樹状整列課題計画書

第 1 版　2021/05/05
第 2 版　2021/05/12
第 3 版　2021/05/29
第 4 版　2021/06/23
作成者　岡本悠里(PM)

プロジェクト予定期間

2021 年 4 月 29 日～2021 年 7 月 23 日

プロジェクトメンバ

岡本悠里(プロジェクトマネージャ,デザイナ)
岡山紘大(サブプロジェクトマネージャ,プログラマ)
杉橋真輝(プログラマ)
山内龍我(プログラマ)
近藤英雅(チェッカ)
梶原隆太郎(チェッカ)

この計画書は京都産業大学情報理工学部 2021 年春学期ソフトウェア工学 2 の長期課題である樹状整列を作成するプロジェクトについて記したものある。プロジェクトの計画を示している。

0. 要旨

本プロジェクトは京都産業大学情報理工学部 2021 年春学期ソフトウェア工学 2 の長期課題の一部である。樹状整列を実現するアプリケーションを作成し、実際の企業で行われるプ ロジェクト開発を体験することが目的である。

クライアントを本講義の担当講師青木淳先生と仮定し、演習を行う。クライアントから要求仕様書を受理し、要求仕様書に従ったアプリケーションを完成させることを目標とする。本チームではこの目標を達成するために、クライアントとの協議を始め、試行錯誤を 繰り返し、その過程を記録、反省することでより良い成果物を提出する方針で開発を進める。開発はプロジェクト管理の知識体系 PMBOK[1]に沿って行う。

本計画書ではプロジェクトの計画について記す。ステークホルダーに配布することを想定している。

1. プロジェクトスコープマネジメント

1.1. 立ち上げ

樹状整列課題を行うにあたってプロジェクトチームを立ち上げる。プロジェクトマネージャ(以下、PM)を岡本悠里とし、プロジェクトを発足する(2021/4/29)。

1.2 スコープ計画

クライアント(青木先生)より、要求仕様書を受け取る。要求仕様書より作業範囲を確認する。確認した内容は以下の通りである。

<基本方針>

起動時：ウィンドウが一枚開く

設計：UML を使ってオブジェクト指向設計。デザインパターン MVC を採用する。

管理：全ドキュメント及び全ソースコードをリポジトリに格納する。リポジトリ自体を成果物に含める。

実装：コーディング規則遵守。

クラスの名称：アッパーキャメルケース

メソッドの名称：ロワーキャメルケース

リテラルの名称：アッパーケースやアッパーキャメルケース

変数やメソッドなど：ロワーケースやロワーキャメルケース

すべての名称は省略せずに意味を明確に示す。

テスト方針：ユーザの操作に関わるテストケースは設計段階でテスト仕様書にまとめる。

xUnit（JUnit や SUnit など）によるテストケースを行う。

テスト結果は報告書にまとめる。

各リリース、大きな修正の直後に全件実施（回帰テスト）を行う。

リリース方針：オープンソース。

リポジトリからチェックアウトして全ソースプログラムを獲得。

ソースプログラムから実行モジュールであるアプリケーションを構築するための手段（Makefile や Ant のようなコンパイル・リンク・インストール・テスト・アーカイブ作成などを自動化する手立て）を提供。

アプリケーションは簡単なコマンドによって起動。

取り扱い説明書（マニュアル）をリリースに含める。

<基本要件>

アプリケーションの用途：木構造をウィンドウの中で可視化できるようにする

対象者：コンピュータを利用する人たち全て

ハードウェア： Name: Apple Macintosh シリーズ （MacBook / MacBook Air / MacBook Pro / Mac mini / iMac / Mac Pro など）

Processor: Intel Core 2 Duo 2GHz クラス以上（64bit アーキテクチャの Xenon などを含む）

Memory: 1GB 以上

Hard Disk: 250MB 以上の空き容量

ソフトウェア構成：

OS X Yosemite (10.10) / OS X El Capitan (10.11) / macOS Sierra (10.12)以上

プログラミング言語　Java: JDK 1.8 以上　Python: Python 2.7 以上　Smalltalk:VisualWorks 7.6 以上

目標性能：一つのウィンドウに表示させるようにする

フォントサイズは Serif 系の標準体(plain)で 12 ポイント、ノードやリーフの間隔は横 25 ピクセル, 縦 2 ピクセルで表示。

木のサイズがウィンドウ内に表示できない場合もあるからスクロールでウィンドウ内を動かせるようにしておく。

木の内部をマウスでクリックしたら標準出力で名前を書き出す

100 個以下のノード&リーフは 3 秒以内に表示できるようにする。

木が完成していく過程をアニメーションとして見せる。

<開発体制>

プロジェクトマネージャは開発スケジュールの段階ごとにクライアントと仕様のチェックをする。

開発の不明点もクライアントと協議する。

<開発スケジュール>

開発スケジュールを開発計画書の中に明記。

ソフトウェア工学 II の講義期間内に行う。

1.3. スコープ定義

1.2 を参考にスコープ定義を行う。スコープ定義にあたっては、WBS を用いる。作業をトップダウンで分解し、階層構造で表す。作成した WBS は以下の図 1 である。

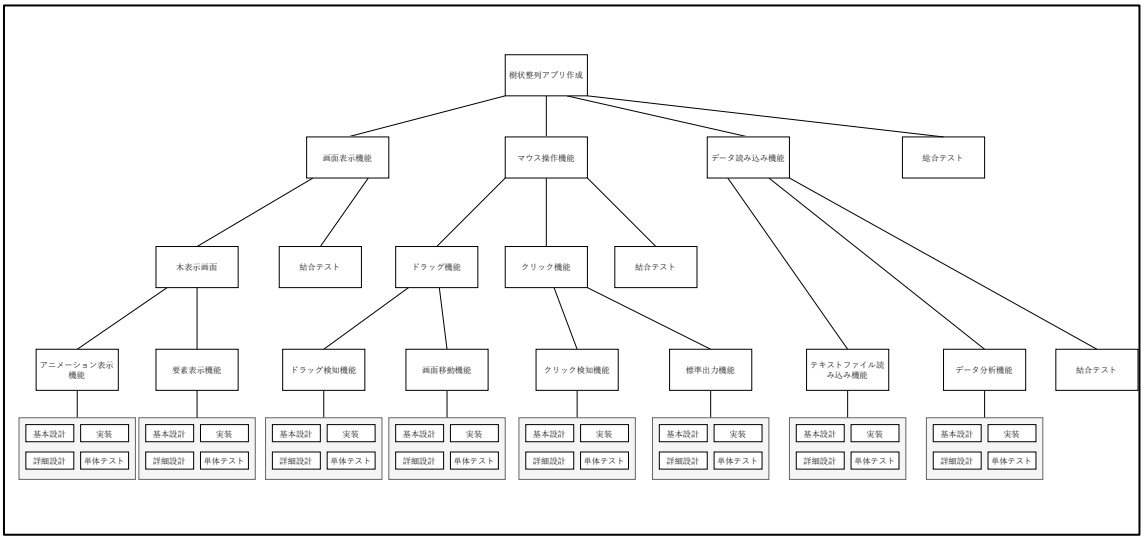

図 1：WBS による今課題のスコープ定義(Ver.3.0)

1.4. スコープ検証

上記のスコープ定義をステークホルダー全員に承認を受ける。第 1 版をもとにメンバ全員で協議後、第 2 版(図 2)を作成し、第 2 版をクライアントに提出した。クライアントからの指摘を踏まえ、第 3 版(図 3)を作成した。

1.5. スコープ変更管理

WBS の変更履歴を以下に示す。

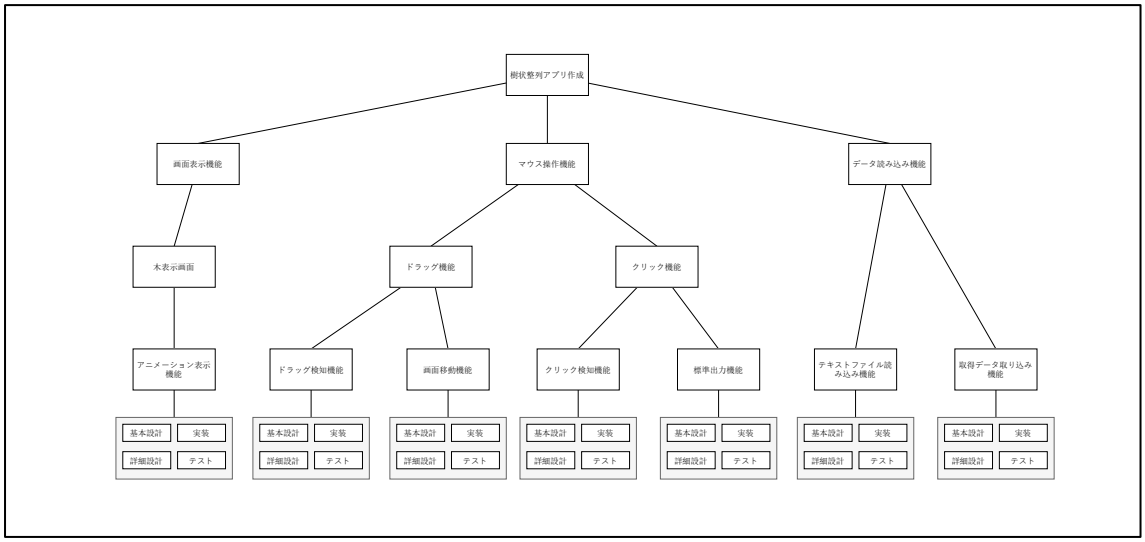

図 2：Ver.1.0

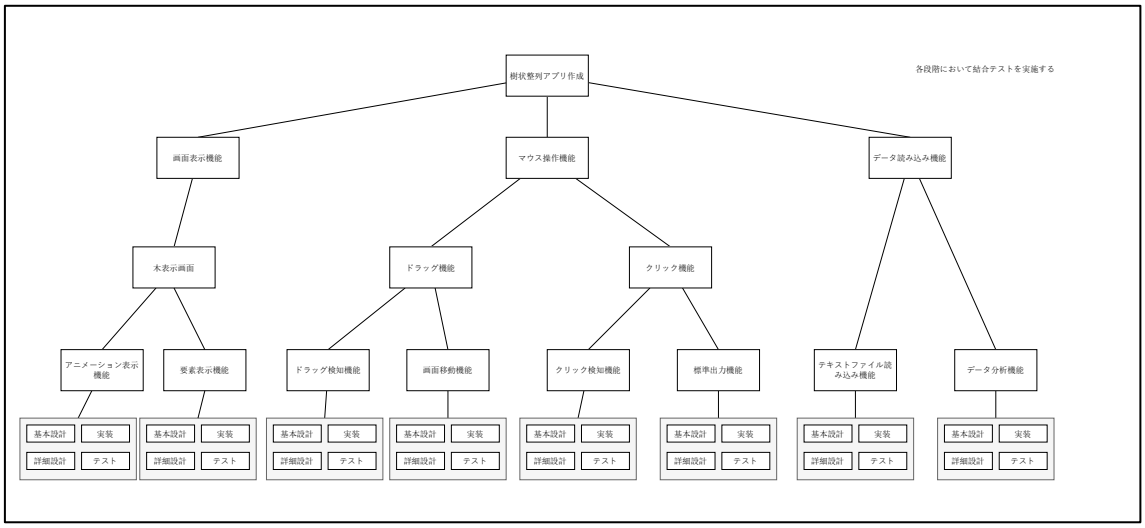

図 3：Ver.2.0

図 1～3 作成者：岡山紘大

2.1. プロジェクトタイムマネジメント

2.1. アクティビティ定義・アクティビティ順序設定・アクティビティ所要期間見積

1.3.において作成した WBS に基づき、アローダイアグラムを作成する。作成したアローダイアグラム(PERT 図)(図 4)を以下に示す。

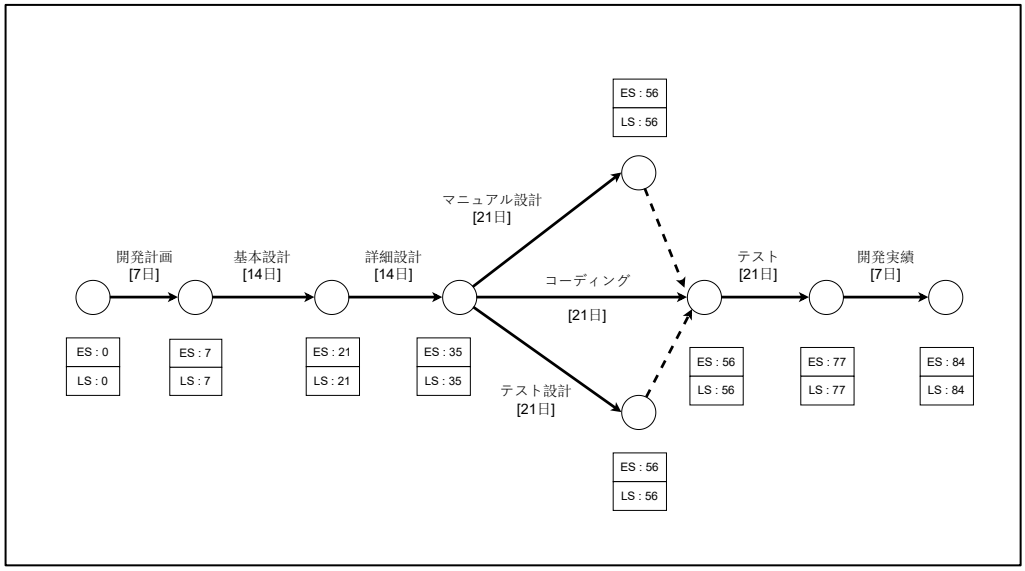

図 4：PERT 図

また、基本設計、詳細設計、コーディング、テストの詳細な PERT 図は以下の図 5 である。

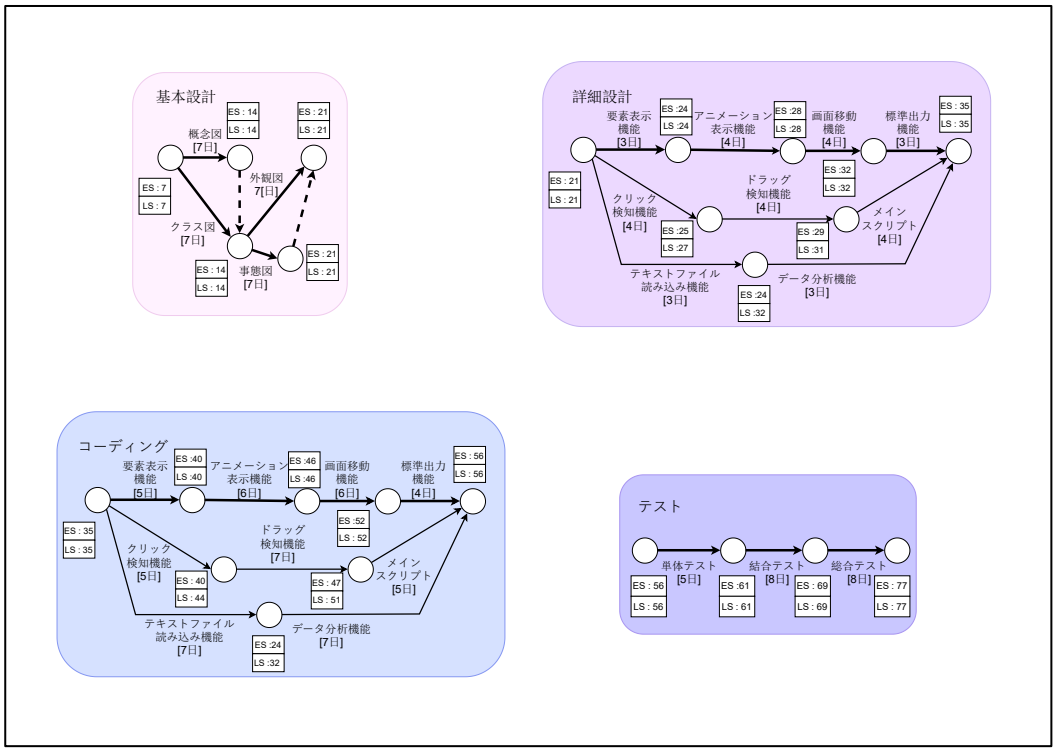

図 5：基本設計、詳細設計、コーディング、テストの詳細な PERT 図

図 4、5 作成者：岡本悠里、岡山紘大

2. 2. スケジュール作成

2.1.を元にガントチャートを作成する。作成したガントチャートは以下の図 6 である。使用ツールは GanttProject[2]である。

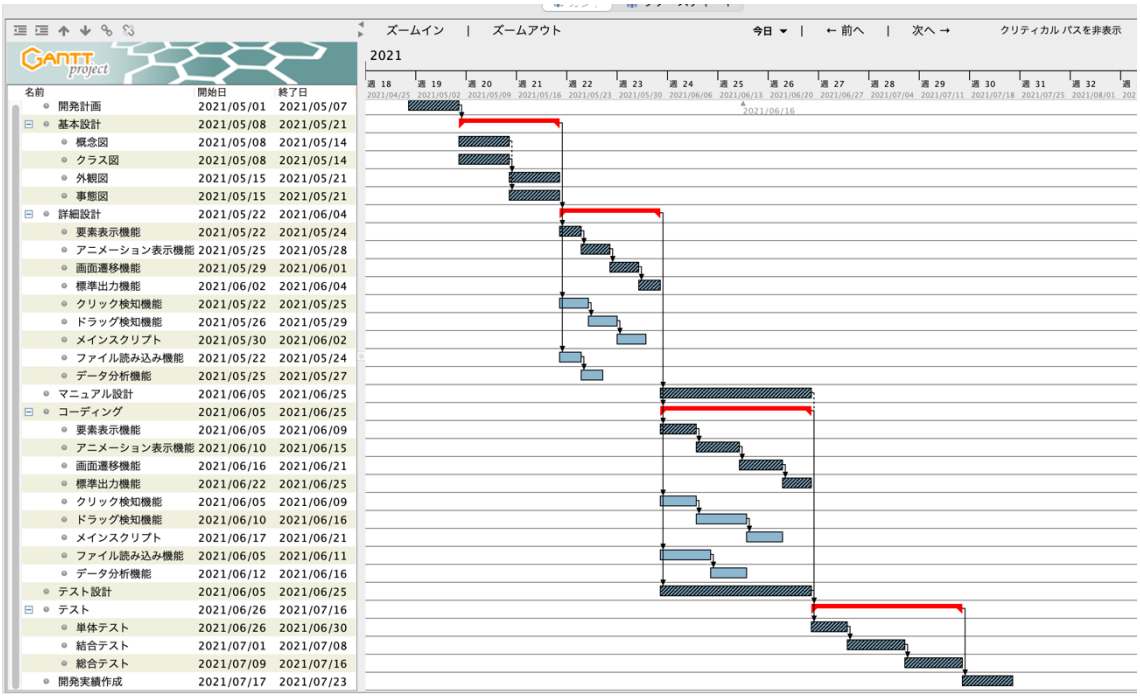

図 6：ガントチャート

また、ドキュメントは随時進行に合わせて作成することとし、特に時間は設けないこととする。

この作成したガントチャートをメンバ全員で作成したのち、クライアントに提出し、了承を得た(2021/05/11)。

図 6 作成者：岡本悠里

2. 3. スケジュールコントロール

計画通り行われているかどうか進捗状況を PM 及びサブ PM が常に監視、把握する。計画した日程より 3 日遅れが生じた際はメンバ全員で協議を行い、状況の再確認や開発体制の見直しを行うものとする。予定より早く進行している場合についても、必要に応じて計画の見直しを行う。

3. プロジェクトコストマネジメント

3.1. 資源計画

必要となる人員は 5 人である。プロジェクトの開始と同時に 5 人全員が必要となる。メンバは以下の 5 人である(50 音順)。

岡本悠里
岡山紘大
梶原隆太郎
近藤英雅
杉橋真輝

2021/06/01 追加
山内龍我

3.2. コスト見積

今回のプロジェクトは大学の授業でおこなわれる演習である。新たに人員以外の資源を必要としないので、以下の作業は割愛する。

3.3. コストの予算化

省略

3.4. コストコントロール

省略

4. プロジェクト品質マネジメント

4.1. 品質計画

品質は総合テスト、結合テスト、単体テストの 3 段階に分けて行う。総合テストの品質は JIS X O 129-1 (ISO 9126[3])に従って評価することとする。機能性、信頼性、使用性、効率性、保守性、移植性の 6 つの観点より評価する。

<総合テスト>

機能性は要求仕様書をもとにチェックリストを作成する。また、考えうる様々な状況を予想し、予想した条件をクリアできるかテストする。作成したチェックリストは以下の図 7 である。

概要	テスト内容
表示時間のテスト	100個以下のデータを複数パターン試し、それぞれが3秒以内に表示が完了するかどうかを確かめる。
マウスドラッグのテスト	アプリケーションがマウスドラッグ可能な画面でマウスドラッグを受け付けるかを確かめる。
スクロール時の描画のテスト	画面をスクロールしたとき、画面表示が正しく綺麗に表示されるかを確かめる。
クリックのテスト	アプリケーション全体を通して、クリック可能な箇所で正常にクリックができるかを確かめる。
ノードのテスト	ノードをクリックした時に、ノードの情報が正しく表示されるかを確かめる。
正常なデータの読み込みテスト	誤りのない木構造のデータを正常に読み込まれるか、100個以下のデータの読み込みが3秒以内に終わるかどうかを確かめる。
不正なデータの読み込みテスト	誤りのある木構造のデータを読み込んだ際、読み込めなかったことがきちんと表示されるかを確かめる。
文字コードのテスト	文字化けが発生していないかどうかを確かめる。
ウィンドウタイトルのテスト	ウィンドウタイトルがアプリケーションの名称になっているかどうかを確かめる。
異なる拡張子のファイルを開くテスト	ファイル選択時に、txt以外の拡張子を持つファイルを開くことができないかどうかを確かめる。
壊れたファイルを開くテスト	ファイル選択時に、壊れたファイルを選択すると読み込めなかったことがきちんと表示されるかを確かめる。
ノードの詳細が確認できるかのテスト	ノードをクリックした時に、そのノードの詳細のダイアログが表示されるかを確かめる。
ノードのダイアログが複数表示されないかのテスト	ノードをクリックした時に、すでにダイアログが表示されている場合何も起きないことを確かめる。
ウィンドウの最大化のテスト	ウィンドウを最大化して、表示が崩れないかを確かめる。
ウィンドウの最小化のテスト	ウィンドウを最小化して、表示が崩れないかを確かめる。
ウィンドウの最小幅が機能しているかのテスト	ウィンドウのサイズを小さくする際、縦400×横600より小さくならないことを確かめる。

図 7：機能性チェックリスト

使用性は以下の点を確認する(図 8)。以下の点を、良いを 5、どちらともいえないを 3、悪いを 1 として第三者に実際に使用してもらい、アンケートを取る。

- □ 操作方法がわかりやすいか
- □ スクロールの存在に気づいたか
- □ スクロールにストレスを感じたか
- □ クリックの機能に気づいたか
- □ ボタン配置に違和感を感じなかったか
- □ 起動時間にストレスを感じないか
- □ 見えにくい色使いではないか

図 8：使用性チェックリスト

効率性は profile、主に JConsole[4]を用いて検証する。以下の図 9 にその検証項目を示す。

概要	テスト内容
CPU使用率のテスト	jconsoleを用いてCPU使用率が40％を超えることがないことを確かめる。
使用メモリ量のテスト	jconsoleを用いてメモリの使用量が200MiBを超えることがないことを確かめる。

図 9：効率性チェックリスト

保守性は以下の図 10 の項目を確かめる。定めるコーディングスタイルは PMD[5]である。HTML のリンターには Another HTML Lint-Gateway[6]を採用する。

概要	テスト内容
JavaDocのテスト	JavaDocに必要事項が全て記入されているかどうかを確かめる。
コーディングスタイルのテスト	PMDを用いて、定めたコーディングスタイルに違反していないかを確かめる。
HTMLドキュメントのテスト	HTML文書全てを Another HTML Lint - Gateway でリント掛けし、100点であることを確かめる。

図 10：保守性チェックリスト

信頼性は以下の図 11 の項目について検証する。

概要	テスト内容
CPU負荷状態でのテスト	yes commandを用いて意図的にCPUに負荷をかけた状態でアプリケーションに不具合がないかを確かめる。
長期間稼働のテスト	24時間稼働させ、アプリケーションが途中で終了するなどの不具合が発生しないかを確かめる。

図 11：信頼性チェックリスト

移植性は mac, windows, Linux で検証を行う。実験 PC の具体的な性能を記した上で、機能性を全て満たすかどうかを確かめる。

<結合テスト>

結合テストは内部結合テストと外部結合テストに分けて実施する。内部結合はクラス内の 2 つのメソッド間で正しく連携できているかを確認するのに対し、外部結合テストは 2 つのクラス間で正しく連携できているかを確認する。MVC 間を見る時には 3 つの間でテストをする。2 つのダミーを用意し、注目する MVC のうち 1 つのみ完成したものとダミー2 つとの 3 つの間で連携を確認する。これを全ての MVC でクリアすれば全てダミーではない MVC でテストする。

<単体テスト>

単体テストは JUnit[7]を用いて検証する。

4. 2. 品質保証

行なったテストの詳細をテスト仕様書、テストの結果をテスト結果のドキュメントにそれぞれまとめ、提出する。

4. 3. 品質管理

品質全体の方針決定、監督及び品質の責任は PM が持つこととした(2021/4/29)。

新たなメンバの追加により、具体的なテスト方法については山内龍我が担当する(2021/06/02)。

5. プロジェクト人的資源マネジメント

5.1. 組織計画

このプロジェクトは京都産業大学の講義、ソフトウェア工学 2 での演習である。従って、メンバは受講生、クライアントは教師で構成される。メンバの技術的役割は以下の通りである。

プロジェクトマネージャ：岡本悠里
サブプロジェクトマネージャ：岡山紘大
デザイナ：岡本悠里
プログラマ：岡山紘大(V,M)、杉橋真輝(C,メイン)
チェッカ：梶原隆太郎、近藤英雅

組織構成を 2021/06/01 に更新した。更新された組織構成は以下の通りである。

プロジェクトマネージャ:岡本悠里
サブプロジェクトマネージャ:岡山紘大
デザイナ:岡本悠里
プログラマ:岡山紘大(V,M)、杉橋真輝(C,メイン)、山内龍我
チェッカ:梶原隆太郎、近藤英雅、山内龍我

また、M は MVC のモデル、V はビュー、C はコントローラを担当することを示す。メインは MVC のメインを担当することを示す。

5.2. 要員調達

5.1.に基づき、要員調達を行った。全員を調達することができた。

2021/06/01 に、メンバに新たに 1 人(山内龍我)を追加した。主にプログラム班での活動を予定しているが、初期のメンバより技術と経験において優れていると判断されるため、テストに関する分野でも助言を行う立場とする。

5.3. チーム育成

目標に向かってチーム意識を持って行動できるようにするため、定期的に会議を行い、連絡を密に取ることとする。会議はチャットのみではなく、音声通話を中心として行う。メンバの技術的なスキルアップのために、ツールの使用方法などの勉強会を開く。会議の進行は PM が中心となって行う。オープンな議論ができるように常にメンバ全員に気を配ることを

意識する。

6. プロジェクト伝達マネジメント

6.1. コミュニケーション計画

情報共有を確実に行うために、メンバ間の連絡は Discord[8]を使用して行う。メンバ全員が参加したサーバを新たに立ち上げる。サーバ内はドキュメント別、役職別などにカテゴリを分け、ドキュメントの種類、役職の種類などで区分したチャンネルを設立する。また、会議で話し合った内容は議事録として保管する。役職ごとにチャンネルを分けるのは、役職ごとに連携を取れるようにするためである。また、ドキュメントごとにチャンネルを分けることで、情報の整頓を試みる。

開発にあたって、メンバ全員の状況をお互いが把握する。定期的に会議を開き、PM が中心となって、達成状況や問題の有無などを確認することとする。会議は週 1 回以上をベースに音声通話で行う。極力全員が参加できるように日程の調整を行う。会議では、成果物の進捗状況の確認と問題の原因特定を行う。心理面にも配慮し、全員がモチベーションを維持できるように注意する。円滑なコミュニケーションはプロジェクトの進行において非常に重要な点である。メンバ全員は連携を密にとり、常に状況の把握に努めることとする。また、サブ PM は PM の補助及び、何らかの問題により PM が活動不可能になった際に一時的またはプロジェクト終了まで PM として活動することとする。不測の事態に備え、引き継ぎがいつ行われても良いように常に情報共有は密に行う。

役職間での連携を良好に保つために、役職ごとに臨時会議を開くことを適宜検討する。

クライアントとの連絡は、Gmail[9]または Teams[10]のチャットを通して行う。もしくは、必要に応じて、ビデオ会議または研究室へ直接訪問を行う。クライアントと協議を行う際は、原則として PM とサブ PM でクライアントと 3 者協議を行う。これは、2 者間による客観性や発言の責任を確保するためである。

6.2. 情報配布

メンバ間の情報または資料の共有は 6.1.で設立した Discord で行う。バージョン管理を行い、メンバ内で情報認識の差異が発生しないように努める。また、必ず資料配布の際は資料の説明を付随させることとする。メンバ全員は Discord の使用に慣れており、本手段での情報共有に支障はないと判断する。Discord 内での情報管理は PM が責任を持つ。

一時的にクライアントに情報を提供する際は画面共有などを用いて電子で提供する。完成した成果物はリポジトリである GitHub(git)[11]を用いて全ドキュメント及び全ソースコードを提供する。

6.3. 実績報告・完了手続き

別途報告書にて実績について記す。

7. プロジェクトリスクマネジメント

7.1. リスクマネジメント計画

リスクの特定はプロジェクトの進行過程でも繰り返し行うこととする。新たにリスクが生じた際は速やかにリスク対応計画を PM 及びサブ PM が立て、メンバへの周知及び協議を行い、実行に移す。リスク管理は PM が中心となって行う。リスク管理責任は PM が持つ。リスクマネジメントを行う前提条件は、プロジェクト終了まで地球が存在かつ人類が生存可能であることとする。

7.2. リスク識別

以下に考えられるリスクを仕分けして列挙する。

<環境が原因で発生するリスク>

自然災害や感染症の拡大などにより、授業の継続が困難になる。

<メンバに発生するリスク>

PC やネットワークの故障により連絡が取れなくなる。
体調不良または事故など身体的な事情により参加が困難になる。
感染症に罹患し、メンバ全体で集団感染が起こる。
心理的な問題が原因でメンバの一部でコミュニケーションに支障が生じる。
単位を落とす(または学びの成果がない)。
PM がなんらかの障害により一時的または永続的に継続が困難になる。

<クライアントに発生するリスク>

事故や病気などにより参加が困難になる。

7.3. 定性的リスク分析

7.2.で示したリスクについて、発生確率と影響度の予想を行い、以下の表 1 にまとめた。

表 1：定性的リスク分析結果

総合リスクランキング	リスク	発生確率	影響度
1	心理的な問題が原因でメンバの一部でコミュニケーションに支障が生じる。	大	大
2	自然災害や感染症の拡大などにより、授業の継続が困難になる。	大	大
3	事故や病気などにより参加が困難になる。(クライアント)	中	大
4	PM がなんらかの障害により一時的または永続的に継続が困難になる。	中	大
5	体調不良または事故など身体的な事情により参加が困難になる。(メンバ)	中	中
6	感染症に罹患し、メンバ全体で集団感染が起こる。	小	大
7	単位を落とす(または学びの成果がない)。	中	小
8	PC やネットワークの故障により連絡が取れなくなる。	小	小

7.4. リスク対応計画

7.3.を元に以下の対応を計画する。

授業の継続が困難になる場合は教師の指示に従う、リスクの転嫁を行う。

メンバがネットワークの不具合などで物理的に連絡が取れなくなった際は、連絡が取れなくなった対象者が早急に復旧を試みる。必要があれば外部の設備などを利用し、できるだけ速やかに PM に知らせる。状況を PM が把握し、継続可能か不可能かを判断する。また、代替措置などを検討する。決定事項は速やかにメンバ全員に周知する。続投不可能の際は緊急会議を開き、メンバ全員で協議を行い、人員配置の変更を行う。リスクの軽減を行うこととする。

メンバが一時的またはプロジェクト終了まで参加が困難になると見られる際には、メンバは必ず PM に報告を行う。会議の無断欠席などがないように事前連絡を行うこととする。計画に悪影響が懸念される場合は緊急会議を開き、メンバ全員で協議を行い、一時的または永続的に人員配置の変更を行う。リスクの軽減を行うこととする。

COVID-19 拡大の現在、集団感染が起こらないように原則として会議はオンラインでおこなうこととする。対面で会議を行う際は、飲食を伴わない会議とする。アルコール消毒及びマスクの着用を行う。換気を行い、距離を確保する。リスクの監視、コントロールを行う。

心理的な問題が原因でメンバの一部でコミュニケーションに支障が生じることがないよう、会議の際はメンバ全員に PM が気を配る。PM 及びサブ PM はこのリスクのトリガー発生を常に監視する。メンバ全員がこのリスクを理解し、円滑にプロジェクトが進行するよ

うに努める。リスクの監視、コントロールを行う。

落単にならないように、メンバ全員でこのプロジェクト課題に取り組む。学びの成果が得られるように、メンバ全員は積極的に参加する。それでも落単した際は素直に受け止める。リスクの軽減と受容を行う。

PM が一時的または永続的に欠けた際は、PM が復帰するまでサブ PM が速やかに PM となって活動する。そのために、本プロジェクトはサブ PM を設置する。

クライアントは、事故などに遭わないように安全運転などを心がけ、感染症予防のために日本政府が推奨する対策を取るよう要請する。体調管理に気を配り、授業の円滑な遂行を望む。

8. その他

このプロジェクトは PMBOK に従って計画及び実行されるが、調達マネジメントと利害関係者マネジメントはコストマネジメントと同様に今回のプロジェクトは大学の授業でおこなわれる演習であることから、割愛する。

9. 出典

[1] PMBOK® -Project Management Institute
https://www.pmi.org

[2] GanttProject
https://www.ganttproject.biz

[3] ISO/ISE 9126-1:2001
https://www.iso.org/standard/22749.html

[5] JConsole
https://docs.oracle.com/en/java/javase/15/management/using-jconsole.html#GUID-77416B38-7F15-4E35-B3D1-34BFD88350B5

[5] PMD
https://pmd.github.io

[6] Another HTML Lint-Gateway
http://www.htmllint.net/html-lint/htmllint.html

[7] JUnit5

https://junit.org/junit5/

[8] Discord
https://discord.com

[9] Gmail -Google
https://www.google.co.jp/mail/help/intl/ja/about.html?vm=r

[10] Teams -Microsoft
https://www.microsoft.com/ja-jp/microsoft-teams/group-chat-software

[11] GitHub
https://github.com