1　実践的保証ケース作成方式の概要

3.1 モデルに基づく保証ケースの統一的作成法

【目的】

多様なモデルに対する保証ケースの作成を容易化するために，任意のモデルに対して適用できる保証ケースの統一的な作成法を確立する．

【課題】

これまで，モデルごとに保証ケースの分解パターンを用意していた．しかし，多様なモデルに対して個別に分解パターンを用意するのは限界があった．

【手法】

どのようなモデルも，必ずモデル要素と要素関係によって定義されている．したがって，モデルを保証する場合，モデル要素と要素関係に基づいて，保証ケースを一般的に分解することができる（モデル構成に基づく分解であることからこの分解をアーキテクチャ分解と呼ぶ）．また，モデル要素とその関係が期待される品質特性を持つことについて，品質特性の下位特性ごとに分解して保証することができる（品質特性の構成に基づく分解であることから，この分解を品質特性分解と呼ぶ）．更に，モデル要素とその関係が必要な品質特性を持つ上でのリスクとその対策が実施されていることに対して分解することができる（リスク対策についての分解であることから，この分解をリスク分解と呼ぶ）．このように，対象とするモデルが品質特性を持つことを，アーキテクチャ分解，品質特性分解，リスク対策分解に従って統一的に保証ケースを作成できる．

まず「システム」に着目して「システムが特性を満たす」という主張をアーキテクチャ分解パターンで構成要素とその関係に分解する．次に，「特性」に着目して，下位特性に分解できる場合は，特性分解パターンで下位特性に分解する．最後に，下位特性に着目して，下位特性のリスクに基づいて，リスク分解パターンで分解することにより，リスク対策の証拠を明らかにすることができる．

モデルは，構成要素とその関係によって定義される．例えば，ユースケース図の場合，アクターとユースケースがモデルの構成要素，アクターとユースケースの相互作用がモデルの関係である．同様にオープングループのアーキテクチャフレームワークであるTOGAF（The Open Group Architecture Framework）［TOG2011］のアーキテクチャ記述言語ArchiMate［Josely2013］［Yamamoto2016b］によるモデルも同様に，構成要素とその関係で定義できる．例えば，ビジネスアーキテクチャ（BA）モデルには，ビジネスイベント，ビジネスプロセス，ビジネス対象，ビジネス情報形式などの構成要素と，トリガ関係，利用関係，実現関係などがある．

したがって，図2に示すように，モデルに対して統一的なモデル分解パターンを構成できる．まず，モデルの構成に基づいて，モデルの構成要素とその関係で主張を分解する．次に，構成要素の種類並びに，構成要素関係の種類で主張を分解する．更に，構成要素と構成要素関係の実体に基づいて主張を分解する．

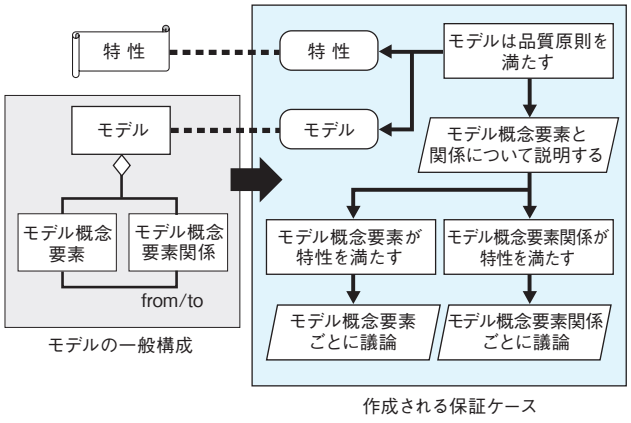


図2　モデル構造に基づく保証ケース作成の考え方

上述した着想に基づいて集合論の記法を用いることにより，定式化した保証ケース生成アルゴリズムを以下に示す［Yamamoto2015b］［Yamamoto2015c］．

【定義】保証ケース生成アルゴリズム

モデルをA＝<ConceptSet, RelationshipSet>とする．

ここで，ConceptSet＝{<Nc, Cc>｜Ncは概念名，Ccは概念種別}，RelationshipSet＝{<Nr, Cr>｜Nrは関係名，Crは関係種別}である．

Aについて，以下の集合を定義する．ここで，xはAの概念の具体例，rはAの関係の具体例である．

Aに出現する概念の集合

ConceptCategory（A）＝{ Cc｜<x, Cc>はAのConceptSetの要素}

Aに出現する関係の集合

RelationshipCategory（A）＝{ Cr｜<r, Cr>はAのRelationshipSetの要素}

Aに出現する概念のインスタンス集合

ConceptInstance（A）＝{ x｜<x, C>はAのConceptSetの要素}

Aに出現する関係のインスタンス集合

RelationshipInstance（A）＝{ r｜<r, Cr>はAの RelationshipSetの要素}

これの集合に基づいて，保証ケースを以下の手順で作成する．

最上位の主張を，「モデルAが特性を満足する」

第2レベルの主張に分解するための説明を「モデルAの概念と関係について説明する」として

第2レベルの主張を「モデルAの概念が特性を満足する」と「モデルAの関係が特性を満足する」とする

第3レベルの主張に分解するための説明ノードのラベルを「概念種別ごとに説明する」と「関係種別ごとに説明する」として

第3レベルの主張をConceptCategory（A）とRelationshipCategory（A）の要素ごとに以下のようにして作成する．ここで，CはAに出現する概念種別，CrはAに出現する関係種別である．

「モデルAの概念種別Cが特性を満足する」

「モデルAの関係種別Crが特性を満足する」

第4レベルの主張に分解するための説明ノードのラベルを「概念ごとに説明する」と「関係ごとに説明する」として第4レベルの主張をConceptInstance（A）とRelationshipInstance（A）の要素ごとに作成する

「モデルAの概念種別Cのインスタンスxが特性を満足する」

「モデルAの関係種別Crのインスタンスrが特性を満足する」

第5レベルの主張を，第4レベルの主張に対するリスクに次のようにして作成する

「モデルAの概念種別Cのインスタンスxが特性を満足する上でのリスクに対応できる」

「モデルAの関係種別Crのインスタンスrが特性を満足する上でのリスクに対応できる」

（アルゴリズム終わり）

このアルゴリズムに基づいて，図3に示す簡単なユースケース図から保証ケースが作成できる例を以下に示す．図3のユースケース図には，アクターaとb，ユースケースwがある．アクターとユースケースの関係として，r1とr2がある．ここで概念要素がa，b，w，概念間の関係がr1とr2である．このアルゴリズムに基づいて作成された保証ケースを図4に示す．ただし，紙幅の関係から，図4では，リスク対応については省略している．

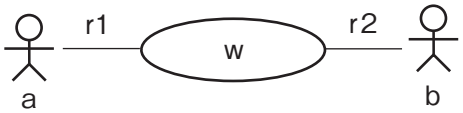


図3　ユースケース図の例

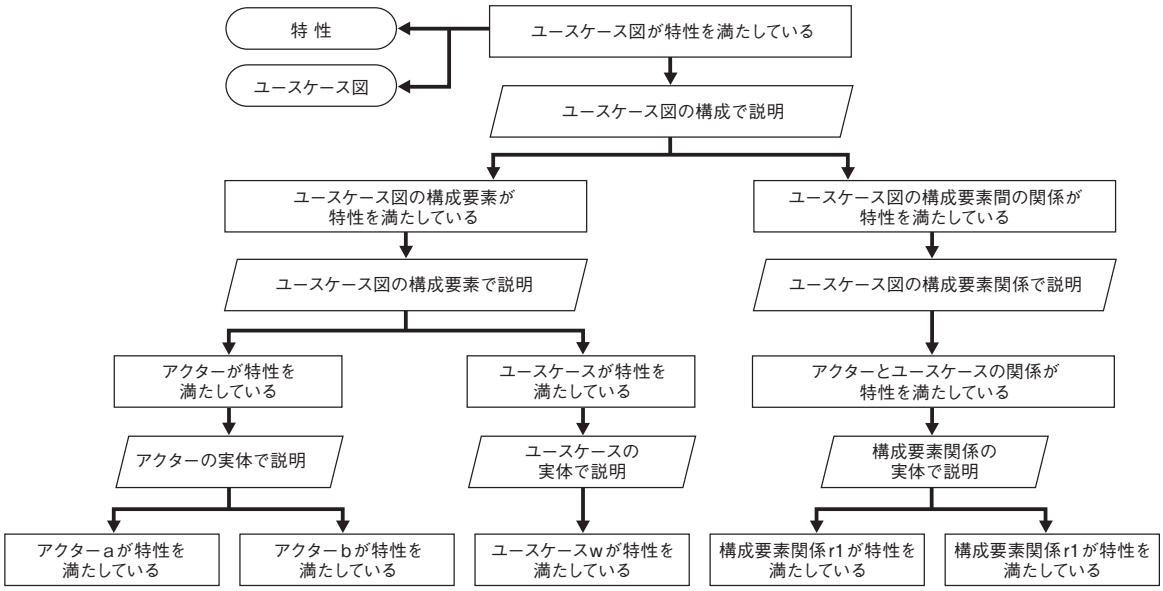


図４　ユースケース図に対応する保証ケース

また，このアルゴリズムに基づいて，ArchiMateのBAに対して保証ケースを作成できることについては文献[Yamamoto 2015b][Yamamoto2015c]を参照されたい．このように，上述したアルゴリズムを用いることにより，任意のモデルに対して統一的に保証ケースを作成できる．

3.2　コードに対する保証ケース作成法

【目的】

モデルが記述されていない既存コードに対する保証ケース作成法を具体化する．

【課題】

既存システムを保証する場合，モデルが定義されていないことが多いため，コードに対する保証ケースの作成法がないという問題があった．

【前提】

既存コードに対して，仕様はあるが，モデルはないことを前提としている．オープンソースソフトウェアなどの場合には，仕様とコードはあるがモデルは存在しない場合が多いことから，このような前提を置くこととした．また，仕様からモデルを作成して統一的保証ケース作成法によって保証ケースを作成する場合，保証すべき特性を「安全性」．概念要素を「関数の引数」，概念要素の関係を「引数関係」として，仕様の安全性を確認する保証ケースを作成することになる．したがって，モデルに基づく保証ケース作成法では，証拠の作成までは具体化していない点，コードに対する保証ケースではない点，コードの安全性という具体的な特性を考慮していない点が異なる．したがって，コードとその仕様がある場合には，モデルを作成するのは冗長であり，本節で述べるような直接的な保証ケース作成法が必要である．

【手法】

コードに対する保証ケースの作成では，対象となるコードへの入力と出力に着目する．入出力仕様に対して，対応するコードでは指定された入力に対する処理によって出力を作成する必要がある．

この点に着目して，コードに対する保証ケースでは，図５に示すように，「引数を説明」と「引数関係を説明」からなる入出力分解層で必要な入出力に対する主張を分解することにより，対応するコードについて具体化すべき証拠層を用意する．次いで，コードを探索して対応するコードがあれば，証拠として明記する．もし対応するコードがなければ，証拠がないことになり，仕様を実現するコードが抜けていることを検出できる．この考え方を，証拠に基づく欠陥摘出原理(Defect Detection by Evidence, DDBE)と呼ぶ．

保証ケースに基づくコード保証手順は以下のようになる．

【手順１】入出力仕様に基づき入出力制約を作成

【手順２】入出力制約に基づき，証拠が未定義になっている保証ケースを作成

【手順３】対応するコード断片を証拠に用いて，保証ケースを説明

【手順４】保証ケースの主張を説明するコード断片がない場合，コードの欠陥として指摘

（手順終わり）

保証ケースの証拠を上述した手順に従って説明できない場合，コードには欠陥があることになる．つまり，この手順はコードの欠陥を摘出する具体的な方法を与えている．この基本的な考え方を図示すると，図５のようになる．

この保証ケースでは，「コードが安全である」という主張を最上位ゴールとする保証ケースを入出力引数の関係仕様を前提として分解している．ここで，入出力引数や入出力引数関係に対する制約が引数仕様並びに引数関係仕様である．引数関係仕様には，入力引数間の関係，入力引数と出力引数の関係，出力引数間の関係がある．このような引数仕様並びに引数関係仕様が成立することを確認するコードがあるとき，「引数や引数関係が安全である」と言う．

最下位の証拠が，入力引数，出力引数，入出力引数の関係についての安全性を確認するために必要なコードを確認した結果に対応している．

3.3　保証ケースの客観的なレビュー法

【目的】

保証ケースレビュー観点の分類，観点に応じたレビュープロセスを定式化する．

【課題】

既存の保証ケースをレビューする場合，客観的なレビュー法

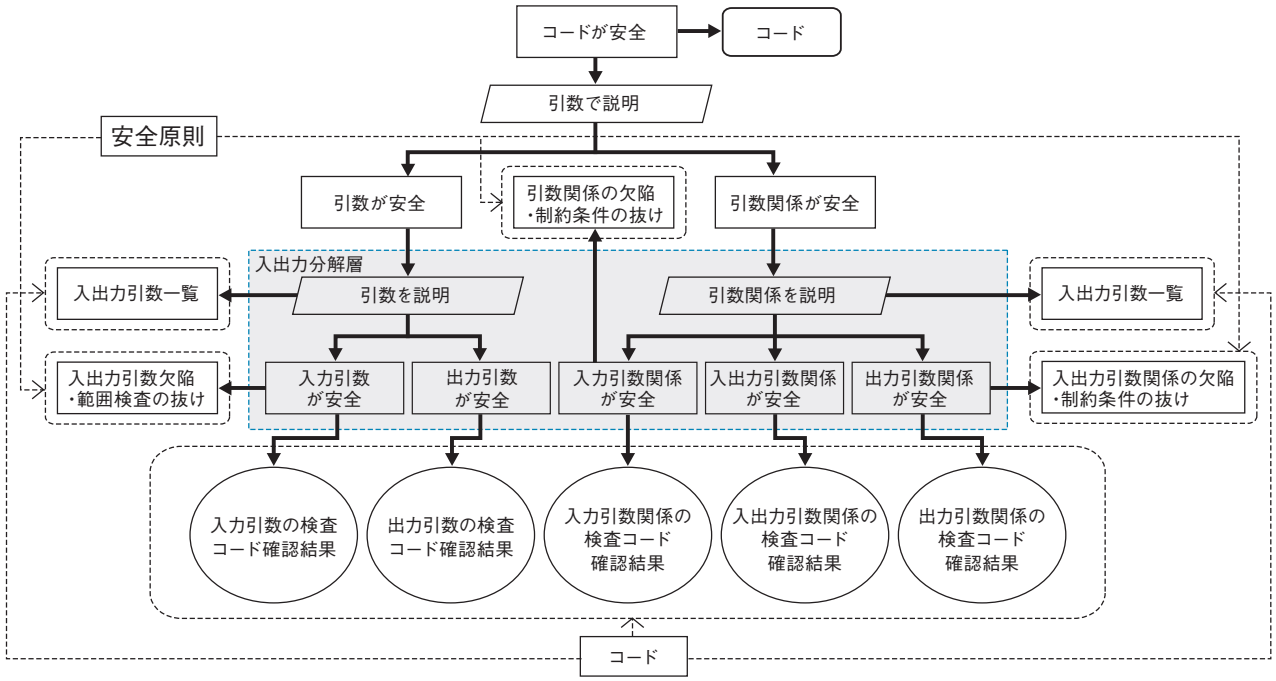


図5　コードに対する保証ケース作成法

が明確ではないため，属人的なレビューになりやすいという問題があった．とくに，保証ケースの主張では，「システムが安全である」などのように，日本語文を用いるため，用語関係があいまいになりやすいという問題があった．

【手法】

保証ケースのレビュープロセスを，理解，問題識別，原因分析，欠陥修正から構成し，理解段階で，保証ケースからシステミグラム［Boardman2008］を作成することにより，システミグラム上で問題識別，原因分析を客観的に実施できるようにする．また，この結果に基づいて，保証ケースの欠陥修正を容易化できる．

ここで，システミグラムは，名詞をノードとし，動詞をノード関係で表現する単純な図式である．システミグラムを用いて，保証ケースを構成する主張を理解することができる．

保証ケースの妥当性を保証対象の内容に踏み込んで客観的に確認するためのレビュー法を確立するため，システミグラムを用いて保証ケースが対象とする構成情報を図式化することにより，保証ケースの内容に基づくレビュー法について述べる［Yamamoto2015d］．

システミグラムのノードは，人工物，エージェント，重要属性を表す．ノードリンクは，ノード間の関係を表す．ノードの包含関係では，ノードの内部ノードで，下位ノードやノード属性を表すことができる．

システミグラムを用いた保証ケースのレビュー法では，レビューする保証ケースの主張を，以下の2種類に分類する．

［特性主張］＜対象＞が＜特性＞を満足している
［対策主張］＜対策＞によって＜リスク＞に対応している

特性主張は対象Sと特性Pの構成要素に従って，下位の特性主張に分解される．また，特性主張と対策主張には，上位の特性主張を下位の対策主張が説明するという関係がある．このとき，分解で指定される前提はリスクRの構成要素である．

対策Mについての主張は証拠Eによって説明される．すなわち，このような対策主張は最下位の主張である．特性主張を構成する対象と特性に対して，対象の内部状態として，「特性を満足している」を持つシステミグラムを対応付ける．対策主張の場合，「対象が特性を満足する」という上位の主張を実現するために，「対策によってリスクに対応している」という主張が必要になる．

上述した関係をまとめると，図6に示すようなシステミグラムになる．

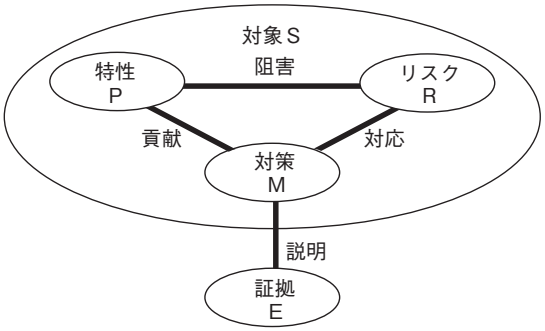


図6　対象，特性，リスク，対策，証拠とシステミグラム

システミグラムを用いて，保証ケースのレビュー手順を，①内容理解，②問題識別，③原因究明，④修正から，次のように構成できる．

［内容理解］保証ケースに対するシステミグラムを上述の方法で作成する
［問題識別］レビュー観点（表1）に基づき，用語関係，特性関係を分析して，問題を識別する
［原因究明］対応項目の欠落・誤りを検出することによって，問題原因を特定する
［修　　正］原因究明された問題について，欠落・誤り項目を補完・訂正する

表1　レビューの観点

観 点	定 義	指 標
完全性	必要な項目が含まれていること	不足項目数
明確性	あいまいさがないこと	不明項目数
適切性	不必要な項目が含まれていないこと	孤立項目数
追跡性	根拠が明確であること	追跡不能項目数

4　適用実験

4.1　統一的保証ケース作成実験

定義で示したアルゴリズムに従って，保証ケースを作成する支援ツールUC2CT(Unified Context to Claim Tool)を試作した(図7)．このツールでは，XMLで記述したモデル定義情報を入力として，保証ケース情報を図8に示すような表形式画面に生成して編集できる[Yamamoto2015a]．

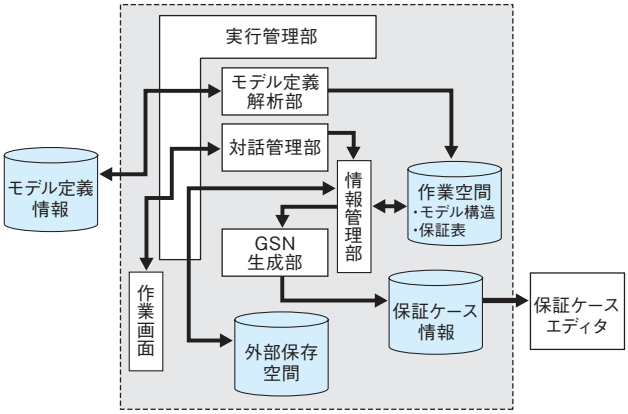


図7　保証ケース作成支援ツールの構成

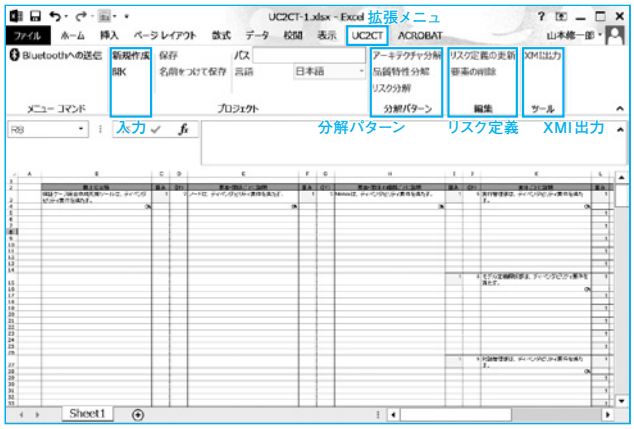


図8　UC2CT画面例

出力された保証ケース情報を保証ケースエディタに入力することにより，モデルに基づく保証ケースを確認できる(図9)．このツールによって任意のモデルに対して保証ケースを半自動的に作成できる．

図9　生成された保証ケースの編集画面例

試作ツールを用いた保証ケース作成実験の結果を図10に示す．

実験対象としたモデルは，保証ケース作成支援ツールUC2CTのシステム構成を示す図7である．このシステム構成図に対する保証ケースの規模は，主張218個，戦略53個，前提47個，証拠165個となった．ツールを使わずに保証ケースを作成した時間は275分であった．これに対してツールを使用して保証ケースを作成した時間は47分であった．この結果，ツールの利用により，保証ケース作成時間を約5.6倍(約82%削減)に向上できることが判明した．

また，後者の作成時間にはモデル定義情報，品質特性情報，リスク対策情報を記述するためのXML定義時間28分を含む．もし，モデルからXML定義を自動抽出できれば，保証ケース作成時間が約14.5倍になることも判明した．なお，この19分の内訳は，ツールへのXML入力，アーキテクチャ分解，品質特性分解，リスク対策分解，保証ケースエディタ変換を含むことを注意しておく．

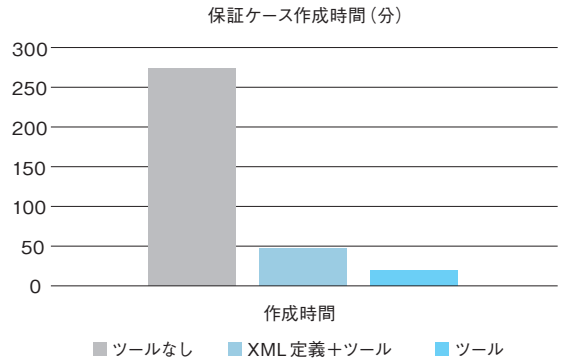


図10　保証ケース作成時間の比較

4.2　コードに基づく保証ケース作成実験

3.2節で述べたコードに対する保証ケース作成法の有効性を確認するために，具体的なコードを対象として保証ケースを作成する実験を実施した[Miyabayashi2015]．実験対象の概要は，次のようである．

対象とする入力仕様は，SSL/TLS Protocol V 1.0で，3584行からなる文章である．対象とするコードは，オープンソースのOpenSSL 1.0.1j s3_clnt.cである．コード行数は，3469であった．

被験者は，名古屋大学の学部4年生が1名である．

まず，入力仕様分析によって，11個のTBE(To Be Explained，証拠としてのコードの断片に対応する説明項目)を摘出した．対応するコードの断片を発見できれば，それを証拠としてTBEを置き換える．この入力仕様分析に要した時間は10時間．入力仕様に基づく保証ケース作成の時間は5時間だった．

次いで，Open SSLのコードを探索して10件のTBEに対する証拠として対応するコードの断片を具体化したが，残り1件のTBEについては，対応するコードがなかった．この1件の具体化できなかったTBEに対応する保証ケースの部分を図11に示す．

この図では，「セッションIDが空であるか，クライアントが送信した値と一致しない場合は安全である」という主張を説明しようとしている．この場合，対応するコードで鍵の入力条件を確認して，「暗号スィートの方式が，入力鍵を有効としない場合の処理は安全である」ことを説明する必要がある．このため，鍵の入力条件が不適切であるから，「エラー処理を中断していることを説明」する必要がある．この理由は，「不適切な入力処理の続行を禁止」する必要があるからである．したがって，主張「暗号スィートの方式が，入力鍵を有効としない場合に処理を中断している」に対応する証拠となるコードがなくてはならない．

ところが，このTBEに対応するコードが確認できなかった．このTBEが具体化できなかった部分が実際にOpen SSLの脆弱性に対応していた．このコードに基づく保証ケースの説明時間は8時間であった．

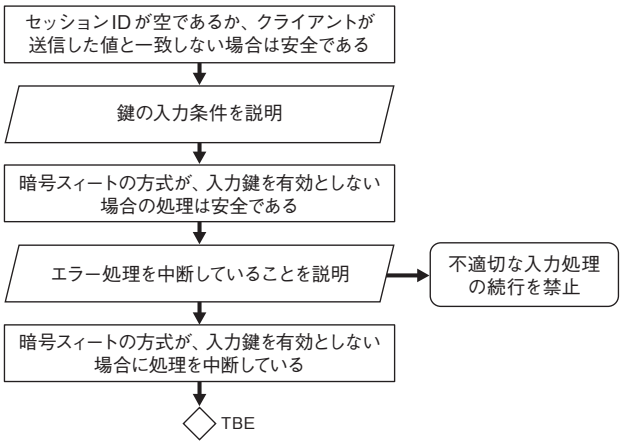


図11　TBEに対する保証ケース

4.3　保証ケースレビュー実験

同一の保証ケース事例を対象として，保証ケースを直接レビューする場合(直接法と呼ぶ)と，システミグラムを用いてレビューする場合(間接法と呼ぶ)とを比較する実験を実施することにより，間接法による保証ケースレビュー手順の有効性を確認する[Yamamoto2016a]．

実験対象システムは，電気ポットの加熱制御システムである．このシステムに対して，「加熱が安全である」という主張に対する保証ケースを14名の被験者(大学院学生)が作成した．

作成された14件の保証ケースを，保証ケースについて経験を持つ2名の大学院学生が次のようにしてレビューした．まず，直接法によって保証ケースをレビューした．次いで，システミグラムを用いた間接法で保証ケースをレビューした．

ここで，システミグラムを用いた間接法による保証ケースレビュー手順は以下のようになる．

a)保証ケースからシステミグラムを作成する手順を定義しておき，

b)この手順に従って被験者の作成した保証ケースからシステミグラムを作成して，

c)作成されたシステミグラムに対して，レビュー基準を用いて指摘項目を作成した．

図12に被験者による保証ケースから，レビュー実施者が作成したシステミグラムの例を示す．このように，保証ケースをシステミグラムに書き直すことでノード間の接続関係を明確化できることが分かる．

直接法と間接法による保証ケースのレビュー結果を表2に示す．ここで，数値の単位は件数である．また，2名のレビュー実施者が共に指摘した場合，共通欄の件数，そうでない場合差異欄の件数として計上した．

表2　レビュー実験結果

項目	直接法		間接法	
	共通	差異	共通	差異
完全性	0	0	15	10
明確性	0	63	53	6
適切性	0	0	4	53
追跡性	10	5	4	53
合計	10	68	76	122

間接法における適切性と追跡性の件数が一致しているのは，孤立ノードについては，上位ノードとの関係が追跡できなくなるためである．またシステミグラムを用いたレビューでは，対象とした保証ケースで多くのリスクが挙げられているものの，リスクに対する対策が一つも挙げられていない事例が多いことが判明した．また，対策が挙げられていないため，保証ケースの証拠ノードで示された内容が，不必要な証拠として指摘された．この結果，指摘件数が多くなったと思われる．

表2の結果から，直接法と間接法の平均指摘件数と共通指摘率を図示すると図13のようになった．ここで，平均指摘件数と共通指摘率は表2の合計欄の数値に基づいて下式で求めた．また14は被験者数である．なお，平均差異件数を求めるために，レビュア数2で除した．

平均指摘件数＝(共通合計件数＋(差異合計件数／2))／14
共通指摘率＝共通指摘件数／合計指摘件数

間接法の平均指摘件数は，約9.79，共通指摘率は約55.7%である．これに対して直接法の平均指摘件数は約3.14，共通指摘率は約22.7%となった．

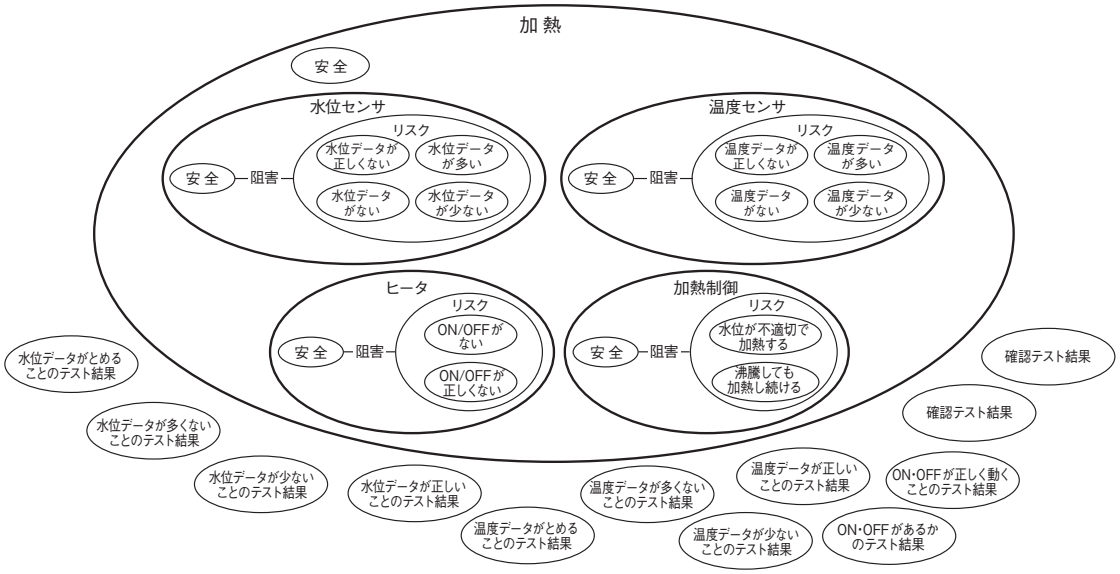


図12　保証ケースから作成したシステミグラムの例

この結果から，間接法が直接法よりも指摘件数で約3.1倍，指摘内容の共通率で約2.4倍になることが判明した（図13）．

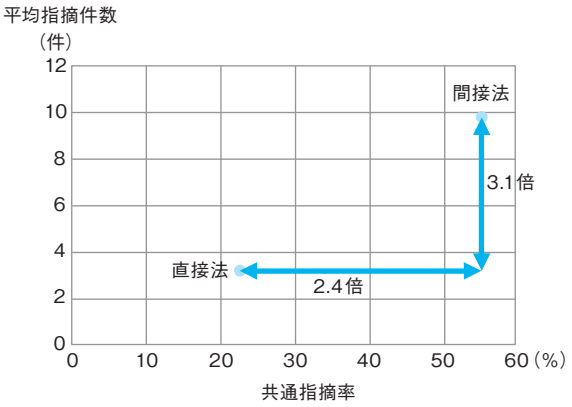


図13　保証ケースレビュー法の比較結果

5　まとめと今後の課題

本論文では，システムの安全性や高信頼性を保証するために必要となる保証ケースの作成を実践的に支援できる保証ケース作成方式を提案し，その有効性を実験的に確認した．提案した実践的保証ケース作成方式は，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成法，客観的保証ケースレビュー法からなる．

モデルに基づく保証ケースの統一的作成法では，要素と要素関係を持つ任意のモデルから統一的に保証ケースを作成できる保証ケース作成アルゴリズムを定式化した．また，このアルゴリズムに基づき，保証ケース作成支援ツールUC2CTを試作した．UC2CTでは，XMLにより，モデル，品質特性，リスク対策を定義しておくことにより，保証ケース情報を半自動的に生成できる．生成された保証ケース情報を保証ケースエディタに入力することで保証ケースを作成できる．

このUC2CTを用いて保証ケースを作成する実験で評価した結果，手動でモデルから保証ケースを作成する場合に比べて，UC2CTを用いることにより，作成時間を約5．6倍に向上できることを明らかにした．

コードに対する保証ケース作成法DDBEをオープンSSLに適用した結果，保証ケースで12個の具体化すべき証拠を識別した．この証拠に対して，コードを分析した結果11個の証拠にはコードが対応したが，対応すべきコードがない証拠を1件検出した．このコードが欠落した証拠がオープンSSLの脆弱性の欠陥原因であることを確認した．したがって，もし，オープンSSLのコードレビューにDDBEを適用していれば，オープンSSLの脆弱性を未然に防止できた可能性が高いことから，DDBEの有効性を確認できた．

客観的保証ケースレビュー法では，システミグラムに基づく保証ケースレビュー法を具体化した．また，保証ケースのレビュー指標として，完全性，明確性，適切性，追跡性を定義することにより，システミグラムを用いた間接法による保証ケースのレビューと，直接，保証ケースをレビューする直接法を比較するレビュー実験を実施した．この結果，間接法が直接法よりも，レビュアによる指摘内容の共通性が約2.4倍高いこと，また指摘件数が約3.1倍高いことを明らかにした．

本研究の制約として，保証ケース作成実験で対象としたモデルとコードがそれぞれ1件であること，また保証ケースレビュー実験の課題も1件であることなどの限界がある．ただし，これらの制約を考慮したとしても，各手法の有効性が明確になっており，ほかの事例に対しても，本手法の有効性を実証できると考えている．

今後の課題として，上述した制約の解消と，産業界における適用事例の拡大がある．また，本論文で提案した手法の教育と共に，3手法を統合した保証ケース作成支援環境の実現に向けた取り組みが必要である．

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「2015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ある．

【参考文献】

[Shinmura1991] 新村出編，広辞苑，岩波書店，第四版，1991.
[Kelly1997] T. Kelly, A Six-Step Method for the Development of Goal Structures, York Software Engineering, 1997.
[Kelly1998] T. Kelly, Arguing Safety, a Systematic Approach to Managing Safety Cases, PhD Thesis, Department of Computer Science, University of York, 1998.
[Yamamoto2009] 山本修一郎，要求工学第58回アシュアランスケースとGSN，ビジネスコミュニケーション vol.46, No.8, pp.68-71, 2009.
[Jackson2008] D. Jackson, et al, Software for dependable systems– sufficient evidence?, NATIONAL RESEARCH COUNCIL, 2008.
[European Organisation2006] European Organisation for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, 2006.
[Despotou2007] G. Despotou, T. Kelly, Design and Development of Dependability Case Architecture during System Development, proceedings of the 25th International System Safety Conference (ISSC), System Safety Society, 2007.
[Bloomfield 2010] R. Bloomfield and P. Bishop, Safety and assurance cases: Past, present and possible future – an Adelard perspective, procedings of the 18th Safety-Critical Systems Symposium, pp.51-67, 2010.
[DEOS2013] DEOSプロジェクト，http://www.jst.go.jp/crest/crest-os/
[Denney2012] E. Denney and G. Pai, I. Habli, Perspectives on Software Safety Case Development for Unmanned Aircraft, proc. 42nd Annual IEEE/IFIP Intl. Conf. Dependable Systems and Networks(DSN2012), pp.1-8, 2012.
[TOG2011] The Open Groupe, TOGAF Version 9.1, Van Haren Publishing, 2011.
[Josely2013] A. Josely, A., et al., ArchiMate® 2.0, A Pocket Guide, The Open Group, Van Haren Publishing, 2013.
[Boardman2008] J. Boardman, and B. Sauser, Systems Thinking: Coping with 21St Century Problems, CRC Press, 2008.
[Yamamoto2013a] S. Yamamoto, Y. Matsuno, An Evaluation of Argument Patterns to Reduce Pitfalls of Applying Assurance Case, 1st International Workshop on Assurance Cases for Software-intensive Systems, Assure 2013, pp.12-17, 2013.
[Yamamoto2013b] 山本 修一郎，主張と証拠，ダイテックオンデマンド出版，2013，978-4-86293-095-8.
[Yamamoto2015a] S. Yamamoto, Assuring Security through Attribute GSN, ICITCS 2015, pp.1-5, 2015.
[Yamamoto2015b] S. Yamamoto, An approach to assure Dependability through ArchiMate, Assure 2015, pp.50-61
[Yamamoto2015c] 山本修一郎，森崎修司，渥美紀寿，正田稔，モデルに基づく統一的保証ケース作成手法の提案，AI学会，KSN研究会，2015. 10.
[Yamamoto2015d] S. Yamamoto, An assurance case review method using Systemigram, AAA2015, 2015.
[Miyabayashi2015] 宮林 凌太，渥美 紀寿，森崎 修司，山本 修一郎，入力分析に基づくコード保証方法の提案，KBSE研究会，Vol.115, No.281, KBSE2015-39, pp.17-22, 2015.
[Yamamoto2016a] 山本修一郎，森崎修司，渥美紀寿，近藤純平，大林英晶，GSNレビュ実験と評価について，AI学会KSN研究会，2016, 3.
[Yamamoto2016b] 山本修一郎，現代エンタープライズ・アーキテクチャ概論 - ArchiMate入門 - , p.132, デザインエッグ社, 2016.7, ISBN-10:4865436804

2016年度はIPA第三期中期計画（2013年度～2017年度）の4年目の年にあたり、最終年度での着実な事業目標達成を見据えて活動を実施してきた。本特集では、「社会全体を支える情報処理システムの信頼性向上」を目指した2016年度のSECの主な成果を「IoT時代の安全安心」、「重要インフラ等システム障害対策」、「定量的プロジェクト管理」、「組込みソフトウェア」、「システム構築上流工程強化」、「ソフトウェア工学分野の先導的研究支援事業」、「プロモーション」の活動に分けて紹介する。

IoT時代の安全安心に向けて

SECソフトウェアグループリーダー　中尾 昌善

1 はじめに

IoT（Internet of Things）時代の到来を迎え、新ビジネス創出の動きが活発化している。一方で、IoT時代の機器／システム／サービス（以下では、これらを総称して「IoT」と呼ぶ）は、従来のような単一的で閉じた範囲で扱われるものではなく、複合的でオープンな環境を対象とするため、その安全安心にかかわるリスクの増大が懸念される。そこで、IPA/SECでは、IoT開発時のリスクを低減するための活動を推進しており、それらを以下に紹介する。

2 IoTの特徴

機器などが単独で持っていた本来機能に、通信機能を付加することにより、新しい価値のIoTが生み出される。更に、IoTとIoTがつながって新たなIoTを形成する。これは、SoS（System of Systems）という概念で捉えられている。このように、個々のIoTが複合的に融合し、アメーバ状につながりが拡大していくのがIoTの特徴の一つである（図1）。

3 IoT時代の安全安心に向けた活動

3.1 IoT開発に資するガイド類の策定

IoTは、一つの企業単独で開発できるとは限らず、異分野の企業の協業によって発展を遂げる可能性がある。その開発においては、品質の保持やリスクへの備えなど、分野を超えて互いに留意すべき事項への共通認識が必要となってくる。そこで、その留意点を取りまとめた「つながる世界の開発指針」と、そこから派生して必要となる各種ガイド類を作成し、産業界に展開する活動を行っている。

3.2 システムズエンジニアリングの推進

IoTのつながりによる新サービスの創出は、ビジネス構造を劇的に変化させる可能性を秘めている。一方で、従来の方法論にとらわれていると、その変化に柔軟に対応できず、逆にビジネスリスクをもたらす危険がある。そのパラダイムシフトに備えるための開発方法論として注目されているのが、欧州を中心に適用が進みつつあるシステムズエンジニアリングであり、我が国での適用を推進している。

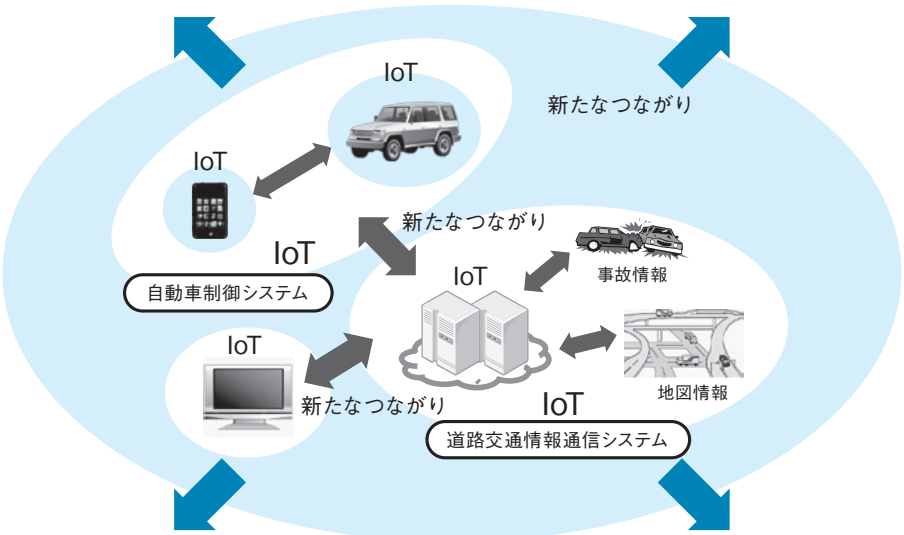


図1　つながりで拡大するIoTの世界

IoT時代の安全安心に向けて

「つながる世界の開発指針」の普及展開

SEC調査役 宮原 真次　SEC研究員 小崎 光義　SEC研究員 丸山 秀史
SEC研究員 遠山 真　SEC研究員 西尾 桂子

1 はじめに

IoT(Internet of Things)に向けた新しい製品・システム開発やサービスが創出され始めており、新しい価値の創造や利便性が高まりつつある。このような状況の中で、IPA/SECはIoTの安全安心を確保するための考え方を示した「つながる世界の開発指針」を2016年3月に公開した※1。本稿では、「つながる世界の開発指針」の産業界への普及展開の状況について報告する。

2 普及展開の状況

2.1 IoT政策への展開

国のIoT政策としてIoT推進コンソーシアムが設立され、その中の専門WGとしてIoTセキュリティWGが発足した。このIoTセキュリティWGに「つながる世界の開発指針」を提案し、IoT機器・システムのセキュリティの指針として全面的に「IoTセキュリティガイドライン」に採用された(2016年7月公開)※2。

2.2 産業界や企業への展開

IoTのセキュリティ強化を目指す業界団体や企業に「つながる世界の開発指針」の展開を働きかけ、協調した活動を推進した。その結果、一般社団法人重要生活機器連携セキュリティ協議会(CCDS)では、「つながる世界の開発指針」をベースにして、車載器、IoTゲートウェイ、金融端末(ATM)、決済端末(POS)の4つの分野のセキュリティガイドラインを作成し公開した(2016年6月公開)※3。

また、個別の企業に対しては、ITベンダや家電メーカ、組込みシステム機器メーカなどに働きかけ、「つながる世界の開発指針」をベースにしたチェックリストの運用や社内規定などに採用いただいた。

2.3 セミナー開催

「つながる世界の開発指針」を産業界へ周知するために、IPA主催のセミナーや業界団体と協調したセミナー、外部の展示会・セミナーなどで、講演を実施した。2016年度は、合計30回の講演を実施し、のべ395社に参加いただいた。また、民間や業界団体などが主催する展示会では、IPAブースを開設しブースプレゼンなどを通じて約5,000人の方々に「つながる世界の開発指針」の書籍を紹介した。IPAホームページに開設したダウンロードサイトからのDL数は、約7,000件の実績となった。

3 関連施策

3.1 IoT高信頼化機能への具体化

「つながる世界の開発指針」を開発現場に展開するために、IoT機器・システムを開発するときに考慮すべき安全安心を実現する12の機能要件と23の機能をまとめた。

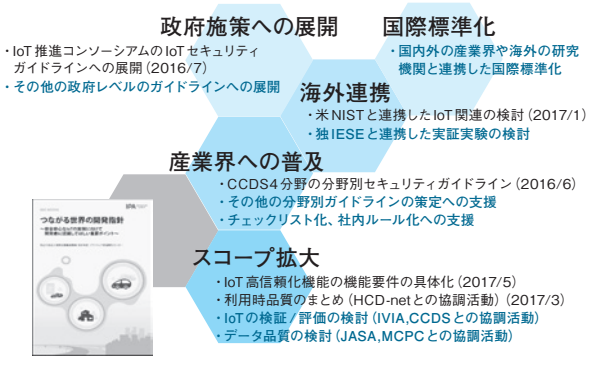
また、IoT高信頼化機能で取り上げた異常の監視機能の一部をFAシステムとスマートエネルギーシステムの連携モデルで実証実験を実施した(2017年5月公開)。

3.2 利用時品質の観点からの留意点抽出

IoTを利用する人や場面は、拡大の一途をたどるが、誤操作などで安全が損なわれることが懸念されている。そこで、利用者視点に立った安全安心なIoT製品・システムを開発するために、考慮すべき利用時品質の視点をまとめた(2017年3月公開)。今回まとめた利用時品質は「つながる世界の開発指針」にも反映し、改訂版を発行した(2017年6月発刊)。

4 今後の活動

2017年度は、業界団体との協調による更なる普及活動の加速と共に、国際標準化に向けた活動とIoTの品質にかかわる検証・評価の検討に取り組む予定である。



脚注

※1 http://www.ipa.go.jp/sec/reports/20160324.html
※2 http://www.meti.go.jp/press/2016/07/20160705002/20160705002.html
※3 https://www.ccds.or.jp/public_document/index.html

IoT時代の安全安心に向けて

「『つながる世界の開発指針』の実践に向けた手引き[IoT高信頼化機能編]」の策定

SEC研究員　小崎 光義　SEC研究員　丸山 秀史　SEC調査役　宮原 真次

1 はじめに

IoT(Internet of Things)時代に向け、各国の様々な産業分野においてIoT機器や関連システムの開発が進んでいる。しかし、安全安心の基準が異なるシステムが相互接続することで、当初は想定していなかったリスクが顕在化することも懸念されている。

IPA/SECは、安全安心なIoTの開発に向けた着眼点を示した「つながる世界の開発指針」(以降、「開発指針」と略記)を2016年3月に策定した。この「開発指針」を参考にしながら開発するには、「具体的な機能の解説が必要」という声に応えるために、「開発指針」のうち技術面での対策が必要になる部分を更に具体化し、「『つながる世界の開発指針』の実践に向けた手引き[IoT高信頼化機能編]」(以降、「実践に向けた手引き」と略記)を2017年5月に策定した。

2 取り組みの概要

「開発指針」では、安全安心なIoTの実現に向けての着眼点と対策例を示した。その中で、例として挙げている対策は物理対策、人的対策、管理対策、技術対策から成るが、「実践に向けた手引き」では、これらのうち開発時に活用できる技術対策にフォーカスして解説することとし、次のような工夫により、開発者により具体的なイメージを持っていただくことを目指した。

(1) 設計段階から考慮して欲しい要件とIoT高信頼化機能の具体例を解説

(2) 分野間で連携するユースケースと、リスクや脅威、機能定義や機能配置の具体例を掲載

3 「実践に向けた手引き」の内容

3.1 安全安心なIoT機器・システムの開発の要件と機能

IoT機器・システムの安全安心を確保するための機能を「IoT高信頼化機能」と定義した。IoT機器・システムのライフサイクル(図1)を考慮すると、保守・運用時の視点で必要となる機能を設計時に作り込むことが重要である。例えば、運用中にシステムの障害が発生した場合、検知や回復に関する機能が必要であり、それらの機能を設計工程で作り込むなどの対処を実施する。保守・運用中の対策としては、「予防・検知・回復」の3つが必要であるが、更に、IoT機器・システムは環境や構成が絶えず変化することから、サービス開始や接続時、及びサービス終了や廃棄時の対策が必要であるので、「開始」と「終了」を追加し、開発者が

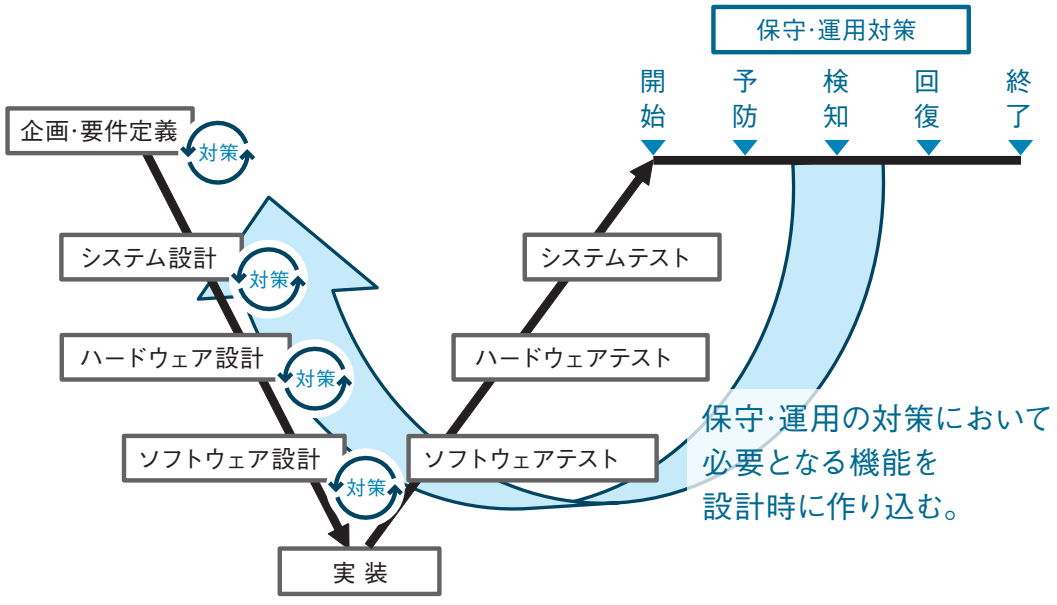


図1　ライフサイクル

設計段階から考慮すべき5つの要件を整理した。次に、それらの要件をソフトウェアで実装する場合に必要な12の機能要件に細分化し、更に、機能要件を実現するための具体的な23の機能を解説した(表1)。

また、IoT機器・システムについて、その機能配置として、エッジ層、フォグ層、クラウド層から成るIoTの構成(図2)を想定した。例えば、エッジ層では、リソースが少なく、コストがかけられないことが想定され、一律にIoT高信頼化機能を実装することが難しいので、それらのIoT特有の条件を考慮するためのポイントを解説した。更に、リスクアセスメントを行って必要な機能を搭載する手順についても解説を行った。

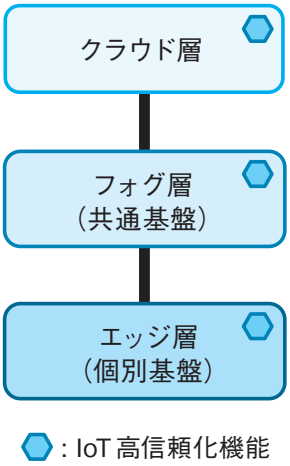


図2　IoTの構成

このように、ライフサイクルと機能配置の二つの軸で整理することにより、開発者が実装方法や必要な機能の網羅的な検討や、経済合理性や寿命を考慮した現実味のある検討に活用できると考えている。

3.2 ユースケースと、リスクや脅威、機能定義や機能配置の具体例

IoTの分野間の連携に着目した5つのユースケース(①車両と住宅の連携、②VPP(Virtual Power Plant)と分散型電源監視サービスとの連携、③宅内機器連携、④戸締り競合制御、⑤産業ロボットと電力管理の連携)を掲載した。

例えば、「戸締り競合制御」の例では、一つの住宅環境制御IoTシステムに、目的が相反する複数の制御系機能が接続された場合のリスク分析を行っている。この例では、複数の制御の競合の検知が監視機能として必要であることを説明している。

4 おわりに

「実践に向けた手引き」を活用いただくことによりIoTの安全・安心に寄与できると考えている。これまで、IoTに関連する各企業、業界団体、業界横断的団体に対して「開発指針」の普及展開を行ってきたが、今後は具体的な現場での活用に向けて本書を紹介していく予定である。「開発指針」は、多数のIoT関連の民間事業者が参画する「IoT推進コンソーシアム」が策定している「IoTセキュリティガイドライン」で採用されており、また、「IoTセキュリティガイドライン」は今後の国際標準化の提案のベースとしての活用が見込まれる。本書は国際標準化に向けた提案においてテクニカルリファレンスとしての提案も視野に入れて進めていく予定である。

「実践に向けた手引き」は以下のWebサイトで公開しているので、積極的に活用いただきたい。

https://www.ipa.go.jp/sec/reports/20170508.html

表1　IoT高信頼化の要件と機能

IoT高信頼化要件		IoT高信頼化を実現するための機能要件	対応するIoT高信頼化機能
開始	導入時や利用開始時に安全安心が確認できる	1. 初期設定が適切に行われ、その確認ができる	初期設定機能、設定情報確認機能
		2. サービスを利用するときに許可されていることを確認できる	認証機能、アクセス制御機能
予防	稼働中の異常発生を未然に防止できる	3. 異常の予兆を把握できる	ログ収集機能、時刻同期機能、予兆機能、診断機能、ウイルス対策機能
		4. 守るべき機能・資産を保護できる	アクセス制御機能、ログ収集機能、時刻同期機能、暗号化機能
		5. 異常発生に備えて事前に対処できる	リモートアップデート機能
検知	稼働中の異常発生を早期に検知できる	6. 異常発生を監視・通知できる	監視機能(競合の検知を含む)、状態可視化機能
		7. 異常の原因を特定するためのログが取得できる	ログ収集機能、時刻同期機能
回復	異常が発生しても稼働の維持や早期の復旧ができる	8. 構成の把握ができる	構成情報管理機能
		9. 異常が発生しても稼働の維持ができる	診断機能、隔離機能、縮退機能、冗長構成機能
		10. 異常から早期復旧ができる	リモートアップデート機能、停止機能、復旧機能、障害情報管理機能
終了	利用の終了やシステム・サービス終了後も安全安心が確保できる	11. 自律的な終了や一時的な利用禁止ができる	停止機能、操作保護機能、寿命管理機能
		12. データ消去ができる	消去機能

IoT時代の安全安心に向けて

IoTの高信頼化に向けた分野間連携実証実験

SEC研究員　丸山 秀史　SEC研究員　小崎 光義　SEC調査役　宮原 真次

1 はじめに

IPA/SECでは、IoT(Internet of Things)時代の安全安心な製品を開発するための「つながる世界の開発指針」を2016年3月に策定し、その中の異常の早期検知の対策、並びに接続機器の信頼性確認の対策の有効性を実証実験で確認した※1。2017年5月には上記開発指針の内容を具体化した「『つながる世界の開発指針』の実践に向けた手引き[IoT高信頼化機能編]」(以下「実践に向けた手引き」)を策定し、IoTの重要な特徴の1つである分野間連携におけるリスク分析の例を提示している。今回の実証実験では、その内容を踏まえ、分野間連携システムを対象としたリスクへの対策例を示し、それらの実現性を確認した。

2 実証実験の方針

分野間連携システムでは、機器やシステムの異常が、異なる分野のシステムに影響を及ぼす可能性がある。また、異なる分野のシステム間の相互作用により、単一分野のシステムでは発生しない異常が起こり得る。そこで、分野間連携システムでの異常をより早い段階で検知するための対策を検討し、実証実験システムで実装した。

分野間連携システムとしては、産業ロボットシステムとそれが稼働している工場の電力を管理するエネルギーマネジメントシステムとが連携したシステムを想定した(図1)。エネルギーマネジメントシステムは神奈川工科大学のHEMS認証支援センターの環境を使って実装し、そこにシミュレータで代用した産業ロボットシステムを持ち込んで実証実験システムを構築した。

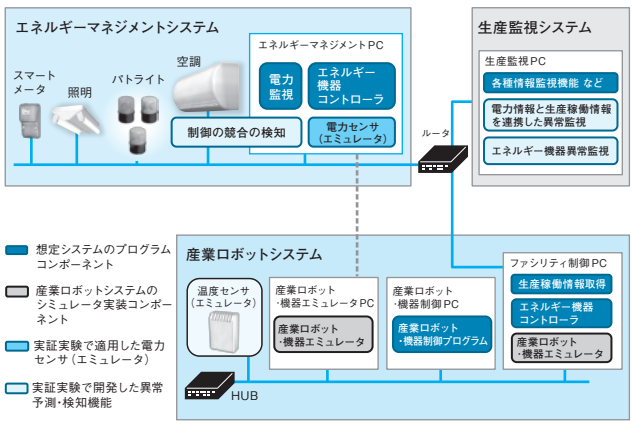


図1　実証実験環境での分野間連携システムの構築

3 実証実験の体制

本実証実験に参加した各団体を以下に示す※2。

表1　実証実験の体制

団体名	役 割
IPA	実証実験の仕様決定、実証実験用プログラムの開発、実証実験の実施、評価と報告書作成
ORiN協議会	産業ロボットのシミュレーションソフトの提供、実験内容の検討
エコーネットコンソーシアム	エネルギーマネジメント仕様(ECHONET Lite規格)の技術提供、実験内容の検討
神奈川工科大学	エネルギーマネジメント用IoT機器接続ソフト(SDK)の提供と開発技術提供、実験内容の検討

4 実証実験のテーマ選定

「実践に向けた手引き」の【機能要件3 異常の予兆を把握できる】及び【機能要件6 異常発生を監視・通知できる】に対応する以下の対策をテーマとして選定した。

4.1 異分野システムの監視情報の連携による異常検知の能力の向上【機能要件3並びに6】

中小企業の工場の生産環境では、老朽化した機器や設備が使用されていることがあり、それらの状態や状況の変化によって故障や漏電などの異常が発生し、深刻な問題につながることが懸念される。異常が発生する前に、できるだけ早く異常の兆候を予測して対策を取れること、また、異常が発生してしまったときに影響が大きくなる前にいち早く検知できることが重要となる。

エネルギーマネジメントシステムと産業ロボットシステムの2つのシステムの監視情報を連携することによりそれを可能とする仕組みの実現性を検証した。

4.2 異なるシステムによる制御の競合の検知【機能要件6】

工数節約のために、例えばエネルギーマネジメントシステムは既製のソフトを使って構築し、産業ロボットシステムは導入している機器の要件に対応して開発したソフトを使って構築したとき、それぞれのシステムの優先事項が異なる状況が発生し、一つの機器に対して相反する指示を出し続ける事態(制御の競合)となる可能性がある。制御の競合を検知してシステムの運用者が優先する対応を行えるようにする対策の実現性を検証した。

5 実証実験1：異分野システムの監視情報の連携による異常検知の能力の向上

5.1 対策技術

想定している産業ロボットシステムは、産業ロボット・機器が定型的な動作を繰り返して製造物を加工するものであり、この一定の加工サイクルでの消費電力は何らかの原因で機器やシステムが正常時と異なる状態にならない限りはほぼ一定である。そこで、1加工サイクル当たりの電力量を随時導出し、その値が正常時から乖離しているかどうかを監視して異常につながる兆候や異常を検知する方式を実装した。

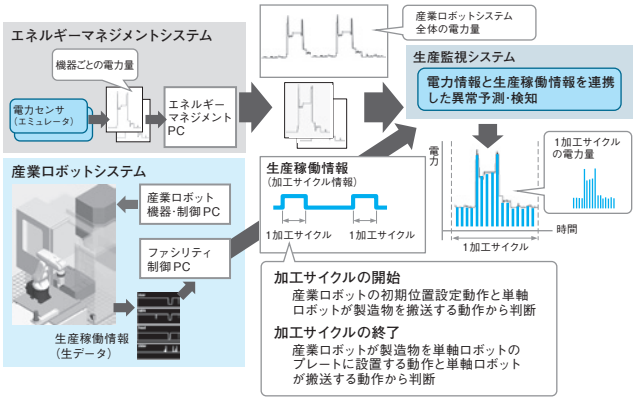


図2　異分野システムの監視情報の連携

5.2 実証実験結果

実証実験システムで産業ロボットシステムの電力量が正常時より大きい状態を作ったところ、本対策で実装した機能が、1加工サイクル当たりの電力量が正常時より大きいこと（ここではNC装置の電力量が大きくなっている）を検知し、パトライトにより通知することを確認した（図3）。

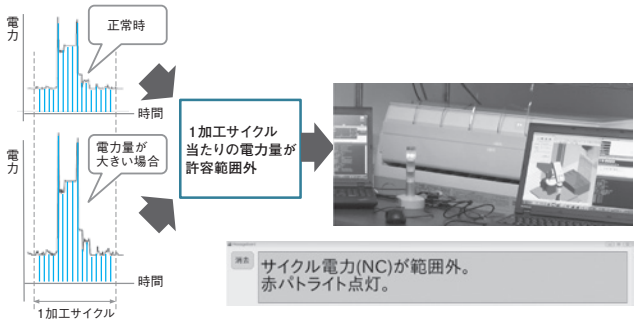


図3　1加工サイクル当たり電力量の監視

6 実証実験2：異なるシステムによる制御の競合の検知

6.1 対策技術

実証実験システムでは、エネルギーマネジメントシステムは期間内電力量が上限に近づくと電力量の大きい機器として空調に稼働停止を指示し、他方、産業ロボットシステムは温度を下げるために空調に稼働開始を指示する。この制御の競合を検知する機能を空調に実装する、という想定で疑似的にエネルギーマネジメントPCに実装した。

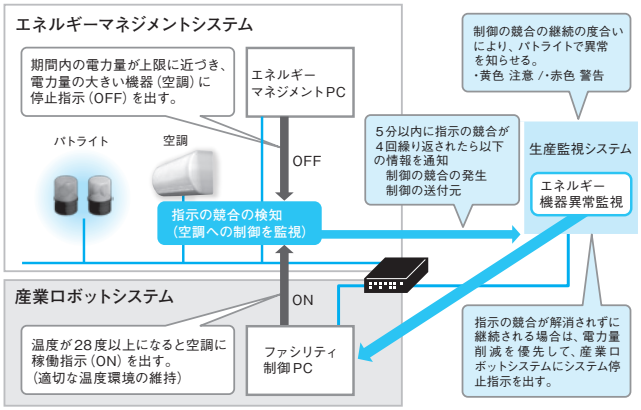


図4　異なるシステムによる制御の競合の検知

6.2 実証実験結果

実証実験システムで、温度が高く、かつ期間内電力量が上限値に近い状態を作ったところ、空調に対する制御の競合が発生し、本対策で実装した監視機能がそれを検知してパトライトにより通知することを確認した（図5）。

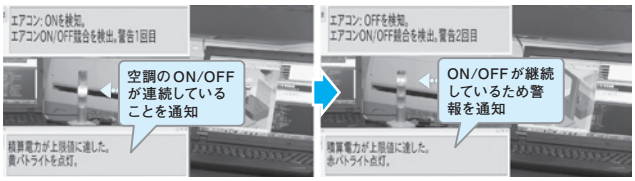


図5　空調への制御の競合の検知

7 まとめ

中小企業の工場を想定し、そこでの分野間連携システムで有効と思われる異常予測と検知の機能を、比較的小さな工数（約3カ月）かつ実現可能な方式で実装したところ、システムの本来機能に影響を与えることなく動作することを確認した。

IPA/SECは、引き続き、IoT機器の安全安心を確保する対策において有効な方式や考慮事項を検討し、「つながる世界の開発指針」及び「実践に向けた手引き」の普及展開につなげていく。

脚注

※1　http://www.ipa.go.jp/sec/reports/20160511_3.html

※2　産業ロボットシステムのシミュレータ導入で協力していただいた企業
シミュレータ（富士通製VPS）：デジタルプロセス株式会社
産業用ロボットエミュレータ：株式会社デンソーウェーブ
6軸ロボット3Dデータ：株式会社デンソーウェーブ
単軸ロボット3Dデータ：ヤマハ発動機株式会社
NC装置3Dデータ：一般財団法人 機械振興協会、ファナック株式会社

IoT時代の安全安心に向けて

「つながる世界の利用時品質」の策定

SEC研究員　西尾 桂子　SEC研究員　遠山 真　SEC調査役　宮原 真次

1 はじめに

IoT(Internet of Things)では、様々な「モノ(things)」がネットワークに接続されるが、IoT製品／サービスに不慣れだったり、習慣や文化の異なるユーザの増加が想定される。そこでは、利用状況を把握し、安全性と操作性を考慮した製品の開発が不可欠となっている。また、事業者や一般ユーザが開発者の想定と異なるつなぎ方を容易に実施できる設計にしていると、つなぐことで脆弱性が発生し(セキュリティリスク)、その結果安全性も損なわれる(セーフティリスク)可能性もある。これらユーザの特性や利用状況などで左右される、ユーザの観点から見た品質を、SQuaRE(ISO/IEC 25000シリーズ)では「利用時の品質」と呼んでいる。そこで、IPA/SECはつながる世界(IoT時代)ならではの利用時の品質に関連するリスクに着目し、ユーザの観点から見たリスク対策のための報告書を取りまとめることとした。

2 利用時の品質の必要性

本報告書で言う利用時の品質とは、性能や耐久性といった製品品質に対して、実際にユーザが利用する際の有効性や満足度、危険な状況を招かないリスク回避性などを指す。IoT製品／サービスの利用時の品質を向上することで、以下のような効果が期待される。

① ユーザの満足度が高まる
② 市場に出てからの失敗が減る
③ 新しい「ユーザ経験※1」を生み出す

つながる世界では、家電や自動車など10年以上利用される機器やシステムも多く、その間に利用環境が大きく変化する。また、IoTを構築する事業者や一般ユーザが様々な製品を組み合わせてインターネットにつなげるケースが想定される。このため、開発者の想定外の利用状況や利用環境で利用され、リスクが発生する可能性がある(図1)。安全安心(セーフティ・セキュリティ・リライアビリティが確保されている状態)なつながる世界のためには、企画・設計時から利用時の品質を考慮することが必要となっている。

利用時の品質については、従来はマーケティング担当者やデザイン担当者が中心に検討してきたが、IoT製品／サービスではその機能の多くがソフトウェアによって実現されている。そのためソフトウェア開発者も利用時の品質の向上の一端を担う必要がある。

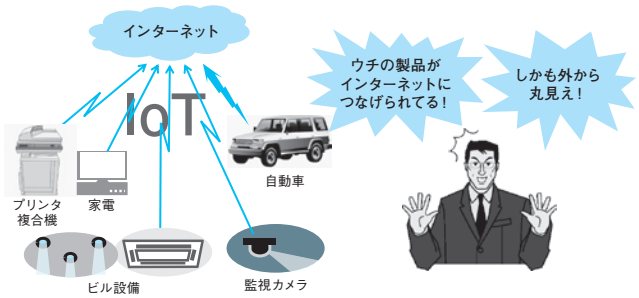


図1　つながる世界の安全安心対策の課題

3 利用時の品質向上に向けた開発プロセス

SQuaREでは、機器やシステムの品質として「製品品質」「利用時の品質」「データ品質」のモデルを定義している。設計した結果が製品として実現され、製品品質となってユーザの手にわたり利用時の品質となるが、利用時の品質は機器やシステムとのインタラクション(利用とその反応)の結果に関係する特性であり、製品品質と比較してユーザの主観的な評価が必要となるものである(図2)。

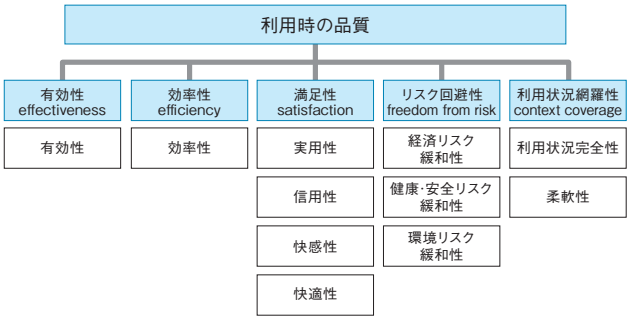


図2　SQuaREおける利用時の品質モデル

利用時の品質については、製品が開発者の手を離れている段階で評価されるため、実ユーザや実利用環境における調査によって明らかにする必要がある。実ユーザの特性や実利用環境は極めて多様であることから、従来の定量的な品質管理とは異なり、インタビューやユーザの利用状況の観察などに基づいて掘り下げていく定性的な管理が必要である。また、ここで得られた利用時の品質に関する情報は、企画・設計段階にフィードバック

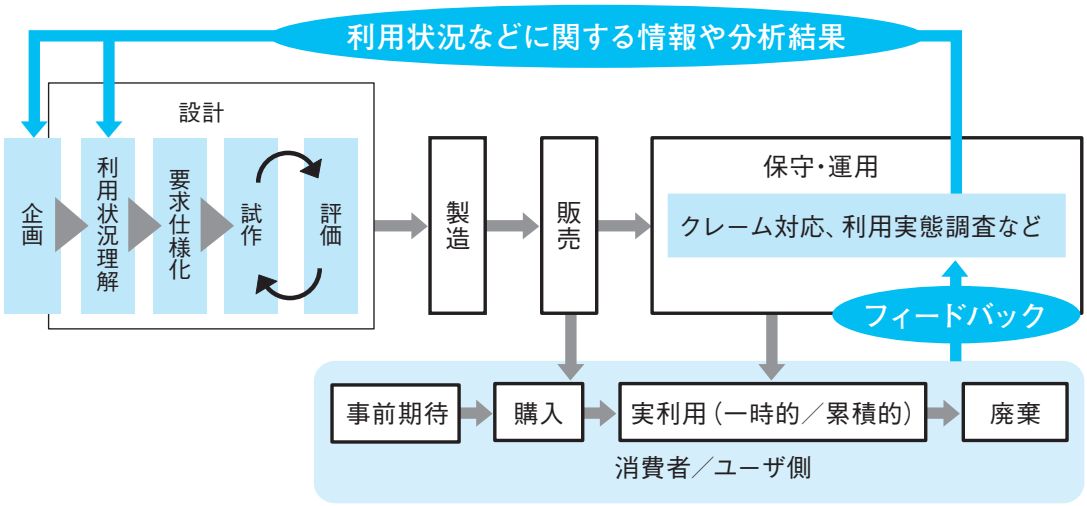


図３　利用時の品質向上に向けたソフトウェア開発プロセス（イメージ）

することが何より重要であり（図３）、これを怠るとユーザの実情に合わせた改善が行われないことになる。

4 「つながる世界の利用時品質」の策定プロセス

本報告書の策定においては、UI・ユーザビリティ研究者、UXデザインの専門家、及び自動車、情報家電、電気機器、クラウドサービスなど多様な産業分野において製品のユーザビリティ向上を図る部門の識者から成る「利用時品質検討WG」を立ち上げ、WGメンバーのコンセンサスを取りながら検討を進めた。また、過去に発行した「つながる世界のソフトウェア品質ガイド」「つながる世界のセーフティ＆セキュリティ設計入門」「つながる世界の開発指針」などにおいて得られたセキュリティとセーフティの関係の整理などの知見も活用した。

その上で、以下のプロセスにて策定した。

（１）利用時の品質に関係する失敗事例・成功事例の把握

WGメンバーから実際に経験した利用時の品質に関する失敗事例・成功事例・開発現場で懸念している事項、合計26件の事例を持ち寄った。

（２）課題を抽出・分類

持ち寄った26件の事例を分析し、原因となる課題を一般的な項目に分類した。

（３）ソフトウェア開発者として留意すべき視点を整理

分類した課題から、解決の方向性としてソフトウェア開発者が留意すべき15の視点に整理し、そのポイントと対策をまとめた（表１）。

5 つながる世界の利用時品質

5.1 概要

各視点の解説は以下の項目から成る。

- 視点：利用時の品質向上のための留意点
- ポイント：各視点において、検討すべき項目。各視点に１から３項目あり、具体的な解説も添付
- つながる世界での注意点：つながる世界の利用時の品質を考える際の注意点

各視点については、以下のように利用いただくことを期待している。

- 視点を自社の状況と照らし合わせ、関係する場合にはそのポイントを検討いただく
- とくに、「現状」と「今後（つながる世界）」の両面から、対応状況を確認いただく

5.2 視点の例

各視点は、視点／ポイント／解説／つながる世界での注意点により構成されている。イメージを示すイラストを加え、意図を伝える工夫をした。以下に特徴的な２つの視点について、意図を説明する。

［視点６］ユーザ経験を収集・分析・評価する

利用時の品質を評価するために、製品の利用環境やユーザ経験の収集、分析を行う。ユーザ経験には、利用中だけでなく利用後の感想や利用前の期待なども含む。つながる世界では、つながりによって利用環境が拡大し続けるため、ユーザ経験は変化する。このようなユーザ経験の変化を把握するには、ネットワークを介して利用状況を把握することも考えられる。ただしプラ

表1　視点一覧

区　分	視　点	
組織文化	視点1	つながる世界の利用時の品質を意識する
	視点2	他部門と連携して取り組む文化を作る
	視点3	自社や顧客の責任者の意識を変える
	視点4	利用時の品質向上にかかわる人材を育成する
把握・分析	視点5	ユーザの特性や経験、文化、利用環境を考慮する
	視点6	ユーザ経験を収集・分析・評価する
	視点7	間接・受動的ユーザやプライバシーにも配慮する
	視点8	利用状況や利用環境の変化の影響を考慮する
設　計	視点9	企画・設計段階からユーザを巻き込む
	視点10	ユーザを安全な操作に導く設計をする
	視点11	第三者に機能や情報を使わせない設計をする
	視点12	操作結果やメッセージを確実に伝える設計をする
保守・運用	視点13	ユーザや関係者からフィードバックを得る仕組みを作る
	視点14	知見を開発時及び出荷後の利用時の品質向上に活用する
	視点15	つながるリスクの周知と安全設定の仕組みを作る

イバシーにかかわる内容も多いため、ユーザに必要性を理解いただくことが必要である。

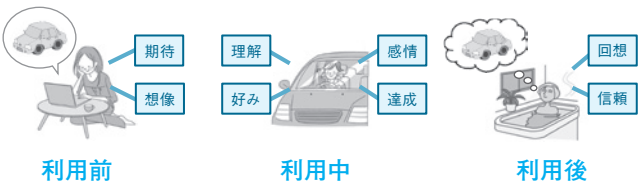


図4　ユーザ経験の例

[視点9]企画・設計段階からユーザを巻き込む

近年、短いサイクルでソフトウェア開発を繰り返す手法や、最小構成で製品やサービスをリリースするビジネスモデルを導入する企業が増えている。これにより、早期にユーザに利用してもらい、ユーザ経験に基づいた改良を行うことが可能となる。

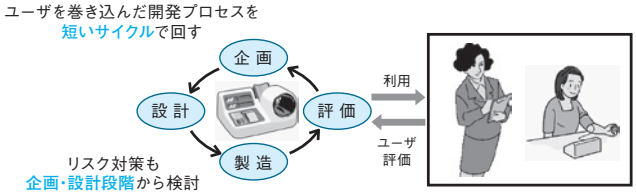


図5　ユーザを巻き込んだ開発プロセス

6　おわりに

本報告書は、ソフトウェア開発者とマーケティング担当者やデザイン担当者などが課題認識を共有し、協力して製品開発に取り組むために活用されることを想定している。

IPA/SECはIoTならではのリスクに着目し、開発者向けにリスク対策のための「つながる世界の開発指針」を取りまとめ2016年3月に公開した。開発指針では開発者の観点でリスク対策をまとめたのに対し、本報告書ではユーザ観点でリスク対策をまとめている。本報告書の観点を、開発指針の改訂版に取り込む予定である。

「つながる世界の利用時品質」は以下のWebサイトで公開しているので、積極的に活用いただきたい。

http://www.ipa.go.jp/sec/reports/20170330.html

脚注

※1　ユーザ経験：UX(User Experience)とも表現される。製品やサービスの利用によって生じる人の認識と反応であり、利用前、利用中及び利用後に発生するあらゆる感情、信頼、好み、認識、物理的及び精神的な反応などを含む。

IoT時代の安全安心に向けて

システムズエンジニアリングの推進

SEC研究員 齊藤 善治　SEC調査役 西原 栄太郎　SEC調査役 室 修治

1 はじめに

IoT(Internet of Things)時代には、様々な機器やシステムがつながることにより新しい価値が生み出される。一方で、それらを構成する要素間の相互関係が複雑なものになり、従来の開発方法ではその価値の実現と品質の担保が難しくなっている。

このような環境変化に対応するために、開発のパラダイムシフトとして注目を集めているのが「システムズエンジニアリング」である。欧米では航空・宇宙などの分野の取り組みを基に体系化され、一般的な製品やシステムへの適用も進み、有効性が認知されている。しかし、我が国では一部の大学や企業での取り扱いにとどまり、広く浸透していないのが実態である。

システムズエンジニアリングの推進は、日本の産業界が将来にわたって成功していくために有用な情報を発信し、産業界のフィードバックも得ながら共に考えて、共有知として蓄積していくことを念頭に実施する。

2 システムズエンジニアリングの有用性の発信

システムズエンジニアリングの普及には、まずその有用性が広く認知されることが重要である。そこで、IoT時代のビジネスチャンスから説き起こし、経営層と開発現場の両者の課題意識を喚起し、新しい開発のアプローチの必要性を説明することとした。啓発用資料として「経営者のためシステムズエンジニアリング導入の薦め」(2017年3月)、「開発者のためのシステムズエンジニアリング導入の薦め」(2017年5月)を作成し公開した※1。

2.1 「経営者のためのシステムズエンジニアリング導入の薦め」

システムズエンジニアリング適用による経営的利点及びそのために経営者が考慮すべき事柄を記載している。

主な内容を以下に示す。

- IoT時代のリスクと新たなアプローチの必要性
- システムズエンジニアリングの4つのポイント

とくに4つのポイントを事例と共に紹介している。

①目的指向と全体俯瞰
②多様な専門分野を総合
③抽象化・モデル化
④反復による発見と進化

2.2 「開発者のためのシステムズエンジニアリング導入の薦め」

開発者がシステムズエンジニアリングを適用するための基本的な情報などを、とくにシステムズエンジニアリングが効力を発揮する場面(システム設計、評価・解析など)での考慮事項や活動を中心に事例と共に解説している。

3 今後の推進計画

今年度は開発者、開発/技術管理者が実際に現場での適用を進める上で参考となる基礎解説書を作成していく。

脚注

※1 http://www.ipa.go.jp/sec/reports/20170329.html

「システムズエンジニアリング」に関する国際的な協議会である INCOSE(International Council of Systems Engineering) IW(International Workshop) (2017年1月27日から米国ロスアンゼルスで開催)への参加

過去のINCOSE IWは宇宙・航空、軍用の分野を中心に200名程度の参加であったが、近年は、交通・ヘルスケアなどの分野も加わり、今回は520名前後とのことであり多分野での関心が高まっていることが感じられた。主催者側も初回参加者への個別のガイドやガイダンス、及びWebなどでの資料の充実などを図り、会員拡大・広報に力を入れている。

基調講演はテルモの医療機器への応用であり、ここでもシステムズエンジニアリングの一般産業への適用が進んでいることが実感できた。ただし日本からの参加者は10名程度であり、国内でのシステムズエンジニアリングの浸透や関心の低さが表れているようである。システムズエンジニアリングの推進の必要性を再認識すると共に、今後も海外の動向には注意を払っていきたい。

IoT時代の安全安心に向けて

大規模・複雑システムの障害原因診断手法

SEC調査役　三原 幸博　SEC調査役　十山 圭介　SEC調査役　石井 正悟
SEC研究員　松田 充弘

システムの障害が社会に大きな影響を与える可能性が広がっており、コンピューターによる複雑な制御機能を持つ大規模システムの障害原因を迅速に追究し、再発防止のための提言を行う体系が必須となっている。この課題に対処するべくIPA/SECでは、2014年度より「大規模・複雑化した組込みシステムのための障害診断手法」の確立を目指して活動している。

1 障害原因診断フレームワークの確立

本活動では、システムのソフトウェアと付随するネットワークコミュニケーションの障害に注目し、それらの原因追究と解決への提言を行う方法論として、図1に示す構成の「事後Verification & Validation（V&V）」という考え方と、その要素技術を提案してきた。V&Vは要求仕様や設計の不備をレビューする視点で作られた体系であるが、設計時点での想定不足が運用後の障害を引き起こすと考えられることから、障害の原因診断にも使えるとして「事後V&V」と名付けている。設計・実装のミスだけでなく、要求段階の仕様ミスや市場での運用ミス、環境変化への対応ミス、セキュリティアタックによる侵害など多様な原因があるので、このような体系的な考え方が必要となる。

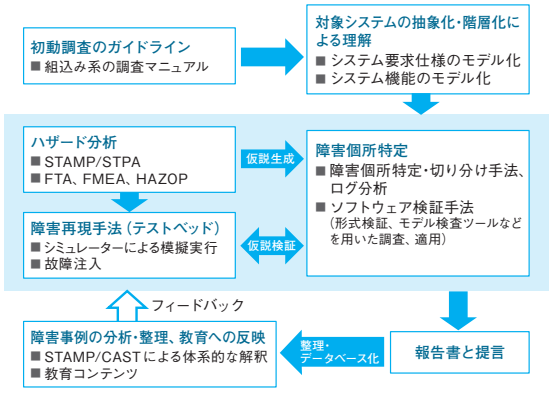


図1　事後V&Vの全体像

2 要素技術の展開

2014年度は、事後V&Vの全体像と既存技術として、その中で用いられる要素を紹介し、2015年度に、ハザード分析手法と原因個所の絞り込みについて具体化し、2016年度は比較項目や事例の追加で検討を深めた。

対象システムのモデル化ではSysML※1などのツールとの連携を含めてSTAMP/STPA※2手法を詳細検討し、2016年度はHAZOP※3との比較も行った。また、障害原因の仮説の絞り込みとしては、予兆監視にビッグデータを応用する事例、モデル検査を適用して要求仕様の不備を導く事例を報告した。

また、これら要素技術の検証・確認用のサンプルシステムとして、2014年度には化学プラントシミュレーター、2015・2016年度では倒立二輪車の制御システムを開発し、一般に利用できるよう公開した。

3 まとめと今後の取り組み

近年の大規模・複雑システムの障害は、開発時の既存V&Vをすり抜けて起こるものや運用後の環境変化への適応不足によって起こるものなどがある。これらの原因を究明する魔法の杖があるわけではないが、V&Vの考え方に基づいて、障害が起こった後に第三者の立場でシステム設計を再検証する方向性は妥当であろうと考える。

化学プラントシミュレーターや二輪倒立ロボットの模擬事故を事例としたが、人間と機械の協調制御で使われる製品やシステムが今後ますます増える状況にあり、このようなシステムのハザード分析の検討（図2）を通じて従来手法の限界と本活動による新しい手法の可能性が得られた。

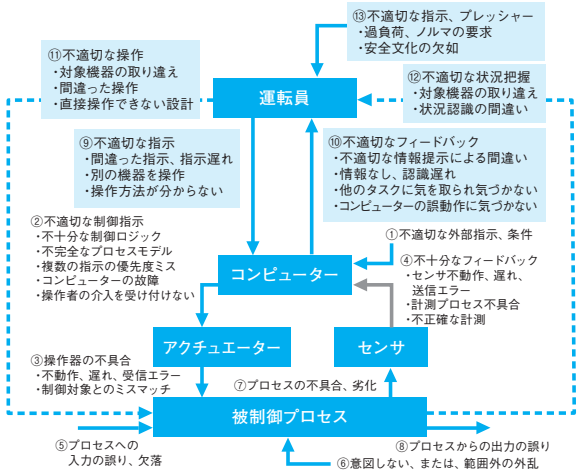


図2　人間系を含むハザード分析のガイダンス

今後、オープンエンドなシステムの安全性担保のため、解析手法への要請がますます高まる。STAMP/STPAの解析能力拡大のため、ヒューマンファクタや組織要因を取り込み、またレジリエンスエンジニアリングを活用することで、本活動をシステム安全性向上技術へと展開していく。

脚注

※1　Systems Modeling Language
※2　STAMP（Systems-Theoretic Accident Model and Processes）：システム理論に基づくアクシデントモデル
STPA（System -Theoretic Process Analysis）：STAMPに基づくハザード分析手法
※3　Hazard and Operability study

IoT時代の安全安心に向けて

システム理論に基づく新しい安全性解析手法STAMP/STPA

SEC調査役 三原 幸博　SEC調査役 石井 正悟　SEC調査役 十山 圭介
SEC研究員 松田 充弘　SEC調査役 三縄 俊信　SECシステムグループ主任 八嶋 俊介
SEC研究員 金子 朋子

近年の我々の生活に満ちあふれている車や列車、航空機、ロボット、家電製品などの工学システムは、その内部にコンピューターと無線ネットワーク機能を持ち、高度なソフトウェアによって制御されているが、これらはますます大規模化・複雑化しつつある。IoT、AI※1の時代と言われるゆえんである。しかし既存の安全性解析手法や安全規格は、このような複雑システム、とくにソフトウェアの安全性評価に対応できていない。そこで、マサチューセッツ工科大学(MIT)のNancy Leveson教授は2012年に、新しい安全性解析手法としてSTAMP/STPAを提唱した。欧米では産業界へのSTAMP普及が進みつつあるが、日本では認知されているとは言えないのが現実である。そこで、IPA/SECではIPA/SECでは日本でのSTAMP普及を目指して活動している。

1 STAMP/STPAとは

従来のシステムの多くは、ハードウェア機器がシステムの基幹を担う要素となっていた。それ故、アクシデントは、機器の故障や人間のオペレーションミスが根本原因であり、それがほかの機器や人間に伝搬し、最終的にアクシデントに至るものと捉えられていた。その延長で、従来の安全分析では、システム構成機器を組み合わせたものをシステム全体と捉えて分析していた。

しかし、近年は、システムの基幹を担う要素がソフトウェア中心に変化すると共に、システムにかかわる要素(人、ソフトウェア、ハードウェア)も爆発的に増大してきた。そのため、要素間の相互作用も複雑になり、個々の要素の役割を理解しただけでは、もはやシステムを理解できなくなった。そして複雑なシステムにおけるアクシデントの原因は、一つの構成要素に限定できる要因だけではなく、複数の要素間の相互作用による要因も考えなければならなくなった。

このような状況において、Leveson教授は、現代のシステムのアクシデントの多くは、システム構成要素の故障によって起きるのではなく、システムの中で安全のための制御を行う要素と制御される要素間の相互作用が働かないことによって起きるという「STAMP：システム理論に基づくアクシデントモデル」を提唱した。

STAMPはアクシデントを説明するモデルである。STPAはSTAMPをベースとしたハザード分析手法である。従来のハザード分析手法と比べて**STPAは、複雑なシステムの「ソフトウェアの要求・設計ミス」を識別するのに適したもの**とされている。

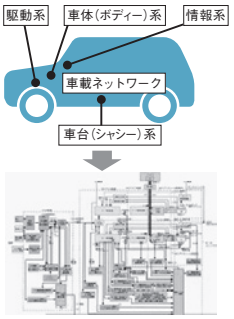


図1　システム中心の安全分析

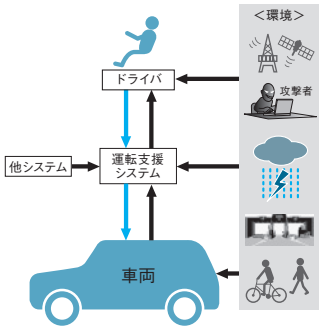


図2　システム理論に基づく安全分析

2 はじめてのSTAMP/STPA

IPA/SECは、2016年4月にSTAMP初心者向けにSTPA手順を解説する「はじめてのSTAMP/STPA」を発行し※2、多くの産業界の方々に参考にされている。

本書では、国内で実際に運用されている鉄道踏切制御装置にSTAMP/STPAを実施し、勘違いしやすい点や理解しにくい点についての注意や導入の勘どころをSTAMP/STPAの手順に沿って具体的に解説しており、初心者向け入門解説書として有用な一冊である。

図3　はじめてのSTAMP/STPA

3 はじめてのSTAMP/STPA(実践編)

「はじめてのSTAMP/STPA」では基本的な考え方と教科書に近い応用事例を中心に解説したが、2017年3月に公開した「はじめてのSTAMP/STPA(実践編)」※3では、産業界でのニーズを考慮した多様な事例についての安全分析を試みた。ここで取り上げた事例は、いずれも教科書で例示されているような標準的な制御構造とは異なっており、それぞれに分析の工夫をしている。

図4　はじめてのSTAMP/STPA(実践編)

3.1. フィードバックがない「とりこ検知」事例

Control loopに着目すべしと言われても、Feedbackがないからloopがないんですけれど・・

この事例の安全制御構造にはフィードバックがほとんど存在しない。フィードバックを持たないことは、一般的に安全の阻害要因であるが、その逆にもなり得る。この分析結果は、その両面を考えるきっかけになるであろう。この事例では、フィードバックがないことがむしろ安全上の強みになっている。

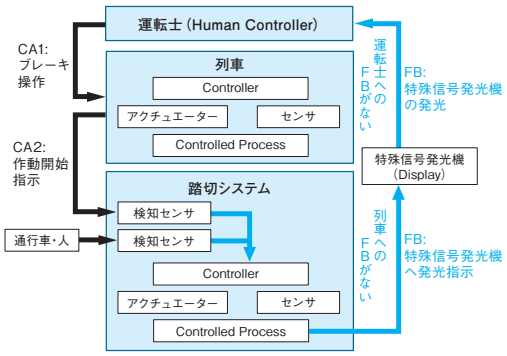


図5　フィードバックのない事例

3.2. 組織や人が絡む業務ワークフロー事例

心配なのはコンピューターやメカじゃないんですけれど・・

安全分析したい対象システムが必ずしもコンピューターシステムとは限らない。この事例では、踏切工事にかかわる人・組織、そしてその業務を分析対象とした。この事例では、コントロールストラクチャー図を書くことによって、制御行動の抽出不足に気づきやすくなった。

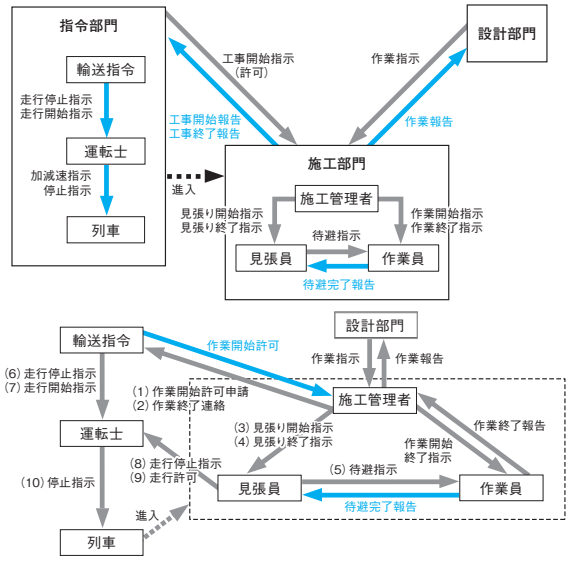


図6　視点を変えた2つのコントロールストラクチャー

3.3. エンタープライズ系の「ネット通販」事例

STAMPを適用したいのは制御系じゃないんですけれど・・

この事例の分析では、識別された2つのコントロールアクションが、別々のアクションのままでは安全ではなく、1つの不可分なアクションにまとめられる必要があるという結論に至った。この分析の結果、「損失防止」のための新たな制御行動の欠陥を見つけることができた。

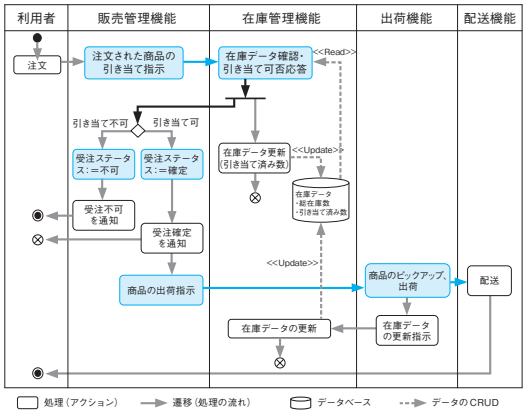


図7　ネット通販の事例

3.4.「機械 対 機械」以外の要素に対するヒントワード

「人」へのヒントが「生成の欠陥」？　「組織」へのヒントが「コンポーネント故障」？　そう言われても・・

現状STPAが提供する、HCF※4特定のためのガイドワードは、制御機械と稼働機械の組み合わせのみであるが、今後、人・組織・機械が協調動作するシステムの安全性解析がますます重要になると考えられることから、それらの組み合わせに適合したガイドワードをヒントワードと呼称することとした。

表1　HCF特定のためのヒントワードの種類

制御＼被制御	人	機械	組織
人	人 対 人	人 対 機械	人 対 組織
機械	機械 対 人	機械 対 機械	
組織	組織 対 人		組織 対 組織

4　STAMPワークショップ in Japan

2016年12月には、第1回STAMPワークショップin Japanを開催し多様な産業界からの参加を得て、多くの参加者が、新しい時代の安全性をどうやって確保していくか悩んでいる現状を目の当たりにした。2017年11月には、第2回を東京地区にて開催の予定である※5。

欧米では日本よりも早くからSTAMPワークショップが定期開催されている。IPA/SECは、欧米と連携してSTAMPワークショップを開催していく所存である。

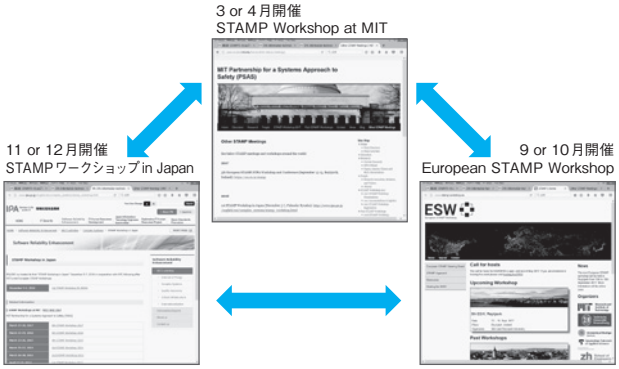


図8　日米欧が連携するSTAMPワークショップ

5　まとめと今後の取り組み

STAMPに限った話ではなく、実際の開発現場では「手法適用が教科書通りにいかない」ことはよくあることで、また「立て板に水」のごとく定石通りに分析を進めるだけでは安全化を達成できない場合もあるため、分析の工夫が必要であることを「はじめてのSTAMP/STPA(実践編)」に示した。

今後IPA/SECは、更なる工夫によってSTAMP理論の進化に貢献し、STAMP/STPAの解析能力拡大・活用容易化に向けた活動を進めていく。

脚注

※1　IoT(Internet of Things)：物のインターネット
　　AI(Artificial Intelligence)：人工知能
※2　http://www.ipa.go.jp/sec/reports/20160428.html
※3　http://www.ipa.go.jp/sec/reports/20170324.html
※4　HCF(Hazard Causal Factor)：ハザード誘発要因
※5　2nd JSW, Tokyo(第2回STAMPワークショップ)
　　http://www.ipa.go.jp/sec/events/20171127.html

IoT時代の安全安心に向けて

制御システム セーフティ・セキュリティ検討活動

SEC調査役 石田 茂　SEC研究員 細目 紀子　SEC専門委員 中谷 博司

1 はじめに

プラント、鉄道、電力など社会の重要インフラを担う制御システムは、障害発生時に人命や環境に与える影響の大きさから、ライフサイクル全般にわたる安全性(セーフティ)を重視した"ものづくり"が行われてきた。

一方、2020年に予定されている東京オリンピック／パラリンピック開催に向け日本全体としてのサイバーセキュリティ対策が急がれている今日、重要インフラを手がけるものづくり企業では以下のような課題がある。

- 重要インフラのサイバーセキュリティ対応の重要性は理解しているが、セーフティ、セキュリティ双方に精通した技術者は極めて限られる
- セキュリティ要件を実現した場合、セーフティに及ぼす影響を評価する上での手がかりが欲しい
- セーフティに比べてセキュリティの歴史は新しく、対応が後手になりがち
- セーフティシステムに対するセキュリティ脅威分析の具体的な進め方が分からない

上記の課題は、いずれも容易には解決できない内容だが、IPA/SECでは変化する時代の要請に対応した活動が必要であると考え、課題を共有し、セーフティとセキュリティが連携し双方の要件をすり合わせる枠組みを提示することを目的とした「制御システムセーフティ・セキュリティ検討WG」を設立した(なお、WG名称は昨年度「組込みシステムセーフティ・セキュリティ検討WG」としていたが、検討対象システム及び活動内容を端的に示すため、本年度からは「制御システムセーフティ・セキュリティ検討WG」としている)。

2 活動経緯

2017年度からの本格検討に先立ち、2016年度は、その準備段階として、IEC 61508※1若しくはISO 26262などの国際機能安全または、セキュリティ認証取得(例えばEDSA認証※2、Achilles認証※3)の実務経験を有する企業の方々、学術機関の専門家の方々11名に委員として参加いただき、本検討を円滑に進めるための叩き台となる資料を作成し、2017年度以降の進め方についてディスカッションを行ってきた。

3 活動方針の策定

3.1 活動方針

以下のような方針で進めることとした。

(1) セーフティファースト
セーフティとセキュリティの要件検討は、セーフティゴール(安全性、可用性などの確保)を前提としてセキュリティを考える

(2) グローバルスタンダード(国際規格)準拠
プラント、鉄道、自動車など、重要インフラ、製品における各国及び、国際的な動向を踏まえ、ISO/IEC国際規格及び業界スタンダード(IEC 61508、IEC 62443※4)に基づき、上流プロセスにおける要件のすり合わせ検討を行う

(3) 国際標準化活動との連携
IEC/TC65/WG20※5など国際的な検討活動との連携を意識した活動を推進する

3.2 目標成果物

以下のような内容が反映された「制御システム セーフティ・セキュリティ要件検討のためのガイド(仮称)」を作成することを2017年度の目標に設定した。

- IEC 61508をベースに、セーフティプロセスにセキュリティ要件、運用上の留意事項などを関連付ける
- 上記プロセスを用いて検討を行う際の手順例
- 参考となるような検討フォーマット、サンプル例

4 WG活動の概要

2017年度に進めるWG活動内容について検討した。ここではその中から幾つかのポイントを示す。

4.1 検討対象システム

検討の具体性を確保するために架空のFAシステムを想定した。これは架空の事業者製造工場内に構築された、組み立て生産を行うための架空の設備である。本WGでもよりどころとしているセキュリティ国際規格IEC 62443との関係を含め図1にその全体像について示す。

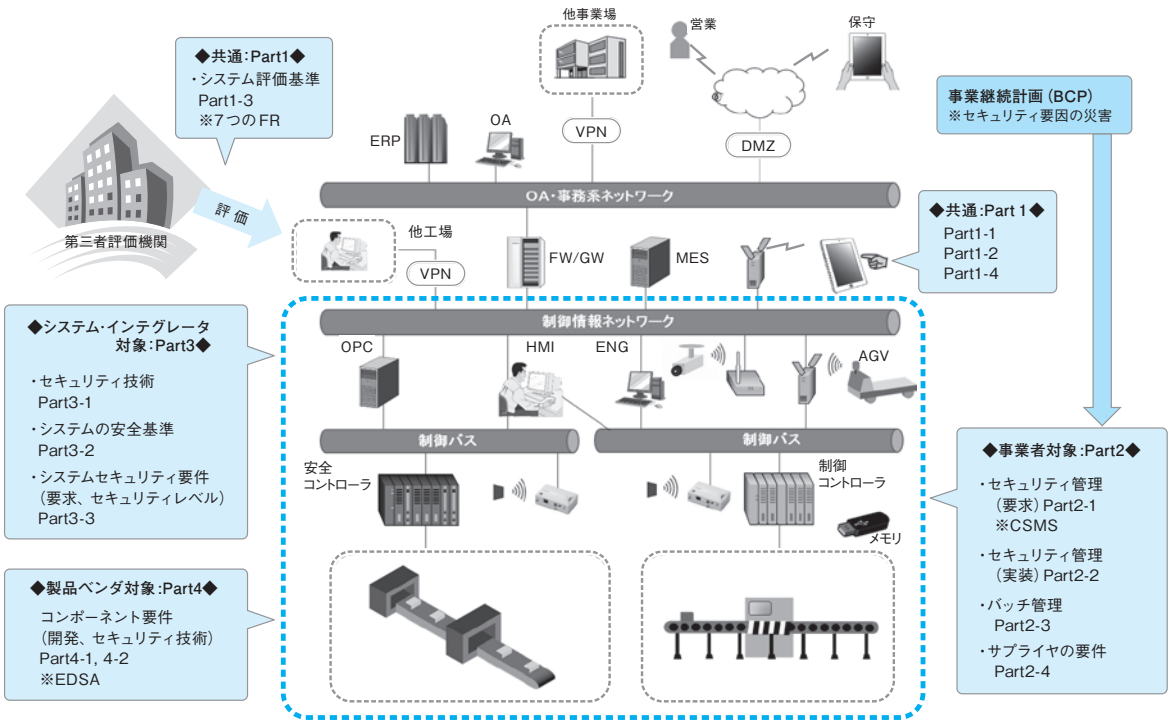


図1　制御システムとIEC 62443の関係

4.2　セーフティな制御コントローラ

前掲の製造工程で使用される制御システムコントローラは、当該製造工程の安全性要求に鑑みてIEC 61508に適合済みと想定している。具体的には、図2に示される安全ライフサイクルに基づく設計、製造、運用が適切になされているとの前提に立っている。

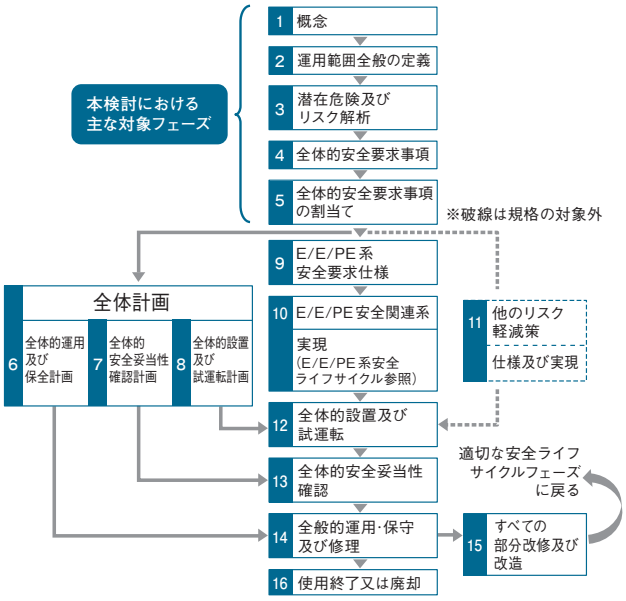


図2　IEC 61508 安全ライフサイクル

4.3　IEC 62443によるセキュリティ要件検討

セキュリティ脅威が識別され、セキュリティ対策が検討・立案された場合に、既存のセーフティ機能、運用に及ぼす影響の抽出と対策の検討が重要となる。

そのため、プラントなどの産業制御用コントロールシステムのためのセキュリティ国際規格で、米国で発行されているNIST SP800-82などでも参照されているIEC 62443に基づくサイバーセキュリティ分析を試行し、セーフティ要件への関連性、影響を検討する。

4.4　国際認証取得経験者の知見を活かす

WGではIEC 61508、EDSA認証などの国際機能安全、セキュリティ対応の実務を経験した方々の知見を得ると共に、開発現場目線を意識したより実践的な活動とすることを目指す。

5　今後の取り組み

制御システムを実際に開発・運用している現場の方々の課題を共有しつつ、同様の課題を抱える多くの企業の問題解決の糸口とできるよう検討を進め、2017年度末に成果物リリースを行う予定である。

脚注

※1 IEC 61508：IEC(国際電気標準会議)が制定した基本安全規格。プロセス産業における電気・電子・プログラマブル電子(E/E/PE)機能安全に関する国際規格。

※2 EDSA認証：Embedded Device Security Assurance。制御機器(組込み機器)のセキュリティ保証に関する認証。

※3 Achilles認証：Achilles Certification。ネットワーク接続装置(コントローラ等)の認証。

※4 IEC 62443：制御システムセキュリティの事業者、インテグレータ、装置ベンダを対象とした汎用的な国際標準規格。

※5 IEC/TC 65 Industrial-process measurement, control and automation WG20 Framework to bridge the requirements for safety and security

重要インフラ等システム障害対策

SEC調査役 三縄 俊信　SEC研究員 目黒 達生　SEC研究員 村岡 恭昭
SECシステムグループ主任 八嶋 俊介　SEC調査役 三原 幸博　SEC研究員 松田 充弘
SECシステムグループリーダー 山下 博之

2015年度に引き続き、重要インフラ分野などのシステム障害事例からヒアリングなどにより障害事例情報を収集し、その分析と対策の検討を行った。ITサービスシステムは産業分野横断で活用可能な普遍的な教訓を6件導出し、2015年度までの教訓と併せて分類整理した上で教訓集として公開した。また、2010年から収集している、報道されたシステム障害事例について横断的に傾向分析を行い、頻度の高い「ヒューマンエラー」と「システムの高負荷／過負荷」に起因する問題を取り上げ、詳細解説を教訓集に追加した。更に、ITサービスシステムの障害事例情報を共有する仕組みの構築に向けた支援活動を行い、新たに3つの産業分野で情報共有の仕組みを構築し運用を開始した。また、組込みシステムの教訓をもとに障害未然防止のための設計知識を整理する手法をガイドに取りまとめ、公開した。

ITサービスシステム

1 背景

情報処理システムは、銀行や証券などの金融サービス、各種手続きのための行政サービス、ソーシャルネットワークなどの情報通信サービス、交通機関の運行制御など、私たちの生活や社会・経済基盤を支える重要インフラ分野などのITサービスに深く浸透し、ひとたび障害が発生するとその影響は非常に大きい。私たちが安全で安心な生活や社会・経済活動を続けるためには、重要インフラなどを支えるITサービスの一層の信頼性向上が求められている。

社会に多大な影響を与え、報道されたITサービス障害の発生件数は図1に示すように、2009年から2016年にかけて増加傾向にあり、2016年は、鉄道・航空分野の旅客サービス障害、マイナンバー制度や電力自由化対応などの制度開始に伴う初期障害が発生した。

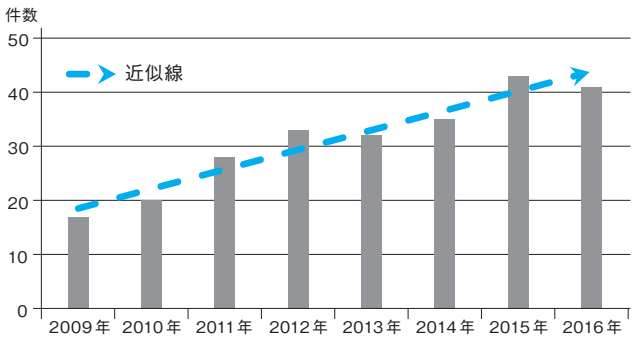


図1　報道されたITサービス障害の発生件数の推移

従来、情報処理システムの障害に対する原因分析と再発防止策の実施は、多くの場合、当事者においてのみ行われ、その情報は公開されてこなかった。そのため、当事者以外のシステムにおいて、あるいは他業界・分野のシステムにおいて、類似の障害が発生することがあった。

情報処理システムの構築・運用やその管理は、社会や技術の進展につれて複雑化・多様化しており、一人や一企業のカバーできる範囲には限界がある。そして、その複雑性・多様性は今後ますます拡大することは明らかである。従って、情報処理システムの構築・運用及びその管理にかかわる信頼性面での課題を解決するために、より多くの人たち・企業の経験を社会全体で共有・伝承することが求められている。

そこで、システムの障害事例情報の分析や対策手法の整理・体系化を通して得られる「教訓」を業界・分野を超えて幅広く共有し、類似障害の再発防止や影響範囲縮小につなげる仕組みの構築に向けた活動を2013年度から実施している。

2 障害事例の収集と教訓化

2016年度も継続して重要インフラITサービス高信頼化部会※1の活動を通じ、障害事例を収集し、障害原因の分析を行い普遍化した上で6件の教訓を導出し（表1、表2）、2015年度に取りまとめた教訓36件に追加して、計42件の教訓を収録した「情報処理システム高信頼化教訓集（ITサービス編）2016年度版」（以下、教訓集2016年度版）を公開※2した。

表1　2016年度に導出した教訓の分野別件数

産業分野	教訓数
交通分野	2件
金融分野	1件
行政・自治体分野	2件
その他	1件
計	6件

表2　2016年度追加教訓（ITサービス編）

教訓ID	教訓概要
ガバナンス／マネジメント領域	
G15	保守作業は「予期せぬ事態の発生」を想定し、サービス継続を最優先として保守作業前への戻しを常に考慮すること
G16	本番環境へのリリースは、保守担当が無断でできないような仕組みを作るべし！
技術領域	
T23	障害監視は、複数の観点から実装し、障害の見逃しを防げ！
T24	サービス縮退時の対策を考慮せよ
T25	障害原因が不明でも再発予防と発生時対策はできる
T26	既存システムの流用開発はその前提条件を十分把握し、そのまま利用可能な部分と変更する部分を調査して実施する

教訓集2016年度版では、2010年から2016年まで蓄積している、報道されたシステム障害情報を分析し、問題の傾向と対策を記載した新たな章を追加した。

- ヒューマンエラーの問題と対策
- システムの高負荷／過負荷に関する問題と対策

3　システム障害情報共有の仕組み構築

各業界団体などにシステム障害情報の共有の仕組み構築を働きかけ、2016年度に新たに3つの情報共有グループを構築し、その運営を開始した。

表3　情報共有体制を構築した産業分野

産業分野	企業・業界団体など	組織概要
クレジット分野	（一社）日本クレジット協会	システム研究会に参加する会員企業
地方団体	北海道IT情報共有グループ	北海道地域のインフラ事業者など
地方団体	（一財）関西情報センター（KIIS）※3	サイバーセキュリティ研究会に参加する会員事業者

また、2015年度までに運用を開始した6つの情報共有グループについて、IPAによる支援活動・意見交換を継続して実施した（図2参照）。

4　普及展開活動

①教訓集などのダウンロード
＜2016年度に実施されたダウンロード件数＞

成果物	件数
教訓集2015年版	1,575件
教訓作成／活用ガイドブック	1,713件
個別教訓リンク集	12,806件

②教訓集活用メールマガジンの発行
教訓集などをダウンロードする際に、IPAからの情報提供を希望された方を対象として、「教訓集活用メールマガジン」の発信を9月から開始（2017年3月現在、読者数約1,000名）。

③教訓集ダイジェスト2016年度版作成
普及展開活動を推進するために、教訓集2016年度版を要約した小冊子を作成した。

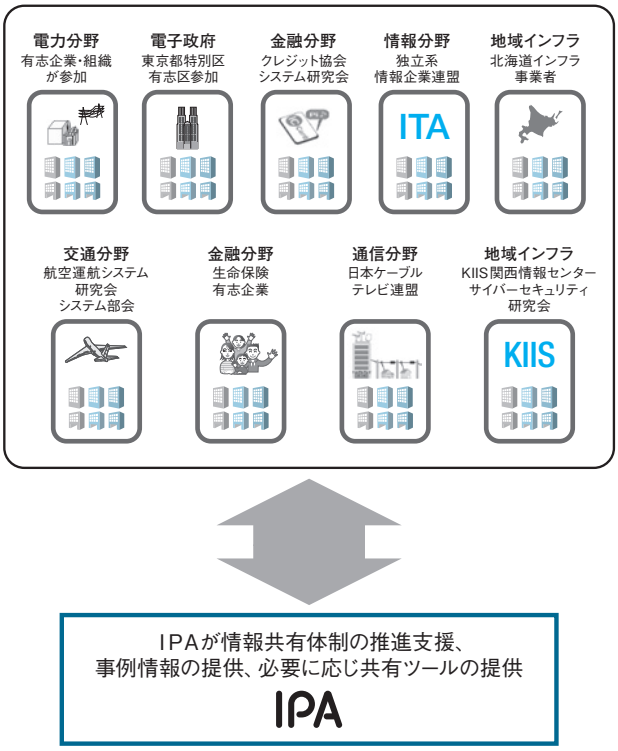


図2　2016年度までに構築した情報共有グループ

5　今後の予定

2017年度も引き続き障害事例を収集し、その普遍化を行い教訓として整理する活動を継続し、教訓集として更なる充実を図っていくと共に、情報処理システムの高信頼化に向けて有益な情報発信を強化していく予定である。

また、社会インフラ情報システムの一層の信頼性向上を目指し、活動を開始したシステム障害情報の共有の仕組みの運営を支援すると共に、新たな産業分野にも普及を働きかけ、自律的な活動を促しつつ、システム障害情報共有の裾野を拡大していきたい。

脚注

※1　重要インフラITサービス高信頼化部会：銀行、保険、証券、電力、鉄道、情報通信、政府・行政などの情報処理システムの有識者・専門家で構成する委員会
※2　http://www.ipa.go.jp/sec/reports/20170327.html
※3　KIIS（Kansai Institute of Information Systems）

組込みシステム

1 活動概要

組込みシステムの障害対策は、前述のITサービスシステムの運用視点とは異なり、モノ作りの視点で議論されてきたため、2016年度は、設計段階で障害回避処理を盛り込むための「障害未然防止のための設計知識の整理手法ガイドブック(組込みシステム編)」※1を検討し公開した。

2 設計知識の整理手法ガイドブック

2.1 背景と目的

情報処理システムや組込みシステムを開発するベンダ企業の多くは、過去の障害事例を一定の様式で記録し障害情報データベースとして蓄積している。一般に「過去トラ(DB)」と呼ばれており、類似障害の再発防止や未然防止のために活用できる情報が埋蔵されているが、実際のところ、ソフトウェア障害に関して効果的に活用されている事例は聞かれない(図1)。

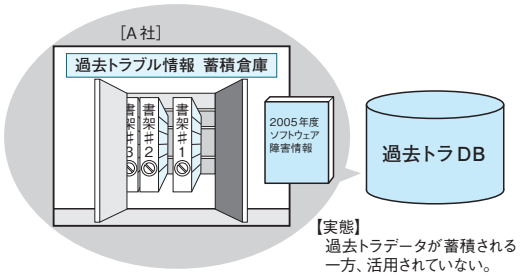


図1 活用されない過去トラDB

「過去トラDB」が有効活用されない理由として、表面的な障害の事象と、どこにどんな対処をしたか程度の情報しか書かれていないことが多く、障害を防止するためのノウハウが整理されていないことが挙げられる。更に、障害事例の原因と対処が装置やサービス固有の具体的な表現で記載されるため、記載内容を見る利用者にとっては、自身の扱うシステムと無関係な障害事例に見えることがある。また、膨大化した「過去トラDB」は、有効情報を取り出す工夫がなされていないことも理由の一つとしてある。一方、業界が抱える課題の一つに技術伝承があり、ベテラン技術者の豊富な経験やノウハウの断片が埋蔵されている「過去トラDB」は、技術伝承の観点においても有効活用したい(図2)。

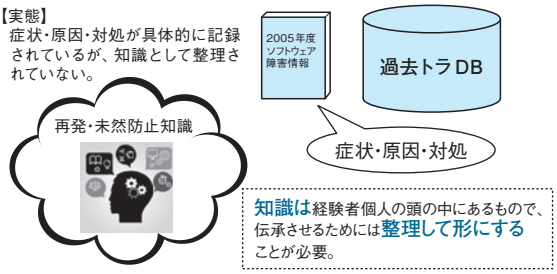


図2 ベテラン技術者の経験やノウハウが埋蔵された過去トラDB

本ガイドブックは、「過去トラDB」を障害の再発防止や未然防止の活動に活用するために、障害情報記録から設計知識を抽出する方法、更に様々なソフトウェアに共通して再利用できるようにタグ設定の考え方を提示する。

2.2 設計知識の整理手順

障害を未然防止するための設計知識を整理する手順を図3に示す。

① 「過去トラDB」から設計知識を抽出する
② 抽出した設計知識を構造化する
③ 更に設計知識を一般化表現に変換する
④ 設計知識の再利用を促すための分類タグを抽出する

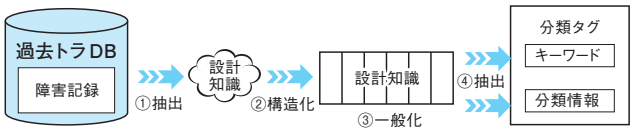


図3 設計知識の整理手順

2.3 設計知識の構造と文脈

障害を未然防止するための設計知識は、図4の構造で整理する。構造化表現された設計知識の知識要素(1)～(4)及び(6)をつなげると下記の設計知識の文脈ができる。知識要素(5)は、(1)～(4)の要素を文章に組み立てたものを入れる。

(1) 障害を引き起こす機能・処理	(2) 考慮漏れしやすい視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

図4 設計知識の構造

【設計知識の文脈】
「(1)の機能や処理を考えるときに、(2)の考慮が漏れていると、(3)が起こった契機で(4)の障害が発生する。その障害の発生を防ぐためには(6)の処理を作り込んでおく。」

2.4 分類タグ

「過去トラDB」から抽出した設計知識の再利用性を高めるために、設計知識に分類タグを付けてDB化する。

分類タグは、図5に示す4種類を提案する。

分類タグは、その知識が何に役立つものかを直感的に理解するためのキーワード"何が""どうなる"(分類タグ2)と、検索の視点で付加する機能・処理(分類タグ1)、装置・デバイス(分類タグ3)、混入プロセス(分類タグ4)で構成している。

(分類タグ 1) 機能・処理	(分類タグ 2.1) キーワード "何が"	(分類タグ 2.2) キーワード "どうなる"	(分類タグ 3) 装置・デバイス	(分類タグ 3.1) 装置・デバイス	(分類タグ 4) 混入プロセス	(分類タグ 4.1) 混入プロセス

図5 分類タグ

脚注
※1 http://www.ipa.go.jp/sec/reports/20170321_1.html

定量的管理による信頼性・生産性向上

SEC研究員　峯尾 正美　SEC専門委員　佐伯 正夫　SEC専門委員　森下 哲成
SEC研究員　塚元 郁児　SEC調査役　三原 幸博　SEC研究員　松田 充弘
SEC研究員　田代 宣子　SECシステムグループリーダー　山下 博之

SEC設立以来、定量的に管理されたソフトウェア開発データを業界から広く収集・分析し、ソフトウェアの信頼性・生産性向上のための統計データを「ソフトウェア開発データ白書」として公開している。2016年度は、従来と同様の全体の分析結果を掲載した本編と、3業種について業種ごとの分析結果を掲載した別冊業種編とから成る「ソフトウェア開発データ白書2016-2017」を発行した。また、ベンチマーキング情報を品質マネジメントなどに活用するための手引きと事例集、上流工程の強化が信頼性向上につながることを定量的に示した分析結果を公開した。組込みソフトウェアに関しては、開発プロジェクトデータの収集を継続すると共に、収集データについて次版データ白書に向けた分析を行った。

エンタプライズシステム分野

1 「ソフトウェア開発データ白書2016-2017」の発行

IPA/SECでは、ソフトウェア開発における定量的管理の普及促進の一環として、国内の多様なソフトウェア開発のプロジェクトデータを収集・分析した「ソフトウェア開発データ白書」を定期的に発行している。

その最新版である「ソフトウェア開発データ白書2016-2017」を2016年10月1日に発行した※1。併せて、白書に掲載しているグラフの元データをダウンロードできるサービスを開始した。

図1 データ白書（本編）

最新版白書（図1）の主な特長、アピールポイントは以下の通りである。

（1）新たに526件のプロジェクトデータを追加し、累計4,067件のプロジェクトデータを使用した分析を実施

（2）生産性・信頼性※2変動要因の網羅的な分析や工程ごと成果物量の分析などを掲載

（3）データ件数の多い金融・保険業、情報通信業、製造業に関し、業種ごとの分析を実施し、別冊化（図2）

金融・保険業編　情報通信業編　製造業編

図2　データ白書（業種編）

信頼性変動要因の分析例を表1に示す。

表1　信頼性変動要因の分析例（SLOCに基づく分析、新規開発）

通番	区分	＊	変動要因候補
1	業種	△*2	業種
2	QCD要求	×	信頼性の要求レベル
3		×	性能・効率性の要求レベル
4		×	重要インフラタイプ
5	実現手段	×	アーキテクチャ
6		○	主開発言語
7		△	プラットフォーム
8		△	開発フレームワークの利用
9	実施体制	×	月あたりの要員数
10		△	外部委託比率
11		×	PMスキル
12		◎	テストスキル
13		○	品質保証体制

通番	区分	＊	変動要因候補
14	開発プロセス	×	設計文書化密度
15		◎	設計レビュー工数密度
16		×	設計レビュー指摘密度
17		×	テスト密度
18		×	テスト検出不具合密度
19		◎	上流工程での不具合摘出比率
20	ユーザ要求管理	×	要求仕様の明確さ
21		×	ユーザ担当者の要求仕様関与
22	組織の成熟度	×	定量的な出荷品質基準の有無
23	開発プロセス	○	テスト検出能率

＊：Welchのt検定結果（P値）、◎：1%有意、○：5%有意、△：10%有意、×：有意でない
＊2：FPに基づく分析では◎

この分析の結果、業種、テストスキル、設計レビュー工数密度、上流工程での不具合検出比率などが信頼性に関連する要因であることが明らかになった。

また、業種編の提供により、各社のプロジェクトの条件により近いデータの参照が可能となった。

2 「統計指標に基づく品質マネジメント実践集」を公開

ソフトウェア開発データ白書や個々の企業が持つ内部ベンチマーク※3情報を活用した「定量的管理による信頼性向上のヒン

トや具体的な改善事例集、または品質マネジメントのための具体的なベンチマーキング※4方法の手引き」として、「統計指標に基づく品質マネジメント実践集」(以下本書)を作成し、2016年7月に公開した※5。本書のベンチマーキングにおける位置付けを図3に示す。

本書の特長は、以下の通りである。

(1)統計指標に基づいた品質マネジメントの代表的なシーン(15シーン)に沿って具体的なベンチマーキング方法を解説

(2)IT企業における具体例(13事例)を含む、豊富なベンチマーキングの具体例(32事例)を掲載

(3)プロジェクト・マネジメント/組織改善につなげていく改善を重視した具体的なベンチマーキング方法を掲載

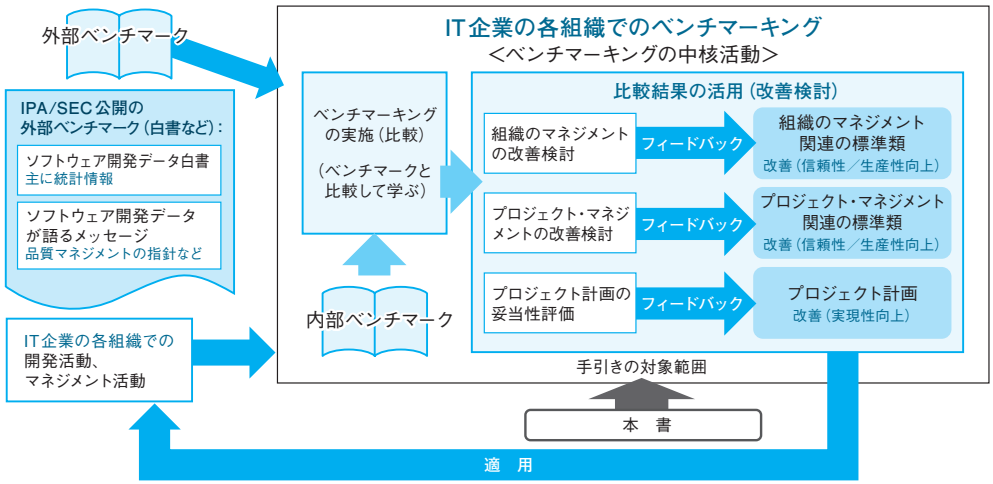


図3 ベンチマーキングにおける本書の位置付け

3 「ソフトウェア開発データが語るメッセージ『設計レビュー・要件定義強化のススメ』」を公開

「ソフトウェア開発データが語るメッセージ」(以下メッセージ)は、ソフトウェア開発の定量的管理の主な目的である「プロジェクト計画の妥当性評価」や「組織の品質マネジメントの改善」に向けて、最新の白書に掲載したデータを分析し、そこから導いたプロジェクトの評価や改善の指針となる考察結果をまとめたものであり、2017年3月に公開した※6。

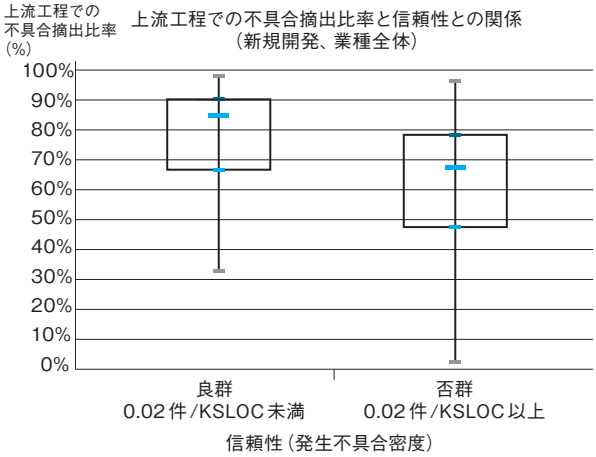
本メッセージでは上流工程(本書では、要件定義、基本設計、詳細設計、製作工程)の強化が信頼性向上のために重要であることを初めて「定量的」に示した。

主なポイントは以下の通りである。

(1)上流工程(基本設計~製作)での不具合摘出比率を高めることによって、信頼性向上が期待できる

上流工程での不具合摘出比率(中央値)は、信頼性が高いグループでは約85%だったのに対し、低いグループでは約66%であった(図4)。このことから、以下のメッセージを導き出した。

「プロジェクト計画/再計画や品質マネジメント改善などのシーンにおいて、上流工程での不具合摘出比率の目標を、目安として85%程度に高めて設定することを目指そう。」



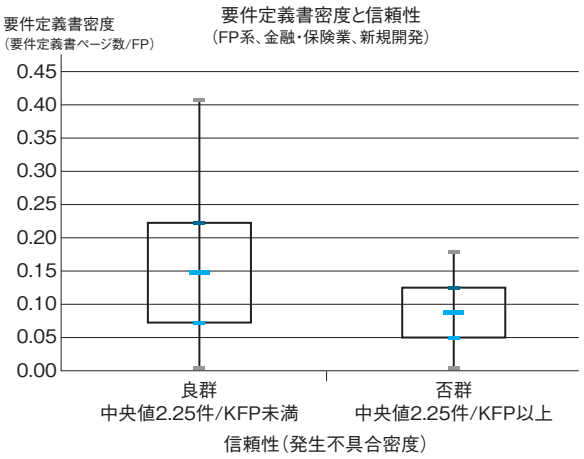
上流工程での不具合摘出比率(%)

発生不具合密度	N	最小	P25	中央	P75	最大	平均	標準偏差
良群 0.02件/KSLOC未満	43	0.0%	65.9%	84.6%	89.4%	97.8%	73.4%	25.5%
否群 0.02件/KSLOC以上	43	1.7%	46.7%	66.3%	78.4%	95.3%	59.3%	26.8%

図4 上流工程での不具合摘出比率と信頼性

(2)要件定義を質、量共に強化することによって、信頼性向上が期待できる

要件定義書密度の中央値は、信頼性が高いグループでは約0.15ページ/FPだったのに対し、低いグループでは0.08ページ/FPだった(図5)。このことから、以下のメッセージを導き出した。



要件定義書密度(ページ/FP)

発生不具合密度	N	最小	P25	中央	P75	最大	平均	標準偏差
良群 中央値2.25件/KFP未満	24	0.007	0.072	0.145	0.228	1.104	0.202	0.228
否群 中央値2.25件/KFP以上	23	0.009	0.048	0.077	0.117	0.274	0.094	0.069

図5　要件定義書密度と信頼性

「プロジェクト計画／再計画や品質マネジメント改善などのシーンにおいて、要件定義を質、量共に強化し、量の目安として要件定義書密度を0.15ページ／FP以上とすることを目指そう。」

4　定量管理の推進

4.1　多様な開発スタイルにおける工数増大リスク軽減の検討

信頼性メトリクスWGでは、多様な開発スタイル(パッケージ利用開発、流用開発、モダナイゼーション)における工数増大のリスクについて調査・検討した。その結果は、今後の定量的プロジェクト管理の普及促進やシステム構築上流工程強化の活動に反映する予定である。

4.2　データ提供企業間での独自分析実施

データ提供企業及び高信頼性定量化部会の委員の中から有志を募り、データ分析実証WGとして、各社独自の切り口での分析を試行し、最終的には「信頼性に裏付けられた生産性」をテーマに信頼性と生産性の関係を分析した。その分析結果は、3節で述べた「メッセージ」に反映した。

4.3　定量的管理の普及推進

定量的管理の普及促進のため、下記の活動を実施した。

- 「ソフトウェア開発データ白書2016-2017」の普及活動の一環として、組込み/IoT総合技術展(ET/IoT2016)にて、パネル展示やセミナーを実施
- 「ソフトウェア開発データ白書2016-2017」並びに関連成果物の普及活動の一環として、関連セミナーを3回実施
- JISA Digital Masters Forumにてベンチマーキング関連を発表
- JUAS QCD研究会にてベンチマーキング関連を講演

5　蓄積ソフトウェア開発データの活用促進

蓄積されているソフトウェア開発データをより一層活用し、ソフトウェアの信頼性・生産性の向上につながる新たな分析手法の発見などを目指し、所定の守秘義務の下で蓄積データを大学などに貸与し、分析方法の研究に活用いただいている。

2016年度は、法政大学、東海大学、静岡大学のほか、米国カーネギーメロン大学SEI(ソフトウェア工学研究所)に貸与し、各大学・研究所の研究に貢献した。

組込みシステム分野

1　組込み分野のプロジェクトデータ収集と分析

「組込みソフトウェア開発データ白書」は、2015年11月に初版を発行し、収集した200件近いプロジェクト管理データから、組込み業界の信頼性や生産性の統計情報を公開した。IoT時代を牽引する日本の組込み業界が、国際競争力を育んでいくためには、まずは業界の実態を統計情報で踏まえる必要がある。本来、組込み業界では、外部に持ち出すことのなかったプロジェクト管理データを、活動の趣旨をご理解いただいた協力的な企業から提供を受けて分析し、公開できたことは、前例がなく高い評価を受けた。

2016年度の活動は、データ提供企業を地道に勧誘することに注力した。その結果、新たに数社からデータ提供を受けることができ、分析対象標本の累計件数が400件を超えた。2015年版では、製品特性に応じて信頼性や生産性は異なるはずだという仮説を定量的に示すことができなかったものの、標本数が増えたことで、自然環境の影響を受ける組込みシステムは、影響を受けないものに比べて、総合テストの工数が多くかかるなど、今まで見えなかった傾向が幾つか見えるようになっている。

2016年度までに収集したデータの分析結果をまとめたデータ白書は、2017年11月に開催される組込み総合技術展(ET2017)での公開を予定している。

脚注

※1　http://www.ipa.go.jp/sec/reports/20161012.html
※2　信頼性：ここでは、出荷後の発生不具合密度
※3　ベンチマーク：特定のITプロジェクトのパフォーマンスが、組織内外のITプロジェクトと比較してどのレベルに位置するかを評価するため、比較対象として利用する組織内外の参照情報
※4　ベンチマーキング：良い成績を収めているプロジェクト群と比較し、それらのやり方(開発プロセス、マネジメント・プロセス、組織の特性など)を参考にして、自組織の業務改善及び組織の改善を進めること。
※5　http://www.ipa.go.jp/sec/reports/20160701.html
※6　http://www.ipa.go.jp/sec/reports/20170331.html

コーディング作法ガイド(ESCR※1)整備の取り組み

SEC調査役 三原 幸博　SEC調査役 十山 圭介

1 コーディング作法ガイドの改訂

IPA/SECでは組込みソフトウェアのソースコード品質の向上を目的に、ESCRとして、コーディングの際に注意すべき事柄やノウハウを取りまとめ、公開している。ESCRでは、コーディングにおける基本的な考え方(作法)と、作法を対象言語に合わせて具体化した個々のルールとを、ソフトウェア品質特性の観点で整理している。組織でコーディング規約を決める際や、コーディング時の参考、また個人のプログラミング学習のために、書籍やPDF版など、これまで3万部を超えて多くの方々にESCRを利用いただいている。

● ESCR[C++言語版]の改訂

ESCRにはC言語とC++言語に向けた2種があり、それぞれ言語規格の更新に追随して改訂を行っているが、2016年度はC++言語の新しい標準規格C++11及びC++14に準拠し、C言語版との整合性も確保したESCR[C++言語版]の改訂作業を終え、2016年10月にVer.2.0として発行した※2。

この改訂について、方針を図1に、変更個所のまとめを図2に示す。また、今回、パブリックコメントを実施すると共に、中田育男先生(筑波大学名誉教授)にご監修をいただいて表現の分かりやすさと正確性の向上を図った。

● セキュアコーディングに向けたESCR[C言語版]の改訂

セキュリティに配慮したコーディングの重要性が高まっており、ESCR[C言語版]を対象としてセキュアコーディング対応の検討を開始した。

これまでのESCRの改訂では、セキュリティ品質特性に対する説明の追加と脆弱性やCERT※3 Cとの対応付けの追加にとどめていたが、以降CERT Cルールの一部やIPAセキュリティセンターからの提案を、新規ルールの追加や解説の拡充といった形でESCRの中に取り込む改訂作業を進めている。

2 コーディング作法ガイドに関する海外連携

MISRA CとMISRA C++はMISRA※4が策定しているコーディングガイドラインで、安全で信頼性あるソフトウェアの開発のため、自動車業界を中心に広範に運用され標準技法としての地位を築いている。IPA/SECでは、ESCRとMISRA Cとで相互引用や、改訂時のレビューを行うなど、MISRAと連携して活動している。

今回のESCR[C++言語版]の改訂では、日本語版と英語版を同時に発行したので、MISRAに英語版を送付し、MISRA WGのメンバから多くのコメントをいただいた。それらへの対応も実施中である。

また、コーディングルールに関して2014年よりNIST※5、CMUとも意見交換を行っているが、今年度は2017年1月にCMU/SEI※6を訪問し、IPAからは、ESCRの改訂状況、及びESCRのルールとCERT Cルールとの対応表の現状について説明した。

(1)セキュリティへの対応
- JIS X 25010で追加された品質特性に対応する説明を追加

品質特性	品質副特性		コードの品質
セキュリティ	機密性	製品またはシステムが、アクセスすることを認められたデータだけにアクセスすることができることを確実にする度合い。	アクセスが認められたデータだけにアクセスすることができることを確実にする度合い。
	インテグリティ	コンピュータープログラムまたはデータに権限を持たないでアクセスすることまたは修正することを、システム、製品または構成要素が防止する度合い。	プログラムまたはデータ領域に権限を持たないでアクセスすることまたは修正することを防止する度合い。
	否認防止性 ・・・		

(2)作法・ルールの改訂方針
- 新機能について、C++11を中心にコーディングの決まりとしてある程度成熟していると思われるルールを取り込む

図1　ESCR C++改訂の方針

ESCR C++ Ver. 1.0 からの作法・ルールの修正数

		Ver.1.0	Ver.2.0	追加	修正	削除
信頼性	作法詳細	21	23	2		
	ルール	60	65	5	19	
保守性	作法詳細	29	30	1	1	
	ルール	82	82	2	27	2
移植性	作法詳細	6	7	1		
	ルール	15	16	1	7	
効率性	作法詳細	1	←			
	ルール	10	←		2	
合計	作法詳細	57	61	4	1	—
	ルール	167	173	8	55	2

- 追加ルール
  - ✓C++11対応　5
  - ✓ESCR C 整合　2
  - ✓他(改善)　1
- 修正ルール
  - ✓C++11対応　24
  - ✓ESCR C 整合　25
  - ✓他(改善)　9
- 削除ルール
  - ✓不要と判断(ESCR C 整合)　2

※軽微な変更は「修正」としてカウントせず　※重複カウントあり

図2　作法・ルール変更個所のまとめ

脚注

※1 ESCR：Embedded System development Coding Reference
※2 http://www.ipa.go.jp/sec/reports/20161018.html
※3 CERT：Computer Emergency Response Team
※4 MISRA：The Motor Industry Software Reliability Association
※5 NIST：National Institute of Standards and Technology
※6 CMU/SEI：Carnegie Mellon University / Software Engineering Institute

上流工程の強化

SEC研究員　山本 英明　SEC研究員　村岡 恭昭　SECシステムグループリーダー　山下 博之

システム構築上流工程の作業不備による開発プロジェクトの失敗や運用後のシステムトラブルがなくならないという背景から、要件定義と再構築の二つの課題に取り組んだ成果をまとめ、ガイドブックを出版した。一つは、ユーザ企業自身が、ベンダ企業の協力を得つつ、抜け・漏れ・あいまいさのない要件定義を行うためのガイドである。もう一つは、システム再構築において、現行業務仕様の理解不足など、不確定な状況に伴うリスクをユーザ・ベンダ間で認識し、その対策について合意の上で開発を進めるために、再構築に特有の内容をまとめたガイドである。

上流工程の課題を解決するユーザガイドを公開

1　背景

ITシステムの役割が現場中心からビジネス中心へ、個別最適から全体最適へと変化する時代に、経営層がITシステム開発の上流に深くかかわることの重要性を2000年代に発信した。10年が過ぎたが、その間も上流工程の重要性が指摘されてきたにもかかわらず、図1のように上流工程の作業不備(手戻り)が発生し続けている。

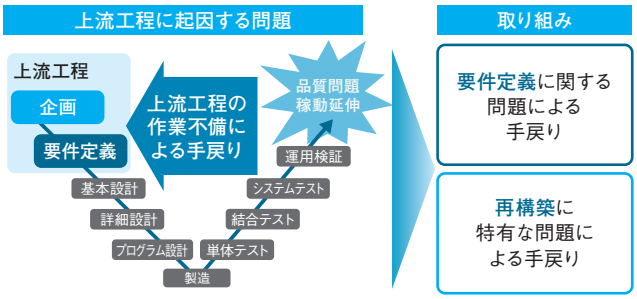


図1　上流工程の作業不備による手戻り

手戻りには二つの傾向がある。要件定義に関する問題によるものと、既存資産を用いた再構築に特有な問題によるものが考えられる。これら二つの問題への取り組みの必要性を鑑みて、それぞれに対応すべき問題を図2のように洗い出し、問題を解決するための課題を整理した。

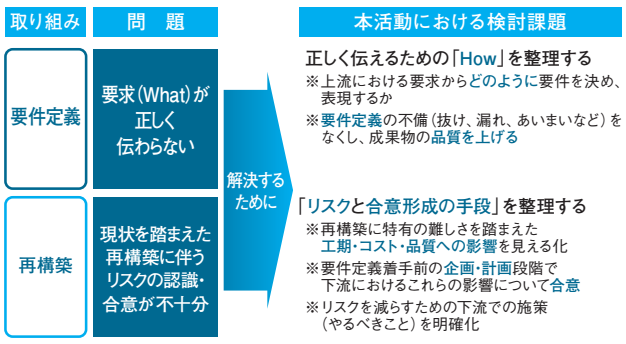


図2　上流工程の主な問題と検討課題

2016年3月にIPA/SEC内に設置したシステム構築上流工程強化部会※1、及び同部会配下のシステム化要求WG、モダナイゼーションWGでは、図3の通り課題解決のための活動テーマを設定し、ガイドブックを作成した。

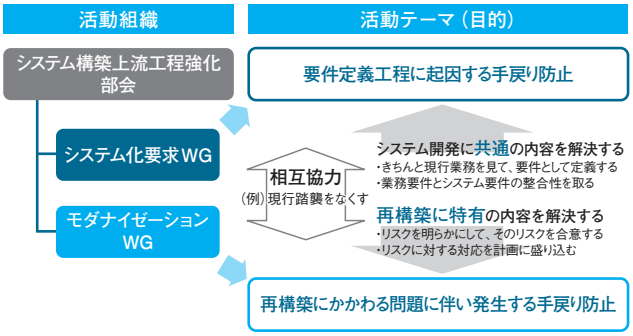


図3　活動組織と活動テーマ

図4に示す通り、システム化要求WGは基本領域の一つである要件定義を対象に、上流における要求からどのように要件を決め、表現し、要件定義の不備(抜け、漏れ、あいまいなど)をなくして、成果物の品質を上げるための勘どころをまとめた。一方、モダナイゼーションWGは応用領域であるモダナイゼーションを対象に、システム再構築に特有の難しさを踏まえた工期・コスト・品質への影響を見える化した上で、要件定義着手前の企画／計画段階で下流における影響について合意して、リスクを減らすための施策(やるべきこと)を明確化した。

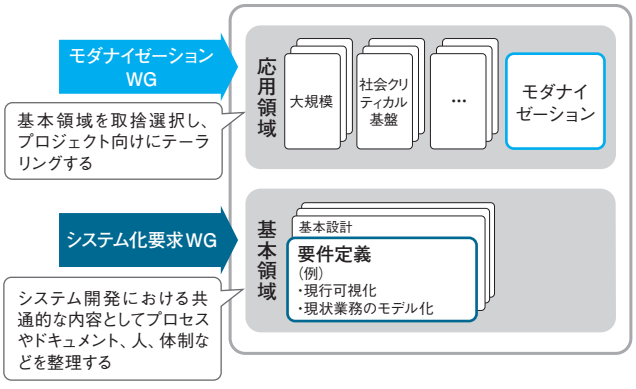


図4　ソフトウェアエンジニアリングの領域と取り組み

二つのガイドのターゲットと主な内容を図5に示す。

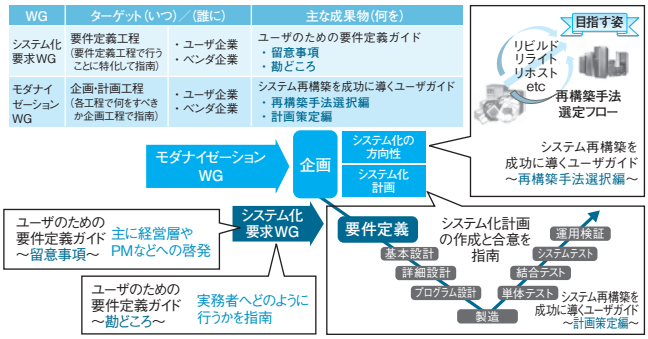


図5　ガイドブックのターゲットと主な内容

2 ユーザのための要件定義ガイド

ユーザ企業からベンダ企業に要求が正しく伝わっていないと、開発プロジェクトの現場で実装する機能は要求を正しく反映したものにはならない。この問題の解決には、ユーザが要求を抜け・漏れなく定義するために実施すべきことを明確にすることが重要である。そこで、システム化要求WGの活動を通じて、ユーザ企業とベンダ企業の知見やノウハウをまとめた“勘どころ(コツ)”を示した「ユーザのための要件定義ガイド ～要求を明確にするための勘どころ～」(以下、要件定義ガイド)を出版※2した。

要件定義ガイドの構成と、ポイントとなる留意事項と勘どころの内容、及びその狙いを図6に示す。

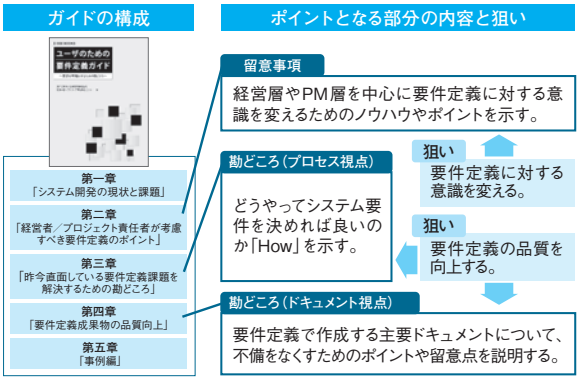


図6　要件定義ガイドの構成

本書は、主にユーザ企業でITシステムの要件定義を実施する読者を対象に、要件定義において発生する問題と、その解決方法をまとめた。

システム開発で発生する問題の半数は、「要件定義」の不備に起因していると言われている。要件定義の不備は、工程が進めば進むほど修正に多大な労力が必要となる。要件定義を行う過程では、明確な目標の設定、膨らむ要求のコントロール、業務の複雑性の軽減、多様なステークホルダとの合意など、様々な課題に直面する。これらに適切に対応すれば、「下流工程で多大な修正が発生する」といった問題の発生を抑制できる。

本書では、こうした課題について、熟練した有識者がこれまでのプロジェクト経験から「ありがちな間違いとその解決策の勘どころ」を、具体例を挙げて分かりやすく解説した。

3 システム再構築を成功に導くユーザガイド

長期にわたって保守開発をしてきたシステムの再構築は、簡単ではない。そこには、「現行業務仕様の理解不足」などによる再構築に特有の難しさという問題が内在する。この問題の解決には、上流工程においてリスクを明らかにして、対策をユーザ企業とベンダ企業で合意することが重要である。そこで、モダナイゼーションWGの活動を通じて、ユーザ企業とベンダ企業の知見やノウハウをまとめた「システム再構築を成功に導くユーザガイド ～ユーザとベンダで共有する再構築のリスクと対策～」(以下、再構築ガイド)を出版※3した。

再構築ガイドの構成と、ポイントとなるリスクの把握から合意までの内容、及びその狙いを図7に示す。

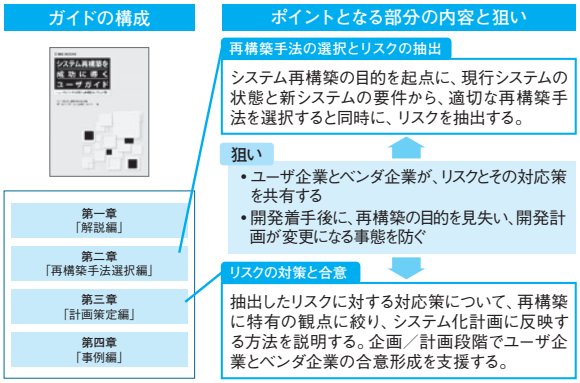


図7　再構築ガイドの構成

本書は、ユーザ企業が現行システムを再構築する際に、最適な手法を選択し、正確でかつベンダ企業側と齟齬を生じさせない「システム化計画」の策定を支援する。

システム再構築における開発プロジェクトの失敗や運用後のシステムトラブルには、多くの企業が頭を抱えている。そのような問題が発生する原因は、時間が経過して不明瞭になっている現行システムの仕様があいまいなまま、それをもとに新システムの開発に着手してしまうことである。これを解決するためには、過去に同様の課題に直面した企業の知見に倣い、再構築の企画／計画段階で検討すべき観点を整理することが有用である。

本書では、ユーザ企業がシステム再構築の企画／計画段階で留意すべきポイントを一覧化し、下流工程でのリスク回避のための具体的な計画の策定方法を、実践に即した形式で紹介している。ユーザ企業はベンダ企業との合意形成をする上で、本書を活用できる。

4 今後の予定

ガイドブックの普及のため、セミナーやイベントでの発信を行うと共に小冊子を作成する。経営層など企業の上位層や、経験が浅い読者層にもガイドブックの内容を理解いただけるようにまとめる予定である。

また、再構築ガイドは続編を作成する。品質保証における重要な観点である「業務継続性の担保」を深掘りして、内容を拡充する。また、再構築のパターンのうち、昨年度はスコープ外とした、業務仕様を変更するパターンや、パッケージを選択するパターンを追加する。

脚注

※1　ユーザ企業やベンダ企業の専門家や経験者による議論を行う委員会
※2　http://www.ipa.go.jp/sec/publish/tn16-008.html
※3　http://www.ipa.go.jp/sec/publish/tn16-009.html

ソフトウェア工学分野の先導的研究支援事業について

SEC 調査役　小沢 理康

IPA/SECでは我が国におけるソフトウェア工学・システム工学分野の研究の促進及びその成果の産業界への展開を図るため、「ソフトウェア工学分野の先導的研究支援事業」を2012年度より実施してきた。開始から数えて20件の研究を支援しており、このうち最後の3件が2016年度に完了し、成果をIPAのWebサイトで公開した。本稿では2016年度に完了した研究成果について報告する。

1　研究支援事業の概要

ソフトウェアは、あらゆる産業や市民生活を支える基盤として不可欠な存在となってきており、複雑化・大規模化するソフトウェアの高信頼化や開発プロセスの高度化、それらの運用や保守についても様々な課題が存在している。このような課題に対して工学的なアプローチで解決策を提供しようとするソフトウェア工学や複雑な統合システム(System of Systems)へのシステム工学の適用にかかわる研究、また、ソフトウェアの経済的効果に関する研究についての一層の振興をねらいとして本事業を実施してきた。なお、2012年度から支援してきた研究すべてが完了したことから、本事業は終了することとなった。完了した20件の研究成果は以下のURLにて閲覧可能となっている。

http://www.ipa.go.jp/sec/rise/index.html

2　2016年度に完了した研究の成果

2016年度に完了した研究は、2014年度に採択した期間が2年半の研究が1件と、2015年度に採択した期間が2年の研究が2件の計3件である(表1参照)。

それぞれの研究成果の概要を以下に示す。

日本のソフトウェア技術者の生産性及び処遇の向上効果研究：アジア、欧米諸国との国際比較分析のフレームワークを用いて

(学校法人同志社 同志社大学)

ソフトウェア技術者の心的生産性と労働条件を国際的に比較すると共に、生産性、労働条件の決まり方を解明する目的で調査研究を実施した。調査対象としたのは、日本、アメリカ、ドイツ、フランス、中国の5カ国で、各国ともソフトウェア技術者個人を対象に2015年～2016年にかけてアンケート調査を行った。調査の結果、日本のソフトウェア技術者は、仕事に対する「やりがい」や「満足度」といった心的生産性指標が、調査対象国の中で最下位であることが明らかになった。また、週実労働時間は最長となり、他国と比較して労働環境が問題となっていることが浮き彫りになった。

D-Caseに基づく議論構造可視化支援ツールの開発と、スマートコミュニティにおける合意形成の実証

(国立大学法人電気通信大学)

複雑化するスマートコミュニティ(人間系＋リアルタイムセンシング)の要件定義から評価改善フェーズにおけるディペンダビリティ合意形成を支援する手法を開発、その有効性を実証し社会還元するため、(1)D-Case※に基づく利害関係者間のコミュニティ合意形成支援ツール「Smart Structure」(以下ツール)の開発、(2)実際のスマートコミュニティ「ポケットガイガー」のSNSコミュニティへのツール適用とツールの有効性評価、(3)利害関係者間が合意形成に至るインタラクションのモデル化、ツール・ライブラリのオープン化による社会実装を行った。

測定評価と分析を通じたソフトウェア製品品質の実態定量化および総合的品質評価枠組みの確立

(学校法人早稲田大学 早稲田大学)

開発・運用・検討中のソフトウェア製品の品質を、定量的かつ総合的に評価可能とし、改善や選択に役立てることを目的とした研究。研究チームが国際的にリードする国際規格群及び品質測定法群を発展させ、日本の主要ソフトウェア製品群の品質の実態調査を通じて、ソフトウェアの内部品質、外部品質、顧客・利用者からの満足度・評価を含む利用時の品質を定量的に測定評価し、異なる品質間の関係を総合的に明らかとする枠組みを確立した。

表1　2016年度に完了した研究

期 間	研究テーマ名	提案者名
2年半	日本のソフトウェア技術者の生産性及び処遇の向上効果研究：アジア、欧米諸国との国際比較分析のフレームワークを用いて	学校法人同志社 同志社大学
2年	D-Caseに基づく議論構造可視化支援ツールの開発と、スマートコミュニティにおける合意形成の実証	国立大学法人 電気通信大学
2年	測定評価と分析を通じたソフトウェア製品品質の実態定量化および総合的品質評価枠組みの確立	学校法人早稲田大学 早稲田大学

脚注

※ D-Case：システムのディペンダビリティをシステムにかかわる人たちが共有し、理解し合うと共に、第三者への説明責任を果たすための手法とツール。

プロモーション活動

SECプロモーショングループリーダー　佐藤 康彦

例年実施しているセミナー開催や展示会出展に加え、地方自治体との連携セミナーや国際会議への参加など、普及領域の拡大を図ると共に、事業成果の効果的なプロモーションに注力した。本稿では、2016年度の事業成果の普及活動を紹介する。

1 SECセミナー開催

IPA/SECの事業成果をテーマにIPA職員や外部有識者によるセミナーを毎年度開催している。2016年度は、43回開催し、延べ参加人数は1,946名。海外または国内地域団体(行政機関)と連携するSEC特別セミナーは、4回開催し、延べ参加人数は691名と、多くの方に参加いただいた。

2 展示会出展

事業成果普及の一環として、組込み総合技術展(ETWest2016(7月7日～8日開催)、ET/IoT2016(11月16日～18日開催)、主催：一般社団法人組込みシステム技術協会)に出展。組込み系に関連する事業のパネル展示や資料配布、デモなどを実施した。「つながる世界の開発指針」や「組込みソフトウェア開発データ白書」「STAMP/STPA」への関心が高く、展示ブースは連日多くの来場者でにぎわった。

3月20日から24日までドイツ、ハノーバーで開催されたCeBIT2017にも出展。展示ブースで事業成果を紹介し、IoT分野での国際連携を視野に入れた普及活動を実施した。

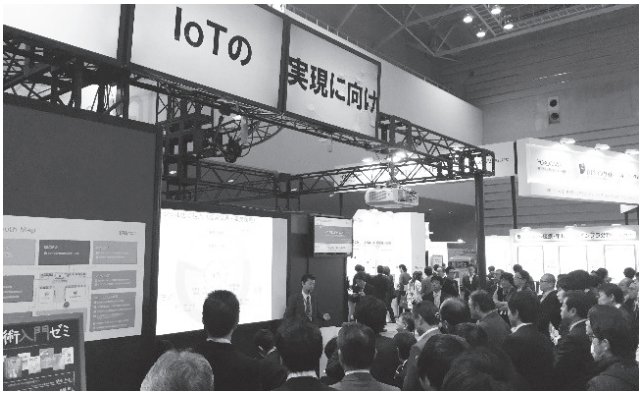


写真1　ET/IoT2016 IPAブースの様子

3 カンファレンス／ワークショップ共催

IoTをテーマに経済産業新報社と共催した「IoTイニシアティブ2016」(11月4日開催)をはじめ、「第1回STAMPワークショップ in Japan」(12月5日～7日開催)や「システムズエンジニアリング・公開ワークショップ」(12月19日開催)など、業界団体や大学などと連携したイベントも多数実施した。例年、国立研究開発法人宇宙航空研究開発機構(JAXA)と共催しているクリティカルソフトウェアワークショップ(WOCS[2]：Workshop on Critical Software System)ではIPA理事長による基調講演のほか、「より良き未来のために見る・知る・学ぶ・考える」と題したテーマのパネルディスカッションを企画し実施した。

4 映像制作

写真2　動画「つながる世界の開発指針」の一場面

IPA/SECは、安全・安心が確保された信頼できる製品を開発するために「つながる世界の開発指針」を策定し、2016年3月に公開した。その内容を分かりやすくかみ砕き、実写ドラマ形式の動画を制作した。本動画は、セミナーや展示会の幕間で投影するほか、以下のIPA channel(YouTube)でも視聴が可能である。

http://youtu.be/wz90l_nwmko

5 まとめ

2017年度は、IPA第三期中期計画(2013～2017年度)の最終年度にあたり、中期計画の総括の年でもある。事業成果を活用する側に立ち、これまで以上に事業を分かりやすく伝える工夫と普及を図っていく。

産業サイバーセキュリティセンターの取り組み

IPA 産業サイバーセキュリティセンター 副センター長 片岡 晃

2017年4月に「産業サイバーセキュリティセンター（Industrial Cyber Security Center of Excellence, ICSCoE）」（以下「センター」）が設立された。本稿では、センター設立の背景、センターが提供する価値について紹介する。

1 センター設立の背景

近年、社会インフラに物理的なダメージを与えるサイバー攻撃のリスクが増大している。海外においては、既に、ほかの国家などからのサイバー攻撃により、重要インフラや産業基盤の安全が脅かされる事案が発生している。幸い日本では、重要インフラを著しく破壊するようなサイバー攻撃はいまだ確認されていないが、2020年の東京オリンピック／パラリンピックを控えた今、我が国の経済や社会を支える重要インフラや産業基盤のサイバー攻撃に対する防御力を抜本的に強化する必要がある。

実際、2012年ロンドンオリンピック／パラリンピックでは、毎秒1万件の不正通信が行われ、更に開会式会場の電力システムへの攻撃情報が寄せられたことから、直前に手動システムに切替えを行うなどの処置を講じた。2016年リオオリンピック／パラリンピックでは、大会運営に大きな影響を及ぼすサイバー攻撃はなかったものの、大規模なDDoS攻撃が行われたほか、開催期間中には約400万もの脅威が発見され、対処に追われたという。

本センターでは、サイバーセキュリティ人材の育成にかかわる事業を行うと共に、実際の制御システムの安全性・信頼性の検証や、最新の攻撃情報の調査・分析などを通じて、サイバーリスクに対応する人材・組織・システム・技術を生み出し、官民が共同してサイバーセキュリティ対策強化を図るための中核拠点となることを目指していく（図1）。

2 センターが提供する価値

まず、「人材育成事業」について、実際にどのように人材育成を行っていくのかを紹介したい。本センターでは、情報系システムから制御系システムまでのシステム全体の安全性・信頼性を客観的に評価し、サイバーセキュリティ確保に必要な技術・コストの精査を行い、総合的な戦略として経営幹部に働きかけを行っていくことができる人材、更に、日々高度化を続けるサイバー攻撃について、最新のトレンドに精通し、他業界や海外の対策状況を把握すると共に、それらを自社の対策立案に効果的に反映することができる人材を育成していく。

具体的には、民間企業から派遣された研修生に対し、「中核人材育成プログラム」と呼ばれる約1年の教育プログラムを提供する（図2）。情報系システム及び制御系システムのテクノロジースキル講座を核とし、約1年間にわたって下記のような様々なコースが展開される（図3）。

① 制御システム固有のセキュリティ（CSS）関連技術及びマネジメントを理解し、模擬システムなどを用いた演習及びアクティブラーニングを通し、CSSの計画立案・対策に関する理解度を高めるコース

② 制御システムにおけるサイバーインシデントを想定した社内の対応体制と指揮について、必要な知識・ルール・スキルを網羅的に習得し、インシデント発生時の緊急体制への切替えタイミングとその際の情報共有、そのために日頃か

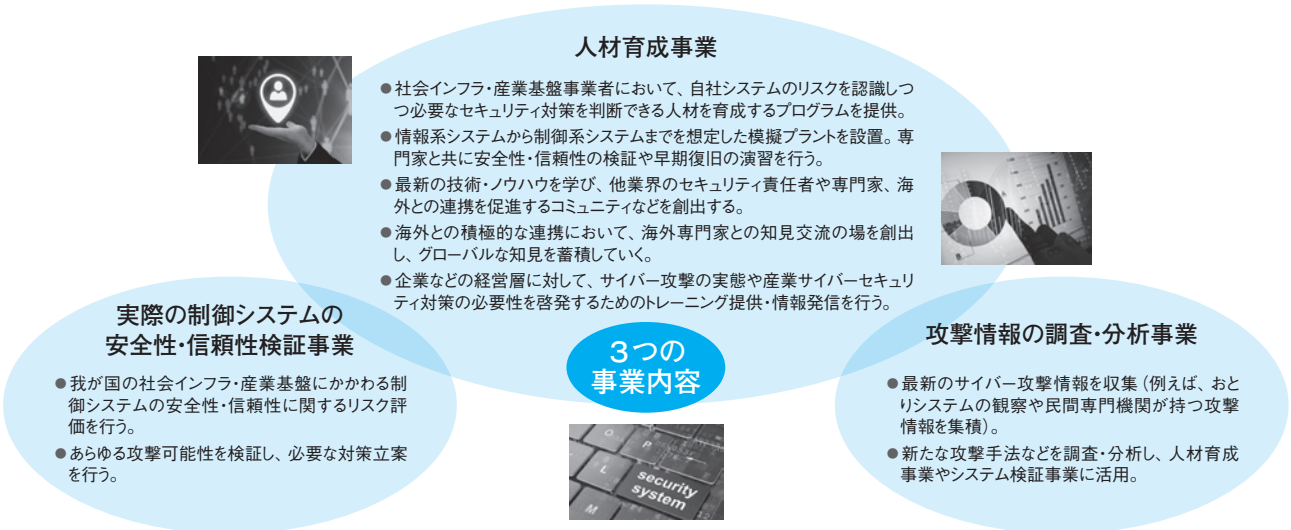


図1 産業サイバーセキュリティセンターの事業内容

（出典）IPA作成 産業サイバーセキュリティセンター設立にあたってのパンフレットより

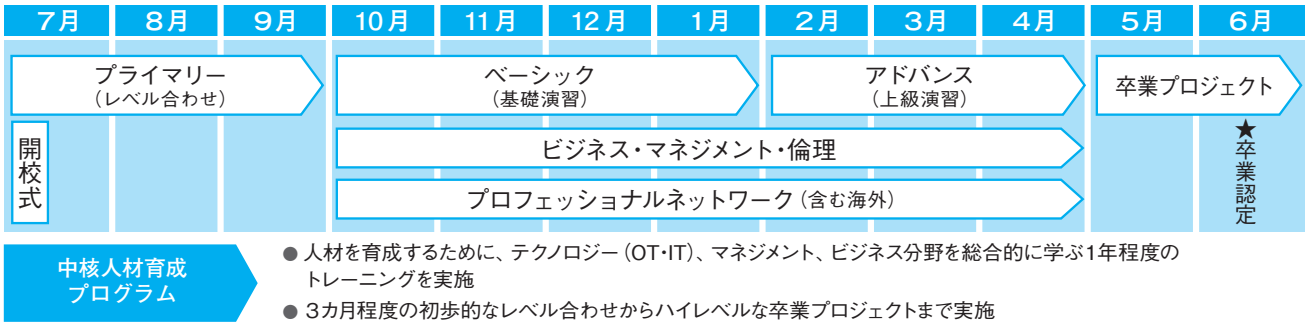


図2　産業サイバーセキュリティセンター 中核人材育成プログラムの流れ

（出典）IPA作成 産業サイバーセキュリティセンター設立にあたってのパンフレットより

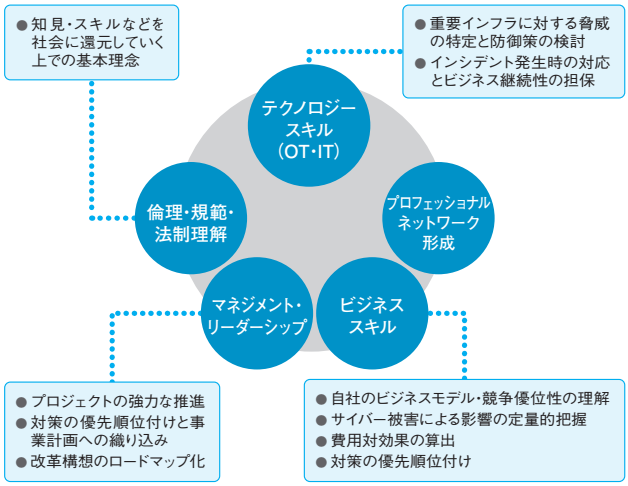


図3　産業サイバーセキュリティセンター 中核人材育成プログラムの内容

（出典）IPA作成 産業サイバーセキュリティセンター設立にあたってのパンフレットより

ら行っておくべき対応など総合的なマネジメントの理解を深めるコース

③ サイバーセキュリティに関するガバナンス、リスクマネジメント、インシデント対応、脅威情報の共有及びセキュリティ技術など、組織をサイバー攻撃から守る上で必要な知識をハンズオン形式も取り入れることにより網羅的に習得するコース

など、実践力を強化するための内容を多く盛り込んでいる。

更にサイバー被害による影響の定量的把握や費用対効果の算出方法、対策の優先順位付けと事業計画への織り込みなどを学ぶビジネスマネジメントコース、倫理・規範・法制理解を促すコースなども設けている。加えて、世界最先端の知見を獲得できる場も不可欠であるとして、現在、米国国土安全保障省（Department of Homeland Security, DHS）と共に演習プログラムを実施する予定で調整を進めているほか、欧州の国際標準などにフォーカスしたクラスの提供や、成績優秀者などを対象とした海外学会への参加などを検討していく。

そして、卒業前最後の2カ月間は、個人やグループで総合的なプロジェクトに取り組むこととし、各研修生が自社に戻った際、より実践的なサイバーセキュリティ対策を構築できるよう支援をしていく。

受講生は、1年間という長期プログラムであることを活かすことで、業界の壁を越えて交流し、講師である専門家、海外コミュニティも巻き込みながら、将来にわたっての協力関係・プロフェッショナルネットワークを構築していくことが期待される。

また、本センターでは、企業の経営層などを対象とした短期プログラムを開催する。このプログラムでは、最新のインシデント事例、セキュリティ脅威に対する戦略的視点、リスクに対処するための最先端アプローチの紹介を行う、また、セキュリティインシデントを特定、管理、解決するために備えるべき検討事項を取り上げ、インシデント管理のあり方や改善方法についての考え方を共有すると共に、擬似的なサイバー攻撃のシナリオと、それに基づいて発生し得るイベントに沿ってグループ学習を行う。更に、業界別に企業のCIOやCISOが直面するサイバーセキュリティ上の具体的な課題（法的規制への対応、体制整備、人材確保、人脈構築、技術確保、社内文化醸成など）についてのトレーニングなどの内容についても検討を進めている。企業の経営層が、迅速にサイバーセキュリティ対策の指揮を執り、組織体制の変革を行うことは、サイバーセキュリティ人材が社内で正しく評価されることにつながり、より効率的な対策の実装につながっていくと期待する。

次に、「実際の制御システムの安全性・信頼性の検証事業」を紹介する。この事業では、重要インフラや社会基盤を支える民間企業において、実際に使用されている制御系システムの安全性や信頼性を検証するためのリスク評価（ペネトレーションテスト）を実施する。テストにおいては、あらゆる攻撃可能性を検証し必要な対策立案を行うと共に、そこで得られた知見を用い、業界全体向けのガイドラインを整備し、サイバーセキュリティ対策の総合的なノウハウを創出していくことを目指す。

最後に、「最新の攻撃情報の調査・分析事業」を紹介する。この事業では、本センターに所属する専門家が中心となり、民間専門機関が持つ攻撃情報の収集や最新のサイバー攻撃情報収集などを実施していく。今年度はIPA内部の関連組織と連携をしながら、まず体制の構築を進めていく。

このように、産業サイバーセキュリティセンターは、我が国におけるサイバーセキュリティ対策強化のための中核拠点となるべく、人材育成事業から調査分析事業に至るまで、多角的機能を担っていく。官民が有機的に連携することで、サイバーセキュリティ対策の効率的かつ効果的な実装を促し、強靭な重要インフラや社会基盤を構築していきたいと考えている。

米国における有力組織との意見交換

SECシステムグループ 主任 八嶋 俊介　SEC研究員 峯尾 正美
SEC研究員 小崎 光義　SEC職員 山田 彩歌

1 はじめに

IPA/SECでは、国際連携活動の一環として、米国の有力なソフトウェア技術拠点であるNIST（米国商務省国立標準技術研究所※1）、SEI（カーネギーメロン大学ソフトウェア工学研究所※2）と定期協議を行っている。今回もこの2組織を訪問し、最新の取り組み事項について意見交換を行った。2017年1月23日から1月27日にかけての上記米国出張について、その内容を報告する。

2 NISTとの意見交換

写真1　NISTとの意見交換の様子

（1）NISTの活動状況について（NIST）

NISTの関連活動として、セキュリティを中心としたITL※3の取り組み、EL※4のスマートグリッドへの取り組みとCPS Frameworkの最新状況に関する説明が行われた。昨年度も伺ったスマートグリッドに加えて、スマートシティに向けたフレームワークの検討が進められていることが分かった。また、現在発行されているCPS Frameworkではセーフティに関しては今後の検討となっていたが、米SAE※5と連携して自動車分野への機能安全を考慮した適用について取り組み、その結果のフィードバックが計画されていることが分かった。その他、テストベッドの検討や、UMLでモデル化するためのオープンソースツールの検討が行われていることが分かった。

（2）つながる世界の開発指針に関連する活動状況について（IPA/SEC）

IPA/SECが取り組んでいる、つながる世界の開発指針、開発指針に基づいた実証実験、及び開発指針の普及状況について説明した。IPA/SECもNISTもそれぞれ、IoT、CPSをSystem of Systemsとして捉えていることについて共通理解が得られた。

IPA/SECからは、とくに、日米独での安全安心なIoTについての国際標準化についての連携について提案した。現在、IoTに関して、日米独で、それぞれ個別に標準の整備が行われており、それらをすり合わせて国際標準化を進めることが必要である。一方で、そのためには、コモンランゲージ（共通の用語）がないと、それぞれの標準の差異の明確化が進まないことが想定されるため、NIST CPS Frameworkは、OSIの7層モデルのように、標準化する場合の共通モデルとして使って欲しいという立場であることが分かった。今後は、開発指針の国際標準化にあたってNIST CPS Frameworkのモデルを意識して提案するなど、引き続き相互に連携して進めていく予定である。

3 SEIとの意見交換

写真2　SEIとの意見交換の様子

（1）ソフトウェア開発データ分析にかかわる意見交換

一昨年、NDAを締結して送付したIPA/SECデータ白書のデータに基づいたSEI側の研究テーマ内容に関して、状況確認と意見交換を行った。

SEI側の研究テーマは、「IPA/SEC、TSP（Team Software Process）データにおける因果関係モデリングツールを用いた因果関係の分析」であり、分析項目として、昨年IPA/SECより送付した「ソフトウェア開発データが語るメッセージ2015」で述べた分析項目のうち、IPA/SEC、TSPで共通に収集している項目が候補となる。訪問時点では、プレ分析を実施済みで、幾つかの項目では、IPA/SEC側と同様な相関関係が得られているとの報告があった。

今後は、ツールを用いて、それらの間の因果関係を分析するとのことで、データの更新を含め、協力して研究を推進したい。

（2）システムズエンジニアリング推進の取り組み状況について

IPA/SECが行っている、システムズエンジニアリング推進のための活動や、欧州における導入状況（IESE※6への委託調査結果）について紹介し、意見交換を行った。SEIにおけるシステムズエンジニアリングに対する考え方の概要は、以下の通りである。

- システムは複数の観点から、検討されるべきである。
- 今や、ソフトウェアに関係しないシステムはあり得ない。
- システムズエンジニアリングには、以下の3つのタイプが存在する。
  ①要求をいかに実現するかの「プログラム型」
  ②実現可能性を検討する「発見型」
  ③論理的なアプローチを取る「アプローチ型」
- システムズエンジニアリングの考え方は、万国共通的なものである。
- 米国の、とくに大企業では、システム全体を見る専門の組織があって、すべてのインターフェースに責任を持つという点で、日本とは組織構成上も大きな違いがある。
- システムズエンジニアリングのエンジニアは、技術的だけでなく、人間的な面にもかかわる場合がある。
- システムズエンジニアリングのエンジニアは、自分の専門的知識を深く学ぶ一方で、多方面の知識を広く浅く学び、「T字型」の知識を持つ必要がある。更に、最近のソフトウェア、セキュリティに関しては、双方向に幅広く知識を持つ必要がある。こうした人材の育成については、米国でも大きな課題となっている。

（3）IPA/SECの組込み分野への取り組みについて

IoTと深い関係のある組込み分野について、IPA/SECが取り組んでいる活動（ESCR※7）の紹介と、ESCR C++の改訂状況（ver.2.0を発行／公開）の説明を行った。昨年のC言語版（英語版）に引き続き、C++言語版（英語版）の書籍を持参し、成果物の内容について具体的に紹介した。また、C言語版は、SEIから公開されているSEI CERT C Coding Standard※8に対応する形で改訂作業中であることを説明した。SEIでは、SEI CERT C Coding Standardのツールも公開しているとのことで、今後の事業への活用を検討する。

（4）STAMP/STPAを中心とした安全性解析手法についての意見交換

IPA/SECの安全性解析手法普及への取り組みとして、STAMP/STPA※9を用いた日本国内のシステムの安全性解析の事例や、MIT※10の研究者や日本国内の産学の有識者を招いてSTAMP/STPAの活用事例について講演・議論を行ったワークショップ※11などの紹介を行った。SEIからは、STPAとAADL※12を組み合わせた原子力発電所の制御システムや医療システムの安全性解析に関する研究の紹介があった。

（5）SEIの研究“SEI Agile in Government”について

SEIにおける米国政府へのアジャイル普及活動の紹介があった。具体的には、①政府に特化したアジャイルのトレーニング、②アジャイル適用に関する政府組織への指導、③ソフトウェア開発から調達など全ライフサイクルにわたる技術的経験の提供であり、コスト削減のみでなく、早期の価値提供や変化へのリスク低減などのアジャイルの価値を、上層部を含めた関係者に納得してもらい実現しているとのことであった。

また、アジャイル開発の効果を定量的に測定することも行っており、外部関係者に向けた全体システムレベルと、開発管理用のチーム内のレベルでの測定手法があるとの紹介があった。

4 おわりに

NISTに関しては、今回は、CPSやIoTに関する取り組みについて相互の具体的な活動に関して、昨年度以上に理解が深まり有意義であったと考えている。IoTの安全安心に関する国際標準化に向けては、今回の訪問でNIST CPS Frameworkの方針が把握できたように、連携体制の構築には、各国の標準類への取り組みの方針を確認しつつ構築していくことが必要であると感じられた。

SEIに関しては、ソフトウェア開発データ分析における協力研究の進捗とIPA/SECの分析との方向性の一致が確認できた。また、システムズエンジニアリングや組込み分野のセキュアコーディングなどの各分野においても、引き続き協力していくことを確認した。

脚注

※1 NIST：国立標準技術研究所（National Institute of Standards and Technology）は、アメリカ合衆国商務省の技術部門であり、計量、標準化、基礎技術研究などを主な任務としている。

※2 SEI：カーネギーメロン大学ソフトウェア工学研究所（Carnegie Mellon University, Software Engineering Institute）は、アメリカ合衆国ペンシルベニア州に本部を置くカーネギーメロン大学に設置されているソフトウェア開発、ITセキュリティなどの研究機関である。

※3 ITL：情報技術研究所（Information Technology Laboratory）は、国立標準技術研究所に設置された研究ユニットの一つである。

※4 EL：情報技術研究所（Engineering Laboratory）は、国立標準技術研究所に設置された研究ユニットの一つである。

※5 SAE：SAE（Society of Automotive Engineers, Inc.）は、モビリティの専門家を会員とするアメリカの非営利団体である。

※6 IESE：フラウンホーファー研究機構（Fraunhofer-Gesellschaft）実験的ソフトウェア工学研究所（Institute for Experimental Software Engineering）。

※7 ESCR：Embedded System development Coding Referenceは、組込みソフトウェアを作成するにあたって、ソフトウェアのソースコードの品質をより良いものとするために、コーディングの際に注意すべきことやノウハウを体系的に整理したものである。

※8 SEI CERT C Coding Standardは、C言語を使ってセキュアコーディングを行うためのルール（Rule）とレコメンデーション（Recommendation）を定めたものである。

※9 STAMP：System Theoretic Accident Model and Processesとは、マサチューセッツ工科大学（MIT）のNancy G. Leveson教授が提唱したシステム理論に基づく事故モデルであり、STPA：System Theoretic Process Analysisとは、STAMPの理論に基づく、相互作用する機能単位でリスクを考える新しい安全性解析手法である。

※10 MIT：マサチューセッツ工科大学（Massachusetts Institute of Technology）は、アメリカ合衆国マサチューセッツ州に本部を置く私立大学であり、5つのスクールと1つのカレッジ、51の研究機関が設置されている。

※11 「第1回 STAMPワークショップ in Japan」2016年12月5日から3日間、九州大学にて開催。

※12 The SAE Architecture Analysis and Design Language：システムの構造や動作を、処理時間などの非機能的な要素も含めて可視化するアーキテクチャ記述言語。

CeBITへのIPA出展について

SEC副所長 和田 恭　SEC専門委員 新谷 勝利
HRDイニシアティブセンター 調査役 林口 英治

1 はじめに

2016年の伊勢志摩サミットで、メルケル首相から安倍総理への要請があり、2017年3月開催の情報技術展CeBIT（ハノーバー）のパートナー国を日本が務めることとなった。そのため、我が国の産業界からも同展示会への積極的な出展が求められていた。IPAも2017年3月20～24日の期間、現地にて出展を行い、事業内容の普及に努めた。IPAから富田達夫理事長以下が関連式典に出席すると共に、CeBITに展示参加し、事業成果の発信を行った。

2 CeBIT関連行事

① 歓迎パーティー

3月19日、安倍総理、メルケル首相や関係大臣、企業トップなど計3,500人が参加した歓迎パーティーがドイツメッセで開催された。主催者であるBITKOM会長から、ドイツにおけるインダストリ4.0の進捗状況報告、教育研究大臣によるスタートアップのCeBIT表彰などが行われた。安倍総理からはドイツにおけるインダストリ4.0を踏まえ、日本は更に自由貿易と投資に基づくイノベーションを、日独が協力しながら進めていこうというメッセージが出された。

② 展示

CeBITでは、3月20～24日の5日間にわたり展示が行われた。IPAは、ジャパンパビリオンを総括するJETROの情報提供ブースの中で展示を行った。IPA展示に関しては、国際的にあまり類を見ないユニークな取り組みだということと、その内容の有効性及び公的機関としての活動の位置付けに関心を示す方が多かった。

CeBITは、情報産業の展示会だということもあり、もの作りよりB2BでのIoT導入や適用に力点が置かれており、SAPや、スタートアップ企業によるサービスの展示が主力であった。ただし、インダストリ4.0の推進という意味では、その中心的な推進事業者とされているシーメンスやボッシュも、4月のハノーバーフェア（同じくドイツメッセで開催）への出展に注力している模様である。

写真1　SEC展示（右側）と対応者（左から林口、新谷）

③ フラウンホーファーIESEとの特別イベント「Fraunhofer welcomes Japan」

ドイツの中立的な研究機関であるフラウンホーファー研究機構はCeBITにも出展しており、同機構ブースにて、3月22日、日独の協力関係を記念した「Fraunhofer welcomes Japan」イベントが開催された。IPAは、同機構の一つである実験的ソフトウェア工学研究所（IESE）と12年来の協力関係にあり、これまでの協力成果と今後の方向性に向けたプレゼンテーションを行い、IESEとIPAの良好な協力関係を打ち出すことができた。

写真2　Fraunhofer welcomes Japanの登壇者（左端が和田）

④ CeBIT会場における日独協力の成果発表

CeBITの開催に合わせ、日独両国政府間で日独協力の方向性を定めた「ハノーバー宣言」が発出され（1）IoT/インダストリ4.0に関するサイバーセキュリティ、（2）国際標準化、（3）規制改革、（4）中小企業支援、（5）研究開発、（6）プラットフォーム、（7）デジタル人材育成、（8）自動車産業、（9）情報通信分野　の協力が合意されている。IPAとしては、この中の（1）IoT/インダストリ4.0に関するサイバーセキュリティ分野において、IoTセキュリティガイドライン策定に協力した経験を活かし、IoTセキュリティに関する国際標準化の議論に参画した。

3 本イベントを振り返って

今回は国際的な展示会にIPAが参加する貴重な機会であった。展示を通じた視察者とのやり取りで、国際的に見たIPA活動の独自性を評価するコメントも複数あり、日本の国や文化に関心を持っている個人もここ数年では増えつつあることも確認できた。今回の経験を、今後のWebを通じた情報発信や海外機関訪問時の事業PRなどに活かしていきたい。

米国 STAMP Workshop2017参加報告

SEC調査役 十山 圭介　SEC研究員 金子 朋子

2017年3月27日～30日までマサチューセッツ工科大学MIT（ボストン、米国）で開催されたSTAMP Workshop2017に参加し、STAMP※1の状況や動向について情報収集を行った。

1 ワークショップの特徴

2012年より始まった本ワークショップは今回で6回目となり、参加数は275～300名で昨年度より12%増えた。参加者は増え続けており、新規参加者は73%を占める。24カ国から参加があり、日本からは15名で、米国に次ぐ参加であった。20程度の産業分野から参加があり、航空宇宙・防衛が40%で最も多く、自動車が20%でそれに次ぐ。多様な企業、政府機関、大学と産官学が参加。Boeing社とEmbraer社はそれぞれMITと共同で分析を行っている。航空機や自動車の事故だけでなく、生産・建設現場での安全性や社会システムの安全性分析、人と自動化のインタラクションに関する分析が出てきている。

2 ワークショップの内容

本ワークショップの講演数は32件、ポスター発表は9件（表1）。ワークショップは、初日のチュートリアルと3日間のプレゼンテーションで構成される。チュートリアルでは基本編としてSTAMP、STPA、CAST※2のイントロダクション、および応用編としてそれぞれの演習を実施。また、拡大テーマとして、今年はヒューマンファクタと作業現場での安全についてのプレゼンテーションが行われた。

Birds of a Featherセッションも開催され、産業別の交流が図られた。当初航空宇宙・防衛と自動車の2グループが設定されていたが、会場で新たにヘルスケア、石油とガス、作業安全も追加され、活況を呈した。

表1　主な発表内容

タイプ	対象	業界
方式（手順）の提案	ハザードシナリオの生成方法	
	MIL-STD-882EとSTAMPとの対応付け	航空宇宙・防衛
	ISO26262との対応付け	自動運転車
	推進のポイント	産業界
STPAの適用	現場安全	航空
	空調制御	航空
	無人航空機システム	航空
	ロータークラフトの設計	航空
	脅威・エラー管理	航空
	フライトテストでの過度の自動化のシナリオ	航空
	電気自動車の高電圧対策	自動車
	レーン保持アシスト	自動車
	建設現場の安全性	建設
	飲用水供給システムのリスク分析	ライフライン
	蝸牛インプラントシステム	医療用ソフトウェア（CPS）
	セキュリティ	航空
		サプライチェーン
	IoTセキュリティ	
	軍事的意思の決定	防衛
	自動運転車	自動車
	海洋石油プラント	石油
STAMPの考えの適用	メンテナンス	鉄道
CASTの適用	ベビーフードによる事故分析	薬品
	自動運転での道路安全	自動車
	航空機事故	航空
	航空機事故	航空
	滑走路侵入	航空

3 注目すべき発表

コンチネンタル社は自動車の機能安全規格であるISO26262に適合した自動運転に対する安全アーキテクチャをSTPAに適用した事例を発表した。ルノー社は自動運転での道路安全に対し、ヒューマンエラーを分類し、CASTを拡張してTesla社の事故を解析した事例を発表した。ほかには、自動運転に伴い課題となるヒューマンエラーに対して、ドライバの注意散漫に着目したレーン保持アシスト、電気自動車の高電圧対策へのSTPAの適用、自動車関連の適用事例が多数発表され、迫力のあるセッションであった。またISO26262の次期バージョンにはSTAMPが入る見込みであり、今後日本でも急速な普及が予想される。

セキュリティに関しては、チュートリアルに加えて事例3件も発表された。Embraer社はフライトマネジメントシステムに対するSTPA-Sec※3を適用。米国の権威ある研究所であるLawrence Livermore National LabはIoTに対するSTPAを適用。IoTのもたらすセキュリティ上の脅威に対してSTPAを利用すると、これまで文献上で得られていた脆弱性が6～10件程度だった事例に対し200以上のセキュリティ上の脆弱性、ハザードなどを発見できた旨を発表した。

4 ポスターセッション

ポスターセッションは、2日目の17：30～20：00で実施された。IPA/SECからの発表タイトルは「A proposal for “hint words” to identify hazard causal factors for systems including human」で、ヒューマンファクタを考慮する際のハザードを導くモデルと「ヒントワード」を提案し、質問やコメントを多くいただいた。

5 日米欧の連携

9月には欧州、11月には日本でワークショップが開催される。STAMP WSは日米欧で3極体制を構築する予定であり、本ワークショップ参加を通じ交流がなされ、日本のIPA/SEC Webサイト（英文版） http://www.ipa.go.jp/english/sec/complex_systems/stamp_workshop.html より、MITと欧州（ESW）のWebサイトに相互リンクが貼られた。

写真1　MIT Nancy Leveson教授と共に

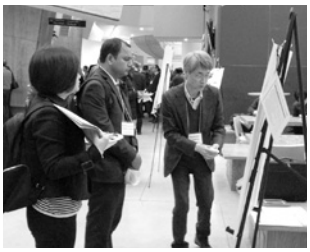


写真2　ポスターセッションでの光景

脚注

※1　STAMP（Systems-Theoretic Accident Model and Processes）とは、MITのNancy Leveson教授が提唱したシステム理論に基づく事故モデルとプロセス

※2　CAST（Causal Analysis based on STAMP）とは、STAMP理論に基づいて、事故後に、なぜ起きたかの検証と分析を行う手法

※3　STPA-Sec（System-Theoretic Process Analysis for Security）とは、STPAをセキュリティに適用した分析手法

編集後記

本号では2016年度のSEC成果報告を特集しています。IoTの時代に、社会全体を支える情報処理システムの信頼性向上を目指したSECの各種取り組み、詳細については本誌特集記事をご覧ください。取り組みの成果は、書籍やIPA/SECのWebサイトで公開しています。また、セミナーという形で直接皆様にお伝えする場も設けております。ぜひご活用ください。

さて、クラウドファーストという言葉があるように、今では当たり前のように利用している「クラウド」ですが、数年前はどうだったでしょうか。本当に使えるのか、セキュリティは大丈夫なのかと信じるための活動が必要だったことを懐かしく思い出しました。当初はクラウドに求めるものも、コンピューティングパワーでしたが、現在ではそれに加えてクラウド上で次々生み出されるサービスをいち早く利用できるといった側面も生まれてきたと思います。ITシステムが利便性や効率性を求めるツールの役割だけでなく、価値やビジネスを創出するツールとしての役割が期待されている現在、クラウドを駆使したイノベーションの創出が時流に合ったものなのだと感じています。（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など、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

夏の海　撮影：k.Hasegawa

SEC journal 第13巻 第１号（通巻52号） 2017年７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。

IoT 時代に活躍する**【組込みシステムの腕利きエンジニア】**を目指す！

国家試験 エンベデッドシステムスペシャリスト試験

高度な実践能力の証明に！

▶ 身近な場面を想定した出題を通して、最適な組込みシステム実現のために必要となる高度な実践能力（レベル 4）を問います。
（レベル 4 の定義）：専門分野において、自らのスキルの活用によって、独力で業務上の課題の発見と解決をリードするレベル。

技術要素	開発技術	管理技術
プロセッサ、メモリ、バス、計測・制御、リアルタイム OS、プラットフォーム、電気・電子回路、ネットワーク、セキュリティ	・要求分析の実行とレビュー ・設計の実行とレビュー ・テストの実行とレビュー	・開発環境マネジメント ・知財マネジメント ・構成管理、変更管理

▶ 近年の試験では、「無線通信ネットワークを使用した安全運転支援システム」、「3 次元複写機」、「通信機能をもつ電子血圧計を用いた健康管理システム」、「非接触型 IC カードを使用した入退場ゲートシステム」などのテーマを出題しました。

▶ 自動車、家電、モバイル機器などに搭載する組込みシステムや重要インフラの制御システムを、ハードウェアとソフトウェアを適切に組み合わせて構築し、求められる機能・性能・品質・セキュリティなどを実現できる組込みエンジニアを目指す方に最適です。

試験概要

【試験区分】エンベデッドシステムスペシャリスト試験（情報処理技術者試験 高度試験の 1 区分として実施）
【日　　時】年 1 回の実施（毎年 4 月第 3 日曜日）
【申込受付】毎年 1 月中旬から 2 月下旬（予定）まで WEB・郵送で申込み受付

詳しくは、Web ページをご覧ください。http://www.jitec.ipa.go.jp/index.html
試験概要の最新情報、過去問題、活用事例などをご紹介しています。

IPA Better Life with IT

SEC journal No.49
第13巻第１号（通巻52号）
2017年７月１日発行

ISSN 1349-8622