ソフトウェア・シンポジウム 2020 (オンライン開催)

論文集

ソフトウェア技術者協会

募集案内

ソフトウェア・シンポジウムは，ソフトウェア技術に関わるさまざまな人びと，技術者，研究者，教育者，学生などが一堂に集い，発表や議論を通じて互いの経験や成果を共有することを目的に，毎年全国各地で開催しています．

第 40 回目を迎える 2020 年のソフトウェア・シンポジウムでは，この数年間で試みてきた新しい取り組み（チュートリアルや Future Presentation など）をさらに発展させたものにしたいと考えています．このほか，SS2019 に引き続き，論文発表や事例報告と，ワーキンググループで議論を行います．

開催概要

■ 日程 ： 2020 年 6 月 17 日（水曜日）～ 19 日（金曜日）

■ 場所 ： いわて県民情報交流センター（アイーナ） を予定しておりましたが，
COVID-19（coronavirus disease 2019） 感染症対策として，オンライン開催に変更しました

■ 主催 ： ソフトウェア技術者協会

■ 後援 ： 情報処理推進機構

■ 協賛 ： 岩手県情報サービス産業協会，ソフトウェアテスト技術振興協会，アジャイルプロセス協議会、オープンソースソフトウェア協会，情報サービス産業協会，情報処理学会，ソフトウェア・メインテナンス研究会，電子情報通信学会，日本ソフトウェア科学会，組込みシステム技術協会，日本 SPI コンソーシアム，日本ファンクションポイントユーザ会，派生開発推進協議会，日本科学技術連盟，組込みソフトウェア管理者・技術者育成研究会，TOPPERS プロジェクト，PMI 日本支部

スタッフ一覧

実行委員会

実行委員長

野村 行憲（岩手県立産業技術短期大学校）
漆原 憲博（ジェーエフピー）

実行委員

伊藤 昌夫（ニルソフトウェア）
市川 尚（岩手県立大学）
小笠原 秀人（千葉工業大学）
小楠 聡美（HBA）
岸田 孝一（SRA）
栗田 太郎（ソニー）
小松 久美子（帝塚山学院大学）
杉田 義明（福善上海）
鈴木 裕信（usp lab.）
中野 秀男（中野秀男研究所）
奈良 隆正（NARA コンサルティング）
本多 慶匡（東京エレクトロン）
宮田 一平（SHIFT）
三輪 東（SCSK）

プログラム委員会

プログラム委員長

日下部 茂（長崎県立大学）
河野 哲也（ディー・エヌ・エー）

プログラム委員

秋山 浩一（富士ゼロックス）
安達 賢二（HBA）
鯵坂 恒夫（和歌山大学）
天嵜 聡介（岡山県立大学）
荒木 啓二郎（熊本高等専門学校）
伊藤 昌夫（ニルソフトウェア）
臼杵 誠（富士通）
梅田 政信（九州工業大学）
大平 雅雄（和歌山大学）
小笠原 秀人(千葉工業大学)
小楠 聡美（HBA）
小田 朋宏（SRA）
落水 浩一郎（University of Information Technology, Myanmar）
片山 徹郎（宮崎大学）
北須賀 輝明（広島大学）
楠本 真二（大阪大学）
栗田 太郎（ソニー）
後藤 徳彦（NEC ソリューションイノベータ）
古畑 慶次（デンソー）
小林 展英（デンソークリエイト）
阪井 誠（SRA）
酒匂 寛（デザイナーズデン）
菅原 広行（ソニー）
鈴木 裕信（usp.lab.）
鈴木 正人（北陸先端科学技術大学院大学）

実行委員会

鈴木 克明（熊本大学）
高木 智彦（香川大学）
田中 康（奈良先端科学技術大学院大学）
張 漢明（南山大学）
角田 雅照（近畿大学）
土肥 正（広島大学）
富松 篤典（電盛社）
中谷 多哉子（放送大学）
中西 恒夫（福岡大学）
中森 博晃（パナソニック）
西 康晴（電気通信大学）
布川 博士（岩手県立大学）
根本 紀之（東京エレクトロン）
野呂 昌満（南山大学）
端山 毅（NTT データ）
久住 憲嗣（九州大学）
藤川 昌雄（富士通九州ネットワークテクノロジーズ）
本多 慶匡（東京エレクトロン）
松尾谷 徹（デバッグ工学研究所）
松本 健一（奈良先端科学技術大学院大学）
水野 修（京都工芸繊維大学）
宮本 祐子（宇宙航空研究開発機構）
宗平 順己（武庫川女子大学）
村上 純（熊本高等専門学校）
森崎 修司（名古屋大学）
諸岡 隆司（中電シーティーアイ）
八木 将計（日立製作所）
山本 修一郎（名古屋大学）
米島 博司（パフォーマンス・インプルーブメント・
アソシエイツ）
劉 少英（法政大学）

事務局

栗田 太郎（ソニー）

ソフトウェア・シンポジウム 2020（オンライン開催） 目次

■論文・報告 ［要求］

■論文・報告 ［開発／深層学習］

■論文・報告 ［マネジメント］

■論文・報告 ［QA／派生開発］

■論文・報告［教育］

■論文・報告［形式仕様／モデル検査］

■論文・報告 [Future Presentation]

多様なステークホルダの満足度に着目したシステムの安全性・セキュリティ要件の抽出と検証

System Requirements Analysis for Safety and Security with Multiple Stakeholder Perspectives on STAMP/STPA

柳原 靖司
ブラザー工業株式会社
yasushi.yanagihara@brother.co.jp

吉田 邦雄
オムロン株式会社
kunio.yoshida@omron.com

石川 冬樹
国立情報学研究所
f-ishikawa@nii.ac.jp

栗田 太郎
ソニー株式会社
taro.kurita@sony.com

要旨

本論文では，ステークホルダの多様な視点を取り入れながら客観的かつ系統的にシステムの安全性・セキュリティ要件の抽出と検証を行う手段として，SSMS 法（Safety and Security requirements analysis method with Multiple stakeholder perspectives on STAMP/STPA）を提案する．近年，AI/IoT に代表されるようにソフトウェアが大規模・複雑化する中で，抜け漏れなくシステムの安全性・セキュリティ要件を抽出し，かつステークホルダの間で合意形成することが難しくなっている．SSMS 法を用いることで，対象ドメインの知識を十分に有してなくても複雑なシステムのハザードと脅威に対する対策群を実用的な量で獲得し，要件の満足度に関する評価指標で定量的に評価することができる．

This paper presents a safety and security requirements analysis method with multiple stakeholder perspectives on STAMP/STPA called SSMS Method. In recent years, as software has become larger and more complex, as represented by AI/IoT, it has become difficult to extract safety requirements comprehensively and to form consensus among stakeholders adequately. SSMS Method facilitates developers to acquire safety and security requirements without sufficient knowledge of the target domain and to evaluate them quantitatively based on their assent.

1. はじめに

システム開発の上流工程において抜け漏れなくシステムの要件を抽出し*1，ステークホルダ間で合意形成することは難しく，大規模・複雑化する先進システムの開発ではその傾向が顕著である．システム要件の中でも運用段階で生じるハザードや脅威に対応するための安全性やセキュリティといった要件の抽出は重要であり，次世代システム安全性解析手法のひとつである STAMP/STPA（Systems-Theoretic Accident Model and Processes/ System-Theoretic Process Analysis）による解決アプローチが注目されている．米国を中心に航空宇宙，プラントの分野で適用実績が積まれてきており，最近では AI/IoT といった領域にも応用範囲が広がっている．システムに関わる人，環境，機械といった要素をモデル化し，その相互作用に着目することで安全・安心を脅かす要因を系統的に解析していくが，解析者が持つ思考特性によって導出される結果に偏在性が生じる等，手法が持つ適用汎用性から生じる課題も分かってきている．そこで，我々はシステム構築の要求分析・要件定義の分野で用いられているステークホルダの合意形成プロセスの知見を STAMP/STPA に導入し，安全性とセキュリティの対策を網羅的に抽出できるようにした上で，要件の満足度の指標で定量的に評価するプロセスモデル SSMS 法（Safety and Security requirements analysis method with Multiple stakeholder perspectives on STAMP/STPA）を考案した．

*1 要件定義における「抜け」とは，ユーザ企業（発注元）からの要求に適合するように網羅的に要件化されていないこと．「漏れ」とは，考慮漏れのことで，ビジネス目標を達成する上で要件が要求を満足できていないこと（非機能要件の不完全性等）．

第三者による実験では，STAMP/STPA に多様なステークホルダの視点を導入することで抽出されるシステム要件の視点の豊かさが増し，かつドメインの有識者でなくても量と質（目的合理性）の点でドメインの有識者に近いシステム要件を客観的かつ系統的に抽出できることが分かり，SSMS 法の有効性が確認できた．以下，本論文の構成を述べる．

まず，2 章では現状分析と課題提起を行う．次に 3 章では関連技術について言及し，4 章，5 章で夫々，解決策の提案と評価結果を示す．6 章で評価結果に関する考察を行い，最後に 7 章で成果と将来への発展で結ぶ．

2. 解決すべき課題

2.1. 現状分析

AI/IoT に代表される複雑なソフトウェア集約型システムが登場しており，コンポーネント単位の信頼性を担保するような従来型の品質保証に加え，多数のコンポーネントが相互に連携するつながるシステムを俯瞰的に捉えながら，安全・安心を保障することが社会の課題として認識されている．この課題に対するアプローチとして，要求工学の分野ではステークホルダが連携しながらシステムのライフサイクル全体を分析するためのガイドラインが構築されている[1]．システムの安全設計・安全性論証の分野では STAMP/STPA[2]，アシュアランスケース等の活用研究が進められている．

2.2. 課題提起

現在，システミック・アプローチで複雑なシステムの安全性を解析する技術として STAMP/STPA が知られている．FTA（Fault Tree Analysis），FMEA（Failure Mode and Effect Analysis），HAZOP（Hazard and Operability Study）といった従来手法とは異なり，解析対象システムに影響を及ぼす人，環境，機械をモデル化し，構成要素の創発特性から生じるハザードを解析するアプローチを用いる．解析手法としての汎用性があるため，様々な産業のシステムに適用可能であるが，解析者が持つ思考特性により抽出されるハザードと脅威の対策群に偏りが生じる課題が本研究の予備実験から分かっている[3]．予備実験では文献[1]を参考にしながら，抽出された対策群を表 1 に示すような 3 つのステークホルダの視点で分類したが，例えばベンダの開発者であれば，システムの実装に関する対策群が解析の結果として抽出されやすい傾向があり，その他の視点は検討の優先順位が下がる．システム開発の要求分析や要件定義では，多様なステークホルダ間の合意形成を促進することが望ましいので，システムの安全設計・安全性論証の領域にも要求工学の知見を取り入れる必要がある．

3. 関連技術の説明

我々の研究の技術的拠り所として用いている STAMP/STPA 及び AGORA[4]（Attributed Goal-Oriented Requirements Analysis Method）について述べる．

3.1. STAMP/STPA

STAMP/STPA では，図 1 に示すようなコントロールストラクチャ（CS: Control Structure）を用いてシステムの構成全体をモデル化し，コントローラから被コントロールプロセスに対して発行されるコントロールアクションの乱れ（UCA: Unsafe Control Action）に着目しながら，安全制約を逸脱するようなハザード要因（HCF: Hazard Causal Factor）を解析していく．ハザード要因の解析には STPA の中で予め提供されているガイドワードを利用することで，様々な視点で HCF を強制発想できるように工夫されている．HCF が抽出された後は，各 HCF に対する対策群を立案することで解析が完了する．当初の STAMP/STPA

表 1 各ステークホルダが持つ視点

ステークホルダ	ステークホルダが持つ視点
ユーザ企業の管理者	システムを実現する上での制約よりも，システム稼働後のビジネス収益性や業務変革の視点から価値を最大化することを目的とする．災害時の事業継続やセキュリティリスクに高い関心を持つ．
ベンダの開発者	受託企業としてユーザ企業とのトラブルは避けたい．QCD のバランスに配慮して手堅く進めたい．無理な要件の実装は避けたい．
ユーザ企業の担当者	現場業務の改善（業務の効率化）の思考が強く，やや保守的な変化を好む傾向を示す．要件定義に当たっては既存業務との整合性を重視し，業務変革レベルの視点が弱い．

では機能安全に関する領域で研究が進められていたが，その後，システムのセキュリティに関する脅威を解析することができるようにプロセスモデルが拡張されている[5][6].

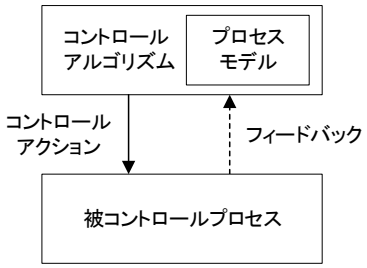


図 1 STAMP/STPA CS の例

3.2. AGORA（属性つきゴール指向要求分析法）

AGORA とはゴール指向要求分析手法のひとつで，AND-OR ツリーグラフ（図 2）に属性値を付与した上で，下流に向かって対案となるサブゴールを展開していきながら，ステークホルダ間の要求獲得に関する合意形成を促進させるための手法である．属性値のひとつとして満足度行列（図 3）があり，例えば，各ステークホルダが顧客・管理者・開発者の視点でサブゴールの評価値を 3 行 3 列の行列に登録すれば，ユーザに関する要素の総和からゴール適合度（式 1）を求めたり，列方向の要素の分散値から意見の対立度を求めたりすることができる．

なお，ゴール指向要求分析手法には AGORA 以外にも NFR（Non-Functional Requirements）フレームワーク[7]，GSN（Goal Structuring Notation）[8]等が知られているが，先行研究[9]の成果を踏まえ，AGORA の満足度行列が要件のゴール適合度を求めたり，ステークホルダ間の意見対立を可視化したりする上で分かり易く，かつ現場の実務に導入し易いと考えた為，AGORA を採用した．

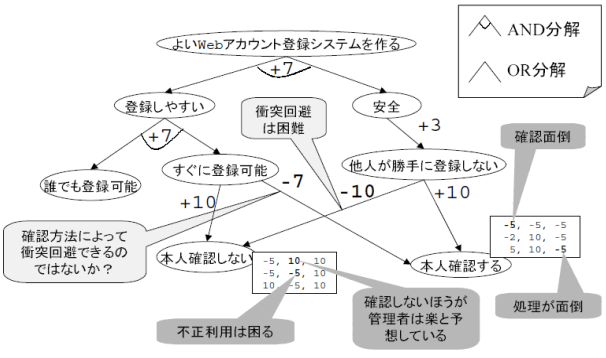


図 2 AND-OR ツリーグラフ[4]

評価の視点

評価者	顧客	管理者	開発者
顧客	8	8	(4)
管理者	5	5	2
開発者	4	8	-6

凡例：(4) 顧客が開発者の視点で要求を評価した結果

図 3 満足度行列[4]

$$Cup(g) \stackrel{\text{def}}{\equiv} \frac{\sum_{s \in Stakeholder, c \in Customer} m(g)_{s,c}}{|\,Stakeholder\,| \times |\,Customer\,|}. \quad (1)\,[10]$$

$Cup(g)$は，ゴールgに対する顧客の選好度，$Stakeholder$は全てのステークホルダ（顧客，管理者，開発者），$Customer$はユーザに関するステークホルダ（顧客，管理者）である．式(1)の分子はゴールgの満足度行列に含まれる$Customer$の要素の総和，分母は$Stakeholder$と$Customer$に関する集合の濃度の直積集合として計算すれば$Cup(g)$が求まる．

図 3 の満足度行列を例に計算方法を示す．

$Cup(g)$：6.3

分子：38

$Customer$に関する素点の合計値

{顧客の素点}＋{管理者の素点}

= {8 + 5 + 4} + {8 + 5 + 8}

分母：6

次の値の平方根

{ $Stakeholder$の濃度：9}×

{ $Customer$の濃度：4}

※集合の濃度とは，個数である．

4. 解決策の提案

4.1. 課題の解決方針

我々は 2.2 節で取り上げた課題を解決するため，SSMS 法を提案する．SSMS 法は多様なステークホルダの視点を STAMP/STPA に導入しながら，客観的かつ系統的にシステムのハザードとセキュリティ脅威の対策群を抽出した後，これらを安全性・セキュリティ要件として AGORA の満足度行列を用いて評価するプロセスモデルである．このモデルが狙っている効果は，システム安全性

の解析結果(STAMP/STPA の対策群)の多様性を創出することであるが，解析者が持つ思考特性がシステムの安全性解析結果に偏在性を生じさせる性質に対応するために，予めシステムの開発において利害関係が生じやすい複数のロールの立場で思考制約を設けてSTAMP/STPA による解析を実行する．SSMS 法では，図 4 に示すように管理者，担当者，開発者の視点を導入している．STAMP/STPA による解析結果で得られた対策群をシステムの安全性要件として実装に移していく過程では，どの要件を採用するべきか，ステークホルダ間で合意形成が必要である．SSMS 法では，抽出された対策群を要件の満足度の観点で評価した後，要件毎に満足度行列の形で整理し，さらに「ゴール適合度」と「意見対立の相関」の2つの指標で定量化する．要件の質(目的合理性)が定量化できるので，システムの安全性・セキュリティ要件の決定プロセスにおいて，ステークホルダ間の合意形成促進に貢献できる．

なお，SSMS 法のプロセスモデルでは，要件の抽出プロセスに AGORA のツリーグラフではなく，STAMP/STPA のシステミック・アプローチを導入している．これは AI/IoT に代表される先進システムの要素間の複雑なインタラクションからもたらされる創発特性を捉えやすくすることを期待している．特に人とシステムの相互作用に関わる解析では，ツリーグラフのように上流から下流に展開される構造モデルでは解析が困難だからである．また，STAMP/STPA を導入する別の利点として，熟練技術者が持つ業務知識や知見への依存度を抑える役割もある．STPA の手順化されたプロセスによって，技術の専門家ではないユーザ企業の管理者や担当者をシステム安全性解析フェーズに巻き込み易くなると考える．以上の点から SSMS 法では，多様なステークホルダの視点を活用できる点で従来手法に比べ優位性がある．

4.2. SSMS 法の解析手順

図 4 に示すプロセスに従い，システムの安全性・セキュリティ要件の抽出と検証を行う．

STEP1: システムのゴールを明確化し，システム構成図を作成する．

STEP2: STAMP/STPA の解析に必要なアクシデント，ハザード，安全性制約を列挙する．

STEP3: システム構成図から構成要素，コントロールアクション，フィードバックを特定し，コントロールストラクチャを定義する．(モデル化)

STEP4: システム開発に携わるユーザ企業の管理者，担当者及びベンダの開発者が，STAMP/STPA を用いて夫々，解析を行う．このとき，各解析者は自身の役割(ロール)を適切に意識し，ロール毎の思考制約の中で解析を行う．(UCA 抽出→HCF 特定→システムのハザードとセキュリティ脅威に対する対策群の立案)

STEP5: 対策群を集約し，システムの安全性・セキュリティ要件としての評価を行う．要件を定量化する手法として AGORA の満足度行列を用い，前記ステークホルダが自身と他のステークホルダの視点で-10から+10の範囲で要件としての満足度を評価し，素点を付ける．そして，満足度行列から「ゴール適合度」と「意見対立の相関」を計算することで定量化を行う．

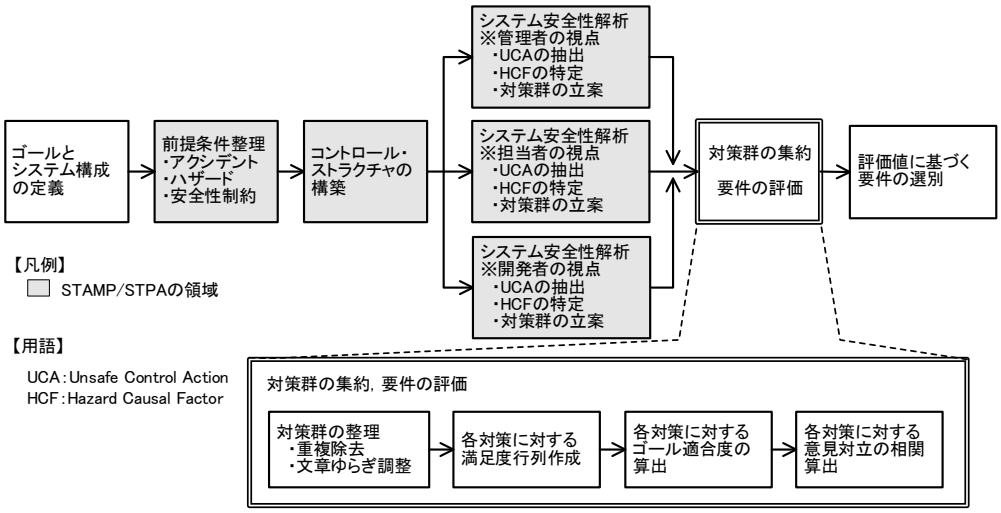


図 4 SSMS 法のプロセスモデル

STEP6: ステークホルダの間で要件の評価値を見ながら要件を選別する.

4.3. 仮説と研究設問

SSMS 法で解決しようとする仮説と研究設問を述べる.

[仮説]

多様なステークホルダの視点を STAMP/STPA に導入してシステムのハザードと脅威に対する対策群を抽出し, これらを要件としてゴール指向で分析すれば, 客観的かつ系統的に合意形成されやすい安全性・セキュリティ要件を獲得できる.

[研究設問]

RQ1:解析者にロールを与えてシステムのハザードとセキュリティ脅威を解析することで, 多様な視点(異なるステークホルダの視点)で安全性・セキュリティ要件を獲得できる.

RQ2:ドメイン知識を十分に有しなくても複数のロールを持たせて SSMS 法で解析することで, 当該ドメインの有識者と同等量の安全性・セキュリティ要件を獲得できる.

RQ3:ドメイン知識を十分に有しなくても複数のロールを持たせて SSMS 法で解析することで, 当該ドメインの有識者が抽出した要件に近い質(目的合理性)の安全性・セキュリティ要件を獲得できる.

5. 解決策の評価

5.1. 評価方法

仮説の検証は, 図 5 に示す仮想の QR コード決済システム(以降,「対象システム」と呼ぶ)に対して RQ1, RQ2, RQ3 の妥当性を確認することで実施した. 対象システムは Enterprise システムの一種であるが, システムのハザードと脅威の解析に当たっては, 端末とセンターサーバ間で発生するトランザクションの他, 人とシステムの協調という総合的なシステム運用に関する知識が要求される.

今回, 対象システムの解析を実施する上で, 第三者の被験者 I群, II群(IIa 群, IIb 群), III群を用意した(表 2). 各被験者は独立で, 夫々, 表 2 に示すような役割と前提

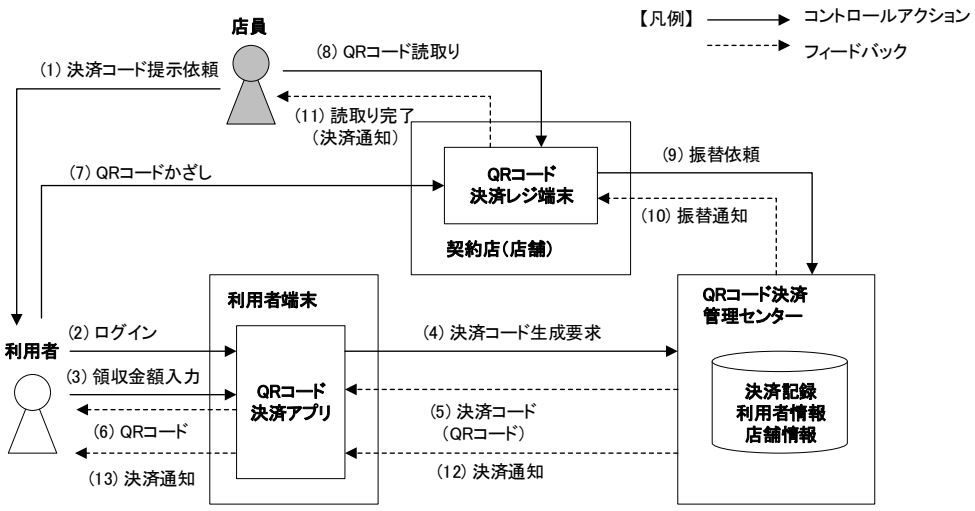


図 5 仮想 QR コード決済システム(STAMP/STPA のコントロールストラクチャ(CS))

表 2 被験者の役割と前提条件

被験者	役割	条件
I群	解析と対策群の抽出	3 名; Enterprise システム開発に非精通; 特定ロールを不付与
IIa 群	解析と対策群の抽出	3 名; Enterprise システム開発に精通; 特定ロールを付与
IIb 群	解析と対策群の抽出	3 名; Enterprise システム開発に非精通; 特定ロールを付与
III群	対策群の評価(定量化)	3 名; Enterprise システムの技術を解釈可; 特定ロールを付与

[補足] 特定ロール:ユーザ企業の管理者, ユーザ企業の担当者, ベンダの開発者

条件を持つ. 実験では, ロールなしの被験者(Ⅰ群)についてはロールを意識しないで対策を抽出してもらい, 他方, ロールありの被験者(Ⅱ群)については夫々対応するひとつのロールを意識してシステムを解析してもらった. その上で, 解析結果として得られたシステムのハザードと脅威に対する対策群に対し, 第三者(Ⅲ群)がロールを考慮しながら評価した. 解析時間の目安として, KKD(解析者の経験と勘から対策を見つける方法) は 120 分, STAMP/STPA は 240 分*2とし, 基準時間の目安から10%を超過した場合には実験を再試行することにしたが, 今回の実験では各被験者の抽出時間はすべて基準時間内に収まった.

ここで, 各被験者のドメイン知識とロールに関する特記事項を示す.

[ドメイン知識]

・Enterpriseシステム開発に精通(Ⅱa群):
Enterpriseシステムのベンダに所属し, 開発経験あり

・Enterpriseシステム開発に非精通(I群, IIb群):
車載システムもしくは産業システムのベンダに所属し, 開発経験あり

[ロールに関する実務経験]

・ユーザ企業の管理者:
所属する企業の役職が管理職であり, ユーザ企業の管理者の視点を理解できる.

・ユーザ企業の担当者:
ベンダの開発者としてユーザ企業の担当者のシステム開発における要求を理解できる.

・ベンダの開発者:
ベンダの開発者としてシステム開発に求められる基礎的な知識と業務経験がある.

5.2. 評価結果

[RQ1 に関する評価結果]

特定ロールを付与しない被験者I群と特定ロールを付与した被験者II群に対象システムのハザードとセキュリティ脅威を解析してもらい, 対策の抽出量を比較した.

図6は各被験者の対策抽出量であるが, 被験者にロールを付与すると, 保有するドメイン知識の量(Enterpriseシステム開発の精通度)に係わらず付与したロールに対応する対策の抽出量が最多となった. 逆に, 被験者にロールを付与しないとそのような傾向がみられなかった. 特記事項として, Ⅱa群のロールあり(管理者)とⅡb群のロールあり(開発者)に該当する被験者は, 他の被験者と比べ対策群を2倍以上抽出したが, これは個人差から生じていると推察される. 例えば, Ⅱa群の管理者ロールを担当した被験者は, 国内大手のSIerで20年以上に渡りシステム開発に従事し, 所属企業ではベテラン管理職としてプロジェクト横断で業務支援を行っていることから, Enterpriseシステム開発の精通度が高い特性があった.

次に, 図7では各被験者が抽出した対策群をⅠ群とⅡ群でロール属性ラベル毎に合計し, ロールの有無によって抽出量に与える影響を調べた*3. 同図から分かることは, ロールを付与しないと, 管理者のロール属性ラベルに該当する対策抽出量が少なくなる点である. Ⅰ群では

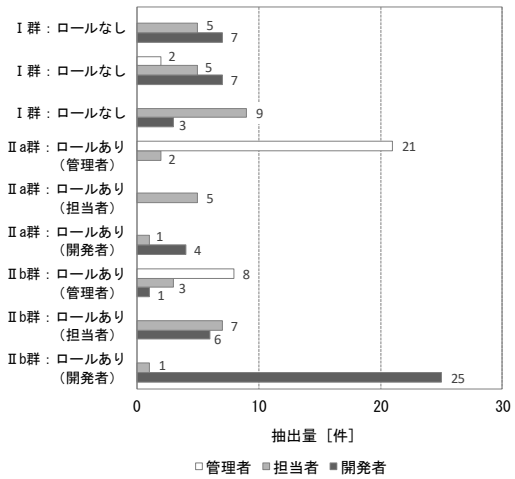


図 6 被験者毎の対策抽出量

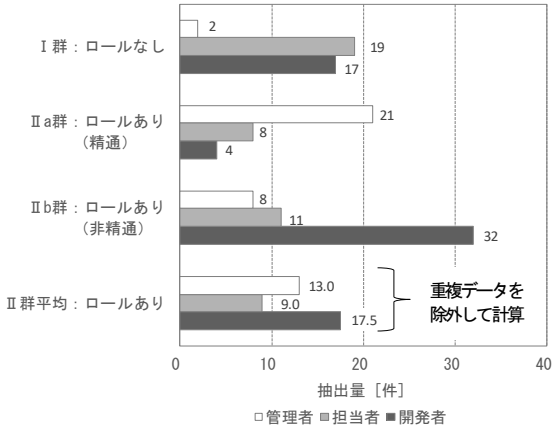


図 7 ロール有無毎の対策抽出量

*2 STAMP/STPA による抽出では専用ツールを用いて対話型の解析を行う為, KKD の2倍の時間を設定した.

*3 I群とII群の被験者数を揃えて比較する為, II群については重複データを除外してからIIa 群とIIb 群の抽出量を平均した.

合計抽出量 38 件の 5.3%(2 件)が管理者のロール属性であるのに対して，Ⅱ群では 32.9%(13 件)がそれに該当した．実際，Ⅰ群の被験者にインタビューしたところ，システムの安全性・セキュリティ要件の解析に当たって思考制約を与えないと，現場の視点での抽出が容易な担当者や開発者に関するロール属性ラベルの対策が優先的に検討され，管理者の視点での検討が後回しにされることが分かった．以上のことから，システムの解析時にロールという思考制約を与えることで，そのロールに適合する視点で解析された対策が抽出され易くなると言える．ここで，本実験より抽出された対策群の例を表 3 に示す．同表に示したものは各ステークホルダの視点で抽出された対策群の一部であるが，システムの安全性とセキュリティに関する対策が抽出されていることが分かる．

[RQ2 に関する評価結果]

ドメイン知識を有する被験者IIa 群とドメイン知識を有しない被験者IIb 群の夫々が KKD と STAMP/STPA を適用し，対象システムのハザードとセキュリティ脅威を解析した．図 8 はⅡa 群とⅡb 群の被験者が抽出した対策群を対策立案部位毎に纏めたものである．対策立案部位とは，図 5 におけるコントロールもしくはフィードバックに該当し，(1～13)と共通部を加えて計 14 箇所ある．

まず，各被験者群が抽出した対策立案部の網羅率に着目すると，Ⅱa 群は 10 箇所(1,2,3,4,5,8,9,10,12,共通)から抽出しているのに対し，Ⅱb 群は 7 箇所(1,2,3,4,7,8,9)から抽出している．網羅率を計算すると，前者が 71%で後者が 50%であり，ドメインの知識を有しない被験者Ⅱb 群の抽出能力が低くみえるが，これは，STAMP/STPA の解析プロセスがコントロールアクションに対して解析を行う仕組みであることが起因している．図 5 のコントロールアクション 7 箇所(1,2,3,4,7,8,9)に限定すると，Ⅱb 群の網羅率は 100%である．一方，KKD で解析したⅡa 群は，コントロールアクションに加えてフィードバックからも対策を抽出している点で幅広い部位から抽出できることが分かった．次に，全体の抽出量に着目すると，Ⅱa 群は 33 件であるのに対して，Ⅱb 群は 51 件であり，1.5 倍多く抽出できている．特に，「QR コード読み取り」や「振替依頼」からの抽出量が多いが，被験者の解析内容を調べると STAMP/STPA の UCA を精緻に掘り下げて分析しており，ドメインの知識が少ない解析者であっても STAMP/STPA の解析プロセスにあるヒントワードを使うことで抽出する対策のバリエーションを増やすことができていた．ここで，STAMP/STPA を使った具体的な解析事例を表 4 に示す．担当者の視点にある「QR コードかざし」に関する対策は KKD によるⅡa 群の被験者からは抽出されなかったが，STAMP/STPA によるⅡb 群からは 8 件抽出された．表 4 の解析プロセスをみると，対象システムの制約として提示された「決済に要する時間は現金決済よりも速い」に対して，「利用者が認知しづらい QR コード読取り完了の表現手段を採用した」為，「QR コードの読取り完了前に，かざし動作を止めてしまう」という因果のシナリオに基づいて対策を導出できていることが分かる．STAMP/STPA のモデルベースの解析プロセスが解析者の対策抽出を助長しているものと考えられる．

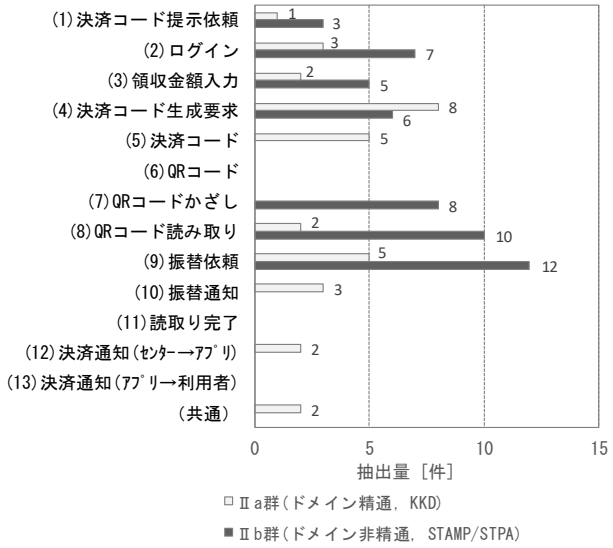


図 8 ドメイン精通度と解析手段に応じた対策抽出量

表 3 各ステークホルダの視点から抽出された対策の例

抽出された対策例	対策の種別	対策の視点
BCP（事業継続計画）の観点からディザスタリカバリ，業務継続拠点の確保を検討する．	安全性	管理者
QR コード決済レジ端末は搭載するセンサの自己診断を行い，店員に結果を提示する．	安全性	担当者
決済管理センターから時間内に振替通知を受信しなかったら，自己診断処理を起動する．	安全性	開発者
ユーザの個人情報の漏洩対策を強化する．	セキュリティ	管理者
ログインされたときは利用者に通知を送信する．	セキュリティ	担当者
決済が済んだデータはクリアする．	セキュリティ	開発者

表 4 STAMP/STPA による対策の抽出プロセス例

対策の視点	管理者	担当者	開発者
図 5 CS 該当矢印	(1) 決済コード提示依頼	(7) QR コードかざし	(8) QR コード読取り
UCA	QR コード決済の意図や行動のタイミングが伝わらず, 利用者が困惑する.	QR コードの読取り完了前に, かざし動作を止めてしまう.	当該店舗での取引と無関係の QR コードを読み取ってしまう.
違反する制約	決済に要する時間は現金決済よりも早い.	決済に要する時間は現金決済よりも速い.	決済金額を間違ってはならない.
HCF	店員から利用者に対する QR コード提示の働きかけ方が判りづらい.	利用者が認知しづらい QR コード読取り完了の表現手段を採用した.	過去に生成された QR コードがアプリ内部に保持されており, 誤使用される.
対策	接客応対に関するオペレーションの習熟をはかる. (教育活動の強化)	LED や効果音等の表現手段により, 利用者に読取り完了状態を伝える.	QR コード生成時刻等を QR コードに含め, 期限切れの場合に警告表示する.

[RQ3 に関する評価結果]

被験者 IIa 群と被験者 IIb 群の夫々が KKD と STAMP/STPA を適用し, 対象システムのハザードとセキュリティ脅威を解析した.

まず, 前処理として, 抽出された対策群の中から, 表 5 に示す 12 個の対策区分*4 に該当する KKD と STAMP/STPA の対策群に対して AGORA で用いられている満足度行列を作成した. そして, この満足度行列に基づいて「ゴール適合度」と「ステークホルダの意見対立の相関*5」を各対策について求めた. ここで, ゴール適合度とは, 満足度行列におけるユーザ企業の担当者と管理者の数値の合計値を正規化した値である. ステークホルダの意見対立の相関とは, 満足度行列における列方向の 2 系列の共分散である. 以下, 統計処理により KKD と STAMP/STPA のサンプルに有意な差が無いことを検証した結果を述べる. 検証のゴールは, 各サンプルの母集団が正規分布に従っていると仮定した上で, KKD と STAMP/STPA から得られた系列の母集団に差が無いという帰無仮説が正しいことを t 検定により確認することである. 少数サンプルの検定では系列群の等分散性と観測者の独立性が重要なので, F 検定の計算値により等分散性を判断し*6, 被験者が所属する産業分野以外の因子(会社, 役職, 業務, 年齢等)が異なることを確認することで独立性を判断してある.

STAMP [#1-#12]と KKD [#1-#6]*7について t 検定を実施した結果, ゴール適合度については検定統計量が 0.17(＞0.05), 意見対立の相関については検定統計量が 0.10(＞0.05)となり, いずれも KKD と STAMP/STPA の2つのサンプルに有意な差が無いことが確認された. 一方, 被験者Ⅱb 群のみが抽出した STAMP [#7-#12]と KKD [#1-#6]でt検定を行うと, 2つのサンプルに有意な差があることが確認された. このことから, Ⅱb 群が抽出した対策群の全体では, ドメイン知識を十分に有しなくても STAMP/STPA を用いることで, 有識者が経験と勘で解析した結果に近い質(目的合理性)の対策群(安全性・セキュリティ要件)を獲得できる場合もあるが, ドメインの知識量が少ない解析者が関わると成立しない可能性があると言える.

具体的に, Ⅱb 群の被験者が STAMP[#7-#12]で抽出した対策群のひとつを確認してみると, 「#7 店舗口座設定は, 郵送(簡易書留)で行う」(ゴール適合度: -4.67, 意見対立の相関: 58.33)というものがある. Enterprise システム開発に精通した技術者の視点で判断すると, #7 の対策にはシステム運用時のコストや煩雑さに課題があると推

*4 対策区分は, 対策立案部から抽出された対策の視点から分類したものである.

*5 満足度行列における列方向の「評価の視点」の2つの系列(対策立案者の系列と, それとは分散値が最も離れているもうひとつの系列)の共分散である. 計算値が大きい場合, 各ステークホルダで該当要件に対する満足度が対立している状態になる.

*6 F 検定(片側確率)の結果, ゴール適合度については 0.11(＞0.05)で等分散が確認されたが, 意見対立の相関については 0.04 (＜0.05) で等分散性が確認されなかった. 従って, 後者については Welch の t 検定を用いた.

*7 Ⅱa 群の被験者は, 対策区分#7-#12 に該当するものを抽出しなかった為, 表 5 に数値が入っていない.

表 5 各対策の評価値(左:ゴール適合度, 右:意見対立の相関)

ゴール適合度

#	対策区分	STAMP	KKD
1	認証機能	6.67	6.83
2	認証機能②	5.00	3.67
3	金額確認	6.17	6.33
4	店舗確認	3.17	3.50
5	QR ｺｰﾄﾞ改竄・破損ﾁｪｯｸ	3.50	3.67
6	性能向上	7.17	7.33
7	店舗登録	-4.67	-
8	店舗登録②	0.67	-
9	誤操作防止	2.83	-
10	誤操作防止②	2.67	-
11	障害対策	2.83	-
12	障害対策②	2.67	-
	平均	3.22	5.22
	分散	9.91	3.22

意見対立の相関

#	対策区分	STAMP	KKD
1	認証機能	8.00	6.33
2	認証機能②	6.00	6.50
3	金額確認	10.00	10.00
4	店舗確認	13.00	23.00
5	QR ｺｰﾄﾞ改竄・破損ﾁｪｯｸ	1.33	1.33
6	性能向上	1.00	1.00
7	店舗登録	58.33	-
8	店舗登録②	39.00	-
9	誤操作防止	35.17	-
10	誤操作防止②	11.00	-
11	障害対策	9.67	-
12	障害対策②	35.33	-
	平均	18.99	8.03
	分散	333.59	65.49

察されるが, Ⅲ群の(管理者ロールの)評価者についても各ステークホルダの視点で満足度行列に-10pt.の素点を 3 つ付与しており, システムの安全性・セキュリティ要件の質(目的合理性)を低下させ, 意見対立の相関を高めた.

6. 考察

6.1. 得られた知見

[研究設問に対する評価]

5.2 節に示す評価結果によれば, 解析者に異なるロールを与えて多様な視点でシステムのハザードとセキュリティ脅威を解析すれば, 獲得できる安全性・セキュリティ要件の多様性が増し(RQ1), さらにシステムの解析手段として SSMS 法を適用することで量と質(目的合理性)の点で有識者が抽出したものに近い安全性・セキュリティ要件が条件付きで得られることが分かった(RQ2, RQ3).

ここでの「条件」とは, システム解析者のドメイン知識の量や業務経験に関するもので, RQ3 に関する評価結果で述べたとおり, 対象システムの開発に精通していない解析者が SSMS 法を使用するとプロジェクトの QCD の点で有識者が暗黙知で除外するようなシステム要件を抽出する可能性があることから, SSMS 法を効果的に適用するのであれば, 解析者のドメイン知識を一定水準以上に揃えることが望ましい.

[提案手法の改良案]

今回, SSMS 法では, STAMP/STPA のようなモデルベースのシステム解析手法をベースに AGORA で利用されている満足度行列を用いて要件の質(目的合理性)と意見対立の相関を検証する手法を提案した. 5.2 節の実験では, STAMP/STPA は特定の対策立案部(領域)を深掘りして抜けのない対策抽出ができる一方で, KKD のように幅広い領域で万遍なく対策を抽出することが弱い課題が確認された. そこで SSMS 法の改良案として, 図 4 に示すプロセスモデルの「コントロールストラクチャの構築」の後に, 前段でロールありの KKD で対策を抽出してから, 後段でロールありの STAMP/STPA で対策を抽出するような仕組みを作れば, KKD と STAMP/STPA の両者の強みを併合できるのではないかと考えている.

6.2. 妥当性への脅威

[内的妥当性への脅威]

RQ3 に関する実験の検証で用いたサンプル数は計 18 個であるため, 統計処理を行う上で必ずしも十分とは言えない. ただし, 検定では母集団に有意な差が無いことを示す上で, 等分散性と観測値の独立性に配慮した.

[外的妥当性への脅威]

今回の実験はひとつのドメインに対して実施したものであるため, 手法の汎用性を示すためには異なるドメインでの実験が必要である.

7. まとめ

7.1. 成果

本研究ではステークホルダの視点を STAMP/STPA に導入し, SSMS 法としてモデルを構築することで, 解析者がドメインの知識を十分に有しなくてもハザードと脅威に対する対策群を実用的な量で獲得し, これらを要件の満足度に関する評価指標で定量化できることを示した. 本報告では Enterprise システムを例に挙げその有効性を評価したが, SSMS 法は STAMP/STPA が持つ汎用性を活かしながらプロセスモデルを拡張してあるので, 制御系が含まれる AI/IoT システムにも適用可能であると考える. 今後, 複雑化する先進システムの開発で STAMP/STPA が展開されていく際に, 多様なステークホルダの視点を用いたシステム安全性解析の考え方が取り込まれ, 解析の視点の抜け漏れ防止に繋がることを期待したい.

7.2. 将来への発展

・SSMS 法では, 前段で解析したハザードの深刻度(リスク)を抽出された対策群の評価にフィードバックしていない. ゴール指向要求分析で使われる指標に加え, リスクを考慮しながら総合的に対策群の良し悪しを判断する仕組みを考慮できるとよい.
・SSMS 法では, STAMP/STPA の解析時にステークホルダの視点を導入しているが, コントロールストラクチャを作成する段階から多様な視点を導入すると, 獲得される安全性・セキュリティ要件の多様性が更に増大する可能性がある.
・情報セキュリティの学術領域ではリスクコミュニケーションを考慮した定量分析が提案されている[11][12]. SSMS 法に本領域の知見を融合すると, モデルを精緻化できる可能性がある.

8. 謝辞

本研究は, 日本科学技術連盟 2019 年度 ソフトウェア品質管理研究会 研究コース 5(分科会:仕様と要求のエンジニアリング)の中で取り組んだ内容です. 所属分科会の荒木啓二郎アドバイザーには, 多方面にわたり御指導を賜りました. また, 研究コース 5 の研究員の皆様, オムロン株式会社の紺田隆一郎氏, 古賀純平氏には実験に御協力を頂きました. 関係者の皆様に厚く御礼申し上げます.

参考文献

[1] システム構築上流工程強化部会 システム化要求 WG, ユーザのための要件定義ガイド 第 2 版, 情報処理推進機構 社会基盤センター(2019)
[2] Nancy G. Leveson, “Engineering a Safer World – Systems Thinking Applied to Safety”, The MIT Press (2011)
[3] 柳原靖司, 吉田邦雄, 栗田太郎, 石川冬樹, 多様なステークホルダの視点を STAMP/STPA に導入する試み, AI/IoT システムのための安全性シンポジウム(2019)
[4] 海谷治彦, 佐伯元司, 海尻賢二, 属性つきゴール指向要求分析法, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス 101(673)(2002)
[5] William Young, Reed Porada, System-Theoretic Process Analysis for Security (STPA-SEC):Cyber Security and STPA, STAMP 2017 Conference(2017)
[6] 金子朋子, 髙橋雄志, 大久保隆夫, 勅使河原可海, 佐々木良一, 安全解析手法 STAMP/STPA に対するセキュリティ視点からの脅威分析の拡張提案, コンピュータセキュリティシンポジウム(2017)
[7] John Mylopoulos, Lawrence Chung, Brian Nixon, Representing and Using Nonfunctional Requirements: A Process-Oriented Approach, IEEE Transactions on Software Engineering, Vol.18 No.6 pp.483-497(1992)
[8] SCSC Assurance Case Working Group, Goal Structuring Notation Community Standard Version 2, https://scsc.uk/publications (2020 年 4 月 30 日)
[9] 新原敦介, 河野仁一, 海谷治彦, 佐伯 元司, ゴール指向要求分析を用いたステークホルダの対立の検出, 情報処理学会研究会報告, Vol. 2004, No. 30, pp.99-106(2004)
[10] 佐藤慎一, 石川冬樹, 猪原健弘, 貢献度と顧客のニーズに関する妥当性の間のコンフリクト検出指標, ソフトウェアエンジニアリングシンポジウム(2011)
[11] 佐々木良一, 石井真之, 日高悠, 矢島敬士, 吉浦裕, 村山優子, 多重リスクコミュニケータの開発構想と試適用, 情報処理学会論文誌, Vol.46, No.8, pp.2120-2128(2005)
[12] 佐々木良一, 日高悠, 守谷 隆史, 谷山充洋, 矢島敬士, 八重樫清美, 川島泰正, 吉浦裕, 多重リスクコミュニケータの開発と適用, 情報処理学会論文誌, Vol.49, No.9, pp.3180-3190(2008)

要求仕様の誤解釈を検出する Domain Word Modeling の提案

柏原 一雄
株式会社デンソークリエイト
kazuo.kashiwabara.j3a@jpgr.denso.com

長井 亘
テックスエンジソリューションズ株式会社
nagai.wataru.d1@tex-sol.com

不破 慎之介
株式会社デンソークリエイト
shinnosuke.fuwa.j8h@jpgr.denso.com

林 香織
株式会社デンソークリエイト
kaori.hayashi.j4e@jpgr.denso.com

石川 冬樹
国立情報学研究所
f-ishikawa@nii.ac.jp

栗田 太郎
ソニー株式会社
taro.kurita@sony.com

要旨

仕様が自然言語のような曖昧さを含む記法で記述されている場合，仕様の誤解釈を誘発しやすい．実際に我々の組織では，「抽象的に表現された用語」を「不足している前提知識」で解釈することにより，誤解釈が発生していた．誤解釈の可能性のある用語を検出するために，用語の解釈を可視化する手法「Domain Word Modeling」を考案した．「Domain Word Modeling」は，導入のしやすさを重視した手法である．実験で，実開発において定義された要求仕様を入力として，考案手法を実行し，効果を確認した．

1. はじめに

ソフトウェア開発において，仕様は関係者の間の情報ハブ[1]となり，仕様策定者は多数の関係者と仕様によりコミュニケーションをする．仕様が自然言語のような曖昧さを含む記法では他の人にわかるように表現することは難しく，用語の曖昧さや関連の曖昧さが，読み手の解釈を必要とし，異なる理解による問題を生じさせることが，文献[1]で述べられている．

実際に我々の組織では，要求仕様の誤解釈による手戻りが発生していた．誤解釈は，「抽象的に表現された用語」を「不足している前提知識」で解釈することにより発生していた．要求仕様のレビューは実施していても，このような用語の誤解釈を防止しきることはできていなかった．

本研究では，要求仕様の誤解釈による手戻りを減らすことを目的とし，誤解釈を誘発する可能性のある用語を検出するための手法を考案した．考案手法は，導入障壁をできる限り低くし，コストやスケジュールの制約が厳しいプロジェクトでも効果を得やすくした．

考案手法は，「Domain Word Modeling」と呼び，モデルの記述ルールとモデリングの手順を技術要素とする．考案手法で作成する Domain Word Model は，要求仕様と前提知識を入力として，用語の関連を木構造で表現する．また，手法の導入障壁をできる限り低くするため，先行研究の「自然言語による仕様記述の改善のアプローチ[2]」と「マインドマップモデリング[3]」を参考にしている．

実験では，実開発において自然言語で記述された要求仕様を入力として，考案手法を実行し，考案手法が以下の要求を満たすかを評価した．

- レビューで見逃した「誤解釈を誘発する用語」を検出できる
- 要求仕様を自然言語で記述しているプロジェクトに対して導入しやすい

実験の結果，誤解釈を誘発する「抽象的に表現されている曖昧な用語」と「解釈のための前提知識が不足している用語」を検出できることを確認した．また，実開発で使用されている要求仕様の形式を変更せずに手法が実行できること，Domain Word Modeling 未経験でも手法の説明を受けただけで実行できることを確認した．

実開発において，Domain Word Modeling を活用することで，要求仕様の誤解釈による手戻りを減らす効果が期待できる．

これ以降の本稿の構成は次のとおりである．2 章で現

状分析の結果と研究の課題を示す. 3 章では, 課題解決の参考とした先行研究を示す. 4 章では, 考案した解決策を提案する. 5 章では, 提案手法の評価結果と考察を示す. 6 章では, まとめと今後の進め方を示す.

2. 課題設定

2.1. 現状分析

(1) 要求仕様の誤解釈の要因

我々の組織では, 要求仕様のレビューを実施していても, 要求仕様の誤解釈が防止しきれず, 手戻りが発生していた. 要求仕様を誤解釈した事例を分析すると,「抽象的に表現された用語」を「不足している前提知識」で解釈していることにより発生していた.

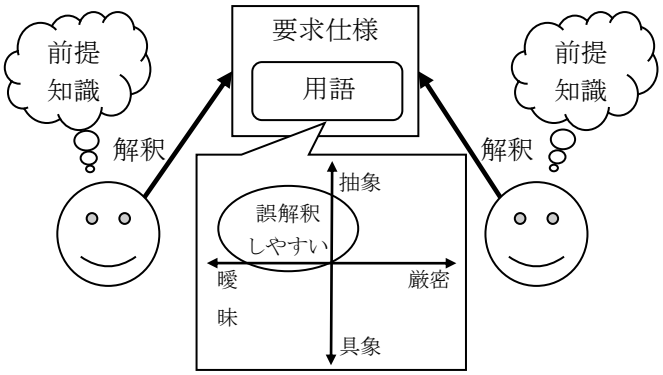


図 1 誤解釈のメカニズム

特に, 派生開発[4]では,「抽象的に表現された用語」を「不足している前提知識」で解釈する状況になりやすく, 要求仕様の誤解釈が発生しやすいと考えられる.

文献[4]では,「部分理解」の下で作業しなければならないことが, 派生開発の特徴として示されている. 派生開発においては, 担当者がベースソフトを部分的にしか理解できていない状態, つまり前提知識が不足している状態で, 開発する状況になりやすい.

また, 文献[5]では, 派生開発において定義される変更要求仕様では, ベースソフトから変化しない既存仕様が抽象的に表現されやすく, テストケースの漏れを誘発しているという調査結果が示されている.

(2) 誤解釈を誘発する用語

我々の組織で, 要求仕様の誤解釈を引き起こした事例を調査した. その結果, 抽象的に表現された用語が誤解釈を誘発していることがわかった. また, 調査により抽出された用語は, 単に抽象的というだけでなく, 他の用語と関連がある用語であった. つまり, 単独の用語ではなく用語間の関連が把握できていなかった.

実際に誤解釈を誘発した用語の具体例を表 1に示す. 表 1の用語を分類すると, 次の 3 つの種類に分けられる. どれも抽象的に表現された用語であり, 種類・構成要素・対象を示す用語との関連がある.

a. 複数の種類がある概念
b. 複数の構成要素からなる概念
c. 対象によって具体的な行為が変わる概念

表 1 誤解釈を誘発した用語の例

用語	説明	分類
ユーザ	X ユーザ,Y ユーザなど, 複数の種類がある	a
支店	X 支店,Y 支店など, 複数の種類がある	a
英語	米国英語,英国英語,豪州英語など, 複数の種類がある	a
原点	左上原点,右下原点など, 複数の種類がある	a
改行	CR+LF,LF,CR など, 複数の種類がある	a
診断情報	DTC,Freeze frame data など, 複数の構成要素から成る	b
記憶	対象データや対象製品によって記憶場所が変わる	c
開始	対象機能によって開始タイミングが変わる	c
中止	対象機能によって中止時に実施する処理が変わる	c

調査対象には, 業務システム・情報システムの要求仕様, 組込みソフトのミドルウェアの要求仕様など, 様々な種類が含まれている.

(3) 要求仕様のレビュー

要求仕様のレビューで,「抽象的に表現された用語」を問題として指摘することは難しい. また, 要求仕様の読み手にとって「不足している前提知識」を明らかにするこ

とも難しい.

レビューは, レビュー対象の成果物から, 問題を検出する行為である. レビューで, 人によって問題か否かの判断が分かれることを指摘することは難しい. また, 成果物に表現されていないことを指摘することも難しい.

厳密な仕様[1]を記述するときには, 記述する仕様の抽象度と, 記述の方法を決定する必要がある. 最適な抽象度は, 開発の目的や事情によって異なる. したがって, レビューで, 抽象的に表現された用語を, 一律に問題として指摘することはできない.

レビューでは, レビュー対象の成果物である要求仕様は確認されても, 要求仕様を読むうえで必要となる前提知識は可視化されておらず, 確認されにくい. したがって, レビューで,「不足している前提知識」を明らかにすることは難しい. 小林は, 文献[6]で, レビューで確認する情報の背景・前提を可視化する必要性を述べている.

2.2. 課題提起

本研究では, 要求仕様の誤解釈による手戻りを減らすことを目的とし, 誤解釈を誘発する可能性のある用語を検出するための手法を考案することを課題とする.

考案手法は, 導入障壁をできる限り低くし, コストやスケジュールの制約が厳しいプロジェクトでも効果を得やすくする. 考案する手法は, 以下の 2 つの要求を満たすものとする.

- レビューで見逃した「誤解釈を誘発する用語」を検出できる
  レビューで防止できなかった誤解釈は,「抽象的に表現された用語」を「不足している前提知識」で解釈することにより発生していた. 誤解釈を誘発する「抽象的に表現されている曖昧な用語」と「解釈のための前提知識が不足している用語」を検出できる手法とする.
- 要求仕様を自然言語で記述しているプロジェクトに対して導入しやすい
  プロジェクトに新たな手法を導入しやすくするためには,「プロセスを変更するコスト」「手法を習得するコスト」「必要となるツールを用意するコスト」を下げる必要がある.「プロセスを変更するコスト」を下げるため, 要求仕様の形式を変更せずに実行できる手法とする. 要求仕様の形式を変更しなければ, 要求仕様でやりとりをする関係者のコミュニケーションの仕方を変える必要がなくなる.「手法を習得するコスト」を下げるため, 考案手法を未経験の状態からでも, 手法の説明を受けただけで実行できる手法とする.「必要となるツールを用意するコスト」を下げるため, フリーソフトウェアで実行できる手法とする.

文献[7]で, あいまい表現が類型化されている. 定義されている類型の一つに「用語のあいまいさ」がある. 本研究では, この「用語のあいまいさ」が誘発する誤解釈による手戻りを減らすことを目的としている.

3. 先行研究

3.1. 自然言語による仕様記述の改善のアプローチ

大森らは, 文献[2]で, 自然言語による仕様記述の品質向上を目的として, 関係者の共通理解を得やすい自然言語記述と, 厳密な検証に基づく安定したモデル化を可能とする形式モデルのそれぞれの長所を両立させるアプローチを提案している. 図 2のように, 自然言語記述から形式モデルへ変換し, その形式モデル上で検証を行い, 検証結果のフィードバックによる自然言語記述の改善を繰り返し行うというアプローチである.

要求仕様を自然言語で記述している開発プロセスに対して, 要求仕様の形式を変更する必要がなく, 導入しやすいアプローチであると考えられる.

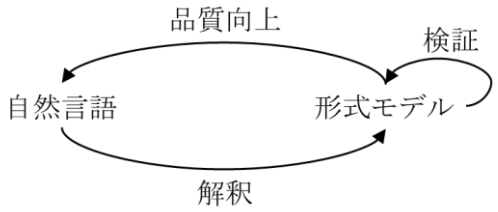


図 2 要求の洗練プロセス

3.2. シソーラス

用語の関連を表現するものとしてシソーラスがある.

加藤らは, 文献[8]で, シソーラスは, 要求分析者のドメイン知識を補完し, 要求の欠落を防止するために有用であることを示している. また, 文献[9]で, 日本語で書かれた技術文書を入力に, 形態素解析を活用し, シソーラスを構築する手法を提案している.

小川らは, 文献[10]で, シソーラスを作成する行為は, 関係者が相互の理解の相違を確認する手段となることを示している.

JIS X 0901:1991[11]では, シソーラスにおける概念間関

係が区別され，関係を表現するための略語が定義されている．以下に，シソーラスの記法の例を示す．

- 類種関係
  BTG　類種関係の上位語
  NTG　類種関係の下位語
  例:
    サル　BTG　霊長類
    霊長類　NTG　サル
- 全体部分関係
  BTP　全体部分関係の上位語
  NTP　全体部分関係の下位語
  例:
    中枢神経系　BTP　神経系
    神経系　NTP　中枢神経系

シソーラス及びその作成行為は，前提知識の不足を補う目的と前提知識の相違を明らかにする目的のために有用であると考えられる．ただし，シソーラスを活用するためには，シソーラスの記法を学習する工数や記法からの逸脱を指摘・修正する工数が必要となる．

3.3. マインドマップモデリング

文献[3]で説明されているマインドマップモデリングは，マインドマップを使って，自然言語からモデルを作成する手法である．自然言語から，UML によるモデルを直接作成するのは難しいという考えから考案された手法である．マインドマップモデリングは，以下の手順で実施する．

1. 文章の中から名詞を抽出する
2. 文章の中から動詞を抽出する
3. 名詞の構造（is-a 関係，has-a 関係）を定める
4. SVO（名詞-動詞-名詞）で考える

マインドマップモデリングは，木構造で用語をつなげるだけのシンプルな記法を使用している．そのため，記法を学習するコストはシソーラス等に比べると低いと考えられる．また，マインドマップを作成するためのツールがあれば利用可能な手法であり，ツールにはフリーソフトウェアも存在する．

4. 解決策の提案

4.1. 課題の解決方針

考案手法は，モデリング実施者とモデルのレビューアの間で解釈の相違を検出するために，用語を解釈するための前提知識を，モデルで可視化し，レビューする．モデルの作成とモデルのレビューを通して，「抽象的に表現されている曖昧な用語」と「解釈のための前提知識が不足している用語」を検出する．モデルは，要求仕様とそれを解釈するための前提知識を入力として，用語の関連を表現する．用語を解釈するための前提知識も，モデリングにより可視化されることで，確認が可能となる．

考案手法は，要求仕様の形式を変更せず実行できる手法にするために，先行研究の「自然言語による仕様記述の改善のアプローチ」を参考とした．要求仕様の検証をするために，要求仕様のレビューに加えて，用語の関連を表現するモデルの作成と作成したモデルのレビューをする．仕様策定者と「プロジェクトの外部」「プロジェクトの次工程」の担当者とのコミュニケーションは，手法導入前と変わらず，自然言語で記述された要求仕様を使用して行う．

考案手法は，作成するモデルをできる限りシンプルな

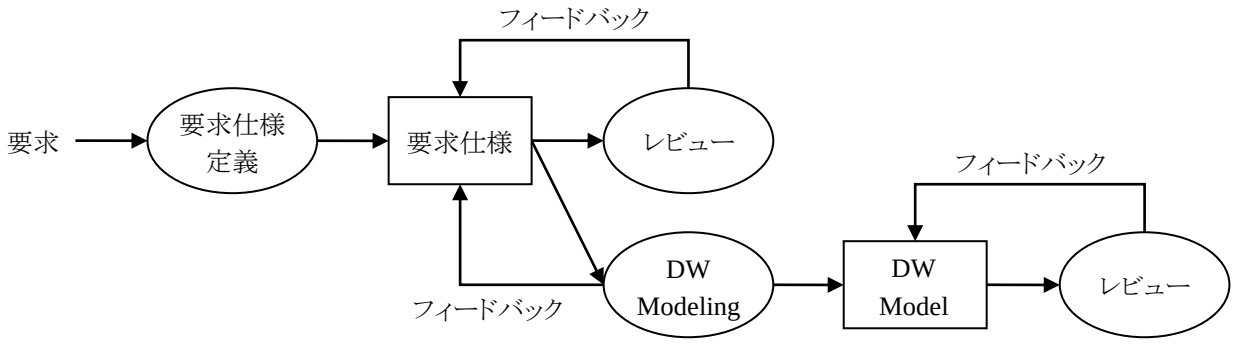


図　3 DW Modeling の概要

記法で表現するため,「マインドマップモデリング」を参考とした. マインドマップモデリングで作成されるモデルは, 概念と概念間のつながりを木構造で示すだけのシンプルなものであるが, is-a 関係と has-a 関係を表現し,「抽象的に表現されている用語」を特定することが可能である. また, マインドマップはフリーソフトウェアで作成することが可能である. マインドマップをベースとした手法とすることで, 手法を習得のためのコスト, 手法の実行に必要となるツールを用意するコストの両方が小さくなる.

考案手法は, Domain Word Modeling(DW Modeling)と呼び, 手法により作成された用語の関連を表現したモデルを Domain Word Model(DW Model)と呼ぶ. 図 3に, DW Modeling の概要を示す. 考案手法は, DW Model の記述ルールと DW Modeling の手順を技術要素とする.

図 4に, DW Model のイメージを示す. これは, テスト設計コンテスト'14 の ASTER 自動販売機ユースケース仕様書に登場する用語を入力に作成したモデルの一部である.

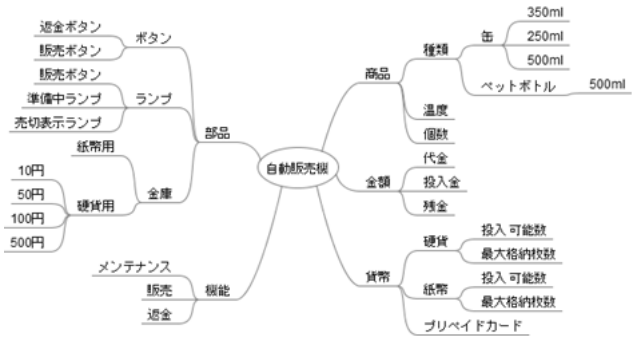


図 4 DW Model のイメージ

4.2. ドメイン用語の定義

DW Modeling では, 開発対象のシステムを定義するために必要となる用語を, ドメイン用語(Domain Word)と呼ぶ. ドメイン用語は, サ変名詞を含む名詞を対象とする. 対象によって具体的な行為が変わる概念もモデルに表現するため, 行為を示すサ変名詞もドメイン用語とする.

4.3. Domain Word Model の記述ルール

DW Model の記述ルールを以下に定義する.

- 同義語が存在しない状態にする. 同義語は 1 種類のみ登場させる.
- 各階層には, 集約(A has-a B)と汎化(A is-a B)の関係を混在させない.
- 必要に応じて要求仕様で使用されていない用語をモデルに登場させてよい.
- 末端の枝にのみ, 名詞ではなく, 文・式・値等を示してもよい.

4.4. Domain Word Modeling の手順

DW Modeling は,「1.形態素解析」,「2.モデリング」,「3.モデルのレビュー」の 3 ステップで, 要求仕様で使用されている用語の解釈を可視化する. 形態素解析では, ドメイン用語を漏れなく効率的に抽出する. モデリングでは, 抽象的な用語に対して下位概念となる用語を抽出し関連付け, 用語の解釈を可視化する. モデルのレビューでは, モデリング実施者とモデルのレビューアの間で解釈の相違を明らかにする.

DW Modeling の手順を以下に定義する.

1. 形態素解析
   要求仕様を入力に, 形態素解析を行い, 要求仕様で使用されている単語を抽出する. 複合語として正しく抽出できていない単語や未知語と判断される単語がある場合には, ドメイン用語と判断すべき単語を形態素解析で使用する辞書に追加し, 形態素解析を再実施する.
2. モデリング
   形態素解析で抽出した単語から, 要求仕様で使用されているドメイン用語を特定する. DW Model の記述ルールに従い, 特定したドメイン用語を木構造で関連付ける. 抽象的な用語に対して下位概念となる用語を抽出し関連付けをするために, 必要に応じて入力としている要求仕様以外の技術文書等も参照する.
3. モデルのレビュー
   モデリング実施者以外で, 用語を解釈するための前提知識をもっている人をレビューアとする. レビューでは, DW Model の記述ルールを満たしているかを確認し, 更に木構造で表現されたモデルの各階層が MECE な状態であるかを確認する. モデルの確認によって, モデリング実施者とモデルのレビューアの間で解釈の相違があった用語を検出する. レビューで, 不足している前提知識が判明した場合, 入力とした要求仕様以外の技術文書等から必要な情報を得て, モデリングを再実施する.

5. 解決策の評価

5.1. 評価方法

考案手法が以下の 4 つの条件を満たすか確認し，考案手法の有効性と導入容易性を評価する.

A)「抽象的に表現されている曖昧な用語」を検出できる
抽象的に表現されていることが要因で誤解釈を誘発する可能性のある曖昧な用語を，モデリング実施者が検出できたかを確認する.

B)「解釈のための前提知識が不足している用語」を検出できる
モデリング実施者の前提知識の不足が要因で誤解釈を誘発する可能性のある用語を，モデリング実施者またはモデルのレビューアが検出できたかを確認する.

C) 要求仕様の形式に依存せず実行できる
複数の異なる形式の要求仕様を入力に DW Modeling を実行し，DW Model が作成できることを確認する.

D) DW Modeling が未経験の状態からでも実行できる
DW Modeling が未経験の状態から，手法の説明を受けるだけで，DW Model が作成できることを確認する. 更に，モデリング中に，手法に関する質問が発生しないことを確認する.

評価のために表 2に示すデータを計測する.

評価のための実験は，以下の条件下で実施している.

- 入力とする要求仕様は，実開発で使用されている文書で定義されているものとする.
- 入力とする要求仕様は，レビューが完了している状態とする.
- モデリング実施者は，DW Modeling 未経験者とする.
- モデリング実施者は，DW Model の記述ルールと DW Modeling の説明を受ける.
- モデリング実施者は，要求仕様に記載されている用語を知っている.
- 形態素解析は，KH Coder[12]を使用して実施する.
- モデリングは，マインドマップ作成が可能なフリーソフトウェアを使用して実施する.

5.2. 評価結果

実験の結果を表 3に示す. 被験者:1～4 は同じ要求仕様を入力とした. 被験者:1 は自分自身が書き手である要求仕様を入力とした. 被験者:1 以外は他者が書き手である要求仕様を入力とした.

実験の結果から，DW Modeling が. 評価方法に示した A)から D)の 4 つの条件を全て満たすことが確認できた. 確認結果を以下に示す.

- 全ての被験者が，「抽象的に表現されている曖昧な用語」と「解釈のための前提知識が不足している用語」のどちらかを検出できた.「解釈のための前提知識が不足している用語」の検出数が少ない場合，つまりモデリング実施者が要求仕様の理解度が高い場合，「抽象的に表現されている曖昧な用語」を検出できる傾向があった. また，一部の誤解釈を誘発する可能性のある用語は，モデルのレビューの前に，モデリングをしながら検出された.
- 形式が異なる複数の要求仕様を入力として，DW Model を作成できた. 要求仕様:A は，派生開発で作成された変更要求仕様書であり，仕様の変化点のみが記述されている. モデルを MECE な状態にするため，DW Model に，入力とした文書にない用語が多く追加された.
- DW Modeling が未経験の状態でも，一通りの説明を受けるだけで，全ての被験者が DW Model を作成できた. モデリング中に，被験者から手法に対する質問もなかった.

また，複数のモデリング実施者から，以下の所感が得られた.

- 変更要求仕様書[4]に定義されている一件の変更要求仕様に対しても実行可能である.
- 知らない用語については，モデルに用語の関連を表現することができない.
- 入力とする文書に同義語が多いと，同義語の判定をする手間がかかり，モデリングの効率が落ちる.

表 2 計測データの種類

データ			説明
用語数	要求仕様		要求仕様に登場する名詞の数. KH Coder で抽出された名詞の数. 出現回数ではない.
	DW Model	要求仕様にあり	DW Model に登場する用語の中で, 要求仕様に存在している用語の数.
		要求仕様になし	DW Model に登場する用語の中で, 要求仕様に存在していない, モデリング実施者が追加した用語の数.
誤解釈の可能性のある用語数			要求仕様から検出した, 誤解釈を誘発する可能性のある用語の数.
	表現の要因		抽象的に表現されている曖昧な用語の数.
	知識の要因		解釈のための前提知識が不足している用語の数.
手法への質問の数			モデリング中に発生した, 手法に対する質問の数.

表 3 実験の結果

被験者 ID	要求仕様 ID	立場	用語数			誤解釈の可能性のある用語数		手法への質問の数
			要求仕様	DW Model		表現の要因	知識の要因	
				要求仕様にあり	要求仕様になし			
1	A	書き手	30	13	38	2	1	0
2	A	読み手	30	33	11	0	4	0
3	A	読み手	30	26	15	1	2	0
4	A	読み手	30	26	10	1	3	0
5	B	読み手	61	35	3	4	1	0
6	C	読み手	23	17	5	2	1	0

5.3. 結果の考察

DW Modeling は, レビューでは見逃した「誤解釈を誘発する用語」を検出できる有効な手法である. モデリング実施者が厳密に解釈できていない用語はモデルに表現することも難しいため, 自分自身で誤解釈の可能性に気づくことができる. モデリングを通して, 「わかったつもり」[13]の状態に気づくことができる. また, 用語の解釈が可視化されることで, 関係者間で解釈の比較することが可能となり, 誤解釈する可能性のある用語を検出できる.

DW Modeling は, 要求仕様を自然言語で記述している開発プロセスに対して導入しやすい手軽な手法である. 要求仕様の形式を変更しなくても実行できる手法であるため, 要求仕様を情報ハブとしたコミュニケーションの仕方を変える必要もない. また, 木構造で用語をつなげるだけのシンプルな記法であり, 覚えるルールが少ないため, DW Modeling が未経験の状態でも, 一通りの説明を受けるだけで実行できる.

DW Modeling の有効性と効率性は, 「モデリング実施者とモデルのレビューアがもっている前提知識」と「入力文書の質」に依存すると考えられる. 本稿の評価では, モデリング実施者とモデルのレビューアがもっている前提知識の違いにより, 「誤解釈を誘発する用語」の検出結果がどのように変化するかは確認できていない. また, 入力文書の質の違いにより, 考案手法の効率がどのように変化するかは確認できていない. 今後, 考案手法により効果を得るための前提条件を明確にすることが必要となる. また, DW Modeling は要求仕様のレビューに加えて追加で実行する手法であり, 導入障壁を下げるために, 更に

実行する工数を小さくする対策も必要となる.

DW Modeling を実行する工数を小さくし, 更に手法を導入しやすくするためには, モデリング実施者の所感から, 以下の 3 つの対策が有効であると考えられる.

- 小さく繰り返す
  DW Modeling は, 要求仕様書の全体ではなく, 一部の記述に対しても, 実行可能である. 開発期間が限られている場合でも, DW Modeling の対象とする記述を絞り込むことで, 現実的に投入可能な工数で実行可能となる. 更に, 作成されたドメイン用語のリストと DW Model は再利用可能であり, 手法を繰り返し実行する度に, 必要となる工数は低減される.
- 書き手が実行する
  DW Modeling は, 要求仕様の書き手が実行しても, 自分自身で定義した要求仕様に対して「誤解釈を誘発する用語」を検出することが可能である. 書き手が自分自身で,「わかったつもり」の状態に気づくことができる. 要求仕様の書き手であれば, 解釈が難しい用語は少ないため, 短時間でモデリングができる. 効率的に仕様記述の改善をするためには, DW Modeling を要求仕様の書き手が実行したほうがよい.
- 入力文書の質を高めたうえで実行する
  要求仕様書内に, 対象とするシステムを定義するためには不要である用語が多く存在する場合に, ドメイン用語を特定する作業の効率は悪くなる. 例えば, 同義語, 誤記が多い文書, 例が示されている箇所が特定しにくい文書, 主語が異なる記述(例:サーバーとクライアント, システムとサブシステム)が混在している文書, 仕様と設計の位置づけの記述が混在している文書などである. DW Modeling をする前に, 入力とする要求仕様の記述をモデリングがしやすいように改善する取り組みをすると, 効率が向上する可能性がある.

6. おわりに

用語の曖昧さや用語間の関連の曖昧さは, 読み手の前提知識をもとにした解釈を必要とし, 要求仕様の誤解釈による手戻りにつながる可能性がある. 実際に, 自然言語で要求仕様を記述しているプロジェクトにおいて, 抽象的に表現されている用語が, 要求仕様の誤解釈を誘発していた.

本研究では, 要求仕様の誤解釈による手戻りを減らすことを目的とし, DW Modeling を考案した. 実験で, DW Modeling は, 以下の要求を満たす手法であることが確認できた.

- レビューで見逃した「誤解釈を誘発する用語」を検出できる
- 要求仕様を自然言語で記述しているプロジェクトに対して導入しやすい

DW Modeling は,「わかったつもり」の状態を抜け出すための有効かつ手軽な手法である. また,「小さく繰り返す」「書き手が実行する」「入力文書の質を高めたうえで実行する」という工夫をすることで, DW Modeling の効率を上げられる可能性が高い.

DW Modeling は, 実開発においてもスムーズに導入でき, 要求仕様のレビューに加えて実行されることで, 要求仕様の誤解釈による手戻りを減らす効果を生み出すことが期待できる.

今後は, DW Modeling の有効性と効率性を更に高めるために, 以下の取り組みを行う.

- 考案手法で効果を得るための前提条件を明らかにする. 具体的には,「モデリング実施者とモデルのレビューアがもっている前提知識」と「入力文書の質」が考案手法の実行結果に与える影響を確認する.
- 考案手法を効率的に実行するためのポイントを明らかにする. 具体的には,「小さく繰り返す」「書き手が実行する」「入力文書の質を高めたうえで実行する」という工夫の有効性を確認する.
- 実開発で考案手法を継続的に活用し, その結果をもとに, 手法を改善する. 具体的には, DW Model の記述ルールを見直す.
- ドメイン用語を漏れなく誤りなく効率的に抽出するために, 自然言語処理技術の活用を検討する. 具体的には, 複合語や未知語を適切に抽出するための技術の活用を検討する.
- 開発全体を効率化するため, 要求仕様を参照する次工程でも DW Model を活用することを検討する. 例えば, DW Model をテスト観点ツリー[14]のベースとして活用することを検討する.
- 仕様記述ルール・ガイドや文書校正ツールなどを活用し, 入力とする文書の質を高め, 考案手法の効率が上がることを確認する. 例えば, USDM[4], SRS 記述ガイド[15]などの活用を検討する.

謝辞

本研究に対して有益なご助言をいただいた ソニー株式会社 栗田太郎氏, 国立情報学研究所 石川冬樹氏, 熊本高等専門学校 荒木啓二郎氏, 南山大学 張漢明氏, 第 35 年度ソフトウェア品質管理研究会研究コース 5 のメンバに感謝の意を表する.

本研究の実験にご協力いただいた 株式会社デンソークリエイトとテックスエンジソリューションズ株式会社のプロジェクトメンバに感謝の意を表する.

参考文献

[1] 厳密な仕様記述WG委員, “厳密な仕様記述入門”, 独立行政法人情報処理推進機構, 2013
[2] 大森洋一, 荒木啓二郎, “自然言語による仕様記述の形式モデルへの変換を利用した品質向上に向けて”, 情報処理学会誌 プログラミング Vol.3 No.5 18-28, 2010
[3] 浅海智晴, “マインドマップではじめるモデリング講座”, 翔泳社, 2008
[4] 清水吉男, “「派生開発」を成功させるプロセス改善の技術と極意”, 技術評論社, 2007
[5] 柏原一雄, 白井正人, 小嶋秀和, 都築功, 新留光治, 村上雅哉, “派生開発におけるテスト漏れを防止する Difference Statement Coverage 分析法の提案”, ソフトウェア品質シンポジウム 2019, 2019
[6] 小林展英, “D-Case を用いたレビューを見える化する方法の導入事例”, 12th Workshop on Critical Software System, 2015
[7] 阿部圭一, “情報伝達型の日本語文章に現れるあいまい表現の類型化とその改善例”, 情報処理学会デジタルプラクティス Vol.5 No.1 70-79, 2014
[8] 加藤潤三, 佐伯元司, 大西淳, 海谷治彦, 山本修一郎, “シソーラスを利用した要求獲得方法（THEOREE）”, 情報処理学会論文誌 No.50 No.12 3001-3017, 2009
[9] 加藤潤三, 佐伯元司, 大西淳, 海谷治彦, 林晋平, 山本修一郎, “要求獲得のためのシソーラス構築支援”, 情報処理学会論文誌 No.57 No.7 1576-1589, 2016
[10] 小川清, 斉藤直希, 吉川直邦, “リアルタイム組み込みソフトウェアの用語の木”, 電子情報通信学会技術研究報告 コンピュータシステム 102(700) 13-18, 2003
[11] “JIS X 0901:1991 シソーラスの構成及びその作成方法”, 日本工業規格, 1991
[12] 樋口耕一, KH Coder 3, http://khc.sourceforge.net/, 2019
[13] 西林克彦, “わかったつもり 読解力がつかない本当の原因”, 光文社新書, 2005
[14] 羽田裕, 青木教之, “テスト視点による上流工程での予防活動と検知活動の成熟度向上”, 組込みソフトウェアシンポジウム 2013, 2013
[15] 不破慎之介, 山田ひかり, 蛸島昭之, “要求記述のスキル不足に対する SRS 記述ガイドの有効性評価”, ソフトウェア・シンポジウム 2018, 2018

「意図」の位置

漆原 憲博
株式会社ジェーエフピー
japanfp@jfp.co.jp

佐々木 千春
株式会社ジェーエフピー
sasaki1000@jfp.co.jp

要旨

納品したシステムが，発注者の最終の目的あるいは端的に「意図」とでもいうべきものを満たしていないがゆえに，受入の承認を受けられない場合がある．自社製品の最終の妥当性確認が得られなかった場合も含む．本論では「意図」の解釈の随意なところがその原因であると考える．また「意図」を，システムに多大な改変を与える大きな要求と考え，まずは本稿を始める．その後，「意図」を解釈する上で基本な考えがあるということと，それを踏まえて，意図が開発の最後に問題を起こさないようにするための意図の扱い、すなわち開発プロセスでの管理法を提案する．挿話的に意図の前向きな効果も入れた．最後に「意図」とシステムの関わりを述べ，意図の研究を開発論の中に取り入れることを提案する．

1. はじめに

システムが，要求文書（要求仕様書）を過不足なく満たしているにもかかわらず，そのシステムが（受入検査において）承認されない場合がある．ここで，要件文書は利害関係者に承認された（権威のある）ものとする．

システムは要求文書を満たしているならば，承認されるのが当然である．しかし，当然ではない事態がまれに起こる．そしてこの「ご無理（な要求）」が土壇場で通ったりもする．この変転の因子は何か．「システムは X を満たしていないならば，X は承認されない」という X が存在するといわざるを得ない．

X の候補は何か．「意図」と思われる．次のような問答は，各位多少覚えがあるかもしれない．

「タブレットの立ち上がりが十分速く，と言った筈なのにこれでは遅すぎる」
「十分速いと思いますが．他社より速いですよ．」
「一番？」
「調べたかぎりでは」
「そう？　でもホントに一番でないとダメだ．他社も全部調べて．」

この会話では，「十分速く」は相互に承認されているようである．が最後は，発注者の発言は，「一番速く」に変わっている．

「十分速くとは，もっとも速く，とほぼ同義なはずだ．もっと速くとは一番速くという意味だ．」

かなり無茶である．そして最後は，「別途請求は無理だよ．」（と，いったとか，いわないとか．）

「意図」の意味は辞書にはこうある[1]．

① 何かをしようと考えること。「 ーした半分もできない」
② こうしようと考えていること。めざしていること。「敵のーを見抜く」

この①と②の例にならえば，発注者からいえば，「意図した肝心の部分ができていない」となり，受注者（以下，「開発者」と呼ぶ）からいえば，「発注者の意図を見抜けなかった」ことになる．

このような開発をめぐる行き違いは海外への委託でも頻繁にある．ちなみに「意図」は「intention」というが，この用語は論理学用語の内包「intension」の原義をなしている[2]．内包は外延「extension」と対をなす．先例においては，内包の抽象的な意味を外延化し，具体的な意味に起こし直せばよかったのであろうか．そこでこう言ってみる．

開発者：「では，何秒だとよかったんですか？」
発注者：「あなた方プロでしょうが．いずれ客先でうちの営業マンがタブレットを開けたとき，なんと思う？　こんなに待たせるなんて，使い物にならんな．」

どうも，「一番」ということもなさそうであるが，いかんせん話がこじれてしまった．ただし，事がおさまって後日聞けば，営業マンに「世界一速いと思える」専用タブレットを持たせ，自信を持って営業をしてもらいたかったそうだ．

[1] 三省堂 大辞林 第三版

[2] https://www.lexico.com/

このような相手に少し寄り添ったやり取りが開発の当初からあったならば，と思うが，後悔先に立たず．人は実物を見ないと実感にはいたらない．そんな人間特性も開発の一部かもしれない．

現実には取りに足らぬ意図もあるわけであるが，他方傾注に値する意図を，この例を参考に定義すると（微妙に異なる事例はたくさんあるとして），意図とは感性的な要求で，かつ開発工程の最終あたりに登場し，納入リスクをはらむ要求といえる．ここで感性的な要求とは，数値等では決めがたいもの，さらにいえば，決めようとすると開発者（受託者）が，発注者に押し切られてしまいそうなものともいえる．逆に押し切るくらいに，発注者は強い動機を内在している．内在とはここでは本人も言葉に出して明確には言えないもの，というぐらいの意味である．

ところで，このような話は社内にもある．むしろ，要求文書を明確にやり取りしない社内の製品の開発にこそ多いかもしれない．本論は，社外であれ，社内であれ，意図により，無駄が発生しないために，あるいは振り回されないようにするためにどうすべきかを論じる．

2. 意図と要求の関係，また登場人物たち

意図が開発サイクルの中で，要求にどう関わってくるのかを確認する．このことは意図を扱う上での基本である．

先の発注者と開発者の関係を模式化するとすると，意図と要求，また発注者と開発者の関係が下図のように描ける．

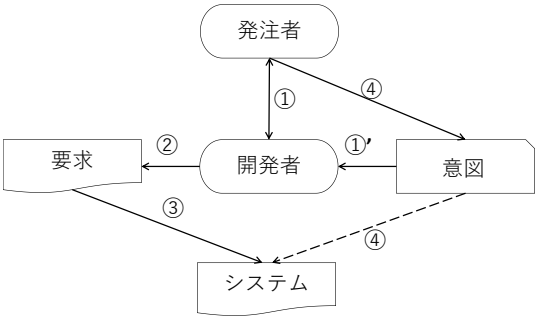


図1　要求と意図の関係

図1では，発注者と開発者が協議し合意の上で（①），要求文書が作成され（②），その要求文書に基づいて，システムが開発されること（③）を物語っている．④では出来上がったシステムの最終的な妥当性の確認をすることを示している．そしてこのとき，妥当性の基準は「意図」であることを，発注者からの矢印④で示している．

またこの意図は，「①‘」でも表現されている．これは①と同じ初期の工程でも，発注者の意図を開発者は「汲むべき」ということを表現している．②は，①で両者が合議したものを開発側で作成していることを表わしている．また，加えて「①‘」で意図を開発者とつなげることで，②での要求文書に開発者は意図も反映させることを示している．

さて，図がこうであるとするならば，④において，妥当性の承認が下されないのは，おかしいということになる．なぜなら，工程①において，また①‘において，意図は要求文書に十分盛り込まれているはずだからである．

事例ではしかし，騒ぎが起っていた．仔細は後に論じるとして，このような騒ぎに巻き込まれるであろう関係者を挙げてみる．後に描く図の各工程に，彼らの多大な工数とストレスが発生することを，ご想像ねがうためである．

開発者は，要求開発者，設計者，実装者，検証者に分かれる．検証者は開発者に含まれるが，他方開発されたシステムの妥当性を確認する者は発注者に含まれる．また，発注を責任をもって行う者がいる．発注責任者である．この発注責任者が通常は妥当性確認の最終的な判断（承認か不承認）をする．

他方，開発者側にも責任者がいる．この開発責任者が，要求文書の開発責任者であり，要求文書の承認，さらに妥当性確認の承認か不承認の通知を受ける人，またその通知に対して同意か否かの諾否を下す人とする．図がこれら登場人物一覧である（図2）．

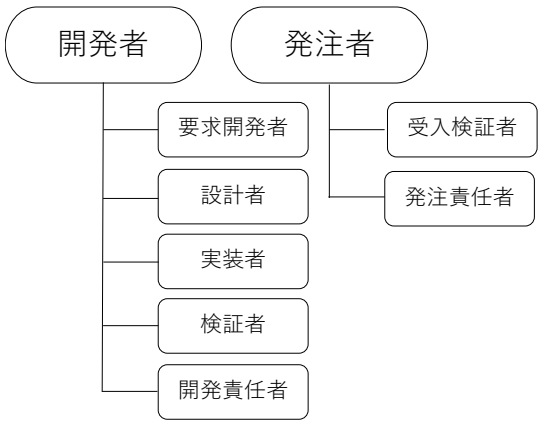


図2　登場人物

3. 課題の精査

図1の通りに開発工程が進めば問題ないはずのものが，問題が起きた．工程の各々を疑ってみる．「あやしさ」や「ずれ」また「反映の程度」などを「不確かさ」と呼び，図

3に表す. 図3では⑤が新たに加わっている.

(1) 発注者と開発者の意図と要求の合意のあやしさ. (①, ①‘).
(2) 合意を開発者は要求文書に完全に反映できているか(②). また開発者は発注者の意図を要求事項として作成(要求化)し, 要求文書に完全に反映できているか(①‘)
(3) 要求文書が完全であったにしても, 開発したシステムにその要求が完全に反映されていない可能性がある. そのとき意図はどこに？要求の中に在る？(③)
(4) 意図が開発の当初と終わりごろでは変わっていなかったか(④).
(5) システムは承認か(⑤).
(6) 不承認の原因は意図か(⑥)

図3では“No”が不合格を示し, 工程を遡るべきことを

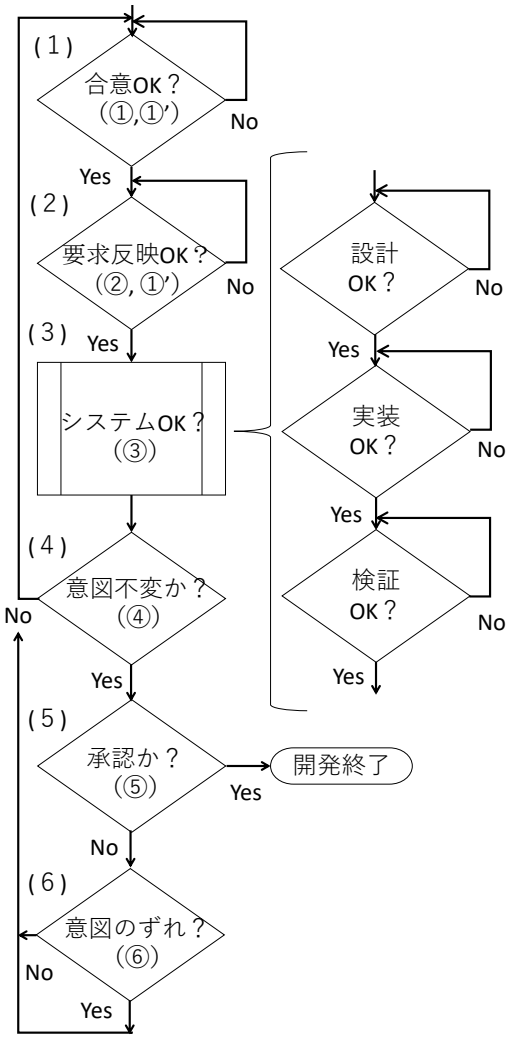


図3 発注者と開発者の不確かさの発生可能箇所

示している. “Yes”になるまで行う.

図3では(5)でシステムの承認の可否が問われ, 不承認であった場合に, (6)で意図がその原因かと問うている. このパスは(4)では意図は最初と変わっていないと言っている. しかし, 最後に突然意図が変わっている.

このようなことがありうるのだろうか. (4)の意図と, 最後の(6)の意図が, 言葉は「意図」で同じだが, 中身は別物という他ない. 何かが現れたのである. これが本論の結論だが, 果たしてそのようなことがあるのだろうか, 一体なにが現れたというのか.

3.1. 「意図」が明らかに関係する工程の点検

工程を遡ると, まず最初の工程の「合意 OK？」の中に

意図に関係する「①‘」がある．「要求反映 OK？」の中にも「①‘」がある．途中で意図が変わっていないか（「意図不変か？」（④））も意図に関係する．よって，意図はこれらの工程で変わっているのはと思わせるが，本論では（4）での意図は最初の意図と変わらなかったとしているので，それに従う．

他方，意図ではなく，要求の変更はこの意図の変化に影響をおよぼさないであろうか．要求が変更され，開発が先頭にまで戻るような場合は，要求の大きな変化である．要求も軽微な変更であれば，意図まで影響しないと経験的にいえるが，大きな変更の場合はどうであろうか．

大きな要求の変更は意図の変更に等しいと考えられる．というのも，意図は要求の上位の概念に属し，意図は要求を含んでいるからである．その小さいはずの要求が大きく変わるというのであれば意図に影響しないはずはない．「より速く」が「一番に」に変わったとすれば，大きな要求の変更である．志向することが変わったからである．

事例では，営業マンへの動機づけは，相対的なもの（「より速く」）から，唯一絶対（「一番に」）に変わった．これは，システムの技術的な変更におさまらない，大きな経済的な価値の変化ではないか．なにしろ「一番」なのだから．

もちろん，発注者は不変と言うかもしれないが，最終のシステムの出来栄えを見て，これなら「一番を」と趣旨変えをしたのかもしれない．意図が（当初の要求の）技術的機能の境界を超えて，約束外のところに向かうところに，意図のおさまりきれないところがある．

ところで，要求に理由を付した要求の記述法がある[3]．次の例がある．事例 B とする（先を A）．

要求：省電力設定をするかしないかを選択できるようにする。

理由：省電力よりもマルチメディアコンポの性能を引き出したい場合があるため。

一般に理由は意図の代替と考えてよい．またこの例Ⓑでは，要求と意図（理由）は意味が近い．他方先の例（Ⓐとおく）では，「より速く」とか「一番速く」という意図は，実現させるとすれば，意味のかけ離れた事項を実現させることによって叶うことになろう（メモリ増強，その他諸々の機能の実現によって）．

[3] 例：https://www.jfp.co.jp/slp/exercise.html 理由の記述は，渡辺滋氏（サイドクロス社）．https://www.side-cross.com/

この例Ⓑのように要求と理由が意味的に近い場合は，変更があったにしても開発プロセスの中で十分吸収可能であろう．

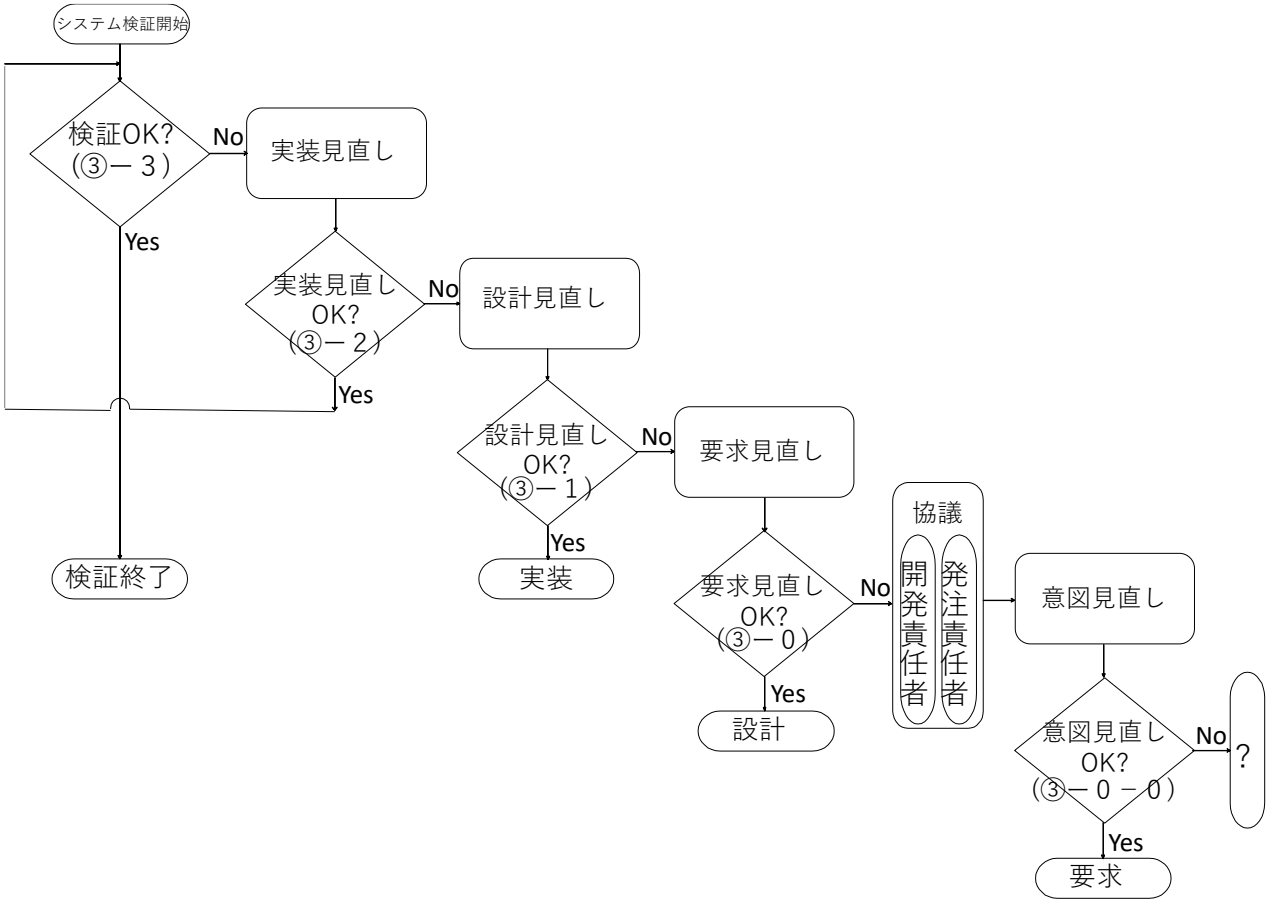


図4　検証工程の前の工程への遡及

3.2. 意図はもう関係しないと思われる工程の確認

先の図3には設計や実装の工程もサブシステムとして書かれている．図3の段階(4)で意図は不変としたので，このサブシステムには言及しなくてもよさそうだが，意図や要求がどう扱われているのか念のため確認しておく．

設計や実装は，要求の指示通り，であるのが原則であるので，これらの工程で問題が起きたなら，検討は要求工程に遡り，そこで吸収されると考えられる．しかし，検証はどうか．

図4では，先の図3の③「システム OK？」の検証工程を拡大し，工程の遡及を描いている．

単体テストやその上の結合テストをするあたりでは(③－2～③－1)，実装から出発し，設計の見直しあたりまで遡及する場合もあろう．設計を再検討する際には，要求が問われ，その場合には先の(例Ⓑでのような)理由も問われるであろうが，要求を大きく変更する「意図」問題は出てこない．

そして，設計，実装がいずれフィックスし，総合テストに行き，多少の小さな課題は出たにしても，最終的には妥当性の確認で OK がでる．これで本来なら開発は完了である．発注者も納得である．

しかし，図4では要求の見直しにまで遡り(③－0)，発注と開発の両責任者が協議することとなっている．そしてさらにこの図4は，発注者がそもそもこのシステムは，意図を満足していないのではないかと「意図」を持ち出している(③－0－0)．

しかし，そもそも検証とはどういうものであったか．検証の基本は，要求文書に基づいて検証すべき事項を検証仕様として書き，それに基づいて検査を行うことである．

そして，「意図」がここで万が一登場するとしても，意図

は抽象的(内包的)なものであるため，検証者は意図から具体的な検証仕様をつくることはできない．「より速く」とか，「十分速く」といわれても，定量的な数字の提示は(通常な)検証者の責務の範囲外である．

4. 新たな方策

結論からいえば，意図の内包の危うさ，また随意さに対する対応策は，検証工程で定量化が求められたように，内包の外延化である．「十分速く」と「一番速く」は，内包的意味は明確に異なる．一番速ければ，十分速いといえようが，十分は速くとも，一番速いとはいえない．しかし，本論の冒頭で発注者は，十分速いという意味は，一番速いことを求めていることだと言う．しかも後には「一番」は「世界一」に変わった．無理を言うが「内包」の融通無碍さである．

このように発注者には内包は自由であるが，それゆえ開発者にはきつい話である，しかし，両者の均衡を取るには，月並みであるが，内包の意味を具体的に合意するしかなかろう．開発者は，内包の融通無碍(発注者には)を，具体性すなわち外延で縛ると強く肝に銘ずべきである．

例えば，「いまどこどこの会社(A 社)のタブレットの起動がもっとも速いのでそれを超えること，しかし，どの会社もしのぎを削っているので，現時点では A 社を超えることを，当該意図の内容とする」．

このように発注者と開発者とは，具体的に取り決めをすべきである．このことは社内においても，である．

しかし，内包には先の事例で述べたように，同じもののはずが(「意図」)が変わる．そして．変わった意図の説明が，今までの文脈とは異なる対価などという経済的理由であったりする．意図が要求の上位概念などという場合，論理的な意味ではなく，世俗的な優先度だったりもする．

技術論としては興味の薄れる点かもしれないが，多くの現実のシステムは，技術以外の因子，多くは経済的合理性の中で動いている．従って，意図や内包，すなわち”intention”や”intension”の融通無碍さを開発プロジェクト論として論じてもいいのかもしれない．これらがプロジェクトに様々に影響するから，である．

ときに現場の技術者はこのような経済合理性の議論に無力で，結果的に不合理とも思える時間を強いられる．システム開発が決してスマートなものでないことは世に流布して久しいが，意図や内包の概念を，プロジェクト論としてうまく論ずるものは皆無かもしれない．ときに無理を強いる意図を事前に見抜き，開発の効率を制御したい．

4.1. 入念だが，実の乏しい合意形成かも

表1では，意図を制御できないかと考えている．表1では①‘として意図を要求とは別に取り上げている．

表1　開発プロセスに応じた当事者の合意形成

工程	①				①‘				②					③			④			
	要求依頼				意図依頼				要求承認					開発			納入検査			
発注者	依頼		協議	合意	依頼		協議	合意			受け	協議	合意				納品物検査	納品物判定		協議
	↓	○	↓	↓	↓	○	↓	?		○	↓	↓	↓			○	□	↓	○	↓
資料	要求素案			要求原案	意図素案			意図原案		要求文書			正式要求文書			システム		納品結果書		
開発者	○	↑	↑	↑	○	↑	↑	?	□	↑	○	↑	↑	□	□	↑		○	↑	↑
		受け	協議	合意		受け	協議	合意	要求文書作成	提出		協議	合意	システム開発	完了	搬入			同意 or 不同意	協議

しかし，先の図3では意図は何も変わっていないと言いながら(発注者も開発者も)，なお最後に発注者は出来上がったシステムは「意図」とは違うと言った．

表1では意図は①‘の工程で明示的に扱われている．意図原案となっている．そして，その意図原案は，次の工程で要求文書に組み込まれているはずである．

発注者と開発者は「依頼」とか「協議」をし，矢印↓は資料等を提示する方向を示し，○は受けとめ，？は不確かだが，本来確かにしなければならないことを示し，また四角印(□)は自陣での作業を示している．

表1は入念な作業の様子を描き，要求①と意図①‘の工程の後は，工程は要求承認②，開発③，納入検査④と進むことを示している．

表1の作業により，先の図3の(4)では意図は当初(＝原案)と変わらないと言ったわけだが，しかし，そう言ったにもかかわらず，図3の(6)ではシステムは意図とは違うと発注者は言ったのであった．

何が変わったか？システムは変わっていないのは事実なので，変わったのは発注者の気持ちである．すなわち，寸前まで(段階(4))変わっていなかった意図が変わったのである．

入念に作業したはずの表1の中に「？」があった．「？」は「不確かだが，本来確かにしなければならない」はずのものであるが，確実化していなかったと疑われる．

しかし，意図を確かにするとは，そもそもどういうことで

あろうか. それを明確にしなくては, 担当者に問いようがない.

そこがなくては, 表1は見かけだけの, 合意形成フローに過ぎないかもしれない.

4.2. 「意図」に他の視点を入れる

すでに見た通り, 意図は最後に突然意趣変えをするなど「我が儘」になるが, これを少し黙らせる方法はないであろうか. 要求は意図に近いので, 要求文書の品質を向上させる国際規格は役にたたないだろうか. 国際規格 IEEE Std.830 に関する研究資料に考えを借りる[4]. 同規格には, 要求文書のための8つの品質特性があるとされる(表2参照).

表2では, 表1の要求, 意図から開発へと行くプロセスに, これらの要求品質特性を当てはめ, 表1の文書がどのような品質特性に気をつければならないかを確認している. その際, 品質特性を分類し直し, 対象とする文書にだけに当てはまる品質特性と, 当該文書以外にも当てはまる品質特性とに分けた. Traceable(追跡可能)は他の文書と関係する品質特性である.

また文書内だけのものは, さらに「要求自体が正しいか否か」を問うものと, 「要求を第三者的に見る」ものとに分けた[5]. 要求の正しさには, 要求は Correct(妥当), Unambiguous(非曖昧),Complete(完全),Consistent(無矛盾)が求められ, また第三者的視点では, Ranked for importance and/or stability(重要度/安定度のランク付け),Verifiable(検証可能),Modifiable(変更可能)が求められる.

他の文書と関係する「追跡可能」性は, 要求事項の追跡関係ばかりではなく, 記載履歴, メモのようなものも含んだものである(脚注, 4の文書「要求定義で, , , 」より).

よって, 意図に戻れば, 意図に関する事柄は, 記載履歴, メモなど追跡的に管理すればよいと思われる. これを表2では, 「◎」の記号で表現している.

先の表1では, 「意図原案」, 「合意」まではいいとしても, 実際に合意形成をしたか否かが「？」となっている.

そこで, ここは国際規格 IEEE Std.830 を援用し, 意図に関しても文書化を義務とし, 意図が納入物になるとしてもよいであろう. 表ではそれも「◎」で表現している.

また, 表2では, STAMP/STPA の考え方を取り入れている. すでに見た通り, 意図は当初の合意にもかかわらず, 最後に当初とは反することになりかねないからである. 制止をきかず, 制御の輪を潜り抜け, 一種のハザードたりうる[6].

システム開発全体も制御的な業務フローにもたとえることができる[7]. 任意の作業をよく確認しないで次に進むとハザードの原因となることがある. 意図はそのなかで, つい「わかった」ぐらいの確認で前に進みがちである.

また表2では STAMP を模し, 「与えられた場合のハザード」を「文書が与えられた場合」としている. 対比的に「文書が与えられない場合のハザード」も存在するわけである. この「与えられない」場合は, 要求仕様書が無くて現場が四苦八苦する際のハザードを思い浮かべるだけでよい. 今回は議論を割愛した(斜線).

表2では, 「与えられた場合のハザード」に関して, 文書の品質特性ごとに「・」で示し, 登場人物が品質特性の確認をなすべきことを示している. STAMP では, さらに「早すぎ, 遅すぎ, 誤順序」, また「早すぎる停止, 長すぎる適用」の視点がある. この時間や順序性の視点を表2に適用すれば, 任意の品質特性の検討の遅れや検討順序のミスが他の品質特性に悪影響を及ぼすなどはすぐに理解できる. ただし, 表が複雑になるのでこれらの視点は割愛した.

以上のように要求の品質特性や, 業務制御の考え方を適用することで, 開発業務がより周到に管理されるであろうことが理解できる.

[4] ただし本稿では, 「要求定義で困ってませんか？要求仕様書の品質に関する研究成果報告」(要求工学ワーキンググループ 2007 年 1 月 24 日)より引用.

[5] 特性の数8つは多いので, よりシンプルなクルーピングを試みた.

[6] 「はじめての STAMP/STPA」等. IPA サイト参照.

[7] 事務業務に対する STAMP/STPA の適用は, システム開発に対する適用の可能性を示唆する.
「STAMP/STPA～業務系システムへの応用検討～」, (IPA/SEC「システム安全性・信頼性分析手法 WG」委員, 日本電気株式会社, 向山輝)
chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.ipa.go.jp/files/000056599.pdf

5. おわりに

本論では，冒頭で厄介者の意図を取り上げ，通常の開発プロセスをたどり，開発に関わる登場人物たちを思い浮かべながら，意図の様相を探った．そこでは意図は時折顔をのぞかせ，意図は変わっていないと言ってみたり，最後にはシステムは意図とは違っていると言ったりした．意図は，対価にも言及し，払う，払わないと，機能以外の話を持ち出したりする．

そこで要求の品質特性や，業務制御の考え方を適用し，開発プロセスをより周到に回すことを考えた．しかし，このようなことで，意図は本当に矛をおさめてくれるであろうか．

答えはノーである．理由は，意図は要求の上位に位置する概念だからである．あるいは，意図を要求では御しきれないからである．

出来上がったシステムを見て，くだんの人は違うと否定したが，しかし他方，もし万が一意図した以上のシステムを作ったとしたならば，中には誰かはが素晴らしいと賞賛するであろう．

前もって持っていた内包概念（intension）の中に，これあれというような具体的なイメージがあったとして，その具体的なイメージにそぐわないときは否定で，具体的なイメージをまったく超えた，しかしそれが同じ内包概念の範疇のものであることを気づかされたとき，それが求めていたもの，賞賛へとつながる．

この賞賛とは逆に，くだんの人が，意図が同じと言ったそのあとに，実際のシステムを見て意図と違うと言ったのは，好意に理解すれば，イメージにそぐわなかったに過ぎない．氏は何ら矛盾してない．

意図が要求の上位概念という意味は，このように自由な裁量権を要求以上に多く持っているからといえる．また意図は意味を膨らませることができる．あるいは，要求を増やすことができる．意図が内包だとすれば，要求は具体

表2　要求文書のリスク管理と「意図」の位置づけ

工程	順番	登場人物	制御行為	文書が与えられない場合のハザード	文書が与えられた場合のハザード							
					要求を当該文書内だけで扱う特性							要求を当該文書外(成果物,関連文書)と関係づける特性
					要求自体が正しいか否かの特性				要求を第三者的に見る特性			
					Correct	Un-ambiguos	Complete	Consistent	Ranked for importance and/or stability	Verifiable	Modifiable	Traceable
					妥当性	非曖昧	完全	無矛盾	重要度/安定度のランク付け	検証可能	変更可能	追跡可能
①	1	発注者	発注依頼		・	・	・		・			
		開発者	依頼受け		・	・	・	・		・	・	・
	2	両者	協議検討		・	・	・	・	・	・	・	・
			要求概要作成		・	・	・	・	・	・	・	・
	3		合意		・	・	・	・	・	・	・	・
①‘	1	発注者	意図提示		・	・	・		・			
		開発者	意図検討		・	・	・	・		・	・	・
	2	両者	協議検討		・	・	・	・	・	・	・	・
			意図文書成		・	・	・	・	・	・	・	◎
	3		合意		・	・	・	・	・	・	・	・
②	1	開発者	要求文書作成		・	・	・	・	・	・	・	・
	2	開発者	要求文書提出									
	3	発注者	要求文書受け									
	4	両者	協議検討		・	・	・	・	・	・	・	・
	5		要求文書合意		・	・	・	・	・	・	・	・
③	1	開発者	システム開発		・	・	・	・	・	・	・	・
	2		開発完了									
	3		搬入		・	・	・	・	・	・	・	・
④	1	発注者	納入物検査		・	・	・	・	・	・	・	・
	2	発注者	納品物判定		・	・	・	・	・	・	・	◎
	3	両者	同意or不同意		・	・	・	・	・	・	・	◎

的な外延といえる. 内包と外延には上下関係があるわけではないが, 内包は定性的に物事を把握するゆえ, 少ない言葉でより多くの領域(世界)に言及できる. 本論の文脈で, 意図を上位とする理由である.

繰り返すが, この自由な上位の概念を制御する方法は, 対としての外延で合意を取り付ける他はなかろう. 内包(意図)が大きく膨らんでもそれは自由であり, しかし契約は外延的な具体的なもので行うことしか, 内包を納得させることはできない. 表1の意図原案の合意が「?」であったのは, 意図を外延化することが合意であることを知らなかったからである. よって, 問うことも問われることもなく, したがって, どうしたらよいかわからなかったのである.

くだんの氏の,「より速く」とか,「世界一」とか, この定性的な用語も内包の用語である. この内包的用語のどれを選ぶか, それこそ意図を問わないといけない.「営業マンが云々」とあるが, どうもこの辺に真意がありそうだ. さてそれで, 外延化は? 「顧客が大きな, 身の丈に合わない意図を持っている」などと否定的なことを言ってはならない. 真剣に耳をそばだて, 寄り添い, そうしなければ, 意図の本当の声は聞こえない. そして, 外延化できない.

こうすれば, 先に「入念だが, 実の乏しい合意形成かも」と, その効果を疑った開発プロセスも生きてくる. 意図を制御し, 開発プロジェクトが魅力を取り戻す.

以上, 意図について試論した.「意図したシステム」などというように, 意図はシステム全体をくくる.「要求したシステム」というより, そのくくりが強い. システムをくくる「意図」は, システムの発注者の意思と強く関係する. スポンサーであったりする. あるいは, ああでもない, こうでもないと試行錯誤するアーティストかもしれない. ゲームやプロジェクションマッピングなどヴィジュアル系の世界は, 要求文により作られる世界ではなく, 表現者の意思, 意図で作られる. そのような世界というべきであろう. そして, アートではない自動運転なども, その走行の評価・調整は, 感覚的なものであり, 最後はなんらかの「意図」で決まる. もしかしたら, 決めた本人たちもその実, なぜそうなのか説明ができないかもしれない. もちろん, 安全規格はクリアしてるが, 例えばデジタル的な処理が幾重にも重なった結果生まれた乗り心地感. この決定は「意図」ではあるまいか.

システムの周辺には「意図」がひしめき, システムと相互作用をしている. システムと意図の関係をひもとき, 納得感を得たいものである.

参考文献

[1] 『人月の神話』, フレデリック・ブルックス, 1975, 20 周年記念版;1995

「意図」は氏の説く, システムのコンセプトの完全性と通じているかもしれない. つまり,「意図」はシステム全体の整合性を無意識のうちに求めているというような擬人的特性を持つ.

本論では本書の引用はしていないが, システム開発者の意思,「意図」を本書は随所に感じさせ, 本論のヒントになったので, 参考文献とした.

[2] *Integration of Two Kinds of Syntax for Requirements Description and Its Future Development*, Norihiro Urushibara ; Chiharu Sasaki, 2018, IEEE This

当該論文では, システムのための要求記述が整合性を求め, かつこの要求は, 設計, 実装の各工程の成果物に概念の「連続性」とでもいうべきものを求めると主張する.

この求めは, システムという人工物を作る開発者たちの「意図」の居場所を求めていることではないか, と本論を書きつつ, ここでも意図の擬人化を連想した. 当該論文は本論では引用していないが, 本論執筆の誘導をしてくれたので, 参考文献とする.

機能共鳴分析法と行動分析を併用したユーザーストーリー分析

日下部 茂
長崎県立大学
kusakabe@sun.ac.jp

要旨

ユーザーの視点を重視したソフトウェア開発の上流工程でのシナリオ分析に機能共鳴法と行動分析を併用することを提案する. この提案の背景には, 大学学部生を対象にモバイルアプリケーション開発の演習指導を行った際の経験がある. 作業成果物に以下の様な問題を持つものが多かった. シナリオの不明瞭さやシナリオ中の齟齬, やり取りの不明確さといった問題があった. 演習ではユーザーストーリーマッピングなど行ったが, それでも不十分であったため, 行動分析と機能共鳴分析法を併用することを試みた. ユーザーストーリー分析において, ストーリー中の構成要素とその間の流れをより明確化することでシナリオの完成度を高め, また, シナリオで想定されているユーザーの行動が生起する根拠の分析を強化することを目指した. 大学の学部生による実験でその効果を評価した.

1. はじめに

大学の学部生向けのモバイルアプリケーションの設計や開発の演習指導を行った際, 上流工程の作業成果物の多くに類似する問題があった. 演習では人間中心設計の手法の一部をとり入れ, ユーザーストーリーマッピングを作成し, ユーザー視点を考慮の上実装戦略を考える試みなどを行っている. しかしながら, シナリオの不明瞭さ, シナリオの流れにおける齟齬や, やり取りの不明確さといった問題があった. 演習ではユーザーストーリーマッピングなど行ったが, それでも不十分であったため, 行動分析と機能共鳴分析法を併用することを試みた. ユーザーストーリー分析において, モデルの記法を用い構成要素と流れをより明確化することでシナリオの完成度を高め, また, シナリオ内で想定されているユーザーの行動が生起する根拠の分析を強化することを目指す.

モバイルアプリケーションの設計には人間中心設計(Human Centered Design 以降 HCD と略記)[1]が有用と考え筆者が行っている演習でも HCD を部分的にとり入れている. HCD はシステムの使い方に焦点をあて, 人間工学やユーザビリティの知識と技術を適用し, インタラクティブなシステムをより使いやすくすることを目的とする, システムの設計と開発へのアプローチである. そのアプローチを推進している人間中心設計推進機構では, HCD はより有効で使いやすい, 満足度の高い製品やサービスを提供するための一連の活動プロセスと定義されている. 本研究では, このような HCD のユーザー要求事項の明確化において, ユーザーストーリーに沿ったシナリオ分析を行う際に, 機能共鳴分析法 (Functional Resonance Analysis Methods, 以降 FRAM と呼ぶ) の分析手法[2]と, 行動分析[3]の ABC 分析との 2 つをとり入れる試みを行った.

機能共鳴分析法 FRAM を用いることにより, ユーザーストーリーがその起点から終点まで, 適切な抽象レベルでユーザーの行動とシステムの連携がつながっていること, またそれがどのように成功となるかの確認が行いやすくなると考えた. 後述するように FRAM では, 機能を中心に対象をモデル化する. 各機能は, 入力(I), 前提条件(P), 時間(T), 資源(R), 制御(C), 出力(O)という六つの側面を持ち得る. このような側面でつながりを記述し, 機能間の相互作用をモデル化し分析する. ユーザーの行動をユーザー側の機能とし, システムの動作をシステムの機能として, ユーザーストーリーのシナリオが完結するか, それらの間の相互作用がインターフェースを介して適切につながっているかといったことを確認する.

また, ユーザーストーリーに沿ったシナリオの分析に, 行動(Behavior)そのものだけでなく, 行動のきっかけを与える先行刺激(Antecedent stimulus)と行動した結果与えられる後続刺激(Consequent stimulus)も分析する行動分析の ABC 分析をとり入れる. ユーザーストーリーの記述や分析では, 誰が, 何を, なぜ, といった点が重要となる. ペルソナを設定した ABC 分析により, ユーザー行動の先行刺激や後続刺激がシステム中に存在し, 対象行動が実際に引き起こされるかについて, 行動分析の知見に基づく根拠を持った記述や分析を行うことで効果的なユーザーストーリーの記述や分析ができると考えた.

このような二つの分析を追加した効果を次のように評価した[4]. 情報系の大学学部男子学生 2 名に, こちらが準備した課題に従ってペルソナ・シナリオ法を実施してもらった後, ABC 分析とFRAM による分析を追加的に実施してもらった. ペルソナ・シナリオ法単体で実施した場合と, ABC 分析と FRAM とを追加した場合を比較した結果, 後者の場合, 要求事項およびインターフェース画面のほとんどで項目数が増加し, より気づきを増やす効果があったと考える.

本稿の構成は次の通りである. 2 章で人間中心設計とシナリオ分析について説明する. 3 章で FRAM とそのシナリオ分析への適用を, 4 章で行動分析とそのシナリオ分析への適用について説明する. 5 章で評価の方法と結果について述べ, 最後にまとめと今後の課題を述べる.

2. 人間中心設計 HCD とシナリオ分析

2.1. 人間中心設計 HCD

Web サービスやアプリケーションには, 先進的な技術だけでなく, 優れたユーザー体験 (UX) が欠かせなくなっている. 新しいサービスやアプリケーションの企画段階からユーザーインターフェース(UI)デザインも含め, そもそもそれがユーザーにとってどういった価値があるのか, そしてユーザーのどういった行動に焦点を合わせるのかといった UX の観点が必要であり, そのような観点をプロジェクトに取り入れるためのプロセスとして HCD が提案されている[1]. HCD は ICT 技術が現在のように普及するより前に, UI を改善するユーザビリティ(使い勝手)の分野から生み出されたものであるが, Web サービスやモバイルアプリケーションの開発にも有用であり, 筆者の担当講義演習でも参考にしている.

HCD は主に次の四つの活動からなるプロセスであり, 図 1のように繰り返し行われることが想定されている.

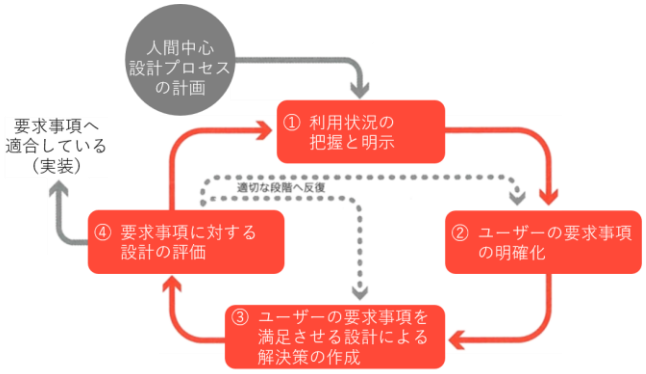


図 1 HCD サイクル[1]

① 利用状況の把握と明示
② ユーザーの要求事項の明確化
③ ユーザーの要求事項を満足させる設計による解決策の作成
④ 要求事項に対する設計の評価

2.2. シナリオ分析

前述の四つの活動で用いる主な手法・サブプロセスとして表 1 のようなものがあるとされている.

筆者の講義演習では時間の制限もあり, 表 1 に示された活動分類およびその対応手法やサブプロセスをすべてその通りには実施していない.

表 1 HCD サイクルに対応する手法[1]

HCDサイクルの４つの活動	主な手法・サブプロセス	
① 利用状況の把握と明示	● 利用状況の把握 ● 利用状況の調査 ● アンケート ● フィールドワーク	● エスノグラフィ ● ダイアリー法 ● インタビュー
② ユーザーの要求事項 の明確化	● グランデッドセオリー法 ● ペルソナ ● シナリオ法 ● 品質機能展開	
③ ユーザーの要求事項を満足させる 設計による解決策の作成	● 発想法 ● パターンランゲージ ● 共感的デザイン	● 参加型デザイン ● プロトタイピング
④ 要求事項に対する 設計の評価	● ユーザビリティテスト ● インスペクション法 ● 心理的尺度	● 生理学的手法 ● 長期的な評価

このうち, 本稿では, 活動②の「ユーザーの要求事項の明確化」にあるシナリオ法に焦点を当てた議論を行う. 特に, ユーザーストーリーマッピングの際に, FRAM での分析とABC 分析を用いた行動分析を取り入れることについて論じる.

3. FRAM

3.1. FRAM の概要

FRAM は複数の組織や人, 機器といった構成要素が相互作用を行うシステムを対象にした, レジリエンスエンジニアリングの手法として提案されている. 従来の安全性と異なる新しい安全性や, 回復・復元力について広義のエンジニアリングを行う. 従来のように, 安全性を「物事が悪い方向へ向かわない状態」としてアプローチするものは Safety-I, 「物事が正しい方向へと向かうことを保証する」と考えてアプローチするものを Safety-II として区別し, Safety-II の考えに基づくモデリング手法の一つとして提唱されている. 以下の四つの原則に基づいて, ものごと

が悪い方向に向かうことより，正しい方向に向かうことに焦点を当てて分析を行う．

① 成功と失敗の等価性の原則，
② 近似的な調整の原則，
③ 創発の原則，
④ 機能共鳴の原則

FRAM では，機能について以下に示す六つの側面によってモデル化を行う．

- 入力(I)：機能が処理の対象とし，変化するもの．あるいは機能を開始するもの．
- 前提条件(P)：能が実行される前に存在・成立すべき条件．
- 時間(T)：機能に影響を与える時間的制約(開始時刻，終了時刻，継続時間など)
- 資源(R)：機能が実行されるときに必要とされるもの，あるいは出力を提供するために消費されるもの．
- 制御(C)：機能がどのようにモニターされ，あるいは制御されるか．
- 出力(O)：機能の結果であり，何らかの実体か状態変化．

FRAM の図式表現では，図 2 のように，各機能を六角形で示す．図式表現でのモデリングでは，典型的には FMV (FRAM Model Visualizer) のようなツールを用いてモデルを記述し，視覚化する[5]．

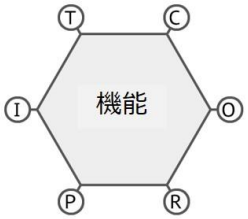


図 2 機能の図式表現

3.2. FRAM の分析

FRAM の分析には，複数の機能間の相互作用に着目した，システムの特性の分析が含まれる．機能の共鳴の観点から，複数の機能が相互に作用した結果，外乱に柔軟に対応して回復する場合もある一方，逆に変動の影響が増幅し，安全を脅かす場合もあるといった分析も行う．

FRAM では機能に着目したモデリングを行い，機能について，その目的，行っている処理，存在する入出力といったものを明らかにし，機能に名前を付け定義する．定義を進めるにあたり，各側面の要素と名前，相手の機能の名前も明らかにする．しかしながら，モデルの構築や分析の具体的な方法に唯一絶対のものはない．例えばモデルのテキスト表現を重要視したものもある一方[2]，図式表現でのネットワークトポロジーに着目した分析法もある[6]．

文献[6]では以下のような観点でのモデル化と分析が紹介されている．

- 機能が受け取るもの
- 条件が変わった場合の適応
- 正常ではない条件への反応
- 正常ではない条件の発生頻度
- リソースが安定的に供給されるか，不安定要因は何か
- 外部要因はどのくらい安定するか，不安定要因は何か
- 前提条件の取りこぼしはないか
- 時間制約にかかるプレッシャーの場所
- 特別なスキル，機能，高信頼性の必要箇所
- 最適な実行方法はあるか

上記のような観点をふまえて機能の特徴を十分に引き出した後，シナリオの成功要因やリスク要因の分析を行う．今回は，ユーザー行動の可変性や調整を前提に，想定する状況下での資源や制約を明確にし，行動分析の手法を併用してユーザーの行動の結果をより予測しやすくする．これにより，期待しているユーザー行動が本当に生じるかどうかの分析を促進しようとするものである．

3.3. ユーザーストーリーへの FRAM の導入

ユーザーストーリーの記述や分析ではステークホルダの共通理解が重要とされている[7]，FRAM の図式表現がそのような共通理解に有効ではないかと考えた．ユーザーストーリーの記述には，特定タイプのユーザーがやりたいと思うことを表す短い動詞句を書くカードを用いたりするが，これは FRAM のモデリングで短い動詞句を機能として扱うことに対応づけ可能と考える．ストーリーの起点や終点，その間のナラティブフローのつながりといったものも FRAM のモデル記述で明確化できると考えた．

このような FRAM の活用は，ユーザーストーリーを使ったアジャイルサイクル(図 3)での確認や構築のフェーズで有効と考える．HCD のサイクル(図 1)の観点からは，このような FRAM を活用したユーザーストーリーの記述や分析は，②ユーザー要求の明確化を中心としたその前後の活動で有効と考える．

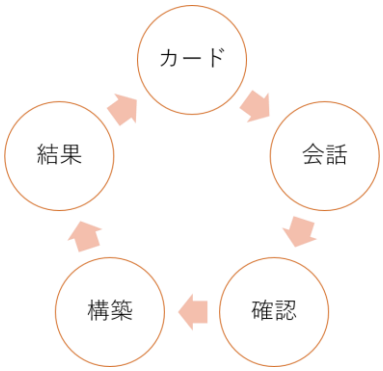


図 3 カードを起点としたユーザーストーリーマッピングを用いたアジャイル開発サイクル[7]

4. 行動分析

前述のような FRAM によるモデリングで，ユーザーストーリーの起点や終点，その間のナラティブフローのつながりといったものの明確化を目指した．ユーザーストーリーでは，ユーザーが必要とすることである成果は何であるか，ユーザーがなぜその製品を使うのかといった観点の分析も重要とされている．ペルソナを設定し，ユーザーの苦痛や喜び，報酬といったものも考慮する．そのような分析を効果的に行うために，行動分析の知見の活用が有効と考えた．ここでは，想定されているユーザーの行動が生起する根拠の分析について，行動分析の手法を取り入れることを論じる．

4.1. 随伴性

有機体の行動には，刺激によって反射的に誘発されるレスポンデント行動と，そのような誘発刺激によらずに自発されるオペラント行動がある．このような行動のそれぞれのタイプに対し，その変容の二つの原型として，不随意的な行動の変容過程(条件反射) であるレスポンデント条件付けと，随意的な行動の変容過程であるオペラント条件づけがある[3].

今回の取り組みはユーザーの随意的なオペラント行動を対象とする．オペラント条件付けは，有機体が環境などに働きかけるために能動的に自発する行動であるオペラント行動の条件付けであり，あるオペラント行動に刺激を随伴させるとそのオペラント行動の頻度が増加するとき，その行動は強化されたといい，そのような操作をオペラント強化という．そのような操作でオペラント行動の頻度が増加することをオペラント条件付けという．オペラント条件付けがなされたオペラント行動は，自発されたときに強化されないと頻度が減少する．このような操作と事実をオペラント消去と呼ぶ．ユーザーストーリーで，ユーザーの苦痛や喜び，報酬といったものも考慮する際に，行動が強化されるか消去されるかといった観点を導入する．

提示されることで，あるオペラント行動の自発が強化される強化刺激を正の強化刺激や好子と呼ぶ．除去されることであるオペラント行動の自発が強化される刺激を負の強化刺激や嫌子と呼ぶ．好子の出現や嫌子の消失は直前のオペラント行動の自発頻度を増大させる．このような随伴性の関係を表 2に示した．

表 2 随伴性の基本パターン

	好子	嫌子
出現	好子の出現による強化	嫌子の出現による弱化
消失	好子の消失による弱化	嫌子の消失による強化

ユーザーストーリーでは，単に作ろうとしているものについて話すのではなく，誰のため，なぜなのかといったことににについて話すのが重要とされている．ペルソナを設定した上で行動分析の観点で随伴性を考慮することでより効果的なストーリーの記述と分析を行うことを目指す．

上述のような随意的なオペラント行動の仕組みに着目して分析を行うにあたって，行動分析学の三項随伴性の分析を用いる．三項随伴性は，きっかけとなる先行条件・弁別刺激(Antecedent stimulus)の下に生じたオペラント行動(Behavior)の後に引き続く強化刺激(Consequent stimulus)が随伴すると，その行動が増えるというという関係を示したものである(図 4参照)．先行条件・弁別刺激，行動，強化刺激の頭文字をとってこの分析を ABC 分析と呼ぶ．

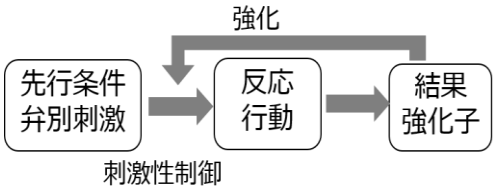


図 4 三項随伴性

行動分析では前述のような行動の随伴性を考えるときは慣習的に随伴性ダイアグラムと呼ばれるものを作成し，

行動の直前直後を記述して分析する. ダイアグラム記述により行動の随伴性による影響を整理し, 行動の強化・弱化について言葉だけで記述するより分かりやすく捉えることが可能とされている. しかしながら, 記法の統一された定義やツールなどはない[8][9]. ここでは, 随伴性の分析を, ツール支援の下, 属人性を排した統一的な記法で行うことでエンジニアリングの観点で効果的に実施することを目指し, FRAM のモデリングと関連付けた随伴性分析を行う. ユーザーの行動に加え, 弁別刺激や好子や嫌子の機能と関連づけて FRAM のモデリングを行う. 機能間のつながりを, 三項随伴性の刺激と反応の連鎖の観点で, 行動分析学の知見と照らし合わせた分析を行う. 成功と失敗の等価性の原則など, FRAM の四つの原則を用い, ユーザーストーリーがうまくいく場合だけでなく, いわゆるリスクストーリーの分析なども行う.

5. 評価

この取り組みでは FRAM によるモデリングで, ユーザーストーリーの起点や終点, その間のナラティブフローのつながりといったものの明確化を目指した. さらにユーザーストーリーにおいて, ユーザーが必要とする成果は何であるか, ユーザーがなぜその製品を使うのかといった観点の分析を効果的に行うために, FRAM のモデリングと ABC 分析を組み合わせることにより行動分析の知見の活用を試みた.

上述の二つの分析を追加した方法の効果を次のような実験で評価した[4]. 情報系の大学学部男子学生 2 名に, 準備した課題に従い後述のペルソナ・シナリオ法[10]を実施してもらった. 準備した課題の概要は, 通学や帰省に公共の交通機関を利用している男子大学生を想定し,「電車・バスの乗り過ごしや乗り遅れを改善すること」を支援するツールを開発するというものである. 次にペルソナ・シナリオ法で作成したサービスシナリオに対し FRAM でのモデル記述と ABC 分析を行ってもらった. おおよそ, 図 3の構築フェーズの準備を含めた確認フェーズに相当し, ユーザーインターフェース画面の概要検討も含んでいる. 評価では再記述と分析により気付くことが出来た必要な画面, ユーザーの行動, システムの機能を画面フローとその画面ごとのユーザー要求に追記してもらいその差分をとった.

5.1. ペルソナ・シナリオ法

以下のようなペルソナ・シナリオ法を用いた[10]. 仮想ユーザーをペルソナと呼び具体的なプロフィールを定義したペルソナの要求を満足させるシステムまたはサービスを設計する. システム開発者は, 厳密に定義されたペルソナがどのようなシステムを必要とするか, またどのようなシステムであったらペルソナは満足するのか, ということを考えシステム開発を行う. これにより, 開発者の独りよがりのシステムではなく, ペルソナを中心に据えたユーザー中心の開発が可能となる.

ペルソナ・シナリオ法の工程は以下の 6 つである[10].

①ペルソナ生成
②サービス立案
③メインペルソナ決定
④サービス選定
⑤インターフェース決定
⑥サービスシナリオ記述

ここではペルソナとして, 通学や帰省に公共の交通機関を利用している男子大学生を想定し,「電車・バスの乗り過ごしや乗り遅れを改善すること」を支援するツールを開発する設定でペルソナ・シナリオ法を実施した. 以下, ペルソナ・シナリオ法のステップ④から⑥を中心に, 評価の概要を説明する.

5.2. 被験者 A

サービス立案とサービス選定

まず初期のサービス案として以下の四つを立案した.

① 目的地と出発地の電車の発着情報を教える
② 目的地に着くときにアラームを鳴らす
③ 目的地までの時間を判定する
④ 降りたい駅の三つ前の駅でバイブレーション鳴動

サービスの選定を行い, ①については既に有名なものがあるサービスという理由で, ②についてはアラームでは他の乗客に迷惑となる可能性があるという理由でそれぞれ破棄となった. その結果, 設計の検討を行うサービスは③と④に決定した. 以下, ③と④についてはそれぞれ乗り遅れ A, 乗り過ごし A と呼ぶ.

サービスシナリオ記述

選定したサービスについて, 画面インターフェースの概要を考えながらシナリオを記述した. 乗り遅れ A の概要は, 電車やバスの駅, 停留所への家からかかる時間を逆算し, そのルートの要所ごとにそのエリアにいれば間に

合うという時間を設定し，常に画面に表示する等して知らせる．そのサービスシナリオは以下の通りである．

1) アプリを起動
2) 目的地となる駅もしくは停留所を入力する
3) 現在地との距離を計算し，到着までの見積もり時間を表示する
4) 現在地と目的地の最短ルートを表示する
5) そのルートにある要所’(コンビニ，トイレ等)を入力する
6) 距離を計算してかかる時間を表示する
7) 家から要所までのまでの距離，要所から目的地までの距離から，要所に到達しておけば良い時間を計算して表示
8) 表示した時刻より遅れている場合は，要所または目的地までどのくらいの速さで移動すれば間に合うかを表示
9) 時間通りに目的地に着く

乗り過ごしAは，位置情報を用いて最寄り駅の三つ前ほどでバイブレーションするアプリとし，そのサービスシナリオは以下の通りである

1) バスや電車に乗ってアプリを起動
2) 最寄り駅を設定
3) 位置情報より，最寄り駅の3つ前でバイブレーションする
4) 最寄り駅で降りる

画面の連鎖と画面ごとのユーザー要求の作成

サービスシナリオにもとづいて図 5のようなスケッチを描き，ユーザーストーリーマッピングを行いながら乗り遅れAの一連の画面とその画面ごとのユーザー要求を以下のように検討した．(以下，画面案は画像を略し番号の

図 5 乗り遅れAサービスの画面案(一部のみ記載)

(ア) 画面 1 と画面 2 では，ユーザーが枠をタップして目的地を入力する．
(イ) 画面 2 と画面 3 では，ユーザーが枠をタップして要所を入力する．
(ウ) 画面 4 と画面 5 の間で，システムが位置情報を取得し，マップを表示する．
(エ) 画面 4 と画面 5 の間で，システムが現在地から見た目的地と要所の距離，位置を表示する．
(オ) 画面 5，画面 6，画面 7，画面 8 で，ユーザーが目的地に到着したい時刻を入力する．
(カ) 画面 9 で，システムが目的地に到着したい時刻を表示する．
(キ) 画面 10，画面 11 で，システムが要所に到着したい時刻を表示する．
(ク) 画面 10，画面 11 で，システムが元の画面に移動する．

乗り過ごしAの画面の連鎖と画面ごとのユーザー要求は以下の通り．

(ア) 画面 1 と画面 2 で，ユーザーが最寄り駅を入力する．
(イ) 画面 2 と画面 3 の間で，システムが位置情報を取得し，マップを表示する．
(ウ) 画面 2 と画面 3 の間で，システムが最寄り駅から 3 つ前の駅を表示する．
(エ) 画面 4 で，システムがバイブレーション機能を使用する．

5.3. 被験者 B

まず初期のサービス案として以下の四つを立案した．

① 目的地に着いたときにアラームを鳴らす
② 目的地に着いたときにバイブレーションを鳴らす
③ GPS を用いた各地点でアラームが鳴る
④ 交通機関の遅延と早着を知らせる

①，②，③については内容が類似しているという点から，イメージが最も膨らんだという③を選定し，①と②を廃案にした．その結果，設計するサービスは③と④に決定した．以下，③と④についてはそれぞれ乗り遅れ B，乗り過ごし B と呼ぶ．

前述の被験者 A の場合と同様に，サービスシナリオを記述し，画面の連鎖と画面ごとのユーザー要求を作成した．(詳細は略)

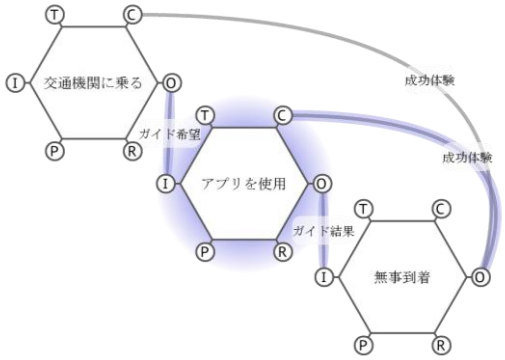


図 6 抽象レベルでの FRAM モデル

5.4. FRAM によるモデリングと ABC 分析

FRAM によるモデリングで，ユーザーストーリーの明確化を目指し，開発成果物をユーザーが使うことが動機づけられるかについて行動分析の知見の活用を試みた．この FRAM のモデル化や行動分析は同時並行的に進んだり，繰り返し行われたりする．アプリを使用する動作に焦点を当てた抽象レベルのモデルの例を図 6に示す．交通機関を使うときに時刻表などのガイドを必要と考えたときに(弁別刺激)，アプリを使用することで(対象行動)見通しよく目的地に無事到着すると(強化刺激)，その成功体験により，再び，公共の交通機関をアプリの支援を使って利用することが促進されることを表現している．詳細化を進めていく段階でも，図 7 のような FRAM モデルを作成し分析を行う．ツールとしては FMV を用い，ナラティブフロー同様に左から右に記述したり，動作主体ごとに色を変えたり，レベルに合わせて上下の位置を変えたりすることが出来る．ユーザーインターフェースを考える段階で，アプリケーション側の機能の出力がユーザー側のどの側面につながっているか，ユーザー側機能の側面で受け取るものが，ユーザーの行動の先行刺激や強化刺激になっているか，といった観点での記述や分析を行う．例えば図 7 の例では，行動分析の観点も含めて FRAM のモデル上でストーリーを追うことで，ユーザー行動の分類や，ユーザー行動の契機の不明瞭さなどに気付くことが出来た．

5.5. 比較

FRAM のモデリングと行動分析の導入により，被験者 A では，2 つのサービスそれぞれについて画面は 5 つと 2 つ増え，ユーザー要求は 4 つと 2 つ増えた．

被験者 B では， 乗り遅れ B についてはユーザー要求が4つ増え，乗り過ごし B については画面が 2 つとユーザー要求が 6 つ増えた．

このことから，提案手法では気づきを増やすことが出

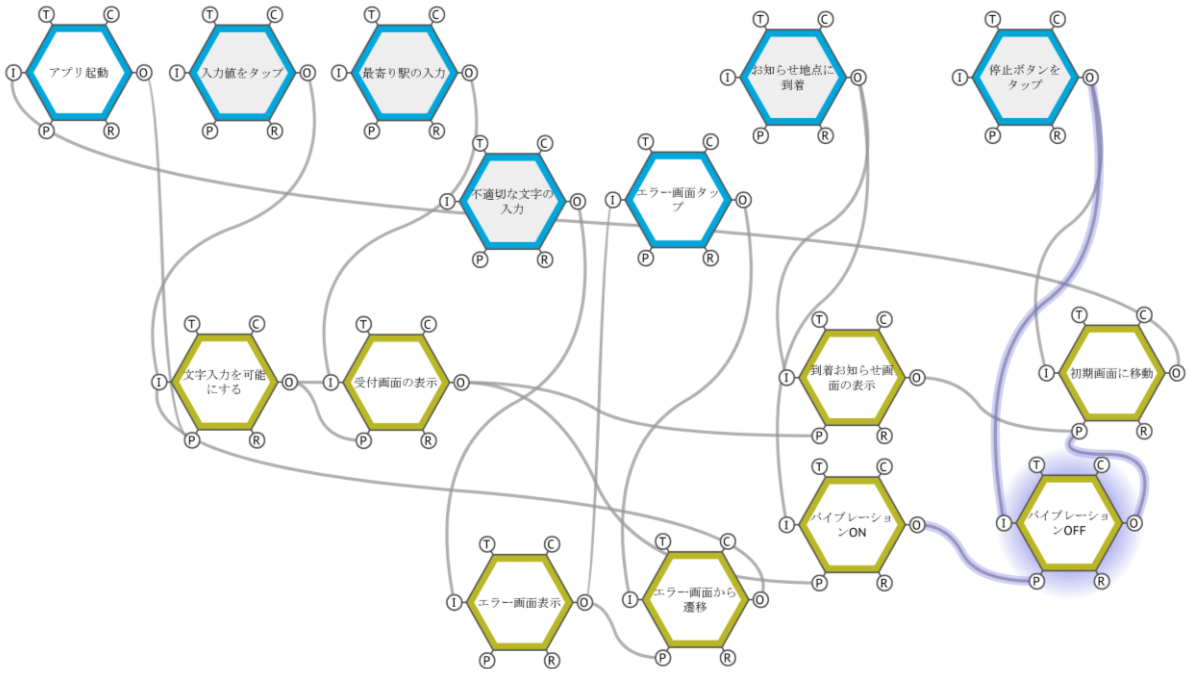


図 7 サービスの FRAM モデル例

来たと言える. カードを使ったユーザーストーリーと比べ, FRAM でのモデリングでは動詞句に相当する機能を並べるだけでなく, 機能間のどの側面に関連があるかを明示的に記述する点が異なる. そのような関連の明示は, 三項随伴性のようなアクティビティ間の関連を分析したり共有したりするのに有用であったため, 気づきが増えたと考える.

追加的にモデル記述や分析を行うのでそのために必要な時間が追加されるが, 手戻りで必要な時間より短くなりトータルの時間が短縮されている可能性もあると考える. 今回の評価では学部四年生を対象にしており, 実務経験などもないため, 何らかのガイドラインとなるものがあることが特に有効であった可能性もある. 豊富な実務経験を持つ場合は, FRAM でのモデリングや行動分析の導入をあらためて行う効果は大きくない可能性もある. 定量的比較など今後の課題である.

6. おわりに

人間中心設計 HCD のアプローチを参考に, ユーザー視点を重視したソフトウェア開発の上流工程でのシナリオ分析に機能共鳴法と行動分析を併用する方法を提案した. HCD サイクルのユーザーの要求事項の明確化のフェーズで, シナリオに対応するユーザーストーリーをユーザー視点で明確に記述し分析するために, 行動分析と機能共鳴分析法を併用することを提案した. また, 実装に向け, ユーザー側とアプリケーション側の機能の相互作用の関係を明確にすることも目指した. 大学の学部生による実験でその有効性を評価した結果一定の効果を確認した. より実験数を増やし効果の確認を続けるとともに, 提案手法の実施による追加的なコストの評価なども試みる予定である.

参考文献

[1] 人間中心設計推進機構 「人間中心設計入門編～エンジニアの方々へ～」, HCD-Net, 2018

[2] Erik Hollnagel (小松原明哲監訳). 社会技術システムの安全分析-FRAM ガイドブック, 海文堂出版, 2013

[3] 佐藤方哉, 行動理論への招待, 大修館書, 1976

[4] 角田拓磨, FRAM と行動分析を併用したシナリオ分析でのユーザー要求事項の明確化, 長崎県立大学卒業論文, 2020

[5] FMV FRAM Model Visualizer, https://functionalresonance.com/FMV/index.html

[6] 野本秀樹, 道浦康貴, 石濱直樹, 片平真史, FRAM-機能共鳴分析手法による成功学に基づく安全工学, SEC journal, 14(1), pp.42-49, Aug. 2018

[7] Jeff Patton (川口恭伸監訳), ユーザーストーリーマッピング, オライリー, 2015

[8] Mark A. Mattaini, Contingency Diagrams as Teaching Tools, The Behavior Analyst, 18(1), pp.93-98, 1995

[9] Sandy Toogood, Using contingency diagrams in the functional assessment of challenging behavior, Int'l Journal of Positive Behavioural Support, 2-1, pp.3(10), 2012

[10] 伊原誠人他, ホームネットワークシステムにおけるサービス開発へのペルソナ ・ シナリオ法の適用と評価, 電子情報通信学会技術研究報告 IN, 情報ネットワーク 106(578), pp.405-410, 2007

3D 計測点群データからの電柱抽出処理とその応用

高志毅，加藤徹，高橋弘毅，土井章男（ソフトウェア情報学部），榊原健二，細川智徳（株式会社TOKU/PCM），原田昌大氏（(株)タックエンジニアリング）

要旨

宮古市の末広町では，車の通行を抑制して，安全で安心して歩ける道路空間の整備を目指している．そこで，本研究では道路空間を 3D 計測し，3D 点群データから電柱を削除し，道路空間の整備のシミュレーションを行った．さらに，3D 点群データを 2 次元画像に変換し，ハフ変換を用いて電柱を識別することで，電柱化抽出処理の自動化を試みた．

1　はじめに

宮古市宮古駅近いにある「末広町」は，中心市街地に位置する商店街のほぼ中央を通る重要な道路であり，商店街を中心とした魅力あるまちとして，賑わい強化につなげていくためには，地域住民との合意形成を図りつつ，歩行者を優先した安全・安心，かつ快適な道路として整備を進めていく必要がある．（図 1）

しかしながら，市道末広町線通りの現状は歩道が設置されておらず，路側帯をカラー舗装で塗り分けることで歩行空間として区別している．歩行空間には電柱が設置されており，電柱を避けて通行する際や歩行者のすれ違いの際には，車道部にはみ出すことが確認されている．また，商店街への来訪や荷捌きの車両が歩行空間を跨いで停車することで，歩行空間が占用分断されており，歩行者は車道部に大きくはみ出して通行する状況が散見される．

無電柱化は「電線類地中化」と「電線類地中化以外」に大別され，無電柱化により歩道幅員を広くとることが可能となり，良好な景観が形成され，同時に快適な道路空間が創出される．しかしながら，これまでは市街地の無電柱化が十分ではなく，3D 計測を行った場合，道路には多くの電柱や電線が計測されてしまうため，自動で電柱の点群データを抽出することが求められている．

そこで，本研究では，道路の 3D 計測点群データに対して無電柱化処理を行い，無電柱化された道路空間のビジュアルシミュレーションを行った[1]．

図 1　宮古市末広町

2　3D 計測点群データの取得方法

2. 1.　計測装置およびソフトウェア

3D 計測で使用する装置は，Faro Focus3D 120 レーザースキャナー（レーザー計測装置），識別用マーカー，GNSS 測量機，座標識別点である（図 2，図 3）．本研究では，土木建築用の点群処理ソフトウェアである Autodesk ReCap とビジュアルシミュレーション用ソフトウェアである Autotodesk InfraWorks を使用している．

図 2　レーザー計測装置と識別用マーカー

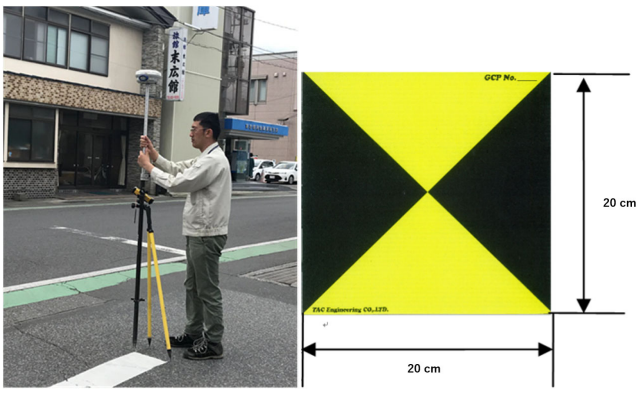


図 3　GNSS 測量機と座標識別点

2. 2.　3D 計測から点群データの取得

末広町の点群データは，地上付近にあるものに関してはレーザー計測装置により生成する．地上付近の対象はレーザー計測装置を用いて計測する．この装置は全方位へレーザー光を照射し，その反射光で距離計測を行う．こうして得られた

計測機からの距離情報を元に，3D 点群データを作成する．この時，指標となるマーカーを同時に計測する．

予め3Dモデルの位置決定のための標定点を９箇所設置し，さらにGNSS測量機で各標定点の座標を測定する[2][3]．

このようにして計測されたこ各点群データに対して，位置調整を行う．末広町全体のデータは，同時に撮影した標定点を基準に世界座標への変換を行う．また，レーザー計測によって得られたデータも同様にマーカーを基準に世界座標への変換を行い，それらを結合する[4]．

2. 3. 点群データに対する対話的無電柱化処理

点群データ無電柱化処理は，Autodesk ReCap を使用して，対話的に削除することも可能である．ReCap の「フェンス機能」を用いて，点群データの電柱部分を選択し，電柱を順に削除する．この作業をすべての電柱部分に対して行うと，作業時間は ReCap に熟練した CAD ユーザで約 6 時間を要する（図 4，図 5）．そこで，3 章では電柱のみを自動で抽出する方式を提案する．

図 4　電柱ある末広町の点群データ

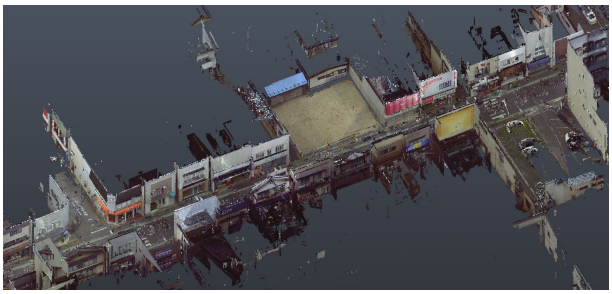


図 5 無電柱化した末広町の点群データ

3　無電柱化自動処理実現手順

図 6 は自動無電柱化処理の概要である．３次元計測点群から直方体を配置して，その直方体内部の点群から 2 次元画像を生成する．次にこの 2 次元画像に対して，ハフ変換により電柱の中心と半径を取得することで，電柱の点群データを抽出する．

電柱とは、空中に張った電線・ケーブル類を地上に引く際にこれらを支持するための柱状の工作物．図 7[5]はさまざまな種類の電柱である。

図 7[5]　様々な電柱

3. 1.　元データの分割

今回用いた点群データの総量は約 7GB であり，非常に処理速度が低下した．そのため，全体のデータに対して，ReCap で対話的に分割して，読み込み作業を行う．今回は 300MB 単位で分割した（図 8）[6][7]．

3. 2. ハフ変換で 2D 画像の円識別

ハフ変換（Hough 変換）は，デジタル画像処理で用いられる特徴抽出法の一つである．古典的には直線の検出を行うものだったが，更に一般化されて様々な形態に対して用いられている[8]．本研究では，ハフ変換の円の検出アルゴリズムを使用する．以下はハフ変換の抽出原理である．

ハフ変換の円検出原理：

2 次元上の円は，中心位置（a，b）と半径 r の 3 個のパラメータで表現できるため，図 9，図 10 に示すように，3 次元パラメータ空間内の 1 点で，円一つを表すことができる．入力画像に対するエッジ検出結果を使って，全ての円候補画素について(a, b, r)パラメータ空間への写像を行い，その結果を調べることで円のパラメータを算出できる．

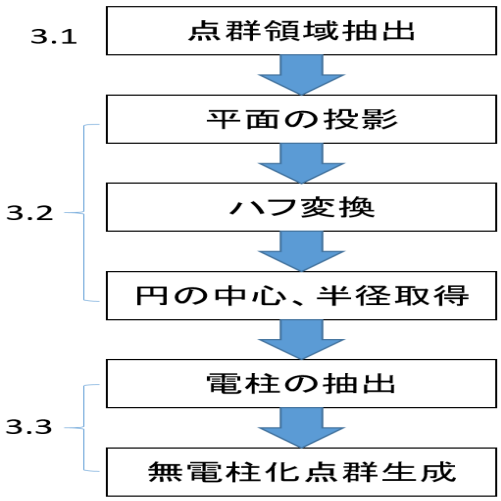


図 6　自動抽出処理の概要

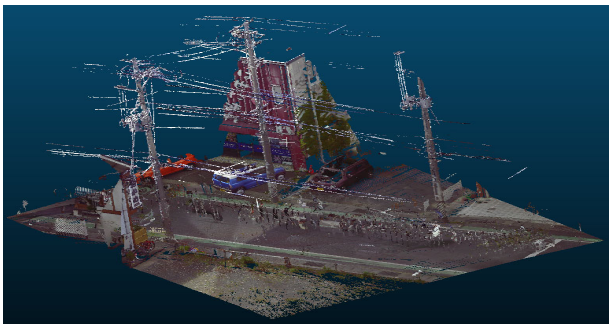


図 8 分割した点群データ例

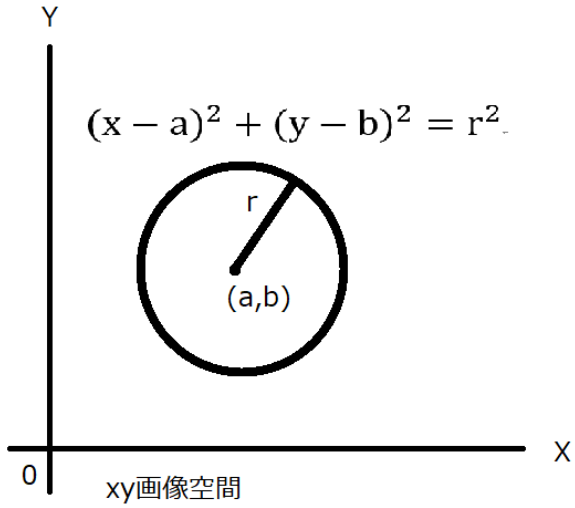


図 9　2 次元画像空間

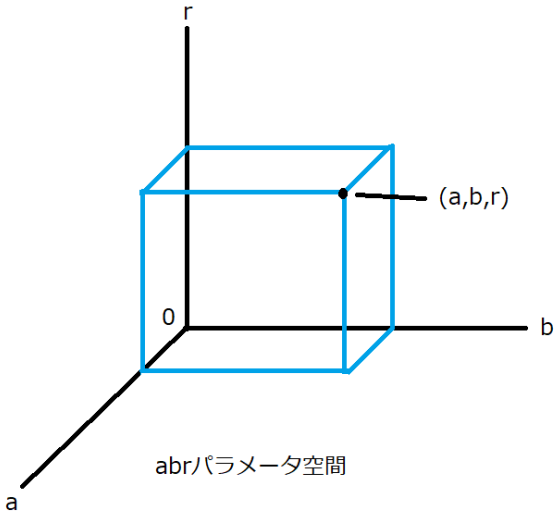


図 10　(a, b, r) パラメータ空間

電柱データは 3D 計測点群データであるため，3D 計測点群から 2 次元画像を変換することが必要である．そこで，地上から，5.0 m から 5.3 mの範囲, 7.0 m から 7.3 m の範囲, 9.0 m から 9.3 m の範囲に対して，3 次元空間内の点群を取り出し，各点群データの Z 座標を削除して，2 次元画像を生成する（図 15，図 16）[9].

3D 計測点群データから 2 次元に変換した画像は点の画像ですが，線の画像ではない，から点を繋げて、曲線を作成することは要がある．

ガウシアンぼかし（Gaussian Blur）の知識を用いて、点から曲線を作成する．

式 1 はガウシアンぼかし（Gaussian Blur）の式である．標準偏差 σ のガウシアンぼかしとは、n 次元の入力画像 A[i, j, ...] に対し n 次元ガウス関数（図 11）である[10].

$$G_\sigma(x,y,\ldots)=\frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^n}\exp\left(-\frac{r^2}{2\sigma^2}\right)\ (\text{ここで } r^2=x^2+y^2+\cdots)$$

式 1

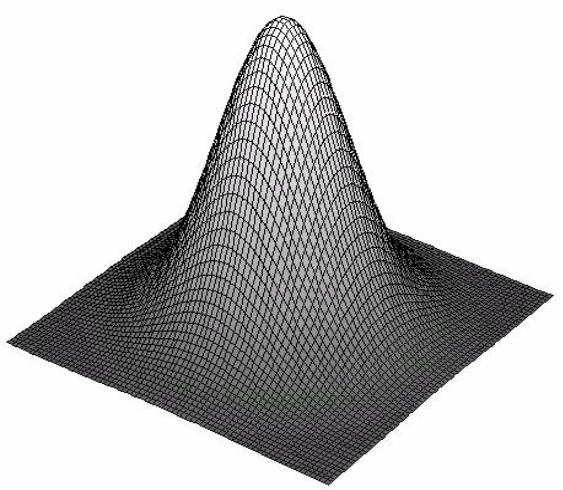


図 11　二次元ガウス関数[11]

画像処理において、ガウシアンぼかし（Gaussian Blur）とは、ガウス関数をもちいて画像をぼかす処理である．こちらの原理は 2 次元点（図 12）をガウシアンぼかし(Gaussian Blur）でぼかす処理して（図 13），黒白二値化して（図 14）、点と隣接点間のギャップを埋め、点を曲線に変換することである．

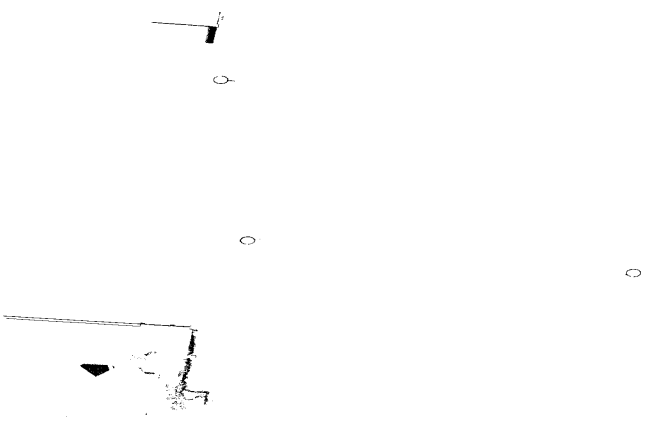


図 12　2 次元点の画像

図 13　ガウシアンぼかしした画像

図 14　黒白二値化した画像

各範囲（5m-5.3m，7m-7.3m，9m-9.3m）は使用されている電柱の高さを元に決定した．通常，5.0 mからは地上ノイズが少なく，電柱の高さは約 12mまであるが，5m，7m，9m の 3 分割で十分であった．

次に，2D ハフ変換アルコリズムを利用して，2D 図面中の円を抽出する．図 17，図 18，図 19 は，高さ 5.0 m，7.0 m，9.0m での 2 次元画像から，ハフ変換により円を抽出した結果である．抽出される円はこのように複数個の円が抽出されるため，電柱のみを選択する必要がある．

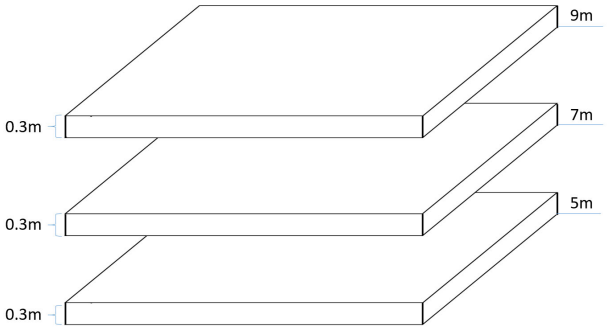


図 15　高さ 5m，7m，9mのZ軸0.3m データ抽出

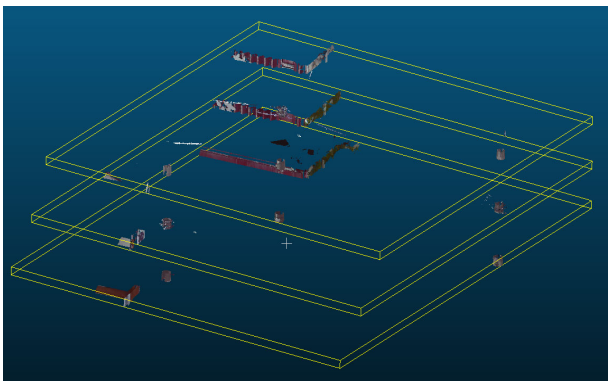


図 16　抽出した点群データ

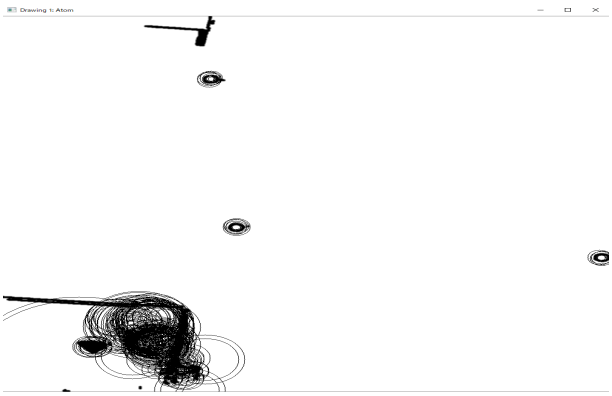


図 17　ハフ変換した 5.0 m の 2D 図面

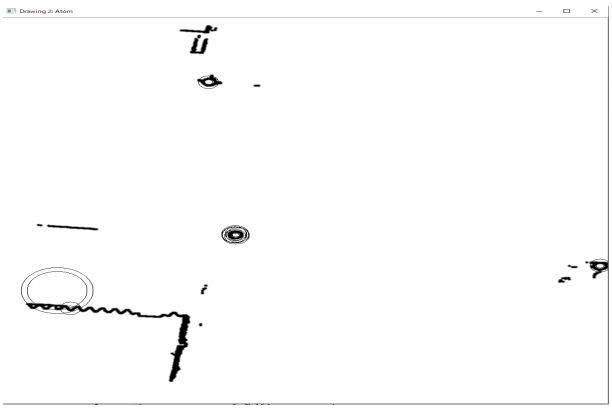


図 18　ハフ変換した 7.0 m の 2D 図面

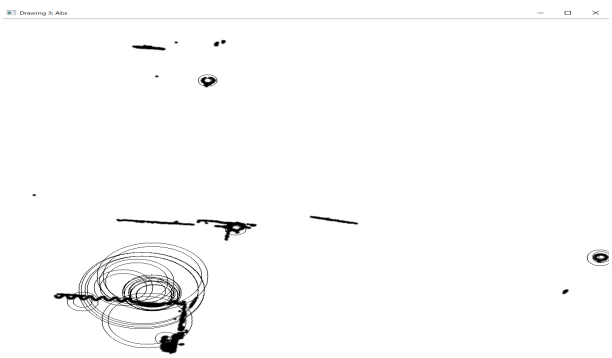


図 19　ハフ変換した 9.0 m の 2D 図面

3.3 3D 点群データ電柱の識別方式

次に各 2D 図面を並べて，同じ場合で，かつ，半径が電柱の長さの円が電柱である．図 21 はその原理を分かり易く表現した図である．最終的には円筒表面の点群が電柱の 3D計測点群データとなる．図 20 は図 17，図 18，図 19 の 2D図面に対して，抽出された電柱に対応する円を赤色で示している．

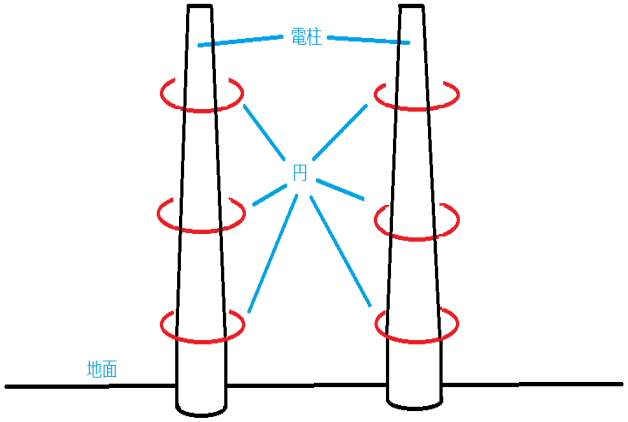


図 21　2D 図面の円から 3D 点群データの電柱識別

図 22 と図 23 は，本方式に得られた電柱の 3D計測点群データと無電柱化後の 3D計測点群データである．

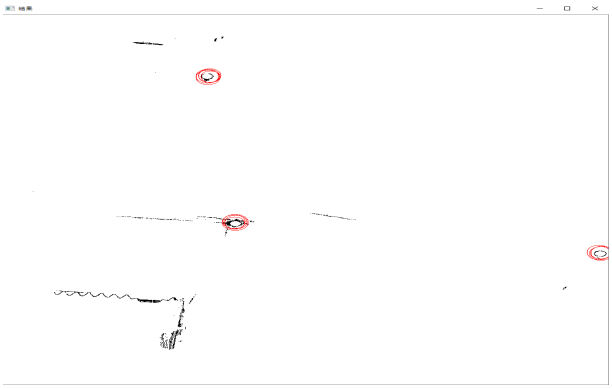


図 20　電柱に対応する円の識別（赤いマル）

図 22　抽出した電柱の点群データ

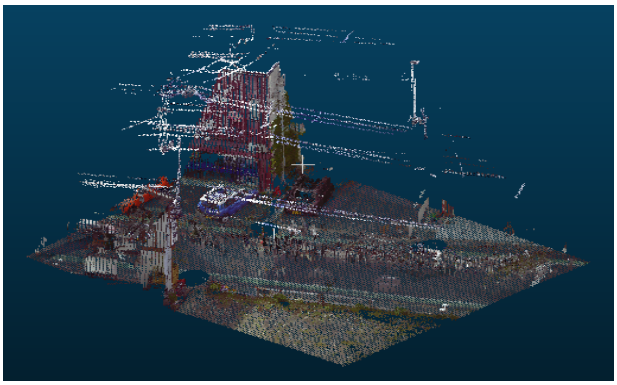


図 23　無電柱化点群データ

電柱は円錐台に近似するため、異なる高さから取得した２D 図面の同じ電柱に対応する円の半径が違う．図 24[12]を示して、R は下図面の円半径、r は上図面の円半径．半径が違うが、この二つ円の円心の二次元座標値が同じである．円心が同じ場合は電柱の可能性が高いである．

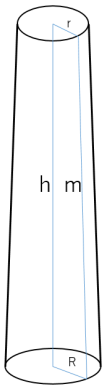


図 24　円錐台

4. 評価

4.1. 3D計測点群データからの電柱の抽出結果

末広町の点群データを４部分に分割して、計算した結果を表１に示す．

図 25, 図 26 は前半および後半の点群データから抽出された電柱の点群データ(No.1,No3)である．計算時間は主記憶 32Gバイトの汎用パーソナルコンピュータで約 8 分を要した．

番号	No.1	No.2	No.3	No.4
データサイズ	820MB	850MB	660MB	560MB
点群数	2964 万	3072 万	2385 万	2024 万
計算時間	608.4 秒	630.7 秒	420.3 秒	334.2 秒
実電柱数	10	7	6	6
識別した電柱数	7	5	4	4
各識別率	70%	71.4%	66.7%	66.7%
総識別率	68.97%			

表 1　計算結果

図 25　抽出された電柱の 3D計測点群データ(No.1)

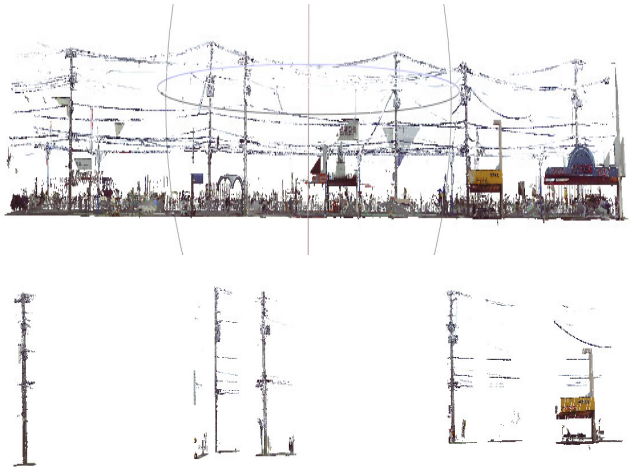


図 26　抽出された電柱の 3D計測点群データ(No.2)

図 27　抽出された電柱の 3D計測点群データ(No.3)

図 28　抽出された電柱の 3D計測点群データ(No.4)

4.2　視覚評価

無電柱化した点群データに対して, InfraWorks を用いて, 歩道のＣＧ画像を貼り合わせて, ビジュアルシミュレーションを行った（図 29）. 使用したソフトウェアは Autodesk InfraWorks である.

また, 末広町線の現状と道路計画案の比較ビデオを生成した（図 30, 図 31）[13][14][15].

5.　おわりに

末広町の道路を 3D 計測し, 膨大な点群データと世界座標値を取得した. さらに得られた点群データから, 電柱のみを自動で抽出する方式を提案した. また, 得られた 3D 計測点群データを用いて, ビジュアルシミュレーションを行った.

ReCap による対話的方式に比べて, 自動無電柱化処理は膨大な点群データに対して, 自動で識別して削除することが可能である. 本方式の問題点は, 円識別の精度があまり高くない点である. 間違った円が抽出されることが発生した.

今後の取り組みとして, 電線や人影などのノイズの削除も自動化することを検討している.

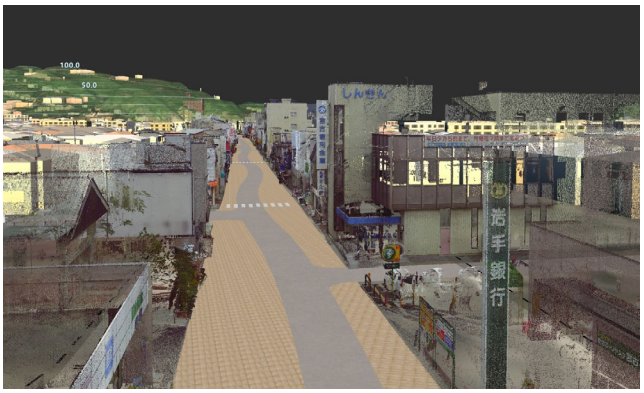


図 29　InfraWorks でモデル生成

図 30　末広町線の現状

図 31　末広町線の計画案

参考文献

[1]　“末広町通り社会実験については”　：https://www.city.miyako.iwate.jp/toshi/suehirocyo_jikken.htm

[2] 松本 裕稀, 緑川 佳孝, 齋藤 和人, 増田 宏”柱状物体の点群処理に適した非剛体レジストレーション ”，　2018 年度精密工学会春季大会，p.　567-568，2018/03/15 - 2018/03/17

[3] 辺　春日, 森　悠真, 小平　圭祐, 増田 宏“点群画像を用いた道路縁石の抽出”, 2018 年度精密工学会春季大会, p.　547-548，2018/03/15 - 2018/03/17

[4] 重田 航平, 江藤 信輔, 和田 太一, 増田　宏”工業設備の大規模点群からの部材形状の抽出と認識”，2019 年度精密工学会秋季大会講演論文集, N03, 2019/09/04

[5] 電柱 (Wikipedia)：https://ja.wikipedia.org/wiki/%E9%9B%BB%E6%9F%B1#/media/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Upoles_ottawa.jpg

[6] 峯村 晃平, 田島 晃太, 松本 裕稀, 増田　宏 ”移動計測による点群と画像を用いた物体の抽出と分類”，2019 年度精密工学会秋季大会講演論文集, N09, 2019/09/04

[7] 森　悠真, 増田　宏“機械学習を用いた道路周辺地物の自動抽出”，　2018 年度精密工学会春季大会 p.　549-550, 2018/03/15 - 2018/03/17

[8] ハフ変換 (Wikipedia)：https://ja.wikipedia.org/wiki/%E3%83%8F%E3%83%95%E5%A4%89%E6%8F%9B

[9] 増田　宏, 森 悠真”移動計測で取得した点群と画像からの道路周辺地物の自動抽出”，　画像ラボ, Vol. 30, No. 8, pp.42-47, 2019/8

[10] ガウシアンぼかし (Wikipedia)：https://ja.wikipedia.org/wiki/%E3%82%AC%E3%82%A6%E3%82%B7%E3%82%A2%E3%83%B3%E3%81%BC%E3%81%8B%E3%81%97

[11] 二次元ガウス関数 (Wikipedia)：https://ja.wikipedia.org/wiki/%E3%82%AC%E3%82%A6%E3%82%B7%E3%82%A2%E3%83%B3%E3%81%BC%E3%81%8B%E3%81%97#/media/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Gaussian.JPG

[12] 円錐台 (Wikipedia)：https://ja.wikipedia.org/wiki/%E5%86%86%E9%8C%90%E5%8F%B0#/media/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Kegelstumpf.svg

[13] 高志毅, 土井章男, 榊原健二, 原田昌大, 細川智徳, 今野公顕, ”文化財庭園の 3D モデル化と復興加速化への活用”，日本バーチャルリアリティ学会, 第 32 回テレイマージョン技術研究会, 2017/6/22-23.

[14] S. Kou, K. Satoh, A. Doi, K. Sakakibara, T.，“3D modeling of cultural property gardens and utilization for acceleration of disaster reconstruction”，AROB2018, 2018/1/19-21.

[15] 土井　章男, 大棒　秀一, 榊原　健二, 細川　智徳, 原田昌大　”3D プリンタによる景勝地 (ジオポイント), 三王岩と津波石のモデル作成と活用”　H2 地域協働研究-Ⅰ-15

オープンソース・ソフトウェア開発コミュニティにおける ライフサイクルの視覚化

増田 礼子
フェリカネットワークス株式会社
Ayako.Masuda@FeliCaNetworks.co.jp

松尾谷 徹
有限会社 デバッグ工学研究所
matsuodani@debugeng.com

要旨

オープンソース・ソフトウェア *(OSS : Open Source-Software)* は目覚ましい発展を遂げている．開発を推進する *OSS* 開発コミュニティは，*5* 年 〜 *10* 年という期間をかけて成長しており，ライフサイクルが長いという特徴がある．近年では，日々約 *5* 万件の *OSS* 開発プロジェクトが登録されているが，半年以上継続して開発活動を行っているプロジェクトは 5 % 程度であり，約 95 % は半年以内に開発を終了している．この要因の一つには，メンバのモチベーションの低下があると考えられる．本研究では，モチベーションを維持する要因を探るため，コミュニティの活動状況という観点から継続している *OSS* 開発コミュニティのライフサイクルの視覚化を試みた．視覚化は，コミュニティの開始から終了までをライフサイクルと定義し，*GitHub* をケーススタディ対象としてその変化を計測した．具体的には，メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求めた．ここでは，ライフサイクルの測り方と視覚化について報告する．この手法を用いて分析を行った結果，*OSS* 開発コミュニティが成長期にある場合には，ジニ係数の値が大きくなることが明らかとなった．本研究で提案する *OSS* 開発コミュニティのライフサイクルの視覚化手法を活用することにより，メンバのモチベーションなどについて新たな知見を得ることができる．また，*OSS* の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．

1. はじめに

オープンソース・ソフトウェア (OSS : Open Source Software) は目覚ましい発展を遂げている．Web サーバにおける LAMP (Linux, Apache, MySQL, PHP / Perl / Python) [1–6] や携帯端末における Android [7] などが核となり，さまざまなアプリケーションが開発され，無償で使用されている [8]．OSS は，普及の過程において標準化が進んでおり，企業にとっても，投資コストの低価格化という点で注目を浴びている．近年，情報産業のビジネスの中心が，ハードウェアからソフトウェアに，そしてさらにソリューションへと移行していることから，OSS が企業戦略の中に深く取り込まれたり，企業間ネットワークを形成する手段となったりしており，OSS と企業の関係は進化を続けている [9].

開発を推進する OSS 開発コミュニティは，5 年 〜 10 年という期間をかけて成長しており，ライフサイクルが長いという特徴がある．近年では，日々約 5 万件の OSS 開発プロジェクトが登録されているが，半年以上継続して開発活動を行っているプロジェクトは 5 % 程度であり，約 95 % は半年以内に開発を終了している．この要因の一つには，メンバのモチベーションの低下があると考えられる．本研究では，モチベーションを維持する要因を探るため，継続している OSS 開発コミュニティのライフサイクルの視覚化を試みる．

OSS 開発コミュニティは，企業などが主催し雇用や契約によって結束したチームとは異なる形態である．企業において一般的に用いられている「プロジェクト」とは，Project Management Body of Knowledge (PMBOK) において，独自のプロダクト，サービス，所産を創造する

ために実施する有期性のある業務と定義されている [10]. しかしながら，OSS 開発コミュニティにおけるプロジェクトは，明確な活動期限を持たないものが多く，割り当てられた業務でもないため，PMBOK の定義とは全く異なる性質を持つものである．OSS 開発は，継続的インテグレーション (Continuous Integration) により頻繁にリリースを行いながら，メンバのモチベーションによって開発が継続される．また，OSS 開発の主体も多様なコミュニティであり，開発プロジェクトの運用もコミュニティによってさまざまである．このようなことから，OSS 開発では，プロダクトの工程からライフサイクルを定義することはできない．そこで，OSS 開発を支えているオンライン上のコミュニティ・メンバの活動状況という観点からライフサイクルを捉えることとした．コミュニティのライフサイクルを視覚化することができれば，オンライン上のコミュニティ・メンバのモチベーションなどについて新たな知見を得ることができるようになると考える．また，OSS の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．

本研究では，OSS 開発コミュニティの開始から終了までをライフサイクルと定義し，GitHub [11] をケーススタディ対象とする．GitHub API v3 [12] を用いて，ライフサイクルの変化を測り，マクロとミクロの特性の 2 階層で視覚化を試みる．マクロな特性は，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求める．ミクロな特性は，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合を視覚化する．本研究では，ライフサイクルの測り方と視覚化について報告する．

2. OSS 開発コミュニティのライフサイクル計測に関連する先行研究

本章では，まず，OSS 開発の特徴を述べ，次にインターネット上のコミュニティにおける活動状況の計測に関連する先行研究をサーベイし，本研究における課題を示す．

2.1. OSS 開発の特徴

OSS の開発は，主に GitHub [11] や Bitbucket [13] といったインターネット上のソフトウェア開発のプロジェクトホスティングプラットフォームを利用して行われている．OSS 開発は，プラットフォームで OSS 開発プロジェクトを公開することにより，インターネット上で複数の開発者が並行して開発を進めていくことができる．プラットフォーム上では，多種多様なソフトウェア開発が行われるため，開発対象ごとに Repository と呼ばれる単位で一元的にデータの構成管理を行っている．OSS 開発プロジェクトの特色としては，予算を持たない，開発期間の設定がない，開発活動への参加義務がない，などがある．

一般的に，OSS 開発の利用者，開発者，愛好者らの集団を OSS コミュニティと呼ぶ．OSS コミュニティは，開発を中心とした開発コミュニティと利用を中心としたユーザコミュニティに大別できる [14]．OSS 開発コミュニティは，複数のプロジェクトを並行して開発しているものもあり，必ずしもプロジェクト間に関係性があるとは限らない．そこで，本研究では，単体のプロジェクトに相当する GitHub の Repository を計量の単位とした．そのため，本研究における OSS 開発コミュニティは，Repository 単位での集団を指す．

2.2. インターネット上のコミュニティにおける活動状況の計測

本節では，インターネット上のコミュニティの活動構造を表す研究について概観する．

北山は，29 のメーリングリストにおける投稿数を個人毎に計測し，投稿数の分布をジニ係数 (Gini Coefficient) とローレンツ曲線 (Lorenz Curve) で表した [15]．ジニ係数は，分布の偏りを表す指標であり，その分布をグラフで表したものがローレンツ曲線である．ジニ係数とローレンツ曲線は，経済学では，国民の所得がどのように分配されているのかを示す指標とグラフとして利用されている [16, 17]．北山がジニ係数を用いたのは，メーリングリストにおける個人毎に集計した投稿数の分布に偏りがあったためである．

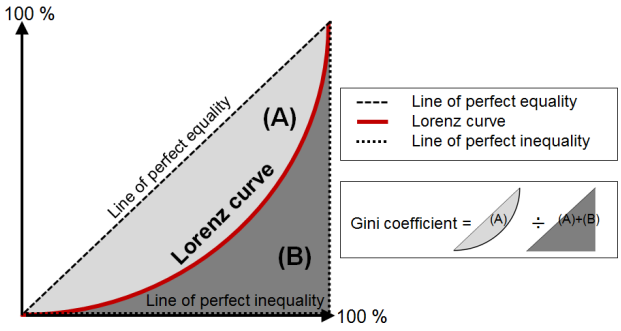


図 1: ジニ係数とローレンツ曲線

図 1 に，ローレンツ曲線の例を示し，国の世帯所得の分布を例にジニ係数とローレンツ曲線について説明する．ローレンツ曲線は，世帯を所得の低い順番に並べ，横軸に世帯数の累積比をとり，縦軸に所得の累積比をとって，世帯間の所得分布をグラフ化したものである．世帯間の所得格差が存在せず，全ての世帯の所得が同額であるならば，ローレンツ曲線は 45 度線 (Line of perfect equality) と一致する．世帯所得の分布に偏りがある場合には，ローレンツ曲線は下方 (Line of Perfect inequality) に膨らんだ形になる．ジニ係数は，ローレンツ曲線の下方への膨らみ具合を，図 1 の (A) の面積と (A) + (B) の面積の比で表したものである．ジニ係数の値は，0 と 1 の間をとるため，ジニ係数の値が 0 に近ければ分布の偏りは小さく，1 に近ければ分布の偏りが大きい．

北山の研究の他にも，ネットワーク上のデータの分析にジニ係数を用いた研究がある [18]．この Dewan の研究では，Web の閲覧行動における特定のサイトに集中する程度をジニ係数で表現し，比較している．このように，ジニ係数とローレンツ曲線は，初期の目的である所得の均等度合いを表す以外にも活用されている．ジニ係数は所得均等性の評価指標として提案された係数であるが，分布の偏りを評価したり，表したりする指標として，正規分布以外の分布を持つさまざまな事項へ応用が可能である．

ジニ係数は，計算が容易であり，データの特性をひとつの値で表すことができる．一方で，分布の形が異なっていてもジニ係数の値は同値となる場合があるため，この点には注意しなければならない．分布の偏りの程度は，ジニ係数によって表現することができる [15]．

2.3. OSS 開発における活動構造の計測

最近では，OSS 開発コミュニティの開発形態を分析した先行事例が多数報告されている．Jensen らは，3 つの OSS 開発コミュニティにおけるメンバの役割の変化を分析し，比較している [19,20]．伊原らは，OSS 開発コミュニティのメンバ間のコミュニケーションについて，メーリングリストのデータを用いてネットワーク分析を行い，開発者，ユーザ，バグ報告者の 3 者間を橋渡しする人物らが多数の参加者と常にコミュニケーションを行うことによってサブコミュニティ間の活動が円滑に行われている傾向が見られたと報告している [21]．Guimarães らによる OSS 開発コミュニティにおけるライフサイクルモデルの研究では，SourceForge [22] のダウンロード，バグ，フォーラムメッセージ，ソースコードアクティビティ情報などを用いて主成分分析を行い，スタートアップ，成長，成熟，衰退，死といった OSS 開発コミュニティのライフサイクルの段階 (Stage) によってコミュニティの有効性レベルと活動レベルに変化があることを示している [23]．

Mockus らは，Apache 開発プロジェクトの CVS データとバグ報告データを分析し，Apache コミュニティの 4 % の開発者が 88 % のソースコードを追加し，66 % の不具合修正を行っていることを明らかにした [21,24]．筆者らは，50 件の大規模 OSS 開発プロジェクトの開発記録を著作者単位で分析し，全開発量の 80 % が少数のメンバによって行われていることを示すと共に，活動実態はべき分布であり，開発規模が増加すると開発効率が向上することを示した [25]．

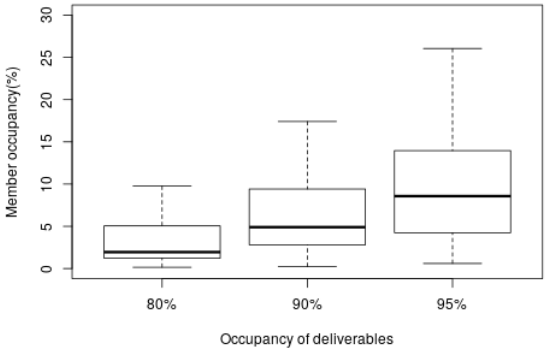


図 2: 50 件の OSS 開発 PJ における著作者毎の開発量

この研究では，著作者の活動分布に関して，図 2 に示す結果，すなわち，各 OSS 開発プロジェクトにおける全開発規模の 80 %，90 %，95 % の規模を登録した著作者数の割合の分布を測定した．図 2 より，全開発量の 80 % における著作者数の全体比率は，最小が 0.15 %，第 1 四分位が 1.2 %，第 2 四分位 (中央値) が 1.9 %，第 3 四分位が 5.1 %，最大が 9.8 % であり，極少数のメンバが開発を推進していることを示している．また，この研究では，著作者毎の開発量を時系列で集計し，順位データを作成した結果，平均値や分散などで特性を表現できないべき分布であることを示した．図 3 に，“RIOT” コミュニティ [26] の著作者毎の累積開発量の時系列グラフを示す．この時系列記録は，累積登録規模が多い上位 16 名の著作者を対象としている．図 3 のグラフ上で線が途切れていたり，途中から線が始まっていることから，活動期間 (以降，活動量と示す) がメンバ毎に異なることが分かる．

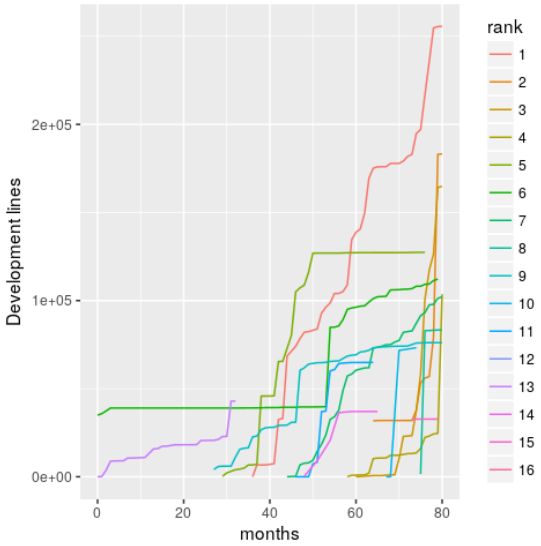


図 3: RIOT における著作者毎の累積開発量

この研究を受けて，筆者らは，開発活動の負荷分散に着目し，ジニ係数を用いて活動量の分布の計測を行った結果，OSS 開発コミュニティの活動構造の分析にジニ係数を用いることができることを示唆した [27]．しかし，この研究では，全期間の活動量を 1 つのジニ係数で表しており，ジニ係数の時系列推移を用いた分析が課題であるとしている．

2.4. 本研究における課題

本研究では，OSS 開発コミュニティの開始から終了までをライフサイクルと定義し，先行研究 [27] で示唆し，次の課題となっていたジニ係数を用いた OSS 開発コミュニティの活動構造の時系列変化をライフサイクルの変化として捉え，マクロとミクロの特性の 2 階層で視覚化を試みる．マクロな特性は，OSS 開発コミュニティメンバの活動量を Repository 単位で視覚化する．具体的には，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態をジニ係数の変化として求めることとした．また，先行研究 [24,25,27] が示す通り，ジニ係数の変化には，開発を推進するメンバ (以降，コアメンバと呼ぶ) の活動が影響している．そのため，ミクロな特性は，OSS 開発コミュニティのコアメンバに焦点を当て，コアメンバを中心とした活動割合を視覚化することとした．本研究では，ライフサイクルの計測手法と視覚化手法を提案する．

3. 分析対象と計測方法

本章では，本研究でケーススタディとして用いた分析対象と，ライフサイクルの計測手法について示す．

3.1. 分析対象の選定

ソフトウェア開発のプロジェクトホスティングプラットフォームの中でも，GitHub は最も勢いを増している [28]．そこで，本研究では，GitHub における Repository をケーススタディとして用いることとした．対象 Repository および各種データの抽出は，GitHub API v3 を用いた．

まず，2010 年 1 月 1 日から 2019 年 5 月 31 日までの期間において，乱数を用いて 200 日の特定の日を選定した．この特定の 200 日に登録された全 Repository を対象とし，次の条件に当てはまる Repository を抽出した．

(a) 更新期間が半年以上

(b) Fork 数が 1 以上

(c) Star 数が 1 以上

コミュニティのライフサイクルの計測が目的であることから，継続して開発が行われている Repository を分析

対象とするため，(a) の条件を設定した．GitHub 上に登録されている Repository には，更新頻度が低く，Contributor 数が少ないものが多いことから，個人的な入れ物として利用しているものが含まれていると推察される．Fork 数と Contributor 数の相関を確認したところ，両者には相関があり，Fork が行われていない Repository は，個人での開発であるものが多いことが明らかとなった．本研究の分析対象は OSS 開発コミュニティであることから，Contributor 数が 少ない Repository を対象から除外するため (b) を設定し，開発のためにサーバ上で Repository の複製を作成する行為である Fork が全く行われていない Repsitory を対象から外すこととした．また，OSS 開発ではプロダクト自身に価値があるものであるため (c) を設定し，OSS の利用者が評価する Star が全く付いていない Repository を対象から外すこととした．これらの条件に当てはまる Repository は，合計で 91,787 件であった．コミュニティのライフサイクルを計測するためには，ある程度の人数以上が開発に参加している Repository である必要がある．そこで，この 91,787 件の Repository から，コミュニティの人数が 30 人以上の Repository を抽出した．コミュニティの人数が 30 人以上であった Repository を抽出した結果，1,707 件となった．

続いて，OSS 開発コミュニティの多様性を考慮するため，OSS コミュニティを分類する観点を検討した．コミュニティへの参加状況は，対象 Repository に何らかの貢献を行った Contributor 数から見ることができる．開発プロダクトへの評価状況は，Star 数から見ることができる．そこで，コミュニティの人数が 30 人以上である 1,707 件を次の 2 つの観点で 4 段階に分類した．

総 Contributor 数 ：開発活動への参加状況

総 Star 数 ：開発プロダクトへの評価状況

分類には，各観点に対する 1,707 件の四分位を用いた．本研究は OSS 開発コミュニティの分析であることから，開発活動への参加状況の観点を重要視し，まず，総 Contributor 数の各カテゴリから 20 件ずつ，合計 80 件をサンプリングした．次に，サンプリングした Rpository において，もう一つの観点である開発プロダクトへの評価状況として総 Star 数のカテゴリ毎の分布を確認した所，大きな偏りなく分類できていたため，この 80 件の Repository を分析対象とした．80 件のサンプリング Repository のカテゴリ毎の分布を表 1 に示す．各観点毎のランクに記載されている値は，Rank 1 が全体の 25 % 未満，Rank 2 が全体の 25 % 以上 50 % 未満，Rank 3 が全体の 50 % 以上 75 % 未満，Rank 4 が全体の 75 % 以上に分類された Repository 数である．

表 1: 80 件のサンプリング Repository のカテゴリ分布

		Contributors			
Rank		1	2	3	4
Stars	1	4	5	5	3
	2	5	6	3	8
	3	8	5	6	5
	4	3	4	6	4

3.2. 計測手法

OSS 開発では，労務管理そのものが存在しないため，開発に費やした時間の記録は存在しない．それゆえ，OSS 開発においては，企業におけるソフトウェア開発で一般的に用いられている工数 (Person-Months) を単位として計測することは難しい．分析を行うためには，工数に相当する単位を定義する必要がある．そこで，OSS 開発コミュニティに参加し，何らかの著作物を Repository に Commit した Contributor を計測対象とし，開発期間を 1 ヶ月単位で区切り，その区間内で 1 回以上の Commit 記録があれば，1 単位として定義した．Person-Months と区別するため，ここでは Effort Person-Months と呼ぶ．

OSS 開発コミュニティは，要員を雇用していないため，稼働率をマネージできない．それゆえ，稼働率に代わる計測項目が必要となる．そこで，ジニ係数と呼ばれる指標に着目した．稼働率は，Effort Person-Months や Effort Person-Hours を Total Months か Total Hours で割った値である．Occ: 稼働率 (Occupancy rate) は，Ef: 全 Effort Person-Months の平均，To: Total Months とすれば，式 (1) で表される．

$$Occ = \frac{Ef}{To} \tag{1}$$

雇用関係が存在しない OSS 開発コミュニティでは，Total Months を固定的に決めることが出来ないので，Effort Person-Months の分布の偏りを表すジニ係数を

用いる．*Gini*: ジニ係数は，Effort の分布を $L(F)$ とすると式 (2) で表される．

$$Gini = \frac{\frac{1}{2} - \int_0^1 L(F)dF}{\frac{1}{2}} = 1 - 2\int_0^1 L(F)dF \quad (2)$$

ジニ係数を用いて稼働率に相当する値を出すとすると，式 (3) で表される．

$$Occ \fallingdotseq 1 - Gini \quad (3)$$

$L(F)$ の値は，OSS 開発コミュニティのある測定期間における，Contoributor の Effort Person-Months を用いる．i 番目の Contributor C_i の Effort Ef_{C_i} は，測定期間において活動が記録されている月数とする．たとえば，1 年間の測定期間において，3 ヶ月間に 5 回の Commit 行った場合，Commit 回数の “5” ではなく，Commit 記録のある 3 ヶ月の “3” を Effort Person-Months とする．活動量は，GitHub API v3 で取得した Commit 記録から求める．全 Contributor の Ef_{C_i} が $L(F)$ に対応する．ジニ係数は，Ef_{C_i} のベクトルから R の統計パッケージを用いて求める．

GitHub では，GitHub に登録した日より前から開発が始まっている Repository が存在し，取得したデータには登録日以前の記録も含まれている．そのため，分析対象とする Repository の開発期間はさまざまである．そこで，本研究では，それぞれの開発開始時点から 12 ヶ月単位で期間を区切り，1 年を単位として分析データを整形した．

ジニ係数を算出するための Effort Person-Months は，前述の定義に従い，次の手順で集計する．

1. Repository 単位で全期間の Contributor 毎の Commit 記録を収集
2. 収集した Commit 記録を年単位で分割
3. 年毎に月単位で Contributor 毎の Effort Person-Months を算出
4. Contributor 単位で Effort Person-Months を集計

年単位で集計，ジニ係数を算出した結果を時系列で視覚化することにより，OSS 開発コミュニティのライフサイクルの変化を観測する．

4. ライフサイクルの計測結果の視覚化

本章では，3 章で示した分析対象に対して計測，算出した結果を視覚化する手法を事例を基に示す．4.1 節で，マクロな観点からの計測結果の視覚化の事例を示し，4.2 節でミクロな観点からの計測結果の視覚化の事例を示す．

4.1. マクロな観点からの視覚化

本節では，マクロな観点からの視覚化の事例として，“compiler-explorer” コミュニティ [29] のデータを例に視覚化手法について述べる．この Repository は，総 Contributor 数，総 Star 数の両観点において，第 3 四分位以上のカテゴリに属しており，分析対象とした 1,707 件のデータの上位 25 % に属している．

表 2 に，3.2 節で示した手法により集計した値の総数と，算出したジニ係数の値を示す．

表 2: compiler-explorer コミュニティ

Year	Contributors	Total Commits	Gini Coefficient
2011	2	102	0.4803922
2012	3	32	0.5208333
2013	5	104	0.7115385
2014	3	114	0.6315789
2015	9	618	0.8137361
2016	21	1,184	0.8072716
2017	42	953	0.8189427
2018	51	756	0.8312066
All	99	3,863	0.9038247

表 2 より，2011 年は，2 人の Contributor が 102 件の Commit を行っており，ジニ係数は，約 0.48 であった．2 人の Contributor が同数の Commit をしていた場合には，ジニ係数の値は，0 となるため，片方の Contributor の開発活動が活発であったと読み取れる．また，2016 年には，Contributor は 21 人に増え，総 Commit 数は 1,184 件と増加している．この時のジニ係数は，約 0.81 と高い値になっており，Contributor が増えることにより，開発活動のばらつきが大きくなっていることが読み取れる．このように年単位で算出した値を，横軸に開発期間 (年) を取り，縦軸にジニ係数と Contributor 数を取り視覚化したグラフを図 4 に示す．

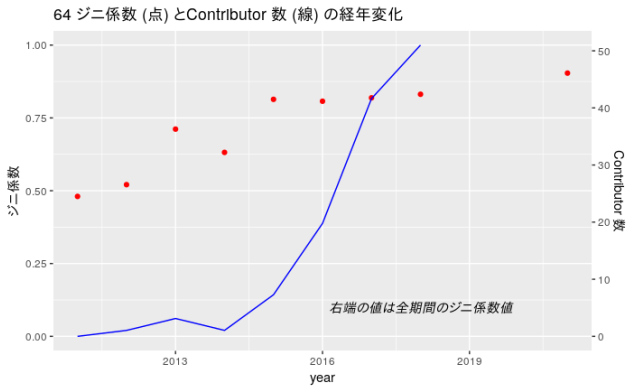


図 4: compiler-explorer コミュニティのライフサイクル

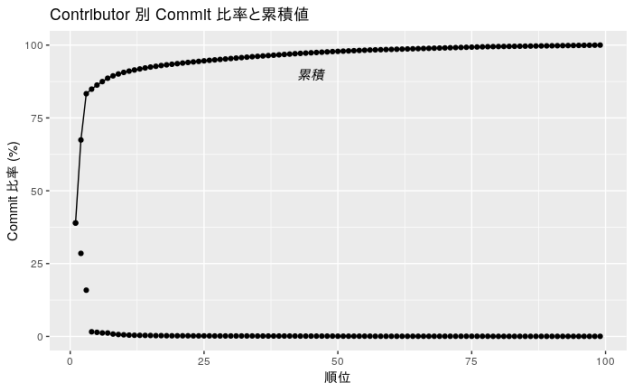


図 5: compiler-explorer コミュニティの Contributor 別の Commit 比率と累積値

図 4 の点が年毎に算出したジニ係数の値を示し，折れ線グラフは Contributor 数の変化を示している．右端の点は，全期間を対象として算出したジニ係数の値を示している．本手法により，OSS 開発コミュニティのライフサイクルをマクロな観点で視覚化することができる．

図 4 からは，compiler-explorer コミュニティの Contributor が 2015 年以降増加していることが読み取れる．一方で，ジニ係数の値は，Contributor が増加し始めた 2015 年に上昇して以降大きな変化は見られない．マクロな観点での視覚化だけでは，OSS 開発コミュニティの活動構造の詳細を読み取ることはできない．そのため，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合をミクロな観点から視覚化し，ジニ係数の変化の背景を探ることとした．次節にて，ミクロな観点からの視覚化について述べる．

4.2. ミクロな観点からの視覚化

本節では，ミクロな観点からの視覚化の事例として，前節同様に compiler-explorer コミュニティのデータを例に視覚化手法について述べる．ミクロな観点からの視覚化は，コアメンバを中心とした活動割合という観点から視覚化する．

図 5 は，横軸に，Contributor を Commit 数の多い順に上位 1 位から 100 位までを並べ，縦軸に Commit 数とその累積をパーセンテージで示したものである．

図 5 からは，全体の Commit 数の約 80% を 3 人の Contributor が行っていることが読み取れる．このことから，compiler-explorer コミュニティにおいて，Contributor 数が増加がジニ係数に大きな影響を及ぼさなかった要因が，Commit の多くは極少数のコアメンバが行っており，増加したメンバの Commit 数が少なかったためであることが明らかになった．

このように，本手法を用いてマクロな観点とミクロな観点での視覚化を組み合わせることにより，OSS 開発コミュニティのライフサイクルを確認することができる．

5. 考察

本研究では，ケーススタディとして用いた全 80 件の OSS 開発コミュニティについて，第 章で示した計測と視覚化を行った．本章では，ライフサイクルの例を示し，ライフサイクルの分類について考察する．

5.1. ライフサイクルの例

本節では，多様なライフサイクルの例として 2 件の OSS 開発コミュニティの事例を示す．

まず，“docs.larastars.cn” コミュニティ [30] の事例を図 6 と図 7 に示す．docs.larastars.cn コミュニティは，総 Contributor 数，総 Star 数の観点からは，いずれも下位 25 % 未満に属する Repository である．

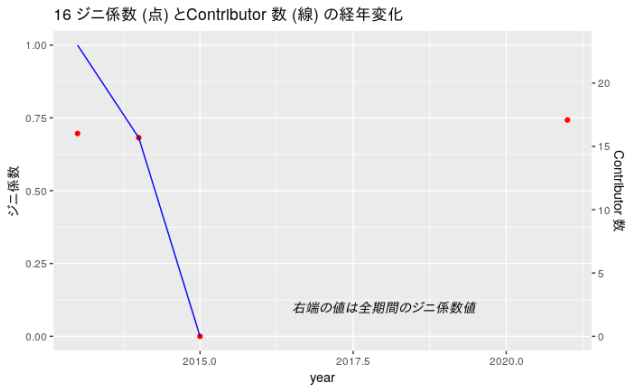


図 6: cocs.larastars.cn コミュニティのライフサイクル

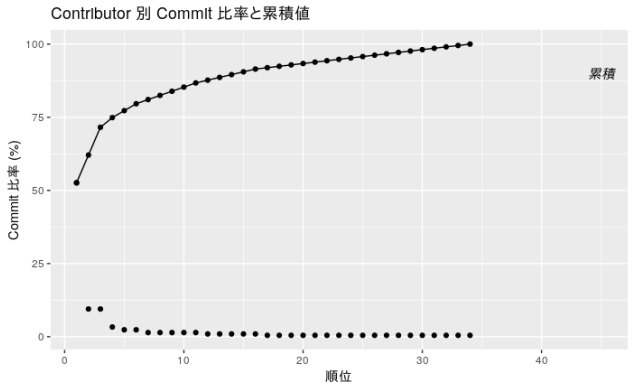


図 7: cocs.larastars.cn コミュニティの Contributor 別の Commit 比率と累積値

このコミュニティは，2013 年に開発を開始した当初は 23 人の Contributor が 107 件の Commit を行っており，ジニ係数は，約 0.70 であった．2014 年には Contributor が 16 人に減少し，総 Commit 数は 90 件と減少したが，ジニ係数は約 0.68 と大きな変化は見られない．図 7 より，cocs.larastars.cn コミュニティは 1 人の Contributor が総 Commit 数の半分を行っており，主に 3 人のコアメンバが開発を推進していたことが分かる．2015 年には，Contributor は 1 人となり，総 Commit 数も 13 まで減少し，図 6 の折れ線グラフが示す通り，以降の開発記録は見られない．

続いて，“u-boot” コミュニティ [31] の事例を図 8 と図 9 に示す．u-boot コミュニティは，総 Contributor 数は上位 25 % に属し，総 Star 数では，全体の 25 % 以上 50 % 未満に属する Repository である．

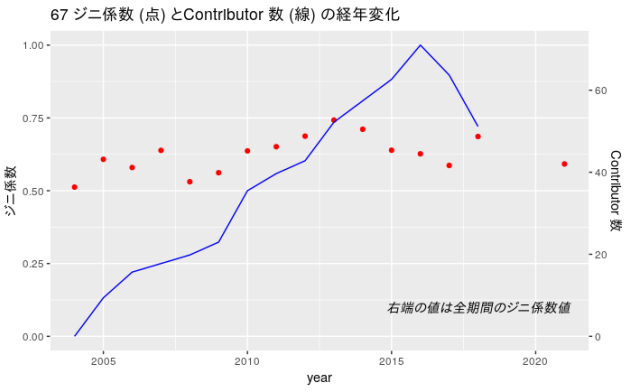


図 8: u-boot コミュニティのライフサイクル

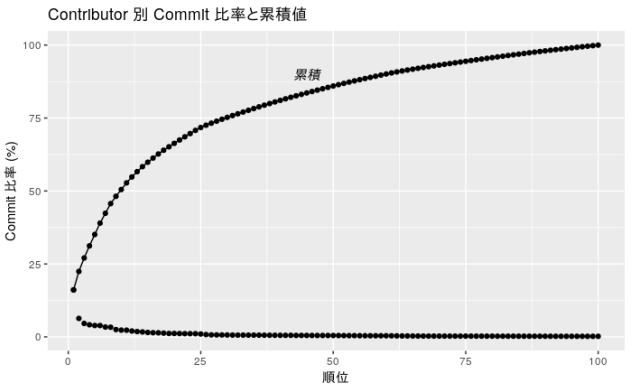


図 9: u-boot コミュニティの Contributor 別の Commit 比率と累積値

このコミュニティは，2004 年に 3 人の Contributor によって始まり，総 Commit 数は 108，ジニ係数は約 0.51 であった．図 8 からは，その後，Contributor が年々増加し，それに伴い総 Commit 数も増加していることが分かる．一方で，Contributor 数は 3 人から最大で 53 人まで増加しているが，ジニ係数の値は必ずしも Contributor 数の増減と共に高い値になっているわけではないことが分かる．図 9 より，これまでに示した OSS 開発コミュニティと比較すると，Contoributor 間の Commit 数の差異が小さいことが分かる．このコミュニティは，1 人の Contributor が全体の約 15 % の Commit を行い，それに続く Contributor が継続的に開発活動を続けていると読み取れる．

GitHub のサイト上で容易に閲覧可能な Commit 数，Contributor 数，Star 数などの値から，OSS 開発コミュ

ニティのライフサイクルや活動構造を推察することはできるが，本研究で提案した手法により，多様な OSS 開発コミュニティのライフサイクルと活動構造を視覚化することができることが明らかになった．

5.2. ライフサイクルの分類

本節では，視覚化情報を基にし，ライフサイクルパターンの分類を試みる．本研究で用いた 80 件の OSS 開発コミュニティは，継続して成長し続けるものや，成長後に衰退するもの，一度衰退して再び成長するものなど，多様であった．まず，これらの共通点を基に，OSS 開発のライフサイクルのステージを「開始期」,「成長期」,「維持期」の 3 つに分類できると考えた．

開始期 ：コミュニティの開始時期であり，参加メンバは少ない．この時期のジニ係数の値は小さい．

成長期 ：メンバが増加する時期であり，コミュニティへのメンバの参加・離脱が多く発生する．ジニ係数の値は大きくなる．

維持期 ：コミュニティへの新規参画者の減少，離脱者の増加により，持続的に活動しているコアメンバが残っている時期．ジニ係数の値は成長期より小さくなる．

本研究では，コミュニティのライフサイクルを対象としているため，開発開始後に Contributor が参加し，コミュニティと成長した OSS 開発プロジェクトである．OSS 開発プロジェクトの多くは，開始メンバ以外の Contributor が参加せず，コミュニティへと成長することなく，開始期を継続している．開始期は，Repository の登録者を中心とした開始メンバが中心となって開発しているため，ジニ係数の値は小さい傾向が観測された．成長期の新規参画者は，継続して参加し続けるタイプと，一時的に参加して離脱するタイプの 2 つのタイプに分類できる．成長期には，一時的に参加するメンバが大量に発生するため，全体のメンバ数が増加する．しかし，一時的に参加するメンバの一人当たりの活動量は少ないため，ジニ係数の値が大きくなる傾向が観測された．維持期は，新規参画者が減少し，離脱者が増加するため，開発を推進するコアメンバが中心となり活動を継続している．活動量の少ないメンバが減少するため，成長期よりジニ係数の値は小さくなる傾向が観測された．開発の終了に至る経緯は，必ずしも開始期，成長期，維持期と推移する訳ではなく，いずれのステージにおいても開発記録が見られなくなり，開発終了に至ったと想定されるライフサイクルが観測された．いずれの場合においても，開発を推進するコアメンバの存在が観測されなくなったタイミングで開発が終了している．このことから，OSS 開発コミュニティが終了するかどうかは，コアメンバが開発を持続するかどうかが大きく関係していると推察する．一方で，維持期であっても，新規参画者が増加し，再び成長期となるケースも多く観測された．これは，開発プロダクト利用者の増加などによる新たなサービスの提供や新規機能の開発着手などの要因が考えられる．OSS 開発コミュニティは，活動期限を持たず，また，業務ではないため，常に開発をし続けなければならないわけでもない．そのため，維持期や終了を迎えたとしても，メンバの知的好奇心や，利用者の要望などがモチベーションとなり，再び成長期を迎えることがあるのだと考える．

6. おわりに

本研究では，モチベーションを維持する要因を探るため，コミュニティの活動状況という観点から継続している OSS 開発コミュニティのライフサイクルの視覚化を試みた．視覚化は，コミュニティの開始から終了までをライフサイクルと定義し，GitHub をケーススタディ対象とし，GitHub API v3 を用いて，ライフサイクルの変化を測り，マクロとミクロの特性の 2 階層で視覚化を試みた．マクロな特性は，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求め，視覚化した．ミクロな特性は，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合を視覚化した．本研究で提案する OSS 開発コミュニティのライフサイクルの視覚化手法を活用することにより，メンバのモチベーションなどについて新たな知見を得ることができる．また，OSS の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．本研究では，提案した手法で OSS 開発コミュニティを視覚化できることを示すと共に，視覚化により得られた観測値から，ライフサイクルのステージとして「開始期」,「成長期」,「維持期」といった分類を行った．次の課題は，考察したライフサイクル

ステージのさらなる検証とライフサイクルパターンの分類である．特に，OSS 開発コミュニティが成長する仕組みに着目して，引き続き調査，分析を行っていく所存である．

参考文献

[1] Linux, https://www.linux.com/ (accessed:2020/03/16)

[2] Apache, https://www.apache.org/ (accessed:2020/03/16)

[3] MySQL, https://www.mysql.com/ (accessed:2020/03/16)

[4] PHP, http://php.net/ (accessed:2020/03/16)

[5] Perl, https://www.perl.org/ (accessed:2020/03/16)

[6] Python, https://www.python.org/ (accessed:2020/03/16)

[7] Android, https://www.android.com/ (accessed:2020/03/16)

[8] Lee, James and Ware, Brent, “Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP,” Addison-Wesley Professional, 2003

[9] 竹田昌弘,「オープンソース・ソフトウェアとビジネスとの関係に関する考察」, 立命館経営学, Vol.44, No.3, pp.49–66, 立命館大学経営学会, 2005

[10] Project Management Institute，プロジェクトマネジメント知識体系ガイド (PIMBOK ガイド) 第 4 版，Project Management Institute，2009

[11] GitHub, https://github.com/ (accessed:2020/03/16)

[12] GitHub API v3, https://developer.github.com/v3/ (accessed:2020/03/16)

[13] Bitbucket, https://bitbucket.org/ (accessed:2020/03/16)

[14] 石井達夫,「Let’s Postgres：OSS の開発コミュニティってどんなところ？」，https://lets.postgresql.jp/documents/tutorial/oss_community/1 (accessed:2020:0523)

[15] 北山聡, 「組織内コミュニティの計量: ジニ係数とべき分布の視点から」, 東京経済大学 コミュニケーション学会, 2009

[16] Gastwirth, Joseph L, “The estimation of the Lorenz curve and Gini index,” The review of economics and statistics, pp.306–316, JSTOR, 1972

[17] 中村和之, 「経済指標の見方・使い方：所得格差を測る指標－ジニ係数とローレンツ曲線－」, http://www.pref.toyama.jp/sections/1015/ecm/back/2005apr/shihyo/ (accessed:2020/03/16)

[18] Dewan, Rajiv M and Freimer, Marshall L and Seidmann, Abraham and Zhang, Jie, “Web portals: Evidence and analysis of media concentration,” Journal of Management Information Systems, Vol.21, No.2, pp.181–199, Taylor & Francis, 2004

[19] Jensen, Chris and Scacchi, Walt, “Modeling recruitment and role migration processes in OSSD projects,” ProSim05, Vol.39, 2005

[20] Jensen, Chris and Scacchi, Walt, “Role migration and advancement processes in OSSD projects: A comparative case study,” roceedings of the 29th international conference on Software Engineering, pp.364–374, IEEE Computer Society, 2007

[21] 伊原彰紀, 大平雅雄, まつ本真佑, 亀井靖高, 松本健一, 「複数のサブコミュニティを有する OSS コミュニティを対象としたネットワーク分析」, 情報処理学会 マルチメディア, 分散, 協調とモバイル (DICOMO2008) シンポジウム, pp.297–303, 2008

[22] SourceForge, https://sourceforge.net/ (accessed:2020/03/16)

[23] Guimarães, André LS and Korn, Helaine J and Shin, Namchul and Eisner, Alan B, “The life cy-

cle of open source software development communities," Journal of Electronic Commerce Research, Vol.14, No.2, pp.167, 2013

[24] Mockus, Audris and Fielding, Roy T and Herbsleb, James D, "Two case studies of open source software development: Apache and Mozilla," ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.11, No.3, pp.309–346, ACM New York, NY, USA, 2002

[25] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦, 「大規模オープンソース・ソフトウェアプロジェクトにおける開発効率の計測」, 電気学会論文誌 C (電子・情報・システム部門誌), Vol.138, No.8, pp.1011–1019, 一般社団法人 電気学会, 2018

[26] RIOT-OS/RIOT, https://github.com/RIOT-OS/RIOT/ (accessed:2020/05/23)

[27] Ayako Masuda, Tohru Matsuodani, Kazuhiko Tsuda, "Team Activities Measurement Method for Open Source Software Development Using the Gini Coefficient," 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp.140–147, IEEE, 2019

[28] Thung, Ferdian and Bissyande, Tegawende F and Lo, David and Jiang, Lingxiao, "Network structure of social coding in GitHub," 2013 17th European Conference on Software Maintenance and Reengineering, pp.323–326, IEEE, 2013

[29] mattgodbolt/compiler-explorer, https://github.com/mattgodbolt/compiler-explorer/ (accessed:2020/03/16)

[30] larastarscn/docs.larastars.cn, https://github.com/larastarscn/docs.larastars.cn (accessed:2020/03/16)

[31] fourkbomb/u-boot, https://github.com/fourkbomb/u-boot (accessed:2020/03/16)

CNN-BI システムの複数モデルによる精度向上のための研究

小川 一彦
放送大学
kgg03771@nifty.com

中谷 多哉子
放送大学
tinakatani@ouj.ac.jp

要旨

ソフトウェアの品質を向上させるため，これまで多くの研究が行われてきた．その方法の一つにソースコードの不具合を推論する方法がある．不具合があると推論された箇所を重点的にデバッグとレビューを行うことで品質の向上に役立てている．一例として，メトリクスを求め不具合の起こる可能性が高いロジックの推論を行うことで，品質の向上に貢献してきた．最近では，メトリクスのような統計的手法だけでなく，機械学習及び深層学習を用いて，ソースコード片からソフトウェアの不具合の混入を推論する方法などがある．我々は，ソースコードを画像化して不具合の有無でカテゴリ分けを行い，深層学習を用いて学習させた．学習の結果，作成されたモデルを使用して，プログラムの不具合を推論した．本稿は，不具合の推論を行う上で課題であった推論精度の向上を試みた．これまで，全てのプログラムを学習させて一つのモデルを作成してきたが，本稿ではプログラマを条件によりグループ化し複数のモデルを作成した．条件は，プログラムを作成したプログラマの経験年数およびスキルを評価した結果を用いた．この条件を用いて評価の近いプログラマでグループ化している．グループ化したプログラムで，それぞれ学習を行い複数のモデルを作成した．同等のスキルを持ったプログラマが作成したプログラムは，コードの品質が揃うため，深層学習において不具合の特徴を捉えやすくなる．複数の学習モデルを用いて推論した結果を，集約して全体の推論結果とすることで，精度が向上すると考えたのである．推論の精度が向上することを確かめるため，実験を行った．

1. はじめに

これまで，システム開発におけるプログラムの品質を確保するため，開発者による自己レビューや設計者との対面レビューが行われてきた．年々システムの規模は増大するが，開発費用などのコストは抑えるといった状況にある．コストが抑えられると，ソフトウェア開発は品質に悪い影響が及びやすい．成果物であるソフトウェアの作成が優先され，品質管理が後回しにされた結果，品質向上を行うために必要な期間と費用を削減せざるを得ない状況に追い込まれることがあるからである．品質管理は，コスト削減によりこれまでと同じ手法では品質を保つことが難しくなってきている．

本稿では，畳み込みニューラルネットワーク（CNN）で作成したモデルを適用し，ソフトウェアの不具合の箇所を発見させる．このシステムを，CNN based bug inference system（略して CNN-BI system）と呼ぶ [12].

研究では，プログラムの不具合の傾向を予測するため，プログラマのスキルや経験年数が異なる複数のモデルによる推論の結果，精度が向上することを検証する．複数のモデルを使用して推論をすることは，深層学習の精度向上のために行われる．

研究の目的は、以下の問いに答えることである．

- 複数のモデルを深層学習により学習した結果，不具合の推論は精度が向上したか？
- 複数のモデルで推論ができた不具合とできなかった不具合はどのような内容であったか？
- 複数のモデルの推論結果と，実際に不具合対応した内容を比較した結果，課題は何であるか？

これらの研究課題に対する答えを得るためには，プログラムの作成者をスキルと経験年数により　グループ化し

学習する必要がある．プログラムのソースコードを色付きの要素を持つ画像に変換し，CNN-BI システムを用いて学習した．

我々は，これまでプログラムを単一のモデルで学習し，CNN-BI システムを用いて推論を行い，プログラムの不具合の傾向を推論できるかどうか，適合率と再現率と F 値で評価を行った [12]．本稿では，プログラマをグループ化して学習した複数のモデルを使用し，各モデル単位でプログラムの不具合の推論を行い，推論結果を多数決により集約し，プログラムの不具合を推論する．不具合の推論の結果，精度が向上することを検証した．

本稿は，以下の内容で構成されている．第 2 節は，関連研究を示す．第 3 節は，研究内容について説明する．第 4 節で，考察を示す．第 5 節で，まとめを行い，研究の結果と今後の課題を述べる．

2. 関連研究

これまで，ソフトウェアの不具合検出は，統計学的な手法により推論されてきた．Ruchika Malhotra [8] は，品質を推論するための評価基準は、ソフトウェア開発の初期段階でソフトウェアの品質を高めるために不可欠であると言っている．CK メトリクス [2] と QMOOD メトリクス [5] を使用して設計段階で障害の可能性を推定するモデルを作成する．機械学習の手法であるランダムフォレストとバギングを使用した．

また，近藤 [10] らは，コード分析による不具合推論では粒度が荒く不具合推論のフィードバックが遅いという問題を解決するために深層学習を適用している．Yang ら [7] は，変更に対するソフトウェアメトリクス [1] [3] [4] に深層学習を適用したが，ソースコード片に対して深層学習の方法の一つである畳み込みニューラルネットワーク (CNN) から W-CNN [7] を提案し推論が可能であることを示した．

森崎 [11] はソフトウェアの品質を管理するために，ソフトウェアの読みやすさの評価に深層学習を適用した．

本稿では，CNN-BI システムを用いて，バギングを考慮した複数の学習モデルを用いる推論が，プログラムの不具合予測で，精度が向上することを検証する．

3. 研究内容

我々は，プログラムのソースコードを画像に変換し，深層学習を用いてプログラムの不具合の有無でカテゴリ分けを行ったのち，教師あり学習をした．本稿では，教師あり学習を行うにあたり，プログラマのスキルを条件としてグループ化を行う．教師あり学習の対象となるプログラムは，グループに属するプログラマの作成した全てのプログラムを用いる．学習はグループ毎に行われるため，グループの数とモデルの数は同じである．学習結果のモデルを用いて，各グループのモデル単位にプログラムの不具合を推論する．学習と推論は，深層学習の手法の一つである畳み込みニューラルネットワークにより行われる．本稿の学習及び推論を行うために，Google の TensorFlow-gpu 1.12.0 に別途 KERAS 2.2.4 をインストールした環境を用いる．OS は，Windows10 HomeEdition 64bit である．ハードウェアの構成は，CPU : Corei7-6700K，Memory : 24GB，GPU : GeforceRTX2070Super で実験を行う．

3.1. 概要

本稿では，6 人のプログラマが作成したプログラム 75 本を使用する．プログラムは，VisualBasic2008 を用いて作成されている．このプログラム 75 本を，学習データ（学習に使用するデータと学習を検証するためのデータ）とするため，画像に変換する．プログラムを画像に変換する理由を以下に示す．

1. 画像から不具合を起こすプログラムの特徴を学習させる
   不具合を起こすソースコードは，コードの記述に特徴があるはずである．その形を学習させるために，ソースコードの画像を用いる．ソースコードの特徴を画像として識別できるようにするためソースコードの構成要素毎に色分けを行う．
2. プログラム作成中の不具合推論を可能にする
   ソースコードを画像化することで，コーディングの完了・未完了を問わず不具合の推論を行うことが出来る．

3.2. 研究のアプローチ

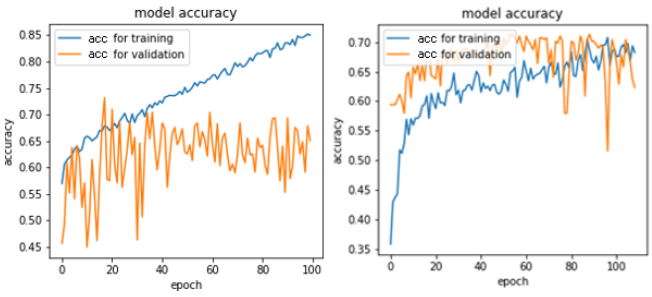
本稿の研究は，以下の手順で行う．

1. 学習データと推論データの収集.
   575 本のプログラムより，6 人のプログラマが作成した 75 本を選択した．この 75 本全てを学習に使用する．学習では，学習（training）と検証（validation）が行われ，学習データを 7 対 3 の割合で分割している．推論データは，学習に使用していないプログラムを，別途 5 本ランダムに選択した．この 5 本のプログラムの作成者は，学習データのプログラマとは異なるようにしている．すべての推論で，同じ 5 本のプログラムを用いる．

2. プログラムを画像に変換するための色分けを行う.

   - 命令文：青
   - 予約語：オレンジ
   - コメント：緑
   - モジュール名：明るい緑（太字）
   - グローバル変数：青（太字）
   - ローカル変数：黒
   - コントロール名：ピンク
   - ユーザ関数，サブルーチン名：赤
   - ユーザ定義名：ブラウン
   - 文字列：紫

3. 変換したプログラム画像より学習データを作成.
   プログラムのソースコードを 30 行単位に分割して画像に変換する．フォントは MS ゴシックで 6.8pt であり色分けがされた箇所は Bold とした．1 行は 135 文字であり，1 行に収まらない場合は行を折り返す.
   ソースコードの画像化にあたり，画像化のためのツールを作成した．不具合の修正前と修正後のソースコードを用意して，プログラムの内容を比較する．プログラムの相違のある箇所が，どのカテゴリに属するか判断し，ソースコードを画像化する．画像化されたソースコードは，画像 1 枚単位にカテゴリ分けされたフォルダに格納される．ツールを使用することで，プログラム 75 本全ての学習データの作成が 1 時間程度で出来た．手作業で作成した場合，27 本で 50 時間程必要としていたため，学習データ作成の短縮が可能となった.



Flattenを使用　　Global Average poolingを使用

図 1. 学習経過の違い

4. CNN-BI システムを使用して学習を行う.
   学習には，学習済みデータセットである VGG16 [6] を使用した．VGG16 の，最後の畳み込み層である BLOCK5 と全結合層を，研究で使用する学習データで学習する．全結合層には，Global Average Pooling(GAP) [9] を使用した．Flatten [6] を使用する方法と比較した学習の経過の違いを，図 1 に示す．図 1 のグラフは，縦軸が適合率，横軸が学習回数である．青い線は学習データのうち学習に使用したデータの適合率，オレンジの線は検証に使用したデータの適合率である．青い線とオレンジの線が共に右上がりのグラフが良い学習結果であり，過学習ではオレンジの線である検証用データの適合率が向上しない.
   Global Average Pooling を使用した学習モデルが，Flatten を使用する方法より過学習を抑制していると判断した．プログラムを画像化して学習する場合に，学習データのうち検証に使用したデータ（validation）の適合率が向上していることが図 1 より確認できるためである.
   学習のパラメータは以下の通りに設定した.

   - 最終結合層のモデル
     `GlobalAveragePooling2D()(x)`
     `Dense(256, activation='relu')(x)`
     `Dropout(0.5)(x)`
     出力は、softmax 関数を使用する.
   - モデルのコンパイルパラメータ
     `batch_size : 16`
     `optimizer:SGD(lr=0.00002, momentum=0.9, decay=1e-6, nesterov=True`

loss：categorical_crossentropy

5. 学習と推論.
本稿の学習と推論は，以下の手順で行う.

(1) プログラマの経験年数とスキルを条件として，3 つのグループに分ける.

(2) 3 つのグループが，それぞれ学習を行う.

(3) 学習した結果、作成された 3 つの学習モデルを，学習に使用していないプログラムを用いて不具合の有無を推論させる.

(4) 不具合を推論した結果を，多数決戦略により結論を得て，評価を行う.

(5) 評価結果から多数決戦略の修正を行い，再度評価する．評価は，適合率と再現率と F 値で行う．F 値は推論精度の指標である．適合率と再現率はトレードオフの関係にあるので，F 値を用いて精度を評価する.

3.3. 学習モデルの作成方法の見直し

学習モデルを作成するため，6 人が作成したプログラム 75 本を使用する．各プログラマが作成したプログラムの本数を，表 1 に示す.

表 1. 各プログラマの作成した本数

作成者	作成本数
pgmA	37
pgmB	11
pgmC	6
pgmD	7
pgmE	8
pgmF	6

表 1 のプログラムの本数は，プログラマにより本数に違いがある．続いて，6 人のプログラマの経験年数とスキルを表 2 に示す.

経験年数とスキルは，システム開発当時のプロジェクト内部の評価である．プログラマを経験年数とスキルで比較した結果，プログラマ 6 人のスキルに違いがあることが分かる．経験年数は，必ずしもスキルと一致していない．プログラマをグループ化するため，プログラマの

表 2. プログラマ毎の経験年数とスキル

作成者	経験年数	スキル (品質)
pgmA	15 年	C
pgmB	5 年	B
pgmC	6 年	B
pgmD	10 年	A
pgmE	4 年	A
pgmF	12 年	A

スキルを条件としてグループを分けた．スキルの近いプログラマであれば，グループに所属するプログラマが作成するプログラムは同程度の品質となると考えられる．グループ化した結果，学習モデルは 3 つのグループに分割した．プログラマ pgmA をグループ 1(Mdl1)，プログラマ pgmB と pgmC と pgmE をグループ 2(Mdl2)，プログラマ pgmD と pgmF をグループ 3(Mdl3) としてグループ化した．学習データは，各グループに所属するプログラマの作成したすべてのプログラムを使用する.

3.3.1 学習カテゴリの見直し

本稿では，学習カテゴリの分類方法を見直す．不具合の有無のみでカテゴリを分けるのではなく，不具合の分類でカテゴリを増やす．不具合の分類単位で，ソースコードの見え方が変わることにより，ソースコードの画像から特徴を捉えやすくなるのではないかと考えた．カテゴリ分けの結果を表 3 に示す.

表 3. 学習のカテゴリ

カテゴリ	カテゴリ名	不具合の内容
OK	不具合なし	
NG1	制御の修正	IF 文など制御構文の修正
NG2	その他修正	代入、SQL 文などの修正
NG3	ロジック追加	ロジックが挿入された
NG4	ロジック削除	ロジックが削除された

表 3 のカテゴリ分けを使用して学習を行う．学習を行ったグループ 1 の学習経過を図 2 に示す.

図 2 のグラフは，右図は図 1 と同じく適合率（accuracy）のグラフである．左図は損失 (loss）のグラフとな

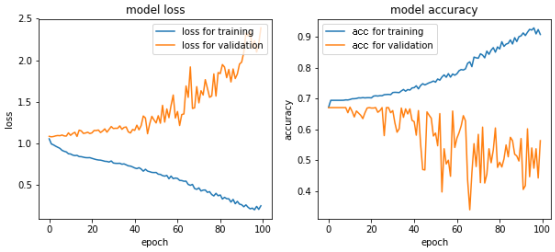


図 2. NG を 4 カテゴリにした場合の学習経過

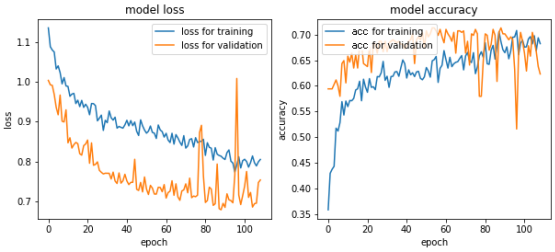


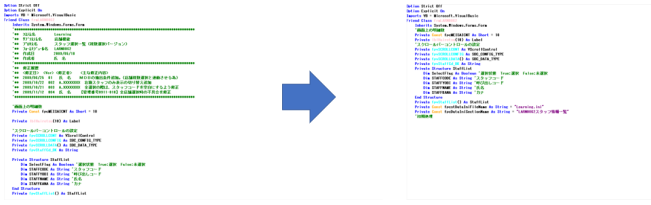
図 3. NG を 2 カテゴリにした場合の学習経過

る．縦軸は損失（loss）であり横軸は学習回数（epoch）である．
学習の経過は，学習データである学習用データ（training）および検証用データ（validation）が共に，損失（loss）は右下がりで適合率（accuracy）は右上がりになることで，過学習を起こさず正常に学習がされていると判断することができる．学習の経過を確認したところ，検証用データの損失と適合率が過学習の傾向を示した．正常に学習させるため，カテゴリ分けの再検討を行う．4 つの不具合の種類により分けたカテゴリのうち，2 つは不具合を記述の種類で分けている．残りの 2 つは，ロジックの挿入と削除であるため，記述の内容に関わらずカテゴリ分けが行われている．カテゴリ分けは，ソースコードの記述の違いで行った方が良いと考え，2 つのカテゴリに分けた．ソースコードの記述を画像化して学習データとするからである．カテゴリ分けを変更した結果を表 4 に示す．

表 4. 修正後の学習のカテゴリ

カテゴリ	カテゴリ名	不具合の内容
OK	不具合なし	
NG1	制御の修正	IF 文など制御構文の修正
NG2	その他修正	代入、SQL 文などの修正

不具合を，記述の種類で分けるという同じ基準でカテゴリを分けることで，学習データの画像の形も同じ傾向になると考えた．プログラムの記述の違いでカテゴリ分けをすると，画像化したソースコードの画像の違いがそのままカテゴリ分けとなる．2 つのカテゴリの内訳は，表 4 にある制御の修正とその他の修正である．カテゴリを変更した後，学習を行ったグループ 1 の学習経過を図 3 に示す．



リファクタリング前　　リファクタリング後

図 4. リファクタリングのイメージ

カテゴリを変更したことで，過学習の抑制が出来ている．学習データの特徴をより捉えやすくなった結果と判断した．

3.3.2　プログラムを画像化する方法の見直し

プログラムを画像化する際に，画像の内容を均一化するため，プログラムのソースコードをリファクタリングした．画像の内容を均一化することで，画像の情報量を増やし，より特徴を捉えやすくできないか考えたからである．リファクタリングのルールを以下に示す．

- ヘッダコメントの除去
- 空行の除去
- インデントは変更しない

リファクタリング前後の画像を図 4 に示す．図 4 では，左図がリファクタリング前であり，右図はリファクタリング後である．リファクタリングを行うことで，ヘッダコメントと空行が詰められている．

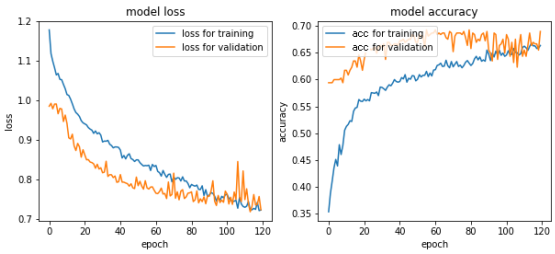


図 5. 学習経過

3.3.3 学習結果

プログラマのグループ化，学習カテゴリの見直し，プログラムを画像化する方法の見直しを行ったのち，学習データによる教師あり学習を行う．学習の経過を，グループ 2 を例に図 5 に示す．学習の経過から，学習データである学習用データ（training）および検証用データ（validation）が共に，損失（loss）は右下がりで適合率（accuracy）は右上がりになることで，過学習を起こしていないことを確認した．確認したのち，すべてのグループの学習を行う．3 つのグループが学習した件数と学習結果を，表 5 に示す．表 5 の OK 枚数と NG1 枚数と

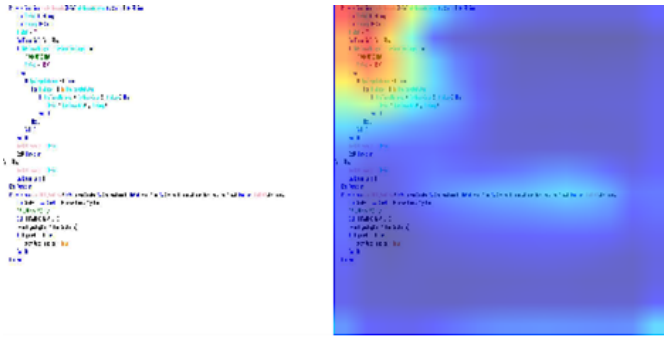
表 5. 学習件数と学習結果

グループ	OK 枚数	NG1 枚数	NG2 枚数	学習結果の適合率
Mdl1	731	405	502	71 %
Mdl2	386	103	83	70 %
Mdl3	219	48	34	81 %

NG2 枚数は各学習カテゴリの枚数である．OK は不具合なし，NG1 はロジックの不具合，NG2 はスクリプトの不具合である．

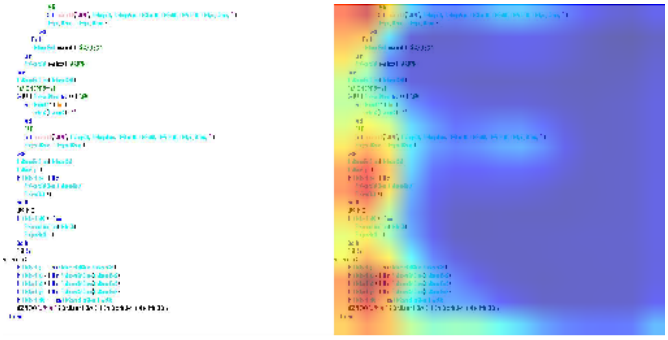
3.3.4 学習結果の視覚化

深層学習の結果をヒートマップ [13] [14] により視覚化した．視覚化により，学習モデルが画像のどの部分に着目しているかを確認することができる．記述の誤りを教師あり学習のカテゴリとしたことで，着目している箇所を目視して，カテゴリ分けの通りに学習していると判断できると考えた．以下に，可視化した結果を示す．



ソースコード　　ソースコード＋ヒートマップ

図 6. カテゴリ：OK の場合

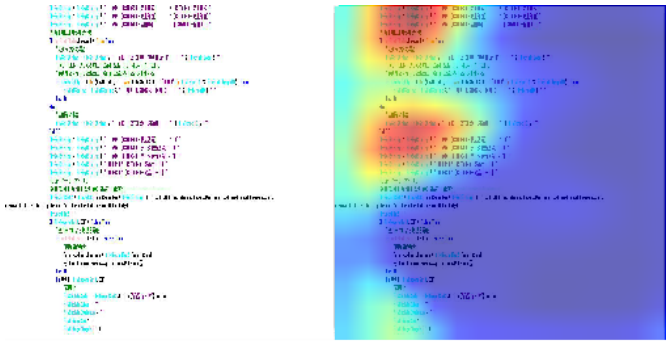


ソースコード　　ソースコード＋ヒートマップ

図 7. カテゴリ：NG1 の場合

- カテゴリ：OK
  「不具合のない場合」のヒートマップ例を図 6 に示す．
- カテゴリ：NG1
  「IF 文など制御構文の修正」のヒートマップ例を図 7 に示す．
- カテゴリ：NG2
  「代入、SQL 文などの修正」のヒートマップ例を図 8 に示す．

左図がソースコード，右図がヒートマップである．ヒートマップは着目しているほど赤色に近くなる．ヒートマップで確認した結果，カテゴリが NG1 であればロジックを全体的に，カテゴリが NG2 であればスクリプトである文字列（紫色）を中心に着目している．カテゴリ NG1 はロジックの誤りを示すカテゴリで，カテゴリ NG2 はスクリプトの誤りを示すカテゴリである．各カテゴリ単



ソースコード　　ソースコード＋ヒートマップ

図 8. カテゴリ：NG2 の場合

位に，ヒートマップをランダムに 10 例取得して確認を行った結果，全件ではないが取得した範囲で同じ傾向であったため，学習の経過に誤りは無いと考えた．

3.4. 学習モデルによる推論

深層学習により作成した，3 グループの学習モデルを使用して推論を行う．

3.4.1　推論結果

3 つのグループがそれぞれ推論した結果を集計する．

- グループ 1 の推論結果を，表 6 に示す．

表 6. グループ 1 の推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	0	0	8	8	100 %	67 %	0.8
B	0	0	2	2	100 %	100 %	1
C	0	0	21	9	43 %	47 %	0.45
D	0	0	5	0	0 %	0 %	0
E	2	0	53	37	67 %	49 %	0.57

ID　　：プログラムの ID
NG1　：カテゴリ NG1 と判断した枚数
正解　：NG1 のうち実際に起きた不具合と一致した枚数
NG2　：カテゴリ NG2 と判断した枚数
正解　：NG2 のうち実際に起きた不具合と一致した枚数

- グループ 2 の推論結果を，表 7 に示す．

表 7. グループ 2 の推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	0	0	6	6	100 %	50 %	0.67
B	1	0	6	2	29 %	100 %	0.44
C	7	2	16	6	35 %	42 %	0.38
D	11	0	12	0	0 %	0 %	0
E	9	3	49	27	52 %	40 %	0.45

- グループ 3 の推論結果を，表 8 に示す．

表 8. グループ 3 の推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	7	1	3	3	40 %	33 %	0.36
B	4	0	1	1	20 %	50 %	0.29
C	5	0	8	3	23 %	16 %	0.19
D	36	2	5	0	5 %	1 %	0.01
E	60	14	36	29	45 %	57 %	0.5

- 各グループが決定したカテゴリを多数決した推論結果を，表 9 に示す．

表 9. 各グループが決定したカテゴリを多数決した推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	0	0	6	6	100 %	50 %	0.67
B	0	0	2	2	100 %	100 %	1
C	0	0	13	5	38 %	26 %	0.31
D	1	0	4	0	0 %	0 %	0
E	2	1	38	34	88 %	47 %	0.61

- 各グループの推論した割合の数値を集計し，多数決した推論結果を，表 10 に示す．

3.4.2　推論結果の比較

推論結果は以下の 3 通りを取得する．

- グループ単位の推論結果
  推論した結果は，各カテゴリ毎に 0 から 1 の範囲で

表 10. 全体を多数決した推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	0	0	7	7	100 %	58 %	0.74
B	0	0	2	2	100 %	100 %	1
C	0	0	16	9	56 %	47 %	0.51
D	1	0	2	0	0 %	0 %	0
E	1	0	44	38	84 %	51 %	0.63

出力される．一番大きい数値を出力したカテゴリを推論結果とした．

- 各グループの推論した結果を多数決する
  グループ 1 が OK，グループ 2 が NG1，グループ 3 が NG1 と推論した場合，多数決で NG1 が推論結果となる．全てのグループが，別のカテゴリを推論した場合は OK として扱う．
- 各グループの推論したカテゴリの割合を多数決する
  各グループが推論した結果より，出力した数値をカテゴリ単位に全て加算する．一番大きい数値となったカテゴリを推論結果とする．

グループ単位の推論結果では，グループが異なると学習に使用したプログラムも異なるため，推論結果に違いが出る．全グループを多数決した結果，適合率はそれぞれのグループの良い結果を集約したように向上している．多数決は，各グループの推論結果を多数決するより，各グループの推論した割合の数値を集計する方法が，再現率および F 値で良い結果となる．

不具合の推論で重要となる再現率は，多数決をする場合とそうでない場合で，大きな変化はない．ただ，必ずしも各グループは同じ推論結果を出していない．推論結果に誤りの多いカテゴリ NG1 に，不具合は無い（カテゴリ OK）と判断していることが多かった．

しかし，各グループのカテゴリ NG1 の推論結果に，NG1 の推論結果の数値がほかのカテゴリ OK となった推論結果より，高い値を示す少数意見があることに着目した．このカテゴリ NG1 の推論結果を集計した値に，閾値を設定する．閾値は，推論が誤っている場合に，正解となるカテゴリ NG1 の平均値である 0.7 とする．推論に使用したプログラムに限定されるという前提であるが，0.5 から 1.5 まで，平均値を中心に 0.1 刻みで閾値を変更したが平均値である 0.7 のときの再現率と F 値が最も良い結果を示している．閾値を設定することで，精度に変化があるか確認するため，閾値を超えた推論結果がある場合にカテゴリ NG1 の可能性があるとして推論した結果を，表 11 に示す．

表 11. 多数決で少数意見を含めた推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	8	3	7	7	67 %	83 %	0.74
B	10	0	2	2	17 %	100 %	0.29
C	28	5	16	9	32 %	74 %	0.44
D	92	10	2	0	11 %	71 %	0.19
E	139	30	44	38	37 %	91 %	0.53

表 11 の再現率は，表 10 より向上する．しかし，F 値が大幅に下回る結果となった．

他の手法による精度向上の実現のため，各グループの推論結果を多数決するときの重みとして，プログラマのスキルを推論結果に反映させた．プログラマのスキルは，表 2 より 3 段階で評価している．このスキルを，段階ごとに A が 3 点，B が 2 点，C が 1 点で評価し，各グループの平均値を集計した．集計した結果を，表 12 に示す．

表 12. 各グループのスキル値と決定した重み

グループ	平均スキル (品質)	平均点数	重み
Mdl1	C	1.0	0.7
Mdl2	B	2.3	1.0
Mdl3	A	3.0	1.3

プログラマのスキルを考慮して，決定した重みを乗算し多数決した推論結果を，表 13 に示す．

表 13. 多数決でプログラマのスキルを考慮した推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	0	0	7	7	100 %	58 %	0.74
B	0	0	2	2	100 %	100 %	1.0
C	0	0	13	9	69 %	47 %	0.56
D	4	0	4	0	0 %	0 %	0.0
E	3	1	43	37	83 %	51 %	0.63

表 11 と同じく，閾値を超えた推論結果がある場合に，

カテゴリ NG1 の可能性があるとして，推論結果に含めた場合の推論結果を，表 14 に示す．

表 14. スキルと少数意見を考慮した推論結果

ID	NG1	正解	NG2	正解	適合率	再現率	F 値
A	8	3	7	7	67 %	83 %	0.74
B	10	0	2	2	17 %	100 %	0.29
C	28	5	13	9	34 %	74 %	0.47
D	88	9	4	0	10 %	64 %	0.17
E	136	32	43	37	39 %	92 %	0.54

4. 考察

ソースコードを画像化して，プログラムの不具合を推論した．プログラムの不具合とソースコードの記述の見た目に関連があり，画像化し教師あり学習を行うことで，プログラムの不具合を推論できると考えたためである．不具合の推論結果は，推論を行ったプログラムで精度にばらつきがあった．本稿では，精度のばらつきが，すべてのプログラマの作成したプログラムを一つの学習モデルとして学習させたため，不具合の特徴を捉えにくかったのではないかと考えた．一つの学習モデルでは限界のあった推論の精度を向上させるため，6 人のプログラマが作成したソースコードを，プログラマのスキルにより 3 つのグループに分けて教師あり学習を行った．学習の結果，各グループが学習したモデルを使用し，不具合の推論を行った．推論した結果を用いて，多数決により不具合の有無を決定することを試みた．不具合の有無を，実際に発生した不具合の結果と比較し，推論が正しいことを検証した．

4.1. 複数のモデルによる推論

プログラマのスキルでグループ化した複数のモデルによる推論は，教師あり学習に使用するソースコードが異なるため，それぞれ異なる推論結果を示した．しかし，同一プロジェクトのプログラムを使用した推論であるため，精度の高い推論結果と精度の低い推論結果は，推論したプログラムの単位に同じ傾向を示している．各グループの推論結果を多数決すると，各グループの推論結果を適合率が高い方に集約することができた．しかし，プログラマを 3 つのグループに分けたことで，各グループの学習データの件数が減少した．少ない学習データを用いた検証方法は課題となる．

4.2. 推論の精度は向上したか

複数のモデルによる推論結果を，多数決で全体の推論結果とする手法は，適合率の向上が確認できた．しかし，不具合の推論では再現率の向上が必要である．本稿では，多数決で推論結果を決定しているため，少数意見となる推論結果があった．少数意見を汲み上げるよう推論結果を修正すると，再現率が向上した．ただし，そのまま推論結果としてしまった場合，対象となる不具合の数も増加するため，適合率と F 値が減少する．少数意見として，注意喚起をするなどの手法により，レビューを行う際の判断の助けにするような方法が良いのではないかと考えた．また，ロジックの修正であるカテゴリ NG1 の推論が，全体的に精度が低い結果となっている．ロジックの修正による不具合を推論することは，複数のモデルで推論し，多数決で推論結果を修正したとしても精度を向上させることができなかった．学習結果のヒートマップではロジックに着目しているが，ロジック修正のカテゴリである NG1 の推論は，不具合なしのカテゴリ OK と判断されることが多かった．ロジックの修正による不具合を，より高い精度で推論することは，今後の課題となる．

5. まとめ

本稿では，6 人のプログラム作成者を，経験年数とスキルが近い 3 つのグループに分けた．全てのグループで，個別に教師あり学習を行い，作成した 3 つの学習モデルを用いて，各モデルに同じプログラムを推論させた．全ての推論結果を使用し，多数決によって集約した推論結果を求め，精度が向上することを検証した．検証結果から，単一のモデルで推論した結果と比較して，すべてのモデルで多数決をとることで，適合率が向上した．さらに，推論結果の少数意見を取り入れることで，完全ではないがプログラムの不具合を推論する再現率を向上させることができた．本稿の問いに対する答えを以下に示す．

- 複数のモデルを深層学習により学習した結果，不具合の推論は多数決をすることで，適合率を向上させた．

- SQL 文などの埋め込みスクリプトの推論ができているが，ロジック修正の推論の精度が低かった．
- 複数モデルの推論の精度は，各モデルの推論結果の多数決をとり，加えて推論結果の少数意見を汲み上げることで，限定的に再現率を向上させることができた．再現率向上のため，ロジックの修正による不具合の推論は課題である．

我々が作成した推論のための学習モデルは，ロジック修正のような正常なプログラムと判別が難しい不具合の推論が課題である。課題の解決のため，プログラムの不具合の場所を特定する必要がある．CNN-BI システムは，1 枚の画像に収められたソースコードの範囲で，プログラムの不具合を推論するため具体的な箇所の特定が難しい．ロジック単位で推論することにより，推論が難しい不具合についてもロジック単位の学習で精度を改善できる可能性がある．今後，不具合のあるロジックを特定させるため，プログラムのロジック単位に，図やモデルなど画像の形を変形し，学習データを作成することで推論の精度を向上させたい．

参考文献

[1] W.Li, W.Henry: Object-Poiented Metrics that Predict Maintainability, In Journal of Sytems and Software, Vol.23, Issue.2, pp. 111-122, Nov. 1993.

[2] S.Chidamber, C.Kemerer: A Metrics Suite for Object-Oriented Design, IEEE transactions on Software Engineerring, Vol.20, Issue.6, pp. 476-493, June 1994.

[3] M.Lorenz, J.Kidd: Object-Oriented Software Metrics, Prentice-Hall Inc., July 1994.

[4] F.Brito e Abreu, W.Melo: Evaluating the Impact of Object-Oriented Design on Software Quality, 3rd IEEE International Software Metrics Symposium, pp. 90-99, Mar. 1996.

[5] J.Bansiya, C.G.Davis: A Hierarchical Model For Object Oriented Design Quality Assessment, IEEE transactions on Software Engineerring, Vol.28, Issue.1, pp. 4-17, Jan. 2002.

[6] Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, pp. 1409-1556, Apr. 2015.

[7] X.Yang, D.Lo, Y.Zhang, J.SUN: Deep Learning for JustInTime Defect Prediction, 2015 IEEE International Conference on Software Quality, pp. 17-26, Aug. 2015.

[8] R.Malhotra, A.Jain: Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality, Journal of Information Processing Systems, Vol.8, No.2, pp. 241-262, June 2012.

[9] Min Lin, Qiang Chen, Shuicheng Yan: Network In Network, International Conference on Learning Representations 2014, Mar. 2014.

[10] 近藤将成, 森啓太, 水野修, 崔銀恵: 深層学習による不具合混入コミットの予測と評価, ソフトウェアエンジニアリングシンポジウム 2017，pp. 35-45, Aug. 2017.

[11] 森崎 雅稔: ディープラーニング活用事例と使いこなしの勘所 :［最適化・推論分野］6．AI によるソースコードのレビュー -ディープラーニングでコードの美しさを診断する-, 情報処理, Vol.59, number.11, pp. 985-988, Oct. 2018.

[12] K.Ogawa and T.Nakatani: Predicting Fault Proneness of Programs with CNN, 11th International Conference on Agents and Artificial Intelligence, Vol.1, pp. 321-328, Feb. 2019.

[13] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra: Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, International Journal of Computer Vision, Vol.128, Issue4. pp. 336–359, Apr. 2020.

[14] François Chollet: Python と Keras によるディープラーニング , 株式会社マイナビ出版, 第 5 章, pp. 166-186, Dec. 2018.

経食道心エコーシミュレーションソフトウェアの開発

高橋 弘毅† 加藤 徹† 土井 章男† 朴澤 麻衣子‡‡ 森野 禎浩‡‡

†岩手県立大学 〒020-0693 岩手県滝沢市巣子 152-52
‡岩手医科大学 〒020-0023 岩手県盛岡市内丸 19-1
E-mail: †t-hiroki@iwate-pu.ac.jp, ‡ymorino@smile.email.ne.jp

あらまし 経食道心エコー検査は，エコー画像のみを参照しながら医師が手動でプローブの深さ，扇状の超音波照射角度を感覚的に調整して行う．しかし，エコー画像のみでは，心臓の 3 次元的な位置把握が困難である．そこで，CT 画像を用いて，経食道心エコー検査をシミュレーションする診断支援システムを構築した．本システムは，CT 画像から経食道を指定して，その位置に心エコー装置のプローブを配置して，心エコー風の画像を作成する．

1. はじめに

狭心症や心筋梗塞などの心臓疾患を特定する方法として，経食道心エコー検査が挙げられる．経食道心エコー検査は，エコー画像のみを参照しながら医師が手動でプローブの深さ，扇状の超音波照射角度を感覚的に調整して行う．しかしながら，エコー画像のみでは，心臓の 3 次元的な位置把握が困難である．また，経食道心エコー画像は画質が低く，部位の位置や形状を正確に捉えることが困難である．検査の際には嘔吐を防ぐために麻酔が必要となるため，検査前の食事制限や検査後の休憩が必須である．そこで，CT 画像を用いて，経食道心エコー検査をシミュレーションする診断支援システムを構築し，その有効性を評価した [1].

2. 経食道心エコー診断支援システム

本研究で作成した経食道エコー診察支援システム(以下，本システム)は以下の機能を有する．

2.1. 心エコー画像を模した CT 断面の表示

心エコー画像はプローブ位置を中心とした扇形の画像である．本システムはこれを模倣し，仮想プローブ位置から指定した範囲，角度の CT 断面(以下 CT エコー画像)を生成する．これにより心エコー画像との対応付けが容易となる．

2.2. エコー領域とボリュームデータの同時表示

ビームシミュレーションにより表示している領域を，ボリュームレンダリング表示と同じ三次元空間に表示する．この機能により，本システムを実際の心エコー検査と同時に使用することで，医師が操作中のプローブの現在位置を三次元的に正確に把握することが可能となる．

2.3. エコー領域の計測

現在表示中の部位に対し，mm 単位での距離計測及び面積計測が可能である．また，距離計測の際の始点，終点はボリュームレンダリング上の CT エコー画像上にも表示され，三次元空間上の線分として見ることができる．

2.4. 疑似的な心エコー画像表示

CT 画像に対して，マスク画像を用いたフィルタ処理で，心エコー画像に近づけることにより，心エコー画像と CT 画像の対比を容易にできる．

2.5. 経食道指定

疑似プローブを CT 画像内の経食道に沿って移動させるために，医師が目視で経食道を指定し，プローブはその軌道上を移動する．

3. おわりに

CT 画像から擬似的な心エコー風の画像を作成し，CT 画像と心エコー風画像との対比を行うことで，より正確な診察が可能になった．さらに心エコー風画像上での距離や面積を計測することでより正確な術前計画策定が可能となる．

また，経食道心エコーによる診察の品質向上を目的に，CT 画像から疑似経食道心エコー画像を作成した．医師からは高い評価が得られ，本システムの有用性が確認できた．しかしながら，CT 画像のエコー画像風表示や，より直感的なインタフェースの実装などに関しては改善の余地がある．

謝辞

本研究では，公益財団法人 JKA の「平成 28，29 年度心臓カテーテル手術を支援する心臓定量化ソフトウェアの研究開発補助事業」の研究支援を得ています．

参考文献

1) M. Hozawa, Y. Morino, Y. Matsumoto, R. Tanaka, K. Nagata, A. Kumagai, A. Tashiro, A. Doi, K. Yoshioka, “3D-computed tomography to compare the dimensions of the left atrial appendage in patiens with normal sinus rhythm and those with paroxysmal atrial fibrillation“, Journal of Heart and Vessels, ISSN 0910-8327, Springer , 2018.

動画配信事業におけるプロジェクト管理

清水　祐作
千葉工業大学
s1742051jq@s.chibakoudai.jp

小笠原　秀人
千葉工業大学
hideto.ogasawara@p.chibakoudai.jp

要旨

近年の動画広告市場規模の拡大に目をつけ，多くの企業が YouTube 等の動画配信サービスを使った事業（以後 YouTube 事業とする）を立ち上げているのをよく耳にする．しかし，この YouTube 事業は，多くのタスクが同時並行で進むマルチタスクなプロジェクトであることが多いため，スケジュール管理を容易に行えない．筆者が参加した YouTube 事業立上げプロジェクトでも同様に，スケジュール管理が上手く出来ておらず，プロジェクトが思うように進まないことも少なくなかった．

本論文では，実際に 筆者が参加した YouTube 事業立上げプロジェクトで起こった問題やその解決策等を記す．

1. はじめに

2019 年国内動画広告市場の規模が前年比 41%増の 2592 億円になる見通しである（図 1）．その背景には，動画配信サービスや SNS の利用拡大があると考えられる．また，20 年以降に次世代通信規格「5G」が普及すれば，さらに市場の拡大が続くだろう．

この動画広告市場（特に YouTube）の拡大に目をつけ，多くの企業が参入してきているのが現状と言える．しかし，この YouTube 事業の運営をする上で，筆者はスケジュール管理の難しさを感じている．その理由として，複数のチャンネルを運営する上で，1 つ 1 つのチャンネルの動画撮影や動画編集などの工程を全て同時並行して行うマルチタスクなプロジェクトであることが多く，スケジュール管理が容易ではないことが挙げられる．このスケジュール管理が上手くいかないことによって，コンテンツの質の低下だけでなく，それらによる視聴回数や視聴者定着率の低減が引き起こされることが想像できるだろう．

本論文では，実際に筆者が参加した YouTube 事業立上げプロジェクトで起こった問題やそれに対する解決，それらの効果等を記す．

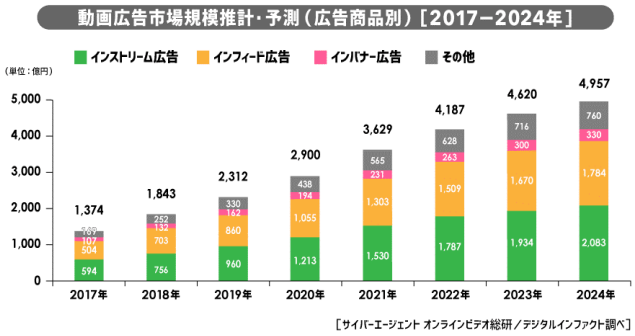


図1 動画広告市場規模推移・予想[1]

2. 目的

1 本の動画を制作していくだけでも，多くの作業工程が必要である。例えば，最近，動画配信サービスや SNS 上で流行している「漫画動画」を作るには，図 2 を見てわかる通り主な工程だけでも，7 つの工程を全て行う必要がある．また，シナリオはライター，漫画はイラストレーター，音声は声優，動画は動画編集者に委託することが多い．そのため，1 本の動画を制作するのに 1 ヶ月程度かかる．1 週間に複数の動画を投稿するとなると，この制作を同時並行でいくつも行うことになる．そのため，1 つのチャンネルのスケジュールを管理するだけでも多くの時間と労力がかかる．それだけでなく，スケジュールの管理ミスによる遅延などのリスクも高くなることが考えられる．

そこで，このようなマルチタスクプロジェクトにあったスケジュール管理を行うことで，少しでも時間や労力，リスクを低減しようと考えた．

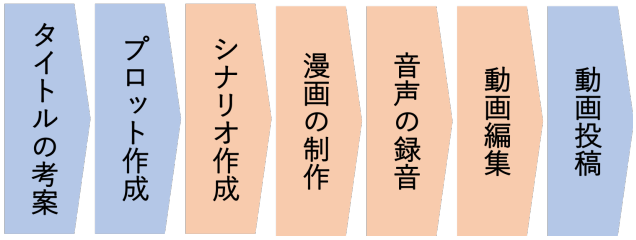


図2 漫画動画の主な制作工程

3. プロジェクトの概要

プロジェクトの概要を以下の表にまとめる.

表1　プロジェクト概要

対象	YouTube 事業立上げプロジェクト
メンバー	5〜6 名
内容	漫画動画チャンネル，玄人のチャンネル，ニュース，Vlog，インタビュー，バラエティ，etc (3 月 16 日時点では全 9 チャンネル)
開発方針	自社企画・・・漫画動画，インタビュー，バラエティ，ニュース 提携案件・・・その他のチャンネル
配信頻度	定期配信(毎日，週3，週1，月2，・・・)であり，チャンネルによって配信頻度は様々
体制	・リーダー1名を含むフラットなネットワーク型チーム(図 3 参照) ・必要に応じて専門性(ディレクター，営業，放送作家，etc)を持った別チームのメンバーを巻き込む ・外部委託(漫画等をフリーランスに委託する)
企画方法	・チームで検討 ・必要に応じて専門性を持った他チームのメンバーに相談
開発方法	コンテンツ毎に独自のプロセスを持つ

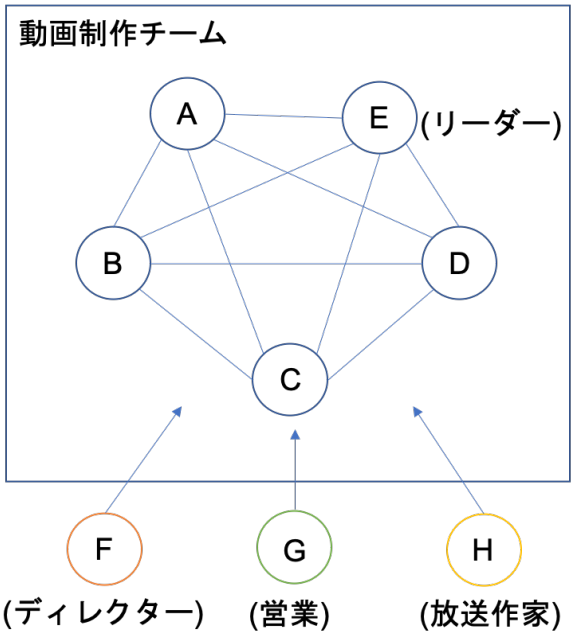


図3　ネットワーク型チームの体制

4. 課題

今回は，前章で示した YouTube 事業立上げプロジェクトの中でも，特に漫画動画チャンネルの運営に焦点を当てる．そして，このチャンネル運用を進めていく中で発生したいくつかの問題が，なぜ起こったのか，その理由として考えられることを以下に記す．

⑴時期やメンバーによって，このチャンネル運営に割ける時間が少ない，又はほとんどない．

そもそも，それぞれのメンバーは，このチャンネル運営の他にも，多くのチャンネルの運用に携わっている．また，このプロジェクト以外のタスクをこなさなくてはならない中，このチャンネルの運用を分担しているため，メンバーによって，タスク量が多くなり過ぎてしまう状況が，少なからず見受けられた．

⑵次のタスクに進まず停滞してしまう．

これは，プロットを作成したメンバーが，そのままシナリオをライターに依頼するものだと考えているメンバーと，それらのタスクは別だと認識しているメンバーが混在し，プロットを作成したままという状況が出てしまうようなことが起こった．

⑶誰がどこまで関与しているのかわからない

このチャンネルが突発的に始まったこともあり，誰がこのチャンネル運営に関わるのかというものを示される機会がなかった．そんな中，チャンネル開設直後は，他チームのメンバーも運用に携わっていたため，誰がどのタスクをどのくらい請け負っているのかが知り得なかった．

⑷大幅な見積もりミスとリスク管理不足

漫画動画を作成するのには，2章でも記したように多くの人と時間が必要となる．また，外注するフリーランサーによっても，能力だけでなく，案件に対する真剣度合いもそれぞれ違う．そのため，外注先によって見積もりを変化させるだけでなく，リスク管理もしなくてはならなかった．しかし，このような外注メインのコンテンツ作りに不慣れであったこともあり，当初は冬休み期間に毎日投稿するつもりが，図 4 のように，毎週月・金曜日の 2 本投稿するのがやっとであった．

図 4 は，当時スケジュール管理に使用していたスプレッドシートである．これを見るとスケジュールの見積もりがしっかりと出来ていないため，予定日が半分程度しか定まっていないことがわかる(図中の橙枠内部)．

作品タイトル	公開予定日	状態	漫画発注日	漫画納品日	漫画締め切り	録音日	編集開始日	編集締め切り
	2019/10/18(金)	公開						
	2019/10/21(月)	公開						
	2019/10/26(金)	公開						
	2019/10/30(月)	公開	2019/09/26	2019/10/04				
	2019/11/01(金)	公開	2019/10/08	2019/10/17				
	2019/11/04(月)	公開	2019/10/04	2019/10/17				
	2019/11/08(金)	公開	2019/10/1	2019/10/15				
	2019/11/11(月)	公開						
	2019/11/15(金)	公開						
	2019/11/18(月)	公開	2019/10/23	2019/11/05				
	2019/11/22(金)	公開	2019/10/23	2019/11/08				
	2019/11/25(月)	公開	2019/10/18	2019/11/07				
	2019/11/29(金)	公開	2019/10/28	2019/11/12	2019/11/29	2019/11/26	2019/11/27	2019/11/29
	2019/12/02(月)	公開	2019/10/30	2019/11/13	2019/12/16	2019/12/17	2019/12/18	2019/12/20
	2019/12/06(金)	公開	2019/11/16	2019/12/01	2019/12/06	2019/12/10	2019/12/11	2019/12/13
	2019/12/09(月)	動画作成中	2019/11/22	2019/12/02	2019/12/16	2019/12/17	2019/12/18	2019/12/20
	2019/12/13(金)	動画作成中	2019/10/30	2019/12/03				
	2019/12/16(月)	音声依頼中	2019/11/11					
	2019/12/20(金)	動画作成中	2019/11/24		2019/12/13	2019/12/16	2019/12/17	2019/12/19

図 4 12 月まで使用していたスプレッドシート

5. 提案

前章の課題を受けて，筆者は以下の3つの提案をした．また，それらの提案の方針と目的，方法をまとめたのが表2である．

表2 提案内容とその目的

課題	方針と目的	方法
(1) 時間が足りない	①メンバーのタスクを可視化 →仕事の分担	Trello（タスク管理ツール）の利用
(2) 停滞	②誰がどのタスクを行うか決め，共有する →実行責任が生まれる	定例会議を設ける
(3) 誰が関与しているか分からない		
(4) 見積もりミスとリスク管理不足	③大幅な見積もりを自動入力 締め切りが近くなったら進捗情報の確認 →リスク管理	スプレッドシートの改善

以下，個々の取り組みについて説明する．

① メンバーの抱えているタスクとその進捗の可視化

今回例に挙げている YouTube 事業立上げプロジェクト以外にも，メンバーはそれぞれ他のプロジェクトにも参加していたり，漫画動画チャンネル以外のチャンネル運用を日々こなしていたりする．そのため，メンバー間で誰がどれくらいの仕事を抱え込んでいるのかが分からなかった．そこで，メンバー同士でそれぞれが抱えているタスクの量を共有することで，仕事の分担を各々でしやすくなると考えた．そこでTrello[2]というツールを用いて，メンバーの抱えているタスクとその進捗を可視化できるようにした（図 5）．そうすることで，メンバー間でタスクが多い人からタスクの少ない人への担当作業の受け渡しを容易にできるようになった．これにより，前章でも記した，時期やメンバーによって，このコンテンツに割ける時間が少ない，又はほとんどないという問題の解決ができるのではないかと考えた．

② 定例会議を設ける

誰がどのタスクをどこまで進めるのか，進めたのかということがスプレッドシートだけでは読み取れず，進められずに停滞してしまうタスクが見受けられた．そこで，週1回の定例会議を行うことで，誰がどのタスクを何処まで実行したか，何処まで進めるのかということを確認し合うようにした．そうすることで，タスクを任されたメンバーには実行責任が生じ，タスクを期間内にこなせないということが減った．これにより，前章で述べた次のタスクに進まず停滞してしまうことや，誰がどこまで関与しているのかわからない事態を防げると考えた．また，この定例会議では，スケジュール管理についての話し合いだけでなく，これからのコンテンツの改善点や，動画の内容について話し合う場とし

ても機能させ，コンテンツ自体のクオリティの向上にも繋がるのではないかと考えた．

③　スプレッドシートの整理

図 4 を参照し，前章でも触れたようにスケジュールを管理するツールとして，スプレッドシートを用いていた．しかし，全てが手動での入力であったこともあり，予定日が半分程度しか定まっていなかったのが事実である．

そこで，今までプロジェクトを進めてきた中での経験から，それぞれの作業にかかる時間をおおよそ予想し，制作する動画の公開予定日を入力することで，図2で示した7つ全ての工程をいつ始めて，いつまでに終わらせればよいか自動で入力されるようにしようと考えた．そして，それと共に，1つ1つのタスクの締切り日が近くなると，図6からもわかるように，可視化して緊急性を見て取れるよう改善することにした．また，4 章(2)で挙げた問題の解決策の1つとして，進捗状況の項目を増やすことで，より細かい進捗がスプレッドシートから読み取れるように改善した．図 6 は実際に整理されたスプレッドシートであり，改善前と改善後のスプレッドシートの違いをまとめたのが表 3 である．

表 3　スプレッドシートの主な変更点

項目	改善後
アラーム	各タスクの納期が近くとそれに合わせたアラームが出る
進捗状況	進捗状況項目の増加：8→13
スケジュール入力	公開予定日を入力すると，その他の予定日を自動入力

これらの提案が実際に実行された結果を次章に記す．

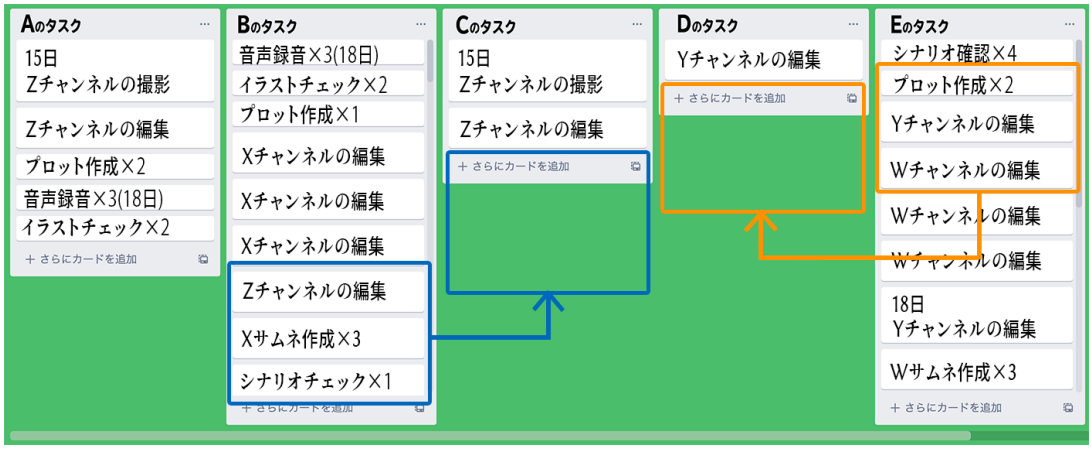


図 5 Trello の使用例

作品タイトル	進捗状況	漫画締め切り	録音日	編集開始日	編集締め切り	公開予定日
[redacted]	再掲載	2020/02/13	2020/02/14	2020/02/15	2020/02/17	2020/02/17
[redacted]	公開済	2020/02/17	2020/02/18	2020/02/19	2020/02/21	2020/02/21
[redacted]	公開済	2020/02/20	2020/02/21	2020/02/22	2020/02/24	2020/02/24
[redacted]	公開済	2020/02/22	2020/02/23	2020/02/24	2020/02/26	2020/02/26
[redacted]	公開済	2020/02/24	2020/02/25	2020/02/26	2020/02/28	2020/02/28
[redacted]	公開済	2020/02/26	2020/02/27	2020/02/28	2020/03/01	2020/03/01
[redacted]	公開済	2020/02/27	2020/02/28	2020/02/29	2020/03/02	2020/03/02
[redacted]	予約済	2020/02/28	2020/02/29	2020/03/01	2020/03/03	2020/03/03
[redacted]	動画作成中	2020/02/29	2020/03/01	2020/03/02	2020/03/04	2020/03/04
[redacted]	再掲載	2019/12/16	2019/12/17	2019/12/18	2019/12/20	2020/03/05
[redacted]	動画作成中	2020/03/02	2020/03/03	2020/03/04	2020/03/06	2020/03/06
[redacted]	再掲載	2020/03/03	2020/03/04	2020/03/05	2020/03/07	2020/03/07
[redacted]	再掲載	2020/03/04	2020/03/05	2020/03/06	2020/03/08	2020/03/08
[redacted]	漫画家清書中	**2020/03/05**	**2020/03/06**	**2020/03/07**	**2020/03/09**	**2020/03/09**
[redacted]	再掲載	2019/11/29	2019/11/26	2019/11/27	2019/11/29	2020/03/10
[redacted]	漫画家清書中	**2020/03/07**	**2020/03/08**	**2020/03/09**	**2020/03/11**	**2020/03/11**
[redacted]	再掲載	2020/03/08	2020/03/09	2020/03/10	2020/03/12	2020/03/12
[redacted]	シナリオ完成	2020/03/09	2020/03/10	2020/03/11	2020/03/13	2020/03/13
[redacted]	再掲載	2019/12/06	2019/12/10	2019/12/11	2019/12/13	2020/03/14
[redacted]	再掲載	2020/03/11	2020/03/12	2020/03/13	2020/03/15	2020/03/15
[redacted]	動画作成中	2020/03/12	2020/03/13	2020/03/14	2020/03/16	2020/03/16
[redacted]	再掲載	2019/12/16	2019/12/17	2019/12/18	2019/12/20	2020/03/17
[redacted]	漫画家ラフ依頼	2020/03/14	2020/03/15	2020/03/16	2020/03/18	2020/03/18
[redacted]	漫画家清書中	2020/03/15	2020/03/16	2020/03/17	2020/03/19	2020/03/19

図 6　改善後のスプレッドシート

6. 結果

提案前の 12 月と提案からある程度の時間が経過した現在(3 月)のスケジュール管理に関して, 以下の表 4 で比較した.

表 4 を見ると, 提案前よりも提案後の方がスケジュール管理を上手く出来ていることがわかる. そして, 提案前に期待していたことのほとんどが思惑通りの結果として現れた. また, その他にも, お互いのタスク量を把握し合い作業量の多い人から少ない人へのタスクの受け渡しや, 定例会議で進捗管理だけでなく, 企画などに関しても話し合うことで, コンテンツ自体のクオリティや作業効率も上がったように思える. その理由としては, スケジュールの見通しが以前よりも立っていることや, スケジュールのほとんどが, 計画通りに実行できているため, 計画段階でタイトルやプロット, シナリオなどのイラスト作成前の作業に時間が割けるようになり, 作品自体の完成図がほとんど全て出来上がっている状態で, それぞれのタスクの作業を開始できていることが言える.

今回, プロジェクトを進めていく中で出てきた課題に対しての提案によって, 意図していたことを改善できただけでなく, 意図していなかった, プロジェクトメンバー全員での話し合いの場を設けたことにより, メンバー間でのコミュニケーションが増え, チームの雰囲気も良くなったように感じたことなども, 二次作用として良い方向に改善されたと言える.

表 4 12 月と 3 月の比較

比較内容	12 月	3 月
投稿本数	・冬休みの毎日投稿失敗 ・週 2 本投稿	・3 月(春休み)の毎日投稿が出来そう ・4 月からも週 3 投稿の見通しが立っている
計画	・半月先までしかスケジュール計画が出来ていない	・2 ヶ月先までスケジュール計画が出来ている
計画に対する実行度合い	・40%程度しか予定通り実行出来ていなかった.	・75%程度予定通りに実行出来るようになった.

7. 今後の課題と考察

今回参加したプロジェクトは, プロジェクトマネジメント, プログラムマネジメント, ポートフォリオマネジメントが混在していると言えた. このようなプロジェクトは, 多数の案件を受託しているソフトウェア会社, 研修制作会社, テレビ番組制作会社, イベント運営会社などでも類似した状況が見られる. そのため, 今回の経験は, 多くの事業活動をする上で, 汎用性が高いと言える.

そして, プロジェクトを進めていく中でスケジュールが予定通り進まないという問題があったが, その原因を考え, 基本的な方法で改善することで以前よりもスケジュール管理がよく出来るようになっただけでなく, 基本的な手法の重要性の再確認にもなった. また, それらの改善により, コンテンツ自体のクオリティの上昇や, メンバ間のコミュニケーションが円滑になり, プロジェクトを進めていく上で多くのことがより良く改善された.

しかし, 表 4 を見てわかる通り, まだスケジュール通りには 7 割 5 分程度しか実行できていない. この精度を上げていくには, 動画を運営に必要な本数よりも多く制作することでストックを増やし, プロジェクトに余裕も持たせることや, 図 2 のシナリオ作成を省き, プロットから漫画制作をするといったような動画制作工程を簡略化することなどが必要だと考えた.

また, 立案段階で今回の経験を活かしたプロジェクト計画をすることができれば, より計画に近い状態でプロジェクトを進めていくことができるだろう.

今後の課題としては, 1つのチャンネルの管理だけでなく, 複数のチャンネルで, 多くの動画制作が必要とされる YouTube 事業のプロジェクト全体を管理できるようなプロジェクト管理手法の模索が挙げられる.

参考文献

[1] 「国内動画広告の市場調査」サイバーエージェント/シード・プランニング調べ
(https://www.cyberagent.co.jp/news/detail/id=24125)
[2] タスク管理ツール Trello (https://trello.com/)

リスク構造を読み解いてアプローチする FRI(Factor-Risk-Influence)モデルによるリスク構造の見える化

安達 賢二
株式会社 HBA
adachi@hba.co.jp

要旨

IT 関連プロジェクトのリスクマネジメントでは、メンバーのリスクのとらえ方がバラバラ、重要度と対応優先度の同一視、特定要員だけが対応しているなど、多くの問題が存在し、思うような効果が得られていないケースが散見される。
これらの問題を解決するために、リスク要因ーリスクー影響の関係性を全体で構造化してアプローチする手法を提案する。当事例では、アプローチの詳細と期待効果、および適用上の注意事項を述べる。

1. はじめに

IT 関連プロジェクトのリスクマネジメントでは、以下の状況が散見される。

表 1.プロジェクトリスクマネジメントの問題点

No.	問題点	その結果
1	リスクに関する用語の使い方が人によりばらばら。 例:トリガ、リスク要因、リスク、影響、ペリル、ハザード、など。	話が通じにくく、相互認識共有しにくい。対処が遅れる原因になることも。
2	多くの場合、リスクと影響度と発生可能性を列挙したリスク管理表を活用して運営している。	一つ一つのリスクを個別処理している。
3	リスク管理表(→表 2)で表現されたリスク値と優先度を同一視してしまうケースが多い。 →実際にはリスクの重要度を決める＋対応規模を決める際の参考値に活用するもの。	リスク値が大きいものばかりに目が行きがち。一見小さなリスクを見逃し、あとで大きな問題になることも。
4	管理表に掲載されると暗黙のうちにその時点ですべて対応することになってしまう。	空欄のままにできないため、形式的にでも埋めようとしてしまう。
5	結果として面倒で効果実感が得られにくいリスク管理は、特定の要員(リーダなど)だけが実践することが多い。	他のメンバーは普段はなるべく係わらずに形式対応に終始することが多い。

この状況を打開するための一案として、リスク要因ーリスクー影響の構造分析によるアプローチを提案する。

表 2.リスク管理表(例)

優先	リスク	発生確率	影響度	リスク値	対処
1	○○○○○	5	5	25	
2	□□□□□	5	3	15	
3	△△△△△	3	3	15	
4	◇◇◇◇◇	3	1	3	

2. FRI モデルによるアプローチと効果

2.1. FRI モデルによるアプローチ

表 1 の問題を解決するために、FRI モデルによるアプローチを提案する。
FRI とは、Factor-Risk-Influence の略で、リスク要因ーリスクー影響の構造分析結果(モデル)によるアプローチである。
プロジェクトにおいては、以下の手順で FRI モデルを構築し、リスクを評価して優先度と対処を決める。

0.リスク関連用語を統一する。
1.関係者でリスク要因を洗い出す。
2.それぞれのリスク要因からリスクと影響を導出する。
3.それぞれの要素を時間軸上に配置する。
4.要素間の関係性を分析にする。(FRI モデル)
5.構造の意味・価値を評価する。
6.守るべきものの優先事項を決める。
7.対処すべき対象を選択する。

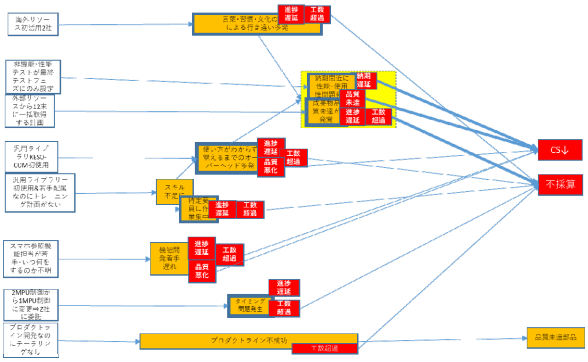


図 1.FRI モデル(例)
白箱=リスク要因、橙箱=リスク、赤箱=影響

2.2. FRI モデルによるアプローチの期待効果

FRI モデルによりアプローチした場合の期待効果を下表に示す。

表 3.FRI モデルによるアプローチの期待効果

No.	FRI モデルによるアプローチ	期待効果
1	リスク関連用語をプロジェクトで統一する。 例:リスク要因-リスク-影響	話が通じやすく、相互認識共有しやすくなる。
2	FRI モデルにて評価し、優先度設定ポリシーから対処の優先度を決める。	重要度を考慮しつつ、現時点から見た優先度で判断できる。 単品処理ではなく、全体の関係性(構造図のパス上のリスク要因ーリスク群ー影響群)を見て必要な粒度とリスクシナリオから対処すべきリスク(要因)を特定できる。 必ずしもその時点のすべてのリスク(要因)に対処しなくてもよくなる
3	以上の結果、リスク対応を関係者と一緒に議論し、実践できる基盤を作りやすい。	メンバーと一緒に議論できる。 -よりよい対処案が出やすくなる。 -合意形成が容易になる。
4	これらの過程をメンバー全員で実践する。	プロジェクトリスクの捉え方、対処の仕方を共に学ぶ(スキルをアップする)場にできる。

2.3. 適用上の注意事項

今回の事例はとあるプロジェクトのプロジェクト計画立案時点でのものである。そこから想定される適用上の注意は以下である。

・リスク管理表と FRI モデルの特徴を整理すると、お互いに長所・短所を補完する関係性があることがわかる。よって実践時には両方を活用していくのが理想である。

表 4.リスク管理表と FRI モデルの特徴

	リスク管理表	FRI モデル
強み	個別要素の詳細を把握しやすい	全体像と関係性により状況把握と判断ができる
	要素を網羅的に管理できる	対策を共通化しやすい
		重要性・優先度の両面を総合して判断できる
弱み	全体像と関係性把握がしにくい	個別要素の詳細把握がしにくい
	個別、単品管理になりやすい	どちらかといえばピンポイント管理になりがち
	重要度(リスク値)と優先度が同一化しやすい	

・リスク管理表で管理することに慣れているため、当手法に切り替えるための手間が最初の障壁に駆る可能性がある。(当手法に限らず新手法を適用する際は同様)
・見た目の印象で FRI モデル化に目が向きがちであるが、実際にはリスク要因の洗い出し、リスク、影響の導出が成否を決める。
・当初の実践はできたとしても、プロジェクトが進行する過

程の状況変化を FRI モデルにリアルに更新し、継続してリスクを監視するタスクを手抜きなくやり切る必要がある。

参考文献

[1] プロジェクトリスクの表現に関する研究 木野 泰伸
[2] リスクをめぐる基本用語について 田村 祐一郎
[3] SS2018 札幌事例発表「リスク構造化を用いたリスクマネジメント手法の提案と効果分析～「未来予想図」を用いたリスクマネジメント PDCA サイクル～」
水野昇幸 安達賢二

ケースメソッドを適用したプロジェクト・マネージャー育成の取り組み

木村 良一
産業技術大学院大学
a1815rk@aiit.ac.jp

要旨

本稿では，専門職大学院である東京都立産業技術大学院大学（以降，「本学」とする）における，模擬シナリオを用いたケースメソッドを適用したプロジェクト・マネージャー育成の取り組みについて報告する．

1. はじめに

変化の激しい経営環境において，プロジェクトが成功するためには，プロジェクト・マネジメントの知識だけでは不十分である．本学では PM の初級教育にケースメソッドを取り入れ，気づきを中心にした PM 教育を行っている．

2. 一般的なケースメソッド

高木・竹内によると，ケースメソッドを用いた学習のポイントは以下に整理される．

[1] ケース教材：企業で実際に発生した出来事を切り取りケース教材とする．
[2] 授業内容：ケース教材について，個人による予習を行い，グループ討議を経て全体でのクラス討議を行う．クラス討議では、個人の気づきをもとに，学習者同士が議論することにより，新たな考えに触れ，自身の気づきを深める．
[3] 講師の役割：講師はディスカッションリーダーシップをとることでクラスの議論が有益な展開になるように論点の流れの舵をとる．

3. プロジェクト・マネージャー育成の取り組み

本学では，1 年次にて基本的な知識・スキルを学修し，2 年次にて PBL 型教育で業務遂行能力を獲得する．ケースメソッドを用いた授業は PBL 型教育の一環として，初級者 PM を対象に実施している．

以下にそのポイントについて示す．

3.1. シナリオ型ケースメソッドによる全体像の把握

学習者は立ち上げから終結までの一貫した模擬シナリオをもとに作成したケースで学習する．この「シナリオ型ケースメソッド」により，学習者はプロジェクトの全体像を俯瞰することができるとともに，個々の局面におけるプロジェクト・マネジメントの勘所を理解することができる．

3.2. 2段階の「気づき」が得られる構成

授業の構成は，少人数の PBL 型教育のため，個人による予習とグループ学習の 2 段階としている．

個人による予習では，事前テストにより該当フェーズの学習に必要な知識を整理した後にケースの事前学習を行うことで，学習者個人としての「気づき」（内発的気づき）を得る．グループ学習では，学習者全員の参加により，事前テストについての知見の共有，議論を行った後，ケースに基づく議論を行い，グループとしての「気づき」（外発的気づき）を得る．さらに，復習課題の議論により獲得した「気づき」を深めていく．

3.3. 学習者自身によるディスカッションリード

授業の目的は，プロジェクト・マネージャーの育成である．授業そのものも一つのプロジェクトとして捉え，原則として学習者自身によるファシリテーションで運営する．

4. おわりに

ケースメソッドを適用したプロジェクト・マネジメントの学習は知識や技術の獲得ではなく，学習者自身の思考特性や行動特性の変容に効果的であると考えており，今後．中級，上級者への教育についても考えていきたい．

参考文献

[1] 高木晴夫，竹内伸一，ケースメソッド教授法入門，慶應義塾大学出版会，2010
[2] 高木晴夫，組織マネジメント戦略，有斐閣，2005

QA チームによる顧客要求の抽出・整理を支援する仕組みの構築

柏倉 直樹
株式会社 ディー・エヌ・エー
naoki. kashiwagura@dena. com

要旨

エンターテイメントサービスにおいては利用時の品質のうち快感性や快適性を高めることが重要である. 分業を前提とした開発の場合, 快感性や快適性を高めるためにマーケティングチームなどの専門チームが顧客要求の獲得を担当する事が多い. 一方でアジャイル開発などの非分業を前提とした開発の場合, プロダクトオーナーと QA が密接に協力して開発を進めるというプラクティスが提唱されるなど, これまでの QA の役割を超えてチームに貢献することが求められており, 要求の獲得にも QA が貢献する必要性が生じてきていると考えた. そこで, 顧客要求の獲得に不慣れな QA 担当者でもフレームワークに当てはめることで顧客要求を獲得でき, 要件として取りまとめ, 優先順位の提案まで含めてプロダクトオーナーへインプットする仕組みの構築に取り組んだ. なお今回の取り組みでは, QA 組織はエンドユーザーからのお問い合わせ情報にアクセスしやすいという特性を活かし, お問い合わせ情報をインプットとする仕組みを構築した.

1. はじめに

本稿はエンターテイメントサービスの一つであるライブ配信サービスにおける利用者の「期待」を VOC (Voice of Customer: 一般的には顧客の声全般を指すが, 本稿ではカスタマーサービスに寄せられるご意見・ご要望を指す) を元に抽出し, 快感性や快適性の向上に活かす仕組みを構築する取り組みについて述べたものである. ここでいう「期待」は当たり前品質[1]だけでなく快感性や快適性, UX (User Experience) [2]の向上につながる可能性のある顧客要求を指す.

ライブ配信サービスとはスマートフォンなどの通信機器を用いてライブ動画の配信・視聴を行うエンターテイメントサービスであり, サービスの利用者には配信者と視聴者がおり, それぞれがリアルタイムでコミュニケーションし合っている. 利用者の満足度を高めるためには, 配信者と視聴者それぞれのサービスに対する期待を把握し, 優先順位をつけて対応する必要がある.

期待の獲得は, 分業を前提とした開発の場合はマーケティングチームなどの専門チームが担当することが多い. 一方でアジャイル開発などの非分業を前提とした開発の場合はプロダクトオーナー (以降, PO と略す) と QA が密接に協力して開発を進めるというプラクティスが提唱される[3]など, これまでの QA の役割を超えてチームに貢献することが求められており, 期待の獲得にも QA が貢献する必要性が生じてきていると考えられる.

しかし, 筆者が所属する組織はこれまで分業型の開発スタイルをとってきており, QA チームは期待の獲得に不慣れであった. このため, QA チームによる期待の獲得を支援する仕組みの構築が必要である.

ここで, QA チームは VOC にアクセスしやすいという特性を持っている. VOC として届く利用者の声は多くが要望 (不満を解消するために利用者がサービスまたはサービスを提供するシステムに求めているものであり, 当たり前品質を満たすための顧客要求) と不満 (こうしてほしい等の提案を含まずサービスの満足できない点だけを述べたもの. 例えば「画面が見辛い」など) であり, 一見するとネガティブに捉えてしまい是正に走りがちである. しかし裏返せばサービスの今後の発展を楽しみにする気持ちの表れという側面をもっており, 多くの期待を抽出できる可能性を秘めている.

以上のことから本稿では, VOC をフレームワークに当てはめることで不満や要望の裏に隠れた利用者の期待を抽出し, 快感性や快適性向上に活かす仕組みの構築を試みる. 以降 2 章では議論の背景理解を容易にするためにライブ配信サービスの概要と性質について示しつつ解決すべき課題を述べる. 3 章で課題の解決方針と具体的な解決策を示し, 4 章で実証を試みる. 5 章でまとめと今後の課題を述べる.

2. 課題設定

2.1. 現状分析

ライブ配信サービスのステークホルダーは配信者，視聴者，ライブ配信事業者（以降，事業者）の 3 者である．ライブ配信サービスとは，PC やスマートフォンから生放送（ライブ）形式で動画等の配信が行えるものであり，視聴者と配信者との間でリアルタイムでのやりとりが行えるなどの特徴がある[4].

3 者の関係を図 1 に示す．配信者はライブ配信サービスを利用してライブ動画を配信する．ライブ配信サービスは複数の配信スペースで構成されており，原則として 1 つの配信スペースに 1 人の配信者が存在し，視聴者は 1 つの配信スペースに複数名参加できる．

視聴者はライブ配信サービスを利用してライブ動画を視聴し，リアルタイムでコメント投稿やアイテム送信を行う．視聴者が投稿したコメントやアイテムは，配信者と他の視聴者全員が閲覧することができる仕組みになっているため，視聴者と配信者間のコミュニケーションはもちろん，視聴者どうしのコミュニケーションもリアルタイムにとれる．

事業者は配信スペースを盛り上げるためにアイテム（例えば拍手アニメーションなど）を提供する．このアイテムと交換できる有料ポイントを視聴者に購入してもらうことで事業者は収益を上げる（売り上げの一部は協力費として配信者へ分配される）．

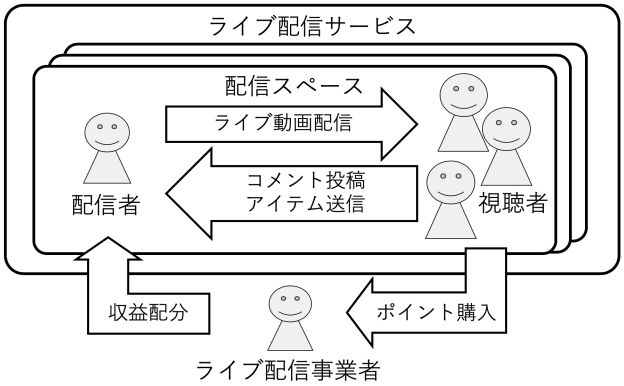


図 1 ステークホルダーの関係図

ステークホルダーごとの動機を表 1 に示す．ここで動機とは，なぜサービスを利用するのかという理由や目的のことである．

事業者はライブ配信を通じて配信者と視聴者を喜ばせ，楽しんでもらうことで収益を得ている．例えば配信者と視聴者を盛り上げるための施策の一例として事業者は様々なイベント（配信者同士で配信の盛り上がり度を競い合うなど）を開催する．イベントで条件をクリアした利用者には景品として TVCM 出演権・デジタルサイネージ出演権・雑誌出演権などの各種出演権や，日常生活で利用する各種家電，食料品など様々なものが贈られる．

配信者は視聴者とのコミュニケーションを楽しむため，あるいは有名になるため，あるいは収入を得るためなど様々な理由でライブ配信サービスを利用する．

視聴者は配信者との会話や視聴者同士の会話を楽しむため，あるいは配信者を応援するため，配信者のパフォーマンスを楽しむためなど様々な理由でライブ配信サービスを利用する．

表 1 ステークホルダーごとの動機

ライブ配信事業者	• サービスを通じて配信者・視聴者を楽しませる • 利用者が夢を実現するための場を提供する • 有料ポイントを購入してもらい収益を上げる • 利用者に楽しんでもらいサービス提供者としての喜びを得る
配信者	• 視聴者から応援してもらうことで承認欲求を満たす • TVCM 出演や雑誌掲載などの報酬を通じて有名になる • 報酬を目的とせず友達とおしゃべりする感覚で楽しむ • 分配金による収入を得る
視聴者	• 応援に対するお礼を言われることで応援欲求を満たす • 配信者の上位入賞に貢献する • 同じ趣味を持つ利用者が集まりコミュニケーションを楽しむ • スポーツチームを応援するような感覚で，視聴者同志でライブ配信を盛り上げて楽しむ

各ステークホルダーの動機を継続させるために，事業者は配信者・視聴者それぞれのサービスに対する満足度を向上させる必要がある．もし配信者の満足度が低く活発に動画を配信してもらえない場合は，視聴者は楽しい配信に巡り会う機会が減ってしまう．その結果視聴者

の満足度も下がり，利用者の離脱に繋がる．そして配信者は視聴してもらう機会が減るとモチベーションが低下し，離脱してしまうという負の連鎖になる．

サービスの改善のために事業者は VOC を活用している．カスタマーサービスにて VOC を集計し，担当者の知見から利用者が求めていることを読み取った場合はその内容を添えて PO へレポートしている．

図 2 に示すように VOC はその多くが要望と不満である．一見すると VOC は要望と不満の塊であるが，見方を変えればこれらは利用者がサービスの今後の発展を楽しみにする気持ちの現れと捉えることができる．なぜなら利用者はどうでも良いサービスのためにわざわざ意見を届けようとは思わないためである．

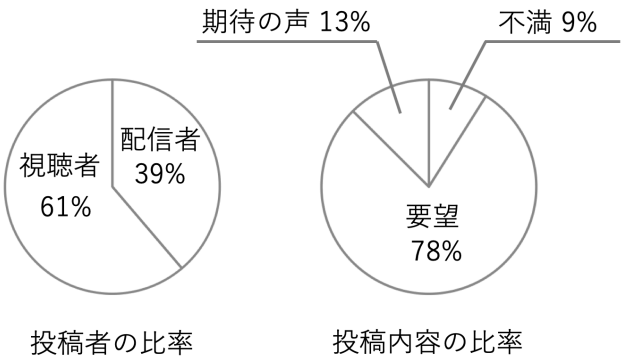


図 2 VOC の内訳

実際に VOC を確認したところ，一見すると不満の言葉ではあるがその裏側に利用者のサービスに対する期待を感じ取ることができるものが見つかった．そこで本稿では VOC に着目し，VOC の裏側に隠れた利用者の期待を取り出し，サービスの改善に活用することに取り組む．

要望に対応することも当たり前品質を確保するために重要なことである．しかし，利用者にさらなる楽しさを届け，満足度の向上につなげるためには要望に応えるだけでは十分ではない．

非分業を前提とする開発において QA チームが多忙な PO をサポートし顧客満足度の向上に貢献するために，VOC をフレームワークに当てはめることで不満や要望の裏に隠れた利用者の期待を抽出し，PO にインプットする仕組みを構築する必要がある．

2.2. 先行事例

本稿での課題を先行事例で解決できるか調査を実施した．以下に調査結果を示す．

内山ら[5]は，テーマに関してフリーフォーマットで自由に意見を述べる形式で取得されたアンケート結果から「～てほしい」と言い換えが可能な文を要求文として取り出し，さらにその要求文を「～について～てほしい」と言い換えることで要求内容を同定する方法を提案している．この手法はパターンに当てはめて言い換えを試みることで，言い換えの実施者に依存せず高い精度で要求内容を同定することができるが，なぜその要求が発生したのかという背景事情までは考慮していないため，期待に置き換えることは難しい．

京屋ら[6]は，VOC から真の顧客要求を抽出することこそが製品開発を成功させる重要なポイントであるとし，LIVEVOICETM を用いて VOC から要求品質を抽出し，整理することで顧客要求全体の構造を把握することができると述べている．しかし VOC から要求品質を抽出するプロセスの内容については触れておらず，VOC の裏側を読み取っているかについては言及されていない．

以上の結果から，先行事例では本稿の課題を解決することは出来ないと判断し，VOC から期待を抽出する方法を検討することとした．

3. 解決策

3.1. 課題の解決方針

開発サイクルは，まとまったアップデートであれば基本的に 2 週間サイクル，軽微なアップデートはデイリーとなっている．情報のフィードバックをこのサイクルに合わせ，継続してフィードバックし続けるためには，改めて利用者からデータを収集する手間を省き，誰でもわかる手順を検討する必要がある．また，従来のフィードバック方法も継続することで要望と期待の両面から多角的に施策の優先順位を決定できるようにする．

3.2. 解決策

期待抽出プロセスを構築し，VOC の裏に隠れた利用者の期待を抽出してサービスの改善に活用できるようにする．図 3 は期待抽出プロセスを導入後の，顧客の声のフィードバックの仕組みを表している．期待抽出プロセスを導入することで PO はより多くの情報から対策を検討可能になる．図 4 は期待抽出プロセスの全容である．以降で期待抽出プロセスの各ステップの詳細を述べる．

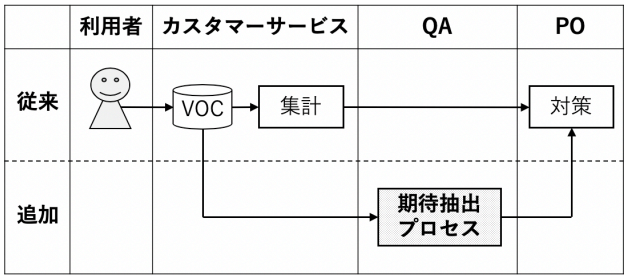


図 3 顧客の声のフィードバックの仕組み

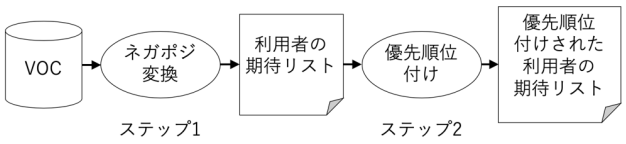


図 4 期待抽出プロセス

ステップ 1:ネガポジ変換

期待は, それを利用者自身が自覚している場合は期待 VOC となるが, 利用者自身も気が付いていない場合は要望または不満の VOC の中に隠れているため抽出する必要がある. 本稿では要望または不満から期待を抽出し, その期待を満足させるための要件を導き出すプロセスをネガポジ変換と定義する. ネガポジ変換は図 5 に示すネガポジ変換チャートに沿って進める.

ポジ対策(図 5)とは, 分析した WHY(背景やユースケース, 利用者の心情などを考慮して, その意見が生じたと考えられる理由)および WANT(WHY から推察される VOC の裏に隠れた利用者の期待)から, 可能な限り根本的な対策となるものである. 要望または不満の VOC で, その通りの対策以外を探索して見つけるようにするとよい.

ネガ対策(図 5)とは, 要望または不満の VOC そのままの解釈を基にした直接的な対策となるものである. ポジ対策がネガ対策と異なる差分があることを確認することで, ポジ対策がより根本的な対策となっていることを評価するためにネガ対策を作成する. 要望または不満の VOC で, その通りの対策を当てはめるようにするとよい. 例えば追加／削除してほしいという要望, または不足／余計であると訴える不満 VOC に対して, 直接対応する機能の追加／削除を対策とする.

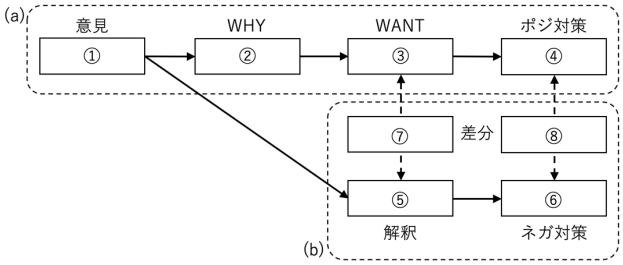


図 5 ネガポジ変換チャート

はじめに図 5 の破線(a)で囲んだ部分を記入する.

① 意見欄に利用者の要望や不満を VOC から一つ選んで記入する.
② 背景や利用者の心情などを考慮して, 意見が生じた理由を WHY 欄に記入する.
③ WHY から推察される VOC の裏に隠れた利用者の期待を WANT 欄に記入する.
④ WANT を満たすための対策案をポジ対策欄に記入する.

次に, (a)で導き出した WANT とポジ対策の妥当性をチェックするために, 破線(b)で囲んだ部分を記入する.

⑤ 解釈欄に意見を直接的に解釈した内容を記入する.
⑥ ネガ対策欄に解釈にもとづいた直接的な対策内容を記入する.
⑦ 差分欄に解釈と WANT の差分を記入する.
⑧ 差分欄にネガ対策とポジ対策の差分を記入する.

ここで, もし差分が明確に記入できない場合は WANT, ポジ対策と思っているものが実はまだ解釈, ネガ対策である可能性がある. ただし意見欄に記入した内容がそもそも利用者の期待に該当する場合には差分は発生しない.

実際にネガポジ変換で期待を抽出した例を表 2, 表 3 に示す. (表中の①～⑧はネガポジ変換チャート(図 5)と対応) 単純に意見を解釈した場合とネガポジ変換を適用した場合の対比をわかりやすくするため, 表の前半に意見・解釈・ネガ対策を示し, 表の後半で WHY・WANT・ポジ対策・差分を示す.

表 2 不満にネガポジ変換を適用した例

意見①	ライブ配信スタート通知がお気に入り登録している配信者全員から届くので通知だらけになってしまい煩わしい
解釈⑤	お気に入り登録は利用者の自由意志であり事業者では対策不可
ネガ対策⑥	デバイスの通知設定を OFF にするようアナウンスする
WHY②	・デバイスの通知設定を OFF にすることでは要望を満たせない ・お気に入り登録を減らしたくないという心情がうかがえる ・ユースケースとして,お付き合いでお気に入り登録している人もいる
WANT(抽出された期待)③	本当に通知が欲しい配信者からのみ通知が届く状態
ポジ対策④	・配信者ごとに通知の ON/OFF 設定する機能の実装 ・お気に入りに段階を設け通知を受ける段階を設定できるようにする
差分⑦	解釈では対策不可と考えているが WANT では機能改善で対策可能である
差分⑧	ネガ対策では本当に通知が欲しい人まで通知が届かなくなってしまう

表 3 要望にネガポジ変換を適用した例

意見①	不快なワードを含むコメントを削除できるようにしてほしい
解釈⑤	不快なワードを含むコメントを削除できるようにすれば良い
ネガ対策⑥	コメントの削除機能の実装
WHY②	気分が害されライブ配信を楽しめない
WANT(抽出された期待)③	不快なワードを含むコメントに遭遇しない状態
ポジ対策④	不快なワードを投稿できなくする
差分⑦	WANT は不快なワードを含むコメントにそもそも遭遇したくない
差分⑧	不快なワードを含むコメントを未然に防止するか事後に削除するか

ステップ 2:優先順位付け

ステップ 1 で抽出した期待に優先順位を付け，優先度の高いものから順に対策を実施することが，より効率的に顧客満足度を向上させるために重要である．優先順位を付けるため，ステップ 1 で抽出した結果を 6 つのカテゴリー「好奇心」「希望感」「挑戦心」「感情移入」「社会性」「非日常」に分類する．この 6 つのカテゴリーはエンターテイメントサービスにおける顧客満足度を考える上で重要な要素である「楽しさを感じる要因」を，特性要因図[7]の表現方法を流用して導出したものである（図 6）．なお，カテゴリーは不変的なものではなく，様々な要因によって変化することが予想されるため，柔軟にアップデートする必要がある．

人間の心理面を考慮して期待を分類することで，利用者の期待は心理的にどの要因に集中しているのか把握し施策検討に活かすことができる．また，特性要因図を作成しておくことで，分類の担当者が複数いる，あるいは交代になった場合でも担当者ごとのバラツキを抑制する効果が期待できる．例えば表 2 に示す例の場合，WHY にある「お付き合い」はコミュニティに関する心理であり「社会性」に分類することができる．

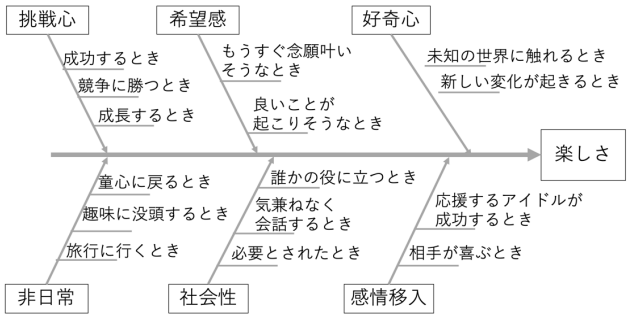


図 6 楽しさの要因を検討した特性要因図

4. 解決策の検証

4.1. 検証方法

ステップ1の有効性を，どの程度の期待を抽出できたか計測することで検証する．もしほとんど期待を抽出できないとすればステップ1の効果は無かったと言うことになる．

ステップ2の有効性を，期待を要因ごとにカテゴリー分けした結果，特定の要因に偏りが見られるかをチェックすることで検証する．もし偏りが見られない場合，心理面を考慮して分類しても優先順位付けすることができず，効果が無かったと言うことになる．

4.2. 検証結果

過去に寄せられた392件のVOCに対しステップ1を適用したところ，70 件の期待を抽出することができた．392件の VOC の内，期待 VOC は 41 件であったことから，ステップ1適用後の 70 件を加算すると 111 件となる．このことから，ステップ1の適用前後で期待が約 170%増という結果が得られた．なお，同一人物からの同一の期待は重複として除外してある．

次に，ステップ2で要因毎にカテゴリー分けした結果を表 4 と図 7 に示す．もっとも注目すべきは視聴者の期待である．ステップ1適用前の期待を分類した結果では希望感が9件でトップ，社会性が5件で3番目という内訳になっている．しかしステップ1適用後の期待を分類した結果では大きく件数が変化し，社会性が 32 件でトップ，希望感は9件のままで3番目となり，ステップ1を適用した場合としない場合とでカテゴリー分けの順位が大きく入れ替わる結果となった．

表 4 期待抽出プロセス適用前後の期待数比較

() 内は適用前の件数

	好奇心	希望感	挑戦心	感情移入	社会性	非日常	総計
配信者	0(0)	3(2)	23(9)	0(0)	9(2)	0(0)	35(13)
視聴者	0(0)	9(9)	0(0)	16(6)	32(5)	0(0)	57(20)
共通	0(0)	3(3)	0(0)	2(2)	14(3)	0(0)	19(8)
総計	0(0)	15(14)	23(9)	18(8)	55(10)	0(0)	111(41)

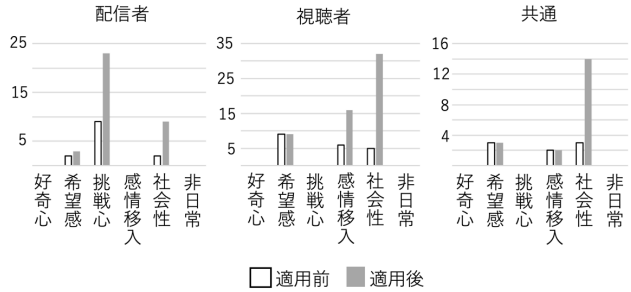


図 7 期待抽出プロセス適用前後の期待数比較グラフ

4.3. 考察

期待抽出プロセスの効果について

過去の VOC に期待抽出プロセスを適用したところ，期待のボリュームが 170%増加し，カテゴリーごとに分類したランキングも順位が入れ替わる結果となったことから， 要望や不満の裏には多くの期待が隠れており，きちんと抽出した上で分類しなければ施策の優先順位を正しく判断できないことがわかった．もし期待抽出プロセスを適用しなかった場合，利用者が最も求めているカテゴリーの対策が後回しになってしまい，優先順位の判断を誤る恐れがあった．

また，期待抽出プロセスを適用することで，利用者がサービスのどのような部分に基本的な価値を見出しているのかを定量的に計測できることがわかった．例えば今回の検証結果では，社会性に関する期待が要望や不満の形で多く寄せられていたことがわかる．不満は当たり前品質が満たされていないときに発生するため，不満が多い部分ほど利用者が当たり前品質として求めていると言える．期待の裏返しとしての不満を計測することで，利用者にとっての当たり前品質，すなわち基本的な価値がどこにあるかを定量的に明らかにできる．

PO 視点の効果としては，時間をかけずにこれまでより多くの情報を施策検討に活用できるようになるというメリットがある．VOC は量が膨大であり，同一人物から同じ内容の意見が複数届くことや，人によって様々な文章表現で届くため内容の解釈にある程度の時間を要することがあるなど，そのままでは扱いにくい傾向がある．QA が VOC から期待を抽出し整理する役割を担うことで，PO の負担を増やさずにより多くの情報から施策検討できるようになる．

VOC の偏りについて

第 2 章で述べたように VOC の利用者ごとの内訳は配信者が 39%，視聴者が 61%となっている．実際の配信者と視聴者の人数比率から考えると，視聴者の VOC が少ないと言える．コンテンツ提供者である配信者から VOC が多く寄せられるのは当然の傾向であるが，一方で視聴者は事業者の収入源であるため非常に重視しなければならない対象である．今後は VOC のバランスを考慮した方法を検討する必要がある．

期待抽出プロセスの汎用性について

本プロセスはチャートに当てはめるだけでライトに使用できるプロセスであり，ライブ配信サービスの VOC 分析以外にも様々な用途で活用できる汎用性がある．以下にその一例を示す．

(1)エンターテイメントに限らず様々なドメインでの VOC 分析
(2)アンケートやインタビュー結果の分析
(3)要求分析などの設計工程での活用
(4)人材のトレーニング教材
(5)議論のベース

戦略的な楽しさの構築について

図 7 で示したように配信者と視聴者では楽しさの要因が明らかに異なっている．これは，特性要因図を活用することで利用者ごとの楽しさを感じる特徴を定量的かつ定性的に把握できたことを示している．これを応用すれば，サービスの特徴を把握して適切な施策を打つことで，特性要因の組成を戦略的にコントロールし，利用者に飽きられずビジネス成長を図ることが期待できる．

5. まとめ

5.1. 結論

サービスの満足度向上のため，利用者の期待把握と優先順位付けを支援するための枠組みである期待抽出プロセスを提案した．期待抽出プロセスは VOC から期待を抽出するネガポジ変換と，特性要因図から導き出した「楽しさを感じる要因」にカテゴライズして，利用者の心理面を考慮して期待の優先順位付けを行うという 2 ステップで構成される．

過去に寄せられた VOC に対しネガポジ変換を適用したところ，利用者の期待をネガポジ変換適用前より 170%多く抽出できた．

ネガポジ変換で期待を抽出した後に「楽しさを感じる要因」ごとに期待を分類した結果，要因ごとに件数の偏りが見られたことから，期待を分類することで優先順位付けできることが確認できた．また，ネガポジ変換により期待を抽出した場合とそうでない場合とでは分類結果の偏りに違いが見られたことから，ネガポジ変換で期待を抽出した後に分類する必要があることがわかった．

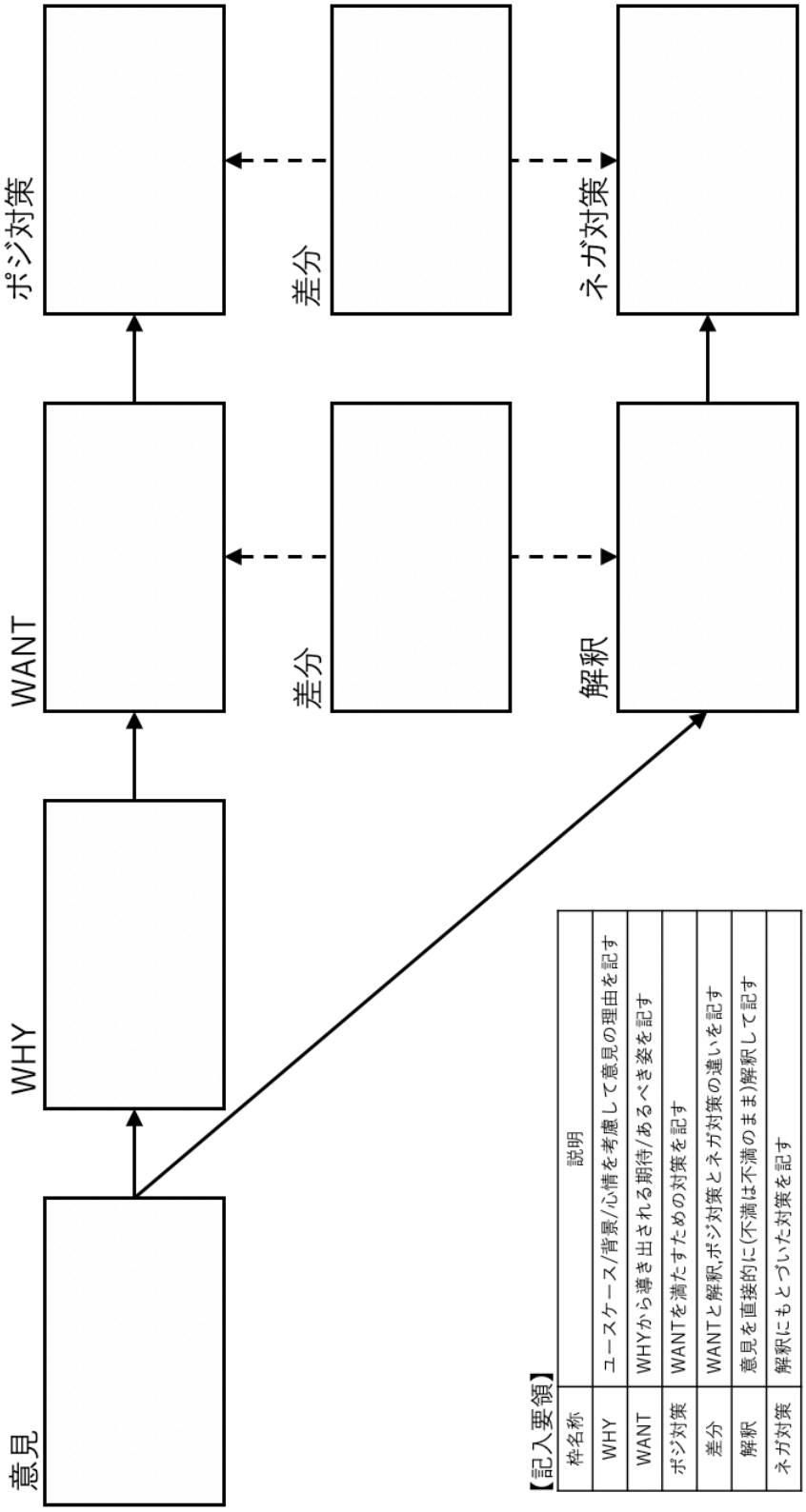
5.2. 今後の課題

今回は顧客満足度を向上させるための施策検討までのプロセスを考案した．しかしこのプロセスを用いることで実際に顧客満足が向上するかまでは検証できていない．エンターテイメントサービスにおいては利用者がいかに楽しんでいるかが重要であり，利用者がどの程度サービスを楽しんでいるかを定量的に計測することでサービスの満足度を定量評価することができる．この"利用者がどの程度サービスを楽しんでいるか"という度合いをワクワク度と定義し，ワクワク度を計測するための KPI を見つけ，サービスの満足度を定量的に評価する仕組みを構築すること，そして期待抽出プロセスでサービスの満足度が向上するか検証することを今後の課題とする．

参考文献

[1] 狩野 他，“魅力的品質と当たり前品質”，品質，14 巻，2 号，pp. 39-48(1984).
[2] 安藤，“UX デザインの教科書”，丸善出版(2016).
[3] 永田，“なんたって”DevQA”アジャイル開発とQAの合体が改善を生む”，スクラム冬の陣 2017 ～みんなで学ぶスクラム～(2017).
[4] 消費者庁，“資料 1 ライブ配信サービス（投げ銭等）の動向整理”，インターネット消費者取引連絡会，第 31 回，p. 3(2018).
[5] 内山 他，“自由回答アンケートにおける要求内容の分析”，言語処理学会年次大会発表論文集，第 10 回(2004).
[6] 京屋 他，”顧客の声を起点にした商品企画プロセス”，東芝レビュー，60 巻，1 号，pp. 36-39(2005).
[7] 石川 他，“品質管理入門”，日科技連出版社(1956).

付録　ネガポジ変換チャート(テンプレート)



【記入要領】

枠名称	説明
WHY	ユースケース/背景/心情を考慮して意見の理由を記す
WANT	WHYから導き出される期待/あるべき姿を記す
ポジ対策	WANTを満たすための対策を記す
差分	WANTと解釈,ポジ対策とネガ対策の違いを記す
解釈	意見を直接的に(不満は不満のまま)解釈して記す
ネガ対策	解釈にもとづいた対策を記す

プロダクトマネージャーが求めるアジャイル開発における ソフトウェアテストのあり方と生産性向上への取り組み

坂東　塁
株式会社リクルートライフスタイル
rui_bando@r.recruit.co.jp

羽鳥　温子
株式会社 ProVision
a-hatori@pro-vision.jp

要旨

大規模なサービスを過去 10 年以上にわたりウォーターフォールでプロダクト開発を進めていた我々の現場において，生産性の向上を目的として開発体制の一部でアジャイル開発を採用し，リーンアプローチを続けていた．しかし，その中でも小さいウォーターフォールのような体系は残ることが多く，プロダクトマネージャーから始まり，エンジニア，QA の各メンバー同士の役割も大きく変化することはなかったため，結果として生産性向上と逆効果となる可能性さえあった．

そこで，これまでチーム内での役割が専門スキルごとに明確に分担されていた部分に着目し，QA は固定化されたタイムボックスで動くのではなく，コミュニケーションをとりつつ全体の進捗に応じて緩やかな連携を行うことで，「生産性の向上」を図った．企画段階からプロダクトマネージャーと連携し，仕様の理解やフィードバックなど上流工程での品質の担保などへの取り組みを進めていた．このような取り組みの結果，役割の変化により，QA メンバーの意識の変化に繋がり，テストフェーズの効率化により，企画・仕様の手戻りの防止にも繋がった．それだけではなく，プロジェクトにおける”テスト”だけという役割で，プロジェクトへ関われきれない想いもあった中で，案件に上流工程から関わる中，「プロダクト」における「品質意識」の向上にも繋がった．

しかし，「品質への意識」改善はされたものの「生産性」をいう観点ではチームとして期待するような改善が見られず，チームとしての課題を改めて見直す必要がでてきた．これまでの取り組みでは，主にシフトレフトを中心としたアプローチにおいて，QA メンバーの企画・仕様部分への染み出しが多く，実装・テスト部分での QA メンバーの役割については従来どおりであった．

今回，新たなチーム作りにおいては，チームに残る課題を整理する中で，生産性向上を目指すために“QA スキーム”における「品質」と「テスト」の違いに焦点を当てた．そして，課題をチーム内で整理する中で出てきた「不具合の増加による生産性の未改善」「プロダクトに対する品質意識のズレ」という我々の課題に向き合うことで，チームが目指す生産性の向上を実現させることを目的とした．

我々は，品質を担保する（ソフトウェアテスト）一連の工程を”QA スキーム”と定義し，従来の“QA スキーム”では，「品質」の担保が QA メンバーにおけるテストのみに依存されていた．その中で，「品質」は QA メンバーがリードするものではあるが，QA メンバーのみが担保するものではないという考えからチーム全体で「品質担保」を行いそれぞれが新たな役割を担う“QA スキーム”を再定義した．これまで置き去りにされていた実装フェーズ以降の”QA スキーム”の見直し，「テスト」におけるチームの関わり方の再考，そして「品質」と「テスト」の違いを明確にすることで，改めて生産性へのアプローチを見直した．

結果，「品質」を QA メンバーの「テスト」で担保していたスキームからチームでの「品質」の担保に取り組むことで，チームの意識自体の変化にも繋がった．サイロ化されたQA 意識から脱却することにより，チームとして「品質担保」を行った．その取り組みにより，昨対比による不具合削減にも繋がり，手戻り現象に加え，不具合見逃し等のリスクも軽減されたことで，当初求めていた「生産性の向上」が少しずつ効果として現れてきている．

本発表では，以上の取り組みの中でソフトウェアテストの中心を担う QA メンバーの役割をどのように再定義したのか，チームメンバーと新たな取り組みに対してデメリットやリスクを会話する中で，生産性向上に向けて最適な解を出せるように取り組んだ事例を紹介する．

ネットワーク型データ構造による SW 部品の関係の可視化 ~SW 部品選択におけるグラフ DB の適用と評価~

川井　隆之
株式会社デンソー
takayuki.kawai.j4x@jp.denso.com

小川　雄太
株式会社デンソー
yuta.ogawa.j2t@jp.denso.com

水藤　倫彰
株式会社デンソー
tomoaki.suito.j4s@jp.denso.com

要旨

排気センサーシステムは，複数の車両メーカーの様々な顧客要求に対して，ソフトウェア（SW）部品を組み合わせて対応している．SW 部品は多岐に渡り，対応する仕様や部品間の共有関係や選択時の制約が複雑に絡むため，熟練者が最適な部品を選択し一覧表で管理している．しかし，選択した SW 部品の妥当性を一覧表だけで確認することは非常に困難である．

本報告では，SW 部品の関係を，ネットワーク型データ構造を持つグラフ DB で表現し，部品選択結果の確認に必要な情報を可視化した．さらに，グラフ DB による SW 部品関係の可視化は，部品選択以外の開発プロセスにおいても非常に有効に機能することを示した．

1. はじめに

我々は，複数顧客の SW 要求を分析，仕様化した機能要求セットを用意しておき，そこからリリース先顧客に合わせて必要な仕様を選択する．そして，各仕様に対応した SW 部品を選択し組み合わせて SW を構成する．

SW 部品の選択は熟練者が実施し一覧表としている．SW 部品は多岐にわたり，共有関係や選択制約が絡むことで，仕様と部品の関係は複雑になっている．そのため，熟練者以外では SW 部品の組み合わせの妥当性を確認することが困難となっている．

2. アプローチ

従来の表形式では個々の 1:n の関係しか表示できないため，SW 部品の複雑な関係を示すことは難しい．そこで，グラフ DB[1]を適用し，SW 部品の関係をネットワーク型のデータ構造で表現し，検索により可視化する．

今回は実験用に定義した，仕様と SW 部品の関係を Neo4j[1]を用いてグラフ DB 化し，検索によりそれらの対応関係を可視化した．

3. 実施結果

Neo4j を使って仕様と SW 部品の関係を出力した画面を図 1 に示す．ネットワーク型データ構造で SW 部品の全ての情報を記述することにより，必要な観点から仕様と部品の関係を抽出し，グラフとして容易に表示できることを確認した．

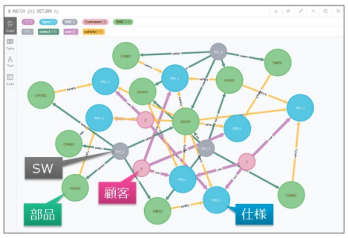


図 1 仕様と SW 部品の関係の全体像

4. 考察

グラフ DB は，必要な観点からの関係を容易に可視化できるため SW 部品の組み合わせの妥当性確認において有効である．また，グラフ表現は直感的にわかりやすく，ブラウザ上で自由に操作できることも利点である．

本手法は SW 部品選択の妥当性確認以外にも有効である．例えば，不具合部品の使用先顧客の特定や影響が出る仕様の特定が迅速かつ確実に行える．

5. まとめ

グラフ DB を用いて SW 部品の関係をネットワーク型データ構造で表現し可視化した．本手法は，SW 部品選択の妥当性確認に有効であることを確認した．また，不具合解析などにおいても活用できると考えられる．

参考文献

[1] Ian Robinson，et al.，Graph Databases，O'Reilly Media，2015

産学官連携による岩手県域における「震災復興支援家族ロボット教室」

新井 義和
岩手県立大学
arai@iwate-pu.ac.jp

高田　亨
岩手県商工労働観光部
t-takada@pref.iwate.jp

秋田 敏宏
一関工業高等専門学校
akita@ichinoseki.ac.jp

今井 信太郎
岩手県立大学
s-imai@iwate-pu.ac.jp

冨手 壮一
岩手県工業技術センター
tomite@pref.iwate.jp

江口 かおる
有限会社イケハウス
eguchi.kaoru@gmail.com

要旨

東日本大震災を機に岩手県域を巡回する形で開始された「震災復興支援家族ロボット教室」の *8* 年間の活動を振り返り，産官学連携の活動事例として報告する．

1. はじめに

2011 年 3 月の東日本大震災を受けて「自分達らしい復興支援ができないか」との機運が高まった富士通コンピュータテクノロジーズ（以下，FCT）の呼びかけに岩手県関係者が呼応し，同年 12 月から岩手県域において小学生向けの「震災復興支援家族ロボット教室」の活動が開始された．本報告では，8 年間の活動を経て，新体制による再出発へ向けて同活動を振り返る．

2. 実施体制

FCT が主催となり，教室の機材一式を提供し，進行役ならびに講師を務めるとともに，トレーナー育成を担当した．そこに，岩手県立大学および一関工業高等専門学校がトレーナーとして参加し，参加家族 1 組に対して 1 名のトレーナーが担当する手厚い指導体制を構築した．開催地の選定および参加者の募集においては，岩手県職員が各自治体との調整ならびに同自治体の教育委員会へ協力要請を行うことによって，県内各地でより多くのご家族の参加が得られる体制とした．

3. 活動内容

おおよそ月に 1 回のペースで岩手県内の各自治体に遠征し，教室を開催する．教室の内容としては，光センサを用いて床に引かれた黒線に沿って走行するライントレースロボットを題材とし，命令ブロックを配置するプログラミングを行う．子供達は，暗黙の内に逐次構造，分岐構造，反復構造の必要性に気づかされ，さらには細い黒線を見つけるためにサンプリング間隔の重要性を痛感する．最後に，コースを周回するレースを実施する．

4. 活動の効果・気づき

活動を継続する内に教室の知名度も少しずつ上昇し，リピーターの子供達も現れた．結果として 8 年間の開催実績は 63 回にのぼり，1,317 組のご家族に参加頂いた．そこには，FCT による活動のモチベーション維持と岩手県職員の参加者募集への尽力が不可欠であった．

一方，トレーナーとして参加した各教育機関の学生達はボランティア活動であるにも関わらず，一対一の指導体制を維持するために多数参加した．そして，その多くが普段の受け身の授業では得られない「教える喜び」を味わった．終盤には，学生達が講師を任せられるまでになった．これは本活動が子供達の体験のみならず，学生達に対する人間教育に寄与したことを示唆している．以上から，本活動は産学官連携による活動の好事例になると思われる．

5. おわりに

8 年間にわたる「震災復興支援家族ロボット教室」の活動を振り返った．岩手県関係者を主体とした新体制による再出発ならびにその展開が今後の課題である．

謝辞

教室の開催にあたってノウハウを提供して頂いた NPO 法人 WRO Japan 様，ご協力くださった開催地域の皆様，実施各機関の数多くの協力者に深く感謝申し上げる．

インシデントリテラシ向上のための情報セキュリティゲームに導入教育を採用することの検証

廣瀬　司
放送大学
ergate24@gmail.com

藤井　辰雄
放送大学
robitalemon@gmail.com

中谷　多哉子
放送大学
tinakatani@ouj.ac.jp

要旨

近年，個人情報漏えい事件の増加により，情報セキュリティ教育が重要視されている．これまで我々が開発したゲームは，学習者が体験したインシデントを説明するゲームと，インシデントの解決策を手持ちの札から選ぶゲームから構成されていた．しかし，インシデントを説明する際，会話の内容がインシデントに焦点を当てることができずに解決策を選ぶゲームのためのインシデント事例を十分に収集できないという課題があった．この課題を解決するために，一つ目のゲームを行う前に，インシデントの導入教育を行うことにした．本研究は，ゲームの前に情報セキュリティ教育を導入教育として採用することにより，ゲームの有効性が向上したかを検証した．結果，導入教育を行ったことで，知識が増え，ゲームでの話題が情報セキュリティに関するインシデントに焦点が当たったといえる．したがって，インシデントリテラシ向上のための情報セキュリティ教育ゲームに導入教育を採用することは有効であった．

1. はじめに

人に起因する情報セキュリティの事件・事故 [1] は，情報セキュリティの重大課題とされており，情報セキュリティ教育は重要である [2]．企業が行う社員向けの情報セキュリティ教育 [3] には，集合型研修，テキストやメールの配布，ビデオや e-Learning コンテンツの受講などが存在する．これら知識伝達型講義は，組織での情報セキュリティに関する情報共有が可能である一方で，学習者が能動的に参加できない，理解不足などの課題がある．情報セキュリティ教育では各人が内容と対策について深い理解を示すことは重要 [4] である．
情報セキュリティ事象 [5] は情報セキュリティインシデント (以下，インシデント) を内包する,「違反」「不具合」であると定義されている．さらにインシデントは「望ましくない」「予期しない」情報セキュリティを脅かす確率が高いものとされている．

我々は，学習者を主体とする情報セキュリティ教育ゲーム（以下，本ゲーム）を開発し [7]，学習効果があったことを検証した．本ゲームは，学習者によるインシデントを収集する第一ステージと，続けて収集したインシデントを利用して，学習者が解決策を思考し，正解を決める解決策を思考する第二ステージの二つで構成される．

第一ステージでは，インシデントを説明する方法として，情報セキュリティ事象で不安なこと，危険な目にあったことを学習者が発話する「いやな話」ゲームを行う．参加者する学習者は順に前の人より「いやな話」を行い，最も「いやな話」をした人を勝者とする．学習者が自分で，インシデントを発見し，具体的に危険性を説明できることができれば，情報セキュリティ事故を想定できると期待する．
本ゲームは Salen and Zimmerman のいう「マジックサークル」というゲームの場と時間，ゲームに参加するという日常空間からは切り離された安全な空間の中 [8] で行い，その中での制約に従う.「いやな話」ゲームはインシデントの会話ゲームであるが，役割と順番を決めて制度的会話とする.「いやな話」の会話でのルールは，以下のとおりである．

- 話す順番はあらかじめ決めておく

- 全員が 1 話話すことを 1 巡として何回廻ったらゲームを終えるかあらかじめ決めておく，ただし最初の人は前の人が不在という不利な条件を避けるために 2 巡以上廻る
- 学習者の一人が話し手となるとき，周囲は聞き手となる
- 話す内容はインシデントに関する内容とする
- 前の人より「いやな話」をする
- 自分の順の「いやな話」をしないで終える，パスは禁止する
- 周囲が不快になる反社会的話題は禁止する

参加者は「キング　オブ　いやな話」を目指し，前の人よりいっそう「いやな話」をする．勝者を決めるのは，それまでの話と比較することで内容が凡庸になるのを防ぐ，ゲームの進行が滞るのを防ぐ，参加者の気分を盛り上げるためである．同様に,「いやな話」を提供しないのは場が白け るので，会話に参加しない「パス」を認めない．第一ステージで目指す成果はインシデントの話を収集することである．

次に行う第二ステージの「解決の沼」は，インシデントの解決策を思考演習するゲームである．解決策のヒントとなるカードを作成しておいて，ゲームの前に手札として学習者に配っておく．場に置いたインシデントカードをデックから順に引き，そのインシデントにふさわしい解決策を学習者が順番に自分の手札を使って説明していく．その解決策がインシデント解決にふさわしいか否かは他の参加者の話合いで決める， 解決策がふさわしくないと，順の学習者の手札が増えていく．解決策にふさわしいと手札を捨てることができ，手札が減っていく．最初に手札が空になった人，あるいは時間内に手札が一番少ない人が勝者となる．インシデントをカードデックにして学習者自身にどのインシデントが割り振られるかは，カードを引くまでわからない．解決の手札もどの解決策がを手札として得られるかわからない．どのインシデントが学習者自身にふりかかってくるか判明しない状況がインシデントの疑似体験に近しい．ふさわしい解決策を参加者全員の話合いで決めるのは，自分の順番にあたらなかったインシデントについても，全員で情報共有して理解を深めるためである．勝者を決めるのは，ゲームの進行が滞るのを防ぐ，参加者の気分を盛り上げるためである.「解決の沼」でも「パス」を認めない．インシデントについて不明な点は「いやな話」でインシデントの文を提供した人が「解決の沼」に参加しているので，その場で質問をし，疑問を解消できる．第二ステージで目指す成果はインシデントを自分の事として解決策を思考し，他の参加者に説明することで解決策の思考演習を行うことである．

また，本ゲームではゲームを円滑に進行するためにファシリテータを関与させる．ファシリテータの役割はゲームのルールを理解できない学習者に対して，説明をする，さらにインシデントの会話に積極的でない学習者の発話促進を行うためである．また，学習者の話題がルールから逸脱しないよう，ファシリテータが制御する．さらにファシリテータは，学習者だけで決めた解決策の中身が，誤った解決策になるのを防ぐ必要がある．ただし，ゲームの進行や制御のためにファシリテータが学習者の発言を否定したり，会話を遮ったりすると，学習者の主体性を失わせ，ゲームの不活性化を導く可能性がある [9]．本ゲームは学習者主体であることを重視して, ファシリテータは，教師的にふるまわないこととする．

2018 年，我々は本ゲームを実施した実験群と別の情報セキュリティ教育ゲームを実施した統制群との間で，インシデントの解決策について知識の定着の比較を行った．結果，本ゲームの有効性がみとめられた．課題として，第一ステージの「いやな話」で学習者がインシデントの説明に会話の焦点を当てることができなかった．情報セキュリティではなく，学習者の感じている日常の不満や不安に注意を向けてしまった．これにより，次の解決策を選ぶゲームのためのインシデント事例を十分に収集できなかった．この課題を解決するために，本研究において第一ステージのゲームを行う前に，インシデントの知識を増やすために導入教育を採用することにした．

本稿では，ゲームの前に導入教育を採用することによって，情報セキュリティに関するインシデントが収集できたかを検証する．また，インシデントリテラシとは，インシデントを具体的に説明できるようになること，そして，そのインシデントと解決策を関連付けることができることと定義する．

本研究の目的は，学習者がインシデントリテラシを身につけることである．本稿は以下の構成になっている．次の 2 章では，関連研究を述べ，3 章では研究のアプローチを述べ，4 章では，実験内容を述べ，5 章では結果を述べ，6 章では考察を述べる．

2. 先行研究

近年，ゲームを利用した教育についてさまざまな効果がみとめられている．井上 [10] は，ゲームを利用した教育において，学習効果は内発的な動機づけから発生し，能動的な学習態度が期待され，協調学習，学習者中心の学習，学習者同士の意見交換，繰り返し行うことで知識の蓄積がのぞまれると述べた．標葉らは，ゲームを利用した教育で，学習者が問題の解決のために， 多様な視点を獲得するカードゲームを開発し，その効果を測定した [11]．学習者はゲーム内で提示された一見，不可思議な問題に対して解決策の思考演習を行う．その後，ファシリテータが提示した正解を聞いて，学習者 はその内容について続けて議論を行う．標葉らは，学習者が想定し得なかった正解と解決策に，問題に対する多様な考え方を獲得することができると述べた．多角的視点を涵養するために学習者の日常から乖離した問題と正解を用意するため深く議論を重ねることとなった．一方で問題と正解の解決策の説明が困難で学習者の理解が得られず，ファシリテータによるゲームの遂行が困難であるという課題があった．このゲームには，勝敗が存在しないことから，ゲームの終了が判明しにくく，学習者の興味が薄れてしまう場合があった．

矢守ら [12] が開発した，実際に発生した災害事象を題材としたカードゲーム,「防災クロスロード」は正解がない．ジレンマが起きるような災害の場面で，参加者に災害担当者としての役割を与え，あなたなら， Yes/No の二択のどちらを選ぶかを問うゲームである．ゲームで提示される問題は文字数を制限して，学習者の想像力が働くよう，あえて説明を粗にしている．実際の災害現場で様々な体験をした学習者は多様な状況を考慮して，解答の選択を行う．一方，災害事象に詳しくない学習者は，問題を理解しないまま，Yes/No の解答選択を行う．吉川ら [13] は，学習者が勝敗にこだわりすぎて，ゲームに勝つために思考演習に至らないという難点があると述べている．

また，情報セキュリティ教育を題材とした教育ゲームも開発されている．情報セキュリティ教育では，最新のインシデントとその対策を早く周知させる目的を果たすために，開発に多大な時間を費やすことはできない．カードゲームは装置や多大な開発時間がかからないため，手軽な開発手法であるといえる．国内では，情報セキュリティカードゲームとして,「情報モラルかるた (2017 年発表)」 [14] が有る．このゲームは読み札に合わせて，場のとり札を取るかるたの方式である．ゲーム教育を行う前に，導入教育として情報システムを利用した成功例の話をし，ICT 機器活用の負の面への過度な訴求 (危険，怖いなど) を取り除き，楽しく学ぶことを目指している．田中によるとグループで楽しく学習活動の展開ができた，学習者が札の文言を覚えて他人に薦めたいと思うなど教育効果があったとしている．一方で，出題する全ての札はゲームの開発者が考えて提案したものであり，必ずしも学習者の興味をひくものではなかった．学習者が興味を持たない札は，内容を理解しないままであったという課題があった．

3. 研究のアプローチ

3.1. 導入教育

学習者にインシデントの説明をさせるには，学習者自身が体験したインシデントについて考えさせる必要がある．ゲームの前に学習者にインシデントの話題に注意を集められるように情報セキュリティの知識を増やす教育を行う．教育は学習者の周辺にどのようなインシデントが存在するか，発見する手がかりとする．ゲームではインシデントの教育を受けた後に，学習者のインシデント体験が説明される．学習者は自分が話す前までの人のインシデント体験を聞き，それらの内容よりもさらに,「いやな話」をする．参加者が一番共感した話が「キング　オブ　いやな話」に決まり，この話をした学習者がゲームの勝者となる．

図 1 にインシデント会話ゲーム「いやな話」の構成イメージを示す．ゲームでは，教育，いやな話，前の人のいやな話を聞いた上で語られるキングオブいやな話とで，情報提供者とそれぞれの内容の構成要素が異なる．詳細を表 1 に示した．教育はゲームを開始する前に一回だけ行われる．いやな話は　参加者の人数　X　任意の回数行われる.「キング　オブ　いやな話」は全ての「いやな話」が終わったのちに，参加者の共感を最も得たインシデント体験として存在する．

3.2. 導入教育の選択

導入教育は，学習者自身が直面するインシデントについて考えさせるために，身近な話題である必要がある．2017 年の時点で，日本国内の情報通信機器の世帯保有

図 1. インシデント会話ゲームの構造イメージ

表 1. インシデント会話ゲーム「いやな話」の構成要素

内容	情報提供者	構成要素
教育	ファシリテータ	導入教育
いやな話	学習者	教育で知識を得た上で会話するインシデント体験
キングオブいやな話	学習者全員	周囲の人のインシデントの知識を得た上で参加者の共感を最も得たインシデント体験

率はスマートフォンがパソコンを上回り [15]，スマートフォンは身近な通信機器として普及している．スマートフォンには，経済産業省のキャッシュレス化の推進 [16] により，携帯電話決済機能や送金機能が追加された．2019 年の 11 月期に行われた電子決済には 349 万台のスマートフォンが使用され，2018 年 12 月には 273 万台であったことから 76 万台の増加があり [17]，スマートフォン決済が増加したことがわかる．またショートメッセージサービス (以下，SMS とする) 機能は，スマートフォンの基本機能であり，携帯電話会社からスマートフォン契約者への料金案内などに利用されている．この SMS を利用したフィッシング詐欺 (以下，スミッシング) は図 2 に示したように，増加 [18] 傾向にある．導入教育は，近年，スマートフォンの利用者が増加し，スマートフォンの決済機能などをねらった詐欺事件が増加していることから，学習者に起こりうるインシデント導入教育として，スミッシング防止対策を選択する．本研究では，スミッシングの内容をまとめたものを教材として開発した．

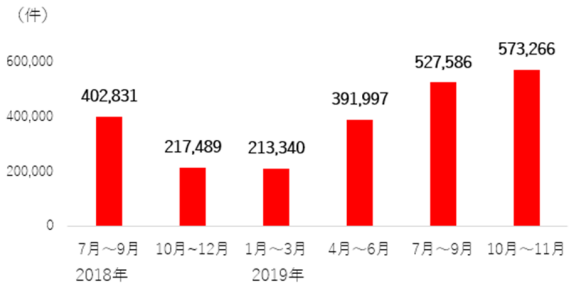


図 2. 不正サイトへ誘導された国内モバイル利用者数推移

3.3. 導入教育の内容

導入教育の目的はスミッシングの仕組みと概要を知り，インシデントの知識を増やすこととする．

教育対象は，スマートフォンを所有し，SMS を見る機会がある人とする．学習目標は，教育を受けた受講者がスミッシングの概要と内容を説明できるようになることとする．教育の方法は，パワーポイントを使ってファシリテータが説明する．時間は 15 分程度とする．内容 1 として，スミッシングの概要を知る スミッシングの仕組と被害増加の要因を説明する．内容 2 として，スミッシングメッセージの確認方法を説明する．また，スミッシングの対策法として，教育後，受講生が最新情報を得ることができる関連情報サイトを紹介し，継続的な情報収集を行えるよう配慮した．

3.4. 共起ネットワーク図による分析

情報セキュリティの導入教育を行い，その後ゲームを利用して収集したインシデントの文と，過去にゲーム前に導入教育を行わなかった実験で収集したときのインシデントの文を比較する．インシデントの文の特徴を分析するために，どのような単語が同時に出現しているか，単語同士にまとまりがあるかを感覚的にとらえるために，テキストマイニングツール KH Coder3 を使って共

起ネットワーク図を描いて分析する．
次に共起ネットワーク図の中から，まとまりの数と，単語の出現回数，まとまりの中で中心性が高いものを重要な役割を果たしている単語とし，単語が実際にどのように使われているか確認する．単語が，情報セキュリティに関係する単語であれば，ゲームで集めたインシデントは適切であったとする．本研究では，インシデントの文を収集する際の会話分析は行わない．学習者が発話したゲームのインシデントをファシリテータが，その場で要約し,「こういった内容ですね」と学習者に確認したものを，テキストデータ化し，導入教育の有無の別に分け分析する．

4. 実験

4.1. 実験の流れと概要

情報セキュリティ導入教育による教育を行い，その後にゲームを実施した．導入教育のテーマは，スマートフォンのスミッシングについて取り上げた．
また，実験を円滑にすすめるために，実験者がファシリテータとなり，導入教育の説明を行い，ゲームをすすめる役割を担った．実験は，参加者に特定の条件（年齢，職業，性別など）を付けなかった．導入教育がスミッシングであることから，携帯電話を持っている人を対象とした．
参加者を実験者の職場の会議室に集合させ，実験内容を説明し，参加を呼びかけた．参加人数に制限を設けず，実験に協力してくれる有志を集めた．参加者には，事前に実験の主旨を説明し，自由意志でいつでも参加を中止できることを伝えた．この条件で参加者を募集したところ，職員他 4 人が参加者となった．4 人の年齢は 30 代，40 代，50 代，60 代であった．最初に導入教育でスミッシング防止対策の教育を行い，次に学習者主体によるインシデントの収集ゲーム「いやな話」を 2 巡行った．参加者が勝者である「キング・オブ・いやな話」を決め，ファシリテータが「いやな話」の内容を「こういうことですね」と発話者に確認してテキストデータにして終えた．
実験日時 : 2020 年 1 月 20 日休憩時間を利用して，15 時から実施
場所：実験者の職場会議室
実験に要した時間 : パワーポイントとスクリーンを利用して導入教育に 15 分，インシデント文収集に 20 分間要し，8 文集まった．

4.2. 導入教育無の場合との比較

過去に実施した本ゲームの実験時に収集したインシデントの文と，今回のインシデントの文の比較を行った．

5. 結果

5.1. 導入教育無ゲームで収集したインシデントの文

過去に実施した本ゲームの実験は，2018 年 2 月 1 日，2 月 5 日，5 月 11 日の 3 回であった．実験参加者はそれぞれ 3 名ずつであった．この際に参加者に特定の条件（年齢，職業，性別など）を付けなかったが，参加者は実験者の職場の職員と放送大学のクラブ活動参加者であった．参加者の内容は，本稿の実験メンバーとは重複していなかった．毎回,「いやな話」をそれぞれ 2 巡ずつ行い，インシデントの文は 18 文集まった．過去 3 回実施した本ゲームで収集したインシデントの文を導入教育無データとしてまとめた．18 文のうち単語の総出現回数は 587 回，異なり単語数は 209 個であった．
この導入教育無データを共起ネットワーク図にしてを描画した．図 3 に示す．共起ネットワークを描く条件と

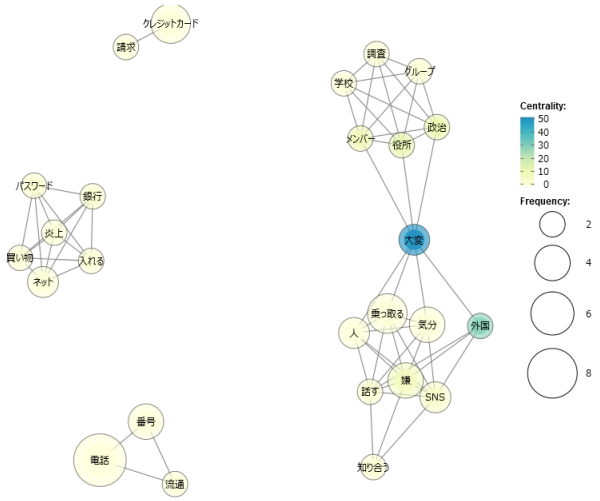


図 3. 導入教育無ゲームで収集した情報セキュリティインシデント文の共起ネットワーク

して，全ての単語を網羅すると，出現単語が多すぎて，描画が複雑になり特徴を捉えることが難しくなる．そこで単語の出現回数は 3 回以上に設定した．単語の出現回数の多さは円の大きさに比例する．また，円と円との関係が強いところは線の太さで表現した．共起ネットワークの中で単語が中心的な役割を果たす中心性（centrality）の度合いが強い単語は濃い色で示した．
出現回数の多いものは（以降，単語としてあらわすときは，わかりやすくするために『』で囲む）『電話』,『クレジットカード』,『乗っ取る』であった．

まとまりは 4 つあり，それぞれが結びつきのないものであった．大きなまとまりは,『大変』を軸にして，2 個のまとまりを 1 つにした形であった．この大きなまとまりは，それぞれ，役所，政治，学校などを結ぶまとまりと，外国，乗っ取る，気分，嫌，SNS，などを結ぶまとまりからできている．他には，クレジットカード，電話，炎上などを囲む小さなまとまりが存在する．

中心性の強いものは,『大変』が強度 50，次いで『外国』30，次が『役所』『政治』20 であった．文脈の中では中心性の強い『大変』は調査，隠す，ハラスメント，外交，政治,『外国』は飛行機，降ろす，日本人，目的,『役所』はマスコミ，不愉快，学校，忖度，乗っ取り,『政治』は人，言う，忖度，学校，大変という単語とともによく出現していた．
統制群で出現したインシデントは，政治やマスコミに対する不満，SNS を介した人付き合いが嫌になる話など情報セキュリティとは関係ない内容があった．

5.2. 導入教育有ゲームで収集した情報セキュリティインシデントの文

次に本研究で行った実験で収集した導入教育有データで共起ネットワークの描画を行った．本研究では 4 人が参加し,「いやな話」ゲームを 2 巡行い，集まった情報セキュリティインシデントの文は 8 つであった．単語の総出現回数は 351 回，異なり単語数は 129 個であった．

この導入教育有データを共起ネットワーク図にしてを描画した．図 4 に示す．出現回数の多いものは『電話』,『クレジットカード』,『インターネット』であった．まとまりはおよそ 3 つあり，全て複数の結びつきがあった．単語『サイト』を中心にして『特定』を経て『攻撃』や『メール』に繋ぐ関係と,『詐欺』を経て『クレジットカード』に繋ぐ関係と『買い物』から『電話』を経て『プログラム』に至る関係が結ばれていた．全てのまとまりは何らかの関係でつながっていた．中心性の強さを表す指標の高いものは,『サイト』が一番強い 50，次いで『クレジットカード』40，次が『攻撃』『メール』『プログラム』30 であった．文脈の中では中心性の強い『サイト』は詐欺，SSH，海外，買い物，攻撃,『クレジットカード』は，ない，使う，決済，情報，買い物,『攻撃』は，SSH，受ける，ズレ，サイト，個人,『メール』はアドレス，経由，電子，ウィルス，パソコン,『プログラム』は，ブロック，仕込む，削除，ハッカという単語がよく出現していた．
実験群で説明されたインシデントは，サーバがゼロディ攻撃に合い，苦労して復旧した話，インターネットで買い物をした際のトラブルなど情報セキュリティに関する話があった．
また，スミッシング詐欺被害防止導入教育の利用にあたり，スミッシング被害体験の有無を学習者にたずねたが，全員，スミッシング被害体験が無かった．

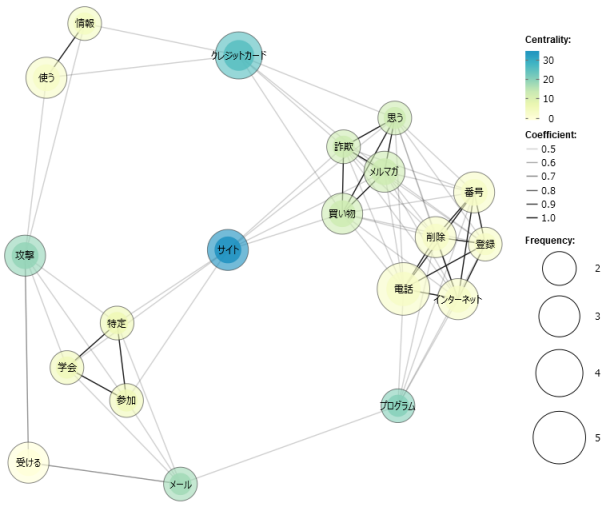


図 4. 導入教育有ゲームで収集した情報セキュリティインシデント文の共起ネットワーク

6. 考察

インシデントの文の中の単語の出現回数について比較すると，導入教育無ゲームでも，導入教育有ゲームでも，『電話』が 1 番出現回数が多かった．近年，電話はほとんど携帯電話を指し，携帯電話はほとんどコンピュータの機能を搭載している．学習者の最も身近なコンピュー

タであるという見方をすると,『電話』はコンピュータの代名詞であるともいえる．次いで,『クレジットカード』であった．クレジットカードが経済的被害を及ぼすという意味で上位に挙がったと考えられる．今後，携帯電話の決済機能の利用増加があると，さらに『電話』『クレジットカード』の出現割合が上昇する可能性がある．3 番目に出現回数の多い単語は，導入教育無ゲームでは『乗っ取る』であった．乗っ取るは気分，マスコミ，人格，社会，役所といった単語とともに文脈に出現し，情報セキュリティ関連としての『乗っ取る』とは異なる意味で使われている．導入教育無ゲームのインシデントの文が情報セキュリティの話題でないものが含まれていることがわかる．導入教育有ゲームで 3 番目に出現回数が多かった単語は『インターネット』であった．導入教育有ゲームのインシデントの文は情報セキュリティの話題が多かったことがわかる．

共起ネットワーク図のまとまりを比較したところ，導入教育無ゲームでは，4 つのまとまりに分かれていたが,．まとまりの間に関連はない，導入教育有ゲームでは，まとまりは 3 つであるが全てのまとまりに関連があった．導入教育無ゲームより，導入教育有ゲームのほうが，インシデントの文の単語に共通性があった．インシデントの文における単語の重要な役割をはたす中心性の強さでは，導入教育無ゲームで，最も中心性のある単語が『大変』である.『大変』の意味からインシデントが学習者にとって「はなはだしい程度」であり,「一大事件」であることを示している．周囲には,『外国』,『役所』,『政治』があり，これらの出現する文脈では，政治や外交，マスコミや不愉快，学校や忖度などの単語がよく出現し，政府やマスコミに対する不満，人付き合いへの嫌悪感などが話された．これは必ずしも情報セキュリティに関する話題でないことがわかる.「いやな話」のルールとして,「いやな話」をするという部分は作用していたといえるが,インシデントに関する内容というルールは守られていなかったといえる．実際には学習者はインシデントの話といっても何を言っていいかわからないことがある．誰かが話題を身の回りの不満や不安の話に変えると，それに引きずられる傾向にあった．ファシリテータがインシデント以外の話題への変化に気づき，学習者の気分を損ねないように制御することが求められるところである．しかし．ファシリテータの力量のみに頼るのは難しい．

導入教育有ゲームでは，最も中心性のある単語は『サイト』であった.『サイト』は「Web サイト」であり，以下,『クレジットカード』,『攻撃』,『メール』,『プログラム』と続く．これらの出現する文脈では，SSH や攻撃，買い物や詐欺，ウィルスやブロックなどの単語がよく出現し，情報セキュリティに関することが話された．これは,「いやな話」をする前に導入教育を行ったことで，話題が情報セキュリティに関するインシデントにまとまったといえる．学習者は何を言っていいかわからないという混乱もなく，遅滞なくインシデントの説明を行えた．したがって，インシデントリテラシ向上のための情報セキュリティ教育ゲームに導入教育を採用することを検証した結果，有効であったといえる．

しかしながら，導入教育がインシデント収集に影響を与えるならば，収集したインシデントはスミッシング被害と一致するはずである．実験に参加した学習者全員にスミッシング被害体験が無かったことに関連があるかは不明である．次に本ゲームを行う際には，学習者に関連のある別の導入教育を用意して効果を確認する必要がある．また，本稿では，インシデント収集の際の課題解決に導入教育を採用することを検証したが，導入教育が，今後,「解決の沼」解決策思考演習においてどのような影響があるかを検証する必要がある．さらに本研究を一般化するにあたり，異なる状況設定，異なる参加者を設定し，実験を行う必要がある．今後は実験を整備し，他集団において実験することが必要である．

参考文献

[1] 独立行政法人情報処理推進機構. 情報セキュリティ１０大脅威　2018, 2018.

[2] JNSA(日本ネットワークセキュリティ協会). 2015 年情報セキュリティインシデントに関する調査報告書個人情報編, 2016.

[3] 株式会社日本シュレッダーサービス. 情報セキュリティに対しての教育, 2018.

[4] 日本工業規格. JIS Q 27002:2014 情報セキュリティマネジメントの実践のための規範, 2014.

[5] 日本工業規格. JIS Q 27000:2014 情報セキュリティマネジメントシステム-用語, 2014.

[6] 藤本徹. 効果的なデジタルゲーム利用教育のための考え方. コンピュータ & エデュケーション, Vol. 31, pp. 10–15, 2011.

[7] 廣瀬司, 中谷多哉子. 情報セキュリティ教育のためのカードゲームの検証. 研究報告ソフトウェア工学 (SE), Vol. 2018, No. 21, pp. 1–7, 2018.

[8] Katie Salen Tekinbas and Eric Zimmerman. *Rules of Play Game Design Fundamentals (The MIT Press)*. The MIT Press, new edition, 9 2003.

[9] 堀公俊. ファシリテーション入門. 日本経済新聞社, 7 2004.

[10] 井上明人. ゲーミフィケーション - ゲームがビジネスを変える. NHK 出版, 1 2012.

[11] 標葉靖子, 江間有沙, 福山佑樹. 科学技術と社会への多角的視点を涵養するためのカードゲーム教材の開発. 科学教育研究, Vol. 41, No. 2, pp. 161–169, 2017.

[12] 矢守克也. 巨大災害のリスク・コミュニケーション: 災害情報の新しいかたち. ミネルヴァ書房, 9 2013.

[13] 吉川肇子, 矢守克也, 杉浦淳吉, 主著に, リスク・コミュニケーション, 福村出版, 京都大学大, 生活, 防災. クロスロード・ネクスト. 続: ゲームで学ぶリスク・コミュニケーション-. ナカニシヤ出版, 223pp, 2009.

[14] 田中康平. 「情報モラルかるた」を活用した、楽しく学ぶ情報モラル・情報セキュリティ. 第 43 回全日本教育工学研究協議会全国大会, pp. 229–232, 2017.

[15] 総務省. 平成 30 年版情報通信白書, 2019.

[16] 経済産業省. キャッシュレスビジョン, 2018.

[17] 日本銀行. 決済動向 2019 年, 2020.

[18] 独立行政法人情報処理推進機構. 情報セキュリティ 10 大脅威 2019, 2019.

コミュニケーション改善を目的とした論文形(かた)研修

株式会社 SRA
阪井　誠
sakai@sra.co.jp

要旨

コミュニケーション改善を目的とした論文形研修の事例を報告する. ソフトウェア開発においてコミュニケーションは重要な問題である. 本事例では, 開発者の論理的思考力が向上すれば, 齟齬が生じにくくなると仮説を立て, 論文形のワークショップを実施した. 受講者の感想を集計した結果, 8 名中 6 名が仕事に役立つ, 2 名が有効だがさらに訓練が必要であると回答し, 研修の有効性が確認できた.

1. はじめに

ソフトウェア開発においてコミュニケーションは重要な問題であり, 1975 年に書かれたブルックスのソフトウェア開発の神話にもバベルの塔を引き合いに「コミュニケーションとその成果である組織は成功のための条件である」と述べている[1].

コミュニケーションを適切に行うには, 論文を書く際と同じように, 問題を把握し, 整理し, 論理的な構造で伝える必要がある. しかし, 多くの社会人は論文執筆の経験がないので, その形(かた, 以降「論文形」と呼ぶ)を知らず, 冗長な表現, 誤解, 勘違いなどのコミュニケーションの問題を生じさせていると考えた.

そこで, 論文形を中心とした技術文章の研修がコミュニケーション改善に役立つと仮説を立て, 仕事に役立つかをアンケートで確認した.

2. 研修の形態と内容

研修は未公開の技術情報を扱うことから社員限定の研修とし, 2 時間×4 回実施した. 具体的には, 講義を 1 回, 論文形式の概要を 1 ページ記述するワークショップを, はじめに, 本文, 概要の 3 回に分けて実施した[2].

研修で重視したのはパラグラフライティングなどの形である[3]. 論文は結論を先に説明するパラグラフライティングの入れ子構造になっているが, 初心者には難しいと思われたのでワークショップを取り入れた. ワークショップでは作文, 発表, 受講者の感想のあと, 添削指導した.

受講者の感想はあまり多くなく, 査読者のつもりで評価してもらうなどの工夫が必要だった可能性がある.

3. 研修の感想

研修の最後に, 仮説の評価を目的として論文形研修の内容が仕事に役立つか感想を書いてもらった. 感想を集計した結果, 8 名中 6 名が仕事に役立つとし, 2 名が有効だがさらに訓練が必要であると回答した.

この 2 名の研修中の表情を見ていると, 身につまされているようであった. マネージャには切実で, さらにトレーニングが必要と回答していた. 役立つとした 6 名の中でも文章だけや, 問題設定の「やったことの裏返し」などが部分的に役に立つとした 2 名の受講者は, 実践と結びつかず, どこかぴんと来ていない様子が見られた.

4. おわりに

コミュニケーション改善を目的として論文形研修を行った. 日本では技術文書の書き方を大学院まで教えられておらず, 典型的な形式に整理して報告することに慣れていない社員が多かったからである.

感想を見ていると, 研修を通じてパラグラフライティングなど形の重要性はある程度わかってもらえたと考えられる.

今後は追加の研修や OJT を通じて指導したい.

参考文献

[1] F. P. ブルックス, ソフトウェア開発の神話, p.81, 企画センター,1977.
[2] 阪井, 論文の書き方, http://sakaba.cocolog-nifty.com/sakaba/cat23887317/index.html
[3] 石原尚, パラグラフライティングの作法 -書き手にもメリットのある文配置ルール, http://www.ams.eng.osaka-u.ac.jp/user/ishihara/?p=566

探索的仕様記述のための履歴ツールの提案と実装

小田 朋宏
株式会社 SRA
tomohiro@sra.co.jp

張 漢明
南山大学
chang@nanzan-u.ac.jp

山本 恭裕
公立はこだて未来大学
yxy@acm.org

中小路 久美代
公立はこだて未来大学
kumiyo@acm.org

荒木 啓二郎
熊本高等専門学校
araki@kyudai.jp

要旨

ソフトウェア開発の過程を困難なものとしている一つの要因として，複雑なものを対象にした試行錯誤が挙げられる．ソフトウェアシステムを作成する過程において，複雑な対象ドメインやソフトウェアシステム自身への理解を探索的に深め，得られた知見を整理し，ドキュメントとして記録することが求められる．ソフトウェア開発における履歴の管理と整理は，ソフトウェア工学の古くからの課題であり，多くのツールが実現されて広く利用されている．本論文では，ソフトウェア開発における試行錯誤の中でも，形式仕様記述における個人作業の中での探索的な作業に注目して，仕様記述の編集や表現式の評価実行のような細粒度の履歴の整理のための手法を提案する．そして，実行可能な仕様記述言語 VDM-SL での仕様記述における探索的試行を記録し，後に仕様記述者自身がその過程を整理し，記録するためのツールを提案し，その実装を紹介する．

1. はじめに

ソフトウェア開発を困難なものにしている要因として，複雑さがある．ソフトウェアシステム自体の複雑さに加え，ソフトウェアシステムの対象ドメインの複雑さや，ソフトウェアシステムを利用する人の振る舞いに関連した複雑さから，開発の開始時点では開発に必要な知識を前もって獲得することは困難である．ソフトウェアシステムを作りながら，対象ドメインに存在する概念や，ユーザーの振る舞いから利用シナリオを抽出し，かつ，それら対象ドメインや利用シナリオに対してどのようなソフトウェアシステムが適合するか理解を深めていく必要があることから，多くのソフトウェア開発では探索的な試行によって理解を深めていくことが求められる．

機能仕様は開発対象の機能や特性を記述したものであり，他の工程の多くで参照される．機能仕様の策定にも探索的な試行が行われるが，試行には部分的な失敗を含むことが多いことから，他の工程に波及するとコストがかかる．形式仕様はソフトウェアシステムを実装する以前にその特性を分析することを可能にすることから，仕様記述段階での探索的試行に適している．著者らは形式仕様工程のうち探索的試行が多くおこなわれる初期段階を探索的仕様記述として，それを支援するための開発環境 ViennaTalk [1] を開発してきた．ViennaTalk は仕様記述言語として VDM-SL [2] を採用し，その実行可能なサブセットによる仕様アニメーションを中心にして，VDM-SL 仕様記述を実行可能なプロトタイプとして様々な利用シナリオへの適合性を評価し，探索的な試行を支援するための機能を提供している．

探索の結果として得られる知見は，最終的に記述された VDM-SL 仕様だけではない．探索の過程で試した多くの暫定的な仕様記述や，その上で検討した利用シナリオや，その利用シナリオでのシステムの動作を記録することで，その仕様記述の機能項目がなぜそのように定義されているかを説明するドキュメンテーションとしての役割を果たすことができる．

本論文では，探索的な試行の履歴を整理するためのモデルと，その実装例として VDM-SL 用のウェブ IDE で

ある VDMPad [1] に探索的試行の履歴を整理する機能を追加した VDMPad-EpiLog を紹介し，その利用例を示す．第 2 節で探索的試行の履歴を整理するための構造とそれに対する操作を提示し，その実装として VDMPad-EpiLog の機能とその利用例を第 3 節に示す．第 4 節では関連する既存のツールとして github および git を中心に比較検討する．第 5 節に議論とまとめを示す．

2. VDM-SL 仕様記述の履歴モデル

本節では，VDM-SL 仕様を記述する個人作業における探索的試行の履歴を整理するための構造を提案する．本研究では開発における探索的試行について，対象に関する部分的な知識に基づいて暫定的仮説的なモデルの記述を作成し分析を行うことを 1 単位の試行とした上で，1 つの試行を通して深められた理解に基づいて次の施行を扱う，サイクルプロセスと捉える．仕様記述をおこなう中で，探索的に繰り返された試行を抽出することで，履歴を構造化し整理する．

VDM-SL は，実行可能なサブセットを持つ，モデル規範型の形式仕様記述言語である．仕様記述者は対象を内部状態および内部状態に対する操作としてモデル化し，データ型，定数，関数，状態空間，操作の 5 種類の定義によって記述する [2]．データ型を `types` 部で宣言し，定数および関数をそれぞれ `values` 部および `functions` 部で定義することによって，そのシステムが扱う概念とその関係を記述する．VDM-SL では定数および関数は参照透明である．`state` 部では，システムが持つ状態空間を定義するために，静的に型付けられた変数のリスト，状態に対する不変条件，および初期状態を宣言する．システムが提供する操作を `operations` 部で定義する．操作は状態に対する参照や変更を行うことができる．

形式仕様記述言語としての VDM-SL の大きな特徴として，そのサブセットをインタプリタで実行可能であることが挙げられる．VDMTools や Overture Tool [3] を始めとする VDM-SL 開発環境の多くが，REPL (Read-Eval-Print Loop) インタプリタを提供し，仕様記述者は編集中の仕様記述に対してインタプリタを起動して，表現式を入力して評価実行することができる．仕様記述の探索的試行において，インタプリタ実行は編集中の仕様記述の特性を把握する上で，重要な役割を果たしている．

VDM-SL によるモデル化の前段階として，システムの機能仕様に対する原始要求として，システムの利用シーンをいくつか想定し，それぞれの利用シーンをユースケースとして，どのような操作をどのような順番でどのような判断を伴いながら実行するかを想定することが多い．システムの動作を表現した記述として，ユニットテスト [4] がある．

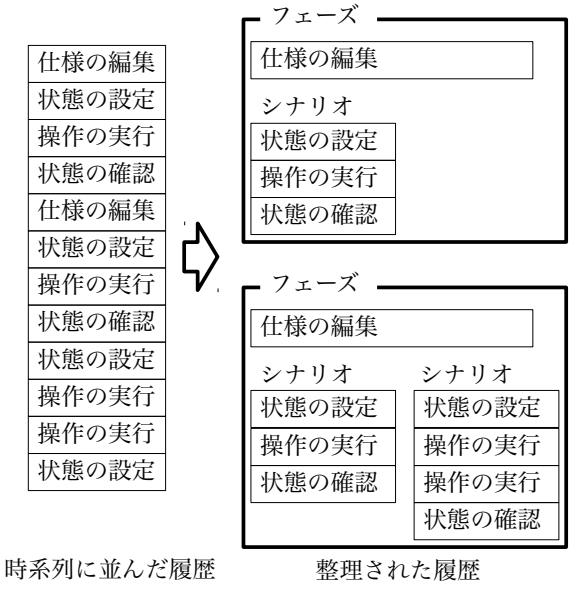


図 1. 履歴整理の例

VDM-SL 仕様に対するユニットテストフレームワークは vdmUnit や viennaUnit [5] などが既に開発され利用されている．

ユニットテストでは，システムの構成部品に対して想定される利用シナリオ毎に，テストケースを定義する．各テストケースは，シナリオの前提となるシステム状態の設定，システムに対する一連の操作の実行，操作結果の確認を，テスト用の操作として記述したものである．VDM-SL 仕様のユニットテストは，記述対象に対する利用シナリオへの適合性の検査を自動化したものと見なすことができる．

本研究では，ユニットテストを参考にして，探索的試行のモデルを定義した．探索的試行の中では，仕様記述の編集，状態の編集，操作の評価実行が行われる．仕様記述者は，それぞれの時点での暫定的な仕様記述に対して，利用シナリオをいくつか想定して，利用シナリオを状態編集と操作列として模擬的に実行し，実行結果が利用シナリオが想定した通りかを確認する．履歴を，仕様記述のバージョンに対応する**フェーズ**，利用シナリオに

対応する**シナリオ**，個々の状態編集や操作実行に対応する**ラン**の 3 つの単位にまとめることとした．

フェーズは，記述対象に関する理解の変化を捉えることを主眼にした単位である．仕様記述は，仕様記述者が対象をモデル化したものであり，仕様記述者による対象に対する理解を表現している．本節冒頭で述べた通り，本研究では仕様記述における探索を，仕様記述者による対象への理解を表現した暫定的仮説的な仕様記述とその記述に対する一連の検証を 1 つの試行とする．フェーズは 1 つの仕様記述とそれに対する検証をおこなうための一連の評価実行をまとめたものであり，1 単位の試行と見なすことができる．

シナリオは，仮説検証における検証の単位である．仕様記述で定義されたシステムの動作が対象ドメインや原始要求に対して妥当であるかどうかを確認するために，システムの特定の動作を切り出したものをシナリオとする．1 つのフェーズの中で，複数のシナリオによる妥当性の確認が行われる．シナリオには，原始要求に含まれる利用シナリオや対象ドメインからくるものもあれば，仕様記述に定義された機能単位からくるものもある．

ランは，シナリオの妥当性を確認するために行う，仕様記述者による操作の単位である．VDM-SL 仕様はシステムを状態空間と操作群により定義することから，VDM-SL 仕様記述での探索では，状態の初期化や，操作の評価実行を試行の基本単位として扱うことができる．具体的にどのような種類のランがあるかは，開発環境のユーザインタラクションの設計による．REPL (Read-Eval-Print Loop) インタプリタの場合，インタプリタが提供するコマンドのうち，アニメーション実行に影響を与えるものをランとして扱うことができる．Overture Tool には，仕様記述が定義する操作をボタンにした簡易 GUI によるアニメーション機能があるが，この場合は各ボタンの押下をランとして扱うことができる．

図 1 に，仕様の修正をともなう探索的試行の履歴について，時系列からフェーズ，シナリオ，ランの 3 つの粒度の構造に整理した例を示す．左側には，探索の中で行われた仕様の編集，状態の設定，操作の実行，状態の確認が上から下の順に時系列で列挙されている．仕様記述における探索的試行では，暫定的な仕様記述を作成し，その仕様記述が利用シナリオにどのように適合するか，システムがどのように動作するかを確認する．いくつかの利用シナリオが試され，システムの動作が仕様記述者の意図と異なる場合には仕様記述を編集する．1 つの暫定的な仕様記述に対する一連の履歴項目を，1 つのフェーズとする．フェーズは，履歴の時系列から，仕様記述の編集から，次の仕様記述の編集までの区間を切り出すことで，容易に抽出することができる．

フェーズの中では，いくつかの利用シナリオに関する適合性の確認が行われる．ここでの利用シナリオは，原始要求の中で明示的に示されたものもあるが，仕様記述者によるドメインへの理解から仮説的に想定しているものもある．したがって，シナリオの同定は必ずしも機械的にできるとは限らない．探索の履歴を整理する上で，シナリオは原始要求や仕様記述者の理解を反映する重要な単位であることから，探索の履歴を整理するためのツールを設計する上で重要なポイントになる．

ランはシナリオを構成する基本要素であり，シナリオの内容はランの時系列として表現される．ランは多くの場合，開発環境が収集した時系列の履歴の各項目から抽出することができる．

3. VDMPad-EpiLog

本節では，探索的試行の履歴を整理するツールの実装例として VDMPad-EpiLog を説明する．VDMPad-EpiLog は， 探索的な仕様記述の特に初期段階の試行を想定して設計されたウェブ IDE である VDMPad [1] に対する拡張として実装された．

図 2 に VDMPad の画面例を示す．VDMPad は仕様アニメーションを中心に設計されている．仕様アニメーションとは，仕様をインタプリタ等で評価実行することで，与えられた仕様を実装したシステムがどのように動作するかをシミュレートする技術である．VDMPad は，VDM-SL で記述された仕様のソースと，仕様が定義する状態空間におけるシステムの現在の状態を保持し，それをプロトタイプとしてウェブページ上に表示する．仕様記述者はプロトタイプに対して，仕様記述，状態，評価式を編集して与え，評価実行ボタンを押すことで評価結果を得るとともに，プロトタイプの状態を更新する．VDMPad は，柔軟な探索のために，仕様 / 状態 / 評価式のいずれも常に編集可能であり，それらはいずれもプロトタイプへの修正操作として扱われる．また，プロトタイプの仕様を編集しても，プロトタイプを再起動することなく，継続してプロトタイプに対する操作を行うことができる．

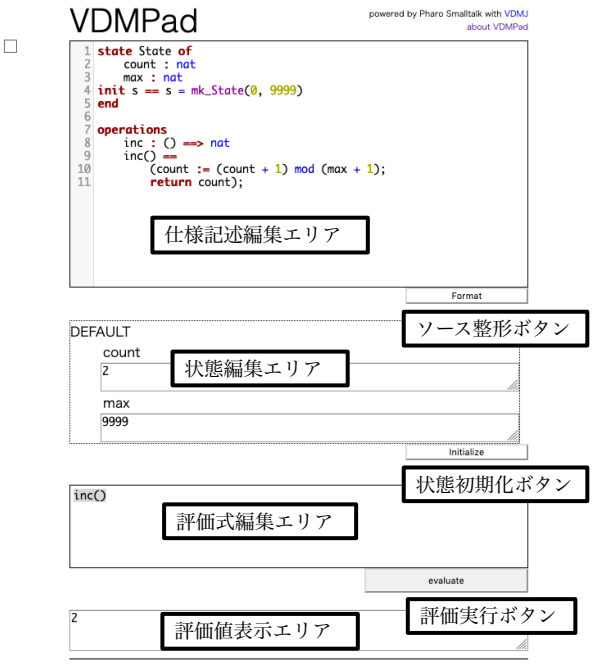


図 2. VDMPad の画面例

3.1. VDMPad-EpiLog の概要

VDMPad-EpiLog は，VDMPad を拡張して，履歴を整理する機能を追加したものである．VDMPad では，プロトタイプは受動的であり，仕様記述者が評価実行ボタンおよび状態初期化ボタンを押すことで，プロトタイプを駆動する．評価実行ボタンが押されると，VDMPad は仕様記述のソース，現在の状態，評価式をサーバに送信し，評価結果の値，評価後の状態，エラーの場合にはエラーメッセージを返す．VDMPad-EpiLog はそれを捉えて，エラーでない場合のみ，仕様記述のソース，評価前の状態，評価式，評価結果の値，評価後の状態を保存し，以下の手順で自動的にフェーズ，シナリオ，ランとして整理する．

- 仕様の断絶：仕様記述が直前のものと異なる場合には，新しいフェーズを生成する．
- 状態の断絶：上の条件に当てはまらず，かつ，評価前の状態が直前の履歴での評価後の状態と異なる場合には，新しいシナリオを生成し，最新フェーズのシナリオリストに追加する．

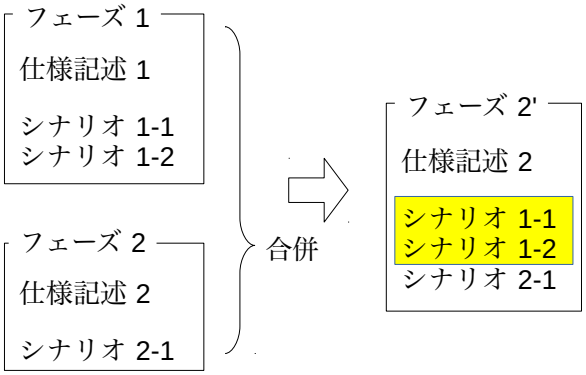


図 3. フェーズの合併

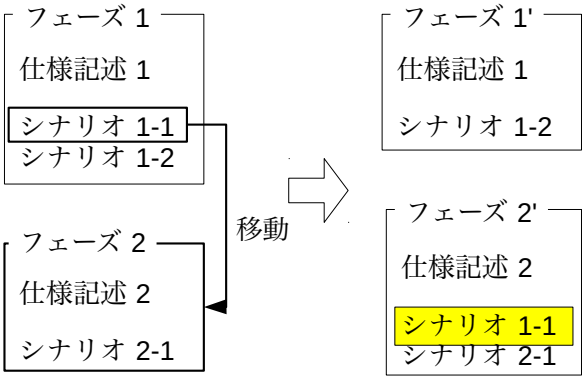


図 4. シナリオの移動

- 連続した実行：いずれの条件にも当てはまらない場合には，最新シナリオの末尾にランとして追加する．

こうして自動的に整理された履歴は，ユーザによる探索的試行を実行の時系列に忠実に反映した履歴である．しかし探索の結果得られた知見としてまとめる上では，いくつかの問題点が考えられる．

仕様記述を編集するごとにそれに対応するフェーズが生成されるが，そうしたフェーズにはあまり重要ではないものも多い．仕様記述の修正のうち，知見として残す必要のない軽微な修正は，重要なフェーズと合併する，あるいは削除することで，重要なフェーズのみを残すことができる．図 3 にフェーズの合併を示す．フェーズ 1 の仕様に軽微な問題があり，フェーズ 2 の仕様でそれが解決された場合，フェーズ 1 をフェーズ 2 に合併することで両方のシナリオを修正後であるフェーズ 2 の仕様

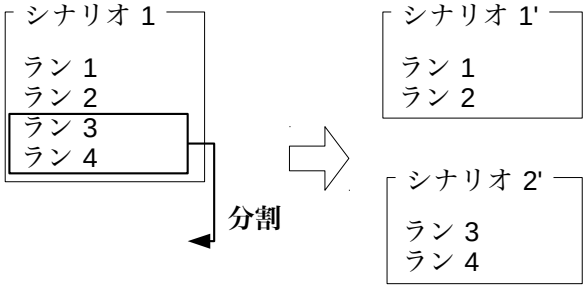


図 5. シナリオの分割

の上でのシナリオとすることができる．ただし，フェーズ 1 にあった 2 つのシナリオは，試行時にはフェーズ 1 の仕様で実行されたものであり，フェーズ 2 の仕様での実行結果とは異なる可能性がある．したがってフェーズ 1 から移動したシナリオに含まれるランの実行結果は無効マークが付与される．無効とされたランは，移動先のフェーズの仕様で再実行することで，無効マークを取り除くことができる．また，図 4 に示すように，特定のシナリオのみ次のフェーズに移動することもできる．

探索的試行の中で，同一フェーズ内で連続して複数のシナリオを検討する場合，シナリオ開始時に特に状態を変更する必要がない場合がある．状態の変更を伴わずにシナリオでのランを開始すると，VDMPad-EpiLog はシナリオの切れ目を検出できず，同一シナリオの継続として処理する．探索的試行の中で意図したシナリオに分割するために，シナリオの特定のラン以降を別のシナリオとして分割することができる．シナリオの分割操作を図 5 に示す．

フェーズやシナリオ，ランには繰り返し同じものが現れたり，特に重要ではないため知見として残す必要のないものも多い．したがって，不要なフェーズやシナリオやランを削除する操作も提供される．

3.2. VDMPad-EpiLog の UI

本節では VDMPad-EpiLog の UI を説明する．図 6 に VDMPad-EpiLog の画面例を示す．VDMPad のメニュー中の VDMPad-EpiLog ボタンを押すと，従来の VDMPad の右側に構造化された履歴が表示される．ただし，ウェブブラウザの横幅が足りない場合には，従来の VDMPad の下部に表示される．構造化された履歴の表示部分を，本論文では EpiLog 画面と呼ぶことにする．

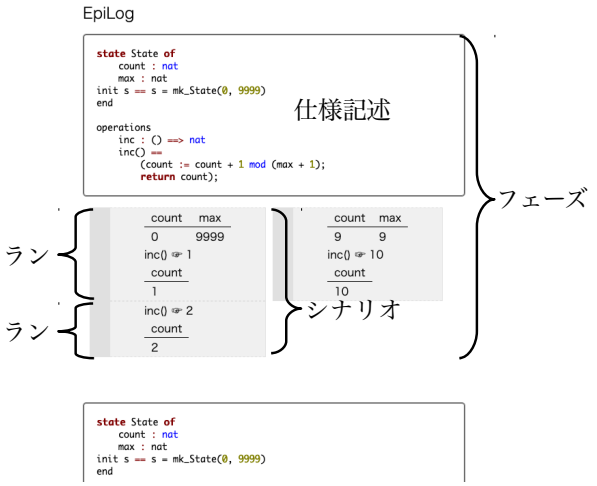


図 6. VDMPad-EpiLog の画面例

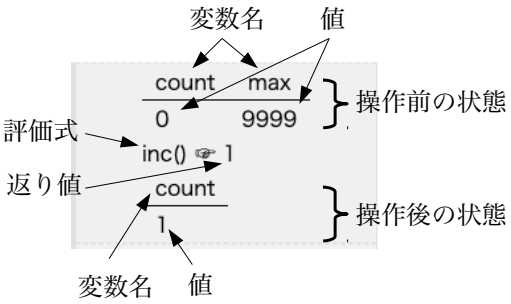


図 7. ランの表示例

EpiLog 画面には，フェーズごとに仕様記述とシナリオが表示される．各フェーズでは上段に黒枠で囲まれた仕様記述の下に，背景色がついたシナリオが列挙される．各シナリオには，左側に太い帯状の濃いグレーが 1 つながりで表示され，その右側にランが上から下に並べて表示される．シナリオ中のランとランの間は，薄い横線で区切られる．

各ランには，上段に実行前の状態，中段には評価実行する表現式とその返り値，下段には実行後の状態が表示される．図 7 にランの表示例を示す．水平に引かれた区切り線の上に変数名，下にその値が表示される．実行前の状態については，当該ランの直前のランの実行後の状態からの差分が表形式で表示される．シナリオ中の多くのランでは，操作前の状態は直前のランの実行後の状態

を引き継ぐため，差分がなく，表示は省略される．ただし，各シナリオの最初のランは，前のランがないため，実行前の状態がそのまま表示される．中段には，右方向を指差したシンボルの左側に表現式，右側にそれを評価した返り値が表示される．下段には，実行前の状態からの実行後の状態の差分が表形式で表示される．

フェーズ，シナリオ，ランに対する修正操作は，操作対象を選択して行う．操作対象を選択すると，その対象に対して実行可能な操作に対応したボタン列が表示されるので，ボタンを押すことで操作を実行することができる．フェーズの選択は，仕様記述の黒枠内をクリックして行う．シナリオの左側の帯をクリックすると，シナリオが選択される．帯の右側のランの上をクリックすると，ランが選択される．

3.3. VDMPad-EpiLog の使用例

本節では，VDMPad-EpiLog の典型的な使用例として，VDMPad 上でのカウンタの仕様記述の探索的試行の例を紹介し，その履歴を整理するステップを示す．記述するカウンタは，自然数のカウント値を 1 ずつ増加させていくものである．ただし，上限値が設定され，上限値を越えると 0 に戻るものとする．カウンタに対する操作として，カウント値を進める inc，カウント値を 0 にリセットする reset，上限値を設定する setMax の 3 つを提供するが，ここでは，まずはカウント値を進める inc を定義するための記述作業を対象とする．他の 2 つの操作 reset および setMax の定義は省略し，VDM-SL 仕様中に記述しない．

図 8 に示した画面では，2 つのフェーズが表示されている．最初のフェーズ (以後，フェーズ 1) では，暫定的な仕様記述を入力し，2 つのシナリオが検討された．最初のシナリオでは，inc を連続して 2 回実行して，カウント値が期待通りに 1 ずつ増加することを確認した．次に，カウント値が上限値を越える場合の動作を確認するために，カウント値 count と上限値 max にそれぞれ 9 を設定し，inc 操作を行った．すると，カウント値が 0 になることが期待されていたところが，10 となった．仕様記述者はこの時点での仕様記述に誤りがあることに気付いた．

仕様中の inc 操作の定義の中で，count + 1 mod (max + 1) とある式は，正しくは (count + 1) mod (max + 1) を意図していた．表現式 count + 1 mod

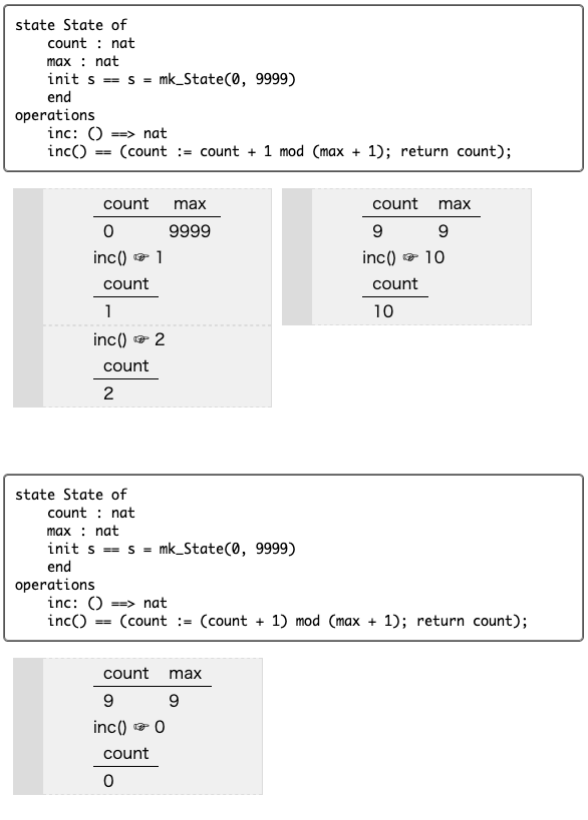


図 8. 整理開始時の画面例

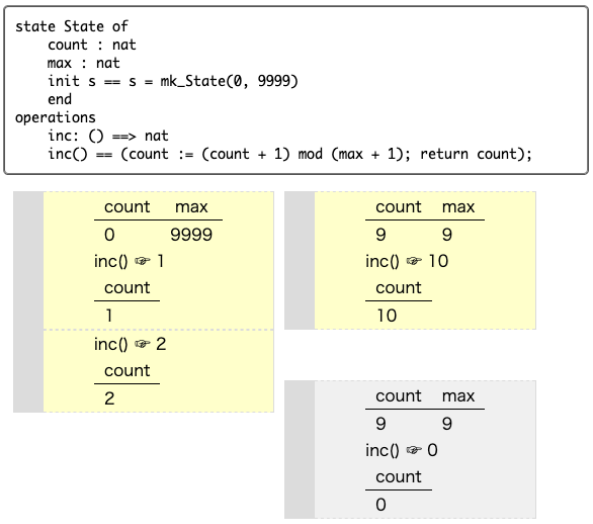


図 9. 軽微な修正のフェーズを合併した画面例

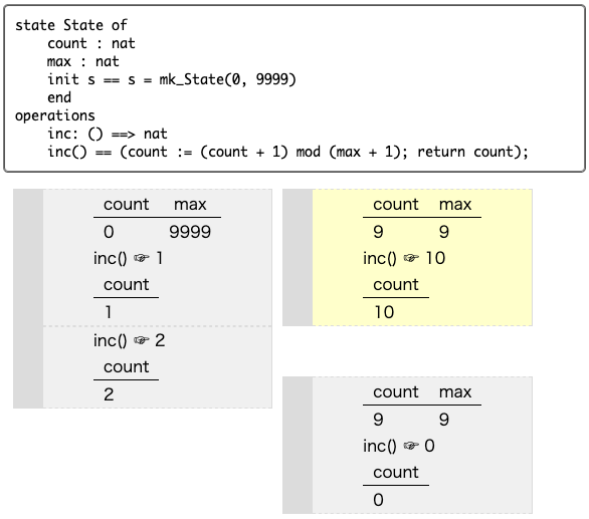


図 10. シナリオを再実行した画面例

(max + 1) は，演算子の結合順位により count + (1 mod (max + 1)) と解釈され，1 mod (max + 1) は常に 1 であることから，inc 操作は上限値に関係なく常にカウント値を 1 ずつ増加させる動作となっていた．仕様記述者はこの誤りに気付き，修正を行い，再びカウント値 count と上限値 max にそれぞれ 9 を設定し，inc 操作を行った．これが 2 つ目のフェーズ (以後，フェーズ 2) であり，実行結果は仕様記述者の意図通り，カウント値が 0 になった．

フェーズ 2 はフェーズ 1 からの軽微な修正であり，ここではフェーズ 2 を中心に探索的試行をまとめることにする．ただし，フェーズ 1 では初期状態から inc 操作を 2 つ行うシナリオを実行しているが，フェーズ 2 では実行していない．本来はフェーズ 2 でも実行すべきであるが，仕様記述者は上限値を超えた時の動作の修正を行っており，他のシナリオは念頭になかった．そこで，フェーズ 1 で検討したシナリオをフェーズ 2 にまとめるため，フェーズ 1 をフェーズ 2 に合併する．

図 9 に合併の結果を示す．合併の結果，フェーズは 1 つのみになった．このフェーズではフェーズ 2 にあった，誤り修正後の仕様記述が使われる．元のフェーズ 1 から合併で移動した 2 つのシナリオに含まれるランは，元の誤りを含む仕様記述で実行されたため，このフェーズの履歴としては正しくない．これら 2 つのシナリオの合わせて 3 つのランは無効であり，無効であることを示すために黄色を背景色にして表示されている．

無効なランは，新しい仕様記述の上で再実行することで，有効にすることができる．図 10 に，1 つ目のシナリオを再実行した結果を示す．初期状態から inc 操作を 2 回実行し，結果は元のフェーズ 1 での実行結果と同一であった．最初の探索的試行の段階では，誤りを含んだ仕様記述での実行を確認しただけであったところが，その履歴をまとめる作業を通して，修正後も同様に意図通りの動作をすることを確認することができた．

次に，カウント値が上限値を超えた場合のシナリオの再実行を行う．カウント値が上限値を超えた場合については，仕様記述を修正する前は意図した通りの動作をしなかった．図 10 では，元の意図した通りの動作ではなかったランが無効化された状態で表示されている．このランを再実行することで，この仕様記述では意図した通りの結果に置き換わることを確認することができる．再実行の結果を図 11 に示す．元のフェーズ 2 で実行されたシナリオと同一の結果であることが確認できる．

カウント値が上限値を超えた場合のシナリオは内容が同一であることから，冗長であり削除することにする．削除することで，一連の探索的試行の整理が完了する．履歴を整理した結果を図 12 に示す．この探索で到達した仕様記述と，その上で検討された 2 つのシナリオが並んで表示されている．1 つ目のシナリオは，初期状態から inc 操作を 2 回連続で行うもので，仕様記述者の意図

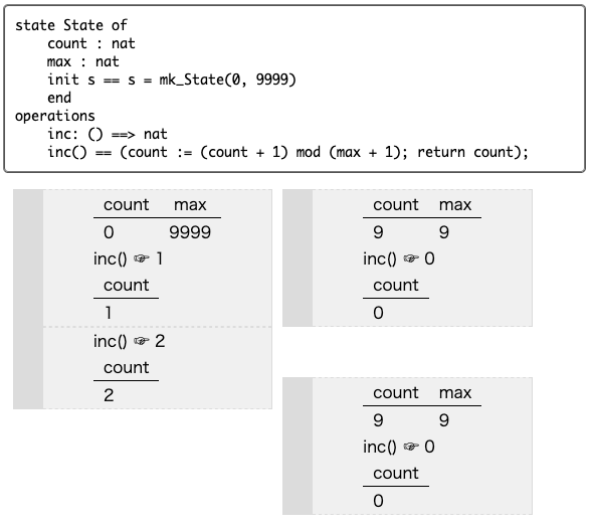


図 11. 全シナリオを再実行した画面例

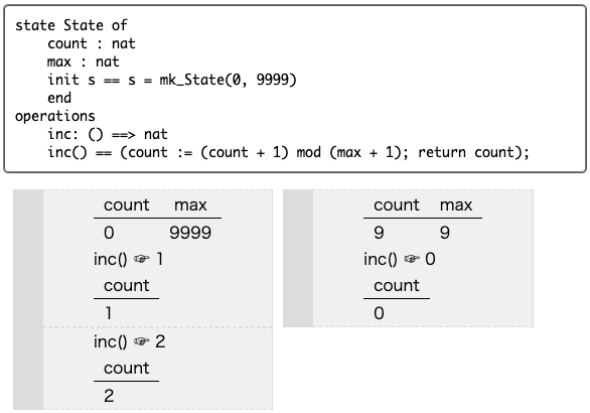


図 12. 重複シナリオを削除し整理を完了した画面例

通りの動作であることが確認された．2 つ目のシナリオは，カウント値が上限値を越える場合の `inc` 操作を行うもので，これも仕様記述者の意図通りの動作であることが確認された．これら 2 つのシナリオがこの仕様記述で想定された利用シナリオであり，それぞれユニットテストにおけるテストケースとして利用することができる．

4. 関連する既存ツール

ソフトウェア開発を支援するための工学研究では，開発の履歴の保存や集積，追跡性の確立は古くからの課題の 1 つである．現在ソフトウェア開発で多く利用されている統合開発環境では操作履歴に関する機能として undo/redo を提供するものは多いが，履歴に構造を与えて系統的にまとめることは行われていない．大きな粒度での開発の履歴の保存や集積，追跡性の確立は，コードレポジトリおよびそれと統合された開発チームホスティングサービスが提供している．github は現在オープンソースをはじめ広く利用されている開発チームホスティングサービスである．ソースコードレポジトリは分散バージョン管理システム git によりバージョン間の関係を有向グラフとして保存することができる．また，イシュー管理システムおよび修正提案 (pull request) から特定のバージョンへの参照を記録することができることから，あるバージョン差分の意図がどの問題を解決するためのもので，それがどの修正提案として提案され，誰によりソースコードレポジトリに取り入れたかを追跡可能にしている．このことから，github は開発コミュニティにおける長期間での開発の記録を整理し保存し，追跡性を確保した形で集積する機能を持つサービスと言える [6]．ただし，github でのバージョン管理，イシュー管理および修正提案は，開発の中での極めて大きな粒度での試行を捉えるものであり，基本的に個人間の円滑な連携や，新規参入者が文脈を理解しやすくするための仕組みである．本研究が扱うような，個人作業の内部でおこなう 1 つ 1 つの試行的な評価実行を対象にしたものではない．

ローカルな git レポジトリにも，バージョン差分を併合する機能があることから，個人作業での試行の履歴を整理することができる．しかし git レポジトリが扱う対象はソースコードであり，インタプリタ上あるいはコンパイルされた実行ファイルを動作させた履歴は扱わない．ユニットテスティングでのユースケースのバージョン差分を管理することで，各試行で検討したシナリオの変遷

を記録することが可能だが，本研究で扱う極めて短期間での試行とは，扱う試行の粒度が異なると考えられる．

5. 議論とまとめ

探索的な仕様記述で行われる探索的試行の履歴から，探索全体を整理し知見として残すための枠組みとして，フェーズ，シナリオ，ランによる構造化と，構造化された履歴への編集操作を提案した．具体的なツールとしてVDMPad-EpiLog を実装し，その利用例を示した．

VDMPad-EpiLog は VDM-SL 仕様を対象として扱うことを前提に設計されている．VDM-SL はモデル規範型の形式仕様記述言語であり，モデルが取りうる状態空間を厳密に規定する．状態空間が明確に規定されていることで，試行における評価実行の文脈を容易に保存することができる．

プログラミング言語でのユニットテスティングでもシナリオを再現するための状態設定を行うが，一般にプログラミング言語では状態空間が明示的でなく，多くの不確定要素を抱える．例えば，ファイル入出力は多くの場合にはバッファリング IO を通して，オペレーティングシステムのシステムコールが提供する共有リソースにアクセスする．他のプロセスとの相互作用も暗黙的に発生する可能性がある．これらはソースコードには明示的に記述されない部分であり，VDMPad-EpiLog が行うような細粒度での再実行を実現するのは容易ではない．シナリオが前提とする状態の再現の容易さと，それがもたらすシナリオに対する編集操作は，モデル規範型の形式仕様記述言語 VDM-SL の利点として挙げることができる．

今後の課題として，より複雑な仕様記述での探索的試行の支援が挙げられる．VDMPad は比較的小規模な仕様記述を想定して設計・実装されているため，複数モジュールに分割された複雑な仕様記述を扱うのに適していない．より複雑化した VDM 仕様を扱うために，VDMPad-EpiLog と同様な機構を ViennaTalk にも実装し，比較評価したい．

関連研究で挙げた科学研究向けノートブック環境への応用も考えられる．状態の再現性の高さ，および，操作実行の結果の検証の容易さは，再現性や検証可能性が要求されるノートブック環境の実現にも有用であることが期待できる．

6. 謝辞

本研究は，JSPS 科研費 (18K18033) および JSPS 科研費 (19K11911) の助成を受けた．有益なご指摘を頂いた査読者の方々に感謝する．

参考文献

[1] 小田朋宏, 荒木啓二郎 “形式仕様工程の初期段階に着目した統合仕様記述環境 ViennaTalk”, コンピュータソフトウェア, 34 巻, 4 号, ppp. 4_129-4_143, 日本ソフトウェア科学会, 2017.

[2] J. Fitzgerald and P. G. Larsen “Modelling Systems – Practical Tools and Techniques in Software Development”, Cambridge University Press, 1998.

[3] P. G. Larsen, N. Battle, M. Ferreira, J. Fitzgerald, K. Lausdahl and M. Verhoef “The Overture Initiative – Integrating Tools for VDM”, *SIGSOFT Softw. Eng. Notes*, Vol. 32, No. 1, pp. 1–6, 2010.

[4] P. Hamill “Unit Test Frameworks”, O ’Reilly, 2004.

[5] 小田朋宏, 荒木啓二郎 “アジリティのある探索的形式仕様記述のためのテストフレームワーク”, ソフトウェアシンポジウム 2018 論文集, 2018.

[6] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb “Social coding in GitHub: transparency and collaboration in an open software repository”, in Proceedings of the ACM 2012 conference on computer supported cooperative work, pp. 1277-1286, 2012.

行列計算に基づくモデル検査技術

熊澤 努
株式会社 SRA
kumazawa@sra.co.jp

小田朋宏
株式会社 SRA
tomohiro@sra.co.jp

要旨

モデル検査は，状態遷移系として与えられたソフトウェアのモデルが，望ましい性質を満たすことを形式的に検証する技術である．状態遷移系は，ソフトウェアシステムの振る舞いの記述に適しており，有向グラフで表現することが多い．有向グラフは隣接行列を使って表すことができるので，状態遷移系もまた行列で記述することが可能である．行列は，機械学習などの様々な分野での標準的なツールとして広く利用されている．実用上，*GPU* や分散環境上で大規模な行列の計算が扱われることも多く，演算のための様々なライブラリが整備されている．本論文では，行列計算を使用したモデル検査技法を提案する．提案する検査技法は，並行システムの検証を対象として，モデル検査の主要なアルゴリズムを行列計算で構成する．今後 *GPU* を使った高速化，高性能化が期待できる．本手法の実現可能性を評価するために，試作プログラムを作成してデッドロック検出問題に適用し，実行性能については改善が必要ではあるものの，全ての事例についてデッドロックが検出できることを確認した．

1. はじめに

モデル検査 [1, 2] は形式的検証技術の一つである．ソフトウェアやハードウェアで構成されたシステムの振る舞いや，通信プロトコルの正しさを自動検証するために，モデル検査が広く利用されている．ユーザが与えるモデル検査への入力は，システムの挙動を記述したモデルと，システムが満たすべき性質である．モデルは，クリプキ構造や有限オートマトンなど，有向グラフで表現可能な状態遷移系で記述されることが多い [2, 3]．一方，性質は時相論理式 [2, 3] などの論理式で表現される．モデル検査は，モデルが性質を満たすかどうかを網羅的に調べ，満たさない場合には反例と呼ばれる診断情報を返す．モデル検査には様々な技法が研究されているが，オートマトン理論を用いたモデル検査技法 [4, 5, 6] は，時相論理式を有限オートマトンに変換することで，検証を有向グラフの探索問題に帰着させる．

モデル検査は，並行システムの検証で有効であることが知られている [5, 6]．並行システムは，複数の自律動作するプロセスが相互に通信しながら計算を進める複合的なシステムである．その振る舞いは，各プロセスの挙動の組合せにより構成され，しばしば大規模かつ複雑な有向グラフで記述される．そのため，不具合の検出が難しいことが知られている．本論文では，並行システムの検証を対象として，オートマトン理論を用いたモデル検査技法に注目する．モデル検査の新しい技法として，行列計算を使用した検査アルゴリズムを提案する．

行列は，機械学習や科学技術シミュレーション，ソーシャルネットワークの解析など様々な分野で使われている [7]．例えば，ソーシャルネットワーク解析では，ユーザ間の関係をグラフ構造で表すことが多い．一般にグラフは隣接行列で表すことができるので，ソーシャルネットワークの解析の際に隣接行列が利用される [8]．実際の応用では，しばしば数千万次から数億次に至る規模の行列が扱われる．近年は，GPU を搭載した高性能計算機が手に入れやすくなったことに加えて，高速な行列計算ライブラリを気軽に利用できるようになったことで，大規模な行列の計算を計算機で実行することが可能である．

オートマトン理論を用いたモデル検査は，モデルや性質を有向グラフで表現しその探索問題として解くことができる．本論文では，有向グラフを隣接行列で表し，モデル検査アルゴリズムを隣接行列の演算によって構成す

る．各プロセスの振る舞いを結合したグラフを構成する同期積演算を，行列のクロネッカー積により実現する．探索と反例の生成には，代数的グラフアルゴリズム [9] を使用する．さらに，モデル検査の一手法として，デッドロックの検出も行列演算で構成する．

提案手法の実現可能性を確認するために，行列計算を用いた検証プログラムを試作し，評価実験を行った．実験では，複数のプロセスから構成される並行システムのデッドロック検出問題を対象として，いくつかの異なる規模について，提案手法を用いてデッドロックの検出を行った．その結果，実行速度，メモリ消費の点では改善が必要ではあるものの，全ての問題でデッドロックを検出することができた．

本手法の利点として，モデル検査アルゴリズムを代数的に記述することができることが挙げられる．加えて，行列計算は，GPU 上での実行や分散環境での並列化の実現が期待できる．従来は，モデル検査に特化した並列化や，特定のプロセッサ向けのアルゴリズムが研究されることが多かったが，数値計算や高性能計算分野での研究成果を取り入れた高性能化も可能であると思われる．

本論文の構成は次の通りである．2 節では，関連する先行研究について議論する．3 節で，技術的な背景として，モデル検査の概要を簡単な例を使いながら説明する．4 節で，提案する行列計算を用いたモデル検査手法の詳細を述べる．加えて，同じ技術により実現可能なデッドロック検出技法についても説明する．5 節では，提案手法を実装した試作プログラムの概要を述べ，試作プログラムを用いて行った実験の結果を議論する．6 節で結論と今後の課題を述べる．

2. 関連研究

従来のモデル検査は，記号的モデル検査とオートマトン理論を用いたモデル検査に大きく分類される．記号的モデル検査 [2] は，ソフトウェアの挙動を，順序付き二分決定グラフに代表されるように論理式で記述して検証を行う技法である．一方，オートマトン理論を用いたモデル検査 [4, 10, 2] は，ソフトウェアの挙動を有限オートマトンで明示的に表現し，その受理可能性を判定する．有限オートマトンは有向グラフとみなせるので，効率的なグラフ探索アルゴリズムを使用できる点に特徴がある．加えて，On-the-fly 検査法や半順序簡約法などの，検査の効率を向上させる最適化技術を組込むことで，検査系の実用化に成功している [2, 3]．Spin [5] や LTSA [6] が代表的なモデル検査系である．GPU 上で高速に検査をするための，オートマトン理論を用いた検査アルゴリズム [11] や，On-the-fly 検査法 [12]，半順序簡約法 [13] などの技術も提案されている．オートマトン理論を用いた既存のモデル検査は，ソフトウェアの挙動を有限オートマトンとしてモデル化し，有向グラフ上を網羅的に探索することで，ソフトウェアの挙動に関する性質を検証する技術である．ソフトウェアの挙動を有限オートマトンでモデル化する点が本論文で提案する手法と共通しているが，既存技術が有向グラフ上の探索アルゴリズムを利用するのに対して，提案手法は，隣接行列で表現して，行列演算を用いて検証する点が異なる．

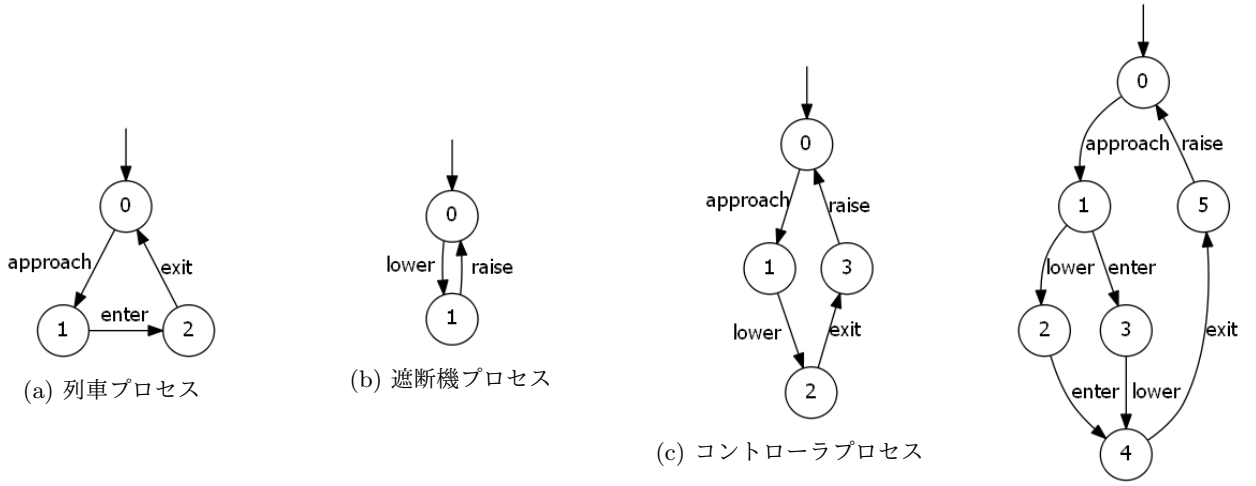
ソフトウェアの検証問題を既存のソルバーを使って解くことで，高速化，効率化を図るアプローチも積極的に研究されている．Alloy [14] や有界モデル検査 [15] は，検証問題を充足可能性判定問題に帰着させ，高速な SAT ソルバーを活用して検証を行う．

確率的モデル検査 [3] は，マルコフ連鎖やマルコフ決定過程を対象とした検証技術である．これらは推移確率行列や状態遷移系で表現できることが知られており，行列計算によりシステムの確率的な挙動の検証が可能である．Wijs 等は，確率的モデル検査を GPU 上で実行する際の効率的な疎行列とベクトルの演算技法を提案した [16]．文献 [17] では，複数のサブシステムから成る複合システムの確率的モデル検査法が提案されている．

3. オートマトン理論を用いたモデル検査

本節では，2 節で紹介したオートマトン理論を用いたモデル検査のうち，Giannakopoulou と Magee による技法 [10] の概要を，簡単な例を使って紹介する．

本論文では，ラベル付き遷移系（LTS） [6] を用いた状態遷移系によって，システムの振る舞いを記述する．LTS は，状態の集合，状態間の遷移関係，初期状態で構成される．遷移関係には，アクションあるいはイベントと呼ばれるラベルが付与されている．初期状態から状態遷移を繰り返すことで，システムが起こすイベントの列を表現できる．図 1 に踏切の遮断機制御システム [3] を LTS で記述したモデルを示す．遮断機制御システムは列車，遮断機，コントローラの 3 つのプロセスから成る．列車プロセス（図 1a）は，踏切付近の列車の動きをモデル化したプロセスである．図 1a で，丸が状態，矢印



(a) 列車プロセス　(b) 遮断機プロセス　(c) コントローラプロセス　(d) システム全体の挙動

図 1. 踏切の遮断機制御システム

が遷移関係，approach などのラベルがイベントである．初期状態は，遷移元のない矢印が指す状態 0 である．このモデルは，列車が踏切に近づき (approach)，次に進入して (enter)，退場する (exit) という一連の挙動をイベントで表現している．遮断機プロセス（図 1b）は，バーを下ろす (lower)，上げる (raise) という挙動を示す．コントローラプロセス（図 1c）は，列車が踏切に近づいたらバーを下ろすように，また，列車が踏切を出たらバーをあげるように，遮断機に命令を発行する．

個々のプロセスの同期積 [5] による振る舞いの組み合せが，システム全体としての振る舞いである．LTS の場合，プロセス同士の並列結合 [6] によって，システムの振る舞いを表現する．図 1d に遮断機制御システムの全体の振る舞いを示す．2 つのプロセスの並列結合演算の概要は次の通りである．

- 2 つのプロセスに共通するイベントが付与された遷移については，両プロセスは同時に遷移する．この演算はプロセス間の同期機構である．例えば，列車プロセスとコントローラプロセスにおいて，approach と exit が共通するイベントであり，どちらのプロセスも状態 0 に位置する場合にのみ，共に状態 1 に遷移する．これは，図 1d の状態 0 から 1 への遷移に相当する．各プロセスの状態の順序対を，この同期機構による結合後の状態と考えることもできる．この見方の下では，図 1d の状態 0 と 1 は，それぞれ状態 $(0,0)$ と状態 $(1,1)$ とみる．

- 2 つのプロセスで共通には現れないイベントが付与された遷移については，各プロセスが任意の順序で交互に遷移する（インターリービングといわれる）．この演算は各プロセスの非同期実行に該当する．例えば，列車プロセスとコントローラプロセスの状態 1 からの遷移を考えると，イベント enter と lower は両プロセスで共有されないので，その順序は任意である．これは，図 1d の状態 1 から 4 への遷移のうち，2 を経由した場合と 3 を経由した場合によって表現される．上と同様に列車プロセスの状態とコントローラプロセスの状態の順序対を考えると，それぞれ，状態 $(1,1)$ から状態 $(1,2)$ を経由して状態 $(2,2)$ に達する場合と，状態 $(1,1)$ から状態 $(2,1)$ を経由して状態 $(2,2)$ に達する場合を表す．これは，イベント lower が発生する場合には，列車プロセスは同一状態にとどまってコントローラプロセスのみが遷移し，逆にイベント enter が発生する場合には，列車プロセスのみが遷移するとみることもできる．

この例のような，複数のプロセスで構成される並行システムは，システム全体としての挙動が複雑になり，分析や検証が難しい．本論文では，このような特徴を持つ

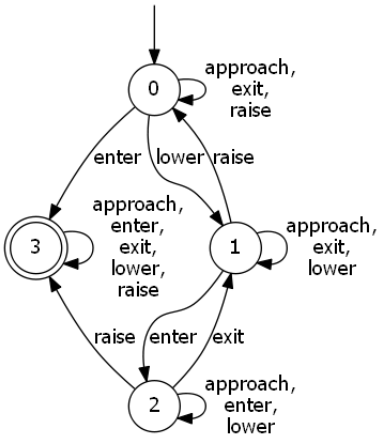


図 2. 性質の Büchi オートマトン

並行システムの検証を対象とする．図 1d のモデルを検証するために,「列車が踏切に入るならば，出ていくまで遮断機は下りている」という性質が成り立つかを考える．性質は時相論理式で形式的に記述されるが，オートマトンを用いたモデル検査では，論理式で書かれた性質を Büchi オートマトン [2] に変換する．Büchi オートマトンは受理状態を無限回通過する無限語を受理する有限オートマトンである．性質「列車が踏切に入るならば，出ていくまで遮断機は下りている」を表現した Büchi オートマトンを図 2 に示す．図 2 の二重丸は受理状態である．このオートマトンは，性質を満たさないイベント列を受理するように構成されている．例えば，遮断機のバーが下がる (lower) 前に列車が踏切に進入する (enter) イベント列は，状態 0 から 3 に遷移した後に，状態 3 を繰り返し通過するために受理されるが，性質に反する．

次に，図 1d と図 2 の並列結合によって，性質に違反するシステムのイベント列のみを受理するオートマトンを構成する (図 3)．このオートマトンの受理可能性を判定し，受理可能な場合には，受理するイベント列を反例として返す．受理可能性の判定には，受理状態を含む閉路探索を実行する．探索には，Tarjan のアルゴリズム [18] などの有向グラフの強連結成分 (SCC) の探索アルゴリズムや二重深さ優先探索法 [5, 2] が用いられる．SCC とは，与えられた有向グラフの部分グラフで，遷移を繰り返し辿ることで，その任意の 2 状態間が互いに到達可能であるものをいう．図 3 では，初期状態 0 から到達可能な受理状態 5，6，7，8，9 から成る SCC が存在する

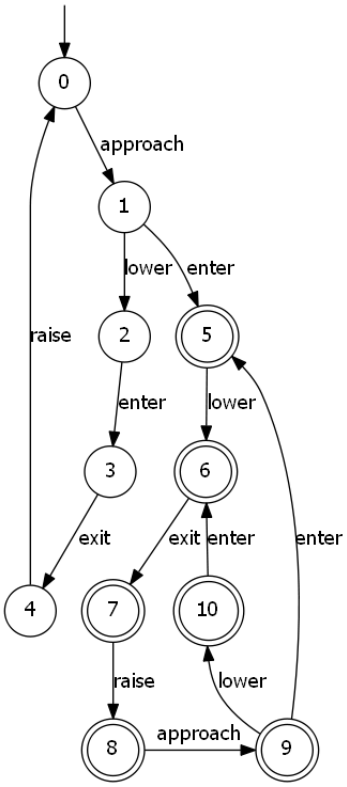


図 3. 並列結合オートマトン

ため，性質に違反すると判定される．よって，反例は，lower, exit, raise, approach, enter から成る閉路と，その閉路へのイベント列 approach, enter である．この反例は，遮断機制御システムには，遮断機のバーが閉まる前に列車が踏切に進入する恐れがあることを示している．

4. 行列計算に基づくモデル検査

本節では，提案する行列計算に基づいたモデル検査技法を詳しく述べる．また，その応用として，並行システムで問題となるデッドロック検出技法についても述べる．

4.1. 隣接行列による状態遷移の表現法

提案するモデル検査技法は，3 節で説明したオートマトン理論を用いたモデル検査に基づく．LTS や Büchi オートマトンを有向グラフとみなして，それぞれ隣接行列で表し，それらの演算によってアルゴリズムを構成する．なお，本節での行列の要素は真理値とみなすことと

し，要素の積と和は，それぞれ論理積と論理和とする．

隣接行列は状態間の遷移関係を記述した正方行列である．その (i,j) 要素は，状態 i から状態 j への遷移が存在する場合には 1，存在しない場合には 0 である．また，i 行目の行ベクトルは状態 i から出ていく遷移を，j 列目の列ベクトルが状態 j に入る遷移を表す．例えば，状態に与えた番号に 1 を加えた値を行番号として，イベントを無視すると，図 1a の隣接行列は次の通りである．

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

隣接行列を用いた検査アルゴリズムの特に重要な構成要素は，次の 3 点である．

1. LTS や Büchi オートマトンのようなイベントラベルのある有向グラフに対する隣接行列表現の構成法
2. 並列結合演算
3. 閉路の検出法と反例の生成法

第 1 の隣接行列表現の構成では，遷移に与えられているイベントの扱いが重要である．そこで，イベントごとに隣接行列を構成する．列車プロセス（図 1a）を例にとると，次の 3 つの行列は，左からそれぞれイベント approach，enter，exit についての隣接行列表現である．

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

列車プロセスにおけるイベント lower のようなプロセスに出現しないイベントについては，単位行列 I を行列表現とする．単位行列は，各状態が自身への遷移，つまり，自己閉路のみを持つことを意味する．これは，非出現イベントではプロセスが他の状態に遷移せず，同一状態にとどまることと解釈できる．

第 2 の並列結合演算は，それぞれのイベントについての隣接行列同士のクロネッカー積で実現する．ここで，$m_1 \times m_2$ 行列 $A = (a_{ij})$ と $n_1 \times n_2$ 行列 $B = (b_{ij})$ のクロネッカー積 $A \otimes B$ は次式で定義される $m_1 n_1 \times m_2 n_2$ 行列である (一般の定義は [19] に詳しい)．

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m_2}B \\ \vdots & \ddots & \vdots \\ a_{m_1 1}B & \dots & a_{m_1 m_2}B \end{pmatrix} \qquad (1)$$

すなわち，$a_{ij}b_{kl}$ はクロネッカー積の $((i-1)n_1+k, (j-1)n_2+l)$ 要素である．このことは，状態 i から状態 j および状態 k から状態 l に共通イベントで遷移する場合，並列結合した LTS には状態 $(i-1)n_1+k$ から $(j-1)n_2+l$ の遷移が存在すると解釈できる．順序対 (i,k) は $(i-1)n_1+k$ と，順序対 (j,l) は $(j-1)n_2+l$ とそれぞれ 1 対 1 に対応するから，順序対で表した並列結合の状態 (i,k) から状態 (j,l) への遷移に対応するとみなしてよい．例として，図 1b と図 1c の LTS を考えると，両プロセスで共通するイベント lower に関しては，次のようにクロネッカー積を計算できる．

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

遮断機プロセスの状態 0 から状態 1 への遷移と，コントローラプロセスの状態 2 から状態 3 への遷移は，並列結合の状態 1 から状態 6 への遷移になることを意味する．

2 つの LTS に共通には現れないイベントの場合には，隣接行列の構成法から，式 1 の A または B は単位行列 I である．$A = I$ とすると，対角成分 $a_{ii} = 1$ で他は 0 なので，クロネッカー積 $A \otimes B$ の $((i-1)n_1+k, (i-1)n_2+l)$ 要素が b_{kl} で，他の要素は 0 である．これは，隣接行列 B で表されるプロセスのみが遷移する場合，すなわち，順序対で表した状態 (i,k) から状態 (i,l) への遷移に相当する．$B = I$ の場合も同様である．

SCC の検出を行う際には，探索空間に相当する有向グラフがあればよいので，各イベントを除いた単一の隣接行列を構成する．そのために，並列結合により得られた各イベントに対応する隣接行列の和を求める．

第 3 の閉路検出と反例生成については，行列計算を使用した探索技法である代数的グラフアルゴリズム [9] を使用する．モデル検査の実現のために，SCC の検出アルゴリズム [9] と幅優先探索法（BFS） [8, 9] を使用する．

SCC の検出アルゴリズム [9] は次の手続きから成る．並列結合オートマトン（図 3 参照）の n 次隣接行列を A，n 次単位行列を I として，$C = (A + I)^{n-1}$ を計算する．ここで，行列 $A + I$ の (i, j) 要素は，状態 i から状態 j に高々1 回遷移を辿って到達可能であれば 1，不可能であれば 0 である．同様に，C の (i, j) 要素は，状態 j が，状態 i から高々 $n - 1$ 回遷移を辿ることで到達可能ならば 1，不可能であれば 0 である．したがって，C は各状態から到達可能な状態の集合を情報として持つことに注意する．次に，C と C の転置行列 C^T の要素ごとの積（アダマール積）$S = C \circ C^T$ を計算する．C^T の (i, j) 要素は，状態 i が状態 j から到達可能ならば 1，不可能であれば 0 である．よって，行列 D は，状態 i と j が互いに到達可能な場合は 1 で，そうでない場合は 0 になるような (i, j) 要素から成る n 次対称行列である．S の各行ベクトルで，値が 1 の成分の集合が SCC を構成する．

検出した SCC に受理状態が含まれていた場合には，反例を生成する．反例は，検出した SCC と，初期状態からその SCC への路とで構成される．従来のモデル検査では，性能上有利であるとされる深さ優先探索（DFS）により探索が行われることが多い．しかしながら，隣接行列表現に対しては，BFS は行列の積により容易に実現が可能だが，DFS の実現は難しい．一方で，BFS で探索した路は最短路となるため，ユーザにとって理解しやすい反例を求めることができる [20]．そこで，初期状態から SCC への路を生成するために，SCC に含まれるいずれかの状態までの路を BFS により探索する．行列計算を用いた BFS の概要は次の通りである [9]．まず，探索の開始状態を行ベクトルで表現する．簡単のため，図 3 のように初期状態を 0 とすると，第一成分のみ 1 で他の $n-1$ 成分が 0 の単位行ベクトル $\boldsymbol{x}_0 = (1, 0, \ldots, 0)$ である．次に，$\boldsymbol{x}_1 = \boldsymbol{x}_0 A$ を計算する．$\boldsymbol{x}_1$ は，状態 0 から遷移を 1 回辿って到達可能な状態に相当する列のみ 1 となる行ベクトルである．同様に，i 回の探索結果 $\boldsymbol{x}_i$ に対して，$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i A$ を算出すると，$i+1$ 回遷移を辿ることで到達する状態の集合が得られる．探索の冗長性を除くために，$\boldsymbol{x}_{i+1}$ から i 回の遷移で既に到達した状態の集合を除いた後に，この手続きを繰り返せばよい．

4.2. デッドロックの検出

複数のプロセスから成る並行システムの場合には，プロセス間の相互作用の不備により，全てのプロセスが停止してしまうデッドロックが起こりうる．デッドロックは並行システムで発生する代表的な不具合の一つであり，多くのモデル検査系に検出機能が備わっている．ここでは，行列計算を用いたデッドロック検出技法を述べる．

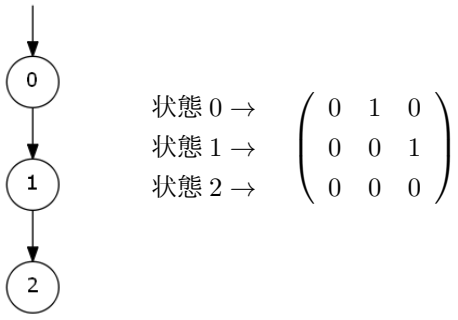


図 4. デッドロック状態（状態 2）のある LTS とその隣接行列表現

LTS で記述されたシステムでは，デッドロックは，各プロセスを並列結合した LTS においてイベントが発生しない状況を指す．よって，デッドロックの検証には，性質を記述した Büchi オートマトンは不要である．デッドロックを検出するには，初期状態から LTS の各状態を網羅的に探索して，出遷移のない状態（デッドロック状態）を検出する．隣接行列表現の場合には，第 i 行を取り出した行ベクトルが零ベクトルならば，対応する状態 i は出遷移が存在せず，デッドロック状態である．例えば，図 4 において，状態 2 はデッドロック状態である．隣接行列表現では，状態 2 に対応する第 2 行の行ベクトルが零ベクトルとなることがわかる．そこで，初期状態から BFS を実行して，到達した状態の行ベクトルが零ベクトルかどうかを判定し，零ベクトルならば，その状態でデッドロックが発生したと判断する．デッドロックを検出した際には，BFS で探索した初期状態からデッドロック状態への路を反例として返す．

5. 評価実験

提案手法の実現可能性を調べるために，Python 言語を用いて試作プログラムを実装し，評価実験を行った．Python は高性能な数値計算用パッケージが充実し，広く利用されていることから，実現可能性の検証に適していると判断した．Python のバージョンは 3.6.8 を利

用し，データ分析や機械学習の領域で広く使われている科学技術計算向けパッケージである SciPy [21] および NumPy [22] を使用した．

5.1. 評価用プロトタイプの実装

行列を扱うためのデータ構造には，NumPy の numpy.ndarray や numpy.matrix が知られている．これらのデータ構造は隣接行列の全要素の値を格納するので，状態数の 2 乗のオーダーのメモリが必要である．そのため，大規模な行列の計算には適していない．そこで，試作プログラムでは，SciPy の疎行列パッケージ scipy.sparse を用いた．疎行列とは，要素の多くが 0 であるような行列である．行列計算を効率化するために，scipy.sparse に実装されたデータ構造は，非零要素のみをメモリ上に保持する．したがって，隣接行列が疎行列ならば，scipy.sparse が適切なデータ構造である．実際，5.3 節で示すように，実験で使用したモデルは全要素数に対する非零要素数の割合が非常に小さいので，疎行列を使用するのは妥当であると考えられる．提案手法では，行列の要素の値は 0 または 1 であり，それらの値の演算には論理演算のみ使用するので，行列の値は Python の真理値型とした．また，並列結合演算を実現するために必要な疎行列のクロネッカー積は，scipy.sparse パッケージの演算ライブラリを使用した．

試作プログラムの入力は，システムを記述した LTS と，性質を記述した Büchi オートマトン，検証項目である．なお，システムが複数のプロセスで構成される場合には，それぞれを記述した LTS を入力とした．3 節で取り上げた遮断機制御システムの例の場合，入力する LTS は図 1a，図 1b，図 1c であり，Büchi オートマトンは図 2 である．LTS と Büchi オートマトンの入力には，LTS 用のデータフォーマットの一つである Aldebaran フォーマット [23] を用いた．検証項目は，性質の Büchi オートマトンを用いたモデル検査，あるいは，デッドロック検出のいずれかであり，それぞれ 4.1 節，4.2 節で述べた検査を実行する．反例を検出した場合は，試作プログラムは反例をイベント列で返し，そうでない場合は Python の組込み定数 None を返すこととした．

5.2. 評価に用いた例題

実験では，食事をする哲学者問題 [24] を採り上げた．この問題は，円卓を囲んで食事をする哲学者の人数に応じた数のプロセスから成る並行システムの排他制御を扱った問題である．文献 [24] や文献 [6] で具体的なモデルの例と共に詳しく議論されている．文献 [6] には，排他制御が正しく行われていないことが原因でデッドロックが発生する事例が，LTS の例と共に掲載されている．実験の際には，哲学者の人数を変更しながら，この事例の LTS をモデル検査系 LTSA [6] で生成した．その後，LTSA に備わっている Aldebaran フォーマットへの変換機能で，LTS を試作プログラムへの入力フォーマットに変換した．そして，提案手法を用いてデッドロックの検出を行った．図 5 と図 6 に使用した LTS を示す．図 5 は i 番目の哲学者プロセスで，席に着くと，左右の二つの食器を取って食事をした後に，立ち上がるという挙動を繰り返す．図 6 は補助プロセスである食器を表してい

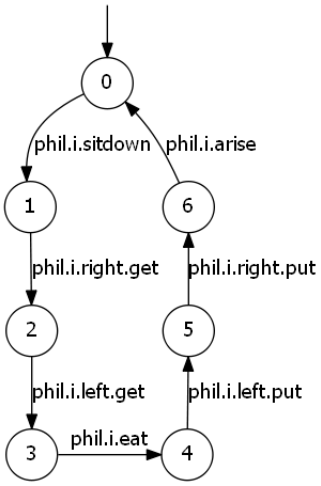


図 5. 哲学者 i プロセス

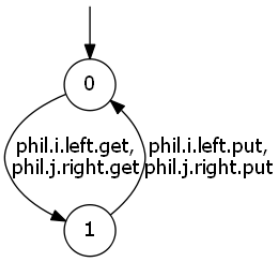


図 6. 食器 i プロセス．図中の j は，N を哲学者の人数として，$j = (i + 1) \mod N$ である．

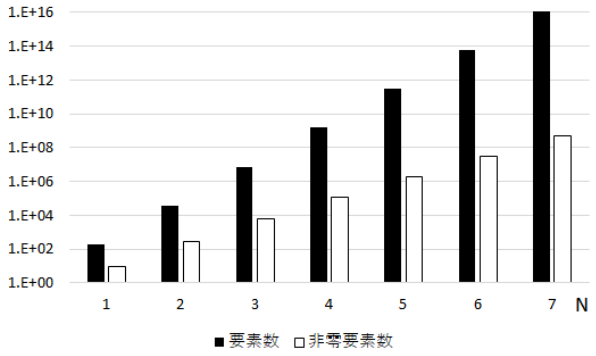


図 7. 哲学者の人数 ($N = 1 \ldots, 7$) に対する隣接行列の要素数と非零要素数

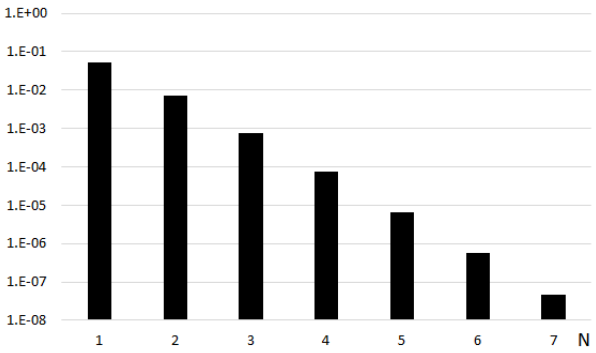


図 8. 哲学者の人数 ($N = 1 \ldots, 7$) に対する隣接行列の非零要素数の割合

る．食器は，その両側に座っている哲学者の一方にのみ使用される．本事例は，n 人の哲学者に対して，n 個の食器があるために $2n$ 個のプロセスで構成されている．

5.3. 評価結果 1: 行列表現の所要メモリ

最初に，試作プログラムで，食事をする哲学者問題の状態空間を隣接行列で生成して，その疎性を調べた．図 7 に，哲学者の人数を 1 から 7 まで変えた場合について，隣接行列の要素数と非零要素数を示す．図 8 には非零要素数に対する要素数の割合を示す．これらの結果から，人数が増加するにつれて非零要素の割合が指数関数的に減少しており，疎性が高くなる傾向を読み取ることができる．表 1 に，隣接行列とその疎行列それぞれについて，メモリ消費量の測定値を示す．隣接行列

表 1. 哲学者の人数 ($N = 1 \ldots, 7$) に対する隣接行列とその疎行列表現のメモリ消費量

N	隣接行列 [byte]	疎行列表現 [byte]
1	196	110
2	3.84×10^4	2.19×10^3
3	7.53×10^6	4.04×10^4
4	1.48×10^9	7.02×10^5
5	-	1.18×10^7
6	-	1.91×10^8
7	-	3.06×10^9

については numpy.ndarray を測定し，疎行列については圧縮行格納形式 scipy.sparse.csr_matrix を測定した．scipy.sparse.csr_matrix は，非零要素値の配列（data），非零要素の列番号を格納する配列（index），列番号格納配列において各行の開始位置を格納する配列（indptr）によって各要素を参照するので，それらの所要メモリの合計値とした．表 1 の「-」は，メモリ不足のために Python の例外 MemoryError が発行され，測定ができなかったことを示す．表 1 からも，食事をする哲学者問題では，疎行列が妥当なデータ構造であることがわかる．

5.4. 評価結果 2: 所要時間と実行時消費メモリ

次に，試作プログラムを用いて，哲学者の人数 N を 1 から 7 まで変更しながら，食事をする哲学者問題に含まれるデッドロックの検出を行い，実行時間と検証実行時の最大消費メモリ量を測定した．実験で使用した計算機は，Intel Core i7-8700，CPU 3.20GHz，RAM 16.0 GB を備えた HP ProDesk 600 G4 SFF である．また，OS には Windows 10 Pro を使用した．

表 2 に提案手法の試作プログラムが検出した反例の長さ，並びに試作プログラムの性能を測定した結果を示す．同じ表に，同じ LTS のデッドロックを LTSA で検出した結果も示す．どちらも 5 回実行した平均値をとった．LTSA はスタンドアローンで動作するモデル検査系のため直接性能を測定することが困難である．そのため，結果については，実行時のログに出力された解析時間とメモリ消費量から算出した．本実験の目的は性能の比較ではなく，提案手法の実現可能性の評価なので，LTSA の結果は参考値である．

表 2. 哲学者の人数 ($N = 1 \ldots, 7$) に対する提案手法と LTSA の反例の長さと性能の測定結果

	提案手法			LTSA		
N	反例長	実行時間 [sec]	最大メモリ消費量 [MiB]	反例長	実行時間 [sec]	最大メモリ消費量 [MiB]
1	2	0.042	93.1	2	0.0002	11.4
2	4	0.092	93.7	4	0.000	11.5
3	6	0.154	94.4	6	0.000	11.8
4	8	0.154	100.0	8	0.0002	12.8
5	10	0.624	182.1	10	0.0014	14.1
6	12	5.317	1547.8	12	0.008	21.1
7	14	2968.4	2232.1	14	0.063	24.2

実験の結果，実験したすべてのモデルについて，デッドロックの検出に成功した．また，LTSA と同様に，デッドロックに至る最短のイベント列を反例として返した．例として，$N = 3$ の場合について，反例を次に示す．

phil.0.sitdown
phil.0.right.get
phil.1.sitdown
phil.1.right.get
phil.2.sitdown
phil.2.right.get

哲学者は phil.0，phil.1，phil.2 の 3 人である．上の反例は，各哲学者が席に着いて自分の右側の食器を取ったところで食卓に食器がなくなり，デッドロックに陥ることを表している．N が他の値の場合でも，検出した反例は同様の手順でデッドロックとなった．

最後に，性能評価の結果を論じる．表 2 から，現状の試作プログラムは，実行速度，実行時メモリ消費量のいずれの効率性の点でも LTSA には及ばないことがわかる．特に，$N = 7$ になると，提案手法の実行時間が急激に悪化している．これは，主として並列結合の性能の問題である．クロネッカー積によって並列結合を実現すると，モデル検査では不要な，初期状態から到達不可能な状態も生成されるため，行列の規模が巨大化する．例えば，3 節で扱った踏切の遮断機制御システムでは，提案手法で構成される並列結合オートマトンの実際の隣接行列の次数（状態数）は，クロネッカー積の定義から，図 1a，図 1b，図 1c，図 2 の状態数の積 96 となるが，図 3 より初期状態から到達可能な状態数は 11 である．本実験の場合，$N = 7$ のときの隣接行列の次数が約 1×10^8 と大規模となり，計算機の CPU の性能では扱うことが困難になったと考えられる．このことは，モデル検査の主要な研究課題である状態爆発問題 [2] が発生したと解釈できる．試作プログラムには状態爆発問題への対策されていないため，$N = 7$ で性能の劣化が著しくなったと思われる．今後，さらに大規模なシステムとその隣接行列を扱えるようにするためには，使用するデータ構造を工夫する，到達不可能な状態を除去する機構を実装する，などの対処をする必要がある．

6. おわりに

本論文では，線形代数，特に行列計算を用いてモデル検査技法を構成した．従来のオートマトン理論を用いたモデル検査は，有向グラフの探索技術に重点を置いている．本論文では，グラフを隣接行列で表すことで，行列演算を用いて検査に必要なアルゴリズムを構築した．モデル検査は大規模なグラフから成る探索空間を扱うことが多いが，機械学習などの技術の進展とともに，大規模行列の演算を計算機で実行できるようになってきた．また，安価な GPU の普及により，行列演算を用いたモデル検査の更なる高性能化も期待できる．デッドロック検出問題を対象として，提案手法の実現可能性について評価実験を実施した結果，実験に使用した問題に含まれるデッドロック，並びに，既存のモデル検査系と同程度の長さの反例を検出することができた．したがって，提案手法は実現が可能であると考えられる．一方で，性能面では，状態爆発が発生し，既存のモデル検査系には及ばないという結果となった．

今後取り組むべき課題は二点ある．第一に，Büchi オートマトンを使ったモデル検査の評価実験を行う．この場合，強連結成分の検出を行う必要があるので，計算時間

とメモリ消費量の双方がデッドロック検出よりも増大することが予想される．第二に，検査性能の改善を行う必要がある．本論文の評価実験は CPU 上での実現性の確認にとどまっている．そこで，GPU を搭載した計算機や分散環境での検査の実現に向けて，並列化などの高性能計算分野の成果 [25] を活用を進める．GPU 上で実行する On-the-fly 検査法 [12] や半順序簡約法 [13] などのモデル検査の最適化技術を行列計算に取り入れる．クロネッカー積の効率的な計算には，文献 [26] で研究されている科学技術計算向けの汎用高速計算技法を組込むことも有効であると考えられる．

謝辞

有益なご指摘を頂いた査読者の方々に感謝する．

参考文献

[1] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, 1981.

[2] Edmund Clarke Jr., Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model Checking*. MIT Press, 2nd edition, 2018.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[4] Moshe Y. Vardi and Pierre Wolper. An automata theoretic approach to automatic program verification. In *Proceedings of First Symposium on Logic in Computer Science*, pages 332–344, 1986.

[5] Gerard Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2003.

[6] Jeff Magee and Jeff Kramer. *Concurrency: State Models & Java Programs*. John Wiley & Sons, Inc., 2nd edition, 2006.

[7] 櫻井鉄也. 巨大行列はなぜ重要か. **数学セミナー**, 59(2):8–11, 2020.

[8] 藤澤克樹. 巨大行列とグラフ解析. **数学セミナー**, 59(2):24–28, 2020.

[9] Kayhan Erciyeş. *Guide to Graph Algorithms: Sequential, Parallel and Distributed*. Springer, 2018.

[10] Dimitra Giannakopoulou and Jeff Magee. Fluent model checking for event-based systems. In *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 257–266, 2003.

[11] Jiri Barnat, Petr Bauch, Lubos Brim, and Milan Češka. Designing fast LTL model checking algorithms for many-core GPUs. *Journal of Parallel and Distributed Computing*, 72(9):1083–1097, 2012.

[12] Ezio Bartocci, Richard DeFrancisco, and Scott A. Smolka. Towards a GPGPU-parallel SPIN model checker. In *Proceedings of the 2014 International SPIN Symposium on Model Checking of Software*, pages 87–96, 2014.

[13] Thomas Neele, Anton Wijs, Dragan Bošnački, and Jaco van de Pol. Partial-order reduction for GPU model checking. In *Automated Technology for Verification and Analysis*, pages 357–374, 2016.

[14] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2006.

[15] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems, Lecture Notes in Computer Science*, volume 1579, pages 193–207, 1999.

[16] Anton J. Wijs and Dragan Bošnački. Improving GPU Sparse Matrix-Vector Multiplication for Probabilistic Model Checking. In *Model Checking Software*, pages 98–116. Springer Berlin Heidelberg, 2012.

[17] Pedro Rodrigues, Emil Lupu, and Jeff Kramer. LTSA-PCA: Tool Support for Compositional Reliability Analysis. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 548–551, 2014.

[18] Robert Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[19] 佐武一郎. **線型代数学**. 裳華房, 1974.

[20] Viktor Schuppan and Armin Biere. Shortest counterexamples for symbolic model checking of LTL with Past. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 493–509, 2005.

[21] SciPy: https://www.scipy.org/.

[22] NumPy: https://numpy.org/.

[23] Aldebaran Format: https://www.mcrl2.org/web/user_manual/language_reference/lts.htm.

[24] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., 1985.

[25] Ilja Zakharov. A survey of high-performance computing for software verification. In *Tools and Methods of Program Analysis*, pages 196–208. Springer, 2018.

[26] Paul L. Fackler. Algorithm 993: Efficient computation with kronecker products. *ACM Transactions on Mathematical Software*, 45(2), 2019.

IT システムの現行ソフトウェアにおける技術的負債可視化の重要性について

増井　和也†，馬場　辰男（ソフトウェア・メインテンス研究会）

はじめに

多様な利用分野向け IT システムで，稼働中ソフトウェアの不具合に起因する障害が多発している [12)-15)]。その原因の一つと考えられるものに，IT システムを構成する現行ソフトウェア（アプリケーションやインフラ等のソフトウェア）の老朽化（レガシ化）が指摘されている [16)17)]。当該老朽化の状態を可視化する研究も世界のソフトウェア科学者により行われている。特に老朽化を示す指標の一つである「技術的負債（Technical Debt）[1)]」の研究は，比較的盛んな分野一つのようである [2)-6)]。

ただ，日本の企業等が所有する IT システムは，日本独自の制度等から，技術的負債（以下 TD と略す）の実態把握を外部から客観的に可視化させることは容易ではない状況がある。本稿では，TD 可視化困難性（IT 資産価値低下可視化困難性）について分析し，将来に向けてなすべき施策（法令規制，適正なソフトウェア保守・品質管理体制）について提言する。

１．なぜ IT システムのTD は可視化されないのか

TD の具体例は後述するが，TD が可視化されない原因は大きく 2 つ考えられるのではないか。

一つ目の原因として，IT 部門が今や IT 投資の主導的役割を果たせていないという指摘がある [18)]。IT 部門の新任責任者は，前任から引継ぎを受けた際，実は IT 投資予算の多くを TD 返済に充てなければならないのが実態だと分かったとする。新任の責任者は，緊急返済が必要な TD に対応（多くにはソフトウェアの保守作業でTD 返済や TD 拡大防止）が余儀なくされる。しかし，開発対応が主要ミッションという IT 部門への大方の見方が，TD 返済の必要性を正面切って強く主張できない背景がある。結局，IT 部門が IT 投資の主導的役割を果たすためのコストが，見えないところで TD 対応に廻され枯渇していく。

もう一つの原因は，法的な規制の強制力である。自動車業界，機械業界，建築業界では，法令により，品質基準について厳しい規制がある。ここ 1～2 年において法令で定められた品質基準を（長い場合は何十年も前から）満たさない状態のまま製品出荷されていたという不適切対応報告が続いた。この事実がどのような過程で白日の下になったかは別として，その行為は明白な法令違反であり，法人や部門責任者が送検される対象になりうる。不正の実態を公表した企業は，厳しい社会的批判を浴び，謝罪，課徴金支払い，是正措置の報告を余儀なくされた [7)-11)]。一方，IT システムには，そういった品質基準に法的な厳しい規制は今のところない。TD の放置・拡大が法人として規制されない現状認識が TD の可視化がされない原因といえる。

２． TD を増大させる行為

表 1 に TD を拡大させる行為（IT 以外の製造業界では明らかに法令違反が疑われる行為）を示す。これらが蔓延，常態化している状況でも，IT 部門は現状法令違反に問われない。そのため，重大なシステム障害などで大きな社会的損失を発生させない限り，これらの行為から蓄積された TD の大きさの実態調査が外部組織によって行われることは少ない。

表 1 TD を拡大させる行為（他産業では法令違反の場合有）

No	不正行為	TD を拡大させる悪影響（例）
1	品質保証確認データの改ざんや捏造	テスト作業を実施せず，過去のエビデンス等を不正に使い品質資料を捏造。品質基準クリアが未確認な部分を残す
2	テスト仕様書や設計書レビューの手抜き	文書に誤記，記述漏れ，曖昧表現が存在するまま，マスタ文書として登録。当該文書で次の作業を行うと，手戻り作業発生の恐れあり。
3	おざなりなプログラム改修	機能改修は済んでも，非機能性（性能，信頼性，保守性等）が不十分な改修。条件により不具合が発生。再改修が必要となる場合も。
4	デグレード未確認（影響分析，回帰テスト不十分）	改修によるデグレード発生のリスクを軽視し，影響分析や回帰テストをまったく行わないか軽視する。
5	属人化の放置（情報共有の欠落）	担当者以外知らないことが大量にある状態で，担当本人でないと調査工数が大きくなる
6	保守担当者交代での不十分な引継ぎ	保守担当者が暗黙知として持っている情報が，十分引き継がれることなく次の担当者と交代（改修生産性の劣化）
7	バックログの放置	客先や運用部門からの改修要望の未対応案件が累積し，IT 部門の信用を無くす

今後国内外で TD を精度よく定量化・可視化する学術研究が進展しても，IT 部門が必要な情報や実態を開示しない限り，TD の現状や増減傾向は可視化されないままである。特にソフトウェア保守を担当者個人やベンダーに任せっきりのブラックボックス化状態では，IT システムの所有者にとっても TD の実態が不透明な部分が残る。

３．このまま放置すると何が起こるか

TD が可視化されているかいないか別にして，仮に TD 増大が止まらない IT 部門では，ソフトウェア作業の生産性は低下の一途をたどる。なぜなら，TD の返済（調査時間やテスト時間の増大）を陰で行いつつ，予算化対象のソフトウェア作業を行うことになるからである。対応人材の削減がされるとますます生産性は低下する。

このように，IT 部門の予算確保のために実施を約束した開発案件も，TD 返済で少なくなったコスト内で抑える必要があるため，まさに貸金業からの借金の自転車操業のように，多重債務者が陥るような，さらなる TD 増大黙認に手を染め続ける恐れがある。TD

† ソフトウェア・メインテナンス研究会 代表幹事

が可視化されないと，TD(債務)の大きさも不感症となる [19]。

レガシなシステムのすべてが危険な水準の TD を抱えているとは断言しないが，長期間稼働しているシステムが多く残っているという報告[20]があり，調査結果のグラフを引用する（図 1)。図1の稼働後が長いシステムは，TD を可視化して管理し，必要な TD 返済のため各期 IT 部門予算に返済枠が別にない限り，そのつけはIT システム担当者に回り，過重労働となる恐れがある。また下請けいじめといった担当者にその返済を押し付けるブラックな IT 部門の状態が加速する。優秀な人材が IT 部門に定着しない要因の一つかもしれない。

４．すぐ TD の可視化と報告義務の法令規制が必要

このことから，IT システムの TD の可視化（最終的に金額に換算して定量化）と健全性（TD が存在しても規模からみて妥当な範囲内にはあること）を有価証券報告書等で認証する法規制の制定が喫緊の課題である。TD が経営規模等から判断して，危険レベルと評価された企業は，行政指導（適切なソフトウェア保守体制や改修品質保証体制等の維持）を受け，従わなければならない。また更に危険な TD 状態では，システム障害予防の観点から IT システムのサービス停止命令というような強力な措置が可能な立法が必要と考える。システムのサービス停止命令は，IT システムに依存している企業等にとっては破綻（致命傷）と同じである。

５．おわりに

日本の企業等が所有する IT システムにおけるソフトウェアのTD可視化は喫緊の課題である。溜まった TD はどうやって返済（多くが保守対応で既存ソフトウェア修復を行い返済）するのか。それとも返済を諦めシステムを放棄（破綻）するのか。ただ，今や IT システムが停止した企業は企業活動の続行がほぼ不可能である。そのため，突然の IT システムの破綻（TD 返済の放棄）も軽々に選択できるものではない。

経営活動続行のために DX のようなシステム刷新により旧システムの TD 返済を今後あまりせずにすませる方法も選択肢としてはある。しかし，DX 投資失敗で TD を逆に増大させてしまうリスクも小さくない。現行 IT システムの TD をどう処置するか，適切な経営判断のためにも TD の法令による強制力を前提とした可視化は早急な実施が必要である。「2025 年の崖」[16]から気づかないうちに転落してしまうのを回避するためにも。

参考文献

1) Ward Cunningham ‘The WyCash Portfolio Management System’ 1992.3.6
2) Sasha Rezvina ‘Our 10-Point Technical Debt Assessment’ 2017.10.12
3) Terese Besker, Antonio Martini, Jan Bosch ‘The Pricey Bill of Technical Debt – When and by whom will it be paid?’ 2017.9 ICSME2017
4) Fiorella Zampetti, Cedric Noiseux, Giuliano Antoniol, Foutse Khomh, Massimiliano Di Penta ‘Recommending when Design Technical Debt Should be Self-Admitted’ 1017.9 ICSME2017
5) Everton da S. Maldonado, Rabe Abdalkareem, Emad Shihab and Alexander Serebrenik ‘An Empirical Study On the Removal of Self-Admitted Technical Debt’ 2017.9 ICSME2017
6) 亀井靖高，伊原彰紀，「技術的負債エンジニアリング ― 優先的に解決すべき技術的負債の解明とモデル化」 2018 年度実施報告書
7) 「当社グループにおける不適切行為に関する報告書」2018.3 株式会社 神戸製鋼所
8) 株式会社 SUBARU 向け「完成検査における不適切な取扱いに関する調査報告書」 2018.9 長島・大野・常松法律事務所
9) スズキ株式会社向け「完成検査における不適切な取扱いに関する調査報告書」2019.4 長島・大野・常松法律事務所
10) 「整備作業等の適正な実施に向けた是正措置について」 2019.5 株式会社 IHI
11) 「当社および当社の子会社が製造した建築物免震・制振用オイルダンパーの検査工程における不適切行為に関する外部調査委員会の調査報告について」2019.2 ＫＹＢ株式会社
12) 「障害報告書」2011.5 みずほ銀行システム障害特別調査委員会
13) 『「7pay（セブンペイ）」 サービス廃止のお知らせと これまでの経緯、今後の対応に関する説明について』2019.8 株式会社セブン&アイ・ホールディングス
14) 「東京リージョン （AP-NORTHEAST-1） で発生した Amazon EC2 と Amazon EBS の事象概要」 2019.8 https://aws.amazon.com/jp/message/56489/
15) 「ＪＲ指定券・乗車券類発売におけるクレジットカード取り扱い不可について」2020.2 鉄道情報システム株式会社
16) 『DX レポート～IT システム「2025 年の崖」の克服と DX の本格的な展開～』2018.9 経済産業省
17) 「金融機関のシステム障害に関する分析レポート」2019.6 金融庁
18) 「基幹系でも IT 部門は用無し、社長や IT ベンダーに「無視される悲惨さ」日経コンピュータ 2019.7.18 号
19) 「多重債務者対策を巡る現状及び施策の動向」2018.6 金融庁/ 消費者庁/ 厚生労働省（自殺対策推進室） / 法務省
20) 「ソフトウェアメトリックス調査 2019 システム開発・保守調査 ~システム開発・保守の実績プロジェクトデータを元に分析」日本情報システム・ユーザー協会

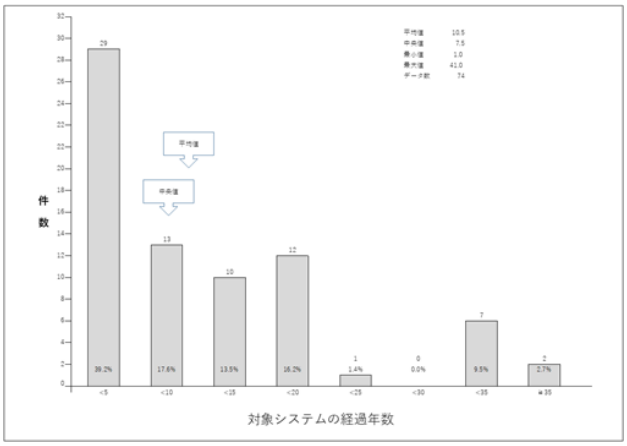


図 1 参考文献 20)の調査によるシステムの経過年数例

コミュニティ型チーム活動の進化 ～ COVID-19 対策サイト開発から学ぶ ～

松尾谷 徹
有限会社 デバッグ工学研究所
matsuodani@debugeng.com

増田 礼子
フェリカネットワークス株式会社
Ayako.Masuda@FeliCaNetworks.co.jp

要旨

この *Future Presentation (FP)* は，*IT* 技術者の働き方やモチベーションについての提案と，その根拠となった事例の説明です．事例は，従来の働き方とは異なる形態で驚異的な成果を示した「*COVID-19* 対策サイト開発」です．従来の働き方とは，企業や職場に閉じたクローズド型です．異なる形態とは，開発を受注した企業が *Open Souce Software (OSS)* として *GitHub* のオープン・コミュニティを活用したことです．ここでは，クローズド開発とオープン・コミュニティ開発を比較しました．

成果の大きな差は，開発や修正のスピードに現れており，二桁ほどの差がありました．一方，生産性やコストについては，オープン・コミュニティが参加者に対価を支払うことがないので，計測すらありません．職場などのリアルな環境での活動ではないこともあり，階層型の管理や予実管理は存在せず，自律管理で運用されていました．一番の差は，働く技術者のモチベーションです，クロースド型の職場は，ロイヤリティやチーム結束力がモチベーションです．オープン・コミュニティでは，存在承認に近いものがモチベーションになっていると推測されます． そのモチベーション源は雇用者や仲間からではなく，外部から与えられていることに注目して，この提案を行います．

1. はじめに

働き方改革は「労働時間を短縮して生産性を上げる」ための知恵を求めています．そのケーススタディとして「東京都 新型コロナウィルス感染症対策サイト開発」[1,2] (以降,「COVID-19 対策サイト開発」と呼ぶ) を取り上げて，技術者の観点から提案をまとめます．

事例分析の結果では，開発や修正のスピードにおいて，二桁ほどの差がありました．この差を生む仕組みや原因について分析を行いました．その背景に，管理監督によって技術者の労働を制御する方式と，技術者の自律性から労働する方式の差があります．ここでの主張は，自律的な行動の源となるモチベーションに関する分析がメインテーマです．この提案は次に示す節構成です．

2. 事例の背景： Open Souce Software (OSS) とオープン・コミュニティ開発
3. 事例「COVID-19 対策サイト開発」の概要
4. 分析結果：桁違いの顕著な特性
5. オープン・コミュニティの原動力
6. 提案のまとめ

2. OSS とオープン・コミュニティ開発

ソースコードの共有は，コンピュータの利用が始まって以来，学術機関や研究機関の間で行われています．1970 年代に入り，ソフトウェアを商用に利用するため，ソフトウェアの配布に制約を付与するプロプライエタリ・ソフトウェアと，ソースコードを非公開とするクローズドソースの商習慣が出来上がりました．国内の大手ソフトウェア企業は，現在でもこのビジネスモデルが中心であり，プロダクトの生産性や品質を高める努力を 40 年に渡り続けています [3].

一方，OSS は，商用とは別に発展を続け，1982 年には GNU プロジェクトが始まり，ボランティアによるコミュニティだけではなく，非営利団体として技術者を雇用し開発を拡大しました [4]．1991 年には，Unix カーネルと結合してシステム一式が完成し，PC などへの提供が行われました [5]．さらなる大きな進展はリーナス・トーバルズによる「Linux」開発です．それまでの OSS 開発もコミュニティで行われていましたが，リーナスはオープン・コミュニティで開発を始めました [6]．この仕組みにより，21 世紀に入ってから，OSS は爆発的な発展を始めます．この頃から，OSS 開発には，ホビイストやハッカーだけでなく，団体や企業からも技術者が参加するようになりました．日本では，現在でも OSS をホビイストやハッカーによる趣味の作品と思い込んでる人も少なくありませんが，実態は IT 系サービス・ビジネスの中核になっています．

リーマンショック後，プロプライエタリ・ビジネスは急激に衰退し，サービス系のビジネスが躍進しました．システムの複雑化から依存関係が増大し，OSS を含まないシステム提供ができなくなました．2010 年頃から，欧米の多くの企業がプロダクト型ビジネスをやめ，サービス型ビジネスへとシフトしました．その過程で，社内のローズド開発を OSS 開発へ移行しました．その受け口として，GitHub [7] の Organization (以降，Org と略す) が使われ，OSS のビジネス利用が始まりました [8]．GitHub には，日本の人口を超えるリポジトリ (プロジェクトに相当) が登録されていますが，Star の数などによる評価が高いものは，企業や団体が Org を運用するリポジトリであり，有償のサービスを併設している場合もあります．

今回のケーススタディも，商用サービスを提供する企業が開発に OSS を活用したものです．その特徴は，企業に閉じた開発チームではなく，オープン・コミュニティとして開発チームを運用した点です．そこから生まれたクローズド開発とは違った大きな効果に着目して，この提案を行います．

ここで用いたデータは，GitHub 社が提供する REST API v3 を使いました [9]．API インターフェースで得られないファイル別の修正などの詳細な記録は Git の log から分析しました [10]．いずれも公開されているデータです．

3. 「COVID-19 対策サイト開発」の概要

この提案の事例として，東京都が開設した新型コロナウイルス感染症 (COVID-19) に関する最新情報を提供するサイトの開発をとりあげました．東京都がある企業に情報サイト開発を発注し，その企業が GitHub のリポジトリを使って OSS 開発を行いました．このサイトの機能は，さまざまな情報をネット経由で提供する表示型のサイトです．処理の記述は，メイン言語として JavaScript フレームワークである Vue.js が使われています [11]．他にも，TypeScript，JavaScript，Python が使われ，データは，json，md (MarkDown)，yml で記述されています．

時系列で開発を観ると，2020 年 2 月 29 日に GitHub に登録され，ここから開発が始まり，3 月 5 日に初版がリリースされ，サイトが公開されました．その後，オープン・コミュニティの特性から，多くの支援者が参加し，その数は 250 名を超えています．並行して，札幌，名古屋，福岡，大阪，長野，浜松，など 40 を超える都市で流用され，それぞれの情報サイトが開設されました．

開発記録から直接確認できる特徴として，次のものが挙げられます．

- 開発の速さ
- 新たに参加する技術者 (貢献者) の立上りの速さ
- 問題処理 (Issue) の回答率と対応の速さ
- 他サイトへの流用容易性と流用の速さ

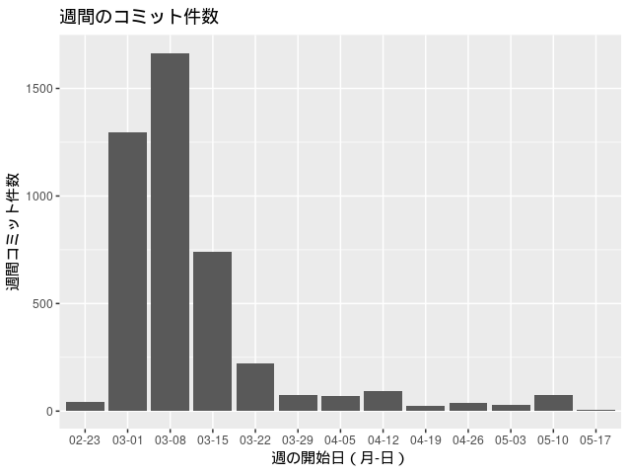


図 1: 週単位のコミット件数

開発の速さは，期間あたりのリポジトリへのコミット数から推測することができます．図 1 は，週単位で区切った期間でコミット件数の変化を表したものです．コミット件数は，3 月の第 1 週は 1,300 件，第 2 週には 1,700 件に達しています．クローズド開発における最速のアジャイル方式でも，マスタ・リポジトリに対して 1 週間あたり二桁のコミットを行うのは非常に稀です．事例の開発スピードは四桁なので，少なくとも二桁以上の差があると思われます．

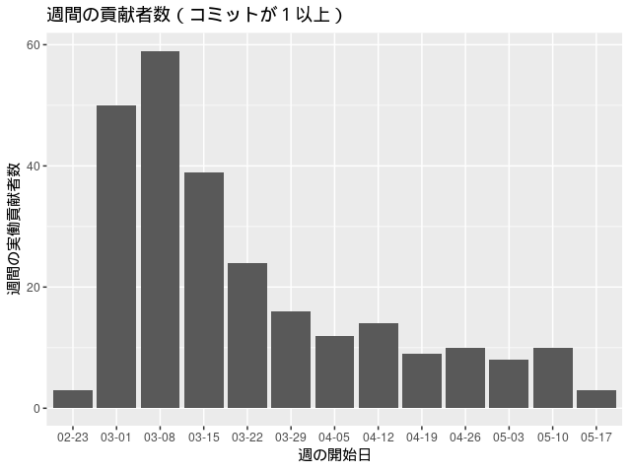


図 2: 週単位の実働貢献者数

初めて参加した技術者 (GitHub では Contributer：貢献者と呼ぶ) の立上りの速さとは，たとえばクローズド開発において，何らかの理由から急遽，工数投入が必要となった時に，技術者を確保して投入し，投入による成果が現れるまでの期間に相当します．図 2 は，週単位で区切った期間における技術者の人数を表しています．3 月の第 1 週で 50 名を超え，第 2 週では 60 名に達しています．図 1 は，開発の成果を表すもので，参加と同時に成果を出しています．週単位で技術者の増減があっても，引き継ぎなど成果に直結しない間接的な活動で開発が停滞する現象は見られません．技術者の人的資源調達や，投入時の立上りなど，クローズド開発と比べ，そのスピードは桁のオーダで差が見られます．この背景に，どのような管理システム，あるいはソフトウェア構造があるのかなど，興味深いテーマです．

GitHub における問題処理は Issue と呼ばれるシステムで処理されます，Issue には，バグの処理だけでなく，新規開発機能の担当募集や，仕様変更などさまざまな案件を対象にします．Issue の状態は，open と closed の 2 つです．クローズド開発のように，管理者が案件の担当者を決め，作業指示を行うシステムはありません．Issue が open になると，対応できる技術者が自律的に対策を作り，プルリクエスト (対策の確認依頼) を返します．

2 つの点で，クローズド開発とのスピード差が見られます．一つは，バグの発見が早いこと，もう一つは修正が速いことです．クローズド開発では，バグを見つけ出すためのレビューやテストに大きなコストを投入します．統合試験工程で発見されるバグもあり，後戻りコストが課題になっています．一方，この事例におけるバグ検出は，技術者が担っています．後述する GitHub workflow により，1 行の修正であっても，関連する機能について最新ビルドを用いてシステムレベルで動作確認を行います．その結果，多くの眼で，最新のシステム挙動を確認することになり，バグや GUI の不整合を早く発見することができています．

記録から，5 月 24 日現在のデータを観ると，3,814 件の Issue が提案されいて，当日発生分を含め closed は，3,698 件であり，対応率は 96.9 % です．一般的に，バグ票や問題処理票の回答時間分布は，正規分布とは異なっており，平均や分散で特性を示すことができませんが，四分位数で表すことで評価することができます．この事例では，Issue の 50 % の回答時間は 0.5 時間 (30 分) 以内で，75 % は 18 時間以内でした．

GitHub における他サイトへの流用は，Fork です．Fork は，GitHub に登録した人が，自分のリポジトリに Clone (Git のチェックアウト機能) する操作なので，どのように流用したのかを確認するには，Fork 先を調べる必要があります．この事例の Fork 件数は 1,900 件と増え続けており，その中で Fork 後に何らかの修正を行ったリポジトリは 70 件ありました．明示的に，都市名を示して Fork しているものもあり，40 件ほどは，流用が行われていることを確認しました．

4. 桁違いの特性原因を探る

ここでは，従来型の開発文化，すなわちクローズド開発の観点から，事例で観測された特性について考察します．

まず，ここでのクローズド開発について簡単に定義すると，プロジェクト開発のように有期限でリソース制約 (予算や工数) が存在し，目的を持った開発活動を想定しています．目的や制約は，顧客との契約が強く影響し，

変更するには経営的な稟議が必要です．開発体制は，雇用や契約により技術者を確保し，階層的な職位によって活動の指示と進捗管理を行います．技術者の役割は，たとえばサブシステム開発や品質保証やテストなどのように，機能的に分割されています．

事例とした COVID-19 対策サイト開発は，クローズド開発とは全く異なる異文化での開発のように見えます．その違いは，管理監督の役割や規範がない，テストフェーズや第三者テストがないなど真逆に近いものです．しかし，東京都から受注した企業は，顧客との間で何らかの契約や開発計画があったはずです．実際に行われた開発は，GitHub によるオープン・コミュニティ開発であり，OSS 開発であり，GitHub workflow でした． その結果，企業側が雇用した技術者とは別に，支援者として多くの技術者がオープン・コミュニティに加わり，開発を加速したのです．雇用側の技術者も，オープン・コミュニティ型の開発を取り入れ，開発スピードを加速しました．

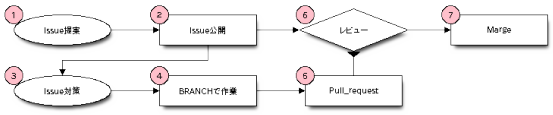


図 3: GitHub workflow の概要

オープン・コミュニティ型の開発とは，具体的には，GitHub workflow (または GitHub flow) のことです．このワークフローは，多人数で同時開発や保守を実行する時，混乱なく速く行うためのものです [12]．ワークフローの概略を図 3 に示し，それぞれの work について概略を示します．

1. Issue 提案
   議論が必要な機能要求やバグ連絡など課題を提起する

2. Issue Open (公開)
   課題が公開され，意見 (サポート) を待つ

3. Issue 対策
   対策提案者 (Open Issue に対する対策や回答を行う貢献者) が対策を提案する

4. BRANCH で作業
   対策を Topic ブランチで開発し，テストは Master から対策用のブランチを作成して評価する

5. Pull request
   Issue に対する解が得られれば，対策内容のレビュー要求を出す
   この時点で最新の統合状態で動作テストを実行する

6. レビュー
   関係者がレビューする
   関係者はそれぞれの状況で異なる

7. Marge (Issue Closed)
   レビュー後，問題がなければ Master にマージし，Issue を Closed とする

この GitHub workflow は，活動規範として共有されているため，見ず知らずの多くの技術者がオープン・コミュニティに参加しても，競合を起こすことなく，独立して開発を進めることができています．ソフトウェアの構造も，この規範に則って極力独立に作られ，相互に影響することを少なくしています．

GitHub workflow と，クローズド開発でも用いられている Git flow の差に注目する必要があります．ライブラリの入れ物としては同じ Git ですが，作業フローは異なります．GitHub workflow では，マスタに本番リポジトリを用います．Git flow では，本番用とは別に開発用のリポジトリや保守用のブランチを用います．GitHub workflow における Issue に対する修正活動は，修正を行う直前に本番リポジトリから Clone し，修正を確かめてプルリクエストを出すことで，対策活動中に本番リポジトリが他の修正対応等のマージが生じないよう，短い期間で修正する方式です． これによりマージされていない修正間の相互影響を減らすことができます．

5. オープン・コミュニティの原動力

事例で紹介した COVID-19 対策サイト開発は，オープン・コミュニティによる開発が特徴です．OSS は，もともとコミュニティ活動 (目的や理念を共有する社会集団の活動) です．OSS のコミュニティは，決してクローズドではないのですが，GNU など高度なコミュニティへの参加には障壁がありました．GitHub のオープン・コミュニティは，参加障壁を低くしたことが大きな特徴と考えられます．その結果，非常に多くの人が開発に参

加しています．GitHub 社も，ドキュメントの誤字脱字の修正から貢献することを推奨しています．

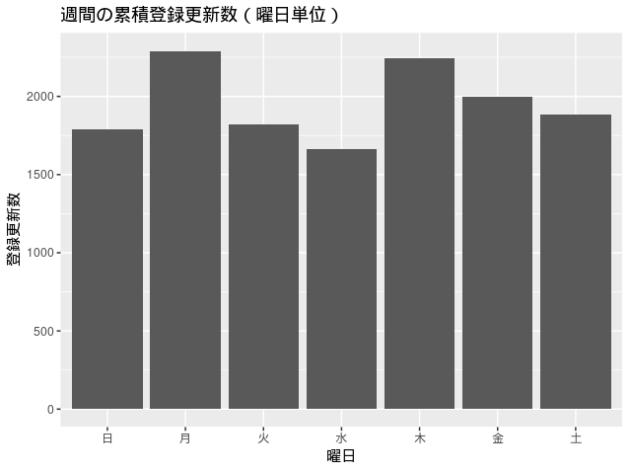


図 4: 曜日ごとの累積コミット分布

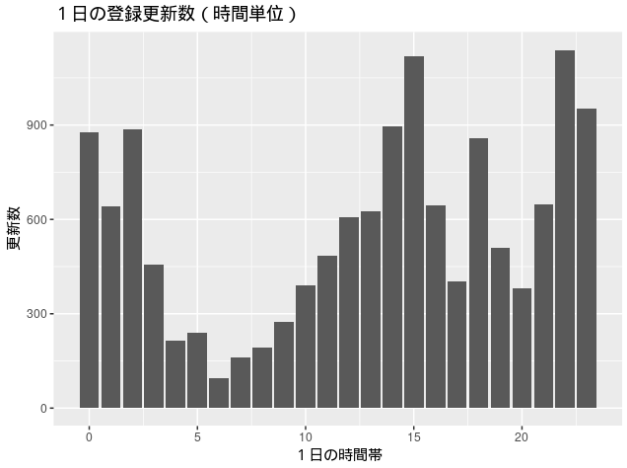


図 5: 24 時間の累積コミット分布

クローズド開発においては，契約や雇用など，ロイヤリティとの交換取引として，量的あるいは質的な貢献が求められます．厄介なことに，貢献の評価者や評価の価値観が主観的で技術的な価値とは異なる場合もあります．同じ企業内でも，部門によってその価値観は異なり，技術的な貢献が評価されないこともあり，技術は停滞する傾向にあります．

どちらが技術者にとって働きやすく，どちらが成果を出せるのでしょうか．机上で議論するとさまざまな意見が出てくるでしょう．ここでは，データから，オープン・コミュニティの働き方を見てみます．図 4 は，一週間のどの曜日で，どのくらい働いたかをコミット数の分布で示したグラフです．図 5 は，1 日 (24 時間) の何時にリポジトリを更新したかを表しています．図 4 と図 5 は，ある個人の働き方ではなく，全メンバの累計から求めたものです．クローズド開発 (労働指示の場合) であれば，かなりひどい労働基準法違反ですが，オープン・コミュニティでは，空いた時間で貢献する強いモチベーションによる自律的な行動として読み取ることができます．この指示型ではあり得ない働き方の特徴は，今回の事例だけでなく，地域を問わず多くの OSS 開発 / オープン・コミュニティで観測されています．

では，何がモチベーションとして働き，このような自律的な活動を生み出しているのでしょうか．従来のコミュニティ活動であれば，コミュニティとしての集団 = チームとしての団結から生じていると思われていました．しかしこの事例でも，そのようなメンバ間でのコミュニケーションや結束は見られません．目的共有は考えられますが，仲間としての団結 = 同一化については確認できません．

活動が活発な人に見られる特徴として，フォロワーの多さが挙げられます．GitHub に限らず，ネット上で個人が活動するさまざまなサイトでは，フォロワーの仕組みを取り入れています．

「フォロワーが多いと，モチベーションが維持され，広く活躍する」と考えられます．フォロワーの数と活動の関係は，GitHub の場合，一つのリポジトリ (プロジェクト) に閉じて論じることができません．オープン・コミュニティとは，包括プロジェクトのような垣根を超えた活動です．図 6 は，フォロワーの数と，その技術者が主催するリポジトリの数を示した散布図です．関係があることを回帰線で示しています．

フォロワー数に対数を用いたのは，その分布が対数正規分布であるためで，回帰分析を有効に使う分析上の理由です．リポジトリとは，今回の事例「COVID-19 対策サイト開発」のような開発プロジェクトで，技術者個人でも開設することができます．活発な技術者は，多くのリポジトリを自ら開設し，かつ，他のリポジトリも援助しています．

ここまで，事例「COVID-19 対策サイト開発」の活動だけを示して，驚異的と表現しましたが，支援した技術者の多くが，図 6 に示すように，自身でもリポジトリを主催して活動しています．また，ここでは分析していませんが，事例以外のリポジトリへの支援もあります．

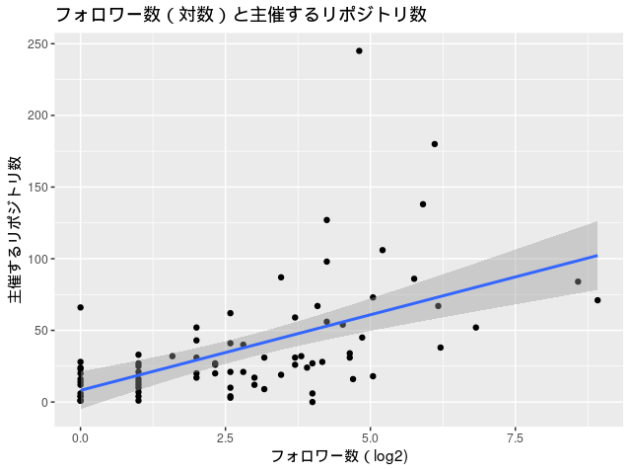


図 6: 技術者のフォロワー数 (対数) と保有リポジトリ数

6. 提案のまとめ

事例として取り上げた「COVID-19 対策サイト開発」は，クローズド開発と比べ，二桁ほど速い開発を実現していました．しかも，前章の最後に示しように，その活躍は一つのリポジトリに閉じていないのです．歩く速さでたとえると，少なく見積もっても，時速 4 km と，時速 400 km 程度の差です．差が大きすぎて比較対象外と考え，多くの人は，ひとまず先送りを選ぶかもしれません．

ここでの提案は，OSS 開発やオープン・コミュニティへの改宗を推奨するものではありません．技術者にとって，働きやすいとはどういうことでしょうか？なぜ，寝食を忘れて没頭するのでしょうか？これらについて，事例がヒントを与えていると考え，事例を紹介し，その推進力の一部を調べ，さらに調べ学ぶ価値があることを提案します．

7. おわりに

働き方改革「労働時間を短縮して生産性を上げる」という課題は，意義は解るけれども誰も達成できなかった課題です．しかし，「COVID-19 対策サイト開発」の事例を観ると想定を上回る成果を出しています．この事例に対しては，自分たちとは事情が異なるのでできないとする「言い訳論」と，過剰に成果を称賛する「称賛」の 2 極の論戦が考えられますが，ここで提案するのは，技術者の社会行動としてモチベーションの影響です．現段階では証明できていませんが，おそらく「社会的な承認欲求」だと考えています．

クローズド開発では，さまざまなストレスに耐えるモチベーション源として，仲間，チームが有効であり，我々は，「職場 = チーム」と考えチームの活性化について研究を進めてきました．数年前から OSS 開発における状況の調査を始めて，全く異なる働き方に遭遇しました．ここで示した「COVID-19 対策サイト開発」の状況と同種の傾向が，多くの OSS 開発で観測されています [13]．この傾向は，10 年前の OSS 開発データでも現れています．働き方のターニングポイントが近い予感がしています．

参考文献

[1] 東京都 新型コロナウィルス感染症対策サイト, https://stopcovid19.metro.tokyo.lg.jp/ (accessed:2020/03/16)

[2] 東京都 新型コロナウイルス感染症対策サイト (GitHub) (Tokyo COVID-19 Task Force website), https://github.com/tokyo-metropolitan-gov/covid19 (accessed:2020/03/16)

[3] オープンソースソフトウェアの歴史 (Wikipedia), https://ja.wikipedia.org/w/index.php?title=オープンソースソフトウェアの歴史&oldid=72952775 (accessed:2020/05/28)

[4] GNU (Wikipedia), https://ja.wikipedia.org/w/index.php?title=GNU&oldid=76163596 (accessed:2020/05/28)

[5] UNIX (Wikipedia), https://ja.wikipedia.org/w/index.php?title=UNIX&oldid=77135146 (accessed:2020/05/28)

[6] Linux (Wikipedia), https://ja.wikipedia.org/w/index.php?title=Linux&oldid=77752398 (accessed:2020/05/28)

[7] GitHub, https://github.com/ (accessed:2020/03/16)

[8] Organization の作成 - GitHub ヘルプ, https://help.github.com/ja/enterprise/2.19/admin/user-management/creating-organizations (accessed:2020/05/28)

[9] RES API v3, https://developer.github.com/v3/ (accessed:2020/03/16)

[10] Git - git-log Documentation, https://git-scm.com/docs/git-log (accessed:2020/03/16)

[11] Vue.js (Wikipedia), https://ja.wikipedia.org/w/index.php?title=Vue.js&oldid=75855212 (accessed:2020/03/16)

[12] ワークフローを設定する - GitHub ヘルプ, urlhttps://help.github.com/ja/actions/configuring-and-managing-workflows/configuring-a-workflow (accessed:2020/05/28)

[13] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦, 「大規模オープンソース・ソフトウェアプロジェクトにおける開発効率の計測」, 電気学会論文誌 C (電子・情報・システム部門誌), Vol.138, No.8, pp.1011–1019, 一般社団法人 電気学会, 2018

デジタル変革を推進する上での課題

山本 修一郎
名古屋大学大学院
syamamoto@acm.co.jp

要旨

本稿ではデジタル変革(Digital Transformation, DX)を日本企業が推進する上での課題を提示する. 経産省が公開したDX推進指標などの限界を指摘するとともに, 日本のユーザ企業とベンダ企業に求められる新しい関係構築の必要性について展望する.

1. はじめに

筆者が委員として参加した経産省「デジタルトランスフォーメーションに向けた研究会」による DX レポート[1]を契機に日本企業では DX 推進部署の設置が進んでいる. また, DX レポートで提起された DX の「見える化」指標は「DX 推進指標」として公開され, 民間企業における DX 状況の評価が始まった[2].

本稿では. 経産省の取組みの限界をまとめるとともに, DX を企業が推進する上での課題を明らかにする.

2. 経産省の取組み

2.1. DX レポート

「DX 推進システムガイドライン」については, 一般論が説明されているので, 各企業では, DX の推進策を具体的に定義する必要がある[3]. IT システム構築におけるコスト・リスク低減策について, 4 項目の課題を列挙しただけである. 各企業がこれらのコスト・リスク低減策を具体化する必要がある.

ユーザ企業・IT ベンダ企業の目指すべき姿と双方の新たな関係の構築についても必要性を指摘しているだけである. 各企業が自ら目指すべきと相互関係を具体化する必要がある.

DX 人材の育成・確保については, 必要となる DX 知識が欠落している. 各企業が自ら DX 知識を定義して人材育成計画を策定する必要がある.

IT システム刷新の見通し明確化について, デジタル技術によるビジネスモデル創出, システム刷新へのベンダ企業による支援の必要性を指摘しているが, 具体策まで踏み込めていない. たとえば、ビジネスモデルとITシステムの整合性を確立するためのDX方法論を提示すべきである。

2.2. DX 推進指標

DX 推進指標の診断スキームには, 経営のあり方と IT システム構築について定性および定量指標がある. 定性指標は9個の主質問と25個の従質問からなり, 証跡文書に基づいて回答する. 証跡文書としては経営計画, 事業計画, バランススコアカード[4]などが例示されている. ただし, DX 推進指標の用語関係は明示されてはいない.

現在, 試行評価が完了して公開された段階である. 今後各企業が DX 推進指標を用いて自己評価するとともに, 必要があれば指標をカスタマイズしていく必要がある.

3. DX 推進課題

3.1. デジタルエンタープライズの実現

DX は目的ではなく, デジタルエンタープライズを実現するための手段である[1,2,3,5]. このため, DX 用語, DX 戦略, DX ロードマップ, DX プロセス, DX アーキテクチャ, DX リポジトリなどからなる総合的な DX 方法論を EA(Enterprise Architecture)[7,8]に基づいて体系的に確立する必要がある. 大企業の DX プロセスの例には[9]がある.

日本では EA は廃れているという認識[6]もあるが, 海外では EA による DX が進展している[10-15]. たとえば, マイクロサービスアーキテクチャ[16-25]ではビジネスケイパビリティと整合するアプリケーションアーキテクチャを開発する必要があるので, EA が有効である[15]. TOGAF の図式言語 ArchiMate[26]ではビジネスアーキテクチャとアプリケーションアーキテクチャをサービスによって関係付けることができる. また ArchiMate により主要なビジネスモデルを記述できる[27].

なお, デジタルエンタープライズの概念については日本にも先駆的な提案がある[28].

3.2. ユーザ企業とベンダ企業の関係

上述した総合的な DX 方法論をユーザ企業だけで構築するのは不可能である[29]. ベンダ企業が DX を支援する DX 方法論を早期に構築することにより,ユーザ企業の DX を連携して推進すべきである.

4. まとめ

本稿では, DX を推進するために民間企業とベンダ企業の関係を再構築する上での課題を明かにした.

参考文献

[1] 経済産業省,DX レポート ～IT システム「2025 年の崖」克服と DX の本格的な展開～, http://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/20180907_report.html, 2018

[2] 経済産業省,「DX 推進指標」とそのガイダンス, https://www.meti.go.jp/press/2019/07/20190731003/20190731003-1.pdf

[3] 山本修一郎, デジタル変革をどうするか, 日本経営協会, オムニマネジメント, Vol.7, No.12, pp.2-7, 2019

[4] Kaplan, R., Norton, D., The Balanced Scorecard: Measures that Drive Performance, Harvard Business Review. Jan-Feb. pp.71-79, 1992.

[5] 山本修一郎,デジタル変革に向けたデジタルバランススコアカード DBSC の提案, KBSE 研究会, 2020/1/25

[6] 室脇 慶彦, IT 負債 基幹系システム「2025 年の崖」を飛び越えろ, 日経 BP, 2019

[7] The Open Group, The TOGAF® Standard, Version 9.2, C182, 2018

[8] Yamamoto, S., Olayan, N., Morisaki, S., Another Look at Enterprise Architecture Framework, Journal of Business Theory and Practice, Vol 6, No 2, pp.172-183, 2018, DOI: http://dx.doi.org/10.22158/jbtp.v6n2p172

[9] Sebastian, I., Ross, J., Beath, C., Moloney, K., Fonstad, N., How Big Old Companies Navigate Digital Transformation, MIS Quarterly Executive, September 2017 (16:3), pp.197- 213.

[10] The Open Group, DPBoK (Digital Practitioners Body of Knowledge), S185, 2019

[11] Giovanni Traverso, Loh WoeiMin, Brian Ng, Customer Experience-Driven Enterprise Architecture: How to Revitalize your DSP Business, W166, The Open Group, 2016

[12] Urbach, N., Röglinger, M. Eds., Digitalization Cases, How Organizations Rethink Their Business for the Digital Age, Springer, 2019

[13] Zimmermann, A., Schmidt, R., Sandkuhl, K., Mohring M., Evolving Enterprise Architecture for Digital Transformations, Digital Enterprise Computing 2015, pp.15-26, 2015

[14] Zimmermann, A., Schmidt, R., Sandkuhl, K., Wiβotzki, M., Jugel D., Bogner, J., Mohring M., Digital Enterprise Architecture –Transformation for the Internet of Things-, 19th Enterprise Distribute Object Computing Workshop, pp.130-138, 2015

[15] Bogner, J., Zimmermann, A., Towards Integrating Microservices with Adaptable Enterprise Architecture, 20th Enterprise Distribute Object Computing Workshop, pp.158-163, 2016

[16] Balakrushnan, S. et al., Microservice Architecture, TOG white paper w169, 2016

[17] Richardson, C., Microservice Patterns, MANNING, 2018

[18] Dragoni, N., Giallorenzo, S., Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., Safina, L., Microservices: yesterday, today, and tomorrow, 2017

[19] Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S., Mazzara, M., From Monolithic to Microservices – An Experience Report from the Banking Domain, IEEE Software, May/June 2018, pp.50- 55.

[20] Knoche, H., Sustaining Runtime Performance while Incrementally Modernizing Transactional Monolithic Software towards Microservices, ICPE'16, pp.121-124, 2016.

[21] Knoche H., Hasselbring, W., Using Microservices for Legacy Software Modernization, IEEE Software, pp.44-49, 2018.

[22] Levcovitz, A., Terra, R., Valente, M., Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems, 3rd Brazilian Workshop on Software Visualization, Evolution and Maintenance (VEM), p. 97–104, 2015.

[23] Dragoni, N., Dustdary, S., Larsenz, S., Mazzara, M., Microservices: Migration of a Mission Critical System, IEEE Transactions on Services Computing, pp.1-14, 2017

[24] Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S., Mazzara, M., From Monolithic to Microservices – An Experience Report from the Banking Domain, IEEE Software, May/June 2018, pp.50- 55.

[25] Taibi, D., Lenarduzzi, V., On the Definition of Microservice Bad Smells, IEEE Software, May/June 2018, pp.56- 62.

[26] The Open Group, ArchiMate® 3.1 Specification, C197, 2019

[27] Yamamoto, S., A Comparative Analysis of Business Model Notations, Journal of Business Theory and Practice ISSN 2372-9759 (Print) ISSN 2329-2644 (Online) Vol. 7, No. 3, 2019, pp. 111-123

[28] 永田守男, 次世代デジタルエンタープライズのコンセプト, 信学技法, KBSE2000-71, pp.39-46, 2001.1

[29] 山本修一郎, DX 推進を阻む日本企業の 7 つの壁, 経理情報, pp.1, 2019.12.20

ソフトウェア・シンポジウム 2020
論文集

2020 年 6 月16日　初版発行

編者　　小笠原秀人
　　三輪東

発行者　ソフトウェア技術者協会

〒157-0073 東京都世田谷区砧二丁目 17 番 7 号
株式会社ニルソフトウェア内
ソフトウェア技術者協会 事務局
TEL: 03-6805-8931
http://sea.jp/
ISBN 978-4-916227-26-3
