プログラミングの学習環境に関する研究
（既存の授業システムとの比較）

Research on Programming Learning Environments
(Comparison with existing teaching systems)

プロジェクトデザイン工学専攻（電気情報系）
小畠　一泰（指導教員　井上　浩孝）
Kazuhiro Kobatake

This research focused on issues that the author feels need to be addressed in programming education at the National Institute of Technology, Kure College. We were able to have 40 students program in a short time in an environment similar to that of an actual lesson and to conduct a statistical survey and questionnaire survey. The results showed that the new system was beneficial to the learners, especially in specific areas of programming education. It was also demonstrated that the new system can withstand practical situations and may contribute to improving the current educational environment.

Keywords: *Programming Education, Improve Education, WebAssembly, JavaScript*
プログラミング教育，教育効率の改善，WebAssembly，JavaScript

１．はじめに

本研究では呉工業高等専門学校において，筆者が感じたプログラミング教育に関する課題とそれを解決するために構築した学習システム，そのシステムを利用したことによって変化した学習効果について調査した結果を述べるものである．

２．課題と要因について

筆者が感じたプログラミング教育に関する課題とは，卒業時の学生間において習熟度に大きな開きがあることである．

この要因について過去の研究 [1]において検討した結果，「授業方法や実習方法に問題がある場合」や「友人のコードを写して課題をこなしているが，実際には理解していない場合」，「授業時間外で学習する機会が少ない場合」などの複合的な要因が原因であると考えられた．それらを解決する方法はないかと考え，先行研究 [1]ならびに本研究へと至った．

３．構築したシステムについて

先行研究 [1]において構築されたシステム（以降，旧システムと表記する．）では，さくらインターネット株式会社より提供されているさくらの VPS を 1 台契約し，そのサーバー上に手動で各種ソフトウェアのインストールやデプロイを行なっていた．旧システムは Web ブラウザー上のエディタで，プログラミングのコンパイル，実行などができた．また旧システムの特徴として事前に用意したテストと呼ばれる，プログラムが正しく動作するか自動的に検査する機能がある．これを利用することで学生は自身のプログラムの動作を手軽に確認することができ，トライアンドエラーの高速化・効率化が図られたことが，先行研究 [1]の評価より明らかである．

しかし，旧システムは後に行う評価において同時接続・同時利用に耐えかねると考え，より効率的かつ大規模なアクセスにも耐えうるシステムへと再構築することとした．

再構築したシステム（以降，新システムと表記する．）の構成を図 1 に示す．

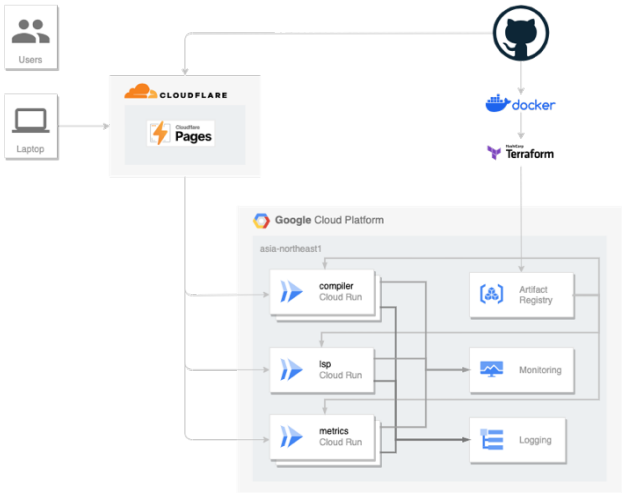

図 1　新システムの構成

図 1 の通り，フロントエンドは旧システムより変わらず GitHub 上で管理され，GitHub Actions 等を通じて継続的デリバリーされ，Cloudflare Pages 上にデプロイした．

バックエンドでは，Google Cloud Platform を主軸

とし，サービスには Cloud Run を利用した．

Cloud Run には GitHub 上で管理されるソースコードをもとに，Docker を利用してイメージを作成，Artifact Registry へ保存後，Terraform を用いてデプロイした．

Cloud Run では最大同時接続数を 150 に設定し，Cloud Run のオートスケール機能を利用することで，同時接続数が 150 を超えた場合に自動的にインスタンスが増加するように設定した．また Cloud Run はインスタンスの CPU 使用率が 60%を維持しようとするため，CPU 使用率が 60%を超えるようになると自動的にスケールする設定となっている [3]．各サーバーにおける CPU，メモリ，オートスケールの設定は，実際に学生が利用する際にどのような負荷がかかるかを予測し，表 1 の通り設定した．

表 1　Cloud Run の性能

サーバー	CPU コア	メモリ	オートスケール
compiler	2	1GiB	0〜10
lsp	4	4GiB	0〜1
metrics	2	1GiB	0〜2

また，全てのサーバーメトリクスは自動的に，Google Cloud Platform でモニタリングされる．ログについては自動的に Google Cloud Logging に保存される．

4．比較調査ついて

本研究では既存の授業で利用されるシステム（以降，授業システムと表記する．）と新システムにおいて同様の演習課題を行い，各種項目について比較調査，検討を行なった．

4.1　被験者について

呉工業高等専門学校の電気情報工学科 2 年に在籍する学生 43 名（以降，被験者と表記する．）に研究協力を依頼した．

被験者へプログラミングの経験などをアンケートした結果を表 2〜表 4 に示す．

表 2　C 言語を使い始めてどのくらい経ちますか？

選択肢	人数	割合
1 年未満	2	5.9%
1 年以上 2 年未満	29	85.3%
2 年以上 3 年未満	3	8.8%
3 年以上 5 年未満	0	0.0%
5 年以上	0	0.0%

表 3　C 言語以外を含むプログラミングには授業時間以外で普段どのくらい触れていますか？

選択肢	人数	割合
毎日触れている	0	0.0%
週に 5〜6 日ほど触れている	1	2.9%
週に 3〜5 日ほど触れている	1	2.9%
週に 1〜2 日ほど触れている	16	47.1%
月に 1 日程度触れている	4	11.8%
ほとんど触れない	10	29.4%
触れない	2	5.9%

表 4　プログラミングは得意ですか？

選択肢	人数	割合
得意	1	2.9%
やや得意	5	14.7%
普通	12	35.3%
やや不得意	9	26.5%
不得意	7	20.6%

表 2，表 3 より多くの学生にとってプログラミングは授業において学ぶことが多く，授業時間以外で触れる機会は少ないことがわかる．またそれに伴ってか，表 4 の通りプログラミングに苦手意識を持っている学生が多いことがわかる．

4.2　調査方法について

本研究では研究実施者による先入観が入らぬよう，また被験者に配慮し，次の通り行なった．

1. 各被験者に，個人を特定できない形で無作為に作成した符号を割り当て，その符号を用いて被験者のデータを管理した．
2. 全体を通じてウィンドウ名，キー入力数が保存されるようにした．
3. 授業システムではコンパイル・実行の際に，ソースコード，コンパイル時間，コンパイル結果，実行時間，実行結果が保存されるようにした．
4. 新システムでは被験者は，自身の符号を用いてシステムにログインした．その際に個人情報を入力することはなかった．そして授業システム同様に，ソースコード，コンパイル時間，コンパイル結果，コンパイル・実行・テストの利用率などが保存されるようにした．
5. 集計を行うために，研究実施者が介入せず，個人を特定できない形で回収した．

授業システムと新システムの利用には，普段の授業環境とできる限り近づけるため原則として被験者自身のパソコンを使用してもらった．

保存された各種記録より「タイプ数の変化」，「コ

ンパイル頻度の変化」，「コンパイル結果の変化」などを調査した．別途アンケートを実施し，主観的な感想や使い勝手も合わせて調査した．

実施時間は授業の都合ならびに，測定準備により表 5 の通りとなった．

表５　実施時間

項目	開始〜終了	時間
研究説明・測定準備	08:50〜09:45	55 分間
既存の環境での演習	09:45〜10:00	15 分間
システムの詳しい説明	10:00〜10:08	8 分間
システムでの演習	10:08〜10:18	10 分間

また演習とは付録 1 に示すソースコードを配布し，関数を作成することとした．

付録１　演習 1. c

```c
#include <stdio.h>

/* 注意
  - 関数名は変更しないでください。
  - main 関数の中身は自由に変更して構いません。
*/

int add(int a, int b) {
  // 引数を足し合わせた値を返す関数を作ってください。
}

float div(int a, int b) {
  // 引数 a を引数 b で割った値(型は float)を返す関数を作ってください。
}

int total(int max) {
  // 0 から max まで(max を含む)を足し合わせた値を返す関数を作ってください。
}

int main() {
}
```

4.3　既存の環境について

授業システムとして GitHub, Inc.によって提供されている Codespaces を利用した，CS50 と呼ばれる環境を利用している．これはウェブ上で bash や clang, gcc, docker を利用することができ，学習者は自身のパソコンに環境を構築する必要がない．

4.4　新システムについて

新システムでは付録 1 と同じ内容がウェブ上のエディタに存在し，テストとして表 6 に示すテストケースが用意されていた．この時小数点以下の精度とは，例えば 10/3 は 3.333…のように無限小数となり，これをプログラムで正しく表現することは難しく，実際の返り値と比較することもまた難しい．従って小数第 N 位までにおいてどの程度予期した返り値から外れるかをテストに含めることができるようにしてある．従って 3 とすれば例えば 3.3334 となってしまっても正しいとなる．

表６　テスト内容

関数呼び出し	返値	精度
add(1, 3)	4	0
add(-5, -3)	-8	0
div(10, 3)	3.333	3

4.5　調査結果について

各小見出しの通り結果の記載と見解を述べる．

4.5.1　キー入力数の変化について

表 7 にそれぞれのシステムにおける統計値を示す．図 2 に横軸に新システムの入力数を，縦軸に授業システムの入力数をプロットした．図 3 に被験者ごとのキー入力数の差をヒストグラムで示す．

なお，本調査では授業システムでの演習の後に，新システムでの演習を行なったため少なからず影響があることに留意したい．

それらを念頭に置いた上で各図表を見ると，100 字から 300 字程度の減少が確認できる．これは留意事項のみにならず，新システムのエディタで利用できる補完機能によるところが大きいと考えられる．

表７　キー入力数の変化

	授業システム	新システム
サンプル数	26	26
最小値	0	0
最大値	592	321
平均値	189.1	120.0
中央値	182	98
最頻値	0, 228	93
上位 3 つ	592, 478, 288	321, 292, 235
下位 3 つ	0, 0, 1	0, 2, 8

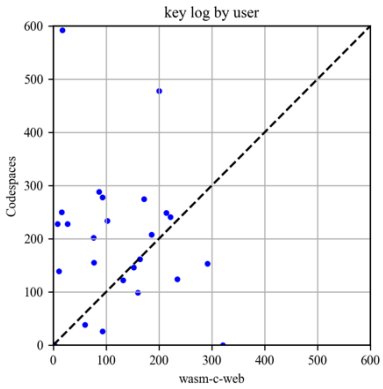

図２　被験者／システムごとのキー入力数の分布

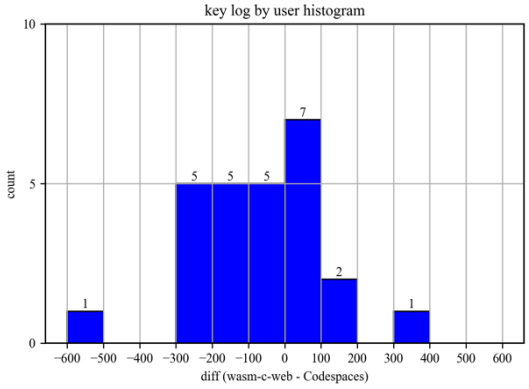

図３　被験者／ごとのキー入力数の差

4.5.2　コンパイル数の変化について

表８にそれぞれのシステムにおける統計値を示す．図４に時間軸に対するそれぞれのシステムでのコンパイル数を示す．図５，図６に被験者ごとのコンパイル順での結果を利用し，コンパイルに成功した場合はカウントを増加させ，失敗した場合は減少させた場合を示す．また同経路の被験者がいた場合は少しずつ太くした．

なお，本評価においても授業システムでの演習の後に，新システムでの演習を行なったため少なからず影響があることに留意したい．また，授業システムのサンプル数が著しく少ない点について，配布ファイルのコンパイルが 37 人であり，配布ファイルのコンパイルを除外した結果 10 人となった．従ってコンパイルできなかったためにコンパイル数が少ないという可能性は低いと考えられる．次に考えられる原因として被験者は記録用のスクリプトを用いなかった，または正しく認識できておらず使用していなかったことにより記録できていないだけである可能性だが，提出されたプログラムを見たところ多くの被験者がコンパイル不可な状態であった．これらから多くの学生は日常的にコンパイルすることなく課題を提出しているのではないかという可能性が生じた．

以上を踏まえ表８を見ると，総コンパイル数の減少が見られた．これはエディタ上にコンパイルすることなく，エラーがある場合はその箇所に赤い波線が自動的に表示されるからだと予想される．

次に図４を見ると，演習を開始してからコンパイル数が急激に増加していることがわかる．これは新システムを利用することで手軽にコンパイル・実行・テストができるため，トライアンドエラーの効率化・高速化に貢献していると考えられる．

また図５と図６を見ると，新システムの方が回数を重ねるごとにコンパイルに成功している様子が取れ，コンパイルエラーの発生が抑制されていると考えられる．

表８　コンパイル数の変化

	授業システム	新システム
サンプル数	10	39
最小値	1	1
最大値	14	9
平均値	5.0	3.5
中央値	3	3
最頻値	2	3
上位 3 つ	14, 11, 7	9, 8, 8
下位 3 つ	1, 2, 2	1, 1, 1

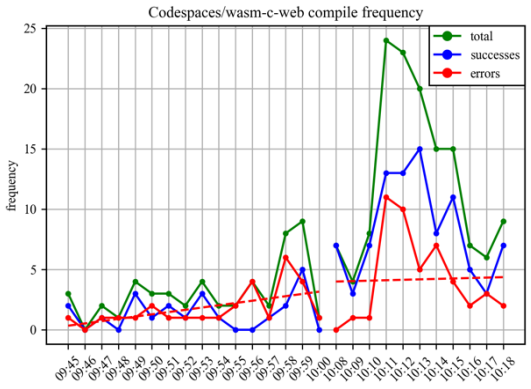

図４　時間軸に対するコンパイル数と結果

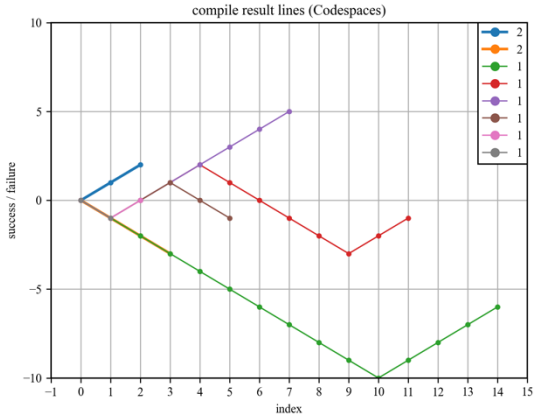

図５　授業システムでのコンパイル結果路

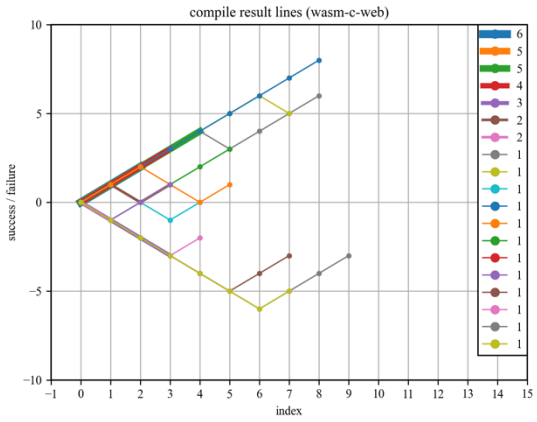

図６　新システムでのコンパイル結果路

５．アンケートついて

前項における統計調査以外に，演習終了後，被験者に対して通常の授業の様子と新システムを利用した上でのアンケートを実施した．

質問項目は「通常の授業の良いところ/悪いところについて」や「新システムの動作速度について」,「新システムの良いところ/悪いところについて」,「授業で現在の学習方法から新システムに変更された場合の影響について」などである．

なおサンプルサイズは 34 である．

5.1　通常の授業の良いところ/悪いところについて

良いところとして「全て手打ちだったので構文などを覚えられた」は 21 票，「様々なソフトに触れられたコンパイルコマンドなどに詳しくなれた」は 14 票などが挙げられた．また悪いところとして「全て手打ちなので時間がかかってしまう」は 14 票，「動作が正しいか確認するのが大変」は 13 票などが挙げられた．

5.2　新システムの動作速度について

表 9 に速度に関する回答を示す．

本調査では被験者の所有するパソコンを用いて新システムを利用したため，多様な環境であったが，表 9 より多くの場合において使用に耐えられる速度であったことがわかった．

表９　新システムの動作速度について

選択肢	人数	割合
速い	13	38.2%
やや速い	14	41.2%
普通	5	14.7%
やや遅い	2	5.9%
遅い	0	0.0%

5.3　新システムの良いところ/悪いところについて

良いところとして「テスト」は 25 票，「コンパイル」は 17 票，「実行」は 16 票，「整形（ボタンをクリックするとソースコードが自動的に整えられる機能）」は 10 票，「自動保存（500ms 毎に自動的にブラウザ上に保存される機能）」は 6 票などが挙げられた．また悪いところとして「コンパイル，テスト，整形」などが 2 票，「実行，補完」などが 1 票となったが，自由記入欄に記入がなかったためどういった点で悪かったのか不明である．

5.4　授業で現在の学習方法から新システムに変更された場合の影響について

動作面と操作面から効率について問うたところ，「操作が手軽なので学習効率が上がる」は 19 票，「動作が軽快なので学習効率が上がる」は 13 票，「操作が難しいので学習効率が落ちる」は 4 票，「動作が遅いので学習効率が落ちる」は 2 票となった．

６．新システムの性能評価について

6.1　コンパイル時間について

授業システムではコンパイル・実行を Codespaces 上の環境で行う．従ってコンパイル時にはコンパイル時間のみが必要であり，実行後はテキストのみの通信となるため非常に効率的である．一方，新システムはコンパイル時にプログラムを送り，WebAssembly にコンパイルされたバイナリが返却されるためレスポンスサイズが大きくなりやすい．

表 10 に付録 1，2 を授業システム，新システムでそれぞれコンパイルしてできたファイルサイズを記載する．

付録２　日本語.c

```c
#include <stdio.h>

int main() {
  printf("あ¥n");
}
```

表 10　コンパイル後のファイルサイズ

対象	授業システム [KB]	新システム [KB]
付録 1	16.7	1.7
付録 2	33.1	61.2

このように常にどちらかのシステムにおいてファイルサイズが増加するというわけではないことがわかる．以上より，システムの性能評価には単

にコンパイル時間を比較するのではなく，新システムはコンパイル後の転送時間も含める必要があると考える．これらを踏まえてそれぞれのコンパイル時間を箱ひげ図[2]にしたものを図 7 に示す．

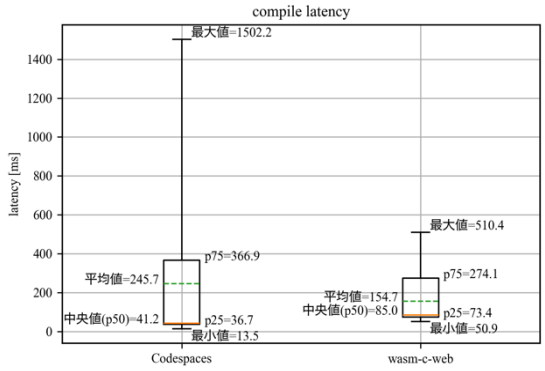

図 7　各システムのコンパイル結果時間

図 7 より授業システムの最小値，最大値が新システムに比べて外れているが，これらは被験者の使用したパソコンのスペックによるものであると考えられる．授業システムに比べて第一四分位数，中央値では劣ってしまうが，新システムは第三四分位数を見てもわかるように，分散が小さく安定していることがわかる．

6.2　新システムのサーバー性能について

図 8，図 9 にコンパイルサーバーと LSP サーバーについて時間軸に対する各種負荷状況を示す．

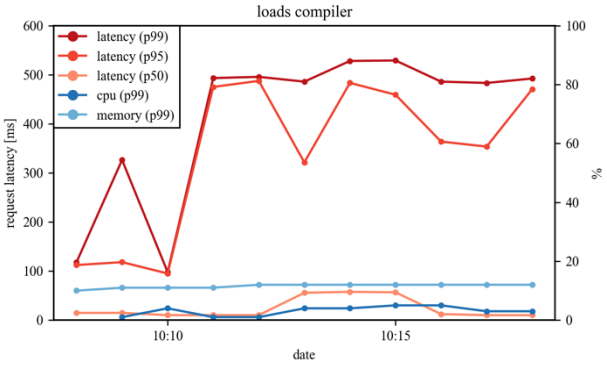

図 8　コンパイルサーバーの負荷状況

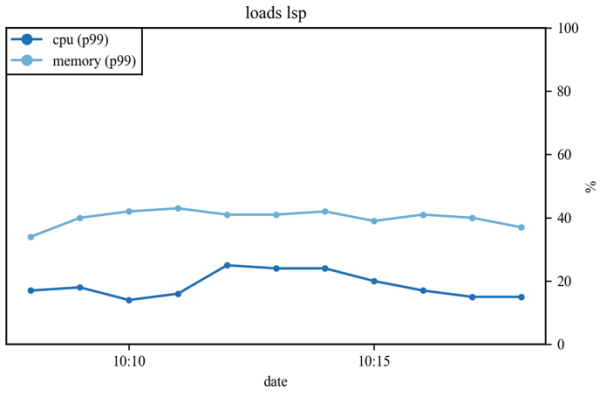

図 9　LSP サーバーの負荷状況

図 8 では CPU 使用率(p99)，メモリ使用率(p99)が全体を通じてそれぞれ 3%～5%，9%～12%と低い使用率であった．またレイテンシー（ここでは実際にサーバーがリクエストを受信してから，レスポンスを返し終わるまでの時間を意味する[3].）は 50 パーセントタイルで 10ms～60ms を推移している．また，99 パーセントタイルでは 480ms～530ms 程度を推移しており，コンパイル内容によっては一部時間がかかってしまうが，全体を通じて使用者が遅いと感じる速度ではないと考える．またインスタンス数は常に 1 であった．

図 9 では CPU 使用率(p99)は 14～25%を推移し，メモリ使用率(p99)は 34%～42%を推移した．またレイテンシーの記載がない理由は，LSP サーバーではエディタと WebSocket を利用して常に接続をしているため，使用者がブラウザを閉じるなどしない限り接続後時間が経つに連れて単調増加し意味をなさないからである．

7．まとめ

本研究では短い時間ながらも 40 名近い学生に対し，実際の授業と同様の環境でプログラミングを行ってもらい，統計調査並びにアンケートを実施することができた．アンケートの結果，多くの学生が現在の授業方法に問題を感じていること，構築した新システムで大きく改善されるであろうことがわかった．また統計調査によりどの分野がどの程度学習効果があるかが明らかとなった．そして授業利用などにも耐えうるシステムであることがわかった．

参考文献

1) 小畠一泰，笠井聖二：WebAssembly を用いたブラウザ上で完結する C 言語の TDD 式学習システムの構築と提案，令和 3 年度　呉工業高等専門学校　電気情報工学科　卒業研究報告会予稿集，2022
2) 高遠節夫，ほか 5 名：新　確率統計，大日本図書株式会社，2018.12
3) Cloud Run(https://cloud.google.com/run?hl=ja), 2024.1 閲覧