第5回 組込みシステム技術に関するサマーワークショップ
～～　わくわくする組込みシステム開発　～～

「アジャイルプロセスの現状
―組込みへの適用可能性を探る―」

7/24(木)チュートリアル２セッションE1:

橋本隆成

自己紹介

- これまでの業務
  - 三菱スペースソフトウエア(株)
    - 艦船搭載用リアルタイムシステム
      - 弾道計算プログラム
      - 戦闘指揮装置
      - 射撃制御装置
      - コンソールシステム
  - 日本ヒューレット・パッカード(株)
    - 分散システムの構築
  - (株)オージス総研
    - オブジェクト指向技術のコンサルテーションとトレーニング
  - ソニー(株)
    - オブジェクト指向、方法論、開発環境導入の社内コンサルテーション業務
    - プロセス改善業務

Copyright:© 2006 Hashimoto Software Consulting International Inc.

はじめに
～このチュートリアルの位置づけ～

- 『アジャイルプロセス』の紹介
  - なぜ、『アジャイルプロセス』が注目されるのか？
  - 『アジャイルプロセス』の主張とメリット
  - 現在『組込みシステム開発』に必要なことは？
- アジャイル・プロセス、CMMI、PMBOKなど
  - 『組込みシステム開発』に有効か？
  - 『組込みシステム開発』には不向きか？
  - それとも分野に関係ないのか？
  - 注意点は？
- セッションF1:『組込みソフト開発で持続可能な開発プロセスとは？』での議論のための問題提起

Copyright:© 2006 Hashimoto Software Consulting International Inc.

アジェンダ

- 第１幕：アジャイル・プロセスとは？
- 第２幕：アジャイル・プロセスの紹介～有名なものを中心に～
  - スクラム（Scrum）
  - FDD(Feature-Driven-Development)
- 第３幕:『組込み分野』へのアジャイルプロセスの適用を探る

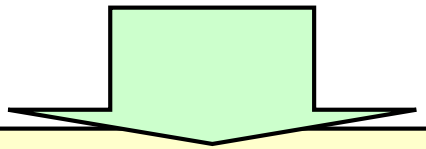
Copyright:© 2006 Hashimoto Software Consulting International Inc.

第１幕
アジャイル・プロセスとは？

Copyright:© 2006 Hashimoto Software Consulting International Inc

アジャイルプロセス出現の動機①

- 現代のソフトウエア開発の課題は？
  - 厳しい競争
    - 開発資金、マンパワーの不足、開発時間の絶対的不足
  - ビジネスの急速な変化
    - ニーズの変化
    - 国際的なビジネス環境
  - 大規模化、複雑化するシステム
    - なかなか決まらない要求
    - 新しい技術への急速な移行



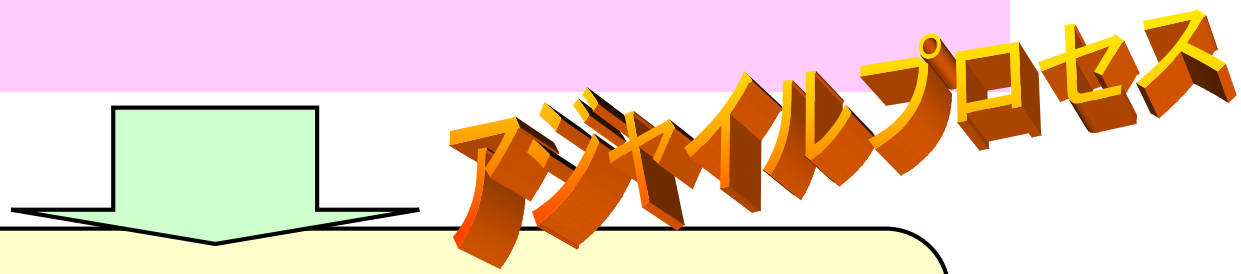
- 課題の多くはエンジニアリング以外にも多く存在
- 外圧に対応することが重要となっている

アジャイルプロセス

Copyright:© 2006 Hashimoto Software Consulting International Inc.

アジャイルプロセス出現の動機②

- 組み込みシステムでの期待されるのは？
  - 航空宇宙、軍事・防衛など一部を省き、中小のプロジェクトが多い
    - CMMIやRUPなどより適用しやすい、お金がかからない
  - 近年は技術的な問題より、プロジェクト管理、要求定義・管理で頭が痛い
  - 競合他社の競争によるシュア獲得
  - etc



カオス理論、CAS理論
断続平衡説、チームの理論
組織論、経営論、教育の問題

Copyright© 2006 Hashimoto Software Consulting International Inc.

アジャイル・プロセスとは？①

- 「状況に合わせて臨機応変に，効率よくソフトウェア開発やプロジェクト管理を行なうアプローチ」
  - 開発期間の短縮化、低コスト化、
  - 多様化＆変化しつづけるユーザニーズ（要求仕様）やビジネスニーズへの対応
  - 本質的な開発作業に専念し、無駄を省き、効率を追求
  - プロセス、規約は存在するが最小限、状況や制約に応じて臨機応変に拡張

Agileソフトウェア開発宣言

我々は，自らAgile開発を実践するとともに，
人々がAgile開発を実践するための支援を通じて，
より優れたソフトウェア開発方法を見つけようとしている．

この活動を通じて，我々は

人と人同士の相互作用を，プロセスやツールよりも
動くソフトウェアを，包括的なドキュメントよりも
顧客との協力を，契約交渉よりも
変化に対応することを，計画に従うことよりも

尊重するに至った．

これは，右側にある項目の価値を認めつつも，
左側にある項目の価値をより一層重視する，ということである．

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Ker
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

(c) 2001, 上記の著者たち
この宣言は，この注意を含めた全文である場合に限り，
どのような形でも自由に複製してよい．

Copyright:© 2006

アジャイル・プロセスとは？②

- 開発プロセスアジャイルであるか否かは実は明確な定義は存在していない
  - エンジニアリング、プロジェクトマネージメント
  - 方法論、プロセスの対象範囲もまちまち
- **アジャイルアライアンス**
  - アジャイル開発のアライアンスがアジャイルについての「共通理念の宣言」やアジャイル開発のための「活動」「メーリングリスト」「記事」「ニュース」などを紹介

アジャイルアライアンスのWebサイト
（http://www.アジャイルalliance.org/home）

Copyright:© 2006

アジャイル・プロセスの仲間たち

- **アジャイル開発プロセスの仲間**
  - XP
  - スクラム
  - クリスタル
  - FDD(Feature-Driven-Development)
  - アダプティブソフトウェア開発(ASD)
  - MDA(Model Driven Architecture)
  - Etc

＊今後も増えることが想定される

『価値観』や『前提条件』など、プロセスの持つ“哲学”に注意

価値体系の軸の一例		
方法論の動機	開発者の役割	プロジェクト管理者の役割
顧客の役割	人間関係の位置づけ	協調関係の位置づけ
競争の位置づけ	コミュニケーション位置づけ	技法(技術)の位置づけ
開発Tool	開発の目的	満足のいく仕事
製品出荷への考え方	開発規模	人と技術の関係

[表:価値体系の軸の一例]

Copyright:© 2006 Hashimoto Software Consulting International Inc.

方法論に見られる特徴の比較

方法論 / 価値体系の軸	開発方法論			
	Rational Unified Process	RAD方法論	XP	スクラム方法論
人の位置づけ	重視するがプロセス中心。 人は作業の分担と責任が明確化されている	重視する。 人は作業の分担と責任が明確化されている	非常に重視する。 人が中心。プロセスや技法は手段に過ぎない。 基本的に分析～実装、テストまで同じ人が担当	非常に重視する。 人やチームが中心。 プロセスや技法は手段に過ぎない
エンジニアリング作業手順の定義	詳細に定義されている	詳細に定義されている	特に決められていない。基本的な作業の流れは存在する	特に決められていない
プロジェクト管理作業手順の定義	ワークフローとして手順、技法が詳細に定義されている	タイムボックス手法として手順、技法が定義されている	具体的なプロジェクト管理はあまり明確になってない	基本的な管理作業ワークフローが存在する。スクラムMeeting、ProductPlanningMeetingなど
開発者間のコミュニケーションの重要性	重視する	重視する	口頭によるコミュニケーションを非常に重視する	口頭によるコミュニケーションを非常に重視する
ドキュメントなど成果物の位置づけ	重視する。モデルドリブン開発を推奨	重視する。モデルドリブン開発を推奨	可能な限り作成しない。成果物は作業のオーバーヘッドを生むソースコードが重要	可能な限り作成しない。成果物は作業のオーバーヘッドを生む。ソースコードが重要
技法の位置づけ	作業ワークフローの中に定義されている	作業ワークフローの中に定義されている	ペアプロ、リファクタリング、テストファースト、計画ゲームなど。	定義していない。状況に応じて選択し、利用する。
再利用性の考慮	重視する	重視する	考慮しない	考慮しない
開発規模	中規模～大規模	特に限定していないが、小規模～中規模が中心	小規模	小規模。但し,中規模以上への適用方があり、実績もある
対応分野	汎用的	基本はIT分野。他の分野にも適用は可能	汎用的。ただし、ミッションクリティカルな分野はあまり意識してない	ITが中心。但し、他の開発分野でも利用は可能。
Toolの位置づけ	非常に重視する。方法論をサポートするToolの使用を推奨	非常に重視する。方法論をサポートするToolの使用を推奨	特に規定しない。ただし、テストToolやリファクタリングToolを利用する	特に規定しない
変化に対する適応性	考慮している	考慮している	非常に意識しており、変更に強い	非常に意識しており、変更に強い
方法論のタイムスコープ	1つのプロジェクトから組織の成熟度までカバー	基本的にプロジェクト単位に注力するが、プロセスをテンプレート化して組織の成熟度までカバー	基本的にプロジェクト単位	基本的にプロジェクト単位
方法論の目的	最適化された手順とメトリックスを用いた生産性と品質向上	ソフトウエアによるビジネス･リエンジニアリングと短期間によるシステム開発	質の高いシステムを効率良く開発する。 システムを期日までに確実に完成させる	費用対効果に見合う意思をし、システムを効率良く開発する システムを期日までに確実に完成させる

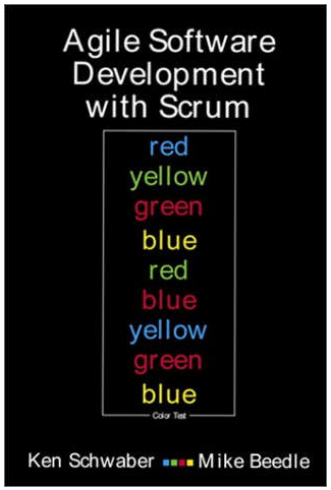
Copyright:© 2006 Hashimoto Software Consulting International Inc.

出典：技術評論社「SoftwarePeopleVol.1」

第２幕
アジャイル･プロセスの紹介
～有名なものを中心に～

Copyright:© 2006 Hashimoto Software Consulting International Inc

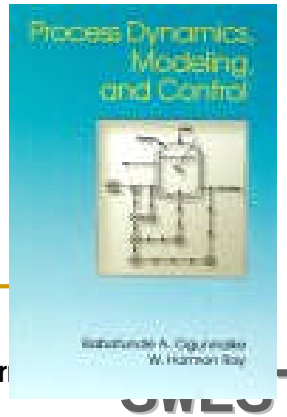
スクラム（Scrum）



Copyright:© 2006 Hashimoto Software Consulting International Inc

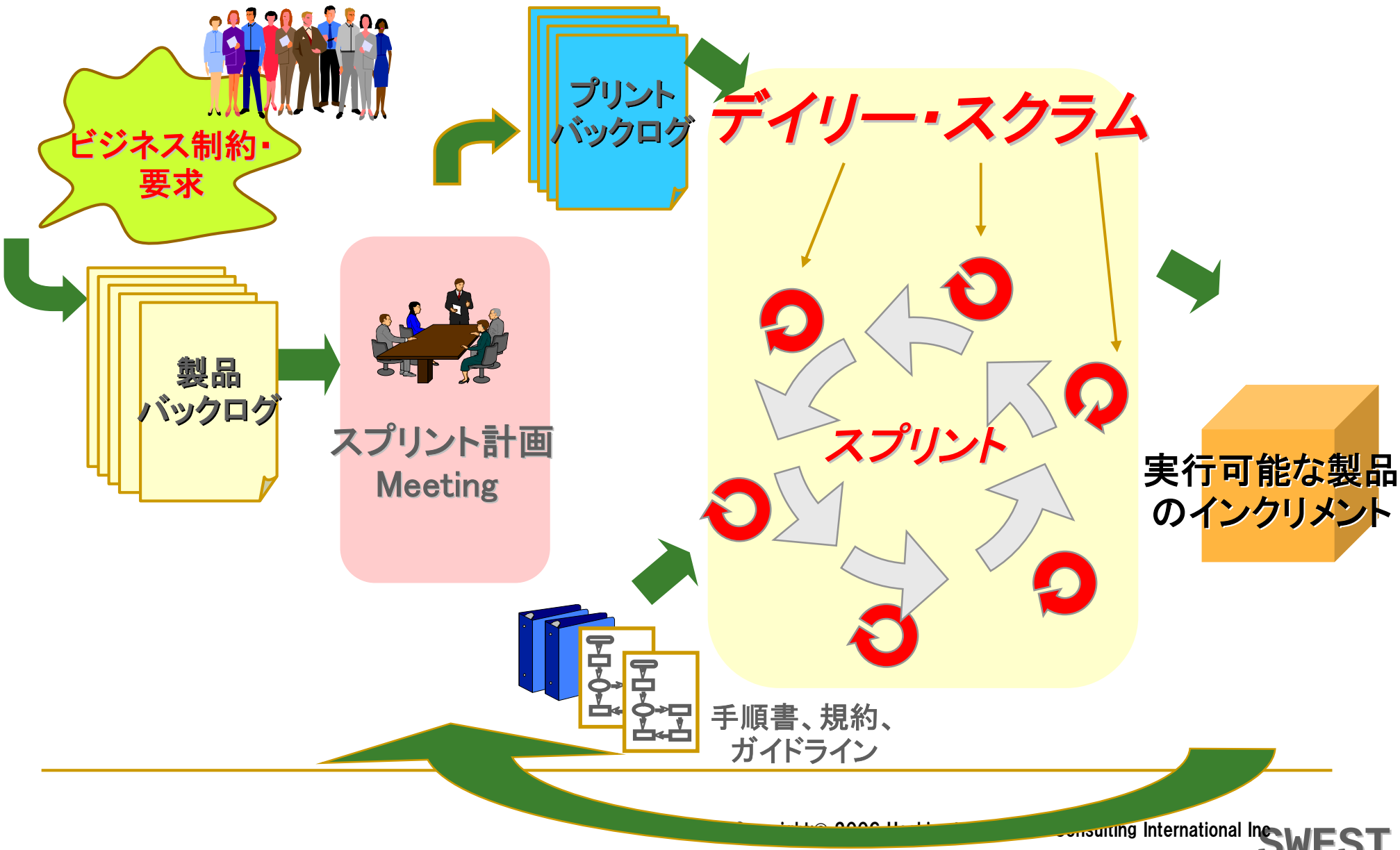
スクラムとは？

- Ken Schwaber、Mike Beedlenの両氏により発表
  - 発表以前からスクラム（の原型）は存在した
- エンジニアリング・プロセスで無く**プロジェクト管理プロセス**
  - 具体的なエンジニアリング・プロセス、手法は定義・指定は無し
- スクラムのプロセス・アーキテクチャ
  - **Sprint**と呼ばれる**30日間（前後）**のサイクルを反復
    - 各反復は**全力疾走**
    - Sprint毎に決められた**作業に集中**
  - 製品責任者，スクラムマスター，スクラムチームなど明確な役割分担
  - 少人数による開発チームを対象
    - 原則 **7±2人位**
    - 中・大規模の場合は**階層化により実施**



Copyright:© 2006 Hashimoto Software Consulting Inter

スクラムのプロセス・アーキテクチャ



スクラムのプロセス解説①～主な人物と役割

- **スクラムチーム**
  - **自己組織化**するチーム
  - 細かな**役割分担はない**
- **スクラムマスター**
  - チームの障害を解決する
    - メンバーの机の配置
    - PCなど開発環境
    - コーヒーサーバーの設置
    - etc
  - Dailyスクラムでの報告先
  - Sprintのキャンセルなどの権限もある
- **プロダクトオーナー**
  - 製品バックログの優先度を決められる
  - プロダクトバックログの唯一の管理者
  - プロダクト管理者（商品開発）、プロジェクト管理者、ユーザー部門の管理者（社内開発）

Copyright:© 2006 Hashimoto Software Consulting International Inc.

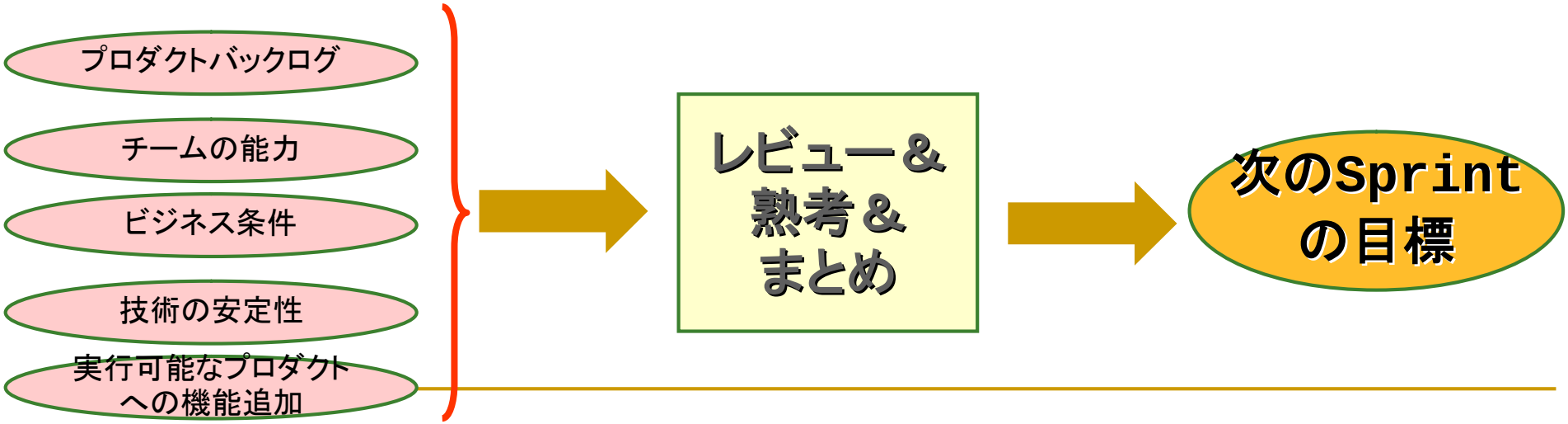
スクラムのプロセス解説②～Meeting①

- **Dailyスクラム**
  - 毎日の進捗確認ミーティング
  - 約15分（技術的な話は後で➔フォローアップMeeting）
  - 「スクラムルーム」と呼ばれる会議室（部屋）
  - 同じ時間（時間厳守：罰金制）、同じ場所で実施
  - 輪になって座る（立つ）
  - ブタ（チームメンバー）とニワトリ（発言はできない）
  - 左回りに一度に一人だけ、3つに事柄につき発言
    - 前回のスクラムミーテキングから何をしたか？
    - 何を行う（予定）か？
    - 障害があるか？
- **SprintレビューMeeting**
  - Sprintの成果（プロダクト・インクリメント）をデモ
    - パワーポイントの作成は禁止
    - 4時間程度の情報提供Meeting
    - 顧客、ユーザー、管理者（層）、製品責任者（プロダクトオーナー）に提示

Copyright:© 2006 Hashimoto Software Consulting International Inc.

スクラムのプロセス解説③～Meeting②

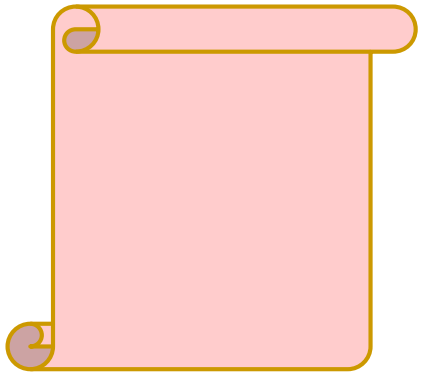
- Sprint計画ミーティング：2つのミーティング
  - 顧客、ユーザー、管理者（層）、製品責任者（プロダクトオーナー）、スクラムチームが次のSprintの目標と機能性を決める
  - チームはSprint内でどうやってプロダクトに機能を構築するかを決定する



Copyright:© 2006 Hashimoto Software Consulting International Inc.

スクラムのプロセス解説④～リスト

- 製品(プロダクト)バックログリスト
  - 顧客と一緒に作成
  - バックログには優先度が付与
  - Sprint計画Meetingで決定
  - 製品責任者(プロダクトオーナー)のみ(1人)が管理

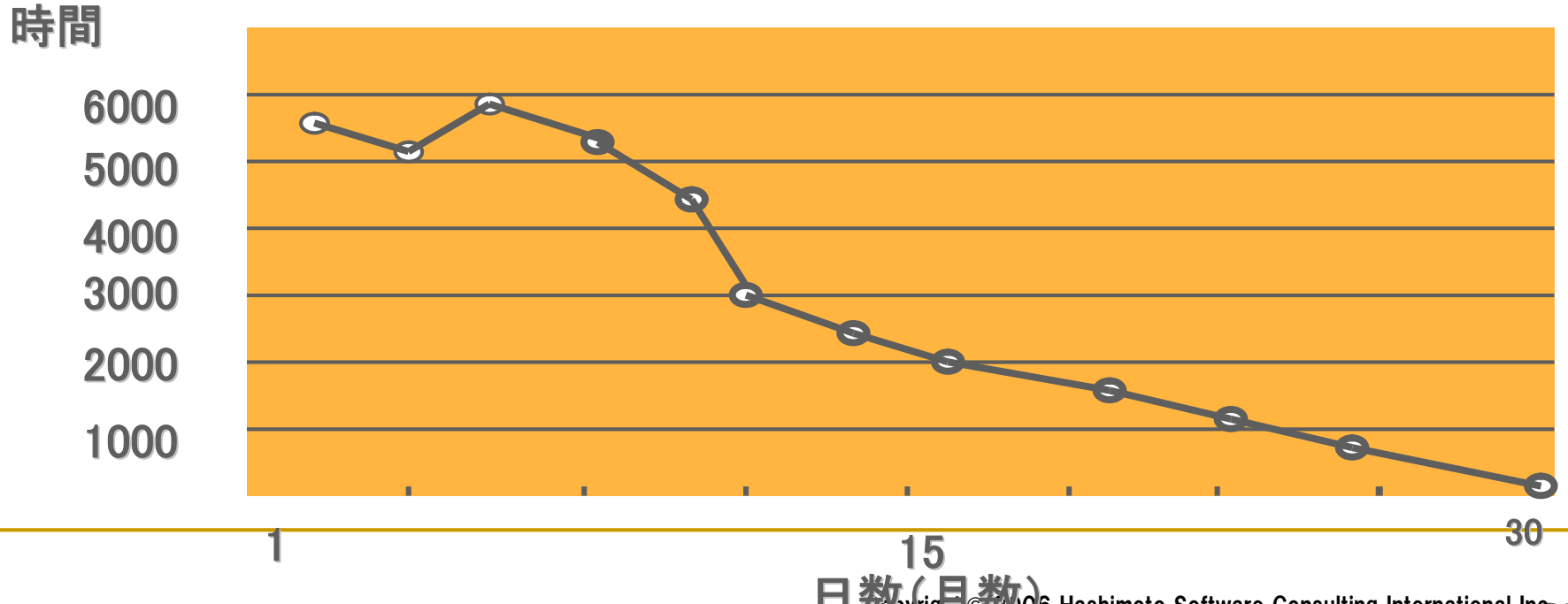


- Sprintバックログ
  - Sprint(30日)の中で実装されるアイテムのリスト
    - 日々必要に応じてSprintバックログ中に追加されていく
    - 環境設定、雑用なども全て考慮する

Copyright:© 2006 Hashimoto Software Consulting International Inc.

スクラムのプロセス解説⑤
～進捗管理・見積もり方法

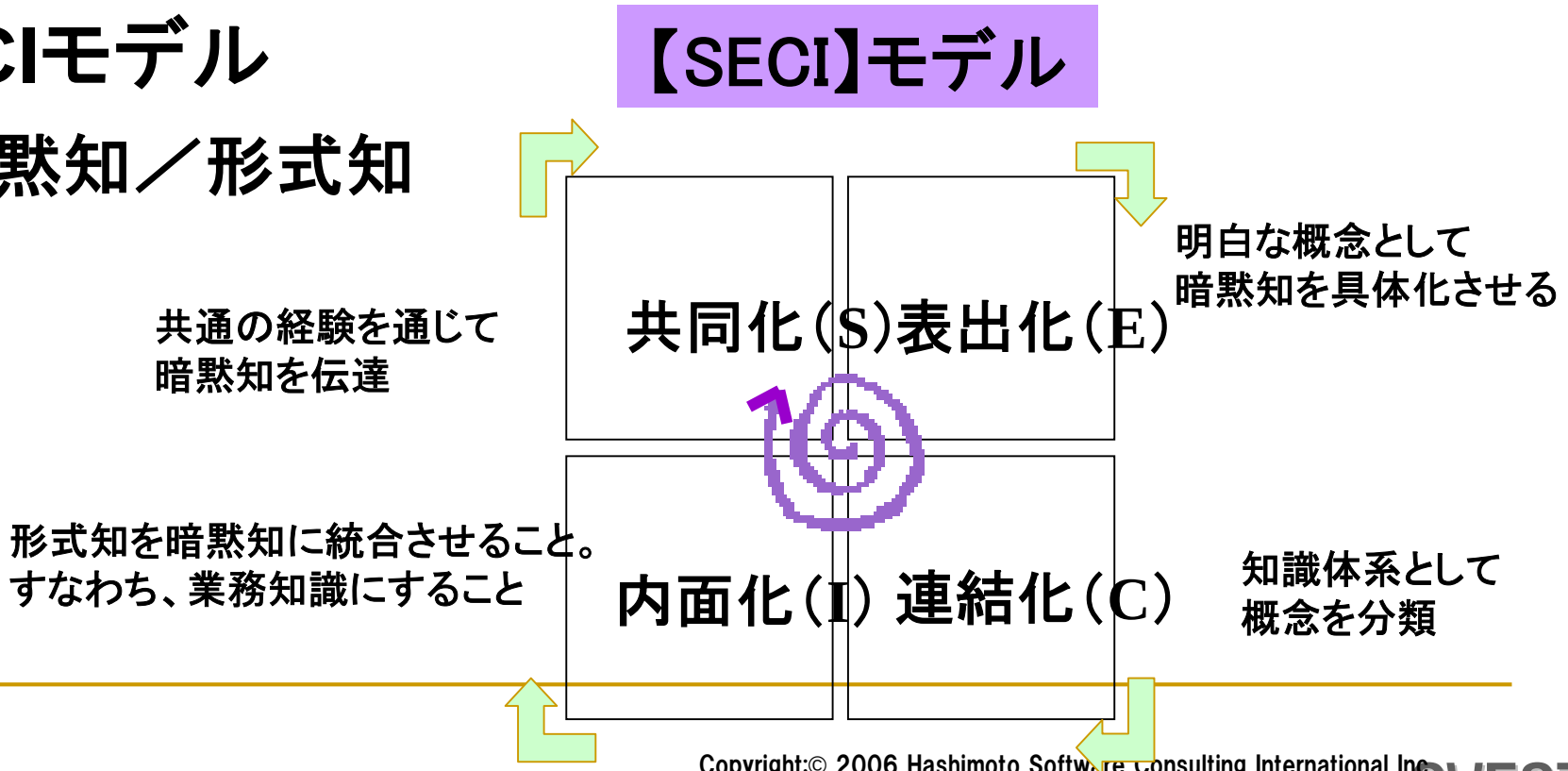
- **残り作業量と期日だけをスクラムは扱う**
  - **期間**はスクラムに存在しない
  - 作業日数はプロダクトバックログの合計をチームがSprint毎に達成できる作業量で割る
  - 結果指向、プロセス指向でない



Copyright © 2006 Hashimoto Software Consulting International Inc.

スクラムの理論背景

- 組織のコミュニケーション
  - 古典的なモデルのTuckmanの“ステージ・モデル”
  - Gersickの“Punctuated Equilibrium（断続さ平衡説）モデル“
- SECIモデル
  - 暗黙知／形式知



Copyright:© 2006 Hashimoto Software Consulting International Inc.

スクラムの主張①

- **「予見型プロセス」や「最適化型プロセス」への疑問**
  - 現在まで主流となっている「ソフトウェア開発プロセス」や「プロジェクト管理プロセス」は、「予見型プロセス」や「最適化型プロセス」
  - 「開発過程はすべて事前に予見可能であり、作業を綿密に計画すべし」
  - 「開発にあたり手順や成果物を事前に明確にし、作業を最適化しよう」
  - 「目標としている結果を得るためには、作業を計画し、管理が重要であり、必要である。そして、プロセスは管理可能であり、結果、プロセスから生成される成果や成果物に対して効果的である」という哲学が存在す
  - 担当者(エンジニア、管理者など)にはあらかじめ決められた作業フロー、成果物を計画に沿って実践し、成果を出す任務が課せられていることを意味

「予見型プロセス」や「最適化型プロセス」の特徴
各フェーズ中の作業1つ1つが明確に示されており、詳しく解説されている
作業間のフローが明確で、Best Practiceな作業の進め方が示されている
作業と成果物が対応づけられており、担当者の定義も明確になっている
開発ライフサイクルのタイミングによる作業の進め方の違いが定義されている
反復型開発、プロジェクト管理の方法が定義されている

【表:「予見型プロセス」や「最適化型プロセス」の特徴の例】

Copyright:© 2006 Hashimoto Software Consulting International Inc.

スクラムの主張②

- 「予見型プロセス」や「最適化型プロセス」の考え方は、「製造業」のプロセスから発展
  - これはテイラー（1911年、1916年）が発表したテイラーの論文が起源
  - クロスビーの製造業の生産性効率・品質向上のMMM(Manufacring Maturity Model)が、ソフトウェア開発に導入、応用したものが今日利用されているプロセスの基盤
  - 専門的な教育を受けていない従業員をいかに管理し、効率を向上されるかという時代のプロセスで「生産プロセス」
- 製造業のプロセスは「生産」のプロセスであり、ソフトウェア開発は「開発」である
  - 従業員は高度な教育を受けており、可能な限りの創造性や知的作業を要求される
  - 開発するシステムの要求から定義し始め、新しい技術によって実装される、新製品である
  - ソフトウェア開発はその名の通り「開発」の作業であり、「生産」ではない

Copyright:© 2006 Hashimoto Software Consulting International Inc.

FDD
(Feature-Driven-Development)

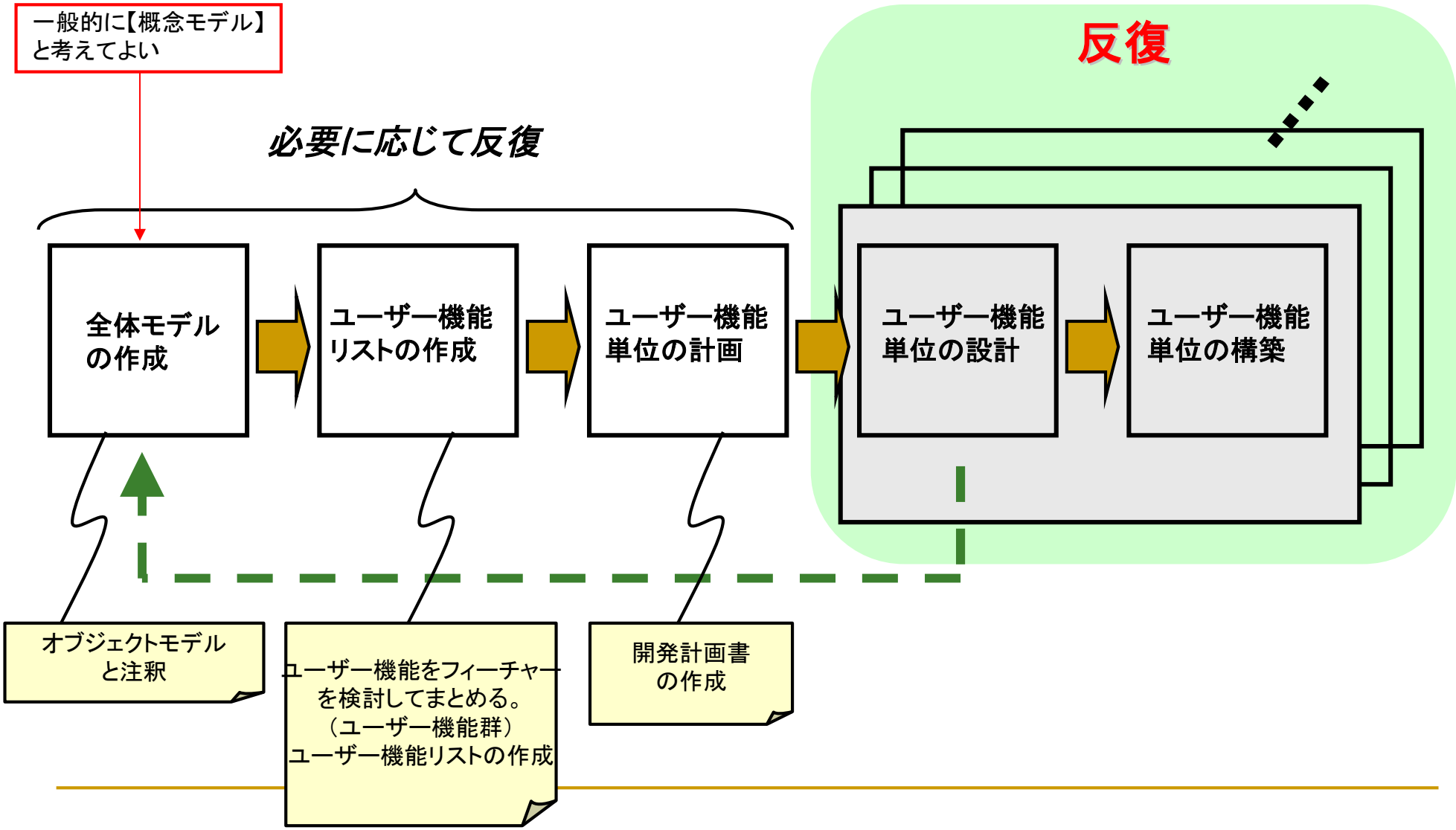
Copyright:© 2006 Hashimoto Software Consulting International Inc

FDDとは？

- ユーザー機能（Feature）駆動
  - ユーザーが必要とする機能で進めていく
    - イテレーション、プロジェクト管理
  - オブジェクト指向モデリングが中心にある
    - オブジェクト指向技術を指定していると言ってよい
    - ビジュアルな進捗管理を重視
  - 人やチームの位置づけを重視している
    - オブジェクトの静的構造とフィーチャーが直交してるため
    - 責務を明確にして作業を進める
      - クラスオーナー（責任をもつクラスのオーナー）
      - チーフ・プログラマ（フィーチャーセットに責任）
  - 作業と担当者を明確にしている
- 中・大規模に対応したアジャイルプロセス

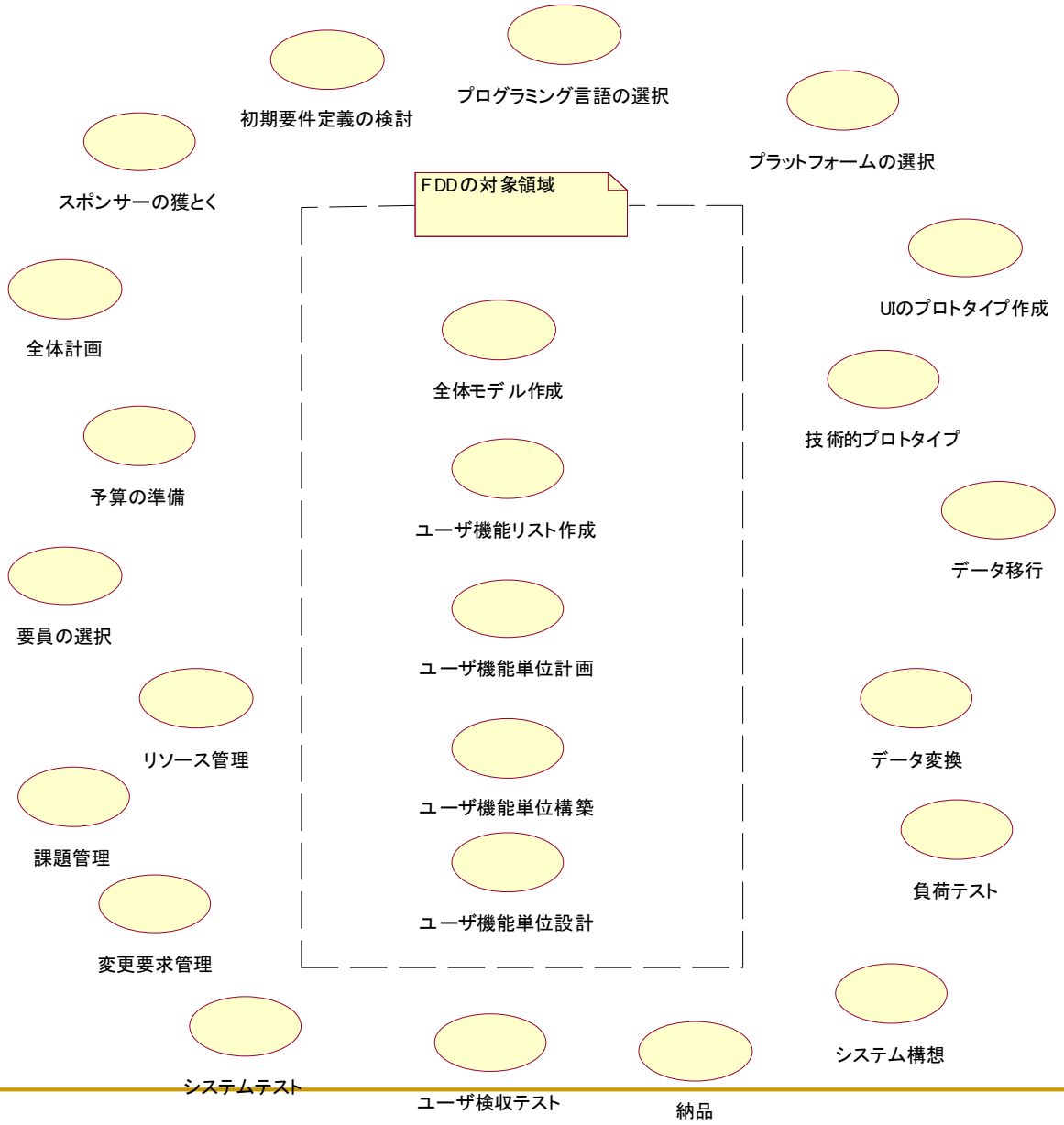
Copyright:© 2006 Hashimoto Software Consulting International Inc.

FDDのプロセス・アーキテクチャ



Copyright:© 2006 Hashimoto Software Consulting International Inc.

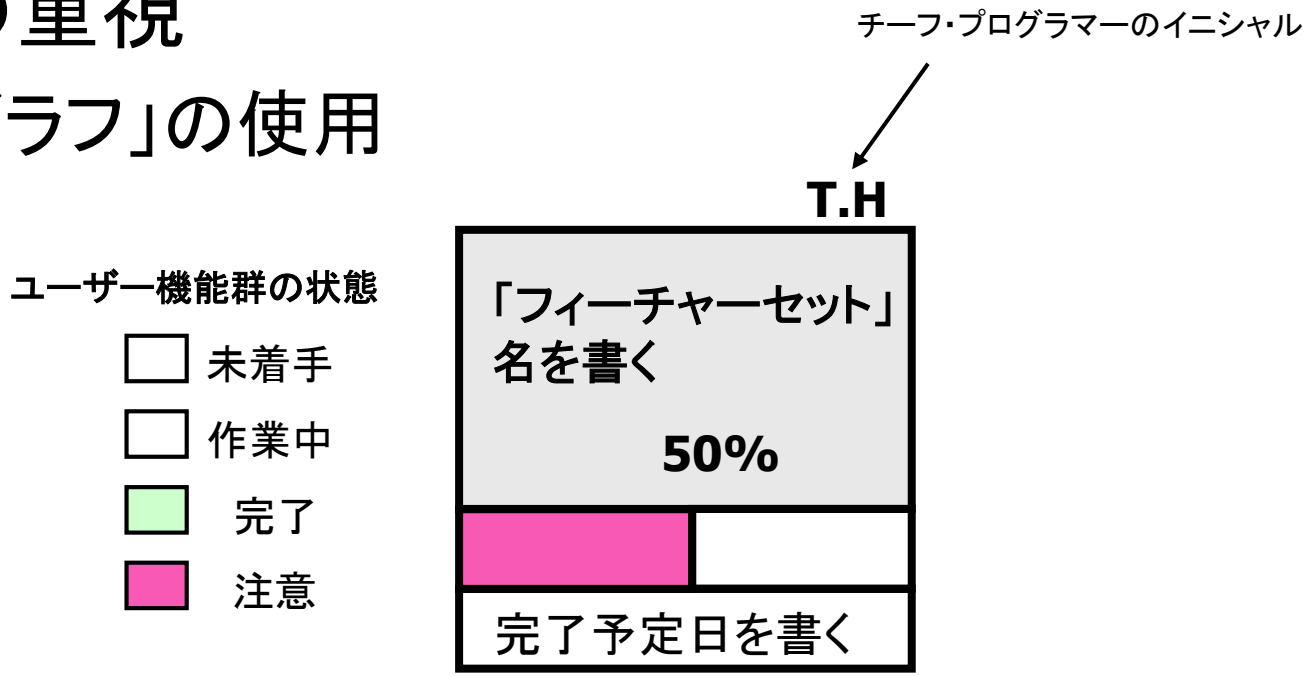
アジャイルプロセスの組込み開発への可能性を探る



Copyright:© 2006 Hashimoto Software Consulting International Inc.

FDDの進捗報告

- 可視性の重視
  - 「進捗グラフ」の使用



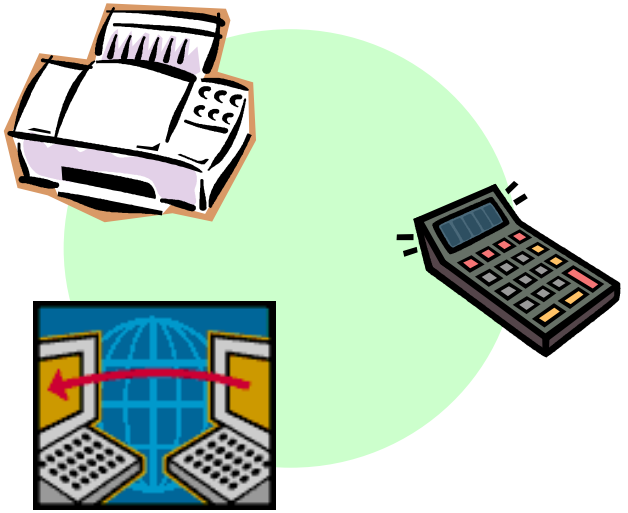
Copyright:© 2006 Hashimoto Software Consulting International Inc.

他のプロセスとの比較

-
  - は支配
    - 少人数でのみ実施可能
    - 少しでも規模が大きく、複雑になれば明確な構成管理が不可欠
- XPと違い明確な作業担当を決める
  - XPは担当者は基本的に上流～テストまで行う
- RUPなどのUse Caseは不十分と指摘
  - UseCaseは粒度についてあいまいである
  - 反復開発に利用するには明確な基準が必要
    - ➔FDDでは2週間の粒度

Copyright:© 2006 Hashimoto Software Consulting International Inc.

第３幕
組込み分野へのアジャイルプロセスの適用を探る



Copyright:© 2006 Hashimoto Software Consulting International Inc

組込み／リアルタイムのプロセス ～MDAを中心に～

- 組込み／リアルタイムのプロセス
  - Executable-UML（シュレイアー&メラー法）
  - Ropes（Rhapsody）
  - RoseRealTime（Object Time）
    ＊Rational Unified Process

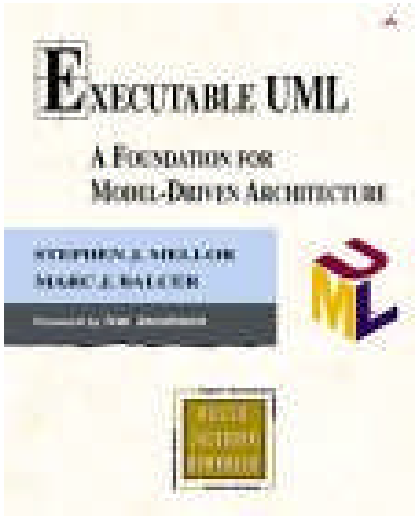
Copyright:© 2006 Hashimoto Software Consulting International Inc.

MDAがアジャイルプロセスなのか？

- 低開発コストの実現
  - MDAによる自動化
  - 効果的な開発プロセス/管理プロセス
  - 分析/設計/ソースコードの再利用
- 高品質の保証
  - 分析/設計/ソースコードの再利用
  - 効果的な開発プロセス/管理プロセス
  - 効果的なテスト
  - MDAによる自動化
- **製品開発エンジニアの工数の多くがテスト&デバック**
  - ソースコード作成、テスト、デバックが仕事ではない
  - 魅力ある製品開発業務に集中してもらいたい

Copyright:© 2006 Hashimoto Software Consulting International Inc.

XUML(シュレイアー&メラー法)



Copyright:© 2006 Hashimoto Software Consulting International Inc

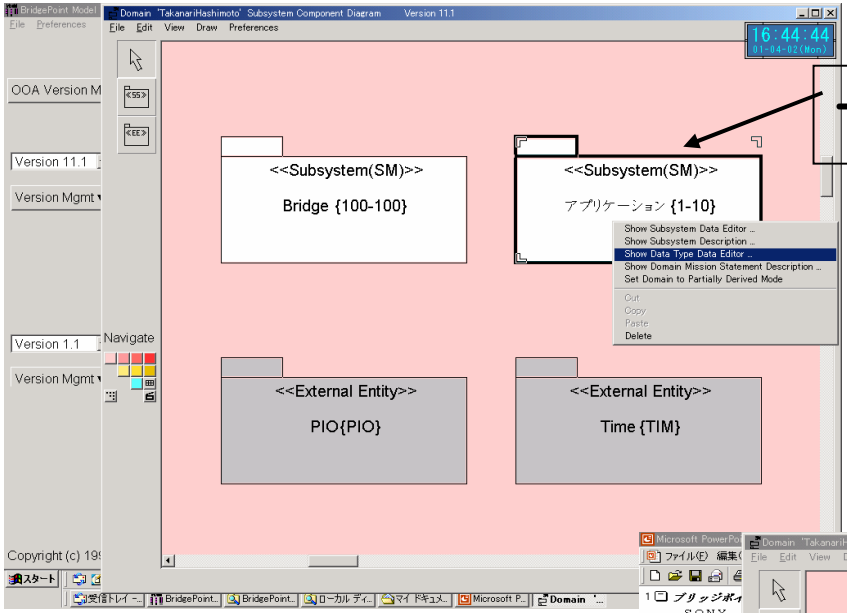
XUML(シュレイアー&メラー法)

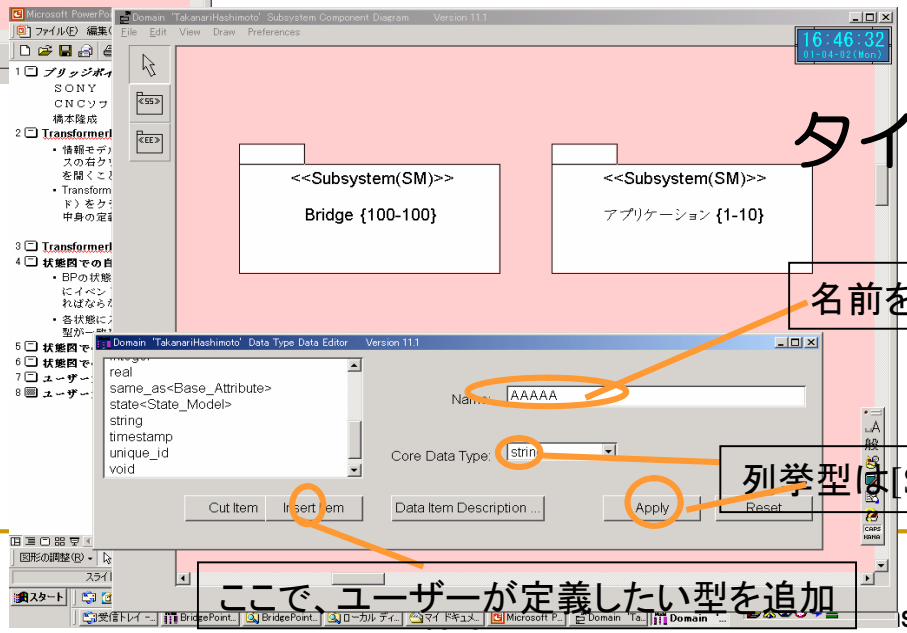
- 特徴
  - MDAのパイオニア
  - 方法論自体がMDAを前提としている
  - 分析と設計の分離
    - 分析結果の再利用の促進
    - 設計(アーキテクチャ)はシステム毎に選択可能
    - アクション・セマンティック言語の利用
  - 分析者と設計者(アーキテクト)の完全な分離
    - モデル作成(アクション言語による記述含む)、パース、モデルシミュレーション(ベリファイアー)、カラーリング、etc➔分析者
    - (SM法で言う)アーキテクチャ作成➔設計者(アーキテクト)
  - トランスレーション（変換型）のソースコード生成

Copyright:© 2006 Hashimoto Software Consulting International Inc.

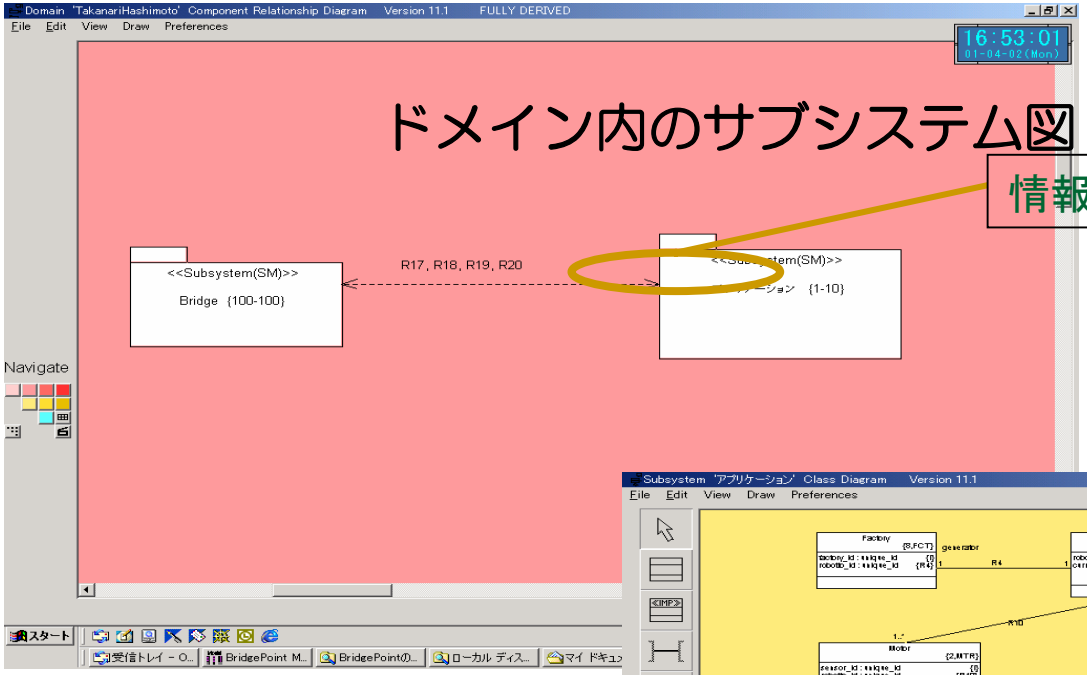
MDAとオブジェクト指向方法論

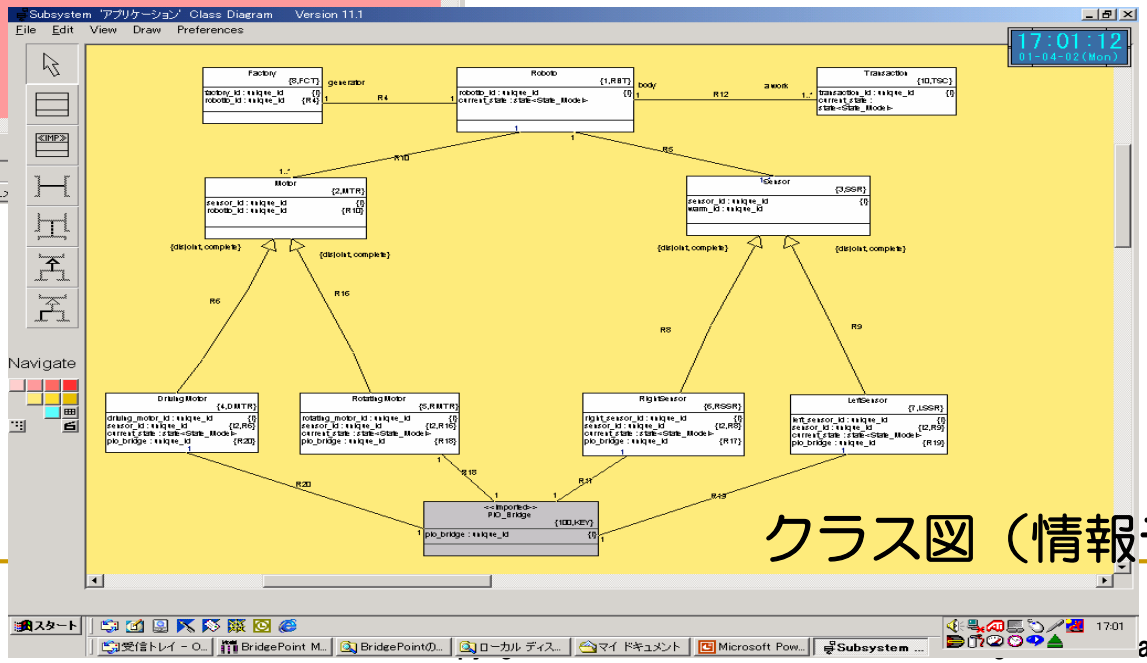
XUML(シュレイアー&メラー法)





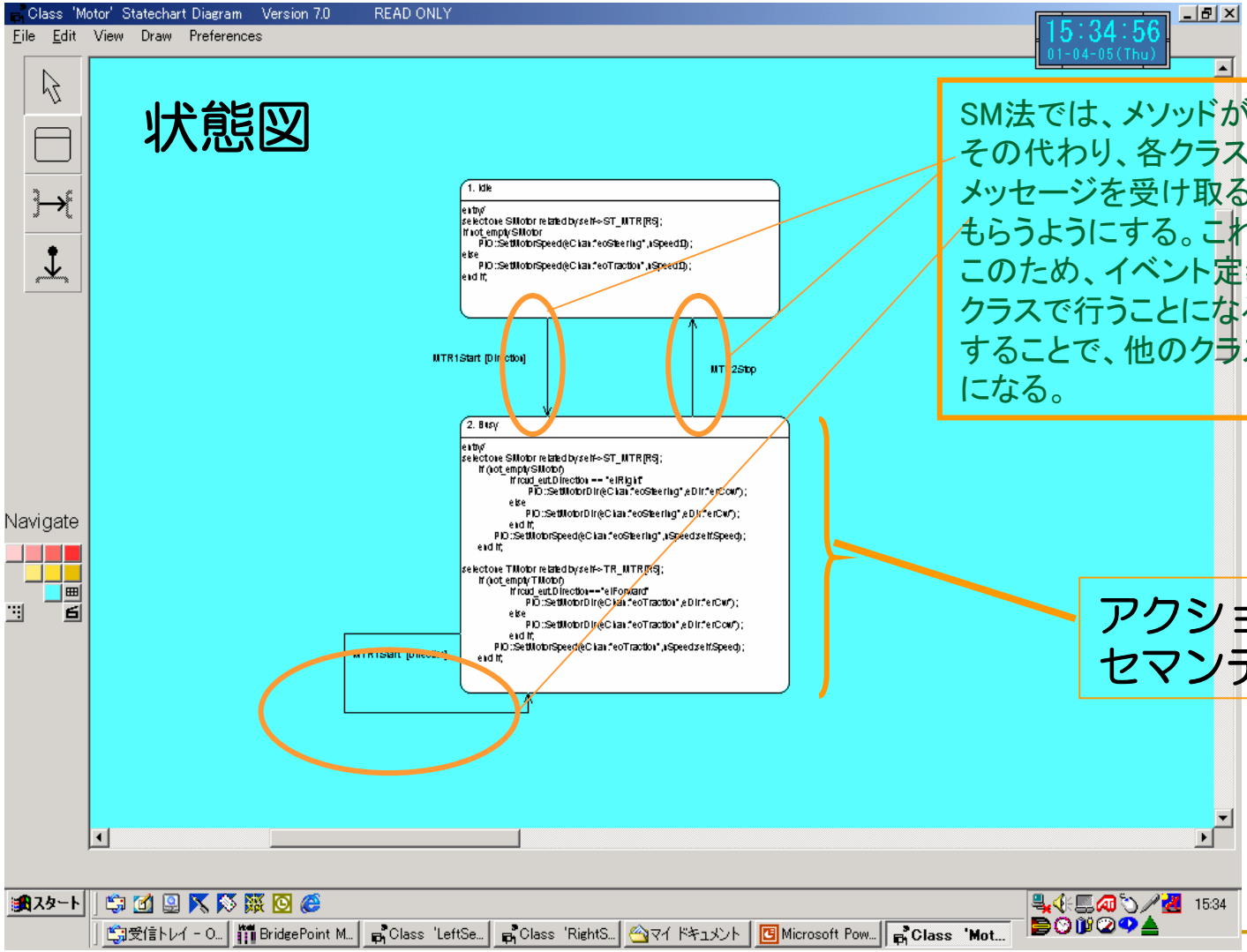
MDAとオブジェクト指向方法論
XUML(シュレイアー&メラー法)





MDAとオブジェクト指向方法論

XUML(シュレイアー&メラー法)



SM法では、メソッドが定義をクラス図に描かない。
その代わり、各クラスには、他のクラスから
メッセージを受け取る場合には、イベントを送って
もらうようにする。これは、状態図で記述する。
このため、イベント定義は、イベントを受け取る
クラスで行うことになる。つまり、このイベント定義
することで、他のクラスに対するインターフェース
になる。

アクション
セマンティック言語

Copyright:© 2006 Hashimoto Software Consulting International Inc.

Rose Real Time
（ROOM：Real Time Object Modeling）

Copyright:© 2006 Hashimoto Software Consulting International Inc

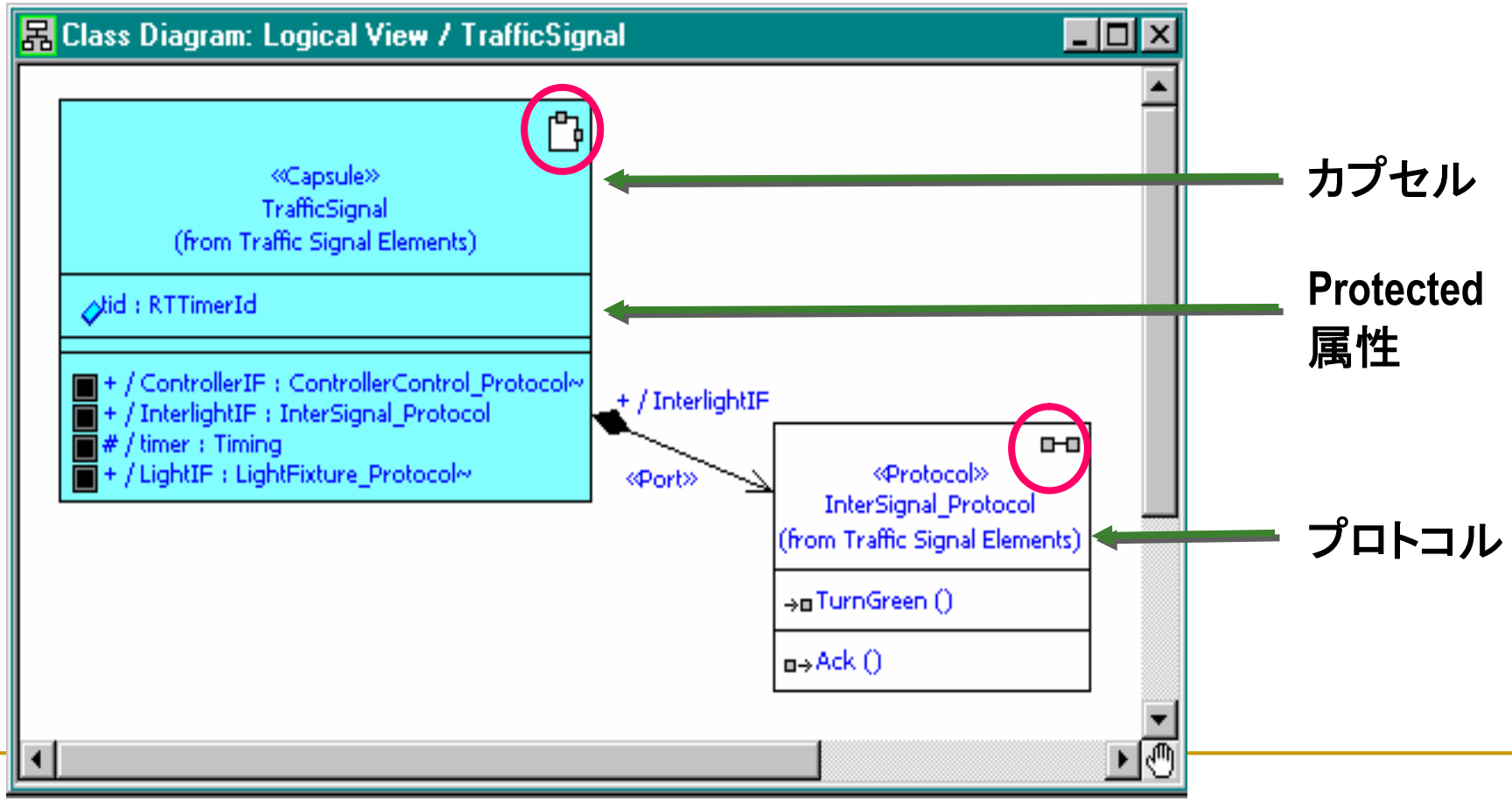
MDAとオブジェクト指向方法論 Rose Real Time(ROOM)

- 特徴
  - 並列／並行性を積極的に対応
    - マルチCPU開発環境を考慮
    - 方法論がMDAを前提
  - コンポーネントレベルの再利用
    - 設計レベルでモデリング
    - 静的構造と動的な振る舞いの両方をモデリング
  - UMLを拡張したアイコンを使用
    - カプセル、ポート、プロトコル、etc.の追加
      - 分析より設計が充実
      - 並行動作するタスクとタスク間通信をモデリング
    - モデル上でシミュレーション
  - 状態図にターゲット言語でアクションを記述

Copyright:© 2006 Hashimoto Software Consulting International Inc.

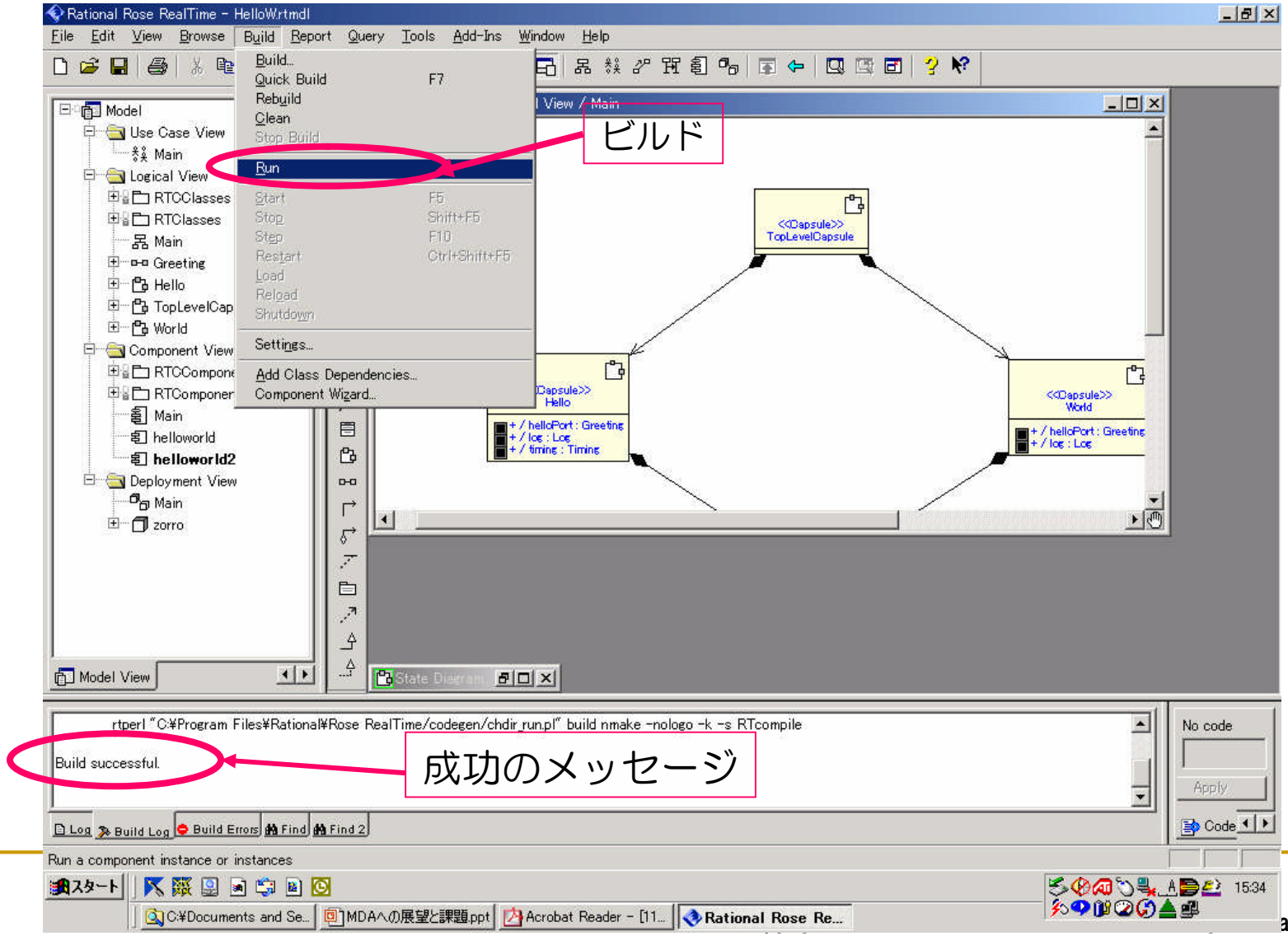
MDAとオブジェクト指向方法論 Rose Real Time(ROOM)

- リアルタイム/組込み開発用にUMLを拡張
- 並行性および分散性にカプセルとプロトコルを使用

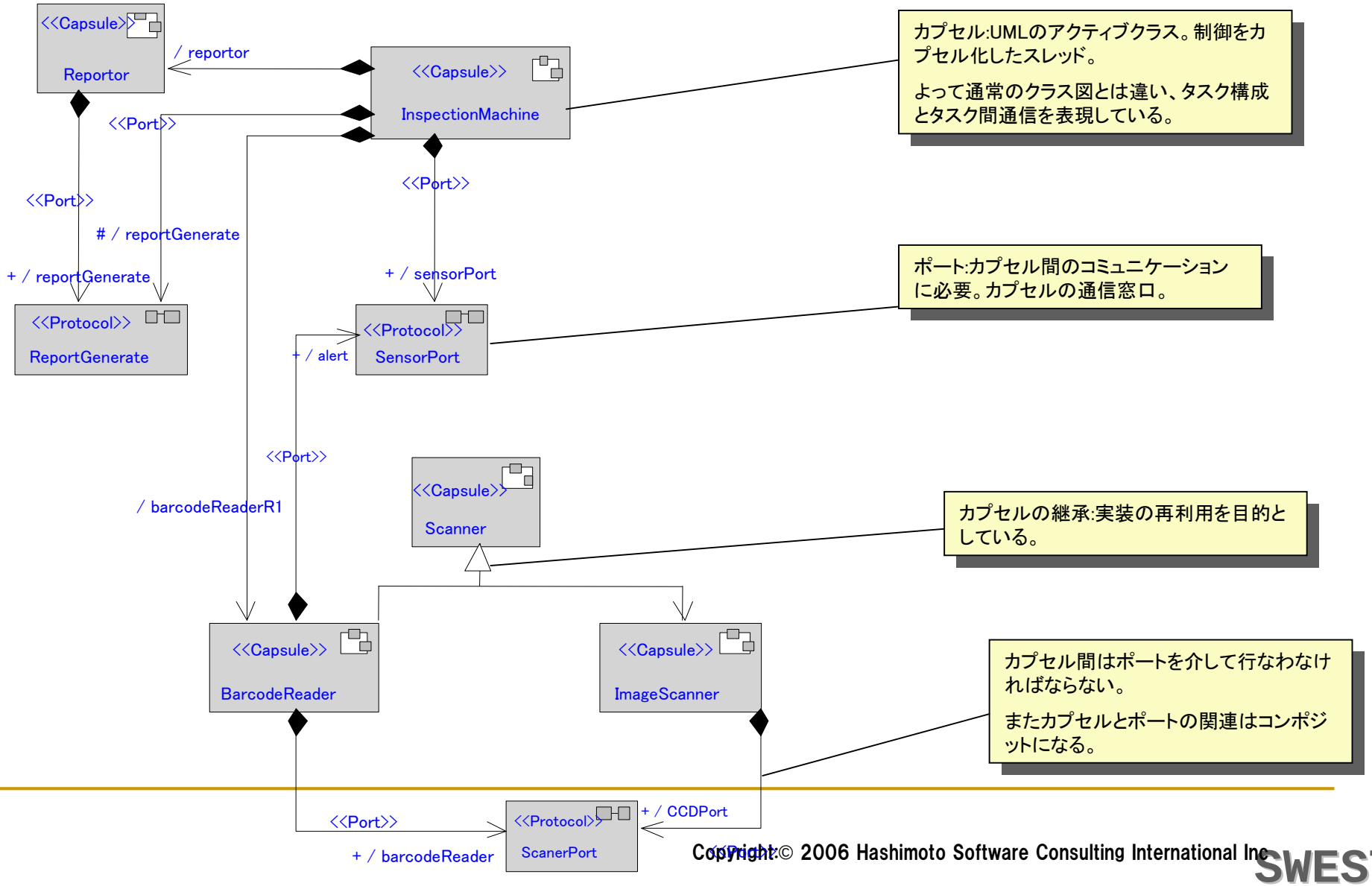


Copyright:© 2006 Hashimoto Software Consulting International Inc.

MDAとオブジェクト指向方法論
Rose Real Time(ROOM)

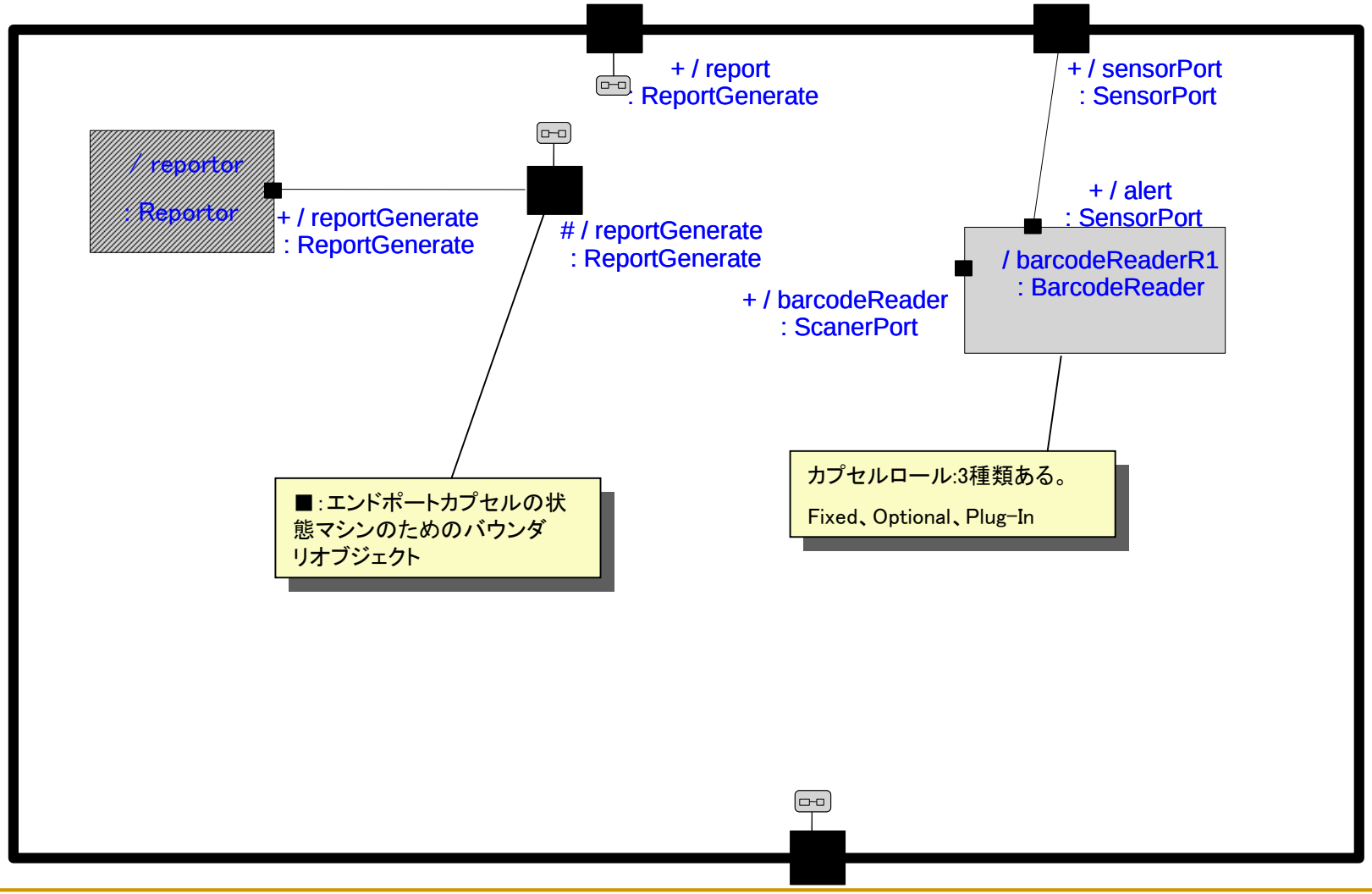


MDAとオブジェクト指向方法論
Rose Real Time(ROOM)



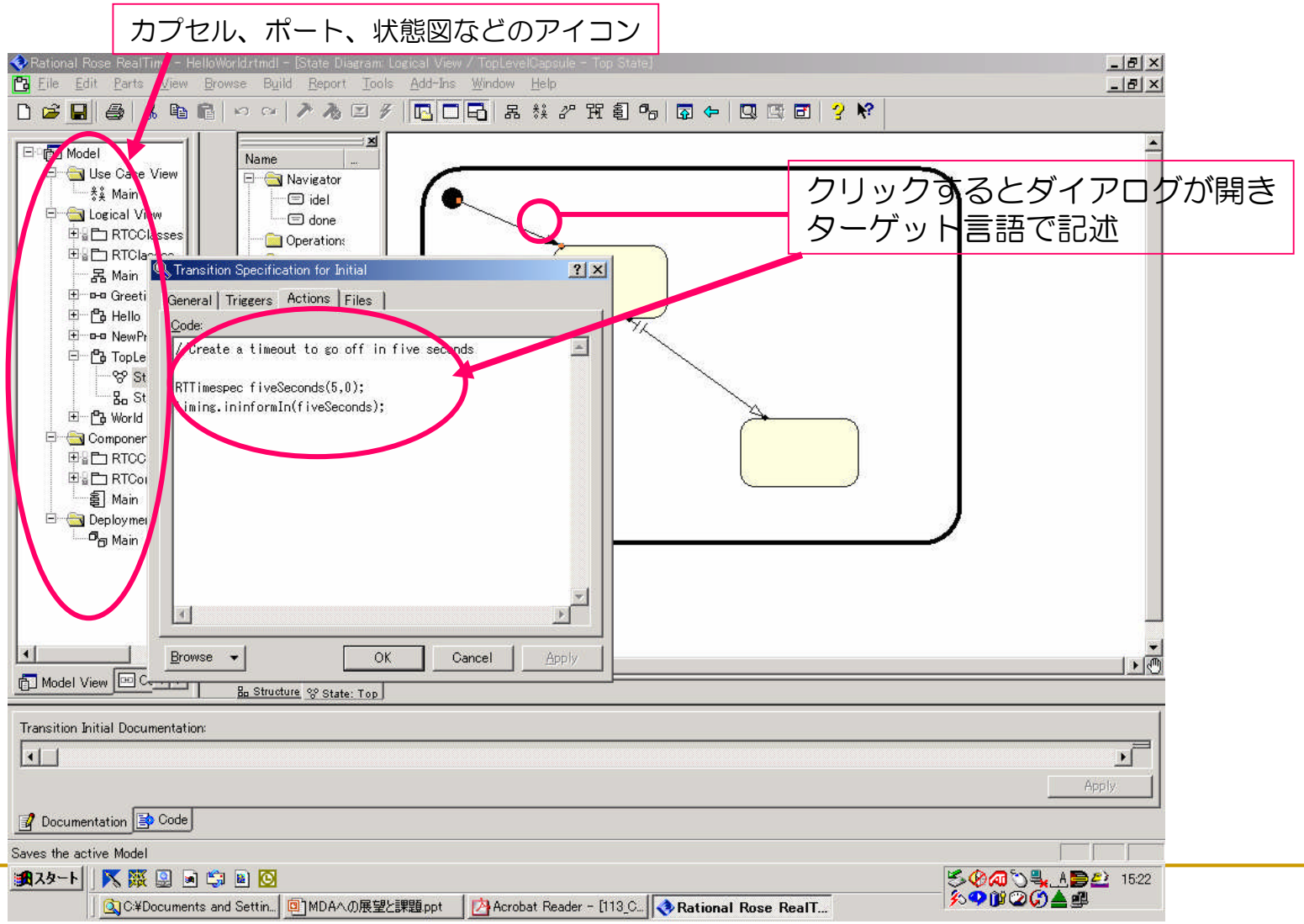
Copyright © 2006 Hashimoto Software Consulting International Inc.

MDAとオブジェクト指向方法論
Rose Real Time(ROOM)



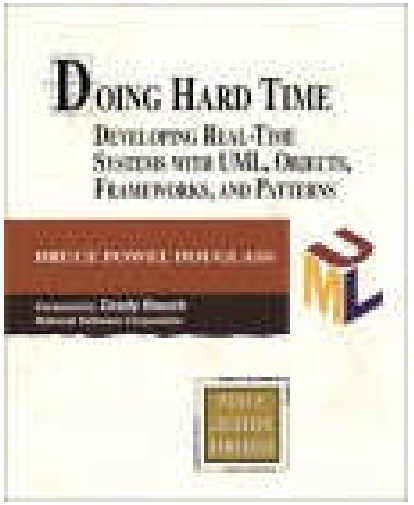
Copyright:© 2006 Hashimoto Software Consulting International Inc.

MDAとオブジェクト指向方法論
RoseRealTime(ROOM)



Copyright:© 2006 Hashimoto Software Consulting International Inc

Ropes（Rhapsody）



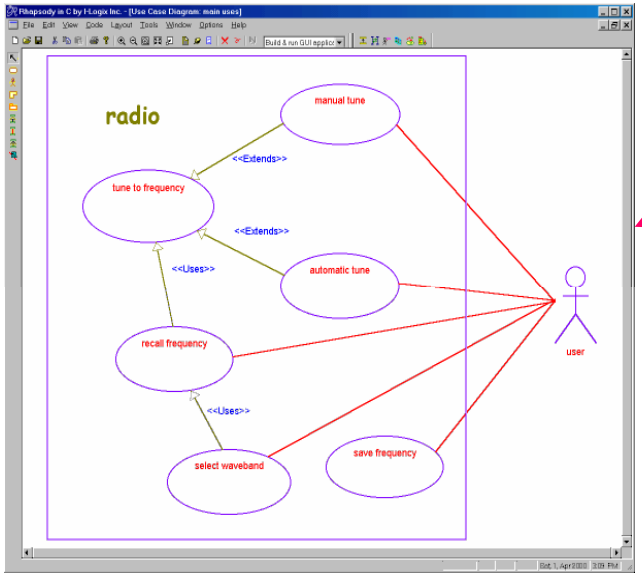
Copyright:© 2006 Hashimoto Software Consulting International Inc

MDAとオブジェクト指向方法論 ROPES

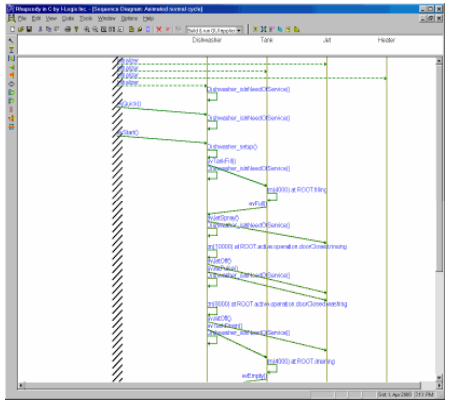
- 特徴
  - オーソドックスOO開発論+リアルタイムシステム手法
    - UMLを拡張したアイコンの利用
    - マルチCPU開発環境を考慮
    - 方法論をサポートするMDA環境を提供
      - ➔方法論自体はCASE-Toolの利用をMUSTとしてない
    - 状態図にターゲット言語でアクションを記述
  - リアルタイムシステム向けデザインパターンの利用

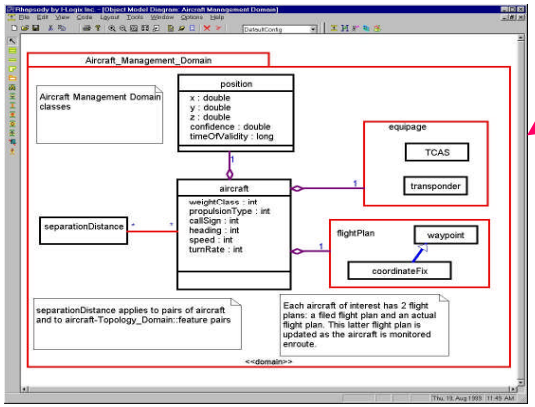
Copyright:© 2006 Hashimoto Software Consulting International Inc.

MDAとオブジェクト指向方法論 ROPES

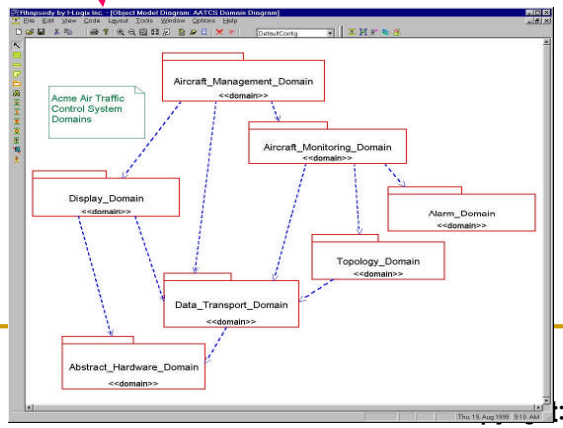


UseCase駆動型開発



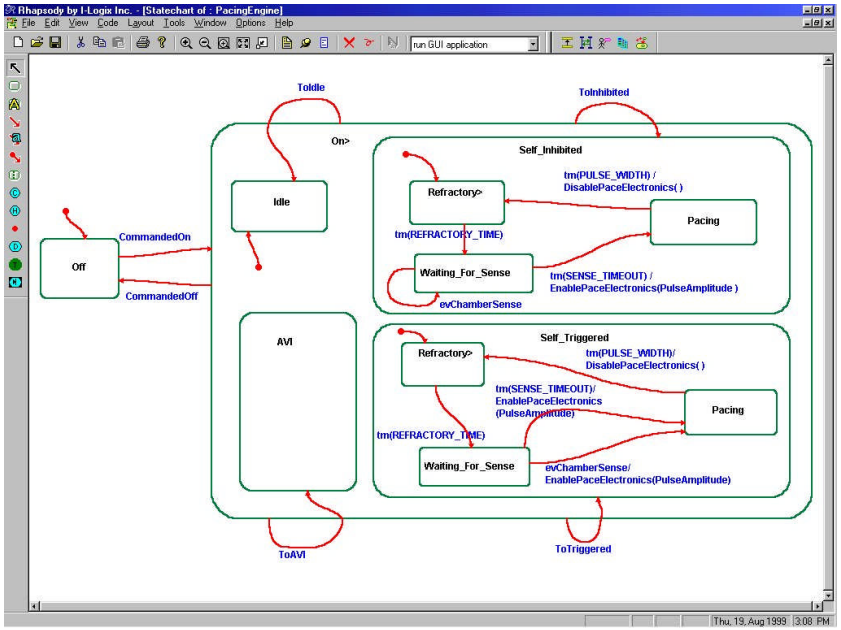


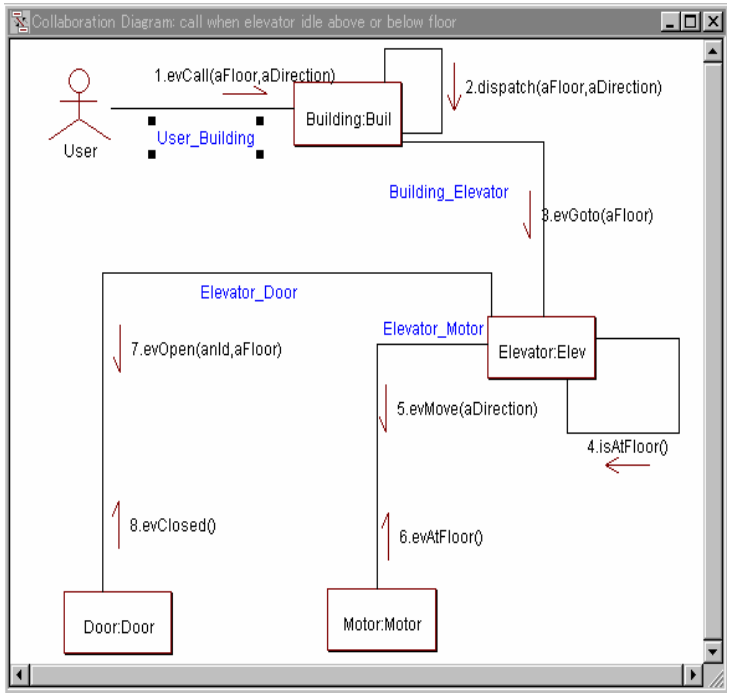
静的構造はクラス図、パッケージ図

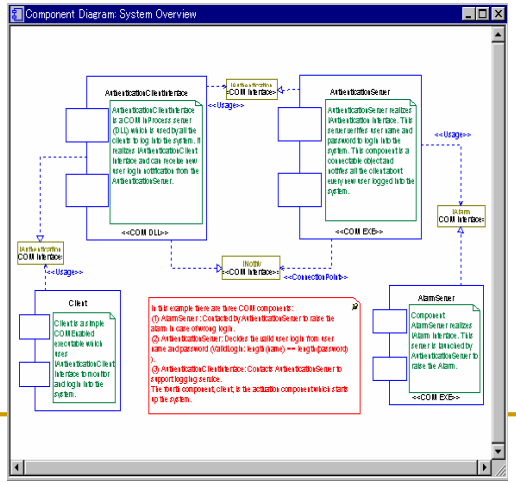


:© 2006 Hashimoto Software Consulting International Inc.

MDAとオブジェクト指向方法論 ROPES

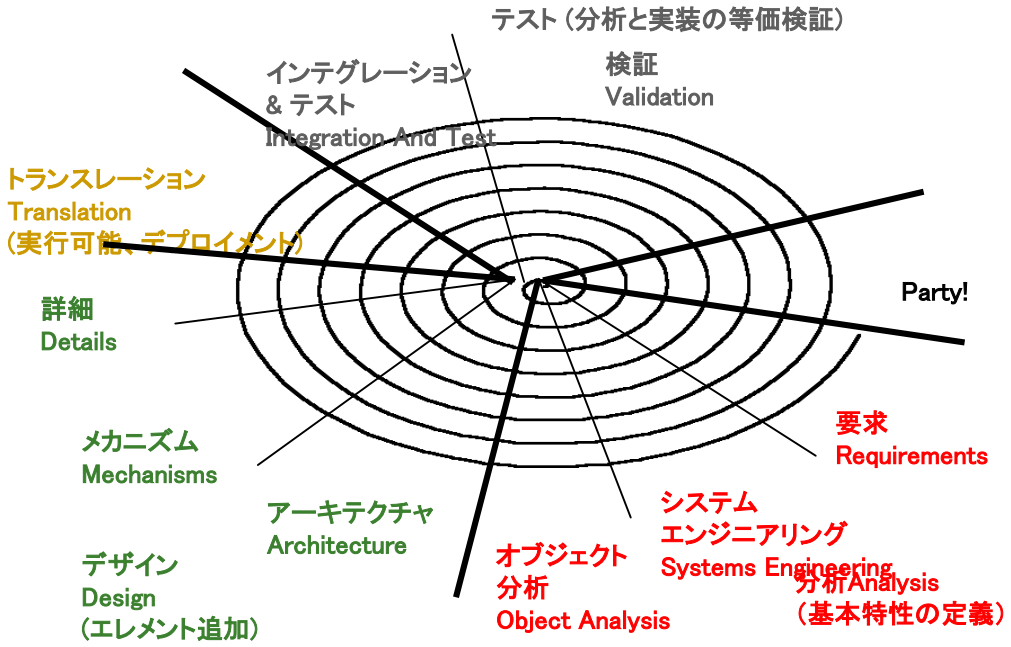


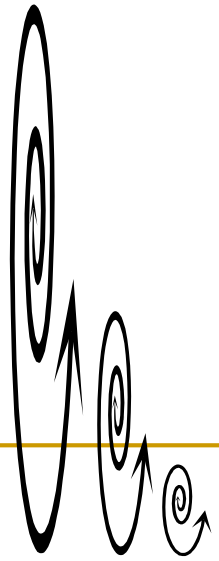


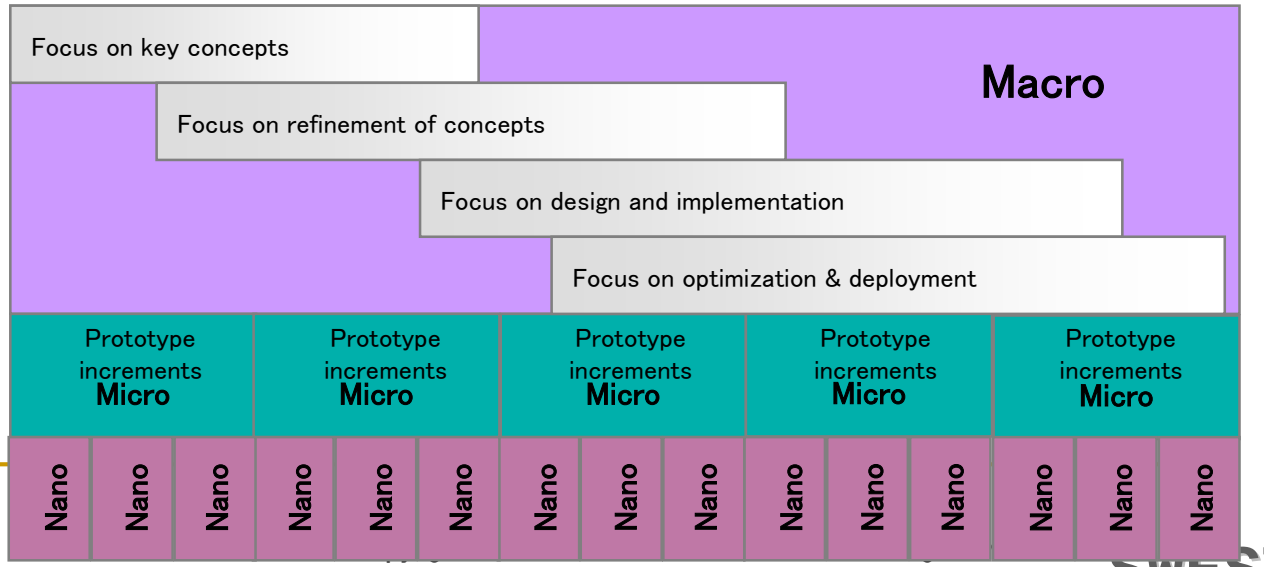


Copyright:© 2006 Hashimoto Software Consulting International Inc

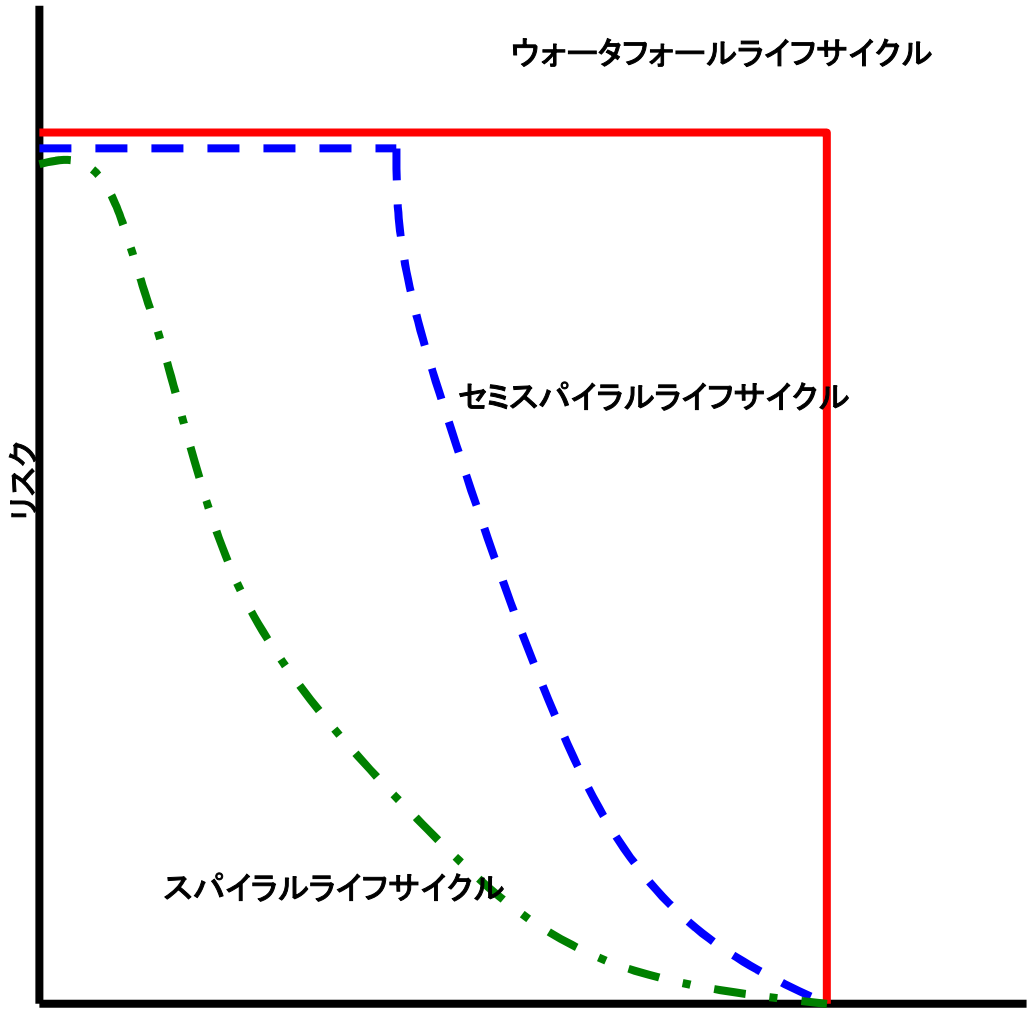
MDAとオブジェクト指向方法論　ROPES

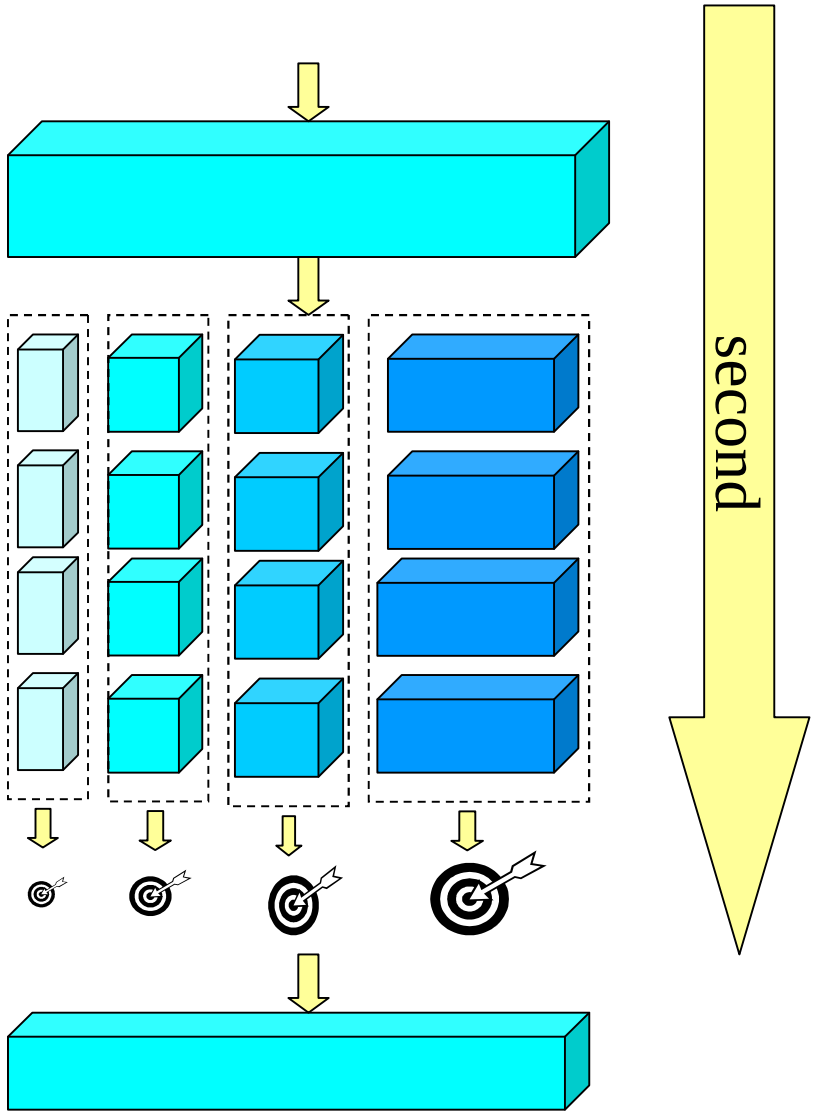






MDAとオブジェクト指向方法論　ROPES





Copyright:© 2006 Hashimoto Software Consulting International Inc.

まとめ

Copyright:© 2006 Hashimoto Software Consulting International Inc

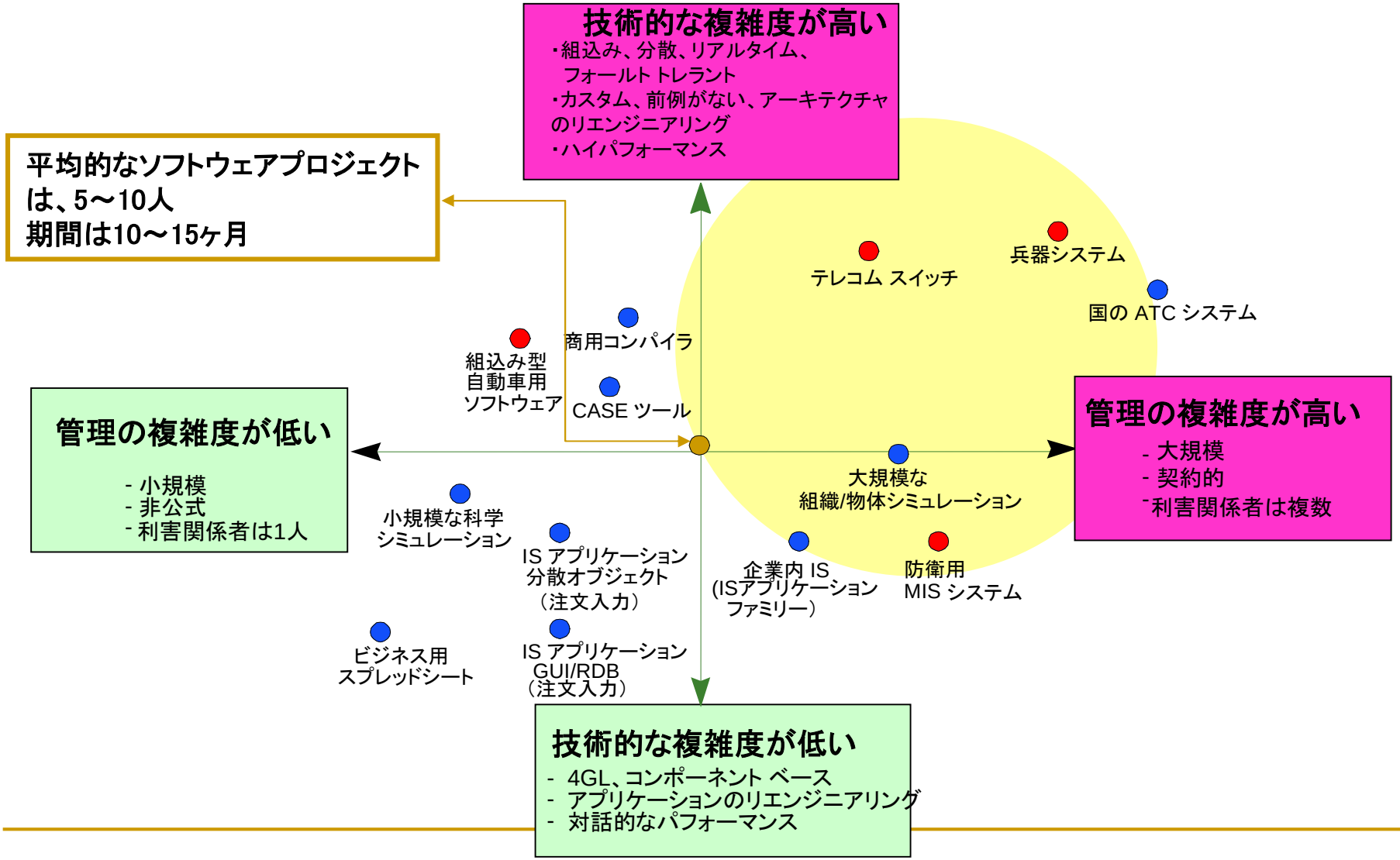
ユビキタス・バリューの時代へ

- 「組込みシステム」を特徴づける主な性質
  - ROM化
  - 多種多様なOS上での稼動
  - リアルタイム性
  - 限られたリソースの制約
  - etc

•一方で、最近のユビキタス・バリュー時代の「組込みシステム」の傾向
  •実にさまざまなシステムが存在している
  •**複雑化**、**多様化**、**大規模化し**、特徴を一口に語れない
  •システムの特徴に合わせた**開発戦略**が必要な時代

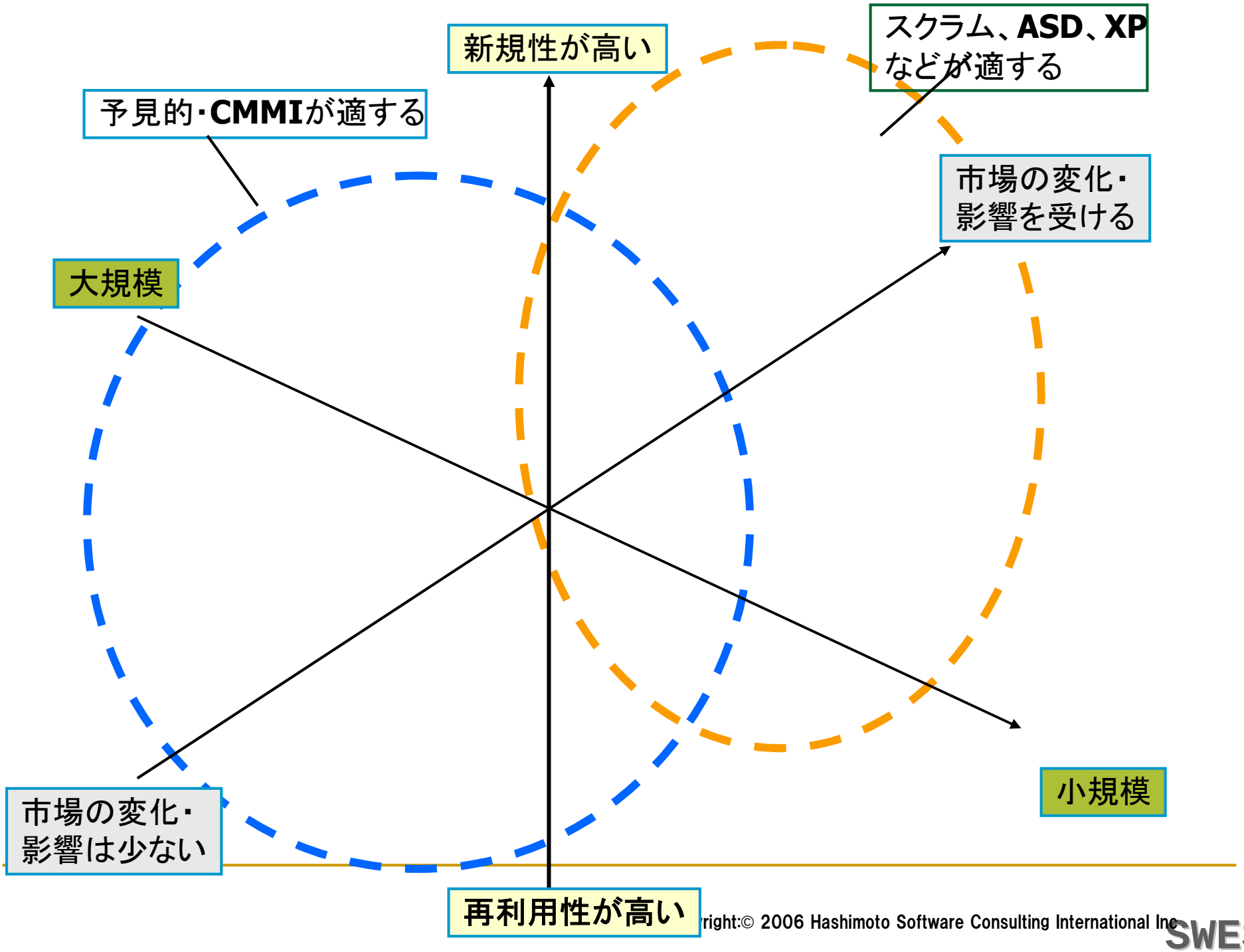
Copyright:© 2006 Hashimoto Software Consulting International Inc.

多様化する組込みシステム



Copyright:© 2006 Hashimoto Software Consulting International Inc.

アジャイルプロセスの組込み開発への可能性を探る



Copyright:© 2006 Hashimoto Software Consulting International Inc.

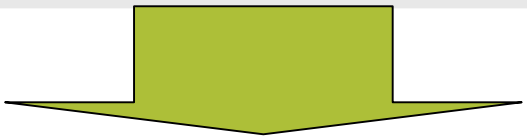
アジャイルでの注意点①

- 具体的なテクニックやスキル向上の方法は示されてない方法論、プロセスもある
  - 優れた要求仕様書の書き方は？
  - 外注管理、COTSに対する品質
  - 見積もり方法はこれでOKか？
    - もう少し定量的に行いたいときはどうするのか？
  - クライアントの効果的なコミュニケーションを行うには？
    - 優れた人材の不足が叫ばれている
- 高い前提スキルが要求
  - スクラムマスター
  - XPのリーダー
  - チームメンバーも高いスキルが必要!?
    - オブジェクト指向、開発言語の能力、etc

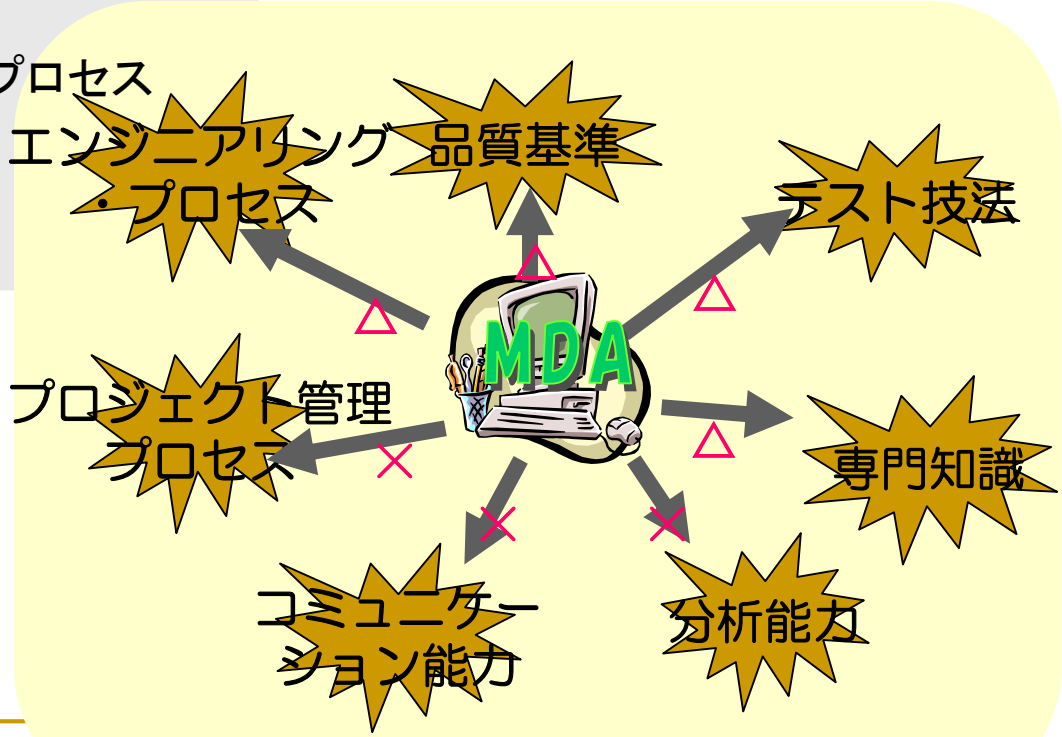
Copyright:© 2006 Hashimoto Software Consulting International Inc.

アジャイルでの注意点②
MDAで何が解決できるか？

- **MDA**利用はオブジェクト指向技術の理解が不可欠
  - **UML**
  - オブジェクト指向方法論
  - 各種オブジェクト指向ノウハウ
- プロジェクト管理プロセスが不可欠
  - **CMM**、**PIMBOK**、**RUP**、アジャイルプロセス
- 開発分野の専門性
  - ドメインエキスパートの経験



専門家／開発者にとって
代わるものではない！



Copyright© 2006 Hashimoto Software Consulting International Inc.

アジャイルでの注意点③

- **分科会２セッションF1:「組込みソフト開発で持続可能な開発プロセスとは？」に向けて**
  - 適用しようとしているプロジェクトに「**本当に必要なもの**」は？
    - 要件開発／管理能力の向上
    - 技術的な問題よりプロジェクト管理
    - 基礎的な設計、プログラミング技術
  - Agileプロセスの多くは細かなタスクをしていない分、分野に関わらず重要な示唆を与えていないか？
  - プロジェクトに本当に必要なものを見極める必要がある。
    - CMMI、PMBOKにも当然大きな魅力がある
      - CMMIは誤解も大きい➔実践されていないのに「**食わず嫌い**」になっている
    - アジャイルを採用するのはなぜか？
  - プロジェクトの特色、企業文化、関係企業との商文化を考慮する必要がある

Copyright:© 2006 Hashimoto Software Consulting International Inc.

参考文献

- 「XPエクストリーム・プログラミング検証編」ジャンカルロ・スッチ他著　小野剛その他訳　ピアソン・エデュケーション・ジャパン
- 橋本隆成『Software People』,Vol.1, 技術評論社, 2002.
- 橋本隆成『Software People』,Vol.2, 技術評論社, 2003.
- 山田正樹『Software People』,Vol.2, 技術評論社, 2003.
- Schwaber Ken、Mike Beedle著、Agile Software Development With Scrum (Series in Agile Software Development)
- 原田 保 、楠木 建、関西生産性本部 著　超社員術―会社に依存しない自律創造型「仕事人」への道　英二出版
- 野中 郁次郎、竹内 弘高 著, 梅本 勝博　翻訳　知識創造企業　東洋経済新聞社
- チームはどのように発展していくのかGersickのグループ発展モデル"Punctuated Equilibrium Model"http://www.geocities.co.jp/WallStreet/4716/teamdevelop.htm
- http://www.glocom.ac.jp/itplat/archive/0405sympo/panel1/2_nonaka_1.html キーノートスピーチ:「組織的知識から社会的知識創造へ」野中郁次郎
- http://dm.nikkeibp.co.jp/free/nde/kiji/no204/report08.html「日経デジタルエンジニアリング 4月号」

Copyright:© 2006 Hashimoto Software Consulting International Inc.