コンポーネントベース開発方法論
No.1

コンポーネントベース開発とは

鄭 顕志

2010年 6月8日

Rev. 1.0

この講義で何を学ぶのか

- コンポーネントベース開発方法論（CBD）による産業ソフトウェアの分析/設計モデリングにおける実用ノウハウ習得
  - 全体が変更に強く、部分を再利用可能な（大規模）ソフトウェアの迅速な開発と保守
  - 具体的なCBD: Catalysis, UML Components, KobrA
  - 実問題に近いシステムのオブジェクト指向分析/設計、UMLツールによるモデリング演習
  - 各種CBDに共通/異なる特徴の明確化、適用ノウハウ（具体的手順、適用領域）
- 習得ノウハウを、各自が抱える問題に適用する応用力

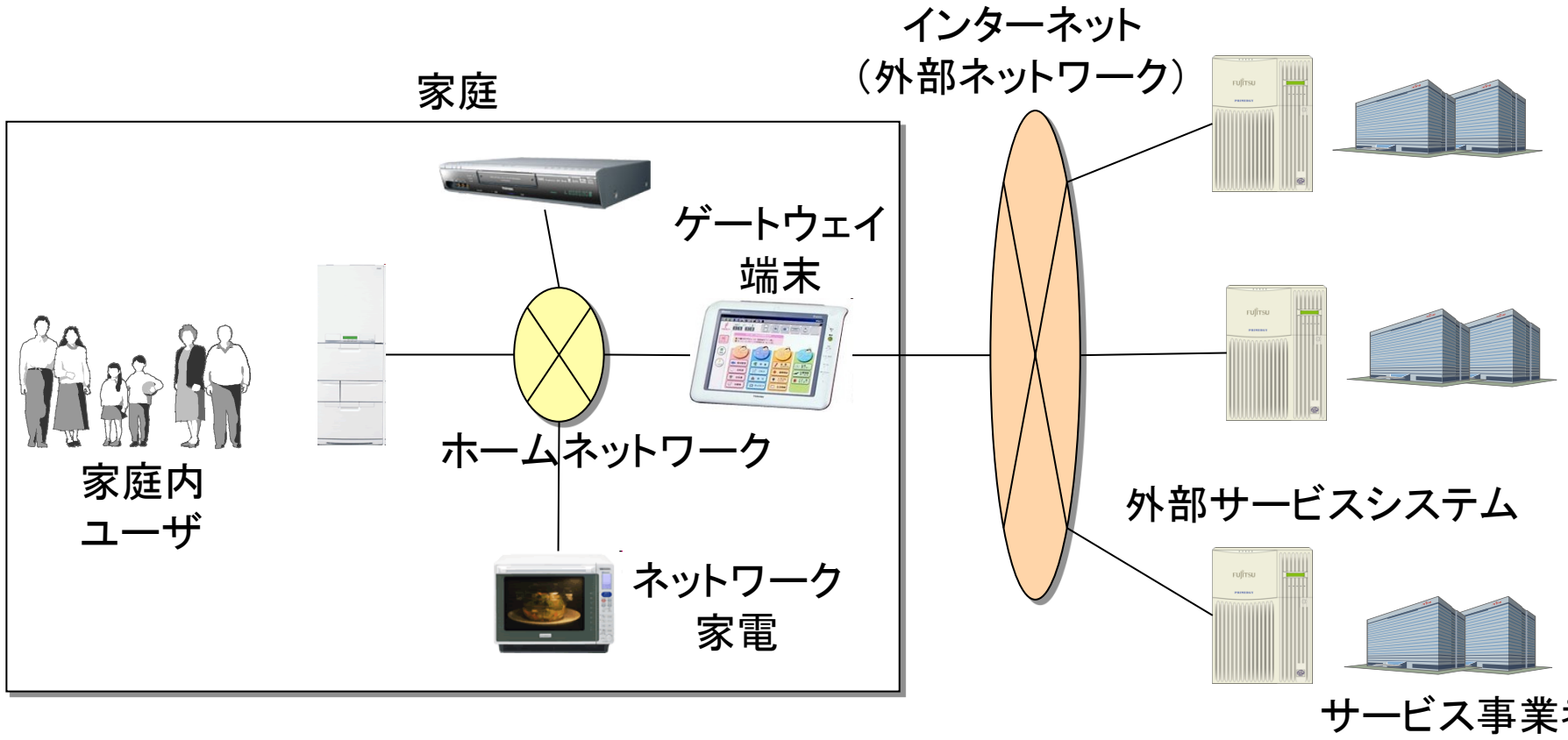
本講義の構成

- No.1 コンポーネントベース開発とは
- No.2 オブジェクト指向分析
- No.3 オブジェクト指向設計
- No.4 オブジェクト指向からコンポーネントベース開発へ
- No.5 プロダクトラインとフィーチャモデリング
- No.6 インタフェース定義とOCL
- No.7 CatalysisおよびUML Componentsとは
- No.8 UML Components による分析
- No.9 UML Components による設計
- No.10 KobrAとは
- No.11 KobrA による分析
- No.12 KobrA による設計
- No.13 コンポーネントベース開発事例と効果
- No.14 手法比較
- No.15 全体のまとめ

情報システム

- システム: 個々の要素が有機的に組み合わされた、まとまりをもつ全体
- 情報システム
  - 要素: ハードウェア、(ハードウェア上の)ソフトウェア
  - 組み合わせ: ネットワーク
  - 全体: ハードウェア／ソフトウェア間のネットワーク接続によって大規模/複雑な要求を満たす
- 例: ネットワーク家電システム
  - 要素: ネットワーク家電(ハード＋ソフト)、外部サービスシステム(ハード＋ソフト)
  - 組み合わせ: ホームネットワーク、外部ネットワーク
  - 全体: ネットワーク家電間のホームネットワーク接続、および、ネットワーク家電と外部サービスシステムの外部ネットワーク接続によって大規模/複雑な要求を満たす

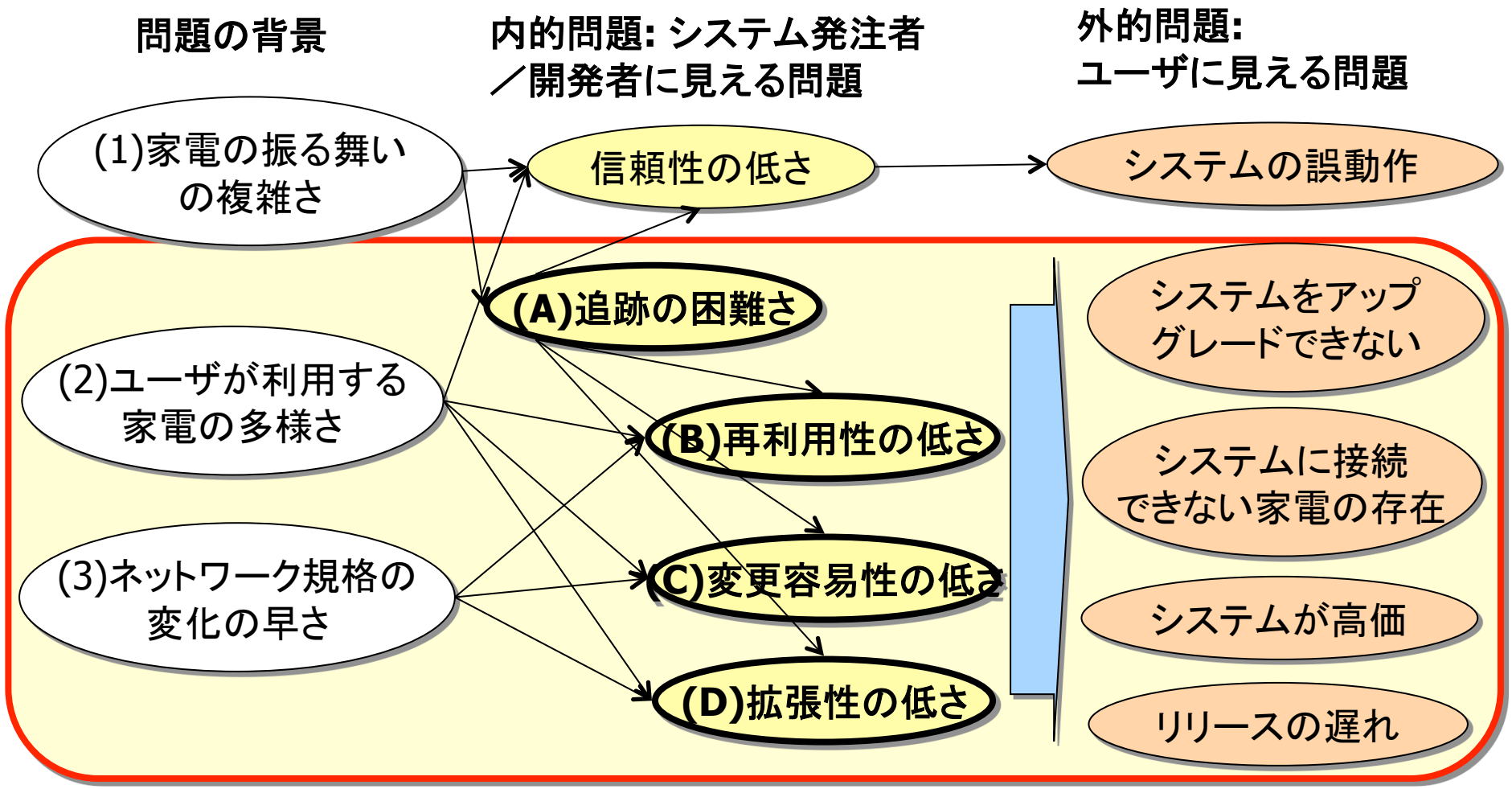
ネットワーク家電システムの概念図



ニーズとシーズ: ネットワーク家電システムを例に

- ニーズ（要求）: 要求の高度化、高付加価値の要請
  - ネットワーク家電単体ではなく、複数のまとまりによって、より価値（機能性、安全性、信頼性）の高い機能実現の要求
  - 複数のネットワーク家電の組み合わせ
    - 例: DVDレコーダが、HDDレコーダ内の映像を、DVDにコピーする。
  - ネットワーク家電と外部サービスの組み合わせ
    - 例: 電子レンジが、入れられた食品状況を調べて、調理可能な料理とその方法を、料理情報配信サイトに問い合わせる。
  - 複数のネットワーク家電と外部サービスの組み合わせ
    - 例: 冷蔵庫が、庫内の食品状況と他の調理家電が提供する調理方法を調べて、不足する食品をスーパマーケットに自動注文する。
- シーズ（技術基盤）:
  - ネットワーク家電に搭載可能なソフトウェアサイズの増大
  - ホームネットワーク接続技術の発展、規格の標準化
  - 外部ネットワーク接続技術の発展、規格の標準化

コンポーネントベース開発が解決する問題

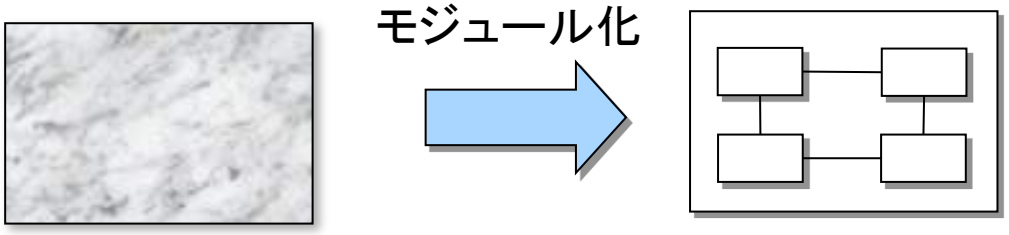


複雑さへの挑戦:
問題を解決する3つの技術

- 根元的問題: 複雑な要求を、高品質なソフトウェアとして効率よく開発し、効率よく管理・保守したい
- 2つの技術
  - モジュール化
  - 抽象化

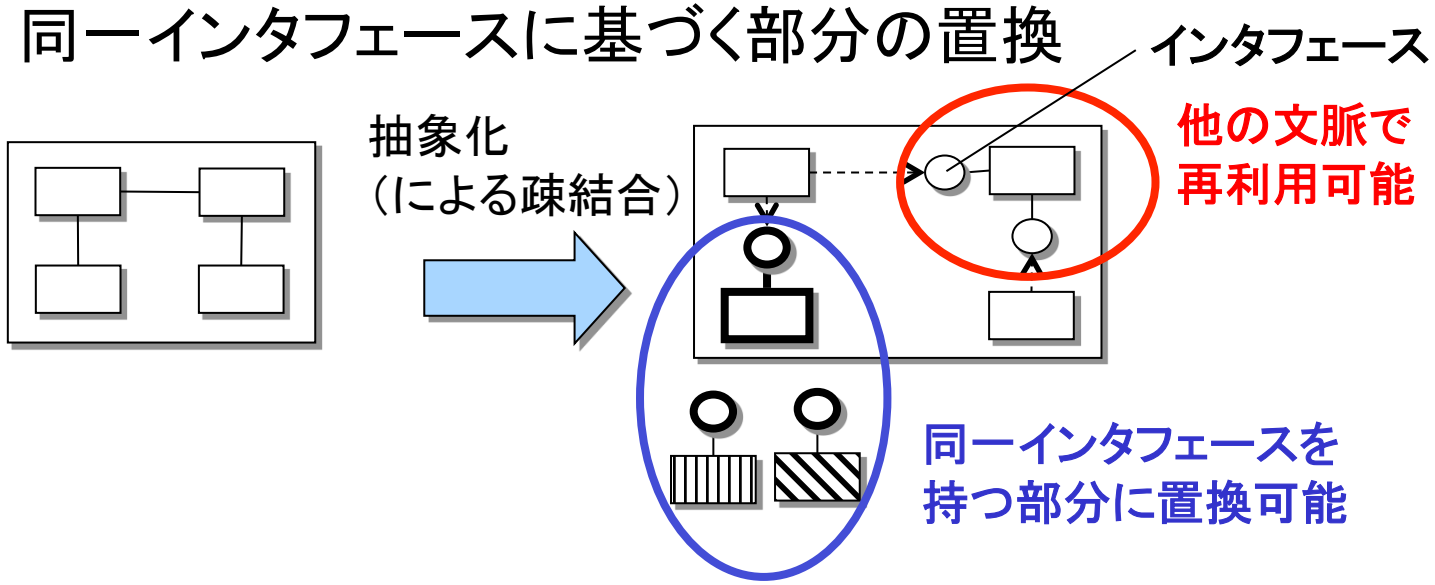
技術1: モジュール化によるソフトウェアの分割統治

- 分割統治
  - 複雑な全体問題を、小さな部分問題の集合に分割し、個々の部分問題の解を集めて全体問題の解を得る
- モジュール化
  - ソフトウェア全体を、一枚岩ではなく、幾つかの部品から構成されるように分析・設計する
- 効果
  - 問題の切り分け、部分単位の開発
  - 問題や影響の局所化

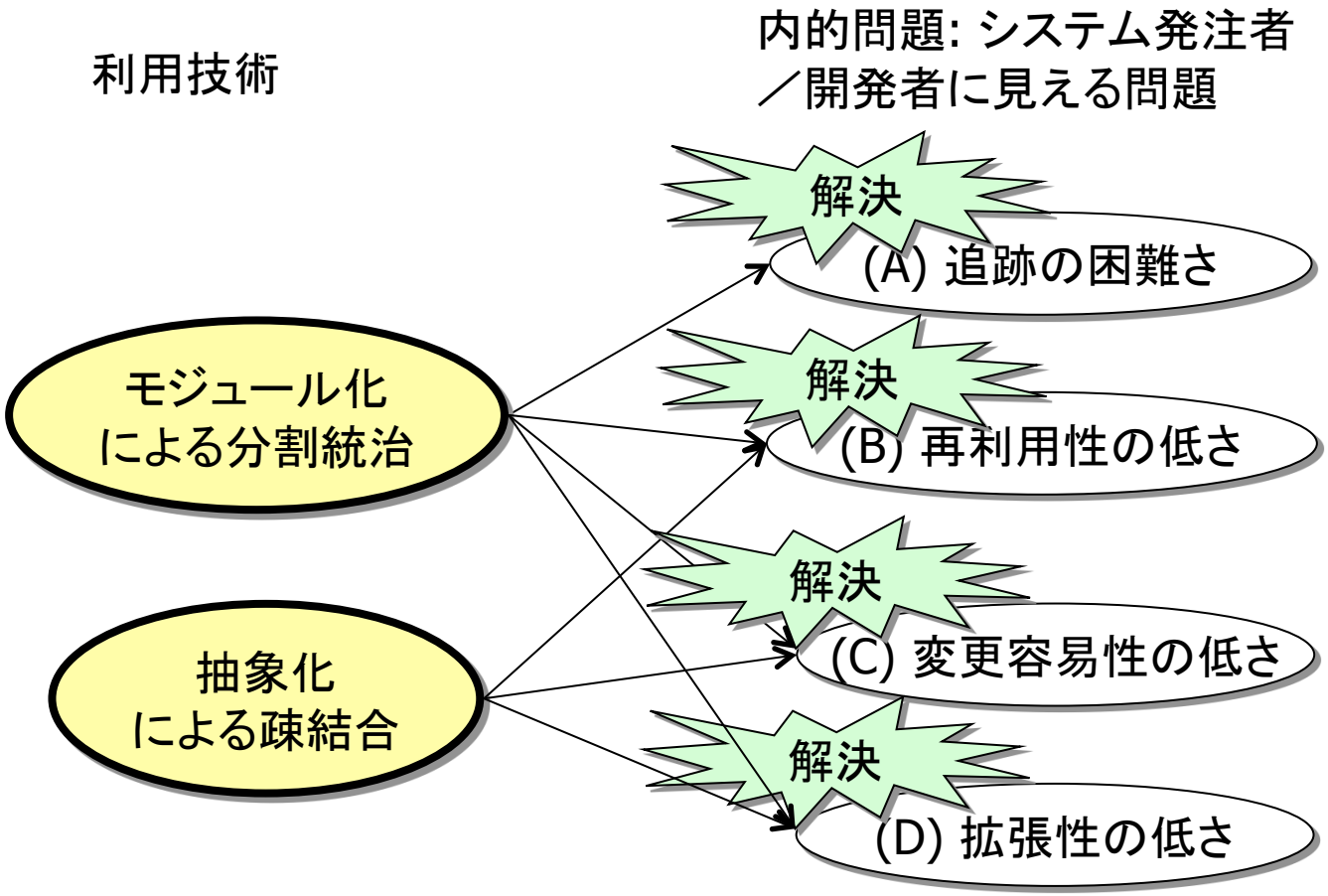


技術2: 抽象化（による疎結合）

- ソフトウェアの部品が他の部品に直接依存せずに、特定の機能側面を表すインタフェースに依存するように分析・設計する
- より実践的には、アーキテクチャの標準化が必要
- 効果
  - 部分単位の切り出しと再利用
  - 同一インタフェースに基づく部分の置換



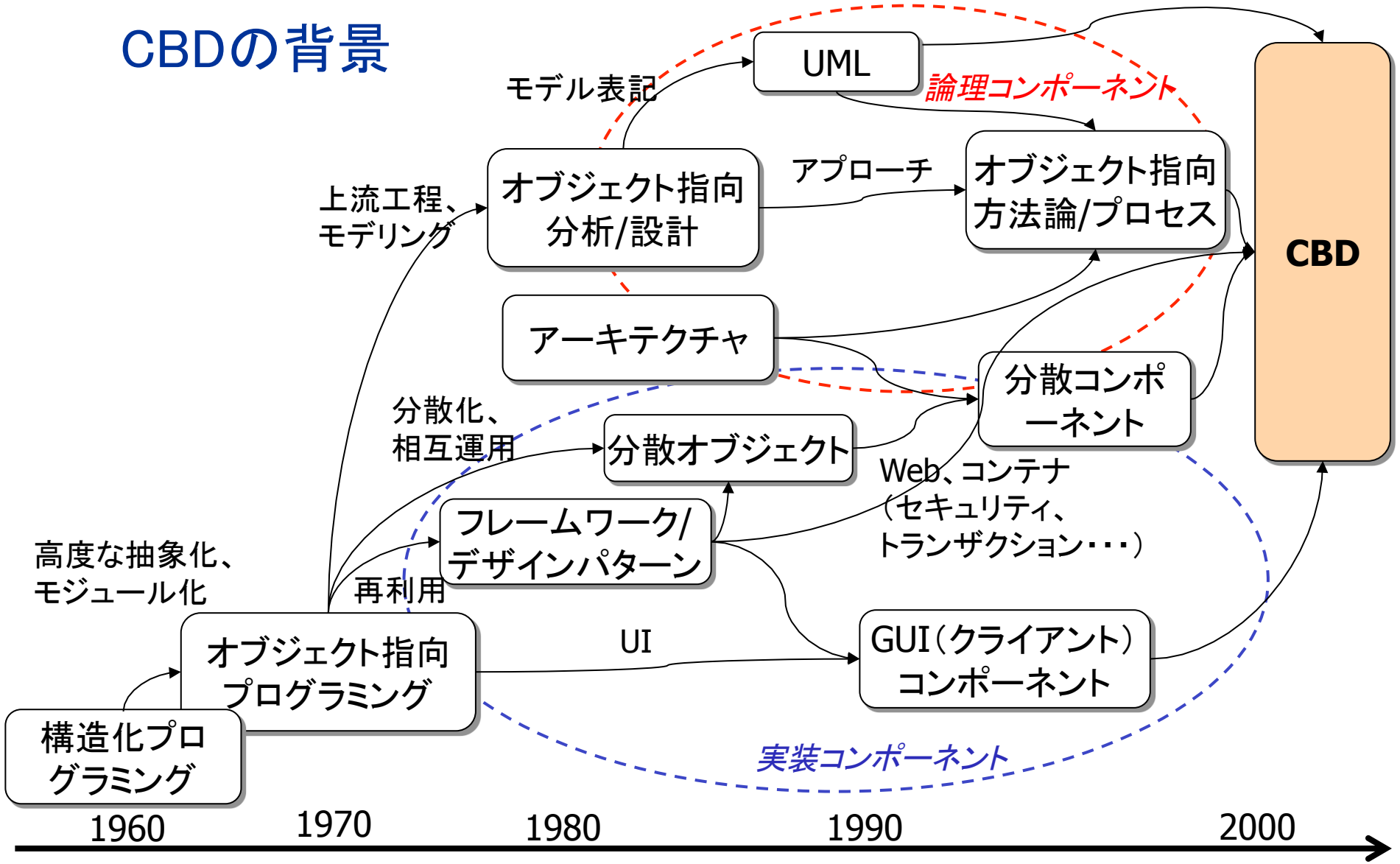
問題と利用技術の関係



何が必要なのか

- 抽象化とモジュール化により再利用や組合せを促す分析/設計手法・開発方法論
  - 主に再利用可能、分離独立開発可能な論理コンポーネント集合としての分析と設計（Dev. for Reuse）
  - コンポーネントリポジトリ内の再利用可能な実装コンポーネントを参照して組み入れることを意図した分析/設計
  - 実装コンポーネント技術に非依存
  - 例: Catalysis, UML Components, KobrA
- 実装レベルの組合せ技術
  - 主に再利用および組み立てて実装（Dev. by Reuse）
  - コンポーネントベース開発方法論に非依存
  - 例: J2EE/EJB, Web MVC FW(Struts, Turbineなど), GUI部品（ActiveX, JavaBeansなど）

CBDの背景



コンポーネントベース開発を簡単にまとめると

- コンポーネント
  - 明確に定義されたインタフェース集合を提供/要求する、自立した、開発のあらゆる側面/段階に対応するソフトウェア構成単位
  - 論理コンポーネント(ビジネスコンポーネント): 機能要求を分割した可視化、管理単位
  - 実装コンポーネント: 機能要求対応実行、テスト、再利用単位
- コンポーネントベース開発(≠EJB/J2EE、CORBA Components …)
  - 分析から実装まで一貫した、オブジェクト指向を発展させたコンポーネントアプローチ
  - 自立(NOT 孤立)と協調、部品化再利用、反復的プロセス
  - 連続/個別再帰な多層非循環アーキテクチャ

⇒ 開発/変更コスト削減(1/2～1/5)、個別分離開発、再利用、スケーラブル

モデルとモデリング

- モデル
  - 状況の、捨象・抽象・単純化によって得られる、ある人による明示的な解釈
- モデリング
  - モデルを作る行為
  - 主観を客観にする行為

悪いモデル

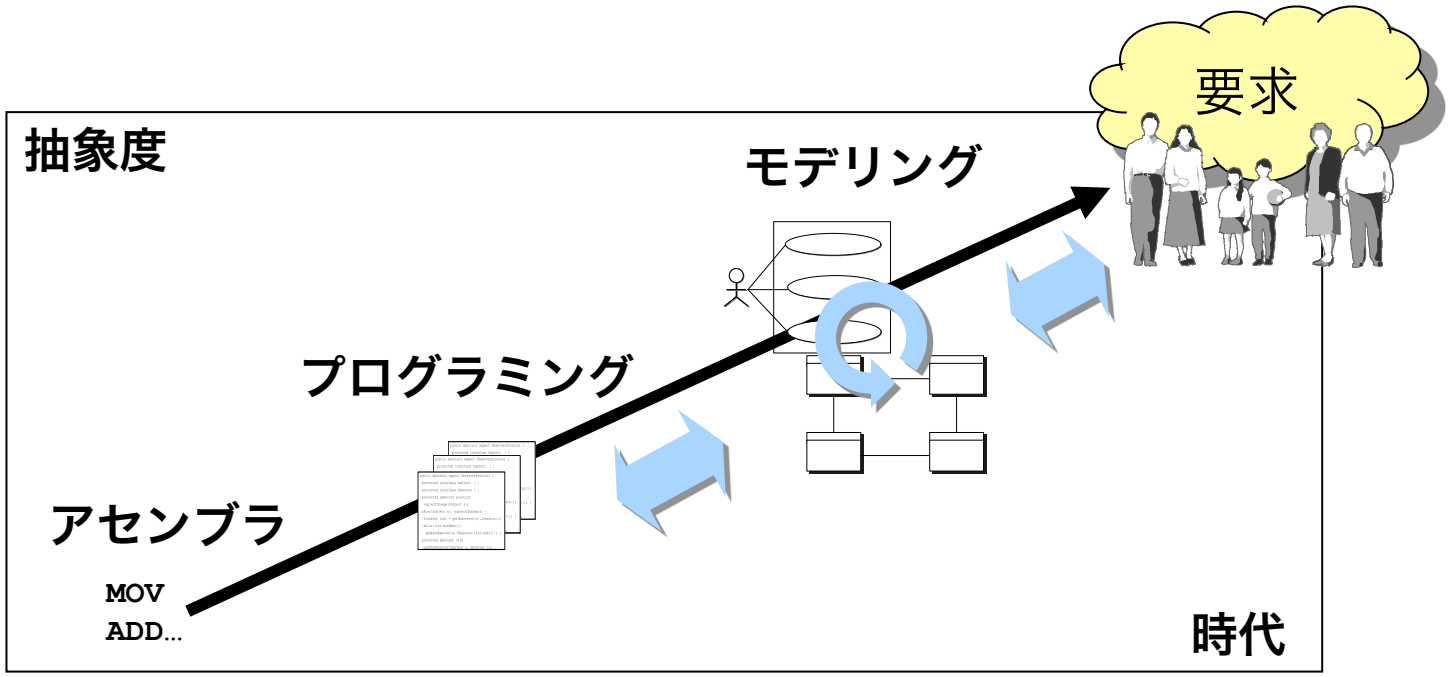
異なる事柄が混じっている
誰もが一意に解釈できない
他のモデルとの関係が不明確

良いモデル

1つの注目する事柄の本質のみ表す
誰でも一意に解釈できる
他モデルとの関係を追跡できる

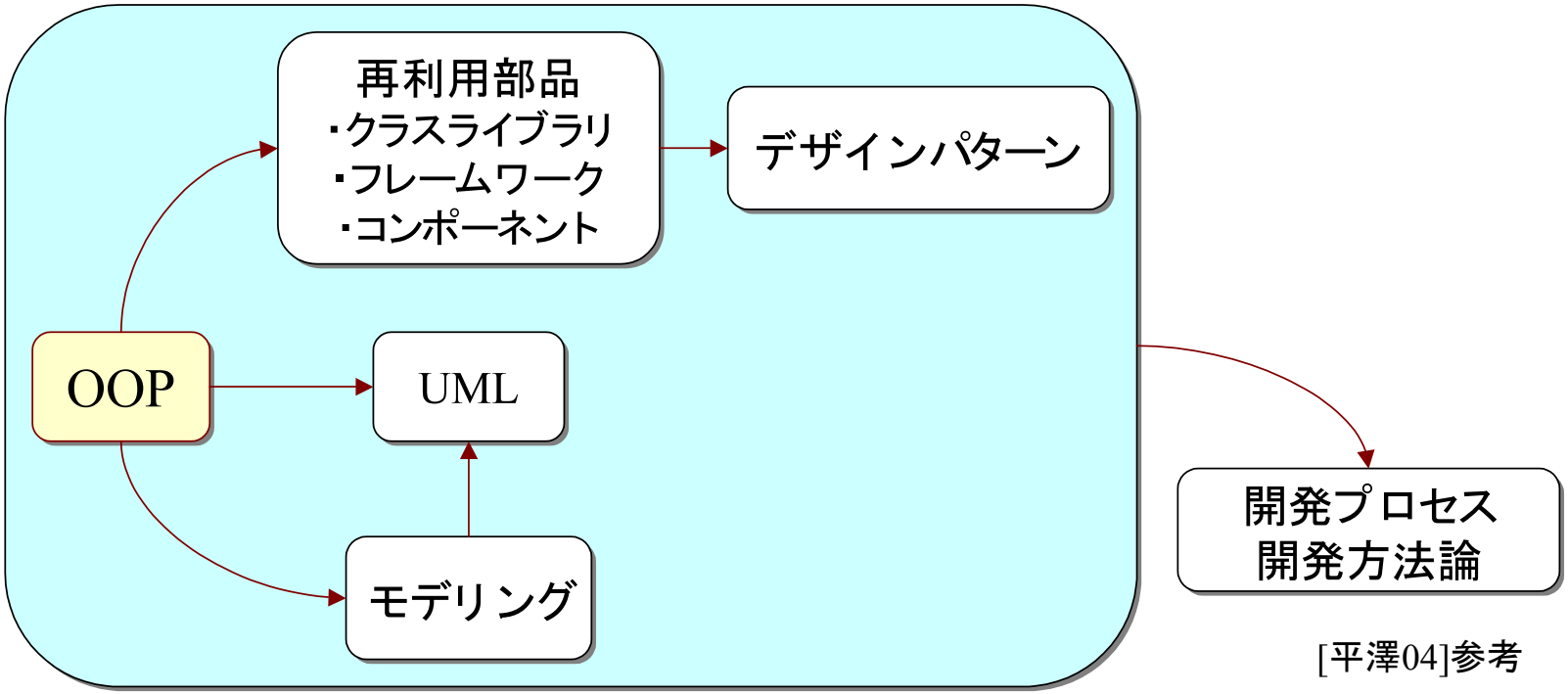
プログラミングからモデリングへ

- 「どう動かすのか」から「何をしたいのか」へ
- 問題の早期分離検討。理解共有。職人芸から工業へ。



オブジェクト指向プログラミングからオブジェクト指向開発方法論へ

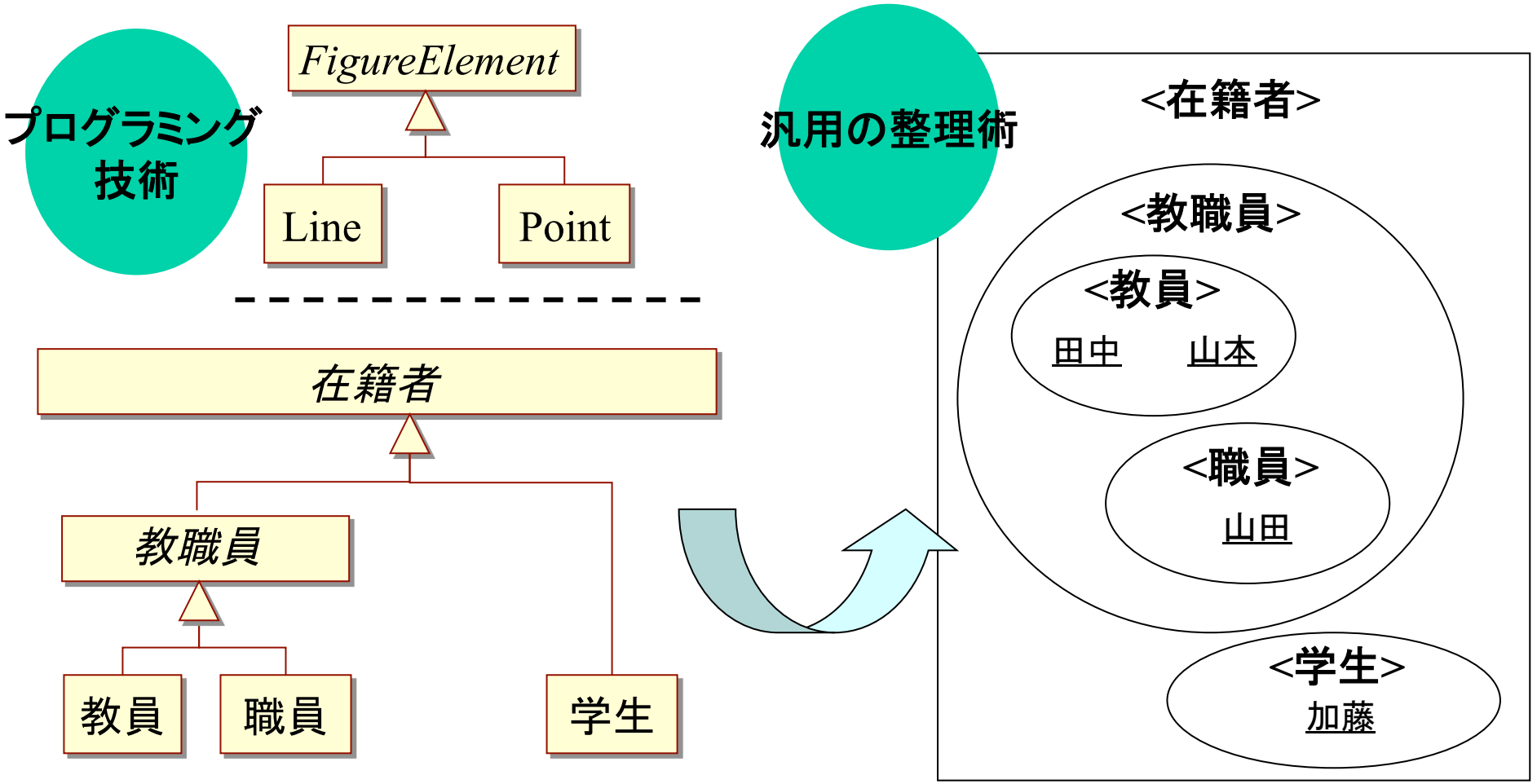
- 業務分析、要求定義（概念モデリング／ユースケース）、オブジェクト指向設計
- OOPにおける仕組みを上流工程へ流用
  - クラス: 集合論に基づくモデリング
  - メッセージパッシング: 役割分担に基づくモデリング



[平澤04]参考

クラスと集合論

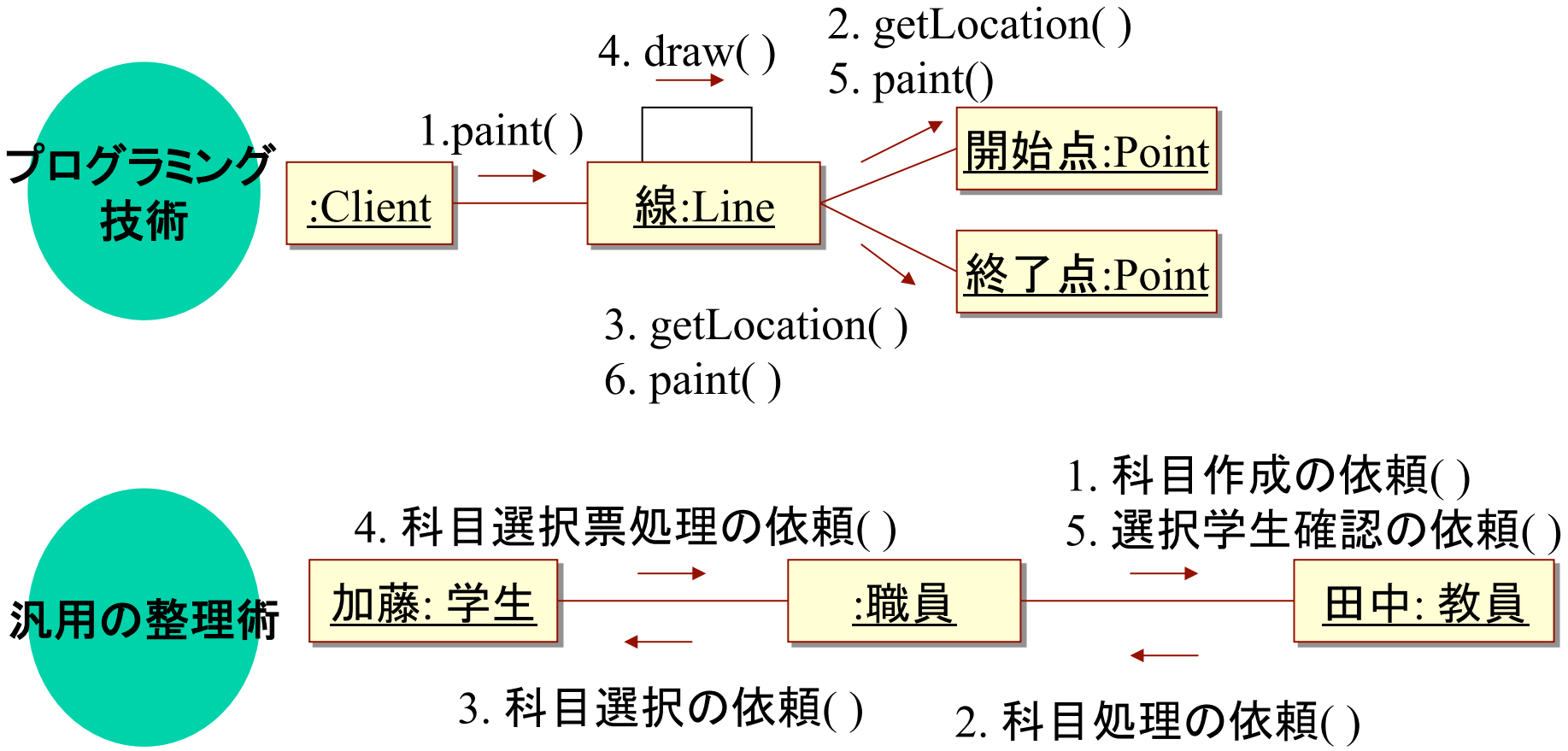
- クラス、サブクラス、オブジェクト
- 集合、部分集合、要素



[平澤04]参考

メッセージパッシングと役割分担

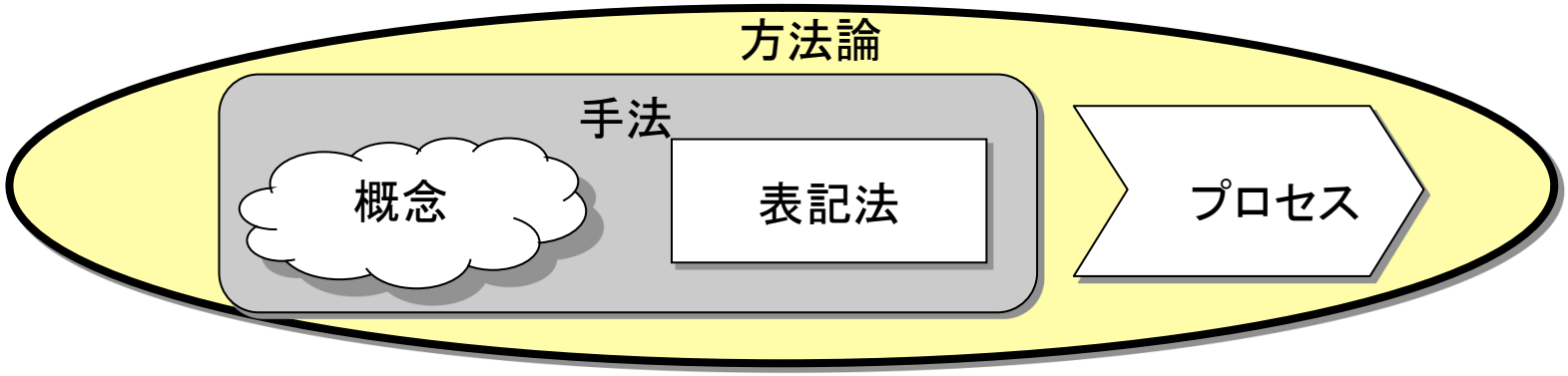
- クラス、メソッド、メッセージパッシング
- 役割、仕事、仕事の依頼（あるいは命令）



[平澤04]参考

オブジェクト指向（OO）方法論

- オブジェクト指向開発方法論≠手法
  - よくある定義: オブジェクト指向に基づく開発手法＋プロセス
    （本当の定義: オブジェクト指向開発手法を比較し論じる枠組み）
  - オブジェクト指向開発手法＝表記法＋概念
- 要素
  - 概念=オブジェクトの分類方針、抽出方針
  - 表記法＝UMLで統一
  - プロセス=どの段階で、オブジェクトをどう捉えて関連付けて表記するかという手順の順序集合



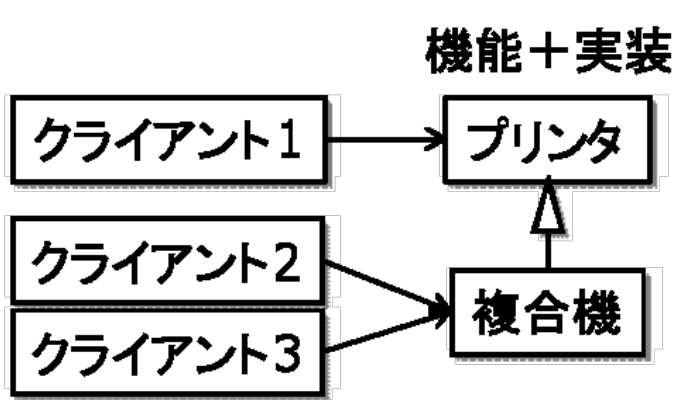
概念: オブジェクトの分類

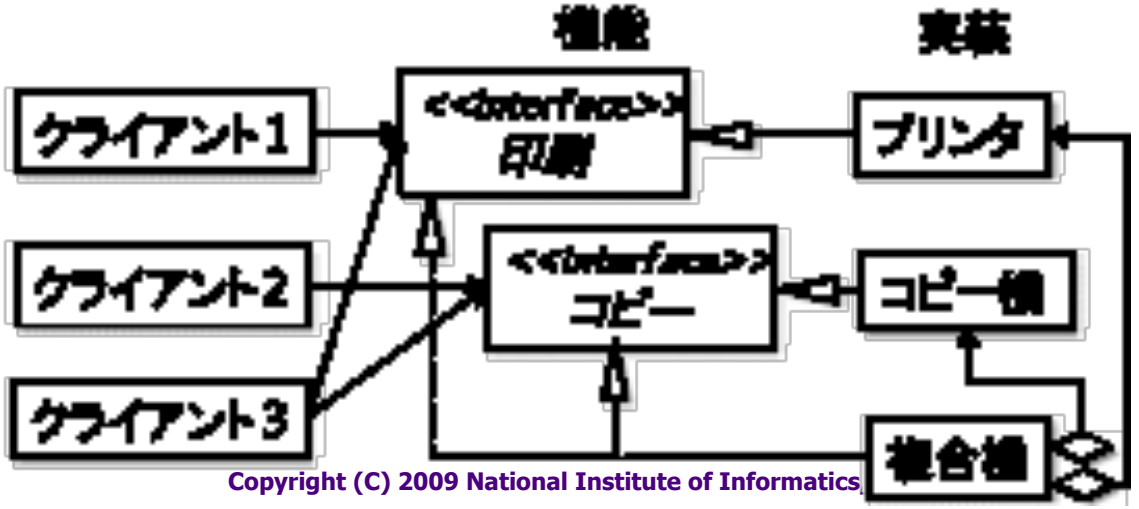
拡張(継承)の重視

- 拡張に基づくis-a階層によるオブジェクト集合の分類
- インタフェース継承と実装継承の両利用 →結合度の増大
- ホワイトボックス利用

集約の重視

- 集約に基づくpart-of階層によるオブジェクト集合の分類
- インタフェース継承のみ利用
  → 機能と実装方法の分離、
  結合度の減少
- ブラックボックス利用

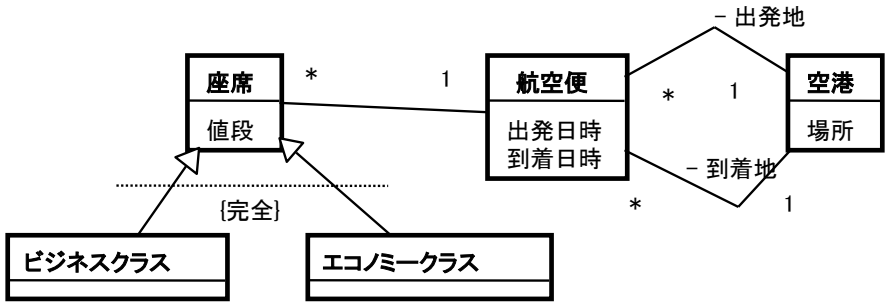




概念: オブジェクトの捉え方(抽出)

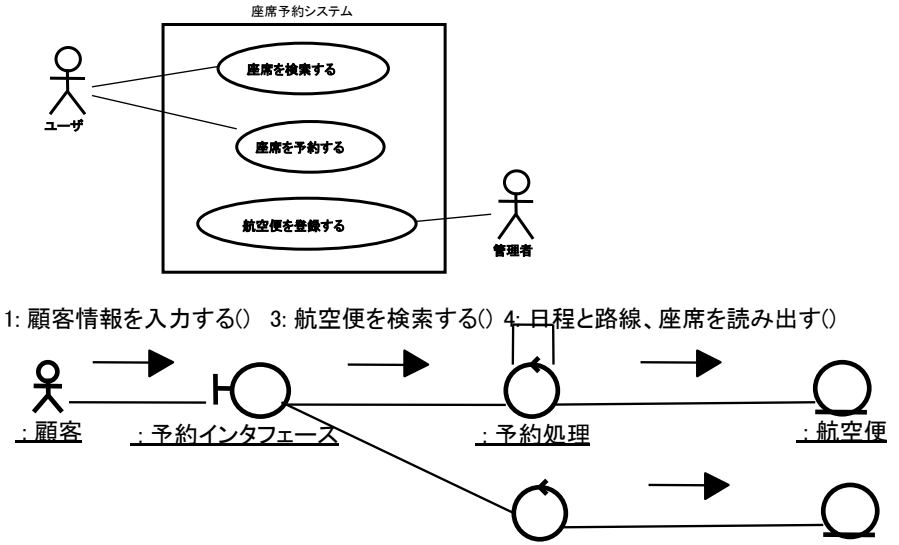
データ駆動

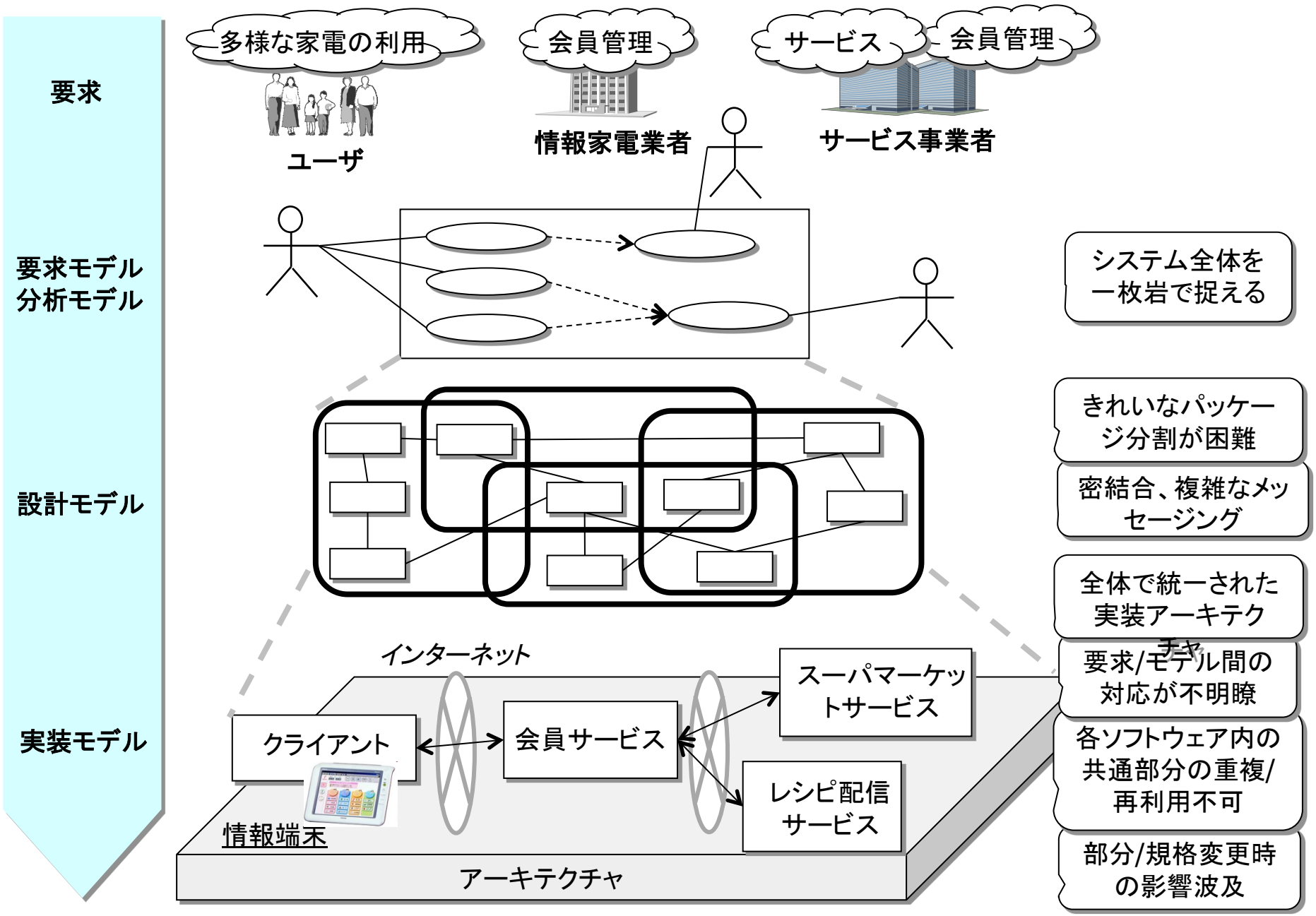
- データ(情報)に着目してオブジェクト抽出
- 例: OMT法、Coad-Yourdon法、Shlaer-Mellor法



責任駆動

- 責務に着目してオブジェクト抽出
- 例: CRC法、OOSE法(ユースケース駆動)

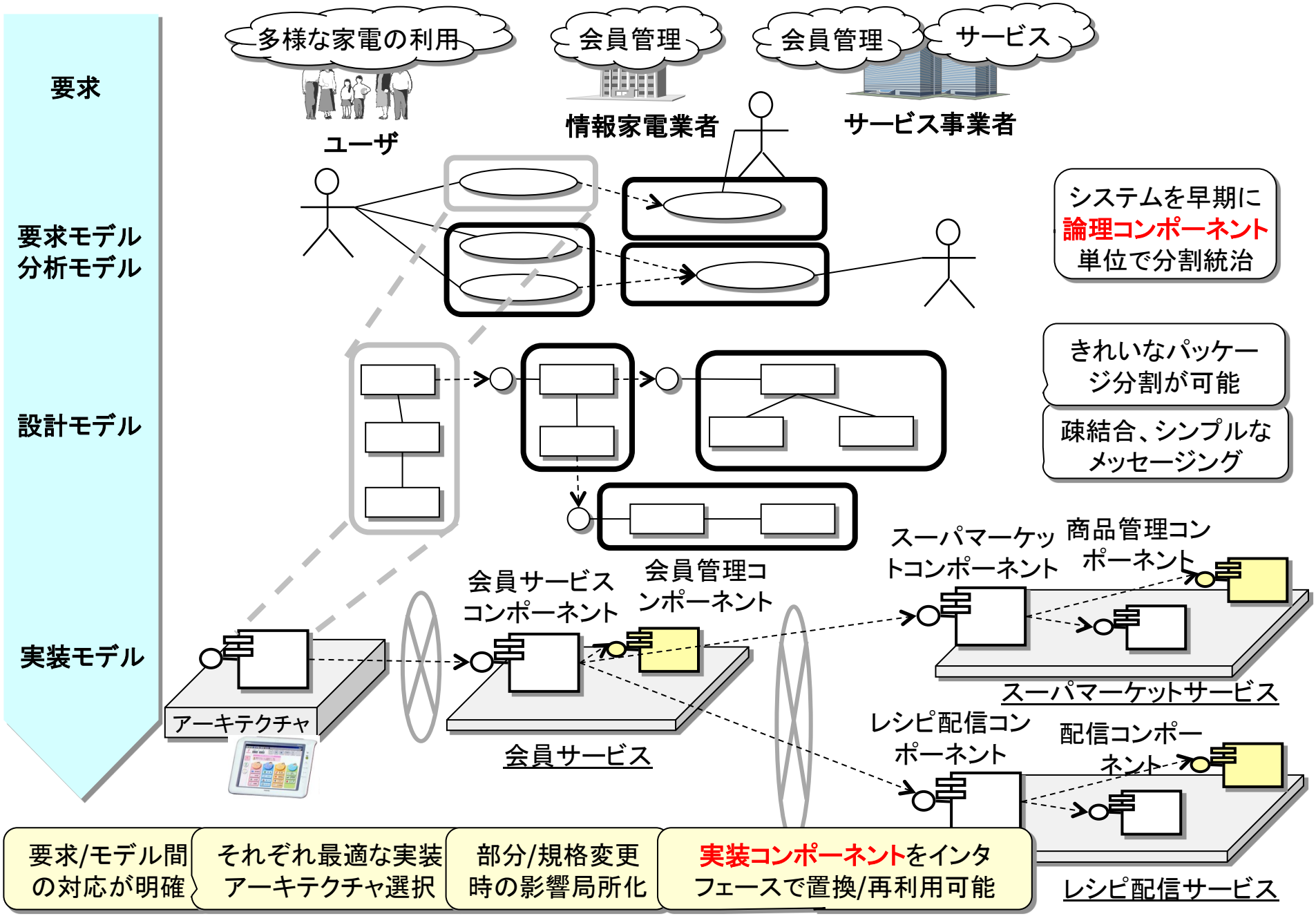




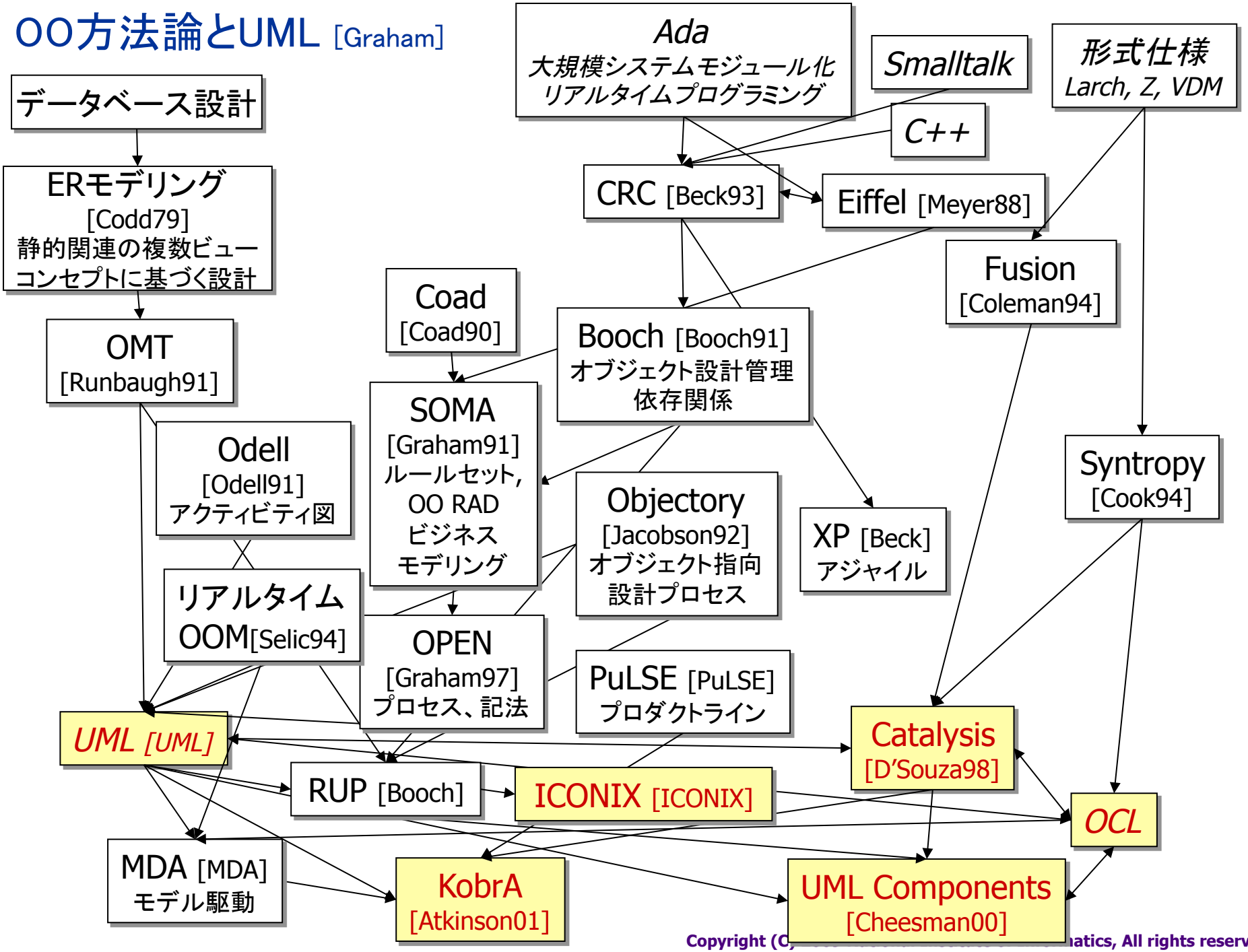
オブジェクト指向開発からコンポーネントベース開発へ

コンポーネントベース開発（CBD）：

- 元々は、大規模エンタープライズアプリケーションを、分散コンポーネントの組み合わせによって実現するようにオブジェクト指向開発方法論を発展させたもの
- ただし、CBDの多くは
  - 早期のインタフェースに基づく分割統治を目指す
  - 継承よりも委譲を重視する
  - 機能分割もしくはデータ分割手順を明示する
- 最近では、重量級コンテナを用いる分散コンポーネントに限らず、様々な実装コンポーネントを利用する



OO方法論とUML [Graham]



代表的なCBDと比較

- Catalysis [D'Souza99]
  - Alan Wills（TriReme）、Desmond 'Souza（Kinetium）、1998
  - 表記法にUMLの独自拡張
  - 業務分析・システム分析・コンポーネント設計
- UML Components
  - J. Cheesman（Component Source）, J. Daniels、2000
  - 表記法にUML
  - 業務分析・システム分析・コンポーネント設計
  - Catalysisをベース
- KobrA
  - Colin Atkinson, Joachim Bayer, Dirk Muthig（IESE）、2001
  - 表記法にUML
  - システム分析・コンポーネント設計
  - プロダクトライン開発手法PuLSEをベース

CBD	表記法	分類	抽出	プロダクトライン
Catalysis	UML拡張, OCL	集約	データ駆動	非考慮
UML Components	UML, OCL（UML 2.0）	集約	データ駆動	非考慮
KobrA	UML	拡張（継承）	責務（機能）駆動	考慮

他のCBD

- ComponentAA（Component-based Application Architecture）[ComponentAA]
  - 銀林純ほか（富士通）、2000
  - コンポーネントの利用を前提として、業務分析・エンティティ分析・シナリオ分析・ビジネスモデル／サービス仕様定義・システム構築を行う開発体系と製品群
- コンポーネント指向業務設計技法（HIPASE/AGORA）[AGORA]
  - 団野博文，湯浦克彦，岩渕史彦，津田道夫（日立製作所）、1999年
  - 独立した業務機能をカプセル化するコンポーネント（アプリケーションコンポーネントとビジネスコンポーネント）概念を導入した業務設計とソフトウェア設計
- コンポーネントベース・フレームワーク開発手法 [吉田02]
  - 吉田和樹（東芝），本位田真一（NII）、2001年
  - 処理フローに対応して、DAG状に組み合わせ可能なコンポーネント集合としてフレームワークを分析／設計する手法
- Castek's CBD/e [Castek]
  - Castek、2000年
  - 組織にコンポーネントベース開発手法を教育・導入するプロセス（アセスメント・計画・教育・インフラ構築・実行・改善）
- MaRMI III [MaRMI]: Catalysisベース、品質管理・プロジェクト管理の追加
- Select Perspective [Select]: Catalysisベース、MDA・文書テンプレート

CBDの実適用と効果

- 潜在的な再利用の可能性 [W.Tracz]
  - アプリケーションコードの40～60%は他で再利用可
  - アプリケーション特化なコードは全体の高々15%程度
  - ビジネスアプリケーション設計の60%は再利用可
  - プログラム機能の75%は、複数プログラムに共通
- CBD調査報告
  - 企業システムの80%は何らかの形でコンポーネントを利用している [Cutter]
  - 2003年までに少なくとも70%のアプリケーションはコンポーネントを利用して構成される [Gartner]
- CBD個別事例
  - ComponentAA: 300プロジェクト適用、品質向上、生産性1.5～2.5倍 [Ginbayashi00]
  - Castek: 開発コスト20%削減、新規開発時の機能10%をリポジトリから再利用 [Castek]
  - UML Components: 実ホワイトボードシステム開発、厳密なインタフェース定義と部分開発/更新に成功 [Kudo05]

討論:「9つの神話(Myth)」は正しいか? [Tracz88]

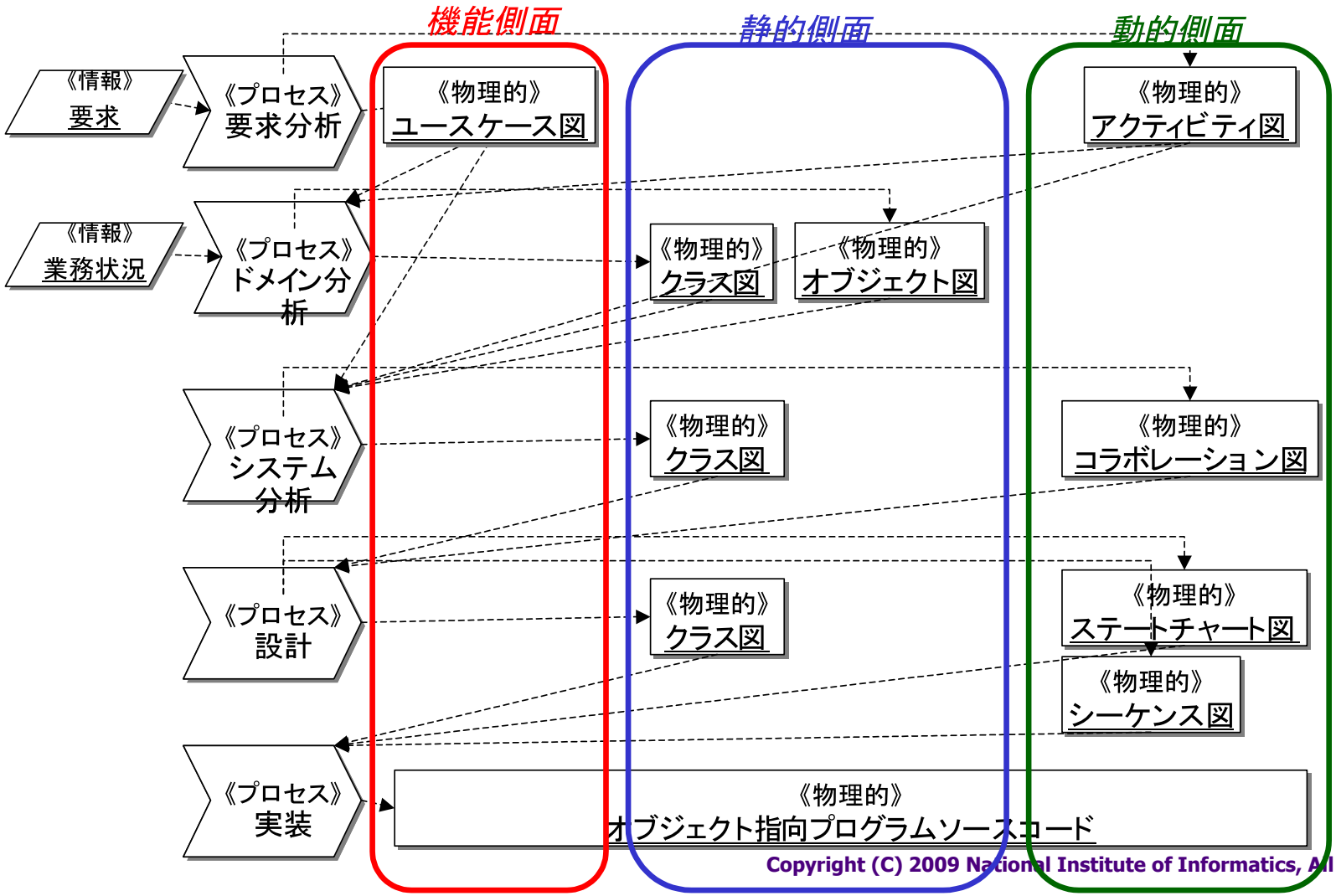
1. 再利用は、技術的な問題だ
2. 特別なツールが、再利用には必要だ
3. コードの再利用は、生産性を飛躍的に向上させる
4. 人工知能が、再利用問題をいずれ解決する
5. 日本は、再利用問題を解決した
6. Ada(に代表される高度な実装言語)は、再利用問題を解決した
7. 再利用可能部分からソフトウェアを設計するのは、集積回路からハードウェアを設計するようなものだ
8. 再利用されたソフトウェアは、再利用可能なソフトウェアだ
9. 再利用は、今まさに行われ始めたことだ

ICONIXとは

オブジェクト指向開発プロセス

- オブジェクト指向の考え方に基づき、UMLを用いて、分析から実装までのモデル変換を行う手順
- 一般にプロセス単体ではなく、考え方・表記法とセットで定義される
- 例:
  - オブジェクト指向開発方法論（Rational Unified Process（RUP）／Unified Process（UP））が定義する開発プロセス
  - コンポーネントベース開発方法論（Catalysis、UML Components、KobrA）が定義する開発プロセス

一般的なオブジェクト指向プロセスにおけるUMLの使用



5つの開発工程

- ドメイン分析: 業務状況はどのようなものか
  - 業務状況→概念モデル
- 要求分析: システムで何を実現したいのか
  - 要求（＋概念モデル）→要求モデル
- システム分析: システムで何を実現したいのか
  - 概念モデル＋要求モデル→分析モデル
- 設計: システムをどのように実現したいのか
  - 分析モデル→設計モデル
- 実装: システムをどのように、どのような環境に配置したいのか
  - 設計モデル→実装モデル

5つのモデル

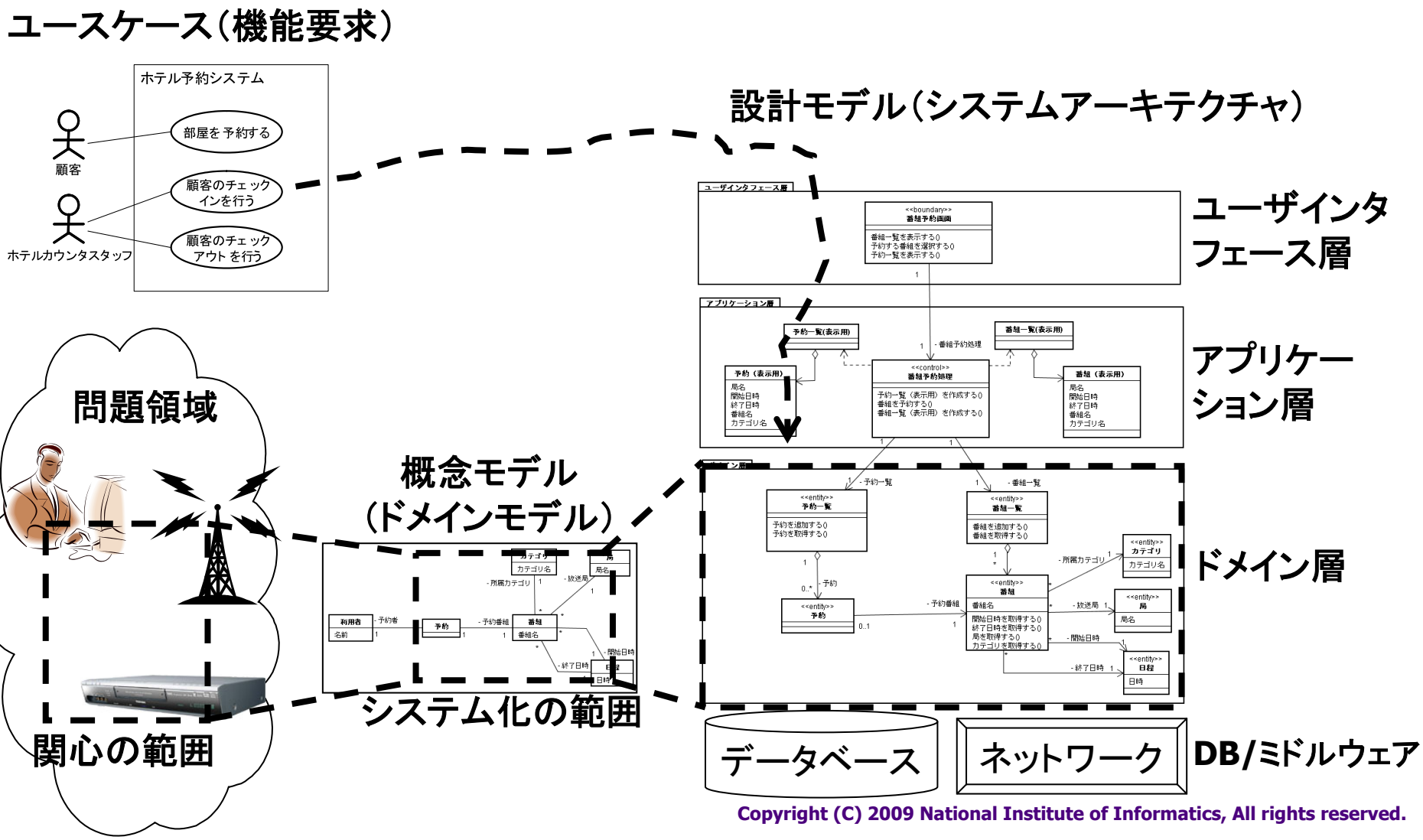
- 概念モデル
  - 対象業務の世界を構成する概念と概念間の関係を表すモデル
- 要求モデル
  - 顧客がシステムに望む事柄を表すモデル
- 分析モデル
  - 実装方法を関知せずに、対象業務についてシステム化する事柄を表すモデル
- 設計モデル
  - システム化する事柄と、その実装方法を表すモデル
- 実装モデル
  - プログラムソースコード、(コンパイル後の)プログラム

ICONIX [ICONIX]

- ユースケースを中心に実装まで行う軽量なオブジェクト指向開発プロセス
  - 分かりやすい
  - 属人性が低い
  - 小規模開発に特に適する
- 概念
  - オブジェクトの分類: (どちらかといえば)継承の重視
  - オブジェクトの抽出: 最初はデータ駆動、途中から責任駆動

ICONIXによる典型的開発イメージ

- ポイント: データはシームレス



まとめ

- 大規模化・複雑化する情報システムの開発における種々の問題
- 問題に対する基本アプローチ
  - 抽象化
  - モジュール化
- コンポーネントベース開発の位置づけ
  - 方法論とUML
  - オブジェクト指向開発との相違:
    - アーキテクチャ, 分割統治, インタフェース, 可変性・共通性, 再利用[のための/による]開発
- ICONIX
  - まずは従来のオブジェクト指向開発を振り返って、開発上の問題を把握しよう

参考文献

- [Graham] Ian Graham or Alan Wills: UML – a tutorial, http://www.sci.brooklyn.cuny.edu/~kopec/uml/uml_tutorial.pdf
- [Codd79] E. F. Codd: Extending the Database Relational Model to Capture More Meaning. ACM Trans. Database Syst. 4(4): 397-434(1979)
- [Beck93] Kent Beck, CRC: finding objects the easy way, Object Magazine, v.3 n.4, p.42-44, Nov./Dec. 1993
- [Meyer88] B. Meyer: Object-Oriented Software Construction, Prentice Hall, 2nd Edition, 2000
- [D'Souza98] D'Souza and Wills: Objects, Components, and Frameworks with UML, Addison-Wesley, 1999.
- [Cheesman00] John Cheesman and John Daniels: UML Components: A simple process for specifying component-based software, Addison-Wesley, 2000.
- [Atkinson01] Atkinson et. al.: Component-Based Product Line Engineering with the UML, Addison-Wesley, 2001.
- [ICONIX] D. Rosenberg and K. Scott: Use Case Driven Object Modeling with UML, Addison Wesley, 2000(邦訳:『ユースケース入門』)
- [ComponentAA] 富士通株式会社: ComponentAA/BRMODELLING, http://software.fujitsu.com/jp/product/use/compo/caa.html
- [AGORA] 団野博文, 湯浦克彦, 岩渕史彦, 津田道夫: コンポーネント指向業務設計技法(HIPACE/AGORA)の開発, オブジェクト指向シンポジウム'99, 1999.
- [吉田02] 吉田和樹, 本位田真一:コンポーネントベース・フレームワーク開発手法におけるコンポーネントの抽出・設計方法論, 情報処理学会論文誌, Vol.43, No.1, 2002.
- [Castek] Castek, http://www.castek.com
- [MaRMI] Dong-Han Ham, et al.: MaRMI-III: A Methodology for Component-Based Development, ETRI Journal, Volume 26, Number 2, April 2004
- [Select] Select Perspective: http://www.selectbs.com/
- [Cutter] Cutter Consortium Business Technology Trends and Impacts Services research, http://www.cutter.com/consortium/index_trends.html
- [Gartner] Gartner Group, http://www.gartner.com/
- [Ginbayashi00] J. Ginbayashi et al.: Business Component Framework and Modeling Method for Component-based Application Architecture, IEEE EDOC'00, 2000.
- [Kudo05] T.N. Kudo et al.: Using UML Components for the specification of the Whiteboard tool, TIDIA, Information Technology in Advanced Internet Development, http://www.tidia.fapesp.br/
- [Tracz88] Will Tracz: RMISE Workshop on Software Reuse Meeting Summary, In Software Reuse: Emerging Technology, IEEE CS (1988)