5. フォーマルメソッド導入のコストと効果の考え方

本章では、フォーマルメソッドの導入について検討する際のコストと効果に関する考え方と考慮すべき点について示す。概要は以下の通りである。

対象読者	(1)ベンダー上級管理者 (2)開発プロジェクト管理者 (3)発注者（CIO, CTO 等)
目的	フォーマルメソッドを組織として導入するか検討するための判断材料として、フォーマルメソッドの効果とコストの全体像を示し、それぞれの要素がどのぐらいの規模になるか実績や具体例を示す。これにより、事故リスク等の規模などまだ十分には認識されていないものについて、考慮すべき点を示す。
想定知識	ソフトウェア開発プロセスの概略。
得られる知見等	● フォーマルメソッド導入の際に検討すべきコストと効果の全体像 ● 損害リスクの規模感の認識とリスク評価の重要性 ● 組織として導入を検討する際の注意点 ● 実践的な応用事例に基づく現状の把握 ● 実践的な応用事例（付録）の利用法

5.1. コストと効果に関する検討手順の概要

組織へのフォーマルメソッドの導入の可否を判断する上で、コストと効果に関する検討のための参考手順の概要を以下に示す。

(1) 本ガイダンスに示すフォーマルメソッドの効果とコストに関する全体像(5.2 章)を把握し、それを参考に実際の対象事例・分野について主な要素を洗い出す。

(2) 効果とコストの主な要素について、ガイダンスに示す具体例や考え方(5.3 章、5.4 章)に基づき、対象事例・分野について規模を大まかに見積もる。

(3) 適用事例によっては、品質向上、コスト削減、リスク低減等の効果を相互に比較できるほどの定量化を行うことが難しい場合も多いため、適用事例の状況に応じて、優先すべき効果の要素を中心に考える。

(4) コスト（学習コスト、設計コスト、ツール導入コスト等）の考え方、事例を参考にして、対象事例のコストの規模を大まかに見積もり、効果の全体あるいは優先すべき特定の効果の規模と比較・検討することにより、導入を判断する。

フォーマルメソッド導入のコストと効果の要素には定量的に比較することは難しいものが多い。

しかし、主要な要素を見落とすなど偏った判断を避けるために、主な要素と全体像を把握して、各要素の大まかな規模を見積もることは非常に重要である。本章で示す内容は、コストと効果の考え方やそれらの規模に関する参考情報であり、費用対効果に関する厳密な定量評価手法を示すものではない。また、そのような定量化手法を示すことは、現実の場では困難な場合が多いと言える[24]。

5.2. フォーマルメソッドの効果とコストの全体像

フォーマルメソッドの効果には、従来のソフトウェア・テストでは発見が困難な不具合の検出やソフトウェアが要求を満たすことを論理的に保証することなどによりソフトウェア等の安全性・信頼性等の品質[25]の向上や、上流の設計工程で不具合を除去することによりテスト工程の工数を削減することなどが挙げられる。これらの効果は、適用対象、適用手法、適用度合いによって異なるため、それぞれの事例に応じて効果を評価することが必要である。品質の向上に関する効果は、不具合による情報システムの障害等が発生して始めて顕在化するものであり、最終的には、情報システムに関わる組織(ユーザ、ベンダー等)の事業や企業価値への影響として捉えられるものである。これらを適切に捉えるためにはリスク評価の考え方が必要になる。多くの企業等では、情報システムに要求される品質に対して十分な対策をとっていると考えているが、実際に発生している情報システムの不具合に起因する事故の件数やその被害の影響規模の大きさをみれば、対策が十分とは言えない[26],[27],[28]。対策が十分であるかどうかは、利用される情報システムの重要度に応じて異なる。PCなど個人用途や基幹系ではない事務向けなどの一般的なソフトウェアであれば、ソフトウェアの信頼性の高さよりも、機能の新規性や有用性を重視し、市場にいち早くリリースすることが市場競争力を決める決定的要因となっている[29],[30],[31],。一方で、重要インフラ、企業の基幹系、決済系、セキュリティなどのソフトウェアにおいては、その影響度は極めて大きいため、ソフトウェア品質に対する要求は高い。

フォーマルメソッドの効果の全体像に把握するためには、IT投資対効果の評価手法であるVMM(Value Measuring Methodology)[32],[33]などを参考にすることができる。VMMやプロジェクト

---

[24] 機能安全標準(IEC61508)では、障害を確率的ハードウェア障害(Random hardware failure)と系統的障害(Systematic failure)に分類し、前者についてはハードウェアの物理的劣化による障害発生の定量的取り扱いを行うが、後者については設計や実装の不具合など確定的にきまるものであり、その不具合の混入確率を定量的に評価することは困難であるため、開発の手順や手法などの定性的な取り扱いによる対策を定めている。

[25] ソフトウェアの品質や機能要求、非機能要求等の関係については、第 6.1 章を参照。

[26] 障害を発生させない、被害を拡大させないためのシステム対策ガイド、JUAS, IPA、2009

[27] 情報システムの信頼性向上ガイド, JUAS, 2010

[28] 高信頼化のための開発手法ガイドブックー予防と検証の事例を中心にー、IPA,2010

[29] Michael Cusumano, Critical Decisions in Software Development:Updating the State of the Practice, IEEE Software, 2009.

[30] Michael Cusumano, Software Development Worldwide - The State of the Practice, IEEE Software, 2003.

[31] Michael A. Cusuman, ソフトウェア企業の競争戦略, サイコムインターナショナル

[32] 米国連邦調達庁(GSA)、ハーバード大学、ブーズ・アレン・ハミルトンコンサルティングなどにより開発された IT 投資対効果の評価手法で、米国政府内のさまざまなプロジェクトに活用されている。

[33] CIO council, Best Practice Committee, Value Measuring Methodology, 2002.

管理等の手法では、通常、このような技術投資の効果は、(1)価値の向上、(2)コストの削減（効率化）、(3)リスクの低減の3つの要素に大きく分類される。フォーマルメソッドの場合について言えば、表 5-1 のように分類される。価値の向上は、フォーマルメソッドのそもそもの導入目的であるソフトウェアの品質向上であり、コストの削減は、生産性の向上による工数の削減、リスクの低減は、特定の工程における負荷の集中による開発要員の変動や、開発手戻り等による工数見積りの大きな変動等による納期リスクの低減である。

表 5-1:フォーマルメソッド導入の主な効果

分類	主な例	説明
(B1)品質向上	不具合の除去。 要求仕様の動作保証。 説明責任の向上。	バグに起因するシステム障害により、サービス停止、人命・経済損失等の被害をもたらす可能性を減らす効果がある。また、不具合が発見されることにより、企業に対する信用失墜の損害も大きい。
(B2)コスト削減	開発工数の削減	開発上流の要求・設計工程に重点を置くことで、コーディング、テストの工数を削減できる場合がある。また、開発の効率化により、現場の負担を軽減する。
(B3)リスク低減	工数、予算見積もり、品質のバラつきを抑える。	要求・設計工程の重点化により、手戻り等による工数の変動リスクや、品質のバラつきリスクを抑える。

品質向上の効果は、具体的には、不具合による製品の修理回収やシステム障害による損害の回避や、さらには、事故やシステムに不具合の顕在化により企業に対する信用失墜、企業価値の毀損の回避などが挙げられる。前者については、人命に関わるセーフティクリティカル・システムや企業等の基幹業務に関わるミッションクリティカル・システムのみならず、情報家電においてさえ、不具合が見つかった場合の回収修理コストなどで 100 億円程度の大規模な損害が発生している例もある[34]。後者の規模については、一般に、把握している人は少ないが、欧米を中心とする情報セキュリティ経済学の分野では企業の市場価値に基づく評価手法が示されており、その影響はきわめて大きいことが示されている[35,36]。あるベンダーが開発したシステムに不具合が見つかり企業の信用が失墜すれば、その製品の売上げのみならず、同社のビジネス全体に大きな影響を与える場合がある。フォーマルメソッドは、ソフトウェアの新しい機能を生み出すものではなく、機能などの品質を高めることが目的の一つであるため、具体的な効果は障害の発生とうい不確定な事象

[34] http://www.sony.co.jp/SonyInfo/News/Press_Archive/200107/01-0706/
[35] Lawrence A. Gordon, Martin P. Loeb, Managing Cybersecurity Resources A Cost-Benefit Analysis, McGraw-Hill,2006.
[36] Masaki Ishiguro, Hideyuki Tanaka, Kanta Matsuura, The Effect of Information Security Incidents on Corporate Values in the Japanese Stock Market, WESII 2006.

に依存して決まる[37]。

コスト削減の効果については、フォーマルメソッドの導入により要求分析や設計などの上流工程で、仕様の検証やシミュレーション実行などの解析が可能となり、開発工程の早期に不具合を取り除き、テスト工程の工数を削減するとともに、テスト工程からの手戻りによる工数増大を防止する効果が大きい[38]。また、従来の開発プロセスのように、テスト工程に負荷のピークが集中することで、開発要員の許容負荷のオーバーや手戻りコストの変動による納期遅延のリスク[39]を低減する効果が得られる。

一方、フォーマルメソッドの導入のための投資コストについては、主に以下のように分類される。

表 5-2:フォーマルメソッドの主な投資コスト

主な投資コスト	内容
(C1)フォーマルメソッド習得コスト	フォーマルメソッドごとに手法の基礎、ツールの使い方についての学習。技術指導・コンサルティング委託費用もある。
(C2)設計工数の増大	従来の設計にフォーマルメソッドを組み込む場合、設計工数が増加する場合がある。
(C3)ツール取得コスト	無償のツールは多いが、統合開発環境となっているツールで有償の製品がある。

フォーマルメソッドの習得コストは、利用する手法や対象システムにフォーマルメソッドをどの程度のレベルまで適用するかによって異なる。学習コストは、手法により異なるが、特定の手法に関する入門書、参考書の通読と例題等による実験等の時間から見積もることができる。また、設計工数の増大に関しては、従来の設計に比べて、フォーマルメソッドにより厳密で曖昧性のない仕様を作成し、設計工程で、仕様の矛盾や整合性等の検証を行う場合、従来の設計に比べて工数が増大する場合が多い。ツール取得コストに関しては、フォーマルメソッドの場合、無償のツールが中心であるが、統合的な開発環境として有償のものも存在する。これらの投資コストは、効果に挙げたリスク評価に比べ、見積もりは容易である。

フォーマルメソッドの効果は、開発現場の視点から工数や生産性向上などの短期的な観点のみならず、事業者や利用者などの視点からシステム障害による損害などビジネスリスクも考慮した広い視点から考えることが重要である。

以上のような効果とコストの要素の中には、定量化することが難しいものもある。しかし、要素ご

[37] 「(B1)品質」は、開発プロジェクトの成果物に関する価値であり、事故による損害リスクは、ソフトウェアの品質に依存するため(B1)に含める。「(B3)リスク」は、開発プロジェクトの失敗や遅延等などのプロジェクトのリスクを対象とする。

[38] Marko Auerswald, EASIS Guidelines for the Development of Dependable Integrated Safety Systems, 2006

[39] (B3)のリスクには、プロジェクトの遂行におけるメンバーのコミュニケーションギャップ、技術者の意欲の低下など様々な要因によるプロジェクトの失敗や納期遅延が考えられるが、開発事例や組織に依存するため、本ガイダンスでは詳細には触れない。

とに適度な度合いでその規模の目安を評価することはフォーマルメソッドの導入判断において不可欠である。マネジメント分野において、ピーター ドラッカーは「測定できないものは管理できない。」[40]と言い、また、ソフトウェア工学の分野においても、T. デマルコの有名な「計測できないものは制御できない」[41]と言っている。適度なレベルで評価をしなければ、フォーマルメソッドの費用対効果を適切に把握することはできない。

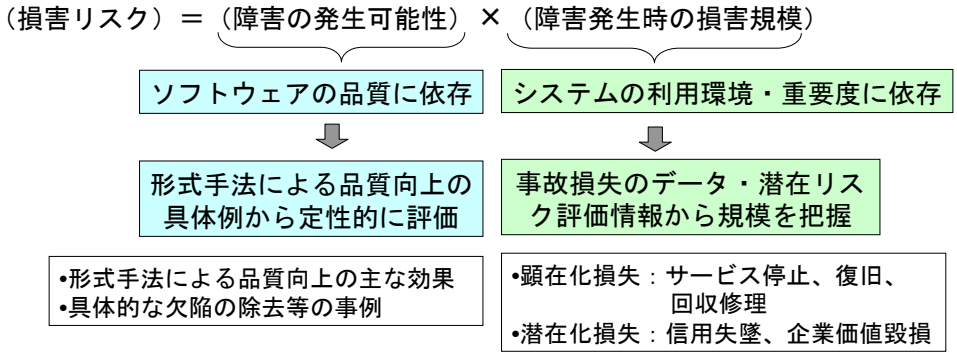
以上に挙げた効果とコストの主な要素(B1～B3, C1～C3)について、以下の章で具体的にどの程度の規模であるか、実績や具体例を示すことで、各組織における導入判断の参考情報を提供する。これらの全体像を把握することは、フォーマルメソッドの導入判断で重要である。

5.3. フォーマルメソッドの効果について

5.3.1. 品質向上

フォーマルメソッドの適用による効果のうち品質向上は、ソフトウェアの出荷・稼動後の不具合の顕在化やIT事故による損害の発生として始めて具現化する。情報セキュリティ経済学においては、損害リスクは、システム障害により発生する直接的な損失よりも、事故により企業の信用が失墜することで企業価値を毀損することによる潜在化損失の割合が大きいことが示されている[42]。

情報セキュリティや一般の事故等による損害リスクは、大まかな概念としては事故の発生可能性と事故が発生した場合の損害規模(影響の度合い)という2つの要素の積の関係として捉えられる[43],[44],[45]。ソフトウェアの場合においても、不具合に起因する事故(システム障害)の損害リスク(損害規模の期待値)についても概念的には以下のような関係で表わすことができる[46]。



[40] Drucker, P., The Practice of Management. New York, NY: Harper Business "you can't manage what you can't measure", 1993

[41] "Controlling Software Projects: Management, Measurement, and Estimates," Tom Demarco, "You can't control what you can't measure."

[42] Lawrence A. Gordon, Martin P. Loeb, Managing Cybersecurity Resources A Cost-Benefit Analysis, McGraw-Hill, 2006.

[43] 機能安全(IEC61508) part 5, リスク評価

[44] 畠中伸敏, 情報セキュリティのためのリスク分析・評価 第2版, 日科技連出版社

[45] ISO/IEC Guide51

[46] ソフトウェアには、ハードウェアと異なり、故障という概念がない。ソフトウェアの不具合は設計や実装段階で確定している障害(Systematic Failure)のみで、不具合が潜在する確率を評価することは困難だが、ここではソフトウェアの不具合に起因する情報システムの障害を対象と考える。

この関係の「(障害の発生可能性)」(以下、「発生可能性」と呼ぶ。)は、ソフトウェアの品質に依存する[47]。一方、「(障害発生時の損害規模)」(以下、「損害規模」と呼ぶ。)は、ソフトウェアの重要性や利用環境によって決まる。利用者が多ければ多いほど、また、システムの用途が重要であればあるほど、システムに障害が発生した場合の損害規模は大きくなる。前述のシステム事故による企業の信用失墜による企業価値の毀損もこれに含まれる。

表 5-3:ソフトウェアの不具合に起因する損害リスクの要素

要素	影響要因	フォーマルメソッドとの関係
システム障害の発生可能性 (発生可能性)	ソフトウェアの品質が高い程障害の発生頻度は下がる。	フォーマルメソッドの適用によりソフトウェアの品質を高めることで、発生頻度を下げる効果が期待できる。
障害が発生した場合の損害規模 (損害規模)	ソフトウェアの利用状況、規模、重要性によって決まる。	フォーマルメソッドは直接影響しないが、フォーマルメソッドの導入判断で考慮すべき重要な点。

フォーマルメソッドの適用によって品質が向上すれば、「発生可能性」を下げる効果が期待できるが、事故が起きた場合の「損害規模」に直接的には影響を与えない。このようにリスク評価では、障害の発生可能性と損害規模を区別して考えることが一般的である。ソフトウェアによる障害の発生可能性は、ハードウェアの故障確率のような数量的に扱うことは困難である[48]。そのため損害リスク全体を定量的に評価することは困難であるが、「損害規模」の目安を把握することは非常に重要である。障害の影響は最悪の規模を想定して、対策を検討することがリスク管理の基本的な考え方だからである。図 5-1 は、発生可能性と障害発生時の影響度とリスクの関係を示したものである。

[47] ソフトウェアによる事故の種類によって影響の度合いは異なるが、事故の種類ごとに見れば、大まかな概念として、障害の発生可能性と障害発生時の損害規模の積として捉えることができる。

[48] ソフトウェアによる障害の発生可能性を数量化することが困難な理由は以下のように考えることができる。障害の発生可能性は、利用環境における実行パスの実行頻度と実行パス上に不具合が潜在する可能性の重み付き総和として大まかに捉えられる。不具合の混入は、人のミス等によるもので、実行パスの実行頻度とは無関係であるため、それらの重み付き総和は、数量的な尺度とはならない。

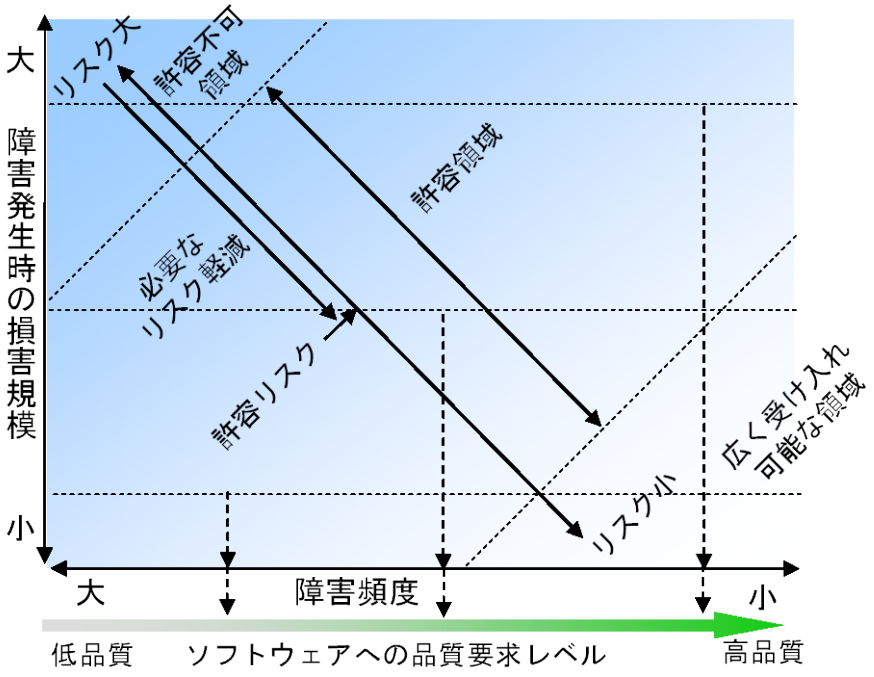


図 5-1:発生可能性と障害発生時の影響度とリスクの関係

図 5-1 において、縦軸の「損害規模」と横軸の「発生可能性」の積によってリスクがきまるため、グラフの対角線上の左上ほどリスクは高く、右下ほどリスクは低い。情報システムは、あらかじめ求められるニーズや用途などのシステムの重要性に応じて「損害規模」が決まるため、リスクを一定レベルに抑えるためには、品質を向上させることにより、発生可能性を減らすことで、リスクを一定レベルに抑える必要性がある。

フォーマルメソッドは、ソフトウェア全体のどの部分に適用し、どの程度検証するかにより、期待できるソフトウェアの品質向上やコストが大きく異なる。フォーマルメソッドによりソフトウェアの品質を高める場合、高い品質になるに従い、より多くのコストが発生する傾向にある。損害規模は、システムの用途や利用環境により、きわめて大きな幅を持つため、フォーマルメソッドの導入検討においては、損害規模を考慮することは不可欠である。ソフトウェアの品質を高めるためにいくらコストをかけても、不具合による障害の損害規模が小さければ、採算性がとれないという結果になる。一方、セーフティクリティカル・システムや重要インフラシステムなど影響度の大きなシステムには、それに見合った大きな投資コストをかけても、ソフトウェア品質の向上が求められる。ソフトウェアに求められる品質は、ソフトウェアの重要度(抱えるリスク)に応じて検討すべきであり、別々に議論することはできない。

一定の規模の複雑さをもつソフトウェアでは、どんなにクリティカルなものでも、完璧なものは期待できないことは世の中の事故事例が物語っている。完璧なソフトウェアを追求すれば、急激なコストの上昇を伴うことを覚悟しなければない。機能安全の考え方では、許容可能リスク、許容不可

リスクの境界を定め、許容範囲内にリスクを抑えるための対策レベルを検討する[49]。また、マーケットで受け入れられているソフトウェアの品質が必ずしも高いとは限らない。パソコン等一般のコンシューマーに利用されるソフトウェアでは、新しい機能をマーケットに早期に投入することが成功の要因となる場合が多い[50],[51]。現実的に考えれば、システムが利用される環境からその影響度を考慮し、コストとのバランスからフォーマルメソッドの導入を検討すべきである。日本学術会議の安全に関する緊急特別委員会報告「安全学の構築に向けて」[52]においても、「絶対安全」から「リスクを基準とする安全の評価」への意識の転換が必要であることが述べられていることも、同様の考え方である。

5.3.1.1. 障害発生時の損害規模について

「損害規模」に関しては、システム障害によるサービスの停止、復旧コスト、製品回収修理コストなどの顕在化損失と、企業の信用やブランド価値に与える影響からくる潜在化損失がある[53]。

（損害規模）＝（顕在化損失）＋（潜在化損失）

顕在化損失については、前述の通り情報家電の不具合修理のための回収コストで 100 億円を超えるものがある。セーフティクリティカル・システムの例を挙げると、顕在化損失だけで、衛星1機の打ち上げ失敗の損害は 200 億円、航空機では数百億円以上、原子炉やITSやスマート・グリッドの構築等大規模なインフラシステムにおける致命的損失となれば兆円レベルにも及ぶことが指摘されている[54]。顕在化損失の具体的な規模を示すものとして、経済産業省が実施する組込みソフトウェアに関する事故被害の額を示したものがある（図 5-2）。

[49] IEC61508 part 5
[50] Michael Cusmano, Critical Decisions in Software Development, 2009.
[51] Michael. Cusumano et al., “Software Development Worldwide:The State of the Practice,” IEEE Software, Dec. 2003, pp. 28–34.
[52] 日本学術会議
[53] 経済産業省 リスク定量化ワークショップ, 2007 年, “IT 事故と情報セキュリティ対策が企業価値に与える影響分析”, 石黒正揮（三菱総合研究所）, 松浦幹太（東京大学）, 田中秀幸（東京大学）
[54] 藤枝純教、経営者はアークテクチャと形式手法を忘れてはいけない、IPA SEC Journal

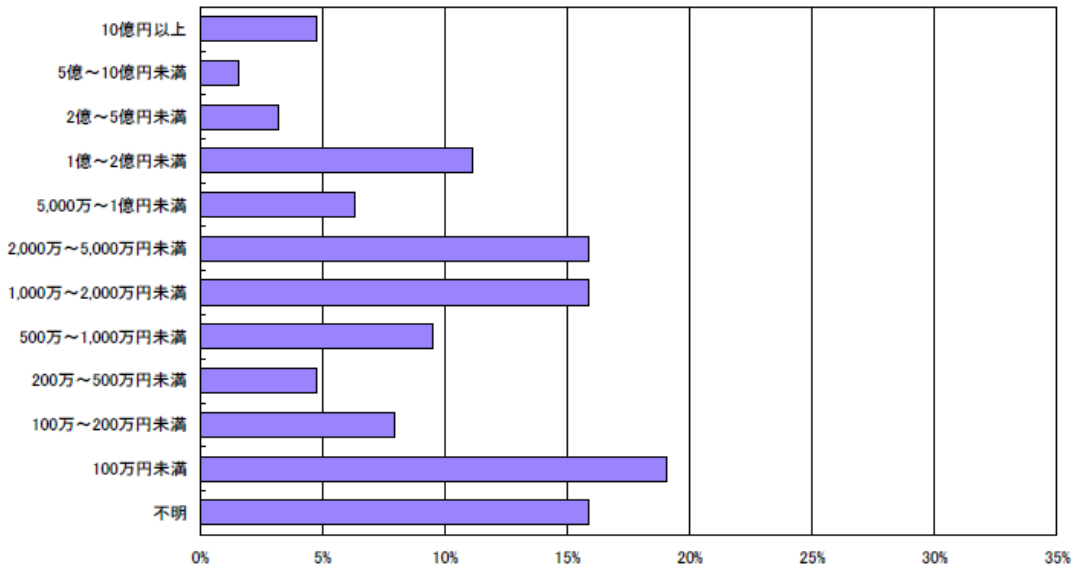


図 5-2:不具合が発生したことによる対策費の総合計(2008 会計年度)[55]

情報処理推進機構の報告書[56]には、ソフトウェアに起因するシステム障害事例として、決済に関わる事故、情報漏えい、重要インフラに関わる障害、メール等のWEBサービスの障害などにおける顕在化被害の事例が多数挙げられており、その深刻さを認識する上で参考となる。

一方、潜在化損失については、まだ世の中では、その規模について余り認識されていない。その規模の具体例を知ることは非常に重要である。情報システムのセキュリティに関わる損害事故の規模については、情報セキュリティ経済学の分野である程度の評価を行う手法が確立されてきている,。この分野では、事故の種別(不正アクセス、個人情報漏洩等)、企業規模等に応じて事故事例を分類し、それぞれの事例集合について、セキュリティ事故を要因とする企業価値の毀損額を評価し、セキュリティ事故によって企業価値が統計的に有意に毀損している場合の損害額を評価している。これらの結果に基づき、経済産業省の報告書では、情報システムの不具合等による情報セキュリティ事故(不正アクセス、機密情報漏洩等)に係わる日本の上場企業が抱える潜在リスクは 29 兆円と見積もっている[57],[58],。この結果に基づけは、上場企業について情報セキュリティ事故だけでも、無形資産[59]を含む企業価値の 5.4%のリスクを抱えていることになる。これは顕在化損失を上回る規模であり、上場企業1社当たり平均で 79 億円の潜在リスクを抱えているという計算になる[60]。

顕在化損失と潜在化損失の規模の関係を把握するために、簡単な試算をすると以下のように

[55] 2010 年版 組込みソフトウェア産業実態調査:事業責任者向け調査、経済産業省
[56] IPA, 高信頼化のための開発手法ガイドブックー予防と検証の事例を中心にー, 2010
[57] 経済産業省, 「グローバル情報セキュリティ戦略」, 2007
[58] この推定値は、情報セキュリティ経済学の分野において、企業の市場価値(無形資産を含む)に関する統計的に有意な結果(信頼度 95%)にもとづいたものである。
[59] 企業の市場価値から有形の純資産を除いたもの。
[60] 定量的な数値に関しては、環境や組織によって異なるが、大まかな規模感を知ることは、その重要性を認識する上で大切である。

なる。日本ネットワークセキュリティ協会(JNSA)の報告書[61]によると、情報システムに関わる個人情報漏洩事故（人的事故 27%を含む）の総被害額は、利益 10 億円、売上げ 100 億円のインタネットショップ部門の場合、3.8 億円と試算されている。一方、同規模のサービス・情報通信企業の情報セキュリティ事故による損害は、11.7 億円と試算される 。両者は、業種の違いなどはあるが、比較可能なものとして参考にすると、顕在化損失が 3.8 億円に対して、類似の事故で潜在化損失と顕在化損失を合計した企業価値全体の損失は、11.7 億円であるため、潜在化損失は、顕在化損失の大まかに2倍の規模になることが分かる。

情報セキュリティ以外のIT事故損害も考慮すれば、潜在リスクはさらに大きくなる。無形資産[62]を含む企業価値の毀損は、企業に対する信用失墜により将来に渡って企業のビジネスに大きな影響を与えることによるものである。その影響は、通常、リコールや製品回収コストなど顕在化損失よりも、大きいことを示している。以上のような状況を考慮すれば、IT事故による企業の損失は、一般に認識されている顕在化損失だけでは、不十分であり、上記のような損害規模の認識は極めて重要であるといえる。

障害発生時の影響度は、利用分野ごとに異なり一般化できないため、分野ごとにユーザ、事業者、発注者、ベンダーなどのステークホルダーが集まりリスクの洗い出しを行う必要がある。その際に、従来の調査研究結果から以下のような項目を参考に、具体的な障害を複数想定し、それぞれについて評価することにより幅広い観点からその影響度を把握することが重要である。

表 5-4：障害発生時の影響度を把握するための主な検討項目（参考例）

（文献[63],[64],[65],を参考に作成）

区分		項目	内容
人命への影響		死亡	死亡人数
		負傷	負傷の程度×人数
経済的な影響	顕在化損失	顧客の被害	被害内容（サービス停止、製品欠陥、料金誤請求、個人情報漏洩等）の重要度と件数の組合せ
		顧客対応コスト	問合せ・クレームの対応時間
		逸失利益	サービス停止による収益機会損失
		業務影響コスト	社内業務の低下・停止のコスト
		復旧コスト	復旧時間の人件費
		損害賠償・補償	損害賠償・補償費用

[61] JNSA, 「2003年度情報セキュリティインシデントに関する調査報告書」
[62] 無形資産：企業における特許等の知的財産、従業員の持つ技術や能力、企業文化や経営管理プロセスなど物的な実態を伴わない資産。
[63] 経済産業省, 「企業における情報セキュリティガバナンスのあり方に関する研究会 リスク定量化に関する検討資料」
[64] IPA, 「国内・海外におけるコンピュータウイルス被害状況調査 被害額推進報告書」(2004 年4月)
[65] JNSA, 「2003 年度情報セキュリティインシデントに関する調査報告書」

	潜在化損失	企業ブランド 価値毀損 (信用失墜)	信用失墜による将来の製品・サービス売上げ減少の正味現在価値(NPV : Net Present Value)

障害発生時の損害規模について具体的に検討する場合、以下のような項目について、情報システムのドメインに関する知見をもっとも持っているユーザ(事業者)が検討しなければならない。

5.3.1.2. 障害の発生可能性について

損害リスクのもう一つの要素である「発生可能性」は、フォーマルメソッド等を適用してソフトウェアの品質を向上させることにより減少させることができる。フォーマルメソッドの適用により、ソフトウェアの品質向上にもたらす効果について、従来のソフトウェア・テストでは得られない効果として主に以下のようなものがある。

表 5-5:フォーマルメソッドによる品質の向上に関する主な効果[66]

分類		フォーマルメソッドの効果	従来テストとの比較
性質保証		設計仕様が要求仕様を満たすこと[67]、与えた性質を設計仕様が満たすことを、形式検証が成功した場合、保証することができる。	保証したい性質について、いくらテストケースを増やしても、保証できない(ことが多い)。
不具合の検出	反例発見	与えた性質を満たさない反例を示し、その原因箇所の特定を支援する情報を示す。(モデル検査)	反例を検出するとは限らない。
	デッドロックの検出	デッドロックが存在する場合検出できる。(モデル検査)	デッドロックを検出することは困難。
	仕様の矛盾検出	矛盾がある場合に検出できる。	人手によるため、検出できるという保証はない。
	仕様の定義漏れ検出	仕様の定義漏れがある場合検出できる。	人手によるため、検出できるという保証はない。
	デッドコードの検出	発生しない場合(デッドコード)が存在する場合検出でき	人手によるため、検出できるという保証はない。

[66] EASIS, Marko Auerswald, Guidelines for the Development of Dependable Integrated Safety Systems, NASA,FORMAL METHODS SPECIFICATION AND VERIFICATION VOLUME I, VOLUME II 等の文献をもとに整理

[67] 6 章に示すとおり、Verification(検証)と Validation(妥当性確認)の区別が重要である。検証は、仕様どおりに正しく設計、実現されていることを検査し、妥当性確認は、仕様自体が正しいことを確認するものである。形式手法で性質を保証できるのは、Verification の意味である。

		る。(モデル検査)	
曖昧性の除去		曖昧な仕様による誤解や実装の誤りを回避できる。 暗黙知を明確化する。	曖昧性から、誤った実装につながる場合がある。

これらの効果について、具体的なイメージをつかむために、付録のケーススタディについて具体的な例を以下に示す。

表 5-6:効果の具体例

効果の分類(一部)	具体例(ケーススタディの例)	予想される影響等
性質保証	ブルーレイディスクの操作制御に関するミドルウェアの設計に対して、「要求されたディスク操作(API)は、必ずデバイスで実行され、待機状態に戻る。」という性質を記述として与え、形式検証が成功したため、その性質が必ず満たされることを保証することができる。	ブルーレイディスク以外の制御系(例:電力、航空機、鉄道)で、ユーザが要求した操作が実行されないと、それが安全に関わる処理であれば、大きな事故につながる可能性がある。
反例の発見	「全ての操作に関して要求された API は、必ずデバイスで実行される。」という性質には、再生操作 API に反例があることが発見された。	要求した操作が実行されない場合を発見して、改修することにより、そのような状況が発生することを減らすことができる。
デッドロック検出	操作APIの要求と、ドライブからの割り込みが同時に処理されるとき、制御状態を保持する共有変数に関してデッドロックが発生し、ミドルウェア全体が停止するという致命的な傷害が発生する。	デッドロックは、システム全体の処理が進行しなくなることを意味しており、重要インフラ系で同様なことが起きれば事故につながる可能性がある。
矛盾の検出	ミドルウェアを構成する2つのモジュールレイヤーがそれぞれ保持する動作に関する状態は整合性が取れているという性質は、満たされないことが示された。(当初から予想されていたが、厳密にそれが示された。)	あらかじめ想定された状況の発生が確認されたものであり、そのような状況でも不具合が発生しないことを検証する必要がある。
曖昧性の除去	自然言語による仕様書から、形式仕様記述を行う際に、59 件の曖昧な記述を見つけた。	設計仕様の曖昧性により、設計仕様からコーディングの過程で、意図しない処理が行われる可能性がある。

これらの例は、いずれも従来のテストだけでは検出することが困難な不具合であり、フォーマルメソッドを適用することにより、ソフトウェアの品質向上の効果を特徴的に示している。ただし、フォーマルメソッドは、従来のテストを丸ごと置き換えるものではなく、従来のテストと補完関係にあり、双方の有効な部分を組み合わせることで、全体の品質向上を実現することが経済的にも効果的であることに留意しなければならない[68]。

機能安全の分野では、ソフトウェアの設計や実装の不具合の存在確率については定量的な評価が困難であるため、開発の手順と適用する手法によるプロセスを認証することで安全性を確保するという考え方が一般的になっている。フォーマルメソッドの適用が、システム障害の発生確率の低減にどの程度の効果があるかについては、正確には定量化できないが、障害が発生した場合の損害額の大まかな規模をつかむことは重要である。機能安全標準 IEC61508 の考え方を参考にすれば、リスクの大きさに応じて、ソフトウェアのユーザ（事業者）が、ソフトウェアの品質レベルを設定し、その結果に対して責任を持つことが求められる。

フォーマルメソッドによる検証等による具体的な効果を整理したものが下図である。これらなども参考にその効果を現場で評価するとよい。

[68] Jonathan Bowen, Ten Commandments of Formal Methods, Ten Years Later, 2006

効果の分類	開発現場における現状の問題点	形式手法により期待される効果	具体例
不具合対策			
不具合の発見	・並行プロセスのタイミングに依存する不具合は再現性が低く、発見が困難。 ・検査したい性質について、系統的に場合分けを網羅するテストケースの生成が困難。 ・そもそも、タイミングに係わる不具合の可能性さえも気が付かない場合が多い。 ・バグを減らすために、並行プロセスや条件分岐を減らすなど消極的なプログラミングを強いられる。 ・テストツールの自主開発は、コストがかさみ、また、転用が難しい。	・網羅的に検査するために、検査で不具合が見つからなければ、正しさが保証される。 ・モデル検査では、系統的、網羅的に自動検査が可能であるため、効率的。 ・不具合が存在する場合、不具合に至る実行トレースが提示されるため、原因を究明できる。	・ブルーレイディスクにおいて、操作要求とデバイス割込みが並行して発生する場合に、デッドロックが発生することがある。 ・再生実行が、処理されず停止に戻る場合がある。(通常の操作では発生しないことを確認)
不具合の原因特定	・不具合の存在が分かっても、再現性が低い場合、毎回挙動が異なるため、原因の特定が困難。 ・めったに再現しない不具合は、コストの制約から修正できないまま出荷せざるを得ない場合もある。 ・サイクロマチック数が30を越えると、設計した本人でも修正が困難となる。	・再現性が低い不具合であっても、網羅的な検査を行うため、不具合の原因特定が可能。 ・処理の分岐が複雑であっても、系統的に自動検査可能である。	・ブルーレイディスクの動作におけるデッドロックに至るトレースを追跡することで、原因が特定できた。
不具合の修正の影響分析	バグの修正の影響範囲は人手によるコードレビューに頼っている。	一度、検証性質を記述すれば、モデルの修正後、繰返し検査に利用することができる。	
検証			
与えられた性質を満たすことを保証	・並行動作のタイミングに依存する不具合は、テスト回数をいくら増やしても、完全な保証が得られない。	・網羅的な検査を行うため、不具合が検出されなければ、与えられた性質に関して正しさが保証される。	・ブルーレイディスクの操作制御モデルに、デッドロックが発生しないことを保証できる。
検証結果の評価に対する科学的根拠	・テストツールを開発しても、網羅性の保証は困難で、テスト結果に対する評価は、ベテランの感に頼ることが多く、科学的な根拠が示せない。 ・テストカバレッジを計測する方法が普及しておらず、パスを通った結果が正しいか人手に依存する。	・検査結果に網羅性が保証される。 ・形式仕様から、テストカバレッジを自動的に計測することができる(VDM)。	・ブルーレイディスクの制御モデルに関して、デッドロックが発生しないという性質を科学的に保証できる。 ・ファイルシステムにおいて、ファイルオープン操作のテストカバレッジが計測できる。(VDM)
従来テストでは扱えない性質(安全性、到達性等)の検査	・従来テストでは、状態の網羅的な検査が必要な安全性や、実行の無限系列に関する到達性に関する性質が扱えない。	従来テストでは、扱いにくい性質を時相論理式として表現し、検証が可能。	・「どのようなリクエストに対しても、いずれは待機状態に戻る」という性質は、従来のテストでは扱いが困難。
妥当性確認			
仕様の曖昧性排除	・仕様書の曖昧性により、設計者と開発者の異なる思い込みにより、バグが入り込む。 ・要求仕様の曖昧性により、発注者の意図と設計者の理解にズレが生じることがある。	・性質に関する記述を、条件の論理的な組合わせ(かつ、または、否定などの構造化)で表現することにより、条件の範囲や掛かり方が明確化される。 ・条件が満たされる時間の範囲や、検証対象とする実行経路の範囲や、係り受けが明確化され、誤解が回避される。	入退室管理システムで「閲覧室が空室のとき以外、常に案内は変更されない」という表現は、「閲覧室が空室の時以外、決して案内は変更されない」の方が意味が明確。 「常に性質pを満たさない」という表現は、『「いずれは、性質pを満たす。」の否定』、または、『「常に性質pを満たす」の否定』のいずれであるか明確化が必要。

図 5-3:開発現場が抱える問題点とフォーマルメソッドの適用で期待される効果の例

5.3.2. コスト削減

フォーマルメソッドの適用によるコスト削減に関する効果については主に以下のようなものが挙げられる:

- 設計、要求分析工程で、実装する前に検証やシミュレーションなどの解析を実行することにより仕様の不具合を検出でき、手戻りコストを削減する。
- 設計、要求を、曖昧性のない厳密な言語で記述することにより、コーディング等の後工程における誤解などによる不具合の混入を防ぐ。

形式仕様記述言語を用いて要求分析、設計を行うことにより、コーディングを行う前に上流工程で、それらの仕様の検証、シミュレーションが可能になり、早期の不具合除去に有用である。図 5-4 は、ソフトウェアプロジェクトに関するNISTの調査に基づきCMU/SEIが作成したソフトウェアの除去コスト等を定量的に評価したものである[69],[70],[71]。

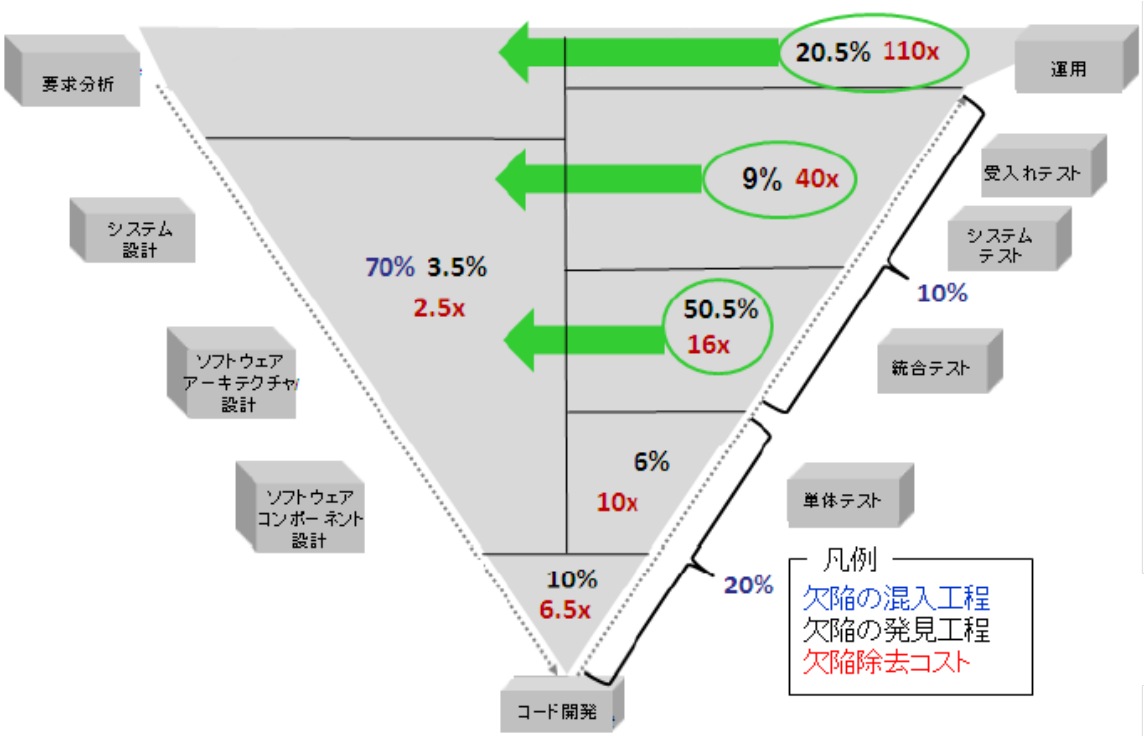


図 5-4:欠陥の混入工程、発見工程、除去コスト

(Peter H. Feiler et al, System Architecture Virtual Integration: An Industrial Case Study,

[69] CAD/CAM/CAE/PDM(of computer-aided design/computer-aided manufacturing/computer-aided engineering/product data management software)の 179 ユーザと 10 のソフトウェアベンダーに対するインタビュー調査結果。

[70] Peter H. Feiler et al, System Architecture Virtual Integration: An Industrial Case Study, November 2009, TECHNICAL REPORT CMU/SEI-2009-TR-017

[71] NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing, 2002.

November 2009, TECHNICAL REPORT CMU/SEI-2009-TR-017 を元に作成）

これによれば、欠陥の 70%は、設計工程までに混入し、そのうち設計工程までに検出される欠陥は、3.5%に過ぎず、全体の 50.5%は、統合テスト工程で発見されることが示されている。欠陥を発見する工程が下流であればあるほど手戻りコストが増大し、統合テストでは、16 倍、システムテストでは、40 倍にかさむことが示されている。このようなことから、フォーマルメソッドを用い開発プロセスの要求と設計の上流に重点を置くことで、開発コストを下げる効果が期待できる。また、航空機、鉄道分野では、形式仕様記述言語を用いて設計を行うことにより、設計記述からプログラムソースコードを自動生成し、従来テストを大幅に減らす事例が多く、トータルでのコスト削減に効果が得られている例もある[72],[73]。例えば、航空機のフライト制御ソフトウェアの場合、設計にフォーマルメソッドを用い、自動コード生成を行うことにより、コーディング、レビュー、テストのコストを 70～90%削減できるという事例が示されている[84],[74]。

フォーマルメソッドの導入は、必然的に上流工程の要求分析、設計に重点を置くことになる。さらに、実装を記述することなく、要求仕様や設計仕様に対して検証や解析を行うことが可能になり、上流工程の成果物の品質を向上させることができる。McDonaldの文献[75]では、上流工程を重点化する欠陥予防開発プロセス(Defect Prevention focused Process)は、下流工程のテストに時間をかけるテスト中心プロセス(Test-Centric Process)よりも生産性が2倍に達することを示している。

本書付録のケーススタディにおいては、フォーマルメソッドを適用しない場合のテスト工数に対して、モデル検査の適用により、制御アルゴリズムに関する不具合が設計時点で除去できることから計算して、テスト工程が、2ヶ月から1ヶ月に短縮できることが示された。この効果は、テスト工程における不具合の発見により、手戻りが削減されることを考慮すればさらに大きな効果になると言える。一般的なソフトウェア開発において、テスト工程が全体の50%以上を占める[76]という実態からしても、テスト工程の短縮効果は大きいと言える。

一方で、フォーマルメソッドによる設計および検証は、プログラムを抽象化したレベルで実施し、検証された設計に基づき、人手によりコーディングを行うケースもある。このような場合、テスト工数は、幾分削減することは可能かもしれないが、トータルの工数としてはフォーマルメソッドを導入しない場合に比べて増大する場合が多い。しかし、このような場合であっても、設計が検証されていることによりソフトウェアの品質向上の効果が設計コストの増加を上回るならば採算性があると判断される。

従来の下流工程のテスト中心開発プロセスから、欠陥予防開発プロセスに移行するには、未経験の新しいプロセスを取り入れる際の不安や回避行動があるため、採用が進まないのが現状であ

[72] Model-Based Development for Critical Embedded Software in Aerospace & Defense, Esterel Technologies.
[73] IPA, 形式手法適用調査, 2010
[74] ROI Analysis, Esterel Technologies, 2009
[75] The Practical Guide to Defect Prevention, Marc McDonald, Microsoft Corporation, 2008
[76] ROBERT M. HIERONS, et.al., Using Formal Specifications to Support Testing, ACM Computing Surveys, Vol. 41, No. 2, pp. 9-76, 200

るが、上流工程に重点をおいた開発プロセスの重要性が理解されれば、フォーマルメソッドも自然に受け入れられるものである。

5.3.3. リスク低減

リスク低減の効果については、見落とされがちであるが、重要な要素である。管理者の視点からは、プロジェクトが予算通りに実施されること、工数の大幅な変動が発生せず、納期が守られることは重要である[77]。また、開発プロセス全体のうち、特定の工程に、負荷が集中するボトルネックが存在する場合、全体の生産性に悪影響を与えると共に、人員の稼動状況に非効率性が発生するなどの問題点が指摘されている[78]。

フォーマルメソッドにより開発の上流工程に重点を置いた開発プロセスのフロントローディングを実現することで、従来のテスト工程のピーク負荷を緩和し、プロセス全体のピーク負荷を抑えるとともに、手戻り発生等による工数の大幅な変動リスクを抑えることにも役立つ。図 5-5 は、EU のフォーマルメソッド実践プロジェクト EASIS の資料に基づき、フォーマルメソッドを用いた開発のフロントローディングにより、ピーク負荷の工程は、従来のテスト工程から、上流の設計工程にシフトし、プロセス全体のピーク負荷が低減されること様子を示している。フォーマルメソッドを用いたこのようなフロントローディングは、航空機分野や鉄道分野など特定の分野においてはある程度実現されつつある。

このように、開発フロントローディングによる生産性の向上、納期遅延などのリスクの回避などの効果をもたらすものであるが、フォーマルメソッドの導入は、上流工程で厳密に設計し、検証を行うため、必然的に開発フロントローディングを実現することにつながるといえる。

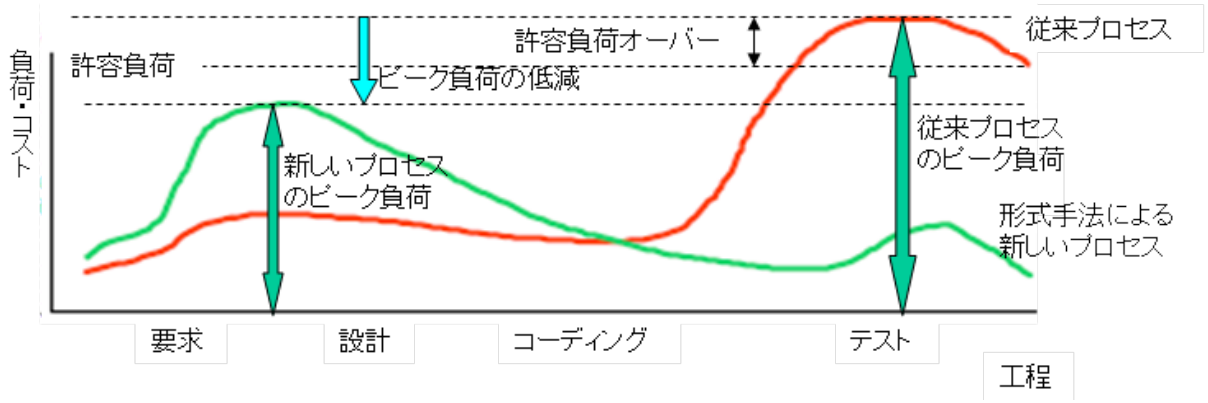


図 5-5: 開発プロセスのフロントローディングと許容負荷
（文献 66 を元に作成）

多くのフォーマルメソッドの文献に書かれているように、フォーマルメソッドを開発の上流工程に

[77] これら以外にも、Capers Johons は、ソフトウェア開発だけでなく保守も対象にしたリスクに関して洗い出し、顧客の要求仕様変更など60のリスクをあげている 。これらについてもリスクを低減する上で参考になる。
[78] 制約理論（TOC）では、プロセス全体の生産性は、ボトルネックの改善によってのみ実現されるとされる。

適用することで工数の削減効果とともに、開発工数の変動リスクを低減する効果が期待できる[79]。

5.4. フォーマルメソッドの投資コストについて

フォーマルメソッド導入のための投入コストには、主な要素として(1)学習コスト、(2)設計コスト、(3)ツール導入コストがある。以下では、それぞれのコストについて具体的な規模の例示と考え方について示す。

5.4.1. 学習コスト

手法の学習コストは、手法の種類と学習の達成レベルによって異なる。手法の種類による学習コストの違いとして、日本語の学習用入門書が比較的充実している手法は習得が容易になるというメリットがある。例えば、SPIN, Bメソッド、NuSMV、VDM++などの文献は比較的多い[80]。これらの手法の基礎的な技術の習得は、入門書を数冊読むための時間から学習コストを見積もることができる。手法の習得は、必ずしも最初に上級レベルまで一気に習得する必要は無い。初級レベルに到達した段階で、試行的にフォーマルメソッドの適用を始められるが、実践を繰り返しながら、ノウハウを習得するとともに、上級参考書によりスキル向上を図らなければ、フォーマルメソッド適用により効果が得られるレベルに到達しない。また、成功事例とされる多くのプロジェクトは、フォーマルメソッドの適用時にフォーマルメソッドの専門家に定期的にアドバイスを受けられる環境にあることを挙げている[81],[82]。海外にはフォーマルメソッドの導入コンサルティングを行う企業も出てきている[83],[84]。成功している適用事例は、導入初期には、このようなコンサルティングサービスを利用しているものが多い。それは、高度で特殊なスキルをすべてマスターしてから適用するのでは初期コストが大きすぎるため、早い段階で実践を開始し、問題に直面した際に、外部専門家から指導を受けることで必要なスキルを効率的に習得することを目指す。

これらの手法の入門レベルの習得は、フォーマルメソッド特有の概念を理解する点で難しさはあるが、Java 言語などプログラミング言語と比較して、特別多くの時間を要するという訳ではない。

本プロジェクトケーススタディでは、SPIN を用いた制御設計の検証に関連して技術習得のために以下のような工数が発生している。

---

[79] Abernethy, Technology Transfer Issues for Formal Methods of Software Specification, NASA, 2000.
[80] 学習者向け参考書は第 17.1.3 章等参照。
[81] Jonathan Bowen, Ten Commandments of Formal Methods ... Ten Years Later, 2006
[82] Jonathan Bowen, Ten commandments revisited a ten year perspective on the industrial application of formal methods, 2005.
[83] ClearSy, http://www.clearsy.com/index-en.php
[84] Esterel Technologies,
http://www.esterel-technologies.com/services/fast-ramp-up-services/methodology-preparation

表 5-7:SPIN の技術習得コストの例

		学習時間(h)	実践適用時間(h)
被験者1	入門レベル	34	176
被験者2	入門レベル	63	370
被験者3	上級レベル	562	143

フォーマルメソッドの習得コストを、単一のプロジェクトで回収しようとすることは、合理的な考え方とは言えない。それは、プログラミング言語の習得を単一プロジェクトのためだけに行うわけではないことと同じである。フォーマルメソッドの初期の学習コストを考えると、極めて大規模なプロジェクトあるいはきわめてクリティカルなシステムでなければ、単一のプロジェクトで採算は取ることは難しい。プログラミング言語と比較して、フォーマルメソッドにおける採算性評価で難しい点は、フォーマルメソッドの効果に関して 5.3 章で示したように、損害リスクの低減がドメインや対象システム、利用環境に依存して評価が難しいためである。したがって、採算性の評価結果は、損害リスクの低減効果がどれだけ適切に行われるかによって大きく異なる。その点からも、ドメイン知識を持つ導入者が判断することが重要になる。

5.4.2. 設計コスト

フォーマルメソッドを用いた設計は、その詳細度により効果や影響が異なる。大まかに以下の2つのレベルに分けられる。

表 5-8:フォーマルメソッドを用いた設計レベル

レベル	内容	影響
基本設計レベル	アルゴリズムの一部などを抽象化した設計レベル[85]。	形式仕様記述言語で記述した基本設計をもとに、マニュアルによる詳細設計、コーディングを行う場合、基本設計レベルで形式検証を行った結果は、そのままでは保証されない。
詳細設計レベル	ソースコードが生成できる程度に詳細なレベル。	形式検証を行った詳細設計から、コード生成を行う場合、コード生成系が Qualification 済であれば、生成されたソースコードの信頼性は高い。

[85] モデル検査等のフォーマルメソッドの応用では、基礎設計レベルでは、アルゴリズムやそれを抽象化したものを対象とすることが多く、アーキテクチャ設計への応用はこれまであまり実績がないため、ここではアルゴリズムを中心に考える。

設計コストの増加は、これらの設計レベルおよび適用する手法により異なる。Bメソッド、VDMの場合、詳細設計レベルまで形式仕様を記述する場合が普通である。その場合、自動コード生成系を使うことが可能であり、生産性の向上が期待できる。

表 5-9 は、付録のケーススタディ(ブルーレイディスク、入退室管理システム)において、 制御に係わる設計仕様についてSPINを用いた形式モデリングと検証を行った際の追加工数である。フォーマルメソッドでは、設計段階で、検証まで行えるためそれらの工数も含めたコストを考える。

表 5-9:ケーススタディにおけるフォーマルメソッドの設計付加コスト(時間)

	ブルーレイディスク	入退室管理システム
設計書の理解	196	150
モデリング	40	70
モデルの形式記述	40	80
検証性質記述	3	20
不具合特定等	40	50

SPINなどモデル検査系では、状態爆発の問題があるため、抽象レベルの設計を対象とする場合が多い。この場合、形式検証済みの設計仕様に対して、人手により詳細設計やコーディングを行うことになり、人手によるプロセスから不具合が混入する可能性がある。

一方、設計と検証を比較的厳格に行う手法として代表的なBメソッドに関してフランスの鉄道分野で実践したプロジェクト[86]における設計コスト、工程全体の比率を示したものとして以下のものが参考になる。

表 5-10:鉄道システムに対するBメソッドの適用プロセスの工数比率

工程		工数比率	
プロジェクト管理・その他		13%	
抽象モデリング	対象分析等	55%	18%
	インスペクション		5%
	証明		16%
	その他		6%
具体モデリング	証明	24%	11%
	その他		!3%
最終管理(文書、確認等)		8%	

[86] Using B as a High Level Programming Language in an Industrial Project: Roissy VAL, ClearSy and Siemens Transportation Systems

フォーマルメソッドの産業応用に関する調査 によると、適用事例ごとに、適用した対象の範囲、適用工程、適用の深さ(適用した設計の詳細度)はまちまちであり、決して対象システムの全てのコンポーネント、全ての工程に一様に適用しているわけではない。対象ごとに、フォーマルメソッドの適性、限界などを考慮し、目的やリスクに応じて、適用範囲や適用の詳細度を選択することが成功の重要な要素とされている。

フォーマルメソッドの導入には、技術の習得に一定の初期コストが必要になるため、単一のプロジェクトで採算性をとるのではなく、複数のプロジェクトで利用することを想定した組織としての採算性を考えることが合理的と言える。実際、フォーマルメソッドの適用コストは、初回から、2 回目で劇的に下がることがいくつも報告されている[87]。

5.4.3. ツール導入コスト

フォーマルメソッドは、無償のツールが比較的多い。実践で比較的多く利用される B メソッド、VDM++に対応したツールやモデル検査系の SPIN, NuSMV は無償で提供されている。

一方、モデル検査系を含む SCADE や SAL など有償のツールも存在する。有償のツールの中には、エディタ、検証ツール、要求トレーサビリティ支援機能など開発に必要な一連の機能を統合した開発環境として提供するものもある。これらのツールの中には、さまざまな機能オプションやサービスと組合せとして価格がきまるものがあり、個別に費用の見積もりが必要になる場合がある。いずれにしても、ツール導入コストは、フォーマルメソッド導入前に、判断することができる。

5.5. 導入判断における留意点のまとめ

本章で示した考え方、留意点をまとめると以下の通りである。

- 5.2 章を参考に効果とコストの主要な要素と全体像を把握する。
- 効果とコストの主な要素について、5.3 章に示した具体例や考え方を参考に、ドメイン知識に基づき大まかに規模を評価・把握することが重要。特にソフトウェアの不具合による損害リスクのうち、情報システムに関する事故による企業価値の毀損を含めて、大まかな規模を把握することは重要。
- 組織として複数のプロジェクトで採算性を確保すると考える。プログラミング言語と同様に、単一のプロジェクトで、採算をとることは現実的な考え方ではない。
- コストと効果は、適用対象、適用手法、適用範囲(広さ)、適用レベル(深さ)などによって大きく異なるため、応用事例集(12 章)の情報なども参考に、対象システムとの類似性、適用レベル、効果の概要を把握して、効果とコストの規模に関する感覚をつかむ。

[87] Woodcock, J., Larsen, P. G., Bicarregui, J., and Fitzgerald, J. 2009. Formal methods: Practice and experience. ACM Comput. Surv. 41, 4 (Oct. 2009), 1-36.