形式手法の特徴と応用

酒匂寛
Designers' Den Corp.
sako.hiroshi@gmail.com

本日の話題

- 仕様の位置付け
- 形式手法とは何か
- 形式化の簡単な例
- 形式手法の適用事例
- 形式手法適用の課題

自己紹介

- 1980 年代
  - COBOL とワークステーションの時代
  - 汎用機・オフコンを用いた COBOL 開発
  - ワークステーション上での開発環境の開発
- 1990 年代
  - 大規模プロジェクトとオブジェクト指向の時代
  - 2000人月超の大規模プロジェクトをOOで
  - 組込み機器のソフトウェアをフレームワーク化
- 2000 年代
  - モデリングと形式手法の時代
  - 証券、組込み機器に形式手法を適用
  - 領域・課題・仕様・設計・検証のフレームワーク化

宣伝：本の紹介

VDM++によるオブジェクト指向システムの高品質設計と検証（翔泳社）

ジョン・フィッツジェラルド、ピーター・ゴルム・ラーセン、ポール・マッカージー、ニコ・プラット、マーセル・バーホフ (著), 酒匂 寛 (翻訳)

VDM++を用いて、形式モデリングの基礎と応用を学習します。他の形式手法にも応用可能です。

本日の話題

- ⇒ 仕様の位置付け
- 形式手法とは何か
- 形式化の簡単な例
- 形式手法の適用事例
- 形式手法適用の課題

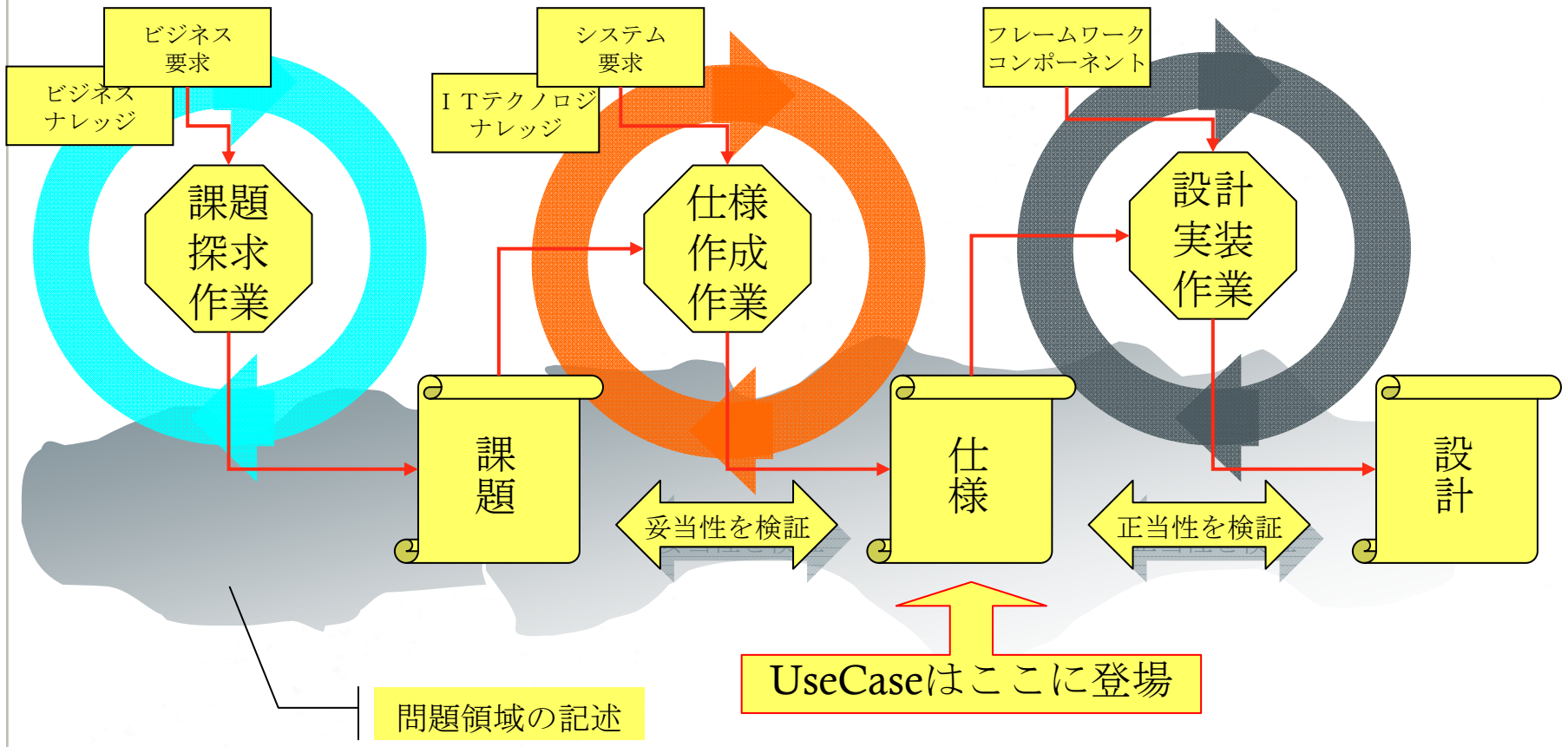
仕様の位置付け

- 「仕様」は課題と設計をつなぐもの
  - 言い換えれば「問題の解決」が「問題の解法」にどのように結び付けられるかを記述するもの
  - 「問題の解決」とは：
    - どのような条件のもとでどのような効果・状態が得られたり維持されたりすれば、問題が解消されたといえるのかを規定したもの
  - 「問題の解法」とは：
    - 与えられた材料で指定された効果、状態を創出するために具体的に行うべき手順を書いたもの

システム開発の３つの視点

- 課題 (要求＋問題領域)
  - 利用者の視点
    - 現実世界にどのような問題を抱えており、そのうちどの部分をシステム化して解決したいと思っているのか
    - どのような状態が「問題の解決」といえるのかを定義
  - 入力：ビジネス要求、利用者の要求、問題領域の知識
- 仕様
  - 利用者と開発者の視点
    - 上で挙げられた課題の解決を、どのようなシステムで支援するのか
    - 「問題の解決」と「問題の解法」の対応関係を定義
  - 入力：課題＋システム要求
- 設計
  - 開発者の視点
    - 要求されるシステムの仕様をどのように設計すべきか
    - どのようにして求められる効果を生み出すか➔「問題の解法」を具体的に定義
  - 入力：仕様＋最新構築技術

課題-仕様-設計の関係


ビジネス
要求
ビジネス
ナレッジ
課題
探求
作業
システム
要求
ＩＴテクノロジ
ナレッジ
仕様
作成
作業
フレームワーク
コンポーネント
設計
実装
作業
課
題
妥当性を検証
仕
様
正当性を検証
設
計
UseCaseはここに登場
問題領域の記述

課題・仕様・設計

目的地への誘導

道案内をする

課題

どのような入力（位置、選択条件）で、どのような出力（経路）を得たいのか

経路探索仕様

個別の機能をどのように組み合わせて目的を達成するか→操作仕様

課題は複数の機能仕様に写像される

仕様

どのような計算方法を用いて、求められる結果を得るのか

設計

経路探索設計

仕様記述が満たすべき性質

- [課題] 対応する課題の指定
  - どの問題・課題を解決するためにこの仕様は存在しているのか
- [目的] 目的（得られる効果・状態）の明記
  - 前提条件が満たされたという仮定の下で、どのような振舞いが期待されているのか
- [前提] 前提条件（文脈）の明記
  - どのような条件、入力が与えられた情況でこの仕様が求められるのか
- [依存] 依存する仕様の定義
  - どの仕様に依存しているのか

多くの仕様書では、これらがきちんと書かれていない
→（曖昧、未分化、未定義、重複、記述モレ）

仕様記述の品質を評価する

- 何より「検証可能」でなければならない
  - 「検証可能」な記述を体系的にテストすることによって品質を評価することができる
- 検証可能性に対する３つの視点
  - 抜けモレの排除
  - 記述の一貫性、完全性の確保
  - 記述間（要求⇔仕様、仕様⇔設計、設計⇔実装）の整合性の保証

抜けモレの排除

- 資産
  - テンプレートを使う
  - 標準的なチェックリストを用意する
- 記述
  - 一覧性のよい記述（表など）を使う
  - 対称性の悪い部分を見つけやすい記述を使う
- 知識
  - 「専門家」のレビュー

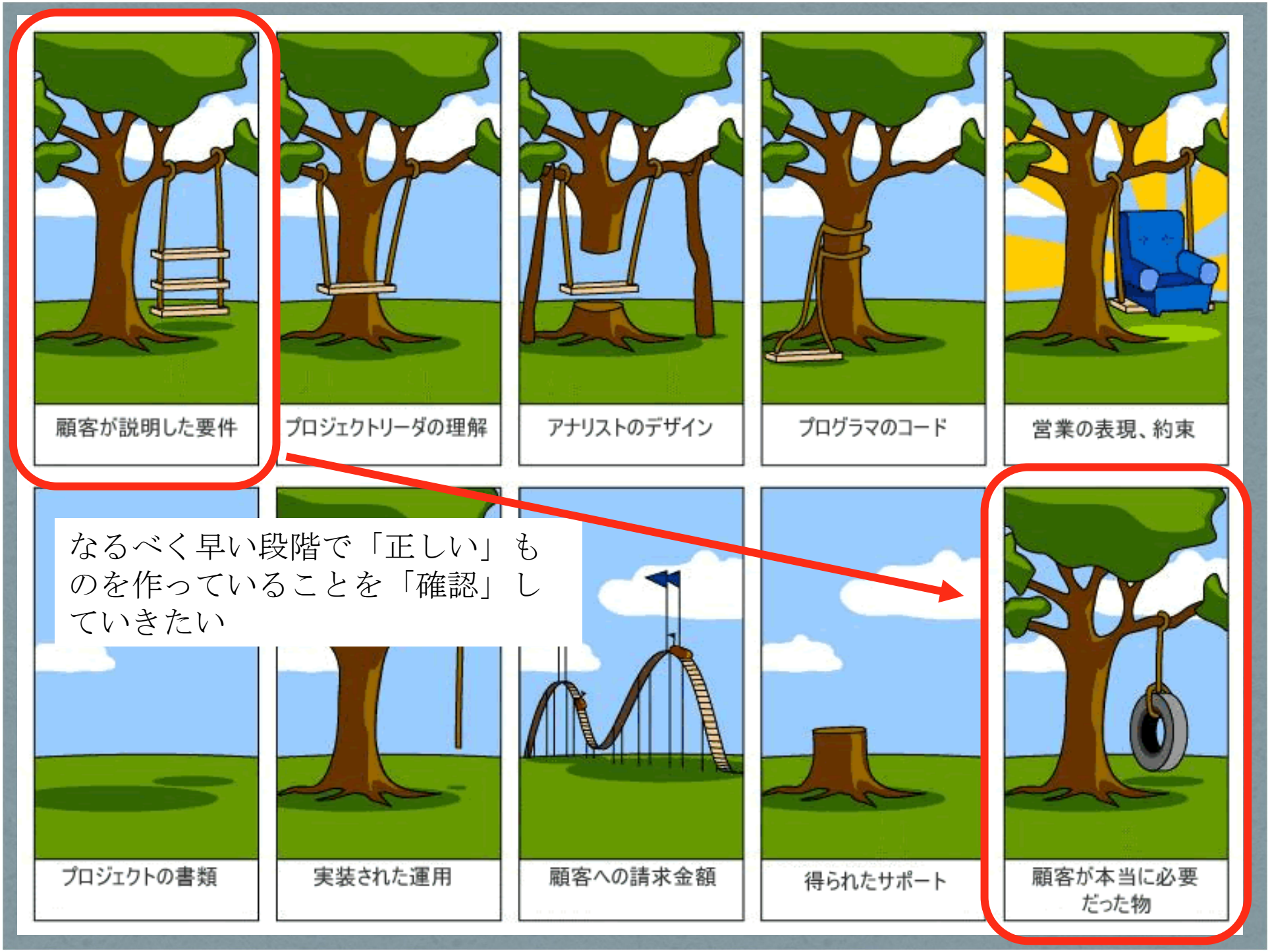
記述の一貫性、完全性の確保

- 資産
  - 用語集の整備
  - 標準仕様記述書の整備
- 記述
  - 統一形式で記述する
    - テンプレート、表、図、箇条書きほか
  - 記述に適した「言語」を用いる
    - 仕様記述言語など

記述間の整合性の保証

- 資産
  - 標準的なテストケース
  - 標準的なチェックリスト
- 記述
  - トレーサビリティのよい記述（要求、仕様、設計項目間の突合を容易にするため）
  - 上位記述と下位記述の間の「詳細化」を助ける記述の採用
    - 上位記述：何をする
    - 下位記述：どのようにする

詳細化


顧客が説明した要件
プロジェクトリーダの理解
アナリストのデザイン
プログラマのコード
営業の表現、約束
プロジェクトの書類
実装された運用
顧客への請求金額
得られたサポート
顧客が本当に必要だった物
なるべく早い段階で「正しい」ものを作っていることを「確認」していきたい

本日の話題

- 仕様の位置付け
- ⇒ 形式手法とは何か
- 形式化の簡単な例
- 形式手法の適用事例
- 形式手法適用の課題

形式手法とは何か

- 直接的な意味は
  - ソフトウェアに登場する様々な「記述」を一定の規則に従った形式で行なうこと
  - 構文的な形式だけではなく、意味的な裏付けもしっかりとした記述であるものが大部分である
- この結果
  - 曖昧さを各種の文書から取り除き
  - 検証を可能にし
  - 機械的支援が得やすくなる

曖昧な記述の例

- 「誰でも休日が好きだ」の意味は？
  - 誰でも全ての休日が好きだ。つまり誰でも仕事が休みになれば嬉しいという意味
  - 皆が好きな休日が少なくとも一つある。つまり誰でもある特定の休日 – 例えば大晦日 – が好きだという意味
  - 誰でも自分が好きな休日が少なくとも一つはあるという意味。例えば太郎は正月が好きで、花子は海の日と緑の日が好きといったこと
  - 誰でもちょうど一つだけ好きな休日があるという意味。例えば太郎は正月だけが好きで、花子は建国記念日だけが好きというような場合

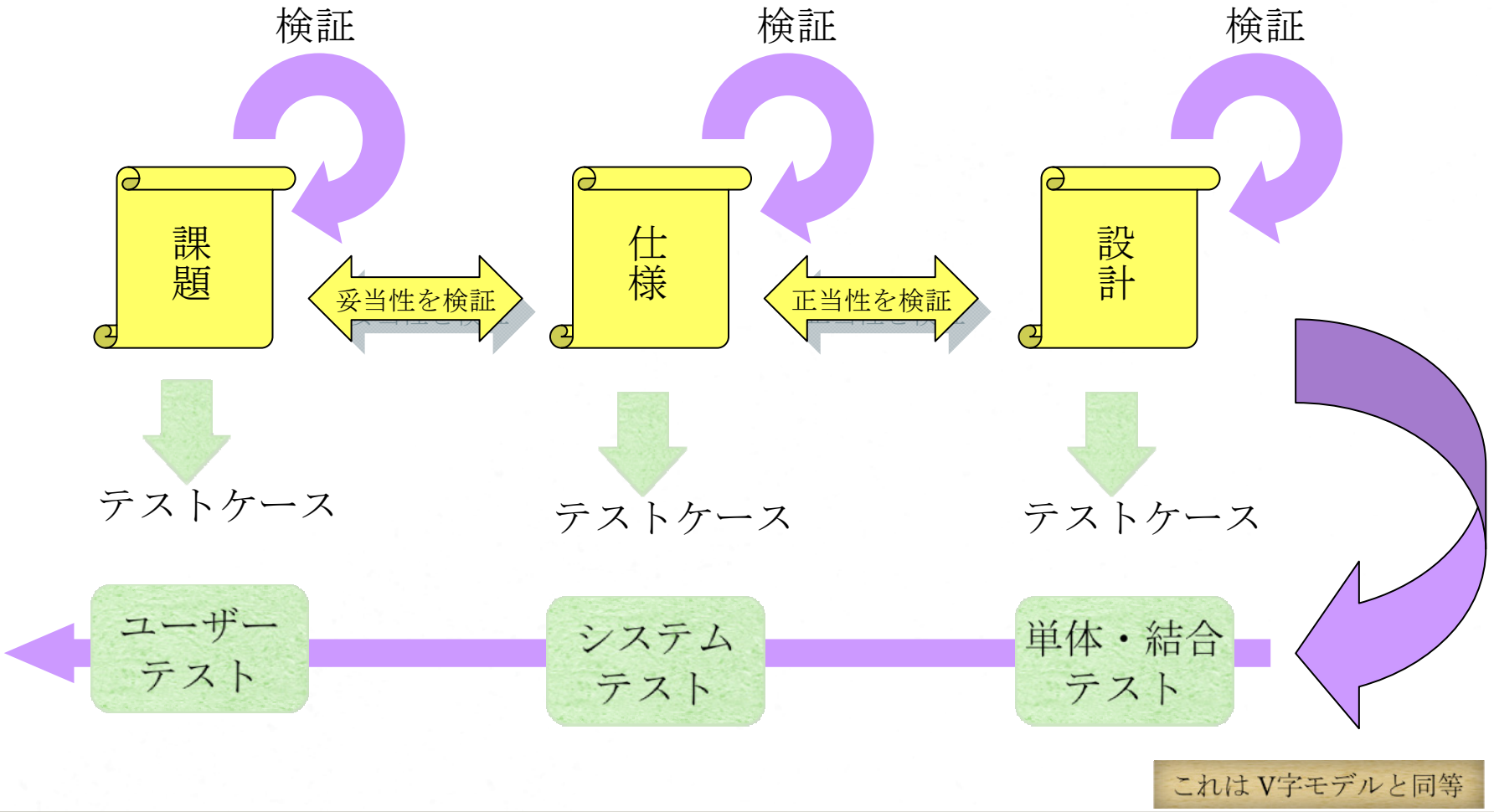
曖昧な記述の除去

- 以下のような述語を用意する
  - 好き(x, y) = x は y が好きである
- このとき以下の記述を見てみよう
  - 誰でも全ての休日が好きだ。つまり誰でも仕事が休みになれば嬉しいという意味 [仕様1]
  - ∀ x:人, y:休日・好き(x, y)

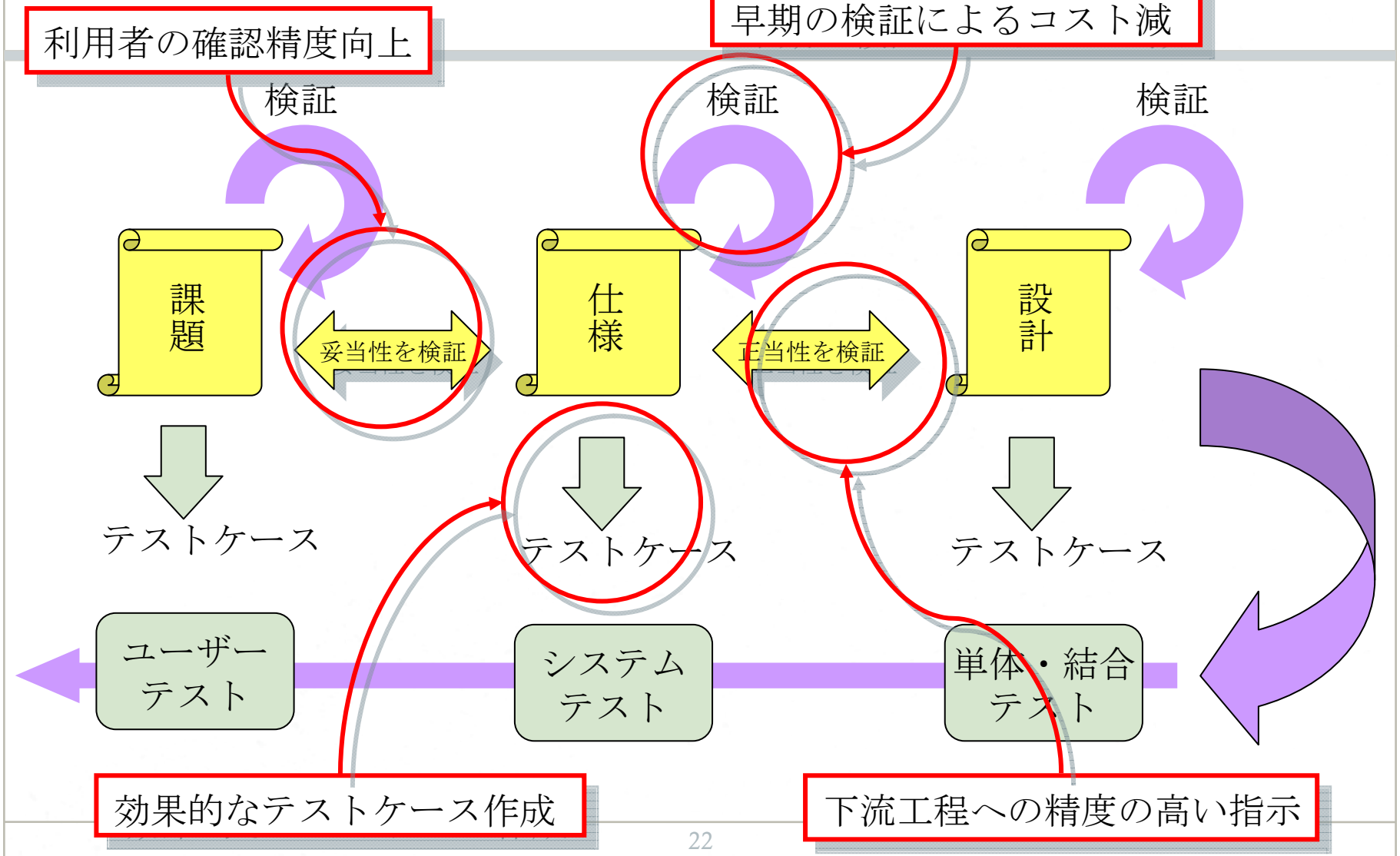
曖昧な記述の除去（続き）

- 皆が好きな休日が一つある。つまり誰でもある特定の休日 – 例えば大晦日 – が好きだという意味 [仕様2]
  - ∃ y:休日・(∀x:人・好き(x, y))
- 誰でも自分が好きな休日が少なくとも一つはあるという意味。例えば太郎は正月が好きで、花子は海の日と緑の日が好き[仕様3]
  - ∀x:人・(∃y:休日・好き(x, y))

記述とテスト


検証
検証
検証
課題
仕様
設計
妥当性を検証
正当性を検証
テストケース
テストケース
テストケース
ユーザーテスト
システムテスト
単体・結合テスト
これはV字モデルと同等

記述の効果 (仕様を中心に考えた場合)



形式記述/モデルのご利益

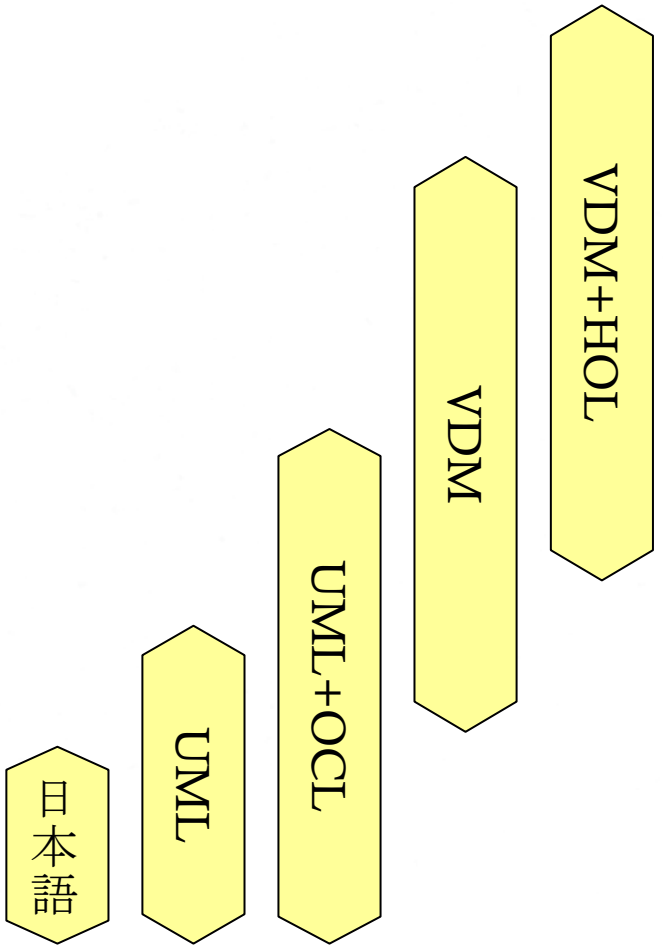
形式記述/モデル

人間に対する
より抽象的で理解しやすい記述の提示

機械処理による検証
（内容の一貫性保証、
自動回帰テストなど）

仕様記述としてのレベル

level 5	証明可能 .. 厳密な証明を行うことができる
level 4	検証可能 .. 評価した仕様を各種の「表明」と機械的に比較検証できる
level 3	評価可能 .. 部分的にでも仕様を機械的に「評価」できる
level 2	変換可能 .. ある一定の文法で書かれている
level 1	ただ記述しているだけ .. 抜けモレ不整合があってもわからない



本日の話題

- 仕様の位置付け
- 形式手法とは何か
- ⇒ 形式化の簡単な例
- 形式手法の適用事例
- 形式手法適用の課題

形式化の簡単な例

「誰でも休日が好きだ」の意味は？再び。

- 誰でも全ての休日が好きだ。つまり誰でも仕事が休みになれば嬉しいという意味 [仕様1]
  - ∀ x:人, y:休日・好き(x, y)
- 皆が好きな休日が一つある。つまり誰でもある特定の休日 – 例えば大晦日 – が好きだという意味 [仕様2]
  - ∃ y:休日・(∀x:人・好き(x, y))
- 誰でも自分が好きな休日が少なくとも一つはあるという意味。例えば太郎は正月が好きで、花子は海の日と緑の日が好き [仕様3]
  - ∀x:人・(∃y:休日・好き(x, y))

VDM++による記述(1)

```
人集合 : set of 人 := {<太郎>, <花子>,<竜二>};

休日集合 : set of 休日 := {<元旦>, <昭和の日>, <子供の日>};

好きな休日 : map 人 to set of 休日 := {
        <太郎> |-> {<元旦>},
        <花子> |-> {<元旦>, <子供の日>},
        <竜二> |-> {<子供の日>}
      };
```

VDM++による記述(2)

```
好き : 人 * 休日 ==> bool
 好き(p, h) ==
       return h in set 好きな休日(p)
       pre p in set dom 好きな休日;

 public 誰でも休日が好きだ1 : () ==> bool
 誰でも休日が好きだ1() ==
      return forall p in set 人集合, h in set 休日集合 & 好き(p, h);

 public 誰でも休日が好きだ2 : () ==> bool
 誰でも休日が好きだ2() ==
      return exists h in set 休日集合 &
            forall p in set 人集合 & 好き(p, h);

 public 誰でも休日が好きだ3 : () ==> bool
 誰でも休日が好きだ3() ==
      return forall p in set 人集合 &
            exists h in set 休日集合 & 好き(p, h);
```

Demo

ツールを使った簡単なデモを行います

VDMTools の特徴

http://vdmtools.jp を参照

- VDM で記述された仕様の検証
  - 静的検証
    - 構文、型
  - 動的検証
    - 仕様の評価・実行。カバレージの生成
    - 事前事後、不変条件のチェック
- 外部ツールとの連携
  - XMI を介した UML との相互変換
  - API を介した「仕様サーバー」の提供
  - 仕様から実行コードを生成 (Java, C++)
  - Java から VDM++を生成

本日の話題

- 仕様の位置付け
- 形式手法とは何か
- 形式化の簡単な例
- ⇒ 形式手法の適用事例
- 形式手法適用の課題

形式手法の適用事例

- 証券システム - TradeOne
  - オンライン証券システムのバックオフィスの一部の開発に形式手法を適用
- 組込みソフトウェア - FeliCa
  - モバイル FeliCa チップのファームウェア開発に形式手法を適用

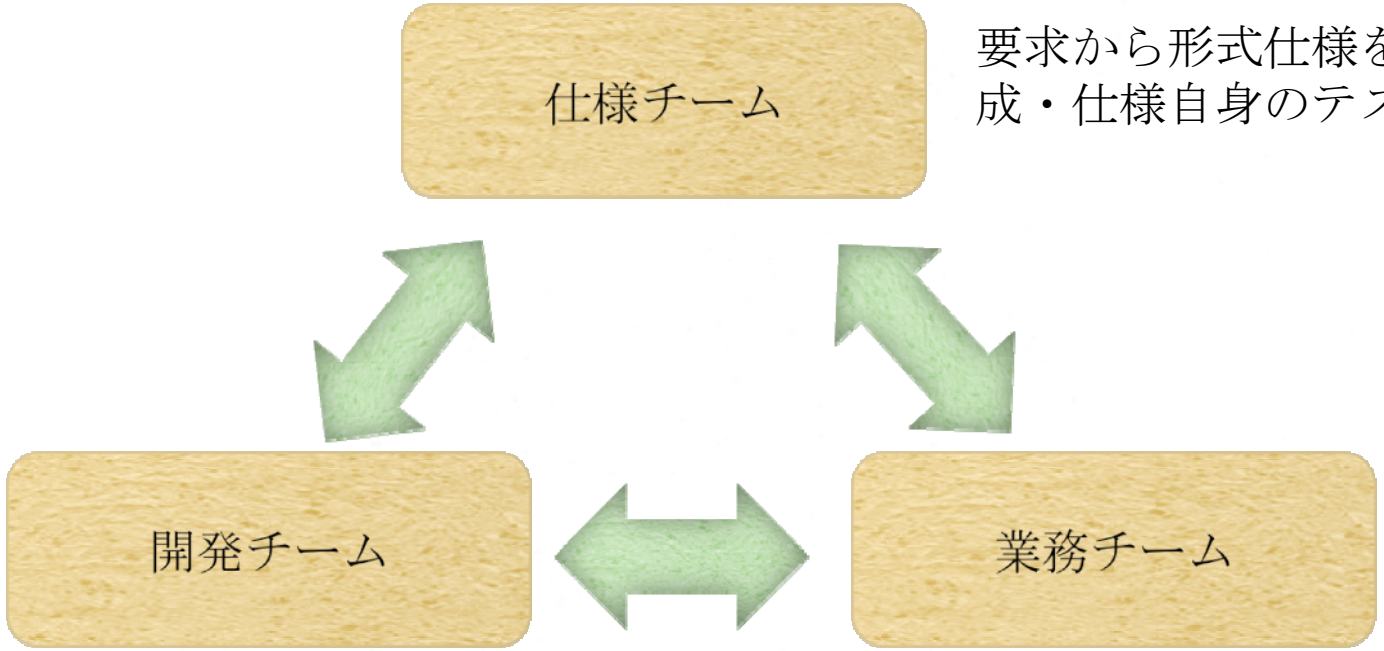
形式手法の適用範囲

- どちらのプロジェクトも形式仕様記述言語 VDM++ を使って記述と検証を行った。設計実装は開発者が手で行っている
- TradeOne
  - ユースケースレベルの記述を行った
  - 業務論理仕様を詳細に記述し、回帰テストを行った
- FeliCa
  - 外部仕様を詳細に記述し、回帰テストを行なった
  - 外部仕様書の一部を形式仕様から自動生成した

開発プロファイル

	規模	市場欠陥	生産性見積比	開発期間見積比	開発期間	開発者
TradeOne	C++ 80K VDM 30+30K	0	2.5倍	45%	６ヶ月	形式手法知識 2 全員 40 歳超
FeliCa	C/asm 82K VDM 30+53K	0	1.3倍	100%	２年 ２ヶ月	形式手法知識 0 ほぼ 30 歳以下

TradeOne のチーム構成



要求から形式仕様を作成・仕様自身のテスト

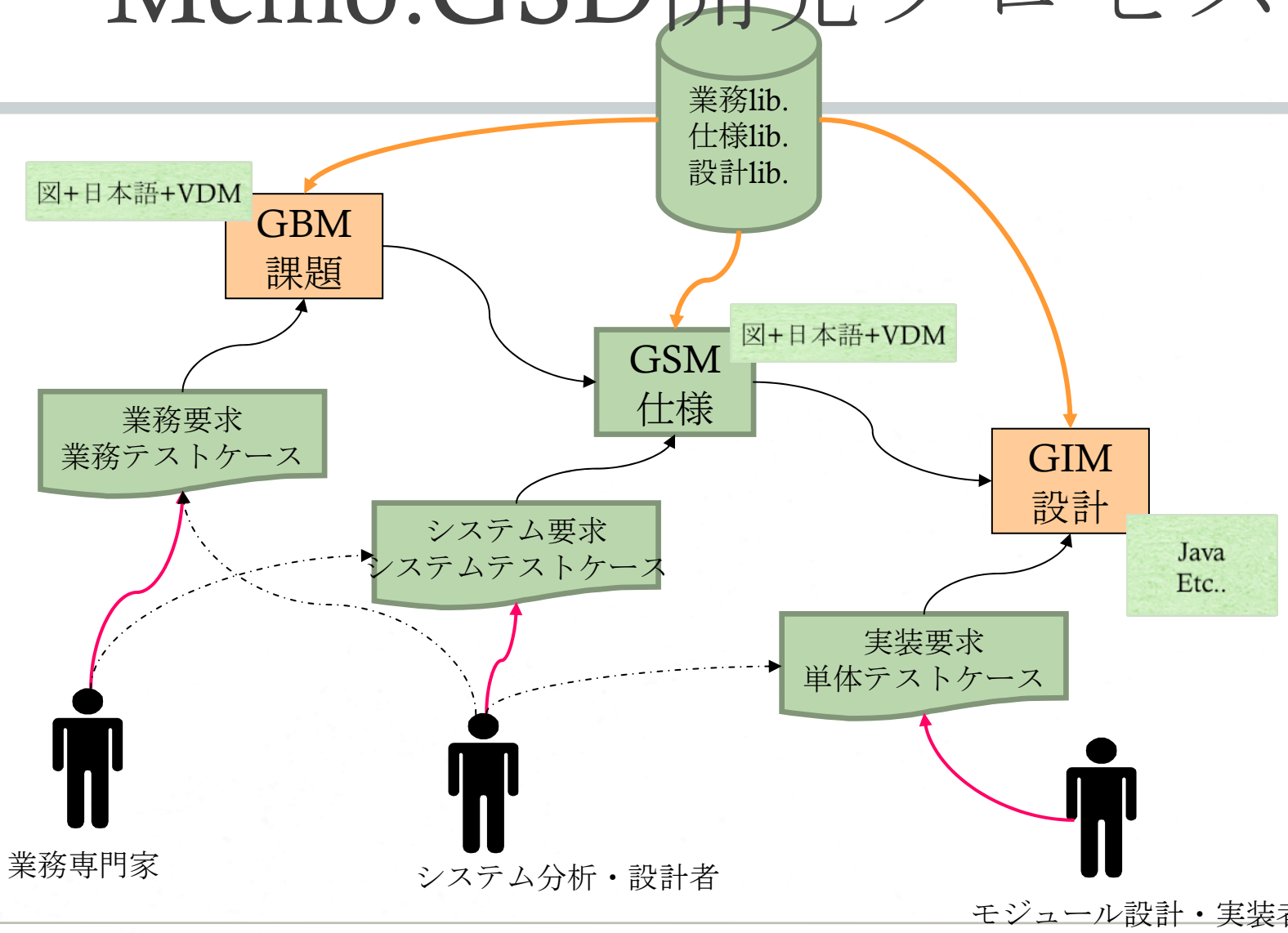
フレームワーク設計
形式仕様から設計・実装・単体テスト

要求の提示
結合、システムテスト

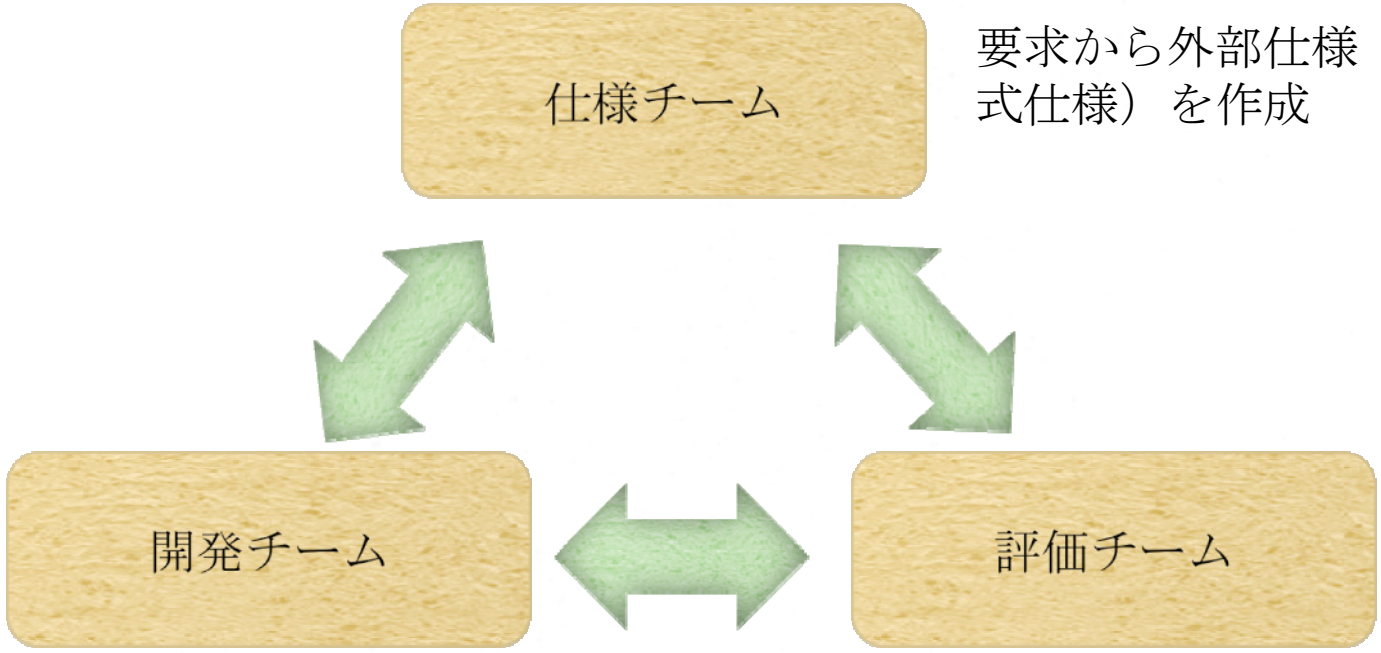
TradeOne適用の効果

- 業務理解の促進
  - 仕様チームは証券ドメインの知識をほぼ持っていなかった。しかし業務チームの提示する要求を形式的に記述する作業を通して急速に業務知識を習得した
- 業務知識の明確化
  - 業務チームにとっても、暗黙的な知識が可視化され専門家間に存在した微妙な解釈の違いも明確になった
- 仕様→設計の道筋を明確化
  - 仕様、設計それぞれにフレームワークを用意し、対応関係を明示することによって作業の間違いを減らし、検証も容易にした

Memo:GSD開発プロセス



FeliCa のチーム構成



要求から外部仕様（形式仕様）を作成

N個の設計・実装

(N+1)個の評価
テスト仕様自身の評価

FeliCa各チームの満足度

- 仕様チーム
  - 効果の高さは全員が理解した
  - 仕様変更の行き先が一元化されて管理も楽
- 開発チーム
  - 満足 30%、普通 50%、不満～無回答 20%
  - スキルの高い人ほど満足度合いが高い（仕様策定に関わることができた）
- 評価チーム
  - 満足 80%、普通 15%、不満～無回答 5%
  - 一般的に評価チームはよりどころのない作業を強いられるため

本日の話題

- 仕様の位置付け
- 形式手法とは何か
- 形式化の簡単な例
- 形式手法の適用事例
- ⇒ 形式手法適用の課題

形式手法導入の課題

- 適用範囲の設定
- 形式記述の準備
- 導入計画
- 要員教育
- 外部リソースの活用

適用範囲の設定

- 全てのシーンに適用可能であるため、逆にどの部分に適用するべきかをきちんと決めることが大切
- 記述することによるメリットが大きい部分への適用が望まれる
- 記述対象例
  - 問題領域の指示、定義
  - 問題領域の制約、型
  - 外部仕様の記述（シグネチャ、事前事後条件、不変条件）
  - 仕様のテストケース（実装のテストにも利用出来る）

形式記述の準備

- 問題領域分析を行い以下の準備を行なうのが理想
  - 適切な用語の抽出
  - 各種制約条件の定義
- 仕様記述のためのフレームワークを準備
  - 仕様のテストに利用することを意識する
  - 仕様記述の粒度が揃えられるようにする
- 記述ガイドラインの準備
  - 命名規則
  - テストケースの作成方針

導入計画

- 形式手法を取り込んだプロセスを可視化しておく
  - 記述準備・要員教育
  - 要求の記述と仕様への反映手順
  - 仕様の検証と検査仕様の管理
  - 仕様の構成管理・版管理
  - 各種文書の生成
- 最初は普通の開発計画と大きく異なるフローを考える必要はない
  - 仕様策定の部分に時間が掛かるように見える、これは仕様の部分のテストを早い段階で行っているから

要員教育

- 一般に読むのは容易だが、書くためにはある程度の経験が必要
- VDM++ の場合
  - 読むだけ：数日の教育で可能
  - 記述する：フレームワークが準備されている場合、指導者について１ヶ月実践

外部リソースの活用

- コミュニティの利用
  - VDMの場合 → http://vdmtools.jp/
  - 多くの場合、最初は専門家のアドバイスを得て試行する方がうまく行く
- 書籍も増加中
  - VDM++ 以外にも B メソッドの教科書や、モデル検査の SPIN の教科書なども出版されている

まとめ

- 形式手法は既に実用レベルに達しています
  - 例えばVDM++ は開発のさまざまな段階の記述に利用することができます
  - 単なる「記述」ではなく様々な分析検証が可能であるところがUMLなどに比べて有利な点です
- 形式手法は記述を改善します
  - 適用範囲を心得て使えば失うものは何もありません
  - 伝達ミスを大幅に減らし、プロジェクトの風通しを良くします
- 形式手法は関係者に「安心」を与えてくれます
  - 検証済の仕様を常に手元に置いておけることは何にも代えがたい安心感につながります
  - 問題を人間の頭の外に書き出すことによって、隠された秘密が減り、作業量に関するより詳細な予測を立てやすくなります

最後に

- 形式手法は強力ですが
  - 魔法の杖ではありません。
- 他の有用なソフトウェア工学のあらゆる手法と併用して利用しなければなりません。
- しかし技術者にとっても組織にとっても、投資する価値のある技術です。

蛇足:組み合わせて活用したい道具立ての例

- プロブレム・フレーム : M. Jackson
  - 問題そのものの構造を捉え、定式化と仕様化の良いガイドとなる
- 状態機械モデル : UML etc...
  - 更にモデル検査を使って安全性などを検証
- 仕様記述言語 : VDM++, B など
  - 仕様が中心。領域の記述にも威力を発揮（指示、定義、制約、型）。課題、設計も記述可能
  - 検証：体系的テスト、証明、仕様アニメーション etc ...
- 「記述」を自由に処理／加工できる環境 : Unix 系の様々なツールなど
  - TeX、スクリプト言語、構成管理、その他もろもろ