

ソフトウェアエンジニアリングの全体 とIoT時代のモデリングおよび関連す る品質

わしざき ひろのり
鷺崎 弘宜

早稲田大学

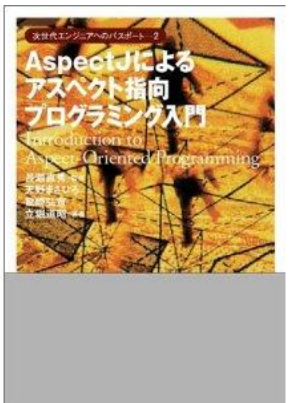
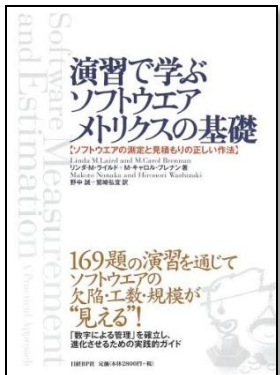
グローバルソフトウェアエンジニアリング研究所

Twitter: @Hiro_Washi washizaki@waseda.jp

<http://www.washi.cs.waseda.ac.jp/>



- 早稲田大学グローバルソフトウェアエンジニアリング研究所所長・教授
- 国立情報学研究所客員教授
- 株式会社 システム情報 社外取締役
- ISO/IEC/JTC1/SC7/WG20 Convenor
- SEMAT Japan Chapter Chair
- IEEE CS Japan Chapter Secretary
- IEEE ICST 2017 PC Co-Chair



世界に何を学び、日本のどの強みを伸ばすべきか

ソフトの総合的な品質評価が必要

データ+経験則パターン大事

ビジネス、システムも含めた捉え方

日本は優れた技術を持ちながら世界のスピードに追い付いていない

世界のソフトウェア開発の潮流はアジャイル開発

ソフト開発の人材づくりが重要

IoT時代のセキュリティ、プライバシーは重要な研究課題

最大グローバルソフトウェアエンジニアリング研

グローバル時代をひらく



プログラミン教育に役立つゲームソフトの学習効果検証は研究テーマのひとつ。タに基づき国際比較で明らかにするのが同研究所のモットーだ。企業などから20人近い研究員を招き、中国やインドネシア、マレーシアなど多様な国の学生も参加。プログラミンク言語やソフト設計・再利用、品質保証、開発マネジメント、工学教育を研究し、15年にはソフトの総合的な品質評価の枠組みを確立するための調査研究を始めた。

未来の開発者育成探る

の考え方が大きく異なる。世界に何を学び、日本のどの強みを伸ばすべきかを考える。現在のテーマの柱の一つが、ソフト開発人材の基盤をつくる。みるのが、子どもが自ら学ぶ、プログラミング教育。政府が推進する教育用ゲームソフトだ。

約30人の小学生が模擬授業で、セキュリティやプライバシーも研究対象に加えていく。考えた。(桜井佑介)

ビジョン 鷲崎弘宜所長

ソフト開発を評価するには「データ」だけでなく、開発者へのインタビューで明らかになる経験則も重視する必要がある。開発現場には「こうするとうまくいく」「こうなるといけない」といったパターンがある。これらに名前を付けて明確にすれば、日々の開発・運用の効率を高めることにつながる。



データ+経験則 社会を変えたい

だ。社会をよりよくするためにソフトはあるのだから、研究対象はビジネスや社会システムに広がって当然。現在は研究所の名前に「ソフトウェア」としか入っていないが、社会システムを含めた「システムズ・アンド・ソフトウェア」と改称してもいいかもしれない。ソフトを取り巻く生態系の全てを明らかにしていきたい。社会を変える気概を持っている。

も20年度に小学校でプログラム学習効果を確認するため、関西などでも開き、315年まで延べ1,000人規模のデータを集積。教材として選択する際の参考にしよう。強いフジテレビキッズ(向・)学習効果のより高いソフト開発に力も注げる狙い。あらゆるものがインターネットにつながる「IoT」の6種類のソフトをそれぞれ時代になり、鷲崎所長は「セキュリティやプライバシー」も研究対象に加えていく。考えた。(桜井佑介)

世界を席巻する大きな日本のソフトウェアを生み出すには何が重要なのか。世界的な基準から製品をその開発現場に人材を確保し、国際競争力を高めよう。早稲田大学が2010年に開設したのがグローバルソフトウェアエンジニアリング研究所だ。「日本は優れた技術を持ちながら世界のスピードに追いついていない」。早稲田大学術院の教授で同研究所所長の鷲崎弘宜氏は日本のソフトウェアの課題をこう指摘する。現在は「アジャイル開発」と呼ぶ手法。顧客を巻き込んで短時間でソフトの発表と検証を繰り返す。後で直すことを前提に修正しやすい仕組みを取り入れる。最初から完全を目指し、時間をかけて精緻化する日本の手法とは品質

目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- モデリングと品質
- IoT時代のモデリング: 参照アーキテクチャと接続
- IoT時代のモデリング: 測定・学習さらには育成
- まとめ

正統なエンジニアリング

職業人が果たす判断，
行為，助言が，社会で
実質的な**価値**を形成

コミュニティで妥当と確
認される**知識**が**科学的
基盤**に基づく

知識・適格性の妥当
性を**コミュニティ**で
判定できる環境

知識の島々から体系へ

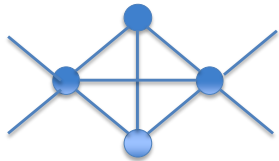
知識

プロフェッショナリズム

体系上のパ
ターンや手法



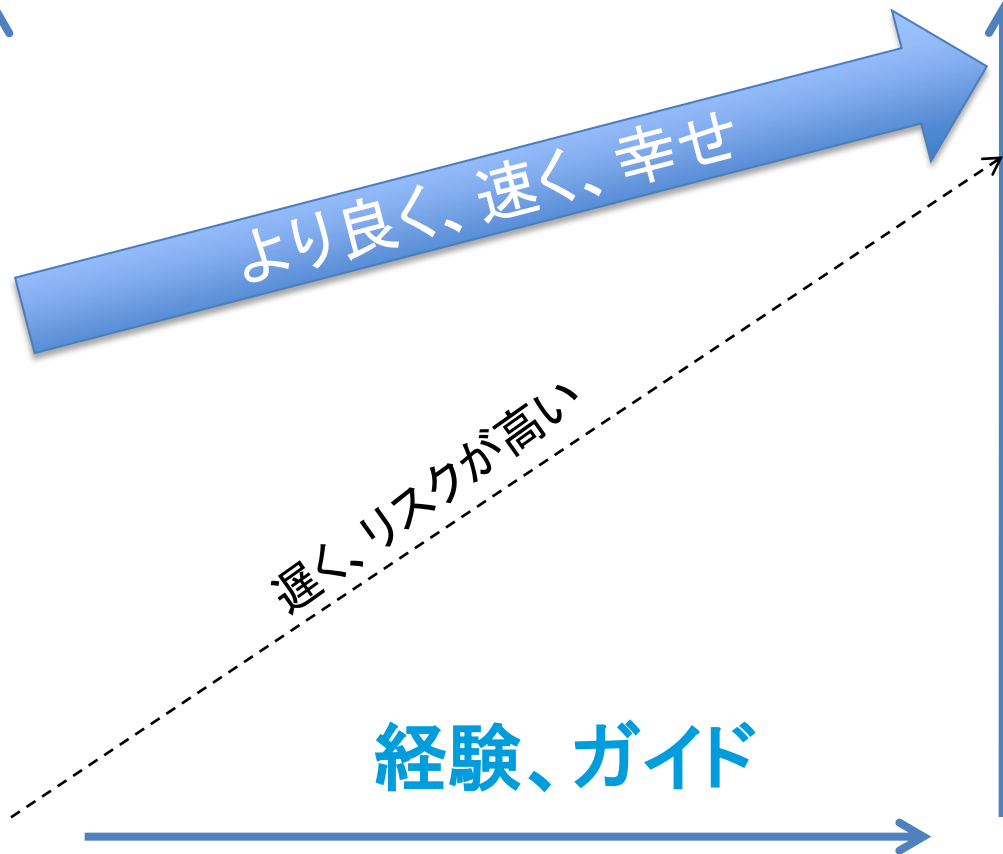
知識の体系



知識の島々



カバレッジ UML
アジャイル
Java, C#, Python



ベストプラクティ
スに裏打ちさ
れた能力

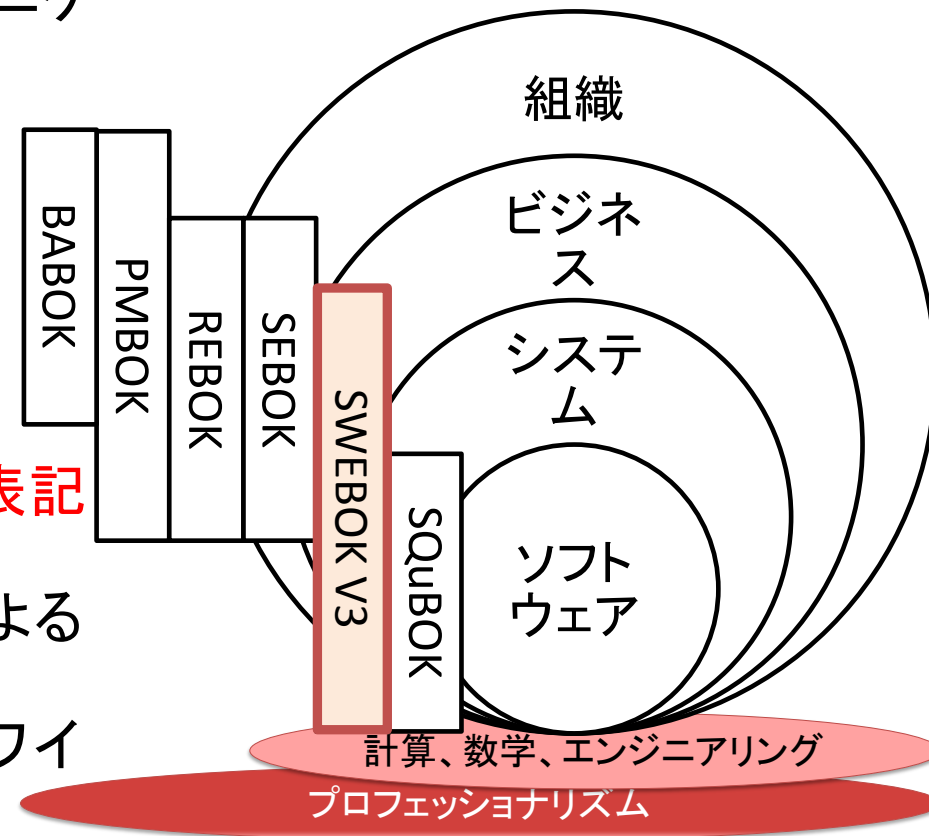
Guide to the Software Engineering Body of Knowledge (SWEBOK)

v3 2014 <http://www.swebok.org/>

要求 要求の基礎 要求プロセス 要求抽出 要求分析 要求仕様 要求妥当性確認 実践上の考慮事項	設計 設計の基礎 設計の主要問題 構造とアーキテクチャ ユーザインタフェース設計 設計品質の分析評価 設計の表記 設計戦略・手法 設計ツール	構築 構築の基礎 構築の管理 実践上の考慮事項 構築技法 構築ツール	テストイング テストの基礎 テストレベル テスト技法 テスト関連計量尺度 テストプロセス テストツール	保守 保守の基礎 保守の主要問題 保守プロセス 保守技法 保守ツール
構成管理 SCMプロセスのマネジメント 構成の識別 構成制御 構成状態記録・報告 構成監査 リリース管理・配布 構成管理ツール	マネジメント 開始と範囲定義 プロジェクト計画 プロジェクト実施 レビュー・評価 終結 計量 マネジメントツール	プロセス プロセス定義 ライフサイクル プロセス査定・ 改善 計量 プロセスツール	モデル・手法 モデリング モデル種別 モデルの分析 開発手法	品質 品質の基礎 品質マネジメントプロセス 実践上の考慮事項 品質ツール
プロフェッショナル 実践 プロフェッショナリズム 集団ダイナミクス・心理 コミュニケーションスキル	経済 経済の基礎 ライフサイクル経済 リスク・不確実性 経済分析手法 実践上の考慮事項	計算基礎	数学基礎	エンジニアリング 基礎

V3改訂に見るソフトウェアエンジニアリング進化

- プロフェッショナリズムの重要性
 - 倫理、法令、技術者認定、コミュニケーション
- **ソフトウェアの外への拡大**
 - ビジネス、経済
 - 計算、数学、エンジニアリング
- 深化や定着
 - 各種ツール・自動化
 - **モデリング: モデリングの原則、表記、種別、分析**
 - 構築技法: テンプレート、契約による設計、MDA
 - テスト技法: モデルベース、ペアワイズ、テスト駆動
- **新たな品質特性**
 - ユーザビリティ: UI設計、インタラクションテスト
 - セーフティ: Airborne Systems and Equipment Certification

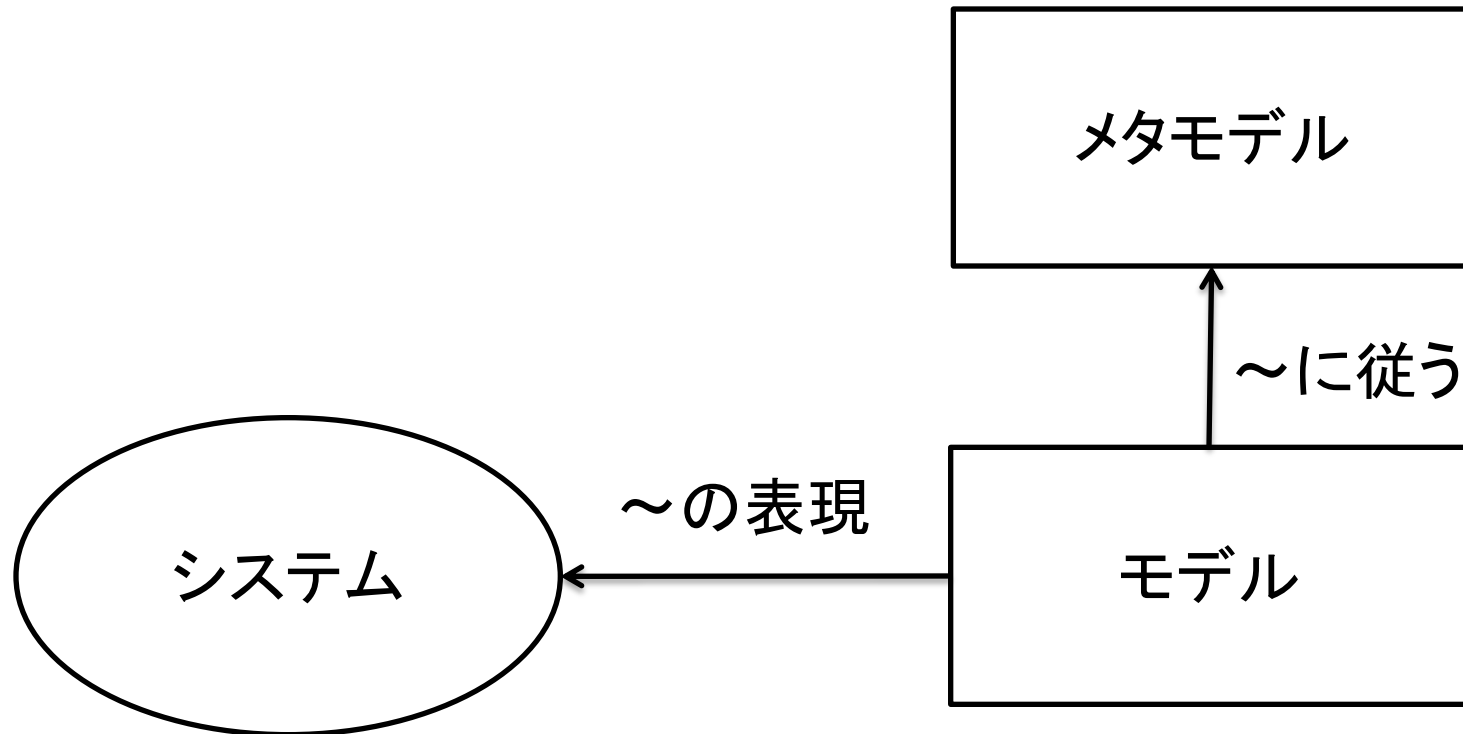


目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- モデリングと品質
- IoT時代のモデリング: 参照アーキテクチャと接続
- IoT時代のモデリング: 測定・学習さらには育成
- まとめ

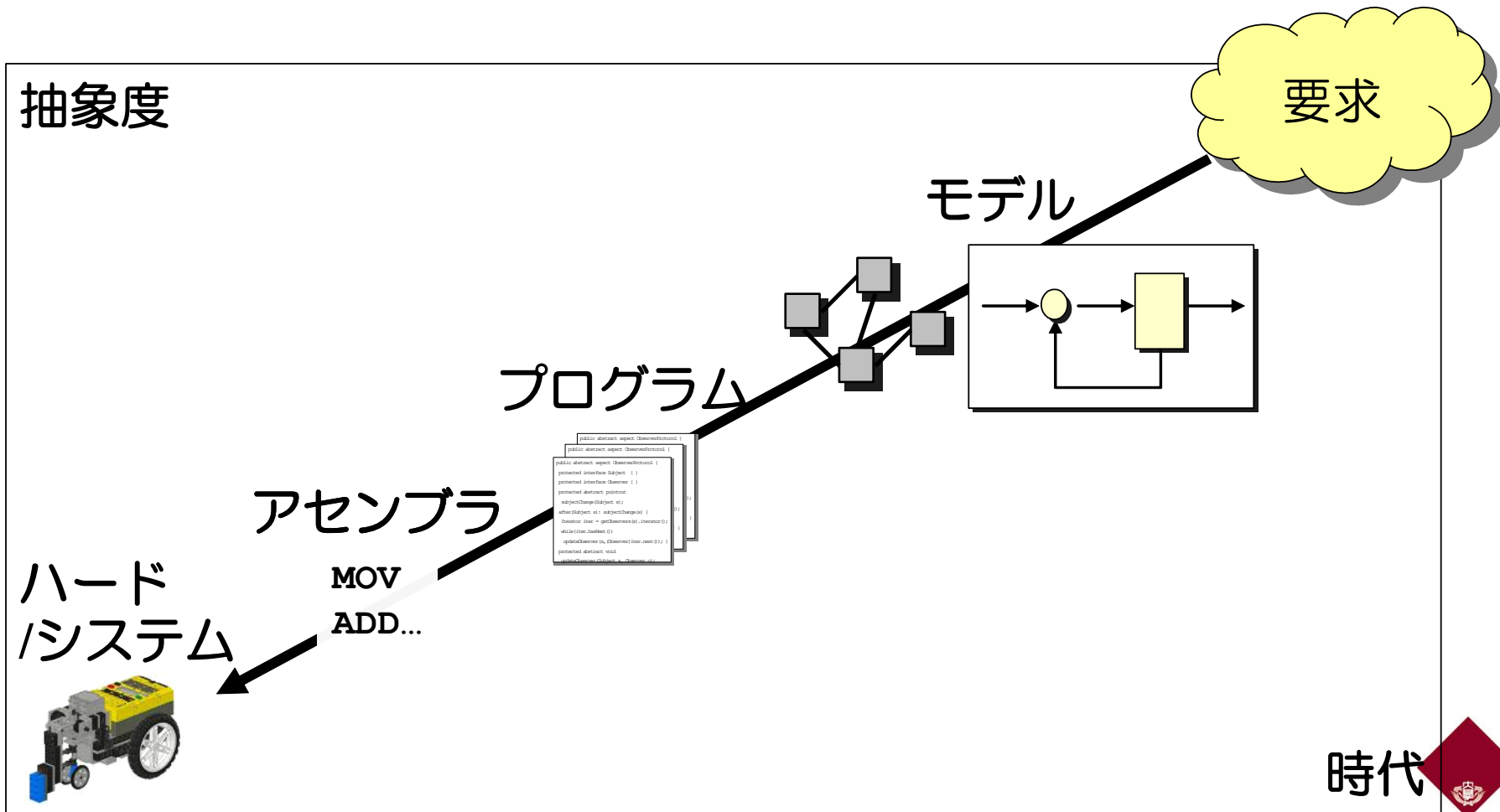
モデルとは

「現実世界の何か」に対する「認知上の目的」
のためのコスト上効率的な「代用物」



プログラミングからモデリングへ

- 「どう動かすのか」から「何をしたいのか」へ
- 品質の早期の作り込みと検証
- コミュニケーション



モデリングの成熟度

Lv0: モデルなし

- 口頭伝達、その場しのぎの修正

Lv1: 文書のみ

- あいまい、主観がそのまま

Lv2: 文書 + 図

- 理解しやすい

Lv3: 図 + 文書

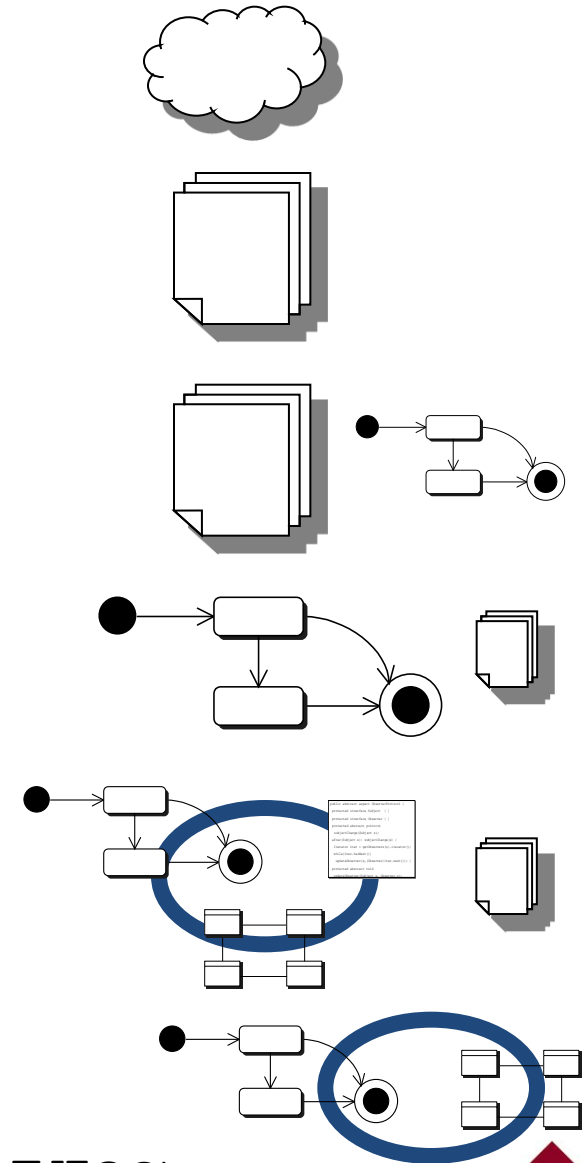
- 主観が客観に

Lv4: 精密な図 + 文書

- コードとの自動接続

Lv5: 図のみ

- モデル駆動開発が目指す未来



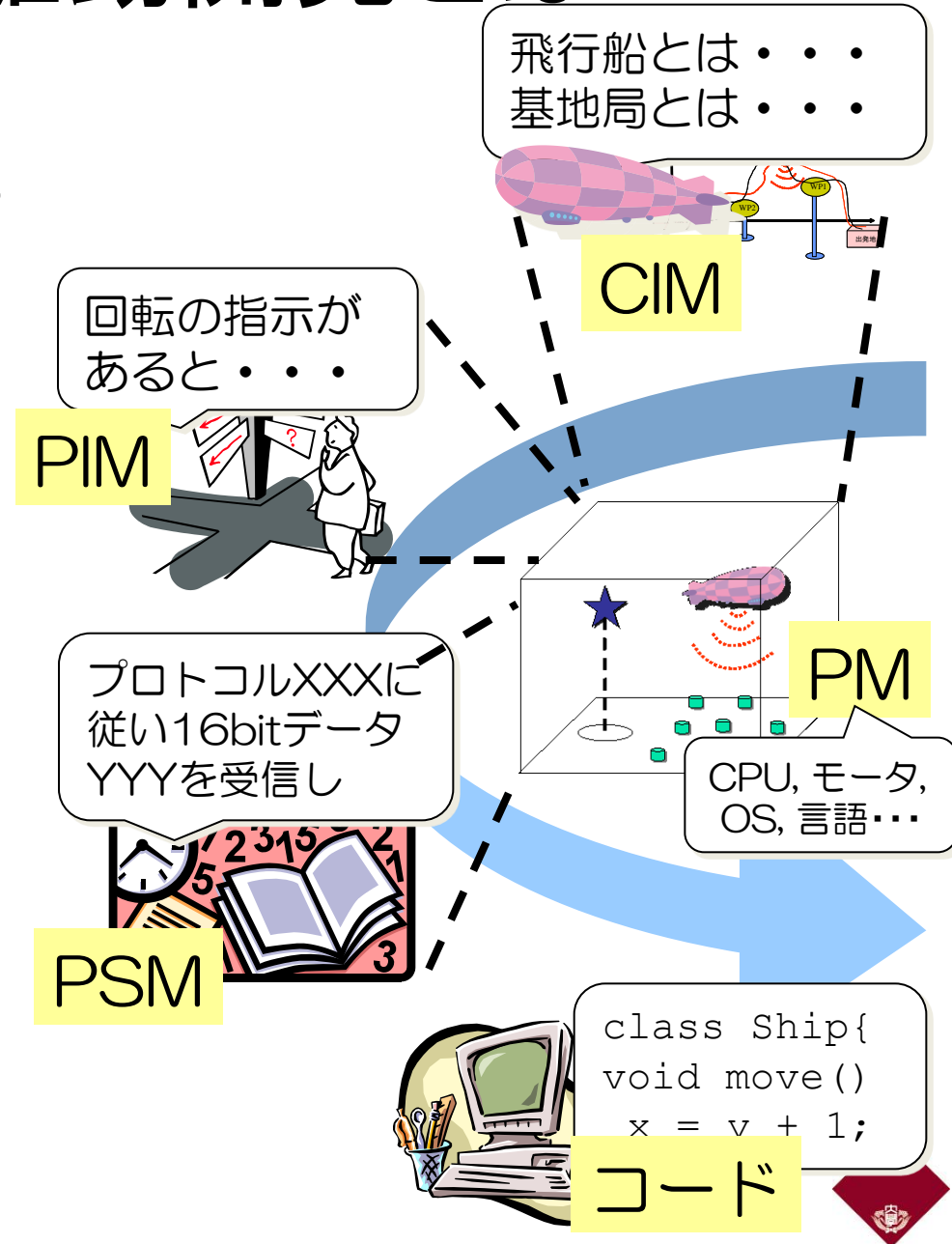
モデル駆動開発とは

モデル変換

- モデルに付加情報を加えて、異なる視点のモデルやコードを得る
- 視点の設定、モデルの記述・操作の重要性
- 関心事の分離
 - Computation Independent
 - Platform Independent
 - Platform Specific

モデル駆動開発 (MDD)

- 分析・設計情報をモデルとして表現
- モデル変換の繰り返し
- 可搬性、再利用性、相互運用性向上



ソフトウェア・システムの様々なモデル

- システム全体
 - SysML, ADL
 - 要求分析、設計、シミュレーション
- アプリケーション
 - UMLなど
 - 設計や検証、コード生成
- 制御
 - Simulinkなど
 - 設計やシミュレーション、コード生成
- 電気・機械
 - CADなど
 - 設計や解析

周辺領域との関係

- ドメイン特化言語 (Domain-Specific Language; DSL)
 - 特定の問題領域(ドメイン)用の開発言語
 - MDDにおけるモデリング言語として
- ドメイン駆動設計 (Domain-Driven-Design; DDD)
 - ドメインモデル中心の考え方
 - MDDがパターン・実現方法の一つ
- プロダクトラインエンジニアリング (Product-Line Engineering; PLE)
 - 共通資産を再利用した組織的・戦略的なプロダクト系列開発・管理
 - MDDが実現方法の一つ

目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- **モデリングと品質**
- IoT時代のモデリング: 参照アーキテクチャと接続
- IoT時代のモデリング: 測定・学習さらには育成
- まとめ



SWEBOKに見るモデリングと品質

- モデル・手法
 - モデリング、モデル種別、モデルの分析、開発手法
- エンジニアリング基礎
 - 統計分析、モデリング・シミュレーション・プロトタイピング・
..
- 要求
 - 要求の基礎、要求プロセス、要求抽出、要求分析、要求仕様、要求妥当性確認、実践上の考慮事項
- 設計
 - 設計の基礎、設計の主要問題、構造とアーキテクチャ、ユーザインタフェース設計、設計品質の分析評価、設計の表記、設計戦略・手法、設計ツール
- 品質
 - 品質の基礎(ソフトウェアセーフティ)
- 計算基礎
 - セキュア開発・保守(要求セキュリティ、設計セキュリティ・
..)

リダイレクトシステム開発の例 [繁在家08]

• 要求

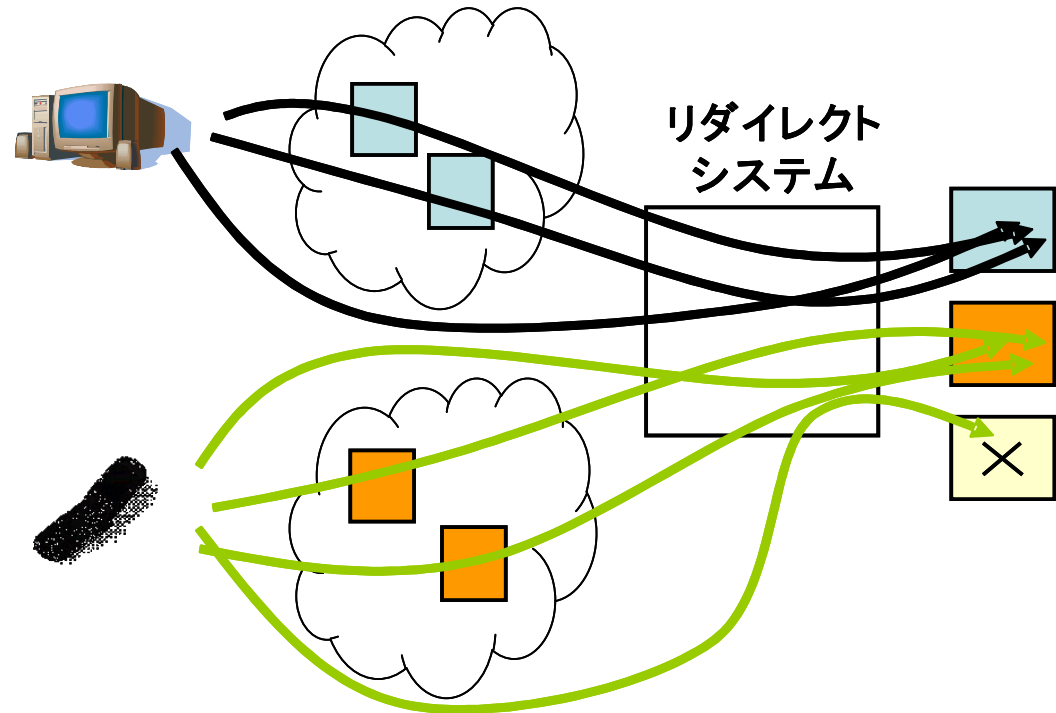
品質要求を明確化できるか？

- シナリオ
- ゴール指向分析

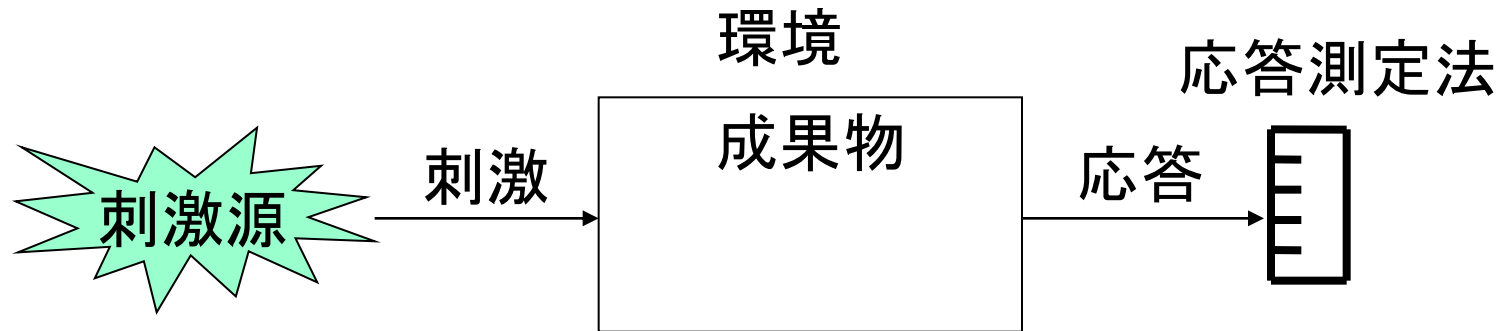
• 設計

品質を着実に組み入れられるか？

- パターン
- 品質駆動設計・評価



要求: 品質シナリオ



A7

(環境) 通常稼働時に
(刺激) リダイレクト情報に変更があった場合 (応答) 運用を止めずに反映できる。

A1

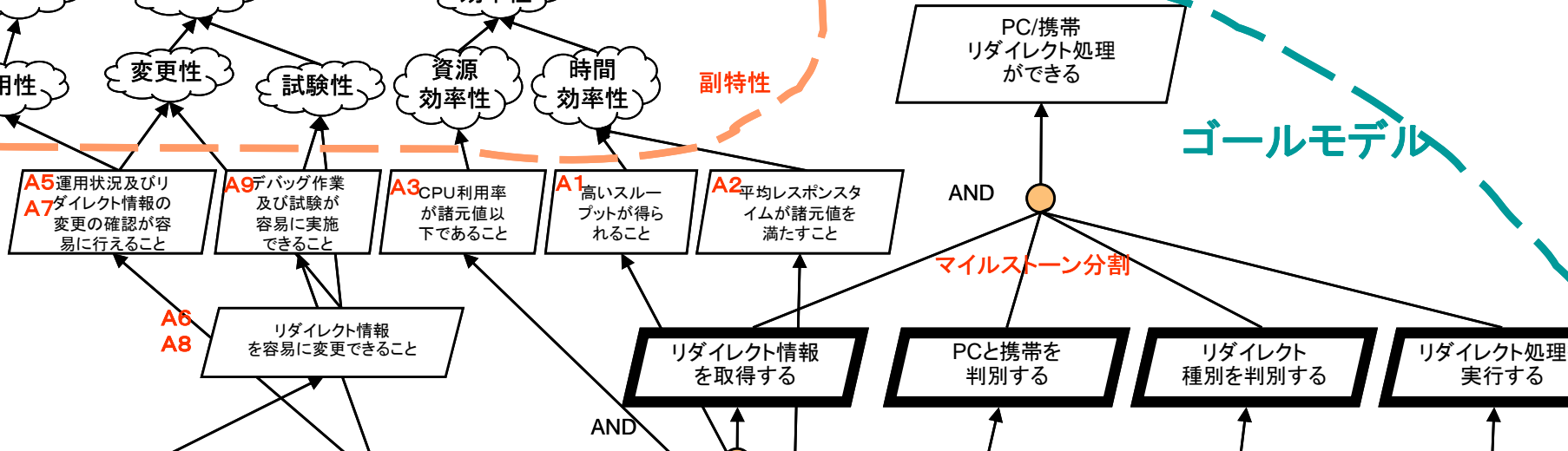
(環境) 通常稼働時に
(刺激) JMeterで高負荷をかけた場合
(応答) HTTPリクエストを秒間1000トランザクション処理できる。

要求: ゴール指向分析

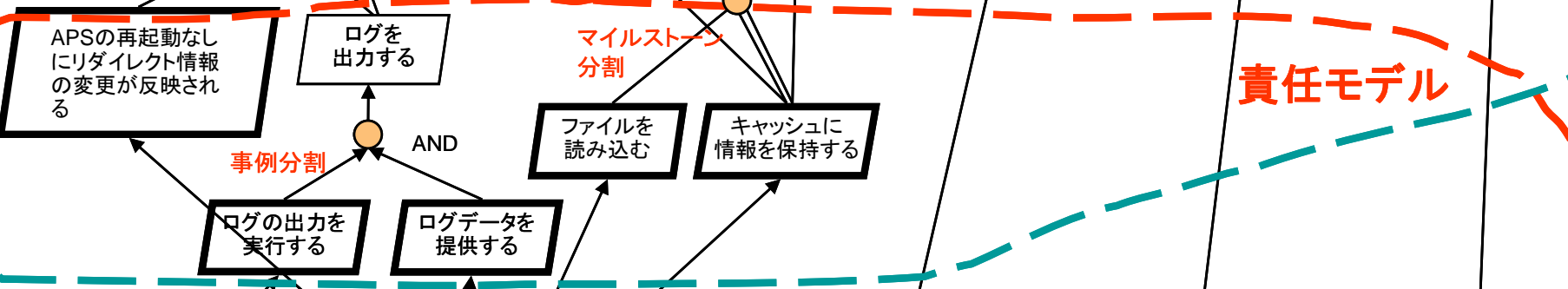
品質特性群



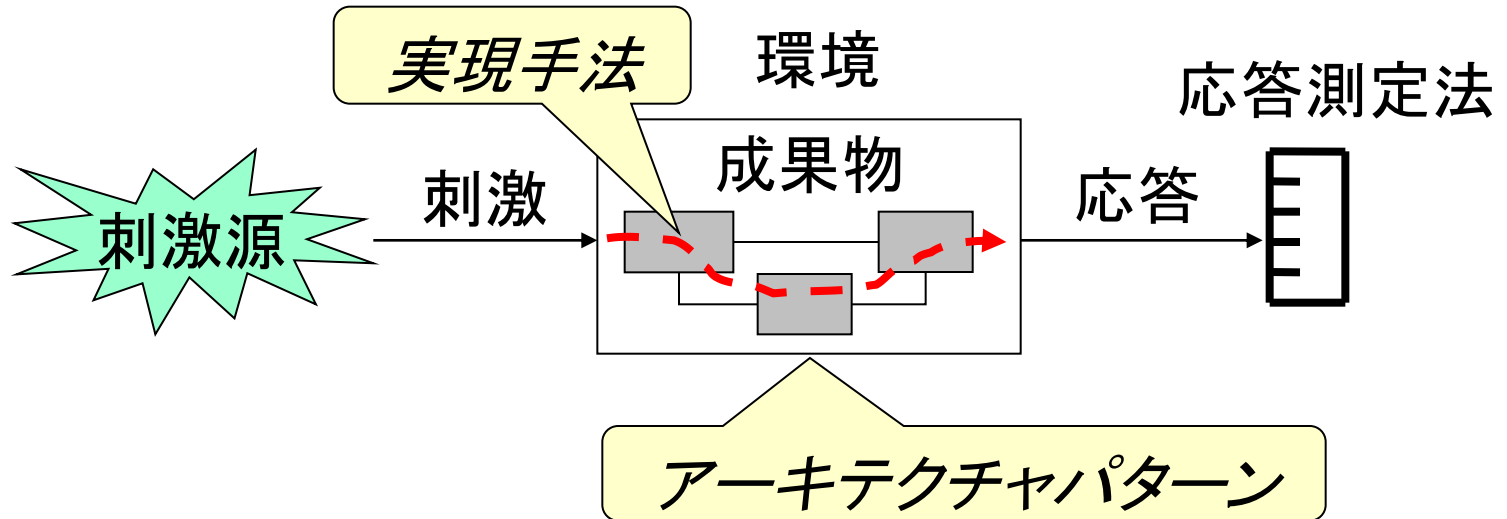
ゴールモデル



責任モデル



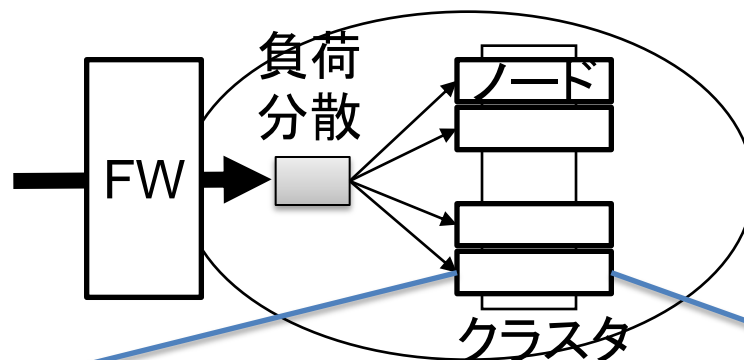
設計: パターンと品質駆動設計



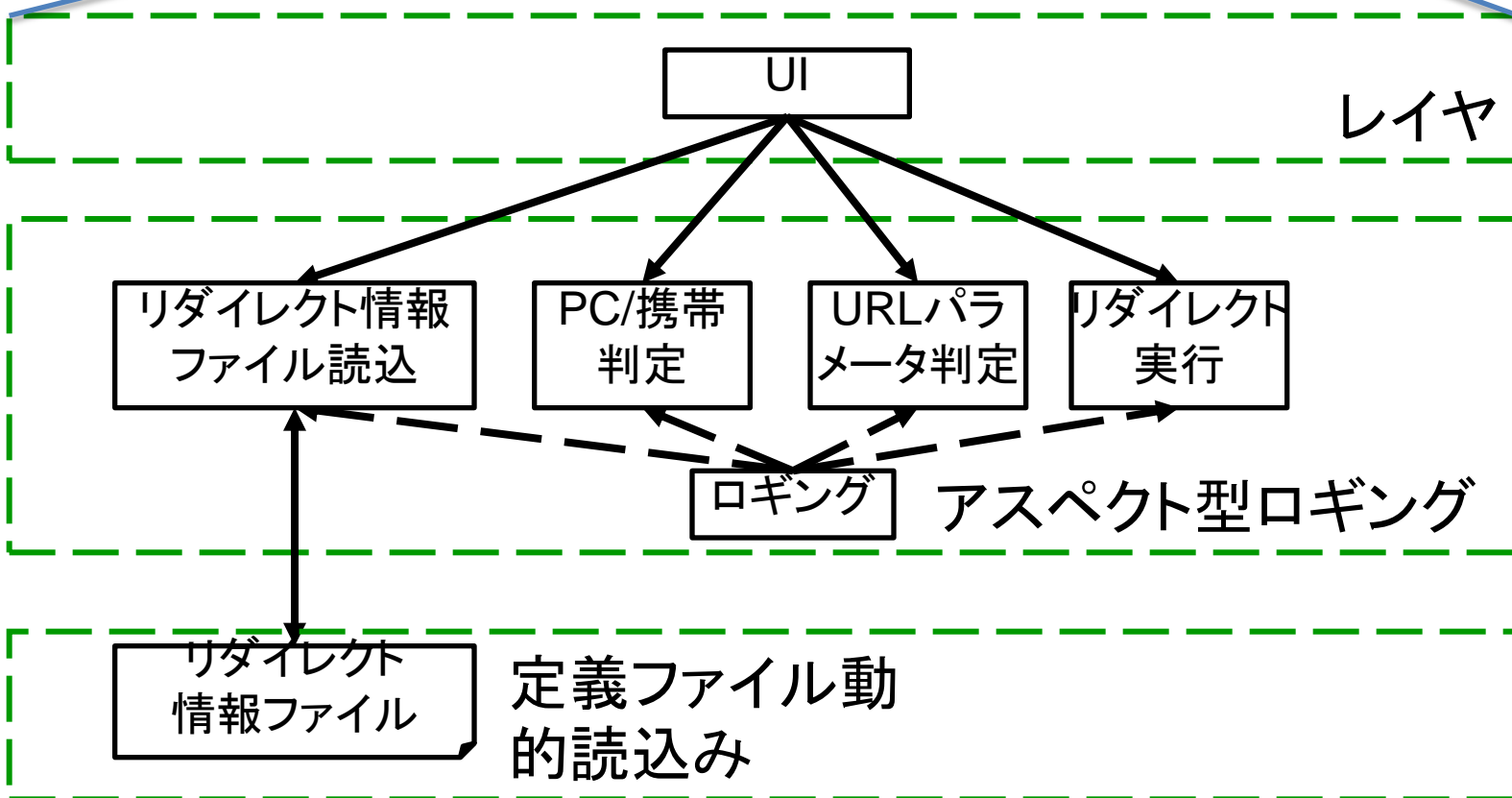
アーキテクチャパターン

	定義ファイル 動的読み込み	多重化・ 負荷分散
A7. リダイレクト情報の 変更を停止無しで反映	↑	
A1. HTTPリクエストを秒間 1000トランザクション処理	↓	↑

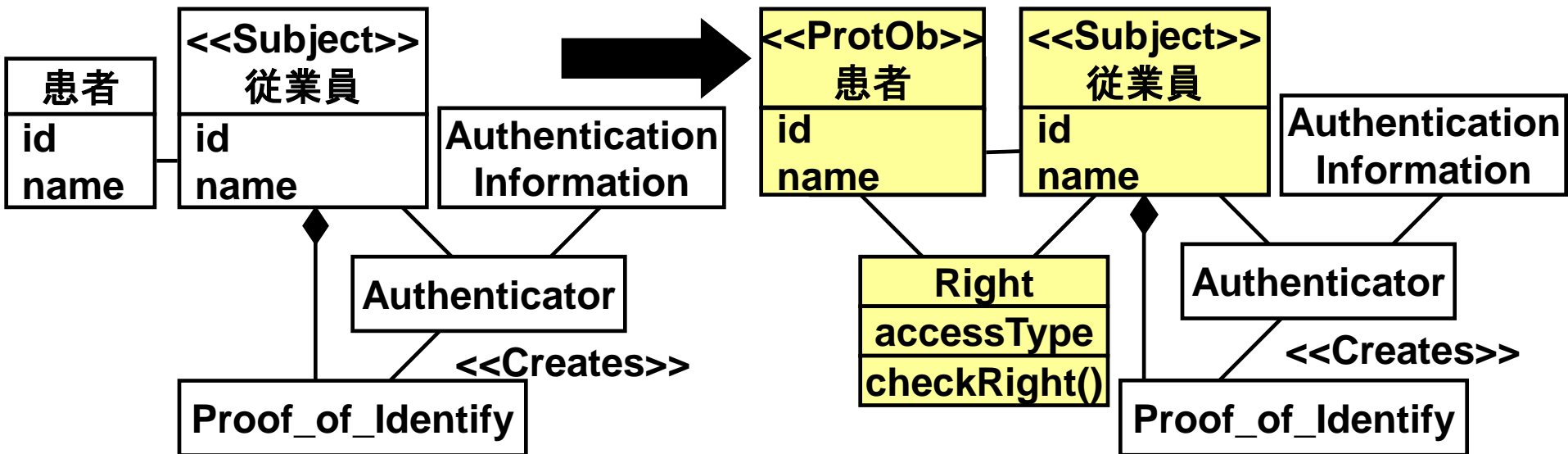
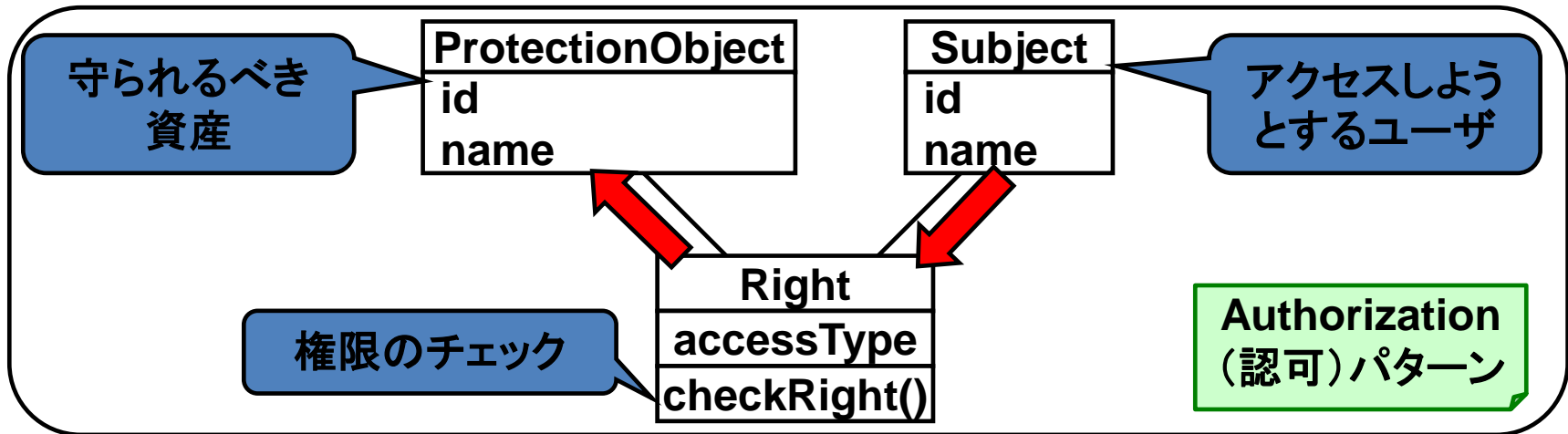
システム・基盤



アプリケーション



MDDとセキュリティパターン



```
window.onload = setEventHandler;
function setEventHandler() {
    $("reg_type").onchange = calcPrice;
    ...
    $("reg_addcart").onclick = addCart;
};

function calcPrice() { ... };
function addCart() {
    if(isValidInput()) {
        reqRunTrans();
    } else {
        alert("Invalid user inputs");
    }
};

function reqRunTrans() {
    new Ajax.Request("runTrans.php", {
        method: "GET", parameters: getParams(),
        onSuccess: succeeded });
};

function succeeded() { disableAll();
    jumpToConfirm();
};
```

Price: \$500

Type

All days ▾

Attendee

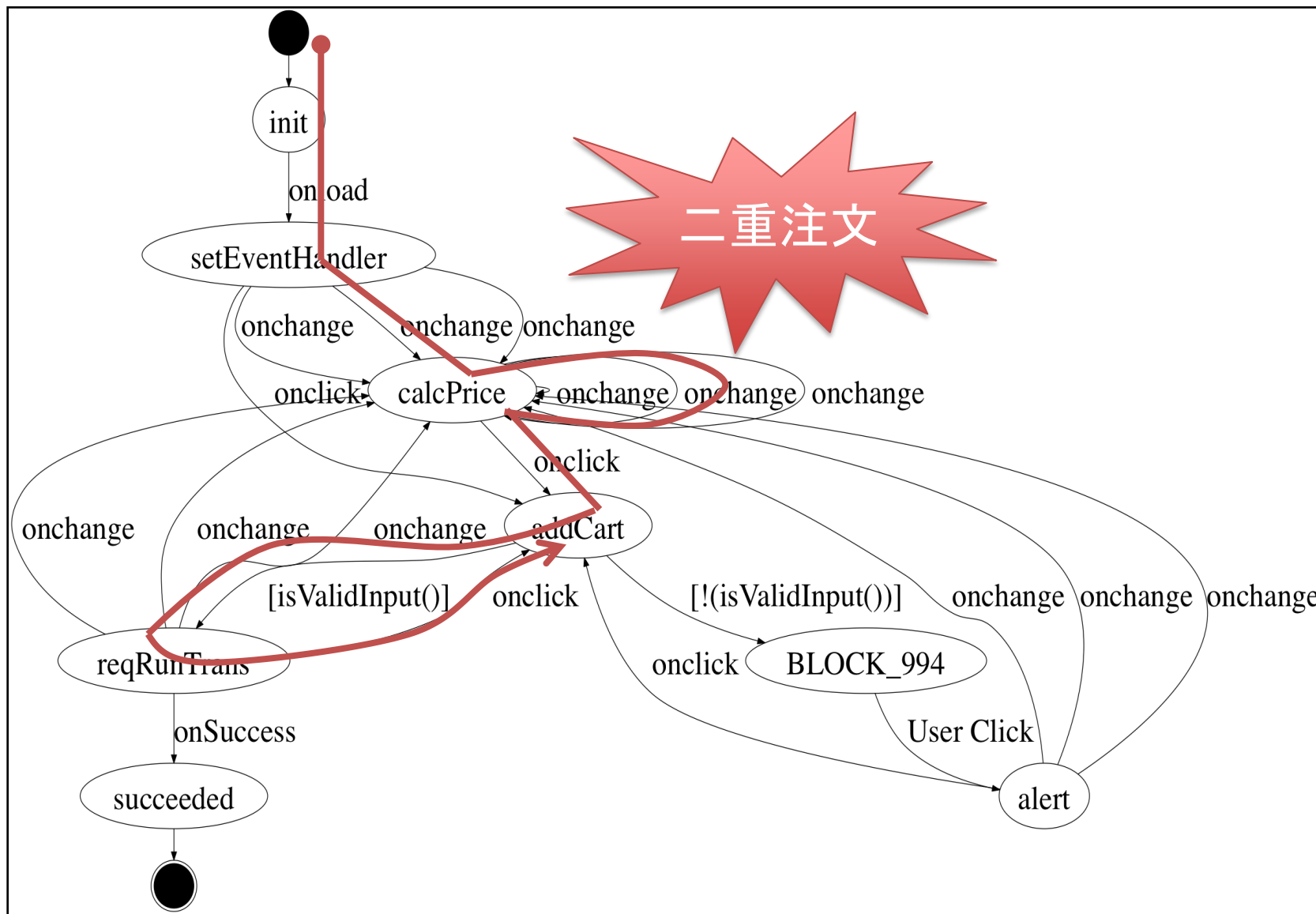
Regular ▾

Payment

Early ▾

Quantity:

モデルのリバーズエンジニアリング



```

window.onload = setEventHandler;
function setEventHandler() {
    $("reg_type").onchange = calcPrice;
    ...
    $("reg_addcart").onclick = addCart;
};

function calcPrice() { ... };
function addCart() {
    if(isValidInput()) {
        $("#addCart").disabled = true;
        reqRunTrans();
    } else {
        alert("Invalid user inputs");
        $("#addCart").disabled = false;
    }
};

function reqRunTrans() {
    new Ajax.Request("runTrans.php", {
        method: "GET", parameters: getParams(),
        onSuccess: succeeded }); };
function succeeded() { disableAll();
    jumpToConfirm(); };

```

Price: \$500

Type

All days ▾

Attendee

Regular ▾

Payment

Early ▾

Quantity:

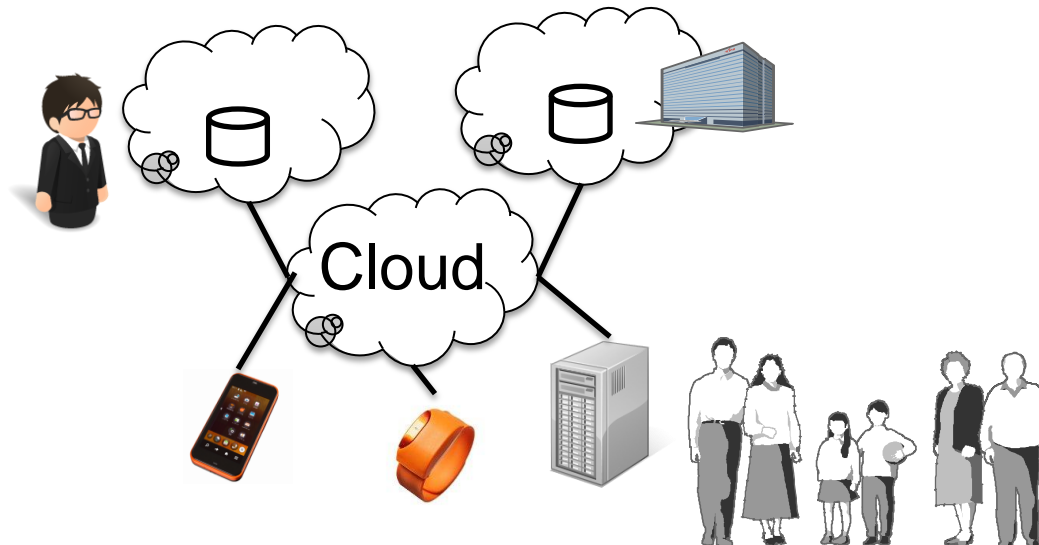
目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- モデリングと品質
- IoT時代のモデリング: 参照アーキテクチャと
接続
- IoT時代のモデリング: 測定・学習さらには育
成
- まとめ



IoT/CPSの時代

- Internet of Things
 - 物理世界にあるモノのインターネットによる接続
 - IoEへ: データ・情報、生体の接続
- Cyber Physical System
 - 物理世界の情報とサイバー世界の情報の融合
- 効率化、新ビジネス・価値の創発



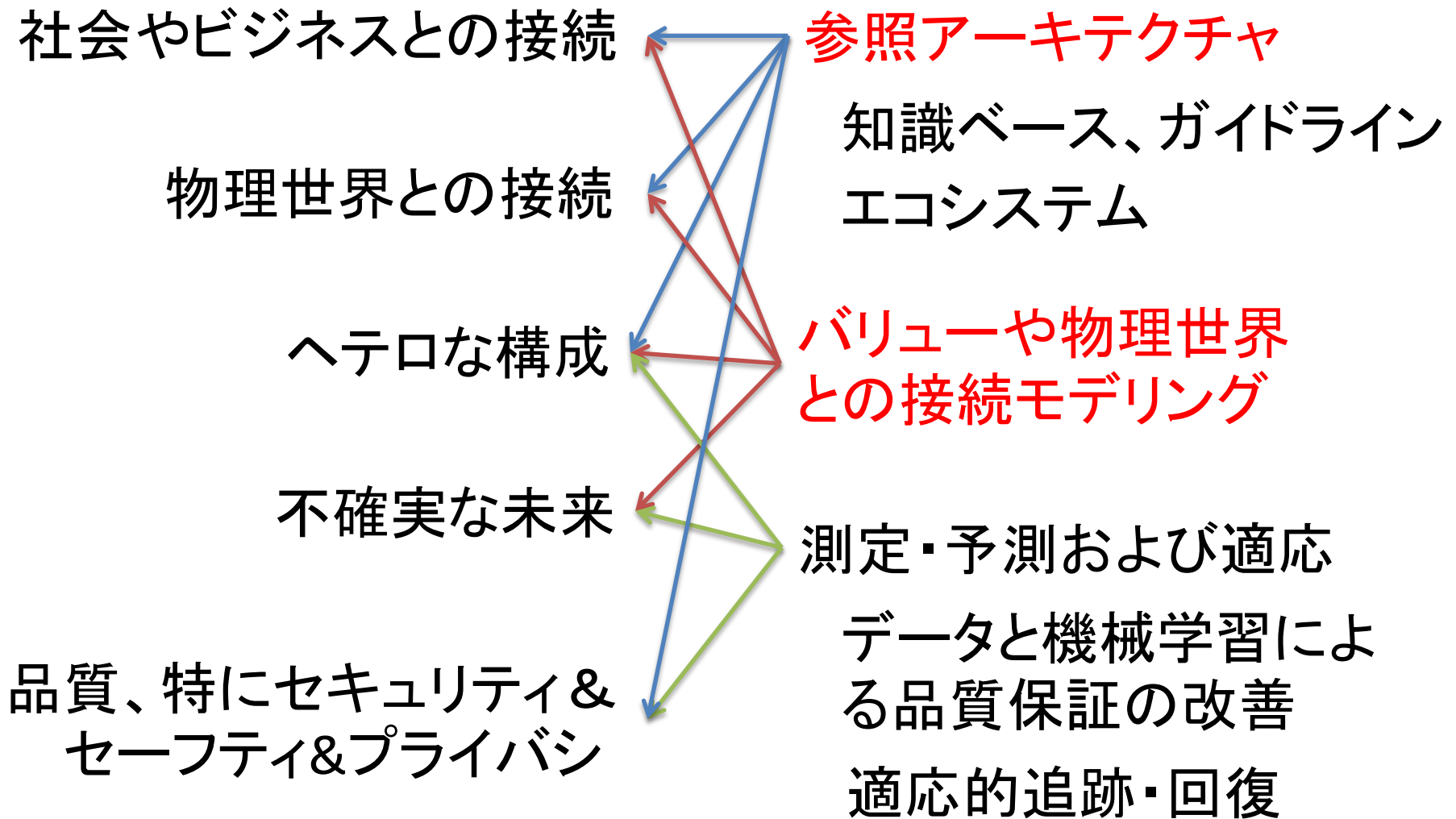
IoT/CPS時代の課題

- 社会やビジネスとの接続
 - バリューをどう創発するか？
- 物理世界との接続
 - ギャップは？(連続/離散、(非)同期)
- ヘテロな構成
 - 異領域やライフサイクルをどう扱うか？
- 不確実な未来
 - 変化を予測できるか？対応できるか？
- 品質、特にセキュリティ&セーフティ&プライバシー
 - リスクの増大、全体を保証できるか？



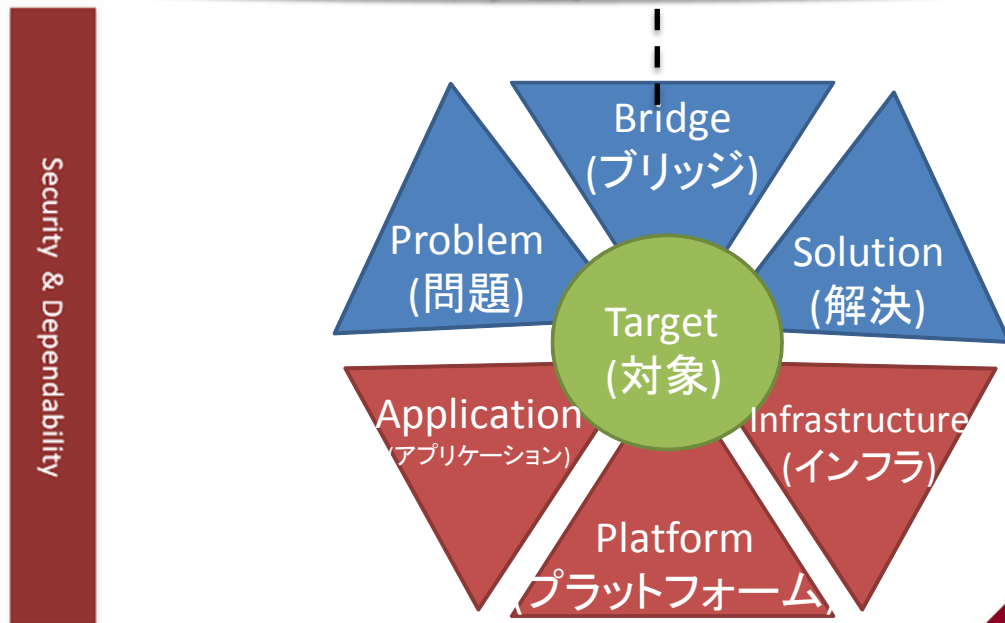
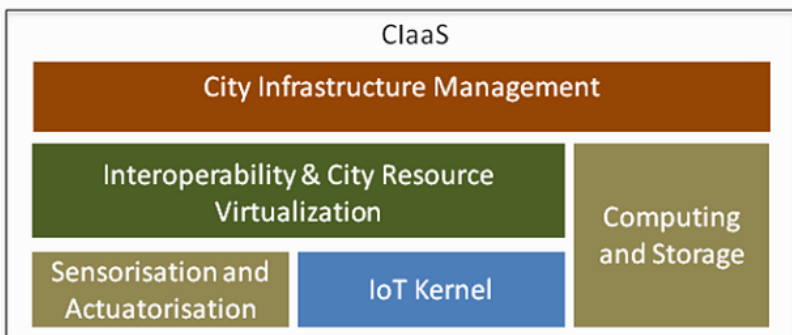
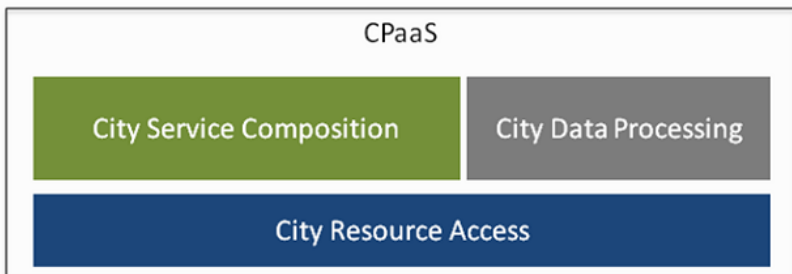
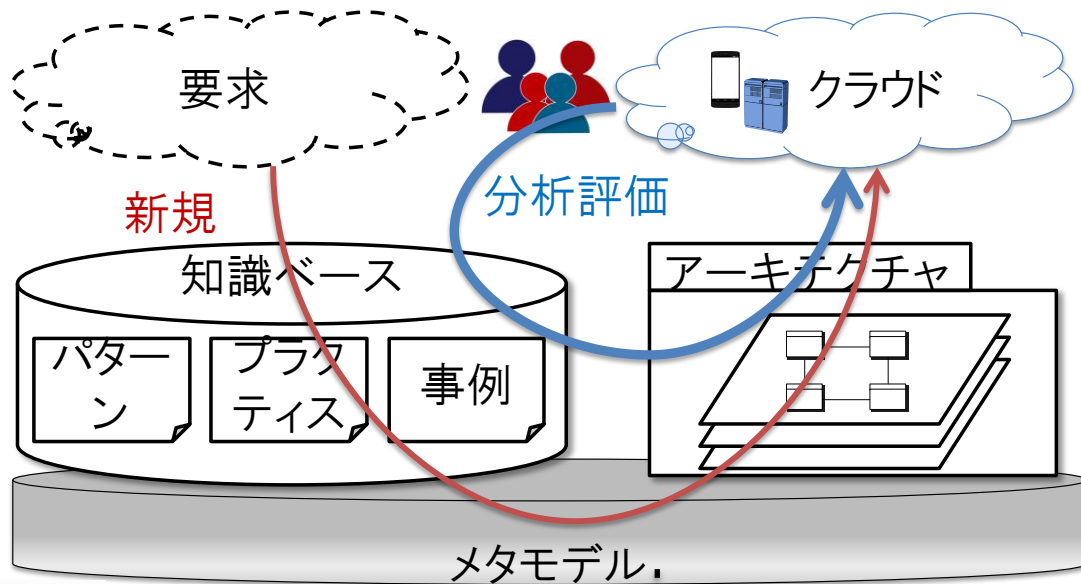
利用時の品質 Quality in use							
有効性 Effectiveness	効率性 Efficiency	リスク回避性 Freedom from risk	満足性 Satisfaction	利用状況網羅性 Context coverage			
内部品質・外部品質 Internal and external quality							
機能適合性 Functional suitability	互換性 Compatibility	セキュリティ Security	信頼性 Reliability	使用性 Usability	性能効率性 Performance efficiency	保守性 Maintainability	移植性 Portability
Functional completeness	Co-existence	Confidentiality	Maturity	Appropriateness recognisability	Time-behavior	Analysability	Adaptability
Functional correctness	Interoperability	Integrity	Fault tolerance	Learnability	Resource utilization	Modifiability	Installability
Functional appropriateness		Non-repudiation	Recoverability	Operability	Capacity	Reusability	Replaceability
		Accountability	Availability	User error protection		Testability	
		Authenticity		UI aesthetics		Modularity	
			Accessibility				

IoT/CPS時代の挑戦の可能性



参照アーキテクチャに向けて

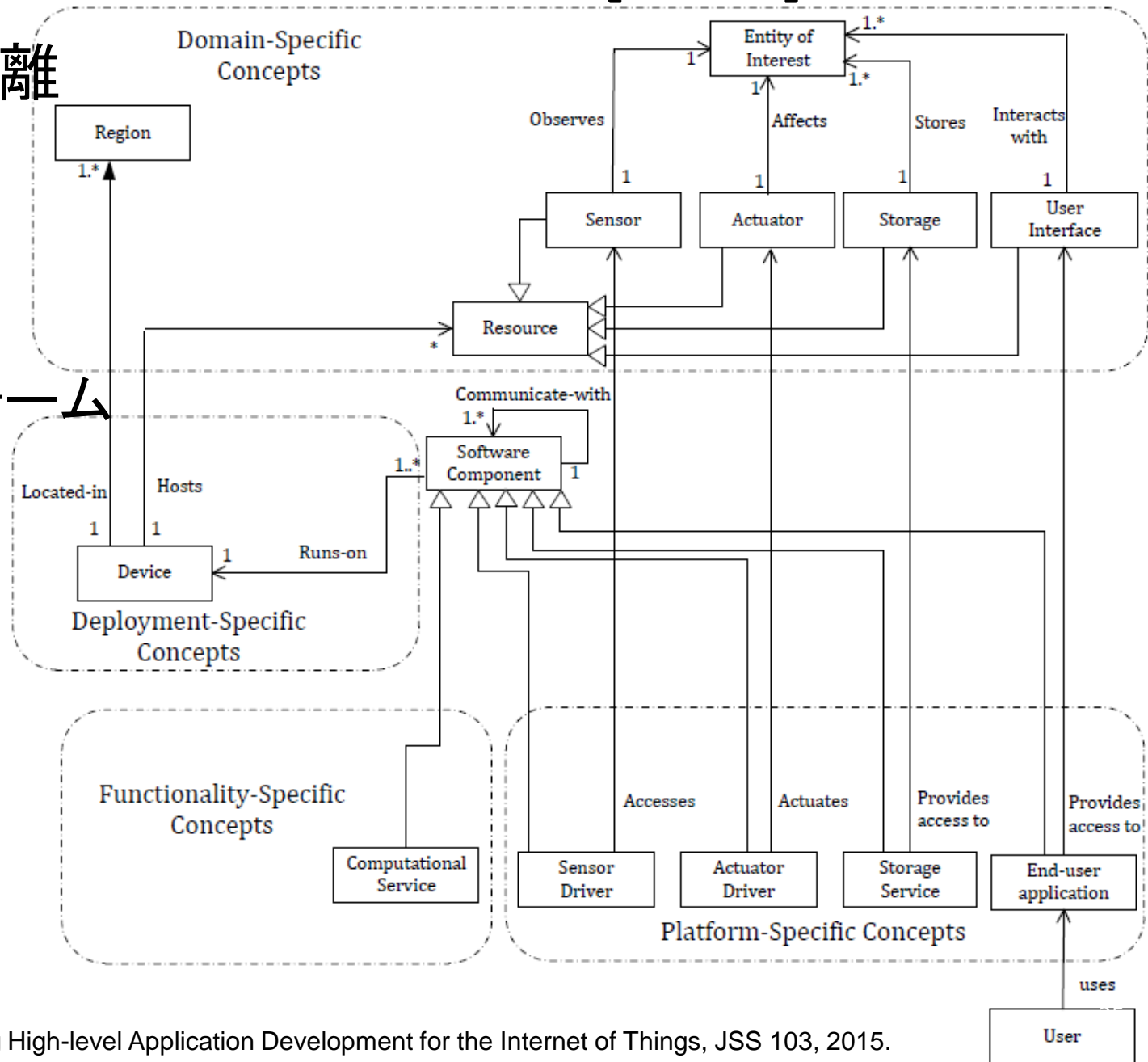
- IISF-SSR「クラウドサービス・セキュリティ・プライバシーシメタモデル」代表: 鷲崎
<http://www.iisf.or.jp/SSR/>
- EU ClouTプロジェクト
<http://clout-project.eu/>



モデル駆動IoT開発[Patel]

● 関心事の分離

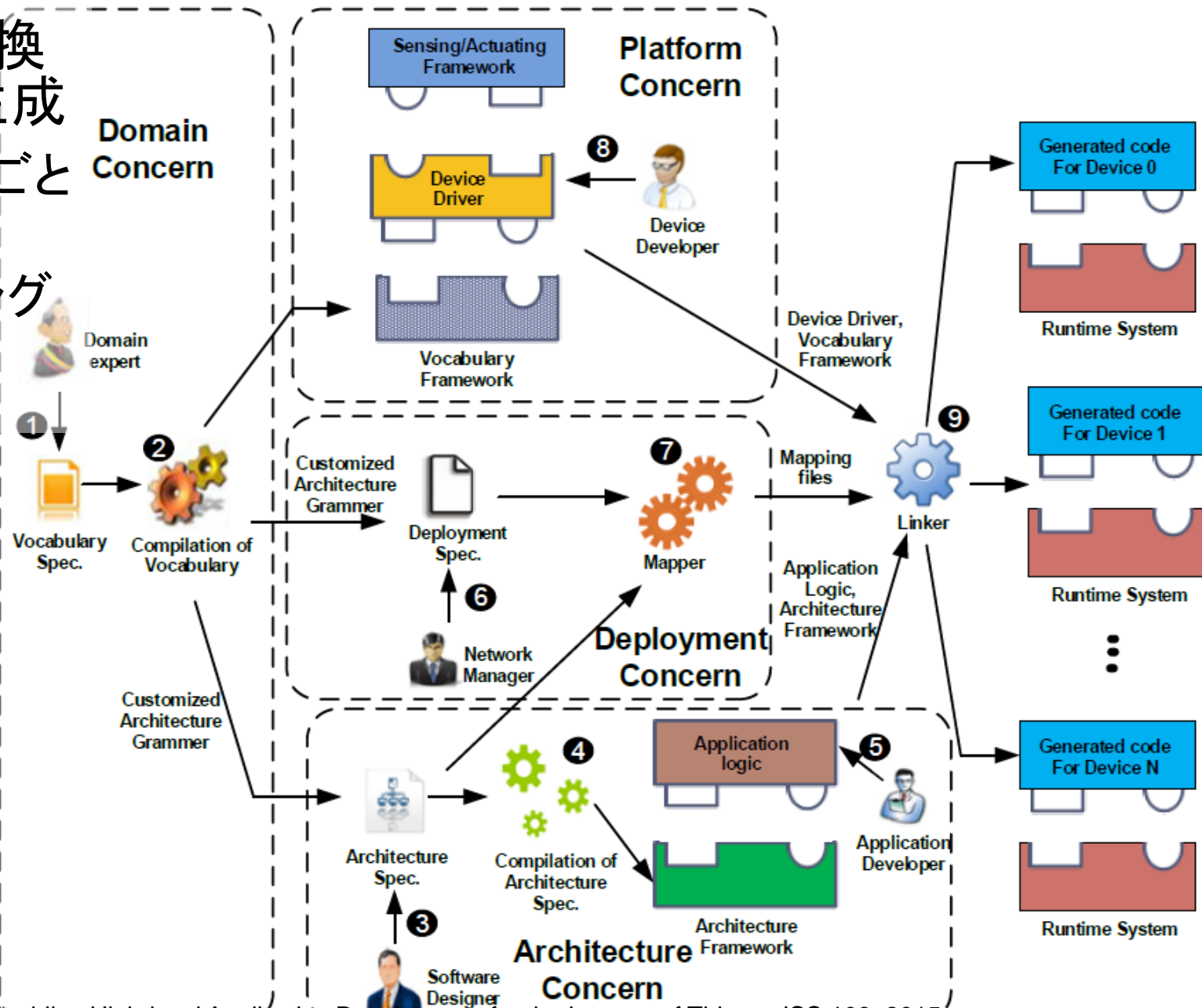
- ドメイン
- 機能
- 配備
- プラットフォーム



モデル駆動IoT開発(つづき) [Patel]

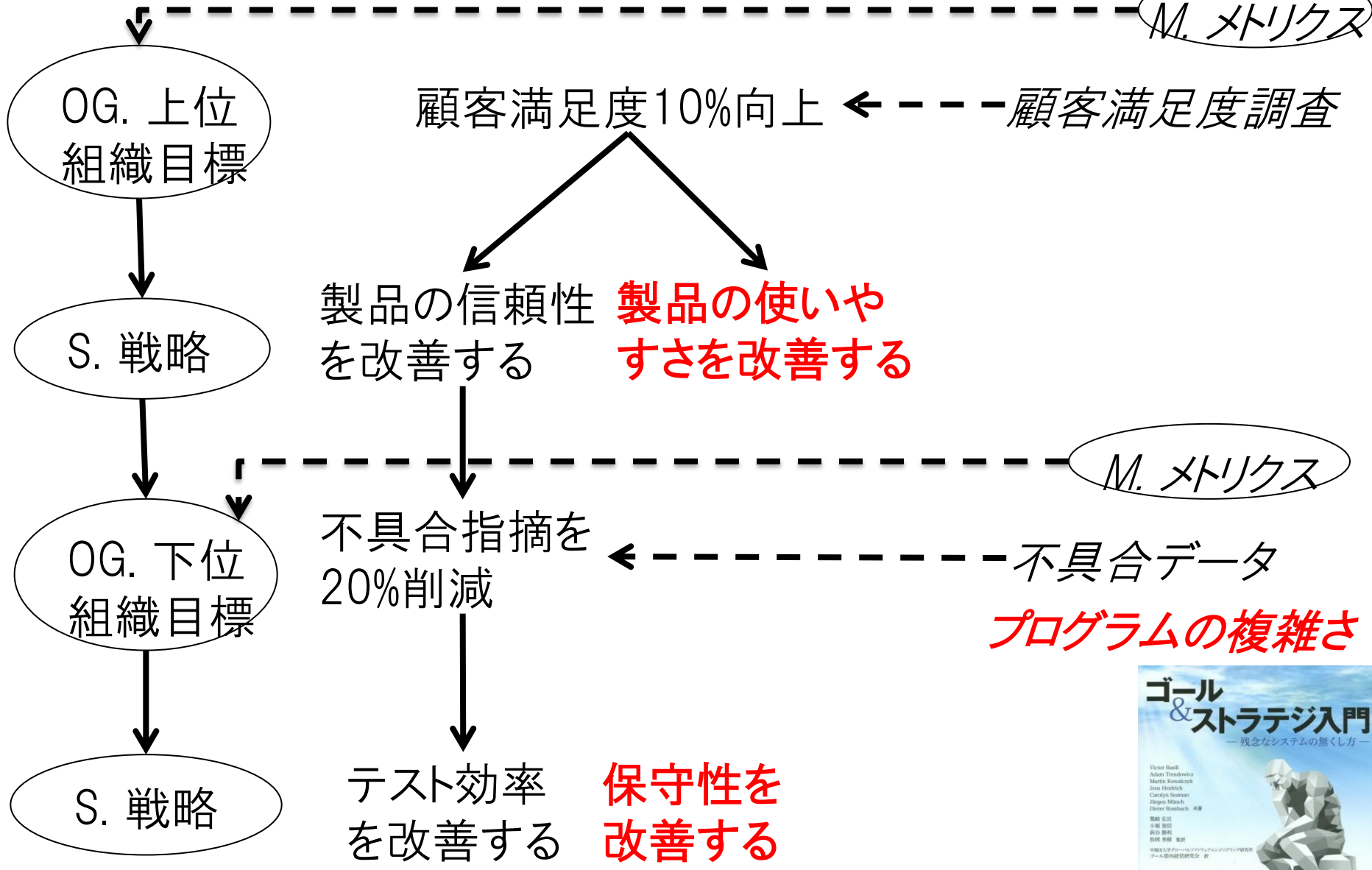
- モデル変換とコード生成

- 関心事ごとのDSL
- マッピングとリンク



ゴール指向による接続・整合 GQM+Strateges

M. メトリクス



Jens Heidrich, Adam Trendowicz, “測定を基にした、ソフトウェア戦略とビジネス目標の整合” IPA/SEC資料
鷲崎ほか監訳, ゴール&ストラテジ入門: 残念なシステムの無くし方 (GQM+Strateges), オーム社, 2015



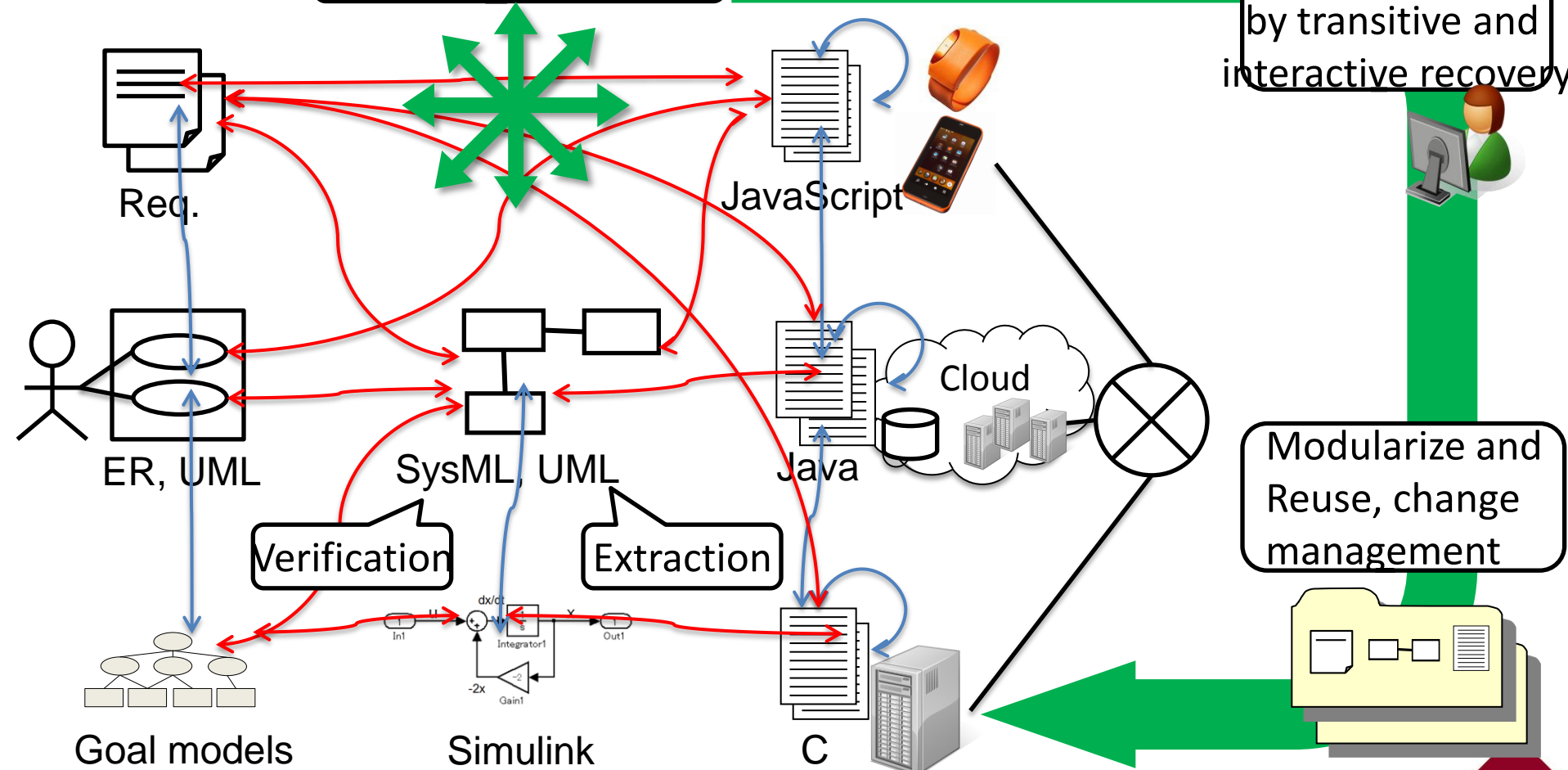
Trace ANY ('16-'20 代表: 鷺崎)



Trace in all directions

Reliable trace by transitive and interactive recovery

Modularize and Reuse, change management

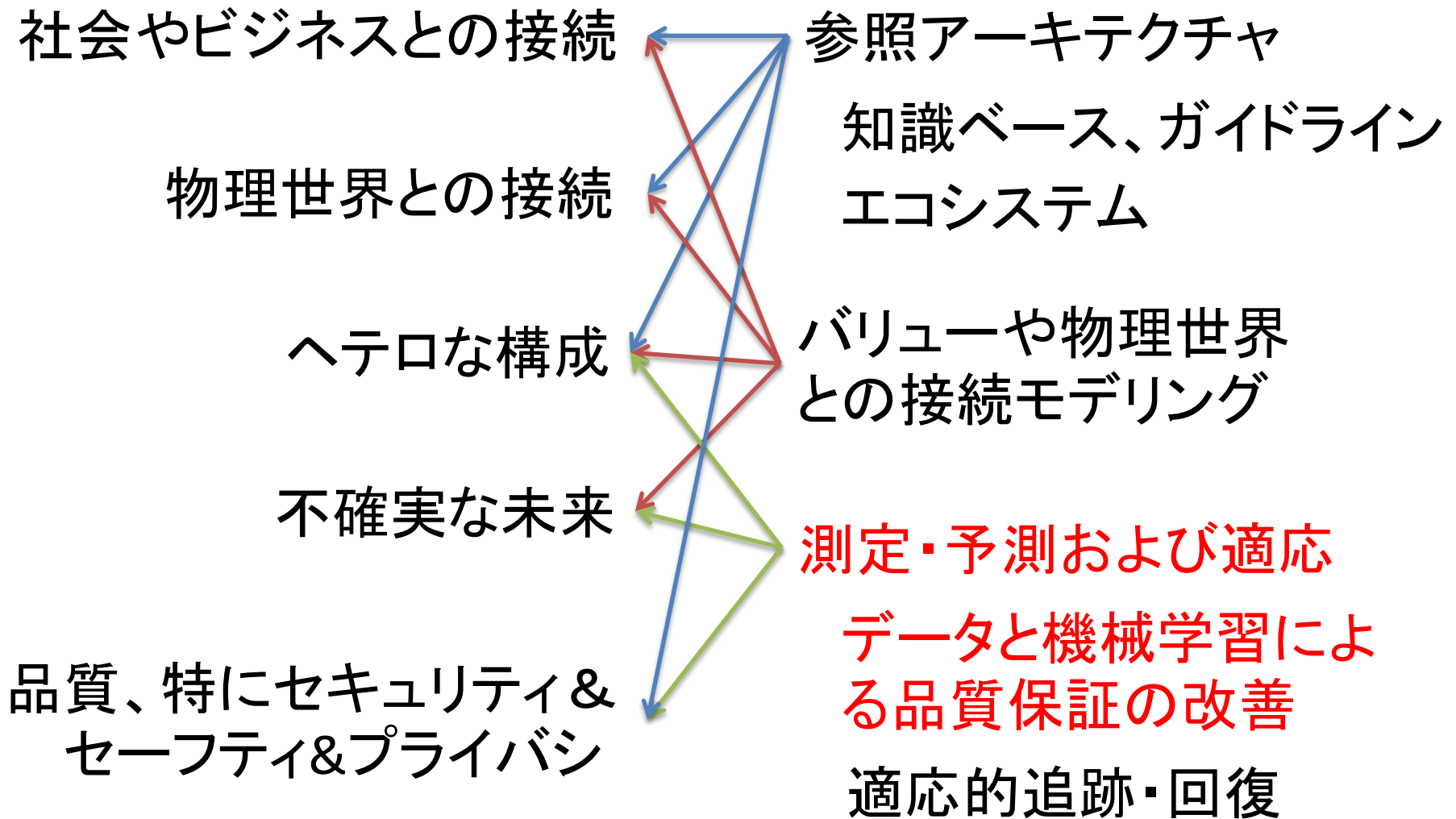


目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- モデリングと品質
- IoT時代のモデリング: 参照アーキテクチャと接続
- IoT時代のモデリング: 測定・学習さらには育成
- まとめ

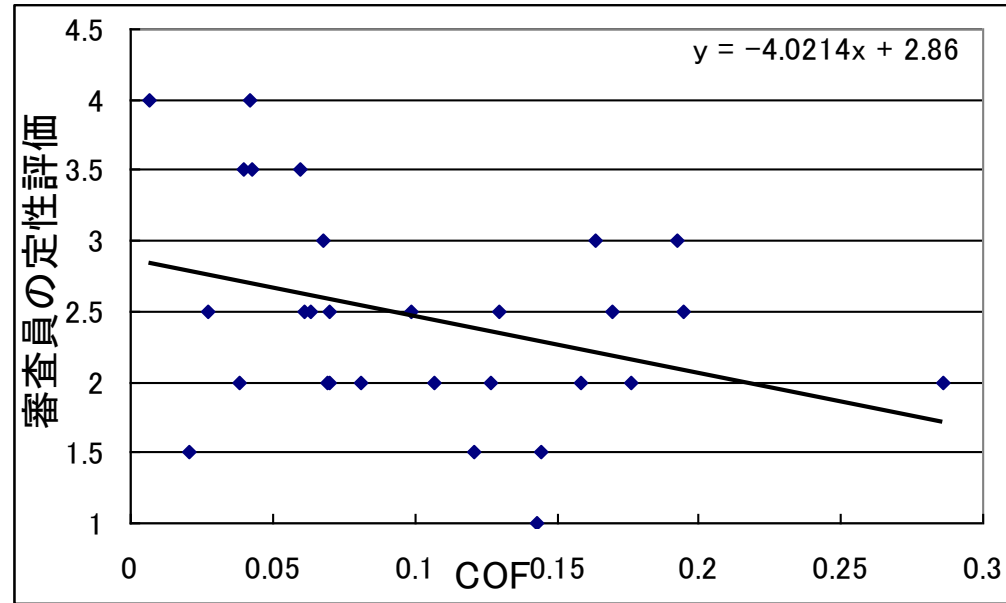


IoT/CPS時代の挑戦の可能性



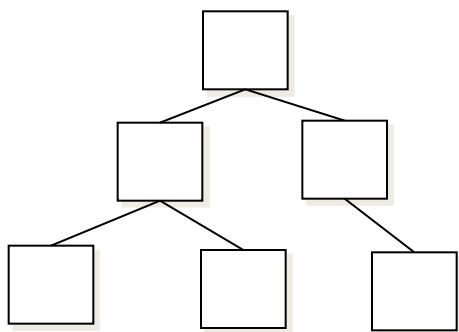
品質測定: 構造的複雑さ(結合性)

- Coupling factor (COF) [Abreuら 95]
 - オブジェクト指向クラス間凝集度のメトリック
 - ETロボコン実証 [Washizaki06]

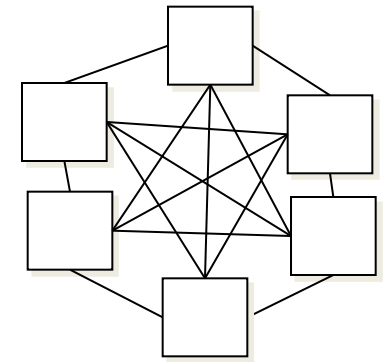
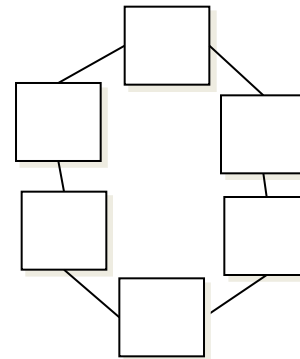
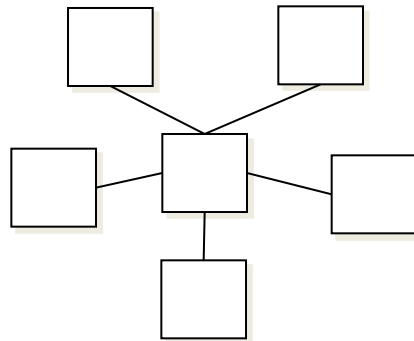


(継承を除く有向参照辺の数)

$(\text{頂点数})^2 - (\text{頂点数}) - (\text{全継承子クラス数} \times 2)$



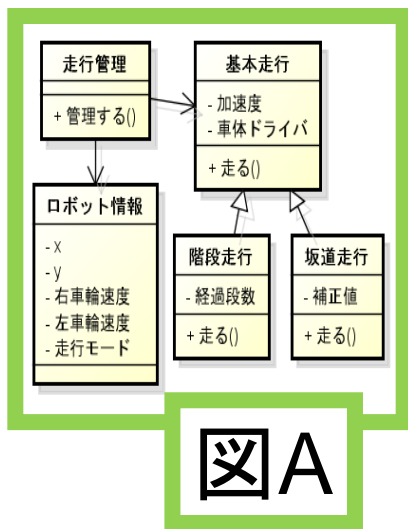
モジュール数: n



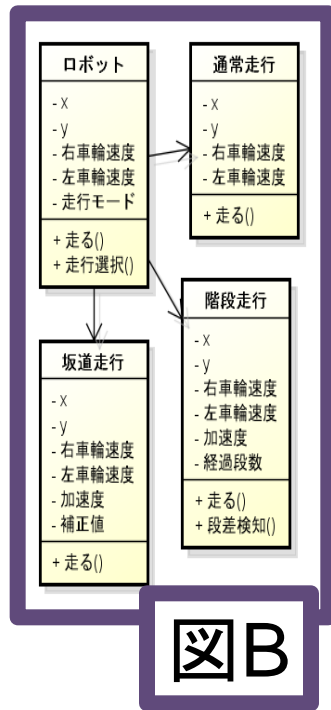
$n(n-1)/2$

品質測定: 構造的複雑さ(凝集性)

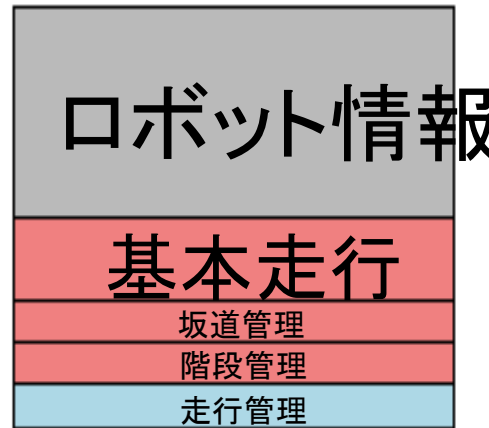
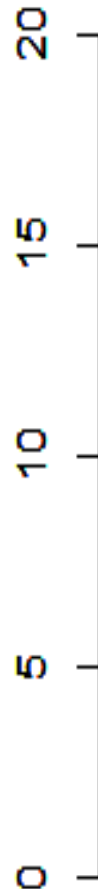
- 責務が大きいクラスを含まない設計モデルは構造を十分検討されていない



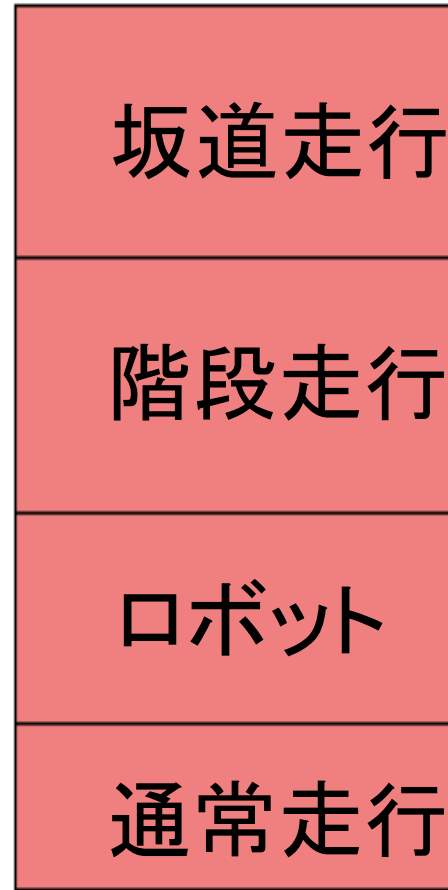
図A



図B



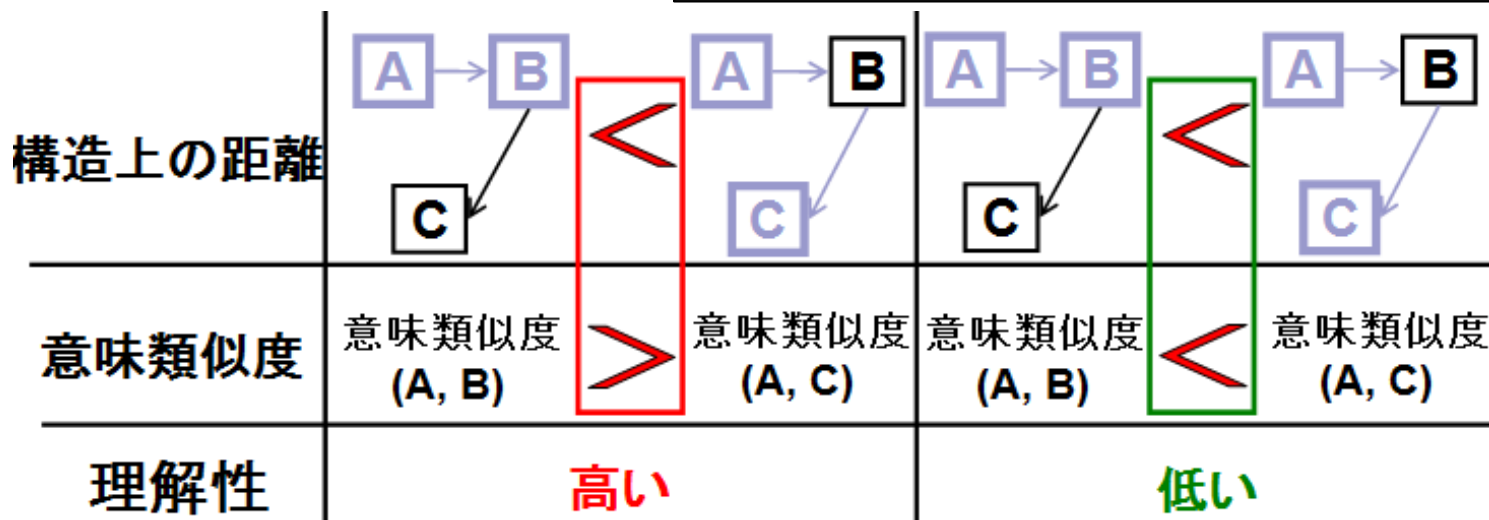
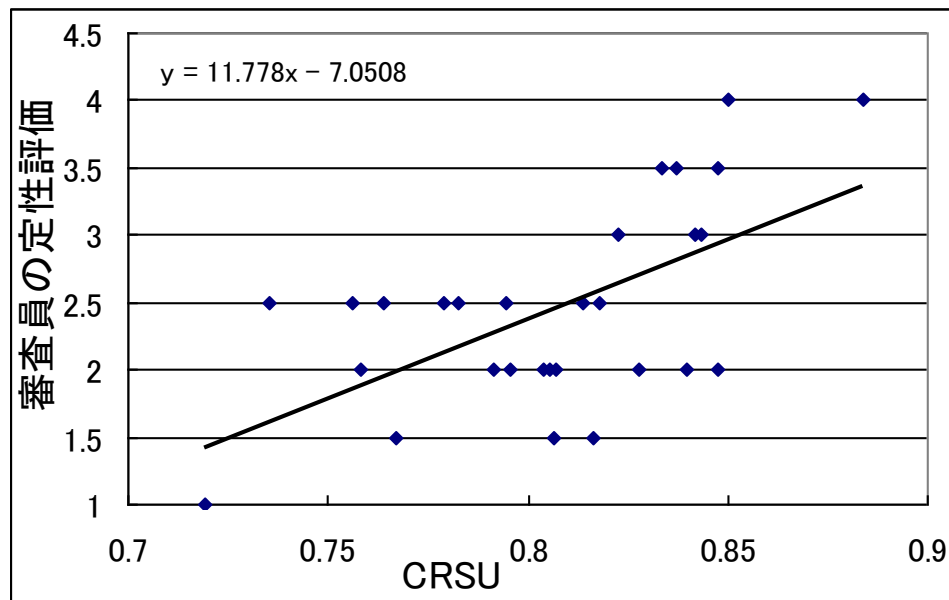
図A



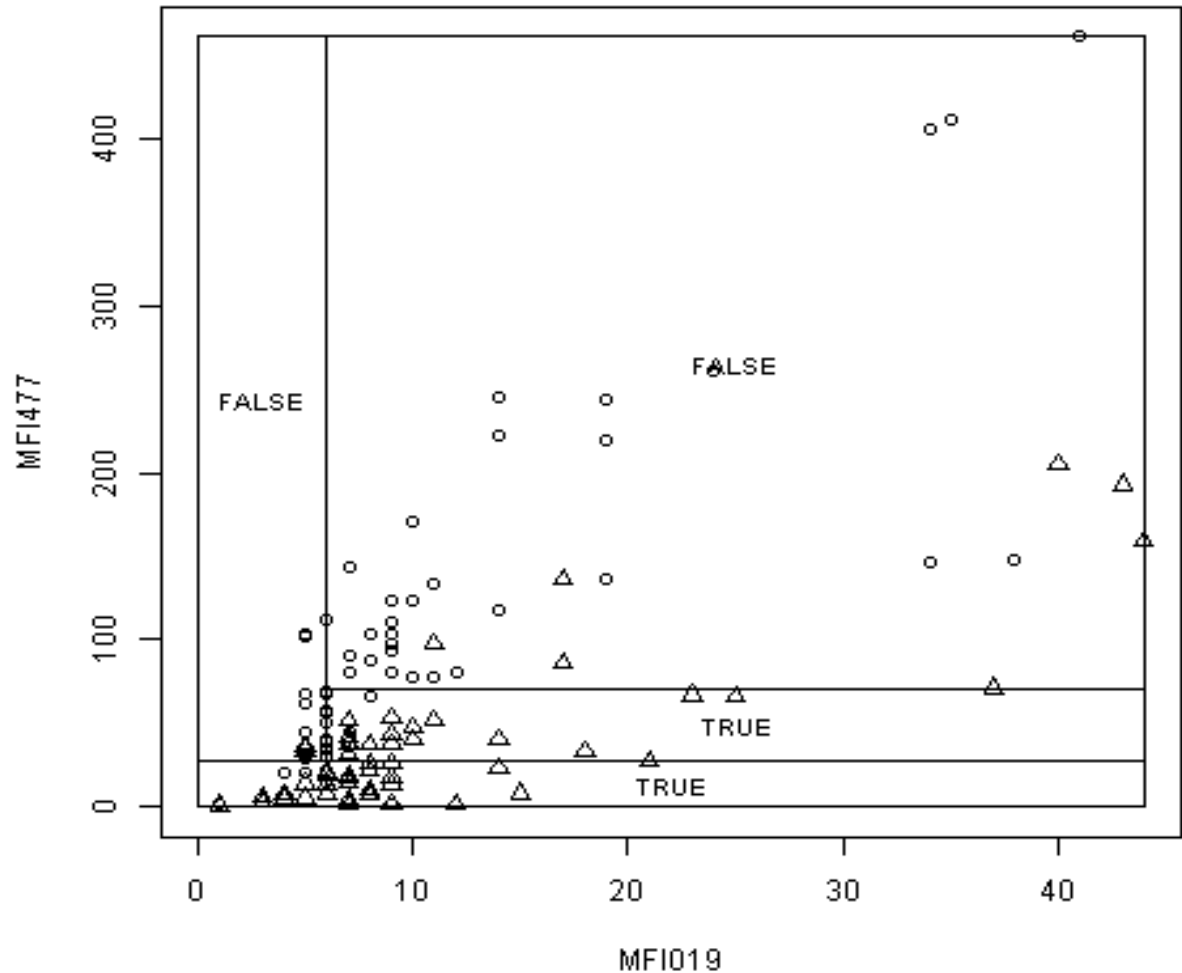
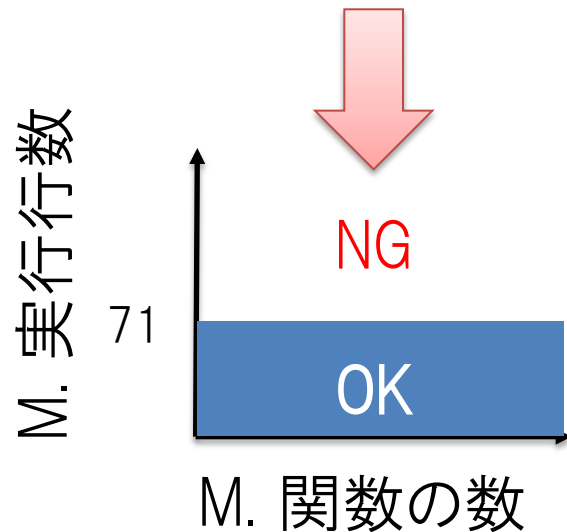
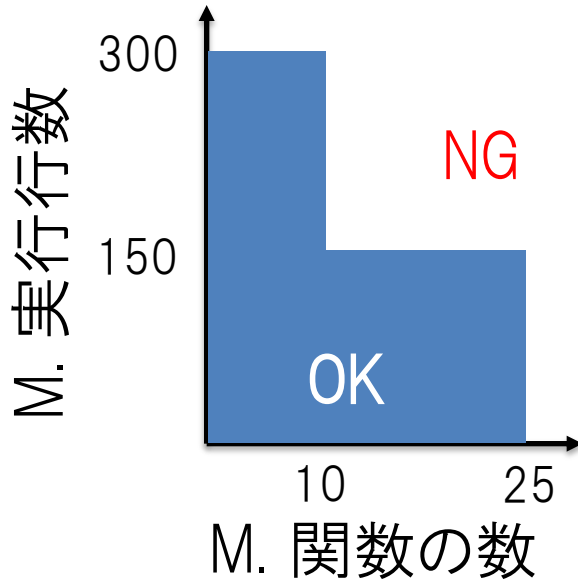
図B

品質測定: 意味的複雑さ

- Class Relational Structure Understandability (CRSU) [Nakamura11]
 - 構造と意味のギャップの定量化



事例: 機械学習によるメトリクス・システム改善



人材育成に向けて

- 抽象化能力がカギ [Kramer07]
 - 詳細の捨象、共通部の一般化
 - 「測定」による能力把握と向上
 - 数学、問題中心のモデリング作業による習得
- 様々な領域把握と接続がカギ
 - 様々なビジネス、システム、ソフトウェア
 - ゴール指向
- アーキテクチャ設計能力がカギ
 - 様々な品質特性: セキュリティ、セーフティー・・・
 - 品質駆動設計
 - 各種パターン

目次

- 正統なエンジニアリングと知識体系
- モデリングとモデル駆動開発
- モデリングと品質
- IoT時代のモデリング: 参照アーキテクチャと接続
- IoT時代のモデリング: 測定・学習さらには育成
- まとめ



取り上げた技術のSWEBOK上のつながり

要求	設計	構築	テスト	保守
要求の基礎 要求プロセス 要求抽出 要求分析 要求管理	設計の基礎 設計の主要問題 構造とアーキテクチャ ユーザインタフェース設計 設計品質の分析評価 設計の表記 設計戦略・手法 設計ツール	構築の基礎 構築の管理 実践上の考慮事項 構築技法 構築ツール	テストの基礎 テストレベル テスト技法 テスト関連計量尺度 テストプロセス テストツール	保守の基礎 保守の主要問題 保守プロセス 保守技法 保守ツール

シナリオ
ゴール指向分析

品質駆動設計・評価
パターン
モデルの測定

品質特性

構成管理
SCMプロセスのマネジメント
構成の識別
構成制御
構成状態記録・報告
構成監査
リリース管理・配布
構成管理ツール

マネジメント
開始と範囲定義
プロジェクト計画
プロジェクト実施
レビュー・評価
終結
計量
マネジメントツール

プロセス定義
ライフサイクル
プロセスアセスメント・改善
プロセス計量
プロセスツール

モデル・手法
モデリング
モデル種別
モデルの分類
開発手法
UML
MDD

品質
品質の基礎
品質マネジメントプロセス
実践上の考慮事項
品質ツール

形式検証

プロフェッショナル実践
プロフェッショナリズム
集団ダイナミクス・心理
コミュニケーションスキル

経済
経済の基礎
ライフサイクル経済
リスク・不確実性
経済分析手法
実践上の考慮事項

計算基礎
クラウド

数学基礎

エンジニアリング基礎
セキュリティパターン

まとめ

- 共通基盤上でより良く、早く、幸せに
 - 知識体系上での知識の整理、深耕、拡大
 - プロフェッショナリズム
- モデリング再考
 - モデリング成熟度とモデル駆動開発
- モデリングと品質
 - 検証、作り込み
- IoT/CPS時代の課題と挑戦
 - 品質特性の移り変わり
 - モデル駆動開発と参照アーキテクチャ
 - バリューとの接続モデル
 - 測定とシステム改善
 - 人材育成

