コミュニケーションスキルを重視したソフトウェア技術者教育の試行

齋藤 祐一郎

2013 年 2 月

修士 (経営システム科学) 論文

指導教員

主指導教員 久野 靖 教授
副指導教員 佐野 享子 准教授
副指導教員 中谷 多哉子 准教授

筑波大学 大学院 ビジネス科学研究科
経営システム科学専攻

概要

企業活動としてのソフトウェア開発を遂行するには、コミュニケーションが欠かせない。一方、コミュニケーションが苦手なソフトウェア技術者 (以下、技術者) が存在し、特に新人技術者に顕著である。本研究は、新人技術者となってゆく大学 4 年生の学生 (以下、学生) を対象に、ソフトウェア開発に必要なコミュニケーションスキルを獲得すべく、教育方法を考案・試行し、効果を測定する事を目的とする。

教育の試行にあたり、教材を作成し実験を行った。

教育の指針として、技術者に求められるコミュニケーションスキルの定義を「ドキュメント・ソースコード等の成果物を基にチームメンバー同士がインタラクションを行うスキル」と定めた。その上で、教材は Katz が示した管理者に必要な 3 つのスキル「ヒューマンスキル」「テクニカルスキル」および「コンセプチュアルスキル」を軸に作成した。教育方法は、「ケースメソッド」と「ロールプレイング」を組み合わせた。

実験にあたり、2 大学 6 名の被験者を集めた。実験期間はそれぞれ 7 日であり、1〜6 日目は被験者が各自の環境で、7 日目は筑波大学 東京キャンパスにてワークショップ形式で実施する。実験のデータは、「アンケート」「SYMLOG」「ビデオ」を用いて分析を行う。作成したケースは「Twitter の mash-up サイトを作ろう」である。

第 1 回実験は、7 日目のワークショップ時、コミュニケーションが希薄な状況であった。各人が「このように実装しているはずだ」という、希望的観測に基づいた作業進行が行われていた。また、実験後に各メンバーが「より密にコミュニケーションを行うべきだった」とアンケートに残している。

第 2 回実験は、被験者に対し事前にコミュニケーションに関する事前指導を実施した。その中で、被験者は自分自身の理解と相手の理解は違う事を確認し合った。7 日目のワークショップ時、第 1 回と比較して常時緊密なコミュニケーションが図られ、チーム内で成果物に対する齟齬が最小限に抑えられていた。しかし、テクニカルスキルが不足しており、プログラミングが適切に進行しない問題が発生した。

2 回の実験を通じて、事前指導とケースメソッド＋ロールプレイング形式の学習によって、コミュニケーションスキルを高められる事を確認した。しかし、テクニカルスキル・ヒューマンスキル両方を満たす実験結果を得る事はできなかった。ここから、講義や研究を通じて各個人で獲得したテクニカルスキル、最低でもプログラミングに関するスキルがあってこそ、本研究で用いた教育の効果が高まると考えられる。

Software engineer education curriculum with enhancement of communication skills

SAITO, Yuichiro
Graduate School of Business Sciences
University of Tsukuba, Tokyo, Japan

Abstract

Communications are indispensable in professional software development activities. However, some software engineers have been weak communication, that especially pronounced in freshman. Therefore, training curriculum for freshmen software engineers which enhances communication skills are needed. The objective of this research is development and evaluation of such curriculum.

The definition of communication skills, "Skills as can interact with team members based on artifacts, which is documents and source code.", Based on the definition, the teaching method was designed, with three important skills presented by Katz, namely technical skills, human skills and conceptual skills in mind. The method is the combination of role playing and case method. Task for the case is "Let's make a mash-up 'Twitter'".

For evaluation purpose, experimental training courses were conducted with six students from two universities. They were organized as two teams with three persons each, and single round of the training course was held for each of the teams. One round consists of seven days; for the first six days the activities were held in them local environment, and on the final day an workshop was held at the authors' site. The data gathered in the form of questionnaire, SYMLOG evaluation and video recording.

In the first experiment, at final day, the communication situation was tenuous. Progress of workshop on the basis of each person "should have implemented this way", and wishful thinking has been done. Also, they had written in the questionnaire that they should have communicated more tightly.

In the second experiment, the course were preceded by a prior guidance in which they could experience the importance and difficulty of communication. They have recognized that their thoughts tend to differ significantly. In this experiment, tighter communication were observed in the workshop, and conflicts among their artifacts could be kept small; the difference from the first experiment was apparent. Nevertheless, technical skills are insufficient, the problem does not proceed properly programming has occurred.

In the neither of two experiments, participants could not incarnate both of the technical skills and human (communication) skills sufficiently. We conclude that technical skills including programming skills (through traditional curriculum) are important anyway, and when the participants have enough of those skills, our case method and role playing courses will be effective.

目次

第 1 章　研究の背景と目的 9
1.1　はじめに 9
1.2　ソフトウェア開発におけるコミュニケーション 10
1.2.1　開発プロセスとコミュニケーション 10
1.2.2　コミュニケーションが苦手な技術者の存在 12
1.3　学生のコミュニケーションスキルを高める教育の試行 12
1.4　まとめと本論文の構成 13

第 2 章　関連研究 15
2.1　はじめに 15
2.2　ソフトウェア開発プロセス 15
2.2.1　ソフトウェア開発プロセスの位置づけ 15
2.2.2　ウォーターフォール型開発モデル 16
2.2.3　アジャイル型開発モデル 17
2.2.4　Contextual Design 18
2.2.5　ソフトウェア開発プロセスのまとめ 18
2.3　日本におけるソフトウェア開発とコミュニケーションの関係 19
2.3.1　日本におけるソフトウェア開発とコミュニケーションの関係の位置づけ 19
2.3.2　情報システム・ソフトウェア取引トラブル事例集 20
2.3.3　管理者に求められる 3 つのコアスキル 20
2.3.4　高業績の技術者が持つスキルの調査結果 22
2.3.5　日本におけるソフトウェア開発とコミュニケーションの関係のまとめ 22
2.4　技術者のスキル指標 22
2.4.1　技術者のスキル指標の位置づけ 22
2.4.2　ITSS 23
2.4.3　CMMI 23

2.4.4 ISO/IEC15504 23
2.4.5 教育水準の指標のまとめ 24
2.5 日本における高等教育カリキュラム 24
2.5.1 日本における高等教育カリキュラムの位置づけ 24
2.5.2 J07 24
2.5.3 高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム 25
2.5.4 汎用形教育コンテンツ 26
2.5.5 日本における高等教育カリキュラムのまとめ 26
2.6 集団構造を分析する指標 26
2.6.1 集団構造を分析する指標の位置づけ 26
2.6.2 個人のチームワーク能力を測定する尺度 27
2.6.3 Interaction process analysis 27
2.6.4 SYMLOG 27
2.6.5 集団構造を分析する指標のまとめ 29
2.7 まとめ 29

第 3 章 教材の作成指針 31
3.1 はじめに 31
3.2 技術者に求められるコミュニケーションスキルの定義 32
3.3 獲得を目指すスキルの定義 32
3.4 技術者が関わるステークホルダーの範囲 34
3.5 スキル養成指標の調査と選択 35
3.6 教材の提供方法の検討 35
3.7 チーム構成の検討 38
3.8 まとめ 38

第 4 章 実験実施計画の作成 39
4.1 はじめに 39
4.2 養成するスキルの設定 40
4.3 実験工程と評価基準点の設定 41
4.4 評価のための準備 41
4.4.1 評価の観点 41
4.4.2 事前アンケートの作成 42
4.4.3 在宅での実験時に提出するレポートフォーマット作成 42
4.4.4 ワークショップの評価指標の作成 43
4.4.5 最終アンケートの作成 43
4.5 被験者の募集 43

4.6 ケースの作成 43
4.6.1 ケース作成の方針 43
4.6.2 ケースの概要 44
4.6.3 評価方法 45
4.7 成果物の定義 45
4.8 倫理的配慮 47
4.9 まとめ 47

第 5 章 第 1 回 実験 49
5.1 はじめに 49
5.2 実験実施概要 49
5.3 被験者のプロフィール 50
5.4 1 回目〜3 回目 納品物の確認 51
5.5 7 日目 ワークショップの状況 51
5.5.1 全体の流れ 51
5.6 実験結果 52
5.6.1 ソフトウェアのスクリーンショット 52
5.6.2 アンケート結果 52
5.6.3 SYMLOG を用いた小集団構造把握 53
5.6.4 会話内容 54
5.6.5 被験者毎の行動 57
5.7 まとめ 58

第 6 章 第 2 回 実験 61
6.1 はじめに 61
6.2 実験実施概要 61
6.3 被験者のプロフィール 62
6.4 事前指導 63
6.4.1 事前指導方法の検討 63
6.4.2 事前指導時の考慮点 63
6.4.3 事前指導の状況 64
6.5 1 回目〜3 回目 納品物の確認 64
6.6 7 日目 ワークショップの状況 65
6.6.1 全体の流れ 65
6.7 実験結果 65
6.7.1 ソフトウェアのスクリーンショット 65
6.7.2 アンケート結果 66

6.7.3 SYMLOG を用いた小集団構造把握 66
6.7.4 会話内容 70
6.7.5 被験者毎の行動 72
6.8 まとめ 72

第 7 章 考察 75
7.1 はじめに 75
7.2 第 1 回実験 — α 大学チーム 75
7.2.1 ヒューマンスキル 75
7.2.2 テクニカルスキル 76
7.2.3 コンセプチュアルスキル 76
7.2.4 全般 77
7.3 第 2 回実験 — β 大学チーム 77
7.3.1 ヒューマンスキル 77
7.3.2 テクニカルスキル 78
7.3.3 コンセプチュアルスキル 79
7.3.4 全般 79
7.4 第 1 回と第 2 回実験の比較 80
7.5 まとめ 81

第 8 章 結論 83

参考文献 89

付録 A 被験者 募集要項 93

付録 B 事前アンケート 書式 97

付録 C レポートフォーマット 103

付録 D SYMLOG 用 評価シート 109

付録 E 最終アンケート 書式 111

付録 F ケース — Twitter の mashup サイトを作ろう 117

付録 G 事前指導 資料 127

第 1 章

研究の背景と目的

1.1 はじめに

企業活動としてのソフトウェア開発を遂行するには、コミュニケーションが欠かせない。また、企業においてソフトウェア開発はチームで行うものである。そして、チームで開発を遂行するためには、メンバー毎に割り当てられた役割をもとにチームメンバーとコミュニケーションを図る必要がある。

一方、コミュニケーションが苦手なソフトウェア技術者 (以下、技術者) が存在している [1] [2]。コミュニケーションが苦手な技術者は、ソフトウェア開発にはコミュニケーションが必要であるのにも関わらず、積極的なコミュニケーションをとろうとしない。そのため、コミュニケーションが苦手な技術者とチームメンバーとの間で理解の齟齬等が発生し、ひいてはソフトウェア開発が滞る要因となっている。

また、企業が新卒採用を行う際、コミュニケーションスキルに注目している調査結果がある。それを示す資料として、一般社団法人日本経済団体連合会が全業種の企業会員のうち 1,274 社に対して行った調査 [3] によると、「コミュニケーション能力」は新卒採用時に人事担当者が最も注目する能力であり、その割合は調査企業の 8 割に上っている (図 1.1)。ただし、本調査は全業種を対象とした調査であるため、ソフトウェア技術者のコミュニケーションスキルと同じものであるか、特定はできない。

本研究では、高等教育機関でソフトウェア開発の教育を受けている学生に対し、新人技術者となるにあたって必要となる技術とともにコミュニケーションスキルを獲得すべく、教育方法を考案・試行し、効果を測定する事を目的とする。

次節以降では、ソフトウェア開発において技術者に求められるコミュニケーションの要素や、コミュニケーションが苦手な技術者の実態について述べる。

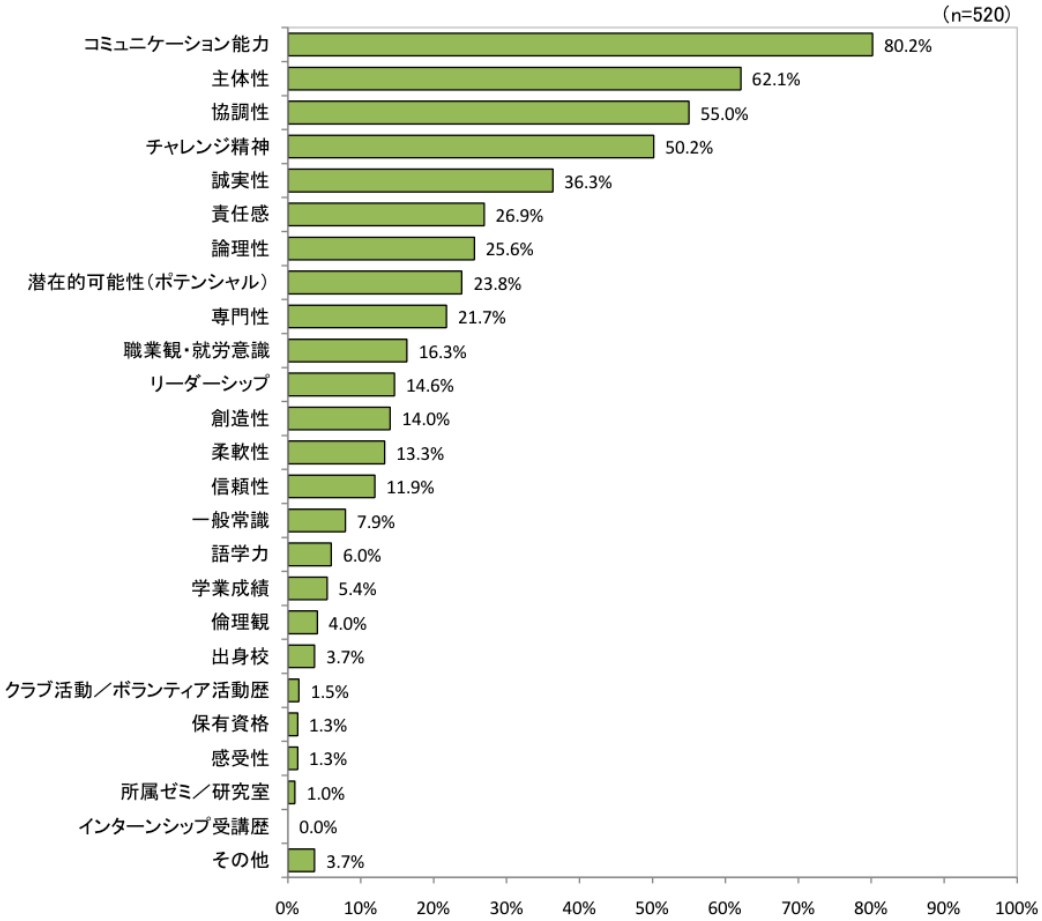


図 1.1　新卒 (2011 年 3 月卒業) 採用時に選考にあたって特に重視した点 (1 企業 5 つ選択) (出典: [3])

1.2　ソフトウェア開発におけるコミュニケーション

1.2.1　開発プロセスとコミュニケーション

ソフトウェア開発では、ほぼ全てにおいて複数人で、即ちチームを組んで、作業を進める。山口 [4] は、チームには一般に次の 4 つの条件が存在すると述べている。

1. 達成すべき問題が存在する
2. メンバー同士が互いに依存し合う
3. 果たすべき役割が割り振られている
4. チームメンバーとそれ以外の線引きが明確である

上記を、ソフトウェア開発の実情に沿って、次の通りに解釈する。
条件 1 は、ソフトウェア開発を通じて 1 つの製品を完成させることにあたる。

条件 2 は、同職種の技術者を集め開発規模によって業務を分担を行う事や、他職種、例えばプログラマとテスタに分かれ、それぞれの専門分野を通してチームに貢献する事に相当する。

条件 3 は、タスク管理者がゴールから逆算した Work Breakdown Structure(以下、WBS) を定義する。即ち、1 つの大きな成果物に対し、小さな成果物に分割しタスクを定義することにあたる。その際、2 について述べた分担毎に適切な業務を割り当てることとなる。

条件 4 は、顧客や技術者などステークホルダーがソフトウェア開発に携わる中で、特に技術者がソフトウェア開発のチームメンバーとして携わることである。それ以外の人物は、ソフトウェア開発の目的にそって、チームメンバーに依存する事もなければ、役割が割り当てられる事もない。

以上より、ソフトウェア開発において、チームの活動に際し様々な形でのコミュニケーションが必要であると言える。

Wood [5] は、コミュニケーションは一般に次の 4 つの側面を持つと述べている。

1. 動的なプロセスである
2. 体系的である
3. 2 つのレベル — 情報量と深さがある
4. シンボルとインタラクションを通じて生まれる

上記の 4 つの側面をもとに、ソフトウェア開発におけるコミュニケーションに当てはめ、具体的に解釈する。

1 については、コミュニケーションは始まりと終わりが明確に定義されていない、常に進行しているプロセスである。

2 については、コミュニケーションが行われる時は、ただ単にその状況だけを見ても理解する事はできず、誰が何を伝えようとしたのかという体系が意味づけるものである。コミュニケーションが成立するには、理解するための背景をあわせる事が必要である。言い換えれば、チーム内での役割が違えば、同じ内容の情報を受け取っても違う形で理解される可能性がある。

3 については、情報量と、深さがコミュニケーションの意味を変えてくるものである。深さとは、人物の間にある力関係に代表される背景を指すものである。

4 については、シンボルを用いたインタラクションが、人がコミュニケーションの内容の理解を深めるとある。従って、相手に理解されるとき、全く同じ言葉だったとしても、これまでの経緯・言葉の抑揚・その時の感情が理解に影響を与える。

以上から、ソフトウェア開発におけるコミュニケーションで重要なことは、コミュニケーションは自分自身と相手の理解の違いが存在する事が前提となっている。その上で、チームで開発を遂行するにあたって、プロジェクト運営が円滑に進むよう、様々なソフトウェア開発プロセスが考案され、現場で利用されている。詳細は 2.2 節で述べる。

1.2.2　コミュニケーションが苦手な技術者の存在

前節では、ソフトウェア開発はチームで行い、その中でコミュニケーションが欠かせない事を述べた。一方で、コミュニケーションを苦手とする技術者が存在している。Cusumano [1] は、組織運営で問題が発生しているソフトウェア開発企業のアセスメントを行った際、コミュニケーションに関して次の問題点を指摘している。

- グループ間の意思疎通・フィードバック・学習プロセスが貧弱。
- 適切ではないコミュニケーションが、勤労モラルの低下とともに、高いストレスや過労を引き起こした。

上記とは別に、新人ソフトウェア開発者の弱みとして、「コミュニケーション」「認知」そして「既存の環境に対する適応」が挙げられている。これは、Begel [2] による、入社 1〜7 ヶ月目の新人技術者に対して行った教育に関する調査を通じて観察された。Begel の実験は、新人技術者がどのような問題に遭遇し、大学までの教育でどのような問題を解決しなければならないかを調査する目的で実施されている。

その中で、「コミュニケーション」について、新人技術者は速やかにかつ適切な情報をまとめて質問がすることができなかった。これは、いつ・どのように・他人に質問するかを理解しきれなかったためである。無駄な調査が重なったり、逆に必要な情報を調査せずに質問に出向いてしまうという行動が観察されている。調査において、新人技術者は、知識にある人を見つけて順に質問する者と、孤立した状態で資料を読み漁る事に終始した者とに分かれていると述べられている。

以上より、全ての技術者ではないにしても、コミュニケーションを得意としない技術者の存在が、ソフトウェア開発を円滑に進めるにあたって問題となる場合がある事がわかった。特に、新人技術者はこれまで通っていた大学等でコミュニケーションに関する教育をあまり受けていないか、あるいは受けていても教育の効果が充分もたらされていない事がわかる。

1.3　学生のコミュニケーションスキルを高める教育の試行

前節まで、ソフトウェア開発にはコミュニケーションが求められる事、及びコミュニケーションスキルが不十分な新人技術者の存在について、述べた。そこで、大学を初めとした高等教育機関でソフトウェアに関する教育を受けた大学 4 年生 (以下、被験者) を対象とし、ソフトウェア開発に必要となるコミュニケーションに関する教育を試行 (以下、本研究) し、効果を測定することを本研究の目的とする。

試行するにあたり、次の 3 点を重視する。

1. チームでソフトウェア開発を行う経験を積む

2. ソフトウェア開発におけるコミュニケーションスキルを高める
3. 教育期間を 1 週間程度とする

1 については、前節の研究において、高等教育機関で学んだ学生はチームでソフトウェア開発を行う経験を持っていない者がいる。一方、業務で取り組むソフトウェア開発のほとんどはチームで実施される。そこで、一人でプログラムを開発する時と、チームでソフトウェアを開発する時とで、どのような違いがあるかを知る事が出来るよう、教育を行う。その上で、新人技術者としてより早く現場で活躍できることを目標とする。

2 については、ソフトウェア開発で必要となるコミュニケーションスキルとは何であり、そして高等教育機関に通った学生にコミュニケーションに関する教育をどのように施せば良いかを明確にしていく。

3 については、被験者の研究や他の講義に出来るだけ影響を与えないためである。

1.4　まとめと本論文の構成

本章では、ソフトウェア開発におけるコミュニケーションの課題と新人技術を述べた。

続いて、ソフトウェア開発におけるチームとコミュニケーションの関係を述べた。まず、チームが何であるかを確認し、続いてコミュニケーションとは何であるかを確認した。コミュニケーションは単に会話をする事ではなく、人間同士のインタラクションを行う際の背景やシンボルの存在が重要である事を確認した。

また、コミュニケーションが苦手な技術者、特に新人技術者の存在について確認した。チームでソフトウェア開発を行うにはコミュニケーションが欠かせない一方、新人技術者はコミュニケーションを苦手としている。

そこで、本研究では高等教育機関でソフトウェア開発の教育を受けている学生に対し、新人技術者となるにあたって必要となる技術とともにコミュニケーションスキルを獲得すべく、教育方法を考案・試行し、効果を測定する事を目的とすることを述べた。

以下、2 章ではソフトウェア開発の現状や教育カリキュラムについて調査した内容をまとめる。3 章では教材の作成指針について説明する。4 章では、実験実施計画および作成した教材について述べる。第 5 章及び 6 章では、先章で作成した教材をもとに実施した試行実験を内容と得られた知見について説明する。第 7 章では、実験結果を踏まえ考察を行う。第 8 章では、本論文を総括し、結論を記す。

第 2 章

関連研究

2.1 はじめに

1 章では、新入社員にコミュニケーションスキルを高める教育の必要性と、その背景を述べた。本章では、コミュニケーションスキルを求められる背景をより深く調査するため、次の 5 点について関連研究の調査を行う。

1. ソフトウェア開発プロセス
2. 日本におけるソフトウェア開発とコミュニケーションの関係
3. 技術者のスキル指標
4. 日本における高等教育カリキュラムの事例
5. 集団構造の分析

1 については、2.2 節において、過去から現代に至るまでに用いられてきた代表的なソフトウェア開発プロセスについて調査し、各モデルで誰とどのようなコミュニケーションが図られているかを確認する。2 つについては、2.3 節において、日本で行われてきた調査を通じ、ソフトウェア開発でコミュニケーションスキルとは何かを明らかにする。3 については、2.4 節において、技術者のスキルの水準を示す指標と、その中で新人技術者に求められる水準を明らかにしていく。4 については、2.5 節において、日本の高等教育機関にて技術者教育を施す際に、どのようなカリキュラムや手法が用いられるかを調査する。5 については、2.6 節において、集団行動の分析を行うための指標について調査する。

2.2 ソフトウェア開発プロセス

2.2.1 ソフトウェア開発プロセスの位置づけ

ソフトウェア開発プロセスとは、製品を生み出す工程を指す。目的は、要求をもとにソフトウェアを適切に実装し製品とするために、手順・管理モデル・ツール・環境および文書体系を標準化し、広く開発に適用できるようにするものである。

大森ら [6] の調査結果を確認すると、2001〜2012 年の間に少なくとも 100 以上のソフトウェア開発プロセスに関する研究がある。そのため、全てを取り上げる事が難しい。本研究ではソフトウェア開発プロセスにおいて代表的なライフサイクルモデルに絞り、取り上げる。

2.2.2　ウォーターフォール型開発モデル

ウォーターフォール型開発モデル [7] [8] は、1970 年に Royce によってまとめられたソフトウェア開発プロセスである。2 つの特徴があり、1 つ目の特徴は工程間に明確な区切りを置くと共に、その間は形式化された文書で受け渡す (図 2.1)。2 つ目の特徴は、工程間の手戻りを極力なくすように作られている。要求という抽象度の高い内容を、プログラムというより具体的な内容に落とし込むため、ウォーターフォールと言われる。また、テストは最も具体的な部分であるプログラムのテストから始まり、最後は最も抽象度が高い要求を満たしているかを確認する流れとなっている。

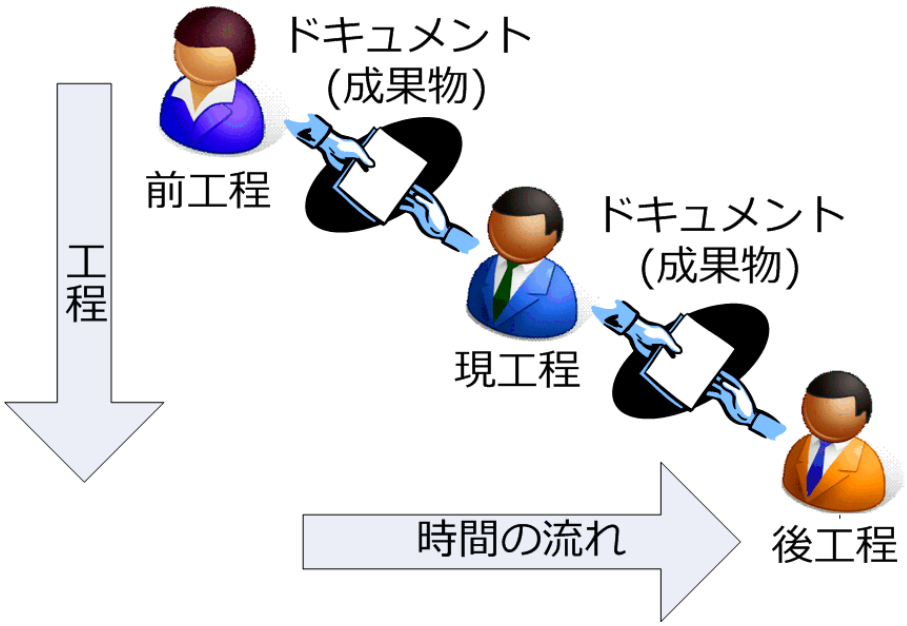


図 2.1　ウォーターフォール型開発モデルの流れ

工程の間を結ぶ段階で、ドキュメントを成果物とし、かつコンテキストとしたコミュニケーションが発生する (図 2.1)。前工程の技術者がまとめたドキュメントをもとに、現工程の技術者が作業を実施する。そして、現工程の担当者の作成した成果物をもとに、後工程の担当者が作業を実施する。そのため、前工程の担当者の作成した成果物に誤りがあると、現工程はもちろん、後工程で問題が連鎖的に発生する。

従って、工程間で成果物をやり取りする時に、前工程の技術者が作成した成果物にバグや情報の不備が存在していないかを現工程の技術者が確認する事と、現工程の技術者が成果物の内容を充分に理解しているか前工程の技術者が確認する時に、コミュニケーション

が必要となる。

本研究では、コミュニケーションがソフトウェア開発プロセスで行われる一例として、ウォーターフォール型開発モデルを取り上げる。

2.2.3　アジャイル型開発モデル

アジャイル型開発モデル [9] は、1〜4 週間の区切りをもとに開発を実施し、それを、反復することでソフトウェアを完成に導く手法である。Koch [10] によると、顧客・開発者混成の開発プロジェクトチーム内で、積極的にコミュニケーションを図ることが特徴である。

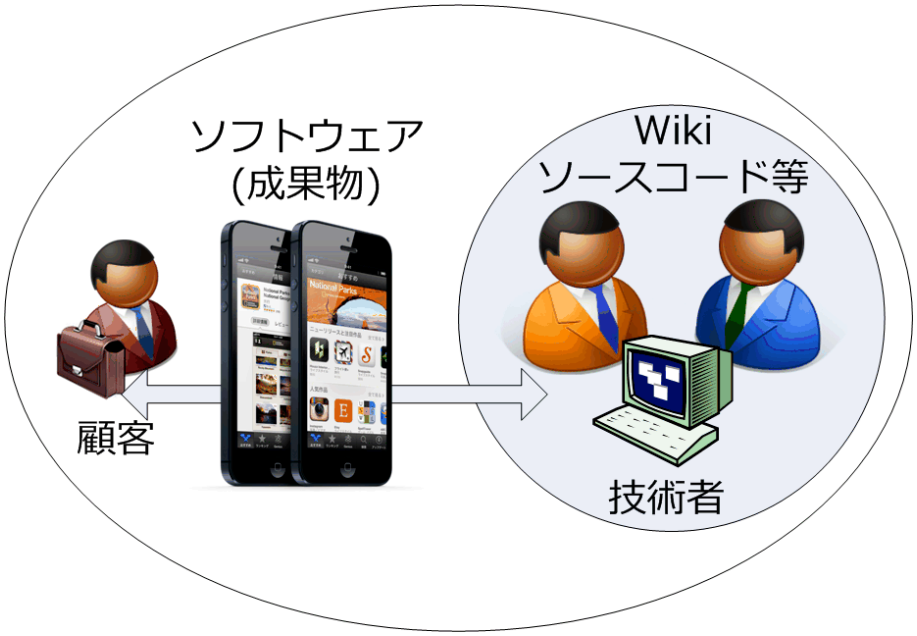


図 2.2　アジャイル型開発モデルの流れ

アジャイル宣言 [9] によれば、アジャイル型開発モデルでは大きく分けて 2 つのコミュニケーションが求められる (図 2.2)。

1. 技術者内
2. 顧客と技術者

1 については、「プロセスやツールよりも個人と対話」と「包括的なドキュメントよりも動くソフトウェア」が求められている中で、いかにコミュニケーションを図るかを考慮しなければならない。その中で、主にプログラムのソースコードをコミュニケーションのコンテキストとして用いている事が特徴である。Beck [11] によると、「テスト駆動開発」「ペアプログラミング」および「リファクタリング」を行う事を薦めている。テスト駆動開発とは、開発したプログラムが正しく動作しているか、テストを行うプログラムをセッ

トで開発し、反復してテストを実施する事である。ペアプログラミングとは、2 人 1 組で作業を進める事で、コードが正しく記述され、作業が問題なく進捗しているかを相互確認する手法である。「リファクタリング」とは、作成したプログラムの構造を整理する事で、プログラムに記述された仕様の文脈を理解を助けるものである。また、自然言語を基にした情報を共有する「Wiki」を通じて、プログラムだけでは表現できない情報を共有する事もある。

2 については、特に「包括的なドキュメントよりも動くソフトウェア」と「契約交渉よりも顧客との協調」を解決しなければならない。その中で、主に完成したソフトウェアをベースにコミュニケーションのコンテキストとして用いられている。なお、開発初期は、顧客と要求をすりあわせるために、Rasmusson [12] は「ペルソナ」「ペーパープロトタイプ」を用いることを薦めている。ペルソナ [13] とは、利用が想定される人物像を詳細に定義し、その人物の要求を満たせるようソフトウェアを設計する手法である。「ペーパープロトタイプ」は、実際にプログラムを開発する前に画面レイアウト等の青写真を描き、それを基に顧客と技術者がコミュニケーションを行ってゆく。また、開発するソフトウェアは 1 週間前後でレビューを実施し、仕上がりと要求の齟齬を出来るだけ発生させないようにする点が特徴である。

本研究では、コミュニケーションがソフトウェア開発プロセスで行われる一例として、アジャイル型開発モデルをウォーターフォール型開発モデルとともに取り上げる。

2.2.4　Contextual Design

Contextual Design [14](以下、CD) とは、ソフトウェアを利用するユーザの利用イメージを「文脈」として捉え定義する、プロセス中心の開発である。CD のプロセスは、次の 9 段階からなる。

CD に参加するメンバーは「UI デザイナ」「エンジニア」「ドキュメントライター」「ユーザビリティの専門家」「内部のビジネスユーザ」「マーケティング担当者」と「ビジネスアナリスト」と、開発者側・顧客側ともに多岐に渡る (図 2.3)。

CD では顧客組織の各業務担当と技術者集団が一体のチームとなり、ワークショップ形式で要求を明確化して行き、ソフトウェア開発の目的を確固たるものにしている。

本研究では、コミュニケーションがソフトウェア開発プロセスで行われる一例として、先に挙げたウォーターフォール型開発モデルとアジャイル型開発モデルとともに取り上げる。

2.2.5　ソフトウェア開発プロセスのまとめ

ウォーターフォール型開発モデルとアジャイル型開発モデル、そして CD は工程の進め方こそ異なるもの、ウォーターフォール型開発モデルと CD は主にドキュメント、後者は

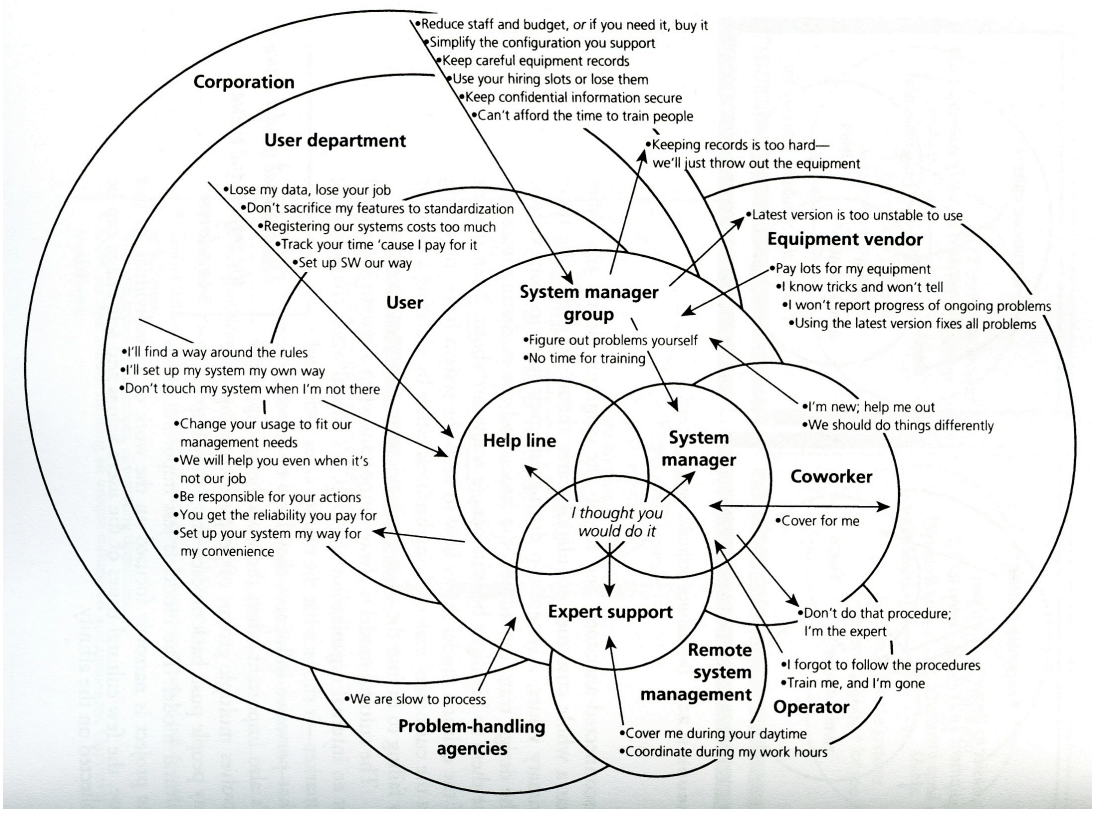


図 2.3 Contexual Design によるコミュニケーションの流れ (出典: [14])

主にソースコードをもとに、技術者同士がコミュニケーションをはかる。媒体の形こそ違うもの、ソフトウェア開発には成果物を通じたコミュニケーションがなくてはならないものであることがわかる。

また、技術者と顧客についても、ソフトウェア開発モデル毎に媒体こそ違うものの、成果物を通じてコミュニケーションを図っていることがわかる。

2.3 日本におけるソフトウェア開発とコミュニケーションの関係

2.3.1 日本におけるソフトウェア開発とコミュニケーションの関係の位置づけ

日本において、報道等でソフトウェア開発の失敗事例を目にするのは珍しい事ではない。その中で、技術的問題はさることながら、顧客との間の理解の齟齬を初めとしたコミュニケーションに起因する問題も存在する。

本節では、技術者がどのようなスキルを求められているのか、特にコミュニケーションスキルに焦点を当て、基準となるモデルや調査結果についてまとめた。

2.3.2　情報システム・ソフトウェア取引トラブル事例集

「情報システム・ソフトウェア取引トラブル事例集 [15]」は、2010 年に経済産業省が調査を実施し、まとめた資料である。主に契約の問題について取り上げられたものであるが、その中で開発のコミュニケーションが問題となっているものが、4 章 9〜13 節の内容の不明確に起因するもの、4 章 18 節,19 節で述べられている。

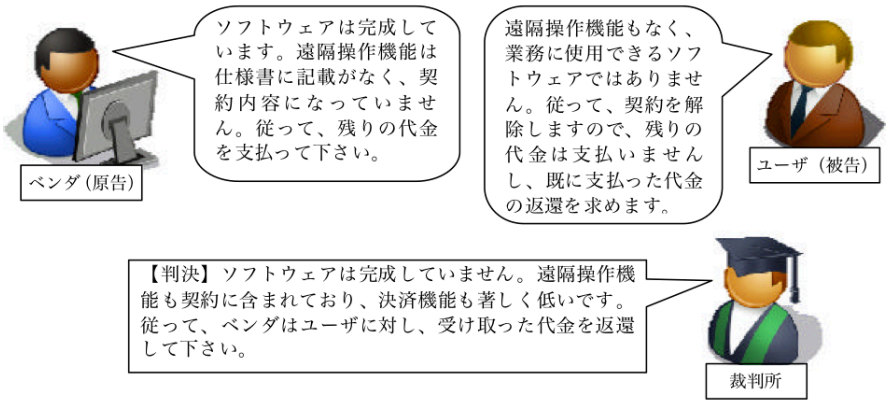


図 2.4　13. 業務範囲・完成基準が曖昧なためにトラブルになった事例
(出典: [15])

図 2.4 は、4.13 節で述べられている「業務範囲・完成基準が曖昧なためにトラブルになった事例」の概要である。争点は二つあり、曖昧な要求仕様策定が引き起こした双方の理解の齟齬が起きている事、もう一つは検収条件の不備により開発完了の理解が双方で食い違っており代金の支払いが発生するかの有無が問われている。尚、判決では、要求仕様を要件定義段階で明確にすべきであった事、また顧客側が検収対応を曖昧に行ったことが問題となり、開発会社側が敗訴した。

法的な側面は技術者のみで解決する事は難しい。しかし、図 2.4 の事例で挙がっている問題の一つである要求仕様策定が曖昧であった点は、技術者と顧客の間で解決できる問題である。また、事例に技術的問題で紛糾している事例が希少である。このことから、コミュニケーションスキルがソフトウェア開発の成否の左右する要因となっている事を確認できる。

本研究では、本事例をコミュニケーションの問題がソフトウェア開発でトラブルを引き起こすことがあることを確認するに留める。

2.3.3　管理者に求められる 3 つのコアスキル

「管理者に求められる 3 つのコアスキル [16]」とは、1974 年に Katz が提唱した、管理者に対して求められるスキルをまとめたモデルである。本モデルは、以下の 3 つに分類さ

れている。

- テクニカルスキル
- ヒューマンスキル
- コンセプチュアルスキル

テクニカルスキルは、業務遂行に関わるスキルである。ソフトウェア開発では、システムの設計やプログラミングのスキルにあたる。特に、技術者にとってはなくてはならないものである。

ヒューマンスキルは、対人活動を円滑に進めるスキルである。ネゴシエーションスキルや、コミュニケーションスキルが該当する。

コンセプチュアルスキルは、全体像を把握し新しい体系を構築できるスキルである。ソフトウェア開発は、既存業務の改善や全く新しいビジネス展開の核となる業務である事が少なくない。技術者としてエキスパートを目指すには不可欠のスキルである。

3 つのコアスキルは、職位によって相対的に求められる量が違う。その内訳を図 2.5 に示す。テクニカルスキルは低い職位ほど多く求められ、逆にコンセプチュアルスキルは経営者といった高い職位でより多く求められる。その中で、ヒューマンスキルは、どの職位にあっても等しく求められるスキルである。

本研究では、教育を実施する際の軸となるスキルとして、本モデルを取り上げる。

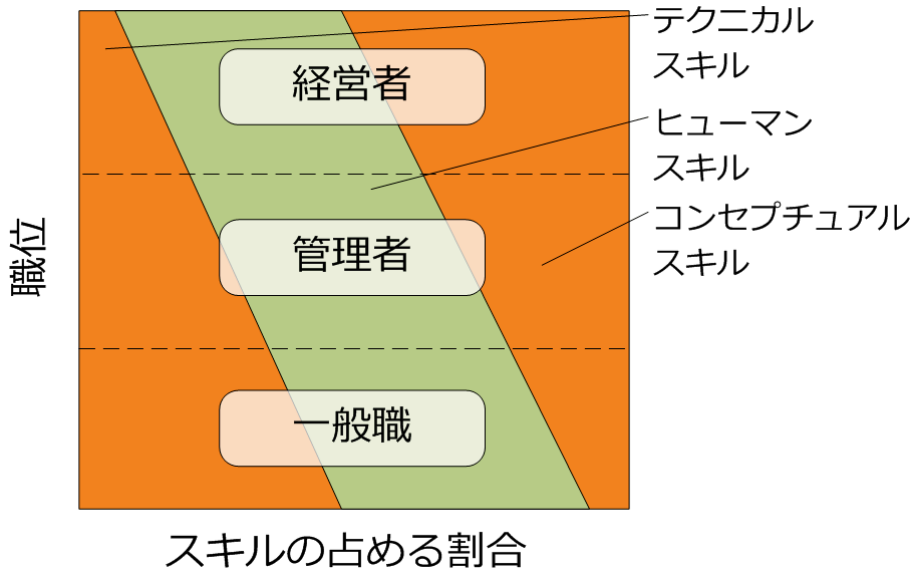


図 2.5　管理者に必要な 3 つのスキル ([17] を参考に作成)

2.3.4　高業績の技術者が持つスキルの調査結果

「IT 技術者のスキルに関する調査 [18]」は (以下、IT スキル調査)、財団法人 日本情報処理開発協会が行った調査をまとめた資料である。これは、独立行政法人 情報処理推進機構 (以下、IPA) が作成した IT スキル標準 [19](以下、ITSS) をもとに日本の技術者の現状を把握しつつ、ITSS の妥当性を評価すること、また高度なスキルを獲得するまでのプロセスを解明しようとするものである。

熟達した高業績の技術者 (以下、高業績者) が重視している事項として「リーダーシップスキル」が挙がっている。また、年代別に分析した結果、20 代では特にコミュニケーションスキルが重視される度合いが高い。30 代以降ではリーダーシップスキルやネゴシエーションスキルが重視される事を考えると、コミュニケーションスキルの土台があってこそのリーダーシップスキルである事がわかる。

また、高業績者では、専門的スキルと同時に社会的スキルに要求される技術の要求水準も高いことが特徴であると述べられている。

本研究では、コミュニケーションスキルがソフトウェア開発にて重要である事を理解するための事例として取り上げる。

2.3.5　日本におけるソフトウェア開発とコミュニケーションの関係のまとめ

まず、技術者に限らず、スキルは 3 つのコアスキルを基に養成する事が求められる事がわかった。その中で、コミュニケーションスキルはいかなる職位であっても求められるものである事がわかった。

また、ソフトウェア開発において、高業績の技術者の多くは、コミュニケーションスキルが高い事が示されている。ソフトウェア開発でより高い価値を提供する人材を育成するには、コミュニケーションスキルを高める事が有用である事を確認できた。

2.4　技術者のスキル指標

2.4.1　技術者のスキル指標の位置づけ

教育水準として用いる技術者のスキル指標を検討する。ソフトウェア開発を実施するにあたって、技術力をどのように測るかはしばし議論になる。それは、技術力と一口に言っても、定性的なものだからである。

そこで、指標として現在公開されている開発標準モデルやスキル標準を基に考慮する事とした。選択肢として、引地の調査 [20] をもとに 3 つ取り上げる。

2.4.2 ITSS

ITSS [19] は IPA が作成したもので、企業内においてどのような人材戦略を構築するか、という問題点に対しての解決策を示す指標の一つである。効率的な人材育成を行い、かつ社内外の人材育成水準の共通の指標 [21] として用いる事が出来るよう設計されている。

ITSS には職種と、それに対応するレベルが設定されている (表 2.1)。

職種	マーケティング			セールス			コンサルタント		ITアーキテクト				プロジェクトマネジメント				ITスペシャリスト								アプリケーションスペシャリスト		ソフトウェアデベロップメント			カスタマサービス			ITサービスマネジメント				エデュケーション	
専門分野	マーケティングマネジメント	販売チャネル戦略	マーケットコミュニケーション	訪問型ソリューティングセールス	訪問型製品セールス	メディア利用型セールス	インダストリ	ビジネスファンクション	アプリケーションアーキテクチャ	インフラストラクチャアーキテクチャ	インテグレーションアーキテクチャ	システム開発	アウトソーシング	ネットワークサービス	ソフトウェア製品開発	プラットフォーム	ネットワーク	データベース	アプリケーション共通基盤	システム管理	セキュリティ	業務システム	業務パッケージ	基本ソフト	ミドルソフト	応用ソフト	ハードウェア	ソフトウェア	ファシリティマネジメント	運用管理	システム管理	オペレーション	サービスデスク	研修企画	インストラクション
レベル7																																				
レベル6																																				
レベル5																																				
レベル4																																				
レベル3																																				
レベル2																																				
レベル1																																				

表 2.1 ITSS レベルと対応する職種 (出典: [19])

レベルには、創造する価値水準を規定するため、達成すべきテクニカルスキルが定義されている。

本研究では、教育を通じて獲得するスキルの指標として利用する。

2.4.3 CMMI

CMMI [22] は、カーネギーメロン大学 ソフトウェア工学研究所にて作成された。ソフトウェア開発プロセスを管理・改善するための参照モデルである。ソフトウェア開発を行う組織が、どの程度の開発プロセスの標準化を推進しているかを示している。主に、組織外に自社のソフトウェア開発力を評価される際に用いられる。

本研究では、個々人のスキルを測る指標としては不向きであるため、用いない。

2.4.4 ISO/IEC15504

ISO/IEC15504 [23] は、国際標準化機構 (ISO) によって提起された。ソフトウェア全般の品質を測定するた めの尺度を定義する国際規格である。機能性、信頼性、使用性、効

率性、保守性、そして移植性といった 6 つの品質特性をもとに、開発するソフトウェアがどの程度の品質を担保しているかを示すものである。

本研究では、個々人のスキルを測る指標としては不向きであるため、用いない。

2.4.5　教育水準の指標のまとめ

技術者の技術力を示す指標は、主に個人の習熟度を規定する指標と、組織としての技術力を示す指標の 2 種類が存在する事がわかった。ここから、ITSS を用いて個人の技術力を定量的に測定する事が可能である事がわかった。

しかし、技術者のコミュニケーションスキルの水準を測定するための指標はこの中にはなく、本研究ではそのまま適用する事が難しい事を確認した。

2.5　日本における高等教育カリキュラム

2.5.1　日本における高等教育カリキュラムの位置づけ

本節では、日本における情報・計算機科学に関する高等教育のカリキュラムについて調査を行う。新入社員が企業に就職するにあたって、どのような事柄を学んでいるのかを知る。その上で、コミュニケーションスキルを高める教育を行っているか、行われている場合はどのような教育が施されているかを評価する。

2.5.2　J07

J07 [24] は、一般社団法人 情報処理学会によって作成された。高等教育機関における情報処理教育のカリキュラム設計のリファレンスとして用いられる事が目的である。作成にあたり、Association for Computing Machinery(以下、ACM) で作成された CC2005 [25] をベースに国際的な整合性に配慮しつつ、最低限習得すべき知識の内容と深さについての目標を示すものである (図 2.6)。

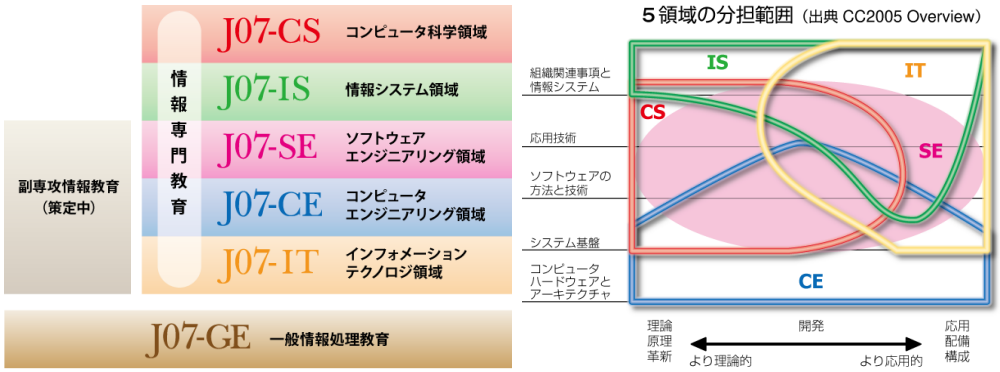


図 2.6　J07 カリキュラムのモデル (出典: [24])

J07 は、カリキュラムの領域を CS(コンピュータ科学)，IS(情報システム)，SE(ソフトウェアエンジニアリング)，CE(コンピュータエンジニアリング)，IT(インフォメーション) の五つをあげていることが特徴である。その上で、それぞれ領域についてカリキュラム標準を示している。ただし、具体的な指導方法は、各教育機関・担当者に委ねられている。また、技術者に必要なコミュニケーションスキルについては言及されていない。

以上より、J07 はテクニカルスキルに関する教育内容は参考となるものの、コミュニケーションスキルに関する教育内容を満足しているとは言えず、本研究を通じてコミュニケーションに関する教育を考慮して行く必要がある。

2.5.3 高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム

高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム [26] は、筑波大学の博士前期課程のカリキュラムであり、産学協同で実施されている (図 2.7)。組み込み系やエンタープライズ (大規模業務システム) の設計・開発の人材を育成するため、Project Based Learning(以下、PBL) を中心に、実践的な教育カリキュラムを組んでいる事が特徴である。

PBL は開発工程を通してソフトウェア開発に必要な技術・知識を学び、その中でコミュニケーションスキルを獲得する事になると思われる。しかし、技術者に必要なコミュニケーションスキルに焦点をあてて設計されているものではない。

本研究では、コミュニケーションスキルを高めることに重点を置く事を考慮すると、教育内容を満足しているとは言えない。

2.5.4 汎用形教育コンテンツ

汎用形教育コンテンツ [27] は、高等教育で用いるための教育カリキュラムのひな形を提供するものである。論理的思考力やプロジェクトベースの開発を通じ、実践的な教育を行えるようになっている。大きく分けて 3 つのカリキュラムに分かれている。

- プロジェクト型システム開発チーム演習教育コンテンツ
- パーソナルスキル（ロジカルシンキング）養成教育コンテンツ
- ソフトウェア開発技法実践的演習教育コンテンツ

いずれも、現実の開発に即したスキルを獲得するために設計されている。ただし、どのコンテンツも 15 コマ、大学の講義期間に換算すると 1 学期〜半期分の期間を要する。そのため、教育にあたっては充分な時間を用意することが求められる。

本研究では、1 週間程度の教育を行う事を想定しているため、教育期間を満足しているとは言えず、そのまま適用する事が難しい。

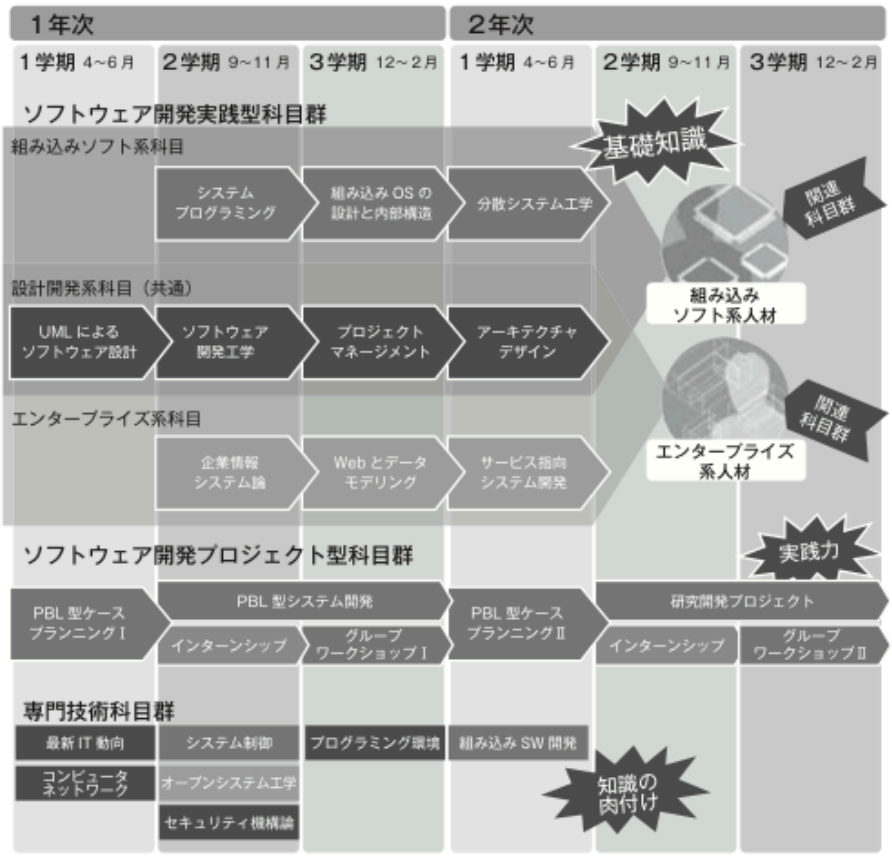


図 2.7　高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム カリキュラム
(出典: [26])

2.5.5　日本における高等教育カリキュラムのまとめ

日本における高等教育は、主にソフトウェア開発プロセスの全般を通してテクニカルスキルを向上させる目的で作成されている。設計技法の習熟やプログラム開発能力は技術者にとって欠く事が出来ない能力である。しかし、新人技術者に不足しているコミュニケーションスキルを解決することを明確に設定したカリキュラムではない。

2.6　集団構造を分析する指標

2.6.1　集団構造を分析する指標の位置づけ

集団構造を分析する指標は、定性的になりがちな人物のコンピテンシーを定量的に測定するための指標である。本研究で採用を検討した指標を 3 つピックアップし、それぞれの指標の特徴を述べる。

2.6.2 個人のチームワーク能力を測定する尺度

個人のチームワーク能力を測定する尺度 [28] は、相川らによって開発された、チームメンバー個々人のチームワークの能力を測定し統計的に分析するための尺度である。特徴は、特定のチームに所属した時に発揮するチームワークの能力ではなく、コンピテンシーとしてのチームワークの能力を測定できることである。

尺度として、5 つ能力「コミュニケーション」「チーム志向」「バックアップ」「モニタリング」そして「リーダーシップ」が設定されている。「コミュニケーション」は、他のメンバーの意思を適切に解読できる能力である。「チーム志向」は、他のメンバーとの対立を避け、チームの目標に向かうために調和を重視する能力である。「バックアップ」は、他のメンバーを励ます情緒的サポートや、助力を行う能力である。「モニタリング」は、チームが置かれている状況を観察し、状況に応じて対応する能力である。「リーダーシップ」は、メンバー同士の相互作用を促し、チームの目標を達するために行動できる能力である。

本研究では、統計的に有為な水準になる数の被験者を集める事が難しい。従って、本研究での採用は見送る。

2.6.3 Interaction process analysis

Interaction process analysis [29] は、Bales によって提案された。会話を中心とし、小グループの間である課題の解決や意思決定の際になされる相互行為の進行過程を分析を行う手法である。

Bales は、相互作用の過程は行為・言語・象徴・反応および動作等の継続的な流れであると述べている。また、流れはグループ全体に及ぶ事もあればそうでない事もあり、滑らかに流れる事もあれば分裂する事もあると述べている。その上で、図 2.8 の相互作用を解釈するためのカテゴリが用意されている。

しかし、本手法は集団の関係性を考察することができない問題がある。作業工程の中で会話を全て文字に起こしてから分析を行う等、作業工数が多くなる点も難点である。従って、本研究では用いる事が難しい。

2.6.4 SYMLOG

SYMLOG [31] は、Bales によって提案された。Interaction process analysis と同様、会話を中心とし、小グループの間の行動観察を行う手法である。また、Interaction process analysis で問題となった集団の関係性を分析できない弱点を克服している。

対人関係を、3 次元「支配的〜服従的」「友好的〜非友好的」および「課題志向的〜感情表出的」で表現する。これによって、観察者による対人行動の逐次的に観察が記録できると同時に、相互作業の過程が完了したあとに、観察者が全体的印象に基づき各個人の対人

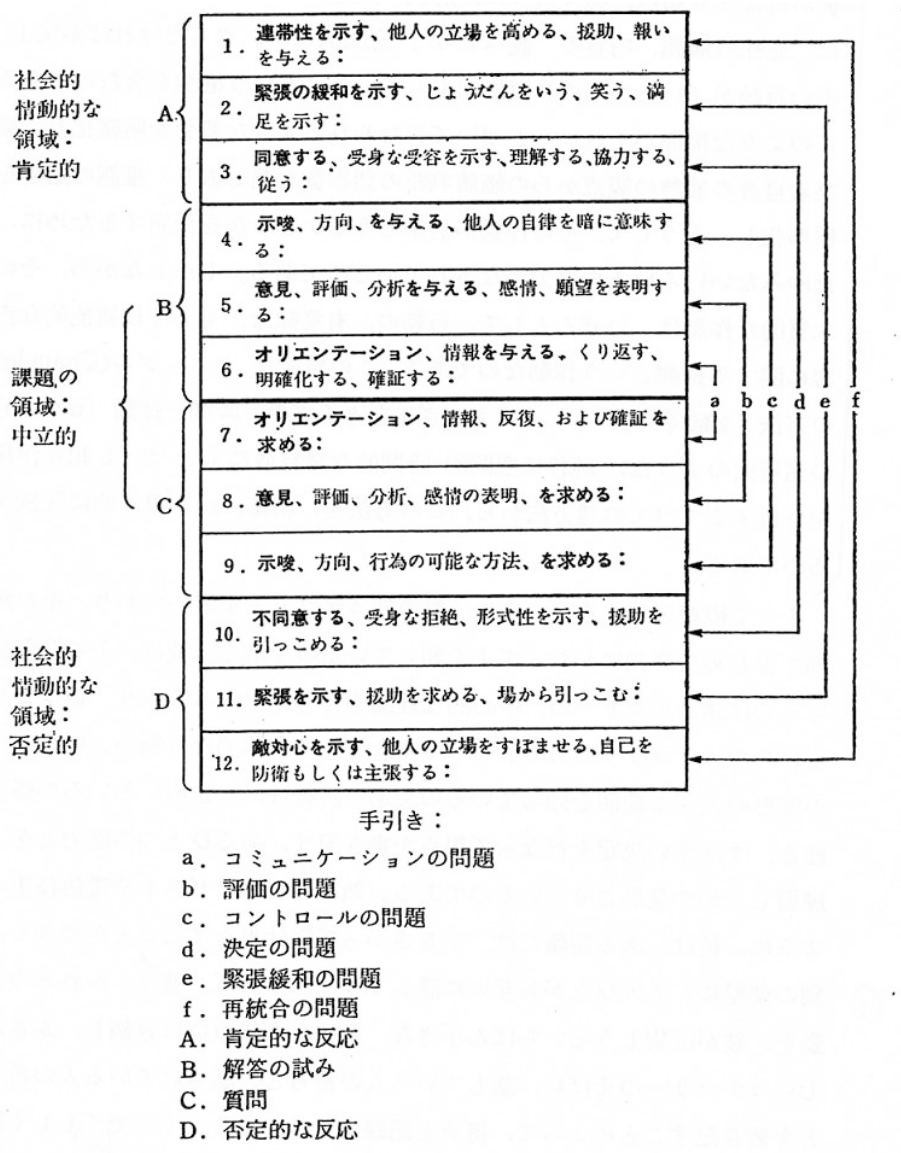


図 2.8　カテゴリとその間の主要な関係 (出典: [30])

的行動を評価する事ができる。

また、奥田・伊藤は、日本語版 [32] として改良版を出している。これは、日本と米国の文化差を吸収し、かつオリジナル版の複合的カテゴリを除いている。具体的には、「課題志向的〜感情表出的」となっている次元を「向文脈的〜脱文脈的」と解釈され直され、かつ評価項目が 26 項目から 12 項目へ整理されている事が特徴である。

SYMLOG は、本研究において個々人のソフトウェア開発のコミュニケーションスキルを定量的に図る事ができ、有用である。

2.6.5 集団構造を分析する指標のまとめ

集団構造を分析する指標として、主に集団を統計的に分析する方法と、個々人の行動を得点化する方法を取り上げた。

今回は、3 章以降で述べるが、集まった被験者が 6 名であり統計的に処理するには少ない。そのため、個々人の行動を得点化して指標化する SYMLOG の日本語改良版を用いる事で分析を進めて行く。

2.7 まとめ

本章では、関連研究として、ソフトウェア開発を取り巻く環境、および教育カリキュラムについて調査した。

2.2 節では、ソフトウェア開発プロセスについて、代表的なものを取り上げた。こから、ソフトウェア開発はどのような開発モデルを用いても、成果物を媒体とするコミュニケーションをとる必要がある事がわかった。

2.3 節では、高業績の技術者はコミュニケーションスキルが高い事が示されている。そのため、コミュニケーションスキルを高める事が技術者にとって必要な事を確認した。

2.4 節では、技術者のスキルを示す指標が各種ある事が判明した。特に、ITSS は個人のスキル獲得状況を示す指標として有用である事がわかった。一方、コミュニケーションスキルの指標がないことを確認した。

2.5 節では、日本の高等教育機関において開発・運用されている教育カリキュラムについて述べた。これらを学ぶ事が出来るのは一部の学生であり、かつ長い期間の教育が必要である。従って、技術者に求められるコミュニケーションスキルを学ぶ機会をより増やしていくために、別の手法を講じる必要があると考える。

2.6 節では、集団行動を分析する手法は、集団全体を統計的に分析する方法と、個々人をそれぞれ分析する手法の大きく 2 つがある事がわかった。

これらの議論から、現在の技術者の多くはコミュニケーションスキルをより獲得する必要がある事、そしてより短期かつ実践的な教育を作成し導入する必要があるという結論を得た。技術者にテクニカルスキルは必須であるが、それだけではソフトウェア開発を滞りなく進める事は難しい。

そこで、本章の議論を基に、1 章の要求を満たすべく、より短期間かつ実践的なソフトウェア開発の教材を作成し、試行して行く。次章では、この新しい教材の作成と評価方法について説明する。

第 3 章

教材の作成指針

3.1 はじめに

1 章では、開発チームを組織すると必ずコミュニケーションが発生する事を示した。その上で、本研究では高等教育機関で学ぶ学生を対象に、ソフトウェア開発に必要なコミュニケーションスキルを獲得できる教育の実施と効果測定を目的とする事を述べた。

本章では、コミュニケーションスキルの向上を重視した技術者教育手法を開発するための指針を検討する。その要点を次の 6 つにまとめた。

- 技術者に求められるコミュニケーションスキルの定義
- 獲得を目指すスキルの定義
- 技術者が関わるステークホルダーの範囲
- スキル養成指標の調査と選択
- 教材の提供方法の検討
- チーム構成の検討

3.2 節において、技術者に求められるコミュニケーションスキルとは何かを定義する。3.3 節において、教育を通じて獲得するスキルの目標を立てる。過去の研究を基に、必要となるスキルの軸を定め、その上で技術者に必要なコミュニケーションスキルとは何かを明確にする。3.4 節において、技術者が関わるステークホルダーの範囲を確認する。他者と関わるにあたって、その範囲と種類を特定するものである。3.5 節において、スキル養成指標の調査と選択を実施する。3.6 節において、技術者を教育する際に用いられる教材の提供方法を調査し、本研究で採用する方法を調査する。3.7 節にて、教育時のチーム構成について検討を行う。

3.2　技術者に求められるコミュニケーションスキルの定義

1 章で述べた、Wood [5] によるコミュニケーションの構成「動的なプロセス」「体系的」「情報の量と深さ」そして「シンボルとインタラクションを通じて生まれる」事を基に、技術者に求められるコミュニケーションを次の通りに定義する。

ドキュメント・ソースコード等の成果物を基に
チームメンバー同士がインタラクションを行うスキル

ウォーターフォール型開発モデル [7] を例に取り、説明する。まず、「動的なプロセス」とは、ソフトウェア開発プロセスそのものである。ソフトウェア開発のプロセスの中にはフェーズ (工程) は存在するが、それらは独立して存在する訳ではなく、プロセスの中に存在し、それぞれがつながりを持っている。次に、「体系的」は、ソフトウェア開発中に交わされる成果物は前段階のフェーズで作成された成果物を基に作成される。さらに「情報の量と深さ」は、成果物の量、及び言及されている情報の深度を示す。最後に「シンボルとインタラクションを通じて生まれる」は、シンボルとなる成果物を通じ、常にチーム内で協議・検討され、合意をとることである。

以上は、2.2 節で挙げた、どのソフトウェア開発プロセスにも適用可能なものとし、その上でコミュニケーションスキルを獲得するために必要な要素を考慮していくものとする。

3.3　獲得を目指すスキルの定義

ソフトウェア開発をより円滑に進められるようにするために、技術者が獲得すべきスキルセットとは何であろうか。本研究では 2.3.3 節で示した 3 つのコアスキル、本研究が対象とする新人技術者が獲得すべきスキルを定義する。

1. ステークホルダーの役割分担を理解し業務に結びつけている
2. 自らのプレゼンスがステークホルダーに理解され業務で活かされている
3. 人に頼る部分・自ら解決する部分を切り分けながらコミュニケーションを行える
4. カリキュラム内に織り込まれている要素技術を理解し製品開発に活かしている
5. システムのデザインを行う事が出来る力が育まれる

1〜3 はヒューマンスキル、4 はテクニカルスキル、5 はコンセプチュアルスキルが該当する。なお、ヒューマンスキルは、3.2 節で挙げたコミュニケーションの定義をもとに記述する。

1 については、自らが関与するステークホルダーについて、役割分担や意思決定者を理解する。開発プロジェクトチームという組織行動をする上で、自分自身以外にどのような役割を持った人が所属しており、組織が結合して運営されているのかを理解する。

2 については、自らが関与するステークホルダーから、自分自身の役割分担について理解を深めてもらう。組織である以上、当初からリーダーとなる人物から役割が与えられる。しかし、例えば単に「プログラマ」と一口に言っても、経験 1 年程度の初歩の技量の事もあれば、何十年の経験の積み重ねの上で簡潔かつ高速なアルゴリズムを書く事が出来るのとでは、そのレベルは全く違うものである。

3 については、自分自身に与えられた業務範囲の中で、どのような貢献が出来るのかを考える力を身につけ、さらにステークホルダーに伝える力を身につけることである。これにより、ステークホルダーに対してプレゼンスを高めることができ、チームメンバーが自分自身に何を頼れば良いのか、より理解が深まる。

その上で、自分自身が抱える問題について、自分自身で調査をすることができる範囲と、それを越えステークホルダーに頼らなければならない分界点 (マッピング) を理解する。そのようにする事で、組織行動をとる時に一人ではなし得ない結果を生み出すための力を発揮できるようにする。また、IPA の「IT 人材白書 2011 [33]」によると、調査対象となった IT 関連の業界団体 (社団法人電子情報技術産業協会・社団法人情報サービス産業協会・社団法人コンピュータソフトウェア協会・社団法人日本情報システム・ユーザー協会) の会員企業と東京商工リサーチ社データベース登録企業等 (計 3,000 社) の中から回答を得た 533 社にて、10 年以上勤続している社員が 50% 未満の企業は 40% 存在している事がわかっている。そのため、他業種に比べ相対的に人材流動性が高い職種であると言え、短い期間で柔軟にに組織構造を理解するスキルを獲得することは欠かせないものである。

また、ソフトウェア開発ではステークホルダーが存在し、そしてその中で役割分担が存在する。要求を出す者と開発する者、そして開発する者の中でもライブラリアン[*1]やサブシステム毎の担当者が存在する。その中で各人に設定された役割分担の中で、ソフトウェアを完成させる一つの目標に向かって開発を進めるためには、コミュニケーションなくして進める事は出来ない。

4 については、「テクニカルスキル」に該当する。技術者に対する教育であるため、当然のことながらソフトウェア開発に関する技術を学ぶ必要がある。

5 については、「コンセプチュアルスキル」にあてはまる。ソフトウェア開発を継続して行くためには、教育を通じて学んだ技術ばかりではなく、将来にわたり継続して自分自身でソフトウェア技術を学び続ける必要がある。また、ソフトウェア開発を進めるにあたって、顧客から提示される要求をより明確かつ適切な形に咀嚼し、完成させる能力が必要となる。その元となる、調査や考察を行うプロセスを教育に織り込んで行く。これは、情報フルーエンシー [34] が目標としている「情報科学・ 情報工学の知識を援用して、自分の活動を分析しながら、情報環境改善を行なう。」こととも合致する。

[*1] ライブラリを開発・管理する担当者を特にライブラリアンと言う

3.4　技術者が関わるステークホルダーの範囲

コミュニケーションスキルを獲得するに当たり、関与するステークホルダーの範囲について、3.1 の 4 者に設定する。定義する。ステークホルダーとは、特にソフトウェア開発にあたって技術・要求を初めとした開発業務に直接関与する情報の疎通が行われる人物を対象とする。

1. 子弟・先輩後輩 — 入社初期に新人教育や部署配属直後にその関係が構築される。多くはマンツーマン、または数人で構成される。すべてのドメインの中で、最も身近に存在する。
2. チームや部署内 — 先輩、後輩の次に接するドメイン。所属する企業内でのチーム・部署などの 10 人〜数十人で構成される組織を指す。子弟・先輩関係に次いで身近なドメインであること、また上司などの会社内の公式組織の仕組みが明確に出る点で特徴がある。
3. 顧客・協力会社 — 顧客や協力会社[*2]など、所属している組織外ではあるが密接に関与するステークホルダーのことを指す。所属組織の枠を離れるため、より格式を重んじる人間関係を構築していく必要がある。
4. 広く一般 — 自らはその存在を認める機会は少ないが、自分自身を知る他者についてを指す。人間関係が一方通行であることが特徴である。

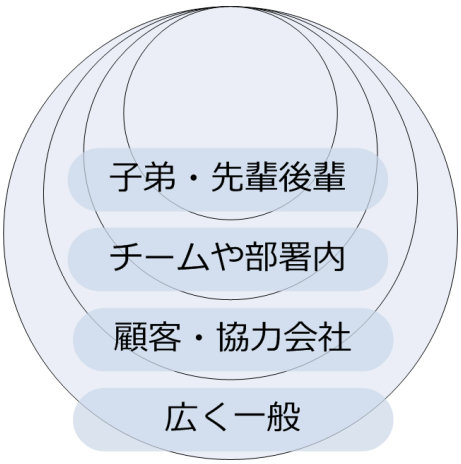


図 3.1　ステークホルダーの範囲

社会生活の人間関係において、身近なステークホルダーから順に範囲を拡大した区切りとし、上記を定義した。まず、新人技術者は技術者自身の至近にいるステークホルダーとの人間関係を構築する。そして、徐々にその対象を広げ、より多くのステークホルダーと

[*2] いわゆる外注・下請け企業のことを IT 企業はこのように総称する

表 3.1　到達目標点

内容	ITSS レベル	備考
頼ることができる人物について知っている	レベル 1	
部署内の人物の役割分担を知っている	レベル 1	
自らの専門分野を把握し自力で調べる所/他人に聞く所を区別ができる	レベル 1〜2	レベル 2 への準備
意思決定にかかわる人物を知っている	レベル 2	
自らのプレゼンスが部署内で理解されている	レベル 2	レベル 3 への準備

の人間関係を構築していくことになると考えている。

ここで、本研究は新人教育を目的とするため、業務開始後 3 年以内に主に関与する「子弟・先輩後輩」および「チームや部署内」の範囲を中心に検討を進める。「子弟・先輩後輩」は必ず関与する事となる人間関係のため必要であり、また「チームや部署内」は、組織でソフトウェア開発を進めるにあたってなくてはならない人間関係となるためである。

3.5　スキル養成指標の調査と選択

本研究では、2 章にて技術者のスキル指標を調査した結果を基に、ITSS [19] を用いる。その理由は、日本国内でオーソライズされた技術者教育のための指標であること、また技術の側面の到達点が明確でありレベルに応じた人間関係構築能力を養うことで技術者の能力をより確固たるものにすることができるからである。

ITSS レベル 1 の段階は実戦投入前の教育段階であるから、シラバスにある IT 科目を参考にしつつ主に子弟・先輩関係からステークホルダーのドメインを広めていくことを主眼に置く。ITSS レベル 2 の段階は、実戦投入が始まっているとしチーム内や部署内での自分自身のプレゼンスを確立していくことを主眼にする。なお、ITSS レベル 3 以上は、顧客・協力会社およびそれ以上の人間関係を伴うものであり、新人教育では賄いきれない要素が多々あるため本研究では対象としない。

教育内容は段階的にあげるものとし、到達する目標点を表 3.1 に挙げる。

3.6　教材の提供方法の検討

教育にあたって、本研究に対して用いる事ができる教材の提供方法について検討を行った。その際、インストラクショナルデザイン [35] で用いられている教育手法や、筆者が個人的に知る手法を取り上げた。

- 座学
- ケースメソッド
- ロールプレイング

- ペアプログラミング
- ビデオ鑑賞
- プロジェクトごとの評価
- ペーパーテスト

上記手法の位置づけを図 3.2 に示す。縦軸は、左に行く程に理論を基にした学習の側面が強く、右に行く程に実務を通じた学習 (OJT) の側面が強い事を示す。横軸は、上に行く程に座学の側面が強く、下に行く程に実践の側面が強くなる事を示している。

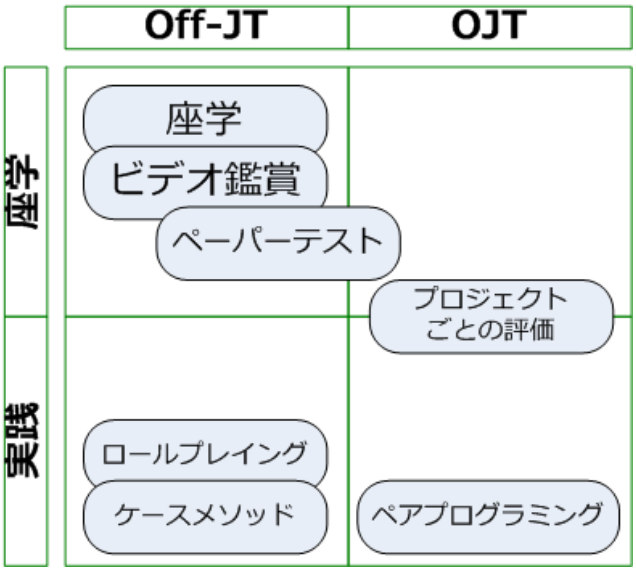


図 3.2　教育方法の違い

座学は、体系的な知識を学習するための座学を実施する。これを、従来型の教育手法と定義し、今回研究対象とする教育方法との比較の基礎とする。座学は古典的な手法であり、技術や人間関係に関する教育カリキュラムも既に存在し、準備が容易である。同時に数十人に対して教育を行うことができるため、他の方法に比べてスケーラビリティが高い。体系的に学習を行える反面、実践的な学習にはなりにくいのが難点である。

ケースメソッド (以下、ケース) は、仮想のシステム開発計画を設定し、被験者がケースに基づいて開発を行うものである。IT スキル調査 [18] によると、短期間かつ実践的な教育を施す手法として IT 企業で用いられているとあり、ケース実施時のプロジェクトを通じ、技術の獲得を行いながら人間構造の理解を進める。1 名から数名のグループで活動し、活動結果の評価は教育担当者が行う。技術力と人間関係両面の教育が可能であり、より実践的な知識を獲得するにあたって効果的である。しかし、一定期間に集中して行う必要があるため、業務と並行で進めるには難しい。また、ケース作成にあたって教育担当者に時間的・技術的負担を強いる問題もある。

ロールプレイングは、あらかじめ設定されたシナリオをもとに、被験者が人とのコミュニケーションを通じて問題を解く手法である。トラブルシューティングなどの短時間で実

施できるものを設定する。数名のグループでロールプレイングを行う。ケースメソッドと違う点は、学習そのものに用いる時間がロールプレイングを行う数十分〜一時間程度で済み、より短時間で行う事が出来るのである。

ペアプログラミングとは、プログラミングや講義ではシステム構築全般にわたって、教育担当者と新人がペアとなって 1 つのタスクをこなすものである。Williams [36] によると、実施にあたっては、教育担当者が実務に当たる時間は削られてしまうデメリットがある。しかし、新人一人で業務にあたる時に比べて問題が発生した時の作業の手戻りを減らし、さらに教育担当者が既に獲得している技術が実務を通じて直に学ぶ事が出来るメリットがあることが報告されている。

ビデオ鑑賞は、対人折衝の方法をはじめとした、人間の立ち居振る舞いに関わる学習を行うのに有効である。例えば、表 3.1 で挙げている「まず頼ることができる人物について知っている」事を学ぼうとする時、文章ベースの学習では会話中のやり取りの要領を掴む事は難しい。そこで、教育目的に沿った「悪い例」と「よい例」を 3 分間のビデオで実演し、より短時間で数多くの情報を伝えようとするものである。

プロジェクトごとの評価は、プロジェクトごとの活躍について、事前に課題を提示した上でアクションラーニング [37] を通じて学習する。アクションラーニングとは、グループで現実の問題に対処し、その解決策を立案・実施していく過程で生じる実際の行動と、振り返りを通じて、個人、そしてグループ・組織の学習する力を養成するものである。結果は、教育担当者が評価を行う。他の手法に比べ、時間の占有をしないため業務を圧迫しにくいことが特徴である。しかし、最低限の実務投入のための知識を要するため、教育初期の段階で用いることはできない。

ペーパーテストは、紙面によるテストを行う。座学と同様に古典的な手法であり、到達度を客観的に測る手法として便利である。教育に当たる所要時間も所要時間も数時間で済み、かつ同時に数十人対し教育を行う事が出来るため、実施も容易である。ただし、プロジェクトごとの評価と同様に教育初期の段階で用いづらい点があり、一定の教育に区切りがついた段階で用いることになるかと思われる。

以上の特徴や弱点を検討した結果、検討した結果、次の 3 つの理由からケースメソッドとロールプレイングを採用した。

1. 開発工程を通して学習できる
2. 役割分担を設定して学習できる
3. 積極的なコミュニケーションを通じた学習を期待できる

1 については、ケースメソッド形式を使えば企画から設計、開発、テスト、そして完成までを通して学ぶ事ができる。他の手法は、特定の工程にに対して学ぶ事に絞られるため、今回は採用しなかった。

2 については、ロールプレイング形式を用いる事で開発時にプログラマ・デザイナ・テスタと言った役割を設定できる。実際の開発では全員が同じ役割を担う事はほぼなく、よ

り現実に即した状況を作り出す事を意図している。

3 については、1,2 を通じ、各担当者が積極的なコミュニケーションを行う事を必要とする環境を作り出すことができる。その上で、コミュニケーションスキルを向上させるための学習を促すことが期待できる。

3.7　チーム構成の検討

チームを構成するにあたり、「プログラマ」「デザイナ」そして「テスタ」の 3 つの独立性の高いロールを設定している。役割毎に割り当てられる仕事は、次の通りである。

- プログラマ — ソフトウェア内部の開発を行う
- デザイナ — ソフトウェア外部の開発を行う
- テスタ — プログラマ・デザイナが開発したソフトウェアの品質を確認する

理由は、独立性を高める事で、常にソフトウェア開発の各工程で要求・設計が合致しているかを確認しなければならなくなるため、コミュニケーションが発生しやすくなる。また、コミュニケーションが積極的に求められる状況より、被験者にコミュニケーションの必要性の気づきを持ってもらいやすくなることが期待できる。

また、上記のチーム構成を構築するために、被験者は最低 3 名で構成する必要がある。

3.8　まとめ

本章では、教材の作成指針についての議論を行った。

3.2 節において、技術者に求められるコミュニケーションスキルについて定義した。3.3 節において、3 つのコアスキルをもとに教材を作成することを確認した。同時に、新人技術者のコミュニケーションスキルが何であるかを定義し、教材に織り込んでいく事を示した。3.4 節において、教材作成時に対象とするステークホルダーの範囲を、「子弟・先輩後輩」および 「チームや部署内」とした。新入社員時代に最低限関与する範囲に限定したためである。3.5 節において、教材作成時の指標を ITSS をベースとする事とした。3.6 節において、技術者を教育する際に用いられる教材から、ケースメソッド形式とロールプレイング形式を選択した。また、短期間で体系的に教育を実施できるものがケースメソッドである事を示した。3.7 節において、ロールの独立性を高め、コミュニケーションが積極的に行われるようチーム構成を検討した。

以上より、技術とともにコミュニケーションを学ぶ事で、コミュニケーション単体で学ぶ教育方法に比べより効果的に技術者に教育が行えるかを検証して行くものとする。

第 4 章

実験実施計画の作成

4.1 はじめに

先章にて、3 つのコアスキル「テクニカルスキル」「ヒューマンスキル」そして「コンセプチュアルスキル」をもとに技術者が獲得する事が望ましいスキルを述べた。その上で、これまでテクニカルスキルに重点が置かれてきた技術者教育にヒューマンスキル、即ちコミュニケーションスキルの教育を加える事で、コミュニケーションが積極的に行える新人技術者が養成できると考えた。

以上より、3 つのコアスキルを技術者がケーススタディ形式で理解できるようにした。

作業計画の作成は、次の 6 項目を行った。

- 養成するスキルの設定
- 実験工程と評価基準点の設定
- 評価のための準備
- 被験者の募集
- ケースと実験スケジュールの作成
- 納品物の定義

4.2 節において教材を通じて養成するスキルを設定する。4.3 節において、実験の工程および工程内で評価を実施するタイミングを定義する。4.4 節において、評価を実施するにあたっての指標やアンケートの準備を行う。4.5 節は、本研究を検証する際に必要となる被験者の募集について述べる。4.6 節において、作成するケーススタディと実験のスケジュールを明記する。4.7 節において、被験者からケーススタディ実施時に提出を受ける納品物を定義する。最後に、4.8 節において、倫理的な配慮について記述する。

4.2　養成するスキルの設定

実験を行うにあたって、新人技術者に対して養成するスキルの設定から開始した。3.3 節において述べた 3 つのコアスキル「ヒューマンスキル」「テクニカルスキル」そして「コンセプチュアルスキル」を基に設計する。

ヒューマンスキルは、コミュニケーションスキルを獲得することを主眼に、次の通りに細分化する。

- 自分の役割の理解 — 課題に対し、自分自身の貢献を事前にどのように理解しているか。
- グループ内での役割分担の理解 — チーム内において、他者との役割分担を理解し、気配りを行き届かせているか。
- チーム形成に対する貢献 — チームに置かれた自分自身が、「意思と実践との乖離」「過去の経験[*1]」そして「リーダーかメンバーかの特質の把握」をどのように捉えているか。

テクニカルスキルは、新人技術者としてなくてはならないスキルを、次の通り定義した。

- プログラミング学習 — 技術者として技術の根幹となるプログラミング能力を獲得しているか。
- システム開発力 — プログラミングそのものだけではなく、システムのデザインや、テスト、自動化などの工程全般で必要となる能力を獲得しているか。
- 工程管理力 — システム開発を進めるための進行管理の能力を獲得しているか。トラブルに対する対処能力も含まれる。

コンセプチュアルスキルは、技術者がシステムに対する要求を理解し形にするための力となるものである。理解を進めるにあたっては、自分自身に対する知識を蓄え、さらに開拓して行くための力を下記の通り定義する。

- 解決力 — 問題が発生した際に、解決を行うための手段をどれほど持ち合わせているか。情報収集だけではなく、思考法なども含まれる。
- 情報収集力 — 情報収集のための手段をどれほど持ち合わせているか。
- 実践力 — 自分が持っている情報や知識を、実践を通じて理解を深めているか。
- 発信 — 理解した情報を、他者に伝える事によって理解が定着しているか。

[*1] 学生のためアルバイトや生徒会活動を含める

4.3 実験工程と評価基準点の設定

実験を実施するにあたり、工程を定める。

実験実施前に、被験者となる情報系学部の 4 年生の学生を募集し、今回の実験についての説明を行う。同時に、被験者が所属している研究室の指導教授への説明も実施し、協力をより仰げるよう心がけた。最後に、初回アンケートを通じて被験者の現状を把握し、ケースの作成や後述する評価に役立てる。

実験の準備は、主にケースの作成を行う。被験者のレベルに適切にあわせられるよう、実験実施前に情報を収集し、その能力を評価した上でケースの作成に入る。

ケース完成後は、実際に被験者に対してケースを適用する。この適用を 2 段階に分ける。まず、1 週間をかけて設定したケース課題をチームで実施する。そして、最後の日にワークショップを催し、システムを完成させて行くのである。また、実践している中で表出する能力獲得のプロセスを観察する。

最後に、被験者の事後の状態を確認する。事前・事後の状況を評価し、実験の効果を確かめる。

4.4 評価のための準備

4.4.1 評価の観点

新人に対する評価の観点は、次の項目を想定している。観点の作成にあたっては 3.3 節で挙げた 5 つのスキル獲得目標をもとにピックアップした。

- ケースで目標としている人間構造の理解の水準を達成しているか — 「ヒューマンスキル」の獲得が進んでいるかを確認するものである。
- ケース内に織り込まれている技術要素の理解が進んでいるか — 「テクニカルスキル」および「コンセプチュアルスキル」の理解が正しく進んでいるかを評価するものである。
- 作業経緯や人との交流を通じた省察が積極的に行われ課題の目的を理解していたか — すべてのコアスキルに関するものとなり、新人教育後の実践段階においても自分自身で人間関係・技術の問題解決を行い、学習を行う力が備わっているかを確かめるものである。

この観点に立ち、工程毎に収集を行うデータを表 4.1 のように定義する。

表 4.1　実験工程 (凡例: 人: ヒューマンスキル, 技: テクニカルスキル, 描: コンセプチュアルスキル)

工程	評価方法	人	技	描	評価観点
実験実施前	アンケート	○	○	○	実験前の状態を確認する
ケースの実施	レポート	-	○	-	ケースの進捗を確認する
ワークショップ	ビデオ撮影・SYM-LOG･完成したソフトウェア	○	○	○	コミュニケーションの状況を観察し、成果物を完成させられるかを確認する。
実験実施後	アンケート	○	○	○	実験実施前との比較を行う

4.4.2　事前アンケートの作成

まず、実験実施前の被験者の状況を把握するために作成したアンケートについて説明する。

アンケートを作成するにあたって、被験者が大学 4 年生の情報系学部に所属している事を前提とし、用語や回答欄は、記入のしやすさに配慮した。

また、職場でアルバイトをしている大学生が情報系学部に所属していた事から、業務時間外を用いて内容のレビューを実施した。

アルバイトの学生から得た情報は、次の通りであった。

1. プログラムのコーディングは講義の実習を通じて学習を進めているため、プログラミング自体の技術用語は理解できる。
2. ソフトウェア工学、特に工程に関する講義を受けている学生が一部のため、理解していない学生もいる事を考慮する必要がある。
3. チームでシステム開発を行うワークショップ等は余り行われていない。
4. 論理的に話すための訓練も始まっているため、回答時に論理的に曖昧な要素を残さないほうがよい。
5. 自分より優れた人がいる事を理解し始めているので、相対評価の場合は同年代に絞り込んだ方が精度の高い回答を得やすい。

以上の意見を踏まえて作成したアンケートを、付録 B として掲載する。

4.4.3　在宅での実験時に提出するレポートフォーマット作成

被験者が作業を滞りなく進めているか確認を行うことと、被験者自身も提出する成果物の情報をまとめやすいよう、レポートフォーマットを作成した。レポートフォーマットは付録 C として掲載する。

4.4.4 ワークショップの評価指標の作成

7 日目のワークショップ時、特にヒューマンスキルの獲得状況を評価にあたり、SYM-LOG の日本語改良版 [32] を用いる。あわせて、ビデオを用い作業の模様を撮影し、後日評価を行いやすくする。なお、本論文では用語のわかりやすさを重視し「向文脈的〜脱文脈的」の評価軸は「冷静〜非冷静」と表現を変更している。

「テクニカルスキル」は、事前の準備に基づき、当日はソフトウェアを完成させる事が出来るかを確かめることで、評価する。

評価シートは、付録 D として掲載する。

4.4.5 最終アンケートの作成

最終アンケートでは、事前アンケートで取得した情報をもとに、3 つのスキルに対してどの程度の変化が現れているかを測定する。

完成品は付録 E の通りである。

4.5 被験者の募集

本研究において、被験者の募集は最も重要な作業であった。特に、私自身が所属する研究科の存在するキャンパスでは、フルタイムの学生は在籍していないため、直接参加者を募るのが困難な状況であった。そこで各種連絡手段を用い、2 大学 2 研究室から合計 6 名の参加者を募る事が出来た。以下、被験者と記載する。

- 国立 α 大学 情報系学科所属 大学 4 年生 3 名
- 国立 β 大学 情報系学科所属 大学 4 年生 3 名

各研究室の指導教授から連絡を受けた後、本研究についての説明を実施した。説明文は付録 A として掲載する。

4.6 ケースの作成

4.6.1 ケース作成の方針

ケースメソッド形式の実験実施に際し、1 章と 3 章をもとに次の 5 点を考慮しケースを作成した。

1. 普段慣れ親しんだものを利用する
2. 7 日間で形にできる
3. 役割分担を作る

4. 工程を設定する
5. 能動的に楽しめる

1 については、業務知識を改めて学ばずとも理解できることを目指した。情報系の学生は Twitter をはじめとしたソーシャルメディアを利用している場合が多い。そのため、開発するソフトウェアの企画を被験者自身が育てやすいテーマを選択した。

2 については、短い実験期間で、実際に被験者が早い段階でソフトウェアが動いている事を確認しやすいものを開発できることを目標にした。そこで、mashup [38] と呼ばれる、既存の Web サービス提供元が外部のソフトウェアと連携する技術を利用した開発を行うよう設定した。

3 については、ロールプレイングの要素として「プログラマ」「デザイナ」そして「品質管理」の 3 つの分担を組み込んだ。これは、それぞれの役割分担を理解しながら一つのソフトウェアを開発する状況を再現するためである。

4 については、組織での開発を通して経験がない被験者に向けて、一通りの開発工程を学べるよう作成した。ウォーターフォール型開発モデルとはなっているが、被験者同士のインタラクションを行う必要のある成果物を定義する事で、開発工程を学ぶと同時にコミュニケーションスキルを学べる形をとっている。

5 については、何よりも楽しんで開発できるよう、開発するソフトウェアに自分たちの「意思」を込められる要素を残すようにした。目的は、被験者のモティベーションを高める事で積極的な実験参加を促進し、さらにソフトウェア開発そのものを楽しんでもらえるよう配慮した。

4.6.2 ケースの概要

作成したケースは「ケース 1 — Twitter の mashup サイトを作ろう」である。
作成にあたり、先節の考慮点を次の通り解決している。

- 普段慣れ親しんだものであること — 辰己ら [39] の行った、東京農工大学へ 2012 年度に入学した大学一年生 881 名中 865 人に対する調査では、学生は Twitter をはじめとしたソーシャルメディアを利用している場合が多いと示している。そのため、開発するソフトウェアのイメージを被験者自身が扱いやすいテーマを選択した。
- 7 日間で形になるもの — 実験期間が 7 日間と短い状況下で、実際に目で見て動いている事を確認しやすいものを開発するために、mashup と呼ばれる手法を用いる事が出来るようにした。
- 役割分担を作る — 今回「プログラマ」「デザイナ」そして「品質管理」の 3 分野の分担を設定した。これは、それぞれの役割分担を理解しながら一つの製品を開発する状況を再現するためである。

- 工程を設定する — 事前アンケートより、組織での開発を通して経験がない被験者に向けて、開発の工程を粗ではあるものの通すよう設計した。ウォーターフォール型開発モデルでの開発となるが、要求は被験者自身で決めるため柔軟な開発・仕様変更が可能と考えたため、あえて設定した。
- 能動的に楽しめる — 楽しんで開発できるよう、自分たちの「意思」を込められる要素を残すようにした。

完成したケースおよび工程中の作業内容は、付録 F を参照されたい。

4.6.3 評価方法

3.3 節で示した「ヒューマンスキル」「テクニカルスキル」そして「コンセプチュアルスキル」の 3 つのスキルの獲得状況を確かめるべく、4 つの段階毎にケース実施前後の評価方法を検討した。

- 実験前
- 実験中 1〜6 日目
- 実験中 7 日目
- 実験後

実験前は、3 つのスキルに関する事項を調査するアンケートを作成し、実験実施前の被験者のスキル獲得状況を確かめる事とした。

実験中 1〜6 日目は、与えられた課題に対して適切に工程を終えているか、課題を提出してもらう。ここで、特に実験前との「テクニカルスキル」と「コンセプチュアルスキル」を比較し評価できるようにする。

実験中 7 日目は、ワークショップ中にソフトウェア開発を進めるにあたり、実装や仕様の微調整を通じてコミュニケーションスキルを培えるかを評価する。評価時は、ワークショップの前半・後半を時間で分け、被験者が「ヒューマンスキル」が向上しているかを評価する。

実験後は、付録 E を用い 3 つのスキルに関するアンケートに答えてもらい、実験実施前との比較を行う。その中で、今回のケース実施によって被験者が能力を獲得できているか、最終的な評価を行う。

以上を基に、データの収集方法とフォーマットを作成する。あわせて、実験実施スケジュールを表 4.2 に示す。

4.7 成果物の定義

作業が滞りなく進んでいるかを確認するため、実験工程に対応する成果物と納品日を表 4.3 の通りに定義した。

表 4.2　スケジュール

日程	所要時間	作業内容
1 日目	1〜2 時間	コンセプト作成・外部設計
2 日目	1〜2 時間	外部設計
3 日目	1〜2 時間	内部設計
4 日目	1〜2 時間	内部設計・画面設計
5 日目	1〜2 時間	テスト計画・リソース制作・テストプログラム試作
6 日目	1〜2 時間	テスト計画・リソース制作・テストプログラム試作
7 日目	6 時間	プログラム開発・テスト実施 (於 筑波大学 東京キャンパス)

表 4.3　成果物の一覧

日程	成果物
第 1 回 (2 日目)	選択したプラットフォームの情報
	コンセプトのメモ
	作業分担リスト
	外部設計の図面
第 2 回 (4 日目)	機能設計書
	画面遷移図
	画面設計図
第 3 回 (6 日目)	テスト計画書
最終回 (7 日目)	完成品のプレゼンテーション
	ソースコード
	テスト結果が記入されたテスト計画書

第 1 回は、「選択したプラットフォームの情報」は、スマートフォン向けの開発か、PC サイト向けの開発かを明示する。プラットフォームを狭める事で、開発の負担を軽減しているためである。「コンセプトのメモ」は、どのような形でソフトウェアを利用するのか企画を練った結果を記載する。「作業分担リスト」は、ケースで定義された分担に誰があたるかを記載する。最後に「外部設計の図面」は、各アクターがどのような通信を実施するのか全体像を明記するものである。

第 2 回は、「機能設計書」は、開発するソフトウェアにどのような機能を実装するかを列挙する。「画面遷移図」は、ソフトウェア全体でどのような画面が存在し、画面感をどのように遷移するかを表す。「画面設計図」は、各画面のレイアウトを定義するものである。

第 3 回は、「テスト計画書」を作成する。第 2 回までに作成した資料を基に、ソフトウェアが正常に動作しているかを確認するための事項を列挙する。なお、今回は実験期間の関係から異常系のテストは含まない事とした。

最終回は、完成したソフトウェアをもとに定義する。「完成品のプレゼンテーション」

は、出来上がったソフトウェアについて、特徴と扱い方について説明するものである。「ソースコード」は、開発したソフトウェアのソースコード全文をデータで取得する。「テスト結果が記入されたテスト計画書」は、第 3 回で提出したテスト計画書に、テスト結果を記入し、提出する。

4.8 倫理的配慮

本研究での実験の実施にあたっては、国立大学法人 筑波大学 ビジネスサイエンス系 研究倫理委員会からの承諾を得ている。また、被験者および被験者の指導教授に対してはプライバシーの配慮から個人を特定できる本名や所属する大学の校名を表出しないこと、また参加の是非や結果によって不利益が生じない旨の説明を行い、承諾を得た。

4.9 まとめ

本章では、実験を実施するにあたっての計画について述べた。3 つのコアスキルを軸に、コミュニケーションとソフトウェア開発工程を同時に学ぶことができるケーススタディを作成する事が出来た。また、ケーススタディ実施前後のスキルの変化を確認できるよう、アンケートをはじめとした定量的・定性的データの収集の基盤を構築した。

4.2 節において、3 つのコアスキルを基に、新人技術者に対し養成するスキルを定めた。4.3 節において、養成するスキルの獲得経緯を調べるための工程とその中での評価基準点を設定した。4.4 節において、4.2 節で定めた項目に従い、評価観点を定めた。ここでは「アンケート」「レポート」、またワークショップ時に「ビデオによる行動観察」「SYMLOG」を用いることを決めた。4.5 節において、本実験が対象としている大学の情報科で学ぶ大学 4 年生を 2 大学 計 6 名を集める事が出来たことを述べた。4.6 節において、ケースの作成方針と経緯を述べた。その中で、1 章と 3 章で述べた目的に沿って「ケース 1 — Twitter の mashup サイトを作ろう」を完成させることができ、あわせて実験スケジュールをまとめた。また、成果物は 4.7 節で定義している。ソフトウェア開発を通して学んでいない事を想定し、順を追って開発を進められるよう、段階を追って納品が進められるようにしている。その上で、ソフトウェアが次第に完成する事を学生が体感できるようにしている。最後に、4.8 節において被験者に対する倫理的配慮について述べた。

次章及び次々章 (5 章と 6 章) では、実験実施結果を説明する。

第 5 章

第 1 回 実験

5.1 はじめに

先の 4 章で作成した実験実施計画を基に、α大学の学生に対して実験を行った。

本章では、5.2 節において実験の概要を説明する。5.3 節において、被験者のプロフィールを記載する。実験中の状況は、5.4 節では 1〜3 回目の納品まで、5.5 節では 7 日目のワークショップについて述べる。実験結果は 5.6 節に、実験の中で発生した問題は 5.7 節にて記載した。考察は、6 章で述べている第 2 回実験との比較を含め、7 章において述べる。

5.2 実験実施概要

第 1 回は、α大学の学生 3 名 (以下、α大学チーム) に対して実験を行った。実験期間は、2012/09/16 〜 2012/09/22 の 7 日間である。開催場所は、6 日目までは各自の自宅または大学、7 日目は筑波大学 東京キャンパス 116 号室にて実施した。

実験のスケジュールを、図 5.1 に示す。

2011/09/01 に、α大学の学生および指導教授に対して、実験内容のガイダンスを実施した。場所はα大学の講義室を借りて実施した。その際、次の事項を説明した。

1. 全体像の説明
2. 倫理委員会にて規定された事項の説明
3. システム要件 (背景・機能要件・非機能要件)
4. 作業工程の説明
5. スケジュール
6. 成果物の納品方法
7. 準備する物品
8. 連絡事項
9. 連絡先

タスク名	ID	開始日	終了日	期間
コンセプト作成	1	2012/09/16	2012/09/16	1d
外部設計	2	2012/09/16	2012/09/17	2d
納品 第1回	3	2012/09/17	2012/09/17	0d
内部設計	4	2012/09/18	2012/09/19	2d
画面設計	5	2012/09/19	2012/09/19	1d
納品 第2回	6	2012/09/19	2012/09/19	0d
テスト計画	7	2012/09/20	2012/09/21	2d
リソース制作	8	2012/09/20	2012/09/21	2d
(テストプログラム試作)	9	2012/09/20	2012/09/21	2d
レポート提出 第3回	10	2012/09/21	2012/09/21	0d
プログラム開発	11	2012/09/22	2012/09/22	1d
テスト実施	12	2012/09/22	2012/09/22	1d
最終アンケート	13	2012/09/22	2012/09/22	0d

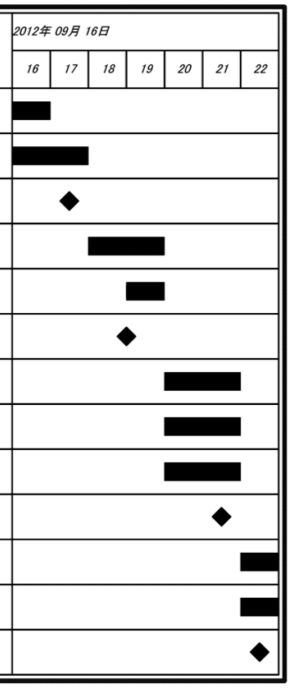


図 5.1　α大学 実験スケジュール

付録 F にて、実際に使用した説明資料を掲載した。

5.3　被験者のプロフィール

α大学チームの被験者は 3 人である。共通する事項はいずれの 3 人も男性であり、同じ研究室に所属している。そのため、ほぼ毎日顔を合わせており、チーム内での私的な人間関係は良好である。

続いて、各個人のプロフィールを、アンケートから取得した情報を基にまとめる。

■被験者 A　プログラマ担当。これまで、講義や実験、また私的な活動でプログラミング経験があり、チーム内で最もプログラミングスキルを持つ。かつて、小規模ながらコミュニティ系 Web サービスを書き上げて運用していた事もあった。ただし、ユーザインタフェイスやアイコン等のリソース作成は不得意であり、当初から他のメンバーの支援を求めていた。

■被験者 B　デザイン担当。講義や実験でプログラムを書いた事はあるものの、本人は不得手であると言う認識。画面設計やユーザインタフェイス制作で力を発揮したいと希望していた。また、チーム内でリラックスした空気づくりを行えるよう、取り組もうとしていた。

表 5.1　α大学 ワークショップのスケジュール

時刻	作業内容
10:30〜10:45	本日の作業の流れの整理
10:45〜12:30	各自 担当作業
12:30〜13:30	昼食
13:30〜15:00	各自 担当作業
15:00〜15:50	プログラム・画面 結合作業
15:50〜16:00	問題点の整理
16:00〜16:40	プログラム・画面 結合作業
16:40〜17:00	問題点の整理
17:00〜19:50	プログラム・画面 結合作業
19:50〜20:00	作業結果確認 ソフトウェアの内容のプレゼンテーション

■被験者 C　リーダー、及びテスト担当。当初よりリーダーを志望していた。講義や実験はもとより、個人的に Windows 上でゲーム等のプログラミングを行った経験を持っている。同時に、個人的なゲーム開発の経験から、ユーザインタフェイスのリソース作成についても知識があった。そのため、今回の実験で実施する作業の経験範囲は最も広い人物であった。

5.4　1 回目〜3 回目 納品物の確認

1 回目から 3 回目まで、予定通り成果物の納品を受けた。

担当割りは、ガイダンス時に本人たちの立候補でスムーズに決定した。

納品時点で、作成したコンセプトや外部設計資料の作成が問題なく進んでいると確認した。残す作業はプログラムの作成及び結合となっていた。

5.5　7 日目 ワークショップの状況

5.5.1　全体の流れ

7 日目のワークショップは、プログラムを完成させる事を目的とし、被験者に作業に取り組んでもらった。実際の作業タイムスケジュールは、表 5.1 の通りとなっている。

当初、終了は 18:00 であったが、スタート時にネットワーク接続のトラブルが発生した事と、作業の遅延が発生したため、19:50 まで延長して実施した。

5.6　実験結果

5.6.1　ソフトウェアのスクリーンショット

β 大学チームが開発したソフトウェアのスクリーンショットを、図 5.2 に示す。なお、個人を特定できないよう、被験者の twitter ID と所属研究室を伏せ、アイコンを架空のものに設定している。

図 5.2　α 大学チームが開発したソフトウェアのスクリーンショット

5.6.2　アンケート結果

事前及び事後アンケートから、被験者の変化を分析する。評価にあたっては、3.3 節をもとに行う。

まず、表 5.2 に、「ケースで目標としている人間構造の理解の水準を達成しているか」即ちヒューマンスキル部分についての内容を記述する。アンケートの書式は、事前は付録 B、事後は付録 E に掲載した。

続いて、表 5.3 に、「ケース内に織り込まれている要素技術を理解」(テクニカルスキル) 部分についての内容を記述する。「プログラミング能力」で○をつけた項目は、上から順に 1 番〜6 番とした。また、「ソフトウェア開発能力」は、上から 1 番〜15 番とした。

続いて、表 5.4 に、「システムのデザインを行う事が出来る力が育む」(コンセプチュアルスキル) 部分についての内容を記述する。

*1 全て終わらなかったため判断できないとの事

表 5.2 α大学チーム 前後比較 — ヒューマンスキル

項目		事前の希望	事後の感想
自分自身の役割	A	プログラマとして活動する	できなかった。
	B	アイディア・企画提案	デザイン担当になり問題なくこなせた。
	C	プログラミング・作業進捗管理	テスターとして実装の詳細まで踏み込んで計画を立てたかった。
仲間の役割	A	UI, 画をお願いしたい	(無回答[*1])
	B	プログラムの重要な所は任せたい。	テストは出来なかったが、項目は作り込まれていた。プログラミングは最後までがんばっていた。
	C	自分の作業だけなくアイディアを提供したい	デザイナは要求を満たしながらわかりやすい UI を作った。
自分自身の配慮	A	周りの邪魔をしないで静かにやる。	躓いた時に声を上げて助けを求めた。
	B	固くならないよう会話をする。	作業をしていていつものように話を振れなかった。
	C	お互いの気持ちを把握する。	全体像は共有できた。技術部分は共有しきれなかった。
これまでの仕事経験	A	飲食店・塾講師 プログラム業務無し。	私用で Web サイト開発をした経験が役立った。
	B	鉄道会社・飲食店	あまり活かせなかった。
	C	弁当屋・ソフトのデバッグ	デバッグ経験は活きた。
チーム内での立ち位置	A	メンバー	メンバー あまり他のメンバーと交流がなかった。
	B	メンバー	メンバー リーダーに丸投げせず協力できた。
	C	リーダー	リーダー 役割は果たせなかった。プログラム結合時に配慮できなかった。

5.6.3 SYMLOG を用いた小集団構造把握

SYMLOG を用い、α大学チームの集団構造を分析した結果を、図 5.3 に示す。評価は、筆者が実施した。

バブルチャートの縦軸は数値が高い程友好的な接し方を心がけていた事を示し、横軸は右へ行く程に冷静な対応を示している事を示し、円の半径が大きい程リーダーシップを発揮している事を示している。前半は主に 10:30〜15:00 の間、後半は主に 15:50〜19:50 の間を対象としている。「被 A」とは被験者 A を指し、B, C についても同様である。

表 5.3　α大学チーム 前後比較 — テクニカルスキル

項目		事前の状況	事後に獲得
プログラミング能力	A	1,3	1,2,3
	B	1,2	1,2,3
	C	1,2,3,4	1,2,3,4
ソフトウェア開発能力 (事前:”習った”以上) (事後:”手伝った”以上)	A	7,9,14	1,2,3,4,5,6,7,9,11,14
	B	10,11	1,2,3,4,5,9,10,11
	C	4,5,9	1,2,3,4,5,7,8,9,10
工程管理	A	最初に UI を設計し 機能を後から盛り込む	初期設計が適切に仕上がっていればプログラミングはただの作業
	B	目標の設定→企画→設計→実装→評価	自分が考えている事と相手が考えている事に相違があるとプログラムは結合できない
	C	UI 設計→内部処理の構成と流れを考える→実装	工程を管理するとプログラムを設計に合わせていく コードを書く比率が小さくなる

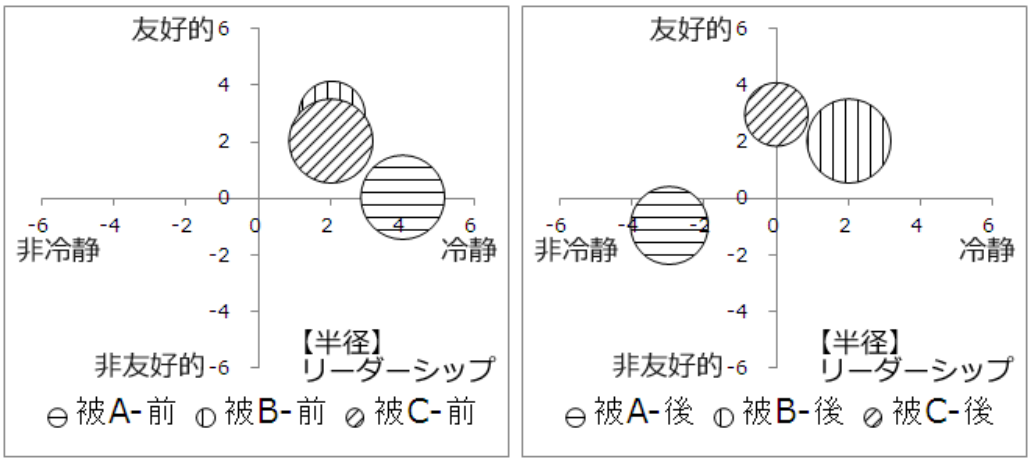


図 5.3　α大学 ワークショップ内での構造評定 (前半:左 後半:右)

5.6.4　会話内容

ビデオより、被験者のコミュニケーションスキルを垣間みる事ができる部分を引用し、前半・後半に分けて以下にまとめる。被験者 A, B, C はそれぞれアルファベット 1 字に略して記載する。「全」とは、全員を示す。

表 5.4 α大学チーム — コンセプチュアルスキル

項目		事前の想定	事後の理解
問題解決(技術)	A	2ch, 友人に相談	小さなテストプログラムを書き、それを大きくした。
	B	Google で検索, 友人に相談, 仕様書を見直す	jQuery Mobile の API ドキュメント,「jQuery Mobile で簡単スマーフォン向け Web サイト制作」(相澤祐介 著)
	C	Google で検索, ドキュメントを読む, 知人に相談, 原因を切り分ける	(同上)
問題解決(担当者間)	A	-	プログラムと HTML テンプレートの間の変数形式を必要不可欠なものに絞り込んだ。
	B	-	Twitter の仕様を充分理解しておらず連絡に手間取った。
	C	-	プログラムと HTML テンプレートの変数埋め込み時にうまく行かず作業が止まった。
普段の情報収集	A	ネットでサンプルプログラムを探す, 図書館	Twitter API の使い方や用語を知っていて役立った。
	B	ネット上のニュース, 技術系雑誌購読	Twitter API の技術使用をもっと理解しておくのだった。
	C	ネット上のニュース, メールマガジン	特になし
実践を通じて得た知識・技術	A	-	エラーログの確認方法
	B	-	jQuery Mobile の技術仕様
	C	-	一通りの開発プロセスを通す経験
実践	A	知人と Web サービスを 3 年間運営	-
	B	特になし	-
	C	Windows 用ゲーム開発, 開発補助ツール開発	-
発信	A	特になし	-
	B	特になし	-
	C	Wiki で開発 Tips をまとめて公開	-

■前半　12:00 頃

全　(開発作業が続いていた所でふと会話が始まる)
A,C　(黙々とコードを書いている)
B　(でんと座って A に問いかける) リストの名前…会話一覧は…
A　(眉間にしわがよる) どうやるんだっけ？
B　リストってさ、公開と非公開と 2 つあるんだよ。どっちも拾える？
A　(薄笑いを浮かべる) 拾えない臭い。
全　(場がしーんとする)

■後半　16:40 頃

全　(進捗が遅れている事がわかり始めている その中で小休止を終え作業を再開する前に会話が始まる)
B　(コーヒーを手に) 今どんな？
全　(一瞬の静寂の後に笑い。失笑に近い)
A　(コードを読みながら)mention から、項目をたどって…んーと、どうたどればいいかは、この中にあって。そのたどっていった個々の tweet を、どう管理っていうか。
B　ああー
A　が、今の課題。
C　たどってって、tweet の本文に行くと…
A　はい、ポンポンって出て。あとは順番に読んでって。
C　それは変数 (筆者注:内部プログラムから画面に渡す時の変数) に入れておけばいいじゃん。
A　何個あるかわかんない (筆者注:mention のやりとりの長さは API を 1 回実行しただけではわからない)。どれだけの長さなのか。
C　(身振りを使い) 名前があって、この変数があって…都度拾えば。
A　何回も API(筆者注:twitter API) を叩くとそれはよろしくないので (筆者注:twitter API は単位時間当たりに呼び出せる回数に制限がある)。少ない回数でやるには、1 つの tweet から、順に会話を追いかけて、追いかけるのと同時に会話を保存したいのと、1 つの項目に対して複数の経路が (筆者注:1 つの tweet に対して複数人が mention 可能) で、会話が分岐している所があって、いくつか追いかけないと行けない。こういくのと、こういくのと、こういくのと。一番新しい方から追いかけて行くから、一番新しい方がこういくの。それを見ながら保存して行くんだけれど、その追いかけるのと保存するのを 1 つのタスクでやっているので、このあとでこれをやると、これ消えちゃうの、データが。保存するのと、見つけるのを、分ければいいんだけれど、そうすると 1 つの会話を見るのに 2 周する事になっちゃうの。上

を探すのと、これを探すので。

A,C うーん…

A それは、面倒だなー、って言う。

A あ〜

A (眼鏡を直しつつ髪をかき分ける)

B (頭に両手をのせて伸び上がる)

C (何か言いたくて手のひらが前に出るもののなかなか言葉が出ない)

C 探しているときは、先頭かどうかわかんない？

A (コーヒーをすすりながら C を見る)

B うーん。

C (身振りを入れつつ話す) 保存した奴ってさ、どっかのバッファに積んでおいて、あとで会話をマージするとか。たどって行く時に、こうやってやって、もう一回回す時に集めてマージして、つながりをみるの。会話がこう、分岐していると、長い方を優先して。で、消せば。

A 単純にマージすれば良いってさ、どういうのを思い浮かべてるの？

B (笑)

C んーとー、これをこうくっつけて、時系列順につなげて。

A できるだろうね。

C で、tweet の ID(筆者注:tweet 毎に重複しない ID がつけられている) が被っているかを見て、もし被っていたらどっちかを残して。それできんの？

A (メモを取り始める)

C (引き続き身振りを入れつつ話す)tweet の ID があるよね。で、tweet の固有の ID がある。あとは、被っているかどうかを時系列に見る。で、会話を分岐して行ったのを見て…

B それなりに、時間はかかるかも。

全 (結局まとまりそうにない)

5.6.5 被験者毎の行動

先の SYMLOG、アンケートおよび会話内容をもとに、被験者毎の行動をまとめる。

■被験者 A 前半と後半で冷静さの観点で評価に大きな変化が発生している。後半に入り、プログラムが画面と結合する際に適切に結合が進まない状況が発生した際、本人の想像を超えてスムーズに行かないいらだちが垣間見えた。それが、冷静さを失わせてしまう結果となっている。また、ビデオを評価するにあたって、後半に入り急速に顔色、特に髪型の乱れが顕著に発生し、冷静さを失って行く姿を垣間みる事が出来た。

■被験者 B　画面の準備作業はワークショップ前にほぼ完了しており、当日はプログラムに対して微調整を施すのみであった。その微調整に対して、被験者 A と積極的に対話を重ね協力していた。また、積極的にリーダーシップをとる事はなかったが、状況に応じて冷静に行動する事を忘れずに実行し、徐々にリーダーシップを発揮するようになっていた。しかし、最後には被験者 A の冷静さを失って行く態度に、少し萎縮し始めている所も伺えた。

■被験者 C　当初はリーダーシップを発揮し、チーム全体の方向を整理しながら作業を進める事が出来ていた。また、メンバー同士で友好的に接するよう、コーヒーをつぐ等の気配りをする様を垣間みた。しかし、被験者 A が冷静さを失う中で自分自身がどのような貢献をして行けば良いかわからなくなり、行動が徐々に抑制されて行く様子を確認する事が出来た。

5.7　まとめ

本章では、第 1 回実験の内容について述べた。

実験中、進捗に影響する問題が発生した。α 大学チームはプログラミングに関するテクニカルスキルは充分にあった模様だが、ヒューマンスキルとコミュニケーションに関する部分で問題が発生している。

1. 進捗確認のタイミングが遅かった
2. プログラム実装部分の擦り合わせが少なかった

1 つ目の問題点は、作業を進める中で、作業の進捗確認が作業後半に入ってからとなってしまっていた。そのため、プログラムと画面の結合作業に滞りが出ている事が表面化するのが遅れ、結果後の作業が遅延した。これは、スケジュール通りに作業が進んでいるかの確認をお互いで逐次確認し、現状を良しとする合意がとれていなかった事が原因の一つであると考えられる。

2 つ目の問題点は、プログラムと画面を結合する際にうまく擦り合わせが出来ず、プログラムの開発に遅延が生じてしまった事である。特に、作業後半で 2 つの事象が発生した。まず、HTML テンプレートに埋め込む際のロジックの合意不足である。ワークショップ中、被験者 A,B との間で実装方法の合意がとれていなかった。続いて、変数名の連絡不足である。変数名を定義していたものの、開発中に変更した際の命名方法の合意が適切にとれていなかった。これは、画面とプログラムを結合する時に事前に仕様を詰め切れていなかったことが原因であると考えられる。なぜそのような状況に至ったかを被験者 A,B に直接確認をとった所、自分自身が考えている事は相手も同じ水準で問題を理解しているはずだと思っていたとの事であった。

ヒューマンスキルは、第 2 回実験で教育内容の改善を要する箇所である。

実験結果を基にした考察は、後述する 7 章に記載している。

第 6 章

第 2 回 実験

6.1 はじめに

5 章にて説明した第 1 回の実験を踏まえ、コミュニケーションスキルを養っていくためには「問題の理解度合いの齟齬」を埋めて行く事が必要であることがわかった。そこで、事前にコミュニケーションに関する指導を実施する事とした。

本章では、6.2 節において実験の概要を説明する。6.3 節において、被験者のプロフィールを記載する。事前指導の状況は、6.4 節に記載した。実験中の状況は、6.5 節では 1〜3 回目の納品まで、6.6 節では 7 日目のワークショップについて述べる。実験結果は 6.7 節に、実験の中で発生した問題は 6.8 節にて記載した。考察は、5 章で述べている第 1 回実験との比較を含め、7 章において述べる。

6.2 実験実施概要

第 2 回は、β 大学の学生 3 名 (以下、β 大学チーム) に対して実験を行った。実験期間は、2012/10/14 〜 2012/10/20 の 7 日間である。開催場所は、6 日目までは各自の自宅または大学、7 日目は筑波大学 東京キャンパス 623 号室にて実施した。

実験のスケジュールを、図 6.1 に示す。

2011/10/03 に、β 大学の学生および指導教授に対して、実験内容のガイダンスを実施した。場所は被験者 A が所属する研究室を借りて実施した。その際、次の事項を説明した。

1. 全体像の説明
2. 倫理委員会にて規定された事項の説明
3. 事前指導
4. システム要件 (背景・機能要件・非機能要件)
5. 作業工程の説明
6. スケジュール

タスク名	ID	開始日	終了日	期間	2012年 10月 14日 14 15 16 17 18 19 20
コンセプト作成	1	2012/10/14	2012/10/14	1d	
外部設計	2	2012/10/14	2012/10/15	2d	
納品 第1回	3	2012/10/15	2012/10/15	0d	
内部設計	4	2012/10/16	2012/10/17	2d	
画面設計	5	2012/10/17	2012/10/17	1d	
納品 第2回	6	2012/10/17	2012/10/17	0d	
テスト計画	7	2012/10/18	2012/10/19	2d	
リソース制作	8	2012/10/18	2012/10/19	2d	
(テストプログラム試作)	9	2012/10/18	2012/10/19	2d	
レポート提出 第3回	10	2012/10/19	2012/10/19	0d	
プログラム開発	11	2012/10/20	2012/10/20	1d	
テスト実施	12	2012/10/20	2012/10/20	1d	
最終アンケート	13	2012/10/20	2012/10/20	0d	

図 6.1　β 大学 実験スケジュール

7. 成果物の納品方法
8. 準備する物品
9. 連絡事項
10. 連絡先

付録 F にて、実際に使用した説明資料を掲載した。

6.3　被験者のプロフィール

α 大学チームの被験者は 3 人である。共通する事項は、いずれの 3 人も男性であり、被験者 B, C は研究室に所属している。そのため、α 大学チーム程ではないが日々の交流も少なからずあり、チーム内での私的な人間関係は良好である。

続いて、各個人のプロフィールを、アンケートから取得した情報を基にまとめる。

■被験者 X　プログラマ担当。これまで、講義や実験でプログラミング経験があり、チーム内で最もプログラミングスキルを持つ。尚、個人的にプログラミングを行う事は余りないとの話であった。

■被験者 Y　デザイン担当。講義や実験でプログラムを書いた事があり、かつユーザインタフェイスに関する研究にも取り組んでいる。プログラミングには自信が無いので、他のメンバーにお願いしたいと希望を出されている。

■被験者 Z　リーダー、及びテスト担当。当初よりリーダーを志望し、チームメンバー募集にあたっても積極的にメンバー集めを行っていた。講義や実験でプログラミング経験があり、かつ他のメンバーよりも広い範囲で技術の見聞を持っていることがわかっている。

6.4　事前指導

6.4.1　事前指導方法の検討

先の 6.1 節で述べたが、第 2 回実験から被験者に対して事前指導を実施する事とした。教育方法は、2 つの方法を検討した。

1. 小規模なワークショップを実施
2. 工程毎に必要となるコミュニケーションを学ぶ

1 つ目の方法は、30 分程度のワークショップを通じて、同じ課題に対して自分自身の問題意識と他人のそれとは違う事を知り、最終的にはすりあわせていく事で 1 つの結論に到達するように教育してゆくものである。

2 つ目の方法は、合意する内容は工程毎に違う事を知る事で、よりスムーズに開発を進められるようにしようとするものである。ただ、本方法はソフトウェア工学の工程管理の教育と重なる事と、「問題の理解度合いの齟齬」を埋める事は難しい。

以上から、小規模なワークショップの実施を決定した。

6.4.2　事前指導時の考慮点

事前指導方法として選択したワークショップのために、レジュメを作成した。作成にあたり、次の 3 点を考慮した。

1. 想像力を働かせる
2. ソフトウェア開発が「失敗する事がある」事を知ってもらう
3. 情報科に通う学生が身近に感じられる

1 つ目の考慮点は、自分自身の問題意識と他人のそれとは違うことを想像する行為を促した事である。成果物を通じてコミュニケーションをはかるにあたり、何を理解し合わなければならないのか、そして理解を通じて目的を完遂できるにはどのようにすれば良いのか、個々人が考える事から始まるからである。

2 つ目の考慮点は、「学校」と「業務」は違うことをはっきりさせ、今回の実験の意義を理解するきっかけとしたことである。学生の多くは、講義の単位を取る事に注力こそしているが、評価はそれほどこだわりを持っていない場合がある。しかし、業務でのソフトウェア開発は、顧客の要求通りに完成させる事は最低限であり、顧客の想像を上回る価値

表 6.1　β 大学 事前指導のスケジュール

番号	時間	作業内容
-	19:00〜19:20	(全体像の説明・倫理委員会規定の説明)
1	19:20〜10:35	自分自身で列挙する
2	19:35〜19:40	全員の情報を集めて共通部分とそうではない部分を分類する
3	19:40〜19:47	共通しなかった部分から 1 つを選び選択理由を考える
4	19:47〜19:50	選択した項目についてプレゼンテーションを行う
-	19:50〜21:00	(実験に関する説明)

を提供する事が大切になる。

3 つ目の考慮点は、社会人として多様な人生経験を獲得する以前の学生に対して、想像力がより働きやすいきっかけが必要だったためである。レジュメでは、自分自身の問題意識と他人のそれとは違う例として、プログラムの変数定義が引き起こす問題を取り上げている。これは、情報科の学生であれば、誰しもがプログラムをコーディングする経験を持っていること、そして変数名が 1 字違うだけでプログラムが動かないという点に着目したためである。

ワークショップの演習では、先に挙げた 3 点の考慮を踏まえ「ブラウザの機能を挙げる」演習を実施した。インターネットサイトを閲覧する「ブラウザ」は、情報系の学生であれば普段慣れ親しんだソフトウェアの 1 つである。そのソフトウェアであっても、必要となる機能の認識は各々が違うことを知り、認識をすりあわせていく過程を通じ、コミュニケーションの重要性を理解できるようにした。

作成した内容は、付録 G の通りである。

6.4.3　事前指導の状況

事前指導は、実験の詳細な説明に入る前に実施した。これは、実験の意図を理解してから事前指導を実施してしまうと、コミュニケーションのプロセスと重要性を理解する意識をそいでしまう可能性があったためである。説明の流れは、表 6.1 に示す。

演習のはじめ、ソフトウェア開発が失敗する事がある話に「そういう事があるのか」と少し驚いていた。その後は、事前指導の目的である「自分自身の問題意識と他人のそれとは違う」点を少しずつ理解し始め、演習を通じてその事実を受け止めていたようである。

6.5　1 回目〜3 回目 納品物の確認

1 回目から 3 回目まで、予定通り成果物の納品を受けた。

担当割りは、前回と同様、ガイダンス時に本人たちの立候補でスムーズに決定した。

修正を伴う再提出が 2 回発生しているが、これは納品時点で作成内容の不備 (誤字やス

表 6.2　β 大学 ワークショップのスケジュール

時刻	作業内容
11:10〜12:30	技術調査
12:20〜12:30	調査状況 共有
12:30〜13:00	技術調査・プログラミング
13:00〜13:10	作業進捗 共有
13:10〜13:40	昼食・技術調査・プログラミング
13:40〜13:50	調査状況 共有
13:50〜15:00	技術調査・プログラミング
15:00〜15:15	作業進捗 共有
15:15〜16:30	技術調査・プログラミング
16:30〜16:55	作業進捗 共有
16:55〜17:00	問題点の整理
17:00〜18:20	本日の作業を踏まえ設計資料の更新
18:20〜18:30	作業結果確認 ソフトウェアの内容のプレゼンテーション

キャンした情報の不鮮明) であり、内容そのものの問題ではなかった。

6.6　7 日目 ワークショップの状況

6.6.1　全体の流れ

7 日目のワークショップは、プログラムを完成させる事を目的とし、被験者に作業に取り組んでもらった。実際の作業タイムスケジュールは、表 6.2 の通りとなっている。

当初、終了は 18:00 であったが、作業の遅延が発生したため、18:30 まで延長して実施した。

作業工程であるが、α 大学チームの図 5.1 に比べ、作業開始当初から密に進捗状況の共有が行われている。これは、当初実施した事前指導を通じて、コミュニケーションを行う重要性が理解され実践されている一つの現れと考えられる。

6.7　実験結果

6.7.1　ソフトウェアのスクリーンショット

β 大学チームが開発したソフトウェアのスクリーンショットを、図 6.2 に示す。

図 6.2　β 大学チームが開発したソフトウェアのスクリーンショット

6.7.2　アンケート結果

事前及び事後アンケートから、被験者の変化を分析する。評価にあたっては、3.3 節をもとに行う。

まず、表 6.3 に、「ケースで目標としている人間構造の理解の水準を達成しているか」、即ちヒューマンスキル部分についての内容を記述する。アンケートの書式は、事前は付録 B、事後は付録 E に掲載した。

続いて、表 6.4 に、「ケース内に織り込まれている要素技術を理解」(テクニカルスキル) 部分についての内容を記述する。「プログラミング能力」で○をつけた項目を、上から順に 1 番〜6 番とした。また、「ソフトウェア開発能力」は、上から 1 番〜15 番とした。

続いて、表 6.5 に、「システムのデザインを行う事が出来る力が育む」(コンセプチュアルスキル) 部分についての内容を記述する。

6.7.3　SYMLOG を用いた小集団構造把握

SYMLOG を用い、β 大学チームの集団構造を分析した結果を、図 6.3 に示す。評価は、筆者が実施した。分析方法、及び描画方法は、図 5.3 と同様である。前半は主に 11:00〜14:00、後半は主に 14:00〜18:30 を指している。

*1 実際は満たせている

表 6.3 β 大学チーム 前後比較 — ヒューマンスキル

項目		事前の希望	事後の感想
自分自身の役割	X	プログラマ	わからない事が多く他のメンバーに迷惑をかけた
	Y	画面のデザイン・部品作成	当初の要求を満たしきれなかった[*1]
	Z	プログラミング・設計資料作成	計画の作成までは果たせた。リーダーとしての面目は立てた。しかし技術が足りなかった。
仲間の役割	X	プログラミングを分担してほしい	リーダーはメンバーの意見をまとめつつ、テスト仕様も問題なく仕上げていた。
	Y	プログラミングを支援してほしい	リーダーはメンバーをまとめフォローをしていた。プログラマは積極的に調査をし解決へ努力していた。
	Z	画面はお願いしたい。開発は速度を上げて望みたい。わからない事は相談に乗りたい。	プログラマは、技術的問題を解決できず役割を果たし切れていない。デザイナは予定通り作業が進み、役割は果たせていた。
自分自身の配慮	X	進捗を常に把握、質問しやすい状態にする、一緒に考える、積極的に声がけする。	当初考えていた事が実践できた。
	Y	普段のように楽しい会話を絶やさない。苦しいときも励まし合う。	当初考えていた事が実践できた。
	Z	ぴりぴりしないよう言動に気をつける。苦しいときも励まし合い、協力し合う。	当初考えていた事が実践できた。
これまでの仕事経験	X	食料品店	声がけすることで互いの状況を把握できた。
	Y	コンビニ, 開発のインターン	活かせる場面はなかった
	Z	なし	なし
チーム内での立ち位置	X	メンバー	メンバー フォローされてばかりでリーダーを支援できなかった
	Y	メンバー	メンバー リーダーをフォローし、互いに連絡を取り合えた。
	Z	リーダー	リーダー 意思疎通は役割を果たせた。ただ、技術と意思決定部分は率先できなかった。

表 6.4　β大学チーム 前後比較 — テクニカルスキル

項目		事前の状況	事後に獲得
プログラミング能力	X	2	1,2
	Y	1,2	1,2
	Z	1,2	1,2,4
ソフトウェア開発能力 (事前:”習った”以上) (事後:”手伝った”以上)	X	1,9	1,2,3,4,5,7,8,9,11,15
	Y	7,9	1,2,3,4,5,7,9
	Z	4,5,7,9	1,2,3,4,5,7,9,10,11
工程管理	X	ひな形を作り、動きを考えながら機能を追加。	設計の段階で話し合っておく事で、作業の流れがはっきりし、マンネリ化しづらくなる。緊張感も高まる。
	Y	大まかな仕様→アルゴリズム→プログラミング→テスト	企画から考える所、及び担当を分けて作業する点が違う。
	Z	要求を列挙→作業順序を考える→作業毎に関数を書く→テストとデバッグを繰り返す	外部設計の不備が問題を明確にしてしまう。メンバ同士の進捗を正確に把握する必要がある。密なコミュニケーションが欠かせない。意見がまとまれば一人でやるよりも圧倒的に早く開発できる。

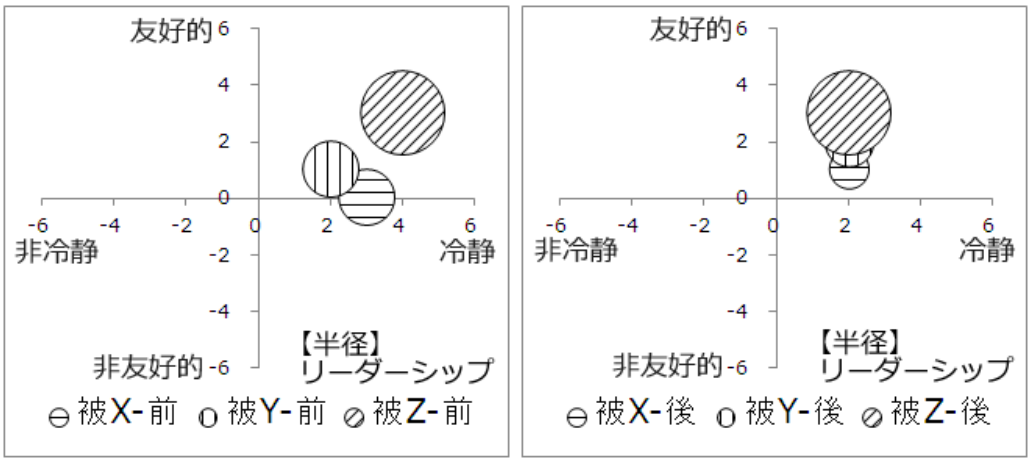


図 6.3　β大学 ワークショップ内での構造評定 (前半:左 後半:右)

表 6.5 β 大学チーム — コンセプチュアルスキル

項目		事前の想定	事後の理解
問題解決(技術)	X	Google で検索, 技術書を読む	Google で検索
	Y	Google で検索, 友人に相談, 本を調べる	クライアント・ホスト間のやり取りをイメージして何が起こるかを考えた
	Z	友達に聞く, 本で調べる, オンラインリファレンスを参照する	HTTP クライアント・サーバの構造を理解していなかったため問題を理解できなかった
問題解決(担当者間)	X	-	互いの成果物を連携させる所までたどり着けなかった
	Y	-	担当者間の連携以前に実装前に基礎知識を固めるべきだった
	Z	-	各自がどの程度の知識を持っているか正確に把握できなかった 特にプログラマの動きが見えづらかった
普段の情報収集	X	Google で検索, 論文を読む	特になし
	Y	Googlede 検索, 技術書を読む	特になし
	Z	ニュースサイトを読む	node.js の環境構築方法, プログラムの組み方
実践を通じて得た知識・技術	X	-	JavaScript のコーディング力
	Y	-	CSS と JavaScript の連携
	Z	-	企画段階のやり取り方法, 自分自身の能力の把握
実践	X	特になし	-
	Y	特になし	-
	Z	HSP でゲームを開発	-
発信	X	特になし	-
	Y	特になし	-
	Z	特になし	-

6.7.4　会話内容

ビデオより、被験者のコミュニケーションスキルを垣間みる事ができる部分を引用し、前半・後半に分けて以下にまとめる。被験者 X, Y, Z はそれぞれアルファベット 1 字に略して記載する。「全」とは、全員を示す。

■前半　11:15 頃

全　(今日のスケジュールの調整を始める)
Z　(関西弁で) 先にそのー POST GET の部分の調査、プログラムの調査を一人ぐらい割り当てて、API の調査を…一人でできます？
X　(笑) グッフフ
Z　GET はほんまにどういう仕組みで動いてるかっていう。
X　GET でクライアントとサーバがつながっているって言う？
Z　GET 命令ってのがサーバにくるんだけれど、それがどういう命令なのかサーバ側で GET メソッドでサーバが理解するって言う。
X　GET 命令をサーバで理解する…
Z　多分そうだと思うんです。そこら辺も含めてちょっと。
Y　(画面の設計を見ながら) あのー、ログオンの何タラかんたらみたいな。で、今開いているんだけれど…
Z　あれ、開けないなぁ。
Y　…、で時間割は。
X　(白板を書き始める)
全　(A が白板を書く間 しばらく目立った会話はない)
Z　(白板がだいたい書き終わった事を見計らい)GET ができたら…そっから 15 時くらいでなくて？ 15:30 にしようか。で、ブラウザから GET 情報を受け取ってそれを表示する。
X　13:30 まで昼休み。
Z　そこら辺は随時。で、そっからは 15 時まで GET の API で…。で、から、17 時までに受け取った情報をブラウザで表示する。
X　15 時くらいまで HTML を作っておかないと。
X　(白板を書き終える)
Y,Z　ありがとうございます！

■後半　17:50 頃

全　(今日の作業を踏まえ画面設計資料の更新を行っている。作業の最後に、読み合わ

せて問題ないかを確かめている。)

Z　2 画面に分けてまして (筆者注:メニューの枠とデータを表示する枠が左右に分かれている)。

Y　(Z を向いて)2 画面のトップは…「トップに戻る」で 1(筆者注:画面 ID を振っている) に戻るってのが、違和感がある。ログアウトを飛ばして 1 に戻るって、何か変だよね。

Z　(メモを取りながら) んー、でも勝手にログアウトしてくれた方が自然じゃない？

X　(天を仰ぎながら) トップページが先ず必要かどうかって話が。

Z　(関西弁で) それあると思うね。イランと思うね、実際。

Y　認証画面に戻れば良いんじゃないかって言う。

Z　あーそれね。でもそれやるとなんかさ、トップページはトップページって、なんかよくわからん事になるから、ワンクッション挟んだ方が良いかなって。体裁を保つ。

Y　こいつを何とかしないとなー

X　(苦笑) あー

X　(ペンで画面設計書を指しながら) これをさ、こうすれば。

Y　(納得) あー！そうだそうだそうだ、そうね。つまりね、ここをトップと書いたのが間違いだったんだ。

Z　というと？

Y　ココにしよう、2 番。

Z　ほー。

Y　なんだろうな、なんだろう。

Z　言いたい事はわかる。そこまで戻れたら便利ってことだよね。

Z　(画面遷移図を書き直す)

Y　あー、そうそうそうそう。

X　(画面遷移図を見ながら) ここをこうして、こうすれば。

Y　なるほどね〜！あまり書き換えなくていいね。

X　(画面遷移図を指しながら) 戻るボタンで、こうで、こう、こう、こう。ログアウトで、こう、こう、こう。

Y　これのほうがいいかもしれないね。

X　そうだね。どうなんでしょうね。

全　(笑)

X　(画面遷移図に手を入れる)

Y　トップページの戻りはいらない。消す。

X　んじゃ、こうで、こうで、こうで…

X　(Z の書く画面設計図を見ながら)2 画面消さなくて大丈夫。

Z　あー、よかったー。

Y　Excel 使えば良かったね (筆者注:Excel を方眼紙のように使って図を描画する)。
Z　あーそうやった。
全　(くすくす笑う)
全　(だいたい話がまとまった。A と C が設計図を改めて修正している。)

6.7.5　被験者毎の行動

先の SYMLOG、アンケートおよび会話内容をもとに、被験者毎の行動をまとめる。

■被験者 X　被験者 Z の調査について積極的に耳を傾け、自身のプログラム開発に活かそうとする姿勢が随所に出ており、チームワークを大切にする姿勢を垣間みる事が出来た。一方、相対的ではあるが終始受け身の姿勢になりがちであった。自ら意見を述べる事は余りなく、指示のもとで着実に自分自身の作業を進める状況であった。

■被験者 Y　被験者 X,Z の内容を理解し、自分が今何をすべきかを考えながら行動しようとする姿勢が特徴的だった。しかし、やるべき事に対して自分自身の技術スキルが追いついていない事に、わだかまりを感じていたことをアンケートから確認する事が出来た。

■被験者 Z　当初よりリーダシップを発揮していた彼は、ワークショップ中もその姿勢を遺憾なく発揮していた。自分自身の作業はもとより、技術調査にあたっては被験者 A では手が回らない部分について分担して調査を行い、twitter API の構造について知見を深めていた。そして、深めた知見を滞りなく共有する事で、作業を進めるにあたってどのような知識が必要かをチーム全体で共有できていた。

6.8　まとめ

本章では、第 2 回実験の内容について述べた。
実験中、進捗に影響する問題が発生した。

1. 技術調査の準備が不足していた
2. 問題に対して全く違う解決法を選択していた

1 つ目の問題点は、技術の点で課題になるのが twitter API と node.js のプログラミングに関する 2 点であった。このことは、実験開始当初から β 大学チーム一同が問題になるとも認識していた事を、ワークショップ中の会話からも確認する事が出来ている。twitter API は被験者 C が調査を進めていたものの、node.js に関する調査はチーム全体で余り進んではいなかった。特に、node.js を用いて Web アプリケーションを実装する際に必要となる、HTTP の GET, POST メソッドの動きを初めとした基礎知識が不足していた事が大きな問題となった。

2 つ目の問題点は、後半、先の node.js による Web アプリケーションの実装に関する調査が行き詰まった際に発生している。内容は、実装方法を当初計画していたサーバサイドを中心とした方法から、クライアントサイド中心の実装方法に切り替えようとしていたことである。HTTP で用いられる POST, GET メソッドの挙動が理解ができないからといって、残り 3 時間の状況でアーキテクチャを切り替える設計変更が功を奏すとは考えにくい。また、クライアントサイドでの実装は jsonp を初めとした別の実装の問題が浮上する。これは、選択を急に切り替えた事が問題なのではなく、うまく行かないだけの理由で別の選択をとる「回避」行動に出たと考えられる。

実験結果を基にした考察は、後述する 7 章に記載している。

第 7 章

考察

7.1 はじめに

本章では、5 章で述べた第 1 回実験、及び 6 章で述べた第 2 回実験の結果を踏まえ、考察を行う。

まず、7.2 節において第 1 回実験について実験の前後での比較を行い、3 つのコアスキルが獲得できたかの確認を行う。続いて、7.3 節において第 2 回実験についても同様に比較を実施する。最後に、7.4 節において第 1 回実験と第 2 回実験の結果を比較する。

なお、被験者 A, B, C および X, Y, Z は「被験者」を省略して表記する。

7.2 第 1 回実験 — α大学チーム

7.2.1 ヒューマンスキル

5 章の表 5.2 を基に分析を実施する。被験者 3 人全員に共通する点は、3 点ある。

1. 会話が少なかった。
2. 役割分担を自分自身が理解し、かつ他者からも理解されていた。
3. 自分自身の役割を全うしきれなかったと感じていた。

1 については、A は「あまり他メンバーと交流がなかった」、B は「作業をしていていつものように話を振れなかった」、また C は「プログラム結合時に配慮できなかった」とアンケートに記述していた。被験者 3 名は同じ大学の同じ研究室に所属し、普段から公私ともに交流が深い関係である。しかし、本研究での実験ではその関係を踏まえた上でコミュニケーションが滞っていてしまった。これは、5.6.4 節に掲載した会話からも確認できる。

特に、A については、SYMLOG によると 7 日目の実験前後で有為な差が見られた。実験前半では冷静に作業を進めていたものの、後半で冷静さを失っていた。このことが、他のチームメンバーに対し心理的に悪影響を与え、より会話が難しい状況を作り出したのではないかと考えられる。また、C が述べていた「プログラム結合時」に必要となるコミュ

ニケーションがどのようなものかが、被験者に理解されていなかった事も問題の 1 つと考えられる。

2 については、アンケートより、実験開始時に設定した役割分担を各自が理解し、結合までの業務は問題なく全うしていた。また、メンバーからも設定された役割分担を理解されていたことを読み取る事が出来た。

3 については、役割を理解はしていたものの、プログラム結合時に発生した問題をチーム内で解決させられる事が出来ず、各自が責任を感じていることが確認できた。これは、結合などのチームメンバーの間でインタラクションを行いながら作業を進める段階で問題が発生していると、被験者が認識していると考えられる。特に、被験者 B の「自分が考えている事と相手が考えている事に相違があるとプログラムは結合できない。」というコメントがその象徴である。

以上より、コミュニケーションがあまり行われておらず、ヒューマンスキルがあまり獲得できていない事を確認できた。

7.2.2　テクニカルスキル

5 章の表 5.3 を基に分析を実施する。被験者 3 人全員に共通する点は、2 点ある。

1. 幅広い工程に携わる経験を積む事が出来た。
2. 設計を適切に行う重要性を感じていた。

1 については、被験者らがそれまで経験した事がない、要求定義〜設計〜プログラミングといった幅広い工程に携わる事が出来たことが確認できる。大学の講義や実験では「1 人」かつ「特定の工程」、例えばあらかじめ詳細な仕様が定まった課題をもとにプログラムを開発していたとのことであり、より業務の開発現場に近い経験が出来たものと考えられる。

2 については、適切な設計を行う事で、プログラムを書く負担が軽減できると理解している事が読み取れる。プログラムを書く作業さえやれば最終製品が作れると考えがちだが、どのように組み上げるのか事前に理解しなければ失敗してしまう事を、本実験を通じて理解しているようである。

以上より、テクニカルスキルはこれまでのプログラミング中心の狭い範囲から、本実験を通じて工程全体を俯瞰する知識を獲得した事を確認した。

7.2.3　コンセプチュアルスキル

5 章の表 5.4 を基に分析を実施する。

被験者 3 人全員に共通する点は、情報収集を行う手段を自ら開拓する力を獲得したことが挙げられる。これまでは、主に「Google で検索」と「友人に相談」に終始していた。今

回、外部サービスである twitter の仕様を知るにあたって、API ドキュメントに関する資料を調達し、自分自身でテストプログラムを書く事で各自が理解を深めていた。

しかし、今回の実験を通じて、twitter の使い方を大きく変えるような目新しい機能の実装や提案は行われていない。

従って、本実験を通じてコンセプチュアルスキルが向上する、明確な結果は得られなかった。

7.2.4 全般

実験を通じ、α 大学チームの特徴を 3 つのコアスキル毎に示す。その上で、獲得したスキルが何かを確認する。

- ヒューマンスキル — 作業後にコミュニケーションの必要性を理解した。
- テクニカルスキル — 当初から順序立てて設計を行う重要性を理解した。
- コンセプチュアルスキル — 特に獲得できていなかった。

ヒューマンスキルについては、集団でのソフトウェア開発において設計書を通じたコミュニケーションの必要性を、実験後に被験者各人が認識している所まで行き着いていた。開発作業を進める際、事後に別のチームメンバーと結合作業を行わなければならない箇所について事前に充分に調整がとれていなかった事をメンバー全員が理解していた。アンケートにおいても「自分が考えている事と相手が考えている事に相違があるとプログラムは結合できない」とコメントがあった。そのため、今後の研究・業務を通じた活動や、第 2 回実験で実施した事前指導を実施していれば、学習を深められたと考えられる。

テクニカルスキルについては、ソフトウェア開発の工程を通じて理解したと考えられる。被験者は、要求をもとに設計を段階的に詳細化し、プログラム開発は設計の中で決まった事項を実装していく作業に過ぎないことを認識し理解を深めていた。これは、アンケートにおいて「初期設計が適切に仕上がっていればプログラミングはただの作業」の記載に代表される。これは、ウォーターフォール型開発モデルがコンセプトとして掲げている「段階的に詳細化する [7]」点にも合致している。

コンセプチュアルスキルについては、アンケート結果より、本実験を通じてソフトウェア開発に必要なスキルを獲得できていないと確認した。

7.3 第 2 回実験 — β 大学チーム

7.3.1 ヒューマンスキル

6 章の表 6.3 を基に分析を実施する。被験者 3 人全員に共通する点は、3 点ある。

1. 継続的なコミュニケーションを通じて互いに理解を深め合っていた。

2. 役割分担を自分自身が理解し、かつ他者からも理解されていた。
3. 問題が発生している際の分担の再分配が出来ていた。

1 については、表 6.1 やアンケートの「コミュニケーションは円滑に行えた」というコメント、および 6.7.4 節の会話内容から、円滑にコミュニケーションが実施できた事を確認した。また、6.7.4 節に掲載した後半の会話を確認すると、相手の理解を深めようと自分自身が考えながら理解の擦り合わせを行い、ソフトウェアを完成させる目的に向かって活動している事がわかる。これは、第 2 回実験のみで実施した「事前指導」の効果が出ているものと考える。

2 については、第 1 回実験同様、アンケートから実験開始時に設定した役割分担を各自が理解し、結合までの業務は問題 なく全うしていた事がわかった。また、メンバーからも設定された役割分担を理解されていたことを読み取れた。

3 については、ビデオや表 6.1 の作業スケジュールより、X が技術的な問題に遭遇し作業進捗が滞り始めた際、リーダーである Z が X の調査内容を確認し、チームメンバー全員に調査作業を再分配していた。状況に応じて、役割分担を変更する事でより円滑にチーム運営が行えるよう配慮できていた点は特筆できる。SYMLOG からも、各メンバーが終始モティベーションを維持し切れた事も確認できた。また、リーダーである C の音頭を基に行動しており、チームマネジメントを積極的に行っている点も認められた。

以上より、事前指導と本実験を通じてコミュニケーションを通じたコミュニケーションを形成するスキルが獲得でき始めており、ヒューマンスキルの獲得が進んでいることを確認した。

7.3.2 テクニカルスキル

6 章の表 6.4 を基に分析を実施する。被験者 3 人全員に共通する点は、2 点ある。

1. 幅広い工程に携わる事が出来た。
2. 事前の技術調査が不足していた。

1 については、第 1 回実験と同様、これまでプログラム中心だったところに、要求定義〜設計〜プログラミングを通した幅広い工程に携わる事が出来たことが確認できる。被験者に直接ヒアリングした所、講義ではチームを組んで開発を行うことはなく、一人である程度決まった仕様に沿ってプログラムを実装するのが主であったとの事であった。

2 については、6.8 節でも述べたが、Web アプリケーション開発の基礎知識である GET, POST メソッドの動きについて理解が深まっておらず、プログラムの実装が行えていなかった。口頭で、1〜6 日目の作業時に事前調査を行っていたかの旨を確認した所、特に行っていなかったとの事であった。また、実装に利用する node.js, twitter API についても同様に理解が深まっていなかった。

第 1 回実験に参加した α 大学チームの A は、もともと個人で Web アプリケーション開発を行った経験があった一方、X には個人での Web アプリケーション開発の経験がなかった。しかし、A は twitter API の知識については全くない状態でありながらも、1〜6 日目の設計作業中に事前調査を実施し、知識を獲得した上で 7 日目のワークショップに臨んでいた。仮に X に Web アプリケーション開発の経験があったとしても、twitter API の問題が解消したとは考えにくい。従って、β 大学チームの事前調査の不備が目立っている。

以上より、本実験がソフトウェア開発の工程全体を知る機会になったものの、実験を遂行するための知識・調査が不足しており、テクニカルスキルが獲得できたとは言えない状況である。

7.3.3 コンセプチュアルスキル

6 章の表 6.5 を基に分析を実施する。

被験者 3 人全員に共通する点は、本実験を通じて明確に獲得できたスキルは認められなかった。従って、本実験を通じてコンセプチュアルスキルが向上した、明確な結果は得られなかった。

7.3.4 全般

実験を通じ、β 大学チームの特徴を 3 つのコアスキル毎に示す。その上で、獲得したスキルが何かを確認する。

- ヒューマンスキル — コミュニケーションを通じて理解し協力し合った。
- テクニカルスキル — 知識・調査不足が発生し作業が滞りがちだった。
- コンセプチュアルスキル — 特に獲得できていない。

ヒューマンスキルについては、積極的かつ定期的なコミュニケーションを実施し、チーム内で発生している問題を互いに理解し協力するスキルを獲得した事を確認した。ここから、本実験を通じてソフトウェア開発に必要なヒューマンスキルは獲得できていると考える。

目立ったのは、チーム全体でコミュニケーションを積極的に行っていた点である。自分自身が持っている情報を相手に納得できるまで伝える事、そして自分自身の現状を包み隠さず伝える事を積極的に実施していた。そして、作業開始当初より積極的に実施している事により、リーダーは問題点を逐次把握する事が出来る状態にあった。また、メンバーも同様にチームにどのような問題が発生しているかを理解し、自分が何をすべきかを冷静に思考できる状況を創り出す事が出来ていた。

テクニカルスキルについては、作業の前提となる知識・調査が不足しており、プログラムの結合作業へたどり着く事が出来ていなかった。従って、本実験を通じてテクニカルスキルは獲得に至っていないと判断する。

なお、要求仕様の一致については、α大学チーム同様、β大学チームでも達成されていた。なお、開発するシステム全体の分量は、機能設計書ベースではβ大学チームはα大学チームに比べて半分であった。これは、チームメンバー全体で書き上げる事が出来る分量を考えた時、当初示された要件を満たしつつ出来る限り小さな工数で達成させようとする意図があったとのことであった。

コンセプチュアルスキルは、アンケート結果より、本実験を通じてソフトウェ ア開発に必要なスキルは獲得できていないと確認した。

7.4　第 1 回と第 2 回実験の比較

2 回の実験を、「ヒューマンスキル」と「テクニカルスキル」の 2 点に分け、図 7.1 に示す。

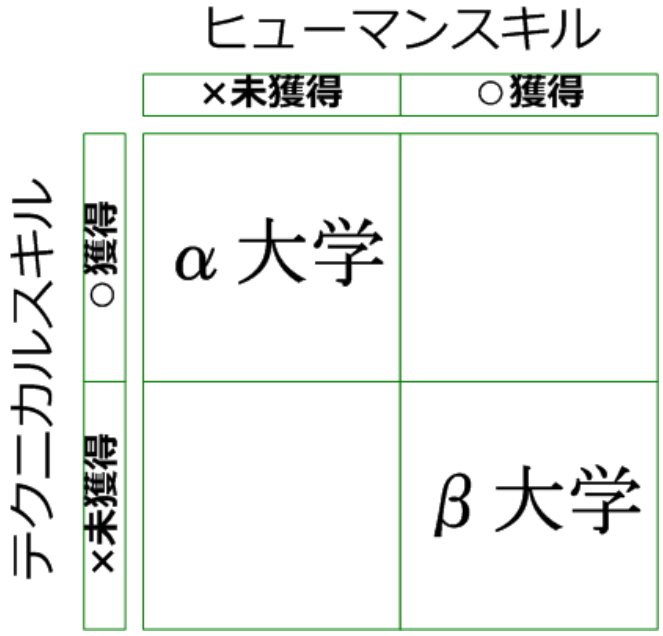


図 7.1　第 1 回と第 2 回実験の比較表

今回目指すべきであったのは、図 7.1 における右上の象限、即ち「ヒューマンスキル」を獲得しながら「テクニカルスキル」を獲得する事であった。しかし、本研究ではどちらも満たす結果を得る事は出来なかった。

原因は、以下の 2 点が考えられる。

1. α大学チームは事前にコミュニケーションに関する教育を実施していなかった
2. β大学チームはケースの中で利用する技術を理解していなかった

1 については、β 大学チームの方が、α 大学チームよりコミュニケーションスキルを獲得している事が認められた。β 大学チームは、事前指導より、コミュニケーションを通じて互いを理解し合い、目的を達成するために協力し合いながら実験を遂行することで、コミュニケーションの必要性を理解し実践する事ができた。一方、α 大学チームは事後に自分自身の失敗を通じてコミュニケーションの必要性を理解できたものの、実践には至らなかった。

2 については、第 2 回実験に参加した β 大学チームには、ケースを実践するためのテクニカルスキル、特にプログラミングスキルが不足していた。これは、本実験を行う前に、実験に必要なテクニカルスキルを学ぶ機会を設ける事で解決が必要だと考える。

7.5 まとめ

本章では、第 1 回・第 2 回の実験結果を踏まえて考察を実施した。

7.2 節では、第 1 回実験の結果を踏まえて考察を行った所、テクニカルスキルについては問題なく獲得できたものの、ヒューマンスキルの獲得で問題が発生した。特に、実装に関するコミュニケーションが不充分で、プログラムを結合させる事が出来なかった。しかし、事後に被験者全員がその問題を認識していた。そのため、今後同様の問題が起きた場合は、対処できる可能性を残していることも確認できた。

7.3 節では、第 2 回実験の結果を踏まえて考察を行った所、事前指導を通じてヒューマンスキルの獲得は問題なく行われていた事を獲得した。これは、事前指導の効果があったと考えられる。しかし、テクニカルスキルはケースを実施できる程に獲得できておらず、開発途中で問題が多発する結果となってしまった。

7.4 節では、第 1 回・第 2 回実験の比較を行った。その結果、ヒューマンスキル・テクニカルスキル両面を満たして獲得したグループがいなかった事がわかった。ケースを実施するには、テクニカルスキルの中のプログラミングスキルがケースをこなすにあたって充分にあることが前提であり、その上でコミュニケーションスキルが獲得できる状況になる事を確認した。

以上より、事前指導とケースメソッド＋ロールプレイング形式の学習によって、コミュニケーションスキルを高められる事を確認した。しかし、テクニカルスキルとコミュニケーションスキル、両方を満足した実験結果を得る事ができなかった。今後は、第 1 回実験と同程度のスキルを持っている被験者を募集し、第 2 回実験と同じ形で進める事で 2 つのスキルが同時に獲得できるかを確認する必要がある。

第 8 章

結論

企業活動としてのソフトウェア開発とは、技術者を中心としたチームを組んで一つのソフトウェアを完成させることである。その時に、コミュニケーションを通じて理解を深める事を目的に、チームメンバーの間でドキュメントやソースコードなどの媒体を通じ、インタラクションが行われる。一方で、コミュニケーションが苦手な技術者、特に新人技術者が存在し、コミュニケーションが円滑に行われない事象が発生している。コミュニケーションが滞る事はソフトウェア開発にも悪影響を及ぼしている。これは、高等教育機関でソフトウェア開発に関するコミュニケーションについての教育を受けていないことに起因している。

そこで、本研究は新人技術者の候補となる情報系の学部に所属する大学 4 年生を対象に、チームでソフトウェア開発を行う教育を試行し、効果を測定する事を目的とした。

1 章では、ソフトウェア開発におけるチームとコミュニケーションとは何かを確認した。その上で、コミュニケーションが苦手な技術者の存在とその問題点を指摘した。その上で、本研究において、高等教育機関に所属する学生に対し「チームで行い」「コミュニケーションスキルを高め」かつ「1 週間程度」の要件を満たす教育手法を試行する事を説明した。

2 章では、関連研究「ソフトウェア開発プロセス」「日本におけるソフトウェア開発とコミュニケーションの関係」「教育水準の指標」「日本における高等教育カリキュラム」および「集団構造を分析する指標」について、調査を行った。その結果、既存の教育方法では短期間でコミュニケーションスキルを高める教育が不足していることを明らかにした。また、コミュニケーションスキルを定量的に測定できる指標の存在を確認した。

3 章では、1 章と 2 章の議論や調査を基に教材の作成指針を立てた。養成するスキルを、管理者に求められる 3 つのコアスキルである「ヒューマンスキル」「テクニカルスキル」そして「コンセプチュアルスキル」をもとに設定した。また、ヒューマンスキルにあたるコミュニケーションスキルを高めるために、技術者に求められるコミュニケーションスキルについて定義した。

また、実験時に用いる教育手法をケースメソッド＋ロールプレイング形式とした。これ

は、実際に作業を行う事を通じて理解を促し、またチームを組んでワークショップを実施する事で、他者とコミュニケーションを行う行為を意識的に行わせるためである。

4 章では、3 章の教材の作成指針を基に、実験実施計画を作成した。この中で「養成するスキルの設定」「実験工程と評価基準点の設定」「評価のための準備」「被験者の募集」「ケースの作成」「実験のスケジュール」および「成果物の定義」を行った。ケースには、現在の大学 4 年生が普段利用していると思われる twitter を用いた mashup サイトを作成するものを設定した。これは、開発難易度がそれほど高くない事、また普段使っている twitter を通じたアプリケーションであることから開発に興味を持ちやすく、楽しんで実験に参加してもらう事を意図したものである。また、被験者は 2 大学からそれぞれ 3 名を集める事が出来た。

5 章では、国立α大学 情報系学科所属の大学 4 年生 3 名 (α大学チーム) に対し、4 章で作成した実験実施計画を基に実験を行った。結果、途中までの開発は順調だったものの、ソフトウェアの結合時に問題が発生した。原因は、結合作業を行う前段階の設計を行う際、チーム内でのコミュニケーションが不足していた事が確認でき、被験者たちも認識した。また、工程を通して開発を行う経験を通じて、設計の重要性を理解することができた。

6 章では、国立β大学 情報系学科所属の大学 4 年生 3 名 (β大学チーム) に対し、コミュニケーションに関する事前指導を行った後に、4 章で作成した実験実施計画をもとに実験を行った。結果、設計時や開発時のチーム内でのコミュニケーションは積極的に行われるようになった。しかし、被験者にケースに基づいたソフトウェア開発を行うためのテクニカルスキルが不足している問題が明るみになった。

7 章では、5 章と 6 章で実施した実験に関する考察を行った。その中で、α大学チームはヒューマンスキルの獲得はできなかったものの、獲得に至る糸口を掴む事はでき、テクニカルスキルについては工程を理解する形で獲得する事が出来た。一方、β大学チームは事前指導を通じ実験内でヒューマンスキルを獲得する事が出来たものの、テクニカルスキルについては問題を残す結果となった。

以上より、本研究では、新人技術者となっていく高等教育機関の学生がコミュニケーションスキルを獲得するためには、コミュニケーションに関する事前指導を踏まえて、ケースメソッド＋ロールプレイング形式の学習を行う事で、コミュニケーションスキルを高めることができる結果を得ることが出来た。

しかし、テクニカルスキルとコミュニケーションスキル、両方を満足した実験結果を得る事が出来ない課題が残った。本研究では、事前に第 1 回目に参加したα大学チームと同等のテクニカルスキル、少なくともプログラミングスキルが獲得できていなければ、ヒューマンスキルを養成することが難しい事がわかった。その点では、第 2 回実験の前にテクニカルスキルを同程度の水準まで引き上げることが必要だったと考えている。なお、コンセプチュアルスキルが獲得できた明らかなデータは得る事ができなかった。

今後、テクニカルスキルとコミュニケーションスキル双方のスキルをより確実に獲得で

きるようにするために、次の点の改善に取り組み、更に研究を進めていきたい。

1. ケースに臨むために必要なテクニカルスキル獲得状況の確認方法の作成
2. コミュニケーションスキルを獲得する手法の更なる開発
3. ソフトウェア開発のケースの更なる整備
4. コミュニケーションスキル獲得の指標の開発

1 については、先の課題を踏まえ、ケースに臨むにあたって必要となるテクニカルスキルが既に獲得できているかを確認する方法を開発する。これを踏まえ、テクニカルスキルが不足している場合は事前に個々人に必要な教育を実施する。これを通じ、ケースに臨む際にスキルが不足し作業が完遂できない可能性を低下させ、教育が適切に行える環境を整備しやすくできると考えている。

2 については、今回はソフトウェア開発の工程を全て通して実験を行った中で、特に設計工程でコミュニケーションを行うコミュニケーションが重要である事がわかってきた。その中で、ドキュメントやソースコード等の媒体を通じ、チーム内でどのようなコミュニケーションをとるべきか、より深く分析をした上でケースメソッド等に展開できるようにしたいと考えている。

3 については、今回は Web 開発に関するケースであった。これを、クライアント・サーバ形アプリケーションや、組み込み型アプリケーション等、更に展開していく余地があると考えている。

4 については、獲得したコミュニケーションについて、測定する指標を開発していく必要があると考えている。今回は SYMLOG を用いたが、チームマネジメントや経営組織論の研究から適切な指標を調査し、ソフトウェア開発に関する事柄に最適化していく事を想定している。その中で、技術者がコミュニケーションスキルを獲得したか否かの、客観的な測定が出来るようになるはずである。

謝辞

本論文は、多くの方のご指導、ご助言、ならびにご協力いただく事でまとめることができました。ここに紹介できない方々を含め、全ての方々に謹んで感謝の意を表します。

筑波大学 大学院 ビジネス科学研究科、佐野 享子 准教授からは教育学の観点、中谷 多哉子 准教授にはソフトウェア工学の観点から、本論文と研究について親切なご指導とご助言をいただきました。厚く感謝致します。

今回、実験参加を承諾・協力いただいたα大学・β大学の学生の皆様、そして指導教授の皆様に、深く感謝申し上げます。皆様があって、本研究は成立しました。また、学生の皆様が、本研究で取り組んだ実験を通じて経験した事が、今後の研究や社会人として技術者になった際に活躍する際の糧になれば、幸いです。

筑波大学 大学院 ビジネス科学研究科、吉武 博道 教授からは、これまで勤務されていた新日鉄におけるソフトウェア開発の歴史を基に、ソフトウェア開発で必要なコミュニケーションや組織マネジメントについてアドバイスを頂戴しました。ありがとうございました。

同大学院 大木 敦雄 准教授には、ソフトウェア開発を通じた技術者同士のインタラクションの内容について、どのように分析を進めるかのアドバイスを頂戴しました。ありがとうございました。

筑波大学 システム情報系 生稲 史彦 准教授には、筆者が所属するビジネス科学研究科にて経営戦略論の講義でお世話になって以来、ソフトウェア開発企業の研究を行われている見地から、研究に対し様々なアドバイスを頂戴しました。ありがとうございました。

筆者の勤務先でもある株式会社ハートビーツ[*1]にアルバイトとして勤務している、高村 成道 さん・吉川 竜太 さんには、アンケート作成にあたり貴重なアイディアを頂戴しました。被験者と同年代の現役大学生であり、かつソフトウェア技術者としての職業経験がある彼らから、大学生にどのような言葉を問いかければ良いかをアドバイスをもらい、大変参考になりました。ありがとうございました。

また、本論文の執筆にあたって、応援いただいた勤務先の株式会社ハートビーツの全ての仲間へ感謝します。特に、馬場 俊彰 CTO には、勤務時間の調整に快諾いただきました。そして、藤崎 正範 代表取締役には、高等教育を受けてこなかった私に、「せっかく勉

[*1] http://heartbeats.jp/

強できる機会なのだから、今、ちゃんと学校へ行きなさい。」と声をかけていただきました。その結果、私を研究に充分な時間を割く事ができ、本論文を完成させる事ができました。ありがとうございました。

また、時間を割く事に協力してくれた家族に感謝します。

同大学院 久野ゼミの有志の皆様からも、少なからぬ助言をいただきました。

同ゼミのメンバーでありながら、東京農工大学を本務校として活動されている辰己 丈夫 准教授からは、情報教育の研究者の見地から、様々な場所でアドバイスや相談に応じていただきました。大変感謝しております。

同ゼミ OB の西森 丈俊 博士には、本研究科へ進むためのきっかけを作っていただきました。3 年前、「ああ、自分にももう少し計算機に関する知識あれば。」[*2][*3][*4] と嘆いた時に、研究の道を紹介してくださいました。そして、入学後も本論文のレビューに多くの時間を割いていただきました。日本語の表現から、論述内容に至るまで、多くのフィードバックをいただけたことで、本論文の完成度を高める事ができました。新しい道筋に導いていただいた事を心から感謝します。ありがとうございました。

最後に、本論文の作成を一貫してご指導いただいた、筑波大学 大学院 ビジネス科学研究科 久野 靖 教授に感謝致します。本論文の作成指導はもちろんの事、被験者の紹介や研究会発表など、様々なご縁を通じて研究を進める環境を作っていただけた事で、こうして論文を完成させる事が出来ました。本当に、ありがとうございました。

[*2] http://twitter.com/koemu/status/19054766615
[*3] http://twitter.com/nistake/status/19054966433
[*4] http://twilog.org/koemu/date-100721

参考文献

[1] Michael A. Cusumano. *The Business of software: what every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad.* Free Press, 2004.

[2] Laurie Williams 著, 久野 禎子, 久野 靖 訳. 若葉マークのプロ: 最近の卒業生、初めてのソフトウェア工学のお仕事, 第 26 章. Making Software — エビデンスが変えるソフトウェア開発. オライリージャパン, 2011.

[3] 一般社団法人日本経済団体連合会. 新卒採用（2011 年 3 月卒業者）に関するアンケート調査結果の概要, 2011.

[4] 山口裕幸 (編). コンピテンシーとチーム・マネジメントの心理学, 第 26 巻. 朝倉書店, August 2009.

[5] J T Wood. *Gendered Lives: Communication, Gender, and Culture.* Cengage Learning, tenth edition, 2012.

[6] 大森隆行, 丸山勝久, 林晋平, 沢田篤史. ソフトウェア進化研究の分類と動向, pp. 3–28. 一般社団法人 日本ソフトウェア科学会, 2012.

[7] W. W. Royce. Managing the development of large software systems: concepts and techniques. pp. 328–338, March 1987.

[8] 玉井哲雄. ソフトウェア工学の基礎. 岩波書店, 2004.

[9] Kent Bech ほか. *Manifesto for Agile Software Development.* http://agilemanifesto.org/, 2001.

[10] Stefan Koch. Agile Principles and Open Source Software Development: A Theoretical and Empirical Discussion. In Jutta Eckstein and Hubert Baumeister, editors, *Extreme Programming and Agile Processes in Software Engineering*, Vol. 3092 of *Lecture Notes in Computer Science*, pp. 85–93. Springer Berlin / Heidelberg, 2004.

[11] Kent Beck and Cynthia Andres. Extreme Programming Explained: Embrace Change (2nd Edition). November 2004.

[12] J Rasmusson. *The Agile Samurai: How Agile Masters Deliver Great Software.* Pragmatic Bookshelf Series. O'Reilly Vlg. GmbH & Company, 2010.

[13] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona. *ACM SIGCOMM Computer Communication Review*, Vol. 39, No. 4, p. 135, August 2009.

[14] Hugh Beyer and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems (Interactive Technologies)*. Morgan Kaufmann, 1997.

[15] 情報システム・ソフトウェア取引高度化コンソーシアム編. 情報システム・ソフトウェア取引トラブル事例集. Technical report, 2010.

[16] Robert Lee Katz. *Skills of an Effective Administrator*. Harvard Business Review Classics. Harvard Business Press, 1974.

[17] 情報処理推進機構. IT スキル標準活用の手引き : 企業導入の考え方:. 情報処理推進機構 IT 人材育成本部 IT スキル標準センター, 2009.

[18] 学校法人 産業能率大学財団法人 日本情報処理開発協会. 我が国ＩＴサービス市場に関するスキル動向等調査研究報告書. 経済産業省, 2011.

[19] 独立行政法人情報処理推進機構. ＩＴスキル標準 (ITSS). 独立行政法人 情報処理推進機構, 2002. http://www.ipa.go.jp/jinzai/itss/.

[20] 引地一将. ソフトウェア開発プロセスにおける定量的管理指標の導入支援. 2006.

[21] 情報処理推進機構. IT スキル標準導入活用事例集. 情報処理推進機構 IT 人材育成本部 IT スキル標準センター, 2011.

[22] CMMI Product Team. CMMI for development, version 1.3. Technical report, Carnegie Mellon University, November 2010.

[23] Process Assessment and ISO/IEC 15504: A Reference Book (The Springer International Series in Engineering and Computer Science): Han van Loon: 9780387231723: Amazon.com: Books.

[24] 情報処理学会情報処理教育委員会, J07 プロジェクト連絡委員会編. 情報専門学科におけるカリキュラム標準 J07, 2003.

[25] Russell Shackelford, Barry Lunt, Andrew McGettrick, Robert Sloan, Heikki Topi, Gordon Davies, Reza Kamali, James Cross, John Impagliazzo, and Richard LeBlanc. Computing Curricula 2005: The Overview Report. *ACM SIGCSE Bulletin*, Vol. 38, No. 1, p. 456, March 2006.

[26] 駒谷昇一, 田中二郎, 北川博之. 筑波大学における高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム. 工学教育, Vol. 57, No. 4, pp. 92–98, July 2009.

[27] 独立行政法人情報処理推進機構. 汎用的教育コンテンツ, 2012.

[28] 相川 充, 高本 真寛, 杉森 伸吉, 古屋 真. 個人のチームワーク能力を測定する尺度の開発と妥当性の検討. 社会心理学研究, Vol. 27, No. 3, pp. 139–150, 2012.

[29] R F Bales. Interaction process analysis. *Cambridge, Mass*, 1950.

[30] Robert Freed Bales 著, 友田不二男, 手塚郁恵 訳. グループ研究の方法. 岩崎学術出版社, 1971.

[31] C R F Bales and P Stephen. SYMLOG: A system for the multiple level observation of groups. 1979.

[32] 奥田 達也, 伊藤 哲司. SYMLOG の日本語改良版—小集団構造把握のための簡便な評定項目の作成 (対人的相互作用¡特集¿). 実験社会心理学研究, Vol. 31, No. 2, pp. p167–174, 1991.

[33] 独立行政法人情報処理推進機構. IT 人材白書 2011. 独立行政法人 情報処理推進機構, 2011.

[34] 辰己 丈夫, 中野 由章, 野部 緑, 川合 慧. 情報フルーエンシーを意識した大学の一般情報教育のカリキュラム提案. 情報処理学会研究報告. コンピュータと教育研究会報告, Vol. 2009, No. 9, pp. 1–8, 2009.

[35] William W. Lee, Diana L. Owens, 日本イーラーニングコンソシアム, 清水康敬. インストラクショナルデザイン入門 : マルチメディアにおける教育設計. 東京電機大学出版局, 2003.

[36] Andrew Begel, Beth Simon 著, 久野 禎子, 久野 靖 訳. ペアプログラミング, 第 17 章. Making Software — エビデンスが変えるソフトウェア開発. オライリージャパン, 2011.

[37] 小林雅史, 箱守知子, 磯部匡志, 赤坂幸彦, 村松充雄. アクションラーニング手法を取り入れたメンタリングによるプロジェクトマネージャ育成手法の提案 (我が社の PM 事例). プロジェクトマネジメント学会誌, Vol. 9, No. 1, pp. 46–49, February 2007.

[38] Jeffrey Wong and Jason Hong. What do we ”mashup” when we make mashups? In *Proceedings of the 4th international workshop on End-user software engineering - WEUSE '08*, pp. 35–39, New York, New York, USA, May 2008. ACM Press.

[39] 辰己丈夫, 江木啓訓, 瀬川大勝. 大学 1 年生の情報活用能力と ICT 機器やメディアの利用状況調査. 学術情報処理研究, Vol. 16, pp. 111–121, 2012.

付録 A

被験者 募集要項

情報教育に関する実験協力のお願い

斎藤 祐一郎*

2012 年 4 月 25 日

1 要旨

私、斎藤 祐一郎 (以下、私) は、筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻 博士前期課程 (以下、大学院) において、情報教育、特にソフトウェア技術者の新人教育に関する研究 (以下、本研究) を行っております。

研究を遂行するにあたり、私が作成した情報教育に関する教育プランが従来の教育方法よりも成果があがるものであるかどうかを実験が必要となっております。その際に、貴校、特に皆様のもとで研究に励まれている学生さんにご協力願えたらと考えております。

2 私について

私は社会人ではありますが、大学院において経営と計算機科学を学びながら、久野 靖 教授のもとで情報教育の研究を行っております。

普段は、AR を用いたスマートフォンのソフトウェア開発、及びインターネットサービス用のサーバのシステム設計・構築・運用に携わっております (以下、現職)。

3 研究について

本研究のタイトルは「人間関係を重視したコンピュータ技術者教育手法」であります。

まず、現在のシステム開発がアジャイルや Contexual Design を初めとした顧客・チームメンバーそれぞれの人間関係を理解し積極的な会話を通じてインタラクティブに行うことが主流となっております。しかし、多くのソフトウェア技術者は会話が苦手であり、開発時に有為なコミュニケーションを行えず、結果としてシステム開発の結果が顧客に評価されづらい状況が出ております。

そこで、新人社員の頃からインタラクティブな開発に馴染みやすい素養をつけるべく、これまでの「座学」だけの教育だけではなく、ケーススタディやペア・プログラミングを通じた教育を行おうとするものです。

4 実施内容

学生さんには、プログラミング、またはサーバ構築に関するケーススタディまたはワークショップを行っていただきます。1 つのケーススタディあたり 1 日 2〜3 時間を 1 週間程度、ワークショップは 1 回 6 時間程度を予定しており、どちらかを最低 2 回実施致します。

来ていただく学生さんは、計算機科学の基礎[*1]を学んだ 3 年生または 4 年生、人数は 1 校 3 名以上を希望しております。可能であれば、2 チームに分けて比較実験を行いたい事から、6 名以上集まっていただけますと大変助かります。

5 スケジュール

学生さんの時間が確保しやすい夏休みに合わせて実施したいと考えております。そのため、7 月から 9 月の間に実施日を設定致します。

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻

*1 C 言語で関数・ポインタの理解が進んでいる水準

研究にあたり、次の段階で学生さんのデータを取得致します。

- 実施前 — 当初のスキルの確認 (アンケート) およびチーム割り時のバランス確認
- 実験 1 回目終了後 — 到達度の確認 (技術に関するテスト・アンケート・インタビュー)
- 実験 2 回目終了後 — 到達度の確認 (技術に関するテスト・アンケート・インタビュー)

お伺いする際、私自身は平日は業務がありますが、勤務先には大学院での研究活動として平日において数時間の研究活動が認められており、かつ土休日は自分自身で制御可能な時間がございます。その中で、先生と学生さんとお話させていただく事になります。

メールについては、業務に携わっていない時間にご連絡可能です。

研究結果は、12 月末までに修士論文として仕上げます。あわせて、情報処理学会をはじめとした学会の研究会での発表を行う予定です。

6　参加時のメリット・デメリット

学生さんには、実際に職業としてコンピュータ技術者に就いた際のイメージを知るきっかけになるのではと考えております。これは、自分のためだけではない、誰かのためにプログラムを書き上げる楽しさを知ることができることになります。その上で、就職活動やインターン活動でお役立ていただけるものと期待しております。

また、実験参加に際しまして、ご参加いただく学生さんの自由意志によるものであります。従いまして、学生さんが実験協力を辞退した事による不利益は一切生じる事はありません。

7　謝礼等

ワークショップに参加いただく際の謝礼は、薄謝ながらお出しする事が可能です。

また、論文への謝辞の記載を行います。

8　連絡先

8.1　研究に関する問い合わせ

趣旨にご賛同いただける場合、以下にご連絡いただけましたら幸いです。

- 氏名: 斎藤 祐一郎
- 所属: 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻
- メールアドレス: saitou@gssm.otsuka.tsukuba.ac.jp (大学院宛)

8.2　倫理的な問題に関する問い合わせ

本研究に関する倫理的な問題については、筑波大学 ビジネスサイエンス系 研究倫理委員会 (社会人大学院等支援室・研究支援担当) までご相談ください。

- 筑波大学 社会人大学院等支援室・研究支援担当
- メールアドレス: hitorinri@office.otsuka.tsukuba.ac.jp

何卒、ご協力賜りたく、よろしくお願い致します。

以上

付録 B

事前アンケート 書式

「人間関係を重視したソフトウェア技術者教育手法」実施前アンケート

斎藤 祐一郎*

2012 年 7 月 11 日

1　はじめに

今回は、「人間関係を重視したソフトウェア技術者教育手法」の研究に関する実験に協力いただき、誠にありがとうございます。

本実験の実施前に、ご協力いただける皆様の現在の状況を知りたく、アンケートにご協力ください。忙しい所、お手数ですがどうぞよろしくお願いします。

回答は LaTeX のソースへ直接書き込んだ後、PDF に変換したものを送付してください。

2　基本事項

2.1　お名前

(記入)

2.2　所属

(記入: 大学名 - 学部名 - 学科名 (専攻名))

2.3　性別

1 つ選択してください。

1. 男
2. 女

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻

3　人間関係について

3.1　自分自身の役割について

今回、システム開発の実習を実施する予定です。その際、チームで仕事をする事になった場合、特に開発のどの辺りで自分自身が貢献できると考えていますか。例えば「プログラミング」、「画面 (UI) のデザイン・部品作成」または「チームの作業進行管理」という具体的な形で答えてください。複数あっても結構です。
(記入)

3.2　仲間の役割について

上記に続き、仲間に対してどのような貢献を求めますか。例えば、「自分は画面制作が出来ないので代わりに作成をお願いしたい」、「プログラム開発を分担して開発速度を上げたい」などで結構です。
(記入)

3.3　自分自身の配慮について

チームで仕事をするにあたり、技術側面以外で仲間に対してどのような気配りを心がけますか。例えば、「ピンチの時に冷静になれるよう士気を安定させる」、「普段楽しく進捗できるよう会話を絶やさない」などです。
(記入)

3.4　これまでの経験について

これまで、どのようなアルバイトやインターンに従事しましたか。業務内容を新しいものから順に時系列で記入してください。会社名は書かなくて結構です。
(記入 例: 飲食店 - ホールスタッフ)

3.5　チーム内での立ち位置について

これまで、仕事やクラブ活動などの社会活動全般において、強いて言えば自分自身はどちらの立ち位置だと思いますか。

1. リーダー
2. メンバー

4　技術について

4.1　プログラミング能力について

プログラミングにあたって、自分自身 (一人) で出来るものを 1 つ以上選択してください。

1. 技術書や講義のレジュメを見ながらプログラムを書き写して動かすことができる
2. 既にあるプログラム (ライブラリではない) を改造したりバグ修正をすることができる
3. ゼロから (ライブラリは既存のものを利用可) でアプリケーションを開発する事が出来る
4. ライブラリを解析し問題点を特定する事が出来る
5. 自分自身でライブラリを開発する事が出来る
6. 既存のプログラムを他のプラットフォーム (例:iPhone から Android) に移植できる

4.2　ソフトウェア開発能力について

ソフトウェア[*1]開発にあたって、該当する箇所をそれぞれ 1 つ選択してください。

	知らない	聞いた事がある	習った	個人的に触った	同年代では出来る方
ソフトウェア企画					
要件の整理					
アーキテクチャ設計					
内部設計					
インタフェース設計					
負荷見積					
開発環境構築					
成果物管理					
プログラミング					
テスト計画作成					
テスト					
自動的なデプロイ					
運用計画作成					
運用					
トラブルシューティング					

4.3　工程管理について

自分自身がソフトウェアを開発するにあたって、どのような工程ですすめているか、大まかに記載してください。
(記入)

[*1] スマートフォンのアプリケーションや Web サービス

5　学習について

5.1　問題解決

　ソフトウェア開発時に問題が発生した際、どのような方法で解決手段を導き出しますか。思い当たるだけ、項目を挙げてください。
　(記入 例: Google で検索する)

5.2　普段の情報収集について

　ソフトウェア開発に取り組むにあたって、普段どのような情報収集の手段を設けていますか。思い当たるだけ、項目を挙げてください。
　(記入 例: 情報処理学会の研究会へ出席している)

5.3　実践について

　普段、業務や学校での学習以外で、ソフトウェア開発をどのような形で実践していますか。思い当たるだけ、項目を挙げてください。
　(記入 例: Android アプリを自分で作っている)

5.4　発信について

　普段、業務や学校での学習以外で、ソフトウェア開発の結果をどこかで公にしていますか。思い当たるだけ、項目を挙げてください。
　(記入 例: Google Play で自作のアプリを公開している)

ここまでです

　質問は以上です。ありがとうございました。

以上

付録 C

レポートフォーマット

ケース 課題提出用帳票

斎藤 祐一郎*

2013 年 1 月 2 日

はじめに

今回は、「人間関係を重視したソフトウェア技術者教育手法」の研究に関する実験に協力いただき、誠にありがとうございます。

ケース提出用のファイルです。必要事項を記入の上で、PDF にして提出してください。提出内容については、ケース説明の資料を参照してください。記入時は”(記入)”などの指示が書いてある部分は削除してしまって結構です。

別紙がある場合は「別紙 XX 番 参照」とし、提出時に同時に出すようにしてください。

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻

1　第 1 回 提出物

1.1　選択したプラットフォーム

(記入)

1.2　作業分担リスト

- リーダー:
- プログラム:
- テスト:
- デザイン:

1.3　コンセプトのメモ

(記入)

1.4　外部設計の図面

(記入 または別紙添付)

2　第 2 回 提出物

2.1　機能設計書

(記入 または別紙添付)

2.2　画面遷移図

(記入 または別紙添付)

2.3　画面設計図

(記入 または別紙添付)

3　第 2 回 提出物

3.1　機能設計書

(記入 または別紙添付)

3.2　画面遷移図

(記入 または別紙添付)

3.3　画面設計図

(記入 または別紙添付)

付録 D

SYMLOG 用 評価シート

ビデオ観察　チェック指標

評価者：
被験者A:
被験者B:
被験者C:
評価基準：　しばしばした:3点，時々した:2点，ほとんどしなかった:1点

No.	行動基準	確認項目	被A−前	被A−後	被B−前	被B−後	被C−前	被C−後	備考
1	P	みんなの気持ちを引き立てるように振る舞った							
2	P	雰囲気を和らげるよう振る舞った							
3	N	相手の話の腰を折った							
4	N	相手の言うことに何でも反対した							
5	U	みんなに指示を与えた							
6	U	みんなの意見をまとめた							
7	D	相手の意見を聞いてから発言した							
8	D	相手の意見に従った							
9	F	みんなに冷静な態度で接した							
10	F	よく考えてから発言した							
11	B	周りにしらけた態度を示した							
12	B	周りの雰囲気にそぐわないふるまいをした							

付録 E

最終アンケート 書式

コミュニケーションスキルを重視した ソフトウェア技術者教育手法 最終アンケート

斎藤 祐一郎*

2012 年 10 月 9 日

1 はじめに

今回は、「コミュニケーションスキルを重視したソフトウェア技術者教育手法」の研究に参加いただき、誠にありがとうございました。
最後に、アンケートにご協力ください。忙しい所、お手数ですがどうぞよろしくお願いします。
回答は LaTeX のソースへ直接書き込んだ後、PDF に変換したものを送付してください。

2 人間関係について

2.1 自分自身の役割について

今回、担当として割り当てられた作業の役割について、問題なく役目を果たせたと思いますか。また、それはなぜですか。
(記入)

2.2 仲間の役割について

他のメンバーに割り当てられた役割に、各人が当初の通りに果たしていたと思いますか。また、それはなぜですか。メンバー毎に記入をお願いします。
(記入)

2.3 自分自身の配慮について

チームで仕事をするにあたり、実験実施前のアンケート (以下、実施前) に自分自身で心がけると記述した点について、配慮が適切に行えましたか。加えて、配慮した事があれば記述してください。
(記入)

2.4 これまでの経験について

実験実施前に記述いただいた、これまで従事したアルバイト等の経験は活きましたか。また、それはなぜですか。

2.5 リーダーのみ: リーダーとしての活動について

リーダーの方は、当初求められていたリーダーとしての役割を果たせましたか。それは、なぜですか。
(記入)

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻

2.6　メンバーのみ: メンバーとしての活動について

メンバーの方は、リーダーはリーダーとして求められた仕事ができていましたか。また、リーダーをフォローすることはできましたか。そして、それはなぜですか。
(記入)

3 技術について

3.1 プログラマ: プログラミング能力について

プログラミングにあたって、今回の実験を通じて自分自身 (一人) で出来るようになったものを 1 つ選択してください。実験前に既に丸をつけたものには丸はつけないでください。

1. 技術書や講義のレジュメを見ながらプログラムを書き写して動かすことができる
2. 既にあるプログラム (ライブラリではない) を改造したりバグ修正をすることができる
3. ゼロから (ライブラリは既存のものを利用可) でアプリケーションを開発する事が出来る
4. ライブラリを解析し問題点を特定する事が出来る
5. 自分自身でライブラリを開発する事が出来る
6. 既存のプログラムを他のプラットフォーム (例:iPhone から Android) に移植できる

3.2 ソフトウェア開発能力について

ソフトウェア[*1]開発にあたって、今回の実験を通じてどの程度理解が深まったかを選んでください。

	わからない	存在を知った	手伝った	担当した	同年代では出来る方
ソフトウェア企画					
要件の整理					
外部設計					
画面設計					
内部設計					
負荷見積					
開発環境構築					
成果物管理					
プログラミング					
テスト計画作成					
テスト					
自動的なデプロイ					
運用計画作成					
運用					
トラブルシューティング					

3.3 工程管理について

これまでのプログラムのみを書く課題と比較して、工程を通してソフトウェア開発を行った際に、どのような違いがありましたか。気づいただけ、記入してください。
(記入)

[*1] スマートフォンのアプリケーションや Web サービス

4　学習について

4.1　問題解決:技術的側面

実装中、担当した工程において、技術的に行き詰まった所は何ですか。また、どのように解決しましたか。出来るだけ詳しく挙げてください。利用した書籍やサイトがあれば、用いたものをリストアップしてください。
(記入)

4.2　問題解決:担当者間の側面

担当者間で実装を調整、または成果物を連携するにあたって、苦労した点は何ですか。また、どのように解決しましたか。出来るだけ詳しく挙げてください。
(記入)

4.3　普段の情報収集について

実験前に既に知っていて、役立った知識や情報があれば、出来るだけ詳しく挙げてください。
(記入)

4.4　実践について

実験を通じた実践で、特に獲得できたと自分自身が実感できる能力があれば、挙げてください。
(記入)

4.5　完成したアプリケーションについて

完成したアプリケーションについて、350 字程度で紹介をお願いします。
(記入)

5　感想

感想を記入してください。実験内容、仲間へのメッセージ等、何でも結構です。
(記入)

ここまでです

質問は以上です。ありがとうございました。

以上

付録 F

ケース — Twitter の mashup サイトを作ろう

※本資料にはガイダンス時の説明が含まれている

ケース 1 — Twitter の mash-up サイトを作ろう

斎藤 祐一郎*

2012 年 8 月 31 日

1　はじめに

今回は、「人間関係を重視したソフトウェア技術者教育手法」の研究に関する実験に協力いただき、誠にありがとうございます。

今回、実験に参加していただく皆さんに対し、ソフトウェア開発を学ぶためのケースを作成しました。よく読んでいただいて、不明点や質問がありましたら遠慮なく斎藤までお願いします。

2　やること

Web アプリケーションを開発します。

期間は 2012/10/14(日)〜2012/10/20(土) の 1 週間です。開発の準備に 6 日 (1 日あたり 1〜2 時間)、プログラミング・テストに 1 日 (6 時間) をかけて実施します。

開発にあたり、皆さんで分担を決めて進めて行きます。分担と役割については後述します。その中で、斎藤はプロジェクトマネージャとして皆さんの進捗や成果物の管理を進める立場を務めます。

少し大変かと思いますが、がんばって行きましょう。

3　おことわり

■本実験結果は論文として発表します　実験の途中経過、及び最終成果物については、論文に記載をします。

■論文に所属大学名・研究室名を記載します　論文中に、所属大学の法人の分類 (国立・私立等) と大学名のイニシャルを記載します。これは実験の実施状況の説明と謝辞欄で用います。なお、個人の表記は「被験者 A, B, C」といった匿名表記とし、氏名を初めとした個人が特定できる形では書かないようにします。

■最終日はビデオ撮影があります　実験中、コミュニケーションの状況を調べるために、ビデオを撮影します。このビデオ撮影はログとして残す事が目的です。もし、公開する事がある場合は、改めて個別に承諾をとるようにします。

4　参加のメリット・デメリット

実際に職業としてコンピュータ技術者に就いた際のイメージを知るきっかけになるのではと考えています。これは、自分のためだけではない、誰かのためにプログラムを書き上げる楽しさを知ることができることになります。その上で、就職活動やインターン活動で役立てていただけたら幸いです。

また、実験参加は、みなさんの自由意志によるものであります。従って、学生さんが実験協力を辞退した事による不利益は一切生じる事はありません。

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻

5 システムの要件

5.1 開発するアプリケーション

「Twitter 上における友人・知人の会話の流れが読みやすくなる Web アプリケーションの開発」

5.2 開発の背景

Twitter、皆さん使っているでしょうか[*1]。

使った事がある人ならわかると思うのですが、follow する人が増えれば増える程、友人や知人を初めとした頻繁にチェックしておきたい人の tweet を見逃しやすくなります。そして、そんな tweet の中に、日頃の世間話の種や研究・仕事の話が混じっていたりする事も珍しくありません。次の日の朝、直接会った時に話について行けないというのは、いささかさみしいものがあります。

Twitter はその回答として、Lists を作りました[*2]。Lists を通じて友人・知人の tweet を積極的にチェックする際に活用している人もいらっしゃるでしょう。でも、会話の流れをつかむと言う観点では、これで本当に使いやすいと言えるでしょうか。

Twitter 上で友人・知人と会話をすると、それは 1〜2 回のやり取りで済まないことがあります。しかし、tweet の表示はあくまで時系列。いろいろな発言の中から、自分の目で抽出して追いかけなければ、その話の経緯は見えません。サードパーティの Twitter クライアントでは話の流れを追いかける「スレッド」機能がついているものがありますが、それを使うためにはいくつかのステップを踏まなければなりません。余り便利ではありません。

そこで、みなさんに友人・知人の会話の流れが読みやすくなる Web アプリケーションを開発していただきたいと思います。

5.3 機能要件

■認証機能　サービスを使うために、Twitter の oAuth を用いた認証機能を設けてください。これは、個人を特定して表示する内容を動的に切り替えるためのキーとなる情報となります。また、Twitter は認証のない通信は API[*3]の利用回数が著しく制限される[*4]ため、認証しなければ恐らくアプリケーションとしては使い物にならなくなる可能性があります。

■Lists 情報取得機能　認証機能を用いて認証したユーザの Lists 情報を取得します。ここに、友人・知人の Twitter アカウントを登録した Lists が入っているという前提にします。

■Timeline 表示機能　Lists に登録されている Twitter アカウントの Timeline を表示します。単に時系列で表示するだけではダメです。表示方法は、開発を担当する皆さんで「提案」し、実装してください。ツリー方式でもよいのですが、それ以上にいい何かがあればよりよいです。

■Timeline へ表示する情報　Timeline には次の情報を表示してください。

- 発言者のアイコン
- 発言者の ID
- 発言者内容
- 発言の流れを示す情報
- その他 Twitter のサイトを観察していて必要と思った機能

[*1] 斎藤のアカウントは @koemu です

[*2] Twitter's New 'Lists' Feature Finally Introduces Grouping, Offers An Alternative To The SUL — TechCrunch - http://www.techcrunch.com/2009/09/30/Twitters-new-Lists-feature-finally-introduces-grouping-offers-an-alternative-to-the-sul/

[*3] Application Program Interface

[*4] 認証時で 350 回/時, 非認証時で 150 回/時. https://dev.Twitter.com/docs/rate-limiting

5.4　非機能要件

■クライアントのプラットフォーム　PC かスマートフォン、どちらかを選んでください。また、スマートフォンの場合は、Android か iPhone のどちらかに絞ってもらっても構いません。

■クライアントのブラウザ　PC の場合は、最新の Google Chrome と Apple Safari で動くように開発してください。スマートフォンの場合は、Andorid 2.3, iOS 5.1 に搭載されているブラウザで動くようにしてください。

■サーバ側のアーキテクチャ　サーバ側は、動的・静的、どちらでも構いません。動的な場合はサーバサイドで処理できるような仕組みを作る事になります。静的な場合は、JavaScript を用いてブラウザ上で動作するよう開発してください。

■プログラミング言語　サーバサイドは JavaScript (node.js) を推奨します。もし、得意なプログラミング言語があればそれを選択してもらっても構いません。

クライアント側は、HTML + JavaScript とします。ネイティブのアプリケーション[*5]や Adobe AIR, Flash をはじめとしたプラグインが必要となるアプリケーションとしての開発は行わないでください。

ライブラリ・フレームワークについては、使用制限はありません。好きなものを自由に使ってください。node.js で書く場合は、Express を使うと良いのではと思います。

■サーバサイドの環境　OS は Debian GNU/Linux 6.0 (squeeze) 64bit になります。ポートは 22(SSH), 80(HTTP) を解放します。セキュリティ上の配慮から、他のポートは開いてはなりません。

サーバサイドのミドルウェアについては、次のものを用意します。

- MySQL 5.1
- SQLite 2.8
- node.js 0.8.8

ミドルウェアは、上記以外のものを使っても構いません。追加インストールは、自分自身で進めてもらって構いません。

6　作業工程

6.1　リーダーの決定

まず、リーダーを決定してください。リーダーは次の役割を負います。

- 斎藤との連絡窓口
- チーム内での最終意思決定権 (意見が割れた時はリーダーの意見が優先)
- チーム内での意思疎通に関する配慮 (技術的、精神的、物理的なもの全て)

リーダーでないメンバーの方は、リーダーに最大限の協力をし、存在を尊重してください。

6.2　作業に関する役割分担の決定

次の役割を決めてください。

- プログラム担当
- デザイン担当 (特に UI[*6])
- テスト担当

プログラム担当は、その名の通りプログラムを担当します。メインプログラマーとして、作り上げるプログラムに責任を負ってください。そのために、プログラムを進める前に行うべき企画や設計について適切に準備が必要になる事も、理解してください。

[*5] PC やスマートフォン上で実行可能なバイナリ形式
[*6] User Interface

デザイン担当は、画面のデザインや HTML 制作を担当し、責任を負ってください。同時に、アプリケーション全体の (広義の) デザインについても思慮しなければなりません。
テスト担当は、プログラム担当・デザイン担当が開発したアプリケーションをテストするための計画を作成し、実際にテストを行ってください。テストで見つからかったバグが出た場合、自分自身に責任があると考えてください。また、テスト担当は開発途中に出てくる成果物について、これで問題なくシステム開発が行えるかを目を光らせる必要があります。言うなれば、他の２人とは少し距離を置いて事に当たる必要があります。
各作業工程に、責任を持つ役割名を「主担当」として明記していますので、参考にしてください。
ただし、**役割にこだわり過ぎない**ようにしてください。手が空いているのなら手が足りない仲間を手伝う等、役割はあくまで「責任を負う範囲」だけであることを理解して、決めるようにしてください。

6.3 コンセプト作成

主担当: 全員
今回、開発するアプリケーションのコンセプト、いわゆる企画を考えてください。特に、Lists 表示にどのような工夫を施す事で、利用者の人にこれまでと違う体験 (Experience) をどのように提供できるのか、考えてみてください。
成果物としては、議事録やメモがある程度で構いません。

6.4 外部設計

主担当: 全員
外部設計とは、アプリケーションがどのように外の物事と関わるかを示す資料です。外とは、今回で言えば Twitter のサービス本体、実際に利用するユーザが挙げられます。その関係と、どのようなデータ・物事がやり取りされるのかを定義してください。
成果物は、概要を示した１枚の図を作成してください。手書きをスキャンしたものでも良いですし、Visio などのビジュアライズツールで作成したものでも良いです。きれいに作るために時間をかける事より、アプリケーションと外部との相関関係はっきりしていればそれで問題ありません。

6.5 機能設計

主担当: プログラム担当
機能設計[*7]とは、アプリケーション自体にどのような機能を実装するかを明確にします。言い換えれば、プログラムを書く際にどのような機能を持ったコードを書けば良いのかを定義します。
作成時、機能の分類を「大分類—中分類—小分類」と分けると作成しやすいです。
成果物は、機能を箇条書きでまとめた１枚の紙を作成してください。小分類が１行程度で収まるレベルで書いてもらえれば問題ありません。

6.6 画面設計

主担当: デザイン担当
画面設計は、大きく分けて２つに分かれます。
まず、画面遷移図です。どのような画面が存在し、どのような順序で動くのかを定義します。
続いて、画面レイアウトの設計です。画面内にどのような情報が表示され、どのような操作 (ボタンがある、等) が行えるのかを定義してください。
成果物は、画面遷移図が１枚、画面レイアウトは画面の数だけ作成します。画面の数は多ければいいと言う訳ではありません。

6.7 テスト計画書

主担当: テスト担当
テスト計画とは、開発したアプリケーションが正しく稼働しているかを検証する時に用いる、チェック項目そのものです。これらは、機能設計や画面設計からどのようなテストが必要かを読み取り、テスト項目をリス

[*7] 本来は外部設計の一部なのですが、今回は明確に分けました。

トアップします。アプリケーションの姿をどこまでルックスルーできるかが、腕の見せ所です。
　基本は、作成した機能が正しく動作しているかを検証する項目を作成します。これを正常系と言います。
　成果物は、テスト項目書となります。フォーマットをお渡ししますので、埋めてください。

6.8　リソース作成

　主担当: デザイン担当
　リソース作成は、画面の HTML や画像ファイルを作成する工程です。画面設計に従って、作成してください。
　成果物としてはファイルが出来上がると思いますが、こちらは現段階で提出しなくてよいです。

6.9　テストプログラム試作

　主担当: プログラム担当
　テストプログラム試作は、実装する機能の中で技術的に検証をしておいた方が良いと思われる部分についてのプログラムを先行開発する事です。心配な事があればやっておいてください。
　成果物としてはファイルが出来上がると思いますが、こちらは現段階で提出しなくてよいです。

6.10　プログラム開発

　主担当: プログラム担当
　出来上がったリソースを用い、機能設計に基づいたプログラムを開発します。がんばってコードを書いてください！テストを実施した後、バグ修正もありますので最後までやり抜いてください。
　なお、プログラム開発中に画面デザインの変更等の仕様変更が発生する事もあるかと思います。それは時間がある限り変更してもらって構いません。その節は、デザイン担当の人も協力してください。
　成果物は、出来上がったアプリケーションそのものと、コードを提出してください。

6.11　テスト実施

　主担当: テスト担当
　出来上がったアプリケーションに対して、テスト計画書に基づいてテストを実施してください。プログラム担当が作成したアプリケーションを無事納品できるかは、テストを適切に行えたかにかかっています。
　成果物は、テスト結果を記入したテスト項目書となります。

7　スケジュール

7.1　日程

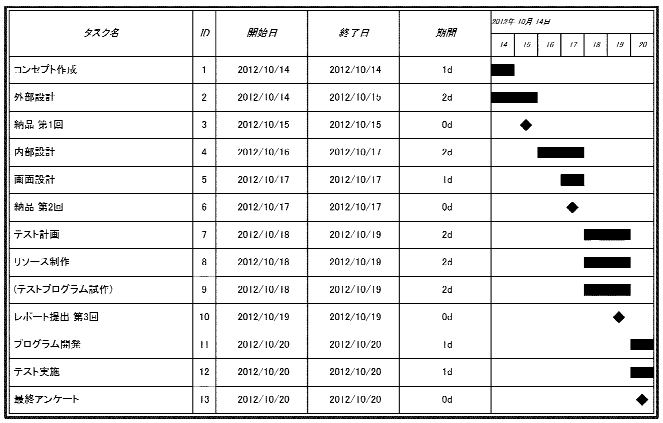
　作業の目安となるスケジュールを図 1 に示します。講義や研究のスケジュールもあるかと思いますので、絶対に守らなくてはならないものではありません。調整が必要な場合は、その旨を事前に教えてください。少なくとも、無断で変更は行わないようにしてください。

7.2　最終日について

　最終日は、プログラミングのワークショップを行います。チームで連携し合いながら、アプリケーションを完成させてください。斎藤は立ち会いますが、ワークショップ中の内容の質問については基本的に応じられませんので、準備を整えて来訪してください。
　場所は次の通りです。大変かと思いますが、よろしくお願いします。

■日時　2012 年 10 月 20 日 (土) 10:00〜17:00 (休憩 1 時間)

■場所　筑波大学 東京キャンパス 文京校舎 623 教室
　https://www.tsukuba.ac.jp/access/bunkyo_access.html
　最寄り駅: 東京メトロ 丸ノ内線 茗荷谷駅



タスク名	ID	開始日	終了日	期間
コンセプト作成	1	2012/10/14	2012/10/14	1d
外部設計	2	2012/10/14	2012/10/15	2d
納品 第1回	3	2012/10/15	2012/10/15	0d
内部設計	4	2012/10/16	2012/10/17	2d
画面設計	5	2012/10/17	2012/10/17	1d
納品 第2回	6	2012/10/17	2012/10/17	0d
テスト計画	7	2012/10/18	2012/10/19	2d
リソース制作	8	2012/10/18	2012/10/19	2d
(テストプログラム試作)	9	2012/10/18	2012/10/19	2d
レポート提出 第3回	10	2012/10/19	2012/10/19	0d
プログラム開発	11	2012/10/20	2012/10/20	1d
テスト実施	12	2012/10/20	2012/10/20	1d
最終アンケート	13	2012/10/20	2012/10/20	0d

図 1　日々のスケジュール

■交通費　支給します。手続がありますのでおつきあいください。

8　成果物の納品

8.1　納品について

システム開発には、成果物の納品がつきものです。その納品物と方法について示します。それぞれの提出日は先述のスケジュール表を見てください。〆切時刻は指定日の 23:59:59 とします。

納品に際して、情報をまとめやすくするために実施日前に納品資料のフォーマットを渡します。また、納品物 (後述) は提出後に更新してもらっても構いません。

8.2　納品方法について

リーダーから斎藤宛に、メールで行います。数 MB を越える大きなファイルがある場合は、貸与するサーバにアップロードしてください。連絡先は最後に記載しました。

8.3　第 1 回 納品物

- 選択したプラットフォームの情報 — A4 1 枚 (空白が多くて構いません)
- 作業分担リスト — A4 1 枚 (空白が多くて構いません)
- コンセプトのメモ — A4 1 枚
- 外部設計の図面 — A4 1 枚

8.4　第 2 回 納品物

- 機能設計書 — A4 1 枚以上
- 画面遷移図 — A4 1 枚
- 画面設計図 — 1 画面 A4 1 枚

8.5　第 3 回 納品物

- テスト計画書 — A4 1 枚以上

8.6　最終回 納品物

- アプリケーション — 実際に動かして試します
- アプリケーションのコード — ディレクトリ構造を保持した状態で ZIP ファイルに固めてください
- テスト結果が記入されたテスト計画書 — A4 1 枚以上
- アンケート — 記入用紙は最終日に渡します

9　準備関連

9.1　各自準備を要するもの

9.1.1　機材

■ノートパソコン　開発環境として必ず用意してください。 OS は問いません。
また、次のソフトウェアを必ずインストールしてください。

- SSH クライアント — サーバで作業する際に必要です。Mac, UNIX 系 OS は標準であります。Windows は putty をインストールしてください。
- SFTP クライアント — ファイルをサーバにアップロードする際に必要です。Cyberduck が便利です。
- エディタ — 好きなエディタを使ってください。
- メールクライアント — 好きなものを使ってください。

■スマートフォン　スマートフォンを対象プラットフォームとした場合は必要です。PC の場合は不要です。

9.1.2　手続等

■Twitter アカウント　1 人 1 つ、必ず作ってください。既に持っているものがあれば、それを用いてもらって結構です。また、開発用に 2 つ以上追加することも構いません。

9.2　貸与するもの

■サーバ　こちらでサーバを用意します。アカウント等も必要になった時に渡します。

■参考となる資料　コピーしてお渡しします。

10　連絡事項

10.1　サーバについて

原則、自分自身でいじってもらってかまいません。root 権限は sudo 実行可能の形で出します。
ただし、やり方がわからない事が出たら、私の方でサポートをしますので遠慮なく言ってください。サーバインフラ構築作業に集中する事は、今回の課題の範囲を逸脱するのでがんばり過ぎないようにしてください。
また、万が一に備えて、データのバックアップはとるようにしてください。
尚、次の事だけは避けるようにしてください。

1. 22, 80 以外のポートを開く
2. ユーザを作る — もし必要になったら私の許可を取ってください

10.2　デバッグについて

仮想化ソフトを用いて、手元で開発環境を作る事も出来ます。無料で済ませるなら、手元の PC へ VirtualBox をいれ、そこに Debian GNU/Linux を入れてもらえれば良いかと思います。ただし、構築は自分自身でやる事になります。手順はお渡しします。

10.3 調査について

あらゆる資料、そして人脈を活用して調査を進めてもらって構いません。
ただし、斎藤からは本ワークショップ遂行にあたって不明点がある場合を除き、質問には答える事はありません。その点の留意をお願いします。

10.4 評価について

本実験に際して、評価の観点は技術だけではありません。チームワークや、情報収集能力など、アプリケーションを完成させるためにどのような行動をとったかを見ます。その結果は、事前・事後のアンケート、納品物、そしてビデオを通じて計測します。
従って、凄いアプリケーションを作れば評価が高い、と言う訳ではないので気をつけてください。

10.5 アプリケーションの著作権について

実験に参加いただいた皆さんに帰属します。アプリケーション開発時に作成したソフトウェアのコードはあくまで評価の際の情報として用い、本研究目的以外で流用することはありません。

10.6 今後の開発について

もし、モティベーションが沸いてくるようでしたら、実験後もアプリケーション開発を続けてみてください。ただし、サーバの環境は皆さんで調達してもらう事になります。ちょっとした相談は乗れると思うので、その気になったときは声をかけてください。
ただ、私の研究の都合から、公開される場合は 2012 年 12 月以降でお願いします。

10.7 目を通すといい参考情報

10.7.1 Twitter 関連

Twitter Developers Document: https://dev.twitter.com/
Twitter の色んな値を JSON で取得する方法 — WEB-PARK.ORG — サイト制作に使用した自作 jQuery コードのご紹介: http://web-park.org/javascript/twitter_json.html

10.7.2 設計関連

企画書を作るのに便利な WEB サービスまとめ - NAVER まとめ: http://matome.naver.jp/odai/2130370365787952601
だれも教えてくれなかった外部設計の「極意」ITpro: http://itpro.nikkeibp.co.jp/article/COLUMN/20080515/301810/
WEB ディレクターなら参考にしたい手書きの画面設計書 | designaholic -Creative Column-: http://designaholic.cc/2011/02/post-82.html

10.7.3 テスト計画関連

知るだけで天地の差が出る、テスト仕様書の必須項目＆表現方法 － @ IT 自分戦略研究所: http://jibun.atmarkit.co.jp/lskill01/rensai/writing12/01.html

10.7.4 プログラミング関連

node.js: http://nodejs.org/
サーバサイド JavaScript の本命「Node.js」の基礎知識 - @ IT: http://www.atmarkit.co.jp/fwcr/rensai2/nodejs01/01.html
node.js と MySQL で割と普通のデータベースウェブアプリを作ってみるチュートリアル — さくらたんどっとびーず: http://sakuratan.biz/archives/3101
Express - node.js web application framework: http://expressjs.com/
jQuery: The Write Less, Do More, JavaScript Library: http://jquery.com/
MySQL :: MySQL 5.1 Reference Manual: http://dev.mysql.com/doc/refman/5.1/en/index.html
Official Ubuntu Documentation: https://help.ubuntu.com/ (Debian も同様に操作可能)

11　連絡先

11.1　実験に関する問い合わせ

■氏名　斎藤 祐一郎 (さいとう ゆういちろう)

■大学・専攻名　筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻 博士前期課程

■所属研究室　久野 靖 (くの やすし) 研究室

■メールアドレス　saitou [at] gssm.otsuka.tsukuba.ac.jp

■電話番号　XXX-XXXX-XXXX
※電話は緊急時のみに限定してください。なお、平日日中は社会人として仕事をしていますので、そこは避けてください。

11.2　倫理的な問題に関する問い合わせ

本研究に関する倫理的な問題については、筑波大学 ビジネスサイエンス系 研究倫理委員会 (社会人大学院等支援室・研究支援担当) までご相談ください。

- 筑波大学 社会人大学院等支援室・研究支援担当
- メールアドレス: hitorinri [at] office.otsuka.tsukuba.ac.jp

以上

付録 G

事前指導 資料

ケースにとりかかるにあたって

斎藤 祐一郎*

2012 年 10 月 9 日

1　はじめに

ケースにとりかかる前に、皆さんに知っていただきたい事があります。

チームでソフトウェア開発に携わる事は、実習や趣味で一人でコードを書く事とは違う事があります。その違いについて事前に少し学んでみましょう。

2　チームで開発する時に必要な「合意形成」

2.1　自分が考えている事と人が考えている事は違う

ソフトウェア開発がうまく行かない事例があります。情報科の学生さんであれば、何度か聞いた事があると思います。でも、大学や企業でソフトウェア開発について学習をし経験を深めている人が、なぜ失敗するのだろうと考えた事はありますか？

一人で開発する時と、チームで開発を行うとき、決定的に違うのはチームメンバーやお客様との「合意形成」が必要になることです。言い換えれば、自分が考えている事と人が考えている事は違う、だからすりあわせなければならないのです。プログラムは変数名 1 つを間違えただけで動かなくなります。その変数名、相手に受け渡す時にわかりやすくなっていますか？逆に、受け取る時にそれが自分で開発が継続できる形で整備されているでしょうか。

今回は、チームで開発する時に必要となる「合意形成」の力をつけるために、ケースを通じて理解を深めていただきたいと思います。

2.2　演習: ブラウザの機能を挙げてみよう

所要時間:20 分
準備するもの: 真っ白な紙・ペン

1. ブラウザ (Internet Explorer, Firefox, Chrome, etc...) に実装されている機能を、自分自身が「必要」と思うものを優先してリストアップしましょう。
2. リストアップした機能を、チーム内で読み合わせましょう。その中で、全員が合致したものを丸、2 人が合致したものを△としましょう。
3. 全く合致しなかったものについて、1 人 1 つ、必要性について主張しましょう。
4. チーム全員で、それぞれの主張を鑑み、1 つも合致しなかった機能について 1 つだけをピックアップしましょう。
5. ピックアップした 1 つの機能について、一人が代表してその理由を発表しましょう。

以上

* 筑波大学 大学院 ビジネス科学研究科 経営システム科学専攻