JEITA 組込み系ソフトウェア・ワークショップ2015
組込み系開発の実践的モデリング～モデリングを成功させるには～

モデリングの原理と組込みシステム開発における実践へのガイド

青山 幹雄
南山大学　理工学部 ソフトウェア工学科
大学院 理工学研究科 ソフトウェア工学専攻
mikio.aoyama@nifty.com
http://www.nise.org/

2015年11月13日（金）

お伝えしたいこと

1. モデル=仕様(Specification)
2. モデリングのない開発=仕様のない開発
3. 良いモデリング=良い開発&良いシステム

シナリオ

1. モデリングとは
2. 組込みシステムのモデリング
3. 実践へのガイド

モデリングとは
モデルとは

☞ モデル(Model)は実世界の抽象=仕様(Specification)

- A model is an abstraction of a (real or language-based) system allowing predictions or inferences to be made [Kühne '06]

Google Maps
「南山大学
名古屋
キャンパス」
付近

モデル=地図

実世界=航空写真

参考文献: T. Kühne, Matters of (Meta-) Modeling, J. of Software and Systems Modeling, Vol. 5, No. 4, Dec. 2006, pp. 369-385.

モデリングとは
モデリングの黄金律とモデルが持つべき特性

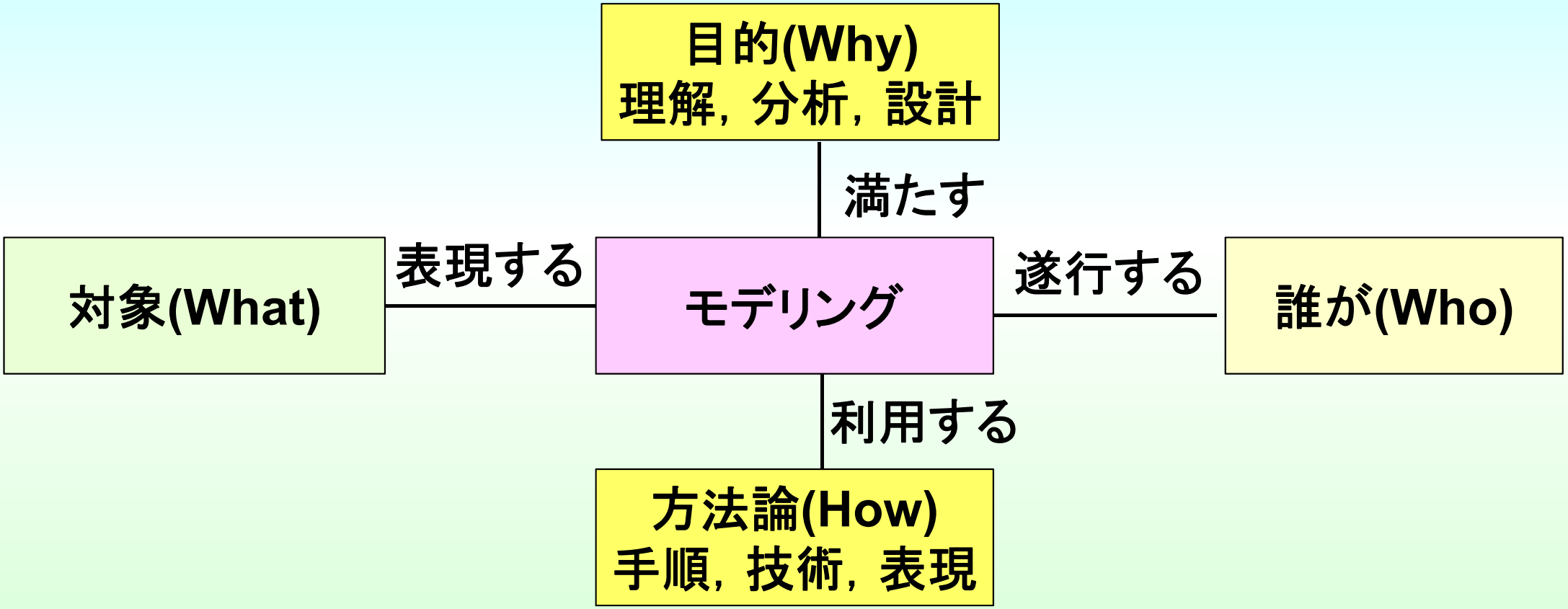
- **モデリングの黄金律(Golden Rule of Modeling)**
  - モデルの複雑度は対象を表現できる必要十分であるべき [A model shall be as complex as necessary; no more or less]
- **モデルの持つべき特性　[Seidewitz '03]**
  - 写像(Mapping): 実世界の「もの」「こと」と対応
    - Model is based on an original (system)
  - 簡略化(Reduction): 実世界の「もの」「こと」の適切な性質の抽出
    - A model only reflects a relevant selection of an original's properties
  - 実用性(Pragmatic): 特定の目的をもって，工学的な用途に利用可能
    - A model needs to be usable in place of an original with respect to some purpose.

モデルは
射影(Projection)
[部分空間への写像]
と呼ぶこともある

参考文献: B. A. Lieberman, The Art of Software Modeling, Auerbach Pub., 2007.
E. Seidewitz, What Models Mean, IEEE Software, Vol. 20, No. 5, Sep./Oct. 2003,pp. 26-32.

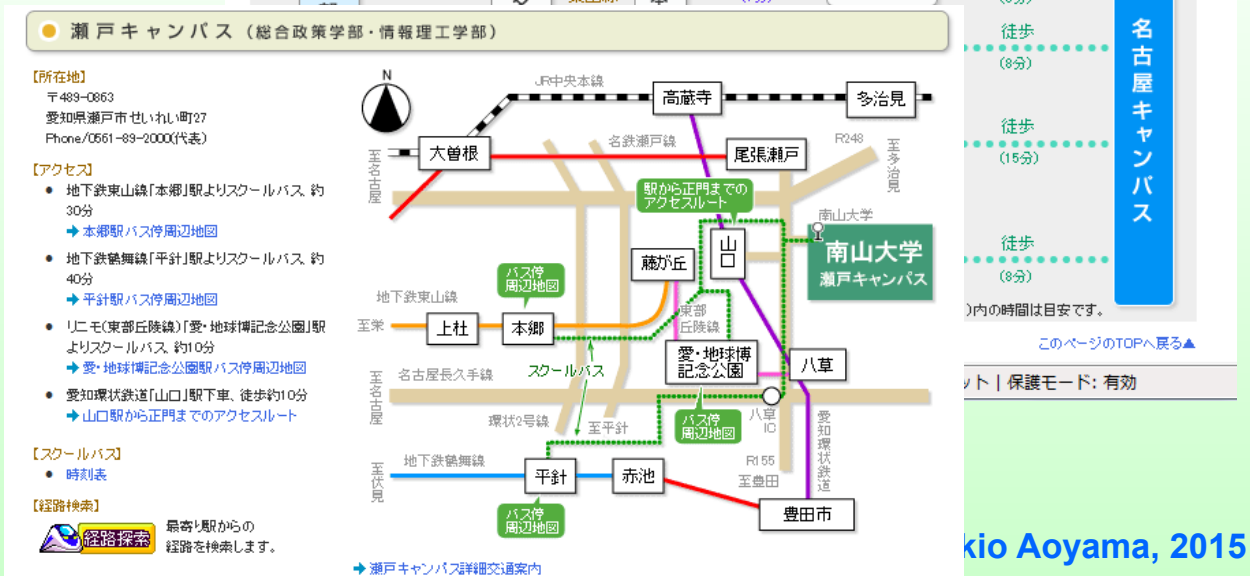
モデリングとは
モデルのモデル

☞ モデリングの(メタ)モデル



モデリングとは
モデルの目的

- ちなみに，南山大学のWebページで
- 「名古屋キャンパス」のWebページは?
- モデルには目的がある
  - 目的＝「当該場所への誘導」
  - 必要な情報を必要なだけ



モデリングとは
目的と表現形

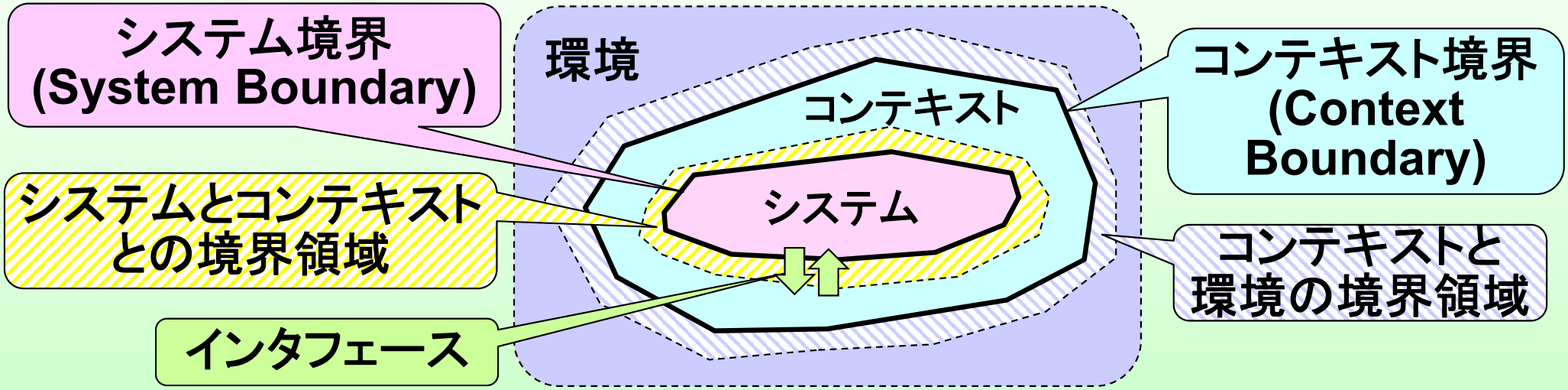
☞モデルは目的に適した形(Form)[表現(Representation)]をとる

目　的	モデルの形/表現
予測(Prediction)	経済モデル, 株価チャート, 天気図
コミュニケーション	言語, アイコン
構成・構築	設計図, UMLの図
推論	論理(命題, 述語), ルール
誘導(Navigation)	図書インデックス, 地図, 標識
感情表現	楽譜

モデリングとは
モデリングの対象: システムとコンテキスト

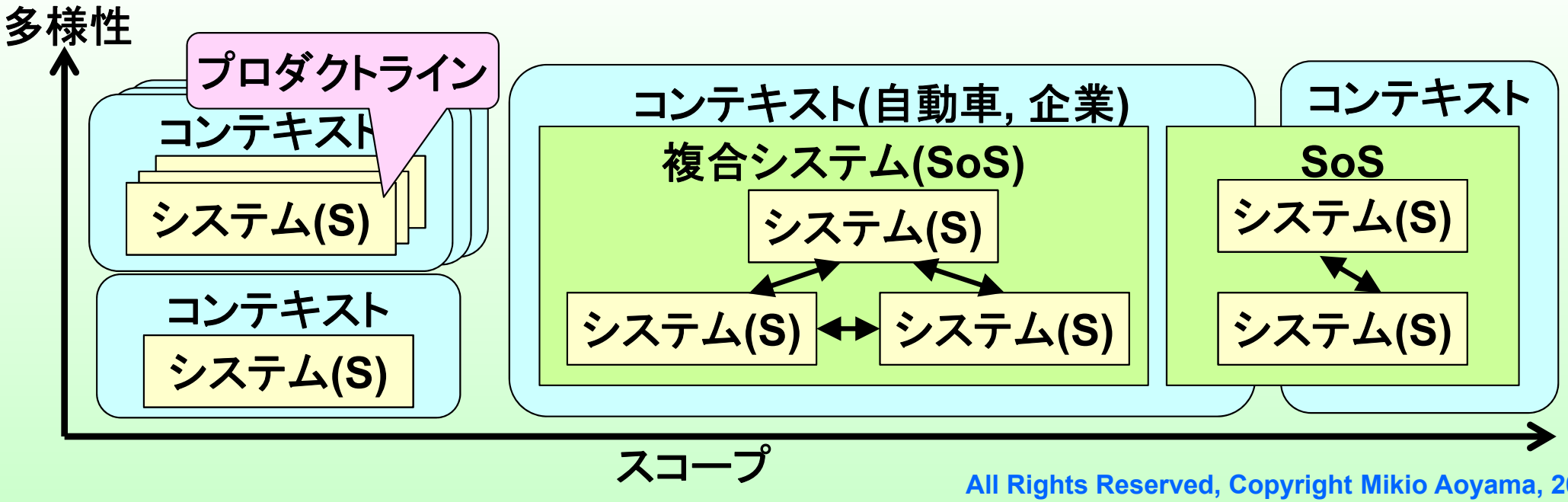
- 原則: システムは環境に依存する
- システム境界: システムとして定義，設計する範囲を規定
- コンテキスト: 環境の中で，システム定義に影響する範囲
  - システムコンテキストとも呼ぶ
  - ユーザ(ドライバ)，路面，交通状態，関連法規，など

- システム/コンテキスト境界の定義: システム開発の基礎
  - 境界領域: システム/コンテキストの境界は明確な分離が難しい



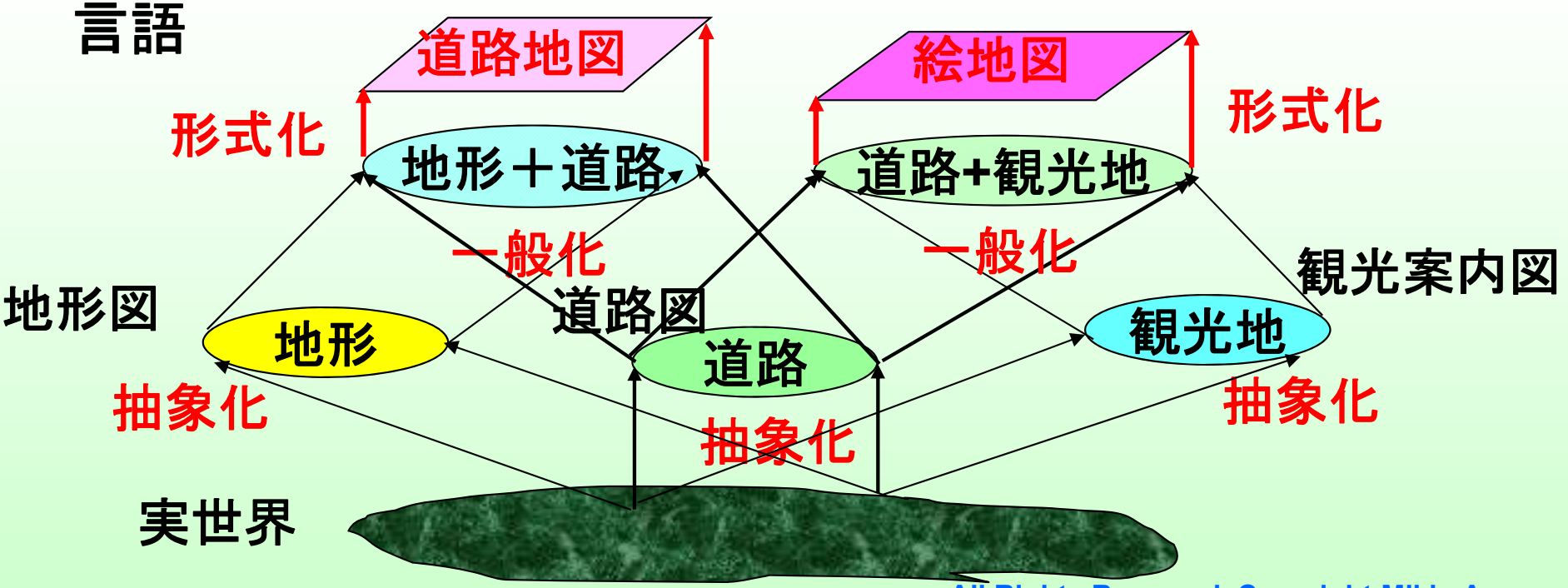
モデリングとは
モデリングの対象: システムとコンテキスト

- 複合システム(SoS: Systems-of-Systems)
  - 複数のシステムの複合体
  - Systems of systems are large-scale concurrent and distributed systems that are comprised of complex systems
- プロダクトライン(Product Line)[製品系列]
  - 製品戦略に基づき主要な特性を共有する一連の製品群
  - 市場, 顧客の多様性への対応



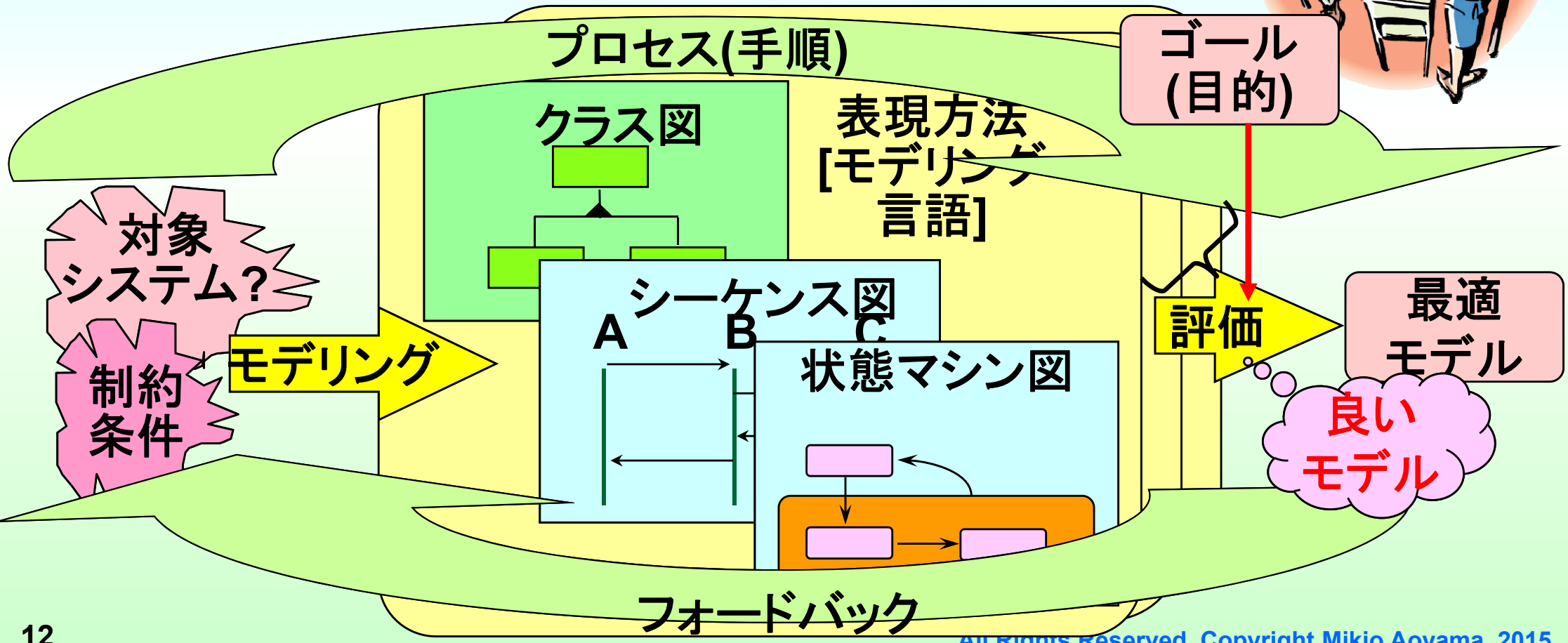
モデリングとは
モデリングの方法と方法論: モデリングの一般的方法

- モデリング=抽象化＋一般化＋形式化
- 抽象化(Abstraction): 本質(必要な情報)の抽出
- 一般化(Generalization): 共通部分の抽出
  - 共通部と個別部の分離
- 形式化(Formalization): 共通認識の抽出と表現
  - モデリング言語(Modeling Language): モデルの形式的表現言語



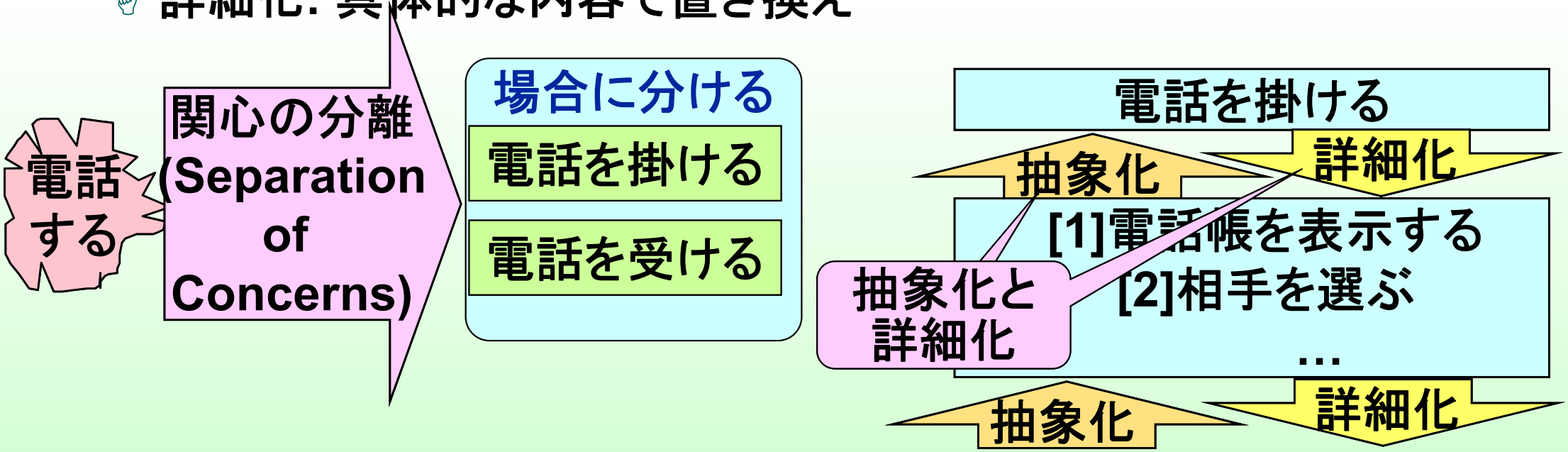
モデリングとは
モデリングの方法と方法論: モデリング方法論

- モデリング方法論: プロセス(手順)+表現+検証評価
  - 手順: 反復型プロセス(フィードバックプロセス)
  - 表現: モデリング言語, ドメイン固有言語(DSL)
  - 検証評価:モデルの正しさの検証と良さの評価



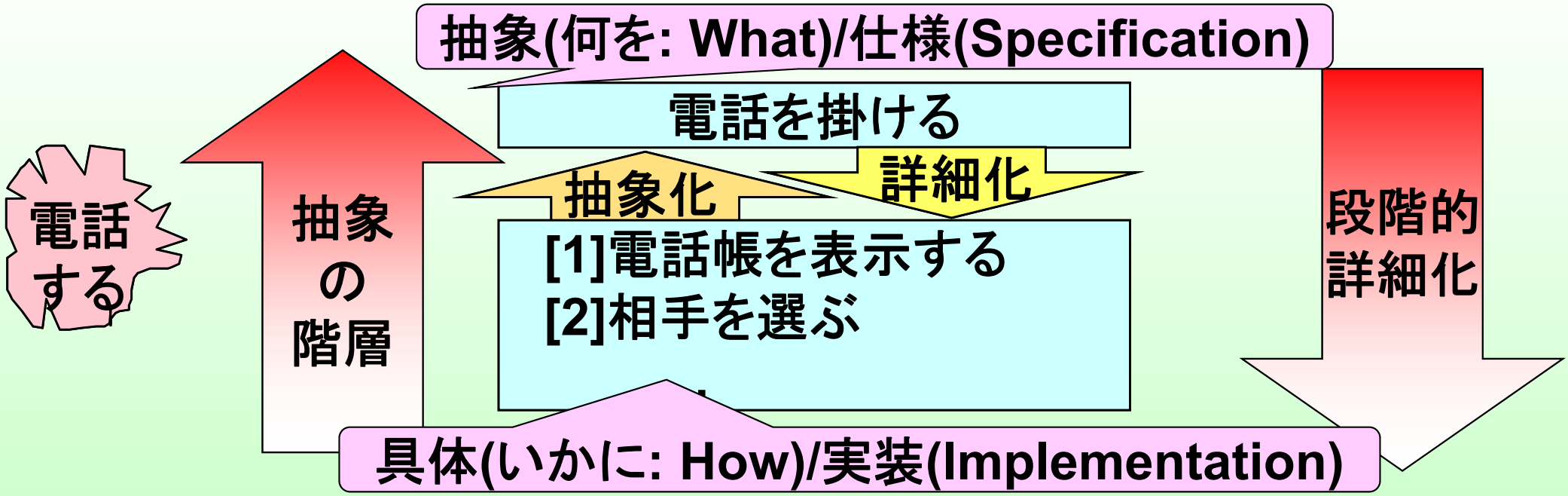
モデリングとは
モデリングの方法: 関心事の分割と抽象化/詳細化

- 分割統治(Divide and Conquer): 分けて考える
- 関心の分割[Separation of Concerns]
  - 問題を分けて単純化：複雑度の軽減
- 抽象化（Abstraction)と詳細化(Refinement):
  - 抽象化: 詳細な内容の隠ぺい
  - 詳細化: 具体的な内容で置き換え



モデリングとは
モデリングの方法: 抽象化とは

- 抽象化: 問題の本質のみを抽出[本質的でないところは捨象]
  - 目的: 複雑度の軽減(人間が扱える)
  - 仕様(What)と実装(How)
- 抽象化の方法: 階層的抽象化(Hierarchical Abstraction)
  - 抽象の階層(Levels of Abstraction)
  - 繰返し抽象化することにより階層構造を成す



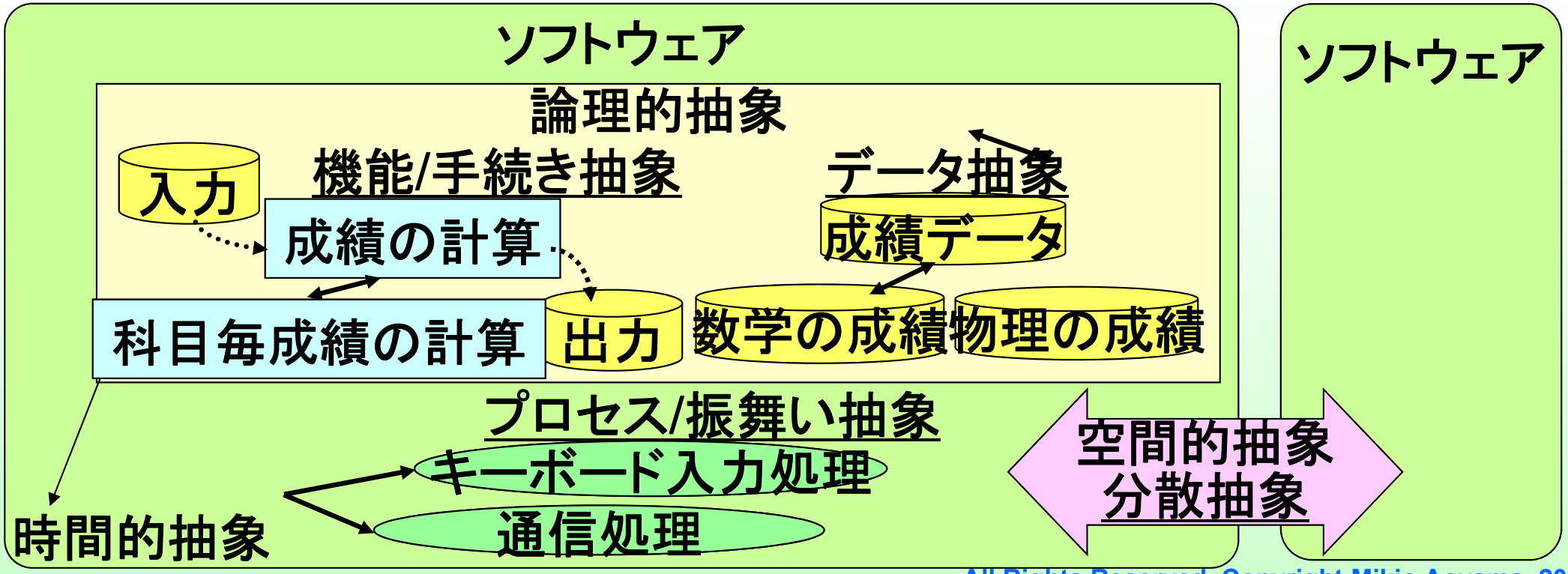
モデリングとは
モデリングの方法: トップダウンとボトムアップ

☞抽象化と詳細化が導くモデリング方法

	トップダウン(Top Down)	ボトムアップ(Bottom Up)
方法	(抽象/全体)から下(具象/部分)へ[段階的詳細化]	(具象/部分)から上(抽象/全体)へ[部品組み合わせ]
長所	全体像を知る, モデル化の初期には詳細が分からない	具体的な要素を早期に知る, 直ちに実現へ
短所	具体的要素を抽出できるまで時間がかかる, 進捗(どこがボトムか)が不明	ビューが垂直的で狭くシステム横断的性質が不明, 変更による手戻り

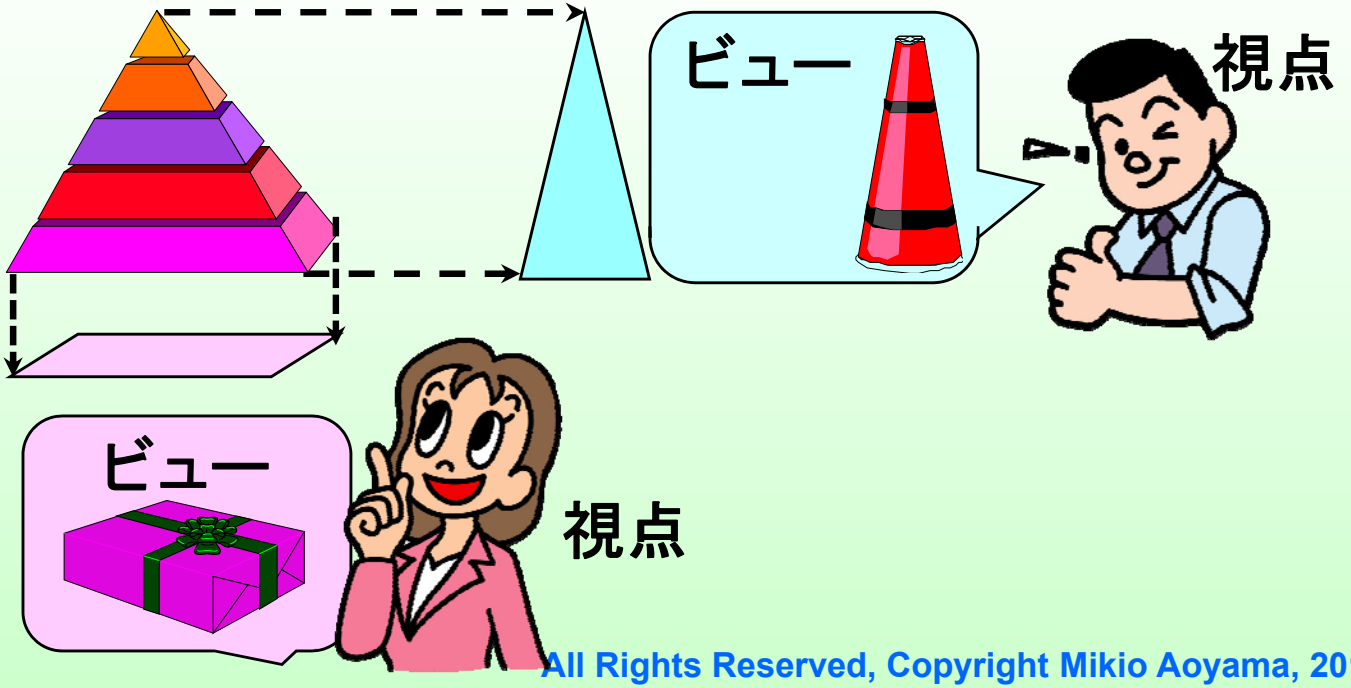
モデリングとは
モデリングの方法: ソフトウェアにおける3つの抽象化の軸

- 手続き抽象(Procedural Abstraction)/機能抽象(Functional Abstraction): ソフトウェアが果たすべき機能(=手続き)に着目
- データ抽象(Data Abstraction): 処理対象であるデータに着目
- プロセス抽象(Process Abstraction)/振舞い抽象(Behavior Abstraction): 実行のタイミングや並行性に着目



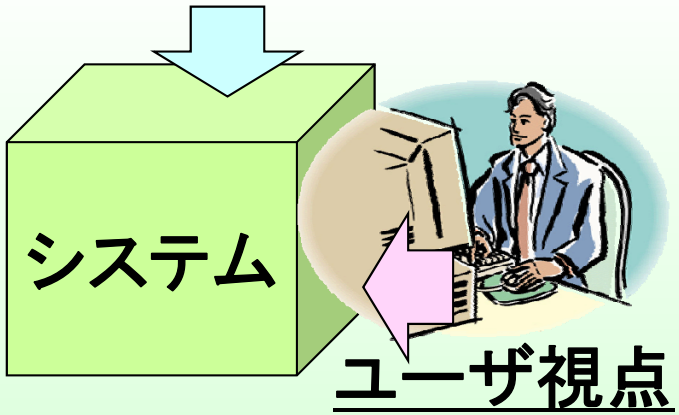
モデリングとは
モデルの表現: 複数視点のモデリング

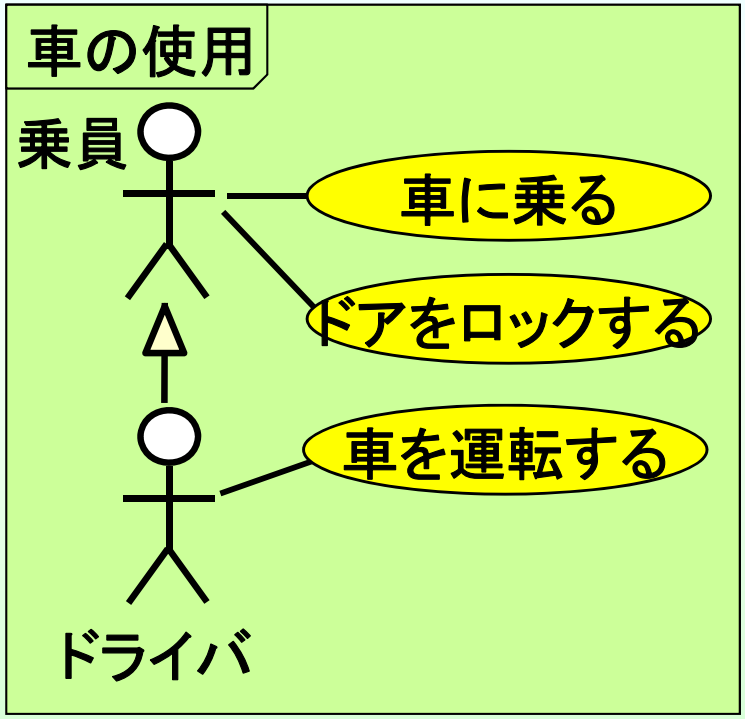
- 視点(View Point)とビュー(View)
  - 視点: システムを見る関心
  - ビュー視点から見た見え方(眼に映った情景)
- 複数視点によるモデル化＝複数視点に分けてモデル化
  - システムを正しく理解
  - 複雑度の軽減: 3次元立体⇒2次元図面×3



モデリングとは
モデルの表現: 俯瞰的視点とユーザ視点

- 俯瞰的視点(Holistic Viewpoint): システムの全体を俯瞰して捉える
  - 例: Zachmanフレームワーク, ユースケース図
- ユーザ視点(User Viewpoint): ユーザの視点から対象を捉える
  - 例: ユースケースのシナリオ, ユーザストーリ





「車を運転する」シナリオ

事前条件

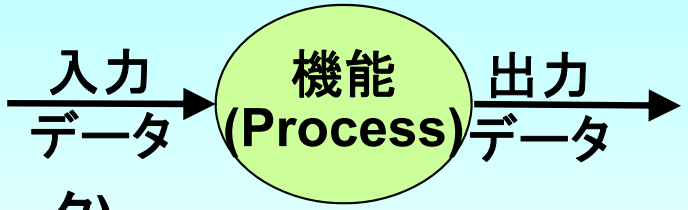
1) エンジンが停止している
2) ドアがロックされている
3) ギアがニュートラルになっている

主シナリオ

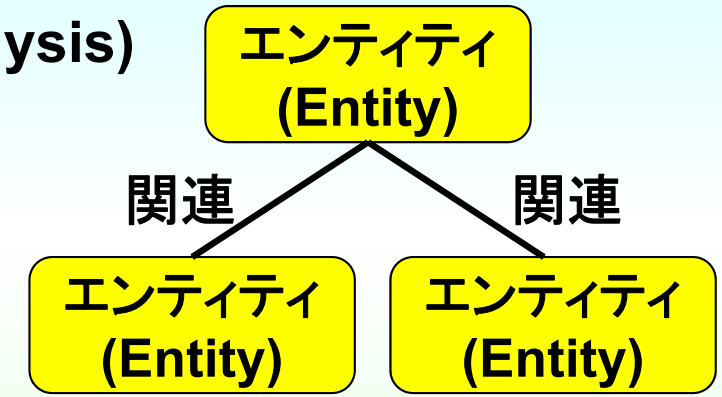
1) エンジンを掛ける
2) ギアをDに入れる
3) パーキングブレーキを解除する
・・・・

モデリングとは
モデルの表現: 複数視点によるモデルの表現

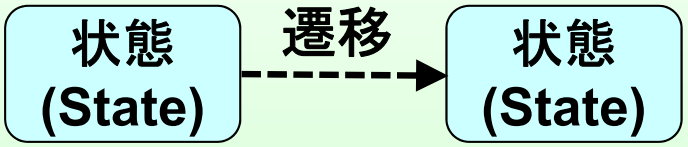
- 機能の視点に基づく分析方法[詳細後述]
  - 視点: 機能=業務/処理(Process)
  - モデル: データーフロー(入力データ・処理・出力データ)
  - 分析方法: データーフロー分析(Data Flow Analysis)



- 構造の視点に基づく分析方法
  - 視点: エンティティ(Entity)=データ
  - モデル: ER(Entity-Relationship: 実体・関連)
  - 分析方法: 概念データモデリング

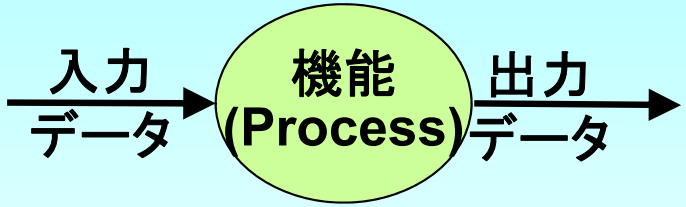


- 挙動(振舞い)の視点に基づく分析方法
  - 視点: 状態(State)
  - モデル: 状態遷移(State Transition)
  - 分析方法: 状態分析

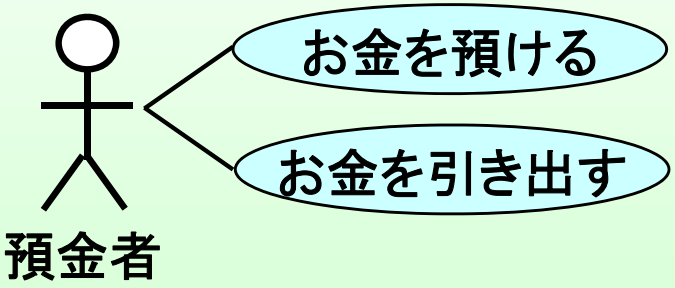


モデリングとは
モデルの表現: 機能の表現方法

- データーフローモデル(処理中心)
  - モデル: IPO(入力データ・処理・出力データ)
  - 分析方法: データーフロー分析(Data Flow Analysis)
  - 表現: データフロー図(DFD: Data Flow Diagram)



- ビジネスプロセス/ワークフローモデル(作業中心)
  - モデル: ビジネスプロセス, ワークフロー(作業(Work)とその入出力)
  - 分析方法: ビジネス分析(業務分析), ワークフロー分析
  - 表現: アクティビティ図, BPMN(Business Process Modeling and Notation)
- インターラクションモデル(ユーザ視点)
  - モデル: ユースケース(Use Case)
  - 分析方法: ユースケース分析
  - 表現: ユースケース図, ユースケースのシナリオ



モデリングとは
複数視点によるモデリング: 3つの視点による俯瞰的モデル化

- 複数の視点：関心事の分離によるモデルの複雑度の軽減
  - 機能(Function): システムの果たす機能
  - 挙動(Behavior): システムの振舞い
  - 構造(Structure): システムの構成

システムの果たす機能
とその処理の流れ（何を）

機能の視点
（データフロー図）

顧客マスター

顧客 → カード → 1.0 顧客認証 →

妥当でない ↓

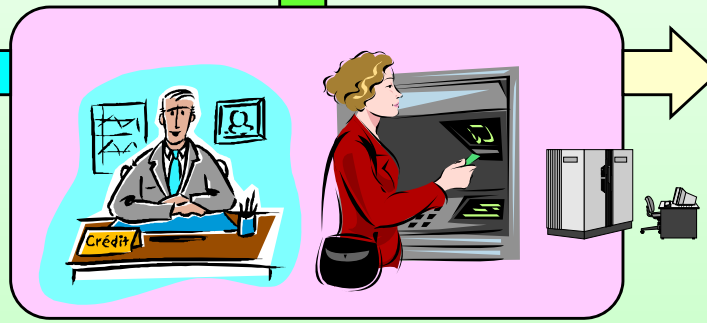
5.0 認証誤り処理

構造の視点（クラス図）

銀行端末システム

窓口端末　ATM

システムの構成要素と
その関係（誰が・何が）



システムの実行順序と
そのタイミング（いつ）

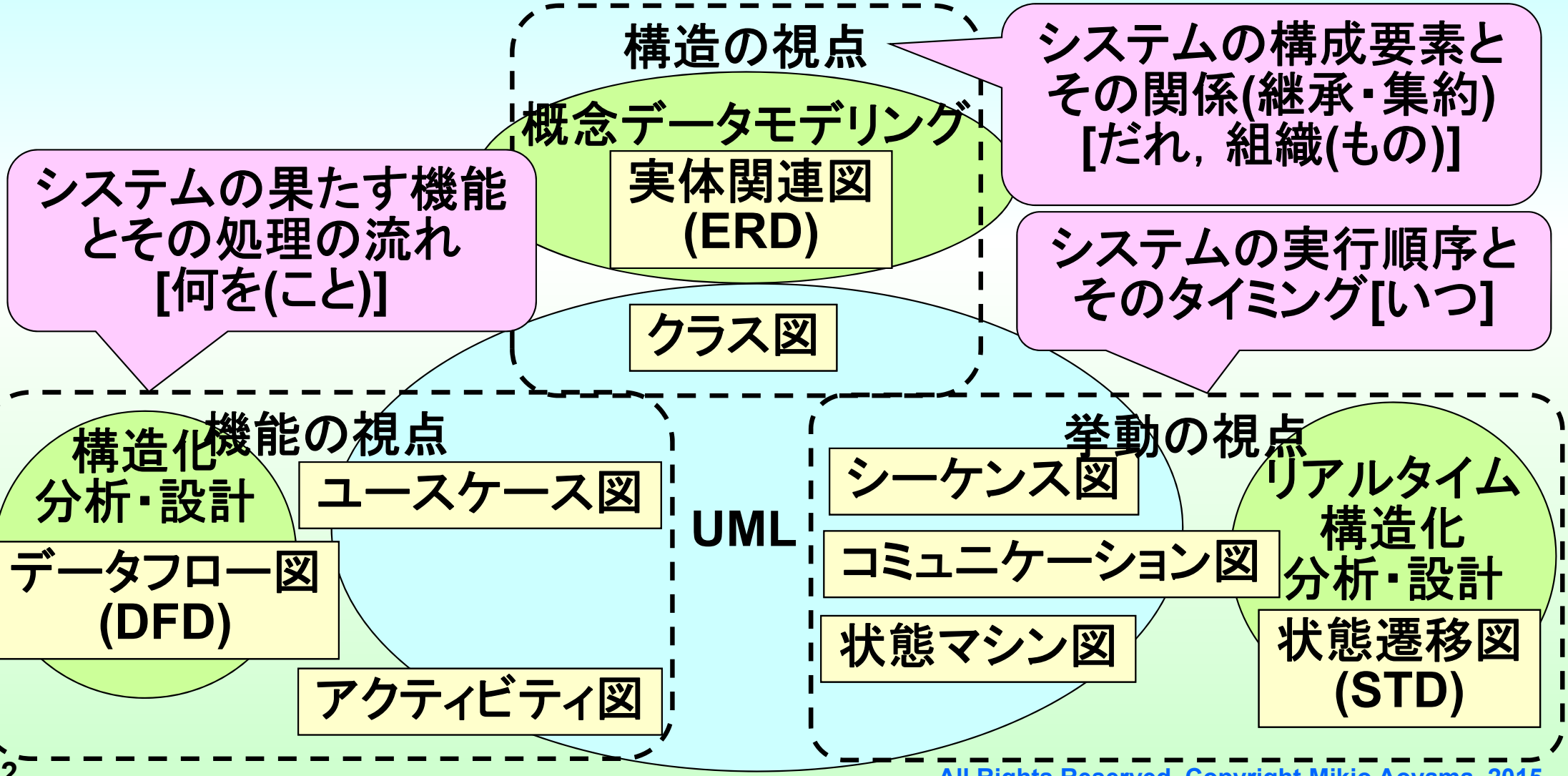
挙動の視点
（状態遷移図）

受付

認証中

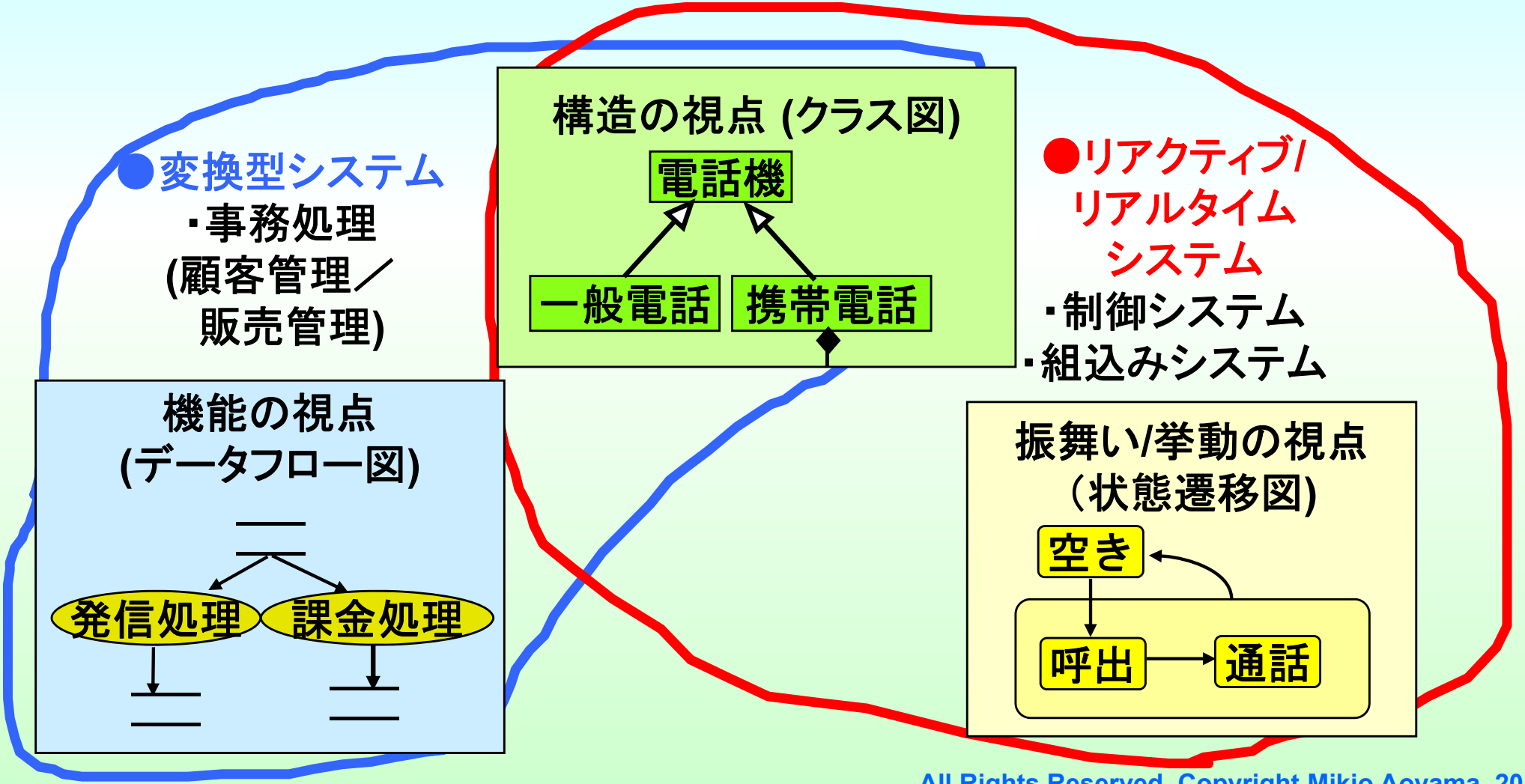
モデリングとは
複数視点によるモデリング: 複数視点とモデリング言語

- 3視点に対応する表現方法: UML, DFD, ERD, STD
- UMLから派生したドメイン固有モデリング言語: SysML, AADL



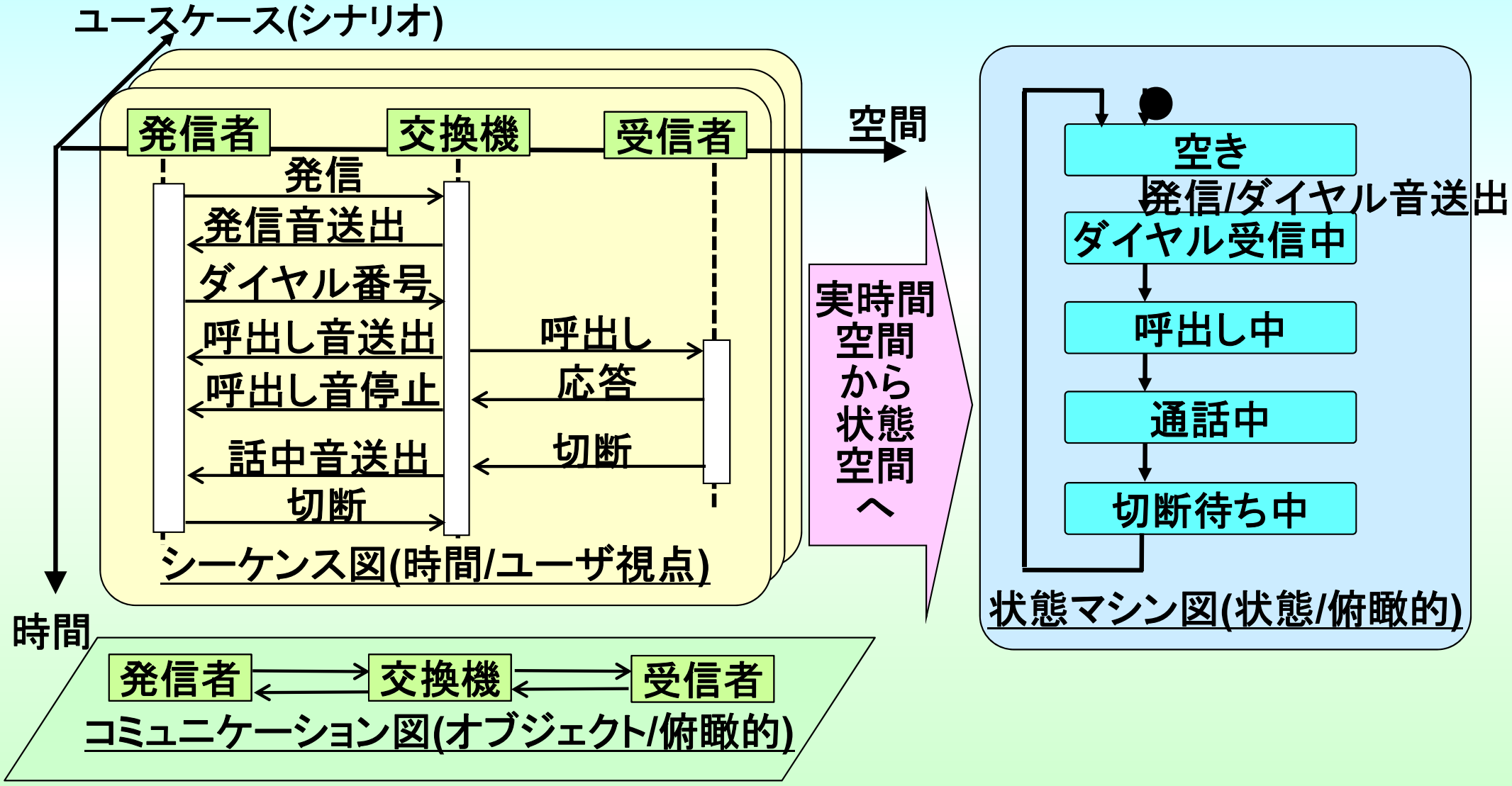
モデリングとは
複数視点によるモデリング: 複数視点と対象ドメイン

☞ 3視点は対象システムの特性により重みが異なる



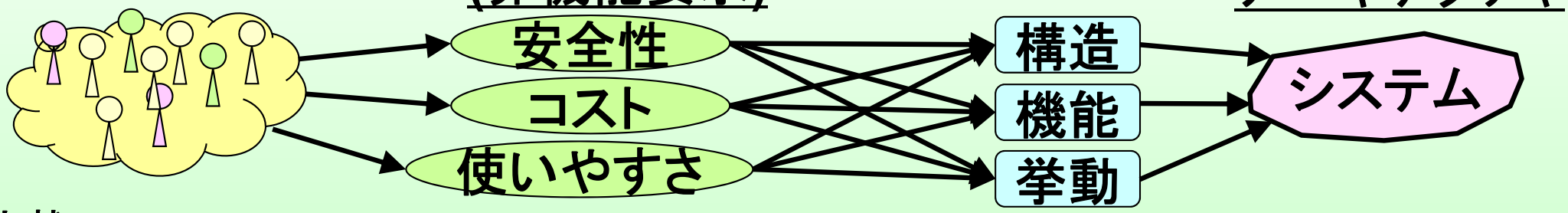
モデリングとは
複数視点によるモデリング: UMLの挙動モデル

☞ 振舞いモデルの詳細化: 3つの視点



モデリングとは
複数視点によるモデリング: 視点と関心事

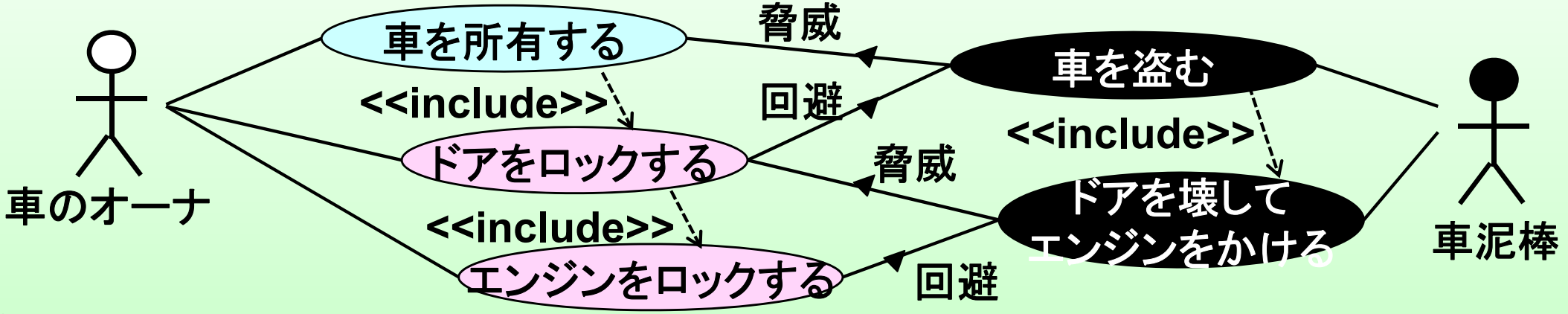
- 視点の選択と設定
  - 視点＝システムへの関心事の集約(少ない視点で適切に見える)
- 関心事=視点
  - ユーザの関心事: ステークホルダの視点からの要求の関心事
  - システムの視点: 開発者の視点からのアーキテクチャの関心事
- 非機能要求(品質要求)とアーキテクチャ
  - 非機能要求(品質要求)がアーキテクチャを決定



参考文献: P. Lago, et al., Software Architecture: Framing Stakeholders' Concerns, IEEE Software, Vol. 27, No. 6, Nov./Dec., 2010, pp. 20-24.

モデリングとは
あってはならない機能のモデリング: ミスユースケース分析

- セキュリティ問題の多くは業務(ビジネスプロセス/運用)の問題
- ミスユースケース(Misuse Case)
  - システムに危害を加えるユースケース
  - 悪意のアクタ，誤った操作を行うアクタ
- ミスユースケース分析の目的
  - 起こってはいけない要求の定義: セキュリティ要求，安全性要求
  - ユースケースに対する脅威とその回避で表現
  - 回避手段＝セキュリティ要求



モデリングとは
メタモデル:「メタ」ということ: 質問とメタ質問

- 質問
  - プロセスに関する質問
    - クライアントは誰か
    - ソリューションの価値は何か？
  - プロダクトに関する質問
    - このシステムはどんな問題を解決するか？
    - この計算の必要な精度は？
- メタ質問
  - 質問の数は適切か?
  - 質問は的を得ているか？
  - この質問に答える適任者ですか？

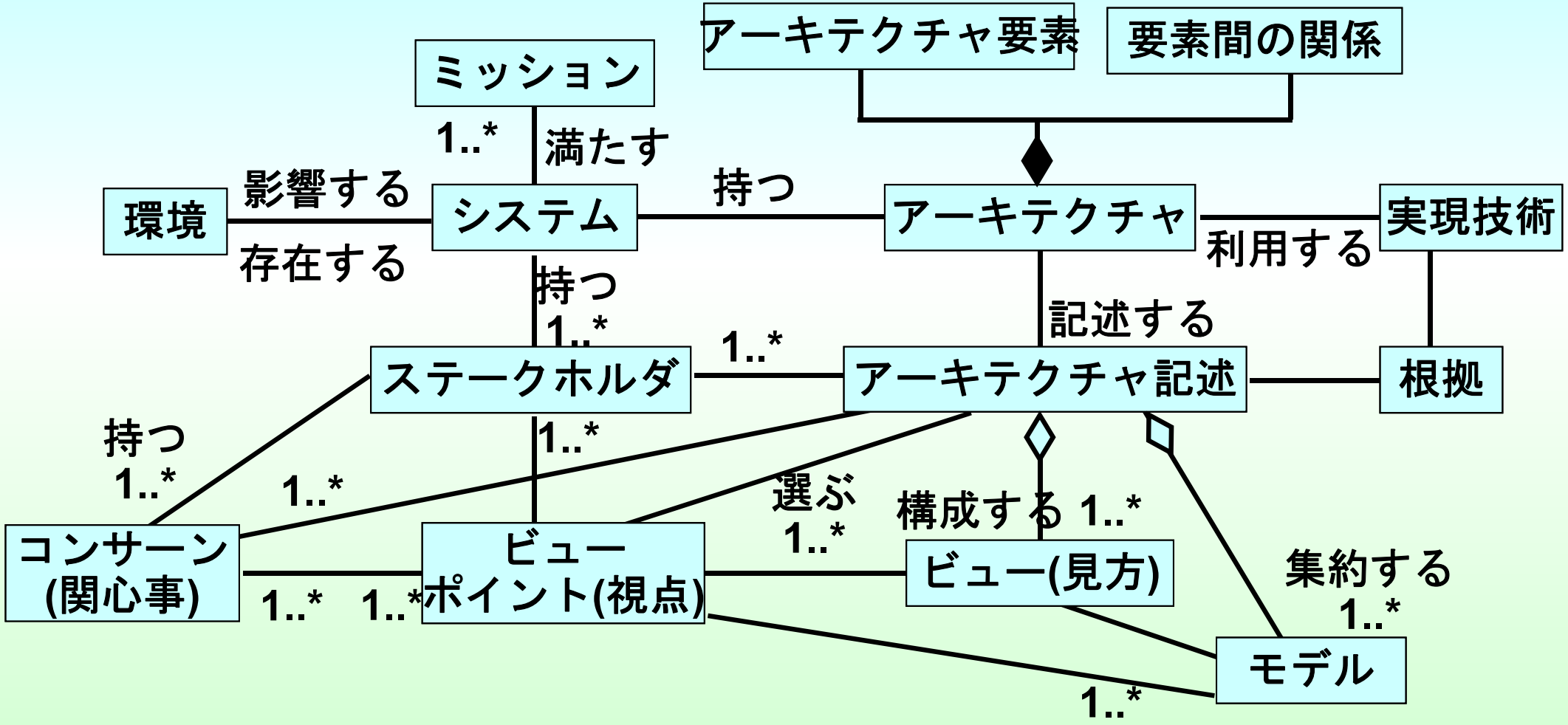
meta=beyond
ある概念に対し，1段抽象化された概念
例: meta-analysis(メタ解析: 解析結果群を解析し，より高次の解析を行う)

meta=about
ある概念を説明する概念: 例: meta-data

参考文献: D. C. Gause and G. Weinberg, Exploring Requirements: Quality before Design, Dorset House, 1989 [柳川 志津子(訳), 要求仕様の探検学，共立出版，1993].

モデリングとは
メタモデル: ISO/IEC 42010のアーキテクチャメタモデル

☞アーキテクチャの決定要因とその間の関係



参考文献: ISO/IEC 42010, Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems, 2007 [IEEE Std. 1471-2000].

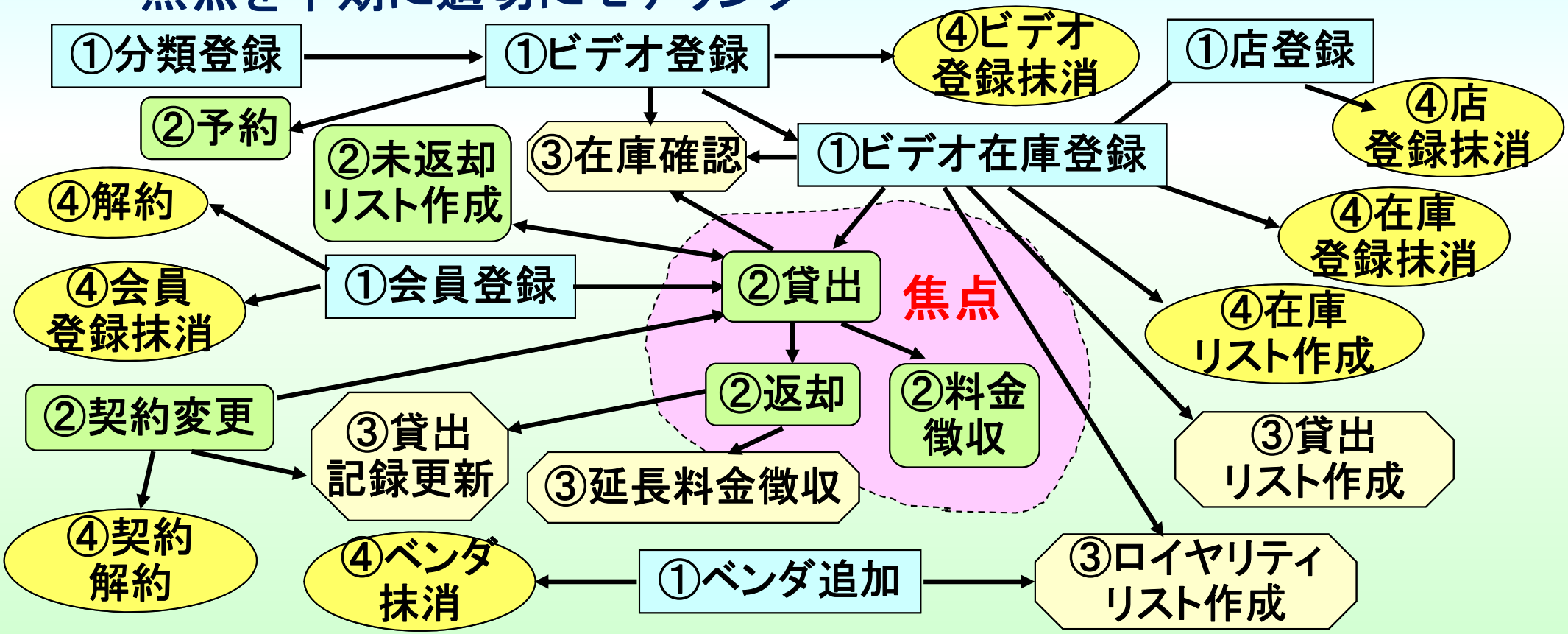
モデリングとは
モデルの正しさと良さの評価

- 表現の正しさ: 正当性(Verified, Well-Defined, Well-Formed)
  - 完全性(Completeness): 抜け，漏れがないこと
  - 一貫性(Consistency)[無矛盾性]: 矛盾や相反がないこと
  - 無曖昧性(Unambiguity): 曖昧な表現がないこと
  - 検証可能性(Verifiable): 検証可能であること
  - 追跡可能性[トレーサビリティ: Traceability): モデルの源泉，あるいは，詳細との対応付けが可能であること
- 内容の正しさ: 妥当性(Validated)
  - モデルが対象の内容を表現できていること
- モデルの良さの評価
  - 簡潔性(Simplicity): モデルに冗長や無駄がないこと
  - 理解容易性(Readability): モデルが分かりやすいこと
  - スケーラビリティ(Scalability): 規模の増大への対応可能性

モデリングのセンス?

モデリングとは
モデルの良さ: モデルの焦点

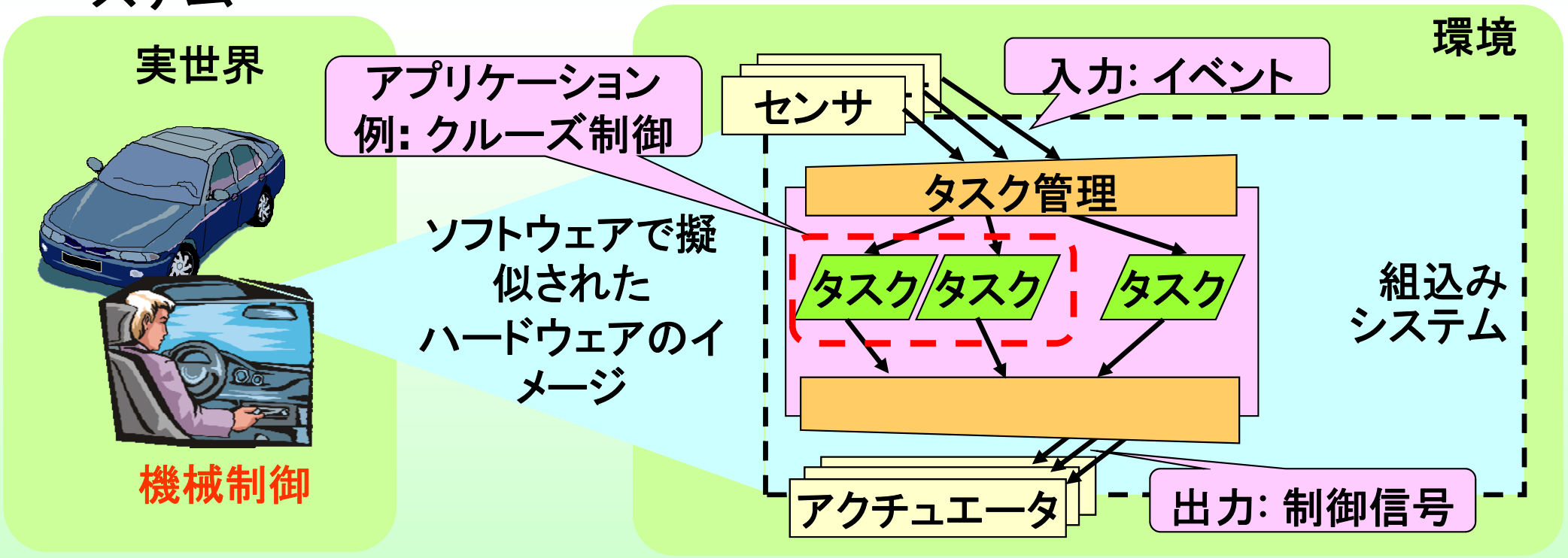
- モデルには焦点がある
  - 焦点: モデルの意味の中心となる重要な部分
  - 例:ビデオレンタルシステムの機能関連図
- 焦点を早期に適切にモデリング



参考文献: T. Rowlett, The Object-Oriented Development Process, Prentice Hall PTR, 2001.

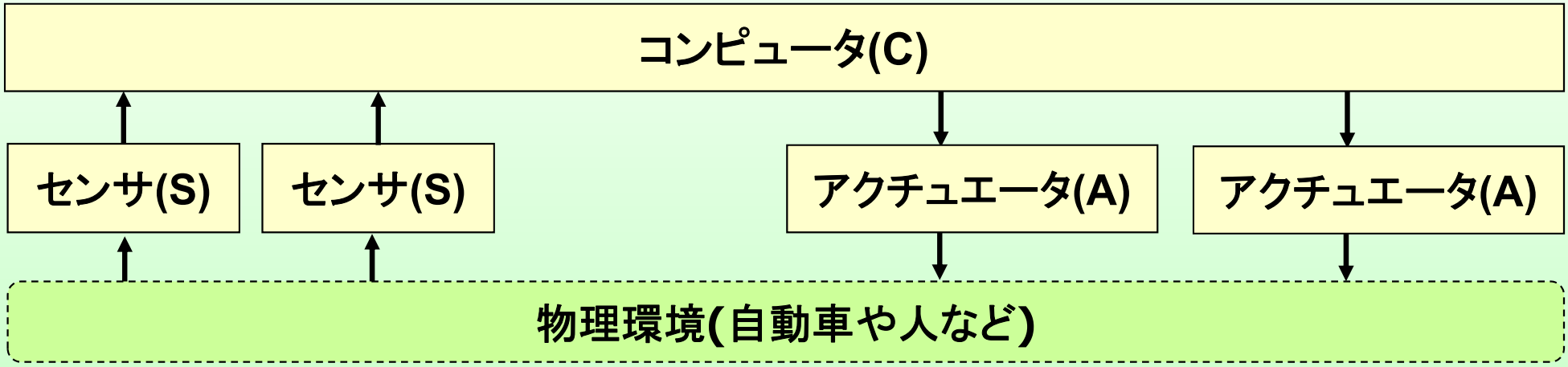
組込みシステムのモデリング
組込みシステムの本質的特性: 挙動

- 組込みシステム/ソフトウェア=実世界の擬似
  - 本質的特性=挙動
- 挙動の制御: 自律制御と応答制御
  - 自律制御: 目標値に追随:フィードバック制御, 連続時間システム
  - 応答制御: 外部からの操作などに反応: イベント制御, 離散時間システム



組込みシステムのモデリング
組込みシステムの本質的特性: SACアーキテクチャパターン

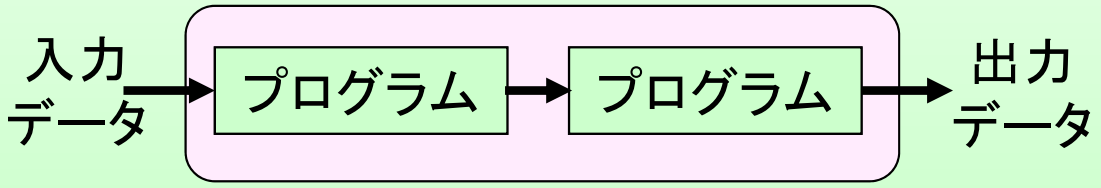
- SAC(Sensor Actuator Controller) パターン
- 物理環境を通した非明示的(Implicit)フィードバックシステム
- コンピュータ: 物理世界の疑似と制御機能
  - 処理のタイミング
  - 時間駆動(Time-Driven/Triggered): コンピュータは一定周期でセンサから取り込み，処理を起動
  - イベント駆動(Event-Driven/Triggered): コンピュータは任意のタイミングでセンサが検出したイベントに応じて処理を起動

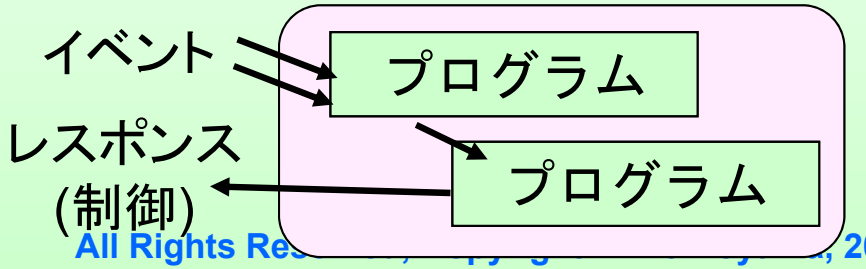


組込みシステムのモデリング
組込みシステムと情報システム

☞ 反応(リアクティブ)型と変換型

分 類	変換型(Transformational) [エンタープライズアーキテクチャ]	反応型(Reactive) [組込みアーキテクチャ]
意 味	ソフトウェアは入力データを変換する	ソフトウェアは外部からの操作や信号に反応して何らかの制御を行う
例	ビジネスアプリケーション	交換機，航空管制，自動車など
ソフトウェアの特徴	・必要な時のみ実行 ・処理効率：処理データ量(スループット) ・計算結果の信頼性	・常時実行(連続性) ・リアルタイム処理: 要求時間内に処理を完了(応答時間) ・常時実行できる信頼性・安全性
設計の視点	機能/データ[データフロー]	挙動/制御[制御フロー，状態遷移]



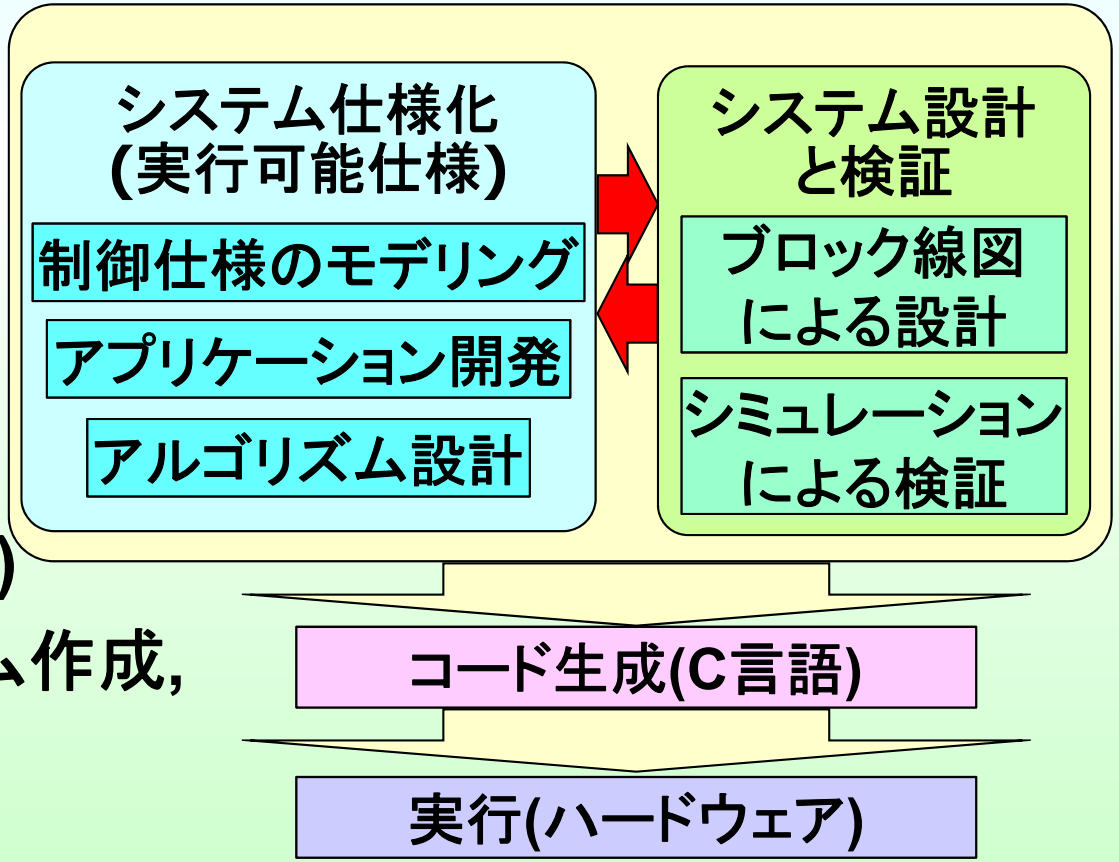


組込みシステムのモデリング
モデル駆動とモデルベース: 離散時間モデルと連続時間

- モデル駆動: 離散時間モデル
  - モデル: (離散)状態遷移マシン
  - 表現: 状態遷移図(UML 状態マシン図)
  - モデル駆動開発(MDA/MDD: Model-Driven Architecture/Development)
- モデルベース: 連続時間(制御)モデル
  - 自動車のエンジン制御, ブレーキ制御
  - モデル:微分方程式⇒行列により定義されるベクトル空間における状態遷移
  - 表現: 数式, ブロック線図
  - モデルベース開発(MBD: Model-Based Development)

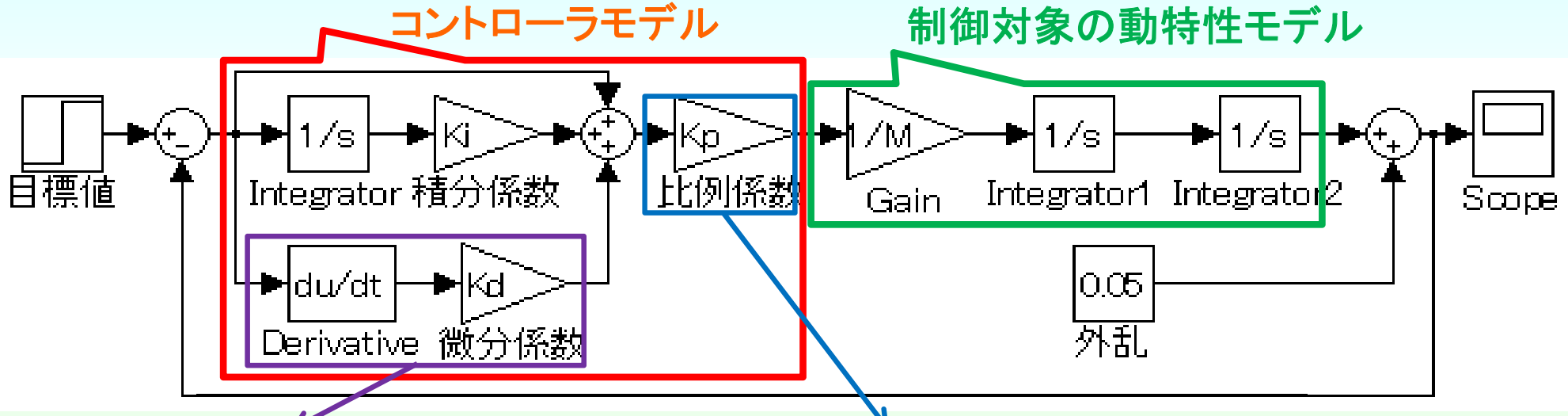
組込みシステムのモデリング
モデルベース開発のモデリング: モデルの意義

- モデル＝制御の実行可能仕様(Executable Specification)
  - (フィードバック)制御システム=連続時間システム
    ⇒微分方程式⇒状態ベクトル空間モデル⇒行列のベクトル演算
  - 例: 自動車のエンジン制御, ブレーキ制御
- モデルベース開発
  - モデルによるシステム設計
  - モデルレベルのシステム検証
  - モデルからコード生成
- モデルベース開発のツール例
  - MATLAB: 数式演算(行列など)
  - Simulink: ブロックダイアグラム作成, シミュレーション

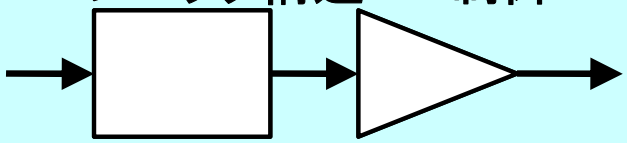


組込みシステムのモデリング
モデルベース開発のモデリング: ブロック線図によるモデリング

- MATLAB/Simulinkによる制御モデルの作成とシミュレーション
  - モデリングの対象: 制御対象とコントローラ
  - 表現:ブロック線図(ブロック構造とパラメータで構成)



ブロック構造: PI制御



➢ブロックの組み合わせ方
➢制御の内容を決定

パラメータ

➢ブロックで定義されている値
➢ブロック内での計算を決定

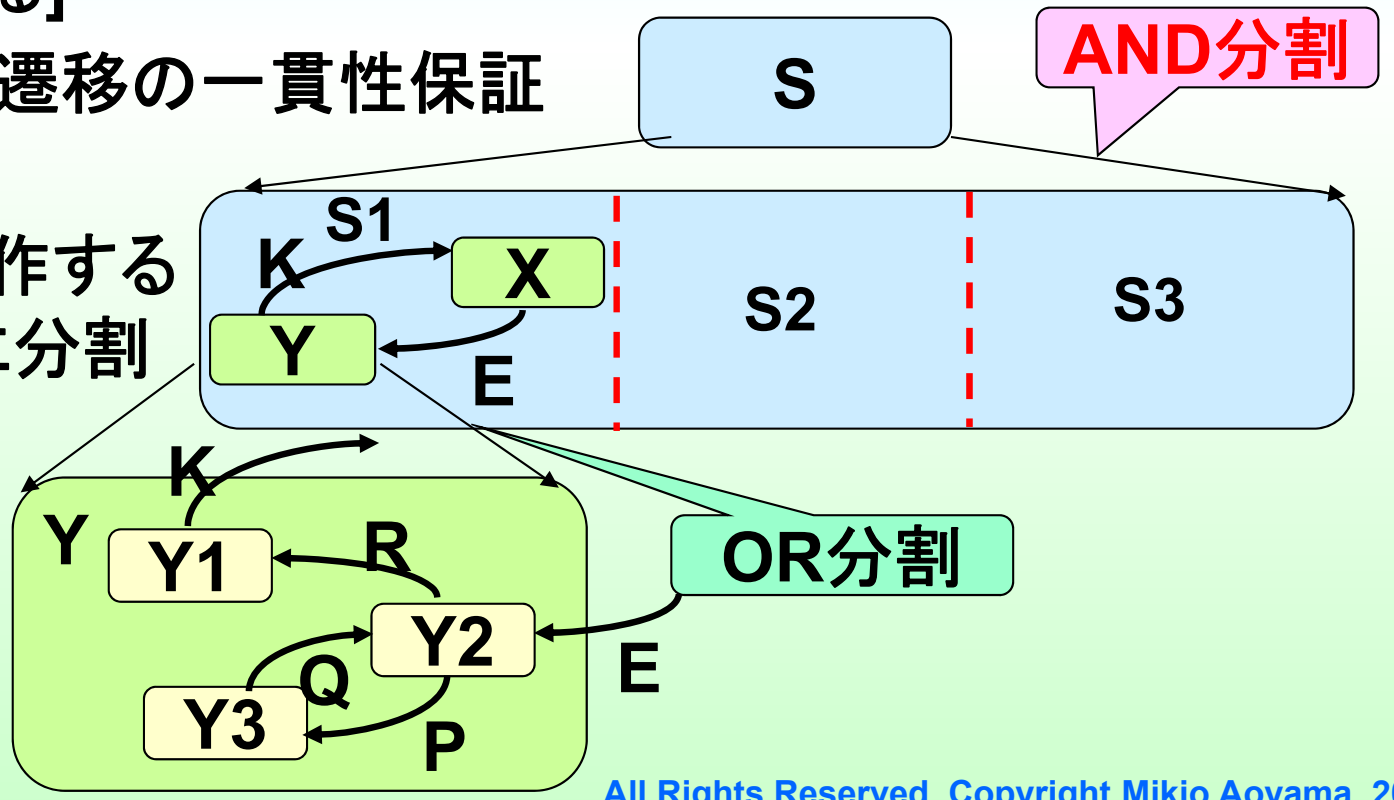
組込みシステムのモデリング
状態マシン図による挙動のモデリング

☞状態の階層構造による振舞いの階層構造化

- 状態の段階的な階層構造化: AND分割とOR分割
- AND分割: S=S1 ｜ S2 ｜S3[S1, S2, S3の状態に同時並行にある]
- OR分割(状態の入れ子構造): Y= Y1 or Y2 or Y3 [Y1, Y2, Y3のいずれかの状態にある]
- 階層間でイベントと遷移の一貫性保証

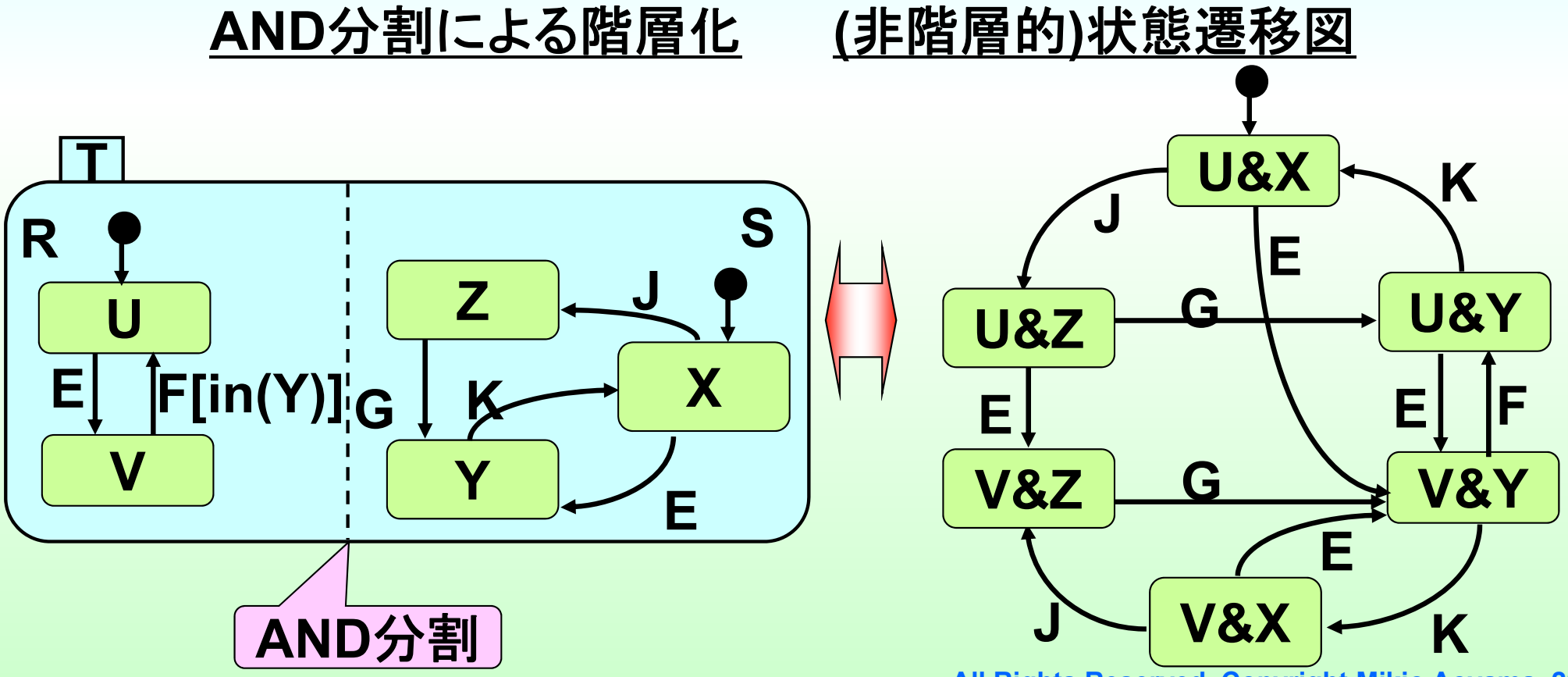
3つの並行に動作するサブシステムに分割

状態Yの振舞いをサブ状態遷移として詳細化



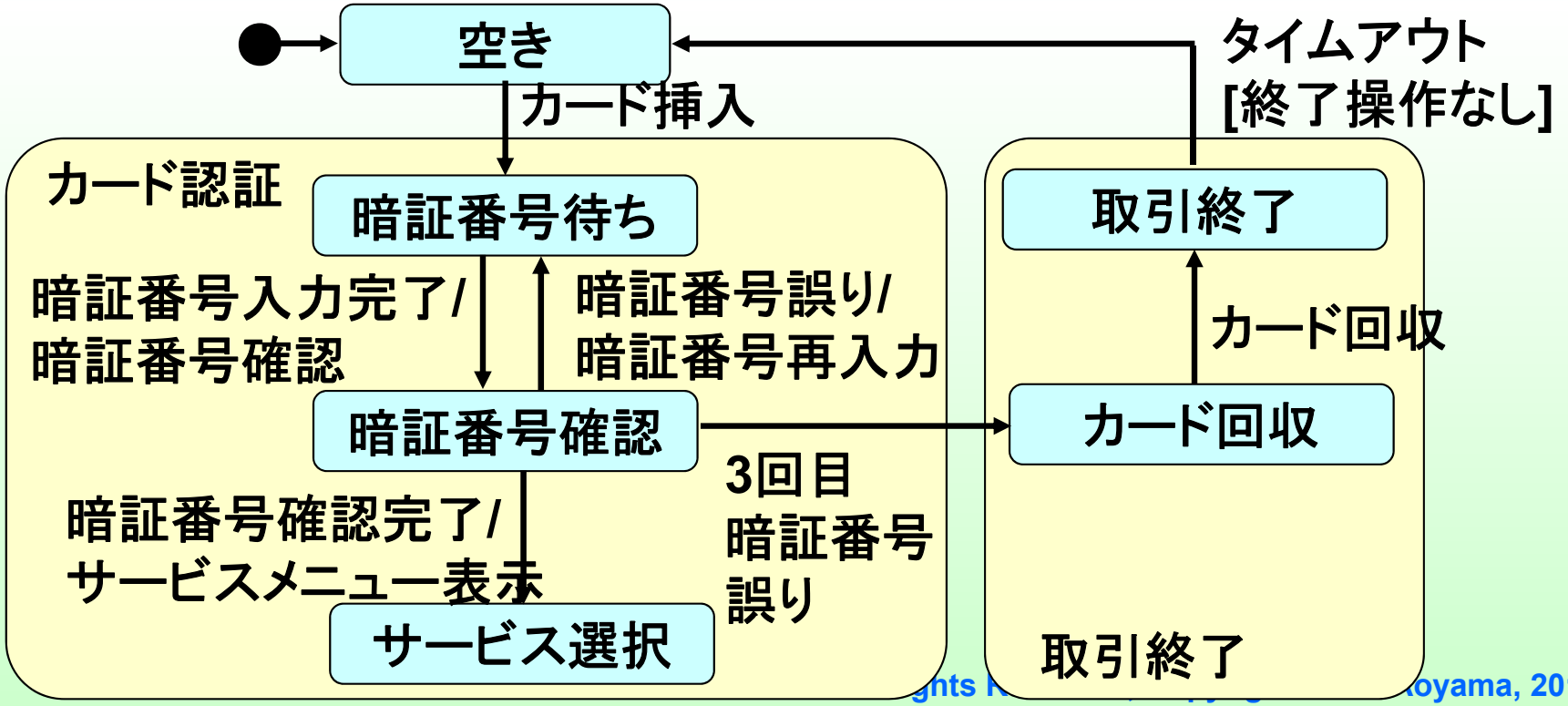
組込みシステムのモデリング
状態マシン図による挙動モデルの階層構造化: AND分割

- AND分割(AND Decomposition): 分割された状態は独立に並行実行可能
- 分割された状態をAND状態あるいは直交領域(Orthogonal Region)と呼ぶ
  - AND分割は破線で表す
- AND分割は状態を統合した(非階層的/古典的)状態遷移図に展開可能

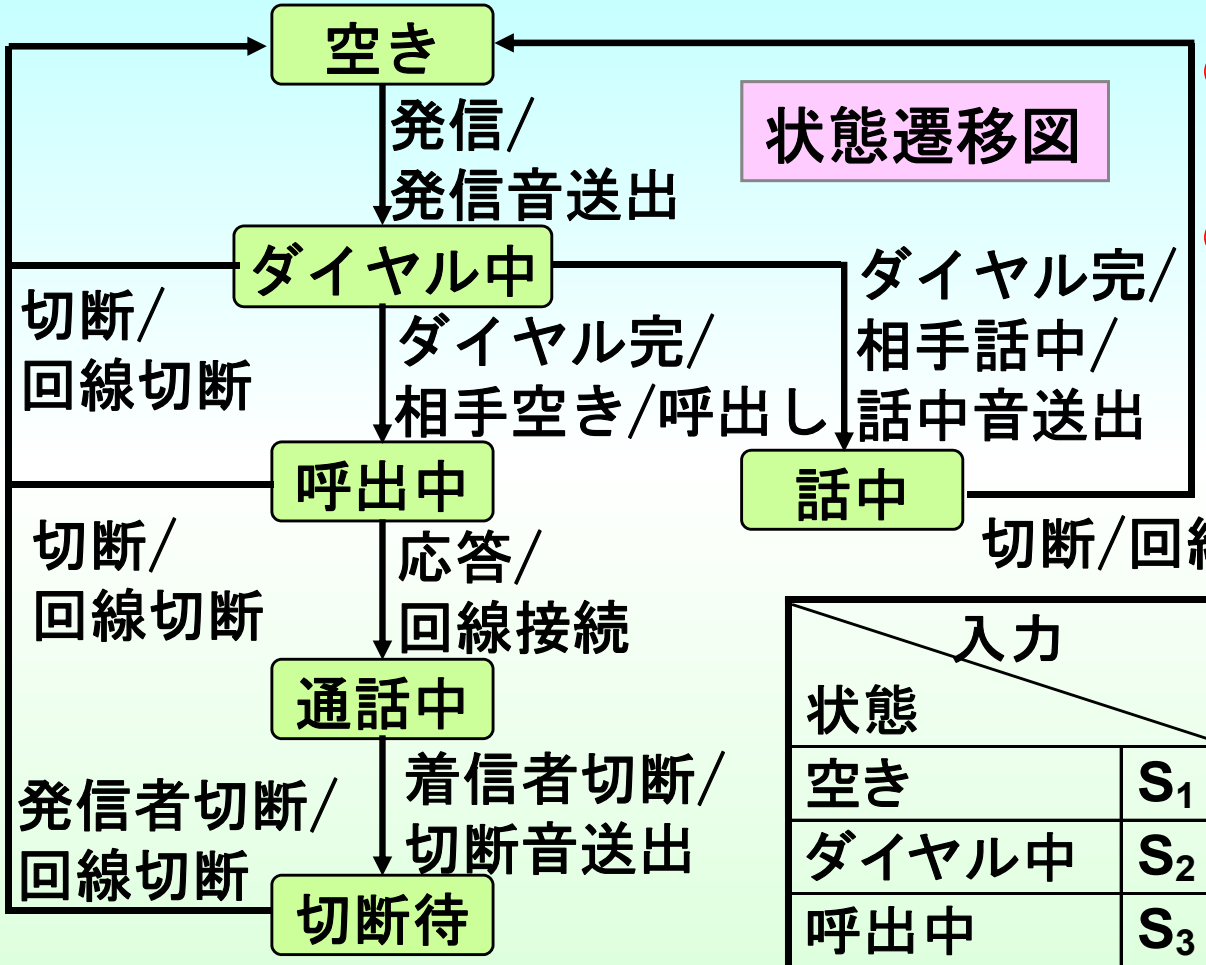


組込みシステムのモデリング
状態マシン図による挙動モデルの階層構造化: OR分割

- OR分割: 状態の包含(Composition)関係は集合の包含関係を表す
  - スーパー状態: 外側から包含する上位状態
  - サブ状態: 内側に包含される下位状態
- 状態は下位状態の何れか一つにある: 下位状態は相互排他
  - 例:「カード認証」の状態は,「暗証番号待ち」,「暗証番号確認」,「サービス選択」のいずれかにある[同時にはなれない]



組込みシステムのモデリング
状態遷移図と状態遷移表



状態遷移図

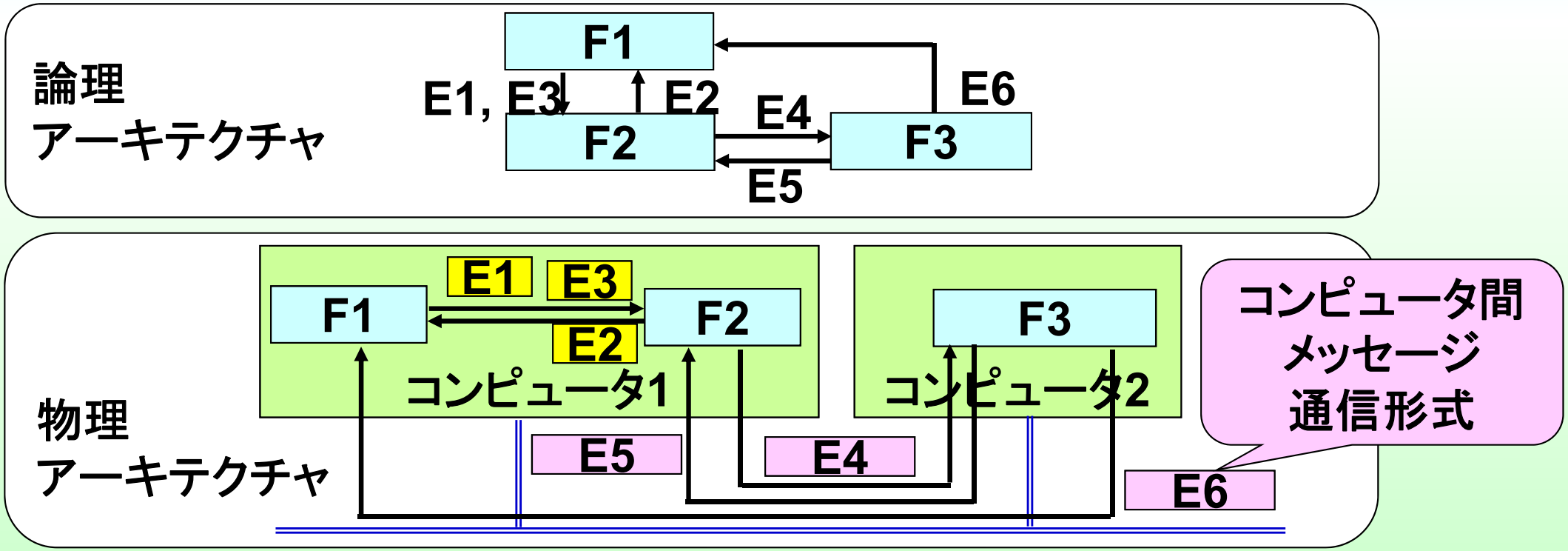
- 状態遷移図と状態遷移表は等価
- 状態遷移表からテーブル駆動(Table-Driven)プログラミングが可能

状態遷移表

状態＼入力		発信	切断	相手空き	相手話中	応答
空き	S_1	S_2				
ダイヤル中	S_2		S_1	S_3	S_4	
呼出中	S_3		S_1			S_5
話中	S_4		S_1			
通話中	S_5		S_6			
切断待	S_6		S_1			

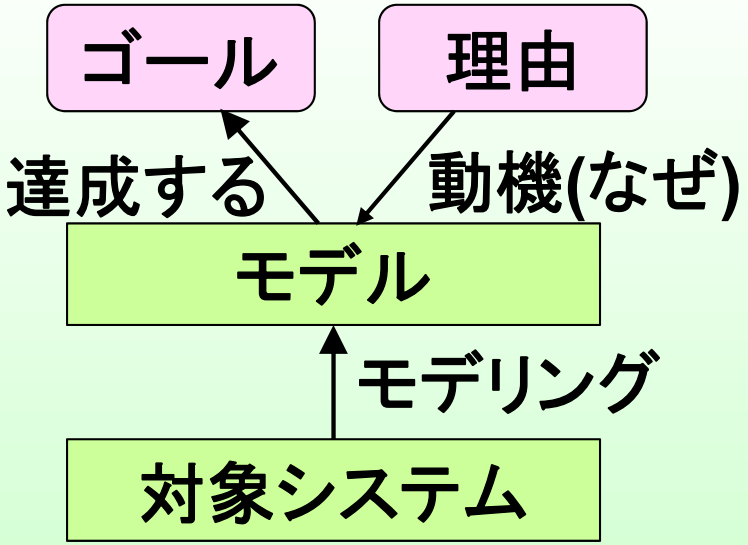
組込みシステムのモデリング
組込みシステムのモデルの抽象化: 論理/物理アーキテクチャ

- 論理アーキテクチャ(機能アーキテクチャ, 機能分割)
  - プラットフォームや実装に依存しないアーキテクチャ
- 物理アーキテクチャ(配置アーキテクチャ, コンピュータ割当)
  - プラットフォームや実装に対応した(依存する)アーキテクチャ
  - コスト, 信頼性, 性能などの違い



実践へのカイド
ゴール(目的)から始めよう

- ゴール(Goal)[目的(Purpose)]
  - モデルが表現すべき姿
  - 理由(Rationale)[なぜ(Why)]: なぜ，そのモデルが必要か?
- ゴールの意義
  - 良いモデリングはゴールの達成
  - まず，ゴールを定義する
- ゴールの特性
  - コンテキスト(対象とスコープ)に依存



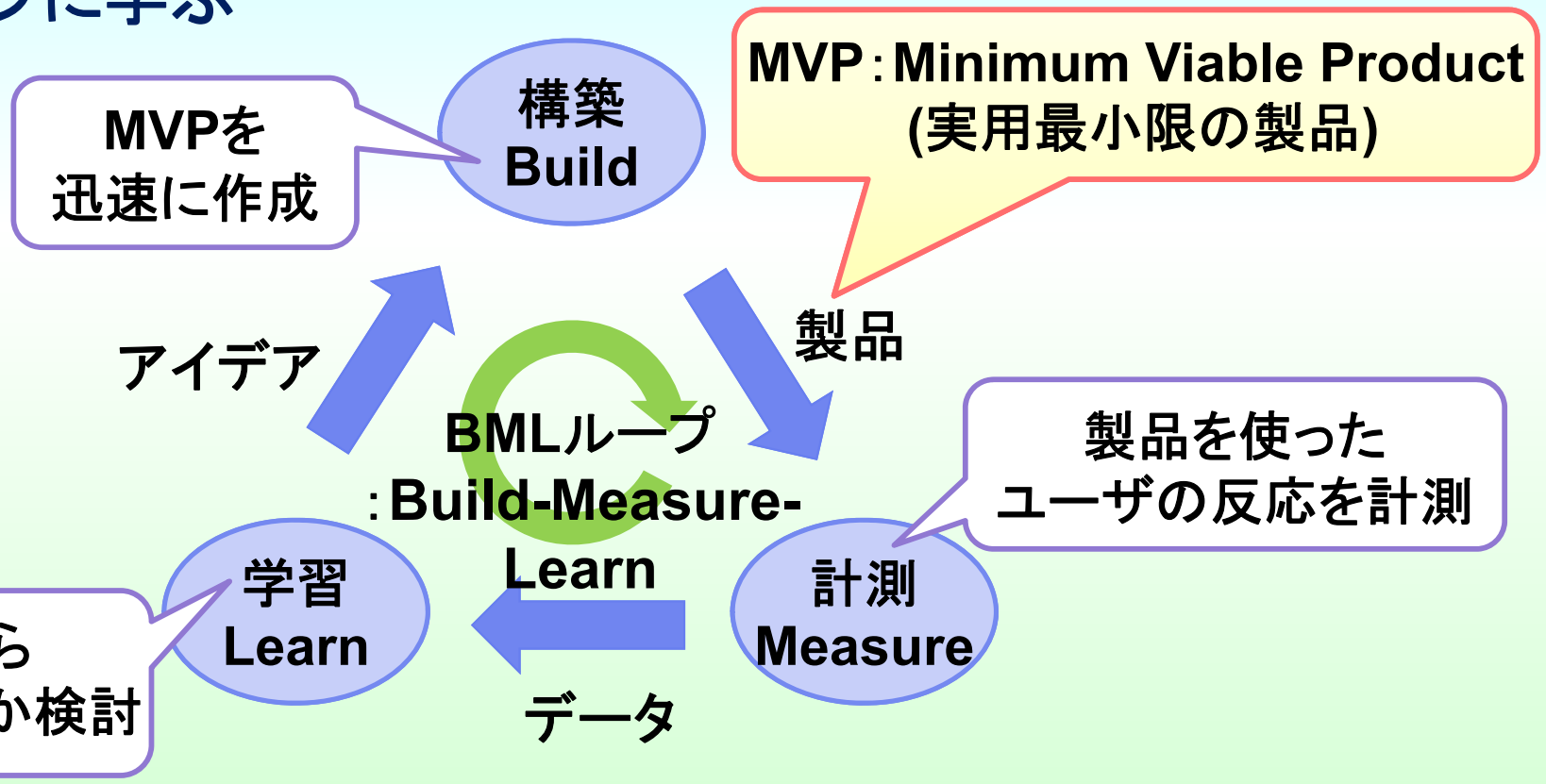
実践へのカイド
仕様としてのモデルを活用しよう

- 「モデル=仕様」: システムの本質的な性質の定義
- 適切なモデリング
  - モデリング/開発対象システムの性質によって適切なモデリング方法の選択
- 組込みシステムのモデリング=挙動の仕様定義
  - 挙動の複雑さ=時間の扱い
- 複雑さ軽減へのアプローチ
  - 関心事の分離/複数視点のモデリング
  - 俯瞰的/ユーザ視点のモデリング
  - 例
    - シーケンス図: ユーザ視点で時間に関するモデル
    - 状態マシン図: 俯瞰的で挙動の構造に関するモデル

実践へのカイド
BMLフィードバックを回して良いモデリング

- モデリング=繰返しフィードバックによる学習プロセス
- 2重の学習: 対象システムの学習×モデリング技術の学習
- BMLループに学ぶ

良い文章を書くための秘訣:
三多
(多く読み，多く書き，多く推敲する)



参考文献: E. Ries, The Lean Startup, Crown Business, 2011
[井口耕二(訳), リーン・スタートアップ，日経BP, 2012].

まとめ

- モデル=仕様
- モデリング
  - 目的に応じた適切なモデリングの必要性: モデリングの黄金則
  - モデリングの意義
    - 複数視点による複雑度の軽減と正しい理解
    - エンジニアリングの基礎: 分析，設計，．．．
    - 理解の共有: コミュニケーションツール
  - 良いモデルとは?
- 組込みシステムのモデリング
  - 挙動のモデリング: 制御フロー/状態遷移に基づくモデル
  - リアルタイム(時間)を扱う難しさ
- 良いモデリングの秘訣
  - 目標から始める，対象に応じた適切な方法，学習フィードバック


良いモデリングで良い開発！
ご清聴ありがとうございます

より深く知るために
参考文献

1) C. Y. Baldwin and Kim B. Clark, Design Rules: The Power of Modularity, MIT Press, 2000 [安藤 晴彦(訳), デザイン・ルール: モジュール化パワー, 東洋経済新報社, 2004].
2) J. Cooling, Software Engineering for Real-Time Systems, Addison-Wesley, 2002.
3) E. W. Dijkstra, et al., Structured Programming, Academic Press, 1972 [野下 浩平ほか (訳), 構造化プログラミング, サイエンス社, 1975].
4) ISO/IEC/IEEE 42010:2011, Systems and Software Engineering - Architecture Description, 2011.
5) JISA REBOK企画WG(編), 要求工学知識体系(REBOK), 第1版, 近代科学社, 2011.
6) T. Kühne, Matters of (Meta-) Modeling, J. of Software and Systems Modeling, Vol. 5, No. 4, Dec. 2006, pp. 369-385.
7) B. A. Lieberman, The Art of Software Modeling, Auerbach Pub., 2007.
8) B. Liskov, Data Abstraction and Hierarchy, Proc. ACM OOPSLA '87, Oct. 1987, pp. 17-34.
9) D. L. Parnas, On the Criteria To Be Used in Decomposing Systems into Modules, Comm. of the ACM, Vol. 15, No. 12, Dec. 1972, pp. 1053-1058.
10) 坂本 賢三, 「分ける」こと「分かる」こと, 講談社学術文庫, Vol. 1767, 講談社, 2006.
11) E. Seidewitz, What Models Mean, IEEE Software, Vol. 20, No. 5, Sep./Oct. 2003,pp. 26-32.
12) M. T. Sewell and L. M. Sewell, The Software Architect's Profession, Prentice-Hall, 2002 [職業としてのソフトウェアアーキテクト, ピアソン・エデュケーション, 2002].

講師紹介

- 1980年: 富士通入社, 大規模通信ソフトウェア, 新端末ソフトウェアの開発・保守に従事
- 1985年～1995年: 大規模通信ソフトウェア(FETEX-3000)の開発管理(プロジェクト管理, 生産性・品質向上等ソフトウェア工学全て)
- 1986-88年: 米国イリノイ大学客員研究員: 高信頼分散処理ソフトウェアの開発技術の研究に従事
- 1995年~2001年: 新潟工科大学情報電子工学科教授
- 2001年~: 南山大学数理情報学部情報通信科(2009年から情報理工学部ソフトウェア工学科)教授

研究室の掲げるテーマ: 実践的ソフトウェア工学

NISE (Network, Information and Software Engineering)

ユーザ指向要求工学, ソフトウェアアーキテクチャ, SOA (Service-Oriented Architecture), クラウドコンピューティング, 組込みソフトウェア設計方法論, 自動車ソフトウェア工学