2. フォーマルメソッドの概要と特徴

本章では、フォーマルメソッドの概要と特徴についてまとめる。本章の概要は以下の通りである。

対象読者	(1) 発注者(CIO, CTO 等) (2) ベンダー上級管理者 (3) 開発技術者等
目的	フォーマルメソッドを導入する組織の管理者やフォーマルメソッドを初めて学ぶ人に、フォーマルメソッドの概要や特徴について分かりやすく説明する。
想定知識	ソフトウェア開発の概要
得られる知見等	● フォーマルメソッドの概要 ● フォーマルメソッドで何が得られるか ● フォーマルメソッドの特徴

フォーマルメソッド(=形式手法)は、数学的に厳密に意味付けられた言語(「形式仕様記述言語」と呼ぶ)を用いて情報システム(ソフトウェア、ハードウェア等)の要求[11]や設計[12]等の仕様を記述し、情報システムがユーザの要求等を満たしているかなど論理的に推論するための仕組みを提供する手法である。

フォーマルメソッドは、基礎とする数学理論などによって異なる多数の手法の総称であり、100 以上の異なる手法[13]が提案されており、それぞれ記述する対象範囲や検証目的等の適性に違いがある。比較的、実践でよく利用されるフォーマルメソッドには、Bメソッド、Event-B、VDM++、SPIN、NuSMV、Zなどの例がある。また、ProVerifのように、プロトコル検証など特定の目的に対応した専門家向けのフォーマルメソッドなどにも有用なものがある。フォーマルメソッドは、鉄道分野、航空分野、金融・セキュリティ分野など、セーフティクリティカル・システムやミッションクリティカル・システムなどにおいて、欧米を中心に実システムに対する適用事例が増えている[14]。

フォーマルメソッドと対比される従来のソフトウェア・テストは、ソフトウェアの品質[15](安全性、信頼性、セキュリティ等)を高めるため、不具合(バグ)を発見する目的で利用されるが、不具合が存在しないことを示すことができないという問題がある。特に、並列動作するソフトウェアや組込みシステム等において、並列動作のタイミングに関わる処理は再現性が無いため、いくらテスト件数を増

[11] ソフトウェアに求められる機能などを明確に規定したもの。
[12] ソフトウェアに求められる機能をどのように実現するかその方法を規定するもの。
[13] フォーマルメソッドの個々の手法を包括的に挙げた WEB サイトとして、Jonathan Bowen のサイト http://formalmethods.wikia.com/wiki/Formal_Methods_Wiki が代表的である。
[14] IPA, 「形式手法適用調査」, 2010
[15] 第 6.1 章に国際標準に基づくソフトウェアの品質についてまとめている。

やしても、すべての可能性について保証することができない。また、従来のテストでは、実行されなければいけない処理をテスト・ケースとして与えて動作確認をすることには向くが、テストケースだけを用いて、安全性やセキュリティなどのような望ましくない事が起きないことを示すことは困難である。

フォーマルメソッドでは、ソフトウェアがある性質を満たしていることを論理的に検証するため、一定の不具合が存在しないことを保証できる。そのような点で、完全な網羅性を持たない従来のテストの問題点を解決することができる。ただし、フォーマルメソッドを大規模なシステムに適用する場合などにおいて、コストが増大することもあるため、現実には、ソフトウェア・テストと部分的に使い分けるなど補完的な利用が想定される。

下図は、従来の開発プロセスにおける成果物とフォーマルメソッドの関係を示したものである。

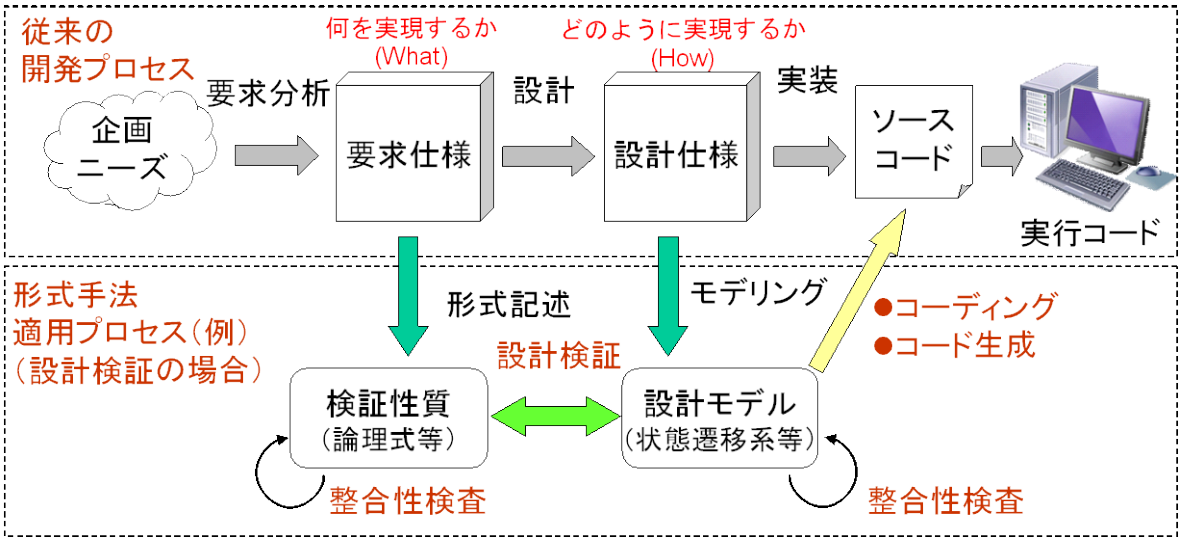

図 2-1:フォーマルメソッドの位置付け(設計検証の場合)

形式検証は、検証の基準(検証性質)と検証の対象(設計仕様)の両方について形式仕様言語を用いて記述し、それらをツールに入力することにより検証対象が、検証基準を満たしていることを検証する。

より具体的には、下図のとおり、日本語やダイアグラム等で書かれた要求仕様と設計仕様から、それぞれ形式仕様言語を用いて、検証性質と設計モデルをテキストベースで記述する。その両方をモデル検査ツールの入力とし、ツールの自動検査機能を用いて、検証性質が満たされるか、反例が存在するか確認することができる。

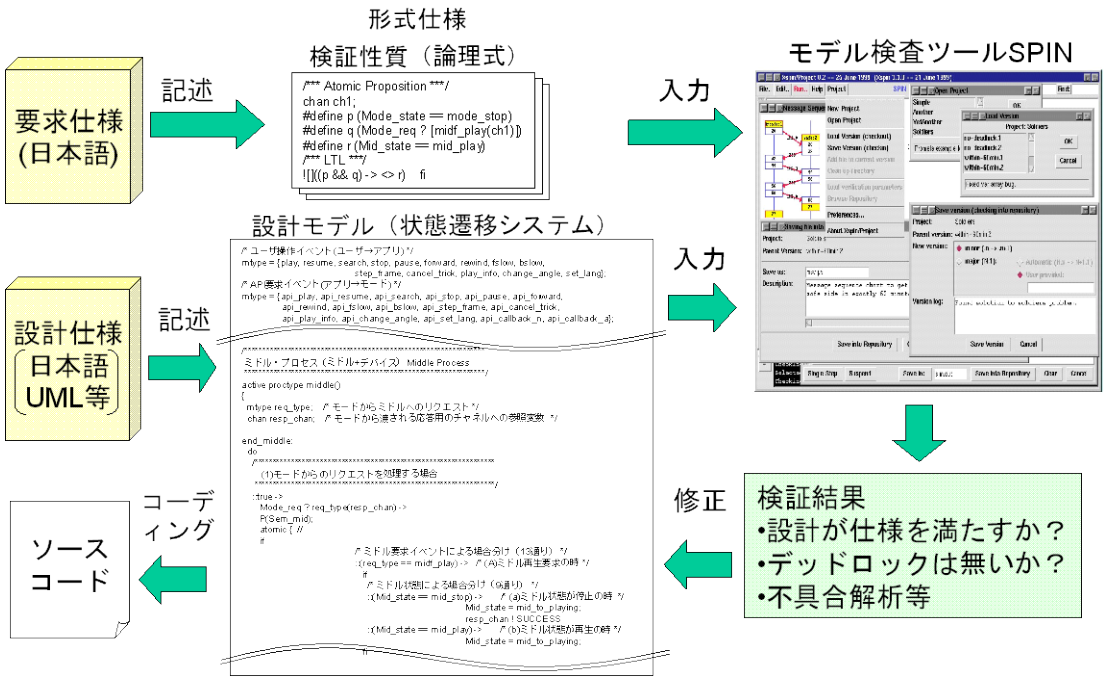

図 2-2:フォーマルメソッドの具体的イメージ(モデル検査 SPIN の場合)

開発プロセスにおいて、形式仕様記述をどの範囲、どの程度(詳細度)まで行い、形式検証をどの程度(検証項目のうち検証する比率)まで実施するか検討すること(「フォーマルメソッドの適用レベル」と呼ぶ)は、非常に重要である。それは、適用レベルによって、開発コストや得られる効果は大きく異なるためである。それは適用対象に応じて判断することが重要である。適用レベルは、大まかに以下のように分けられる。

表 2-1:フォーマルメソッドの適用レベル

適用レベル	説明
レベル0	形式仕様記述 数学的な記述が可能である形式仕様記述言語を用いて仕様を記述する。形式仕様記述言語には VDM++や Z 記法などがよく用いられる。記述だけでも、要求仕様の厳密な定義ができ、曖昧な定義による誤解の発生の防止などバグの低減に有効な場合がある。
レベル 1	形式的開発および検証 形式仕様を詳細化することでプログラムを開発、またはプログラムの性質を証明する。B メソッドなどが用いられる。
レベル 2	自動検証 プログラムの性質を自動的に検証する。

一方、フォーマルメソッド・ツールが提供する形式検証機能は、大まかに以下の2つのアプローチに分けられる。

表 2-2:形式検証のアプローチ

検証法	説明
モデル検査	対象システムを状態遷移系によりモデル化し、検証したい性質を、時相論理式と呼ばれる論理的な式で記述し、状態遷移系の状態を網羅的に探索し、時相論理式が満たされるか検査する。検証は自動で実行されるため一般向けに導入しやすい面はあるが、状態爆発と呼ばれる問題により検査処理が終了しない場合があるなど、技術的な制約がある。性質が満たされないことが判明した場合、反例としてエラー状態に至る実行系列を表示させることができ、これにより不具合の原因特定に役立てることができる。代表的なツールに、SPIN, NuSMV などがある。
定理証明	システムを論理式の集合などにより記述し、公理と推論規則に基づく証明を行う。証明においては人の支援が必要な場合が多く、証明をおこなうために専門知識を必要とする場合がある。モデル検査では困難な数値計算などを扱うことにも適している。代表的なツールとして、B メソッド、PVS などがある。

フォーマルメソッドは、文書(自然言語)で書かれた要求や設計の記述と比較して、厳密に記述するため記述コスト自体は大きくなるケースが多いが、ソフトウェア開発プロセスの上流に適用することにより、設計や要求の不具合を早期に発見し、テスト工程からの手戻りコストを大幅に抑える効果が期待できる。また、厳密な記述言語を用いてモデリングすることにより、ソフトウェアの安全性や信頼性などの品質が向上するという効果が期待できる。

適用対象に応じて、適用するフォーマルメソッドを適切に選択し、ソフトウェア全体のうち特定の部分に適用することで高い効果が得られる場合が多くみられるため、そのようなノウハウを身に着けることが重要である。本ガイダンスでは、そのような点についても手引きを示すことを目的としている。