卒 業 研 究 報 告 書

題　目　　ダンジョン探索型ゲームによる
ソフトウェアリポジトリ間のメトリクスの可視化

指導教員　　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　18122037

氏　　名　　玉井　陽博

令和4年2月10日提出

ダンジョン探索型ゲームによるソフトウェアリポジトリ間のメトリクスの可視化

令和 4 年 2 月 10 日　　18122037　玉井　陽博

概　要

ソフトウェアを構成するソースコード等のファイルメトリクスを分析し，結果を可視化することでソフトウェアの開発を支援することができる．中でもインタラクティブなインタフェースにより可視化されることは，ソフトウェアメトリクスへの理解を深めることへの動機付けとなる．

当研究室の先行研究にリポジトリ探訪者がソフトウェアリポジトリ内をインタラクティブに探索できるダンジョン探索型ゲーム GitQuest が存在する．これは，Git リポジトリの 1 つのディレクトリごとにマップが，ディレクトリ内のファイルからマップに配置するオブジェクトが生成され，そのマップ内を探索するというものである．

また，オープンソースソフトウェアやバージョン管理システムを用いたソフトウェア開発が近年では普及しており，これらのシステムに開発支援のための可視化を適合するにあたって，2 つのリポジトリおよびバージョン間での分析を行える機能が必要である．

本研究では GitQuest に，2 つのリポジトリ間のメトリクスの比較結果を反映させる機能の追加を中心に，他に GitQuest における可視化ツールとしての問題点の改善を目的とした GitQuest における機能の拡張を行い，これを通じて新たな 2 つのリポジトリ間のメトリクスの違いをインタラクティブに可視化する手法を提案する．

目　次

1. 緒言

ソフトウェアの開発状況の把握を支援するために，ソフトウェアメトリクスの様々な可視化の研究が行われている [1]．中でも可視化ツールにインタラクティブなインタフェースを導入することにより，ソフトウェアメトリクスへの理解を深めることへの動機付けを与える試みがなされている．例として Heijo[2] や Codosseum[3], IslandViz[4] および CodeHouse[5] といった可視化ツールの存在が挙げられる．

また，近年ではオープンソースソフトウェア開発が積極的に行われるようになったことや，バージョン管理システムである Git (注 1)により大量のソフトウェア開発履歴を収集し利用出来るようになったことにより，一層プロジェクトやコミット間でのリポジトリマイニングの有効性が高まっていると考えられる．そのため，インタラクティブなインタフェースによるソフトウェアリポジトリ間の可視化手法の進展が今後のソフトウェア開発支援や，リポジトリマイニングによる研究のために重要であると考えられ，実際にそのような可視化ツールの研究も行われている [6].

当研究室の先行研究として既に Git リポジトリ内をインタラクティブに探索できるウェブアプリケーション GitQuest v1.0[7] が存在する．これはバージョン管理システムである Git の管理しているリポジトリを対象にディレクトリの構造やファイルの場所，および規模からメトリクスを取得し，それを用いてダンジョンのマップを生成し，その中をキャラクターの操作により自由に探索するダンジョン探索型ゲームであり，ゲームを通じて Git リポジトリ内のメトリクスを可視化できる．

本研究は，GitQuest のアプローチからソフトウェアリポジトリおよびバージョン管理システムにおけるバージョン間の可視化を行うために，GitQuest v1.0 における問題点の改善を行い，2 つのリポジトリ間のメトリクスの比較結果を反映させる機能を追加した上で，それを GitQuest v2.0 とし，これを通じて 2 つのリポジトリ間のメトリクスの違いをインタラクティブに可視化する手法を提案する．

本稿の構成を次に示す．第 2 章では本研究の目的を述べる．第 3 章では本研究のベースとなる GitQuest v1.0 の基本機能および問題点について述べる．第 4 章では GitQuest v2.0 におけるう変更点について，可視化対象，表現方法，実装方法を順を追って説明する．第 5 章では本研究の利用ケースについての考察を行う．最後に第 6

(注 1)：https://git-scm.com/

章で以上の結果をまとめる.

2. 目的

近年のソフトウェア開発にはGitなどのバージョン管理システムが用いられており，バージョンごとの差分を評価する機会が多い．そこで本研究の目的は，ユーザーが直感的かつ抵抗なく気軽にリポジトリ間のメトリクス評価を行える可視化ツールの開発である．このツールは，リポジトリ間のソースコードメトリクスを，Git等のバージョン管理システムを用いたソフトウェア開発経験を持たないユーザーに対しても直感的に伝わり，かつ気軽に評価を行えるような実装をコンセプトとしている．

直感的に伝わり，さらに手軽に取り組むことができる要素を持つインタフェースの一つに，ゲーム形式のインタフェースが考えられ，これにより手動でファイルを確認することでリポジトリ内の探索を行うことに伴う煩わしさの削減が期待できる．

当研究室の先行研究に「GitQuest v1.0」というリポジトリ内をインタラクティブに探索できるウェブアプリケーションがある．これはバージョン管理システムであるGitが管理している単一のリポジトリからメトリクスを取得し，メトリクスを基にダンジョン探索型ゲームのマップを生成し，キャラクターを操作することでその中を自由に探索できるというものである．

ところが,「GitQuest v1.0」にはいくつかのインタラクティブな可視化ソフトとしての問題点があり（詳細は次節にて説明する.），また単一のリポジトリしか可視化することが出来ないため，そのままでは本研究の目的であるリポジトリ間のメトリクスの可視化を達成出来ない．しかし直感的かつ抵抗なくリポジトリの可視化を行うという点においてこれを満たしているため，これの機能を拡張することにより本研究の目的を満たすことが可能であると考えた．

以上より,「GitQuest v1.0」にリポジトリ間のメトリクスを比較し，マップに反映させる機能を追加することにより「GitQuest v1.0」の機能を拡張した「GitQuest v2.0」を開発することで以上の要件を満たせると考え，実装に踏み切るに至った．

また,「GitQuest v1.0」には可視化ツールとしての問題点があったため，そちらの改善も行う．

「GitQuest v1.0」の仕様および改善した問題点については次章にて詳細を説明する．

3. GitQuest v1.0

本章では，先行研究の GitQuest v1.0[7]（以下，本章においては GitQuest と表記する）について述べる．

3.1 仕様

GitQuest は，Git リポジトリ内のメトリクスを解析し，ディレクトリからマップ，ファイルからノンプレイヤーキャラクターを生成し，ユーザーがプレイヤーを操作することでマップ内をインタラクティブに探索できるウェブアプリケーションである．プレイ画面を図 3.2 に示す．

プレイヤー以外のキャラクターのグラフィックにはファイル拡張子に応じてグラフィックが設定される．(図 3.1)

他のキャラクターに隣接することによってそのキャラクターに対してアクションが実行でき，該当するキャラクターのファイル名が表示される．

マップにはディレクトリの規模に応じて木々や段差や建物といった移動が制限される障害物が配置される．

マップの随所に設置されたゲートは子ディレクトリのゲーム内表現であり，くぐると現在のマップが消去され，該当するディレクトリのマップが新規作成される．移動先ディレクトリでは，ゲート，キャラクターに加え上層のディレクトリに帰る BACK ゲートが存在する．

また，同一のディレクトリ構造からマップを生成する場合，オブジェクトの配置を含めたマップの地形は常に同一のものが生成される．

図 3.1 GitQuest のキャラクターグラフィック

図 3.2　GitQuest のプレイ画面

3.2 使用方法

以下に GitQuest を起動するまでの手順および操作方法を示す.

3.2.1 起動手順

1. オンラインに接続し，GitQuest を起動する.
2. ホストアドレスに”/#/user/reponame”を指定し，GitHub 内にあるリポジトリのリポジトリ名を [reponame] に，所有するユーザー ID を [user] に指定してサイト内遷移する.
3. マップが生成され，表示される.

3.2.2 操作方法

- 十字キーでプレイヤーを操作する.
- Space キーで他のキャラクターに対してアクションを実行する

3.3 実装

GitQuest は図 3.3 に示すワークフローで動作している.

リポジトリ情報およびリポジトリ内のファイルメトリクス取得は GitHub API を用いて行われ，ページの更新，ディレクトリの移動などが行われる度に GitHub API からデータを取得しマップも再生成する.

GitQuest は JavaScript を用いたフレームワークである CreateJS (注 2)及び React (注 3) を用いて作られている．React はルーティングに用いられ，また CreateJS はゲーム画面の描写に用いられている.

3.4 問題点

GitQuest にはリポジトリ内をインタラクティブに探索し可視化するアプリケーションとしていくつかの問題点がある．そのうち，本研究で改善した 3 つの問題点を以下に示す.

(注 2): HTML5 を用いたリッチコンテンツを構築するためのライブラリ，https://reactjs.org/
(注 3): ユーザーインターフェースを構築するためのライブラリ，https://createjs.com/

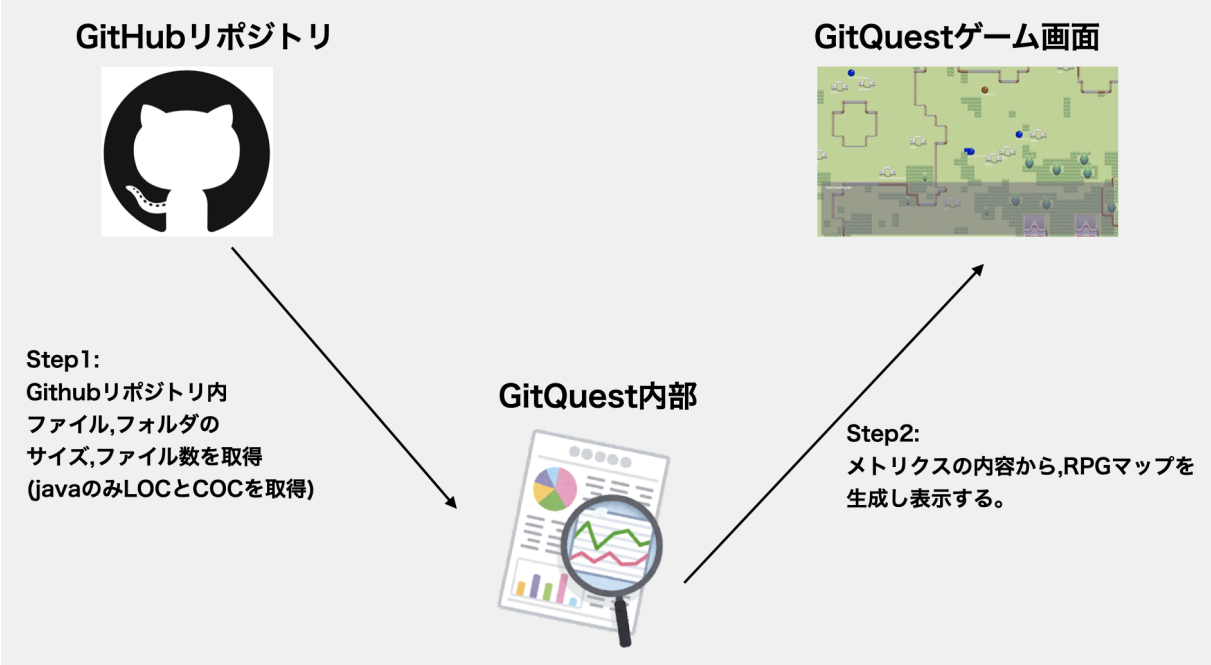

図 3.3　GitQuest のワークフロー

問題 1. ゲーム開始を直感的に行うことができない

GitQuest では URL にリポジトリ名やユーザー ID を入力して対象のリポジトリを指定し，起動していた．この仕様では起動方法が分かりづらいため，タイトル画面を用意し，対象のリポジトリ名やユーザー ID ををGUI を用いて入力することでゲームを起動できるようにする必要がある．

問題 2. GitHub API 以外からメトリクス解析やディレクトリ構造の取得を行うことができない

GitHub API からは，ディレクトリ内のサブディレクトリ，ファイル一覧およびそれらのファイル名，ファイルサイズといったメトリクスが取得出来ていたが，ファイルの内容そのものを取得することが出来なかった．そのため例えばファイル内の関数やクラスの総数といった詳細なメトリクスを解析することが出来ない．また，GitHub 内に存在しないリポジトリを対象にすることも出来ない点も問題である．さらにマップ生成の度に GitHub API に問い合わせを行わねばないため，規模の大きなリポジトリを対象にした場合動作が不安定になり，また頻繁にマップ生成を行うと GitHub API のリクエスト制限を受けてしまうという欠陥がある．

問題 3. リポジトリ内探索の動機付けが不足している

リポジトリ内探索の動機付けのため，ゲーミフィケーションを用いる必要がある．ゲーミフィケーションとは，あるタスクにゲームの要素を持ち込むことでユーザーの動機付けを行うものである [8][9]．リポジトリ内探索をゲーム化することでプログラミング知識のないユーザーが積極的にリポジトリ探索を行いやすい環境を作る．

4. GitQuset v2.0

本章では，リポジトリ間の比較機能を中心に，本研究で追加した機能について述べる．

4.1 ゲーム起動までの流れ

ゲーム起動までの手順を示す．GitQuestServer についての詳細は 4.6.1 節で述べる．

1. GitQuestServer 内の LocalRepo ディレクトリに解析したいリポジトリファイルを追加する．
2. GitQuest v2.0 と GitQuestServer を起動する．
3. GitQuest v2.0 のタイトル画面に参照するリポジトリ名を入力する．
4. オプションを入力し，「Create Gamedata」ボタンをクリックすると解析データが生成される．
5. 「Play」ボタンをクリックするとゲームが起動する．

手順 1 は手順 4 以前であれば任意のタイミングで行うことができる．手順 4 は以前に全く同一のリポジトリから生成された解析データが存在する場合は省略できる．また，比較対象に同一の Git リポジトリ 2 つを指定し，それぞれのリポジトリファイルのコミットをブランチの切り替えなどを用いて別々に指定することで同一リポジトリにおけるバージョン間の違いを可視化することもできる．

4.2 タイトル画面

GitQuest v2.0 では GitQuest v1.0(以下 v1.0) での問題 1 を解決するためにタイトル画面を導入した．タイトル画面を図 4.1 に示す．

①では対象とするリポジトリを指定する．2 つのリポジトリを入力した場合リポジトリ間の比較を行うモードになる．Repoisitory Name2 が空欄であった場合は Repository Name の入力を参照して v1.0 と同様に 1 つのリポジトリを探索するモードとなる．

②の「Play」ボタンはクリックすることでゲームを開始出来る．

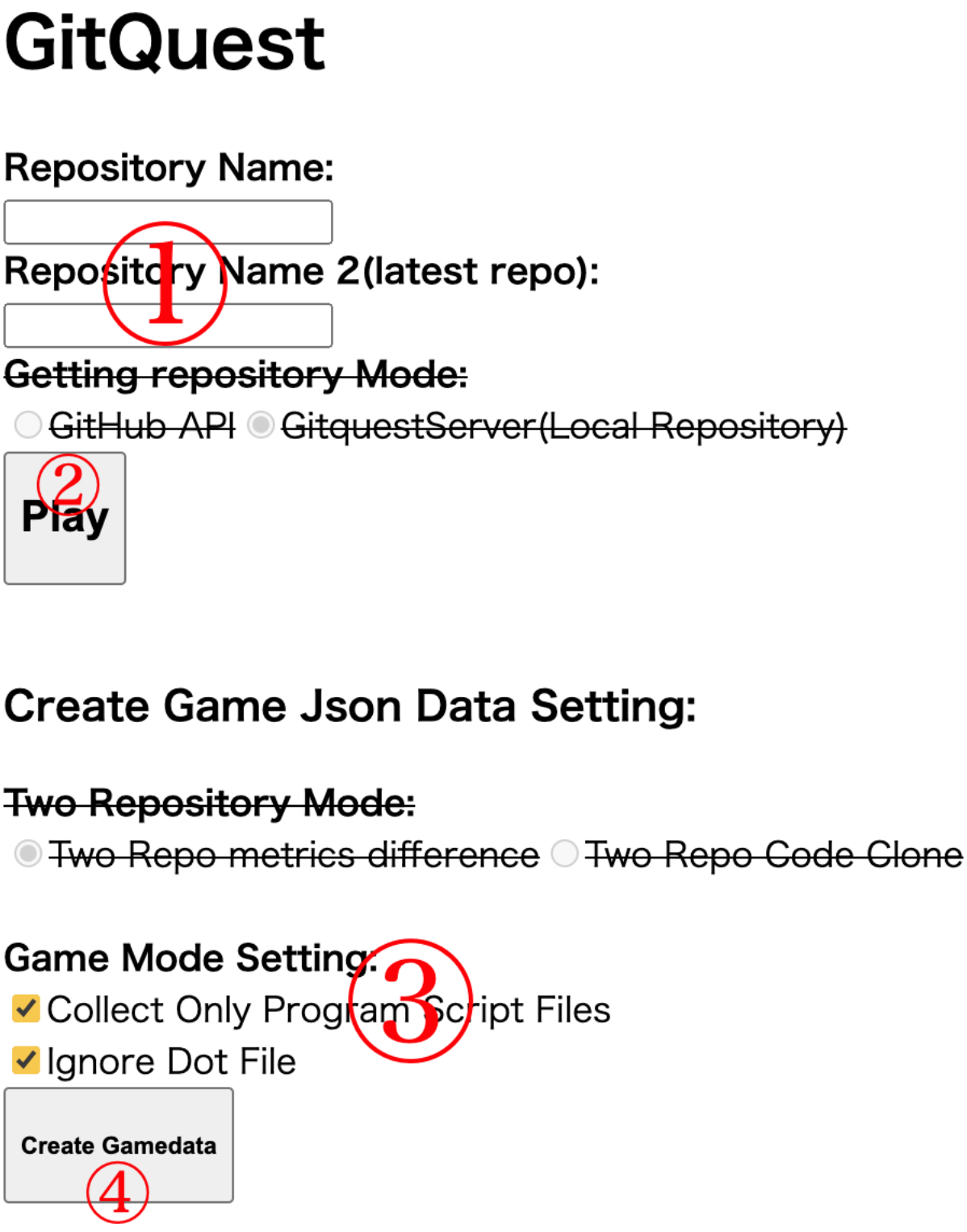
GitQuest
Repository Name:
Repository Name 2(latest repo):
Getting repository Mode:
GitHub API
GitquestServer(Local Repository)
Play
Create Game Json Data Setting:
Two Repository Mode:
Two Repo metrics difference
Two Repo Code Clone
Game Mode Setting:
Collect Only Program Script Files
Ignore Dot File
Create Gamedata
1
2
3
4

図 4.1　タイトル画面

③ではマップデータ生成時のオプションを指定できる.「.」から始まるディレクトリやファイルを対象外とすることが出来たり，優先ボーナスを獲得出来るファイルの対象を java,c,cpp といったスクリプトファイルのみを対象とするかどうかが選択出来る．優先ボーナスについての詳細は 4.5 節にて述べる.

④の「Create Mapdata」ボタンをクリックすることでゲームを開始出来る.

4.3 ゲーム画面

GitQuestv2.0 ではプレイヤーキャラクターのグラフィックの変更や，ゲーム画面に表示する情報の追加を行った．図 4.2 にゲーム画面を示す.

①の「SCORE:」には現在のスコアが表示される.

②はプレイヤーである.

③の「CD:」には現在プレイヤーが存在するディレクトリが表示される．これによりどのようにディレクトリを移動すれば目的のファイルまで移動可能であるかどうか v1.0 と比べて容易に判断出来るようになった.

④の「NEXT:」には 4.5 節で述べる優先ボーナスを得ることの出来るファイルが表示される.

4.4 可視化の対象

それぞれのリポジトリ間での可視化するメトリクスおよびその活用方法を表 4.1 に示す.

ここで，**LOC**(Line Of Code, ファイルの行数),**FOC**(Function Of Code, 関数の数) および **COC**(Class Of Code, クラスの数) といったメトリクスは Java,C++,C 言語といったプログラミング言語の，java,cpp,c 形式のファイルに対応している．ただし C 言語にはクラス機能が無いため COC を取得することは出来ない.

GitQuest v2.0 では，以下のような仕様によりファイルメトリクスを表現する.

4.4.1 ソースコード規模

ファイルに対応するキャラクターグラフィックにソースコードの規模（以下コード規模）を反映する．このコード規模は LOC,FOC,COC, を用いて以下の式 4.1,4.2 に

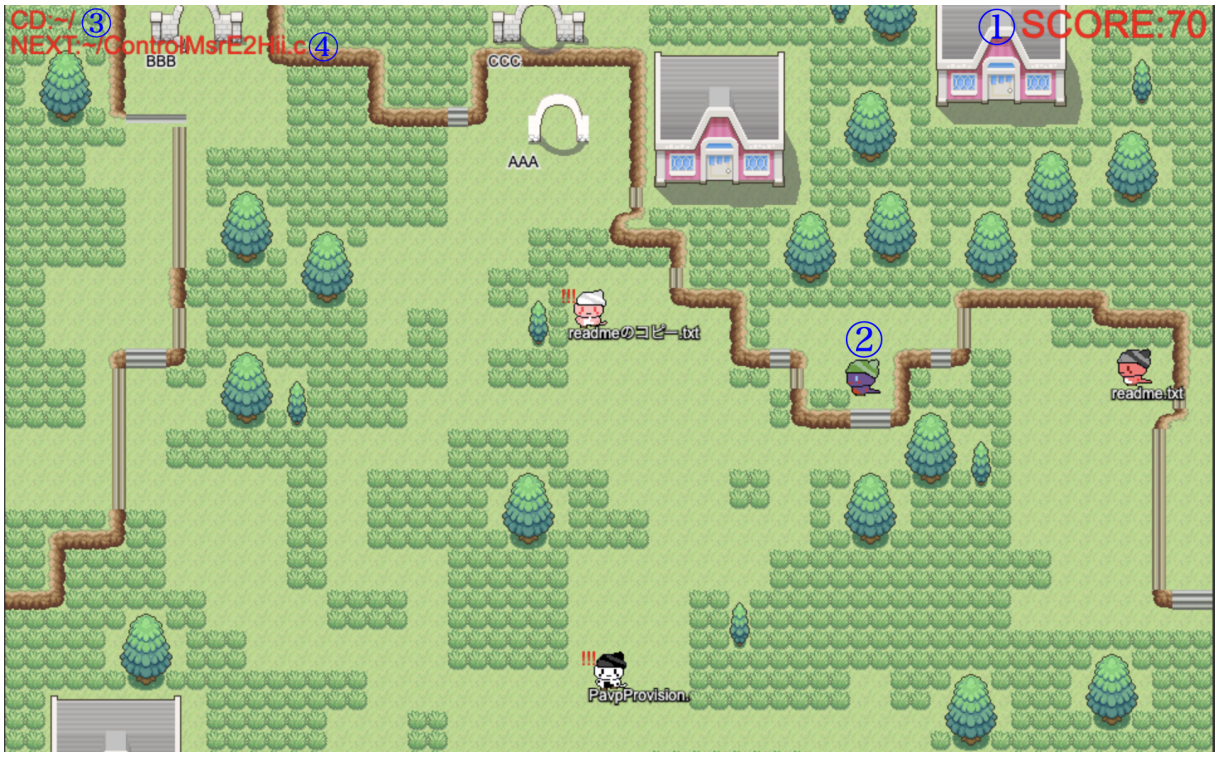

図 4.2　ゲーム画面

表 4.1　可視化するメトリクス

取得メトリクス	活用方法
ファイル名	ファイルに相当するキャラクターを区別する
ディレクトリ名および子ディレクトリ	子ディレクトリをゲートの形で表現する
LOC (Line Of Code, ファイルの行数) FOC (Function Of Code, 関数の数) COC (Class Of Code, クラスの数)	ファイル規模の算出に用いる
ディレクトリ内のファイル数 リポジトリサイズ	マップの広さの決定に用いる

よって表される．K は 0-8 段階で表されるコード規模であり，K の小数点以下は切り上げる．

$$k = LOC + 10FOC + 25COC \tag{4.1}$$

$$K = 2\log_{10}(k) \tag{4.2}$$

4.4.2　2 つのメトリクス間における相違の視覚化

メトリクス間の相違を可視化するモードでは，新リポジトリと旧リポジトリをそれぞれ指定して，それぞれのフォルダやファイルが同一のディレクトリに存在するかどうかを調べる．

ファイルに対応するキャラクターのグラフィックは，両リポジトリの同ファイルのコード規模を比較し，その差によってグラフィックが変更される．新リポジトリの方が規模が大きい場合は図 4.3，旧リポジトリの方が規模が大きい場合は図 4.4 のグラフィックのようにキャラクター本体の色がコード規模に対応した濃淡で表現される．また，両リポジトリで規模が同一の場合は図 4.5 の左端になる．ここで，一方のリポジトリにしかファイルが存在しない場合は，存在しないリポジトリ側にコード規模 0 のファイルが存在するものとしてコード規模を比較する．また，両リポジトリ間でのファイルの有無は図 4.5 のようにキャラクターが被っている帽子の色で表現される．

ディレクトリ間の比較について，対応するディレクトリのグラフィックは図 4.6 のようなゲートで表現される．親ディレクトリに移動する階段は図 4.7 のように表現する．

4.5　リポジトリ内探索の動機付け

v1.0 での問題 3 を解決すべく，リポジトリ内探索の動機付けのため，キャラクターを調べることでスコアが入手できるゲーム機能を導入した．スコアは画面右上に表示される．

未だに調べていないキャラクターを調べるとスコアをコード規模 K(式 4.2 を参照)に応じて $15K + 10$ 点入手することが出来る．すでに調べているキャラクターに話し

図 4.3　新リポジトリの規模が大きい場合．右に行くにつれて規模の差が大きくなる．

図 4.4　旧リポジトリの規模が大きい場合．右に行くにつれて規模の差が大きくなる．

図 4.5　左より順に，リポジトリ規模が同一な場合のキャラクター，新リポジトリに存在しない場合の帽子，新リポジトリにのみ存在する場合の帽子，どちらにも存在する場合の帽子．

図 4.6　ゲートのグラフィック．左から順に，削除されたディレクトリ，両方に存在するディレクトリ，追加されたディレクトリを示す．

図 4.7　階段のグラフィック．

かけてもスコアは増減しない．未だに調べていないキャラクターは左上に「!!!」マークが表示されているため判別することが出来る．

また，リポジトリ規模の差分が大きいファイルを順番に調べると追加で 30 点のスコアを入手する事ができ，次に調べるファイルのパスは画面左上の「NEXT:」で表示される．

4.6 実装

4.6.1 GitQuestServer

GitQuestServer は，Node.js というフレームワークによって動作するバックエンドサーバーである．GitQuest v2.0 において，リポジトリのメトリクスを解析して結果を JSON 形式でファイルに保存し，リクエストがあればフロントエンドに JSON 形式でその内容を送信するという機能を持つ．これにより v1.0 における問題 2 を解決することが出来た．

4.6.2 GitQuest 側の変更点

フロントエンドである GitQuest について，v1.0 では GitHub API からメトリクスを取得するようにしていたが，v2.0 では GitHubAPI との通信機能を廃止し，代わりに GitQuestServer から JSON 形式でデータを受け取りそれを基にマップを生成する．

4.6.3 システムのワークフロー

GitQuest v2.0 は GitQuestServer と連携することで動作する．本節では図 4.8 に示す GitQuestv 2.0 のワークフローについて説明する．

1. GitQuest v2.0(以下，フロントエンド) が GitQuestServer（以下，バックエンド）にリポジトリ名を指定しメトリクス解析の命令を行う．
2. バックエンドがバックエンド内の LocalRepo ディレクトリ内を参照し，指定されたリポジトリのメトリクス解析を行う．
3. バックエンドが解析結果をバックエンド内の Gamedata ディレクトリ内に JSON 形式で出力する．

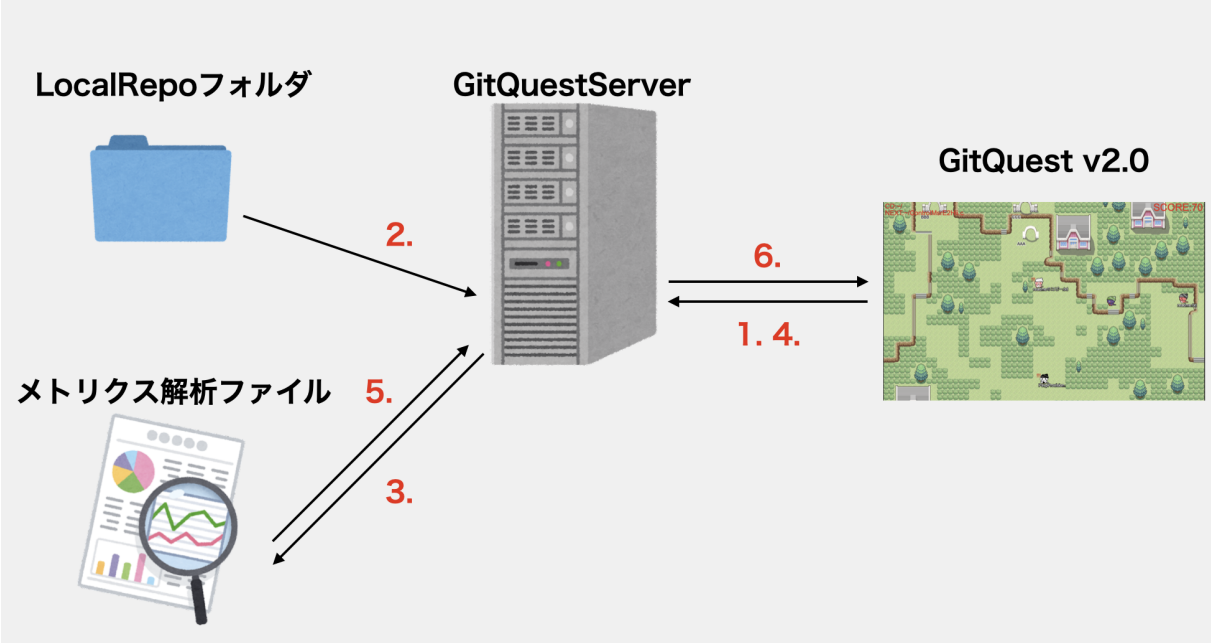

図 4.8　GitQuest v2.0 のワークフロー

4. フロントエンドがバックエンドにリポジトリ名を指定し解析結果データを問い合わせる．
5. バックエンドは Gamedata ディレクトリを参照し，指定されたディレクトリのメトリクス解析ファイルが存在する場合は，json 形式のままフロントエンドに送信する．
6. JSON 形式のデータを受け取ることに成功した場合は，フロントエンドでゲームが開始される．

5. 利用シナリオ

Git を用いた開発経験が殆どない A さんが，Git を用いたチームでのソフトウェア開発に初めて携わることになった．ここで，そのソフトウェアは，1 つの git リポジトリで管理されており,「master」ブランチはメジャーリリースの管理を行い,「staging」ブランチは最新の開発確認環境の管理を行っている．

ある日，A さんは同じチームの B さんからバグ修正を任されたがこの時,「staging」ブランチを pull した後に新たな修正対応用のブランチを作成し，その後変更を commit してから push した後に B さんのコードレビューを経て staging ブランチにマージする，という手順で対応するように指示を受けた．

この時，間違えて master ブランチから新しいブランチを作って対応してはいけないのだが，これはメジャーアップデート直後等の状況を除いて両ブランチの最新コミットにおけるファイルの内容が異なるからである．

そこで，A さんが GitQuestv2.0 を用いて master ブランチと staging ブランチ間で，どれだけファイルが違うかを直感的に確認することができれば，対応するブランチを間違えたり，無闇に master ブランチへ別のブランチをマージしたりしてはいけないということの理解を深めることが出来る．

また，上記 2 つのブランチ以外の主要なブランチも対象にして GitQuestv2.0 を使用すると，より Git への理解を深めることができる．

このように，GitQuest v2.0 の利用シナリオの 1 つに教育目的での利用が考えられる．

6. 結言

本研究ではダンジョン探索型ゲームによる，2 つのリポジトリ間のメトリクスの可視化機能を持つ，GitQuest v2.0 を開発した．

他にもタイトル画面の作成やグラフィックの変更といった利便性の向上や，スコア機能の追加といった動機付けの向上，GitQuestServer による詳細なメトリクス取得機能等といった拡張も行った．

これらの拡張された機能は，ソフトウェア開発におけるバージョン管理に関する経験の無いユーザーがバージョン管理システムに関する知見を得る足掛かりに役立つことが期待される．

また，GitQuest v2.0 に直感的かつ手軽に取り組むことができる要素を持つインタフェースを導入し，GitQuestServer の導入により詳細なファイルやリポジトリのメトリクス解析機能に関する拡張性が得られたことから，より詳細な Git リポジトリでのバージョンごとの遷移の視覚化や，コードクローン [10] 等といった新たなメトリクスの視覚化，より強力な動機付けのためのゲーム機能の拡張など，GitQuest には今後も改良，拡張を行っていく余地がある．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系水野　修教授，崔　恩瀞助教に厚く御礼申し上げます．本研究の基礎となった GitQuest v1.0 の研究をなされ，さらに本研究にあたっても貴重な助言を多数頂きました，本学情報工学専攻　上北　裕也先輩，本報告書執筆にあたり貴重な助言を多数頂き，校閲にもご協力を頂いた本学情報工学課程，渡邉　紘矢君をはじめとする，ソフトウェア工学研究室の皆さん，学生生活を通じて著者の支えとなった友人，著者の学生生活を全面的に支援して頂いた家族に深く感謝致します．

参考文献

[1] M. Lanza, “The evolution matrix: Recovering software evolution using software visualization techniques,,” Proceedings of the 4th International Workshop on Principles of Software Evolution, pp.37–42, 2001.

[2] 大神勝也，中才恵太郎，畑　秀明，松本健一，“Heijo: 動的なコード実行可視化による java/android アプリケーションのリアルタイムプロファイラ,” コンピュータソフトウェア，pp.93–105，2019.

[3] 上村恭平，中才恵太朗，大神勝也，畑　秀明，一ノ瀬智浩，松本健一，飯田　元，“Codosseum : Oss プロジェクトモニタリング web サービス,” 日本ソフトウェア科学会大会論文集，pp.93–105，2017.

[4] M. Misiak, A. Schreiber, A. Fuhrmann, S. Zur, D. Seider, and L. Nafeie, “Islandviz: a tool for visualizing modular software systems in virtual reality,” Proceedings of the 6th Working Conference on Software Visualization (VISSOFT), pp.112–116, 2018.

[5] A. Hori, M. Kawakami, and M. Ichii, “Codehouse: Vr code visualization tool,” Proceedings of the 7th Working Conference on Software Visualization (VISSOFT), pp.83–87, 2019.

[6] R. Wettel and M. Lanza, “Visual exploration of large-scale system evolution,” Proceedings of the 15th Working Conference on Reverse Engineering(WCRE), pp.219–228, 2008.

[7] 上北裕也，“ソフトウェアリポジトリからロールプレイング風ゲームを生成するツールを利用したソースコードメトリクスの可視化,,” Technical report，卒業研究報告書, 京都工芸繊維大学，2020.

[8] S. Deterding, “Gamification: designing for motivation,” Interactions, vol.19, no.4, pp.14–17, 2012.

[9] W. Oliveiraa, O. Pastushenkob, L. Rodriguesa, A.M. Todaa, P.T. Palominoa, J. Hamaric, and S. Isotani, “Does gamification affect flow experience? a systematic literature review,” Proceedings of the 5th International GamiFIN Conference, vol.2883, pp.110–119, 2021.

[10] M. Godfrey and C. Kapser, "Copy-paste as a principled engineering tool," Making Software, pp.531–543, O'REILLY, 2010.