2023年5月26日
派生開発カンファレンス2023

SWEBOK V3からV4への改訂に見るソフトウェアエンジニアリングの発展とソフトウェア保守・進化

鷲崎弘宜
早稲田大学 / 国立情報学研究所 / システム情報 / エクスモーション
IEEE Computer Society 1st Vice President
washizaki@waseda.jp

V20230524

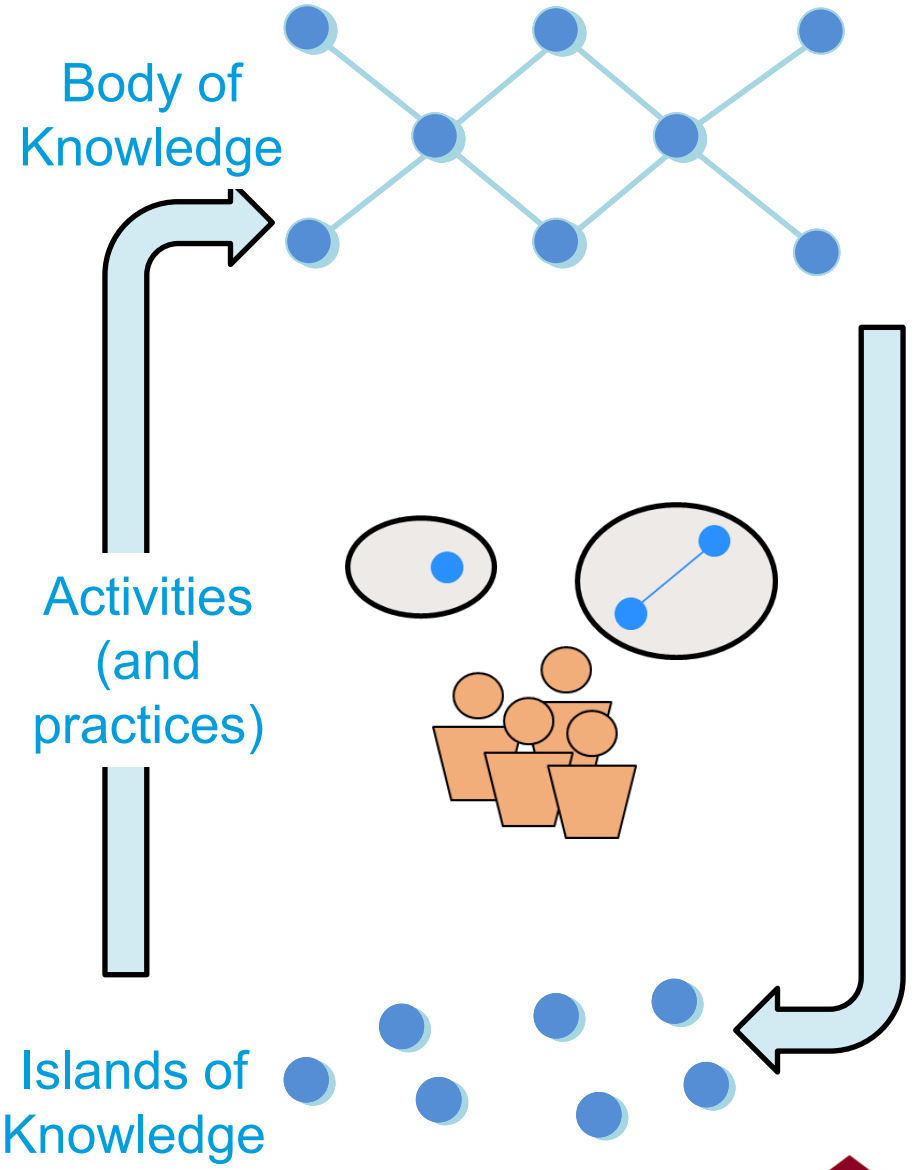
目次

- SWEBOK V4
- アジャイル
- IoTソフトウェアエンジニアリング
- AI/MLソフトウェアエンジニアリング
- まとめ

Guide to the Software Engineering Body of Knowledge (SWEBOK)

http://swebokwiki.org

- 歷史: 2001 v1, 2004 v2, 2005 ISO/IEC Technical Report, 2014 v3, **2022 v4 Beta, 2023 v4 Final soon!**
- 目的
  - Guiding learners, researchers and practitioners to identify and have common understanding on **"generally-accepted-knowledge"** in software engineering
  - **Defining boundary** of software engineering and related disciplines
  - Providing **foundations for certifications and educational curriculum**
- 採用
  - IEEE-CS software professional **certification programs** based on SWEBOK (Associate Software Developer, Professional Software Developer, Professional Software Engineering Master)
  - **ISO/IEC 24773-4: Certification of software and systems engineering professionals** - Part 4: Software engineering
  - **Software Engineering Competency Model (SWECOM)**



SWEBOK GuideのV3からV4への進化

- 現代的ソフトウェアエンジニアリングの反映、プラクティスの更新、知識・新領域の成長
- 公開レビュー https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-committee/swebok-evolution

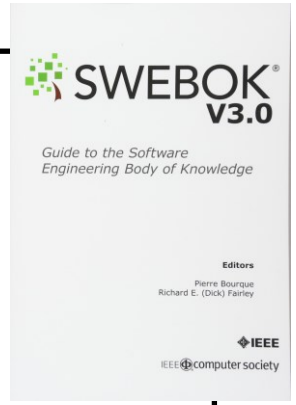
SWEBOK V3

Requirements

Design
Construction
Testing

Maintenance
Configuration Management
Engineering Management
Process
Models and Methods
Quality

Professional Practice
Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations



SWEBOK V4

Requirements
Architecture
Design
Construction
Testing
Operations
Maintenance
Configuration Management
Engineering Management
Process
Models and Methods
Quality
Security
Professional Practice
Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

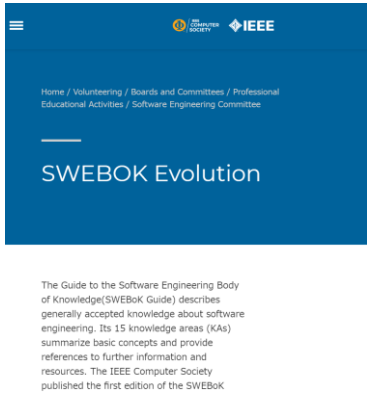
Agile testing
…

Agile, DevOps

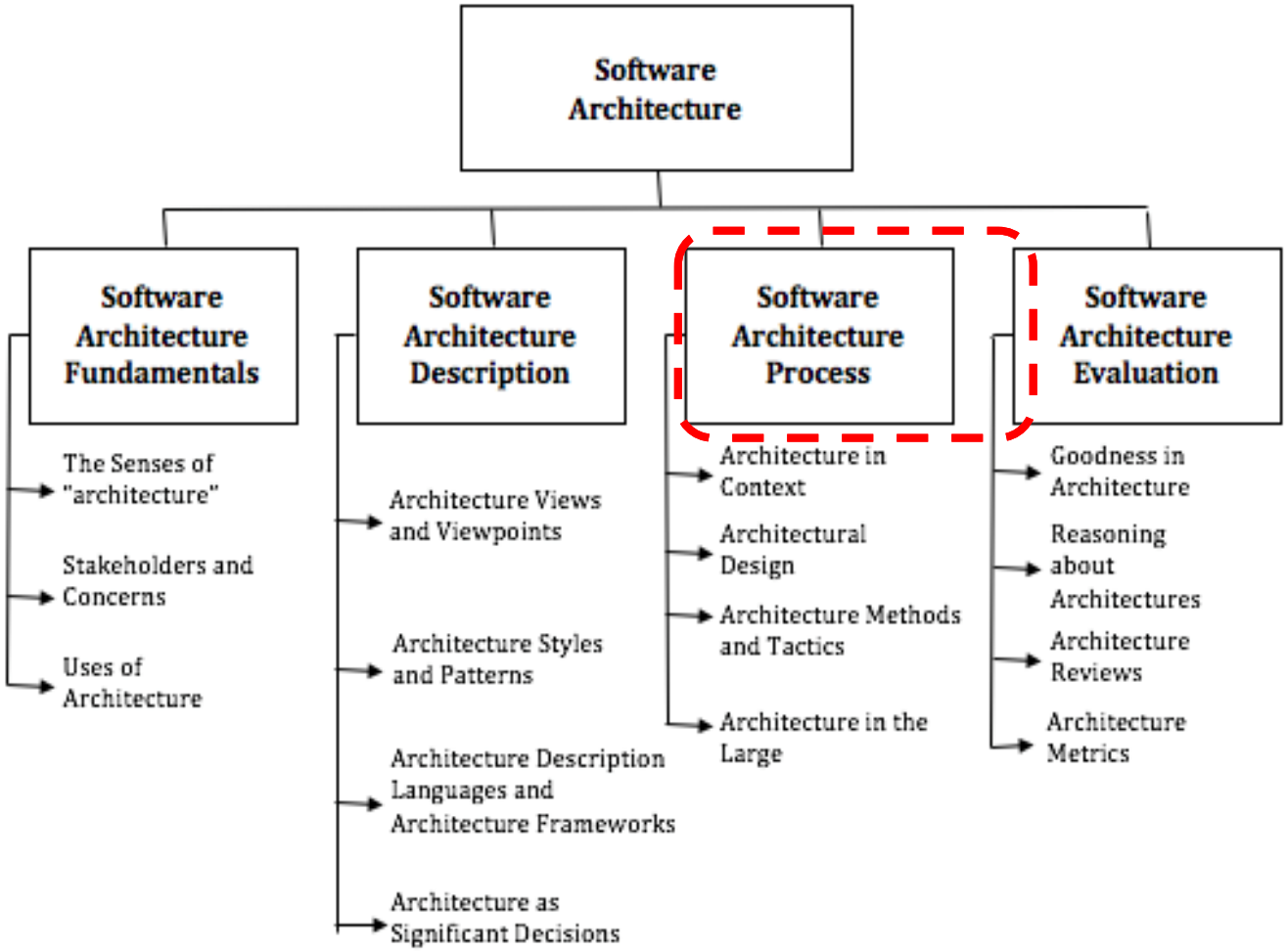
Agile security
…

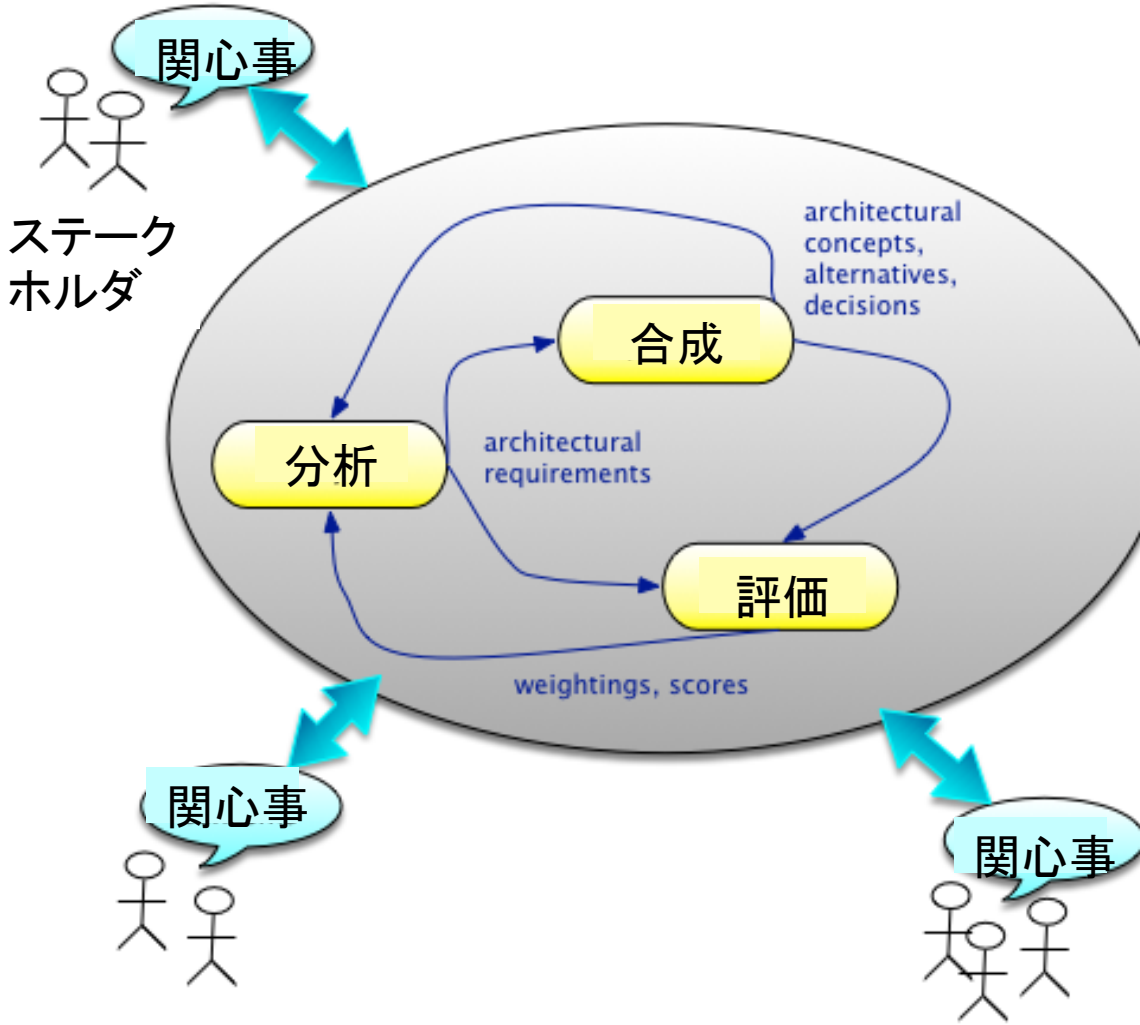
IoT

AI ←→ SE

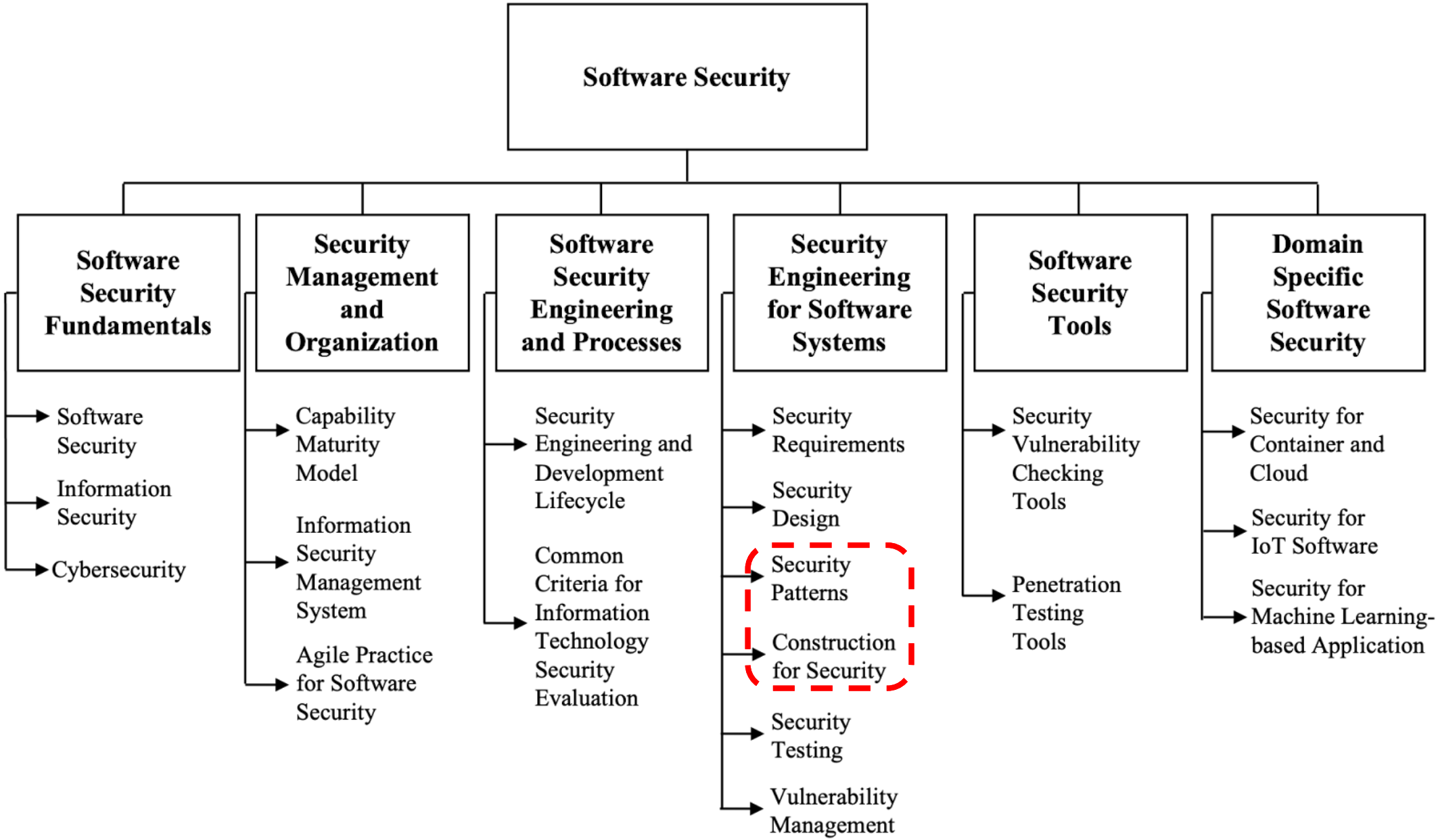


新知識領域: ソフトウェアアーキテクチャ





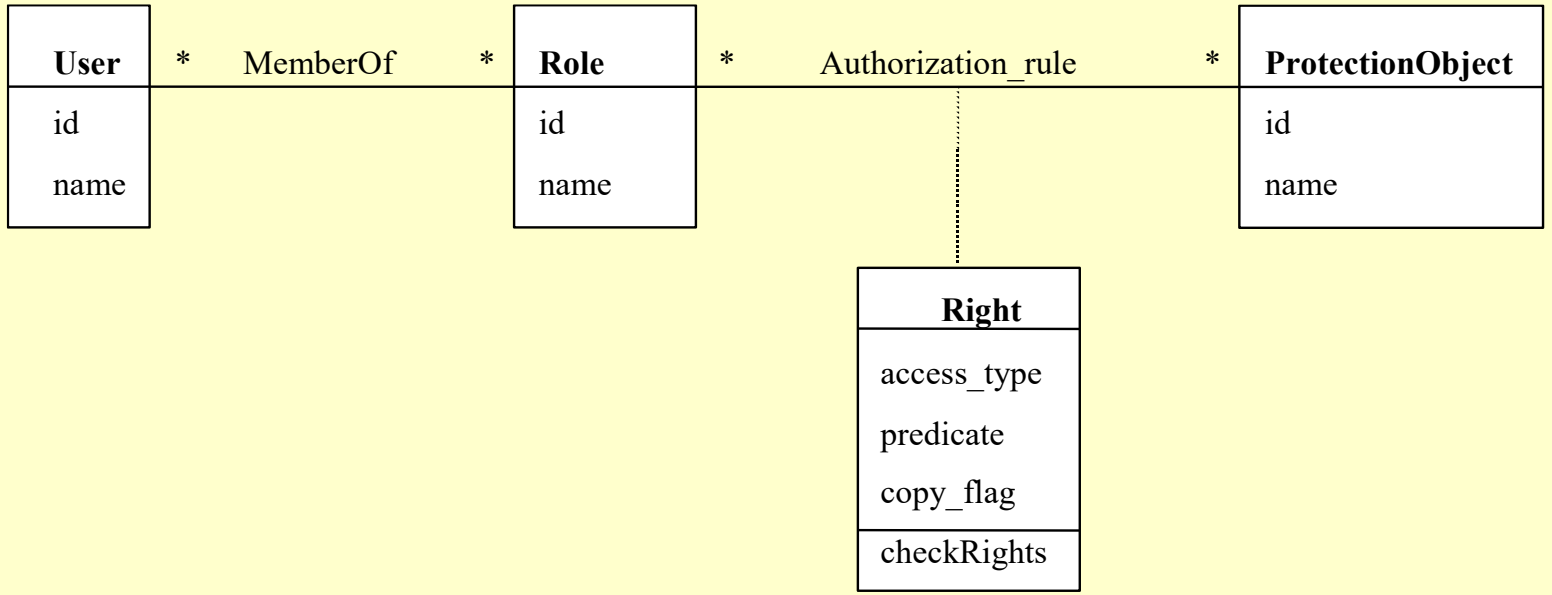
新知識領域: ソフトウェアセキュリティ



セキュリティパターン

- 特定の文脈上で頻出のセキュリティ問題と対応する解決策・経験をまとめたもの

  - 名前: ***Role-Based Access Control (RBAC)***
  - 問題: How do we assign rights to people based on their functions or tasks?
  - 解決: Assign users to roles and give rights to these roles so they can perform their tasks.
  - 関連パターン: ***Authorization***, ...



注意: 本講演時点における検討事項であり今後変更の可能性があります。

セキュリティ構築(実装)

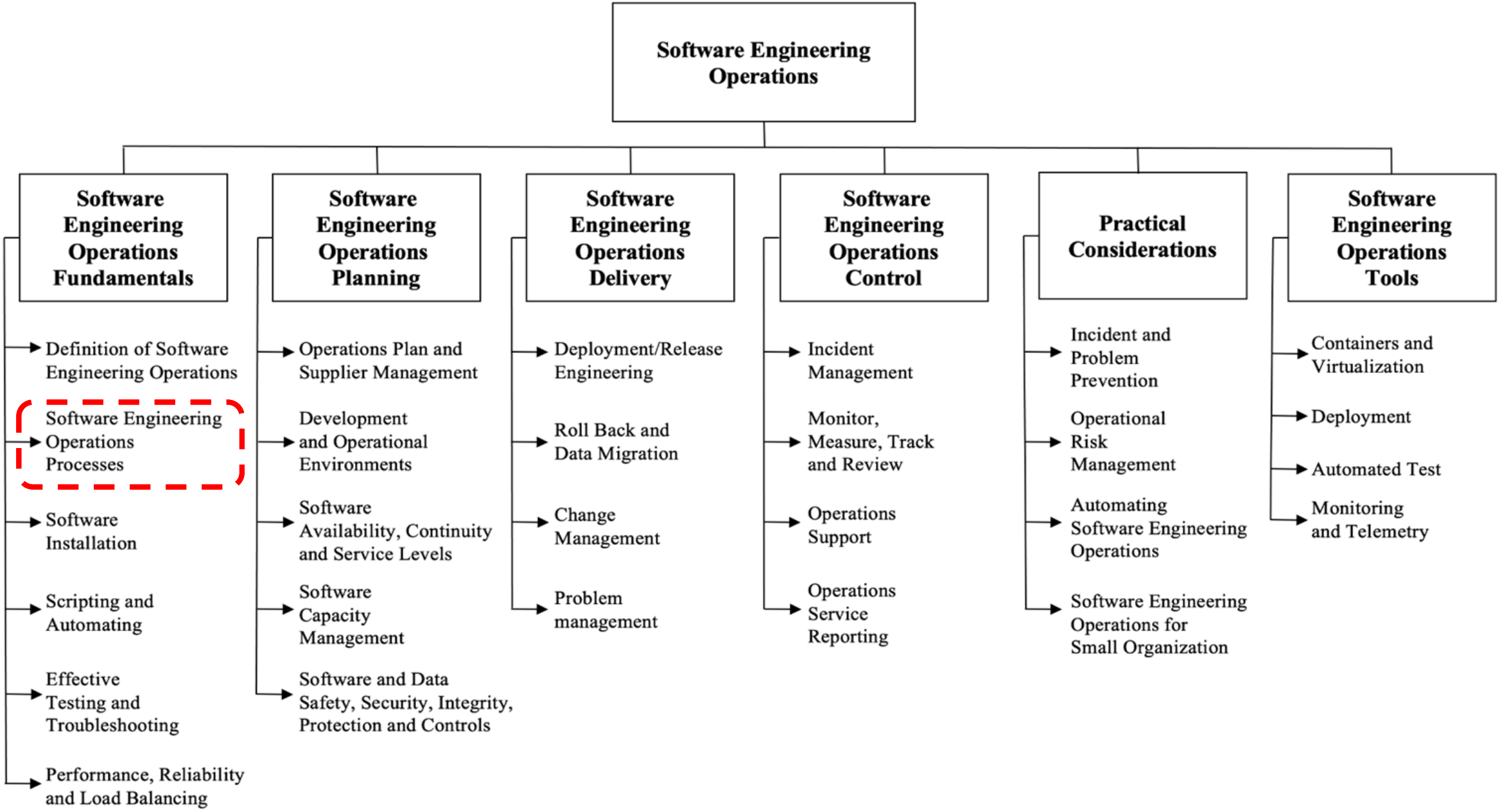
例: CERTセキュアコーディング標準

- 入力を検証
- コンパイラの警告に注意
- セキュリティポリシーを考慮した設計
- シンプル
- デフォルトで拒否
- 最小特権の原則を遵守
- 他のソフトウェアに送信されるデータをサニタイズ
- 防御策の実践
- 効果的な品質保証技術を使用
- ソフトウェア構築のセキュリティ標準を採用

Robert C. Seacord, The CERT C Secure Coding Standard, Addison-Wesley Professional, 2008

注意: 本講演時点における検討事項であり今後変更の可能性があります。

新知識領域: ソフトウェア運用



運用知識領域における DevOps

IEEE Std 2675: IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment

- 原則
  - ビジネス・ミッションファースト
  - 顧客志向
  - レフトシフト（特にCDやテスト、品質活動の早期からの取り組み）
  - CX（Continuous Everything）
  - システム思考
- 組織文化
  - リーダーシップ
  - 組織構造とダイナミクス
  - 効果的なコミュニケーションと協調
  - 学習する組織とナレッジマネジメント
  - 適応力、レジリエンス、組織変革
- ライフサイクルプロセス
  - 契約プロセス、組織的なプロジェクト実現プロセス、技術管理プロセス、技術プロセス

注意: 本講演時点における検討事項であり今後変更の可能性があります。

横串としてのアジャイル＆DevOps

- アーキテクチャ知識領域
  - アジャイル開発ではしばしばアーキテクチャ設計フェーズが明示されず
  - DevOpsなどの継続的開発におけるアーキテクチャプロセスの応答性メトリクスへの注目: 変更のリードタイム、デプロイ頻度、サービス復元平均時間、変更失敗率など
- セキュリティ知識領域

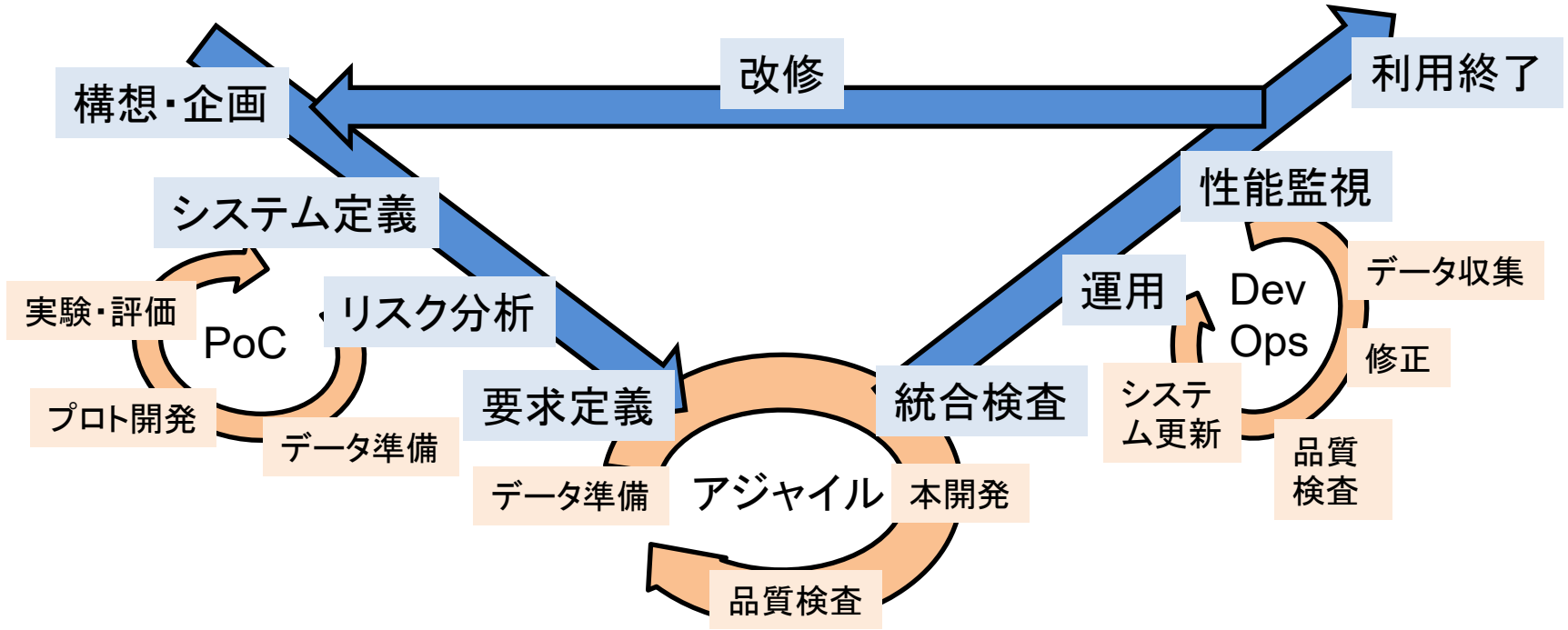
  アジャイルセキュリティプラクティス [Bell17]
  - セキュリティがブロッカーではなく、イネーブラーであること
  - 変化を受け入れ、より速く、より反復的に作業し、セキュリティリスクとリスク管理の方法について、段階的に考慮
  - セキュリティチームと開発者の関与、有効化、自動化、アジャイルチームに追従するためのアジリティ

Laura Bell, Michael Brunton-Spall, Rich Smith, Jim Bird, Agile Application Security, O'Reilly,2017

注意: 本講演時点における検討事項であり今後変更の可能性があります。

アジャイル開発・DevOpsと動的品質

- 対象: データ、ソフト、システム
  - クローズよりもオープンへ
  - 静的よりも動的へ
- プロセス
  - アジャイル＆DevOpsが標準
  - 計画よりも変化へ
- 基準
  - 単純・確実から複雑・不確実へ
  - 100%よりも95%の正しさとリスクへ
  - 要求よりも価値へ



プロセス参考: 産業技術総合研究所サイバーフィジカルセキュリティ研究センター テクニカルレポート CPSEC-TR-2020001, 機械学習品質マネジメントガイドライン, https://www.cpsec.aist.go.jp/achievements/aiqm/

主要なモダン・ソフトウェアエンジニアリングの側面

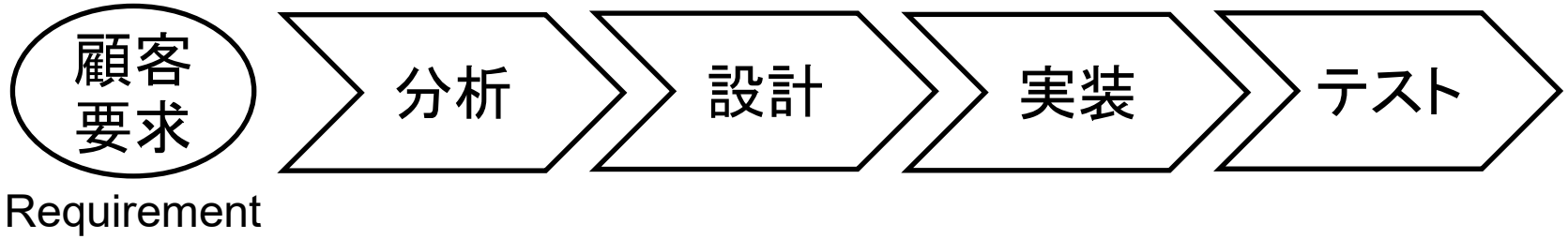
知識領域	アジャイル・DevOps	IoT	AI/ML
アーキテクチャ	アーキテクチャプロセスの応答性メトリクス		
テスト	アジャイルテスト	IoTテスト	
保守	アジャイル保守と課題（軽量な文書化、頻繁なテストなど）	IoT保守と課題（相互運用性など）	デバッグ、パイプライ統合など
運用	IEEE Std 2675: IEEE Standard for DevOps		
セキュリティ	アジャイルセキュリティプラクティス	IoTセキュリティ	
コンピューティング基礎			AI for SE, SE for AI

目次

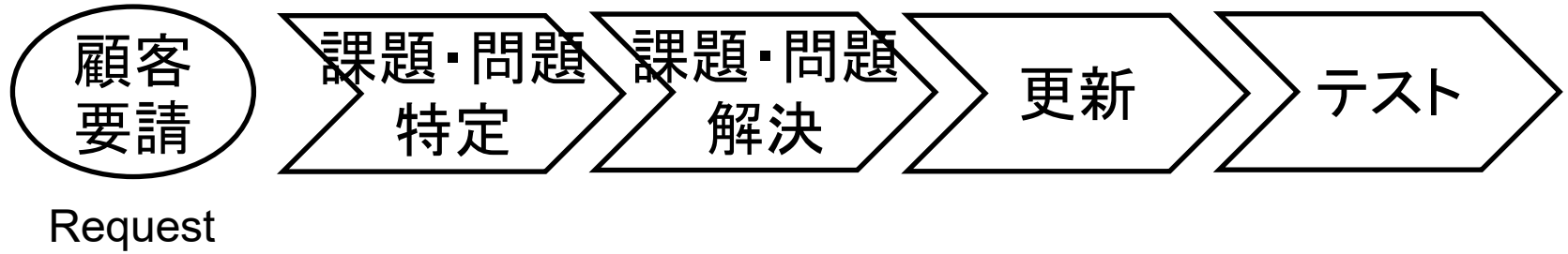
- SWEBOK V4
- アジャイル
- IoTソフトウェアエンジニアリング
- AI/MLソフトウェアエンジニアリング
- まとめ

開発と保守のプロセス

開発プロセス



保守: 継続的進化プロセス



Liguo, Y.; Mishra, A. Risk Analysis of Global Software Development and Proposed Solutions. Automatika 2010, 51, 89–98.
Almashhadani, M.; Mishra, A.; Yazici, A.; Younas, M. Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment. Information 2023, 14, 261.

アジャイル保守の課題

保守プラクティス	品質別の保守課題
繰り返し(イテラティブ)開発	透明性: プロジェクトの透明性の欠如は、プロジェクト見解の不明確さにつながる。 管理性: 管理性の欠如は、スプリント内での衝突を引き起こす。 基盤: 基盤の不十分さが開発・保守のプロセスに繰り返し影響
作業成果への注力	透明性: プロジェクトの透明性の欠如は、プロジェクト見解の不明確さにつながる。
緊密なチームワーク	透明性: プロジェクトにおける透明性の欠如はチームのコラボレーションに影響する。コラボレーション: コラボレーションが欠如していると、チームワークが悪くなる。
緊密な顧客参加	コラボレーションとコミュニケーション: 顧客とチーム間のコラボレーションとコミュニケーションの欠如は、プロジェクトの品質に影響を与える。 管理性: 管理性の欠如は、顧客の要求と保守チームとの間の衝突につながる。
対面のコミュニケーション	コラボレーションとコミュニケーション: 顧客とチーム間のコラボレーションとコミュニケーションの欠如は、プロジェクトの品質に影響を与える。
軽量な文書化	管理性: 管理性の欠如は、顧客の要求と保守チームとの間の衝突を引き起こす。
頻繁なテスト	スケーラビリティ: 拡張性の欠如は、保守で不可欠な頻繁なテストプロセスに影響を与える。 基盤: 基盤が整備されていない場合、頻繁に行われるテストに影響。
共同所有を通じた動機づけ	コミュニケーションとコラボレーション: チーム間のコラボレーションとコミュニケーションの欠如は、共同所有を阻害する。
オープン化による知識伝達	コラボレーションとコミュニケーション: チーム間での欠如は、知識の伝達を阻害する。 透明性: プロジェクトにおける透明性の欠如は、保守チームのメンバー間の知識移転に影響。

Almashhadani, M.; Mishra, A.; Yazici, A.; Younas, M. Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment. Information 2023, 14, 261.

アジャイル品質パターン
QA (Quality Assurance) to AQ (Agile Quality)

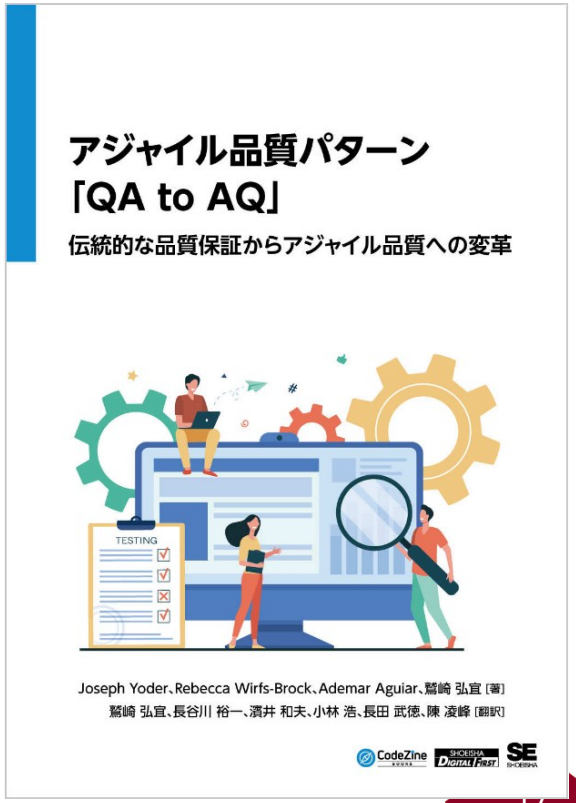
- アジャイル品質の考え方と推奨される活動の23+のパターン集
  - 2014年 Joseph Yoder, Rebecca Wirfs-Brock, Ademar Aguilar発表、以降、鷲崎も加わり拡充
- Joseph Yoder、Rebecca Wirfs-Brock、Ademar Aguiar、鷲崎 弘宜 著、翻訳：鷲崎 弘宜、長谷川 裕一、濱井 和夫、小林 浩、長田 武徳、陳 凌峰、“アジャイル品質パターン「QA to AQ」伝統的な品質保証からアジャイル品質への変革”、翔泳社、2022 https://www.amazon.co.jp/dp/B0BGRP1VRV/

中核パターン

品質のアジャイルなあり方

品質の特定

品質の可視化

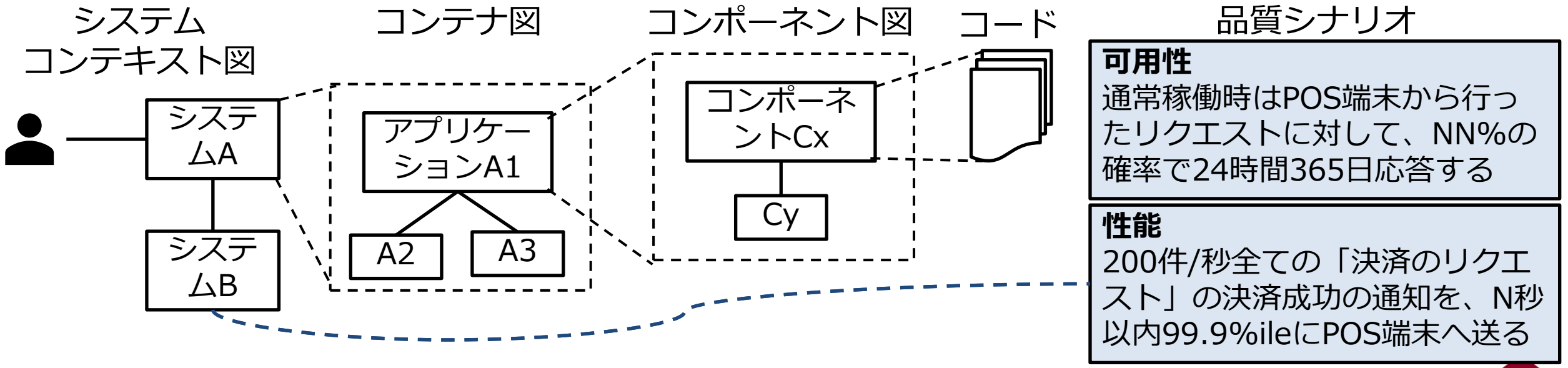


品質シナリオ

開発中に対処する必要がある重要な品質特性について、どのようにすればよりよい理解と大まかな見解を得ることができるでしょうか？

プロセスの早い段階で、手軽な方法を利用して、性能、負荷、信頼性、セキュリティなどの重要な非機能要件を扱う大まかな品質シナリオを作成しましょう。

C4モデルから目標に至る品質定義（NTTデータ 長田さんのScrum開発事例）



J.Yoder、R. Wirfs-Brock、A. Aguiar、鷲崎 著、翻訳：鷲崎、長谷川、濱井、小林、長田、陳、“アジャイル品質パターン「QA to AQ」 伝統的な品質保証からアジャイル品質への変革”、翔泳社、2022

できるだけ自動化

重要な品質特性について、迅速なフィードバックを支援し、現在の状態を共有し可視化するために、ツールや環境をどのように確立できるのでしょうか？

環境を整え、ツールを使用して、バリューを高める部分をできるだけ早く自動化しましょう。開発の終盤まで自動化のタスクを先送りしないようにしましょう。

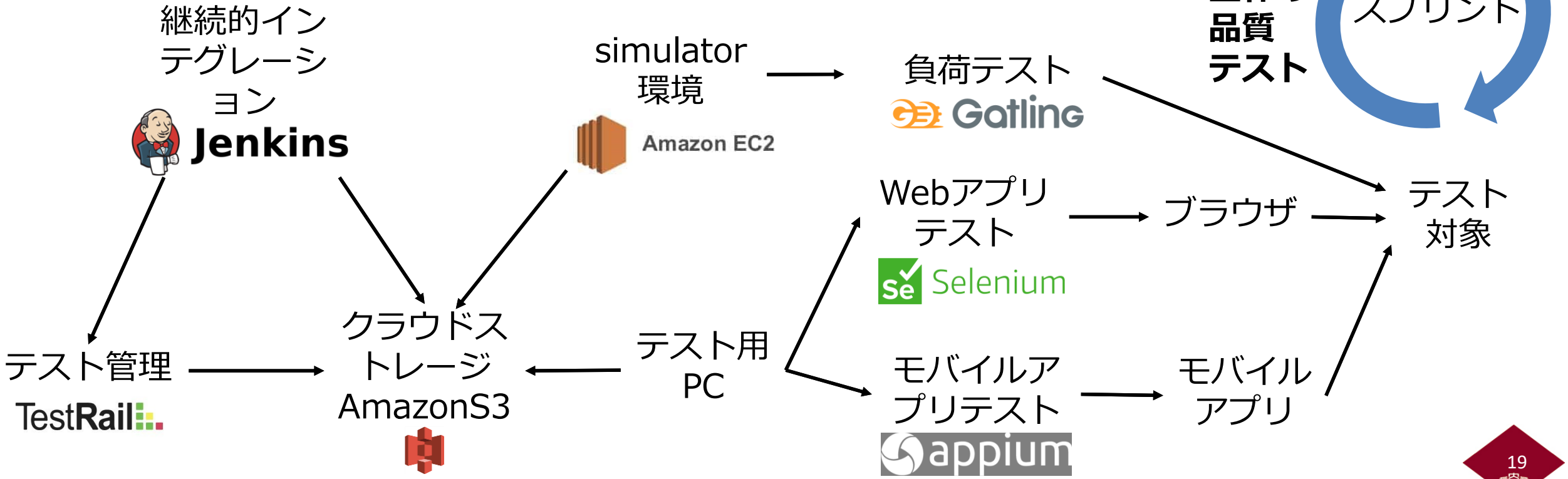
完成部分の品質テスト

デイリー
スクラム
など

全体の品質テスト

スプリント

テスト自動化（NTTデータ 長田さんのScrum開発事例）

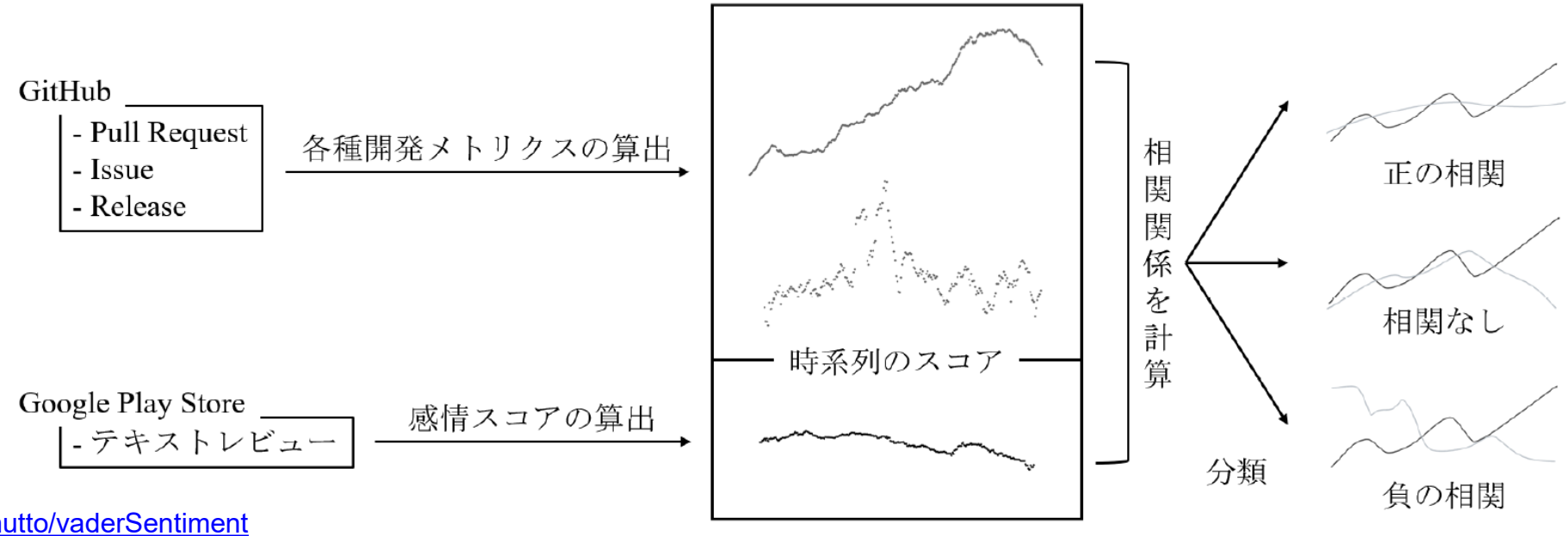


J.Yoder、R. Wirfs-Brock、A. Aguiar、鷲崎 著、翻訳：鷲崎、長谷川、濱井、小林、長田、陳、“アジャイル品質パターン「QA to AQ」 伝統的な品質保証からアジャイル品質への変革”、翔泳社、2022

どのようなアジャイルメトリクスが顧客満足に繋がるのか？

（早稲田大学、デジタルハーツ・AGEST 共同研究）

- 対象: アジャイルらしいOSSのAndroidアプリ37件
  - 一定規模、継続的統合＆プルリクエスト（WordPress, Firefoxなど）
- メトリクスと方法:
  - アジャイルプロセスメトリクス: 課題の起票からリリースまでの時間、マージされたプルリクエスト数、マージまでの時間、課題の生存期間、プルリクエストの生存期間
  - 利用者満足: アプリのユーザレビューコメントの感情スコア（VADER利用）
  - 分析方法: 遅延を考慮した上で時系列の相関関係



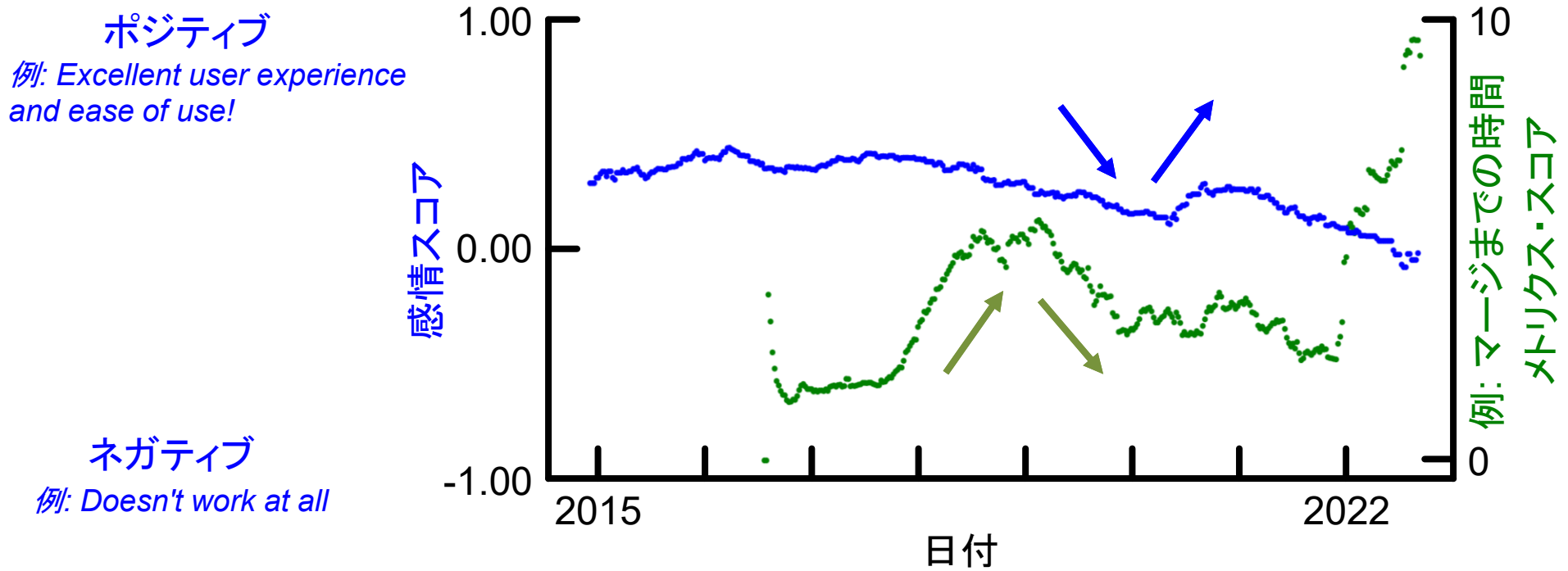
https://github.com/cjhutto/vaderSentiment

楊 敏舜, 佐藤 靖治, 鷲崎 弘宜, 深澤 良彰, 髙橋 寿一, “利用者満足度に影響を与えるアジャイル開発プロセスの特徴の特定”, 情報処理学会 第213回ソフトウェア工学研究会, 2023

どのようなアジャイルメトリクスが顧客満足に繋がるのか？（続き）

- 結果と考察
  - 感情スコアが全体的に低下傾向のアプリでは、マージまでの時間、課題の生存期間、プルリクエストの生存期間が感情スコアと強い負の相関

    → 体制やタスクの優先順位付け、承認プロセスなどの改善を通じて**迅速な課題処理やマージが望ましい**可能性
  - 課題の起票からリリースまでの時間、マージされたプルリクエスト数は感情スコアとの間に強い**相関関係は確認できず**
- 展望: メトリクス拡大、実開発検証、利用者・顧客満足から開発者満足へ



楊 敏舜, 佐藤 靖治, 鷲崎 弘宜, 深澤 良彰, 髙橋 寿一, “利用者満足度に影響を与えるアジャイル開発プロセスの特徴の特定”, 情報処理学会 第213回ソフトウェア工学研究会, 2023

目次

- SWEBOK V4
- アジャイル
- IoTソフトウェアエンジニアリング
- AI/MLソフトウェアエンジニアリング
- まとめ

IoTの課題とソフトウェアエンジニアリング

- 調査: A Systematic Mapping study on Internet of Things challenges (SERP4IoT'19)
  - IoTの課題（challenge）を扱う76編の論文を特定、うち論文誌掲載45編

課題	主要な話題	ソフトウェアエンジニアリング上の取り組み
多様な応用機会とコンテキスト	• ヘルスケア（9編）、スマートシティ（7）、スマートファクトリー（5）、家庭・スポーツ他（4） • プライバシ、セキュリティ、セーフティ • デバイスの多様性	• コンテキスト・アウェアIoTサービス、環境考慮によるグリーンIoT • リアルタイムデータを扱えるモデリングフレームワーク • （プロダクトライン）
通信技術	• 通信における低消費電力、高スケーラビリティ、低遅延、M2M・D2D通信	• （QoS制御、センサ分散配置最適化、データ集約など）
相互運用性	• 動的、変化 • 資源発見・選択、構成・接続の正しさ • 状態モニタリングとレジリエンス	• 自己完結マイクロサービス化 • 構成のハイレベルな記述と自動化、**モデル駆動開発** • 自動修正・再構成、セマンティックWeb • **アーキテクチャ・デザインパターン**

Aleksandr A. Lepekhin, Alexandra Borremans, Igor V. Ilin, Sami Jantunen: A systematic mapping study on internet of things challenges. SERP4IoT@ICSE 2019

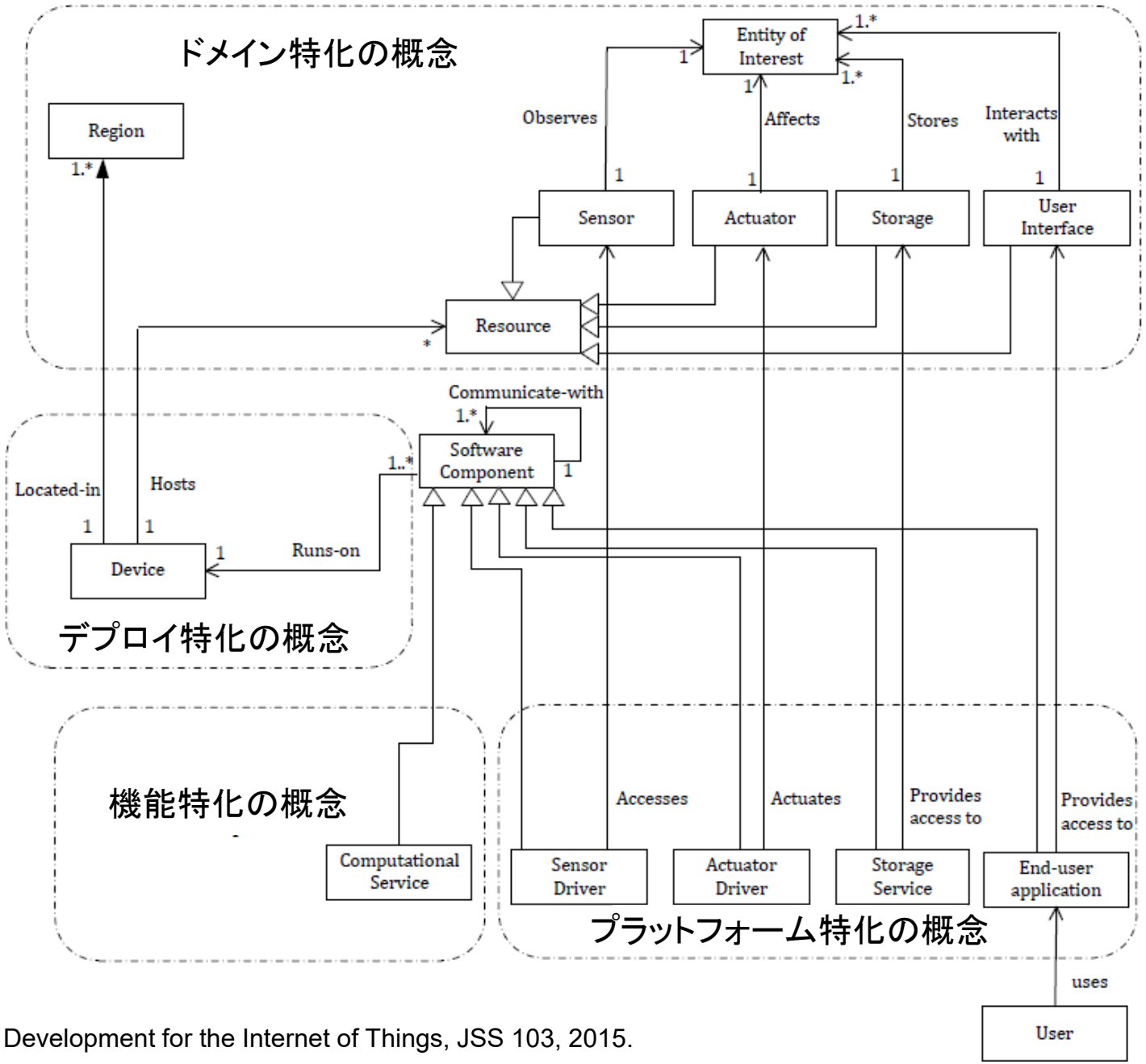
IoTの課題とソフトウェアエンジニアリング（つづき）

課題	主要な話題	ソフトウェアエンジニアリング上の取り組み
セキュリティ	• 多様なデバイス、環境、頻繁な変更 • 技術、組織、方法論上の課題	• セキュアなプロセステーラリング • 認証・認可アーキテクチャ、モデルベース • ファームウェア修正・更新 • （アクセス制御、暗号化、キー配布、ポリシー）
データとプライバシ	• ビッグデータの3V（量、速度、種類） • ノイズやセンサエラーに起因するデータの不確実性 • データプライバシ、個人情報保護	• セマンティックWeb技術を通じたIoTデータの意味的アノテーション • （データ・アクセス プライバシ技術全般）
IoT開発上の考慮	• 組織上の課題 • バッテリー、消費電力 • デバイスの多様性、組み合わせ • 継続的配備の必要性 • 多様な利害関係者、サイロ • IoTビジネス・エコシステムの不十分な理解	• エンタープライズアーキテクチャへの統合 • バッテリー・消費電力の明示的扱い • ビジュアルプログラミング言語 • ミドルウェア、（モデル駆動、プロダクトライン） • **アジャイル開発** • ユニバーサル・参加型デザイン • エコシステムにおける人間関係・トラスト

Aleksandr A. Lepekhin, Alexandra Borremans, Igor V. Ilin, Sami Jantunen: A systematic mapping study on internet of things challenges. SERP4IoT@ICSE 2019

モデル駆動IoT開発の例

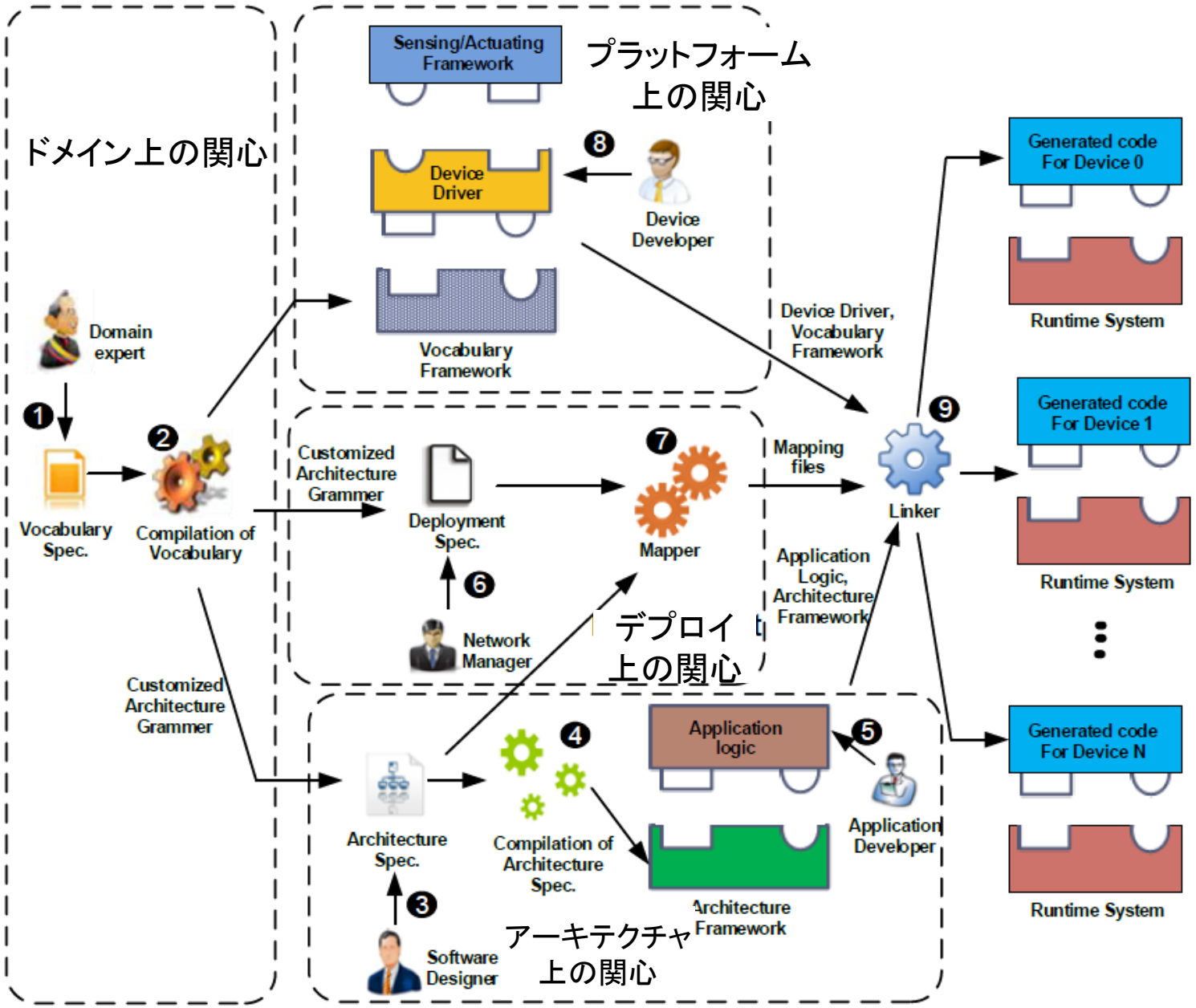
- 関心事の分離
  - ドメイン
  - 機能
  - 配備（デプロイ）
  - プラットフォーム



P. Patel and D. Cassou, Enabling High-level Application Development for the Internet of Things, JSS 103, 2015.

モデル駆動IoT開発の例（つづき）

- モデル変換とコード生成
  - 関心事ごとのドメイン特化言語
  - マッピングとリンク



P. Patel and D. Cassou, Enabling High-level Application Development for the Internet of Things, JSS 103, 2015.

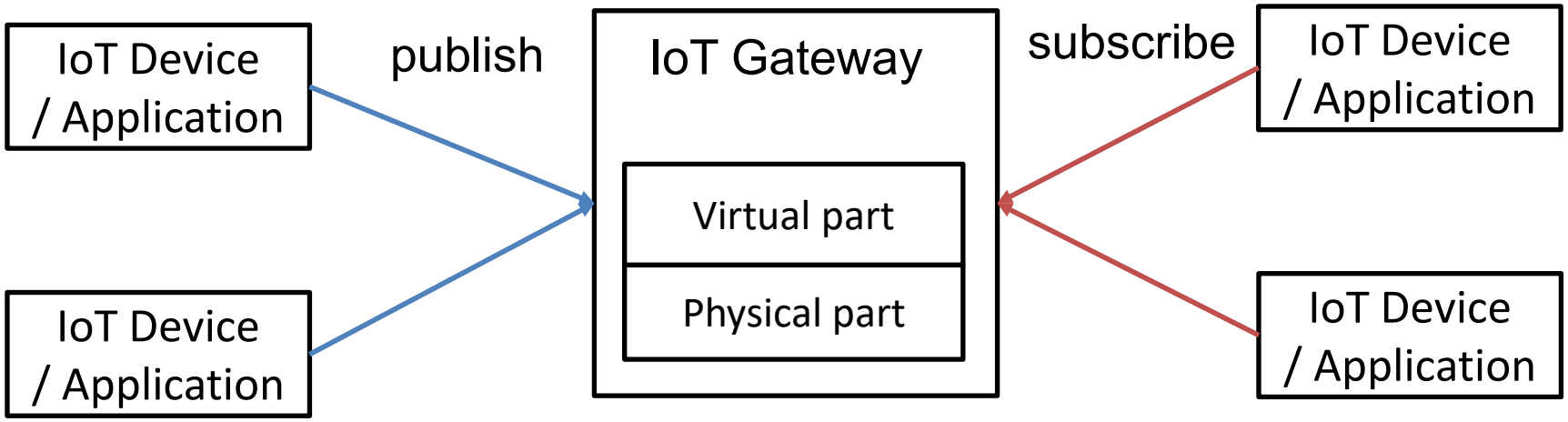
IoTアーキテクチャ・デザインパターンの体系的調査

- 抽象度の異なる様々なIoTアーキテクチャ・デザインパターンは、十分に分類および研究されていない
- RQ1.学術的な文献では、IoTのアーキテクチャや設計パターンはどのように扱われているのか？
  - **IoTアーキテクチャ・デザインパターンの関連論文32編**
- RQ2.既存のIoTアーキテクチャやデザインパターンは本当に全てIoTパターンなのか？
  - **抽出された143パターンのうち、57%が非IoTパターン**
- RQ3.IoTアーキテクチャ・デザインパターンは分類できるか？
  - 抽象化レベル、ドメイン、品質特性に沿って分類可
- RQ4.IoTアーキテクチャ・デザインパターンはどのようなものがあるのか？
  - **多くのIoTパターンは、相互運用性、セキュリティ、保守性を扱う**
  - **多くのIoTアーキテクチャパターンはドメイン固有**

Hironori Washizaki, Shinpei Ogata, Atsuo Hazeyama, Takao Okubo, Eduardo B. Fernandez, Nobukazu Yoshioka, “Landscape of Architecture and Design Patterns for IoT Systems,” IEEE Internet of Things Journal, Vol. 7, No. 10, pp.10091 – 10101, 2020

例: IoTゲートウェイ・イベントサブスクリプション

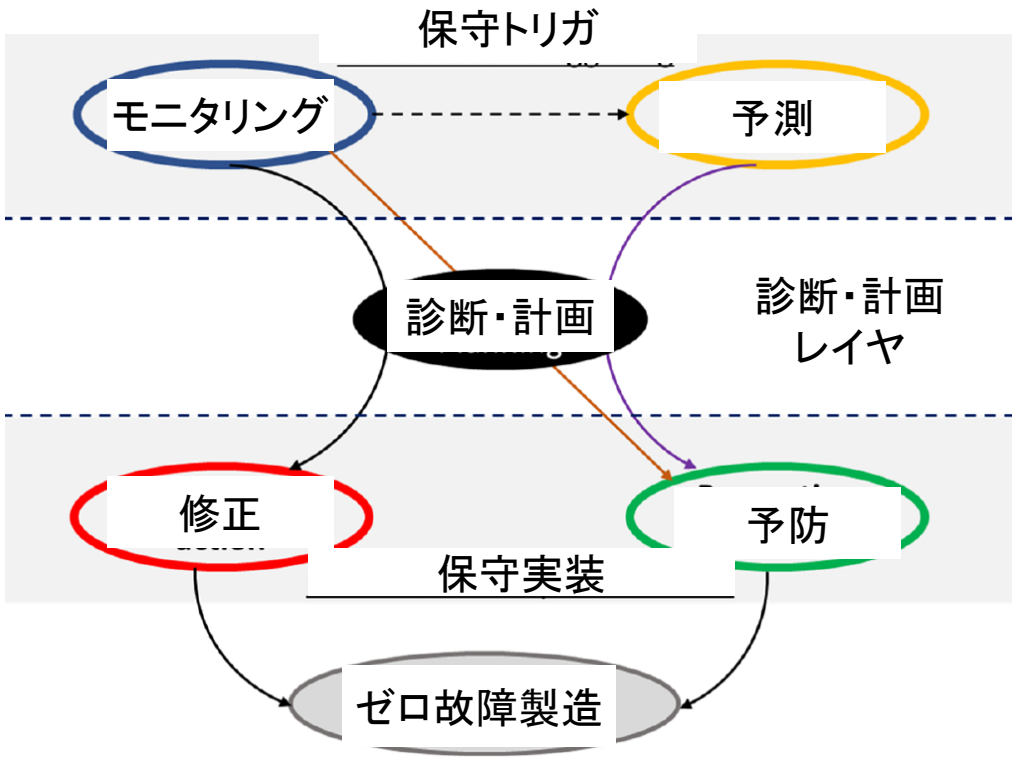
- IoTゲートウェイに**サブスクリプションの**仕組みを採用
- センサによって得られたデータと要素間メッセージを**非同期かつ相互に伝送**可能

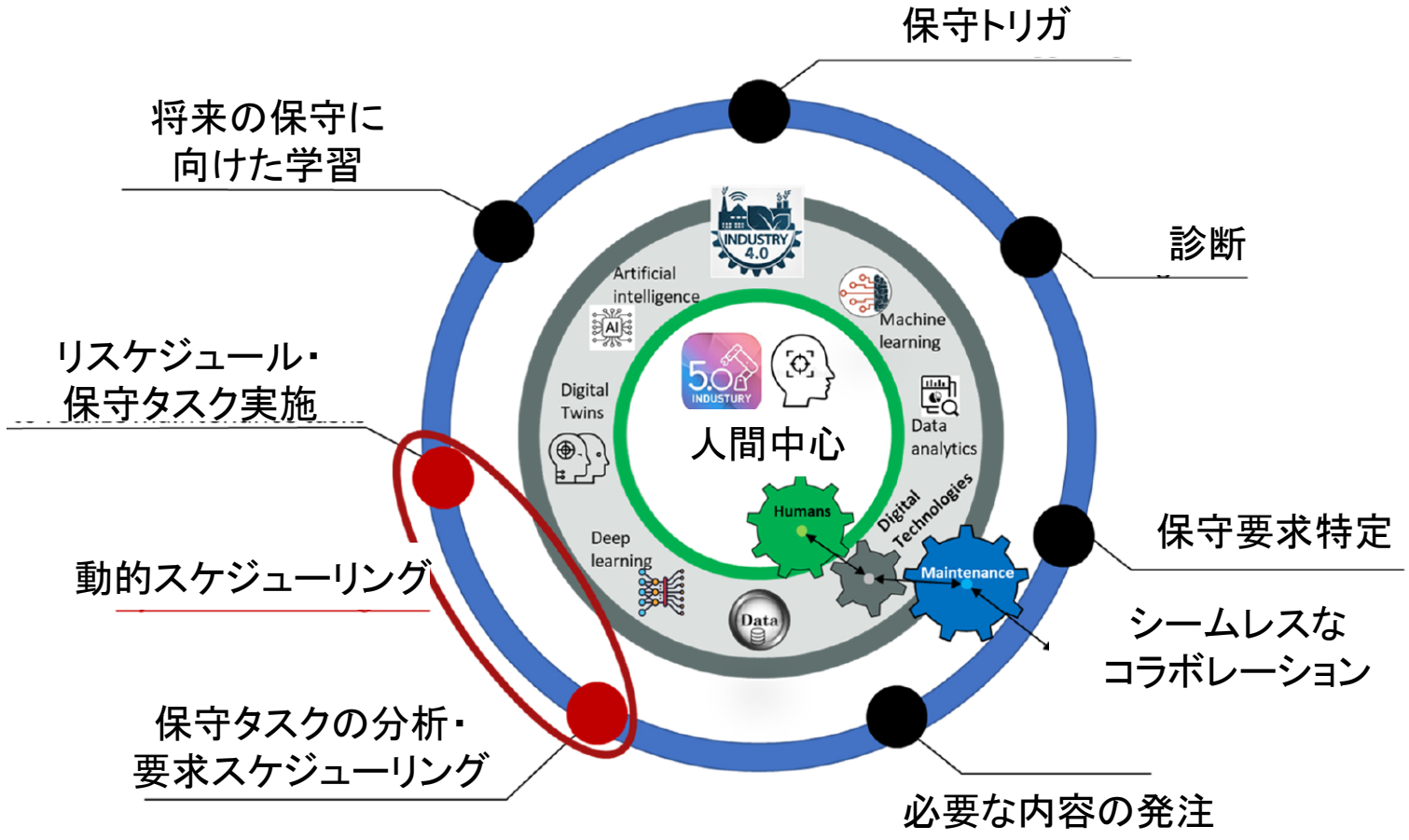


R. Tkaczyk, K. Wasielewska, M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and G. Fortino, “Cataloging design patterns for internet of things artifact integration,” in 2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018

製造業における Maintenance 5.0

- 人間中心の戦略と、AI主導の戦略の統合
- ZDM（ゼロ故障製造）システムにおける効率的で持続可能な保守実現





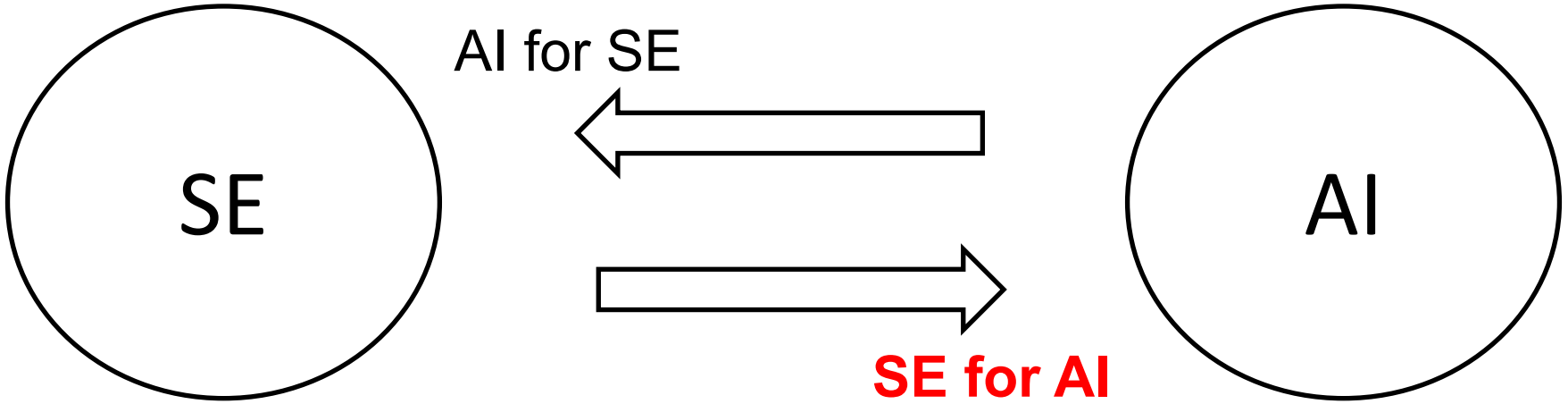
Foivos Psarommatis, Gokan May, Victor Azamfirei, Envisioning maintenance 5.0: Insights from a systematic literature review of Industry 4.0 and a proposed framework, Journal of Manufacturing Systems 68 (2023) 376–399

目次

- SWEBOK V4
- アジャイル
- IoTソフトウェアエンジニアリング
- AI/MLソフトウェアエンジニアリング
- まとめ

AI/MLとソフトウェアエンジニアリング

- ソフトウェアエンジニアリングにおけるAIの応用（AI for SE）
  - 人間の開発者の行動を再現して高品質・効率的開発
  - 曖昧な要求の解決から保守性の予測まで全開発段階に
  - 特に、欠陥予測、テストケース生成、脆弱性分析、プロセス評価など
  - 課題: 不確実で確率的な挙動
  - 課題: 十分にラベル付けされ構造化されたデータセットの必要性
- AIシステムのためのソフトウェアエンジニアリング（SE for AI）
  - 振る舞いが学習データに基づく点が従来とは異なる
  - データサイエンティストとソフトウェアエンジニアの学際的共同チーム
  - 大規模で変化するデータセットに焦点を当てたソフトウェア進化
  - 倫理・公平性 要求工学
  - 機械学習デザインパターンなど



Hironori Washizaki, et al., The Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), V4, 2023

AI for SEの傾向

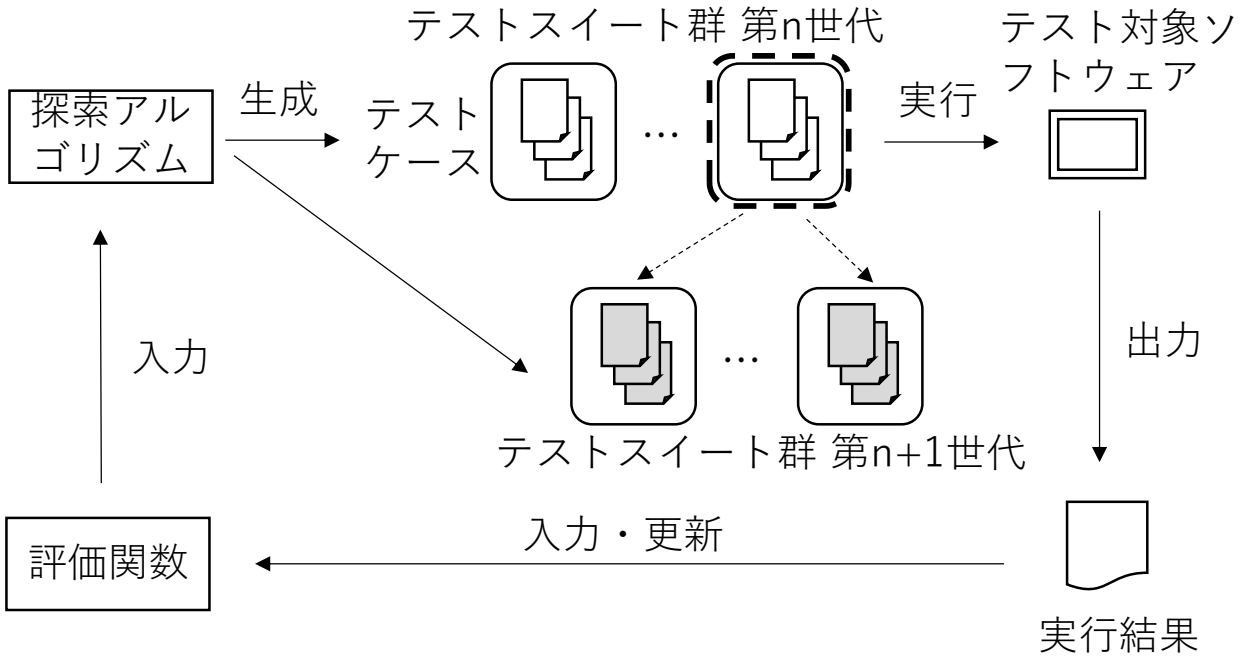
- 背景
  - 社会の基盤としてソフトウェアシステムが大規模化および複雑化，繋がり，極端に不確実
  - 探索的な開発や頻繁な変更，拡張，環境適応
- 品質の作り込みや検証・評価・改善・管理の活動も自ずと探索的および適応的
  - 汎化性能を備えた機械学習による新たな対象や状況へのデータ駆動による適応
  - メタヒューリスティクスである遺伝的アルゴリズム（Genetic Algorithm; GA）による探索

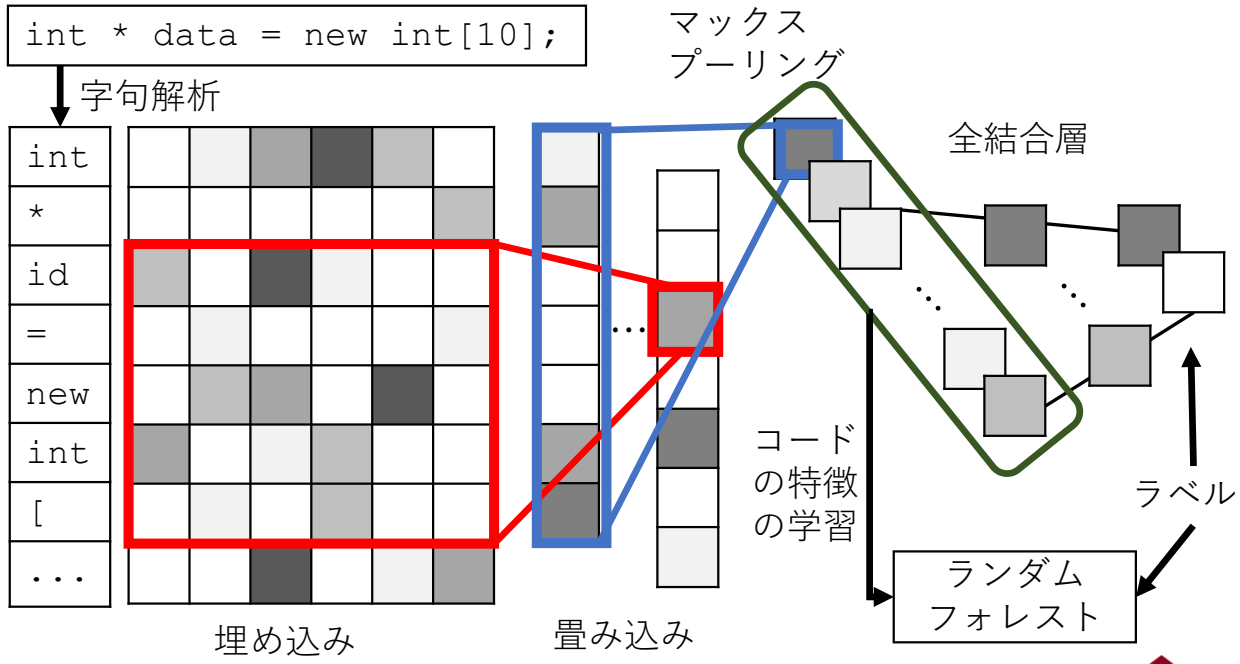
目的	要求	設計・実装	テスト・品質管理	保守
予測	要求品質評価	多目的設計品質予測、改訂予測	テスト工数予測 欠陥予測 品質評価	保守工数予測
特定	要求分類 要求検証	アーキテクチャ自己適応 再利用	テスト優先順位 欠陥限局 脆弱性検出 欠陥報告管理	デザインパターン検出 追跡関係特定
変換	要求要約	自動プログラミング	テスト生成 自動プログラム修正	リファクタリング

鷲崎弘宜, 機械学習を中心としたAI活用によるソフトウェアの品質保証, システム/制御/情報, Vol. 66, No. 5, pp. 1-7, 2022.

テストにおけるAI活用

- テストの計画: テスト工数や欠陥の高精度な予測
- テストの設計と実施:テストケース生成、探索的なサーチベーストテスト
- テストの繰り返し: テストケース優先順位付けとリグレッションテスト
- 個別テスト技術
  - UIテスト: UI変更を検出しテスト自動修正
  - セキュリティテスト: 脆弱性検出



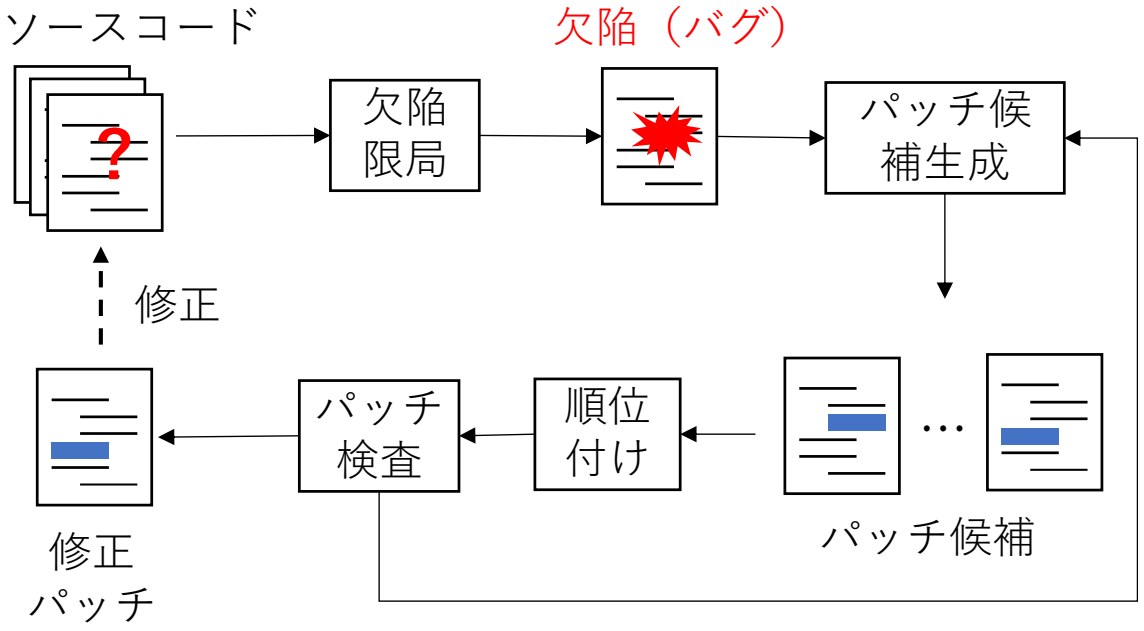


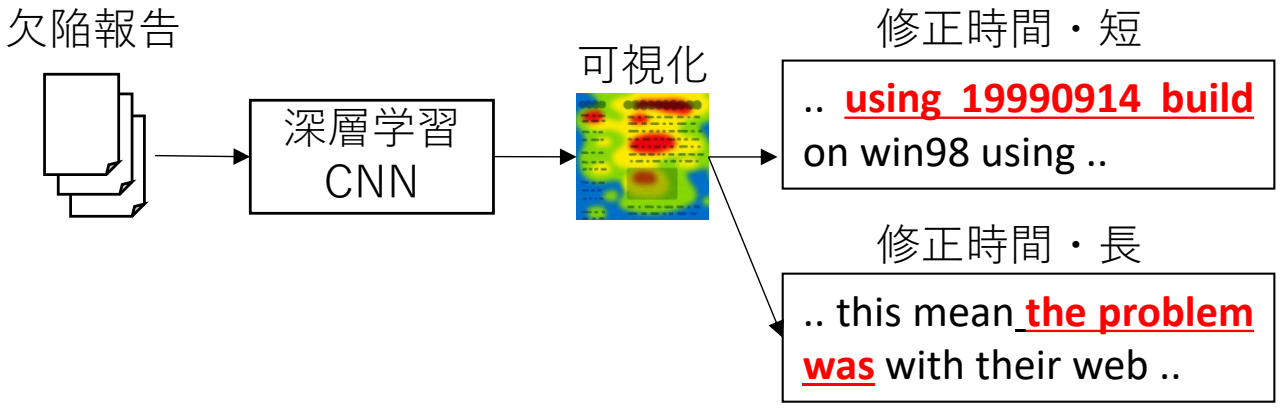
鷲崎弘宜, 機械学習を中心としたAI活用によるソフトウェアの品質保証, システム/制御/情報, Vol. 66, No. 5, pp. 1-7, 2022.

R. Russell, et al., “Automated Vulnerability Detection in Source Code Using Deep Representation Learning,” ICMLA 2018.

デバッグ、品質管理におけるAI活用

- 欠陥限局（Fault Localization; FL）
- 自動プログラム修正（Automated Program Repair; APR）
- 品質評価および予測
- 欠陥報告管理



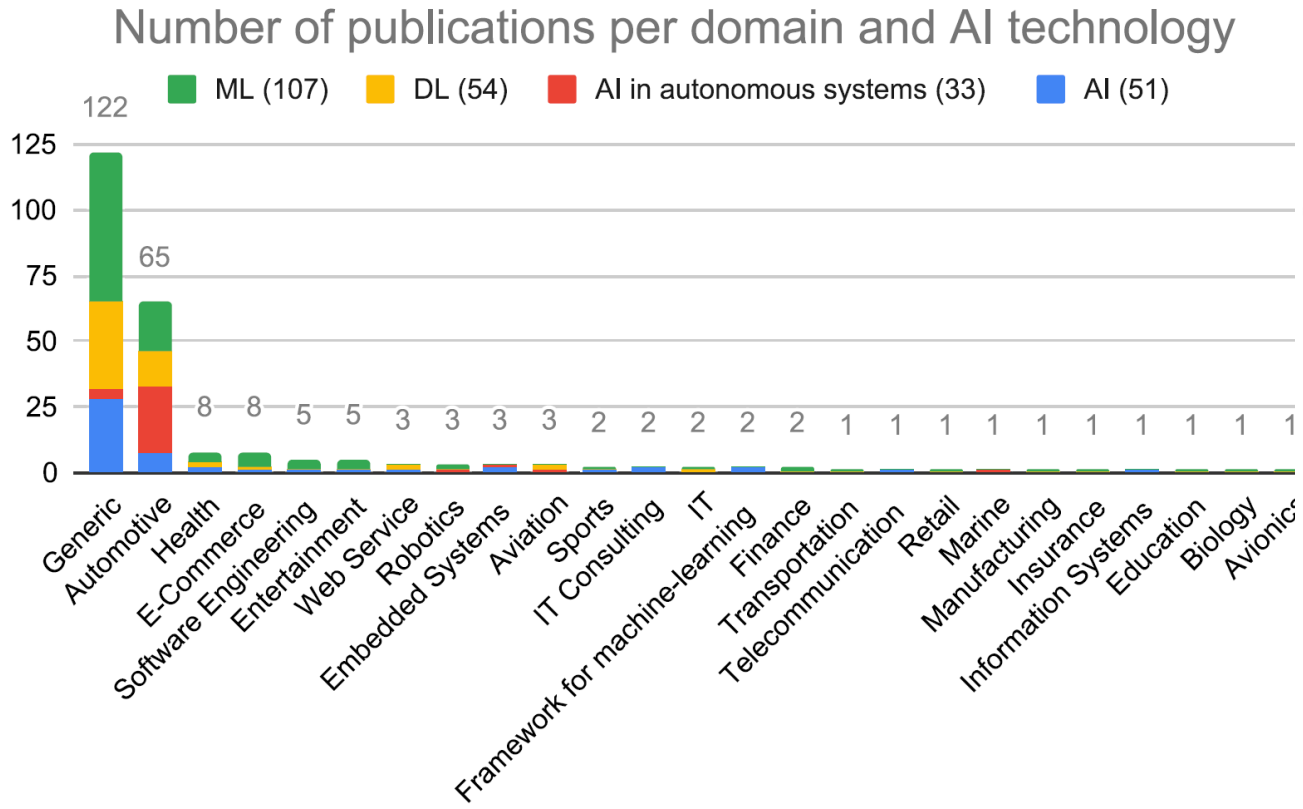


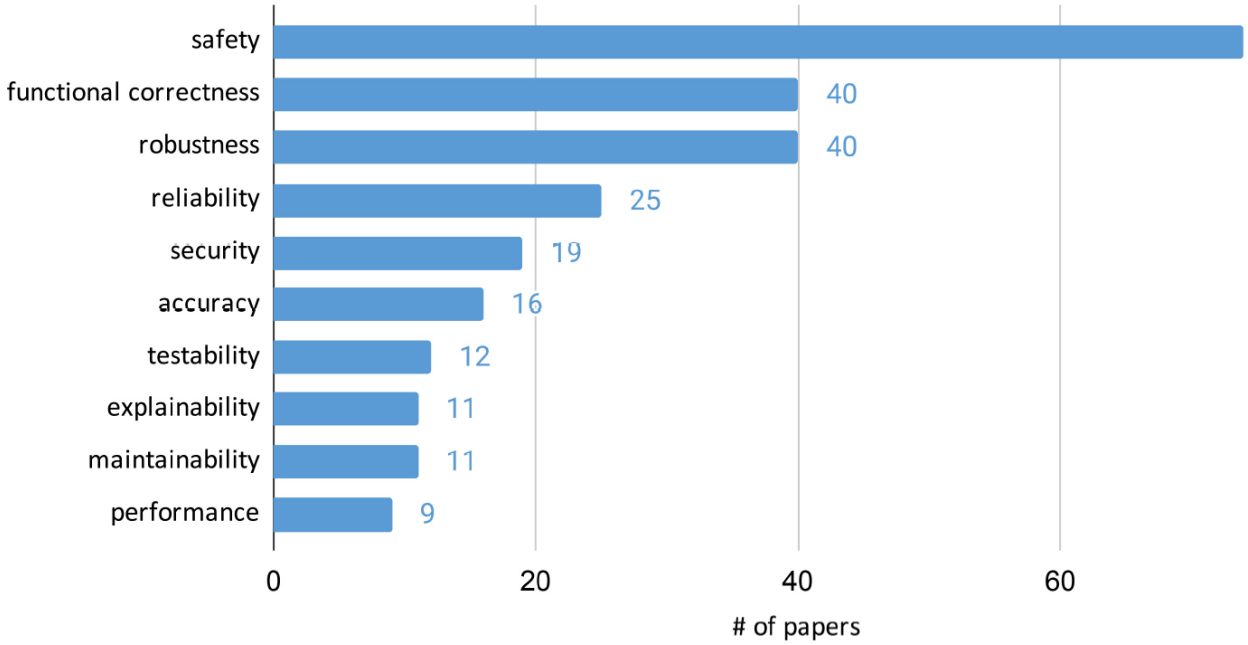
X. Li, et al., “DeepFL: Integrating Multiple Fault Diagnosis Dimensions for Deep Fault Localization,” ISSTA 2019.

Y. Noyori, et al., “Extracting features related to bug fixing time of bug reports by deep learning and gradient-based visualization,” ICAICA 2021.

SE for AIの傾向

- 調査: Software Engineering for AI-Based Systems: A Survey (TOSEM'22)
  - 2010-2020に248編の論文、うち2/3以上は2018年以降
  - 対象: 一般、ならびに自動車が大部分
  - 品質特性: ディペンダビリティ、セーフティの扱いが多い
  - 領域: テスト、品質が多い。保守などは軽視。データ関連が頻出研究課題。





Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, Stefan Wagner: Software Engineering for AI-Based Systems: A Survey. ACM Trans. Softw. Eng. Methodol. 31(2): 37e:1-37e:59 (2022)

SE for AIの傾向: 領域別

領域	数	傾向
要求	17	• AIに特化した新しい品質特性への注目 • 確率的な結果や曖昧さを扱う仕様や表記法（例：部分仕様） • AIのための要求工学プロセスの取り組みは限定的
設計	34	• 安全性や信頼性などに対応する設計戦略 • マイクロサービスとしてのモデル共有など、具体的なAI基盤 • システムレベルでは**パターン、設計標準、リファレンスアーキテクチャ**の提案は少ない
構築	23	• 構築上の課題とガイドライン • ツールやプラットフォームの提案あり、成熟度や、選択の根拠、採用など不十分
テスト	115	• テストケース（55件）、うちテストケース生成（40）とテストケース選択（12） • テスト手法: **メタモルフィックテストが最多（16件）**、ファジング（6）、ミューテーションテスト（5） • テストメトリクスに関する論文では新規カバレッジ基準提案が多い（14）
保守	6	• 限定的 • AIシステムのバグや**デバッグ**関連。

Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, Stefan Wagner: Software Engineering for AI-Based Systems: A Survey. ACM Trans. Softw. Eng. Methodol. 31(2): 37e:1-37e:59 (2022)

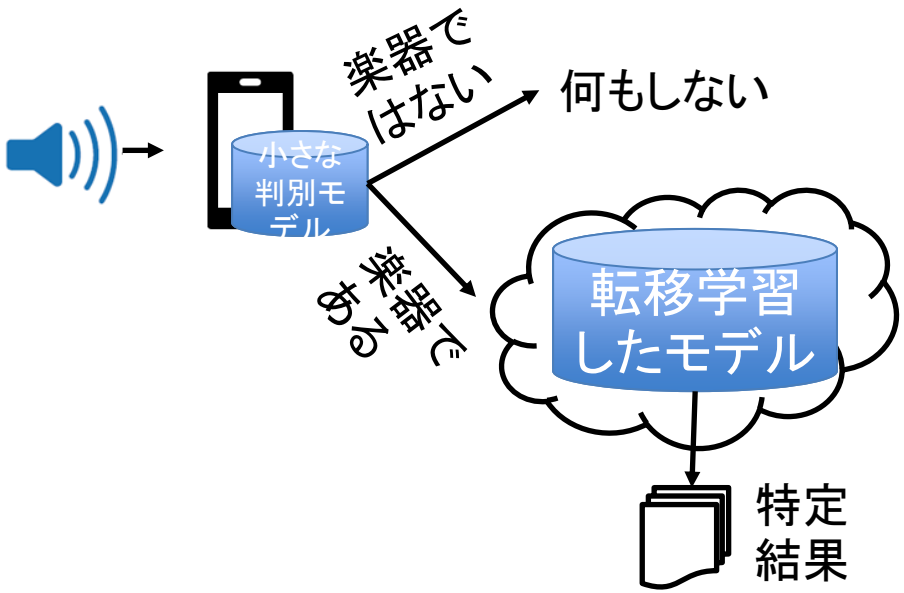
SE for AIの傾向: 領域別（つづき）

領域	数	傾向
プロセス	31	• ソフトウェアエンジニアやデータサイエンティストを含む学際的なチーム形成 • 多くのプロセスは大企業によるAIシステムの初期の経験に基づくアドホックな構築 • モデルレベルでのAIパイプラインに焦点を当てた研究あり（6件） • プロセス関連のサポートとしてツール（6）やフレームワーク（3）
モデル・手法	38	• AIコンポーネントを用いたCPSの検証と妥当性確認（18件） • セーフティ、自動運転領域の取り組みが多い
品質	59	• ISO 25000やISO 26262などの国際規格の更新必要性の提起 • 多くはMLの品質特性、フレームワーク、品質保証、認証に焦点
その他	4	• マネジメント、経済、構成管理、プロフェッショナルプラクティス

Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, Stefan Wagner: Software Engineering for AI-Based Systems: A Survey. ACM Trans. Softw. Eng. Methodol. 31(2): 37e:1-37e:59 (2022)

設計と保守に向けた機械学習デザインパターン

- スマホで拾った音について楽器の種類を特定し、種類に応じた記録や対応を実現したい。
- しかし、スマホのメモリや性能は限られており、大きな深層学習モデルは載りそうにない。どうすればよいか？



- スマホ上の小さなモデルでは音が楽器かどうか判定し、楽器の場合にのみクラウド上の大きなモデルで種類を特定する **2段階予測** としよう。
- 特定にあたり、様々な音データによる既存の訓練済みモデルを用いて**転移学習** したモデルを用いよう。

パターン	問題	解決
2段階予測 Two-Phase Predictions	もともと大規模で複雑なモデルの性能を、エッジデバイスや分散デバイスにデプロイした場合も維持する必要性	利用の流れを2つの段階に分けて、シンプルな段階のみをエッジ上で実行
転移学習 Transfer Learning	複雑な機械学習モデルを訓練するために必要となる大規模データセットの不足	訓練済みモデルの一部の層を取り出して重みを凍結し、訓練対象とせずに類似問題の解決のため新モデルで利用

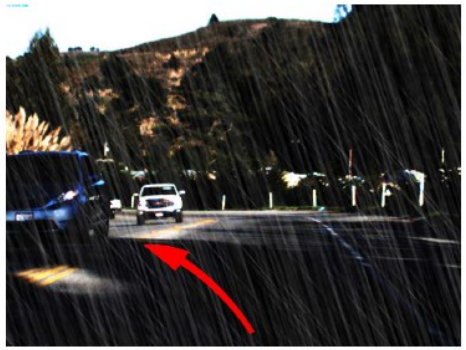
Valliappa Lakshmanan, Sara Robinson, Michael Munn著, 鷲崎弘宜, 竹内広宜, 名取直毅, 吉岡信和 訳, “機械学習デザインパターン – データ準備、モデル構築、MLOpsの実践上の問題と解決”（原著 “Machine Learning Design Patterns”）, オライリージャパン, 2021. https://www.amazon.co.jp/dp/4873119561

メタモルフィックテスト

- メタモルフィック関係に基づき大量に試験
- 入力への変化により、出力の変化を予想できる関係
  - 例: A の検索結果数 ≧ A かつ B の検索結果数
  - 例: sin(x) = sin(x + 360度)

入力の変化	出力の変化
並び変え	無し
ノイズの追加	
意味的に同じもの	
統計的に同じもの	
経験的に近いもの	僅か
定数の加算、乗算	定数の加算、乗算
狭める	部分集合
全く異なるもの	互いに素

自動運転車の場合 [Tian'18]



Y Tian, et al., DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars, ICSE 2018
https://arxiv.org/pdf/1708.08559.pdf

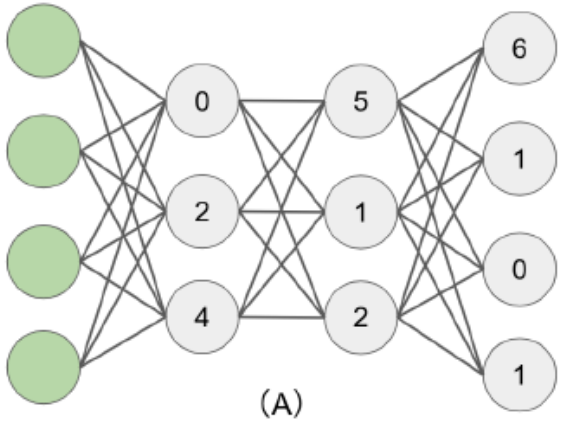
参考: S. Segura et al., "Metamorphic Testing of RESTful Web APIs," IEEE Transactions on Software Engineering, 2017
参考: C. Murphy, "Applications of Metamorphic Testing", http://www.cis.upenn.edu/~cdmurphy/pubs/MetamorphicTesting-Columbia-17Nov2011.ppt

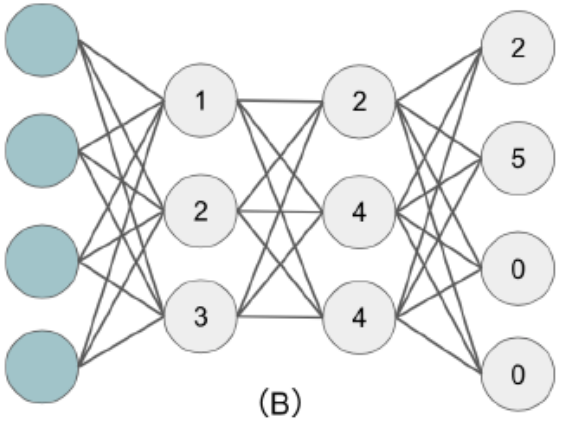
39

ニューラルネットワークデバッグ（修正）

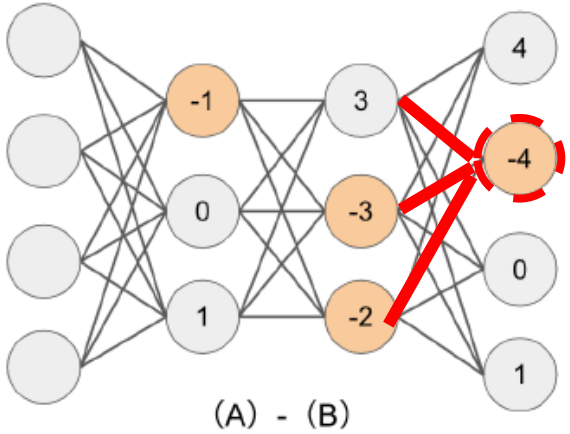
- 再訓練, オンライン学習, データ拡充: 生成 [a]、選択 [b]、拡張 [c]など
- 特定サンプルに応じたパラメータ直接変更 [d][e]

検証データによる平均値





誤認識データによる平均値



最優先で修正すべきニューロン

[a] Generative adversarial nets, NIPS 2014
[b] MODE: automated neural network model debugging via state differential analysis and input selection, ESEC/FSE 2018
[c] Autoaugment: Learning augmentation policies from data, arXiv:1805.09501, 2019,
[d] 松井 健, 鵜林 尚靖, 佐藤 亮介, 亀井 靖高, 敵対的サンプルに対するニューラルネットワークモデルの学習無し修正とその評価, JSSST大会 2019,
[e] Search Based Repair of Deep Neural Networks, arXiv:1912.12463, 2019

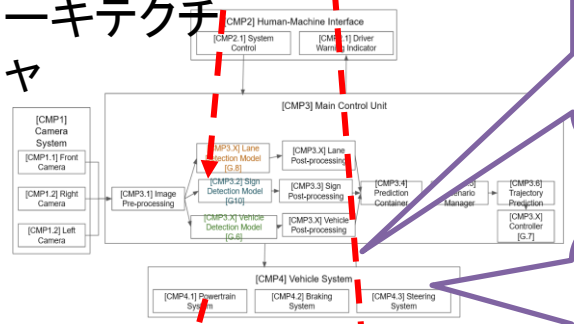
高信頼ML開発にフレームワークはなぜ必要なのか？

従来の開発

要求・制約

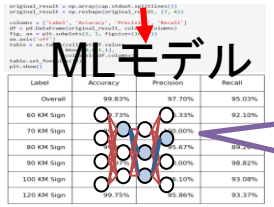
1. Prioritization for 60KM signs accuracy using DNN Repair tools should also be conducted for the blinking scenarios while protecting ~~higher~~ **lower** speed sign accuracy. (誤り)
2. ~~Data augmentation should be employed to emulate blurry 60 KM sign data.~~ (過剰)

システムアーキテクチャ



データ・コード・結果

データ



MLモデル

安全性を含む様々な側面間の整合性を維持したままでの開発保守困難

追跡性を欠き、一貫したままでの改訂が難しい

システム及びMLコンポーネントの安全性解析対策困難

ソリューションの選択や適用に根拠を欠く

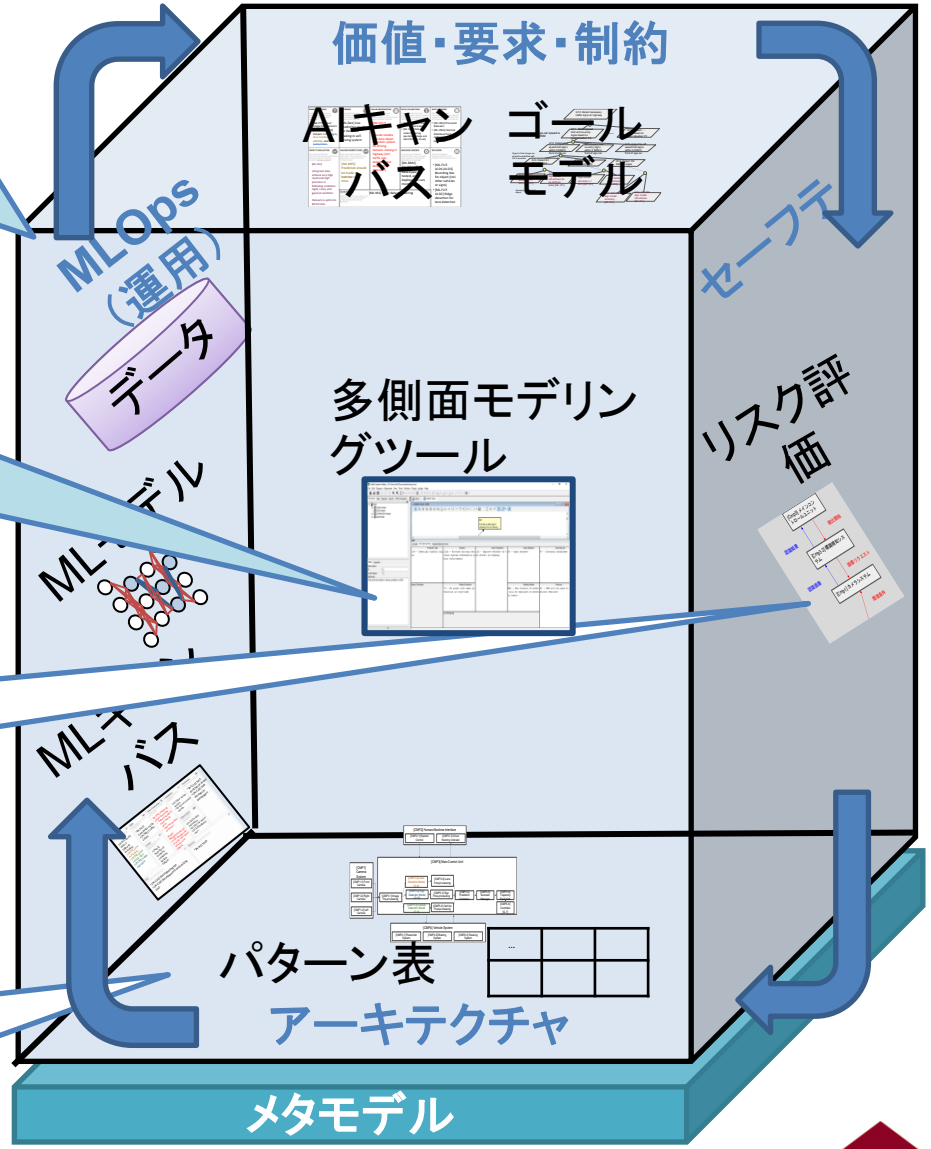
A. 安全性を組み入れた多面的分析・設計プロセス: 現場一気通貫採用可、変更への適応・改訂容易

B. 多面的・統合モデリングツール: 複雑なAI要求の顕在・整合化

C. 安全性解析・リスク評価: 安全・信頼を主軸に

D. ソリューションガイド: eAI技術群の使いこなし

フレームワークに基づく開発



Jati H. Husen, Hironori Washizaki, Hnin Thandar Tun, Nobukazu Yoshioka, Yoshiaki Fukazawa, Hironori Takeuchi, Hiroshi Tanaka, Kazuki Munakata, “Extensible Modeling Framework for Reliable Machine Learning System Analysis,” 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN’23)

A/B. モデリングとMLパイプライン統合の流れ

安全性・高信頼性を含む要求・アーキテクチャモデリング

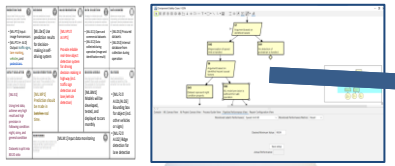
astah System Safety

DNN訓練・評価・修正パイプライン

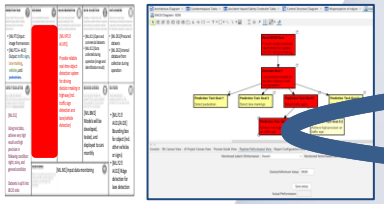
DVC

具体化・洗練

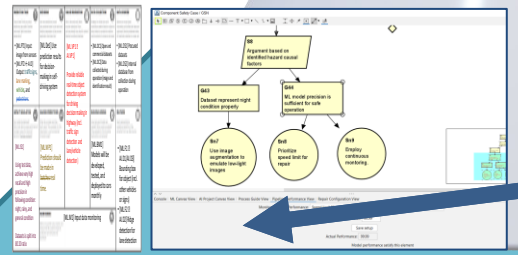
要求分析・設計



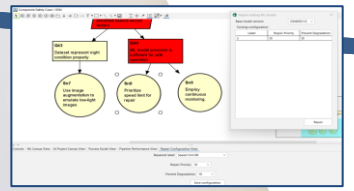
問題の可視化



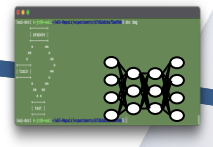
解決の可視化



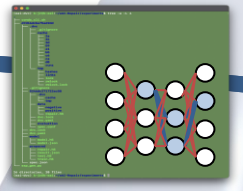
修正戦略の追加



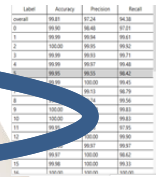
DNN訓練



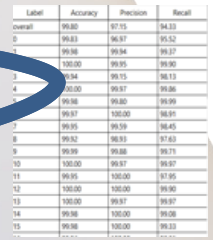
DNN修正

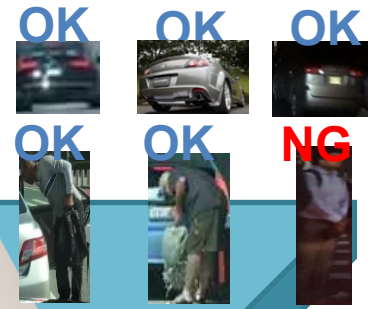


DNN評価



DNN評価





メタモデル

Jati H. Husen, Hironori Washizaki, Hnin Thandar Tun, Nobukazu Yoshioka, Yoshiaki Fukazawa, Hironori Takeuchi, Hiroshi Tanaka, Kazuki Munakata, “Extensible Modeling Framework for Reliable Machine Learning System Analysis,” 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN’23)

目次

- SWEBOK V4
- アジャイル
- IoTソフトウェアエンジニアリング
- AI/MLソフトウェアエンジニアリング
- まとめ

まとめ

- SWEBOK Guide V4
  - 新知識領域: アーキテクチャ、運用、セキュリティ
  - 現代的な取り組みと技術領域拡充: アジャイル、IoT、AI/ML
- アジャイル保守
  - 軽量な文書化、頻繁なテストなど
  - 実現に有用なアジャイル品質パターン QA to AQ、アジャイルプロセス測定評価
- IoTソフトウェアエンジニアリングと保守
  - 課題と取組み: 通信、相互運用性、セキュリティ、データとプライバシ、IoT開発考慮
  - モデル駆動IoT開発、IoTデザインパターン
  - 製造業におけるMaintenance 5.0: 人間中心とAIの統合
- AI/MLソフトウェアエンジニアリングと保守
  - AI for SE: 欠陥予測、テストケース生成、脆弱性分析など
  - SE for AI: 機械学習デザインパターン、テスト・デバッグ、パイプライン統合
- さらなるソフトウェア保守・進化に向けて
  - アジャイル保守プロセスとしての「派生開発」の価値再発見
  - IoT・AIエンジニアリング基盤との接続
  - パターン化と測定評価・改善へ