特集 フォーマルメソッドの新潮流 5

Part III：新領域の開拓

# フォーマルメソッドの新展開
## —検証進化可能電子社会の中核技術—

二木厚吉 ● 北陸先端科学技術大学院大学情報科学研究科

## はじめに

フォーマルメソッドへの現場からの期待が高まっている．フォーマルメソッドはソフトウェアそのものとほぼ同じ長さの歴史を持ち，現場からの期待の度合いも時代とともに変化してきたが，何度目かにあたる最近の期待の高まりは主に以下の2点に起因している．

(1) 自動車の車載ソフトウェアに代表されるように，日常生活に浸透しかつ人命に直接かかわるソフトウェアが複雑化し，その安全性や信頼性を確保するためにはフォーマルメソッドが必要となってきた．

(2) 世界規模の情報ネットワークが重要な社会基盤となり，ソフトウェアが，工業製品だけでなく，我々の生活を支えるさまざまな社会・行政システムやビジネスシステムに深く浸透するにつれて，問題領域・要求・設計のモデル化・記述・解析にフォーマルメソッドを用いる必然性が高まってきた．

フォーマルメソッドは新しい展開を迎えており，多くの技術者が，フォーマルメソッドの基本的な特質を理解し，その可能性と限界を自らの業務に基づき判断することが重要となりつつある．

**形式化の効用と限界：** フォーマルメソッドはformal methodsをカタカナで綴った訳語である．形式手法という訳語も定着してきた．ほかに形式的方法，形式的手法といった訳語もあるが，「形式的」という日本語が「内容がなく形式に堕している」というコノテーションが強いので，「形式化」の力を積極的に評価する立場のformal methodsの訳語としては形式手法またはフォーマルメソッドが適切であろう．以下では形式手法という訳語を用いることにする．「形式（form）」という言葉・概念は，「形式は物事の本質を客観化した形」であり，また同時に，「形式は物事の本質を切り落とした形だけのもの」である，という二面性を持つ．この二面性は，理論と実践，科学と技術，基礎と応用などの間にある緊張関係と同質のものであり，時代や分野などに依存して色々な形で現れるが，時には深刻な対立関係に発展することもある．形式手法は薫り高い研究成果を誇るが役には立たないという批判は，システムの信頼性や安全性を確保する中核技術としての形式手法への期待が高まる現在でも，たびたび繰り返される．

**汎用科学技術としての形式手法：** 形式手法の定義には，「数学に基づくソフトウェア開発法」，という非常に一般的であいまいなものから，「仕様，設計，プログラムなどの正しさを数理論理学（mathematical logic）に基づき厳密に検証できるソフトウェア開発法」，といったより限定的なものまで，さまざまなものがある．形式手法の基本的な要件を何に求めるかについては，研究者の間にさえ共通の認識があるとはいえない．形式手法に関する最も包括的なWebサイトであるFormal Methods Virtual Library（http://www.afm.sbu.ac.uk/）には96に上る記法，方法，ツール（notations, methods, tools）が掲載されている☆1．形式手法は，ある人にとってはSPINシステムによるプログラムコードのモデル検査であり，また，ある人にとってはCafeOBJ言語システムやRAISE言語を使った問題領域（domain）や要求（requirement）のモデル化と記述であるといったように，広範な領域をカバーしつつある．形式手法は，ハードウェア，ソフトウェア，ハイブリッドといった計算機科学技術が対象とするシステムだけでなく，生物システムや社会システムなどの広範なシステムを対象とする分野横断的な汎用科学技術（generic science and technology）に発展しつつある．

本稿の目的は，形式手法の新しい展開の可能性と限界を客観的に評価するために必要と思われる基本事項を解説することである．まず，形式手法の現在までの発展の軌跡を概観し，形式手法の特質について述べる．次に，形式手法の新しい展開の例として，検証進化可能電子社

☆1 本稿の以下の記述に現れる方法，言語，ツールの詳細に興味を持たれる読者はこのWebページを参照されたい．

会の実現を目指す取り組みを紹介する．最後に，形式手法の将来を展望し，技術者教育の重要性を指摘する．なお，本稿は参考文献 4）と 5）を基にして必要な改訂を施したものである．

## 形式手法の発展の軌跡とその特質☆2

形式手法の最も一般的な定義は「計算機科学において培われてきた数学的な手法に基づく科学的なシステムの構築法」とすることができる．計算機の誕生が「数学的な計算（数値計算）を機械で行う」ことに動機付けられていることを考えれば，計算機とそれを動かすソフトウェアの発展は数学的な手法に裏打ちされたものであった，と言うことができる．しかし，情報ネットワークの急速な進展に伴い，ソフトウェアシステムとして実現される機能は多様化し，数値計算に代表される「計算」はそのごく一部になってしまい，ソフトウェアシステムは我々の生活に必要なあらゆる機能を実現することが期待されるようになっている．これに伴い，ソフトウェアシステムの科学的な構築を支援する数学的な手法も，純粋な計算を支援するものから，複雑なシステムの構築に必要な論理的な推論や検証を支援するものへと変容を遂げてきた．以下では，現状の形式手法への発展を可能とした基盤技術を概観することで，形式手法の特質について述べる．

### ●プログラム検証技術

プログラム検証のアイディアは電子計算機の誕生当初の早い時期からあったとされる．電子計算機上で動作し人間が望む仕事を自動的に行うプログラムは，人類にとってまったく未知の新しいものであり，その動作の正しさを納得するためには数学的な検証が必要である，と当事者が考えたであろうことは容易に推測される．単一プロセッサから多プロセッサへ，単一処理から多重分散ネットワーク処理へと計算機システムが複雑化するのに対応して，プログラム検証技術はソフトウェア技術の中核技術として発展を遂げている．プログラム検証技術はいまだに活発な研究活動が展開されており，その可能性を結論的に述べるのは難しいが，以下の点を指摘しておく．

**プログラム検証の完全な自動化はできない：**プログラム検証の完全な自動化の問題は，原理的に機械には解けない問題（決定不能問題；unsolvable problem）を含むので，すべてのプログラムを自動的に検証するプログラムを開発することは不可能であると認識すべきである．応用領域を限定したり，プログラムの形を制限したりすることである程度の自動化が期待できるが，プログラムの自動検証は決定不能問題と遭遇する可能性を常にはらんでいる，という事実を前提にすることが大切である．

☆2 個々の具体的な形式手法の最近の動向については，文献 3）が包括的な良質の解説である．

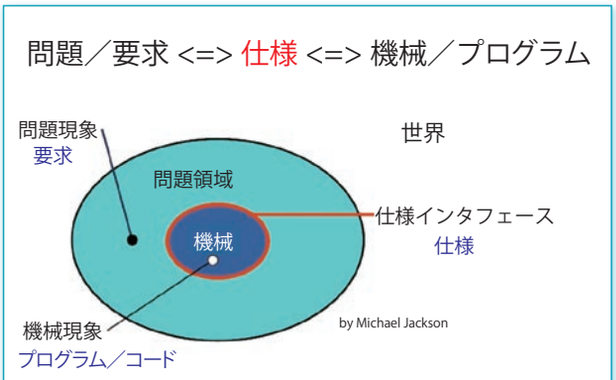


図 -1　インタフェース／契約の定義としての仕様

**プログラム検証は人間が適切な仕様を与えることで意味を持つ：**プログラム検証はプログラムが仕様のとおり動作していることを示すことであり，検証の自動化の可能性は，ユーザがどのような種類の仕様をどのように表明するかに依存して決まる．プログラム検証の自動化を考える以上に，適切な仕様の与え方を定めることが重要である場合が多い．プログラム検証のための仕様は，図 -1 に示されるとおり，問題（あるいは要求）と機械（あるいはプログラム）のインタフェース（あるいは両者の間に成立する契約）を定義するものとするのが適切である．図 -1 は Michael Jackson によるが，この考え方は Bertrand Meyer の「契約によるプログラミング」などと同じである．

**正しいプログラムを仕様から自動生成する「生成による検証（proof by generation）」の技術が有望な分野が多い：**プログラムの開発に先立ち仕様を適切に与えることが，システム開発全体の効率を高めるためには重要である．コーディングの前に仕様が開発されているのであれば，仕様から正しいプログラムを自動的に生成するほうが，コーディングの後でプログラム検証を行うよりは効率的であり，より高い信頼性を獲得できる可能性が高い．MDA/MDD などは，こうした洞察に基づく開発方法論である．

### ●形式仕様言語

モデルや仕様を記述するための言語としては，実務家の間では UML（Unified Modeling Language）がデファクトとして受け入れられつつある．しかし，UML に形式手法を適用し検証を行おうとすると，UML に厳密な意

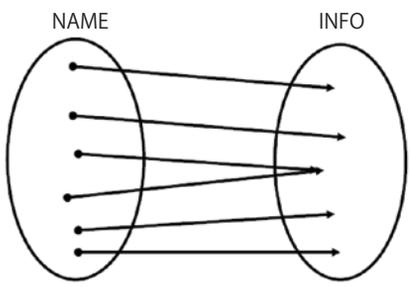


図 -2　VDM, Z におけるデータ辞書のモデル

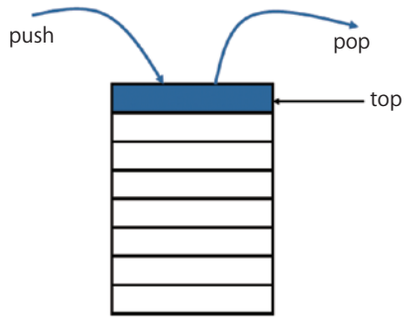


図 -3　OBJ, CafeOBJ におけるスタックのモデル

味を付加する必要があり，UML の形式化（または UML の数学的な意味定義）の研究が盛んになってきた．

形式手法の前提となる数学的な意味定義を伴った仕様言語（formal specification languages）の研究開発は，UML などの図式言語の研究開発とは独立に行われてきた．以下がその代表的なものである．

VDM，RAISE，Z，B：集合，写像，関数といった数学的な概念に基づき，システムの仕様を記述する形式仕様言語．数学的な意味は型付けされた一階述語論理（typed first order predicate logic）を用いて与えられる．これらの仕様言語はモデルベース形式仕様言語として分類されるが，これは集合，写像，関数などの数学モデルを用いてシステムの仕様を定義する立場からきている．たとえば，各々の名前にどのような情報が付与されているかを記憶したデータ辞書は，図 -2 のように，写像としてモデル化され，それに基づき仕様が定義される．VDM と Z は，ソフトウェアシステムの仕様を数学的に記述するための言語として，最も早い時期に設計開発された形式仕様言語であり，当初は仕様記述のための数学記法（mathematical notation）であり，計算機上で実行可能ではなかった．VDM と Z は，仕様記述者とプログラム開発者の意思疎通の記法として，いくつかの実プロジェクトで使用された実績を持つ．最近では，検証システムとの連携により仕様の検証を行おうとする研究が盛んに行われている．

OBJ，CafeOBJ，Maude，CASL：システムの仕様を，ものの集まり（ソート（sort）と呼ばれる）とその上に定義されたオペレータ・演算・関数（operator，function）を基本単位として記述する形式仕様言語．ものを多種類に分けて分類することで型付けが導入され，多ソート代数（many sorted algebra）が仕様の数学的なモデルとなることから，代数仕様言語（algebraic specification languages）と分類される．たとえば，LIFO（Last In First Out）型のデータ構造であるスタック（Stack）は，図 -3 のように，push，pop，top という 3 つの演算でモデル化され，それに基づき仕様が定義される．仕様の数学的な意味は仕様が定義する代数を規定することにより与えられる．オブジェクト指向の先導概念である抽象データ型（abstract data type）の形式仕様言語としてスタートし現在では動的システムをも含む一般的なシステム用の形式仕様言語に発展している．OBJ，CafeOBJ，Maude は仕様の意味を規定するために等式（equation）だけを用いる立場を基本とし，その等式を書換システムと解釈することで実行可能仕様を実現している点が，VDM，Z などの実行可能性を前提としない形式仕様言語と異なる．実行可能性を有することで検証システムを自然に内包した仕様言語システムが実現できるという大きな特徴がある．

## ●形式手法

形式手法を，個別の手法やツールの集合体としてではなく，統一的な分野として捉えるためには，「形式仕様言語による形式仕様の開発が形式手法の中心にある」とする見方が適切と思われる．この立場では，形式手法の目的をシステムの信頼性や安全性のできる限り客観的な解析・検証であるとし，それを達成するための形式手法の基本要件を以下の 2 点とすることが多い．

(1) 意味が数学的に定義された形式言語によりシステムのモデルが記述される．
(2) その記述が数学的に解析・検証される（数理論理学などに基づく論理的な解析・検証法がある）．

ただし，上記(2)の要件は，「解析や検証を伴わないものは形式手法としない」ということではなく，「解析や検証に耐えるように意味が定義された仕様言語で形式仕様を作成することが形式手法の基本である」と理解すべきである．

**計算機科学と形式手法**：形式手法は，数学的なモデル化

法と解析法に基づく手法であるが，それはまた計算機科学，特にプログラムとソフトウェアに関する理論や技術から発展したものである．すなわち，以下のような理論や技術は，従来の数学の対象ではなかったが計算機科学の中で発展し，それらに基づきシステムの汎用的なモデル化，記述，解析・検証を目的とする形式手法が整備されてきたといえる．

- プログラムとプログラム言語の理論と技術：アルゴリズムの概念が定式化されその理論が生まれたのは1930年代である．1940年代にコンピュータが発明され，それ以後コンピュータを正しく効率的に動かすための理論と技術として，手順や手続きを厳密に記述・解析するためのプログラムとプログラム言語の理論と技術が発展し，形式手法の基礎を形成した．
- 分散アルゴリズムや並行分散システムの理論と技術：複数の主体から構成されるシステムを記述し解析するための理論と技術．特に，分散システムのモデル化・記述法として発展してきた，遷移システム，並行プロセス，プロセス代数などの理論と技術は形式手法の重要な基盤である．
- 形式仕様と汎用システムモデルの理論と技術：システムの要件や仕様を数学的な表現(つまり式)として記述しそれを解析・検証するための形式仕様の理論と技術は形式手法の中核を成す．ソフトウェアのオブジェクトモデルやUMLに代表される汎用的なシステムのモデル化法などは，形式化に距離を置いて発展してきたが，最近は形式仕様・形式手法の考え方を積極的に取り入れようとする動きが活発である．

## ●システム検証技術

検証技術はシステム技術一般に広く普及し多様化している．ハザード（hazard）や失敗の可能性または事後の解析技術として発展してきた失敗木解析（fault tree analysis）の技術，統計的・確率的な信頼性解析技術などのいわゆる信頼性工学の手法には，分野に特化したものも含めて，多くの手法・方法が存在する．ソフトウェアが最終的には論理的・数学的な構造物であるという前提のもとに，主にソフトウェア技術の分野で発展してきた検証技術として，定理証明（theorem proving）とモデル検査(model checking)がある．

**定理証明技術**：定理証明という名前が示すように，「数学の証明を計算機上で実行する」という動機からスタートした技術である．数学の証明の過程を形式化しそれを計算機で支援させるシステムがいくつも開発されている．こうした研究の成果として，100年以上のあいだ未解決で1976年に初めて解決された4色問題が，定理証明システムCoqを使って検証されたというニュースが2004年末に話題になった．数学的に厳密な検証を計算機で支援させ，できる限り自動化を図るという研究は，計算機科学における最も挑戦的なテーマの1つである．定理証明システムは，対話型検証システム，証明支援システムなどと呼ばれることもあり，ケンブリッジ大学・ミュンヘン工科大学のIsabelle/HOL，フランス国立研究所・パリ大学のCoq，スタンフォード研究所のPVSといったシステムが有名である．数学的な検証過程を計算機で支援するので，以下で説明するモデル検査技術のような「検証すべきシステムを有限状態システムとしてモデル化する必要があるという制限」はない．システム検証技術において定理証明技術を役立たせるためには，(1) システム開発に役立つ十分に厳密なモデル化・仕様記述技術，(2) 数学の定理証明ではなくシステム検証向きの証明過程の定式化とシステム化，(3) システム検証向きのユーザインタフェースの開発，などの問題を解決する必要がある．こうした課題の克服を目指して，CafeOBJやMaudeなどの実行可能形式仕様言語システムを発展させて，システム検証向きの対話型検証システムが研究開発されている．

**モデル検査技術**：モデル検査の「モデル」は，通信ソフトウェアや論理回路(半導体回路)の検証技術から発展してきた「有限状態モデル」を意味する．モデルという言葉は万能の意味合いが強いので，モデル検査におけるモデルが有限状態モデルであるという事実は注意を要する．モデル検査技術は，現場で使われ多くの現実の問題を解決しているという意味において，現時点で最も成功した検証技術である．有限状態モデルに限定することで，一般的な検証では困難なLTL（Linear Temporal Logic）やCTL（Computation Tree Logic）といった時制論理式(temporal logical formula)で記述された性質の検証も自動化できるという特徴を持つ．またBDD（Binary Decision Diagram）などの巧妙なデータ構造を採用することで，大きな状態集合を持つシステムを合理的な時間内で検査できるようにした点も，大きな技術的なブレークスルーであった．モデル検査がもたらした技術的な成果は大きいが，(1) 有限状態モデルでは対応できないシステム検証問題は多く存在する，(2) 現状のモデル検査システムでも対応できないほど大きな有限状態を持つシステムも多く存在する，(3) モデル検査は反例の発見には有効であるが検証・証明には限界がある，といった諸点は認識しておいたほうがよい．モデル検査と対話型検証を統合したより強力な検証技術を開発することが，当面の最重要な技術課題である．

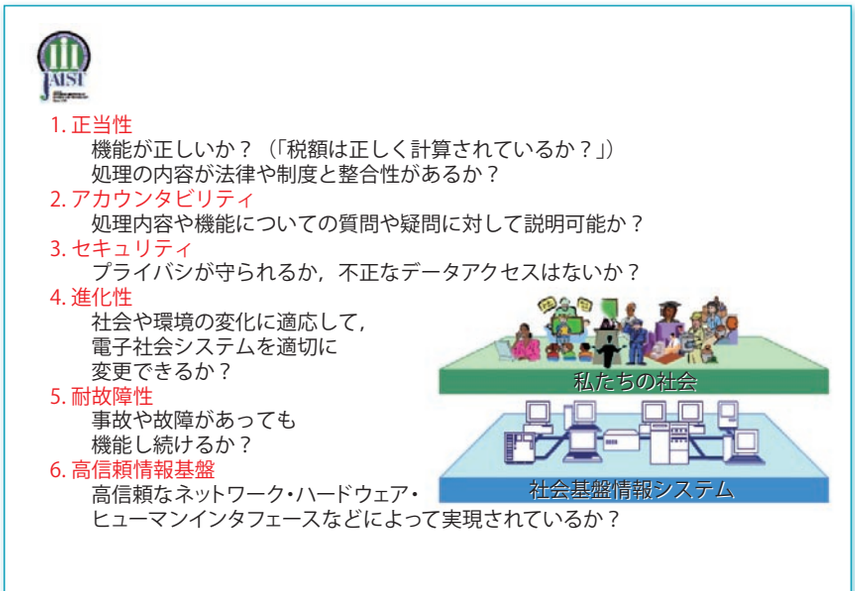


図 -4　検証進化可能電子社会の要件

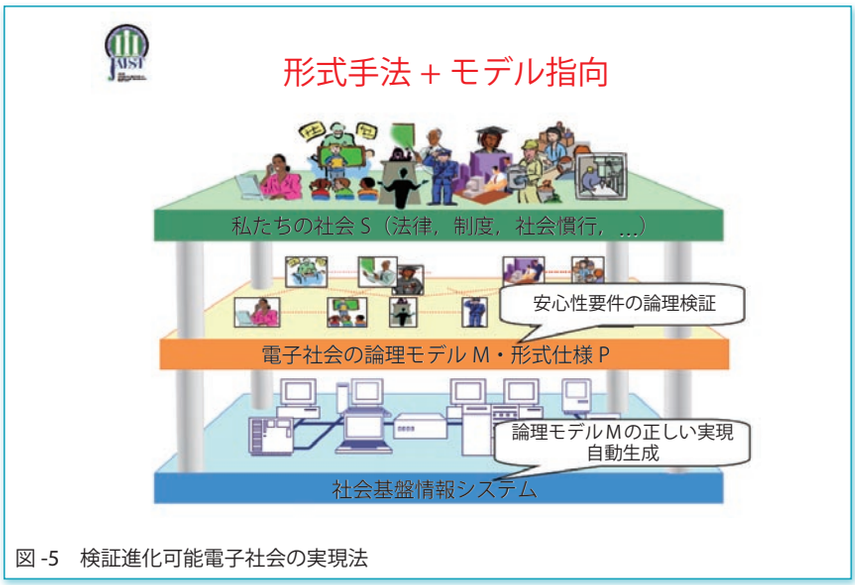


図 -5　検証進化可能電子社会の実現法

## 検証進化可能電子社会と形式手法

電子社会の進展に伴い，電子社会として実現される可能性のある社会・行政システムやビジネスシステムの客観性の高い記述を体系的に開発し，その記述を通してシステムの健全性や安全性の解析や検証をする技術がますます重要になりつつある．このように，形式手法は問題領域（ドメイン），要求，設計などを記述し解析する汎用技術としての重要性を増しつつある．以下では形式手法に基づき検証進化可能電子社会の実現を目指す取り組みを紹介する．

### ●検証進化可能電子社会プロジェクト

北陸先端科学技術大学院大学（JAIST）では，文部科学省の 21 世紀 COE プログラムの中で，検証進化可能電子社会プロジェクトを展開している．電子社会の構造や機能が関連法令によって規定されることから，法令文の形式論理表現の獲得を出発点として，(1) 法令文書の形式論理への変換による電子社会形式仕様獲得と推論，(2) 形式論理体系における推論と数学的基礎，(3) 形式推論による電子社会モデルの形式検証法と検証メカニズム，(4) 電子社会モデル化技術とアカウンタビリティ・進化機構，(5) セキュリティ，耐故障などの安心基盤技術，などの研究開発を行っている．その成果の一部は「法令工学の提案」[2] として出版公表されている．

**検証進化可能電子社会の要件：**JAIST21 世紀 COE プロジェクトでは検証進化可能電子社会の要件を，以下の図 -4 のように整理し，それに基づきネットワーク・ハードウェアレベルから応用領域までの各層にわたって研究を展開している．

**検証進化可能電子社会の実現法：**JAIST 21 世紀 COE プロジェクトの特徴の 1 つは，それが形式手法を重要な中核技術の 1 つとして位置づけている点である．すなわち，以下の図 -5 のように，ネットワークやコンピュータからなる基盤情報システムと応用領域である社会・行政・ビジネスシステムの間に，論理モデル・形式仕様の層を設けることで，電子社会の健全性，安全性・安心性，進化可能性などを確保しようとしている．これは多くの研究開発要素を含む挑戦的なプロジェクトではあるが，形式手法の重要な新展開を指し示している．

### ●ドメインの形式仕様

社会・行政・ビジネスシステムの系統的な作成・解析・検証・保守のためには，そのシステムが運用され意味を持つ領域（ドメイン）の記述が重要である．ドメイン記述の重要性は，そのドメインである意図を持って定められた規則や方針（ポリシー）の整合性や正当性などの解析・検証においてより顕著となり，ドメインの記述は客観的で厳密な解析や推論が可能である形式仕様であることが望まれる．ドメインの形式仕様の開発技術は，検証進化可能電子社会の実現のための最重要の基盤技術の 1 つである．

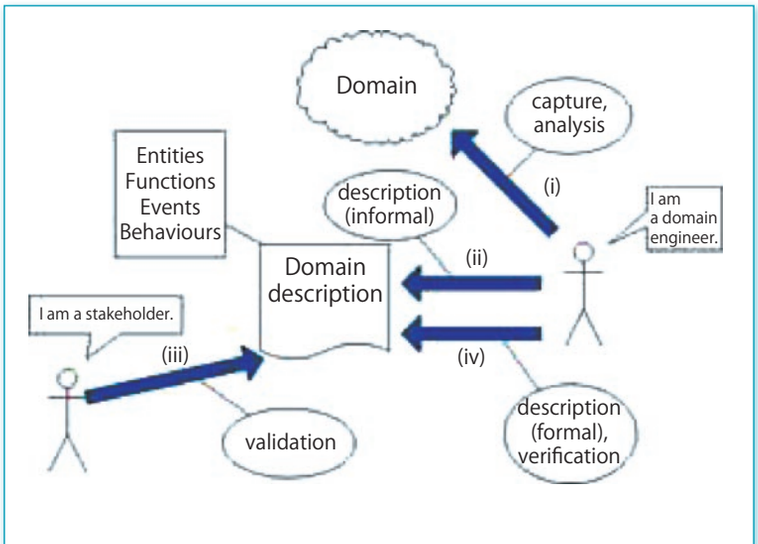


図 -6　ドメインの形式仕様の開発法

**ドメインの形式仕様の開発法**：ドメインの形式仕様の開発法としてはさまざまなものが考えられるが，ここでは Dines Bjorner のドメインの記述法[7] や CafeOBJ 言語による振舞仕様の形式記述法[6), 8), 9)] で採用されているものを紹介する．すなわちドメインの形式仕様は「ドメインでの観測可能な現象の記述」であり，「観測可能な現象」とは，実体 (entity)，関数 (function)，事象 (event)，そして振舞い (behavior) のことであるとする．ここで，実体，関数，事象，振舞いは以下のように定義される．

**実体**：実体とは対象とするドメインを構成する「もの」である．実体は型 (type または sort) により分類される．

**関数**：関数はドメインに存在する種々の機能 (function) を抽象化したものであり，実体を入力すると実体を出力する．関数を働かせて入力から出力を得ることを関数適用という．

**事象**：事象はドメインに変化を引き起こす（つまりある時点の状態を入力として次の状態を出力とする）関数適用である．

**振舞い**：振舞いは事象の系列である．

すなわち，問題としたいドメインを構成する実体とそれら実体に適用される関数を特定することで，そのドメインの基本構造をモデル化する．実体は種類により分類することで型付けされ，関数も実体の型を用いて型付けされる．こうして得られた実体と関数の（名前の）集合は，ドメインについて語るための言語となる．ドメインのさまざまな性質は，この言語を用いて記述・形式化される．ドメインに存在する実体が有する機能や，実体の間の関係などは，実体や関数を用いて直接的に記述・形式化されることが多い．しかし，ドメインが示す振舞いを記述・形式化するためには，ドメインの状態を入力として新たな状態を出力する (特殊な) 関数をいくつか特定し，ドメインにおける事象 (event) をこれらの関数の適用として定義し，それら事象の系列として振舞いを記述・定義する．

ドメインの形式仕様の開発はおおよそ次の手順に従って行われる．

(1) ドメイン情報の収集 (capture)，解析を行う (図 -6 の (i) の部分)．

(2) 自然言語，図などを使ってドメインの記述 (必ずしも形式仕様ではない) を開発する (図 -6 の (ii) の部分)．

(3) 利害関係者 (stakeholder) による確認 (validation) が行われる (図 -6 の (iii) の部分)．

(4) 検証 (verification) を行いつつドメインの形式仕様を開発する (図 -6 の (iv) の部分)．

(5) (i)，(ii)，(iii)，(iv) を繰り返しつつ了解性の高い記述と，検証可能な形式仕様を完成する．

確認とは，記述や形式仕様が利害関係者の意図と一致しているか確かめる作業のことであり，検証とは，形式仕様が記述者の意図するものと一致していることを証明することである．確認により「正しいドメイン記述」(the right domain description) を得ることができ，検証により「正しくドメイン記述を行う」(describe the domain right) ことができる．確認は図や自然言語などで記述されたドメインの記述を利害関係者に提示することによって行われ，検証は形式仕様の表しているモデルがそのドメインが持つべき性質を満たしていることを論理的に証明することによって行われる．確認と検証は補完的なものであり，確認により形式仕様が改良され検証が促進され，検証により記述が改良され確認が促進される．

ドメインの形式仕様の重要性は認識されつつあるが，その事例は少ない．以下では，筆者の研究グループの研究事例を紹介する．

**安全プロトコルの形式仕様と検証**[6),10)]：通信プロトコルや安全プロトコル (secure protocol) のモデル化と解析・検証の研究は，情報通信システムや電子商取引システムの発展に伴い進展してきた．各種プロトコルは情報ネットワークの基盤をなす規約であり，人間社会における法令に対応する．

安全プロトコルのモデル化と解析・検証においては，「安全性」をどのように定義するか，すなわちドメインをどのようにモデル化するか，がモデル化の最大のポイントになる．最近では，確率などを導入した精密なモデル化も試みられているが，解析・検証が複雑になる傾向があり，解析・検証がより容易な古典的なモデル化で十分であるとする立場もある．現状は「どのモデル化が妥当であるかは状況に応じて判断する」である，といえる．

我々は，Dolev-Yao の古典的な安全性のモデル化に基

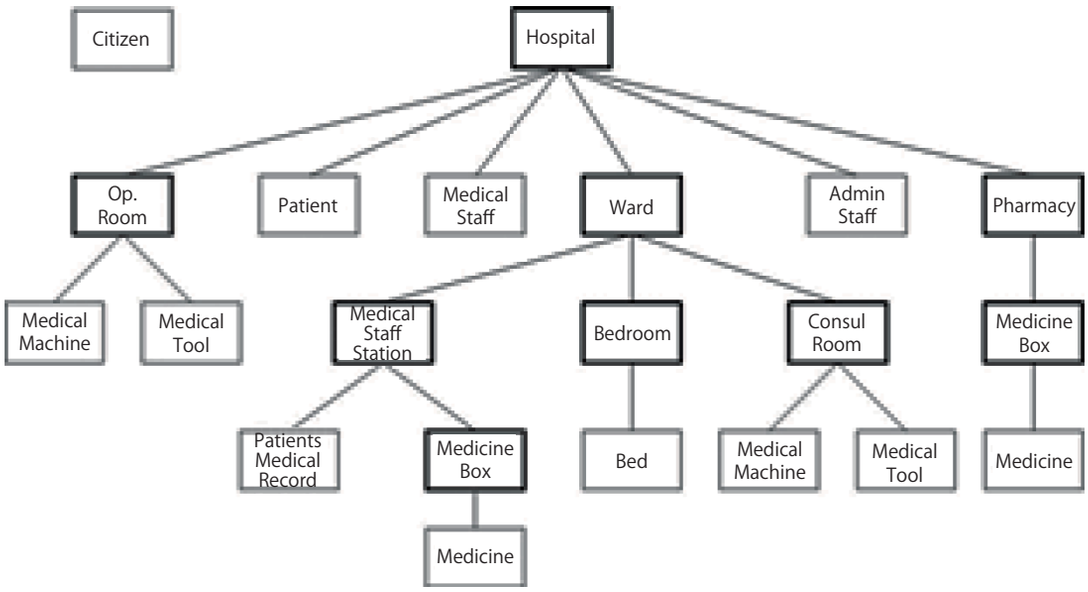


図 -7　病院ドメインにおける実体

づき，上述したドメインの形式仕様を採用し，安全プロトコルにおいて「交信のセッションを特定するノンスがもれることはなく，なりすましは行えない」といった安全性の検証の事例をいくつか開発してきた．これらの事例を開発した時点ではドメインの形式仕様という意識は強くはなかったが，結果的に興味深いドメインの形式仕様と検証の事例となっている．

**病院ドメインの形式仕様と検証**[1)]：ドメインの形式仕様と検証を具体的に研究することを目的として，病院ドメインを取り上げ，形式仕様と検証の事例研究を行っている．医療事故は社会的に大きな問題であり，その客観的な解析・検証には病院ドメインの体系的な形式仕様が有効であると期待される．また，病院ドメインは比較的多くの人が共通の理解を持っている対象であり，形式仕様と検証の方法論の研究開発の対象として適当であると考えられる．

病院ドメインにおける実体は，図 -7 に示されるようになる．このようにして定められた実体の上に，患者の入退院や治療行為を構成する関数を定義することで病院ドメインの形式仕様を作成する．この形式仕様に基づき，たとえば，患者の入退院や治療行為により，患者数が病院の能力を上回らないかどうかといった性質が解析可能となる．以上で紹介した病院ドメインの形式仕様の作成・解析技術を発展させ，内部統制（internal control）の形式仕様の作成と解析を行っている．

**公式文書ドメインの形式仕様と検証**[11)]：ドメインの形式仕様が，行政システムや会社システムなどの社会システムの解析・検証の新しい方法論となり得ることを目指して，公式文書ドメインの形式仕様と検証の事例研究を行っている．

公式文書ドメインとは，法治国家における法令や省令などの制定，改訂，運用を，公式文書に対する生成（制定），改訂，閲覧，複写，破棄などの操作と，それらの操作を誰が行い得るかの権利関係としてモデル化したドメインである．この公式文書ドメインは，電子政府の機能をネットワーク上に流通する電子文書に施される操作とそれらの権利関係の視点から捉えたものでもある．また行政組織や会社組織における統治や管理の機能を，それらの組織内を流通する文書に施される操作とそれらの権利関係の視点から捉えたものでもある．

我々は法治社会における公式文書ドメインを図 -8 のようにモデル化し，そのモデルにおいて「制定された法令や公開された公式文書が十分に利用者に解放されている」という意味の透明度（transparency）の概念を定義し，その解析を行っている．

## おわりに

形式手法の現場への適用は，以下のような 2 つの領域に二極化していくことが予想される．

(1) プログラムコードを直接の対象として，その正しさを解析・検証する分野．
(2) プログラムコードを作成する以前に，またはプログラムコードを作成することを前提とせず，問題，要求，設計の仕様を記述しその正しさを解析・検証する分野．

**プログラムコードの検証**：プログラムコードの正しさの検証は計算機の誕生以来の研究テーマである．すでに述

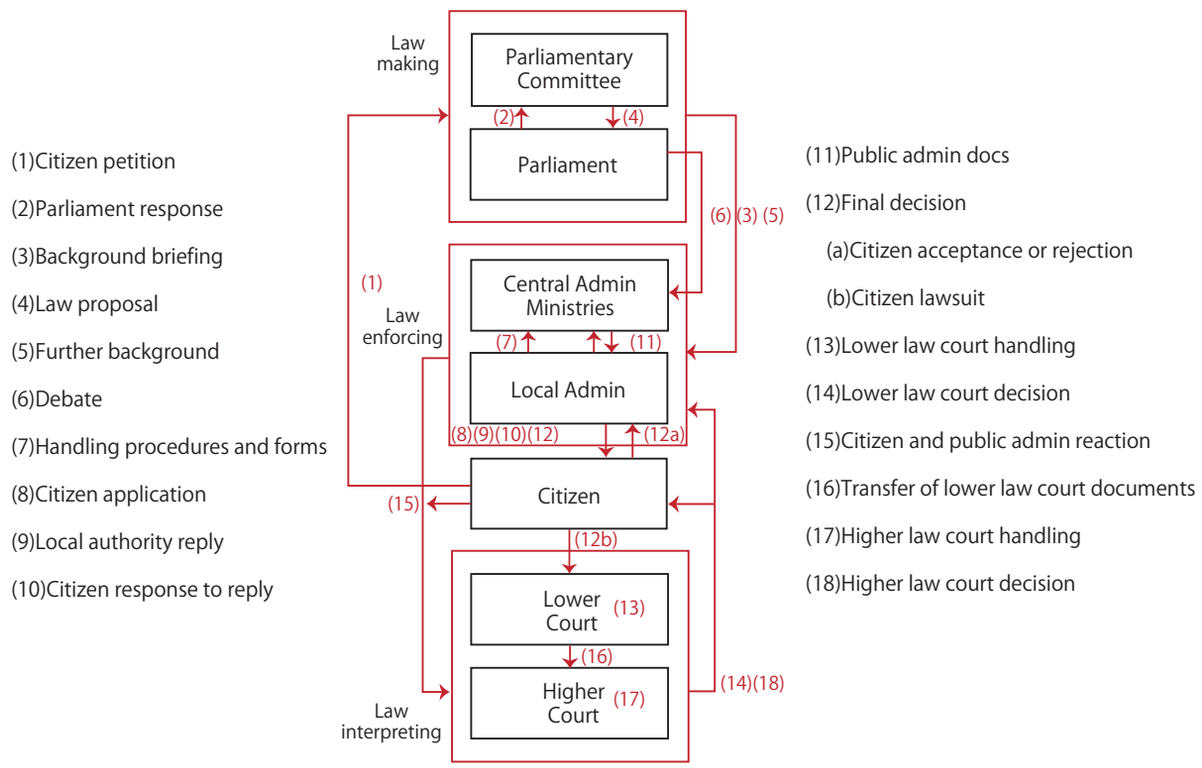


図 -8　公式文書ドメインのモデル

べたとおり，あらゆる種類のプログラムの正しさを統一的に検証することが不可能なことは，理論的にも技術的にも証明されている．しかし，ある特徴を共有するプログラムのクラスを対象とする検証システムや，人間と検証システムが対話しながらプログラムを検証する対話型検証システムは着実に進歩してきた．特に，プログラムを有限状態機械としてモデル化し到達可能な状態をすべて検索することで，プログラムが望みの性質を持つことを検証するモデル検査技術は，プログラムの検証技術に大きな進歩をもたらし，その適用領域を拡大させている．以下にプログラムコードの検証技術の今後を予測する上でのいくつかのポイントを掲げる．

- マイクロソフトがプログラムコードの自動検証技術に大きな研究投資をしていることからも推測できるとおり，この技術は適切に適用することで，多くのソフトウェア開発現場において大きな効果を発揮することが期待される．
- プログラムコードの自動検証の中核技術は，モデル検査と静的解析（static analysis）技術である．モデル検査システム（たとえば SPIN など）の使い方を習得するだけではなく，こうした技術の背景にある理論も含め，それらに対する深い理解を持った技術者を養うことが大切である．
- PVS，HOL，Coq といった定理証明システムを使った対話型のプログラムコード検証技術を通常のソフトウェアの開発現場で使うのは現実的でない．しかし，こうした技術を使って初めて検証できるプログラムコードも多くある，ということは認識すべきである．
- プログラムの全自動検証は魅力的であるが，完全な全自動検証は原理的に不可能であり，何らかの条件をつけて限定的にしか実現できないと再認識しておくことが重要である．
- 領域を特定したプログラム生成による検証（verification by generation）の手法は，適切に適用することで効果を発揮する可能性が高く，適用可能性を常に検討すべきである．

**問題・要求・設計の記述と検証**：システム開発の上流工程の問題仕様，要求仕様，設計仕様に誤りがあると，それがコード開発，テスト，運用といった下流工程において甚大な被害を及ぼす．システム開発においては，なるべく上流で仕様を体系的かつ客観的に記述し，それを解析・検証する技術が重要である．以下に要求仕様や設計仕様の記述と検証に関連した分野での形式手法の展開を予測する上でのポイントを掲げる．

- UML などのオブジェクトモデリングを支援する言語やシステムは，システムの要求や設計の記述を支援する技術として発展し，Modeling Language がシステム

開発において重要であるという認識を高めたという功績は大きい．しかし，UML で記述されたシステムの性質の解析や検証には，形式手法を導入することが不可欠である．

- 要求や設計の検証技術が実用化されれば，システム開発におけるその効果はコードレベルの検証よりもむしろ大きい．しかし，この分野でモデル検査技術により期待できる自動化は限定的であり，対話型の検証技術が必要であろう．特に，モデル化→記述→検証→モデル化→記述→検証→…というサイクルの中で，現実の問題の明確化・定式化を進めるためには，対話型のモデル化・検証技術が不可欠であろう．
- 要求や設計の検証は，本質的に困難な問題を多く含むので，この分野における形式手法の現場への適用は，モデル化と記述から始まり，検証に関しては特に信頼性と安全性への要求が高い分野に限定的となる可能性が高い．
- 問題・要求の分析と記述は問題領域そのものの分析と記述であり，必然的に多くの分野にまたがる活動であるので，経済学，社会学などの関連する分野との連携が重要となる．

**技術者教育の重要性**：形式手法はコストの高い技術であるといわれる．これは，この技術を使うために，

(1) データ構造とアルゴリズムの理論，
(2) 述語論理・等式論理・時制論理などの数理論理学の素養，
(3) 関数プログラミングなどの現代的なプログラミング技術，
(4) コンパイラ技術などのプログラム言語の基本技術，
(5) オブジェクトモデリングなどのモデル化技術や形式仕様言語を含む仕様開発の基本技術，
(6) 対話型検証システムやモデル検査システムの基礎理論やそうしたシステムを使う能力，

といった基礎基盤的な知識や能力を持った人材を必要とするからである．すべてのソフトウェア技術者にこうした能力を要求するのは現実的ではない．しかし，各々の問題領域ごとに形式手法の適用可能性とその限界を見極めるためには，こうした能力を有する中核的な人材を育てることが大切である．

**参考文献**


1) 有本泰仁，二木厚吉：ドメインの形式記述と検証，電子情報通信学会研究報告 SS2007-15, pp.35-40 (2007).
2) 片山卓也編：法令工学の提案，COE Research Monograph Series, Vol.2, JAIST Press (2007).
3) 中島　震：ソフトウェア工学の道具としての形式手法，ソフトウェアエンジニアリング最前線 2007，pp.27-48 (Aug. 2007)（修正改訂版が以下からダウンロード可能：http://research.nii.ac.jp/TechReports/07-007J-j.html）.
4) 二木厚吉：10 年後の形式手法，ZIPC WATCHERS, Vol.10, pp.4-9 (2006).
5) 二木厚吉，緒方和博，有本泰仁：法令対象ドメインの形式記述と検証，片山卓也編「法令工学の提案」2) 第 4 章，pp.71-93 (2007).
6) 二木厚吉，緒方和博，中村正樹：CafeOBJ 入門 (1)-(6)，コンピュータソフトウェア (2008 年 4 月から掲載予定).
7) Bjorner, D. : Software Engineering, Vol.3 : Domains, Requirements and Software Design, Texts in Theoretical Computer Science, the EATCS Series, Springer (2006).
8) CafeOBJ Home Page : http://www.ldl.jaist.ac.jp/cafeobj/
9) Futatsugi, K. : Verifying Specifications with Proof Scores in CafeOBJ, Proc. of 21st IEEE International Conference on Automated Software Engineering (ASE'06), pp.3-10 (2006).（an invited keynote talk）
10) Ogata, K. and Futatsugi, K. : Rewriting-Based Verification of Authentication Protocols, Proc. of 4th International Workshop on Rewriting Logic and its Applications (WRLA 2002), ENTCS 71, Elsevier (2002).
11) Chen, X., Kong, W. and Futatsugi, K. : Formal Support for e-Government System Design with Transparency Consideration, Proceedings of the 1st International Conference on Theory and Practice of Electronic Governance (ICEGOV 2007), ACM, pp.20-29 (2007).


（平成 20 年 3 月 21 日受付）


二木厚吉（正会員）
futatsugi@jaist.ac.jp
北陸先端科学技術大学院大学情報科学研究科教授．フォーマルメソッドの研究を進め，HISP，OBJ，CafeOBJ といった先駆的な形式仕様言語を設計開発してきた．フォーマルメソッドを核とした先端的なシステム開発法の産業界への浸透と，汎用科学技術としてのフォーマルメソッドの確立を目指している．