CJ-007

コーディング面接の練習を支援するシステムの構築
An Interactive System for Supporting Programming Interview Practices

杉山 悠司[1)]　矢谷 浩司[1)]
Yuji Sugiyama　Koji Yatani

1　はじめに

ソフトウェアエンジニアなどプログラミングを行う技術者を企業側が評価する方法の一つとしてコーディング面接が存在する．コーディング面接とは，面接官と参加者が対面ないしオンラインで面接を行う中である問題が提示され，その問題に従ってコーティングを行うという面接形式の一つである．参加者は指定された課題に対してその場で一からコードを書き始め，コーディング中に適宜面接官からのコメントや質問に答えながら解答を完成させていく．コーディング面接の形式は，会場にあるホワイトボードに手で疑似コードを書き進めていくものや用意されたパソコンを用いて行うもの，共有のエディタやドキュメントに記入していくものなど様々存在する．

まず，コーディング面接においてどのような課題があるか明確にするため，参加者側としてコーディング面接を経験したことのある実験参加者を事前に集めインタビュー調査を行った．インタビューの結果，コーディング面接の形式に慣れるための練習が事前にできなかったという意見や，問題を解く練習は行っていたものの本番で面接官からの質問にうまく答えることができなかったという意見が特に多く見られた．

そこで本研究ではコーディング面接を受ける参加者側に注目し，参加者がコーディング面接特有の形式に慣れることができるようなシステムを構築することを目指す．コーディング面接では面接官とのコミュニケーションや受験者の意図しないタイミングでの質問，慣れていない環境への適応など，通常のプログラミング開発には見られない要素が多く存在する．こうしたことから，コーディング面接に限定したシステムの設計が必要であり，参加者はこのようなシステムを事前に使うことでコーディング面接特有の形式に慣れることができると考えた．

本稿ではまず第 2 章で現在までに行われている周辺分野の研究についてまとめた上で，本研究で行うべき貢献の範囲や内容について明確にする．次に，第 3 章ではシステムの作成に先行して行ったインタビュー調査の内容を紹介する．このインタビュー調査によってコーディング面接において解決すべき課題を明確にする．第 4 章では実際に設計したシステムの概要を紹介した上で，3 章で取り上げた要件がどのような形で含まれているかについて論じる．第 5，6 章ではこのシステムを用いて実施したユーザ実験について述べる．このシステムを用いることでコーディング面接で必要とされる要素がどの程度網羅されているかなどについて定性的に述べる．

1）　東京大学大学院 工学系研究科

2　関連研究

2.1　コーディングをオンラインで実施できる既存の Web サービス

コーディング面接は対面で実施されることが多いが，Web に問題を公開して個人で問題を解くことができるサービスや，オンラインでコーディング面接の環境を提供するようなサービスは多く存在する．例えば LeetCode[1)] というサービスは実際に各企業で出題された問題や類似した傾向の問題を数多く揃えることで事前の練習環境を実現している．また，Pramp[2)] というサービスは面接官の役割をこなす他のユーザがいれば，エディタと問題文を提示するリモートでのコーディング面接を即時でセッティングすることができる．

2.2　technical interview におけるユーザの心理に関する研究

研究論文においてコーディング面接はしばしば”technical interview”というキーワードで取り上げられ，プログラマが就職活動を行う際に受ける技術試験の一つと位置づけられている．Technical interview では，実際にコーディング面接を実施する際に受験者がどのような知覚情報を得ているかなどについて調査を行った事例が多い．以下ではその具体例を紹介する．

まず，Behroozi ら [2] はコーディング面接を実施する際に用いられることの多いホワイトボードに注目し，ホワイトボードで問題を解く際のユーザが感じる認知負荷を計測することを試みた．彼らはホワイトボード上と紙を用いたコーディング面接で認知負荷にどの程度差異が生じるかに注目し，視線追跡が可能なヘッドマウントディスプレイを用いて実験を行った．実験の結果をもとに，ホワイトボード環境下に特有の緊張による集中力の低下や認知負荷の増加を結論づけ，コーディング面接を実施する環境の改善案を今後の展望としてまとめた．

インタビューをする側とされる側の心理的な要求についてインタビュー調査を行った研究も存在する．Ford ら [5] はしばしばインタビューする側とされる側で重要視する事柄の不一致が技術面接において発生していることに注目し，この認識のずれを特定することを試みた．この目標を達成するため，彼らは大学内において擬似的な技術面接を実施して双方からのフィードバックを集計し，この結果を grounded theory [3] を用いて分析することで双方の認識の違いを明らかにした．具体的には，面接官は受験者が思っているよりも技術的な能力の高低ではなく，それを伝えるコミュニケーション能力に重きを置いていることなどが挙げられる．

ここに挙げた研究は技術面接に焦点を絞ったものであり本研究と分野がかなり類似しているものである．こうした既存研究を踏まえ，本研究ではコーディング面接における現在の問題点を整理した上でそれらを解決するような手法を提示することを目指す．

1）　https://leetcode.com
2）　https://pramp.com

2.3 プレゼンテーションを支援する研究

人前で成果物を発表する際，様々な場面でプレゼンテーションが実施される．こうしたプレゼンテーションを実施するには多くの準備や工夫が必要であり，これらの質を高めるためにこれまでに様々な研究が行われてきた．本節ではプレゼンテーションを支援する既存の研究事例についていくつか紹介し，それぞれの研究について本研究との類似点や相違点について述べる．

Hoque ら [6] は就活時のインタビューなどでの会話能力を向上させるため，MACH と呼ばれるバーチャルアシスタントを実装した．このシステムでは PC に取り付けられたセンサとカメラを用いて話者のジェスチャーや声の大きさ，イントネーションなどの様々な特徴量を記録することができる．記録された結果をもとにより良いパフォーマンスを実現するために必要なアドバイスを決定し，それをバーチャルアシスタントがフィードバックとして返すような構造になっている．また，バーチャルアシスタントによるフィードバック以外にも図の下部のような特徴ごとの視覚的なフィードバックも提供されており，自分のインタビューの評価を正確に行うことができる．彼らはこのシステムを用いて大学生約 90 人にインタビューの練習を行ってもらい，使用しなかった大学生と本番でのパフォーマンスを比較することでこのシステムの有用性を示した．

Trinh ら [13] はプレゼンテーションにおけるリハーサルに注目し，リハーサルの存在がしばしば軽視されることによって本番のプレゼンテーションの質が低下していることを指摘した．そこで彼らは Pitch Perfect と呼ばれるシステムを提案した．このシステムはプレゼンテーションを行う際にしばしば使われる Microsoft PowerPoint の機能拡張を実現するようなものであり，要素ごとに覚え書きや話のフローチャートの追加や発表内容の暗記を補助する機能，本番に近い環境でスピーチの練習ができる機能などを含んでいる．彼らはこのシステムを 12 人の実験参加者に使用してもらい，通常の PowerPoint を用いてリハーサルを行った場合との差異について検証を行った．その結果，時間の管理や発表中の自信の表れなどいくつかの部分において差異が見られたことからこのリハーサルシステムの有効性を示した．

プレゼンテーションにおける口頭での発表練習に特化したインターフェースも存在する．Trinh ら [12] は RoboCOP と呼ばれる人の頭の形を模したロボットを製作し，そのロボットに向けて話しかけることで実際のプレゼンテーションに近い環境でのリハーサルを可能とした．このロボットは会話の質や内容の網羅度，聴衆へのアイコンタクトの度合いなど様々な要素を定量的に評価することが可能となっており，自分の発表の質をリハーサルを通して知ることが可能となっている．実際に実験参加者にこのロボットを使用してもらうことで高い満足度が得られたことを示す一方で，body languange への対応など様々な要素を課題点として提示した．

個人の特徴に合わせて発表内容をカスタマイズすることを試みた研究も行われている．Rijnsburger ら [9] は既存のプレゼンテーションでは聴衆の個々の知識レベルに適応することが難しいことに注目し，個々の聴衆に合わせた発表内容の注釈をヘッドマウントディスプレイを通して提供するインターフェースの提案を行った．彼らはヘッドマウントディスプレイを通して得られる視線の動きを元にして最適な注釈の挿入方法の検討を行い，通常の発表と比べて理解度や快適さにどの程度の差が出るかの調査を行った．その結果，全ての説明を全員に共有する発表と比べて聴衆側の理解度の向上につながるという結論を得ることができた．

ここで述べた研究はプレゼンテーション一般について広く述べたものが多く，必ずしも本研究の主旨と直結しているとは限らない．しかし，コーディング面接が自分の書いたコードや考えを相手に提示するという特性を持っている以上，こうした研究も本研究と関連性を持っていると考えたため紹介を行った．

3 システム設計に先立ったインタビュー調査

3.1 インタビュー調査の実施

コーディング面接を支援するツールを開発するにあたり，過去にコーディング面接を実施したことのあるユーザがどのような感想を抱き，どのような問題点を認識していたかについて把握することが必要となってくる．既存研究のみを根拠として新しい手法を提案しても，それが実際の現場で必要とされている要素と乖離した必要性の薄いインターフェースデザインとなってしまう恐れがある．

そこで私たちはインターフェースデザインの提案に先立ち，コーディング面接を過去に受けたことのある人たちを対象にインタビュー調査を実施した．本研究ではコーディング面接を受ける側を支援するインターフェースを構築することを想定しているため，就活やインターンシップなどの機会にコーディング面接を経験した学生を中心とした調査を行った．インタビュー調査で具体的に尋ねた内容は以下の通りである．

- どのような目的でコーディング面接を受けることになったか
- 受けたコーディング面接の形式はどのようであったか (使った媒体など)
- 事前にどのような準備を行ったか
- 面接中に感じた懸念事項や反省点などはあったか

基本的にはあらかじめ設定した質問事項に沿ってインタビューを進めていき，話の内容に応じて追加で質問項目を追加していくという形式をとった．

インタビュー参加者の募集の際には筆者の知人に SNS やメールを通じて直接的に呼びかけをしたほか，自身の所属するプログラミングサークルに実験参加者募集の文面の掲載を行った．実験参加者の募集条件は過去に何らかの形式のコーディング面接を受けたことがあることとした．このような条件で募集を行ったところ，過去にコーディング面接を経験したことのある男子大学生 8 名にインタビューを実施することができた．インタビュー時間は 40 分程度とし，対面でのインタビューが可能な人に関しては対面で，遠方に住んでいて対面でのインタビューが難しい人はビデオ通話を用いたインタビューを実施した．インタビューは 1 対 1 形式で行い，インタビュー後には謝金として 2000 円分の図書カードを支払った．8 名のインタビュー参加者の学年やプログラミング経験に関するデータを表 1に示す．

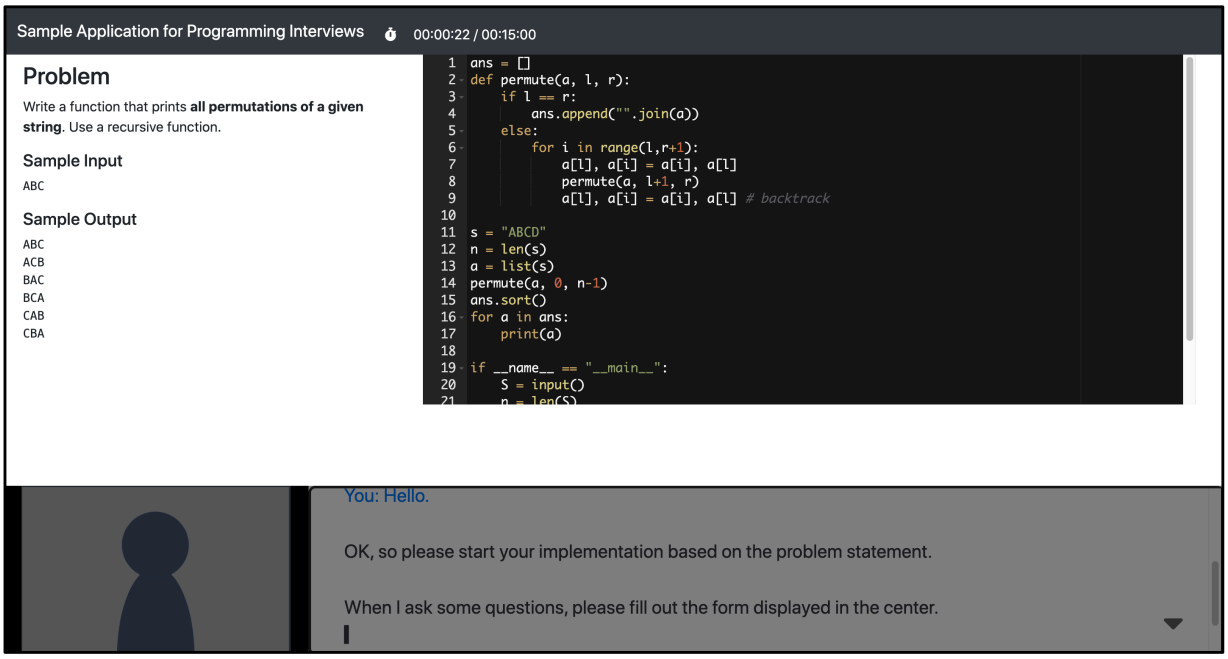

図 1: コーディング面接の練習を行うシステムの全体図．左に問題文，右にエディタを配置し，画面下部には面接官とのコミュニケーションを想定したチャットの画面が配置されている．

表 1: 事前調査に参加した学生のプログラミング経験などの情報．

#	学年	プログラミング経験・用途
PA1	学部 4 年	3 年程度，Web サービス制作
PA2	学部 3 年	3 年程度，競技プログラミング
PA3	修士 1 年	4 年程度，CS の研究
PA4	修士 1 年	5 年程度，Web サービス制作
PA5	修士 2 年	1 年程度，趣味，就活対策
PA6	学部 3 年	6 年程度，Web サービス制作
PA7	修士 1 年	8 年程度，競技プログラミング
PA8	学部 3 年	2 年程度，低レイヤ開発

3.2 インタビュー結果

インタビュー調査を行った結果，コーディング面接を経験したことのあるユーザからどのような意見が得られたかについてその概要をまとめる．ユーザから得られた意見は大きく「面接前の準備」と「面接中に感じること」の二種類に分類することができた．この二種類の状況に分けてユーザから得られた意見を実際の発言とともに紹介する．

3.2.1 面接前の準備

事前にコーディング面接の形式などが知らされている場合，ユーザは面接で高いパフォーマンスを実現することができるような事前準備を行うことが多い．準備の種類は出題される問題を想定したもの，面接時に使用するホワイトボードなどのツールに特有のもの，面接官との会話を意識したものなど様々なタイプが存在する．インタビューを実施した実験参加者 8 人のうち 6 人がコーディング面接に臨むにあたって何らかの準備を行ったと回答した．

面接前の準備は，大きく分けて問題の対策を行う場合と，面接で使うツールや環境に慣れるための対策の二種類に分類することができた．問題の対策に関しては，オンラインジャッジなどのサービスを用いて，出題される問題を想定した上でその分野の問題の知識を事前に身につけておく準備が数多く見られた．一方で面接の環境に慣れるための準備に関しては，面接で使うエディタに慣れることや面接官と会話するシチュエーションを想定して説明の練習をすることなどが挙げられた．

> 「ソートとかだったら，いろいろなソートの違いとかも説明できるように　全部のソートを書き下せるようにしといた　そこら辺はいろいろ調べて聞かれるって書いてあったから」(PA5)

> 「Topcoder の標準のエディタとかでやると，なかなかしんどい思いしながら行けるからなれるかもしれない．結構そういう恵まれた環境じゃなくてテキストファイルとかで練習するのはいいかな」(PA4)

3.2.2 面接中に感じること

面接中に感じることとしては，コーディング面接に特有のシチュエーションに関する不安などについて言及するものが多かった．コーディング面接時では複数人の面接官に見られている状況でコーディング作業を行うため，通常時のコーディングと比べてプレッシャーを感じ，いつも通りの実力を発揮することができないという意見が見られた．また，コーディング面接では必ずしもコンパイルをして明確な回答を出すタイプの形式ではないことから，自分のコードが正しく書けているかを確認する能力やそれを説明に落とす能力が必要とされることも分かった．

「考える時間の猶予が短いかなって感じが　待たせてる感って言うのが自分の中にあって詰まった時に考えてる時に焦っちゃうってのが自分の中にあるなって感じた 自分が書いているコードの意図を説明しながら書かなきゃいけないので」(PA6)

また，ホワイトボードを用いたコーディング面接では手書きでのコードとなるため厳密性がそこまで重要でなくなる．このような場合では面接官とのコミュニケーションの比重が大きくなり，ユーザ体験に大きな違いが出てくることがわかった．

「ホワイトボードの方がやりやすかったね　ホワイトボードでやってるとちょっと適当でいいかなって気持ちで挑めるんだけど、キーボードでやってるとちゃんと動くものを書かないといけないなっていう意識が出ちゃって」(P4)

3.2.3　インタビュー結果のまとめ

問題を解くために必要なアルゴリズム的な能力に関しては問題の演習を積み重ねていくことで伸ばしていくことができるが，コーディング面接に特有であるコードの説明能力やコーディング中の能力に関しては事前の対策があまり存在せず本番で期待通りの実力を発揮できないという意見が多く得られた．また，コーディング面接の形式に慣れるためのシステムはあまり存在せず，本番での形式に戸惑うケースも見られることがわかった．そこで本研究では，コーティング面接を受ける前の準備段階に注目し，コーディング面接に見られる特有の形式に対応できるようなシステムを構築することを目的として設定する．

4　システム実装

4.1　システムの概要

前章のインタビュー結果からコーディング面接の練習を支援するようなシステムを設計することで練習環境が周囲にないようなユーザでも事前に面接の練習を行うことができると考えた．本章では実際に練習を実施するために設計したシステムを導入し，このシステムが持っている機能について説明を行う．図 1に本システムのメインとなる画面を示す．

4.1.1　問題文およびエディタの配置

問題文およびエディタは現在広く普及しているコーディング支援のシステムである LeetCode のデザインを採用し，左側に問題文，右側にエディタを表示する方式をとった．エディタ部分は JavaScript で作成された Web ベースのエディタである Ace[3] を使用した．Python や C++など様々な主要言語のシンタックスハイライトおよび補完をサポートしている．

本システムに採用する問題はコーディング面接で問われる問題の中から代表的なものを選択した．出題する企業の目的によってその分野は様々存在するが，今回は比較的出題されることが多く出題形式も体系化しやすいアルゴリズムに関連した分野を選択した．ユーザによって適切な難易度の問題は異なることから，採用する問題には難易度の幅を持たせて以下の四種類を選択した．なお，問題の傾向や難易度はコーディング面接に関する対策を論じている書籍である Programming Interviews for dummies [7] を元に決定した．

- 与えられた整数が素数であるかを判定する問題
- 与えられた数を素因数分解する問題
- 順列の全列挙を再帰関数を用いて実装する問題
- Fenwick tree を実装する問題

4.1.2　面接官とのチャット機能の実装

コーディング面接を意識したリハーサルをするために，コーディング中の面接官とのコミュニケーションを想定することは重要である．本システムでは単に問題とコーディングができる環境を提供しただけではなく，コーディング中に特定のタイミングに面接官からの質問を受け取ることで模擬的な対話環境を実現した．以下では具体的に実装したシステムの説明を示す．

図 2が面接官から質問が来た時の画面を示している．面接官からの質問は後述する特定のタイミングに提示され，ユーザは画面中央に表示されたフォームに回答を入力する．回答フォームは敢えてコードエディタの前面に表示することでフォームの入力中はコーディングの作業を中断させるようなインターフェースとした．このようにコーディングの作業を意図的に中断することで，自分のペースではコーディングを進めることができずに定期的に面接官から説明を求められるようなコーディング面接の状況を再現しようと試みた．

面接官からは現在までに実装した内容の説明を問うことで，実装能力だけでなく実際に説明する能力も練習できることを試みた．問題の種類ごとに具体的な質問を用意することも考慮したが，現時点でのシステムでは下記のような共通の質問を提示することで現在のコードに対する理解度や説明能力を評価することを試みた．

- 今のところどの程度まで実装が進んでいるか
- 問題を解く上で今の方針と異なるものは存在するか

4.2　面接官からの質問提示のタイミング

本システムで面接官を提示するタイミングについて，コーディング作業中のどの段階で送信するのがコーディング面接の状況をよく想定したものとなるかを考える．実際のコーディング面接では参加者が意図しないタイミングで面接官からのコメントや質問が入ることがある．本システムはこうした本番特有の状況にできるだけ近づけるため，質問を送るタイミングを考えるにあたり user interruptibility と呼ばれる概念を導入した．

user interruptibility とは「ユーザが今の状態でどれだけ外部からの阻害要素を受け入れることができるか」という客観的な指標である [1]．この指標が高い時には「外部からの阻害を容認でき，現在のタスクに影響をあまり及ぼさない状態」であり，逆に低い時には「現在のタスクの効率を大きく阻害する状態」ということになる．この user interruptibility をユーザの行動から推定する方法は様々存在する．例えば小林らが行った研究 [8] では，メールの通知を user interruptibility の高い状態を

3） https://ace.c9.io/

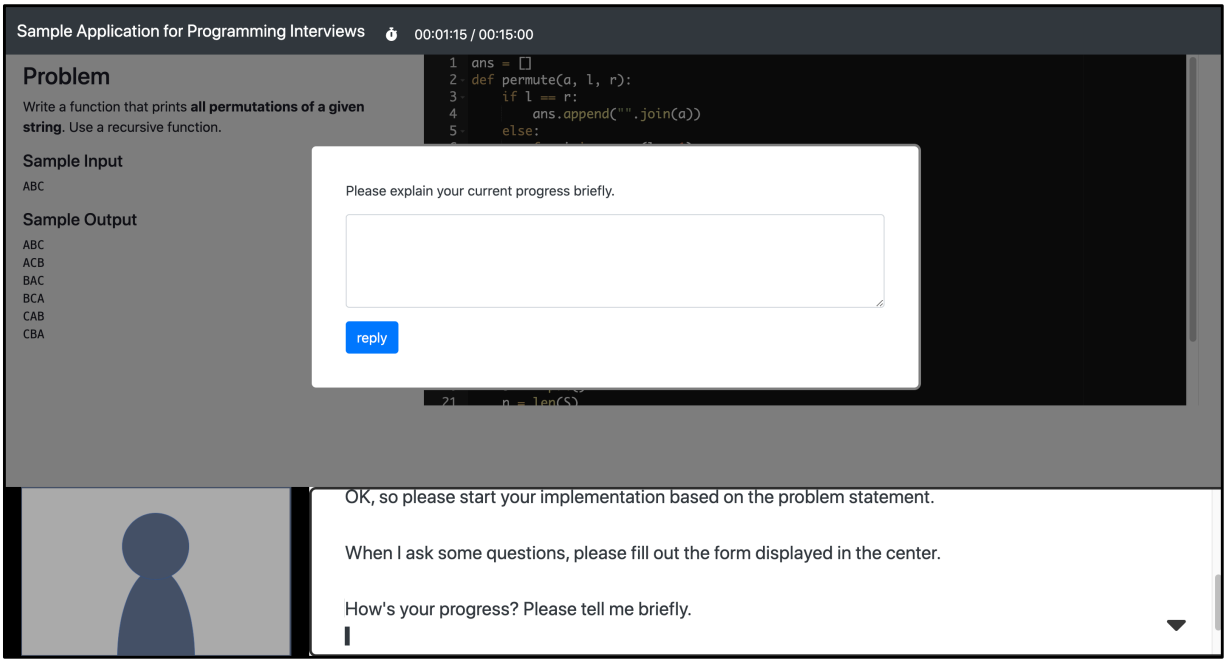

図 2: 面接官からのチャットが来た時の様子．ユーザは面接官からの質問に答えるように画面中央に出現するテキストボックスに回答を入力する．

待ってから送信するようなシステムの設計を行った．user interruptibility の高さはパソコン内部のアプリケーションスイッチの動作を検知することで推定し，低い状態にメールを受信したら通知処理を中断することで進行中のタスクを阻害しないような工夫を行った．

user interruptibility を検知する方法はアプリケーションスイッチを監視する他にもキーボードの入力を用いたもの，パソコンの前に感圧パットを敷くことで物理的な動作を取得するもの [4,10,11] など様々な種類が存在する．本研究では追加の装置を必要としないことやコーディングという作業が文字の入力作業の寄与がとても大きいことなどから，ユーザのキー入力を user interruptibility を推測する指標として用いることとした．

5 システムを用いたユーザ実験

現時点で制作したシステムについて，実際にコーディング面接を経験したことのあるユーザに使用してもらうユーザ実験を実施した．この実験の目標は現在のシステムをユーザ側から評価してもらい，どのような点を今後改善していくべきかの議論につなげるためである．

5.1 実験の流れ

ユーザはデプロイされた本システムに自身の所持するパソコンでアクセスすることで実験を行う．実験にかかる時間はシステムの使用とインタビューを通して 40 分程度である．実験の一連の流れを図 3に示す．ユーザは実験に関する説明を受けた後，先にあげた四種類の問題の中から自分に適したレベルの問題を選択して実験を開始する．難易度の目安としてはこちらで設定したシステムの制限時間である 15 分程度で実装することのできる難易度をユーザの主観で選択してもらった．実験中には基本的にシステムが提示する指示にしたがって実装を行ってもらうが，進め方についてわからないなどの質問に限ってこちらから返答するような形で実験を進めた．

表 2: ユーザ実験に参加した学生のプログラミング経験などの情報．

#	学年	プログラミング経験・用途
PB1	学部 4 年	6 年程度，Web サービス制作
PB2	学部 4 年	3 年程度，Web サービス制作
PB3	学部 4 年	2 年程度，低レイヤ開発
PB4	修士 2 年	5 年程度，Web サービス制作

5.2 実験参加者

実験参加者は前回のインタビューと同様にコーディング面接を過去に経験したことを条件に募集を行い，20 代の大学生 4 人を対象に実験を行った．各学生のプログラミング経験などの情報を表 2に示す．実験はリモートで行い，実験中のユーザのコーディングの様子およびインタビュー時の音声および作業中の実験参加者のパソコンの画面の録画を行った．

6 結果

本章ではユーザ実験を行った実験参加者 4 名から得られた現在のシステムに関する意見や感想を類似したトピックごとにカテゴライズし，具体的な発言内容と共に紹介する．

6.1 補完機能やエディタの性能について

現在のシステムでは補完機能やシンタックスハイライトを有効にした状態で実験を行った．この機能について便利だという意見があった一方，実際のコーディング面接とは異なる環境であるという意見も見られた．

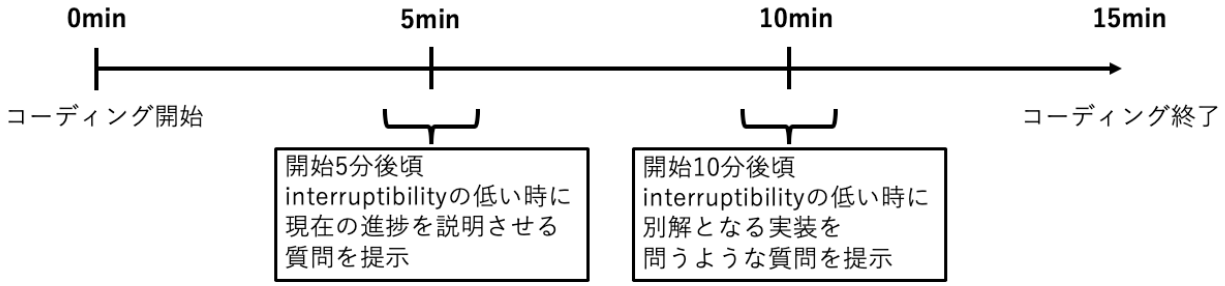

図 3: 今回のユーザ実験の時系列．コーディング開始から 5 分後と 10 分後のタイミングでキー入力から user interruptibility を計測し，値が低い状態の時に面接官から質問が提示される．

「補完とかシンタックスハイライトが効かないと不便だと思ってて，今日はその辺が効いてるのでだいぶやりやすいなと思いました.」 (PB1)

「ホワイトボードコーディングに関してはもっとコーディングに関して話すよりは実装方針こうですよねみたいなコミュニケーション的な側面がグッて上がるのかなって気がしていて.」(PB2)

6.2 質問の提示方法について

現状のシステムでは特定の時間にコメントの提示を行ったが，この提示方法の改善案などの意見もいくつか見られた．

「何分間隔でメッセージが来ますみたいに事前に書いてあれば心の準備ができるのかなと思っていて，5 分間隔という頻度はこれ以上は入れない方がいいのかなと思った.」(PB2)

「ヒントを型みたいに出してくれると嬉しいし，自分の手が止まっているのをわかってくれると嬉しいかな.」(PB4)

「理想は相手の進捗状況に合わせて具体的なものがあるとユーザ体験は良さそう.」(PB3)

「実は裏で回っていてこれ本当にコンパイルに通ると思いますかと教えてくれたりとか，このケースダメなんじゃないですかとかをするとそれっぽい質問を生み出せるんじゃないかなと.」(PB1)

6.3 コーディング面接の評価指標に関して

コーディング面接では実装と説明の両方の能力が評価されるが，どちらに重点を置いているかを明確にすべきであるという意見が見られた

「詰まったらコミュニケーションでカバーすれば点数になるのか，それともちゃんとコードをかかせるコーディングが重視なのか，どっちに力を注いでいいのかがわからなかった.」(PB2)

「コード 70%返信内容 30%って書いてあれば精神的にコーディングの方やった方がいいのかとかがわかる.」(PB2)

6.4 その他システムに関する意見など

特定のカテゴリには属していないものの，システムの構造やユーザ体験について重要だと感じた意見をいくつか紹介する．

「一人でやるには限界があるかなってのがあって友達同士でできるものがあったらそれはそれで練習になるかなとは思った　複数人でやるツールとしては使い勝手が良さそう」(PB1)

「インタラクションがないならもう少しインプットとアウトプットを明確に書いてくれると嬉しいかもしれない」(PB4)

7 考察

本章ではユーザ実験およびインタビュー調査で得られた結果を整理しつつ，よりコーディング面接のリハーサルを想定したシステムとするために本システムが満たすべき要件について明確にしていく．インタビュー調査において頻出した話題について取り上げ，それぞれについて今後着手していくべき課題について述べる．

7.1 コーディングとコミュニケーションのバランスについて

LeetCode などのコーディング面接の問題を解くことのできる既存のサービスでは，説明を行う機能などは存在せず純粋にコーディングをする能力をみている．本システムでは実際のコーディング面接にみられるような面接官とのコミュニケーションを想定することを目標としていることから，こうした既存サービスがカバーしていないインタラクティブな対話の部分を重点的にカバーする必要がある．実際に実験参加者にコーディングを行ってもらったところ，対話の機能は備わっているもののコーディング作業の占める割合がほとんどであるため既存の問題を解くサービスとあまり差が見られないという意見が見られた．

現状のシステムではコーディングをメインに据えたシ

ステムで特定の時間にコーディング中の特定の時間に質問を提示するという方式をとっているが，より質問の種類や頻度を増やし説明する能力を問うようなシステムにしていくことが考えられる．ただ，正しいコードを書くことを重視するケースもあるため，このバランスをユーザが自分で制御できるような機能を搭載することも想定している．

7.2 ユーザ側からの面接官に対するアクションについて

ユーザが自分の実力より難しい問題を選択した場合，実験の途中で手が完全に止まってしまうケースが見られた．通常のコーディング面接ではこのような状況になると面接官からの提案やヒントなどが出されるが，本システムでは user interruptibility およびスタートからの経過時間を基準に提示を行っていたためユーザの状況に合わせた動的なアクションの提示ができていなかった．手が止まっている状況ではユーザは何らかの課題に直面していることが多い．通常の業務であれば検索をすることで解決することも多いがコーディング面接の種類によっては検索が許可されていないものも存在する．こうした状況において，ユーザ側からのヒントの要求などの必要性が生じてくる．ユーザが適切なタイミングで手をあげることによってシステムが複数の選択肢の中から現時点で最も適したヒントの提示を行うというシステムが想定される．現時点で最も適した選択をどのように行うかについてはあとの項でその手法の案について述べる．

7.3 システムに搭載されたエディタが持つべき機能について

本システムはオープンソースのライブラリである ace.js を用いて実装しており，実験を行った状態ではシンタックスハイライトや補完機能を有していた．通常の業務であればこれらの機能は必要なものであるが，ホワイトボードや通常のプレーンエディタによる形式のコーディング面接が存在するため，これらの機能は必ずしもコーディング面接のリハーサルに貢献するとはみなされないことがわかった．

また，ユーザが書いたコードの正当性を評価するためコードの実行機能についても様々な意見が得られた．現在のシステムではコーディングの内容よりもそこに至った思考や説明能力を重視したいという理由でコードの実行機能を搭載していなかった．この設計について，実際の面接ではコードの実行はなく擬似的なコードを書いたからリハーサルとして適切という意見がみられた一方，自分のコードが正確に書けているか確かめるためにはコードの実行があった方がいいという意見も挙がった．この 2 つの両者の意見の折衷案としては，バックグラウンドで定期的にコードを実行し，不正解となるような入力に基づいたコメントを行うことで動的にコメントを提示するような対話的環境ができるというものが挙がった．このようにすることでユーザの進捗状況をシステムが定量的に把握できるようになり適切なフィードバックを返すことができるようになることが期待される．

```
1
2  def get_factors(N):
3      # write your code here
4
5
6
7  if __name__ == "__main__":
8      N = int(input())
9      factors = get_factors(N)
10     for f in factors:
11         print(f)
```

図 4: テンプレートとなるコード断片の例．初期状態で書かれている部分は書き換え不可にすることでユーザごとの実装の構成をある程度統一することを目指す．

7.4 ユーザの書いたコードを元にした進捗の推定

現在のシステムではユーザが課題に応じてコードをどの程度実装したかは取得しておらず，特定のタイミングで固定の質問を提示している状況である．理想的な対話環境では現在のコードを元にタスクがどの程度進んでいるかを動的に取得し，その段階に合わせた質問を提示すべきである．ここでは今後の展望としてユーザの記入したコードを元に進度に応じた適切な応答を提示するシステムの概要を示す．

まず言語は Python に限定し，それぞれの問題に対して正しい解法となるようなコードを収集する．同一の解法に対する様々な書き方を収集したいことから，コードはクラウドソーシングなどを用いて一定数集めることを考える．なお，ユーザごとに大きく書き方がぶれることを防ぐためにコードには図 4のような基本となる型を用意してそこにコードを追加していくことを考える．このようにして収集したコードに対して，ある程度まとまりのある実装ごとにコード断片に分割しタグを付与することを考える．コード断片への分割は Python の構文解析木を用いてパースすることも考えられるが，改行やインデントの位置などを用いてルールベースでの分割を想定している．ユーザがコーディングを行っている最中には一定のタイミングでコードの中身を確認し，コード断片をチェックすることで現在の実装の想定解に対する網羅度をチェックする．この処理を行うことでユーザがどの段階まで進んでいるかをある程度の精度で予想することが可能となり，現在の進捗に合わせた適切なフィードバックが可能になると推測される．

7.5 本章のまとめ

本章でユーザ実験の結果をもとにした今後行っていくべきシステムの改善案について言及した．ここまでに述べたことを簡潔にまとめると以下のような主要な要素に集約することができる．

- コーディング作業と対話的な説明をどの程度の比率で組み込むかどうかの検討
- ユーザの今の進捗をシステムを定量的に把握しフィードバックを提示すること
- 補完やシンタックスなどの機能を想定するコーディング面接に合わせて調節する機能

- ユーザ側からの自発的な質問やアクションへの対応

このような要素をシステムに取り入れていくことで，より各ユーザの要望に沿ったユーザ体験の提供が可能になっていくのではないかと考えている．

8 おわりに

本研究ではコーディング面接の練習を支援するようなシステムを構築することを目標とし，システムに必要な要件を確認した上でプロトタイプを作成した．また，作成したプロトタイプを用いて実際に実験参加者にコーディングを行ってもらうことで現在のシステムに足りていない点を明確にし，よりユーザの行動に沿った対話的なシステムを実現するために今後行っていくべきポイントの洗い出しを行った．ユーザ実験によって得られた意見を元にして，今後はユーザの進捗状況に応じた適切な質問や情報の提供，ユーザ側からの自発的なアクションに対する反応などを中心に機能の追加および改善を進めていく．

謝辞

本研究を進める上でご協力いただいた実験参加者および研究室のメンバーの皆様に感謝いたします．

参考文献

[1] Brian P Bailey, Joseph A Konstan, and John V Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Interact*, Vol. 1, pp. 593–601, 2001.

[2] Mahnaz Behroozi, Alison Lui, Ian Moore, Denae Ford, and Chris Parnin. Dazed: measuring the cognitive load of solving technical interview problems at the whiteboard. In *2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*, pp. 93–96. IEEE, 2018.

[3] Kathy Charmaz and Linda Liska Belgrave. Grounded theory. *The Blackwell encyclopedia of sociology*, 2007.

[4] James Fogarty, Andrew J Ko, Htet Htet Aung, Elspeth Golden, Karen P Tang, and Scott E Hudson. Examining task engagement in sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 331–340, 2005.

[5] Denae Ford, Titus Barik, Leslie Rand-Pickett, and Chris Parnin. The tech-talk balance: what technical interviewers expect from technical candidates. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 43–48. IEEE, 2017.

[6] Mohammed Ehsan Hoque, Matthieu Courgeon, Jean-Claude Martin, Bilge Mutlu, and Rosalind W Picard. Mach: My automated conversation coach. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 697–706. ACM, 2013.

[7] Eric Butow John Sonmez. *Programming Interviews For Dummies*. 2019.

[8] Yasumasa Kobayashi, Takahiro Tanaka, Kazuaki Aoki, and Kinya Fujita. E-mail delivery mediation system based on user interruptibility. In *International Conference on Human-Computer Interaction*, pp. 370–380. Springer, 2015.

[9] Wyko Rijnsburger and Sven Kratz. Personalized presentation annotations using optical hmds. *Multimedia Tools and Applications*, Vol. 76, No. 4, pp. 5607–5629, 2017.

[10] Takahiro Tanaka and Kinya Fujita. Study of user interruptibility estimation based on focused application switching. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pp. 721–724, 2011.

[11] Takahisa Tani and Seiji Yamada. Estimating user interruptibility by measuring table-top pressure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pp. 1707–1712. 2013.

[12] Ha Trinh, Reza Asadi, Darren Edge, and T Bickmore. Robocop: A robotic coach for oral presentations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 1, No. 2, p. 27, 2017.

[13] Ha Trinh, Koji Yatani, and Darren Edge. Pitchperfect: integrated rehearsal environment for structured presentation preparation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1571–1580. ACM, 2014.