論文

プログラミング教育への導入のための 情報システム概念に基づくアンプラグドワークショップ

An Unplugged Workshop for Introducing System Development Process in Programming Education

プログラミング教育への導入のための 情報システム概念に基づくアンプラグドワークショップ

荒木恵† 松澤芳昭†† 杉浦学† 大岩元†††

本稿では「お絵かきプログラム開発演習」という，情報システム概念に基づく，プログラミング入門教育の導入に適したアンプラグドワークショップについて報告する．このワークショップでは「魅力的な絵を，期限内に，正確に，だれにでも描けるような日本語プログラム」を開発するプロジェクトを数名のグループを組んで実施する．参加者は「発注者」「設計者」「プログラマー」「テスター」を分担し，絵や文書のみでコミュニケーションをしながらプロジェクトを遂行し，結果に関する議論を行う．我々はこのワークショップを大学生を中心に，高校生や社会人にも実践している．いずれの対象でも，コミュニケーションの難しさを中心とするソフトウェア開発の本質的な討論がなされる．このワークショップは前提知識が必要なく，60 分で実施できる．また，文系・理系を問わず，参加者全員が楽しんで参加できる内容である．ソフトウェア開発の全体像とプログラミングの関係が理解でき，その後のプログラミングの授業内容の理解を助ける効果がある．副次的な効果として，プログラミングに対するマイナスイメージを払拭し参加者同士の交流を深めるアイスブレーキングの効果もある．

An Unplugged Workshop for Introducing System Development Process in Programming Education

Megumi Araki† Yoshiaki Matsuzawa†† Manabu Sugiura†
Hajime Ohiwa†††

1. はじめに

プログラミング教育はコンピュータを使いこなすための情報教育として行われてきたが，近年ではアプリケーションソフトウェアの普及に伴って，プログラミングそのものはコンピュータを使用するためには必須のものではなくなりつつある．しかしながら，プログラミング教育は，プログラミング能力の獲得だけでなく，そのプロセスを通して論理的な問題解決能力を育成することを目的とする教育であり，この目的を主とし，情報教育の柱として発展していくべきと，我々は考えている．

こうしたプログラミング教育は，複雑な状況から問題を同定し，多様に視点からの評価を経て問題解決をはかる，情報技術分野の一つである「情報システム(Information Systems)学」のアプローチを取り入れていく必要がある[1]．このとき，プログラミング教育は，言語を習得するだけではなく，コンピュータの本質を知るだけでもなく,「自分ができる何らかのタスクを他人（または装置）がやれるように，十分に詳細に，完全にその手続きを記述する」[2]ことを目指されなければならない．

このような教育は，実際にプログラミングを利用する職業を目指していない文系の学生にも重要である．なぜなら，文系の学生といえども，社会に出て，ソフトウェアの発注や評価などに関わる機会は少なくないからである．

しかし，多くのプログラミング初学者は，プログラミングとは「コーディングをすること」や「プログラミング言語の文法を覚えること」というイメージを持って，授業を履修する．はじめに，授業で行うことの全体像を示し，我々の考える「プログラミング教育」とは何かを学生に伝える必要がある．

我々はこのことを説明する導入授業を，過去に講義形式で行っていた．講義は，プログラミングのプロセスの解説や，ソフトウェア開発におけるコミュニケーションの問題を皮肉った「ブランコの漫画」（図 1: [3]を原典とする[4]より引用.）の解説を交えながら，シラバスを紹介するという形で行ってきた．

* † 慶應義塾大学 政策・メディア研究科 .
†† 静岡大学 情報学部
††† 慶應義塾大学 環境情報学部

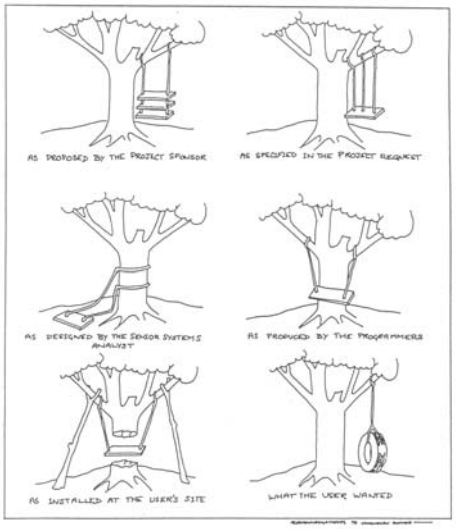


図1 ブランコの漫画

ブランコの漫画とは，30 年前に書かれた，発注者の「３つの遊び方ができるブランコを作ってほしい」という要求が，設計，実装の工程に誤って伝えられて，ユーザの要求に合わないものを作ってしまう，という，ソフトウェア開発におけるコミュニケーションの問題を皮肉った漫画である．この漫画ではユーザが本当に欲していたものは「タイヤを木にぶら下げただけのブランコ」であった，という落ちをつけている．このような，コミュニケーションの問題から使われないソフトウェアが完成する，余計なコストがかかる，といった問題はソフトウェア工学が発展した今もなお健在である．

しかし，このような解説は，プログラミング初学者には理解できない．このような問題が何故起こるのかを理解するためには，実際にコミュニケーションの問題を体験する必要がある．学生が楽しみながら，体験的に，「十分に詳細に，完全にその手続きを記述する」ことを学ぶ意欲を高めるための導入授業が必要である．

我々は，導入授業の一案として，学生が楽しみながら論理的コミュニケーションの難しさを体験できる，コンピュータを使わないワークショップ,「お絵かきプログラム開発演習」を開発し，実践している．

このワークショップでは，先にあげたブランコの漫画で示されているような，様々なコミュニケーションの問題が再現されるので，社会人でも新たな発見があり，議論を楽しめる．それだけでなく，コンピュータを使わないことから，プログラミング初心者や文系学生にも好評で，プログラミングに対する不安感を取り除く効果や，その後のクラスの雰囲気が変わり，受講者同士のコミュニケーションが円滑に進むようになるアイスブレーキングの効果もある．

本稿では，このワークショップ「お絵かきプログラム開発演習」を紹介し，このワークショップの実践方法を示し，その効果を考察する．

2. お絵かきプログラム開発演習とは

お絵かきプログラム開発演習の目的は，プログラミング初学者がソフトウェア開発におけるプログラミングの位置づけを理解し，ソフトウェア開発の過程でおきるコミュニケーションの問題を体験し，ソフトウェア開発プロセスの概要に触れることにある．

この演習は学生が授業の全体像を理解することを助け，その後の授業において，手順を論理的に記述することと，論理的な思考プロセスを踏むことへの意識を高める効果がある．

2.1 研究の位置づけ

この演習の中心となる活動は，要求を満たす絵を描くための日本語プログラムを作成することである．この演習での「プログラム」とは，仕事の手順のことである．日本語で，絵を描く手順を書いた文書を日本語プログラムと呼ぶ．言語は人間に対して使う普通の日本語である．

絵を，特定の手順に従って描くというワークショップには，他に「Kid Fax」[5] や「図形の作文ワークショップ」[6] が挙げられる．提案するワークショップは，上記のワークショップと，「コンピュータそのものの仕組みに興味を持たせ」「コンピュータサイエンスの面白さに直感的に触れる」という点で共通している．

我々の提案する演習には，コンピュータサイエンスだけでなく，情報システムの概念も取り入れている．情報システムの概念を取り入れた，コンピュータを使わないワークショップであるため,「情報システム概念に基づくアンプラグドワークショップ」と名付けた．

2.2 お絵かきプログラム開発プロジェクトとは

お絵かきプログラム開発プロジェクトは，5 名のメンバーで構成される．お絵かきプログラム開発プロジェクトに与えられる「問題」は「魅力的な絵を，誰でも，正確に，期限内に描けるような日本語プログラムを開発する」ことである．

プロジェクトは要求分析，設計，実装，テストの４つのフェーズで構成される．これらのフェーズは，施主（発注者），設計者，プログラマー，テスター(2 名)が担当するが，すべて異なる人が作業を行う．各フェーズ間のコミュニケーションは書面でのみ行われる．それぞれのフェーズにおいては，時間期限も定められ

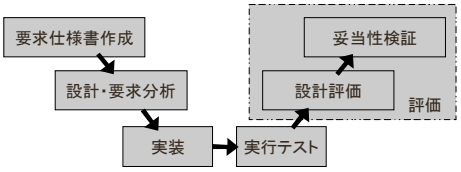


図2 プロジェクトのプロセス

ている．

プロジェクト終了後には，メンバー全員がそれぞれの立場からプロジェクトの評価を行い，プロジェクトの成功や失敗について議論する．

2.3 プロジェクトのプロセス

お絵かきプログラム開発プロジェクトは，図2に示したプロセスで進行する．

ここでは「三日月」を描く日本語プログラムを開発するプロジェクトを例にとって，実際の成果物例（図3）を挙げながら説明する．

2.3.1 要求仕様書の作成

施主が，日本語でどのような絵を描くプログラムが欲しいかを要求仕様書に書く．

2.3.2 要求分析・設計

設計者が施主から要求を受け取り，要求を満たす絵を設計し，設計書に描く．必ず絵を描くことが条件である．日本語による説明などを加えてもよい．

2.3.3 実装

プログラマーが設計者から設計を受け取り，設計された絵を誰でも，正確に，期限内に描けるようなプログラムを記述する．使用言語は日本語である．ここでは，日本語のみしか用いてはならない．

2.3.4 テスト

テスターが，プログラマーが記述した日本語プログラムを解釈し，絵を描く．

プログラマーによるプログラムの記述のよしあしと，テスターによるプログラムの解釈のよしあしを区別するために，2 名のテスターによって一つのプログラムが検証される．人間がテストを行うため，同じプログラムでも解釈や読解力の差によって違う実行結果が得られることがある．

2.3.5 評価

プロジェクトメンバー全員がそれぞれの立場からプロジェクトの評価を行う．施主は要求を満たすものができたかを中心に評価し，それ以外のメンバーは成功や失敗の原因，改善点を考察する．評価の際にはプロジェクトの成果物をすべてまとめ，プロジェクトの全体像を見られるようしてから評価を行う．

3. お絵かきプログラム開発演習の運営方法

我々は，お絵かきプログラム開発演習を授業内で効率的に実施する運営方法を開発した．この運営方法では，以下の点を重視している．

- 学生が複数の立場を体験できるようにする
- 複数のプロジェクトの比較ができるようにする
- 短時間で実施できるようにする
- 誰でも実施できるようにする

この運営方法はパッケージ化されており，説明書・ワークシートはお絵かきプログラム開発演習のホームページ

http://www.crew.sfc.keio.ac.jp/projects/2007DrawingProject

からダウンロードできる．

以下に，具体的な演習の運営方法を示す．

3.1 導入

授業のはじめに，コンピュータを使わないプログラミングワークショップを行うことを伝える．その後，ソフトウェア開発のプロセス（要求・設計・実装・テスト・評価）について簡単な紹介を行い，ワークショップで使う語句について説明する．

このワークショップでは発注者を「施主」と呼んでいる．多くの場合，「発注」という行為の具体的なイメージが湧かない．学生に「高いお金をかけて自分の欲しいものを注文し，多くの人が関わって完成する」ことをイメージしてもらうには，「マイホーム」などの建

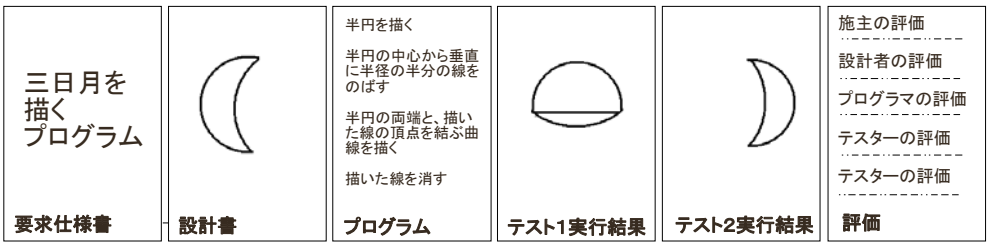


図3 プロジェクトの成果物例

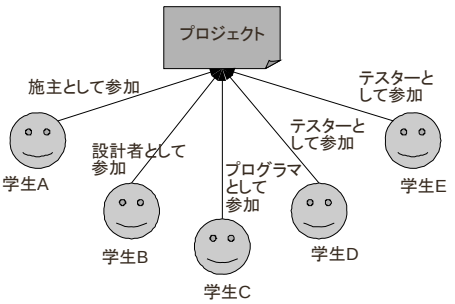


図4 プロジェクトとメンバーの関係

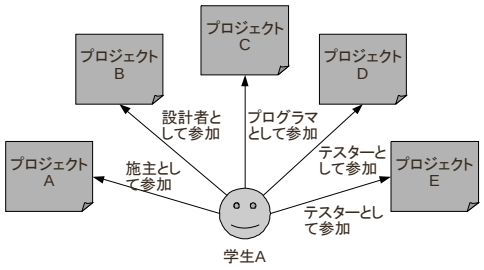


図5 メンバーとプロジェクトの関係

物の例を用いるのが適当と考え,「施主」という言葉を用いている．この演習における「設計者」は，要求を，実装の観点から検討し，要求を満たしかつ実現可能な設計を作る者として,「テスター」は手順を解釈し，実行する者として抽象化し,これらの名称を用いている．

実際のソフトウェアの設計は絵を描くことではないし，実際のテストでは，実行のたびに結果が変わることはない．用語説明ではこの点に触れる必要がある．

3.2 ワークショップの準備

次に，お絵かきプログラム開発演習の実施方法を学生に解説する．授業でお絵かきプログラム開発演習を行う場合，効率よく（60 分で）進めるために，演習の説明書を作成した．説明書は付録の図 9 に示す．

この説明書に従うと,プロジェクトのメンバー全員が施主になる時間，設計者・プログラマー・テスターになる時間があり，時間単位でプロセスが進行する．その結果，１プロジェクトには５名が参加し（図 4），１人は５プロジェクトにそれぞれ違った立場で参加することになる（図 5）．

この方法で演習を行うことにより，全員がすべての役割を体験することができ，グループ内で複数のプロジェクトの実施されるため，立場の違いやプロジェクトの比較などの議論ができる．

手順は説明書にすべて書かれている．説明書を学生に 5 分ほどで読ませ，分からないところがあったら質問をするように指示する．

図6 プロジェクト A の成果物

3.3 ワークショップの実施

お絵かきプログラム開発演習を行う．演習の実施方法は2.2節で述べたとおりである．演習時間のあいだは,指導者は説明書に書かれたタイムテーブルにしたがって時間管理を行う．指導者は議論の時間になったら，各グループを巡回し，グループごとの面白い失敗や，成功について考察をするように促す．

3.4 全体討論・まとめ

議論の時間終了後に，クラス全体で失敗や考察，成功例などについて共有する全体討論の時間を設ける．ここで，お絵かきプログラム開発演習で実際に起こった失敗に関連付けて，ソフトウェア開発で頻出する問題（あいまいな要求・非現実的な設計・プログラムのバグ・目的の共有）を取り上げ，事例を紹介する．

最後に，ソフトウェア開発のプロセスについてもう一度紹介し，ブランコの漫画を見せる．ブランコの漫画で描かれているような問題が実際のソフトウェア開発プロジェクトでも起きていることを説明する．

4. プロジェクトの例

本章では,提案する演習の実践結果について述べる．実践結果を踏まえた効果の考察は次の５章で述べる．

4.1 プロジェクト A：メロンがはかりに

プロジェクト A の結果を図 6 に示す．このプロジェクトでは，施主は「メロン」を発注している．設計者はマスクメロンを設計している．プログラマーはこの絵を「円と，T 字と，細かい線でできている図形」と分析して，プログラムとして記述している．

このプログラムをテスターが実行したところ，１人のテスターは「細かい線」の意味が分からず，独自の解釈で「はかり」だと考え,「はかり」を描いている．もう一人のテスターは，メロンを描くことができてい

図7 プロジェクトBの成果物

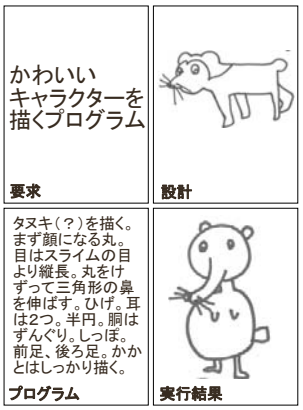


図8 プロジェクトCの成果物

た.

このプロジェクトの評価として，テスターは「メロンならば,メロンだとプログラムに書いてほしかった」と評価している．プログラマーはこれに対して「メロンと書けば，すぐに分かったはずだった」と反省している．このような失敗例を用いて，プログラミングの授業で，コメントに目的を書くことの意義についての講義を円滑に進めることができる．

4.2 プロジェクトB：美しい日本

このプロジェクトの結果を図7に示す．このプロジェクトでは，要求が「美しい日本」という抽象的なものが設定されている．設計者はその要求に困ってしまい，制限時間内で設計を済ませるために「日本列島を輝かせる」という設計を行っている．設計を受けとったプログラマーは，これを「日本列島と，小さな島と，いくつかの十字型のもの」と解釈してプログラムを書いている．

そのプログラムの実行結果は，日本列島の周りに，小さな丸い島と十字型が配置された絵になっている

このプロジェクトの評価として，施主は「美しい日本とは，紅葉や富士山，海など，旅行パンフレットに出てくるような美しい日本の景色のことだった」とコメントし，結果に満足していないと評価している．これに対して，設計者は「そんな要求だとは分からなかった，分かったとしても，５分でその絵を描くことはできない」と反論している．こうしたあいまいな要求による問題は，実際のソフトウェア開発現場でも起きている．

4.3 プロジェクトC：かわいくないキャラクター

プロジェクトの結果を図8に示す．このプロジェクトでは，施主は「かわいいキャラクター」を要求している．しかしながら，設計者の画力不足により，すでに設計の時点で，キャラクターはかわいいとはいえないものになっている．

このプロジェクトの失敗原因は，設計者が「かわいい」ものを描くための知識がなかったためだと思われる．かわいい絵を描くためには，「顔の要素を顔の下半分に集め，3 頭身以下にする」というパターンがある.設計者がこのパターンを知っていれば,「かわいい」という要求から，適切な絵を設計することができたであろう．施主は「ぜんぜんかわいくない」と，この結果を不満足と評価している．しかしながら，要求記述という観点からは，設計者の前提知識を考慮せず「かわいい」というあいまいな要求をした発注者にも問題があるだろう．

このように，単純な言葉遣いのミスや，背景知識の違いにより，意図と異なるものになるプロジェクトが多い．これは受講者には意外であり，結果の評価は大いに盛り上がり，指定された時間内に議論が終わらないことも多く，全体討議では積極的に面白い失敗例が発表される．

5. 演習の効果に関する考察

4章で紹介したように，抽象的な要求，目的の書かれていないプログラムなど，実際のソフトウェア開発で起こる問題をこの演習を通じて体験することができる．

演習の体験が，その後のプログラミング授業に与える効果を，(1)論理的なコミュニケーションの難しさが分かる，(2)ソフトウェア開発プロセスの概要が分かる，(3)アイスブレーキングの３つの観点から考察する．

5.1　論理的なコミュニケーションの難しさが分かる

5.1.1 コメントの必要性がわかる

コメントは，ソースコードの可読性を高めるために必要である．しかし，学生はコメントの必要性が理解できず，何をコメントとして書けばよいのかが分からない．

演習では，「メロンがはかりに（4.1節）」の例のように，細かい手順を書くことに集中してしまい，目的を書かない学生が多い．演習を通じて，目的を書くと，何をするプログラムなのかが人に伝わるということが分かり，目的を書く必要性がわかる．

「メロンがはかりに（4.1節）」の事例を紹介し，コメントの必要性と，コメントにはどのようなことを書いたらいいかを教えると，学生はコメントの必要性を理解することができる．

受講者の反応の例として，ソフトウェア開発経験のある学生が，演習にプログラマーとして参加して失敗した経験から，「目的を書いていいんだ！」という感想を述べた．この学生は，演習において，プログラムには目的を書いてはいけない，と誤解しており，手順だけを詳細に記述していた．しかし，他のプロジェクトの例から，目的を書いたプログラムは読みやすく，意図が伝わりやすい，ということを学び，実際のソフトウェア開発でも，コメントに目的を書くことで，読みやすいソースコードにすることができるということを発見することができた．

5.1.2 手順を正確に説明することの難しさが分かる

演習に参加する学生の多くは，仕事の手順を正確に記述するという経験がない．

演習を通じて，手順を正確に記述することは難しいということ，手順を正確に記述するためには，自分がその手順を理解していなければならないことが分かる．同時に，正確でない手順は，間違って実行される，ということが分かる．

実際の例として，プログラムのテストを担当した学生が，「手順が正確に書かれていないと，実行できない！」と発言した．こうした発言は「プログラムは正確でないと動かない」ことを，コンピュータの立場に立って理解するきっかけになると考える．

授業で演習の失敗例を取り上げることで，プログラムは正確でなければ意図どおりに動かないということ，プログラムを書くためには，自分自身がプログラムでやりたいことの手順を理解していなければならないことを伝えることができる．

5.1.3 要求を表現することの難しさが分かる

学生は，要求を伝わるように表現する機会はほとんどない．プログラミング入門授業で，要求の書き方が分からず，学生が「要求って，どう書けばいいんですか？」という質問を発した．スタッフは書き方を詳しく指示せず，「読む人に伝わるように書くように」とだけ指示した．この学生にとって，自分の要求をどのように書いたら伝わるのかを考えることは初めてであった．このような学生に対して，要求分析と，要求の記述が重要である，と説いても理解は難しい．

この演習を通じて，「美しい日本」プロジェクトのように，要求があいまいなプロジェクトはうまくいかない，ということを体験的に理解することができる．この事例を紹介して，冒頭にある「ブランコの漫画」や実際のソフトウェア開発の問題を論じると，学生は発注者の要求がソフトウェア開発におよぼす影響について理解しやすい．

5.2　ソフトウェア開発プロセスの概要が分かる

学生は「設計」「実装」「テスト」というプロセスや用語になじみがない．このようなプロセスは個人でのプログラミングにも活かせるが，プログラミングの入門教育で，設計やテストについて知る機会はあまりなく，そもそも用語すら知らない場合が多い．

要求・設計・実装・テストとは何かの概要を学生が理解した例をいくつか述べる．お絵かきプロジェクトの評価フェーズで，プログラマーから「設計が複雑すぎると，時間内に描けない」という意見が出た．設計する際に，要求だけでなく，実装（実現可能性）も考慮しなければならないことへの気づきが伺える．他にも「要求を満たしていないので，プロジェクトは失敗だと思う」「設計が良かった．ここが成功の原因だ」など，ソフトウェア開発プロセスの概要を掴み，議論に使用する姿が見られる．

この演習を通じて，学生は要求・設計・実装・テスト・評価とは何かの概要をつかめる．同時に，コーディングはソフトウェア開発のプロセスの一部分に過ぎないといる理解が得られる．演習の例を用いることで，学生がソフトウェアを作成する際に，自分が現在行っている作業がソフトウェア開発のプロセスのどの部分に位置付けられるのかを意識させることや，次に学ぶべきことへの興味を引き出すことが可能である．

しかし，この演習で用いている用語は，あくまで「お絵かきプログラム演習においての要求・設計・実装・テスト・評価」であるので，その後の授業で，実際のソフトウェア開発プロセスで行われることはどういったものかについてや，このプロセスは個人でのプログ

ラミングにどのように活用できるのか，といった解説が必要である．

5.3 アイスブレーキング

本演習はコンピュータを使わず，面白い絵が描かれて議論が盛り上がるという特徴があるため，コンピュータに苦手意識のある学生も取り組みやすく，プログラミングに対する姿勢をポジティブにする効果がある．

以下に演習に参加した学生の感想を掲載する．

5.3.1 不安感や苦手意識の払拭

「プログラミング＝数学＝苦手＝やりたくない」という意識を持っていたが，初回の授業で楽しい演習があり，「日本語の手順もプログラムの一種」と言われ，自分でもできそう，と思った（プログラミング初学者の感想）

なによりも一番印象的だった授業は，第１回目の授業です．プログラミングなんてもってのほかだと私は思っていたので，ただ必修だから取るという感じでいました．そこに待っていたのが，何やら楽しいゲーム．こんなにも楽しいものなのかと思いました．導入としてはすばらしい授業だったと思います．（プログラミング初心者の文系学生）

5.3.2 アイスブレーキング

グループメンバーと面白い失敗について盛り上がり，その後の授業でも相談できる友達ができた．（プログラミング初学者の感想）

5.4 注意点

本ワークショップには，他人がやれるように手続きを記述するために必要な要素である，論理的なコミュニケーションの難しさを体験的に理解させる効果がある．しかし，ワークショップを行うことで「十分に詳細に，完全に手続きを記述できる」ようになるわけではない．この演習は，直接的にその能力を伸ばすものではなく，あくまで，そのために学ばなければならないことがある，ということへの気づきを与えるものである．

この導入を活かすためには，演習の後に「自分ができる何らかのタスクを他人（または装置）がやれるように，十分に詳細に，完全にその手続きを記述する」ことを目指したプログラミング教育[7][8]を行う必要がある．

6. おわりに

本稿では，学生が楽しみながら論理的コミュニケーションの難しさを体験し，プログラミングを通じて学ばなければならないことへの気づきを与える，コンピュータを使わないワークショップ,「お絵かきプログラム開発演習」を紹介し，効果を検証した．

実践結果から，このワークショップには，(1)論理的コミュニケーションの難しさが理解できる，(2)ソフトウェア開発プロジェクトのプロセスの概要を知ることができる，(3)アイスブレーキング，の３つの効果があり，このワークショップを経験することで学生の「手順の記述」への意識が高まりその後のプログラミング授業を円滑に進められることが分かった．

今後はワークショップにおける学生の感想からヒントを得て，ワークショップ後に行う授業をより多く考案し，ワークショップの効果を活かす授業作りを目指す必要があるだろう．

謝辞　研究の目的や位置付けについて考えを深めるきっかけとなる査読をしていただいた査読者の方に感謝いたします．

参考文献

1) 大岩元: 情報教育と２１世紀の教育， SSS2007 論文集 (2007)
2) UNESCO .ICT Curriculum for School /Program of Teacher Development. (2002) http://unesdoc.unesco.org/images/0012/001295/129538e.pdf
3) University of London Computer Center Newsletter, No.53, March 1973
4) 有澤誠:　ソフトウェア工学, 岩波書店 (1988)
5) Bell, T . , Fellows, M .& Witten, I. , 兼宗進 訳: コンピュータを使わない情報教育 アンプラグドコンピュータサイエンス,イーテキスト研究所 (2007)
6) 佐藤雅彦, 村上泉, 山本晃士, 菅俊一, 石川将也, 佐藤匠: コンピュータサイエンスにおける教育原理, 平成 14 年度ハイテク・リサーチ・センター報告書, pp.488-501 (2004)
7) 松澤芳昭, 杉浦学, 大岩元: Squeakで学ぶ論理思考とプログラミング, イーテキスト研究所 (2008)
8) 杉浦学, 松澤芳昭, 大岩元: アルゴリズム構築能力育成の導入教育：実作業による概念理解に基づくアルゴリズム構築体験とその効果, 情報処理学会論文誌 Vol.49（掲載予定） (2008)

付録

問題：『魅力的な絵を、**期限内に、正確に、だれにでも描けるような**日本語プログラムを作成しなさい。』

演習の手順：

学生ひとりひとりが、施主・設計者・プログラマー・テスター（プログラムの実行者）のすべての役割を体験し、プログラムを作る。書かれた設計書・プログラム・実行結果は最後に施主に返され、施主が評価を行う。

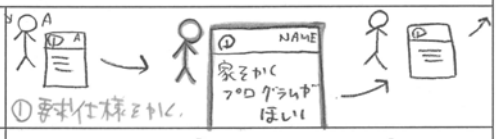
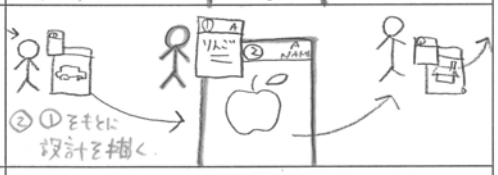
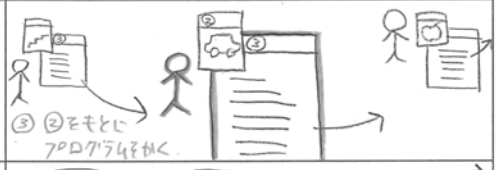
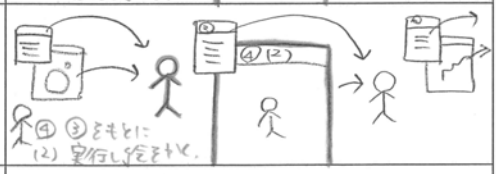
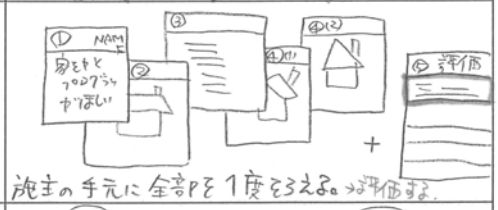
タイムテーブル	
施主になる時間（2分） ひとりひとりが、要求仕様を①の要求仕様書に書きます。時間がきたら、①を隣の人にわたします。	
設計者になる時間（5分） もらった①をもとに、②の設計仕様書に設計を描きます。絵を必ず描くこと。時間が来たら、②を隣の人にわたします。	
プログラマーになる時間（１０分） もらった②をもとに、②を描くプログラムを日本語で書きます。絵や図は使ってはいけません。日本語のみです。時間がきたら、③を隣の人にわたします。	
プログラムの実行テストをする時間（５分） ③のプログラムをもとに④のテスト結果用紙に絵を描きます。時間がきたら、③のプログラムを隣の人にわたします。	
プログラムの実行テストをする時間（５分） ③のプログラムをもとに④のテスト結果用紙に絵を描きます。時間がきたら、③のプログラムと④の絵を施主に渡します。	
評価の時間（３分×５） 施主が、自分が施主になっている紙をすべて集め、評価用紙をかさねてクリップで留めます。書き込みます。３分たったら、紙の束を隣の人に渡します。これを５回繰り返します	
ディスカッションの時間（１０分） 結果についてディスカッションを行います	

図9 演習説明書