SEAMAIL

Newsletter from Software Engineers Association

Vol. 13, Number 6 - 7 July, 2002

目　次

編集部から		1
Special Workshop on UML		2
□ UML解説 － その肝と心	羽生田 栄一	5
□ モデルとUML	玉井 哲雄	66
□ UML の大学教育	中谷 多哉子	76
□ UML の実践教育	桜井 麻里	82
□ プログラマから見た UML	東田 雅宏	85
□ OMG における UML の位置付け	伊藤 友里恵	90
□ UML を用いたシステム開発事例	石橋 民男	102
□ 組込み用にアレンジした eUML の狙い	渡辺 政彦	120
□ 開発ツール/プロセスと UML	伊藤 昌夫	149
□ システム分析設計と UML	酒匂 寛	160
Workshop Report	熊谷 章	167

ソフトウェア技術者協会
Software Engineers Asociation

ソフトウェア技術者協会 (SEA) は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985年 12月に設立されました．

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約 200人にすぎなかった会員数もその後増加し，現在，北は北海道から南は沖縄まで，500 余名を越えるメンバーを擁するにいたりました．法人賛助会員も 25社を数えます．支部は，東京以外に，関西，横浜，名古屋，九州，広島，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます． SEA は，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

代表幹事： 深瀬弘恭

常任幹事： 荒木啓二郎 高橋光裕 田中一夫 玉井哲雄 中野秀男

幹事： 伊藤昌夫 大場充 落水浩一郎 窪田芳夫 熊谷章 小林修 桜井麻里
酒匂寛 塩谷和範 篠崎直二郎 新谷勝利 新森昭宏 杉田義明 武田淳男
中来田秀樹 野中哲 野村行憲 野呂昌満 端山毅 平尾一浩
藤野誠治 松原友夫 山崎利治 和田喜久男

事務局長： 岸田孝一

会計監事： 橋本勝 吉村成弘

分科会世話人 環境分科会(SIGENV)：塩谷和範 田中慎一郎 渡邊雄一
教育分科会(SIGEDU)：君島浩 篠崎直二郎 杉田義明 中園順三
ネットワーク分科会(SIGNET)：人見庸 松本理恵
プロセス分科会 (SEA-SPIN))：伊藤昌夫 塩谷和範 高橋光裕 田中一夫 端山毅 藤野誠治
フォーマルメソッド分科会(SIGFM)：荒木啓二郎 伊藤昌夫 熊谷章 佐原伸 張漢明 山崎利治

支部世話人 関西支部：小林修 中野秀男 横山博司
横浜支部：野中哲 藤野晃延 北條正顕
名古屋支部：筏井美枝子 石川 雅彦 角谷裕司 野呂昌満
九州支部：杉田義明 武田淳男 平尾一浩
広島支部：大場充 佐藤康臣 谷純一郎
東北支部：布川博士 野村行憲 和田勇

賛助会員会社：ジェーエムエーシステムズ ＳＲＡ :ＰＦＵ テプコシステムズ 構造計画研究所 富士通
オムロンソフトウェア キヤノン 富士通エフ・アイ・ピー 新日鉄ソリューションズ
ダイキン工業 オムロン 富士電機 ブラザー工業 オリンパス光学工業
リコー アルテミスインターナショナル NTTデータ ヤマハ 福井コンピュータ
日本ネスト オープンテクノロジーズ ＳＲＡ西日本 日本総合研究所 ハイマックス
（以上25社）

ＳＥＡＭＡＩＬ Ｖｏｌ．１３，Ｎｏ．６－７ ２００２年７月３１日発行 編集人 岸田 孝一
発行人 ソフトウェア技術者協会（ＳＥＡ）
〒１６０-０００４ 東京都新宿区四谷３－１２ 丸正ビル５Ｆ
T: 03-3356-1077 F: 03-3356-1072 sea@sea.or.jp
印刷所 有限会社 錦正社 〒１３０-００１３ 東京都墨田区錦糸町４－３－１４
定価 １、０００円 （禁無断転載）

SEA 特別ワークショップ「UML を徹底的に分析評価する」報告書の特集

熊谷章（PFU）

去る 2002 年 2 月 7 日と 8 日に開催された UML に関するワークショップの内容を報告する。発表者のご好意により、発表に使用された資料も併せてこのレポートの一部として公開できることになった。改めて、発表者の皆さんのご好意とご協力にこの紙面で感謝の意を表したい。

さて、この報告書の内容を次のように編集することにした。

ワークショップの参加者募集とプログラム
発表者の発表資料
ワークショップの報告

最近、UML はソフトウェアが使用されているあらゆるところで、従来考えられなかったような使い方として使い始められ、新たな展開を見せ始めている。そこには、新しい問題と挑戦が待っているに違いないが、ほとんどの人々がそれを良しとして、ポジティブに受け入れ、日常の仕事に活かそうと努力しているようである。この冊子が、いま社会で起きている様々な現象に対して役に立てば良いという願いを込めて発行するものである。

SEA特別ワークショップ　「UMLを徹底的に分析評価する」

コーディネータ　熊谷章（PFU）

いまや UML はオブジェクト指向とコンポーネント思考の共通言語として定着した観があります。コンピュータ科学、ソフトウェアエンジニアリング、ドメインエンジニアリング、アプリケーション開発などソフトウェアが使用されているあらゆる分野でホットスポットとして扱われています。

一方、全世界的なレベルで普及しているUMLに対してクールな批評的態度も見受けられます。代表的なものは、プロフェッショナルを自認するオブジェクト指向プログラミング言語を駆使するプログラマです。彼らは、UML抜きでここ二十数年ほど多くのシステムを作り上げて来ました。UMLが彼らの活動に対して貢献できることはほとんどない、というのが彼らの主張です。

しかし、新しい動きもあります。従来、システムLSIや組込みシステムでは、旧来のソフトウェアはあまり注目されませんでしたが、開発対象システムの高機能化によって、ソフトウェアがシステム開発のキーポイントになり始めました。このため、ソフトウェア開発の新しい方法とツールが必要だという認識からUMLを一つの解として採用しています。

当然、従来のシステム開発方法の表現技術としてUMLが脚光を浴びていることも事実です。１９７０年代に多くのメインフレームメーカが提供したシステム開発方法論とツール群はいまやその使命を終え、UMLを含んだ新しい方法論に置き換えられつつあります。

このUML普及に拍車をかけているのが分散コンピューティングとXML技術です。インターネットを介した分散コンピューティングは、Web Information System や Web Engineeringとして、CORBAの普及と共に社会システムの基盤技術になりました。また、SGMLから始まった文書をコンピュータライズする動きは、HTMLとXMLに至っています。これらの技術は、ITと呼ばれ社会のIT革命のキー技術とみなされ、ナレッジマネージメントの必要条件として評価されています。

今回の特別ワークショップは、上述の背景の下に「UML を徹底的に分析評価」し、UML技術の本質とその効用を実践レベルで明らかにしたいと考えています。次にこのワークショップのプログラムを紹介します。

狙い：UML（Unified Modeling Language）が何者で、どこから来て、どこへ行こうとしているかを下記の観点から理解する。

コンピュータサイエンス

プログラミング言語、モデル、フォーマルメソッド
ソフトウェアエンジニアリング
モデル、開発方法論、ツール、プロセス

効果：UML を理解し、仕事に有効に使用できるようになること。
UML とはなにか
種々の方法論と UML の関係
どのような分野にどのように使えば有効か
システム開発のライフサイクル上での UML が果たす役割
UML の過大評価と過小評価
UML に欠けているもの（機能、表現、考え方など）

プログラム：
2002 年 2 月 7 日
9：30－9：40　プログラム紹介
9：40－11：10　UML の解説（羽生田栄一：豆蔵）
UML 技術のキモ、ミソを分かり易く解説：次の論点を含む
－UML を言語と定義した理由
－UML の abstract syntax, well-formedness rule, semantics の説明
－UML の今後の計画
11：10－12：10　モデルと UML（玉井哲雄：東京大学）
コンピュータサイエンスとソフトウェアエンジニアリングの視点から
モデルと UML の関係を論じる
12：10－13：10　昼休み
13：10－13：40　UML の大学教育（中谷多哉子：S-lagoon）
大学で UML のなにをどのように教えているか、その狙いと効果
13：40－14：10　UML の実践教育（桜井麻里：TIPS）
実践セミナーでなにをどのように教えているか、その狙いと効果
14：10－14：40　プログラマからみた UML（東田雅宏：PFU）
15：00－17：30　UML の意義と将来性（パネル討論）
発表者全員が参加し UML の意義と将来性に関して議論
17：30－21：00　BOF（徹底討論会）：発表者全員
UML とあなたはどうかかわるべきか
2002 年 2 月 8 日
10：00－12：00　実践紹介（事例研究）
伊藤友里恵（OMG Japan）；日本の普及実態と他との関係

石橋　民男（JFITS）：UML を用いたシステム開発事例
渡辺政彦（eUML）：組込み用にアレンジした eUML の狙い

12：00－13：00　昼休み

13：00－13：30　開発ツール／プロセスと UML（伊藤昌夫：ニルソフト）
「UML とどう関わるべきか」という実践的視点から UML を論じる

13：30－14：00　システム分析設計と UML（酒匂寛：Designner's DEN）
顧客の要望に基づいて実際にシステムを分析設計しているシステム
アナリストの目から UML を分析評価

14：00－16：30　UML を有効に使うために（パネル討論）
今後我々はなにをなすべきかを論じる
パネラーは二日目の発表者全員

開催要領

主催　ソフトウェア技術者協会（SEA）
期日　2002 年 2 月 7 日（木）、8 日（金）
場所　東京労働スクウェア（東京都八丁堀）
参加費 会員　20,000 円、賛助会員　30,000 円、一般　40,000 円
会場にて現金にてお支払い下さい。
定員　70 名
参加申込　必要事項を記入の上下記指定アドレスまで e-mail で申込み下さい。
sea@sea.or.jp
申込〆切　2002 年 1 月 31 日

参加申込：
SEA 特別ワークショップ「UML を徹底的に分析評価する」申込

氏名：
所属：
会員の種別：SEA 会員、賛助会員、一般
参加費：
連絡先電話か Email：

SEAワークショップ
UML解説　その肝と心

2002年2月7日

株式会社　豆蔵
羽生田栄一

本トレーニングテキストの一部または全部を著作権法の定める範囲を超え、
無断で複写、複製、転載、テープ化、ファイル化することを禁じます。

豆蔵 MAMEZOU

目次

▶UMLとは何か　成立の経緯と存在意義
▶UMLダイアグラム　10種の一瞥
▶UML1.4超入門　1.3との差分
▶UMLの段階的理解
▶UMLセマンティクス入門
▶OCLとUMLメタモデル断片の例
▶UMLの今後
▶まとめ

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 2

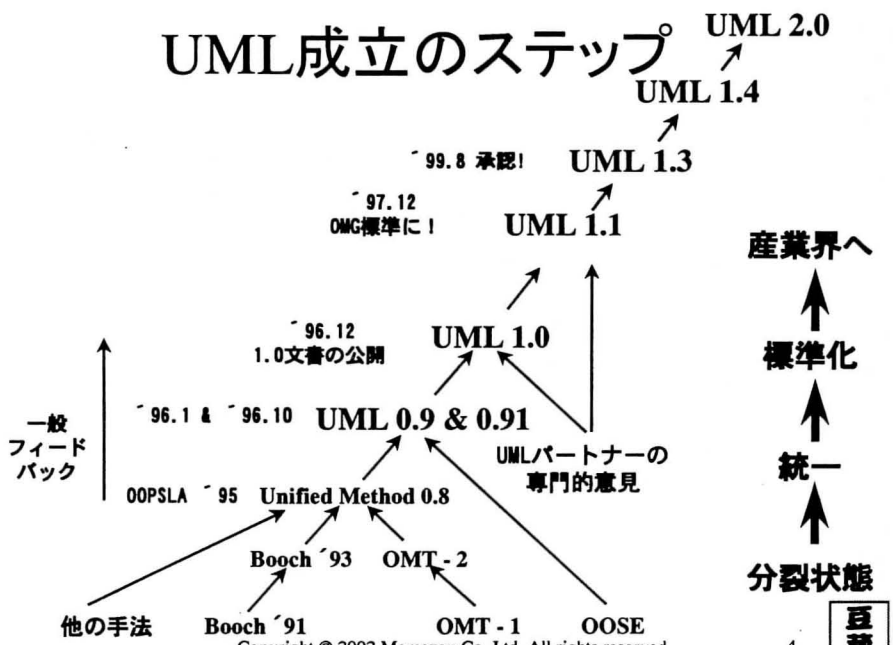
豆蔵 MAMEZOU

目次： UMLとは何か
成立の経緯と存在意義

- ▶UML成立のステップ
- ▶UMLという設計言語の設計課題
- ▶UMLの源流
- ▶UMLの目標
- ▶UMLの特徴と適用範囲
- ▶なぜUMLはデファクトになれた？

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 3

UML成立のステップ



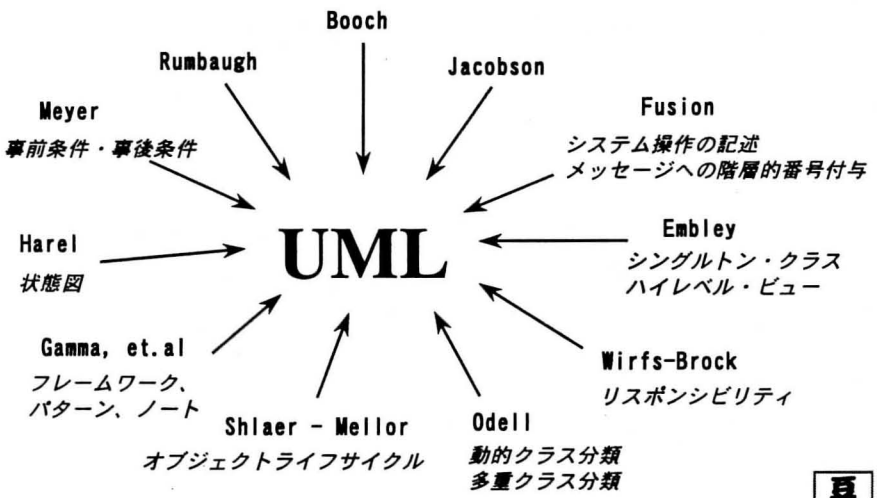
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 4

UMLという設計言語の設計課題

- Booch, OMT, OOSE(Objectory)の統一
- モデリングの重要概念の整理:メタモデル
- 現在/近未来のソフトウェア開発に貢献する
  - 大規模システム開発
  - Java, C++, Smalltalk, Ada, VB, PB, ...
  - 分散と並列、マルチスレッド
  - デザインパターン
  - コンポーネントウェア
  - ビジネスモデリング
- 以上を満たしながらできるだけ単純に

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 5

UMLの源流



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 6

UMLの目標

- 方法論の群雄割拠時代の収束
  - Booch法、OMT，Objectoryの発展形
  - 従来のダイアグラム・記法の集大成
- 記法も仕様もオープン
  - OMG,業界標準をめざす
  - ステレオタイプで拡張・カスタマイズ可能
- 実際的な開発での適用
  - 要求・分析・設計・テストと一貫表現
  - 大規模開発への対応
  - 標準的なOO開発プロセスのための記法

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 7 豆蔵 MAMEZOU

開発プロセスから見たUMLの特徴

- 第2世代方法論をベースに再設計
  - メタモデルによるモデル概念の見直しと統一
  - 既存のほとんどの方法論からのいいとこ取り
- 大規模システム対応
  - パッケージとその階層化，インターフェース概念
- 構造と機能/振舞いの相補的な視点
  - オブジェクト(構造と振る舞い),コンポーネント
  - ユースケース(外部機能)，パターン
- 全開発プロセスをフルにサポートできる表現力
  - アーキテクチャの表現(4＋1ビュー)

豆蔵 MAMEZOU

何の「統一」か？

- 既存の様々な方法論
  ＝＝＞ モデリング記法 の統一
- 様々な問題領域での
  仕様記述、システム記述のスタイルの統一
- ソフトウェア開発の様々な局面での成果物やモデルの構成要素の統一

- しかし、
  - 開発プロセスそのものの統一ではない！
    ＝＝＞ **Unified Process** の試み

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 9

UMLの適用範囲

- ビジネスモデリング
- 要求定義
- ドメイン分析
- アーキテクチャ定義
- 設計
- 実装の表現(リバースモデル)
- *テスト対象のモデル化*
- *プロジェクト管理対象のモデル化*
- *リリース管理対象のモデル化*

. . . その他何でもシステム思考できれば

UMLの適用範囲外

- ビジネス:経営戦略やビジョン、計画や意図
- 要求:ユースケース内容自体の定義、要件管理、リスク、・・・
  - アクティビティ図で一部記述できるが...
- アーキテクチャ:複数の図に跨る制約、非機能要求の一部、・・・
- 設計:設計ポリシー、時間制約の一部、ユーザビリティ、・・・
  - アーキテクチャパターン、デザインパターン、状態マシンで表現困難なもの
- 実装:プログラムコードそのものや設定
- テスト:テスト方針、テストケース、テストスクリプト
- UMLモデル自体あるいは関連成果物の版管理・構成管理
- その他、プロジェクト運営用の各種成果物やチャート
- アドリブで書くホワイトボード上のさまざまなポンチ絵
- その他、迅速な小規模コミュニケーションでの融通無碍な感覚

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 11

なぜUMLはデファクトになれた？

- 方法論戦争後のUML登場のタイミング
- 実績ベースで有用な記法のイイとこ取り
- お目付け役としての3アミーゴの存在
- サポータとしてのRational Softwareの存在
- 厳密さを犠牲にしたセマンティクス規定
- 強大なコミュニティの形成に成功
- とりあえずの導入は容易...

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 12

目次: UMLダイアグラム 10種の一瞥

- ダイアグラム
- ダイアグラム間の関係

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 13

UMLのダイアグラム

定義内容	視点	ダイアグラム名
要求図	モデル化対象に対する要求を表す	ユースケース図
構造図	モデル化対象の静的な構造を表す	クラス図
		オブジェクト図
		パッケージ図(クラス図の一種)
振舞い図	モデル化対象の動的な振舞いを表す	シーケンス図(相互作用図)
		コラボレーション図(相互作用図)
		アクティビティ図
		ステートチャート図
実装図	モデル化対象の物理的な実装を表す	コンポーネント図
		配置図

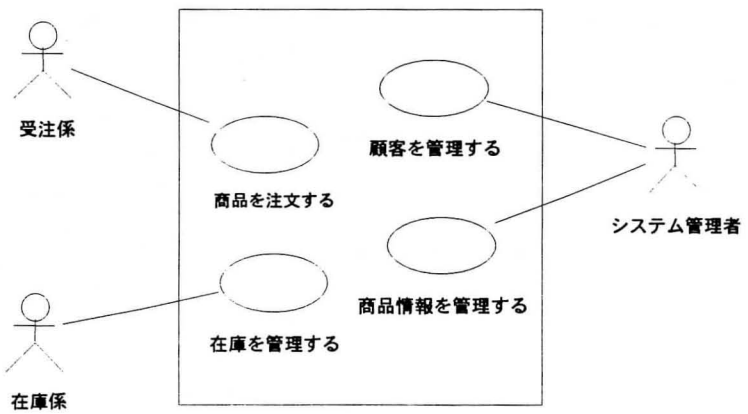
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 14

UMLのダイアグラム

- 要求図
  - ユースケース図
    - ユーザーから見たシステムの機能を表現する

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 15

ユースケース図



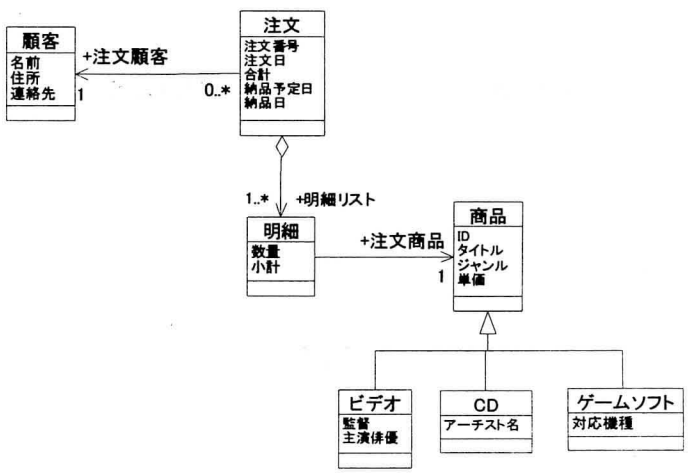
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 16

UMLのダイアグラム

- ▶構造図
  - ▶クラス図
    - ▶クラスおよびクラス間の静的な構造を表現する
  - ▶オブジェクト図
    - ▶クラス図で表現された静的な構造の、ある時点におけるスナップショット(インスタンス間の関係)を表現する
  - ▶パッケージ図
    - ▶クラス図の特殊なもの

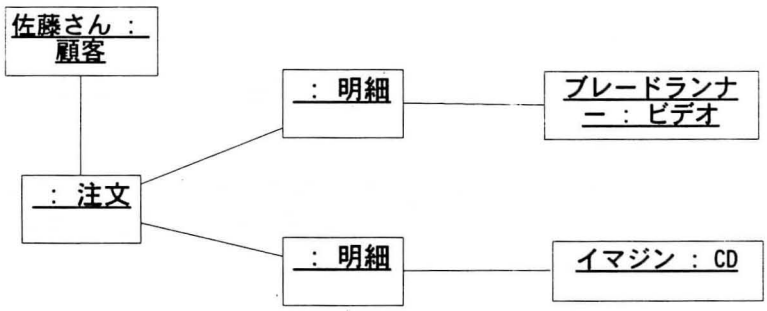
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 17

クラス図



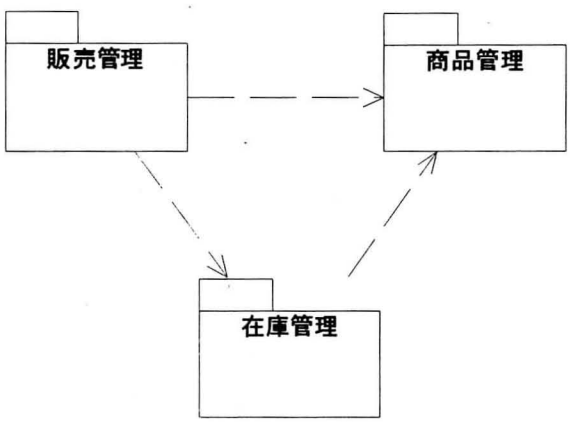
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 18

オブジェクト図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 19

パッケージ図



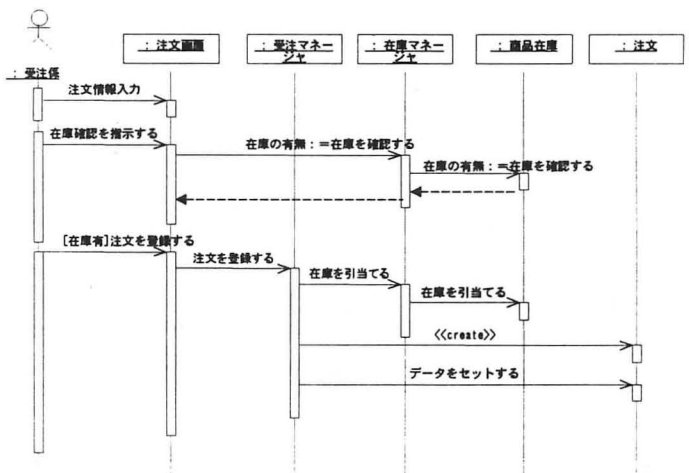
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 20

UMLのダイアグラム

- 振る舞い図
  - 相互作用図：シーケンス図，コラボレーション図
    - 複数のオブジェクトによる協調動作を表現する
  - アクティビティ図
    - 業務や処理のフローを表現する
  - ステートチャート図
    - オブジェクトのライフサイクルにおける状態遷移を表現する

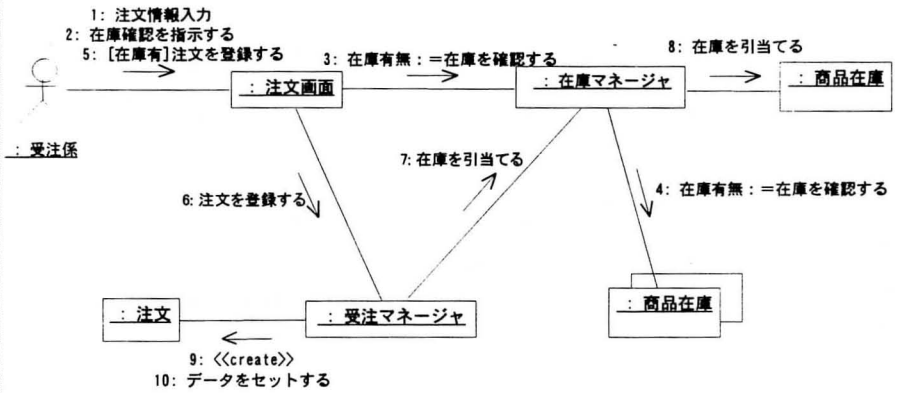
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 21

シーケンス図



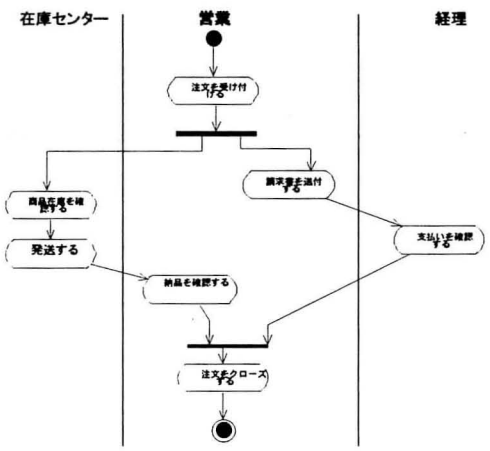
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 22

コラボレーション図



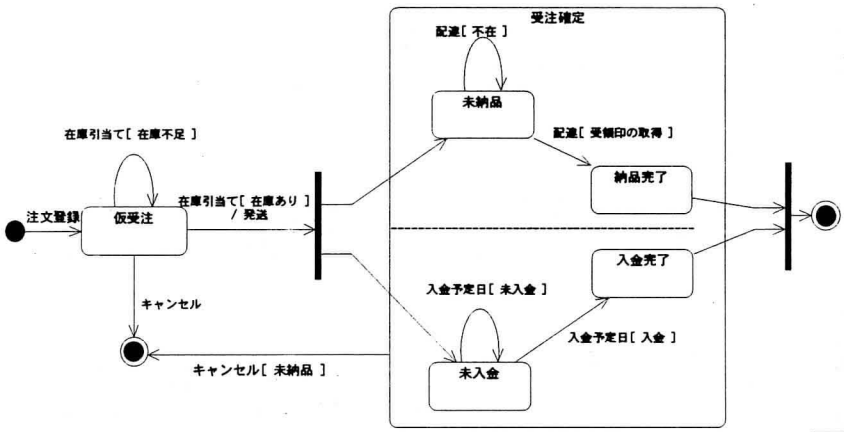
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 23

アクティビティ図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 24

ステートチャート図



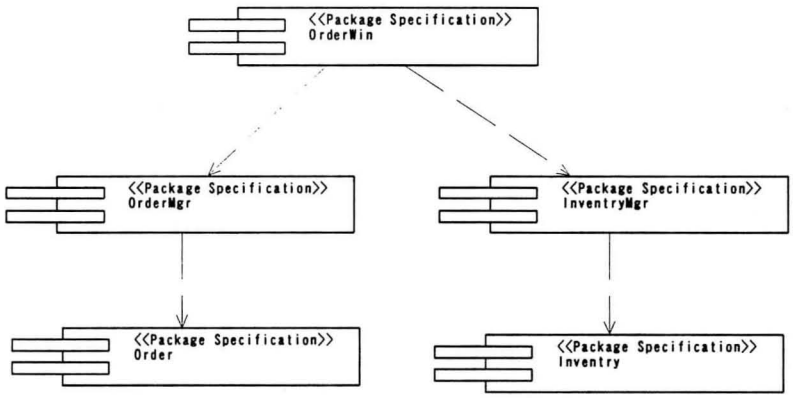
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 25

UMLのダイアグラム

- ▶実装図
  - ▶コンポーネント図
    - ▶ソフトウエアの物理構成を表現する
    - ▶ソフトウェアモジュール間の依存関係を表現する
  - ▶配置図
    - ▶コンポーネントやオブジェクトのノード上への配置構成やノード間の接続仕様等を表現する

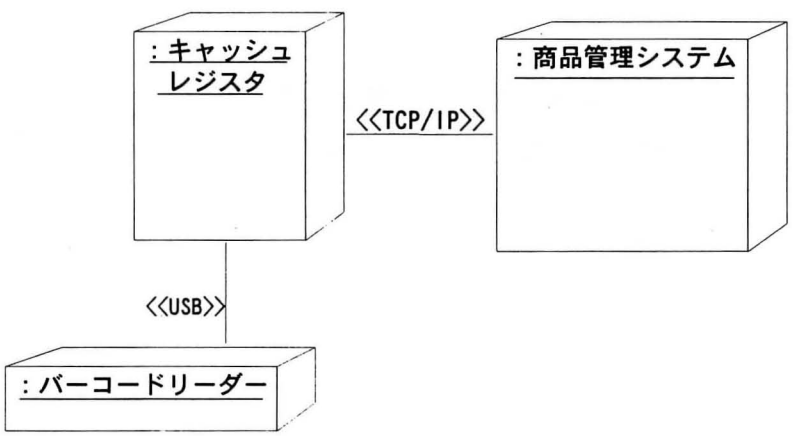
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 26

コンポーネント図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 27

配置図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 28

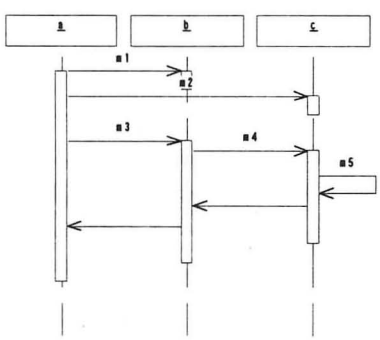
ダイアグラムがわかれば十分か？

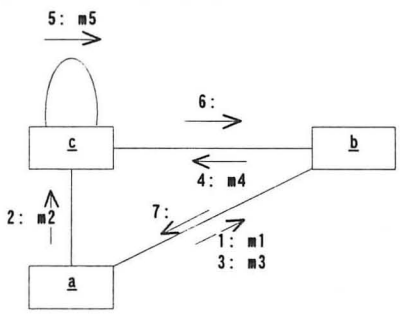
- ▶各ダイアグラムがわかるとはどういうことか？
  - ▶アイコンの記法以上の意味
  - ▶人との解釈の相違の解決
  - ▶自分なりに拡張解釈していい部分、悪い部分
- ▶複数のダイアグラムの間の関係は？

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 29

シーケンス図とコラボレーション図

- ▶インタラクション図
  - ▶シーケンス図
  - ▶コラボレーション図

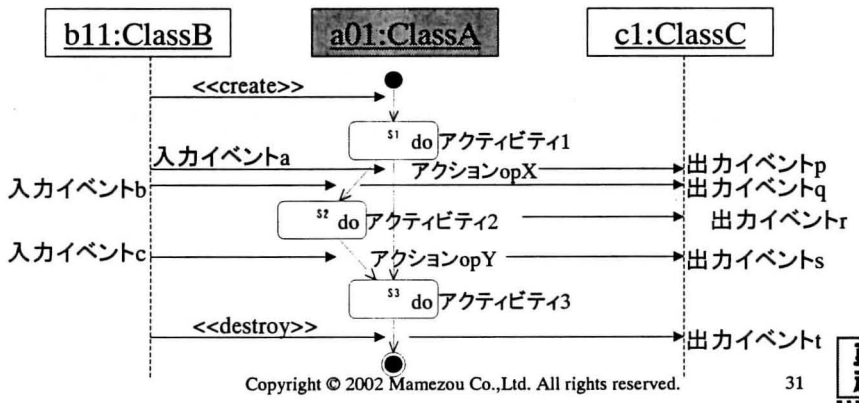




Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 30

シーケンス図とステートチャート図の関係

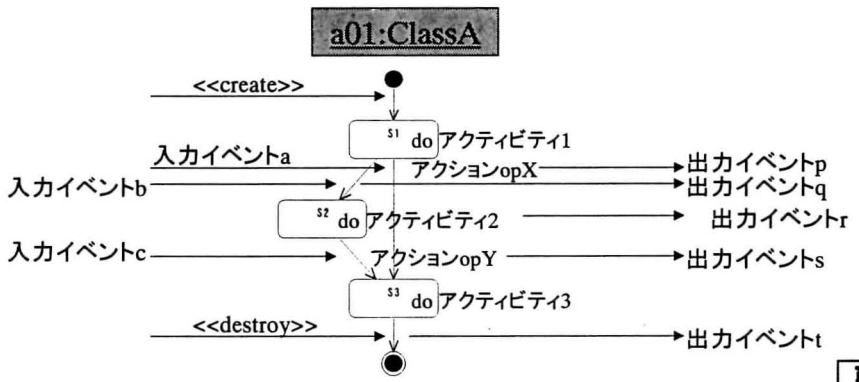
- オブジェクトのライフサイクル表現
  - ステートチャート図
  - イベントによる状態遷移とアクション



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 31

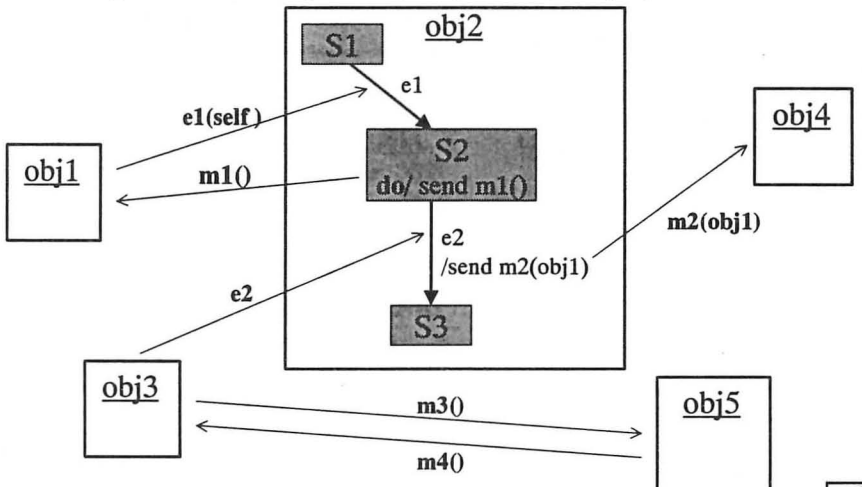
ステートチャート図とコンポーネント仕様

- オブジェクトの振る舞いの「プロトコル」
  - 入力イベントとアクションの出力イベント



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 32

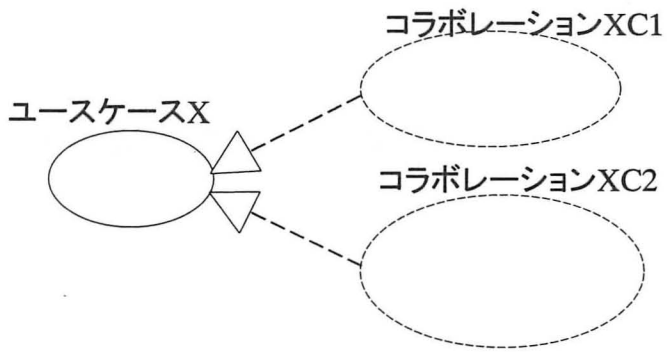
オブジェクト間の相互作用とオブジェクトライフサイクル



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 33

ユースケースとコラボレーションの関係

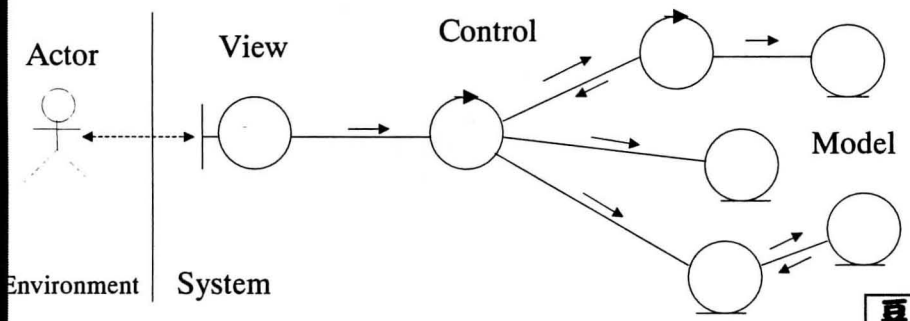
- ユースケースと対応するコラボレーションの間のrealization実現関係



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 34

コラボレーションとMVCパターン

- ユースケースの実現としてのコラボレーション：MVCアーキテクチャとロバストネス分析
  - View:アクターごとにバウンダリobj
  - Control:コラボレーションに1個コントローラobj
  - Model:ドメインの知識を表現するエンティティobj



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 35

UMLの段階的理解

- モデル要素
- 関係
- 共通メカニズム
- ダイアグラム
- アーキテクチャに対する複数のビュー

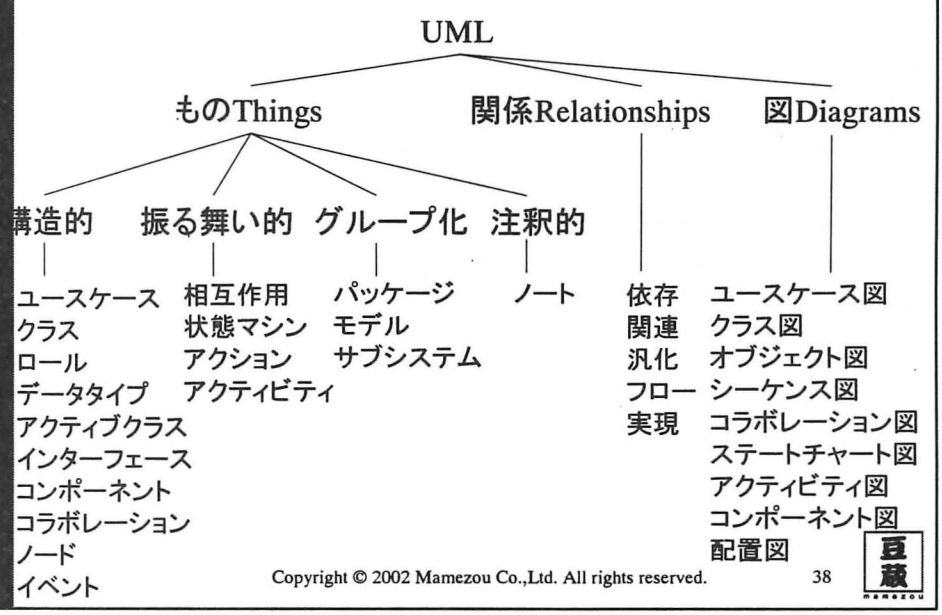
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 36

1. モデル要素

- クラス/オブジェクト
- インターフェース
- コラボレーション
- ユースケース
- プロセス/スレッド
- コンポーネント
- 相互作用
- 状態マシン
- ノード
- パッケージ
- ノート

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 37

UMLの基礎語彙



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 38

構造モデル要素

- クラス
  - 共通の構造と挙動を持つオブジェクトの集合に対する定義
- インターフェース
  - オブジェクトの外部公開部(公開操作集合)
- コラボレーション
  - オブジェクトの集団の協調
- ユースケース
  - 名前の付いたシステムの振る舞い。コラボレーションによって実現される

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 39

構造モデル要素

- アクティブクラス
  - プロセス/スレッドを表すアクティブなクラス。他のアクティブなクラスと並行動作可能。
- コンポーネント
  - 再利用可能な部品。論理的な部品と物理的な部品の両方を表現。
- ノード
  - ソフトウェア(コンポーネント)が配置され実行されるハードウェアデバイスの抽象化

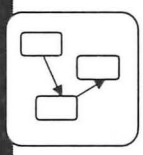
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 40

振る舞いモデル要素

- ▶相互作用

msg ▶オブジェクト間のメッセージ交信による情報の伝搬や状態変化のトリガー

- ▶状態マシン
  - ▶オブジェクトのライフサイクルとそれにともなうアクション/アクティビティ



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 41

グループ化と注釈・モデル要素

- ▶パッケージ
  - ▶モデル要素のコンテナ（容れ物）
- ▶ノート
  - ▶コメント、説明、注釈等のテキスト記述単位

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 42

関係

- ▶関連(集約は関連の特殊形)

r1 ＿R＿ r2 2つのインスタンス間の意味的な接続関係

- ▶汎化関係

——▷ モデル要素およびそれと置換可能なサブモデル要素との間の関係

- ▶依存関係

----------> 別のモデル要素を使用するという関係

- ▶実現関係

----------▷ 要求・仕様・定義と実現手段・方式の対応関係

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 43

UML意味規則

- ▶名前:その要素の呼び名
- ▶スコープ:名前を与える際のコンテキスト
- ▶可視性:その名前の他からの見え方、使われ方
- ▶統合:他の要素との関係づけ
- ▶実行:動的モデルの実行に伴う意味

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 44

UML共通のメカニズム

- 共通して使われる2項分類
  - クラス/オブジェクト
    - タイプ/インスタンスの区別の導入
  - インターフェース/実装
- 拡張メカニズム
  - ステレオタイプ <<キーワード>>
  - タグ付き値（プロパティ名＝値）
  - 制約 {その対象が満足すべき条件，命題}

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 45

UMLダイアグラムの構成

- 構造図
  - クラス図 (パッケージ図はクラス図の変種)
  - オブジェクト図
- 振る舞い図
  - ユースケース図
  - シーケンス図
  - コラボレーション図
  - ステートチャート図
  - アクティビティ図
- 実装図
  - コンポーネント図
  - 配置図

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 46

構造図

- クラス図
  - モデルの組み立て部品の集合：クラスと関係
- オブジェクト図
  - システムのある時点でのスナップショット
- パッケージ図
  - モデルの構成の管理
  - クラス図のバリエーションとみなせる

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 47

振る舞い図

- ユースケース図
  - システムの外部機能（ファンクショナリティ）
- シーケンス図
  - 相互作用するオブジェクトの時間順序系列
- コラボレーション図
  - オブジェクトの集団の相互作用の直接表現
- ステートチャート図
  - 状態遷移図。オブジェクトのライフヒストリー
- アクティビティ図
  - 1つのインタラクション全体における手続きの制御フロー

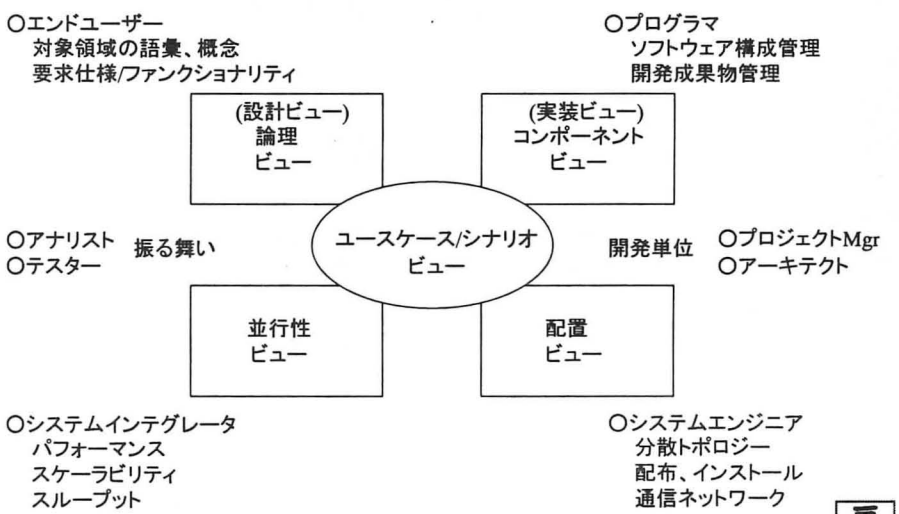
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 48

実装図

- ▶コンポーネント図
  - ▶ソフトウェア・ユニット間の依存関係
- ▶配置図
  - ▶コンポーネントやオブジェクトの、計算ノード(ネットワーク上のCPUやプロセス、デバイス等の抽象化)上をまたがる分散配置や相互作用

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 49

アーキテクチャと4+1ビュー



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 50

目次:UMLセマンティクス入門

- ▶UMLの標準拡張機構
- ▶OCL入門
- ▶4層メタモデルアーキテクチャ
- ▶UMLメタモデルの構造
- ▶UML抽象構文とその注釈の例
- ▶UML適格性規則と追加操作の例
- ▶UML意味論記述の例

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 51

UMLの標準拡張機構:ステレオタイプ

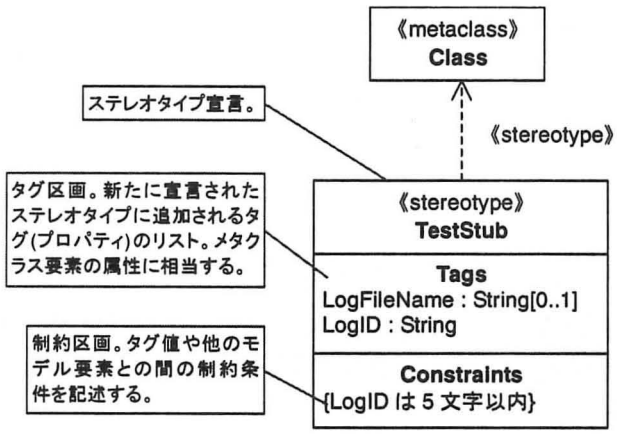
ステレオタイプ定義:

- ▶UMLのコアモデル要素をベースに それを特殊化することで定義する
- ▶ 以下の4つの指定を行う
  - ▶ベースクラス(メタクラスないしは他のステレオタイプ)
    - ▶特殊化するべき元のモデル要素の指定
  - ▶制約
    - ▶What is special about this stereotype?
  - ▶タグ (プロパティ)
    - ▶追加すべき属性をタグ付き値として指定する
  - ▶アイコン
    - ▶もし見た目も元のベースクラスとは区別したければ。。。

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 52

UMLによるステレオタイプ定義

▶UML1.4で導入された記法



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 53

UML制約のステレオタイプ

▶UML制約の3種類の既定ステレオタイプ:

- ▶《 invariant 》: クラスに対し
  - ▶そのクラスのすべてのインスタンスに対し常に満たされていなければならない制約
- ▶《 precondition 》: 操作に対し
  - ▶その操作の実行前に真でなければならない制約
- ▶《postcondition 》: 操作に対し
  - ▶その操作の実行後に真でなければならない制約

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 54

OCLとは何か

- オブジェクト制約言語OCL
  - Object Constraint Language
- クラス図などのビジュアルモデルは，正確で曖昧性のない仕様には不十分
- モデル内のオブジェクトについて付加的な制約を記述する必要
- OCL:読み書きの容易な形式言語
  - 自然言語より厳密で、1階述語論理等の数学的な記法より理解しやすい

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 55

OCLの特徴

- 純粋な式言語、OCL式は副作用なし
  - モデル内のいかなるものも変化させない
- OCLは，プログラム言語ではない
  - ロジックや制御の流れは書けない
- 各OCL式は型をもつ
- OCLの型はUMLにおけるClassifierの一種

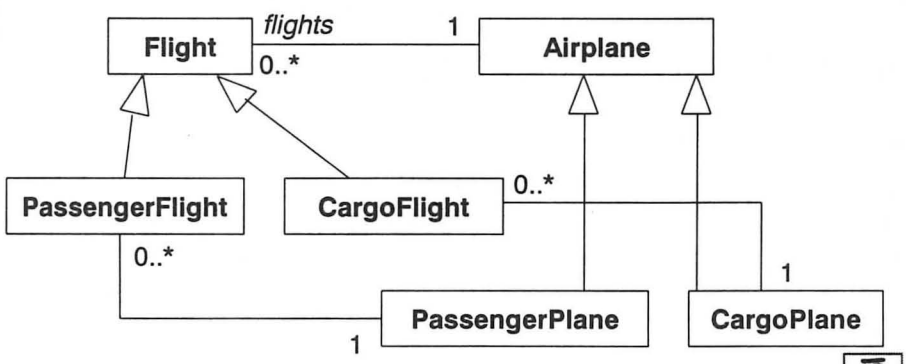
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 56

OCLとモデリング

- *クラスモデルにおける，クラスおよび型の不変制約を規定*
- *ステレオタイプに対し型の不変制約を規定*
- *操作及びメソッドに関する事前及び事後条件を示す*
- *ガードを示す*
- *ナビゲート用言語として使用する*
- *操作に関する制約を示す*
  - *operation(a : Type1, b : Type2) = expression*

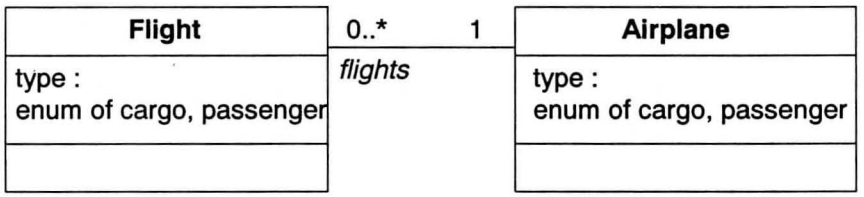
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 57

OCL説明用クラス図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 58

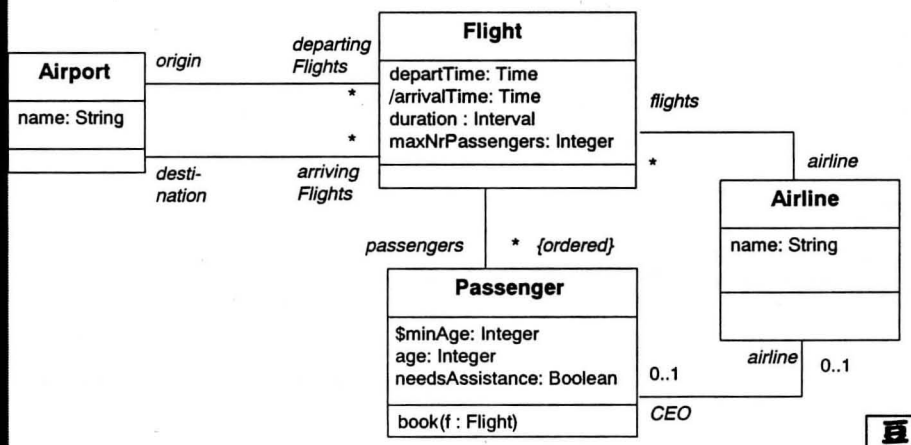
OCL適用クラス図例



{context Flight
inv: type = #cargo implies airplane.type = #cargo
inv: type = #passenger implies airplane.type = #passenger}

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 59

OCL説明用クラス図



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 60

OCL式と context, inv:, self

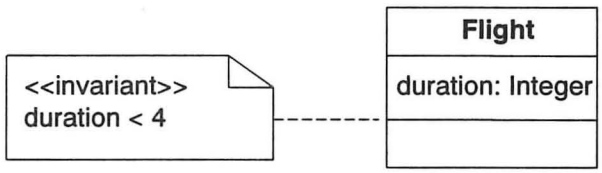
▶OCL式1(selfあり):

context Flight inv: self.duration < 4

▶OCL式2 (selfなし) :

context Flight inv: duration < 4

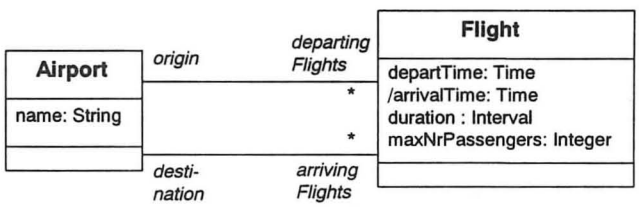
▶クラス図では:



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 61

(.)ドット記法による誘導

```
context Flight
  inv: origin <> destination
  inv: origin.name = 'NRT'
context Flight
  inv: airline.name = 'JAL'
```



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 62

OCLにおけるコレクション操作

- コレクションに対する操作は(.)ではなく(->)で!
  - collection->operation

    collection->collect (e : T | expr)

    collection->collect (e | expr)

    collection->collect (expr)
- ただし省略記法では(.) が使える:

  collection.expr
- 例:

  context Airport inv:

    self.arrivingFlights -> collect(airLine) ->notEmpty

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 63

基本タイプと値

- *基本タイプと値*
  - *Boolean, Integer, Real, String*
  - *true, false, 1,2, 1.5, 3.14, 'To be'*
- *enumタイプ*
  - *enum{ val1, val2, val3 }*

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 64

基本タイプとその操作

タイプ	操作
Integer	*, +, -, /, abs
Real	*, +, -, /, floor
Boolean	and, or, xor, not, implies, if-then-else
String	ToUpper, concat

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 65

オブジェクトとプロパティ

- プロパティ
  - 属性、関連ロール、問合せ操作/メソッド
- 属性
  - Person self.age
  - Person self.age >= 0
- 操作 aPerson:Person, aDate:Date
  - aPerson.income(aDate)
  - Person::income(d:Date):Integer
  - Company self.stockPrice()

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 66

関連と多重度

- 関連ロールの多重度に依存
  - Company
    self.manager -- Person
    self.employee -- Set(Person)
- コレクションのプロパティ
  - Person
    self.employer->size -- # of companies he works for
  - Person
    self.emplyer->isEmpty -- no job now?

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 67

関連とナビゲーション

- 多重度0..1をまたぐナビゲーション
  - Company
    self.manager->size -- aSet.size=1
    -- 存否の確認目的で敢えて->を使用
- 関連クラスへのナビゲーション
  - Person self.job
- 関連クラスからのナビゲーション
  - Job -- 関連クラス
    self.employer
    self.employee

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 68

関連とナビゲーション2

- 限定子付き関連
  - Bank
    self.customer -- Set(Person)
    self.customer[17463] -- aPerson
- パッケージとパス名
  - Packagename::Typename
- スーパークラスへのアクセス
  - SubClassB
    self.SuperClassA::op1()
    self.op1()

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 69

全オブジェクト共通の操作

- 全オブジェクト共通の操作
  - oclType: OclType
  - oclIsTypeOf(t : OclType): boolean
  - oclIsKindOf(t : OclTYpe): boolean
- タイプ自身への操作 allInstances
  - Person.allInstances : Set(Person)
  - Person.allInstances->forAll(p1, p2 |
    p1 <> p2 **implies** p1.name <> p2.name)

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 70

コレクション

- Collection: Set,Sequence,Bag
  - Set{ 1, 2, 4, 8 }
  - Set{ 'apple', 'orage', 'lemon' }
  - Sequence{ 1, 2, 4, 8, 4, 2, 1 }
  - Sequence{ 1..10 }
  - Bag{ 1, 1, 2, 3, 3, 3, 1 }
- 自動的にフラット化され以下は同値!
  - Set{ Set{1, 2}, Set{3, 4}, Set{5,6} }
  - Set{ 1, 2, 3, 4, 5, 6 }

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 71

タイプ間の整合性conformance

- Set(Bicycle) ==> Set(Transport)
- Set(Bicycle) ==> Collection(Bicycle)
- Set(Bicycle) ==> Collection(Transport)

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 72

コレクションに対する操作

- ***Select, Reject, Collect***
  - *Company*

```
self.employee->select(age > 50)
self.employee->select(p | p.age > 50 )
self.employee->reject( isMarried )
self.employee->collect( birthDate )
self.employee->collect(p | p.birthDate )
```

- *Collectの簡略形*
  - *self.employee.birthDate --Bag(Date)*
  - *self.employee.birthDate->asSet --Set(Date)*

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 73

コレクションと全称、存在

- *forAll操作*
  - *Company*

```
self.employee->forAll( forname = 'Jack' )
self.employee->forAll( e1, e2 |
 e1<>e2 implies e1.forename<>e2.forename )
```

- *exists操作*
  - *Company*

```
self.employee->exists( forname = 'Jack' )
```

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 74

反復子

- *iterate操作*
  - *collection->iterate(elem:Type;acc:Type=<exp>| <exp-with-elem-and-acc>)*
- *reject, select, forAll. Exists, collect, electはすべてiterateで実現できる*
  - *collection->collect(x:T | x.property)*
  - *collection=>iterate(x:T; acc:T2 = Bag{} | acc->including(x.property))*

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 75

定義済みOCL型

- OclType
- OclAny
- OclExpression
- Real, Integer, String, Boolean
- Enumeration
- Collection
- Set, Bag, Sequence

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 76

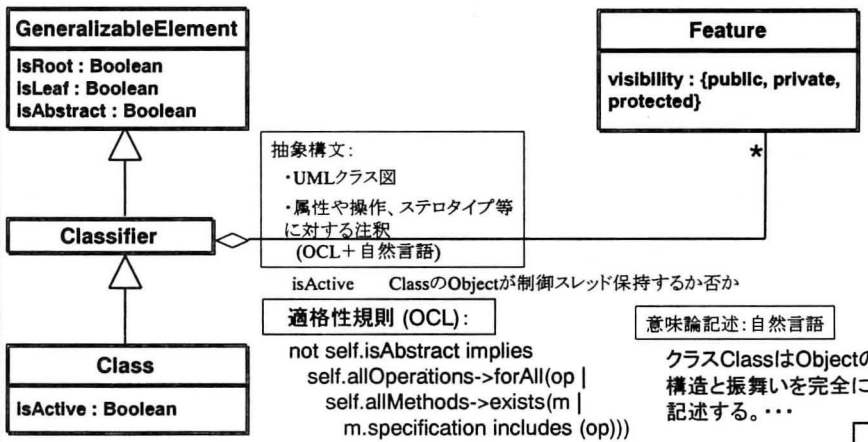
操作の仕様記述とOCL

- 不変制約
  - Person self.age > 0
- 操作の事前条件・事後条件の定義
  - Person::birthdayHappens()
    post: age = age@pre + 1
  - Company::hireEmployee(p: Person)
    pre: not employee->includes(p)
    post: employees->includes(p) and
    stockprice() = stockprice@pre + 10

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 77

UMLメタモデルとは

- UML(＋OCL＋自然言語)によるUMLのモデル



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 78

4層メタモデルアーキテクチャ

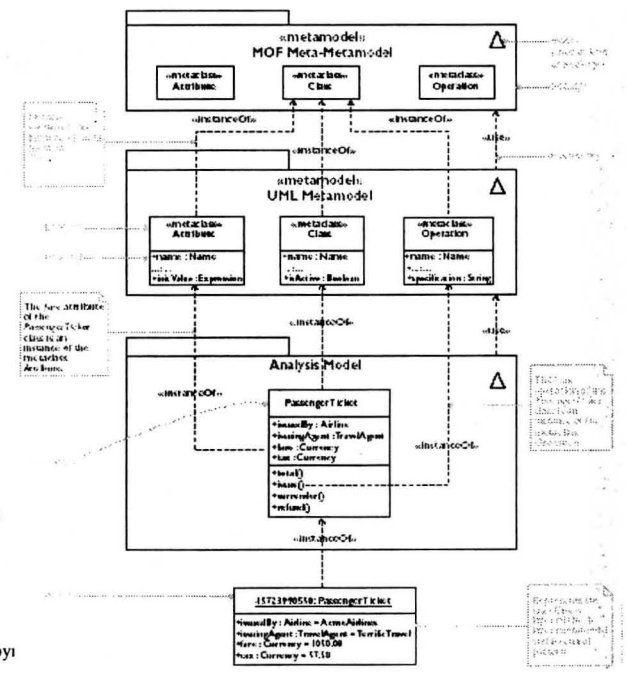
- メタメタモデル層
  - メタモデリングアーキテクチャのインフラを提供。
    メタモデルを記述するための言語の定義。UMLではMOFを利用。
  - 例: MetaClass, MetaAttribute, MetaOperation
- メタモデル層
  - メタメタモデルのインスタンス。モデルを記述するための言語の定義。
  - 例: Class, Attribute, Operation, Component
- モデル層
  - メタモデルのインスタンス。特定ドメインを表現する言語の定義。
  - 例: StockShare, askPrice, sellLimitOrder, StockQuoteServer
- ユーザーオブジェクト層
  - モデルのインスタンス。特定ドメインの具体的表現。
  - 例: <Acme_Software_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 79

4層メタモデルの例

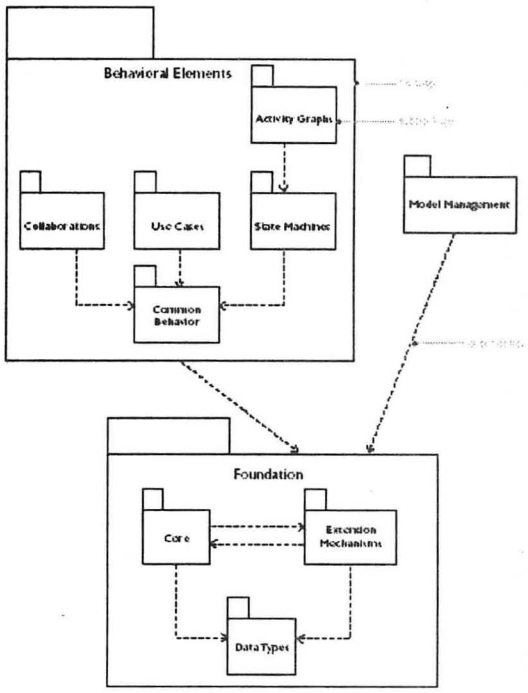
- 《metaclass》MOF MetaMetamodel::Class
- 《metaclass》UML Metamodel::Class
- PassengerTicket
- 45723990550: PassengerTicket

- 出典: UML2001:Standardization Odyssey, by Cris Kobryn



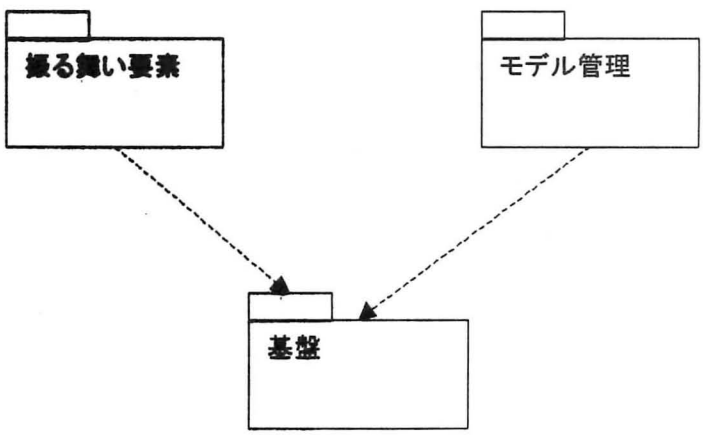
Copyr

UMLメタモデル層の構造



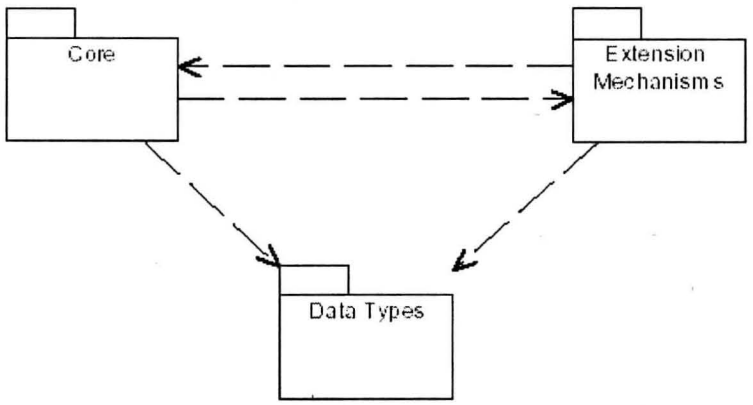
Copyright ©

UMLメタモデルの最上位構造



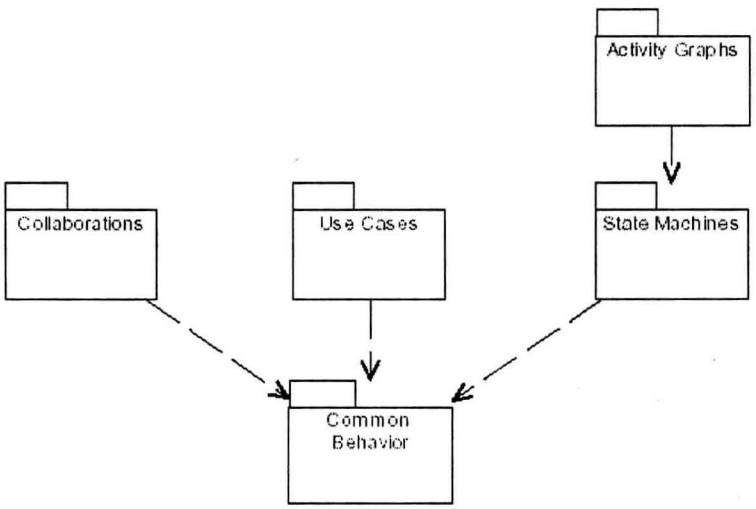
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 82

UMLメタモデル：Foundation 基盤

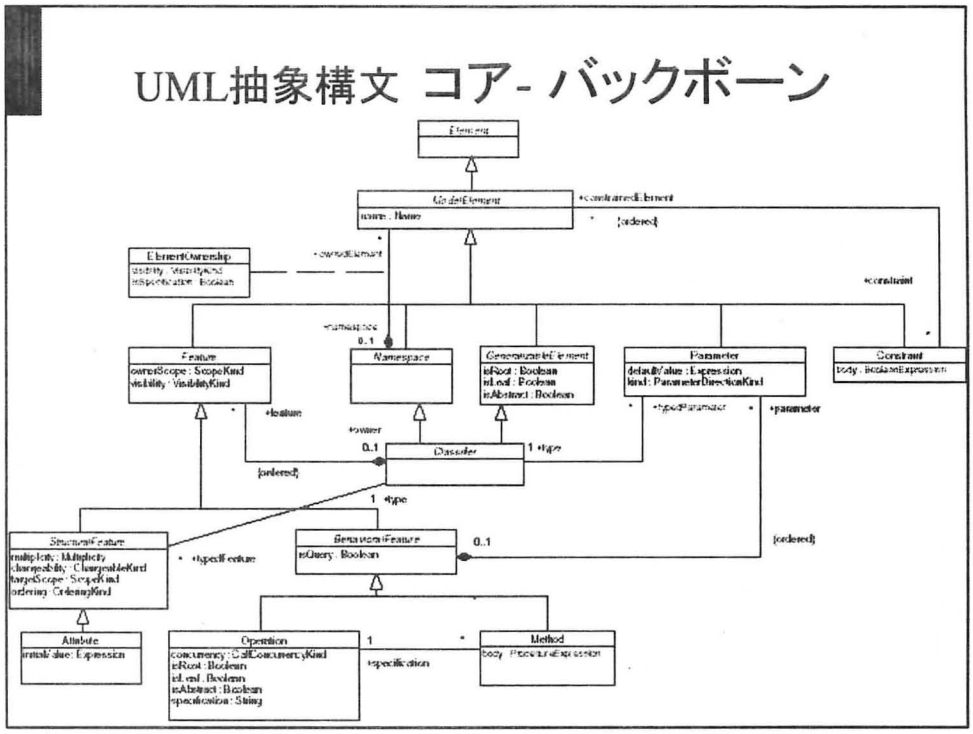


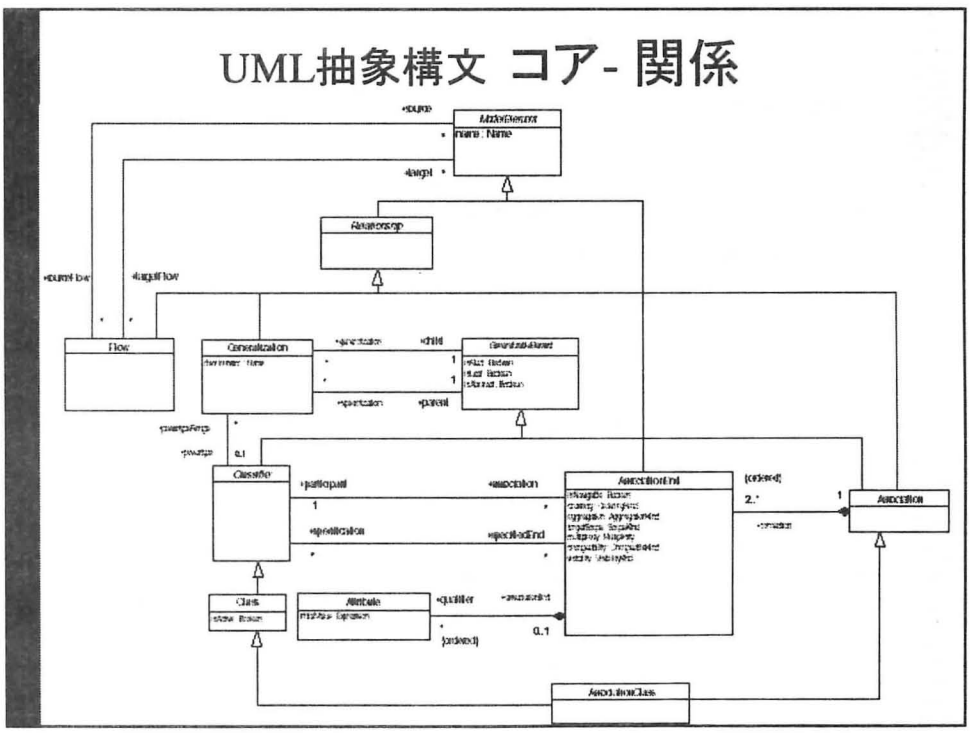
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 83

UMLメタモデル：振舞い要素



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 84


UML抽象構文 コア- バックボーン
Element
ModelElement
name : Name
ElementOwnership
Feature
ownerScope : ScopeKind
visibility : VisibilityKind
Namespace
GeneralizableElement
isRoot : Boolean
isLeaf : Boolean
isAbstract : Boolean
Parameter
defaultValue : Expression
kind : ParameterDirectionKind
Constraint
body : BooleanExpression
Classifier
StructuralFeature
multiplicity : Multiplicity
changeability : ChangeableKind
targetScope : ScopeKind
ordering : OrderingKind
BehavioralFeature
isQuery : Boolean
Attribute
initialValue: Expression
Operation
concurrency : CallConcurrencyKind
isRoot : Boolean
isLeaf : Boolean
isAbstract : Boolean
specification : String
Method
body : ProcedureExpression


UML抽象構文 コア- 関係
ModelElement
name : Name
Namespace
Flow
Generalization
GeneralizableElement
Classifier
Association
AssociationEnd
Class
Attribute
AssociationClass

UML抽象構文 *AssociationClass* 注釈

An association class is an association that is also a class. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not any of the classifiers.

Inherited Features

AssociationClass inherits features as specified in both Class and Association.

In the metamodel, an AssociationClass is a declaration of a semantic relationship between Classifiers, which has a set of features of its own. AssociationClass is a subclass of both Association and Class (i.e., each AssociationClass is both an Association and a Class); therefore, an AssociationClass has both AssociationEnds and Features.

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 87

AssociationClass 適格性規則

- [1] The names of the AssociationEnds and the StructuralFeatures do not overlap.
  - `self.allConnections->forAll( ar |`
  - `self.allFeatures->forAll( f | f.oclIsKindOf(StructuralFeature)` **implies** `ar.name <> f.name ))`
- [2] An AssociationClass cannot be defined between itself and something else.
- `self.allConnections->forAll(ar | ar.participant <> self)`

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 88

AssociationClass 適格性規則

▶ ***Additional operations***

[1] The operation allConnections results in the set of all AssociationEnds of the AssociationClass, including all connections defined by its parent (transitive closure).

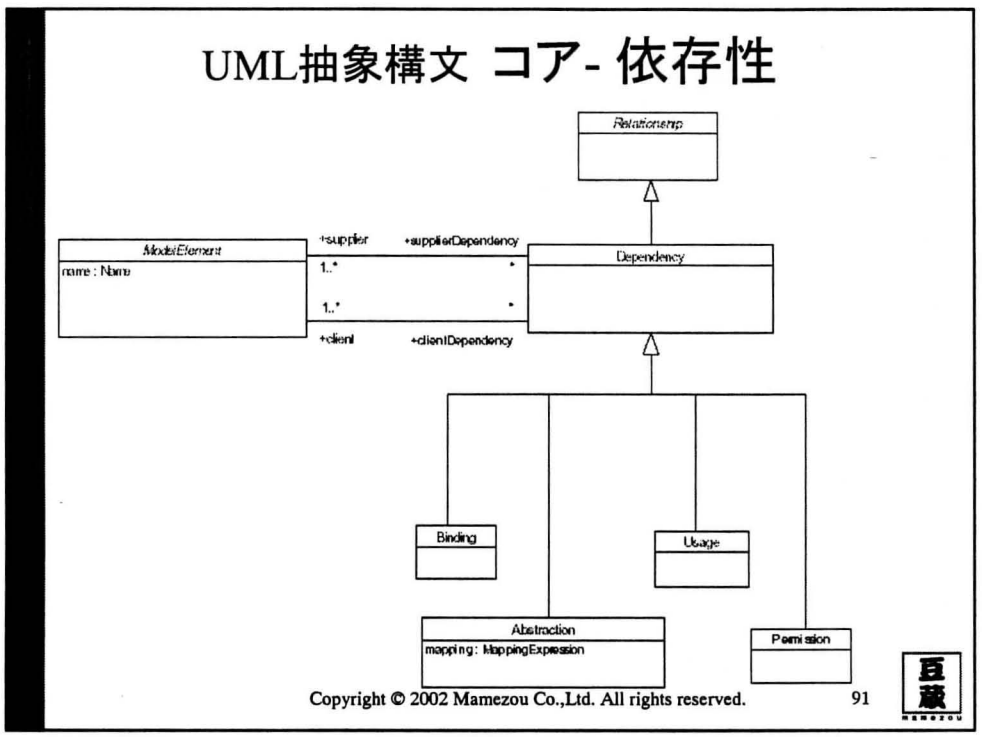
```
allConnections : Set(AssociationEnd);
allConnections = self.connection->union(
  self.parent-> select (s |
      s.oclIsKindOf(Association))->collect
            (a : Association |
            a.allConnections))->asSet
```

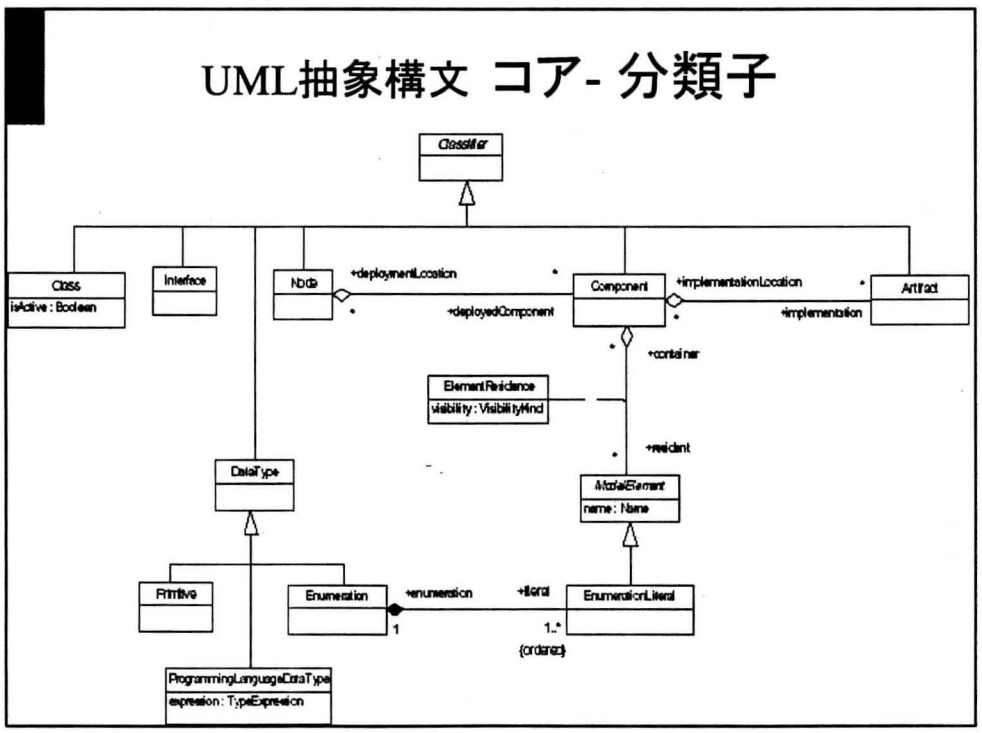
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 89

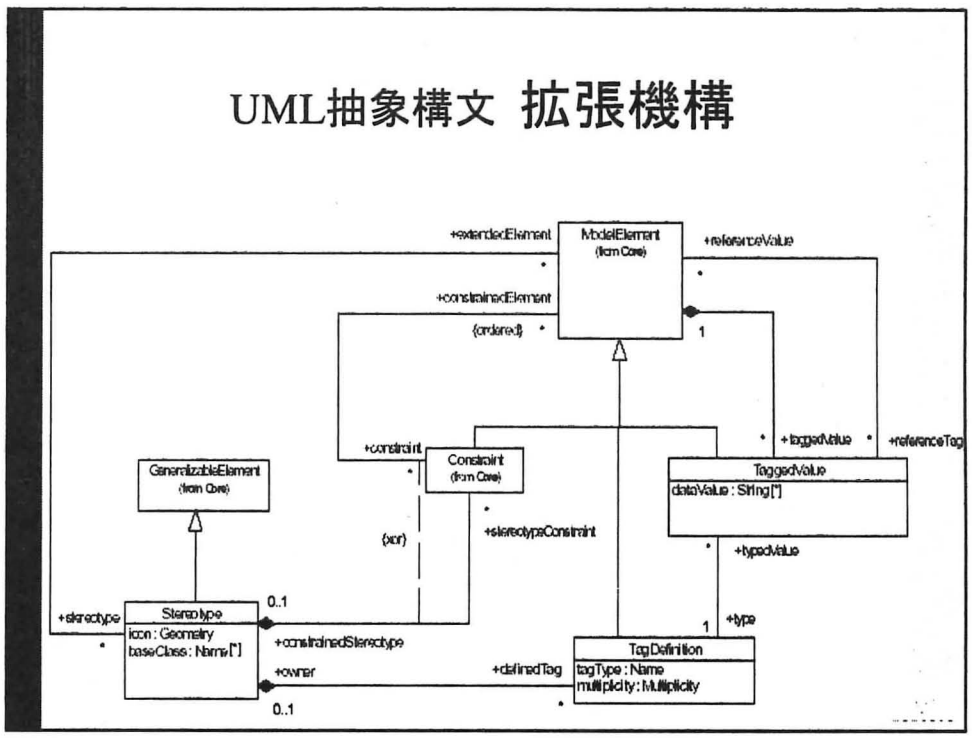
AssociationClass Semantics記述

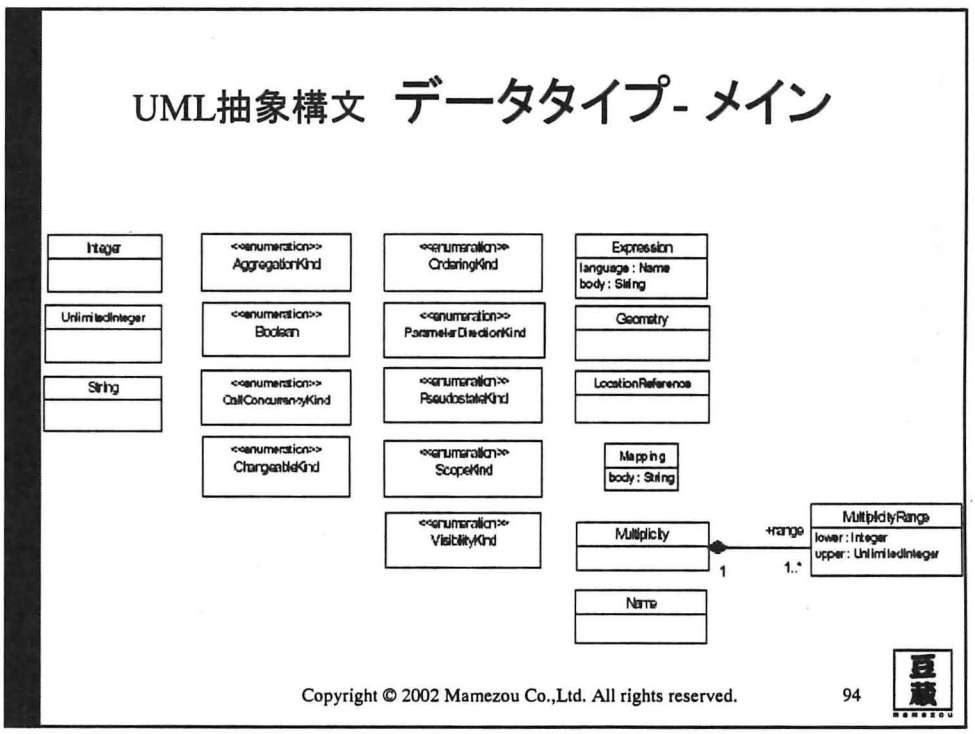
- An association may be refined to have its own set of features (i.e., features that do not belong to any of the connected classifiers) but rather to the association itself. Such an association is called an association class. It will be both an association, connecting a set of classifiers, and a class, and as such have features and be included in other associations. The semantics of such an association is a combination of the semantics of an ordinary association and of a class.
- The AssociationClass construct can be expressed in a few different ways in the metamodel (e.g., as a subclass of Class, as a subclass of Association, or as a subclass of Classifier). Since an AssociationClass is a construct being both an association (having a set of association-ends) and a class (declaring a set of features), the most accurate way of expressing it is as a subclass of both Association and Class. In this way, AssociationClass will have all the properties of the other two constructs. Moreover, if new kinds of associations containing features (e.g., AssociationDataType) are to be included in UML, these are easily added as subclasses of Association and the other Classifier.
- The terms child, subtype, and subclass are synonyms and mean that an instance of a classifier being a subtype of another classifier can always be used where an instance of the latter classifier is expected. The neutral terms *parent* and *child*, with the transitive closures *ancestor* and *descendant*, are the preferred terms in this document.

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 90

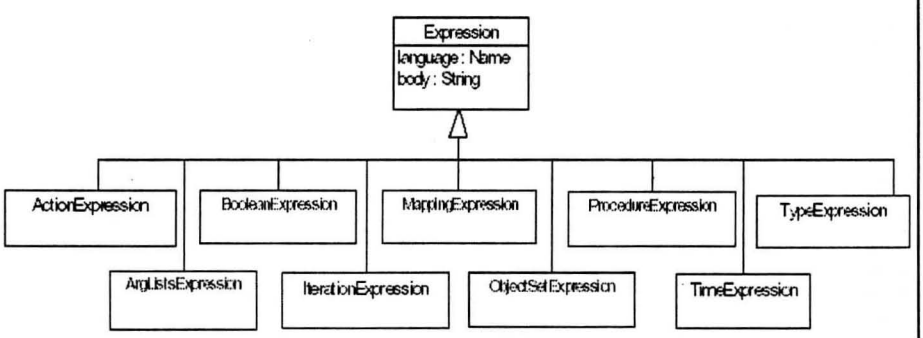

UML抽象構文 コア- 依存性
Relationship
ModelElement
name : Name
+supplier
+supplierDependency
1..*
1..*
+client
+clientDependency
Dependency
Binding
Usage
Abstraction
mapping : MappingExpression
Permission
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved.
91


UML抽象構文 コア- 分類子
Classifier
Class
isActive : Boolean
Interface
Node
+deploymentLocation
+deployedComponent
Component
+implementationLocation
+implementation
Artifact
+container
ElementResidence
visibility : VisibilityKind
+resident
DataType
ModelElement
name : Name
Primitive
Enumeration
+enumeration
1
+literal
1..*
{ordered}
EnumerationLiteral
ProgrammingLanguageDataType
expression : TypeExpression


UML抽象構文 拡張機構
+extendedElement
ModelElement
(from Core)
+referenceValue
+constrainedElement
{ordered}
1
+constraint
Constraint
(from Core)
+taggedValue
+referenceTag
GeneralizableElement
(from Core)
TaggedValue
dataValue : String[*]
{xor}
+stereotypeConstraint
+typedValue
+stereotype
Stereotype
icon : Geometry
baseClass : Name[*]
0..1
+constrainedStereotype
+type
1
+owner
+definedTag
TagDefinition
tagType : Name
multiplicity : Multiplicity
0..1

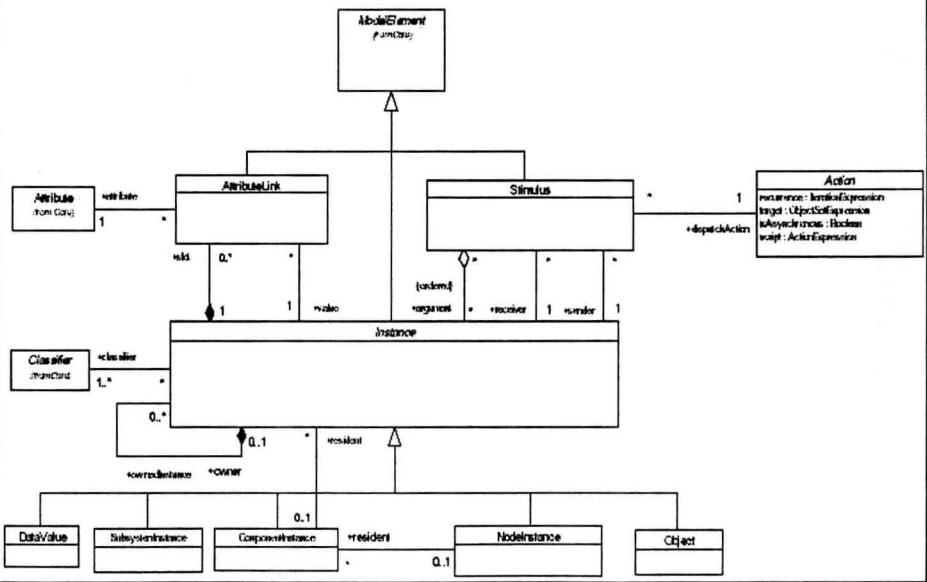

UML抽象構文 データタイプ- メイン
Integer
UnlimitedInteger
String
<<enumeration>>
AggregationKind
<<enumeration>>
Boolean
<<enumeration>>
CallConcurrencyKind
<<enumeration>>
ChangeableKind
<<enumeration>>
OrderingKind
<<enumeration>>
ParameterDirectionKind
<<enumeration>>
PseudostateKind
<<enumeration>>
ScopeKind
<<enumeration>>
VisibilityKind
Expression
language : Name
body : String
Geometry
LocationReference
Mapping
body : String
Multiplicity
+range
1
1..*
MultiplicityRange
lower : Integer
upper : UnlimitedInteger
Name
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved.
94

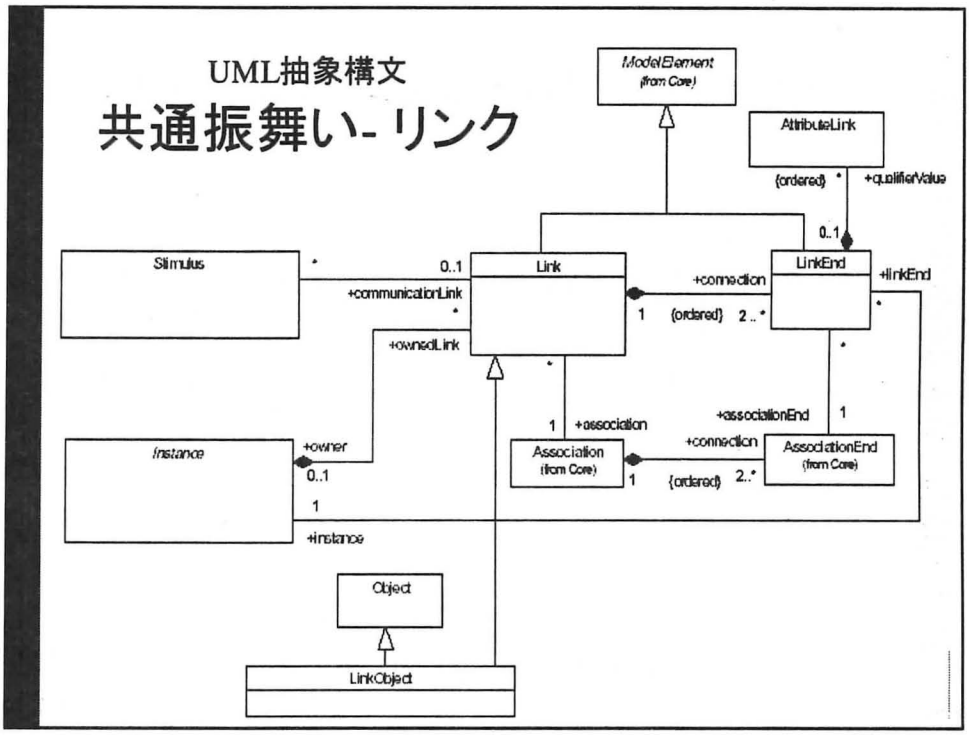
UML抽象構文 データタイプ- 式

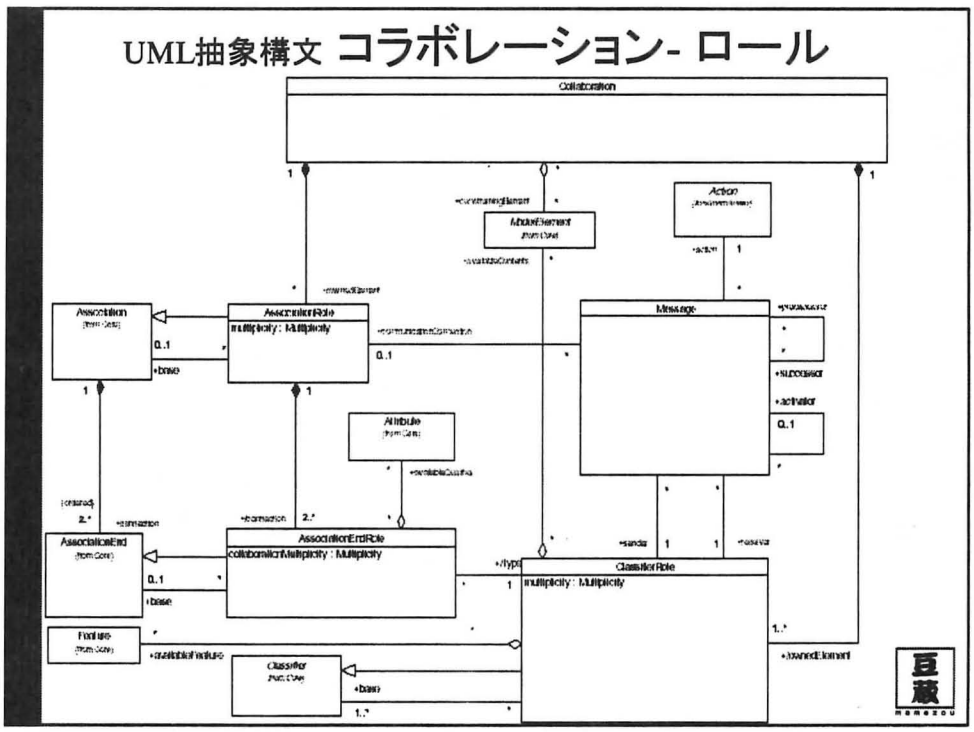


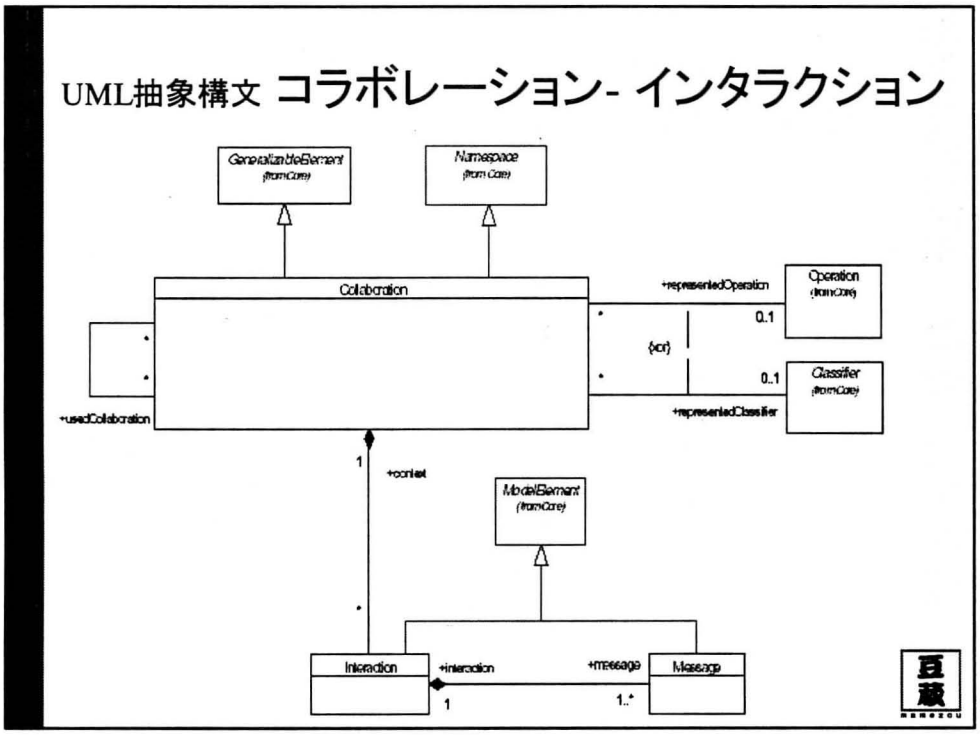
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 95

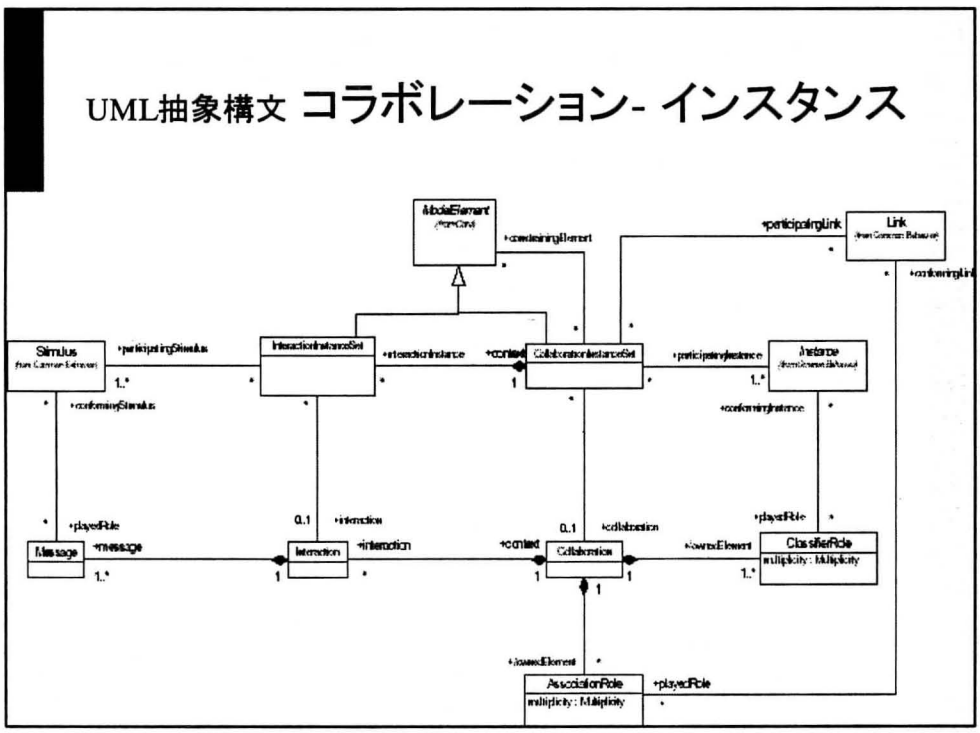
UML抽象構文 共通振舞い- インスタンス

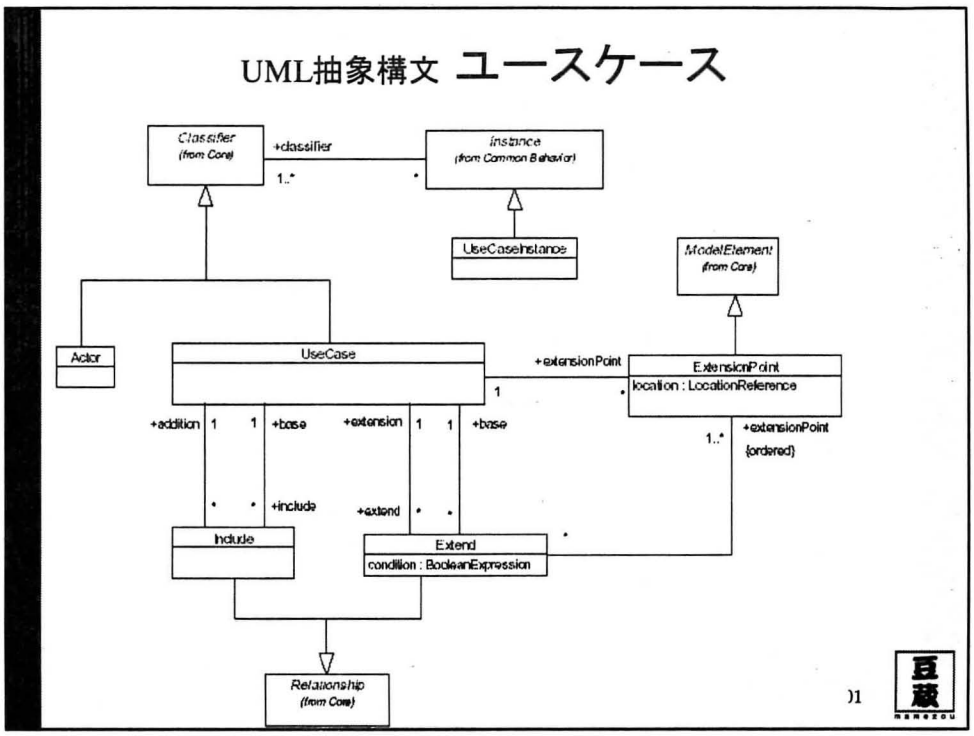


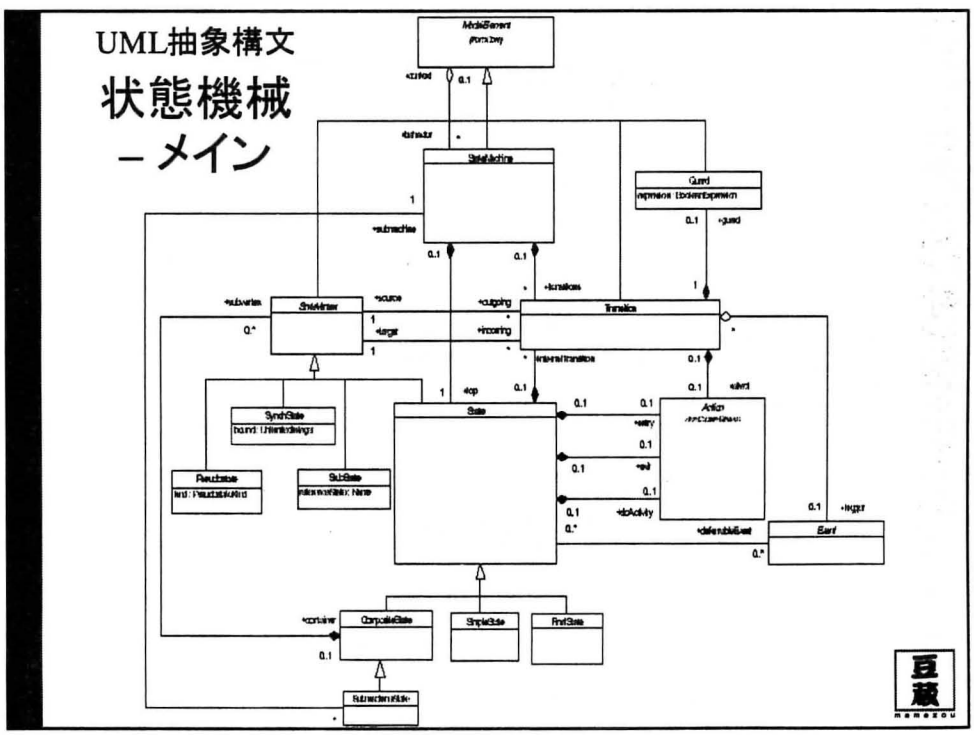

UML抽象構文
共通振舞い- リンク
ModelElement
(from Core)
AttributeLink
{ordered}
+qualifierValue
0..1
Stimulus
0..1
Link
+communicationLink
+connection
LinkEnd
+linkEnd
1
{ordered}
2..*
+ownedLink
+association
+associationEnd
1
Instance
+owner
0..1
Association
(from Core)
+connection
AssociationEnd
(from Core)
1
{ordered}
2..*
1
+instance
Object
LinkObject

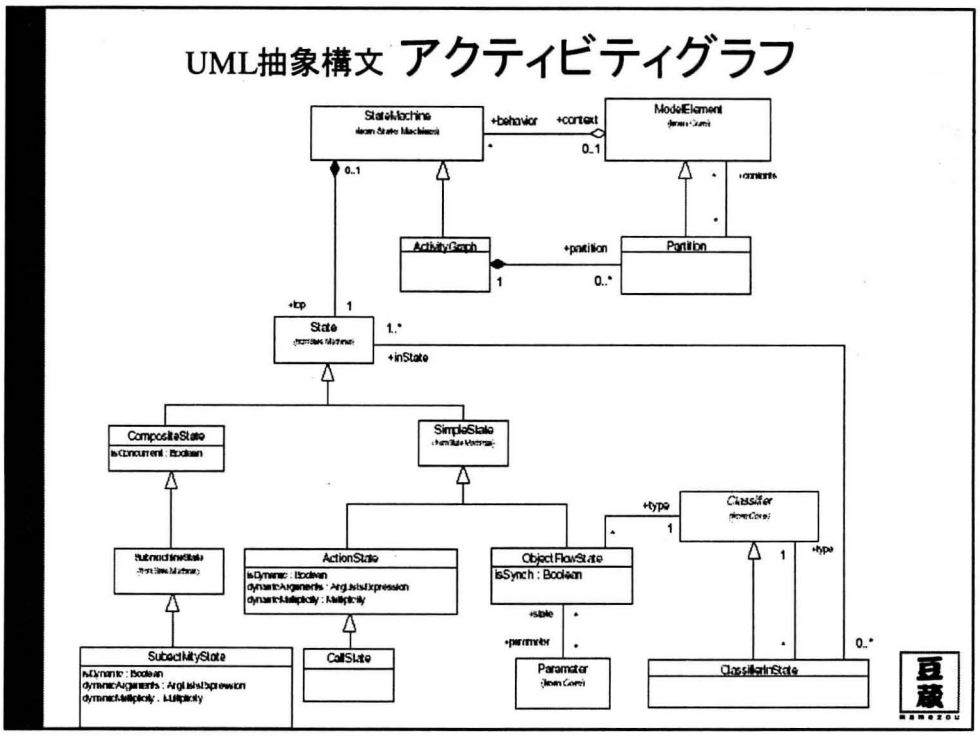

UML抽象構文 コラボレーション- ロール
Collaboration
Action
ModelElement
(from Core)
Association
(from Core)
AssociationRole
multiplicity : Multiplicity
Message
Attribute
(from Core)
AssociationEnd
(from Core)
AssociationEndRole
collaborationMultiplicity : Multiplicity
ClassifierRole
multiplicity : Multiplicity
Feature
(from Core)
Classifier
(from Core)
+base
+successor
+activator
0..1
豆蔵

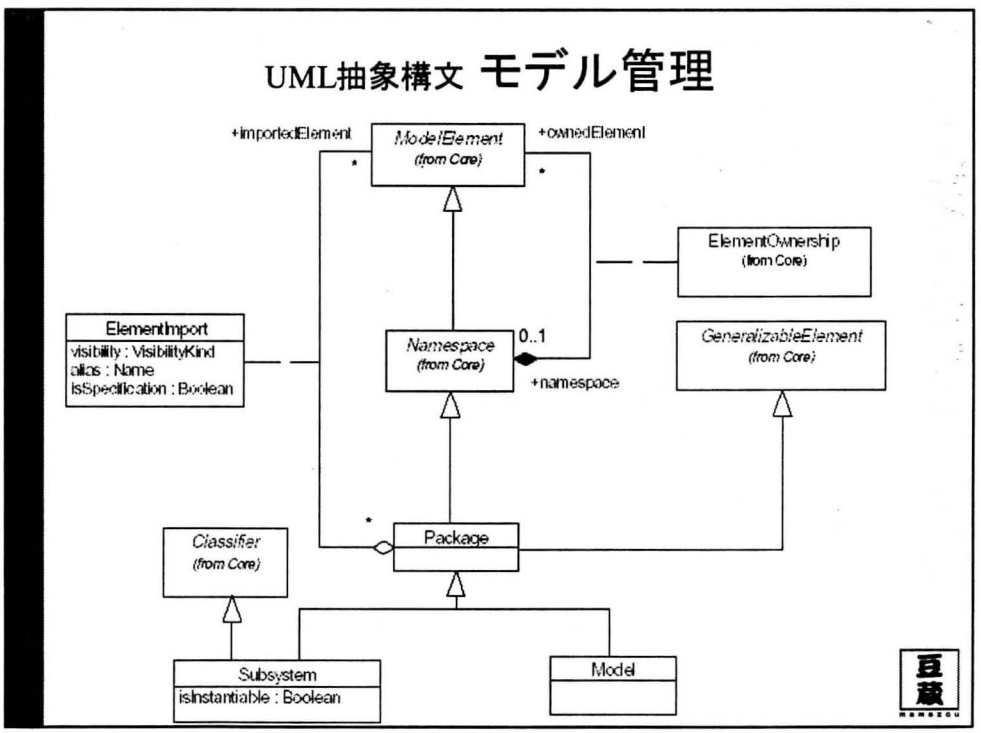

UML抽象構文 コラボレーション- インタラクション
GeneralizableElement
(fromCore)
Namespace
(fromCore)
Collaboration
+representedOperation
Operation
(fromCore)
0..1
{or}
0..1
Classifier
(fromCore)
+representedClassifier
+usedCollaboration
1
+context
ModelElement
(fromCore)
Interaction
+interaction
+message
Message
1
1..*
豆蔵


UML抽象構文 コラボレーション- インスタンス
ModelElement
+containingElement
+participatingLink
Link
Stimulus
+participatingStimulus
InteractionInstanceSet
+interactionInstance
+context
CollaborationInstanceSet
+participatingInstance
Instance
1..*
+conformingStimulus
+conformingInstance
+playedRole
0..1
+interaction
Message
+message
Interaction
+context
Collaboration
+collaboration
ClassifierRole
multiplicity : Multiplicity
+ownedElement
AssociationRole
multiplicity : Multiplicity
+playedRole


UML抽象構文 ユースケース
Classifier
(from Core)
+classifier
1..*
Instance
(from Common Behavior)
UseCaseInstance
ModelElement
(from Core)
Actor
UseCase
+extensionPoint
ExtensionPoint
location : LocationReference
+addition
+base
+extension
+base
+extensionPoint
{ordered}
+include
+extend
Include
Extend
condition : BooleanExpression
Relationship
(from Core)
)1
豆蔵


UML抽象構文
状態機械
－メイン
ModelElement
StateMachine
Guard
SubMachine
Transition
SubVertex
SubVertex
Action
State
SynchState
Pseudostate
SubState
Event
CompositeState
SimpleState
FinalState
SubmachineState
豆蔵


UML抽象構文 アクティビティグラフ
StateMachine
+behavior
+context
ModelElement
0..1
ActivityGraph
+partition
Partition
+top
State
1..*
+inState
CompositeState
isConcurrent : Boolean
SimpleState
+type
Classifier
SubmachineState
ActionState
isDynamic : Boolean
ObjectFlowState
isSynch : Boolean
+state
+parameter
SubactivityState
CallState
Parameter
ClassifierInState
豆蔵


UML抽象構文 モデル管理
+importedElement
ModelElement
(from Core)
+ownedElement
ElementOwnership
(from Core)
ElementImport
visibility : VisibilityKind
alias : Name
isSpecification : Boolean
Namespace
(from Core)
0..1
+namespace
GeneralizableElement
(from Core)
Classifier
(from Core)
Package
Subsystem
isInstantiable : Boolean
Model
豆蔵

UML抽象構文 *Subsystem* 注釈

- A subsystem is a grouping of model elements that represents a behavioral unit in a physical system. A subsystem offers interfaces and has operations.
- 仕様要素と実現要素に分離され、前者はサブシステムの操作群とともに後者によって実現される。つまりSubsystemのOperationとともにSubsystemに含まれる振舞いの仕様は、後者のModelElementによって実現される。任意の種類のModelElementは、仕様要素にも実現要素にもなりうる。
- Subsystem は PackageおよびClassifierのサブクラス。したがって、OperationかReceptionsかAssociationのいずれかの制約をもつFeatures群をもてる。
- 仕様要素と実現要素の間の関係は、コラボレーションを用いたり«realize» 依存性によって定義することができる。
- ***Attributes***
  - *isInstantiable*
    States whether a Subsystem is instantiable or not. If false, the Subsystem represents a unique part of the physical system, otherwise there may be several system parts with the same definition.

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 105

Subsystem 適格性規則の記述
追加操作

Additional Operations

[1] The operation allSpecificationElements results in a Set containing the Model Elements specifying the behavior of the Subsystem.

```
allSpecificationElements : Set(ModelElement)
allSpecificationElements = self.allContents->select(c |
  c.elementOwnership.isSpecification )
```

[2] The operation contents results in a Set containing the ModelElements owned by or imported by the Subsystem.

```
contents : Set(ModelElement)
contents = self.ownedElement->union(self.importedElement)
```

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 106

Subsystem 適格性規則の記述

[1] For each Operation in an Interface offered by a Subsystem, the Subsystem itself or at least one contained specification element must have a matching Operation.

```
self.specification.allOperations-> forAll(interOp|
      self.allOperations->union
            (self.allSpecificationElements->select(specEl|
                  specEl.oclIsKindOf(Classifier))-> forAll(c|
                        c.allOperations))-> exists( op |
                              op.hasSameSignature(interOp) ) )
```

サブシステムによって提供されるInterfaceの各Operationに対し、サブシステム自身またはそれに含まれる少なくとも1つの仕様要素は、対応するOperationを持たなければならない。

[2] For each Reception in an Interface offered by a Subsystem, the Subsystem itself or at least one contained specification element must have a matching Reception.

[3] The Features of a Subsystem may only be Operations or Receptions.

```
self.feature->forAll(f | f.oclIsKindOf(Operation) or
                         f.oclIsKindOf(Reception))
```

サブシステムのFeatureとしては、OperationあるいはReceptionのみが許される。

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 107

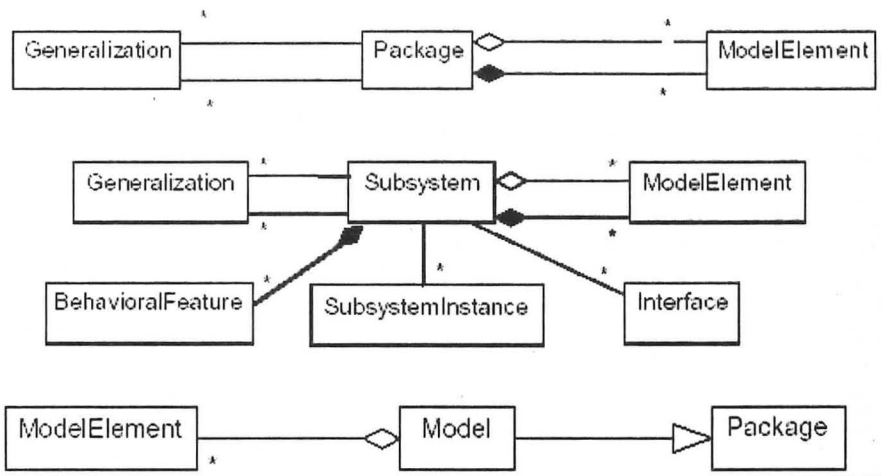
Subsystem 適格性規則の記述

- [4] A Subsystem may only own or reference Packages, Classes, DataTypes, Interfaces, UseCases, Actors, Subsystems, Signals, Associations, Generalizations, Dependencies, Constraints, Collaborations, StateMachines, and Stereotypes.

```
self.contents->forAll ( c |
     c.oclIsKindOf(Package) or
     c.oclIsKindOf(Class) or c.oclIsKindOf(DataType) or
     c.oclIsKindOf(Interface) or
     c.oclIsKindOf(UseCase) or
     c.oclIsKindOf(Actor) or c.oclIsKindOf(Subsystem) or
     c.oclIsKindOf(Signal) or c.oclIsKindOf(Association) or
     c.oclIsKindOf(Generalization) or
     c.oclIsKindOf(Dependency) or
     c.oclIsKindOf(Constraint) or
     c.oclIsKindOf(Collaboration) or
     c.oclIsKindOf(StateMachine) or
     c.oclIsKindOf(Stereotype) )
```

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 108

パッケージ、サブシステム、モデル



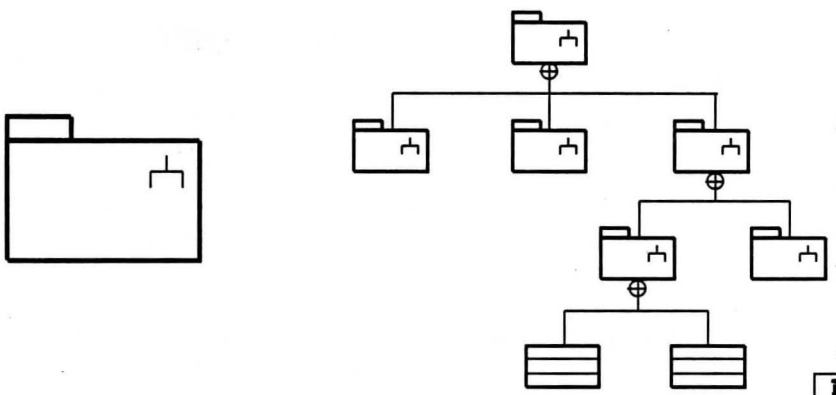
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 109

Subsystem Semantics部の記述
(動的意味論の指定; 要点のみ一部)

- The purpose of the *subsystem* construct is to provide a grouping mechanism for specifying a behavioral unit of a physical system. Apart from defining a namespace for its contents, a subsystem serves as a specification unit for the behavior of its contained model elements.
- The contentsof a subsystem are divided into two subsets: 1) *specification elements* and 2) *realization elements*.
- The *specification* of a subsystem thus consists of the specification elements together with the subsystem's features (operations and receptions).
- A subsystem has no behavior of its own.
- The correspondence between the specification and the realization of a subsystem can be specified in several ways, including collaborations and «realize» dependencies.
- *Importing* and *accessing* subsystems is done in the same way as with packages, using the *visibility* property to define whether elements are public, protected, or private to the subsystem. Both the specification part and the realization part of a subsystem may include imported elements.
- A subsystem can have *generalizations* to other subsystems.
- A subsystem may offer a set of *interfaces*.
- In cases when the physical system has several parts with the same definition, the subsystem is specified to be *instantiable*.

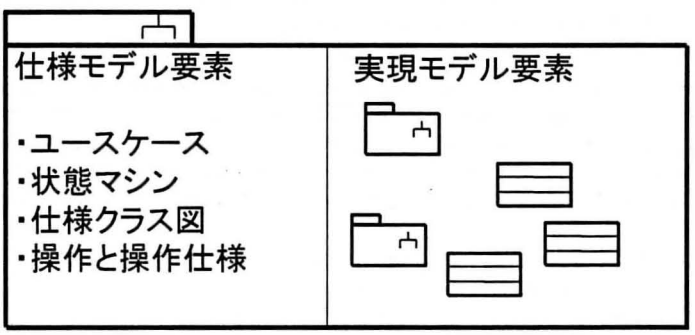
Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 110

サブシステムとは



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 111

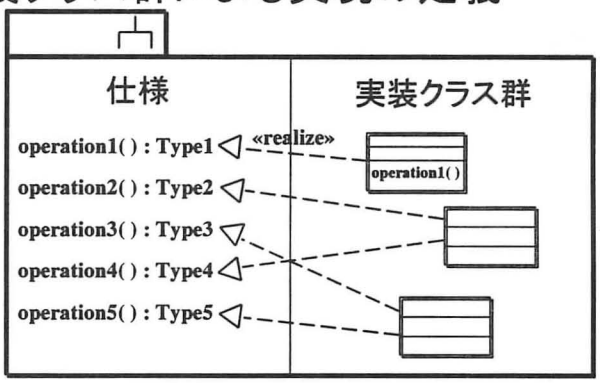
サブシステムの仕様と実現



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 112

サブシステム：操作仕様と実現

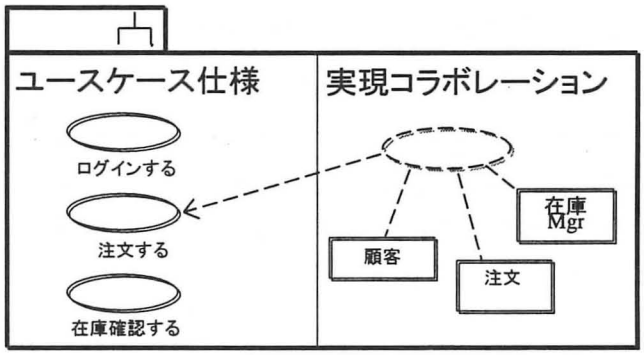
- 操作群による仕様の定義
- 実装クラス群による実現の定義



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved.

サブシステム：ユースケースとコラボレーションによる仕様実現

- 仕様部はユースケース群
- 実現部はオブジェクト群とそのコラボレーション



Copyright © 2002 Mamezou Co.,Ltd. All rights reserved.

UMLの今後とまとめ

- ▶UMLの最近のバージョン
- ▶UML2.0に向けて
- ▶まとめ
- ▶参考ソース

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 115

UMLの最近のバージョン

- ▶UML1.1：初めてのOMG標準 1997.11公開
- ▶UML1.3：技術的改善 1999.8公開
  - ▶記法・メタモデルの不整合の除去
  - ▶モデル交換フォーマットの定義(XML)
- ▶UML1.4：minor upgrade 2001.9公開
- ▶UML2.0：major upgrade 2002? 2003??公開
  - ▶実行モデル記述用アクションセマンティクス
  - ▶ステレオタイプ以上に柔軟な言語拡張機構
  - ▶版・構成管理の仕組み
  - ▶UMLセマンティクスの形式的仕様記述？

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 116

UML2.0に向けて

- 拡張性に関する改良
  - 拡張機構を抜本的に改良する
  - 4階層メタモデル構成と整合する拡張機構の実現。言語のカスタマイズのためのプロファイル仕様の規定。
- アーキテクチャ
  - MOFメタモデルとの整合を保証する形で物理メタモデルを定義する
  - カーネル言語、UMLプロファイル、標準モデルライブラリの再構成とガイドライン作成
- コンポーネント
  - EJBやDCOM等コンポーネントベース開発のサポート拡張

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 117

UML2.0に向けて(2)

- 関係
  - 洗練とトレースに対する実質的なセマンティクスを追加
  - 複数の抽象レベルで関連のセマンティクスを定義する
- ステートチャート図とアクティビティ図
  - それぞれ別のセマンティクスで定義
  - 並列性を認める。状態マシン間の汎化関係の定義
- モデル管理
  - マルチビュー実現のためモデルとサブシステムの詳細定義
- 一般メカニズム
  - モデルの版管理機構の定義
  - 図交換機構

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 118

まとめ

- UML仕様書の読み方のポイントを示した
  - UML記法ガイドだけでなく
  - UML意味論の本文が読める最低限の知識を紹介
- UML自体は言語として未成熟
  - 現状のUML1.4からUML2.0へ
- UML利用は目的に応じて
  - S/W開発で必要な部分を適宜利用すればよい
  - 世界的に使われていて便利
  - きめ細かい仕様や設計の指定にはOCLも利用

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 119

UML情報源

- UML, MOF等OMGモデリング標準
  - http://www.omg.org/uml/
  - http://www.dstc.edu/au/mof
- UML 2.0情報
  - http://www.celigent.com/omg/adptf/wgs/uml2wg.htm

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 120

UML標準テキスト(UML1.3ベース)

- UML仕様書
  - 2000 March, OMG UML Specification, Version1.3
    - 2001、アスキー、OMG Japan SIG翻訳委員会UML作業部会
- 3 Amigosの著作がAddison-Wesleyより
  - ユーザーガイド: Unified Modeling Language User Guide
    - 1999, Grady Booch, and Ivar Jacobson, James Rumbaugh
    - 1999、ピアソン、羽生田/オージス総研訳
  - レファレンスマニュアル: Unified Modeling Language Reference Manual
    - 1999, James Rumbaugh and Grady Booch, Ivar Jacobson
    - 2002、ピアソン、石塚/日本ラショナル訳
  - UMLベースの開発プロセス: The Unified Software Development Process
    - 1999, Ivar Jacobson, and Grady Booch, James Rumbaugh
    - 2000、ピアソン、藤井/日本ラショナル訳
- OCL解説
  - The Object Constraint Language, Precise modeling with UML
    - 2000, Addison-Wesley , Jos Warmer, Anneke Kleppe, ISBN 0-201-37940-6

Copyright © 2002 Mamezou Co.,Ltd. All rights reserved. 121 豆蔵 mamezou

UMLセミナー

モデルとUML

2002年2月7日

玉井 哲雄
東京大学 大学院情報学環

モデルの図式表現

- モデルとは
  - ◆構造を持った対象を，その性質や動作を理解するために抽象化したもの
- モデル構築の目的
  - ◆対象世界（領域）の把握 → 分析
  - ◆開発するシステムの表現 → 要求仕様

図式による記述

- モデル記述言語の構文として図式，とくにグラフを使うものが多い．
  - ◆「モデル」らしい
  - ◆モデルを，基本的構成要素とそれらの間の関係として捉えることは，一般性が高い．
  - ◆推移的な関係の追跡が，グラフの経路解析に対応
- 視覚的効果には限界
  - ◆ちょっと大きな図になると訳が分からない．
  - ◆絵を描いただけでモデルができた気になる．

図式表現の例

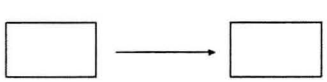
- 制御フロー
  - ◆フローチャート，PAD，HCP，SPD
- データフロー
  - ◆DFD，SADT，HIPO
- データ構造
  - ◆ジャクソン構造図
- モジュール構造
  - ◆モジュール階層図
- データ関連
  - ◆ER図，意味ネットワーク，クラス図
- プロセス関連
  - ◆JSDシステム仕様図，STATEMATEモジュール図
- 状態遷移
  - ◆状態遷移図，Statechart，ペトリネット

共通なグラフ構造

- 丸だか四角だか箱 → 頂点
- 折れ線だか点線だか矢印 → 辺
- グラフ構造
  - ◆頂点集合：S
  - ◆辺集合：S×Sの部分集合

グラフ表現はソフトウェアには限らない

- 教科書，報告書，企画書，論文，発表スライド，．．．で用いられる図の多くがグラフ表現
- 箱と結ぶ線（箱は丸でも四角でも三角でもよい．線は矢印があってもなくてもよい）



これらの図は往々にして曖昧

■矢印の意味の揺らぎ，混在
◆因果関係，時間関係，順序関係
◆ものの流れ，情報の流れ，制御の流れ
■箱の意味の揺らぎ，混在
◆処理とデータ，動作主体と動作対象
■矢印や箱の種類を分けるのが一つの手段だが，簡明さを失えば価値は減少
■[例]

グラフによる静的モデルと動的モデル

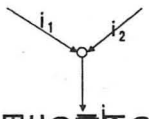
■静的モデル
A ー B
◆AとBはある関係にある
A → B
◆AはBとある関係にある(ERモデル)
■動的モデル
A → B
◆AからBに動く/移動する(状態遷移，フローチャート)
◆AからBに何かが流れる(データフロー)

グラフによるモデルの解釈

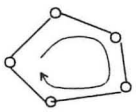
■グラフと物理量
◆グラフは接続関係(topology)を与えるだけ
◆多くは頂点と辺に物理量を結びつけモデル化
★電流@辺，電位@頂点
■物理量とトポロジーとを結びつける法則
◆Kirchhoff's Law(キルヒホッフの法則)

Kirchhoff's Law

■電流の法則： 頂点に流入(出)する電流の総和は0



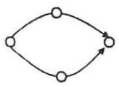
■電圧の法則： ループの周りの電圧の総和は0



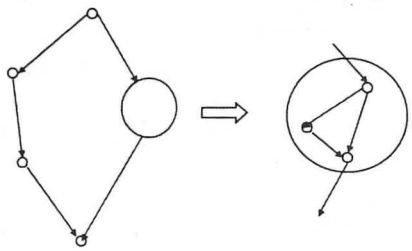
2つの見方

■頂点の周りで
■経路に沿って
◆直列

◆並列



部分グラフ

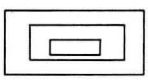


グラフの弱点

- ■1枚の図に描ける頂点・辺の数は限られる. ⇒ 魔法の数字7±2(G.A.Miller)
  - ◆物理的制約(紙, ディスプレイ)
  - ◆認識上の制約
  - ◆クモの巣状態

いくつかの工夫

- ■配置, 線の交差の最小化, その他の美化
- ■ズーミング, スクローリング
- ■一般化(ハイパーグラフ)
- ■階層化
  - ◆上位グラフの頂点を展開すると下位グラフとなる
  - ◆頂点(箱)の入れ子構造
  - ◆(Statechart)



UMLの位置付け

- オブジェクト指向モデル記述言語
- これまでの分析や設計ノウハウを図を中心に集大成
- グラフ構造を持つモデル表現法のカタログ

ここでの視点

- ■分析設計ノウハウとモデル表現のカタログという視点
- ■グラフとしての特性に注意
- ■モデル(図)の間の差異の明確化
  - ◆表現力を増そうとすると
    - ★モデル(図)が複雑化
    - ★異なるモデルが相互に似てくる
  - ◆夾雑物を取り去り, 本質を把握

UMLの図式

- ■構造図
  - ◆クラス図
  - ◆オブジェクト図, コンポーネント図, 配備図
- ■振舞い図
  - ◆動作図, 系列図, 協調図, Statechart図
  - ◆使用事例図
- ■モデル管理図
  - ◆パッケージ図, サブシステム図, モデル図

UML図のグラフ構造

	頂点	辺
クラス図	クラス	汎化, 集約, 関連
状態遷移図	状態	遷移
動作図	動作状態	制御の流れ
協調図	オブジェクト	メッセージの流れ
系列図	メッセージ受発信の時点	メッセージの流れ

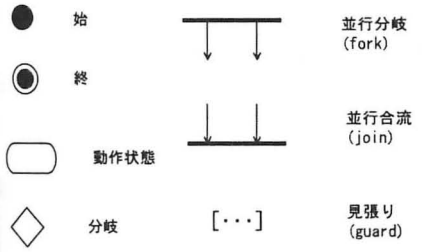
UMLにおける振舞いモデル

- ■使用事例 (use case)
- ■相互作用 (interaction)図
  - ◆系列 (sequence)図
  - ◆協調 (collaboration)図
- ■動作 (activitiy)図
- ■状態遷移図 (Statechart)

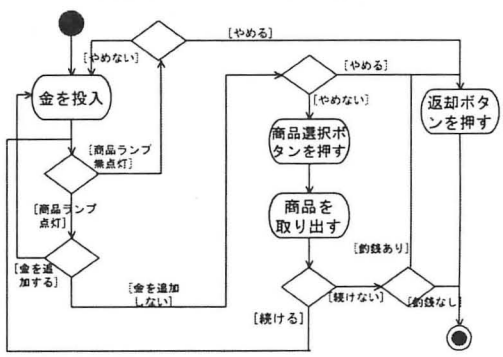
動作図
Activity Diagrams

- ■動的, 制御の流れ
  - ◆手順を図示
- ■フローチャートと類似, ただし
  - ◆概念レベルのモデルに使用することも想定
  - ◆並行動作の記述が可能
- ■用途
  - ◆システムの動作(状態遷移図, 系列図, 協調図と関連)
  - ◆ワークフロー, ビジネスプロセス, 開発プロセス

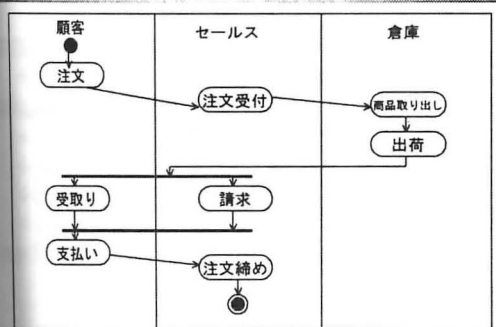
動作図の要素



自動販売機 動作図(利用者の視点)



動作主体の明示
(Swimlane;プールのコース)



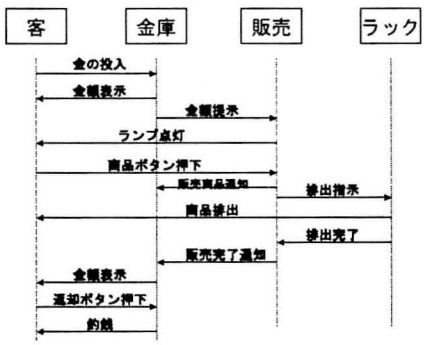
動作図のグラフ特性

- ■経路 → 実行経路
  - ◆分岐はOR, 並行分岐はAND
  - ◆PERTのような解析も可能
- ■部分グラフ
  - ◆「動作状態」の展開と縮約
    - ★動作状態をまた動作図として展開できる.

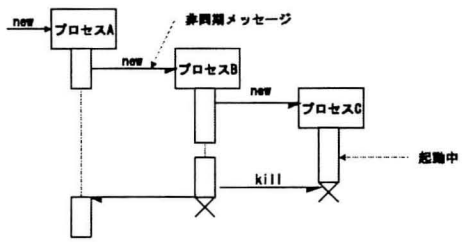
系列図

■**辺**: メッセージ送信/メソッド起動
頂点: (明示的でないが)オブジェクトが発信・受信した状態の開始点
縦方向(時間軸)に沿って隠れた別種の辺の存在を想定してもよい.
■**時間軸**: 上から下に流れる
■**オブジェクト**: メッセージを受発信. それぞれ1本の縦軸に対応
■**なお, メッセージの発信条件を[条件]という記法で書いてもよいなど, 他にいくつかの拡張がある.**

自動販売機 系列(sequence)図



並行プロセスの表現



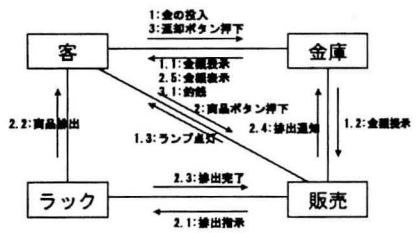
系列図のグラフ特性

■グラフとしてはかなり特殊
■部分グラフによる再帰構造はなし

協調図

■**頂点**: オブジェクト
辺: オブジェクト間の結合
メッセージ: 辺に向きを付けて関連づけ
番号で時間順序を表現
10進式番号で付随メッセージを表す.
■**意味的には系列図と等価. 系列図と同様, メッセージの発信条件の記述などの拡張がある.**
(しかし, 系列図と等価な意味づけをするのは, 意味がないのではないか?)

自動販売機 協調図



グラフとしての特性

■協調図をオブジェクト間の相互作用を表現した図として用いるなら，部分グラフによる階層構造は，想定できる．
■オブジェクトの連結成分分解などの操作も可能

状態遷移モデル

■頂点 → 状態(持続時間を想定)
辺 → 遷移(瞬時)
■事象： 遷移を引き起こすもの．通常システムの外部から来る．(状態は通常外部から見えない)
■状態の中に
◆初期状態(通常1つ)
◆終了状態(一般に1つ以上)

状態遷移モデル(続き)

■状態機械は，外部から見ると記憶を持つ機械として見える．
■その記憶，すなわち内部状態は，事象の列で決まるが，事象の列は内部的には状態遷移図の経路に対応する．
■1つの状態機械は，ある時点ではちょうど1つの状態にいる．

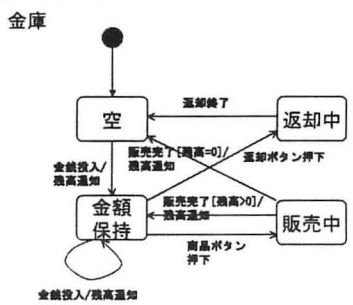
状態遷移モデルの拡張

■遷移に際しての出力 → 事象の発生と解釈してもよい
■遷移に際しての条件
■並行プロセス
◆1つの状態遷移図は1つのプロセス
◆複数の状態遷移図が事象を共有(相互作用)
◆CSP(C. A. R. Hoare), CCS(R. Milner)
◆ペトリネット

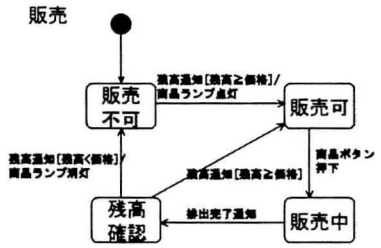
フローチャートとの区別

■類似点
◆動的:「今」どこにいるかが意味をもつ．
■相違点
◆頂点の意味:FCは処理単位，STは状態
◆「実行」の起こる場所:FCは頂点，STは辺
◆移動のきっかけ:FCは処理の終り，STは事象
◆途中での外部とのやりとり:FCなし，STあり
■結局，FCは変換型，STは反応型向き

自動販売機の例

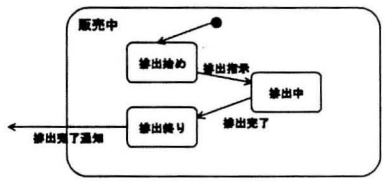


自動販売機の例(2)



自動販売機の例(3)

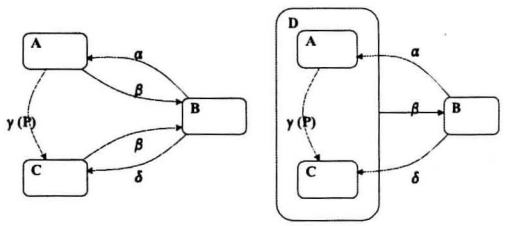
販売の「販売中」状態の
階層的分解



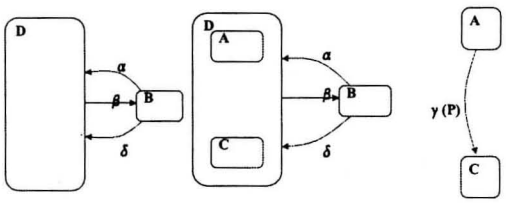
Statechart

- D. Harelの考案による状態遷移モデルの拡張
- 関連ツール: StateMate
- OMTやUMLでもオブジェクトの動的振る舞いを記述する記法として導入
- 状態の爆発を防ぐ工夫
  - ◆状態のクラスター化による階層構造
  - ◆And結合による並行プロセス

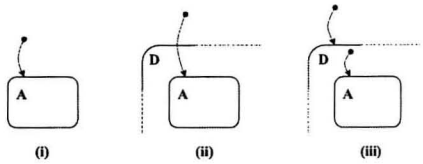
Statechartの記法(1)



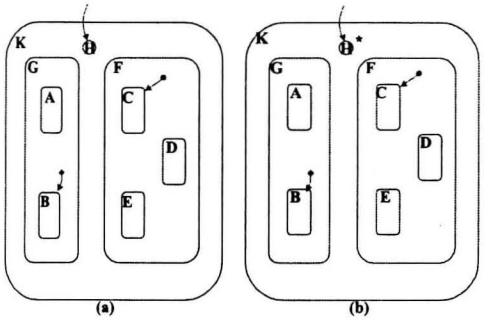
Statechartの記法(2)



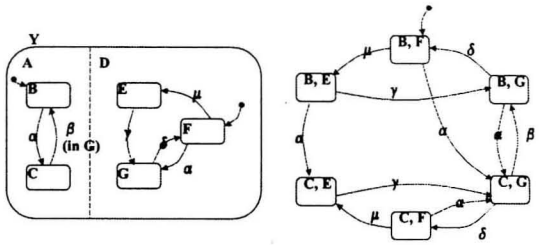
Statechartの記法(3)



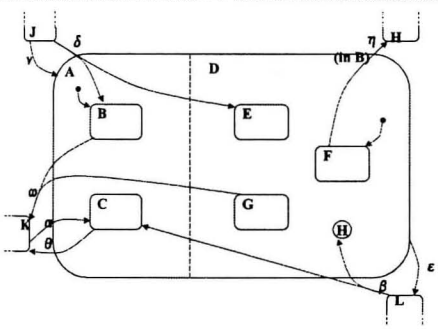
Statechartの記法(4)

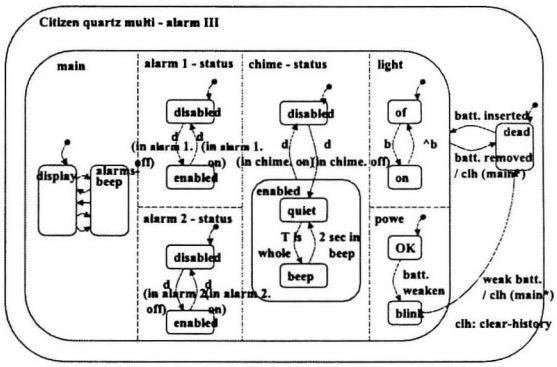


Statechartの記法(5)



Statechartの記法(6)





クラス図

■オブジェクト指向モデルにおけるクラスの関係を示す図

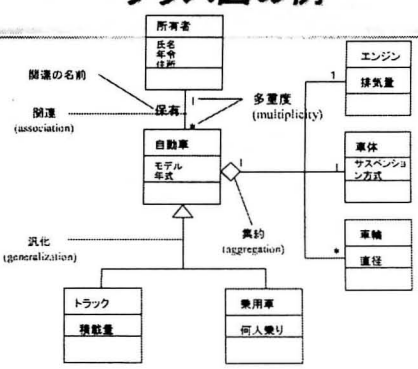
■頂点：クラス(クラス名，属性，演算)
辺：静的関係

- 汎化(generalization)
- 関連(association)
- 集約(aggregation)
- 依存(dependency)

ラベル

- 名前
- 役割(role)
- 多重度(multiplicity)

クラス図の例



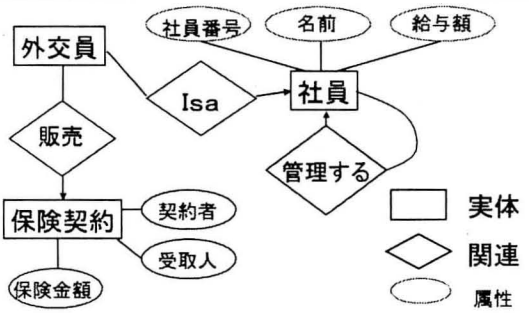
クラス図に先行する類似モデル

■ER図
■意味ネットワーク

ERモデル

■**Peter Chen. The entity-relationship model: Towards a unified view of data.** *ACM Transactions on Database Systems*, 1(1), 1976.
■その後，属性(attribute)を加えるなど，さまざまな拡張，変形

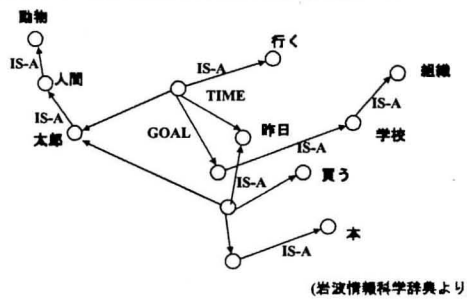
ERモデルの例(保険)



意味ネットワーク

■**M. Ross Quillian. Semantic memory.** In M. Minsky, editor, *Semantic Information Processing*. MIT Press, Cambridge, MA, 1968.
M. Ross Quillian. The teachable language comprehender, *Communications of the ACM*, Volume 12 Issue 8, August 1969.

意味ネットワークの例



(岩波情報科学辞典より)

UMLについての補足的なコメント

■クラス図から使用事例への力点の移動
◆OOSEの流儀が主流化
◆実世界の分析モデルよりシステム設計モデルにシフト
■テキスト表現の扱い
◆たとえばOCLなどの動き
◆Harel等のアプローチ

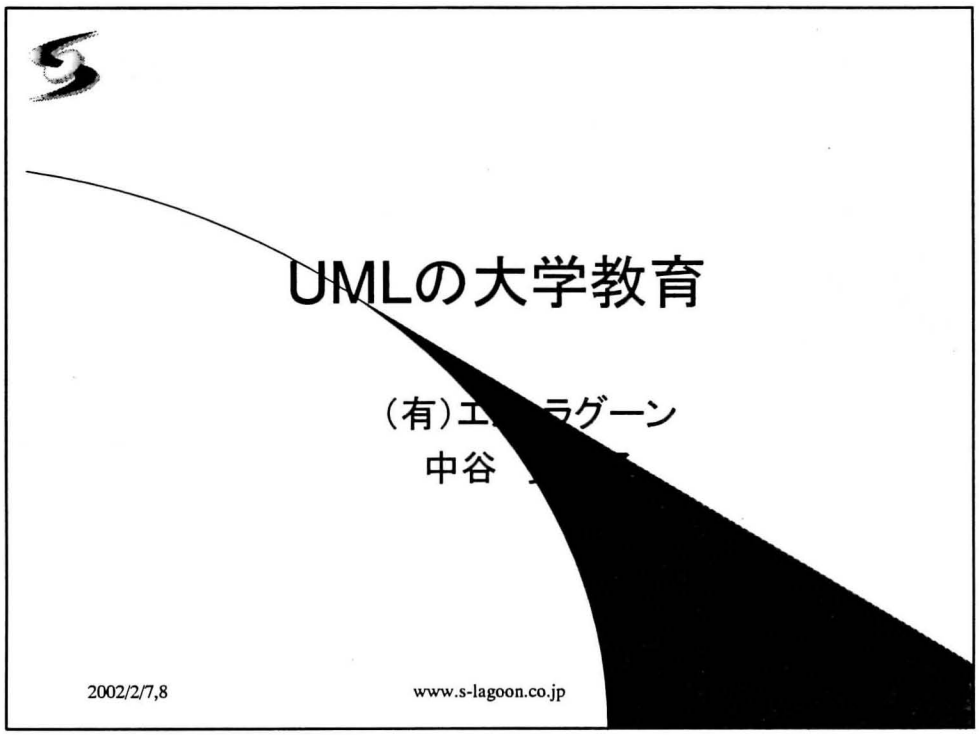
UMLについての補足的なコメント(2)

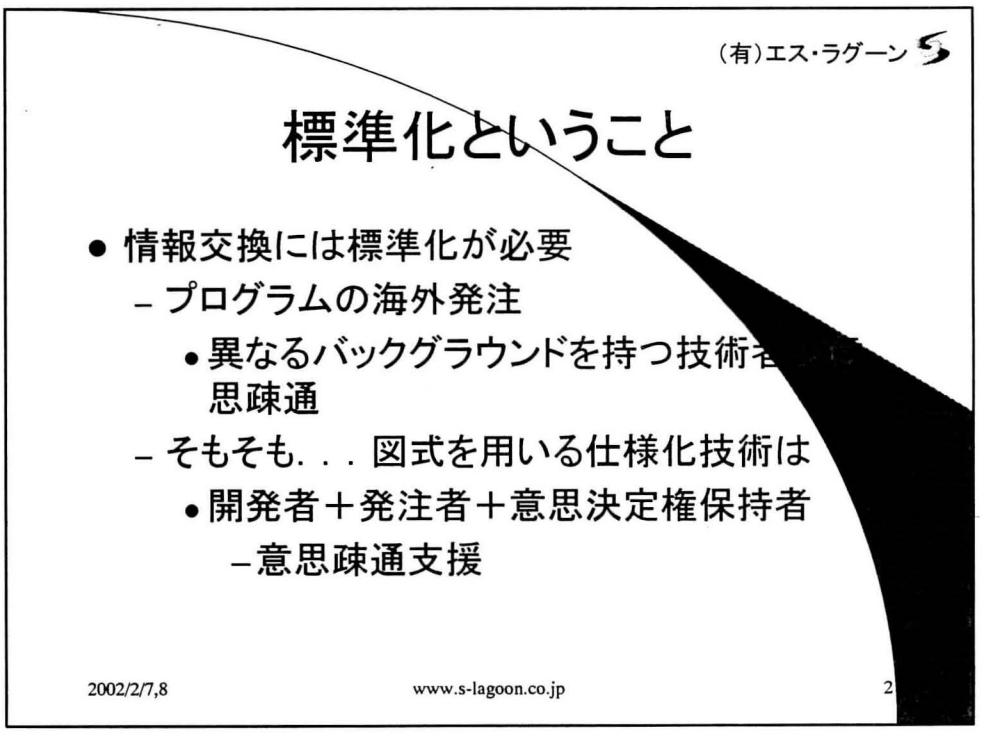
■表現モデルと解法モデル
- ◆UMLでは表現が目的のモデル
  - ★実世界/領域の写像
  - ★システム設計としての表現
- ◆問題解決を目指しグラフ上に定式化されるモデル
  - ★グラフ上の最適化問題(最短路問題, 最大流問題, 最小費用流問題, 輸送問題, スケジューリング, さまざまな組み合わせ最適化)
  - ★方程式の求解(均衡解, 定常流, 振動, . . .)
  - ★解法を伴うグラフ上の問題モデル

まとめ

■UMLの各図には, 分析・設計のノウハウが凝縮されている.

■各図の差異を, 明示的に意識すべきである.


UMLの大学教育
(有)エ ラグーン
中谷
2002/2/7,8
www.s-lagoon.co.jp


(有)エス・ラグーン
標準化ということ
● 情報交換には標準化が必要
– プログラムの海外発注
● 異なるバックグラウンドを持つ技術者
思疎通
– そもそも. . . 図式を用いる仕様化技術は
● 開発者＋発注者＋意思決定権保持者
–意思疎通支援
2002/2/7,8
www.s-lagoon.co.jp
2

(有)エス・ラグーン

UMLと大学教育

- 大学生は何を学ぶべきか
  - オブジェクト指向は大前提として. .
  - Java, C++も大前提として. .
  - 開発方法論, UMLは？
    ＝開発者間の相互理解は必要か？

2002/2/7,8 www.s-lagoon.co.jp 3

(有)エス・ラグーン

企業が導入する・している開発形態

- 海外への発注
- アーキテクトによるアーキテクチャ中心開発
  - コンポーネントに基づく開発
    - インタフェースの早期決定
    - コンポーネント内はより小さな組織で開発
- パターン導入による知識, ノウハウの蓄積と共有
  - ノウハウの文書化は重要という認識

2002/2/7,8 www.s-lagoon.co.jp 4

(有)エス・ラグーン

UMLの位置づけ

- 開発方法論，UMLは？
  ＝開発者間の相互理解は必要か？
- 発注者:受注者
  - 海外への発注＝曖昧性のない仕様書
- アーキテクト:コンポーネント設計者
  - クラス図，シーケンス図による意思疎通
- ベテラン:初心者
  - パターンによるノウハウの伝授

2002/2/7,8 www.s-lagoon.co.jp 5

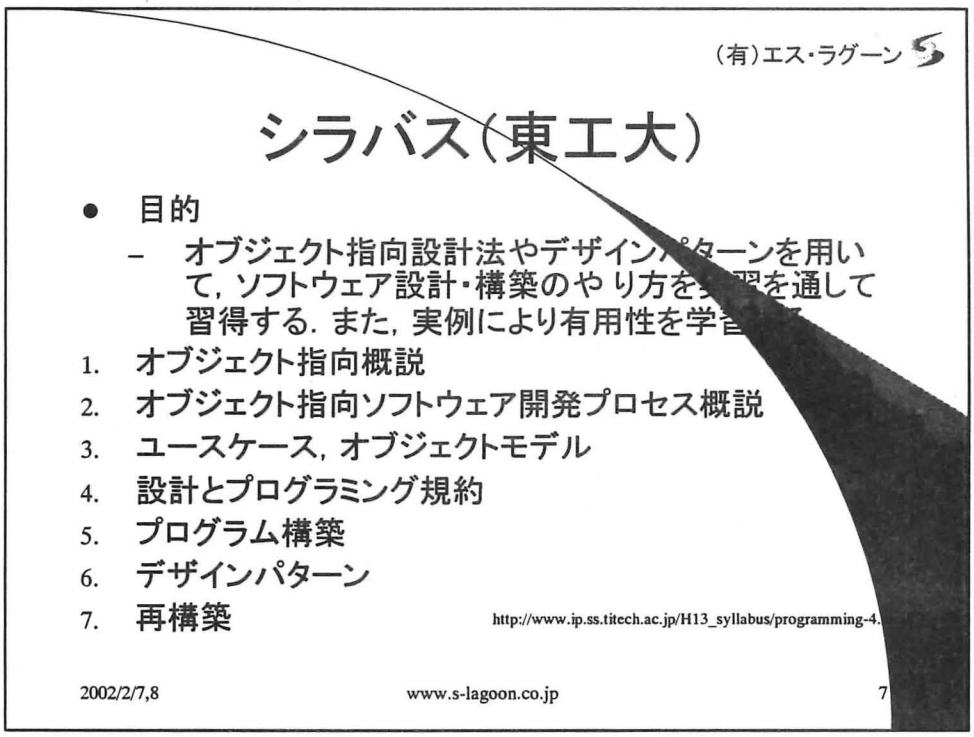
(有)エス・ラグーン

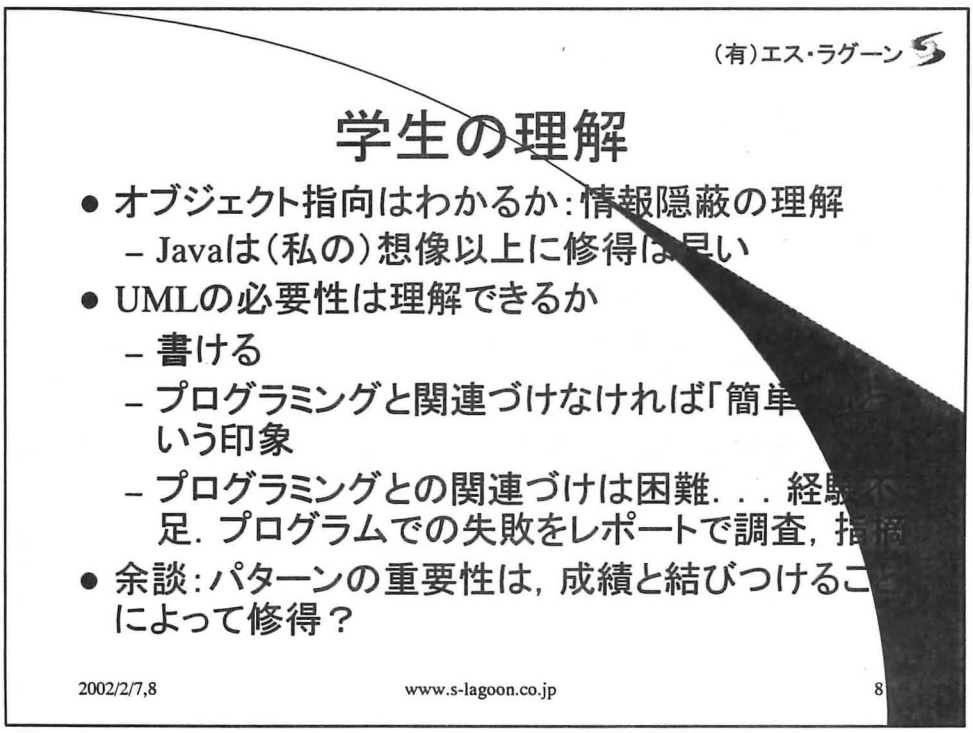
大学教育との関わり

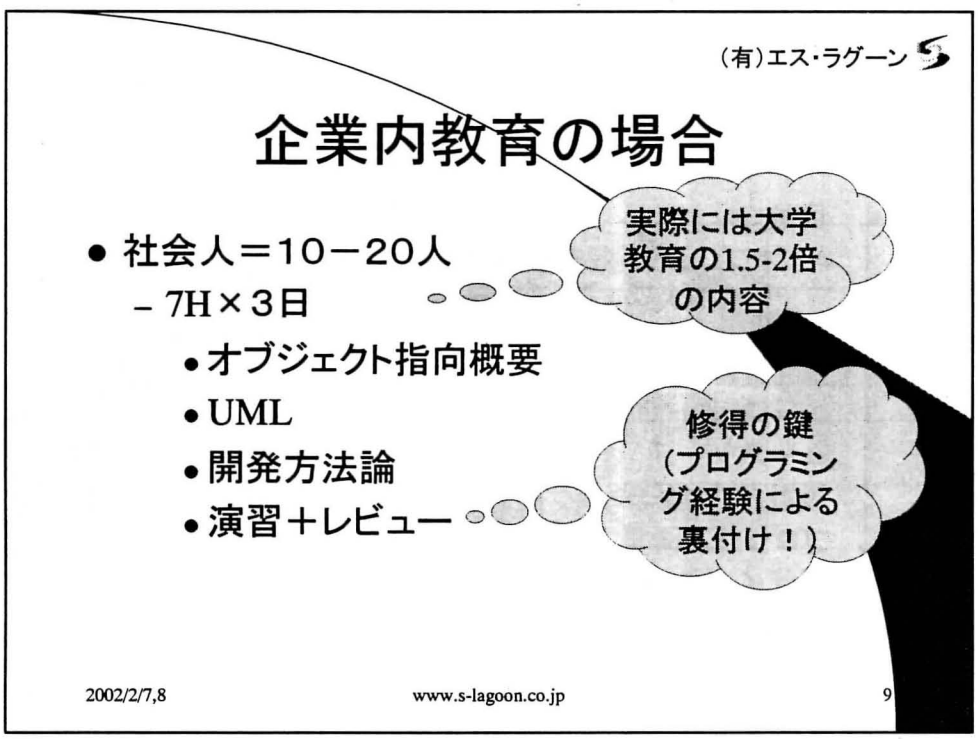
- 東京工業大学(情報工学科，3年)
  - 30−40人
- 和歌山大学(システム工学部，3年)
  - 40−50人
- 芝浦工業大学(システム工学部，3年)
  - 60−70人
- 日本工業大学大学院(工学研究科1年)
  - 5−8人
- 1.5H×12回＝18H＝7H×3日！

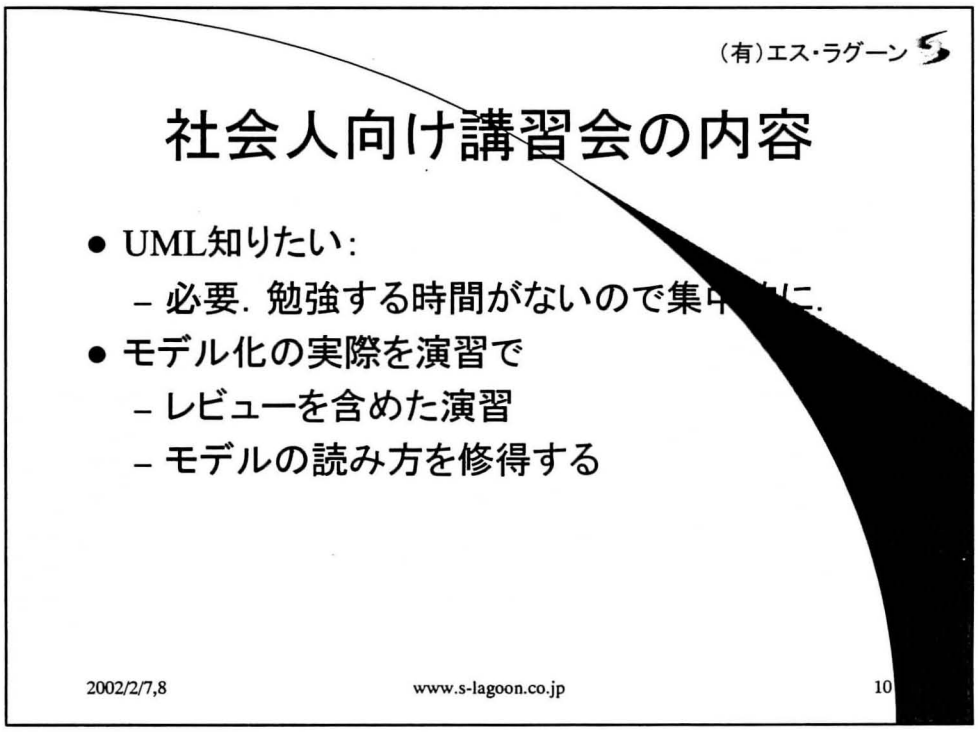
木目の細かい
指導は困難

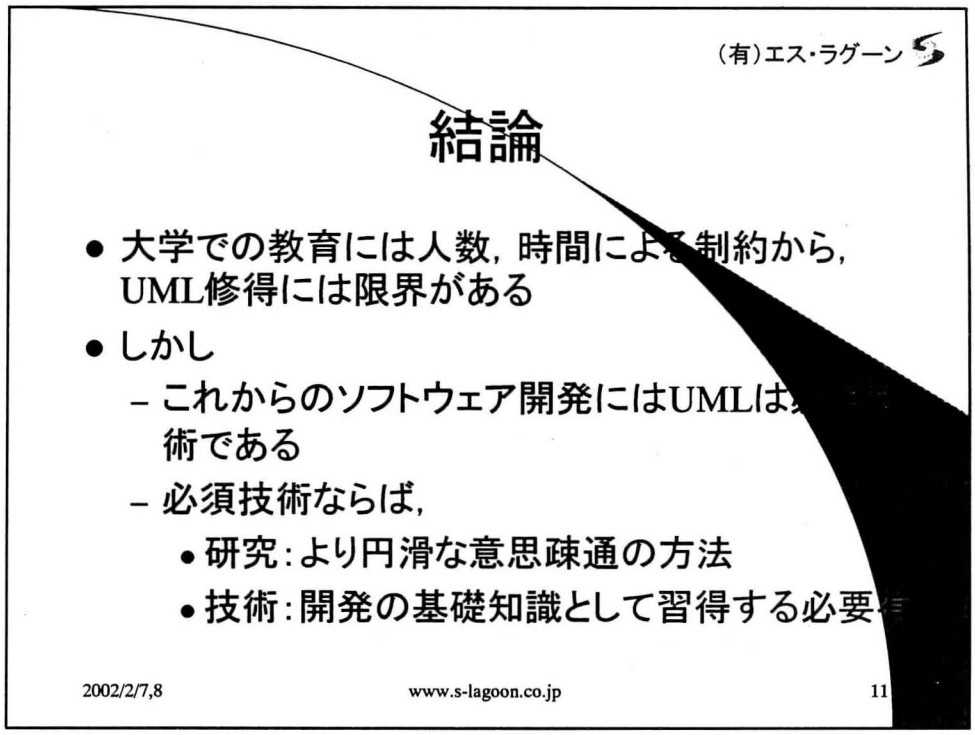
2002/2/7,8 www.s-lagoon.co.jp 6


(有)エス・ラグーン
シラバス(東工大)
目的
– オブジェクト指向設計法やデザインパターンを用いて, ソフトウェア設計・構築のやり方を実習を通して習得する. また, 実例により有用性を学習
1. オブジェクト指向概説
2. オブジェクト指向ソフトウェア開発プロセス概説
3. ユースケース, オブジェクトモデル
4. 設計とプログラミング規約
5. プログラム構築
6. デザインパターン
7. 再構築
http://www.ip.ss.titech.ac.jp/H13_syllabus/programming-4.
2002/2/7,8
www.s-lagoon.co.jp
7


(有)エス・ラグーン
学生の理解
オブジェクト指向はわかるか:情報隠蔽の理解
– Javaは(私の)想像以上に修得は早い
UMLの必要性は理解できるか
– 書ける
– プログラミングと関連づけなければ「簡単
いう印象
– プログラミングとの関連づけは困難. . . 経験不
足. プログラムでの失敗をレポートで調査, 指摘
余談:パターンの重要性は, 成績と結びつけるこ
によって修得?
2002/2/7,8
www.s-lagoon.co.jp
8


(有)エス・ラグーン
企業内教育の場合
● 社会人＝10－20人
– 7H×3日
• オブジェクト指向概要
• UML
• 開発方法論
• 演習＋レビュー
実際には大学教育の1.5-2倍の内容
修得の鍵（プログラミング経験による裏付け！）
2002/2/7,8
www.s-lagoon.co.jp
9


(有)エス・ラグーン
社会人向け講習会の内容
● UML知りたい:
– 必要. 勉強する時間がないので集中的に.
● モデル化の実際を演習で
– レビューを含めた演習
– モデルの読み方を修得する
2002/2/7,8
www.s-lagoon.co.jp
10


(有)エス・ラグーン
結論
● 大学での教育には人数，時間による制約から，
UML修得には限界がある
● しかし
– これからのソフトウェア開発にはUMLは必須技術である
– 必須技術ならば，
• 研究：より円滑な意思疎通の方法
• 技術：開発の基礎知識として習得する必要
2002/2/7,8
www.s-lagoon.co.jp
11

UMLの実践教育

セミナー講師の立場から
桜井　麻里
mari-s@da2.so-net.ne.jp

2002年2月7日　SEA UML特別ワークショップ

大学での教育と違うところは

- 「実践」
- 教える相手:
  - ソフトウェア開発技術者、経験者
    - 現場で実際に使いたい。少なくとも必要性は理解している。
    - OO開発の経験はあるとは限らないが…。
    - 自分の開発経験や抱える課題に照らし合わせて考えることができる。
- 教育期間:
  - 短期集中 （1日、3日、せいぜい一週間）
    - 長期間プロジェクトを離れることは困難。
- 教育の形態:
  - トレーニング・セミナーが中心。
  - 長期的にはメンタリング、コンサルティング、OJT で。

定例トレーニング・セミナーの例

- （株）豆蔵の場合
  - UMLによるモデリング入門 (1日間)
    - 簡単な例題を使って、UML の主要な図を一通り書いてみること、OOAD開発演習に繋ぐ。
    - モデリングをやってみようという気になってもらう。
  - UMLによる実践OOAD開発演習 (3日間)
    - 3日間共通の問題を使って、分析から設計まで。
    - なるべく実践に近い例題を取り上げて、考えてもらう。
  - いずれの場合も、定員は20名。
  - 演習が中心 （後述）

何を教えているか

- UMLの表記
  - 主要な図の主要な表記のみ教える。
    - 表記の詳細は、各自必要に応じて UML の仕様を調べてもらえばよい。
    - コンポーネント図と配置図はテキストには載せているが、演習では使わない。使う必要がほとんど無いから。
- 開発プロセスや方法論
  - UML はプロセスや方法論からは独立であることを教える。
    - いつどのようなときにどの図を使うかなどは、UML では特に決まりは無いので、適宜選択して使ってもらえばよい旨、教える。
    - ただし、演習の都合上、そのときに使う図を指定している。
  - RUP や XP があることは教える。

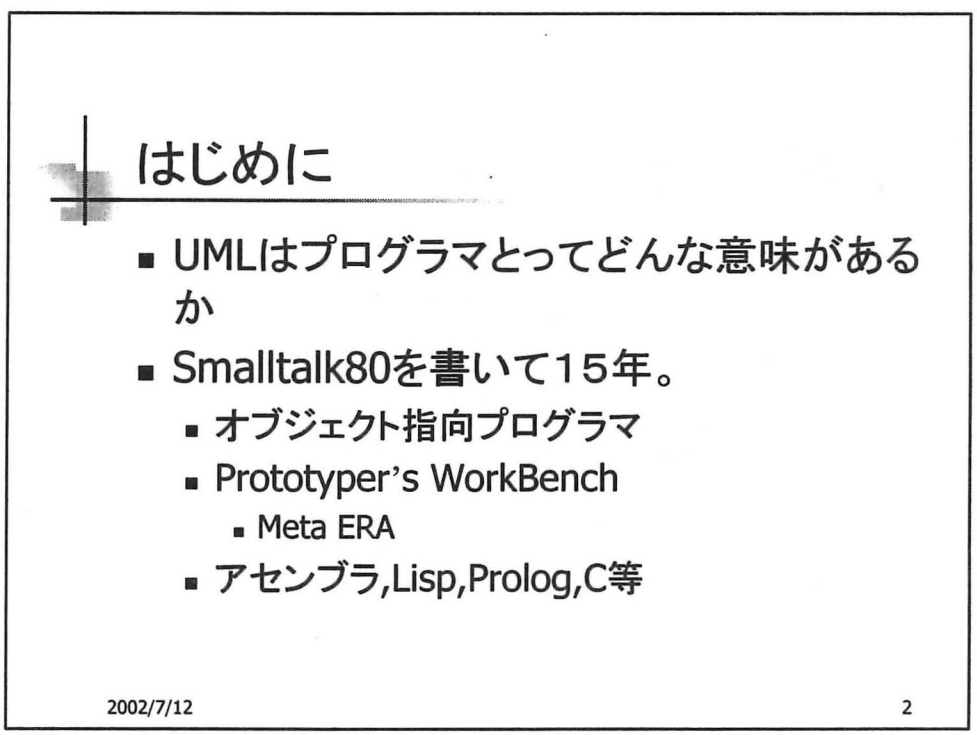
どのように教えているか

- 演習が中心
  - まず一人で考えてもらってから、グループ(4名程度)演習。
    - グループ討論にすることには良い面も悪い面もあり。
  - 演習には Rational Rose を使用。
    - 手書きのほうが良いという意見もある。
- レビューの時間を多めに取る
  - グループごとの発表、それに対する皆の意見。
    - みんなで考えてもらった成果を発表してもらい、UML がコミュニケーションの道具として使えることを実感してもらう。
  - 各自に気づいてもらうようにするため、講師は教えすぎないように心がけている。

課題

- 1 日や 3 日で全部わかるわけがない。
- UML は Language なのだから、使っていなければすぐに忘れる。
  - 現場に戻っても使えるようにする環境、文化が必要。
    - メンタリング、コンサルティング...
- 講師、メンターも育てないと...

プログラマから見たUML
東田 雅宏
(株)PFU
mashiro@k6.dion.ne.jp
2002/7/12
1


はじめに
UMLはプログラマとってどんな意味があるか
Smalltalk80を書いて15年。
オブジェクト指向プログラマ
Prototyper's WorkBench
Meta ERA
アセンブラ,Lisp,Prolog,C等
2002/7/12
2

プログラマ

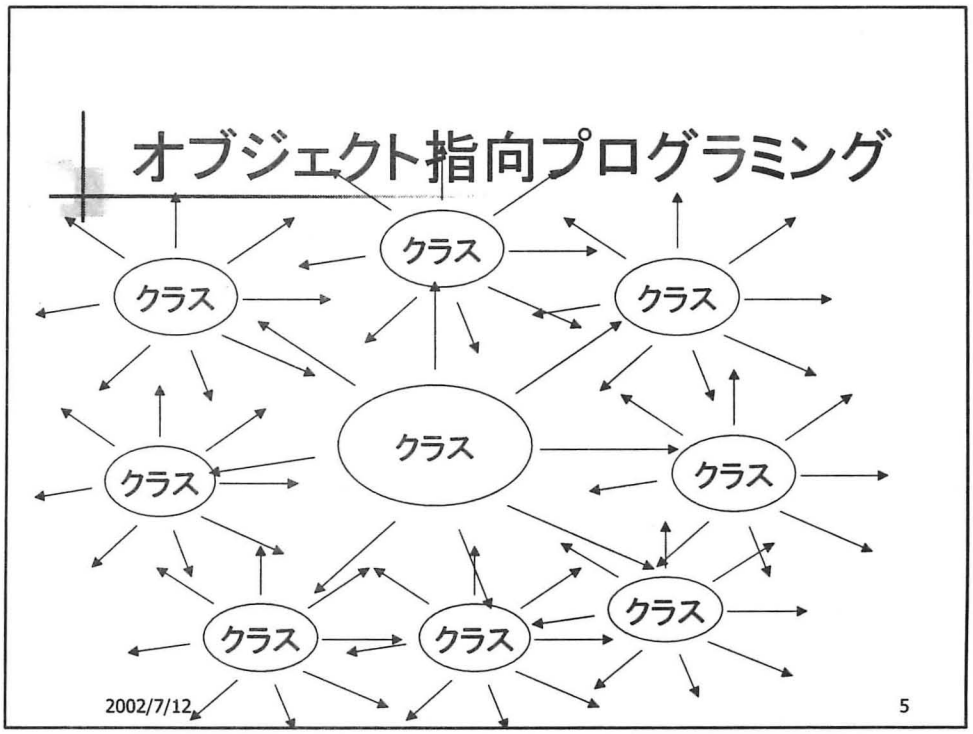
- プログラミング言語で思考する。
  - クラス名、メッセージ名
- ツールで動く
  - Smalltalk80
    - システムブラウザ
    - ワークスペース
- プログラムを動かすことに意義を感じる
- 文書書きは、二度手間、中途半端

2002/7/12 3

Smalltalkプログラミング

- とにかく、クラス定義
  - とにかく、インスタンス変数
- とにかく、メソッド定義
  - メッセージ・セレクタ
  - とにかく、コピー＆ペースト
- とにかく、doIt
- 何事もなく動くようになるまで？
- 冗長、重複を省く。やめる。

2002/7/12 4



UML

- 言語の意味はどこにあるのか
- ツールがない
  - 体得する手段がない。
    - 腑に落ちるまで
  - Smalltalkの開発環境を凌駕するものはない。
- 動かない
  - 動いてうれしいと思うか
- 文書書き

2002/7/12 6

創造について

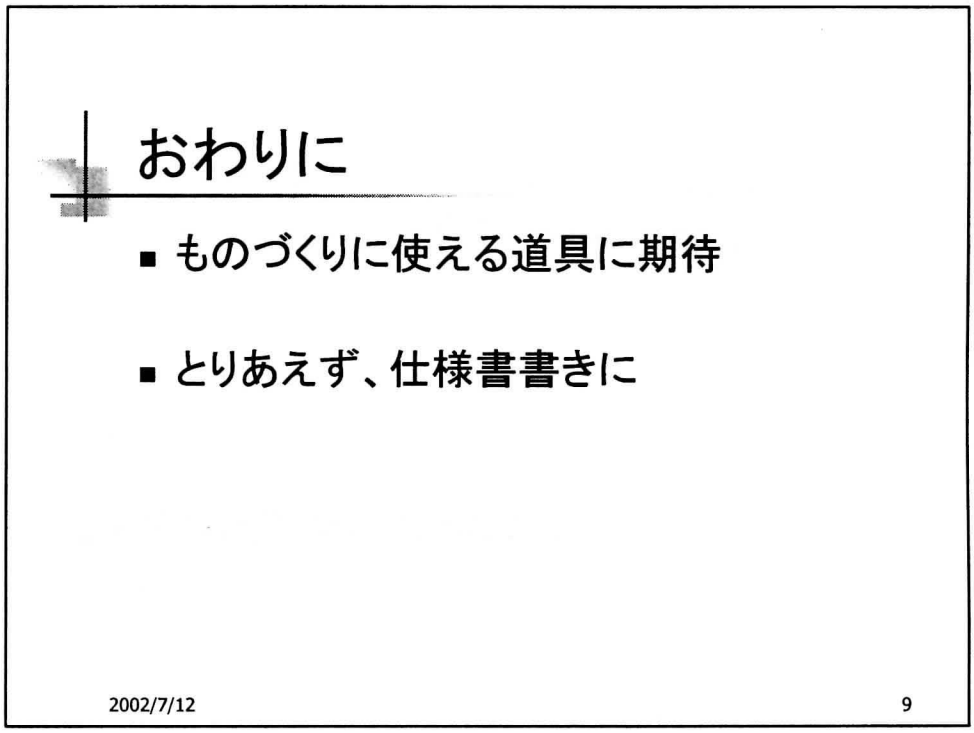
- オブジェクト指向のクラス設計
- デザイン・パターンとよばれるもの
- アプリケーション・フレームワーク

2002/7/12 7

UMLで設計書

- UMLはもりだくさん
  - 難しい上に、体得する手段が
- プログラムの一部をUMLにマッピング
- 仕様書の内容として

2002/7/12 8


おわりに
■ ものづくりに使える道具に期待
■ とりあえず、仕様書書きに
2002/7/12
9