FeliCa Networks

【ET2014】Embedded Technology 2014
情報処理推進機構（IPA）ブースプレゼンテーション

【先進技術適用事例紹介】
システム開発への形式手法の適用による品質の確保
～文書の記述力とチームのコミュニケーション力を鍛える～

フェリカネットワークス株式会社 開発部 2 課

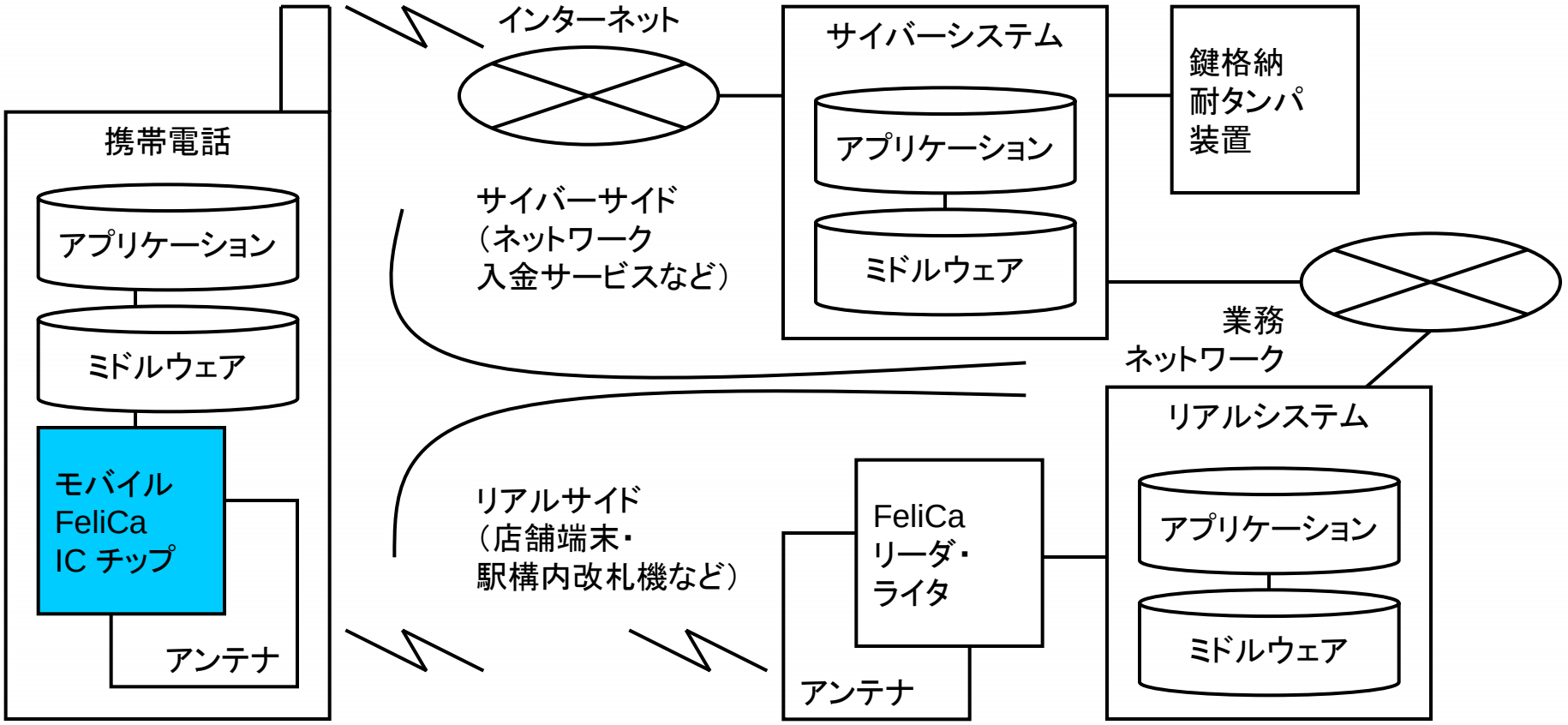
栗田 太郎

2014 年 11 月 21 日

概要

少人数ではないチームにおいて，設計や実装，テストを正確に行い品質を確保するために，また，運用や保守，派生開発において品質を維持し続けていくために，仕様や文書はどのような役割を果たすのでしょうか．「おサイフケータイ」のセキュリティ製品開発プロジェクトにおける，形式仕様記述手法の適用により品質を確保した事例とともに，文書の記述力やチームのコミュニケーション力を向上させるための取組みとその結果を紹介しながら，そのことについて考えていきます．

モバイル FeliCa システム



モバイル FeliCa IC チップファームウェアの開発

- 電子マネーや公共交通機関の乗車券，定期券などに応用される，社会インフラを構成するセキュリティソフトウェアである
- 日本中の携帯電話に組み込まれる

- 社会的責任が重い
- システムやサービスの根幹に関わる重大なトラブルが発生することで，フェリカネットワークス自身のみならず，一般ユーザの生活や，サービス事業者，携帯電話メーカ，移動体通信事業者のビジネスに影響が及ばないよう，品質の確保に当たらなければならない

私たちの現実

- 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らずストレスばかりがかかる

問題のひとつ

- ソフトウェアの開発現場では，曖昧な仕様に起因するトラブルが多い
- 下流工程の修正コストは，上流工程の 10^n 倍になる
- 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，手続的日本語プログラミングを行っている
- 業務を知らないコーダーが手続的に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大量に生産されている

正しい (?) 仕様の記述

- 正しい仕様を自然言語で書くのは困難である (形式仕様記述言語を用いても困難である)
- 曖昧さの排除が困難である
- ツールによるチェックができない
- 仕様を書きながら「擬似コード」を考えなければならないことが多くあるが，文法の検討に時間がかかる
- 図表のレイアウト検討や記述にも時間がかかる
- 変更管理が困難である
- 改善方法やその基準が分からない… (仕様書がメンテナンスできない，されない…)
- 仕様っぽいものが誰にでも書けてしまう…

- UML なども適用が困難である?
  - 仕様記述言語がなく (あるいはまだ整備中で)，自然言語で詳細を記述するため，結局自然言語仕様と同様の問題が発生する
  - ツールによるチェックがほとんどできない
  - 記法に曖昧さが多いため，形式手法より学習が困難である
  - 再利用のための仕掛けがない

プログラムは「数学系」

- **C/C++ 言語や Java 言語で書かれたプログラムは，検証が困難な数学系 (数理モデル) である**
- 誰かが，どこかで，数学系に変換しなければならないのだが，現場では業務を知らないコーダーが数学系に変換している
- 誰かが，どこかで，数学系に変換しなければならないのであれば，最初から数学系で書いてもいいだろう

  「いつか最後に世界を見ようと思ってたんですが，それなら最初でもいいだろう」(平田オリザ)

- 書き手と読み手の双方にとって客観性が必要
- 第三者評価ができなければならない

ISO/IEC 15408 第三者セキュリティ CC 評価・認証

- 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，運用などに関する国際標準
- 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を規定
- 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
- 情報セキュリティ評価基準として 1999 年に採択

= ITSEC（Information Technology Security Evaluation Criteria）

= CC（Common Criteria）

ISO/IEC 15408 のパートとクラス

- 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（EAL（Evaluation Assurance Level））として，実装の確かさの評価方法について，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: 一般民生向け
  EAL 4: 政府機関向け
  EAL 5～7: 軍用レベル，政府最高機密機関レベル向け

形式手法とは

- 形式手法とは，システム開発，特にソフトウェア開発の手法で，**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多面的な検証が可能となる上，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

形式手法の一部の分類

モデル規範: VDM･Z 記法･B メソッド等と，ツールを利用

- 集合論や命題論理，述語論理が基礎となる
- 不変条件，事前条件，事後条件を記述する
- 情報の構造や，ある状態から他の状態への変化をモデル化する
- 専用言語の利用によるコンパクトな仕様記述，曖昧さの除去，仕様の「実行」，「テスト」，「回帰テスト」，定理証明等の可能性が広がる

モデル検査: PROMELA，LTS 等の言語と SPIN (PROMELA), LTSA (LTS) 等のツールを利用

- 振る舞い仕様を，状態遷移モデルと，モデルが満たす条件として記述することで，全状態空間を生成し，自動検査する
- 従来の「テスト」では見つからない潜在的な障害を発見できる

VDM++ ＋ VDMTools

- **VDM の選定理由**
  - 仕様策定段階から動作可能
  - モデル化から動作確認まで広い範囲での適用が可能である

- **VDM の特徴**
  - VDM++ = VDM-SL (ISO で標準化) + OO
  - VDMTools
    - 仕様の構文チェック
    - 仕様の型チェック
    - 証明課題の生成
    - 実行可能仕様の逐次実行とデバッグ支援
    - 実行可能仕様のコードカバレッジ計測
    - 実行可能仕様から C++ 言語や Java 言語への変換
    - Java 言語から VDM++ 言語への変換
    - 各種 CASE ツールとの連動
    - 仕様の清書支援

仕様の構造 {P} A {Q} (Floyd-Hoare Triple)

- **P: 事前条件**
  - 処理 A の実行前に満足すべき条件
- **Q: 事後条件**
  - 処理 A の実行後に満足すべき条件
- **正しさの基準として事前条件・事後条件**
  - 処理 A のみから正しさを判断することはできない

```
public 最大値を取り出す: set of nat -> nat
最大値を取り出す(正数集合) ==
  Impl_最大値取得(正数集合)
pre
  正数集合 <> {}  -- 事前条件
post
  RESUTL in set 正数集合 and
  forall n in set 正数集合 & n <= RESULT;  -- 事後条件
```

形式手法の適用レベル

【レベル 0】

形式的に仕様を記述する．数理論理学に基づく記法を用いて厳密に仕様を記述する．この記述を元にプログラムを開発する．

【レベル 1】

形式的に開発と検証を行う．段階的詳細化により仕様からプログラムを作成したり，仕様やプログラムの性質を証明したりする．

【レベル 2】

機械支援による証明を行う．定理証明器や証明支援器を用いて，仕様やプログラムの性質を証明する．

- 私たちの適用はレベル 0

チームへの形式手法の導入

- 当初，形式手法や形式仕様記述手法に通じているメンバはいなかった
- 専門家による 5 日間の教育を受けた
- VDM++/VDMTools は，ソフトウェア開発技術者にとって馴染みやすいものである

目的

- **厳密な仕様の策定と記述の継続**
  - 第三者テストが可能になる
  - 第三者評価が可能になる
  - 運用・保守が可能になる
  - 再利用性が向上する

- **仕様を活用したイテレーティブな開発プロセスの確立**
- **仕様の多方面からの精査**
- **仕様のテスト**
  - 仕様のテスト
  - 「テスト仕様」のテスト

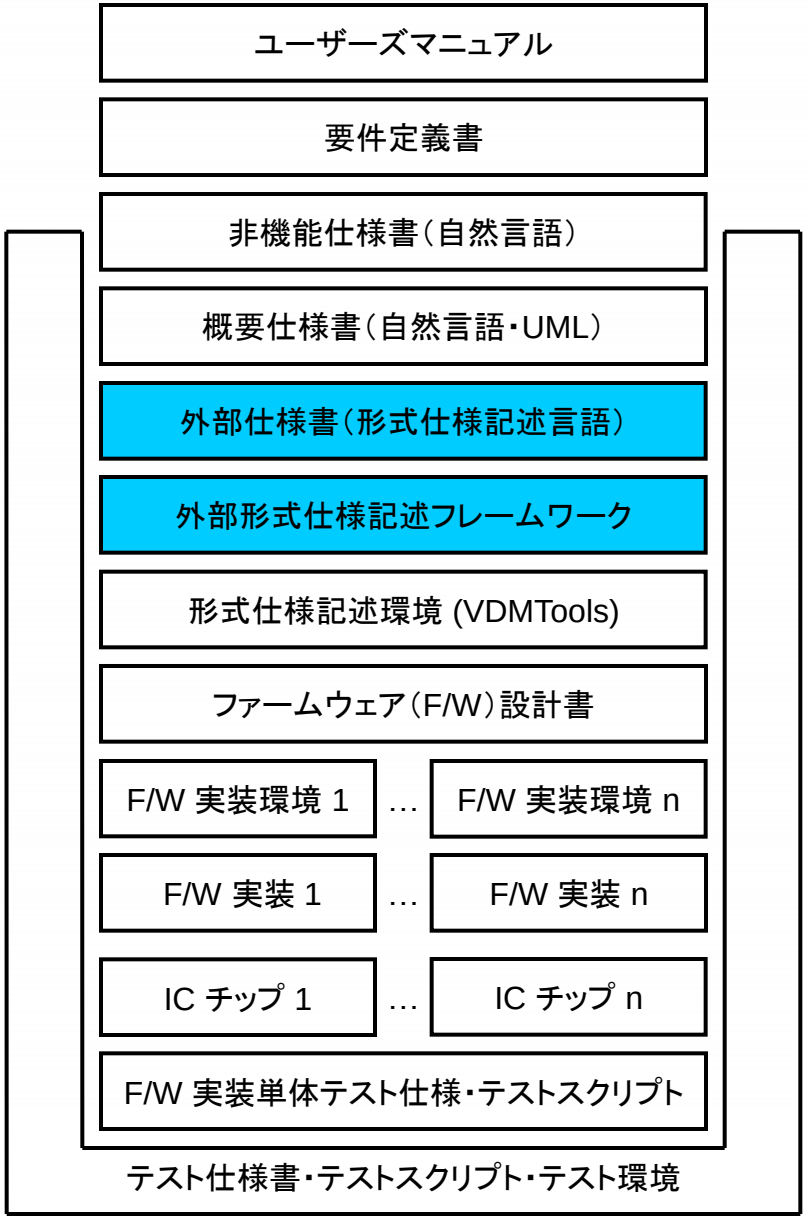
- **コミュニケーションの促進**

適用範囲 – モバイル FeliCa OS

- **実行可能な外部仕様を形式的に記述，テスト (形式手法のレベル 0)**

- **仕様記述フレームワーク上に仕様を展開**
  - VDM++ 汎用ライブラリ
  - ドメイン固有の用語や型や述語の定義
  - ドメイン固有のデータ構造
  - 仕様記述フレームワーク
  - 仕様記述テンプレート
  - テスティングフレームワーク

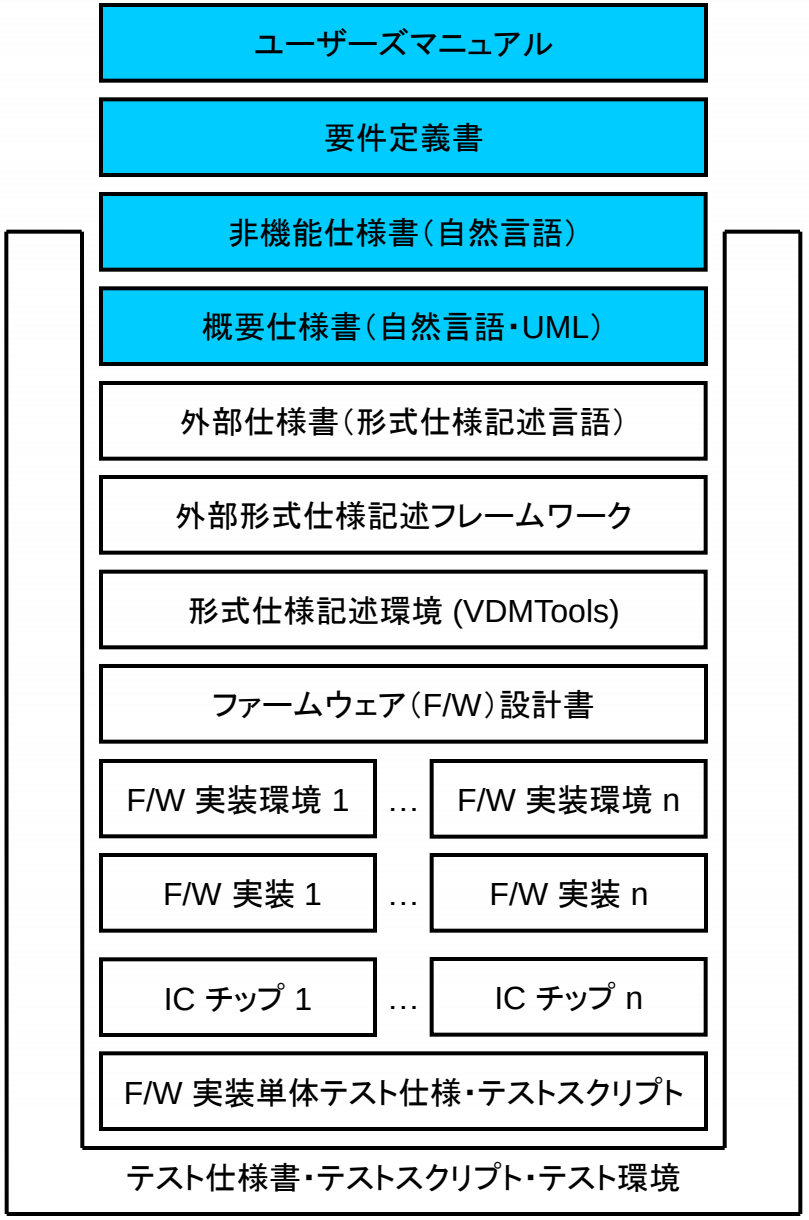
- **フレームワークと仕様の境界，すなわち動作する仕様と仕様の明確化が困難であった**

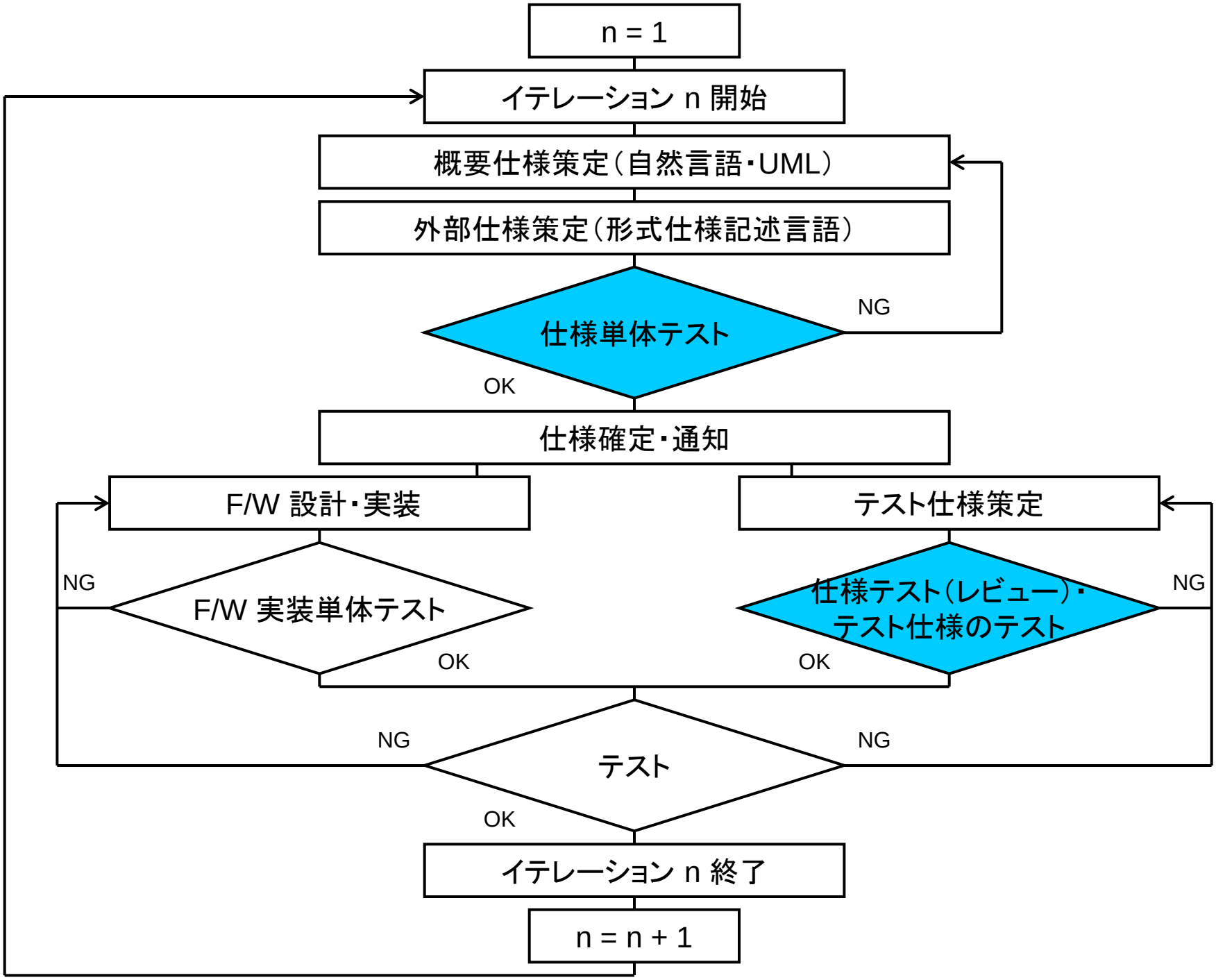


適用範囲 – モバイル FeliCa OS

- **以下は自然言語で記述した**
  - ユーザーズマニュアル
  - 要件定義書
  - 非機能仕様書
  - 概要仕様書

- **形式言語と自然言語それぞれによる記述の整合性確保，自然言語で書いた文書の品質確保に課題が残る**




n = 1
イテレーション n 開始
概要仕様策定（自然言語・UML）
外部仕様策定（形式仕様記述言語）
仕様単体テスト
NG
OK
仕様確定・通知
F/W 設計・実装
テスト仕様策定
NG
F/W 実装単体テスト
OK
仕様テスト（レビュー）・
テスト仕様のテスト
NG
OK
NG
テスト
NG
OK
イテレーション n 終了
n = n + 1

形式仕様記述手法の適用成果

- 仕様記述言語 VDM++ と仕様開発環境 VDMTools を用いて，外部機能仕様を，動作する形式仕様として表した
- 作成した仕様書は以下の通り:
  - 自然言語による 383 ページのプロトコル仕様書
  - 形式仕様記述言語 VDM++ による 677 ページの外部仕様書

- 仕様書のコード量はテストコードを含め，約 10 万行
- C++ 言語によるファームウェアのソースコードは一種類のチップにつき約 11 万行

- 開発時における仕様関連のトラブルは少なかった
- IC チップの出荷後，ファームウェアの品質に関連するトラブルはない

ファームウェア実装フェーズにおける不具合原因の割合

表 2　不具合原因の割合

不具合原因	割合
仕様記述もれ	0.2%
仕様記述誤り	0%
仕様不明確	1.8%
仕様見落とし	5.6%
仕様理解不足	10.7%
仕様確認不足	0%
仕様変更通知不徹底	0.2%
その他仕様関連外	81.5%

→ 仕様は書けている

ファームウェア実装フェーズにおける不具合原因の割合

表 2　不具合原因の割合

不具合原因	割合
仕様記述もれ	0.2%
仕様記述誤り	0%
仕様不明確	1.8%
仕様見落とし	5.6%
仕様理解不足	10.7%
仕様確認不足	0%
仕様変更通知不徹底	0.2%
その他仕様関連外	81.5%

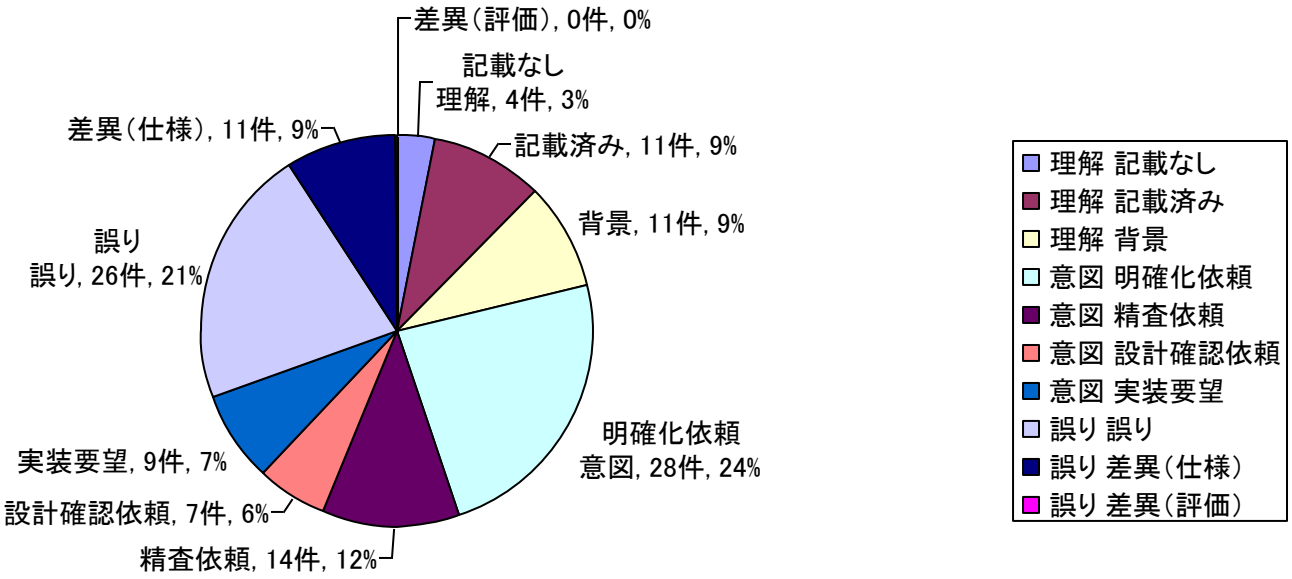
→「読ませる仕様」の記述が課題である

→ DSL (仕様記述フレームワーク) の検討と構築が重要

結果・考察 – 仕様とコミュニケーション (1/3)

- 自然言語によるマニュアル

→ 明確化依頼が多い

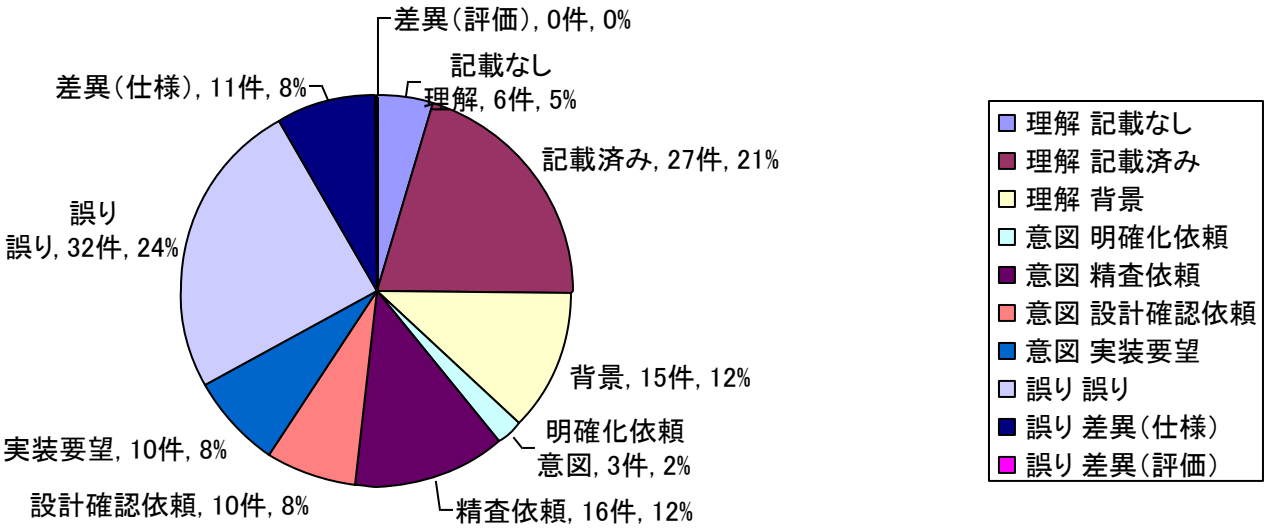


結果・考察 – 仕様とコミュニケーション (2/3)

- 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



結果・考察 – 仕様とコミュニケーション (3/3)

- 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

  → まずよくわからない…「明確化依頼」=「わかった」でとどまってしまう

- 形式仕様記述言語

  → 中途半端に「理解」できない．背景までつきつめて「理解」したい，納得したい

- 日本語での議論はつらい?

  → 記述から人格を消して「問題対私たち」とする

まとめ

- 上流工程，とくに仕様策定工程における成果物の品質向上に効果
- 仕様策定・実装・テストのイテレーションを多数回しても，ノイズが増幅されず，仕様を洗練することができる
- 曖昧さを排した記述により，第三者による客観的な確認を行うことができる
- プロジェクト内のコミュニケーションの活性化に寄与する
- 他の工夫との組み合わせにおいて効果を発揮する
- 開発ドメインに応じて，他の手法を組み合わせる

形式手法の活用の効果

- **直接的な効果**
  - 記述と検証 → 書くだけでも効果があるが，ゆくゆくは証明や段階的詳細化へ?
  - 品質の確保

  → チームによる記述や検算の可能性を開く

- **間接的な効果**
  - ドメインに対する認識・理解
  - コミュニケーションの活性化
  - 技能の向上 (ドメインエンジニアリング，エンジニアリング，コミュニケーション)

  → ストレスの軽減や生活の改善につながる