5A-05

形式的仕様を段階的に作成するためのシステムのモデル化技法

山田隆弘†

宇宙航空研究開発機構宇宙科学研究所†

1. はじめに

ソフトウェアの仕様を形式的に記述する形式手法と呼ばれる方法[1][2]が注目を集めている。形式手法を実際のシステム開発に適用して成功したという事例もいくつか報告されている[3]。しかし、実際のシステム開発の現場においては、形式手法の導入はそれほど広まってはいないように思われる。形式手法が現場で広まらない理由の一つは、従来の開発手法の中に形式手法をいきなり組み込もうとしてもスムーズにはいかないことが多いからであると思われる。形式手法を開発の中にスムーズに組み込むためには、開発の初期の段階から形式手法の導入を意識して開発を行う必要がある。

筆者は、人工衛星などの大規模システムに対して形式手法を適用する方法に関して検討を行い、人工衛星を制御するためのシステムを単純化した仮想的なシステムに対する要求仕様を形式手法言語である VDM++ [2]を用いて記述してみた[4]。しかし、従来のシステム開発で行われている作業と VDM++による仕様記述との間にはかなりの乖離があることが判明した。従って、特別な工夫を行わない限り、形式手法が開発の現場に浸透することはないと思われる。

本論文では、開発の現場で形式手法を容易に活用できるようにするために、実際に行われているシステム開発の方法と親和性のある形で形式手法を導入するための方法を提案する。具体的には、モデルと形式仕様とを試験（テスト）計画に基づいて段階的に作成するという方法を提案する。

2. 形式仕様とモデル化

システム開発を行うときには、システムの構造や振る舞いを厳密に記述するためにモデルを作成することが多い。しかし、実際のシステム開発においては、作成されたモデルとシステム仕様との関係がはっきりしないことが多い。多くの場合、モデルは仕様を解説するための道具として使われ、製造や試験（テスト）などの下流の行程でモデルが参照されることはほとんどない。

しかし、モデルは仕様を厳密に表現するための道具であるべきであり、試験も「モデルが製品として正しく実装されたか」という観点で行われるべきである。また、形式仕様を VDM++等の言語を用いて記述する場合、形式仕様で参照される概念はすべてモデルの中で規定されている必要がある。従って、モデルは「形式仕様や試験計画を作成するための共通のレファレンス」として作成すべきである。

A Modeling Method for Generating Formal Specifications of Systems in a Step-by-Step Manner

† Takahiro Yamada, Japan Aerospace Exploration Agency, Institute of Space and Astronautical Science

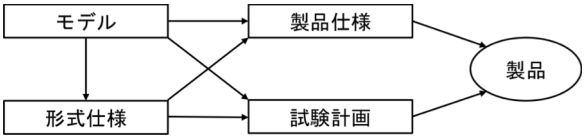

図１　本稿で提案するシステム開発技法

形式仕様は、製品の仕様（特に要求仕様）の基礎となるべきものであるが、それは同時に試験（特にブラックボックス試験）で検証されるべきことの規定にもなっている。逆に言えば、試験で検証されるべき事項を開発の初期の段階で規定したものが形式仕様であるとも言える。

本稿で提案するシステム開発技法を図１に示す。まず、システムのモデルと形式仕様とを作成する。形式仕様で参照される概念はすべてモデルで規定されていなければならないので、モデルの作成と形式仕様の作成とは同時に行われるべきである。

システムに対する要求仕様は、[4]でも述べたように、システムが提供すべきインタフェースとして規定すべきである。ここで言うインタフェースは、UML や Java で規定しているインタフェースの概念と同一であり、そのインタフェースを通じて起動できるオペレーションとそのシグネチャ（オペレーションに対する入力変数と出力変数の集合）を規定したものである。各々のオペレーションに対する要求を規定するために、オペレーションのシグネチャとともにオペレーションの事前条件（オペレーションを実行する前に成り立っているべき条件）と事後条件（オペレーションを実行した後に成り立っているべき条件）とを規定する。

事前条件と事後条件の設定は、「試験で何を検証すべきか」という観点で行われるべきである。すなわち、事前条件は、各々のオペレーションの試験を実施するときに整えておくべき条件として表される。また、事後条件は、各々のオペレーションの試験を実施した後で成立しているべき条件として表される。これらの条件は、そのまま粗いレベルの試験計画になっている。

また、事前条件と事後条件で参照される概念（例えば、オブジェクトのアトリビュート）は、すべてモデルの中で（例えば、クラス定義の中のアトリビュート定義として）規定されているべきである。逆に言うと、形式仕様で参照される概念（アトリビュート等）をシステム要素として体系的に規定したものがモデルであると言える。

製品に対する仕様（ソフトウェア設計仕様など）は、モデルと形式仕様を詳細化することによって作成される。試験計画もモデルと形式仕様を別の観点から詳細化することによって作成される。

開発は表面的には図１に示した順序で行われるが、頭の中では、試験計画→形式仕様→モデル→製品仕様の順序で進むことになる。

3. 段階的モデル化

一つのシステムは、多くの場合複数のコンポーネントより構成される。このような場合、まずシステムに対する形式仕様を作成すべきである。システムに対する形式仕様は、上で述べたように、システムに対する試験（テスト）で何を検証すべきかに基づいて設定すべきである。また、形式仕様で参照される概念はシステムのモデルとして規定する。

システムに対する形式仕様とモデルが完成したら、それをどのようにコンポーネントに分解すべきかを考えながら、コンポーネントの形式仕様とモデルを作成する。コンポーネントに対する形式仕様は、そのコンポーネントの単体試験で何を検証すべきかに基づいて設定すべきである。また、形式仕様で参照される概念はコンポーネントのモデルとして規定する。

VDM++等の既存の形式手法では、事後条件としてアトリビュートの値を指定することが多い。オペレーションを実施した結果としてアトリビュート値が変化する場合は、このように事後条件を設定できるが、そのように事後条件を設定できない場合もある。例えば、コンポーネント A があるオペレーション X の実行結果として別のコンポーネント B のオペレーション Y を起動するような場合を考える。この場合は、オペレーション Y の実行結果はコンポーネント B の責任であり、コンポーネント A の責任ではない。従って、コンポーネント A の単体試験では、オペレーション Y を起動したことが確認できれば十分である。このような場合は、オペレーション X の事後条件として以下のように記述することにする。

```
post invoked ComponentB`OperationY
```

4. 実例

ここでは、上で述べた方法を適用した例として図２に示す熱制御システム(Thermal Control System, TCS)を考える。このシステムの目的は、ヒータを使って温度の測定値を設定値の上下２度以内に制御することである。システム全体に対するオペレーションとしては、設定値を設定するためのもの(SetTemp)と温度を制御するためのもの(ControlTemp)とを設ける。これらのオペレーションに対する形式仕様は、システム試験で検証すべきことに基づいて設定される。また、形式仕様で参照される概念はシステムのモデルとして規定する。システムに対するモデルと形式仕様を VDM++で記述したものを以下に示す。

```
class TCS
instance variables
 measuredTemp: int
 desiredTemp: int
operations
 public SetTemp: int ==> ()
  SetTemp(Temp) == is not yet specified
  postDesiredTemp = Temp
 public ControlTemp: () ==> ()
  ControlTemp == is not yet specified
  post MeasuredTemp < DesiredTemp+2 or
   MeasuredTemp > DesiredTemp-2
```

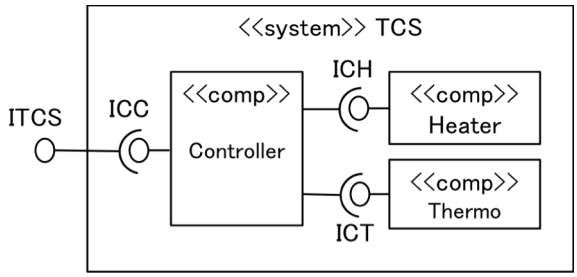

図２　Thermal Control System (HCS)の構成

次に、このシステムをコンポーネントに分解する。ここでは、制御のコンポーネント(Controller)、ヒータのコンポーネント(Heater)、温度計測のコンポーネント(Thermo)の三つを考える。これらのコンポーネントに対する形式仕様は、単体試験で検証すべきことに基づいて設定される。Controller コンポーネントに対するモデルと形式仕様とを VDM++で記述したものを以下に示す。

```
class Controller
instance variables
 desiredTemp: int
operations
 public SetTemp: int ==> ()
  SetTemp(temp) == is not yet specified
  post desiredTemp = temp
 public ControlTemp: () ==> ()
  ControlTemp == is not yet specified
  ext rd Thermo`measuredtemp: int
  post
   if Thermo`measuredTemp > desiredTemp+2
   then invoked Heater`TurnOff
   elseif Thermo`measuredTemp <
          desiredTemp-2
   then invoked Heater`TurnOn
```

5. おわりに

本稿では、モデルと形式仕様とを試験計画に基づいて段階的に作成する方法を提案した。形式仕様というと特別に作成すべきものと考える開発者が多いが、試験計画は必ず作成すべきものであり、試験計画から形式仕様を作成するという考えは多くの開発者に受け入れられるものと考えられる。今後は、ここで提案した方法を実際の人工衛星開発に適用したいと考えている。

参考文献

[1] 荒木, 張: プログラム仕様記述論, オーム社 (2002).
[2] Fitzgerald, J., et. al.: *Validated Designs for Object-oriented Systems*, Springer (2005).
[3] 情報処理推進機構: 厳密な仕様記述における形式手法成功事例, 調査報告書 (2013).
[4] 山田: 大規模システムに対する形式的要求記述の一方法, 情報処理学会第 77 回全国大会、2B-02 (2015).