ビジネスモデリングとパターン

2002/08/19

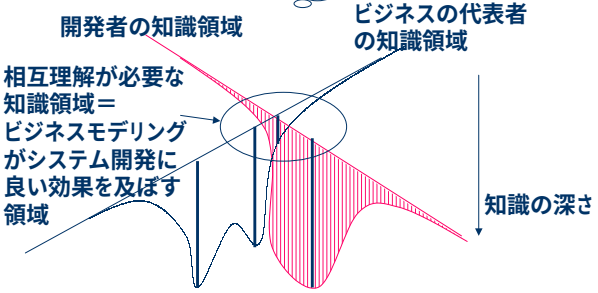
有限会社エス･ラグーン
中谷　多哉子

要求仕様は誰が書くのか

- ビジネス代表者が書く
  - 要求仕様書は不適切／不完全であることも多い
  - ユーザインタフェースの記述に専念してしまう傾向
- 仕様書を設計の専門家が書く
  - ビジネスやビジネスニーズを十分理解できない
  - 情報を構築する「方法」を決定することに専念する傾向
    - これは時期尚早，意思決定が，本質的な要求の性質や機能と直接矛盾する可能性

模式的に描くと．．．

ビジネスモデリングにはビジネスと開発という二面性がある



開発者が考えるべきこと

- どのような情報システムが求められているか
  - これは多くの場合把握されている
- いかにして情報システムを開発すべきか
  - 開発者はこれを考える専門家である
- システムが提供すべき機能は，ビジネスの全体像から決定する
  - 開発者だけでは，これはできない

開発者がやるべきこと

- ビジネスで使われるオブジェクトの抽出
- オブジェクトの役割の定義
- オブジェクト間の関係の定義
- オブジェクト間の相互作用の定義
- ビジネスの効率的／効果的な支援

ビジネスの理解は必須ではないのか？

結論

- 要求仕様書は良いビジネスモデルに基づいて記述されるべきである
- 要求仕様書がビジネスモデルに基づいて記述されないとしたら，情報システムがビジネスを適切に支援することはできないであろう

ビジネスの理解は必須である

ビジネスモデリング

ビジネスとは

- 効果的な組織
- 効率的な組織の運営
- 各組織での効率的な業務の遂行
- 資源の保有，利用，活用
- 一つ以上の目的
  - 利益，改善，支援（含：非営利活動）
- **情報システムへの要求を定義する主体**

ビジネスモデリングに対する動機

- ビジネスのキーを握る機構をもっと理解する
- 情報システムを支援する拠り所とする
- ビジネスを改善する／完全に変える
- 新ビジネスの機会を見極める
- アウトソーシングする機会を見極める

開発者側の意図

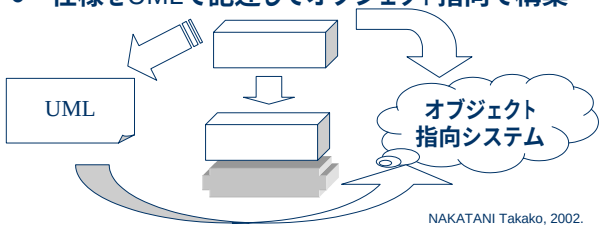
- ソフトウェアの再利用の機会を示す
- 同じビジネスモデルが複数の情報システムの拠り所となるならば
  - 個々のシステムの要求を定義する拠り所の入力として再利用可能
  - 互換性のあるシステムの開発

ビジネスモデルに従来型システムを描く方法

- 人のように、従来型システムを単一の実体としてモデル化
- リバースエンジニアリング

従来型システムの取り扱い

- 既存のシステムをモデリングする
- コンポーネントとしてカプセル化する
- 仕様をUMLで記述してオブジェクト指向で構築



モデリングのために必要な技術的知識とは

パターンとは

- 発生しやすい問題と，それを解決するための手段とを関連づけて文書化したもの（ノウハウ）
- 特定の領域で発生しやすい問題に対するパターンをまとめたものをパターンランゲージと呼ぶ

ノウハウとは？

- ノウハウ＝暗黙知
- 特定の状況のもとで，問題が発生したとき，（最悪ではない）策を適用して，問題を解決する知恵

ノウハウの文書化による知恵の共有は可能か？

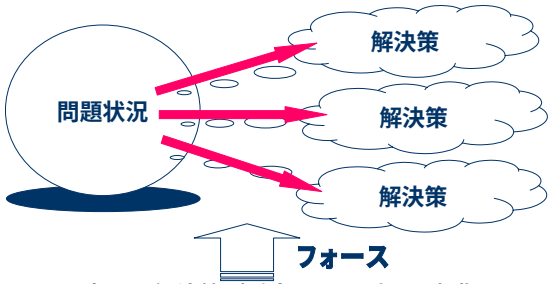
パターンの例

- アンチパターン：プロジェクト管理など失敗事例
- アナリシスパターン：分析／モデリング
- デザインパターン：設計
- イディオム：プログラミング（言語依存）
- ソフトウェアアーキテクチャ

ノウハウの文書化（パターン化）

- ノウハウの共有が目的
- 文書化の鍵となる項目
  - 発生しやすい問題
  - 解決策選択の根拠：フォース
  - 解決策
  - 解決策を適用したときに得られる効果
  - 保証：3つ以上の適用事例

フォースとは



選択する解決策が妥当であると考える根拠

パターン文書化のためのテンプレート

- 目的，別名，動機
  - なぜ定義したか，適用する目的
- 解決できる問題，適用可能性
  - どのような問題を解決できるか
- 構造，構成要素，協調関係
  - クラス図を用いた解説
- 結果，実装，サンプルコード
- 使用例，関連するパターン
  - 新たに発生する問題を解決できるパターン
  - そのパターンを組み合わせて使用することが多いパターン

＋フォース

資源とルールのパターン

- 資源の有効利用，活用，管理
- 資源
  - 人，組織と役割との関係
  - ものとものに関する情報との関係：静的情報
  - 情報の履歴管理：動的情報

資源とルールのパターン：人と組織と役割

- Actor-Role：コンテキストと主体と役割
- Contract：企業間契約
- Employment：雇用契約
- Organization and Party：組織構造

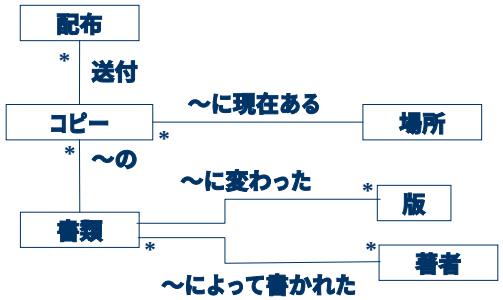
資源とルールのパターン：静的情報の管理

- Business Definitions：ビジネスを定義するときの用語の定義と使用法
- Core-Representation：ビジネス取り扱い対象とエイリアスの関連づけ
- Geographic Location：地理情報の意味づけ
- Thing-Information：管理対象と管理対象の属性情報の管理／把握／保持
- Title-Item：管理対象物固有の属性情報とそれを製品として説明するための属性情報とを区別する

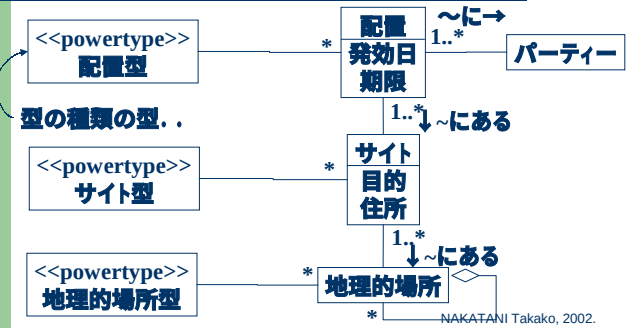
資源とルールのパターン：動的情報の管理

- Business Event-Result History：ビジネス上の意思決定とその結果の履歴管理
- Document：文書管理（所在，原本，コピーの管理）
- Product Data Management(PDM)：製品，製品の分類，製品から生成されたデータ，データの分類の関連づけ
- Type-Object-Value：実体と，それがモデル化されて定義された複数のオブジェクトとを識別する

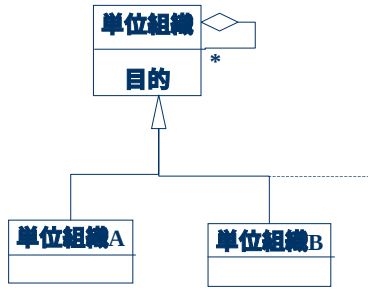
Document



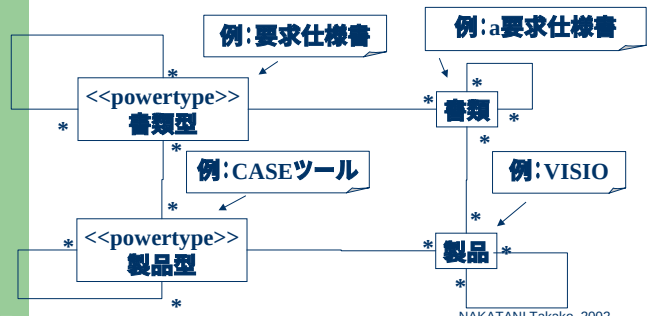
Geographic Location



Organization and Party



Product Data Management(PDM)



ゴールをモデル化するためのパターン

- Business Goal Allocation：プロセスにゴールを割り当て，ゴールをプロセスの達成目標として示す．
- Business Goal Decomposition：ゴール分割／詳細化
- Business Goal-Problem：問題と対処，対処が有効なコンテキストの関係を示す

プロセスパターン

- 定型的なビジネスプロセスのモデル化
  - 資源を消費し，他のプロセスを制御し，依存関係を持つ
- プロセスを改善するための基本構造
  - 評価プロセスの導入
  - フィードバックプロセスの取り込み
  - 定型的な改善活動の採用
- プロセスのモデルの再利用／拡張のための抽象化

プロセスパターン：プロセスの構造

- Basic Process Structure：基本的なビジネスプロセスの形
- Resource Use：実行されるプロセスと，そこで使用される資源との関係を表す
- Process Interaction：プロセス間の相互作用を表す
- Process Layer Supply：活動によって生成される資源を介した依存関係
- Process Layer Control：活動の制御関係

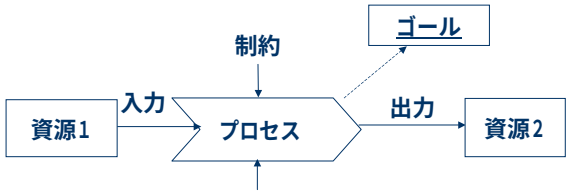
プロセスパターン：プロセスの改善

- ✓ Time-To-Customer：サービスを提供するまでに要する時間を短縮する（ビジネス遂行のノウハウ）
- Process Feedback：プロセスの成果をフィードバックさせる
- Action Workflow：活動の定型サイクル

プロセスパターン：プロセスの抽象化

- Process-Process Instance：定義されたプロセスと実行されたプロセスを識別する
- Process Instance State：プロセスインスタンスの状態（繰り返し等）と各状態で変わる活動や対処との関係を抽象化を用いて整理したもの

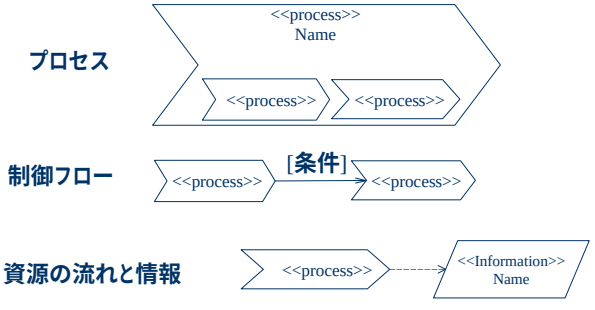
Basic Process Structure：基本的なビジネスプロセスの形



33

NAKATANI Takako, 2002.

Process Interaction：プロセス間の相互作用を表す

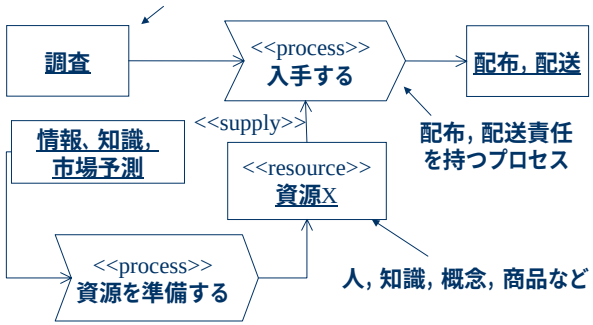


34

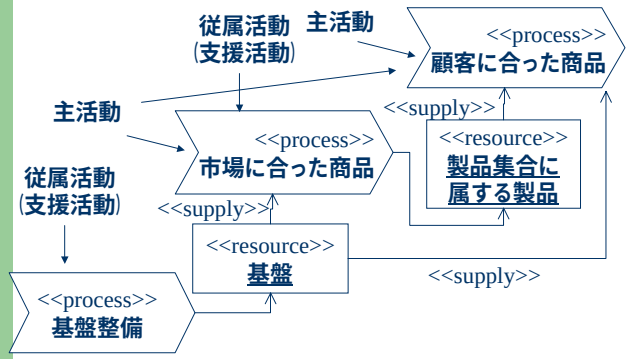
NAKATANI Takako, 2002.

Time-To-Customer:リードタイム短縮

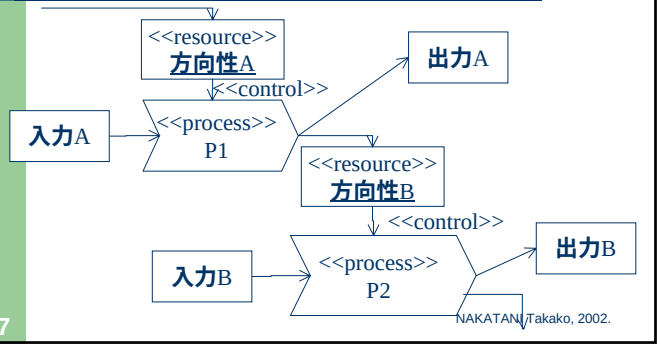
プロセスを起動させる役割を持つオブジェクト



Process Layer Supply:
主活動と従属活動の階層



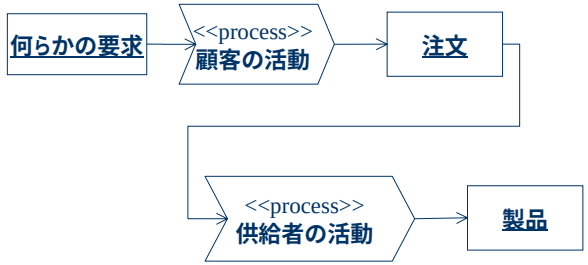
Process Layer Control :階層型制御機構



NAKATANI Takako, 2002.

37

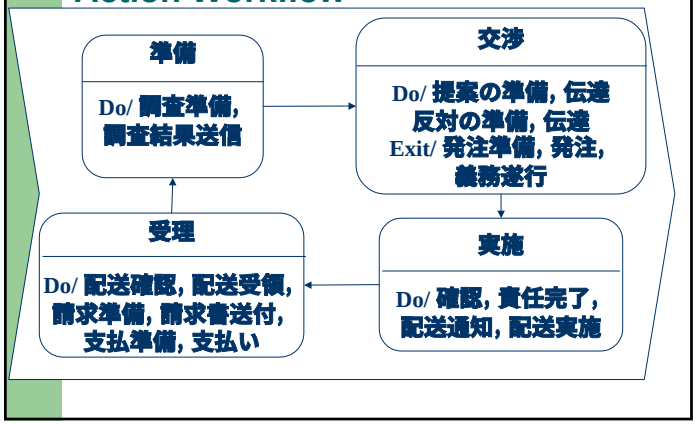
Action Workflowの例



NAKATANI Takako, 2002.

38

Action Workflow



オブジェクト指向システム開発に向けて

要求分析とビジネスモデリング

- 現状：「要求の根拠」が明確でない
  - 要求には，ビジネスを成功させる必然性がある
- 機能的要求：
  - 欲しい機能！
  - なければ困る機能！
  - 業務の自動化ならば，業務作業自体の機能が必要．
- 非機能的要求
  - 目的の情報が得られればそれでビジネスが成り立つわけではない

非機能的要求：性能(performance)

- “どの程度「良く」機能するか”
  - 効率：Efficiency
    - どの程度資源を上手に使うか
  - 完全性：Integrity
    - どれくらい安全か（問題を起こさないか）
  - 信頼性：Reliability
    - どのくらいその機能が信頼／信用できるか
  - 稼働性：Survivability
    - 予期せぬ事態が発生したとき，どのくらい正しく動くか
  - 使いやすさ：Usability
    - どのくらい使いやすいか

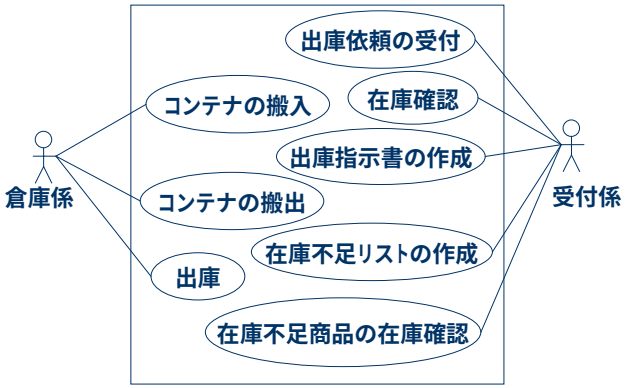
非機能的要求：設計(design)

- “どれくらい正しく設計されているか”
  - 正確性：Correctness
    - どのくらい定義された要求と一致しているか
  - 保守性：Maintainability
    - 修正／保守がどのくらい容易か
  - 検証可能性：Verifiability
    - その動きを検証することがどのくらい容易か

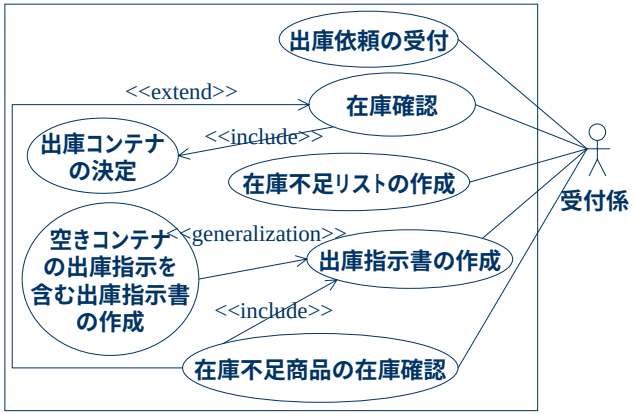
非機能的要求：適応可能性(adaptability)

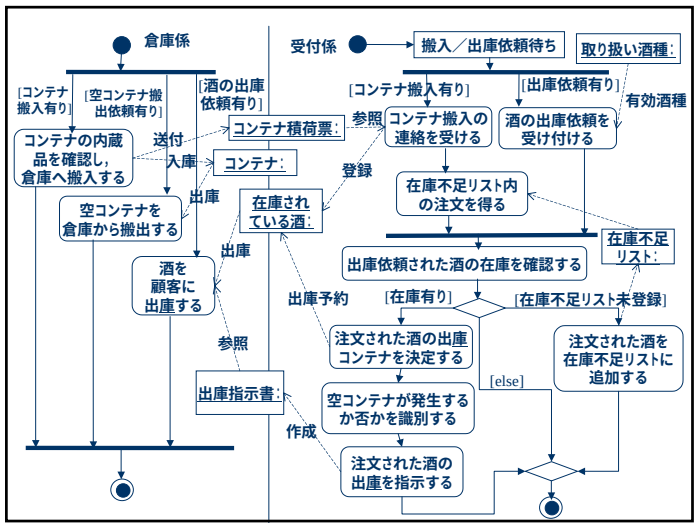
- “どれくらい適応しやすいか”
  - 拡張可能性：Expandability
    - その動きや可能性をどのくらい容易に拡張／更新できるか
  - 柔軟性：Flexibility
    - 変更がどのくらい容易か
  - 相互運用性：Interoperability
    - どのくらい他のシステムに問題を起こしやすいか
  - 可搬性：Portability
    - どのくらい容易に他へ移行可能か
  - 再利用性：Reusability
    - 他のアプリケーションで使うために変換することがどのくらい容易か

従来型要求定義の始まり

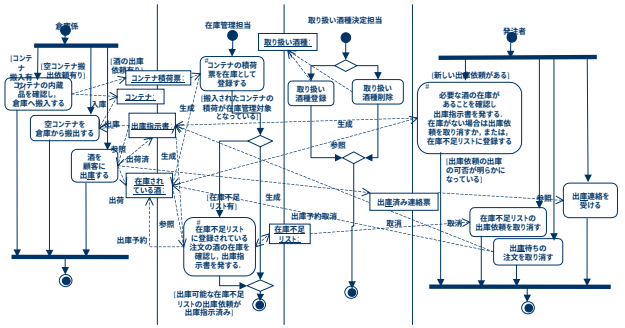


UMLによるユースケース図





ビジネスと関連づけた修正版

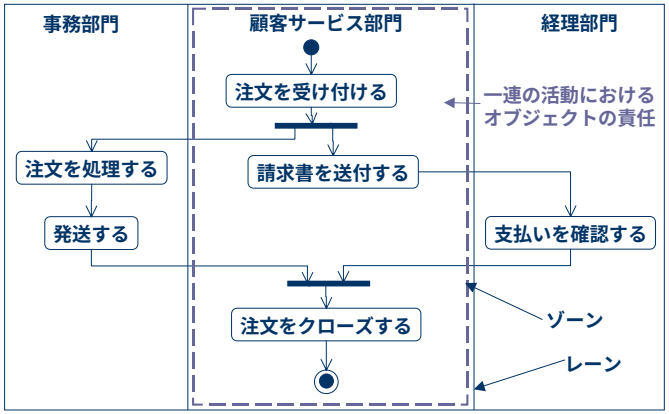


アクティビティ図 (UML)

- ～プロセスモデルの代用として使える．
- 差分
  - プロセスモデル：活動とゴールとの関連を定義できる
  - アクティビティ：同期，非同期，条件分岐を表記可能
- アクティビティ図の中から，システム支援対象のアクティビティを選択すると，
  - アクティビティの順序関係，開始／終了の条件
  - 主体

を明示できる．

アクティビティ図の例

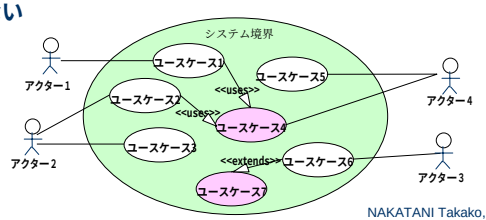


ユースケース図

- システム境界の視覚化
- システムと関与する役割の視覚化
- システム境界上のユースケースは，アクティビティ図内のアクティビティ，あるいはプロセスに対応していなければならない．
  - システム内部のユースケースは開発者向けの機能分割指針を表すものであって，要求定義とは直接関係はない．

ユースケース図の例

- アクター：アクティビティ図のレーンの主体
- ユースケース：システム境界内のアクティビティ
- アクティビティの順序関係，事前／事後条件は視覚化されない



ユースケース

- 名前:顧客登録ユースケース
- 目的:顧客へのサービスを拡充するために，新規顧客の登録を行うこと
- アクター:顧客
- 事前条件: アクターは未登録顧客である
- 事後条件:アクターが登録顧客となっている
- 基本系列:
  1. アクターは顧客登録の意思を発する
  2. システムはアクターに顧客登録に必要な情報の入力を求める
  3. アクターは顧客登録に必要なデータを入力する
  4. システムは入力されたデータを検証し，登録を受け付ける
  5. システムはアクターに登録が完了したことを通知する
- 代替系列:
  - 基本系列4で，アクターが未成年の場合は，未成年顧客登録ユースケースを起動する．

登録手順の検討と例外事象の抽出／対処を決定する．

まとめ

- ビジネスモデリングのパターンを蓄積することによって，ビジネス改善／革新を迅速に進めることが可能となる．
- ビジネスモデリングは，ビジネス変化に柔軟に対応できるシステム開発の鍵である．
- ビジネスモデリングは，システム開発とは独立した技術としても有効であるが，その成果物はシステム開発の要求仕様書作成の拠り所ともなる．

54

NAKATANI Takako, 2002.

参考文献

- 非機能的要求の分類について
  - L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos 著:
  - Non-Functional Requirements in Software Engineering,
  - Kluwer Academic Publishers, 1999.

55

NAKATANI Takako, 2002.