# MDAへの期待と課題

ソニー株式会社
橋本　隆成

# 私の立場~所属部署のミッション~

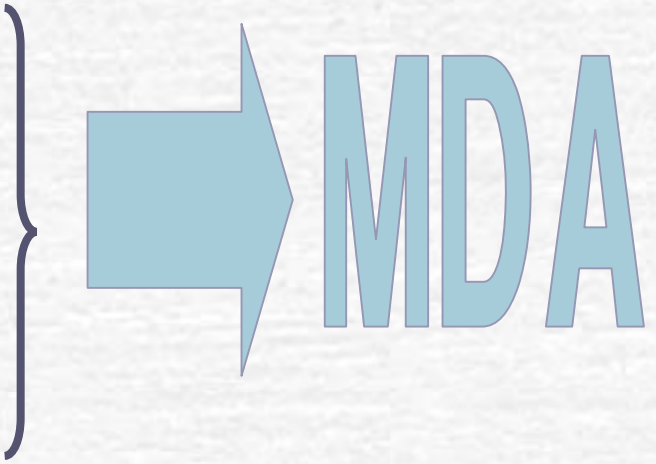
- ソフトウエアサイエンス基づく社内の開発手法の導入と推進
  - オブジェクト指向技術
  - CASE-Toolによる自動化
  - テスト設計&実施技法
- 品質保証/管理推進
  - CMM/CMMI、PMBOK、etc
  - 品質検証技術の確立(ISO9126、ISO15504、etc)
  - etc

# MDAは色々な解釈で用いられているが
# 我々のMDAの定義(期待)

- MDAをサポートするCASE-Toolの利用を前提
  - UMLの利用
  - 組込み・リアルタイムシステム向けオブジェクト指向CASE-Toolの利用を前提
  - 分析・設計の段階でシミュレーション可能
  - OS&プログラム言語非依存
  - 自動ソースコード生成
  - 自動テストToolや他のToolの連携
  - etc

# MDAへ期待する動機①

- 企業or部署の明確なゴール
  - 低開発コストの実現
  - 高品質の保証
  - etc
- ゴールを実現する戦略
  - 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
    - MDAによる自動化
      - 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へ期待する動機②

- 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
    - アジア諸国、インドとの競争力強化
  - 魅力ある製品開発業務に集中してもらいたい

# 方法論&CASE-Tool

- MDAでメジャーな組み込み・リアルタイムシステム向け方法論&CASE-Toolの検討の注意点
  - ポイント(何を期待しているのか)を絞らない比較は無意味
    - 組み込み・リアルタイムには色々なシステムがある
    - それぞれ狙いとターゲット対象がある
    - サポートするオブジェクト指向方法論のコンセプト&ポリシーに依存する
  - 全てに優れたToolは無いのが現状

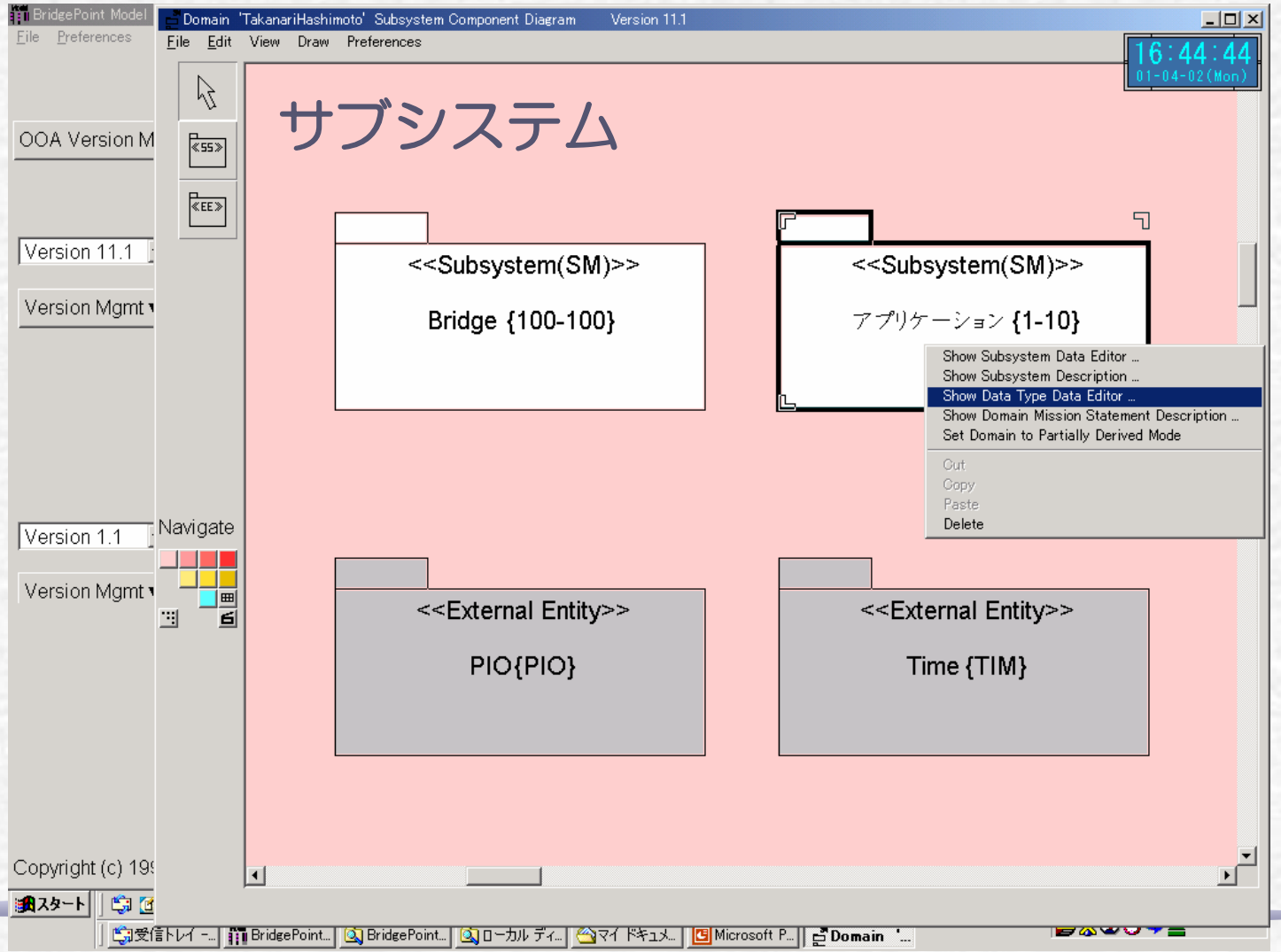
# 方法論&CASE-Toolの特徴&紹介

# MDAとオブジェクト指向方法論
# 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

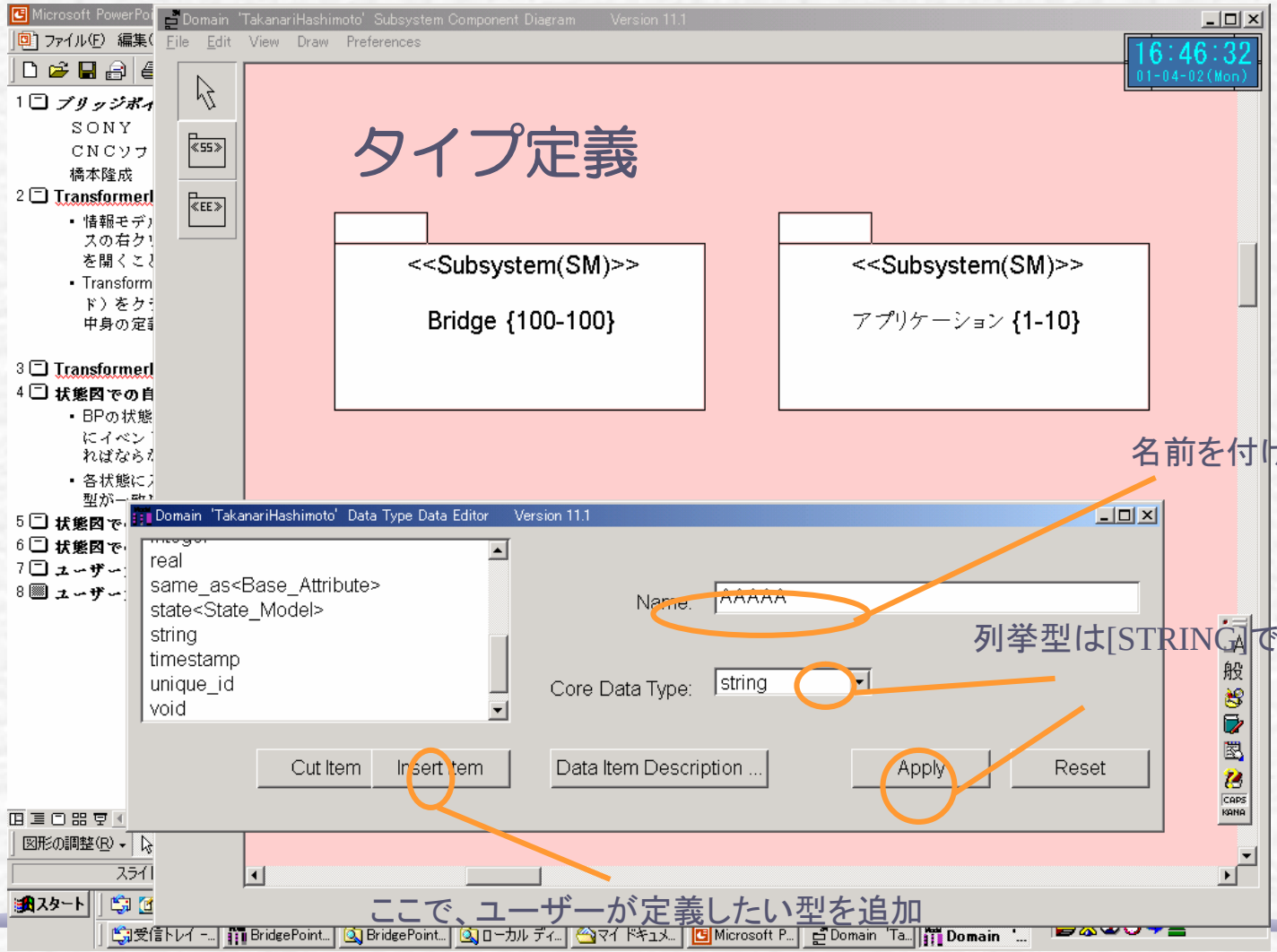
# MDAとオブジェクト指向方法論

## XUML(シュレイアー&メラー法)



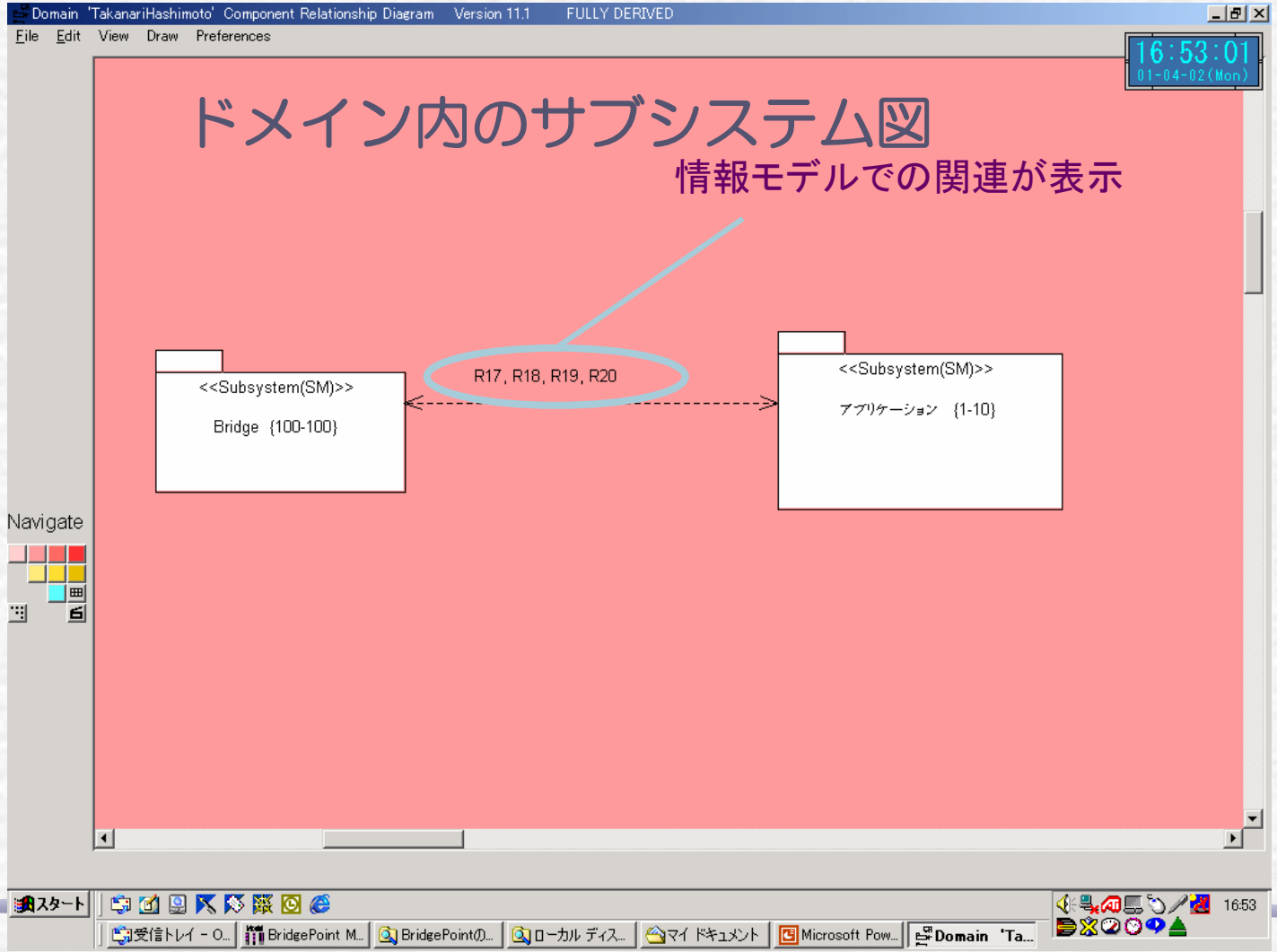
# MDAとオブジェクト指向方法論

## XUML(シュレイアー&メラー法)



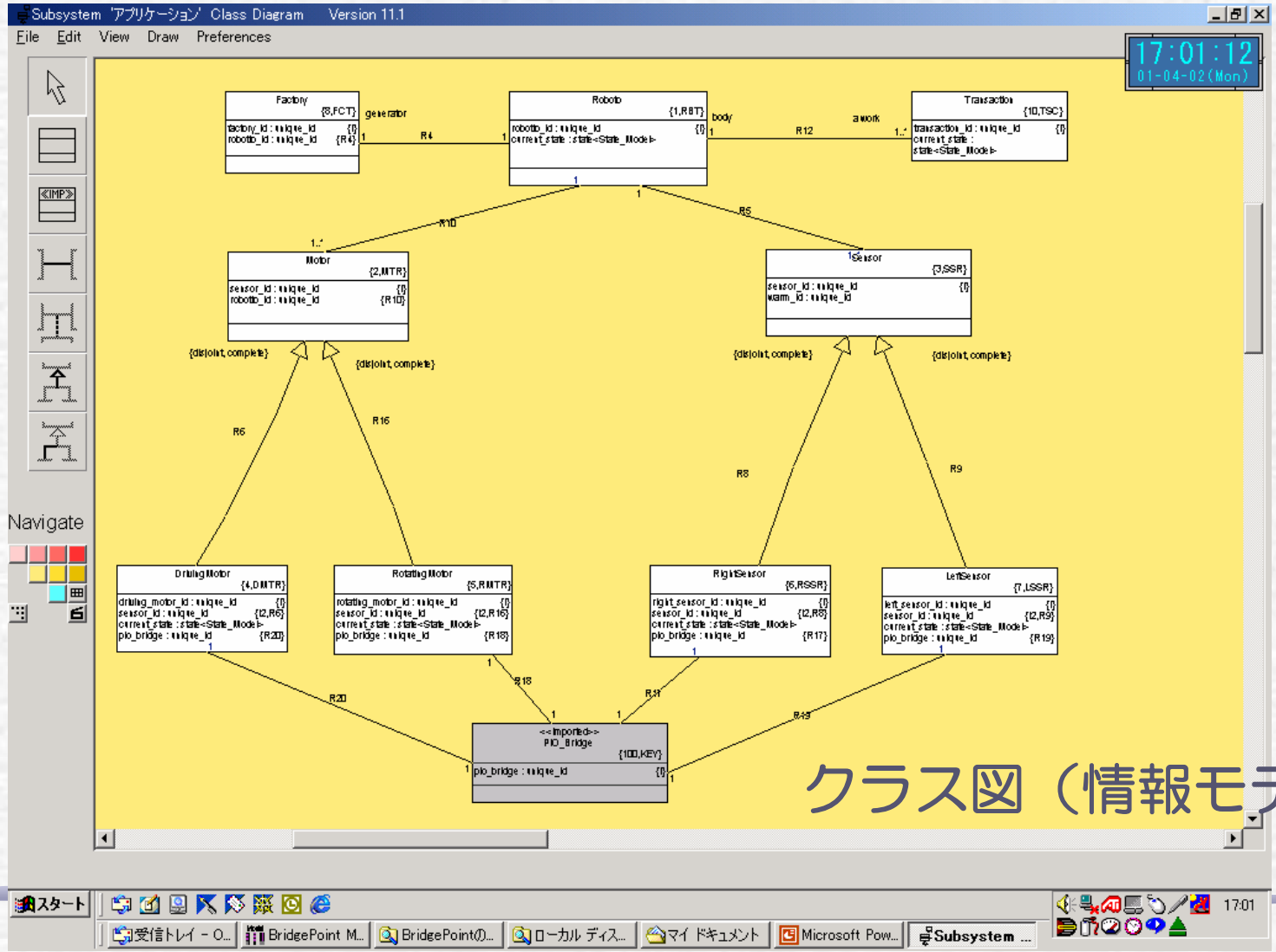
# MDAとオブジェクト指向方法論

## XUML(シュレイアー&メラー法)



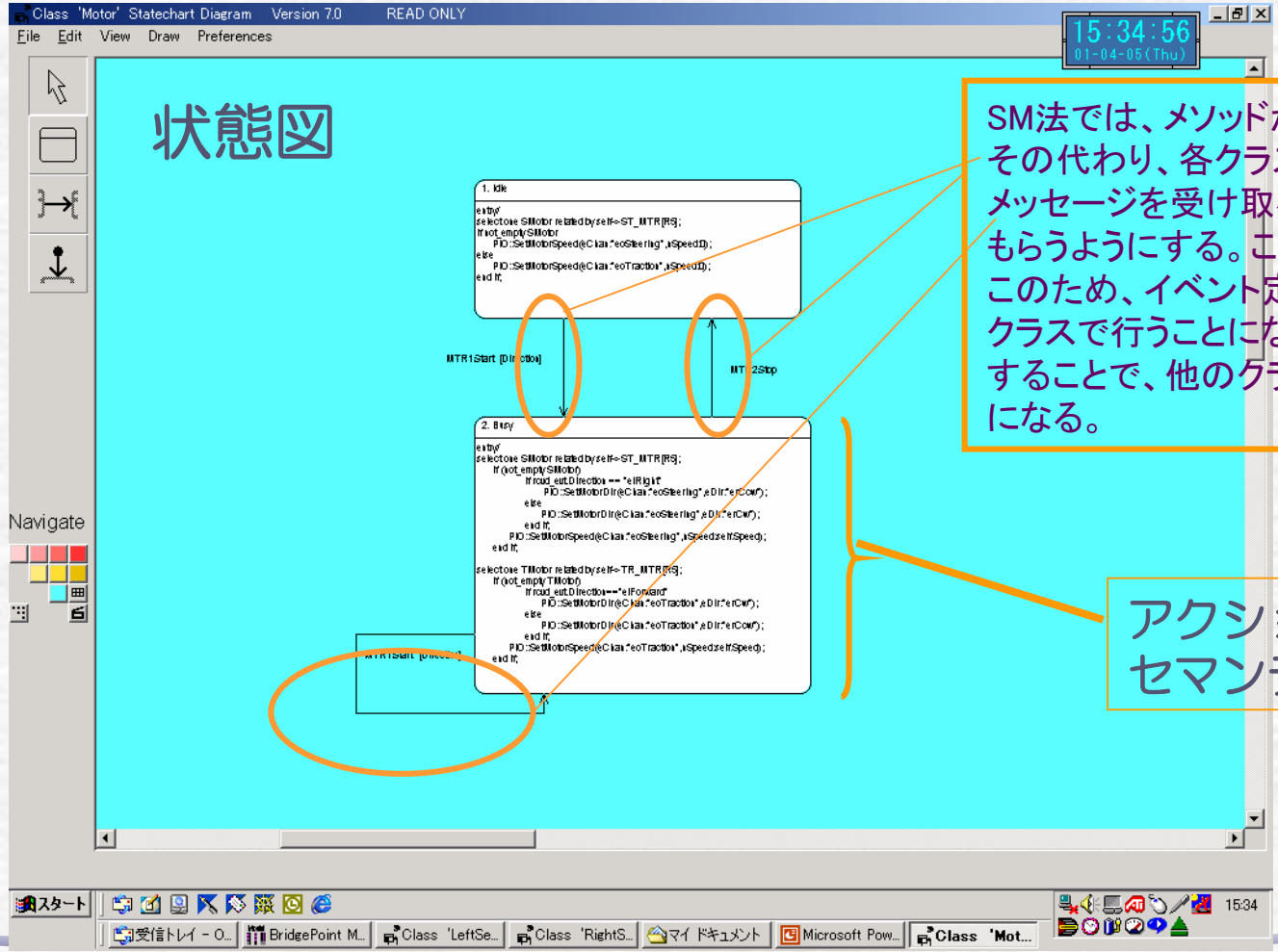
# MDAとオブジェクト指向方法論

## XUML(シュレイアー&メラー法)




クラス図（情報モデル）

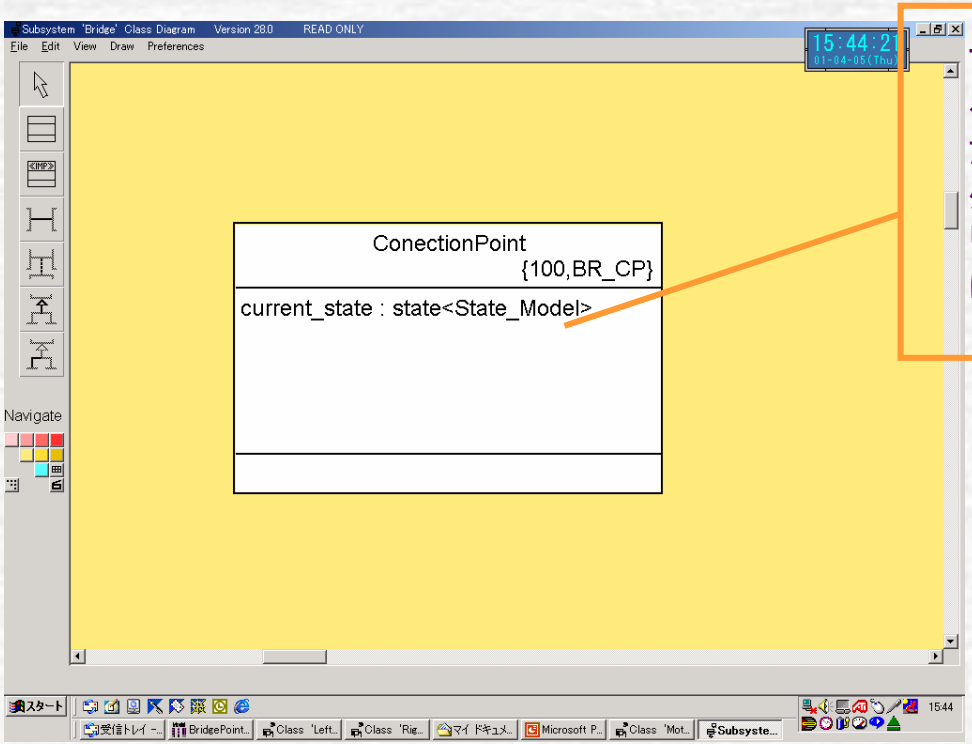
# MDAとオブジェクト指向方法論
## XUML(シュレイアー&メラー法)




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。

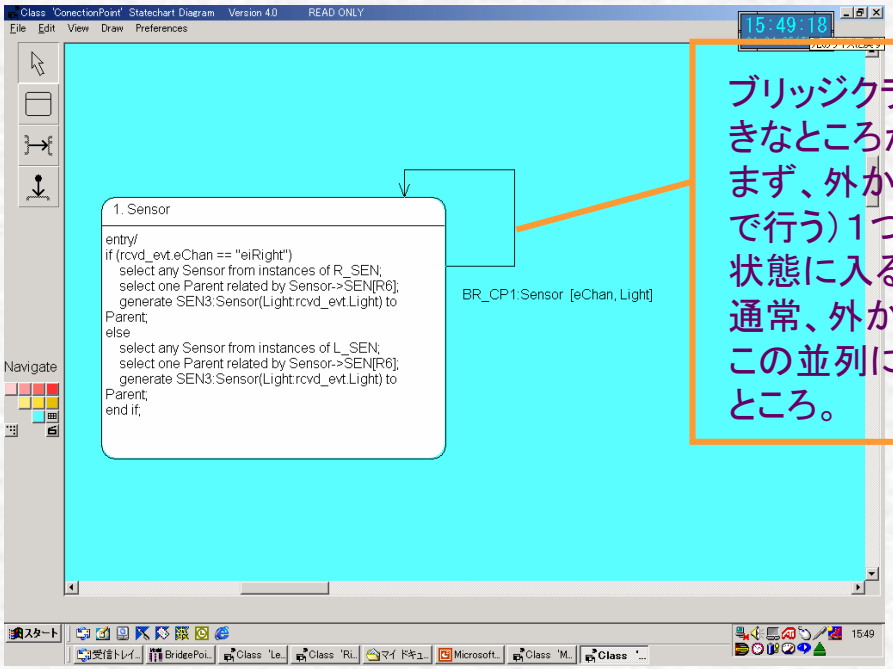
アクション
セマンティック言語

# MDAとオブジェクト指向方法論
## XUML(シュレイアー&メラー法)




ブリッジのサブシステムには、クラスを1つ用意する。このクラスには、外からのイベントを受け取るための入り口となる。
外からのイベントを受け取り、処理するのでこのブリッジクラスは、状態をもつアクティブオブジェクトにする。

# MDAとオブジェクト指向方法論
## XUML(シュレイアー&メラー法)



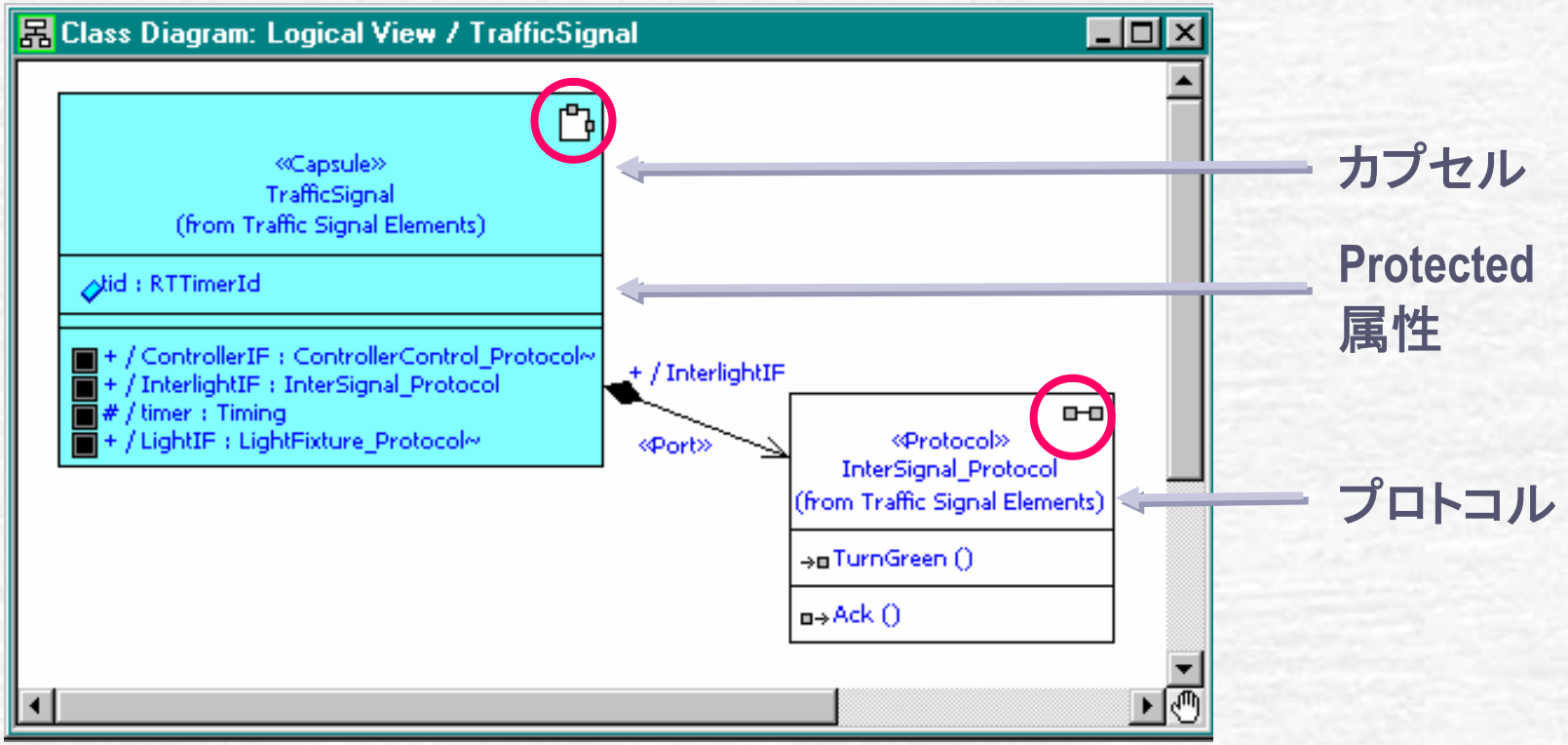

ブリッジクラスは、SM法の中でもたのクラスとは違い、少し特徴できなところがある。
まず、外からのイベント(ただし、イベントの定義はブリッジクラス側で行う)1つにつき、1つ状態を用意する。そして必ず自分自身の状態に入る再起的なイベントの描き方をする。
通常、外からのイベントは複数あるので、状態は複数並列に持つ。この並列に状態を複数持つところが、他のクラスと少し違っているところ。

# MDAとオブジェクト指向方法論 RoseRealTime(ROOM)

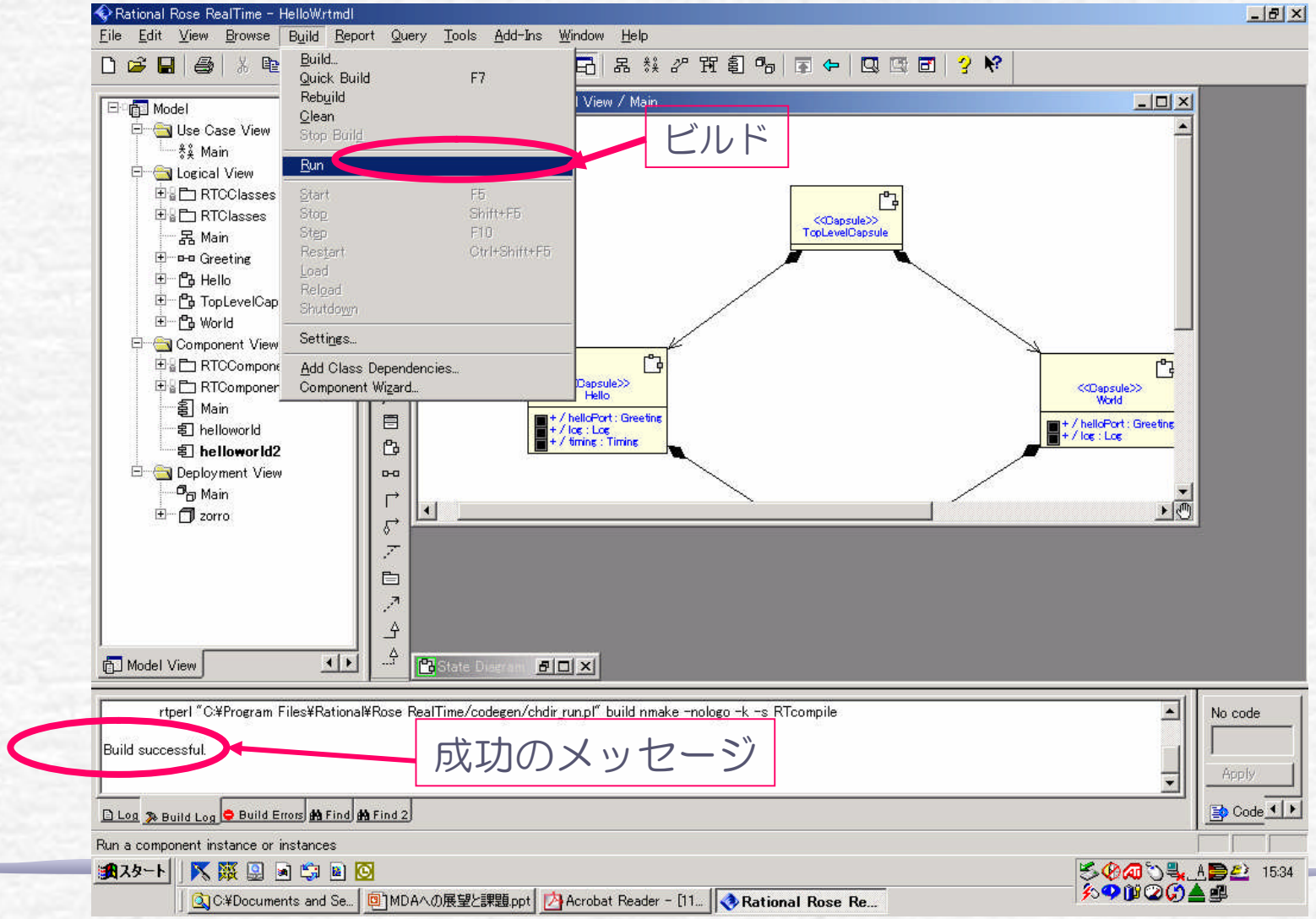
- 特徴
  - 並列／並行性を積極的に対応
    - マルチCPU開発環境を考慮
    - 方法論がMDAを前提
  - コンポーネントレベルの再利用
    - 設計レベルでモデリング
    - 静的構造と動的な振る舞いの両方をモデリング
  - UMLを拡張したアイコンを使用
    - カプセル、ポート、プロトコル、etc.の追加
      - 分析より設計が充実
      - 並行動作するタスクとタスク間通信をモデリング
    - モデル上でシミュレーション
  - 状態図にターゲット言語でアクションを記述

# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)

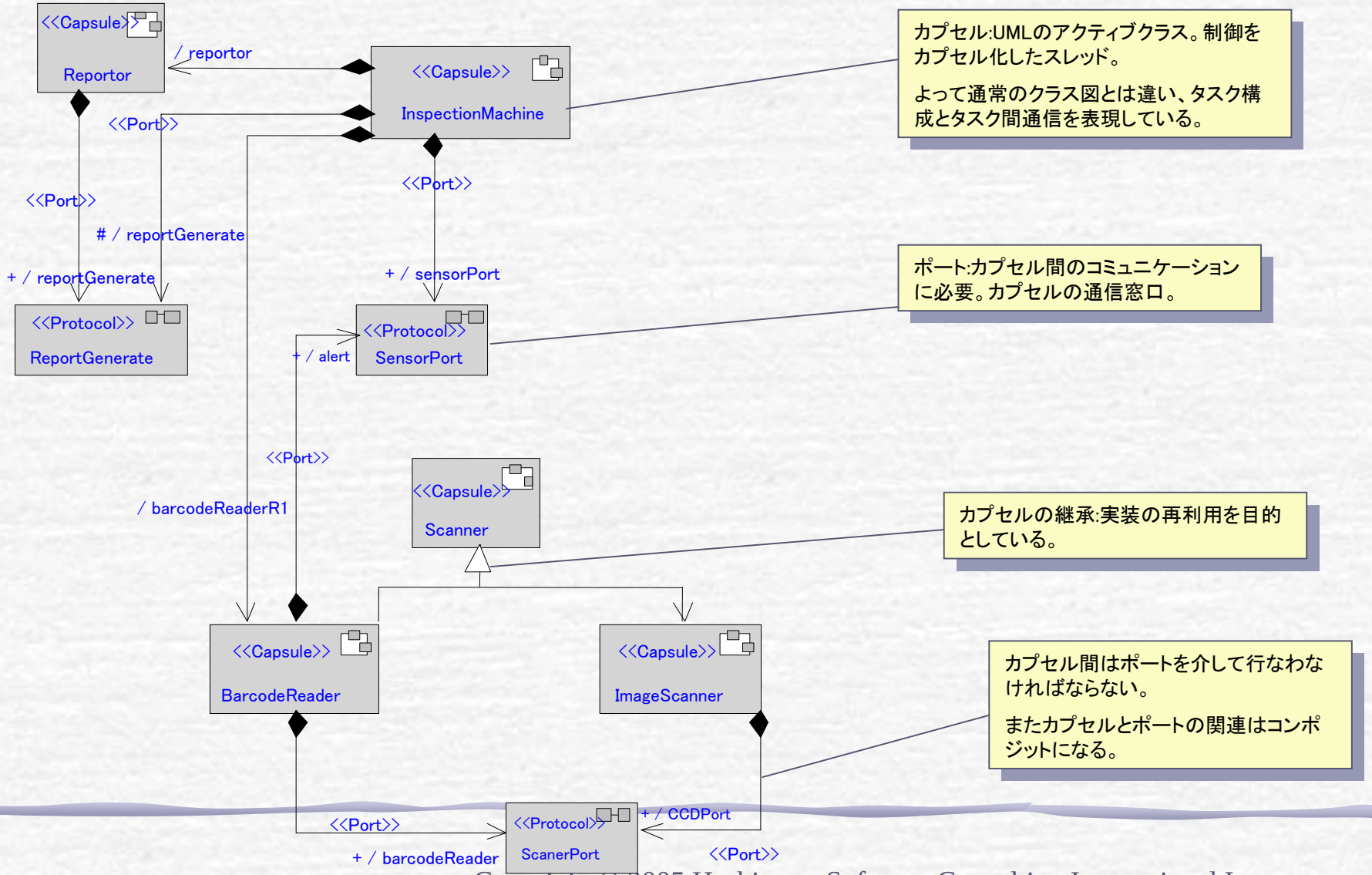
- リアルタイム/組み込み開発用にUMLを拡張
- 並行性および分散性にカプセルとプロトコルを使用



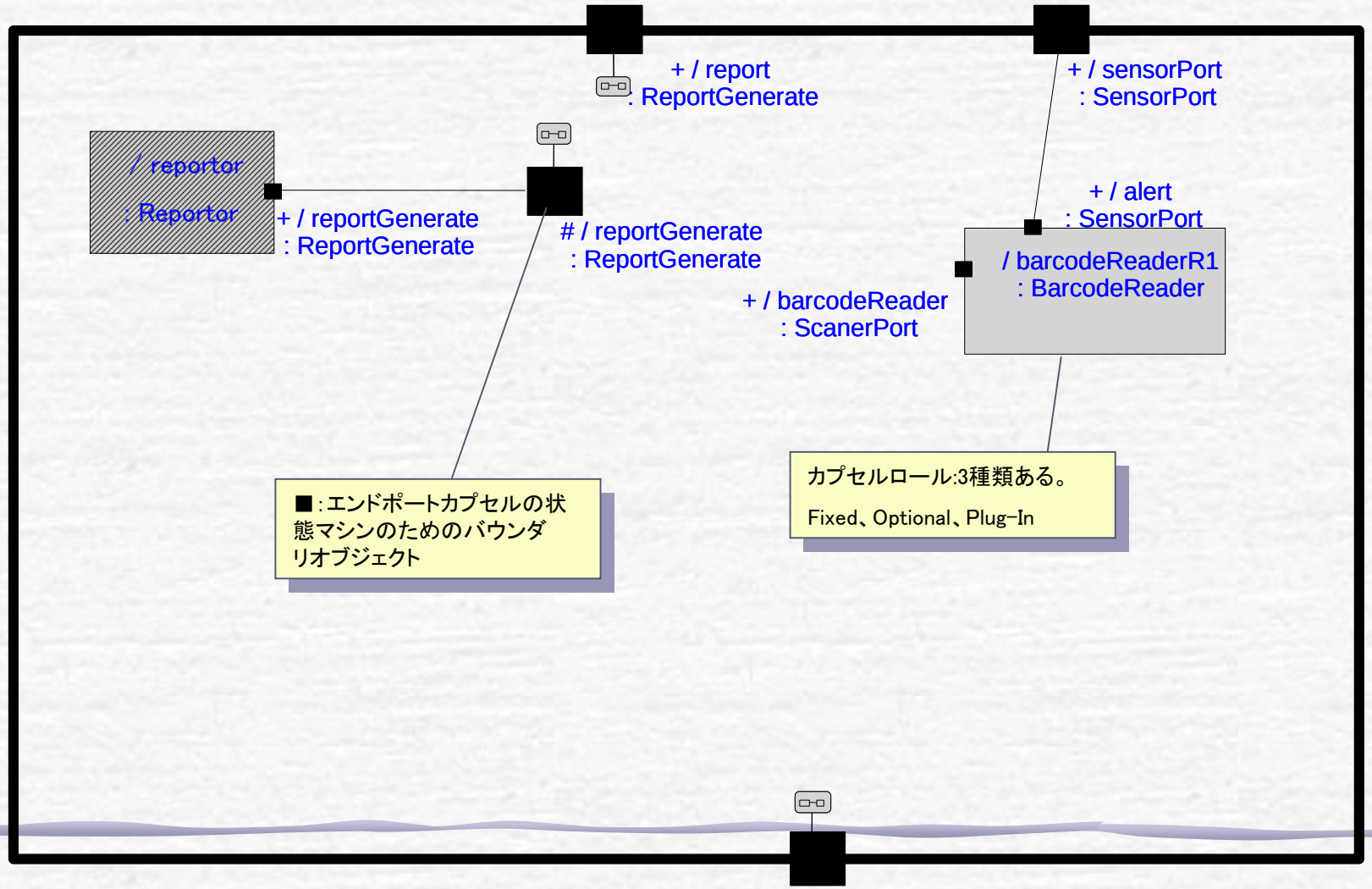
# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)



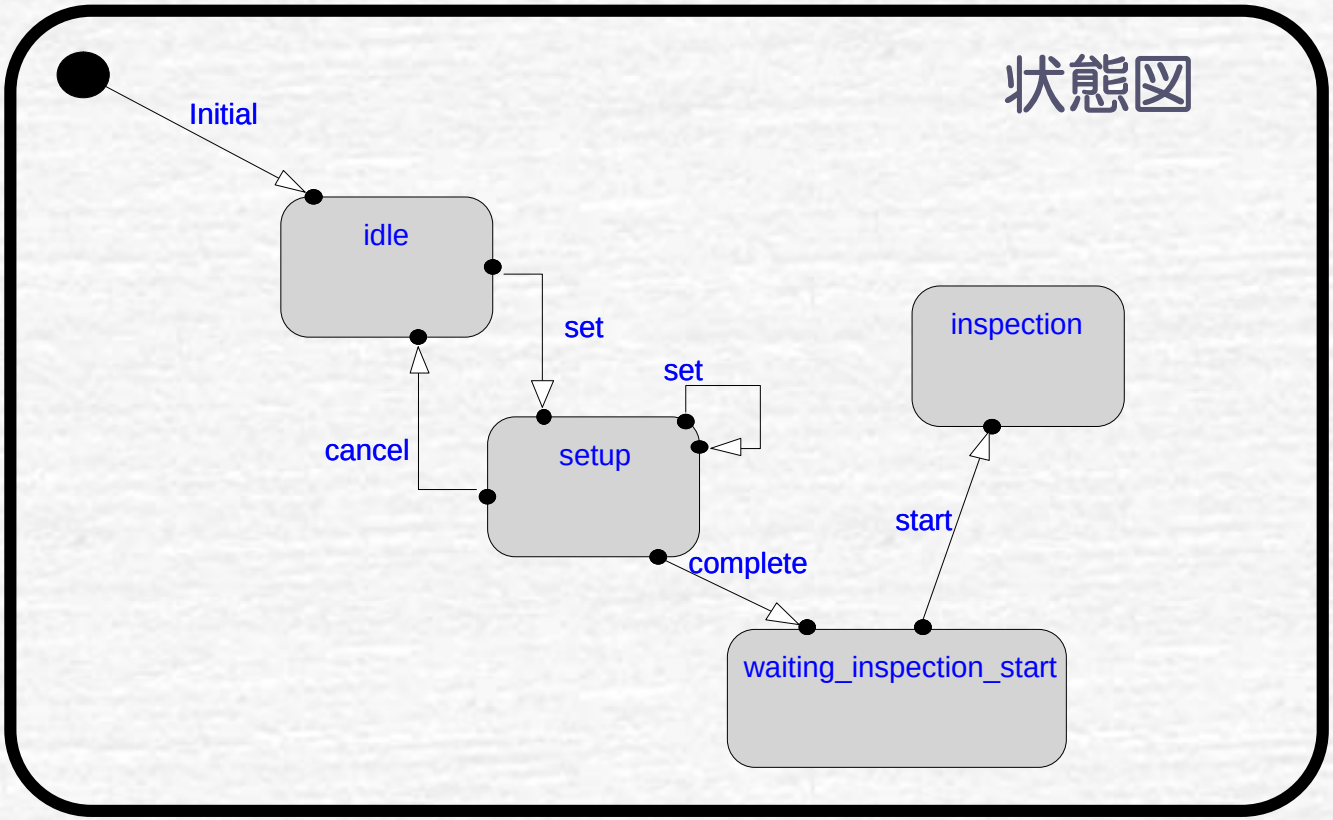
# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)



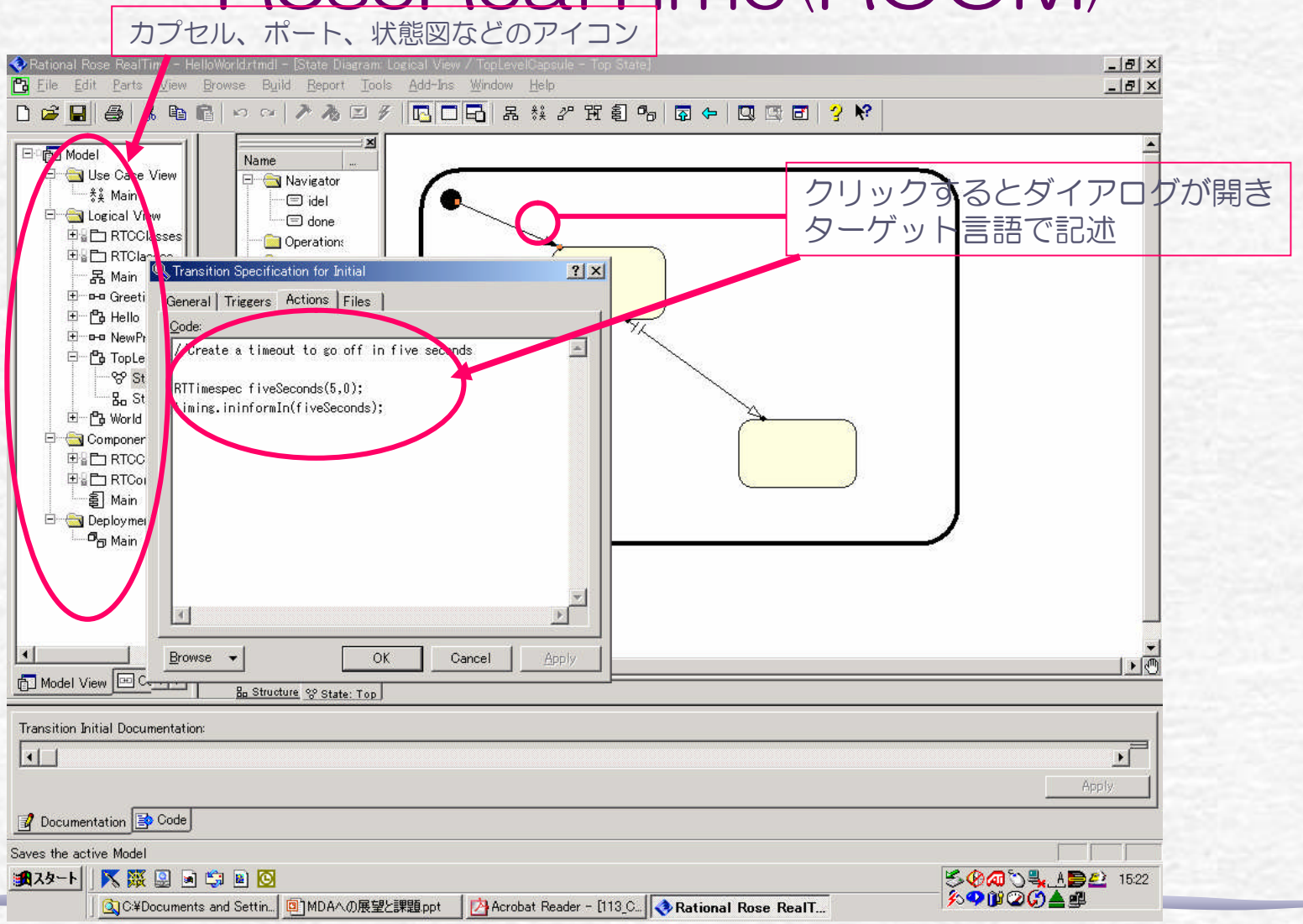
# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)



# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)



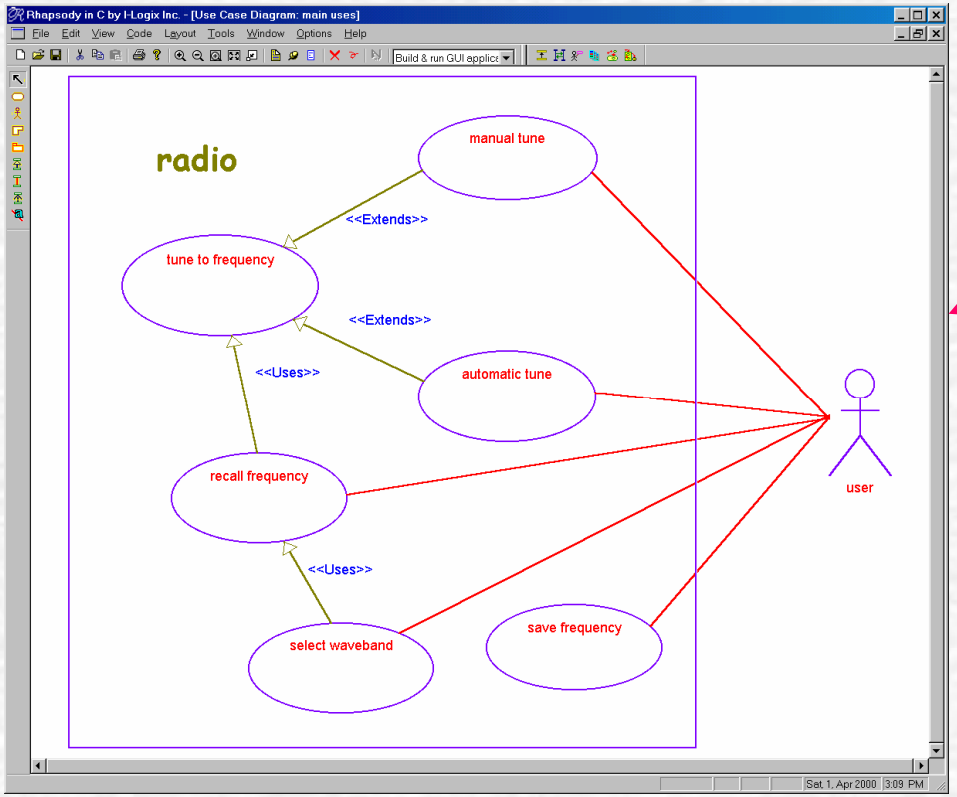
# MDAとオブジェクト指向方法論
# RoseRealTime(ROOM)



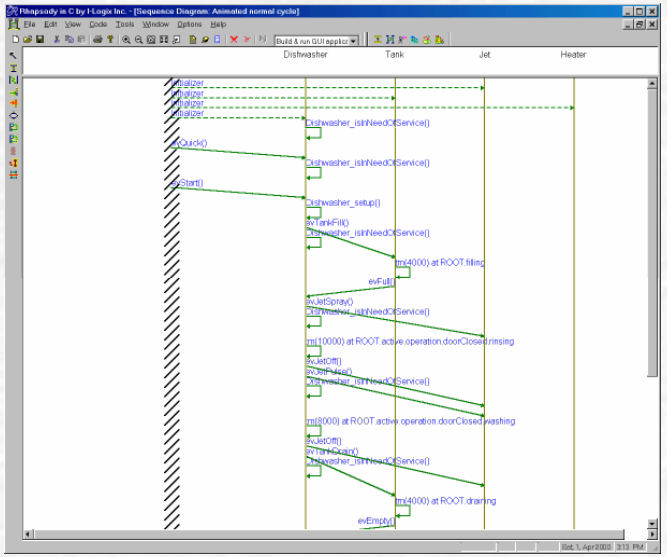
# MDAとオブジェクト指向方法論
# ROPES

- 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

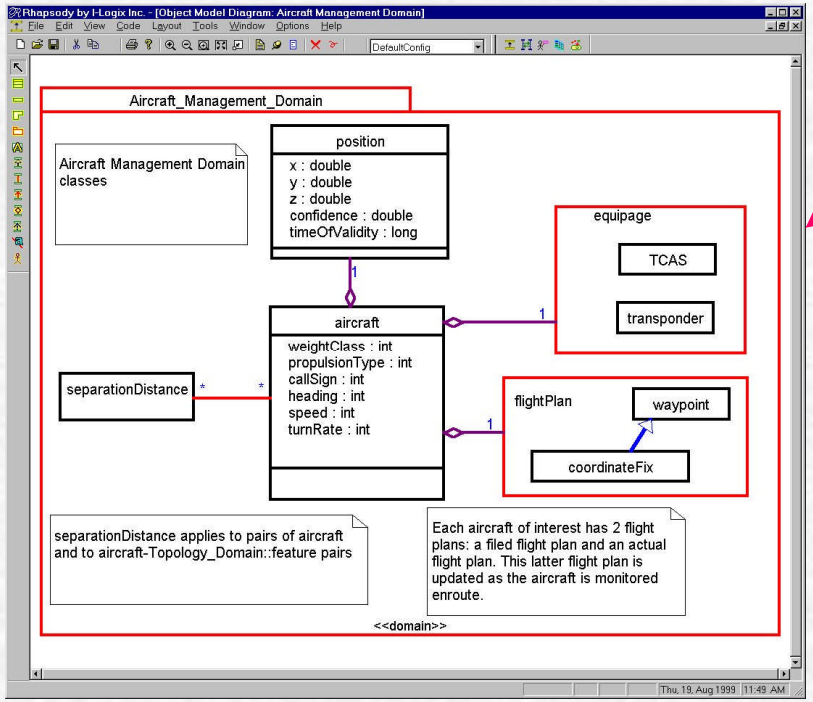
# MDAとオブジェクト指向方法論
# ROPES



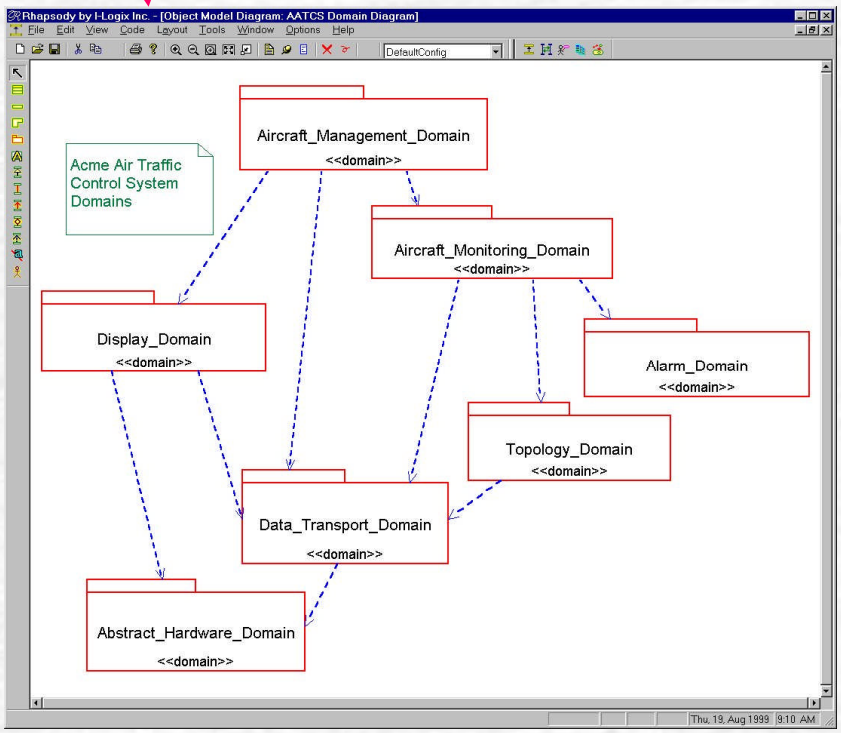

**UseCase駆動型開発**



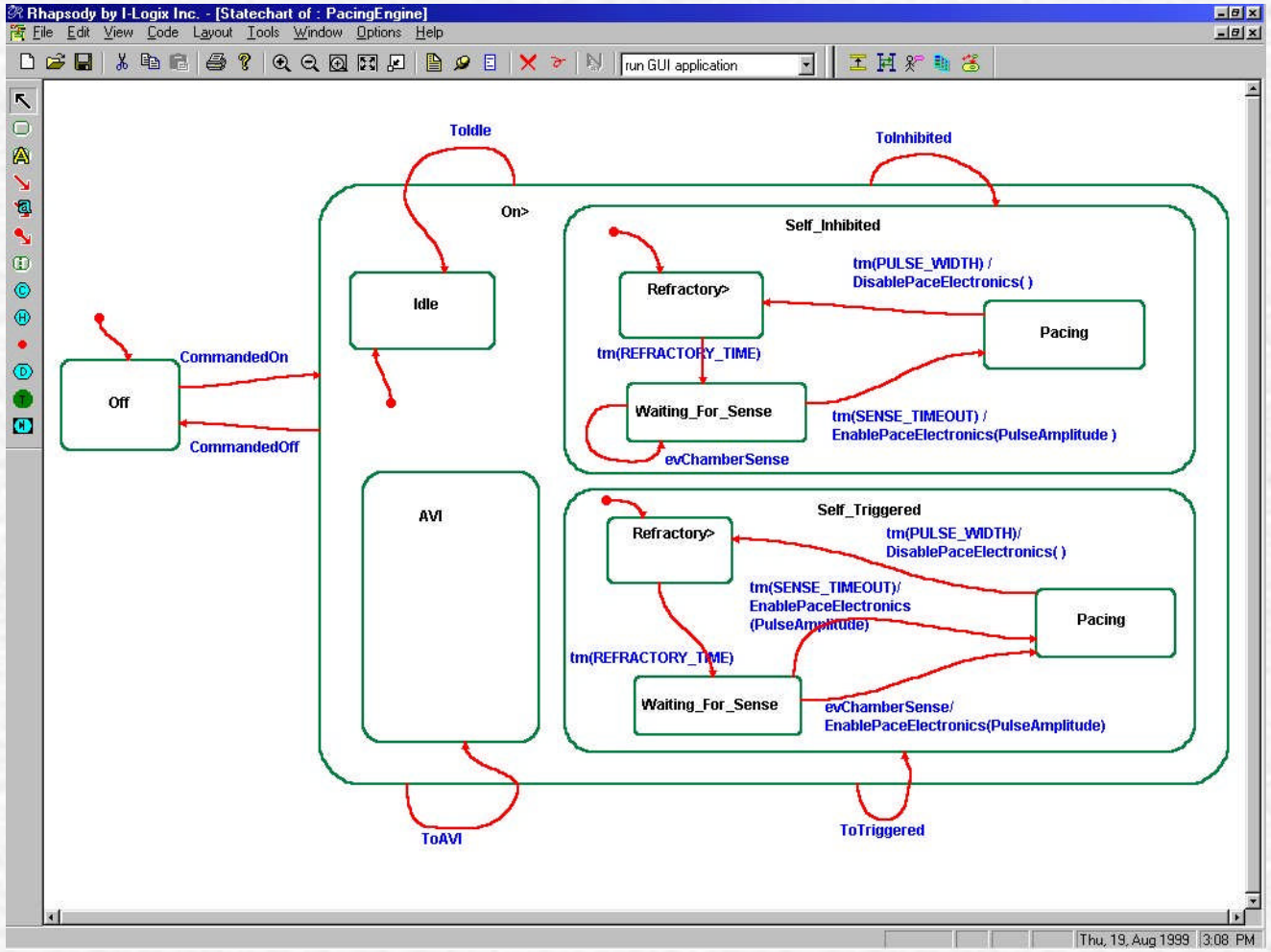
# MDAとオブジェクト指向方法論
# ROPES



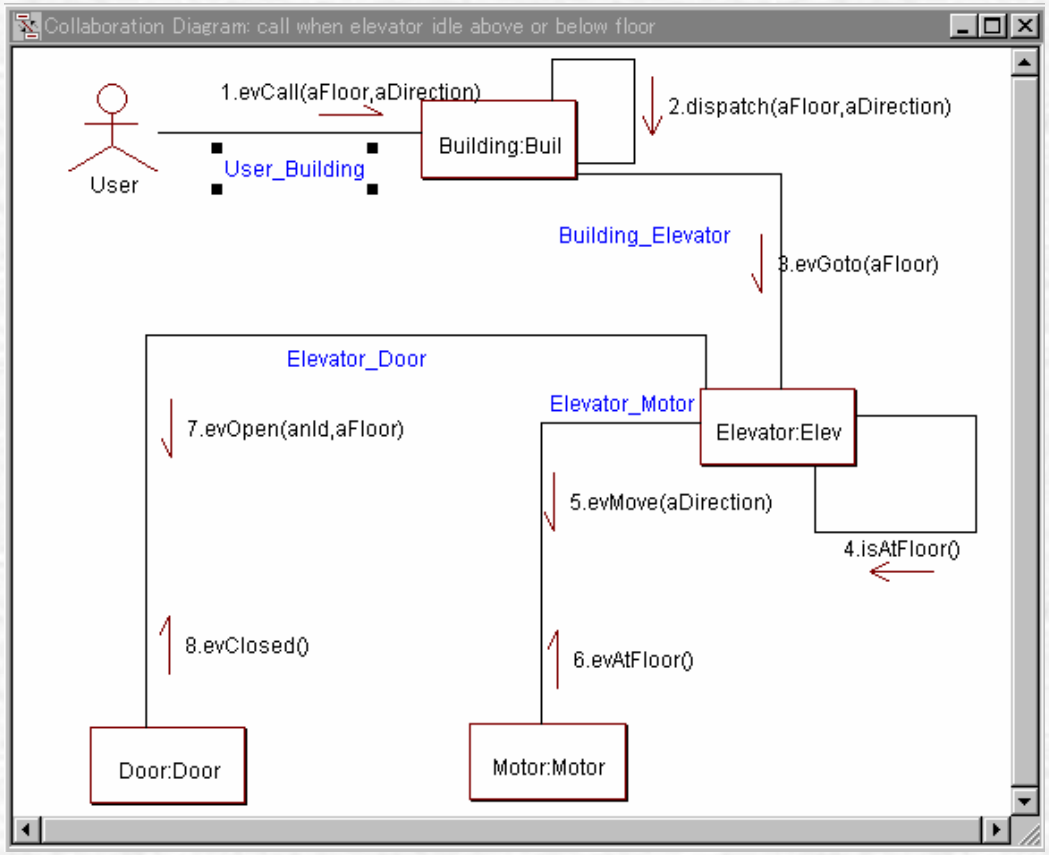

静的構造はクラス図、パッケージ図

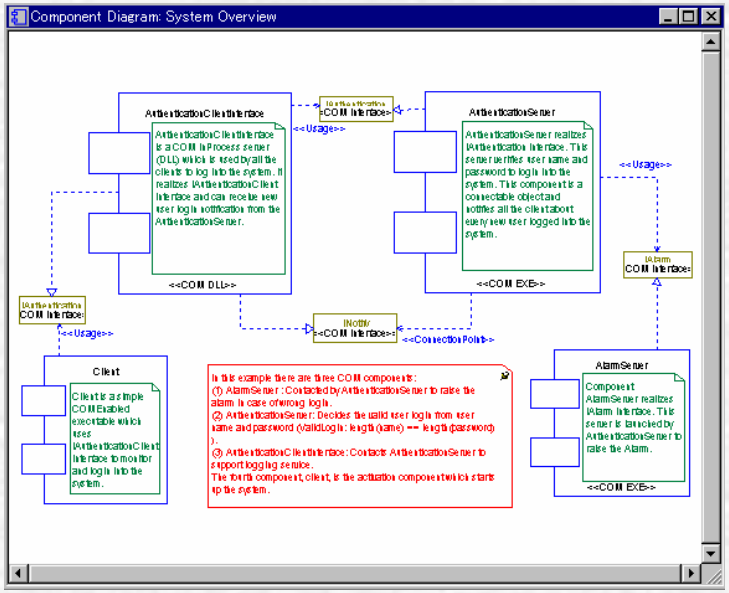


# MDAとオブジェクト指向方法論
# ROPES

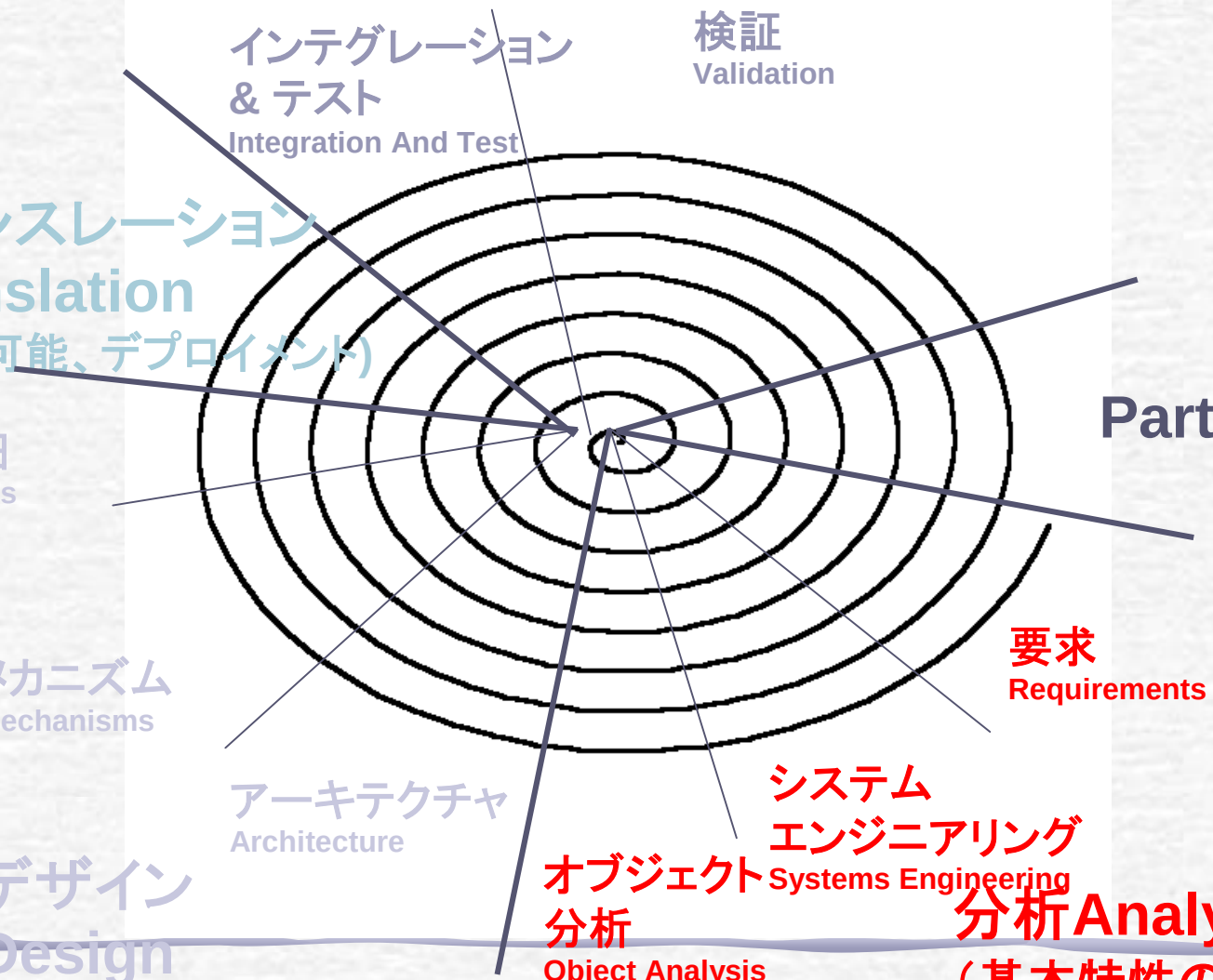


# MDAとオブジェクト指向方法論
# ROPES

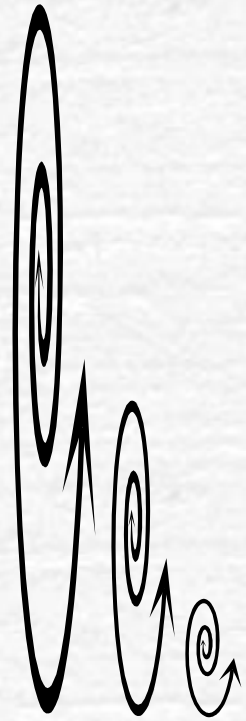


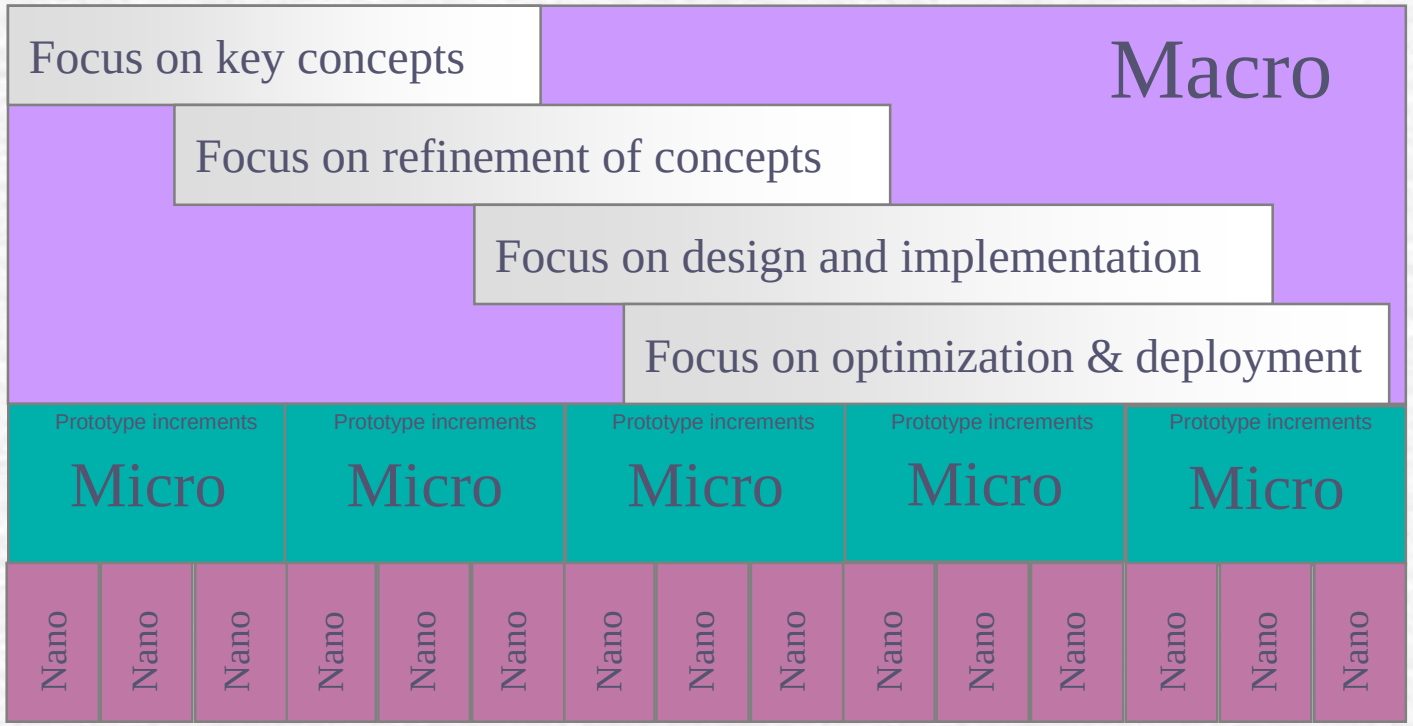



# オブジェクト指向方法論の例
# ROPES



# オブジェクト指向方法論の例
# ROPES

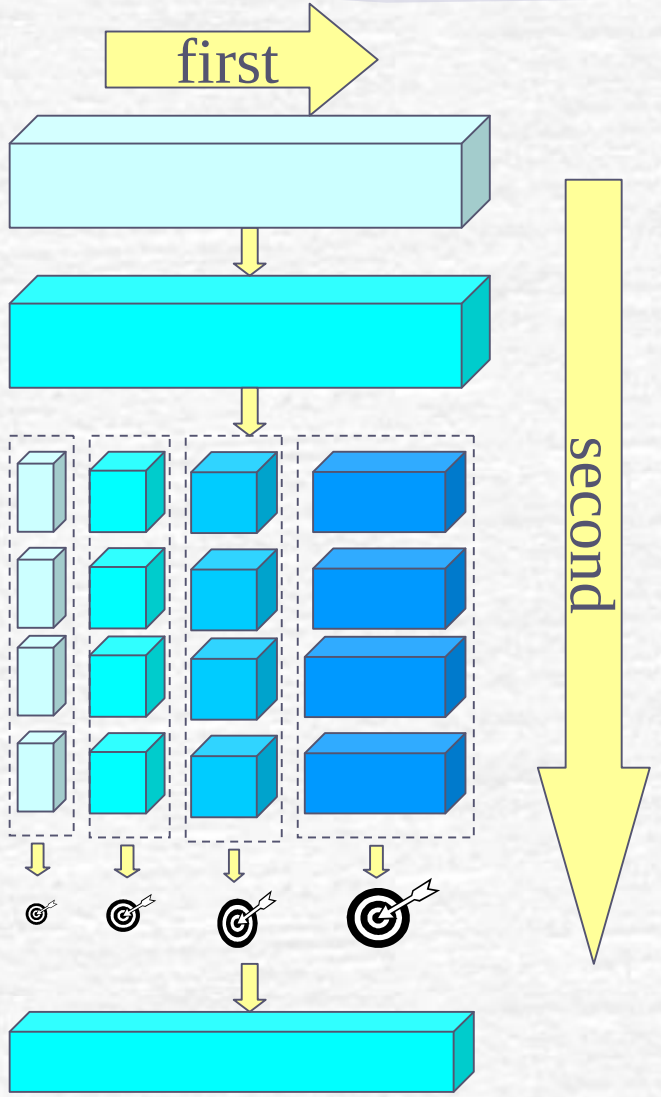




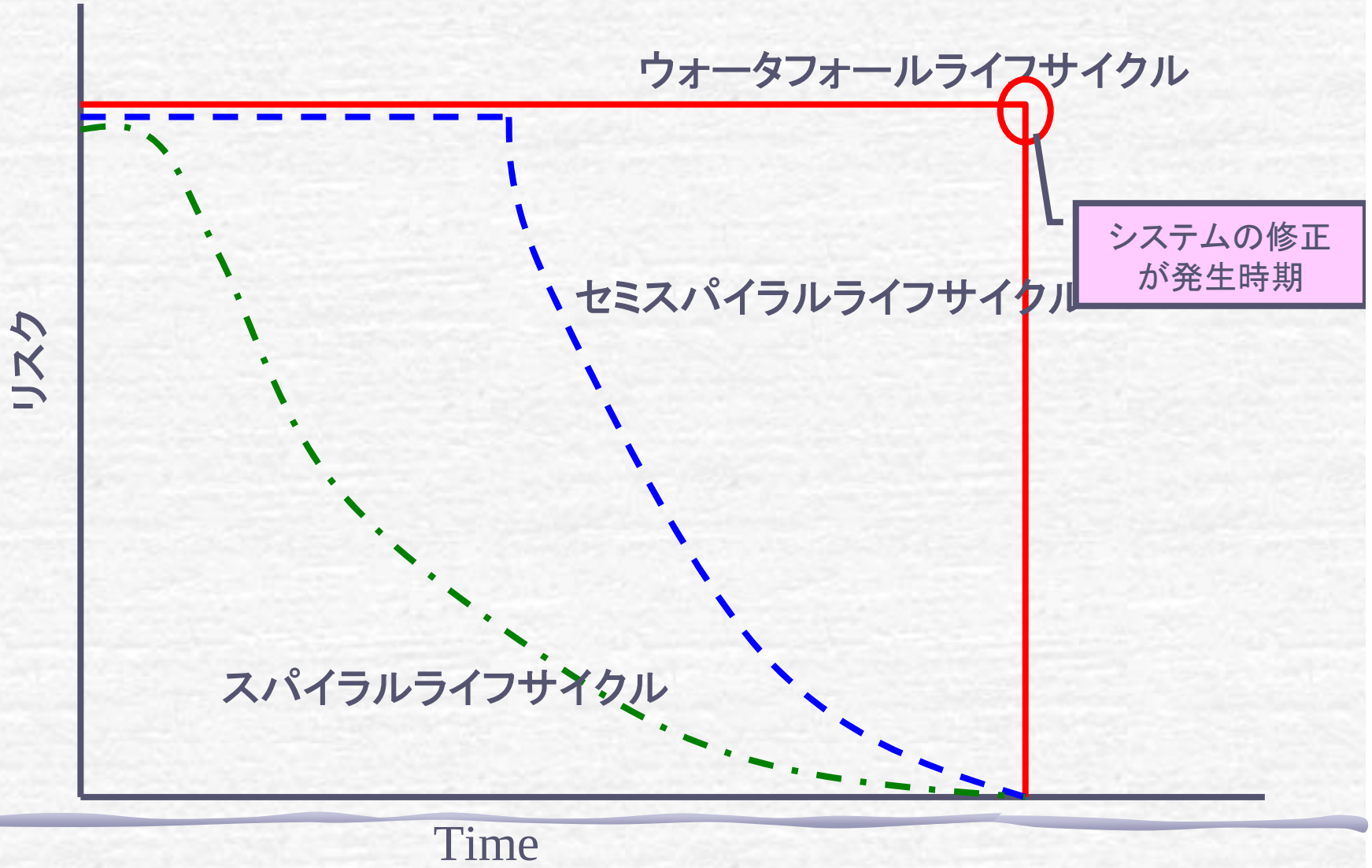
# オブジェクト指向方法論の例
# ROPES

- スパイラルの利点を適用するが、完全にはスパイラルを適用しない
  - 大規模で複数のチームを構成するプロジェクト
  - コ・デザインを行うことが重要
  - 一つの段階に適用

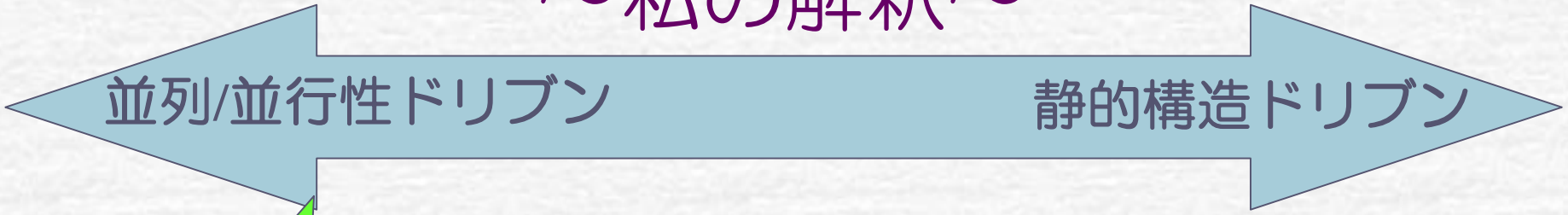
    (Funding for only one phase)
- 分析はウォータフォールで実施される
- 設計はスパイラルで実施される



# オブジェクト指向方法論の例
# ROPES



# 3つの方法論の特徴
## ～私の解釈～




システムの特徴にあわせて選択が現実的

シュレイアー&メラー法

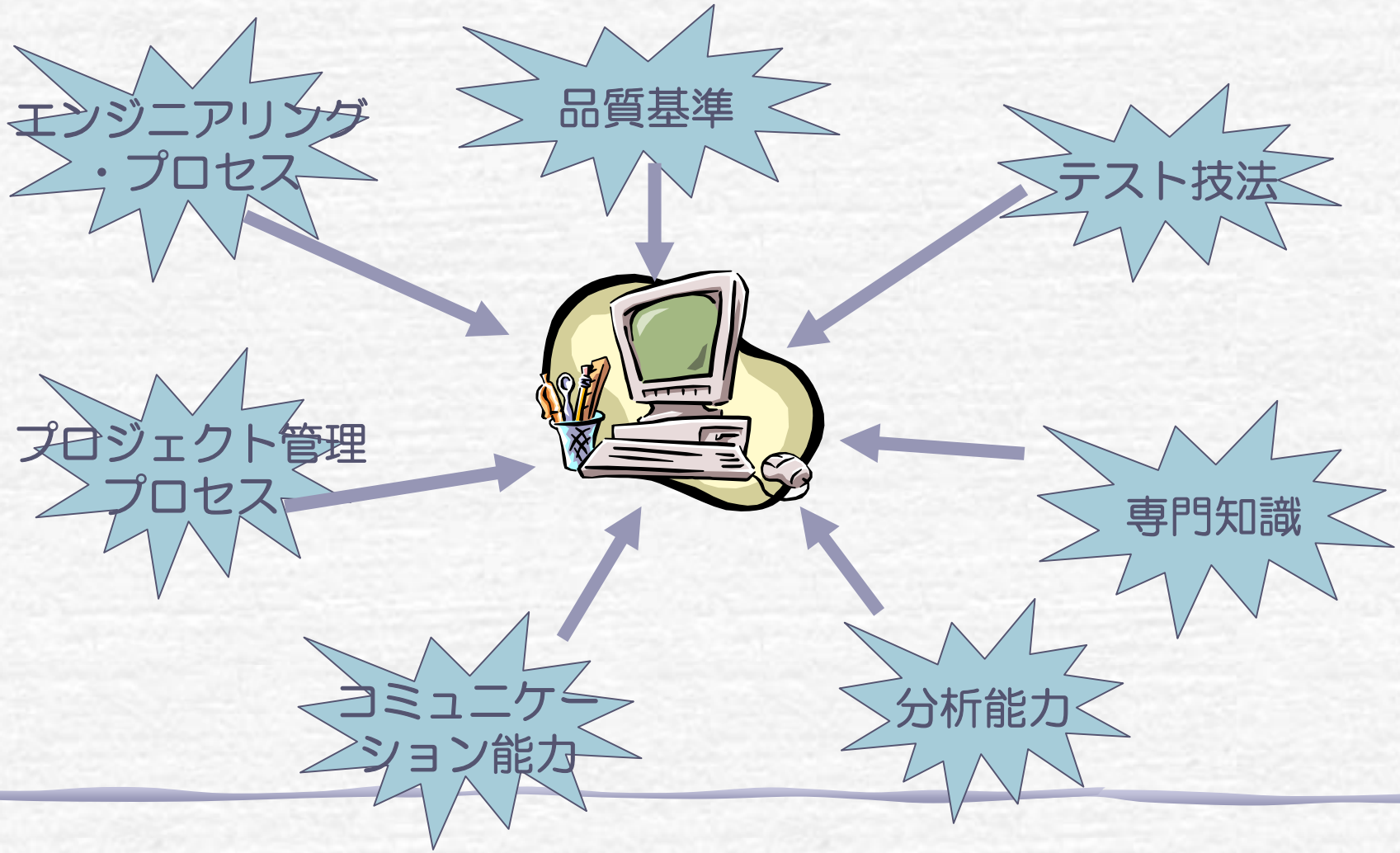
ROPES

Rose Real Time（ROOM）

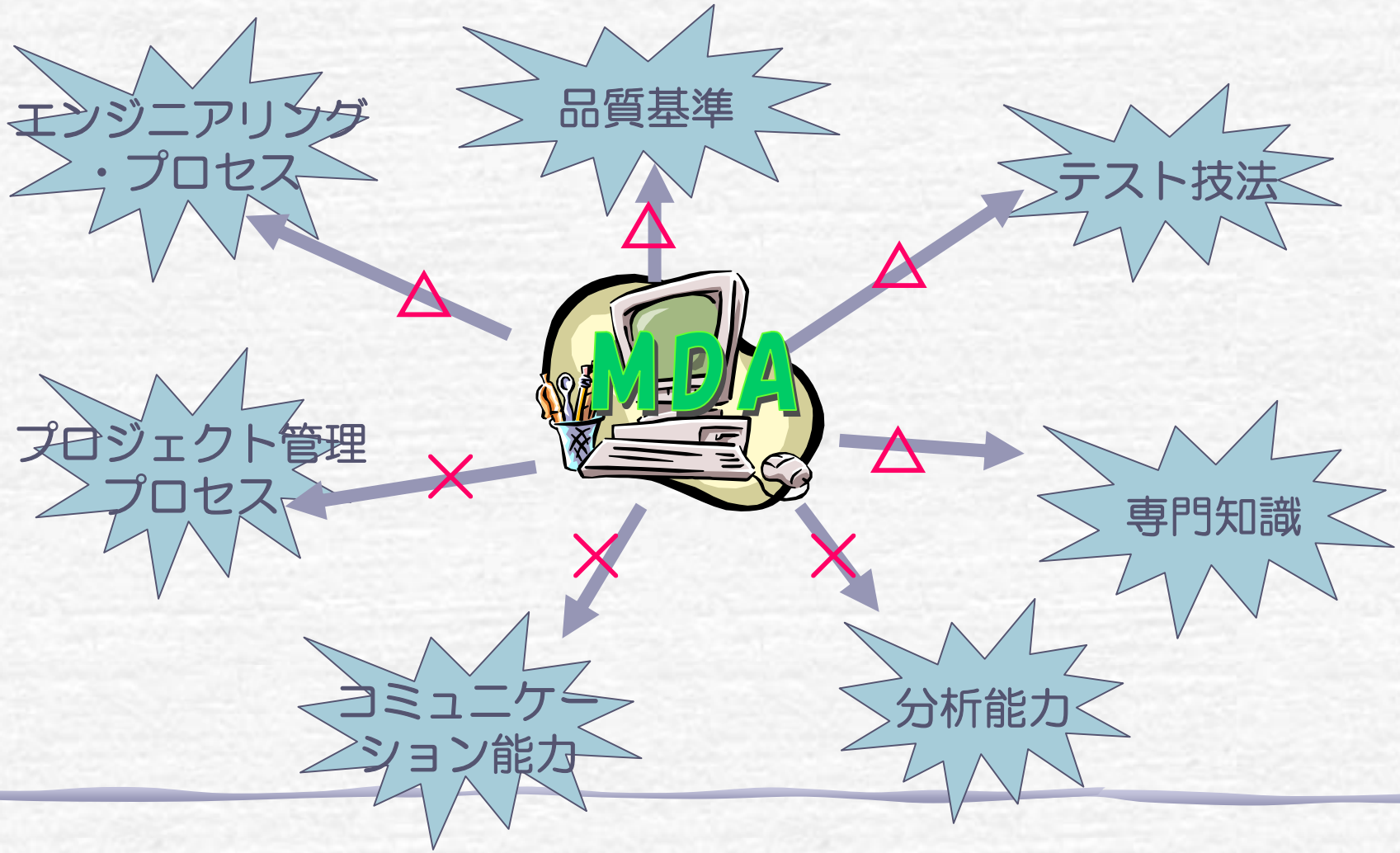
いずれの方法論も進化していくと考えられる

# まとめ
# ～MDAへの期待と課題～

# 高品質・生産性向上のために
# ソフトウエア開発に必要な要素

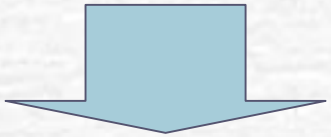


# MDAで何が解決できるか？



# MDAの限界(?)

- 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、Agileプロセス、etc
- 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！*

# MDAの課題①

- 各ベンダーのCase-Tool毎のアーキテクチャーに依存
  - 自動ソースコード生成メカニズム
  - シミュレーション方法のポリシー&メカニズム
  - オブジェクト指向&MDAの哲学
- 互換性が無いのが現状
  - オブジェクト指向方法論に依存
  - 各ベンダーのCase-Toolに長所&短所が存在
  - 開発製品の特徴に応じてCASE-Toolを使い分けることが必要

# MDAの課題②

- MDAはシミュレーション&テストも自動化されることが課題&期待
  - 自動ソースコード生成されたものを人がデバックはナンセンス
  - 上流で可能な限り評価&検証したい
- CASE-Toolがまだ発展途上の状態にある
  - ソースコードサイズ
  - パフォーマンス速度
  - 理想と現実の機能に、まだ大きな差がある
  - システム全体にMDAを適用するにはまだ課題が多い
    - 効果的な適用に工夫が必要

# MDAの課題③

- CASE-Toolの適用ノウハウをナレッジ・マネージメントする必要性
  - 教科書が無い
  - トレーニングが不十分
  - Toolのマニュアルが貧弱
- CASE-Toolベンダーのサポート、技術力に大きく依存
  - 自社内で解決できない問題も存在
- CASE-Toolと方法論のトレーニングやノウハウの初期投資が必要
  - 決して小さくない学習期間と初期投資額
  - CASE-Toolの制約やアーキテクチャを理解することが必要

# MDAのCASE-Tool①
## ～組込み・リアルタイムシステム向けCASE-Tool ～

- 組込み・リアルタイムシステム向けCASE-Toolに見られる共通的特徴
  - どの『方法論』や『CASE-Tool』にも多くの魅力と課題がある
    - 利用者のシステムやプロセスに適切なものを選択することがキー
  - ソースコード自動生成機能がある
    - ターゲットマシン用のソースコード生成
  - オブジェクトの状態マシンを重視
    - 状態図にアクションを記述する
    - ソースコード生成のもととなる
  - タイマー、イベント、イベントキュー、etcなどシステムに必要なインフラは提供されている
    - 各CASE-Tool独自
    - 何らかのシミュレーション機能を装備している

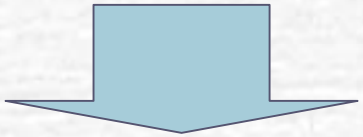
# MDAのCASE-Tool②

## ～組込み・リアルタイムシステム向けCASE-Tool ～

- テストCASE-Toolとの連携が進められている
  - まだ発展段階
- 多機能、高価
  - ソースコード生成、シミュレーション、ホスト-ターゲット環境への対応、C/C++、Java、Adaなどの対応
- 使いこなすのに(ある程度)時間がかかる
  - CASE-Toolのもつメカニズムの理解
  - トレーニングがやや不十分
  - 資料がやや不十分
  - 自分達のエンジニアリング・プロセス、管理プロセス、文化に融合させるにある程度試行錯誤が必要
    - *必ずしも全てを取り込まなくても良いはず*

# MDA導入の前に①
## ～MDAを成功させる条件～

- 企業の目的と戦略の明確化
  - オブジェクト指向技術、MDA技術をなぜ導入したいのか？
    - 初期投資や学習コストがそれなりにかかっても明確なビジネスの狙いがあれば“取り組む価値は大”
  - どのように利用するのか？
    - 自分達の開発プロセスのどこに適用するのか？
    - 新たにプロセスを導入するのか？
  - 具体的な導入スケジュールとマイルストーン策定



**戦略なき導入は成功が困難！**

# MDA導入の前に②
## ～MDAを成功させる条件～

- MDAとCASE-Toolの過度の期待はしない
  - ＭＤＡを導入には*充分な調査と投資期間の認識*
  - 初期投資や学習コストがそれなりにかかる
    - CASE-Tool自体も高額
    - 急激な生産性の期待は無理
  - MDAやCASE-Toolに引きづられないように
    - CASE-Toolは『ある特定の方法論』をサポートしている
    - ブームやトレンドに関わらず、自分達の*システムの特徴に適したもの*を
    - テストや品質は*今のところは、*MDAやToolだけでは解決できない
      - テストデーターの抽出は設計者が行なう
      - テストもまだエンジニアが現実的には実施
      - テストの上流が大切
      - 分析、設計は*エンジニアの能力にかかっている*