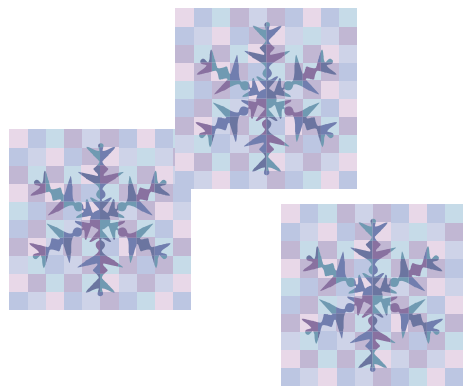
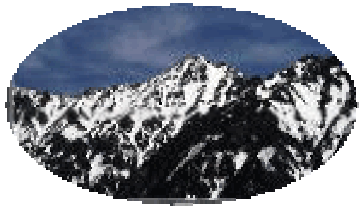
オブジェクト倶楽部
2003年度クリスマス・イベント

オブジェクト指向再入門
ー　仲間を増やそう編　ー

金澤 典子
2003年12月16日

I. なぜ、いま、オブジェクト指向？
II. 分析・設計・実装・UML
III. オブジェクトという共通概念
IV. モデリングという共通概念
V. 分析とは？　/ 設計とは？





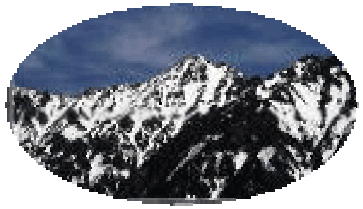
最初にお願いを

- オブジェクト指向による分析を考えていきます。

- 導入に関して一番の問題は組織ですが、組織の話はしません。
- XPやRUPにかかわる話もしません。

- どうぞ、みなさん、ご協力をください　m(__)m
  - 今年は、大変な年でした!!
  - OO関係、UML関係の本、HP、情報はたくさんありますが、
    オブジェクト指向を活用するためには高い敷居があるようです。
    それをどうにかできないでしょうか？

  - でも実りの多い年でもありました！

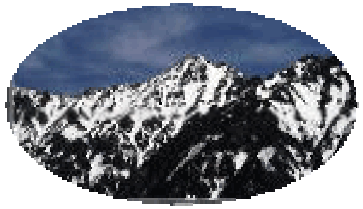


なぜ、いま、オブジェクト指向？

- **動機**（2003年採集）
  - 社内にオブジェクトチームはすでにあります。その一員として働くために基礎知識・技術の習得が必要です。
  - これからはオブジェクト指向での開発に取り組もうという社内チームが発足しました。私はその推進役です。
  - UMLも使って、さらにはCASEツールも使って、システム構築をしてきました。けれど、なんだか不安なのです。
  - 社内のメンバでJavaでの実装までできるようになりました。
    けれど、オブジェクト指向のメリットは、まったく現れないのです。

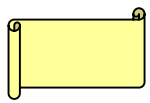
85％

  - Javaで構築したシステムのドキュメントをかいて整理しなくてはいけないのですが。。。
  - お客さまから、汎用的なAPIを設計してほしい、と言われました。UMLを使うとそれができるのだとお客さまが言っています。。。
  - これまでお付き合いしてきたIT会社から、これからはオブジェクト指向でシステムを構築するといってきました。提案書が読めません。
  - 会社のビジネスモデルをUMLで表現したいのです。
  - ハードウェアを含めたシステムのデザインにUMLを使いたい。

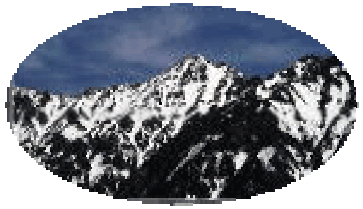


なぜ、いま、オブジェクト指向？

- 取得したい知識
  - オブジェクト指向のコトバ
    - すでに知っています、その知識を整理したい、コトバの心を知りたい
  - UMLの描き方
    - 基本的な要素、文法、使用方法
    - ダイアグラムの作成
  - オブジェクト指向による分析→設計→実装の一連の流れ
    - 基礎
    - メリットを生むには
  - 社内での教育、展開方法へのサジェスチョン

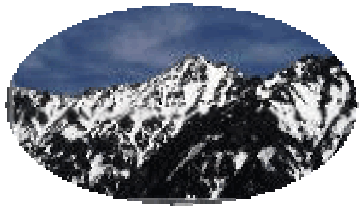


- 参考：大学でのオブジェクト指向教育の拡大
  - 経営学部
    - 大学を卒業した人の大半がなんらかの形（発注者、構築者など）でITに関わる。UML、Javaの知識は必須。IT業界の多くの人には身についていない経営の知識をUMLで表現する力をつけることで学生たちに自信をもたせてやりたい！
  - メディア学部
    - UMLは自己表現技法の1つ。学生の内的世界を豊かにしてあげたい。



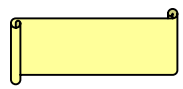
オブジェクト指向導入について

- 数年前までの導入方法
  - オブジェクト指向の基礎知識：
  - 演習を取り込んだ分析・設計法：
  - メンター制度を導入したシステム構築：
- 現在：
  - これまで開発サイトが自分たちのためにOOに取り組んできたが、いまは顧客からの要望にOO開発がある。必然的に短期間に適切な技術者を育成する必然性。

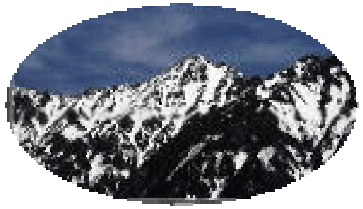


オブジェクト指向導入について(2)

- 現在
  - 実装技術、実装環境の整備がすすんだ。
  - OOプログラミング言語(Java、C++、C#etc)の知識のある人が増えた。
  - UMLを(言葉や効用だけであっても)知っている人が増えた。
  - 実装技術についての何らかの知識をもったうえで、ソフトウェア開発全般に、オブジェクト指向のアイディアを生かしたい！
    しかし、開発プロセスは。。。
    分析と設計は？　分析/設計と実装は？



- 情報が増大したのに、オブジェクト指向をベースにしたシステム構築を可能とするための「コスト」は軽減されない。

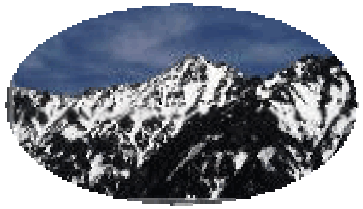


日本の事情

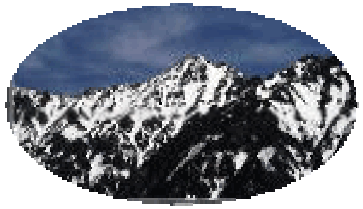
- 分業の徹底：
  - 機能仕様の作成までを実施し、実装は外注　(海外へも)
  - 下請け的な受注では採算がとれなくなってきた、上流工程からの受注をめざす
  - 提案型の受注をしていきたい
- ……

- ＜ニーズ＞
  オブジェクト指向プログラミングの経験、知識が
  十分とはいえなくても、
  “オブジェクト指向”に基づくUML仕様書が作成できるようになりたい/ならせたい。

困ったこと



- オブジェクト指向と私：
  - オブジェクト指向は、＜もの＞を整理する1つの方法
  - ソフトウェアを用いた解決策は、
    当然のことですが、すべてOOで　どうにかなるものではない。
    - →　ソフトウェアの基礎的な知識
    - →　ソフトウェアパターン (熟練者の知恵を文書化したもの)
    - →　マルチパラダイムデザイン
  - しかし、オブジェクト指向のアイディアによりソフトウェア世界は豊かになった
  - 楽しい！　　Aha!
- 私の不満：
  - 日本には優秀な技術者が多いのに、どうしていつも後追いなのか



分析・設計・実装・UML

情報が増大したのに「コスト」は軽減されないのはなぜ？

- 1995年ごろの考え方

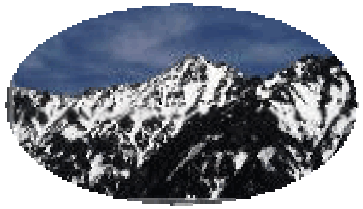
要求分析

- OO手法（OMT法、Coad-Yourdon法、OOSE法など）はセマンティックギャップを解消する

設計

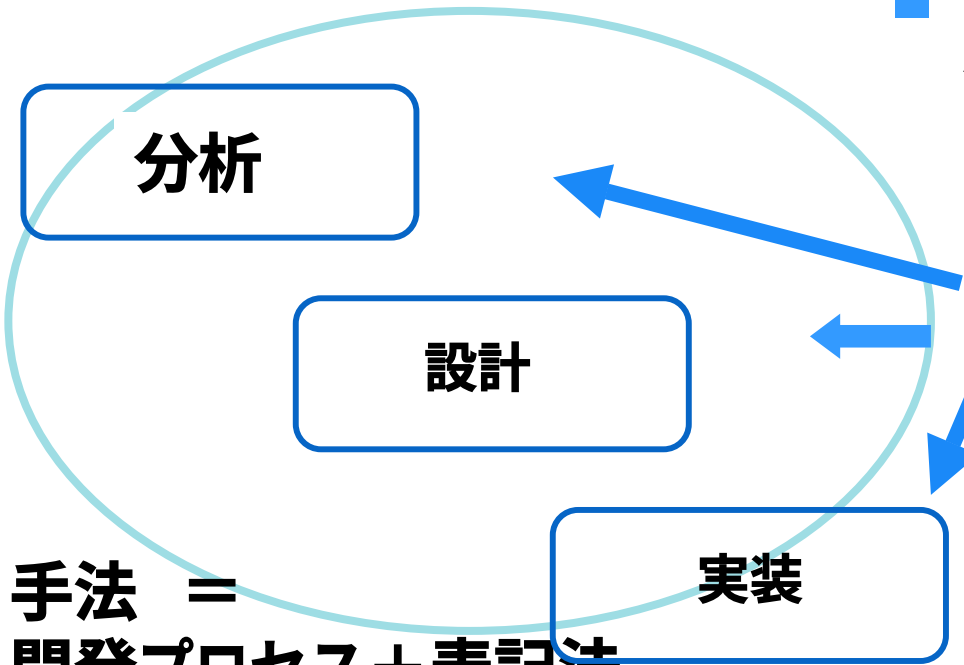
- 従来の方法では、サブシステム分割などの設計が個人の技量に依存する

実装



分析・設計・実装・UML

- 1995年ごろの考え方



- OO手法（OMT法、OOSE法など）はセマンティックギャップの解消する

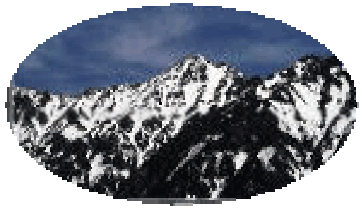
オブジェクトという共通概念

- 手法　＝
開発プロセス＋表記法

→ 開発プロセスにはその成果物が含まれる

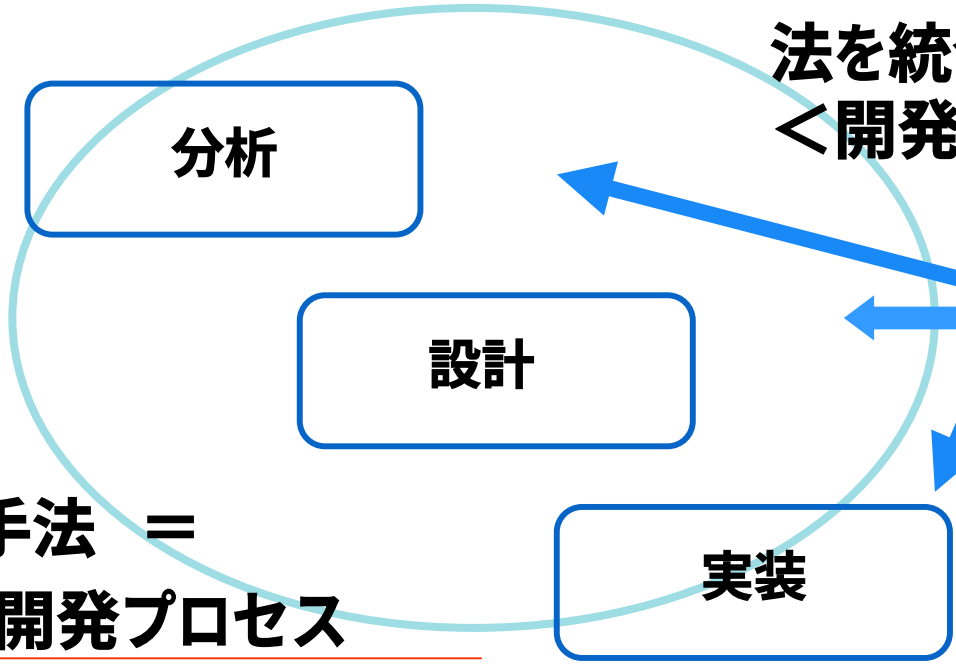
表記法：成果物を表現するために使用する

プログラミング言語の世界



分析・設計・実装・UML

- 2000年ごろの考え方
- UML：さまざまなOO手法の表記法を統合
  ＜開発プロセスと表記法の分離＞

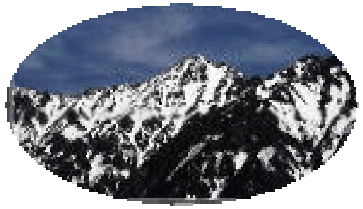


オブジェクトという共通概念

- 手法　＝
  開発プロセス
  ＋表記法

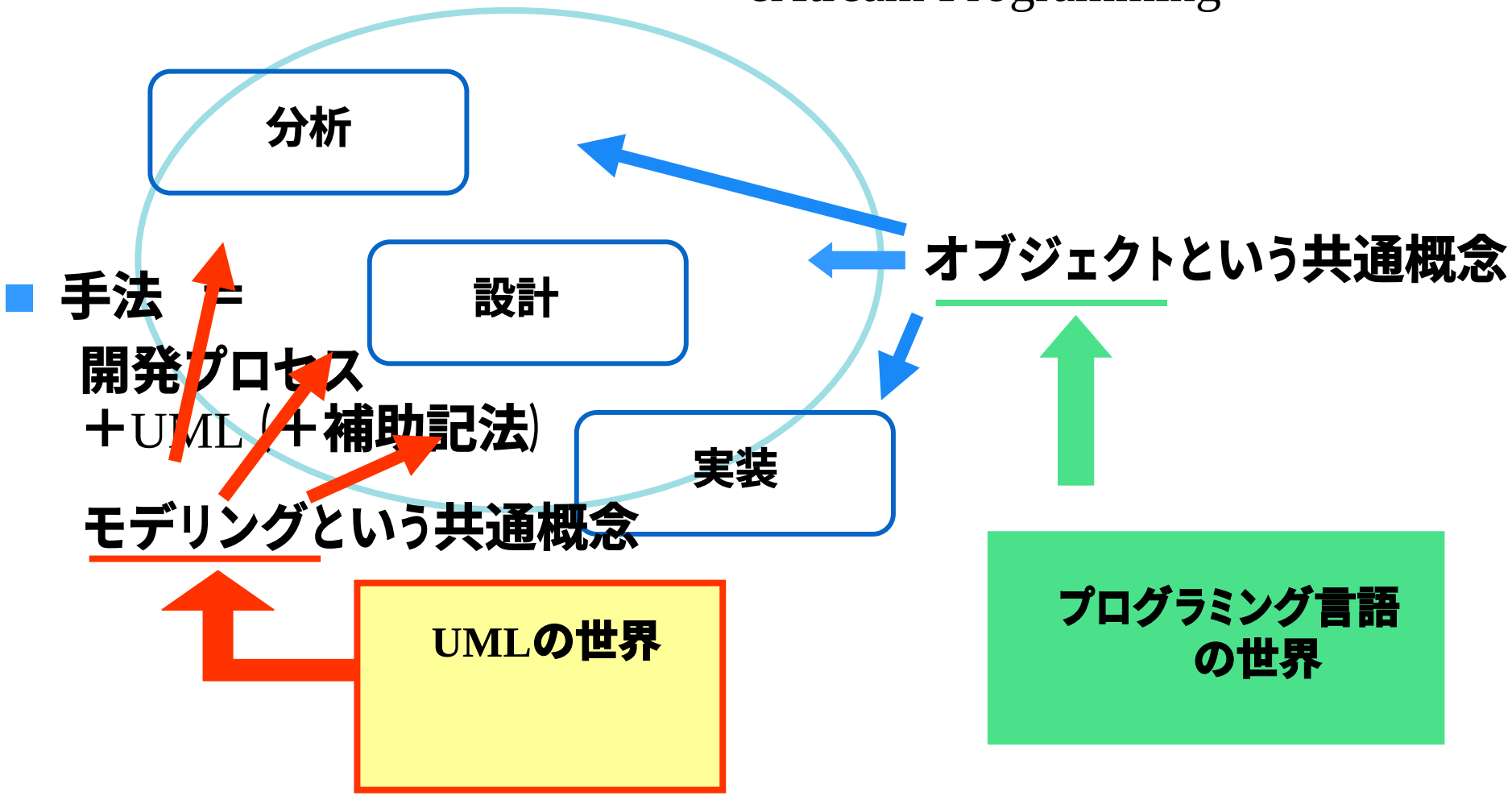
UMLの世界

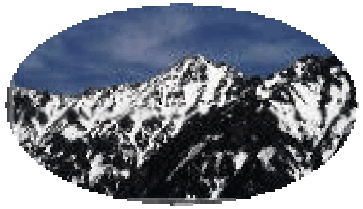
プログラミング言語
の世界



分析・設計・実装・UML

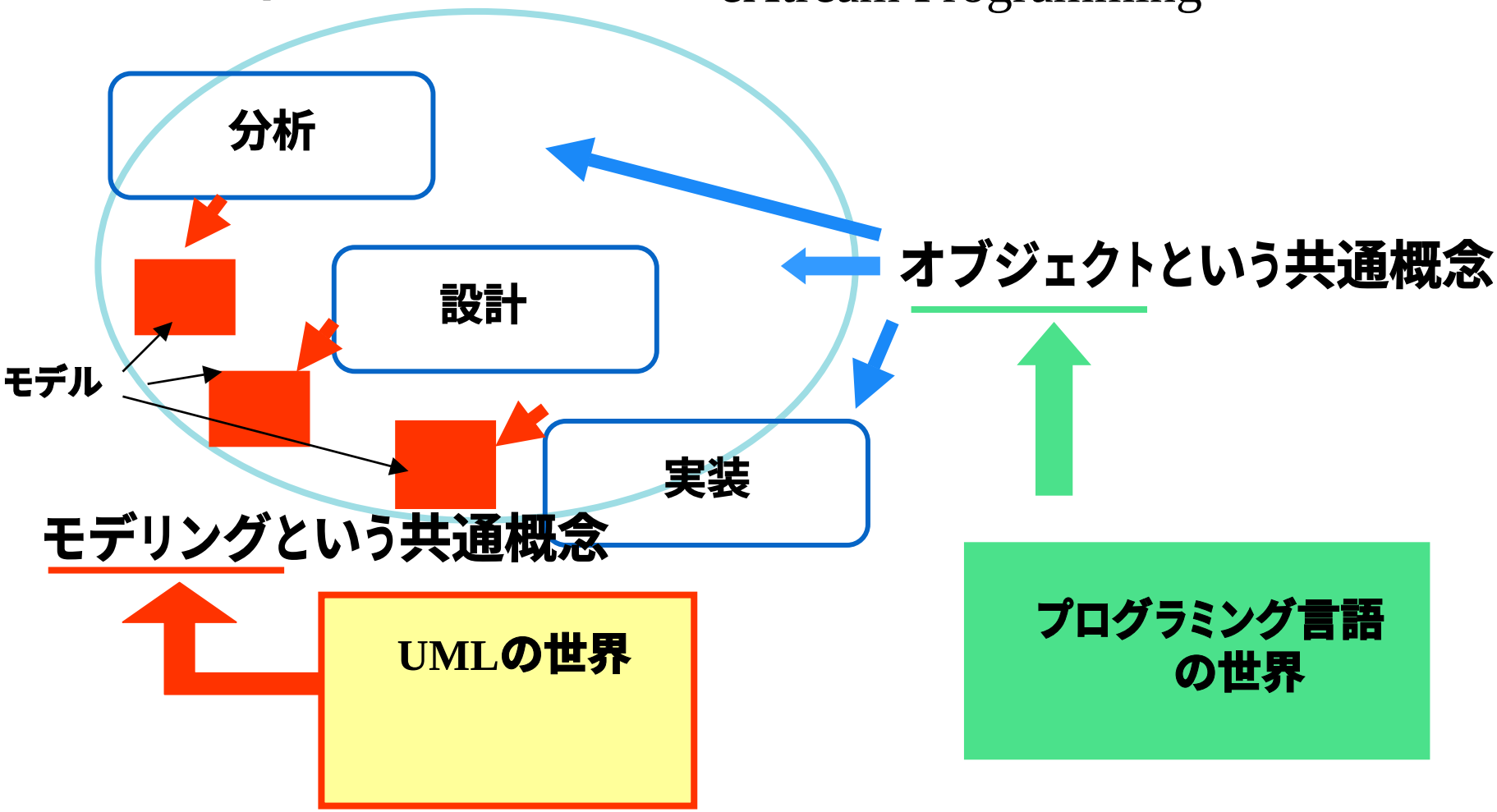
- 2000年＋α
- eXtream Programming

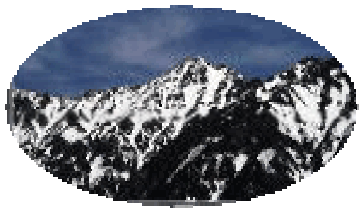




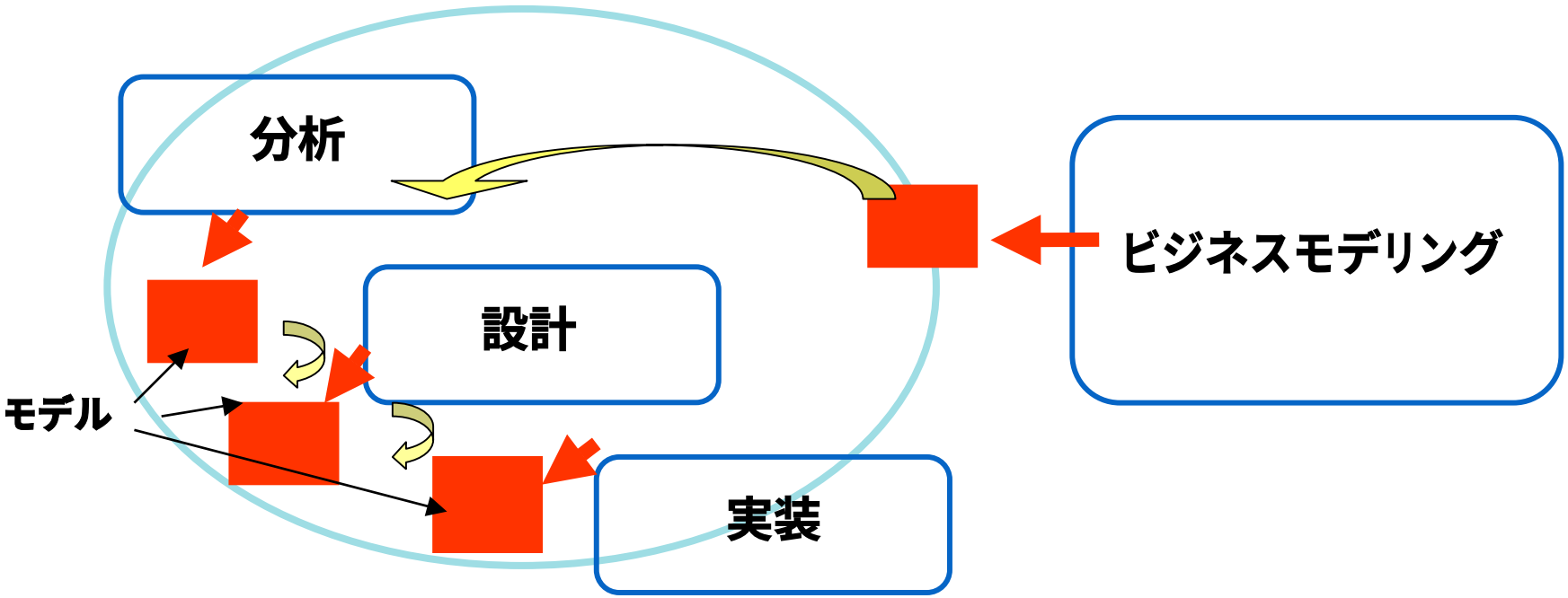
分析・設計・実装・UML

- 2000年＋α
  - eXtream Programming

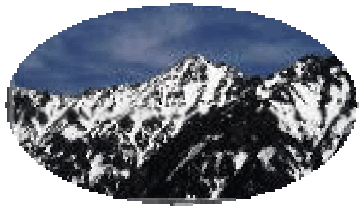




分析・設計・実装・UML

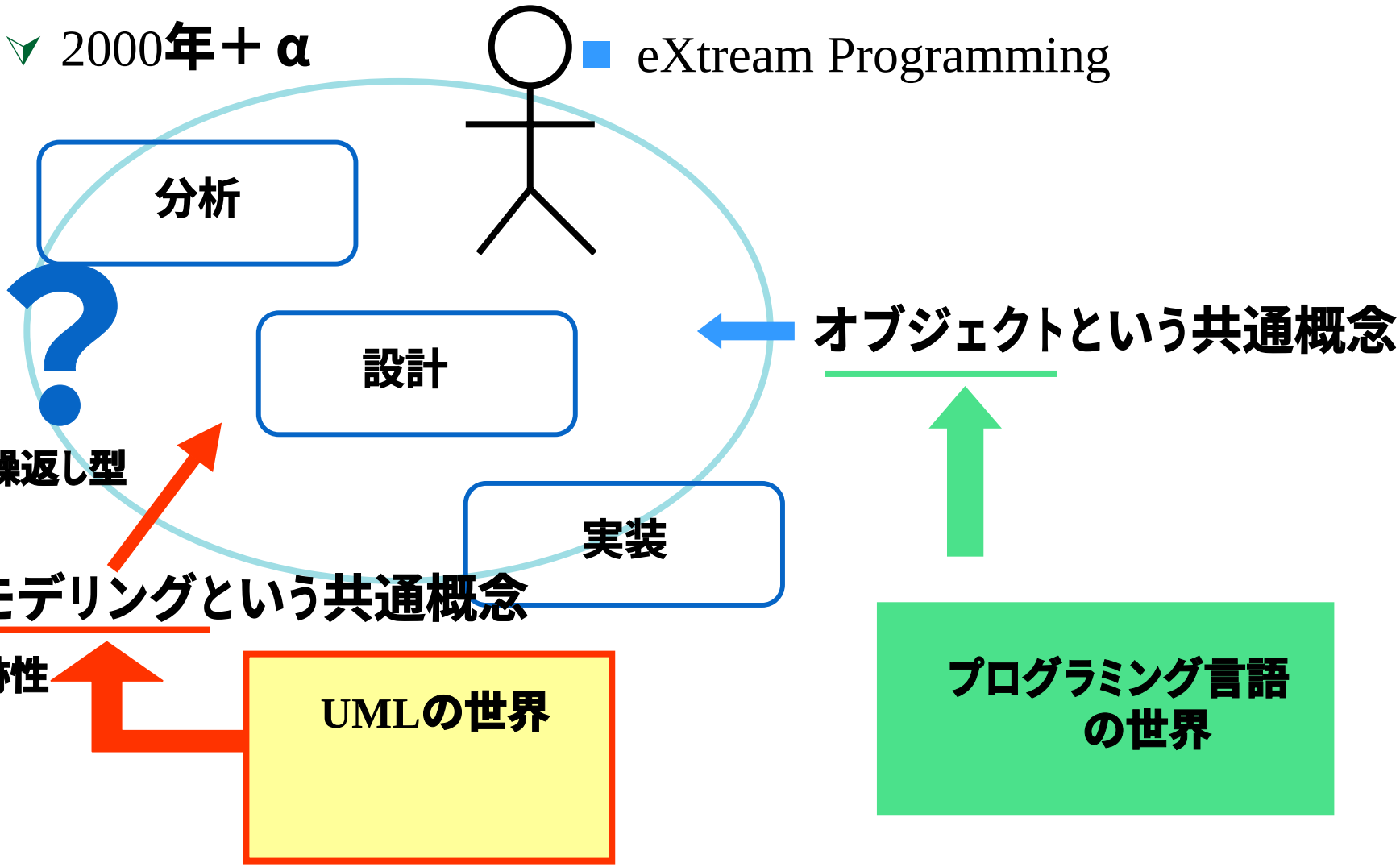


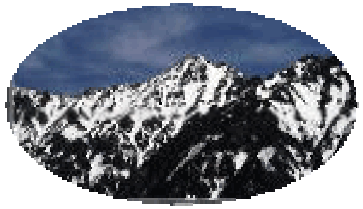
追跡性



分析・設計・実装・UML

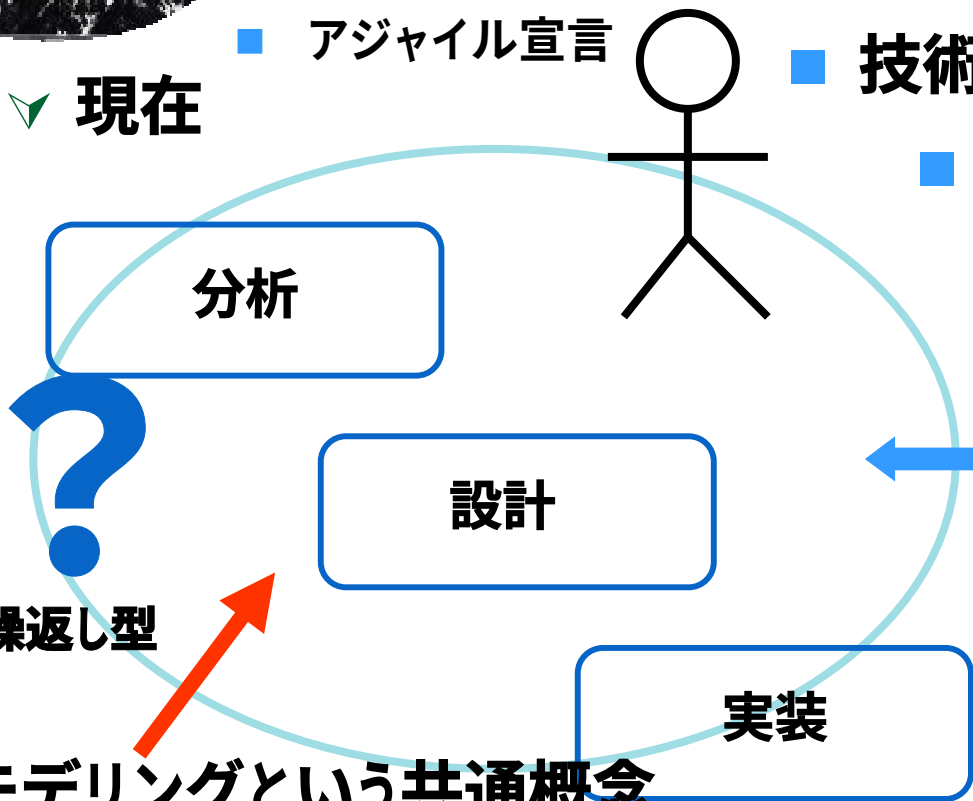
- 2000年＋α
- eXtream Programming





分析・設計・実装・UML

- 現在
  - アジャイル宣言
  - 技術者の資格化
  - SWEBOK (http://www.swebok.org/)



繰返し型

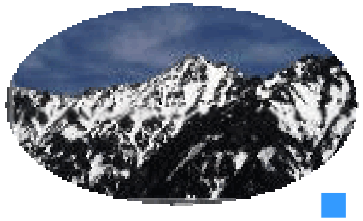
オブジェクトという共通概念

モデリングという共通概念

追跡性

UMLの世界

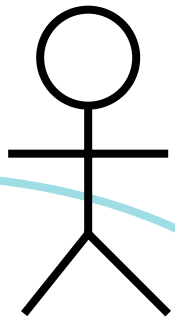
プログラミング言語の世界



分析・設計・実装・UML

- アジャイル宣言
- SWEBOK
- 技術者の資格化

現在



分析

設計

実装

繰返し型

- オブジェクトのアイディアを取り込んだ技術の進展

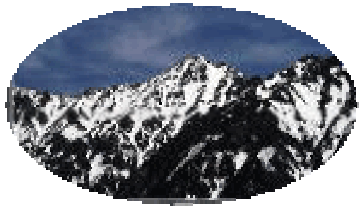
オブジェクトという共通概念

モデリングという共通概念

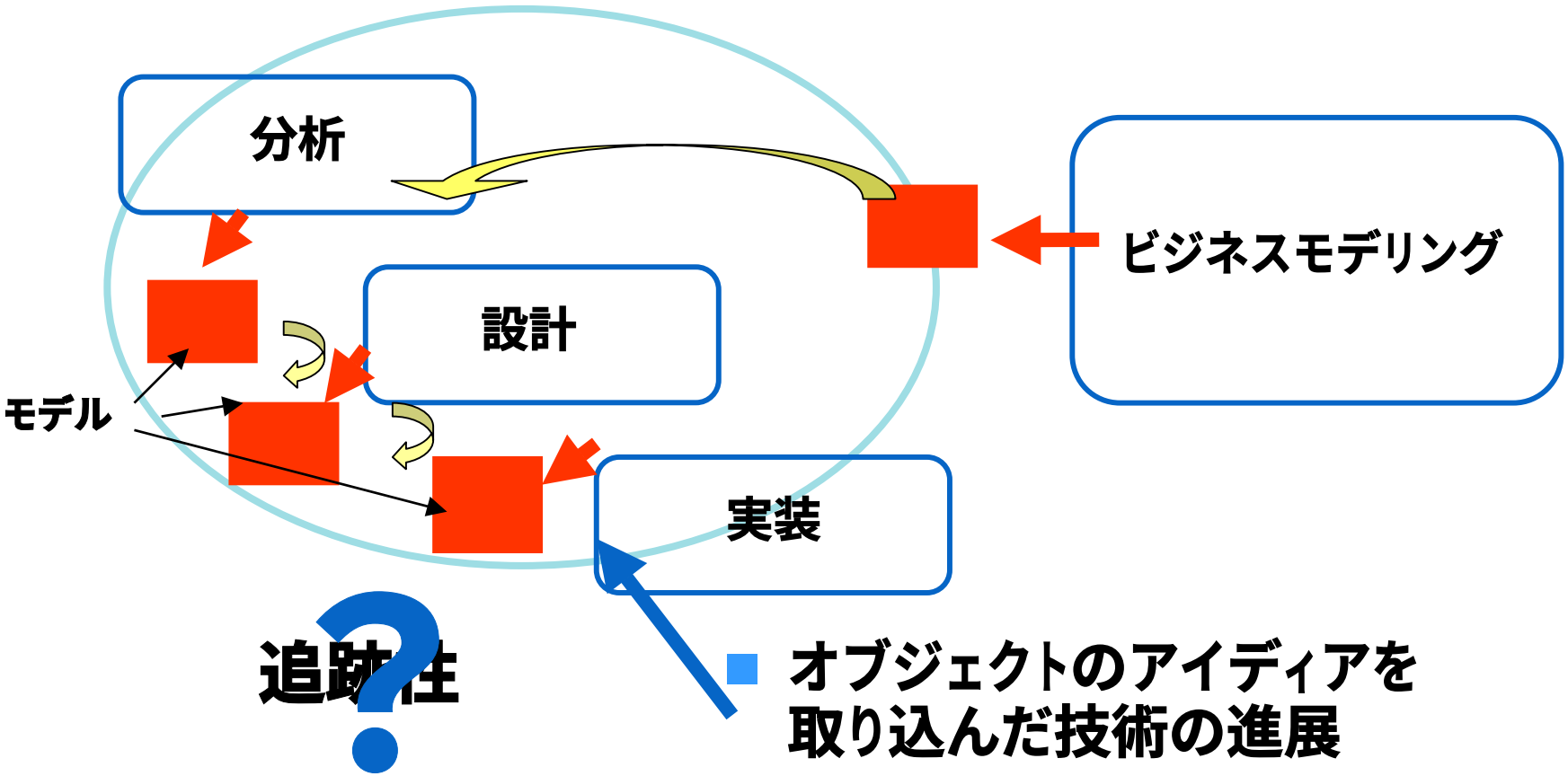
追跡性

UMLの世界

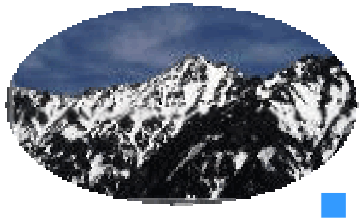
プログラミング言語の世界



分析・設計・実装・UML



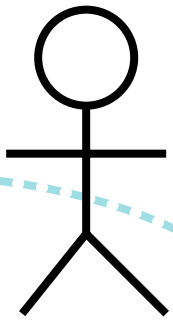
- オブジェクトのアイディアを取り込んだ技術の進展



分析・設計・実装・UML

- アジャイル宣言
- SWEBOK
- 技術者の資格化

▼ 現在



分析

設計

実装

繰返し型

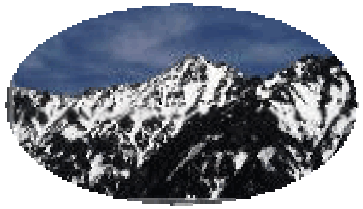
■ オブジェクトのアイディアを
取り込んだ技術の進展

オブジェクトという共通概念

モデリングという共通概念

UMLの世界

プログラミング言語
の世界

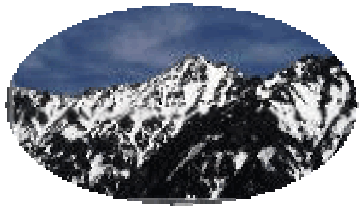


なぜ、いまオブジェクト指向？　再掲

- 取得したい知識
  - オブジェクト指向のコトバ
    - すでに知っています、その知識を整理したい、コトバの心を知りたい
  - UMLの描き方
    - 基本的な要素、文法、使用方法
    - ダイアグラムの作成
  - オブジェクト指向による分析→設計→実装の一連の流れ
    - 基礎

そのような状況の中で何が模範になるか？

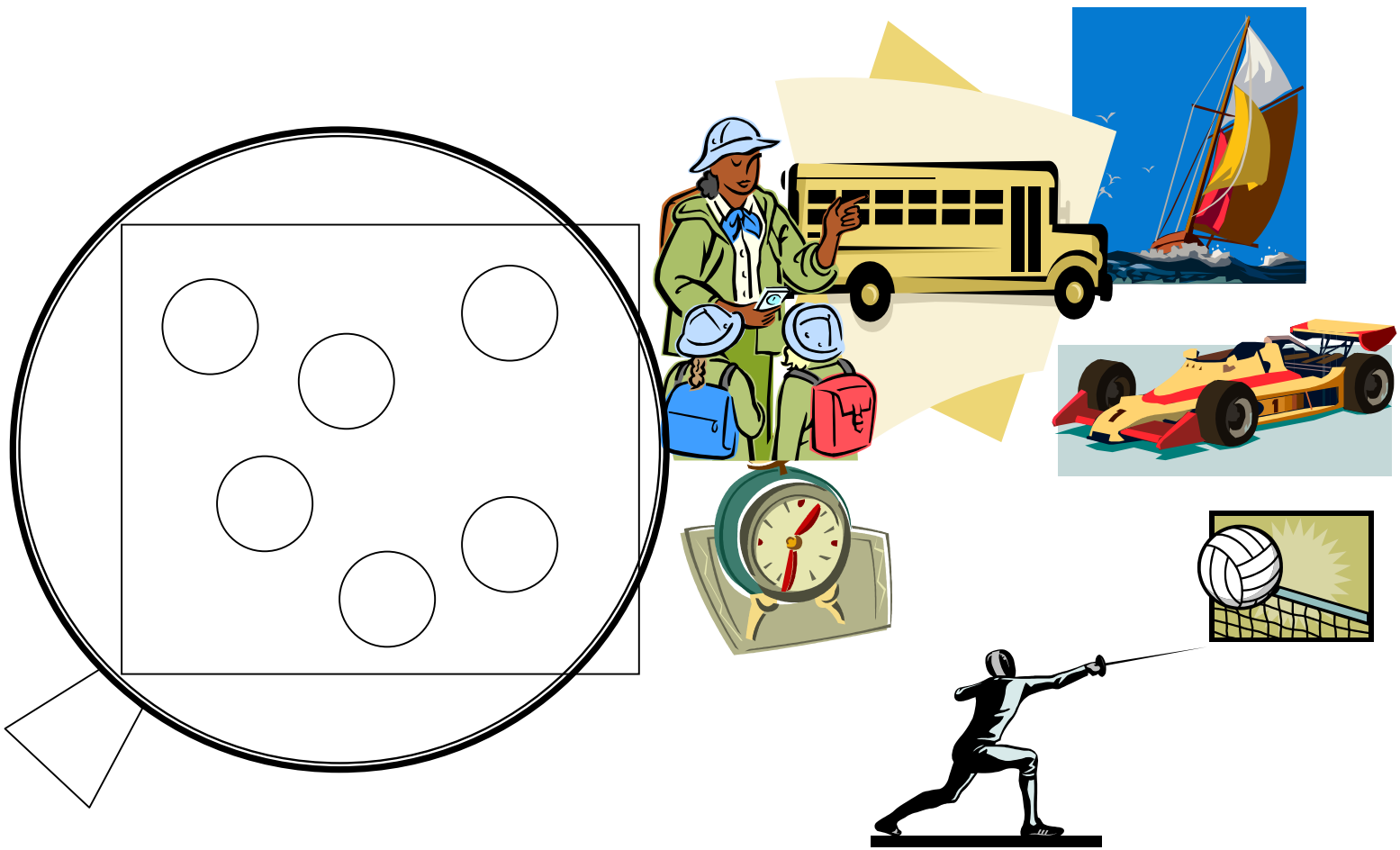
- 参考：大学でのオブジェクト指向教育の拡大
  - 経営学部
    - 大学を卒業した人の大半がなんらかの形（発注者、構築者など）でITに関わる。UML、Javaの知識は必須。IT業界の多くの人には身についていない経営の知識をUMLで表現する力をつけることで学生たちに自信をもたせてやりたい！
  - メディア学部
    - UMLは自己表現技法の1つ。学生の内的世界を豊かにしてあげたい。

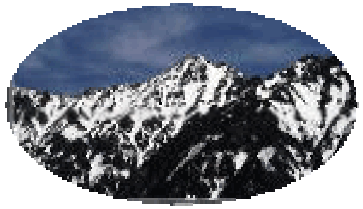


オブジェクトという共通概念

OOPに習熟していなくても、OOモデリングをするためには

オブジェクト指向プログラミング言語の世界



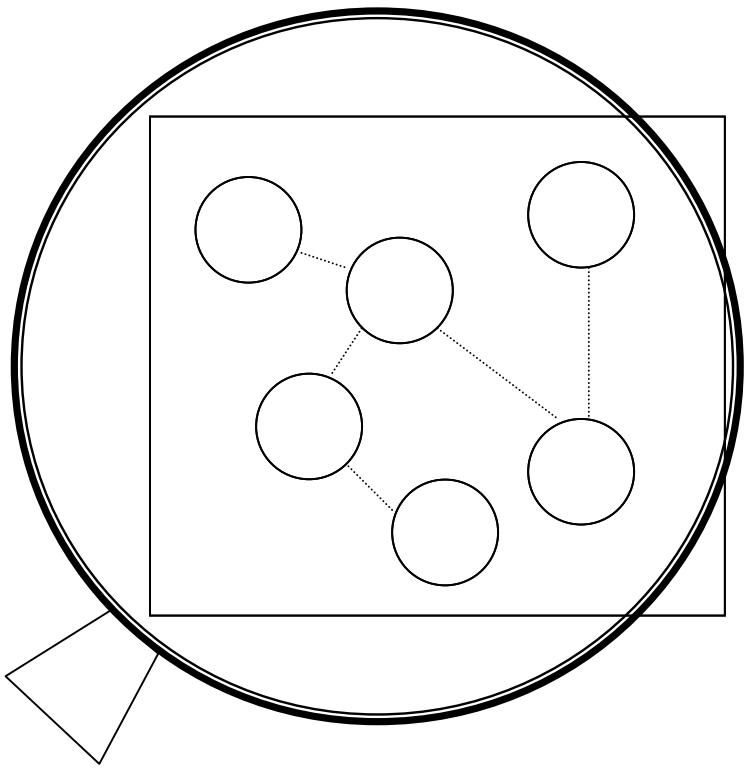


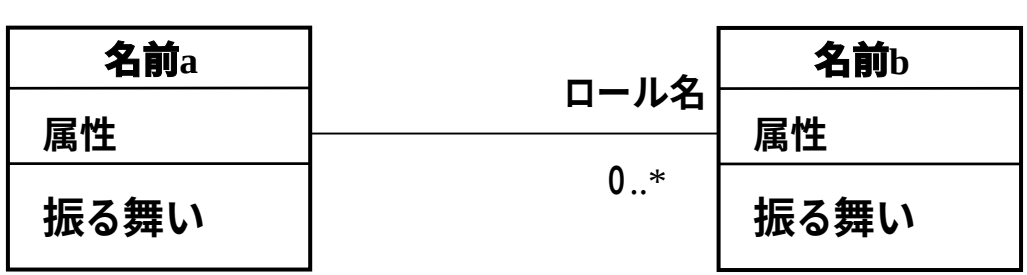
オブジェクトという共通概念

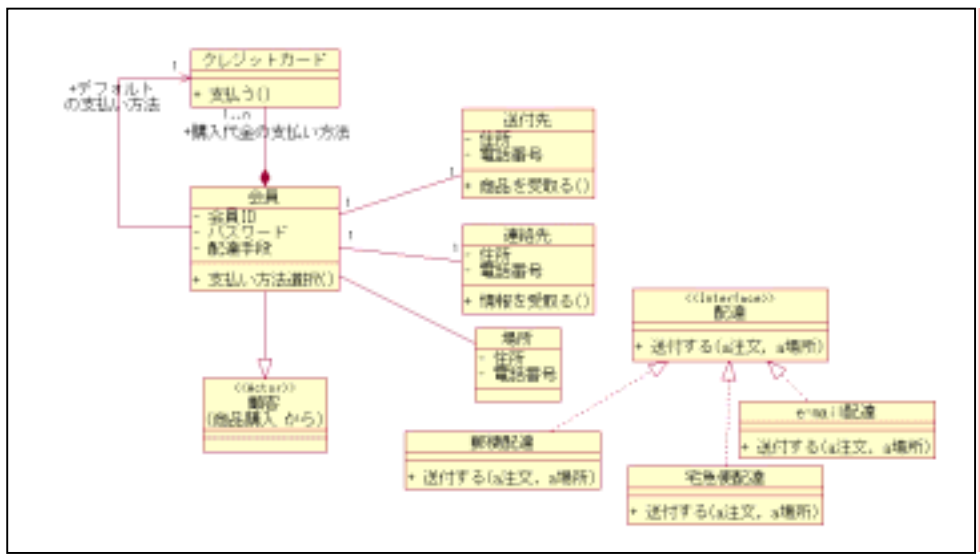
オブジェクト指向プログラミング言語の世界

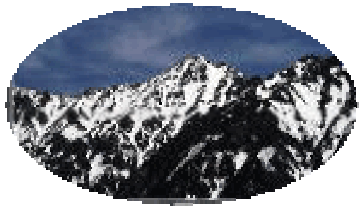
ソフトウェア工学のアイディア

カプセル化



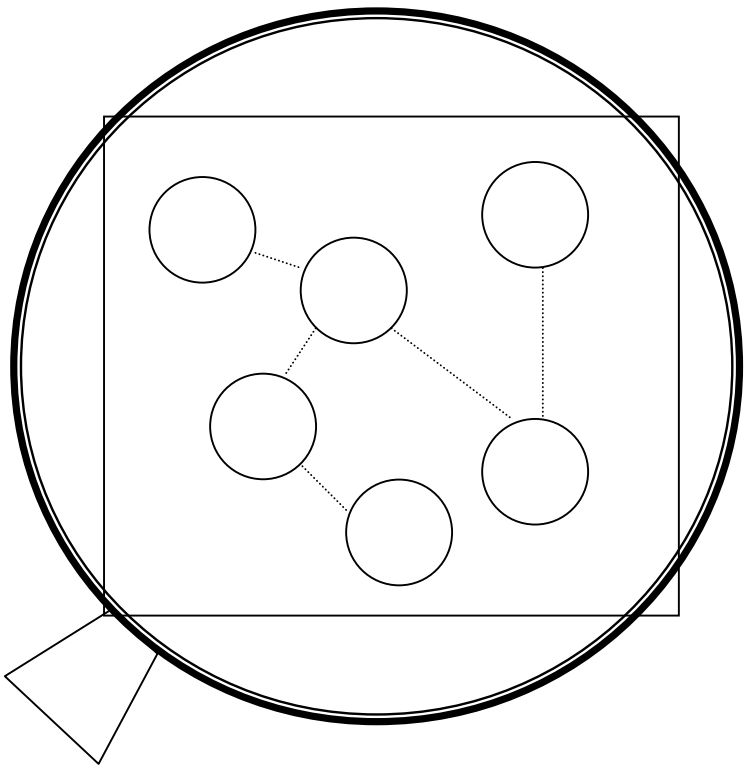


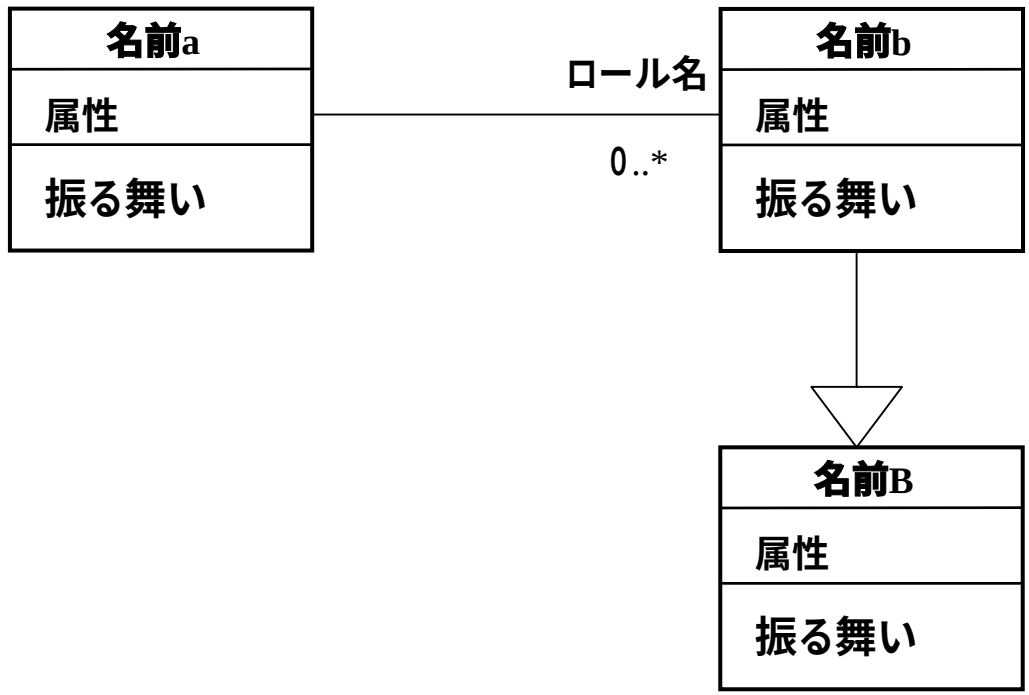


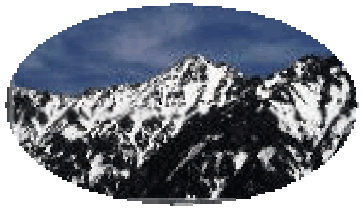


オブジェクトという共通概念

オブジェクト指向プログラミング言語の世界







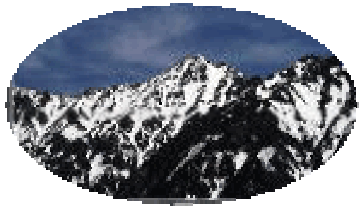
継承の種類

- D. G. Firesmith et. Al., “Dictionary of Object Technology”, 1995, SIGS Books ISBN 1-884842-09-7
- class inheritance, class-instance inheritance, type inheritance
- dynamic inheritance, static inheritance
- implementation inheritance, interface inheritance (specialization inheritance, strict inheritance)
- single inheritance, multiple inheritance (mixin inheritance, repeated inheritance(….))
- private inheritance, protected inheritance, public inheritance
- object-object inheritance
- inverted inheritance=selective inheritance
- event inheritance ｏｏｏ

「分析」では

- **上位概念**
-
- ｏｏｏ

さらに考察が必要な分野では？
ソフトウェアパターンの可能性

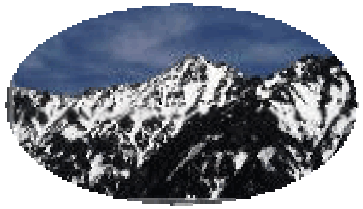


名前の重要性

- プログラミングでは、
  a. 名前空間
  b. コーディングガイドなどで命名規則
  c. さらに、スタイルのパターン
     瀬戸川教彦,「プログラミングのパターン：スタイルの紹介」,
     2003.12 (http://patterns-wg.fuka.info.waseda.ac.jp/study/1st.html)
     - INTENTION REVEALING METHOD NAME by Jeff Langr.

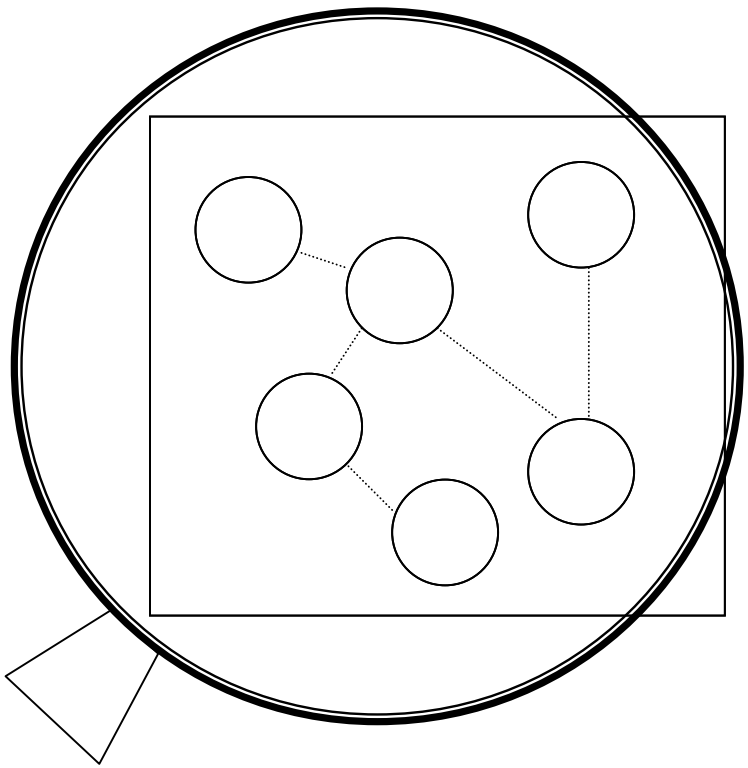
- 「分析」では、
  a. ……. 経験的には何かあるかもしれない
  b. 振る舞いの名前は（目的語＋）動詞
  c. クライアントから見た視点で、振る舞いの名前をつける
     ？？
     . 「クラス名にXXX情報はいけない」？？
     - ooo

ソフトウェアパターン の可能性



オブジェクトという共通概念

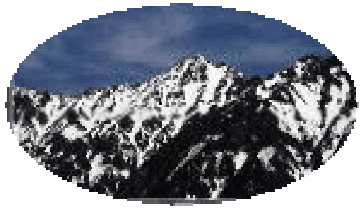
オブジェクト指向プログラミング言語の世界



＋

- **メッセージパッシング**
  - **オブジェクト間のコラボレーション**
- **状態マシン**
  - **オブジェクトは状態をもつ**

UMLの世界

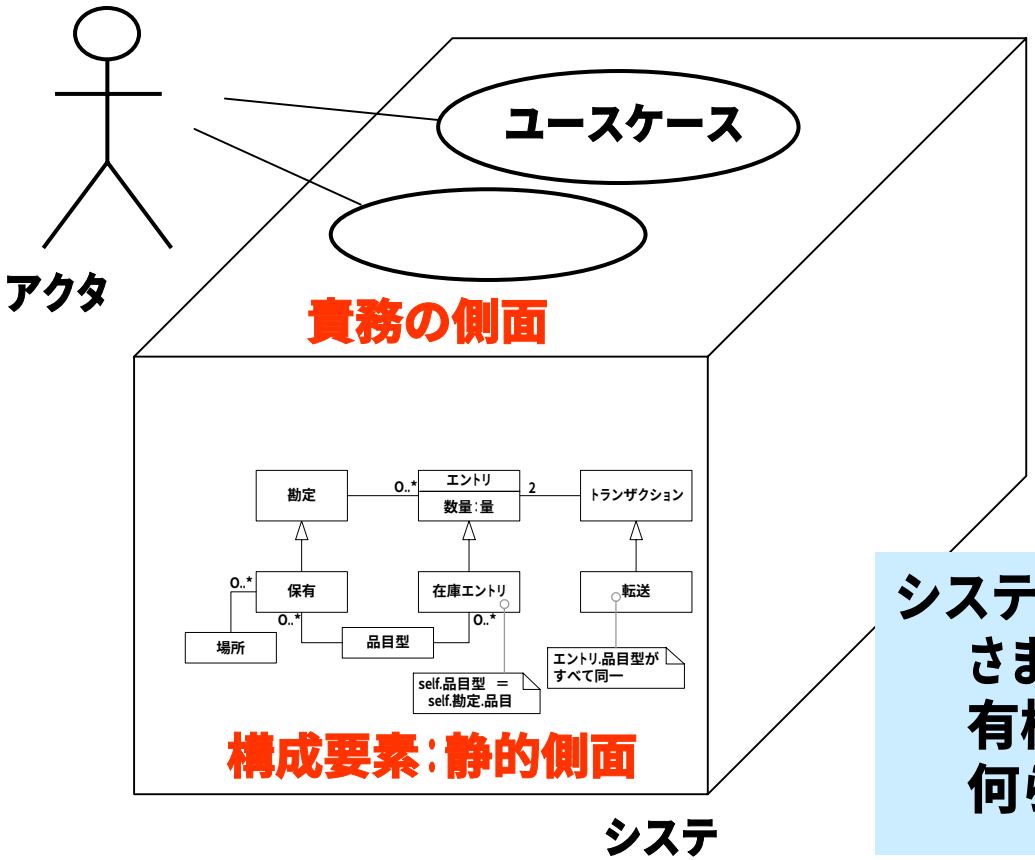


モデリングという共通概念

UMLによるモデリングの特徴

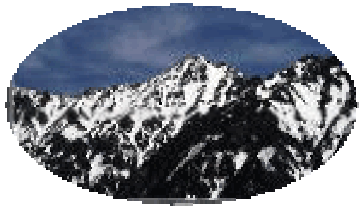
UMLの世界

UML：視覚化、コミュニケーションツール、仕様記述



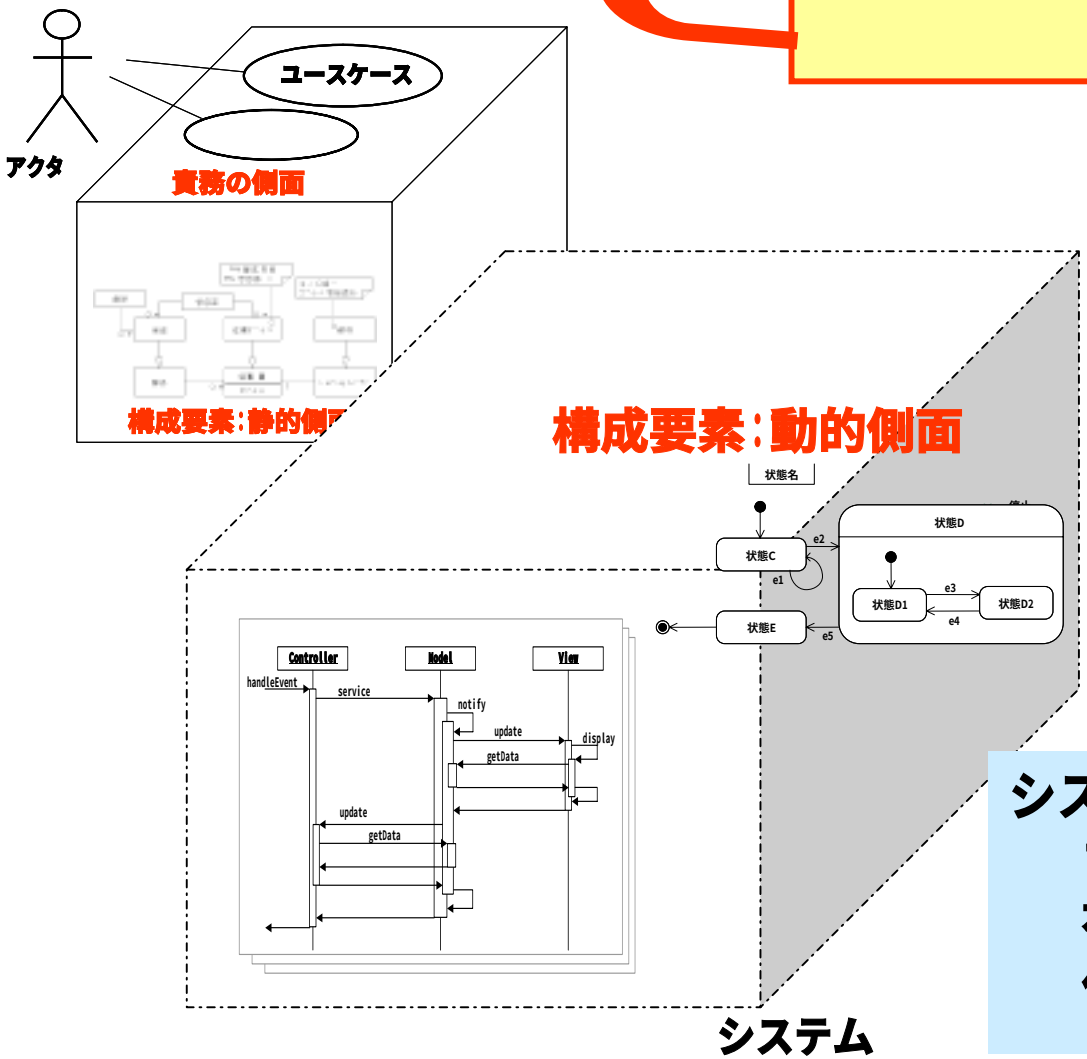
- システムの論理構造と物理構造
- システムの静的構造と動的構造

システムとは、
さまざまなタイプの構成要素が
有機的に協調しあって
何らかの機能をはたすもの



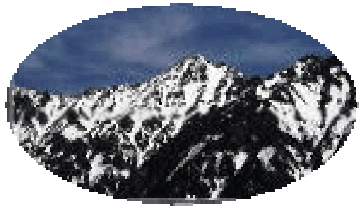
モデリングという共通概念

UMLの世界



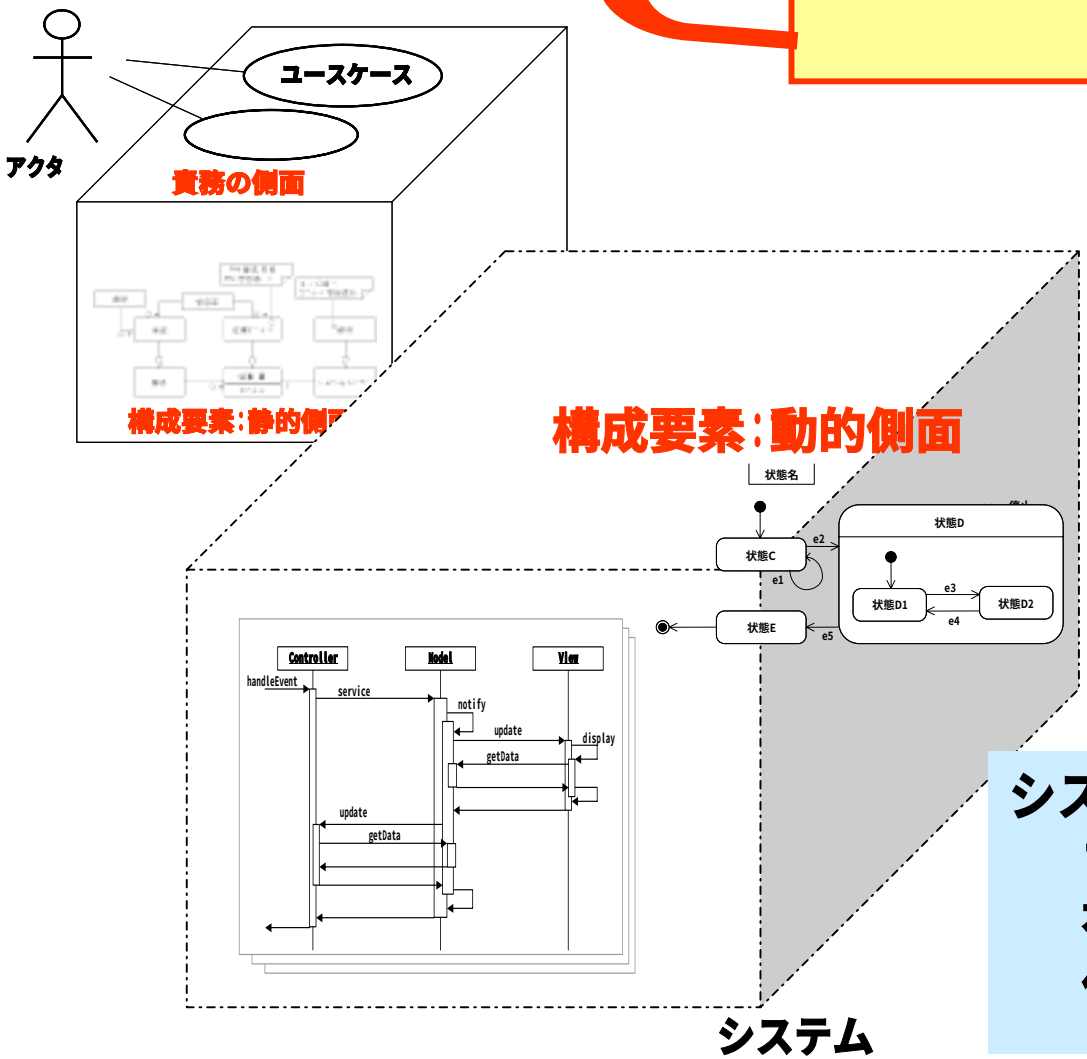
- システムの論理構造と物理構造
- システムの静的構造と動的構造

システムとは、
さまざまなタイプの構成要素が
有機的に協調しあって
何らかの機能をはたすもの



モデリングという共通概念

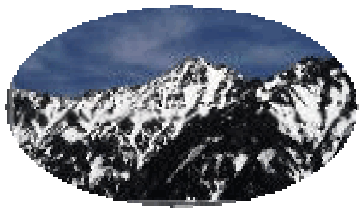
UMLの世界



モデルとは、
　関心対象に焦点をあわせて、
　その仕組みを
　表現したもの

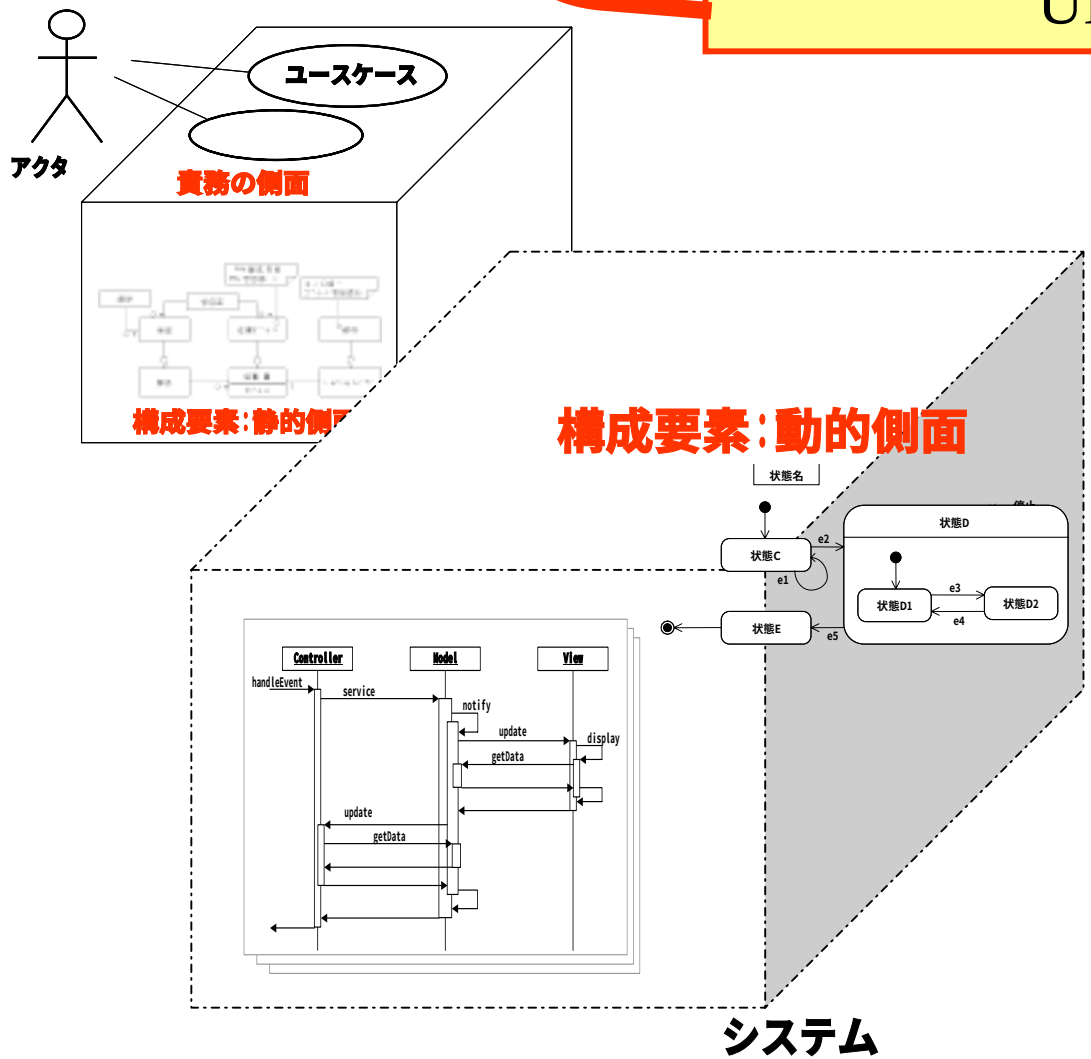
UMLは
　モデルの整合性を保証する

システムとは、
　さまざまなタイプの構成要素が
　有機的に協調しあって
　何らかの機能をはたすもの



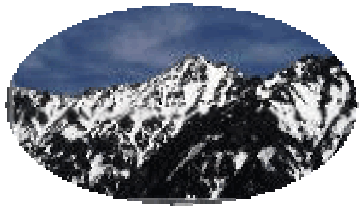
モデリングという共通概念

UMLの世界



一方で、

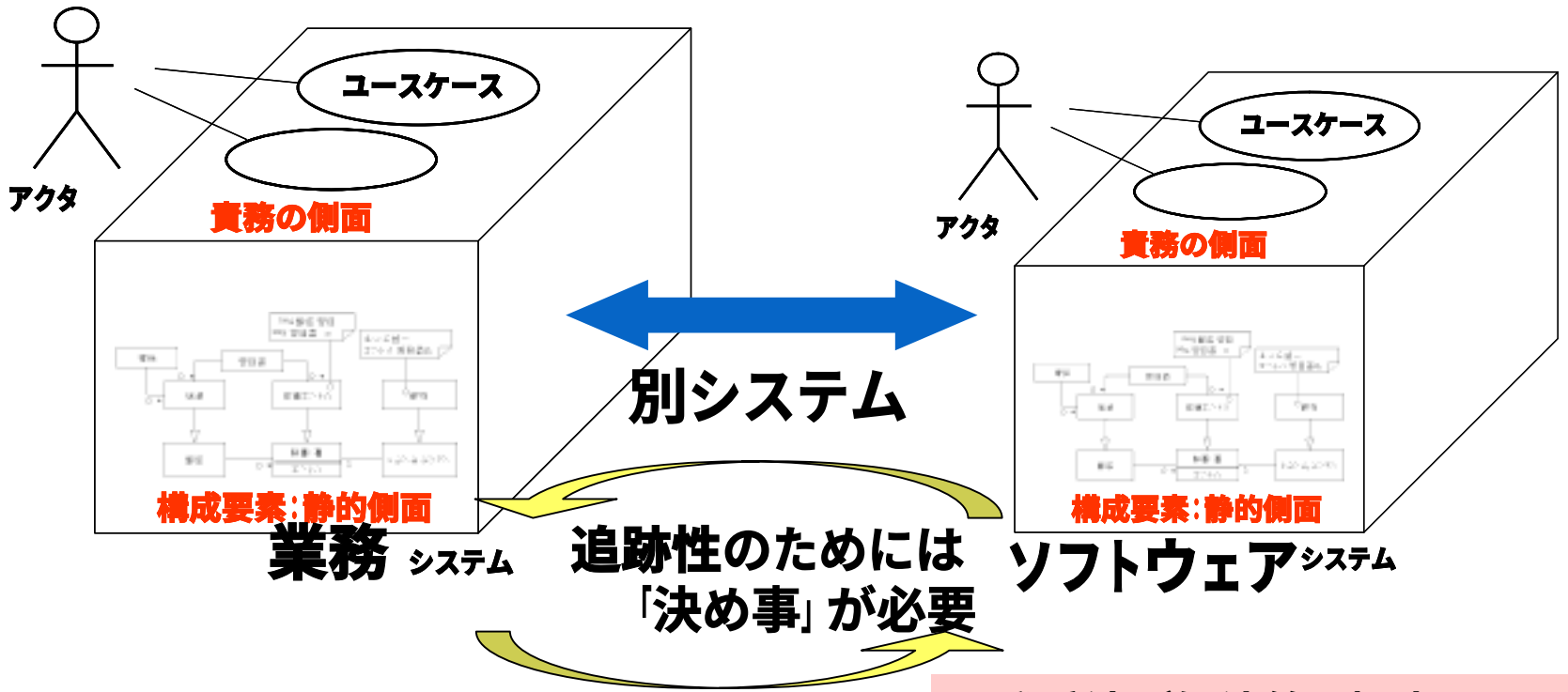
- 静的モデル、動的モデル、機能モデル（OMT法）. /ユースケースモデル（RUP）
- 1つのタイプのダイアグラムを単独で使用することもできる
- 参考）個々のダイアグラムは、構造化手法で提案されたもの、あるいは、その拡張
- XML技術とクラス図



モデリングという共通概念

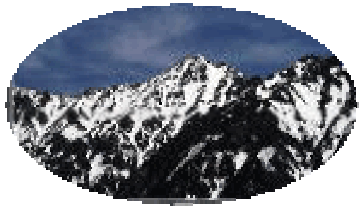
UMLの世界

- ビジネスも、ハードウェアも、ソフトウェアシステムも、システム



開発手法が解決策を提案していた

ソフトウェアパターン の可能性

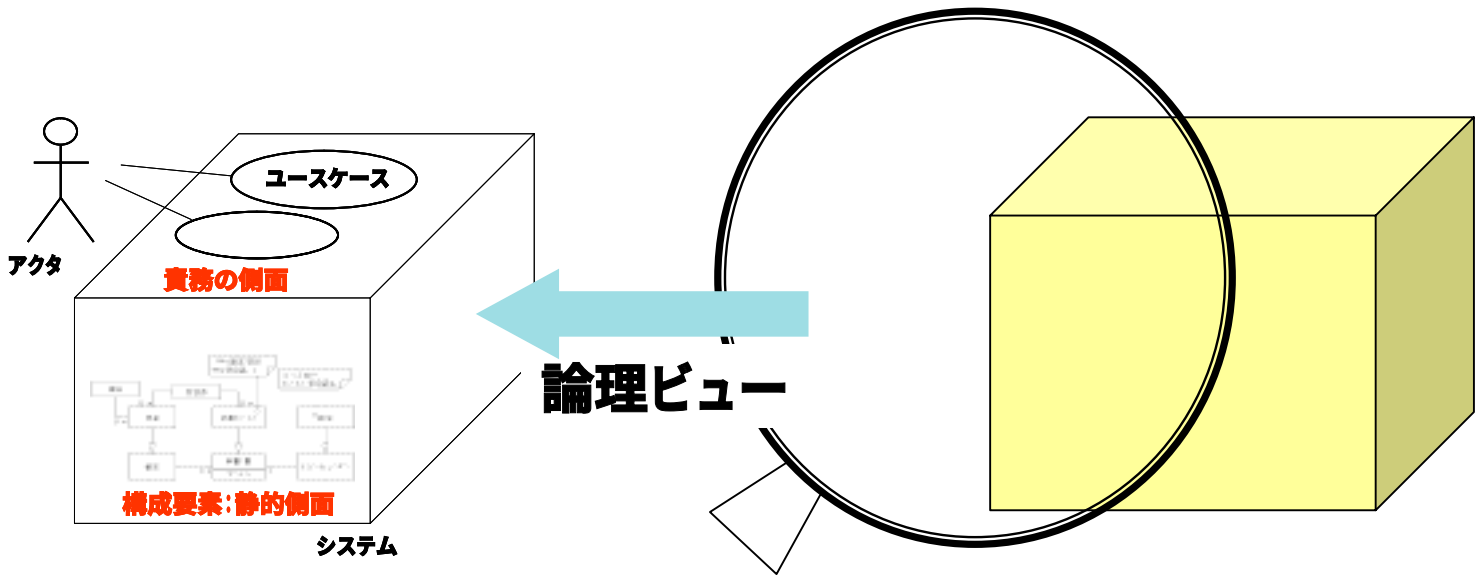


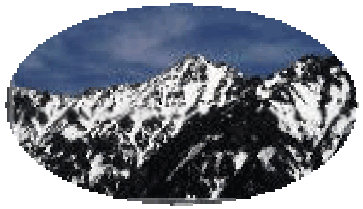
モデリングという共通概念

UMLの世界

- モデリングのビュー
  - “4+1”アーキテクチャビュー

    ユースケースビュー
    論理ビュー、物理ビュー、プロセスビュー、開発ビュー (RUP)



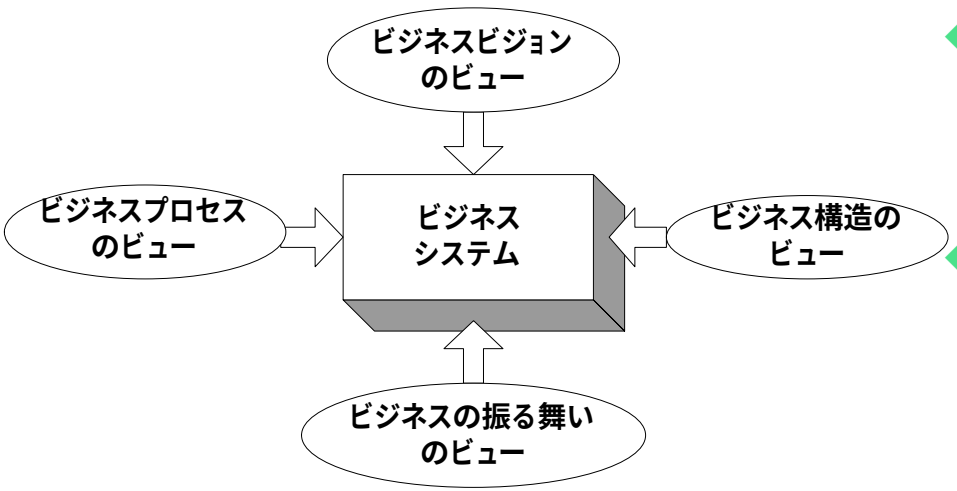


モデリングという共通概念

UMLの世界

ビジネスモデリングのビュー（Eriksson／Penker法）

Hans-Erik Eriksson and Magnus Penker, Business Modeling With UML: Business Patterns at Work, John Wiley & Sons, January 18, 2000



ビジネスビジョンの観点

- ミッションステートメント，TOWSマトリックス
- 「ゴール図」（オブジェクト図で表現）
- 「コンセプト図」（クラス図で表現）

ビジネスプロセスの観点

- 「プロセス図」（アクティビティ図のステレオタイプ）
- 「アセンブリライン図」（アクティビティ図ステレオタイプ）

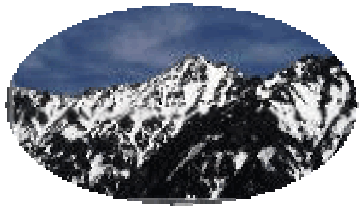
ビジネス構造の観点

「リソースモデリング」，「情報モデリング」，「組織モデリング」を行って，リソース間の静的構造を明らかにする（クラス図を用いる）．
ビジネスルールの記述にはOCLを活用する．

ビジネスの振る舞いの観点

ビジネスの動的側面をモデリングする．
シーケンス図やコラボレーション図，もしくはステートチャート図を用いる．

事業模型倶楽部（http://www.bm-study.org/）
Eriksson／Penker法によるモデル
Marshall に基づくプロセス



モデリングという共通概念

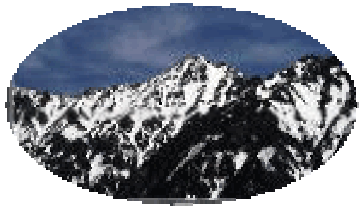
UMLの世界

- UMLの進展：ソフトウェアの実装技術を表現できるように
  ex.「パターン」という構成要素

  - プログラミングコードとUMLの関係
    - プログラミングコードで表現されたクラス
      vs
      UMLで表現されたクラス
    - わかりやすさは？
    - 正確さは？

名前、属性、振舞い(操作)をみてとれる表記
(メソッドは見えない)

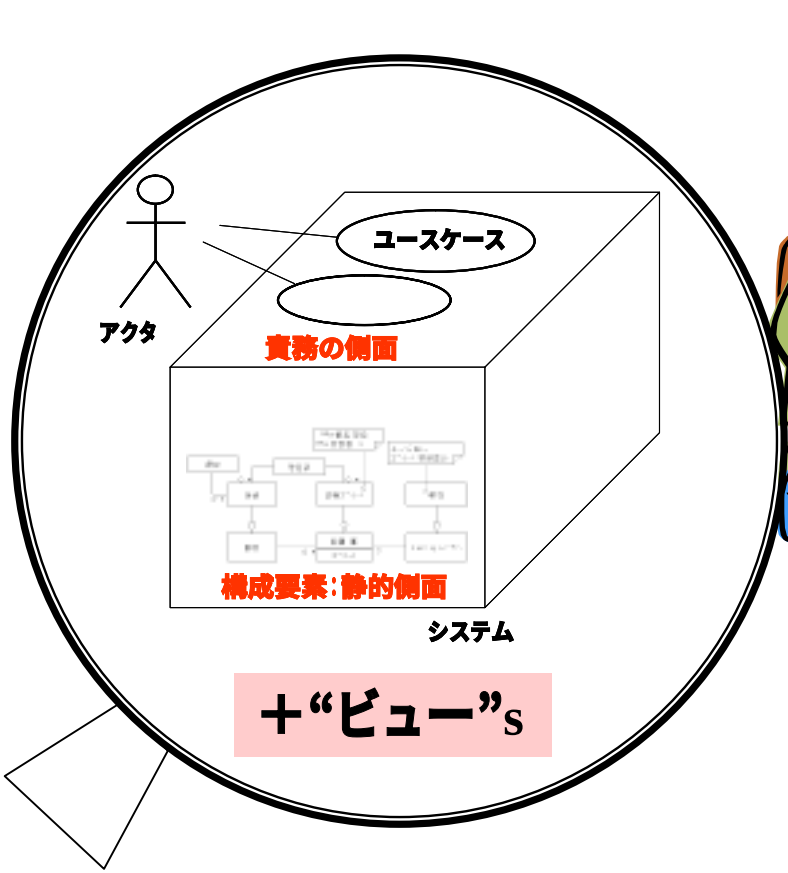
Point
int x int y
boolean equals(Object obj) Point getLocation() double getX() double getY() void move(int x, int y) void setLocation(int x, int y) String toString() void translate(int dx, int dy)

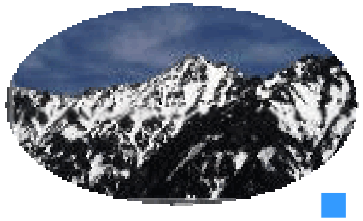


モデリングという共通概念

UMLの世界

- UML：特定の実装技術ではなく、それを抽象化して

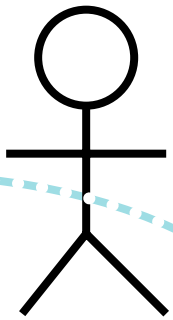




分析・設計・実装・UML　再掲

- アジャイル宣言
- SWEBOK：プロセスフレームワーク
- 技術者の資格化

現在



分析

設計

実装

これまでの
システム開発経験

繰返し型

- オブジェクトのアイディアを
取り込んだ技術の進展

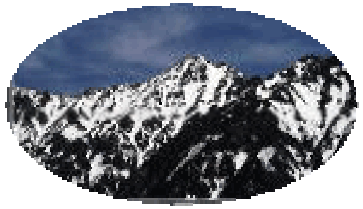
オブジェクトという共通概念

モデリングという共通概念

追跡性

UMLの世界

プログラミング言語
の世界



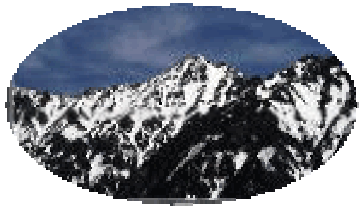
分析とは？/設計とは？

追跡性を取り戻すために

- Coad/Yourdon法の時代：分析＝「現実世界」のモデリング
- RUPでは、分析＝ユースケースモデルの解析
  アーキテクチャ分析

- 参考）CatalysisSMでは、
  - ビジネスプロセスモデリング
  - システムコンテキスト設計と仕様作成
  - コンポーネント設計とコンポーネントの仕様作成
  - コンポーネントのオブジェクト設計と実装

http://www.catalysis.org/

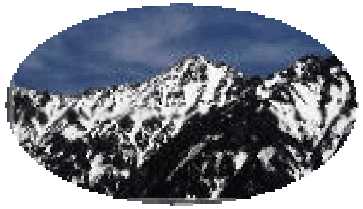


参考：

CatalysisSMの成果物

- ビジネスモデル(あるいはドメインモデル)
  - ユーザの世界の記述
  - ソフトウェアによる実現とは無関係
- 要件の仕様
  - システムが何をするかの記述
  - システムの実装とは無関係
- コンポーネント設計
  - 主要コンポーネント間の(高レベル)協調関係の記述
- オブジェクト設計
  - コンポーネント間の協調関係をクラスやオブジェクトのレベルで記述
- コンポーネントキットアーキテクチャ
  - コンポーネント集合を纏め上げるための方式の記述 (内部プロトコルなど)

http://www.catalysis.org/

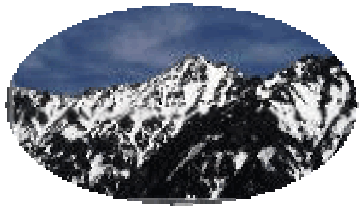


分析とは？/設計とは？

- ？「**分析は** “What” v.s. **設計は** “How”」

http://dictionary.goo.ne.jp/

- **分析**:**a･nal･y･sis**
  —— *n.* ⟨*pl.* a･nal･y･ses⟩ **分解**; **分析**; 【数】 **解析** ⟨学⟩ ; 【文法】 **解剖**;
- **設計**¦**de･sign**
  —— *n.* **計画**, **目的**, **意図**; ⟨*pl.*⟩ **陰謀**; **設計**, **筋書**; **下絵**; **図案**, **デザイン**.
  —— *v.* **計画** [**企図**] **する**; **予定する** ((for, as)); **もくろむ**; **設計する**, **図案を作る**.

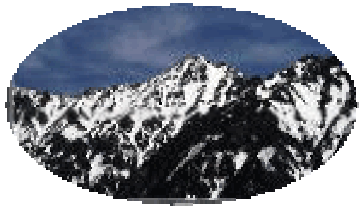


分析とは？/設計とは？

- ソフトウェアシステムのための分析とは？
  - 基本設計書、概念設計書、外部設計書etc. を作成するために、必要なXsを分析する（分解して、わかりやすい形にする）

- Xs = { 業務分析,
  データ分析,
  要求分析：　機能要求、
  非機能的な要求
  制約：
  ハードウェアに対する制約
  ソフトウェアに対する制約
  開発期間に対する制約・制作費に関する制約…,
  現在利用可能な技術の分析
  コモナリティ分析/バリアビリティ分析}

- 分析結果は？



参考)

概念モデル

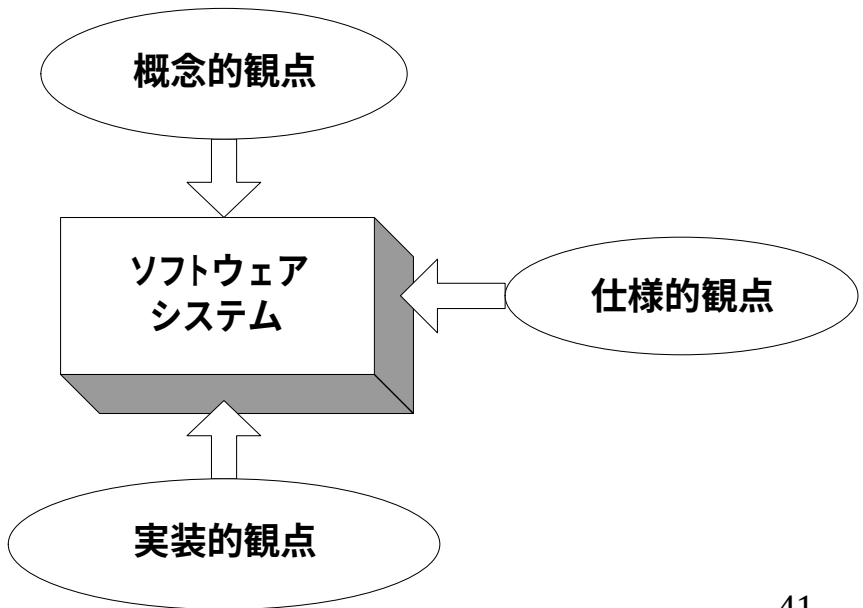
- **概念モデル　≒　分析結果**

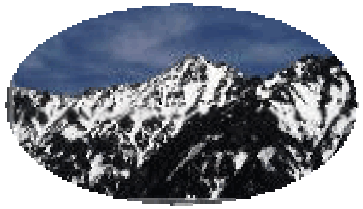
Steve Cook and John Daniels, “Designing Object Systems: Object-Oriented Modeling with Syntropy”, Prentice Hall, 1994

ソフトウェアシステム開発のためのクラス図は

- **概念の観点**
- **仕様の観点**
- **実装の観点**

から考察する必要がある

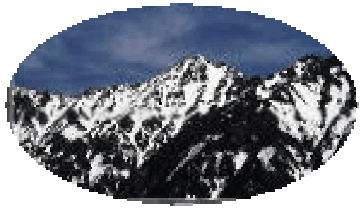




参考）

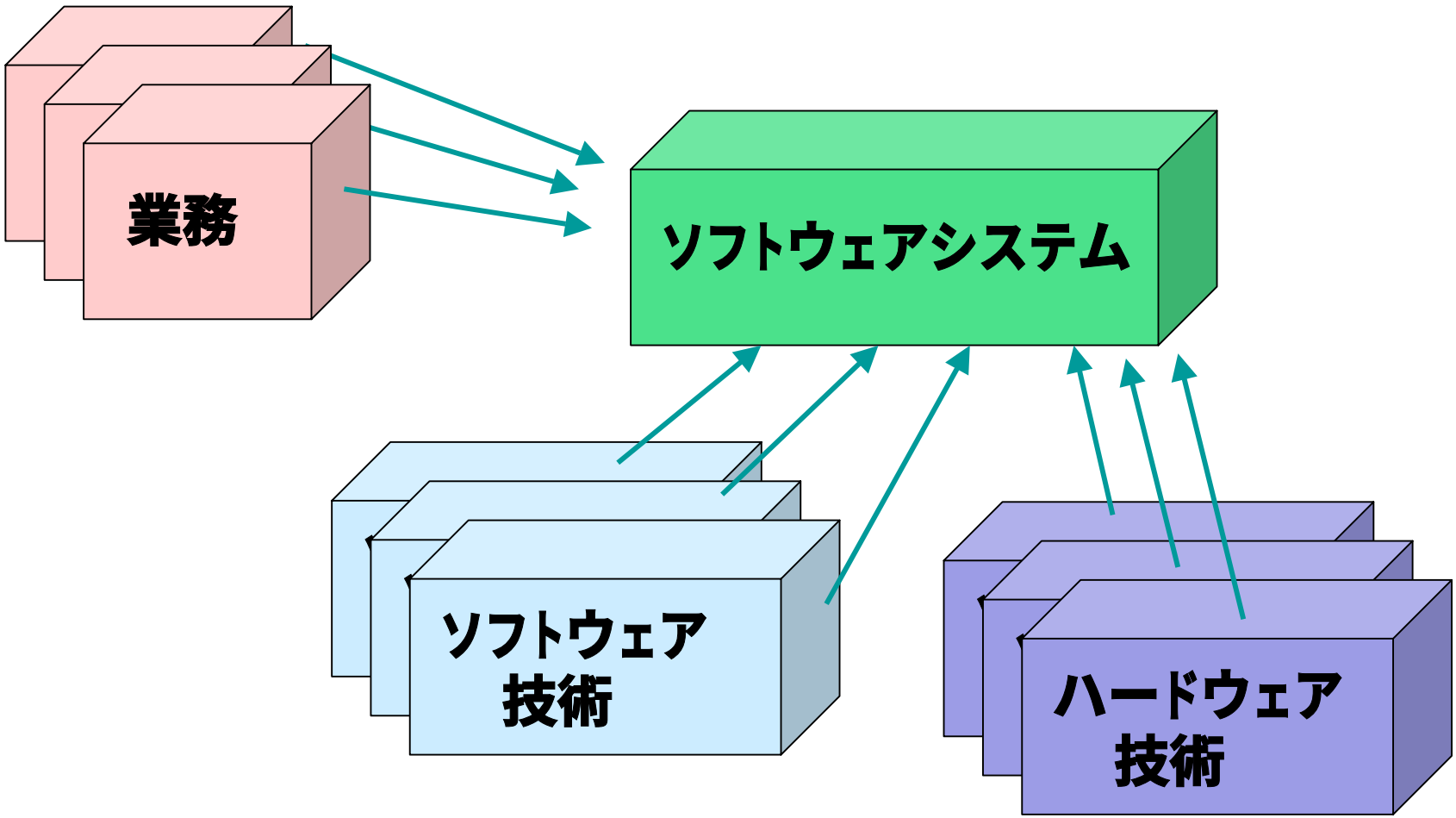
To-Beモデルと As-Isモデル

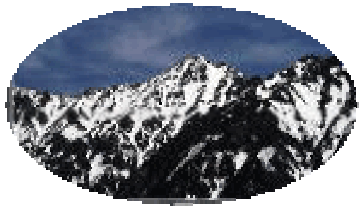
- 現状を表現するモデルをAs-Isモデルという
- それに対して、「こうであるのが望ましい（楽しい、新しい責務の可能性を追加etc.）」を表現したモデルをTo-Beモデルという



ドメインという考え方

- ドメイン：「定義域」





ソフトウェア構築のドメイン

問題 (problem) ドメイン

業務

リレーション

ソリューション (solution) ドメイン

ソフトウェアシステム

問題ドメイン

ドメインには、スコープがある

ソリューション間の
拮抗を
どう解決するか

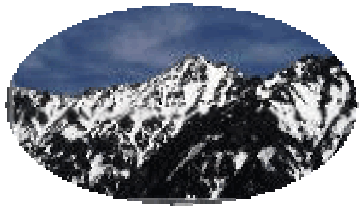
ソフトウェアパターン の可能性

ソフトウェア技術

ソフト
技術

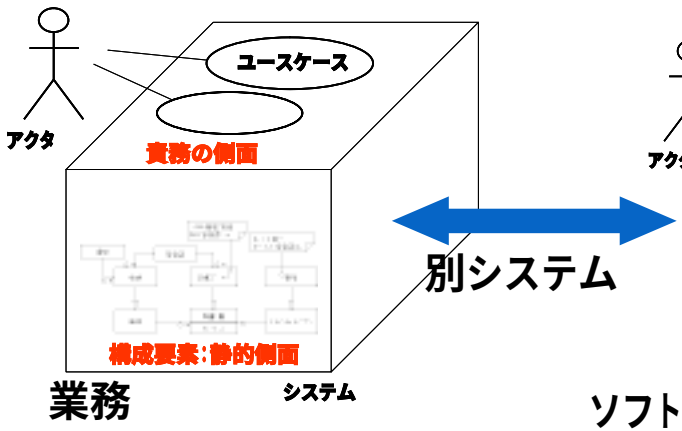
ハードウェア
技術

ソリューションドメイン

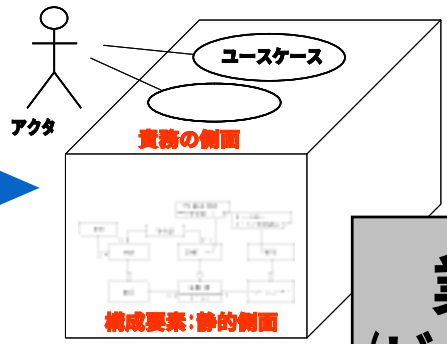


業務のモデルとソフトウェアシステム

- 業務の変更に即応できるソフトウェアシステムという理想



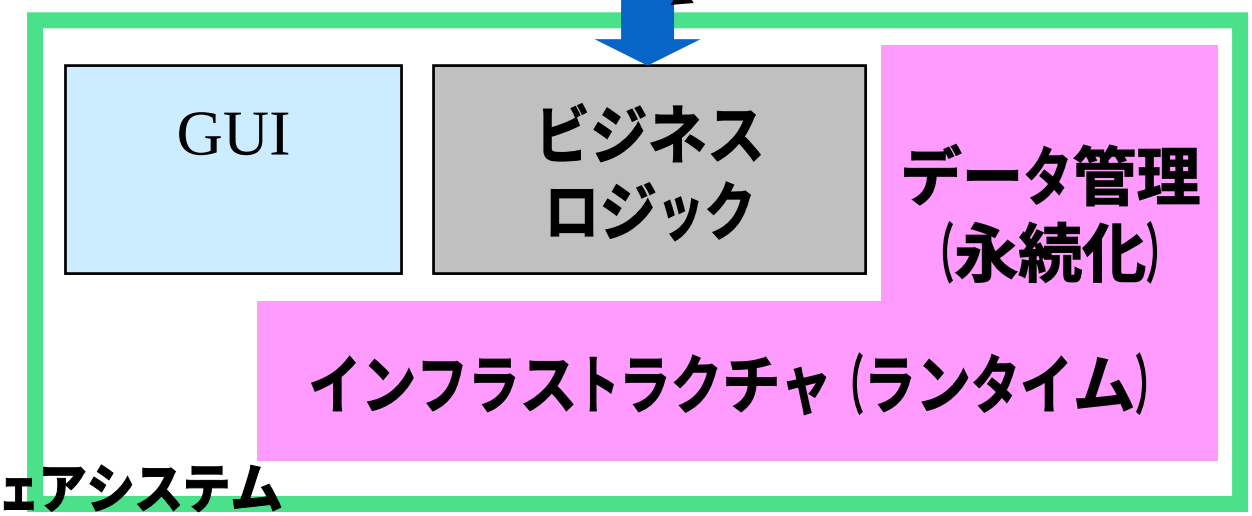
別システム

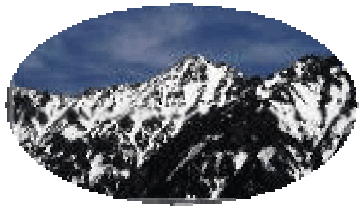


追跡性を実現する
1つの方法

業務モデル
(ビジネスモデル)

現在は、
業務モデルのオブジェクト
に相当するオブジェクトを
作り上げるとはかぎらない





What と How

- 仕様とその実現(implementation)　「インタフェースと実装の分離」

UMLのサブシステムの表記　　ビジネスロジック

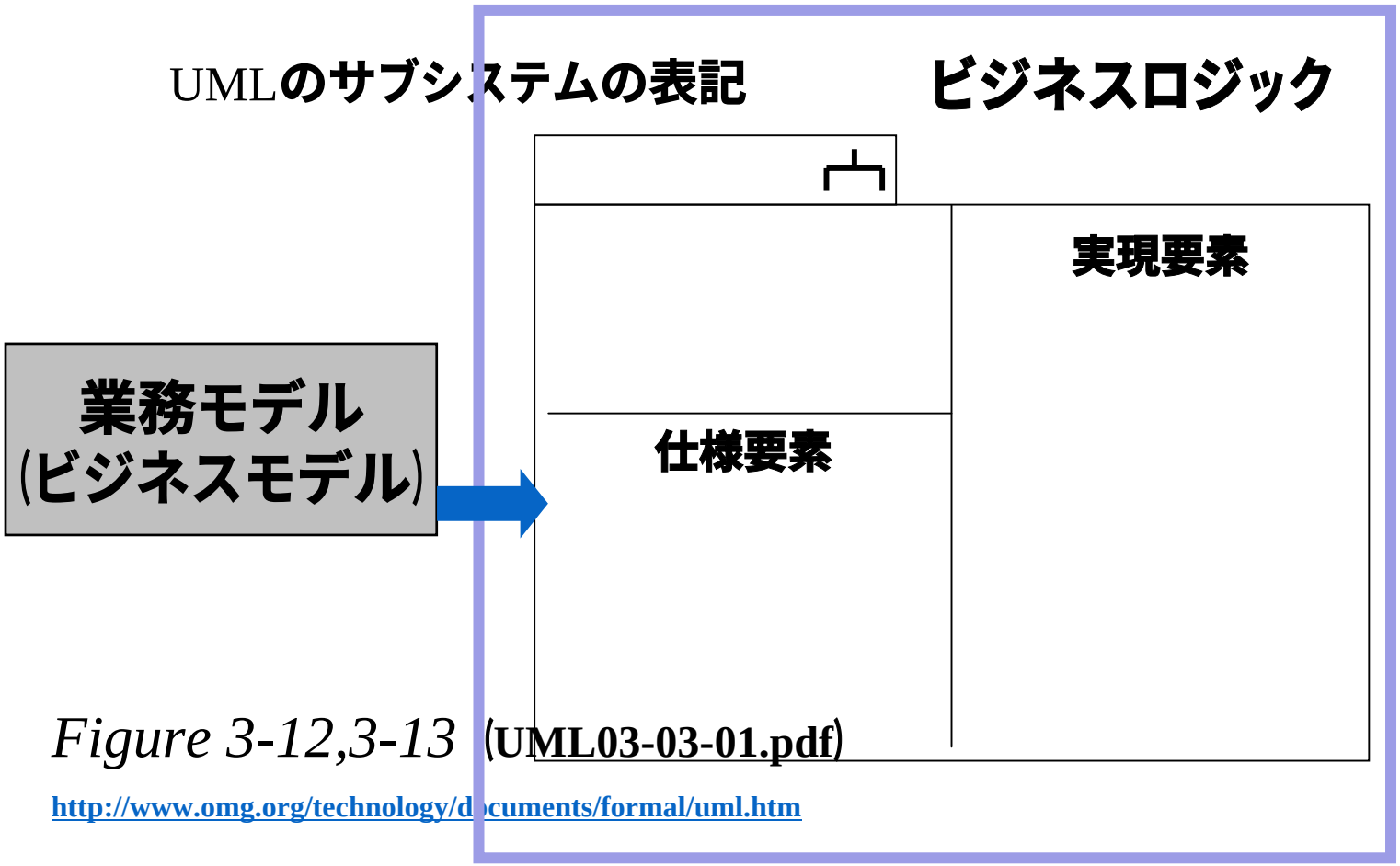
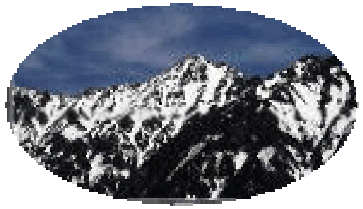


Figure 3-12,3-13 (UML03-03-01.pdf)

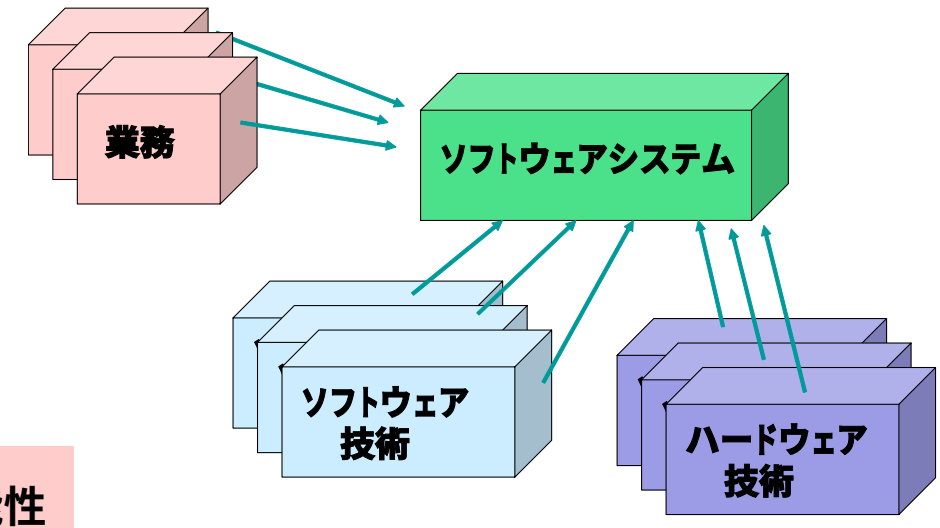
http://www.omg.org/technology/documents/formal/uml.htm



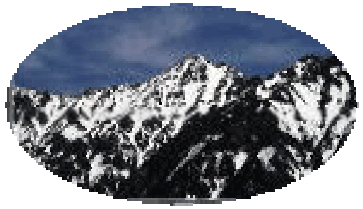
コミュニケーションとソフトウェアシステム

- ドメインの分析のためには、
  知識･経験を持っている人から情報を聞きだすことが必要

- 知識は、そのドメインのエキスパートに。
  - 1度にすべてを聞き出すことはできないのでは？
    → 繰返し型
    → カプセル化
  - ＜私の経験＞
    業務知識の形式化と
    ユースケースは親和性が
    よい



ソフトウェアパターン の可能性



- 質のよい経験･知恵を共有しよう！
  → 1つの方法として、ソフトウェアパターン
- “コミュニティ”の中で伝わるものがあります！
  → “オブジェクト倶楽部”という方法

どうもありがとうございました

そして、 *Merry Christmas!*