# PBL 形式による組込みシステム開発演習教育 – 開発プロセスの発見と品質特性の実現 –

鈴木 正人, 矢竹 健朗, 青木 利晃, 落水 浩一郎†

北陸先端科学技術大学院大学 (JAIST) では、産学連携専門教育プログラムとして「高信頼組込みシステム開発技術に関わる基盤的人材育成プログラム」を推進している。当プログラムでは、アジア各国からの留学生を対象に、将来北陸地区の企業に就職して組込みシステム技術者として活躍するために、必要な知識と経験を、現実の製品開発体験を通じて習得させることを目標とする。本プログラムの特徴は以下の 3 つである。(1) 演習は PBL(Project Based Learning) 形式で行う。(2) 課題は北陸地区の協力企業が過去に開発した現実の製品に基づく。(3) 開発手順 (プロセス) を教師が与えるのではなく、受講生自らが発見、設計、改良する。

平成 21 年度、22 年度の 2 年間に渡り、計 5 個の課題について演習講義を実施した成果を報告する。特に新しい試みとして実問題への対応、アーキテクチャ設計/テスト設計、品質の作り込みについての教育手法とその効果について述べる。

# Report of PBL based Education for Embedded Systems – Creation of Processes and Quality Assurances –

MASATO SUZUKI, KENRO YADAKE, TOSHIAKI AOKI, KOICHIRO OCHIMIZU

We have been engaging in a Industry-University-Collaboration program ”Strategic human resource development program for highly dependable embedded system technologies.” This is a carrer development program for foreign student from many Asian countries. They can study about embedded systems through develpmemnts of actual product and introduced for getting job at companies in Hokuriku area after they get degrees. Fetures of this program are as follows: (1) Practice is based on PBL, (2) Use actual development problems based on items developed by companies as products, (3) Processes are not given, but are found by students themselves. Also they must improve the processes in order to achive quality requirements.

This article we discuss about the effort of our practical education in 2009-2010, with five case studies. Management of requiements in real world, design of architechtures/tests, and assurance of quality in the processes are our original features.

## 1. はじめに

北陸先端科学技術大学院大学 (以下 JAIST) では、開学以来 20 年にわたり留学生の教育に力を入れているが、平成 20(2008) 年度から北陸地区の ICT 関連企業との連携により高信頼組込みソフトウェア開発技術者の教育育成プロジェクトを発足させることになった。これは経済産業省と文部科学省の支援により、アジア各国からトップレベルの大学の工学系学部を優秀な成績で卒業した学生を国費留学生として募集し、留学生は、JAIST での半年間の研究生および 2 年間の修士課程学生として通常の研究生活を送るのと同時に、従来の講義体系では扱うことが難しかった、組込みシステムに特化した開発に必要な知識や経験を、実践を通じて習得させるものである。なお修了後留学生は、北陸地区の企業に就職し、組込みシステムの技術者として最低 5 年程度従事した後は各自のプランに従ってそのまま日本あるいは出身国に戻って活躍を続けることができる。これにより、大学 (JAIST) は優秀な留学生を獲得、企業は現在および将来のアジア各国への組込み事業展開の中核となる人材を獲得、留学生は日本への留学と組込みシステム開発技術者としての知識と経験およびその後の就職の道を得られるという、3 者が Win-Win-Win の関係になることを狙いとしたものである[1)]。特に留学生の修了後のキャリアパスまで視野に含んだプロジェクトは他に類を見ないものである。

本プロジェクト「高信頼組込みシステム開発技術に

† 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

関わる基盤的人材育成プログラム」では、以下の各項目に示す能力の習得を目的としている。

- 組込みシステム開発に必要な知識およびツールを使用する能力
- 信頼性実現のための先端技術を実際の開発に応用する能力
- 日本の企業でチームで活動するための協調およびコミュニケーション能力

なお対象となる留学生は、文部科学省の支援による「高信頼組み込みシステムに向けた大学院教育コア形成の促進」プロジェクトで設置された高信頼組み込みコースに他の日本人学生と共に参加している。このコースは組み込みシステムの開発に必要となる基礎科目を必須としたものであり、修了すると通常の修士の学位のほかにコース修了証を得ることができる。

平成 20 年度から現在まで、著者ら (主にソフトウェア工学を専門とする) は、組込みシステム開発のための PBL(Project Based Learning) 教材の開発および実践教育に従事してきている。平成 21,22 年度の 2 回にわたり、延べ 15 名の留学生がこの演習講義を受講しており、23 年度も 11 名に対して実施予定である ( 図 1. 矢印部分は日本語教育など)。

本稿では PBL 演習講義を通じて組込みシステム開発に必要な先端技術とそれらを開発現場に応用する能力を習得させるためのカリキュラムの設計、教材の設計、および実践によって得られた知見について報告する。特に高品質なシステムを開発するために、受講生自身に品質特性の実現方法の提案とそれを可能とする開発手順 (プロセス) を設計させ、品質に対する深い理解とその実践スキルを習得させる点が特徴である。

本稿の構成は以下の通りである。2 節では PBL 演習講義のカリキュラムとして、製品開発とプロセス設計の 2 つの講義を行うに至った経緯を説明する。3 節で教材開発の手順および採用した 5 個の課題の特性および狙いについて述べ、4 節では平成 21 年度に実施した演習講義の詳細を、5 節では結果の評価と得られた知見について述べる。カリキュラム、課題、使用ツールなどについて改善を行い、22 年度に実施した内容を 6 節で、21,22 年度を通じて得られた PBL 全体に対する受講生および参画企業からの評価を 7 節で述べる。

## 2. カリキュラム設計

### 2.1 全体方針

本演習講義では企業と大学が協力して教材開発および教育カリキュラムを設計/実践することを目標としている (図 2)。プロジェクト開始時点で各年度あたり

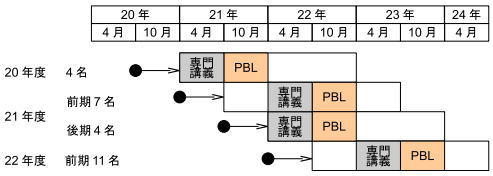


**図 1** 事業スケジュール

2 単位 (1 回 1.5 時間× 15 回) の講義☆を 2 回行うことは決定していたが、教育内容および時間配分等は未定であった。著者らに岸知二氏 (当時本学特任教授、現早稲田大学教授、企業におけるシステム開発に精通) を加えたプロジェクトチーム内で平成 20 年 4 月から 6 月にかけて何回か検討会議を行い、目標達成のために以下のような基本方針が決定された。

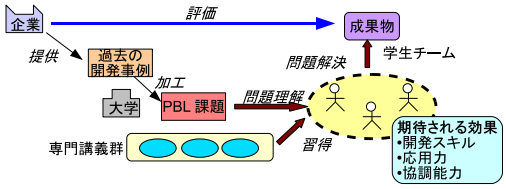


**図 2** PBL におけるステークホルダーの関係

- 先端知識を実問題にどう応用するかを考えさせる演習とする。
- 実際の製品を題材とすることで、現場で発生している問題を受講生に理解させる。
- 単に作り方を教えるのではなく、プロセスを自分達で発見/改善させる。
- 自分で設計したプロセスを適用することで製品 (の一部) を開発させる。
- 様々な品質特性とその実現方法を受講生に理解させる。
- 開発過程を通じて組込み固有の開発環境に関する知識や信頼性実現技術を習得させる。

### 2.2 演習講義内容

演習講義は JAIST の専門講義で学んだ先端技術を現場に応用する能力を習得することを目的としている。このため受講生に必要な知識を以下のように定めた。

(1) ソフトウェア開発の一般的な手順 (要求分析/設計/実装, テスト) を理解していること
(2) 計算機アーキテクチャおよび言語/コンパイラ等の環境について基礎的な知識を得ていること

☆ JAIST では 2 単位の講義は週 2 回× 8 の 2ヶ月間で実施される

**表 1** 高信頼ソフトウェアプロセス設計演習シラバス (初版)

| | |
|---|---|
| 第 1 回 | プロセスモデル概論 |
| 第 2 回 | 品質特性概論 |
| 第 3-7 回 | 標準プロセス設計 [演習] |
| 第 8 回 | 中間発表 |
| 第 9 回 | 演習課題特性説明 |
| 第 10-13 回 | 固有プロセス設計 [演習] |
| 第 14 回 | ドキュメント作成 |
| 第 15 回 | 最終発表 (成績評価) |
| 第 16 回 | 学外発表 (企業による評価) |

**表 2** 高信頼ソフトウェア開発演習シラバス (初版)

| | |
|---|---|
| 第 1 回 | 組込みシステム開発概論 |
| 第 2 回 | 組込みシステム開発環境 |
| 第 3 回 | 演習課題説明 |
| 第 4-7 回 | アーキテクチャ/コンポーネント設計 [演習] |
| 第 8 回 | 中間発表 |
| 第 9 回 | 実装/テスト技法概論 (講義) |
| 第 10-13 回 | 実装/テスト実施 [演習] |
| 第 14 回 | ドキュメント作成 |
| 第 15 回 | 最終発表 (成績評価) |
| 第 16 回 | 学外発表 (企業による評価) |

(3) C 言語によるプログラムおよび UML 図の読解能力があること

これらの条件を満たすために、受講生には入学後ただちに専門講義 (ソフトウェア開発方法論、計算機言語論など) を受講させ、PBL は修士 1 年目の秋 (10-11 月) と冬 (12-01 月) に以下の 2 つの講義として実施することが決定された (図 1)。

(1) 高信頼ソフトウェアプロセス設計演習
(2) 高信頼ソフトウェア開発演習

2 つの演習講義の初版のシラバスを表 1, 2 に示す。他に受講生は 3 名で 1 チームとする、最初に一般的な開発技術やプロセスモデルなどの講義を行った後は 4 回をひと区切りとした演習を複数回行い、各演習の終了後に中間/最終発表を行う、なお 4 回で演習課題が終了しない場合は放課後などに適宜演習時間を設けることにする、などの細部が決定された。なお最終発表は受講生の理解度の確認および成績評価のための学内発表と、参画企業の方に出席して頂いて現場の観点から意見を頂くための学外発表の 2 回とした。これは評価の観点が異なる他に、受講生のほとんどが留学生であり対外発表を行うためには発表資料 (主に日本語文章) の添削が必要となる点を考慮したためでもある。

当初計画では図 3 上に示すようにプロセス設計演習を先に実施して標準プロセスを基に受講生に課題固有プロセスを作成させ、ソフトウェア開発演習では自分で作成したプロセスに従って課題の開発を実行させる予定であったが、21 年 4 月に実施した受講予定者へのインタビューの結果、過半数の学生が体系的なソフトウェアの開発およびテストに関する知識と経験を持っていないことが判明した。この状態から受講生が満足な課題固有プロセスを設計することはほぼ不可能であると考え、同図下のように、(プロセスを与えずに) 先に開発演習を実施することで開発に伴う問題を認識させ、その後にプロセスの導入によりそれらを解決するとともに品質特性実現のためのプロセスの改善方法を考えさせるという順序に変更した。

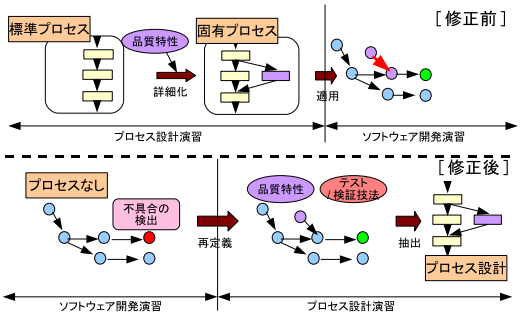


**図 3** 演習講義の実施手順変更

## 3. 教　　材

### 3.1 教材開発手順

平成 20 年 7 月から参画企業の製品開発担当者との協議を行い、受講生が開発を体験する教材を選定した。いずれも高信頼性の実現を大きな軸とした上で課題毎に並列制御や資源の有効利用などの特徴を持たせたものとなっている。各参画企業から教材として提供された課題を検討し、開発期間や予算などの条件により以下の 5 課題を選択した。他に複数の CPU を接続した液体充填装置の制御などが提案され、組込みの典型的な課題として実現可能性を検討されたが、規模や設備の入手性などの点で残念ながら不採用となった。

**表 3** 課題一覧

| 記号 | 対象 | 特性 |
|---|---|---|
| KN | 家庭用コンロ制御基盤 | 実時間制御 |
| PB | プリンタ出力抑制器 | 複雑な状態制御 |
| DA | ネットワーク情報収集器 | 複数装置の並列制御 |
| EX | 製品画像検査器 | 機械制御、画像処理 |
| SC | ネットワークスキャナー | 資源有効利用 |

いずれも開発期間 (約 1ヶ月) 内に企業側から提供された全部の仕様を満たすことは不可能なので、担当者間で協議のうえ仕様の取捨選択を行った。また必要に応じて課題毎の特性に関連する機能仕様/品質仕様を追

加し、最終的に各課題とも 10 ページ程度の機能仕様および品質仕様書として受講生に説明する資料 (図 4) を作成することとした。フォーマットを実際に企業で使用されている形式にあわせるため整形は協力企業に外注し、21 年 9 月末までに最初の 2 課題 (KN,PB) を整備して 21 年 10 月より実施、残り 3 課題 (DA,EX,SC) は 22 年 9 月末までに整備することになった。

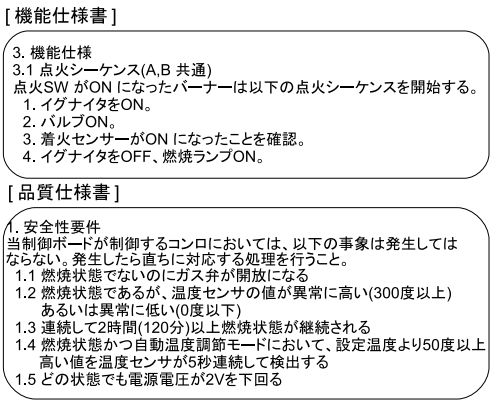


**図 4** 機能仕様書、品質仕様書 (KN の一部)

### 3.2 課題の特性と狙い

当プロジェクトは以下を実現することを目標としている。このためカリキュラムおよび教材には様々な工夫が盛り込まれている。

(1) 受講生自身に問題点を発見させ、解決するプロセスを定義させる。

(2) アーキテクチャの役割を理解し、それに基づいた設計手法を習得させる。

(3) 様々な品質特性とその実現方法を習得させる。

(1) に関しては、ソフトウェア開発において実際に発生する現象を受講生が体験できるように、あらかじめいくつかの問題を教材に作りこんでいる。例えば図 5 は KN 課題のエラー検出およびエラーからの回復に関する部分の仕様である。コンロには A,B の 2 系統があり、エラー検出項目に関しては同等である。

仕様書では複数の系統でエラーが同時に発生した場合の記述を意図的に省略してある。これは受講生が実装またはテストの過程で、例えば A がエラーコード 201 のエラーを発生し、それが解消 (点火 SW-A が OFF) される前に B がエラーコード 302 のエラーを発生するといった状況に遭遇し、そのときの振舞いが仕様に定義されていない、という点に気が付くことを期待している。もし受講生が気づかない場合にはインストラクターが「2 つの系統を同時にテストしたら何が

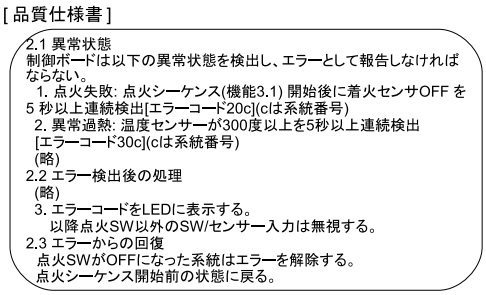


**図 5** エラー処理および回復の仕様 (KN の一部)

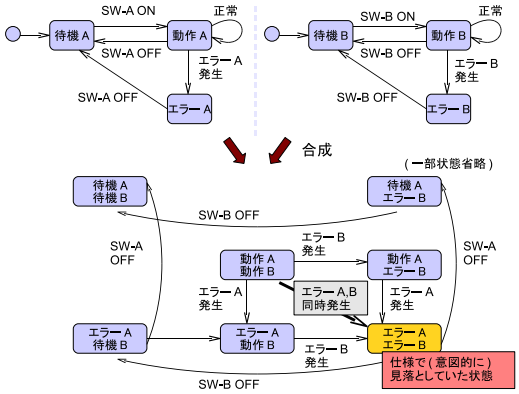


**図 6** エラー発生/回復の状態遷移図とその合成

起こるか?」といったヒントを与えてエラー同時発生という状況に誘導する。いずれにしても今回仕込んだ仕様書の不完全さをできるだけ早期に発見し、それを解決するための手段を考察する機会を与えるのが狙いである。不完全さの発見の手段としては以下のようなものが考えられる

- 仕様のレビューにおいて状態遷移図を作成すること
- 系統毎の検証に加えて複数系統の並行動作を検証するため状態遷移図の合成を行うこと
- これらをアクティビティとしてあらかじめプロセス中に組み込んでおくこと

期待する状態遷移図の例を図 6 に示す。エラー検出を周期的に行っているため、エラー A, エラー B の各イベントが同時に発生する場合があるが、これを仕様では見落としていることが、状態遷移図を作成することによって発見できる。

受講生は状態遷移図については前提講義で学習しているものの、読んだことはあっても実際に記述するのは初めてという者も多く、補講を行って状態の合成方法などの基礎知識を与えた。

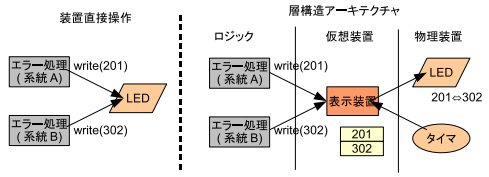


**図 7** アーキテクチャ導入の有無による仕様追加への対応の差

なおこのような状態遷移図の網羅性に関する検証はモデル検査を利用して行うことが理想的であるが、初年度は (主に日程の都合により) 受講生がソフトウェア検証理論に関する基礎的講義を受講していないことを考慮し、モデル検査はオプションとして希望者のみ実施することとした。結果として実施したチームはなかった。

(2) に関しては、アーキテクチャの概念およびトップダウンに設計を行うことの重要性を理解させることを目的とし、後から仕様追加をいくつか発生させることにした。例えば先のエラー処理の例では、物理的に数字表示器 (3 桁の 7 セグ LED) は 1 組しか装備していないため、複数系統でのエラーが同時に発生した場合にはどちらか 1 つのエラーコードしか表示できない。仕様のレビューの結果、受講生がこれに気が付いて解決方法について質問が出た場合には、2 つのエラーコードを時分割 (例えば 1 秒置き) で表示させる機能を追加仕様として与えることになっている。このとき図 7 左のように、エラー処理部が表示器 (デバイス) を直接操作するような設計/実装になっていると、この追加仕様に対応するには構造の大幅な変更を必要とする。

図 7 右は層構造 (layered) アーキテクチャを導入した実現例である。エラー表示器は実際の装置ではなく、LED とタイマを組み合わせた仮想装置として実現されている。上位層 (エラー処理部) は単にエラーコードを `write(v)` で仮想装置に書けばよい。`write` が最初の呼び出しなら引数の値が、2 回目以降の呼び出しなら前回の値に加えて今回の値がバッファに書かれる。バッファはタイマにより定期的に読み出され、そこに書かれた値が一定時間毎に物理装置に表示される。ある系統のエラーが解除されるとエラーコードはバッファから消去される。全系統が解除 (=バッファが空) なら表示を消灯する。このようにアーキテクチャ導入の効果として各部の責務が分離され、実装やテストが容易になることを体験させるのが狙いである。

さらに教材開発時点では想定していなかったことであるが、品質特性として安全性について考察した際に不正操作を検出して排除する [⇒ 5 節 (4) 参照] という仕様追加が受講生の中から提案された。ただしこれを実現するためには、従来は特定の時点での値しか検出していなかった SW の値を一定時間 (例えば 1 秒間) の履歴として記録する必要が発生した。入力については層アーキテクチャになっていなかったため、その変更に時間を要し、結局一部が未完成のままとなってしまった。

(3) は品質特性としては主に信頼性や安全性が注目されがちであるが、他にも理解性や可用性など様々なものがあることを理解してもらうのが目的である。特に使いやすさ (ユーザビリティ) に関しては、仕様では特に言及していないが、受講生からこの品質特性を高めたいという要求が出た場合に備えて、計測や比較実験のための仕組みをあらかじめ用意した。例えば課題 SC においては、GUI 画面を設計するためにあらかじめ大きさなどの異なる数組の部品 (ボタン/スクロールバーなど) を用意し、画面遷移なども自由に設計できるようなフレームワーク (実際はソースコード) を与えることにした。

## 4. 実施 (21 年度)

平成 21 年 10 月より初版のシラバスに基づいて演習講義を実施した。初年度は実時間制御 (KN) と状態制御 (PB) の 2 課題 (2 チーム) で、プロセスの作成と品質特性を実現する方法に重点を置いて演習を行った。講義 (延べ 12 時間) は 4 名の教員が順番に行い、演習 (延べ 24 時間+補講 10 時間) は鈴木、矢竹の 2 名が中心となり毎時間、他の 2 名は適宜サポートを行った。他に開発環境の整備などの事前準備に 10 時間ほどを必要とした。

### 4.1 ソフトウェア開発演習の内容

ソフトウェア開発演習で実施した内容は以下の通りである。なおこれらの作業は後半の開発プロセス設計の素材とするため実際の活動に基づいて WBS(Work Breakdown Structure) を整理させ、入力プロダクト、出力プロダクト、計画時間、実際の所要時間を全て記録させた。なお (3)-(5) については両チームとも実際は逐次的ではなく開発単位ごとに同時並行に実施している。これは後半でプロセスを設計する際に問題となった。

**(1) 仕様書の理解**

本来は要求獲得に相当するフェーズであるが、今回は仕様書は既に与えられている。受講生には事前に機能仕様と品質仕様を配布し、自分達で作成したという想定で内容を理解させた。疑問点があったらまずチーム内で相談し、それでも不明な点はインストラクタに質問すべき内容としてまとめるように指示した。

**(2) 作業計画の作成**

開発対象の全体像を理解した時点で今後の作業手順を説明し、各作業にかかる時間の見積もりをさせた。しかし受講生の多くがはじめての経験であり基準となるデータを持っていないため、ここでは正確性より計画作成という行為の意味を理解させることに重点をおくことにした。すなわち計画の遵守を目指し、遅れが発生した場合はその回復方法をチーム内で検討する、解決できない場合はインストラクタに相談する、などの対策をとるように指示した。結果として両チームともテスト完了までに計画を約 10 時間超過したが、その分は講義時間外に作業時間を設けることで補った。

**(3) アーキテクチャ設計**

受講生にはアーキテクチャを決定する手がかりとして、機能仕様書にシステムのブロック図を与えてあるが、ここでは各部分の構成と振る舞いを UML のクラス図とコミュニケーション図を使用して記述させた。詳細は 5 節で述べる。

**(4) コンポーネント設計**

アーキテクチャを構成する要素の設計と実装で、成果物は UML 図である。ただしコンポーネントを C 言語の関数呼び出しのレベルまで詳細化したチームがあった。これも詳細は 5 節。

**(5) タイミング設計**

実時間に関する制約を理解させるのが目的である。特に成果物の形式は定めていなかったため、文章で記述した受講生が多かった。

ここまでの内容を中間発表で各チーム 30 分ずつ発表させた。

**(6) 実装とテスト**

コンポーネントを単位とする単体テストおよび一部の統合テストを実施させた。単体テスト自動化ツールとして CUnit を使用したが、一部のコンポーネントでテストが行えない実装になっていた点が問題となった。これも詳細は 5 節。

**(7) ドキュメンテーション**

開発の成果を報告書、およびプレゼンテーション資料として作成させた。報告書は成績評価用、プレゼンテーションは修正の後学外発表 (30 分) に使用した。

### 4.2 プロセス設計演習の内容

プロセス設計演習で実施した内容は以下の通りである。最初にウォーターフォール (以下 WF) モデルを示してフェーズの概念と次フェーズへの移行に必要な条件 (文書) を整理させ、次に V モデルを示し機能仕様とテスト仕様を同時に (同粒度で) 作成することの有用性を理解させることを目標とした。その後品質特性を実現するためにプロセスの改善を議論させた。結果として前半で行ったテストの一部をやり直すこととなり 4 時間程度超過したものの予定したテストを完了することができた。

**(1) 標準プロセス設計**

WBS に基づいて前半で行った作業の整理と標準プロセスへの再構成が目的である。開発演習で作成した作業計画と WBS に基づき、各チームが行った作業をアクティビティとして記述させた。次に各アクティビティをその入力/出力プロダクトの依存関係に基づいて WF モデルを用いて整理させた。その結果多くのプロダクトが欠落していることが判明した。次に V モデルを導入し、(外注したという想定で) 各種テスト (統合/システム/受け入れ/ユーザビリティ etc) のうちどのテストがどの時点で必要になるかを理解させ、テスト設計において不足しているプロダクトを認識させた。

**(2) 不足部分の作成**

前半では経験不足および時間の制約により品質仕様 (特に信頼性) の実現においては不十分な点が多かった。各自が作成した WF モデルおよび V モデルに基づいて、必要なプロダクトやアクティビティの追加を行い、プロセスを完成させた。

ここまでの内容を中間発表で各チーム 30 分ずつ発表させた。

**(3) 品質特性とその実現**

標準的な開発プロセスが設計できた上で、品質特性を実現するようにプロセスの改善を行うことを目的とする。品質特性は JIS X0129-1(信頼性/保守性/移植性/効率性/機能性/使用性) の中から各自 1 つ (チームで最低 2 つ) 任意に選択させた。

各チームごとに WF および V モデルに従い、品質特性を実現するために必要となるアクティビティを作成し、それに従って (単体/統合/システム) テストを設計/実施させた。特に信頼性の点で状態遷移図の見落としが多く、テストケースの網羅性が不十分であった。これらの問題を解決する先端技術としてモデル検査を紹介したが、21 年度は準備不足もあり受講生に実際にツールを使用させるまでには至らなかった。これは 22 年度への課題となった。

**(4) ドキュメンテーション**

開発の成果を報告書、およびプレゼンテーション資料として作成させた。プレゼンテーションは前半との相違点を明記するように指導した。特に自分達で開発した固有プロセスと品質特性の実現 (テスト設計と実施) に重点をおいて説明させた。

## 5. 評価と知見

### (1) アーキテクチャ/コンポーネント設計

ソフトウェアアーキテクチャという概念に関しては本演習講義で初めて知ったという受講生も多く、講義が 1 回 (90 分) のみということもあり理解は十分とはいえなかった。特にアーキテクチャでは物理的構造 (モジュールの包含関係) と論理的構造 (メッセージの受け渡し関係) の両方の記述が必要なことを受講生に理解させ、記述方法を模索することに時間を要した。

図 8 は課題 PB のチームが最初にアーキテクチャとして提出したもの (一部省略) である。物理構造は機能単位として (印刷抑制) ルール管理部、印刷制御部、ログ管理部および UI 部を把握しているが、論理構造 (関数呼び出し経路) を無視しこれらを単純に線で結んだものとなっている。

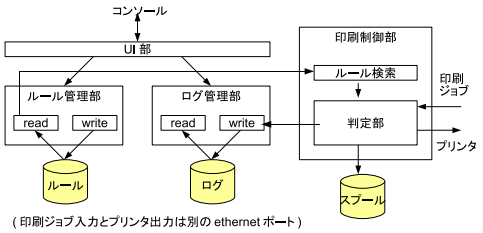


**図 8** アーキテクチャもどき

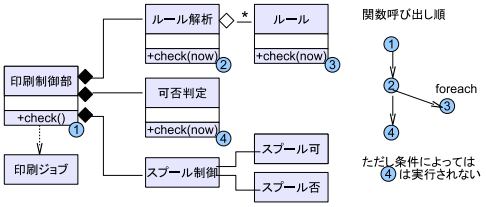


**図 9** 物理構造と論理構造のミスマッチ

物理構造と論理構造を明示できるように、クラス図とコミュニケーション図による記述を指導したが、それでも図 9 のようなミスマッチが発生した。課題作成側の意図としては、例えば抑制ルール (論理構造) とファイル (物理構造) を同一コンポーネント内で扱い、論理構造を変更すれば物理構造に反映されるという解答を想定していたが、これは受講生が時間に追われていて複数の解を比較検討できなかったのも要因と思われる。物理構造と論理構造の不一致は、保守性、再利用性を低下させることにつながるので、改めて両者を一致させることを意識付けしていく必要がある。なおアーキテクチャ理解度との関連性は不明であるが、実装の際に構造体を使用せずに単純型 (int と char*) 変数の組を多用する傾向もみられた。

実施後の受講生へのインタビューなどにより、今回の課題 (KN,PB) には、(1) 構造記述に継承を必要としない、(2) クラスの概念が実質不要 (=ほとんどがシングルトンオブジェクト) である、等の問題点があることが判明した。KN は実時間性がポイントなのであまり影響がなかったが、PB は印刷対象や抑制ルールをオブジェクトとしてそれらの状態変化を扱うシナリオとなっているので、受講生がクラスを発見できないと品質特性の議論が難しくなる。これは 22 年度で課題の改良を行うこととした。

### (2) テスト設計

図 10 は課題 KN のチームが最初に提出したアーキテクチャの一部である。問題は自動モード時の制御部にあり、入力の OFF あるいは異常事態 (電源電圧低下など) が発生するまで戻ってこないメソッドが存在する。

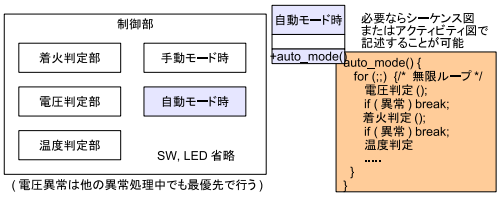


**図 10** テスト不可能なアーキテクチャ

KN の論理構造は単純な関数呼び出し/戻りのシーケンスとなり、それらの関数を単純に統合するだけでも実現は可能である (図 10 の大部分が該当)。しかしそのように設計されたアーキテクチャは例えばエラー検出部の各関数に対する責務の割当てが適切に行われていない。またセンサの値の監視を無限ループの内部で行っている (異常状態を検出するまでループを抜けない) ため、単体テスト (ホワイトボックステスト) を行う際に入出力の境界面を決定できないという問題が判明した。

そこで制御部の { 入力監視、変化による制御または異常検出 } を周期タスクとし、1 回の呼び出しは一定時間内に終了するような修正を指導した。これに監視対象となる入力の時間変化を外部から与えるためのスタブと、出力結果を比較するコードを追加してテストを行うことが可能になった。

スタブの使用例を図 11 に示す。OVERHEAT は 250 以上が 2 単位時間以上継続すると ON になる LED の

番号である。これは入力値 (温度センサ) が T_vec01 に示したように時間変化した場合をテストしている。スタブを分離するアーキテクチャを採用することで、入力ベクトルを変えることで様々な条件下でのテストが可能である。

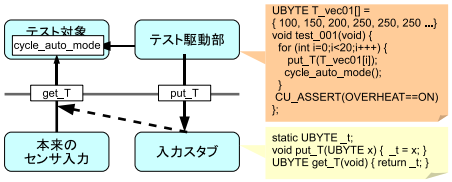


**図 11** スタブを利用した単体テストの例

### (3) プロセスの設計

図 12 は KN のチームが最初に作成した V モデルである。機能要求と品質要求は本来は要求分析の出力であるが、本演習講義では最初から与えられており、アーキテクチャ/コンポーネント/タイミング設計は実際には部分的に並行して実行されている点が特徴である。

標準プロセスに従えば要求分析後に基本設計書を作成しそれを元にテスト計画を作成すべきであるが、実際にはコード作成が完了したモジュールからボトムアップにより単体テストを実行していたため、行わなければならない作業全体のリストがないまま進行し進捗管理が事実上行えない状態であった。また実際のテストケースを作成する際にも発生条件と検出すべきエラーの一覧を整理したドキュメント (ここではエラー状況リストと呼称) を作成していなかったので、プロセスに追加させ、それらが必要となることを示した。

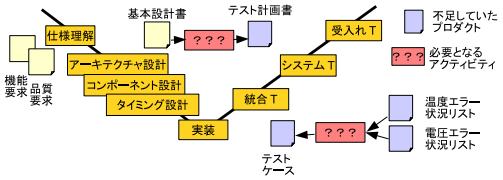


**図 12** プロダクトが不足していたプロセスの例

最初はプロセスの恩恵について疑問に思っていた者もいたが、レポートでは出力ドキュメント (ベースライン) が揃うまで次のフェーズに進んではいけないこと、プロセスに従うことで作成すべきドキュメントの不足が発見できること、各フェーズにおいて最後に必ずレビューを行う必要があることなどを理解し、一定の効果を確認できた。

**表 4** 高信頼ソフトウェア開発演習シラバス (改良版)

| | |
|---|---|
| 第 1 回 | 組込みシステム開発概論 |
| 第 2 回 | 開発環境, 演習課題説明 |
| 第 3 回 | アーキテクチャ記述言語説明 [新規] |
| 第 4-7 回 | アーキテクチャ/コンポーネント設計 [演習] |
| 第 8 回 | 中間発表 |
| 第 9 回 | 実装/テスト技法概論 [新規] |
| 第 10-13 回 | 実装/テスト実施 [演習] |
| 第 14 回 | ドキュメント作成 |
| 第 15 回 | 最終発表 (成績評価) |
| 第 16 回 | 学外発表 (企業による評価) |

### (4) 品質特性のプロセスへの反映

図 13 は KN チームの改善後のプロセスの一部である。品質特性としては信頼性 (安全性) を採用した。これはガスを扱うために万一にも誤動作をしてはいけないという課題の特性に沿ったものである。

要求分析フェーズをチームで議論/再検討した結果、入力 SW の On/Off に関して全ての組み合わせが機能仕様に示されていないこと、操作の時間要素は機能仕様、品質仕様のどちらにも書かれていないことが検出された。そこで不正な操作 (1 分間に 10 回以上の On/Off など) を検出した場合はエラーとすることを品質仕様に、これらの安全性要件のレビューをプロセスのベースラインに追加させた。このときに SW 監視部が On/Off 回数を計数するには、アーキテクチャをどのように改善するべきかの議論を通じて保守性の実現に関する理解を深める結果となった。

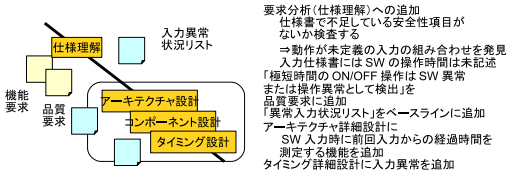


**図 13** 安全性を作りこんだプロセスの例

## 6. 実施 (22 年度)

21 年度の反省を踏まえ、22 年度は表 4, 5 のようにシラバスを修正した。受講生は 4 チームで残りの 3 課題である並列装置制御 (DA), 機器制御 (EX), 資源有効利用 (SC) および状態制御 (PB を改良) を実施した。

### 6.1 ソフトウェア開発演習:改善点と評価

21 年度はアーキテクチャ記述には UML クラス図とコミュニケーション図を使用していたが、(1) 論理構造と物理構造の対応関係を十分に記述できない、(2) 実行単位の並列性や階層関係を記述できない、(3) 時間制約を明示的に記述できないなどの点で不十分であった。

**表 5** 高信頼ソフトウェアプロセス設計演習シラバス (改良版)

| | |
|---|---|
| 第 1 回 | プロセスモデル概論 (WF,V) |
| 第 2-4 回 | WF モデル設計 [演習] |
| 第 5-7 回 | V モデル設計 [演習] |
| 第 8 回 | 中間発表 |
| 第 9 回 | 品質特性と品質保証技術 [新規] |
| Extra | コードレビュー [新規] |
| 第 10-13 回 | 品質特性の実現 [演習] |
| 第 14 回 | ドキュメント作成 |
| 第 15 回 | 最終発表 (成績評価) |
| 第 16 回 | 学外発表 (企業による評価) |

この問題に対処するため、22 年度からはアーキテクチャ記述言語 (Architecture Analysis & Design Language)[3] を導入することとした。AADL はプログラムの階層関係 (スレッドグループ)、プログラム/データの包含関係 (ポート) などを記述できる上、限定的ながら時間制約も明示でき、組み込みシステムの記述言語として有用である。実際に開発環境が eclipse プラグインとして提供されており、組み込みシステム開発において幅広く使用されている。講義では仕様書 (英文約 120 ページ) を日本語 30 ページ程度に要約したものを作成、講義 1 回分を費やして記法と意味を説明、受講生の理解を目指した。また簡単な例題 (生産者/消費者) を用いて開発環境の使用方法を実演した。

改善された PB 課題の AADL によるアーキテクチャ記述を図 14 に示す (斜体は説明のため追加)。21 年版の課題に比べて継承を導入したこと、および実行効率向上のため印刷ジョブを受け取る部分が拡張されているが、それを 3 つの受信スレッドと 1 つの送信スレッドとして設計したこと、各スレッドで操作可能なデータのスコープなどが容易に理解できる。ルールファイルの扱いの理解も容易かつ保守性に優れたアーキテクチャといえる。

このように AADL を用いることにより、図 9 に見られた物理構造と論理構造の不一致の問題を改善することができた。なおツールにはメッセージフロー毎に制約が満たされているか検出する機能もあるが、準備不足などもあり今回は時間制約の記述および検証は十分に行うことができなかった。これは 23 年度への課題である。

他に開発演習中にテストの体系的な説明を実施するように修正した。このため文献[4] などを参考に、テスト技法 (ホワイトボックス/ブラックボックス) とテスト内容 (単体/統合/システム) 毎の適用例 (約 100 ページ) を約 30 ページに要約した資料を作成し、テストを設計する際に必要な手法/対象をこれより選択させた。

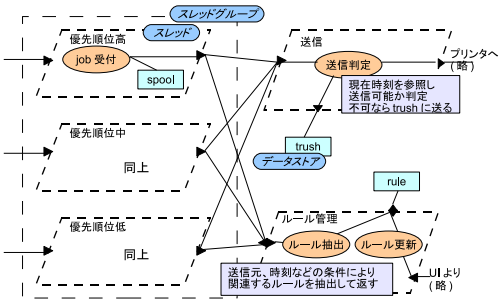


**図 14** AADL によるアーキテクチャ記述例 (PB 改)

### 6.2 プロセス設計演習:改善点と評価

プロセス設計においてはとにかくレビューを徹底的に行わせるため、各フェーズの出力プロダクトにレビュー結果を追加させることにした。さらに品質特性 (信頼性/保守性) の説明の後に、自分が作成した部分のソースコードを全員の前で行毎に説明させて質問を受ける作業 (コードレビュー) を土曜日まる 1 日かけて全員分実施した。これはテストを開始する前に不具合のあるコードを発見し、単体テストが効率よく行えるようにリファクタリングを行うのが目的である。

発見された問題点の一部を表 6 に示す。問題点の中には組込みシステムに限らないものもあるが、やはり状態遷移や時間制約の実現に苦労していたことがうかがえる。

**表 6** コードレビューで検出された問題点

| | |
|---|---|
| 共通 | 関数が長すぎる (50 行以上) |
| 共通 | 同じコードが複数箇所に存在 (コードクローン) |
| PB | if 文 (状態遷移実現) の入れ子が深すぎる (7 段) |
| PB | 共有資源のアクセス方法 (タイミング) 誤り |
| DA | 入力値の範囲検査が欠落している |
| DA | 入力文字列解析と異常値警告が同一関数内 |
| EX | 状態遷移図の内容を正しく反映していない (状態遷移図に非決定性が残ったまま) |
| EX | 時間によって変化する入力値をテストできない |
| SC | 状態変数が多すぎる (複数の関数内に存在) |
| SC | テスト条件を表す変数が不足している |

ソフトウェア検証技術とモデル検査については、品質保証技術の中で基本となる状態モデルおよび簡単な例題でツール (SPIN) の使用方法の説明のみを行い、使用を受講生に任せたところ、結果として 1 チーム (EX) が部分的に使用した。EX は製品画像の判定結果をもとに機械の制御を様々に切り替える必要があり、状態遷移図で記述するのに適していたためと推測される。

## 7. 全体評価

演習講義としての成績評価は受講生が提出した最終レポートおよび成果物の完成度に基づいて行っている。しかしながら母数が 15 と少なく、また授業評価に直結するとあって成功事例だけに絞って記述する傾向があり、これらのレポート結果のみから統計処理を行っても有意な結果は得られないものと判断される。ここではカリキュラムの評価として、インストラクタが指導中に発見した問題点、受講生の最終発表での発表内容 (感想含む) およびそれに対する企業の方々のコメントに基づいて述べる。

### 7.1 PBL 形式について

多くの企業内で実施している OJT との相違点に関する質問があったが、PBL の本質は受講生らに自主的に問題を解決させることであり、インストラクタは解決のヒントのみを与える。一方で OJT では業務の一環として行うため、プロジェクトとしての完成度が求められる点が最大の違いである。本 PBL では成果に関して 100%を求めるのではなく、次に類似の問題に直面した際に学習した知識と経験 (アーキテクチャ、プロセスなど) を応用する能力の習得を目指している。

ただし成果物に関しては、企業の方の前で発表する手前ある程度の完成度が必要なのも事実である。特に 21 年度は受講生の (テストや状態モデルに対する) 前提知識を把握していなかったこともあり、受講生の知識不足が発覚すると補講により補うという場当たり的な対処になり、これが作業時間の大幅な超過や完成度の低下の要因となったことは反省すべき点である。

### 7.2 留学生を対象とする点について

受講生は留学生ではあるが、日常会話および業務に必要な日本語 (ビジネス日本語) については、別カリキュラムで習得しているため、指導に関して日本人となんら変わる点はなかった。

しかしながらソフトウェア開発で使用される専門用語に関する理解は十分とはいえない側面もあった。典型的な例としては、KN,PB などで単体テストで=の有無に起因する誤りが何箇所か発見された。受講生に尋ねたところ、原因は日本語の「以上/以下」(境界値を含む) と英語の greater/less than(境界値を含まない) の違いに関する理解が不十分なためであった。これは日本語教育部門と協力して、用語集を作成することで解決できると考えられる。

### 7.3 実問題への対応について

本演習講義の目的のひとつは実際の開発で発生した問題を受講生に体験させることである。3.2 節に示したように、課題の作成者も想定していなかった問題を受講生自身が発見した事例もあり、この目的は達成されたといえる。

しかしながら、課題に仕組まれた問題点のいくつかは発見はできたものの十分に解決方法が議論できなかった。例えば KN の同時エラー発生については、最初の開発演習では見逃している。プロセス設計演習で発見できたものの、アーキテクチャに不具合があり、テストが未完成であった。アーキテクチャ作成時にテストを考慮するように指導することが必要である。

### 7.4 品質の作り込みについて

受講生は最初はプロセスの知識がなく、試行錯誤を繰り返し開発を進めるケースが多かった。プロセスの導入および品質の作り込みにより問題点を早期に回避できることの発見は受講生にとって大きな進歩となったといえる。

ただし企業側から品質特性は多種多様であり、信頼性のようにプロセス/アーキテクチャの修正により直接実現可能な特性と、移植性や使用性などプロセスだけでは実現できない特性が存在しているという指摘があった。これは今後検討すべき課題である。

## 8. おわりに

3 年間に渡り 5 課題 15 名に対して実施してきた PBL であるが、留学生および企業の双方から一定の評価を頂いたことでこれまでの試みは成功したものと確信している。品質の実現方法など明らかになった点を改善して 23 年度以降も継続して実施する予定である。

## 謝　辞

最初に開始時のプロジェクトメンバであり、全体方針決定に御尽力を頂いた早稲田大学教授　岸知二氏に深く感謝いたします。また企業と共同での教材作成作業にあたりご支援を頂いた吉田 博幸氏、新海 卓夫氏に厚く御礼申し上げます。

## 参考文献

1) 「高信頼組込みシステム開発技術に関わる基盤的人材育成プログラム」ホームページ
http://www.jaist.ac.jp/asia_jinzai/
2) 「組込みシステム概論」
情報処理学会 組込みシステム研究会監修. CQ 出版社 2008.
3) Architecture Analysis & Design Language
http://www.aadl.info/aadl/currentsite/
4) 「ソフトウェアテストの技法 第 2 版」
G. J. Myers, 長尾、松尾訳. 近代科学社 2007.