

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

事例: 成功事例

事例: 成功事例(このモジュールについて)



形式手法導入の成功事例(フェリカネットワークス(株)の事例)から、形式手法導入の検討や計画立案の際に有用な知見を得るためのモジュール

■ 事前知識・経験

- 形式手法の導入を検討、計画している
- 形式手法の有用性についての基礎知識
- 形式手法導入のガイダンス

■ 学習目標

- 過去の成功事例から、自らの導入に際して有用な知見を得る

■ 主な学習項目

- 成功と言われている事例では
 - どのようにプロジェクトに形式手法を導入したか
 - どのような成果を得たか

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

2

講師の方へ:

このモジュールでは、フェリカネットワークスの事例を取り上げる。

1. どのような目的で形式手法を採用したか
2. どのように形式手法を利用したか
3. 関係する人間への影響はどうであったか

といった点を意識させてください。

対応して、

1. 問題点はなんであったか、どのように解決されたか
直接効果と間接効果はなんであったか
2. 自分たちのプロジェクトに応用できる場所はどこか
3. コミュニケーションはどう変わったか

といった点に注意させてください。

本事例に関連して公表された論文も多数ございますので適宜ご参照・ご紹介ください。

【文献】

- 栗田太郎「仕様の記述力を鍛える モバイルFeliCa 開発における形式仕様記述手法の導入事例」日経エレクトロニクス 2007年2月12日号, 133—152.
 - 仕様記述の実際と、効果、評価について
- 栗田太郎「携帯電話組込み用モバイル 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 Vol.49 No.5(May 2008), 506—513.
 - 仕様の記述手順について
- 栗田太郎「モバイル FeliCa のソフトウェア開発における品質確保のための構造と実践 抽象度の制御やコミュニケーションの活性化に向けて」情報処理学会デジタルプラクティス Vol. 1, No. 3 (July 2010), 148—157.
 - モデル検査の適用、テストでの形式仕様の利用、コミュニケーションについて

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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

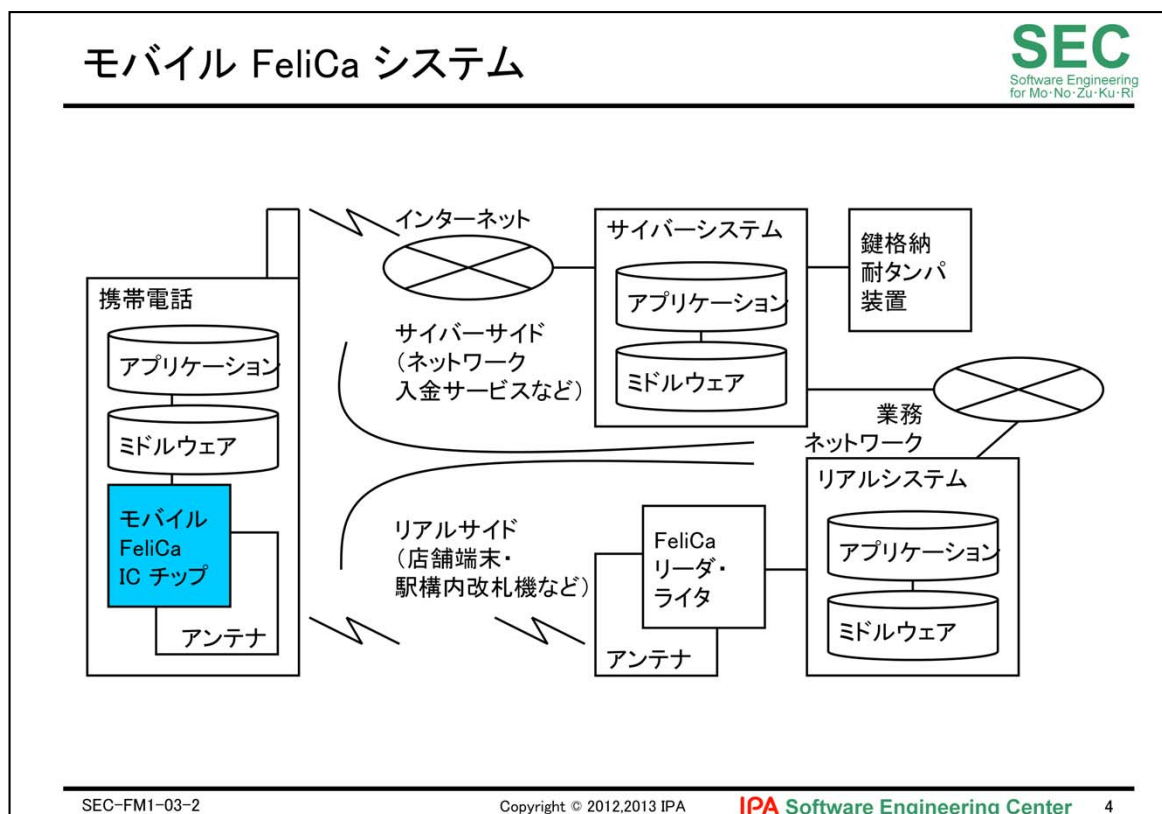
3

モバイル FeliCa は携帯電話への搭載を目的とした FeliCa システムの総称である。

チップの載った携帯電話を全て回収するのは、事実上不可能なので、出荷前に完全な検証が必要であった。

決済機能やプライバシーに関わる情報など、秘匿性の高い情報を扱うので、ユーザ企業の信用、エンドユーザへの影響など責任が重い。

また、技術的な保障は当然ながら、信頼性の根拠を説明できなければならない。



モバイル FeliCa は端末とリーダー・ライターだけでやりとりするシステムではない。

Web サイト、各店舗、自動販売機などがネットワークで結合されたバックエンドをもつ、大きなシステムである。

プロジェクトの概要



【ゴール】

フェリカネットワークス発の次世代モバイル FeliCa サービスの実現

【具体的な目標】

モバイル FeliCa IC チップファームウェアおよび周辺ツールの開発

【期間】

約 3 年 3 ヶ月

【メンバ数】

約 55 名

【平均年齢】

約 30 歳

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

5

形式手法を適用した対象は、ソフトウェアに関連した部分である。(プロジェクト全体に形式手法を適用したわけではない。)

3年3か月の内、最初の半年以外プロジェクト終了まで形式仕様を扱っている(記述の更新が続いた)。ただし、書き始めから一年後には安定し、その後の変更は殆どない。

プロジェクト開始時、形式手法の知識・経験を持つメンバーは無し。

開発に関わるほぼ全員が形式仕様を読む必要があるが、全員が書ける必要はない(7ページ参照)。

ソフトウェア開発における課題への対応



- 【要件獲得・仕様開発】**曖昧な仕様**に起因するトラブル
上流工程の不具合を下流工程で修正するコスト
→ 仕様記述言語を用いた**厳密な仕様の記述**に挑戦
- 【設計・実装】**設計が不明確であること**に起因するトラブル
設計不良によるトラブル
→ 状態遷移モデルの**網羅的なモデル検査**に挑戦
- 【テスト】**テスト項目の増大。項目の抜け漏れ**によるトラブル
人に依存したテスト
→ 効率良く品質を確保するための**組み合わせテスト**技法の活用
- 【プロジェクトマネジメント】**ステークホルダとの調整不足**に起因するトラブル
プロジェクトが**コントロールされていないこと**によるトラブル
- 【チーム運営】**コミュニケーション不足、スキル不足**に起因するトラブル
- 【第三者評価】**客観的な評価の欠如**
セキュリティ実装の妥当性証明の困難性

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

6

どのような課題があり、どのように解決しようとしたか。

[要件獲得・仕様開発]

パワーポイントを使って開発するというのが横行しており、その結果の不具合が非常に多くて困っていた。

それを下流工程で修正するコストが非常に大きなものとなっている。

[設計・実装]

設計が不明確であること・設計不良に起因するトラブルが多かった。

特に並行処理でデッドロックしないという要件のトラブルが多かった。

⇒ モデル検査で問題個所を確認。改修後の再検査で問題が起きないことを確かめた。

[テスト]

特に仕様がない(明確ではない)部分のテスト項目が人に依存しているので、不備があった場合、テストした人のせいになりがちである。

[チーム運営]

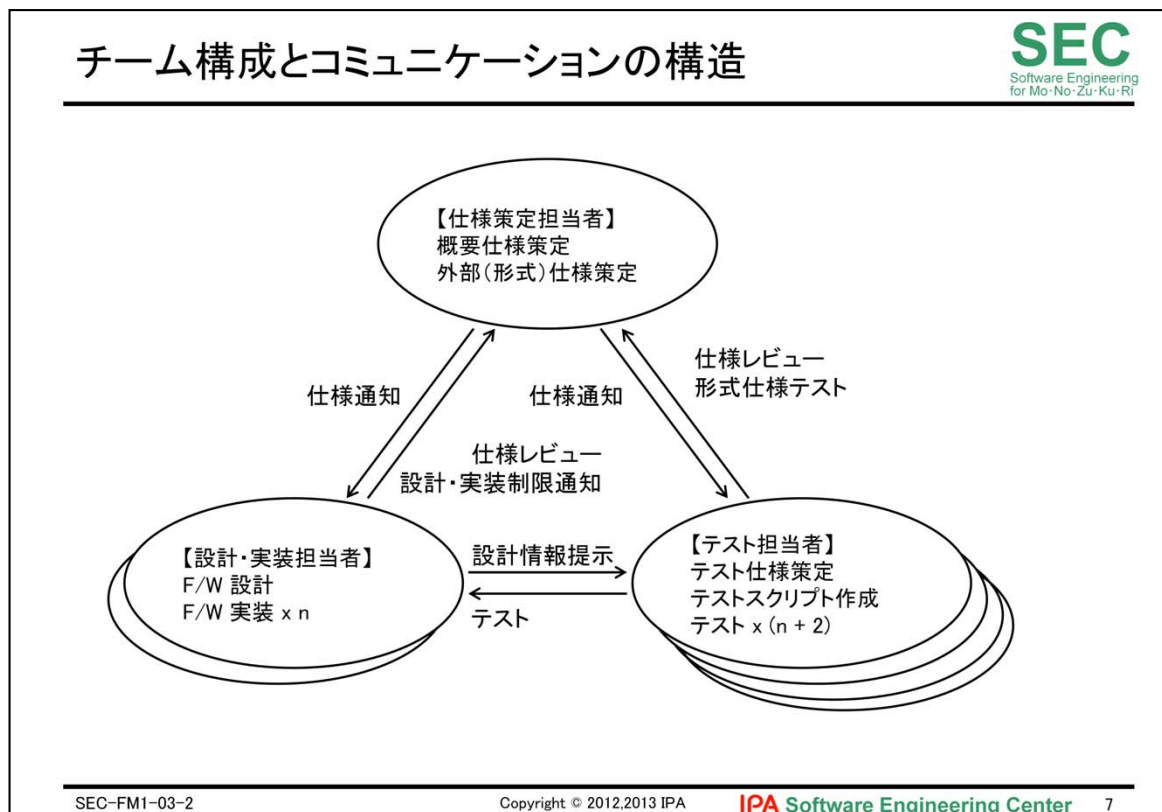
コミュニケーション不足やスキル不足による運営がうまくいっていないという問題もある。

→ チームビルディングや合宿研修を半年に一度と定期的に行なっている。

[第三者評価]

開発者が安全宣言をしても世の中には受け入れられない。

→ 第三者評価については「ISO 15408」のセキュリティの認証の取得をしている。



[チーム構成と担当]

「仕様策定担当者」「設計・実装担当者」「テスト担当者」に最初から完全に分かれている形(各チーム同人数が所属)。

「仕様策定担当者」は自然言語を用いた概要仕様の策定や外部形式仕様の策定を同時におこなった。それを「設計・実装担当者」、「テスト担当者」に渡していく。

「設計・実装担当者」は受け取った仕様を基に、設計と3種($n = 3$)のICカード用セキュリティLSIプラットフォームに対してプログラムを書くための設計を行う。

「テスト担当者」は受け取った仕様を基にテスト仕様の策定を行う。

テスト仕様の策定をおこなった後に、Rubyによるテストスクリプトの作成を行った。

従来の開発と大きく異なり、外部形式仕様が実行可能なモデルとなっているので、テストスクリプトの動作を事前に確認することができる。

テスト担当者は本番のテストを実施する前に、テストスクリプトと実行可能な外部仕様とをつないで、作ったスクリプトが当初の仕様通りか否かを確認する。

そして、そのスクリプトが仕様通りのものであるという自信を深めた後に、設計・実装担当者が作ってきた3つの実装に対して、実装が仕様通りであるか否かを確認する。

(実装 $\times n$ と外部形式仕様と、テストスクリプト自身をテストするため、合計で $(n+2)$ 回テストを行う。)

また、VDMTools で計測できるカバレージを目安として利用した。

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
 - 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
 - 仕様っぽいものが誰にでも書けてしまう…
- UML など適用が困難である？
 - 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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

8

形式仕様言語を使用すれば、自然言語を使用する場合と比べて、


- ・曖昧さの除去がよりできるようになる
- ・ツールを用いたチェックができる
- ・プログラムと同じような書き方ができるようになるので仕様を書きながら、擬似コードなどに関して考えることが少なくなる
- ・プログラムと同じように書けるため構成管理が可能になるので変更管理が可能となる
- ・仕様をかけない人が「仕様っぽいもの」を書くことを排除することができる

だからといって、困難なものが簡単になるとは言い切ることはできない。

仕様を考え検討するという事自体が難しいので、仕様記述言語を使用しても、なお難しい物は存在する。

表現手段としては、自然言語を使用するよりは解決される問題が多くなる。

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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

9

VDM-SL (Vienna Development Method – Specification Language) という ISO で標準化されている仕様記述言語にオブジェクト指向 (Object-Oriented) の機能を追加した VDM++ を使用している。

VDMTools というのは VDM-SL や VDM++ で仕様を書いたり、動作させたりするツール。

- ・仕様の構文チェック
- ・仕様の型チェック
- ・仕様課題の生成
- ・実行可能な仕様を書いていれば逐次実行やデバッグの支援、コードカバレッジの計測、プログラミング言語への変換
- ・Rational Rose や Enterprise Architecture といった UML ツールとの連携
- ・ワードや TeX への清書支援

といった機能がある。

ただし、これらの機能全てを利用したわけではない。

仕様記述の目的



- 厳密な仕様の策定と記述
 - 第三者テストが可能になる
 - 運用・保守が可能になる
 - 再利用性が向上する
 - グローバル展開が可能になる
- 仕様を活用した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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 10

仕様記述の目的:

・第三者(テストチームのメンバー)が仕様だけを見てプログラムにおいて第三者テストをできるようにしたい(メンバーの三分之一がテストチーム)。

ここでは仕様だけを入力として、テストチームが仕様に基づいたテストができるようにするというのを目指す。

・運用や保守が可能となるようにする。

さらに再利用性を高めるということも目指す。

つまり、プロジェクトにおける「バイブル」(全ての関係者が、そこから必要十分な情報を引き出せる)の策定。

付随して

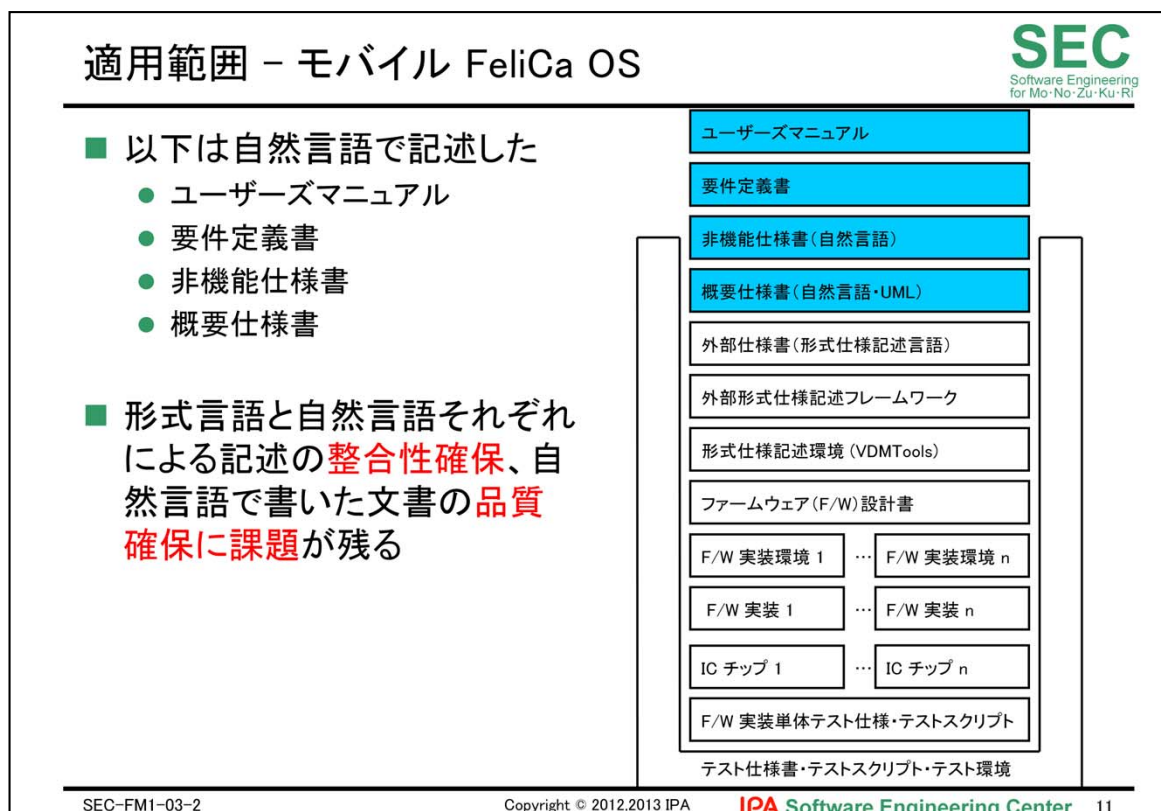
・まずは仕様を活用した仕様の開発プロセスを確立したい

・仕様の多方面からの精査していきたい

・仕様のテストを行いたい

・コミュニケーションの促進も図りたい

といったものがある。



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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

12

作成した仕様書は、

- ・自然言語で書かれた383ページのプロトコル仕様書
- ・形式仕様記述言語VDM++による外部仕様書

である。

仕様書のコード量はテストコードを含めて約10万行であり、その内の6万行がテストコードであった。

そしてその10万行のうちのコード部分である4万行は、C++で記述されたソースコードでは3種類あるチップのうち1種類で約11万行というようになっている。

もちろん全てあわせて33万行あるというわけではなく、11万行のうちの大体8万行くらいは重複したものであった。

結果として開発時における仕様関連のトラブルは少なかった。

ICチップの出荷後にファームウェアの品質に関するトラブルはない。

(2010 年 9 月末 FeliCa IC チップが 4 億 8,300 個出荷。

2010 年 7 月末「おサイフケータイ」用チップが 1 億 5,700 万出荷。

2012 年 3 月末「おサイフケータイ」用チップが 1 億 9,300 万出荷。)

仕様開発フェーズにおけるデバッグ



表 1 仕様開発フェーズで修正した誤りの件数

誤りの発見工程	件数
仕様の記述	162
仕様の実行と単体テスト	116
仕様のレビュー	93
設計実装担当者・テスト担当者とのコミュニケーション	69
合計	440

「デバッグ密度」= $440 / 40,000 =$ 約 11 エラー / 千行

→ 形式手法は、開発の初期段階における誤りの発見に有効である

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

13

仕様開発段階でどのような誤りが発見されたかというのをまとめた。

どの範囲から公式に管理していくかというのは難しい。

この資料にあるのは公式に誤りとして管理しているものである。

そして、それらの誤りは

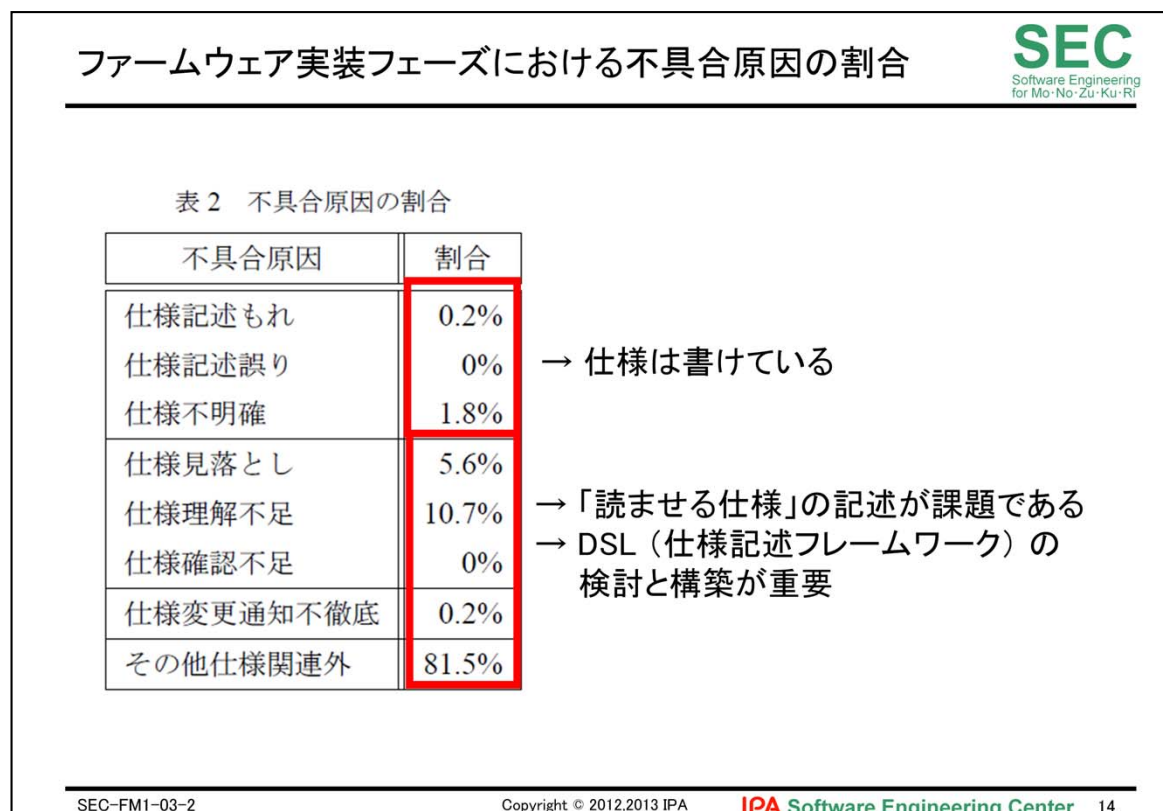
- ・仕様の記述によって162件
- ・仕様の実行と単体テストによって116件
- ・仕様のレビューによって93件
- ・設計実装担当者・テスト担当者とのコミュニケーションによって69件

以上の工程により440件の不具合を修正することができた。

デバッグの密度としては、大体1000行あたりに11エラーを修正していることになる。

仕様の記述や単体テストに関しては、ツールを使って細かく記述することによって発見することが多かった。従来、従来の方式(自然言語を用いてのレビューやコミュニケーション)でこの440件の不具合を発見することは難しかったかもしれない。

もちろん、細かく書かれているからこそ、レビューやコミュニケーションで不具合が発覚するということも言える。



ファームウェア実装における不具合原因の割合について、実装フェーズでデバッグして発見されたものの原因の割合の中で、仕様に関係するものを抜き出して書いたものである。

仕様の記述漏れ、記述文誤り、不明確なものに関しては、0.2%, 0%, 1.8% と合計で2%程度となっている。

今まで経験してきた中では、仕様に関する不具合が全体の2%程度しか無いというのは非常に画期的である。

開発の各段階の後工程でそれより前の工程で発生したバグとどの程度の因果関係があるかを種々のバグ分析手法で調べると、仕様段階に原因があるとされるものは大凡30～70%と言われており、仕様は書けているといえる。

一方で、細かく仕様を書いているが、仕様を見落としていたとか、書いているがよく理解できていなかった、そういった問題があつて、課題になってきている。

単に書くだけではなく、読ませる仕様記述が課題となっている。

仕様変更通知不徹底という問題も従来よりも少なくなっている。

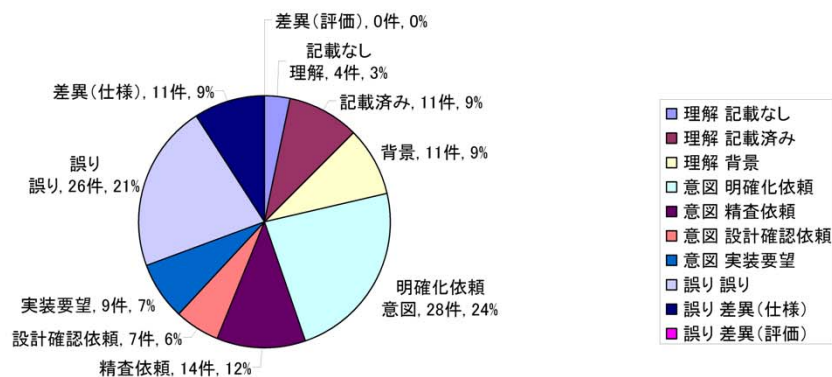
プログラムと同じように構成管理ツールを使って、プログラムの変更を管理できるようになっている。

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

現場で使用しているプロジェクト内問い合わせシステムの履歴を整理した結果、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示したものである。

円グラフを見比べるとわかることだが、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。

自然言語によるマニュアルでは、何度も聞き返しが発生し、それによってコミュニケーション上で問題が発生するという問題もあった。

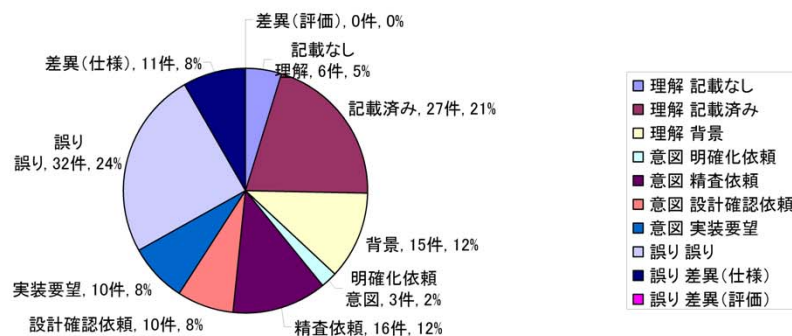
形式手法によるコミュニケーション



■ 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

それに対して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多い。

そこには明確な答えがあるため無機質で奥行きのない回答になりがちという問題がある。

そこで、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と思われる。

コミュニケーションのまとめ



- 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

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center

17

自然言語に関していうと、

「よくわからない」

「明確化依頼を出す事が多い」

「答えを聞いたけどなんとなくわかったような、わからないような…」

といったような流れとなってしまう、何度も聞き返すということが度々起こることになるが、それは聞く側からしても聞かれる側からしても気分のいいものではない。

形式仕様記述言語に関して言うと

「中身はわかるからこそ、あるものを全てきちんと理解したい」

「中途半端に理解出来ない」

「なぜこうなったのかと背景まで突き詰めて理解したい」

という考えに至る。

日本語だけで議論していてもなかなか続かないので、誰かの責任にせず、プロジェクト対メンバーという形の議論の構造を持つていくことが必要なのではないかと思われる。

結果・考察 – 仕様のテストと実装のテスト



- 実装だけに有益なのではない
- 厳密な仕様はテストにも活用することができる
- 仕様 = テスト仕様・テスト環境 = 実装 が確認できる

- テスト担当者は緊張するとともに安心もできる
- 自然言語による仕様の場合、テスト担当者はよりどころになるものがない

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 18

厳密な仕様があるということは、実装だけに有益なのではなく、テストにも活用することができる。

また「仕様＝テスト仕様・テスト環境＝実装」がつながりがあるということが確認でき、この3つをそれぞれメンテナンスしていくことができる。

また、修正した時に回帰的な確認ができるという利点もある。

曖昧な仕様が無いためにテストができないということがなくなるため緊張することになるが、その分検証できるという事は安心にもつながっていく。

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
- 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**

上流工程、特に仕様策定工程における成果物の品質向上に形式手法は効果を示している。

形式手法を取り入れたことで、仕様策定・実装・イテレーションを多数回まわしても、ノイズが増幅されることなく仕様を洗練することができたのではないかとと思われる。

客観的な評価結果を示すことは難しいが、プロジェクト内のコミュニケーションの活性化に寄与した。

形式仕様記述言語のみで活用するのではなく、他の工夫との組み合わせにおいて効果を発揮するというのが非常に大事である。

形式手法の活用の効果



■ 直接的な効果

- 記述と検証 → 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、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

形式手法導入には直接的な効果と間接的な効果がある。

ストレスフリーな開発プロジェクトへ向けて



- 問題に対して即時に対応できるように
 - 影響範囲の明確化
 - 回帰テスト
- マネジメントに安心感を与えられるように
- 何よりも、プロジェクトに参加しているメンバが、日々ストレスフリーに過ごすことができるように

出荷後も何をするかというのも明確になっているので、問題の範囲なども把握できるため、更新時の影響範囲の明確化につながったり、回帰テストがしやすくなったり、派生開発がやりやすくなったりする。これは、マネジメントの安心感につながる。

サマリー



形式手法導入の成功事例から、形式手法導入の検討や計画立案の際に有用な知見を得ることを意図し、以下を学習

- 成功と言われている事例では、どのようにプロジェクトに形式手法を導入したか、どのような成果を得たか

参考文献



- 栗田太郎, 「仕様の記述力を鍛える モバイルFeliCa 開発における形式仕様記述手法の導入事例」, 日経エレクトロニクス 2007年2月12日号, 133—152.
- 栗田太郎, 「モバイル FeliCa のソフトウェア開発における品質確保のための構造と実践 抽象度の制御やコミュニケーションの活性化に向けて」, 情報処理学会デジタルプラクティス Vol. 1, No. 3 (July 2010), 148—157.
- 栗田太郎, 「形式手法の実践に対してよく尋ねられる質問とその回答」, SEC Journal Vol. 7 No. 1 (Mar. 2011), 34—39.

SEC-FM1-03-2

Copyright © 2012,2013 IPA

IPA Software Engineering Center 23

各文献の内容、テーマ

- 「仕様の記述力を鍛える …」

仕様記述作業の実際、見つかった誤り件数、適用した感想など。

- 「モバイル FeliCa のソフトウェア開発における…」

モデル検査の適用詳細、テストでの形式仕様の利用、コミュニケーションについて。

- 「形式手法の実践に対して …」

形式手法の導入を検討する際の典型的な疑問とそれに対する回答。



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

事例: 成功事例

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