卒　業　研　究　報　告　書

平成23年度

研究題目

Issue Tracking Systemを用いた
文書作成の管理システム

指導教員　　上　野　秀　剛　　助教

氏　　名　　西　畑　　洵

平成24年2月28日　提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

概要

ソフトウェアの開発現場では，ソースコードの変更履歴を管理するシステムや，不具合情報を管理するIssue Tracking System (ITS)を用いた開発情報の共有と管理が広く行われており，これらのシステムを用いた進捗管理を支援する研究も行われている．ソフトウェアの設計書や要求仕様書といった開発文書についても同様のシステムがパッケージソフトウエアとして市販されているが，非常に高額である．オープンソースで開発されたシステムも存在するが，開発文書の編集や手直しといった変化をリアルタイムに管理するシステムは存在しておらず，整合性の保持や編集の必要を把握する必要のある開発文書の管理には不向きである．開発文書の管理において，締切の管理と整合性の保持が必要であると考えられる．そして，締切の管理には開発文書を完成させる為のタスク群の管理，タスクの依存関係の把握，マイルストーン別の進捗管理という3つの管理が必要になる．整合性の保持の為には関連文書修正タスクを生成することが必要になる．そこで，本研究ではオープンソースで公開されているITSを用い，チケットによる管理を行っているものを対象とした開発文書管理手法を提案する．本研究では，要求仕様書などの開発文書に対する変更や，関連する作業の管理を行えるよう，チケットに記録する項目を整理し，チケットを用いた開発文書の管理手法について考える．本提案手法では，締切の管理の為に“チケットによるタスク管理”,“親子関係を持ったチケットの登録”,“マイルストーン別の進捗の管理”を用いた管理を提案している．本論文における整合性とは複数ある開発文書に記述された内容に矛盾や差異が無いことであり，本研究では”関連文書の修正タスクの自動生成”の機能を提案している．そして提案した管理手法を実現するために,“親子関係をもったチケットの登録”,“チケットの親子関係の可視化”,“マイルストーン別の進捗の管理”の機能を実装した．整合性の保持を目的とした“関連文書の修正タスクの自動生成”については未実装となっている．実装した機能について，ユーザ像（ペルソナ）と典型的な振る舞い（シナリオ）を想定して評価を行った．評価にあたり，シナリオとして奈良高専情報工学科の卒業研究を想定し，ペルソナとして学生と指導教員を想定した．卒業研究を対象としたのは，卒業論文やスライドと言った文書には依存関係，編集が複数回に渡るなど開発文書と共通点が多いからである．その結果，必要タスクの進捗管理，タスクの依存関係の把握，マイルストーン別の進捗の管理を行うことができるようになり，締切の管理の支援ができる事を確認した．

目次

1 研究背景

ソフトウェアの開発現場では，ソースコードの変更履歴を管理するVersion Control System(以下VCS)や，不具合情報を管理するIssue Tracking System(以下ITS)を用いた開発情報の共有・管理が行われている．そして，これらにシステムを用いた進捗管理を支援する研究[1]も行われている．それぞれのシステムをどのように利用するかは開発プロセスに依存するが，一般にプロセスは開発組織ごとに異なるため，それぞれの組織に合わせて利用するシステムを選択，または作成する必要がある．多くの場合，オープンソースソフトウェアとして公開されているシステムを改良し，利用することで安価で柔軟性の高い開発環境を構築している．

これらのシステムは，ソースコードを対象としたものだけでなく，設計書や要求仕様書などの開発文書を対象としたものがパッケージソフトウェアとして市販されている．しかし，それらのパッケージソフトウェアは非常に高額であることに加え，開発プロジェクトや開発組織が用いている開発プロセスの要求に合わせた変更が難しい．パッケージソフトウェアの販売元にカスタマイズを依頼できる場合もあるが，常に変更が行われていく開発プロセスに合わせた柔軟な変更は難しく，また高額なカスタマイズ費用が必要となる．これは，開発文書を対象としたパッケージソフトウェアが完全にパッケージとして固定化されて販売されているものではなく，開発プロジェクトや開発組織に応じて変更する必要があるからである．つまり，顧客の要望に応じてそのパッケージソフトウェアに変更を加える必要があり，一度作った同一の者を複数の顧客に販売するという形を取りにくいことが，そういったパッケージソフトウェアが高額になる原因である．

オープンソースソフトウェアとして公開されている開発文書を管理できるシステムは，すでに編集の完了した開発文書を対象としており，編集や手直しといった変化が伴う編集途中の開発文書をリアルタイムに管理するシステムは存在していない．また，開発文書には要求仕様者や設計書など複数の種類が存在しており，それらには一方が変更された時，もう一方も変更する必要があるといった依存関係がある．既存のシステムは，このような開発文書間の整合性保持やそれに伴い発生する作業の管理をすることができない．

そこで，本研究ではオープンソースで公開されているTrac [2]やRedmine [3]といったチケットを用いたタスク管理を行なっているITSをベースとした開発文書の管理システムを提案する．本研究ではITSの内，前述のTracを改良の対象とする．Tracをはじめとした ITSはソースコードの不具修正状況や修正の必要をチケットとして登録して管理する事を目的としたシステムである．Tracは，Webシステムとして実装されており，Webブラウザを通して閲覧する．既存のこのシステムは，ソースコードの不具合とバグを管理の対象にしており，開発文書を管理の対象とはしていない．管理対象がソースコードを対象にしているため，文書間の依存

関係，編集の順序関係を考慮した管理ができないシステムとなっている．そのため，文書の管理を行うには機能が不足していると言える．また，前述のRedmineについても同様にチケットを用いたタスク管理を行なっている為，Redmineに置き換えても提案システムを多少の変更を加えることで実現できると考えられる．

そこで，本研究では開発文書の編集の必要や，開発文書の進捗を管理するシステムを提案する．提案システムを用いる事で，ある開発文書の編集や手直しから発生する他の開発文書の編集や手直しの必要を把握する事ができる．そして，提案システムは開発文書を作成する上で必要になるタスク群の進捗を管理し，開発文書の作成が締切に間に合うように作業を進めるのを支援することができる．提案システムを用いる事で既存の開発文書管理システムでは管理できなかった，開発文書の編集や手直しといった変化をリアルタイムに管理する事ができるようになり，開発プロジェクトを効率的に進めることができるようになる．また，本研究ではオープンソースソフトウェアを用いているので，利用者の要望に合わせて改造することができ，無償で利用する事ができる．

2 開発文書の管理

本研究では, 開発文書の管理について, 締切の管理と, 整合性の保持の 2 つの観点に着目してシステムの要件を定義する. 以降の節でそれぞれについて説明する.

2.1 締切の管理

ソフトウェアの開発過程で作成される文書には, 顧客への提出やプロジェクトの進捗に伴った締切りが存在する. 開発文書の作成が締切を超過すると, その文書を用いるすべてのプロセスが正常に行えず, プロジェクト全体の遅延や品質の低下を招く. また, 開発文書はプロジェクト開始時に決定されたスケジュールに従い, 顧客に随時提出するのが一般的である. その為, 締切の超過は契約違反に伴う金銭的な損失や信用の喪失につながる危険性があり, 文章を管理する上できわめて重要な要素である.

本研究では, 締切の管理に必要な管理対象を以下の3つに分割し, タスクの進捗とその前後関係の管理を行うことで, 締切の管理を行う.

2.1.1 必要タスクの進捗管理

ここでいう必要タスクとは, 文書を完成させる為に必要なタスクのことである. 必要タスクとは, 開発文書の編集において要求仕様書の修正や図の作成といったタスク群のことを意味している. ある仕様書の作成の為に必要なタスク群の例を図1に示す. このガントチャートは, 仕様書作成時に「インターフェイスのプロトタイプの作成」,「実装する機能リストの作成」,「機能とインターフェイスの対応表の作成」といったタスクを実施する必要があることを必要である事を示している. インターフェイスのプロトタイプの作成, 実装する機能リストの作成, 機能とインターフェイス対応というタスクを終了させないと, 仕様書作成という着手することができない事を示している.

これらの必要タスクの管理ができていないと, タスクを見落としてしまい文書を完成できない恐れがあり, もしそれが開発文書の作成にて生じてしまうと, 顧客との仕様決定ミーティングに支障をきたす, 開発の進捗に遅れが出るといった問題が起こる可能性がある.

2.1.2 タスクの依存関係の把握

あるタスクを開始する為には事前に他のタスクを完了させなければならない, といった依存関係が存在する場合がある. 例を挙げると, 要求仕様書の作成タスクを開始する為には事前に, 顧客とのミーティングの内容をまとめる必要がある.

つまり, 要求仕様書の作成タスクは, 顧客とのミーティング内容をまとめるというタスクに依存していると言える. このようなタスクの依存関係を把握できていないと, 要求仕様書の締切の間近になって, ミーティング内容がまとまっていないと言った事が起こってしまい, 文書の完成に支障をきたす. タスクの依存関係の例を図2に示す. これは, ソフトウェア開発の為の開発文書において, 顧客へ提出する為の要求仕様書を作成するというタスクがある場合に依存関係を持ったタスクが存在する事を示している. この場合, 要求仕様書を作成する為に, 顧客の要求を抽出, 要求にインターフェイスについてのものがあった場合プロトタイプを作成して図として示すなどといったタスク群が必要になる. そして,「要求仕様書を作成する」というタスクに着手するには,「顧客の要求の抽出」,「インターフェイスのプロトタイプの作成」と言ったタスク群の完了が必要である. これは「要求仕様書を作成する」というタスクが「顧客の要求の抽出」,「インターフェイスのプロトタイプの作成」というタスクに依存していると言える. そこで, 各タスクの進捗だけではなく, それらの前後関係を管理する必要がある.

2.1.3 マイルストーン別の進捗の管理

マイルストーンはプロジェクト進捗の区切りとなる日付のことで, あるタスクの締切日を兼ねるのが一般的である. 具体的には顧客へのソフトウェアリリース日や, 外部委託先への設計書提出日などが挙げられる. マイルストーンは組織外部との契約や販売戦略に合わせて決定される為, 開発状況に合わせた変更が難しい. マイルストーンは1つのプロジェクトで複数存在し, それぞれを遅延無く予定通りに完了させる為には, 関連するタスクを把握し, 適切に実施する必要がある複数のマイルストーンを同時に管理する必要がある.

2.2 整合性の保持

本研究における整合性とは, 文書間に意味的な差異があるかどうかを示し, 意味的な差異が無い状態を整合性が保たれている状態という. 整合性の保持に着目した理由は, 複数の文書が存在した時に, それらに依存関係がある場合がある.

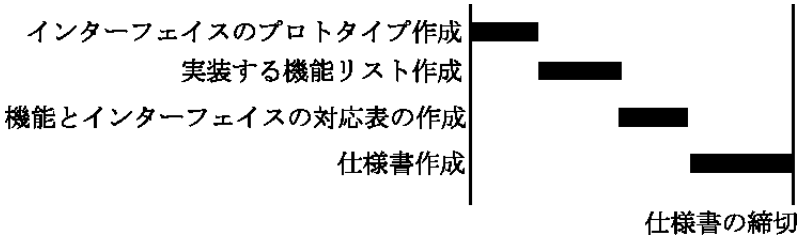

図 1: 必要タスクの例

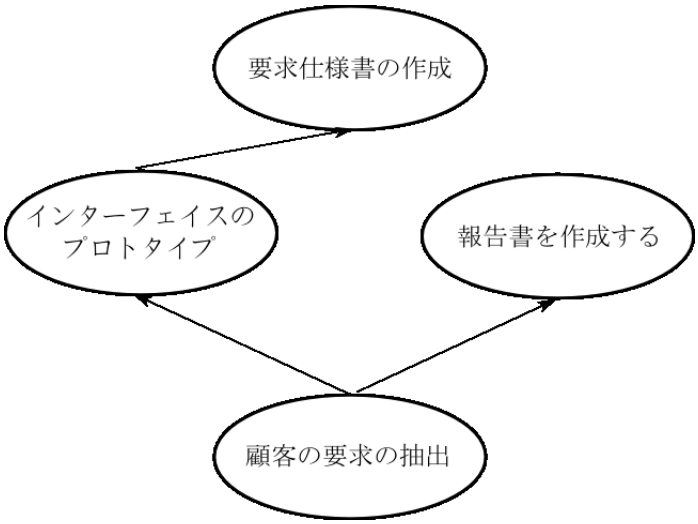

図 2: タスクの依存関係

その時, 依存元に修正が加わった場合に依存先にも修正を加える必要があると考えられる.

例として, 何らかの予約を管理するシステムを開発する場合を想定する. その時, 要求仕様書に「予約を管理したい」と記述されており, それに対応して設計書には「予約を管理する為の関数群の作成」があったとする (図3参照). この場合, 2つの依存関係を持った開発文書間の整合性は保たれていると言える. そして, 顧客が「オンラインで予約を管理したい」と新たな要求を提示した時, 要求仕様書には「オンラインで予約を管理したい」と新たな記述が加わる. それに対応して, 設計書には「オンラインで予約を管理する為の関数群の作成」といったような記述が必要になる. その際に, 要求仕様書の記述に対して対応した記述が設計書に無い時, 要求仕様書と設計書の間の整合性を保てていないと言える (図4参照). この様な状態で開発が進んでしまうと顧客の信頼を失うなどの損失が生まれる可能性があり, 依存関係を持った文書間の整合性の保持の為の管理が必要になると言える.

その様な事例はソフトウェア開発時にも起こると考えられ, 整合性の保持を管理することが重要である. 例を挙げるとソフトウェア開発では要求仕様書に変更が加えられた場合, 元の開発文書を参照して作成された設計書についても変更が必要となる. 卒業研究では予稿が添削された際に対応したスライドを併せて変更する必要がある. 関連を持つ開発文書間で整合性を保てないと開発文書間に矛盾が生じ, 作成されたソフトウェアと要求仕様が一致しない問題が起こると考えられる. 本研究では, 文書間の整合性を保持する為の文書管理手法を提案する.

2.2.1 関連文書修正タスクの生成

提案システムは，関連文書の修正タスクの管理を支援する為に，他の文書と依存関係を持った文書に編集が加えるというタスクが登録された時，依存関係を持った文書に対して編集を加えるというタスクを自動生成する．そして，依存関係をもった文書を編集するタスクがある事を把握する為に，依存先，あるいは依存元のタスクがあるという情報をチケットに持たせる．

2.3 従来プロジェクトにおけるタスク構成の再利用

ソフトウェア開発プロジェクトの多くは新規開発ではなく，機能の追加や不具合の修正を行う拡張開発である．その為，これまでのプロジェクトと同じようなタスクの構成になる可能性が高い．

そこで，本研究では，プロジェクトにおけるタスク構成を再利用する為に，プロジェクトのテンプレート化を提案する．

2.3.1 プロジェクトのテンプレート化

似通ったプロジェクトが繰り返し立ち上げられる場合，過去のプロジェクトで洗い出したタスクの一覧と，その依存関係をテンプレートとして保存し，新たなプロジェクトで利用することでプロジェクト全体の効率化が期待できる．

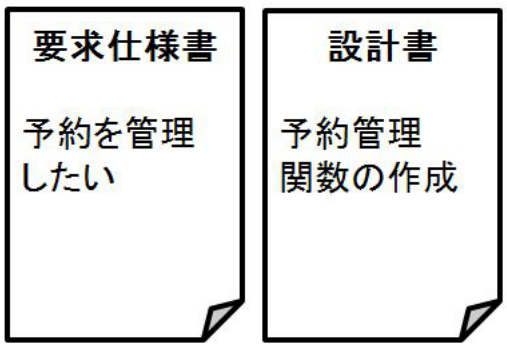

図 3: 開発文書間の整合性が保持された状態

3 提案手法

2章で述べた開発文書の管理における要件を満たすシステムを提案する．提案システムはオープンソースで配布されているITSシステムのうち，チケットによる管理を行うシステムをベースに用いる．チケットとは,「バグを取り除く」,「機能の追加」といったタスクについての様々な管理するために必要な情報を集めたものである．具体的には，重要度，対象となるソースコード，締切や担当者などの情報を管理するための物である．このチケットを用いることでタスクについての情報を一括に管理することが可能となる．チケットを用いたITSとして，TracやRedmineなどがある．本研究では，ITSの一つであるTracを用いて，2章で挙げた問題を解決するための管理手法を提案する．

3.1 締切りの管理手法

2章で挙げた必要な管理2.1.1節〜2.1.3節について，具体的な管理手法を示す.

3.1.1 チケットによるタスク管理

必要タスクの実施漏れを防ぐために，開発文章の作成に必要なタスクをチケットとして登録する．チケットにはタスクの概要や作業対象の開発文書，進捗状況などを入力する．開発者はタスクが発生した際にチケットを登録し，未解決のチケットを管理することで開発文書の作成に必要なタスクを管理することが出来る．

3.1.2 親子関係を持ったチケットの登録

開発者がタスクの管理を行う際に，依存関係を持ったタスクや順序関係が重要なタスクを管理する必要がある．それらのタスクを管理するために，開発者はチケットの親チケット，子チケットの登録項目にそれらを登録してタスクの依存関係を管理する．

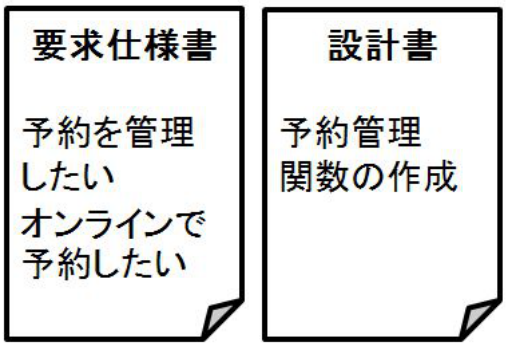

図 4: 開発文書間の整合性が保持されていない状態

3.1.3 チケットの親子関係の把握

開発者が依存関係を持ったタスクや順序関係が重要なタスクを管理する際に，依存関係や順序関係の一覧を見ることで，依存関係のあるタスクや順序関係のあるタスクを把握することを支援出来ると考えられる．

3.1.4 マイルストーン別の進捗の管理

2章で述べたマイルストーン別によるタスクの管理を行うために開発者はチケットの登録の際にマイルストーンの項目を設定する．マイルストーンには，ソフトウェアの顧客へのリリース日，設計書の完成締切りなどが登録される．開発者や管理者は，同時に進行する必要のある複数のマイルストーンの進捗具合を同時に閲覧し，把握することで，あるマイルストーンのタスクに気をとられて他のマイルストーンのタスクがおろそかになるという事態を防ぐ事が出来る．

3.2 関連文書の修正タスクの生成

2章で挙げた整合性を保つために必要な管理について，具体的な管理手法を示す．ここで述べる整合性を保つ必要のある文書がある文書と，整合性を保つ必要のある文書の関係は，文書一つを単位としたものを想定している．関連文書の整合性の保持については，ユーザがTracを運用する際にルールを設けるのと，チケットに拡張を加えることで実現出来る．ソフトウェア開発における開発文書を例に上げる．ソフトウェア開発における開発文書において，整合性を保つ必要のある文書がある開発文書に変更が加わった際に，管理者は変更が加わった開発文書と整合性を保つ必要のある文書を編集するというタスクをチケットとして登録する．この際，開発文書を編集する必要のある人を担当者としてチケットに登録することで，チケットを編集する人に割り当てることができる．こうすることで依存元タスクと依存先タスクの整合性を保つために管理が行えると思われる．

3.3 プロジェクトのテンプレート化

ソフトウェア開発において，開発者や管理者は既存のソフトウェアの機能の追加や不具合の修正を行う拡張開発を行う場合がある．その際，以前の経験を活かすためにプロジェクトのテンプレートを作成するシステムを提案する．

提案するシステムは，プロジェクトのテンプレートの作成を行い，作成したテンプレートから，複数のプロジェクトを作成する事が出来る．よって，ソフトウェア開発現場を例に上げると開発者は以前の経験を元に作成したタスク見積もりをテンプレート化して管理する事が出来る．また，初めて開発に参加する新人の開発者などのタスク配分の目安にする事ができる．例えば，拡張開発を繰り返し

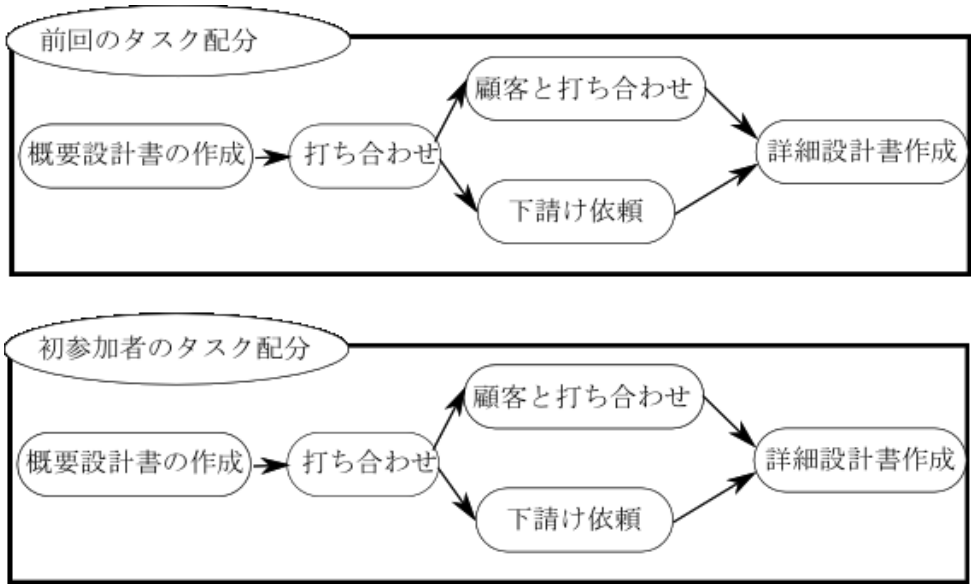

図 5: 拡張開発を初めて開発に参加する人に適用した例

行っているプロジェクトがあった時, 初めて開発に参加する人にテンプレートを適用することでタスク配分をスムーズに行うことができると考えられる図5を例に上げると,“ 前回のタスク配分 ”の様に以前行った開発のタスク配分を参考にしてテンプレートを作成し, 初参加者のタスク配分にそのテンプレートを使用ことで, タスク配分の参考にでき, タスク配分をスムーズに行える.

4 実装

3章で説明した機能を備えたシステムを実装するために，本研究では表1に示した環境を構築した．改良対象のITSはTracを用いた．このTracはチケットによりソースコードのバグの修正タスク，機能の追加などのタスクを管理する機能を提供している．Tracはチケットによるタスクの管理をサポートしているため，3.1.1節で示した手法は既存システムを使用することで実現出来る．

そしてTracのチケットはマイルストーンを登録する機能が用意されているので，3.1.4節で示した手法についても既存システムの利用で実現出来る．また，Tracはプラグインを導入する事で機能を拡張する事が出来，公開されているプラグインを導入する事でいくつかの手法を実現する事が出来る．

4.1 親子関係を持ったチケットの登録

親子関係を持ったチケットを登録するために，チケットの項目に親チケット，子チケットを追加する必要があった．そこで，チケットの項目に親チケットと子チケットの項目を追加するプラグインであるTracMasterTickets[6]をTracに導入した．このプラグインは，子チケットがすべて完了していない親チケットの分類を完了した状態にすることができないようにすることが出来る．また，このプラグインをTracに導入した時のデフォルトの項目名はblockingとblockedbyとなっているので，それぞれ子チケット，親チケットと項目名を変更した．親チケットと子チケットの項目を追加した場合に，Tracのチケット登録画面で出力されるページを図6に示す．チケット登録画面では，チケットの名前，チケットとして登録するタスクの内容や担当者など，チケットとして登録するタスクに関する情報が入力される．図に示したチケットの登録画面のように，チケットの親チケット，子チケットを各チケットに割り当てられたIDを入力することで登録出来る．

この機能を導入することで，2.1.1節，2.1.2節，2.2.1節で示した問題にに対応することが出来る．

表 1: 実装環境

OS	CentOS 5.7
Issue Tracking System	Trac ver0.12.2
HTTP SERVER	Apache2

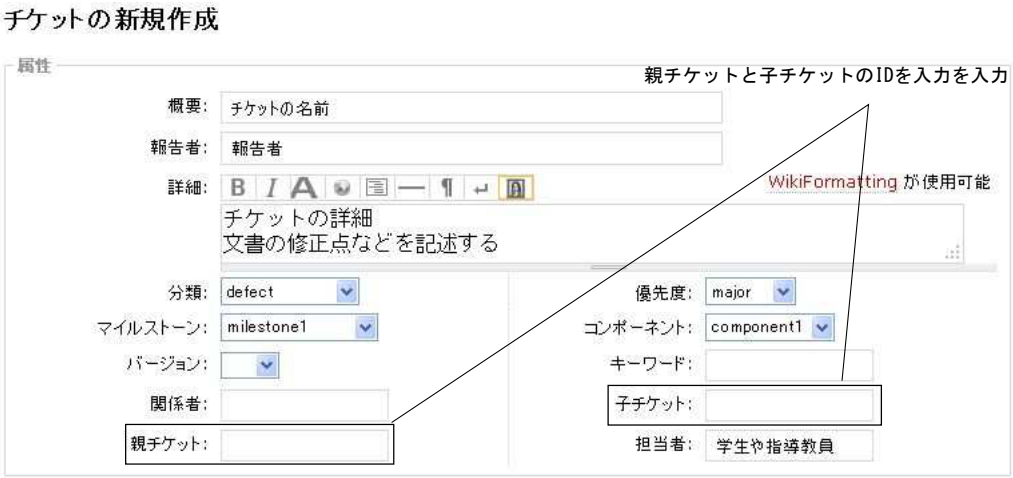

図 6: チケット登録画面（親子関係）

4.2 チケットの親子関係の可視化

3.1.3節で示した手法を実現するために，チケットの親子関係を可視化する必要があると考えられてた．そこで，graphviz [7] と TracTicketDep[8] の 2 種類のプラグインを Trac に導入した．それぞれ，チケットに親子関係をもたせる TracMasterTickets と連携したプラグインであり，TracMasterTickets の機能で追加した親チケットと子チケットの項目に合わせて親子関係の可視化を行うことが出来る．graphviz は Trac に登録された親子関係を持ったチケットの親子関係をグラフとして表示する事が出来るようになる．TracTicketDep は，Trac のチケット閲覧画面において，そのチケットが所属している親子関係を木構造的に表示出来る．これにより，チケット間につながりがなかったものに親子関係が付与され，2.1.2 節で示したの問題に対応出来るようになった．

Trac に登録されたチケットの親子関係を graphviz を用いてグラフ表示している様子を図7に，TracTicketDep を用いて木構造表示している様子を図8に示す．また，図7と図8はチケット ID 番号 4 の持っている親子関係を示すものである．図7は，矢印が指している方が親チケット，矢印が出ている方が子チケットにあたる．それぞれの円は，チケットを示している．図7の場合チケット ID 番号8から着手していくことになる．また，斜線の引かれた円で示されているチケット ID 番号は，図7においてはチケット ID 番号 4 が親子関係を持っているものを示し，灰色で示されているチケット ID 番号は親子関係を持っていないチケット ID 番号を示す．実際のシステムでは斜線部は赤色で表現されている．そして図8は，チケット ID 番号 4「実験の手法を考える」の閲覧画面に表示される木構造の図である．Blocking の項目がこのチケットの親チケット以上のチケットを示し，Blockedby の項目はこのチケットの子チケット以下のチケットを示している．

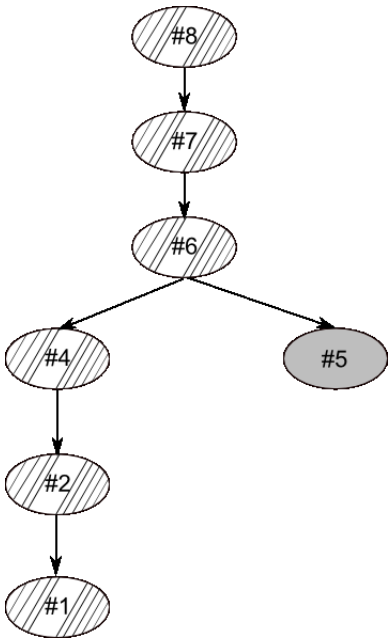

図 7: graphviz によってグラフ化された親子関係

Blocking

Id	概要	マイルストーン
#4	└ 実験の手法を考える	
#2	└ 実験	卒論締切り
#1	└ 卒論作成	卒論締切り

Blocked by

Id	概要	マイルストーン
#4	└ 実験の手法を考える	
#6	└ 研究対象の調査	
#7	└ 研究に近い論文読む	
#8	└ 研究テーマ決める	中間発表

図 8: TracTicketDep によるチケットの木構造表示

4.3 マイルストーン別の進捗の管理

マイルストーン別のタスク管理は, 本研究で使用しているシステムである Trac が標準で搭載している機能である. よって 2.1.3 節で示した問題は解決出来る. Trac でチケットをマイルストーン別に閲覧した際の表示画面を図 9 に示す.

4.4 従来プロジェクトにおけるタスク構成の再利用

4.4.1 プロジェクトのテンプレート化

Trac にはプロジェクトのテンプレートを作成する機能は無く, 公開されているプラグインにもそういった機能を持ったものは存在しない. そこで本研究では新

Milestone 全国大会発表 (2 件が該当)

Ticket	概要	コンポーネント	バージョン	分類
#6	発表スライド作る	スライド作成		defect
#7	論文書く	論文		defect

Milestone 卒業研究発表会 (3 件が該当)

Ticket	概要	コンポーネント	バージョン	分類
#4	実験	実験		defect
#5	分析	実験		defect
#3	卒業論文書く	論文		defect

Milestone 高専カンファレンス (1 件が該当)

Ticket	概要	コンポーネント	バージョン	分類
#8	スライド作る	スライド作成		defect

図 9: マイルストーン別の表示画面

たにTracのプロジェクトのテンプレートを作成，また，テンプレートからプロジェクトを作成する機能を追加した．作成されるテンプレートはユーザが最初にテンプレート名を指定し，その後テンプレートに考えられるタスクをチケットとして登録していくことでテンプレートの作成を行なっている．作成されたテンプレートは，そのテンプレートを参照してユーザがプロジェクトの名前を登録することでプロジェクトとして利用できる．このテンプレートの使用法として例を挙げると，指導教員が各研究室ごと，あるいは研究分野などでテンプレートの作成を行い，チケットを登録する．そして，学生か指導教員がそのテンプレートから各学生用のプロジェクトを作成する．

本研究で作成したテンプレート作成機能とテンプレートからのプロジェクト作成機能は，ターミナルからコマンドを打ち込むことによって実行される．

テンプレート作成用のコマンドを図10に示す．これは，テンプレート作成用のコマンドを使用した際の例である．コマンドの後に作成したいテンプレートの名前，Tracのプロジェクトを格納するディレクトリを指定する事で，コマンドを実行できる．本研究のプロジェクトのテンプレート化におけるテンプレートの作成では，ユーザがチケット，導入されているプラグインの有効，無効を設定できる．このコマンドを実行すると，まず新しいTracのプロジェクトが作成され，簡易的なApacheサーバであるtracdが起動する．その後，ユーザは指定されたURLにアクセスし，表示されたプロジェクトのページでテンプレートとして登録したいチケットを登録し，プラグインの設定を行う．そして作成されたテンプレートからプロジェクトを作成するコマンドを図11に示す．これは，図10に示したテンプレート作成用コマンドによって作成されたテンプレートからプロジェクトを作成するコマンドである．引数として，引用したいプロジェクトのテンプレートのディレクトリ，プロジェクトを作成したい場所，作成したいプロジェクトの場所を

```
#template-create テンプレート名 tracプロジェクトのホームディレクトリ
# ./template-create.sh temp /trac
テンプレートを作成します
tracdを立ち上げます
しばらくお待ちください
http://127.0.0.1:8080/にアクセスしてプロジェクトを編集してください
Server starting in PID 392.
Serving on 0.0.0.0:8080 view at http://127.0.0.1:8080/
Using HTTP/1.1 protocol version
```

図 10: テンプレートの作成

```
#trac_create プロジェクトテンプレートの場所　作成したい場所　プロジェクトの名前
#./trac-create.sh /trac/template/temp /trac/pro
```

図 11: テンプレートからプロジェクトの作成

指定する. このコマンドを実行すると, プロジェクトのテンプレートを元にしたプロジェクトを指定した場所に作成する.

5 評価

第4章で説明した実装についての評価を行う. 本論文では, 評価にあたり実際のソフトウェア開発現場においてこのシステムを適用すると数ヶ月かかってしまい, 卒業研究の期間で評価を行うのは難しい. そのため, 卒業研究における卒業論文や発表用スライドといった文書の進捗管理を対象とする. 卒業研究における文書を対象としたのは, 卒業論文や発表用のスライドなどは何度も変更される点, 予稿とスライドのように依存関係がある点, そして修正などの編集が複数回に渡り行われるといった点が開発文書に似ているからである. これらの点が提案システムの評価に適していると考えられた.

システムを評価するには開発プロジェクト, あるいは卒業研究の期間を通してシステムを運用する必要があるが, 今回はそれが困難である. そこで, 提案システムの主な利用者である学生と指導教員について, 典型的な振る舞い（シナリオ）と, ユーザ像（ペルソナ）を想定し, 提案システムの機能がどのように役立つかを考察する[4]. 評価において, ペルソナがシナリオに基づいてシステムを利用する事を想定し, システムを利用することで2章で挙げた管理に必要と考えられた点をどの程度サポートできるかを考察する. 今回想定したペルソナである学生については, 著者自身が卒業研究を行う上で必要だったタスクについて考察した. 指導教員については自身の指導教員と議論し, 学生のタスクの進捗の把握など, 学生の卒業研究を指導する上でどのような問題があるかについて調査した.

本研究では, 奈良高専情報工学科の卒業研究を想定する. ユーザは学生と指導教員を想定する. 期間は4月から翌年2月, 1つの研究に対して参加する学生は1人とする. この学生は文献調査, 発表用スライドの作成, 卒業論文の作成, 実験, 実験結果の分析, 予稿の作成を行う事を想定している. そして10月に中間発表, 1月に卒業論文の提出, 2月に最終発表を行う事を想定し, 中間発表と最終発表には予稿と発表用スライドが必要となるものとしている. 指導教員は複数人の学生の指導を行い, 学生から提出された発表用スライドや予稿, 論文の添削を行う事を想定する.

5.1 締切りの管理

5.1.1 学生

学生は卒業研究中に中間発表の予稿とスライド, 卒業論文の作成を行い, そのために実験, 分析などを行う. そして, 卒業論文の作成を行う上で, 実験, 分析などのタスクを完了させる必要がある（2.1.1節）. このようなタスクを管理するのに, 実装したシステムの親子関係を持ったチケットの登録機能が役に立つ. 卒

業論文を親チケットとして，実験や分析と言った卒業論文を作成するのに必要であり，卒業論文を作成するタスクに着手する前に必要のあるタスクを子チケットとして登録する．この場合，分析や実験を終わらせなくては卒業論文を書けないので，分析や実験が子チケット，卒業論文が親チケットになっている．そして子チケットの進捗がどの程度進んでいるかを閲覧し，進捗管理を行う事でいつから親チケットである卒業論文に着手する事ができるかが予測できるようになる．もしその日程に無理があれば子チケットの完成を急ぐなど，時間配分を調整する事ができる．

またこの学生は，登録されている子チケットから着手していく事になる．その時，学生は卒業論文を完成させるのに必要なタスクが複数あり，それらに着手する順番を考える必要が出る (2.1.2節)．例えば，実験をしないで分析を行う事はできない，実験をしないといけない時期になって実験手法が間違っていないかを指導教員に相談しに行っていたのでは遅い場合がある．このようなタスクの着手順序を管理する必要がある時，提案システムの「チケットの親子関係の可視化」が役に立つ．たとえば卒業研究の達成に実験が必要だと分かった時，学生は「実験」と「実験の手法を考える」というチケットを登録する．この時,「実験」というタスクを親,「実験の手法を考える」というタスクを子としてチケットを登録する（図12参照）．すると,「実験」のチケットを閲覧した時，図13の様に，親子関係が可視化されているので，タスクの着手順序がわかりやすい．

そして学生は，卒業論文を書くというタスクに着手するためのタスクを完成させ，卒業論文を書き始める．何度か卒業論文を指導教員に添削して貰いながら更新させていくが，卒業論文だけでなく，卒業研究発表会で用いる発表用スライドの作成も行う必要が出てくる (2.1.3節)．5年生が卒業研究発表会で発表用スライドを用いるのは自明と言えるので，このタスクのチケットは早くから登録しておく事ができ，マイルストーンは卒業研究発表会になる．そして，進行中の卒業論文を編集するチケットのマイルストーンは，卒業論文の締切りになる．この2つのチケットのマイルストーンは違うものの，日付は近い．もし学生が卒業論文の編集にとらわれ，スライドの編集を怠っていた場合，卒業研究発表会で発表することができなくなってしまう場合がある．提案するシステムにはマイルストーン別にチケットを閲覧する機能があり(4章　図9)，このような問題を解決できる．

5.1.2 指導教員

指導教員は，学生が卒業研究を進める上で論文や予稿の添削，実験の手法におかしな点はないかの確認，発表練習の計画などを行う．そして，指導教員は学生が卒業研究を進めていく過程で，発表練習や添削のための締切りなどをもうけるために学生の進捗を把握する必要がある（2.1.1節）．その時，提案システムの「親子関係を持ったチケットの登録」を用いて学生が登録した子チケット群の各進

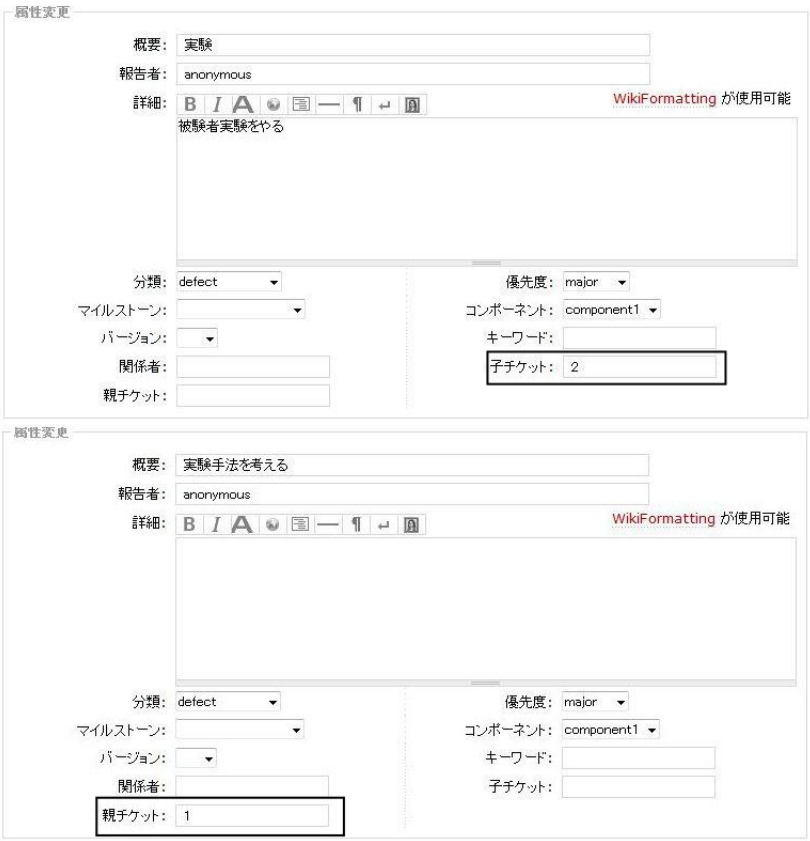

図 12: チケットの登録

捗具合を見る事で，学生がどの位の早さで親チケットを完成させるかを見る事ができる．これによって，指導教員は進捗のない学生に対して指導を行ったり，発表練習などの日程を決めるための目安にする事ができる．

また，指導する学生が卒業研究の経験のない5年生の場合，あるマイルストーンを達成するためにどのようなタスク群が必要かの予測が甘い，あるいはそのようなタスク群がマイルストーンまでに達成するための時間配分が甘いなど，非現実的なスケジュールとなっている場合がある．その時，学生にそれぞれチケットを登録させる事で学生のタスクの見積もりを把握する事ができ，事前にトラブルを回避するように指導できる．

また，学生が卒業研究発表会や中間発表などで発表するのに必要な資料を完成させる事ができそうかを把握する必要もあり（2.1.3節），これは提案システムのマイルストーンごとのチケットの閲覧機能を用いる事で学生のタスクの進捗を把握する事ができる．また，複数のマイルストーンが存在する時，中間発表などの発表練習や，論文の添削のための締切など，タスク群の締切を決めるのにも役にたつ．

図 13: チケット閲覧画面

5.2 文書間の整合性の保持

5.2.1 学生

卒業研究中，学生は依存関係を持った文書をたびたび編集する事になる．中間発表用の予稿と，スライド，卒業研究発表会用のスライドと卒業論文，予稿といったものがそれに当たる．それらの間に意味的な差異が生まれてしまうと，文書間に矛盾が生じてしまう（2章）．そういった問題を防ぐために，提案システムの関連文書の修正タスクの生成で示した手法を用いる．学生は指導教員に予稿や論文の添削を依頼し，添削後のそれらが返却されると学生は対応した文書に変更を加えるタスクをチケットとして登録する．その際に，チケットの親子関係の項目を用いて文書間の依存関係を登録する事で，4章　図7の様にそれを可視化する事ができるため，整合性を保つ必要のある文書を編集するチケットが存在していること把握できる．

5.2.2 指導教員

指導教員は，学生の卒業研究の指導を行う中で，予稿や論文と言った文書の添削を行う時がある．卒業研究において，あるマイルストーンに対して予稿や論文が単独で存在しているという事は少なく，スライドなどの整合性を保つ必要のある文書が存在する．指導教員は，そのような整合性を保つ文書の存在する文書の添削を行い，内容や表現に修正を加える．そして学生に添削後の文書を返却する．その時に指導教員は提案システムを用いて学生に対して整合性を保つ必要

のある文書の修正を行う旨のチケットを割り当てる．そうすることで，整合性を保つ必要のある文書間に差異が生まれることを防止するのを指導教員側からサポートする事ができる．

5.3 テンプレート化による文書作成管理の効率化

5.3.1 学生

卒業研究を始めて行う5年生は，卒業研究の完成のためのタスク配分の目安が付かない場合がある．また，タスクの進捗見積もりが甘いなどが原因で最終的に目指していた目標に到達できない，卒業研究が完成しないといった事が起こることがある．そのような場合，提案システムのテンプレートからプロジェクトを作成する機能を用いる事で，タスク配分の目安をつける手助けにする事ができる．また，専攻科に進学するなどして卒業研究を引き続き行ったり，以前に卒業研究の経験がある学生は自分の過去の経験からテンプレートを作成することで，今後のタスク配分の目安にする事ができる．また，毎回同じようなタスクを繰り返す場合など，チケットの登録の手間を省いたり，チケットを登録し忘れるといったトラブルを防止する事ができる．

5.3.2 指導教員

指導教員は，毎年新たに卒業研究を行う5年生を迎えることになる．そして卒業研究に関しての指導を行うが，学生がタスクをどの様にして管理しているかを把握し，それに誤りやタスクの進捗見積もりが甘いかを把握するのには手間がかかり，困難な場合がある．そのような場合，指導教員は研究室ごとに簡単なタスク見積もりを行なっているプロジェクトのテンプレートを作成し，そのテンプレートから各学生向けのプロジェクトを作成する．そして，各学生向けに作成されたプロジェクトを用いて学生にタスクの管理を行う様に支持する．そうすることで，学生のタスクの進捗見積もりが甘いかを把握する際に，大まかなタスクの見積もりはすでにテンプレートとして配布しているため，学生に新たに発生したタスクの見積もりを把握するだけで済む．また，各研究室である程度決まっているタスクや輪講の形式に合わせたタスク登録，工業外国語などと兼ね合いから研究室ごとにカスタマイズできるので，学生に振られるタスクを割り当てるのを簡略化できる．

6 まとめ

本研究は，開発過程で作成される開発文書をリアルタイムに管理するシステムを作成する事を目的とし，開発文書のリアルタイムな管理を行うシステムを提案した．提案システムは，「締切の管理」と「整合性の保持」を支援することで，開発文書を締切までに完成させるための進捗管理の支援と，複数存在している開発文書間の整合性のを保つ事が出来る．提案システムを実装するにあたり，ITSの一つであるTracをベースにし，Tracのチケットに拡張を加えるプラグインを用いて親子関係を持ったチケットの登録，その親子関係の可視化をおこなった．そして新たに，従来のプロジェクトにおけるタスク構成の再利用を行うため，プロジェクトのテンプレート化を行った．そして作成したシステムに対して評価を行った．本論文では，実際のソフトウェア開発現場にシステムを適用しての評価が難しかったため，システムに対して典型的な振る舞い（シナリオ）と，ユーザ像（ペルソナ）を想定し，提案システムの機能がどのように役立つかを考察した．

その結果，必要タスクの進捗管理，タスクの依存関係の把握，マイルストーン別の進捗の管理を行うことが出来るようになり，締切の管理の支援が出来る事を確認した．そして，関連文書の修正タスクの把握，プロジェクトのテンプレート化によるタスク登録の簡略化が行えることを確認した．

今後の展望として，本論文中で出来なかった実際の卒業研究，ソフトウェア開発の現場においての提案システムの評価が必須と言える．そして，マイルストーンに拡張を加える事があげられる．マイルストーンは本文中に記したように大きな区切りや締切りを示している．卒業研究やソフトウェア開発において，マイルストーンだけでなくそれに向けて細かな締切りが設けられる事が多い．卒業研究で例を挙げると，中間発表というマイルストーンに対して中間発表の発表練習が行われる事があり，それにむけてのスライドを用意する必要がある．この発表練習に向けてのスライドのマイルストーンは中間発表になるが，実質的な締切りは発表練習日ということになる．これでは，締切りの管理を確実に行える事にはならない可能性がある．そこで，マイルストーンに親子構造をもたせ，中間発表に向けた発表練習などを子マイルストーンに，中間発表を親マイルストーンに設定することで，マイルストーンに向けた細かな締切りの管理が行えるようになると言える．他にも，プロジェクトにも親子的な階層構造を持たせる事も必要と考えられる．卒業研究において例を挙げると，現在は研究室内の各学生にたいして一つ一つプロジェクトが割り当てられており，各プロジェクトは独立している．しかし，共同研究を行う場合や研究室内で情報の共有，指導教員が各学生の進捗状況を比較する場合など，研究室全体としてのプロジェクトが必要になると考えられるからである．そして，整合性の保持のためのシステムとして，整合性を保つ必要のある文書がある文書に編集を加えるというチケットが登

録された際に，整合性を保つ必要のある文書についても編集を加えるというチケットを自動登録する機能の実装が必要と考えられる．このような機能を実装することで，チケットの登録漏れを防ぐことができ，整合性の保持を支援できると考えられる．

参考文献

[1] D. Kawrykow, M. P. Robillard, “ Non-Essential Changes in Version Histories ”, ICSE ’11 Proceeding of the 33rd international conference on Software engineering ,pp. 351-360 (2011)

[2] Trac Open Source Project,
“ trac integrated SCM & project Management ”,
http://trac.edgewall.org/,2012/01/25(確認)

[3] nanoc,“ Redmine.JP ”,http://redmine.jp/ ,2012/01/25（確認）

[4] 樽本徹也,“ユーザビリティエンジニアリング　ユーザ調査とユーザビリティ評価実践テクニック ”

[5] 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・センター編,
”共通フレーム2007〜経営者, 業務部門が参加するシステム開発及び取引のために〜”

[6] “ TracHacks MasterTicketsPligin ”,
http://trac-hacks.org/wiki/MasterTicketsPlugin(2012/02/22確認)

[7] “ TrackHacks Graphviz Plugin ”
http://trac-hacks.org/wiki/GraphvizPlugin（2012/02/22確認）

[8] “ TicketDepPlugin ”
http://www.saunalahti.fi/~kankkuri/trac/ticketdep/（2012/02/24確認）

謝辞

本論文の執筆及び研究をすすめるにあたり, 多くの方々にご協力をいただきました. この場を借りてお礼を申し上げます. ありがとうございました.

指導教員である上野秀剛助教には, お忙しい中論文や予稿のチェックを行なっていただき, 的確なご指導をいただきました. 卒業研究発表会などの発表用のスライドについてのご指摘も数多くいただきました. また, 知識や技術など至らないところの多い私に多くのご教授をいただき, 多くの教養を得ることが出来ました. この場をお借りして感謝の意を称したいと思います, 本当にありがとうございました.

中間発表でご質問を頂いた内田眞司准教授には, 提案システムのカスタマイズ性についてのご指摘をいただき, 研究を進める際に大変参考にさせていただきました. ありがとうございました. この場を借りて, お礼を申し上げます.

卒業研究発表会でご質問いただいた2名の先生方にもこの場をお借りしてお礼を申し上げます.

松尾賢一教授には, 提案システムの評価の妥当性についてご指摘いただき, ありがとうございました.

内田眞司准教授には, 提案システムをソフトウェア開発時の開発者に適応した際, 各開発者のプロジェクトを管理者が管理できるのかというご指摘をいただき, ありがとうございました.

同研究室の先輩, 同輩の皆様には卒業研究発表会や中間発表のスライドに関して数多くのご意見をいただき, 大変参考にさせて頂きました, ありがとうございました.