ソフトウェア工学分野における産学連携事業成果の紹介

～産業界での実用化の促進に向けて～

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

ソフトウェア工学分野の先導的研究支援事業（RISE事業）の成果について

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

ソフトウェアはあらゆる産業が供給する製品・サービスで新たな付加価値を生み出しており、見えないところで社会を支える重要な働きをしています。複雑化・大規模化・多様化するシステムへの対応や、それに伴うソフトウェアの高信頼化や開発プロセスの高度化を実現するため、ソフトウェア工学やシステム工学が果たす役割は、ますます高まっています。

このため、独立行政法人情報処理推進機構（IPA）／ソフトウェア高信頼化センター（SEC）では、わが国のソフトウェア工学・システム工学の振興、およびその成果の産業界への移転を通じてソフトウェアの信頼性向上に貢献することを目的に、2012年度より「ソフトウェア工学分野の先導的研究支援事業（RISE※事業）」を実施して参りました。

この度、この事業の成果を広く知っていただき、実用化の促進、さらなる研究開発への発展の足掛かりとするため、これまでの成果を簡潔に取りまとめた成果集を作成いたしました。

詳細に説明した成果報告書などにつきましては、IPAのWebページで公開していますので、詳細をご覧になりたい場合や、成果の利用などにご関心がありましたら、ぜひ、ご覧ください。

IPAといたしましては、本事業を通じて、ソフトウェア工学・システム工学の振興に寄与することを願っております。

※RISE: Research Initiative on Advanced Software Engineering

「ソフトウェア工学分野における産学連携事業成果の紹介」について

本冊子は、大学が産業界のニーズを受けて実施した成果を掲載しており、産業界の課題解決や競争力強化、品質の向上等を実現するために各成果を活用していただくことを目的としています。成果の企業内での活用や、製品化などに関心がありましたら、以下までお問い合わせください。

問い合わせ先：
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　企画グループ
〒113-6591　東京都文京区本駒込二丁目28番8号
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　MAIL：sec-pr@ipa.go.jp
http://www.ipa.go.jp/sec/rise/index.html

RISE事業と産学官の連携

ソフトウェア工学研究推進委員会　委員長
一般社団法人情報サービス産業協会　会長　**横塚 裕志**

RISE事業が始まった2012年という年は、前年に私たちにとって未曾有の大災害である東日本大震災を経験し、本格的な復興が始まった頃でした。一方、情報システムに目を向けると、レンタルサーバサービスのデータ喪失、金融システムのシステム障害などの大規模な障害が発生し、情報システムが社会に与える影響の大きさについて改めて認識された時期でもありました。

このように、情報システムが、大規模化・複雑化する中、信頼性・生産性を飛躍的に向上させるためには、従来とは異なるアプローチによるソフトウェアやシステムを開発・運用して行く必要があります。

そのためには、ソフトウェア工学、システム工学に基づく設計開発や運用保守などが重要ですが、このような先進的な取り組みを行うためには、産業界だけでは難しく、大学などの知恵と、それに対する国の支援が必要です。

RISE事業は、ソフトウェア工学、システム工学分野における産業界のニーズに対して、その実現に向けた大学などによる取り組みを独立行政法人情報処理推進機構が支援し、成果を産業界が活用するという、産学官が一体となって推進してきた事業です。

これら成果は、産業界の課題解決やシステムの信頼性向上につながるものとして、産業界にとって大変有益であることはもちろん、これからの産学連携の一助となることが期待されます。

最後になりましたが、本事業にかかわられた多くの方々に感謝いたします。

CONTENTS

2013年度 ソフトウェア高信頼化

抽象化に基づいたUML設計検証支援ツールの開発

岡山県立大学
情報工学部 情報システム工学科 准教授 横川 智教

1 背景と目的

組込みソフトウェアは、その重要性から信頼性について極めて高い品質基準が要求される。一方で、複雑化・大規模化による開発コストの増加が課題となっている。

組込みソフトウェアでは１件の不具合が社会に及ぼす影響が甚大であるため、その検証には多くのリソースが割かれることになる。しかし、大規模化／複雑化が顕著な組込みソフトウェアにおいては、システムの取り得る状態数は、人手でテストを行う限界を大きく超えている。

この課題を解決するための有効な手段と考えられているのが、モデル検査技術である。モデル検査は、設計の無矛盾や仕様との整合性の検証を完全に自動化することが可能であり、設計工程における不具合の検出に費やすコストを抑えることができる。また、モデルに対する網羅的検証を行うことにより、モデル化した振る舞いが求める特性を満たすか否かについて、完全な保証を得ることができる。

しかし、モデル検査技術を組込みソフトウェアの設計検証に導入するには、種々の問題点が指摘されており、中でも２つの大きな問題点がある。

１つ目の問題点は、モデル作成の困難さである。

モデル検査による設計検証では、ソフトウェアの設計文書を、モデル検査ツール固有のモデル化言語で記述する必要がある。しかし、モデル記述言語の文法や意味論は、組込みソフトウェアの設計技術者が通常使用しているUML（Unified Modeling Language）などの設計文書の記述方式とは、大きな隔たりがある。そのため、設計文書からモデルを生成するには、専門的な知識やノウハウが必要となる。

２つ目の問題点は、状態爆発の危険性である。

前述したとおり、組込みソフトウェアの設計記述は大規模かつ複雑である。モデル検査ではモデルの状態空間の網羅的探索を行う。そのため、設計記述をすべてモデル化しようとした場合、探索する状態数が爆発的に増加することにより、現実的な時間内で検証が完了しないおそれがある。

これらの問題はいずれも設計記述から検証モデルを作成する段階で発生する。本研究は、これらの問題を解決すべく、モデル検査による設計検証を行う際、検証モデルの作成をツールによって支援することで、モデル検査の導入における障壁の低減を目的としている。

2 概要

本研究では、設計記述から検証モデルを自動的に作成するための検証支援ツールを開発した（図１）。

対象とする設計記法は、組込みソフトウェアの開発において広く利用されているUMLとした。また、モデル検査ツールは、モデル記述言語であるSMV（Symbolic Model Verifier）言語の表現力、および検証速度に優れ

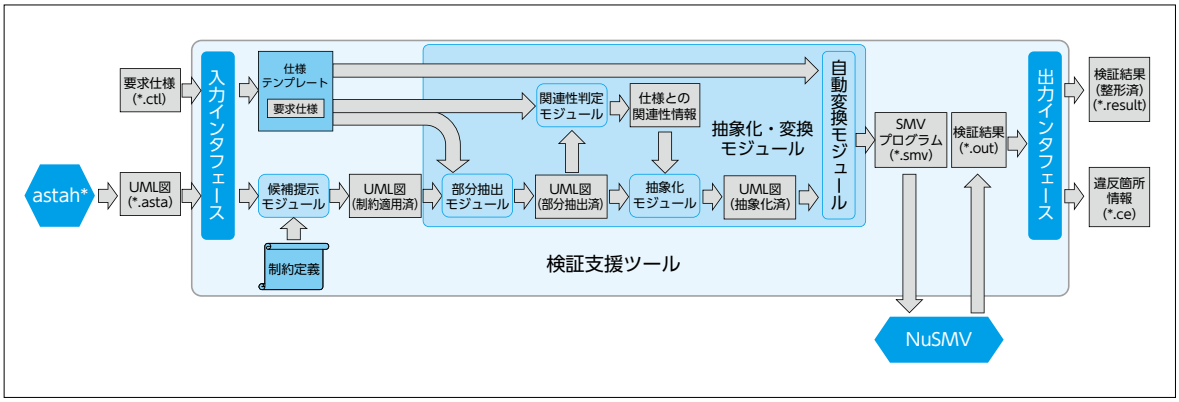


図１　検証支援ツールの概要

たNuSMVを採用した。開発した検証支援ツールは以下の３つの機能を持つ。

機能１　SMV言語への自動変換を目的とした、UML図の抽象化
機能２　モデルサイズ削減を目的とした、UML図の抽出、分割、および抽象化
機能３　UML図からSMV言語への自動変換

機能１：SMV言語への自動変換を目的とした、UML図の抽象化

UMLの記法の中には複雑な振る舞いを記述するためのものが存在する。これらをモデル記述言語で記述すると記述量は著しく大きくなるため、自動変換のコストも大きくなる。しかし、ソフトウェア設計者がこのようなUMLの記法を厳密な意味論に基づいて利用しているとは必ずしもいえず、設計の解釈に曖昧性が含まれる可能性がある。

そこで、対象とするUMLの記法の中で、曖昧な解釈を引き起こす可能性の高いものについては、記述方式に制約を課すことで、曖昧性の混入を回避する。さらに、制約に沿った記述を支援するため、記述の制約に沿わない箇所については、どのように記述を修正すればよいかという候補の提示や、部分的な修正の自動化を行う。

機能２：モデルサイズ削減を目的とした、UML図の抽出、分割、および抽象化

大規模かつ複雑なUML図をモデル化した場合、検証モデルの状態数が爆発的に増加し、網羅的探索による検証は困難となる。

そこで、検査対象システムから検証すべき特性に関連した情報のみを抽出し、モデルの抽象化を行うことで、検証コストを削減する。

検証特性に基づいた抽出・分割、および抽象化を行うためには、システムが満たすべき仕様を適切な形で検証特性としてツールに与える必要がある。

そこで、システムが満たすべき仕様についてテンプレートを用意して入力を定型化することにより、設計者が検証したい特性を容易に入力できるようにする。さらに、仕様に関連するUML図の要素を明示的に示すようテンプレートを設計することで、仕様に基づくUML図の抽出・分割、および抽象化を効率的に行うことを可能とする。

機能３：UML図からSMV言語への自動変換

モデル検査を行うためには、検査対象システムをモデル記述言語で記述する必要がある。しかし、NuSMVのモデル記述言語であるSMV言語の文法や意味論は、検査対象となるUMLとは大きく隔たりがある。

そこで、UML図からその振る舞いを表現するSMV言語によるモデルへの自動変換を行う。

3　産業界で研究成果が適用される場面と期待される効果

この検証支援ツールの実現により、組込みソフトウェア開発における設計検証に、モデル検査を導入する際の諸問題の解決に寄与し、開発コストの削減およびソフトウェアの信頼性向上が期待される。さらに、産業界へのモデル検査技術の普及も期待される。

本ツールは、対象となるUMLの表現能力やインタフェース面に関して制約はある。しかし、与えられたUMLによる設計記述からモデル検査ツールであるNuSMVの入力モデルへの変換をほぼ自動化し、さらにツールによって得られた結果を整形し、わかりやすく表示することが可能である。

また、本研究では検査したい特性を記述するための仕様入力テンプレートを開発している。テンプレートに従ってUML図内の要素を指定すれば、その要素の振る舞いを、NuSMVで検査するための検査式を自動的に生成することが可能である。これらの機能により、モデル検査の専門的知識をまったく持たない技術者であっても、NuSMVを用いた設計検証ができる。

モデル検査では、設計検証の対象となるシステムの振る舞いについて、単にその組み合わせのすべてを網羅的にチェックできるようにモデル化を行っただけでは多くの場合、状態爆発が発生する。このような場合、検証に要する時間や計算資源の規模が、実用的な範囲を越えてしまう。そのため、検査する性質に関連する部分のみの抽出や、不要な部分の抽象化などの対策により、検証モデルの規模を抑える必要が生じる。本ツールでは、関連部分の抽出や不要な部分の抽象化を自動的に行ったうえでモデル化を行うため、状態爆発を回避ができる。

このように、本ツールを利用することで、組込みソフトウェアの設計検証へとモデル検査を導入する上での課題への対応が可能となり、導入への障壁が大きく低減されると期待される。

■「モデル検査導入支援ツール」の紹介
URL http://circuit.cse.oka-pu.ac.jp/tool.html

2012年度 ソフトウェア高信頼化

モデルを含む設計成果物の集積とその活用方法に関する研究

九州大学
システム情報科学研究院 情報知能工学部門 准教授 久住 憲嗣

1 背景と目的

近年、ソフトウェア開発は大規模化／複雑化し、また、諸外国との激しい競争にさらされている。そのため、ソフトウェアの工数削減や設計品質の向上を目的とし、UMLなどのモデルを援用して開発することが、ソフトウェア開発の大きな流れである。モデリング技術を利用することで、正確な設計が可能となり、設計が可視化でき、開発者間でのコミュニケーションが円滑になる。さらにモデルを中心に据えた自動化技術であるモデル駆動開発（Model-Driven Development；MDD）を用いると、開発早期でのシミュレーションや検証を行えるようになるなど、多くのメリットがある。従来のような自然言語とプログラミング言語を中心とした開発から、モデルを用いた開発への移行は不可避である。そのためモデリングやMDDに関する研究やケーススタディが盛んに行われている。

しかし、研究を実証的に遂行する上では、必要不可欠なデータが圧倒的に不足している。その理由は、研究などに利用できるモデルを中心として開発された、第三者が利用可能な設計成果物がほとんどなく、さらに、モデルのバグ情報や編集履歴まで含めて利用できる成果物は皆無なためである。

第三者が利用可能なモデルを含んだ設計成果物がないことは、以下の２点に起因すると考えられる。

- モデルの編集履歴を収集できるツールがない
- 第三者に利用可能なデータを収集する場がない

また、これらの問題に伴って、モデルを含む設計成果物を大局的にとらえるための枠組みが不足していることも考えられる。

そこで本研究では、これらの問題を解決することで、モデルを用いた開発においても実証的に研究を進められるようにすべく、以下のようなことを行った（図１）。

(1) モデルを集積するためのモデリングツールを開発
(2) 講義やProject-Based Learning（PBL）などでモデルを集積
(3) 既存のメトリクスを利用して予備評価

この結果、モデルを対象とした実証的研究への利用、モデルとその編集履歴などの継続的集積、集積したデータを利用するための環境を整備することが可能となる。

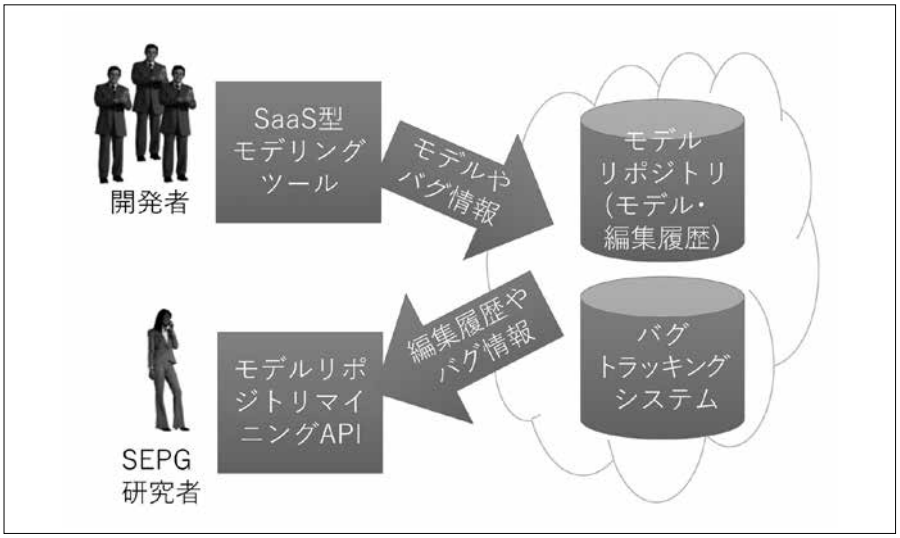


図１　本研究で構築する環境

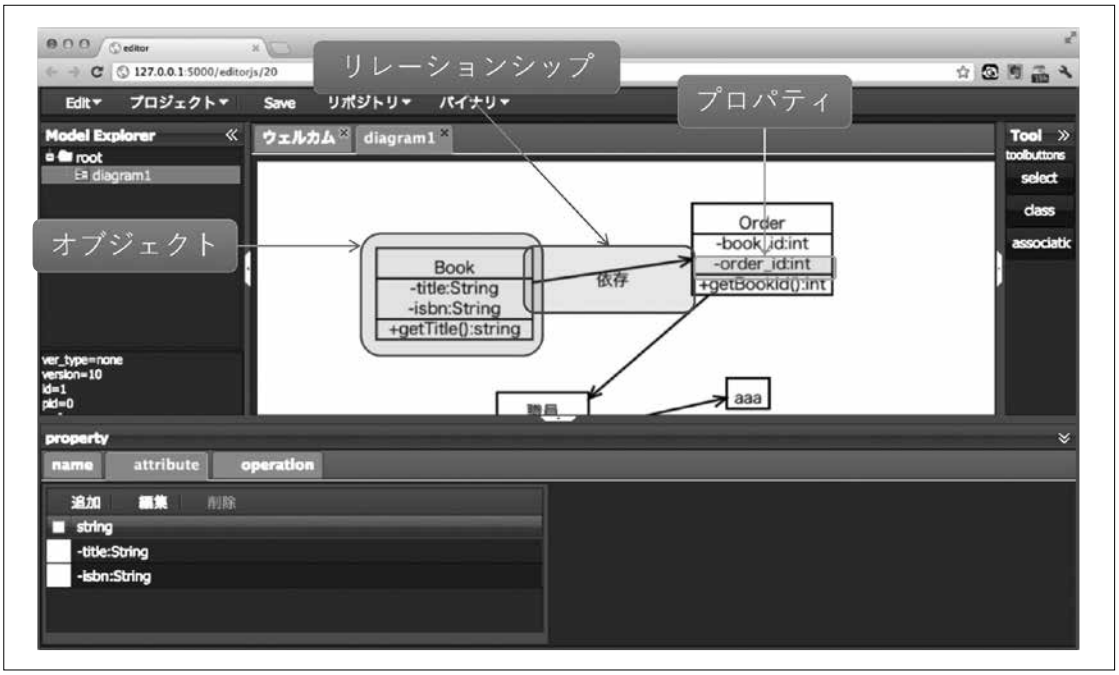


図２　モデリングツール（cloaca）の画面イメージ

2 概要

(1) モデルなどを集積するためのモデリングツールの開発

① モデルとその編集履歴を集積できるように既存ツールを改変

既存のモデリングツールは、モデルの履歴を保存する機能がなかった（図２）。また、既存のリポジトリシステムにモデルを単純に登録するだけでは、モデルの履歴に対して横断的に検索をかけることが困難である。

そこで本研究では、SaaS型のドメイン特化モデリング言語ツールであるcloocaを対象として、図３のようにモデルとその編集履歴を集積し、分散協調開発を行えるようにするためのモデルリポジトリを構築した。本研究の遂行により、モデルとその編集履歴、およびバグ情報などを、ツールの支援のもとに収集できるようにした。

さらに、バグ情報とリポジトリ上の変更を対応づけるために、バグトラッキングシステムとの連携機能を開発した（図４～図５）。

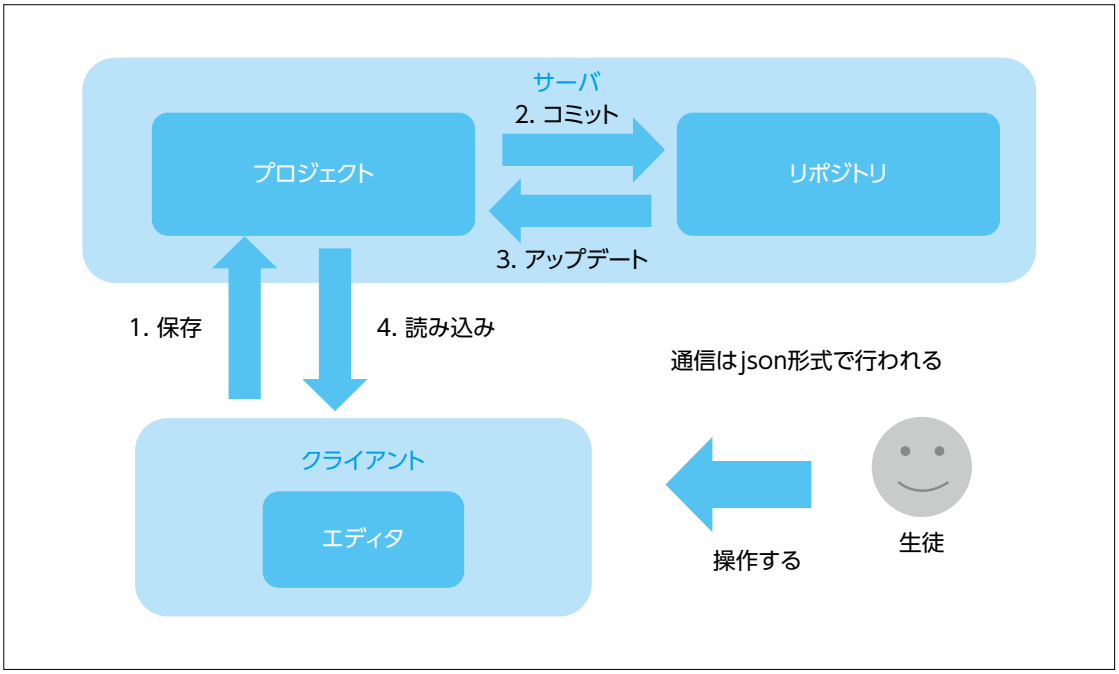


図３　リポジトリシステム概要

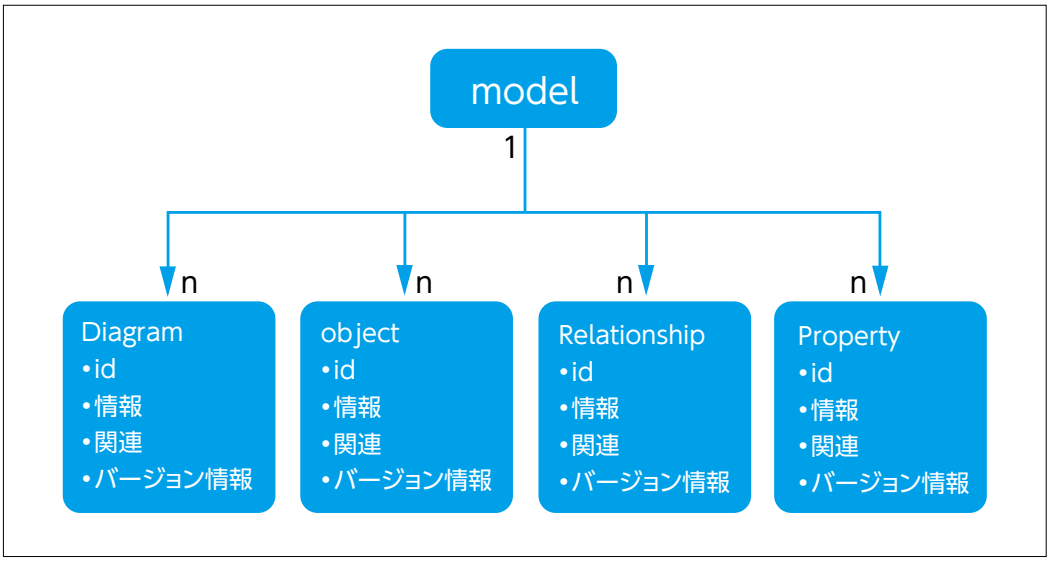


図4　モデルの構造

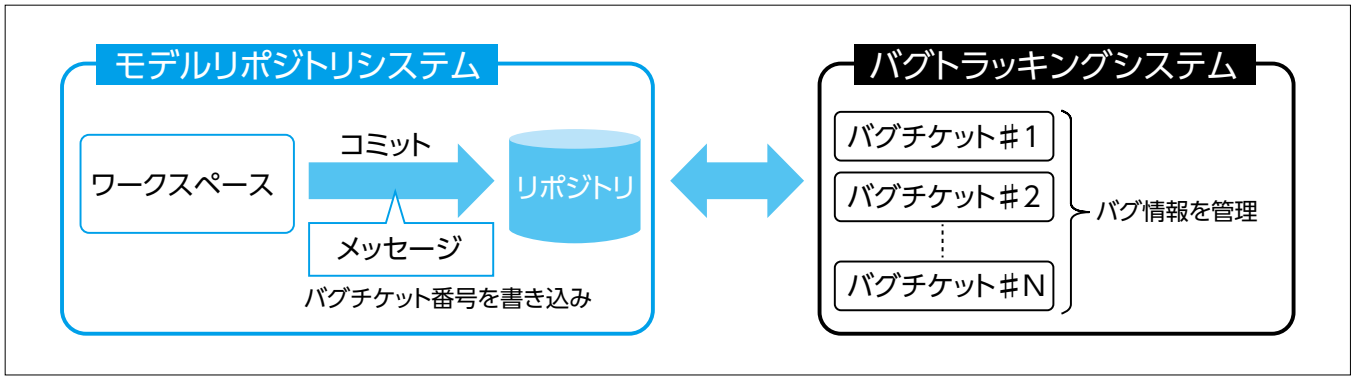


図5　バグトラッキングシステムとの連携

② モデルの編集履歴を検索・利用するためのAPIの構築

①で集積した設計成果物に対するメトリクスを、研究者や教育者の要望に応じて計算／提示することを支援するためのAPIを構築した。 計算したいメトリクスは、研究内容、授業支援内容などにより異なるため、多くのメトリクスを計算できるように開発するのではなく、成長させやすいアーキテクチャとした（図6）。

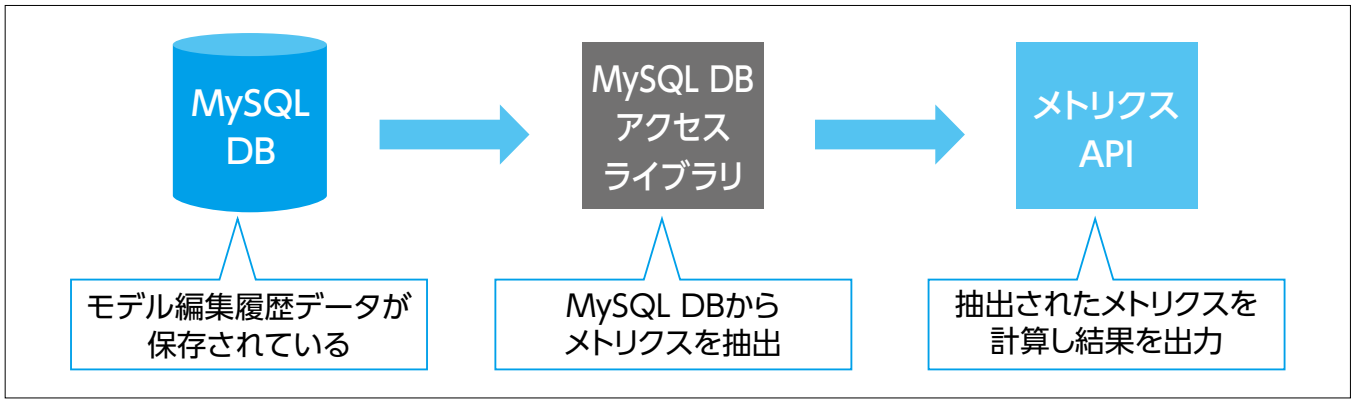


図6　メトリクスAPIアーキテクチャ

(2) 講義、PBLなどでツールを利用して、モデルとその編集履歴を集積

講義やPBLにおいて、受講者に開発したツールを利用させ、モデルとその編集履歴の集積を行った。これによってモデルとその編集履歴を収集し、教育や研究上の知見を得るための材料とした。

講義は２種類実施し、一つはUMLなどのモデルを初めて学ぶ初学者向けの講義、もう一つはUMLを用いたモデル駆動開発を学ぶモデル駆動開発特論における講義である。両者ともにモデリングを必要とするような演習課題を与え、その演習課題を受講生が実施する過程のモデルを収集した。両講義において、架空の運輸会社の自動搬送ロボットの開発業務を請け負うこととする共通の総合演習を実施した（図7）。

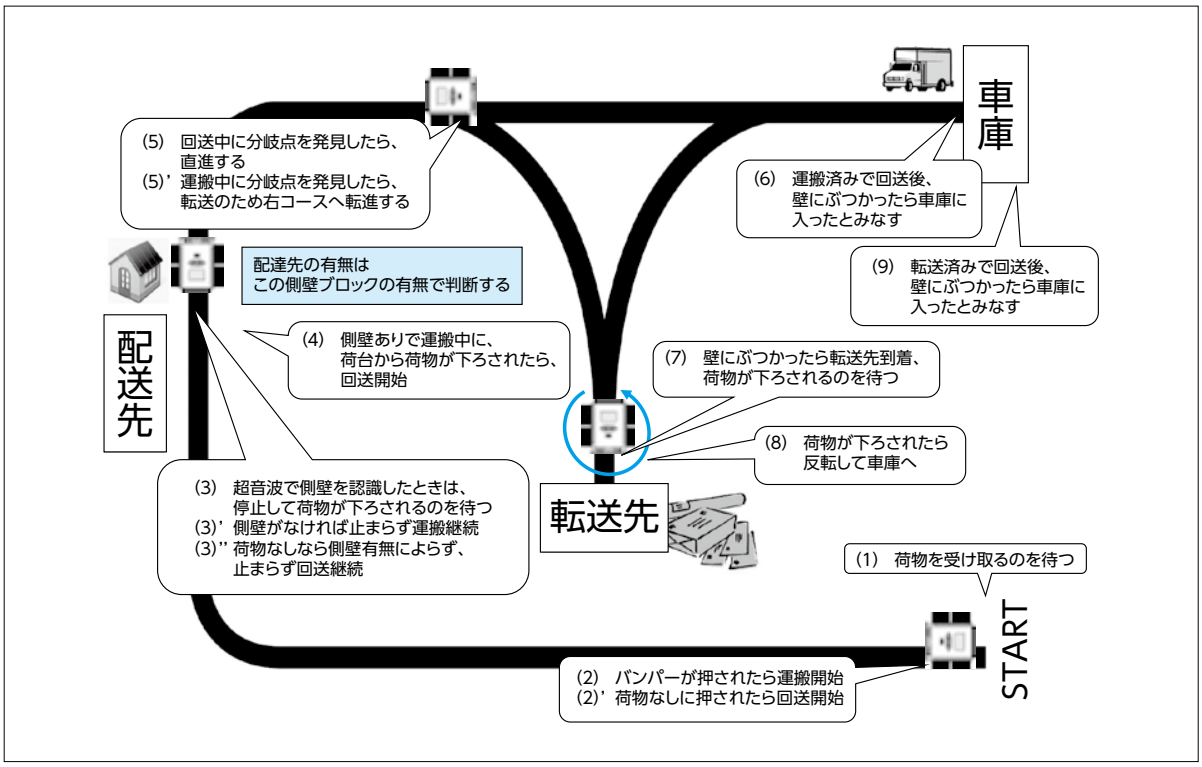


図７　総合演習課題コース

(3) 既存のコードを対象としたメトリクスがモデルでも有効に働くかどうかを評価

(2)において収集したデータを対象として、分析を行った。分析は定量／定性の両面から実施した。その結果、MDDをしたチームとしなかったチームにおけるモデルの品質傾向が明らかとなり、講義に関する改善点が明らかとなった。具体的な改善点は、「UMLモデリングの教育効果を上げるためには、単純にMDDツールを用いた開発をさせるのではなく、コードの自動生成や動作確認の回数を制限した演習を実施する必要がある」「レビューを適切なタイミングで実施すべきである」などである。

3 産業界で研究成果が適用される場面と期待される効果

モデルを対象としたリポジトリマイニングの推進

本研究で開発したツールによって、対象となるデータが少ないためにあまり実施されてきてこなかったモデルと、その編集履歴を対象としたデータの分析、さらにリポジトリマイニングが促進される。その結果、ソースコード対象で実施されているような、バグ密度を推定することによるテストやレビューを効率化する手法が、モデルを中心とした開発においても適用できるようになる。特に従来のソースコード対象の手法では不可能であった上流のモデルにおいても、同様の手法が適用できるようになる。

SaaS型開発環境による開発に関するデータの利活用基盤

本研究で開発したツールの利用が広まれば、従来であれば収集することのできなかったデータを組織横断的に収集することができるようになり、利活用できる。従来の開発環境はクライアント上で動作し、そこで設計成果物を作成したうえで、必要に応じて設計成果物をリポジトリ上に登録していた。また、このリポジトリは通常は組織を超えて共有されない。ところが、SaaS型開発環境では、第三者機関がツールを提供することになる。そのため、組織を超えた設計成果物の収集ができる。このデータの利用には匿名化、抽象化するなどの細心の注意が必要ではあるが、従来は困難であった組織を超えたデータの利活用が可能になる。

2015年度 ソフトウェア高信頼化

要求定義の高品質化のための要求仕様の整合性の検証知識の形式知化と一貫性検証支援ツールの開発

工学院大学
情報学部 コンピュータ科学科 准教授 位野木 万里

1 背景と目的

ソフトウェア開発において、真の顧客要求に応えるためには、上流工程からの品質の作り込みは必至であり、対象ソフトウェアの機能や非機能の範囲を定める要求定義工程は、極めて重要である。近年、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、現状では、各検証は各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。

開発パラダイムと要求仕様の関係において、ウォーターフォール型、アジャイル型、ユーザエクスペリエンス型のいずれの開発でも、要求仕様の高品質化は重要である。ウォーターフォール型の問題点と同様、要求仕様の検証は、属人的な方法で行われる傾向にあり、要求仕様、特に自然言語で記述されたシナリオを対象とした一定の検証手法が必要とされていると考えられる。

本研究では、要求仕様の構成要素であるシナリオを取り上げ、要求仕様の品質特性である「一貫性」に着目し、シナリオの一貫性検証支援ツールを実現する。シナリオの整合性の検証知識とシナリオの一貫性検証支援ツールを提供することで、業界全体で要求定義のノウハウを共有し、わが国における要求定義技術のレベルアップを目指す。そして、魅力あるソフトウェア製品の要求定義の効率化、高品質化により、ソフトウェア開発の国際競争力向上に貢献する。

なお、本研究は、一般社団法人情報サービス産業協会が主催した技術シンポジウムSPES2014のSPES事例研究（経験報告）にて東芝ソリューション株式会社による「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～」で提案された技術を拡張し、適用範囲を向上させたものである。

2 概要

本研究では、ベテラン技術者が経験的に得たシナリオの整合性の検証知識を形式知化し、それら知識に基づくシナリオの一貫性検証支援ツールを実現する。開発するシナリオの一貫性検証支援ツールは、シナリオ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、要求仕様書中の記述と整合していることを検証する。

(1) シナリオの一貫性検証知識の形式知化（形式知化）

仕様書の一貫性を検証するためのノウハウを形式知化した。具体的には先行研究の成果と、今回特に分析した２点の仕様書に基づいて、仕様書の分析、および４社５部門の有識者にインタビューを行い、仕様書の検証ノウハウを分析し、約40個のルールとして定義した（図１）。ルールはツールに組み込むためのクラス／モジュールに対応づけて実装し、複合語の特定、アクター、画面、データ、振舞い用語の識別、定義漏れ、表記ゆれ、用語定義完全一致、NGワードの特定などの種類に分類した。設計要素のアクター、データ、画面、振る舞いを識別するためのキーワードに相当する辞書を、エンタープライズ系の仕様書を対象として定義した（図２）。

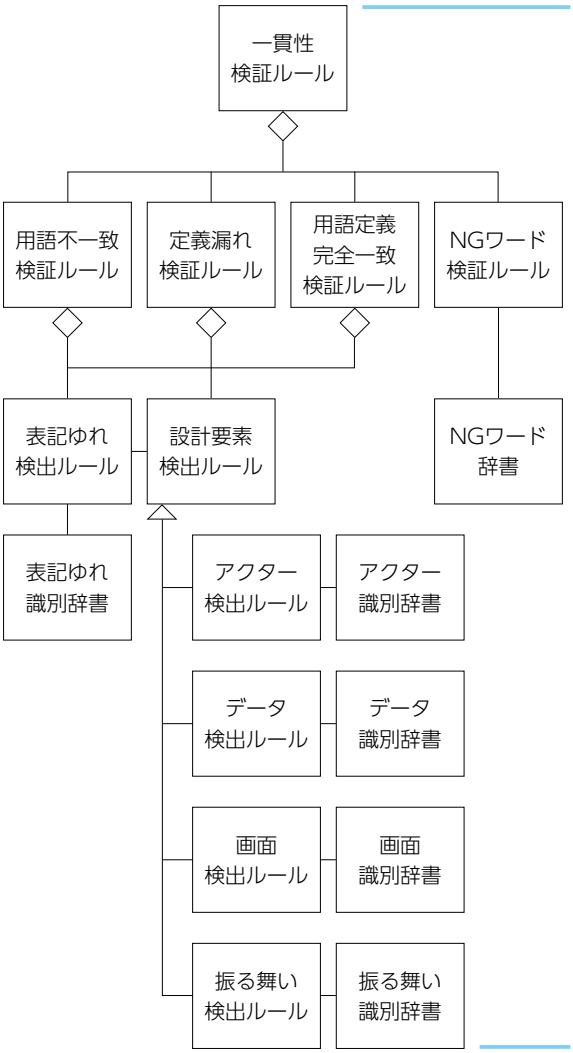
(2) シナリオの一貫性検証の支援ツール

検証ルールと辞書に基づき、シナリオの一貫性検証を支援するツールを開発した。開発したツールの機能は表１のとおり。

次にツールのアーキテクチャを示す（図３）。検証ルール、辞書は、組織や対象市場により異なると考えられるため、拡張可能性を考慮し、ツールのアーキテクチャを定義した。本ツールでは、F1～F4までの検証機能と、F5のシナリオ自動補正からなる機能と、検証ノウハウにあたる検証ルールと辞書、検証エンジンは分離した構造とした。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

れたシナリオを解釈する形態素解析エンジンで構成する。

シナリオにはソフトウェアの要求仕様に固有の表現が含まれる。そのため、OSSの形態素解析エンジン（※1 MeCab）をそのまま用いただけでは、要求仕様の構成要素を特定することは困難である。そのため、複合語や係り受けの関係性を辞書などで抽出可能なようにした。



No.	ルール分類	ルール分類説明	ルール数
1	MeCab 結果変換	形態素解析エンジンMeCabの特性を踏まえて、MeCab出力結果を変換するためのルール。これにより、仕様書中の複合語を適切に抽出する	14
2	アクター識別	仕様書からアクター用語を識別するためのルール	5
3	データ識別	仕様書からデータ用語を識別するためのルール	5
4	画面識別	仕様書から画面用語を識別するためのルール	5
5	振る舞い識別	仕様書から振る舞い用語を識別するためのルール	5
6	定義漏れ検証	No.1 ～ No.5のルール分類で定義したルールを踏まえて、仕様書内の定義漏れを指摘するためのルール	1
7	用語 不一致検証	No.1 ～ No.5のルール分類で定義したルールを踏まえて、仕様書内の用語不一致（表記ゆれ）を指摘するためのルール	3
8	用語 完全一致検証	No.1 ～ No.5のルール分類で定義したルールを踏まえて、仕様書内の用語完全一致ではない箇所を指摘するためのルール	1
9	NGワード 検証	No.1 ～ No.5のルール分類で定義したルールを踏まえて、仕様書中のNGワードを指摘するためのルール	1
10	その他	上記分類以外で、設計要素の識別や検証のためのルール	2

（例）【ルール：AR005】
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みなす
（AR005適用例）シナリオ中から以下をアクターとして抽出し、表記ゆれの検出に用いる
（表記ゆれ例）ユーザ、ユーザ（会員）、ユーザ（非会員）

図1　検証ルールの構造とルールの定義結果

エンタープライズ系要求仕様書を想定し設計要素識別辞書を定義

アクター
- 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
- ○○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アクターとみなす
- シナリオ中に，「顧客（会員）」，「顧客（非会員）」などと記述されていたら，アクターを記述する辞書の「顧客」と「会員」の複合語として，この2つのアクターをアクター用語とみなす

データ
- 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
- ○○情報，○○データなどの表現や辞書に定義された用語の複合名詞をデータとみなす

画面
- 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
- ○○画面，○○スクリーンなどの表現や辞書に定義された用語の複合名詞を画面とみなす

振る舞い
- する，実施，実行，管理
- ○○する，○○実施などの表現や辞書に定義された用語の複合語を振る舞いとみなす

図2　設計要素を識別する辞書

表1　シナリオの一貫性検証の支援ツールの機能一覧

No.	機能名	ID	処理概要
1	定義漏れ検証	F1	検証対象の仕様（シナリオ）および、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。
2	用語 不一致検証	F2	検証対象の仕様（シナリオ）および、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、対応する定義表と別の用語表現（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。
3	用語定義 完全一致検証	F3	検証対象の仕様（シナリオ）および、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、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指摘する。
4	NGワード検証	F4	検証対象の仕様（シナリオ）および、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。
5	シナリオ自動補正	F5	検証対象の仕様（シナリオ）および、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。

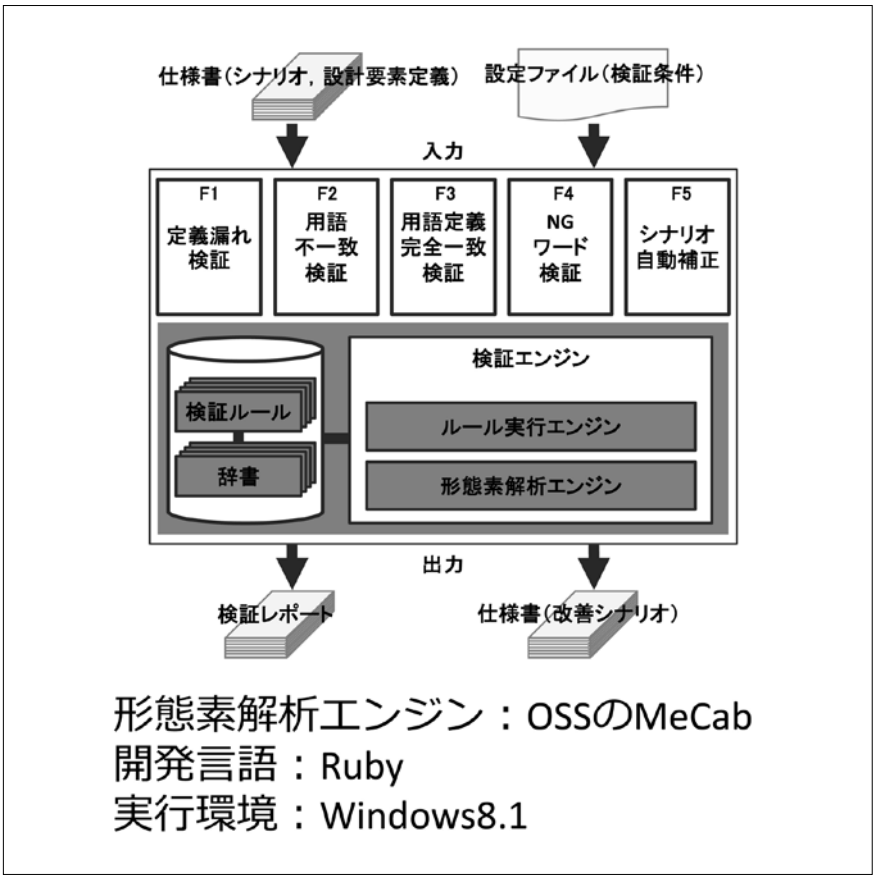


図3　シナリオの一貫性検証支援ツール開発アーキテクチャ

(3) シナリオの一貫性検証の支援ツールの評価

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。検証は実際の案件での要求仕様書／調達仕様書を用いてルール分析、テストデータ作成、ツールテストを実施した。

その結果、表記ゆれの特定と、統一化のための指針の抽出は100％には至らなかったものの、平均して、再現率90％以上、適合率84％以上の結果を得た。

ツールの再現率、適合率について、一部のアクターが特定できず、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。対策としては、アクター用語の識別辞書に登録することで、再現率を向上させることができると考えられる。

データに関しては、形態素解析エンジンMeCabの特性により、一部の単語が分かち書きされるため、データ用語として特定できなかった。現状では、ツール側での改善は困難であるため、ユーザマニュアルにおいて、識別困難であることなどを促す必要がある。

3 産業界で研究成果が適用される場面と期待される効果

本研究の成果は、主にエンタープライズ系システム／サービスの企画立案、要求定義工程で利用するドキュメントで役に立つと考えられる。具体的には要求仕様書、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。それぞれの活用シーンは表２のとおり。

表２　成果の活用シーン

- 初級技術者が要求仕様書や提案書をセルフチェックする
- 担当者が作成した仕様書を管理者が品質チェックする
- プロジェクトメンバー全員で検証レポートを使い要求仕様書をレビューする
- 組織で蓄積した検証ルールや辞書を用いて知識継承や人材育成に取り組む
- 業務マニュアルからアクター用語抽出し、ステークホルダ分析等に利用する
- 要求仕様書と基本設計書のそれぞれの検証レポートを比較し、設計要素の不統一箇所や設計漏れのチェックを行う
- プロダクトライン型開発の各モデル間の検証レポートを作成し、共通部分が継承され、可変部分が考慮されているかを比較する

■「シナリオの一貫性検証支援ツール」の紹介
URL http://www.ns.kogakuin.ac.jp/~wwa1076

※１ MeCab: Yet Another Part-of-Speech and Morphological Analyzer
URL http://taku910.github.io/mecab/#download

2014年度 ソフトウェア高信頼化

保守プロセスにおけるモデル検査技術の開発現場への適用に関する研究

芝浦工業大学
システム理工学部 電子情報システム学科 教授 松浦 佐江子

1 背景と目的

形式手法の1つであるモデル検査とは、「有限の状態空間を網羅的に調べて、与えられた性質が成り立つか否かを調べる技術」である。検査を行うモデル検査器と状態空間を探索するシミュレータから構成され、性質が成り立たない場合には反例を提示するモデル検査ツールが提供されている。ここで、利用者は検査したい対象のシステムモデルと検査式の両方を定義することで、モデル検査器を用いて、その性質が成り立つか否かを検査することができ、反例により、問題点を発見する。モデル検査は、レビューやテストとは異なり、システムの振る舞いがある性質を満たすか否かを網羅的な探索により自動的に検証する方法であり、検証の手段として期待できる。

われわれはモデル検査に着目し、2012年度ソフトウェア工学分野の先導的研究支援事業「要件定義プロセスと保守プロセスにおけるモデル検査技術の開発現場への適用に関する研究」において、ソフトウェア開発工程で考慮されるソフトウェアの満たすべき性質を「業務セオリー」と呼び、要件定義プロセスと保守プロセスの観点から、現場の開発者がモデル検査ツールを用いて様々なシステムの性質の検証を行なう方法を研究した。そして、モデル検査ツールの入力となるシステムモデルと、検査式を現場の開発者が要求分析モデルやソースコードから生成する方法の見通しを得た。

本研究では、現場の開発者がモデル検査ツールを直接操作せずに、モデル検査の恩恵を受けられる方法を開発するために、以下の研究課題を設定した。

- 開発者が理解しやすい形式で「検査したい性質（機能要求・非機能要求）」を「検査対象のふるまいモデル（ソースコード）」と関連付けて定義することで、検査式を自動生成し、到達可能性や安全性の検査を行えるようにする。
- 「検査対象のふるまいモデル」を、状態爆発が起こらないようにシステムモデルへ変換する。検査したい性質に基づく抽象化方法を定義し、作業容易化のための自動化の方法を検討する。
- 検査から得られた反例から問題点を発見する過程を支援する。

上記の作業を、現場の開発者が適切かつ容易に実施できるよう支援する方法を検討する。同時に、現場の開発者が容易に操作・理解できる変換支援ならびに反例解析支援ツールを開発する。

2 概要

(1) 仕様定義に基づく検査モデル・検査式の生成方法の定義

開発者が理解しやすい形式で「検査したい性質（機能要求・非機能要求）」を「検査対象のふるまいモデル（ソースコード）」と関連付けて定義することで、検査式を自動生成し、到達可能性や安全性の検査を行えるようにする。関連付けには、テストによく用いられるデシジョンテーブルで定義された仕様や非機能要求の1つである、セキュリティ要件定義を用いて検討する。

本研究では、本学で実際に稼働している学習支援システムLUMINOUSを、セキュリティ要件の検証対象とする。本システムは、学生が開発したシステムを基に再開発し、機能追加や権限の追加により拡張するという開発形態を取ってきた。仕様書としては、利用シナリオや、一部にはUMLモデルが存在する。

以下の手順で、システムに対する仕様ならびにセキュリティ要件をソースコードと結びつけながら定義し、システムモデル／仕様モデル／検査式を生成し、検査を実施する。

1） LUMINOUSの要求分析モデル（ユースケース図、各ユースケースのアクティビティ図、クラス図）を要求分析手法に従い、既存システムを操作しながら、次のように定義する。まず、基本フローを事前条件・

事後条件とともに定義する。基本フローに対し、例外フローは処理フローの行先ごとにまとめてガードに記述する。オブジェクトノードのクラス名はソースコードから抽出したエンティティモデルのクラス名を参照し、名前はアクションの目的語と一致させる。事前条件・事後条件・ガードに登場する場合はそれとも一致させる。さらに、画面構成に相当するクラスを定義する。ここで､クラス名は各ユースケースの作業状態を表すものとする。項目名は画面上のラベルと一致させる。また、ユースケースの使用するエンティティデータとそのセキュリティ属性をクラス図から特定する。そのセキュリティ属性の満たすべき性質はステートマシン図で定義する。

2) ソースコードの静的解析によりUI要素とそのソースコード表現（UIコード）の対応表を抽出する。ソースコードの静的解析により抽出する情報は、entityクラス、entityクラスの属性、UIクラス、UIクラスのメソッド、ボタンやチェックボックスなどのUI要素とそのソースコード表現の５種類である。

3) ユースケースの事前条件に従い組み合わされた検査シナリオとそれに対応するセキュリティ機能方針表のルールを､次のように定義する。情報セキュリティの国際評価基準（ISO/IEC15408）であるCommon Criteria（CC）に定義されたセキュリティ機能コンポーネントモデルと、要求分析モデルとの対応により、対象システムのサブジェクト（アクター）とオブジェクトに対してセキュリティ属性を定義する。要求分析モデルのアクティビティ図より、すべてのオブジェクトを抜き出し、それぞれを対象とするアクションを抽出する。ここで、各ユースケースにおいて対象とするオブジェクトの資産としての利用環境における価値を考慮する。つまり、資産とそれを何から守るべきかを検討し、オブジェクトにセキュリティ属性を付加して、守るべきルールをセキュリティ機能方針として定義する。

図１は、検査シナリオに基づく検査モデルの全体像を表している。検査シナリオは複数のユースケースに対応するメソッドのモデルを呼び出す。各メソッドが実行される間に、セキュリティ属性の更新メソッドが呼び出され、それに同期して、属性の仕様である状態遷移モデルが変化する。これにより、ユースケースが終了した時点で、セキュリティ属性の状態が想定の値であるかという上述の検査式を検査することで、ソースコードがルールを満たしているかを検査できる。

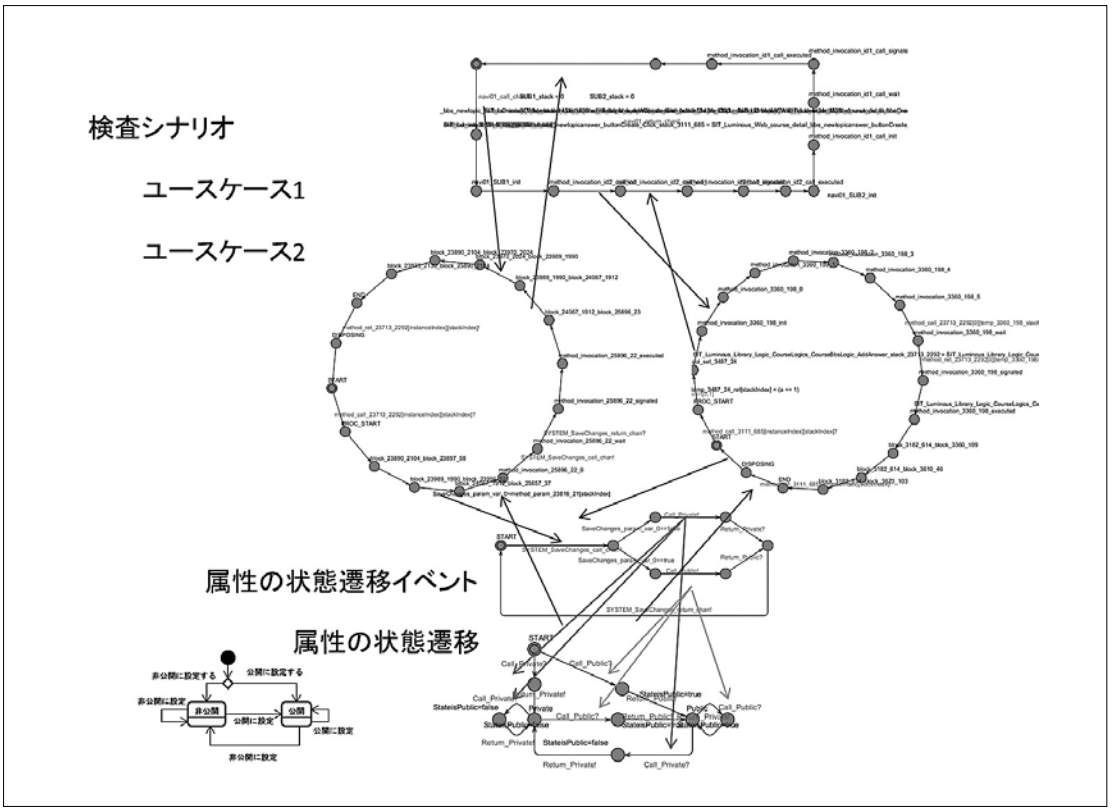


図１　検査モデル

(2) 変換支援ツールの設計と開発

(1)の検査モデル・検査式の方法に基づき、ソースコードをモデル検査用のシステムモデルに段階的に変換する方法を定義し、支援ツールを開発する。支援ツールはソフトウェアの統合開発環境の１つである、eclipseのプラグインとして開発する。モデル検査ツールは、UPPAALを用いる。図２は変換支援ツールのユースケース図である。「仕様モデル」は、UML記述によるユースケースに基づいて定義するセキュリティ要件定義テーブル、およびセキュリティ属性のモデルから生成されるUPPAALモデルである。

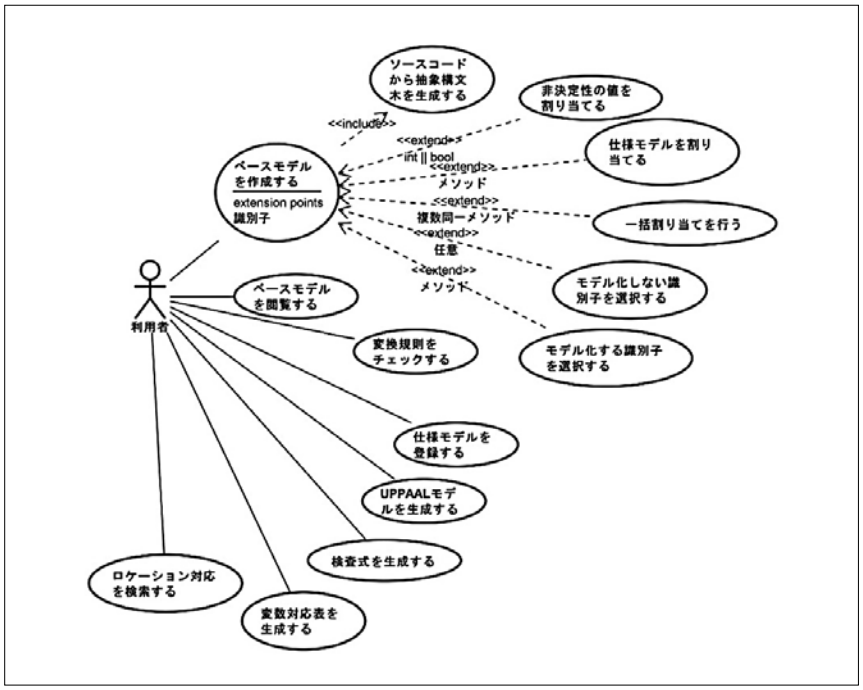


図２　変換支援ツールのユースケース図

(3) セキュリティ要件の定義と検証

事例として、本学で稼働中の学習支援システムLUMINOUSを対象として、セキュリティ要件を満たしているかを検証する。

システムの操作を通じて、LUMINOUSの要求分析モデル(ユースケース図、各ユースケースのアクティビティ図、クラス図)を定義する。この際、各ユースケースの使用するエンティティデータが上述のどの資産であるかを特定し、セキュリティ機能方針のもととなるデータを抽出する(表1)。サブジェクトとオブジェクトに対して、それぞれ必要なセキュリティ属性を定義する。さらに、セキュリティ属性の満たすべき性質をステートマシン図で定義する。

Common Criteria(CC)のセキュリティコンポーネントを参考にして、設定したセキュリティ属性を用いてルールを定義する。この段階では、ソースコードから抽出したエンティティクラスのモデル図により、エンティティクラスとそのセキュリティ属性は仕様とソースコードが結びついた状態となる。ユースケース上のUI要素とUIコードを対応付けることにより、対象ソースコードのセキュリティ要件を検証することができるようになる。

変換支援ツールSource2UPPAALを用いて、以下のように検査モデルを作成し、検査を行う。Source2UPPAALは、2012年度ソフトウェア工学分野の先導的研究支援事業にて開発した、Javaのソースコードから、値の割り当て、メソッドの割り当て、ユーザ定義モデルに基づく抽象化により、UPPAALのシステムモデルを段階的に生成する機能をもつツールである。検査シナリオを「UI要素とUIコードの対応表」を用いて、ソースコードからモデル化すべきメソッドおよびその定義場所の情報を抽出する。この情報から、ユースケースに対応する内部メソッドの特定を行い、「UIコード特定表」を作成する。「UIコード特定表」をもとに、変換支援ツールSource2UPPAALを用いて、UPPAALへの変換対象となる検査シナリオを作成する。

表１　セキュリティ機能方針表

サブジェクト		オブジェクト		操作		ルール		
アクタ	セキュリティ属性	クラス	セキュリティ属性	ユースケース	アクション	FDP_ACF.1	FMT_MSA.3	FMT_MSA.1
学生	役割(学籍番号)	話題	公開／非公開	質問を投稿する	話題を生成する		ルールB1	
		添付ファイル	公開／非公開	話題を閲覧する(学生)	添付ファイルをダウンロードする	ルールA		
				質問を投稿する	添付ファイルを生成する		ルールB2	
教員	役割(教員)	話題	公開／非公開	質問に回答する	回答を追加して話題を更新する			
					話題の公開／非公開を公開に変更する			ルールC1
					話題の公開／非公開を非公開に変更する			ルールD1
		添付ファイル	公開／非公開	質問に回答する	添付ファイルを生成する		ルールB3	
					添付ファイルの公開／非公開を公開に変更する			ルールC2
					添付ファイルの公開／非公開を非公開に変更する			ルールD2

(4) 反例解析方法の検討

UPPAALの検査で検査式に対する反例が得られた場合には、「想定される状態」になった場合と、「想定されない状態」になった場合の両者の反例をグラフ化し、それらを比較して差異ができる部分を表示することにより反例解析を支援する。

検査において「想定されない状態」になるかで反例がでた場合は、それと対になる「想定される状態」になるかの検査を行う。この場合、単純に「想定される状態」の検査を行っても、まったく関係ない状態遷移になる可能性が高い。そのため「想定されない状態」の反例からシステムの仕様で特定できる識別子に着目し、その識別子に関係するUPPAALの変数を検査式に取り込む。システムのふるまいを特定できる状態をそろえ、類似した状態遷移になるため比較が可能になり、図３の左側に示すように「想定される状態」「想定されない状態」の分岐ポイントを見つけることができる。

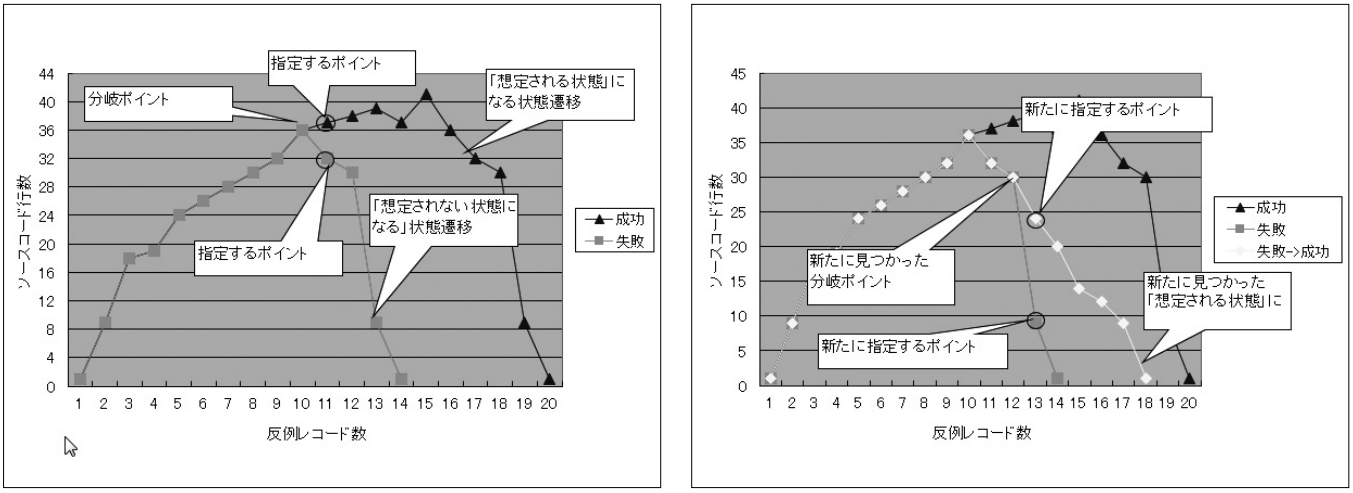


図３　反例解析

しかし、その分岐後で最終的なシステムのふるまいの状態が特定できるとは限らない。分岐ポイント以降の状態遷移を規定するため、分岐ポイントより先の通過指定ポイントを指定した式を追加した検査式を検査する。これを繰り返すことにより、検査結果の精度が高められる。図３右図のように新たな分岐ポイントが見つかる可能性もある。最終的なシステの振る舞いの状態が変化しない分岐ポイントが確定するまで検査を行う。

3 産業界で研究成果が適用される場面と期待される効果

ここでは、形式手法のスペシャリストではない現場の開発者が、モデル検査の恩恵を受けられる方法を研究した。具体的には、ユースケースにより定義した機能要件に対して、「セキュリティ要件」という非機能要件の１つを、セキュリティ機能方針表により定義し、検査を実施するという方法を提案できた。本方法の産業界における活用の場面としては、以下の場面が想定できる。

- システムのマイグレーション時に、本研究で示した仕様定義方法により、仕様の十分でないシステムの仕様を整理し、旧システムおよび新システムのソースコードを検証する。これにより、旧システムが満たす仕様を新システムが満たすことを保障できる。
- 最終的なリリース判断において、テストでは不十分であると思われる特定の仕様を満たしていることを低コストで確認できる。

これらの過程において、仕様が定義されている場合でも、仕様定義が不十分な場合でも、副次的に、検査したいことの観点で仕様を再整理することができる。これにより、仕様定義の教育が実施できると同時に、手戻りの少ない開発に貢献することが期待される。

形式仕様とテスト生成の部分的・段階的な活用
～探索を通したコード中心インクリメンタル型開発の支援

情報・システム研究機構
国立情報学研究所 コンテンツ科学研究系 准教授 石川 冬樹

1 背景と目的

近年、高い注目を集めているアジャイル開発、形式手法、品質保証テストの３つの技術分野は、実際に導入をするにあたって、それぞれの特徴に起因する課題が存在する。加えて、それぞれの技術分野に閉じた議論がなされがちで、相互補完や総合的な施策につながりにくいという課題もある。

第一に、形式仕様記述などの形式手法や、モデルベーステストなど品質保証テストにおけるテスト自動生成において、仕様やテスト設計の厳密な記述を与えることが難しい。これは工数に関する費用対効果の懸念もあれば、実感しにくい宣言的、抽象的な記述において、意図に合った記述を行うことが論理的に難しいという側面もある。

第二にその一方で、強い確信を持ちやすいテストケース（具体例）を開発の拠り所として用いるテスト駆動開発においては、妥当なテストスイート（テストケースの集合）を定めることの難しさがある。これについては、論拠となるテスト設計からテストケースを導出したり、テストケースをテスト設計と照らして確認したり、論拠となるテスト設計について議論をしたりするための支援がないということである。

これらの課題に対しては、テスト駆動開発、形式仕様記述、テスト自動生成の原則・考え方を総合的に活用する分野横断的／融合的なアプローチが求められる。具体的には適切なツールを開発し、適用する。これにより３つの技術分野の導入のための相互補完や総合的な施策につなげることができると期待される。

2 概要

- コードスケルトン（変数定義やメソッドシグネチャ定義）上に書き加えた断片的な仕様やテスト設計を基に、テストケースを探索、提示するツールを構築する。
- 併せて仕様、テスト設計、テストケースを混在させた形で与えることができる言語を提供する。
- ツールは、テスト駆動開発、形式仕様記述、テスト自動生成の原則・考え方を総合的に活用することができる（図1）。

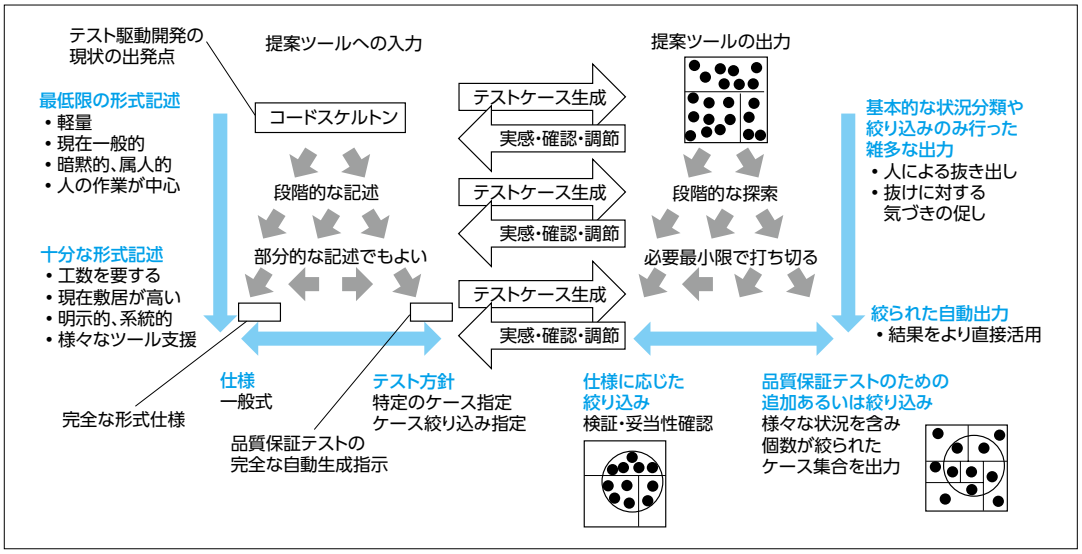


図１　提供言語・ツールの特徴

性質による記述と例示による記述の融合

従来形式手法、形式仕様記述においては、性質（Property）、すなわち宣言的な命題により仕様記述を行ってきた。これはテスト設計（テストスイートに対する仕様）においても同様である。一方でテスト駆動開発からの発展として、例示による仕様（Specification by Example）というキーワー

ドも用いられている。提供言語は、これらの双方を使い分けることを可能とする。さらに、双方を両方与え、互いの確認に用いることも可能になっている。

仕様、テスト設計、テストケース（具体例）の混在した記述

より具体的な実現として、仕様、テスト設計、テストケースを混在させ、互いに互いの確認に用いたり、相補的に用いたりすることができるようになっている。

早いフィードバックを与えるツール

与えた制約式に対し、その式の意味を具体例に示す。これにより、何か記述を行った際に、様々な記述をそこから積み上げる前に、行った記述の意味をすぐに確認することが可能である。すなわち、意味を逐次確認しながら、段階的に形式的な記述を構築していくことができる。これはテスト駆動開発において重要な原則である。また、ある一通りの式を与えないとフィードバックを得られないということはない。これは仕様、あるいはテスト設計について部分的にのみ形式的な記述を与えた場合でもツールを活用できることを意味する。

仕様を得るタスクおよびテストケースを得るタスク双方への活用

提供言語およびツールは、テストケース生成のためのものであるが、とらえ方によっては「仕様とテストを行き来する」ためのものである。すなわち、必ずしも生成するテストケースが成果物であるとは限らず、その生成を通して仕様記述やテスト設計記述の妥当性確認を行っているとも言える。このように提供言語およびツールは、仕様記述やテスト設計を成果物とするタスクにおいても、テストケースを成果物とするタスクにおいても利用できる。

例と反例双方の確認

与えた式を満たす場合だけでなく、満たさない場合も確認できるようにする。特に部分的な条件（弱い条件）を与えた場合には、条件を満たす例は意図に合わないものが多くなる。その場合でも条件を満たさない反例を確認することで、どういう可能性を排除したのかを確認できる。これに限らず、条件が強すぎて意図しない場合が排除されているとき、それを反例の中から見つける可能性があるなど、条件を満たさない反例の確認には意義があると考えられる。

命題に対するテストケース・テスト駆動開発

テスト駆動開発では、まだ記述していない機能や観点に対してテストケース（具体例）を与えたとき、失敗する（Red/Fail）ことが開発サイクルの出発点となる。これはテスト駆動開発・テストファーストかどうかに限らないが、プログラムと異なる仕様記述の場合、まだ記述していないことについては「どういう出力でも受け入れる」と、とらえることができる。すなわち、非決定的な記述が可能である。これに対して、既存の仕様記述では不十分であることを明示するためには、あるテストケースの値が受け入れられるかだけではなく、その入力に対して唯一の受け入れられる出力を定めていることをアサーションとしたい。提供言語はテストケースが決定的であることを宣言することができ、この要求に対応している。

3 産業界で研究成果が適用される場面と期待される効果

提案ツールを実際に利用するシナリオとしては、以下の３つを想定する。一つの対象部品に対し、一つの利用を行うことも、複数の利用を行うことも考えられる。いずれにおいても、記述を段階的に追加しながら、随時確認をすることとなる。

(1) アジャイル開発において、開発者がより所となるテスト設定を行う際に、提案ツールを用いてテストケースを生成させる。これにより、その設定の妥当性確認や、想定以外の例示による漏れの探索を行う。

(2) 従来日本語コメントなどで記述していた、あるいは暗黙であった制約記述を提供言語で与える。提案ツールを用いることにより、与えた制約の妥当性を確認するとともに、十分性についても把握することができる。例えば、明記した制約だけでは弱いが、明記していない点は他の開発者が予測できるであろうことの確認などが挙げられる。ここで、この記述と確認を段階的に繰り返すことにより、ドキュメントとして、あるいは形式検証などに十分な制約記述を得ることもできる。

(3) 従来、人手でテスト技法を適用し、結果のみを記述していた品質保証テスト設計工程において、提供言語でテスト設計を与える。そして提案ツールを用いることにより、テスト設計の結果に流用できる部分的な情報を生成させることができる。加えて、様々なテストケースを生成させることにより、テスト設計の妥当性を確認するとともに、テスト設計の内容自体や記述ドキュメントにおける漏れの気づきも促される。ここで、この記述と確認を段階的に繰り返すことにより、生成されたテストケースをそのままテスト設計とする（十分なテスト設計記述を与え、テストケースは自動生成するようにする）こともできる。

次世代ソフトウェア信頼性評価技術の開発とその実装

広島大学
大学院 工学研究院 情報部門 教授 土肥 正

1 背景と目的

ソフトウェアの高信頼化は近年の大きな課題である。それと同時に、プロセスと製品の信頼性を「定量化」し、プロジェクト管理技術および開発手法に「フィードバック」する取り組みは、多様化するソフトウェアに対して高信頼性を保証するための重要な技術である。

ソフトウェア信頼性モデルは、ソフトウェアテストで発見されるフォールトの計数過程を統計的に推論することで、発見フォールト数の飽和状態を監視し、ソフトウェア信頼度などの定量的評価尺度を評価するのに用いられてきた。このようなフォールトの計数データだけを用いたモデルは取り扱いが非常に簡便であり、いくつかの民間企業において利用されている。

しかし、従来のモデルでは、テスト労力やソフトウェア種別などの情報を明示的に利用せず、フォールト計数データのみを利用している。そのため、それらの要因と定量化された信頼性の因果関係がブラックボックス化している。これによって、「得られた数値の妥当性の検証が難しい」「管理技術や開発手法への明示的なフィードバックが難しい」という重大な欠陥がある。

一方、現実的にはテスト網羅度や欠陥密度のような原因と結果が明確な可観測情報に基づいて、主観的な信頼性を算出する手法を利用することも多い。網羅度や欠陥密度は、信頼性評価尺度のひとつではあるが、当該ソフトウェアの信頼性を正確に表現しているものではない。つまり、網羅度や欠陥密度を導出したとしても、確率・統計理論を用いないかぎり、網羅度・欠陥密度と定量的ソフトウェア信頼性の因果関係において、妥当性を検証することが難しい。

従来のソフトウェアの定量的信頼性評価では、テスト工程で発見されたフォールトの個数情報のみから「残っているバグの個数」などを推定する試みがなされている。これは、「発見されたフォールト個数」という非常に簡単な情報であるため、ソフトウェアの種類にかかわらず、あらゆるソフトウェア開発に対して適用されてきた。

本研究では、このような抽象化・汎用化に対する従来のソフトウェア信頼度成長モデルの利点を活かしながら、より詳細なデータ（統計量）を扱えるモデルへと拡張を行った。開発現場で獲得し得る情報水準に応じ、信頼性評価の方法を分類する次世代のソフトウェア信頼性評価技術の体系化を目指した。

2 概要

本研究では、各々の現場で獲得可能な情報水準に基づいた信頼性評価体系を構築した。具体的には、①ソフトウェアメトリクス、②テストケース情報（テスト入力、テスト実行パスなど）、③ソースコード情報（制御依存グラフ、データ依存グラフなど）の３種類のデータと、テスト工程で観測されるフォールト計数データに対する関係をモデル化した。これによって、各々のデータを有機的に利用するソフトウェア信頼性モデル、およびモデルに基づいた各種信頼性評価尺度の導出に関する方法論を開発した。

本研究で開発された信頼性評価技術は、Excelのインタフェースを利用して手軽に信頼性評価ができる「MSRATS（Metrics-based Software Reliability Assessment Tool on Spreadsheet）」と、統計解析ソフトである「R」上で駆動する「Rsrat（Software Reliability Assessment Tool on R）」の2種類を、信頼性評価ツールとして実装した。

(1) 信頼性評価モデルの構築・推定アルゴリズムの開発・モデルの有効性検証

①ソフトウェアメトリクス、②テストケース情報、③ソースコード情報に基づいたソフトウェア信頼性評価のための確率モデルの解析を行い、現在までに行われてきた研究と新しい成果を組み合わせることで、最良モデル（最も良いカーネルの構造など）の選定を行った。

また、各モデルで必要とされる情報（データ）から未知パラメータを効率よく推定するためのアルゴリズムを開

発し、その安定性解析を実施した。

さらに、実際の事例に基づいた実証分析を通じて、提案技術の有効性を定量的に評価するとともに、一般に公開されているベンチマークデータや開発データを再調査し、実証分析に利用可能なデータを分類・整理した。

(2)信頼性評価ツールMSRATSおよびRsratの開発

(1)MSRATSの概要

MSRATSはC#を用いたExcel AddInとして開発し、ユーザの使いやすさに主眼をおいた設計となっている。

MSRATSの特徴

- Excelのスプレッドシートから時刻データ、あるいは個数データの入力
- 11種類の典型的なNHPPモデル、動的メトリクスを扱うロジスティック回帰によるソフトウェア信頼性モデル、静的メトリクスを扱うポアソン回帰によるソフトウェア信頼性モデル、一般化線形ソフトウェア信頼性モデルのパラメータ推定および信頼性尺度計算の自動化
- Excelのグラフ描画機能を利用した、信頼度関数などのグラフ描画

モデルに基づいて算出可能なソフトウェア信頼性評価尺度

- **予測総バグ数**：単一のソフトウェアモジュール(プログラム単位orシステム単位)に内在する総バグ数の予測値
- **予測残存バグ数**：単一のソフトウェアモジュール(プログラム単位orシステム単位)に現時点で残存しているバグ数の予測
- **ソフトウェア信頼度関数**：一定期間中にバグが発見されない確率
- FFP (fault-free probability)：現在のモジュールにバグがない確率
- 各種MTTF (mean time to failure)、累積MTTF、瞬間MTTF、条件付きMTTF：バグ発見までの平均時間
- Median、Betenlife：信頼度が0. 5, 0. 1になるまでの時間

フォームの例 (MSRATSフォーム)

MSRATSフォームはポアソン回帰によるソフトウェア信頼性モデルを取り扱う (図1)。手順とフォームの例は以下の通り。

1 各モジュールとそれらの静的なメトリクスをMetricsテキストボックスで指定する。
2 指定後に中央のリストにモジュールとメトリクス情報が表示される。
3 リストでモジュールをダブルクリックすることで、NHPPモデルを扱うSRATS、あるいはロジスティック回帰によるソフトウェア信頼性モデルを取り扱うLSRATSフォームが表示される。ここではモジュール毎のバグデータを登録し、モジュール毎のモデルパラメータ推定を行うことができる。
4 すべてのモジュールでモデルを指定した後、MSRATS上のEstimateボタンをクリックし、モジュールの静的メトリクスを考慮した推定を行う。

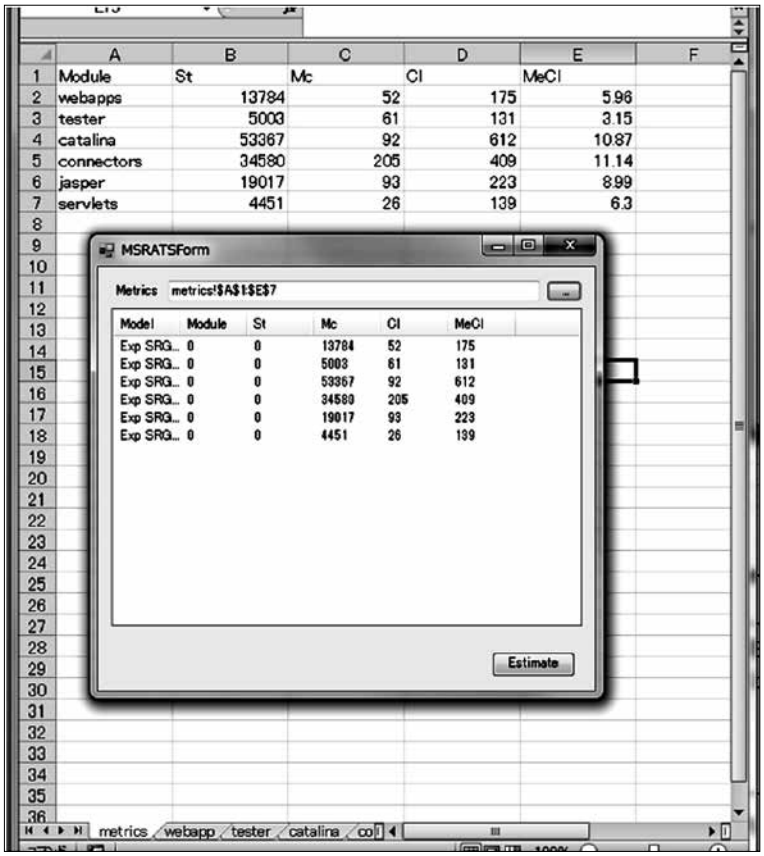


図1　MSRATSフォーム

(2) Rsratの概要

Rsratは、R言語を用いた統計処理ソフトウェアRのパッケージとして開発した。ユーザがR言語を用いて拡張することができるため、実験的なデータ解析や大量のデータ解析を行う研究者に向いている。

Rsratの特徴

- Rのデータフレーム形式を利用した時刻データ、あるいは個数データの入力
- 11種類の典型的なNHPPモデル、動的メトリクスを扱うロジスティック回帰によるソフトウェア信頼性モデル、静的メトリクスを扱うポアソン回帰によるソフトウェア信頼性モデル、一般化線形ソフトウェア信頼性モデルのパラメータ推定および信頼性尺度計算の自動化
- R言語による拡張が可能

モデルに基づいて算出可能なソフトウェア信頼性評価尺度

算出されるソフトウェア信頼性評価尺度、および扱えるモデルはMSRATSと同じである。また、カーネル法の適用に関してもカーネル値を要因として入力し、正則化最尤法を適用する。

3 産業界で研究成果が適用される場面と期待される効果

ソフトウェア信頼性評価は、テストの進捗状況把握と出荷判定の際に行われるべき活動である。レビューなどの静的テストやテストケースを投入しながらの動的テストの現場では、どれくらいのレビューやテストケース準備を行えばよいかの理論的な指針がほとんどない。そこで納期とコストを考慮し、テスト計画に合致するようテスト作業を進めることが一般的である。還元すれば、定量的な信頼性評価は実質的には行われておらず、フォールトが出尽くしたかどうかを判定する成長曲線の飽和状態だけを観測することで、テスト進捗状況把握や出荷判定を行っているケースがほとんどであろう。

本研究で開発したツールを活用することで、テスト作業者がテストの進捗状況を手軽に把握することができるようになる。また開発管理の立場から、製品の出荷判定の根拠となる説明資料として、定量的信頼性評価結果を用いることが考えられる。同時に、ソフトウェアの定量的信頼性を確保することは、ユーザや市場に対するソフトウェア製品の品質に関する説明責任を果たす意味において、極めて重要な社会的意義を持つものと考える。

■「ソフトウェア信頼性評価支援ツール」の紹介

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

データマイニング手法を応用した定性的信頼性／安全性解析支援ツールの開発

広島大学
大学院 工学研究院 情報部門 教授 土肥 正

1 背景と目的

ソフトウェア内の欠陥がシステムの安全性を脅かす事例は枚挙に暇がない。ソフトウェアの欠陥が発生する要因はいくつか存在するが、設計時における障害事象の「考慮漏れによる欠陥」は、フィールドにおいて最も深刻な障害を引き起こす可能性のある欠陥であることが多い。同時にこの種の欠陥は、取り除くのが最も困難な欠陥の一つとして知られている。

考慮漏れによる欠陥を防止するためには、すべての事象を網羅的に分析することが重要となる。特に、安全性を阻害する故障はこれまでに未経験であることが多く、事象の発生を予め想定することが難しい。安全性分析は「想定外を想定する」という、難しい課題に挑戦しなければならない。具体的な安全性分析手法として、FTA（Fault Tree Analysis）、FMEA（Failure Mode and Effect Analysis）、HAZOP（Hazard and Operability Studies）などがある。これらは、信頼性や安全性を数値的に定量化するのではなく、信頼性や安全性を損なう要因を定性的に分析することから、定性的信頼性／安全性分析手法と呼ばれる。

実務レベルで、これらの定性的信頼性／安全性分析を行う場合、２つの大きな問題がある。１つ目は、分析結果が分析者の経験値に依存するという問題である。２つ目は、網羅的な分析に要するコストの問題である。

本研究では、このような問題点を部分的に解決もしくは緩和する目的で、定性的信頼性／安全性分析を支援する新しい技術の開発と、開発現場において分析を支援するツールの開発を行う。

2 概要

本研究では、過去の情報（設計情報や障害事例など）をデータベース上に蓄積し、FTA、FMEA、HAZOPに現れる故障モードやガイドワードなどのキーワードと、対象とするシステムの設計情報（UML / SysML）から関連する障害シナリオを抽出したうえで、それらを重要度に従ってランキングするための学習アルゴリズムを開発した。さらに、対象システムの設計情報と過去の情報（安全性分析結果）から対象システムの設計上で重要なコンポーネント（プロセス）を推定し、優先順位をつけて分析者に提示する支援ツールを開発した（図１）。

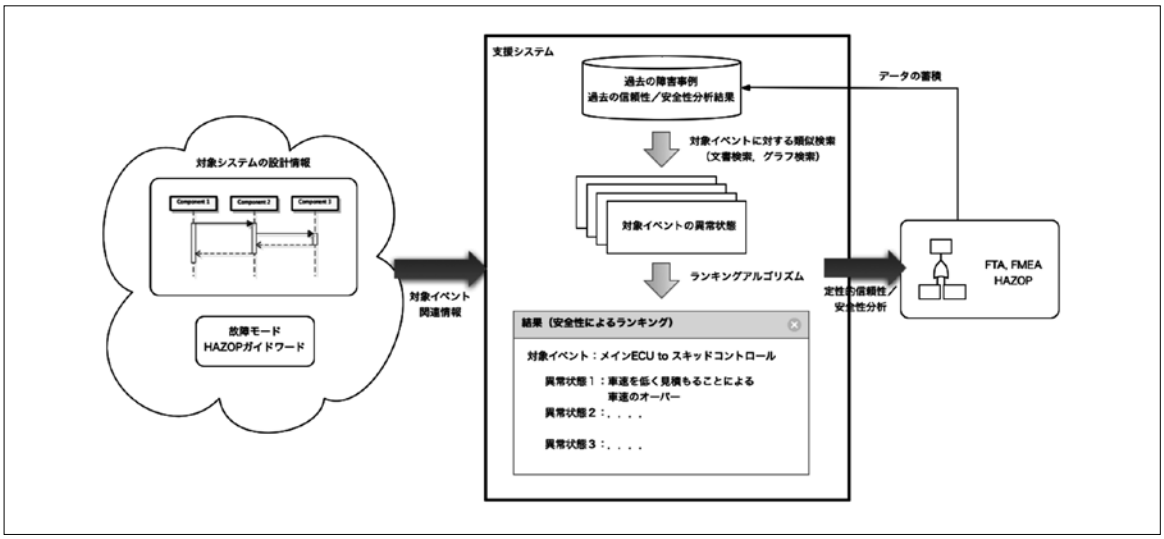


図１　支援システムの概要

支援ツールは過去のHAZOP分析結果と現在の設計の類似度を評価し、その類似度に基づいた重要度スコアリングを行う。そのため、過去の分析結果を学習するフェーズと現在の設計を評価（スコアリング）するフェーズに分けられる。

過去の分析結果を学習するフェーズでは、一般的な文章および定性的信頼性／安全性分析に関連した文章から単語（タグ）を抽出およびベクトル化し、入力された過去の設計書および分析例の分解と登録されている単語による特徴付け（タグ付け）を行う（図2）。

評価フェーズでは、入力された設計書の分解およびタグ付けし、タグに基づいた過去の分析例との類似検索をしたうえで、類似度と重要度のスコアリングを行う（図3）。

開発したツールは、次ページのような機能を有する。

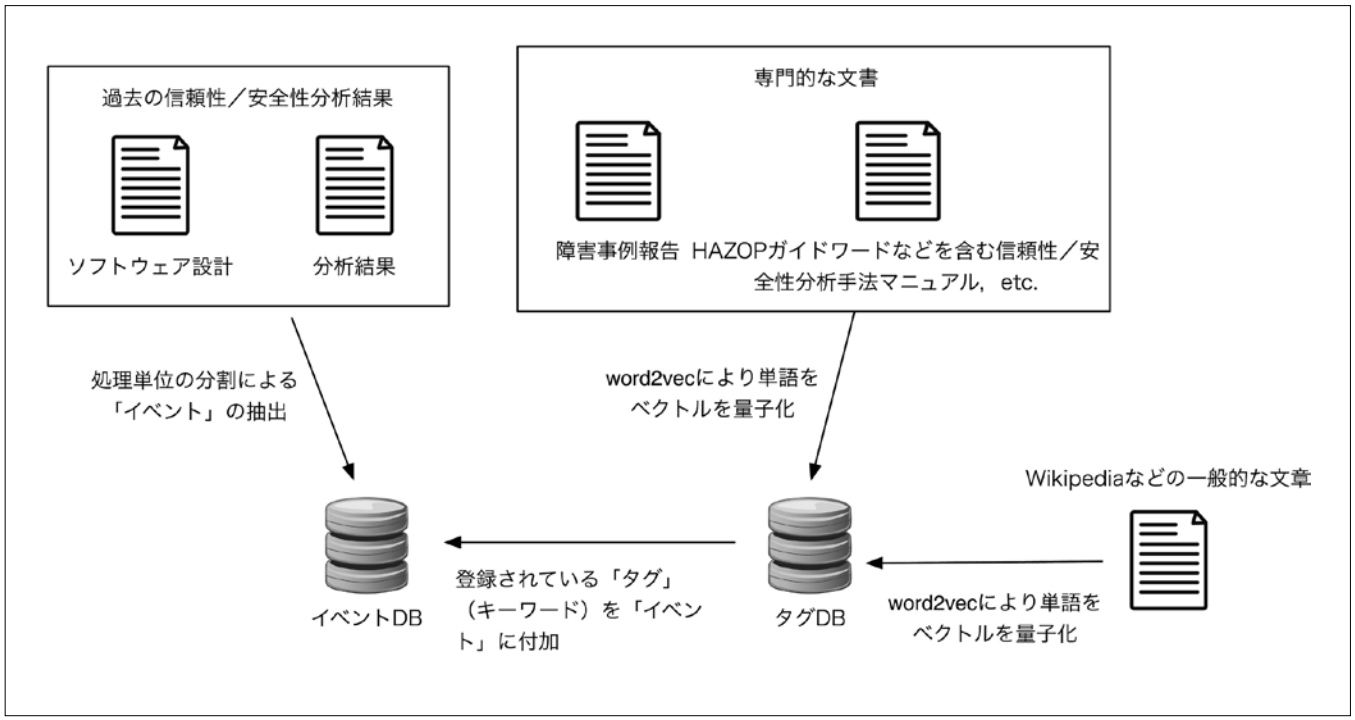


図2　支援システムにおける学習フェーズ

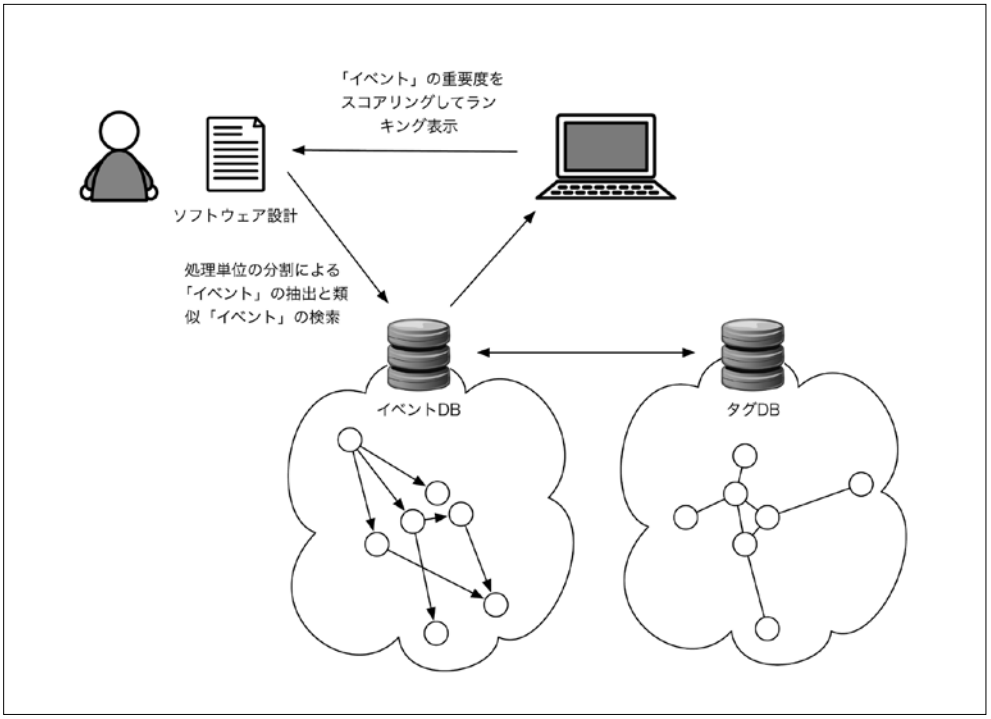


図3　支援ツールにおける評価フェーズ

ツールの機能

- word2vecツールによる出力をツール内のDBに登録する「タグDB登録」
- 過去の設計／過去のHAZOP分析結果／現在のUMLシーケンス図をDBに登録する「イベントDB登録」
- 登録されたイベントと類似するイベントを検索する「類似検索」
- 過去のHAZOP分析結果をもとに現在のシーケンス図のスコアリングを行う「重要度スコアリング」

「タグDB登録」では、word2vecからの出力されたファイル（テキストファイル）を読み込むことで、ツール独自の形式（バイナリ形式）に変換してタグを登録する。

「イベントDB登録」では、XML化されたドキュメントまたはEA（Enterprise Architecture）から出力されるUMLシーケンス図のXMLファイルから、イベントのDB登録を行う。HAZOP分析結果をはじめとするドキュメント類は、Microsoft Excel形式で管理されることも多い。そこで、Excelから入力用のXMLファイルを生成するスクリプトの作成も行った。一方、シーケンス図は、EAがエクスポートするXMLファイルを直接解析して、UMLコンポーネントおよびそれらのリンクを登録できる。

「類似度検索」では、まず検索をするもとのイベントを登録してあるイベントから選択し、指定した検索範囲に該当するイベントすべてに対する類似度を算出する。

「重要度スコアリング」は、開発したツールのメインの機能である（図4）。対象となるUMLシーケンス図とHAZOP分析結果を指定することで、シーケンス図上の各イベントに対する重要度を算出する。

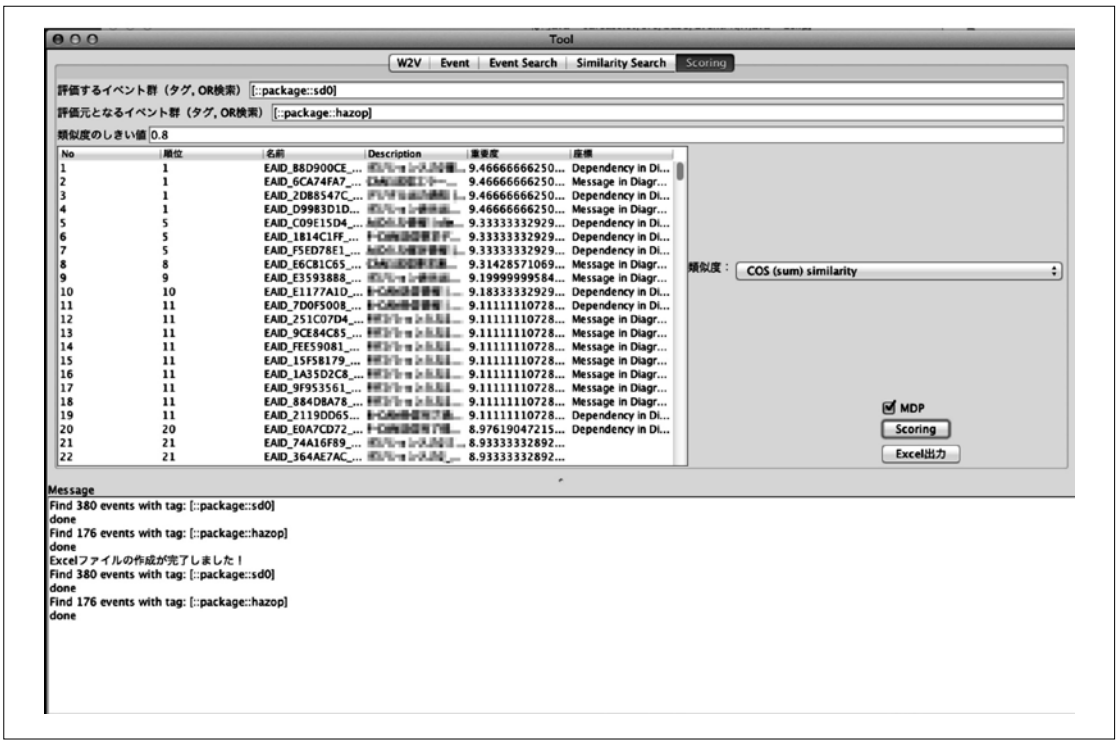


図4　重要度スコアリング

3 産業界で研究成果が適用される場面と期待される効果

今回開発したツールは、機能安全に関連した安全性分析が要請される分野（自動車制御など）において、継続的に安全性分析事例をデータ化することで、安全性分析の工数削減に、ある程度有効であると考えられる。例えば、作成した設計（シーケンス図）に対して安全性分析者がHAZOP安全性分析を行う際、ツールで得られた下位数件のイベント（シーケンス図の一部分）を省略することで、工数の削減を行うなどの利用方法が考えられる。

今後は、開発したツール（特に検索エンジンとスコアリングアルゴリズム）はフリーウェアとして一般公開することを検討している。さらに、システム開発会社やコンサルタント会社の協力の下で、協力企業を増やしながらツールを継続的に活用し、利用実績を積み上げながら改良することも予定している。

実用性が高い形式工学手法と支援ツールの研究開発

法政大学
大学院 情報科学研究科 教授 劉 少英

1 背景と目的

ソフトウェア開発プロセスの上流アクティビティ（要件分析、仕様、設計など)に使われている自然言語や図や表などは、明確な文法または意味論を定義していないため、曖昧性や意味不明の表現などが要求仕様、設計仕様など上流文書に含まれている。このため、上流文書の整合性・正当性の検証や仕様から、プログラムへの変化や作成されたプログラムの検証などは､正しく効率的に行うことが困難である。

形式手法は、数学や論理学に基づく形式的体系を用意し、その体系に基づき、システムの記述や分析を行う。ユーザの要求を正しく定義すれば、作成された最終的なプログラムは、テストや検証などを行わなくても正確な結果を出すことができる。ただし、これはあくまで理論上の話であり、実際のソフトウェア開発プロジェクトに適用する際、以下のような課題が残る。

- 要求分析とシステム設計に役に立つ形式仕様の記述手法がほとんどない。
- 形式仕様の作成は、開発者の抽象化能力と数学の知識が求められるため、一般の実務者にとっては難しい。
- 形式仕様に基づく開発は顧客とのコミュニケーションが難しい。
- 形式仕様記述による要求仕様の変更や修正には、時間、コストおよび忍耐力が求められる。

これらの課題を解決するために、SOFL (Structured Object-oriented Formal Language)形式工学手法が提案された。形式工学手法が目指しているのは、すでに企業で定着している要求分析と設計の図言語、開発手法、テスト、インスペクション (inspection)など技術の系統性、厳密性、有効性、および自動化程度を、形式仕様記述技術の利用によって向上させることである。特に、SOFL三段階形式仕様記述技術が、要求分析のための非形式仕様と半形式仕様の作成によって、形式設計仕様を作成する系統的なアプローチを提供している (図１)。

SOFL三段階技術は、既存の形式手法と比べて実用性が高くなっているが、３つの重要な課題が残っている。その３つとは以下のとおり。

- 作成された非形式仕様、半形式仕様、および形式仕様の内容をどのように顧客に容易に確認してもらい、必要なフィードバックを獲得できるか。
- 形式手法を上手に使えない開発者に対して、どのように補助すればシステム要求と設計のアイディアを明確に理解しながらそれを反映する形式仕様を容易に作成できるか。
- SOFL形式工学手法をどのように効率的かつ容易に応用できるか。

これら問題点を解決するために、本研究では、仕様アニメーション手法および形式仕様パターンに基づく操作の事前条件と、事後条件を作成するアプローチを提案する。これら手法とアプローチをSOFL三段階形式仕様記述プロセスに統合することによって、漸進的な形式仕様作成手法を確立する。さらに、この手法を効率的に支援するソフトウェアツールのプロトタイプを開発する。

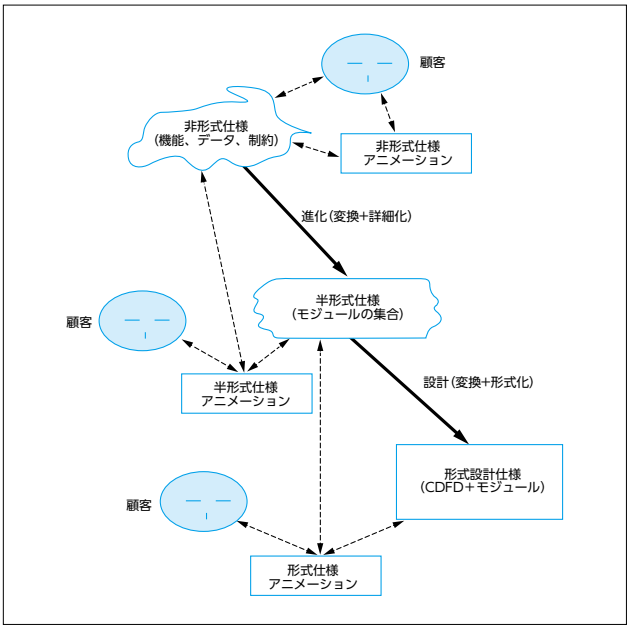


図１　SOFL三段階漸進的形式仕様記述プロセス

2 概要

(1)非形式仕様アニメーション

非形式仕様アニメーションとは、SOFL非形式仕様に記述されているシステムの機能(function)、データリソース、および制約(constraints)を、現実世界の仮想的な環境で動的に表現することである。目的としては、作成された非形式仕様の内容を顧客に確認し、必要なフィードバックを獲得し、非形式仕様を徐々に完成させることである。アニメーションが顧客の完全な要求を見出すことに役立つように、非形式仕様を作成していくに伴い、アニメーションを徐々に見せていく漸進的なアプローチとなる。

本研究では、非形式仕様アニメーションを素早く行うために、非形式仕様アニメーションの支援ツールを、PowerPointにアニメーション用のコンポーネントを追加することによって開発した。非形式仕様アニメーションのために、各種のコンポーネントとテンプレートを導入している(図2)。

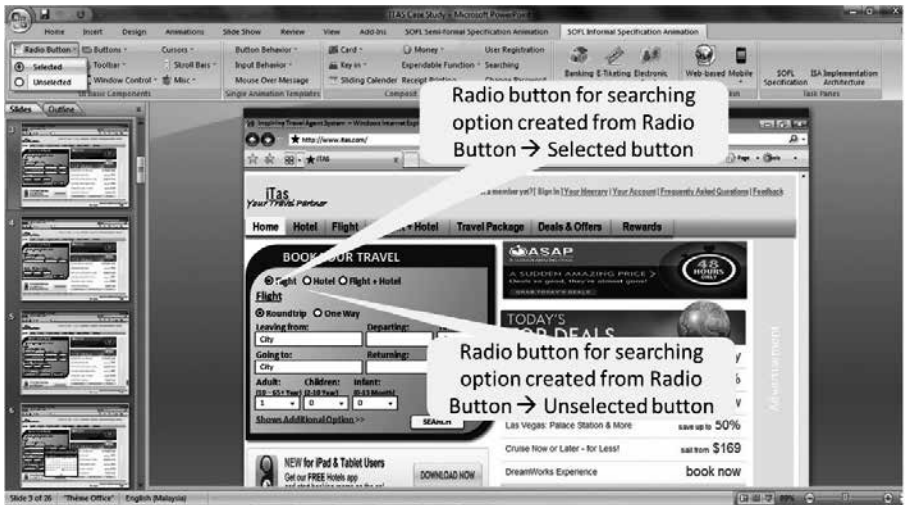


図2　ユーザインタフェースコンポーネントの使用事例

(2)半形式仕様アニメーション

SOFL半形式仕様は、非形式仕様を詳細化して得たユーザの要求をより明確に定義している仕様であり、関連している機能、データリソース、および制約をSOFLモジュールにまとめている。構造としては、SOFL半形式仕様が、SOFLモジュールの集合である。

半形式仕様アニメーションとは、SOFL半形式仕様に含まれているモジュールにおいて定義されているプロセス(操作)の機能を、一つずつ動的に表現することである(図3)。目的としては、半形式仕様で定義されたプロセスのインターフェース、入力変数、出力変数、これらの型、外部変数と型、期待されている振る舞いを、開発者(つまり、半形式仕様の作成者)と顧客に確認してもらい、必要なフィードバックを獲得し、半形式仕様を期待される形に改善する。

半形式プロセス仕様アニメーションの支援ツールは形式仕様アニメーションの支援ツールと同様に、PowerPointにアニメーション用のコンポーネントを追加することによって開発した。

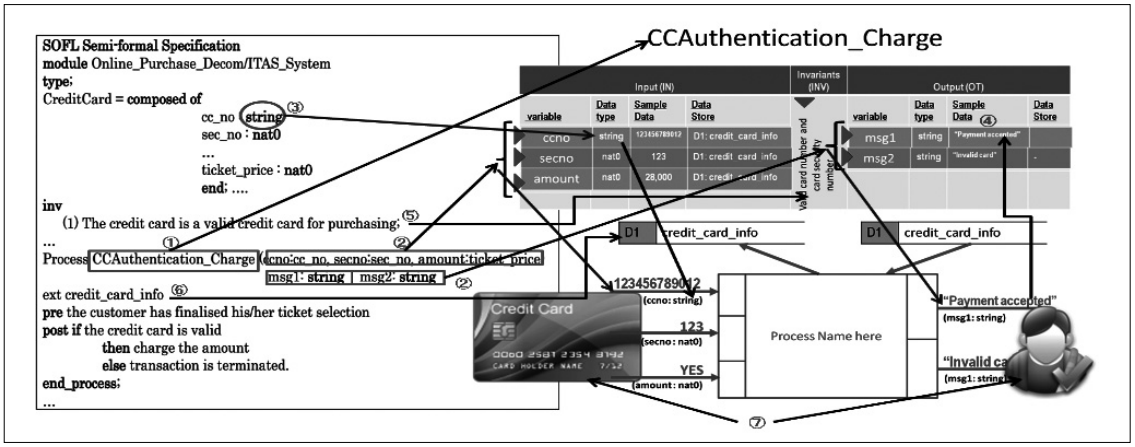


図3　半形式仕様アニメーションの事例

(3)形式仕様アニメーション

SOFL形式仕様は、基本的には階層的なモジュールと階層的なCDFD（Condition Data Flow Diagram）から構成されたものである。CDFDは、イベント駆動の操作意味論を持つ形式的データフロー図であり、システムのアーキテクチャをデータの流れの観点から描画する。CDFDに対応するモジュールには、そのCDFDの各部品（プロセス、データフロー、データストアなど）を、SOFL形式仕様記述言語で明確に定義される。

形式仕様アニメーションとは、CDFDから「システム機能シナリオ」を導出し、その振る舞いを入力によって出力するプロセスを動的に表現することである。目的としては、開発者と顧客に形式仕様で設計したシステムの機能シナリオを確認してもらい、システム機能シナリオの整合性も確保することである。この中で、主な研究課題には以下のような内容が含まれる。

- CDFDからどのようにシステム機能シナリオを自動的に導出するか
- 導出されたシステム機能シナリオの振る舞いをどのように動的に表現するか
- 入力と出力変数の値は、どのように生成できるか

このような課題をすべて解決するためには、長い期間の研究が必要である。本研究では、CDFDからシステム機能シナリオの自動導出する技術を、研究の中心として取り込んだ。

CDFDからシステム機能シナリオを自動生成するアルゴリズムの基本的な考え方は、CDFDから入力データフローと出力データフロー間の関係をより細かく表示するグラフへ変換し、それによりシステム機能シナリオが自動生成されるという方法である。具体的なCDFDを受けると、このCDFDから生成されるすべての機能シナリオが提供される。そのなかで、一つの機能シナリオを選択すれば、そのシナリオのアニメーションを実施することが可能である。

例としてATMシステムのCDFDから選択されたシステム機能シナリオをアニメーションにするステップを示す（図4～図7）。

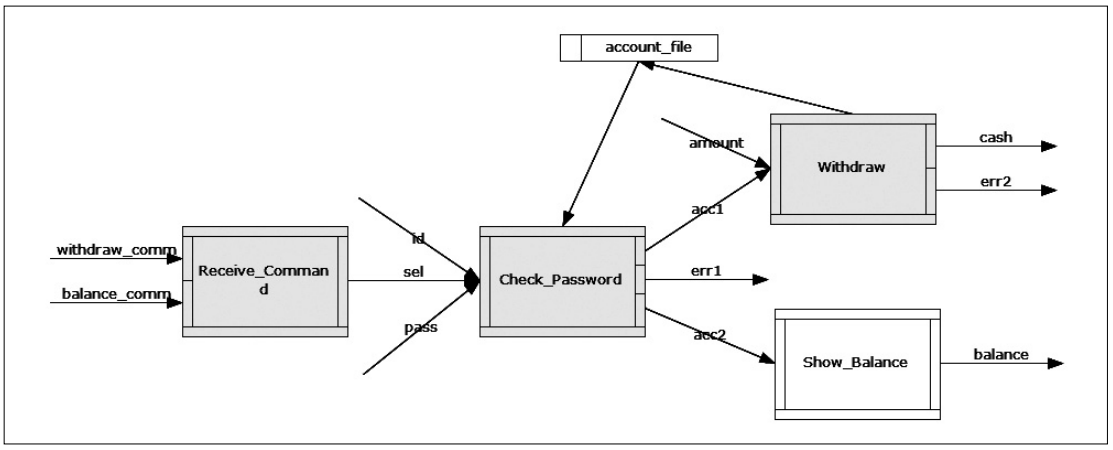


図4　ATMシステムのCDFDから選択された機能シナリオ

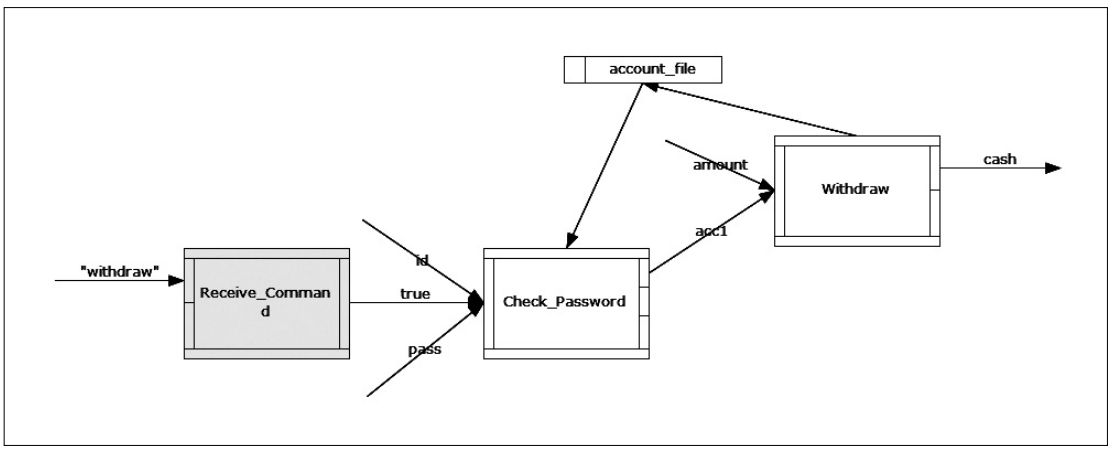


図5　アニメーションのステップ1

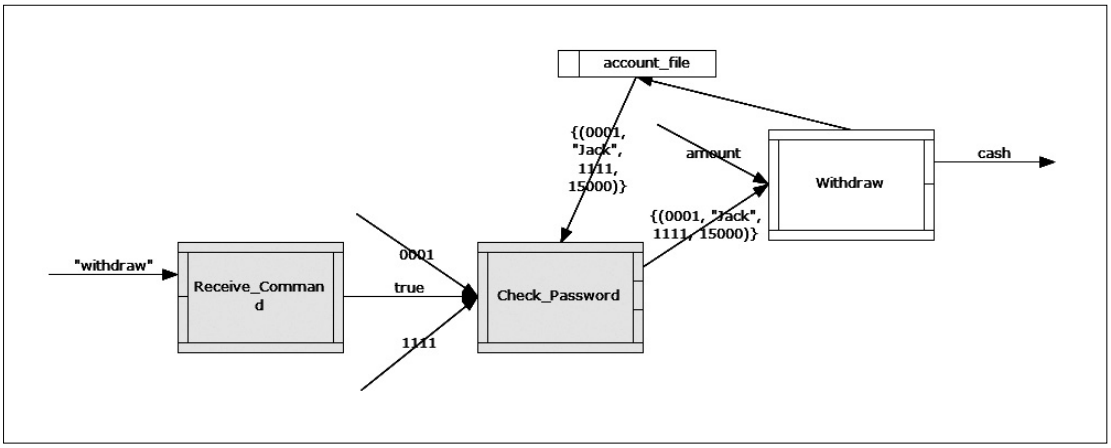


図6　アニメーションのステップ2

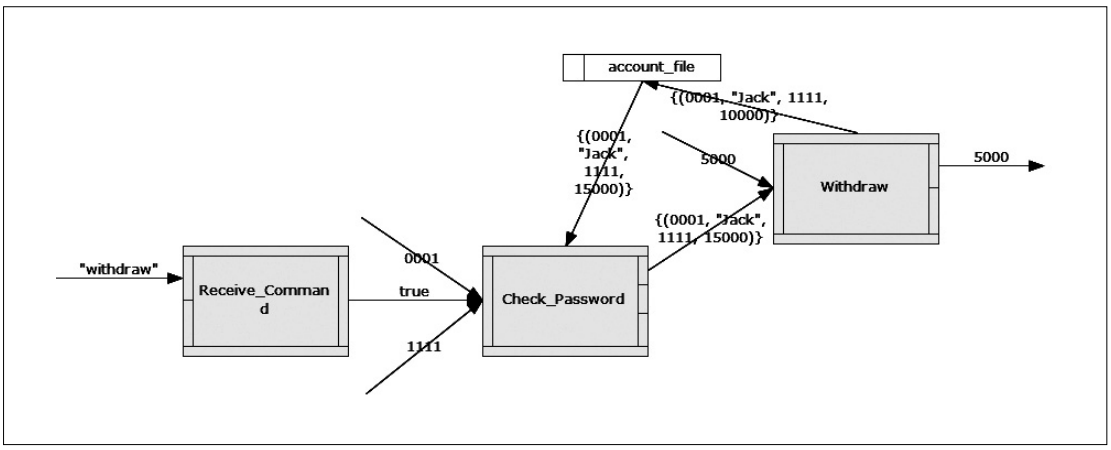


図7　アニメーションのステップ3

3 産業界で研究成果が適用される場面と期待される効果

開発者と顧客間のコミュニケーション強化

SOFL非形式仕様を用いてソフトウェア開発する場合、本研究で提案したアニメーションアプローチに従い、顧客と協力しながらアニメーションを行うことで、開発者と顧客間のコミュニケーションを強化できる。非形式仕様に対するフィードバックを顧客から獲得することで、完全かつ正しい非形式仕様を作成できると考えられる。また、顧客がSOFL非形式仕様を書き、それによってアニメーションシステムを作成し、開発者に顧客自身の具体的な要求を説明することもできると考えている。

グローバルソフトウエア開発プロジェクトへの適用

海外発注ソフトウェアプロジェクトの場合、要求仕様は日本で非形式言語（例えば、日本語）により作成され、海外の実装グループに渡してプログラムを開発するパターンが多い。この場合、海外の実装グループがその要求を正しく理解していない可能性がある。この問題を解決するためには、SOFLの半形式要求仕様を日本で作成し、その上で半形式仕様に定義されている重要なプロセス（操作）の振る舞いのアニメーションシステムを作成したうえで、海外で実装してもらうことが考えられる。これにより実装者は半形式仕様を理解するときに、そのアニメーションを参考にできるので、正確な理解が保証できる。この場合、海外の実装グループは、半形式仕様のアニメーションシステムによって、要求された機能、データ構造、および制約を正しく理解する可能性が高い。実装するプログラムの信頼性も高くなるであろう。

コードクローン分析に基づくソフトウェア開発・保守支援に関する研究

大阪大学
大学院情報科学研究科 教授 楠本 真二

1 背景と目的

近年、ソフトウェア開発・保守を阻害する問題の一つとして、ソースコード上のコードクローンに関する問題が提起され、活発に研究がなされている。コードクローンとは、ソースコード上に存在する同一、または、類似したコード片を意味する。ソースコードの流用により、「バグの拡散」あるいは「機能追加時の一貫性確保の難しさ」などの問題が指摘されている。特に、対象ソフトウェアが大規模な場合、チェックすべき箇所が膨大な数になることから、人間がすべての重複部分を認識しておくことは難しい。このため、大規模ソフトウェアからのコードクローン検出手法や分析結果の利用などが研究されてきている。実際、ソフトウェア工学国際会議においてもコードクローンに関するセッションが企画され、ソフトウェア保守に関する国際会議においても主要な研究テーマとなっている。さらに、コードクローンに特化した国際会議も開催されている。

一方で、上述した国際会議などにおける、ソフトウェア開発企業からの報告はほとんどなされていない。そのため、研究成果の開発現場への普及状況は、あまり十分ではないと推察される。一部のソフトウェアベンダーがソースコード解析サービスとして提供している、あるいは、企業内でのローカルな利用にとどまっているのが現状である。その理由としては、一般的なコードクローン検出／分析手法を研究論文・研究発表の場で提示するだけでは不十分で、実際の開発現場における利用目的や状況に特化した手法の開発とその有用性の評価結果を合わせて提示することが、現場への導入の高い動機付けになると考えられる。

そこで本研究では、ソフトウェア開発や保守の様々な活動、状況（コンテキスト）に応じた支援を行うことを目的として、いくつかのコンテキストに応じたコードクローン検出手法の開発と検出されたコードクローンに対する対策手法の開発を行った。具体的には、ソースコード理解支援、リファクタリング支援、再利用ライブラリ作成支援、違反流用コード発見支援の４つのテーマについて、支援手法の提案、プロトタイプの開発、有効性の評価を行った。

2 概要

ソースコード理解支援

本テーマでは、細粒度で極力多くのコードクローンを高速に検出する手法を開発した。具体的には、既存のコードクローン検出ツールと比較して、冗長なコードクローンをなるべく含まず、処理が高速で、検出したコードクローンの再現率・適合率が高いコードクローン検出ツールの開発を目指した。再現率は「正解コードクローンの中の何割をツールが検出できるか」を表し、適合率は「ツールが検出したコードクローンの何割が正解であるか」を表す尺度である。極力多くのコードクローンを検出するという立場であれば、再現率が重視される。正しいコードクローンを検出するという立場であれば、適合率が重視される。細粒度でなるべく多くのコードクローンを検出できるツールとして著名なものに、CCFinderがある。しかし、CCFinderが検出するコードクローンの中にも冗長なものがある。また、処理が高速なツールであるが、一部詳細な解析を行っているため、数千万行、数億行のコードに対しては多くの時間を要する。そこで、細粒度でなるべく多くのコードクローンを高速に検出することを目指し、具体的な目標として、CCFinderよりも再現率・適合率が高いコードクローン検出手法、ツールの実現を目指した。

冗長なコードクローンとして、ソースコード中に現れる繰り返し部分に着目した。既存の行単位や字句単位の検出手法には、ソースコード中の同じ命令が繰り返し記述された部分（以降、繰り返し部分と呼ぶ）において、冗長なコードクローンを検出する、並びに検出すべきであるコードクローンを検出できない、という課題が指摘さ

れている（図1）。

本テーマでは、冗長なコードクローンが繰り返し部分を含むコードクローンであると考え、より有益なコードクローン検出結果を得るために、ソースコード上の繰り返し構造を折りたたむという検出の前処理を行った上で、コードクローン検出する手法を考案した。

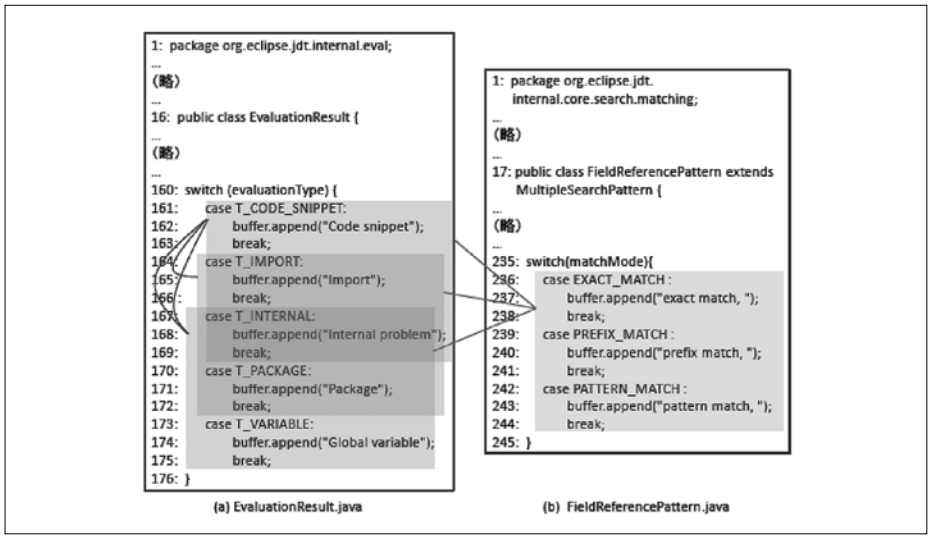


図1　繰り返し部分に対するコードローン検出

考案した手法に基づくプロトタイプシステムを開発し、いくつかのオープンオースソフトウェアと実プロジェクトデータに適用して、従来手法と提案手法で検出されたコードクローンの再現率・適合率を比較した。その結果、すべての適用対象ソフトウェアについて、折りたたみ処理を行うことで、検出されたコードクローン数が削減できた。すなわち、冗長なコードクローンの削減が達成できたと考えられる。一方、再現率と適合率については、適合率はすべての対象について向上したが、再現率については向上したものと悪くなったものが見られた。また、複数のオープンソースソフトウェアに対する検出結果の平均値で、CCFinderよりも早くコードクローン検出が行えた（図2）。

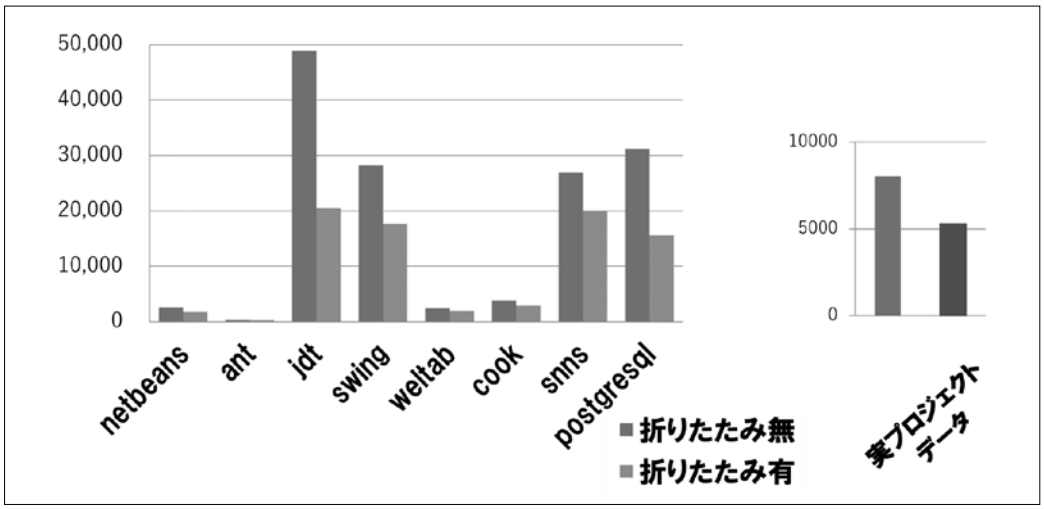


図2　折りたたみ有・無でのコードクローン検出数

リファクタリング支援

本テーマでは集約しやすいコードクローン分析・対処手法を開発した。コードクローンへの対策の一つは集約である。集約とはコードクローンとなっているコード片を1つのメソッドなどにまとめることである。集約により保守の対象となるコードクローンの存在を除去することが可能となる。

集約の方法としては、コードクローンを検出して、いわゆるリファクタリングパターンを適用することが考えられる。コードクローンに対するリファクタリング手法としては、Extract Method（メソッドの抽出）、Pull Up Method（メソッドの上位階層への引き上げ）などがよく行われる。しかし、当然ながらリファクタリングパターンはコードクローンを対象として提案されているものではないので、コードクローンの特徴によっては簡単にリファクタリングができないこともある。また、コードクローン集約作業の初級者が既存のリファクタリングパターンを参照して集約作業を行う際に、途中で集約作業を中止する、もしくはバグを含む修正を行ってしまう状況も考えられる。

一つの解決方法として、コードクローンの特徴に応じた集約方法のガイドラインの作成やリファクタリングに適したコードクローンの自動抽出が考えられる。より多くのリファクタリングパターンに対応できるツールの開発・集約方法のガイドラインの開発を目指し、Template Methodパターンと呼ばれるデザインパターンに着目した。Template Methodパターンとは、「共通の親クラスを持つ類似メソッドを対象とし、メソッド間で共通の処理を親クラスに記述し、メソッドごとに異なる処理を子クラスに記述する」というパターンである。このパターンを用いたコードクローン集約手法では、メソッド間でコードクローンとなっている箇所を共通の処理として親クラスに引き上げ、コードクローンとなっていない箇所を子クラスに記述することで、コードクローンの集約を実現している。この手法の最大の特長として、「対象となるメソッドがコードクローンとなっていない箇所を含んでいても適用が可能である」という点が挙げられる。この手法に基づく支援手法がいくつか提案されている。しかし、既存手法には次に挙げる課題点が存在する。

- 変数名などのユーザ定義名のみが異なるコードクローンを集約できない。
- 意味的に同じ処理を行っているコードでも、その表現方法が異なる場合は集約できない。

そこで本テーマではこれらの課題点を改善するため、プログラム依存グラフを用いたTemplate Methodパターンの適用支援手法について実施した。具体的には、Template Methodパターンの適用候補となるコードクローンを自動的に検出し、ユーザに提示するプロトタイプシステムを開発した（図3）。さらに、いくつかのオープンソースソフトウェアと実プロジェクトデータに適用し、検出されたコードクローンの内容と検出時間を評価した。適用結果は表1のとおり。規模3万行～32万行程度のソフトウェアに対して、表に示す時間で、Template Methodパターンが適用可能なコードクローンを検出できた。また、Apache Synapseから検出した45個の候補に対して、Template Methodパターンを適用して集約を行った。1個あたり10分程度で集約を行え、集約前後で動作が変わらないことを確認した。

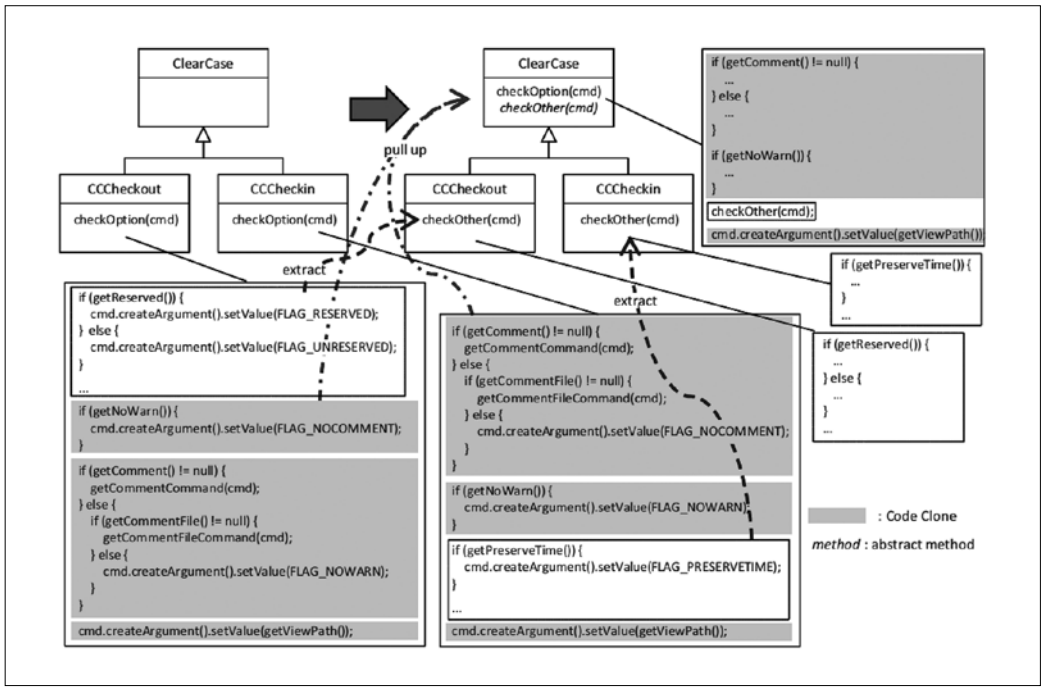


図3　抽出例

表1　適用結果

Target Systems	LOC	# of Candidates	Elapsed Time[s]
Apache Ant	212,401	226	237
Argo UML	328,582	486	1,080
Apache Synapse	58,418	45	95
IT Spiral	31,948	9	45

再利用ライブラリ作成支援・違反流用コード発見支援および違反流用コード発見支援

再利用ライブラリ作成支援・違反流用コード発見支援では、単独で頻繁に再利用されるコード片の検出支援を行った。一般にコードクローンはソフトウェアの保守性を阻害すると言われている。一方、研究チームが過去にいくつかの企業のソフトウェアに対してコードクローン分析を行った時に、品質の高いコード片については積極的にコードクローンとして利用しているという意見があった。しかし、品質が高いコード片であっても、将来的に変更修正が発生しないという保証はない。従って、品質が高いコード片で、多くのプロジェクトで利用(流用、コピー)されているものは、ライブラリとしてまとめておくことが無難である。一方で、ライブラリとしてまとめる部品については、その粒度も検討する必要がある。字句／行単位で検出されるコードクローンは部品とするには小さすぎ、ファイル単位のコードクローンは粒度が大きすぎる。そこで、大規模なソースコード群から、適切な大きさ(例えば、メソッド、関数単位)のコードクローンをなるべく高速に検出するための手法とツールの開発を目指した。

違反流用コード発見支援では、ライセンス違反などが疑われるコードの検出について検討した。各ソフトウェアのライセンスと、再利用しているソフトウェアから到達できるライセンス集合を比較し、矛盾の有無を判定するという研究が活発に行われている。その多くは、ライセンス記述部分を用いた分析が多いが、流用されているソースコードは流用元のソースコードのコードクローンとなっていることが考えられるため、コードクローン分析は分析の一つの方法となる。しかし、ライセンス違反チェックのためには、大量のソースコードからのコードクローン検出が必要となるため、既存のアプローチではコストが非常に多くかかる。また、再利用ライブラリ作成支援でも述べたとおり、適切な粒度も決める必要がある。そこで、目標としては再利用ライブラリ作成支援・違反流用コード発見支援と同じく、大規模なソースコード群から適切な大きさ(例えば、メソッド、関数単位)のコードクローンを、なるべく高速に検出するための手法とツールの開発を目指した(図4)。結果として、再利用ライブラリ作成支援・違反流用コード発見支援、および違反流用コード発見支援では、適用対象をJavaのソースコード群として、検出粒度をメソッド単位とし、メソッド単位のコードクローン検出を行うプロトタイプシステムを開発するとともに、大規模Javaソースコード群に対する適用実験を行った。

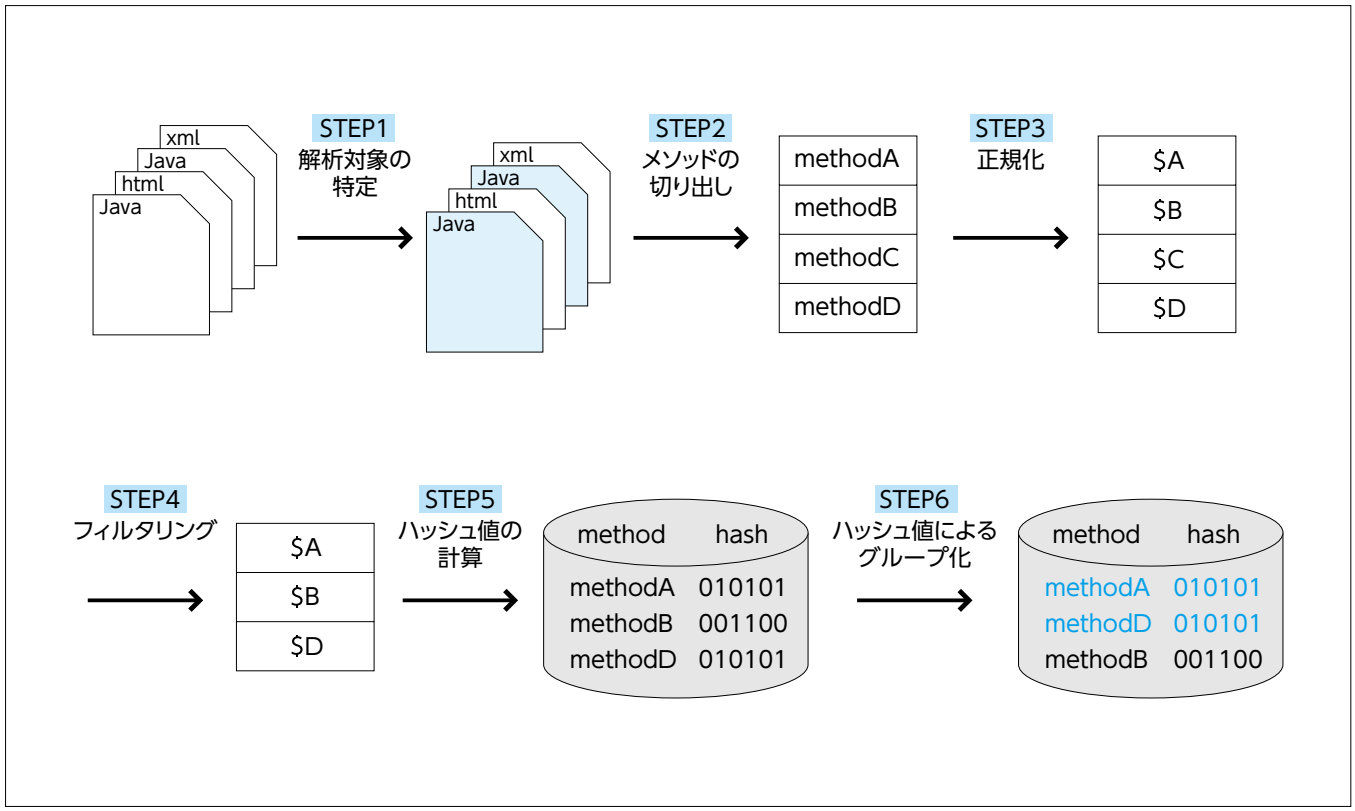


図4　メソッド単位コードクローン検出概要

"UCI source code data sets"（ファイル数：200万、総行数：約４億行）に対して提案手法を適用した結果、約５時間で約81万（要素メソッド数：約300万）のメソッドクローンを検出した。これらの中には、流用が疑われるメソッドクローンや再利用ライブラリとして有用なメソッドクローンが含まれており、本手法の有用性が確認できた。

3 産業界で研究成果が適用される場面と期待される効果

ソースコード理解支援

近年デファクト的に使用されているCCFinderをはじめとして、行単位・字句単位のコードクローン検出手法が検出してしまう冗長なコードクローンを削減することに加えて、細粒度でなるべく多くのコードクローンを高速に検出することが可能となるコードクローン検出手法を開発することは、コードクローン分析の精度、効果を高めることになる。

リファクタリング支援

コードクローンへの対策の一つは集約である。集約とはコードクローンとなっているコード片を１つのメソッドなどにまとめることである。集約により保守の対象となるコードクローンの存在を除去することが可能となる。従来対応可能であった単純なリファクタリングパターンだけではなく、より複雑なリファクタリングパターンへの対応をすることで、リファクタリング作業の効率化に寄与することができる。

再利用ライブラリ作成支援・違反流用コード発見支援

一般にコードクローンはソフトウェアの保守性を阻害すると言われているが、品質の高いコード片については積極的にコードクローンとして利用している例もある。しかし、品質が高いコード片であっても、将来的に変更修正が発生しないという保証はない。従って、品質が高いコード片で、多くのプロジェクトで利用（流用、コピー）されているものは、ライブラリとしてまとめておくことが無難である。

大規模なソースコード群から、適切な大きさのコードクローンをなるべく高速に検出するための手法とツールの開発を行うことで、企業などで蓄積された大規模ソースコード群から有用な再利用部品群を作成することができ、開発効率の改善に役立つ。

違反流用コード発見支援

近年、再利用されたソースファイルのライセンスと再利用先のソフトウェアのライセンス間で不整合が生じていることで、商用のソースコードに対しても、ソースコードの公開が求められたり、販売停止となったりするようなことが発生している。ライセンス違反などが疑われるコードの自動検出により、違反流用に対するリスク削減が可能となる。

ソフトウェア品質の第三者評価のための基盤技術
―ソフトウェアプロジェクトトモグラフィ技術の高度化―

奈良先端科学技術大学院大学
情報科学研究科 教授 **松本 健一**

1 背景と目的

2009～2010年に米国で発生したトヨタ自動車の急加速問題では、当初、エンジンの電子制御ソフトウェアの欠陥が疑われた。しかし、NASA工学・安全センター(NESC)が、「専門知識を有する中立的立場の第三者」として当該ソフトウェアの品質評価を行った結果、ソフトウェアの設計や実装に欠陥は発見されず、急加速の多くは、運転手の操作ミスに起因することが確認された。

こうした「ソフトウェア品質の第三者評価」の必要性・重要性は、今後ますます高まると考えられている。本研究責任者らも、2012年度のソフトウェア工学分野の先導的研究支援事業において、ソフトウェア品質の第三者評価の技術基盤の確立を目指し、「ソフトウェアプロジェクトトモグラフィ(Software Project Tomography)」と名付けた新しい概念を提案し、プロトタイプシステムの実装と実証実験を行った(図1)。

「ソフトウェアプロジェクトトモグラフィ」とは、ソフトウェア開発プロジェクトを「要件」「課題」「プロダクト」「組織」「作業」の5つの観点を持つスナップショットの系列で表現する手法である。プロトタイプ実装と実証実験を通じ、ソフトウェア品質の第三者評価では、「品質評価に必要となるソフトウェアプロジェクトデータの提供」、および「提供されたデータに基づくプロジェクト理解」が容易になるといった妥当性・有用性が示された。

ただし、ソフトウェアプロジェクトトモグラフィは端緒に就いたばかりの研究ということもあり、概念的な議論やシーズ先行の技術開発と言える面もあった。とはいえ、産業界では最近、ソフトウェア品質の第三者評価に関する実験や試行などが開始され、第三者評価に対する具体的なニーズや評価プロセスも明らかになりつつある。それらニーズやプロセスに基づき、ソフトウェアプロジェクトトモグラフィ技術を高度化することができれば、より強固で実践的な「ソフトウェア品質の第三者評価の技術基盤」が確立されると考える。

2 概要

(1)ソフトウェアプロジェクトトモグラフィと本研究

ソフトウェアプロジェクトトモグラフィを考えるうえでモデルとしたのは、医療におけるコンピュータ断層撮影、いわゆるCT (Computed Tomography)である。つまり、ソフトウェア開発プロジェクトを“からだ”に見立て、プロジェクト開始から終了まで(あるいは、現時点まで)のいくつかの時点において、プロジェクトの状況を定量的に表すスナップショットを断面画像のように作成する。CTの場合、スナップショットの数が十分であれば、断面画像の系列から身体の3次元モデルが再構成できる。

ソフトウェアプロジェクトトモグラフィも同様の発想である。スナップショットを作成することで、その系列のプロジェクトを様々な観点で可視化し、ソフトウェアやその品質が実現される過程(プロセス)を表すことを可能にする。これにより、「開発管理」のために収集・蓄積されているソフトウェア開発プロジェクトデータを、「解析と可視化」に都合のよい形式に変換する手法とも位置付けることができる。

本研究では、ソフトウェアデータマイニング・アナリティクスの最新技術を取り入れると共に、「ソフトウェア品質の第三者評価」のニーズやプロセスにも焦点をあて、ソフトウェアプロジェクトトモグラフィ技術を高度化した。

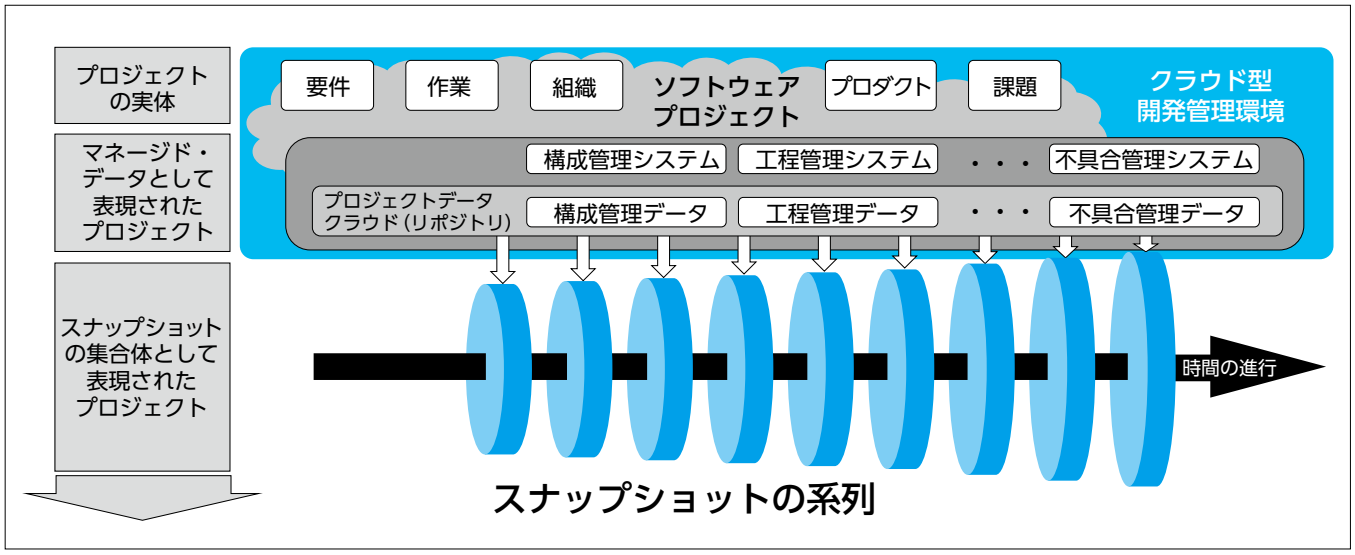


図1　ソフトウェアプロジェクトトモグラフィの基本概念

(2) 欠陥の収束・発散プロセスをトピック別に評価するモデルの構築

ソースコード中の欠陥(課題)が収束・発散するプロセスを、そのトピック別に評価するモデル(欠陥評価モデル)を構築した(図2)。具体的な手順は次のとおり。

1　ソフトウェア開発における欠陥レポート(障害レポート、バグ票など)にトピックモデリングを適用し、欠陥トピックを自動特定する。
2　それら欠陥トピックやトピック間の関係を説明することのできるソフトウェアメトリクスを、SEM(Structural Equation Modeling／構造方程式モデリング)により特定する。
3　SEMで得られたメトリクス間(変数間)の関係情報に基づき、ベイジアンネットワークによって欠陥評価モデルを構築する。

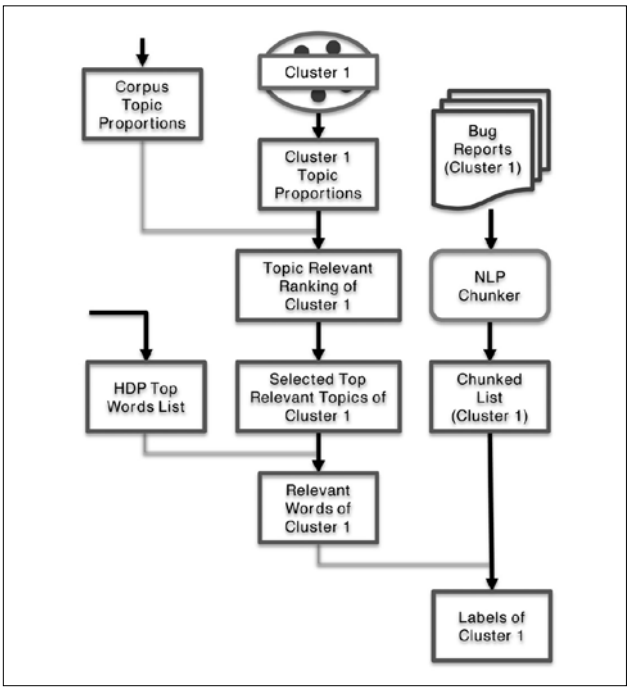


図2　欠陥トピック特定プロセス

トピックはいわば、評価対象とするプロジェクトや欠陥そのものの特性を反映した分類基準である。トピックによって一般性の高い既定の分類基準では顕在化しないような欠陥をあぶり出し、その収束・発散プロセスを示すことができる(図3)。評価対象とするプロジェクトや欠陥に関する知識が必ずしも豊富とはいえない第三者が、プロジェクトの特性や実態を理解する上で大きな助けとなる。また、それら知識が豊富な者に対しても、プロジェクトの体制やプロセスの改善に新たな観点、論点を提供することになる。

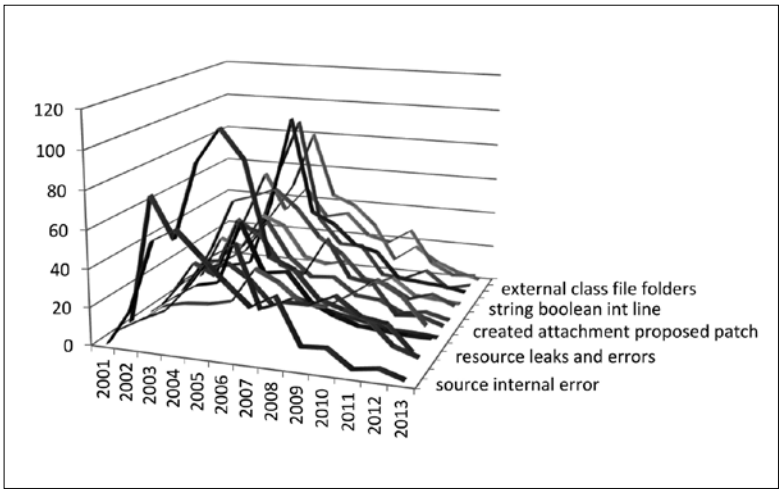


図3　欠陥の収束・発散プロセスのトピック別での可視化例

(3) ソフトウェア障害の波及度を定量的に評価する方式の開発

ソフトウェア開発で発生する、障害の波及度を定量的に評価する方式を開発した障害の波及度は、障害除去の優先度・優先順位を決定する上で重要な指標である。しかし、その定量的評価法はこれまで確立されておらず、暗黙知とされてきた。本研究では、ユーザーから発信される「クラッシュレポート」を用いて、障害の発生時期やOSやバージョンといったユーザー環境の特徴量を抽出することで、波及度を算出する。

算出においては、ソフトウェア開発者へのヒアリングで得られた知見を定式化している。なお、プロトタイプ実装には、統計解析向けプログラミング言語「R」を用いている。波及度の算出例、および波及度に基づく第三者評価シナリオ例は以下図４～図６のとおり。

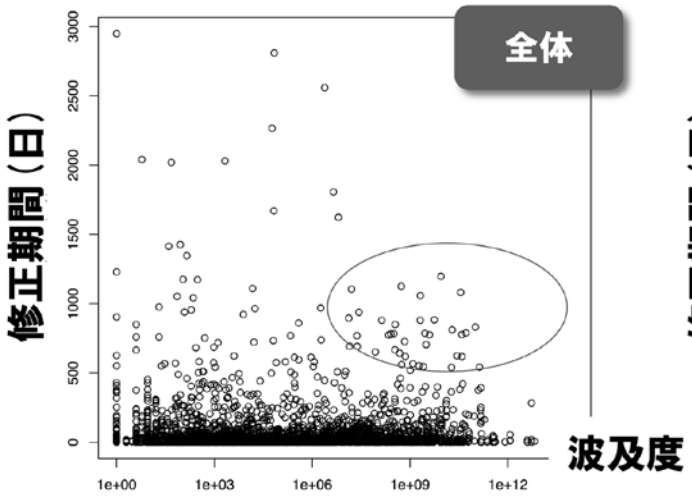


図４　全障害

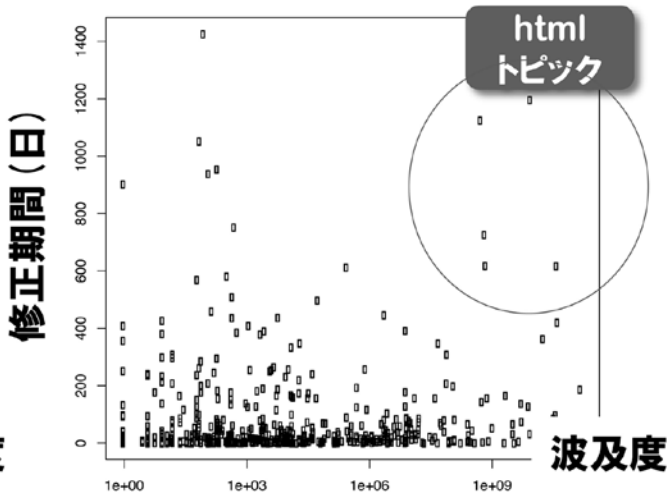


図５　トピック「html」で表される障害

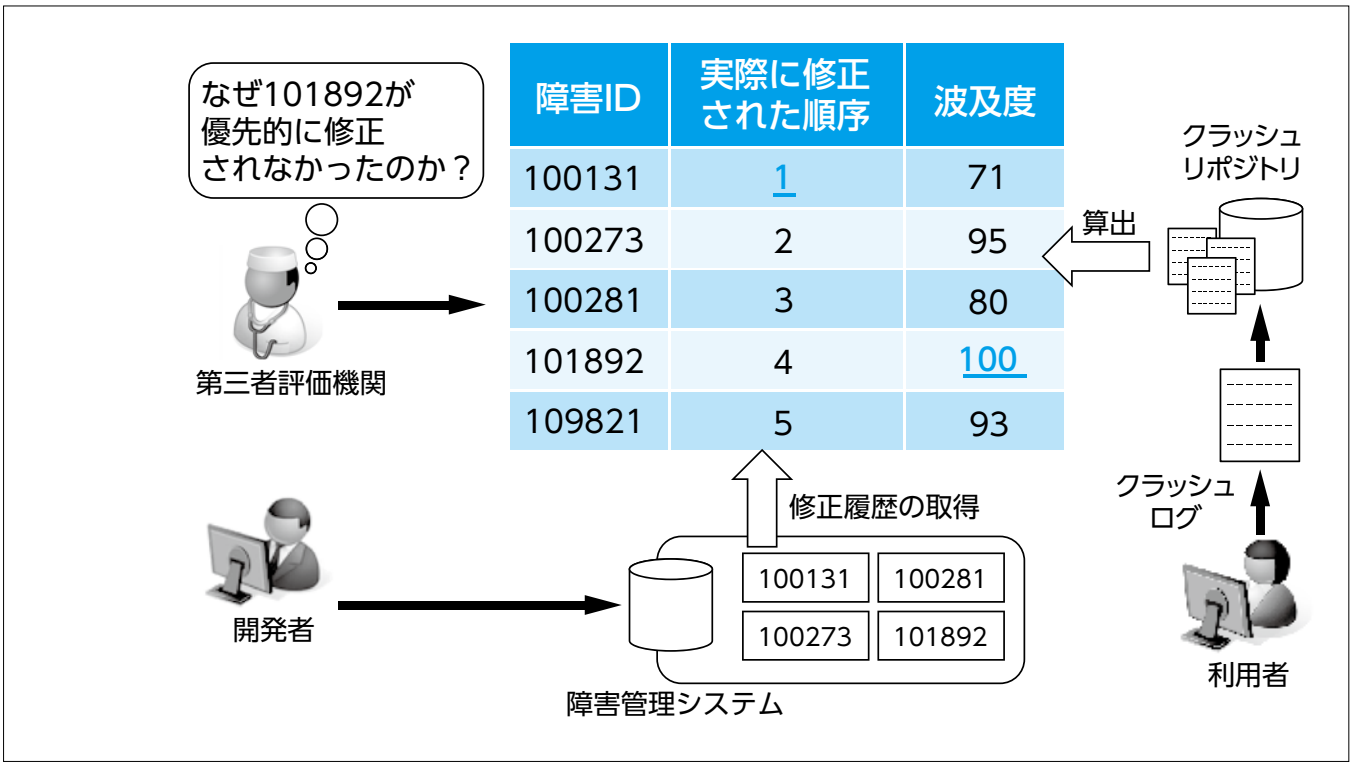


図６　波及度に基づく第三者評価シナリオ例：障害修正プロセス評価

(4)非機能要件の自動評価方式の開発

ソフトウェア開発の上流工程で作成されるRFP（Request For Proposal）において、非機能要件の記述の明確さを自動評価する方式を開発した。具体的には、非機能要件に関連する語句（キーワード）を説明変数とし、要件記述の明確さを目的変数とするモデルを、ランダムフォレスト法により構築した。RFPからの語句抽出には形態素解析を用い、モデル構築においては、TF-IDF法（Term Frequency - Inverse Document Frequency Method）による語句の重み付けも行った。TF-IDF法では、語句の重みがTF（Term Frequency ／単語の出現頻度）とIDF（Inverse Document Frequency ／逆文書頻度）の２つの指標に基づいて計算される。評価対象とする非機能要件は、開発者へのヒアリングや文献調査によって選定した26個である。

Web上から入手可能な70件のRFPを対象とした実証実験の結果は図７～図８のとおり。なお、内訳件数は以下のとおり。

- 図書館情報システム　11件
- 病院情報システム　10件
- 大学情報システム　8件
- 政府機関情報システム　14件
- 自治体基幹情報システム　10件
- 地方自治体業務システム　14件
- その他情報システム　3件

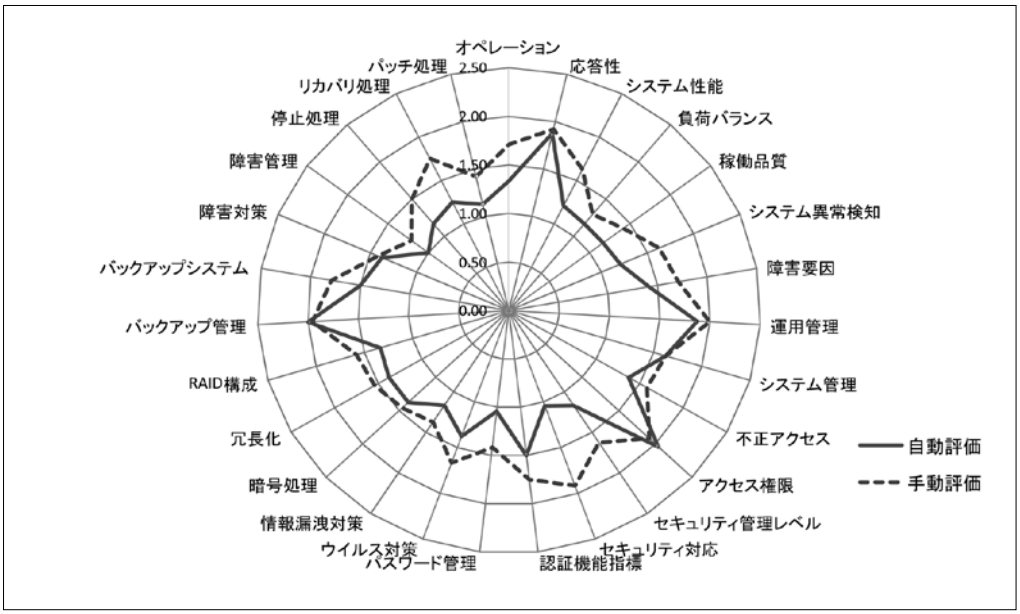


図７　３段階評価

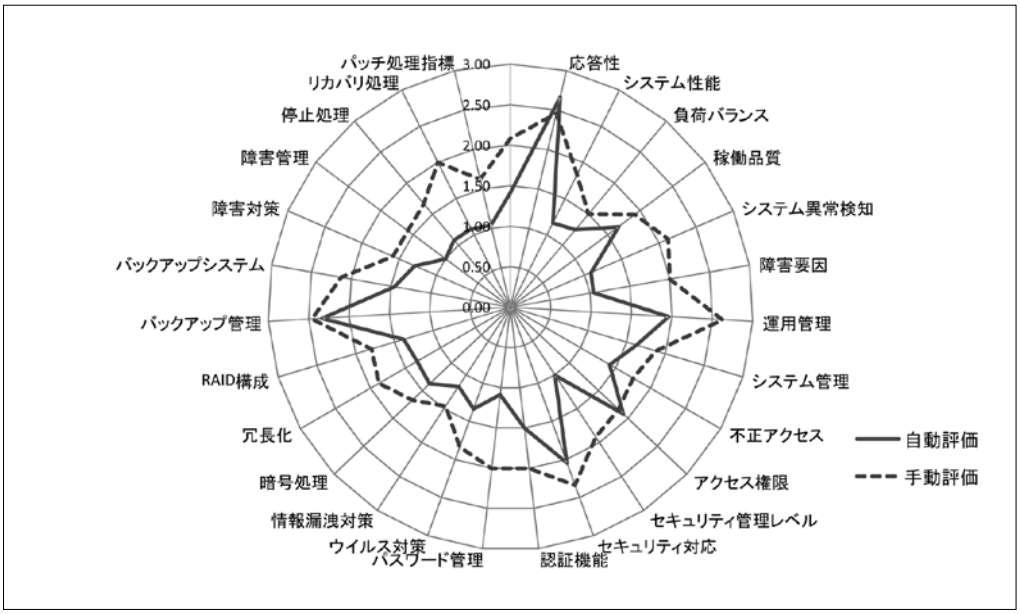


図８　５段階評価

(5) 多人数の開発作業を正当なデータに基づいて解析・可視化する方式の開発

多数の開発者が携わるソフトウェア開発プロジェクトにおける開発タスクを、「正当なデータ」に基づいて解析・可視化する方式を開発した。

具体的には、ソフトウェア開発タスクの実施に伴うアプリケーション実行履歴を、識別のためのハッシュ値と共に記録すると同時に、実行履歴の正当性を検証することのできるシステムを実現した。なお、多人数での開発作業を対象とするため、サーバ／クライアント方式とした。

アプリケーション実行履歴は、開発タスク計測システムTaskPitを通じて収集される。また、改ざんされたアプリ実行履歴の検出だけでなく、改ざんされたクライアントからのサーバ接続要求を拒否する機能も有する。収集されたアプリケーション実行履歴は、開発組織に閲覧可能な状態で蓄積されるため、プロジェクト管理者が組織内で行うプロセス改善に利用することも可能である。さらに、アプリケーション実行履歴と開発作業・タスクの対応関係を機械学習によりモデル化し、それらの間の自動マッピングを90%超の精度で可能にした。これにより、アプリケーション履歴を開発作業履歴と見なし、評価や可視化に利用することが可能になった（図9）。

図9　システムの画面表示例、および、同システムが実現するソフトウェア開発組織と第三者（評価組織）との関係

(6) 低品質モジュールの構造的・時間的特性を解析・可視化する方式の開発

低品質モジュールがプログラム内にどのように分布し、開発作業の進行に伴ってどのように変化していくのか。その構造的・時間的特性を解析・可視化する方式を開発した。

具体的には、モジュール品質に関係すると考えられる属性13個について、その属性値（メトリクス値）の標準値域を、低品質なモジュール（欠陥ありモジュール）と高品質なモジュール（欠陥なしモジュール）それぞれについて明らかにし、ピラミッドグラフとして可視化した。

さらに、それら標準値域との比較に基づき、評価時点、および将来のある時点において低品質なモジュールとなるかどうかを推定・予測するためのモデルを構築した（図10）。

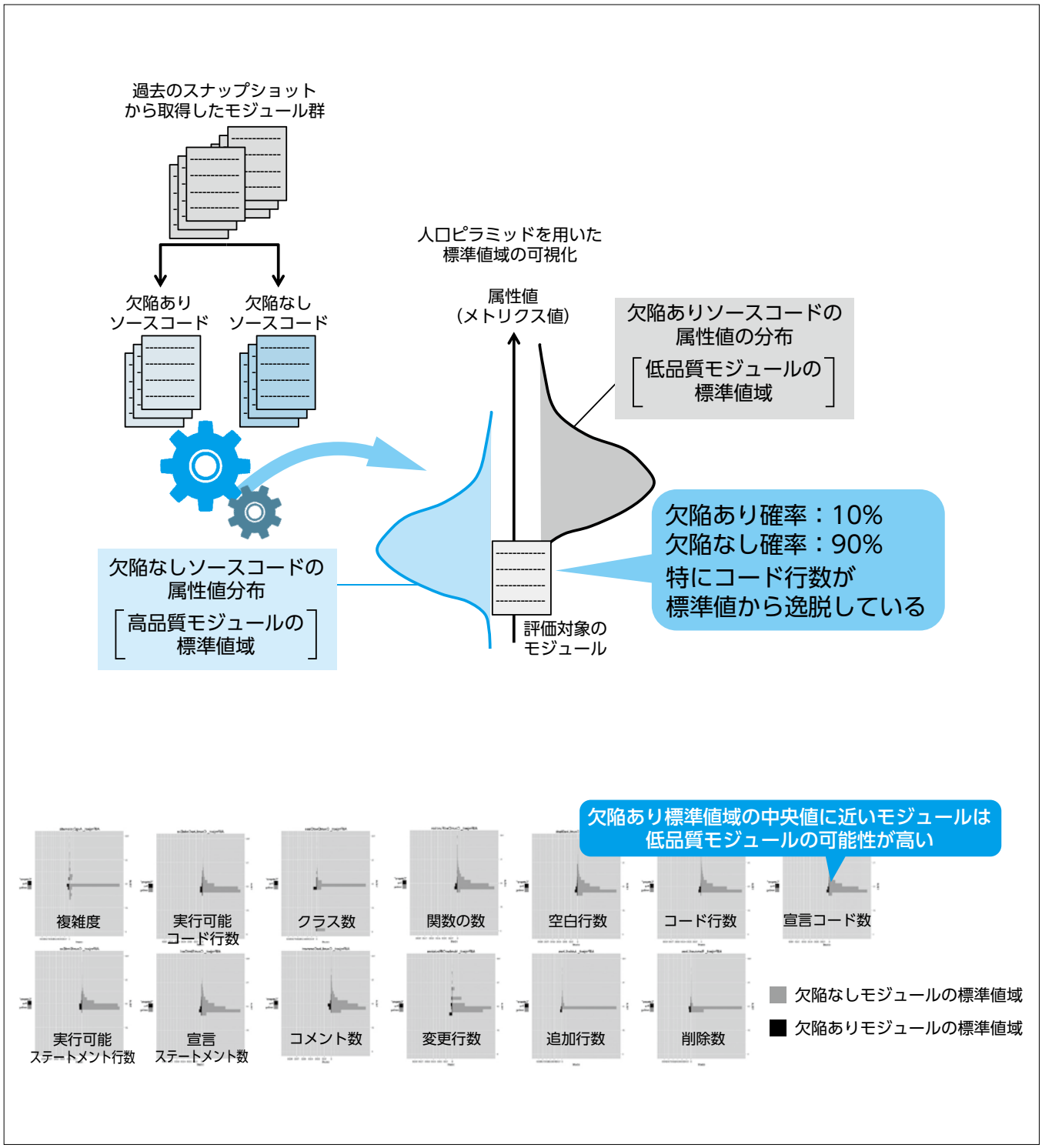


図10　オープンソースソフトウェアの開発データに基づいて算出したソースコード属性の標準値域

(7) ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価

ソフトウェアプロジェクトトモグラフィで定義される５つの視点（要件、作業、組織、プロダクト、課題）で、ソフトウェアプロジェクトデータを俯瞰し、注目する子要素へのズームインや親要素へのズームアウトなどを繰り返すことで、データの量的属性を再帰的、かつ探索的に解析・可視化する方式を開発した。俯瞰表示はTreemap形式で行った（図11）。ただし、偶発的発見を促すことを目的に、ズームインによって観点を自在に変更できる機能と、観点をランダムに変更できる機能を備えたユーザーインタフェースを採用した（図12）。

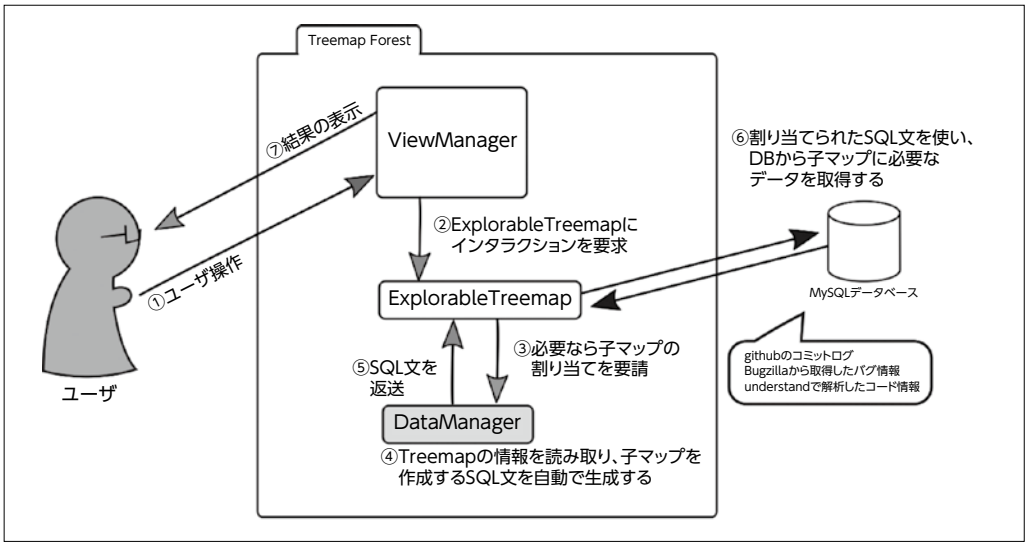


図11　Treemap Forestの動作モデル

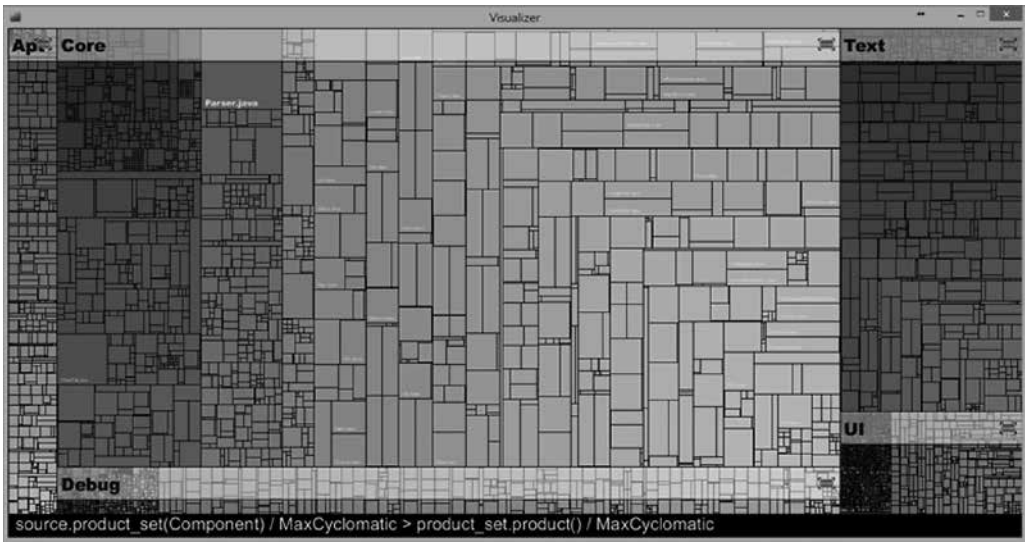


図12　表示例：観点「コンポーネント」→「ファイル」、「サイクロマティック複雑度」

3 産業界で研究成果が適用される場面と期待される効果

本研究の学術的なチャンレンジは、トピックモデリングや人口ピラミッドグラフの応用、クラッシュレポートの活用、機械学習による自動評価・マッピングなどである。また、産業界へのインパクトは、企業実務者へのインタビューなどに基づく技術利用シナリオの作成、企業における実証実験、開発した技術のツール化やウェブサービス化などである。

これら成果はソフトウェア開発プロジェクトの透明性を高め、ソフトウェア品質に関する「シグナリング」の手段を提供し、ベンダーとユーザー間の「情報の非対称性」を解消することになる。ソフトウェア品質の第三者評価制度といった「制度や組織」と相まって、市場をより健全化し、ベンダーには競争力を、ユーザーには安心感を与えることになる。

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

早稲田大学
グローバルソフトウェアエンジニアリング研究所 所長・教授 **鷲崎 弘宜**

1 背景と目的

ソフトウェア製品の開発側や運用側において、開発・保守・運用中あるいは運用検討中のソフトウェア製品の品質を、客観的、定量的かつ総合的に評価可能とし、評価結果を開発・保守における品質改善や取捨選択の判断材料に役立てることを目的とした。さらに開発側では当該製品の次の改訂（バージョンアップ）においても、品質測定評価結果を品質要求定義に利用できる。

2 概要

早稲田大学グローバルソフトウェアエンジニアリング研究所の研究チームが、全体統括、品質測定評価枠組みの確立、測定評価の一部、個々の品質の傾向と品質間の分析を実施した。一般社団法人コンピュータソフトウェア協会（CSAJ）が品質測定評価対象の製品募集等について支援した。品質測定評価の作業について部分的に外部評価機関に委託した。結果を以下に示す。

(1) 測定評価方法の確立

GQM（Goal, Question, Metric）法の適用を通じてISO/IEC 25000（SQuARE[1]）シリーズの測定法を具体化させて、内部・外部、利用時の品質についてそれぞれ66、17のメトリクスを定義し、測定に必要なデータを入力して測定値を得るための記入様式および集約方法を策定した。測定結果から評価を得るにあたっては、メトリクスの単位で測定値をパーセンタイルによりスコア化したうえで、スコアを品質特性・副特性単位で平均化する方式を実現した。概要を図1に示す。

内部品質について記入様式に加えてソースコードの解析結果を併用した。外部品質について、テストで発見さ

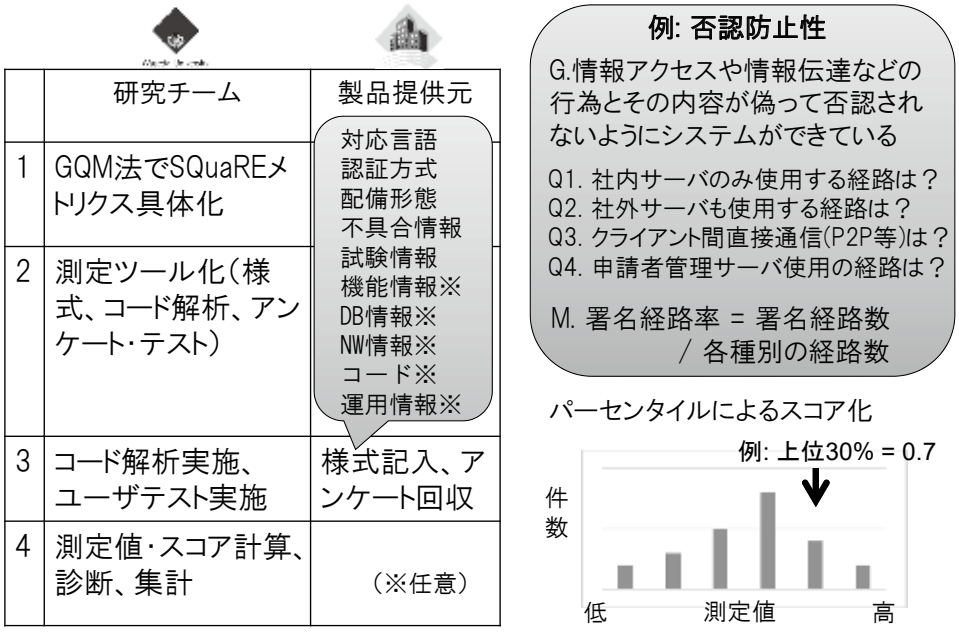


図1　品質測定評価の枠組みの概要

れる欠陥の発見時刻と数の関係を分析することで欠陥数を予測することに着目し、ソフトウェア信頼性モデルを用いてソフトウェア製品群の信頼性を評価した。分析の結果､リリースの前に予測欠陥数が安定な状態である（安定タイプ)、もしくは漸増的に増加する前兆がある（漸増タイプ)、または爆発的に増加する前兆がある（爆発タイプ）が存在することが分かった。利用時の品質について、利用時の利用者アンケートおよびユーザテストの実施結果を併用した。

(2) 個々の品質実態把握

PSQ 認証制度[2]を拡張した総合的品質測定評価制度を定義し、CSAJ の協力を経て、製品を提供する協力企業を募集し 21 の対象製品を決定した。続いて測定評価を実施し、統計処理と品質実態把握を実施した。特徴的な傾向を以下にまとめる（図２)。

- 互換性：スコアの高低において２極化している傾向にある。データ交換などの互換性に通じる仕組みを一部の製品において考慮していないことが原因として挙げられる。また、国際規格側において今後の実効性のあるメトリクスの拡充が望まれる。
- 使用性：測定評価できた製品群において使用性を十分に考慮できていない、あるいは、エンドユーザ対象ではないといったことから意図的に考慮していない製品が多い。
- 信頼性：大多数の製品において高い信頼性を作り込みの上でリリースされている。
- セキュリティ：スコアの高低について２極化している傾向にある。暗号化や破損防止などの高セキュリティ化に通じる仕組みを一部の製品において考慮していないことが原因として挙げられる。
- 保守性：保守性を十分に考慮できていない、あるいは、エンドユーザ対象ではないといったことから意図的に考慮していない製品が多い。
- 有効性：ユーザテスト実施時にタスク実行に難がありタスクを達成しにくい製品が一部見られた。
- 効率性：ユーザテスト実施時にタスク実行に難がありタスクの実施効率が低く時間を要する製品が一部見られた。

(3) 品質間の関係分析

21 製品の測定評価結果から、品質間の関係を分析した。さらに結果を関係モデルとしてまとめあげた。得られたデータの範囲において観察しうる事柄を以下にまとめた。

- 信頼性が高いほど保守性や移植性が高い傾向にあり、かつ統計的に有意と認められた。高信頼が求められる製品において、長期間における保守や様々な環境への移植や適合が求められるために正の相関が見られた可能性がある。

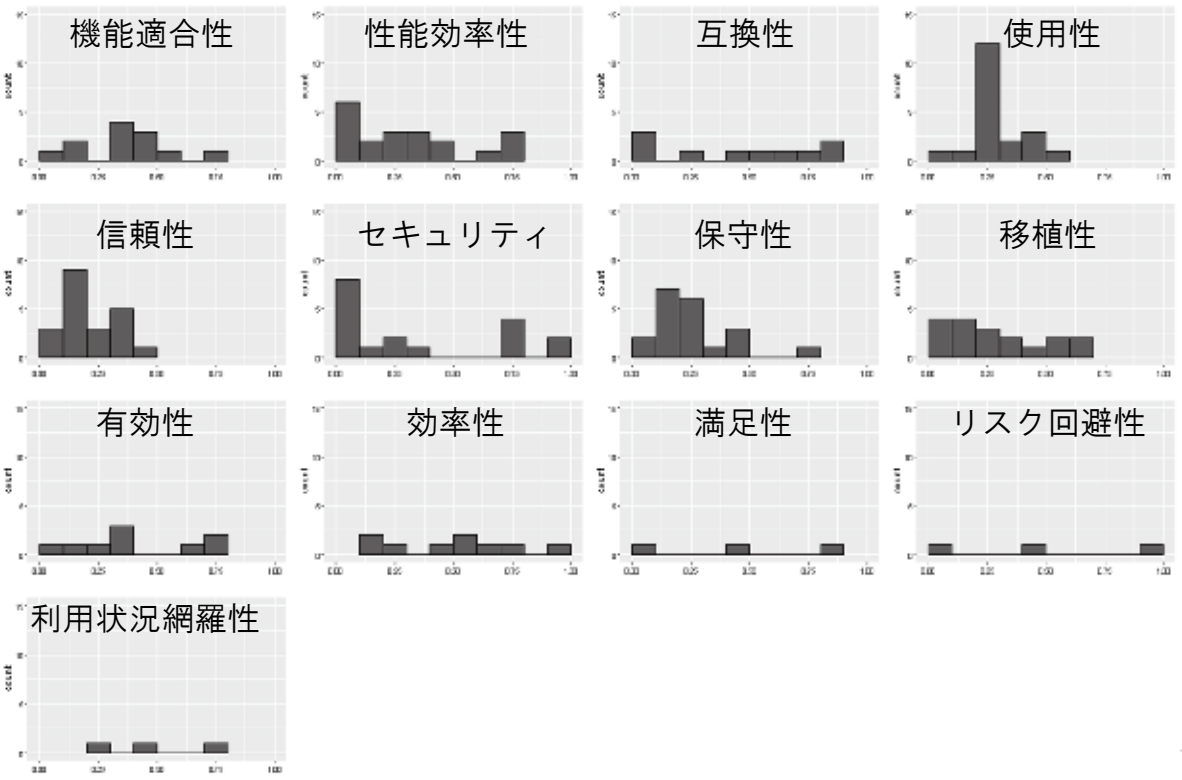


図２　品質特性ごとの傾向（製品数のヒストグラム）

- 移植性が高いほど、使用性や信頼性および効率性が高い傾向にあった。様々な環境に対する移植のしやすさを検討および作り込む過程において、様々な観点や側面から当該製品の品質を確認することとなり、結果として、あるいは、あわせて高い信頼性の作り込みにつながったということが推測できる。
- 機能適合性が高いほど、使用性が低い傾向にあった。計画通りの機能仕様の満足を最重視した結果として、顧客にとって本来重要な製品としての使いやすさを損なってしまったという副作用の可能性を推測できる。
- セキュリティが高いほど、有効性が高い傾向にあった。製品によってはセキュリティ関連の機能（ログインなど）が一定割合を占め、それらをユーザテストにおいて正確に実行できたことが幾らか影響している可能性がある。
- 信頼性タイプと品質特性の関係を分析し特定した。機能適合性、信頼性、有効性について安定タイプにおいて高品質であることがわかった。これは、機能適合性、信頼性、有効性が高いソフトウェア製品については、十分にテストされ欠陥を発見していると考えられる。性能効率性、互換性については爆発タイプにおいて低品質であった。

さらに、各製品のコンテキストと品質評価スコアとの関係を分析した。

- ドメイン別：エンドユーザ向けサービス製品においてセキュリティは極めて高く、数値計算シミュレーション製品においてセキュリティは低い傾向が見られた。今後の本格的な IoT 時代において考慮するように変革が求められてくると考えられる。別の要因として、国際規格 SQuaRE シリーズにおける定義から具体化できたメトリクス数が限られていたこともあげられる。
- 提供種別：パッケージ製品についてはクラウド製品に比べるとセキュリティが著しく低く、強化が課題ということができる。一方、クラウド製品についてはパッケージ製品に比べて移植性がやや低い結果となった。また、信頼性についてもクラウド製品はパッケージ製品に比べて低いが、これはメトリクスにおいてクラウド環境の考慮が不足している可能性がある。

なお、メトリクスや品質特性によっては対象製品数や取得できたデータが少なく統計的に母数として十分な数量とは言い難いものもあることに留意が必要である。

(4) 全体パッケージ化

品質測定評価枠組みと測定評価結果のデータセットを WSQB17: Waseda Software Quality Benchmark として以下に公開した。http://www.washi.cs.waseda.ac.jp/?page_id=3479

3 産業界で研究成果が適用される場面と期待される効果

研究成果の産業界における活用の場面を含めて以下に提言する。

- IoT/IoE 時代に最重要となるはずの品質であるセキュリティおよび 互換性が一部低いパッケージ製品が見られた。ソフトウェアの開発プロセスや方法、取り巻く環境が変革されつつある中で、ソフトウェア品質に対する意識の変革も必要である。
- 機能適合性の作り込みの結果として、本来のユーザにとって重要な使用性が損なわれている可能性がある。製品の価値を判断する立場は本来ユーザであり、様々な品質特性を多面的に考慮するユーザ中心の取組み（例えばユーザ中心の設計など）が今後は求められる。
- 各品質特性単位でみると実際に測定評価できた製品数は半数に満たないものがほとんどであった。多面的な品質測定評価を可能とするためのデータの記録、ならびに、本調査研究の成果としえられたベンチマークを参照したうえでの目標値設定が重要である。

以下を国際標準化団体、特に国際標準化機構 ISO/IEC　JTC1/SC7/WG6 への提言とする。

- SQuaRE シリーズにおける品質測定方法の多くは抽象度が高く定義のままでは適用困難であることが判明した。そこで、本調査研究において実効性のある形で具体化に成功した品質測定評価の枠組みを組み入れて実効性を強化すべきである。
- SQuaRE シリーズにおける品質測定方法の多くは 1990 年代までのソフトウェア製品の形態および開発方法を念頭においており、アジャイル開発やプラットフォームとしてのクラウドに対する考慮を幾らか欠いており、今後の対応強化が求められる。

1 システムおよびソフトウェアの多岐にわたるステークホルダが持つ多様な品質要求を定義し、その実装を評価するための共通の考え方を示す基準のひとつであり、国際規格。

2 ソフトウェア製品の説明資料とソフトウェアの機能が一致していることを確認・認証する、国際規格に準拠した認証プログラムで一般社団法人コンピュータソフトウェア協会が実施している。

2014年度 保証ケース

オープンシステム・ディペンダビリティのための形式アシュランスケース・フレームワーク

神奈川大学
理学部 情報科学科 教授 木下 佳樹

1 背景と目的

システムの目的、目標、環境および性能の変化に適応する。説明責任を絶え間なく達成する。想定されるサービスを、要求された時に要求どおり提供する。こうした能力は「オープンシステム・ディペンダビリティ」と呼ばれている。

SOS(System Of Systems)やIoT(Internet of Things)に関するディペンダビリティ達成のためには、オープンシステム・ディペンダビリティ達成が必要条件だとされている。また、防災計画やセキュリティ対策では想定外の災害や攻撃への対処が重要であるが、オープンシステム・ディペンダビリティ達成のためには、どこまでを想定するのかを明確にすることが求められる。

アシュランスケース(Assurance Case)は、具体的なシステムの安心・安全に関する議論の記録文書であり、システムの利害関係者間の合意事項の記録、契約文書や認証における提出文書、あるいは事故調査委員会の資料などとして、近年急速に注目され始めた。プラントや軍事技術など高度な安全性が要求される、いわゆるSafety Critical Systemに関する安全性議論に用いられることから始まった。現在では車載、鉄道、航空、医療システムの認証に関する国際標準においてアシュランスケースの提出が要請されているなど、その需要は増加傾向にある。

一般にアシュランスケースは大部な文書となるが、一方で、些細な論理的瑕疵が文書全体を無意味なものにする可能性がある。アシュランスケースにとって、論理的な整合性の担保だけでは十分ではないが、必要である。しかし、膨大な記述の詳細にわたって厳密な検査を行うのは容易ではないことから、機械的検査の方法が求められている。

2 概要

本研究では、システムがオープンシステム・ディペンダビリティを達成していることを議論する形式アシュランスケースを書くためのフレームワーク「Formal assurance case Framework for Open systems dependability (FFO)」を作成した。

また、この作成過程で得られた知見をもとに、システムがオープンシステム・ディペンダビリティを達成するためのシステムライフサイクルプロセスへの要件を考察し、国際標準IEC 62853 Open systems dependability策定の技術的根拠を与えた。次に、具体的な技術領域として車載システムおよび地域防災計画を選び、FFOにそれぞれの領域知識を具備したフレームワークを試作した。

FFOの開発

FFOの機能のうち、対象領域によらず共通して必要となるものを、プログラミング言語Agdaによって実装した。議論の最上位ゴールは、「変化し続けるシステムのサービス継続と説明責任のまっとう」である。また、FFOが依拠するシステムライフサイクルと、それを対象とするアシュランスケースの構造を、国際標準体系(IEC 62853およびISO/IEC 15288及び周辺の標準)の中に位置づけた。さらに、特定の技術領域として車載システムと防災システムを取り上げ、それぞれの技術領域にFFOを具体化したものを作成した。

車載システムでは一般社団法人JASPAR(Japan Automotive Software Platform and Architecture)が提供する機能安全テンプレートを基に、アシュランスケースのための語彙定義(オントロジー)を作成し、システムのオープンシステム・ディペンダビリティ達成を主張するアシュランスケースのための議論構造(議論フレームワーク)を、Agda言語によって記述した。

防災システムでは、地域防災計画をもとに防災FFOを作成し、防災システムのオープンシステム・ディペンダビリティを主張する形式アシュランスケースを記述した。

FFOの有効性評価

車載システム、および防災システムの事例に対して作成し

た、対象技術領域に具体化したFFOの有効性評価を試みた。

車載システムでは自動車部品メーカーからの協力を得て、機能安全仕様から技術安全仕様を導く工程（機能安全/技術安全phase）をもとにアシュランスケース記述実験を行った。その結果、我々のフレームワークが有効であるとの評価を受けることができた。

防災システムでは、神奈川県平塚市の協力を得ることができ、同市の地域防災計画に規定されている防災業務を対象として防災FFOを適用したアシュランスケース記述実験を行った。その結果、地域防災計画の記述において、必ずしも明らかではない業務の主体（Who）などを明確にすることは有効であった。業務の階層構造を明確にすることで、地域防災計画に記述すべき部分と、その下位文書の各マニュアルに記述すべき部分との区別が明確になるなどの評価が得られた。

FFOが依拠するシステムライフサイクル概念の確立

システムライフサイクルに関連する既存の国際標準における用語定義を比較対照した上で、これらに矛盾しない形でオープンシステム・ディペンダビリティを達成するライフサイクルモデルとして提供した。具体的には、DEOS（Dependability Engineering for Open Systems）プロセスのグラフを解釈して、意図する遷移系に対してペトリネットを用いて導き出す方法を考案した。

ペトリネットによる定式化に加え、DEOSプロセスの各ステージと、ISO国際標準体系においてシステムライフサイクルプロセスに関する最上位標準と位置付けられている「ISO/IEC/IEEE 15288」が定義するシステムライフサイクルプロセスとの関連を明示したライフサイクルモデルを、DEOSライフサイクルモデルとして国際標準IEC62853策定の委員会に提出した（図1）。

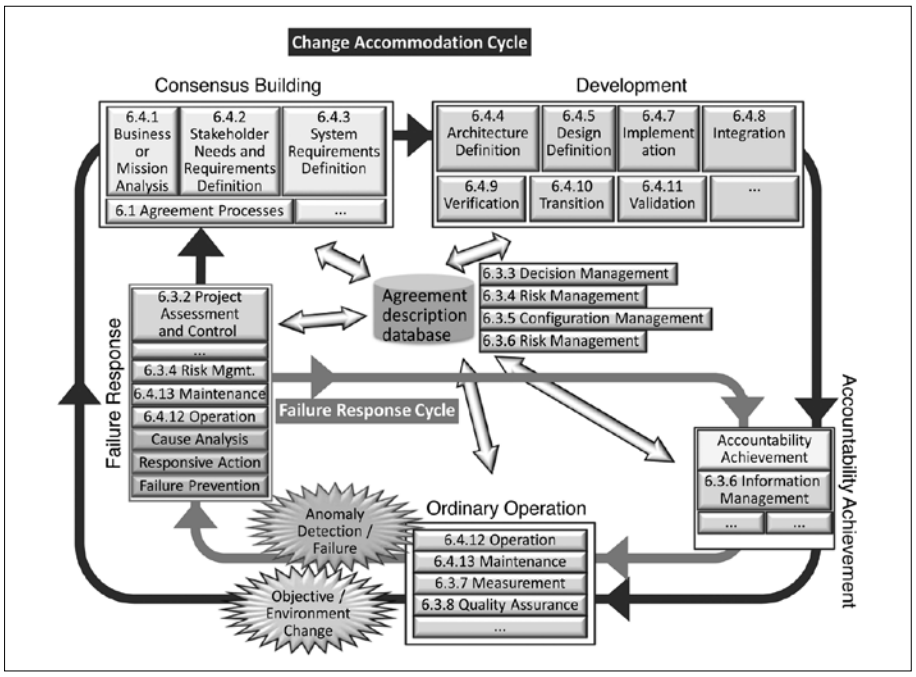


図1　DEOSライフサイクルモデル表現

3 産業界で研究成果が適用される場面と期待される効果

本研究では、オープンシステム・ディペンダビリティ達成の形式アシュランスケースを書くためのツールFFOを開発した。また、その過程で、オープンシステム・ディペンダビリティ達成のための要件を精査して、2017年発行予定の国際標準「IEC 62853」とした。

アシュランスケースの技術が広まっていない現状では、「IEC 62853」の策定が、本研究の成果のなかでもっともはじめに寄与するものと考えられる。この標準が利用されることにより、適合性のために要求されているアシュランスケースがより広く使われるようになれば、アシュランスケースの執筆や認証の効率化が課題として認識され、本研究の本来の目的である形式アシュランスケースの価値が産業界の中でも広く理解されるようになる。車載システムの機能安全に関する「ISO26262」が果たした役割を、一般的システムのオープンシステム・ディペンダビリティに関して「IEC62853」が担うものと考えられる。電気・電子機器の機能安全に関するIEC 61508は、アシュランスケースを要求しているわけではないが、一般のシステムのドキュメンテーション一般について、これと類似の役割を果たした。

2015年度 保証ケース

保証ケース作成支援方式の研究

名古屋大学
大学院 情報科学研究科 教授 山本 修一郎

1 背景と目的

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、保証ケースの作成が必要になっている。たとえば、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。

保証ケースは、開発対象システムが安全性や高信頼性を持つことを論理的に保証するために、用いられる構造的な文書である。保証ケースは、①システムが特性を持つことについての主張②上位の主張を下位の主張で説明するための分解③説明のための前提④主張の根拠となる証拠から構成される。保証ケースの産業界への適用については、様々な課題がある。

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。

これらのモデルには多様な形式がある。各モデルに対して保証ケース作成手法を用意できないと、産業界への保証ケースの適用を進展させることが困難になる。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手法については、明らかになっていない。

ソフトウェアコンポーネント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向上が期待できる。再利用対象コンポーネントの安全性を保証するための保証ケースの作成にあたってコンポーネントのモデルがない場合、コンポーネントのコードに対する保証ケースが必要である。現状では、コードに対する保証ケース作成手法は確立されていない。

作成された保証ケースの妥当性を確認するためには、保証ケースを適切にレビューする必要がある。保証ケースの具体的なレビュー手法については、様々な手法や定義があるが、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手法は確立されていない。

2 概要

モデルに基づく保証ケースの統一的作成法

現行のシステム開発では、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。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した（図1）。

① モデルに基づく保証ケースの統一的作成法を、アルゴリズムで定式化した。このアルゴリズムでは、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。

② アルゴリズムに基づいて保証ケース作成支援ツールを試作した。保証ケース作成支援ツールでは、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。ツールを用いることにより、手動でモデルから保証ケースを作成する場合に比べて、生産性を約5,6倍に向上できた。

③ 保証ケースを統一的に作成するために必要となる知識およびそれらを説明できるスキルを身につけさせる研修教材を開発した（表1）。

表1　統一的保証ケース作成手法　研修教材　カリキュラム

第1章	保証ケースを統一的に作成するための基礎知識
1	システムの構成
2	システムのリスク
3	システムの特性
4	保証ケースの表記法
5	主張の分解
6	リスク対策の証拠
第2章	保証ケースの統一作成手法の知識
1	モデルの定義
2	主張の分解
3	主張の階層的分解
4	分解の網羅性
5	主張の優先順位
6	統一的な保証ケース
第3章	保証ケースによる合意形成
1	議論の合意形成

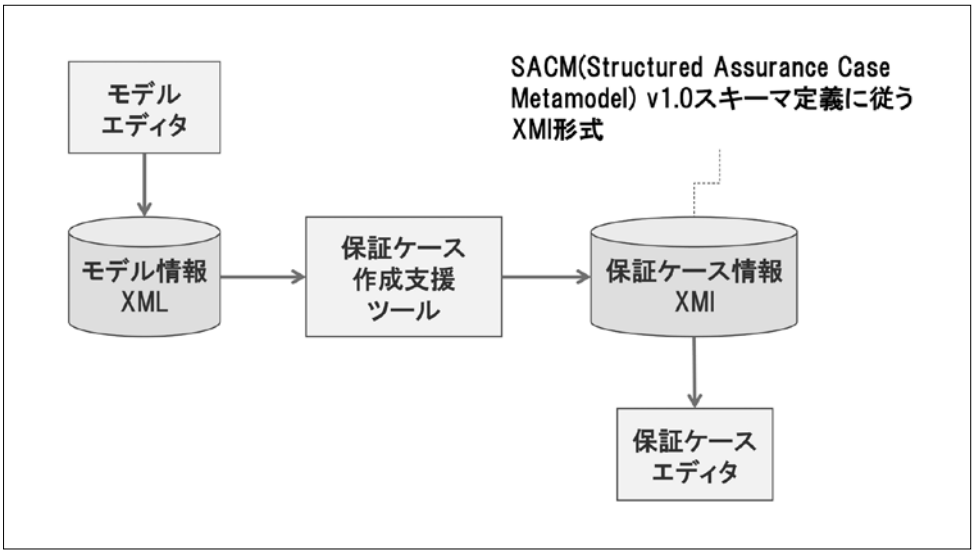


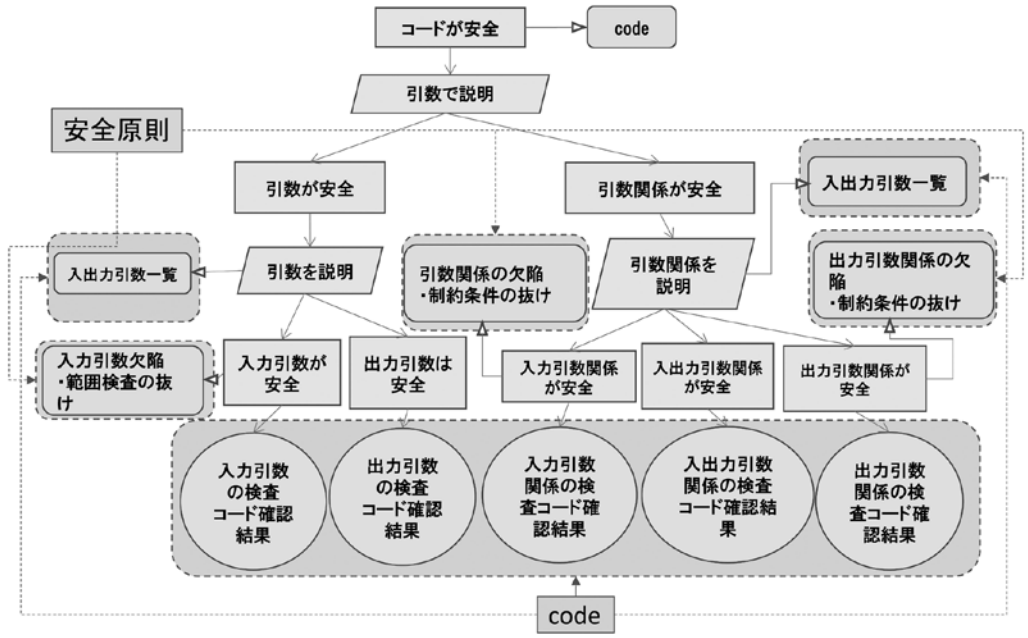
図１　保証ケース作成支援ツールの概要

コンポーネントに対する保証ケース作成法

モデルがあればパターン分解による保証ケースの作成が可能であるが、既存システムを保証しようとしてもモデルが定義されていないことが多いという問題がある。そこで、モデルからではなく、コードに対する保証ケースの作成法を開発した（図２）。

① コンポーネントコードに対する保証ケース作成手法DDBE（Defect Detection By Evidence：証拠による欠陥摘出）を定式化した。

② 保証ケースと対応するリポジトリのメタモデルを、前提、主張、具体化すべき証拠、証拠、識別子、式、ブロックによって具体化した。



【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

図２　保証ケースに基づいてコードの欠陥を摘出する手順と考え方

保証ケースの客観的なレビュー手法

既存の保証ケースをレビューする場合、客観的なレビュー手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。特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。そこで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した。

① 保証ケースが対象とするシステムの成果物や品質特性、リスクなどの構成情報に基づいて、システミグラムを作成することで、より詳細で客観的なレビューができることを明らかにした。

② この結果から、保証ケースが対象とする構成情報をシステミグラムにより図式化するレビュー手法を考案した(図3)。

③ 完全性、明確性、適切性、追跡性からなるレビュー観点に基づき、レビュー指標を具体化した(表2)。

④ 保証ケースを客観的にレビューするために必要とする知識およびそれを説明できるスキルを身につけさせる研修教材を開発した(表3)。

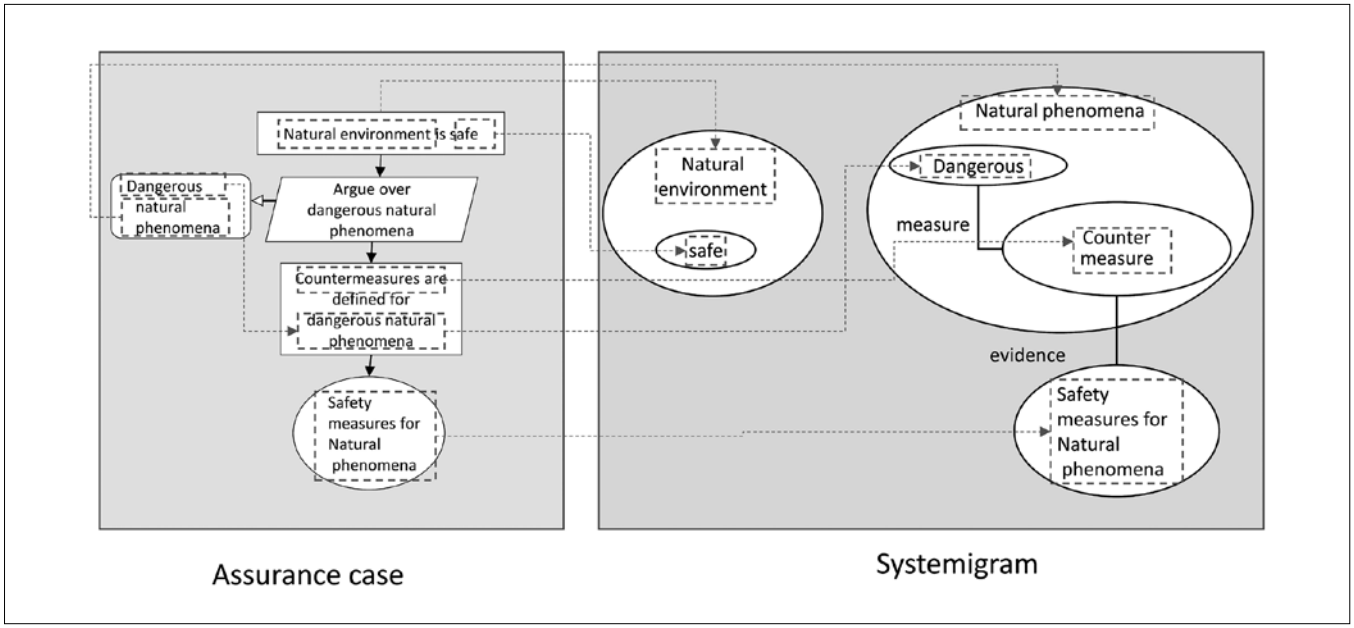


図3　保証ケース(Assurance case)とシステミグラム(systemigram)の対応例

表2　保証ケースレビュー指標

観点	活動	完了基準	完了指標
内容理解	対象とする保証ケースのノードに基づいてシステミグラムを作成する	保証ケースのすべてのノードについて、システミグラムを作成していること	保証ケースのノードに対するシステミグラムの作成完了率
問題識別	システミグラムのノードに対して、必要な項目が含まれていること(完全性)、あいまいさがないこと(明確性)、不必要な項目が含まれていないこと(適切性)、根拠が明確であること(追跡性)を確認する。	システミグラムのすべてのノードについて、完全性、明確性、適切性、追跡性を確認して摘出したすべての問題を識別していること	保証ケースのノードに対する問題確認の完了率
原因究明	識別した問題について、原因を究明する	識別したすべての問題の原因を究明していること	保証ケースの問題に対する原因究明の完了率
修　正	原因が究明された問題ごとに、保証ケースを修正する	すべての問題について、保証ケースを修正していること	保証ケースの問題に対するシステミグラムの修正完了率

表３　保証ケースレビュー手法　研修教材　カリキュラム

第１章	保証ケースをレビューするための基礎知識
1	システム要素の相互関係
2	保証ケースの表記法
3	主張の問題点
4	分解の問題点
5	網羅的なレビュー
第２章	保証ケースをレビューするための知識・スキル
1	システミグラムの表記法
2	システミグラムで主張
3	システミグラムで分解
4	システミグラムで証拠を表現
5	保証ケースのレビュー
6	保証ケースのレビュー指標
7	個人レビュー
第３章	保証ケースによる合意形成
1	グループレビュー

保証ケース手法の実践的導入適合性

保証ケース手法を導入する上で、開発組織が必要な準備能力を備えていることを確認するために、活用ビジョン、活用コミュニケーション、プロダクトデザイン、プロセスデザイン、投資適正、人材開発について37項目を２段階（あり・なし）で評価できる定性評価指標（定性評価37項目版）を作成した。さらに、客観的に導入準備能力を評価するために、保証ケース導入準備能力評価指標（客観評価50項目版）を作成した（表４）。この保証ケース導入準備能力評価指標を新技術の組織への導入準備能力を測定するために適用できるように一般化した新技術導入準備能力評価指標も作成した。

表４　保証ケース導入準備能力評価指標（客観評価50項目版）

能力	評価指標
保証ケース構築（7）	①保証原則の定義 ②保証の根拠証拠の管理 ③保証対象の明確な定義 ④保証すべき主張の明確な定義 ⑤主張間の優先順位が明確 ⑥説明責任部門が明確 ⑦コンプライアンス課題の認識
リスク分析（8）	①保証の欠落がもたらす開発業務への影響を識別 ②リスク管理原則を定義 ③リスク管理計画を定義 ④リスク管理手順を定義 ⑤リスク管理情報を共有 ⑥リスクを評価 ⑦問題情報を共有 ⑧リスク対応手段を定義
保証ケース 活用ビジョン構築（7）	①自社戦略目標とACの役割が明確 ②ACが役割を果たすための組織を制度化 ③AC投資を重点化 ④開発でのACの活用方針を明確化 ⑤AC部門の役割が明確 ⑥AC部門と開発部門の役割が明確 ⑦ACに基づく開発部門の結果責任が明確
保証ケース活用 コミュニケーション（7）	①ACの役割を社員が共有 ②ACの活用方針を社員が共有 ③AC導入目的を開発部門が理解 ④AC導入後の業務変化を開発部門が理解 ⑤部門間でACによる問題解決プロセスが定義 ⑥AC活用事例を社内で共有する仕組みを定義 ⑦経営層、AC部門、開発部門の３部門間で、ACの投資対効果を共有
プロダクト デザイン（5）	①成果物に対する保証品質を定義 ②成果物に対するあるべきAC条件を定義 ③成果物に対するACの活用方策を標準化 ④社内外の開発業務連携の観点で成果物に対するACを標準化 ⑤成果物に対する重複のないACを定義
プロセス デザイン（5）	①開発プロセスの保証計画を定義 ②ACによる開発プロセスを定義 ③開発プロセスのAC活用方策を標準化 ④社内外の業務連携プロセスをACで標準化 ⑤ACの重複のない開発プロセスを実現
保証ケース 投資適正化（6）	①AC資産の構築経費を配分 ②AC部門の独立性を考慮 ③AC導入経費対効果を事前に検証 ④AC導入時に全社最適への適合性を検討 ⑤AC導入後に活用状況・効果を測定 ⑥AC活用問題をAC導入検討時に解決
システム保証 人材開発（5）	①ACを活用した開発プロセス改革の提案人材を育成 ②経営層の身近に開発とACの双方に精通した人材を配置 ③AC人材が経営に関する知識を習得する機会を提供 ④AC人材が現場の開発プロセスを理解する機会を提供 ⑤開発人材にACの活用スキル研修を提供

AC：Assurance Case（保証ケース）

3 産業界で研究成果が適用される場面と期待される効果

モデルに基づく保証ケースの統一的作成法

保証ケース作成支援ツールを用いた統一的作成法は、アーキテクチャ品質評価サービスなどに展開できる。具体的には、システム開発者に対して、開発対象システムのアーキテクチャ品質を評価しようとする場面で、保証ケースを開発対象システムのモデルから作成することにより、システムが所望の品質を満たすことを保証する作業に役立つ。

また、対象物の見方を整理する方法が客観的ではないだけでなく、人ごとに異なる事があるが、統一的保証ケースを用いれば、論理的に説明することが可能である。

コンポーネントに対する保証ケース作成法

コードに基づく保証ケース作成法は、コードレビュー手法などに展開できる。具体的には、システム開発者に対して、開発対象システムのコードの品質を評価しようとする場面で、コードが実現すべき仕様から保証ケースを作成することにより、コードが所望の機能目標を満たすことを確認するレビュー作業の効率化に役立つ。

また、オープンソースをビジネスで使うにはリスクが伴うが、何を確認すべきかが分かれば、本手法で確実にオープンソースのコードを保証できる。

保証ケースの客観的なレビュー手法

保証ケースレビュー手法はSPRME（対象、特性、リスク、対策、証拠）分析に基づく保証ケース作成法（SPRME法）、統一的作成法との統合化手法、保証ケースの全体理解法に展開できる。具体的には、システム分析評価者に対して、開発対象システムの品質を評価しようとする場面で、対象、特性、リスク、対策、証拠を対応付けて明確化することにより、評価すべきシステムに対する保証ケースを系統的に作成することができるので、対象システムが所望の特性を満たすことを保証する作業の効率化に役立つ。

実践的保証ケース教材

保証ケースの基礎知識を持つ学習者に対して、より高度な保証ケースの知識を提供しようとする場合、統一的保証ケース作成法ならびに保証ケースレビュー手法の教材を用いることで、保証ケースの発展的な知識を身につけさせることができる。

保証ケース手法の実践的導入適合性

保証ケース導入準備能力評価指標は、指標を用いた保証ケース導入法、形式手法などの新技術の組織への導入能力評価手法、アーキテクチャ品質評価サービスの設計法として展開できる。具体的には、保証ケースを導入しようとしている場合、その組織の保証ケース導入担当者が、組織への導入を成功させるために必要な能力には何があり、どの能力を向上させるべきかを知ることができる。

■名古屋大学 山本研究室（保証ケース作成支援方式の研究成果）

URL http://de-science.icts.nagoya-u.ac.jp/download.html

2015年度 保証ケース

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

電気通信大学
大学院 情報システム学研究科 教授 田中 健次

1 背景と目的

人間系（コミュニティ）と機械系（センシング）とが複雑に絡み合うスマートコミュニティでは、ユーザからのリアルタイムセンシング情報を統合的に共有・分析・処理、あるいはSNS上で議論・意思決定することで新たな価値が生まれる。こうした利便性の一方、事故や事件などのインシデントが起きた際の社会的影響（リスク）の大きさについても十分な議論が求められる。そこではリスクを的確に可視化し、提供側のみならず利用者側も含めたすべての利害関係者間で適切な社会合意形成がなされることが望ましい。

他方、要求工学などのソフトウェア工学分野においては、ゴール指向のディペンダビリティ合意形成手法として議論構造を合理的に可視化するD-Case（アシュアランスケース）が実用化されつつある。論理構造（ゴール、前提、議論展開、モニタリングおよびエビデンス）がD-Caseで表現されると、利害関係者は、その妥当性を議論し合意を形成することができる。そこでD-Caseを、スマートコミュニティでの議論構造可視化へ適用し、実際に利害関係者間がツールにより協働すれば、スマートコミュニティの合意形成の効率化が期待できる。

本研究では、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）のディペンダビリティ合意形成を支援する手法を開発すると共に、その有効性を実証するため、次の3つを目的としている。

① D-Caseを利用した合意形成支援ツールの開発
②ポケットガイガー[1]を利用した実証実験
③モデル化と研究成果オープン化

2 概要

本研究では3つの合意形成支援ツール（SmartStructure、CrowdTalks、CrowdTalks+）を開発し、それらの有効活用を実証するために図1に示す3つの実験を行った。

プレ実験では、「D-Case（開発ツール「SmartStructure」を利用）」「Web」「新聞記事」の3種類の情報提示により、被験者グループでの合意形成タスクの実施能力がどのように変化するかを確かめた。続くLab実験では、正解のないグレイゾーンの2テーマ（処分場を受け入れるか、原発を再稼働するか）について、開発ツール「CrowdTalks/+」により被験者自らがD-Caseを作成・議論しながら合意形成を行った。これらの結果、図2に示すように、D-Caseは「使い慣れたWeb/新聞記事と同等の自由度」であり、「合理的・科学的な議論に貢献」できることがわかった。これにより、D-Caseがグレイゾーンでの自由な合意形成において有用であると確認できた。

SNS実験ではポケットガイガーFacebookグループの中から参加者を募り、「空間放射線の測定方法や測定値の共有で、一般市民が気を付けるべきこと（市民ルール）を皆で考えよう」というテーマについて、開発ツール「CrowdTalks/+」を使って作成したD-Caseにより議論をファシリテートしながら合意形成を実証した。この実験では一般市民のみならず専門家も集まり、自律的にコミュニティ内でのオピニオンリーダーが形成された。

実験種別	プレ実験	本実験	
		Lab実験	SNS実験
被験者	一般市民（学生） N=23	一般市民（学生） N=16	一般市民（ポケガ利用者） ＋専門家、技術者　N=94
DCASEの提示方法	予め作成されたD-CASEを自動提示	開発ツールでD-CASEを作成しながら議論	ファシリテーターがD-CASEを作成しながら議論
議論内容	正解のある問題 （例：測定方法、ヨウ素剤の使い方）	グレイゾーン （例：原発再稼働、中間処理施設受入）	グレイゾーン （市民放射線測定の「ルール」づくり）
議論方法	相対	相対	Facebook
開発・使用したツール	Smart Structure: D-CASEの提示 	Crowd Talks/+: Web上でのD-CASE作成・SNS分析 	
実験結果、DCASEの効果	論理的・科学的議論に有効だが議論の自由度に課題あり	論理的・科学的議論に有効であり、自由な議論を阻害しない	科学的議論におけるファシリテーションツールとして有効
実験時期	2015/12	2016/8	2016/11-2017/1
実験風景			

図１　実証実験と開発ツールの概要

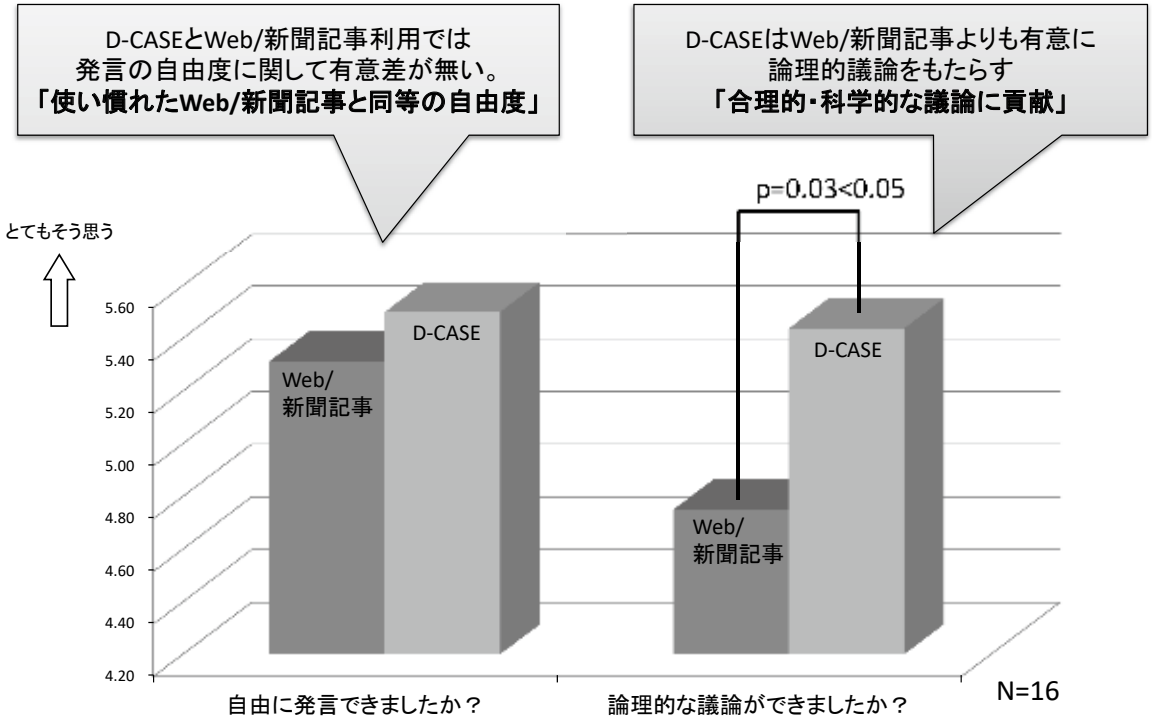


図２　質問紙調査の結果（抜粋）

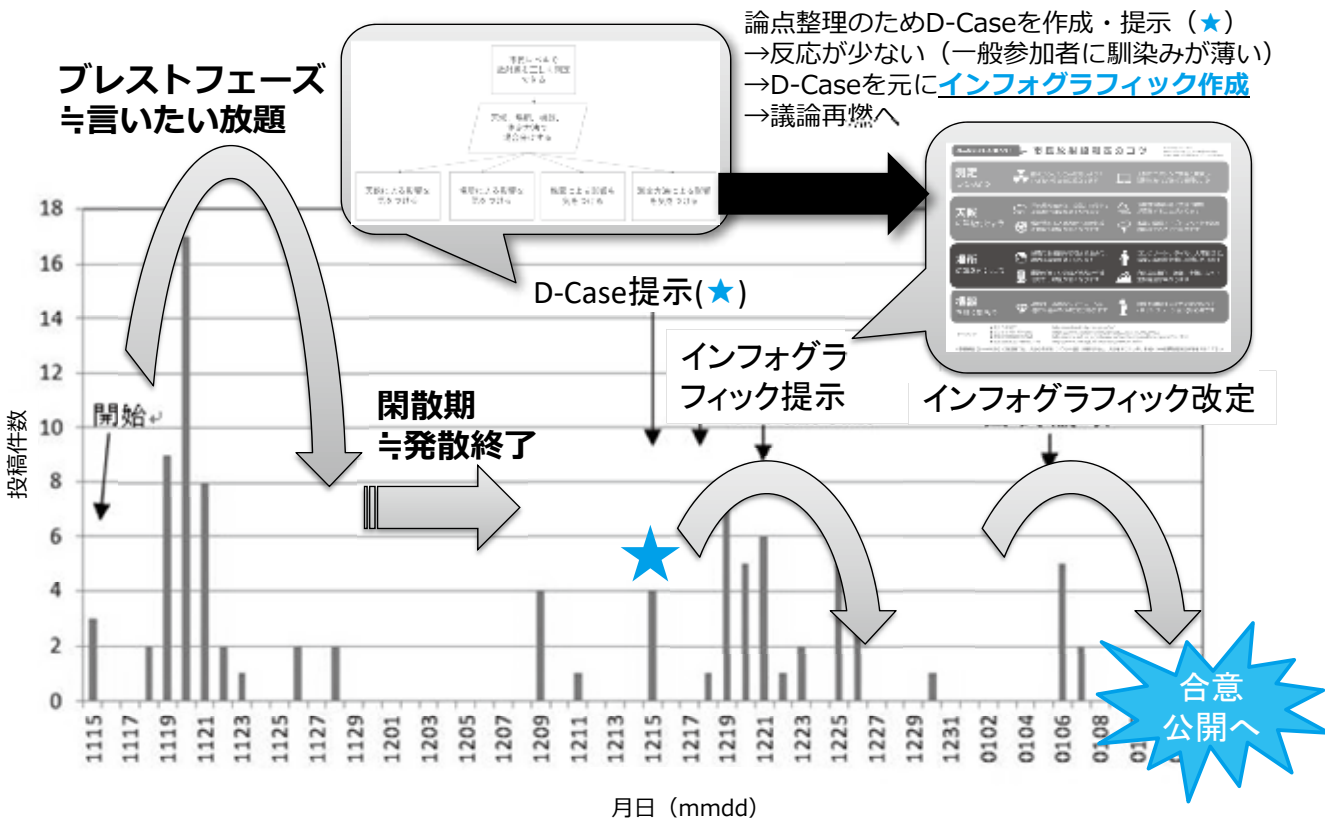


図３　SNS 実験におけるファシリテーションの変遷

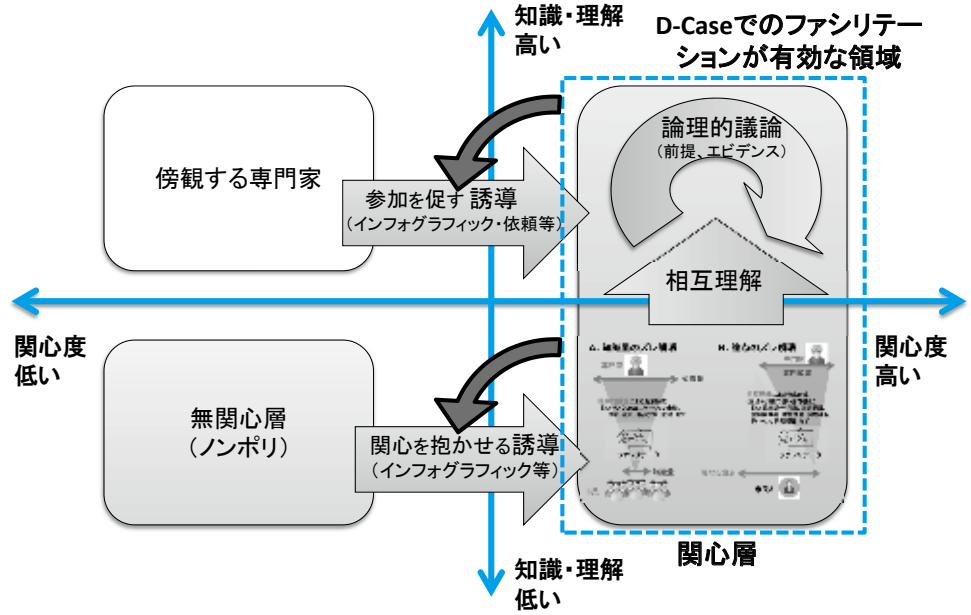
図３に SNS 実験全体の発言数の変遷を示す。当初のブレストフェーズが収束した後に議論内容を集約し D-Case を提示したものの、それだけでは議論の広がりが限定的であったため、D-Case の内容を見やすいインフォグラフィックに置き換えて提示したところ議論が再燃し、様々なフィードバックとして得られた。以上により、ファシリテーションツールとしての D-Case の有効性を実証でき、参加者の知識量・関心度（積極性）に応じた情報提示方法の工夫が必要であることが確認された。

3　産業界で研究成果が適用される場面と期待される効果

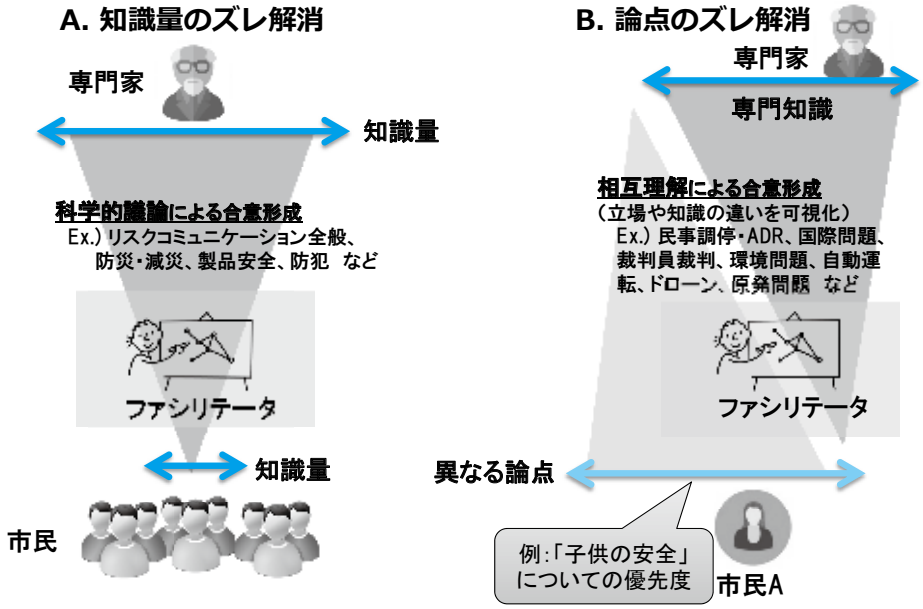
産業界への適用に向け、D-Case による合意形成支援（社会実装）の方向性を示すため３つの一般化モデルを提唱した。D-Case 合意形成ゾーニング（図４上）は、D-Case でのファシリテーションが有効な関心層のゾーン（第一象限と第四象限）及び、傍観する専門家や無関心層を引きつけるためにインフォグラフィック等の利活用が有効であることを表している。

関心層の中で、いかに相互の理解を高めるかについて考察したものが D-Case 相互理解インタラクション（図４中）である。実験により、D-Case を使用した場合の相互理解に及ぼす効果について、知識量と論点という２つの異なる側面からまとめており、特に一般市民と専門家の間でのコミュニケーション上のズレを低減させる効果があると考えられる。

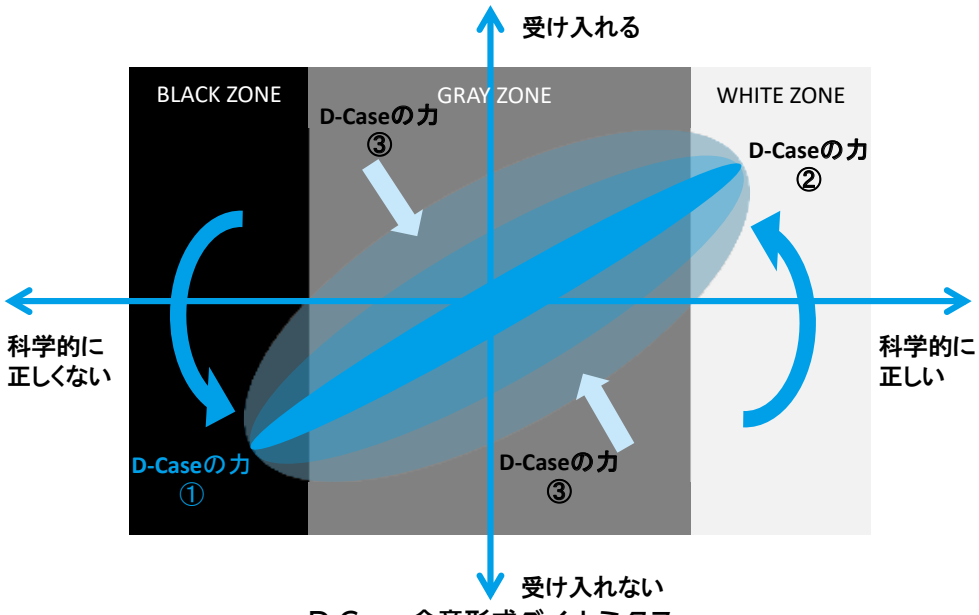
D-Case 合意形成ダイナミクス（図４下）では、D-Case 合意形成ゾーニングと、それに内包される D-Case 相互理解インタラクションの作用によって、合意形成がどのように方向付けをされるかをモデル化した。プレ実験で見てきた D-Case による科学的議論の促進は、WHITE ZONE においては議論が概ね受け入れられ（D-Case の力②）、逆に BLACK ZONE においては議論が受け入れられないような状況（D-Case の力①）を意味する。さらに合理的・科学的な議論がなされれば、GRAY ZONE の中でも、より WHITE に近い部分については受け入れられる傾向が多くなり、より BLACK に近い部分については受け入れられない傾向が強くなる事が望ましい。こうした関係性を示したのが、ラグビーボール状の楕円エリアである（D-Case の力③）。議論のプロセスを通じて、個々人の意見が徐々に上記のようなリニアな関係性に近づいて行くことが合意形成へのステップとなるだろう。



D-Case 合意形成ゾーニング



D-Case 相互理解インタラクション



D-Case 合意形成ダイナミクス

図 4　社会実装のための提案モデル

本研究成果の産業界での具体的な適用場面として、組織内外での連携・合意形成が必要とされる部門（例：オープンイノベーション、カスタマーリレーション、製品安全等）での利用が挙げられる。また、公共分野での利用シーンとして災害リスクコミュニケーションの他、移民受入、気候変動、自動運転など制度設計における合意形成での利用も想定できる。そこではD-Caseによる科学的知識の獲得や、専門家と市民、あるいは政治家の間での論点のズレの解消にも有効に機能するだろう。近年の公共事業はPFI（Private Finance Initiative）やPPP（Public-Private Partnership）の方式が多くなっており、行政と民間の境目が融合した中で適切な各種アセスメントを市民・民間・行政で共に行うための合意形成支援ツールとしての応用も期待される。

もちろんD-Caseがあれば、自動的に合意形成が促進されるわけではない。そこでは、ファシリテータの育成に加え、適切な支援（技術的、金銭的、社会的）も必要とされる。また良質なファシリテータ養成のための教育環境や認定制度の整備、D-Caseそのものの認知度向上や普及活動、さらなるモデル実証を通じた、産業・公共の各分野固有の課題へのローカライズ、諸外国での事例を増やし相互に情報交換するための国際標準化等が課題となるだろう。

1 放射線監視のための国内最大規模のスマートセンシングコミュニティ（http://www.radiation-watch.org）であり、①スマートフォン接続型の安価な線量計（1,850円～）、②データ共有・可視化のためのクラウドシステム、③市民・専門家・開発者が相互に議論するためのFacebookグループによって構成される。

システムモデルと繰り返し型モデル検査による次世代自動運転車を取り巻くSystem of Systemsのアーキテクチャ設計

慶應義塾大学
大学院 システムデザイン・マネジメント研究科 教授 西村 秀和

1 背景と目的

2020年ごろまでに自動車の自動運転化が進むことが社会的に期待されているものの、その実現に向けて検討するべき課題は山積している。アメリカのNHTSA（National Highway Traffic Safety Administration）およびSAE Internationalは、自動運転のレベル定義をしており、現在はLevel 3の自動運転を念頭に、自動車会社をはじめ関連企業が技術的な検討を進めている。

Level 3の自動運転では、加速・操舵・制動すべてを自動運転システムが行えるが、緊急時にはドライバが自動運転システムからの介入の要請に、適切に応答することを前提としている。このLevel 3の自動運転システムを検討する場合に、ドライバの介在に起因するシステム安全の問題がある。特に、自動運転システムにより自動車が制御されている状況から、すべての責任をドライバに移譲されたときの安全性の確保の問題がある。また、ドライバだけでなく、自動車を取り巻く環境も自動運転制御の安全性を脅かす要因になり得る。

こうした問題を取り扱うためには、自動運転車のみならず、ドライバ、情報システムや交通インフラなどの周辺環境を含めたSystem of Systems（以下、SoS）の問題として、検討する必要がある（図1）。また、次世代自動運転車は情報ネットワークと繋がることとなるため、セキュリティの脆弱性も問題視されており、周辺環境を踏まえた検討が求められている。

2 概要

本研究では、自動運転車が導入されたときの交通環境全体としての安全性を保障するため、自動運転車を取り巻くSoSに対して記述したシステムモデルに基づきモデル検査および安全分析を行う。そして実験データに基づきドライバモデルを明確にすることにより、SoSアーキテクチャを構築し、各構成システムに対する安全性要求を明確にする。また、SoSアーキテクチャを設計、更新するための方法を提案する。

(1) 自動運転車を取り巻くSystem of Systems（SoS）全体の基本アーキテクチャ記述

次世代自動運転車の実現に向けては、ドライバと自動運転システムのインタフェースを明確にした上で、双方の運転責任の移譲を円滑に行う必要がある。ここでは、HAVEit（Highly automated vehicles for intelligent transport）が提供している自動運転システム単体のアーキテクチャを解析した。自動運転車のコンテキスト分析をもとに、自動運転車を取り巻くSoSの構成要素としてドライバのみならず、自動運転車とその周辺の道路や信号などの交通インフラや情報システムを定義した上で、これらの構成システム間の関係性およびインタフェースを定義した。これらの過程で自動運転車の安全性を確保するSoSアーキテクチャについて、要求、構造、振る舞い、パラメトリック制約からなるシステムモデルをSysML（System Modeling Language）により記述した（研究目標1）。

この結果、自動運転車を取り巻くSoSでのドライバと自動運転システムの状態遷移を明らかにするとともに、自動運転車を取り巻くSoS構成システムの機能フローをアクティビティ図で、SoS構成システム間の相互接続を内部ブロック図（図1）でそれぞれ表すことにより、SoS全体の基本アーキテクチャを導いた。

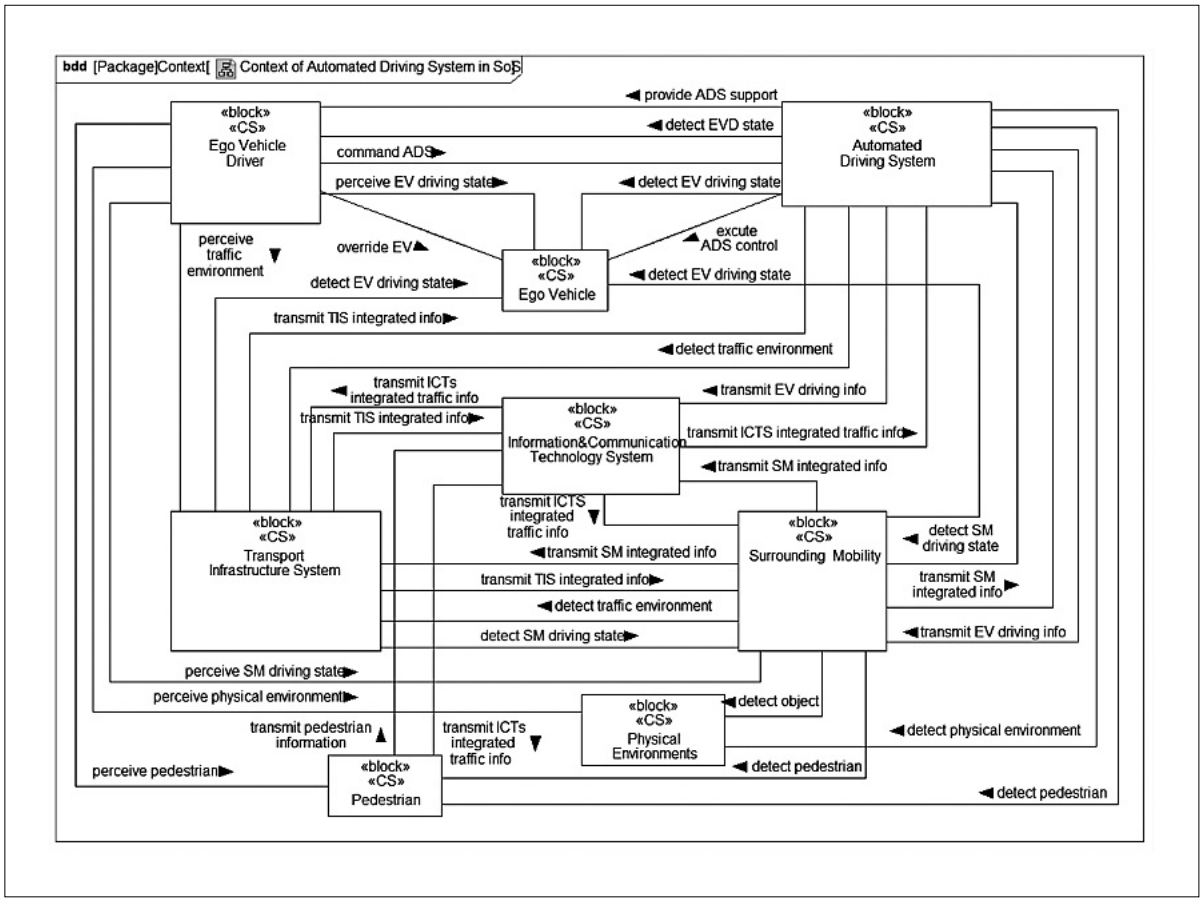


図１　自動運転車を取り巻くSystem of Systemsの構成システムの相互接続図

(2) 自動運転車を取り巻くSoSに対する安全性要求の明確化

FDIR（Fault Detection, Isolation and Recovery）の考えに基づき、自動運転車を取り巻くSoSの各構成システムに対して自動運転システムが満たすべき安全性要求を明らかにした（研究目標２）。自動運転車を取り巻くSoSの中で交通事故が発生することが、ある状況の下で最終的にドライバや歩行者などに危害を及ぼすと考え、SafeML（Safety Modeling Language）を用いた記述を行った。ここでは、自動運転車を取り巻くSoSの中で発生する交通事故を危害の源である危険とし、自動運転システムが搭載されている自動車のドライバや相手車のドライバ、あるいは歩行者などが危害を受けることになる危険状況を洗い出して、危害に至らないよう防御策、すなわち安全対策を検討した。

(3) ドライバモデルの構築

ドライバモデル構築のために、１年目には自動運転下でのドライバモデルを作成するためにプロトタイプ実験を行い、２年目にはドライバの通常運転時の安全確認行動を分析するために公道走行実験を行った（研究目標３）。

プロトタイプ実験では、規定コース上で自動運転車により被験者に走行してもらい、安全確認行動が手動運転時と自動運転時でどのように異なるかを検証した（図２）。この結果、手動運転時の安全確認パフォーマンスを、自動運転時にも引き継ぐ傾向がみられた。レベル３の自動運転では、状況により自動運転の状態からドライバへ運転権限を移譲するケースがあるため、安全確認パフォーマンスがこのような傾向にあることは問題となる可能性を示唆した。

プロトタイプ実験では、規定コース上での限られた環境下であったため、２年目はドライバの安全確認行動をより詳しく調べるための公道走行実験を行った。公道走行実験では、交通事故分析から事故の多い交差点環境を特定し、それらの交差点における安全運転度を検証した。その結果、ドライバは必ずしもリスクに応じた安全運転を行っていないことがわかった。また、同実験でリスクに応じた安全運転を行うドライバは、事前に行った運転適性検査の評価が高かった。

このことから、ドライバの運転挙動には、ドライバが本来持つ特性の一つである安全運転度が影響することが示唆され、ドライバモデルの認知、判断、操作に影響を与えるものとして、ドライバの安全運転度、あるいは交

通環境リスクの認識などの知識を追加する必要があることがわかった。

さらに、信号のない右折交差点の交通環境を構築して、自動運転システムとドライバの相互作用が生じるシナリオでのシミュレーションを実施した。自動運転システムによる運転に対してドライバがオーバーライドした結果として、道路を横断する歩行者との接触事故が生じてしまう可能性を示唆した。安全運転度が低いドライバによる自動運転システムへの安易な介入は、極めて危険なことであるため、こうした介入に対して、自動運転システムおよび他のSoS構成システムがどのように安全を確保するべきかを検討する必要がある。

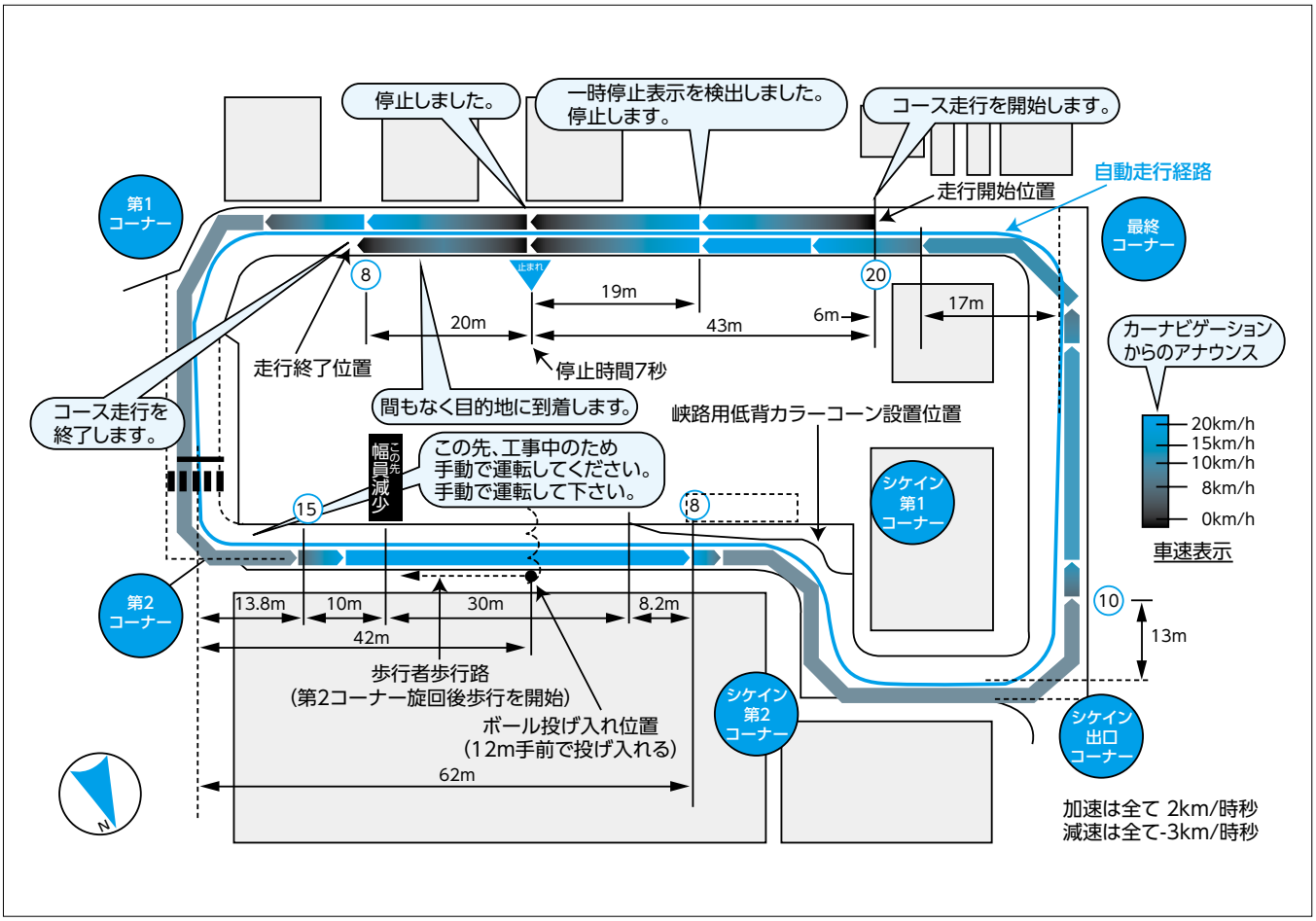


図２　プロトタイプ実験コース概要

(4) モデル検査による安全性の検証

研究目標１から３の検討により明らかにされる自動運転車を取り巻くSoSへの安全性要求に基づき、研究目標１で構築したSoSアーキテクチャのシステムモデルを検証するためにモデル検査を行った（研究目標４）。モデル検査の結果は各研究目標の成果にフィードバックした。

(5) SoSアーキテクチャ設計・更新方法の確立

本研究で実施したSoSアーキテクチャの検討過程に基づき、その設計に必要な検討項目とそれらの関係についてまとめた（研究目標５）。本研究では全体として安全性のビューを設定したが、SoSの構成システムの検討や構成システム間の関係の定義を行う際には、SoSに関係する利害関係者の様々なビューを検討する必要がある。そこで利害関係者とSoSアーキテクチャの関係性を明確にするため検討した。その結果、図３のように利害関係者に基づき４つのレイヤーを設け、それらの間の関係性を明示した。自動運転システムを取り巻くSoS全体の社会受容性について検討するレイヤーとして「社会レイヤー」を置き、次に、自動運転車が社会の中で利用されるシナリオを検討するレイヤーとして「利用レイヤー」を定義した。さらに、各構成システムの実現に向けた具体的な検討を行う「機能レイヤー」、実装のための方法や実体を検討する「実装レイヤー」を定義した。

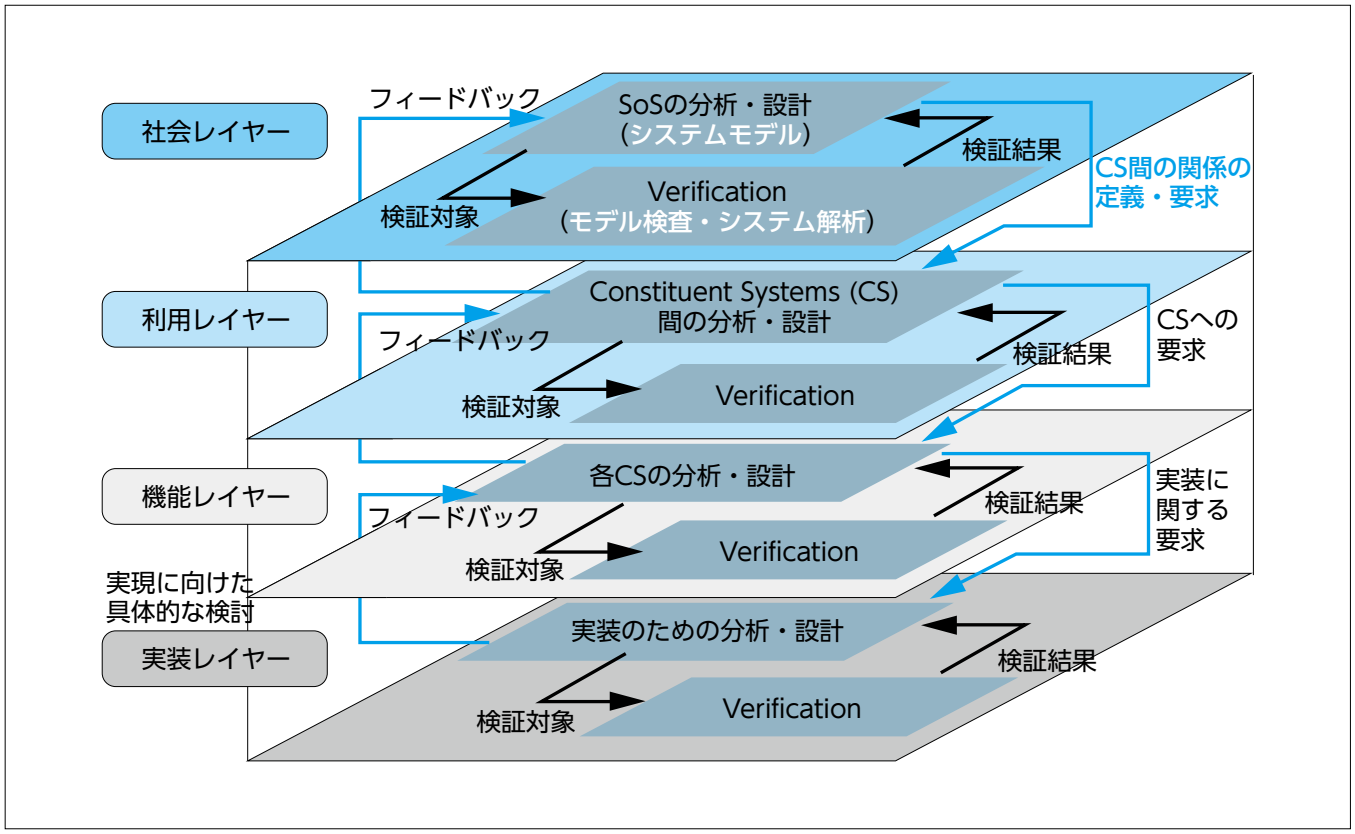


図3　SoSアーキテクチャのための参照モデル

3 産業界で研究成果が適用される場面と期待される効果

今後、自動運転車を社会に導入するにあたり、技術的な関連のみならず、損害保険、サービスなどに関連する企業が自動運転車を取り巻く構成システムに関わる。その場合、その機能や物理的な実装を行う際に、社会レイヤーから実装レイヤーまでの階層を考えておく必要がある。

例えば、本研究で定義したSoSアーキテクチャでは、自動運転システムを搭載する車両のドライバが、運転責任を負うという制約を仮定している。

これは法律の制約によるものである。完全マニュアルモードでドライバが運転しているとき、自動運転システムは何もアシストしないものとする。一方、自動運転モードで自動運転システム自体、または一部センサーの失陥が発生した状況では、ドライバにオーバーライドを求めるものとした。ドライバは、これに応えなければならず、もしドライバが自動運転システムの要求するオーバーライドに応答しない時は、自動運転システムは最小リスク状態へ移行させ、車を安全な場所に停止させる。今後、前提条件としている法律が変わった場合には、それを受けてSoSアーキテクチャを変更する必要がある。また、今後、自動運転システムに繋がろうとする構成システムが現れた場合には、当該SoSアーキテクチャへの接続に関する適合の可否を検討することができると期待される。

本研究の成果は、官公庁、保険会社、法律家、自動車の製造会社など様々な利害関係者を巻き込んだ、自動運転車を導入する社会全体を議論するための基盤となる。自動車―自動車間通信システムや自動車―道路間通信システムなど、自動運転車に必要なシステムを統合する際には、自動車メーカー、サプライヤーの他、ICTシステム、交通インフラシステムのプロダクトやサービスに関連する企業が関係する。こうした企業が、制度面や法律面で関連する利害関係者とともに、安全性やセキュリティなどの関心事を示すビューをもとに、記述されたSoSアーキテクチャに基づき、システム統合に向けた議論をすることができるようになると期待される。

例えば、交通インフラシステムから何らかのVICS（Vehicle Information and Communication System）情報を受け取った場合には、自動運転システムが自車の速度を減速して安全を確保する状況が考えられる。こうした利用レイヤーで検討したユースケースに関与する交通インフラシステムを提供する企業は、この安全性を確保するために必要な機能を検討する、そして、これを自社が開発・提供するシステムに実装するための検討を行うことができるものと考えている。

IPA EPM-Xの機能拡張によるプロアクティブ型プロジェクトモニタリング環境の構築
―次世代の定量的プロジェクト管理ツールとリポジトリマイニング研究基盤―

和歌山大学
システム工学部 准教授 大平 雅雄

1 背景と目的

ソフトウェア品質の確保や納期を遵守するためには、進行中のプロジェクトをリアルタイムにモニタリングし、プロジェクト内で発生する問題を早期に検出し、対処する必要がある。2012年にIPAはソフトウェア開発状況を定量的に把握しプロジェクト管理を支援するツールEPM-Xを公開し、現在は一般社団法人実践的プロジェクトマネジメント推進協会（PPMA）が普及展開活動を行っている。(※) EPM-Xは、基本的な定量データ（ソース規模、工数、進捗、品質など）の自動収集と、グラフ化によるプロジェクト管理支援機能を提供しているが、品質予測や工数予測の機能は提供されていない。そのため、プロジェクト管理において、プロジェクト管理者がグラフなどから問題の発生を目視で発見し、対策を講じるというリアクティブなプロジェクト管理にならざるを得ないという問題がある。

そこで、近年ソフトウェア工学分野で進展の目覚ましいリポジトリマイニング技術やリポジトリマイニング技術を発展させたプロアクティブマイニング技術を取り入れることで、定量的でかつプロアクティブなプロジェクト管理ができるようになることを目指した。

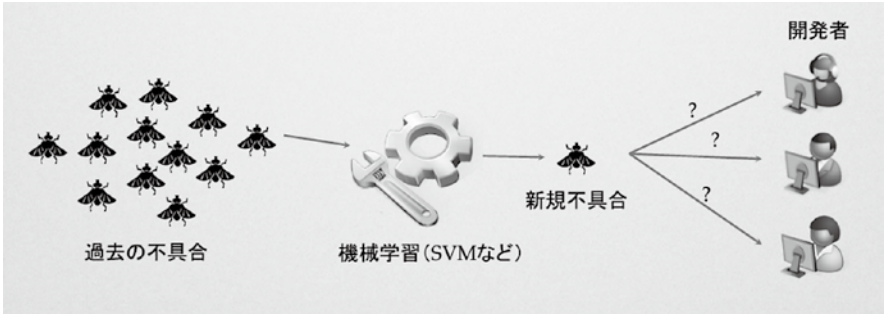
2 概要

リポジトリマイニング技術およびプロアクティブマイニング技術をEPM-Xのプラグインとして実装することで、定量的な品質・進捗予測機能とプロアクティブ型のプロジェクト管理機能を実現する８個のプラグインを開発した。

リポジトリマイニング・プラグイン

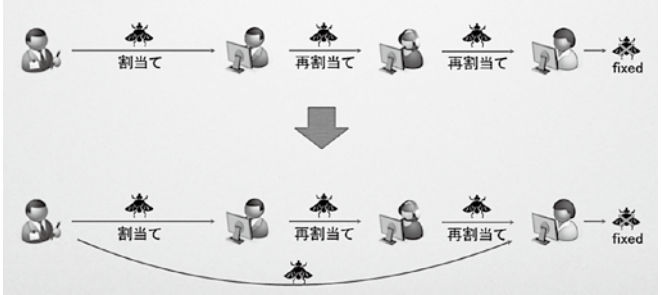
① タスク担当者推薦プラグイン

過去の完了したタスクの情報に基づいて、新規タスクを担当すべき開発者を推薦する。本プラグインを利用することによって、効率的なタスクの割当てを行い、タスク完了にかかる時間を短縮することが期待できる。



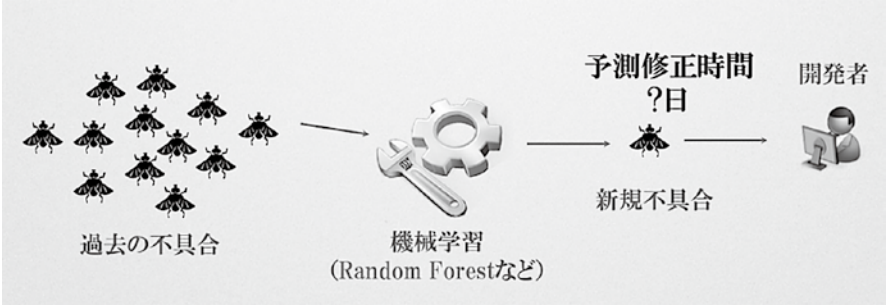
② タスク割当状況可視化プラグイン

過去のタスクの再割当情報を可視化する。本プラグインを利用することによって、効率的なタスクの割当てを行うことができ、タスク完了にかかる時間を短縮することが期待できる。



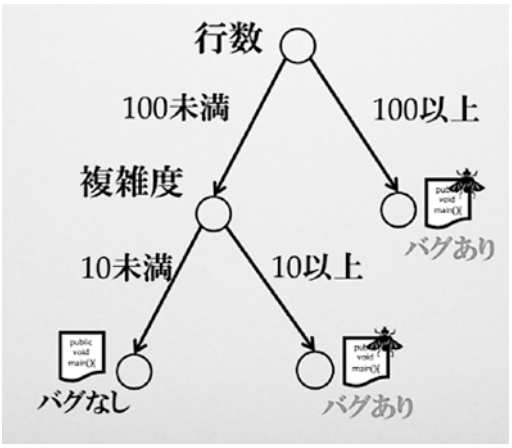
③ タスク完了時間予測プラグイン

過去の完了したタスクの情報に基づいて、新規タスクが完了すると考えられる時間を予測する。本プラグインを利用することによって、適切なタスクの見積を行うことができ、効率的なタスク管理が期待できる。



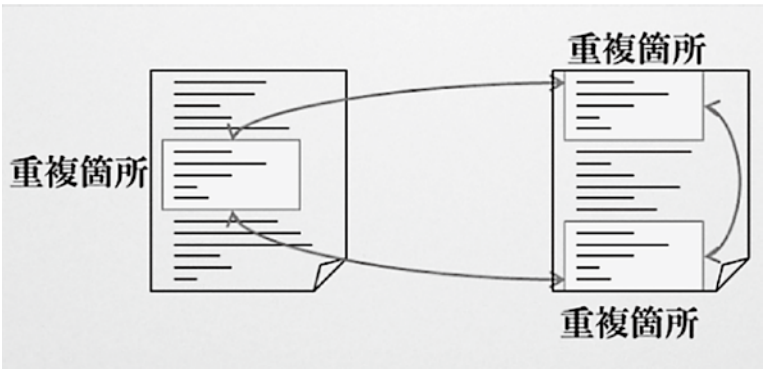
④ バグモジュール予測プラグイン

過去の不具合情報から、不具合が存在すると考えられるモジュールを予測する。本プラグインを利用することによって、どのモジュールに不具合が含まれやすいのかを確認することができ、不具合が含まれると考えられるモジュールに対して重点的にテストするなどによって、効率的なテストを行うことができる。



⑤ 重複コード検出プラグイン

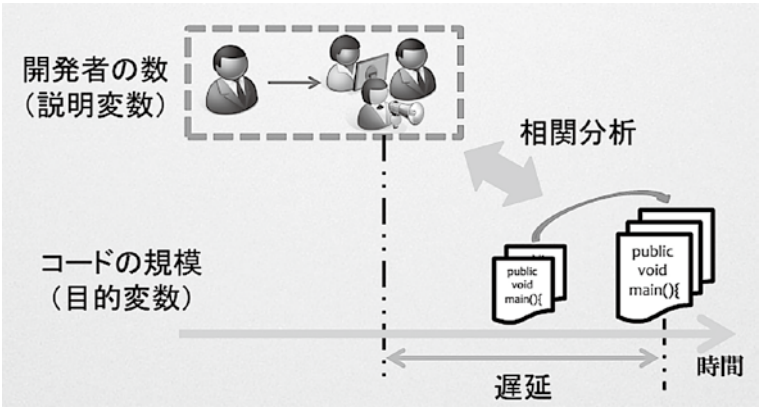
現在開発中のソフトウェアのソースコードに含まれる重複した箇所を抽出する。ソースコードに含まれる重複した箇所を知ることで、保守性の高いソフトウェアを目指したリファクタリングを行う際などに役立てることができる。



プロアクティブマイニング・プラグイン

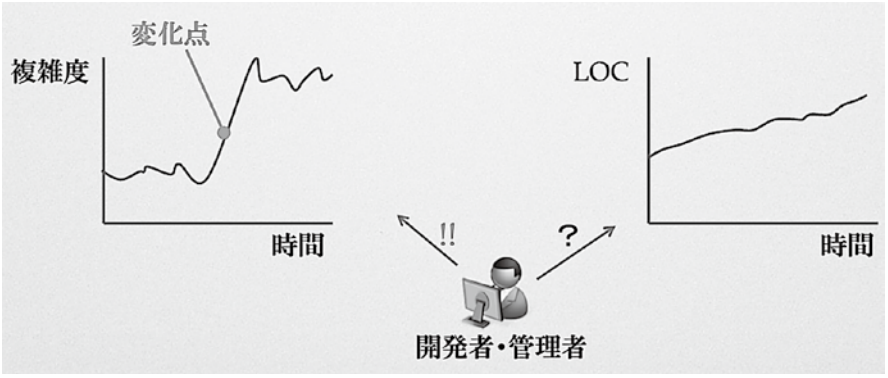
① 遅延相関検出プラグイン

リポジトリから計測されるメトリクス間の時間のずれる関係を明らかにする。メトリクス間の時間のずれる関係とは、事象Aが発生すると一定期間後に事象Bが発生するといった関係を指し、メトリクス間の時間のずれる関係を把握することによって、今後プロジェクトに起こると考えられる事象を知ることができ、ソフトウェア開発・保守の品質および生産性の向上が見込まれる。



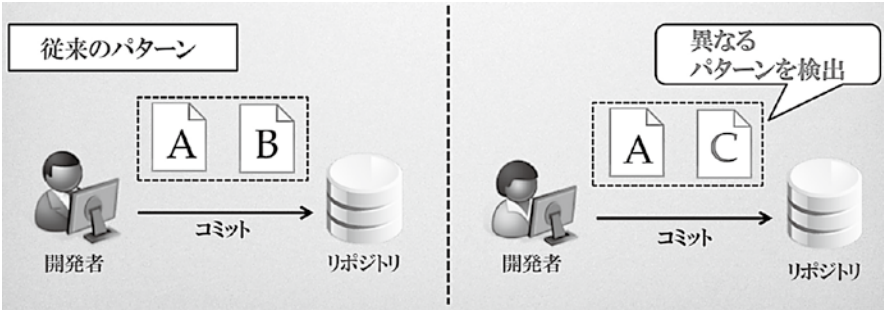
② 変化点検出プラグイン

ソフトウェア開発における開発状況の変化を早期に発見するために、変化点検出アルゴリズムに基づいてリポジトリから計測されるメトリクスの値の解析を行う。ソフトウェア開発における開発状況の変化を早期に発見することによって、潜在的に重大な問題に素早く対応できることが期待できる。



③ 影響波及分析プラグイン

対象リポジトリに含まれるファイルを選択すると、同時に変更されたことのあるファイルおよび、その確率、およびファイルを修正するのにこれまでかかった時間が出力される。あるファイルを修正しようと考えている開発者は、修正にかかるコストを容易に見積もることができる。



3 産業界で研究成果が適用される場面と期待される効果

本研究の研究成果は、これまでプロジェクトマネージャーの勘や経験に頼りがちであったソフトウェア開発をより高度化するものである。経験の浅いマネージャーの意思決定を強力に支援するだけではなく、経験豊富なマネージャーが自らの判断の正しさを客観的・定量的な形でチームに伝えるためのツールとしても利用可能である。

定量的プロジェクト管理における高度データマイニング技術の利用

IPA EPM-Xのプラグインとしてリポジトリマイニング技術を提供することにより、高度な予測機能を「手軽に」利用できるようになる。特に、テスト・保守工程においてのリソース割当の最適化に役立つことが見込まれる。

プロアクティブなプロジェクト管理の実現

異常／予兆検知技術をベースとしたプロアクティブマイニング技術によって、問題の発生を未然に防止するプロアクティブなプロジェクト管理が実施できるようになる。特に、プロジェクト内に発生する品質管理上の異常を早期に検出することが可能となる。また、プロジェクトが危険な状態に入りかけていることをデータとして示すことで、プロジェクトや管理の見直しのための定量的な材料にすることができる。

（※）EPM-X にクロスサイト・スクリプティングおよび任意 DLL 読込みの脆弱性が存在することが判明したため、現在、EPM-X の提供およびサポートは終了しています。
URL http://www.ipa.go.jp/sec/info/20170519.html

2014年度 その他

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

同志社大学
政策学部 教授 中田 喜文

1 背景と目的

本研究では、国内外の多様なソフトウェア技術者の生産性を、技術者が働く組織と社会における評価を重視した指標によって、また、ソフトウェア技術者の労働条件を投入労働時間数や年収あるいは、時間給等の要素投入量の多様性を重視した指標まで、幅広い視点から評価することで、日本のソフトウェア技術者の生産性と労働条件を、多面的に比較可能な形で示す。

この結果、日本のソフトウェア技術者の多様な側面を比較可能なデータによって示すことができる。このことにより、現状のソフトウェア産業政策、ソフトウェア人材政策を評価し、より政策効果の高い政策構築に向けての重要な情報を提供することが可能となる。

本研究は、日本におけるソフトウェア技術者の生産性、処遇、労働条件と環境の現状について、世界の同じ職務に従事するソフトウェア技術者と包括的に比較する最初の研究であり、日本のみならず、国際比較調査の対象国にとっても、自国のソフトウェア技術者の処遇の実態を国際的に比較評価することができる。

2 概要

(1) ソフトウェア技術者の生産性と処遇等労働条件決定に関する仮説と理論モデルの設定

先行研究文献の精査と国内外ヒアリングで得られた情報に基づき、ソフトウェア技術者の能力等の個人要因、職場環境要因、人的および生産管理の在り方等のマネジメント要因、さらには産業構造や労働市場の特性等の外部環境要因の４要因に分類整理し、それらの各分類要因間の関係について、相互の関係の有無と相互作用の向き（プラス、あるいはマイナス効果）に関する仮説を包含する「ソフトウェア技術者生産性・労働条件モデル」（図１）を構築した。

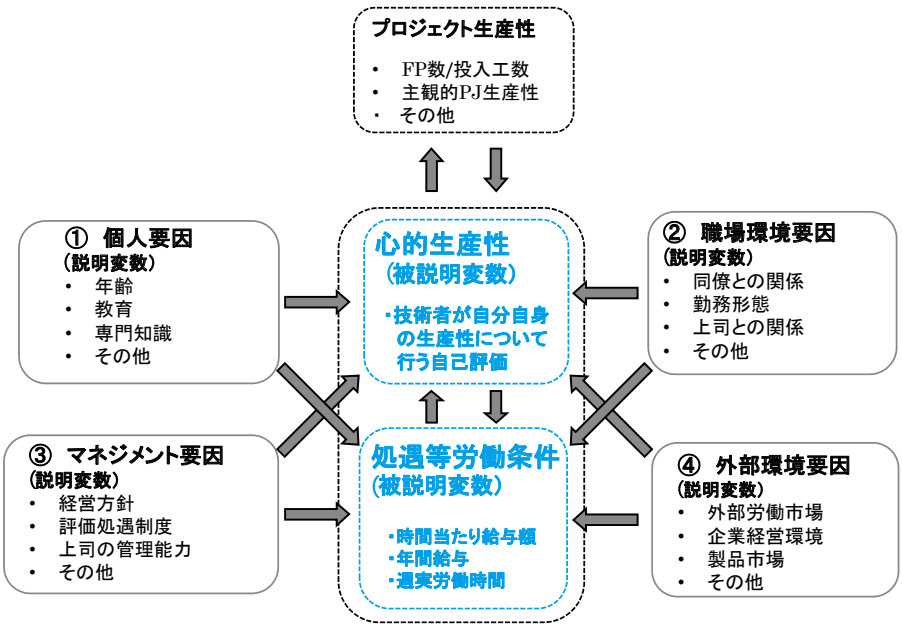


図１　４要因ソフトウェア技術者生産性・労働条件モデル

(2) アンケート調査の実施

上記４要因モデルを構成する①個人要因、②職場環境要因、③マネジメント要因、④外部環境要因に加え、モデルの被説明変数である処遇等労働条件および技術者生産性（心的生産性）に関する質問票を作成し、表１に示すとおり、日本、アメリカ、ドイツ、フランス、中国のソフトウェア技術者にアンケート調査を実施した。なお、ソフトウェア技術者は ERP ソフトウェア技術者、組込みソフトウェア技術者、その他のソフトウェア技術者の３タイプに区分して実施している。

(3) 国内および比較国個票データの統計分析

アンケート調査結果から労働条件および心的生産性に係る代表的なデータの各国比較結果を示す。

①労働条件の比較：月間労働時間の分布比較

各国のソフトウェア技術者（全体）の週当たりの実労働時間分布を図２に現す。左から右へ労働時間の短い技術者からより長い技術者の順で、割合を示している。日本の場合、法定労働時間である週 40 時間を超えない技術者の割合は 4.3％であるのに対し、フランスでは、76％、ドイツでは 92％である。週当たり 10 時間以上の

表１　アンケート実施概要

対象国	実施時期と方法	回収サンプル				協力／委託機関
		回収数	ERP ソフト技術者	組込みソフト技術者	その他のソフト技術者	
日	2015 年 12 月～ 2016 年 1 月 日本国内の電機連合組合員および管理者に対し調査票（紙）で実施	3,115	364	493	164	電機連合（全日本電機・電子・情報関連産業労働組合連合会）
米	2016 年 3 月～ 4 月 アメリカ国内企業の技術者および管理者に対し Web で実施	555	337	115	82	VizQuest Ventures（アメリカマサチューセッツ州）
独	2016 年 4 月～ 5 月 ドイツ国内企業の技術者および管理者に対し Web で実施	504	85	336	13	VDE（ドイツ電気･電子･情報技術協会） Elektronik Praxis BITKOM
仏	2016 年 3 月～ 4 月 フランス国内企業の技術者および管理者に対し Web で実施	344	168	102	21	Enquete& Opinion
中	2016 年 1 月 中国内企業の技術者および管理者に対し調査票（紙）で実施	300	144	111	14	上海竪竪情報技術有限会社

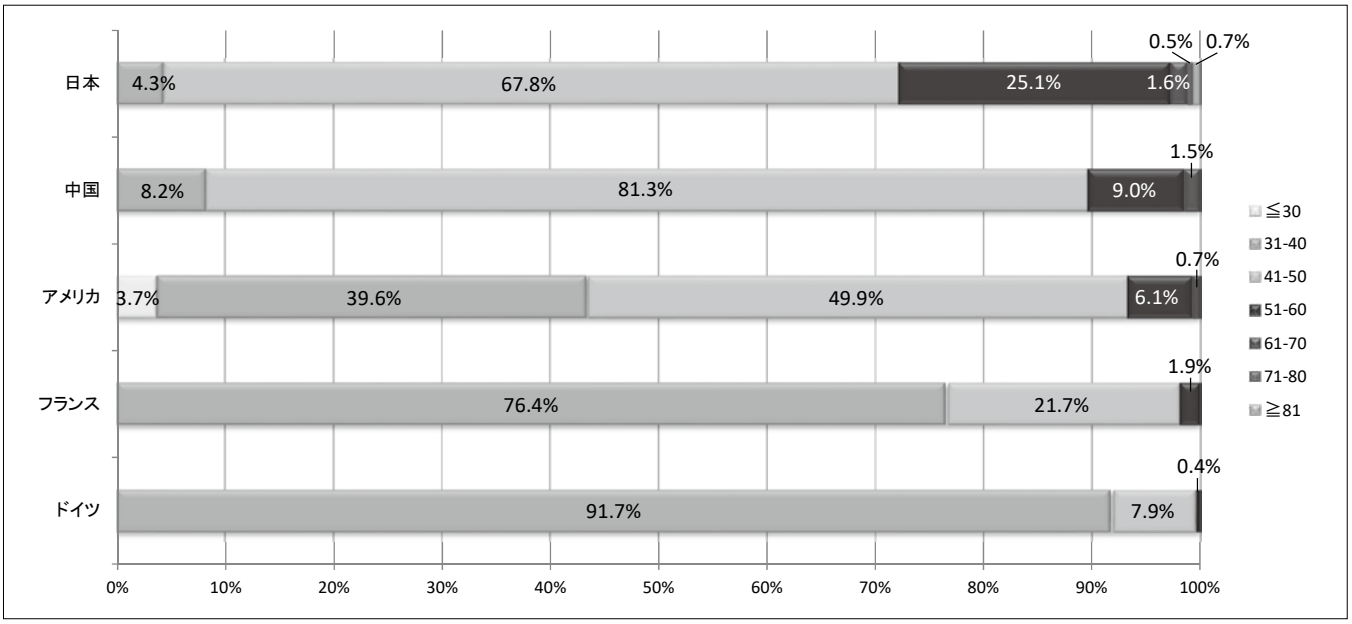


図２　週実労働時間別分布

残業をするソフトウェア技術者の割合は、日本では 27% を超えるのに対し、中国 10.5%、アメリカ 7%、フランス 1.9%、そしてドイツで 0.4% である。このように、日本のソフトウェア技術者の長時間労働が突出していることが明確となった。

②心的生産性指標の比較：主観的生産性指標による比較

主観的生産性指標は、ソフトウェア技術者本人が行った、自己の業務における生産性に関する主観的評価の指標である。具体的には、質問票で尋ねる３つの設問、a. 自分の能力を発揮できているか、b. 自分の担当職務に期待される成果を出せているか、c. 自分の仕事は社会に貢献しているか、に対して、４点評価で回答した５ヶ国の回答者のデータを主成分分析し、得られた個々人の主成分得点を、国ごと、３タイプのソフトウェア技術者ごとに、その平均値を計算したものである。（図３）ただし、５ヶ国３タイプのソフトウェア技術者全サンプルをプールした全体について、その平均は０、標準偏差１となるように標準化されている。それゆえ、この指標のプラス値は、全体平均より高く、マイナス値は、全体平均より低いことを示している。

日本のソフトウェア技術者は、ERP ソフトウェア技術者、組込みソフトウェア技術者、あるいはその他ソフトウェア技術者の如何に関わらず、自己の生産性に関する評価は、すべて大きく５ヶ国平均を下回り、比較５ヶ国中、最低水準である。逆に、アメリカのソフトウェア技術者の主観的生産性の高さは、どのタイプのソフトウェア技術者であっても、他国を大きく上回っている。残りの３ヶ国については、ドイツ、フランスの生産性指標は平均より高くプラスとなっているが、中国はその他ソフトウェア技術者のみがプラスで、ERP および組込みソフトウェア技術者ではマイナスの値となっている。

③心的生産性指標の比較：職務満足度指標による比較

職務満足度指標は、ソフトウェア技術者の職務に関する総合的な満足感の指標である。具体的には、質問票で尋ねる６つの設問、a. 仕事を一緒にする仲間に恵まれている、b. 今の仕事は面白い、c. 自分のペースで働くことができる、d. 自分の納得できる報酬や地位を得ている、e. 今の仕事は自分に合っている、f. 重要な仕事を任されている、の６設問に対する回答を主成分分析して得た各人の主成分得点を、主観的生産性指標の場合と同様、全体平均を０、標準偏差を１となるように標準化したものである。（図４）

アメリカがどのタイプのソフトウェア技術者であっても、最も職務満足の水準が高い。他方、日本のソフトウェア技術者が、３つのタイプの如何に関わらず、すべてのタイプにおいて、その職務満足度指標は大きなマイナス値となっており、職務満足も日本の水準が最も低い。し

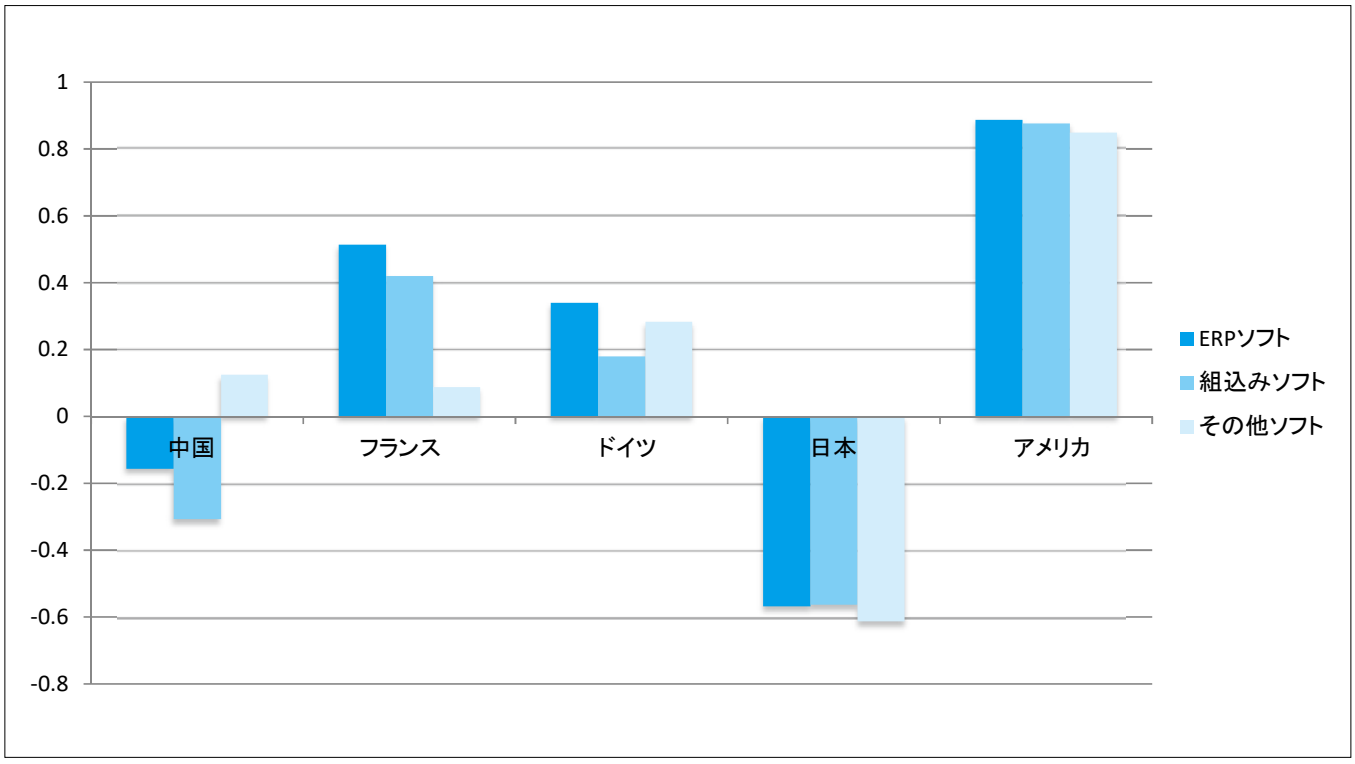


図３　主観的生産性指標による比較：３つの仕事の達成度変数１の第一主成分

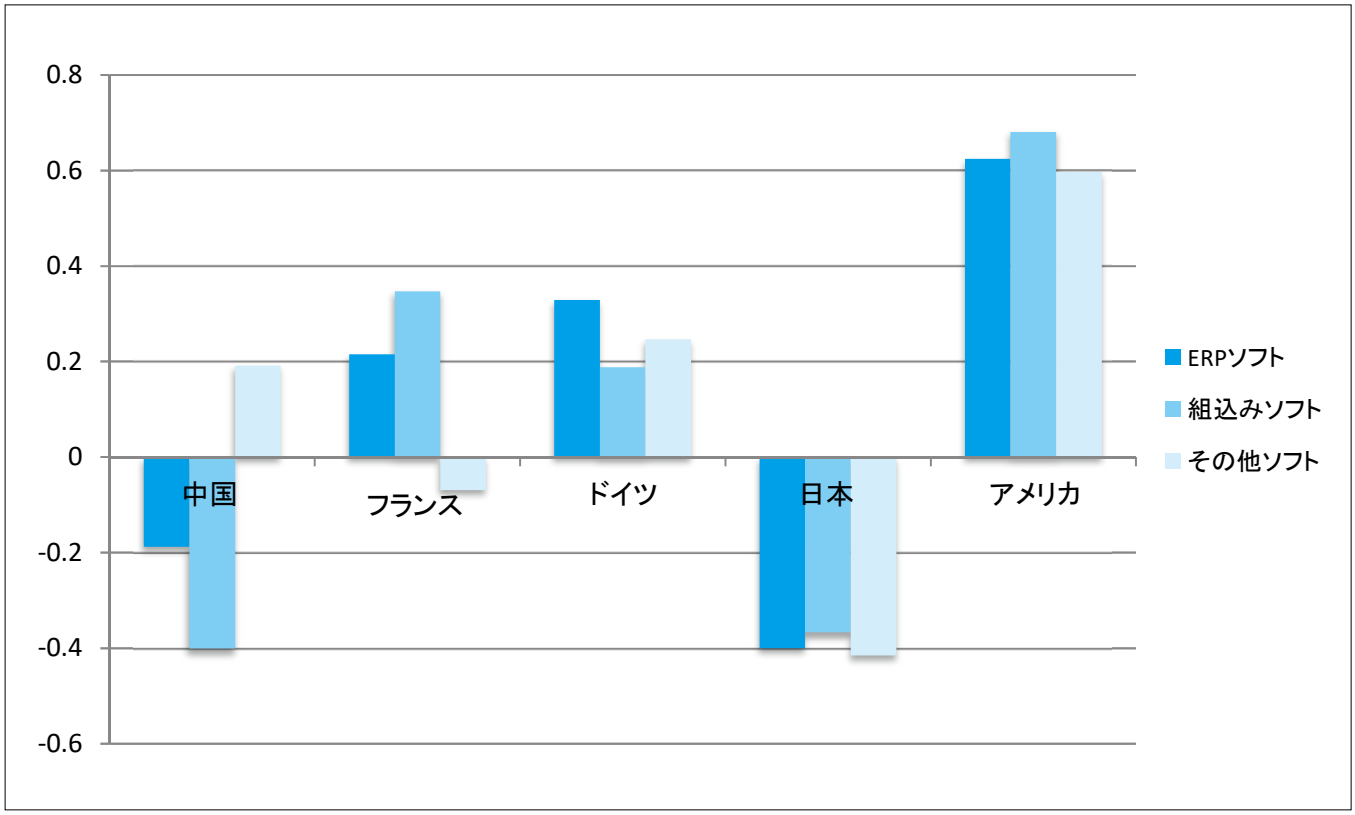


図４　主観的生産性指標による比較：５つの仕事満足度変数の第一主成分

かし、個別のタイプに着目してみると、組込みソフトウェア技術者については、若干ではあるが日本以上に中国の職務満足度が低いこともわかる。また ERP ソフトウェア技術者についても、日本ほどは低くはないが、中国の職務満足の水準も世界的に見れば低いことが確認できる。

3　所感

主要国と比べ極めて質の低い労働環境の中で日本のソフトウェア技術者が働いており、自己の仕事評価も相対的には最も低く、かつ仕事やりがい感も最下位であったことは、我々研究を実施したメンバーにとっても大きな驚きであった。同時に日本の社会とそこに集う国民の現在と将来の生きる可能性にとっても、大きな足かせとなるのではと憂う。それゆえに、個別業界の問題としてとらえるのではなく、国民の、日本社会の大きな課題ととらえ、政官民の一致団結した取り組みが強く求められる。

2015年度 その他

携帯端末用アプリケーションソフトウェアが地方経済に与える効果の実証実験評価に関する研究

福井大学
大学院 工学研究科 准教授 橘 拓至

1 背景と目的

現在、ソフトウェア事業者の大部分が東京を中心とする大都市圏に集中しており、地方では少数の小規模なソフトウェア事業者が存在している状況がある。また、ソフトウェア事業の売上額は、大都市圏への集中が著しい。地方のソフトウェア事業者が請け負う業務も、大半が首都圏や自治体が発注する仕事であり、地域内の企業は大都市圏のソフトウェア事業者に仕事を発注する傾向にある。

地方のソフトウェア産業を発展させるには、以下のような観点が必要になる。

- 地域の企業が、地域内のソフトウェア事業者に発注する。
- 発注ソフトウェアは、同地域にとって利用価値が高いものである。
- そのソフトウェアがもたらす効果を明示する。

しかしながら、このような地方のソフトウェア産業を発展させるようなソフトウェアは、これまで明確には存在しておらず、ソフトウェアがもたらす効果も明らかになっていない。それゆえ、地方のソフトウェア産業を活性できるようなソフトウェアを確立し、その効果を明らかにすることが期待される。

本研究では、携帯端末用アプリケーションソフトウェアが地域経済にもたらす効果を、実証実験によって評価する。この実証実験によって、地方での利用に特化したソフトウェアが地域経済に与える効果・価値を明らかにする。

本実証実験で使用するアプリケーションソフトウェアは、商店ごとに設定された「ゆるキャラ」を育成するゲームであり、キャラクターの育成は、Bluetoothによるすれ違い通信を利用する。キャラクターの育成には商店内に入る必要があるため、来客数の増加が期待できる。また、すれ違い通信によってキャラクター同士のバトルを実行するため、ユーザの街歩きが期待できる（図1）。

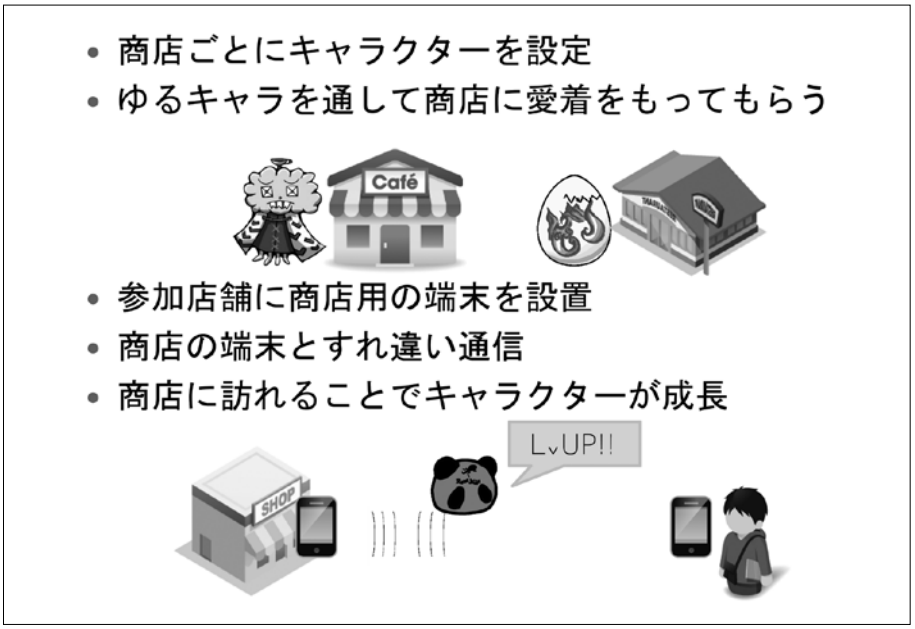


図1　アプリケーションソフトウェア概要

2 概要

アプリケーションソフトウェアおよび環境の開発

① 地域経済活性化に向けたキャラクター育成アプリの開発

研究責任者がこれまでに開発してきたAppleのiOS7対応アプリケーションソフトウェアを基に、同OSの開発環境であるXcodeを用いて、実験用アプリを開発･改良した。開発・改良にあたっては、商店街に活力を与えることを実現するための機能を重視し、新たなゲーム要素の追加として、キャラクターの進化、ボスキャラ・隠れキャラの導入、図鑑ページを用いたコレクション機能の追加などを行った。キャラクターの進化は､レベルが上がるたびに、お店のキャラクターが進化していくという機能になっている。これらの改良により、ユーザが興味をもって参加してくれるように特にゲーム面での工夫を行った。

さらに、ユーザ数の増加を期待して、Android端末用のアプリをスクラッチから開発した。Android OSはBluetoothによるすれ違い通信の実現が困難であったため、新たにBluetooth Low Energyを利用した。また、Bluetooth Low EnergyはBluetoothとの通信が困難なため、開発済のiOSアプリに対しても、Bluetooth Low Energyを使用する改良を行った。

② 実験環境の構築

実証実験を実施するための環境を構築した。開発するアプリには、商店ごとにゆるキャラを８体作成して画像とした。さらに、各キャラクターにはステータスと３つのタイプを設定した。ステータスはレベルとともに変化し、レベルが上がるとキャクラターが進化する（図２）。さらに、ランキング機能を追加するために、アプリに獲得したポイントをサーバに送信する機能を追加した。この送信は、Bluetoothによるすれ違い通信ではなく、インターネット経由でサーバに送信される。また、ランキング結果を表示する専用ホームページを開発した（図３）。

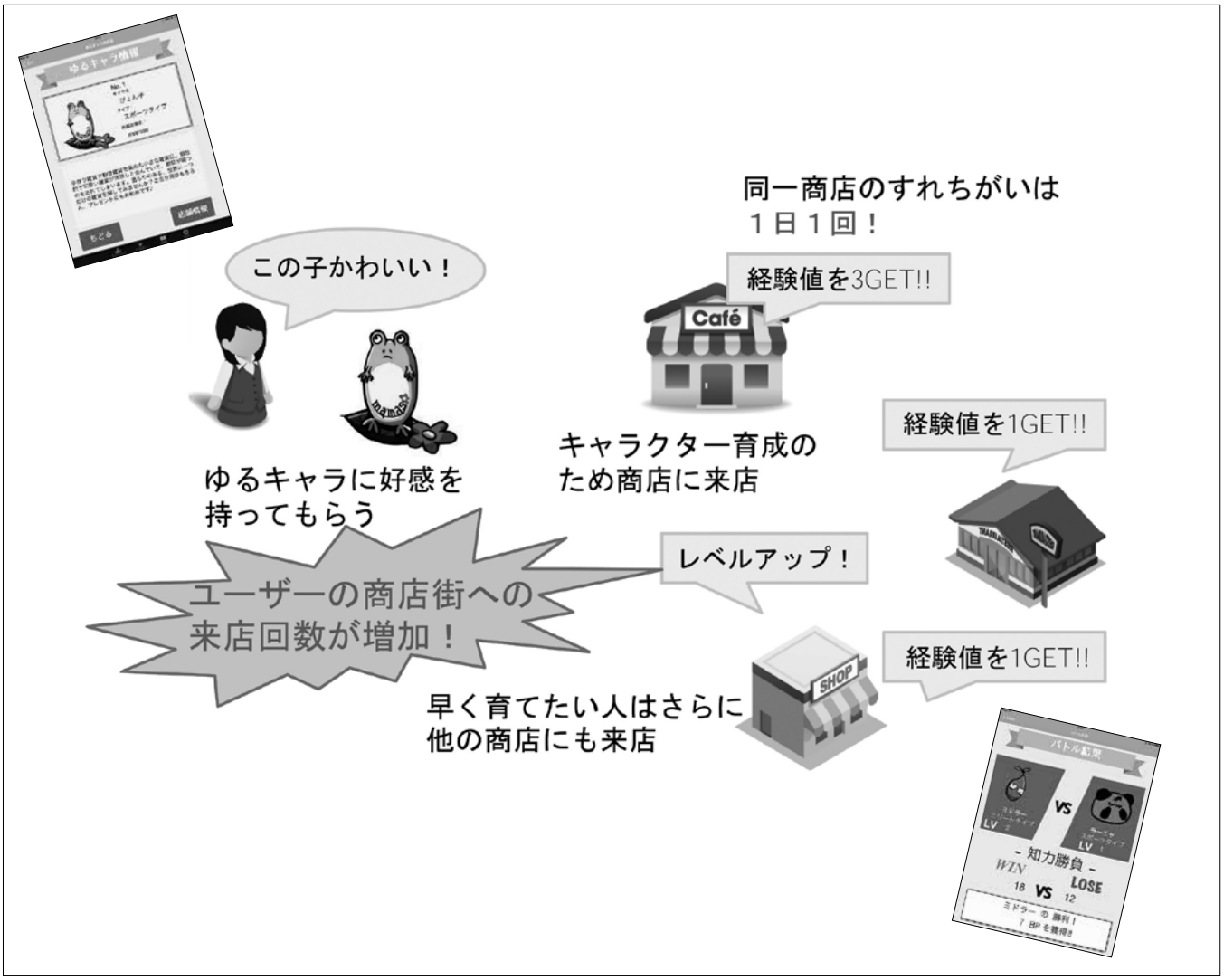


図２　キャラクター育成イメージ

ユーザランキング　店舗対抗ランキング　店舗別ランキング

図３　ランキングページ

実証実験の実施と評価

商店街の８店舗および180名程度のユーザの参加により実証実験を実施した。また、アプリケーションソフトウェアが地域経済に与える効果・価値を明らかにするために、参加店舗、参加ユーザおよび実証実験に参加していない一般ユーザに対してアンケート調査を実施し、その結果を分析・評価した。

今回の実証実験では、アプリケーションソフトウェアの参加店舗数が８店舗という少ない場合には、商店の売り上げに対する経済効果が期待できないことが分かった。一方で、参加ユーザおよび参加商店からは、参加店舗数が増加することで、経済効果の増加が期待できるのではないかという調査結果が得られている。それゆえ、経済効果を生み出すには、多くの商店に参加してもらうことが重要であるといえる。

また、参加店舗数が８店舗の場合では、賑わいを創出することはできないという結果となった。しかし、賑わい創出の観点についても、参加店舗数を増加させることで、本アプリの効果が期待できることが分かった。

3　効果

本研究の成果であるアプリは引き続きユーザに使用してもらい、定期的にバージョンアップや規模拡大を進めていく。iOS対応アプリに関しては、Bluetooth Low Energyを使用する新バージョンへのアップデートを実施する。またAndroid OS対応アプリは、iOS対応アプリとの接続性を確認し、さらに複数端末との通信についても調査し、運用できることが判明した後にアプリの公開を行う予定である。今後も、他の商店街でも利用してもらうために、宣伝活動を含めた普及活動を積極的に進めていく。

アプリを活用した、このような取り組みへの参加は容易ではないことがわかった。一方で、商店側にもこのようなアプリによる地域活性化に期待している面もあることもわかった。そのため、このようなアプリの開発を行っていくことで、地域経済の活性化はもちろんのこと、地方の産業界にもプラスになるのではないかと考えられる。

■ゆるキャラ商店街

URL https://itunes.apple.com/us/app/yurukyara-shang-dian-jie/id983148529?mt=8

ソフトウェア高信頼化センター（SEC）の事業概要

事業の目的

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（IPA/SEC）は、産学連携によるソフトウェア・エンジニアリングの拠点として、2004年に設置され、高度情報化社会を支える情報処理システム及びソフトウェアの高信頼化をミッションとしています。現在は、IoT時代に向けて、組込みソフトウェア産業の競争力強化や、製品・サービスの安全・安心を確保するために必要な要件を定めたガイドライン策定や、重要インフラなどの公共性の高い情報処理システムにおける障害の低減等、以下の事業に取り組んでいます。
http://www.ipa.go.jp/sec/our_activities/index.html

IoTの安全・安心の確保に向けた仕組みの構築

IoT時代の到来を迎え、製品・システムがつながって、新しいサービスを創出したり、様々なデータを用いてシステムを制御する世の中へと変遷しつつあります。IPA/SECでは、このIoT時代のことを別名で「つながる世界」と呼んでいます。つながる世界では、様々な品質の製品が存在しており、メーカーが想定していないつなぎ方等により、安全上あるいはセキュリティ上の問題を引き起こす危険があります。IPA/SECでは、安全・安心なIoT機器やシステムの開発に向けた取り組みを行っています。

システムズエンジニアリングの推進

近年、第４次産業革命をもたらすと言われているIoTの進展や、独立したシステムが互いに関係し合って価値を提供するSoS（System of Systems）のような複雑なシステムの増加、また、システムに対する要求の多様化、高度化、複雑化など、システム開発を取り巻く課題も難しくなってきており、解決が困難になってきています。IPA/SECでは、国内外の事例を参考に、システムズエンジニアリングの効果や有効領域について検討を行っています。

複雑化したシステムの安全性確保

近年の組込みシステムは、個々の構成要素自体の高機能化に加え、各構成要素が接続されて連動動作することにより、益々、大規模・複雑化が進んでいます。IPA/SECでは、そのようなシステムにおけるシステムライフサイクルの全般をカバーした安全性・信頼性・セキュリティ向上手法の調査・研究、およびその普及を目的とした活動を行っています。

システム構築の上流工程強化

様々なシステムが複雑に接続されることによって、システムの開発要件の不確実性が急激に拡大する開発現場の課題にサービス事業者からの要求の定義、設計といった上流工程から対応するため、上流工程に必要な施策を検討する活動を行っています。

重要インフラ分野のシステム障害への対策

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等における情報処理システムの信頼性向上のため、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・分野を越えて幅広く共有し、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を目指しています。

定量的プロジェクト管理の推進

高品質のソフトウェアを効率的に開発するためには、開発に関わる様々な数値データを用いたプロジェクトの進捗管理等が重要であり、その普及活動を推進しています。

ソフトウェア信頼性の見える化

社会全体を支えるITを利用者が安全・安心に使えるようにする為、利用者視点に立ったソフトウェア信頼性の見える化を促進しています。

ディペンダビリティの確保

利用者が製品やシステムをいつでも安心して利用できるようにする為、関連団体と連携して、安全性やセキュリティなどの確保に向けた取組みを行っています。

SEC journalのご案内

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（IPA/SEC）は、ソフトウェア・エンジニアリングの啓発とともに、産業界での国際競争力の強化や技術力の向上を目指して、ガイドラインや入門書、ソフトウェア開発データ白書など、様々な成果物を発行しています。SEC journalは、IPA/SECの活動成果の他、主にソフトウェアの現場に従事する技術者へ向けて、ソフトウェアおよびシステムの有効性、実証的な論文や事例などを掲載しています。（年４回発行）

＜ご購読申込み＞

SEC journal 本誌は、IPA/SECの関連するイベントで配布しているほか、ご購読希望の方には無償でお送りしています。お申込みは、以下のURLをご覧ください。
http://www.ipa.go.jp/sec/secjournal/info.html

＜最新号とバックナンバー＞

SEC journal の最新号およびバックナンバーを以下のURLからダウンロードできます。
http://www.ipa.go.jp/sec/secjournal/index.html

＜既刊（直近３号分）のご紹介＞

特集
システムズエンジニアリング

特集
SEC 2016年度活動概要

特集
創刊50号記念

SEC Journal　論文募集

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では、SEC journalに掲載する論文を募集しています。

＜募集テーマ＞

◎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

- 開発現場への適用を目的とした手法・技法の詳細化・具体化などの実用化研究の成果に関する論文
- 開発現場での手法・技法・ツールなどの様々な実践経験とそれに基づく分析・考察、それから得られた知見に関する論文
- 開発経験とそれに基づく現場実態の調査・分析に基づく解決すべき課題の整理と解決に向けたアプローチの提案に関する論文

◎ソフトウェアが社会経済にもたらす革新的効果に関する実証論文

＜対象分野＞

「キャリア開発」「技術者スキル標準」「開発プロセス技術」「設計手法、設計言語」「支援ツール、開発環境」「定量化エンピリカル手法」「技術者教育、人材育成」「組織経営、イノベーション」「品質向上･高品質化技術」「見積り手法、モデリング手法」「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」「プロジェクト･マネジメント技術」「要求獲得･分析技術、ユーザビリティ技術」

＜評価基準＞

ａ．実用性（実フィールドでの実用性）
ｂ．可読性（記述の読みやすさ）
ｃ．有効性（適用した際の効果）
ｄ．信頼性（実データに基づく評価・考察の適切さ）
ｅ．利用性（適用技術が一般化されており参考になるか）
ｆ．募集テーマとの関係

＜募集要項＞

締 切 り：投稿は随時受付けておりますが、１月・４月・７月・11月の各月末に締切り、２名以上の査読者により審査を行います。
査読結果：約１ヶ月で査読結果を通知します。採録となった論文は年４回発行のSEC journalに掲載されます。
応募方法：応募様式など詳しくはHPをご覧下さい。
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SEC journal 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。

ソフトウェア工学分野における産学連携事業成果の紹介　2017年９月25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：03-5978-7417
URL：http://www.ipa.go.jp/sec/index.html

IPA Better Life with IT