

SWEST23 (2021)

ソフトウェア開発を もっとうまくやってみた

～実践でのソフトウェアエンジニアリングを
活用した問題解決や改善事例～



みずのり (水野のりゆき)

@TEF道、TOCfE北海道、RDRA MeetUpなど

自己紹介とか



水野昇幸 (みずののりゆき)

現所在: 苫小牧市
※お花やさんです

- @NoriyukiMizuno ※Twitter ID
- Jack of all trades: なんでも屋/器用貧乏
- 主に個人事業主とかどこかの契約でお手伝いとか花屋
- JaSST北海道実行委員、ETロボコン実行委員、TEF道
- TOC/TOCfE北海道お世話役
- 趣味: テストなどのモデリング研究
- テスト設計コンテスト2017優勝
- 国際学会発表: 6WCSQ@ロンドン2014
InSTA@東京, 2017 @Sweden, 2018 @西安, 2019
- 簿記3級、2級、JTSQB-FL、ALTM、ALTA、情報処理エンベデッド、プロマネ
- TOC-CCPMスペシャリスト(インプリメンター資格)

大学は仙台

自己紹介：コミュニティ/発表

発表や公開している内容はこちら

ET関連

- ・ ETロボコン実行委員を2008～（ここ数年は幽霊的状况）
- ・ ET-WEST2018：なぜ組込みシステムにおけるテストは難しいのか
- ・ ET-WEST2014：組込み開発でのSWテスト自動化のライフサイクル

ソフトウェアテスト/テスト自動化関連

- ・（共著）InSTA2019@西安：Coexistence of test execution efficiency and test
- ・（共著）InSTA2018@Sweden：Proposal for Enhancing UTP2 with Test Aspects
- ・（共著）InSTA2017@東京：Test Conglomeration - Proposal for Test Design Notation like Class Diagram
- ・ STAC2015：自動家は見た～自動化の現場の真実～
- ・ テスト設計コンテスト2017優勝：テストカタマリーを活用したテスト設計プロセス

TOC（制約理論：Theory of Constraints）およびマネジメント関連

- ・ SaPIDTOC～未来予想型チーム運営ワークショップ
- ・ SQiP2014：CCPMの考え方を活用したテストフェーズにおける課題解決～課題発見、解決までのケーススタディ～
- ・ プロジェクトマネジメントの概要とCCPMによる工期短縮の仕組み
- ・ CCPMのおはなし、CCPM折り紙ワークショップ
- ・ 提案を検証する技術（TOC マフィアオファー（URO）活用技術）

自己紹介：業務関連



2004

2016 2017

2021

主に会社員

<組込み、System of Systems>

- 宇宙系等の通信システム（複数）
 - ・ 要件定義、一部組込みSW開発
 - ・ システムテスト、自動テスト環境構築
 - ・ 保守サポート、マネジメント

<その他、プロセス改善系>

- 組織内プロセスの改善活動
 - ・ テストプロセスの改善

お休み
(勉強タイム)

主に個人事業

<エンプラ系>

- EC、人事向けシステム
 - ・ 要件定義支援、仕様検討
- レンタルビジネス基幹システム
 - ・ 要件定義及びテスト設計支援

<製造業系>

- 車載向けシステム
 - ・ 設計、テストのプロセス整備
- 工場設備管理/DXシステム
 - ・ 要件定義およびマネジメント
 - ・ プロセス整備、テスト検討
- 上記DXシステムのIoT拡張
 - ・ システム設計、マネジメント

自己紹介：業務関連

2004

2016 2017

2021

主に会社員

- <組込み、System of Systems>
■ (比較的お堅めの) (複数)
・ 要件定義、組込みsw開発
・ システムテスト、自動テスト環境構築
・ 保守サポート、メンテナンス
- 組織で活動しつつ、
自主的に改善活動を実施

- <その他、プロセス改善系>
■ 組織内プロセスの改善活動
・ テストプロセスの改善

6WCSQ-2014@ロンドン：
Stepwise Test Design Method

SQIP2014：CCPMの考え方を活用した
テストフェーズにおける課題解決

お休み
(勉強タイム)

主に個人事業

- <テンプラ系>
■ 比較的自由な立場で、
組織の枠にとらわれずに
開発やマネジメントを支援
・ 要件定義及びテスト設計支援
組織サポートはないため、
自主的に各種手法を学習
- 車載向けシステム

InSTA2019@西安
InSTA2018@Sweden
InSTA2017@東京

- ・ プロセス整備、テスト検証
■ 上記
・ シス
- JaSST21東京チュートリアル
価値につながる要件・仕様から
テストを考える
※ET2021でも発表予定

今回の発表が役立つような対象の方

- 以下のような方は今回の発表が役立つ可能性があります。
 - ・ 組織の開発プロセスや仕組みに疑問をもたれている方
 - ・ 現状より「もっとうまくできる」のではないかと感じている方
 - ・ 現場や組織の改善へのヒントを探している方
 - ・ ソフトウェアエンジニアリングに多少なりとも興味がある方
- (SWESTまで来ていないとは思いますが...) 対象外の方
 - ・ 現状の開発で十分、完璧と感じられている方

コンテンツとしては以下となります。

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

幕間：ソフトウェアエンジニアリングに関わるあれこれ

事例2：IoTシステム構築のPoCへ向けた
モデリングとマネジメント事例

事例3：テストの知見を活用した改善事例

さいごに：ソフトウェアエンジニアリングの活用方法

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 背景

- ・ 完全新規のWebプロダクトを開発し、利用予定の顧客へ売り込み予定
- ・ アジャイル的な段階的・反復的（月一リリース）に開発を進めている

■ 課題

- ・ プロダクトの構築はしたが、本当に必要なものが不明である
- ・ 顧客側から複数の解決策が出てくるが、どれが本当によいかわからない
- ・ （アジャイル的なので）仮実装をしてもよいが、案が複数あり手間がかかる

事例1：（ペーパー）プロトタイピング活用



事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 実際の進め方

- ヒアリングで複数案を出す
- 複数案を簡易に絞り込み
- （ペーパー）プロトタイプで表現
- （顧客込み）評価して決定する

事例1：（ペーパー）プロトタイピング活用

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 実際の進め方（タスクのスケジュール・カレンダー連携機能の具体的な例）

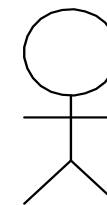
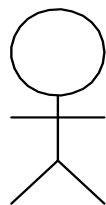
- ・ ヒアリングで複数案を出す
- ・ 複数案を簡易に絞り込み
- ・ （ペーパー）プロトタイプで表現
- ・ （顧客込み）評価して決定する

予定の移動方法（ドラッグによる移動）

開始終了時間同時変更機能

作業日時一斉変更機能

作業期限を自動で再設定する機能



※実画面から自由にまずは意見

事例1：（ペーパー）プロトタイピング活用

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 実際の進め方（タスクのスケジュール・カレンダー連携機能の具体的な例）

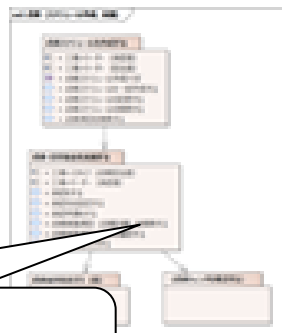
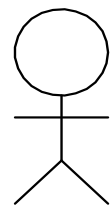
- ・ ヒアリングで複数案を出す
- ・ **複数案を簡易に絞り込み**
- ・ （ペーパー）プロトタイプで表現
- ・ （顧客込み）評価して決定する

予定の移動方法（ドラッグによる移動）

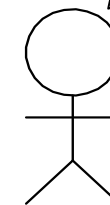
~~開始終了時間同時変更機能~~

作業日時一斉変更機能

~~作業期限を自動で再設定する機能~~



すでにある図や手書きプロトタイプ
を活用してその場でいくつかを絞り込み



事例1：（ペーパー）プロトタイピング活用

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 実際の進め方（タスクのスケジュール・カレンダー連携機能の具体的な例）

- ・ ヒアリングで複数案を出す
- ・ 複数案を簡易に絞り込み
- ・ **（ペーパー）プロトタイプで表現**
- ・ （顧客込み）評価して決定する

追加案も発生

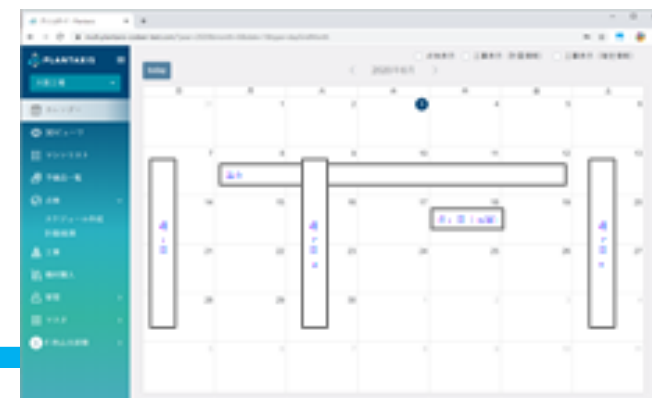
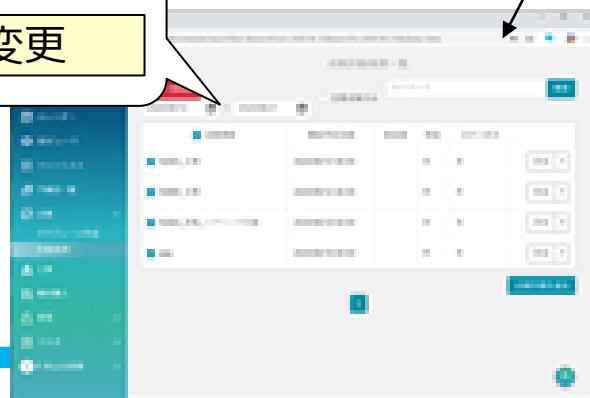
選択した作業の一括日程変更

予定の移動方法（ドラッグによる移動）

~~開始終了時間同時変更機能~~

作業日時一斉変更機能

~~作業期限を自動で再設定する機能~~



事例1：（ペーパー）プロトタイピング活用

事例1：（ペーパー）プロトタイピングを活用したプロセス事例

■ 実際の進め方（タスクのスケジュール・カレンダー連携機能の具体的な例）

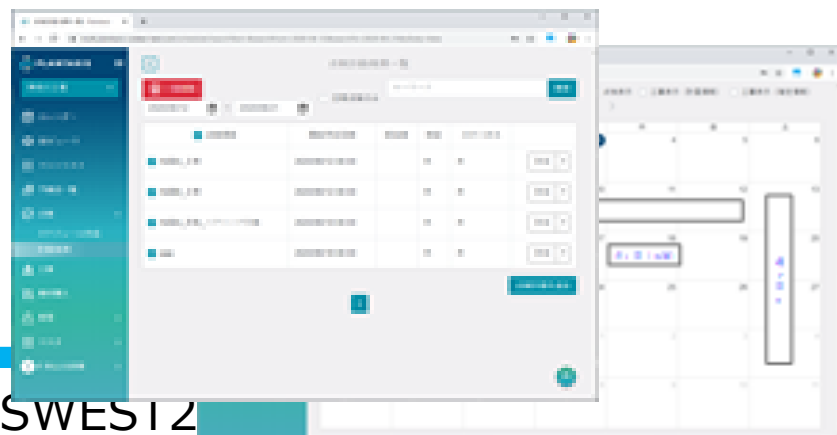
- ・ ヒアリングで複数案を出す
- ・ 複数案を簡易に絞り込み
- ・ （ペーパー）プロトタイプで表現
- ・ **（顧客込み）評価して決定する**
→ 関連して課題や背景を詳細確認できる

予定の移動方法（ドラッグによる移動）

作業日時一斉変更機能

追加案：選択した作業の一括日程変更

特定日の作業ができないケースに対応するなら、追加案でいけそうですね



事例1：（ペーパー）プロトタイピング活用

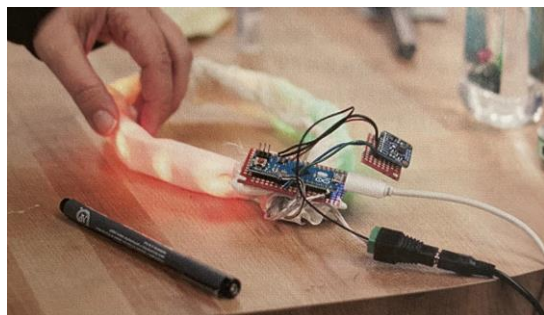
実際には、今回の手順は以下のように活用、応用することもできます。

- プロトタイピングを活用した手順
- ・ ヒアリングで複数案を出す
- ・ 複数案を簡易に絞り込み
- ・ （ペーパー）プロトタイプで表現
- ・ （顧客込み）評価して決定する

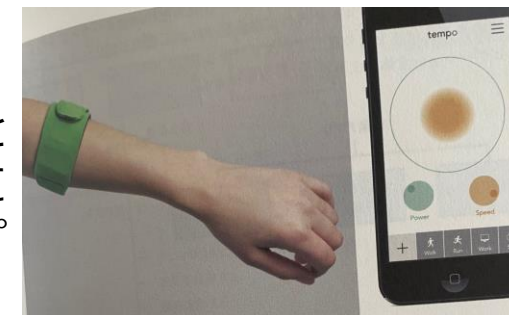
■ 応用例：

- ・ 実プロダクトでやることも可能
確定した部分まで開発をして
実プロダクトを拡張したプロトタイプで
ヒアリングする（インクリメント型）
- ・ HWを含めても同等の方法が利用可能
※ 下記参照

機能優先
ビジュアル非優先
プロトタイプ



デザインや操作を
込みでプロダクトに
近いプロトタイプ

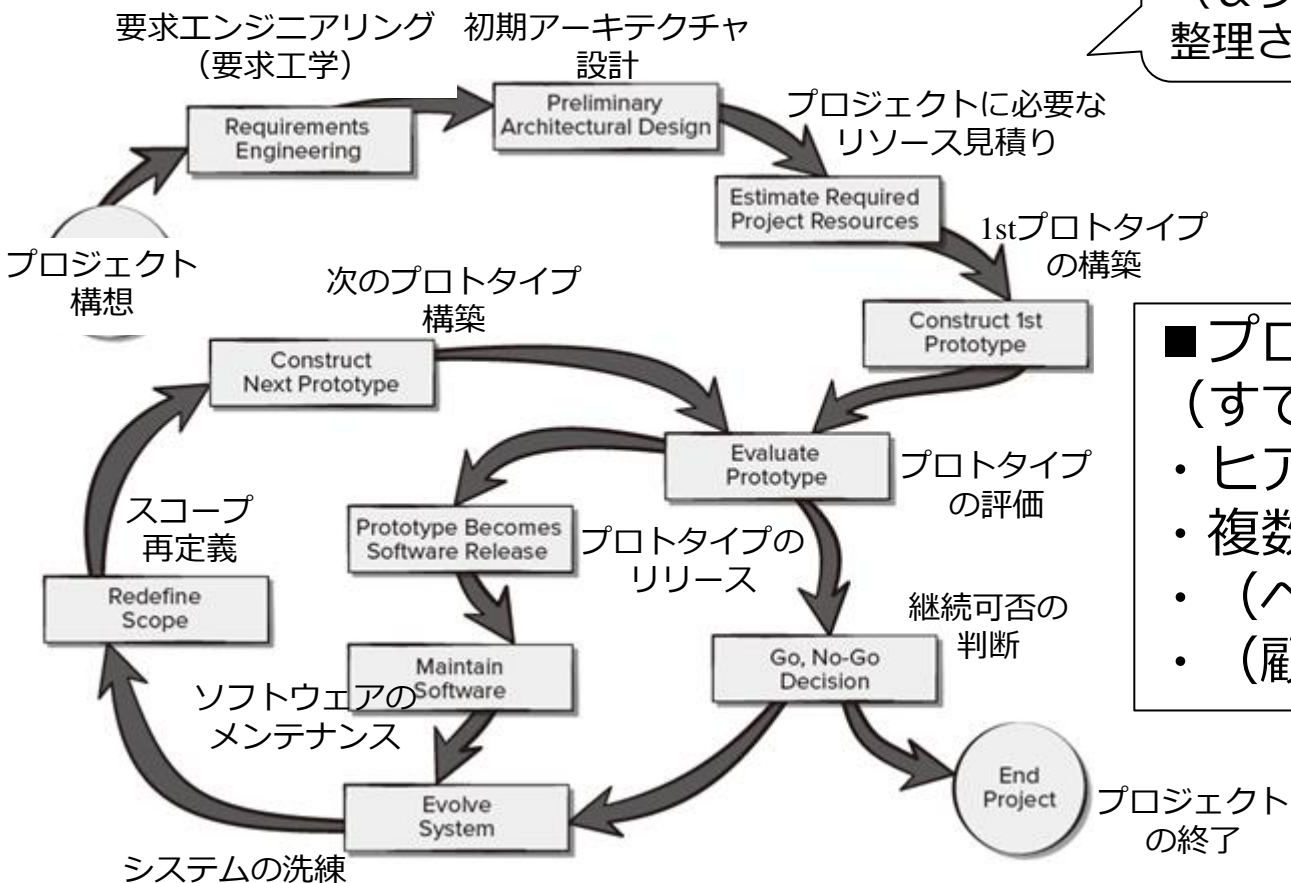


※ 「デザイナーのためのプロトタイピング入門」より引用

事例1：その後…

日本語訳を入れて比較

書籍（教科書）側の方がわかりやすく（&ライフサイクルの全体範囲まで）整理されていた

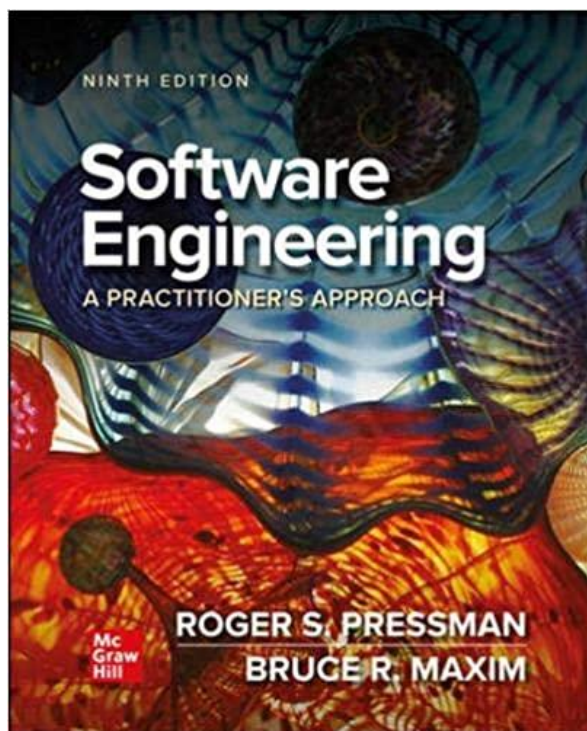


- プロトタイピングを活用した手順（すでに構築したプロダクトあり）
- ・ヒアリングで複数案を出す
- ・複数案を簡易に絞り込み
- ・（ペーパー）プロトタイプで表現
- ・（顧客込み）評価して決定する

Software Engineering A Practitioner's Approach 9th edition 第4章より引用

幕間：ソフトウェアエンジニアリングに関わるあれこれ

※今回は、下記のベストセラー/教科書的な書籍を対象として情報をまとめます。



- 最新版は9th Edition（2019/09発売）
- 今まで300万部出版（らしい）
- 第6版の日本語翻訳有（2005年）→

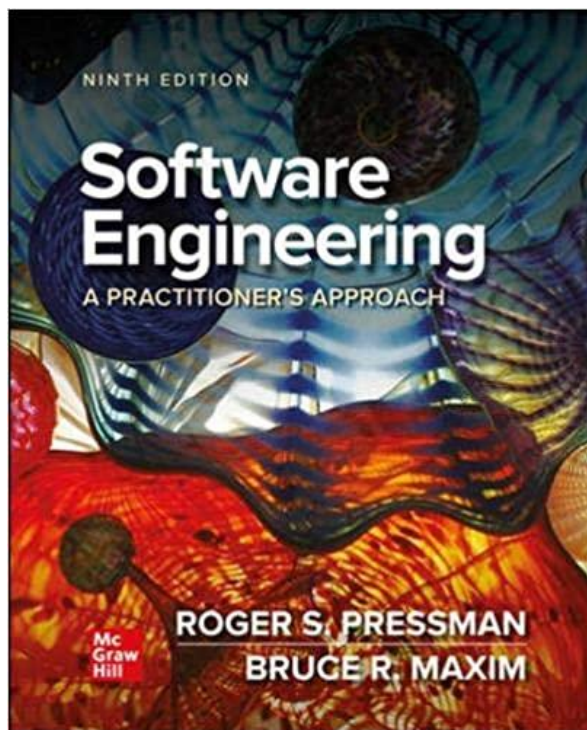


- 欧米では大学生/院生の教科書でもある
- 日本のエンジニアは（実践経験ある）30前後の方が読むとよさそう

書籍「Software Engineering A Practitioner's Approach」について

■ 書籍の変遷

→約5年に1度改版。版改訂で2-3割変化してる。



- 第9版：2019/9/30
 - 第8版：2014/1/23
 - 第7版：2009/1/20
(第6版日本語版：2005/4/2)
 - 第6版：2004/4/2
 - 第5版：2001年
 - . . .
 - 初版：1982年
- ※発売年月日はamazonより



書籍「Software Engineering A Practitioner's Approach」について

■書籍（第9版）構成

→5部/30章構成 + 2章のAppendix

事例1の内容は
第1部の4章

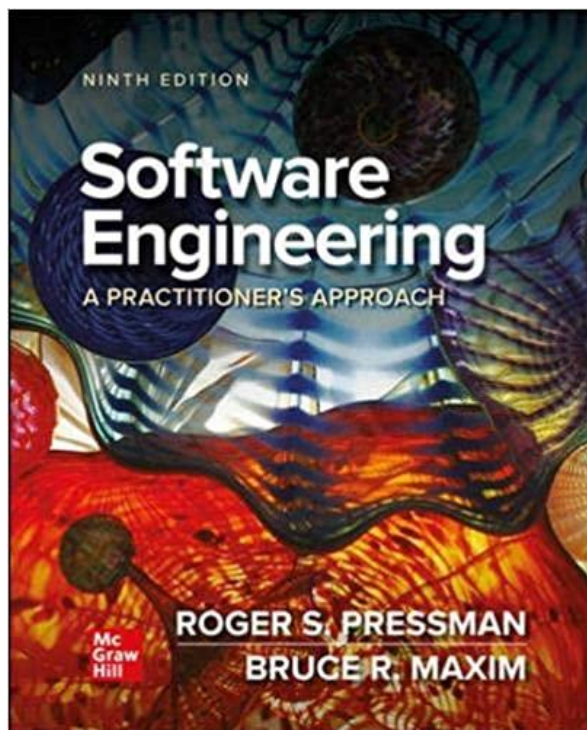
第1部 : The Software Process
ソフトウェアプロセス

第2部 : Modeling
モデリング

第3部 : Quality and Security
品質とセキュリティ

第4部 : Managing Software Projects
ソフトウェアプロジェクトのマネジメント

第5部 : Advanced Topics
先進的な話題



ソフトウェアエンジニアリングの印象（当社調べ、根拠なし）

→ストレートに開発に役立つ情報が無いように見える

<上記理由の想定>

- ・わかりやすい点で「プログラム」や「アルゴリズム」が対象外
- ・プラットフォーム系もほとんど対象外
- ・学習していない場合、基本/応用情報技術者試験を優先に学習を推奨
- ・ひとりで開発するために役立つ情報は比較的少なめ

ただし、基本/応用情報技術者の取得した人などで、
複数名でチーム開発する人、業務として開発をする場合は
業種に応じてつまみ食いすることを推奨（事例で紹介します）

ソフトウェアエンジニアリングの印象（当社調べ、根拠なし）

→「うさんくさい」と感じられる派の方が一定数おられる

<上記理由の想定>

「工学」としてのイメージに対する比較

- ・ 数理的なアプローチが少ない（前述書籍にも確かに少ない）
- ・ 人間系を扱う概念が多く見える、人文、経営、哲学など？
→「理系の学問」っぽく見えない箇所がある、むしろ社会学的

歴史と体系としての安定性が弱い

- ・ 前述したように、5年に1度更新されている（20年で一新？）
→内容がころころ変わっているように見える
学んでも変わってしまうから意味が無いように見える？

ソフトウェアエンジニアリングの印象（当社調べ、根拠なし）

→「うさんくさい」と感じられる派の方が一定数おられる

<上記理由の想定>

「工学」としてのイメージに対する比較

- ・ 数理的なアプローチが少ない（前述書籍にも確かに少ない）
- ・ 人間系を扱う概念が多く見える、人文、経営、哲学など？
→「理系の学問」っぽく見えない箇所がある、むしろ社会学的

参考：

歴史

engineering: the work of an engineer, or the study of this work)

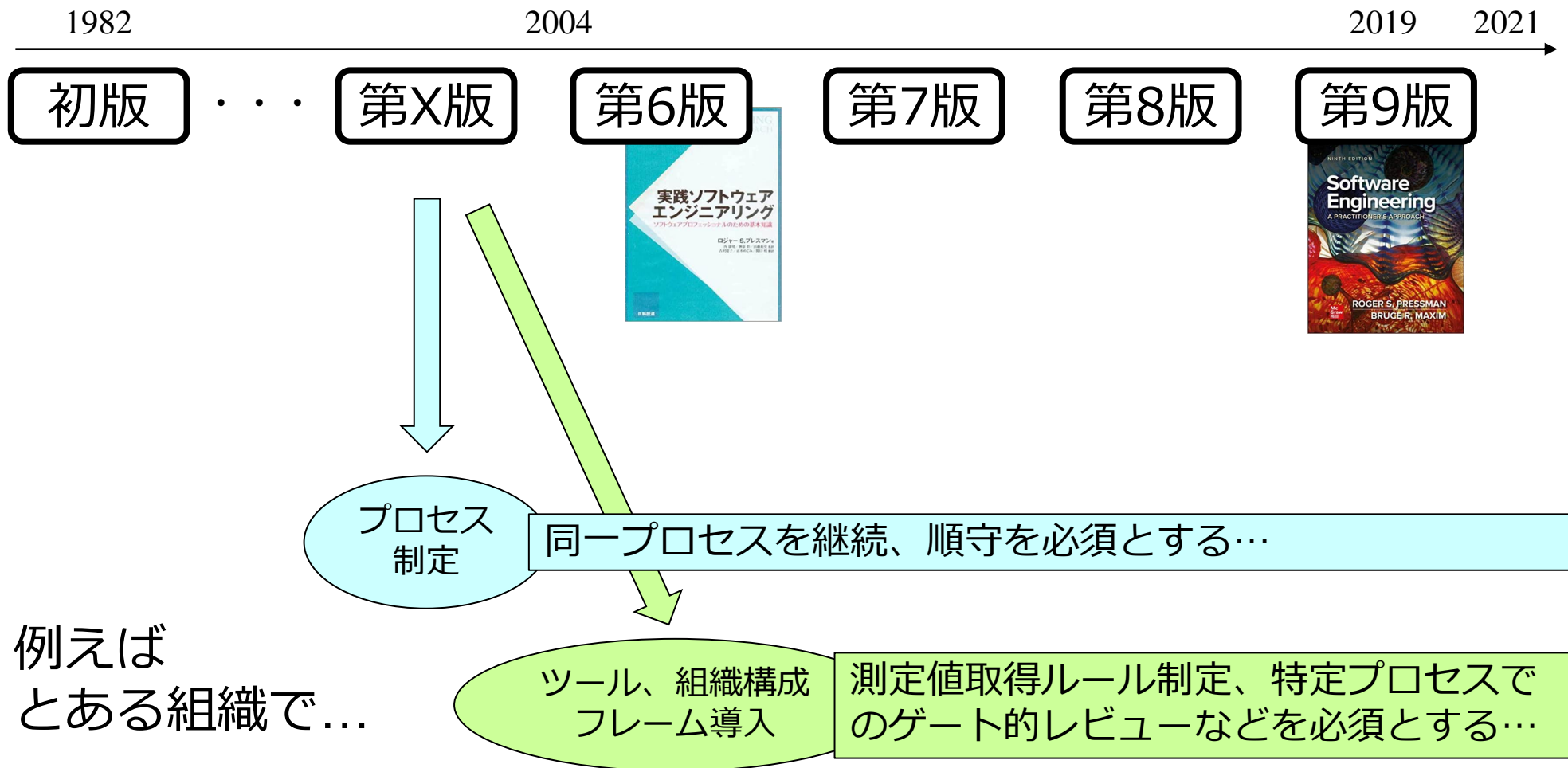
- ・ という意味がある。

一方、工学は study の意味が強い。

上記のため今回は「ソフトウェア工学」ではなく、

「ソフトウェアエンジニアリング」を使っております。

幕間：考えてみよう！



※講演者の所属組織とは関係はないフィクションです

幕間：そして残念ながら。。。

1982

2004

2019

2020

2021

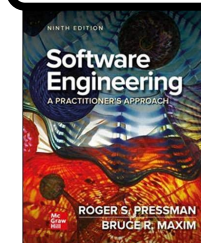
初版

第6版

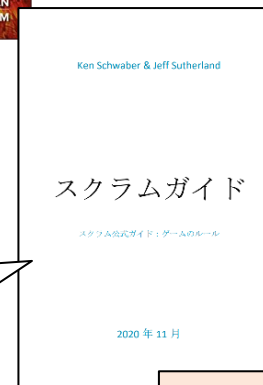
第7版

第8版

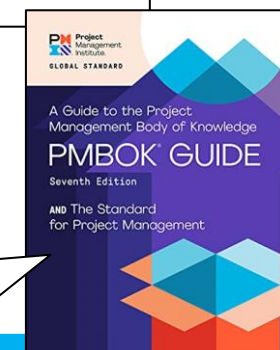
第9版



2020年10月
スクラムガイド最新版
大幅簡易化、ソフトウェア
開発以外にも対応



2021年7月
PMBOK 7th発行
6thまでと大幅に構成と方針
を変更し、アジャイル開発の
概念が大幅に取り入れられる



事例1：（ペーパー）プロトタイピングを活用したプロセス事例

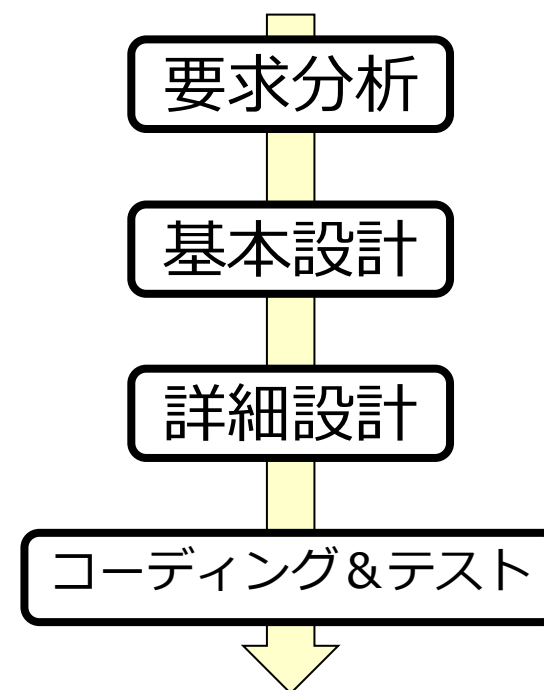
■ 背景

- ・ 完全新規のWebプロダクトを開発し、利用予定の
- ・ アジャイル的な段階的・反復的に開発を進め

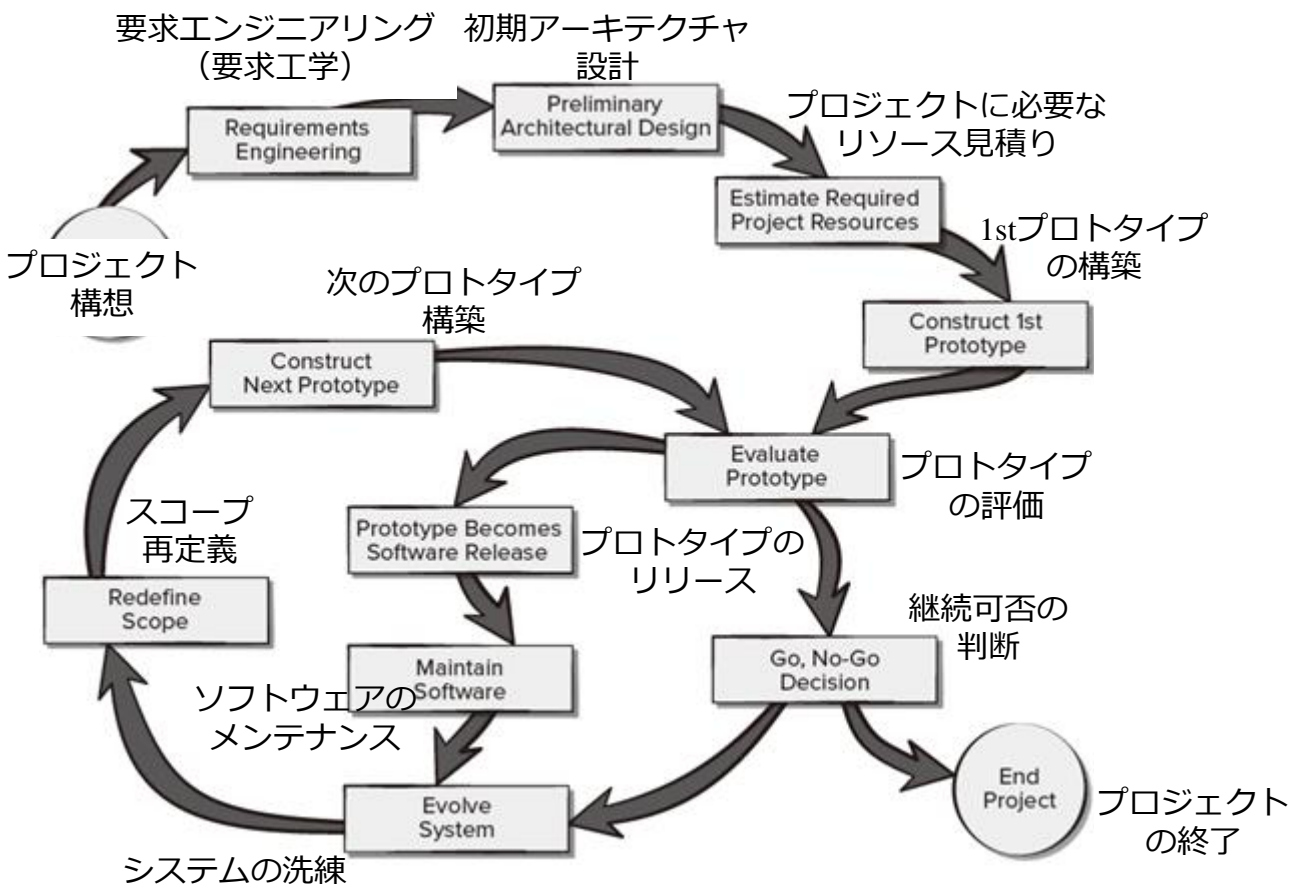
■ 課題

- ・ プロダクトの構築はしたが、本当に必要なものが
- ・ 顧客側から複数の解決策が出てくるが、どれが本
- ・ （アジャイル的なので）仮実装をしてもよいが、

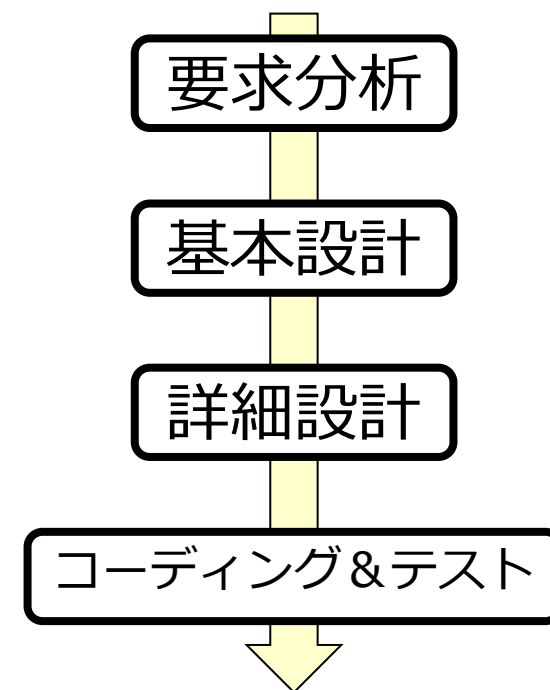
線形逐次型でうまくいく？
（代表：ウォーターフォール）



幕間：考えてみよう！



線形逐次型プロセス
(代表：ウォーターフォール)



Software Engineering A Practitioner's Approach 9th edition 第4章より引用

事例2：IoTシステム構築のPoCへ向けたモデリングとマネジメント事例

■ 背景


- ・事例1で紹介したプロダクトを発展させる必要があり、「連携することによる価値」へIoTシステムとの連携を行うことになった
→まずPoC（概念実証）を実施して効果を実証することから開始となった

■ 課題

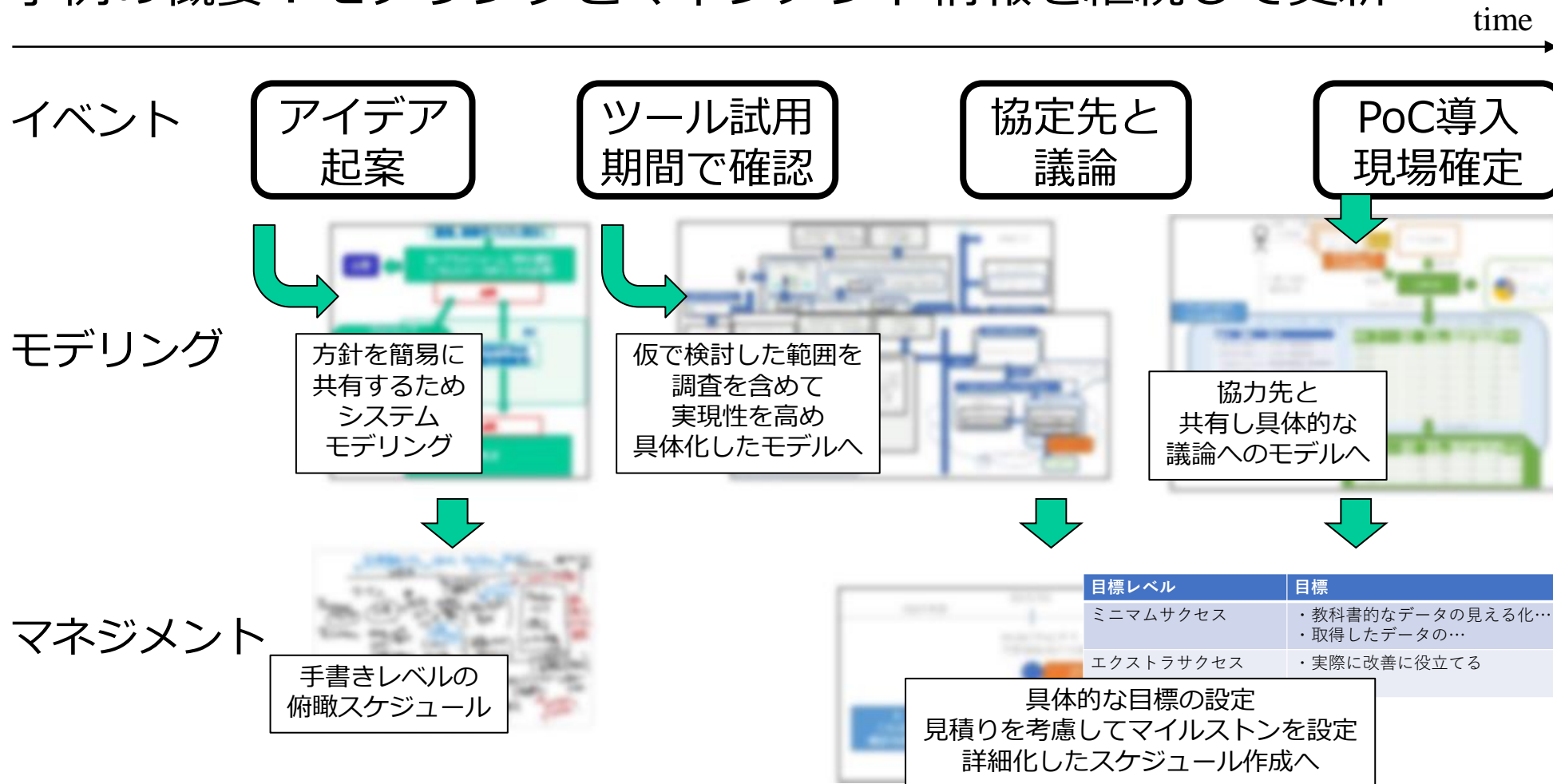
- ・あくまで効果が出そうな見込や一部構想だけがある状況であり、実現方法や具体的に価値を出せるか、どのように進めるかが見えていない
- ・並列して協定相手やパートナー企業も探しており、刻一刻と状況が変わり（年度、月単位など）計画的に進めることが難しい

事例2：IoTシステム構築のPoC事例

こちら、予稿集を提出した時の情報

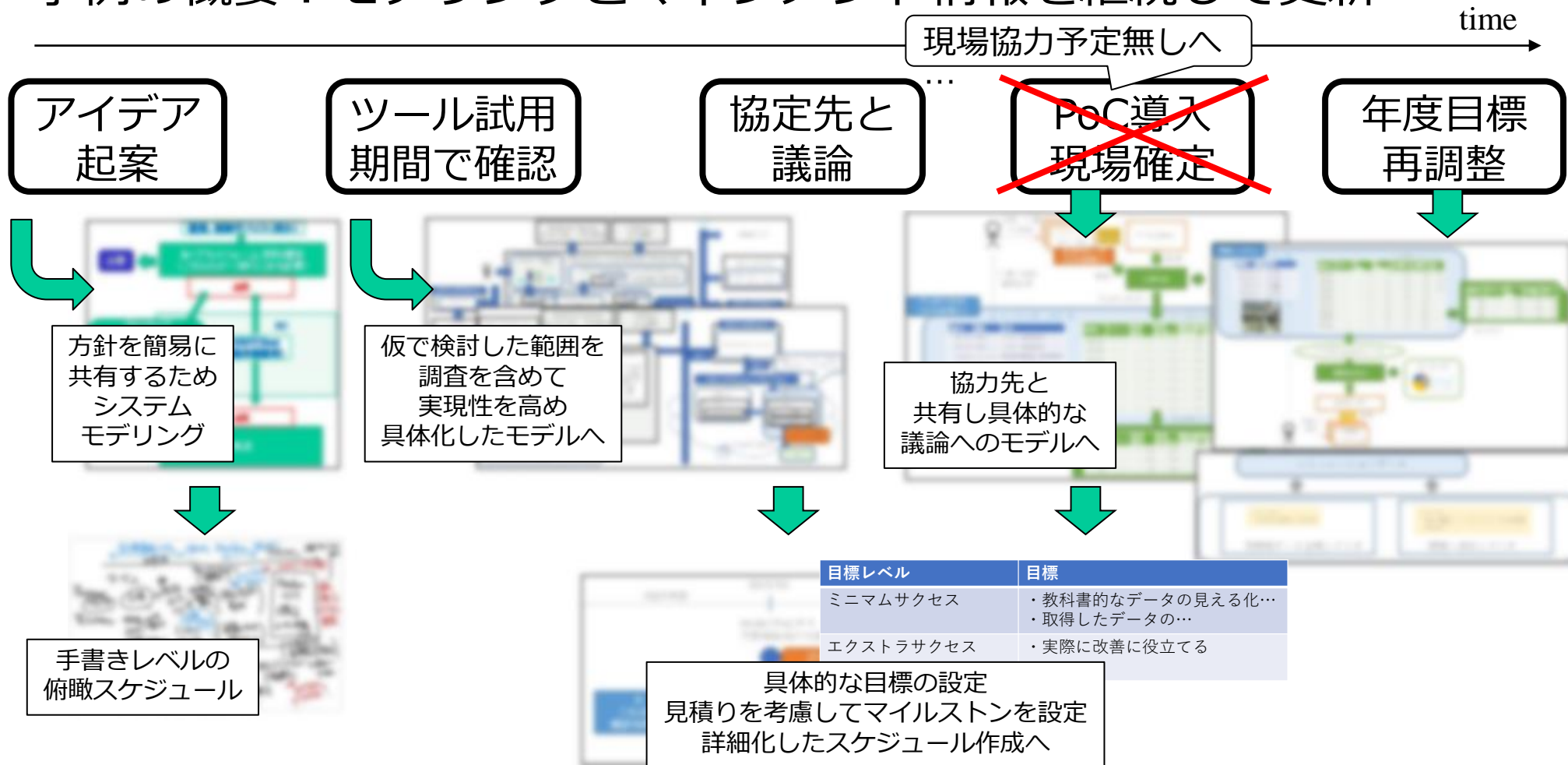


事例の概要：モデリングとマネジメント情報を継続して更新



事例2：IoTシステム構築のPoC事例

事例の概要：モデリングとマネジメント情報を継続して更新



事例2：IoTシステム構築のPoC事例



具体的な流れの紹介は本番のみの特典とします。

事例2：ソフトウェアエンジニアリングの活用

「モデリング」や「プロジェクトマネジメント」の知見を活用

例：

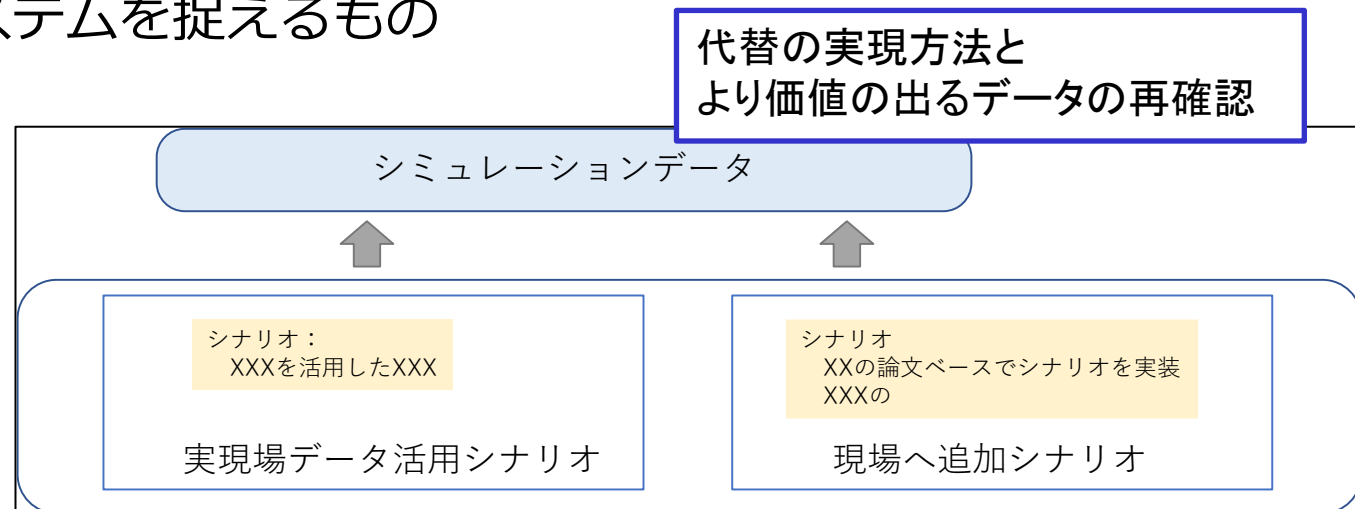
コミュニケーションの原則

第2原則 コミュニケーションの前に準備する

第8原則 何か不明確であれば、図を描く

分析モデルにて「シナリオに基づくモデル」の活用

※利用者の視点でシステムを捉えるもの



「モデリング」や「プロジェクトマネジメント」の知見を活用

例：

- 計画の原則 ※だいたい重要
- 第1原則 プロジェクトのスコープを理解する
 - 第2原則 ステークホルダを計画アクティビティに巻き込む
 - 第3原則 計画は反復的であることを認識する
 - 第4原則 わかっていることから見積りをする
 - 第5原則 リスクを考慮する
 - 第6原則 現実的に考える
 - 第7原則 精度/粒度を調整する
 - 第8原則 どのように品質を保証するか決める
 - 第9原則 どのように変更に対応するか決める
 - 第10原則 頻繁に追跡し、必要に応じて調整する

「モデリング」や「プロジェクトマネジメント」の知見を活用

例：

モデリングの原則

- 第1原則 ソフトウェア開発者の一番の目的はソフトウェアを構築することで、モデルを作成することではない
- 第2原則 不必要なモデルは作らず身軽に進める
- 第3原則 ソフトウェアや、ソフトウェアの解決する問題を、簡潔に説明するモデルを作ることに努める
- 第4原則 変更を取り入れやすい方法で作成する
- 第5原則 モデルを作成した動機を明確に説明できるようにする
- 第6原則 作成したモデルを目の前のシステムに合わせる
- 第7原則 完全なモデルのことは忘れて、役に立つモデルを作る
- 第8原則 モデルの記法を押し付けてはいけない。
情報を伝えることができているなら表現は後回しでよい
- 第9原則 たとえ紙面で見ると正しくても、直感が間違いを訴えるモデルには、注意を向けるだけの理由がある
- 第10原則 できるだけ早くフィードバックを得る

事例2：ソフトウェアエンジニアリングの活用

「モデリング」や「プロジェクトマネジメント」の知見を活用

例：ソフトウェアアーキテクチャ

- ・アーキテクチャとは何か1つの概念を指すものではなく、大小含む数千の決定である
- ・アーキテクチャ上の決定は、システムの構造やステークホルダの満足度への気づきをもたらす

Software Engineering A Practitioner's Approach 9th edition
第10章より引用

SWEST23 (2021)



Architecture Decision Description Template

Each major architectural decision can be documented for later review by stakeholders who want to understand the architecture description that has been proposed. The template presented in this sidebar is an adapted and abbreviated version of a template proposed by Tyree and Ackerman [Tyr05].

Design issue:	Describe the architectural design issues that are to be addressed.
Resolution:	State the approach you've chosen to address the design issue.
Category:	Specify the design category that the issue and resolution address (e.g., data design, content structure, component structure, integration, presentation).
Assumptions:	Indicate any assumptions that helped shape the decision.
Constraints:	Specify any environmental constraints that helped shape the decision (e.g., technology standards, available patterns, project-related issues).

Alternatives:	Briefly describe the architectural design alternatives that were considered and why they were rejected.
Argument:	State why you chose the resolution over other alternatives.
Implications:	Indicate the design consequences of making the decision. How will the resolution affect other architectural design issues? Will the resolution constrain the design in any way?
Related decisions:	What other documented decisions are related to this decision?
Related concerns:	What other requirements are related to this decision?
Work products:	Indicate where this decision will be reflected in the architecture description.
Notes:	Reference any team notes or other documentation that was used to make the decision.

INFO

事例2：考えてみよう！

事例2：IoTシステム構築のPoCへ向けたモデリングとマネジメント事例

■ 背景

- ・事例1で紹介したプロダクトを発展させる必要があり、「連携することによる価値」へIoTシステムとの連携を行うことになった
→まずPoC（概念実証）を実施して効果を実証することから開始となった

■ 課題

- ・あくまで効果が出そうな見込や一部構想だけがある状況であり、実現方法や具体的に価値を出せるか、どのように進めるかが見えていない
- ・パートナー協力状況が変わり、最終的な実現方法も探索が必要な状況

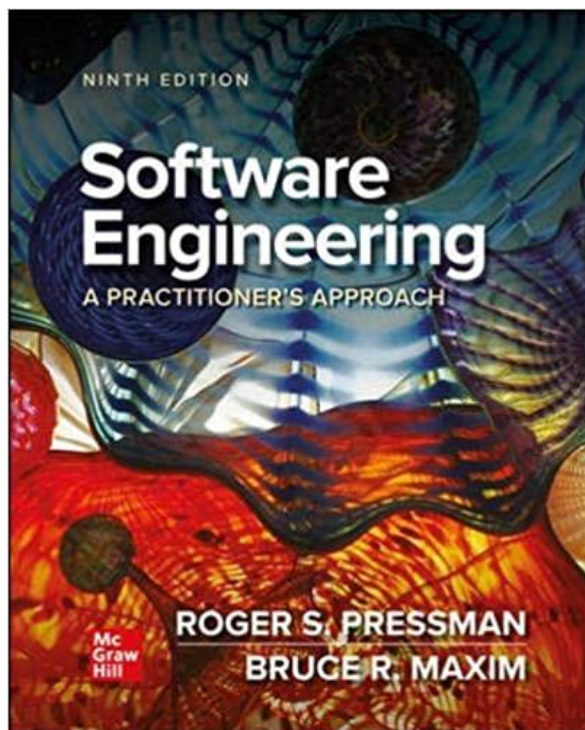
...伝統的なWF的な知見で頑張るとどうなるだろうか？

→いつまでも要件定義/要求分析で停滞、想定で進めて作り直し等になる？

事例2：IoTシステム構築のPoC事例

ソフトウェアエンジニアリングの体系との対応（書籍構成ベース）

■書籍（第9版）構成より



第1部：The Software Process
ソフトウェアプロセス

第2部：Modeling
モデリング

事例2のメイン範囲
は第2部と4部

第3部：Quality and Security
品質とセキュリティ

第4部：Managing Software Projects
ソフトウェアプロジェクトのマネジメント

第5部：Advanced Topics
先進的な話題

事例3：テストの知見を活用した改善事例

■背景 ※ちなみに15年前くらい

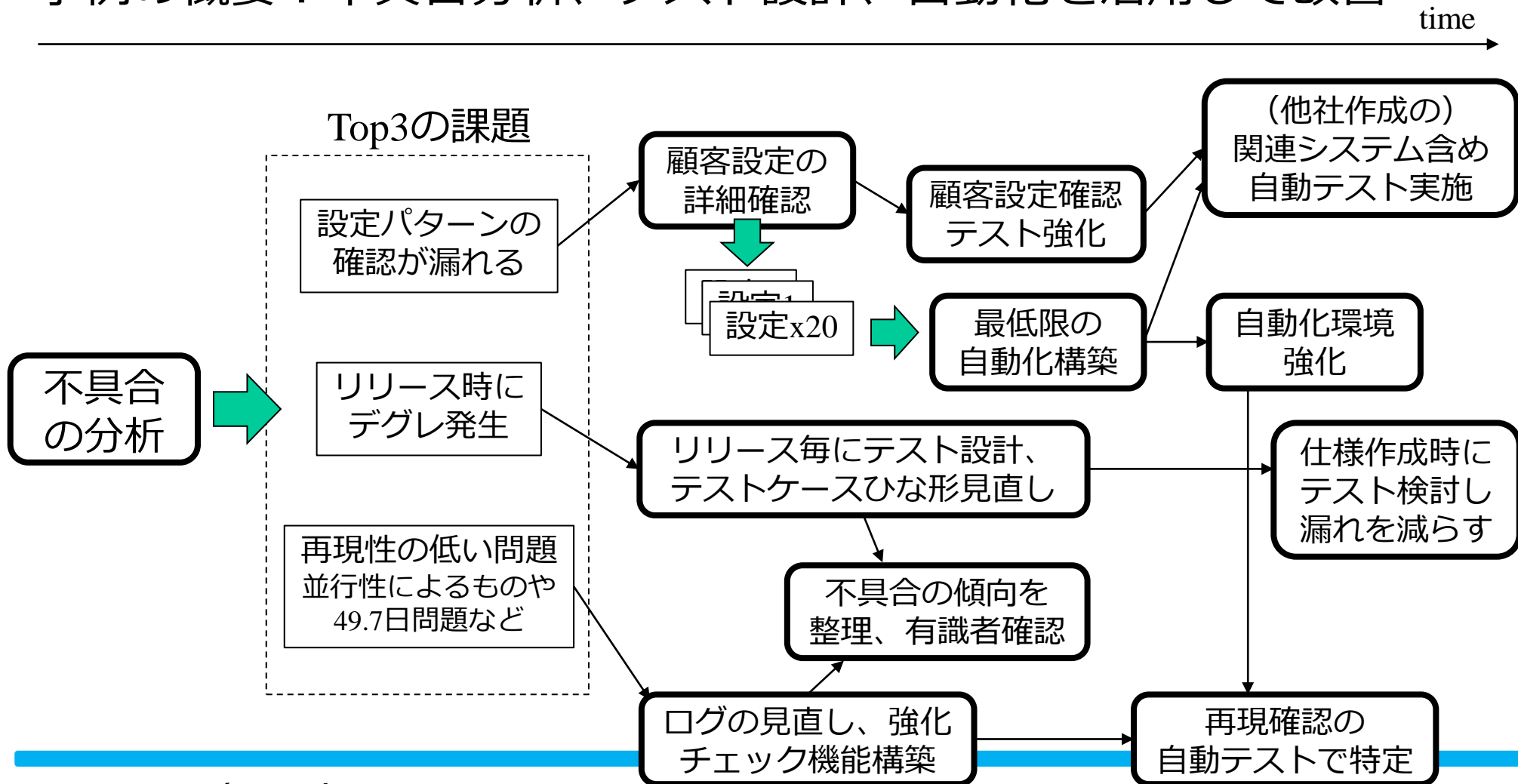
- ・システムをリリースして運用を開始したものの不具合が多発している状況
- ・継続して新しい設定を使うことになるが、新規設定が増えるたびに問題発生
- ・不具合調査、リリース、運用のフォローで負荷が高い状況となる

■課題

- ・不具合が多く、そもそも手を付ける優先順位を決めづらい
- ・設定パターンが多く、全て確認は困難に見えるような状況
- ・再現しないフリーズの問題などが発生もしている

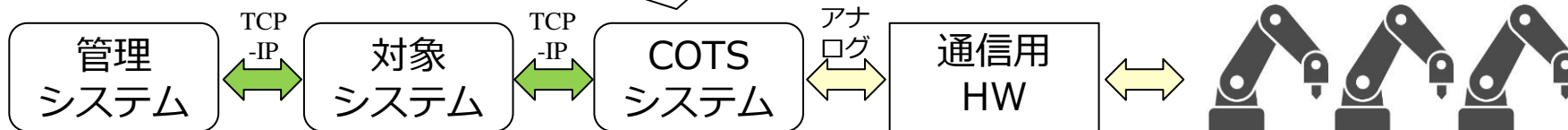
事例3：テストの知見を活用した改善事例

事例の概要：不具合分析、テスト設計、自動化を活用して改善



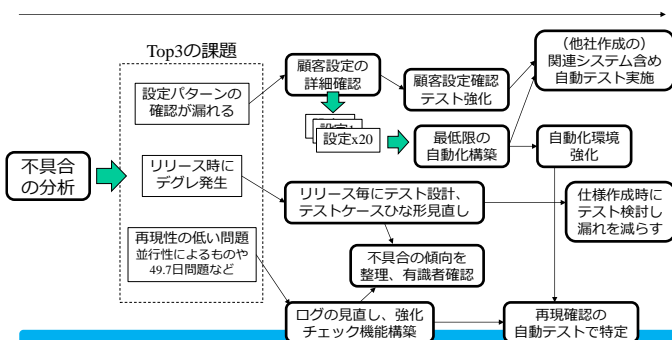
事例3：テストの知見を活用した改善事例

こいつもなかなか
 ひどかった_(:3」△)_

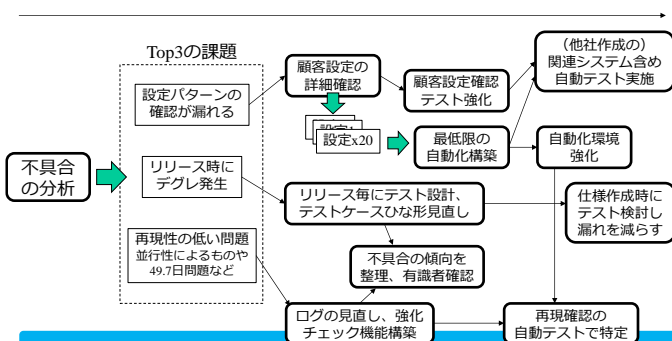
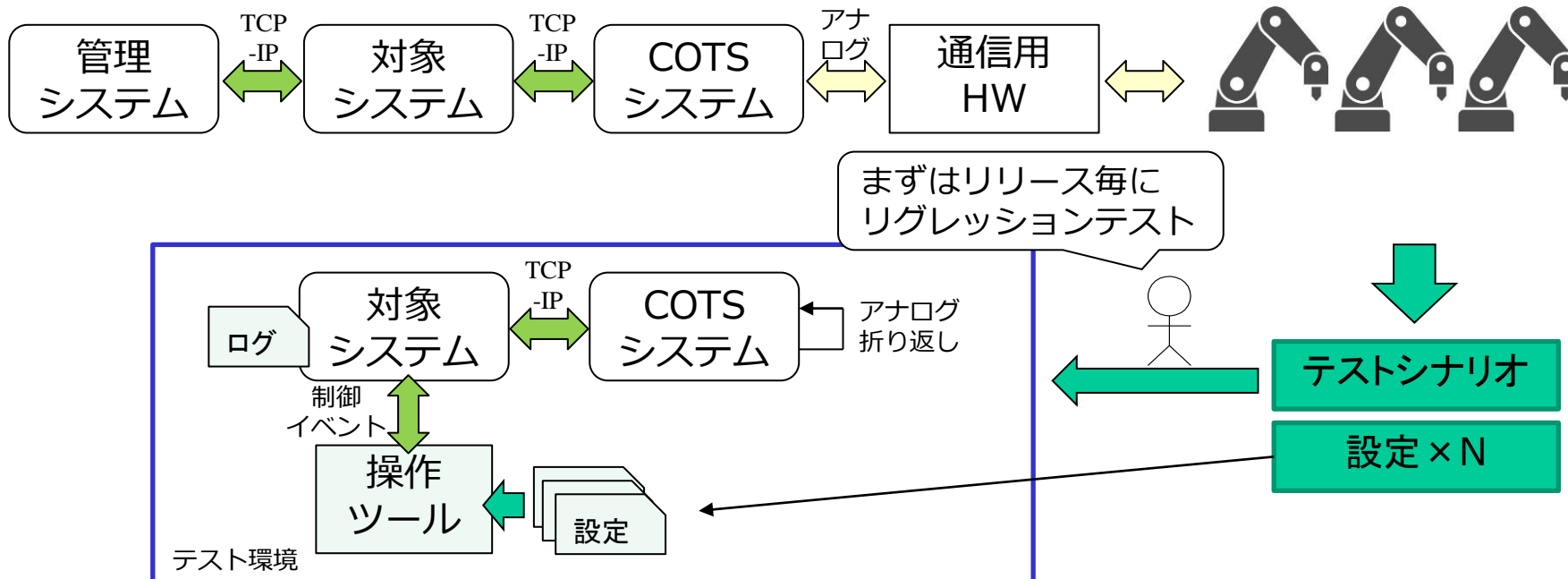


対象システムは
 こんな感じ（超簡略化）
 対象HWはイメージ

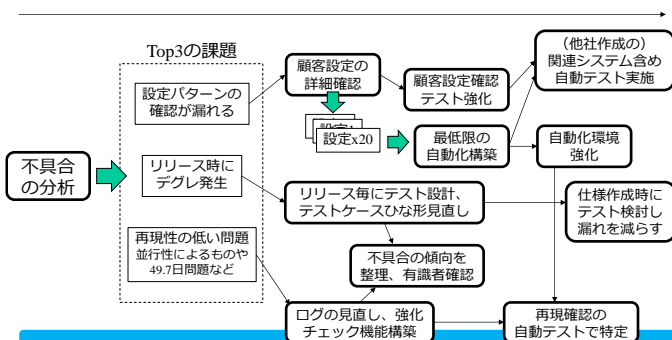
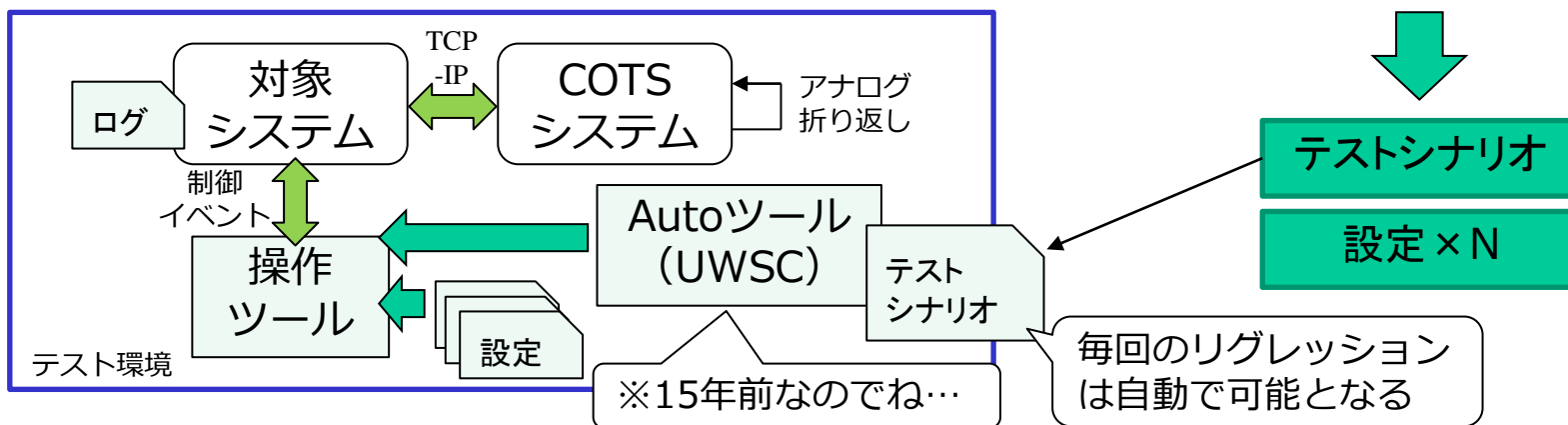
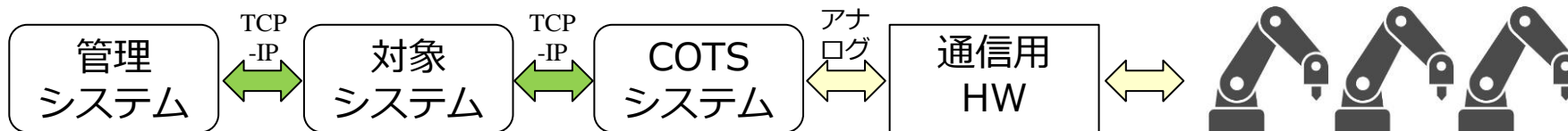
テスト強化対象範囲



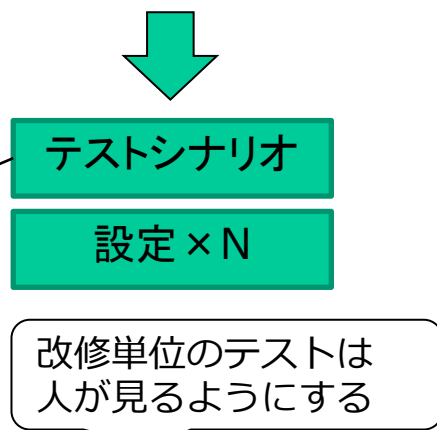
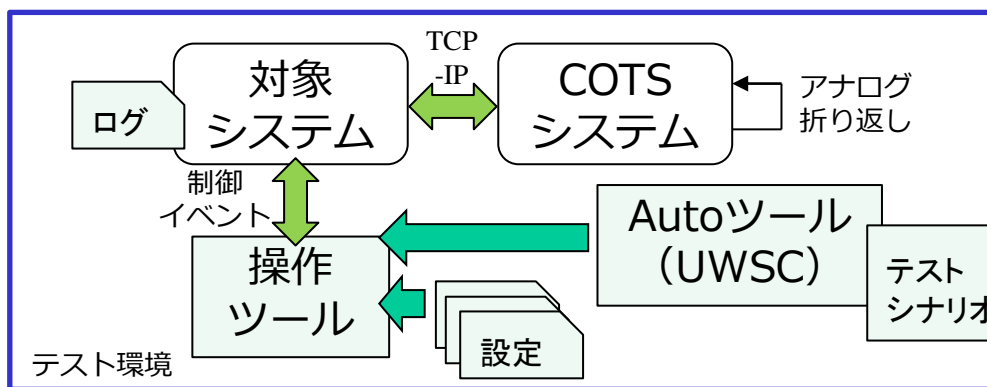
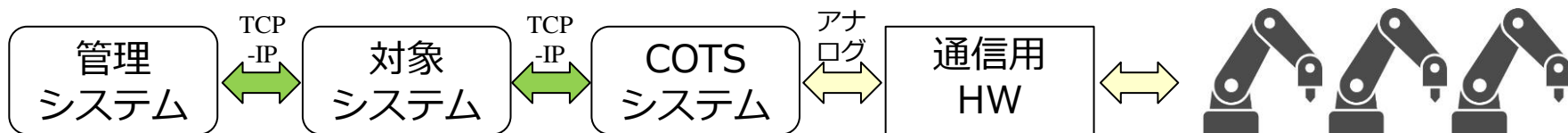
事例3：テストの知見を活用した改善事例



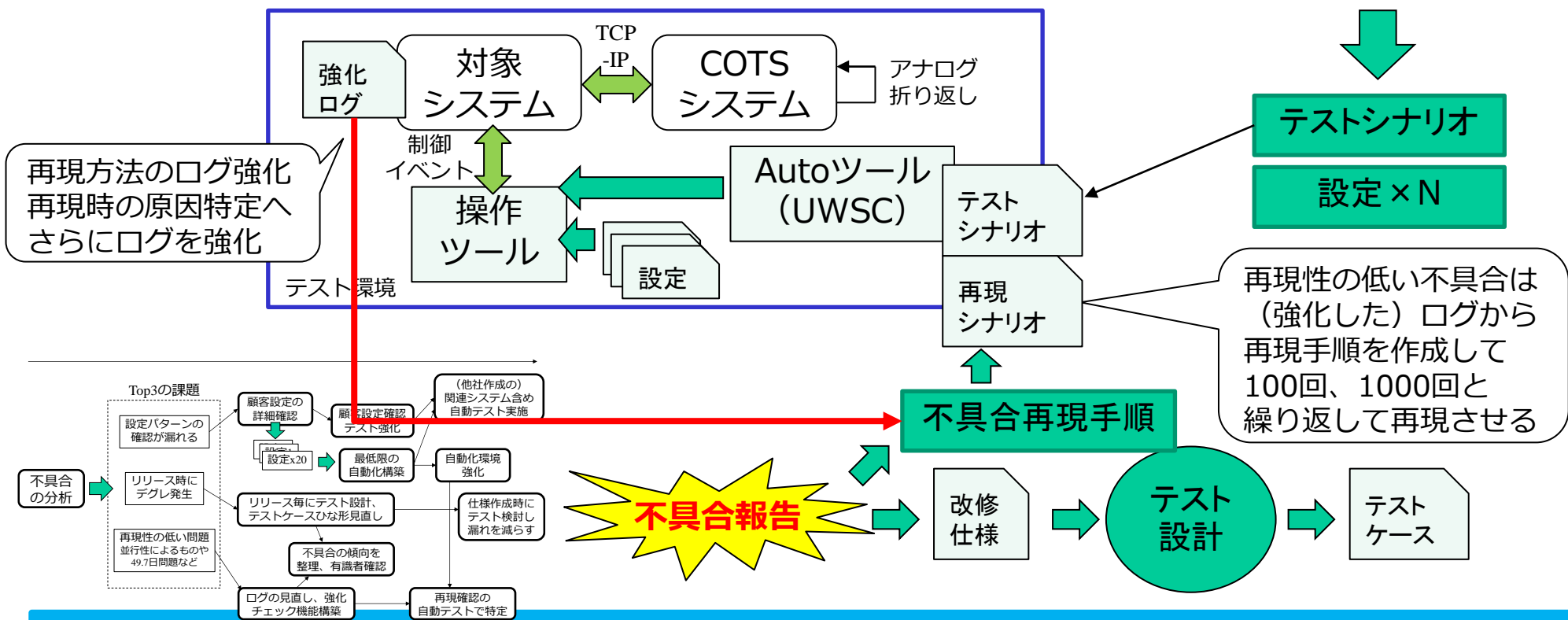
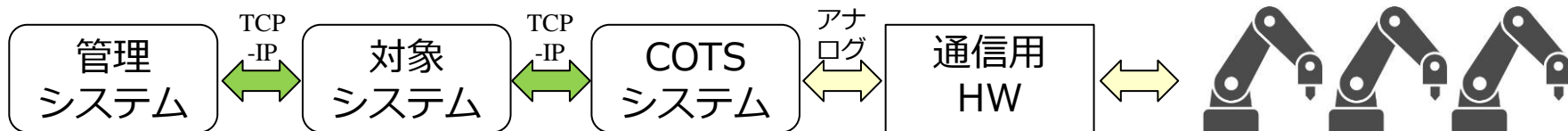
事例3：テストの知見を活用した改善事例



事例3：テストの知見を活用した改善事例



事例3：テストの知見を活用した改善事例



「品質」や「テスト」の知見を活用

例：

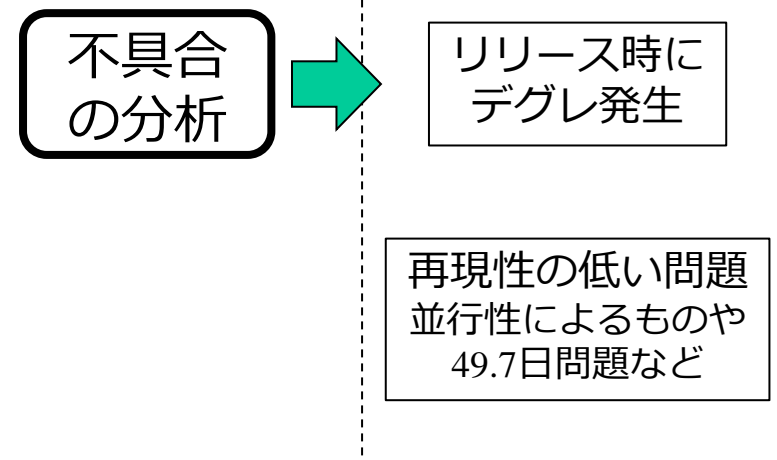
エラー、欠陥の収集と分析

改善の唯一の道は、物事をどう行っているかを測定することである。

エラーや欠陥の情報を収集・分析することで、

エラーがどのように埋め込まれるのか、それをなくすために

ソフトウェアエンジニアリングの活動を適用するのが最善かを把握する。



事例3：ソフトウェアエンジニアリングの活用

「品質」や「テスト」の知見を活用

例：

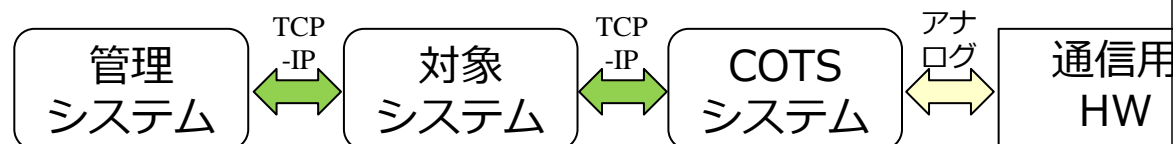
Testing in the wild（実地テスト） ※昨今役立つ知識として紹介

世界中のさまざまなネットワーク環境下で、実際にユーザが利用するデバイスを用い、現実的に起こりうる条件でアプリをテストする

実地テストは、サポート期限の切れたブラウザやプラグイン、固有のハードウェア、不安定なWi-Fiやキャリア回線といった、何が起こるか予測しづらい環境を扱うという特徴がある。

こうした現実の条件を模倣するには、それらの条件から考えられるユーザ層とデバイス環境をテスト担当者が想定している必要がある。

→モバイルアプリでは仕様するスマホやOS、ブラウザ環境を変数として優先順の組合せ想定が必須
多様性をどこまで重視し、どの組み合わせをどれだけテストで扱うか（網羅するか）を決める



フィールド環境において
通信、プラットフォーム、
OS、バージョン、
ブラウザなどが異なる
状況は当たり前...

事例3：ソフトウェアエンジニアリングの活用

「品質」や「テスト」の知見を活用

例：
テスト技法

いまや当たり前となったと思われるので説明省略。

もちろん、Software Engineering A Practitioner's Approach 9th edition でも説明、記載有り。

以下の2冊がとても有効なので、別途紹介しておく。

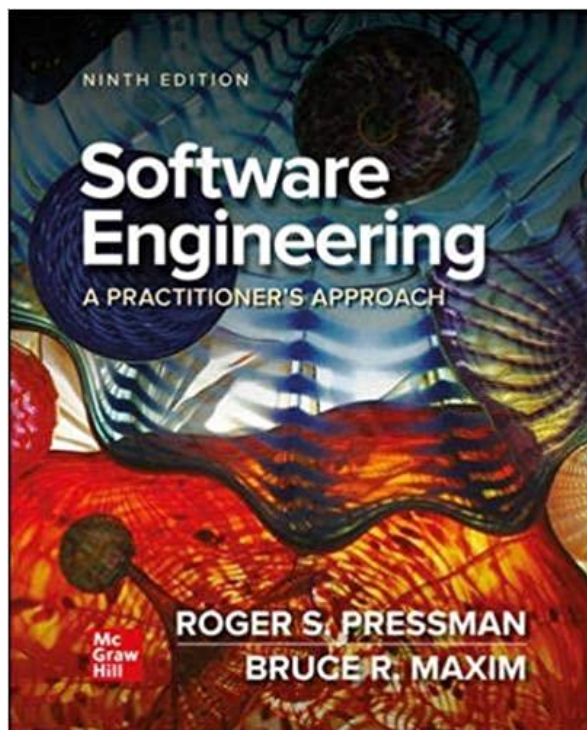
勉強するほど、「設計で必要な技術だよね」という気づきが発生するもの。



事例3：テストの知見を活用した改善事例

ソフトウェアエンジニアリングの体系との対応（書籍構成ベース）

■書籍（第9版）構成より



第1部 : The Software Process
ソフトウェアプロセス

第2部 : Modeling
モデリング

事例3のメイン範囲
は主に第3部

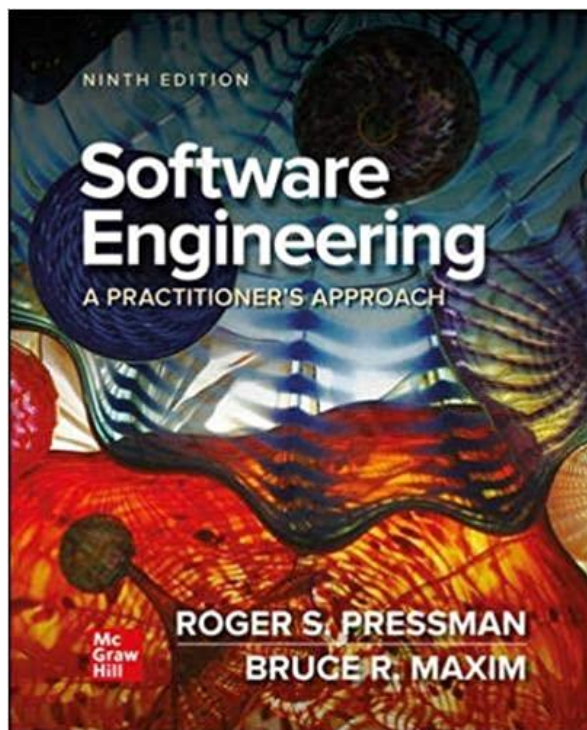
第3部 : Quality and Security
品質とセキュリティ

第4部 : Managing Software Projects
ソフトウェアプロジェクトのマネジメント

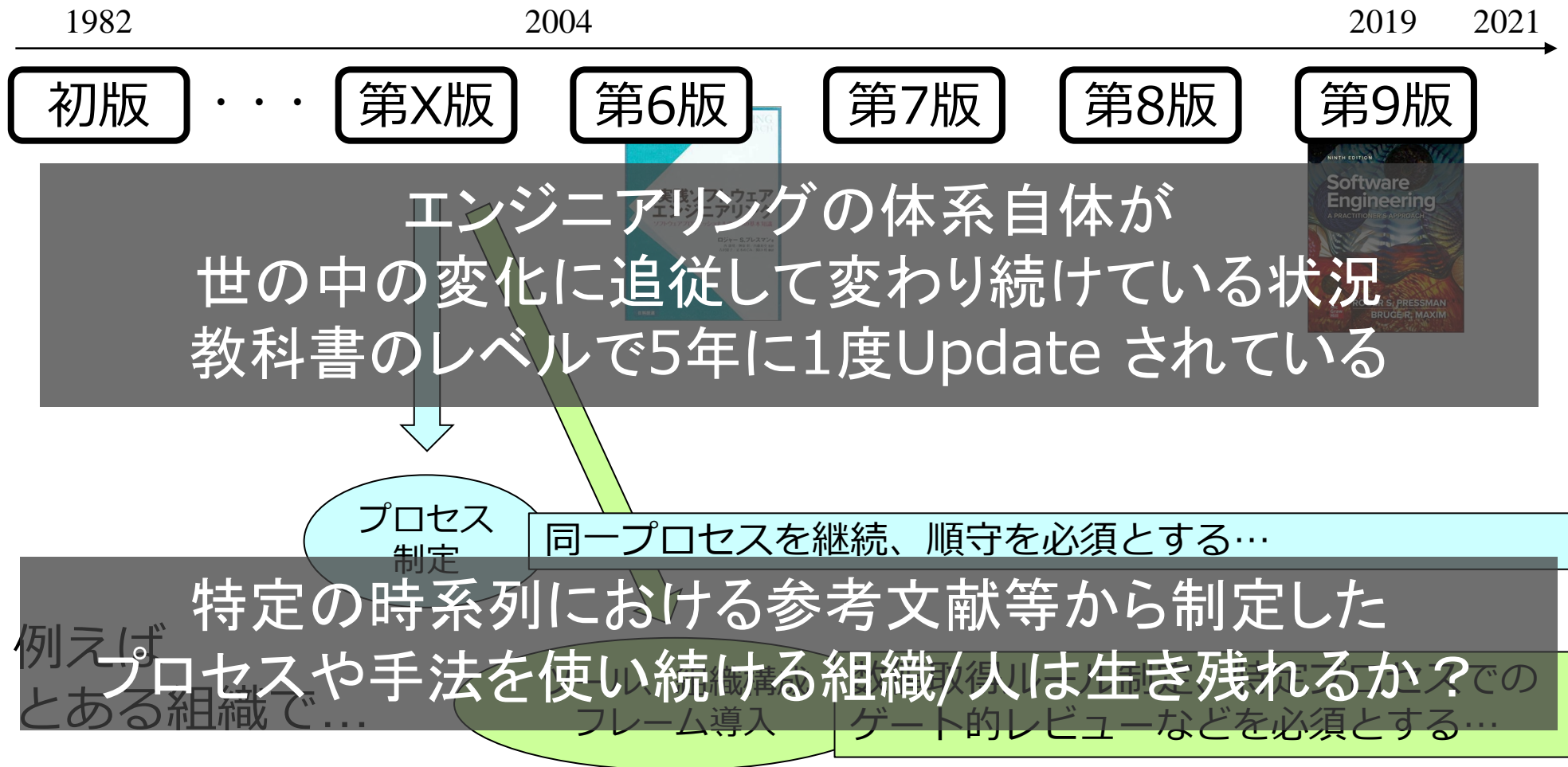
第5部 : Advanced Topics
先進的な話題

さいごに：

- ・ソフトウェアエンジニアリングの付き合い方
- ・ソフトウェアエンジニアリングの活用方法



さいごに：あらためて考えてみよう！



※講演者の所属組織とは関係はないフィクションです

さいごに：そして残念ながら。。。。

1982

2004

2019

2020

2021

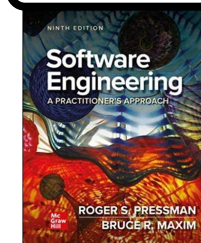
初版

第6版

第7版

第8版

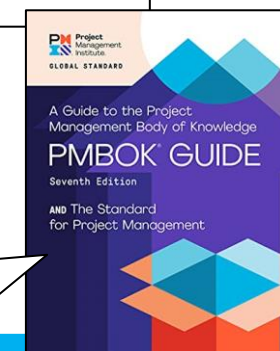
第9版



2020年10月
スクラムガイド最新版
大幅簡易化、ソフトウェア
開発以外にも対応



2021年7月
PMBOK 7th発行
6thまでと大幅に構成と方針
を変更し、アジャイル開発
の概念が取り入れられる



さいごに : そして次の版へ

2004

2019

2020

2021

2024-2025 ? ?

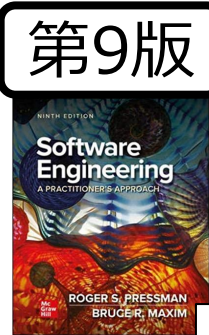
第6版



第7版

第8版

第9版



第10版

Ken Schwaber & Jeff Sutherland

スクラムガイド

スクラムガイド - アドオナーズ

2020年11月

書籍

体系

論文

書籍や体系には過去の書籍や体系をベースとした技術も多数ある

新しい技術
新しい技術
新しい技術
プロセスなど

Software Engineering A Practitioner's Approach 9th editionより

この分野にて学んだことの1つは、ソフトウェアエンジニアリングの実践者は「流行に敏感」ということだ。先へ進む道には、誇大広告にもかかわらず実際にはうまくいかなかったエキサイティングな新テクノロジー（まさに最新の流行）が死屍累々と転がっている。幹線道路に対して方向や幅を変えていくような目立たないテクノロジーによって道は形作られていく。

新しいものも多数発生するが、確実なテクノロジーによって道は作られていくもの。

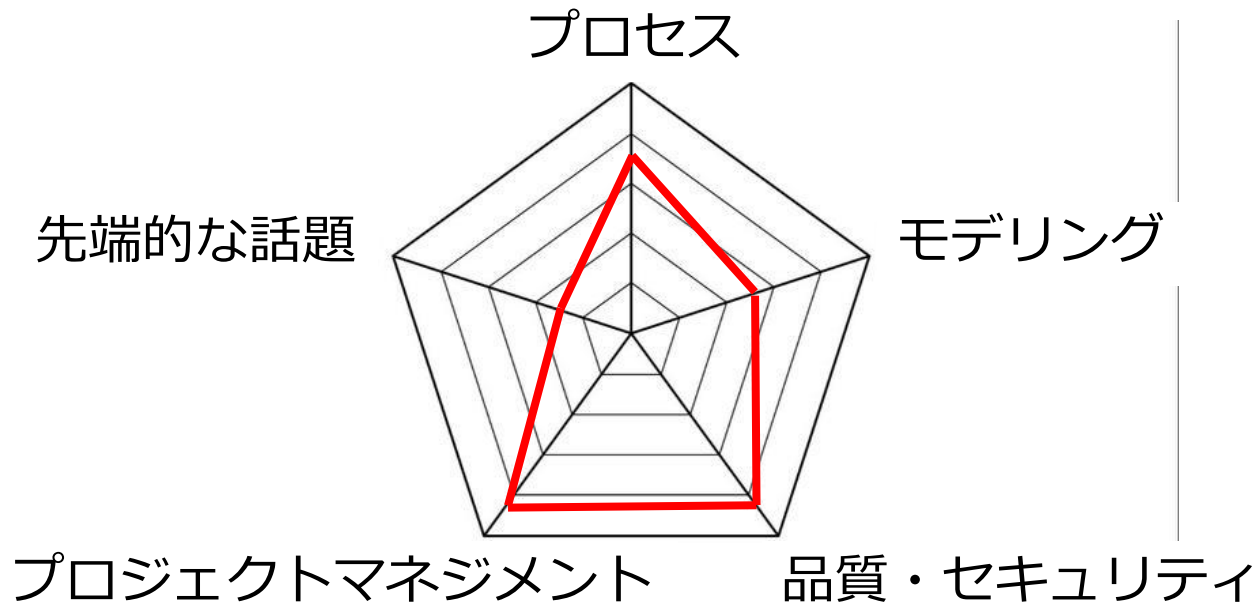
ソフトウェアエンジニアリングは現在も未成熟と書かれている。

そして、しばらく（X十年という単位で...）

ソフトウェア開発は発展し続けると予想される。

ベースラインとして考える、ことも大事。

- ・ 紹介した各部や章の単位で知っている/知っていないを判定。
- ・ 不得意な最低限のレベルを最新の体系をあわせるよう学習。
- ・ 得意分野は当然知っているレベルで、さらに強みを伸ばす。
→場合によっては体系に載せるような技術を紹介していく。



さいごに：体系を超えて新しい知見を！

2019

2020

2021

2024-2025??

第9版



ポイント
体系に追いついていない
レベルの分野は追いつこう

スクラムガイド

2020年11月

書籍や体系には過去の
書籍や体系をベースとした
技術も多数ある

第10版

ポイント
カンファレンス発表や個々の
技術書籍の方が新しい
得意分野はそこから学ぼう

書籍

体系

論文

新しい技術
新しい技術

新しい技術
プロセスなど

将来版

事例

体系

論文

みなさんの
開発事例や書籍
論文などを
将来の体系へ！

新しい技術
新しい技術

新しい技術
プロセスなど

- うまくいっている開発では、ソフトウェアエンジニアリングのエッセンスが何らかに含まれている（その情報を体系化してる）
- 逆にすべて型通りではうまくいかない
 - テラリング（現場との合わせこみ）が必要となる
- 目標の設定等はソフトウェアエンジニアリングが参考となる
- ただし、出来上がるものは少し異なると考えよう
- 現場では段階的かつ、自然とできる部分から試してみよう
例)
 - 現状のプロセスを少しだけ変える
 - 問題を実感できる部分を変える

現場で取り入れるには...

- **得た知識に対して、自分たちで考えて小さく試してみよう**
組織の手続きや技術も古いことがあるので疑ってみる
※「否定」するのではなく、より状況に適したやり方を考える
失敗も価値がある、なぜ失敗したかも考える癖を付けよう
- **小さく試して、役立つ自信がいたらちょっと広げる提案へ**
 - 個人で役立ったことをふりかえりでチームで共有
 - チームで効果が出たものを組織標準の改善案として稟議

- 陥りやすい事項（自分もよく陥った次第） ...
 - ・ **特定の成功した手法が万能でどこでも使えると思いこんでしまう**
自分一人ならよいですが、他人を巻き込むときには特に注意！
うまくいった場合も条件が適合しただけの可能性を考えよう
- 処方上の注意：
 - ・ **銀の弾丸はないということは頭の片隅に置いておこう**
 - （広く使えるものはあれど）特定の手法は万能ではない
 - ・ **自分の手で試そう（仮説検証ができるとさらに効果的になる）**
 - 事前に自分の手で試して説明できること！強制は特に危険！
 - ・ **各手法の「原理」「得意なこと」「苦手なこと」を理解しよう**
 - 特にチームに持ち込む技術は適切かつ十分に理解しておこう
(危険物：生産性メトリクス、ユースケース記述、シーケンス図、WBS、2因子網羅テスト...)

「Software Engineering A Practitioner's Approach」第1章より...

近代的なコンピュータが存在する前に書かれた古典的な著書である「How to solve it」のなかでGeoge Polyaは問題解決の本質について述べている。これはソフトウェアエンジニアリングプラクティスの本質ともいえる。

1. 問題を理解する
2. 解決策を計画する
3. 計画を実行に移す
4. 結果が正しいことを確認する

成功した手法が万能でどこでも使えると思いこんでしまわずに、問題を理解して、各手法の「原則」「得意なこと」「苦手なこと」を適切に把握したうえで解決策を選択、計画しよう！

- Software Engineering A Practitioner's Approach 9th edition
- デザイナーのためのプロトタイピング入門
(主に事例1)
- ソフトウェアテスト技法ドリルーテスト設計の考え方と実際
(主に事例3)
- そしてこちら

<https://www.ohmsha.co.jp/book/9784274227943/>

