E-11

プロジェクトにおける活動分析
-お絵かきプログラミングによる入門教育とPBL-
Activity Analysis on PBL: Drawing Programming Class for Beginners

伊地知 咲希† 永井 絵梨奈† 内田 奈津子††
Saki Ijichi Erina Nagai Natsuko Uchida

1. はじめに

社会環境の変化により，時代が変わるにつれて社会が求める人材像にも変化が起こっている．IT化に伴い，単純作業が機械化される中で協調性や多様性が重視されるようになってきた．今後の社会において課題を発見し，それらを協力し合って解決する力，多様性を生かしコミュニケーションを取りながらものごとを進める力が求められている．[1]また，情報活用能力の育成においても，プログラミングを学ぶだけでなく，プロジェクトを行う上で必要な目標設定・情報伝達・役割分担・計画性・課題解決能力が重要視される．[2]

著者らは，お絵描きプロジェクトを通じてプログラミングを学んだ．実際にプロジェクトを体験し，プログラミングに触れて学ぶことにより，問題を発見し，それらをどう解決したかについて，学んだ．これにより，上述した様々な能力をどのように学んだのかについて，学習者の視点から考察する．

その結果として，プログラミングを学ぶことにより，物事を進める手順についての仕組みへの理解が深まる．また，チームでプロジェクトを行うにあたり，特性を生かした役割分担やコミュニケーションを行う事で，社会において必要な能力やスキル向上にも繋がっていることがわかった．

2. 先行研究

荒木らは，[3]で，お絵かきプログラミング開発演習では，コミュニケーションやソフトウェア開発の難しさを，パソコンを使わずにプログラミングを用いて学ぶことを実践している．荒木らの実践方法は，図1で示すように，４つのフェーズ，要求分析，設計，実装，テストで構成される．プロジェクトは要求分析，設計，実装，テストという４つのフェーズから構成されている．

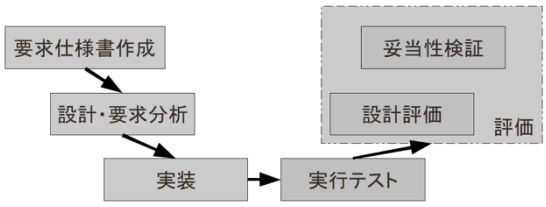

図1 プロジェクトのプロセス

「プログラミングを通じてソフトウェア開発やコミュニケーションの難しさとは何かを知るには，技術力と，それを習得するための長い時間が必要である．それまでに退屈な言語の使い方だけを覚えさせられては，プログラミングはおろか，コンピュータも嫌になってしまう，」という荒木らの記述についての動機は，著者らも同じである．

荒木らは，図2で示すように，絵を描くことによりコンピュータを使わない方法でワークショップを行った．荒木らのプロジェクトは，要求仕様書の作成→要求分析・設計→実装→テスト→評価というような一連のプロセスで行われた．メンバー5名のうち設計者と実行者の二名を選出して行う．まず，設計者が要求仕様書に書かれた注文から，図や文字を用いて設計図を描く．その後，文字のみでプログラムを書き，実行者はそれをもとに実際に図を作成する．そして，このプロセスをメンバー全員がそれぞれの立場からプロジェクト評価を行う．この体験から，正確な情報伝達の重要性やコミュニケーション能力の強化を目指す．

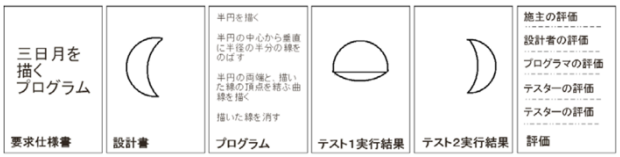

図2 プロジェクトの成果物例

著者らは，実用言語Rubyを用いて，実際にプログラミング言語に触れながらの学習であるため，方法が異なる．

したがって，本論文では，実際にプログラミング言語を用いたお絵描きプロジェクトによって，何を身に付けたか，どのように学んだかについて考察する．

3. 講義の概要

3.1. プログラミング入門をどのように学んだのか？

本プロジェクトは，2019年度後期に，フェリス女学院大学国際交流学部 国際交流専門科目「情報発信と世界」において実施された．1～4年生までが履修可能であり，解放科目であるため音楽学部生や文学部生などの他学部生も履修している．

本講義の目的は，以下の通りである．

1. プログラムの仕組みを学び，簡単なプログラムがつくれるようになる．
2. PBL(Project based Learning)の実践を通して，課題解決能力，コミュニケーション能力などを身に付ける

講義は，全15回で構成され，図3に示すような流れで，進められた．前半の1～6回は，プログラミングの基礎知識を学んだ．後半の7～15回では，グループに分かれてのプロジェクトを行った．　図4に，全15回の講義内容につい

† フェリス女学院大学　国際交流学部
†† フェリス女学院大学　情報センター

て示す.

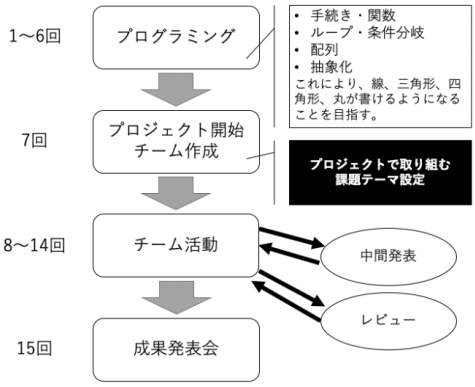

図 3　講義の流れ

回数	内容
1 回目	プログラミングの導入
2 回目	分岐と繰り返し
3 回目	制御構造と配列
4 回目	手続き・関数と抽象化
5 回目	2 次元配列と画像
6 回目	プログラミングのまとめ
7 回目	グループ作成・課題とゴール設定
8 回目	アイデア出し・ブレスト・仕様決め
9 回目	デザイン・設計・分担・制作
10 回目	試作・中間ドキュメント作成
11 回目	中間発表・仕様見直し
12 回目	制作・単体テスト
13 回目	コードレビュー
14 回目	制作・統合テスト
15 回目	最終成果発表会 (まとめ)

図 4　講義内容

前半は，図 5 に示すように，第 1 回から図形を生成するメソッド(図形ライブラリ)を利用し，図を描く教材を用いて簡単な絵を描く．図形ライブラリは，丸，三角，四角，楕円によって構成されている．

回数	項目	内容
1回目	プログラミングの導入	プログラムとは何であるかを理解し，簡単な絵を描く
2回目	分岐と繰り返し	絵を描くことを通じて制御構造を理解し，そのプログラムを書く
3回目	制御構造と配列	配列について理解し，それを用いたプログラムを書く
4回目	手続き・関数と抽象化	手続き (メソッド) の理解と抽象化について理解し，各自の考えた絵を手続きを用いて書く
5回目	2次元配列と画像	配列について理解を深め，2 次元配列と動画の仕組みについて理解する
6回目	プログラミングまとめ	5 回の学習内容を理解し，プロジェクトにつなげる

図 5　前半の講義内容

そして，第 2 回から第 5 回まで図の生成を教材に，プログラミングの基本的知識を学んだ．続けて第 6 回目には，アニメーション作成の方法についても学習した．

3.2.　プロジェクトの過程

後半の 7 回目以降では，テーマを「横浜」とし，4 名 1 チームで 4 チームに分かれてプロジェクトを行った．プロジェクトの課題は，チームごとにテーマに沿ったステッカーとアニメーションを作成することであった．

第 1 著者は，それぞれ 4 枚の絵とアニメーションを作成した．また，1 人 1 枚ずつ絵を作成するのではなく，4 枚の絵を全員で作成するという条件のもとで作業を行った．

第 1 著者のチームでは，「夜の横浜」をテーマに選び，横浜を訪れる「カップル」を対象とした．このターゲティングの目的としては，神奈川県の離婚率が年々増加してきていること [4]に着目し，横浜がデートスポットとして魅力的である事をアピールしたいと考えた．

そこで，横浜市の魅力として挙げられる①街並み景観②夜景をモチーフとし，連想されるモチーフをシンボルとした絵を作成することとした．

第 1 著者のチームを例として，プロジェクトの過程を説明する．

メンバーは 4 年生 1 人，3 年生 1 人，2 年生 2 人の計 4 人の構成であった．

プロジェクトは，ソフトウェアの開発工程に習い，シラバスに沿って進められた．第 1 著者のチームの取り組み概要を，以下に示す

1. メンバーの役割決定
   リーダー・記録・報告・状況確認の 4 つの役割に分かれた
2. テーマと 4 枚のモチーフ決定
   「夜の横浜」から連想された，観覧車・ランドマークタワー・レストラン・赤レンガ倉庫をモチーフとして決定した．
3. 4 枚のモチーフ担当決定
   決定したシンボルの担当をそれぞれ決め，それを中心とした絵を 4 枚作成
4. ステッカーの試作開始
   ・モチーフ内の分担決定
   1 枚に付きシンボルは 1 つで，他 3 人はそれぞれ背景・装飾・統合して 1 枚の絵にする担当を振り分けた．
   ・アニメーションの構想
5. ステッカーのメソッド統合
6. アニメーションの試作

図 6 に，ステッカーの完成作品を示す．

図 6　完成作品

役割分担は，以下のとおりである．加えて，図 7 の日本丸を例として，役割分担を示す．

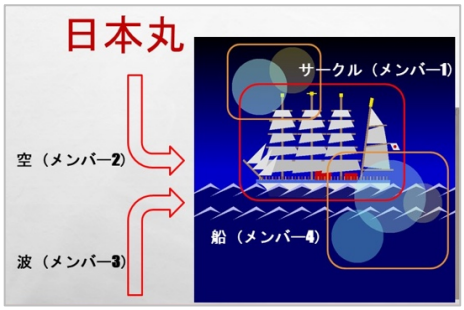

図 7　日本丸

1. レストラン（統合　メンバー3）
   グラス（メンバー2），背景（メンバー1），シャンデリア（メンバー4），机など（メンバー3）

2. 日本丸（統合　メンバー1）
   船（メンバー4），背景（メンバー2），海（メンバー3）

3. 赤レンガ（統合　メンバー2）
   赤レンガ本体（メンバー3），背景（夜空　星）（メンバー2），柱（メンバー4），窓（メンバー1）

4. 観覧車（統合　メンバー2）
   観覧車（メンバー1），背景（メンバー4），ランドマーク（メンバー3）

このように，役割分担を行うことは，ソフトウェア開発の現場では，部品を作り，それを統合することで大規模なソフトウェアが開発されていることを学んだ．

第 1 著者のチームの大半は，担当のモチーフを仕上げる作業であった．そこで当初は 4 つの絵にそれぞれ異なる装飾を加えようと考えていたが，モチーフづくりに予想以上に時間がかかった．そのため，図 7，図 8 のようにメンバーそれぞれが一つずつ作成した図 9，図 10 のような簡単な装飾を全ての絵に再利用することで，4 枚の絵を全員で作成するという条件を満たし，期日に間に合うように完成させることが出来た．

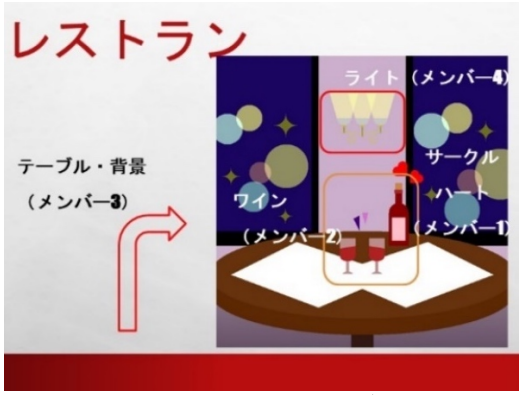

図 8　レストランの分担

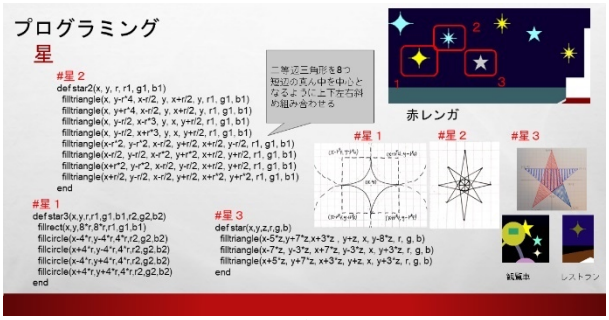

図 9　装飾 星

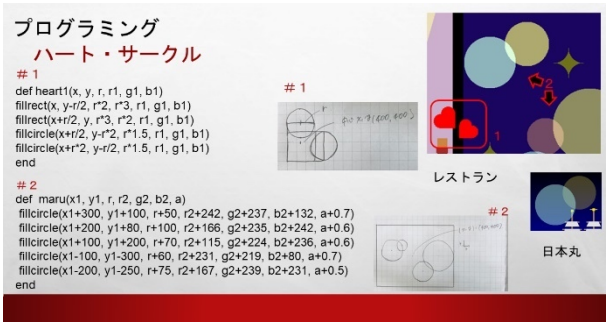

図 10　装飾 ハート・サークル

4.　プログラミングによる活動の分析

4.1.　プログラミングに触れてみえてきたもの

プログラミングに触れての一番の変化は，プログラミングに対してのイメージが変わったことである．プログラミングに対して，初見では文字や数字が沢山並んでいるだけのものにしか見えなかった．しかし，プログラミングをお絵描きプロジェクトとして学ぶことで，どのような仕組みで構成されているか，数字や文字の配列の意味などの理解を深めることが出来た．

特に，四則計算を用いる作業においてこのことが良く言えるだろう．図 11 では，加算のみを用いた式である．

引数の x，y，r を変更することで，図の位置を変更することが可能である．アニメーションを作成する場合は，背景と他の絵を同じ大きさで統合して完成させないといけないため大きさの変更が出来ない．そこで，図 12 のように，乗法で表現することにより縮尺を自由に変更することが可能となり，乗法加法の活用の理解に繋がった．

これらの理解は，小学生の授業内容と合わせて学ぶことで，四則演算を実践しながら学習し学びの価値を感じられるのではないかと考える．

```
def ha(x, y, r, r1, g1, b1)
fillrect(x+5, y+30, 40, 50, r1, g1, b1)
fillrect(x, y+25, 40, 50, r1, g1, b1)
end

def hi(x, y, r, w, h, r1, g1, b1)
fillcircle(x, y, r+10, r1, g1, b1)
fillcircle(x+30, y+30, r+10, r1, g1, b1)
end
```

図 11　加算法のみを用いた式

```
def
  line(x0, y0, x1, y1, w, r, r4, g4, b4)

  fillline(x0, y0, x1-r*2, y1+r*7, w, r4+50, g4+125, b4+100)
  fillline(x0, y0, x1+r*2, y1+r*7, w, r4+50, g4+125, b4+100)
  fillline(x0-20, y0+r*7, x1+r*2, y1+r*7, w, r4+50, g4+125, b4+100)

end
```

図 12　乗法を用いた式

お絵描きという方法を用いることで，丸や四角などで簡単に表現できる．そのため出力された絵の構造や，絵の出力に失敗したときにどう間違えたのか，が分かりやすいというのが利点である．

しかし，手順ややりたいこと(伝えたい事)の矛盾が生じると想像する絵と違うものが生成される．絵が生成される前にエラーとして，どこが間違っているかの表示が出ることで間違いに気づきやすい．例えば図 13 のようなコマンドプロンプトによるエラーでは，何が違うのかが文字列によって可視化される．

```
コマンド プロンプト - irb
Microsoft Windows [Version 10.0.18362.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:¥Users.              cd Desktop

C:¥Users.              ¥Desktop>irb
irb(main):001:0> convert pict1.ppm pict1.gif
Traceback (most recent call last):
        2: from C:/Ruby25-x64/bin/irb.cmd:19:in `<main>'
        1: from (irb):1
NameError (undefined local variable or method `pict1' for main:Object)
irb(main):002:0>
```

図 13　コマンドプロンプトによるエラー

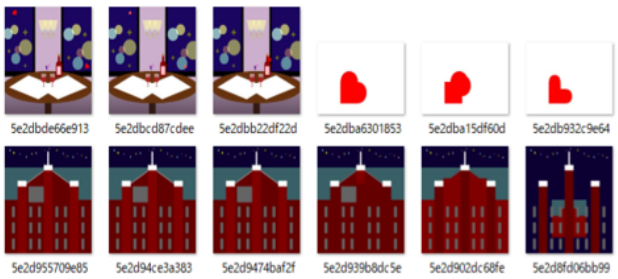

図 14　図生成後のエラー

また，図 14 のように，生成後には図としてエラーが可視化されることで間違いが分かりやすい．しかし，全てが完成してから実行すると，どこが違うのか分かりづらい．そのため，パーツごとに形成し，試作を行いながら完成まで進めることの大切さを学んだ．また，プログラミングによるお絵描きを行うことで，複雑な構造のものでもきちんとルールに従えば再現可能であることを実感することができた．

4.2.　グループメンバーで協力して一つの絵を完成させる作業

授業時間内での完成は難しく，時間外での作業が多かった．そのため，直接対話する事よりリモートでの情報交換を行った．しかし，作業を進めるうちに文章上での情報には限りがあり，上手く伝わらないという問題が出てきた．図 15 は，観覧車の設計図であるが，数字などの説明が一切記載されておらず，イメージは伝わるが，これだけではどのようにコードを書いたら良いのか分からない．

一方，図 16 の赤レンガ倉庫の設計図においては，一マスの大きさや中心座標などがきちんと明記されている．

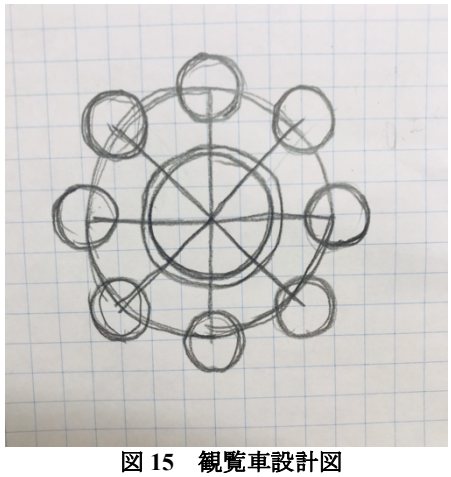

図 15　観覧車設計図

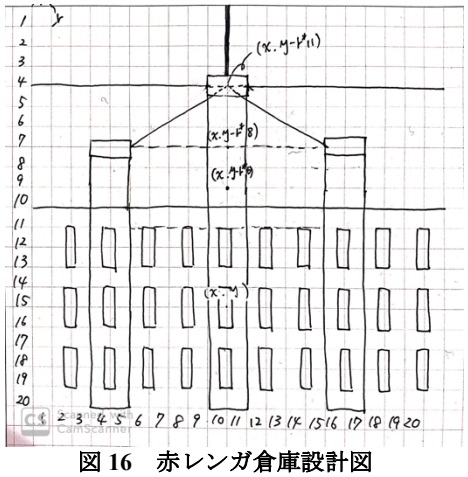

図 16　赤レンガ倉庫設計図

自分だけが理解出来る設計図では，伝わる情報が少なく，行いたいことをチームのメンバーに伝えることが困難である．情報を的確に伝え，共同作業を行う上では，全員で意思相通を図れているかの確認が大切である．そのためには図 16 のように，中心座標や一マスの数値など，細かい説明をきちんと明記することで分かりやすくなる．

設計図での失敗を踏まえて，コードを書く際には，図 17 のようにプログラムの途中に#を付けて補足説明を加えることで，統合時に他の人へ委託した際にパーツの構成が分かりやすくすることを心がけた．

```
#この下に、各自のプログラムを書くこと。レポートには、ここから下をのせること。
def slight(x, y, r)
#試作の段階での黒背景です。合わせるとき消してください。
  fillline(x,y+7*r,x+18*r,y+7*r,15*r,0,0,0)
#三つのライトを書いています。
  fillline(x+r*3,y+r*10.5,x+r*4,y+r*10.5,r*1/4,250,250,153,0.0)
  fillline(x+r*14,y+r*10.5,x+r*15,y+r*10.5,r*1/4,250,250,153,0.0)
  fillline(x+r*4,y+r*10.5,x+r*9,y+r*11.5,r*1/3,250,250,153,0.0)
  fillline(x+r*9,y+r*11.5,x+r*14,y+r*10.5,r*1/3,250,250,153,0.0)
#三つのライト下の縦の棒
  fillline(x+r*4,y+r*9,x+r*4,y+r*11,r*1/3,200,200,0,0.0)
  fillline(x+r*9,y+r*9,x+r*9,y+r*12,r*1/3,200,200,0,0.0)
  fillline(x+r*14,y+r*9,x+r*14,y+r*11,r*1/3,200,200,0,0.0)
#左下の傘の柄のような飾り(rgb値が黒になっているものは、合わせるとき変更をお願い
  fillcircle(x+r*6,y+r*13,r,250,250,153,0.0)
  fillcircle(x+r*6,y+r*13,r*3/4,0,0,0,0.0)
  fillline(x+5*r,y+12.5*r,x+7*r,y+12.5*r,r,0,0,0,0.0)
```

図 17　プログラム作成時の補足

中間発表では実際に作成したプログラムについての説明を行った．この際に，作成した絵の構造を人に説明をすることの難しさを感じた．また，自分一人で問題解決が行えなかった際に，どこが理解できなかったか，どうできなかったかの説明を細かくすることで，相手との意思疎通を図ることができる．発表も相談も，明確な情報伝達を行うことの重要性を感じた．

このことは，社会人としても必要なスキルであると考える．人に分かりやすく説明をするという点において，ビジネスシーンに限らずコミュニケーションを取る上で，人と人との理解を深めるのに必要不可欠な能力であると言える．このように設計図一つをとっても，伝えるべき最小限の情報を記載し，より相手に伝えようとする配慮が大切であることが分かった．

4.3. 本講義とワークを伴う他の講義との比較

これまで，著者らは，グループワークを伴う様々な講義を履修してきた．他の講義でのグループワークでは，あらかじめ設定されたゴールに向かって単にグループで協力して行うものが多かった．

本講義では，お絵描きという点で自分たちで自由にゴールの形を設定出来た．しかし，ゴールを自由に設定できるため，目標を高くしすぎて間に合わなくなるという問題点もある．

第一筆者のチームでは，頻繁に状況確認をすることによって，進捗状況の確認をすることが出来た．加えて期限ややるべきことの計画立て，修正など，少人数だからこそ柔軟に対応できスムーズに進められた．

また，きちんと役割分担を行わないと，グループの中で何もしない人が出てくる可能性がある．水山らは「適切に設計されない『場』は金銭や時間を浪費する存在と成り果てる．複数の関係者に分散している情報や知識を集合知として有機的に活用するためには，それらを支えるコミュニケーション場のメカニズムを適切に設計することが重要である．［5]」と示している．

このように，役割があることで自分の出来ること，役に立てそうなことを探し，自然と居場所を作ることで，チームとしての団結力を生むことに繋がる．このことは，今後において生かせるスキルである．

4.4. 社会に求められる力

文部科学省は，社会環境の変化における求められる人材像として，「学んだ知識を現場に適用し有効に活用していくための能力として課題発見・解決力」，「コミュニケーション能力」等，いわゆる「社会人基礎力」として括られる要素［6]を挙げている．

社会人基礎力とは，職場や地域社会の中で仕事を行っていく上で必要な基礎的な能力をいい，経済産業省［7]では社会人基礎力を，「前に踏み出す力（主体性・働きかけ力・実行力）」，「考え抜く力（課題発見力・計画力・創造力）」，「チームで働く力（発信力・傾聴力・柔軟性・情況把握力・規律性・ストレスコントロール力）」として，12 の要素からなる 3 つの能力として定義し，共通言語として発信している．

これからの人生 100 年時代において，ライフスタイルに合わせて活躍するために自ら行動し，考え，協調性を持って課題解決をする能力が不可欠であると考える．

本講義においても，チームで一つのプロジェクトを行ったことで社会に求められる力を育成することが期待されていた．チームでものごとを進めていく中で，集まったメンバーのそれぞれ特化した能力を生かし，役割分担を行う事でよりスムーズに作業を行う事が出来た．

これらは社会に出てから継続的に必要とされるものであるため，講義を通して学生の内から学んでおくことは大変重要であるといえる．

著者らは，第 15 回仮想政府セミナー「政府における RPA 導入の実践と課題 ～英国政府の経験に学ぶ～」［8]に参加し，RPA について学んだ．RPA とは，PC などを用いた事務作業の一連を自動化することができる技術である．そこでは，世界規模での RPA 導入について議論がなされた．現在では生産人口減少のため RPA の導入が積極的に行われている．しかし， RPA を導入する上で，ただ単に単純作業を自動化することで効率化を図るのではない．

ポッターは，「テクノロジーの一つの分野としてオートメーションを考えると、RPA 自体はクレバーなデジタルツールとして存在していますが、業務における応用局面では技術的な専門家を自組織に探す、あるいは保有する必要はありません。なぜなら技術的な専門家は外部にたくさん存在しているからです。むしろ技術とオペレーションの間に立って翻訳ができる人を組織の中で探していく必要があると考えています。オペレーションのあり方、ポリシー、課題を理解し、技術とオペレーションの間の通訳を果たせるかどうかだと思っています。技術そのものの導入が重要なのではなく、公共サービスにおける職員の業務がどのようなものなのか、ユーザーがそれをどう使っているのか、そこに発生している課題は何なのかを知ることが重要です。そこにどうやって技術を使い改善していくかを見極める力が必要なスキルだと考えています。 [8]」と述べている．

導入対象の現場での問題・課題や人々の動きを把握することで，その場に適した RPA の導入により，単純な自動化を超えた効率化を図ることが可能である．そのために，コンピュータやプログラミングに理解がないことで，専門家に丸投げするのではなく，専門家と一緒に議論が出来るその場の基礎的知識を備えた人材が求められている．

本講義において，社会人基礎力として社会に求められている力を身に付けることは，大変重要な経験であるといえる．RPA 導入による効率化のように，多様な人々や多様な能力を掛け合わせることで，より可能性の拡大に繋がる．

5. まとめ

著者らは，本講義において，プログラミングを学び，お絵描きプログラミングでプロジェクトを行ったことによって行動が可視化され，システム開発における工程への理解が深まった．この方法は，文系学生がプログラミングの仕組みを理解するだけでなく，システム開発の工程を知るためのきっかけとなった．

特に，次の項目に関しては，深い学びや気づきがあった．

1. 設計図の重要（明確な情報伝達，コミュニケーション）
2. 特性を生かした役割分担
3. 試作の大切さ(トライアンドエラー)

これらを行うことにより，効率的かつスムーズに作業を行うことが出来ることが経験できた．

また，目標設定を自由に自分たちで行えることにより，期日から逆算して最終目標を適宜変更することも大切であることがわかった．グループメンバーと頻繁にコミュニケーションをとることで，協力して目標を達成することが出

来た.

４つの図形を組み合わせた単純な教材によるプロジェクトであったが，これだけ多くのことを吸収することが可能であった.

また，これらは社会で必要とされるスキルであり，今後，社会に出てプロジェクトに関わる際に大変役立つものであると考える.

前に踏み出す力(主体性，働きかけ力，実行力)，考え抜く力(課題解決力，計画力，創造力)，チームで働く力(発信力，傾聴力，柔軟性，状況把握力，規律性，ストレスコントロール力)の３つから成る社会人基礎力の全てを，本プロジェクトでは取り入れている．履修者全員が，全てを身につけたわけではない．しかし，学生のうちから社会に求められる力を認識し，強化することによって，学生生活と社会間でのギャップに慣れることができ，どの業界においても活躍できる可能性を広げることができたと考える.

参考文献

[1] 経済産業省　産業人材政策室，“人材像 WG 参考資料集，” 12 2019. [オンライン]. Available: https://www. meti. go. jp/committee/kenkyukai/sansei/jinzairyoku/jinzaizou_wg/pdf/006_s01_00. pdf. [アクセス日: 10 7 2020].

[2] 文部科学省 初等中等教育局 情報教育・外国語教育課 情報教育振興室，“小学校プログラミング教育の趣旨と計画的な準備の必要性について，” 文部科学省，10 2 2020. [オンライン]. Available: https://www. mext. go. jp/content/20200210-mxt_jogai01-100013292_01. pdf. [アクセス日: 9 7 2020].

[3] 恵. 荒木，芳. 松澤，学. 杉浦，元. 大岩，“プログラミング授業の導入としての「お絵かきプログラム開発演習」，” *日本教育工学会研究報告集，* 第 3，pp. 111-118，2008.

[4] 神奈川県ファイナンシャルプランナーズ協同組合，“多様化の時代を生き抜く、おひとり様のライフプランを考えよう，” 30 5 2017. [オンライン]. Available: https://www. fp-kanagawa. com/information/info. php?no=685. [アクセス日: 21 July 2020].

[5] 水山 元，谷口　忠，“特集―集合知メカニズムとコミュニケーション場の設計と応用，” 日本経営工学会論文誌，第 65 巻，第 3 号，P. 143，2014.

[6] 文部科学省　科学技術・学術政策局基盤政策課，“社会環境の変化と求められる人材像，” 2009. [オンライン]. Available: https://www.mext.go.jp/b_menu/shingi/gijyutu/gijyutu10/siryo/attach/1335152.htm. [アクセス日: 8 July 2020].

[7] 経済産業省　産業人材政策室，“「人生 100 年時代の社会人基礎力」と「リカレント教育」について，” 19 March 2018. [オンライン]. Available: https://www.meti.go.jp/report/whitepaper/data/pdf/20180319001_3.pdf. [アクセス日: 9 July 2020].

[8] 東京大学公共政策大学院，“第 15 回仮想政府セミナー「政府における RPA 導入の実践と課題 ～英国政府の経験に学ぶ～」，” 3 December 2019. [オンライン]. Available: http://www.pp.u-tokyo.ac.jp/events/2019-10-31-22691/. [アクセス日: June 2020].

[9] ジェームズ・メリック・ポッター，*英国内閣政府が進める業務自動化への取り組み-RPA 導入を Huma centered の視点で推進する Center of Excellence-，* 行政&情報システム，2019 年 8 月号，一般社団法人 行政情報システム研究所，August 2019.