14. ソフトウェア再利用とフォーマルメソッド

ここではソフトウェア再利用における検証一般の問題について説明した後、そこでフォーマルメソッドがどのように利用できるか、いくつかの利用局面について紹介する。それを踏まえ、ソフトウェア再利用においてフォーマルメソッドを活用する際に必要となると考えられるいくつかの留意点について触れる。

14.1. ソフトウェア再利用

他のソフトウェアの構成要素や構造を、ソフトウェアの開発に利用する技術のことをソフトウェア再利用という。ソースコードやバイナリレベルでの再利用から、設計やアーキテクチャの再利用、要求仕様の再利用など、およそソフトウェアの様々な成果物が再利用の対象となる。

図 14-1 は、再利用の基本的な構造を模式的に示したものである。ここで再利用資産は再利用されるソースコードや設計などを示し、それが複数のプロダクトの開発に使われることを示している。単一の再利用資産を利用するということもあるが、例えばクラスライブラリ、フレームワーク、あるいはプロダクトライン開発におけるコア資産のように、複数の再利用資産を体系的に管理し、各プロダクトの開発において必要なものを取捨選択して再利用することも多く行われている。こうした際には再利用資産の組み合わせ方などに条件や制約が発生することがあるので、再利用資産の妥当な構成方法について考慮しながら再利用することになる。

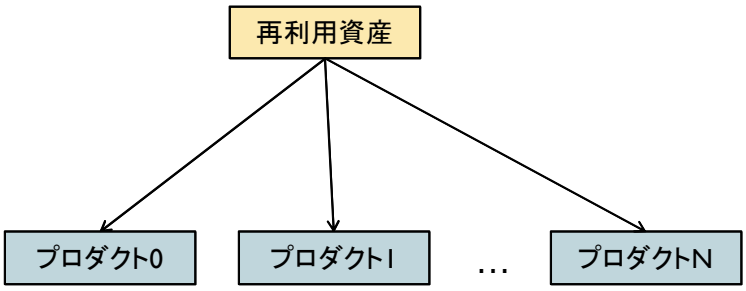

図 14-1:再利用の基本構造

再利用が行われる理由は、それによって様々な利点が期待されるからである。表 14-1 は、ソフトウェア再利用に対する様々な期待を示したものである。ここに示されるように、新規に作るためには人的なリソースがかかり、また開発上のリスクも大きく、さらに利用実績のないものは信頼性の確保が大変であり、開発期間がかかるため、再利用によってより効率的かつ低リスクで信頼性の高いソフトウェアを作るということが期待感としてあることが分かる。

表 14-1:再利用に対する期待[169]

信頼性の向上	実績があり安定したソフトウェアを使うことで、新たに作るよりも信頼性が高まる
開発上のリスクの軽減	これから新規に開発するよりも、検証済みの既存のものを使うほうがリスクが小さい
開発リソースの有効利用	専門家の知識や労力の結果である既存資産を活用 (毎回新規開発すれば専門家を毎回投入する必要がある)
標準化への適合	標準に適合化のために標準的な部品を再利用する (標準的な GUI 部品を使うなど)

[169] Pressmann, Roger S.: Software Engineering, McGraw-Hill, 邦訳:西他訳、実践ソフトウェアエンジニアリング - ソフトウェアプロフェッショナルのための基本知識、日科技連出版社。

開発期間の短縮	すべてを作らずに再利用することで開発期間を短縮し、製品化までの時間(time to market)を短縮する

一方、再利用には様々な難しさもある。表 14-2 は再利用を行う際の困難を示したものである。ここに示されるように、再利用資産を使った開発上の問題だけでなく、保守を含めた運用全般の問題が困難の原因となっていることが分かる。

表 14-2:再利用を行う際の困難

保守コストの増加	再利用部分のソースコードなどが利用できない場合には、保守のコストが増加する
ツール支援の欠如	多くの開発ツールは新規開発を支援するが、再利用の支援が不十分である
NIH シンドローム	ソフトウェア技術者は自分で新たに開発することを好む (保守がやりやすい、新たに作る方がチャレンジング) NIH: Not Invented Here
再利用資産の維持	部品を広め、使わせ、維持することは困難
検索、理解、適用	利用できる部品を探し、その部品を理解し、自分のソフトウェア開発に適用することはいずれも困難な作業

このように、ソフトウェア再利用に対する期待感はあるものの、実際に行うためには様々な困難さがあることも事実である。しかしながらソフトウェアは大規模化、複雑化する一方、開発期間は短くなっており、困難さにも関わらず再利用をしない開発はほとんど考えられない状況になっている。また上述したように、フレームワークやプロダクトライン開発のコア資産のような大規模で体系だった再利用資産の活用も進められており、ソフトウェア再利用の効果的な活用はソフトウェア開発における重要なテーマのひとつとなっている。

本章では、こうしたソフトウェア再利用において、フォーマルメソッドがどう使えるのか、ということについていくつかの考え方や使い方について紹介する。ここではフォーマルメソッドのうち、モデル検査技術に限定する。なおソフトウェア再利用に対するフォーマルメソッドの適用についてはいろいろな検討がなされているが、特定の利用方法が広く認められているわけではなく、また技術的な課題も多いのが現状である。ここで紹介するものは、そうした適用の考え方の一部であることを注意しておく。

14.2. 再利用における検証の課題

ソフトウェア再利用における検証上の課題について考えてみる。ソフトウェア再利用は、再利用されるソフトウェア(再利用資産)を作るという活動と、その再利用資産を利用してソフトウェア(プロダクト)を作るという活動によって構成される。ここで再利用資産は一般に複数回利用されることが通常である。このとき、再利用における検証の課題は、模式的に
図 14-2 のように捉えることができる。

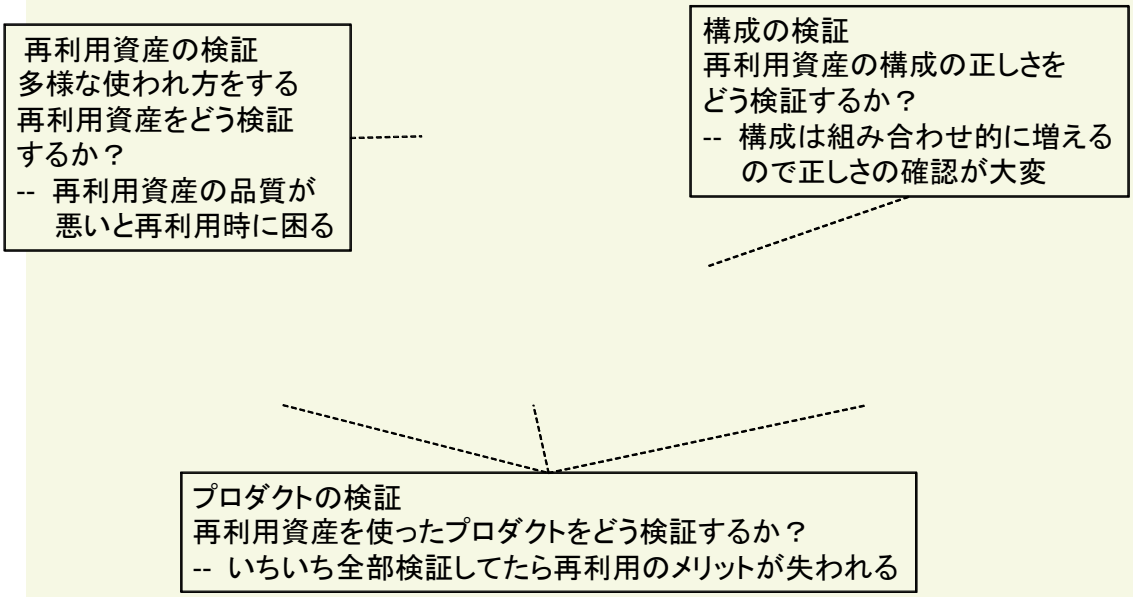

図 14-2:再利用における検証一般の課題

14.2.1. 再利用資産の検証

一般にソフトウェアを検証する際には、そのソフトウェアがどのように使われるかということに照らして検証する。しかしながら再利用資産は様々なソフトウェアの開発に使われるため、その使われ方も多様である。したがって再利用資産に対しては、そうした多様な使われ方に照らして検証を行うことが求められる。もしも再利用資産に対して十分な検証がなされておらずその品質が不十分だと、仮に再利用資産を利用して少ないコストでプロダクトを構築することができたとしても、そのプロダクトの検証においては利用した再利用資産部分を含めて検証を行う必要がでてくる。一般に開発全体に占める検証のコストは非常に大きいため、これでは十分な開発の効率化が期待できない。

しかしながら、再利用資産を開発する時点では、それが将来どのようなプロダクトによってどのような使われ方をするかを決定することができないことが多く、多様な使い方に即した検証をあらかじめ行うことは困難である。仮にある程度の使い方が想定されたとしても、それらすべてをあらかじめ検証することは検証量からできないことも多く、再利用資産開発時の検証を困難にしている。このように再利用資産をどこまでどのように検証するかは重要な課題である。

14.2.2. プロダクトの検証

再利用資産を利用してプロダクトを開発した場合は、利用せずに開発した場合より、再利用資産に関わる部分の検証が楽になることが期待される。上述したように検証のコストが削減できないと再利用によるコスト削減効果は小さくなるからである。

しかしながら現実には再利用資産を開発した時点で十分な検証を行うことは難しいため、そのプロダクトの使い方に即した検証はなされていない事もあり得る。既に実績のある再利用資産であっても、使い方が異なれば不具合が出てくることは十分にあり得る。また再利用の困難さに示されているように、再利用資産はそのソースコードが入手できない場合などもあり、利用者が十分な検証を行ったり、それに基づいて修正を行ったりすることが難しい場合も多々ある。

現実には、プロダクトの開発において過去に実績のない方法で再利用資産を使う場合には、プロダクトの検証時に再利用資産に関わる検証を行わざるを得ないため、そうした検証をどう効果的・効率的に行うかは重要な課題である。

14.2.3. 構成の検証

体系だった再利用を行うためには、必要と考えられる再利用資産を複数整備し、それらを体系的

に管理、活用することが求められる。例えばアプリケーションフレームワークは、特定業務やドメインのプロダクトの開発に必要となるクラス定義などをあらかじめ用意することで、個々のプロダクトを効率的に作ることを狙ったものである。またプロダクトライン開発においては、想定するプロダクト群の共通性と可変性を分析し、そうしたプロダクト群の開発に必要となる再利用資産をコア資産として整備する。

こうした再利用開発においては、複数の再利用資産を活用して行われるが、そうした再利用資産は、例えばある再利用資産を使う際には別の再利用資産も利用することを想定しているとか、ある再利用資産と別の再利用資産は組み合せて使うことは意図していないとかいった様々な構成上の条件や制約が課せられることが一般的である。したがって、複数の再利用資産を用いてプロダクトを開発する人は、自分の利用している再利用資産群の構成が正しいものであるかどうかを確認することが必要となる。

一般に再利用資産の構成方法は組合せ的に考えられるため、組み合せる再利用資産の規模が多くなればなるほど、その正しさの確認は難しく煩雑になるため、そうした構成の正しさをどう検証するかも課題となる。

14.3. 再利用における検証手法の例

こうした再利用における検証の課題に対してどのような対応方法が考えられるのか、テストなどの一般的な検証における例を簡単に紹介する。

14.3.1. 再利用資産の検証

再利用資産の検証に対しては、考えられる利用方法をできるだけ網羅することが求められる。しかしながらあらゆる利用方法を網羅することは困難である。例えばひとつの関数がパラメータを複数持つとき、それらのパラメータの値や組み合わせをすべて網羅した検証を行うことは現実には不可能なことが多い。そのため、一般的なテストにおいては、パラメータの値を同値分割や境界値分割していくつかの代表値で検証したり、直交表などを用いてパラメータの対の組合せをできるだけ網羅するように検証したりすることで、より効果的な検証を行うことなどが行われる。

もちろんこうした手法は一定の有効性を示すが、再利用資産の組み合わせに関わる検証においては、その組合せを網羅することはさらに困難となる。ひとつの素朴な考え方としては、プロダクト毎に使い方が異なる部分に関しては、再利用資産開発時には検証を行わず、プロダクトによらず同じような使われ方をする部分に注力して検証を行うというアプローチもあり得る。しかしながらプロダクトを開発する立場にとれば、プロダクト毎に利用方法が違う部分に対しても、一定の検証がなされていることが期待される。

プロダクトライン開発などにおいてはコア資産が膨大となるため、例えばサンプルアプリケーション戦略(SAS: Sample Application Strategy)などが提案されている。SASは、コア資産検証の段階で、いくつかのサンプルアプリケーションを想定し、それらの使われ方の中でコア資産を検証する方法である。この方法は、実際のアプリケーションの利用方法の中で再利用資産を検証することができるため、現実的な検証が期待される。実際にアプリケーションを作って検証するためコストがかかり、また取り上げたサンプルアプリケーション以外のコンテキストでの検証はできないという問題はあるが、再利用資産開発の段階で早期の確認ができること、再利用資産開発時に想定されているプロダクトをサンプルアプリケーションとして選定するなどすることで、現実的な戦略として活用できる利点がある。

14.3.2. プロダクトの検証

再利用資産を使ったプロダクトの検証においては、例えばそのプロダクトの使い方を踏まえ、十分な網羅性はなくても一通りの検証を行うことで再利用資産の品質を査定し、その程度や状況に応じて、再利用資産に関してどの程度の検証が必要か判断して検証を行うなど、再利用資産に関わる検証のコストを減少させるように運用することが多いと考えられる。しかしながらプロダクトに求められる品質の程度や、再利用資産の状況によっては、再利用資産に関わる検証も新規部分と同

程度に行うこともあり得る。

そうしたプロダクトの検証を効果的に行うために、例えば共通性と再利用戦略CRS (Commonality and Reuse Strategy)などが提唱されている[170]。これは再利用資産開発時にはプロダクトによらず同じように使われる部分の検証に注力し、プロダクト毎に利用方法が変わりうる部分については、その部分を検証するための検証資産を作成する方法である。ここで検証資産とは、例えばテストであればテストデータ、スタブ、ドライバ、期待値など、テストを行う際に必要となる各種の資産を指す。個々のプロダクトの検証においては、共通部分の検証についてはコア資産の検証時に使われた検証資産を利用し、プロダクト固有の使われ方をする部分の検証については、用意された検証資産をそのアプリケーションの使い方に適合させてからテストを行う。この戦略では、検証そのものを減らすわけではないが、検証資産の再利用によって、プロダクトの検証のコストを削減することが期待される。

14.3.3. 構成の検証

再利用資産の構成については、再利用資産設計時に再利用資産間にどのような依存関係や排他関係があるかを定義しておき、再利用資産の特定の組み合わせがその関係を満たしているか、あるいはその関係を満たす再利用資産の組合せとしてどのようなものがあるかを求めることなどがなされる。

こうした構成の検証は、構成管理の分野で一定の技術が培われている。またプロダクトライン開発においては、フィーチャモデルを構成の検証に応用することも検討されている。フィーチャモデルは、本来は要求項目としてのフィーチャの構造を記述するためのモデリング手法だが、それを応用することで再利用資産の構造を導出したり表現したりする提案が多くなされている。図 14-3 はフィーチャモデルの記述例である。ここでは、プロダクトに対するフィーチャが必須、選択(プロダクトによって使われたり使われなかったりする)、代替(プロダクトによって選択肢中のいずれかが使われる)が階層的に示されており、あわせてフィーチャ間の依存関係も示されている。フィーチャモデルはこうした条件や制約を満たしたフィーチャの構成をコンパクトに示していると考えられ、その構成を満たす具体的なフィーチャの組合せを導出する手法の検討などがなされている。

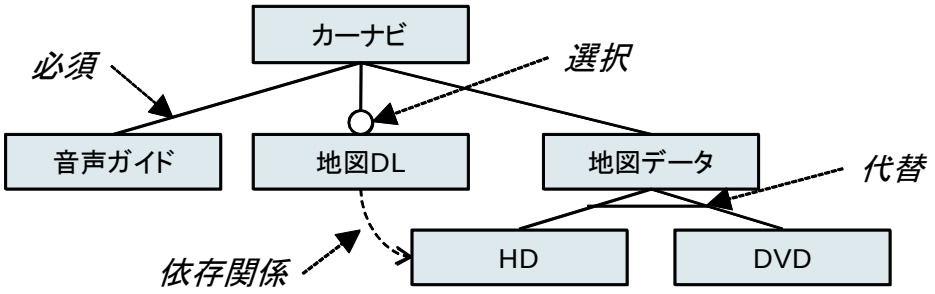

図 14-3:フィーチャモデルの例

14.4. フォーマルメソッドの適用

以上説明してきたように、ソフトウェア再利用はソフトウェア開発にとって重要で不可欠な技術であるが、その活用や運用には様々な困難さがあり、検証についてもいろいろと課題がある。こうした再利用における検証に、フォーマルメソッドを活用するという取り組みはいろいろとある。なおここでフォーマルメソッドとしてはモデル検査技術に限定する。

ソフトウェア再利用における検証にフォーマルメソッドを適用することの素朴な期待感としては、以下のようなものが挙げられる。

[170] Pohl, K. 他: Software Product Line Engineering, Foundation, Principles, and Techniques, Springer, 2005.

- 再利用資産は複数のプロダクトから繰り返し使われるものであるから、品質が高いことが求められる。したがって、フォーマルメソッドのように従来の検証手法より厳密で信頼の高い検証手法の活用が期待される。
- 再利用資産は様々なプロダクトから様々な使われ方をするために、その検証が困難である。モデル検査技術のような手法を用いることで、こうした使われ方に照らしてより網羅的な検証が可能になることが期待される。
- 再利用資産は繰り返し使われるため、単独のプロダクトに比較してより検証コストが高くついたとしても全体としてはメリットが得られることがあり得る。フォーマルメソッドは一般に高価な検証手法と考えられるが、再利用資産の検証という局面で、相対的に有利なコストで活用されることが期待される。

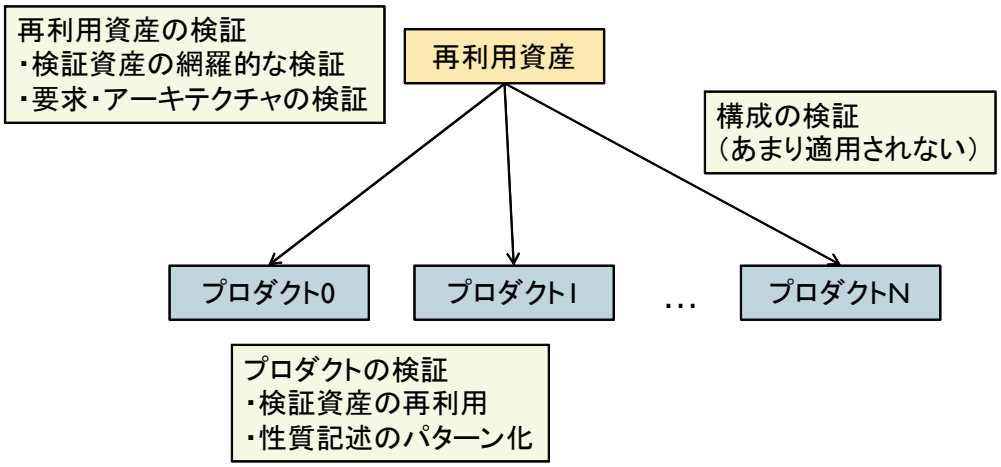

図 14-4：再利用におけるモデル検査技術の適用の概観

図 14-4 は、再利用におけるモデル検査技術の適用について概観したものである。

再利用資産の検証においては、検証資産を様々な使われ方に照らして網羅的に検証するためにモデル検査を利用することが考えられる。この段階では詳細なプロダクトの機能などに関わる部分を検証するのではなく、要求やアーキテクチャに関わる重要なポイントについて検証することが多い。

一方、プロダクトの検証においては、モデル検査によるプロダクトの検証を効率的に行うために、検証資産を再利用することが考えられる。また検証資産の一部である性質記述については、様々なパターン化の試みなどがなされている。

構成の検証に対するフォーマルメソッドの適用例もいろいろとあるが、これらはモデル検査技術以外の技術を利用するものがほとんどであり、ここでは触れない。

以下、再利用資産の検証およびプロダクトの検証に対するフォーマルメソッドの適用例を紹介すると同時に、そうした適用を行うとしたときに、考慮しておくとよいと考えられる事項について簡単に説明する。

14.5. 再利用資産の検証とフォーマルメソッド

14.5.1. 検証方法の例

再利用資産の検証では、様々なプロダクトの使い方に応じた検証が必要であることを指摘した。モデル検査を利用することによって、そうした多様性を考慮した検証を効果的に行う提案がある。例えば、再利用資産の設定によってその振る舞いを変えることができ、プロダクトごとに設定が変わりうる場合、設定によって成り立つ性質が変わってしまう。そのような場合、どのような設定があり、その設定に応じてどのような性質を持つということを定義し、その性質を検証する必要がある。プロダクトライン開発などでは、こうした利用するプロダクトごとの違いを可変性と呼ぶが、再利用資産の検

証において、この可変性を利用することで、プロダクトに応じた多様な性質を一度にモデル検査することが可能となる。

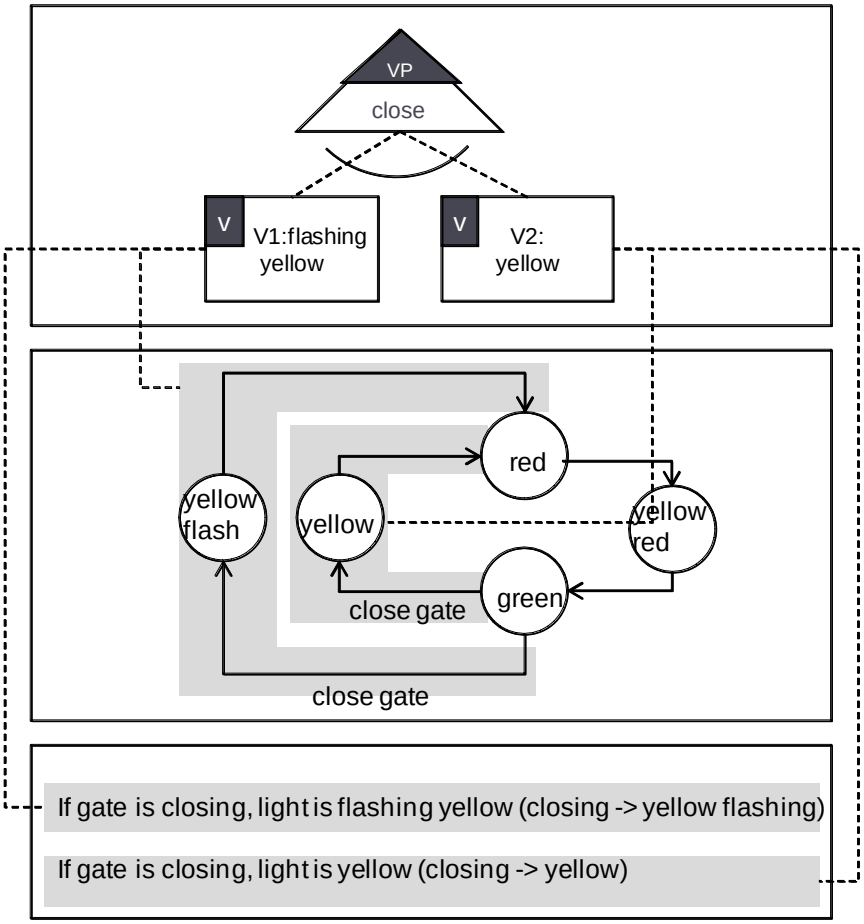

図 14-5:再利用資産のモデル検査の例[171]

図 14-5 は、Lauenroth らによる再利用資産のモデル検査の概略を示すものである。ここでは信号システムの例が示されている。図の上部は、OVM(Orthogonal Variability Model)と呼ばれる可変性のモデルである。ここでは遮断されたときに黄色が点滅するプロダクトと、黄色が点灯するプロダクトがあることが示されている。図の中部は、対象とするシステムのふるまいを示すステートマシン図の一部である。ここで黄色が点滅(yellow flash)する状態と遷移と、点灯する(yellow)する状態と遷移が含まれている。ここで可変性の V1 が選ばれたときは点滅の状態や遷移が、V2 が選ばれた時は点灯の状態や遷移が選ばれる。また図の下部は性質記述で、V1,V2 に応じて異なった性質記述がある。モデル検査を行う際には、どの可変性が選ばれたならどういう性質が成り立つ、というように可変性の選択状態を含めた性質記述を用いてモデル検査を行う。一般に可変性は複数あるので、可変性の選択状態はベクトルで表すことができ、意味のあるベクトル(可変性の組み合わせ)を用いて性質記述をすることになる。なおこの研究では、検証対象は要求記述となっている。

14.5.2. 考慮点

再利用資産の検証を行う際に、プロダクトごとの使い方の多様性を一度に検証しようとすると、上

[171] Lauenroth, Kim, 他: Model Checking of Domain Artifacts in Product Line Engineering, Proc. of Automated Software Engineering, 2009.

述の例のように利用のされ方の多様性を整理する必要がある。プロダクトライン開発においては、こうしたプロダクトごとの違いを可変性として捉え、上記の例のような OVM や、前述したフィーチャモデルなどを使ってモデル化することができる。ただし、多様な性質を検証することで検証が複雑化し、単一のプロダクトの利用方法のみに関わる性質であれば検証が可能であったものが、検証できなくなるということもあり得る。上記の例で検証対象が要求記述となっているのは、設計記述になるとより対象が複雑化するためであるとも考えられる。

しかし仮に多様な性質を一度に検証することをしなくても、可変性モデルなどを使って多様性を捉えることは有効である。可変性モデルを使うことで、多様性の広がりを認識することができ、その一部の状況に応じた性質しか検証していないとしても、それが本来の多様性のうちのどれだけの部分であるかを理解することができるからである。可変性モデルに照らして、どういう状況に関しては検証がなされているかを明確にすれば、プロダクトの開発を行う人は、自分の使い方が検証済の使い方なのかどうかを理解することができ、再利用資産に関わる検証をどれだけ行うかの判断に利用することもできる。

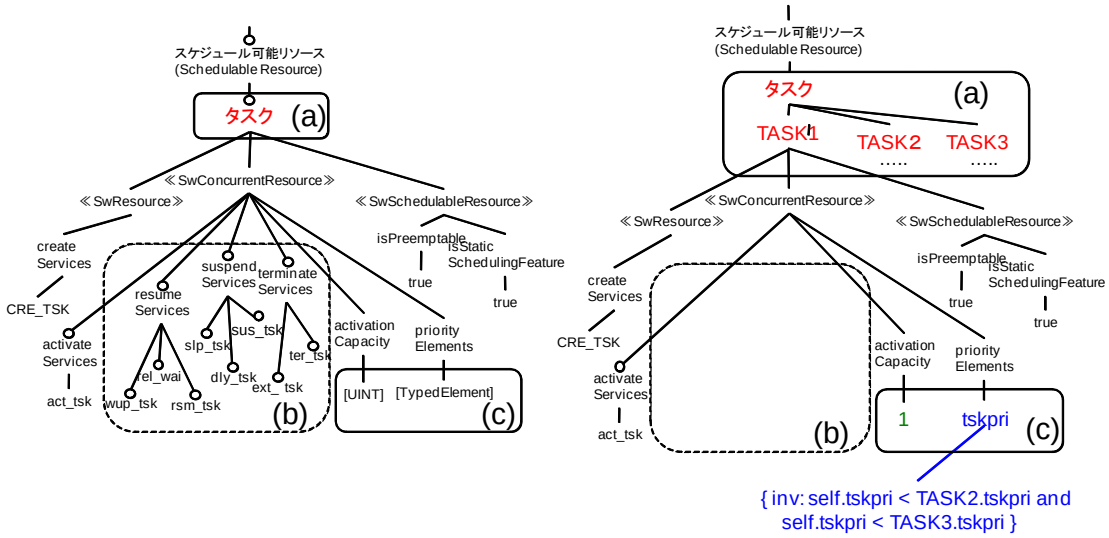

図 14-6:フィーチャモデルを用いた検証状況の明確化例[172]

図 14-6 は、再利用資産を利用する際のリアルタイム OS の設定等の多様性をフィーチャモデルによって表現した例である。詳細は省くが、図の左は考えられる多様性をあらかじめテンプレートとして定義したもの、右はそのテンプレートを踏まえ具体的なプロダクトにおける設定等を明確にしたものである。こうした記述を利用することで多様な設定方法の中で、どういう設定について検証をしたのかが明確になる。

14.6. プロダクトの検証とフォーマルメソッド

14.6.1. 検証方法の例

再利用資産が完璧に検証しきれない以上、どれだけの検証コストを費やすかという程度の議論はあっても、プロダクトの検証において再利用資産の検証をゼロにすることはできない。少なくとも再利用資産の使い方が正しいかどうかはプロダクト側の問題であるから、再利用資産開発時にそれを検証することは不可能である。したがってテストなどの検証で提案されている CRS のような戦略

[172] 朝倉功太, 他: 想定モデリングに基づくソフトウェアプロダクトラインのコア資産検証手法, 情報処理学会 組込みシステムシンポジウム 2009 (ESS2009).

が、モデル検査においても有効になる。すなわち、モデル検査に必要な検証資産を再利用可能な形で用意し、プロダクトの検証時にはその検証資産を再利用することで、検証を効果的、効率的に行おうというものである。

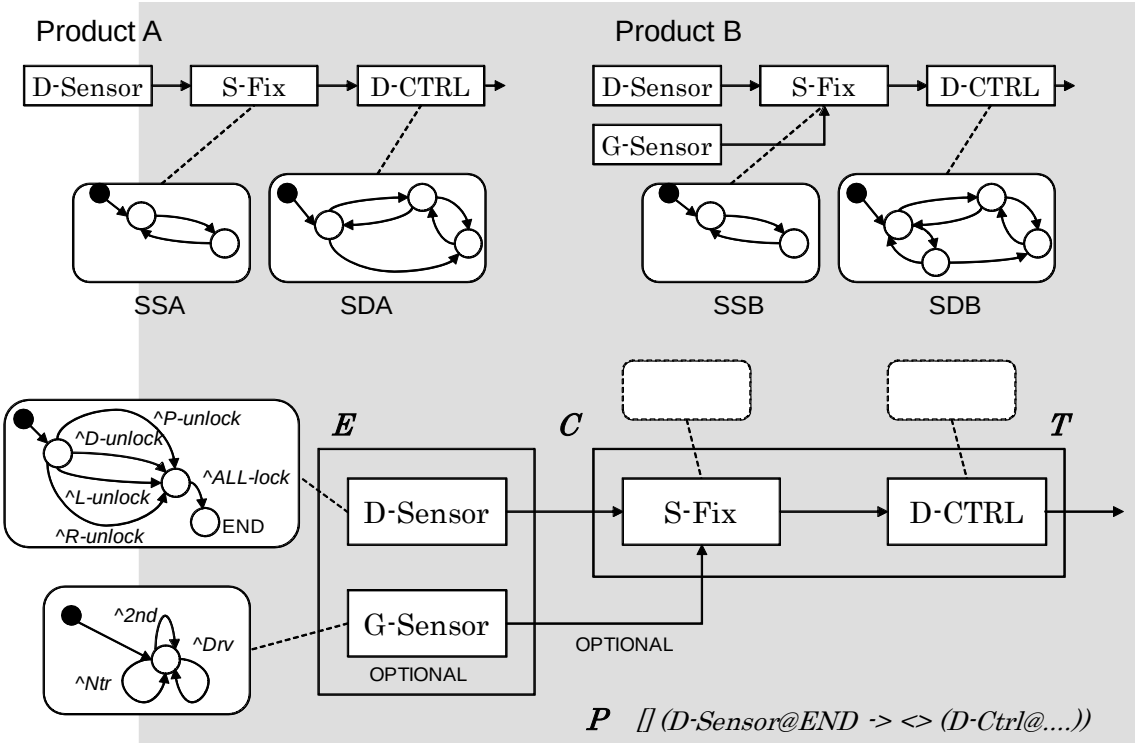

図 14-7:再利用可能な検証資産の例[173]

図 14-7 は Kishi らの提案する再利用可能な検証資産の概略を示すものである。上部には二つの異なったプロダクトの構造が示されている。これらは同じ再利用資産群を利用しているが、使い方や構成が異なる。こうしたプロダクトの多様性を踏まえ、それらの再利用資産の使い方や構成を包括的に含んだ検証モデルをあらかじめ用意しておく。また性質記述もプロダクトごとに変更される部分は変更可能な記述として用意しておく。図の下部がそれらに対応する。プロダクトの検証においては、こうした再利用可能な検証モデルや性質記述から、そのプロダクトに適した検証モデルや性質記述を導出し、モデル検査を行う。

また性質記述に関してはよく利用する性質記述をパターン化するという提案もなされている。例えばDwyerらは、応答(response)や不在(absence)などといったカテゴライズを行い、例えば*S*が*P*に応答するということが常に成り立つなら、CTLでは"*AG(P→AF(S))*"、LTLでは"□(*P*→◇*S*)"と記述するといったように、よく使われる時相論理式のパターンを整理している[174]。こうした提案では、応答性など汎用性の高い性質のパターンを集めているが、特定のプロジェクトにおいてよく使われる性質記述を再利用できるようにパターン化しておくと有用と考えられる。

14.6.2. 考慮点

検証資産を再利用することは、プロダクトの検証において有用性があると考えられる。モデル検査は検証モデルや性質の定義に特有のスキルが必要であるし、注意深く定義しないとささいな間違

[173] Kishi, T. 他: Formal Verification and Software Product Lines, Communications of the ACM, Vol. 49, No12, 2006.

[174] Dwyer, M.B. 他: Patterns in Property Specifications for Fineite-Stte Verification, Proc. of International Conference on Software Engineering, 1999.

いによって、本来の検証目的が達成できない危険性もある。したがって、実績のある検証モデルや性質を再利用することは検証の信頼性を上げる上でも効果があると期待される。また、テストやレビューは、そのプロジェクトで作るべき成果物そのものが対象となるが、モデル検査委技術は多くの場合、そうした成果物とは別に、対象を抽象化したモデルを構築するといったコストが無視できない。したがって一旦構築した検証モデルや性質を繰り返し利用できるとすれば、複数のプロダクトに利用できるという意味で単位あたりのコストを下げることが期待される。

もちろん、検証資産の再利用はソフトウェアの再利用と同様の困難さが伴う。そもそもプロダクトの開発ごとにモデル検査技術によって繰り返し検証すべき重要な性質とは何かという点を明確にすることが本質である。既存の研究では、本質的な要求モデルや、アーキテクチャ設計上の性質に焦点をあてた研究が多い。また、検証モデルは検証したい性質に強く依存しているので、検証性質が異なれば検証モデルそのものを大きく変更しなければならない事もある。したがってソフトウェアそのもの以上に、どの部分をどのように検証するかということを明確にすることが重要となる。特にモデル検査技術の場合、内部状態に応じた性質記述をするので、性質記述に使われる状態が、その検証モデルにおいて把握しやすいようにモデルが作られていることが重要である。こうした点は、検証資産を再利用するという局面に限ったことではないが、再利用においては一層注意を払うべき点と考えられる。

なお検証性質については、いろいろな図式の方法が提案されている。パターン化された性質をこうした図式表現で表すことがそのプロジェクトメンバにとって理解性を向上すると判断される場合は、そうした表現の活用も有効と考えられる。

14.7. まとめ

以上、ソフトウェア再利用とフォーマルメソッドの関わりについて、モデル検査技術による再利用資産の検証、プロダクトの検証という二つの局面から、いくつかの提案を紹介した。冒頭で述べたように、再利用におけるフォーマルメソッドの活用について、広く認められている提案があるというわけではない。ここでは既存の提案の中からいくつか参考になると判断されるものを紹介した。

なおモデル検査技術による検証においては、状態爆発の問題を上手に回避する必要がある。上述したように再利用資産の検証で様々な利用方法に照らして性質を検証したり、プロダクトの検証においてモデルにプロダクト毎の切り替えを含むような仕掛けをいれたりすると、単一プロダクトの検証と比較して状態数が増える可能性が大きくなる。モデルを「上手に」作ることが本質ではあるが、それは一定のスキルが必要であり、工学的にはいくつかの割り切りが必要となると考えられる。

表 14-3：検証量を減らすための工学的な手法例[175]

設計の記述レベル （アーキテクチャへの注目）	設計の抽象度が上がると表現される情報量が減少するために検証量が減る。但しモデルの解像度は下がるので、検証性質は粗くなる。
特性の機能に限定 （範囲の限定）	検証対象を限定化することで検証量が減る。但しその限定した範囲がどういうもので、その範囲を検証するということがどういう意味を持つのかについて十分な検討が必要。
基本的な処理に注目 （パターン化）	パターン化して複数の処理を代表する処理に抽象化することで検証量が減る。但し、パターンで成り立つ性質が、そのパターンに帰属する具体的な処理で成り立つかどうかということに関しては、十分な検討が必要。
シーケンス長を制限 （検証の深さ）	探索の深さを限定することで検証の量が下がる。但し、より長いシーケンスにおける性質は分からないので、検証の質は下が

[175] 岸知二，他：組込みソフトウェア設計検証へのモデル検査技術の適用と考察，SEC Journal 12 号，2007.

	る。

表 14-3 は、状態量を減らすために考えられるソフトウェア工学視点からの方策例である。特段目新しい手法でも、再利用に特化した手法でもないが、こうした工学的な観点からの状態量削減を行うことも有効であり状態数が多くなりがちな再利用に関わる検証を行う際には参考になると考えられる。