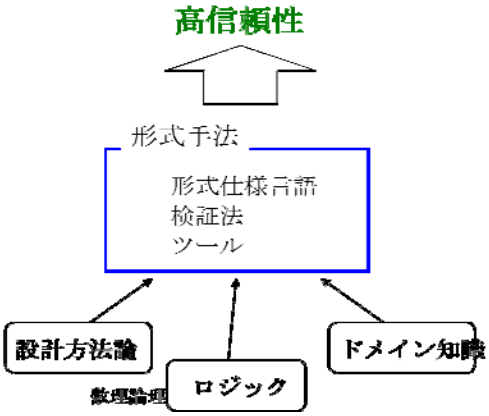
ソフトウェアシンポジウム２０１０

WG3「形式手法モデリング」討議のまとめ

WG3 討論リーダ：
中島　震（国立情報学研究所）
佐原　伸（(株)CSK システムズ）

１　はじめに

形式手法はシステムに求められる信頼性達成への技術アプローチとして、産業界からも大きな期待を集めている。従来の入門的な解説記事や教科書では数理論理に基づく開発技術という側面が強調された。数理論理が重要な技術要素であることは事実であるが、実際は、ソフトウェア工学で論じられてきた設計方法論やドメイン知識に関する技術を含む総合的な技術である。形式手法の中心である形式仕様言語を用いることで、対象を適切な抽象レベルで簡潔に表現することが可能になる。表現が簡潔であることから、逆に、システム開発に際して本質的なモデリングの問題が浮かび上がるといってよい。



本WGでは、形式手法とモデリングの関係について、参加者の形式手法適用経験を基に、３つの観点を強く意識して議論を行った。すなわち、

（１）　ソフトウェア工学で論じられているモデリングと形式手法の関係、
（２）　形式手法適用に必要なモデリングの方法（各技法固有、形式手法共通)、
（３）　形式手法はモデリングに新しい規範を導入するか、

である。WGでの議論は多岐にわたったが、次の点が前提として了解された。

（１）　形式手法は思い込みを排除した客観的かつ繰り返し利用可能な一般性の高い記述を得る技術の総称である。
（２）　先人の成果を正しく理解し、その知見の上に、自分たちの技術を積み上げる。
（３）　きちんと定義した、あるいは合意のとれた、用語をベースとした議論が必要で

ある。自分勝手な用語定義は議論を混乱させるだけである。

（４）　システム開発の上流工程を論じる際には、M. Jackson による問題フレームの考え方が有効と期待できる。

以下、ＷＧの進め方、議論の要点を報告し、最後に形式手法の今後の方向性を整理する。

２　ＷＧの進め方

本ＷＧの企画として、荒木啓二郎教授（九州大学大学院）に以下の題目による招待講演をお願いした。

「システム開発の現場でのフォーマルメソッド適用に向けての課題と方策」

ＷＧへの参加希望者には事前にポジションペーパを提出して頂き、議論の方向性を事前に整理する方法をとった。適用経験や技術的な話題を単に紹介するだけではなく、そこから得られた知見を一般的な形に整理して頂くことをお願いした。便宜上、３つのセッションに分け、ポジションペーパ投稿者による話題提供に引き続き、報告された知見や提起された課題についての議論を密に行った。３つのセッションは以下の通りである。

（１）　セッション１：モデリング（司会：中島）

（２）　セッション２：パターンとイディオム（司会：佐原）

（３）　セッション３：導入・適用の推進（司会：石黒）

また、セッション１に先だって、招待講演（荒木教授）を踏まえて、討論リーダから本ＷＧの討論主旨について説明した。

ＷＧでは、ポジションペーパ投稿者（付録にリストを示す）の他に、聴講だけのオブザーバ参加を許可した。その結果、３つのセッションを通して平均２０名の方々がＷＧに出席された。なお、発表者１３名のプロファイルは、大学・研究機関３名、産業界６名、社会人学生３名、大学院学生１名、である。ソフトウェアシンポジウムの狙い通り、産学のバランスの良い構成になったといえる。

３　ＷＧ議論の要約

３－１　用語の正しい理解

発表ならびに質疑応答の中で、いくつかの技術用語についての定義、理解、解釈について議論となった。以下、話題になった（物議をかもした）用語のリストを示す。

モデル、モデリング、アーキテクチャ、ドメイン、リファインメント、

不変量、抽象化、ポリシー

これらは、未定義のまま使用されることが多い。同じ言葉であっても、使用する文脈によっては異なる実体を指すことがある。また、具体的な議論をする場合には、より詳細な定義を明確化することが大切な用語もある。「曖昧さを排除しよう」という形式手法に関心を持つ場合、使用する用語の定義に細心の注意を払うことは当然であろう。逆に、上記の基本的な用語を不用意に使うようでは、形式手法の技術に関わる資格が欠如していると云われてもしょうがない。

上記の用語は、ソフトウェア工学や形式手法の分野で論じられてきている。そのような先人の知見の上に、自らの議論を展開することが望まれる。すなわち、過去の成果をきちんと「勉強」しなければならない。当時の議論を振り返ることで、「何が発見され、何が現在まで生きており、何が消えていったか」を理解できる。これによって、不要な「再発見」や思い込みによる不毛な議論を避けることができる。

３－２　作業プロセスとスケッチ

教科書での説明と実際の作業プロセスには大きなギャップがある。教科書では、最終的に得られた形式仕様記述が、あたかも天下り的に得られたかの如く、当たり前のものとして提示されることが多い。一方、プログラム作成の場合を引き合いに出すまでもなく、最終的な形式記述が突然、生み出されることは決してない。個人的な作業であっても、プログラミングに先だって、何らかの事前作業を行って作成するプログラムの見通しを得るだろう。さらに、よほど単純なプログラムでない限り、「コンパイル一発、テスト一発」ではなく、「デバッグ」という作業を経る。さらに、デバッグはノウハウ的な側面が強く、明文化されることは少ない。

形式仕様の場合であっても、事前の見通しを得る作業、デバッグに相当する作業が含まれる。後者は、用いる形式仕様言語や支援ツールによって大きく異なるノウハウ的な側面が大きい。そこで、本WGでは、前者の事前作業について議論した。これを「スケッチの問題」と呼ぶことにする。議論の要点を以下にまとめる。

（１）　スケッチの必要性

形式仕様作成の前段階で、作成対象に関する大まかなイメージを整理する試行錯誤過程が必要である。試行錯誤であるから、記述の厳密性よりは、表現の柔軟性と書きやすさに力点がおかれる。多くの場合、ダイアグラムであろう。いわゆる、Box and Arrow の世界である。逆に、形式仕様言語はその厳密さの故にスケッチには向かない。

（２） スケッチで表現する内容

スケッチで何を表現するかは用いる形式仕様言語と、その言語が「自然に」提供するモデリング視点が影響する。たとえば、VDM++による実行可能仕様を得ることが目的の場合、従来からのオブジェクト指向モデリングで用いるクラス図やコラボレーション図の観点をスケッチとして書き表すことが良いだろう。また、Correct-by-Construction に重点を置く Event-B では初期仕様のスケッチだけではなく、最終的に得たい仕様のスケッチならびにリファインメント過程のスケッチ（リファイメント計画といってもよい）を明確にイメージする必要がある。

スケッチ考え方は事前に見通しを得るという意味で、形式手法を用いる場合だけに登場するものではない。F.P. Brooks, Jr. は Global Design, Incremental Building という言葉を使っている。Waterfall スタイル開発への批判として、Start Small による繰り返しスタイルによる逐次開発の方法が有効であることがわかっている。しかし、最初に全体の設計がないままに開始した逐次構築は、「スパゲッティデザイン」に陥る。構築開始時点で、全体像を見通す必要がある。これを、Global Design と呼んだ。

構築の手段である言語がプログラミング言語であるか形式仕様言語であるかは不問である。とにかく、全体設計（Global Design）が欠如しているところでの開発は失敗するだろう。特に、リファインメントベースの方法では、リファインメント過程で「迷子」になっては何をやっているのかわからなくなる。

（３） スケッチと生産物

スケッチは捨て去るものであろうか、あるいは、形式仕様記述と共に保守管理して、システム開発過程の生産物として取り扱うべきであろうか。これについては現実からの論点がある。すなわち、たとえ２つとはいえども、スケッチ（ダイアグラム記法）と形式仕様記述の双方を長期間にわたって保守管理していくことは難しい。現実にはスケッチから開始するが形式仕様記述だけが生産物となる、という考え方である。

一方、J. Wing が提唱した Parallel Iterative Development と呼ぶ開発スタイルがある。これは、Waterfall スタイル開発への批判を含むが、スケッチを出発点として、ダイアグラム等による従来型仕様書、形式仕様記述、プログラム開発、テストなどの複数の工程を行き来しながら、同時並行的に開発を進めるものである。ここでは、従来型仕様書と形式仕様記述の双方が開発生産物となる。ＷＧでは、

「プログラミングを第三者に委託する場合、従来型仕様書が必須」という議論もあった。

スケッチの問題（Global Design を含む）は、形式手法を用いた開発スタイルと強く関連することから、形式手法を展開していくための重要なノウハウとなる。

３－３　要求仕様と多重世界

形式手法の技術は４０年にわたる歴史の中で、２つの流れとして実用的な技術になってきていることは良く知られた事実である。すなわち、開発上流工程の技術（Correct-by-Construction）と最終生産物であるプログラム検査技術（Posteriori Verification）である。後者は自動検証技術のプログラム品質保証への応用である。本WGでは開発上流工程における形式手法の役割について論じた。

開発上流工程で大きな問題となっているのは、いわゆる要求獲得の技術である。システムに関わる多様なステークホルダが合意した結果が要求仕様である。要求仕様を形式手法によって表現し解析することで、開発上流工程からの品質確保を目指す。一方、ソフトウェア工学から派生した要求工学の分野では、過去２０年にわたって、要求仕様の技術に関する知見を持つ。要求仕様として表現すべき内容すべてを形式仕様言語が書き表すことは難しい。いわゆる非機能要件（ilities）は形式化と相性が悪い。機能的な要求に議論を限定することになる。

機能的な要求に限定する場合であっても、要求工学の標準的な考え方と、現状の形式手法適用技術とにはギャップがある。要求工学では、先に述べたように、要求を利害対立するステークホルダの妥協の産物と考える。そのため、互いに矛盾する要求を同時に表現し、いずれか一方を採用した場合の論理的な結果について比較し議論する。すなわち、矛盾を含む多重世界の表現が基礎となる。一方、伝統的な形式仕様の世界では、書き表した形式仕様記述は矛盾してはならない。一貫した整合性（consistency）を示すことを要請する。

もうひとつの側面は、開発過程で、事前に要求を洗い出せるかという問題がある。これを「予測（anticipation）の問題」と呼ぶ。たとえ完全に要求定義を行い、システムの信頼性を保証できたとしても、開発時の予測あるいは仮定が運用時に壊れる可能性がある。M. Jackson は「any useful computing system interacts with the world outside the computer」とまとめ、開発時の技術に限定される限り形式手法の限界を指摘している。

WGでは、矛盾を含む表現体系の必要性、ならびに、運用時に関わる実行時監視への形式手法適用の可能性について議論を行った。また、トレーサビリティ（追跡性）について、

非形式的な仕様から形式仕様へ、要求仕様から設計仕様へ、といった観点から議論があった。今後、さらに考えていくべきテーマである。

３－４　抽象化

形式手法と「抽象化（abstraction）」は密接な関係にある。「抽象」あるいは「抽象化」とは、「事物または表象のある側面・性質を抜き離して把握する」ことで「おのずから他の側面・性質を捨象する」ことである。形式手法は、プログラムのような実現詳細から特定の側面を抜き出し簡明に表現する。これによって、システムが持つ論理的な性質の把握ならびに「正しさ」の確認を容易にする。すなわち、システムの本質を把握する。

上記の言葉の定義からもあきらかなように、抽象化について論じる際、対象の「何を」、「何のために」抜き離すのか、同じことであるが、何を捨象するのかを明確にする必要がある。すなわち、「合目的な抽象化」という論点が大切であり、目的・観点・方法を具体的に論じなければならない。

ソフトウェア工学の分野では従来から、システムの複雑さを乗り切るために、様々な抽象化の方法が考案されてきた。たとえば、階層アーキテクチャは下位機構を隠蔽するという抽象化を提供する。また、コンポーネントやオブジェクト指向概念は内部の実現方法を外部から隠蔽し、自身の機能振る舞いをインタフェースに抽象化している。では、これらの抽象化技法と形式手法に求める抽象化とは何が同じで、何が異なるのであろうか。そのためには、抽象化の目的を明確化することが前提となる。以下、いくつかの目的と具体例を示す。

（１）　流用性

作成した形式仕様記述が１回限りの用途ではなく、繰り返し流用可能な資産となることを指す。特定のシステムに特化した記述は他への転用が難しい。同様な機能を持つ一連のシステム（プロダクトダインと呼ぶ）に共通の性質を、抽象的なシステムとして整理することが好ましい。対象問題領域の背景知識（アプリケーション・ドメインと呼ぶことがある）も流用性という観点で有用である。

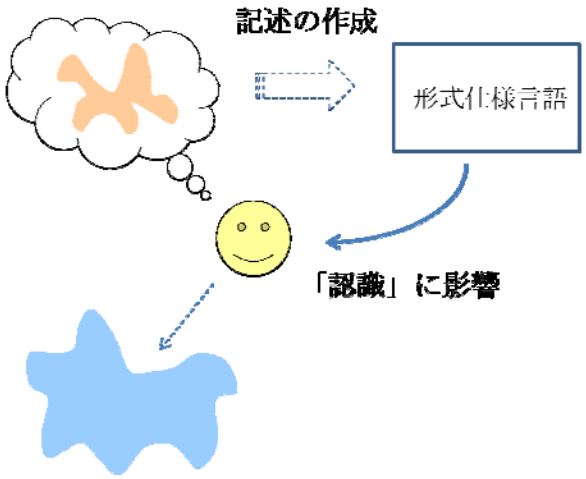
（２）　過大近似（Over-approximation）

不具合発見についての完全性達成を目的として、システムの振る舞い仕様（可能な経路の集まり）を抽象化する方法が知られている。特に、有限状態システムを対象とするロジック・モデル検査は、抽象化支援検証（AAV = Abstraction Aided Verification）と呼ばれるくらい、抽象化の方法と密接である。

（3） 情報隠蔽

上に述べたコンポーネント等と同様に、外部とのインタフェース記述によって、コンポーネント内部の実現方法を隠蔽する。代数仕様言語が最初に目指した抽象化の方法である。

WGでは、もうひとつの観点として、「形式仕様は記述であり、記述手段を与える言語が全てである」という議論も出た。形式仕様を作成するのは人間であり、記述内容は、人間が持つシステムの把握能力に制限される。すなわち、人間が持つ抽象化能力がフィルターとなる。さらに、人間は用いる記述手段あるいは言語によって、その思考や認識の方法が影響を受ける。



たとえば、SPIN を用いる場合、通信オートマトンの考え方で、対象システムを眺める。状態遷移系に基づく操作的な計算の仕組みに基づく視点が抽象化の基盤となる。一方、Event-B では対象システムを、どのような集合によって分割するか、集合要素間の関係は何があるかといった視点が抽象化の中心をなす。

抽象化の問題を一般的に論じることは得策ではない。抽象化によって、何をしたいのか、を明確にし、合目的に具体的な方法を考えていくべきである。同時に、目的に合致した形式手法を適宜採用することが大切である。

4 おわりに

本WGは短い時間の中で非常に密な議論ができた。このような集まりにつきものの、「なかなか理解して貰えない」とか、「否定的なリアクションばかり受ける」等の愚痴に近い「ガ

ス抜き」議論が皆無だった。また、３－１で述べたような用語について自らの思い込みによる私見を提示する議論がなかった。これは、参加者が形式手法の技術に習熟していることの裏付けであり、そのおかげで、有益な議論を集中的に行うことができた。

もうひとつ、関連する集まりとの大きな違いは、個別形式手法の比較優劣に関する議論がなかったことをあげることができる。これも、参加者の多くが、既に、複数の形式手法を使った経験があり、「適材適所」という考え方になれていたからであろう。実際、「エンコードを工夫」すれば表現の対象を広げることが可能である。問題は、「エンコード」によって複雑さをもたらさないか、である。

最後に、日本にいる我々にとって形式手法は欧米発の「渡来品」である。本家の欧米では、多数の形式手法が、歴史的にライバル関係にあり、国際学会は呉越同舟の様である。一方、日本では、三国伝来の七福神が和気藹々と乗った宝船のようである。どれか１つを選ぶというより、宝船と一緒に進んでいくほうが大きな御利益を得られる。すなわち、様々な形式手法を目的に応じて適材適所で使っていけばよい。これが日本の知恵であろう。

付録　ポジションペーパ発表者リスト（発表順、敬称略）

中津川泰正（九州大学大学院／ソニー）
橋本祐介（総合研究大学院大学／ＮＥＣ）
田端一也（ＮＴＴデータ）
佐原伸（(株)ＣＳＫシステムズ）
石黒正揮（三菱総合研究所）
青木利晃（北陸先端科学技術大学院大学）
飯田周作（専修大学）
木村浩一郎（法政大学大学院）
只野賢二（総合研究大学院大学／フェリカネットワークス）
北村崇師（産業総合技術研究所）
酒匂寛（デザイナーズデン）
早水公二（メルコ・パワー・システムズ）
岡本勝幸（(有)イットワークス／ビジネススペースひろしま）