UMTP モデリングフォーラム 2013

企業情報システムの早期アーキテクティングとモデリング

2013年11月20日

情報システム総研
児玉公信

いきさつ

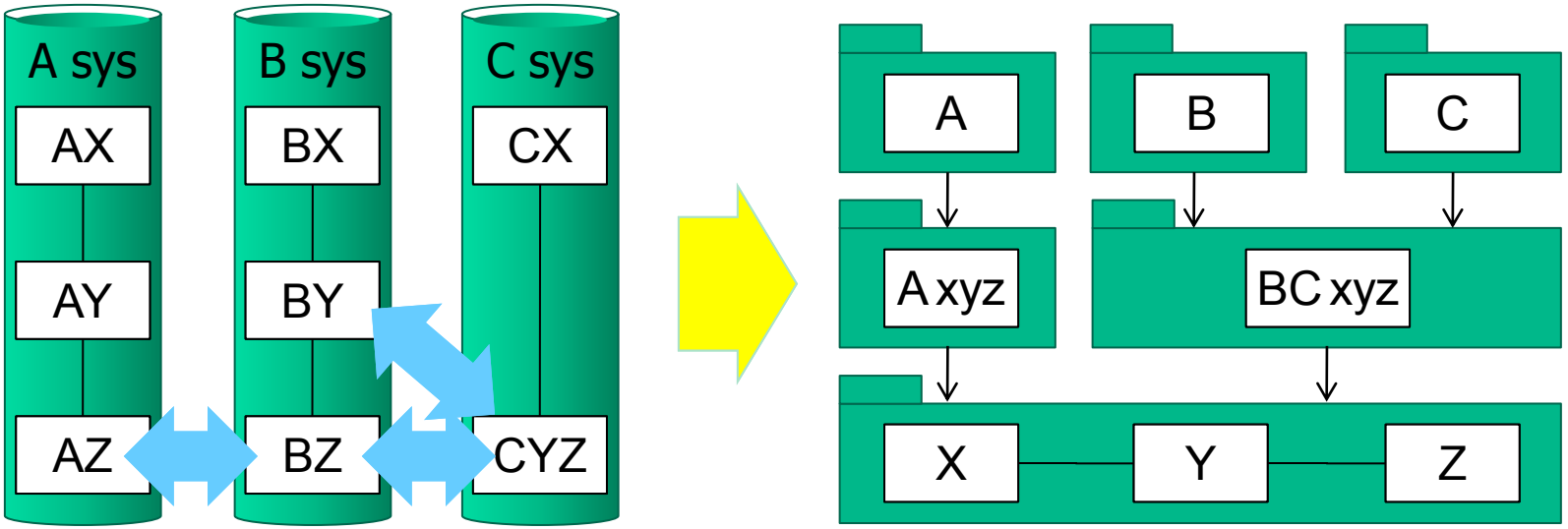
- 去年のUMTPフォーラムのパネルディスカッション
  - ポジション：早期アーキテクティングの考え方
    - システム思考
    - システムライフサイクルプロセス
    - 参照ドメインモデルによる施主要求の揺さぶり
    - Enterprise Systemのアーキテクティング
  - ディスカッション
    - ドメインモデルの磨き方
    - 知識レベルモデルの成り立ち
    - フォークロアモデルを駆逐する
    - モデルの検証
    - モデラの育成
    - モデリングとは何か
      - 組織学習
      - 質的アプローチ
      - “良さ(quality)”の本質

システムとソフトウエア

- JIS X0001 情報処理用語―基本用語
  - ソフトウエア
    - 情報処理システムのプログラム，手続き，規則及び関連文書の全体又は一部分
  - 情報システム
    - 情報処理システムと，これに関連する人的資源，技術的資源，財的資源などの組織上の資源とからなり，情報を提供し配布するもの
- システムとは
  - 「多数の構成要素が有機的な秩序を保ち，同一目的に向かって行動するもの」（JIS Z 8121）
  - 「相互に作用し合う要素の集合」で「オーガナイズされた全体」（ベルタランフィ，「一般システム理論」，1968）
  - 「オブジェクト間の関係を含むオブジェクトの集合（Fall & Fagen, 1968）」だが，「それは観察者の視点・観点に依存する」（ワインバーグ，「一般システム理論入門」，1975）

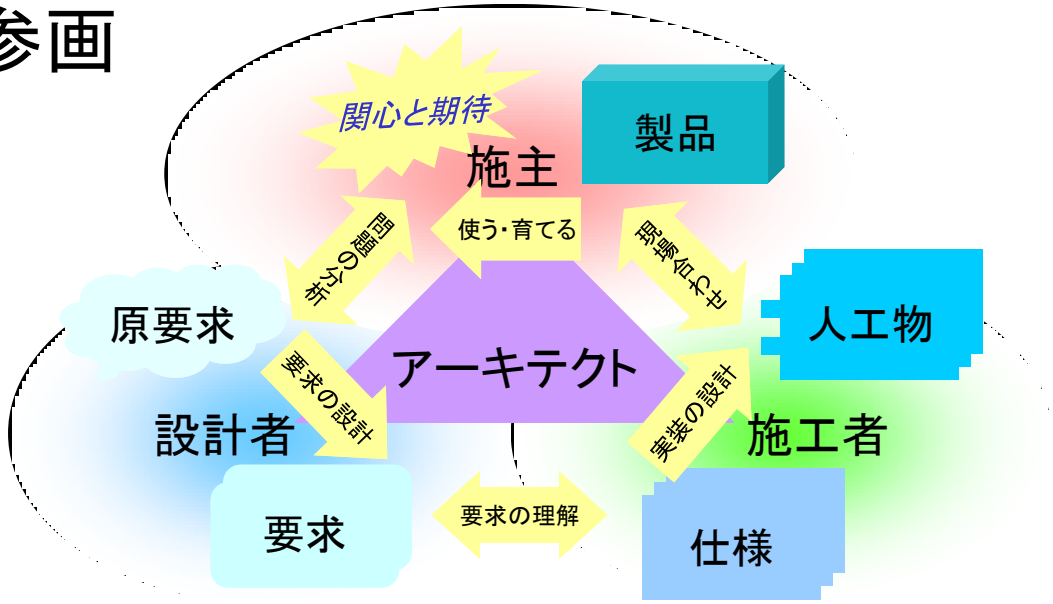
基幹情報システムの再構築

- 煙突システム
  - 企業情報システムとしての連携の阻害
    - 情報の重複・散在，作業のムダ，保守費の増大
    - 情報の脱本質化
- 再構築
  - 複雑さの整理：本質化
  - 堅牢性と変更容易性の追求



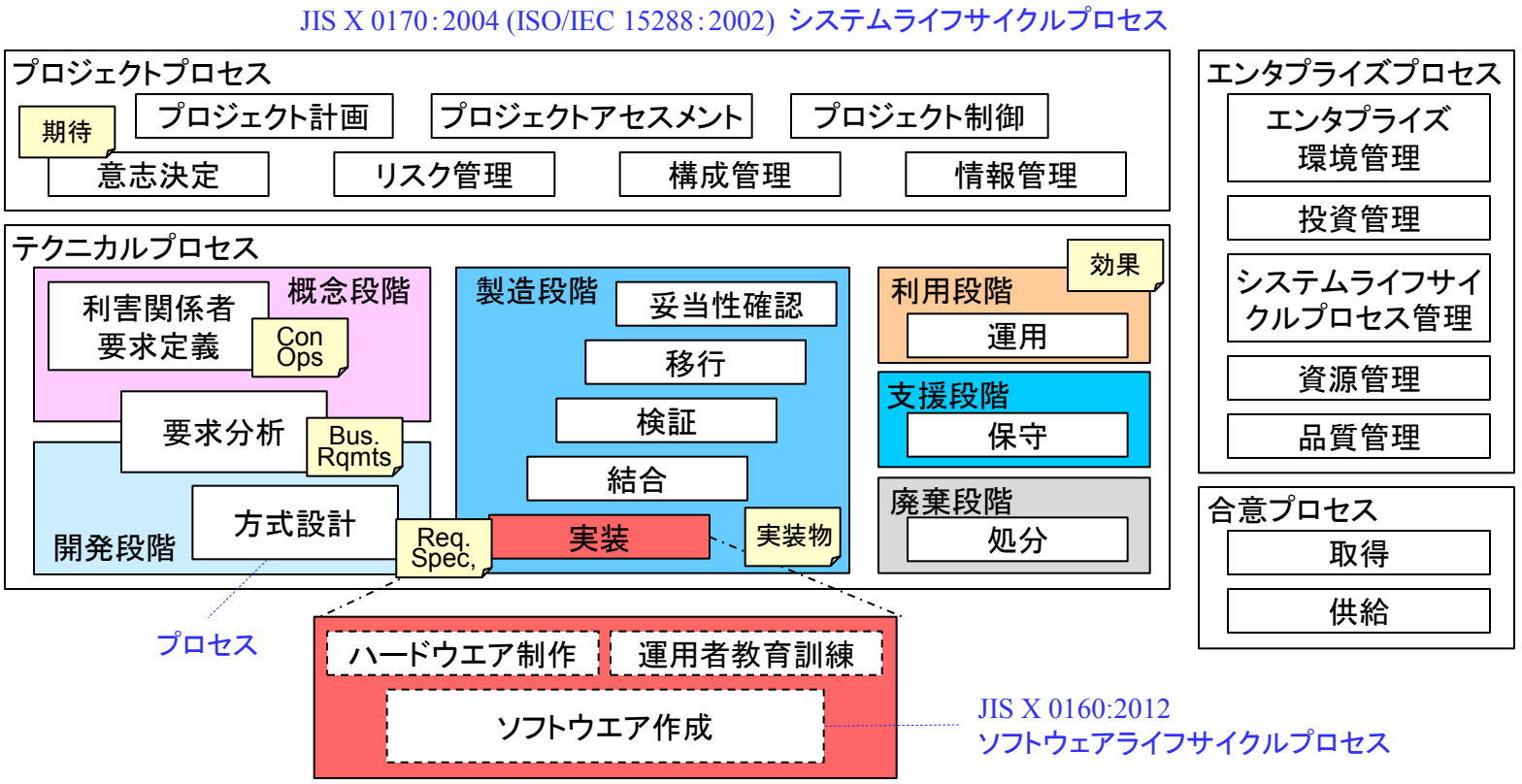
施主主導の情報システム再構築

- 情報システムサイクル
  - “良い”情報システムを目指す継続的な営み
- 基幹システム刷新の真の施主は経営者
  - CIO（施主）チーム
  - 業務設計（設計者）チーム
  - アーキテクチャ（EA）チーム
- プロフェッショナルの参画
  - アーキテクト支援
  - 設計者支援
  - モデラ支援
  - 施工者



システムライフサイクル

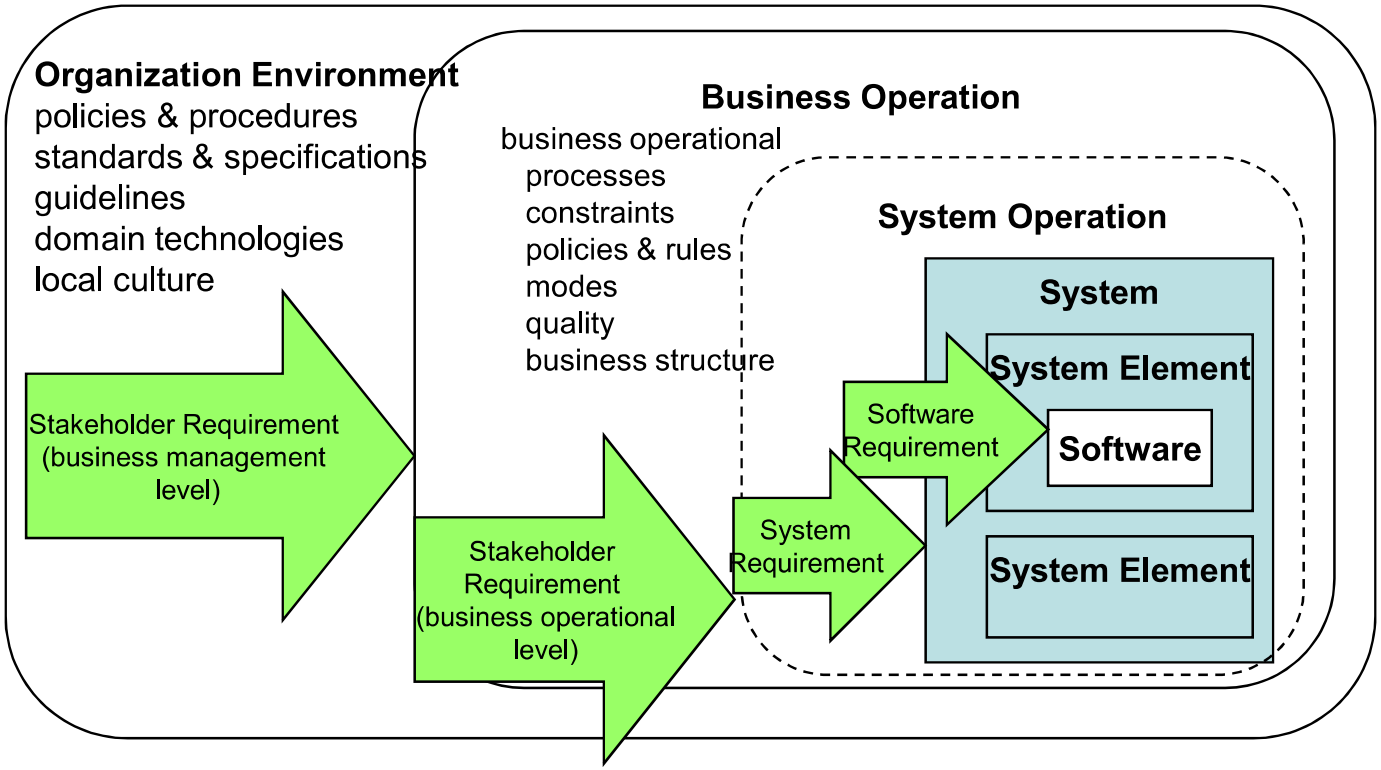
- 二つのライフサイクルプロセス
  - JIS X0170:2004（ISO/IEC 15288:2008）
  - JIS X0160:2012（ISO/IEC 12207:2008）



要求定義

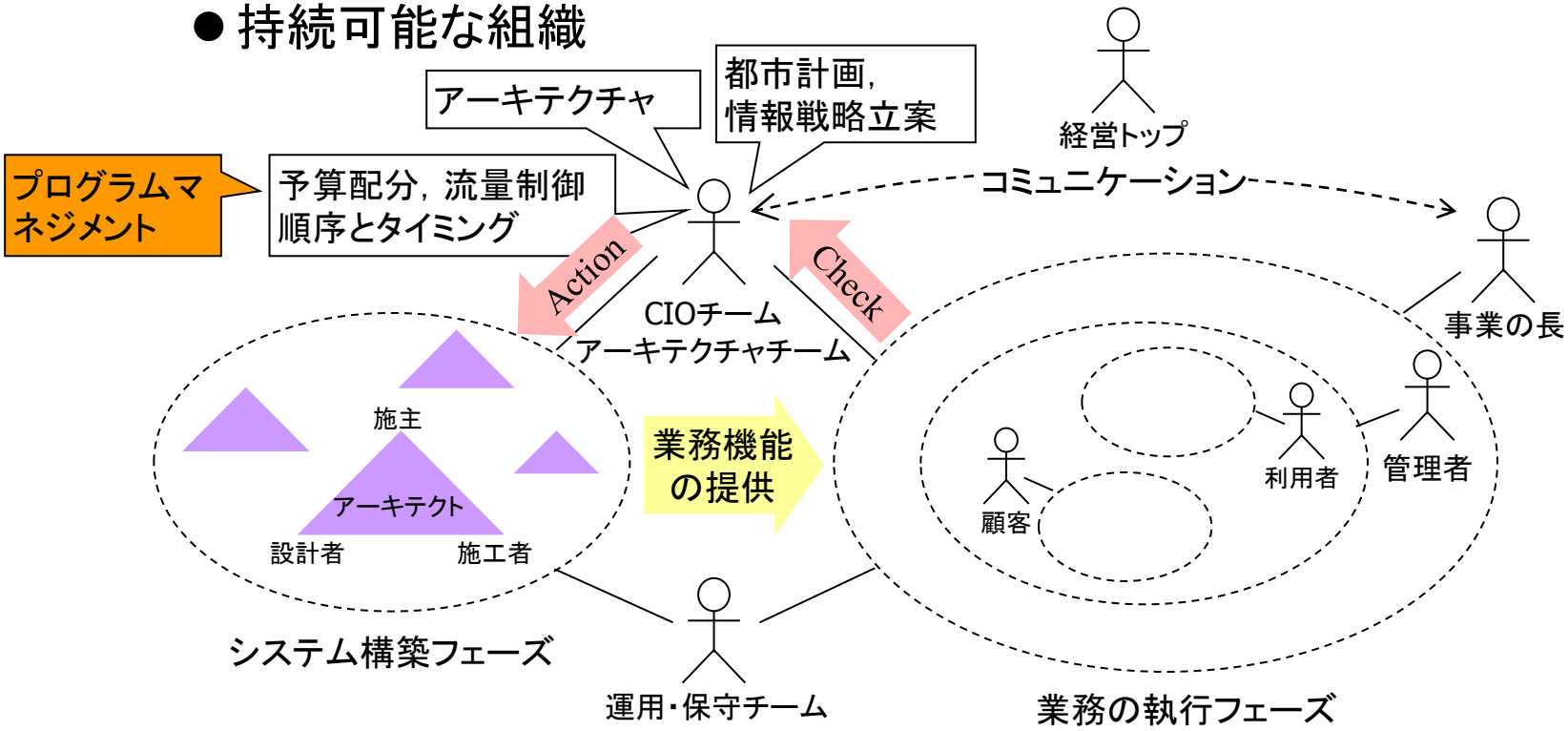
- ISO/IEC/IEEE 29148-2011
  - 要求スコープの例

External Environment
market trends
laws & regulations
legal liabilities
social responsibilities
technology base
labor pool
competing products
standards & specifications
public culture
physical/natural environment



企業情報システムのガバナンス

- CIOの役割
  - 構築フェーズと執行フェーズ
  - 中長期的展望=都市計画
    - アーキテクチャによる統治
    - 全体最適とその指標
    - 持続可能な組織



Zachmanフレームワーク

- 建築学との出会い
  - 断片化された企業情報システムを系統だった分散化に移行する
    - Zachman, “A framework for information systems architecture,” IBM Systems Journal, Vol. 26(3), 1987

ライフサイクル

建築メタファ	ISドキュメント	データの記述	プロセスの記述	ネットワークの記述
バブルチャート	スコープと目標	重要なエンティティの一覧	実施されるプロセスの一覧	営業拠点の一覧
建築家の製図（施主の表現）	ビジネスモデル	ER図	機能フロー図	ロジスティックネットワーク
建築家の計画（建築家の表現）	情報システムのモデル	データモデル	DFD	分散システムアーキテクチャ
生産計画（施工者の表現）	技術のモデル	データ設計	構造チャート	システムアーキテクチャ
個別実施計画（文脈外の表現）	詳細記述	データベーススキーマ	プログラム	ネットワークアーキテクチャ
建物	製品（情報システム）	データ	機能	通信

Zachmanフレームワーク2

- 5W1Hで拡張
  - Enterprise Ontologyへ
    - Sowa & Zachman, “Extending and formalizing the framework for information systems architecture”, IBM Systems J., Vol. 31(3), 1992

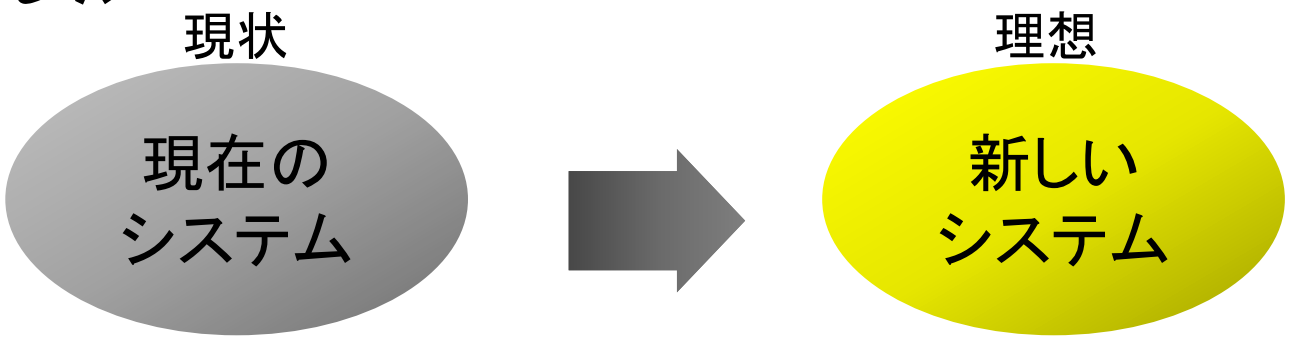
		モデリングの観点					
視座	成果物	データ（what）	機能（how）	ネットワーク（where）	People（who）	Time（when）	Motivation（why）
Planner	スコープ	重要なビジネス事項の一覧	ビジネスプロセスの一覧	営業拠点の一覧	組織の一覧	ビジネス事象の一覧	目標・戦略の一覧
Owner	企業モデル	ER図	プロセスフロー図	ロジスティックネットワーク	組織図	マスタスケジュール	事業計画
Designer	システムモデル	データモデル	DFD	分散システムアーキテクチャ	HIのアーキテクチャ	プロセス構造	知識構造
Builder	技術モデル	データ設計	構造チャート	システムアーキテクチャ	人と技術のインタフェース	制御構造	知識設計
Sub-contractor	コンポーネント	データベーススキーマ	プログラム	ネットワークアーキテクチャ	セキュリティアーキテクチャ	タイミング定義	知識定義
現物	稼働システム	データ	機能	ネットワーク	組織	スケジュール	戦略

要求分析手法の比較

ISO15288	児玉	Wiegers(2006)	Volere(2006)	Maciaszek(2001)
利害関係者要求定義（原要求）	施主 業務構想書(ConOps) 業務シナリオ 設計者 要求関連図		ステークホルダ プロジェクトの方向づけ会議(Blast-off)	
文書化	IEEE Std 1362			
要求分析	設計者 仕事の流れ 「もの・こと」分析 業務フロー図 Customer-Performer	要求アナリスト ビジネス要求 コンテキスト図 ユースケース(ビジネス)	アナリスト 要求のトローリング スノーカード ビジネス事象 ビジネスユースケース	ビジネスアナリスト 要求決定 要求抽出 要求の調整と妥当性 要求依存性行列
	機能の発明 ユースケース(機能)	フィーチャ記述	要求の発明 仕様化	ビジネスモデル コンテクスト図 ユースケース(フィーチャ) ビジネスクラス図
	情報の発見 概念モデル			
文書化			テンプレート	テンプレート
方式設計	アーキテクト 管理階層 階層間インタフェース			
文書化				
要求管理	ファンクションポイント	複数リリース	ファンクションポイント 要求のメタ管理	ファンクションポイント 変更管理，要求追跡

問題解決

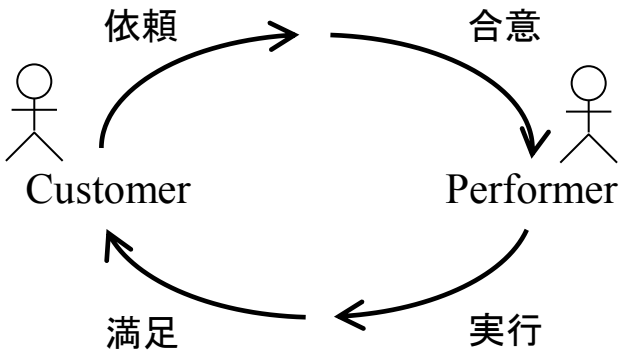
- 問題解決におけるシステムアプローチ
  - 原因除去で問題が解決するとは考えない
    - システムは問題も創発する
    - 問題はシステムの弱いところに表面化する
    - システム内の要素は問題状況に抗えない（せいぜい改善）
    - 全体最適を狙う
  - システムを変えることで問題を転換する
    - 目的を変える
    - 要素間の関わり方（アクション）を変える
    - レベレッジ



問題構造を転換するプロセス

構想−原要求−仕事の設計

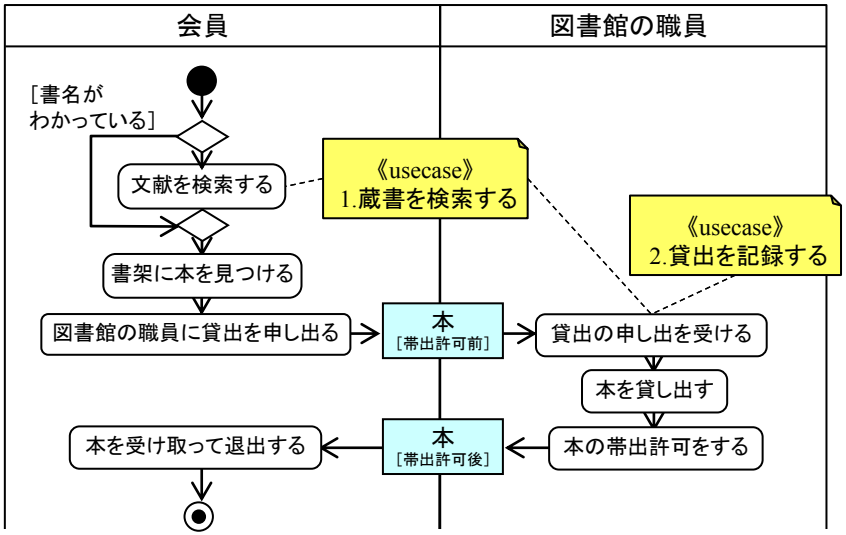
- 業務シナリオによる構想の記述
  - 新しいシステムをありありと描き出す
- 業務シナリオから原要求抽出
  - シナリオ分析
    - M-GTAによる要求の構造
  - 原要求分析：SysMLの要求図
    - 目的，要求，方法を分離
    - 要求の優先づけ
- 要求を満たす仕事の設計
  - プロセス図：「もの・こと」分析
    - 終わりの状態／始めの状態
    - 次元の一致
  - アクティビティ図：Customer-Performer
    - 対話の4象限



Medina-Mora, et al, "The Action Workflow Approach to Workflow Management Technology," Proc. of CSCW 92, pp1-10, 1992

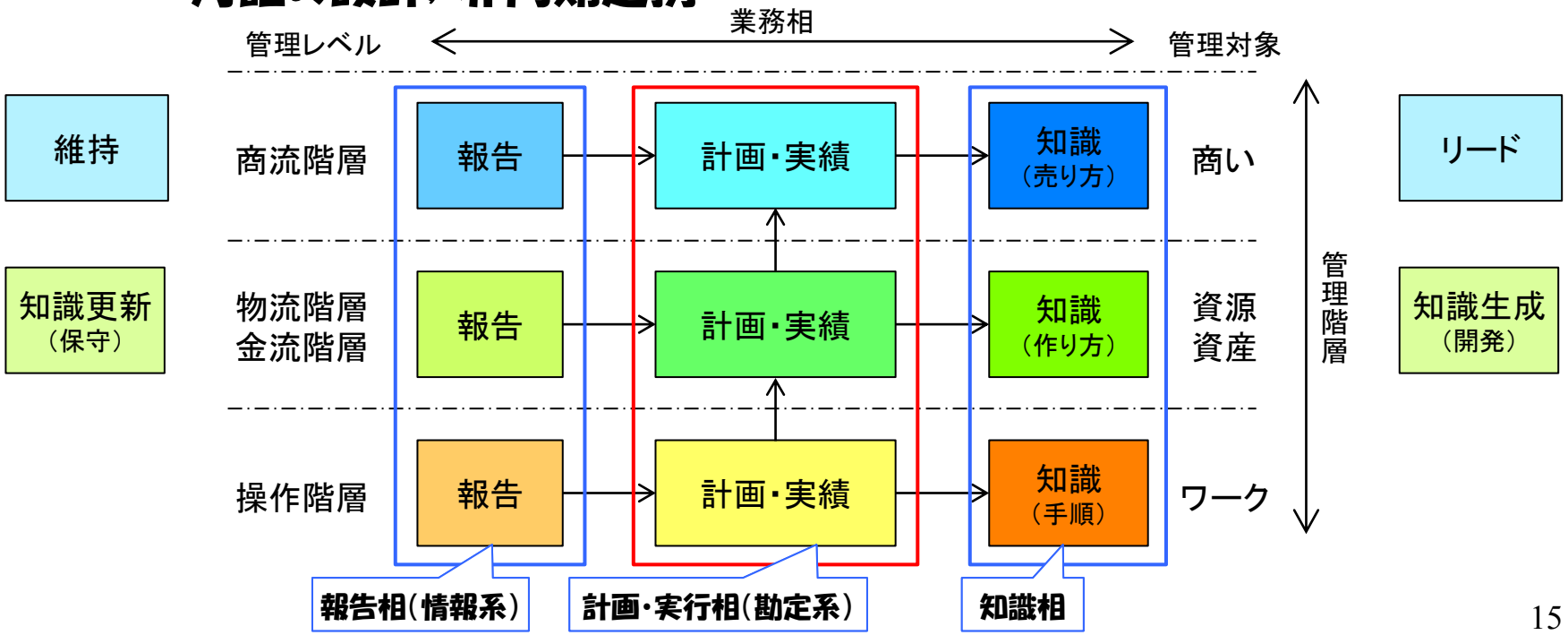
機能の設計−方式設計

- 仕事の達成を支援(enable)する機能の発明
  - アクタがトリガする実世界での基本変換
    - "システム"のレーンは設けない
  - 機能記述：ユースケース記述(要求レベル)
    - ユースケースの目的は，その仕事の達成
    - ユースケースシナリオは，その仕事のインスタンス
- 方式設計
  - Enterprise Architecture
    - 設計原則
    - システム分割の原則
    - 対話の構造
    - 情報システムパタンランゲージ
  - ドメインモデリング
    - 参照モデル
    - 揺さぶり



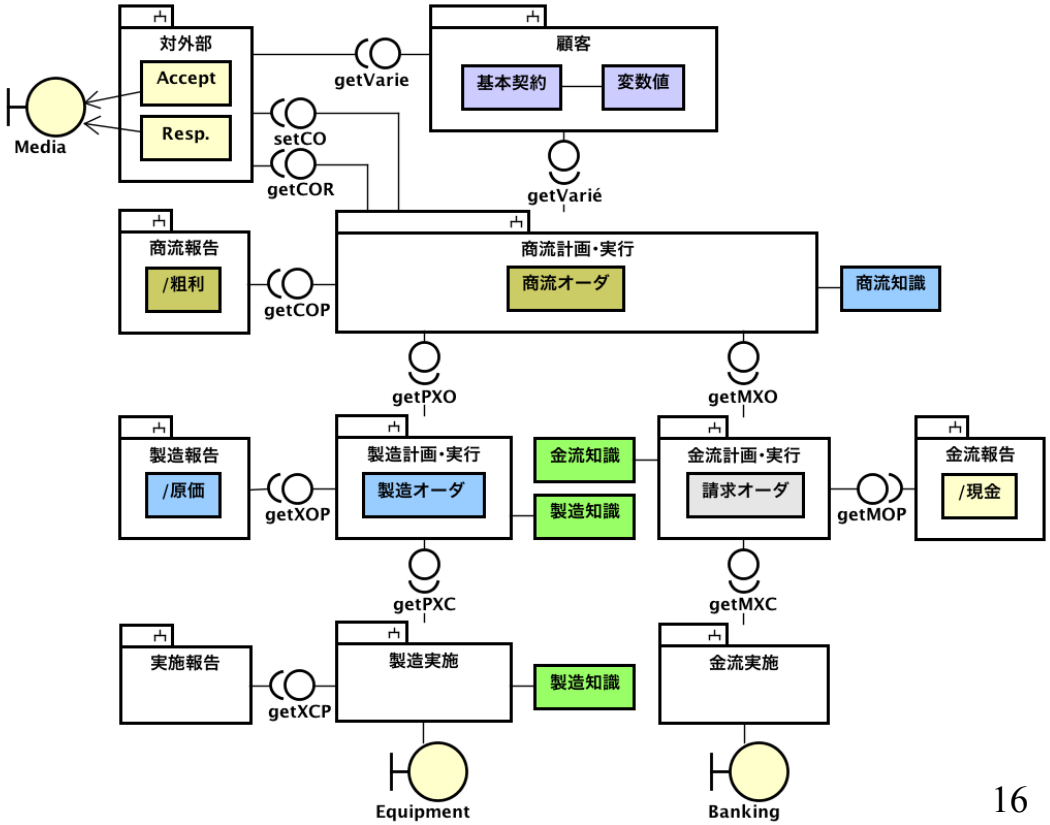
システム分割の指針

- 管理階層と業務相に基づくシステム分割
  - 勘定系ドメインは，関心対象のライフサイクルを管理する
    - セマンティクスの分離：ドメインごとのモデル，アンチ・ユビキタス言語
  - ドメイン間の相互作用
    - 対話の設計，非同期連携



情報処理システムのマクロ構造例

- システム分割指針に沿ってサブシステムを設定
  - ドメインモデルとユースケースのパッケージ
  - よいモジュールの原則
    - 管理階層と業務相(coupling)
    - 機能強度(cohesion)
  - インタフェースに命名
  - 一方向の原則
    - Observerパタン
- 外部複雑性の制御
  - 契約のバリエーション
    - 契約条項の記述
    - ルール形式
  - 商品のバリエーション
  - 製品のバリエーション



インタフェース設計

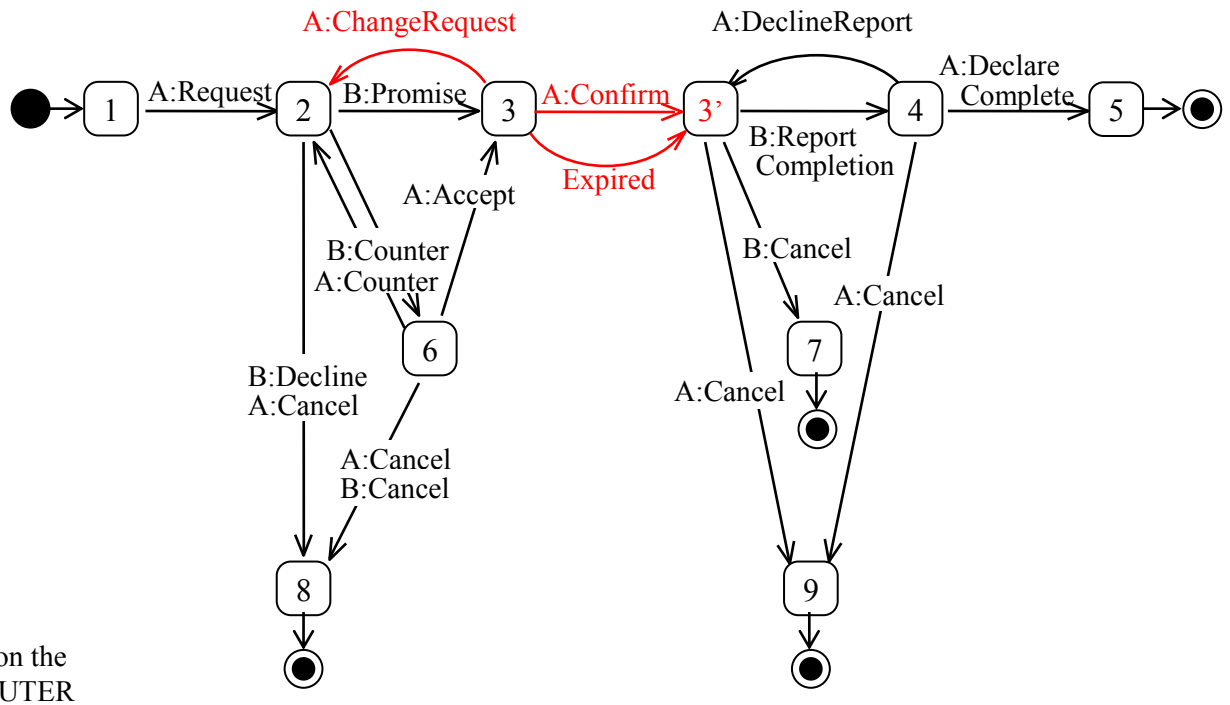
- **対話のモデル**
  - KQML**によるメッセージ設計**→XML/JSON**にコーディング**
  - **疎結合高速メッセージング**
    - ESB(Enterprise Service Bus)

```
setCustomerOrder(
    :verb request
    :message-number nnnnnnnn
    :from 対外部
    :to 案件受付
    :content(
        CO(--CustomerOrder--),
        number-of-records(nn),
        check-sum(nn)
    )
)
```

Finin, Fritzson, McKay, and McEntire: “KQML as an agent communication Language,” Proceeding CIKM '94, p456-463, 1994

対話の構造

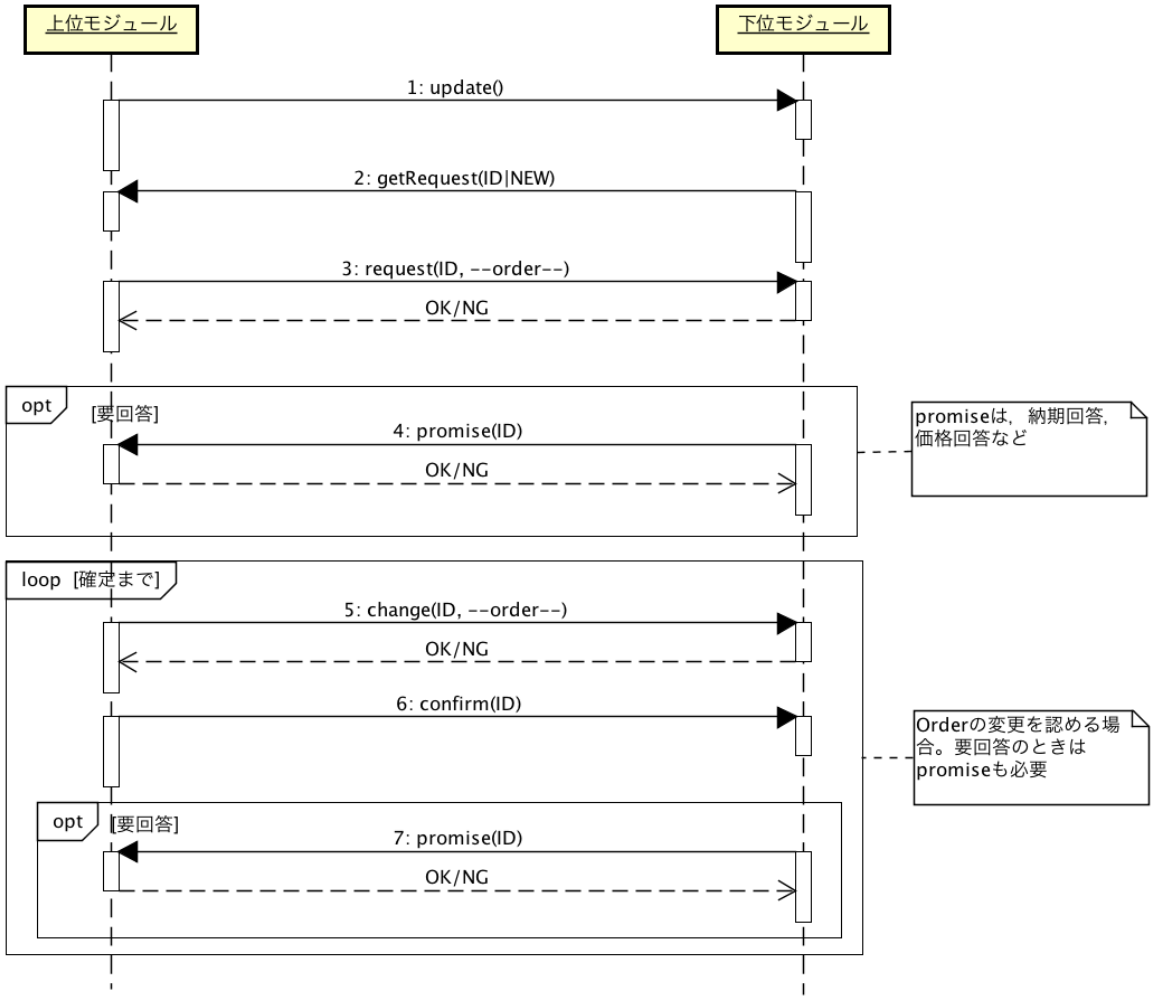
- 状態機械図
  - 拒否，中止，差し戻し
  - 日本型の曖昧なCommitment
    - 1状態と3つの遷移を追加



Winograd, T.: A Language/Action Perspective on the Design of Cooperative Work, HUMAN-COMPUTER INTERACTION, Vol.3, pp.3-30 (1987-1988)

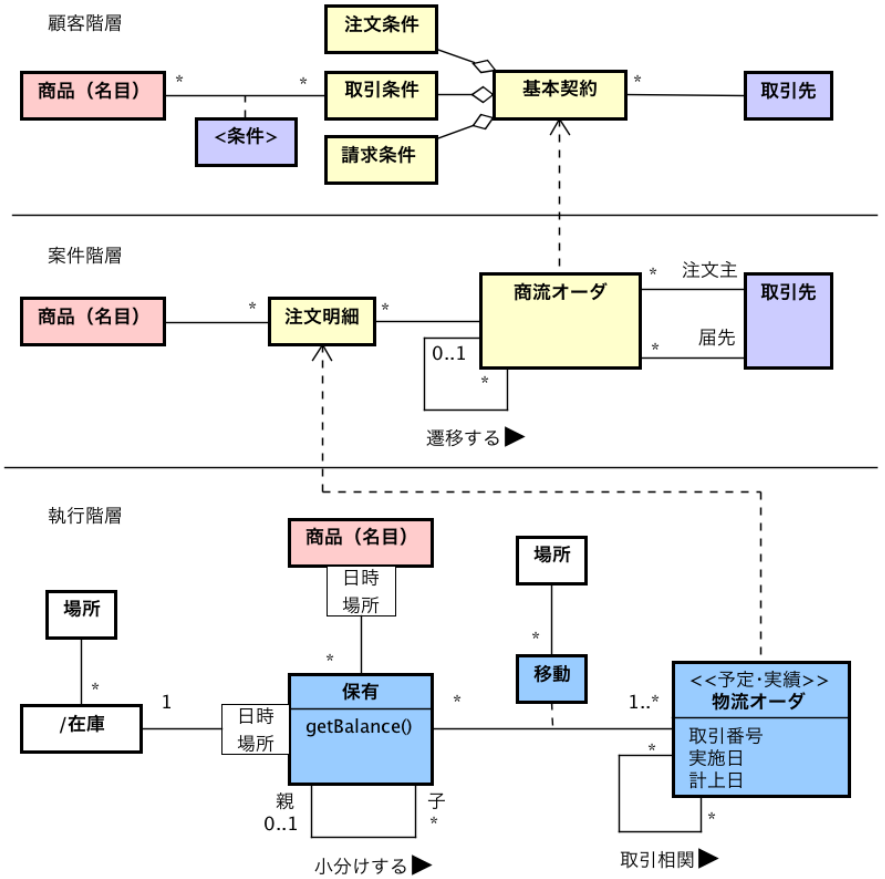
インタフェースのプロトコル

■トランザクションデータの受け渡し



システム間連携のモデル

- 階層間は，Customer-Performerの関係とみなす
  - 一方向で細い参照
  - オーダに対する作業分解
    - 知識による計画展開



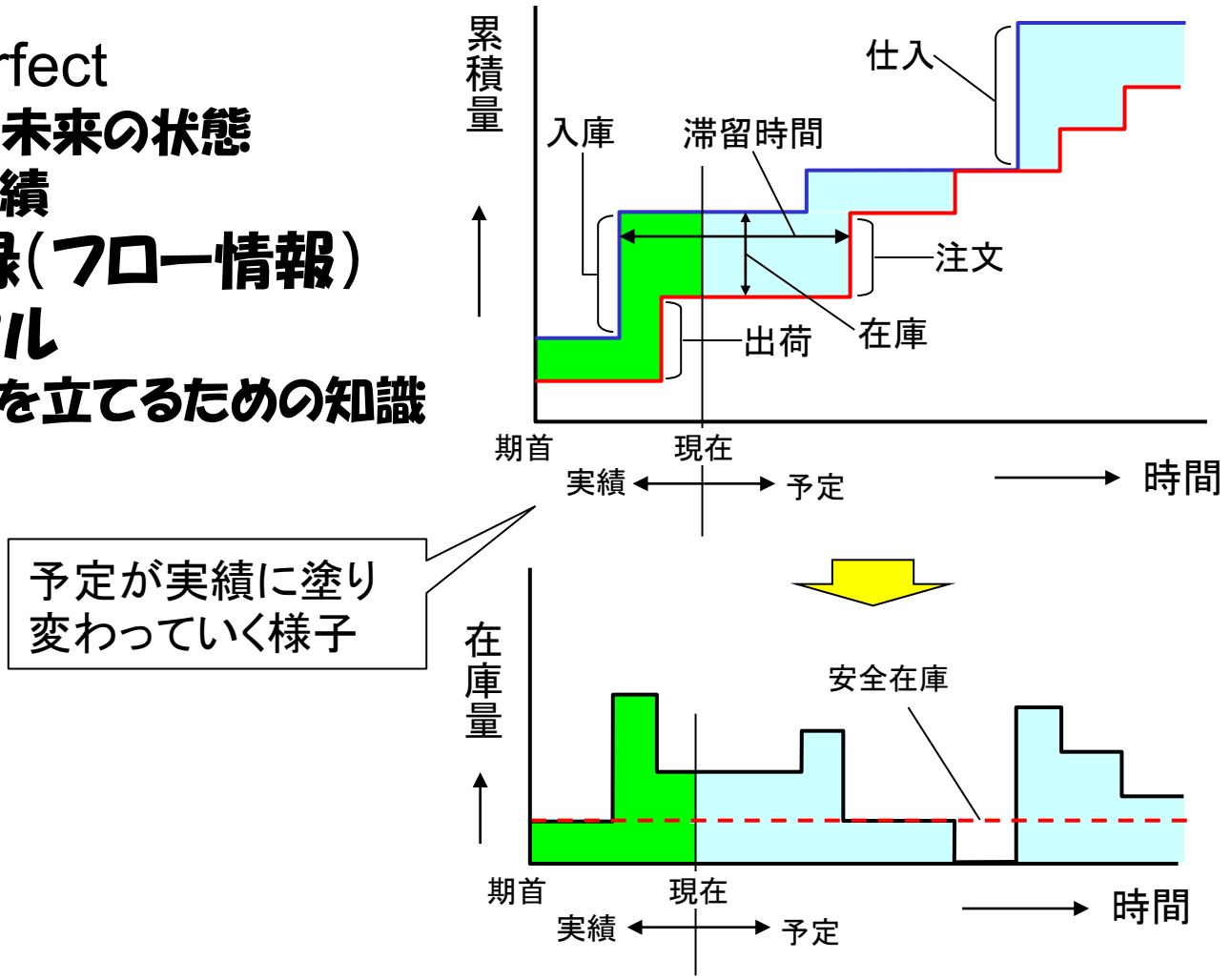
ドメイン駆動

- システムの全体構造の仮説提示
- ドメイン構造の仮説提示
  - 目的，仕事，機能の仮説
  - 蓄積されたドメイン知識

    e.g. 生産管理のto-be，IEC 62264
- ドメインモデルの提示
  - 一般モデル
    - より安定な初期モデル
    - パタンの活用
  - 一般モデル，類似モデルがある場合
    - 構想を支援する高速プロトタイピング
    - たたき台
  - 参照モデルがない場合
    - ユースケースからモデリング

勘定系ドメインモデルの要点

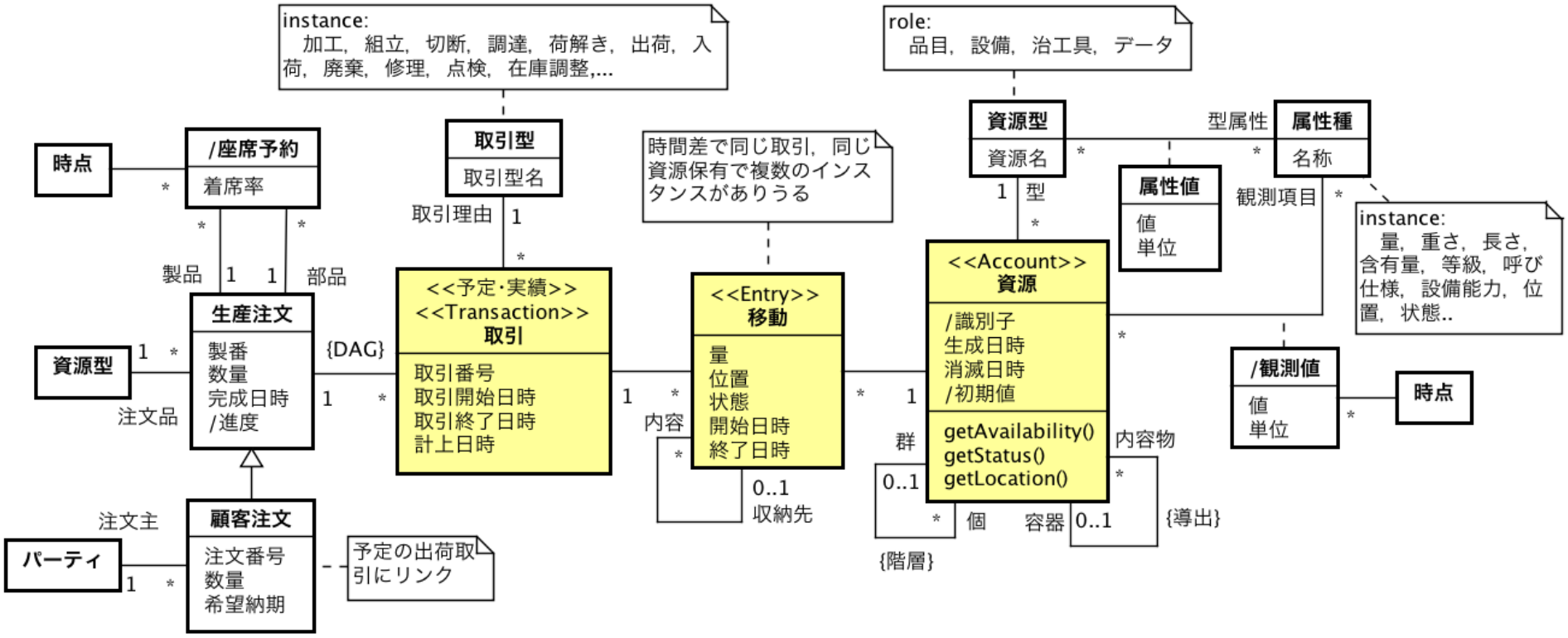
- **時間の征服**
  - Future Perfect
    - **資源の近未来の状態**
    - **計画と実績**
  - **事象の記録（フロー情報）**
  - **学習サイクル**
    - **よい計画を立てるための知識**
    - PDCA



ドメインの一般モデル

■ 生産管理ドメインの一般モデル(CHARM3)

Cross Hierarchical Account=Resource Model v.3



児玉公信，水野忠則，「少量多品種型生産管理システムの一般モデルCHARMの提案」，
情報処理学会論文誌，Vol. 49(2), 902-909, 2008

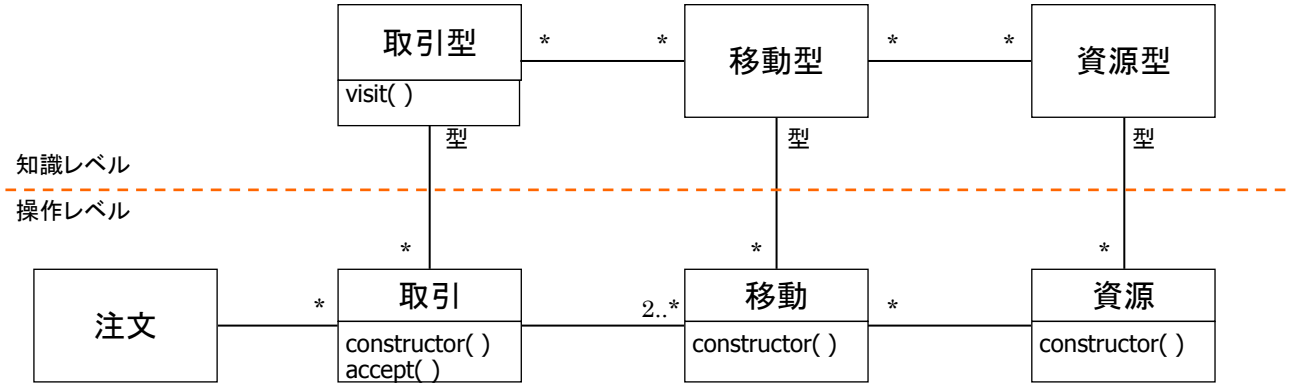
操作レベルと知識レベル

- **生成ルールの外部化**（Externalize Instantiation Rule）
  - **問題**

    インスタンス生成時の生成ルール（知識）が各クラスに散在するが変更も多く，ルールの総合的な意味が分からなくなる
  - **解決策**

    生成ルールをクラスから分離し，知識レベルに集約する

    操作レベルのclientは，知識レベルに複数のインスタンス生成を委託する。知識レベルは生成ルールに従って操作レベルの各クラスのConstructorを順にCallbackする（たとえばVisitorパタン）



操作階層での計画・実行相

■ 空間的距離の対策

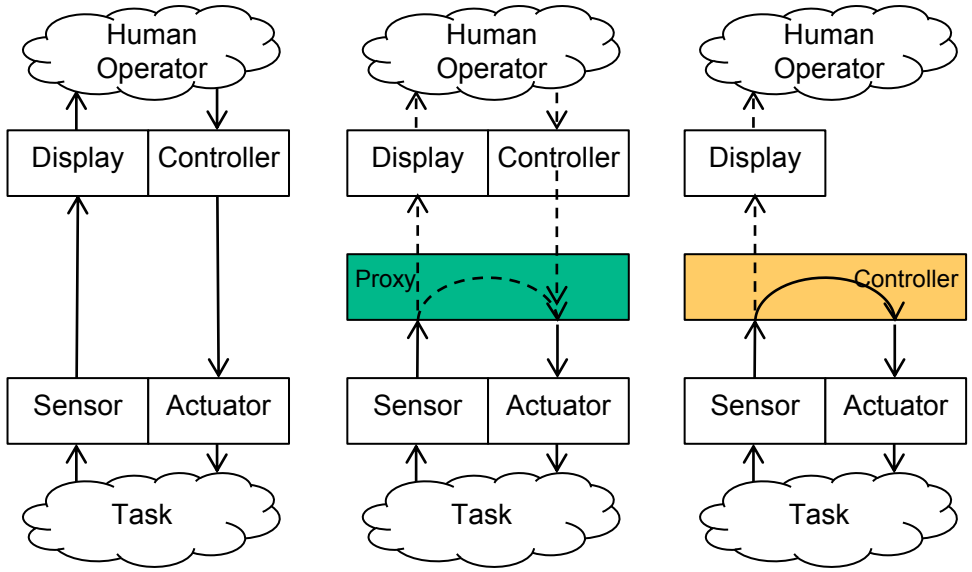
■ Supervisory Control

- **月面車を地球から安全に操作する**

  月面で判断・実行するものと，地球が判断・指示するものをリスクに応じて分割

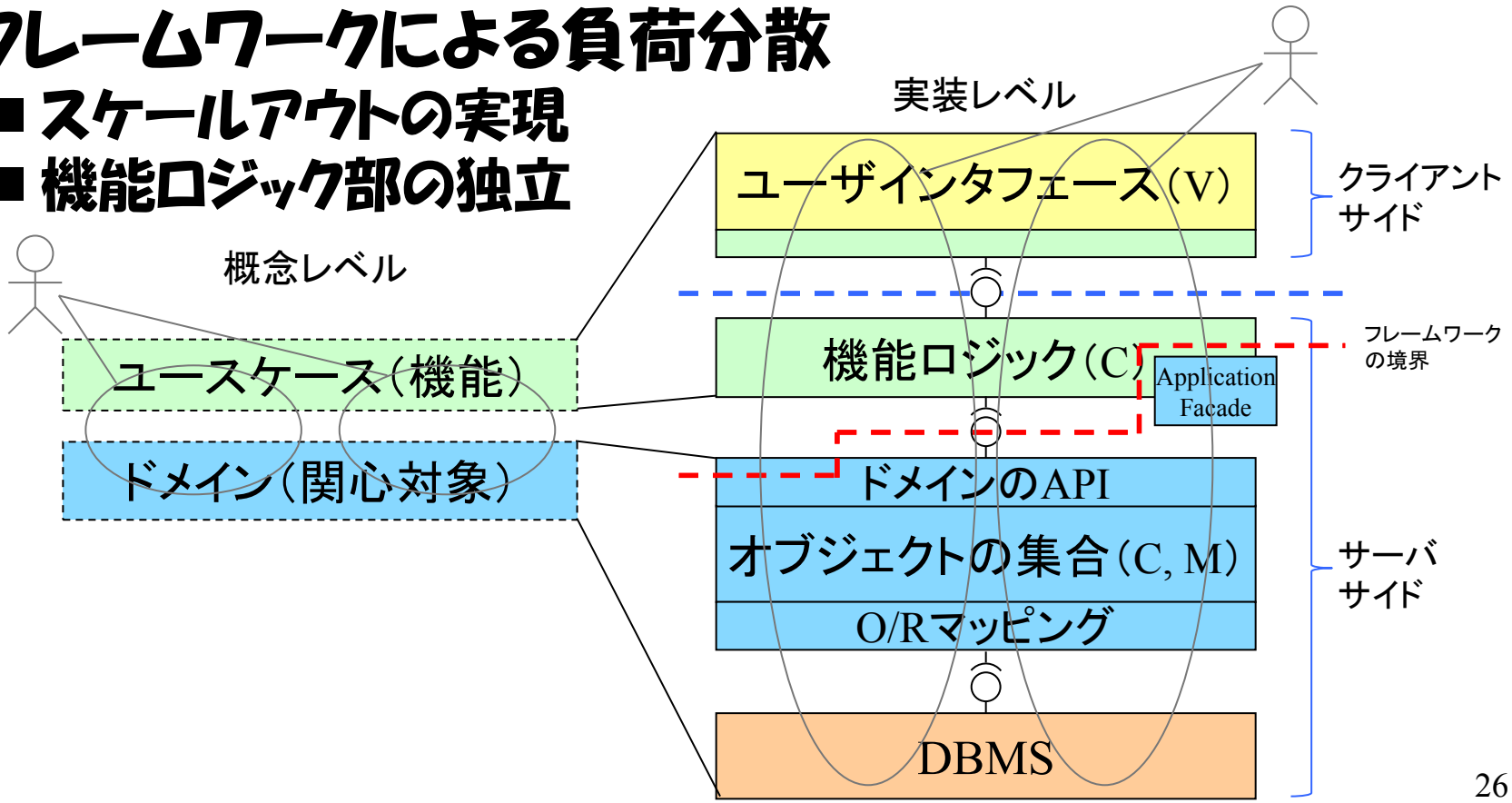
■ MES(製造実行システム)

- **対上位：物流管理層の**Proxy**または**cache
- **対下位：機器の操作**



マイクロアーキテクチャ

- ユースケース実現のアーキテクチャ
  - 階層化とMVCパタン
  - 必要に応じてApplication Façade(導出表の提供)
- フレームワークによる負荷分散
  - スケールアウトの実現
  - 機能ロジック部の独立

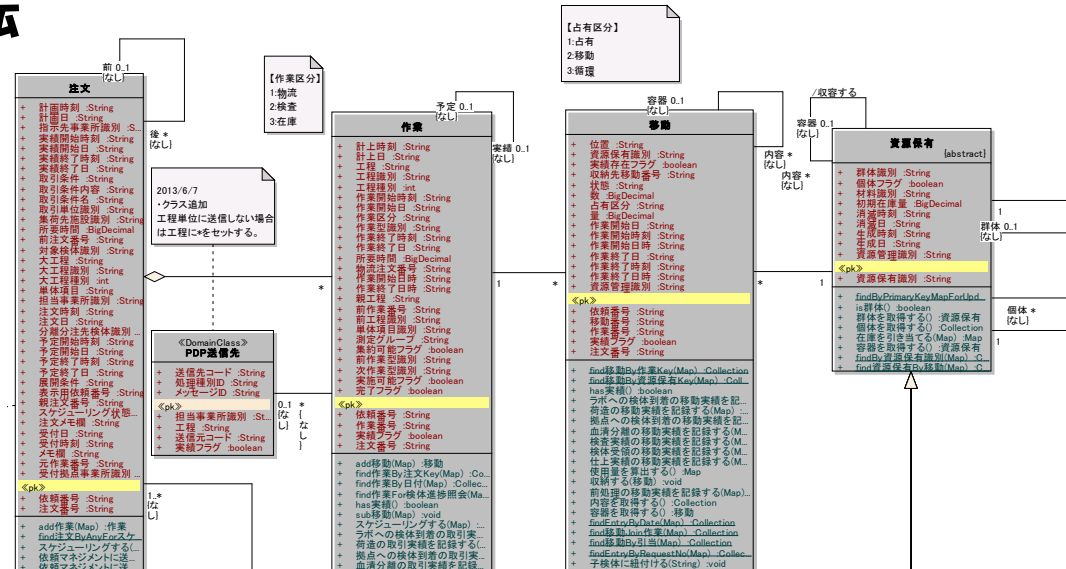


ドメインの先行実装

- 早期アーキテクティング
- 要求定義前に概念モデルがある
  - できれば実装もあるとよい
    - 動かしてみる
    - モデルの理解
  - フレームワークの前提
    - クラウド利用
    - オートスケールアウト
    - バックアップ系の完備
- 概念モデルを参照しての要求定義
  - 想定ビジネスプロセス，ユースケース
  - 業務シナリオによる概念モデルの揺さぶり
  - 実ビジネスプロセス，ユースケースのあぶり出し

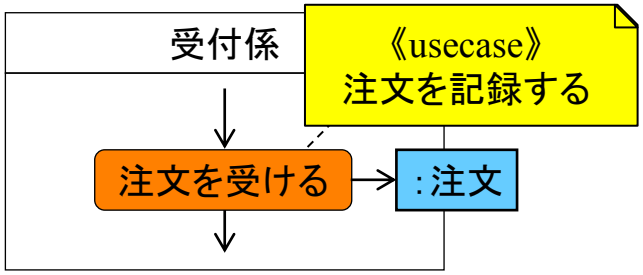
実装モデリング

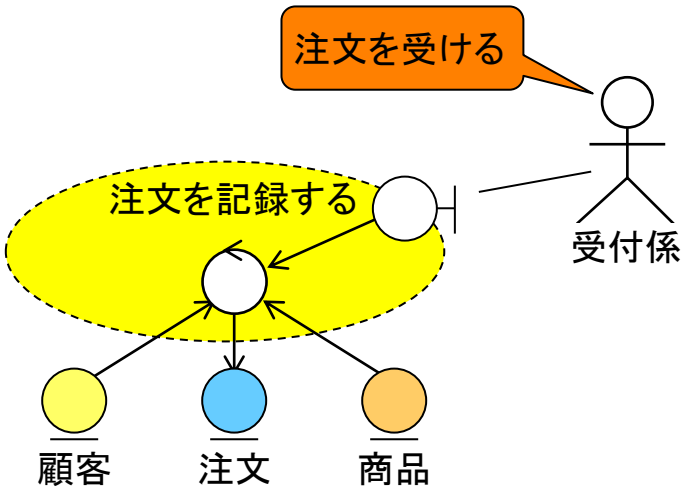
- 型からクラスへ
  - 概念モデルに従う
  - 設計判断
    - オブジェクトの識別方法：OIDとコードの併用
    - 候補キー（ユニーク性）の解釈，参照の解決，データ型の決定
    - 型の内部属性化，下位型の実装
  - メソッドの配置
    - インタフェースの設計：Design by Contract
    - APIの設計，制御の逆転
    - ドメインリッチ化
  - ORマッピング
  - 非機能要求の作込み
    - 性能，安全性
    - トランザクション管理



ユースケースと型の関係確認

- 型と機能の整合性をチェックする
  - 機能の漏れを見つける
    - オブジェクト生成機能（CRUDの完成）
- DFDまたはBCE図
  - DFDはレベル0，分解しない
  - または（入出力の方向を明示した）BCE図
    - 1プロセス（コントロールオブジェクト）に1アクタ
    - 1プロセスに1以上のデータストア（エンティティオブジェクト=型）
    - プロセス間のデータフローは禁止





設計原理の適用

- パタンランゲージとEnterprise Architecture
  - 設計原則の記述形式としてのパタン
  - EAの運営
    - 計画評議会=アーキテクチャチーム
    - 設計原則(パタン)の維持，遵守監視
  - 学習サイクル