Vol.2013-SE-179 No.26
2013/3/12

規範的チーム開発プロセスTSPiに基づく産学連携PBLの事例報告 –OpenFlowコントローラ開発への形式手法導入–

金丸 将平[†1]　岡部 正臣[†1]　山本 大輔[†1]　三善 浩司[†1]　青山 慎二[†2]　松村 成樹[†2]　荒木 啓二郎[1]
日下部 茂[1]　大森 洋一[1]　吉武 浩[2]　岩崎 孝司[2]　佐藤 義治[2]　山田 浩[2]　日下部 雄三[2]

概要：本稿では形式手法を用いて OpenFlow コントローラ開発の信頼性を高めることをテーマとした産学連携 PBL について述べる。特に、企業側の関心事のひとつである形式手法の開発プロセスへの円滑な導入事例の実施と、大学側の関心事のひとつであるチーム開発プロジェクトの教育を両立させるためのアプローチに焦点を当てて報告する。具体的には、開発プロセスとしてはカーネギーメロン大学より提供されているチーム開発プロセスの学生版である TSPi、形式手法にはモデル指向型言語であり要求分析工程に組み込みやすいと思われる VDM++と Alloy を用いて PBL を実施した。その結果、開発プロセスに対して形式手法を導入する知見を得られたほか、チーム開発プロセスに関する教育的な知見を得ることができた。

キーワード：PBL，TSPi，形式手法，OpenFlow

Case Report of Collaboration PBL between Industry and Academia Based on TSPi –Introduction of Formal Methods to OpenFlow Software–

1. はじめに

1.1 背景

上流工程で製品の信頼性を確保するために有効な手法として形式手法が提案されている．しかしながら，形式手法の導入事例はあまり多いとはいえない．こういった問題に対しては，従来の開発プロセスに対する手法の配置方法を含め，様々な事例によって知見を蓄積することで対応されていくものと思われる．

そこで，本 PBL では，OpenFlow コントローラの開発を題材とし，形式手法を導入した開発プロセスを実践しデータを収集することにした．これにより，開発プロセスへの形式手法導入の知見を獲得することができ形式手法導入の知見蓄積に貢献できると考えた．

[1] 九州大学 システム情報科学研究院
Kyushu Uniersity
[2] 富士通九州ネットワークテクノロジーズ株式会社
Fujitsu Kyushu Network Technologies Limited
[†1] 現在，九州大学大学院 システム情報科学府
Presently with Kyushu Uniersity
[†2] 現在，九州大学 工学部
Presently with Kyushu Uniersity

1.2 目的

今回の取り組みでは，チーム開発プロセスの実施及び形式手法による形式的検証という２点に着目し，後述する TSPi をベースとし形式手法である VDM++ならびに Alloy を導入した開発プロセスを実践し，その有用性を明らかなものとすることを目指す．

VDM++によってシステムを構成する要素・機能に関する厳密な仕様の記述，及びそれを基にした構成要素の抽象化を実施する．さらにそれらの記述を抽象化し同じく形式手法である Alloy による仕様の記述・状態遷移に関する網羅的な検証という工程を“要件定義工程”及び“設計工程”に組み込み，製品の品質を確保する．また，プロセスの具体的実践にあたっては，後述する TSPi に従い時間記録や欠陥記録といったデータを収集し，定量的に評価を可能とすることを目指す．

1.2.1 開発プロセスへの形式手法導入

今回の開発プロセスでは，以下に示す４つ考慮した上で

ⓒ 2013 Information Processing Society of Japan

開発を実施する．

(1) 開発プロセスのベースとして TSPi を使用する．
(2) 要件定義工程の作業として，システムを構成する要素に関して VDM++による陰仕様記述をおこなう．
(3) 設計工程の作業として，システム中の操作等に関して詳細設計として利用できるレベルまでの VDM++による陽操作仕様記述をおこなう．
(4) 設計工程での追加作業として，VDM++で記述された操作仕様の一部に対して Alloy を用いた網羅的検証をおこなう

これらの要素を踏まえ，実際の開発工程へと関連付けたものがが下の図 1 である．

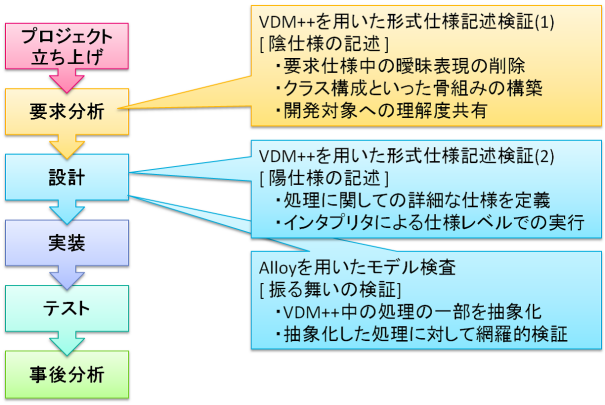


図 1　開発プロセスへの形式手法の導入方法

1.3　本論文の構成

本論文の構成について説明する．第 2 節では、本取り組みの背景にある PBL についての説明をおこなう．また，開発プロセスをどのようなメンバで，どのような役割配分において，どのような開発対象に対して適用したのかについての説明もおこなう．第 3 節では形式手法について述べ，今回の取り組みで導入した言語である VDM++や Alloy についての説明をおこなう．第 4 節では開発プロセスのベースとして用いた開発フレームワーク TSPi について説明し，提案プロセスにおいてどのような開発をおこない，どのようなデータを収集していたかを説明する．第 5 節では 3 節，4 節で説明した形式手法並びに TSPi をもとにして，VDM++と Alloy とを併用して開発した今回提案した開発プロセスの説明，並びに実施した結果どのようなデータや成果が得られたかを報告する．第 6 節では 5 節で説明・報告した導入方法で開発プロセスを構成した結果得られた開発プロセスに関するデータの報告、評価並びに課題点を考察する．そして最後に第 7 節にて今回の取り組みに対する全体的な考察と今後の課題について述べる．

2.　PBL について

本節では，今回の PBL についての説明をおこなう．

2.1　PBL とは

PBL とは、Project Based Learning の略称であり，課題として与えられたプロジェクトの実現や問題解決に学生が主体的に取り組み，ソフトウェア開発とプロジェクトマネジメントに関して基本的なスキル習得を目指す講義である．九州大学大学院システム情報科学府 QITO 社会情報システム工学コースでは実在する企業や大学教員から提示される実践系科目を学習題材として取り扱っており，４～６名で一つの開発グループを構成し，問題の解決にあたる．

なお，今回の PBL では富士通九州ネットワークテクノロジーズ株式会社が課題の提示並びにアドバイスを受けて実施した．

2.2　プロジェクト参加メンバについて

今回の取り組みでは，以下のメンバ構成，及び TSPi に基づく役割配置で開発をおこなった．

チームリーダ　修士 1 年．チーム開発の経験は PBL にて 0.5 年ほど．VDM++及び Alloy での検証経験者．開発メンバ全員と面識がある学生．

計画立案マネージャ　修士 1 年．チーム開発の経験は PBL にて 0.5 年ほど．形式検証の経験は一切なし．

開発マネージャ　修士 2 年．チーム開発の経験は PBL にて 1.5 年ほど．VDM++による検証経験者．開発対象に対する知識を豊富に持つ学生．

品質プロセスマネージャ　学士 4 年．チーム開発の経験は一切なし．VDM++による検証経験者．ソフトウェアの品質に関する研究をおこなっている学生．

サポートマネージャ　学士 4 年．チーム開発の経験は一切なし．VDM++による検証経験者．

2.3　Openflow とは

クラウド技術の発展・普及に伴い，データセンター内部で取り扱う複数の顧客の通信を個別に管理する，あるいは統合したネットワーク機器を適切な構成となるよう管理する，といったことが求められるようになり，管理すべきネットワーク機器の台数が増加してきている．

こういったネットワークの管理に求められる複雑な要求に対して有効と期待されているのが，ソフトウェアによってネットワークの構成や機能をプログラム可能とする「Software Defined Network(以降，SDN と略称で記述する)」という概念である．

この SDN という考えにより，ネットワーク機器の設定及び制御を単一化するという技術が求められ，ネットワーク間での経路制御をおこなうためにオープンなインター

ⓒ 2013 Information Processing Society of Japan

フェースである OpenFlow が着目されてきている．

なお，今回の PBL テーマでは，クラウド環境上で柔軟かつ適切にネットワークを自動制御できるシステムの OpenFlow による実現という背景を持つ．

3. 形式手法

形式手法とは， 数理論理学などに基づき，システムの注目する側面を正確に曖昧さ無く表現する事で，対象とするシステムに対する理解を明確にするとともに，そのシステムの満たすべき性質について科学的・系統的な検証をおこなう手法の総称である．形式手法を用いることで，集合や関数といった数学的な概念に基づいた仕様を記述することができる．例えば形式手法を用いることで顧客から提示された要求を自然言語に潜在する曖昧さを取り除いた仕様として記述することが可能となる．システム開発での形式手法の根底にある考えとして，開発の早期に可能なかぎりシステムの不具合となる原因を取り除くというものがあり，要求仕様や設計の段階で仕様中の不備を発見することで開発の下流工程での不具合発覚に伴う手戻りから生じる人的・時間的コストを削減する事が出来る．

3.1 形式仕様記述

形式仕様記述と分類される手法の多くは仕様の厳密な定義を目的としている．対象とするシステムに対して厳密な記述を行うことで，そのシステムを構成する要素，ならびにそのシステム中で行われる操作の中に矛盾あるいは曖昧さを残した部位や，未定義となっている部位が存在しない仕様書の作成を目指す．このような論理的に正しい仕様を記述するために考案された仕様記述には Z，B，VDM などといったものが存在し，VDM-SL と Z は ISO にて標準化されている．なお，今回の取り組みでは，VDM の中でも VDM++という言語で仕様を記述することとする．

3.2 モデル検査

モデル検査とはシステムの状態をモデル化することで設定した仕様が正しいことを検証する手法である．このモデル検査では，可能な状態遷移をツールに網羅的に調べさせることにより，検証したい性質が成り立つかどうかを検証することができる．「すべての取りうる状態を調べたが，問題となる状態に遷移することはなかった」あるいは「すべての取りうる状態を調べたところ，問題となる状態の遷移が存在した」ということをツールに報告させる．しかし，網羅的な検査とはいうものの，対象とするシステムの抽象化を行って「実行の順序や分岐がどう変わる可能性があるか」という部分に限定した抽象的なモデルを構築し，そのモデルを用いて網羅的な検査をおこなっているため，実際に「網羅的に実行」するというよりは，「網羅的に実行するのと理論的に同等の検査」をおこなっていることになる．なお，今回はモデル検査手法として Alloy を用いて検証をおこなうものとする．

3.3 提案手法に対して立てた仮説

今回の取り組みでは 2 つの形式手法を併用した開発プロセスを以下に挙げる 2 つの根拠をもって提案している．

互いに異なる検証の性質　VDM は仕様記述手法であり，仕様に対する厳密な記述並びにそれに対する型や構文のチェックをおこなうことができる反面，記述した操作のそれぞれが関連してどのような振舞いを取るかといった網羅的な検証はできないという特徴がある．一方で，Alloy はモデル検査手法であり，記述した仕様のモデルに対する状態の網羅的な検証やそのモデルそのものの可視化といったことができる反面，検証時の状態爆発を防ぐために仕様そのものに含まれる要素に対する詳細な記述はおこなえず，抽象化をおこなう必要がある．これをうけて，VDM と Alloy の互いの検証可能な性質を組み合わせることで検証可能となる性質の増加を見込むことができるはずである．

導入手法間の共通した概念　VDM も Alloy も共に集合論を基本として考案されている．それぞれの言語で仕様の記述を行った際，両者の記述中に類似した構文がある．2 つの記述間で記述対象の構成要素の型や構文に関して関係づけた記述をおこなうことで，両者で記述された仕様の整合性を保証することができるはずである．

今回の取り組みでは，上に挙げた 2 つの理由により VDM++と Alloy とを利用した形式的検証を提案し，それを実際に開発プロセスへと導入することとした．

4. TSPi とは

4.1 概要

カーネギーメロン大学ソフトウェア工学研究所によりチーム単位でプロジェクトを計画立案，管理するためのソフトウェア開発プロセス及びそのトレーニングとして提供されている TSP(Team Software Process) がある．なお，TSP は基本的に大規模なソフトウェア中心のシステムのか開発や拡張を実施するエンジニア 20 名ほどのチーム向け産業プロセスである．TSPi はその大学院・大学上級生コース向けフレームワークのことである．TSPi に従い実施した開発においては，実証されたエンジニアリング，そしてチームワーク手法の実践的な経験を得ることができるとされている．

4.2 TSPi の構造と流れ

TSPi は以下の図に示したように複数の開発サイクルを実施して最終製品を開発するというものである．それぞれのサイクルは，立ち上げ，開発戦略，計画立案，要求，設

計，実装，テスト，事後分析といった 8 段階の工程に分割されて管理され，それぞれの工程でどのような作業を実施するのか，そしてどのようなデータを収集するのかといった事項がスクリプトとして定義されている．

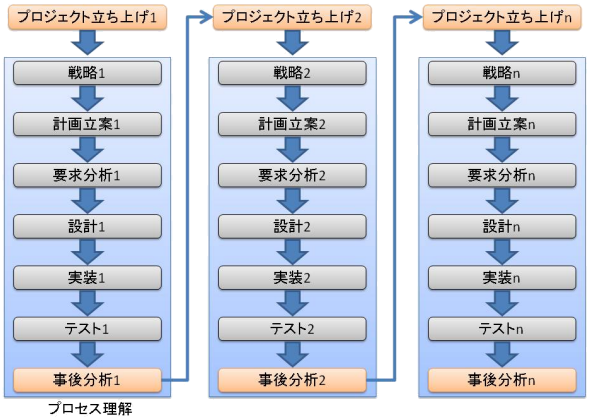


図 2　TSPi で実施する開発サイクルの流れ

そして，これに従い参加エンジニア達はサイクル 1 で作成したベース製品を作成し，以降のサイクルで更に機能拡張の開発を実施していくこととなる．

4.3　方針

本 PBL は SDN という比較的新しい分野のシステムを開発対象としており，適用方法が必ずしも確立していない形式手法を活用することが求められていた．

そのため TSPi をベースに複数の開発サイクルを実施し，段対象ドメインに対する知識を段階的に獲得し形式手法もの段階的に適用していくこととする．

開発に際し、TSPi の仕組みを利用して各種データを収集し，考察をおこなうこととする．

サイクル 1　サイクル 1 では「開発プロセスへ形式手法を導入する際の知見を収集する」を目的として設定し，開発をおこなうものとする．具体的には TSPi による開発にメンバが慣れ，OpenFlow に関する知識を獲得し，そして形式手法による検証の基礎を学ぶため，VDM++単体による仕様記述を要件分析工程，設計工程に対して導入する．

サイクル 2　サイクル 2 では，サイクル 1 で学んだ TSPi や形式手法に関する知見を活かし「SDN 開発に対する形式手法による検証の有効性の確認」を目的として設定し，開発をおこなうものとする．なお，サイクル 2 では形式手法としてモデル検査に類する Alloy による検証をサイクル 1 記述した形式仕様記述に対して導入する．

5.　開発プロセスへの形式手法導入

5.1　形式手法の位置づけ

今回の取り組みにおいては，形式手法を上流工程である“要件定義”および“設計”工程で使用することとした．これは形式手法の特徴である「厳密な仕様の記述」並びに「記述対象の網羅的検証」を最大限に活かすことが出来る配置だと考えての判断である．また，これらの検証で利用したテスト項目を，テスト工程で回帰テスト的に利用することも考慮して上流工程へと配置することとした．

また，TSPi の方針に沿って複数回の開発サイクルを実施することにし，サイクル 1 では形式仕様記述手法の VDM を，サイクル 2 ではモデル検査手法の Alloy を導入している．

5.2　サイクル 1

本節では，形式仕様記述言語である VDM++による仕様記述を開発工程へと導入した開発サイクルの 1 回目について報告する．

サイクル 1 では，まだメンバの一部に開発に関するノウハウが蓄積していないということもあったため，最初から仕様の全てを検証することを目的とはせず，“開発プロセスに対して形式手法を導入する場合の知見を収集する”という目的のもとで開発をおこなった．

サイクル 1 では，以下に示す事由より，形式仕様記述言語の中でも VDM++を選択することとした．

(1) VDM++による検証経験のあるメンバが開発チームに多く存在していた
(2) 実装言語である C++と同じくオブジェクト指向を考慮した言語で記述できる
(3) 開発プロセスでの位置づけがわかりやすい
(4) 仕様の段階でインタプリタによる実行ができる
(5) インタプリタによる実行結果をテスト工程にて再利用できる

具体的には，従来の開発プロセスにおける「要件定義」「設計」という 2 つの工程に対して，VDM++での陰仕様記述，陽仕様記述をそれぞれ割り当てることで仕様の詳細化並びに検証をおこなうこととした．なお，今回の取り組みではチーム開発への形式手法導入の知見獲得という観点より，複数人で VDM++による記述をおこなうことにかすたまいずした導入の仕方となっている．

具体的には，形式仕様記述を作成するにあたり複数回のすり合わせをおこなうことを視野に入れ，下図に示す 5 段階に渡る段階的詳細化を実施した．これにより，メンバ間での開発対象に対する理解の相違の解決，さらに各メンバが担当しているクラス間でやり取りする各種データにおける整合性の確保を可能とし，複数人で単一の作業を並行し

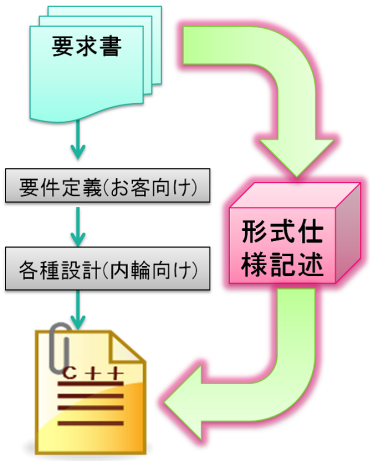


図 3 形式仕様記述で要件定義・設計を行う開発のイメージ

て実施する際に生じる問題を解消することができた．

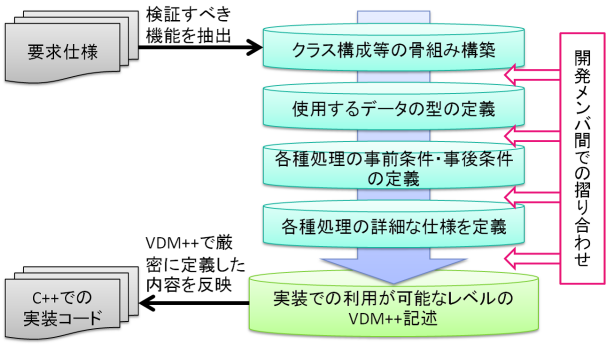


図 4 今回実施した複数メンバでの VDM++記述の流れ

5.3 サイクル２

サイクル２においては，サイクル１で得られた形式手法導入の知見を基にして，モデル検査手法による検証を実施することで，“SDN 開発に対する形式手法による検証の有効性を確認する”という目的を掲げて開発を実施した．

そして，このサイクル２では以下に示す事由により，モデル検査に分類される形式手法の中でも Alloy を選択することとした．

(1) VDM++と同様にオブジェクト指向の記述をおこなうことが出来る
(2) 指定した制約下で経路を発見できるか否かを探索することが出来る
(3) 経路が発見された場合はその経路を追跡することが出来る
(4) (2)(3) の特性を複数のデータに対して同時に検証することが出来る

このサイクル２で実施した導入方法は，サイクル１で実施した開発プロセスに対して上述の Alloy による検証を付加するというものである．具体的には， VDM++ による厳密な仕様の記述をおこなうところまではサイクル１と同様だが，その後 VDM++において記述されている処理アルゴリズムの一部を Alloy で検証し VDM++に反映させるという方法をとっている点がサイクル１とは異なっている．なお，このアプローチは，VDM++による記述を挟まずに検証対象である全ての記述を Alloy で記述するアプローチとは以下の点で異なっている．Alloy は網羅的に探索するという検証の性質上，検証できる範囲が小さいという問題がある．それ故に検証の対象として VDM++で記述している巨大な探索空間を指定してしまうと，検証器が探索可能領域の限界を超えて検証がおこなえなくなってしまう“状態爆発”を起こしてしまう．それ故に，Alloy によるモデル検査では VDM++ で記述した仕様の中でも，特に検証が必要と思われる「経路の計算処理」などといった一部に対して限定的に適用することとした．

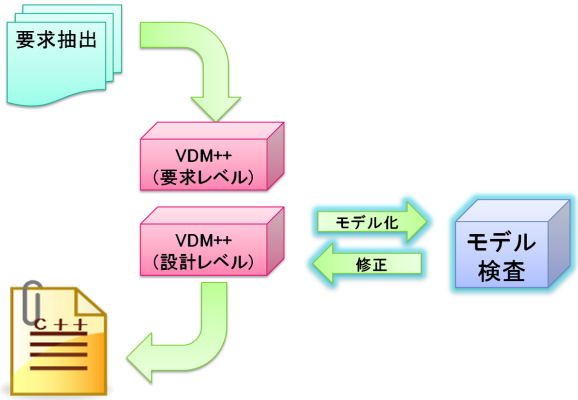


図 5 前述の VDM++を導入した開発プロセスへのモデル検査導入のイメージ

なお，今回実施したモデル検査の流れとしては，充足例の検出と反例の検出という２段構成となっている．１段目では構築したアルゴリズムが正しい経路を算出できているかを充足例の検出により検証し，それが適当であるとわかったならば２段目へと進む．そして２段目では，反例の検出を利用して算出してはならない異常経路を算出していないかの検証をおこなう．そしてこれらの検証を通して得られた情報を記述の元である VDM++に対して反映させるという手法をとった．

6. プロジェクト実施の結果と考察

本節では今回実施した形式手法を導入した開発プロセスで得られたデータを分析し．今回の取り組みの考察をおこなう．

6.1 形式手法導入の結果

本節では、今回の形式手法の導入の結果、VDM++並びに Alloy で発見することができた各種毛かんなについて報告する．

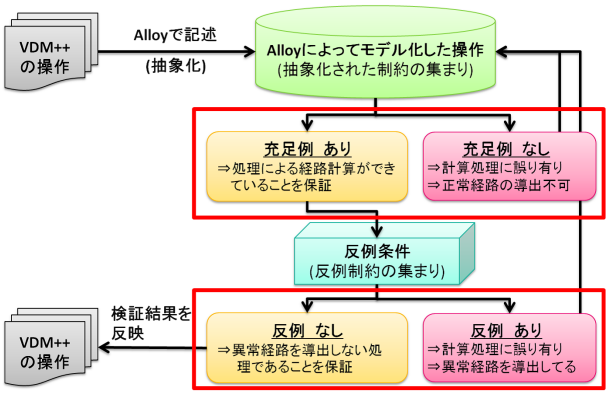


図 6 Alloy による VDM++記述の振る舞い検証の流れ

6.1.1 サイクル 1 での形式手法導入の結果

サイクル 1 では以下に挙げる仕様上の欠陥を検出することができた．

用語の曖昧さ 仕様書に含まれていた複数の意味で受け取ることのできる用語を発見し，それに対して一意となる定義を設定することができた．

データ表現の一貫性 仕様書やメンバ間で担当した VDM++記述内で，同一の対象に対して複数の表現が行われていたことを発見することができた．

メンバ間の解釈の違い 口頭でのミーティングにて自然言語で定義した用語に対して，誤解を抱いていたメンバを発見し，訂正することができた．

インターフェースの不整合 メンバごとに担当したクラス間でやり取りするデータの不整合を解消することができた．

6.1.2 サイクル 2 での形式手法導入の結果

サイクル 2 では以下に挙げる仕様上の欠陥を検出することができた．なお，いずれもの欠陥もある特定の条件下でないと露見するものではないため，人力によるテストでは見落としてしまう可能性が多分にある欠陥といえる．

要件定義の誤り 反例検出で発見された欠陥で，VDM++で記述する前の要件定義で作りこまれたものである．この欠陥は VDM++によるインタプリタ実行で検出できる範囲に含まれるが，ある特定の条件を指定しない限り検出することはできないものである．

事後条件記述の誤り 反例検出で発見された欠陥で，VDM++での操作の記述時に作りこまれたものである．この欠陥も VDM++によるインタプリタ実行で検出できる範囲に含まれるが，ある特定の条件を指定しない限り検出することはできないものである．

6.2 開発プロセスにて得られた時間記録

今回実施した 2 つのサイクルにおいて得られたデータを表 1 に示す．なお，サイクル 2 に関しては，PBL の実施期間の関係上取り組みが終了していない部分があるため記録されていない．本節以降では，特にサイクル 1 のデータを用いて考察をおこなっていくものとする．

表 1 本 PBL での時間記録

実施サイクル	立ち上げ	開発戦略	計画立案	要求	設計	実装	テスト
サイクル 1	14.4h	1.7h	8.4h	38.1h	23.3h	16.8h	11.8h
サイクル 2	4h	6.2h	4.7h	16h	31h	データ未取得	データ未取得

6.2.1 今回の導入方法が各開発工程に与えた影響

本節では，今回の取り組みで得られたデータを，表 2 に示す 2010 年度に九州大学で実施された PBL と比較することで，今回実施した形式手法の導入方法が開発工程に与えた影響について分析していくこととする．

表 2 2010 年度 PBL の時間記録

立ち上げ	要件定義	設計	実装	テスト (システムテスト除く)
118.5h	245.8h	146.28h	172.7h	157.9h (75h)

なお，分析する際の観点としては，「開発作業始動前のプロジェクト立ち上げに要した時間」及び「開発作業始動後の開発時間全体に占める各工程の割合」を取り扱うものとする．

今回のような絶対的指標と相対的指標の 2 つの観点からの分析をおこなうに至った理由としては，各 PBL が対象とする開発物の規模に差が存在することに関係している．本来，開発対象の規模が異なる場合，開発工程のそれぞれで要する時間は大きく異なってくる．しかし，開発を始めるために実施するプロジェクト立ち上げにおいてはその限りではない．プロジェクトの立ち上げ作業で実施するのは，開発計画や品質管理計画，リスク管理計画といったどのようにプロジェクトを進めていくかといった方針を決定するものが多く，作業見積りに使用する概念設計を除き，基本的に開発対象の規模による影響は極めて小さいと考えたためである．また，立ち上げ後は開発対象の規模の代償による影響があり，例えば開発期間が長くなる場合，作業工程のいずれもが要する時間もそれと比例して大きくなると考えられ，開発時間全体に対する割合に着目して分析することが可能となると思われる．このような理由のためにこれら 2 つの基準をもって考察することとした．

また，今回比較対象とした PBL データは，今回の取り組みと同様に「学生以外がテーマを設定したグループ」かつ「形式手法を用いた開発がテーマに含まれてるグループ」を選択している．学生が主体となって取り組むテーマを提案しているグループでは，実際に PBL として活動を始める以前から開発作業を始めていることが多く，今回の観点では公平な比較ができないと判断したためである．また，既に形式手法を用いた開発を経験しているグループであれば，今回我々が提案したプロセスとの対比を行うのに適していると判断したためである．

6.2.2 プロジェクト立ち上げに与えた影響

今回の PBL では予め確立された開発プロセスである TSPi をベースとして開発をおこなったために，開発のどの工程で何を為し，そして何をデータとして記録するかということを予め定義していた．プロジェクト始動に関しても指針とするものがあり，チーム開発そのものに対して経験の浅いメンバがいるにもかかわらず迅速なプロジェクトの立ち上げを行うことができ，スムーズに開発へと移行することができた．なお，今回の取り組みで得られたデータを 2010 年度に九州大学にて実施された PBL と比較したものが図 7 となる．

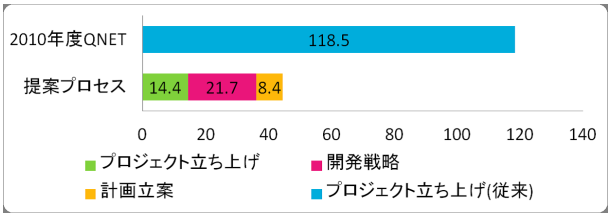
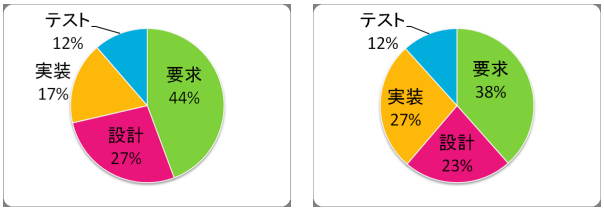


図 7 プロジェクト立ち上げに要した時間比較

6.2.3 開発時間全体に占める各工程の割合

今回の開発プロセスで得られたデータと 2010 年度 PBL グループのデータにおいて，開発時間全体に対する各作業工程の時間比率を円グラフにて表示したものが図 8 である．



(a) 今回の開発プロセス　　(b) 2010 年度 PBL

図 8 各作業工程の時間比率

今回実施した PBL と，2010 年度 PBL グループとが採用した形式手法の導入方法は，形式手法を設計の段階にまで組み込ませたか否かという点で大きく異なっている．今回の取り組みでは設計の工程で VDM++による操作記述を詳細設計としての利用に耐えうるレベルまでに詳細化したが，2010 年度 PBL グループは VDM++の記述を要求レベルまでにとどめ，設計の工程では各種 UML による設計をおこなっていた．

結果として，今回は実装時に VDM++による操作仕様を参照しながらのコード実装が可能となり，テスト工程にかけた時間の比率こそ同じだが，実装にかけた時間比率を形式手法の経験がある 2010 年度 PBL グループよりも小さくすることができたことがわかる．5 名の開発チームの中にチーム開発未経験者と形式手法未経験者が合わせて 3 名いるにもかかわらず，今回のような成果を残す事ができたのは提案プロセスが効果的に働いたからだと思われる．

6.3 今回の形式手法導入に対する評価

今回実施した開発プロセスでは，本節で述べたように形式手法の持つ効果を十分に感じられる状態で開発プロセスに対して形式手法を導入することができた．また，過去の PBL プロジェクトの実施データと比較することで，開発時間全体に占める上流工程の割合が増加する反面，実装やテスト工程においてかかる負担が大きく減らすことができるという効果を確認することができた．

なお，今回の実施メンバ構成は冒頭でも述べているように，形式手法による検証経験者，およびチーム単位での開発の経験者が非常に少ないメンバで構成されている．本来ならば新しい開発のプロセス実施データを採集する場合，チーム開発の経験が豊富な者，使用する技術に精通している者で開発メンバを構成し，実施することが適切であると思われる．しかし，実際の開発では常に何かしらの新しい特殊条件が付与された開発プロセスに従わなければならない，あるいは経験の少ない技術を利用して開発を実施しなければならないという場面も多分にある．

このような開発現場の現実的な状況を考えると，今回のメンバ構成で得られたデータは，チーム単位での開発に不慣れ，あるいは形式手法に不慣れなメンバが含まれたメンバ構成であっても，今回示したデータのような効果を一定して受けることが出来るということを示していると考えることもできる．言うまでもなく，チーム開発の経験や形式手法の使用経験の豊富なメンバがより多く在籍している開発チームなどに対して今回の提案手法を適用したならば，今回以上の開発効率を実現できると考えられる．

6.4 提案プロセスにの分析に対する課題

しかし今回得られた結果は，考察に用いたデータにおいて以下に示す 2 点の問題が含まれているために，正確な分析のためには今後更なるデータの収集と補正をおこなっていく必要がある．

6.4.1 収集データの定義

1 点目は，収集データの定義がある．これはプロジェクト立ち上げ時に予め収集するデータを明確化していたにも関わらず，それらのデータがどのようなものかという厳密な定義をメンバ間で深堀りしていなかったことに起因している．

例えば，本来 TSPi のメンバはその個人開発版とも言える PSP(Personal Software Process) を学習済みであることが求められる．例えば，PSP ではある作業を実施する際に必要となる知識を収集する「学習」という時間は開発作業時間には含まないという計測時間の定義が存在している．しかしながら，PSP を経験していない 2 名の開発メンバは

こういった計測時間の定義を知らなかったために，学習に費やした時間も開発作業時間であるとし計測してしまっていた．こういった見解の相違により，例として挙げた時間の計測に限らず，その他の収集データにおいても定義の相違が存在している可能性がある．そのため，今回のデータはメンバに確認をとるなどしてより時間をかけたデータの分析・修正が必要であると思われる．

6.4.2 未実施フェーズの存在

2 点目は，未実施のフェーズの存在である．今回の取り組みは学内における PBL の講義の一環として実施したものであるため，週間の実働可能時間やプロジェクト実施期間という面において締切が存在していた．そのため，サイクル 1，サイクル 2 ともに正規の時間内にシステムテストを実施する時間を確保することができなかった．システムテストに関するデータを除いたデータの考察しかできていないため，ステークホルダの了承を得て時間外にシステムテストをおこないデータを取得する予定である．

7. おわりに

今回の PBL の実施を通して，目的として掲げていた，TSPi をベースとした形式手法の開発プロセスへの導入や，形式手法による形式的検証の有効性というものを確認することができた．今後は今回獲得した知見を基に開発の効率向上や検証の範囲拡大といった改善を施し，より複雑な開発対象に対する開発データを収集していくことを目指す．

また，今回の研究では収集するデータが不完全であったり，予定していた開発サイクルを完遂できていないといった問題点もある．後の研究においては，プロジェクト参加メンバ全員に対して収集するデータを徹底すると同時に，開発プロセス全てのデータを収集することが出来るよう余裕を持ったプロジェクト実施期間を確保し，より厳密な考察を可能とすることを目指す．

参考文献

[1] Watts S. Humphrey，秋山義博 (監訳),『TSPi ガイドブック』，翔泳社，2008
[2] Daniel Jackson，中島震 (監訳)，今井健夫 (訳)，酒井政裕 (訳)，遠藤侑介 (訳)，片山欣夫 (訳),『抽象によるソフトウェア設計 -Alloy ではじめる形式手法』，オーム社，2011
[3] Jhon Fitzgerald，Peter Gorm Larsen，Paul Mukherjee，Nico Plat，Marcel Verhoef，酒匂 ,『VDM++によるオブジェクト指向システムの高品質設計と検証』，翔泳社，2010
[4] 高宮安仁，鈴木一哉，クラウド時代のネットワーク技術 OpenFlow 実践入門，技術評論社，2013
[5] 情報処理推進機構，形式手法適用調査報告書，2010
[6] SDN/OpenFlow， http://itpro.nikkeibp.co.jp/openflow/, 日経ＢＰ社, 2013-02-12
[7] ディペンダブルシステムのための形式手法の実践ポータル, http://formal.mri.co.jp/, 石黒正揮, 石川 冬樹, 2013-02-12