形式仕様記述(応用編)
イントロダクション

トップエスイープロジェクト
国立情報学研究所　石川　冬樹

目次

- 本講座の位置づけ
- 形式仕様記述の考え方 – 復習
- 代表的な適用事例紹介

形式仕様記述講座シリーズとは？

数理論理学などに基づき品質の高い
ソフトウェアを効率よく開発するための
科学的・系統的アプローチを学ぶ

システムの注目する側面を正確に，曖昧さのない言語で表現する

➡ 開発の早い過程の中間成果物において，

曖昧さや思い込みを排除

科学的・系統的な分析・検証により品質向上

➡ 後の過程の誤り発覚による手戻りを防止

形式手法の思想

- 仕様や設計を「きちんと」書こうとする過程により
  - 曖昧さを解消することになる
  - 不正確，不整合も表面化する
  - システムに対する理解が深まる
- 仕様や設計を「きちんと」書いた結果により
  - 誤解なく情報を共有できる
  - 科学的・系統的な分析・検証ができる
  - 分析・検証をツールに支援させることができる

形式手法の思想

厳密な文法・意味論，
理論的裏付けを持つ
言語を用いて

- 仕様や設計を「きちんと」書こうとする過程により
  - 曖昧さを解消することになる
  - 不正確，不整合も表面化する
  - システムに対する理解が深まる
- 仕様や設計を「きちんと」書いた結果により
  - 誤解なく情報を共有できる
  - 科学的・系統的な分析・検証ができる
  - 分析・検証をツールに支援させることができる

モデル検査
定理証明

「作ってしまう」前に，途中過程の
成果物の質を保証し，引き継いでいく

形式手法への期待・その必要性の高まり

- ソフトウェアシステム開発への要求の高まり
  - 成果物の質・信頼性
  - 開発手法・プロセスの系統化・効率化
- 評価・認証
  - （例）ISO/IEC 15408（Common Criteria）：セキュリティの高レベルな保証の要件として，設計の形式検証が含まれている
- ソフトウェア工学スキル・カリキュラム
  - （例） J-07SE：「形式手法」「ソフトウェアV&V」

形式仕様記述

- 形式手法のうち，以下を重視した手法である「形式仕様記述」を学ぶ
  - データ構造やそれに対する操作
  - 実装（プログラム）へつないでいく過程

モデル検査シリーズ（よりピンポイントな，状態の網羅的な検証）と補完

代表例：VDM

（株）フェリカネットワークスによる適用

代表例：Bメソッド

パリの地下鉄や空港自動運転シャトルなどへの適用
EUプロジェクトによる拡張が進行中（Event-B）

本講座で扱う形式仕様記述の種類

- 最終的なプログラムの構成
  （CやJavaなどの手続き型言語を用いる場合）
  - 変数：システムの状態を保持する
  - 関数（操作，メソッドなど）：入力から出力を導出したり，変数の値を読み書きしたりする
- ➡ これらの構成に基づいた抽象的，形式的な仕様記述を考える言語・手法を扱う
  - 「モデル指向」と呼ばれる
  - 他に「性質指向」がある

講座シリーズ構成

同じ問題への，異なるアプローチによる事例演習

セキュリティ編
セキュリティ問題への適用

応用編
活用プロセス

基礎編
基礎概念

Event-B
最新手法 段階的詳細化

Z
歴史ある手法 定理証明

SPIN
別の視点 モデル検査

VDM
手軽な適用

- プログラムにより近い抽象モデル記述
- 宣言的定義と実行可能定義の二つの詳細度に主に注目
- 実行・テストによる検証，妥当性確認

両極端の対比

Bメソッド
高信頼性の追求

- 数学概念がより直接現れるモデル記述
- 要求からプログラムまで段階的に詳細化する過程をカバー
- 定理証明による厳密な検証

基礎編・応用編で扱う手法・ツール

- VDM
  - 一般のプログラマが親しみやすい，ライトウェイトな手法
  - 日本で最近特に注目（ツールが日本語対応，フェリカ事例）
  - ツール： VDM Toolbox（CSKシステムズ，日本）
- Bメソッド
  - 「正しいコードを得る」過程を追求する手法
  - 鉄道制御への応用が盛んに行われている
  - ツール： Atelier B（ClearSy，フランス）

グループ演習(基礎編・応用編)

- 題材: 先端ネットワーク環境(アドホックネットワーク)におけるルーティングプロトコル仕様OLSR
  - 実在する「新しい問題」に挑む
  - 質が高い(はずの)標準仕様(自然言語)に対する形式仕様の意義を議論する
- 方法: グループ演習・議論
  - 形式仕様記述の適用の目的・意図の設定に応じた様々なモデリング方針, ツール適用方針について議論する
  - 実際の記述や検証等は各個人で行う

評価

- 出欠も評価
  - 半分以上の出席は必須
- 毎回の小課題は練習用（提出不要）
  - できる限り講義中に演習，解説
- 評価対象は中間課題と最終課題
  - 特に最終課題（グループ演習）はトップエスイーの肝であり単位取得のためには必須とされている
  - ➡ 途中結果や考えた結果だけでも出して欲しい（時間的に・内容的に難しい課題であり，また過程にも大きな意義があるため）

目次

- 本講座の位置づけ
- 形式仕様記述の考え方 – 復習
  - モデル化・記述
  - 検証・妥当性確認
- 代表的な適用事例紹介

「モデル」の意味と意義

モデル：計算や予測を助けるために用いられる，システムやプロセスの単純化された記述，特に数学的な記述
（Oxford English Dictionaryの石川訳）

注目する側面に絞り，抽象化・単純化（・厳密化）

- ツールが効率的に正確・科学的な分析を行える
- 人間も重要な（難しい）本質に集中して，問題の明確化・分析を行える

モデル化・記述(１): 厳密化・明確化

- 厳密な文法規則や意味論が定まった言語を用いて，正確に，曖昧さがないように書く

「ユーザは常にAまたはBの状態」

user.isA() or user.isB()

(user.isA() and not user.isB())
　　or
(not user.isA() and user.isB())

どっちの意味？

一意な意味を持つ文法で記述するので，
曖昧さを排除することになる

「isAが定義されていない」
「isAの戻り値がbool型ではない」
といった，完全性や整合性などの
チェック基準が明確に定まる
（そしてツールを用いてチェックできる）

モデル化・記述(２): 抽象化・捨象

- 記述・分析の目的に応じて，不必要な詳細を省き，本質的な側面に絞って抽象的に記述する

```
private user : token;

private data : set of user;


public check : () ==> bool
check ==
  return forall x in set data
              & isConsistentState(user);
```

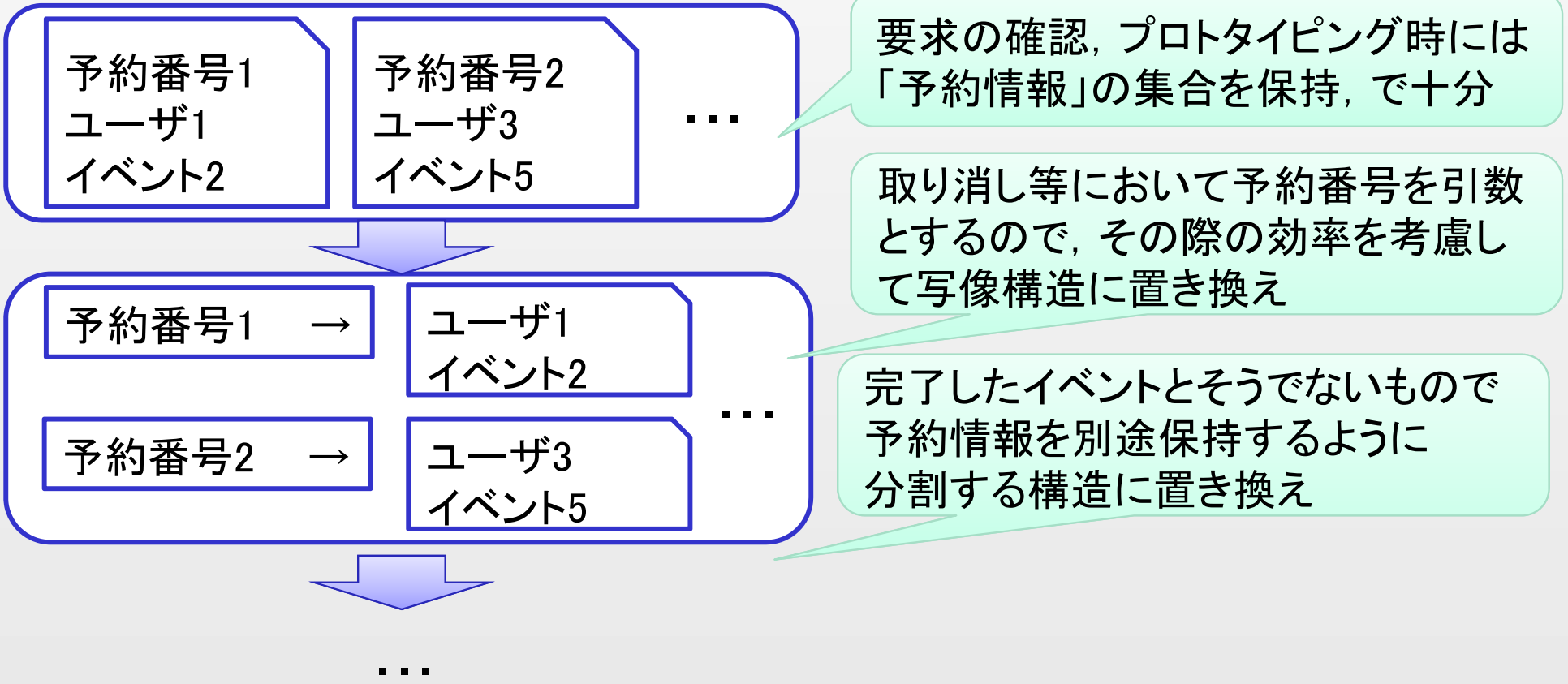
実際には，ユーザ情報として名前，住所等をDBに保持するかもしれない
　しかし，「ユーザのイベントへの登録」機能の明確化・確認・分析の際には「ユーザ1」といった識別情報(トークン)だけで十分

実際には，ループ回数を少なくするなど走査アルゴリズムを決めるかもしれない
　しかし，開発の早い段階では「すべてのユーザがこの条件を満たすか調べる」という，(手順ではなく)効果の説明で十分

実際には，特定の順序での頻繁なアクセスなどを考慮して，データ構造(メモリへの配置と操作アルゴリズム)を選ぶかもしれない
　しかし，開発の早い段階では「システムの利用登録を行ったユーザすべてを保持する」(抽象データ型：集合)というだけで十分

モデル化・記述(３): 段階的詳細化

- 以前の段階で決めたこと，保証したことを引き継ぎ照らし合わせながら，詳細を加えていく



要求の確認，プロトタイピング時には「予約情報」の集合を保持，で十分

取り消し等において予約番号を引数とするので，その際の効率を考慮して写像構造に置き換え

完了したイベントとそうでないもので予約情報を別途保持するように分割する構造に置き換え

モデル化・記述：本講座で学ぶアプローチ

- VDM：ライトウェイト
  - 比較的プログラムに近い記述
    - レコード，集合，列，写像等様々な抽象データ型
    - オブジェクト指向
  - プロジェクト・開発者の意向に応じて抽象化方針や記述対象を決定
    - 特に複雑なデータ構造や機能のみプロトタイピング？仕様全体の厳密化・明確化？・・・
    - 通常，事前・事後条件によるインターフェース定義（陰定義）と動作定義（陽定義）との二段階の記述

モデル化・記述：本講座で学ぶアプローチ

- B：高信頼性の追求
  - 数学的な抽象概念（集合と関係）に基づいた記述
  - 仕様から正しいプログラムを得る段階的詳細化
    - 最初のモデルはすべての機能を含み正しい前提
    - プログラムに落とせるデータ構造やアルゴリズムに「言い換えていく」
- Event-B：より上流過程の難しさに対応
  - 仕様を順次構築していく段階的詳細化
    - 構成要素を順次追加していく

目次

- 本講座の位置づけ
- 形式仕様記述の考え方 – 復習
  - モデル化・記述
  - 検証・妥当性確認
- 代表的な適用事例紹介

検証・妥当性確認(V&V)

成果物を正しく作っている？

- 検証(Verification)
  - 成果物が，定められた基準に照らし合わせて正しい(correct)かどうか(正当性)
  - 基準を満たすかどうかを調べるテスト，モデル検査，定理証明など

意味ある成果物を作っている？

- 妥当性確認(Validation)
  - 成果物が，そもそもの意義・意図を反映した，妥当(valid)なものとなっているかどうか(妥当性)
  - 実装またはプロトタイプを用いた受け入れテスト，人手でのレビューなど

検証・妥当性確認(１): 対象となる性質

- 形式仕様記述中で明記する条件
  - 不変条件: システムの状態が常に満たさなければならない制約
  - 関数・操作の事前条件:関数や操作の実行前に成り立っていなければならない条件
  - 関数・操作の事後条件:関数や操作の実行後に成り立っていなければならない条件

「履修追加」操作の事前条件
「追加しようとする講義と同時間の講義をすでに履修登録していない」

不変条件
「ある学生が同じ時間帯に複数の講義を履修登録していることはない」

「成績上位者取得」操作の事後条件
「戻り値は，Aの数が5個以上の学生を，成績順に整列したものになっている」

講義管理システム

検証・妥当性確認(１): 対象となる性質

- 検証対象となる性質

  「様々な」状態で「様々な」入力に対して「様々な」関数や操作(の列)を呼び出したときに,

  - 不変条件は常に成り立っているか？
    - 違反の例: 事前条件の漏れや操作記述の誤り
  - 操作の呼び出しが起きるときには事前条件が成り立っているか？
    - 違反の例: 条件を満たさない呼び出し記述
  - 操作の呼び出し後に事後条件が成り立つか？
    - 違反の例: 操作記述の誤り

検証・妥当性確認(２): 検証の手段

- 「観測」による検証: テスト，モデル検査
  - テスト: 設定した各ケースにおける状態変化をインタプリタし，性質が成り立つかどうか確認
  - モデル検査: 可能な状態変化を網羅的に探索，性質が成り立つかどうか確認

ケース1:
ユーザ1がイベント1に登録
→ ユーザ1がイベント2に登録
（時間重複で拒否されるはず）

ケース2:
. . .

...

検証・妥当性確認(２): 検証の手段

- 「法則」による検証： 定理証明

```
private int x := 0;
inv x >= 0;

public withdraw : int ==> ()
withdraw(a)
  == x := x − a
pre a <= x
```

不変条件
「預金額は常に0以上」

「引き出し」操作の事前条件
「引き出し額は預金額以下」

この事前条件を満たすような呼び出しであれば，この操作の実行後に不変条件が成り立たない状態に到達することは決してない

a <= x ならば x − a>= 0 なので
（数学の基本的な公理より）

検証・妥当性確認(３): 妥当性確認

- 妥当性確認の手段
  - インタプリタによる実行が可能なら，それを通して受け入れテスト等を行うことができる
    - 実行可能な文法に限る

      「整数の中から1000個の素数を選べ」のような文は厳密に書いてあっても一般的に実行できない
  - 検証対象とすべき性質を，自動的にすべて生成するような機能もある
    - 検証対象となる性質の妥当性を一部保証
    - 妥当な不変条件などを人間が与えた後の話

検証・妥当性確認：本講座で学ぶアプローチ

- VDM（VDM Toolbox）：ライトウェイト
  - 実行，テストによる検証・妥当性確認
    - GUI等のプログラムとの連携が可能
  - ツールによる証明課題生成（参考程度）
- B（Atelier B）：高信頼性の追求
  - 定理証明器による（半）自動定理証明による検証
  - ツールによる証明課題生成（証明必須）

注：講義時間内では現時点での代表的なツールを紹介しているが，実際にはある言語に対し様々なツールが考えられる

目次

- 本講座の位置づけ
- 形式仕様記述の考え方 – 復習
- 代表的な適用事例紹介

適用事例紹介

- VDM
  - フェリカネットワークスの事例が最近では世界的にも代表的
    - その他，オランダのオークションシステムなど
- Bメソッド
  - 鉄道制御（自動運転，プラットフォームドア制御）に関するシステムに多く使われてきている
    - パリ，香港，ニューヨーク，（最近も追加）・・・

VDM適用事例：参考文献

- 仕様書の記述力を鍛える～モバイルFelica開発における形式仕様記述手法の導入事例～
  - 栗田太郎，日経エレクトロニクス2007年2月12日号
- 携帯電話組み込み用モバイルFelica ICチップ開発における形式仕様記述手法の適用
  - 栗田太郎，情報処理学会学会誌2008年5月
- その他，国際会議ICFEM 2008でのチュートリアル（日本人向け）など

VDM適用事例: 導入目的

- 社会的基盤となるシステムの開発の効率化・安定化
  - 厳密な仕様を策定
  - 自然言語・UMLによる概念仕様，VDM++による形式仕様を多方面から分析，精査するプロセスを策定，導入
  - ソフトウェア実装や複数のICチップのそれぞれと組み合わせながら仕様を評価する環境を構築
  - 意識，考察，コミュニケーションを促進

VDM適用事例：適用方法

- 適用対象
  - 外部仕様をVDM++にて記述
    - 異なる種類の内部実装への入力になる
    - 共通仕様として参照されるため，信頼性を確保することが重要である
  - ➡（各実装ごとの）設計・実装担当および評価担当それぞれへのアウトプット

VDM適用事例：適用方法

- 適用プロセス
  - 外部仕様をVDM++にて記述
  - 設計・実装担当
    - 外部仕様に基づき内部仕様，実装を作成
    - 外部仕様へのフィードバック
  - 評価担当
    - 形式仕様の評価
    - 各実装の評価
    - 評価仕様自体の評価
    - 外部仕様，各実装へのフィードバック

VDM適用事例：適用方法

- 活用のための環境構築
  - VDM++フレームワーク，利用ルール
    - VDM++のわかりづらいところを隠蔽
    - デザインパターン，ユニットテストなど一般的なオブジェクト指向のノウハウを活用
  - 異なる動作実現に対して同じテストを行えるような評価環境
    - VDMによる実行可能な仕様
    - ソフトウェアによる実装
    - 実際の様々なICチップ

VDM適用事例：適用方法

- その他
  - VDM記述からのコード生成機能は，生成されたコードが速度などの観点で組み込みに適さなかったため利用せず
  - カバレッジ評価を用いたテストを行った
    - 82%
    - 加えてブラックボックス評価やランダム評価など

VDM適用事例：規模

- 3年3か月
- 50～60人でのプロジェクト（平均30歳程度）
- VDM++約10万行
- 対応するC/C++実装約11万行
- 対応する自然言語仕様書677ページ

VDM適用事例：効果

- 「仕様不明確」ということが不具合の原因となったのは不具合のうち1.8%のみだった
  - 「仕様理解不足」が10.7%，仕様を「読ませる」ことが課題となった
  - 「記述が曖昧で意図が不明瞭である」という質問よりも「仕様に書いてあるがわからない」という質問が増えた
    （「深く読まざるを得ない」のでわかった気になれない？）
- 仕様の評価工程において，ある実行パスでの事後条件エラーなど，レビューで見つけられなかった不具合を発見できた

VDM適用事例：その他ポイント

- 仕様を「読ませる」ために
  - 位置づけ説明，ガイド・フォロー，関連ドキュメント
  - 策定の背景・経緯がコメントに必要
  - 適切なレベルの記法やテクニック
- 仕様もプログラム開発同様のメンテナンス対象
- 「仕様が決まっていること」と「要件が決まっている・変わらないこと」を混同しないこと
  - 仕様策定期間は長くなる
- 実装フェーズにならないと、実装者は仕様を批判的に読み始めない

VDM適用事例：その後

- 「第2世代」開発の取り組み
  - 仕様と設計（WhatとHow）の分離
    - 陰仕様と陽仕様の明確な使い分け方針
    - 例： 仕様を「読む」のは陰仕様
    - 例： VDM記述により必要以上に設計を制約しないように（データ型やエラーのチェック順序など）
  - I/Fだけでなく機能仕様も記述するための方針
  - 可読性
    - コメントではなく，日本語によるVDM++記述を
      - 以前はVDM++記述とコメントの不整合があった

B適用事例：参考文献

（1）パリ地下鉄の自動運転システム

Météor: A successful application of B in a large project

- Matra Transport International, FM’99

（2）パリ空港の自動運転シャトル

Using B as a High Level Programming Language in an Industrial Project: Roissy VAL

- ClearSy & Siemens, ZB 2005

まとめ： Formal Methods in Industry: Achievements, Problems, Future（J. R. Abrial, ICSE 2006）

B適用事例(１): 背景と狙い

- 地下鉄の自動運転システム
  - 高い安全性が要求される
  - 以前も形式手法を利用していた
    - 機能要求を形式言語で記述
    - ソースコードに対応するPre/Postアサーションを追加，半自動証明
      （これらの間の対応関係は人が確認）
  - ➡重いプロセス・弱い自動化であったが，
    形式手法の適用を次の事例でも要求することに
    （注：鉄道企業側の要求）

B適用事例(１): 背景と狙い

- 安全性に関わるソフトウェア部分が適用の対象
  - 設計エラー・コーディングエラーを取り除くためにBメソッドを適用
    - ハードウェアエラーなどを吸収するような実行環境などを用意した上で
  - 360msごとに周期実行される逐次実行ソフトウェア(シングルタスク，割り込み不可)が対象
    - 「Bメソッドの扱う正当性」の範囲

B適用事例(１): 手法・ツールの状況

- 当初はかなりひどかった
  - 型チェック不完全，コード生成なし，・・・
- B手法・ツール自体の実用化に4年
  - 開発チーム，検証チームの役割分担を含めた，Bを用いる新しい開発プロセス
  - 双方のチーム向けの方法論リファレンス
  - ツール性能(モデルの規模・自動証明)の向上
    (例： 自動証明の割合 35% → 80%)
  - 安全なコードコンパイル技術等との連携

B適用事例(１): 開発プロセス

- いわゆるBの開発プロセス
  - 抽象Bモデル
    - 要求文書中の機能に関する情報をすべて含む
  - 具体的なB0モデル
    - ループ構造，データ型など実現方式に関する情報を含む
    - 実行言語・環境上の制約も意識
      （開発者にとっては，B0言語が「実装」言語）
  - コード(Ada)の自動生成

B適用事例(１): 検証プロセス

- 抽象モデルの要求文書に対する検証
  - 自動化できない，もっとも大変なステップ
- Bを用いない部品のインターフェースと照らし合わせての検証
  - 満たす性質を明示化したB部品として記述
    （B適用対象の部分に関する「仮定」となる）
- 追加された証明ルールの妥当性確認
- 生成されたコードの検証
  - 2種類の異なるコード生成器の結果を比較
- 機能テストによる妥当性確認

ここでは（以降の運用でも）バグゼロ

B適用事例(１): 進め方のポイント

- 開発・検証ともに書籍による指針を中心に添えて進めた
  - 例: オートマトンのモデル化手法
  - 例:「アーキテクチャに関する詳細化は最初に」
  - . . .
- トレーニングは開発チーム，検証チームともに2週間ずつ(共通基礎1週間，それぞれの専門1週間)
  - 開発チームの2/3がBのトレーニングを受けた
- Bに詳しいサポートチームを設け両チームを支援

B適用事例(２): 進め方のポイント

- 要求からBへ
  - 要求段階で重要な部分に関してデータフロー図や疑似コードを用い，B抽象機械とのギャップを少なくするようにした
  - 要求からB抽象機械を記述する過程で，質問により多くの曖昧さを排除した
  - 要求からB抽象機械を記述する過程は人手によるため，レビューを実施した
  - 性質をどう形式化するか定かでない部分もあったが，仮の性質記述を定めて証明を試みることによって洗練していった

B適用事例(２): 進め方のポイント

- Bにおけるリファインメント
  - ツールを用いて半自動で行った
    - 集合操作等の抽象的な記述の具体的な記述への変換を自動化
    - リファインメントルールを与えてリファインメントを実行させる
      - 失敗したらルールを修正・追加して繰り返し
      - 実行コードが遅い(線形時間で実行できることを指数時間でやっていた)ことがわかり，ルールを変更し修正した
  - 手動部分も主に「コピー＆変更」でできた

B適用事例(２): 工数データ

- 空港シャトルにおける所要人月の割合
  - 準備: 5%
  - マネージメント: 8%
  - 抽象モデルの構築: 全体の55%
    - うち18%が要求に対する質問，解析
    - うち5%がモデルのレビュー
    - うち16%が証明
  - 具体モデルの構築: 全体の24%
    - うち11%が証明
  - 設定管理，再現，ドキュメント化など: 8%

開発早期に，
かけるべきところに
時間をかけられている

B適用事例：規模

	パリ地下鉄	空港シャトル
生成されたコード（ADA）のステップ数	86000	158000
行われた証明の数	27800	43610
そのうち対話的証明の割合	8.1%	3.3%
対話的証明に要した人月	7.1	4.6