ソフトウェア・シンポジウム 2022 in 盛岡

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 42 回目を迎える 2022 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，SS2021 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 ： 2022 年 6 月 8 日（水曜日）～ 10 日（金曜日）

■ 場所 ： いわて県民情報交流センター（アイーナ）
感染症対策を行い，オンラインとのハイブリッド開催です．

■ 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 ： 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，岩手県情報サービス産業協会

スタッフ一覧

実行委員会

実行委員長

漆原 憲博（ジェーエフピー）
野村 行憲（ワイノムラドットコム）

実行委員

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）
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）
佐々木 千春（ジェーエフピー）
杉田 義明（福善上海）
鈴木 裕信（usp lab.）
富松 篤典（電盛社）
中野 秀男（中野秀男研究所）
奈良 隆正（NARA コンサルティング）
布川 博士（岩手県立大学）
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

岡本 圭史（仙台高等専門学校）
小田 朋宏（SRA）

プログラム委員

秋山 浩一（ASTER）
安達 賢二（HBA）
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)
落水 浩一郎（University of Information Technology, Myanmar）
片山 徹郎（宮崎大学）
紙名 哲生（大分大学）
喜多 義弘（長崎県立大学）
北須賀 輝明（広島大学）
日下部 茂（長崎県立大学）
楠本 真二（大阪大学）
熊澤 努（SRA）
栗田 太郎（ソニー）
河野 哲也（メルカリ）
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）
小林 修（SRA）
阪井 誠（SRA）

プログラム委員会

酒匂 寛（デザイナーズデン）
佐原 伸（ソフトウェア・エンジニア）
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）
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 コネクト）
西 康晴（電気通信大学）
根本 紀之（東京エレクトロン）
野村 行憲（ワイノムラドットコム）
端山 毅（NTT データ）
久住 憲嗣（芝浦工業大学）
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）
宗平 順己（武庫川女子大学）
森崎 修司（名古屋大学）
八木 将計（日立製作所）
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（広島大学）

事務局

栗田 太郎（ソニー）

ソフトウェア・シンポジウム 2022 in 盛岡 目次

■論文・報告 ［データ駆動・検証］

[経験論文] 深層学習における正則化へのドロップアウトデザインの適用
熊澤 努（株式会社 SRA），地嵜 頌子（大阪工業大学），中川 智之（東京理科大学），
室井 浩明（ヱヂリウム株式会社），渡邉 卓也（ヱヂリウム株式会社）
…… 1

[経験論文] OSS の参加者は何を求め，何を得ているのか？
－ 利用者と参加者の行動の視覚化 －
増田 礼子（フェリカネットワークス株式会社），松尾谷 徹（有限会社デバッグ工学研究所）
…… 11

■論文・報告 ［教育・プロセス］

[経験論文] 大学教育における電子商取引システム開発演習事例およびチーム内の役割分析
増田 聡（東京都市大学），松尾谷 徹（デバッグ工学研究所）
…… 20

[経験論文] カスタマーサクセスによる業務プロセス改善の分析とシステム拡張応用
～サブスクリプションモデルにおけるシステム開発のための一つのアプローチ～
八木 将計（株式会社日立製作所），大澤 郁恵（株式会社日立製作所），
堀 光孝（株式会社日立製作所），丸田 絃心（レクシエス株式会社）
…… 27

[事例報告] 「オントロジー」など「汎用語」の情報処理分野への転用史
塚田 良央（株式会社ジェーエフピー），佐々木 千春（株式会社ジェーエフピー），
漆原 憲博（株式会社ジェーエフピー），栗田 太郎（ソニー株式会社）
…… 37

■論文・報告 ［品質・信頼性］

[研究論文] 模範解答を用いたコンパイルエラー箇所指摘の高精度化
中井 亮佑（大分大学），紙名 哲生（大分大学）
…… 47

[経験論文] ソフトウェアレビュー研究結果の認知拡大と適用促進
安達 賢二（株式会社 HBA），中谷 一樹（TIS 株式会社），
上田 裕之（株式会社 DTS インサイト）
…… 55

[研究論文] ソースコードコメントに着目した不確かさとソフトウェア品質の関係調査
渡邉 紘矢（京都工芸繊維大学），崔 恩瀞（京都工芸繊維大学），
水野 修（京都工芸繊維大学）
…… 65

■論文・報告 ［要求・仕様］

[研究論文] 建築設備自動制御のソフトウェアへの問題フレーム適用について
井口 日文（放送大学），中谷 多哉子（放送大学）
…… 75

[経験論文] COTS 利用プロジェクトへの GSN の適用
田中 康（有限会社ケイプラス・ソリューションズ ，奈良先端科学技術大学院大学）
…… 85

[研究論文] 機能共鳴分析手法 FRAM による遠隔授業のシナリオ分析の視覚化
日下部 茂（長崎県立大学），小佐古巴菜（長崎県立大学），
有田 大作（長崎県立大学）
…… 94

■論文・報告 [Future Presentation]

[Future Presentation (1)] ”技術を持つ人” と ”その技術を欲している人” をつなぐ
マッチングソフトはつくれるだろうか？
本多 慶匡 （SEA Hokkaido） …… 101

[Future Presentation (2)] プロセスモデルの補完方法 –モデル・ノウハウ・人–
阪井 誠 （株式会社 SRA） …… 105

深層学習における正則化へのドロップアウトデザインの適用

熊澤 努
株式会社 SRA
kumazawa@sra.co.jp

地嵜 頌子
大阪工業大学
shoko.chisaki@oit.ac.jp

中川 智之
東京理科大学
t_nakagawa@rs.tus.ac.jp

室井 浩明
ヱヂリウム株式会社
muroi@edirium.co.jp

渡邉 卓也
ヱヂリウム株式会社
sodium@edirium.co.jp

要旨

深層ニューラルネットワークの汎化性能を向上させる正則化法に，学習時に一部のノードだけを活性化するドロップアウト法が知られている．ドロップアウト法が活性化するノードを無作為に選択する技法なのに対して，ノードと重みを *1* エポックの訓練で均一に選択する，ドロップアウトデザインと呼ばれる組合せ構造が提案されている．しかしながら，正則化の効果については不明点が多く，実験を通じて解明されることが望まれてきた．本論文では，ドロップアウトデザインの性質を明らかするために，ドロップアウトデザインを用いた正則化法を多層パーセプトロンに適用した．*4* 種類のネットワーク構成について，画像分類問題の精度と損失を評価する実験を行い，すべてのネットワークについてドロップアウト法と同程度の汎化性能を示すことを確認した．一方で，ドロップアウトデザインに固有の特徴的な結果は観察されなかった．以上の成果は，ドロップアウト法とドロップアウトデザインをより深く理解するための手掛かりを今後の研究に提供することが期待される．

1. はじめに

近年，深層学習に関連した研究開発が非常に活発に行われており，画像認識や自然言語処理をはじめとして様々な分野への応用が図られている．深層学習は訓練データに対する最適化を実行するため，訓練データに対しては高い評価性能を示す一方，未知のデータに対する汎化性能は向上しなくなる問題が指摘されている．この過学習の問題を緩和する方法が正則化である．

ドロップアウト法 [1, 2] は最も有力な正則化法の一つとして広く知られている．ドロップアウト法は，ネットワークの訓練の間に無作為にノードを不活性にすることで，一部の重みパラメータの学習のみを進める方法である．不活性にするノードはミニバッチごとに切り替える．訓練を終了した後の，未知データに対する出力値の推定の際には，全てのノードを活性化させる．ノードの不活性化は，部分ネットワークで重みパラメータの値を学習することと解釈できる．そのため，ドロップアウト法は複数のモデル間でパラメータを共有するアンサンブル学習の一種としても理解されている．ドロップアウト法を使用する際には，ノードを不活性にする確率をハイパーパラメータで与える必要がある．

地嵜らは，ドロップアウト法に代わる正則化技法を想定した組合せ構造であるドロップアウトデザイン [3] を提案した．ドロップアウトデザインは，ミニバッチごとの訓練で使用する部分ネットワーク構成の集合である．この部分ネットワークの構成をスーパーブロックという．ドロップアウトデザインは，1 エポックあたりのノードの活性化回数と重みパラメータの更新回数が所与の値で釣合うように，スーパーブロックを構成する．したがって，訓練時に特定のノードだけが活性化されて学習が進む現象を緩和することが期待できる．文献 [3] では，特定のドロップアウトデザインの構成法が示されており，計算機を用いたデザインの構成が可能となっている．その一方で，正則化の効果をはじめ，ドロップアウトデザインの性質については明らかになっていない点が多い．

特に，ドロップアウトデザインを機械学習に実際に適用してその性質を調べる研究はこれまでなされておらず，評価実験に基づいた研究の成果が望まれる状況にある．

本論文の目標は，ドロップアウトデザインを用いた正則化を深層学習に対して適用して，デザインの性質を実験的に明らかにすることにある．実験では，画像分類問題をとりあげた．ニューラルネットワークは，多層パーセプトロン（MLP）を対象とした．全結合層（隠れ層）の層数とノード数が異なる計 4 種類のネットワークに対して，正則化を実行しない場合，従来のドロップアウト法を適用した場合，ドロップアウトデザインを適用した場合についてそれぞれ汎化性能を評価した．その結果，次の二点を確認した．

- すべてのネットワークについて，ドロップアウト法とドロップアウトデザインによる正則化法は同程度の精度と損失を示した．

- ドロップアウト法にはない特徴的な結果は，ドロップアウトデザインを用いた結果には見られなかった．

以上の結果は，ノードを無作為に選ぶドロップアウト法と，ブロックによって均一にノードを活性化させるドロップアウトデザインには，性能上大きな差がないことを示唆している．このことは，組合せ構造であるドロップアウトデザインと無作為性を持つドロップアウト法の関連性を示しており，今後の研究において，組合せ論を使ったことによりドロップアウト法を分析するアプローチが期待できる．本論文のもう一つの貢献は，ドロップアウトデザインをニューラルネットワークに適用する具体的な方法を示したことである．ドロップアウトデザインの適用に際しては，デザインの構造とニューラルネットワークの構成との対応づけ，複数エポックの訓練におけるスーパーブロックの割り当てを決定する必要がある．本論文では，多層パーセプトロンの全結合層に対して正則化を行う場合のドロップアウトデザインの適用法を示し，実験により汎化性能の違いを調べた．

本論文の構成は次の通りである．2 節で研究の背景となるドロップアウト法とドロップアウトデザインについて説明する．3 節では，ドロップアウトデザインの深層学習への適用法を説明する．4 節で，評価実験の結果を報告する．5 節で関連する研究を論じ，6 節で結論と今後の課題を述べる．

2. 背景

ここでは，本論文が着目するドロップアウト法とドロップアウトデザインの概要を説明する．

2.1. ドロップアウト法

ドロップアウト法 [1, 2] は，ニューラルネットワークのノードを無作為に不活性にして訓練を行うことで，汎化性能を高める技術である．ノードを不活性にする確率であるドロップアウト率 p はハイパーパラメータとして与えられる，訓練時には，層ドロップアウトマスクと呼ばれるベクトル $\mathbf{m}$ で層の出力をマスキングすることでノードの活性化と不活性化を行う [4]．ただし，$\mathbf{m}$ の各成分 m_i はベルヌーイ分布に従って確率 p で 0 を，確率 $1-p$ で 1 をとる．すなわち，$m_i \sim Bernoulli(1-p)$ とする．ここで，$\mathbf{m}$ の各成分はノードを表し，1 であれば活性化することを，0 であれば不活性化することを意味する．ドロップアウト法を適用する層への入力 $\mathbf{x}$ と出力 $\mathbf{y}$ との間には以下の関係が成り立つ [4]．

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \circ \mathbf{m} \tag{1}$$

ここで，$\mathbf{W}$ は層の重み，$\mathbf{b}$ はバイアス，f は活性化関数，$\circ$ は成分ごとの積を表す．未知データに対する推定時には全てのノードを活性化し，以下のように平均化を行う．

$$\mathbf{y} = (1-p)f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2}$$

ドロップアウト法の異なる定式化として，訓練時には (1) 式の右辺に $1/(1-p)$ を乗じた値を出力する一方，推定時には (2) 式の $(1-p)$ の乗算を実行しない方式があり，どちらの定式化を用いても，推定時の出力が等しいことが知られている [4]．簡単のため，本論文では両者を区別せず，(1) 式と (2) 式を用いる．

ドロップアウト法で訓練に使われるネットワークは，元のネットワークの部分ネットワークである．したがって，ドロップアウト法を，重みを共有する複数のネットワークによるアンサンブル学習の一種とみなすことができる [1]．

2.2. ドロップアウトデザイン

ドロップアウトデザインは，ニューラルネットワークの重みを更新する回数を均一にすることを目的とした組

合せ構造の一種である．ドロップアウトデザインは複数の点集合から構成される．ネットワークの各層のノードの集合を異なる点集合とみなし，層を横断した部分集合をとることによって活性化するノードを選択するモデルを与える．$V_1, V_2, \ldots, V_n$ を互いに異なる点集合とし，それぞれの部分集合の集合 (スーパーブロックと呼ぶ) を以下のように定める．

$$\mathcal{B} = \{\, \{C_1|C_2|\cdots|C_n\} \mid C_i \subset V_i,\ C_i \neq \emptyset,\ 1 \leq i \leq n\}.$$

各部分集合 C_i をサブブロックと呼ぶ．任意の $0 \leq i \leq n-s$ について，連続した s 個の点集合 $V_{i+1}, V_{i+2}, \ldots, V_{i+s}$ それぞれから選出した任意の $d_1, d_2, \ldots, d_s$ 個の点を同時に含むスーパーブロックが $\mathcal{B}$ の中に λ_i 個存在するとき，$(V_1, V_2, \ldots, V_n; \mathcal{B})$ を $(d_1, d_2, \ldots, d_s)$ 型-ドロップアウトデザインと呼ぶ [3]．λ_i $(0 \leq i \leq n-s)$ はドロップアウトデザインの会合数と呼ばれる．会合数を一定にすることで，ノードの活性化回数及びエッジの使用回数の均一化を実現する．2 層のネットワークでの例を以下に示す．

例 1. 点集合を $V_1 = \{0,1,2,3\}, V_2 = \{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4},\mathbf{5}\}$ とする．このとき，$(V_1, V_2; \mathcal{B})$ は会合数 $\lambda_1 = 1$ をもつ $(2,1)$ 型-ドロップアウトデザインをなす． ただし，

$$\begin{aligned}\mathcal{B} = \{&\{0,1 \mid \mathbf{0},\mathbf{1},\mathbf{2}\}, \quad \{0,1 \mid \mathbf{3},\mathbf{4},\mathbf{5}\}, \quad \{0,2 \mid \mathbf{1},\mathbf{2},\mathbf{5}\},\\ &\{0,2 \mid \mathbf{0},\mathbf{3},\mathbf{4}\}, \quad \{0,3 \mid \mathbf{1},\mathbf{2},\mathbf{4}\}, \quad \{0,3 \mid \mathbf{0},\mathbf{3},\mathbf{5}\},\\ &\{1,2 \mid \mathbf{0},\mathbf{2},\mathbf{3}\}, \quad \{1,2 \mid \mathbf{1},\mathbf{4},\mathbf{5}\}, \quad \{1,3 \mid \mathbf{0},\mathbf{4},\mathbf{5}\},\\ &\{1,3 \mid \mathbf{1},\mathbf{2},\mathbf{3}\}, \quad \{2,3 \mid \mathbf{0},\mathbf{1},\mathbf{3}\}, \quad \{2,3 \mid \mathbf{2},\mathbf{4},\mathbf{5}\}\}.\end{aligned}$$

このデザインでは，例えば，V_1 の 2 個の点 $0, 1$ と V_2 の点 $\mathbf{0}$ を同時に含むスーパーブロックは 1 個だけである．

先行研究において, 直交配列を用いた構成法や，アフィン空間や射影空間の有限幾何を用いた構成法が提案されている．本論文の実験に使用したアフィン空間による構成法は，$(2,1)$ 型かつ $(1,2)$ 型-ドロップアウトデザインを構成することができる (Theorem 5.13 [3])．ここで, デザインから構成されるモデルは, 素数ベキ q, 整数 t, $d \geq 3$ について, 層数は q^{d-t}，各層のノード数は q^t であり，会合数は各 i に対して $\lambda_i = (q^{d-2} - q^{d-t-1})/(q-1)$ をとる．また，各層のノードを活性化する割合は $1/q$ である．したがって，不活性にするノードの割合，すなわちドロップアウト率は $1 - 1/q$ となる．

3. ドロップアウトデザインの深層学習への適用

本節では，ドロップアウトデザインのニューラルネットワークへの適用方法について述べる．

本論文では，訓練に使用するドロップアウトデザインをネットワークごとに固定する．ドロップアウトデザインは，1 エポックの学習に必要な部分ネットワークの集合を定めたものと考える．2.2 節で述べたように，ドロップアウトデザインの点集合は，ノードの集合と 1 対 1 に対応させる．スーパーブロックは，1 つのミニバッチで学習を進める部分ネットワークであり，サブブロックは各層において活性化するノードの集合とする．(1) 式において，サブブロックに含まれる点に対応づけられたノードを 1，含まれない点に対応づけられたノードを 0 とするように層ドロップアウトマスク $\mathbf{m}$ を構成することで，訓練時のノードの活性化と不活性化を実現する．

ドロップアウトデザインの各スーパーブロックを，1 エポックにつき 1 回，重みの更新に使用する．ただし，複数エポックの訓練におけるスーパーブロックの使用法については，ドロップアウトデザインの定義には定められていない．本論文では，第 1 エポックで使用するスーパーブロックの順序を任意に一つ固定するものとし，複数エポックの訓練について，次の二通りの使用法を考える．

- 第 1 エポックで定めたスーパーブロックの使用順序を巡回的に保つ．第 2 エポック以降では，各エポックの開始時にスーパーブロックを 1 だけシフト移動して巡回したデザインを構成することで，スーパーブロックの使用順序を定める．以降では，この使用方法をブロックシフトと呼ぶことにする．ブロックシフトにより構成されるドロップアウトデザインは互いに同値である点に注意する．

- 第 1 エポックで定めたスーパーブロックの使用順序を第 2 エポック以降も引き続き使用する．

例 2. 例 1 のドロップアウトデザイン $\mathcal{B}$ を再び考える．表 1 にブロックシフトの適用例を示す．まず，$\mathcal{B}$ のスーパーブロックの集合を順列と読み替えることで，第 1 エポックでのスーパーブロックの使用順序を表 1a のように定める．このデザインにブロックシフトを適用すると，第 2 エポックでのスーパーブロックの使用順序が 表 1b のように定まる．ブロックシフトを適用しない場合には，表 1a の順序を第 2 エポック以降も使用する．

表 1: ブロックシフトの例

(a) 第 1 エポック

順序	1 層	2 層
1	0, 1	**0**, **1**, **2**
2	0, 1	**3**, **4**, **5**
3	0, 2	**1**, **2**, **5**
4	0, 2	**0**, **3**, **4**
5	0, 3	**1**, **2**, **4**
6	0, 3	**0**, **3**, **5**
7	1, 2	**0**, **2**, **3**
8	1, 2	**1**, **4**, **5**
9	1, 3	**0**, **4**, **5**
10	1, 3	**1**, **2**, **3**
11	2, 3	**0**, **1**, **3**
12	2, 3	**2**, **4**, **5**

(b) 第 2 エポック

順序	1 層	2 層
1	2, 3	**2**, **4**, **5**
2	0, 1	**0**, **1**, **2**
3	0, 1	**3**, **4**, **5**
4	0, 2	**1**, **2**, **5**
5	0, 2	**0**, **3**, **4**
6	0, 3	**1**, **2**, **4**
7	0, 3	**0**, **3**, **5**
8	1, 2	**0**, **2**, **3**
9	1, 2	**1**, **4**, **5**
10	1, 3	**0**, **4**, **5**
11	1, 3	**1**, **2**, **3**
12	2, 3	**0**, **1**, **3**

文献 [3] ではドロップアウトデザインを適用するネットワーク構成について特に仮定は設けられていない．本論文では，MLP を対象として，これらのネットワークの全結合層にドロップアウトデザインを適用する．そのためには，各層の出力ベクトルの各成分とデザインの各点とを一対一に対応づければよい．

4. 評価実験

本節では，計算機による評価実験の結果を報告する．本実験の目的はドロップアウト法に対する優位性を評価することではなく，ドロップアウトデザインによる正則化とドロップアウト法による正則化の効果に違いがあるか比較検討することで，ドロップアウトデザインの性質を調べることである．

4.1. 実験設定

本実験では，深層ニューラルネットワークの全結合層に対して，訓練時の性能と，テストデータを用いた時の汎化性能の評価を行った．性能評価は，ドロップアウトデザインを用いた正則化を施した場合，ドロップアウト法を用いた正則化を施した場合，正則化を行わない場合を対象とした．

本実験で扱う問題は画像分類問題とした．データセットには CIFAR-10 [5] を使用した．CIFAR-10 は，50,000 個の訓練データと 10,000 個のテストデータからなる 10 クラス画像分類用データセットである．各データは 32×32 ピクセルのカラー画像である．3 節で述べたように，実験を行うニューラルネットワーク構成は MLP とした．ドロップアウトデザインとドロップアウト法はネットワークの全結合層だけに適用した．使用するドロップアウトデザインは，4 種類の $(2, 1)$ かつ $(1, 2)$ 型-ドロップアウトデザインとした．全結合層の構成は，実験で使用するドロップアウトデザインに基づいて定めた．表 2 に，実験で扱うドロップアウトデザインと，対応する MLP の全結合層の構成を示す．4 種類の各ネットワークに対してドロップアウト法を適用する場合には，各層のドロップアウト率を表 2 にある値に設定した．この値は，ドロップアウトデザインによって定まるノードの不活性率と同一の値である．ドロップアウトデザインを適用する際には，ブロックシフトを実行する場合と，ブロックシフトを行わず，どのエポックでも同一順序でスーパーブロックを適用する場合の二通りを評価した．スーパーブロックの使用順序の性能への影響を調べるためである．

各ネットワークの損失関数には交差エントロピーを，パラメータの更新には確率的勾配降下法（SGD）を使用した．また，入力層と全結合層の活性化関数は ReLU 関数とし，出力層の活性化関数は Softmax 関数とした．

実験では，各ネットワーク構成について，500 エポックの学習を行った．その後，テストデータに対する精度（正解率）と，交差エントロピーに基づく損失を評価した．精度と損失はテストデータの正解クラスについて求めた．この実験をそれぞれのネットワーク構成について 10 回ずつ行った結果を評価した．ドロップアウト法とドロップアウトデザインの正則化効果だけを識別しやすくするため，データ拡張などの他の汎化性能を向上させる技法は使用しなかった．

実験プログラムは Python 3 で実装した．ニューラルネットワークの訓練ならびにテストの実装には，Keras ライブラリを用いた．ドロップアウト法には，Keras が提供するドロップアウト層の実装を使用した．一方，ドロップアウトデザインに関しては，アフィン空間を利用したドロップアウトデザインの構成法（2.2 節）を実装した．また，Keras のドロップアウト層と同じ実装方法でデザインを用いたノードの活性化を実現するために，従来のドロップアウト層を拡張したデザイン適用層を新たに実装した．このデザイン適用層を用いて，実験プログラムは，構成したデザインから層ドロップアウトマス

表 2: 実験で使用したドロップアウトデザインとネットワーク構成の対応

アフィン空間			ドロップアウトデザイン					ニューラルネットワーク（全結合層）			
次元 d	位数 q	flat の 次元 t	サブ ブロック 数	点 集合 サイズ	サブ ブロック サイズ	スーパー ブロック 数	会 合 数	層 数	1 層の ノード 数	1 層の 活性化 ノード数	ドロップ アウト 率
8	2	7	2	128	64	508	63	2	128	64	0.5
7	3	6	3	729	243	3276	121	3	729	243	0.67
9	2	7	4	128	64	1016	126	4	128	64	0.5
10	2	8	4	256	128	2040	254	4	256	128	0.5

クを生成して，(1) 式に従って訓練を実行する．テスト時には，ドロップアウト法と同様に (2) 式により出力を計算することとした．

実験には，CPU に Intel Xeon W-2133 (3.60GHz)，GPU に NVIDIA TITAN RTX を備えたマシン（Keras 2.3.1，Tensorflow 2.1.0）と，CPU に Intel Xeon W-2245 (3.90GHz)，GPU に NVIDIA TITAN RTX を備えたマシン（Keras 2.4.3，Tensorflow 2.3.0）を使用した．

4.2. 実験結果

MLP の全結合層を 2 層，3 層，4 層にした場合の実験結果をそれぞれ 図 1，図 2，図 3，図 4 に示す．測定した値は，10 回の実験に対する精度の平均と標準偏差，損失の平均と標準偏差である．加えて，第 51 エポックから第 500 エポックまでの各エポックについて，そのエポックを含む過去 50 エポックの区間に対する精度と損失の分散を算出した．これは，訓練の進行に伴う，実験ごとの汎化性能の収束の違いを評価する指標である．各グラフは，正則化を行わなかった結果（**w/o dropout**），ドロップアウト法による正則化を行った結果（**dropout**），ブロックシフトを適用した場合（**design(bs)**）と適用しない場合（**design**）のドロップアウトデザインでの正則化の結果をそれぞれ示している．グラフの横軸はエポック数である．

図 1 から図 4 から，以下の三点が観察された．

1. 本実験では，ドロップアウトデザインを用いることで汎化性能が高まるかどうかを調べるため，精度と損失の平均，標準偏差，分散を測定した．それらすべての項目について，ドロップアウトデザインを用いた正則化は，従来のドロップアウト法と同程度の汎化性能を達成した．
2. 無作為にノードを不活性にする従来のドロップアウト法と異なり，ドロップアウトデザインは活性化するノードの回数に制約を設けた決定論的な組合せ構造である．そのため，ドロップアウトデザインの固有の特徴を示す結果が得られることが期待される．実験の結果からは，精度と損失に関して，ドロップアウトデザインとその適用法に特有の性質や挙動は確認できなかった．
3. ドロップアウトデザインの適用において，ブロックシフトを実行した場合としない場合を比較したとき，精度と損失の平均，標準偏差に大きな違いは確認されなかった．この結果は，スーパーブロックの使用順序は汎化性能に影響しないことを示していると考えられる．

ドロップアウト法との正則化の違いを評価するという点から，まず，上で述べた結果 1 と 2 を詳しく検討する．本実験の結果から，表 2 に示したすべてのネットワーク構成について，ドロップアウト法とドロップアウトデザインを用いた正則化は平均的に同程度の精度と損失であり，各曲線の収束の仕方にも大きな違いは見られなかった．ただし，精度と損失の平均値に関しては，全結合層が 4 層，各層のノード数が 128 の場合には，両者の挙動にわずかながら違いを確認した．訓練データ，テストデータ共に平均精度はドロップアウト法が上回り（図 3a，3b），ドロップアウト法が低損失になるという傾向が観察された（図 3c，3d）．また，同じ構成について，精度の標準偏差の結果（図 3e，3f）から，ドロップアウト法を用いた場合は，ドロップアウトデザインを用いる場合よりもわずかに低い標準偏差を示した．このような結果が得られた原因として，中間層数が最も大きいネットワークを使用していることが考えられる．しかし，各

層を 256 ノードとした結果（図 4e，4f）では同様の傾向は見られないことから，ドロップアウトデザインの固有の特徴を示す結果とはいえないと思われる．ニューラルネットワークの訓練とドロップアウト法における確率的な挙動が影響している可能性もあり，本論文で行った実験だけからでは結論を得ることは難しい．今後，5 層以上の深いネットワーク対する評価実験が必要である．50 エポック区間の精度分散，損失分散に関しては，全結合層 2 層，3 層，4 層のいずれの場合も，ドロップアウト法とドロップアウトデザインを用いた正則化と顕著な違いは見られなかった．以上の考察から，訓練の進行に伴う精度や損失の収束の仕方には大きな違いはないと結論付けられる．

次に，ドロップアウトデザインの結果に注目して，ブロックシフトを実行した場合としない場合を比較した場合を議論する．上の 3 で述べたように，すべての実験を通じて，精度と損失の平均，標準偏差に大きな違いは確認されなかった．なお，全結合層 4 層，各層 128 ノードに対する 50 エポック区間の精度と損失の分散（図 3i, 3j）において，ブロックシフトの方が訓練の初期段階での分散値が低くなる傾向が見られるが，その差は微小である．

最後に，実験結果への影響を与える要因として，過学習の可能性と収束速度について検討する．正則化を行わない場合については，全結合層 3 層（図 2a）と 4 層かつ各層 256 ノード（図 4a）のそれぞれにおいて，短いエポック数の訓練で精度が 1.00 に達して収束し，また，標準偏差が 0 になった．そのため，訓練の進行と共に過学習を引き起こしている可能性が高い．全結合層 4 層，各層 128 ノードの場合は，図 3e においても，エポック数の増加に伴いテストデータに対する精度が低下する傾向があり，過学習となっている恐れがある．これらのネットワーク構成では，正則化技法を用いない場合に最も短い収束速度で高い汎化性能が得られた，という結果を示している．以上の議論から，この現象は過学習が原因である可能性がある．ドロップアウト法とドロップアウトデザインを用いた場合には，過学習の傾向は見られなかった．全結合層 3 層と 4 層については，訓練時の平均精度が収束まで達しておらず，正則化技法を用いない場合と比較して，収束に遅れが見られる．訓練するエポック数をさらに増やすと，精度や損失の結果に違いが現れる可能性がある．

5. 関連研究

ドロップアウト法は後に続く正則化法の研究に大きな影響を与えた．代表的な正則化法の一つにドロップコネクト法 [6] がある．ドロップコネクト法は，学習時にノード間の接続を無作為に切断する方法である．一方，高速ドロップアウト法 [7] は，ガウス近似によりドロップアウト法と同様の正則化効果を発生させることで，学習時の収束速度を高める方法である．より最近では，畳み込み層の出力の周波数成分に対してドロップアウト法を適用するスペクトラルドロップアウト法 [8]，ドロップアウト法を用いた訓練をベイズ推定で近似する MC ドロップアウト法 [9, 10]，学習効果の高い部分ネットワークを進化計算を用いて絞り込む EDropout [11]，強化学習でドロップアウトマスクを学習する AutoDropout [12] などの多くの方法が提案されている．また，R-Drop は，ドロップアウト法で部分ネットワークを二つ構成し，両者が推定した分布の差異を低減するように訓練を行う正則化法である [13]．ドロップアウト法と深く関係する主要な技法は文献 [4] で詳しく論じられている．

組合せデザインをはじめとする組合せ構造の計算機科学への応用には，数多くの研究がある [14]．デザインによるバギング [15, 16] は，バギングにおいて組合せデザインの考え方に基づきデータをサブサンプリングすることで，汎化性能を改善する技法である．バギングは複数の弱分類器で学習を行うアンサンブル学習の一種である．ドロップアウト法もアンサンブル学習と考えることもできるので，デザインによるバギングは，組合せデザインを機械学習に適用する先駆的な試みの一つである．また，組合せデザインと関連が深い分野に実験計画法や品質工学（タグチメソッド）があり，ニューラルネットワークの最適な構成を求める技法 [17, 18] をはじめ，機械学習での活用が研究されている．

6. おわりに

深層学習における正則化に関しては，数多くの研究がなされてきた．本論文で扱ったドロップアウトデザインを用いた正則化技法は，部分ネットワークの組合せ構造に注目した方法である．無作為にノードを不活性にするドロップアウト法とは異なり，ドロップアウトデザインは，ブロックによってノードの活性化回数を釣り合わせるという特徴を持つ．本論文では，ドロップアウトデ

ザインによる正則化の持つ性質を明らかにするために，MLP の全結合層を正則化する技術を実装した．加えて，画像分類問題での評価実験により，全結合層が 2，3，4 層の時の汎化性能を分析した．その結果，いずれの場合についても，ドロップアウト法と同程度の汎化性能が得られた一方で，ドロップアウトデザイン固有の特徴的な結果は得られなかった．これらのことは，訓練時に使用したドロップアウトデザインが持つ組合せ論に基づいたネットワーク構造と，従来のドロップアウト法で得られる無作為な構造との間に，関係性があることを示唆していると考えられる．ドロップアウトデザインの分析を通じて，組合せ構造を用いてドロップアウト法を理解する研究が進むものと期待される．ドロップアウトデザインのニューラルネットワークへの適用法の違いは，実験結果に大きな影響を与えることはなかった．

今後，ドロップアウトデザインとドロップアウト法についてさらに深く理解するために，以下の課題に取り組む必要がある．まず，引き続きより深いニューラルネットワークや，画像分類以外の問題についても実験を進めていく必要がある．また，本論文での評価実験は $(2,1)$ かつ $(1,2)$ 型-ドロップアウトデザインに限った結果であり，ドロップアウトデザインの型が汎化性能に与える影響を分析する必要がある．文献 [19] のように，ドロップアウトデザインの構成法の研究は進展しているが，様々な型のドロップアウトデザインを理論的に構成することは現時点では困難である．そのため，従来のドロップ後法と異なり，ドロップアウト率を自由に設定して訓練を行うことができないという制約がある．そこで，計算機によりドロップアウトデザインを構成する技術を開発することが必要である．計算機を利用して組合せ構造を探索する研究 [20, 21, 22] が参考になると考えられる．最後に，本論文の実験では，ニューラルネットワークのドロップアウトデザインへの適用法の有効性を確認できなかった．今後も有効な適用法を検討する必要がある．

謝辞

本研究は JSPS 科研費 基盤研究 (C) 19K11866 及び，若手研究 21K13845 の助成を受けたものです．

参考文献

[1] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[3] Shoko Chisaki, Ryoh Fuji-Hara, and Nobuko Miyamoto. Combinatorial designs for deep learning. *Journal of Combinatorial Designs*, 28(9):633–657, 2020.

[4] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *arXiv:1904.13310*, 2019.

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[6] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using DropConnect. In *30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066. PMLR, 2013.

[7] Sida Wang and Christopher Manning. Fast dropout training. In *30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 118–126. PMLR, 2013.

[8] Salman H. Khan, Munawar Hayat, and Fatih Porikli. Regularization of deep neural networks with spectral dropout. *arXiv:1711.08591*, 2017.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. PMLR, 2016.

[10] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 1027–1035, 2016.

[11] Hojjat Salehinejad and Shahrokh Valaee. EDropout: Energy-based dropout and pruning of deep neural networks. *arXiv:2006.04270*, 2020.

[12] Hieu Pham and Quoc Le. AutoDropout: Learning dropout patterns to regularize deep networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9351–9359, 2021.

[13] Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. R-drop: Regularized dropout for neural networks. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[14] Charles J. Colbourn and Paul C. van Oorschot. Applications of combinatorial designs in computer science. *ACM Comput. Surv.*, 21(2):223–250, 1989.

[15] Periklis Papakonstantinou, Jia Xu, and Zhu Cao. Bagging by design (on the suboptimality of bagging). *AAAI Conference on Artificial Intelligence*, 28(1), 2014.

[16] Mahdi Hamdani, Patrick Doetsch, and Hermann Ney. Bagging by design for continuous handwriting recognition using multi-objective particle swarm optimization. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 256–260, 2015.

[17] F. Sánchez Lasheras, J. A. Vilán Vilán, P. J. García Nieto, and J. J. del Coz Díaz. The use of design of experiments to improve a neural network model in order to predict the thickness of the chromium layer in a hard chromium plating process. *Mathematical and Computer Modelling*, 52(7–8):1169–1176, 2010.

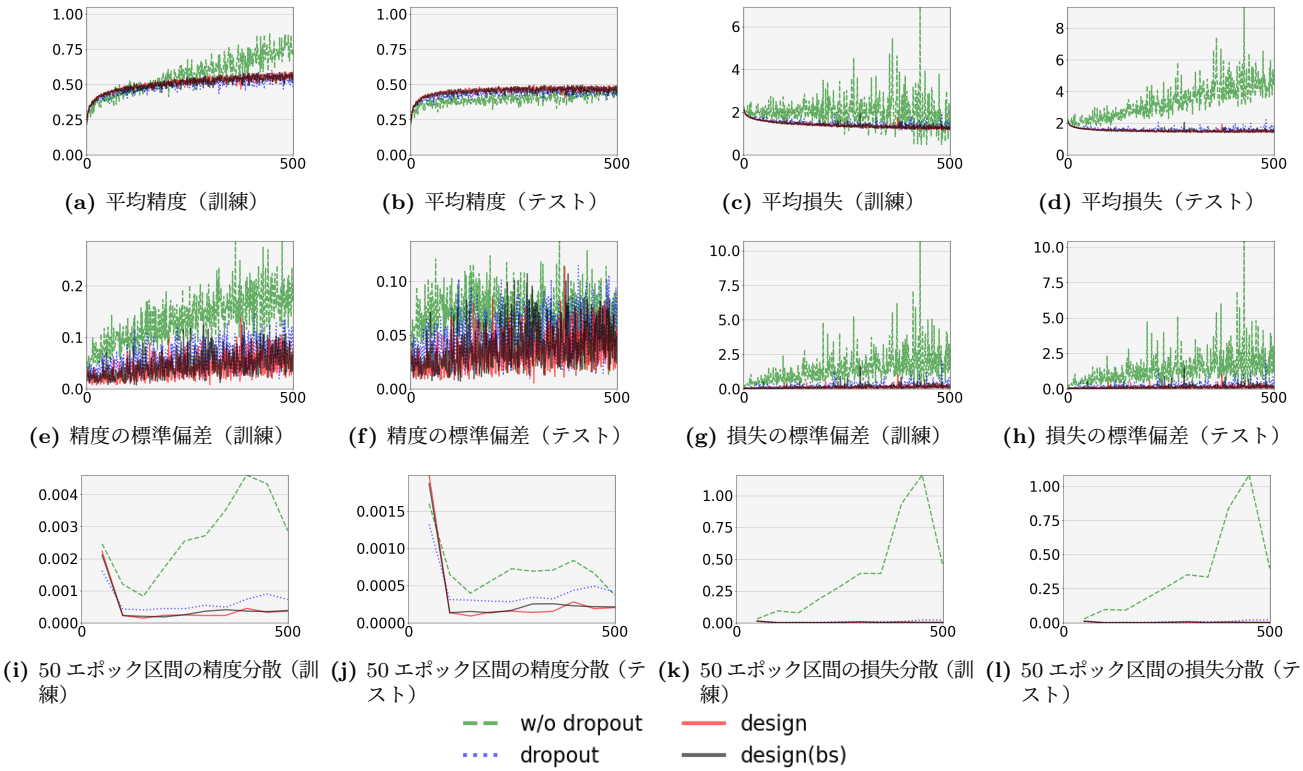
[18] Ahmad M. Karim, Mehmet S. Güzel, Mehmet R. Tolun, Hilal Kaya, and Fatih V. Çelebi. A new generalized deep learning framework combining sparse autoencoder and Taguchi method for novel data classification and processing. *Mathematical Problems in Engineering*, 2018:3145947, 2018.

[19] Shoko Chisaki, Ryoh Fuji-Hara, and Nobuko Miyamoto. A construction for circulant type dropout designs. *Designs, Codes and Cryptography*, 89(8):1839–1852, 2021.

[20] 松中春樹; 丹生智也; 番原睦則; 田村直之. SAT 符号化を用いた釣合い型不完備ブロック計画の構成. **人工知能学会論文誌**, 27(2):10–15, 2012.

[21] 青柳卓; 宮本暢子; 篠原聡. Optical orthogonal codes の探索 II. **明星大学研究紀要（情報学部）**, 22:15–22, 2014.

[22] B. N. Mandal, Rajender Parsad, and Sukanta Dash. Construction of a-optimal balanced treatment incomplete block designs: An algorithmic approach. *Communications in Statistics - Simulation and Computation*, 49(6):1653–1664, 2020.

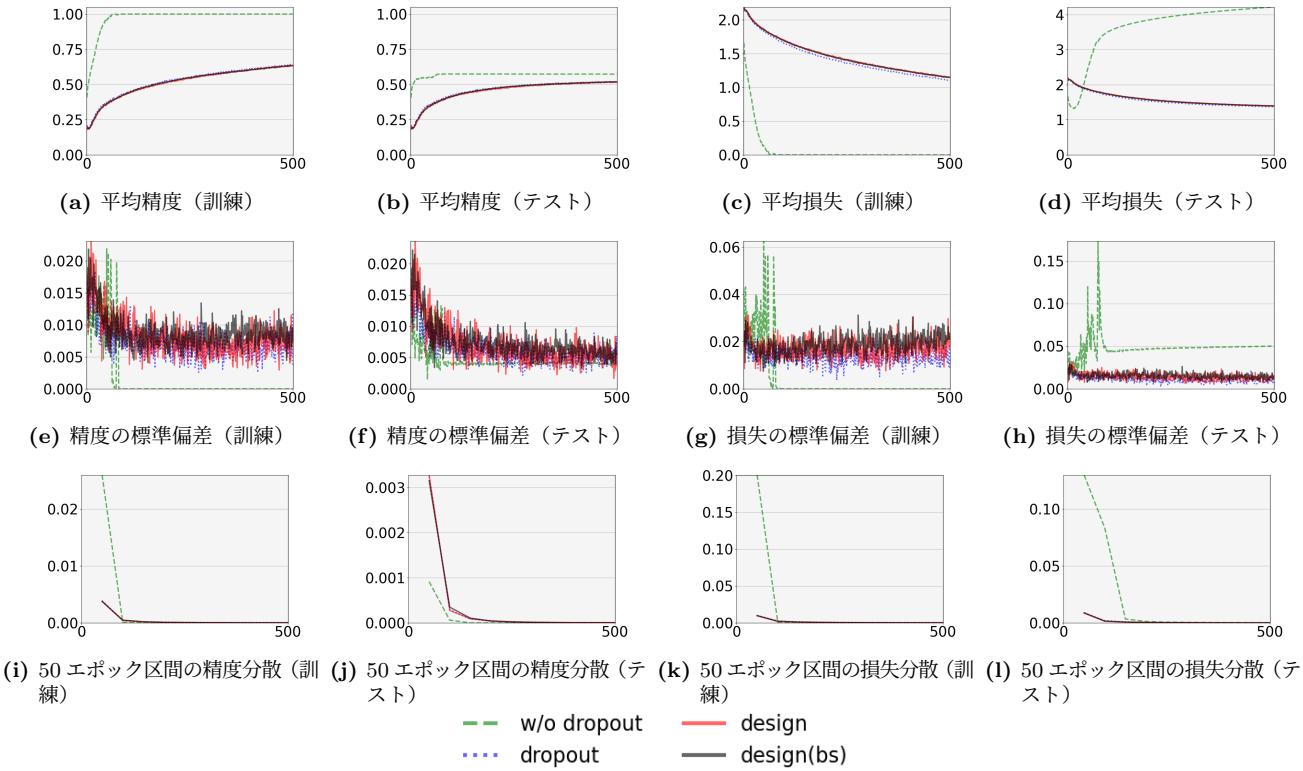


(a) 平均精度（訓練） (b) 平均精度（テスト） (c) 平均損失（訓練） (d) 平均損失（テスト）

(e) 精度の標準偏差（訓練） (f) 精度の標準偏差（テスト） (g) 損失の標準偏差（訓練） (h) 損失の標準偏差（テスト）

(i) 50 エポック区間の精度分散（訓練） (j) 50 エポック区間の精度分散（テスト） (k) 50 エポック区間の損失分散（訓練） (l) 50 エポック区間の損失分散（テスト）

図 1: 全結合層 2 層，1 層当たり 128 ノード，ドロップアウト率 0.5 の実験結果

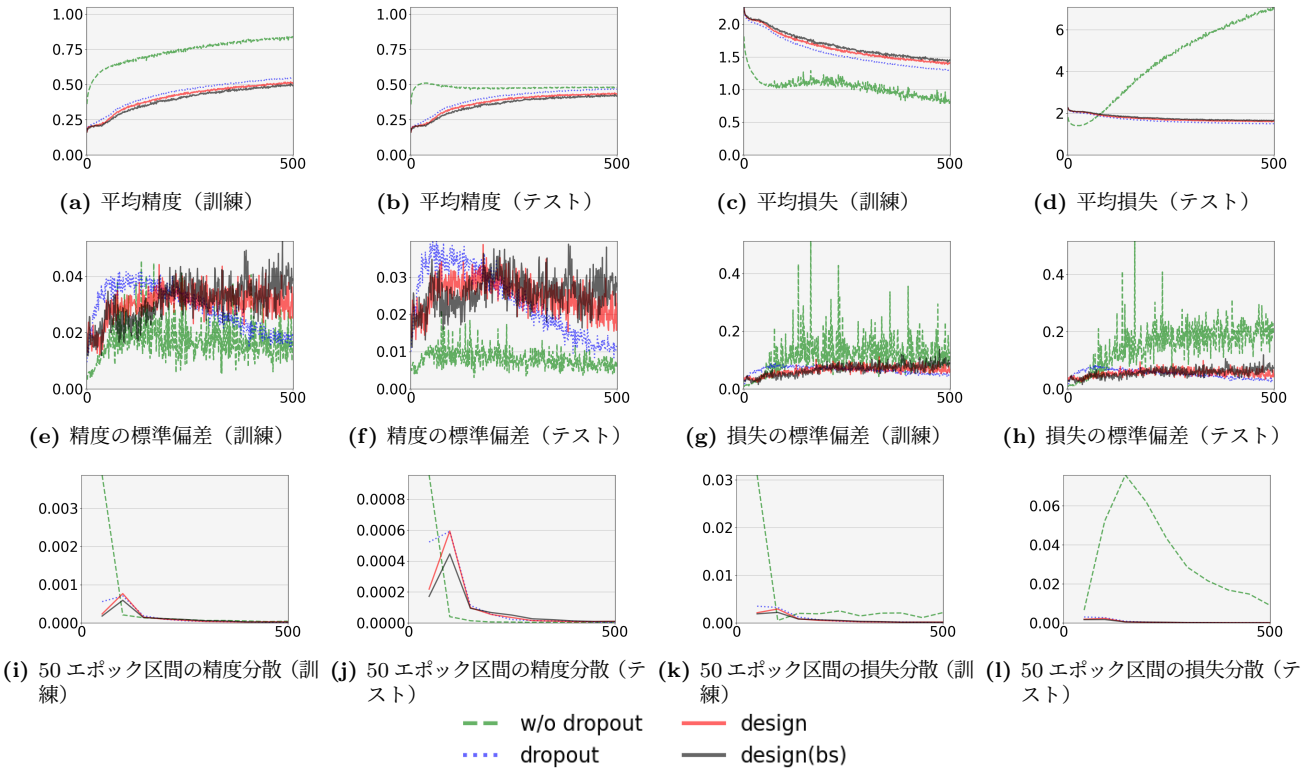


(a) 平均精度（訓練） (b) 平均精度（テスト） (c) 平均損失（訓練） (d) 平均損失（テスト）

(e) 精度の標準偏差（訓練） (f) 精度の標準偏差（テスト） (g) 損失の標準偏差（訓練） (h) 損失の標準偏差（テスト）

(i) 50 エポック区間の精度分散（訓練） (j) 50 エポック区間の精度分散（テスト） (k) 50 エポック区間の損失分散（訓練） (l) 50 エポック区間の損失分散（テスト）

図 2: 全結合層 3 層，1 層当たり 729 ノード，ドロップアウト率 0.67 の実験結果

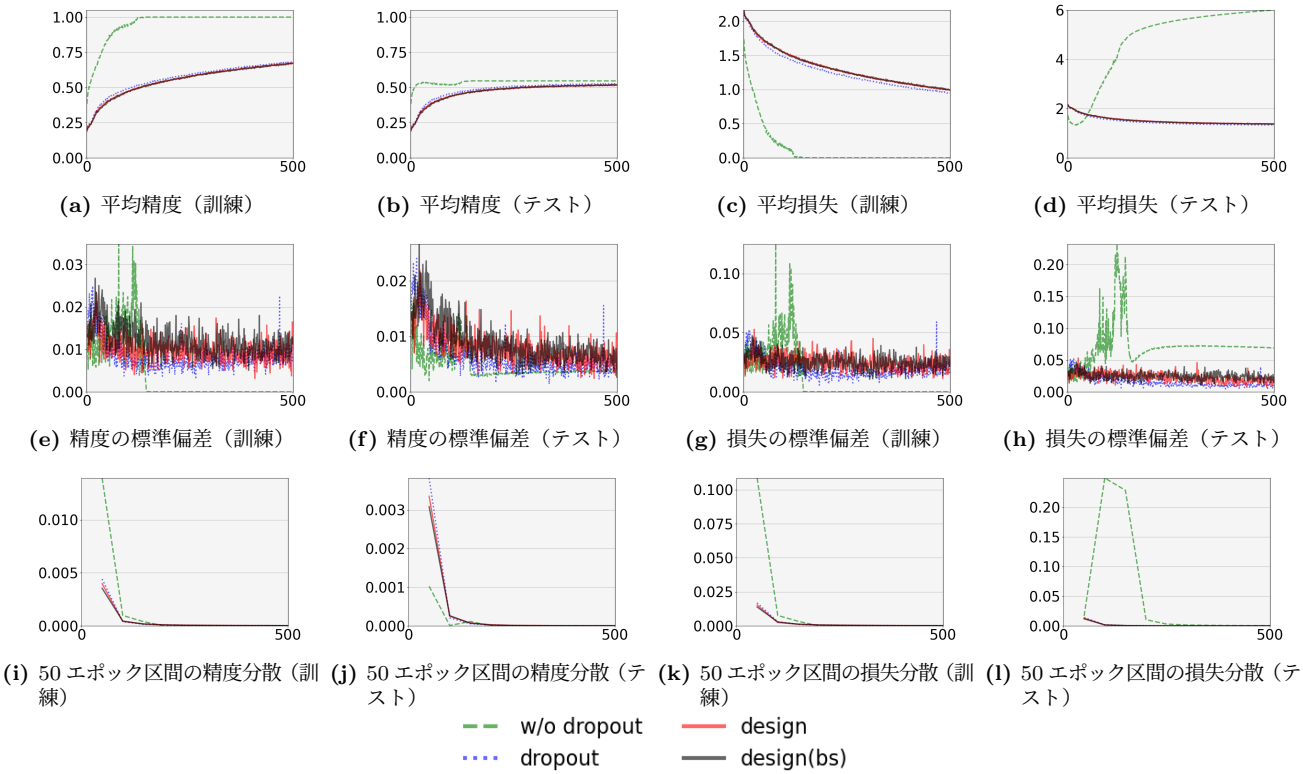


(a) 平均精度（訓練） (b) 平均精度（テスト） (c) 平均損失（訓練） (d) 平均損失（テスト）

(e) 精度の標準偏差（訓練） (f) 精度の標準偏差（テスト） (g) 損失の標準偏差（訓練） (h) 損失の標準偏差（テスト）

(i) 50 エポック区間の精度分散（訓練） (j) 50 エポック区間の精度分散（テスト） (k) 50 エポック区間の損失分散（訓練） (l) 50 エポック区間の損失分散（テスト）

図 3: 全結合層 4 層，1 層当たり 128 ノード，ドロップアウト率 0.5 の実験結果



(a) 平均精度（訓練） (b) 平均精度（テスト） (c) 平均損失（訓練） (d) 平均損失（テスト）

(e) 精度の標準偏差（訓練） (f) 精度の標準偏差（テスト） (g) 損失の標準偏差（訓練） (h) 損失の標準偏差（テスト）

(i) 50 エポック区間の精度分散（訓練） (j) 50 エポック区間の精度分散（テスト） (k) 50 エポック区間の損失分散（訓練） (l) 50 エポック区間の損失分散（テスト）

図 4: 全結合層 4 層，1 層当たり 256 ノード，ドロップアウト率 0.5 の実験結果

OSS の参加者は何を求め，何を得ているのか？ - 利用者と参加者の行動の視覚化 -

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@biglobe.jp

要旨

オープンソース・ソフトウェア *(OSS : Open Source Software)* 作品は広く利用され，利用者視点からの分析は消費者行動の枠組みで調査・研究が行われている．一方，その作品を支えているエンジニア *(*広く参加者と呼ぶ*)* の行動についての調査・分析には課題がある．オープンサイエンスにおけるナレッジの有用性評価の一つの指標として参照量が考えられるが，ナレッジが持つ価値は多様であり，参照量だけでは評価できない．我々はエンジニアが *GitHub* からどのように学び，成長するのかについて，ナレッジの参照や影響を有用性の特性として捉え，指標化を試みている．我々の研究目的はエンジニアの育成や組織化における「意欲」を *OSS* 活動の「行動」から学び活用することである．ここでは，その入口として探索的な計測と分析について報告する．

課題は,「行動」を分析するために何を測りどのように分類するのか，である．本研究では，有用性の特性を捉えるため，時系列での型の違いに注目した．本研究では，時系列分析を用いて行動記録の視覚化を行い，他の分野では見られない特徴的な型を抽出できることを示す．この分析の前処理としての課題に，膨大な数の対象から，分析対象を合理的に選び出すための層別がある．この課題については順位を用いたクロス表により対処した．

1. はじめに

現代の ICT (Information and Communication Technology) 社会において，オープンソース・ソフトウェア (OSS : Open Source Software) 作品は，高度化し，かつ広く利用され，社会インフラとなっている．先端企業がビジネス戦略として先端技術を OSS に公開することもあり，利用技術中心に膨大な数のエンジニアが参加し活動を支えている．

エンジニアの働き方や意欲という観点から，彼らの活動の推進力を研究し活用できれば有用性は非常に高いが，調査や分析には課題がある．OSS として一括りにするには，あまりにも対象が大きすぎて多様であることと，行動を測り分析する方法が開発されていないことが要因であると考える．

オープンサイエンスにおけるナレッジの有用性評価の一つの指標として参照量が考えられるが，ナレッジが持つ価値は多様であり，参照量だけでは評価できない．我々は，インターネット・コミュニティ[1)] における，エンジニアの働き方に関して動的な観点から研究しており，GitHub [2] のデータを用いたデータサイエンスの探索的な分析 [3] を行っている．具体的には，エンジニアが GitHub のナレッジからどのように学び，成長するのかについて，ナレッジの参照や影響を有用性の特性として捉え，指標化を試みている．ナレッジの参照や影響を指標化するには，ナレッジの有用性を特性に分解し焦点を絞る必要がある．本研究では，この分解の手段として，GitHub のタイムスタンプによる記録データを時系列で視覚化する方法を試みた．

本研究では，アンケートやインタビューなどの「認知」アプローチではなく，log 情報等の記録を用いた「行動」アプローチでの分析を試みる．行動を特徴づける変数として時系列に着目し，作品の利用者と参加者の行動記録を分析した．分析の結果，得られた行動を特徴づける

「型」が観察された．ここでは，観察された型と，分析の方法について報告する．

この研究の前段階として，膨大な OSS 作品の中から，エンジニアの行動を探るための調査対象をどのように絞り込むのかという課題がある．本研究では，順位データを用いたクロス表を用いて対応した．

3 章では，分析に用いたデータの説明と，べき分布の視覚化，利用者行動と参加者行動の組合せを層別した結果を示す．4 章では，時系列での視覚化の方法とその結果を示し，5 章で時系列的な相関について分析し，特徴的な結果を示す．最後に，6 章で本研究のまとめとして総括する．

2. 行動に着目する理由と分析の手段

我々はエンジニアの働き方や意欲の研究対象として，OSS 作品を選び，その参加者と利用者について「行動」からの分析を試みている．本章では，本研究の背景や進め方について述べる．

2.1. 研究の背景

本研究で対象とするのは，OSS 作品の参加者と利用者の行動である．利用者の作品に対する評価や選択に関する研究は，消費者行動として盛んに行われ，ランキングなどが公開されている．一方，参加者の行動は就業活動や企業内での業務活動とは明らかに異なるが，その詳細は明らかになっていない．「OSS の参加者」の行動とは，作品を提供している何らかの組織に参加し，協働することである．何らかの組織とは，一種のインターネット・コミュニティであるが，OSS 作品を開発し公開するなど，何らかの活動目的を持っており，参加自由な交流の場としてのコミュニティとは異なる．

「プロのエンジニアでも難儀な作品に向かって活躍する彼らの原動力は何か」,「原動力はどこにあるのか」,「コミュニティとの関係はどのようなものなのか」など，これらの謎が解ければ，エンジニアの成長と働き方に寄与できると考えている．

2.2. どこに集まるのか

本研究で調査対象とした GitHub [2] の作品提供の基となる Repository[2] の新規登録数は，1 日で 数万件を超えている．既存の Repository に新規に参加する参加者の数も膨大だと推測される．

そこで，参加者の行動を調べる対象として，膨大な数の作品のどこに利用者や参加者が集まるのかを調べ，サンプルを層別して絞り込む必要がある．この課題の背景には，OSS における量的な分布の型がべき分布であるということがある [4]．べき分布を可視化するには，Zipf グラフ [5,6] や，分布の割合に着目したローレンツ曲線 [7–9] が有用である．これらの先行研究では，順位に基づく分析が行われていることから，本研究でも順位を使った層別を試みた．この層別については，3 章で述べる．

2.3. 時系列変化はあるのか

ソフトウェアの活動分析で時系列を扱う場合は少ない．バグ成長曲線や課題管理の見積もりと実績に用いる程度で，科学的な技法の応用には至っていない．

データサイエンスにおける時系列分析は，さまざまな分野で予測に利用されている実用技術である [10,11]．時系列分析では，定期周期でサンプリングされた情報 (時系列データ[3]) を用意して，変化の成分を「トレンド[4]」「周期変動[5]」「ノイズ[6]」に分解してモデル化を行う．

ここでの分析目的は，多種多様な作品や参加者の行動に対し，共通した尺度で比較することが出来る分析手法である．分析の精度は，定量的なモデル化まで求めず，探索的な分析で定性的な差を視覚化することを求めている．この実用化については，4 章で示す．

2.4. 時系列での相関はあるのか

OSS においては，利用者の評価が開発者をモチベートするのか，といった利用者と参加者の行動間の相関についても明らかになっていない．特に，時間軸で観測した行動間で相関がある場合があるのかについて分析する．

4 章で示すように，時系列の型は多様であるため，5 章では特徴的な型の組合せから利用者行動と参加者行動の影響の有無を概観する．明らかに影響していないケースが多数ある一方，連動しているものも観測された．これらの観測結果から，明らかになったことを 6 章でまとめ，報告する．

3. 分析対象とその分布

膨大な OSS 活動の場において，利用者や参加者はどこに集まるのか．本章では，膨大な OSS 作品の中から，参加者である開発者の行動分析のための調査対象をどのように絞り込み，収集するのかという課題に対し，事例を通して考える．3.1 節で分析データについて説明し，3.2 節で分布特性を考慮し，3.3 節で分析データを用いた層別を例示する．

3.1. 分析データ

本研究で用いるデータは，GitHub から取得したものであり，参加者や利用者の識別は，GitHub の Repository を単位に行っている．詳細な定義については GitHub の資料 [12] に従っているが，本研究で用いる主なものは次の 2 つである．

- **利用者の行動を代用する特性**：Repository に付与された Star[7] の数や付与した利用者と付与日などのデータ．
- **参加者の行動を代用する特性**：Repository の Issue[8] の数や起票者，起票日などのデータ．

分析には，次に示すサンプルデータ A と サンプルデータ B の 2 種類のデータを用いた．

- **サンプルデータ A (以降，サンプル A と記す)**：n = 1,287．2015 年 3 月 15 日から 1 週間の間に新規に登録された 117,757 件から 6 年経過後も保守が行われているものを抽出した．
- **サンプルデータ B (以降，サンプル B と記す)**：n = 1,395．先行研究 [4] で用いたデータを 2022 年 2 月現在の値でアップデートし，抽出した．データの概略は，2010 年 1 月 1 日から 2019 年 5 月 31 日までの期間内の 200 日をサンプリングして取得したものである．

データの収集は両サンプルとも GitHub API v3 [15] を用いて行った．

3.2. べき分布の扱い方

Star 数の分布や，Issue 数の分布はべき分布と呼ばれる分布を示し，正規分布を基礎とする一般的な統計手法では取り扱うことが困難である．具体的な例で示すと，サンプル A の作品が獲得した総 Star 数は，435,295 件であり，上位 12 件 (0.9 %) で総 Star 数の 50 % を獲得している．

この種の分布を扱う方法としては，順位と対数変換した獲得数や活動量 (Star 数や Issue 数) を用いる方法がある．図 1 は，サンプル A の Star 数の密度分布を示し，図 2 はサンプル A の Star 数，サンプル B の Star 数と Issue 数の対数変換 (y 軸) と順位 (x 軸) を示している．

対数変換し順位で並べると直線に近づく特性は，文書中の単語の出現数の研究において古くから知られており，Zipf の法則と呼ばれている [5,6,16]．もう一つの表現方法は，経済学で用いられているローレンツ曲線であり，母集団の大きさに差があっても比較することができる [7–9]．例としてサンプル A と B の Star 数のローレンツ曲線を図 3 と図 4 に示し，差を確認した．

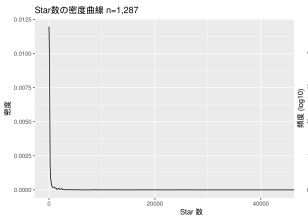


図 1: A：Star 数の分布

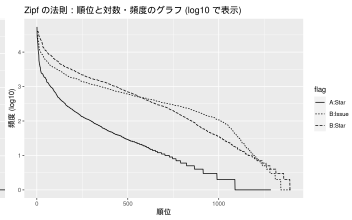


図 2: B：Zipf グラフ

図 3: A：Star 数

図 4: B：Star 数

分布の数値まで見たい場合には図 2 で示した Zipf グラフが，分布の割合を他と比較したい場合には図 3 と図 4 で示した正規化して視覚化されるローレンツ曲線が有用である．いずれも順位を基にデータを取り扱うので，本研究でも順位を使った層別を行う．

3.3. データの層別

4 章以降では，サンプル B を対象に，さらに詳細な分析として Repository 単位で時系列分析を行う．合理的に研究を進めるには，データを層別して探索的な分析を進める必要がある．ここでは，利用者行動 (Star) と参加者行動 (Issue) の 2 変数から層別を行う．

連続量である順位をカテゴリ化して，クロス表を作成する．カテゴリ化はデータを四分割して行った．Star 数を少ない順に {A,B,C,D}，同様に，Issue 数を {a,b,c,d} と表し，表 1 に示すクロス表を作成した．

表 1: Star 数と Issue 数のクロス表

		Issue 数			
Rank		a	b	c	d
Star 数	A	227	72	25	22
	B	92	116	91	51
	C	22	96	124	108
	D	8	63	110	168

このクロス表を基にした層別の結果およびコミュニティ特性の差については，4 章で視覚化して示す．

4. 時系列特性

本章では，活動の変化を時間経過から捉え，さらに層別を行う．時間経過による分析は，時系列分析と呼ばれ「時系列データ」を用いる．本研究では，時系列分析を行うに当たり，データ収集や前処理に課題がある．この課題について，4.1 節で述べる．

もう一つの課題は，膨大な数の作品の中から，調査にコストのかかる時系列分析対象をどのように選ぶのかである．4.2 節では，3 章の分析結果を利用し，精度の高い時系列分析が可能で，時系列のタイプ分けが可能なサンプル集団について述べ，4.3 節では，実際に層別した時系列の型について，参加者と利用者に分けて紹介する．

4.1. 時系列の視覚化の方法

時系列分析を用いた研究では，時系列の数学モデルを作成し，例えば季節商品の需要予測など定量的な取り扱いができる [10, 11]．モデルの要素は「トレンド」と呼ばれる時間的な定常状態と「周期変動」と「ノイズ」である．

ここでの分析は，探索的に時間特性を概観することが目的であるため，数学モデルには立ち入らず，そもそもトレンドのような定常状態が存在するのか，といった事前調査的な位置付けである．そのため，分析に用いるデータも，整形された時系列データではなく，利用者や参加者の活動記録のタイムスタンプを利用する．具体的な手法は，タイムスタンプのあるデータ列を時間順に並べ，平滑化帯域幅を用いて帯域内のサンプル数を数え，数えた数を視覚化して表示し，時系列を示す．原理的には，対象とするデータに活動記録の欠損がなければ，正確な時系列データと同じ値を得ることができる．ただし，利用者や参加者の活動頻度が低い場合には，時間周期を長くしないと欠損になる．具体的なツールは R 言語の ggplot2 ライブラリで提供されている「geom_freqpoly」[17] を用いて求めた．

4.2. 探索対象の選択

時系列データの観測周期は，分析精度に影響を与える．株価の変動や電力消費の分析は，対象とする変動が短時間で生じるため短い時間間隔で行う必要がある．本研究は，参加者や利用者の特性変化を明らかにすることが目的であり，月間程度で十分と推測されたが，月間の活動数が平均 10 件としても，7 年間で 800 件を超えるデータが必要となる．

実際のデータを調べると，3 章の結果では，数年間で 100 件以下のデータが多数を占めており，探索の対象条件を設定して選ぶ必要がある．探索的分析の目的は，対象のトレンドや周期変動を検出するのに必要な周期を明らかにすることである．そのため，大量の時間観測点を含むデータを選定し，トレンドや周期変動の有無と変動の速さ (時間微分) を観測した．具体的には，1,000 件以上のデータを対象としたサンプル B の Star と Issue の観測値から Repository をサンプリングして探索を行った．

4.3. 探索の例とまとめ

まず，利用者行動に対して時系列特性の観測を実施した．作品の利用者行動は，一般には消費者行動に近いと考えられる．図 5 に観測された代表的な例を示す．

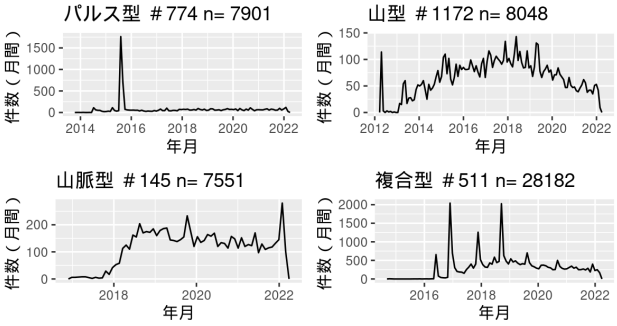


図 5: サンプル B の Star の時系列型例

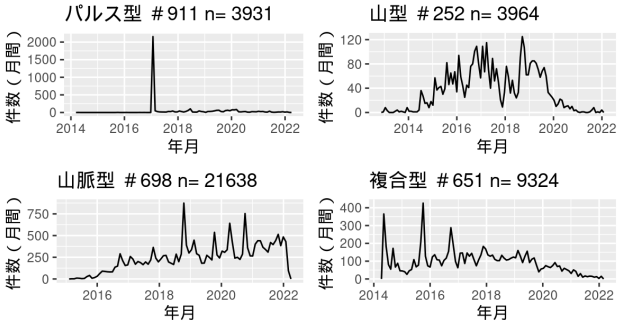


図 6: サンプル B の Issue の時系列型例

- **サンプル #774**：パルス型は，非常に短い期間 (2 〜 3 ヶ月) において，数年間で得られる評価のほとんどを獲得するパターンである．利用者の Star 評価がパルス以降にほとんど生じないが，利用していないことを表すものではない．新規に評価する利用者がいない状態を示している．本研究では，層別をテーマとしているため，その原因推測には立ち入らない．

- **サンプル #1172**：山型は，徐々に増加し，徐々に減少するもので，たとえば 10 年に渡るライフサイクル的な状態を示す．ここで選んだ事例は，最初にパルス的な傾向が見られる．また，途中のギザギザに周期性があるのか否かについては，目視判断なので不明である．

- **サンプル #145**：山脈型は，山型ほど明快な凹凸ではなく，なだらかな減少や増加を示す．ただし，時間軸の幅によっては，山型と変わらない．

- **サンプル #511**：複合型は，パルス型と山脈型や山型との組合せである．

作品の利用は，モノの購入や消費とは異なるので消費財の消費者行動とはかなり異なっている．「どのような分野と共通なのか」や「反応速度」など，詳細については次の課題である．ここで明らかになったことは，異なる型があり，OSS の共通特性ではないこと，パルス型のように非常に短期間に集中するものがあることである．

図 6 は，参加者側の時系列特性を示している．参加者側でも，利用者側と同様な型が観測されている．

- **サンプル #911**：パルス型は，利用者側では多く観測されたが参加者側では少数であった．短期間に月間で 1,000 件を超える案件が処理され，その後は月間数件で継続されるケースが観測された．サンプルデータ外になるが，東京都の Covid-19 サイト [18] も同様の振舞いであった．

- **サンプル #252**：この事例の山型は 6 年間ほどのライフサイクルで活動が活性化され，その後も安定した活動が続いている．山が 2 つのケースも観測された．

- **サンプル #698**：この事例の山脈型は 7 年間に渡り上昇が継続する山脈型の活動であり，上昇方向に振れることはあるが，大きく下がることは少ない．

- **サンプル #651**： この事例の複合型は初期に小さなパルスがいくつかあり，継続し，少し衰退している．

参加者側の型も多様であり，OSS の共通型ではない．パルス型を除けば，長期間に渡って活動が継続している．参加者のデータは活動記録であるため，新規に参加する活動と継続する活動に分離した分析も可能である．

層別で観測された型は本節で示した 4 種類に分類されるが，型分類の閾値などの具体的な値の分析・検証は，次の課題である．

5. 利用者行動との関係

本章では，参加者の行動，すなわち OSS 開発に参加し労力を提供する者の行動と，そのユーザに当たる利用者の行動との関係について調べる．OSS においても「作品の評価」と「作品開発」が連動するケースを層別できれば，そのメカニズムの研究に貢献できると考え，探索的な分析を行った．

OSS は巨大な場であり,「作品の評価」と「作品開発」が連動する / しないのどちらも存在する．本研究は，測定し層別することにより，インターネット空間を介した「働き方」に関する知識を得ることである．本章では，4 章で示した活動の型に着目して，利用者行動と参加者行動の関係が観測されるのかについて概観する．

分析は，次のような組合せでサンプルを選定して実施した．3 章の表 1 で示した利用者と参加者のクロス表から観測対象を選び，両者の関係を時系列で可視化して観察した．利用者側の層別は，活動が少ない順に {A,B,C,D}，参加者側を {a,b,c,d} と表している．これらの組合せを Aa，Db といった形で表現し，Repositry に対応するサンプル番号で識別した．

5.1. 量的相関と時系列相関

利用者の活動量，すなわち Star を付けることで代用される変量と，参加者の活動量に相関はあるのか．表 1 では，Aa から Dd への対角上で値が膨れていることから，明らかに相関が認められる．

たとえば，利用者が多いから参加者が増える，あるいは，その逆，といった因果関係については，今回の分析で用いたような量的な組合せデータだけでは明らかにすることはできない．

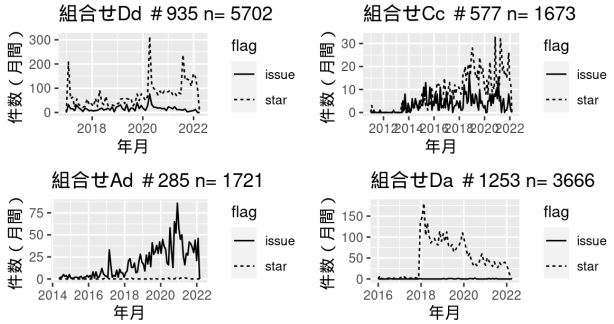


図 7: 時系列における相関の観察例

時系列的相関とは，利用者の増加に続いて参加者が増える，あるいはその逆などの時間遅れで反応があることを指す．反応がある可能性のあるサンプルの組合せは Dd や Cc など，参加者も利用者も多い場合が考えられる．一方，Ad や Da の組合せは，そもそも一方の活動が低調なのに，他方が活発であるデータの集団であり，因果関係は認められなかった．この 4 つの組合せで観測した時系列の例を図 7 に示し，その説明を次に述べる．

- **サンプル ＃935, ＃577**：組合せ Dd と Cc における例で，参加者活動と利用者活動は，時系列的にも連動し相関が認められる．
- **サンプル ＃285**：組合せ Ad は利用者の評価がほぼないが，参加者側は非常に活発な活動を示している．この事例では参加者の活動は上昇型のトレンドが 8 年に渡って継続している．
- **サンプル ＃1253**：組合せ Da は参加者活動が低いが利用者の評価が高い事例を示している．

利用者と参加者の行動に強い相関のある例，まったくない例などがあり，大きな層別では表 1 により，ある程度の絞り込みが可能である．相互の関係として，依存関係 (必要条件など) が強いものは見当たらなかった．

5.2. 利用者型の影響

4 章において，利用者と参加者の活動パターンを時系列の型で層別した．本節では，利用者側の型が参加者側の活動に影響を及ぼしているのかについて概観する．

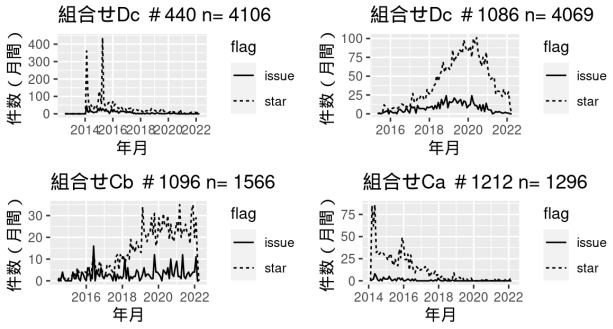


図 8: 利用者特性の型の影響例

図 8 に示した例を用いて，利用者行動の特徴的な型がある場合，参加者側に影響が生じているのか，あるい

は，参加者側の何かが利用者側に影響を及ぼしているのかについて概観する．

- **サンプル #440**：利用者側がパルス状に急増する例で，この事例では 2 回ほど生起している．一方で，参加者側に目立った変化は見られない．
- **サンプル #1086**：利用者側は山型であり，参加者側にも同様な山型が見られる．大きさの大小については多様である．
- **サンプル #1096**：利用者側が増加する山脈型を示している．参加者側も山脈型であるが利用者の増加とは強く連動していないように見える．
- **サンプル #1212**：利用者側が減少傾向を示し，参加者側も同様の変化を示している．

観測した範囲内で，利用者側の変化，特に急激な変化が参加者側に作用していると思えるような連動は見当たらなかった．このことから，商品が売れ始めると量産し，利益を追求するようなビジネスモデルとは異なることが分かる．

5.3. 参加者型の影響

ここでは，参加者側の行動が利用者にどのような影響を及ぼしているのかを概観するため，特徴的な参加者の型に対する利用者側の動きを調べた．参加者側の多様な型が多く見られたのは Cd の組合せであったため，次の 4 例もその中から選んでいる．

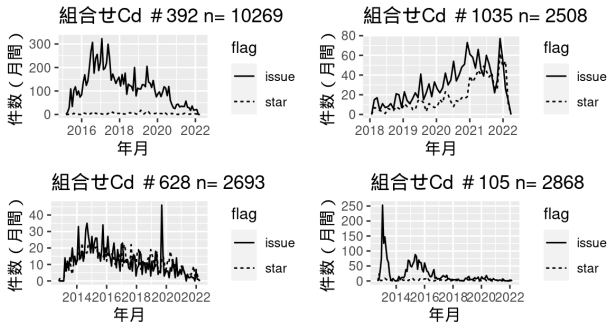


図 9: 参加者特性の型の影響例

図 9 は，図 8 とは逆に，参加者側に顕著な型が見られる場合について例示した．

- **サンプル #392, #105**：連動が見られない，あるいは大きな参加者の活動に比べ，利用者側の相対的な大きさが小さいケースとして層別される．
- **サンプル #1035, #628**：連動して変化しており，何らかの結びつきの研究サンプルとして層別できる．

参加者側の型と利用者側の特性についても，連動がないもの / あるものが存在し，定性的には層別として型が使えることが推測できる．

5.4. 時系列分析の要件のまとめ

OSS 作品は，その種類や数の巨大さだけでなく，動的な変化の多様性についても，商用作品とは全く異なるものであった．4.2 節で述べた通り，探索的分析の目的は，対象のトレンドや周期変動を検出するのに必要な周期を明らかにすることである．データが少ない Repository が数多く存在するため，データが少ない場合には四半期程度の間隔での対応が必要となる．一方，パルス型という短期変動が認められた．パルス型を観察するためには，短い間隔での対応が求められる．

本研究で示した時系列変化の観察により，時系列分析の要件として明らかになったことは，次の 2 点である．

- **短期変動**：パルス状の変化が，特に利用者側で観測された．分析の詳細化には時間幅を 1 ヶ月以下，できれば週単位で観測する必要がある．
- **短期変動以外の変動**：緩やかな変動であり四半期の間隔で対応できると思われる．

6. おわりに

本研究では，OSS の作品に関わる参加者と利用者の行動分析を試みた．膨大な数の OSS 作品から，その参加者や利用者の「行動」に関する情報を得るために，対象作品の層別と選択を行う課題があった．ここでは，前処理で選んだデータ群に対し，参加者と利用者の活動量の順位をカテゴリ化し，クロス表を作成して対処した．

中心となる課題は，行動を特徴づける説明変数として何を用い，どのように分析するかである．ここでは，時間経過による変化に着目し，分析の時間周期を時系列データに倣って，共通化する方法を用いた．この方法に

より，異なる種類の作品であっても，参加者や利用者の行動を比較したり，層別することが可能になった．

今回の分析では，定量的な行動のモデル化には至っていないが，視覚化による定性的な分析に耐える方法を開発することができた．この方法により，パルス型の急峻な行動の型が存在すること，利用者行動と参加者行動の関係は多様であり，さまざまな関係が存在することが明らかになった．

次の課題として，参加者の役割に着目し，その時系列変化の分析がある．これは，エンジニアの成長過程を行動記録から分析する課題である．我々の研究目的であるエンジニアの育成や組織化における「意欲」を OSS 活動の「行動」から学び活用できるよう，次の課題に取り組んでいく所存である．

脚注

1) インターネットのアプリケーションを通じて共通の関心分野，価値観や目的を持った利用者が集まって持続的に相互作用する場であり，提供されるネットワークサービスの総称 [1].
2) 一元的にデータの構成管理を行う単位であり，ファイルやディレクトリを格納すると共に変更履歴が記録される．
3) 周期的な時間間隔で観測したデータ.
4) 時系列の定常性が認められる成分.
5) 季節変動や景気変動などに相当.
6) ノイズはホワイトノイズを意味し，平均 0 の変化成分.
7) GitHub では，Repository に Star を付けるということは，リポジトリメンテナに対してその作業についての感謝を示すことでもあり，GitHub のリポジトリランキングの多くは，リポジトリに付けられた Star の数を考慮している [13].
8) Issue とは作業に関するアイデア，フィードバック，タスク，バグを追跡するために用いられる機能である [14]

参考文献

[1] インターネットコミュニティ，https://ja.wikipedia.org/wiki/インターネットコミュニティ (accessed:2022/03/18)

[2] GitHub, https://github.com/ (accessed:2022/03/18)

[3] Wickham, Hadley and Grolemund, Garrett, “R for data science: import, tidy, transform, visualize, and model data,” O’Reilly Media, Inc. 2016

[4] 増田礼子，松尾谷徹,「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」，ソフトウェア・シンポジウム 2020 論文集，pp.44-54，ソフトウェア技術者協会，2020

[5] Zipf, George Kingsley, “The Psycho-Biology of Language,” Boston-Cambridge Mass. Houghton Mifflin, 1935

[6] Zipf, George Kingsley, “Human Behavior & The Principle of Least Effort, An Introduction to Human Ecology,” Addison-Wesley Press Inc., 1949

[7] Gini, Corrado, “Measurement of inequality of incomes,” The economic journal, Vol.31, No.121, pp.124–126, JSTOR, 1921

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

[9] 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:2022/03/18)

[10] Rob J Hyndman and George Athanasopoulos, “Forecasting: Principles and Practice (3rd ed),” https://otexts.com/fpp3/, Monash University, Australia, 2022 (accessed:2022/03/18)

[11] 北川源四郎,「4-4 時系列データ解析」，http://www.mi.u-tokyo.ac.jp/consortium2/pdf/4-4_literacy_level_note.pdf，東京大学 数理・情報教育研究センター，2020 (accessed:2022/03/18)

[12] GitHub Docs，https://docs.github.com/ja，(accessed:2022/03/18)

[13] GitHub Docs：Star について，https://docs.github.com/ja/get-started/exploring-projects-on-github/saving-repositories-with-stars#about-stars (accessed:2022/03/18)

[14] GitHub Docs,：Issue について，
https://docs.github.com/ja/issues/
tracking-your-work-with-issues/about-issues
(accessed:2022/03/18)

[15] GitHub API v3, https://developer.github.com/
v3/ (accessed:2022/03/18)

[16] 金明哲, 「[連載] フリーソフトによるデータ解析・マイニング第 60 回：統計的テキスト解析 (5)〜統計法則と指標〜」, https://www1.doshisha.ac.jp/
~mjin/R/60/60.html
(accessed:2022/03/18)

[17] geom_freqpoly,
Histograms and frequency polygons,
https://ggplot2.tidyverse.org/reference/
geom_histogram.html (accessed:2022/03/18)

[18] 東京都 新型コロナウイルス感染症対策サイト,
https://github.com/tokyo-metropolitan-gov/
covid19 (accessed:2022/03/18)

大学教育における電子商取引システム開発演習事例およびチーム内の役割分析

増田 聡
東京都市大学
smasuda@tcu.ac.jp

松尾谷 徹
デバッグ工学研究所
matsuodani@gmail.com

要旨

大学教育においても電子商取引 *(EC)* の講義および演習が行われており，教育効果を高めるためどのような演習とするかが課題となっている．本論文では，*EC* システムとして *Web* コンテンツ管理システムの *WordPress* および電子商取引機能のプラグインソフトウェアである *WelCart* を基に，*EC* サイト開発演習を行った経験から得た知見を共有し考察する．受講生へのアンケート回答およびコメントから好意的に受け入れられていることが見られたが，回答項目の因子分析や主成分分析では特筆すべき傾向は見当たらなかった．また，演習サーバーへのアクセス数と開発した *EC* サイトの評価にやや正の相関を見ることができた．さらに，チーム内の役割について分析したところ，役割を決める前後で *EC* サイト開発者のアクセス数が相対的に増えるなど，役割による変化が見られた．

1. はじめに

インターネットを利用した電子商取引 (Electric commerce 以下 EC) が盛んである．社会の要請により大学教育においても EC の講義が行われている．EC に関するビジネスの仕組みやデータ管理の方法，関連する法規など座学も必要であるが，さらに知識を定着させるために実際に EC システムを利用しインターネットショッピングサイト (以下 EC サイト) を開設運営する演習が必要である．しかしながら，大学講義という時間の制約と受講生が保有する情報システム開発能力で利用できる適切な EC システムは見当たらない．産業界で使われている EC システムパッケージソフトウェアは，コンテンツ作成やショッピングサイト開発にプログラミングスキルを必要とする．電子商取引を受講する全ての受講生がプログラミングスキルを有しているわけではない．また，講義時間も限られているため，より簡単に開発でき，かつ，教育効果を高めるため実際の EC サイトに近いものが要求されている．また，教育効果を高めるためどのような演習課題とするか工夫が求められている．

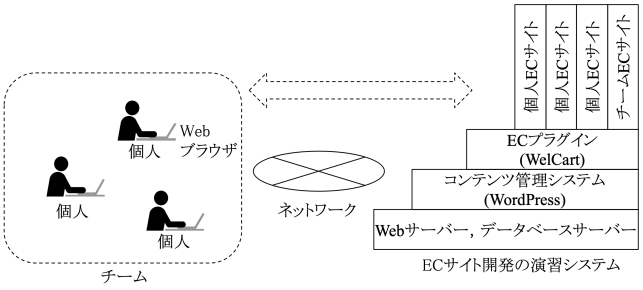


図 1: EC サイト開発演習のシステム概要

本論文では，2017 年から 2021 年まで主に大学の文理融合学部の 2 年生および 3 年生を対象に，EC システムとしてコンテンツ管理システムの WordPress およびその電子商取引機能のプラグインソフトウェアである WelCart を基に，EC サイト開発演習を行った経験から得た知見を共有し考察する．これは，電子商取引の授業を行う方にとって有用な知見であると考える．図.1 は EC サイト開発の演習システムの概要を表している．共有する知見は，事前準備における演習システムの工夫や受講生のアンケートやサーバーアクセスログの分析などから得られたものである．

2. 関連研究

2.1. EC の大学教育例

表 1: 大学のシラバス情報のインターネット検索結果

検索語 [*1]	ヒット数
”ソフトウェア工学”	10,800
”経営システム”	10,300
”電子商取引” OR ”e コマース”	5,490
”プログラミング演習”	5,330
”情報システム概論”	2,180

[*1] 検索語に「”シラバス” site:ac.jp」を追加して検索

大学教育で EC の取り組み状況を簡易的に調べるため，表. 1 のように検索語でインターネット検索サイトを利用して，検索語にヒットする Web サイトの数の調査を行った．例えば，EC は「”電子商取引” OR ”e コマース” ”シラバス” site:ac.jp」という検査語で検索を行い，そのヒット数を調べた．ダブルクオーテーションで囲んだ語は完全一致で検索され，site:ac.jp は ac.jp というドメインを持つサイトのみを対象に検索が行われる．EC 関連の教育は「プログラミング演習」と同程度にシラバスに登場し，大学教育で取り組まれていることがわかる．

ヒットした Web サイトにおいて EC 関連の講義内容は，ビジネスモデル (Business to Consumer，Business to Business)，マーケティング，ネット広告，情報技術などが見られる．システムを利用した EC サイト開発の講義は多くは見られなかったが，マルチチャネルコマースプラットホームの Shopify と連携し e コマース人材の育成強化のためのプログラムを提供している大学もあった [8].

2.2. EC システム例

大企業が本格的に EC システムを構築する際は，自社で構築するかまたは ERP(クラウド含む) などを利用することが一般的である．ERP の EC システムの代表例としては，Salesforce 社 [1] や SAP 社 [5] の EC 関連のサービスがある．一方，スタートアップまたは中小規模の企業が EC を始める際は，EC プラットホームと呼ばれるクラウドに特化したシステムを利用することが増えてきている．EC プラットホームには，Shopify や BASE, STORES などがある [2].

しかしながら，これらの EC システムを用いて EC サイト開発を行うには専門知識を必要とし，時間制限のある大学教育においては適用困難と考えられる．大学教育においては受講生のレベルに合わせて演習を行い成長を促すことが期待される，また，各大学のシステム環境や独自の演習課題などを考慮すると，EC サイトの開発がプログラミング作業を伴なわないような容易で，かつシステム環境の変更に対応できるものが求められる．そこで，著者らはオープンソースである EC プラットホームに着目し，WordPress と WelCart の組み合わせを EC システムとして使用することとした．ホームページ作成で WordPress を使用している受講生もいるため，受講生のレベルにも合っている．現在は，同様のオープンソースである EC プラットホームに「EC CUBE」があるが，当システムを検討した 2018 年当時には未だ使用例も少なく対象とならなかった．

2.3. チームまたは役割に関する分析事例

チーム学習を効果的に進めるための教育方法に関する研究事例に関して，西上ら [9] は，LEGO ロボットとゲーム課題を題材とするプログラミング演習をチームで行った事例をまとめている．この事例では，アンケート評価をもとに，考察において「成績の上位陣と下位陣に分けて分析し，上位陣は，自主的な取り組みが目立ち満足度も高く，下位陣は質問はしていたが，受身の姿勢が見られ反省意見が多かった」とし，受講生の態度に対して指導側の適切な支援が重要であるとしている．また，三浦ら [6] は，チームの状態を初期フェーズ，活性化フェーズ，収束フェーズに分け，チーム学習の演習支援環境を提案している．

3. 演習内容

3.1. 講義および演習内容

講義は 1 回 90 分の授業を 14 回行った [7]．登録受講生の人数は 157 名であり，情報工学を専門としない受講生も半数近く含まれる．第 1 回から 7 回までは座学として，電子商取引概要，マーケティング理論 (消費者行動論，4P(Product， Price，Place，Promotion))，電子商取引サイトの経営戦略と法律，電子商取引システムを支える情報技術の概要，ユーザーエクスペリエンスなどを行った．

(a) ホームページ　(b) カートの中身　(c) ショップ設定　(d) 商品登録

図 2: EC サイトの Web 画面例 (WelCart デモサイト [3] および WelCart マニュアル [4] より)

演習は，まず個人演習として，個人で EC サイト設定の演習を約 40 分を 2 回に分けて行った．個人演習で受講生に演習環境に慣れてもらい，EC サイト作成のスキルをつけることが目的である．次に 1 チーム約 7 名でチーム分けを行い，チーム演習を行った．チーム演習の課題および進め方は表.2 のとおり．

表 2: チーム演習の課題

課題	模擬 EC サイト開発
進め方	1) ショップコンセプト，ショップ名の決定 2) 業界分析（同業他社，自社の強み，法律） 3) 顧客分析 (ペルソナの作成) 4) 販売計画 (EC サイト要件) の作成 5) 模擬 EC サイト開発 6) 発表資料の作成
作成物	発表資料 (パワーポイント) 模擬 EC サイト (WordPress/WelCart)
発表会	発表と EC サイトのデモを行う． 発表時間は 7 分． 受講生による相互評価を行う．

チーム演習では，まず教員から各個人の役割を，司会進行，書記，意思決定，EC サイト開発，その他とし，チーム内で決めてもらった．各役割の内容は，それぞれ以下のとおり．

- 司会進行役：日程に沿って検討や作業の進行を行う．
- 書記：作業シートに皆の発言，検討事項等を記入する．
- 意思決定者：意見がまとまらず期限が迫っている際に決定を行う．
- EC サイト開発者：模擬 EC サイトを開発する．

この時点での役割はチーム作業を円滑にする目的で設定したものであり，この時点では後述するサーバーアクセス数との関連を意図したものではなかった．

3.2. 演習用 EC システム構成

EC システム構成は Linux サーバー上の WordPress および WelCart プラグインを使用した．選定理由は，WordPress と WelCart の組み合わせは，画像のアップロードや文字入力でユーザーインターフェース作成が可能で，配送設定や商品登録で EC サイトの構築が可能であること，また，日本語のヘルプやガイドラインが豊富であるため受講生が参照可能であることである．図 2 は，WelCart で作成された EC サイトの例と EC サイトの設定の例を表している．WelCart で EC サイト設定方法は，ショップ設定，配送設定，商品登録で商品注文が可能となる．プログラミングは必要ない．WordPress や WelCart の具体的な使用方法は，EC サイトを動かすまでの関連する Web サイトのリンク情報を提供し，その Web サイトを用いて説明を行った．

EC システムを使用するユーザーおよびシステム環境は，2021 年度の講義では受講生個人用に 160 ユーザー分，チーム用に 23 ユーザー分それぞれ作成した．個人用のユーザーおよびシステム環境は受講生の個人演習用およびチーム演習時の検討確認用として使用し，チームシステム環境とは分かれている．具体的には，WordPress 上で複数サイトを管理するネットワークサイト管理構成とした．WordPress のネットワークサイト管理の構成は個人やチーム各自で WordPress のプラグイン導入等が

できない制限はあるが，演習上の統一した管理は容易であるメリットがある．また，WelCart には決済機能があるが，誤ってクレジットカード決済等をしないよう代金引換のみとするよう本演習独自にソースコードを変更した．

4. 結果

4.1. アンケート結果

2021 年度の講義ではアンケート調査を行った．個人演習の開始前に，ソフトウェアへの興味やプログラミング経験など事前アンケートを行い，講義最終日に教材 (EC システム) の役立ちやお勧め，自由コメントなどの事後アンケートを行った．図. 3 はアンケートの主要な結果を表している．

アンケート有効回答数 122 であった．この中で，約 85%はソフトウェアに興味があり，プログラミング演習等で経験者が 90%に上っている．プログラミング経験は大学講義における経験も含めて良いとしたので高い割合となった．複数人での共同作業によるソフトウェア開発は，2 年次のソフトウェア開発演習の講義を受講した受講生が経験ありと回答したと推測する．約 60%の受講生は共同作業によるソフトウェア開発の経験は無いと回答した．約 85%は教材の EC システムが役に立ったと回答したが，他人に勧める受講生は 70%になっている．他人へ勧めることを「どちらともいえない」と回答した受講生の自由コメントで「お手本のようなものがない中でサイトを作ったので，難しさがあった」,「外部ツールや CSS を用いたユーザーインターフェースの作成方法も教えてほしかった」などコメントがあった．

図 4 は，アンケート回答を設問ごとの対応をバブルチャートでグラフ化したものである．各グラフの横軸縦軸は選択肢の値の 1 から 5 を表し，グラフ上の円の大きさが回答件数を表している．また，アンケート項目の PRE_Q1 や POST_Q1 などの設問と選択肢は，付録の表 3 にそれぞれ内容を記述した．傾向としては，回答間の組み合わせで特定の回答に偏りが見られる部分がある．これは設問間の関係で，例えば「プログラミング経験」と「チームでのソフトウェア開発経験」との関係などにより，プログラミング経験のない受講生はチームでのソフトウェア開発経験もないなどの関係と考えられる．POST_Q1 はチーム内の役割の設問であり，他の設問と

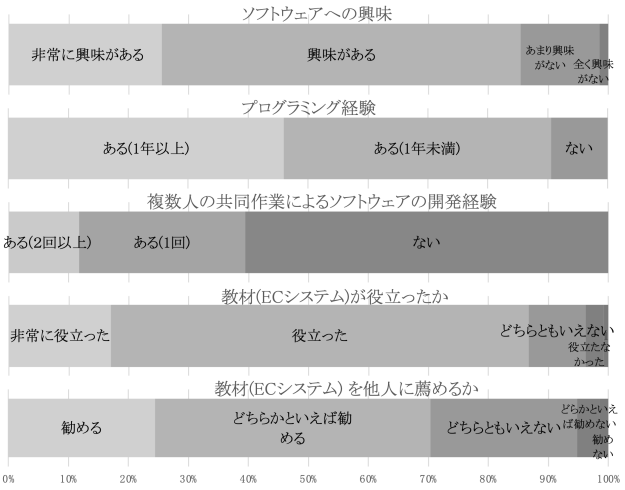


図 3: アンケート回答 (一部抜粋)(N=122)

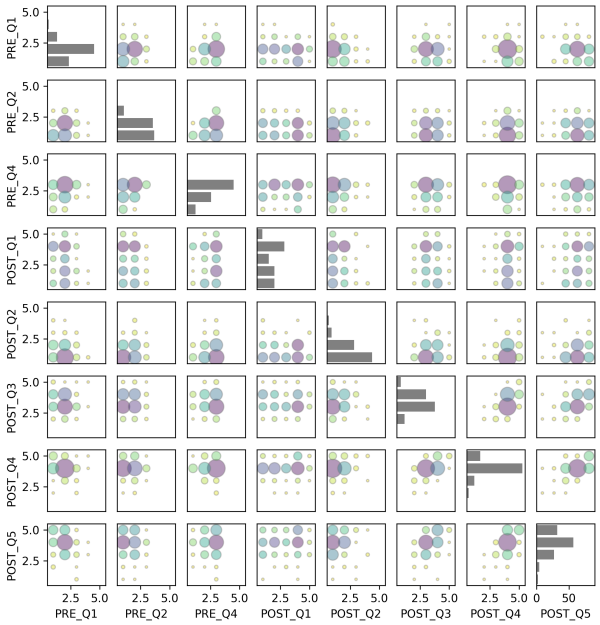


図 4: アンケート回答のバブルチャート行列

比べより均等に分かれているのが分かる．この他，アンケート回答で因子分析，主成分分析を探索的に行ったが，特筆すべき傾向は見当たらなかった．

4.2. 演習 EC システムのアクセスログ分析

演習 EC システムの Web サーバーへのアクセスログである http ログファイルを分析した．http ログファイルに

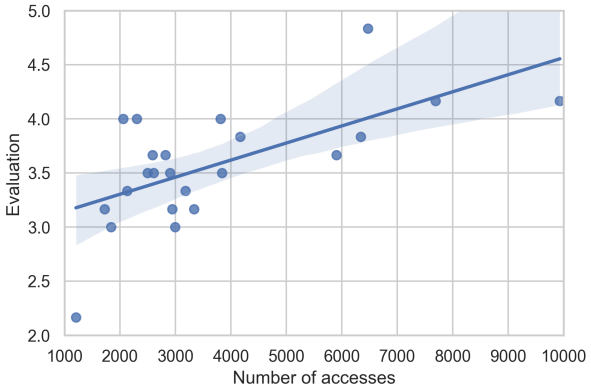


図 5: チームのアクセス数と EC サイトの評価

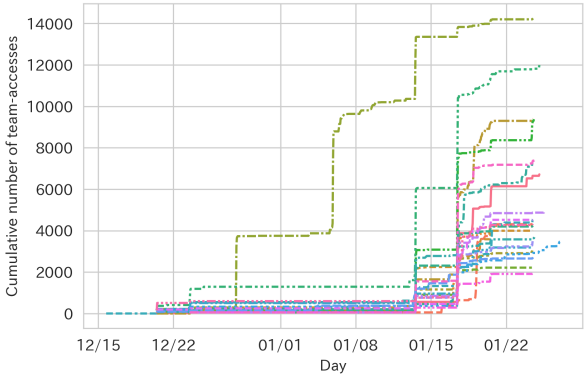


図 6: チームアクセス数の時系列累積和

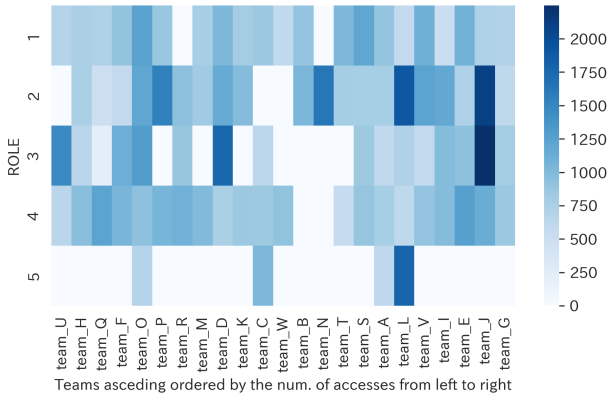


図 7: チーム別役割別の個人アクセス数ヒートマップ

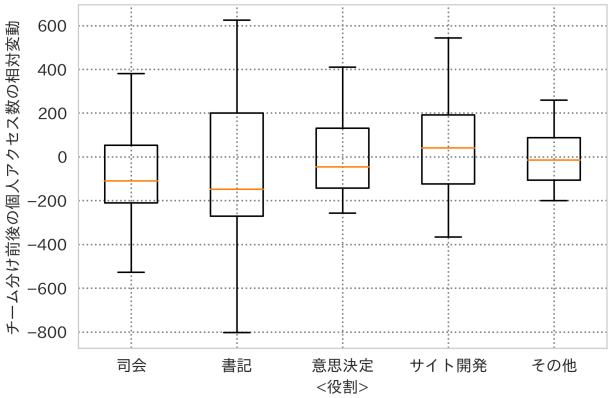


図 8: 役割による個人アクセス数の変動

は，ユーザーが Web ブラウザで WordPress や WelCart のユーザーインタフェースを通して操作した結果が書き出されている．また，ログに書き出されているユーザー ID の情報から個人およびチームのサーバーへのアクセス状況が分かる．チーム ID でログインした個人を特定することはできないが，EC サイト開発の役割の個人がログインしたとみなした．ログファイルの対象期間は，個人演習が開始から終了までの 45 日間とし，チーム演習は開始から同じ終了日までの 38 日間とした．個人演習を最初に開始し，その後にチーム演習を開始した．チーム演習期間中も個人演習環境は使用可能であり，講義最終課題締切日をいずれも終了日とした．これらの対象ログのレコード件数は約 28 万件であった．また，本開発演習の評価はチームに演習の最終発表に対して行っている．評価は，ショップコンセプト，業界分析 (同業他社)，業界分析 (自社の強み，法律)，顧客分析 (ペルソナ)，販売計画，EC サイトの各項目を 5 段階 (5: 良い，1: 悪い) で評価し，平均の値を総合評価とした．

図 5 は，チーム ID によるサーバーへのアクセス数と評価を表していて，ゆるやかな正の相関関係が見られる．さらに，チームのアクセス数と評価の相関係数は 0.6366 となり，アクセス数が多いほど評価が高い傾向があることが分かった．アクセス数の多いチームは，例えば，商品登録数が多く商品カテゴリ分けするなど本格的な EC サイトとなっていて評価も高かった．図 6 は，チームアクセス数の 1 時間ごとの時系列の累積和を表している．アクセス数の多いチームは早い段階からアクセスしていることが分かる．また，講義日に集中してアクセスが増

える様子も見られる．

さらに，チームのアクセス数に関して，Q.1) アクセス数の多い個人が入ったチームがアクセス数が多いのか，Q.2) チーム分け後に活性化し，チームのアクセス数が多いのか，の２点について分析を行った．Q.1) に関しては，チーム分前の個人演習期間中の個人のアクセス数上位の個人が入ったチームと，アクセス数上位のチームの相関を調べると，相関係数は-0.107 となり，相関は見られなかった．つまり，アクセス数の多い個人が入ったチームがアクセス数が多くなった訳ではなかった．Q2) に関しては，チーム活性化の要因を探るため，演習前および演習後のアンケートまたはアクセスログから探索的に分析を行った．

チーム内役割でアクセス数の様子を可視化するため，図７に，チーム別役割別のアクセス数をヒートマップで表した．横軸のチームの順番はアクセス数の昇順で，左から右へよりアクセス数が多くなる．チームアクセス数の最も多い team_G の各個人のアクセスが多いわけではない，状況としては個人環境で確認や検証等をせず，専らチーム ID で EC サイト構築を行なっていたと考える．一方，team_J は役割 2(書紀)，3(意思決定者) の個人アクセス数が多く個人環境で確認や検証等を行なっていた様子が窺える．

さらに，役割決定が個人のアクセス数に与えた状況を調べるため，役割決定が行われたチーム分け前後の個人アクセス数の相対変動を役割別に図８に表した．アクセス数の変動は，アクセス数のチーム分け後のアクセス数からチーム分け前のアクセス数を引いた値で，相対変動はその変動の値の平均との差をとっている．司会進行，書記および意思決定の役割になった受講生は相対的にアクセス数が下がっていて，EC サイト開発者はアクセス数を上げている．

5. 考察

5.1. アンケート項目分析

図３のアンケート結果の上位３項目のソフトウェア開発の興味や経験に関する項目は，受講生が学部２年生および３年生であったことから，予想された結果ではあった．また，今回の EC システムの演習環境が受講生にとって目新しいものであったため，下位２項目の EC システムに好意的であったと考える．今後の改善としては，他人へ勧めることを「どちらともいえない」と回答した受講生のコメントの「ユーザーインターフェースの作成方法も教えてほしい」などへの対応がある．

図５のチームのアクセス数と EC サイトの評価の結果は，より時間をかけアクセス数が多いチームのサイトの方がより良いサイトとなり評価も高いという経験則にも沿った結果であった．この結果は新たな知見とは言えないが，講義主催側にアクセス数を増やすための工夫が必要という気づきを与えるものであった．アクセス数を増やすための方法は，a) 演習時間を増やす，b) 講義時間外にも課題を与える，c) 演習環境をよりアクセスしやすい環境にする，などが挙げられる．c) の演習環境については，今回のアンケートコメントより，魅力的な EC サイトにするために，前述のユーザーインターフェースの作成方法の提供も対応案として検討したい．

5.2. チーム間の差，役割による差

図７および８から，役割がアクセス数に影響を与えている傾向が見られる．役割設定として「EC サイト開発」を独立させたことが影響したと考えられる．つまり，司会進行，書記は EC サイト開発ではないから，アクセスしなくとも良いと受講生が考えた可能性がある．EC サイト開発の役割になった受講生が“やる気を出した”チームの評価が高くなったのか，この点は今回のデータからは分析できなかった．今回は個人の動機の変化やチームの活動状況を客観的に捉えることのできるアンケートの設問となっていなかった．この点は次回の課題とする．

6. おわりに

本論文では，大学教育における EC サイト開発のチーム演習の事例を共有し，アンケートおよびシステムのアクセスログのデータからチーム内役割の分析を行った．分析の結果，複数人のチーム演習では，サーバーへのアクセス数が多ければ，開発された EC サイトの評価も高くなる傾向が見られた．さらに詳細に分析を進めると，アクセス数の多いチームは，チーム演習が始まってからアクセス数が伸びたことがわかった．システム開発が得意な個人や意欲のある個人がたまたま集まったチームのアクセス数が多くなった訳ではなかった．平均的なアクセス数の個人が集まったチームのチームアクセス数が多くなった要因は，チーム内での「役割」にあると考えた．

個人アクセス数の分析の結果，司会進行，書記の役割になった受講生はアクセス数の順位を下げている，意思決定，EC サイト開発，その他の役割はアクセス数の順位を上げている，ことがわかった．

今後は今回の結果から，役割のローテーションなども行いより効果的な講義および演習環境を整えたいと考える．

参考文献

[1] Salesforce Japan Ltd. Saleforce commerce cloud, https://www.salesforce.com/jp/products/commerce-cloud/overview/. 2022 年 3 月 8 日アクセス.

[2] クラウドＥＣ．ＥＣ基礎知識と市場規模, https://www.cloudec.jp/ecnews/ec2021/. 2022 年 3 月 8 日アクセス.

[3] コルネ株式会社. Welcart basic デモサイト, https://themes.welcart.info/basic/. 2022 年 3 月 4 日アクセス.

[4] コルネ株式会社. Welcart マニュアル, https://www.welcart.com/documents/manual-welcart. 2022 年 3 月 4 日アクセス.

[5] SAP ジャパン株式会社. Sap commerce cloud, https://www.sap.com/japan/products/commerce-cloud.html. 2022 年 3 月 8 日アクセス.

[6] 三浦真人, 小林祐介, 島田和幸, 高橋晃一, 清木進, 櫨山淳雄. 個人とコミュニティの支援を有するソフトウェア開発グループ演習環境の提案. 情報処理学会研究報告. GN, [グループウェアとネットワークサービス], Vol. 64, pp. 19–24, jun 2007.

[7] 東京都市大学. 電子商取引論 シラバス, https://websrv.tcu.ac.jp/tcu_web_v3/slbssbdr.do?value(risyunen)=2022&value(semekikn)=1&value(kougicd)=ybb701201&value(crclumcd)=y217001. 2022/5/16 アクセス.

[8] 法政大学. 法政大学と shopify が連携. 2021 年春講義の実習として世界で通用する e コマース人材の育成プログラムを提供開始, https://www.hosei.ac.jp/careerdesign/info/article-20210507091125/. 2022 年 3 月 8 日アクセス.

[9] 西上明普, 加藤聡, 富永浩之. Lego ロボットとゲーム課題を題材とする導入体験としてのプログラミング演習：対話的な事前学習のためのオンライン教材の作成. 電子情報通信学会技術研究報告. ET, 教育工学, Vol. 110, No. 453, pp. 137–142, feb 2011.

A. 付録

表 3: アンケート項目

項目コード	アンケート項目および選択肢
PRE_Q1	ソフトウェア内部の作りや動きに関する興味はどれに当てはまりますか? 1: 非常に興味がある, 2: 興味がある, 3: あまり興味がない, 4: 全く興味がない
PRE_Q2	プログラミングの経験はありますか? 1. ある (1 年以上), 2. ある (1 年未満), 3. ない
PRE_Q4	複数人の共同作業によるソフトウェアの開発経験 (大学での講義実習を含む) はありますか? 1. ある (２回以上), 2. ある (1 回), 3. ない
POST_Q1	チーム演習であなたの役割はどれにはてはまりますか？1. 司会進行, 2. 書記, 3. 意思決定者,4.EC サイト開発者,5. その他
POST_Q2	チーム演習のあなたの役割の作業について，授業時間以外で合計でどれくらい時間がかかりましたか？1:3 時間未満, 2:3 時間以上 5 時間未満, 3:5 時間以上 10 時間未満, 4:10 時間以上
POST_Q3	個人演習，チーム演習も含めて，EC サイトを開発した際，WordPress や WelCart の操作はどれくらい分かりやすいですか? 1. 非常に分かりにくい, 2. 分かりにくい, 3. 普通, 4. 分かりやすい, 5. 非常に分かりやすい
POST_Q4	今回の EC サイト開発の演習は電子商取引の理解に役に立ちましたか？1. 勧めない, 2. どらかといえば勧めない, 3. どちらともいえない, 4. どちらかといえば勧める, 5. 勧める
POST_Q5	今回の EC サイト開発の演習を他の人に勧めますか？1. 勧めない, 2. どらかといえば勧めない, 3. どちらともいえない, 4. どちらかといえば勧める, 5. 勧める

カスタマーサクセスによる業務プロセス改善の分析とシステム拡張応用
～サブスクリプションモデルにおけるシステム開発のための一つのアプローチ～

八木 将計
株式会社日立製作所
masakazu.yagi.zd@hitachi.com

大澤 郁恵
株式会社日立製作所
ikue.osawa.om@hitachi.com

堀 光孝
株式会社日立製作所
mitsutaka.hori.bq@hitachi.com

丸田 紘心
レクシエス株式会社
genshin.maruta@customer-success.college

要旨

本論文では，サブスクリプションモデルで重要になるカスタマーサクセスの実現において，業務モデリング手法 PReP (Products Relationship Process) Model を活用する方法を提案する．サブスクリプションモデルは，継続利用を前提とし，絶えずシステムやサービスを拡張・アップデートしていくことが重要となる．カスタマーサクセスは，システムやサービスをユーザーに使いこなしてもらうことで，継続的な利用を推進する活動である．そのカスタマーサクセスでは，顧客の業務理解が非常に大切になるが，専任者となるカスタマーサクセスマネージャのスキルや経験に依存するという問題がある．本論文では，PReP Model をサービス分析に応用することで，顧客業務の目的とサービスの関係性を構造的に理解でき，効果的にシステムの機能拡張に繋げることができることを示す．

1. はじめに

近年，サブスクリプションモデルを採用する SaaS (Software as a Service) 業界を中心に「カスタマーサクセス(CS: Customer Success)」が注目されてきている．CS とは，2000 年代初頭にセールスフォース・ドットコムが提唱しはじめた概念であり，受動的に顧客の要望を満たすカスタマーサポートとは異なり，顧客の事業成功と自社の収益を達成するための能動的な活動のことである[1][2]．CS は，セールスフォース・ドットコムをはじめ，国内企業にも浸透しはじめている[3]．

CS が求められるサブスクリプションモデルでは，従来の売り切りモデルとは異なり，顧客の購買ハードルが低い反面，比較的容易に解約(チャーン)が発生しやすい．サブスクリプションのビジネスモデルは継続性が重要であるため，新規顧客の獲得による売上よりも既存顧客のチャーン減少や，より高額のプランへのアップグレード(アップセル)／関連する別システムや別サ

にできないため，購入した製品を使おうというモチベーションが働くため，よほどのことがない限り他社の製品に乗り換えるということは発生しにくい．しかし，製品分野がコモディティ化して競合優位性が低下する，技術革新により製品価値の厳選がハードウェアからソフトウェアに推移する，といった時代の流れにより，モノ売りから SaaS に代表されるコト売り=サービスビジネスへの転換が求められるようになってきた(図 1)．

ービス導入(クロスセル)が CS の重要な指標となる．このように，一般的な営業やカスタマーサポートの部隊と異なる指標を持ち，長期的に顧客との関係性を築く必要があるため，CS の専門担当者として，「カスタマーサクセスマネージャ(CSM: Customer Success Manager)」が置かれる．CSM には，自社のシステムやサービスを用いて顧客が事業成功するため，顧客の状況に合わせた適切なアプローチを取ることが求められる．

CSM は，システムやサービスの導入状況に応じて，能動的に顧客へアプローチしなければならないため，顧客の業務や課題を十分に理解する必要がある．しかし，現状の CS 業界では，顧客理解の有効な手法が確立されておらず，CSM のスキルや知識に依存している状況にある．本来であれば，顧客理解において業務プロセスモデリングは重要な方法になるが，BPMN (Business Process Model and Notation)[4]に代表される作業や処理などの「タスク」に注目しているモデリング手法は，複雑になり全体像が見えにくく，自社のシステムやサービスの位置付けがわかりづらくなるという問題があり，あまり使われていない状況にある．

そこで，本論文では，「成果物や情報」のみの関係性から業務を記述する PReP (Products Relationship Process) Model [5][6][7]を用いた，カスタマーサクセス実現のための業務モデリング手法を提案する．また，実際に CS を採用しているサービスに本手法を適用した結果，システムの新機能開発を大幅に短縮するとともに，テストマーケティングにおいて目標を大きく上回る成果を得ることができた．

2. カスタマーサクセスとその課題

2.1. カスタマーサクセスの必要性

従来のモノ売りのビジネスモデルでは，製品を販売して売上が立つまでが重要となる．また，顧客も投資をムダ

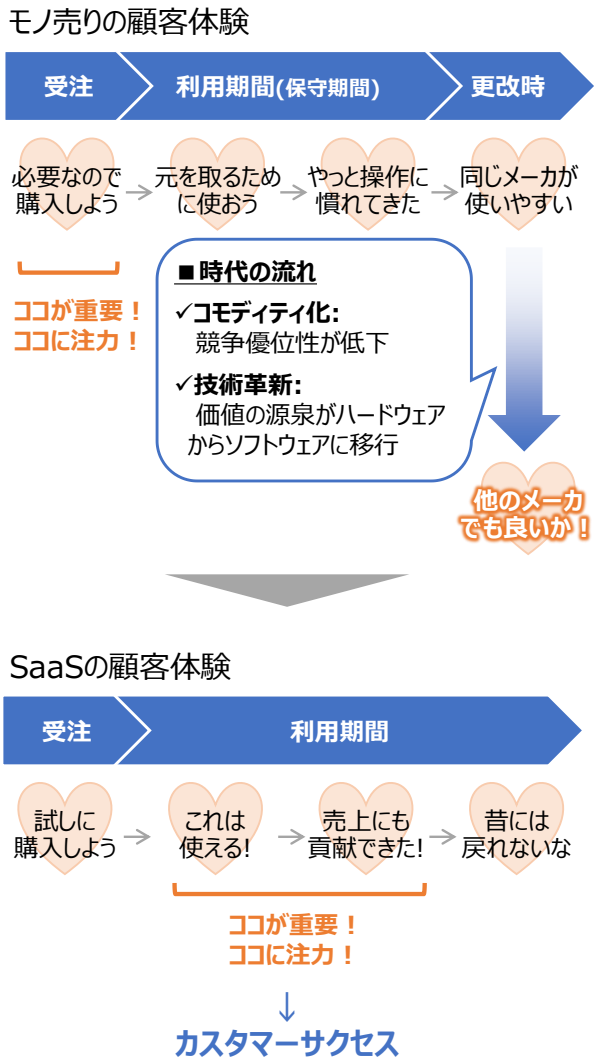


図 1 カスタマーサクセスが求められるようになった背景

SaaS，特にサブスクリプションのビジネスモデルでは，モノ売りとは異なり，利益が出る前に顧客が離れること(チャーンすること)を許すため，購入時の顧客満足度を高めるだけではなく，自社のシステムやサービスを顧客に長く使いこなしてもらう必要がある．

「カスタマーサクセス(CS: Customer Success)」は，2000 年初頭にセールスフォース・ドットコムが自社のシステムやサービスを活用して「顧客の事業成功」に導くことで，顧客がチャーンしないようにする活動として提唱した言葉である．しかし，CS は単なるチャーン防止施策というわけではなく，顧客の事業成功に寄り添うことで，より高額プランへのアップグレード(アップセル)や，関連する別システムや別サービスの導入(クロスセル)の提案がスムーズに行えるため，事業の成長を促進できるといわれている[1][2].

2.2. カスタマーサクセスの活動

CS では，前項に示す通り，チャーン率やアップセル率／クロスセル率といった，指標が必要となるため，従来の営業やカスタマーサポートとは異なる専門担当者として「カスタマーサクセスマネージャ(CSM: Customer Success Manager)」という役職を置く．

CSM は，自社のシステムやサービスが顧客業務にどのように導入されているのかを把握し，適切なタイミングで適切な施策を取ることで，顧客が正しく自社のシステムやサービスを使いこなして，成功に導くことがミッションとなる．そのため，CSM は，自社のシステムやサービスを導入することによる顧客のカスタマージャーニーマップを予め作成しておき，顧客の状況を表わすヘルススコアと呼ばれる指標を測定しながら，顧客へのアプローチのタイミングと施策を決定する(図 2).

アプローチの施策としては，顧客への関わり方によって，手厚い支援を行う「ハイタッチ」，メールやチャットなどのテクノロジーを用いた「テックタッチ」，ハイタッチほど手厚くないもののテックタッチよりも直接顧客に関わる「ロータッチ」という 3 つのタッチモデルがあるが，いずれの場合であっても，顧客の状況に合せて適切な施策を取ることが重要である．

2.3. カスタマーサクセス実現に関する課題

前節に示す通り，CS を実現するためには，適切なタイミングで適切な施策を取る必要がある．そのためには，顧客の業務や課題を熟知していなければならないが，このような顧客業務の理解のための体系的な方法が確立されていない状況にある．そのため，カスタマージャーニーマップやヘルススコアの設計も CS に取り組む企業がそれぞれ手探りで進めている状態であり，CS の品質は CSM のスキルや経験に依存してしまっている．つまり，顧客業務の理解不足が，結果としてチャーンの増大や事業拡大を阻害する要因となっている(図 3).

この顧客業務の理解は，CS を実現する上での重大な課題であるといえ，体系的な顧客業務の理解が可能であれば，CSM 業務を標準化することができ，サービス品質の向上や事業展開も可能になると考えられる．

業務を理解する手法として，BPMN[4]に代表される業務プロセスのモデリングがあるが，これらは次章で示すタスク視点の手法であり，下記のデメリットがあり，CS にはあまり使われていない．

- 記述した業務プロセスモデルの「正しさ」を確かめる有効な方法がない
- 複雑な業務だと自社のシステムやサービスの位置付けがわかりづらくなりやすい

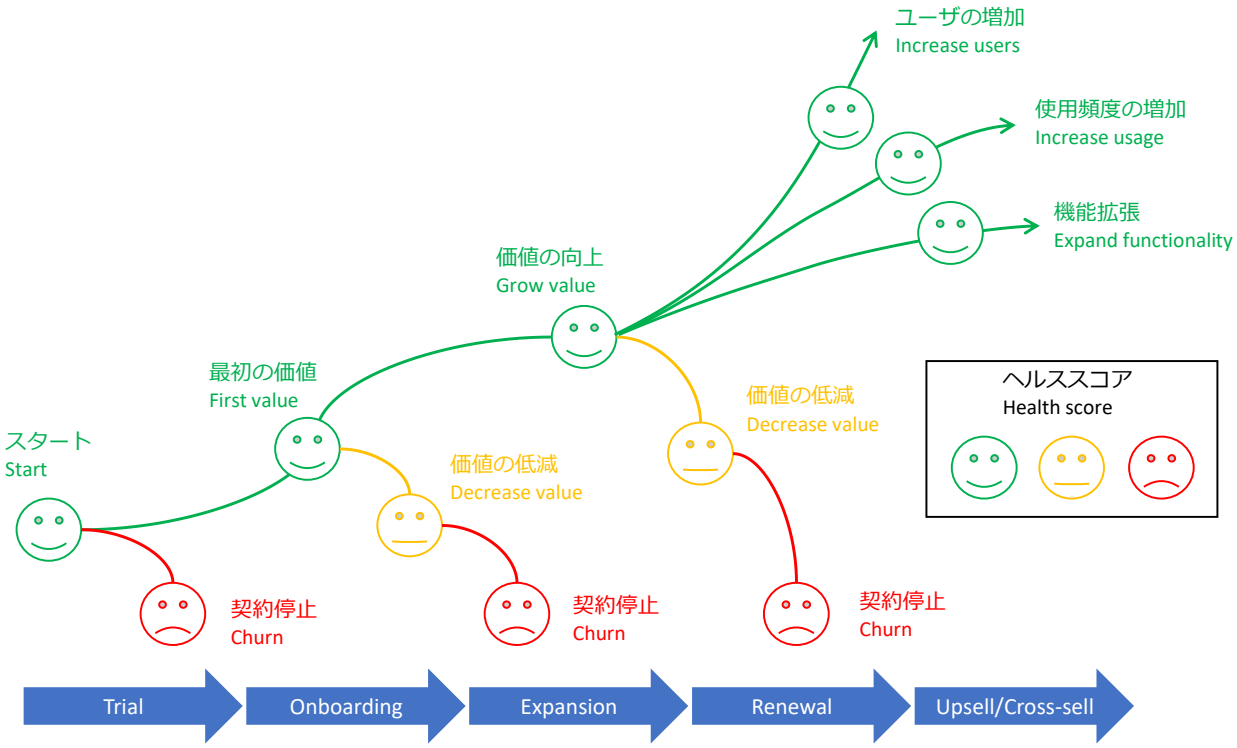


図 2 一般的なカスタマーのシステム・サービス利用のイメージ

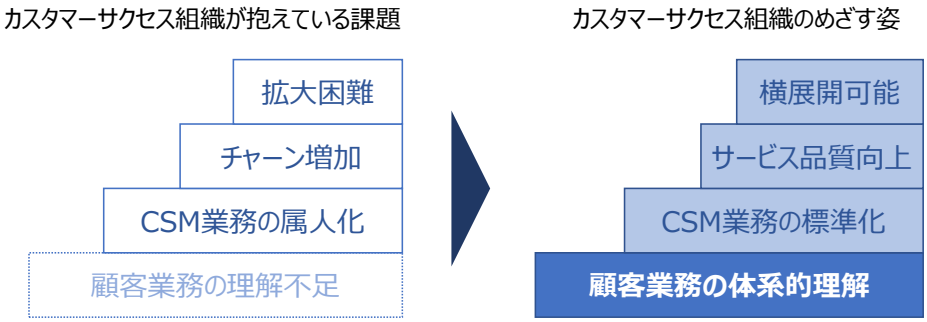


図 3 カスタマーサクセスの課題

3. カスタマーサクセスのための PReP Model

3.1. PReP Model とは

業務モデリング手法には，表 1 に示すタスク視点と成果物視点の二つの分類が存在する．タスク視点は，前述の BPMN のように処理やプロセスを記載するため，モデルを作成・理解しやすい反面，モデルサイズが膨大になりやすく，業務の要点もわかりづらくなる．一方，成果物視点は，成果物同士の関係性に注目して，全体最適な視点で業務を捉えられるため，モデルがシンプルになり，業務の要点も理解しやすいが，モデリングが難しいという側面がある．

PReP Model は，成果物視点の業務モデリングに属している[5][6][7]．そのため，業務を端的に表現できるというメリットがある一方，モデリングにスキルが必要になる．PReP Model の記法は，表 2，図 4 に示すとおりであり，業務を成果物の繋りで表現する．また，業務の役割分担をアクター毎にモデリングし，アクター毎の業務のインプット情報，アウトプット情報を明確化する．さらに， PReP Model の専用ツールを利用することで，構造的なモデルチェックを行うことができる．

表 1 業務モデリング手法の視点

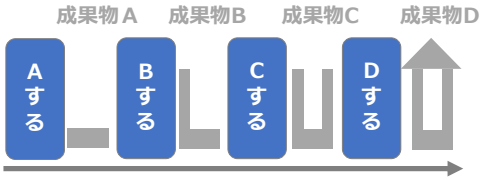
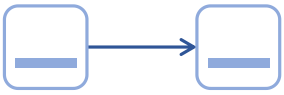
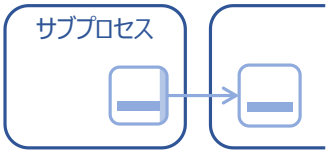
	タスク視点(タスク：処理/行動)	成果物視点(成果物：情報の意味的まとまり)
メリット	•処理/行動はイメージしやすいため、 モデリングしやすく、**読みやすい**	•モデルが**コンパクト**になる •**経営目標**に照らし合わせて成果物を 見直すため、**要点が定めやすい**
デメリット	•抽象的なレベルや作業レベルが混在するため、 **膨大なサイズ**になる •業務が複雑だと**課題が見えづらい**	•経営目標から見た情報や帳票の意味を 洗い出しながら関係性を明らかにするため、 **技能と経験が必要となる**
イメージ		

表 2 PReP Model の記法

記号	説明	記号	説明
	**最終成果物** 当該プロセスから外部のプロセスに出力される成果物		**入出力関係** 入力元から入力先へパラメータの値が入力される関係
	**マイルストン成果物** プロセスの品質リスク管理のためのコントロールゲートとなる成果物		**同期関係** パラメータが相互に関係して、 それぞれの値が決定される関係
	**中間成果物** 上記以外の成果物		**外部プロセス** 当該プロセスに成果物を入出力する外部のプロセス
	**無実体成果物** 知見や声による指示など、実体がないが共有される成果物		**アクターとスイムレーン** 当該プロセスのゴール実現の責務者(アクター)と責務下のレーン
状態を持つ 	**パラメータを持つ** 		**サブプロセス** マイルストン成果物までのひとまとまりであり，プロセスの部分集合

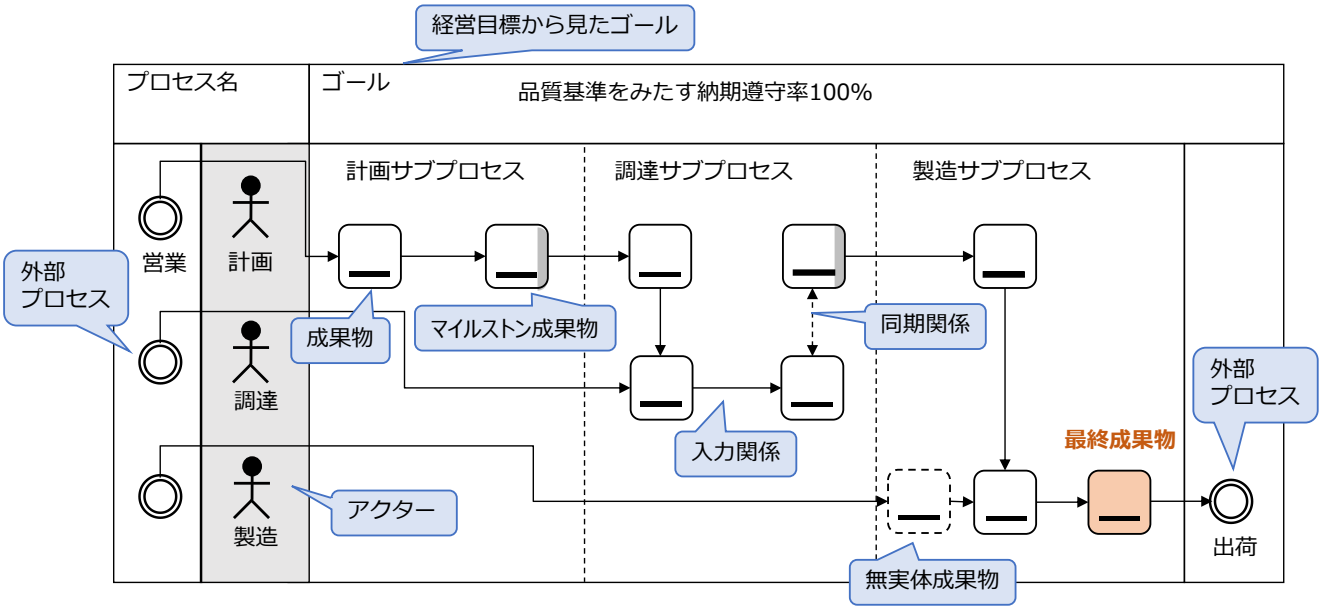


図 4 PReP Model の記述例

3.2. 顧客業務理解のための PReP Model

本報告では，CS 実現を目的とした顧客業務理解のため，PReP Model を用いて，顧客の業務プロセスを分析する方法を提案する．顧客の業務理解のために PReP Model を用いるメリットは下記である (図 5).

- PReP Model 専用ツールによりモデル不整合チェックができるため，「正しい」モデルを記述できる
- 自社のシステムやサービスと CSM の介入ポイントを明確にすることができる

また，CS では，ヘルススコアの設計が重要となるため，顧客の KPI (Key Performance Indicator) をシステムやサービスがどのように変化させるかを見える化する「KPI ツリー」を作成する．PReP Model では，各業務プロセスのゴールと対応する最終成果物を明確にしていくため，顧客業務の KPI を構造化しやすい (図 6). ただし，PReP Model だけでは KPI に関する情報が不足することが考えられるため，その場合は補完が必要となる．

本提案手法である，CS 実現のための業務モデリングの進め方の全体イメージを図 7 に示す．まず，顧客の AsIs 業務モデルを CSM や顧客にワークショップの中で作成し，完成した AsIs 業務の PReP Model から，AsIs の KPI ツリーを作成．顧客の経営ゴールに照し合わせて，自社のシステムやサービスが導入された後の ToBe の KPI ツリーを構造化し，ToBe 業務の PReP Model の設計につなげる．ToBe 業務モデルでは，自社のシステム・サービスや CSM の役割も明確にする．

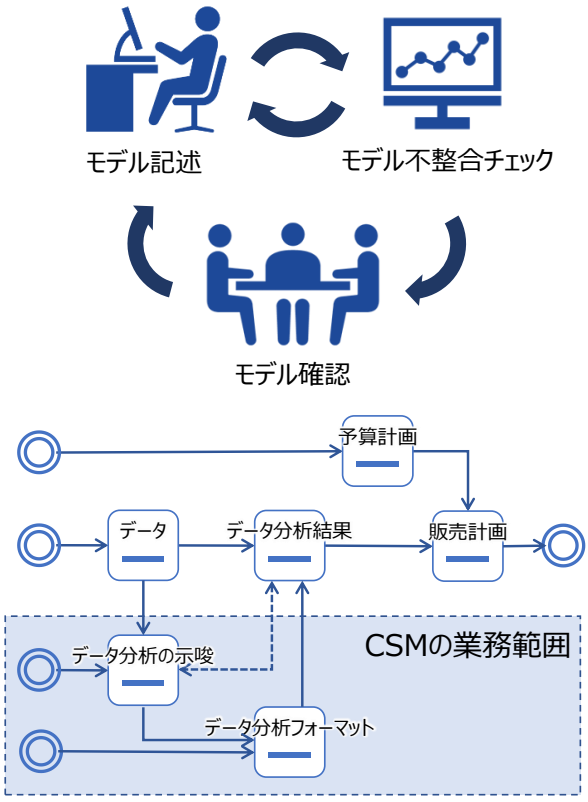


図 5 カスタマーサクセスのための顧客業務理解に PReP Model を用いるメリット

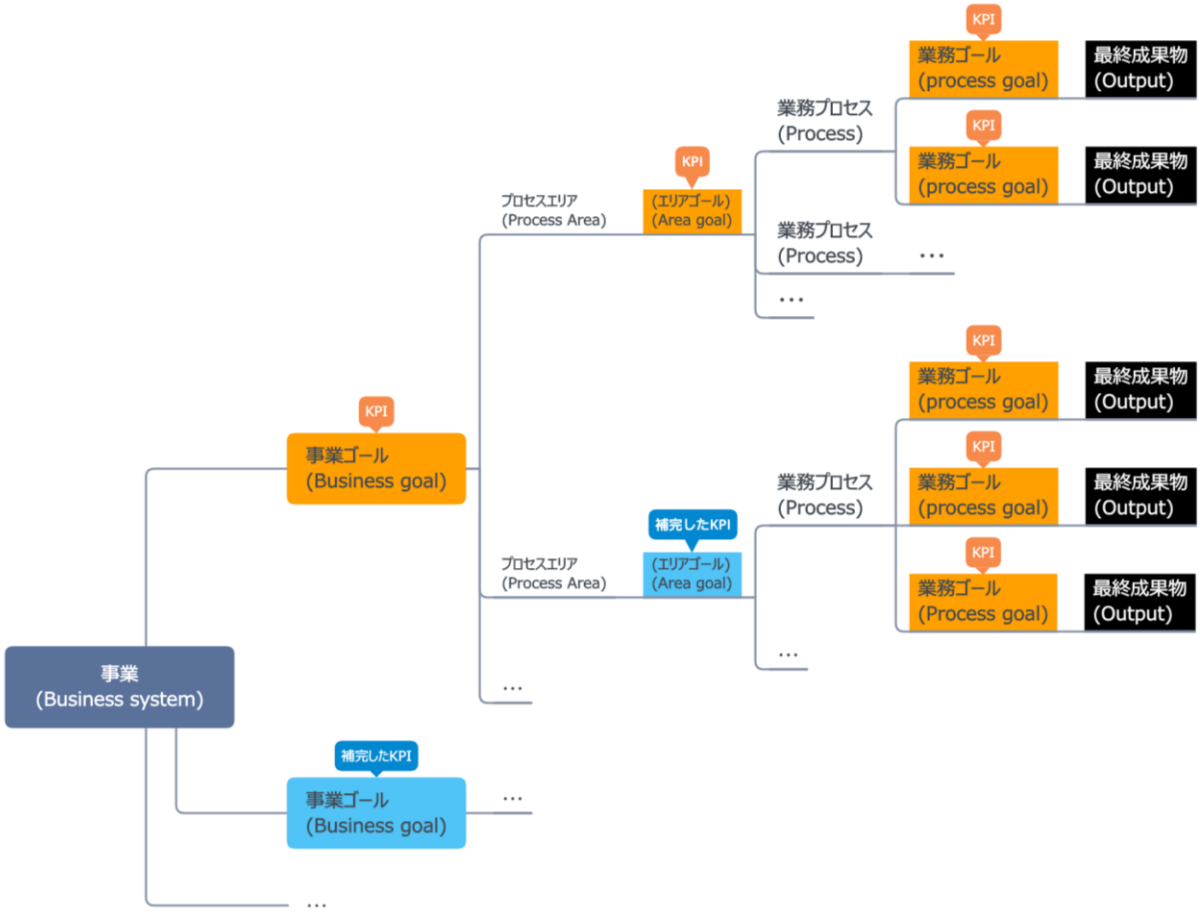


図 6 PReP Model のゴールと KPI ツリーの関係

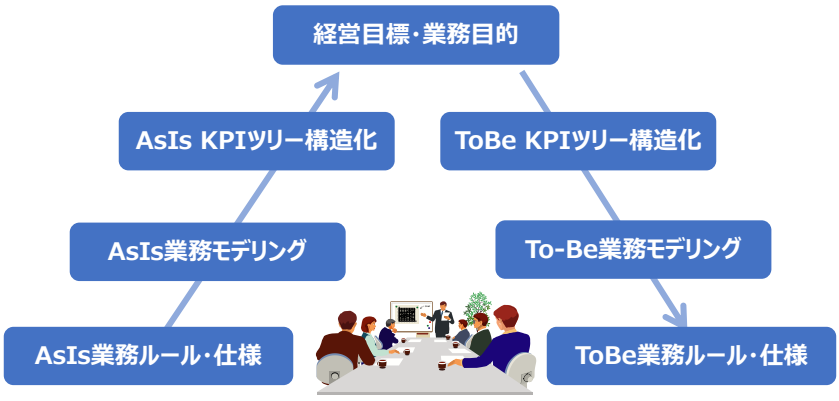


図 7 カスタマーサクセス実現のための業務モデリングの進め方

4. AI 系 SaaS での試行

4.1. 概要とスケジュール

本章では，実際の AI 系 SaaS を用いた業務を対象に，本提案手法の効果を検証する．

全体のスケジュールは，表 3 に示す通り全 8 回のワークショップにて，業務モデリングを実施した．

検証の対象業務は，株式会社 ABEJA の ABEJA Insgiht for Retail[8]*という顧客購買分析サービスを用いたアパレル業務であり， 業務モデリングワークショップには，ABEJA の CSM がメンバーとして参加した．

表 3 業務モデリングワークショップの日程と内容

#	日時	内容
第1回	3/31 17:00-20:00	・顧客課題のヒアリング ・モデル化のテーマ選定
第2回	4/9 17:00-19:00	・AsIsのモデル化(2/4完成)
第3回	4/16 17:00-19:30	・AsIsモデルの振り返り(電子化版の確認) ・残り2業務のAsIsのモデル化
第4回	4/22 17:00-20:00	・AsIsモデルの振り返り(電子化版の確認) ・AsIs/ToBeのKPI確認
第5回	5/7 17:00-19:10	・修正したAsIs/ToBeのKPI確認 ・AsIsモデルの振り返り(電子化版の確認)
第6回	5/14 17:00-19:30	・ToBeモデルの作成
第7回	5/20 17:00-19:00	・ToBeモデルの振り返り(電子化版の確認)
第8回	5/29 17:30-19:00	・ToBeモデルの確認(最終)

4.2. 作成した PReP Model

表 3 のワークショップにて作成した AsIs 業務と ToBe 業務の PReP モデルを図 8，図 9 に示す．図 8 より，PReP モデルで顧客業務の AsIs を描くことで，属人化してしまっている業務を「無実体成果物」が同期関係でつながっているという形で明らかにすることができ，サービスはその業務に介入しようとしていることを明確にすることができた．また，図 9 は図 8 と同じ業務を表しているが，サービスを導入した後の ToBe として，CSM が顧客と関わる線を緑色で表現している．このように複雑に CSM が顧客に寄り添って価値を提供することで，顧客にとっては，サービスがなくてはならないものになるようにしていることがわかった．

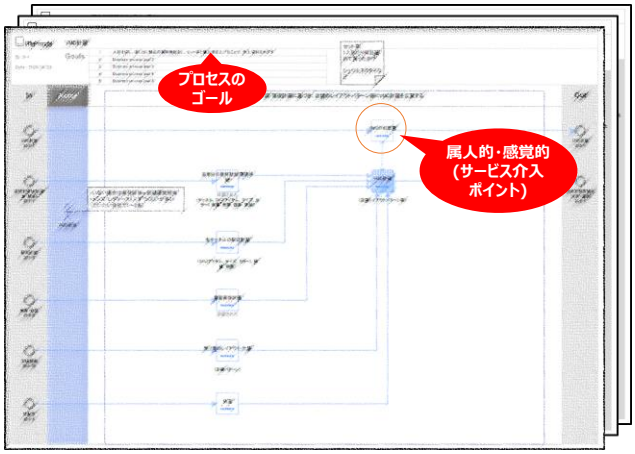


図 8 顧客の AsIs 業務モデル

図 9 顧客の ToBe 業務モデル

4.3. 作成した KPI ツリー

表 3 のワークショップにて作成した AsIs 業務と ToBe 業務の KPI ツリーを図 10，図 11 に示す．図 10 の AsIs の KPI ツリーは，図 8 に示す AsIs の PReP モデルから，KPI に相当するものを抽出し，不足分を CSM にヒアリングすることで，完成させた．さらに，図 11 の ToBe の KPI ツリーにおいて，赤の四角で記載したものが，サービスを導入することで得られる新しいデータ，KPI である．

このように，PReP モデルから KPI ツリーを用いて，指標を構造化することで，サービスの立ち位置と提供価値を明確にすることができたと考えている．

* ABEJA Insight for Retail は，オンライン購買解析のリアル店舗版と呼べるものであり，アパレルの店舗に設置したカメラ画像から，顧客の回遊状況，購買行動などをデータ化・解析することで，商品の配置や接客のやり方などの取り組みの効果を見える化するサービスとなる．

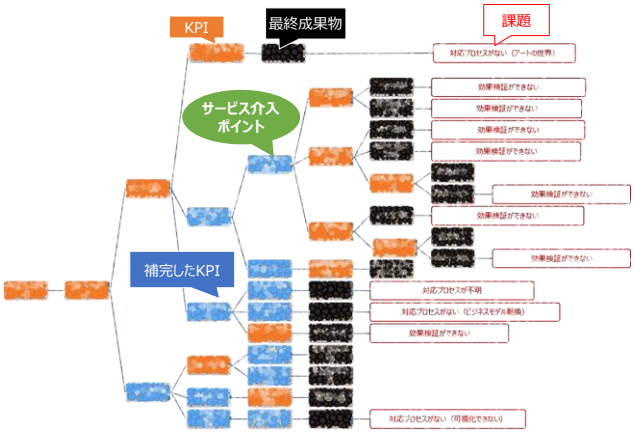


図 10 顧客の AsIs 業務モデル

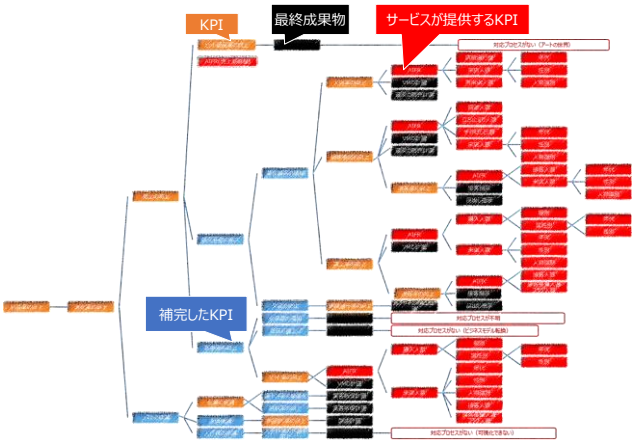


図 11 顧客の ToBe 業務モデル

4.4. 試行で得た知見

本試行で得られた知見を図 12 のようにまとめた．アパレル業界では，多くの場合，売上で重要となる集客販促立案，商品陳列，販売指示といった業務に「肌感覚」という暗黙知が含まれており，どのような施策が売上に効果があるのかが見えなくなってしまっている．試行対象のサービスは，その「肌感覚」を顧客の購買行動という実績データで見える化するサービスであるが，データの分析方法や読み解き方などは難しい側面がある．その難しさを解消するのが CSM の役割となっており，CSM の示唆があることで，システム・サービスが有効に働き，顧客を成功に導いていることがわかった．CSM は，このような示唆をマニュアルなどのドキュメントとして形式知化することで，サービスを多くの顧客へ展開できるようにしている．

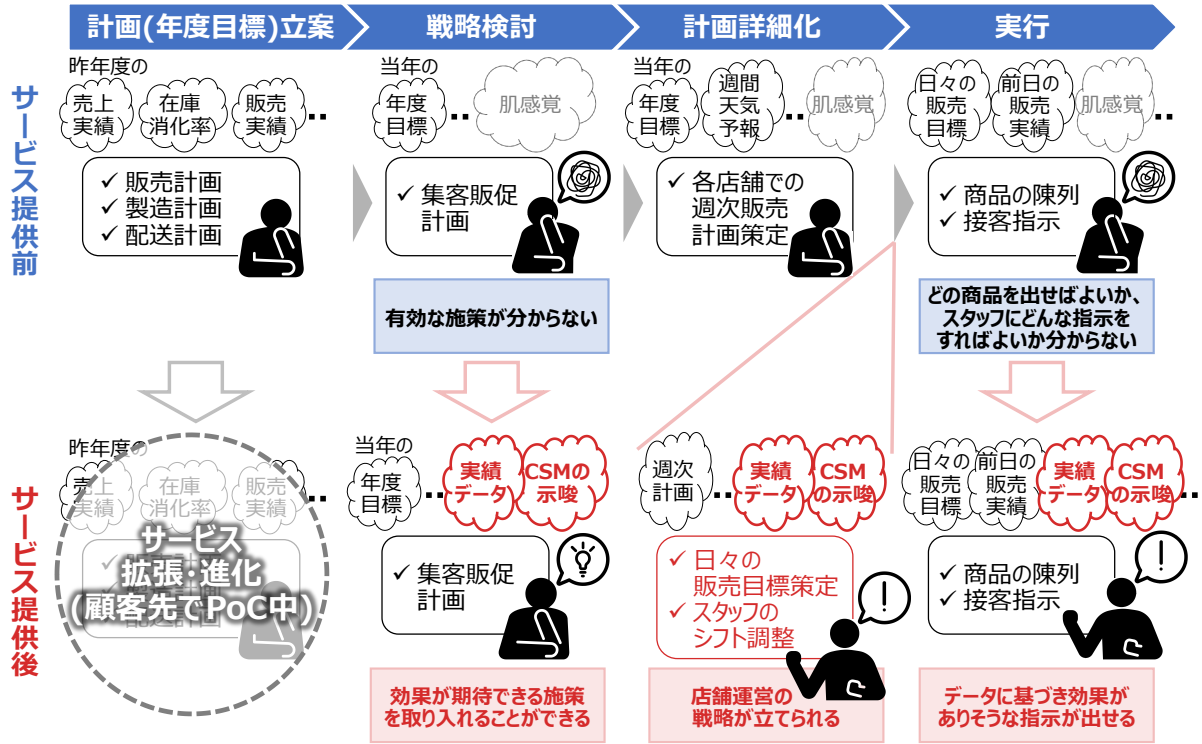


図 12 アパレル業界の顧客業務理解まとめ

4.5. システム機能拡張への応用

一般的に SaaS のシステムは，利用中のサービスを拡張していくため，絶えず機能拡張が行われる．試行対象の AI 系 SaaS も同様にシステム機能拡張の計画があり，本試行で作成した PReP Model，KPI ツリーを開発チームと連携した．

結果，顧客の課題やサービス改善のポイントの伝達がスムーズに進められ，当初 1 年間以上必要と考えていたテストマーケティング用の新機能のα版をわずか 3 ヶ月でリリースすることができた．

さらに，テストマーケティングでの 50%のコンバージョン率(サービスの購入・契約の割合)獲得という目標に対し，83%という大幅な目標達成ができた(表 4)．この結果，社内外の主要なステークホルダーからも高い評価を得ることができた．

表 4 PReP Model による業務分析のシステム機能拡張への応用結果

	想定	本手法の効果
開発期間	1 年以上	3 ヶ月
コンバージョン率	50%	83%

4.6. CSM へのインタビュー結果

本試行に参加した CSM に意見を収集したところ，以下に示す通り，総じて高評価を得ることができた．

Q1：CS 実践の活動を具体化するために，このアプローチは有効か？

めちゃくちゃ有効だと思う．CS の実践活動は，お客さんの理解，自社のサービス理解，CS の戦略策定+実行となる．ここで CS が失敗する 1 番のポイントはお客さんの理解が足らない（課題の粒度深堀が不十分）ことが多く，お客さんのサービスを活用するときの課題感とニーズがつかめてないと，CS の活動が押しつけ的になることが多い．PReP Model があると手段の目的化を防げると思う．CS が自社目線に立つと，サービスを使ってもらう or サービスを理解してもらうというスタンスになる．自社目線に立つとワークしないので，お客さん目線に立っているかが重要なので，本来の目的は「利益創出やビジネスゴールをどう達成するか？どうサービスをつかってもらうか？」について CS が大事にすべきポイントである．

Q2：PReP Model や KPI ツリーについての評価は？

とてもよい．これがあるだけで，CS の戦略活動の質と効率が上がる．ABEJA でも PReP Model の明文化については，半分できていて，半分はできてない印象である．Product Market Fit まではいくつかのステップ (Customer Problem Fit → Problem Solution Fit)をお客さんのことをわかっていてプロセスに落とし込むが，お客さんの理解や構造化は組織体ではやり切れていない．大体の場合は，創業者やトップメンバーがトップ営業で顧客課題をつかんで必要な機能が実装している．スタープレイヤーの属人化した活動を，一般の人が実践するまでの形式知化はできていない．情報を探す価値でグーグルに勝てない以上，情報の成果物価値は，体系化や構造化されていることにあると考えている．PReP Model のようなグーグルにできない体系化や構造化できることは価値になると思う．実際に共同ワークをしてみて，自分の中でぐちゃぐちゃになっているものが体系化されて出てきたのはありがたかった．

Q3：成果物を自身の CS 業務で使えるか？(いつ，どこで，誰に)

使える．

- いつ：計画・立案時，
- どこで：CS 業務，カスタマージャーニーマップ作成など，
- 誰に：CSM のような責任者など，組織活動をしたいと考えている人．

Q4：CS 推進しているサービスビジネス企業にとって，PReP Model の成果物は嬉しいか？

嬉しい．Product Market Fit 前の段階にいるサービサーは喜ぶと思う．ターゲットが決まる B2B には向いている．B2C でもニッチ領域(ターゲットが明確の場合)ならワークする．ただし，プラットフォーマーには響かないと思う．

Q5：改善点があるか？

最初に全体の流れがわかると嬉しい．また楽しかった点でもあるが，プロセスを洗い出すのが難しかった．これがやれる人材は限られているという課題を克服する必要があると思う．ヒアリング相手の言葉を PReP Model に落とし込む翻訳力がボトルネックになっていると思った．

Q6：その他，感想があれば

今回のような成果物が出るのであれば，他社でもニーズは確実にあると思う．実際に自分が外部顧問する会社に持っていきたい．

5. まとめ

本報告では，カスタマーサクセスを実現するための PReP Model の応用手法を提案した．SaaS をはじめとするサブスクリプション型のビジネスモデルでは，顧客が解約しやすくなるため，解約率を下げるカスタマーサクセスが重要となる．カスタマーサクセスは，解約防止だけではなく，上位プランへのアップグレードや別システム・別サービスへの拡張など事業拡大にも貢献する．一方，カスタマーサクセスはまだ新しい概念であるところもあり，方法論が確立できているとはいえない．特に，カスタマーサクセスを担うカスタマーサクセスマネージャは顧客に寄り添い，顧客の事業的成功を支援する立場になるが，顧客の業務を熟知しなければならず，現在はこの顧客の業務理解をカスタマーサクセスマネージャのスキルや経験に依存している状況である．そこで本論文では，顧客業務理解のために業務プロセスの要点を分析できる PReP Model を用いた手法を提示した．

本手法を適用したところ，システム開発チームとの連携が円滑になったことで，1 年以上と想定されていた開発期間が 3 ヶ月に短縮できた．また，テストマーケティングにおいて，50%という

目標に対し，83%と大きく上回るコンバージョン率を達成することができた.

参考文献

[1] ニック・メータ, ダン・スタインマン, リンカーン・マーフィー, “カスタマーサクセス―サブスクリプション時代に求められる「顧客の成功」10 の原則,” 英治出版，東京，(2018).

[2] 弘子ラザヴィ，“カスタマーサクセスとは何か―日本企業にこそ必要な「これからの顧客との付き合い方」,” 英治出版，東京，(2019).

[3] “カスタマーサクセスカンファレンス「SUCCESS4」,” (2019), https://success-lab.jp/success4/

[4] OMG,“BPMN 2.0”, https://www.omg.org/spec/BPMN/2.0

[5] 田中康，“PReP MODEL – 現実世界をデザインする -: PReP モデルによる業務レベル設計，” (2022).

[6] 田中康，飯田元，松本健一，“成果物間の関連に着目した開発プロセスモデル: PReP Model,” 情報処理学会論文誌, Vol.46, No.5, (2005).

[7] 田中康，後神義規，光井邦雄，“要求開発プロセスへの PReP モデルの適用,” ソフトウェアシンポジウム 2015, (2015).

[8] ABEJA Insight for Retail, https://abejainc.com/insight-retail/

「オントロジー」など「汎用語」の情報処理分野への転用史

塚田 良央
(株)ジェーエフピー
tsukada_yoshio@jfp.co.jp

佐々木 千春
(株)ジェーエフピー
sasaki1000@jfp.co.jp

漆原 憲博
(株)ジェーエフピー
japanfp@jfp.co.jp

栗田 太郎
ソニー(株)
taro.kurita@sony.com

要旨

新たな語の登場は，既存の学問や産業領域の転換を想起させてきた．そう考え，情報処理分野で広く用いられている「システム」など,いわば「汎用語」の変遷をたどってみた．本論文で扱う語は，「アーキテクチャ」，「オントロジー」，「システム」，「モデル」とした．変遷とは他の分野からの語の転用（導入）や変化などを指す．これらが，われわれの開発現場にどう影響するのか，効果はどうか，を探ってみた．

「システム」や「アーキテクチャ」は導入当初はいかであれ，いまは不可欠で誤解の生まない，あるいは生みようのない語で，特に「システム」などはどの分野でも，また日常生活のどこででも使用されている語になった．しかし，語の普及また定着の一方で，「システム」などは時代の要請から安全などが問われ，語もまた例えば「system-theoretic（システム理論的）」などと意味を発展させた新たな方法論が進展している．

「モデル」は説明されることはあっても，定義はされない語ではないかと思われる．なんでも「モデル」になる，と時に陰口も聞くが，定義されない，したがって他を定義する語としての役割があるようである．陰口もここに起因するなら「モデル」に罪はなく，むしろ，業界には孝行息子というべきである．

時代の課題（要請）があり，新たな語が投入される．それぞれの語を概観しつつ，最後に「オントロジー」に少し紙幅を費やすこととした．Web 上，また要求仕様分野など，「概念」の「存在」のための方法論であるが，哲学から借りた用語である．

1. はじめに

「オントロジー」などという異分野の，いささか耳慣れない語が，組込みシステムの開発現場にも現れたりすると，現場やその取り巻く環境に新たな変化が起きたのか，などと思う．そして，異分野の耳慣れない語が新たな産業に浸透し，産業を発展させて来たことに気づかされる．

新たな産業といえば，われわれの依拠するコンピューター産業も，社会にその職業が認知されていまだ半世紀前後のものであろうと思われる．「アーキテクチャ」は「オントロジー」と対比するには，あまりに耳慣れた語であるが，「アーキテクチャ」も元々は異分野である建築業界の用語であったろう．「システム」もまた，当業界ではいまやあまりに当たり前であるが出自があったに違いない．そして，ときにバズワードともいわれかねない「モデル」もしかりである．

もっとも，出自といっても，これらの語がすべて海外生まれであることも，日本人には一つの「出自」にはなる．しかし，これらの語はどれも日本生まれではないから，日本のいわば「やまとことば」が持っているとする意味から辿ることはできない．言語間ハンディとでも言うべきものを負っているわけであるが，ここではそのことはいったん置き，本論文で異分野の語という場合，産業や学問などの分野の違いということとする，

以上挙げた語，その中でも「システム」はコンピューターの開発業界では広く使われているという意味で「汎用語」と命名できるかもしれない．あるいは，単に広く使われているだけではなく，仕事上の意味伝達をする上で，基本的な役割をするので「基本語」というべきかもしれない．あるいは，意味の領域区分のようなものがあり，その領域の基盤，あるいは土台のようなものを成しているという意味で，「基盤語」などという命名も可能かもしれない．

本論文では，用語の出自や，用語が当業界に登場してきた経緯などを変遷としてたどる．そして，そのことが現場の，これらの語が持つ課題へのヒントになればと期待するものである．

2. 用語の変遷

2.1. 用語「アーキテクチャ」の変遷

「アーキテクチャ」という語は本来建築分野の用語で「構造物」や「構造」を指すといわれる.

コンピューターに対してこの「アーキテクチャ」の語を使用したのは, 1960 年, Johnson が始まりである[1]. また複合語「システム・アーキテクチャ」は, あるシステムを例に, そのシステムを構成する基本的な要素と, それぞれの機能, およびそれらの機能が協調して動作する方法を意味していた. したがって, システム・アーキテクチャは, 論理設計や回路設計よりも抽象的な階層を指していた. この複合語は, 1964 年の IBM 社のコンピューターSystem/360 で最初に使われたといわれる[39].

ソフトウェアに対して「アーキテクト」の語を使用したのは, 1969 年の会合が始まりである[2]. Naua は会合で, 「ソフトウェアデザイナーの仕事はアーキテクト(建築家)に似ている, これは大規模で異種混交(heterogeneous) な構造を対象とするからである」, と発言した.

続く 1969 年の会合での Sharp の発言では「ソフトウェア・アーキテクチャ」の語を用い, その意義を強調した[3]. Sharp は, ソフトウェアの細部が理論面でも実践面でも素晴らしいものになったとしても, 全体を統合することがうまくいかなければソフトウェア開発は失敗すると論じた. ソフトウェア・アーキテクチャは, この場合, ソフトウェアの全体を統合するという課題を指して用いられた.

その後, アーキテクチャの語は, もっぱらシステム・アーキテクチャ, すなわちコンピューターの物理的な構造に対して使われた[4]. 狭い意味では, コンピューターのインストラクションセットを意味することも多かった[4].

1991 年には Royce 親子がソフトウェア・アーキテクチャの語を用いた[5]. その後, 1990 年代にかけて, ソフトウェア・アーキテクチャの研究が興隆を見た[4].

日本でも, 1970 年ないし 71 年には, 「アーキテクチャ」の語が使われた[6],[7],[8],[9]. これはコンピューターのハードウェアの基本的な構造ないし機能を意味する. またこの語「アーキテクチャ」は System360 や後継の System370 への追随を狙っていた日本の研究者や技術者の間では使用されていたであろう. また 1970 年前半に System370 を導入した日本企業のエンジニアの間でも同様である.

2.2. 用語「オントロジー」の変遷

「オントロジー」は日本語では「存在論」と訳され, 究極の存在に関する哲学的議論とされてきた. 存在の究極を質料と形相とした形而上学を打ち立てた古代ギリシャのアリストテレスに端を発するといわれるが, 「オントロジー」という用語が哲学の領域に正式に登場したのは18 世紀といわれる.

しかし, 存在に関する議論が物理科学の常識をあまりに超えたことから, 経験科学や数理論理学の立場から存在を論ずる議論が起こった. 哲学者カルナップやクワインなどの, 構文と意味規則を定義しつつ, 理論の系をはみ出すことなく存在を議論する方法や, 在るとは何か(what there is)と, 存在記号“ヨ”を問いながら平易に「在ること」を論じる方法は, 20 世紀半ばに大きな流派を築いた一群の議論である.

このような論理的また実証的な立場の哲学の方法が, こんにち Web 意味論など「オントロジー」の方法にも見て取ることができる.

2.2.1. 情報処理分野

Powers は自然言語の機械学習についての論文で「オントロジー」の語を用いている[12]. オントロジーについての言及があるのは記号接地 (symbol grounding) についての節である. Powers はオントロジーという語を明確に定義していないが, 語彙と環境(生物あるいはロボットにとっての) の総体, すなわち存在, という意味合いのようである.

Gruber がオントロジーという用語を使用した当初は, 知識ベースがドメインごとに孤立して存在しており, 知識の共有と再利用が容易でないことが問題意識としてあった[13]. 共通の知識ベースの開発が Gruber の関心の対象である. Gruber は, 「共通オントロジー」を提唱するが, これは, 複数の知識ベースを接続し, 知識の共有と再利用を可能にすることを意図したものである.

その後, Gruber は語「オントロジー」を「概念化の明示的な仕様書(an explicit specification of a conceptualization))と定義し, この語は哲学から借りたという. オントロジーは「存在(Existence)に関する体系的な言説」であり, 「存在するもの(what “exists”)」はまさに表現可能なものであり, それゆえ, 知識ベースの開発に「オントロジー」を用いる, とした

[14].

また，上の「概念化」を Gruber は，Genesereth らを引用し，ある関心領域に存在すると推定されるオブジェクト、概念、その他のエンティティを保有する関係と定義している[15]．そして，Gruber によれば，概念化とは、何らかの目的のために表現したい、抽象的で簡略化された世界の見方である，という[14].

Feilmayr と Wöß は，オントロジーの定義を「複雑さの増大によって要求される高度な意味論的表現能力に特徴づけられる，共有された概念化の形式的で明示的な仕様書」とした[16]．上の Gruber らの定義をさらに詳しくしたものである．

2.2.2. 日本での展開

日本では，1989 年から 1991 年にかけて，定性推論と自然言語処理に関連して「オントロジー」の語が見られるようになった[17],[18],[19]．この時期の「オントロジー」の用法は論者によって異なる．

1993 年には中島らによるソフトウェア工学の論文で「オントロジー」が言及されたが，その応用の可能性については否定的な見解となっている[20]．中島らによる論文では，オントロジーを「システムを説明するために必要な基本的な考え方とそれに基づく具体的な概念の集合」と定義している[20].

他方，同じ 1993 年の人工知能分野での論文では，ティヘリノ A．ジュリらは「オントロジーとは，哲学では存在論という意味であるが，人工知能では物事を表現する際の基本となる最小の単位（プリミティブ）の集合という意味を持っている」としている[21].

このように，1990 年代初頭においては，「オントロジー」という語の定義や用法は論者によって異なり，オントロジーという概念が導入途上であることがうかがえる．ただし，オントロジーに「基本」とか「要素」などプリミティブな意味を持たせているのは共通である．

2.3. 用語「システム」の変遷

1940 年代から 50 年代にかけて，Wiener や Ashby などによって，サイバネティックス論，さらにシステム論という学問分野が盛んになった．"system"という語も，それに応じてよく使われるようになった[22],[23].

一方，ソフトウェアの分野ではどうか．"system software"という語は，例えば 1972 年の文献では，"user software"と対比させて使われている[24]．ベンダーの責務で提供するソフトウェア（system）と，ユーザーの責務で作成するソフトウェア(user)との間に境界を設けている．用途別の形容を加えて，語「システム」を使っている．

その後は，同じ事柄を指すとしても"system software"と"user software"よりも，"operating system"と"application program"または"application software"が多く使われるようになる[25].

1973 年当時日本のコンピューター技術をリードしていた富士通のコンピューターの解説書でも，“システム”と“ソフトウェア”の語はあるが，両語が組み合わされた“システムソフトウェア”の語はない．“オペレーティングシステム”は存在する．また“ソフトウェア”の下位に“プログラム”が位置している．“ハードウェア”は“ソフトウェア”と同じ位置にある．また“アプリケーションプログラム”は“ソフトウェア”，“ハードウェア”と同位置にある[33].

以上のことは，新たな語の導入の歴史が浅ければ浅いほど，指示対象が不安定であることを物語る．と同時に，指示すべき新たな事柄も生まれていなかった，ということも示すであろう．

2.4. 用語「モデル」の変遷

「モデル」という語はさまざまな分野で使われており，「モデル」を含む語句も多くある．ソフトウェアの分野に絞っても，例えば，以下の語句がある．これらの語句は，何らかの方法論を示す「用法」とでも呼べばよいかもしれないが，語の変遷という視点から，単に語句と呼ぶこととする．

ソフトウェア分野で"model"の語句といえば，UML すなわち"Unified Modeling Language"がある．UML には源流となった 3 つのアイディア，Booch，OMT，OOSE がある[31]．このうち"model"という語を使用しているのは OMT，すなわち"Object-modeling technique"のみである[32].

"software process model"という語句もある．この語句の代表例が"waterfall model"であるので，これらの語句の登場の時期を調べると，1986 年には"software process model"という語句が使われている[28]．1983 年あるいは 1988 年には"waterfall model"という語も見える[29],[30]．ただし，"waterfall chart"という語も使われており，こちらの方が使用頻度が高い．"model"よりも"chart"の方に語の利便があったということであろうか．あるいは，"model"が"chart"よりも語としてのなじみがなかったということでもあったろうか．ただし，その後"waterfall chart"は"waterfall model"に置き換

わって行く.

3. 用語の概観

以上の4つの用語を取り上げた，その順は特に意図があったわけではない．要求仕様の開発の現場で，よく「アーキ」が使われているので，そこから論を始めたに過ぎない．「オントロジー」は「アーキ」の連想である．アーキテクチャが落ち着かないなら（いろいろ図を書き変えざるを得ないなら），「オントロジー」なら単にいろいろ書いてみるのではなく，2，3の基本的な文から推論などで「アーキ」を仕上げる手法かと推測したからに過ぎない．その後，調べは「システム」，「モデル」と進んだ．

本章では，以上の調査を経たので，用語を関係づけながら概観することとする．

3.1. システムとアーキテクチャ

「システム」も「アーキテクチャ」も当業界では不可欠な用語である．ただ「システム」の方がより広く使われているであろう．開発現場では業務のプロセス名を，システム開発，システム検証などと「システム」を用いて表し，またなにかを調査する場合には，調査の対象を「○○システム」などと，今度は対象○○に「システム」を付け加える．

また，「アーキテクチャ」は先に見たように，「システム・アーキテクチャ」などと使われ，「システム」と「アーキテクチャ」はよく連結して用いられる．そして，「アーキテクチャ」を簡略に「アーキ」と呼んだりする．

このように「システム」も「アーキテクチャ」も語が「熟して」とでもいうべきか，間断なく，また誤解もなく使われているように思われる．

しかし，ふと「アーキ」と簡略化して言う場合，個別の回路部品を指しているのであろうか，それとも構造を指しているのであろうか，などと疑問が湧く．出自はもともと後者であると思われるが，「このアーキは大丈夫？」などという場合，特定の個体（ここでは部品）を指していることも多い．

もちろん，語は語られる分脈で意味を持つ，と言われれば，この議論は大方片が付くのであるが，多少込み入った議論がある．

先に「変遷」の節で見たように，再確認的にいうとアークテクチャは，要素も指すが，構造を指す．構造を指すのは，アークテクチャが建築分野を出自にするからと納得できる．また「アーキテクチャ」がかつて「設計思想」と訳されたことも，「設計」という構造的側面に視点が置かれたせいと思われる．

しかし，「アーキ」と短く言う所為か，構造ではなく，要素，あるいは個体を指す，ちょっとした意味の変化が見られるような気がする．

他方「システム」はどうか．システムの定義の一つを見ると，システムは単なる要素の集まりではなく，要素どうしの秩序や関係を持つものとある[34]．要素が個別に，いわばバラバラあるのではなく，秩序を持ってきちんとあるものを，特に「システム」と呼ぶということである．

すると何か「システム」も，中に秩序は持っているが，一つとしてふるまうときには単独の個体に見える．すると，「アーキ」も「システム」も同じく個体という意味を指す．したがって，2つの語が連結しても，同じく個体を意味するのだから，同語反復ならぬ，同義反復にならないか，と少しうがった問いが湧いてくる．

しかし，実際には，「アーキテクチャ」もまた「アーキ」も，「システム」と連結されることで，固有の意味を取り戻すようである．つまり「システム・アーキテクチャ」は語が連結されることで，かえって両語の意味を際立たせ，ここでは「アーキテクチャ」は構造という本来の意味を取り戻すといえるのではなかろうか．

語が意味を，おそらくは文脈に応じて臨機応変に変化させ，しかし実際には固有の意味を「いざとなれば」取り返し，したがって誤解を生まないということは，いよいよ語が成熟していることを物語っているのかもしれない．このことが，特定の語が業界で長く生き残っている秘訣ではあるまいか．

ところが，このような機に応じた語の微妙な意味の変化は，すでにみたオントロジーと関係し，ややこしい問題を生む．先に Powers は記号接地をオントロジーの重要課題としている．記号接地とは，記号と記号の指示対象（存在）の関係をいう．記号の指示内容と指示対象が一致しているとき真という同定をわれわれはしている．

ディープラーニング（DL）技術では，この関係が怪しくなることがある．DL が例えば「シマ」と「ウマ」を判断できても，それだからといって「シマウマ」を自動的に合成し判断することはできない．DL には人間のできる認識の合成ができないというオントロジー（存在）に対する課題が残る．

この「シマウマ」の例は，上の「システム・アーキテクチャ」の話に似る．「アーキ」という「アーキテクチャ」の簡略表現が，単に語を簡略にしただけのはずが，オ

リジナルの長い語とは意味が変わってしまう．あるいは変わりやすくなる．

しかし，語が慣用的に使われてきた従来の語の結合に戻ると，従来からの本来の意味を取り戻す．このことは，語という記号と，記号の指示する事柄（意味）が一定ではなく，事情に応じて変化することを指す．「シマウマ」の例と異なり，議論の対象が目に見えない，「概念」の問題であるから，内面的で客観性に欠ける．これも議論をややこしくする．

この議論は語の成熟から発したものであるが，当然ながら，この議論は，文脈問題や複合語の問題で大方片づこう．

しかし他方，語には，生き延びてきた語ほど，語を認識する側（人間）とは独立に，語のサバイバル，概念の進化のような側面もあるのではないかという論点もありうるかもしれない．この論点は，オントロジーを形而上学と解した場合の概念の実在というような議論に近い．また数学の概念を，それを「存在」というフレームで語るときの議論とも似る．数学の世界はどういう様相で存在するか，という議論である．もちろん，「アーキ」などの語は経験的な場で使用されるものなので，同じに議論はできない．

本節の議論は，「システム」や「アーキテクチャ」のものであった．が，議論が DL などに派生したのも，これらの語が成熟し安定したうえで，場に応じたいろいろな意味を持つからと思われる．

3.1.1. System of Systems

挿話的にいえば，このごろ"System of Systems"を耳にする．語句は，複数の Systems を統括する System，という意味であろう．新たな語の導入は，導入を促す環境の変化にあるとすれば，すぐに思いつくのは，近来のシステムの「大規模・複雑化」である．この「大規模・複雑化」という表現は，システムに関する，サイズと内部構造の変化を表している．

"System of Systems"に戻れば，この"System of Systems"という語句は，この語句の先頭の"System"が，"of"のうしろに来る複数の"Systems"を対象物として統括的に制御すること，そして語"System"の階層は複数の"Systems"よりも上位の階層にあることを示している．

"System of Systems"や「大規模・複雑化」が少し誇張気味に表現している背後には，1つは安全性の課題がある．そして，その解法の1つとして STAMP が話題になる．STAMP は安全性の要求分析技法で，"Systems-Theoretic Accident Model and Processes"の略である．ここでは"Systems"に"Theoretic"を付記した方法論として提唱されている．

本論文は語に関する議論が中心なので，ここでの問いは"Systems-Theoretic"とは何か，である．この語は「複数システムの理論（化）」とでも解釈できるが，先に見たように，「システム」の意味は内部の（要素間に）秩序のあることを含んでいる．すると，「複数システムの理論（化）」とは，秩序をもった複数のシステムどうしを更に理論的（Theoretic）に秩序付けること，と理解される．秩序の秩序というような，概念の階層の違いに注意が要りそうである．

また，この概念の階層と関連するかもしれないが，STAMP の方法を適用するうえで，対象とするシステムが「全体は部分の単なる集合にはあらず」というホーリズムがアプローチの仕方と考えられている[40]．概念のいわば過不足の問題ともいえる．

STAMP はシステムに対して，事前にリスクとなる部分を見つけようとする．しかし，事故が発生して初めて，システムに事故の原因が潜んでいたことが明らかになる．すると，潜在的な部分が全体を超えていたということになる．全体，すなわちシステムの，その内部にある部分の秩序が十二分に把握しきれていなかった，ということでもある．

語の議論にもどる．「システム」などは，すでにみたように長くこの業界の語彙群の中核に位置している．他方，「システム」は語としてではなく，今度は安全という観点から，議論の対象となっている．語が単なる言葉としてではなく，具体的な現実の課題（この場合は「安全」）を持ち続けることが，この語の「生命」を生き生きとさせ，語のサバイバルを生き抜いている要因ともいえよう．

3.2. モデル

「システム」や「アーキテクチャ」の語が成熟し，取り立てて対象の要素と要素間の関係を明示的に区別しなくても誤解なく利用されているからであろうか，と，そんな解釈でもしたくなるほど，「モデル」が当業界に広く使用されるようになってきた．UML の普及によるのか，あるモデリングツールの称揚によるのか，「モデルベース」という表現もよく耳にする．

このモデルという語は，対象 What を表すのか，また方法 How を表すのかはっきりしない．モデルベースと称するツールが設計を支援するとすれば，それは How である．How で出来上がったものがモデルと

すれば，What である．

また数理の世界とソフトウェア工学の間では「モデル」の意味がむしろ逆とする指摘もある[35]．論理学や数学では，例えば公理から演繹された式は，「模範的な例」であるゆえ，「モデル」と呼ぶそうである．他方，ソフトウェア工学では事例を集めたもの，また集めてやや共通化し，模範となるものをモデルと呼ぶと指摘する．

前者が演繹的で，後者は一種帰納的であると対比できる．しかし，両者とも，模範的で模式的で，いずれも語「モデル」に含む意味を汲み取っているといえる．

さらに同書には，「モデル」という語が過去の長きに渡って使用されてきた息の長い語であるということが紹介されている．「これは「モデル」が他の用語と結合して使われてきたからであろう」としている．引用が長くなるが、同書には以下のようにある。

「，，，当然のことながら Software という語の出現頻度が他を引き離して多いが，それを除けば，つぎのような順序であった．---1．System，2．Design，3．Specification，4．Model，5．Analysis---

Model という語は第 4 位にランクされただけでなく，1975~94 年の 20 年，はやりすたりなく使われているという特徴をもつ．たとえば第 2 位にランクされた Design は，この 20 年のうちのとくに前半に出現頻度が高く後半は減少するが，Model にはそのような変動がほとんどみられない．これは「モデル」が他の用語と結合して使われてきたからであろう．結合する相手は，たとえば，life cycle， design，system，analysis，process ， product ， quality ， domain ， object ， computation，data，data flow， entity-relationship，function，application，state transition， . . . とおびただしくある．これらの結合相手にははやりすたりがあったが，モデルのほうは結合する相手を変えながら，一貫して人気を保ってきたのである．」[36].

「モデル」が結合しやすい語であるという特徴は，逆に定義される語ではなく，他を定義する基本的な(primitive)語であるということから来るのかもしれない．実際，意味の分析的議論を厳密に行う書に，「モデル」が未定義のままに使用されている[37]．同書に「分析のモデルの構成」という節があるが，同節にはモデルの定義や説明はない．モデルを構成要素たる分析的事項が 11 個記されているが，これら 11 個がモデルを指す．同書の著者には「モデル」が定義不要なプリミティブな語と認識されていたのであろう．

プリミティブな語は定義もされない（できない）語であるが，しかし説明は可能である．実際，同書[37]のモデルとされた分析的事項には説明がある．

「モデル」がプリミティブな語であるとすれば，「モデル」は他の語の説明や定義に使用され，当情報処理分野において様々な概念を作ってきたということもうなずける．

そして反面，他の語との親和性の高さは意味を曖昧にするというマイナス面もあろう．「モデル」はときに「バズワード」と非難される場合もある．しかし，説明すれば足りる．

3.3. オントロジー

新たな語の導入という観点からいえば，語"Ontology"が,組込みシステムの分野に登場したのは,システムの大規模・複雑化が課題となり始めたころと思われる．2005 年に Kaiya らの論文がある[26]．Kaiya らの論文では，組込みシステムを事例として取り上げていることから，「オントロジー」の組込みシステムへの適用と理解される．

Kaiya らは，要求仕様書に対して，オントロジーを適用することによって，要求仕様書から矛盾や不備を減らし，文書の品質を高めることを提案する．Kaiya らは，どの要求仕様書の文も，意味の原子的構成要素（atomic constituents of meaning）に基づいて解釈されるべきで，オントロジーはそのような知識を得るために使われる，とする．その結果，文書はどの関係者にも同じく解釈される原子的概念を持つものとされる．

Kaiya らのオントロジーは，要求仕様を満たす条件や観点からなるフレームワーク（枠組）ではないかと思われる．Kaiya らは次のように考える．

Kaiya らの原子的構成要素とは，「概念（concepts）」と「関係（relations）」である．要求分析は要求項目（items）に対して行うが，その際，概念と関係を（いわばフレームのように（筆者注））あてはめる．概念には，機能（function）や対象（object），環境（environment），制約（constraint），品質（quality）があり，環境の中には「actor」と「platform」がある．関係には，一般化（generalize），集合（aggregate），同義（synonym），異義（antonym），関係づけ（associate），矛盾（contradict），因果（cause），適用（apply），要求（require），支援（support）がある（上記は簡便さのために日本語では体言表現とした）．これらの原子的構

成要素の各々の視点から要求項目を検査する．すると，項目内容の不備や不足，項目どうしの関係が矛盾しているなどが見つかるという．

例えば，任意の項目が他の項目と関係して存在するものである場合，要求仕様書の中に片方の項目しかない場合には，関係して存在するという条件を満たさない．このような場合には，たとえば要求（require）という関係を働かせることにより，関連項目の不足をシステムは要求（require）することになる．

このような条件や観点の枠組，あるいは仕組みがオントロジーである．条件や観点が基本的なものゆえ，「存在」という基本的なことを表わし，かつこの中には推論規則も含まれるゆえ，これも基本的なことを表すといえる．よって，このような仕組みを，基本的な存在物，あるいは基本的な存在の方法（ありかた）であるとみなし，「オントロジー」と称するのであろう．

このようなオントロジカルな方法は何度も繰り返され，要求仕様書は正確になり，また詳細になる．これを精義化とでも呼ぼう．

ところで，Kaiya らは，このような精義化は，自然言語処理（NLP）を適用しなくても可能であるとしている．他方，先に取り上げた Web 意味論では，Gruber は先の文献[14]で，文「どの作家も読者に誤解されている」を，「その人が作家であるならば，どの人も，何人かの作家を誤解している読者を持つ」という風に述語命題に分解することを行っている．

以下が，文献[14]で，述語論理を適用し，自然言語の文を述語論理の構文で再構成する例である．

For example, the following is a KIF sentence that says, “All writers are misunderstood by some reader.”

```
(forall ?W
  (=> (writer ?W)
      (exists (?R ?D)
          (and (reader ?R)
               (document ?D)
               (writes ?W ?D)
               (reads ?R ?D)
            (not (understands ?R ?D))))
```

なお，上記 KIF の文は，文献[14]によれば，「Genesereth & Fikes, 1992」の考案した構文である．どの自然言語の文もこのような量を含んだ文に自動的に分解できるので，文どうしの導出が構文規則に基づき可能となる，といえる．

Gruber らは，自らの構文論（構文と推論規則），意味論（文の真偽の規則）を Web の領域に適用している．これらの構文論，意味論が Gruber らの「オントロジー」に他ならない．そして，これらの道具立てをもって，存在するものは表現可能なものとうたい，自分たちの世界（仕様）を Web 上に表現，すなわち存在たらしめている．

Kaiya らは，自然言語処理（NLP）を使用しないとしている．NLP は不使用としても，自然言語で書かれた要求仕様書を対象にしたい．要求仕様書の品質は上げる必要がある．自然言語の解析を行う要求仕様記述ツールも出ているが[41]，Kaiya らの要求仕様のための条件や観点を含むオントロジカル・フレームは要求項目の不足を補うなど，有効と思われる．今後が期待される．

語の用法に戻る．Feilmayr らは語「オントロジー」を用いているシステムや文献を詳しく調べ，「オントロジー」がこの2，30年，大いに称揚されてきたが，必ずしも成功していないといっている[16]．語の誤用や，またバズワードとしての「オントロジー」の使用もあるとも指摘している．ただし，これは Web やビジネスアプリを対象にしての調査である．

他方，オントロジーの適用領域を広げるべきとする，Pileggi らの最近（2018 年）の，いささか宣伝めく論文がある[27]．方法論の明示はないが，開発のライフサイクル，すなわち要求の開発から廃棄までの全プロセスに応じたオントロジーの方法の開発，適用，普及を促すものである．背景には，通信とモバイル技術の急速な発展に対して多くのソフトウェアを提供しなければならないが，その質が追い付いていないという危機感がある．「オントロジー」を広く検討する価値があろうかと思う．

4. 結び

「オントロジー」への，Pileggi らの呼びかけに対して思い起こすのは，「システム」である．「システム」は万能の概念として，コンピューター分野に適用される以前に，産業や組織，また個別科学などの多くの分野で期待されたという．何ごともシステム的にとらえよ，ということであったろうが，果たしてそれで十分であったかは推して知るべしである．1960 年前後のことのようだが，「システム」はいまや当たり前のことばとなって日常生活の中で用いられている．

「モデル」も「アーキテクチャ」も，どの分野でも使用

されている. しかし, 分野が変われば, 意味も異なる. 定義の難しい「モデル」などは, 分野ごとに指す意味が異なるともいえる.

しかし,「オントロジー」が「システム」のようにどの分野でもごく普通に使われるという光景はやや想像しがたい. Web 上のさまざまのサービスが, Gruber のいう存在するものは表現可能であり, したがってそのような光景が「オントロジー」, すなわち「存在のすがた」といわれてもなかなかピンと来ない.

しかし, 100 年後, あるいはもっと前かもしれないが, Web 上で, 自走車や, 自飛車(造語)などの存在物たちが, 自然界の営みのように, あるいは自然以上に十分制御されて, かつ自律的に動いていたとしたならば. そしてその光景に名をつけよ, とでもいわれたなら,「オントロジカル・ランドスケープ」, すなわち「存在論的景観」などと命名してしまうかもしれない.

空想めく話で恐縮だが, その言い訳に, 明治維新で招聘された, いわゆる雇われ外国人学者の話を引用させていただく. 30 年近く日本で医学を教えたベルツ氏は, 日本人の学問に取り組む姿勢に不満だったらしい. すぐ何に役に立つかと問いたがる態度に, 日本の学問は原理原則を大事にしないと映ったという[38].

こんにちでいえば成果を急ぐあまり基礎的な研究をしない, ということにでもなろうか. 構文論, 意味論, そもそも言語などが, 原理原則論になるのかは異議もあろう. そして中には, Gruber らの言うように, 存在するものは表現できるが, その逆, すなわち表現できるものは存在するのか, つまり表現により概念化(conceptualization)されたもの, すなわちそれは「概念(concept)」となると思うが, その概念は存在するのか, 存在するとすればどこにどう存在するのか, という問いも湧きそうである.

が, このような原理原則というべきか, さらにいっそう原理原則に戻るような議論が存在する一方で, 足下の, 例えば要求仕様書の問題に話が戻ると, 文も論証もよいが, 図などでもっと文書をわかりやすく, などという要望にも目を配らざるをえない.

するとまた, わかりやすさとは? また論理の厳密さとは? あるいは, どちらがソフトウェアの品質や生産効率に奏功するのか? しかも, システムなりソフトウェアなりのライフサイクルの中で, などと注文が出る. 議論が長く, ときに未整理のまま続きそうである. しかし, 前提となることがらを丹念にそろえ, 洗い出し, それこそ原理原則に立ち返って議論してみることが大事そうである.

いささか分を超えてしまったかもしれない. しかし, 当議論が多少面白い話題としていただけたら幸いである. 各位のご議論の美味しい肴たらん!

一品, 語の組合せパズルをつけさせていただいた.

表 語の組合せパズル:行→列と読む, △;可能?

	システム	アーキテクチャ	モデル	オントロジー
システム	○of	○	○	○
アーキテクチャ	△	△of	○	△of
モデル	○	△of	△of	△of
オントロジー	○	○	○	△of

参考文献

[1] Johnson, Lyle (1960). "A Description of Stretch" (PDF). p. 1. Retrieved 7 October 2017.

[2] P. Naur; B. Randell, eds. (1969). "Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7–11 Oct. 1968" (PDF). Brussels: NATO, Scientific Affairs Division. Retrieved 2012-11-16.

[3] Software Engineering Techniques: Report of a Conference Sponsored by the NATO Science Committee, B. Randell and J.N. Buxton, eds., Scientific Affairs Division, NATO, 1970, p. 12.

[4] P. Kruchten; H. Obbink; J. Stafford (2006). "The past, present and future of software architecture". IEEE Software. 23 (2): 22. doi:10.1109/MS.2006.59. S2CID 2082927.

[5] W.E. Royce and W. Royce, "Software Architecture: In-tegrating Process and Technology," TRW Quest, vol.14, no. 1, 1991, pp. 2–15.

[6] 石井 治, 最近のメモリー, 計測と制御, 1970, 9

巻, 8 号, p. 580-589, 公開日 2009/11/26, Online ISSN 1883-8170, Print ISSN 0453-4662, https://doi.org/10.11499/sicejl1962.9.580, https://www.jstage.jst.go.jp/article/sicejl1962/9/8/9_8_580/_article/-char/ja

[7] 特公昭 51-001381

[8] 情報処理 Vol. 12 No. 9 Sept. 1971 pp. ニュース pp.594

[9] 高橋延匡, 牛島和夫, 新田謙治郎, 淵一博, 土居範久, 山地克郎, 亀田尋夫 & 真子ユリ子. (1971). タイムシェアリング・システムの問題点をめぐって. 情報処理, 12(6).

[10] Rudolf Carnap, " Empiricism, Semantics, and Ontology" (in "The Logical syntax of Language A Study in Semantics amd Modal Logic"), The University of Chicago Press, 1947, pp.205-221.

[11] W. V. Quine, "On what there is" (in "From a logical point of view"), Harvard University Press, 1953, pp.1-19.

[12] Powers, David (1991). Preface: Goals, Issues and Directions in Machine Learning of Natural Language and Ontology. AAAI Spring Symposium on Machine Learning of Natural Language and Ontology. DFKI.

[13] Gruber, T. R. (1991). The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In J. A. Allen, R. Fikes, & E. Sandewall (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Cambridge, MA, pages 601-602, Morgan Kaufmann.

[14] 14>15 Gruber, T. R. (1993). "A translation approach to portable ontology specifications". Knowledge acquisition, 5(2), 199-220.

[15] Genesereth, M. R., & Nilsson, N. J. (1987). Logical Foundations of Artificial Intelligence. San Mateo, CA: Morgan Kaufmann Publishers.

[16] Feilmayr, Christina; Wöß, Wolfram (2016). "An analysis of ontologies and their success factors for application to business". Data & Knowledge Engineering. 101: 1–23. doi:10.1016/j.datak.2015.11.003.

[17] 元田浩, & 吉田健一. (1989). 定性推論と深い推論 (< 特集>「定性推論」). 人工知能, 4(5), 538-546.

[18] 元田浩, & 吉田健一. (1991). 定性推論の応用: 定性推論の知識獲得への応用. 情報処理, 32(2).

[19] 牧野武則. (1991). 語彙の概念と知識について. 情報処理学会研究報告自然言語処理 (NL), 1991(37 (1991-NL-083)), 105-112.

[20] 中島震, 小泉昌紀, & 松本正雄. (1993). 問題領域指向リエンジニアリング-GUI ソフトウエア. 情報処理学会研究報告情報システムと社会環境 (IS), 1993(4 (1992-IS-042)), 67-76.

[21] ティヘリノ A. ジュリ, 池田 満, 北橋 忠宏, 溝口 理一郎, タスクオントロジーと知識再利用に基づくエキスパートシステム構築方法論 : タスク解析インタビューシステム MULTIS の基本思想, 人工知能, 1993, 8 巻, 4 号, p. 476-487, 公開日 2020/09/29, Online ISSN 2435-8614, Print ISSN 2188-2266, https://doi.org/10.11517/jjsai.8.4_476, https://www.jstage.jst.go.jp/article/jjsai/8/4/8_476/_article/-char/ja

[22] Norbert Wiener (1948), Cybernetics, The technology Press, John Wiley & Sons, Inc. New York. Hermann et Cie, Paris 同書訳,『サイバネティックス』(池上止戈夫ら訳), 岩波書店 (1962)

[23] W. Ross Ashby (1956). An Introduction to Cybernetics, Chapman & Hall. 同書訳,『サイバネティクス入門』(篠崎武ら訳), 宇野書店(1967)

[24] Kenneth A. Fox, Marc P. Pasture, and Peter S. Showman (1972). A Human Interface for Automatic Measurement Systems, Hewlett-Packard Journal April 1972 Volume 23 Number 8

[25] Shane Dickey (1972). Distributed Computer Systems, Hewlett-Packard Journal November 1972 Volume 26 Number 3

[26] H. Kaiya and M. Saeki, "Ontology based requirements analysis: lightweight semantic processing approach," IEEE,Fifth International Conference on Quality Software (QSIC'05), 2005, pp. 223-230, doi: 10.1109/QSIC.2005.46.

[27] Pileggi, Salvatore F.; Lopez-Lorca, Antonio A; and Beydoun, Ghassan, "Ontology in Software Engineering" (2018). ACIS 2018 Proceedings. 92. https://aisel.aisnet.org/acis2018/92

[28] Basili, Victor R., and H. Dieter Rombach. Tailoring the software process to project goals and environments. 1986.
[29] Boehm, Barry W. "Seven basic principles of software engineering." Journal of Systems and Software 3.1 (1983): 3-24.
[30] Zelkowitz, Marvin V. "Resource utilization during software development." Journal of Systems and Software 8.4 (1988): 331-336.
[31] "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1". Object Management Group. Retrieved 9 April 2014.
[32] Rumbaugh, James, et al. Object-oriented modeling and design. Vol. 199. No. 1. Englewood Cliffs, NJ: Prentice-hall, 1991.
[33] 富士通, FACOM 230 28 解説, 1973
[34] 松田正一,『システム理論序説』, オーム社, 1971, p.8
[35] 玉井哲雄,『ソフトウェア工学の基礎』, 岩波書店, 2004, p. 47
[36] 玉井哲雄,『ソフトウェア工学の基礎』, 岩波書店, 2004, p.48-49
[37] 永井茂男,『分析哲学－言語分析の論理的基礎－』, 弘文堂, 1969, p.206
[38] E.ベルツ,『ベルツの日記』(上), 岩波文庫, 2020, pp.238-240
[39] https://ja.wikipedia.org/wiki/System/360
[40] 兼本茂, 安全性向上員会, 組込みシステム技術協会, 2022 April,『組込みシステムの安全解析と障害原因診断の統合アプローチ』, IPA, 2015
[41] N. Urushibara and C. Sasaki, "Integration of Two Kinds of Syntax for Requirements Description and Its Future Development," 2018 1st International Workshop on Easy Approach to Requirements Syntax (EARS), 2018, pp. 3-8, doi: 10.1109/EARS.2018.00007.

模範解答を用いたコンパイルエラー箇所指摘の高精度化

中井 亮佑
大分大学
v1858239@oita-u.ac.jp

紙名 哲生
大分大学
kamina@oita-u.ac.jp

要旨

C 言語は今でも多く用いられ，プログラミングの入門用の言語としてもよく使われる．しかし，プログラミングの初学者にとって，コンパイラが表示するエラーメッセージは理解しにくい．本論文では，既存のコンパイラが，ブロック構造の閉じ括弧 '}' を忘れた際に誤りの原因となっている行番号を正しく指摘することができない点を改善するために，模範解答と初学者のプログラムを比較し，その差異によってプログラム中のブロック構造の閉じ括弧 '}' 忘れとその位置を指摘する方法を提案する．具体的には，構文解析器が処理するソースコード中の構文要素をイベントとして抽象化し，模範解答と初学者のプログラムのイベント列を比較する．提案手法を評価するために，コードベースを収集し，それらのブロック構造の閉じ括弧 '}' を筆者が適当に削除し，そのコンパイルエラーとその位置を正確に検出できるかを確認した．その結果，収集したコードベースの約半数のコンパイルエラーとその位置を検出することができた．また，被験者を用いた実験も行い，イベント列の抽象化の妥当性を確認した．

1. はじめに

C 言語は，現代でも幅広く使用されている言語であり，プログラミング初学者が最初に学ぶ代表的な言語のひとつである．C 言語初学者がプログラムを作る上で必ず使うものの一つにコンパイラがある．コンパイラはプログラム中の様々な問題に対して，エラーメッセージを表示する．しかし，そのエラーメッセージは初学者にとっては理解が難しいことが多い．例えば，ブロック構造の閉じ括弧 '}' を忘れたとき，既存のコンパイラでは誤りの原因となっている行番号を正しく指摘することはできない．エラーメッセージの理解が難しいことで思うようにプログラムが書けず，初学者の学習意欲をそぐ可能性がある．

本研究では，コンパイルエラーの箇所を適切に表示することで，経験の浅い初学者のプログラミング学習を支援する．本研究で想定している「初学者」は，プログラミングを初めて学ぶ人である．例えば，大学など学校のプログラミングの授業を初めて受ける学生や，入門書を買ってこれからプログラミングを始める人などを想定している．

具体的な手法として，模範解答と初学者のプログラムを比較し，その差異によってプログラム中のブロック構造の閉じ括弧 '}' 忘れとその位置を指摘する方法を提案する．具体的には，構文解析器が処理するソースコード中の構文要素をイベントとして抽象化し，模範解答と初学者のプログラムのイベント列を比較する．

提案手法を評価するために，コードベースを収集し，それらのブロック構造の閉じ括弧 '}' を筆者が適当に削除し，そのコンパイルエラーとその位置を正確に検出できるかを確認した．その結果，収集したコードベースの約半数のコンパイルエラーとその位置を検出することができた．また，被験者を用いた実験も行い，イベント列の抽象化の妥当性を確認した．

本論文の構成は次のとおりである，まず 2 節で関連研究について紹介し，3 節で本研究の提案について詳細に述べる．4 では，本研究で作成したツールの評価を行いその結果を考察する，最後に 5 節で結論と今後の課題を述べる．

2. 関連研究

C 言語からプログラミングを始める初学者のために，コンパイルエラーを分かりやすく表示する静的解析ツールとして，C-Helper [1] がある．従来のコンパイラでは，コンパイルエラー時のメッセージが初学者にとって理解が難しいこと，解決策が書かれていないことなど，プログラミング初学者向けではない点があった．C-Helper では，初学者が陥りがちな間違いを検出し，分かりやすいメッセージを表示する．このツールにより検出できる問題は，文献 [1] によると，インデント乱れ・`char` 型変数への文字配列の代入・`printf()` のパラメタ不整合・`return` 文の不足・関数定義の余分なセミコロン・構造体宣言のセミコロン不足・動的に確保した配列に対する `sizeof` の使用があげられる．

しかし，C-Helper では，`for` 文，`while` 文，`if` 文などを記述する際にありがちな閉じ括弧 '}' 不足を検出することはできない．この閉じ括弧不足は，既存のコンパイラが正確にエラー箇所を指摘しにくい問題の一つである．これらの点で，本研究は C-Helper とは異なり，また，初学者の支援にもなる．

ソースコードを解析する方法として，コードクローン [2] がある．コードクローンとは，ある特定の字句などをクローンとし，ソースコードにそのクローンと類似したものが含まれているかを検出する手法である．具体的な手法としては，文字列レベルのクローンを作成する手法 (手間がかかり，大規模なプログラムの解析には向かないためツールはない)・プログラムの字句解析を用いて字句をクローンとし，クローン解析する手法・さらに粗く行ごとを，ハッシュ関数を用いて一定の長さの文字列に圧縮し，その列をクローンとしクローン発見問題を解く手法・プログラム中の文を，識別子などをパラメタ化したクローンとし，クローン発見を行う手法・関数や手続き，クラス定義全体をクローンとし，等価な要素対を見つけることでクローン発見問題を解く手法がある．

本研究は，3 節で述べるとおり，ソースコードを抽象化したうえで模範解答とユーザのプログラムを比較するという点において，様々なコードクローン検出手法と通ずるものがある．しかしながら，本研究で行う比較方法はエラー箇所検出に特化したものであり，クローンを検出するためのものではない．本研究の提案をコードクローン検出に用いるのが難しいのと同様に，既存のコードクローン検出手法を本研究の用途に用いるのもまた難しいと考えられる．

C 言語初学者向けのインテリジェントな指導システムとして，C-Tutor[3] がある．C-Tutor は，サンプルプログラムから，リバースエンジニアリングによりプログラマの意図をプログラム記述（program description）として抽出する．その後，知識ベースのプログラム解析を行い，入力されたソースコードを実行しながらプログラム記述中の満たされていない目標を調べ，その目標を実装するプランの集合を知識ベースに問い合わせるとともに，プランとソースコードの違いを検出し，初学者に報告する．

この手法は，プログラマの意図をサンプルプログラムとして与える点は本研究と同じであるが，テストケースを予め準備しておく必要があり手間がかかる．また，コンパイル・実行可能なソースコードが対象であり，構文エラーを対象としている本研究とは適用範囲が異なる．

静的解析を用いて初学者向けのプログラミング支援を行う手法として，初学者向けの Java の静的解析フレームワーク [4] がある．この手法は，主にロジック上の誤りの検出や品質のチェックを行うために用いられる．具体的な手法は，模範解答と初学者のプログラムを，正規化したうえで構造の類似性（ループ構造，代入やメソッド呼出の順序など）の比較を行う．また，ソフトウェアメトリクスの計測も行う．

しかしこの手法は，ギャップを埋める問題を前提としており，最初から自由にソースコードを書かせるタイプの問題に同様の類似性の解析が適用できるかは不明である．

開発されたプログラムがコーティング規約を満たすか確認するためのツールとして，CX-Checker[5] がある．CX-Checker は，まずソースコードを構文解析し XML で表現する．ユーザはそれをもとに，XPath や DOM 形式でルールとして与えられたコーティング規約をもちいてプログラム中の規約違反を検出する．

このツールは，構文エラーや初学者の陥りがちなエラーを検出するためのものではないが，ルールの書き方は汎用的なので，模範解答の意図をルールとして記述するといった使い方ができる可能性はある．しかし，閉じ括弧忘れのような初歩的なエラーまで検出できるかは不明である．また，出題意図はルールとして準備しなければならず，模範解答をそのまま用いることができない点で本研究とは異なっている．

3. 提案手法

3.1. 前提

本節では，初学者の陥りがちなコンパイルエラーのなかでも，for，while，if 文などのブロック構造の閉じ括弧 '}' 不足の位置を適切に検出する手法を提案する．この手法では，プログラミング初学者を対象に，C 言語のコンパイルエラーを検出するために，ユーザのプログラムと模範解答のプログラムを用いる．この手法を用いる状況としては，学校の講義や，模範解答が用意されているようなプログラミングの教本や問題集を解くような場面を想定している．

また，この手法は，プログラム中にある程度の深さのネストがある場合に閉じ括弧を書かなかった場合（あるいは編集中にうっかり消してしまった場合）など，構文エラーはあるものの構文解析器が「それっぽい」構文木を作ることができた場合を想定している．意味のある構文木がそもそも作られない場合もあるため，本手法は既存のコンパイラとの併用も前提としている．

既存の IDE の中には，閉じ括弧 '}' を自動補完するものもある．しかし，初学者が IDE を使用しない場面 (いきなり IDE を使わせない授業も根強く残っている) も想定される．また，編集中に閉じ括弧 '}' をうっかり消してしまう事態も想定されるが，既存の IDE ではそれを指摘することはできない．

3.2. アプローチの概要

本手法は，文献 [6] に記述された手法を拡張したものであるため，まずはそこで採用されているアプローチの概要を説明する．まず，ユーザのプログラムと模範解答のプログラムをそれぞれ構文解析する．そして，if，for，while の各抽象構文木 (AST) ノードを深さ優先で訪問して何らかの処理 (例えばエラーメッセージの収集) を行うプログラムのそれぞれのノードへの訪問 (visit) と退去 (leave) を記録したイベント列[1]をそれぞれ作成・比較し，そのイベント列の違いで閉じ括弧の不足を検出する．

イベント列の生成手順は次のとおりである．まず，プログラムの AST を visitor パターンで辿る．訪問したノードが if 文だった際には，if 文の開始であることと

[1]各ノードへの訪問や退去を，AST 探索中に遭遇する事象としてイベントと呼んでいる．以後，イベントの出現順序をプログラムの構文要素に対応させ，それぞれを「開始」「終了」と呼ぶ．

```
1  #include <stdio.h>
2  void main() {
3    int i, j, k;
4    for (i = 1; i <= 10; i++) {
5      for (j = 0; j < 10; j++) {
6        printf("%d", j);
7      }
8      if (i > 10) {
9        printf(",");
10     }
11   }
12   k = 0;
13   while (k < 10) {
14     printf("%d", k);
15     k++;
16   }
17 }
```

図 1. 模範解答のプログラム例

その if 文の出現順序を表すイベントを生成し，イベント列に追加する．そのノードから leave するときには if 文の終了を表すイベントを生成し，イベント列に追加する．for 文や while 文も同様にしてイベント列を作成する．イベントの種類は以下のとおりである．

- i (if 文の開始)
- iend (if 文の終了)
- f (for 文の開始)
- fend (for 文の終了)
- w (while 文の開始)
- wend (while 文の終了)

この操作をユーザと模範解答のプログラムで行い，それぞれのイベント列を生成する．

例として，図 1 の模範解答に対して，ユーザのプログラムが図 2 に示すものであった場合について説明する．ユーザのプログラムでは，4 行目から始まる for 文の閉じ括弧が不足している．gcc などの既存のコンパイラでは，これは main 関数の閉じ括弧が不足していると検出されるが，このアプローチでは以下のようにイベント列

```
1  #include <stdio.h>
2  void main() {
3    int i, j, k;
4    for (i = 1; i <= 10; i++) {
5      for (j = 0; j < 10; j++) {
6        printf("%d", j);
7      }
8      if (i > 10) {
9        printf(",");
10     }
11   k = 0;
12   while (k < 10) {
13     printf("%d", k);
14     k++;
15   }
16 }
```

図 2. ユーザのプログラム例

を比較することにより，12 行目の while 文の前に閉じ括弧が不足しているとして検出することができる．

模範解答	ユーザプログラム
f	f
f	f
fend	fend
i	i
iend	iend
fend	w <- '}' が不足している
...	...

以上が本論文で示す手法の基本的な方針であるが，この方法には次の問題点が存在する．

まず，この方法では閉じ括弧不足の位置を細かくは検出できない．上の例では本来閉じ括弧が不足しているのは while 文の前ではなく，11 行目の代入文の前である．しかし，図 2 のようにイベント間にイベントと関係のない文があった場合，本来イベントと関係のない文の前に閉じ括弧を置かなければいけない場合でも，イベントの直前（図 2 の場合は w の直前）としか指摘することができない．これでは，使用者は間違った位置に '}' を書いてしまい，コンパイルエラーは無くなるが求めていた結果が得られずに混乱させてしまう恐れがある．

二つ目の問題点として，同じ結果を得る for 文と while 文の比較ができない点があげられる．繰り返し処理である for と while は，どちらで行っても同様の結果を得ることができる．そのため，指定がない限りはプログラムを書く側がどちらを用いるかが分からない．模範解答のプログラムを用意しても，模範解答とユーザのプログラムで使用する繰り返し処理が違う場合がある．上で示した手法では，for 文や while 文の出現をイベントとして，イベント列を生成することはできる．しかし，模範解答とユーザのプログラムで使用している繰り返し処理の方法が異なる場合，それらを異なるイベント列として比較するので誤りであると検出してしまう．

3.3. イベント列

本研究では，上記の問題を解決するため，イベントの種類に以下のものを加える．

- return (return 文)
- s (宣言を含めた，return 文以外のブロック構造を持たない文)

これは，イベント列の中でブロック構造とは無関係の文を識別できるようにするためである．return を特別に区別したのは，ブロック構造と同様制御構造に関わる文であるからであるが，本論文ではこれ以上扱わない．

また，s は構文解析の際，高い頻度で出現するが，その出現回数には意味がなく，むしろ有害となる場合もあるため（例えば複数の変数を単一の文でまとめて宣言するか，別々の文を用いて宣言するかによって，同じプログラムでも異なるイベント列として検出される），s の出現は 1 回以上の繰り返しとし，正規表現の記法を用いて {s}+と表記する．

この拡張により図 1 と図 2 のイベント列を比較した結果は以下のとおりである．

```
1  #include <stdio.h>
2  void main() {
3    int i, j, k;
4    i = 1;
5    while (i <= 10) {
6      j = 0;
7      while (j < 10) {
8        printf("%d", j);
9        j++;
10     }
11     if (i > 10) {
12       printf(",");
13     }
14     i++;
15   }
16   k = 0;
17   while (k < 10) {
18     printf("%d", k);
19     k++;
20   }
21 }
```

図 3. ユーザのプログラム例（while 使用）

模範解答	ユーザプログラム
{s}+	{s}+
f	f
{s}+	{s}+
f	f
{s}+	{s}+
fend	fend
i	i
{s}+	{s}+
iend	iend
fend	{s}+ <- '}' が不足している
…	…

これより，閉じ括弧不足がブロック構造を持たない文の直前であることが分かり，図 2 の 11 行目より前に閉じ括弧が必要なことがわかる．

3.4. for 文と while 文の扱い

本手法では，模範解答とユーザのプログラムに使用する繰り返し文 for と while の違いがある場合でも比較することを可能にするために，イベント列の比較方法を変更し，fend と wend を比較する際に '}' 不足を指摘せず，さらに wend の前にのみ余分な {s}+ がある場合は無視するようにした．これは単純な方法ではあるが，ある程度は機能する．例えば，図 1 のプログラムに対し，そこで用いられている for 文を while 文に書き換えたプログラムを用意し（図 3），両者のイベント列を比較すると以下のようになる．

模範解答	ユーザプログラム
{s}+	{s}+
f	w
{s}+	{s}+
f	w
{s}+	{s}+
fend	wend
i	i
{s}+	{s}+
iend	iend
	{s}+
fend	wend
…	…

4. 評価

4.1. コードベースを用いた評価

本節では，コードベースを用いて，閉じ括弧不足があるプログラムの不足箇所を本提案手法がどれくらい正確に検出できるかについて評価を行った．コードベースは，AtCoder[2]内の公式解説と，大分大学の初年次プログラミング科目である「基礎プログラミング」の例題集より，筆者が例題を適当に 20 個ずつ選ぶことによって収集した．扱った例題については，main 以外の関数を使わず，且つ main 内にネストされたブロックがあるものを中心に収集した．なお，AtCoder は競技プログラミングサイトのひとつである．

評価方法は，まず，ソースコード内の '}' を第一著者が適当に[3]一つ削除し，本手法を実装したツールで解析してそのエラーを確認する．次に，表示された位置が正しい位置であるかどうかを確認する．評価項目は，エラー

[2] https://atcoder.jp/home

[3] コードベースの収集や '}' の削除にバイアスが入らないよう，できる限り何も考えずに行うことを意識して行ったという意味である．

表 1. 検出成功数

コードベース	検出数
AtCoder	8 (20 のうち)
基礎プログラミング	11 (20 のうち)

表 2. 既存のコンパイラとの比較

本研究\Visual Studio	正しい	正しくない
正しい	0	19
正しくない	8	13

```
1  #include <stdio.h>
2  int main() {
3    int s = 75;
4    if (s>=90) {
5      printf("あなたの成績はSです\n");
6    else if ((80<=s) && (s<90)) {//}削除
7      printf("あなたの成績はAです\n");
8    } else if ((70<=s) && (s<80)) {
9      printf("あなたの成績はBです\n");
10   } else {
11     printf("がんばりましょう\n");
12   }
13   return 0;
14 }
```

図 4. エラーを正しく検出できなかった例

を検出でき，且つ検出された箇所に閉じ括弧を挿入することによってプログラムが正しく動作した場合に「正しくエラーを検出できた」とする．

評価結果を表 1 に示す．AtCoder のソースコードに本ツールを用いた場合，40%はエラーを正しく検出できた．「基礎プログラミング」のソースコードに本ツールを用いた場合，55%はエラーを正しく検出できた．

ここで，エラーを正しく検出できなかったケースについて考える．図 4 は，模範解答から 6 行目の '}' を削除したプログラムである．これと模範解答のイベント列を比較すると以下のようになる．

```
模範解答    ユーザプログラム
{s}+        {s}+
i
{s}+
i
...
```

このように，ユーザのプログラムでは閉じ括弧がないために 4 行目の `if` 文に相当する AST ノードが作られる前に構文解析が失敗している．このように，構文解析が早期に失敗するようなケースでは本手法は役に立たないことがわかる．一方でこの事例は既存のコンパイラでは正しくエラーを検出してくれそうである．そこで，本ツールと既存のコンパイラを比較するために，評価で用いた閉じ括弧不足のあるユーザのソースコードを Visual Studio Community 2019[4]でコンパイルし，コンパイルエラーが正しく検出されるかどうかを確認した．評価基準は，閉じ括弧不足を指摘し行数を表示している，もしくは，閉じ括弧不足の次のトークンにエラーがあることを指摘しているなら正しく検出しているとし，閉じ括弧不足を指摘していないものは正しく検出していないものとする．

結果を表 2 に示す．本研究で正しく検出できたケース 19 件のうち，既存のコンパイラで正しく検出できたケースは 0 件であり，正しく検出できなかったものは 19 件であった．また，本研究で正しく検出できなかったケース 21 件のうち，既存のコンパイラで正しく検出できたケースは 8 件であり，正しく検出できなかったものは 13 件であった．これは，既存のコンパイラで正しく検出できなかった閉じ括弧不足の多くを本手法では正しく検出できており，逆に本手法で正しく検出できなかった閉じ括弧不足の一部は既存のコンパイラで正しく検出できたことを示している．つまり，構文木がある程度出来上がった段階で見つかるような構文エラーは既存のコンパイラは正しく検出できず，そのようなケースでは本手法が有効に働くこと，逆に既存のコンパイラでエラーが正しく見つかるようなケースでは本手法は役に立たないことをこの結果は示唆している．

4.2. 被験者実験

イベント列による抽象化の妥当性を確認するため，本研究では 1 名の被験者（情報系を専攻している大学学部 4 年生で，C 言語の経験有り）に協力してもらい，ある設問に対するあらかじめ用意した模範解答と，被験者の

[4] https://visualstudio.microsoft.com/ja/downloads/

[設問]
配列 { 1,3,5,5,7,9,11,13 } の最大値を出力するプログラムを C 言語で作ってください.
(main 文だけでお願いします)

図 5. 被験者実験に用いた設問

```
#include <stdio.h>
//模範解答
int main() {
  int max = 0;  //最大値
  int i = 0;
  int a[8] = { 1,3,5,5,7,9,11,13 };
  //最大値の仮定
  max = a[0];
  for (; i < 8; i++) {
    if (max < a[i]) {
      max = a[i];
    }
  }
  printf("最大値は%dです。\n", max);
  return 0;
}
```

図 6. 被験者実験で準備した模範解答

作成したプログラムに対して，イベント列生成・比較が正しく行えるかどうかについての実験を行った.

実験の手順は，まず被験者に図 5 に示す設問を解いてもらう．設問は，main 以外の関数を使わず，且つ main のブロック中にネストされたブロックができるようなものにした．次に，あらかじめ用意した模範解答と被験者のプログラムに対して，本ツールを使用する．図 6 に模範解答のプログラムを示す．それに対し，被験者は図 7 のようなプログラムを記述した．どちらも同じ実行結果になる．両者を，本ツールを用いてイベント列同士で比較した結果は以下のとおりである.

```
#include <stdio.h>

int main() {
  int a[8]={1, 3, 5, 5, 7, 9, 11, 13};
  int i, max;

  max = 0;
  for(i=0; i<8; i++) {
    if(a[i]>max){
      max = a[i];
    }
  }
  printf("最大値は%dです。\n", max);
  return 0;
}
```

図 7. 被験者の書いたプログラム（印刷の都合で一部改変）

模範解答	ユーザプログラム
{s}+	{s}+
f	f
{s}+	{s}+
i	i
{s}+	{s}+
iend	iend
fend	fend
{s}+	{s}+
return	return

模範解答とユーザのプログラムで，変数宣言の順序や変数 max の初期化方法などの違いはあるものの，イベント列として比較すると全く同じになり，適切な比較が行われていることが確認できる.

4.3. 議論

コードベースを用いた評価の結果から，既存のコンパイラで正しくエラー箇所を検出できないケースでも本ツールでは正しく検出できる場合が多く，逆に本ツールで正しくエラー箇所を検出できない場合でも既存のコンパイラだと比較的正しくできる場合があることが確認できた．とくに，両方のツールで正しく行えた場合は，今

回の実験では一つもなかった．これらのことから両者は相補的な関係にあり，場合に応じて適切に使い分けるのがよいと考えることができる．

なお，被験者実験については現在１名のみのデータしか集まっていない．本ツールは，大学初年次教育などの入門的なプログラミング教育で用いられることを想定しており，イベント列レベルで違いが出るような様々なバリエーションが存在する場面はそう多くないと著者らは考えているが，それでも結果の一般性については大きな疑問が残されている．特に、閉じ括弧 '}' 忘れ以外のエラー (余計な {s}+ブロックを置いてしまうなど) が入りうる環境においての，本手法の評価は課題の一つである．今後より多くの被験者を集めて結果の検証を重ねる必要がある．

5. 結論

本論文では，既存のコンパイラが，ブロック構造の閉じ括弧 '}' を忘れた際に誤りの原因となっている行番号を正しく指摘することができない点を改善するために，模範解答と初学者のプログラムを比較し，その差異によってプログラム中のブロック構造の閉じ括弧 '}' 忘れとその位置を指摘する方法を提案した．具体的な実現方法として，構文解析器が処理するソースコード中の構文要素をイベントとして抽象化し，模範解答と初学者のプログラムのイベント列を比較するツールを実装し，収集したコードベースによる評価と被験者実験による評価を行った．その結果，既存のコンパイラが正しくエラー箇所を検出できない場合において，本ツールが正しくエラー箇所を検出できることが多いことが確認できた．また１名の被験者実験においてはイベント列の抽象化の妥当性を確認できた．なお本ツールは閉じ括弧忘れに特化しているが，他の種類の誤りについても同様の方法で検出できるものがないか，今後検討する必要がある．

参考文献

[1] 内田 公太, 権藤 克彦, C 言語初学者向けツール C-Helper の現状と展望, 第 54 回プログラミング・シンポジウム, pp.153–160, 2013.

[2] 井上 克郎, 神谷 年洋, 楠本 真二, コードクローン検出法, コンピュータソフトウェア, 18(5), pp.47–54, 2001.

[3] J. S. Song, S. H. Hahn, K. Y. Tak, and J. H. Kim, An intelligent tutoring system for introductory C language course, *Computers & Education*, 28(2), pp.93–102, 1997.

[4] Nghi Truong, Paul Roe, and Peter Bancroft, Static analysis of students' Java programs, In *Proceedings of the Sixth Australasian Conference on Computing Education (ACE'04)*, pp.317—325, 2004.

[5] 大須賀 俊憲, 小林 隆志, 渥美 紀寿, 間瀬 順一, 山本 晋一郎, 鈴村 延保, 阿草 清滋, CX-Checker：柔軟にカスタマイズ可能な C 言語プログラムのコーディングチェッカ, 情報処理学会論文誌, 53(2), pp.590–600, 2012.

[6] 西村 渉, 模範解答を用いたコンパイルエラー出力法, 令和 2 年度大分大学理工学部卒業論文, 2021.

ソフトウェアレビュー研究結果の認知拡大と適用促進

安達 賢二
株式会社 HBA
adachi@hba. co. jp

中谷 一樹
TIS 株式会社
nakatani. kazuki@tis. co. jp

上田 裕之
株式会社 DTS インサイト
hiroyuki. ueda@dts-insight. co. jp

要旨

日本科学技術連盟ソフトウェア品質管理研究会(以降,“SQiP 研究会”とする)レビュー研究コースでは,当初から 30 本ほどの研究論文を発表している. しかし,議論を重ねて生み出した研究結果が,それを必要とする現場の実務者や管理者にあまり知られていない,活用されていないという現状がある. その打開を目指して方策を検討し,最初の一歩となる公募型ワークを実施して一定レベルの効果を獲得できた. 一方で今後の課題が明らかになった.

1. 取り組みの背景と現状の問題点

1.1. 取り組みの背景

ソフトウェア品質に関するオープンな研究会として,一般財団法人日本科学技術連盟が主催する SQiP 研究会がある. 2021 年度で第 37 年度を迎えた SQiP 研究会の中で,レビュー研究コースは,2010 年に新設された比較的新しいコースである. 毎年さまざまな企業のソフトウェア関連エンジニアや管理者などが参画し,1 年間で受講者が持つ共通の問題意識からチームを構築し,それぞれのチームが設定したテーマで研究を進める. 2022 年 2 月までにのべ 30 本の論文を世に送り出している.

しかし,蓄積されたレビュー研究結果が IT 関連業務に携わる実務者や管理者(以降,“ユーザー”とする)にあまり知られていない,活用されていないという現状がある.

1.2. 現状の問題点

蓄積されているレビュー研究結果がユーザーにあまり知られていない,活用されていない要因には以下のようなものが考えられる.

要因 1:研究成果物が認知されにくい

SQiP 研究会は,東京でのオンサイト開催であったため,主に首都圏や大都市圏の IT 関連メーカー,ベンダーのユーザーが参加し,大都市圏以外の地域や中小組織の参加者は稀である.

また,参加者(の所属組織)が費用を支払い取り組む研究会であるため,1 年間の研究成果発表会(毎年 2 月)は参加者と運営関係者向けに開催される. さらに,毎年の論文と発表スライドは SQiP 研究会 Web サイトに,論文は SQiP Library[1]に登録されるが,実際に活用しているのは SQiP に関わった者などを中心とした一部の組織や個人に限られると推察される.

要因 2:レビューの課題が特定できていない

先行事例[2]より,ユーザーの多くは現状のレビューに対して「有効な指摘が少ない」「ダラダラ時間が長い」「ダメだし大会」「いじめられる」のように自ら感じている問題(以降,“レビューの問題”とする)は所有しているが,必ずしもレビューで獲得したい成果に必要な改善事項(以降,“レビューの課題”とする)を特定し,認識しているわけではないと推察している.

仮に自ら感じているレビューの問題が解決できたとしても部分最適レベルに留まり,本質的な改善や継続した改善には至らない可能性が高い

要因 3:レビューの問題・課題への解決手段の引当てが難しい

SQiP 研究会 Web サイトには年度単位,コースごとに多くの論文・発表スライドが,また,SQiP Library には約 500 件の論文が登録されている. “レビュー”でキーワード検索すると 100 件を超える論文がヒットする. そのため,現状のレビューの問題・課題を解決する研究結果がどれなのか探し当てる,引き当てるのは容易ではない.

要因 4:研究結果はそのままの適用が難しい

研究会での議論は 1 回 5 時間程度,年 8 回と限られた機会と時間の中で実施するため,開発した手法や試行・実験等による効果確認が粗削りなことも多い. さらに,開発した手法が多くの段階を経る包括的な内容になっているケースもある. これらの状況から,実務におけるレビューの状況や制約事項によっては,そのままの適用が難しく,大胆なカスタマイズや現状レビュープロセスの大幅な組み換えが必要になる場合もある.

要因 1 は研究結果への認知を,要因 2～4 は研究結果の適用を阻む要因であるため,本論文で取り扱う課題は以下の 2 点とした.

RQ1：レビュー研究結果の認知拡大
要因 1 を踏まえ，どのようなアプローチでレビュー研究結果の認知を拡大するのがよいか.
RQ2：レビュー研究結果の実務適用の促進
要因 2～4 を踏まえ，どのようなアプローチでレビュー研究結果の実務適用を促進するのがよいか.

RQ1，2 の解決に向けた本論文の以降の構成は次の通りである. 2 章では，今回提起した問題への方策を明確にする. 3 章では，方策を実施可能なレベルまで具体化する. 4 章では，方策を具体化したワークを構築する. 5 章では，ワークの実施と結果を述べる. 6 章では，ワーク結果による方策の評価と抽出された今後の課題を述べる. 7 章で全体をまとめ，今後の活動について述べる.

2. 問題解決に向けた方策の明確化

2.1. ターゲットユーザーの特定

方策を展開するために使えるリソースには限りがある. 方策による効果を可能な限り高めるため，どのようなユーザーをターゲットにすると効果的なのかをあらかじめ特定する.

方策を実展開する際には，

認知 → 適用 → 効果獲得

の段階を経るのが理想的であると考えられる. まず対象者が有効なレビュー手法を認知する必要がある. 次に，現状のレビューに対する問題意識と改善したいという思いを持つことで，適用に至る. そして改善の効果が感じられないとすぐに元の実践方法に戻ってしまい，実質的に適用したことにならないため， 効果獲得段階が必要となる.

よってターゲットユーザーは，「現状のレビューに問題意識を持ち，改善したいと思っている，しかし，改善に効果が期待できるレビュー手法を把握できずにいる人」となる.

2.2. 方策の明確化

ターゲットユーザーにどのようなアプローチを行うと効果が得られるかを明確にするため，1.2 節で述べた要因ごとに方策を検討する. 方策には H1～H5 の ID を付ける.

2.2.1. 要因 1（研究成果物が認知されにくい）への方策

ターゲットユーザーに認知されるには，SQiP 研究会の枠を超えたパブリックな環境で，Web サイト掲載のような受身の情報展開ではなく，自ら働きかけて情報発信する能動的なアプローチが求められる（H1）.

ターゲットユーザーの目に触れやすい場としては，ソフトウェア品質関連イベント，自ら働きかけて情報発信する能動的なアプローチとしては一般公開型勉強会や関連分野のコミュニティ活動などが挙げられる.
また，単発の情報発信では認知を広げる効果があまり期待できない. よって継続的な情報発信が必要になる（H2）.

2.2.2. 要因 2（レビューの課題が特定できていない）への方策

要因 1 を解決したとしても，レビューの問題・課題が特定できていなければ，適用段階に進まないのは容易に想像できる.

そこで，先行事例[2]から多くの実務に存在する共通的なレビューの課題を抽出し，典型的な負の事象群（以降，“レビューの典型的負の事象群”とする）として特定する. レビューの典型的負の事象群を示すことで，ターゲットユーザーが普段感じているレビューの問題・課題との適合率を高め，解決手段の適用を後押しする（H3）.

2.2.3. 要因 3（レビューの問題・課題への解決手段の引当てが難しい）および要因 4（研究結果はそのままの適用が難しい）への方策

要因 1，2 を解決したとしても，多くの論文の中から，自らが持つレビューの問題・課題への解決手段がどの論文に含まれるのかを把握するのは容易ではない. 把握できたとしても，論文に記述された手法が実務レビューにそのまま活用できない場合もある.

そこで，レビューの典型的負の事象群への処方箋となる解決手段を研究結果の構成要素から抽出，再整理し，小さい個別手法にまとめ直して部品化する. それによって，実務適用および実践の容易性を高めることが期待できる（H4）.

しかし，解決手段を部品化したとしても，一度の実践では体得できず，何度も試行し，失敗を重ねながら体得していく必要がある場合も存在する. よって，ターゲットユーザー本人が解決手段を繰り返し実践し，体得できる継続的な取り組みの場が必要となる（H5）.

2.2.4. 方策のまとめ

以上の方策を表 1 にまとめる. ターゲットユーザーに対して H1～H5 の方策を具体化することが必要である.

表 1. 方策のまとめ

主なターゲットユーザー		
現状のレビューに問題意識を持ち, 改善したいと思っているが, 改善の効果が期待できるレビュー手法を把握できずにいる人		
ID	方策	目的
H1	ターゲットユーザーの目に触れやすい関連イベントや一般公開型勉強会など, 当研究会の枠を超えたパブリックな環境で自ら働きかけて情報発信する能動的なアプローチを採る.	認知拡大
H2	単発の情報発信では認知を広げる効果が期待できないので継続的な情報発信を行う.	認知拡大
H3	ターゲットユーザーが持つレビューの問題・課題にヒットしやすくするため, 先行事例[2]から「レビューの典型的な負の事象群」を特定する.	適用促進
H4	「レビューの典型的な負の事象群」を解決するために研究結果を選定し, その構成要素から部品化された解決手段を作成する.	適用促進
H5	必要な技法・アプローチを体得するための継続的な取り組みの場を設ける.	適用促進

3. 方策の具体化

3.1. 認知拡大への方策の具体化

H1 実現のためには, レビューに対する問題意識を持つターゲットユーザーが出没しそうな場を探したり, 作る必要がある. レビューはテストの一形態[3]でもあるので, ソフトウェア品質やレビュー, テスト関連のイベントや勉強会, コミュニティ活動等が一つの有力な候補となる. 具体的には, ソフトウェア品質シンポジウム, ソフトウェアテストシンポジウム(含む, ソフトウェアレビューシンポジウム), ソフトウェアシンポジウムなどが候補として挙げられる.

本研究では, 方策検討時にセッション公募を行っていたソフトウェアシンポジウム 2022 東京(JaSST2022 東京)の公募型セッションに応募し, 方策を実施することとした.

JaSST2022 東京は, 国内のテスト関連イベントの中で最も多くの参加者が集う場である. 受講者が同一時間帯に並列に並べられた複数のコンテンツから受講したいものを選択する形式である. よって, レビューに関するセッションを選択した受講者はレビューへの問題意識が高いターゲットユーザーである可能性が高い. そこで方策を具現化したコンテンツの体験を通じて, レビューの問題・課題の解決に取り組む意欲があるユーザーを特定する.

一方, H2 実現のためには, 一か月～数か月に 1 度程度の継続した実践に加え, 無償で参加できる等の気軽さも必要である. 今回は, Web 上の IT 勉強会支援プラットフォームサービスに「レビュー勉強会」グループを立ち上げ, 勉強会を運営することとした.

勉強会を継続的に実施することで, 参加者による口コミも広がり, さらに新たなターゲットユーザーが参加することを目指すこともできる.

3.2. 適用促進への方策の具体化

H3 の実現のために, レビューで発生しがちな典型的な負の事象の連鎖と循環を因果関係モデルとして図 1 に示す. 因果関係モデルとは, それぞれの要素間に存在する因果関係を矢印で結んだネットワーク状の構造図である.

図 1 のノードは, レビューにおける典型的な負の事象である. 左側にはレビューの過程として, 右側にはレビューの結果や成果を配置することで, レビュー時に発生しがちな負の事象連鎖と循環を表現した.

例えば「一度に大量の成果物をレビューする」ことで「後半は集中力が落ちて薄い確認」となり「欠陥見逃しが増える」という負の連鎖が存在する. また, 「あとから欠陥見逃しに起因する問題が発生」し、「手戻り負担増や費用負担からレビューへのモチベーションが低くなる」ことで以降も「一度に大量(頁数)の成果物を全員でレビューする」となってしまう負の事象も存在する.

H4 の実現のために, 図 1 の連鎖と循環を断ち切るには, レビューにおける典型的な負の事象に着目し, それを打開するための解決手段を割り付ける必要がある.

本研究では, 図 1 の最左部のノードの解決手段を明確にし, 図 2 のようにターゲットユーザーが持つレビューの問題・課題と解決手段とをつなげて把握できるようにした. なお, 図 2 の負の事象への方策において, 実施のためにさらなる具体化が必要である部分には下線を引いた.

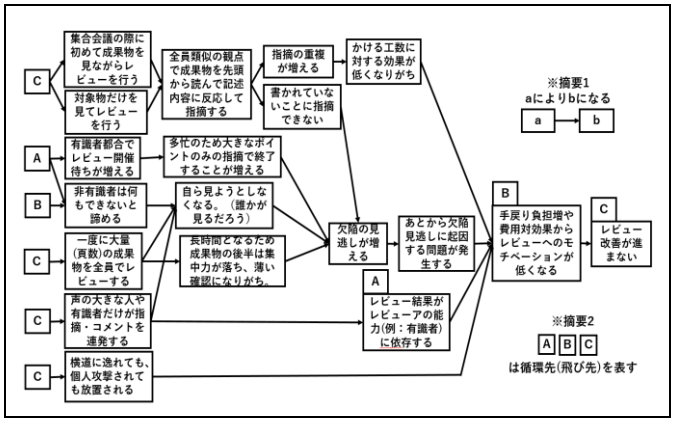


図 1.　レビューで発生しがちな負の事象の連鎖と循環

図 1 および図 2 において，左部のノードを解決することで，その先の負の連鎖が正の連鎖に反転し，レビューの結果や成果が高まることが期待できる．また，ユーザーが持つレビューの問題・課題と解決手段のヒット率を向上させ，適用を促進することも期待できる．

H5 の実現のために，自らのレビューの問題・課題の解決手段を効果的に体得するアプローチが不可欠になる．

ラーニングピラミッド[1]によると，資料を読む，レクチャーを受けるなどの受動的な学習は効果が薄い．グループで討議する，自ら体験する，人に教えるなどアウトプットを伴う積極的な学習が効果的である．よって，受講者が自らレビューの問題・課題解決を実践し，結果をアウトプットする「ワーク形式」で提供することとした．

ワークは一般的にオンサイトで提供されることが前提であるが，現状はコロナ禍であり，受講者の所在も分散しているためオンサイトでの実現は困難である．よって，オンライン会議システム zoom と Web オンラインホワイトボード miro をワーク運営インフラとして活用することにした．

これらツールの選定時には複数の類似サービス，ツールで比較検討した．ワーク環境構築やアクセスの容易さ，参加者数やワークスペースの広さなどの制約が少ないこと，全体像と個別詳細のシームレスな把握が可能であること，機能的な充実度，無料プラン有などの要件に加え，類似の勉強会で利用実績が多数あること，運営者側が利用可能であることを条件として評価し，選定した．

zoom と miro を組み合わせた運営により，受講者の所在に左右されることなくワークを実施することが可能になるため H1 や H2（認知拡大）を後押しすることが期待できる．また，ワークのアウトプットや録画が全員に共有可能となり，参加者同士の議論や運営者からのフィードバックが促進されるとともに，終了後の成果物が参照・再利用が可能であるため，H3～H5 を後押しすることが期待できる．

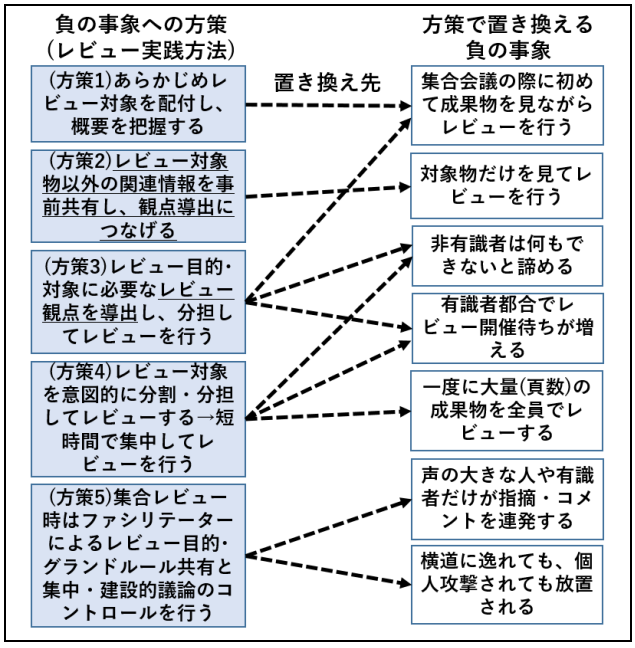


図 2．方策による負の事象への働きかけ

3.3. レビュー観点導出技法の実装（部品化）

本研究では、図 2 の下線部に示された方策記述を具体化するため，表 2 にレビュー観点導出技法として部品化した．

ここでレビュー観点とは，レビューアによるレビュー対象の見方を表している．つまりレビューで検出したい欠陥を見つけるためにレビューアが集中して着目する，対象成果物の側面を意味している．レビューアがどのように確認するのかを表したものである，と考えてもよい．レビュー観点は段階的に詳細化したレビュー目的にも相当し，「利用者課題の解決可能性」のように抽象度が大きい高位レベル観点から，「高齢者が一読で認識できる文字の大きさであること」のように具体的な低位レベル観点）まで，さまざまな粒度の観点がある．

表 2．構築したレビュー観点導出技法

ID	技法の概要
M1	作成者コンテキストからレビュー観点設定
M2	利害関係者の関心事からレビュー観点設定
M3	プロダクトリスクからレビュー観点設定
M4	対象成果物に求められる事項から確認方法設定
M5	対象成果物の典型的な欠陥から確認方法設定
M6	利用シナリオ＋状態遷移図に沿ってレビュー実施

レビュー観点を定めずに漠然とレビューを行うと，レビュー対象物の記述内容に場当たり的に反応することが中心となり，欠陥の見逃しが増える傾向が高まる．さらに複数のレビューアが同じ確認を実施し，指摘事項が重複するなど効率面にも悪影響を及ぼす．

先行事例[2]によると，レビュー実施前にレビュー観点を持つことでレビュー指摘件数や影響度の大きい指摘が増える等の効果が期待できる．

レビューアの能力により，理解し実践できるレビュー観点の種類や粒度は異なる．そのため本研究では，受講者が未経験のアプローチでレビューの効果を実感してもらうために，表 2 に示す M1～M6 に示すさまざまな観点の種類や粒度のバリエーションを取り揃えた．

M1 は､レビュー対象物以外の関連情報（作成者のコンテキスト）を事前に共有することで，対象成果物に作り込まれそうな欠陥を推測するレビュー観点である．M2 は，さまざまな利害関係者の立ち位置，関心事を想定するレビュー観点である．M3 はプロダクトリスクを検討するレビュー観点であり，M4 は対象成果物に求められる要求を検討するレビュー観点である．M5 は対象成果物に作り込まれそうな欠陥そのものを検討するレビュー観点であり，M6 は利用シナリオと状態遷移図を用いて検討するレビュー観点である．

M1，M6 は様々な粒度のレビュー観点を含む．M2 は高位レベルから中位レベルへ，M3 は高位・中位レベルから低位レベルへの落とし込みを行うレビュー観点である．M4，M5 は中位レベルから低位レベルへの落とし込みを行うレビュー観点である．

今回のワークでは特に，M6 としてシステムの正常系と例外系の利用シナリオおよびレビュー対象物の記述内容を忠実に表した画面遷移図を与え，利用シナリオに沿って画面遷移図をトレースするレビュー観点とした．

また，M1・M2 は図 2 の下線部で表されている負の事象への方策 2 を具体化したレビュー観点導出技法であり，M3～M6 は負の事業への方策 3 の下線部を具体化したレビュー観点導出技法である．

このようにレビュー観点導出技法を明確化することにより，それぞれのレビューにおいて質の高い確認を行うことができる．また，複数のレビューを包括的に行う際に，あらかじめレビュー目的に応じたレビュー観点導出技法を取り揃えておき，それぞれのレビューで必要なレビュー観点導出技法を選択するといった方法も容易になる．

さらに効果を高めるためには，当初の方策展開結果からさらなる観点導出技法の見直しやラインナップ充実，適切なパッケージ化を模索する必要があるだろう．

4. 方策実施に向けたワーク構築

4.1. ワークの目標

JaSST2022 東京での方策実施に向けて，図 2 に示した負の事象への方策(レビュー実践方法)をベースにしてワークを構築した．

ワークを主催し，運営する側を“運営者”，ワークを受講する側を”受講者“と呼ぶ．

JaSST2022 東京のセッション時間は 120 分間と限られている．そのため．多くのターゲットユーザーが持つレビューの問題・課題に適合し，解決し，効果を実感するために必要な枠組みを想定したうえで，典型的なレビューの問題・課題解決を部分体験できる内容に仕立てる必要がある．

限られた時間の中で，受講者が今後の継続的な活動グループに登録するためのインパクトを与える必要がある．今回のワークでは，当ワークの目標を以下のように定めた．

- ワーク実践により，受講者が以下のいずれかの状態になり，継続的な活動グループにできるだけ多く登録すること
- レビューの問題・課題を（一部でも）解決できること
- レビューの問題・課題が解決できそうだと感じること
- 知らない手法やアプローチが他にもいろいろあることを把握し，さらに知りたい，やってみたいと思うこと

4.2. チーム編成

実施するセッションには 20 名を超える参加者が集まる可能性があった．筆者の経験則として，ワークの効果を最大にするためには以下の 2 つの条件を両方満たす必要がある．

条件 1：ワーク中の受講者の遊びを最小限とする．
“遊び”とは，議論に加わらない，検討しない，実践しないなど思考停止し，ぼんやり過ごす状態になることを指す．

条件 2：他者との意見交換で自分とは異なる見解や考え方，実践方法に触れる機会が多い．

ワークの規模や複雑さにもよるが，条件 1 を満たすためには，チームとして人数的な余裕がない 2～3 名になることが理想である．しかし，2~3 名では条件 2 が不満足になりがちであるため，ワークを実施するチームの人数は 3～4 名が現実的かつ効果的である．

一方、チーム数が多くなると成果物が増え，運営者の状況把握とフィードバックの手間が増える．運営者 3 名が短時間でやり切るには限界があるため，今回は 1 チーム 4 名を基本とし，端数は 3 名で構成することとした．

4.3. ワークの枠組み

ワークの枠組みと全体像を表 3 に示す．運営者が実施する冒頭のイントロダクションと最後のまとめの間に受講者が実施するワークを 2 段階に配置した．各段階で受講者が普段実施しているレビューと図 2 に示す負の事象への方策（レビュー実践方法）の違いを味わい，問題・課題解決のヒントを持ち帰れるコンテンツとした．

ワーク 1 とワーク 2 を分けることにより，受講者が的確な指摘のために必要なことと，指摘しやすい環境づくりの効果を別々に認識することを目指した．ワーク 1, 2 を通じて，レビューのゴールは，各自が的確な指摘を出せるようになるだけではなく，チームとして心理的に健全な状態で有効な指摘を共有できるようになると理解することを目指した．

ワーク 1 は，事前のレビュー観点導出により狙いを定めて指摘事項を検出し，かつそれができることを実感することが重要である．そのためレビュー観点導出技法の中から，受講者が実践したことがなく，自らの現状を鑑みて「これをやってみたい」「可能性を感じる」と思う技法を選択し，実践する形式とした．レビュー対象を読みながら習慣に従ってぼんやり実施する普段通りのレビューに対する処方箋として「狙い撃ちレビューワーク」と名付けた．

表 3．ワークの全体像

No.	実施項目	時間
1	全体概説＆チームビルディング	15 分
2	ワーク 1：狙い撃ちレビューワーク 対象：レビュー観点設定～個別レビュー 　アプローチ選択肢の共有と選択 　それぞれのアプローチを実践 　ワーク 1 ふりかえり	50 分
3	ワーク 2：コントロールド集合レビューワーク 対象：集合レビュー 　ありがちな集合レビュー（実演 1） 　推奨する集合レビュー（実演 2） 　実演の考察：良いところ／よくないところ 　チーム共有とキャッチフレーズ化 　ワーク 2 ふりかえり	50 分
4	まとめ	5 分

一方ワーク 2 は，集合レビュー時の適切なファシリテーション実践により，建設的議論が進み，懸念点・リスクの掘り下げが促進され，心理的な障壁なしで結果を共有できるレビューのあり方，指摘しやすい場づくりの重要性を実感することが重要である．

そのため，まずワーク運営者が悪い事例となる集合レビューを実演することとした．これを実演 1 とする．実演 1 は，なんとなく始まり，個人攻撃や横道に逸れたまま戻らず時間を浪費，結論を押し付けるなどのありがちな悪い集合レビューの事例である．

次に，良い事例となる集合レビューを実演する． これを実演 2 とする．実演 2 はレビュー目的とグランドルールの共有からスタートし，集中した建設的な議論へのコントロールによる相乗効果の獲得など，推奨する集合レビューの事例である．

そして，受講者が 2 つの実演を観察，考察し，実務での活用に向けたキャッチフレーズ化を行う．

実演 1 は，進行役が不在もしくは機能せず，その場の流れに任せてただ進められている集合レビューを意図している．一方実演 2 は，実演 1 に対する処方箋を意図している．今回は実演 2 を「コントロールド集合レビューワーク」と名付けた．

4.4. ワーク環境の構築

JaSST2022 東京では，情報発信・相互対話用のコミュニケーションインフラとして zoom が使用された．

本ワークでは，受講者が 20 名を超える可能性があるため，1 チーム 4 名で構成する複数のチームがそれぞれワークを実践し，結果を共有，議論するために，Web オンラインホワイトボード miro にチーム単位のワーク環境を構築した．Miro の画面例を図 3 に示す．

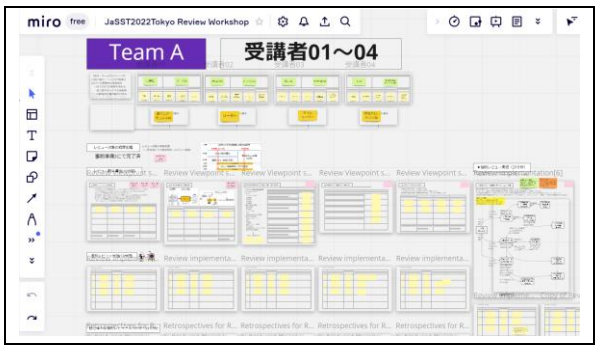


図 3．miro によるオンラインワーク環境
（チーム A メンバーボード・ワーク 1 環境）

また，ワーク環境とは別に，当日のワーク運営用スライド（技法・アプローチ解説を含む），解答例，集合レビュー事例スクリプト（よくあるダメなレビュー編・推奨アプローチ編）などを準備した．

5. ワーク実施結果

5.1. ワーク 1 の結果

当日は 23 名の受講者が集まり，表 3 に示す枠組みでワークを実施した．予定していた内容を一通り踏襲し，大きな問題もなく時間通り無事終了した．

A～F の 6 つの各チームメンバー（受講者）それぞれが，6 つの技法選択肢から普段は実践したことがない技法を一つ選択し，実践した．各チームの技法選択と個別レビューの結果を表 4 に示す．

表中の m-n は，全指摘数(m 件)とそのうちの重大指摘数(n 件)を意味している．なお，重大指摘とは，このレビューで見逃すとシステムリスクの顕在化につながる可能性がある指摘を指す．

すべてのチームで各メンバーが重複なく M1～M6 の技法のいずれかを選択したため，表中の指摘数はすべて個人レビューの結果を示している．

例えば，チーム A では，4 名のメンバーそれぞれがレビュー導出技法 M2，M3，M5，M6 を選択した．M2 を選択したメンバーは，「3-1」の結果，つまり全部で 3 件の指摘を検出し，そのうち 1 件が重大指摘であった．そしてチーム A の指摘件数計は「11-3」，つまり 4 名で総計 11 件の指摘を検出し，うち重大指摘が 3 件であった．

表 4 の M1～M5 については， 10 分間の短い時間で普段実践したことがない技法に取り組んだにもかかわらず，一人の例外を除き 1～4 件，チーム計では 7～15 件の指摘となった。

また，M6 の指摘件数が他に比べて多い傾向となった．これは，M1～M5 の個別レビュー実施時間が 10 分間であるのに対し，M6 は 20 分間としたためと思われる．

また，M6 は，他の技法と比べて重大指摘数が多い傾向となった．M6 は，レビュー対象記述に忠実に表現した状態遷移図に基づき，具体的な利用シナリオに沿って，つまり実際の入力・申請・承認（不承認時の対応を含む）・入金までをトレースしながら確認するものである．利用者の状況に応じた画面遷移や UI，機能が準備できているか，エラー発生時の対処不足はないか等を具体的に確認でき，また M1～M5 に比べ 2 倍のレビュー実施時間という優位性も相まって，特に重大な問題点を浮き彫りにする効果が高かったものと考察できる．

表 4．各チームのレビュー導出技法選択と指摘数
(数値記載がある箇所がチームで選択した技法)

チーム	人数	技法選択と指摘数 全指摘数－うち重大指摘数						指摘数計
		M1	M2	M3	M4	M5	M6	
A	4		3-1	3-0		2-0	3-2	11-3
B	4		3-0	2-1		3-1	4-0	12-2
C	4		1-0	1-1		1-1	8-2	11-4
D	4			1-0	1-1	2-0	7-2	11-3
E	4	1-0	0-0	4-2			2-1	7-3
F	3	4-1	3-1				8-3	15-5

チーム E で M2 を選択した受講者は「0-0」で全く指摘ができなかった．また，すべてのチームの概ね 2 割弱が，M1～M5 でのレビュー観点について，確認方法の具体化が不足したり意味が読み取れない状態になっていた．

受講後のコメントを調査すると，複数の受講者から「実践のための時間が短い」というコメントがあった．したがって，時間内に技法に対する十分な理解ができないままワークを進めざるを得なかった受講者がいたと考察できる．

一方，「短い時間だからこそ集中して実践でき，効果も実感しやすいと感じた．」という意見も複数挙がっていたことから，十分理解した上でワークを進めた受講者もいたと考察できる．

以上より，ワーク 1 は受講者が 2 割程度の理解不足や実践不足を抱えつつも，自らのレビューの問題・課題を一部解決できた，または解決できそうだと感じる体験ができたと結論づけられる．

5.2. ワーク 2 の結果

ワーク 2 は，集合レビュー時のファシリテータの相応しい振る舞いがレビューの効果を高めることを意図したワークであった．

受講者が記録した「良くない集合レビューやり方とその結果」と「良い集合レビューやり方とその結果」を見ると，ほとんどの受講者が集合レビュー時のファシリテーションでポイントとなる事項をほぼ漏れなく抽出していた．

そして各チームが作成したキャッチコピーは，チームとして最も重視した事項をわかりやすい言葉で示していた．

以上のことから，ワーク 2 ではほぼすべての受講者がワークの意図を理解し，集合レビューの問題・課題をファシリテーションで解決できそうだと実感できたと結論づけられる．

表 5. 当ワーク受講者による評価の平均点

理解度	実務有効性	受講満足度
3. 6 /5 点 (72. 2 /100 点)	4. 4 /5 点 (88. 9 /100 点)	4. 8 /5 点 (95. 3 /100 点)

5.3. 受講者によるワークの全体評価とワーク目標への到達度

受講者による受講終了直後の評価結果を表 5 に示す. 表 5 は受講者 23 名のうちワーク終了時の受講者評価結果 18 名分の平均点を示している. 5 名分は記入なしのため除外した. 評価は 5 点満点で行い、表 5 の下行では 100 点満点に換算している. 理解度, 実務有効性, 受講満足度のいずれも 7 割を超える高い得点が得られた.

まず, 理解度

- 多くのアプローチが認識できた
- 作成者のコンテキストを把握してテスト観点に反映する必要性を理解できました.
- しっかり理解したい
- ファシリスキルを磨く.
- スキルを身につけていきたい

理解度の点数は 7 割を超えていることからも, ターゲットユーザーが今後の取り組みを行うよい動機づけの機会になったと考察できる.

一方、他の指標に比べると理解度は相対的に低い. 受講後のコメントを調査すると, ワーク 1 について以下のようなコメントが得られていた.

- ワーク 1 は時間が足りなかったが, 中身をもっと理解したいと思いました.
- ワーク 1 の時間, あと 10 分はほしかった.
- レビュー技法は使い慣れるまでが大変そうだなぁという印象.
- ワーク 1 では時間が短いながらもそれぞれの手法の特徴を把握することができました.

したがってワーク 1 は, 時間が限られている中で初めて実践する内容だったため, 時間内にすべてを完了できなかったこと, そして一部理解不足のまま終わったことが, 理解度が相対的に低くなった主な要因と考察できる.

ワーク 2 については, 以下のようなコメントが得られていた.

- ワーク 2 は「集合レビュー」について改めて進行役の大事さを実感しました.
- ファシリテータが重要であることは理解できましたが, ファシリテータをどう育てるかが課題だと思いました.

したがってワーク 2 では, ファシリテータによる場のコントロールの重要性は理解できたものの, どうすれば身につけられるのかが課題であると認識したことが, 理解度が相対的に低くなった主な要因と考察できる.

また, ワーク 1, 2 について受講後のコメントを調査すると, 以下のようなコメントが得られていた.

- 今日学んだことを自分のプロジェクトに取り入れる.
- レビューの効果を出したい.
- レビューするのが気楽になった.
- あっという間のワークでした.
- ファシリテーションの極意についてはすぐ生かせそうな部分があるなと思いました.
- あんなにしっかりしたファシリテータができるか分からないけど, 実践してみたい.

これは, 実務への有効性と受講満足度は高い評価を得たことからも, ワーク 1, 2 を通じて受講者が持つレビューの問題・課題に適合し, 現実的な解になりそうだと期待できるものであったと考察できる.

したがって本ワークから, 受講者が自ら考えて結果を出力し, チームでそれぞれの結果と体験を共有する形式は学習効果が高く, 方策として適切であったと結論づけることができる. すなわち, 表 1 における適用促進の目的が達成できたと結論づけられる.

また, 受講者 23 名全員が受講後にオンラインの勉強会グループに登録した. すなわち, 所在がバラバラな受講者であったにも関わらず, 表 1 における認知拡大の目的が達成できたと結論づけられる.

6. ワーク結果による方策の評価と課題

6.1. RQ1(ターゲットユーザーはレビュー研究結果を認知できたか)の評価

JaSST2022 東京で実施したワーク結果から, 当論文で提起した課題 RQ1, RQ2 に対する評価を行い, 今後の継続的な取り組みの課題を特定する.

まず RQ1(ターゲットユーザーはレビュー研究結果を認知できたか)を評価する.

認知拡大の方策と今回の実践内容を表 6 に示す.

表 6. 認知拡大のための方策と今回の実践内容

認知拡大	
ID	方策
H1	ターゲットユーザーの目に触れやすい関連イベントや一般公開型勉強会など，当研究会の枠を超えたパブリックな環境で自ら働きかけて情報発信する能動的なアプローチを探る.
H2	単発の情報発信では認知を広げる効果が期待できないので継続的な情報発信を行う.
今回の実践内容	
H1:レビューに対する問題意識を持つターゲットユーザーが出没しそうな場でワークを実施する(今回は，テスト関連イベント JaSST2022 東京を選択). H2:その後オンライン勉強会を継続開催する.	

H1 についてはワーク受講者数で，H2 については勉強会グループへの登録者数で評価を行う.

本研究では H1 について，JaSST2022 東京の公募セッションに応募し，ワーク参加者 23 名が認知できた. H2 については，今回のワーク終了時に提示したオンラインの勉強会グループへの登録呼びかけに対して， 27 名が自ら登録した. ワーク参加者よりも多いのは，参加者からの口コミによるものであった.

したがって本論文では，本ワークのアプローチでレビュー研究結果の認知が拡大することを実証でき，RQ1 に肯定的な結論を得ることができた.

6.2. RQ2(レビュー研究結果の実務適用を促進できたか)の評価

次に RQ2(レビュー研究結果の実務適用を促進できたか)を評価する.

適用促進の方策と今回の実践内容を表 7 に示す.

H3, H4, H5 については， 受講直後の理解度， 実務有効性，受講満足度および勉強会グループへの登録者数で評価を行う.

本研究ではワーク参加者の理解度が 72. 2 点，実務有効性が 88. 9 点、受講満足度が 95.3 点となり，オンラインの研究グループに 27 名が登録した.

したがって本論文では，本ワークのアプローチでレビュー研究結果の実務適用が促進することを実証でき，RQ2 に肯定的な結論を得ることができた. 受講後のコメントからも，時間的な制約により部分的な体験となったものの，ターゲットユーザーは自分たちの持つレビューの問題や課題が一部解決したという実感が得られ，解決できそうだという期待が芽生えたことと評価できる.

表 7. 適用促進のための方策と今回の実践内容

適用促進	
ID	方策
H3	ターゲットユーザーが持つレビューの問題・課題にヒットしやすくするため，先行事例から「レビューの典型的な負の事象群」を特定する.
H4	「レビューの典型的な負の事象群」を解決するために研究結果を選定し，その構成要素から部品化された解決手段を作成する.
H5	必要な技法・アプローチを体得するための継続的な取り組みの場を設ける.
今回の実践内容	
H3:よくあるレビューの状態と想定される結果から相応しいレビュー実践方法(図 2)を導出する. H4:複数のレビュー観点導出技法を部品化(表 2)し，図 2 に連携する. H5:ワーク形式(手法を実践し，アウトプットを出力する形式)で，レビューの課題を解決する手段に継続的に取り組む.	

6.3. 方策の課題

6.3.1. ワーク 1 から得られた課題

最後に，ワークにより明確になった課題を整理する.

レビュー観点導出技法は，一度ワークで実践しただけですぐにノウハウが身につくわけではないものも多い. そして今回は技法実践が短時間であったため，十分な理解と実践ができなかったと想定される.

よって今後の継続した技法実践では，十分な時間を確保したうえで，実践，結果に対するフィードバックを通じて最終的に体得する取り組みが必要である.

また，表 2 の M6 内容を利用シナリオベースの技法と，状態遷移図ベースの技法の 2 つに分離する方がより適切な部品化となる等，作成済みのレビュー観点導出技法のブラッシュアップが必要である.

6.3.2. ワーク 2 から得られた課題

今回のワークでは，受講者が観察，考察するファシリテーションは，受講者が自ら実施したものではなく，運営者が実演するものであった.

そのため，今後は受講者が自らファシリテーションを実践し，体得することが必要である.

6.3.3. 全体を通じて得られた課題

今回のワークのために準備した技法やアプローチは，全 30 件の研究論文のうち 7 件からのみプロトタイプ的に作成したものにすぎない．

一方，他の論文には，レビュー計画，レビューア育成，レビュー結果の蓄積と再利用など，レビューを構成する様々な側面についてのノウハウが組み込まれている．これらのノウハウを抽出，整理して，レビューを体系的に学ぶプログラムとするのが今後の課題である．

また，今回のワークは継続して取り組む必要がある方策の第一歩目に過ぎず，今後継続することで欲しい結果が得られるのかについては未確認である．よって今後も取り組みを継続し，効果を評価していく必要がある．

7. まとめ

レビュー研究結果の認知が広がらない，活用されないという問題を解決するための方策を検討し，最初の一歩となる公募型ワークを実施した． ワークの結果では一定レベルの効果を獲得できたが，方策の課題も明らかになった．

今後は，今回特定された課題を一つひとつクリアしながら勉強会の継続開催と，今後開催される関連イベントでのワーク実施を併用しながら， レビュー研究結果の認知を拡大し，適用を促進していく予定である．

参考文献

[1] SQiP ソフトウェア品質ライブラリ, https://www.juse.jp/sqip/library/

[2] レビュー目的・観点設定の効果と課題，http://www.jasst.jp/symposium/jasst16tokyo/pdf/A2-1．pdf

[3] テスト技術者資格制度 Foundation Level シラバス Version 2018V3.1.J03，http://jstqb.jp/dl/JSTQB-SyllabusFoundation_Version2018V31.J03.pdf

[4] ラーニングピラミッド，https://ja.wikipedia.org/wiki/ラーニングピラミッド

ソースコードコメントに着目した不確かさとソフトウェア品質の関係調査

渡邉 紘矢
京都工芸繊維大学
工芸科学部 情報工学課程
h-watanabe@se.is.kit.ac.jp

崔 恩瀞
京都工芸繊維大学
情報工学・人間科学系
echoi@kit.ac.jp

水野 修
京都工芸繊維大学
情報工学・人間科学系
o-mizuno@kit.ac.jp

要旨

ソフトウェア開発中には開発者の知識不足により，既存プロジェクトのソースコードを十分に理解出来ないことがある．不確かさによるソースコードの複雑化は，ソフトウェアの品質や保守性に影響を与えている可能性があるが，この実証研究はまだ行われていない．そこで，本研究では，ソースコードコメントに存在する不確かさに着目し，不確かさによってソフトウェア品質にどのような影響があるかを調査した．具体的には，まず，ソースコードや設計に問題がある可能性を示す指標であるコードスメルと，ソースコードの品質メトリクスとして利用される循環的複雑度とクラス結合度を用いて，不確かさを表すキーワードをコメントに持つソースコードは品質が低いか調査した．調査の結果，不確かさはコードスメルの数への影響は見られないが，不確かさの影響を受けやすいコードスメルがあることがわかった．また，不確かさがあることで，*C/C++*，*Python* では循環的複雑度が高く，*C++*，*Python* ではクラス結合度が高くなることがわかった．

1. 緒言

ソフトウェア開発中には「不確かさ」と呼ばれる問題がある．不確かさの例としては，開発者の知識不足により，既存プロジェクトのソースコードを十分に理解出来ない，バグの修正方法など実装が正しいか分からないといった問題が挙げられる．機械学習などを用いたクラス分類器を例に挙げると，100%の正解率を達成することが難しく，誤った出力をする可能性があり，出力に不確かさが存在している．このような不確かさは，開発者にとって扱いにくいものであるため，開発者は直面した問題に対して一時的な措置をとる場合がある．この一時的な措置や不確かさが存在することによって，バグの混入や，ソースコードの複雑化を招く可能性がある．そのため，ソフトウェア工学において，不確かさを包容したソフトウェア開発は重要な研究課題の 1 つとして注目されている．

Git で管理されたプロジェクトを対象に不確かさを調査するツールを利用し，コミットメッセージに着目して不確かさを調査した研究がある [1–3]．このように，ソフトウェア開発において開発者が直面する不確かさについて実証研究があるが，その数は少ない．不確かさによってソースコードが複雑化することで，ソフトウェアの品質や保守性に影響を与えている可能性があるが，この実証研究はまだ行われていない．

そこで，本研究では，ソースコードコメントに着目し，不確かさによってソフトウェア品質にどのような影響があるかを調査する．具体的には，まず，ソースコードや設計に問題がある可能性を示す指標であるコードスメルに注目し，不確かさを表すキーワードをコメントに持つソースコードは多くのコードスメルが含まれているか調査する．次に，ソースコードの品質メトリクスとして利用される循環的複雑度とクラス結合度を用いて，不確かさを表すキーワードをコメントに持つソースコードは品質が低いか調査する．調査の結果，不確かさはコードスメルの数への影響は見られないが，不確かさの影響を受けやすいコードスメルがあることがわかった．また，不確かさがあることで，C/C++，Python では循環的複雑度が高く，C++，Python ではクラス結合度が高くなることがわかった．

2. 背景

2.1. コードスメル

技術的負債とは，ソフトウェア開発プロセスにおいて発生した問題を先延ばしにすることで，将来的に必要になるリファクタリング（ソフトウェアの外部から見た振る舞いを変えずに，内部構造を整理する作業 [4]）や修正のためのコストのことである．例えば，短い開発期間でリリースする必要があることから，発生した問題を一時的に修正するパッチを適用することで，適切な対応をとることを先延ばしにすることが挙げられる．これは，コードスメルの導入やバグの危険性の問題があるため，これらを検出し，解消していく必要がある [5].

コードスメルはコードや設計に問題がある可能性を示す指標として Fowler によって提案され，コードスメルが存在するソースコードはリファクタリングの実施が推奨されている [4]．数多く定義されているコードスメルの一例に過ぎないが，コードスメルは God Class，Feature Envy がよく知られている [4, 6]．God Class は他のクラスのことを知りすぎて巨大化したクラスであり，Feature Envy は他のクラスのデータを頻繁に参照していることを示している．これらは，巨大化したクラスを複数の小さなクラスへ分割することや，適切な他のクラスへメソッドを移動させることで，クラス間の依存関係を減らすリファクタリングを実施する必要がある．コードスメルを検出しリファクタリングすることで，ソフトウェアの品質と保守性の向上が期待できる．

ソフトウェア内に含まれるコードスメルを自動的に検査するツールの 1 つとして，SonarQube[1]がある．SonarQube は，事前に定義したルールベースのアルゴリズムに従ってコードスメルを検出するツールであり，ルールはプラグインとして追加できる機能を持っている [7].

2.2. 不確かさ

近年，不確かさを抱擁したソフトウェア開発は，ソフトウェア工学において重要な研究課題として注目されている [8]．不確かさの例としては，開発者の知識不足により，既存プロジェクトのソースコードを十分に理解出来ない，バグの修正方法など実装が正しいか分からないといった問題や，開発の初期段階に発生しやすい曖昧な要求，定まっていない設計など曖昧さによる問題が挙げられる．このような不確かさは，開発者にとって扱いにくいものであるため，開発者は直面した問題に対して一時的な措置をとる場合がある．この一時的な措置や不確かさが存在することにより，バグの混入や，ソースコードの複雑化を招く可能性がある．しかし，初めから不確かさのないソフトウェア開発を進めることは多くの労力と時間を必要とするため難しい.

ソフトウェア開発における不確かさは，Known Knowns，Known Unknowns，Unknown Unknown の 3 つのタイプに分けられる [9]．Known Knowns は不確かさが存在しない開発である．Known Unknowns は不確かさのある問題が存在する開発であり，開発者がその問題を認識している状態である．Known Unknowns における不確かさの問題は，Issue Tracking System やソースコード，仕様書内にメモとして記載し管理されている．Unknown Unknowns は不確かさのある問題が存在するが，開発者は何が不確かであるかを認識出来ていない状態である．

また，Prez-Palacin らは，不確かさを場所，レベル，性質の 3 つの観点で分類している [10]．場所は，どの部分に不確かさが現れているかを示し，モデルに含まれる部分と抽象化されている部分の境界を指すコンテキスト，そのモデル自体の構造，モデルに入力されるパラメータの 3 つに分類される．レベルは，不確かさを 5 段階で分類している．1 段階目は不確かさが欠落している状態 (Known Knowns に相当)，2 段階目は開発者は何らかの知識が欠落しているが，知識が欠落していることを認識している状態 (Known Unknowns に相当) である．3 段階目は開発者は知識が欠けていることを認識できていない状態 (Unknown Unknowns に相当)，4 段階目は自分が認識出来ていないことを認識するためのプロセスが欠落している状態，そして，5 段階目はメタな不確かさである．性質は，認識的と偶発的に分類される．認識的は，得られたデータが不十分である，データに対する理解度の欠落による不確かさ，偶発的は対象が持つランダム性による不確かさを表す．この分類における 3 段階目以上のレベルの不確かさについて，どのような問題があるか調査することが難しいため，既存の不確かさの研究 [1–3] では Known Unknowns のみを扱っている．

ソフトウェア開発において開発者が直面する不確かさについて実証研究があるが，その数は少ない．不確かさは，開発者にとって扱いにくいものであるため，不確かさを抱擁したソフトウェア開発を支援するツールを開発

[1] https://www.sonarqube.org/

する必要がある．実際のソフトウェアに含まれる不確かさを明らかにすることで，影響が大きく優先して解消する必要がある不確かさを指摘でき，この知見がツールの開発に役立つと考えられる．例えば，[1] ではコミットメッセージだけでなく，実際のソースコードや変更履歴と合わせた検証が必要であるとしている．また，コミットメッセージではなく，ソースコードコメントに着目した不確かさの調査はまだ行われていない．

2.3. 品質メトリクス

品質メトリクスとして，McCabe による循環的複雑度 [11]，Chidamber らによるクラス結合度が提案されている [12]．循環的複雑度はプログラムの実行経路の数を示している．例えば，if-else，switch-case などの条件分岐，for などの繰り返し文などが多く含まれることで，実行経路が増えるため，より複雑度が高い値を示すようになる．複雑度が高くなると，ソースコードのテストパターンの数が増えるため，テストコードの用意にかかる時間が増え，その保守性も低下することになる．

クラス結合度とは，他のクラスからメソッドが呼び出されている数，参照されたインスタンス変数の数を示している．つまり，クラス結合度は他のクラスやメソッドへの依存度の高さを示していると考えることができる．1 つのクラスやメソッドに対する仕様変更や修正が，複数のクラスやメソッドに影響するため，仕様変更や修正が困難になる．また，そのようなソースコードは理解も困難である．これらのメトリクスは，ソースコード品質へ与える影響を調査する目的で，広く利用されている [13]．

3. 調査概要

3.1. 調査目的

本研究の目的は，不確かさがあることで，ソフトウェアの品質にどのような影響が表れているか明らかにすることである．そこで，本研究ではソースコードコメントに着目したアプローチをとる．不確かさを表すキーワードを含むソースコードコメントが記述されているクラスやメソッドに着目し，不確かさがソフトウェア品質へ与える影響について調査した．

本研究の目的を達成するため，2 つの研究設問を設け，調査する．

RQ1 不確かさを表すキーワードをコメントに持つソースコードは多くのコードスメルが含まれているか？

RQ2 不確かさを表すキーワードをコメントに持つソースコードは品質が低いか？

3.2. 調査対象のデータセット

本論文で調査対象としてコード片とソースコードコメントの対のデータセットである，Source Code Analysis Dataset（以降 SCAD）を選択した [14]．このデータセットは，ソースコードコメントの予測や自然言語を用いたソースコードコメントの生成などの研究への活用を目的に作成された．

このデータセットがどのように作成されたのかを説明する．まず GitHub[2]から，再配布可能なライセンスを持ち，10 個以上のスターを持つ 108,568 個のオープンソースソフトウェア (OSS) プロジェクトを取得する．取得の対象としているプログラミング言語は，Java，C 言語，C++，Python である．取得したソースコードから，Doxygen[3]を使用してコード片とソースコードコメントの対を抽出する．コード片は，クラス，メソッド，関数，変数宣言の粒度で抽出される．Doxygen は 106,304 個（108,568 個中）の OSS プロジェクトで正常に実行され，合計 16,115,540 件のコード片とソースコードコメントの対が得られた．この対応関係がデータセットとして，提供されている．

4. RQ1 不確かさを表すキーワードをコメントに持つソースコードは多くのコードスメルが含まれているか？

4.1. 動機

不確かさを持つソースコードには，不確かさを持たないソースコードよりもコードスメルが多い可能性がある．プログラミング言語はそれぞれ，深層学習や Web 開発，アプリ開発といった得意な開発分野を持ち，習慣的な設計方法やプログラムの書き方が異なる場合がある．そのため，プログラミング言語によって，不確かさの影響は異なると考えられる．不確かさを持つソースコードとコードスメルの関係が明らかになることで，開発者が

[2]https://github.com/
[3]https://www.doxygen.nl/index.html

感じる不確かさは，リファクタリングの実施，設計の見直しが必要であることの指標になると考えられる．

4.2. 方法

RQ1 に答えるために，不確かさを表すキーワードをコメントに持つソースコードのコードスメルの数と不確かさを表すキーワードをコメントに持たないソースコードのコードスメルの数を比較した．調査方法は，以下の手順で進めた．この手順を図 1 に示す．

手順 1-1 データセットを不確かさを含むもの，含まないものに分類する．

手順 1-2 プログラミング言語ごとに適切なサンプルサイズでデータをサンプルする．

手順 1-3 SonarQube を使ってコードスメルを検出する。

手順 1-4 マンホイットニーの U 検定で有意差検定する。

手順 1-1 では，3.2 節で述べた SCAD の全てのソースコードコメントを不確かさを持つものと持たないものの 2 クラスに分類する．不確かさがあるという定義は，本論文では，ソースコードコメントに不確かさを表すキーワードを持ち，かつ少なくとも 1 個の特徴単語を含む，とする．本論文で利用する不確かさを表すキーワード，およびそれに対応する特徴単語を表 1 に示す．これは，村本らの研究 [1] で，頻出度が高い 20 個の不確かさを表すキーワードとして報告されたものである．

手順 1-2 では，SCAD に含まれているデータをプログラミング言語ごとに適切なサンプルサイズでサンプルする．適切なサンプルサイズは，信頼水準 95%，許容誤差 5%としてサンプルサイズを計算した．この時，取り出すのは不確かさを表すコメントに対する関数やクラスなどのスニペットだけでなく，関数やクラスが含まれるファイル全体を取得する．手順 1-2 でファイル全体を取得することにしたのは，コードスメルの検出に SonarQube を利用しているためである．一般的な SonarQube の利用方法は，プロジェクト全体やファイル群を入力とする．そのため，関数やクラスなどのコード片を SonarQube に入力した場合，構文解析に失敗する場合がある．構文解析に失敗した場合，SonarQube ではコードスメルの検出が出来ない．これを避けるため，今回はファイル全体を取り出すことにした．

手順 1-3 では，2.1 節で述べた SonarQube を用いてコードスメルを検出する．SonarQube がコードスメルの検出に用いるルールは SonarQube Community Edition Version 9.0 で予め導入されているものを利用する．ただし，Java に対する検査では，予め導入されているルールに加え，Antipatterns-CodeSmell プラグイン[4]によるルールを追加している．これは既存のツールで検出できるコードスメルより，クラスごと，クラスメソッドごとなど検出粒度に分けて調査するために，既存のルールでは十分なコードスメルを検出出来ないと考えられているためである．また，このルールのうち，ルールにつけられた重要度が高いもののみ検出する．具体的には，Blocker，Critical，Major の 3 つの重要度のルールを検出し，Minor，Info の 2 つの重要度の低いルールは検出しない．これは，本質的ではないコードスメルが過剰に検出されてしまうことを避けるためである．

一般的に，コードスメルの数はファイル行数が多いほど多くなる．このようなファイル行数の影響を考慮するために，ファイル 1 行あたりのコードスメルの数を求め比較することにする．

手順 1-4 では不確かさを持つクラス群と，不確かさ持たないクラス群との間で，コードスメルの数の差に対してマン・ホイットニーの U 検定の片側検定を行う．また，有意差以外の指標としてノンパラメトリックな効果量測定法である Cliff の $|\delta|$ を利用する [15]．Cliff の $|\delta|$ では，測定された差の大きさを推定することが出来る．Cliff の $|\delta|$ が示す効果量の大きさは，$0.147 < |\delta|$ で無視できる，$0.147 \leq |\delta| < 0.330$ で小，$0.330 \leq |\delta| < 0.474$ で中，$0.474 \leq |\delta|$ で大となる．

4.3. 結果

手順 1-1 で不確かさを表すキーワードとその特徴単語を用いて，不確かさを持つデータと不確かさを持たないデータに分けた．この結果，各キーワードに対して一致したデータ数を表 2 に示す． may が 106,680 個，might が 20,968 個，unknown が 6,565 個検出され，その合計が 145,807 個となる．不確かさを含むデータとして分類されたデータの合計は 155,560 個であることから，3 つのキーワードが占める割合は 93.7%となる．

SCAD 全体とプログラミング言語ごとに対して，コードスメルの数の統計量を表 3，表 4 に示す．また，マン・

[4]https://github.com/davidetaibi/sonarqube-anti-patterns-code-smells

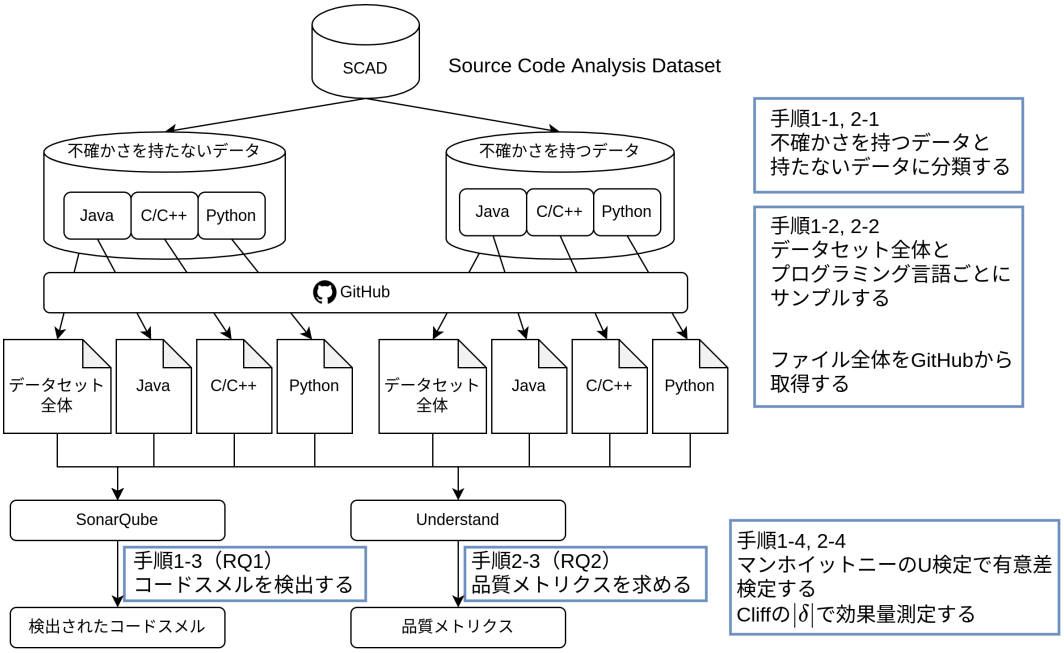


図 1. 調査方法の概要図

表 1. 不確かさを表すキーワードおよびそれに対応する特徴単語 [1]

不確かさを表すキーワード	特徴単語
ambiguous	call, valu, type, case, renam, avoid, name, differ, one, warn, error
arcane	trigger, damag, stack, build, make, take, shot, entri, seem, empti, column, will, error, problem
changeable	field, set, cach, default, make, offset, disk, select, pass, check, addit, start, found, error
debatable	case, system, getput, seem, name, place, want, think, realli
dubious	valu, code, bit, case, replac, get, seem, use, remov, avoid, warn, error
doubtful	see, realli, one, work, want, look
erratic	enabl, issu, behavior, result, logic, workaround, handl, appli, arm, caus, affect, part, bug
fuzzy	option, translat, way
irregular	break, switch, pass, valu, part, usag, approach, place, help, condit, work
may	case, mayb, caus, need, want, sinc, contain, time, differ, happen, mean
might	case, tri, get, want, need, caus, differ, lock, happen
obscure	code, case, problem, result, need, get, call, caus, bug, error
probably	get, need, want, caus, work, doesnt, way
risky	secur, case, data, run, potenti, reduc, dont, avoid, need
tentative	method, implement, problem, call, tentat, support, definit, work, like, allow, way, done, bug
unclear	valu, seem, name, reason, user, differ, like
unknown	type, messag, return, fail, case, tri, ignor, handl, reason, name, report, caus, will, error, warn
unreliable	test, case, detect, run, tri, set, remov, check, chang, time, dont, result, need
unsure	case, seem, work, dont, need, think, result
vague	return, line, function, set, get, attempt, document, time, think, caus, error

ホイットニーの U 検定と効果量として Cliff の $|\delta|$ の結果を表 5 に示す.

検定結果より，データセット全体や，どのプログラミング言語においても有意差はなく，その効果量も無視できる程度であった.

コードスメルの検出ルールについて，頻出度の高い上位 10 件を比較したところ，Java は 7 件，C/C++は 10 件全て，Python は 8 件一致していた．不確かさを持つかどうかによらず，出現頻度の高い検出ルールの項目は似た傾向にあることがわかる．つまり，不確かさを表すキー

表 2. 調査対象のデータセットに含まれるソースコードコメントのうち不確かさを表すキーワードを含むソースコードコメントの数

不確かさを表すキーワード	ソースコードコメントの数
ambiguous	1,375
arcane	48
changeable	146
debatable	25
dubious	6
doubtful	38
erratic	12
fuzzy	247
irregular	202
may	106,680
might	20,968
obscure	156
probably	6,565
risky	65
tentative	83
unclear	157
unknown	18,159
unreliable	381
unsure	143
vague	104
合計	155,560

表 3. 不確かさを持つソースコードに対する 1 行あたりに含まれるコードスメルの数の統計量

	SCAD 全体	Java	C/C++	Python
平均	0.048	0.072	0.029	0.032
第 1 四分位数	0.013	0.036	0.005	0.010
中央値	0.035	0.059	0.012	0.020
第 3 四分位数	0.064	0.088	0.028	0.038

表 4. 不確かさを持たないソースコードに対する 1 行あたりに含まれるコードスメルの数の統計量

	SCAD 全体	Java	C/C++	Python
平均	0.049	0.079	0.041	0.034
第 1 四分位数	0.013	0.035	0.006	0.013
中央値	0.030	0.063	0.013	0.024
第 3 四分位数	0.062	0.097	0.034	0.041

ワードをコメントに持つソースコードのコードスメルの内容と不確かさを表すキーワードをコメントに持たないソースコードのコードスメルの内容の間に，差異は無いことが考えられる．頻出度上位 10 件のうち，片方のみ表れている検出ルールの一覧を以下に示す．C/C++においては，片方のみ表れている検出ルールは無かった．

表 5. 1 行あたりに含まれるコードスメルの数に対するマン・ホイットニーの U 検定の結果と Cliff の $|\delta|$ による効果量

対象データ	有意差	効果量 $\|\delta\|$
SCAD 全体		0.088
Java		0.037
C/C++		0.049
Python		0.088

Java

(1) パッケージの宣言はソースファイルのディレクトリと一致させるべきである

(2) コードの一部をコメントアウトするべきではない

(3) インクリメント (++) およびデクリメント (--) 演算子は，メソッド呼び出しの中で使用したり，式の中で他の演算子と混ぜて使用するべきではない

(4) 例外ハンドラは元の例外を維持するべきである

Python

(1) SystemExit を再度 raise すべきである

(2) 制御フロー文「if」「for」「while」「switch」「try」は深くネストさせるべきではない

RQ1 に対する回答を以下にまとめる．

> 不確かさを表すキーワードをコメントに持つ場合と，持たない場合のコードスメルの数は同程度である．

5. RQ2 不確かさを表すキーワードをコメントに持つソースコードは品質が低いか？

5.1. 動機

不確かさによるソフトウェア品質への影響を明らかにすることで，事前にある一定のソフトウェア品質を確保するためには，どの程度の不確かさを許容できるのか見積もることができる．

5.2. 方法

RQ2 に答えるために，不確かさを表すキーワードをコメントに持つソースコードの品質メトリクスの値と，不確かさを表すキーワードをコメントに持たないソースコードの品質メトリクスの値を比較した．調査方法は，以下の手順で進めた．この手順を図 1 に示す．

手順 2-1 データセットを不確かさを含むもの，含まないものに分類する．

手順 2-2 プログラミング言語ごとに適切なサンプルサイズでデータをサンプルする．

手順 2-3 品質メトリクスを求める．

手順 2-4 マンホイットニーの U 検定で有意差検定する．

手順 2-1~ 手順 2-2 は，RQ1 の手順 1-1，手順 1-2 と同様であるため，本節では省略する．手順 2-2 でのサンプルサイズは，信頼水準 95%，許容誤差 5%として計算した．

手順 2-3 では，2.3 節で述べた循環的複雑度，クラス結合度の 2 つの品質メトリクスを求める．品質メトリクスを求めるため，ソースコード解析ツールである Understand 6.0[5]を利用した．手順 2-2 では，ファイル全体を取得することから，これらの品質メトリクスはファイル全体に対して計算される．また，循環的複雑度，クラス結合度のメトリクスは，ファイル行数が多いほど，循環的複雑度，クラス結合度が高い値を示しやすくなる．このようなファイル行数の影響を考慮するために，ファイルの 1 行あたりの品質メトリクスの値を求め，比較することにする．

循環的複雑度は，メソッド単位で計算されるめ，そのファイル内に含まれる全てメソッドの循環的複雑度の合計を求める．この合計値を，ファイル行数で割ったものを 1 行あたりの循環的複雑度とする．クラス結合度は，クラスから内部クラス，匿名クラスを分離し，それぞれに対してクラス結合度を求める．また，ファイル内にあるインターフェースや Enum クラスに対しても求められるため，それら全ての値の合計を求める．この合計値をファイル行数で割ったものを 1 行あたりのクラス結合度とする．循環的複雑度，クラス結合度の値はどちらも非負であり，非有界である．値は大きくなるほどより複雑であり，より結合していることを示す．

[5]http://www.techmatrix.co.jp/product/understand/index.html

表 6. 不確かさを持つソースコードに対する 1 行あたりの循環的複雑度の統計量

	SCAD 全体	Java	C/C++	Python
平均	0.201	0.191	0.127	0.245
第 1 四分位数	0.151	0.158	0.000	0.202
中央値	0.212	0.200	0.143	0.251
第 3 四分位数	0.264	0.234	0.195	0.292

表 7. 不確かさを持たないソースコードに対する 1 行あたりの循環的複雑度の統計量

	SCAD 全体	Java	C/C++	Python
平均	0.174	0.187	0.116	0.232
第 1 四分位数	0.096	0.149	0.000	0.177
中央値	0.178	0.190	0.083	0.231
第 3 四分位数	0.239	0.228	0.181	0.285

表 8. 不確かさを持つソースコードに対する 1 行あたりのクラス結合度の統計量

	SCAD 全体	Java	C++	Python
平均	0.062	0.084	0.075	0.042
第 1 四分位数	0.024	0.036	0.023	0.019
中央値	0.045	0.067	0.050	0.036
第 3 四分位数	0.734	0.110	0.094	0.055

手順 2-4 では不確かさを含むクラス群と，不確かさを含まないクラス群との間で，品質メトリクスの値の差に対してマン・ホイットニーの U 検定の片側検定を行う．また，RQ1 同様 Cliff の $|\delta|$ を利用する [15].

5.3. 結果

SCAD 全体やプログラミング言語ごとの，循環的複雑度の統計量を表 6，表 7 に示す．クラス結合度の統計量を表 8，表 9 に示す．また，マン・ホイットニーの U 検定と Cliff の $|\delta|$ の結果を表 10，表 11 に示す．

チェックマークは有意差 ($p < 0.05$) を示しており，効果量は Cliff の $|\delta|$ の値を記載している．循環的複雑度は，SCAD 全体，C/C++，Python のみ有意差があり，その効果量として C/C++は無視できる程度，Python は小程度であった．クラス結合度は，C++，Python のみ有意差があり，その効果量として C/C++は小程度，Python は無視できる程度であった．Java に対するクラス結合度は有

表 9. 不確かさを持たないソースコードに対する 1 行あたりのクラス結合度の統計量

	SCAD 全体	Java	C++	Python
平均	0.083	0.115	0.065	0.036
第 1 四分位数	0.017	0.048	0.000	0.011
中央値	0.049	0.087	0.027	0.027
第 3 四分位数	0.107	0.149	0.089	0.048

表 10. 1 行あたりの循環的複雑度に対するマン・ホイットニーの U 検定の結果と Cliff の $|\delta|$ による効果量

対象データ	有意差	効果量 $\|\delta\|$
SCAD 全体	✓	0.190
Java		0.041
C/C++	✓	0.140
Python	✓	0.190

表 11. 1 行あたりのクラス結合度に対するマン・ホイットニーの U 検定の結果と Cliff の $|\delta|$ による効果量

対象データ	有意差	効果量 $\|\delta\|$
SCAD 全体		0.047
Java		0.192
C++	✓	0.154
Python	✓	0.128

意差が出ていないが，その効果量は小程度となっている．これは，Java の場合は不確かさを持つソースコードの方が，クラス結合度は高くなることを示している．追加の調査として，Java を対象に不確かさを持たないソースコードの方がクラス結合度は高くなるか，マン・ホイットニーの U 検定を片側検定で行った．この片側検定では有意差がある結果となり，効果量測定より効果量は小程度であることを示した．RQ2 に対する回答を以下にまとめる．

不確かさを表すキーワードをコメントに持つ方が，C/C++，Python に対して循環的複雑度は高く，C++，Python に対してクラス結合度はより高くなる．Java に対するクラス結合度では，不確かさを表すキーワードをコメントに持たない方がクラス結合度は高くなる．

6. 議論

6.1. RQ1 不確かさを表すキーワードをコメントに持つソースコードは多くのコードスメルが含まれているか？

不確かさを持つソースコードのコードスメルの数と不確かさを持たないソースコードのコードスメルの数をファイル単位で比較した．その結果，全てのプログラミング言語において有意差は見られなかった．コードスメルは，ソースコードの設計上の問題を示す指標である．つまり，ソフトウェア開発において不確かさがあったとしても，ソースコードの設計上の問題への影響は無いことが考えられる．開発中に感じる不確かさは，リファクタリングが必要であること以外に，その不確かさの原因となる他の要素があると考えられる．本論文では，ファイル単位で調査したため，関数やメソッド単位での調査に比べると粒度が大きく，影響が表れにくくなっている可能性がある．そのため，今後は関数やメソッド単位のより細かい粒度で調査し，その影響について明らかにする必要があると考えられる．

不確かさを持つ場合，もしくは不確かさを持たない場合の片方のみ表れている検出ルールがあった．Java の (1) の検出ルールは，コードスメルを解析する際に，ファイルを配置したパスやそのソースファイルのファイル名を変更したため，表れたルールである．そのため，重要なルールではないと考えられる．Java の (2)，(3) に示す検出ルール，Python の (2) に示す検出ルールは可読性へ悪影響を示唆している．Java の (4) に示す検出ルール，Python の (1) に示すルールは例外を無視するなど，エラーハンドリングにおける適切な対応が行われていないことを示している．これらは，開発者がコードを理解しようとする妨げになる可能性がある．そのため，不確かさの原因となるコードスメルである，または，不確かさによる影響を示しているコードスメルであることを示唆

していると考えられる．

6.2. RQ2 不確かさを表すキーワードをコメントに持つソースコードは品質が低いか？

C/C++，Python における循環的複雑度，C++，Python におけるクラス結合度では，ともに有意差が見られ，Java では循環的複雑度，クラス結合度ともに有意差が見られなかった．これは，不確かさによる悪影響を受けるかどうかはプログラミング言語ごとに異なるが，少なくとも C/C++と Python においてソースコードの品質に悪影響を与えている可能性があることを示している．また，ソフトウェア開発において良い品質を保つために，不確かさを解消する必要性があることを実証的に示せたと考えられる．

しかし，初めから一切不確かさを含まないように開発を進めることは難しく，ある程度の不確かさが含まれることを許容する必要がある．そのためには，どの程度の不確かさが含まれていた場合に，品質がどの程度悪くなるか関係を調べる必要があると考えられる．不確かさの程度と品質の関係が明らかになることで，予めソフトウェアの品質を見積もることが可能になるからである．

Java のクラス結合度について，不確かさを持たない方がクラス結合度は有意に高くなる結果が得られた．これは，不確かさによる品質への悪影響として予想していた結果と逆の結果である．また，この結果は，Java を利用したソフトウェア開発において，不確かさを解消した設計はクラス結合度が高くなってしまう可能性があると考えられる．例えば，用いられている設計モデルやフレームワークといったプロジェクトの開発背景による影響があると考えられる．

7. 妥当性への脅威

不確かさの識別には，不確かさを表すキーワードや，そのキーワードと同時に現れる特徴単語を利用し，正規表現を用いてソースコードコメントと照合することで，機械的に不確かさを識別した．この識別方法では，先行研究によって，不確かさを表すキーワードには不確かさの特徴的な傾向がみられ，特徴単語は不確かさの内容を表していると推測出来ることが報告されている．しかし，不確かさを表すキーワードがソースコードコメントに含まれていたとしても，実際に不確かさがあるとは限らない．同様に，不確かさを表すキーワードを含んでいなかったとしても，不確かさがある可能性がある．このような，偽陽性や偽陰性の存在による影響がある．その影響については，特定したソースコードコメントに対して，目視分類する必要がある．

また，不確かさを表すキーワードや特徴単語は，共にコミットメッセージ内に書かれたものを対象として報告されたものである．今回はソースコードコメントを対象に適用することで調査した．コミットメッセージやソースコードコメントは，どちらも自然言語を用いて記述されているため，その傾向に差は無いと考えられる．

8. 関連研究

Famelis らは，不確かさの存在を表現するモデルとして，Partial Model を利用したアプローチを提案している [16]．深町らは，この Partial Model を利用した提案に基づいて，Java プログラミング環境を提案している [17]．また，Esfahani らは，システム開発の初期段階で，不確かさがある状況下でシステムアーキテクチャを設計するためのフレームワークを提案している [18]．不確かさの影響を実証的に調査して得られた知見は，不確かさを管理し，ソフトウェア開発を支援するツールの開発に役立つと考えられる．不確かさに関する実証研究のため，Git で管理されたプロジェクトを対象に不確かさを調査するためのツールが提案されている [19]．このツールを利用し，コミットメッセージに着目して不確かさを調査した研究がある [1–3]．本研究では，ソースコードコメントに着目し，不確かさとソフトウェア品質の関係について調査した．

9. 結言

本研究では，ソースコードコメントに着目し，不確かさがあることで，ソフトウェア品質にどのような影響が表れているか調査した．調査対象は Java，C 言語，C++，Python の 4 つのプログラミング言語である．

調査の結果，4 つのプログラミング言語に対しては，不確かさを持つソースコードのコードスメルの数と不確かさを持たないソースコードのコードスメルの数には有意な差は無かった．品質メトリクスに対しては，不確かさがあることで，C/C++，Python では循環的複雑度が有意に高く，C++，Python ではクラス結合度が有意に高く

なった．一方，Java では不確かさを持たない方が，クラス結合度は有意に高くなる結果となり，予想していた結果とは逆の結果が得られた．

プログラミング言語によって不確かさの影響が異なることが明らかとなったが，プログラミング言語のどの要素が影響しているかは明らかに出来ていない．どのようなプログラム，どのような機能を利用すれば良いかについては，言語仕様の観点から調査し，明らかにする必要がある．

参考文献

[1] 村本大起，江　冠達，村岡北斗，深町拓也，鵜林尚靖，亀井靖高，佐藤亮介，“ソフトウェア開発における不確かさに着目した OSS コミットログ解析,” ソフトウェアエンジニアリングシンポジウム 2017 論文集，vol.2017，pp.122–129，2017.

[2] 村岡北斗，鵜林尚靖，亀井靖高，佐藤亮介，“Revert に着目した不確かさに関する実証的分析,” ソフトウェアエンジニアリングシンポジウム 2019 論文集，vol.2019，pp.31–40，2019.

[3] N. Ubayashi, Y. Kamei, and R. Sato, “When and Why Do Software Developers Face Uncertainty?,” Proc. of QRS, pp.288–299, 2019.

[4] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

[5] W. Cunningham, “The WyCash portfolio management system,” Proc. of OOPSLA, pp.29–30, 1992.

[6] W.H. Brown, R.C. Malveau, H.W. McCormick, and T.J. Mowbray, AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis, John Wiley & Sons, 1998.

[7] V. Lenarduzzi, A. Sillitti, and D. Taibi, “A Survey on Code Analysis Tools for Software Maintenance Prediction,” Proc. of ICSE, pp.165–175, 2018.

[8] D. Garlan, “Software engineering in an uncertain world,” Proc. of FoSER, p.125–128, 2010.

[9] S. Elbaum and D.S. Rosenblum, “Known unknowns: Testing in the presence of uncertainty,” Proceedings of the 22nd International Symposium on Foundations of Software Engineering (FSE), pp.833–836, 2014.

[10] D. Perez-Palacin and R. Mirandola, “Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation,” Proc. of ICPE, pp.3–14, 2014.

[11] T.J. McCabe, “A Complexity Measure,” IEEE Transactions on Software Engineering, vol.SE-2, no.4, pp.308–320, 1976.

[12] S.R. Chidamber, D. Darcy, and C. Kemerer, “Managerial use of metrics for object-oriented software: An exploratory analysis,” IEEE Transactions on Software Engineering, vol.24, pp.629–639, 1998.

[13] G. Bavota and B. Russo, “A large-scale empirical study on self-admitted technical debt,” Proc. of MSR, pp.315–326, 2016.

[14] B. Gelman, B. Obayomi, J. Moore, and D. Slater, “Source Code Analysis Dataset,” Data in Brief, vol.27, pp.1–6, 2019.

[15] G. Macbeth, E. Razumiejczyk, and R. Ledesma, “Cliff’s Delta Calculator: A non-parametric effect size program for two groups of observations,” Universitas Psychologica, vol.10, pp.545–555, 2011.

[16] M. Famelis, R. Salay, and M. Chechik, “Partial models: Towards modeling and reasoning with uncertainty,” Proc. of ICSE, pp.573–583, 2012.

[17] 深町拓也，鵜林尚靖，細合晋太郎，亀井靖高，“不確かさを包容する Java プログラミング環境,” 情報処理学会研究報告，vol.2015-SE-187，no.21，pp.1–8，2015.

[18] N. Esfahani, K. Razavi, and S. Malek, “Dealing with uncertainty in early software architecture,” Proc. of FSE, pp.1–4, 2012.

[19] 村岡北斗, 村本大起, 鵜林尚靖, 亀井靖高, 佐藤亮介, “Git 開発履歴情報に基づく不確かさの可視化,” 情報処理学会研究報告，vol.2017-SE-196，no.29，pp.1–8，2017.

建築設備自動制御のソフトウェアへの問題フレーム適用について

井口日文　中谷多哉子
放送大学教養学部教養学科情報コース
1520810114@campus.ouj.ac.jp

要旨

近年，建築物の冷房用冷熱源設備の自動制御設備（以下、自動制御設備）で利用が増えている *Programmable Logic Controller(*以下、*PLC*）のソフトウェアを開発する際、ソフトウェアの機能を記述する手法は，建築設備設計者に任されている。従来，案件ごとの制御の内容が大きく違わなかったことや，ソフトウェアの定義が必要ない制御で済んでいたため，ソフトウェアの機能定義の要領が確立されておらず，機能の記述をコメントで済ませる設計者も多い．その結果，*PLC* ソフトウェアのテストや改修の際，ソフトウェアに建築設備設計者の設計意図が反映されていることを確認することが困難になっている．自動制御設備に期待される機能は高度化，多様化しており，ソフトウェアの機能が複雑になってきている．ソフトウェアの機能を分析し，定義をする手法が必要であると考えられる．

本研究では，*PLC* の機能を構造化する手順として，問題フレーム（*Problem Frame)* を導入することで，ソフトウェア機能記述の構造化を試みた．また，作成した問題フレームが建築設備の実務者に理解されるものであることをインタビューで確認した．

1. はじめに

近年，自動制御設備に PLC が導入されるようになっている．PLC の機能はソフトウェアで実装されており、その機能を定義するためには、ソフトウェア設計図書を作成する必要がある．しかし，ソフトウェアの設計を専門とする技術者が設計図書を作成するのではなく、建築設備の専門家がその責を担っている。そのため、ソフトウェアの機能を定義するにあたり，ハードウェアの機能と区別することができず，コメント，箇条書きと若干の図表でソフトウェアの振る舞いを記述する方法がとられている。従って，

- 機能の記述が体系的でなく、複雑な制御になった場合、解釈の多義性や矛盾が生じる
- 設計図書の読み手によって、理解の内容が異なり、設計図書の機能を果たさない
- ソフトウェアの開発と検証作業を再現可能なプロセスとして定義できない．
- ZEB（Zero Energy Building）など，エネルギーを効率的に利用するための要求が高度化する一方，十分な仕様記述ができない

などの課題が生じている．

問題フレームは，ソフトウェアが使われる環境の問題とその問題を構成するドメインを定義し，現実世界とドメインの構造と特徴を明らかにする枠組み（フレーム）である [1]．現実世界とドメインの関係を構造化することで，マシンとドメインとの間の関係を関心事として明記してマシンの要求仕様を定義する．

複雑な問題の場合は，段階的詳細化という作業を経て，粒度の小さい問題を発見し，本稿では，問題フレームを自動制御設備の要求仕様を定義するために適用する．

建築設備エンジニアが直観的に対応している現在の状況を解決するためには，分析プロセスを変え，思考の成果を表記するという二つの変革を遂行していかなければならない．我々の課題は，問題フレームによる表記が，建築

設備エンジニアに理解されるか否かを評価することである．本稿では，自動制御設備の仕様の記述に問題フレームを導入し，建築設備エンジニアがこの技術を導入するときの効果と課題を明らかにする．

本稿では，これらの問題を解決するために，建築設備の専門家が、現実世界とソフトウェアとの境界に着目してソフトウェアの要求仕様を定義することを試行した．この手段として要求工学で開発された，問題フレームを適用する．

本稿は，以下のように構成されている．次の２節では本研究の関連研究を紹介し，３節で問題フレームを用いた要求仕様の定義と建築設備技術者へのインタビュー結果を示す．続く４節で考察を述べ最後の５節で本稿をまとめる．

2　関連研究

2.1　自動制御設備の設計手順と設計図書構成

自動制御設備は，フィードバック制御とシーケンス制御の組み合わせを中心に設計されている．複数台の機材の台数制御などの少数の例外的な機能を除いて，センサ，調節部，操作器などをインプット/アウトプットが１対１になるように組み合わせて構成されている．制御設備がハードウェアで構成されているため，調節部，操作部，センサ部，制御対象を明確にし，対応する機材を適切に選定することで，確実に動作する制御設備を構築することができ，その制約の中で設計されている．そのため、機能のあいまいさ，矛盾は制御図やポイントリストを精査することで回避することができる。表１に設計図書の構成を示す．ソフトウェアの機能を記述する項目がない。

近年、制御の高機能化を図るために PLC が利用されるようになっている．PLC の機能は PLC 内部のソフトウェアで定義される．これにより，台数制御，データの演算・集計，制御出力，ネットワークとの接続など多様な機能を一つのコントローラで実現できるようになった．そのため，設計図書に PLC ソフトウェアの機能を記述することが必要になってきた．しかし，表１に示す設計図書には，記述する図面はなく，箇条書きやコメントとして補足のような要領で記述されることが多い．さらに，IoT の導入、制御対象設備の高機能化が進み，PLC ソフトウェアへの要求が複雑化していることに作図要領が追い付いていない．機能を明確に矛盾ない PLC ソフトウェアの仕様を記述する要領がもとめられている。

表 1. 自動制御設備設備の設計図書の構成

図面名称	説明
概要	建物の用途，建築設備の仕様，制御の目的などの全体像を示す.
制御図	発停制御，ループ制御などを示す.
ポイントリスト	起動，停止，監視，測定，演算，センシングするポイントなどの全体を示す.
機器表	使用する制御機器を記述する．指示調節計，リレー，タイマー，センサー，モーターダンパ，モーターバルブなど.
配線図	建物の中の機器配置，配線経路を示す.

表 2. 自動制御の用語定義

用語	定義
調節部	センサーからの検出値と設定値の差から、その差を小さくするよう操作部に信号を出すドメイン．原則として１入力に１出力の対応関係となっている.
操作部	調節部からの信号で動作する、制御対象を操作する機材．電動弁，電動ダンパ，ポンプインバータ，ヒートポンプなど.
センサ部	制御対象の状態（温度，流量など）を検出する機材.
制御対象	制御される対象．本稿では，熱を運ぶ熱媒.
PLC	Programmable Logic Controller ．汎用コントローラ．ソフトウェアで機能を定義する.

2.2　自動制御設備 設計図書　機能表現の研究例

Edward L. Gotowski,P.E. が自動制御における自動制御ソフトウェアの設計意図を記述するための要領として，設計図書とは別に Sequece of Operations[2] と呼ばれるドキュメントの作成要領と構成を提案している．Gotowski の業績は以下のものであった。

- 自動制御設備の設計図書を利用する技術者は，プロ

グラマ，コミッショニングエイジェント，運用者，将来の運用者・エンジニアなど，専門分野が異なる多くの専門家の集まりであることを定義した
- 専門分野の異なる専門家が共有する情報をまとめるため，「Overall System Descrption」を重視した

Overall System Description では，専門用語を利用せずに，主要な制御対象機器とその機器が対象とする範囲を記述することを推奨している．制御ロジック，変数，設定値，制御ループなどを Overall System Description に続いて，すべて記述することを主張している．記述の必要性を宣言したが、具体的な記述要領は定義されていない。これらの内容を「システム全体の説明」としてまとめ，次に，実務に必要な動作の説明，定義に必要な変数の定義，変数の範囲の定義，制御シーケンス，表示方法，機器や機能の命名ルールなどを順次記載する．

我々は，近年の ICT 技術の高度化に伴う課題に対応するため，PLC ソフトウェア機能を構造化して記述する方法が必要であると考えている．

2.3 組込み制御の要件定義について

SESSAME は，「ハードウェア出身のマネージャーに分かっておいてほしい 7 つのこと」[3] において，組込みソフトウェア開発の課題をまとめている．その中で，生産工程が未成熟であること，詳細を決めずに作ることが多いことを課題として指摘している．我々は自動制御設備の設計図書にソフトウェア開発の上流工程の知見を導入することで，ソフトウェアの仕様を現実世界から求められている事項として詳細化する手法を検討する．

2.4 MBSE(Model Based Software Engineering）との関係

MBSE(Model Based Software Engineering）を組込みソフトウェア開発の上流工程に導入する例が IPA(独立行政法人情報処理推進機構）によって提案されている [4]．モデルの表現には，SysML[5] が使われている。SysML は，UML をシステムエンジニアリング向けに適用させたものである．検討対象を，着目点ごとにユースケース図，ブロック定義図，要求図，シーケンス図など複数の観点から作成した図で表現する方法を定義している。開発プロセスを通じて標準的に使えるモデリング言語とした．しかし，SysML はモデリング言語であり，現実世界とソフトウェアの関係を直接検討する手法ではない．そこで，本研究では制御ソフトウェアの機能を表現するためだけでなく，機能を検討する段階の手順を検討した。

3 問題フレームでの分析例と，実務者インタビュー

3.1 分析対象の概要

本稿では，空気調和設備のうち，自動制御設備を備えた冷熱源設備を対象として問題フレームの適用を検討する．冷熱源設備は冷房に必要な冷熱を製造し，熱媒を循環して建物内に供給するシステムである．熱媒は水が使われることが多い．

冷熱源設備を構成する機材である主な冷熱源機器を以下に示す．原則として PLC からの指示で起動/停止/設定変更などを行う．

- 冷熱を製造するヒートポンプ
- 熱媒に循環するエネルギーを与えるポンプ
- 配管
- 熱媒の流量を測定する流量計
- 熱媒の温度を測定する温度計
- 熱媒の圧力を測定する圧力計
- 熱媒の流れを切り替える電動弁

冷熱源設備は、これらの冷熱源機器で構成されているシステムである．冷熱源設備は建物内各所に設置した空調機と，配管により接続されており，熱媒は配管を介して冷熱源設備と空調機の間を循環する．

3.2 分析対象のドメイン

本稿での検討対象のドメイン定義を表 3 に示す．ドメイン構成を図 1 に示す．

図 1 左手の冷熱源設備で，温度管理された熱媒 (往) は，配管を介して建物内各所の空調機に供給され，建物の冷房に利用される．冷房に利用されて温度が高くなった熱媒 (復) は，冷熱源設備に戻り，冷熱源設備内のヒートポ

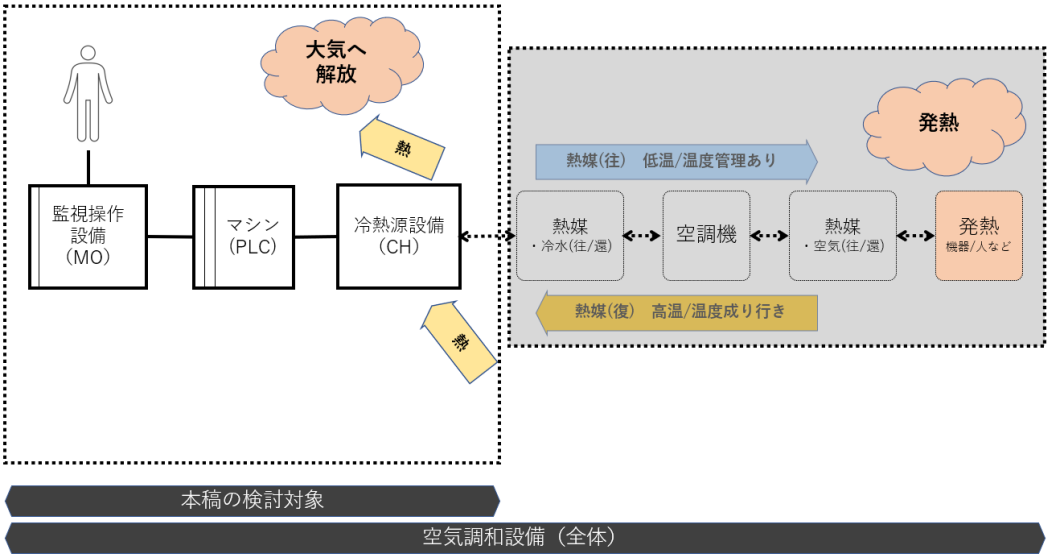


図 1. 空気調和設備のドメイン構成

表 3. 検討対象のドメイン定義

ドメイン	定義
冷熱源設備	冷房に必要な冷熱を製造し供給するシステム
マシン	冷熱源設備の PLC．PLC ソフトウェアで機能を定義される
監視操作設備	冷熱源設備を監視する設備

ンプにより熱を奪われ，冷熱源設備内で温度管理され，ポンプにより加圧され、再び熱媒（往）として建物内各所の空調機へ供給される．熱媒（往）と熱媒（復）の温度差に流量を乗じた冷熱量が，冷房に使用された冷熱である．配管には電動弁，温度計，圧力計，流量計が設置され，熱媒の流量を管理する．ヒートポンプは、熱媒 (復) から奪った熱を大気に放出する．冷熱の製造に伴い発生する排熱は，大気に放出される．

熱媒 (復）の温度や，必要な流量は，

- 建物への日射や外気温などの外乱要因
- 建物室内の人員や機材からの発熱
- 在室人員の変動に伴う換気量の変動

など，多様な要因で変動するため，冷熱源設備は変動に追従することが求められる．また，用途によっては，無停止を求めれらるため，さらに

- 冷熱源機器のメンテナンスや故障など起因の停止
- 電源設備の切り替えによる一時停止/自動再起動

などの要求を満足することが求められる．

今回対象となるマシンである PLC は冷熱源設備を構成する一部であり，その機能は PLC のソフトウェアで定義されている．

3.3 コンテキストダイアグラム

図 2 に，今回の検討対象である冷熱源設備のコンテキストダイアグラムと，ドメイン間の共有現象を示す．

3.4 PLC ソフトウェア機能分析と表現

図 3 から図 6 に，冷熱源設備の代表的な機能である，

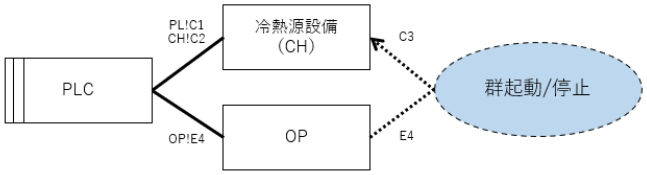
- 冷熱源設備発停（命令された振舞）
- 負荷変動への追従（必要とされる振舞）
- 監視操作設備への表示（情報表示のフレーム）
- 熱量の演算（変換フレーム）

記号	ドメイン間の共有現象
a;PLC!	・起動/停止 ・冷熱源機器の制御指示
a;CH!	・熱媒の状態 ・冷熱源機器の運転状態
b;MO!	・OPの指示
b;PLC!	・熱媒の状態 ・冷熱源設備の運転状態
c;CH!	・熱媒（往）を温度一定にする ・要求量に合う量の熱媒を送り出す
c;WH!	・熱媒（復）をCHに返す
d;MO!	・熱媒（往）を温度一定にして送出する ・要求量に合う量の、熱媒を送り出す
d;OP!	・起動/停止 ・設定値を入力

図 2. 冷熱源設備のコンテキストダイアグラム

について作成した問題フレームを示す.

図 3 に，冷熱源設備の発停に関する問題フレーム (命令された振舞) を示す．オペレータ操作であり，「命令された振舞」を適用する.



PL!C1	冷熱源設備の状態変更指示；起動/停止
CH!C2	冷熱源設備の状態伝達；起動/停止
OP!E4	オペレータの操作；起動/停止（自発的）
C3	熱源群起動/停止
E4	オペレータの操作；起動/群停止

図 3. 冷熱源設備の ON/OFF(命令された振舞)

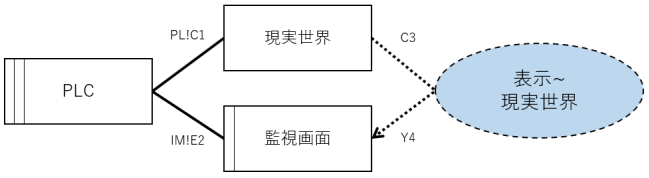
図 4 に，負荷変動に対する冷熱源設備の追従動作に関するの問題フレーム（必要とされる振舞）を示す.

図 5 に, 監視操作設備への情報表示に関する問題フレーム（情報表示）を示す.

図 6 に，熱量の演算に関する問題フレーム（変換フレー

PL!C1	冷熱源設備の設定指示；熱媒（往）温度設定
CH!C2	熱源設備の状態伝達；熱媒（往）状態
C3	要求の変動；要求熱量増/減への追従 要求の変動；熱媒（往）温度一定

図 4. 負荷変動に対する冷熱源設備の追従動作に関するの問題フレーム（必要とされる振舞）



PL!C1	熱源設備の状態を出力
IM!E2	なし
C3	
Y4	

図 5. 情報出力に関する問題フレーム（情報表示）

ム）を示す.

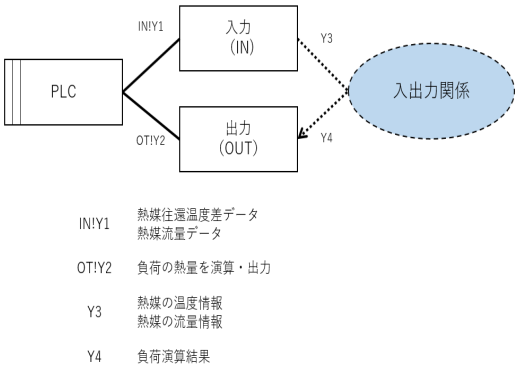


図 6. 熱量の演算関する問題フレーム（変換フレーム）

3.5 建築設備技術者の可読性の確認

問題フレームやドメインはソフトウェア工学の知見であり、建築環境工学や機械工学の知見ではない。そのため、設計図書に記載しても読み取られない可能性がある. 日常業務で，従来の作図方法を使っている技術者に作成したコンテキスト図，問題フレームを提示し，実務者の評価を受けた．実務者は実務経験のある，担当業務が異なる３人のインタビュイを選定した.

3.6 インタビュイの属性と内容

インタビュイの属性を表 4 に示す．自動制御設備の設計，施工一定の業務経験を積んでいるインタビュイを選定した．設計業務のうち，計画寄りで，自動制御設計/機械設備設計/電気設備設計など広い業務範囲の技術者１，施工寄りで自動制御設備に特化してのハードウェアやソフトウェアなどを業務範囲とする技術者３、その中間である技術者２の，業務担当範囲がことなる３名を選定した.

各インタビュイへ，それぞれ２回インタビューを行った．２回とも，同じ図を提示している．１回目は説明なしで読み取り内容を報告させた．２回目は，説明を添えてディスカッション形式で説明を実施し，気づいたこと，評価ポイント，改善点などをヒアリングした.

表 4. インタービュイーの属性

	技術者１	技術者２	技術者３
職種	設備設計者	自動制御設備設計者	自動制御盤/PLC ソフトウェア設計者
担当業務範囲	建築・電気・機械設計	機械設備設計	自動制御設備
実務経験	８年	１５年	８年

各インタビュイへの提示した図は，図 2 のコンテキスト図１面と，図 3～図 6 の問題フレーム４面である.

各インタビュイへのインタビュー手順を表 5 に示す.

表 5. インタビューの手順

試行	説明	インタビュー内容
１回目	なし	ヒアリングにより読み取り内容を説明させた.
２回目	あり	ディスカッション形式で説明．気づいたこと，評価ポイント，改善要望などをヒアリングした.

２回目のディスカッション内容と目的を表 6 に示す.

表 6. インタビューの内容と目的

質疑	目的
説明なしで理解できたか	設計図書として利用できることを確認する.
評価ポイント	現在設計図書が抱えている課題が改善されていることを実感できるか否か確認する.
汎用性	他用途建物に適用可否を意見聴取する．外部妥当性の評価資料を取得する

3.7 インタビューの結果

インタビュー結果を表 7 に示す．ポイントリストで表現した場合と比較して，

- 要求と仕様の関係が，図で記述されているため，分かりやすいこと．従来の作図方法では，要求を推定しながら設計図書を読んでいたが，その必要がないこと

- ポイントリストで表現しようとしていたことに無理があったことが分かったこと．

が報告されている．これらはフレームを使用した表現に由来する効果であり，問題フレームを導入した結果であると考えられる．このことから，問題フレームは，自動制御設備の実務者に理解されうると判断できる．

4 考察

4.1 外部妥当性について

インタビューの中で，問題フレームを冷熱源設備以外に適用できるかヒアリングを行い，全員から，冷熱源設備制御だけではなく一般的な建築設備制御の分野でも使えるという回答を得ており，外部妥当性があると判断される．

4.2 内部妥当性について

インタビュイは半年程度の期間，業務を通じ筆者と共同作業を行っていた。従来の作図方法であれば記述に矛盾やあいまいさが残ることは共通認識があり，解決策が必要であるという問題意識を共有するバイアスが掛かっていると考える必要がある．このため，今後さらに多くのインタビュイの評価を受ける必要がある．

4.3 問題フレームが設計図書へ及ぼす効果

自動制御設備の高機能化に伴い，PLC が利用されるようになっている．PLC はソフトウェアで機能を定義するため，設計図書に PLC ソフトウェアの機能を記述する必要がある．従来，PLC の機能はシーケンス図で記述されてきたが、シーケンス図で記述できる内容よりさらに多くの機能を PLC に期待するようになっている．又、自動制御設備の設計者は，シーケンス図に習熟していないことが多い問題も有る．現在は，ハードウェアで制御設備を構成していた時代の作図方法を踏襲し発展させて対応している．一例として，ポイントリストの例を 7 に示す．ポイントリストは，機能を表に記号で記述する作図方法であり，記号で表示される各行の記述内容は，明快で記述も容易である．機能が複雑化，高度化した場合，表現しきれない機能は自然言語で追記する方法が採られる．また，全体の機能を記述する概要は，リストに記載しきれないため，別図に自然言語やフロー図で記載する．その為，

- 記述の重複，矛盾などが発生することがある．また，実務経験者であっても設計図書の間違いの発見が難しいこと
- 要求が分析・明記されておらず，また要求とソフトウェアの仕様の関係が記載されていないため，ソフトウェアが要求に適合していることを確認する作業が困難であること．また，要求変更が発生した場合の対応が難しいこと
- 要求が複雑になった場合，要求の分析，記述が難しくなること．曖昧な記述になる可能性があること

などの課題がある．この結果，現場では「設計は正」という前提のもと，施工者チェックが働かない課題があった．

問題フレームは複雑な現実世界の要求を分析・分割し，記述する手法である．設計図書に記述したい複雑な要求を，小さな要求に分割し，フレームで記述する．その為，

- 要求とソフトウェアの仕様の関係をフレームで明示できる
- 視点を制限することで設計者の個人差が現れにくくなり，設計図書としての品質が均質化する効果が期待できる

などの特徴がある．

これらの問題フレームの特徴により，

- 要求ごとに一葉ずつ問題フレームが記述されるため，記述の重複，矛盾などを回避することができること
- ソフトウェアが要求に適合していることの確認が容易になる．要求変更が発生した場合の対応が容易になること
- 複雑な要求を設計段階で合理的に分割し，記述できるようになることで，曖昧さを回避できること
- 設計者ごとの設計図書品質のばらつきが少なくなること

などの効果が期待できる．今後，自動制御設備への要求

表 7. インタビュイの回答

インタビュイ	技術者１回答	技術者２回答	技術者３回答
説明なしでの理解	・理解できた	・理解できた	・理解できた
評価ポイント	・冷熱源設備制御の全体と部分の関係を理解できる.	・見積段階では，時間に限りがあり，ポイントリストのみで表現している．関係はその中でコメントや記号で表現しようとしているが，それが無理表現できない関係が一目でわかった.	・見積段階では，ポイントリストのみで表現している．関係はその中でコメントや記号で表現しようとしているが，それが無理なことが一目でわかった.
	・機能と関係が直接記載されている．従来は，関係を読み取るために，配線図と箇条書きと注釈から読み取った結果から関係やそれぞれの役割を読み取る必要があったが，作図上の問題が改善されている.	・部分と部分の関係が若手に伝承が難しい経験値と考えていたが，わかりやすく図示されている.	・読み手の経験値や技術的な背景を期待しないで話ができるため，使いやすい.
	・作図すべき図面が増える．作業時間が取れないので，作図時期を考慮する必要がある.	・作図すべき図面が増える．作業時間が取れないので，作図時期を考慮する必要がある.	・読み手の経験値や技術的な背景を期待しないで話ができるため，使いやすい.
汎用性	・冷熱源設備制御設備以外の制御設備にも適用できる.	・冷熱源設備制御設備以外の制御設備にも適用できる.	・冷熱源設備設備以外の制御設備にも適用できる.

がさらに高度化，複雑化することが見込まれるが，ソフトウェア技術者，建築設備技術者双方にわかりやすい表現が可能になることが期待できる.

4.4 設計図書へのドメインの導入について

問題フレームを導入するにあたり，ドメインを定義する必要がある．しかし，建築設備の分野にはソフトウェア工学の知見が導入されていないこともありドメインを定義する習慣がない.

建築設備の文弥では，冷熱源設備（主に屋外に設置），空調機（主に室内に設置），配管・ダクト，制御（含監視操作設備）という４区分に分けて設計・検討することが一般的である．これは,

- 建物の屋外と屋内にそれぞれ機械を設置し配管で接続し，熱媒を循環して冷暖房を行うという建築的な要因
- 建築計画とは直接関係しない自動制御設備

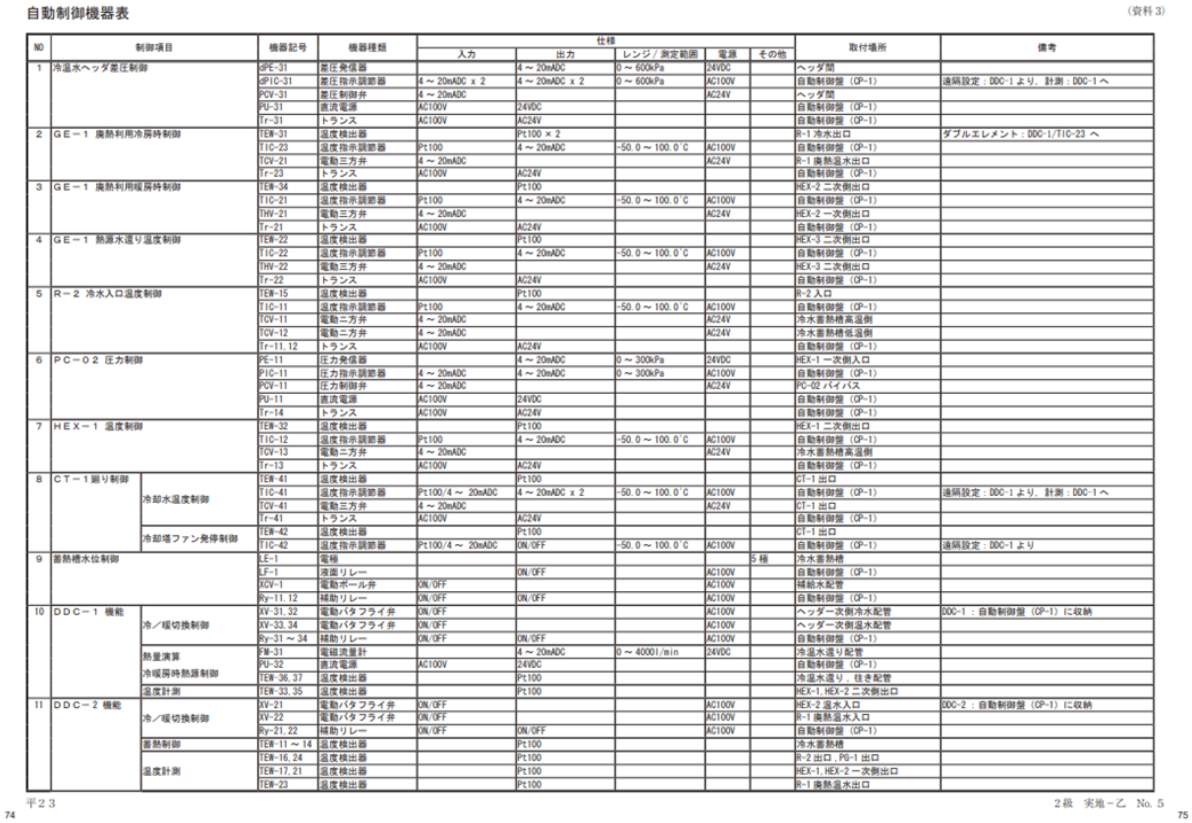
の特性に由来していると考えられる．本研究を進めるにあたり，建築設備技術者にとっての馴染みやすさを考慮し，当初はこの区分をドメインと見做してコンテキストダイヤグラムと問題フレームを作成しようとした．その結果,

- PLC ソフトウェアから現実世界を見た場合に，粒度が違うドメインになっていること
- オペレータのドメインが想定されていないこと

などの不都合が生じた.

本稿では，図１のように，冷熱源設備，配管・ダクト，制御 (含監視操作設備）を一つの例熱源設備として検討対象範囲を決定した．冷熱源設備の中で，マシン，監視操作設備（MO), 冷熱源設備 (CH）をドメインで分けるようにした.

問題フレームを導入するにあたり，設計者は記述したい要求に対し，検討範囲を設定し，その中で適切にドメインを定義する必要がある．建築設備技術者はハードウェアから考える習慣がついているため，習熟に課題があることが考えられる.



自動制御機器表

（資料3）

NO	制御項目		機器記号	機器種類	仕様					取付場所	備考
					入力	出力	レンジ／測定範囲	電源	その他		
1	冷温水ヘッダ差圧制御		dPE-31	差圧発信器		4 ～ 20mADC	0 ～ 600kPa	24VDC		ヘッダ間	
			dPIC-31	差圧指示調節器	4 ～ 20mADC x 2	4 ～ 20mADC x 2	0 ～ 600kPa	AC100V		自動制御盤（CP-1）	遠隔設定：DDC-1 より，計測：DDC-1 へ
			PCV-31	差圧制御弁	4 ～ 20mADC			AC24V		ヘッダ間	
			PU-31	直流電源	AC100V	24VDC				自動制御盤（CP-1）	
			Tr-31	トランス	AC100V	AC24V				自動制御盤（CP-1）	
2	GE－1 廃熱利用冷房時制御		TEW-31	温度検出器		Pt100 × 2				R-1 冷水出口	ダブルエレメント：DDC-1/TIC-23 へ
			TIC-23	温度指示調節器	Pt100	4 ～ 20mADC	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	
			TCV-21	電動三方弁	4 ～ 20mADC			AC24V		R-1 廃熱温水出口	
			Tr-23	トランス	AC100V	AC24V				自動制御盤（CP-1）	
3	GE－1 廃熱利用暖房時制御		TEW-34	温度検出器		Pt100				HEX-2 二次側出口	
			TIC-21	温度指示調節器	Pt100	4 ～ 20mADC	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	
			THV-21	電動三方弁	4 ～ 20mADC			AC24V		HEX-2 一次側出口	
			Tr-21	トランス	AC100V	AC24V				自動制御盤（CP-1）	
4	GE－1 熱源水還り温度制御		TEW-22	温度検出器		Pt100				HEX-3 二次側出口	
			TIC-22	温度指示調節器	Pt100	4 ～ 20mADC	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	
			THV-22	電動三方弁	4 ～ 20mADC			AC24V		HEX-3 二次側出口	
			Tr-22	トランス	AC100V	AC24V				自動制御盤（CP-1）	
5	R－2 冷水入口温度制御		TEW-15	温度検出器		Pt100				R-2 入口	
			TIC-11	温度指示調節器	Pt100	4 ～ 20mADC	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	
			TCV-11	電動二方弁	4 ～ 20mADC			AC24V		冷水蓄熱槽高温側	
			TCV-12	電動二方弁	4 ～ 20mADC			AC24V		冷水蓄熱槽低温側	
			Tr-11, 12	トランス	AC100V	AC24V				自動制御盤（CP-1）	
6	PC－02 圧力制御		PE-11	圧力発信器		4 ～ 20mADC	0 ～ 300kPa	24VDC		HEX-1 一次側入口	
			PIC-11	圧力指示調節器	4 ～ 20mADC	4 ～ 20mADC	0 ～ 300kPa	AC100V		自動制御盤（CP-1）	
			PCV-11	圧力制御弁	4 ～ 20mADC			AC24V		PC-02 バイパス	
			PU-11	直流電源	AC100V	24VDC				自動制御盤（CP-1）	
			Tr-14	トランス	AC100V	AC24V				自動制御盤（CP-1）	
7	HEX－1 温度制御		TEW-32	温度検出器		Pt100				HEX-1 二次側出口	
			TIC-12	温度指示調節器	Pt100	4 ～ 20mADC	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	
			TCV-13	電動二方弁	4 ～ 20mADC			AC24V		冷水蓄熱槽高温側	
			Tr-13	トランス	AC100V	AC24V				自動制御盤（CP-1）	
8	CT－1 廻り制御		TEW-41	温度検出器		Pt100				CT-1 出口	
		冷却水温度制御	TIC-41	温度指示調節器	Pt100/4 ～ 20mADC	4 ～ 20mADC x 2	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	遠隔設定：DDC-1 より，計測：DDC-1 へ
			TCV-41	電動三方弁	4 ～ 20mADC			AC24V		CT-1 出口	
			Tr-41	トランス	AC100V	AC24V				自動制御盤（CP-1）	
		冷却塔ファン発停制御	TEW-42	温度検出器		Pt100				CT-1 出口	
			TIC-42	温度指示調節器	Pt100/4 ～ 20mADC	ON/OFF	-50.0 ～ 100.0℃	AC100V		自動制御盤（CP-1）	遠隔設定：DDC-1 より
9	蓄熱槽水位制御		LE-1	電極					5 極	冷水蓄熱槽	
			LF-1	液面リレー		ON/OFF		AC100V		自動制御盤（CP-1）	
			XCV-1	電動ボール弁	ON/OFF			AC100V		補給水配管	
			Ry-11, 12	補助リレー	ON/OFF	ON/OFF		AC100V		自動制御盤（CP-1）	
10	DDC－1 機能		XV-31, 32	電動バタフライ弁	ON/OFF			AC100V		ヘッダー次側冷水配管	DDC-1：自動制御盤（CP-1）に収納
		冷／暖切換制御	XV-33, 34	電動バタフライ弁	ON/OFF			AC100V		ヘッダー次側温水配管	
			Ry-31 ～ 34	補助リレー	ON/OFF	ON/OFF		AC100V		自動制御盤（CP-1）	
		熱量演算	FM-31	電磁流量計		4 ～ 20mADC	0 ～ 4000l/min	24VDC		冷温水還り配管	
		冷暖房時熱源制御	PU-32	直流電源	AC100V	24VDC				自動制御盤（CP-1）	
			TEW-36, 37	温度検出器		Pt100				冷温水還り，往き配管	
		温度計測	TEW-33, 35	温度検出器		Pt100				HEX-1, HEX-2 二次側出口	
11	DDC－2 機能		XV-21	電動バタフライ弁	ON/OFF			AC100V		HEX-2 温水入口	DDC-2：自動制御盤（CP-1）に収納
		冷／暖切換制御	XV-22	電動バタフライ弁	ON/OFF			AC100V		R-1 廃熱温水入口	
			Ry-21, 22	補助リレー	ON/OFF	ON/OFF		AC100V		自動制御盤（CP-1）	
		蓄熱制御	TEW-11 ～ 14	温度検出器		Pt100				冷水蓄熱槽	
		温度計測	TEW-16, 24	温度検出器		Pt100				R-2 出口，P0-1 出口	
			TEW-17, 21	温度検出器		Pt100				HEX-1, HEX-2 一次側出口	
			TEW-23	温度検出器		Pt100				R-1 廃熱温水出口	

平２３ 74

２級　実地－乙　No. 5 75

図 7. ポイントリストのサンプル

4.5　冷熱源設備へ要求の種類と問題フレーム

本稿では，冷熱源設備の要求として，４例作成した．そのほかの冷熱源設備への主な要求として，冷熱源設備への要求例と、適用する問題フレームの種類を表 8 に例示する．

表 8. 冷熱源設備への要求例と、適用問題フレーム

問題フレーム	要求
命令された振る舞い	冷熱源設備起動（システムの起動/停止）
必要とされる振舞	熱媒（往）温度一定制御
	要求熱媒の増減に対する供給量一定制御
	冷熱製造量に対する冷熱源消費電力の最小化
情報表示	冷熱源設備を構成する機材の稼働状況
	各センサーの読み値表示.
	トレンドグラフの作成.
	運転データの統計値の表示.
変換	各種フィードバック制御.
	需要予測，運転予測.
単純な仕掛品	該当なし.

5　まとめ

近年，建築設備の自動制御に PLC が利用されるようになり，PLC の機能を定義するソフトウェア機能を設計図書に記述する必要が生じている．しかしソフトウェアの機能は，ハードウェアの機能と区別されずに，コメント，箇条書きと若干の図表で記述する方法がとられている．

制御システムに組み込む PLC のソフトウェアに関わる要求を分析し，構造化することで，相互に作用しあう副問題の集まりとして問題を構造化することができ，考慮す

べき事項が明らかになり，段階的詳細化の手順が明らかになると考えられる．

本稿ではソフトウェア工学の知見である問題フレームを，自動制御設備の問題構造化に適用し，ドメインを適切に設定することで，制御構造を明示できることをを確認した．また，作成した問題フレームを実務者に提示し，設計意図が共有できることを確認した。考察として，設計図書の一部として，設計者が問題フレームを作成する場合の課題を考察した．

インタビューの結果，設計意図の伝達に有効であること，ケーススタディ以外の設備にも適用可能と考えられることが確認された．今後，実務に適用していくにあたり，より専門分野が異なる多くのインタビュイへのリアリングによる評価と，他設備への適用にむけた検討が必要である．

参考文献

[1] マイケルジャクソン．プロブレムフレーム　ソフトウェア開発問題の分析と構造化．株式会社廣済堂, 2006 年 5 月 22 日.

[2] EDWARD L GOTOWSKI. *Sequence of Operation for Mission Critical Building Automation Systems.* ASHRAE JOURNAL, JULY,2016.

[3] SESSAME 組込みソフトウェア管理者・技能者育成研究会．ハードウェア出身のマネージャーに分かっておいてほしい 7 つのこと ver1.01. *www.sessame.jp.*

[4] IPA 独立行政法人情報処理推進機構. モデルベースシステムズエンジニアリング導入の手引き.

[5] Ph.D Laurent Balmelli. An overview of the systems modeling language for products and systems development. pp. 149–177, 2007.

COTS 利用プロジェクトへの GSN の適用

田中 康
有限会社ケイプラス・ソリューションズ, 奈良先端科学技術大学院大学
ytanaka@kplus-solutions.com, yasushi-tanaka@is.naist.jp

要旨

業務改革や新サービス開発, そして, IT システムの新規開発およびシステム刷新プロジェクトのコスト超過や有用性実現の失敗を解決するために, 業務プロセス設計主導によるシステム要件定義の取り組みを行なっている[1]. しかし, 業務プロセス設計はユーザ企業の参画含め, 現状の業務プロセスを分析し, 改善のための業務プロセスを再設計するといった一連の過程を踏む必要があり, 期間とコストがかかることが実施を困難にする要因のひとつであった.

業務での活用思想が明確な COTS (Commercial Off-The-Shelf) 製品を適用するプロジェクトでは, 業務プロセス設計フェーズに GSN (Goal Structuring Notation) [2]を適用することによって, 現状プロセスのモデル化と分析の過程を踏まずに, あるべき業務プロセスをトップダウンに設計することができ, さらに, 実働, 期間共に大幅な短縮を実現することができた.

1. はじめに

一般的な上流工程のアプローチは, ユーザからの要求を受け取り, 受け取った要求をシステム要件として解釈して定義し, ユーザとの合意を得るところがプロジェクトの開始地点となっている. しかし, 今でもなお, 約半数近くの IT システムの導入や刷新プロジェクトが失敗に終わっている. さらにその主要な要因に関しても, 要件定義をはじめとする上流工程の不備にあることが原因としてあげられている. この状況はソフトウェア工学がはじまって以来続いている[3]. このような上流工程での問題に対して筆者は, 「逆 V モデル」と呼んでいる業務プロセス設計主導での上流工程アプローチを行なっている[1][4].

1.1. 業務プロセス設計のための逆 V モデル

逆 V モデルとは, 図 1に示すように, 開発の V モデルを逆さまにした形のプロセスであることから命名したものである. 逆 V モデルは, 業務プロセスの分析と設計が行われる「業務レベル」を中心に, 目標とする経営ゴールが定義される「経営レベル」, 業務を支援する IT システムに対する要求仕様が定義される「要求仕様レベル」の 3 層を定義している.

業務レベルでは, 経営目標達成を阻害している業務プロセスの問題を特定するために, 現状の業務 (As-is 業務) プロセスをモデル化して問題の発生原因の分析を行う. そして特定した原因分析をもとに, あるべき業務 (To-be 業務) の設計を行う.

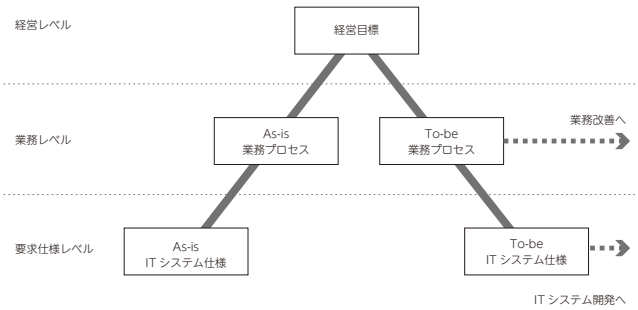


図 1 逆 V モデル

1.2. 逆 V モデル実行上の課題

筆者は, 日立製作所との協業で, 様々なプロジェクトに対して逆 V モデルを適用し, 実績を積み上げてきた. しかし一方で, 逆 V モデルに従って業務プロセスの分析と設計を行う場合, 前半の As-is 業務プロセスのモデル化と分析に約 1.5 ヶ月, To-be 業務プロセス設計に約 1.5 ヶ月の期間が平均して必要となる.

さらに, 業務プロセスの分析と設計を行うためには, ユーザ企業から複数名の参加者を頼まなければならない. 参画を頼むユーザ企業側の人員は, 対象業務に精通している必要があり, また, 設計した業務を現場に実装するフェーズでは, 業務改革の推進的な役割も担う人員である必要があるため, ユーザ企業で中心的な役割を担っている社員である場合が多く, そのような人員の工数を確保することが難しい場合も多い.

ユーザ企業側に業務マニュアルなどがある場合は，それらをもとに現状のプロセスを記述することによって As-is 業務のモデル化の期間を短縮することもできる．しかし，業務マニュアルが実際の業務を十分に反映していない場合も多く，正確な As-is 業務をモデル化することができないこともある．業務の問題分析をするためには，精度の高い As-is 業務プロセスモデルが無いと，問題を発生させている原因の特定が難しくなってしまう．正しく問題を特定することができないと，To-be 業務プロセス自体の設計品質も低くなってしまう．

逆 V モデルに従った上流工程では，業務プロセスの改善と，改善設計された業務プロセスからそれを支援する IT システムの要件を導出することができるため，改善された業務プロセスの定義と，それを支援する IT システムの仕様とのトレーサビリティーを確保することができる．その一方で，上流工程に期間とコストがかかることが問題であった．

そのような中，Salesforce 社の SFA（Sales Force Automation）という営業支援ツールの導入プロジェクト支援の依頼があった．しかし短期間で結論を出さなければならないという制約があり，逆 V モデルの適用が難しいと判断した．代替のアプローチ方法を検討する中で，業務プロセス基本設計へ GSN を適用することによって，プロセス設計を効率よく進められるのではないかという仮説レベルで検討を進めていた方法を適用することとなった．

2. 業務プロセス設計への GSN 適用の検討

欧州では近年，システムの安全性を確保する責任が開発者や運営者に移行している．そのような中，開発するシステムが，要求される安全レベルを達成していることの根拠のある論拠を構築し提示することが求められている[2]．GSN は，そのようなシステムの安全性論証などで用いられる記法である．

2.1. GSN の記法

GSN の一般的な記法を図 2に示す．GSN では，システムが達成すべきゴールと，ゴール達成を実現するための方法や考えかたを構造的に可視化するための記法が定義されている．GSN のトップレベルには，システムが達成すべきゴールが定義される．ゴールの下に，達成すべきゴールを実現するために検討された方針が戦略として記述される．戦略には，戦略が達成されるための前提条件が紐つく．そして，戦略を実現するための方略がサブゴールとして戦略の下に展開される．それぞれのサブゴールには，それが実現されていることを示す証拠が接続される．

システムの安全性を担保するためには，与えられた前提条件のもとで，さまざまな方法（戦略）を考えることができる．前提条件を満足しているのであれば，どの戦略が正しいということではない．GSN を用いてゴールを達成するための戦略を定義し，戦略を実現するための内容が戦略実現に対して十分であれば，末端のノードである証拠を上げることによって，遡求的にゴールが達成できることを保証することができるという論法である．

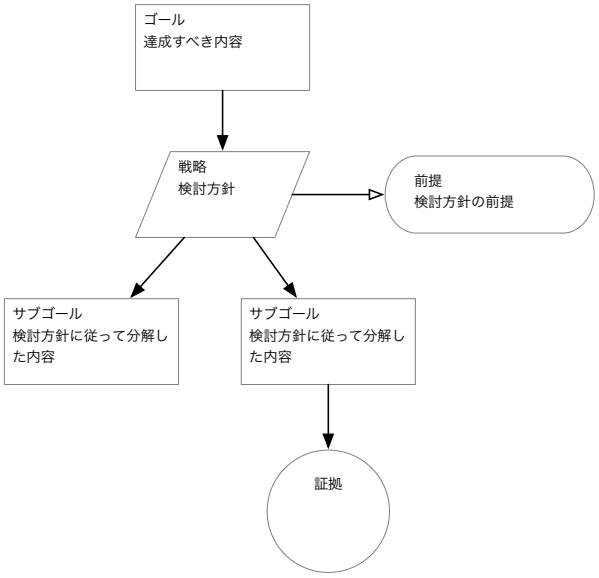


図 2 GSN COMMUNITY STANDARD VERSION 1 に基づく GSN の代表的な記法

2.2. PReP モデルへの GSN 適用の検討

逆 V モデルでは，業務レベルの設計方法として PReP（プレップ）モデルを用いている．PReP モデルは，業務プロセスを，「成果物」と呼んでいる概念実体の関係で構造的に記述するプロセス記述手法である[4] [5].

設計には，その前提条件と，検討された方略がある．多くの場合，設計された結果は提示されるが，その裏で検討された戦略が明示的に記述されることは少ない．さらに，戦略検討の前提条件が整理されて記述されることも少ない．

PReP モデルを用いて逆 V モデルの右側である，業務プロセスの設計を行う際にも，実現のための戦略があり，その前提となっている条件がある．現時点での PReP モ

デルは，設計の結果のみが記述される仕様であるため，検討された戦略やその前提条件を整理する記法は組み込まれていない．GSN は，PReP モデルを記述するツール開発の検討の中で To-be 設計をする際に適用が検討されていた．

2.3. 類似手法と本アプローチとの違い

ゴールとゴールを達成するための戦略とその達成方法を構造的に分析・展開する類似のアプローチとして，ゴール指向のソフトウェアメトリクス手法である GQM（Goal-Question-Metric）[6]に組織戦略の観点を加えた GQM+Strategies の研究がある[7]．

GQM+Strategies の研究では，例えば早稲田大学ゴール指向経営研究会は，組織目標と戦略の整合化への適用を目的としている[8]．この研究は，バランスト・スコアカード（Balanced Scorecard）[9] などと同様な目的を持つものであると考えられる．すなわち，企業の経営目標と，そこから展開される組織戦略とを整理し，さらに財務指標などのメトリクスとの整合性を管理する方法として GQM+Strategies の応用を研究しており，組織目標管理が主要な関心事となっていると理解される[10]．一方で，PReP モデルでは，業務プロセスと組織とは直行するものであるとしている．組織は企業内のリソースを管理するための手段であり，企業価値を生み出すものはあくまでも業務プロセスであると考えている[4]．

また，高井らは，経営目標からシステム要件をつなぐ方法として，GQM+Strategies の適用研究を報告している[11]．高井らのアプローチは，GQM+Strategies を用いて経営目標から構造化された戦略と戦略達成のためのサブゴール構造をシステムエンジニアに引き渡す．システムエンジニアはそこからユースケースなどを用いてサブゴールを実現するためのシステム要件を展開して SysML のアクティビティ図などに展開する．一方で PReP モデルは，高井らのアプローチである GQM+Strategies を用いたビジネス分析とシステム要件定義の間を接続する方法であると考えることができ，概念実体による構造観点で業務プロセスをモデル化することによって，設計した業務プロセスとシステム要件とのトレーサビリティーを確保している[4] [5]．

3. COTS 活用とその失敗

防衛産業や航空宇宙産業などでは，そこで利用されるソフトウェアの品質が人命に直結するため，非常に高い品質が求められる．一方で，高い品質が要求されるソフトウェアを毎回開発するとなるとコストが高くなってしまう．そこで，十分な品質や機能が確認されている市販の既製品，すなわち COTS を活用することで開発期間の短縮やコスト削減を行うことが研究された．民生分野においても同様に，システム開発の一部，もしくは全体に市販製品を採用することを COTS と呼んでいる．

近年では，クラウド上にアプリケーションや API を配備することによって，ユーザー企業がすぐに COTS を利用することができるサービスである SaaS（Software as a Service）も主流となっている．Salesforce 社の SFA もそのような COTS のひとつである．特に，営業業務領域での Salesforce 社の製品はデファクト的な存在であり，多くの企業が導入を試みている．

3.1. 失敗率の多い COTS プロジェクト

システム導入や刷新プロジェクトの約半数の 47.2%が失敗に終わっているという日経コンピュータの調査報告[3]では，最も失敗しやすいシステム導入プロジェクトの報告もある．成功率が最も低かったプロジェクトは CRM（Customer Relationship Management）と SFA で，その成功率は 43.7%であった．まさに，導入プロジェクトの 6 割近くが失敗に終わっている．

業務で用いられる COTS 製品の中でも，例えば会計関係のツールは，会計業務自体が標準化されているため，仕様も固定化しやすい．一方で，顧客管理や営業業務は，その業務プロセスが標準化されておらず，人に依存している部分が多い．また，企業の文化などによって，活動方法などもさまざまであったりする．そのため，仕様変更が相次いだり，COTS に収まらない部分の追加開発が多くなったりしてしまい，最終的に，COTS 利用の目的である開発期間短縮やコスト削減が実現できない結果となってしまうと考えられる．

3.2. COTS プロジェクト失敗の要因

このような COTS プロジェクトの失敗要因と原因構造を営業支援ツールの導入を例に考察する．COTS 製品導入プロジェクトを見ると，開発者としては，導入する COTS 製品の仕様を確定したいと考える．検討した仕様を営業部門に説明するために，COTS 製品がどのように動作するか，もしくは，製品をどのように使うのかを説明する資料が作られる．例えば，「名刺情報を取得する」や「取引先を登録する」といった操作が記述された資料が作成される．これらの記述は，一見，業務プロセスを記述しているように思えるが，実際には，システム開発者が検討してい

るシステムの仕様を記述したものである.

IT システムは, 業務を支援するための手段である. そのため, まず業務プロセスを定義し, 定義された業務プロセスのどの部分をどのように支援するかといった観点でシステムの仕様が定義されるべきである. 前述の逆 V モデルは, このような業務レベル設計を中心にして, IT システムの仕様を定義するための方法を示すものである.

しかし, 図 3に示すように, COTS 製品が適用される先の営業活動の業務プロセスが定義されていない状態で COTS の仕様を定義しようとすると, 仕様が見えてくる過程で, 見えてきた仕様に対する要求が変動したり, 「それだったら, このようなことができる機能も欲しい」といった機能追加が要求されたりする.

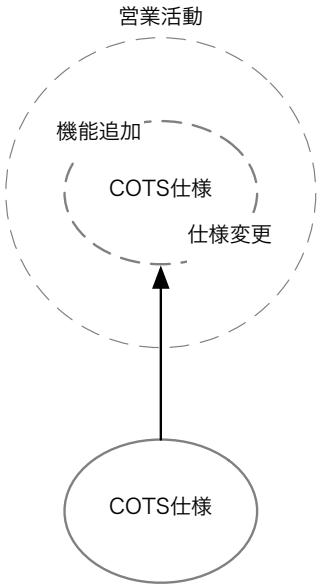


図 3 COTS 導入失敗要因. COTS 製品を適用する先の営業活動の業務プロセスが定義されていない状態で COTS を適用すると, 機能追加や仕様変更が発生してしまう.

4. 人依存の業務プロセスに COTS を適用するには

日本の場合, 営業活動が人に依存している部分が非常に多いと想定される. 実際に, 依頼のあった Salesforce SFA の導入プロジェクトでも, 人依存の営業活動からの脱却が経営目標となっていた.

プロセスが人依存の状態は, ソフトウェア開発組織の能力成熟度モデルでは, レベル 1 と診断される状態である. 営業活動は場当たり的であり, 活動がうまく進む場合もあるが, 成功は個人の能力に大きく依存してしまう. また, 管理層からは, 進捗の見通しが立たないといった状態である.

逆 V モデルに則って, 現状のプロセスを定義しようとした場合, 人に大きく依存した組織では, 営業ごとに異なる様々な活動から, 統一的, もしくは標準的な活動をモデル化する必要があり, 時間とコストが非常にかかると予想される. このような人依存の組織に対して As-is の分析からはじめる逆 V モデルはコスト的に見合わないのではないかと考えられる. しかし, それ以上に COTS 製品を導入する場合に問題となる点がある.

4.1. 逆 V モデルに準じても失敗してしまう可能性

COTS 製品はその設計背景に, 特定の業務の進めかたや, 想定されている利用方法があると考えられる. そのため, COTS 製品の導入が目的化された場合, 業務プロセスの改善が先行せずに, 導入した COTS 製品に仕事の仕方を合わせるといった逆転が起こりがちである.

また, COTS 製品の中には, 導入先からの様々な要求を受け入れて, 当初考えられていた以上の機能が盛り込まれていくものもある. その結果, 機能が総花的になってしまい, 業務での利活用方法が複雑になったり, 曖昧化してしまったりする製品もある.

一方で, しっかりとした業務設計思想に基づいて設計された COTS 製品では, スコープとして想定している業務での活用方法も明確である. ところが, その活用方法を十分に考慮せずに To-be 業務プロセスを設計してしまうと, 図 4に示すように COTS 製品の仕様の背景にある業務プロセスとの差異が潜在してしまう. その結果, 設計した To-be 業務プロセスから導出されたシステム要件と, COTS 製品仕様との不整合が生じてしまう可能性がある.

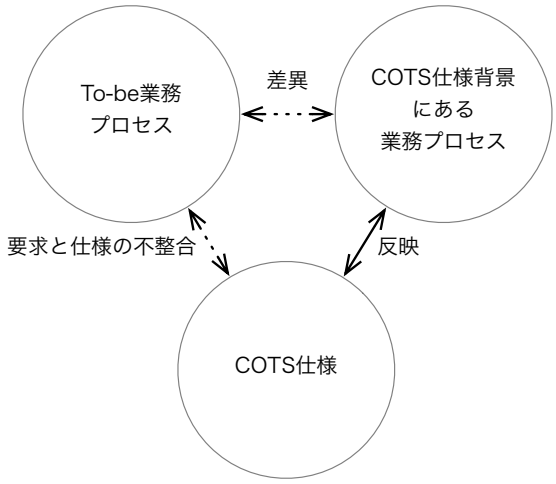


図 4 To-be 業務プロセスと COTS 仕様と COTS 仕様の背景にある業務プロセスの整合性の問題

4.2. COTS 製品仕様の背景にある思想を取り込む

Salesforce SFA は，米国でのインサイドセールスや営業の考えかたが背景にある製品である．その考えかたは「ザ・モデル」として書籍化されている[12]．

日本での営業活動が未だに人依存であるのに対して，米国では，顧客が地理的に離れたところに散在しているという問題から，効率の良い営業プロセスが検討，設計されてきた．Salesforce SFA を使用するということは，この設計された営業プロセスを取り入れるということでもある．製品仕様の背景にある長年検討され，設計された業務プロセスを取り入れることによって，自社の業務プロセス自体を加速的に改善することも期待できる．

しかし一方で，多くの COTS 製品は，製品仕様の背景にある業務プロセスが，導入先の業務プロセスのどの部分にどのように組み込むべきかかが明確に定義されていない場合がある．Salesforce SFA の場合も，手元にあるのは「ザ・モデル」という書籍だけであった，そこから To-be 業務プロセスを一足飛びに導き出すことは難しいように思えた．そこで，その中間段階として，「ザ・モデル」を参照して，今回の顧客の経営的問題を解決するための業務プロセスの基本設計にあたるものを記述してはどうかと考え，以前検討をしていた GSN の適用を試行した．

5. COTS 導入プロジェクトでの GSN の適用

COTS 製品の導入は，それが目的ではない．あくまでも経営ゴール実現が目的であり，COTS はそのための手段である．一方で，COTS 製品は，その設計思想として，経営レベルのゴールが定義されているはずである．そして定義されたゴールを実現するための業務プロセス，そして業務プロセスを支援するために，その製品の仕様が決定されているはずである．しかし，COTS 製品の仕様の背景にある経営ゴールが，自社の経営ゴールと完全に一致するとは限らない．COTS 製品を経営ゴール達成のための手段として取り入れるならば，COTS 製品に想定されているゴールを評価して，自社に組み込む部分を検討し，そして選定する必要がある．

COTS 導入プロジェクトにおけるゴール選定から，業務設計，そして仕様定義までの考えかたと GSN 適用の基本的なアプローチを図 5に示す．まず，COTS が想定するゴールから自社の経営ゴールに組み込む部分を選定する．これに，GSN のゴール定義記述を用いる．次に，ゴールを実現するための戦略を COTS 仕様の背景から定義する．これに，GSN の戦略と戦略の前提条件記述を用いる．さらに，戦略を達成するための方法をサブゴールに展開する．展開したサブゴール構造が業務プロセスの基本設計となる．この基本設計をもとに，PReP モデルによって業務プロセスの詳細設計を行う．PReP モデルでは，設計した業務プロセス上で，それを実行するためのシステムを計画し，システムの要件を定義する．これによって，ゴールからトレーサビリティーの取れたシステム要件までを構造的に導出することができる[4]．

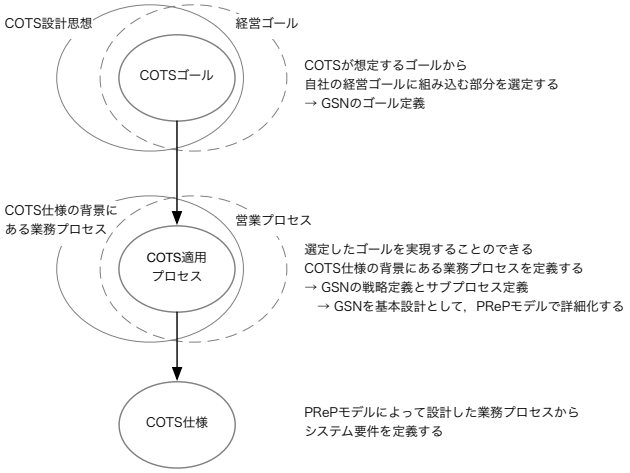


図 5 COTS 導入と GSN による業務プロセス設計の流れ

6. Salesforce SFA での GSN 適用の実際

前項の図 5に示した流れに沿って GSN を適用して，To-be 業務プロセスの基本設計を行なった．基本設計のための入力としては，Salesforce SFA の背景となっている書籍「ザ・モデル」を参照した．基本設計の結果を図 6に示す．GSN による業務基本設計の実例を、順を追って説明する．

6.1. ゴールの設定

組織への Salesforce SFA 導入のゴールとしては，「営業リソースが効率的に活用されている」と設定した．このようなゴールを設定した理由としては，「ザ・モデル」で解説されている営業プロセスの背景に，役割の明確な定義と機能分割による効率的な営業活動が目的であることを読み取ったためである．また，営業活動では，重要な資源は人であり，人という限られたリソースが効率的に活用されている状態は，経営視点から考える営業活動のゴールとして十分性があると考えたからである．

6.2. 戦略の検討

このように定義したゴールを達成するための戦略はいろいろ考えられるが，今回は「営業戦術を外材化し利用する」と定義した．人依存の状態では，効率的に営業活動を行なっている人と，そうでない人がいる．効率的に営業活動を行なっている人の知見を外材化して組織として再利用することができれば，全体としてのリソース効率が上がると考えたからである．

今回のプロジェクトのヒアリング時にも，組織課題として，人依存の状態からの脱却が挙げられていた．ベストプラクティスとしての知見を組織として再利用できる状態にすることによって，特定の人の能力に依存する状態から，組織として管理された状態にレベルを上げることができると考えられる．

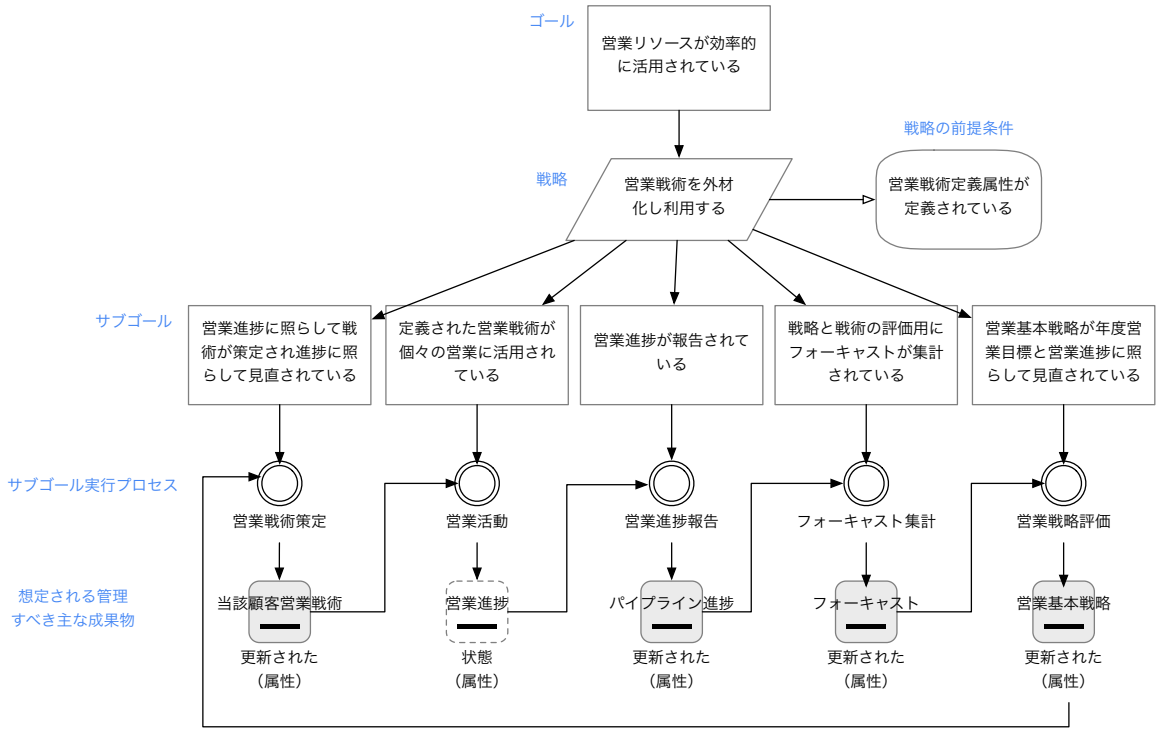


図 6 Salesforce SFA 適用の業務基本設計として GSN を利用した例

GSN での戦略定義では，定義した戦略の前提条件の定義が重要である．どのような状態であれば，営業戦術を外材化し利用できるかが今回の前提条件となる．

一般的にも，知見を外在化するには「何を知見として外材化すべきか」の定義が重要である．能力のある人が，自分がなぜそれをうまくやっているかを適切に言語化できるとは限らない．うまく行なっている行為を構成する独立した要素を言葉として定義すること自体が，技術として重要な部分である．

これに対しても「ザ・モデル」の中で，例えば「（営業）パイプライン」といったような営業活動を管理するための言葉が要所要所で説明されていた．それらの考えかたを整理して，営業フェーズごとに必要となる活動と管理を定義する言葉を整理した．整理と構造化には，図 7に示すように，マインドマップを使用した．

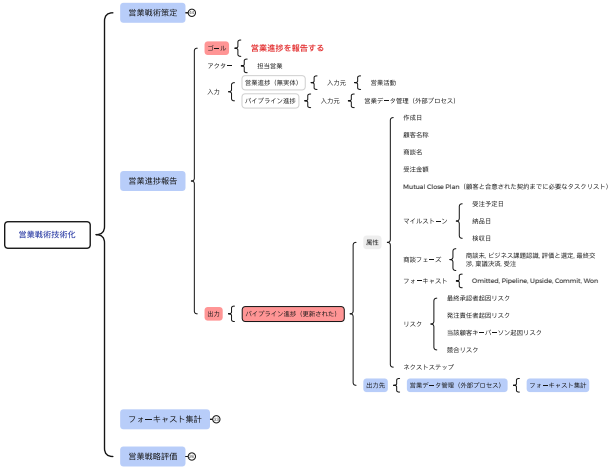


図 7 マインドマップを使った営業活動フェーズごとに使用する概念の整理（部分）

6.3. サブゴールの定義

基本設計の最後に，定義した戦略を実現するためのサブゴールを設計した．サブゴールの設計は，あくまでも設計行為であり，正解はないものである．ただし，ここでも「ザ・モデル」の中で記述されている営業やインサイドセールスといった活度の記述から，戦略の実現に関連しそうな活動を抜き出して組み立てた．

6.4. サブゴールへの業務の紐付け

定義したサブゴールは，例えば「営業進捗に照らして戦術を策定し進捗に照らして見直す」，「営業進捗を報告する」といったように，それぞれがひとまとまりの活動である．業務プロセスをモデル化するための考えかたをまとめたPReP モデルでは，経営ゴール達成のリスクを管理するためのひとまとまりの活動を一つの業務として定義している[4]．GSN で定義したサブゴールも，検討された戦略を実行するひとまとまりの活動であり，PReP モデルで定義する一つの業務に対応すると考えることができる．この部分は，GSN とPReP モデルとをシームレスに繋げる上で重要な点である．そこで，サブゴールの下に，それぞれひとつずつの業務を紐付けた．

さらに，各業務から出力される主要な成果物を定義し，定義した成果物と各業務との入出力関係から，基本となる業務間の構造を定義した．業務とそこから出力される主な成果物（ひとつとは限らない）によって業務の基本関係を記述することを，PReP モデルでは業務鳥瞰図の作成と呼んでいる．対象スコープに含まれる業務とそれらの関係を俯瞰するという意味である．ここまでの段階で，GSN を使ってゴールの定義からPReP モデルの鳥瞰図までを展開することができた．

6.5. To-be 基本業務プロセスの構文設計

GSN を用いた業務基本設計は，業務の静的な関係構造のモデルである．設計では，構造的モデルに加えて構文的なモデル（時間軸での関係）も必要である．そこで，それぞれの業務に登場すると考えられるアクターと，各業務プロセスの構文的な関係を図 8のように定義した．図6で定義した「営業戦術策定」は週次で駆動するとした．商談後の「営業進捗報告」は日次で行い，その結果が「フォーキャスト集計」される．集計された結果をもとに営業戦術が「営業戦術策定」の中で週次で見直される．月次で集計されたフォーキャストをもとに，営業目標に対する進捗が評価され，全体の戦略を見直すための「営業戦略評価」が会議体として開催される．このような定義はあくまでもひとつの設計であり，様々なバリエーションが考えられる．ここで設計した基本プロセスが妥当であるかは，導入先の組織の判断に委ねられる．

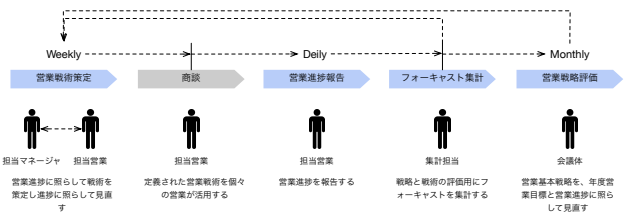


図 8 To-be 業務プロセスの基本設計

6.6. To-be プロセスの詳細設計

PReP モデルの鳥瞰図と基本的な構文設計ができあがれば，定義した業務ごとに詳細設計を行うことができる．鳥瞰図では，それぞれの業務への入出力が定義されているので，それらを手がかりに，それぞれの業務プロセスを設計することができる．図 9は GSN で定義した業務プロセスを PReP モデルで設計した例である．PReP モデルでは，設計した業務プロセス上でシステム要件を定義するが，COTS 製品適用の場合は，COTS 製品で提供されている機能を PReP モデルの業務プロセス上にマッピングすることによって，COTS 製品でカバーできる範囲，あたらに開発が必要な部分などを特定し定義することができる．

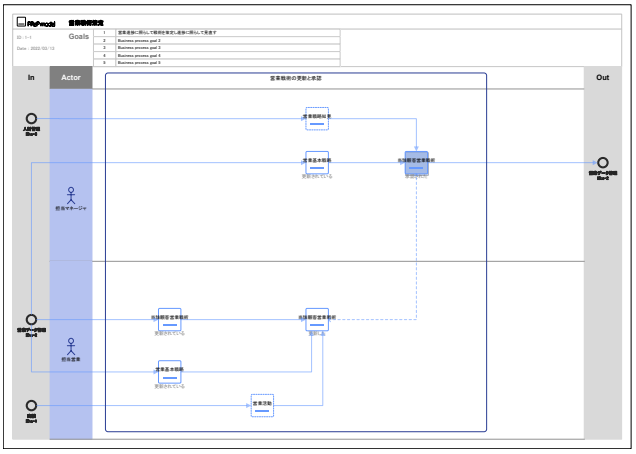


図 9 PReP モデルによる詳細設計の例

7. 基本設計までにかかった時間・期間の比較

逆 V モデルと GSN それぞれに対して，業務プロセスの To-be 設計までにかかった期間と実働工数を表 1に比較する．逆 V モデルでの時間及び期間は，今回の SFA 導入ケースと同規模の業務を想定した場合の経験からの見積もり値である．また，期間はその長さを比較するため，休日も含めた週 7 日で計算した．

逆 V モデルでは，ユーザ企業からの参加が必要であるため，分析や設計のためのワークショップを毎日開催することは難しいため，実質的に週に1回の開催サイクルで進める場合が多い．そのため，期間としては長くかかってしまう．

今回の GSN を適用した設計にかかった期間としては，ヒアリングに 1 日，「ザ・モデル」から Salesforce SFA の背景にある設計思想の把握から基本設計までが 2 日．基本設計から PReP モデルによる業務プロセス詳細設計までが 1 日（実働は 3 時間）であった．

表 1 To-be 業務プロセス設計までの時間および期間比較

方法	実働（日）	期間（日）
逆 V モデル	16	56
GSN	4	4
（GSN/逆 V モデル）%	25	7

8. まとめ

今回の結果を見ると，設計思想が読み取れる COTS 製品の場合は，GSN を利用して，その思想からトップダウンに To-be 業務プロセス設計を進めることによって，時間を大幅に短縮できる可能性があることがわかった．さらには，ユーザ企業からの参加者の時間拘束を行わずに，To-be 業務の設計までを進めることができるため，コスト的にも大きな期待が持てる．

一方で，逆 V モデルの場合は，ユーザ企業からの参加者との協働によって現状の分析から設計までを進める．そのため，To-be 設計が終わったと同時に，ユーザ企業側の主要メンバーの共通理解の獲得と合意形成も完了する．GSN を使ったトップダウン設計では，To-be 設計までが数日で済んだとしても，その結果を提示し，理解と合意を形成する工数と，その段階で差し戻しに合うリスクを考えると，1 回の設計ライフサイクルの単純な比較をするのは危険であるとも考えられる．

COTS 製品を開発する側への指針としては，総花的に機能を盛り込むのではなく，業務目的とその実行プロセスの観点からの明確な設計思想に裏付けされたソフトウェア製品を開発することが，ユーザ側の業務に貢献する価値を提供するという観点では重要であると考えられる．また，その設計思想を外材化し，できれば，想定しているゴールとそれを実現する際の業務のあり方を再利用できる形式で記述するところまでを COTS 製品の開発として捉えることによって，ユーザ側での利用価値が上がるものと考えることができる．

参考文献

[1] 田中康, 渡辺薫, 超上流工程における業務プロセスのモデル化 -PReP モデルの BPM への適用-, ソフトウェアシンポジウム 2013, 2013

[2] Tim Kelly, Rob Weaver, The Goal Structuring Notation - A Safety Argument Notation, Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases, 2004

[3] 日経コンピュータ, 2018 年 3 月 1 日号 pp.26-39

[4] 田中康, PReP MODEL 現実世界をデザインする, Amazon 出版, 2022

[5] 田中康, 飯田元, 松本健一, 成果物間の関連に着目した開発プロセスモデル PReP, 情報処理学会論文誌, 2004 年「社会人学生」論文, 2005

[6] Victor Basili, G. Caldiera, Dieter Rombach, Goal, Question, Metric Paradigm, Encyclopedia of Software Engineering, Vol.1, pp. 528-532, 1994.

[7] Victor Basili, Adam Trendowicz, Martin Kowalczyk, Jens Heidrich, Carolyn Seaman, Jürgen Münch, Dieter Rombach 著, 鷲崎弘宣, 小堀貴信, 新谷勝利, 松岡秀樹 監訳, 早稲田大学グローバルソフトウェアエンジニアリング研究所ゴール指向経営研究会 訳, ゴール＆ストラテジ入門：残念なシステムの無くし方（GQM+Strategies）, オーム社, 2015.

[8] 新谷勝利, 平林大典, 定量的な目標管理手法のウ普及活動の展開～組織目標達成と IT 導入の整合性を図る「GQM+Strategies® 」の活用～, SEC journal, No.33, 2013.

[9] Kaplan R.S, Norton D.P, "The balanced scorecard: measures that drive performance", Harvard Business Review, Jan-Feb, pp71-80, 1992.

[10] 鷲崎弘宣, 新谷勝利, 青木耀平, 志村千万輝, 野村典文, GQM+Starategies による組織目標と戦略の整合化及び目標定量管理の実践と拡張 -SEC WG 及び早稲田大学ゴール指向経営研究会の活動より-, SEC journal Vol.12 No.4 Mar. 2017.

[11] Toshinori Takai, Katsutoshi Shintani, Hideki Andoh, Hironori Washizaki, Continuous modeling supports from business analysis to systems engineering in IoT development, 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI), 1-15 Sept. 2020.

[12] 福田康隆, ザ・モデル, 翔泳社, 2019

機能共鳴分析手法 FRAM による遠隔授業のシナリオ分析の視覚化

日下部 茂
長崎県立大学

小佐古巴菜[1]
長崎県立大学

有田 大作
長崎県立大学

要旨

教育への情報処理技術の導入が進む事例も増え，教育に関する IT インフラや教育者側の IT 利活用能力の差が教育効果に影響を与える懸念も生じている．我々は，このような懸念の解消のため，ソフトウェア工学の知見の援用を試みている．ソフトウェア工学のうち特にプロセスとモデリング技術により授業 IT 化の効果的実現や継続的改善を体系的に支援することを目指している．計算機に閉じたソフトウェアの実行と異なり授業の実施には必ずしも数理的，定量的な評価が適するとは限らない人や組織の振舞いが含まれるため質的アプローチの活用を試みた．特に遠隔授業に焦点を当て質的アプローチの一つであるインタビューを行ったところ，同一授業であっても，学生により評価が異なることがあった．そのため，モデリングには変動を前提に失敗だけでなく成功にも着目するレジリエンスエンジニアリングの手法 FRAM を用いた．インタビュー結果もふまえた FRAM の図式表現により相互作用を視覚化し遠隔授業のシナリオ分析を行い，改善のため検討を行った．

1. はじめに

1.1. 背景

近年は EdTech という造語もできるほど教育と情報処理技術の融合が進んでいる．教育者側は，個人レベルでは自らプログラミングすることはなくとも様々な IT 機器やシステム機能などを活用し授業を IT 化し実践・改善する能力が，また組織レベルではその効果的支援能力が重要となっている．この傾向が続けば，教育者側の能力や成熟度の差により，教育成果の格差や，IT 化の作業による残業時間の不適切な増加などが生じる懸念がある．

例えば，コロナ禍により筆者らの所属する組織でも対面方式での教育が制限され，非対面型授業の準備を急遽行うなどした．その際は IT 技術の活用が必須であったが IT サブシステムを利用・連携させた授業の非対面化の実現や評価において様々な問題に直面した．例えば，要求を実現するサブシステム選定といったアーキテクチャレベルの問題，名寄せの必要性のような要求・設計段階の問題，Google フォームのコピー＆ペーストのような実装段階の問題，想定した動作の確認のような検証段階の問題などを経験し，ソフトウェア開発で用いられる手法を導入することが有効ではないかとの仮説を考えた．

大学教育の非対面化に IT を活用する際，個人レベルの問題に加え，非情報系教員の IT の知識・スキルのばらつき，学生視点不在の不統一なインターフェース，社会情勢や技術革新などの状況変化への対応といった問題に対し，継続的で体系立った組織的支援の必要性も感じた．所属組織に，地方自治体関連組織から，GIGA スクール構想における IT 支援員の教育プログラムの検討依頼があった際に，同様の必要性をさらに強く感じた．

同時に，計算機でのソフトウェアの実行と，授業の実施の相違点から，慣習的なソフトウェア工学の限界も感じた．そのため質的アプローチやレジリエンスエンジニアリングといった手法との融合の検討に思い至った．

1.2. アプローチ

前述のような背景の下，我々は特にプロセスとモデリングに着目し，授業の効果的な IT 化や改善を体系的に支援する方法論に取り組んだ．IT 化された授業もソフトウェアと類似した人工物とみなすことで，その実現や改善にソフトウェア工学の知見を活用できる．例えば，ソフトウェアの品質は開発プロセスで決まるという考えの下提唱された能力成熟度モデル統合 CMMI といったプロセス改善モデル[1]や、ソフトウェアの役割や要求の変化に対応するアジャイルのような取組み[2]は参考になると考える．

プロセスという観点で系統立てて考えると，要求や設計といったものは共通する部分も大きいが，検証はソフトウェアプロセスと授業 IT 化のプロセスに違いが大きいと考える．授業の環境や教材を IT 化しても，人や組織の振舞いに関する部分の検証はソフトウェアのテスト技法といった手法をそのまま適用することはできない．人や組織

[1] 2022 年 3 月長崎県立大学情報システム学部卒業

に関連する部分は，個体間での特性の差異，同一個体での動作の不確実性などにより工学的な量的アプローチでは限界もあり，質的アプローチの導入を検討する．

改善に関して我々の組織でも従来から授業の終了時期に学生が回答する授業アンケートを行っているが，尺度評価が中心で，その評価になった理由や，具体的改善点は直接的にはわからない．評価結果の理由の分析や，改善の検討や実践を行うフレームワークとして，プロセスのフレームワークをベースに，質的アプローチの一つであるインタビューをとりいれ，シナリオ分析[3]の活用を試みる．遠隔授業受講者に個別アンケートを行い，インタビューとモデリングを用いて評価結果の理由も含めシナリオの分析を行い，問題点の混入や除去，ベストプラクティスの導入などをプロセスを関連付けながら検討する．対面型の授業に比べ，コロナ禍で広く普及した非面接型の遠隔授業では学生の反応を直接的に観察しづらく分析が難しいと考え，ここでは遠隔授業に焦点を当てる．

ソフトウェア工学で観点に応じたモデリングが重要な役割を演じたのと同様に，授業 IT 化の取り組みでもモデリングが重要と考える．授業の実施に技術的要素だけでなく多様なステークホルダや環境も影響を持ちうることを考えモデリングはシステムズエンジニアリングの手法を用いる．大規模・複雑化するシステムの抽象化，多様な利害関係者間の合意形成といった観点に加え，非決定的な振る舞いや結果の分析の必要性も考え，ここではレジリエンスエンジニアリングの手法である機能共鳴分析法 FRAM[4]を用いたモデリング[5]に取り組んだ．

授業 IT 化の知見を効果的に共有できれば各教員だけでなく組織的にも有用である．ソフトウェア工学では，必ずしも明確な原理や理論の裏付けがない知見でも，現実制約の下で有用で経験的にベストプラクティスとして扱えれば最大限活用する．今回は FRAM を用い，遠隔授業に関する知見を，プロセス改善などと関連付け可能な形で視覚化し，知見の有効活用を促進することを目指す．

本稿の構成は以下の通りである．第 2 節で，ソフトウェア開発と遠隔講義準備のプロセス例について比較する．第 3 節では，質的アプローチについて述べ，個別インタビューの内容と得られたデータについて説明する．第 4 節では，機能共鳴分析手法 FRAM による遠隔授業のモデリング事例について論じる．第 5 節ではまとめを述べる．

2. プロセス

IT システムやそのコンポーネントの開発には系統的で工学的な取り組みがなされるのと同様に，授業の IT 化やその改善にも，系統的で工学的な取り組みがなされることを想定する．その一環として，ここではプロセスのフレームワークの導入を検討する．

2.1. ソフトウェア開発

ソフトウェア開発のプロセスでは，例えば図 1 のように，主要なフェーズとして以下のようなものが考えられる．

- 要求定義
- アーキテクチャ設計
- 詳細設計
- 実装
- 単体テスト
- 結合テスト
- 統合テスト

これらのフェーズの入出力である作業成果物には以下のようなものがある．

- 企画書
- 要求仕様書
- アーキテクチャ設計書
- 詳細設計書
- 総合，結合，単体テスト仕様書

また，プロジェクトマネジメント，品質保証，リスクマネジメント，構成管理，変更管理，開発環境管理といった，サポートプロセスもある．

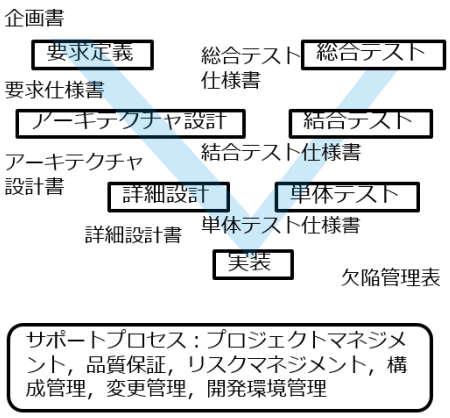


図 1 ソフトウェア開発プロセスの例

2.2. 授業の準備と実施

授業の準備と実施のプロセスやその改善の準備と実施にあたって，筆者らの組織では，シラバスの作成と，学生が回答する授業評価アンケートの結果に対するフィードバックが組織的に実施されている．それら以外の部分は，授業の準備や実施，評価の解釈も含め，各教員の裁

量に任されている(図 2). 著者らの組織では包括的な LMS(Leaning Management System, 学習管理システム)は未導入で, 授業に必要な機能や品質の要求を, 汎用の技術やプラットフォームを活用して各教員が IT 化する. そのような部分に関する可視化や, 系統的で共有可能なベストプラクティス, ガイドラインやテンプレートなどを研究することは有用と考える.

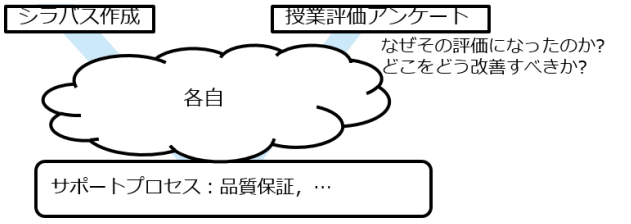


図 2 不明な部分がある授業プロセスの例

教員が作成するシラバスの項目には, 授業概要とテーマ, 到達目標, 授業計画, 成績評価の基準, 成績評価の方法, テキスト, 参考文献, 科目のキーワード, 授業の特徴, 関連科目, 履修上の注意等(履修条件等), といった項目がある. 授業の IT 化においては, これらを要求の一部として考え, ソフトウェア工学などの知見の活用を試みる. 例えば筆者は, シラバス記載の成績評価方法と, 収集可能な電子データの不整合により手戻りをした経験があるが, 同様の問題の再発防止にはデータフローダイアグラムの作成と分析などが有効と考えられる.

授業終了時期に学生が授業評価アンケートを記入しその集計結果を教員は受け取るが, 評価は尺度中心でその評価になった理由は推測を必要とする. そのような推測や改善の具体的取り組みは各自の試行錯誤となる上, その結果がわかるのは次期の授業後である. このような問題に関する知見を見える化し共有できれば各教員だけでなく組織的にも有用である. ソフトウェア工学では, 必ずしも明確な原理や理論に裏付けられない知見であっても現実制約の下で有用と経験的にベストプラクティスとして扱えれば最大限活用する. 同様に授業に関する知見を形式知化し活用可能とすることを目指す.

授業 IT 化をプロセスの観点もふまえて分析するにあたり, 前述のように検証の再検討を行った. 検証の観点として, 教育者側が学生の成績を評価する側の観点もある一方, 授業評価アンケートのようにシステムを利用する学生側の視点からの検証という視点もある. IT 化された授業をソフトウェアのような人工物とみなしても, 人や組織に関連する部分は, 個体間での特性の差異, 同一個体での動作の不確実性などで工学的な量的アプローチでは限界もあり, 検証に質的アプローチを用いることを試みた.

3. 質的アプローチ

前述のように, 筆者らの組織では, シラバスの作成と, 学生が回答する授業評価アンケートの結果に対するフィードバック活動が行われている. 授業評価アンケートには自由記述欄もあるが, 尺度での回答による定量的な評価が中心であるが, 定量的な評価結果だけでなく, 評価結果につながる原因やその混入過程が把握できないと改善は難しい. 本研究では, 授業準備プロセスと評価結果に関する仮説を生成するために, 仮説生成に適するとされる質的研究のアプローチを取り入れた. 今回は特に, 尺度評価中心の授業評価アンケートを補うものとして, 学生側の視点からのインタビューを行った[6].

臨床心理学, 看護学, 社会学でよく用いられる質的研究では, 具体的な事例を重視し, 数値でなく文章や語りに解釈を与える. 当事者の経験と生活世界を客観的に説明・理解することと, 新たな理論を構築することを目的としており, 学生視点での評価に解釈を与えるのに適していると考える. 具体的には, インタビュー, ケーススタディ, グラウンデッド・セオリー・アプローチなどの手法があり, それらを組み合わせるのが望ましいとされている. 分析の手順にはフィールドノートやコード化, カテゴリー化といったものがある. 本研究では, 個別インタビューに後述のようにレジリエンスエンジニアリングの手法を組み合わせる.

3.1. 対象事例

本研究では遠隔授業に焦点を当て, 学生の満足度や授業の達成度が高まる仕組みを分析することを念頭に, 質的調査であるインタビューを行った. インタビュー調査の対象者を, 著者らの所属大学の情報システム学部情報システム学科 4 年生の三名と 3 年生の三名として, 今まで受講してきた遠隔授業に関するインタビューを行った. できるだけ本音を語ってもらうため, インタビュアーも学生とし, 半構造化インタビューを行った. 半構造化インタビューでは, インタビューを行う前に目的に合わせた大まかな質問を用意しておき, 対象者の回答に応じて質問内容を重ねたり, 深掘りしたりするインタビュー形式をとる. まず 4 年生に対してインタビューを行い, 質問内容の一部を変えて 3 年生に対してインタビューを行った.

3.2. 第 1 回インタビュー調査

4 年生を対象者に, 今まで受講してきた遠隔授業について, 以下の 5 項目についてインタビューを行った.

1. 遠隔授業の配信に用いたツールは何か
2. 質問対応に用いた機能，ツールは何か
3. 対面形式と比較して遠隔授業のメリットは何か
4. 対面形式と比較して遠隔授業のデメリットは何か
5. その遠隔授業を対面形式で受けたと仮定した時，どちらの形式の方が高い理解度と予想できるか

これらに対する4年生からの回答を以下に挙げる．

項目1. 遠隔授業の配信に用いたツールは何か

- LMSとして一部LiveCampus[7] を使用した授業もあったが，大半はGoogle Classroom が用いられた．
- 遠隔会議システムとして一部 Google Meet を使用した授業もあったが，大半は Zoom が用いられた．
- 課題の配信や Zoom リンクの共有には電子メールが用いられた．

項目2. 質問対応に用いた機能，ツールは何か

- 一部授業は電子メールや Zoom 利用だったが大半はGoogle Classroom の限定公開コメント機能を用いていた．授業時間内は教員が迅速に対応していた．
- Zoom を用いた授業では，授業時間内に教員へ質問をすると画面共有を求められ，他の受講生に画面を見られたくないと感じた．

項目3. 対面授業と比較し遠隔授業のメリットは何か

- 自分の予定や都合に合わせて受講できる．
- 自分のペースで，本来の拘束時間である 90 分に縛られず受講を進めることができる．
- 動画や資料をいつでも見返すことができる．
- 教員へ質問をする前に自分で調べるようになり，対面授業の時よりも自力でできていると感じられた．

項目4. 対面授業と比較し遠隔授業のデメリットは何か

- 教員へ質問しづらい
- 資料が授業スライドのみだと理解できず，レポートが難しく感じた．
- レポートへのフィードバックや，他の受講生の回答例が示されず，自分の回答が正しいのかわからない．
- 一つ一つのレポートや制作物，授業本体の評価方法が明示されず，授業評価が公開されるまで単位を取得できているか不安に感じた．

項目5. その遠隔授業を対面形式で受けたと仮定した時，どちらの形式の方が高い理解度と予想できるか

- 遠隔授業の良い点は，授業のために大学まで行く必要がなく自分のペースで受けられる点，悪い点は対面授業と比べて教員へ質問しづらく感じてしまい理解度が落ちてしまう．
- 座学系の講義は遠隔でも特に問題なく受講することができた．一方で演習がメインの講義は遠隔では理解度が低かったり，ソフトウェアのインストール環境が整わなかったりして対面授業の方が効果的と感じた．

項目 5 に関しては，同じ授業であっても人により回答が異なるケースが複数あった．良い点に着目しながら分析するという方針の下，良い授業につながったと感じたポイントとして挙がったものが，以下の 3 つである．

1. 不明点や困ったことを教員へ質問しやすいか
2. レポートの有無や難易度，提出頻度が適切であるか
3. 制作物の評価基準や最終評価の方法が事前に明示されているか

上記のポイントのうち，最初の，質問のしやすさは動的な側面を持つ．その点を深掘りすることを含め，質問項目を改版して第2回のインタビューを3年生の学生に行った．

3.3. 第 2 回インタビュー調査

上記三つのポイントが本当に良い授業に繋がるものかを確かめるべく，3 年生へのインタビューでは質問項目を以下の 6 項目へ変更した．

1. 遠隔授業の配信に用いたツールは何か
2. レポートの有無や難易度，提出頻度は適切だったか
3. 受講期間中，教員へ質問をしたことがあるか．わからないことがあったが質問をしなかった場合，どうすれば質問ができていたか
4. 受講中困ることはあったか
5. 制作物やレポートの評価方法は示されていたか
6. 最終評価の方法は示されていたか

3年生からの回答を以下に挙げる．

項目1. 遠隔授業の配信に用いたツールは何か

- LMSとして Google Classroom が用いられた．
- 大学に 15 回分の授業資料をまとめた冊子を置いておき，初回授業日までに学生に各自取りに来させた授業もあった．その際，電子メールで大学まで取りに来るよう連絡がきた．

項目2. レポートの有無,難易度,頻度は適切だったか

- レポート提出を求められる授業が大抵であったが，提出頻度は授業によって様々であった．授業スライドが字面ばかりであると理解度があがらず，レポートも難しく感じてしまう．
- 授業資料だけでは理解が不十分としても，授業内容に沿ったレポートを解くことで理解度があがった．
- 提出頻度が必ずしも理解度と関係してはいない．

項目3. 受講期間中，教員へ質問をしたことがあるか．わからないことがあったが質問をしなかった場合，どうすれば質問ができていたか

- 受講期間中，教員へ質問をしたことがある学生は少

なかった．不明点や困ったことがあった時は，解決しないままにしている学生や，友人と話し合って進めている学生が多いようだ．

・ 友人と話し合っても解決できず，解決できなければレポートや成果物に大きな支障が出てしまう場合には教員へ質問をした．その際，Google Classroom の限定コメント機能を用いていた．

なぜ教員へ質問することをためらってしまうのか，深掘りしたところ，学生からは以下の声が挙がった．

(a) 質問に対して教員からの返答が遅い．
(b) メッセージのやり取り回数が多く面倒に感じる．
(c) 自分の質問内容が他の受講生にも公開される場合，恥ずかしいと感じてしまう．
(d) プログラミング言語や HTML，CSS を扱う授業などではソースコードを用いて質問することもあり，言語化することが難しかった．

項目4.　制作物やレポートの評価法は示されていたか

・ 制作物やレポートの評価方法が示されていない授業が多く，自分の回答が正しいのか不安に感じた．

項目5.　最終評価の方法は示されていたか

・ 最終評価の方法は示されている授業と示されていない授業があったが，示されている授業の方が学生は良い評価を獲得できるように取り組むことができ意欲があがると感じていたようだ．

項目6.　その他

・ 初回のみ対面で実施され，授業内で使用するソフトウェアの動作環境設定ができたことで，その後遠隔でもスムーズに受講できた
・ 動画の端に顔アイコンがあり，教員の話している映像が映っていたので対面で受講しているのと変わらない緊張感を持つことができた

4. モデリング

質的アプローチで得られた，ナラティブなデータの分析の手順にはコード化，カテゴリー化といったものがあるが，本研究では，個別インタビューにレジリエンスエンジニアリングのモデリングを組み合わせる．具体的には機能共鳴分析手法 FRAM によるモデリングで，成功パターンに着目したシナリオ分析について論じる．今回は遠隔講義について，特に 3.2 節の最後に述べた 3 つのポイントのうちの一つで，第 2 回インタビューの項目 3 に関する，質問のシナリオ分析のモデリングを行う．学生のインタビュー結果をもとに FRAM でシナリオ分析を行うことで，学生視点のストーリーの分析につながると考える[6]．文献[9]と同様に，オリジナルの FRAM に追加の記述ルールや便宜的補助線など追加して，抽象的な機能や活動と，それらの間のつながりを記述・分析する．

4.1. 基本シナリオのモデル

まず準備として，遠隔でなく，教員と学生が場を共有した授業の FRAM モデルを作成した(図 4 参照)．横に 2 列に並んでいる六角形のうち上側の並びが教師の活動，下の並びが学生の活動をモデル化したものである．教師はシラバスに基づき学習内容を説明し，試験でなく課題レポートで成績を評価する前提で課題を出す．二回目以降の授業は最初に前回の課題の説明を行う．学生側は授業の説明を受けながら学習を進め，出された課題のレポートを提出する．二回目以降の授業は最初に課題の解答の解説を聞く．FRAM のモデリングツール[10]上では活動主体ごとに活動の六角形の色を変える，ツールから生成した画像に補助線を引くなどすることもある．

今回，質問に焦点を当てたモデリングを行うため，初期 FRAM モデルに質問のアクティビティを追加したもの

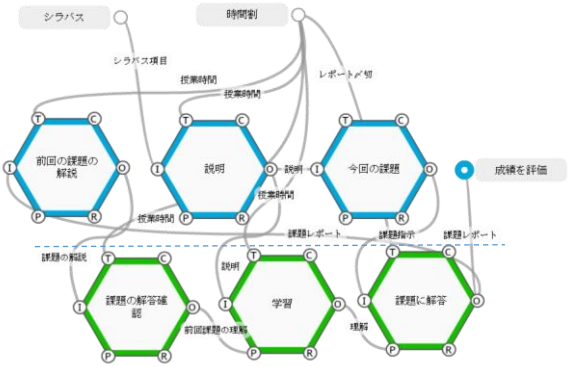


図 4　授業の初期 FRAM モデル例

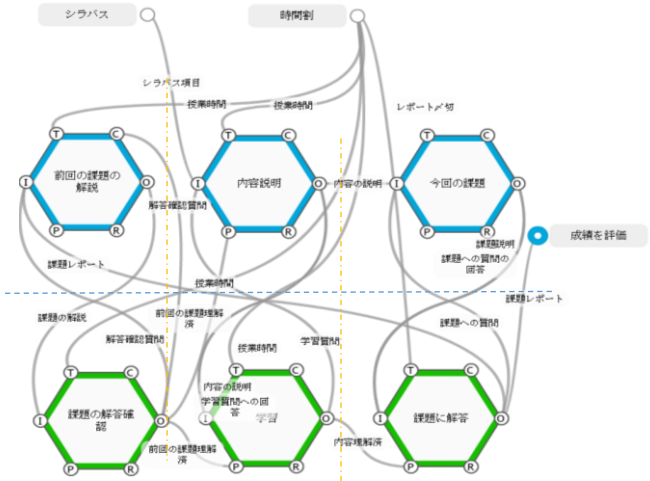


図 3　質問がある授業の FRAM モデル例

を図 3 に示す. 教師側の内容説明を受け, 学生側の学習の途中で質問が出たら, 教師側の内容説明の途中で回答するといったシナリオを想定している. 説明と関連する質問, その回答は同一時間枠で同期的になされることを想定する場合, モデル図上での配置に配慮するとともに補助線などを入れて明示するとわかりやすい.

4.2. 遠隔講義シナリオのモデル化

次に, 同期型の遠隔講義シナリオの FRAM モデル例を図 5 に示す. この例では, 時間割で決められた時間に資料提示の機能が起動されるモデルになっている. 補助線は入れていないが中段に資料提示に関連する機能を並べており, この部分が遠隔講義を支援する機能やアクティビティに対応する. 遠隔講義には何らかの IT 機能が必要で, その部分を,「質問しやすく」詳細化していくことを考える. 著者らの所属大学の場合, これらのシナリオ中の機能は, Zoom や Google Meet のような遠隔会議システムか, Google Classroom や LiveCampus のような教育用コンテンツ管理システムの機能に対応付けられる.

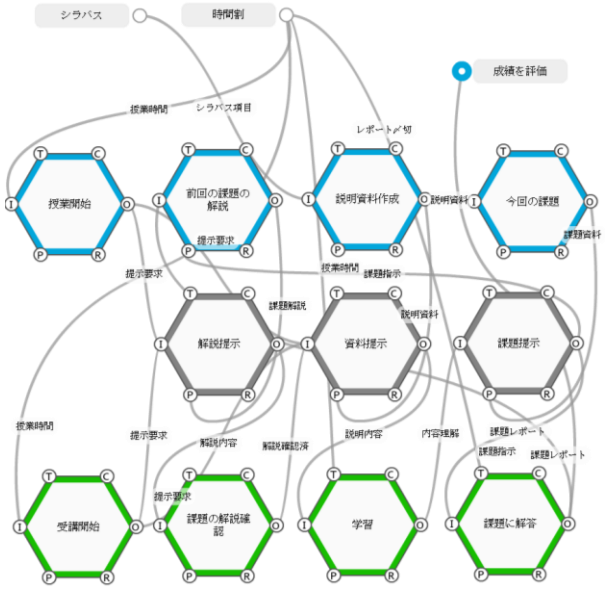


図 5 遠隔講義の FRAM モデル例

上記の遠隔講義シナリオの FRAM モデルをベースに質問アクティビティのある遠隔講義シナリオの FRAM モデル例を図 6 に示す. 先のモデルに, 学生の質問作成, 教員の回答作成に加え, システムでの質問提示と回答提示の機能を追加した. このようなモデルをベースに, 教員側のシナリオだけでなく, アンケート結果などから把握した学生のシナリオに基づいた分析を行う. 以下に, ソフトウェア開発プロセスのアーキテクチャ分析に相当するような例を示す.

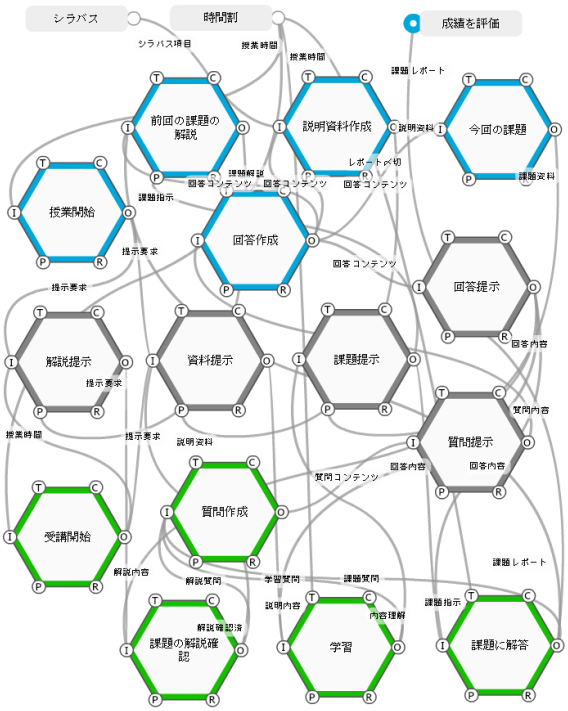


図 6 質問のある遠隔講義の FRAM モデル例

学生が資料を見ながら学習している最中に何か疑問が生じたとする. 頭の中の疑問を他者が読める質問とし教員にそれを送る. 教員側では質問に気づいた後, 質問を読んで回答を作成し学生に送る. 学生は回答に気づいた後, 回答を読んで学習を続ける. このようなシナリオは, 図 6 では,「資料提示」「学習」「質問作成」「質問提示」「回答作成」「回答提示」「学習」のループに相当する. このような抽象的なモデルの部分に対し, 授業の特性, 使うソフトウェアサブシステムの選択によって, 実際の授業での必要な操作ステップ数や利用サブシステムの切り替え操作などはバリエーションがあり, 学生視点の「質問のしやすさ」は異なる可能性がある.

- Google Classroom で教員へ質問する場合は, Google Classroom の限定コメントを開き, 疑問点を文章化する. 文章を教員へ送信し, 教員が回答後に限定コメントの返信を読む.
- Google Meet や Zoom のような会議システムを利用した質問の場合は, チャット機能でのやり取りになるかもしれないし, 挙手機能を使ったのち, 画面と音声を使ってやり取りをするかもしれない.
- 電子メールを用いた質問手順では, ユーザは疑問が発生すると, メーラを開き, 教員のメールアドレスを打

ち込む. 件名を記入し, 本文に自分の名前を明示するなどのマナーに従い, 疑問点を文章化する. そしてメール本文を教員へ送信し, 返信を受け取る.

疑問点の文章化, 教員へ送信, 教員からの回答受け取りといったアクティビティを含む質問シナリオの学生視点の体験に異なり得る. シラバス作成と授業評価アンケートをつなぐ際のアーキテクチャ設計の検討として, 学生がどのサブシステムを使ってどのように質問をし得るか, その際の学生の体験はどうなるかといった検討を, インタビュー結果と FRAM モデルを用いて行うことができる.

FRAM では各機能・アクティビティの関係を 6 つのアスペクトを用いながらモデル化する. 例えば, 回答を得るまでの時間が短いのがよいとされる点については, FRAM の時間のアスペクトを使ってモデル化することが考えられる. また, 資源のアスペクトを用いることによって受講のしやすさといった観点でもモデル化することも考えられる.

現段階では学生を対象者としたインタビュー調査しかできていないが, 今後は教員にもインタビュー調査を行い, 教員が想定していた授業展開の FRAM モデルと, 実際に学生が受講してきた FRAM モデルを比較することも考えられる. 例えば教員は「資料を読み進めてからレポートに回答してほしい」と考えていたが, 学生は資料を見ずにレポートに回答できる, といった教員の想定と学生の実際の行動のギャップなども分析できると考える.

5. おわりに

授業の IT 化の効果的実現と継続的改善を体系的に支援する仕組みを開発する取り組みについて述べた. 筆者らの組織の改善関連の活動にはシラバス作成と授業評価アンケートがあるが, その間をつなぐものとしてソフトウェア開発プロセスの考えが有効と考えた. ただし授業関連の活動には必ずしも数理的, 定量的な評価が適するとは限らない人や組織の振舞いが含まれるため, 質的アプローチとレジリエンスエンジニアリングのモデリング手法の組み合わせを導入した. 遠隔授業のインタビューで, 同一の対象に対しても学生により評価が異なることがあった. そのため, 変動を前提に失敗より成功に着目するモデリング手法の FRAM の図式表現を用いて遠隔授業のシナリオ分析を行った. FRAM にはテキスト表現と図式表現があり, 図式表現による視覚的で直観的な理解の促進だけでなくテキスト記述のレビューなども可能である. インタビュー結果を用いた FRAM でのシナリオ分析はシステムアーキテクチャの検討や設計の検討などに対応づけ可能で, 授業の IT 化の実施や改善に有効と考える.

現段階では学生を対象者としたインタビュー調査しかできておらず, 今後は教員にもインタビュー調査を行い, 教員が想定していた授業展開の FRAM モデルと, 実際に学生が受講してきた FRAM モデルを比較することも考えられる. またより高い抽象度の教育的社会技術システムの視点でのモデル化も考えられる[11]. FRAM については, モデル構築や分析, シミュレーションなどを支援するツールも出ており[12][13], ソフトウェア開発も含め FRAM の活用法について引続き研究を行う予定である.

参考文献

[1] CMMI. ISACA. https://cmmiinstitute.com, (accessed 2022-03-21)

[2] Manifesto for Agile Software Development, https://agilemanifesto.org, (accessed 2022-03-21)

[3] J. M. Carrol, Making Use --- Scenario-Based Design of Human-Computer Interactions, MIT press, 2000

[4] E. 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

[5] 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

[6] 小佐古巴菜, 学生へのインタビュー調査による効果的な遠隔授業に関する研究, 長崎県立大学卒業論文, 2022 年 1 月

[7] 学務支援ソリューション『LiveCampus』シリーズ, https://www.nttdata-kyushu.co.jp/solution/education/001.html, (accessed 2022-03-21)

[8] 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

[9] 日下部茂, 機能共鳴分析法の図式表現を用いた視覚的シナリオベース設計支援, SEA SS2021

[10] FMV FRAM Model Visualizer, https://functionalresonance.com/the%20fram%20model%20visualiser/, (accessed 2022-03-21)

[11] H. Slim, S. Nadeau & F. Morency, The application of the Functional Resonance Analysis Method (FRAM) to evaluate factors affecting times-to-completion and graduation in graduate studies, In Gesellschaft für Arbeitswissenschaft (Francfort, Allemagne, Fev. 21-23, 2018)

[12] R. Patriarca, G. D. Gravio, F. Costantino, myFRAM : an open tool to support FRAM. 12th FRAMily meeting and workshop, pp.11-13, 2018.

[13] FRAM Model Interpreter,functionalresonance.com/the-fram-model-interpreter.html (22-3-21 確認)

仕組み提案：”技術を持つ人”と”その技術を欲している人”をつなぐマッチングソフトはつくれるだろうか？

本多　慶匡
SEA Hokkaido
b00kw0rld@icloud.com

要旨

”ソフトウェア技術を持つ人”と”その技術を欲している人”をつなぐ仕組みを，マッチングアプリを応用して作れるのではなかろうか？

“出来る人”と“欲しい人をつなぐ”．“出来ているもの”と“欲しい人を”つなぐ．“会社”と“学校”をつなぐ．“会社”と“会社”をつなぐ．ここで取り上げたいのはマッチングアプリの可能性です．

1. はじめに

ソフトウェアが社会のなかで位置する重要性は大きくなり専門性は深くなっていて，個人や単一の組織ではとても手に負えない状況となっている．いざ，ソフトウェアを作るぞ！となったときに“その技術を持った人”を探す術はなく，現状は人脈にたよって人探しをしているのである．また，欲しい機能が既に世に提供されていることを知らずにせっせと作ったのはいいものの，“あら，同じものを作ったの？”と脱力する一言をもらうこともしばしば．

この状況を打破したい．とまわりを見回すとマッチングアプリが使えるにではないかと気づく．

マッチングアプリは人と人をつなぐ，人とモノをつなぐといった“つなぐ”ことが出来ている．欲しいものと欲しい人，売りたいものと買いたい人，お友達になりたい人同士をつなげることができている．

ソフトウェアの分野においても”技術を持つ人”と”その技術を欲している人”を，“欲しい機能”と“機能を使いたい人”をつなぐことが出来るのではなかろうか．われわれソフトウェア開発の世界にもマッチングアプリの仕組みが利用できるのでは？ないかと思うのです．

2. 課題

わたしには以下の課題があります．
・”製品間で重複開発が発生して保守工数が肥大化している．”
・”開発した機能を他製品に横展開ができないために，機能リリースに時間がかかる．”
・”知見のない技術を導入する際の学習コストが大きい．”

認識した課題があるにも関わらず，これらが解決されないのはなぜなのでしょうか？わたしなりに考えました．

理由：
・どこでどのような技術を持っているか分からない
・これから何を作る予定があるのかが分からない
・情報の流通は人脈に頼っている

情報共有の機会が足りないのでは？　情報共有会の運営の仕方が悪いのでは？と思われるかもしれませんが，それなりにはやってきました．

これまで実施してきた情報共有：
・開発後に開発内容を紹介する機会
・勉強会と称して，共有会が行われる機会
・開発前にこれから作りたい内容を伝える機会

開発資料、開発コードの共有も実施してきましたが、問い合わせを受けることは少なく、問い合わせをする方も限定的.

以上の通り, 単発的に情報共有は行われるものの, 情報共有の効果は薄い. ここで補足しておきたいが, 開発者が隣で行っている開発に関心がないわけではない. しかし, 結果として, 知りたい方に情報が届いていないので, 情報共有会, 資料の共有の限界と思われる.

仮に, 情報共有会が頻繁に開かれても, 開発者の時間は有限なので, すべての機会に参加はできない. 代表者が参加する形式をとった場合でも代表者の関心が薄いものは引っかからずに落ちていく.

一部の人に情報が集まるのは普通のことであるが, 情報は開発者まで流れていない. それはなぜか？ 情報が集まっていても、それを欲する開発者が彼らと直接的な接点を持たなければ、情報の流通は止まってしまうのではなかろうか？

“個人が持つ人脈の価値を無にすることで, 誰もがつながることができるのではないだろうか？” 個人の持つ人脈に頼るよりも, 関心を持った方が自由にアクセスできるようになった方が情報は活発に流れるのではないかと思うのです. 隠れた人材を見つけることにも注視すべきで, 人脈という表層に出てこない有能な人材を見逃していることがあればもったいない.

3. 可能性

マッチングアプリを応用して”ソフトウェア技術を持つ人”と”その技術を欲している人”をつなぐ仕組みを作ることはできないだろうか.

開発者が思いついた時に自由にアクセスできるといい. 誰かを通してとか, 紹介してもらってとなると, 面倒で時間がかかる.

もし, 知らない人から声がかかったら？ 今のご時世, 警戒して回答しない. 安心できる誰かの紹介なら回答をする. なると, 運営管理者がつなぐのか？ これだとバックトゥザフューチャーで, 過去の世代に戻ったような話になってしまうので良くない.

知らない人から声がかかっても, 安心して会話ができることが必要だ. 知らない人同士のコミュニケーションを円滑にするために, 伝える表現を”やわらかくする”または”ビジネスライクにする”技術も必要となる.

4. 今ある技術の活用

世にあるマッチングアプリでどのような技術で出来ているのだろうか？バックグランドで AI も頑張っていそうです.

ここで, AI の活用シーンを想像してみる.
・自然言語処理をつかった文書作成技術, 要約技術
EC ショッピングサイトの案内文は AI が作成している.

ある開発者はこんな技術を持っています. こんな開発に参加しました. といったことを, いちいち開発者に書いてもらっていては情報の登録が進まない.

開発者が作成した開発ドキュメント, 参加した開発の開発ドキュメントから自動で”概要文書”が作成されるならば技術者と技術が持つ技術の情報が蓄積される. 足りないことがあれば, 人が補足すればよいのである.

AI による文書作成の例として, 北大 調和系工学研究室の成果に“EC サイト用商品紹介文の自動生成”, “人工知能による競輪予想記事の自動生成”がある.
http://harmo-lab. jp/?page_id=4539
http://harmo-lab. jp/?page_id=4908

また, 長文の要約処理が必要であれば, 世には“文書要約 AI”がある. https://ai-tanteki.com/

・会話のサポート技術
初めての方同士をつなげるので, ことば使い, ことばの選び方, 応答の仕方でコミュニケーションが続くと思われる. 前述の通り会話サポート機能も必要です.

コミュニケーションサポートの世のある例として, 相手のとのチャットのやり取りをサポートする“AI 恋愛ナビゲーシ

ョンアプリ”がある. 紹介文にはデート受託率が 8 倍になると書かれていた. https://aill. ai/

・検索技術
キーワードだけでは対象外のものもひっかかる. 情報が選別する手間がかかるものは嫌われる. 検索結果には信頼度も必要で, ”はずれ”をつかまされ続けると, 使われなくなる. 困っている人がたくさんいるので, これを解決する技術はあるだろうと期待する.

5. 情報の登録

開発者、開発情報の登録がすすまないことにははじまらない. これらの登録は登録する方のメリットがないと進まない.

開発者が共有しない理由はなにか？を考える.
・共有することに個人, 組織のメリットがない
・過去の例をみると考え方の共有まではあるが, 構想～コードまで横展開できた事例は少ない
・技術を提供する方は提供する一方で, 見返りがない
・いつの間にか保守することになるなど, 面倒なことに巻き込まれる
・技術提供しているのに立場なのにいつの間にか否定される立場になっている
・目的の違うものを引き合わせようとされて, 余計な気苦労をする. おまけに人間関係もギクシャクするようになる

開発者が他者の開発に気が付かない理由はなにか？
・情報の公開範囲の関係で開発者まで情報がながれていない
・発信されている情報と自分たちが持っている課題との関連性に気が付かない
・そもそも他で似たことを進めているとは思っていない. 自分が最初との意識. これは決して悪いことではない.
他で進めていると分かれば, よほど相手が嫌でなかったら問い合わせるだろう.
・相手が忙しいので, 余計なことに時間を割いてもらうのは申し訳ないとの気持ちははたらくかもしれない

6. メリットの設定

技術を提供した人にメリットはあるのか？ メリットがないことには広がらない

Q. 報酬はいるのかいらないのか？ ときかれれば
A. 報酬は欲しいが, 投資した時間に見合う報酬ではない.
技術を提供したよりも提供された技術を活用する人が評価される. なぜならば, 技術を提供する側の評価は既に終わっており, 技術を受けて活用する方の評価はこれからだからである. 開発者同士でできることは, ありがとうと伝えることと, ゴチくらいである.

社内で行う技術交流会などがあれば、開発者として名前が出れば, 社内の認知度が上がり, 将来的な評価につながるかもしれないが確立は低い. なので, 内容によっては見合わないのである.

7. 制度の必要性

メリットの設定で伝えた通り, 技術を”売る制度”と”買う制度”が必要で, さらには”共同作業”となった際の責任分担＝報酬の分担を整える必要がある.

大学と企業の共同研究の場合,
大学は研究結果の検証という意味で大きな意味を持つのではないかと思う. 企業が支払う大学側への報酬は各大学で決められた”共同研究委託費用”. 特許性があればそれの扱い方で利益は共有できる. 他としては研究プロセスとその成果を学内授業で利用することへの了解などもメリットになるだろうか.

8. 事例検証

共有できたときと出来なかったときの違いは何か？

横展開が出来た事例:
・先行開発の活用
ゼロから作るには工期が足りなかったし, リソース面でも要求仕様のすべてをやり切るには足りなかった.

展開元のコードの可読性が高かった．仕様とコードの関連がすぐに分かる保守性の高いものだった．

・類似機能
利用者が同じ，開発者が同じ，要求元が同じとのことで機能の横展開ができていた．

横展開が出来なかった事例
・そもそも受け手側が横展開されるものを”良い”と思っていなかったので，はなしが進まなかった．

提供する側もあれこれと受け手側から注文，改善提案されるとだんだん，煩わしくなる．

プラットフォーム戦略をとるのであれば，はなしは違うかもしれないが，受け手が 100％の責任を持てないと横展開は出来ないように思われる．

横展開できたときを改めて考えると，
・欲しい人が欲しいものがあるとき
・開発をはじめる前に欲しいものがあることがわかったとき
・元のコードの保守性が高いとき

横展開を継続するには以下のことが必要です．
・技術の提供元の利益を守ること
・技術の提供元がどこなのかが分かるように残すこと
・貢献度に応じた利益のフィードバックを行うこと

9. 実現したいこと

”技術を欲しい人”と”その技術を持つ人”をつなげること． ”知りたい技術”を入力すると”その技術を持つ人”を紹介すること出来たら，開発の初動が早くなる．

重複開発の回避．”知りたい技術”を入力すると“既に用意された技術”が紹介されれば重複開発を回避することができる．

はじめは社内の人材，技術のマッチングから始めて，次は社外とのマッチング，その次は社外間のマッチングとなると知の交流に発展することを期待することが出来る．

10. まとめ

ソフトウェア技術の分野でもマッチングアプリの仕組みはいかせそうである． 技術を提供する方のメリットを明確にして，技術を売る，買う制度を作り，いまあるソフトウェア技術を融合し，足りないところを人は補うことで成長のループをつづければ Human In The Loop． 知の交流が進むだろう． この仕組み提案はここまでである． 仲間と共に実現したい．

こうして出したはなし（仕組み）が既にあるのであれば、活用したいので是非，教えていただきたい。

プロセスモデルの補完方法 -モデル・ノウハウ・人-

阪井 誠
株式会社 SRA
sakai @ sra.co.jp

要旨

本稿では，モデルを補完してソフトウェア開発プロセス（以下プロセス）を成長させる方法について議論する．プロセスはモデルだけでなく，実践するノウハウや人が必要であり，モデルに不足するノウハウを議論することで，新しいプロセスに追加すべきものを明確にし，プロセス改善に役立てるのである．この提案を通して，プロセスの議論が，どのように改善し，成長させていくかといった前向きでオープンな議論に発展することを期待する．

1. はじめに

良いプロダクトの開発には良いプロセスモデルだけでなく，実践に必要なノウハウや人が必要である．これまでプロセスモデルやマインドセットの議論が多く行われてきたが，これらはプロセスの理解に役立つものの，より良いプロセスに成長させるには不十分であった．過去のソフトウェアシンポジウムにおいては，プロセスは文化だと言われてきた．組織内のノウハウや人を無視して，プロセスを単純化したモデルや，その実現に必要なマインドセットを議論しても移行後のプロセス改善にはつながらない．

アジャイル開発を例に挙げると「ウォーターフォールでもしっかり開発出来ないのにアジャイル開発なんて出来るわけがない」[1]という議論がある．アジャイル開発に対する理解不足もあるが，ソフトウェア開発に共通するプロセスモデル以外の要素があることを示している．基本的な技術や業務知識など，開発で考慮しないといけない点はモデルに関係なく共通点も多いからである．アジャイル開発を始める際に，守破離という言葉を使って厳格な実践が求められる場合もあるが，本当にそれだけでよいのだろうか？それが本提案のきっかけである．

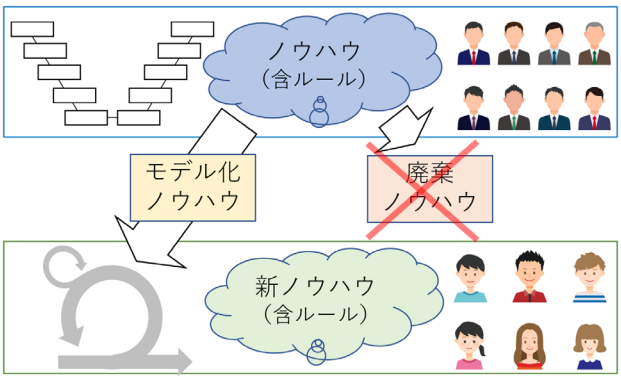


図 1新しいプロセスへの移行

プロセスの成熟にはモデルのほか，多くのノウハウや経験に基づく様々な工夫が必要で，チーム内だけでなく社内外の情報を利用して成長させる必要がある．

2. 新しいプロセスへの移行の問題

図 1 に新しいプロセスへの移行を示す．従来のプロセスモデルで開発を行う中で，うまく適応するための工夫からルールを追加するなど様々なノウハウが生まれ，人と組織は成長する．しかし，モデルの問題や，実践が簡単ではないノウハウがあると，それらの解決が期待できる新しいモデルと新しいノウハウに移行される．

従来のノウハウは新しいモデルやノウハウに取り入れられるものもあるが，廃棄されるノウハウもある．また，廃棄されたノウハウを知っている経験者が新しいプロセスに参画できない場合もある．こうして古いプロセスモデルと共に文化が失われてしまうのである．

新しいプロセスモデルでの文化はすぐには成熟しない．開発チームなどの学びはあるものの，人の入れ替えや，プラセボ（偽薬）のように新しいものは良いものであるとの思い込みから失敗するまで守りに入るからである．

この様な例としてアジャイル開発を取り上げ，新しいプロセスモデルに不足するノウハウの補完法を議論する．

3. アジャイル開発ケーススタディ

3.1. 実現が難しいノウハウとアジャイル開発

「アジャイル開発は，ソフトウェア工学の正統な進化形」[2]と言われる様に実現が困難だったノウハウをアジャ

イル開発はモデル化している．スクラムで解決した問題例を以下に示す．

- 「現地現物」はプロジェクト管理のノウハウだが，従来は成果物，特に実装を確認するまでに時間がかかり，実装確認後のフィードバックが難しかった．スクラムにおいては繰り返しのスプリントごとにレビューを実施することで，適切なフィードバックが可能になっている．
- ハンフリーはソフトウェア開発に「自律的なチーム」や「その最大限の能力を最大限発揮できるようメンバーを動機付け，コーチし，後押しする」ことが重要だとしたが[3]，規律に基づく組織では困難な場合もあった．スクラムではスクラムマスターがチームを支援する奉仕者であり，真のリーダーとなっている[4]．

このように従来は困難だったノウハウがスクラムのプロセスモデルでは解決されている．

3.2. スクラムにおけるプロセスモデルの補完

ベームはリスク管理されていることをリスク駆動と呼び，スクラムに対してマネジメントプロセスは制約の源泉として完全に適用できるが，リスク／機会は制約として適用できるのは部分的とし，強いリスク駆動とはされていない[5]．プロダクトバックログ（PBL）につけられた優先順位に応じてストーリを実現する仕組みだからである．

これに対して，ジェフサザーランドは，プロダクトオーナー研修で「PBL は価値最大，リスク最小で優先付け」と説明し[6]，スクラムのモデルを補完するノウハウを示している．また，開発チームにおいては妨害リストが管理されるなど，ノウハウでプロセスモデルを補完してリスクが管理されるようになっている．

その一方で，イテレーティブな開発を「はじめに小さく失敗する」と説明し，小さな単位で開発することであたかも全てのリスクが減るように表現されることもある．また，リスクに関して記述の少ない書籍も多い．これはプロセスが未熟というよりは，業務によって必要性が異なるからと考えられる．モデルに組み込まれていないノウハウは，業務などによって取捨選択されている可能性がある．

4. 議論

ここではスクラムという汎用的なプロセスをリスク管理で考えたが，各企業のプロセスにおいても，プロセスモデルだけでなく，それを補うノウハウの議論が必要であろう．

プロセスは，プロセスモデルだけでなく，ノウハウや人を含めた文化であり，モデルを忠実に実践するだけでは成熟が困難な場合がある．社内外の知識の利用が必要で，コミュニティや書籍，経験者の知恵を借りることで，より良いプロセス改善することが可能になる．

例に挙げたリスクに関しても，リーンソフトウェア開発の「決定をできるだけ遅らせる」プラクティスなども参考になるだろう[7]．また業務固有のリスクなら，社内外の知見者の意見は大いに役立つと考えられる．このように収集したノウハウを組織に合わせて取捨選択し，モデルと組み合わせて補完すればプロセスを成長させられるだろう．

5. おわりに

CMM ブームの後，TQC を懐かしむ声を聞くことがあった．新しいプロセスを導入した際に，それまでに積み上げてきた文化を無くして移行してしまったからである．単に捨ててしまうのではなく，重要なノウハウを利用してプロセスを成長させていくべきである．今回の提案を通して，どのように改善し，成長させていくかといった前向きでオープンな議論に発展することを期待する．

参考文献

[1] アジャイルよろず相談室，*「ウォーターフォールでもしっかり開発出来ないのにアジャイル開発なんて出来るわけがない」という言い方をどう思いますか？*，https://qr.ae/pGLjdc, Quora, Inc.

[2] Yasuharu NISHI，ちなみに僕のスタンスは，アジャイルはソフトウェア工学の正統な進化形だというものです．進化がアンチテーゼを必須にするというなら，アジャイルは一度ソフトウェア工学を否定しなきゃいけないんでしょうし，それも含めて正統な進化形です．， https://twitter.com/YasuharuNishi/status/1164336449967099904, Twitter, 2019

[3] 阪井，デブサミ運営事務局・SEshop.com 編集部編，リーダーに求められる大切なこと，100 人のプロが選んだソフトウェア開発の名著，pp.20-21 ，翔泳社，https://www.slideshare.net/MakotoSAKAI/ss-16581244, 2012

[4] Ken Schwaber, Jeff Sutherland, スクラムガイド，https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf, ScrumGuides.org, 2020

[5] バリー・ベーム,アジャイルと規律，pp.205-210，日経 BP 社, 2004

[6] Sukusuku Scrum，スクラムプロジェクト準備(公開用) No.31， https://www.slideshare.net/SukusukuScrum/no31-13338313/41, p.41, 2012

[7] メアリー・ポッペンディーク，トム・ポッペンディーク,リーンソフトウェア開発，pp.79-109，日経 BP 社, 2004

ソフトウェア・シンポジウム 2022
論文集

2022 年 6 月 5 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 978-4-916227-28-7

ソフトウェア技術者協会