Vol.2017-ARC-225 No.44
Vol.2017-SLDM-179 No.44
Vol.2017-EMB-44 No.44
2017/3/10

MVCアーキテクチャのメタレベル適用による形式仕様モデルに関する考察

張 漢明[1]　野呂 昌満[1]　沢田 篤史[1]

概要：ソフトウェア開発における形式仕様導入の障壁として，形式仕様の難解さと記述のための適切な指針がないことがあげられる．本稿では，可読性の高い統一仕様モデルとそのモデルに基づいた形式記述法について考察する．形式仕様記述のメタモデルを MVC アーキテクチャの概念に基づいて定義し，詳細化関係を考慮した構成要素および要素間の関係を整理する．形式仕様記述法として，形式仕様言語 VDM-SL による宣言的で簡潔な関数スタイルのテンプレートを提示する．自動販売機システムの事例を用いて仕様モデルの妥当性を議論する．

キーワード：形式手法，形式仕様モデル，MVC アーキテクチャ, 詳細化関係，VDM-SL

Consideration on formal specification model by appling MVC architecture in meta-level

Han-Myung Chang[1]　Masami Noro[1]　Atsushi Sawada[1]

Abstract: Lack of appropriate guidelines on describing readable formal specifications becomes a barrier to successful introduction of formal methods to software development practices. In this paper we propose a structural model for highly readable formal specifications. We also discuss a universal method for describing formal specifications based on our specification model. A metamodel of our specification model is designed borrowing the concept of MVC architectural style. We organize based on this metamodel the constituent elements and relationships between them taking refinement relations into account. We present a set of functional style VDM-SL templates to promote declarative and concise descriptions. We also discuss the validity of our specification model using an example of vending machine system.

Keywords: Formal methods, Formal specification model, MVC architecture, Refinement, VDM-SL

1. はじめに

ソフトウェア開発において，仕様は正当性検証の基準として重要である．開発者は開発対象の設計やプログラムが正しいという正当性を仕様に基づいて検証する．顧客は開発対象が顧客の要求を満たしているという妥当性を仕様に基づいて確認する．ソフトウェア開発者は仕様の重要性を認識しているが，仕様を適切に記述することは容易な作業ではない．開発対象で実現する本質を理解して，簡潔かつ正確に抽象化して記述することが求められる．

[1] 南山大学
Nanzan University, Showa, Nagoya 466–8673, Japan

ソフトウェアの信頼性を向上させるために，形式的なアプローチは必要不可欠であり，形式手法は有望な技術である [1]．日本における形式手法導入の成功事例の 1 つとして，Felica IC チップの開発における VDM を用いた形式仕様技術の適用が有名である [4]．実現に依存しない抽象化した厳密な仕様を形式仕様言語を用いて記述し，テストを用いて仕様に対する高い信頼性を得ることが，最終成果物であるプログラムコードの品質向上に寄与することが報告されている．

ソフトウェア開発における形式仕様導入の障壁として，形式仕様の難解さと記述のための適切な指針がないことが

Vol.2017-ARC-225 No.44
Vol.2017-SLDM-179 No.44
Vol.2017-EMB-44 No.44
2017/3/10

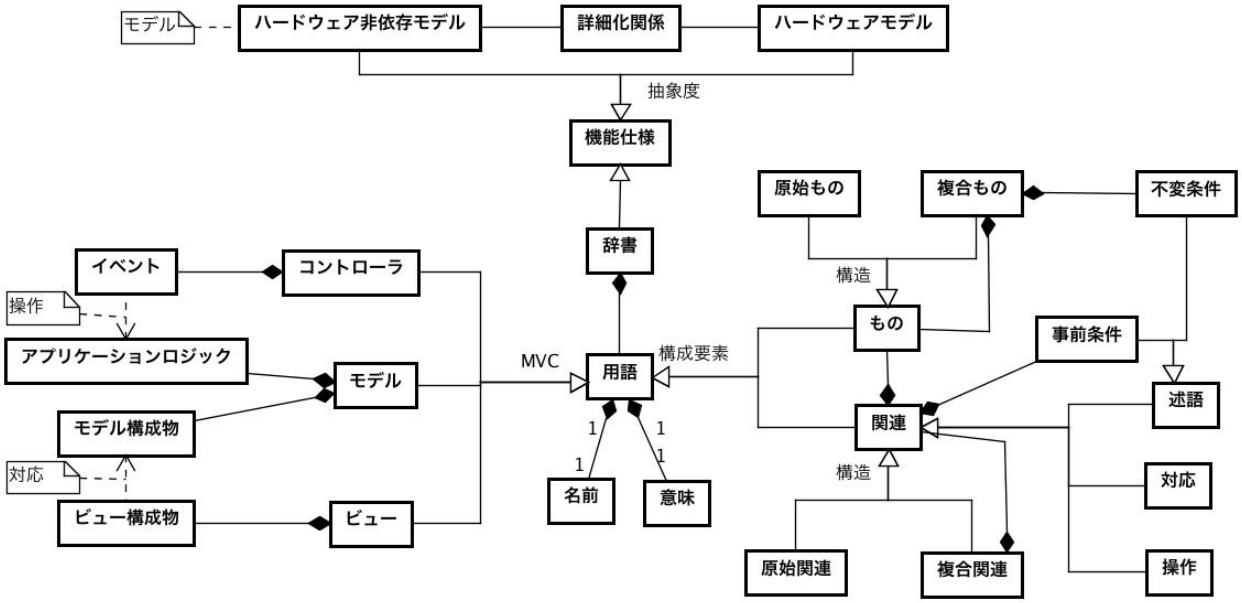

図 1 機能仕様モデル
Fig. 1 A Functional Specification Model

あげられる．形式仕様の難解さの要因として，集合や論理式による記号化された数式の数学的な意味を理解することと，対象概念を数式で表現することの困難さが考えられる．また，プログラムの記述と同様に，数式の書き方が複数あることが，形式仕様の難解さの要因の 1 つとしてあげられる．

本研究の目的は形式仕様を記述するための可読性の高い統一仕様モデルとその記述法を提示することである．形式仕様の可読性を高めることにより，形式仕様が仕様の基盤としての利用価値が高まり，図式表現や自然言語による記述は，直感的な理解や理解を補助する役割を果たす．統一的な仕様モデルとその記述法の提示は，記述法はプロセスを含意しているので，仕様を記述するさいの指針となり得る．また，統一的な記述は可読性の向上にも寄与する．

本研究のアプローチは，MVC アーキテクチャの概念をメタレベルで形式仕様を記述するモデルに適用し，詳細化関係を考慮した構成要素および要素間の関係を整理することである．MVC アーキテクチャはアプリケーションのモデルを規定するが，この概念をメタレベルで形式仕様の機能記述に適用して，機能仕様の書き方を規定するモデルを提示する．MVC アーキテクチャのメタレベルの解釈を以下に示す．

モデル 抽象化された構造とアプリケーションロジックの原始的な記述

ビュー モデルの構成物に対する（出力に関する）別の表現形式の記述

コントローラ イベントによる入力のプロセス記述

機能仕様を「構成要素」,「MVC モデル」,「抽象度」の観点から分類して，これらの 3 つの観点間の対応関係を整理して，形式仕様言語 VDM-SL[8], [9] による記述のテンプレートを提示することを目指す．VDM-SL は実用的で実績のある形式手法 VDM のモデル規範型の形式言語の 1 つである．仕様の記述は宣言的で簡潔な記述を促進する「関数スタイル」を用いる．

ASTER[*1] テスト設計コンテスト’15 における自動販売機の仕様書 [5] を事例として，提案する表記法を用いて VDM-SL 仕様を作成した．構成要素の分類が形式仕様を記述するさいに考慮すべき概念が明確となり，適切なドメイン用語の特定を促進した．適切なドメイン用語が可読性の向上に寄与する．

2. 機能仕様モデル

形式仕様を記述するための可読性の高い統一機能仕様モデルを提示する．機能仕様の要素概念と構造をクラス図で表したものを機能仕様モデルとして図 1 に示す．機能仕様は構造化された「辞書」とみなし，

- 構成要素
- MVC モデル
- 抽象度

の観点から多重分類を用いて整理した．

2.1 辞書

Jackson は文献 [3] おいて「仕様は開発システムの外側にある環境と内側にある機械の間で共有された事象や状態か

[*1] ソフトウェアテスト技術振興協会

らなるインターフェースの定義である」と定義している．本研究では，この仕様インターフェースの定義を対象ドメインの辞書，つまり語彙とその意味を記述することとみなす．図 1 では，

- 辞書は用語の集まりで，
- 用語は名前と意味を定義する

ことを表している．

仕様記述の本質は用語の「名前」と「意味」を定義することである．形式仕様では型や関数などの識別子が名前に対応し，それらの定義が意味に対応する．仕様の形式化は対象ドメインの概念に名前をつけて，その厳密な意味を定義することを強制する．

仕様の可読性は，プログラムの可読性と同様に，変数や関数名すなわち識別子に適切な名前をつけることに依存する．形式仕様記述の困難さは，対象概念を数式で厳密に記述記述することに加えて，対象ドメインの概念を抽出して「概念に適切な名前をつける」という，仕様記述の本質的な難しさを含んでいる．以降で提示する「構成要素」,「MVC モデル」,「抽象度」による分類が対象概念を抽出するさいの指針となる．

2.2 構成要素

機能仕様を形式的に記述するさいの用語の概念を整理したものが「構成要素」による分類である．図 1 では，

- 用語はものと関連に分けられて，
- ものと関連はそれぞれ木構造であり，
- 関連には述語，対応，操作がある

ことを表している．

対象ドメインの概念を抽出するさいには，まず「もの」を特定する必要がある．ものは「原始もの」と「複合もの」から木構造として構成されている．原始ものは分解できない基本構造である．原始ものの定義は，仕様の段階で決定する抽象度を決定する．例えば，仕様のある段階で住所録システムの「住所」の構造を規定しない場合，「住所」を原始ものとする．仕様の段階で考慮する抽象度を明示することにより，仕様の本質的な議論が深まる．複合ものにはものを構成する要素間の「不変条件」が存在する．不変条件は構成要素間が常に満たす条件である．不変条件を設計もしくはプログラムで実現するさいの満たすべき基準として重要である．不変条件は後述で示される述語として表される．

関連は複数のものから構成され「述語」,「対応」,「操作」に分類される．関連は「原始関連」と「複合関連」から木構造として構成されている．原始関連は単体で形式仕様言語の基本構文だけを用いて，他の関連を用いずに定義される．複合関連は他の関連を用いて対象ドメインの用語で定義される．複合関連は用語で構成されるので，自然言語に近い形式で可読性が向上することが期待できる．

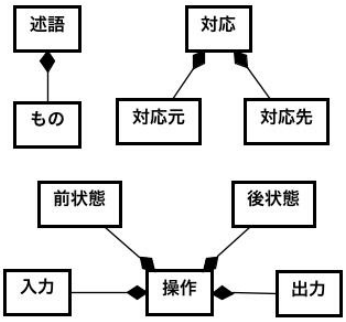

図 2 関連の仕様モデル
Fig. 2 A Specfication Model of Relationship

関連の「述語」,「対応」,「操作」のモデルを図 2 にクラス図を用いて示す．図 2 では，

- 述語は（1 つ以上の）複数のもの，
- 対応は対応元と対応先，
- 操作は前状態と入力とおよび後状態と出力

から構成されることを表している．これらの関連のモデルは，それぞれの構成要素を定義するさいに必要な概念を明示している．

構成要素の分類は，形式仕様を記述するさいに必要な要素を特定している．これらの構成要素はさらに以下の「MVC モデル」と「抽象度」の観点から分類される．

2.3 MVC モデル

組み込みシステムは環境からのイベント（事象）に対して操作および制御して出力するものとして，仕様を「MVC モデル」を用いて構造化するものとする．MVC モデルは対象の構成要素を「モデル」,「ビュー」,「コントローラ」の観点から分割する．図 1 では，

- コントローラはイベント，
- モデルはモデル構成物とアプリケーションロジック，
- ビューはビュー構成物

で構成されることを表している．さらに

- イベントはアプリケーションロジック，
- ビュー構成物はモデル構成物

に依存していことを表している．

コントローラでは，仕様インターフェースとしてイベントを特定し，イベントにアプリケーションロジックに対応させる．アプリケーションロジックは操作を用いて定義される．モデルでは，モデルの構成物を特定しその構成物に対するアプリケーションロジックを操作として定義する．ビューでは，ビューの構成物を特定して，モデル構成物との依存関係を対応を用いて定義する．

2.4 抽象度

大規模なシステム開発では段階的な詳細化の記述と，詳細化関係の記述を明記して詳細化関係にある記述間の追跡

性を確保することが重要である．図 1 では，

- 抽象度はハードウェア非依存モデルとハードウェアモデルに分類し，
- 詳細化関係を用いてハードウェア非依存モデルとハードウェアモデル間の関係を定義する

ことを表している．

ハードウェア非依存モデルは，MVC モデルの「モデル」に相当する．組込みシステムの本質をハードウェアに依存しない形式でモデルとして特定する．その後，ハードウェアモデルを特定し，

- モデルをハードウェアモデルで詳細化，
- イベントとビュー構成物を定義

することで段階的に詳細化する．この詳細化は「詳細化関係」として定義することにより，詳細化の「追跡性」を確保する．

3. VDM-SL による記述法

機能仕様モデルの構成要素に対して形式仕様言語 VDM-SL による関数スタイルののテンプレートを提示する．

3.1 構成要素

構成要素として「もの」と「関連」があり，関連には「述語」,「対応」,「操作」の 3 種類がある．

3.1.1 もの

ものは「原始もの」と「複合もの」で構成され，全てのものに対して型付けを行うことにより，ドメインで共有するものの概念を明示する．

原始もののテンプレートを以下に示す．

```
types
  もの名型 = 基本データ型
```

原始ものは分解できない基本構造で，数値型や文字型などの VDM-SL の基本データ型を用いて定義する．仕様の段階で詳細なデータ構造を未定義にする場合は token 型を用いることにより明示する．

複合もののテンプレートを以下に示す．

```
types
  もの名型::
    構成物名 1: 構成物 1 型
    構成物名 2: 構成物 2 型
       :
    構成物名 n: 構成物 n 型
functions
  INV_複合もの:複合もの型 -> bool
  INV_複合もの (複合もの) ==
    定義
```

複合ものは VDM-SL のレコード型を用いて定義する．レコードを構成する識別子（タグ名）に構成物名（ものの名前）とその型を対応させる．そして構成物間で常に保持すべき条件を「不変条件」として後述の述語を用いて定義する．

3.1.2 関連

関連にももものと同様に「原始関連」と「複合関連」がある．原始関連は他の関連を用いずに，VDM-SL の構文要素だけで定義される基本的な関連である．一方，複合関連は関連を用いて定義される．複合関連はドメインの用語を用いて定義されるので可読性が向上する．

述語

述語のテンプレートを以下に示す．

```
functions
  述語名:『もの型列』-> bool
  述語名 (『もの列』) ==
    条件定義
```

述語は引数として対象とするものをとり，ブール型の値を返す関数として定義する．『もの列型』と『もの列』はそれぞれ長さが $n(n > 0)$ のもの型の列およびものの列

```
もの 1 型 * もの 2 型 *...* もの n 型
もの 1 * もの 2 *...* もの n
```

である．

対応

対応のテンプレートを以下に示す．

```
functions
  対応名:『対応元型列』-> 対応先型
  対応名 (『対応元列』) ==
    対応定義
```

対応は引数として（複数の）対応元をとり，対応先を返す関数として定義する．『対応元型列』と『対応元列』はそれぞれ前述の『もの型列』と『もの列』である．

操作

操作のテンプレートを以下に示す．

```
functions
  操作名:『入力型列』* 対象物型-> 対象物型
  対応名 (『入力列』, 前状態) ==
    後状態定義
types
  対象物型:: ... 出力: 出力型
```

操作は引数として（複数の）入力と操作対象の前状態をとり，対象物の後状態を返す関数として定義する．出力がある場合はその出力を対象物型の構成要素に含める．これは，逐次的な操作を操作 A(操作 B(対象物)) のように関数適用で記述できるようにするためである．

3.2 MVC モデル

対象システムの仕様は前述の「もの」と「関連」を用いて定義するが，組み込みシステムの仕様インターフェースとして，イベントとビューに着目する．イベントはアプリケーションロジックに依存し，ビューはモデル構成物に依存する．これらの記述のテンプレートを以下に示す．

```
types                              functions
  自動販売機型::                     販売:商品型 * 自動販売機型 -> 自動販売機型
    商品集合: 商品集合型             販売 (商品, 自動販売機) ==
    商品 map: 商品 map 型              let
    預かり金: 金額型                     販売後 = mu(自動販売機,
    残高:     金額型                        商品 map |-> 商品 map の在庫更新 (商品, -1, 自動販売機. 商品 map),
    金庫:     金額型                        残高 |-> 自動販売機. 残高 - 自動販売機. 商品 map(商品). 価格)
    出力:     [出力型];                 in
  出力型::                                 if 販売可能な商品がある (販売後)
    商品: [商品型]                         then
    釣銭: [金額型];                          mu(販売後, 出力 |-> mk_出力型 (商品, nil))
  商品型 = token;                          else
  商品集合型 = set of 商品型;                mu(販売額を格納 (販売後), 出力 |-> mk_出力型 (商品, 販売後. 残高)
  商品情報型::                         pre 販売可能 (商品, 自動販売機);
    価格: 価格型
    在庫: 在庫型;
  商品 map 型 = map 商品型 to 商品情報型;  価格型 = 金額型; 金額型 = nat;  在庫型 = nat;
```

図 3 自動販売機のハードウェア非依存モデルの記述例
Fig. 3 An Example of Hardware Independent Model of Vending Machine

ビュー

ビューは「対応」を用いて以下の形式で記述する．

```
functions
  ビュー名:モデル構成物型 -> ビュー構成物
  ビュー名 (モデル構成物) ==
    ビュー構成物への対応定義
```

ビューでは，モデル構成物を用いてビュー構成物への対応を定義する．

コントローラ

コントローラは「操作」を用いて以下の形式で記述する．

```
functions
  イベント名:『入力型列』* 対象物型 -> 対象物型
  イベント名 (入力列, 対象物) ==
    let 入力列 x = 変換 (入力列, 対象物)
    in アプリケーションロジック (入力列 x, 対象物)
```

コントローラでは，イベントの入力列をアプリケーションロジックの入力列に変換して，アプリケーションロジックを適用する．アプリケーションロジック（モデル）は本質的な概念として最も重要であり，コントローラの定義においてイベントの意味をアプリケーションロジックを用いて定義する．

3.3 抽象度

本研究では，抽象度は「ハードウェア非依存モデル」と「ハードウェアモデル」との関係として捉える．ハードウェアモデルでは，

- ハードウェア非依存モデルで記述した MVC モデルにおける「モデル」をハードウェアモデルで詳細化し，
- 「ビュー構成物」と「イベント」を特定して，

これらの間の関係を「詳細化関係」として記述する．「ビュー構成物」と「イベント」の定義は，上述の MVC モデルにおける「ビュー」と「コントローラ」を用いて定義する．

「モデル」の詳細化は以下の対応を用いて記述する．

```
functions
  モデル構成物対応:ハードウェア構成物型 -> モデル構成物型
  モデル構成物対応 (ハードウェア構成物) ==
    対応
```

この対応の定義が，ハードウェア非依存モデルとハードウェアモデルのインタフェースを規定する．ハードウェアモデルを定義するさいに抽象度の高いハードウェア非依存モデルとの対応を明記することにより，抽象度間の記述の追跡性を確保することが可能になる．

4. 考察

4.1 形式仕様の可読性の向上

開発対象の構造は「複合もの」としてレコード型で定義され，機能は「操作」として「複合関連」を用いて定義される．ここで用いられる型と関連の「名前」が開発対象の用語を規定する．複合関連は「述語」と「対応」を用いて VDM-SL の簡潔な基本演算子で表現される．ここで用いられる演算子は，レコードの値を変更するためのレコード修正子（mu），条件式，局所変数を導入する let 式 などの簡潔な基本演算子である．

自動販売機のハードウェア非依存モデルの記述例を図 3 に示す．自動販売機の構成物の構造を「自動販売機型」として定義し，アプリケーションロジックの例として「販売」を定義している．自動販売機の構成は，自動販売機型として，商品集合，(商品の情報を表す) 商品 map，預かり金，残高，金庫で構成され，商品と釣銭を出力することが定義されている．販売は，販売する商品の在庫と残高を更新し，販売可能な商品がない場合は，販売額を格納して残高を釣銭として排出することが定義されている．また，販売可能な（事前）条件は以下の述語で定義されている．

```
販売可能:商品型 * 自動販売機型 -> bool
```

```
販売可能（商品，自動販売機）==
  在庫あり（商品，自動販売機）and
  販売可能残高（商品，自動販売機）
pre 商品 in set 自動販売機.商品集合;
```

「在庫あり」と「販売可能残高」は原始関連として厳密に定義される．

「複合もの」および「複合関連」で定義された仕様は，ドメインの用語と VDM-SL の簡潔な構文要素で表現されるので可読性が高くなる．「原始もの」と「原始関連」は VDM-SL の構文要素を用いて厳密に定義される．仕様書の可読性は開発対象の「ものと関連を抽出して適切な名前をつける」ことに帰着され，形式仕様における数学的な表記の難しさは「原始もの」と「原始関連」の定義に帰着される．

4.2 ハードウェアモデルによる詳細化

ハードウェアモデルの記述は，

- ハードウェアの「もの」と「操作」および
- ハードウェア非依存モデルとの対応

を定義する．

自動販売機のハードウェアとして，販売ボタン，商品機器，金銭機器を想定して，前述の自動販売機型に以下の構成要素を追加する．

```
販売ボタン集合：販売ボタン集合型
販売ボタン map：販売ボタン map 型
商品機器：商品機器型
金銭機器：金銭機器型
```

販売ボタンは，購入する商品を指定するボタンで，商品機器は商品を格納する複数のラックから構成され，金銭機器は紙幣と硬貨を扱う機器から構成される．

商品機器を例として，ハードウェアとハードウェア非依存モデルの対応を考える．商品機器は在庫と関連しているので，この対応を「商品在庫の対応」として定義する．

```
商品在庫の対応:自動販売機型 -> 商品 map 型
商品在庫の対応（自動販売機）==
  {商品 |-> mu(自動販売機.商品 map(商品),
    在庫 |->
      商品機器にある商品在庫（商品，自動販売機.商品機器））|
        商品 in set 自動販売機.商品集合}
```

商品の在庫は自動販売機型の「商品 map」で定義されているので，商品在庫の対応は「商品 map 型」を返す関数として定義する．

販売におけるハードウェアの操作を「商品排出」として定義すれば，販売の記述ではハードウェアの操作を

```
商品機器 |-> 商品排出（商品，自動販売機.商品機器）
```

として追記し，図 3 における販売記述の商品 map の記述を

```
商品 map |-> 商品在庫の対応（自動販売機）
```

に変更すれば，ハードウェアモデルとハードウェア非依存モデルの対応が明記される．

5. おわりに

本稿では，形式仕様を記述するための可読性の高い統一仕様モデルと形式仕様言語 VDM-SL によるテンプレートを提示し，自動販売機の事例に適用した結果を議論した．形式仕様記述の困難さは，数式による対象概念の厳密な記述と可読性を向上するための適切な名前を定義することにある．提案する仕様モデルは，対象概念の数学的な表現を「原始的な」原始ものと原始関連に集約し，「複合的な」複合ものと複合関連では，条件式や更新式などのの簡潔な基本構文とドメインの用語を用いて簡潔な記述を促進する．

MVC アーキテクチャを適用することにより，モデルの本質的な記述に対して，コントローラをイベントによる入力のプロセス記述，ビューをモデルの構造物に対する別の表現形式の記述とみなし，機能仕様の書き方を規定することを試みた．コントローラの記述は入力の仕様とアプリケーションロジックを分離し，ビューの記述はモデルに対して適切な別の表現形式を提供する．

提案した機能仕様モデルを有効に利用するために，構成要素と構成要素間の関係を分析する必要がある．今後の課題として，

- 構成要素の型づけと
- 構成要素間の関係のパターン化

があげられる．

謝辞 本研究の一部は，JSPS 科研費 24220001，16K00110，2016 年度南山大学パッヘ研究奨励金 I-A の助成を受けて実施した．

参考文献

[1] Hall, J.A.: Seven Myths of Formal Methods, IEEE Software, 7(5): 11-19 (1990).
[2] IEEE Standard Board: IEEE Recommended Practice for Software Requirements Specifications, Std 830-1998 (1998).
[3] Jackson, M.: Software Requirements and Specifications, ACM Press (1995).
[4] 栗田太郎：携帯電話組込み用モバイル FeliCa IC チップ開発における形式仕様記述手法の適用，情報処理，Vol. 49，No. 5，pp. 506.513 (2008).
[5] ASTER 自動販売機ハードウェア構成および販売者用機能仕様，http://aster.or.jp/business/contest/doc/2015tdc-v1_1.zip.
[6] Spivey, J.M.: The Z Notation, Prentice Hall (1992), http://spivey.oriel.ox.ac.uk/ mike/zrm/zrm.pdf.
[7] 来馬啓伸：B メソッドによる形式仕様記述，近代科学社 (2007).
[8] Fitzgerald, J. and Larse, P.G.: Modelling Systems, Cambridge, (2009).
[9] 荒木啓二郎，張漢明：プログラム仕様記述論，オーム社 (2002).