

実務家のための形式手法

厳密な仕様記述を志すための形式手法入門 第二版

なぜ形式手法か

なぜ形式手法か(このモジュールについて)



品質の高いソフトウェアの効率よい開発へ向け、形式手法の有用性を理解した上で、形式手法の導入に前向きに検討する姿勢の獲得を意図したモジュール。特に正しい仕様が重要であることを理解する。

■ 事前知識・経験

- 旧来のソフトウェア開発に対する問題意識(ソフトウェア開発の実経験がある
とより好ましい)
- 形式手法の知識・スキルは前提としない

■ 学習目標

- 形式手法とその効果について概略レベルの知識を得る
- 旧来の開発の課題に対する形式手法の有用性を理解し、導入の検討を開始できる
- 教材中の例や自身の経験との関連付けにより、仕様の重要性を理解し、まず正しい仕様の記述に焦点を当てた形式手法導入の検討を開始できる

なぜ形式手法か(このモジュールについて)



■ 主な学習項目

1. 背景 p4
2. 形式手法の定義と期待できる成果 p6
3. ソフトウェア開発の課題と形式手法 p9
4. ソフトウェアの信頼性・安全性への期待の高まりと国際規格・標準での形式手法の推奨 p15
5. 慣習的手法とその限界、およびその解決に有用な形式手法の特性 p23
6. 現場向きの形式手法VDMの概要 p34
7. 自然言語や慣習的な非形式的記述による問題例 p39

(見出しに※がついているスライドは管理者コースや初回学習時にはスキップ可。)

講師の方へ:

各学習項目ごとに以下の点を意識し、斜体部分に注意してお話ください。

1.どのようにしてソフトウェア品質を効率よく確保するか

効率良く品質の高いソフトウェアを開発するための、いわゆる”銀の弾丸はない”中で、科学的に裏づけのある手法の利点を強調してください。

2.形式手法がなぜ着目に値するのか

形式手法は科学的に裏づけを持つ手法で、実際の開発で成果を上げている点を強調してください。

3.慣習的な現場の進め方で効率よく正しいソフトウェアを正しく開発できるのか

自然言語や慣習的な記法を用いることによる問題の解決に形式手法が有効な点を強調してください。コンピュータで動作させるための本質的な数学的性質と仕様記述の目的に注意させるとよいでしょう。

4.ソフトウェアの信頼性・安全性への期待の高まりにどう応えるのか

形式手法は客観的な説明や評価の点から有効で国際規格・標準でも推奨されている点を強調してください。

詳細レベルでは、分野、規格、形式仕様の論理体系による特性の違いに注意してください。

5.慣習的な手法の限界は何か、その原因は何か

仕様の記述に曖昧さがなくなることによってどういった利点があるか

形式手法は検証・証明に強みを持つが、まず厳密な仕様を記述することが重要でその間接的效果も大きいことを強調してください。

慣習的手法との対比では講師や受講生の経験、レビューやテストといった他の手法との比較が有効です。

直接的な成果物である設計情報だけでなく、各メンバの開発対象に対する理解、チーム内またはチームやステークホルダの間での情報共有、といった間接的な面での効果が上がった事例があります。

6.現場向きとされる形式手法にはどういった特徴があるか

本教材ではVDMを取り上げておりその概要の説明でよいでしょう、より詳細な説明は別の教材モジュールにあります。

7.日本語仕様の曖昧さの例にはどのようなものがあるのか

具体的な例題により、可能なら受講生自らに例を考えさせ、日本語仕様の曖昧さとはどのような場面で表れるかを実感させてください。

1. 背景

科学的な裏づけを持つアプローチの必要性



- ソフトウェアの信頼性や安全性に対する関心が高まってきた
- これまで以上に効率良く開発することが求められてきている
- 普通の現場で開発に携わっているのは**普通のチームとメンバ**
- 現場にプレッシャーをかけても品質は高くなり、ストレスばかりがかかる
- 効率良く品質の高いソフトウェアを開発するための「**銀の弾丸**」は**ない**
- 形式手法は科学的に裏づけを持つ手法

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

5

ソフトウェア開発の期間、品質への要求が高まっているが、具体的な方策なしに精神論だけでは成果も期待できずむしろ弊害が大きい。

一瞬にして効果を出す魔法のような何かを期待することも現実的ではなく、現状を冷静に分析し、その対策を工学的・科学的に妥当な方法で考える必要がある。一般に、ソフトウェア開発では正確性、厳密性などが重要であるが、例えば正確性及び厳密さを追求する必要がある部分に、自然言語を使うと、該当部分に関して表現したり、読んだり、コミュニケーションをはかったりする際に、困難がつきまとう可能性がある。慣習的な開発ではそのような問題への対策がたてられていないことが考えられ、科学的に裏づけのある方法を検討することが有効と考えられる。

「銀の弾丸」はない：No Silver Bullet - Essence and accidents of software engineering (1986),
Frederick P. Brooks, Jr., IFIPでの論文。

2. 形式手法の定義と期待できる成果

形式手法(Formal Methods)とは



- 形式手法とは、システム開発に用いられる手法で、**数理論理学**に基づく科学的な裏付けを持つ仕様記述言語を用いて設計対象を表現することにより、**ある側面の仕様を厳密に記述**し、開発の**各工程で利用**する手段の総称である
- 形式記述を行うことで曖昧さが取り除かれ、**機械処理が可能**になり、様々な可能性が開ける
- 形式手法の一分野であるモデル規範型の手法と、形式仕様記述言語、仕様の記述や検証のためのツールの活用により、厳密な仕様の記述や検証が可能となる上、形式手法の導入は、システム開発におけるステークホルダ間の**コミュニケーションの活性化**に寄与する

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

7

形式手法の特徴は、数理論理学に基づく科学的な裏付けを持つ仕様記述言語を用いて、設計対象を表現することにより、客観的な取り扱い、場合によってはツールによる機械処理が可能となり、問題の予防や除去に対して様々な可能性が開けるところである。

形式仕様で検証できるのは、それぞれの形式手法の基礎となる数理論理で表現された側面のみであるが、複数の形式手法を組み合わせることで、システムの多面的な検証が可能となる。

また、厳密な記述を行うことで、ステークホルダ間のコミュニケーションで誤解を生じにくくする。

形式手法による仕様作成の成果例							
	対象システム	適用工程	仕様規模	欠陥数	生産性	開発期間	備考
(1) SCSK	証券業務	実システム UseCaseレベル 要求仕様	3+3万行 仕様+テスト ケース	0 品質	2.5倍 効率	45%	開発チームに 業務知識無し
(2) フェリカ ネットワークス	おサイフ ケータイ ファーム ウェア	実システム API外部仕様	7.4+6.6万行 仕様+テスト ケース	0	2倍	85%	開発チームに 形式手法知識 無し
(3) 産業技術 総合研究所, オムロン	鉄道関係 駅務シス テムデー タ	既存システム 厳密仕様 作成実験	0.75+80万 行 仕様+業務	欠陥 発見 数 29			「暗黙知は 仕様の欠陥」

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

8

(1)と(2)は形式手法の一種である VDM を[実際のシステム開発](#)に適用した例である。

SCSK 株式会社(当時:株式会社CSKシステムズ)とフェリカネットワークス株式会社の例は、対象システム、チーム構成などが対照的な面もあるが、高い品質と効率を達成している。

なお、ここでの欠陥数は、リリース後に発生した仕様の不備を原因とする欠陥の発生数であり、生産性と開発期間の数値は、プロジェクト開始時点での見積もりと実績値との比較である。

3. ソフトウェア開発の課題と形式手法

ソフトウェア開発における曖昧な仕様の問題



- ソフトウェアの開発現場では、曖昧な仕様起因するトラブルが多い
- 下流工程の修正コストは、上流工程よりも大きくなる
- 正しくないプログラムの欠陥修正は非常に難しい、または不可能である

- 自然言語により、曖昧かつ不完全な仕様を記述している
- 仕様記述と称して、日本語プログラミングを行っている
- 対象業務の内容をよく知らないプログラマがプログラミングを行っている
- 結果として、重大な欠陥を含み、保守性・再利用性のないソフトウェアが生産されている

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 10

ソフトウェアの開発現場では曖昧な仕様起因するトラブルが多い。

下流工程の修正コストは上流工程よりも大きなものとなる。

この理由としては、自然言語による曖昧さを残した仕様(暗黙知が通用せず、暗黙知を形式知にしている)に従ってプログラミングしている現状がある。

仕様書の現実※



- 口頭
- ホワイトボード
- 「パワーポイント」独自スタイル
- 表
- UML
- 自然言語
- 「VDM」、形式仕様記述言語

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 11

(※このスライドは管理者コースや初回学習時ではスキップも可能)

どこまでが仕様書と呼べるのか？

今の現場では仕様書とも呼べないものが使われているのではないかな。

人手によるチェックの限界※



- 「これから配布する文書の、ある文字の数をかぞえる【7 分間】

(※このスライドは管理者コースや初回学習時ではスキップも可能)

A4 2 ページの文章中の「の」の数など。

単純な作業だが、なかなか正確に数えられない。

なぜ形式手法か？経験的根拠



■ 日本語で正しい仕様を書くのは「不可能」である

- 曖昧さの排除が困難
- ツールによるチェックが、ほとんど不可能
- 仕様の修正に弱い
- その場で「文法」を考えられない

□ 日本語で仕様を書ききれなくなり、if-then-else など擬似コード的な仕様を書くことが多く、**擬似コードの文法を考えている時間が馬鹿にならない**

■ 形式手法の方が技術移転や蓄積が容易

- 日本語による意思疎通は難しい
- 仕様記述言語を読むのは容易
- フレームワークやライブラリの構築が容易
- 業務知識の蓄積が容易
- 仕様を「動かしてみる」ことができるので、理解しやすい

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 13

日本語による仕様記述の問題として、

- ・修正した時の**変更余波の追跡**が非常に難しい
- ・ツールによるチェックはほとんど不可能
- ・結果として、**曖昧さの排除が困難**

が挙げられる。

仕様の中で擬似コードを使うこともしばしばあるが、複数の箇所で一貫性をもった表記にするためには文法規則が必要になり、時間をとられる。

形式手法による仕様は言語仕様も小さく、**動かせるような記述も可能**なので、理解するのは容易であり、技術移転や蓄積が容易である。

UML1.* や UML2.0 より簡単だという意見もある。

なぜ形式手法か？理論的根拠



■ プログラムは数学系だから

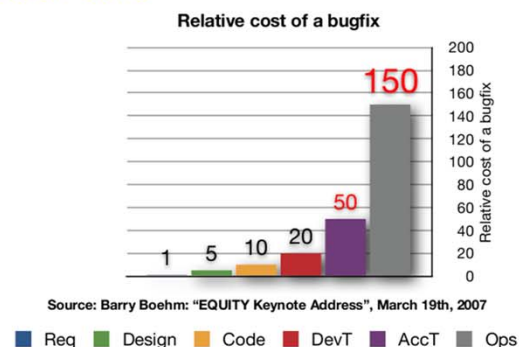
- しかも、CやJavaのプログラムは「**検証が困難な数学系**」
- 要求仕様記述工程からプログラミング工程までのどこかで、数学系に変更しなければならない
- **検証しやすい数学系で記述した方が、品質を上げやすい**

■ 形式手法では、上流工程で仕様を検証

- 欠陥修正コストが10分の1から200分の1で済む([Boehm:07],他)

■ 旧来の手法では、検証が困難

- **レビューだけでは限界**がある



SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 14

C や Java といった**副作用をもつ言語**を検証する論理体系は複雑であり、事実上検証が困難である。

また、上流工程の検証の方が欠陥修正コストが小さく、品質を上げやすい。

DevT = Development Test

AccT=Acceptance Test

Ops = Operation

【さらに学習したい人に】

[Boehm:07]

Barry Boehm: "Revisiting Software Engineering Economics", EQUITY 2007 keynote, March 19th, 2007

ソフトウェア開発ライフサイクルにおける生産性のパラメータについて言及している。

4. ソフトウェアの信頼性・安全性への 期待の高まりと国際規格・標準での 形式手法の推奨

社会基盤としての計算機システム



■ Safety Critical System／Mission Critical System

- 原子力発電所制御システム
- 航空管制システム
- 鉄道信号制御システム
- 放射線治療システム

■ 日常生活

- 銀行オンラインシステム

■ ディペンダビリティ (dependability)

- 信頼性、安全性、頑健性、説明可能性、等々

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 16

情報システムとそれらのネットワークは、もはや我々の日常生活を支える不可欠な社会基盤になっている。

Safety Critical System = 絶対間違いが起きてはいけないシステム

Mission Critical system = 絶対異常終了してはいけないシステム

原子力発電所や医療機器のように直接地球環境や生命に関わるだけでなく、銀行オンラインシステムが動かなくなっても困る。

現代の IT システムには信頼性、安全性、説明可能性などを含む高いディペンダビリティが求められており、技術者はそれに応えるものを作るだけでなく、それらを説明できなければならない。

ディペンダビリティ: 利用者が IT システムに依存して、安心して安全に日常生活を送ったり、仕事をしたることができること。

【さらに学習したい人に】

片山 卓也: 社会基盤ソフトウェアシステム研究開発の推進, 総合科学技術会議重点分野推進戦略専門調査会情報通信研究開発推進プロジェクトチーム会合(第3回) 資料3

ソフトウェアの不具合の例



- 鉄道改札機
- 東京証券取引所
- みずほ銀行
- 航空管制システム
- 搭乗手続きカウンタ
- 電話交換機
- 携帯電話リコール

etc.

「情報システムの障害データ. 情報システムの障害状況」

SECジャーナル, No.27, pp.150-152, 2012

SECジャーナル, No.28, pp.6-8, 2012

SECジャーナル, No.30, pp.139-141, 2012.

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 17

しかし、実際にはさまざまな不具合が生じている。

これらは、近年報道された社会基盤としての計算機システムの不具合である。

【さらに学習したい人に】

松田 晃一, 金沢 成恭: 情報システムの障害データ. 情報システムの障害状況 2011年前半データ, SECジャーナル, No.27, pp.150-152, 2012.

松田 晃一, 金沢 成恭: 情報システムの障害データ. 情報システムの障害状況 2011年後半データ, SECジャーナル, No.28, pp.6-8, 2012.

松田 晃一, 大高 浩: 情報システムの障害データ. 情報システムの障害状況 2012年前半データ, SECジャーナル, No.30, pp.139-141, 2012.

IPA/SECのホームページ(<http://sec.ipa.go.jp/secjournal/index.html>)からダウンロード可。

情報技術セキュリティ評価基準 ISO/IEC 15408



■ 経緯

- 欧米各国独自の情報技術セキュリティ評価基準では、相互認証は困難であったため、1999 年に国際標準として採択
- 日本では:2000年に、JISX5070 として採択

■ 範囲

- セキュリティ製品（ハードウェア・ソフトウェア）およびシステムの開発や製造、運用
- 製品やシステムが備えるべきセキュリティ機能に関する機能要件と、設計から製品化に至る過程でセキュリティ機能が実現されていることを確認する保証要件を網羅した要件集
- セキュリティについての基本概念の説明とともに、評価対象となる製品やシステムのセキュリティ基本仕様を記述するセキュリティターゲット (ST) や、ST のベースとなる文書であるプロテクションプロファイル (PP) についても規定

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 18

情報通信分野では、インターネットで世界中が相互接続されるので、世界標準が求められる。

ISO/IEC 15408 は、情報関連システムや情報関連製品に必要なセキュリティ要件が規定されており、その要件は、

- ・情報技術を用いた製品やシステムが備えるべきセキュリティ機能に関する機能要件
- ・設計から製品化に至る過程でセキュリティ機能が実現されていることを確認する保証要件

の 2 つである。

ISO/IEC 15408 の構成と評価保証レベル



■ 以下のパートから構成

- Part 1. 総則および一般モデル
- Part 2. セキュリティ機能要件
- Part 3. セキュリティ保証要件

■ Part 3 では評価保証レベル Evaluation Assurance Level (EAL) として、7 段階のレベルで保証要件が規定されている

- EAL 1: 機能テストの保証
- EAL 2: 構造テストの保証
- EAL 3: 系統的テストおよび確認の保証
- EAL 4: 系統的計画およびテスト、レビューの保証
- EAL 5: 準形式的設計とテストの保証
- EAL 6: 準形式的な設計の検証とテストの保証
- EAL 7: 形式的な設計の検証およびテストの保証

■ EAL 1～3 が一般向けで、EAL 4 が政府機関向け、EAL 5 ～ 7 が軍用・政府最高機密機関レベル

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 19

ISO/IEC 15408 は Part1, 2, 3 と 3 つあり、Part2 が機能要件、Part3 が保証要件となっている。

Part3 の保証要件については、評価保証レベル (EAL) があり、7 段階のレベルで保証要件が規定されている。

分類としては、EAL1 - 3 が一般向けで、EAL4 が政府機関向け、EAL5 -7 が軍用・政府最高機密機関レベル。

2005 年当時のICカード業界の一般的な保証レベルは EAL4 (系統的計画及びテスト、レビューの保証) ができていればいいという状況であった。

現在のICカード業界では、EAL5 と EAL6 の間くらいで準形式的な設計と検証を行なっていくというのが一般的となっている。

電子機器の機能安全に関する国際規格IEC 61508



- リスクの事前評価=「絶対安全は存在しない」
- リスクの軽減=許容範囲内におさめる
- IEC 61508:2000
 - Functional safety of electrical/electronic/programmable electronic safety-related systems
- JISCO508:2000
 - 「電気・電子・プログラマブル電子安全関連系の機能安全」
- 安全尺度 SIL (Safety Integrity Level)

SIL	信頼性	連続稼動設備における1時間あたりの故障発生率
1	90%以上	10^{-6} 以上 10^{-5} 以下
2	99%以上	10^{-7} 以上 10^{-6} 以下
3	99.9%以上	10^{-8} 以上 10^{-7} 以下
4	99.99%以上	10^{-9} 以上 10^{-8} 以下

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 20


別の規格として 2000 年に制定された IEC 61508 がある。

この規格は、システムに対する絶対的な安全はそもそも存在しないという前提から始まり、リスクを評価して、それを許容範囲に収めるようにするという考え方に基づいている。

これも、国際規格が制定されて、すぐに JIS になっている。

【さらに学習したい人に】

これは、翻訳JISという考え方で、JISに類似のものが存在せずJISとして新しく作成するには時間がかかり過ぎる場合、国際規格を日本語訳してJISとするものである。なお、JIS化した場合、最近新しく翻訳JIS化するものは、ISOの番号と同じ番号 にしているが、それ以前のものはISOとJISの番号は異なる。

IEC 61508 関連の規格			
規格種別	規格名称	対象	
EN規格	EN 61548	一般(計装)	
	EN 50126	鉄道	
IEC規格	IEC 6151	プロセス産業	
	IEC 61513	原子力	
	IEC 62061	産業機械	
	IEC 62304	医療	
	IEC 62800	モータ	
ISO規格	ISO 26262	自動車	
指針類	EEMUA	アラーム指針	
	DFT STAN	防衛(英国)	
	海軍規格(英国D	海軍(英国)	
	MISRA-SA	自動車(英国)	
	計量ソフトウェア指針	モータ	
	EMCガイドライン	IEE(英国)	

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 21

IEC 61508 は総称的な規格であり、色々な分野に対し、派生した国際標準がある。

例えば、自動車関連の ISO 26262 は、元々の IEC 61508 をベースとして自動車向けのシステムに合致するように変えたものである。

【さらに学習したい人に】

EN: 様々な分野の欧州規格 (European Standard, EN)を策定するため、1961年、欧州標準化委員会 (Comité Européen de Normalisation, CEN) が13カ国のメンバーで共同で創設。

IEC: International Electrotechnical Commissionの省略形で、日本語名称は国際電気標準会議。

1908年に創設された電気分野を専門に扱う国際機関。

ISO: International Organization for Standardizationの省略形で、日本語名称は国際標準化機構で、1947年に創設された電気分野を除くあらゆる分野において、国際的に通用させる規格や標準類を制定するための国際機関。

形式手法の推奨



表 3 IEC 61508 における形式手法の推奨例 (R:Recommended, HR:Highly Recommended)

Software safety requirements specification				
Technique/Measure	SIL1	SIL2	SIL3	SIL4
1 Computer-aided specification tools	R	R	HR	HR
2a Semi-formal methods	R	R	HR	HR
2b Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	—	R	R	HR
Software design and development: detailed design				
Technique/Measure	SIL1	SIL2	SIL3	SIL4
1a Structured methods including for example, JSD, MASCOT, SADT and Yourdon	HR	HR	HR	HR
1b Semi-formal methods	R	HR	HR	HR
1c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	—	R	R	HR
2 Computer-aided design tools	R	R	HR	HR
3 Defensive programming	—	R	HR	HR
4 Modular approach	HR	HR	HR	HR
5 Design and coding standards	R	HR	HR	HR
6 Structured programming	HR	HR	HR	HR
7 Use of trusted/verified software modules and components(if available)	R	HR	HR	HR

(出典:IEC 61508 Part1 Annex A より作成)

安全度水準 (SIL: Safety Integrity Level) のレベルが高くなればなるほど、各手法を使うことを強く推奨される。

HR = Highly Recommended (強く推奨)

R = Recommended (推奨)

VDM, Z, CSP 等の形式手法はレベルが高くなれば、強く推奨される。

5. 慣習的手法とその限界、および その解決に有用な形式手法の特性

ソフトウェアの機能性・信頼性の確認方法



- ソフトウェアのレビュー
- テスト
- 検証、証明

通常ソフトウェアの機能が妥当であるか、信頼性は十分かの確認には

- ・レビュー
- ・テスト
- ・検証、証明

といった手段がある。

ソフトウェアのレビュー※



■ 開発の各工程ごとに成果物を検査・評価すること

- ウォークスルー(walk-through)
 - 開発者が進行役となって実施
 - エラーの早期発見が目的
 - 修正の検討と確認はしない(開発者の責任)
 - 資料を事前配付
- インスペクション(inspection)
 - モデレータが進行役となって実施
 - エラーの早期発見と修正の検討が目的
 - 修正結果まで追跡確認(モデレータの責任)
 - ウォークスルーよりは形式的
 - 資料を事前配付
- ラウンドロビン(round-robin)
 - レビューメンバが持ち回りでレビュー責任者を務める

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 25

(※このスライドは管理者コースや初回学習時ではスキップも可能)

レビューは、開発の各工程ごとに成果物を検査・評価することである。

ウォークスルーとインスペクションの2種類があり、ウォークスルーよりもインスペクションは客観的になっている。

インスペクションは、第3者のモデレータがいて、色々問題点を指摘してもらったことに対し、どのように修正したかもトレースできるようになっていて、ウォークスルーよりも手順がはっきりしている。

レビューは、集中力が必要なため長時間の作業は難しく、専門的な知識や経験が必要である。

テスト※



- 「プログラムのテストは、バグの存在を示すには極めて有効であるが、バグが存在しないことを示すには絶望的に無力である」

[E. Dijkstra]

- 不具合の検出
- 信頼性に対する確信度
- 品質指標を用いたテスト項目の作成
 - 要求品質
 - 顧客に依存
 - 品質指標で危険の掘り下げ
 - 品質特性の分類
 - テスト項目
 - 漏れなく、顧客が認める評価項目

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 26

(※このスライドは管理者コースや初回学習時にはスキップも可能)

テストでは、計算機の上で実行させて、テストデータを入力し、テストの結果を見ることができるので、結果を客観的に評価できる。

仕様がはっきりしていれば、どのテストデータに対しテストがパスしたのか、もしくはエラーなのか、または、異常な振る舞いに対しては、異常があるということがわかるが、テストはきりが無い。

非同期処理に対する再現性も保障できない。

テストは 100% 正しいということを保証はできないが、信頼性に対する確信度を上げるための方法である。

ソフトウェア検証 (Software Verification) ※



- 「納得できる正当性の証明が、必要な確信のレベルに到達する唯一の方法であるように思われる」[E.Dijkstra]
 - 公理系に基づく証明
 - 証明の記述、チェック、理解、評価
 - 「プログラムが大規模・複雑になればなるほど、検証が有効・重要になる(何故なら、テストではいよいよ解決できない)」
- 形式手法の基礎
- 理論体系の構築
 - 対象=システム、性質
 - 理論と現実
- 証明することの副産物
 - 概念の明確化
 - 曖昧さのない議論と記述

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 27

(※このスライドは管理者コースや初回学習時ではスキップも可能)

前の頁でテストについて述べたのに続いて、Dijkstra は以下のように述べている。

ソフトウェア検証は、公理系、つまり論理体系、証明体系に基づいて証明する手法である。

証明を書けば、他人も理解でき、どの程度の妥当性、正当性があるかの検証もできる。

理論的な体系に基づいて、システムを記述し、確認あるいは証明をした性質を確定するだけでなく、証明しようとして、問題対象を深く考えることにより、ぼんやりしていたことが少しずつ明らかになるという点も重要である。

プログラム検証から形式手法への歴史※



- プログラムの正しさ(検証)=1960年代～
- 検証理論に関する研究の発展=1970年代～
- 形式仕様記述言語/方法論/ツールの開発
 - VDL(Vienna Description Language),VDM(Vienna Development Method)
 - Z Notation
 - Bファミリー(B Method, Event- B)
 - Larch
 - OBJ
 - HOL
 - モデル検査
 - etc.
- 適用事例蓄積と実用化

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 28

(※このスライドは管理者コースや初回学習時ではスキップも可能)

プログラムが正しいとは、どういうことか。

LISP の開発者 John McCarthy が 1962 年の論文で、数学的な証明について述べている。

1966 年前後からこうした研究が盛んに行なわれ、これが形式手法の基となっている。

VDM や Z Notation/B Method あるいは、システムを抽象化したモデル検査が代表例である。

プロセス代数に基づく OBJ も理論のしっかりした方法論である。

欧州では、基礎研究と実践とが続けられてきた。

「McCarthy:62」

John McCarthy: Checking mathematical proofs by computer, In Proceedings of Symposium on Recursive Function Theory, pp.219-227 (1961).

形式手法適用レベル



J. P .Bowen, et al. : Ten commandments of formal method,
IEEE Computer 28,No.4,pp.55-63(1995)

■ レベル0: 形式仕様記述

- 数学的な記法を用いて厳密な仕様を記述し、証明や分析までは行わずに、この記述を基にしてプログラムを開発する

■ レベル1: 形式的開発および検証

- プログラムの性質を証明したり、詳細化により仕様からプログラムを作成する

■ レベル2: 機械支援による証明

- 定理証明器や証明支援器を用いてプログラムの性質を証明する

どのレベルでもまず厳密な仕様の記述が必要であり、その記述を行うだけでも(レベル0でも)効果が期待できることが多い

システムの厳密な記述がなければレベル1もレベル2もない。

レベル 0 の段階でかなりのことが得られる。

あとは、どれだけの信頼性とどれだけのコストをかけるかのトレードオフである。

形式手法の効果



■ 直接的効果

- 各種記述物
 - 記述物(要求、仕様、設計、コード、テスト、バグ、etc.)
 - 証明結果、分析

■ 間接的効果

- 開発対象の認識と理解
- 開発プロセス改善
- コミュニケーション
- 開発者の自信
- 高品質ソフトウェアの効率的開発

■ 「形式手法」とは「書くこと」

- 書くために理解、認識、議論、分析、共有、etc.
- 書いてあること V.S. 書いていないこと
「決めること」

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 30

形式手法導入の効果には、

・直接的効果

仕様の厳密な記述、ソフトウェア記述の充実

・間接的効果

開発の対象の問題の認識や理解、開発過程の品質底上げ

がある。

直接的効果より、むしろ、間接的効果の方が大きく重要である場合もある。

形式手法で重要なのは、何を書き、何を書かないか、どう書くかを決めること。

書くために問題対象をきちんと認識し、理解し、何をどういう観点から議論し、何を分析し、何を保証するか、ということをきちんと意識してやらなければいけない。

また、書かれたものによって知識や知見、ノウハウを共有できる。

コミュニケーションへの効果例



フェリカネットワークスでの形式手法適用事例より

- 自然言語・形式仕様記述言語の違い
 - 「理解」と「明確化依頼」以外は似通っている
- 自然言語
 - まず書いてあることがよくわからない
 - …「明確化依頼」=「(書いてあることが)わかった」
 - でとどまってしまう
- 形式仕様記述言語
 - 書いてあることはわかる…中途半端で断片的な「理解」だったと自覚できるので、背景・根拠を含めより深く「理解したい」、「納得したい」につながる
- 日本語での議論は(精神的にも)つらい?
 - 記述から人格を消して「問題対私たち」とする

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 31

自然言語に関しては、何度も聞き返すということが度々起こることになり、それは聞く側も聞かれる側も気分のいいものではない。

形式仕様では、記述そのものは理解できるので、背景や理由が議論の対象となる。

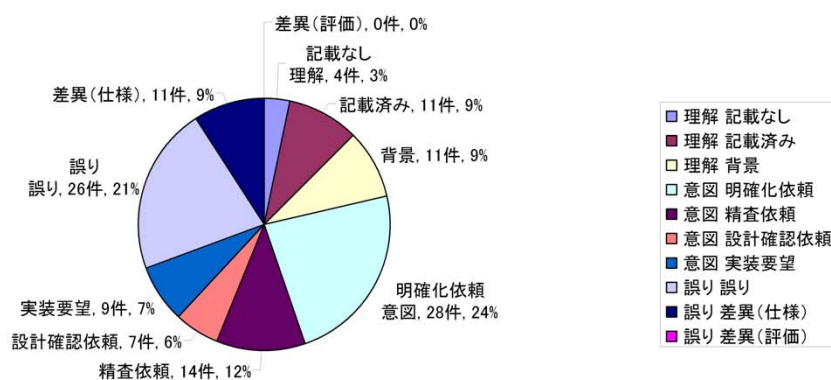
記述そのもの対メンバーという形へ議論の構造を持ていくと議論しやすい。

自然言語のコミュニケーション※



■ 自然言語によるマニュアル

→ 明確化依頼が多い



栗田太郎: 携帯電話組み込み用モバイル FeliCa IC チップ開発における形式仕様記述手法の適用,
情報処理情報処理Vol.49, No.5, pp.506-513, 2008年5月より引用

(※このスライドは管理者コースや初回学習時にはスキップも可能)

フェリカネットワークスの現場で使用している、プロジェクト内での問い合わせシステムにおいて、自然言語によるマニュアルに対して、プロジェクト内で様々な問い合わせがあり、それに対して割合を分析したものである。

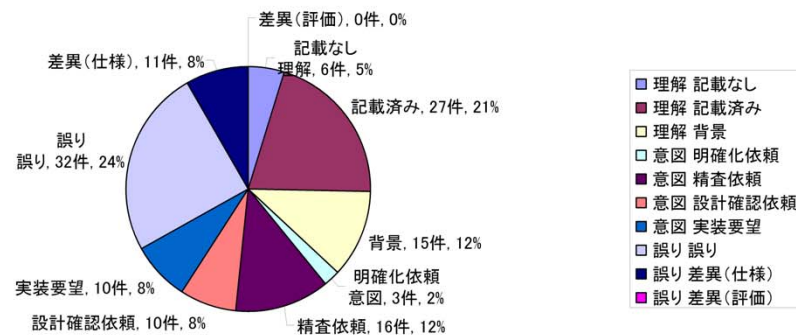
形式手法によるコミュニケーション※



■ 形式仕様記述言語による外部仕様書

→ 「理解」に関する質問が多い

→ 仕様策定背景・経緯はコメントに記述する



栗田太郎: 携帯電話組み込み用モバイル FeliCa IC チップ開発における形式仕様記述手法の適用.
情報処理情報処理Vol.49, No.5, pp.506-513, 2008年5月より引用

(※このスライドは管理者コースや初回学習時ではスキップも可能)

円グラフより、自然言語によるマニュアルでは「明確化依頼」が大きな割合を占めているが、形式仕様記述言語による仕様書では「理解」に関する質問が大きな割合を占めている。

(このことの意味は次のスライドに示す.)

自然言語による仕様策定の背景や経緯を、コメントにして記述することで理解を促すということも必要なのではないか。

6. 現場向きの形式手法VDMの概要

VDMとは？



■ 数学をベースとした仕様記述と検証のための手法

- 1960年代から1970年代にIBMウィーン研究所で開発
- 1996年に、VDM-SLが世界初のISO標準(ISO/IEC 13817)仕様記述言語になった

■ 特徴

- 厳密に定義された仕様記述言語 VDM-SL, VDM++, VDM-RT を持つ
- 制約条件(不変条件、事後条件、事前条件)の記述が可能
- ツールによる証明課題の生成
- 産業界の実用のために拡張された

VDMは、Vienna Development Methodの略。

1960年代から70年代にかけて研究されて、70年代後半ぐらいに言語として、徐々にまとまったもの。

1996年にVDM-SL:VDM Specification Languageが世界初のISO標準の仕様記述言語になった。

他に、オブジェクト指向拡張した VDM++ と、仕様記述をベースにアーキテクチャを決めるためのシミュレーションに使える実時間仕様拡張 VDM-RT がある。

これらは、一般のプログラミング言語と似ているが、

- ・[制約条件を書く](#)ことができる
- ・この条件式を証明すれば内部矛盾は無い、という[証明課題の生成](#)などの違いがある。

形式手法VDMの有用性



- 理論的にも経験的にも形式手法が有効
- VDMToolsで使用するVDMは、**現場寄り**の形式手法
 - 証明やモデル検査は、長期的にはやるべきだが、**すぐにはできない**
- VDM導入は、さほど難しくない
 - 3ヶ月程度の教育とコンサルティング
 - 既存の役立つ**ソフトウェア工学ツール**と協調して、より効果が出る
- モデル化は、以下が重要
 - モデル化の**範囲を決め**、仕様を**分割統治**
 - 名詞から型、述語から関数または操作
 - 陰仕様を作成してから、**静的検証**
 - 陽仕様を作成してから、**動的検証**
- VDMは構造化日本語仕様として使うことができる

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 36

形式仕様記述は構造化設計やオブジェクト指向設計に基づいた UML など既存のソフトウェア工学ツールによる**モデル化技術との親和性も高い**。

そして、どこまでが形式手法で、どこまでがオブジェクト指向で、どこまでが構造化で見つかったのかという分析は、できない。

知識や経験を積んで臨まなくてはならないが、モデル検査や証明と比べて、VDM は比較的短期に効果を実感できる。

モデル検査や証明は、現状では、システムの中でも**特に信頼性を求められる部分**だけに用いられている。

フェリカネットワークスの事例では、文法講習に1週間、適用範囲の設定や記述結果のレビューなど、コンサルティングを含めて3ヶ月程度の専門家による導入支援を行なった。

モデル化にあたっては、対象をはっきりと決め、対象の状態遷移とその条件のみを記述した陰仕様、具体的な状態遷移のアルゴリズムを記述した陽仕様と進めていくのが基本的な進め方である。

VDM は**日本語を識別子に使える**ので、構造化日本語仕様としても利用でき、証明などとの組み合わせも有効である。

構造化日本語仕様としてのVDM ※



VDMは構造化日本語仕様として使える。(慣習的日本語仕様は使えない。)

- 構文を考える時間が不要である
- 構文・型チェック、証明課題レビューで静的に検証できる
- 証明課題で生成される条件式から、見落としていた不変条件や事前条件が見つかる
- 組合せテストにより、動的に正当性検証ができる
- 回帰テストにより、動的に妥当性検査ができる
- VDMソース自体が、要求辞書ともなる
- 日本語仕様より、記述と検証の工数が少ない
 - 特に、仕様修正に強い
- 擬似コード形式の日本語仕様に近い形で、かなりの部分を記述できる

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 37

(※このスライドは管理者コースや初回学習時ではスキップも可能)

VDM は 日本語を識別子に使える ので、構造化日本語仕様としても利用でき、証明などとの組み合わせも有効である。

形式手法 VDM を構造化日本語仕様としてみた場合のまとめである。

- ・回帰テストのフレームワークが用意されており、動的に妥当性検査、正当性検証が可能
- ・回帰テストが可能で、仕様修正の影響範囲がつかみやすい
- ・VDMソース自体にそれぞれの名詞や動詞に対応する型、関数、手続きが正確に定義されており、要求についての用語の辞書として参照できる
- ・擬似コード形式の日本語仕様に近く、動かすのが容易

VDMと慣習的日本語仕様:構造化日本語仕様か? ※					
	構文を考える時間	構文チェック	関連チェック	実行テスト	証明問題生成
擬似コードによる仕様	かなりの時間が必要で記法も統一できない	レビューのみ	レビューのみ	具体的データを想定したコードインスペクションに相当するチェックのみ(通常行なわれない)	不可能
VDM仕様	言語マニュアルを参照すれば良いだけ	ツールでチェック	ツールの型チェック	ツールで実行	ツールで生成

SEC-FM1-01-2Copyright © 2012,2013 IPASoftware Engineering for Mo-No-Zu-Ku-RiIPA Software Engineering Center38

(※このスライドは管理者コースや初回学習時にはスキップも可能)

構造化日本語仕様(VDMによる仕様)と通常の日本語仕様(擬似コードによる仕様)を比較した表。

7. 自然言語や慣習的な非形式的記述 による問題例

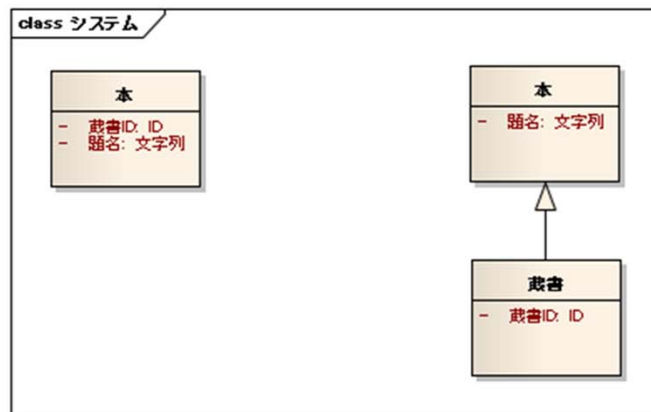
日本語仕様の曖昧さの例



クイズ

■ 以下の2つの「本」は同じか？

- 本を図書館の蔵書として追加する
- 題名で本を検索する



SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 40

「本を図書館の蔵書として追加する」という機能と「題名で本を検索する」という機能がある。

このふたつの“本”という単語は同じ意味か？

英語で言えば、本の copy と book で、本 1 冊 1 冊を対象にするか、概念的な出版物を対象にするかが異なる。

クラス図にすると、題名があるものと、蔵書IDと題名があるというものと、両方の本のモデルが考えられる。

この要求に対して、左のようなモデルを書いてしまうと、本というのは蔵書IDと題名があって、抽象的な本では無く、全部別々の本として、つまり題名も中身も同じでも別々の本として捉えている。

一方、右のモデルだと、概念的な題名をもった本と、蔵書IDをもった具体的な一冊一冊の本というものが、スーパークラスとサブクラスとなって存在する。インスタンスも両方にある。

左のモデルだと、例えば、著者とか題名を書き換えようとする、10冊あったら10冊分のデータを書き換えなければならない。

日本語の曖昧さの例 特急券予約での問題※



- 鉄道会社A 特急券予約サポートセンターとの問答
 - クレジットカードを変更したら、予約できないんですが?
 - **カード**を変更したら、特急券予約を新規に契約して下さい
 - クレジットカード変更前に予約した特急券に引き替えようとしたらできないのですが?
 - **カード**で引き替えできるようになったので、それで引換えて下さい
暗証番号は**カード**の物を使って下さい
 - カードの暗証番号って、何ですか?
 - クレジットカードの暗証番号です
- T駅での問答
 - クレジットカードで特急券に引換えできないんですが
 - 会員証でやってみて下さい
 - 会員証でもできないんですが
 - 変ですねー、こちら(駅員)でやってみましょう。駄目ですねー
 - ひょっとして、古いクレジットカードではどうですか?
 - あ、できました。はい、切符です

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 41

(※このスライドは管理者コースや初回学習時ではスキップも可能)

この例は、実際にあった事例をもとに、教材用に加工作成しています。

“特急券予約”は、おサイフケータイなどで新幹線の座席を予約ができる鉄道会社Bの特急券予約システムで、発車直前まで変更可能で、値段も少し安い。

日本語仕様の曖昧さの例として、どのような問題があったかを3カ月ぐらいの経過をまとめたものである。

ただし、この経験は数年前のものであり、サービスの名称・内容等については現在のもと同じわけではない。

T駅は新幹線の乗車駅であり、鉄道会社A および 鉄道会社B の窓口がある。

この一連の会話の中でも、カードがいくつもある中で、夫々の発言者がどのカードを指して言ってるのかわからない。

その後判明した特急券予約の用語※



■ カードの種類

- 予約会員証（変更前、変更後）、クレジットカード（変更前、変更後）
- その後、今回の件とは無関係と判明したカード
 - ICカード、予約カード

(※このスライドは管理者コースや初回学習時にはスキップも可能)

”カード”には、予約会員証、鉄道会社Bのクレジットカードがあり、しかもそれぞれ変更前、変更後という概念がある。

今回の件とは関係がありそうで無関係と判明したカードに、鉄道会社AのICカード と特急券予約ができる予約カードがある。

IC カードは、電子マネー機能はなく、チケットレスサービスでの予約ができるというカードである。

予約カードは、鉄道会社A発行の特急券予約のための専用クレジットカードである。

曖昧さを修正した特急券予約の問答※



■ 鉄道会社A 特急券予約サポートセンターとの問答

- クレジットカードを変更したら、予約できないんですが?
 - クレジットカードを変更したら、特急券予約を新規に契約して下さい
- クレジットカード変更前に予約した特急券に引き替えようとしたらできないんですが?
 - 予約会員証で引き替えできるようになったので、それで引換えて下さい。暗証番号はクレジットカードのものを使って下さい

■ T駅での問答

- 変更後のクレジットカードで特急券に引換えできないんですが
 - 予約会員証でやってみて下さい
- 予約会員証でもできないんですが
 - 変ですねー、こちら(駅員)でやってみましょう。駄目ですねー
- ひょっとして、変更前のクレジットカードではどうですか?
 - あ、できました。はい、切符です

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 43

(※このスライドは管理者コースや初回学習時にはスキップも可能)

カードを厳密に記述すると上のようになる。

最終的に、モデルを構築し、問題を解決した後に、最初の間答を書き直したもので、当初の日本語が如何に曖昧であったかが分かる。

特急券予約の問題は何だったのか? ※



■ 要求仕様モデルの考慮不足

- 鉄道会社Bのクレジットカードを変更すると、特急券予約会員として新規に契約せねばならず、予約会員証も新規に作成
 - 変更という概念が無く、ユーザに不便を強いている
- 個々の予約が、クレジットカードにリンク
 - クレジットカード変更に弱く、ユーザにも駅員にも分かりにくい
- 予約会員証からクレジットカードにリンク
 - クレジットカード変更に弱い、ユーザにも駅員にも分かりにくい

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 44

(※このスライドは管理者コースや初回学習時ではスキップも可能)

特急券予約の問題は要求仕様モデルの考慮不足である。

- ・クレジットカードが変更された時への対応という概念がない

クレジットカードを変更すると、変更前のクレジットカードへのリンクが切れるので、ユーザーは特急券予約システムを新規に契約しなければならず、お金を払わなければいけない。

- ・個々の予約がクレジットカードにリンクしている

クレジットカードを変更してしまうと、それ以前の予約が古いクレジットカードでしか処理できない。

- ・予約会員証からクレジットカードにリンクされている。

クレジットカードの変更に弱くなっている。

日本語仕様の曖昧さの例 応当日※



■ 信用取引の決済日(期日)を得る

弁済期限とは、信用建玉に対して当社がお客様に信用を供与する期限をいいます。弁済期限は、現在のところ6ヶ月のみを取扱っています

弁済期限が6ヶ月であるということは、信用建玉の建日(信用建玉が約定した日)の6ヶ月目応当日が信用期日となり、この日を超えて建玉を保有することは法律で禁じられています。信用期日が休日の場合には、直近の前営業日が信用期日となります

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 45

(※このスライドは管理者コースや初回学習時ではスキップも可能)

別の日本語仕様の曖昧さの例である。

本例で用いられる用語として以下があり、これは明確に理解されないと仕様書自体が曖昧になる。

- ー信用建玉 (たてぎょく)
- ー建日 (たてび)
- ー決済日
- ー弁済期限
- ー信用期日

証券業務や役所には“応当日”というのがある。

例えば、信用取引日の取引の決済日を6ヶ月目応当日である、という説明がある。

”応当日”を調べるはずが、“決済日”や“弁済期限”といった似た用語が出てくる。

弁済期限が6ヶ月であるということは、信用取引がスタートして、決済日までに決済しないといけない。

それが6ヶ月目応当日が信用期日となり、弁済期限と信用期日の関係がまたわからない。

これでプログラム作っても、確実に欠陥が出る。

日本語仕様の曖昧さの例 応当日※



- 弁済期限と決済日と信用期日との関係は？
- 応当日とは？
- 応当日が月末日を超えたらどうなるのか？
- 直近の前営業日が5ヶ月後になった場合はどうするのか？

SEC-FM1-01-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 46

(※このスライドは管理者コースや初回学習時ではスキップも可能)

証券業務に詳しい方に聞いたときの応答は次のとおり。

・弁済期限と決済日と信用期日との関係は、「これは同じものです」と即答してくれた。

高々半ページずつくらい2ヶ所の文章で、同じものを3つの異なる言葉で記載していると、当然混乱する。

・「応当日とは？」これは、ちょっと説明はあったが、詳細は答えられなかった。

・「応当日が月末日を超えたらどうなるのですか？」これは、「さかのぼります」と即答してくれた。

例えば、8月31日が今日だとして、6ヶ月後となると、2月29日までしかないのに、31日ではなくて、営業日で前にさかのぼっていく。

・「直近の前営業日が5ヶ月後になった場合はどうするのか？」。

6ヶ月後を応当日と言っているのに、前にさかのぼると5ヵ月後になってしまっている。

休日と営業日という話とか、どちらが優先されるのかは、先ほどの文章からは全く分からない。

結論から言うと、「5ヵ月後になります」と、3時間後くらいに返事がきました。

このように、証券業務に詳しい専門家の方でも即答できないほど、日本語仕様には確認が必要な曖昧さがつきまとう。

VDM では、高々40-50行で書けます。

テストケースでもそれはわかるので、いったん業務知識を得たら、あとは検索するだけなので、まず忘れない。

逆に言うと、そういった定義がどこにも書いていなかった。

サマリー

サマリー



品質の高いソフトウェアの効率よい開発へ向け、形式手法とその効果について概略レベルの知識の獲得し、旧来の開発の課題に対する形式手法の有用性を理解し、仕様の重要性和正しい仕様の記述に対する形式手法の有用性を理解した上で、導入を検討するために、主に以下を学習

- 形式手法の定義と期待できる成果
- ソフトウェア開発の課題と形式手法
- ソフトウェアの信頼性・安全性への期待の高まりと国際規格・標準での形式手法の推奨
- 慣習的手法とその限界、およびその解決に有用な形式手法の特性
- 現場向きの形式手法VDMの概要
- 自然言語や慣習的な非形式的記述による問題例

もっと詳しい形式手法の説明※



- 数理的な道具立て/方法に基づくシステムの開発方法
 - 記述、分析、設計、実現、検証、保守
- 30年以上の歴史
- 多種多様な理論、方法論、ツール例えば、J.Bowenのwebでは、100種類以上
[http://formalmethods.wikia.com/wiki/Formal_Methods_Wiki]

(※このスライドは管理者コースや初回学習時ではスキップも可能)

形式手法とは、一言で言うと、数理的な道具立／方法に基づくシステムの系統立てた開発方法である。

また、30 年以上の歴史をもち、開発プロセスのどこにでも有効である。

ただし、一つの道具で全部のプロセスをカバーできるわけではない。

適材適所に活用することが重要である。



実務家のための形式手法

厳密な仕様記述を志すための形式手法入門

なぜ形式手法か

独立行政法人情報処理推進機構
技術本部 ソフトウェア・エンジニアリング・センター
統合系システム・ソフトウェア信頼性基盤整備推進委員会
上流品質技術部会 人材育成WG(編)
2013年3月 第二版発行

記載されている個々の情報に関しての著作権及び商標はそれぞれの権利者に帰属するものです
なお、本書の内容は将来予告なしに変更することがあります