筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

リポジトリデータを利用した
ソフトウェア開発者評価支援システムの開発
―メトリクスデータ可視化モジュールの開発―

大木 遼太

修士（工学）

（コンピュータサイエンス専攻）

指導教員　田中 二郎

2016年3月

概要

昨今, 企業を代表とした組織において, 複数の開発者によるソフトウェア開発プロジェクトが実施されている. プロジェクト管理者はソフトウェア開発者のパフォーマンス (経験や能力) を把握する必要があり, そのための評価材料を収集しなければならない. しかし現状ではアンケートや個人レポート等, 主観的かつ定性的な評価材料に頼らざるを得ない状況にある.

この問題を解決するため, ソフトウェア開発者の定量的な評価支援を可能とする環境構築を目標としたプロジェクトが株式会社日立製作所 (顧客) および筑波大学によって発足された. 本プロジェクトの開発チームは筑波大学の「高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム」の学生 4 名で構成されている. 筆者を含む開発チームは目標を達成するため, メトリクスデータをグラフとして可視化することによりユーザに提供するシステム MITEMIRU を提案し, 開発した. メトリクスデータとは, ソフトウェア開発者のパフォーマンス指標となるメトリクスを構成する要素である. MITEMIRU は, ソフトウェア開発者が関連するリポジトリから取得したデータを利用してメトリクスデータを生成し, グラフとして可視化することによって定量的な評価材料をユーザに提供する.

筆者は MITEMIRU の実装のうち, メトリクスデータ閲覧機能のメトリクスデータ可視化（グラフ表示）モジュールの実装を担当した. メトリクスデータ可視化モジュールの実装に先立ち, 筆者はメトリクスに関連する先行研究調査を実施し, メトリクスおよびメトリクスデータを顧客とともに定義した. この定義に基づき, 筆者はメトリクスデータの生成に必要となるデータの収集と加工手順, および可視化手法を提案した. 各メトリクスデータに関連するグラフを実装した後, 顧客によるレビューでは様々な意見や改善案を頂いた. この指摘に基づき, 情報専門家である三末和男教授とともに可視化手法を再検討し, 筆者は新たな可視化手法を再提案した後, グラフを実装した.

実装作業終了後に行われたエンドユーザ評価テストでは, 開発者全体の状況を把握しやすい, 新たな知見の発見が期待できる等, メトリクスデータ可視化モジュールに対する意見が得られた. その反面, グラフに関連する説明が不足しているため初めての利用に戸惑ったという意見も挙げられた. より簡便かつ直截的な解釈を支援できる可視化手法の実現, 提供できる情報量の増加等がメトリクスデータ可視化モジュールにおける今後の課題として挙げられる. また対応するリポジトリや可視化するメトリクスデータの拡充等, さらなるシステムの発展が期待される.

目 次

図目次

第1章　はじめに

本報告書は, 筑波大学の「高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム (以下, 高度 IT コースと略す)」の一環である「研究開発プロジェクト」における取り組みを述べたものである. 本プロジェクトは, 顧客である株式会社日立製作所 (以下, 日立と略す) による問題提起により発足したものである. 以下の文章は, 開発構想書 (付録 A) に記載された問題提起文である.

「ソフトウェア開発プロジェクトの多くは複数のソフトウェア開発者による共同作業である. しかしプロジェクト管理者にとって, そのプロジェクトに対するソフトウェア開発者のパフォーマンス (能力や経験) を定量的に測れないという課題がある. プロジェクト管理者の主観的かつ曖昧な評価によって, ソフトウェア開発者のモチベーションの喪失し, ひいてはプロジェクト運営の失敗につながる場合も少なからず存在する. この問題の解決のため, プロジェクト管理者が複数のソフトウェア開発者に対して定量的な根拠による客観的な評価をできるようなソリューションが望まれている.」(開発構想書, p.3)

本プロジェクトではこの問題を解決するため, ソフトウェア開発者のパフォーマンスに関連する定量的な評価材料を提供するシステム MITEMIRU を提案し, 開発をおこなった. 本報告書では筆者の取り組みを中心に, 本プロジェクトの内容を詳細に述べる.

まず第 2 章では, 本プロジェクトの概要を述べる. 次に第 3 章にて, 提案および開発したシステム MITEMIRU の概要を述べる. その後, 第 4 章では筆者が実装を担当したメトリクスデータ可視化モジュールと筆者の取り組みについて詳細に述べる. 第 5 章では要件定義および MITEMIRU の実装における筆者の様々な取り組みについて述べた後, 第 6 章を本報告書の締めくくりとする.

第2章　プロジェクトの概要

本章では, 日立および筑波大学によって発足されたプロジェクトの概要を述べる.

2.1　プロジェクトの方針

第 1 章にて述べた通り, 本プロジェクトは日立による問題提起により発足したものである. プロジェクト発足後, 顧客である日立とミーティングを繰り返し, プロジェクトの方針を決定した. 以下の文章は, 開発構想書に記載した本プロジェクトの方針である.

「株式会社日立製作所により提示された問題を解決するため, 本プロジェクトでは MITEMIRU の提案および開発をおこなう. MITEMIRU は, ソフトウェア開発者が関連するソフトウェア開発プロジェクトに対応する構成管理ツール等のリポジトリから取得したデータを集計・分析し, 加工する. 加工データをグラフとして可視化することにより, ソフトウェア開発者のパフォーマンスに関連する定量的な評価材料として提供し, 日立に提示された問題を解決する.

MITEMIRU の実装にあたり, まずはリポジトリマイニングやプロジェクト管理等に関連する文献調査により, リポジトリ上に存在するデータと対応するソフトウェア開発者のパフォーマンスの関係を明確にする. 次に, この知見に基づき, パフォーマンス評価の判断材料となるメトリクスとその要素なるメトリクスデータ, およびメトリクスデータの表現に必要となるリポジトリ上のデータを検討する. そして, 提供するメトリクスデータと具体的な可視化手法を決定する.

なお本プロジェクトでは, 多数のソフトウェア開発者が共同作業するソフトウェア開発プロジェクトを対象に, 客観的な各ソフトウェア開発者の評価支援を可能とする環境構築を目標とする. 」(開発構想書, pp.3-4)

この方針の通り, 本プロジェクトではソフトウェア開発者の客観的な評価支援を可能とする環境となるシステム MITEMIRU を開発する.

2.2　ステークホルダー

本プロジェクトの主要なステークホルダーを表 2.1 に示す.

三末和男教授は課題担当教員であると同時に, 情報可視化の専門家として MITEMIRU のメトリクスデータ可視化モジュールの設計を支援する. 開発チームは高度 IT コースの学生 4 人で構成されており, 全員が開発者である. 筆者は開発者としての役割に加えて, チームリーダー

表 2.1: ステークホルダー

所属	役割	名前
筑波大学	課題担当教員, アドバイザー	三末 和男
	開発チーム	大木 遼太
		柳沼 工也
		孫 嘉成
		鄒 一民
株式会社 日立製作所	顧客, アドバイザー	居駒 幹夫
		土田 正士
		白井 明
		中村 宇佑
		河野 哲也
		須貝 佳彦

としてミーティングの実施や方針の提案, 顧客との情報共有等を率先して実施した. また日立は顧客であると同時に, アドバイザーとしてプロジェクトの遂行を支援する.

2.3 スケジュール

図 2.1 に, 本プロジェクトのスケジュールを示す. プロジェクト発足後, 顧客や開発チームメンバーとのミーティングを繰り返し, 要件定義をおこなった. その後, 要件定義に基づいてシステムの実装に取り組んだ. 開発手法はアジャイル手法 (反復型開発) を採用し, 顧客のレビューにより問題点を明確にしながら改善および実装を進めるものとする. また第 4 スプリントでは開発作業と並行してエンドユーザ評価テストを実施し, システムの有用性を調査した.

2.4 プロジェクト運営

2.4.1 ミーティング

本プロジェクトでは開発チームメンバーや顧客と情報を共有するため, 定期的にミーティングを実施した. 開発チーム内のミーティングの目的は, 主にプロジェクトの方針決定や実装計画である. 筆者はチームリーダーとしてミーティングを主催し, ファシリテーターとしてプロジェクト方針の提案や決定等を積極的に進めた. 基本的には特定の場所に全員集合し直接会話することでミーティングを進めたが, 就職活動等により集合できない場合は, 遠隔地からツールを利用しミーティングを実施した.

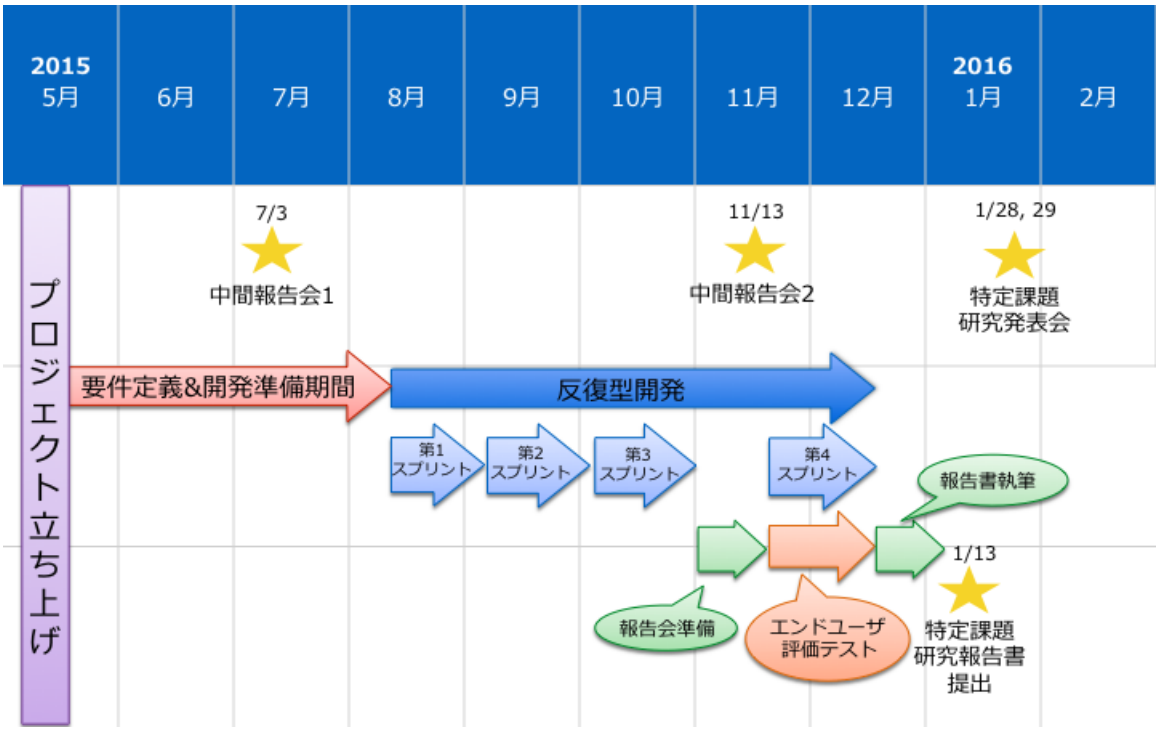

2015
5月
6月
7月
8月
9月
10月
11月
12月
2016
1月
2月
プロジェクト立ち上げ
7/3
中間報告会1
11/13
中間報告会2
1/28, 29
特定課題
研究発表会
要件定義&開発準備期間
反復型開発
第1
スプリント
第2
スプリント
第3
スプリント
第4
スプリント
報告書執筆
1/13
報告会準備
エンドユーザ
評価テスト
特定課題
研究報告書
提出

図 2.1: スケジュール

また顧客と開発チーム内の認識の差異を埋めるため, 遠隔コミュニケーションツールを利用して約 1 週間ごとにミーティングを実施した. 顧客とのミーティング内容は, 主に検討事項の通知や意見交換等である. 要承認事項については, このミーティングにて承諾をいただき, 決定事項とした. なお筆者は開発チーム内のミーティングと同様に, 主にファシリテータとしてミーティングに参加した.

2.4.2 利用ツール

プロジェクトを円滑に遂行するため, 本プロジェクトでは外部ツールを積極的に利用した. 導入したツールを以下に示す.

- **WebEx**
  顧客との遠隔コミュニケーションを図るため, テレビ会議システムである WebEx を日立より提供していただき, 利用した. WebEx は複数のユーザとパーソナルコンピュータの画面共有が可能であるため, 発表資料や成果物を提示しながら情報を共有できた.

- **Dropbox**
  開発構想書やシステム概要図等, プロジェクトの進行に応じて生成されるドキュメントは全て Dropbox にて管理した. Dropbox は任意のメンバーとのドキュメント共有が可能であるため, 開発チーム内に限らず顧客とのドキュメント共有を簡便に実現できた.

- **Slack**
  Slack は, テキストチャット型のコミュニケーションツールである. チャンネル (特定のメンバーを集めたグループ) を自由に作成でき, なおかつメッセージがリアルタイムに反映されるため, チーム内や顧客との迅速なコミュニケーションを簡便に実現できた.

- **Hangout**
  Hangout は, ビデオチャット型のコミュニケーションツールである. 本プロジェクトでは, 就職活動の影響により直接集合してミーティングを実施することは困難であり, 遠隔地でのミーティングが必須であった. しかし, 遠隔地でのコミュニケーションは Slack の利用により実行できたものの, ミーティングのように情報量の多い対話形式のコミュニケーションは煩雑である問題が発生していた. そこで, 対話形式のコミュニケーションに適している Hangout を導入することで, 前述の問題を解消した.

- **GitHub**
  ソースコードを一括管理でき, なおかつ何時でも参照できるようにするため, バージョン管理ツールとして GitHub を導入した. GitHub はインターネット上にリポジトリを設けるため, アクセス環境に依存することなくソースコードを参照できる. またソースコードの改編状況は逐次監視できるため, 必要に応じて迅速なコミュニケーションを実現できた.

- **ZenHub**
  ZenHub は, GitHub のプラグインとして提供されているチケット管理ツールである. 本プロジェクトはアジャイル手法を用いた開発のため, タスクは全てチケットとして管理した. ZenHub はかんばん方式によるチケット管理を実現し, チケットの進行状況をリアルタイムに監視できた. また必要に応じてコメントを書き込むことができるため, チケット単位で情報のやりとりを実現できた.

- **Redmine**
  Redmine は, オープンソースのプロジェクト管理ツールである. チケット管理を目的として導入したが, ZenHub の利用によりチケットの管理は実現できていた. しかし ZenHub は GitHub のアカウントが必要となるため, 顧客とチケットの状況を共有するためには負担がかかってしまう. Redmine は簡便にアカウントの生成が可能であるため, 顧客によるチケットの監視を目的として利用した.

2.5 実装の方針

2.5.1 役割担当

実装に必要となる知識の学習を最低限に抑えて開発効率を向上させるため, 本プロジェクトでは実装する機能に応じて役割分担を行った. 筆者はメトリクスデータ可視化モジュールの実装を担当し, グラフ描画処理に関連する知識の学習に努めた. ただし第 1 スプリントでは機能の実装を実施しなかったため, 横断的な作業分担をおこなった.

2.5.2 開発手法

本プロジェクトには, 開発に関して限られた要素が存在する. その主たるものが, 人員を代表とする資源である. 本プロジェクトの開発チームメンバーは発足時に集められた高度 IT コースの学生で構成されており, カリキュラムの性質上, プロジェクト開始後において開発チームメンバーの変更はほぼ認められない. 開発チームメンバーの変更が不可能である以上開発者の能力もある程度限られるため, 柔軟で横断的な役割分担求められる.

また時間についても, 大きな制限が存在する. 2016 年 1 月 13 日 (金) までには特定課題研究報告書 (本報告書) の提出が求められているため, 報告関連の行事を除いた全ての作業を前述の期日までに完了させなければならない. 学習時間を含め, 実装に費やすことができる時間は約 4ヶ月と短期間である.

以上のことから, 本プロジェクトでは自ずとスコープも制限される. そこで本プロジェクトではアジャイル開発 (反復型開発) を適用し, 顧客とのコミュニケーションを通じてスコープを決定しながらプロジェクトを遂行した.

本プロジェクトの開発手順は, アジャイル開発手法の代表格であるスクラムのプラクティスを参考とした [1]. 筆者はチームリーダーとしてスクラムの学習を進めながらチームメンバーと知識を共有し, 必要に応じてプラクティスの実行に務めた.

実装作業は約 1ヶ月を一巡としたスプリントの繰り返しである. スプリントのはじめにチケットを発行し, 開発チームは期間内にチケットを順次消化する. 実装期間終了後, KPT 分析によってスプリントを振り返り, 最後に顧客とのミーティングにて成果報告および指摘事項の共有をおこなった.

2.5.3 チケットの発行

各スプリントでは, はじめに顧客との話し合いによって定義された要件を抽出し, チケット (タスク) として定義する. チケットの粒度はモジュールを基準とし, 各タスクの重みに大きな差がでないように心がけた.

その後, 開発チームメンバー全員でプランニングポーカーを実施し, 各チケットにポイントを付ける. あらかじめ特定のチケットを 1 つ選択してポイントを設定し, そのチケットの重みを基準として他のチケットにポイントを相対的に見積もって設定した. 全てのチケットの見積もりが終了した後, ポイントの集計をおこなう. 発行されるチケットの合計ポイントは, 開発チームのベロシティと作業見積もり時間から予測されるキャパシティ(見積もり消化ポイント) の範囲に納めなければならない. そのため, キャパシティを上回る場合には優先度の高いチケットを選択し, キャパシティの範囲に収まるようにする. その後, チケット管理ツールである Redmine および ZenHub にそれぞれチケットを発行した.

なお第 1 スプリントでは参考となるベロシティを計測できないため, キャパシティを算出できない. そこで例外として第 1 スプリントでは消化したいバックログに対応するチケットは全て発行し, 優先度の高いものから順次消化するものとした. 第 2 スプリント以降では, 前スプリントで計測されたベロシティに基づいてキャパシティを算出し, それに応じてチケットを発行するものとする.

2.5.4 進捗管理

日々の進捗管理手法は, かんばん方式を採用した. かんばんツールは GitHub のプラグインとして提供されている ZenHub を利用し, 発行したチケットの作業状況が常に監視できる環境を実現した (図 2.2).

以下に, チケットの消化手順を記載する.

1. **チケットの着手**

   要消化のチケットは, 主に”ToDo”に置かれている. 各開発者はここから着手するチケットを選択し, 以下の項目をチケットに記載する.

   - 担当者名

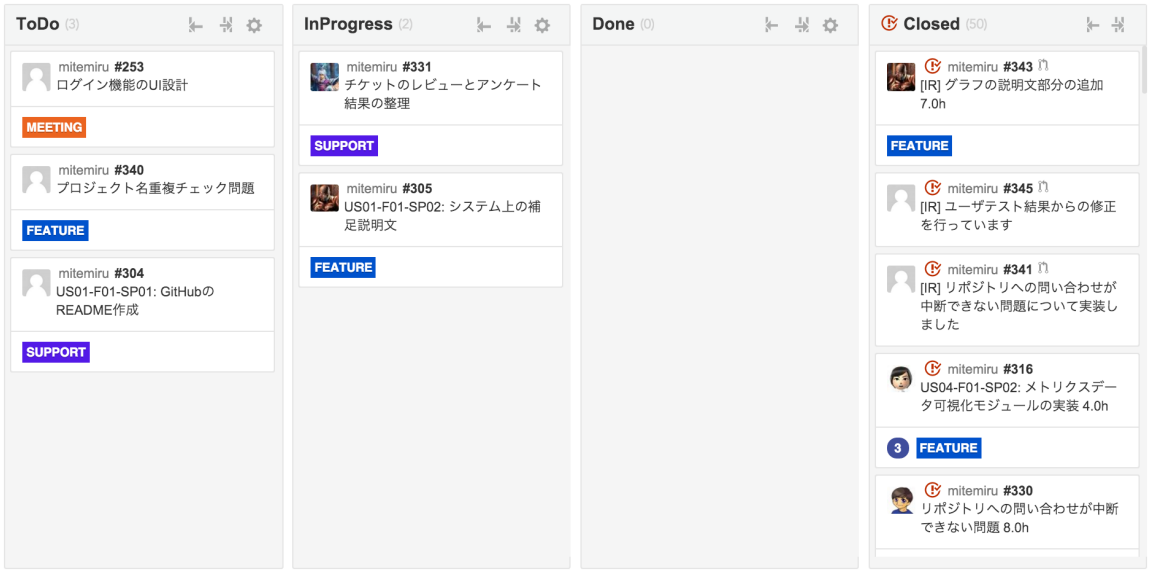


図 2.2: ZenHub を利用したかんばん

- 作業内容の詳細
- チケット終了の定義

以上の項目を入力した後, チケットを”InProgress”に移動させる.

2. **進行状況の入力**
   チケット担当者は進行状況に応じて, 作業時間やコメント等を適宜入力する. 開発チームメンバーはこの入力情報を確認し, 各チケット担当者の進捗状況を把握する.

3. **チケットの終了**
   チケット担当者が必要となる作業を終了した後, チケットを”Done”に移動させる. チームメンバーは”Done”に置かれたチケットの内容を確認し, チケット終了の定義内容を満たしているか評価する. 終了の定義を満たしていないと判断された場合, チケット担当者は再度チケットを”InProgress”へ移動させ, 指摘事項の修正等をおこなった後, 再び”Done”へと移動させる. 2 名以上の評価によってチケットの終了が承認された場合, 評価者によってチケットは”Close”へと移動させられる.

ZenHub にはチケットに記載されたポイント情報を利用してバーンダウンチャートを自動生成する機能が備わっており, 第 2 スプリントまではこれを利用して進捗状況を確認した (図 2.3). しかし第 2 スプリントの振り返りにおいて取り上げられた問題のため, 第 3 スプリント以降は ZenHub によって自動生成されるバーンダウンチャートの利用を廃止し, Excel シートを利用したバーンダウンチャートに変更した (詳細は第 5.3.2 節にて述べる).

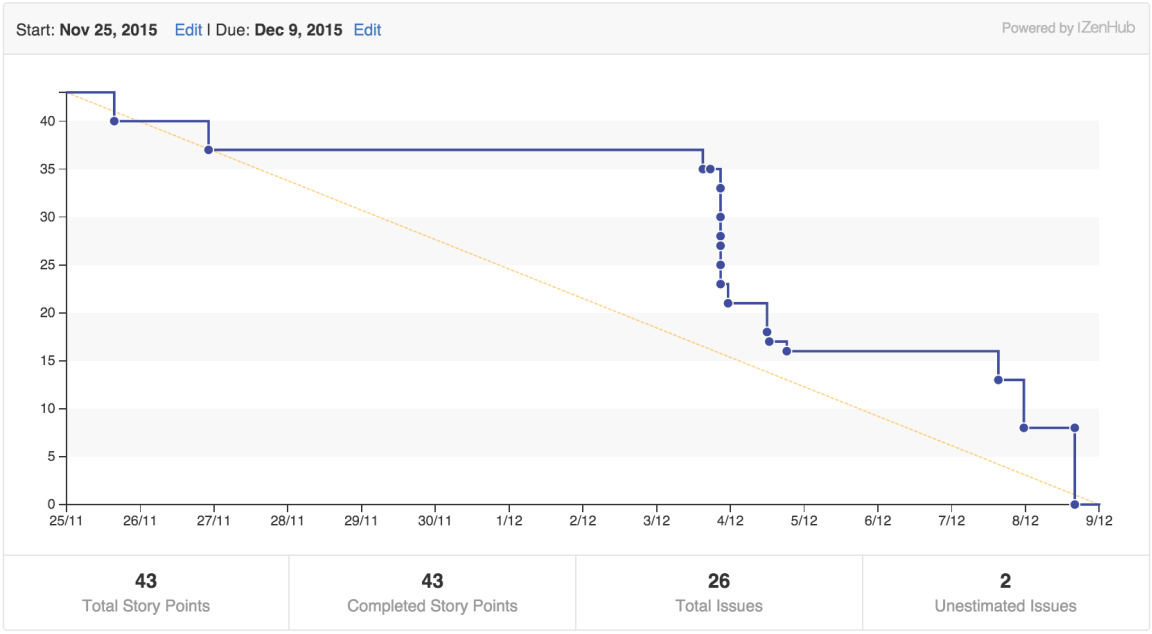


図 2.3: ZenHub により生成されるバーンダウンチャート

第3章　MITEMIRU

本章では, 本プロジェクトにて提案および開発したシステム MITEMIRU の概要を述べる.

3.1　システムのアプローチ

MITEMIRU は, ソフトウェア開発者 (以下, 評価対象者) のパフォーマンスの定量的な評価を支援するシステムである. MITEMIRU は評価対象者のパフォーマンスに関連する定量的な評価材料をリポジトリ上のデータから生成し, 提供する. ユーザは, はじめに評価対象者が所属するプロジェクトに関連するリポジトリの認証情報を MITEMIRU に登録する. 次に, MITEMIRU はリポジトリから必要な情報を取得し, 集計・分析によってメトリクスデータを生成する. メトリクスとは評価対象者のパフォーマンスの指標となる情報であり, メトリクスデータはメトリクスの要素となる情報である. メトリクスデータを生成した後, MITEMIRU はこれをグラフとして可視化する. ユーザはこのグラフからメトリクスデータを読み取ることにより, 評価対象者のパフォーマンスの定量的な評価材料として利用できる. なお主観的な結果による誤解を避けるため, システム上では評価対象者のパフォーマンスに対する評価結果を下さず, あくまでその評価材料を提供するものとする.

3.2　システムの利用対象者

本システムのユーザは, 10 名程度のソフトウェア開発プロジェクトの管理者を利用対象者とする. 特に, 日立のような企業におけるソフトウェア開発プロジェクトの管理者を主な利用対象者とする.

3.3　システムの機能

図 3.1 に MITEMIRU のシステム構成図を示す.
MITEMIRU の主要となる機能は, 以下に示す 4 つの機能に大きく分けられる.

1. **ログイン・ログアウト機能**
   ある環境にインストールされた MITEMIRU は, 複数人のユーザによって利用されるケースが想定される. 本機能は, それぞれのシステムユーザを一意に識別するために実装される機能である. システムユーザは状況に応じて自由に新規登録でき, ログイン時

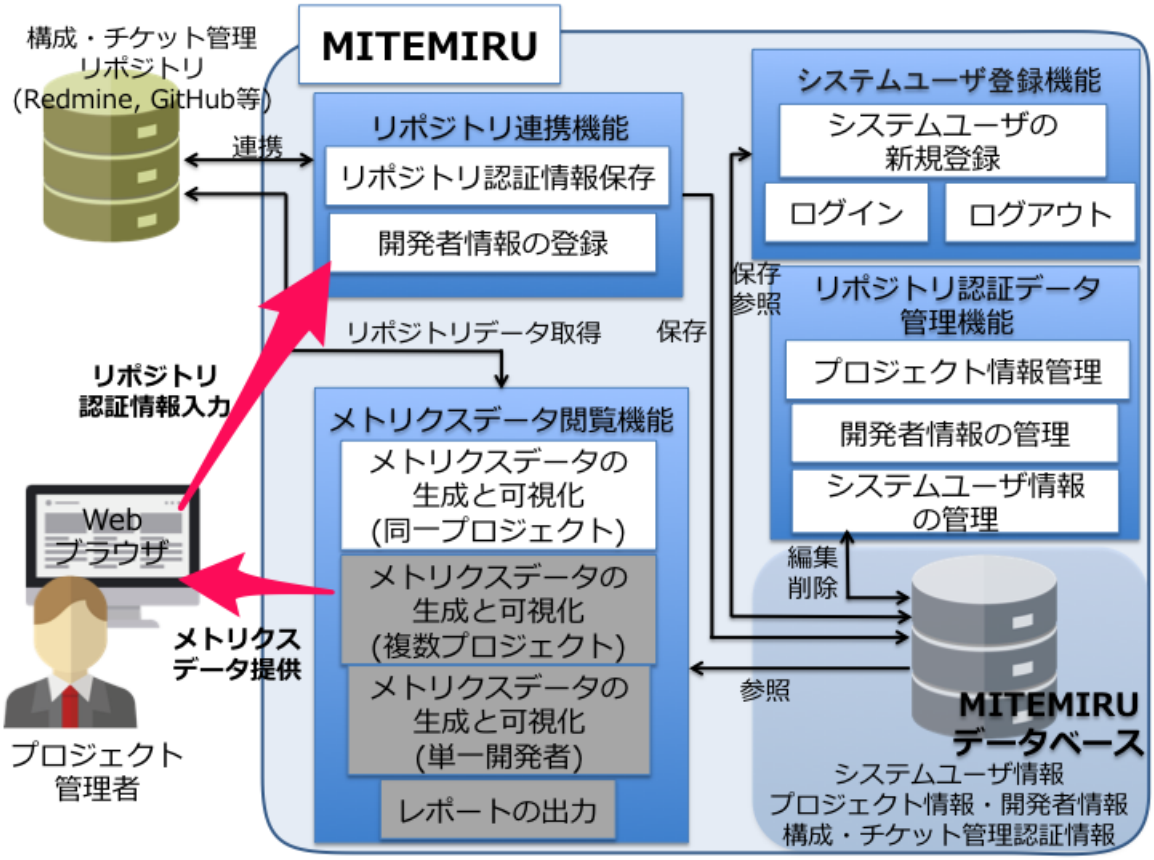

構成・チケット管理
リポジトリ
(Redmine, GitHub等)
連携
MITEMIRU
リポジトリ連携機能
リポジトリ認証情報保存
開発者情報の登録
システムユーザ登録機能
システムユーザの
新規登録
ログイン
ログアウト
保存
参照
リポジトリ認証データ
管理機能
プロジェクト情報管理
開発者情報の管理
システムユーザ情報
の管理
リポジトリデータ取得
保存
リポジトリ
認証情報入力
メトリクスデータ閲覧機能
メトリクスデータの
生成と可視化
(同一プロジェクト)
メトリクスデータの
生成と可視化
(複数プロジェクト)
メトリクスデータの
生成と可視化
(単一開発者)
レポートの出力
Web
ブラウザ
メトリクス
データ提供
プロジェクト
管理者
編集
削除
参照
MITEMIRU
データベース
システムユーザ情報
プロジェクト情報・開発者情報
構成・チケット管理認証情報

図 3.1: システム構成図

のユーザ情報に限り編集および削除できる. また前述のユーザとは別に, システムユーザを管理するスーパユーザアカウントをインストール時に生成する.

2. **リポジトリ連携機能**

   本機能は, 評価対象者が関連するリポジトリへのアクセスに必要となる認証情報の煩雑な反復入力を削減するため, リポジトリ認証情報を MITEMIRU 上のデータベースに登録する機能である. ユーザは, バージョン管理ツールやチケット管理ツール等のリポジトリへアクセスするために必要となる認証情報を, 任意のプロジェクト名とともに登録できる. なおユーザであるプロジェクト管理者は, 評価対象者が過去に利用していたリポジトリへのアクセス権限 (認証情報) を保有していることを前提とし, MITEMIRU に登録する認証情報はそのユーザのものとする. またユーザが登録した認証情報は, ログイン時のユーザ情報に対応づけられるため, 複数の利用者が存在する場合でも, それぞれの利用者が登録した認証情報に限り利用できるものとして実装する. 本プロジェクトではシステムの評価環境として, 本プロジェクトのリポジトリを利用する. そこで本プロジェクトにて利用しているバージョン管理ツールの GitHub, チケット管理ツールの Redmine へのアクセスを実現し, 取得したデータを利用した可視化機能を実装することとした.

3. **メトリクスデータ閲覧機能**

   本機能は, 評価対象者のパフォーマンスの定量的な評価材料となるメトリクスの要素となるメトリクスデータの可視化をおこなう機能である. 可視化項目であるメトリクスデータは, システム上で定義されている. MITEMIRU はメトリクスの生成に必要となるリポジトリ上のデータを取得し, 集計・分析等によってメトリクスデータを生成する. 生成されたメトリクスデータはグラフとして可視化され, ユーザに情報を提供される. 利用者はグラフを通じてメトリクスデータを閲覧することにより, 定量的な評価材料として利用できる.

   対象となる開発者およびプロジェクトは, 大きく 3 つの形式で比較できる. 1 つ目は, 同一プロジェクト内の比較である. 特定のプロジェクト内のソフトウェア開発者の取り組みを比較することで, プロジェクト全体における評価対象者の役割や作業効率等を把握できる. 2 つ目は, 複数プロジェクトの比較である. 複数の評価対象者を比較したい場合, 必ずしも全員が同一のプロジェクトに所属しているとは限らない. それぞれの評価対象者が所属していたプロジェクトから, 同一のメトリクスデータを取得し可視化することで, 複数の評価対象者を比較する. 3 つ目は, ある 1 人の評価対象者に着目した比較である. ある評価対象者が過去に所属していたプロジェクト情報を時系列に従って表示し, 役割や作業効率等の違いを比較する.

   また, これらの可視化結果を様々な場面で参照するため, レポートとして出力する機能を備えている.

4. **リポジトリ認証データ管理機能**

   本機能は, 新規プロジェクト登録機能によってデータベースに保存されたデータを管理する機能である. それぞれのデータに対して, 閲覧, 編集および削除を実行できる. 加え

て登録済みプロジェクト情報に対し, 未登録のリポジトリの認証情報を追加登録できる.

本プロジェクトにて, 筆者はメトリクスデータ閲覧機能のメトリクスデータ可視化モジュールの実装を担当した. メトリクスデータ可視化モジュールは, グラフの描画を実行するモジュールである. システム構成図 (図 3.1) に示す通り, メトリクスデータ閲覧機能には同一プロジェクト内, 複数プロジェクト, 単一開発者の比較をそれぞれ実現するが, 本プロジェクトでは限定されたスコープのため, 同一プロジェクト内の比較に絞り実装することとした. またレポートの出力機能も優先度の観点から, 本プロジェクトのスコープ外とした.

3.4 システムの特徴

MITEMIRU は, 大きく 2 つの特徴を備えている. 1 つ目は, リポジトリ上のデータを利用した評価材料の生成である. プロジェクト管理者によるソフトウェア開発者の評価では, 情報の収集源としてアンケートや面談等における質問の回答が評価材料として用いられるケースが多々挙げられる. しかし, 設問内容によっては定性的かつ曖昧な情報を引き出してしまい, 誤った判断の原因につながる可能性もある. MITEMIRU は評価対象者の活動に応じて生成されたリポジトリ上のデータを利用することにより, 定量的な評価材料の提供を実現できる.

2 つ目は, 簡便な情報収集である. 前述のアンケートや面談等の準備はプロジェクト管理者が行わなければならず, 大きな負担となっている. MITEMIRU はあらかじめ定義したメトリクスデータに基づいてリポジトリからデータを自動で収集するため, 煩雑な準備作業を大きく削減できる. またグラフとしてメトリクスデータを可視化し提供することによって, プロジェクト管理者による簡便かつ直截的な情報収集を実現する.

3.5 システムの提供形式

MITEMIRU は, 評価対象者が関連するリポジトリへアクセスできる環境にインストールされなければならない. GitHub のように外部サーバ上にリポジトリが存在し, インターネットを経由してアクセスする場合はイントール環境の違いは大きな問題とはならない. しかし, 企業をはじめとする組織では外部ネットワークから隔離された環境にリポジトリが設置されるている状況が想定されるため, MITEMIRU はその環境内部にインストールされなければならない. 組織による様々なリポジトリ運営環境に対して柔軟に対応できるようにするため, MITEMIRU はオンプレミス型の運用形態として実装する. また同一組織内の複数関係者による利用も想定して実装する.

第4章　メトリクスデータ可視化モジュールの開発

本章では, 筆者が主に担当したメトリクスデータ可視化モジュールの開発について詳細に述べる.

4.1　メトリクスデータの調査と定義

メトリクスデータ可視化モジュールの実装にあたり, 筆者を含む開発チームメンバーはソフトウェア開発者評価に関連する既存研究を調査した. その結果, 以下に示す情報が得られた.

- **風林火山モデルとメトリクスの関連**
  風林火山モデルは小野和俊によって提案された, ソフトウェア開発者に求められる能力を 4 つに分類したモデルである [2]. 五田篤志らはこの風林火山モデルと関連するメトリクスを定義し, 関連付けることによってソフトウェア開発者の特性分析を実施した [3][4]. ただし, メトリクスの定義は著者による主観によって対応付けられている点に注意し, あくまで参考とするまでに止めた.

- **OSS 開発者の貢献**
  坂口英司らは, GitHub 上にておこなわれているオープンソフトウェアプロジェクトに関連する開発者の貢献を明らかにするため, GitHub に記録されているソフトウェア開発者のイベント情報をメトリクスとして収集した [5]. 「記録されている活動には従来からコミッターを選出するために参考にされている技術的活動 (パッチ投稿, レビュー), 社会的活動 (コメント投稿) などが含まれており, 開発者の活動を評価するための指標としては妥当である.」(p.2) と述べられているように, GitHub の記録情報は開発者の活動を示すメトリクスとして有用であると考えられる.

これらの調査結果から, ソフトウェア開発者のパフォーマンスを示すメトリクスはいくつか提唱されているものの, 確立されていないことが判明した. そこで本プロジェクトでは文献調査よって得られた情報を参考とし, 顧客とのミーティングを通じてメトリクスを定義した. 図 4.1 は, 筆者がまとめたメトリクスリストにおける定義例である.

本プロジェクトではこれらのメトリクスに基づき, ソフトウェア開発者の評価材料となるメトリクスデータを検討した.

計測項目 （主語は開発者）	概要	利用ツール	利用データ	計測方法（暫定）	参考文献	備考
生産性	各開発者の作業が予定通りに進んでいたか分析する. 予定よりも早く終わるほど生産性（効率性）が良く, 遅ければ悪いと判定できる. なお, 見積もりは適切であることを前提とする.	プロジェクト管理ツール （Redmine, JIRA, Trac等）	・チケット情報 － 担当者 － 予定工数 － 実績工数	各開発者が担当するチケットを収集し, 予定工数と実績工数のズレを計測する.	[5][9][17][19][23][26]	
バグの発見数・解消数	各開発者が発見および修正したバグの数を計測する. この数が多いほど, ソフトウェアの品質向上に貢献していると判定できる.	・プロジェクト管理ツール	・チケット情報 － 担当者 － トラッカー	バグに関するトラッカーを持つチケットの生成数を計測する.	[5][13][15][23][26]	
バグの生成数	各開発者が生成したバグの数を計測する. この数値が多いほど, ソフトウェアの品質低下を招いたと判定できる.	・プロジェクト管理ツール ・構成管理ツール	・コミットログ ・チケット情報 － 担当者 － トラッカー	バグに関するトラッカーを持つチケットとコミットログを結び付けてバグを生成した開発者を特定し, その数を計測する.	[12][17][19][21]	
主体性	各開発者が自発的に行動した履歴を定量的に計測する.	・プロジェクト管理ツール ・構成管理ツール	・コミットログ ・チケット情報 － 担当者	コミットログやチケット等の更新頻度を計測する.	[1][11]	
コーディング量	各開発者が記述ないし変更したコード行数を計測する.	・構成管理ツール	・コミットログ	コミットログから追加, 削除行数を計測する.	[5][11][12][13][18][20][21][22][23][25][26]	
専門性	各開発者が利用可能な言語の種類や, その言語を使用したコーディング実績を定量的に計測する.	・構成管理ツール	・コミットログ	ファイルの変更履歴を拡張子ごとに収集し, 対応する言語を特定する. また, それらの変更行数を算出する. 例: foo.c, bar.java	[1]	
コーディング規約の遵守度	各開発者が記述したコードが規約に則っているか判定する.	・構成管理ツール	・コミットログ	開発者が記述したソースコードをコーディング規約検査機によって測定し, 規約違反箇所数を計測する.	[1][21][28]	
ソフトウェア部品の再利用量 （クローン等の不適切な利用を含む）	各開発者が記述したコードのうち, 再利用しているソフトウェア部品量を計測する.	・構成管理ツール	・ソースコード ・コミットログ	ソースコードを静的解析し, 重複するコード記述量を算出する.	[15][29]	

図 4.1: メトリクスの定義例

4.2 可視化するメトリクスデータの決定

顧客とのミーティングを通じて定義されたメトリクスのうち, 実現可能性を考慮しながらメトリクスデータを検討した. その結果, ある 4 つのメトリクスに関連するメトリクスデータを提案し, グラフとして可視化することとした. 可視化するメトリクスデータと関連するメトリクス (括弧内) の一覧を以下に示す.

1. **コミット数 (主体性)**
   主体性は, ソフトウェア開発者が主体的に行動していたかを示すメトリクスである. コミットはソフトウェア開発者主体的な行動によって実行されるものであるため, コミット数を主体性を示すメトリクスデータとして定義した. データ収集源となるリポジトリは, Git や Subversion 等のバージョン管理ツールである.

2. **チケット消化数（役割担当）**
   役割担当は, ソフトウェア開発者が特定のプロジェクトにおいて担当した役割 (作業の種類) を示すメトリクスである. 分類ごとにチケットの消化数を計測することで, ソフトウェア開発者が行った作業内容 (ドキュメント作成, テスト等) の傾向を導き出せる. データ収集源となるリポジトリは, Redmine や Trac 等のチケット管理ツールである.

3. **作業時間 (作業効率)**
   作業効率は, ソフトウェア開発者のタスク消化効率を示すメトリクスである. 見積もり作業時間に対して実績作業時間が短いほど, 作業効率が高いと言える. データ収集源となるリポジトリは, Redmine や Trac 等のチケット管理ツールである.

4. **コメント数 (発言力)**
   発言力は, ソフトウェア開発者が自発的に発言しているかを示すメトリクスである. コ

メント数は自発的な発言に関連するという観点から, 発言力を示すメトリクスデータとして定義した. データ収集源は, GitHub のようなコミュニケーションツールを併せ持つリポジトリとする.

4.3 メトリクスデータ可視化モジュールの設計と実装

実装に先立ち, メトリクスデータの生成手順と可視化手法を明確にする必要がある. そこで, 筆者はメトリクスデータ生成モジュールおよびメトリクスデータ可視化モジュールの詳細仕様を設計した. 開発チームメンバーのレビューを受けながら筆者は修正を繰り返した後, 詳細仕様を決定した. 4 つのメトリクスデータの生成手順および可視化手法を以下に示す.

1. **コミット数 (図 4.2)**
   特定のプロジェクトに所属するソフトウェア開発者のうち, 評価したいソフトウェア開発者のコミット数とその他のソフトウェア開発者のコミット数を棒グラフとして可視化する. なお, その他のソフトウェア開発者のコミット数はすべて一括して集計される. これにより, プロジェクト全体のコミットのうち, 評価対象者がコミットした割合を直截的に解釈できる. 取得データは GitHub(Git) リポジトリのコミットログであり, 開発者ごとにコミットログの数を集計する.

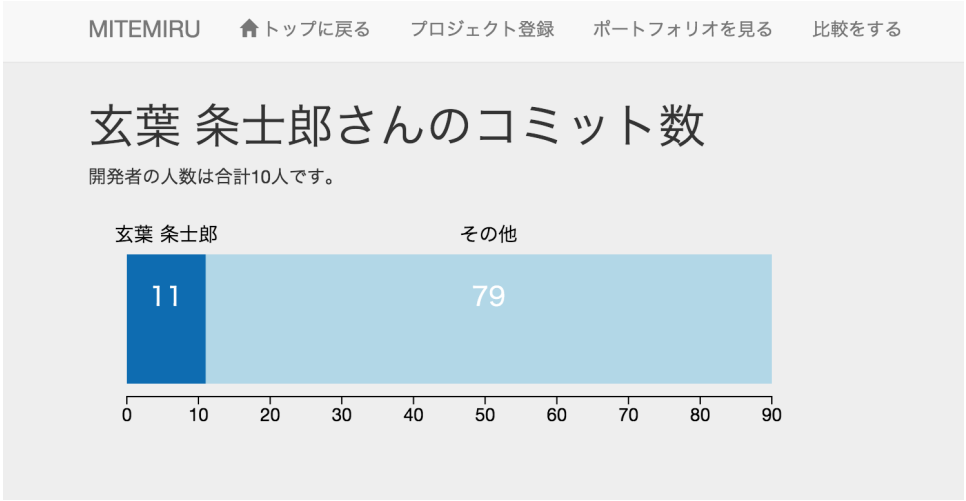


図 4.2: コミット数のグラフ

2. **チケット消化数 (図 4.3)**
   ソフトウェア開発者が担当したチケットをトラッカーごとに分類し, その割合を円グラフとして可視化する. テストやデバッグ担当者であれば消化するチケットのトラッカーはテストやバグというように, 担当に関連した名目が多く現れると考えられるため, 役割

の傾向を導くための参考となる. 取得データは Redmine のチケットデータであり, それぞれのチケットに設定されたトラッカーを集計する.

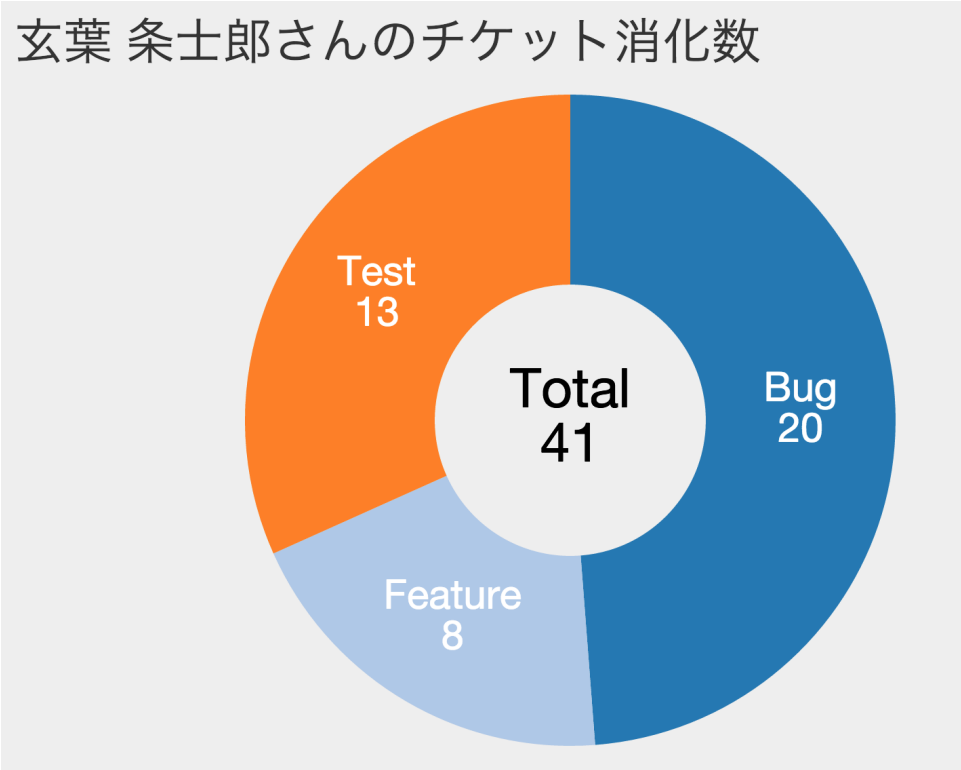


図 4.3: チケット消化数のグラフ

3. **作業時間 (図 4.4)**
   ソフトウェア開発者が担当したチケットをトラッカーごとに分類し, 予定工数 (見積もり作業時間) と実績工数 (実績作業時間) をそれぞれ棒グラフとして可視化する. これにより, ソフトウェア開発者が見積もりに対する実績を参照でき, 作業内容ごとに効率を導き出せる. 取得データは Redmine のチケットデータであり, それぞれのチケットに設定されたトラッカーごとに予定工数と実績工数を集計する.

4. **コメント数 (図 4.5)**
   特定のプロジェクトにおいて, ソフトウェア開発者が他者に送信したコメント数をそれぞれ表示する. コメント数に応じてリンクの太さを変えることで, コメント数の直截的な解釈を支援する. 取得データは GitHub の Issue であり, それぞれの Issue におけるコメントの数を開発者ごとに集計する.

以上のうち, 筆者はコメント数を除くメトリクスデータ可視化モジュールを第 2 スプリントにて実装した. コメント数に関連する作業については, 作業時のタスク量に偏りが発生したた

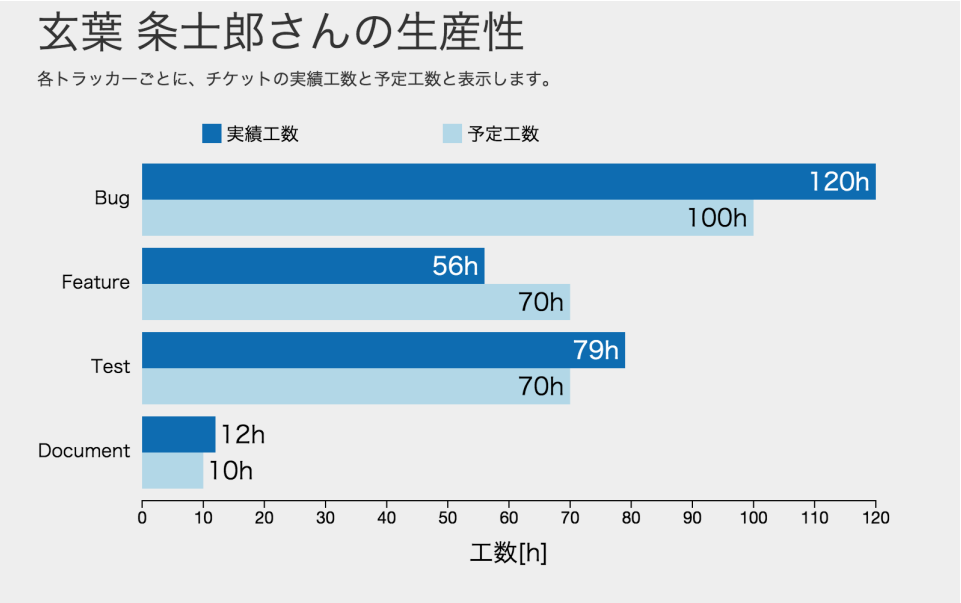


図 4.4: 作業時間のグラフ

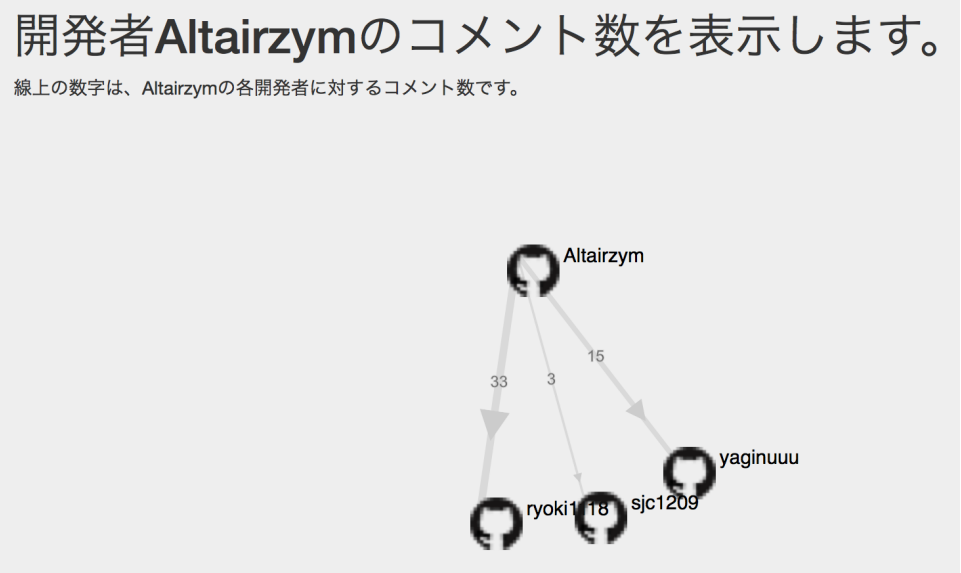


図 4.5: コメント数のグラフ

め, メトリクスデータ生成モジュール実装の担当者である鄒が詳細仕様検討およびメトリクスデータ可視化モジュールの実装を担当した.

メトリクスデータ生成モジュールとメトリクスデータ可視化モジュールの実装は同時並行で行われる. したがって詳細仕様の設計時に決定した生成データの形式に応じてダミーデータを作成し, それを用いてグラフ描画処理の確認をおこなった. その後, メトリクスデータ生成モジュールの実装が終了した時点でダミーデータを実データに切り替え, グラフ描画処理の確認をおこなった.

グラフの描画処理は JavaScript ライブラリである D3.js を用いて実装した. 筆者は JavaScript および D3.js の利用経験がなかったため, 学習を進めながら実装をおこなった. そのため当初は見積もりよりも作業時間が大きく遅れてしまったが, 学習が進むにつれて見積もりよりも早い段階で実装を終えられるようになった.

4.4 メトリクスデータ可視化モジュールの再設計と修正

4.3 節にて述べた 4 つのグラフを実装した後, 顧客によるレビューを行った結果グラフの仕様に関して以下のコメント頂いた.

- コミット数およびコメント数のグラフでは対象者を一人に縛るのではなく, プロジェクトに所属する開発者同士を明示して比較できるようにしてほしい
- 作業時間のグラフでは見積もり工数と実績工数をそれぞれ単に表示するのではなく, 作業効率を示す数値を明示してほしい

以上のコメントに基づき, メトリクスデータ可視化モジュールにおける可視化手法の再設計を実施した. 4.3 節にて述べたグラフの実装では, 筆者が提案したグラフ案を開発チーム内で確認および議論を行った後, 実装した. またメトリクスデータ可視化モジュールの修正では, 情報をより簡潔かつ適切に表現できるグラフを実現するため, 情報可視化の専門家である三末和男教授を交えた議論を筆者は提案し, 実施した. この議論に基づき, 筆者はそれぞれのグラフの修正案を提案し, 決定した. なおこの提案では, 可視化手法の修正に伴ったメトリクスデータ生成モジュールにおける処理の変更も考慮している.

決定したグラフの修正案に基づき, 筆者は各グラフの可視化モジュールの修正を行った. 修正の手順は, 4.3 節にて述べた実装と同様である. なおコメント数のグラフを含め, メトリクスデータ可視化モジュールの修正案の実装はすべて筆者が行った. 筆者が設計および実装した 3 つのグラフを以下に示す.

1. **コミット数 (図 4.6)**

   修正後のグラフでは, 特定のプロジェクトに存在するすべてのソフトウェア開発者ごとにコミット数を表示する. またコミット数に応じたソートを可能とし, 評価対象者と他者の活動の違いを詳細かつ直截的に解釈できるようにした. 一方コミット数の平均値を表示することにより, 大まかではあるがプロジェクト全体における状況も簡潔に解釈できる.

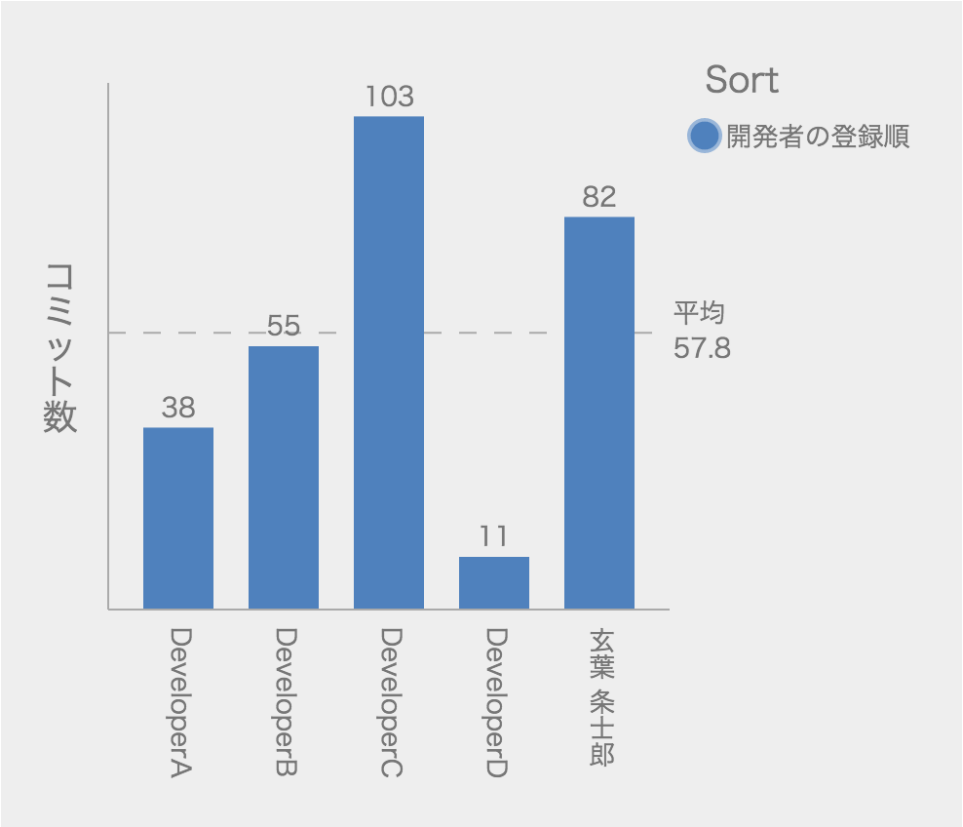


図 4.6: 修正後のコミット数のグラフ

2. **チケット消化数と作業時間 (図 4.7, 図 4.8, 図 4.9, 図 4.10)**

修正後のグラフでは, チケット消化数と作業時間の両方を組み合わせた. 図 4.7 に示すグラフは, 特定のプロジェクトにおける全開発者の状況を一覧できるものである. 利用者はこの中から評価対象者を選択し, 図 4.8 のグラフへと移行する. ここに表示される円グラフの色はトラッカーを示しており, 予定工数に応じて各トラッカーの割合が決定される. 円グラフは 2 層に分かれており, 色の濃い部分は半径が可変する. 色の濃い部分の半径はソフトウェア開発者の作業効率 (予定工数/実績工数) を示しており, 予定工数に対して実績工数が小さいほど半径が大きくなる. つまり, 半径が大きいほど作業効率が高いことを示す. またそれぞれの扇にマウスを乗せると, 数値情報が表示される (図 4.9) さらに扇をクリックすると, 対応するトラッカーごとに作業効率の棒グラフが表示される (図 4.10). これにより, 各開発者との作業効率の比較を実現する. それぞれの棒をクリックすると, 今度は選択した開発者の円グラフ (図 4.8) へと遷移する. このように, チケット消化数と作業時間のグラフでは探索的な情報収集を実現するため, クリックによって簡便に移動できるように筆者は設計および実装した.

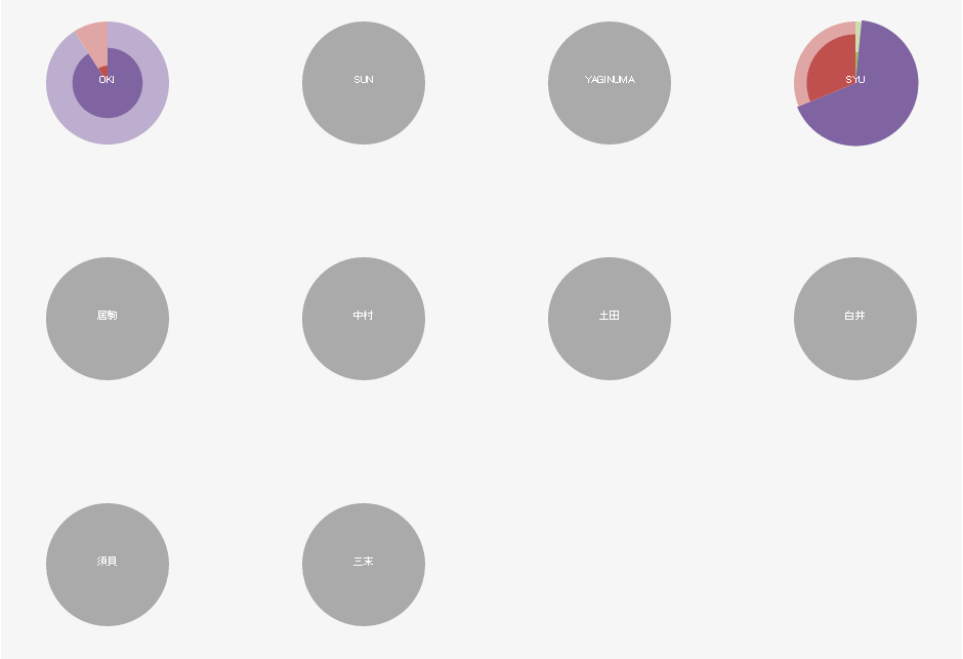


図 4.7: 修正後のチケット消化数と作業時間のグラフ (円グラフによる一覧表示)

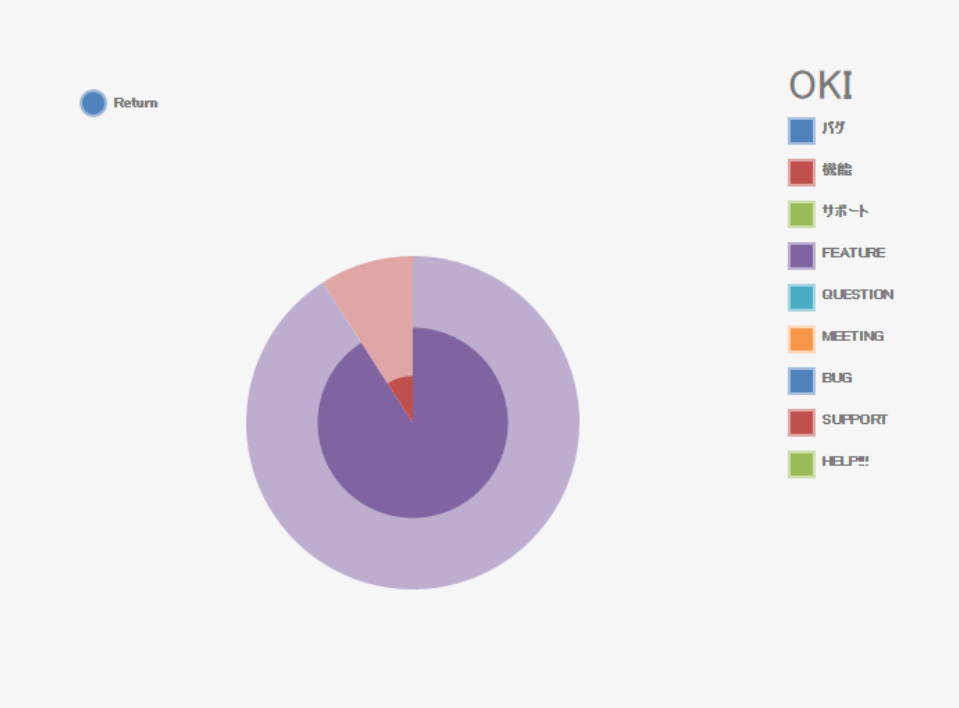


図 4.8: 修正後のチケット消化数と作業時間のグラフ (開発者詳細表示 1)

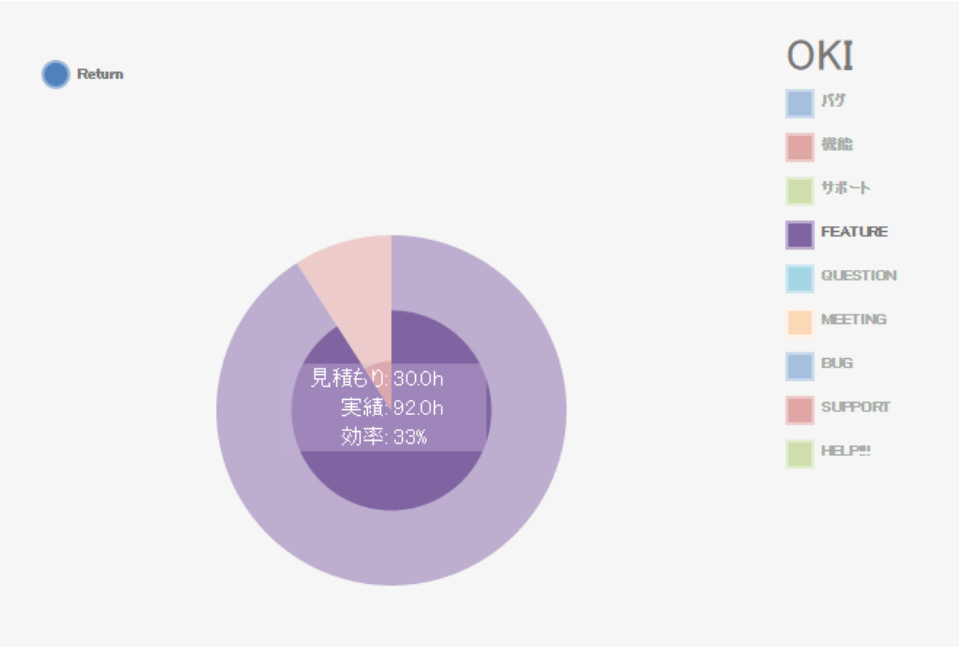


図 4.9: 修正後のチケット消化数と作業時間のグラフ (開発者詳細表示 2)

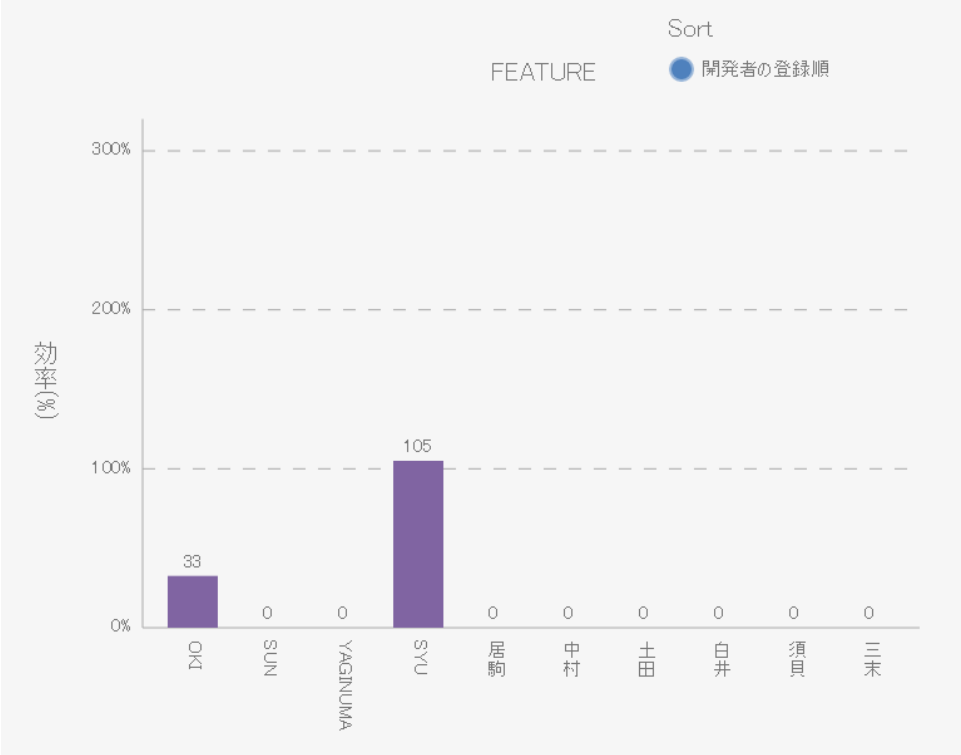


図 4.10: 修正後のチケット消化数と作業時間のグラフ (棒グラフによる一覧表示)

3. **コメント数 (図 4.11)**

修正前のグラフではある一人の開発者に着目していたが, 修正案であるヒートマップでは全開発者のコメントに関連する発言を表現した. 横に表示された開発者名は送信者, 縦に表示されている開発者は受信者を示しており, 各マスの濃さがコメント数の量を示している. また具体的な数値を確認するため, マスにマウスを置くと発言数が数字で表示されるように筆者は実装した. 加えて, 各開発者のコメント送信数および受信数に応じたソートを可能とした.

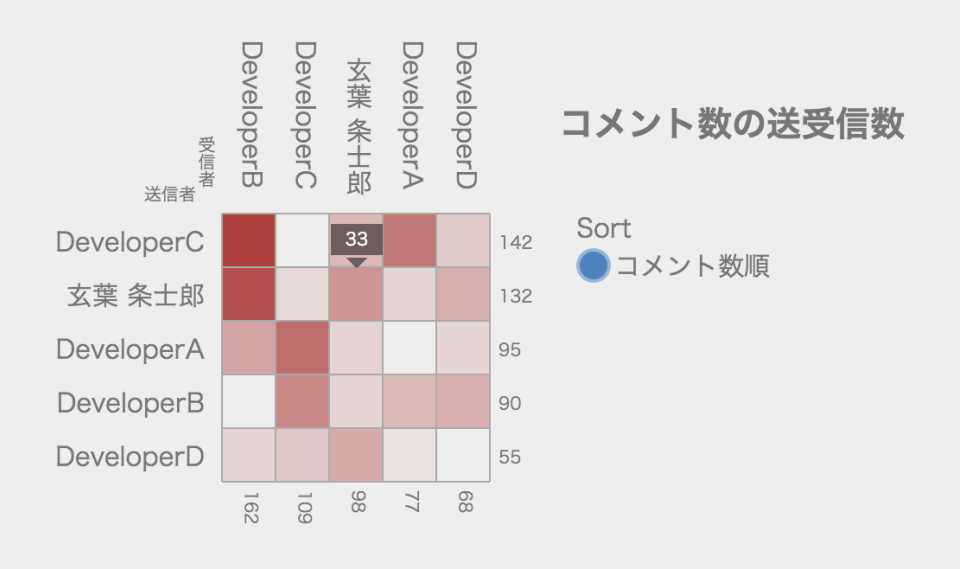


図 4.11: 修正後のコメント数のグラフ

4.5　エンドユーザ評価テストの結果

システムの有用性を客観的に評価するため, 第 3 スプリント終了後 (メトリクスデータ可視化モジュールの修正後) にエンドユーザ評価テストを実施した. エンドユーザの対象者は, 高度 IT コースの教員や日立のプロジェクト管理者である. エンドユーザ評価テストではクラウドサーバ上に展開されているシステムを利用した後, アンケートに回答していただく. 回答結果のうち, メトリクスデータ可視化モジュールに関連する設問 「本システムでそれぞれのメトリクスデータに対し, グラフの表示方法が適切だと思う」において得られた回答とコメントを以下に示す. なお本設問は 1 から 5 の五段階評価であり, 括弧内に示す数値は平均を表している. 回答者の人数は 10 名である.

1. **コミット数のグラフ (平均: 3.8)**
   コミット数のグラフに関しては, 比較的高い評価結果が得られた. コミット数のグラフでは, 各開発者のコミット数や平均値が直感的に解釈ができる点が良いとの意見が見受けられた. また中央値や分散, 時系列に関連した情報等の様々な情報が閲覧できるとなお良いというコメントが寄せられた.

2. **チケット消化数と作業時間のグラフ (平均: 2.7)**
   チケット消化数と作業時間のグラフに関しては, やや低めの評価となった. 主なコメントとして, グラフの操作や見方がわかりにくく, 一目で情報を理解しにくいといった意見が多く見受けられた. その反面, グラフの表示方法としては問題はなく, 必要な情報は収集できるとの意見もあった. 簡単なグラフの解釈を支援するため, 参照できる説明の充実が課題として挙げられる.

3. **コメント数のグラフ (平均: 3.2)**
   コメント数のグラフでは, ソフトウェア開発者のコミュニケーションや関連を閲覧するには適切であるとの意見が得られた. 一方, 見せ方を工夫すれば更なる知見の収集が期待できるという意見もあり, さらなる改善が課題として挙げられる.

アンケートの結果として, グラフによって想定していた情報を利用者に提供できてことがわかった. これらの回答に基づいた新たな可視化手法の検討や提供情報の拡充等が, 今後期待される.

一方, グラフに関連する参照情報が不足しているという意見がアンケート全体を通して見受けられた. 特に可視化手法は多数の情報を簡潔に表現しようとするあまり, 初めてのユーザにとっては理解が難しいものとなっている可能性がある. グラフの表現内容や簡単なコメントによる支援情報の提供等が, メトリクスデータ可視化モジュールにおける今後の課題として挙げられる.

第5章　プロジェクトにおける筆者の活動

本章ではメトリクスデータ可視化モジュールの実装以外に行った筆者の活動を述べる.

5.1　要件定義における活動

5.1.1　開発構想書の作成

要件定義において決定したプロジェクト方針やシステムの仕様等の情報は, 常に参照できなければならない. 筆者はこの記録として, 開発構想書 (付録 A) を作成した. 開発構想書の記載内容は, 主にプロジェクトの方針と提案システム MITEMIRU の概要である. なお開発構想書は実装に先立って決定される要件等を記した文章であるが, 実装開始後も用語の定義変更やドキュメントの追加等により適宜加筆した.

5.1.2　既存システムの調査

MITEMIRU の提案にあたり, 同様の既存システムについて調査をおこなった. この調査には, 大きく 2 つの目的がある.

1 つ目は, MITEMIRU の価値を明確にし, 他のシステムと差別化を図ることである. 機能および提供価値が重複するシステムが存在する場合, 新たにシステムを開発する必要性が著しく低下する. 実装段階に入った後にこの事実が判明すると, 単に開発チームメンバーのモチベーションが低下するだけでなく, プロジェクトの意義が損なわれてしまう. このリスクを避けるため, 調査によって MITEMIRU の価値や特徴等をあらかじめ明確にする.

2 つ目は, 既存システムの活用を検討することである. 本プロジェクトは実装に費やす時間や資源等に大きな制約があるため, 効率の良い開発が求められる. そこで既に存在する類似システムの拡張を視野にいれ, 簡潔な問題解決を試みた. そのためには, 新規開発と拡張開発のいずれかを選択するために, 既存システムを調査する必要がある.

以上の理由から, 筆者は開発チームメンバーと共に既存システムの調査を実施した. その結果, 以下に示す 2 つの類似システムが対象として挙げられた.

- **EPM-X**

  IPA/SEC(独立行政法人情報処理推進機構/ソフトウェア・エンジニアリング・センター) より, 定量的プロジェクト管理ツール EPM-X が公開されている [6]. Redmine と Trac の

二種類のチケット管理ツールに対応しており, いずれの場合もデータ収集機能 (プラグイン形式) とデータ可視化機能 (アプリケーション形式) を備えている.

- **Metrix Viewer**
  Metrics Viewer は, 坂元 康好らによって提案および開発されている個人の開発活動振り返り支援のための Web アプリケーションである [7]. 様々なリポジトリ内からファイルを探索し, 独自の API によってメトリクスの算出をおこなう [8]. 算出されたメトリクスは可視化機能によって, ユーザに提供される.

EPM-X は詳細なドキュメントやソースコード等, オープンソースとして公開されており, これを拡張することで MITEMIRU の基盤構築作業の短縮が期待できる. しかし, データ収集機能はチケット管理ツールのプラグイン形式で実装されており, チケット管理ツールに直接付けられる. そのため EPM-X の拡張形式を選択する場合, Redmine の利用を前提として実装されなければならない.

一方 Metrix Viewer は, MITEMIRU と同様の動作を実現するシステムである. しかし, 利用対象者はソフトウェア開発者自身であり, 振り返りが大きな目的である. そのため, 複数の評価対象者を視野に入れた相対的な比較機能は有しておらず, MITEMIRU が提供したい価値とは異なっている. またシステムは公開されておらず, MITEMIRU の開発のための利用は困難である.

以上のことから, 本プロジェクトでは既存システムの拡張ではなく新規開発することとした.

5.1.3 ペルソナの作成

ソフトウェア開発現場が抱える問題および解決すべき問題点を把握するため, 要件定義では日立のプロジェクト管理者にヒアリングを実施した. 具体的なユーザを明確するため, 筆者はヒアリング結果に基づいてペルソナを作成した. 以下に示すペルソナは, Redmine の Wiki に記載されているものである.

- **戸塚 浩明 (Hiroaki Totsuka)**
- **48 歳 男性**
- **背景**
  大規模プロジェクトを更に分割したサブプロジェクトにて, 約 10 人の開発者を統括する課長である. 開発者の取り組みを監視するとともに, 昇級のための評価を担当している. 開発者の評価基準は定められているが, 部署によって異なる. 場合によっては成果物に大きく貢献しているものの, 結果が見えてくるまでに 1, 2 年経過してしまうがためにすぐには評価されにくい開発者も存在する. 結局のところ, 自分自身の取り組みを積極的にアピールできるような開発者が評価されてしまう傾向があり, 戸塚自身も問題として捉えている. しかし, 途中経過の段階で取り組みを簡単に評価したいとは考えているものの, 開発者 1 人ひとりの取り組みを細かく監視することが困難だと感じている. ま

た新たなプロジェクト編成時には, 必要な人材のスキルを把握した上で人事部に紹介リクエストを出す. しかし, 他部署の管理者等に話を直接聞きに行き, 開発者の情報を収集した上で人材の選択する必要があるため大きな手間となっている.

本プロジェクトでは, このペルソナを MITEMIRU の仕様決定の参考とした.

5.2 第 1 スプリントにおける活動

5.2.1 動作環境の決定

開発環境および本番環境を構築するため, はじめに筆者は MITEMIRU の動作環境を決定した. 第 3.5 節にて述べたとおり MITEMIRU はオンプレミス型のシステムであり, ブラウザを通じて利用される. 本プロジェクトでは資源及び時間が大きく制限されるため, Web アプリケーションフレームワークである Ruby on Rails を利用することとした.

開発環境および本番環境での誤動作を削減するため, 利用ツールのバージョンを表 4.1 のとおり決定した.

表 5.1: 利用ツールのバージョン

ツール	バージョン
Ruby	2.2.2p95
Rails	4.2.3
RubyGems	2.4.8
MySQL	5.6.26

今後の開発では, この環境にしたがってシステムを開発するものとした. また MITEMIRU の開発チームメンバーやユーザがこの環境を簡便に実現できるようにするため, 導入方法を定義した.

5.2.2 データベースの設計

必要となるデータを MITEMIRU 上に保存するため, データベース設計をおこなった. そのうち, 筆者は概念モデルの設計を担当した. 要件定義において提案した機能の実現に必要となるデータを抽出した後, テーブルや関連の設計を進めた. 設計内容は開発チームメンバーのレビューを受けて修正を繰り返し, ER 図として記載した. その後の論理モデルおよび物理モデルの設計は, MITEMIRU の基盤構築とともに柳沼が担当した. なお第 1 スプリント終了後も, システムの機能実装に伴いデータベースの設計変更がおこなわれた. 図 5.1 に示す ER 図は, 本プロジェクト終了時の最終設計である.

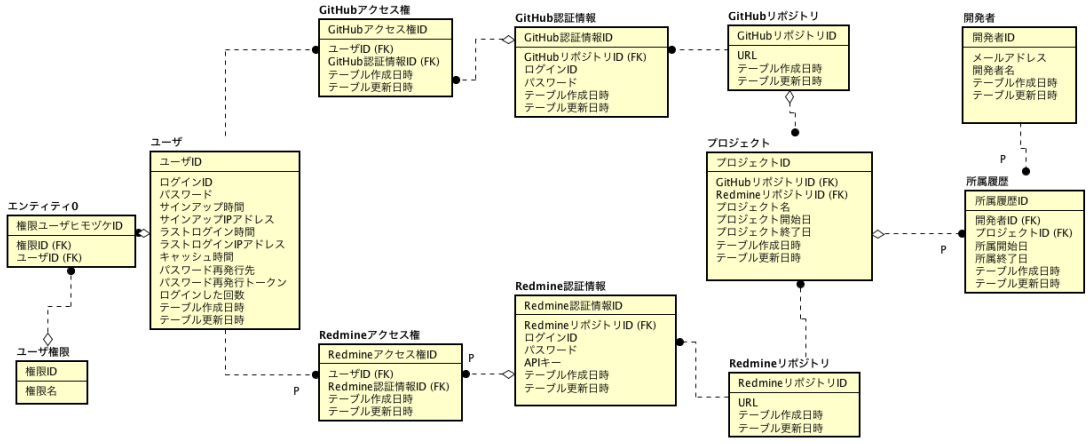


図 5.1: データベースの ER 図

5.3 第 2 スプリントにおける活動

5.3.1 メトリクスデータ可視化モジュールの設計と実装

本スプリントの主な活動は, メトリクスデータ可視化モジュールの設計および実装である. 詳細は第 4.3 節にて述べたとおりである.

5.3.2 日報の導入

第 2 スプリントでは, 見積もり作業時間に対して実績作業時間が大幅に減少してしまった. しかし各開発チームメンバーの活動状況が記録されておらず, 未作業時間の詳細が不明となる問題が発生した. この問題を解決するため筆者は日報を提案し, Excel シートを用いて作成および導入した (図 5.2).

日報には各チームメンバーごとに記入ページを設定し, そこに入力された情報は自動で集計される. スプリント開始時には見積もり作業時間を入力し, その合計値と前スプリントのベロシティ情報を組み合わせて見積もり消化ポイントを算出する. 開発作業開始後には, 毎日の作業に応じて見積もり実績作業時間と消化ポイント数を入力し, その日の作業内容や問題等をコメントとして記載する. これらの入力情報を集計し, 日々のデイリーミーティングにて確認することで各チームメンバーの作業状況を逐次共有した (図 5.3).

また, 第 2 スプリントまでは ZenHub によって自動生成されるバーンダウンチャートを利用していたが, 土日祝日をはじめとする休日が含まれてしまうため, 的確な現状把握ができていなかった. そこで, 入力される消化ポイント情報を利用したバーンダウンチャートを日報上に

日付	作業時間 [h]		消化pt	コメント
	見積もり	実績		
11月25日	7.0	4.0	0.0	#322に着手。日報のバーンダウンチャートを整理しました。3時間ほど会社関連の勉強で時間が取れず。
11月26日	7.0	5.0	0.0	日立のバリデーションの件を整理してSlackに投稿。#322にバグがあったので修正後、merge。#326に着手中。あとはレビュー。今日も会社関連の学習に時間を割いてました。不足分は土曜日に充填します。
11月27日	3.0	1.0	0.0	ちょっとレビューして終わりました。用事が入ったため作業できず。
11月30日	7.0	7.0	0.0	#326に着手。WIFIが死んだので作業が家でできなくなりました…。
12月1日	7.0	5.0	0.0	#326レビュー待ち。#326のデバッグするためにリポジトリに何度も問い合わせることになるので、作業がなかなか進まない。カフェで作業してるので、今後予定よりも時間が取れない可能性大。ミーティングで1時間。この後はドキュメント作業を中心に進める予定で
12月2日	7.0	8.0	3.0	#326をマージ。今日はレビューを中心に活動しつつ、#309の消化とRedmineのWiki編集を実施しました。あとは一民のチケットがなかなか進まないようだったので、そちらを手伝いました。Wi-Fiの件ですが、明日か明後日にWiMAXのルータが届くので、それを使えば家で作業できそうです。
12月3日	7.0	2.0	2.0	体調不良（多分風邪）のため、ほとんど作業できていません…。WiMAXルータが届いたので、作業は家でできます。レビュー待機状態の#309がやっとCloseできました。今は#310に着手。
12月4日	3.0	5.0	0.0	昨日の体調不良が悪化してます。つくばまで移動する気力はないですが、時間がないので作業はぼちぼち進めてます。今日中にUMLとシステム概要図は完成させたい。
12月7日	7.0	7.0	5.0	#310をClose。開発構想書作成してますが、思ったよりも時間がかかってます。
12月8日	7.0	4.0	8.0	開発構想書を編集。デイリーミーティング後に急用が入り、外出していました。
12月9日	7.0	8.0	8.0	#316に着手。効率のグラフを流用すれば実装はすぐに終わりそうだが、問題はデータの量が多すぎること。
合計	69.0	56.0	26.0	

図 5.2: 日報の記入例

日付	作業時間 [h]		消化ポイント	残高 [pt]		ベロシティ [pt/h]	備考
	見積もり	実績		見積もり	実績		
開始時				53.0	53.0		
11月25日	13.0	10.0	3.0	49.2	51.0	0.30	#291[1pt]追加
11月26日	13.0	9.0	5.0	50.3	52.0	0.42	#324[1pt], #325[2pt], #326[3pt]追加
11月27日	11.0	1.0	0.0	44.7	52.0	0.40	
11月30日	21.0	7.0	0.0	41.1	54.0	0.30	#329[2pt]追加
12月1日	21.0	19.0	0.0	38.2	57.0	0.17	#330[3pt]追加
12月2日	18.0	19.0	5.0	33.9	54.0	0.20	#333[2pt]追加
12月3日	17.0	11.0	9.0	27.1	45.0	0.29	
12月4日	13.0	23.0	5.0	20.3	40.0	0.27	
12月7日	21.0	29.0	8.0	13.5	32.0	0.27	
12月8日	21.0	16.0	16.0	6.8	16.0	0.35	
12月9日	21.0	22.0	16.0	0.0	0.0	0.40	
結果	190.0	166.0	67.0	0.0			

図 5.3: 入力情報の集計例

作成した (図 5.4). これにより, 実作業日のみを反映したバーンダウンチャートを参照できるようになり, 的確な現状把握を実現できた.

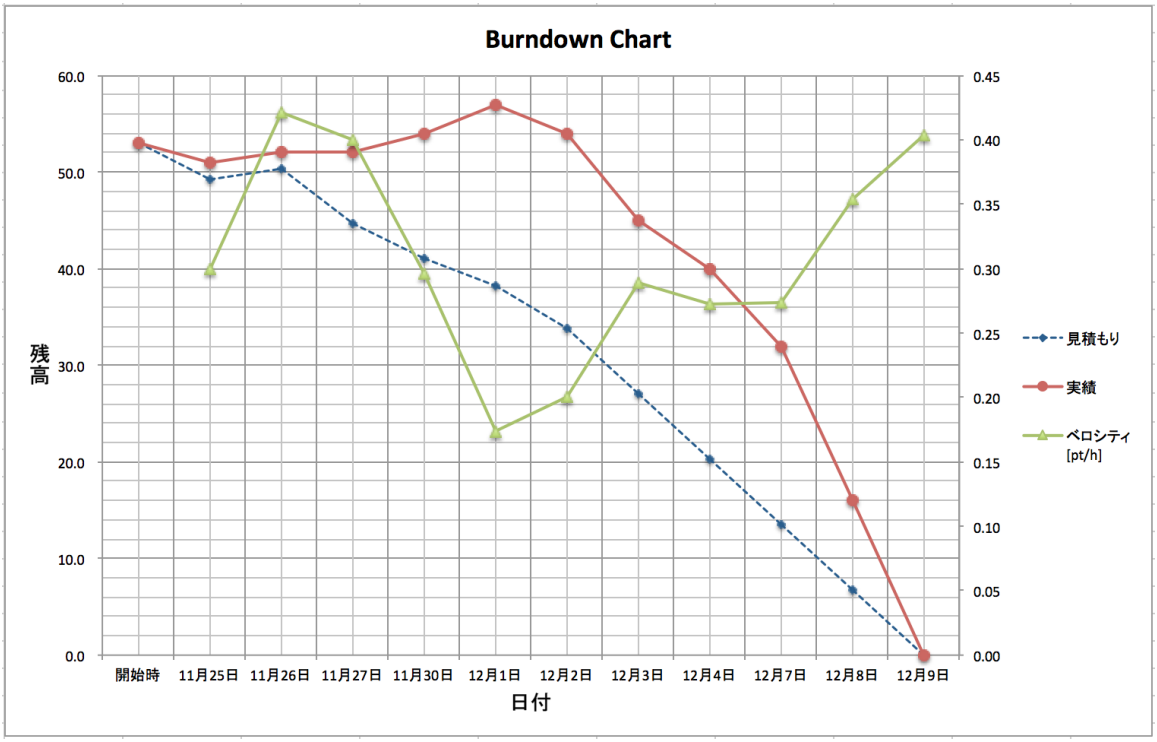


図 5.4: 日報上のバーンダウンチャート

5.4 第3スプリントにおける活動

5.4.1 メトリクスデータ可視化モジュールの再設計と修正

本スプリントでは, メトリクスデータ可視化モジュールの再設計と修正を実施した. 詳細は, 第 4.4 節にて述べたとおりである.

5.5 第4スプリント

5.5.1 グラフのバグ改善

ヒートマップとして作成したコメント数のグラフ (図 4.11) は, 各マスにマウスオーバーすることで対応するコメント数がポップアップで表示される. しかしソート時に表示されるポップアップの位置が乱れるバグが発生したため, 修正をおこなった. またその他のグラフについてもレイアウトの乱れが見受けられたため, 修正をおこなった.

5.5.2 画面遷移図の修正

第 1 スプリントにて画面遷移図を柳沼が作成し, その仕様にしたがってシステムの実装を進めた. しかしスプリントレビューの影響により画面構成の変更が発生しており, なおかつ画面遷移図が更新されていなかった. そこで筆者は, 第 3 スプリント終了段階での画面遷移図の仕様に書き換えをおこなった (図 5.5). なおこの画面遷移図は, 本プロジェクトの最終成果物と一致するものである.

5.5.3 UML の作成

システムの実装を進めていくうちに, 画面遷移図同様, 第 1 スプリント開始時に想定していた設計から幾つかの変更があった. これにより顧客と開発チーム内でのシステム利用フローの認識の差異が発生する可能性があったが, 参照できるドキュメントが不足していた. そこで筆者は参照できるドキュメントの拡充のため, ユースケース図およびアクティビティ図を改めて作成した. 図 5.6 は, 筆者が作成したユースケース図である. また筆者が同時並行で作成したアクティビティ図は, 本報告書の付録 B として記載する.

5.5.4 エンドユーザ評価テストの支援

第 4 スプリントと並行して, システムの提供価値やユーザビリティ等を評価するため, エンドユーザを対象とした評価テストを実行した. エンドユーザ評価の計画および実施は孫が担当したが, アンケートや依頼メール等の文章校正を筆者は担当し, 支援を積極的におこなった.

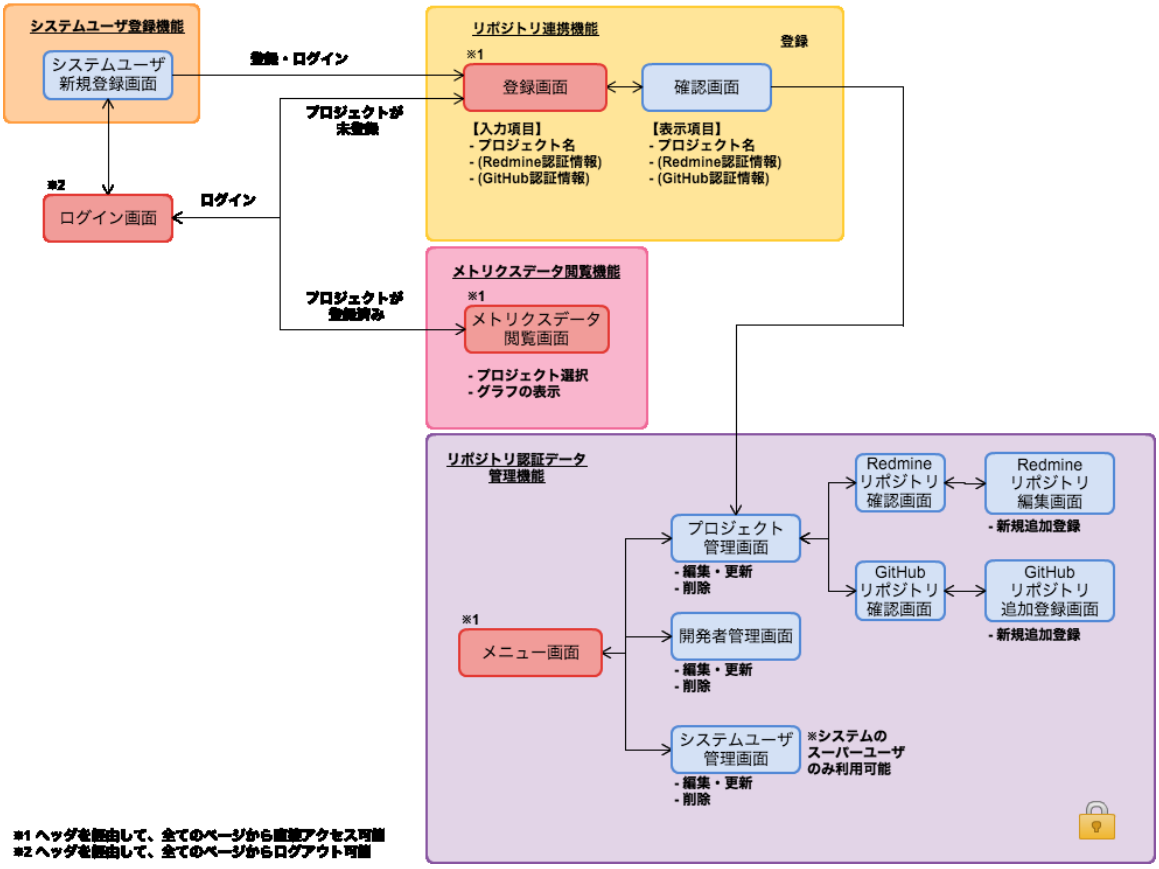

システムユーザ登録機能
システムユーザ
新規登録画面
登録・ログイン
※2
ログイン画面
ログイン
プロジェクトが
未登録
リポジトリ連携機能
※1
登録画面
登録
確認画面
【入力項目】
- プロジェクト名
- (Redmine認証情報)
- (GitHub認証情報)
【表示項目】
- プロジェクト名
- (Redmine認証情報)
- (GitHub認証情報)
プロジェクトが
登録済み
メトリクスデータ閲覧機能
※1
メトリクスデータ
閲覧画面
- プロジェクト選択
- グラフの表示
リポジトリ認証データ
管理機能
※1
メニュー画面
プロジェクト
管理画面
- 編集・更新
- 削除
開発者管理画面
- 編集・更新
- 削除
システムユーザ
管理画面
※システムの
スーパーユーザ
のみ利用可能
- 編集・更新
- 削除
Redmine
リポジトリ
確認画面
Redmine
リポジトリ
編集画面
- 新規追加登録
GitHub
リポジトリ
確認画面
GitHub
リポジトリ
追加登録画面
- 新規追加登録
※1 ヘッダを経由して、全てのページから直接アクセス可能
※2 ヘッダを経由して、全てのページからログアウト可能

図 5.5: 画面遷移図

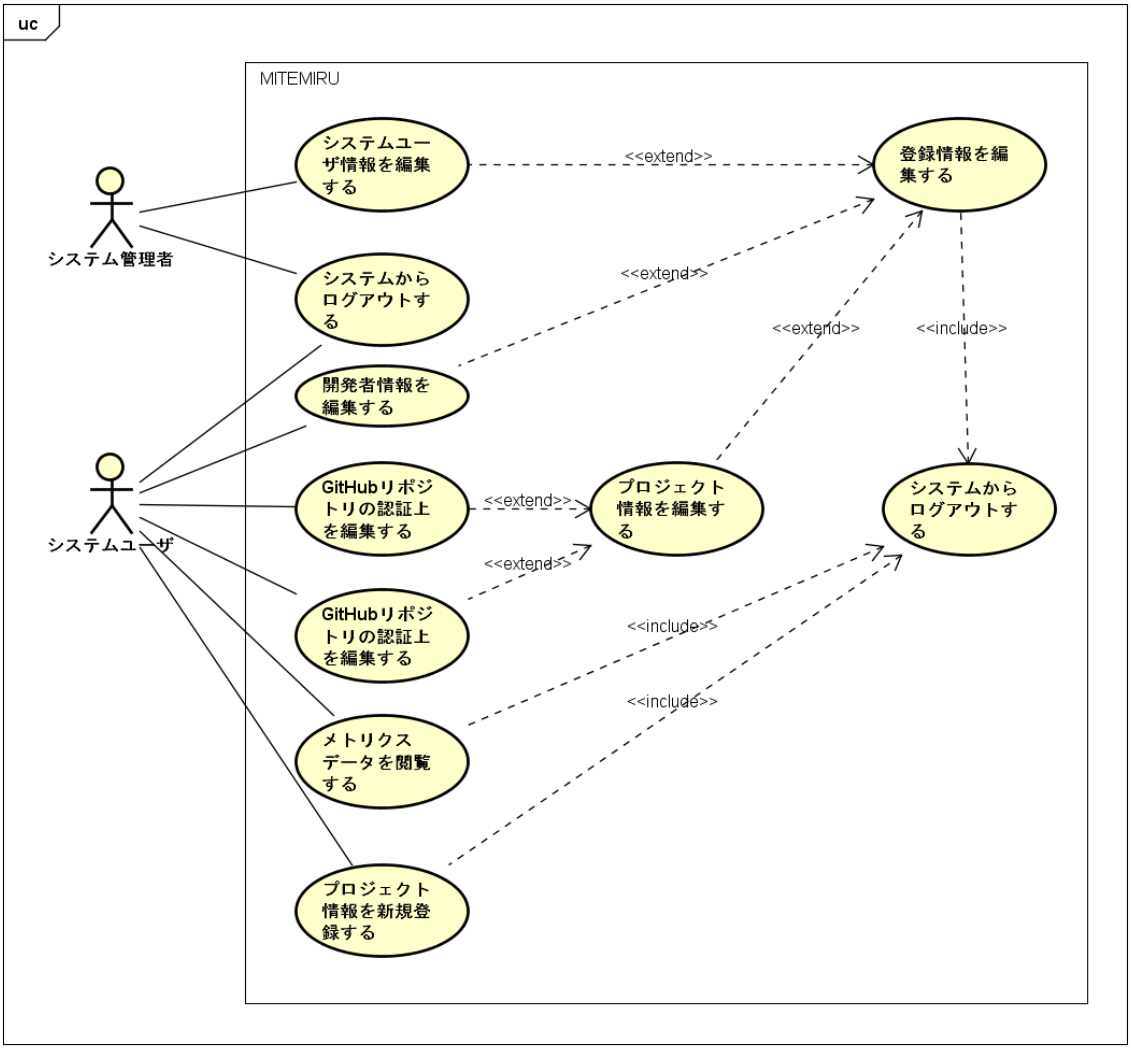

uc
MITEMIRU
システム管理者
システムユーザ
システムユーザ情報を編集する
システムからログアウトする
開発者情報を編集する
GitHubリポジトリの認証上を編集する
GitHubリポジトリの認証上を編集する
メトリクスデータを閲覧する
プロジェクト情報を新規登録する
登録情報を編集する
プロジェクト情報を編集する
システムからログアウトする
<<extend>>
<<extend>>
<<extend>>
<<include>>
<<extend>>
<<extend>>
<<include>>
<<include>>

図 5.6: ユースケース図

第6章　おわりに

本プロジェクトでは, ソフトウェア開発現場におけるプロジェクト管理者によるソフトウェア開発者の定量的な評価支援システム MITEMIRU を提案し, 開発した. 筆者はメトリクスデータ閲覧機能におけるメトリクスデータ可視化モジュールの実装を主に担当した. メトリクスデータ可視化モジュールの実装に先立ち, 筆者はメトリクスに関連する先行研究調査を行い, ソフトウェア開発者の評価材料となるメトリクスおよびメトリクスデータの定義を行った. 定義したメトリクスデータに基づいてメトリクスデータ可視化モジュールの設計を行った後, 顧客によるレビューや開発チームメンバーの意見を取り入れながら実装を進めた.

エンドユーザ評価テストでは,「実際にシステムを利用したい」というアンケートの設問 (1 から 5 の五段階評価) に対し, 平均 3.6 という評価が得られ, 本システムの有意性を立証できた. 約 4ヶ月と短い実装期間ではあったものの MITEMIRU の基盤となる機能を実装でき, 目標であったソフトウェア開発者の定量的な評価支援を実現する環境を構築できた.

筆者が担当したメトリクスデータ可視化モジュールに関しても, グラフの表現が適切で直感的に情報を解釈できたという意見がエンドユーザ評価テストにて得られた. より簡便かつ直截的な解釈を実現できるようなメトリクスデータ可視化モジュールの実現が今後の課題となる. その反面, アンケートではグラフの説明不足に関する意見が多く見受けられた. 必要に応じて情報を参照できる環境作り, グラフの表現内容や簡単なコメントによる支援情報の提供等が早急に求められる.

MITEMIRU はオープンソースプロジェクトとして GitHub 上に公開されている. 本プロジェクト終了後も有志によってシステムの開発が進められ, 提供できるメトリクスデータや連携リポジトリの充実等, さらなるシステムの発展が期待される.

謝辞

本報告書の執筆にあたり, 指導教員の田中二郎教授から丁寧なご指導を賜りました. 本プロジェクトの遂行においては, 課題担当教員および情報可視化の専門家である三末和男教授より日々のご指導, ご助言をいただきました. 株式会社日立製作所の居駒幹夫様, 土田正士様, 須貝佳彦様, 白井明様, 中村宇佑様, 河野哲也様には, プロジェクト運営から本報告書の執筆等, 様々な場面にてご助言やご意見をいただきました. 皆様のご支援により, 本プロジェクトを終えられましたことに深く感謝申し上げます.

エンドユーザ評価テストでは, 筑波大学の教員および株式会社日立製作所の皆様にご協力を頂き, 大変お世話になりました. おかげさまで貴重なご意見をいただくことができ, システムに対する有用な評価結果が得られました. ここに, 感謝の意を表します.

また, 高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムの同期であり開発チームメンバーである柳沼工也, 孫嘉成, 鄒一民には, 仲間として多数の場面でお世話になりました. 心より感謝申し上げます.

参考文献

[1] 平鍋健児, 野中郁次郎, “アジャイル開発とスクラム 顧客・技術・経営をつなぐ協調的ソフトウェア開発マネジメント,” 翔泳社, 2013.

[2] 小野和俊, “Developer のための 5 つの習慣―日本をソフトウェア輸出大国にしていくために,” デベロッパーサミット 2006.

[3] 五田 篤志, 山崎 尚, 玉田 春昭, 畑 秀明, 角田 雅照, 井垣 宏, “開発履歴を利用した風林火山モデルに基づく開発者特性の分析,” 情報処理学会研究報告, ソフトウェア工学研究会, Volume 2014-SE-185, No.9, pp.1-6, 2014.

[4] 五田 篤志, 山崎 尚, 玉田 春昭, 畑 秀明, 角田 雅照, 井垣 宏, “開発履歴による開発者特性とアンケートによる特性の自己診断の関連分析,” 情報処理学会研究報告, ソフトウェア工学研究会, Volume 2015-SE-187, No.5, pp.1-8, 2015.

[5] 坂口 英司, 伊原 彰紀, 尾上 紗野, 畑 秀明, 松本 健一, “複数のオープンソースプロジェクトに参加する開発者による貢献の分析,” 情報処理学会研究報告, グループウェアとネットワークサービス研究会, Volume 2014-GN-92, No.15, pp.1-4, 2014.

[6] 独立行政法人 情報処理推進機構, “定量的プロジェクト管理ツール（EPM-X）：IPA 独立行政法人 情報処理推進機構,” `http://www.ipa.go.jp/sec/softwareengineering/tool/ipf/`, (2016/01/04 アクセス).

[7] 坂元 康好, 本 真佑, 中村 匡秀, “MetricsViewer: サービス指向リポジトリマイニングを活用したソフトウェアメトリクス可視化ツール,” 電子情報通信学会技術研究報告, Vol.112, pp.127-132, 2013.

[8] 坂元 康好, 本 真佑, 中村 匡秀, “サービス指向リポジトリマイニングを効率化するキャッシュ機構の実装,” 電子情報通信学会技術研究報告, Vol.113, pp.73-78, 2013.

付 録A　開発構想書

開発構想書

リポジトリデータを利用した
ソフトウェア開発者評価支援システム
の開発

研究開発チーム ViBi

Date	Revision	Description	Author
2015/06/11	1.0	初版	Ryota Oki
2015/06/26	1.1	レビュー後の修正	Ryota Oki
2015/08/07	2.0	第二版	Ryota Oki
2015/12/08	3.0	第三版	Ryota Oki
2015/12/28	3.1	指摘事項の修正	Ryota Oki

目次

1　序論

1.1　本書の概要

本書は，筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻の「高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム」の一環である，「研究開発プロジェクト」における成果物の開発構想について述べたものである．成果物の開発目的や概要，対象利用や開発体制など，開発に関する情報を明確にするものである．

1.2　問題提起

ソフトウェア開発プロジェクトの多くは複数のソフトウェア開発者による共同作業である．しかしプロジェクト管理者にとって，そのプロジェクトに対するソフトウェア開発者のパフォーマンス(能力や経験)を定量的に測れないという課題がある．プロジェクト管理者の主観的かつ曖昧な評価によって，ソフトウェア開発者のモチベーションの喪失し，ひいてはプロジェクト運営の失敗につながる場合も少なからず存在する．この問題の解決のため，プロジェクト管理者が複数のソフトウェア開発者に対して定量的な根拠による客観的な評価をできるようなソリューションが望まれている．

1.3　本プロジェクトの方針

本プロジェクトは，株式会社日立製作所(以下，日立)による問題提起(第1.2節)によって発足したプロジェクトである．

株式会社日立製作所により提示された問題を解決するため，本プロジェクトではMITEMIRUの提案および開発をおこなう．MITEMIRUは，評価対象者が関連するソフトウェア開発プロジェクトに対応する構成管理ツール等のリポジトリから取得したデータを集計･分析し，加工する．加工データをグラフとして可視化することにより，ソフトウェア開発者のパフォーマンスに関連する定量的な評価材料として提供し，日立に提示された問題を解決する．

MITEMIRUの実装にあたり，まずはリポジトリマイニングやプロジェクト管理等に関連する文献調査により，リポジトリ上に存在するデータと対応するソフトウェア開発者のパフォーマンスの関係を明確にする．次に，この知見に基づき，パフォーマンス評価の判断材料となるメトリクスとその要素なるメトリクスデータ，およびメトリクスデータの表現に必要となるリポジトリ上のデータを検討する．そして，提供するメトリクスデータと具体的な可視化方法を決定する．

なお本プロジェクトでは，多数のソフトウェア開発者が共同作業するソフトウェア開発プロジェクトを対象に，客観的な各ソフトウェア開発者の評価支援を可能とする環境構築を目標とする．

2 システム利用者

2.1 対象となる利用者

MITEMIRU は, 10 名程度のソフトウェア開発プロジェクトの管理者を利用対象者とする．例として，企業における小規模なソフトウェア開発プロジェクト(大型プロジェクトを分割した，小型プロジェクトを含む)の管理者や，PBL の学生評価をおこなう指導教員等が挙げられる．

2.2 利用者が抱える問題

利用者が抱える問題として，ソフトウェア開発者の不適切な評価が挙げられる．ソフトウェア開発プロジェクトは複数の開発者による共同作業であり，プロジェクトの推進には各開発者の役割分担や能力が大きく影響を与える．そのため，プロジェクト開始以前にソフトウェア開発者を適切に選定しなければならない．

他方，企業における人事評価や PBL における成績判定等，プロジェクト開始後には定期的なソフトウェア開発者の評価を実行しなければならない．

いずれのケースにおいても厳密かつ適切な評価は必要不可欠であるが，定量的な根拠に基づいた開発者の経験および能力の把握はプロジェクト管理者にとって困難であり，これは問題の一因として考えられる．

MITEMIRU はこの原因を解消するため，ソフトウェア開発者評価のための定量的な判断材料を提供する．

3 システムの概要

3.1 システムの役割

MITEMIRU は，ソフトウェア開発者評価の定量的な根拠に基づくパフォーマンスの客観評価を支援するシステムである．MITEMIRU は，バージョン管理ツールやチケット管理ツール等のリポジトリに蓄積されたデータを収集し，集計や分析をおこなう．その後，この集計・分析結果をソフトウェア開発者の評価に関連する定量的な判断材料(メトリクスデータ)として，利用者であるプロジェクト管理者に提供する．また MITEMIRU は直截的な解釈を実現するため，メトリクスデータをグラフとして可視化し，利用者に提供する．これにより，2.2 節にて述べた問題の一因解決が期待できる．

3.2　システムが備える機能

3.1 節にて述べた役割を実現するために, MITEMIRU は，大きく 4 つの機能を備える．

1 つ目は，「リポジトリ連携機能」である. 3.1 節で述べた通り, MITEMIRU はバージョン管理ツールやチケット管理ツール等のリポジトリにアクセスし，蓄積されたデータを取得する．リポジトリへのアクセスには認証が必要であり，データ取得時に認証情報を利用しなければならない．MITEMIRU ではリポジトリへの円滑なアクセスを実現するため，認証情報を予め登録するための「リポジトリ連携機能」を実装することとする．この機能は，利用者が保有する複数のリポジトリに対する認証情報を紐付け，MITEMIRU のデータベースに保存するものである．

2 つ目は，「メトリクスデータ閲覧機能」である．これは，評価対象となるソフトウェア開発者の定量的なパフォーマンス評価の判断材料を提供するための，主要な機能である．MITEMIRU はリポジトリ連携機能で予め登録された認証情報を利用して，リポジトリへアクセスし，蓄積されたデータを収集する．その後，集計・分析を実行した後グラフとして可視化し，利用者に提供する．利用者は，このグラフから得られた知見をパフォーマンス評価の判断材料として利用する．グラフはプロジェクト毎に確認でき，そのプロジェクトに所属する全ての開発者に関連するメトリクスデータが表示される．一方，ソフトウェア開発者を指定し，関連する全てのプロジェクトでのパフォーマンスを時系列に並べて確認することも可能とする．

3 つ目は，「各種管理機能」である．主にデータの管理を目的としており，リポジトリ連携機能で登録されたプロジェクトやリポジトリの認証情報の更新や削除を実行する機能である．

これらの主要機能に加えて，システムユーザのログイン・ログアウト機能や，メトリクスデータのエクスポート機能等，様々な機能を検討している．詳細は第 4.2 節に述べる．

3.3　システムの形態

図 1 にシステムの概要図を示す. MITEMIRU は，リポジトリへのアクセスが可能である環境にインストールされなければならない．特に企業では，外部ネットワークから隔離された環境にリポジトリが設置されることが大いに想定されるため, MITEMIRU はその環境に応じてインストールされる必要がある．そこで, MITEMIRU はオンプレミス型の運用形態を想定する．また MITEMIRU

は汎用性を高めるため，ブラウザ経由で利用されるものとする．なお複数人による利用も想定されることから，ログインによる利用者の識別を必須条件とする．

4　システムの要件

4.1　機能要件

本プロジェクトは時間が制約されているため，限定されたスコープで開発を実施する．そこで，顧客とのミーティング等で機能要件の洗い出しをおこなった(表 1)．またそれら対して優先度を設定し，優先度の高いものから実装することとする．優先度の定義は，以下の通りである．

- 高: 利用者の最低限の要件を満たすために，システムの動作上，必須となる機能
- 中: 利用者の要件を満たすため必要ではあるが，不足してもシステムを動作させられる機能
- 低: 利用者の要件として求められてはいるが，必要ではない機能

表 1: 機能要件と優先度

主機能	機能要件	概要	優先度
ログイン・ログアウト機能	システムユーザの新規登録	システムの利用者を一意に識別するためのシステムユーザ情報を，データベースに保存する．	高
	ログイン	保存されたシステムユーザ情報を利用し，システムにログインさせる．	高
	ログアウト	システムからログアウトさせる．	高
	システムユーザ情報管理	データベースに保存されたシステムユーザ情報に対し，更新および削除をおこなう(ログイン時のシステムユーザ情報に限る)．	中
リポジトリ連携機能	リポジトリ認証情報の	リポジトリの問い合わせに必要となる認証情報をデータベースに保存する．	高

	保存		
	開発者情報の登録	リポジトリに関連する全ての開発者情報をデータベースに保存する.	中
メトリクスデータ閲覧機能	プロジェクトの選択	メトリクスデータを閲覧したいプロジェクトを選択できる.	高
	メトリクスデータの生成と可視化（同一プロジェクト）	メトリクスデータを閲覧したいプロジェクトを選択し，リポジトリからデータを取得した後，メトリクスデータとして集計・分析結果を生成する.それをグラフとして可視化し，利用者に提供する．なお，選択されたプロジェクトに所属する全ての開発者に関連するメトリクスデータを算出する.	高
	メトリクスデータの生成と可視化（複数プロジェクト）	メトリクスデータを閲覧したい開発者を複数選択し，リポジトリからデータを取得した後，メトリクスデータとして集計・分析結果を生成する.それをグラフとして可視化し，各開発者のパフォーマンスを比較する形式で利用者に提供する．なお，選択された開発者が所属する各プロジェクトのリポジトリからデータし，それぞれの開発者に応じてメトリクスデータを算出する.	
	メトリクスデータの生成と可視化（単一開発者）	メトリクスデータを閲覧したい開発者を選択し，リポジトリからデータを取得した後，メトリクスデータとして集計・分析結果を生成する.それをグラフとして可視化し，利用者に提供する．なお，選択された開発者に関わる全てのメトリクスデータを算出し，時系列に応じてグラフとして表示する	中
	レポートファイルの出力	メトリクスデータの結果とグラフを外部ファイルとして出力する.	低
各種管理機能	プロジェクト情報管理	データベースに保存されたリポジトリの認証情報に対し，更新および削除をおこなう.	高
	開発者情報管理	データベースに保存された開発者情報に対し，更新および削除をおこなう.	中

	システム ユーザ情報 管理	データベースに保存されたシステムユーザ情報に対し，更新および削除をおこなう(スーパユーザのみ利用可能，全てのシステムユーザ情報が対象)．	中

4.2　非機能要件

4.1 節と同様に，表 2 に非機能要件を示す．なお，表 2 に記載がない項目については，考慮しないものとする．

表 2: 非機能要件と優先度

項目	特徴	概要	優先度
性能・ 拡張性	リソース 拡張性	システムで利用するリソース(リポジトリデータ)の取得先が簡便に追加可能であること．	高
移行性	移行方式	アプリケーション形式でのインストールが可能であること．	高
	移行対象 (機器)	汎用型コンピュータへ簡便に移行可能であること．	高
	移行対象 (データ)	汎用型コンピュータで利用可能な範囲のデータベースに移行可能であること．	高

5　開発体制

5.1　ステークホルダー

本プロジェクトの主要なステークホルダーを表 3 に示す．本プロジェクトの顧客は株式会社日立製作所(日立)であり，三末和男教授の指導の元で筑波大学の学生がシステムの開発をおこなう．加えて，日立はソフトウェア開発分野，三末和男教授は情報可視化分野のアドバイザーとして，開発に参加する．

表 3: ステークホルダー

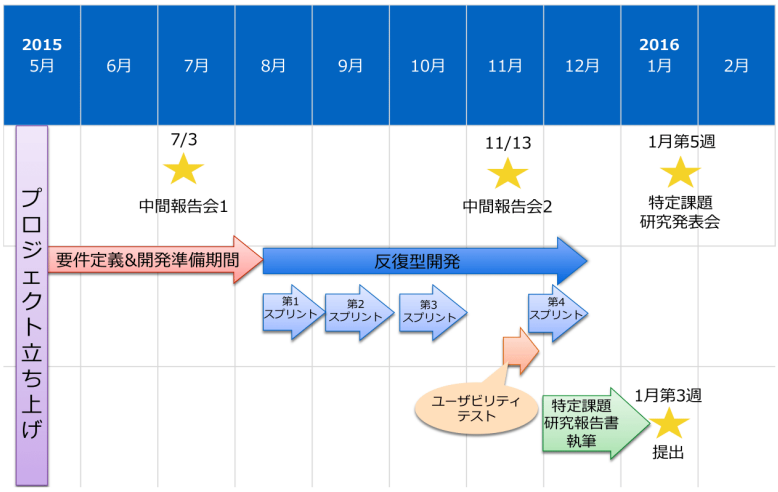
所属	役割	名前
筑波大学	課題担当教員, アドバイザー	三末 和男
	開発チーム	大木 遼太
		柳沼 工也
		孫 嘉成

		鄒 一民
株式会社 日立製作所	顧客, アドバイザー	居駒 幹夫
		土田 正士
		白井 明
		中村 宇佑
		河野 哲也
		須貝 佳彦

5.2　スケジュール

本プロジェクトは，図2に示すスケジュールを想定している．本プロジェクトでは期間の延長は一切認めらないため，大きなスケジュールの変更は想定していない．なお，スケジュールに加えて開発チームメンバーの入れ替えや増員も不可能であるため，適切なプロジェクト遂行のためにはスコープを調整が必要不可欠となる．そこで本プロジェクトでは反復型開発手法を採用し，顧客との密なコミュニケーションによる優先事項の検討と，それに基づいた機能の実装予定している．

図 2: 本プロジェクトのスケジュール



5.3 開発における利用ツール

5.3.1 GitHub

本プロジェクトの成果物のうち，MITEMIRU のソースコードや関連するドキュメントを GitHub リポジトリ上に保存する．なお，これらのデータはすべてオープンリポジトリ上に保存し，公開されるものとする．

5.3.2 Redmine

プロジェクトの進捗管理には,オープンソースのプロジェクト管理ソフトウェアである Redmine を利用する．またプロジェクトに関連する情報は逐次 Wiki に記載され，更新されるものとする．

5.3.3 Dropbox

オンラインストレージサービスである Dropbox を利用し，チームでの活動において生成されるデータやドキュメント等を管理する．必要に応じて学生チームと指導教員，アドバイザーとのデータ共有も同様におこなうものとする．

5.3.4 Slack

チーム内コミュニケーションツールとして，Slack を利用する．なお，学生チームと指導教員，およびアドバイザーとのコミュニケーションについては，

簡便なものに限り Slack 上でおこなうものとする．

5.4 開発におけるリスク

5.4.1 利用者の制約

MITEMIRU ではメトリクスデータ閲覧機能の実装時に，利用対象となるリポジトリを選定する必要がある．そのため，利用者によるレビューを実施するためには対象となる利用者が数多く使用しているリポジトリを選定しなければならず，場合によってはレビューを実行できない可能性も考えられる．

5.4.2 就職活動

2016 年度卒業者向けの就職活動は，経団連の採用選考に関する指針変更により，広報活動を２０１５年の３月１日以降，選考活動を２０１５年 ８ 月 1 日以降におこなうとの指針を提示している．しかし現状は，「倫理憲章」賛同企業もインターンシップやジョブマッチングといった形で様々な広報活動を実施している．その結果，チームメンバーの就職活動は常に活発な状態であり，プロジェクトの参加時間が限られてしまうといった現状がある．

付 録B　UML(アクティビティ図)

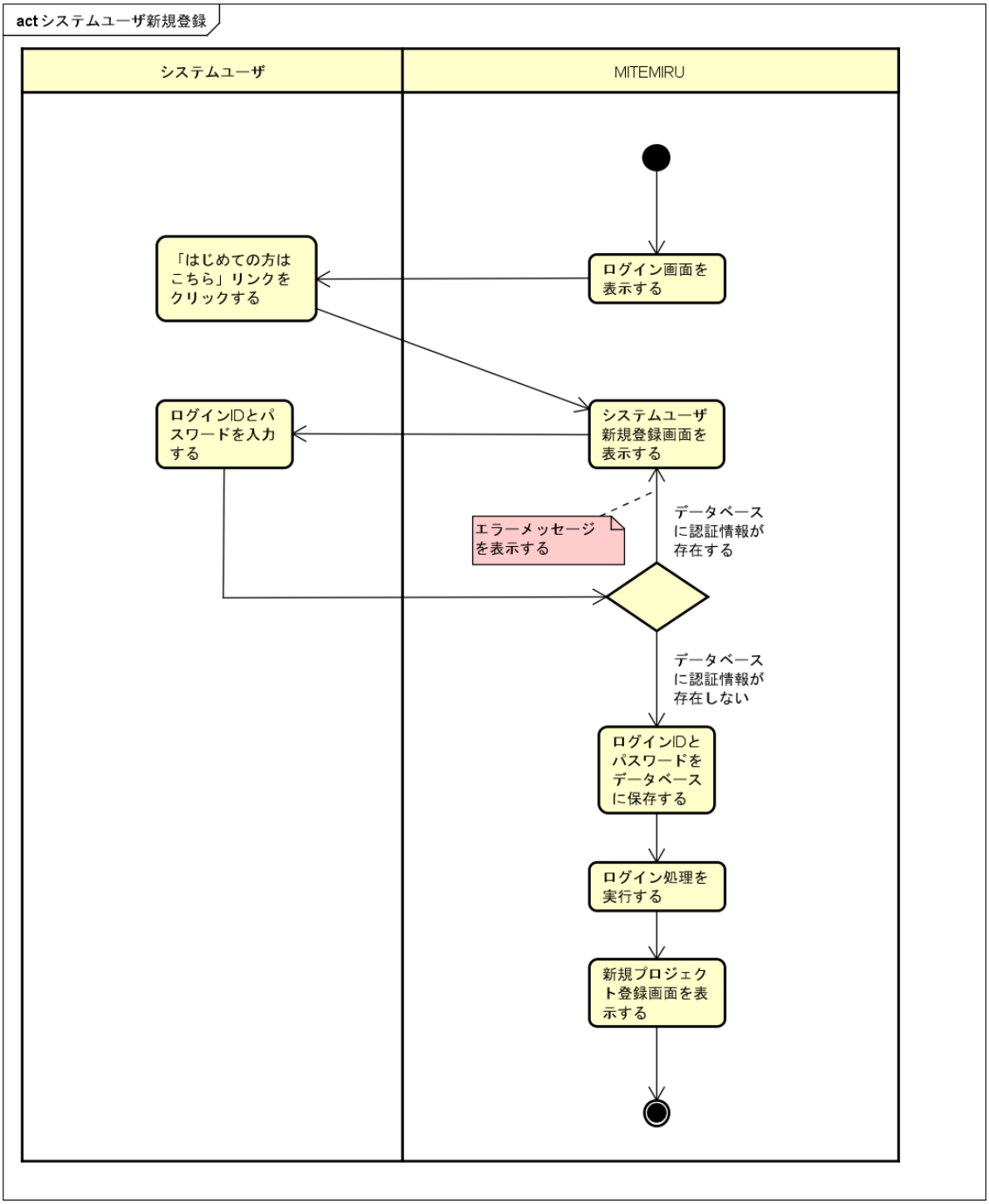


図 B.1: アクティビティ図 (システムユーザ新規登録)

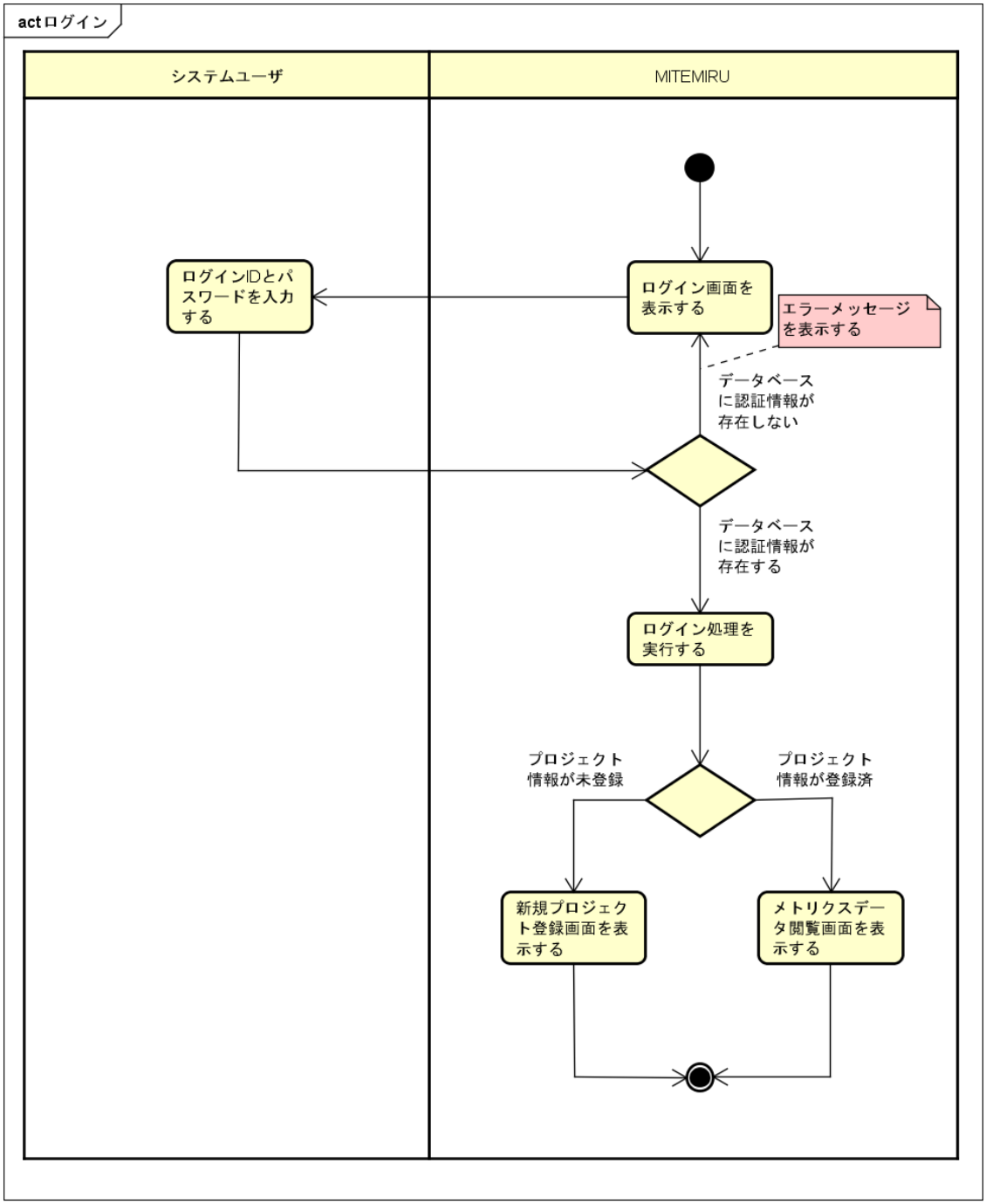


図 B.2: アクティビティ図 (ログイン)

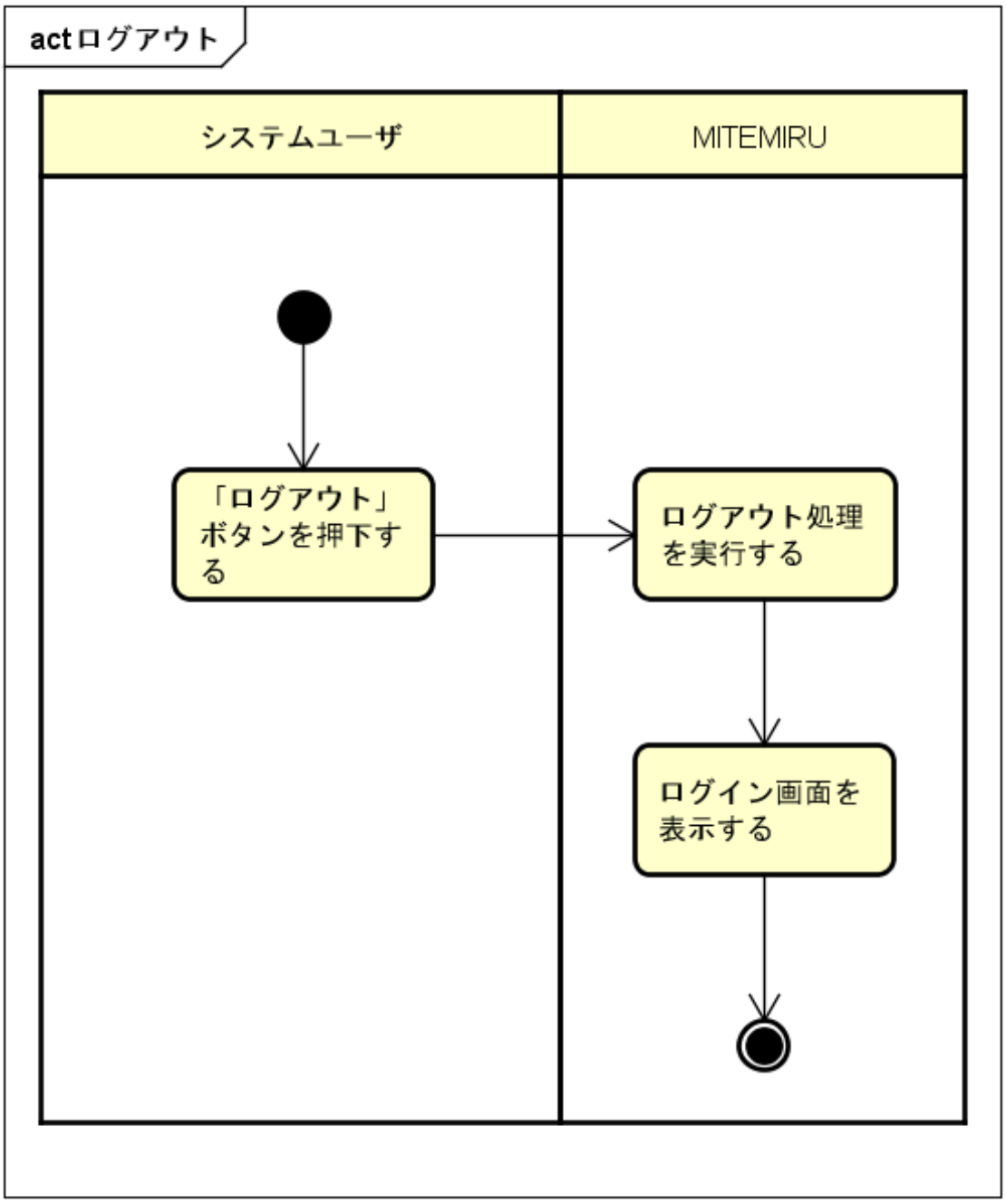


図 B.3: アクティビティ図 (ログアウト)

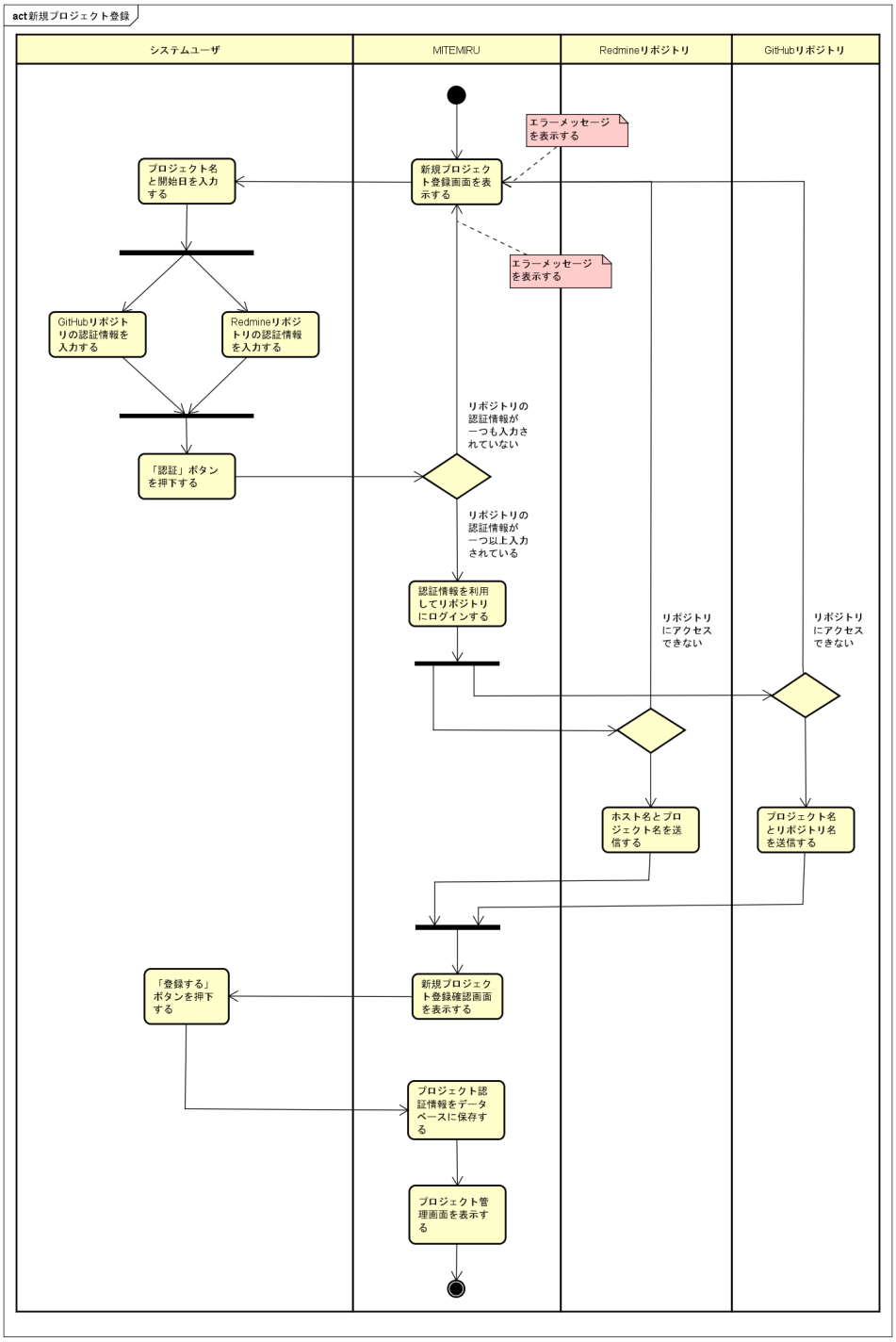


図 B.4: アクティビティ図 (新規プロジェクト登録)

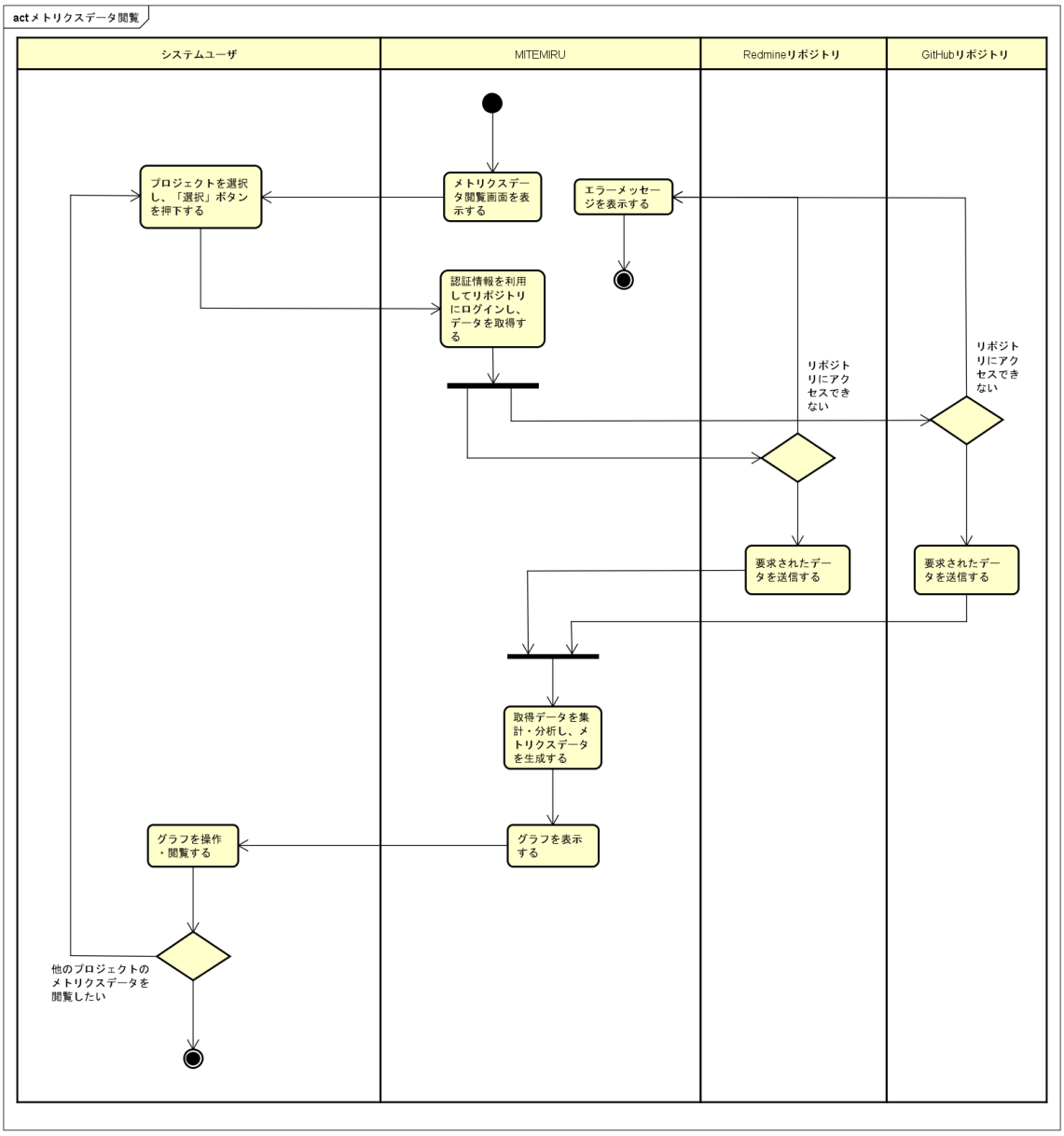


図 B.5: アクティビティ図 (メトリクスデータ閲覧)

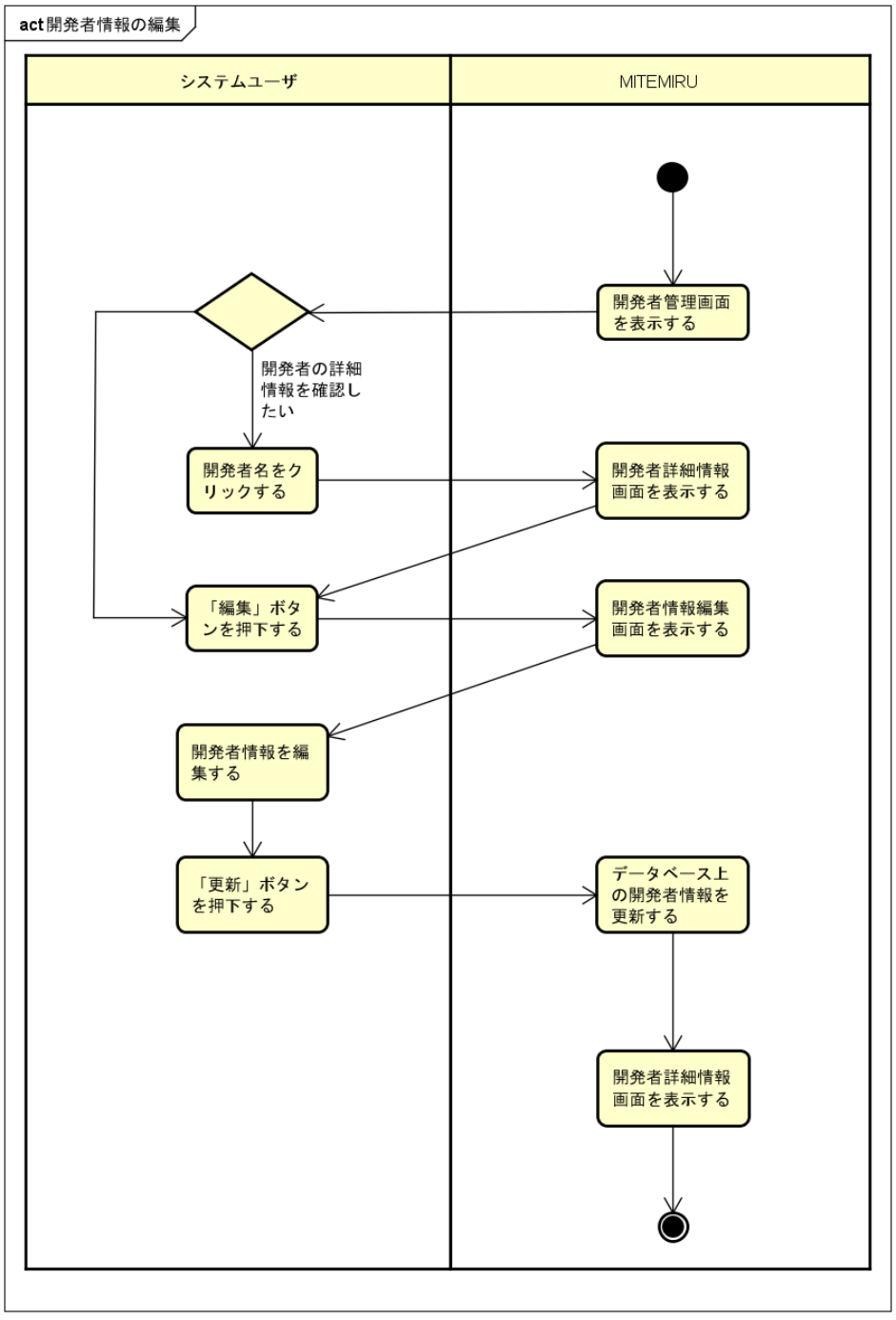


図 B.6: アクティビティ図 (開発者情報の編集)

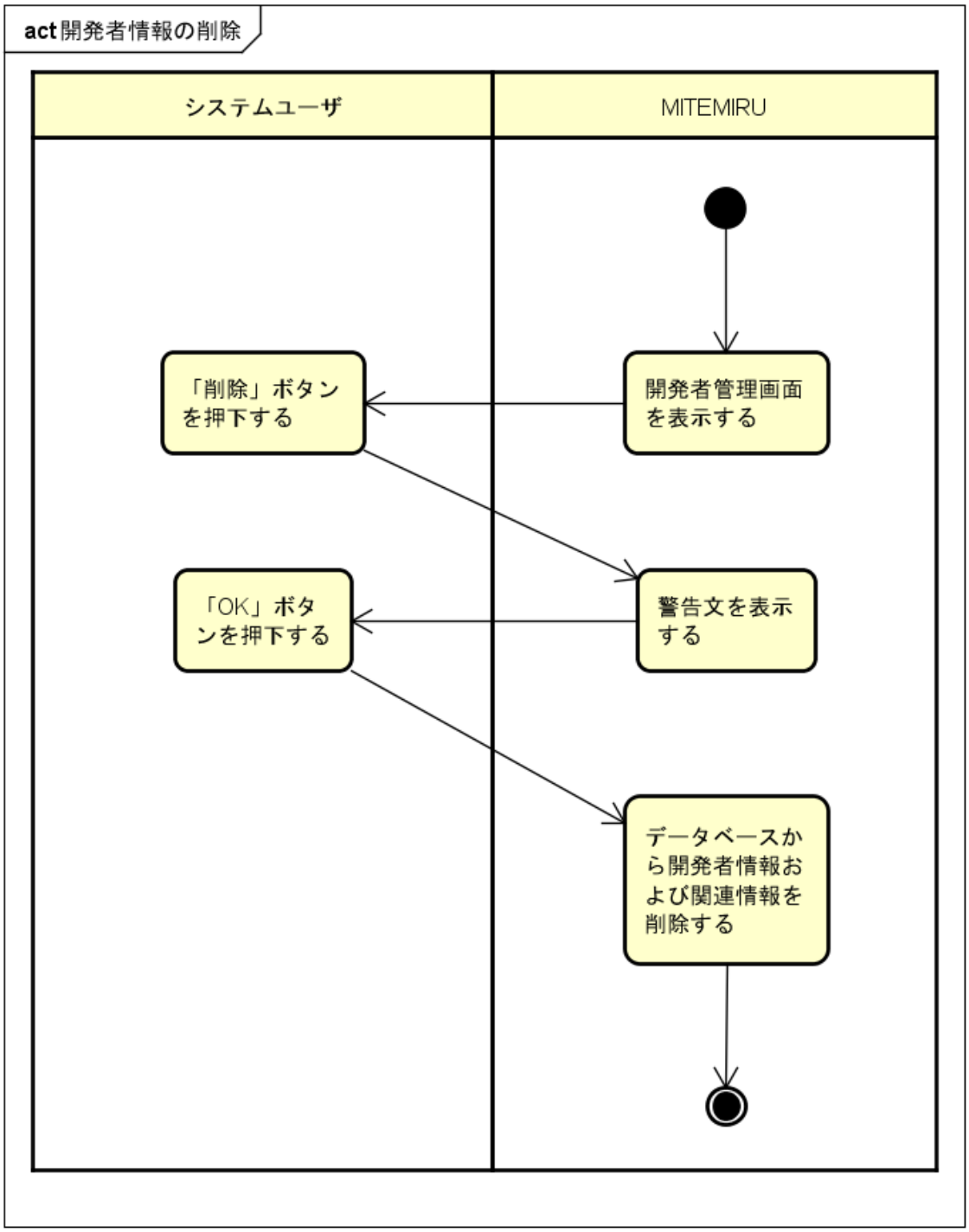


図 B.7: アクティビティ図 (開発者情報の削除)

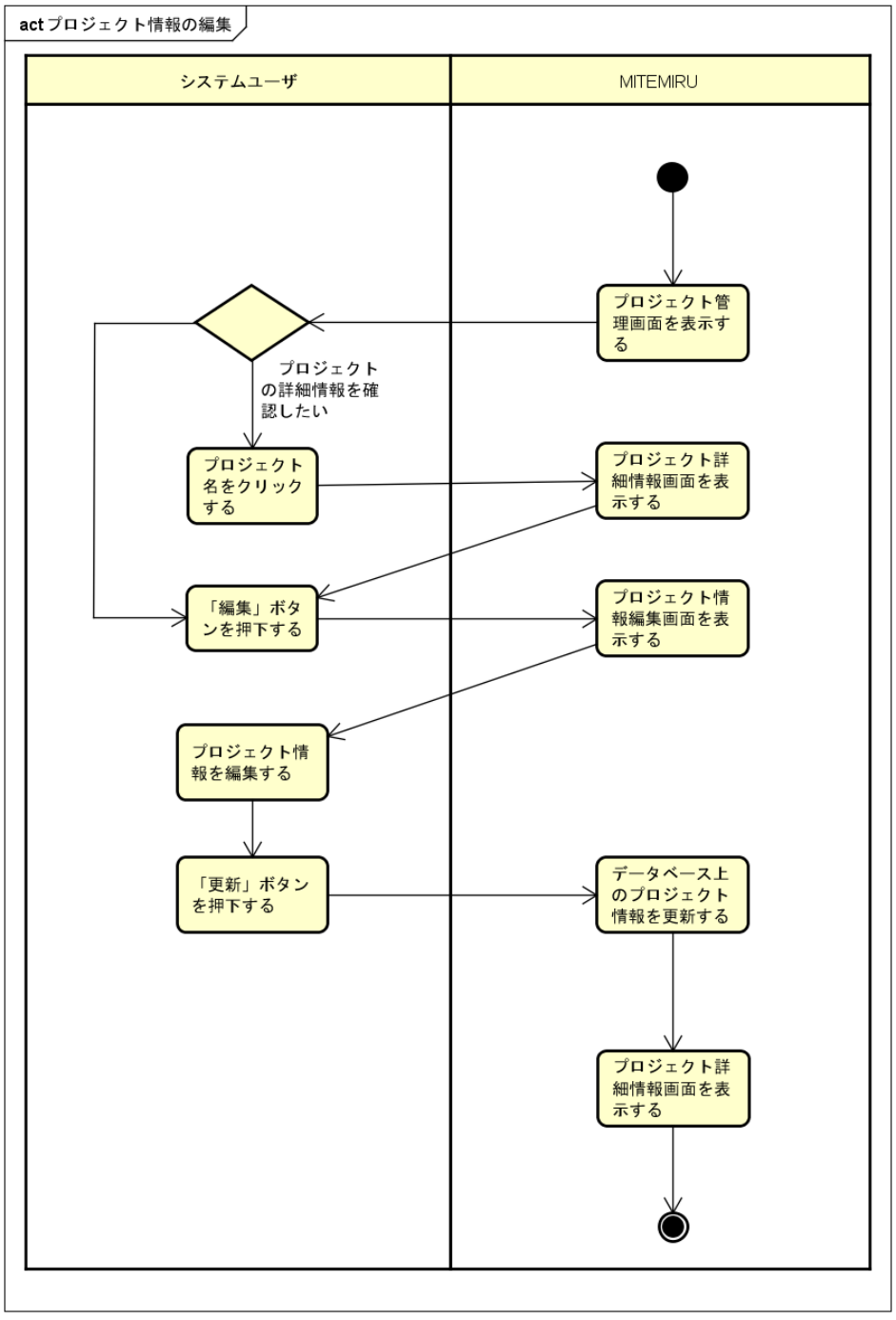


図 B.8: アクティビティ図 (プロジェクト情報の編集)

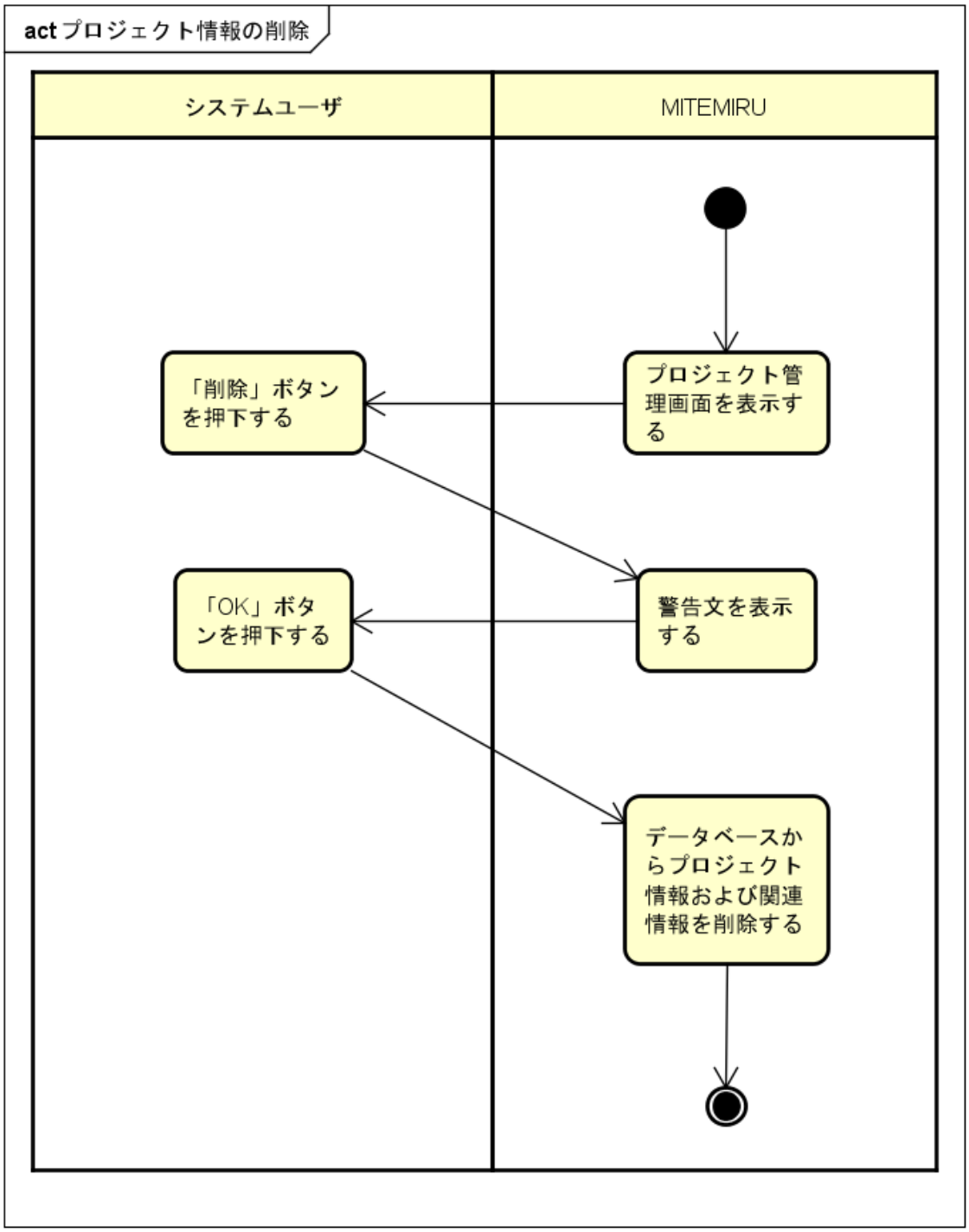


図 B.9: アクティビティ図 (プロジェクト情報の削除)

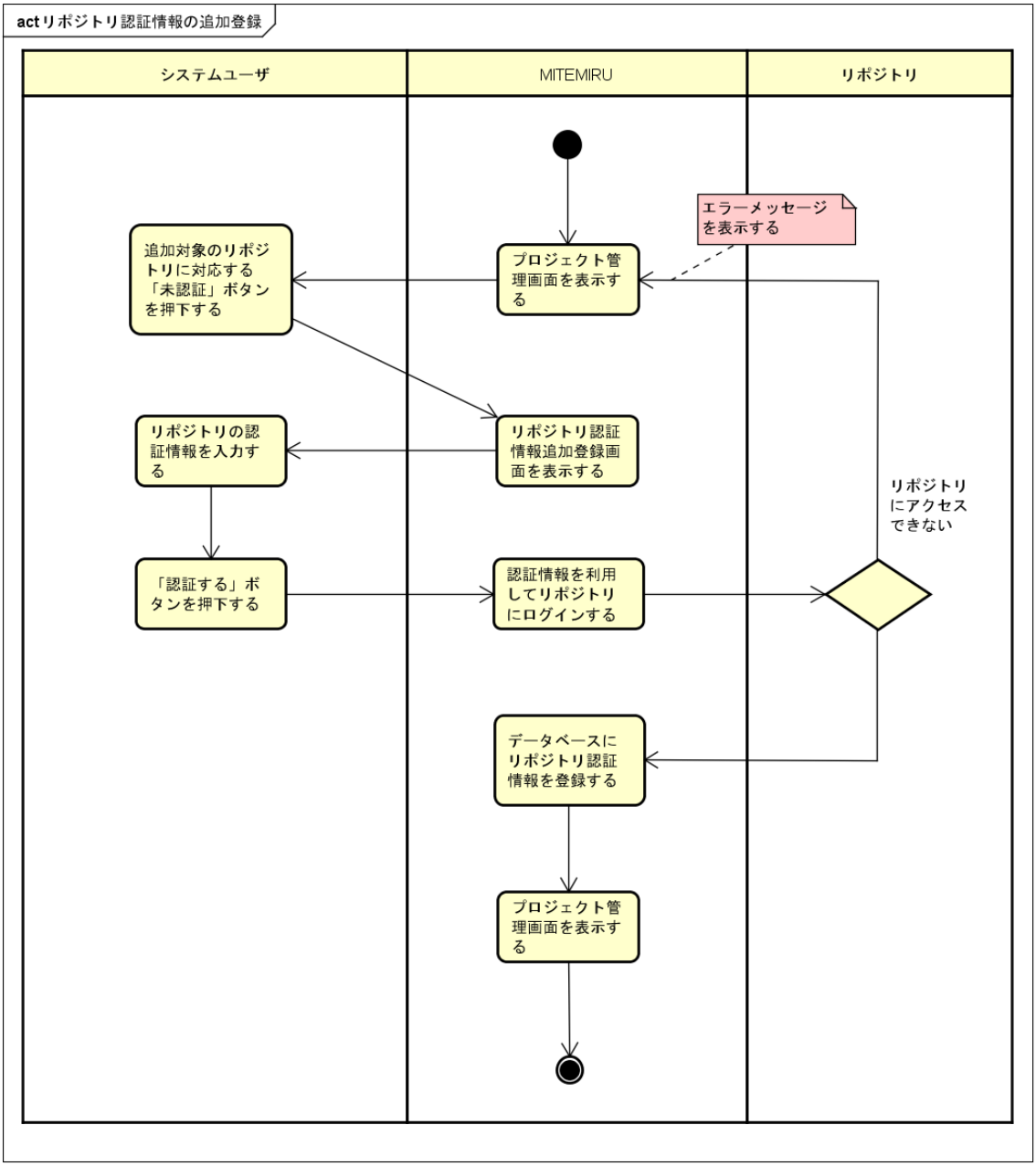


図 B.10: アクティビティ図 (リポジトリ認証情報の追加登録)

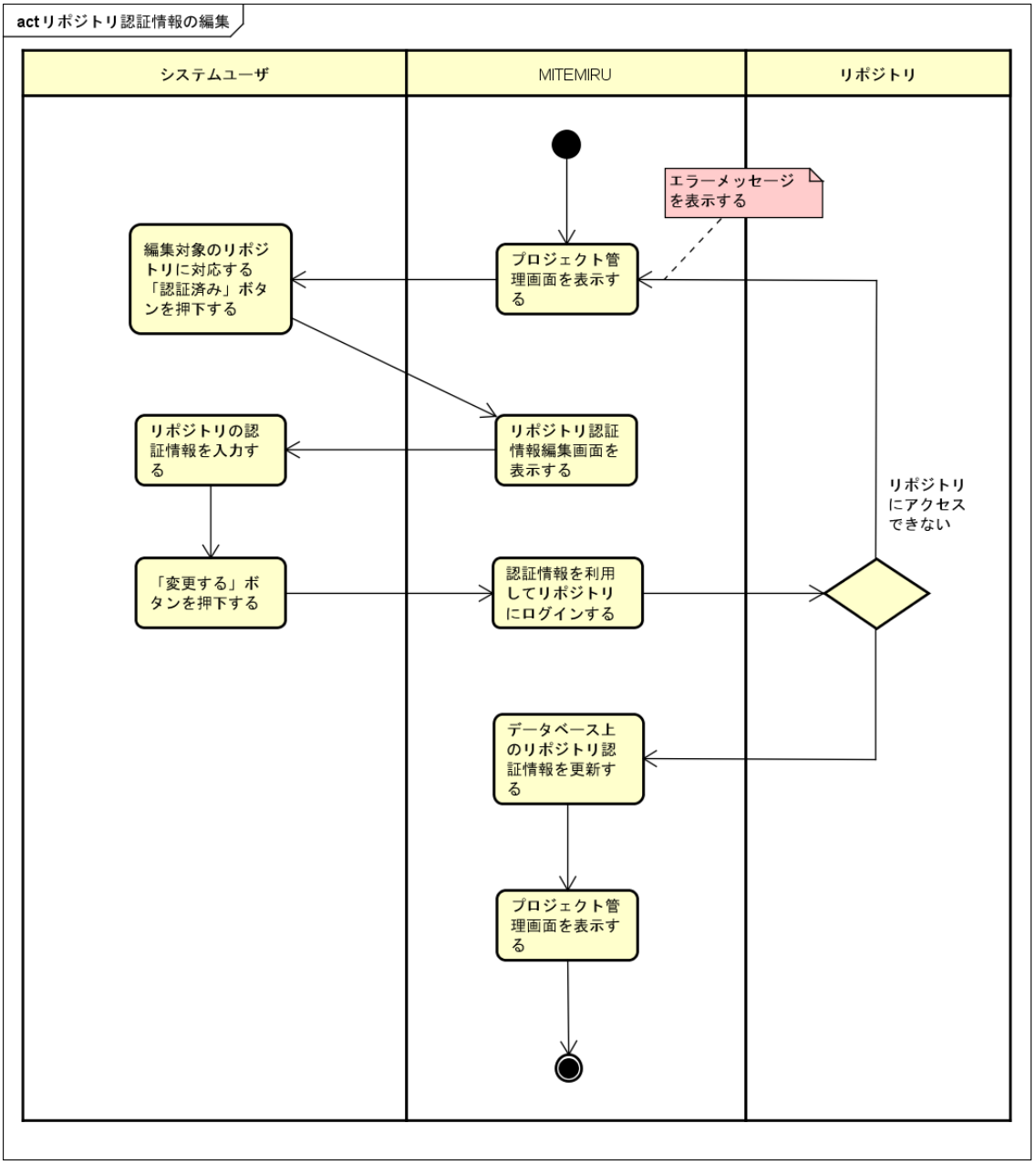


図 B.11: アクティビティ図 (リポジトリ認証情報の編集)

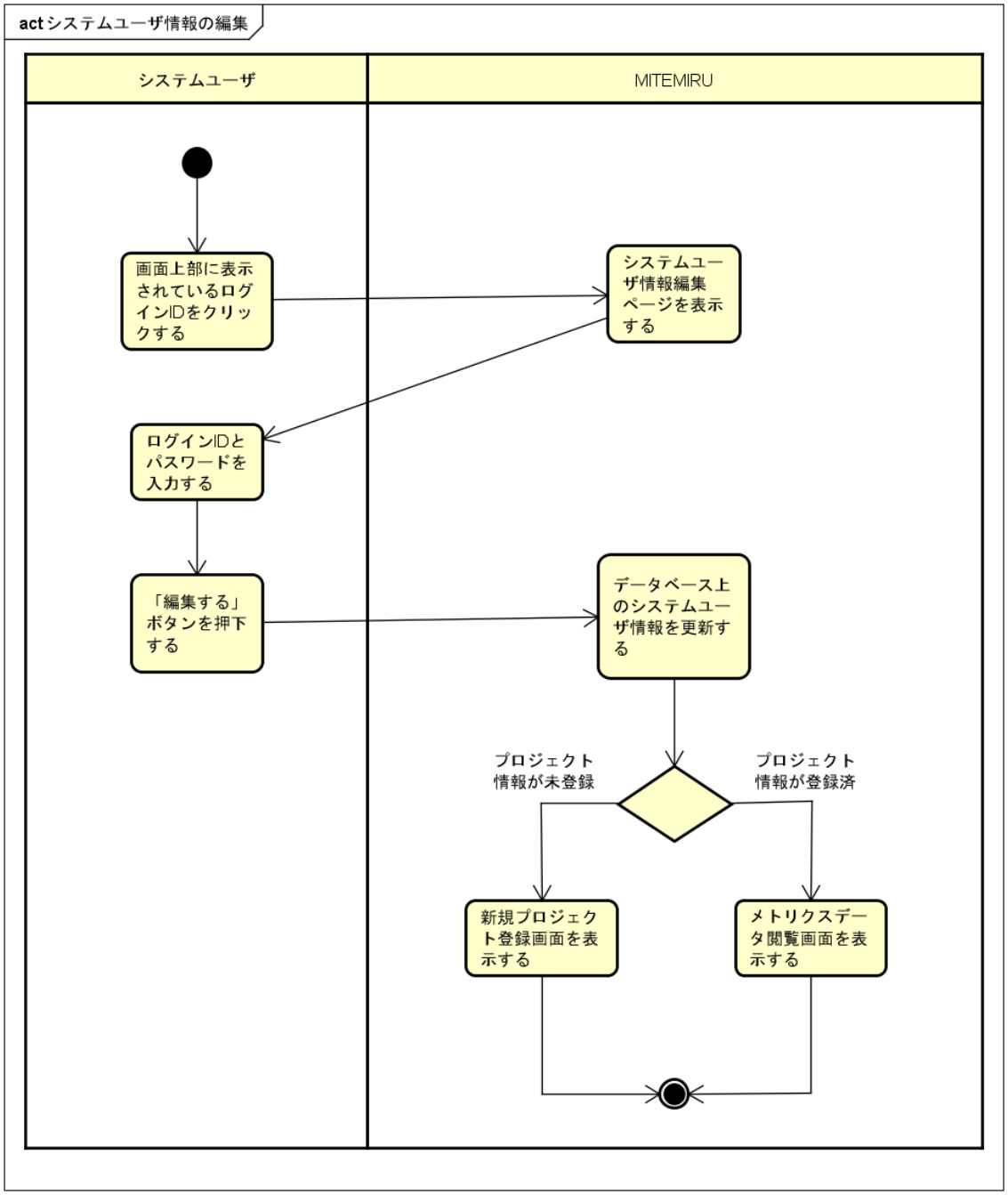


図 B.12: アクティビティ図 (システムユーザ情報の編集)

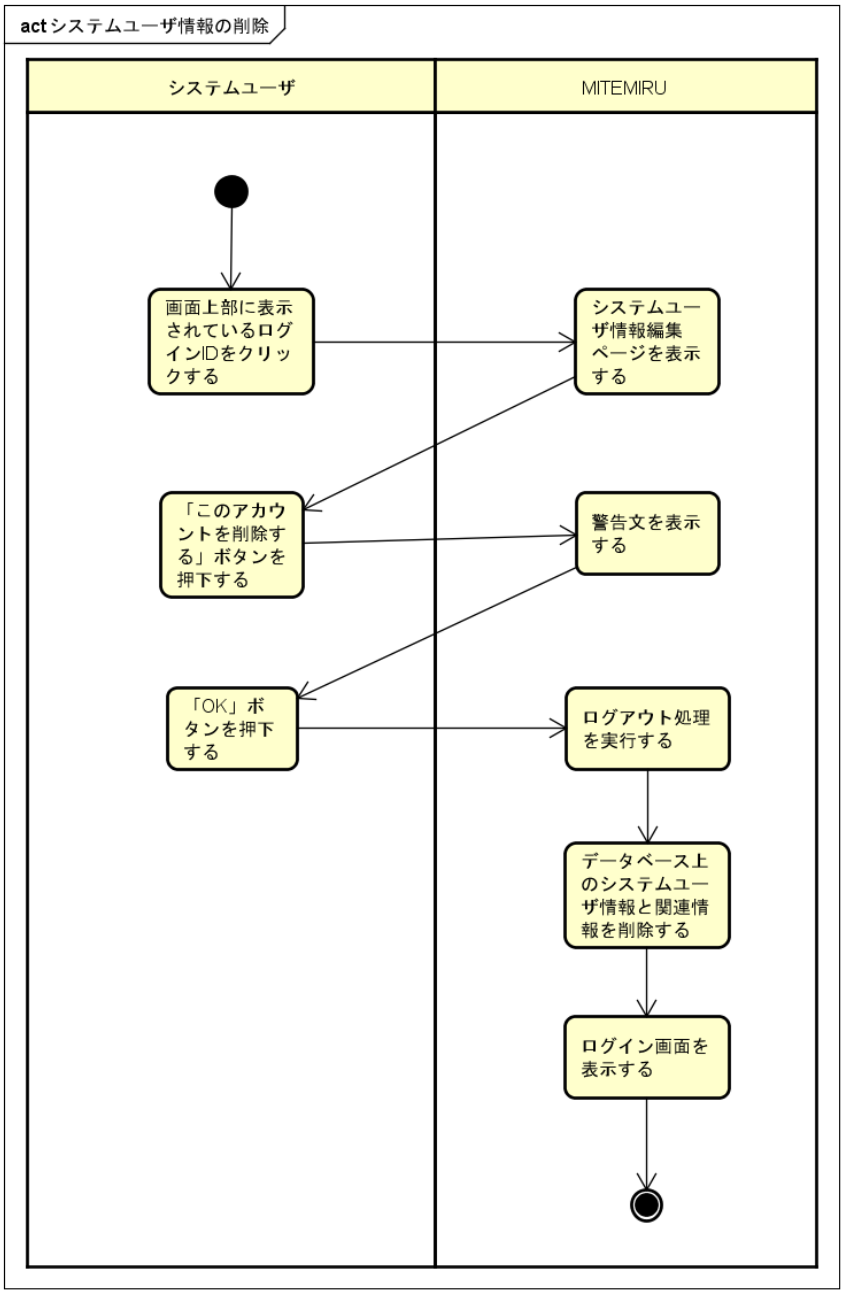


図 B.13: アクティビティ図 (システムユーザ情報の削除)

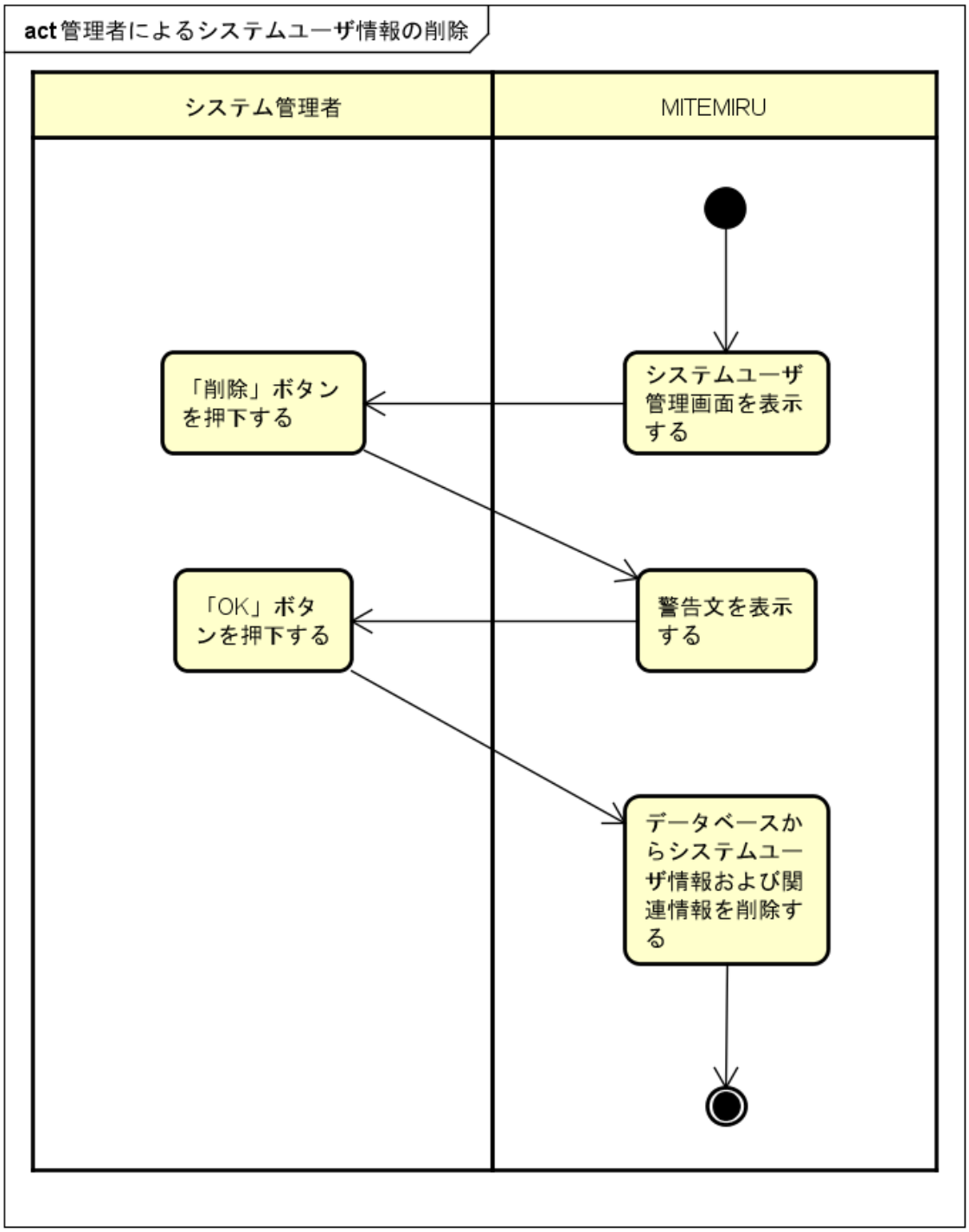


図 B.14: アクティビティ図 (管理者によるシステム情報の削除)