第一工科大学研究報告
第37号（2025）pp.37-44

生成 AI を活用したソフトウェア開発の教育実践

渋沢良太
第一工科大学　工学部　情報・AI・データサイエンス学科（〒899-4332 鹿児島県霧島市国分中央 1-10-2）

Educational Practice on Software Development Using Generative AI

Ryota Shibusawa
1-10-2, Kokubu-chuo, Kirishima, Kagoshima 899-4332 Japan

Abstract: In recent years, with the advancement of large language models, systems that can automatically generate code based on natural language descriptions of the desired program—known as code generation AI—have become practical tools. These AI systems, powered by large language models, have proven highly effective and are now being adopted even by professional programmers in real-world settings. Although current generative AI technologies cannot yet handle all aspects of software development required in the industry, they are significantly reshaping the field of software engineering. Against this backdrop, this study implemented educational practices in software development using generative AI within software engineering courses offered by the Department of Information, AI and Data Science, as well as the Department of Mechanical Systems Engineering at our university. The research examined the educational outcomes of these practices and discussed future curriculum design guidelines.

Key words: 生成 AI，プログラミング教育，ソフトウェア工学，大規模言語モデル，コード生成 AI

1. はじめに

近年，大規模言語モデルの発展に伴い，作成したいプログラムの内容を自然言語で指定すると，それに合ったプログラムを自動生成するシステム（コード生成 AI）が実用化されている．大規模言語モデルを活用したコード生成 AI の実用性は高く，実社会におけるプロフェッショナルのプログラマーにも活用され始めている．現状では，現実社会で要求される全てのシステム開発を，生成 AI のみを使って開発できるわけではないが，ソフトウェア工学のあり方を大きく変えつつある．

このような背景のもと，本研究では，本学の情報・AI・データサイエンス学科，および機械システム工学科で開講している 2024 年度後期のソフトウェア工学の授業において，学生 72 名に対して生成 AI を活用したソフトウェア開発の教育を実践し，今後のカリキュラムの設計指針について考察を行った．

2. 授業の構成

本研究で実施した授業の各回の内容を表 1 に示す．授業は主に，1)ソフトウェア開発の全体像の理解，2)ソフトウェアの設計方法の理解，3)プログラミングパラダイムの理解，4)生成 AI の活用方法の理解，5)生成 AI を活用したソフトウェア開発の実践の 5 つから構成した．

本授業は，情報・AI・データサイエンス学科の 3 年生向け，機械システム工学科の 4 年生向けに開講しており，受講者はプログラミングそのものについて，ある程度理解していることを前提としている．

表 1 ソフトウェア工学の授業内容

回	内容
1	ソフトウェア工学の概要，ソフトウェア品質の国際規格について
2	ソフトウェア開発の工程と IT アーキテクトの業務内容
3	ビジネス・ソリューションアーキテクチャとドメイン駆動設計
4	ソフトウェアアーキテクチャ
5	設計図（UML，DFD，ER 図）
6	モジュール設計，テスト方法
7	プログラミングパラダイム 1（C 言語と Java の比較，オブジェクト指向）
8	プログラミングパラダイム 2（関数型プログラミング：Elixir）
9	プログラミングパラダイム 3（型安全性，メモリ安全性，スレッド安全性：Rust）
10	生成 AI とそのプロンプトエンジニアリング(ChatGPT, claude, gemini), create.xyz，github copilot によるコード生成
11	Dify によるコード生成
12-13	課題作成
14-15	課題発表

学生は，他のプログラミングの授業において，仕様が決められたプログラムの機能をどのようにコーディングされるのか理解できている．また，システム開発のメインがコーディングであると認識している者が多い．本授業では，プログラミングだけではなく，実社会においてどのようにシステムが受発注され，開発が進められていくかについての説明を行い，システム開発工程の一部としてプログラミングがあることをまず理解してもらうようにしている．

ソフトウェアの設計においては，まだ言語化されていない顧客の要求を汲み取って仕様を作成することがまず重要になる．この点については，エリック・エヴァンスによって提唱されたドメイン駆動設計[7]を中心に，ビジネスドメインの理解とシステム化対象の決定，システムの設計について説明を行った．その際，仕様の図式化の方法として，UML, DFD, ER 図について説明を行った．また，モジュール設計の方法とテスト方法も合わせて説明を行った．テスト方法も，学生はソフトウェア工学以外の授業では経験する機会が少ないが，他人に利用されるシステムを高品質に作るためには欠かせない重要事項である．特に単体テストと結合テストの方法，スタブとドライバの具体例等を示した．

情報・AI・データサイエンス学科の学生は，C 言語，Python，Java，PHP を，機械システム工学科の学生は Python について学習している者が多い．プログラミング言語は複数あるが，システム開発においてそれらがどのように使い分けられているかを，プログラミングパラダイムの区分において説明した．C 言語の構造体と Java のクラスの比較，オブジェクト指向について説明し，ソフトウェアの拡張性，保守性をいかにして高めるかについて説明を行った．また，オブジェクトの内部状態の違いによるテストのしにくさの問題点と，その問題を回避する関数型プログラミングについて説明を行った．また，オブジェクト指向と関数型の良いところを併せ持った最近のプログラミング言語として，所有権・借用の仕組みを持つ Rust の説明を行った．

これらの説明を行った後，第 10 回と 11 回の授業において生成 AI を活用したソフトウェア開発の説明を行った．大規模言語モデルの概要と，それらを使って作られた ChatGPT[1], Claude[2]や Gemini[3]といったチャットボットを説明した．その際，one shot learning, few shot learnig, chain of thought といったプロンプトエンジニアリングを説明した．そのあと，コード生成 AI として，Create.xyz[4]，GitHub Copilot[5]，Dify[6]を紹介し，それらを使ったコード生成方法を示した．

最後に各学生に，自らが作成したいアプリを考案させ，生成 AI を使ってそれを実際に開発する課題に取り組ませ，全受講生の前で発表する機会を設けた．

3. 生成 AI を使った課題制作の実践

学生が課題で制作した作品の代表的なものをに表 2 に示す．表 2 中の No.6, 9, 14 のように，ゲームを開発する者も多かったが，それ以外にも普段の生活を便利にするような実用的なプリも多く見られた．

学生たちが課題制作で使用した生成 AI ツールとその人数の内訳を，表 3 に示す．受講者のうち 30 人は，ノーコードでアプリを開発できる Create.xyz と Dify を使用しており，35 人は ChatGPT などを使って Python, JavaScript, HTML を使ったアプリを開発していた．

表 2 学生が生成 AI を活用して制作したアプリの代表例

No.	制作されたアプリの種類と内容
1	専門知識を教えてくれるアプリ．市販薬などの薬を服用する際に，違う薬や食品など一緒に飲んではいけない組み合わせや，具体的な食品名，薬物名，その副作用や症状を詳しく教えてくれる．
2	タスク管理アプリ．タスクの内容，優先度，期限と画像アップロード，メモ入力ができる．
3	学習用テストアプリ．レベルを選択して，レベル別に英単語のテストができる．スピード計算ができる．漢字のテストができる．
4	全国の設備からくり時計の位置と稼働状況を入力，共有できる．
5	霧島市の観光地を，遊びたい内容を入力して検索できる．
6	行動内容の文字入力のみで遊べる RPG ゲーム
7	家計簿アプリ．期間別，カテゴリ別に集計表示ができる．目標金額との差を提示できる．
8	調べたいことを入力すると，wikipedia のサイトから情報を抽出し，小学生が理解できる内容の文に変換して調べられる．
9	3L と 5L の容器を使って，4L の水を作るゲーム．
10	小説を生成するアプリ．ユーザーが希望するジャンル，テーマ，主人公の名前，あらすじなどを入力するだけで，小説を自動生成する．また，出力された分の内容を修正，編集が可能．
11	食材とその量を入力すると，予想カロリーと含まれる栄養素を提示する．
12	体重を毎日記録し，時系列に折れ線グラフ表示する．目標体重との比較をわかりやすく表示する．
13	OpenWeaterMap を使い，地域ごとの天気予報を調べられる．天気，気温，湿度，風速，降水確率等，リアルタイムの天気情報も閲覧できる．
14	ゲーム類．いご，じゃんけん，マルバツゲーム，おみくじ，ハイアンドロー，テトリス，10 秒ちょうどを測るゲーム，あっち向いてホイ，ブラックジャック，ノベルホラーゲーム

表 3 課題制作で使用したツールと人数の内訳

使用した生成 AI	人数
create.xyz	20
ChatGPT	20
Dify	10
gemini	6
gemini advanced	3
CodePen	3
Microsoft Copilot	3

講義の最後に学生にアンケート調査を行った．回答者は 51 名であった．「生成 AI を使って，ソフトウェアをどの程度作成したいと考えたとおりに開発できましたか？」という問いに対する結果を，図 1 に示す．8 割以上の学生は，70%以上は思い通りに開発できたと回答しており，ツールを上手く使いこなせていた．今回の課題では，学生に作りたいアプリを考案させており，アプリの内容が簡素であったため，目的の機能が上手くできたと回答する者が多かったと考えられる．実社会におけるシステム開発では，業務ロジックは複雑な場合が多く，これほど簡単にはシステムの全てを実装できない点には注意が必要である．

また，「生成 AI を使ってソフトウェア開発する時に，思い通りにできたことを教えてください」という質問に対する回答の代表例を，表 4 に示す．アプリの内容はゲームなど，機能が比較的単純なものが多かった． 目的とする機能の他に，デザイン等，UI も思い通りに作れたと回答した学生もいた．

表 4 生成 AI で思い通りにできたこと

No.	回答内容
1	ゲームの動作，ゲーム性の表現がしっかりと出来ていた．
2	視覚的に分かりやすくすること．
3	得点をつけること，色の変更．
4	データを送るだけで、地方・都道府県・市町村に割り振れたこと
5	分かりやすく細かく指示して綺麗に動くようになった．
6	ほぼ一発で，理想的なコマンドにたどり着けた．Gemini Advanced を持っていたことにより，開発時間が予想の半分で済んだ．
7	ある程度の思っていた機能は入れることができたり，修正なども自分の思った通りに反映させることが出来た．
8	思う通りの Web デザインにできた．
9	ゲームの基本機能を作成後，いくつか追加したい機能を追加できた．
10	大雑把な命令にはしっかり聞いてくれることが多かった．

「生成 AI を使ってソフトウェア開発する時に，思い通りにできたことを教えてください」という質問に対する回答の代表例を，表 5 に示す．

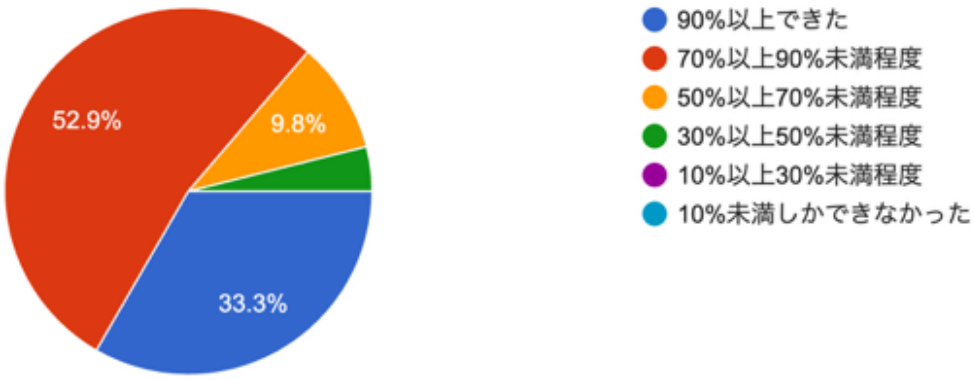

図 1　質問：「生成 AI を使って，ソフトウェアをどの程度作成したいと考えたとおりに開発できましたか？」の回答

表 5 思い通りにできなかったこと

No.	回答の内容
1	1つの機能を追加すると他の機能が消えてしまうことが起きてしまって，いくつかの機能は追加できなかった.
2	段階的にスピードを上げようとしたが，できなかった.
3	トークンの制限により，長文（長コード）の出力が不安定になった.
4	グラフを挿入したかったが思い通りのグラフが表示されなかった.
5	相手の手札と得点を見えないようにしたかったが得点だけ見えなくなって、手札を見えなくする事が出来なかったところ.
6	難易度が思ったものにならなかった. ヒントが直接的になりすぎた.
7	音楽などをつけたかったが，同じファイルに入れても反映されなかった.
8	細かなバグの修正.
9	UIデザイン，見た目.
10	データ容量の関係でたくさんのデータを学習させることができなかった.

学生が考案したアプリの内容の大まかな部分はコード生成 AI で実現できるが，細部の機能修正はうまくできなかったと回答している学生が多い．このことから，仕様を明確に言語化して入力できる汎用的な処理はコード生成 AI によって開発が容易になっているが，うまく言語化できない処理については開発者にコーディングする能力が必要であることがわかる．したがって，システム開発者には，仕様を言語化する能力，コード生成 AI が出力するコードを理解し，改良できる能力が依然として重要である．また，増加し続けてはいるものの，コード生成 AI に入力できるトークン数には上限がある．そのため，大規模な仕様を上手く分割してコード生成 AI に入力し，出力結果を上手く組み合わせる能力も開発者には必要である.

表 6 意外であった気づき

No.	回答の内容
1	なかなか思い通りにいかなかった.
2	発言内容をしっかりと捉えて．希望に沿った適切なコーディングを行ってくれるAIの力に驚かされた．必要な表示を作ってくれる点も良かった.
3	トークン量が多ければ、優秀なプログラマーを月3000円（課金前提）で雇えるほどの大ブレイクだと感じた.
4	ある程度の枠組みだけならそこまでエラーを出すことなく思い通りにつくれるということに感動した.
5	簡単な所は思ったよりも上手くいく．細かい所は中々上手くいかない.
6	プログラムに関する専門用語を使わなくてよかった.
7	プログラムを書かずに作れることがここまでＡＩがかなり進歩したと実感した.
8	基礎から作るよう指示しても．前に作ったプログラミングが残っていて、バグが残ったままだった。
9	検索や情報集めには手間がかかることを知った.
10	使用するAIだけでなく、参考とするPDFをインストールすることで精度を上げたり，自身のニーズにあった改造ができたりする.

「生成AIを使ってソフトウェア開発する時に，意外だったことがあれば教えてください.」という質問に対する回答の代表例を，表 6 に示す．想像していたより簡単にアプリ開発ができたことに驚いていた学生が多かった．一方で，細部まで完璧にプログラムがコード生成 AI によって作られることを期待していた学生は，上手くできない部分があることを回答していた．また，プログラマーの時給と有料版のコード生成 AI の料金を比較し，費用対効果が優れていることに気づいた学生もいた.

「生成 AI を活用したソフトウェア開発の方法を理解できましたか？」という質問に対する回答結果を，図 2 に示す．回答した 94%以上の学生は，本授業を通して生成 AI を使ったソフトウェア開発の方法について理解ができていた．本授業では，生成 AI の使い方，アプリ開発の実践例を教員が全員に対して示したのは授業 2 回分であったため，この時間を増やすことで学生の理解もより深められると考えられる．

4. ソフトウェア工学の授業構成の指針

実社会のシステム開発の全てにおいて，コード生成 AI によってプログラミングが全て不要になるようなことは，現段階では期待できない．しかし完全ではなくとも，生成 AI の活用によりシステム開発の効率は格段に向上する．従って，これからのソフトウェア工学の授業では，システム開発における生成 AI の活用方法を教えることが重要となる．本授業では，プログラミングする部分のみに生成 AI を活用したが，システム開発の各工程において，表 7 のような活用方法が考えられる．

システム開発の各工程ではレビューが重要となるが，レビューに生成 AI を活用することで，標準的な観点から各工程で不足していることを発見することに役立たせられる．

実社会のシステム開発では，費用対効果が重要な評価指標となるため，システム化の対象を決めることが重要となる．その際，まずビジネス上の課題を抽出する必要があるが，これは言語化されていない場合が多い．従って，ビジネス課題を生成 AI で抽出することは難しいが，ビジネスや業

表 7 システム開発の各工程における生成 AI の活用方法の例

工程	活用方法
全工程	レビューする，助言を得る，一般的な解決策を知る．
ビジネス課題の抽出	現状のビジネスの何を改善したら，コストを下げられ，収益を向上させるかを探索する．
ドメインの理解	システム化対象の業務内容について調べる，専門用語の知識を得る．
システム化のドメインの決定	ビジネス課題を踏まえ，システム化の対象業務，範囲を決める．
仕様の作成	UML，DFD，ER 図等を作成する．
コーディング	必要な機能のコードを生成する．デバッグする．
テスト	テストケースを生成する．テストコードを生成する．
運用	ユーザからの質問に回答する．
保守	障害の原因を調査する，

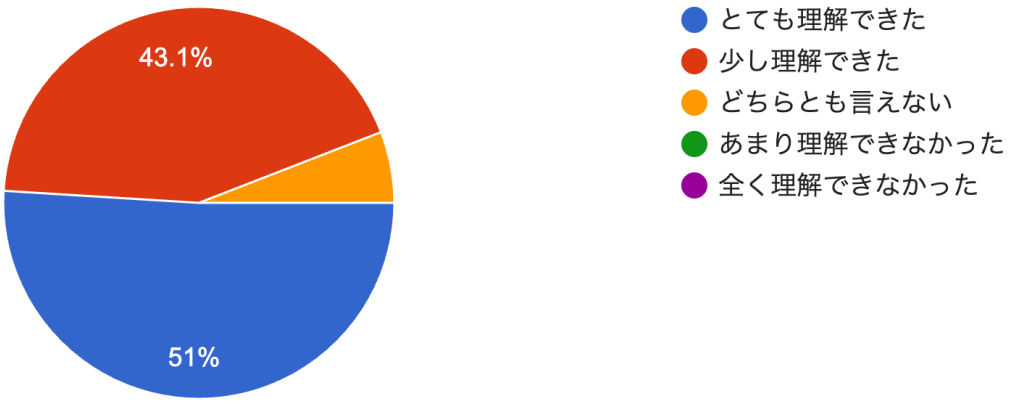

図 2「生成 AI を活用したソフトウェア開発の方法を理解できましたか？」の回答

務内容と，それぞれの業務のパフォーマンスを示す数値データをあらかじめ測定しておくと，生成 AI によるビジネス課題の抽出に役立てられると考えられる．この観点からも，様々な業務のパフォーマンスをデータ化して収集しておくことは重要である．

ドメイン駆動開発でも述べられている通り，ユーザの望むシステムを実装するためには，システム化対象の業務内容，専門用語を開発者が良く理解する必要がある．この工程において，対象業務の一般的な知識については，生成 AI を活用して調べることが可能である．また対象とする企業内での特殊な知識については，企業内のドキュメントを知識源として RAG(Retrieval-Augmented Generation)をするように生成AIツールを設定することで，効率的に知識を調べられる．

仕様の策定も，言語化されていないことを言語化する工程であり，生成 AI の活用の有無を問わず重要である．自然言語で的確に仕様を記述できれば，PlantUML[8]や Miro AI[9]といったツールで一定の UML を自動生成することも可能である．また，コーディングの工程においては，本授業で学生が体験した通り，大まかなコードをツールで生成させられる他，コード補完機能によって関数名やその典型的な使い方を調べなくても実装が可能である．さらに，モジュールの仕様が的確に作成できていれば，生成 AI の活用によってテストケースを自動生成させやすくなるため，特に単体テストの効率を良くできると考えられる．

LLM と RAG を使ったチャットボットが多く実用化されているが，開発したシステムの運用サポート業務においてもこれらは有効である．障害の原因の特定は難しい場合が多いが，システムを構成している機器，稼働内容のデータを収集しておくことで，生成 AI を活用できる機会が増えると考えられる．

生成 AI を活用する場合であっても，従来のソフトウェア工学で教えられている知識の修得は必要であり，プログラミングのスキルの修得も必須である．もし開発者がそれらを修得できていなければ，まず生成 AI によって提示されたことが正しいか否かを理解できない．従って，実社会のシステム開発における生成 AI の活用は，システム開発する能力を修得できている人の開発効率を向上させることに寄与する．しかし，システム開発の能力を修得できていない人にとっては，部分的には開発を容易にできるが，システム全体の開発をかえって難しくしてしまう．そのため，ソフトウェア工学の授業においては，従来のソフトウェア工学の知識を修得させることがまず必要なる．その上で，表 7 に示したように，各工程での生成 AI の活用方法を理解することが重要であると考えられる．

生成 AI の普及が進むにつれて，特に「言語化」の重要性がさらに高まっている．実社会のシステム開発では，ビジネス課題やシステム化の対象業務の内容，システムの仕様など，開発の最初にまだ言語化されてないことも多く，そのままでは生成 AI を活用できない．したがって，言語化されていない事柄をいかにして言語化できるかが，今後もシステム開発で重要になる．ここで，「言語化」とは，対象を自然言語で表現することのみを表しているのではなく，図や記号，数式で表すことも含んでいる．そのため，「言語化」の能力の養成には，UML 等によるモデリングの訓練の他，物理モデリングの訓練や，抽象数学における概念，概念と概念の関係性の理解の訓練，証明の訓練等が有効であると考えられる．

5. おわりに

本研究では，本学のソフトウェア工学の授業において，生成 AI を活用したソフトウェア開発の教育を実践し，その教育効果の検証と今後のカリキュラムの設計指針について考察を行った．生成 AI を活用したソフトウェア開発を実践してみると，従来のソフトウェア開発のスキルや，ソフトウェアの原理，理論的基盤を開発者が理解していることの重要性がより明らかになる．情報技術者の育成においては，これらをきちんと修得させた

上で，生成 AI の活用方法を修得させることが望ましい.

今後，さらなる生成 AI の進展を踏まえ，ソフトウェア工学の授業内容を継続的に改善して検証し，時代に合ったシステム開発のスキルを学生が修得できるようにしたい.

参考文献

[1] OpenAI, ChatGPT, https://chat.openai.com, (2025 年 5 月 10 日参照).
[2] Anthropic, Claude, https://claude.ai, (2025 年 5 月 10 日参照).
[3] Google, Gemini, https://gemini.google.com/app, (2025 年 5 月 10 日参照).
[4] Create.xyz, https://www.create.xyz/, (2025 年 5 月 10 日参照).
[5] GitHub Copilot, https://github.com/features/copilot, (2025 年 5 月 10 日参照).
[6] Dify, https://dify.ai/, (2025 年 5 月 10 日参照).
[7] Eric Evans, Domain-driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2004.
[8] PlantUML, https://plantuml.com/, (2025 年 5 月 10 日参照).
[9] MiroAI, https://miro.com/ai/, (2025 年 5 月 10 日参照).