ISSN 1349-1113
JAXA-RR-15-003

宇宙航空研究開発機構研究開発報告

JAXA Research and Development Report

CODA: JSS2の運用・ユーザ支援を支える チケット管理システム –Redmineの事例と利用のヒント–

木元　一広

2015年12月

宇宙航空研究開発機構

Japan Aerospace Exploration Agency

CODA: JSS2 の運用・ユーザ支援を支えるチケット管理システム

– Redmine の事例と利用のヒント –

木元　一広 [*1]

CODA: Ticket Management System to Support JSS2 Operation and Assistance to Users

– Redmine Implementation and Hints of Its Usage –

Kazuhiro KIMOTO[*1]

Abstract

Redmine is an excellent ticket-management-system software for various purposes, one of the OSS which is getting more attention recently. Supercomputer Division of JAXA has been constructing and running CODA system based on Redmine since 2014, when installation of JSS2 SORA Super Computer system was started. This paper introduces CODA system as an example of Redmine implementation. This paper also discusses the hints and tips of definition, setting and operation of Redmine for better use, based on the experience of CODA.

Keywords: Redmine, JSS2, CODA, ticket-management-system, project management software, チケット管理システム，プロジェクト管理ソフトウェア

概要

Redmine はさまざまな業務に利用できる優れたチケット管理システムで，近年注目されている OSS の一つである．JAXA スーパーコンピュータ活用課では，2014 年の JSS2 SORA スーパーコンピュータ導入を機に Redmine をベースにした CODA システムを構築・運用している．本稿では，Redmine の利用事例として CODA を紹介する．合わせて，Redmine を一層効果的に活用するため，CODA の構築・運用経験から見いだされた定義や設定，運用の工夫を紹介する．

1. はじめに

国立研究開発法人宇宙航空研究開発機構（以下，JAXA）では，JAXA の二世代目のスーパーコンピュータシステムである JSS2（JAXA Supercomputer System Generation 2）-SORA（Supercomputer for earth Observation, Rockets, and Aeronautics）[1] が 2014 年 10 月に稼働開始した．2015 年 4 月には第 2 期運用として主要な計算資源である SORA-MA（Main System）が加わり，本格的な利用が進んでいる．

JAXA スーパーコンピュータ活用課は，JSS2 の運用及びさまざまなユーザ支援活動を行っている．スーパーコンピュータ活用課では，JSS2 の導入を機に CODA（CODA is the Operation and Development Assistant）と呼ぶチケット管理システムを構築した．現在，CODA は運用や支援のための情報共有・進捗管理に使用されており，スーパーコンピュータ活用課の活動に不可欠の存在となっている．CODA は，オープンソースソフトウェアのチケット管理システム・プロジェクト管理ソフトウェアである Redmine をベースにした業務管理アプリケーションである．

本稿は全体を通じて，今後 Redmine のようなチケット管理システムの導入検討や構築・設定をされる方々に有益な情報を提供することを目的とする．まず，Redmine の特徴を概観し，次いで，スーパーコンピュータ活用課でのチケット管理ステムの経験と課題及び CODA の導入経緯，利用状況を紹介し，その利点や業務での活用について述べる．

*　平成 27 年 10 月 6 日受付（Received 6 October, 2015）

*1 セキュリティ・情報化推進部 スーパーコンピュータ活用課
(Supercomputer Division, Security and Information Systems Department)

2016/05/31 巻末に正誤表があります。 There is a corrigendum in the last page.

更に，Redmine の導入・設定での独自の工夫やヒントを紹介する．これは，CODA の構築・運用経験から見いだされたもので，既に Redmine を利用している方にも参考になる実践的な内容である．最後に，今後の CODA 及び Redmine の展望を述べる．

2. Redmine の概要

Redmine は，http://www.redmine.org/ で開発・公開されているオープンソースソフトウェア（OSS）である．一般的には，プロジェクト管理ソフトウェアに分類されることが多い．また，チケット管理ソフトウェアに分類されることもある．

2.1 Redmine の利用環境

Redmine は，Ruby on Rails で開発された Web ベースのクライアント・サーバー・アプリケーションであり，主要な Unix の他，MS Windows や Mac OS X 上にサーバーを構築して使用できる．サーバー側の主な前提ソフトウェアは，RDB（MySQL 等），HTTP サーバー（Apache 等），Ruby 及び Ruby on Rails である．クライアント側は Web ブラウザを使用する．JavaScript をサポートする Firefox, Chrome, Safari, Internet Explorer 等の Web ブラウザが利用可能である．

2.2 Redmine の開発と利用の状況

Redmine の開発は非常に活発である．概ね 4-5 ヶ月毎にバージョンアップが行われており，世界中のユーザからの障害報告や機能追加要望が積極的に取り込まれている．2015 年 2 月には多くの機能強化がされたバージョン 3.0 がリリースされている（本項執筆現在の最新リリースは 3.1.0）．

Redmine を開発・バグ管理に利用している有名な事例としては，Ruby の開発管理（Ruby Issue Tracking System, https://bugs.ruby-lang.org/）がある [2]．また，Redmine 自体の開発・リリースや関連するディスカッションなども http://www.redmine.org/ で Redmine を用いて管理されている．

Redmine は日本国内でも IT 開発・管理部門を中心に最近とみに注目が集まっているソフトウェアの一つである．OSS なので正確な利用者数やシェアの把握はできないが，日経 SYSTEMS 誌 2013 年 6 月号の開発支援ツール徹底調査のアンケート [3] によれば，プロジェクト管理ツールでの導入シェアでは 22.3% で 1 位の Microsoft Project（26.9%）に次ぐ 2 位であり，調査当時の直近 2 年間の導入に限ると Microsoft Project（6.2%）の倍以上の 1 位（15.3%）である．また，Redmine に特化した日本語の書籍・ムック類の発行が相次いでいる．

日本語翻訳が充実していることも Redmine の特徴の一つである．画面上のメッセージ類の日本語表示は意味が明瞭で操作に迷うことは殆どない．これも日本国内での普及に貢献している．

2.3 広い応用範囲と取り組みやすさ

先に，Redmine はプロジェクト管理ソフトウェアに分類されることが多い，と記したが，Redmine には，Microsoft Project のようなプロジェクト管理専用のソフトウェアとは異なる特徴があり，Redmine の応用範囲の広さや取り組みやすさに繋がっている．ここでは，CODA での Redmine 活用に大きく寄与している特徴を三点取り上げる．

（1）汎用的なカードイメージのチケット構造
（2）チーム・共同作業向けの種々の機能
（3）Web による設定・定義と即時反映

2.3.1 汎用的なカードイメージのチケット構造

Redmine はさまざまな機能を提供するが，中核となっているのはチケット管理機能である．その本質的な特徴を一言で表現すると，「ステータス管理機能を備えた汎用的なカードの管理システム」である．チケットの構造を Figure 1 に示す．

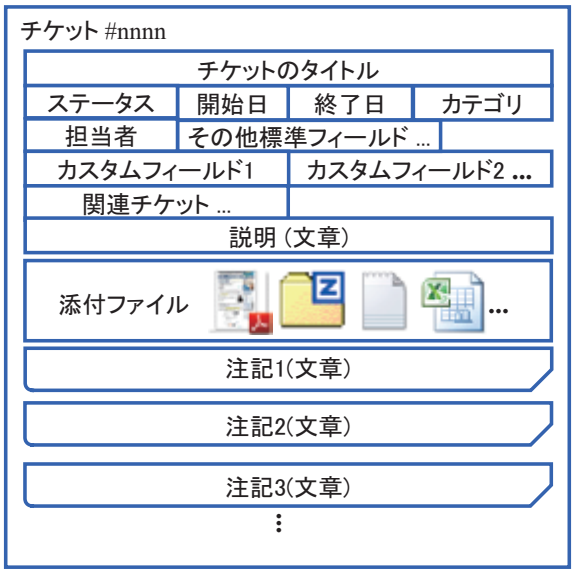

Figure 1　Redmine のチケットの構造

チケットには，チケット番号，タイトル，ステータス，担当者，開始日，終了日などの情報が付く．チケットで扱う内容を具体的に記すには，説明欄を利用する．これらは Redmine の標準フィールドで，整理カードの定型

項目に相当する．チケットで扱う項目に関する作業記録や調査結果などは注記に記す．注記は，カードに貼り付ける「付箋紙」のようなもので，時系列に記録される．チケットにはファイルを添付できる．これにより，作業の記録とその成果物や参照資料をまとめられる．更に，複数のチケットを相互に関連付けて参照することもできる．Redmine では，チケットの用途に合わせた独自のカスタムフィールドを複数定義し，標準フィールドと同じように扱える．

これらにより，1 枚のチケットは全体として以下を束ねた整理カードのように扱える．

（1）定型項目（標準フィールド,カスタムフィールド）
（2）概要記述（説明）
（3）作業記録などのメモ・付箋紙（注記）
（4）関連資料（添付ファイル）

作業記録や関連資料もまとめられることから考えると，カードよりも紙挟み（フォルダ）の喩えの方が適切かもしれない．

チケットの説明文や注記では，Wiki で用いられるマークアップ記法がサポートされており，箇条書きや太字などを容易に表現できる上，Redmine 内の他のチケットや Wiki，文書などの情報（2.3.2 で説明）へのリンクや外部の URL を簡単に記述できる．

チケットは，仕事の進め方を定義するトラッカー（4.1 で説明）毎に複数の種類を用意できる．

また，チケットはさまざまなフィールドを用いて条件を絞り込んだ一覧を取得でき，全文検索も可能である．一覧の CSV・PDF エクスポートやチケットの PDF エクスポートもサポートされている．

2.3.2 チーム・共同作業向けの種々の機能

Redmine はクライアント・サーバー型の Web アプリケーションであるが，単にブラウザで利用できるだけではなく，複数の利用者が共同して作業しやすいようさまざまな機能を提供している．複数利用者のチケットの注記追加やフィールドの更新の衝突に配慮がされているのは勿論，提供されている機能には以下のようなものがあり,扱う業務の性格などにより使用の有無を選択できる．

（1）Wiki
（2）フォーラム（掲示板機能）
（3）各種の更新のメールによる通知
（4）ニュース
（5）ガントチャートやカレンダー表示
（6）文書の保管・配布

前項と本項で述べた特徴から，Redmine は Microsoft Project のような各種の資源見積・実績管理を行う本格的なプロジェクト管理ソフトウェアよりは，むしろ，チケットを主体にしたチーム作業の支援ツールと捉えるのがより適切で取り組みやすく，さまざまな業務で広く利点を享受しやすいと考えられる．

その一方，Redmine はソフトウェア開発用に便利な以下のような機能も備えており，チームでの開発業務での利用でも有用である．

（1）バージョン管理システム（Subversion, Git 等）との連携
（2）ロードマップ（ターゲットバージョン管理）
（3）ファイル配布
（4）工数（作業時間）記録

2.3.3 Web による設定・定義と即時反映

Redmine の設定・定義はほとんど全てを Web ベースの画面で行うことができ，変更は再起動なしに即時に反映される．例えば，カスタムフィールドは形式を複数から選ぶことができるが，リストから選択肢を選ぶ形式の選択肢を追加したり選択肢の表示順を変更したりすることはよくある．Redmine ではこういった変更を僅か数分で行えるので，業務の変化への遅滞ない追従や迅速な改善が可能である．

3. スーパーコンピュータ活用課での利用状況

スーパーコンピュータ活用課では Redmine をベースとした CODA システムを JSS2 の運用や支援のための情報共有・進捗管理に使用している．ここでは，導入の経緯，利用状況，活用ケースを紹介し，更に，CODA の普及の要因について述べる．

3.1 チケット管理ステムの経験と課題及び CODA の導入経緯

現在のスーパーコンピュータ活用課に至る JAXA スパコンの運用部門では，10 年程前から NSIM（Numerical Simulator Incident Manager）という独自のインシデント管理システムを開発・運用していた[4]．NSIM は CODA と同様に Web ブラウザで利用する Web アプリケーションである．NSIM は長年利用されていたが，時間の経過に伴い，以下のような課題が顕在化した．

（1）ドキュメンテーションが不足しており，トラブル時の対応が困難である．
（2）システム種別などの選択肢がソースコード内に散在する，生成されるレポートの形式が固定である，など，新たなシステムの運用に合わせた改修やニーズの変化に対応しにくい．

(3) Web ブラウザ画面の作成に使用している XUL (XML User Interface Language) が特定の Web ブラウザを前提としている上，そのブラウザのバージョンアップに伴って次第に不具合が顕在化するようになった．

このため，NSIM に代わる新たなツールの導入を検討することとなった．検討に当たっては，チケット管理システム，インシデント管理システムと分類されるソフトウェアを複数検討した．検討に当たって重視した点は以下である．

(1) 特定の開発や運用の方法論やスタイルを前提としてそれに依存していない．
(2) チケットの種類や業務を複数設定でき，業務の内容や担当者による使い分けがしやすい．
(3) 導入・構築に当たって，新たに必要な技術の習得が少ない．
(4) JSS2 の導入開始までに試験的な運用を開始できるよう，短期間での構築が可能と見込まれる．
(5) 書籍やインターネットでの情報，特に日本語での情報を入手しやすい．
(6) バグ修正や機能の追加などの開発が活発で，議論がオープンである．

これに並行して NSIM を継承する独自開発ツールの構築も検討したが，Redmine を活用する方が独自開発よりも短期間に少ない工数で高品質のツールを実現でき，長期的に運用への適合の継続性も期待できるとの判断により，Redmine の採用を決定した．

また，以下も Redmine の採用に資するものであった．

(1) Redmine とその前提ソフトウェアを一括導入・設定できるパッケージ（bitnami（https://bitnami.com/stack/redmine）等）がインターネット上で配布されており，Windows ベースの PC で容易に機能確認ができたこと．CODA は Linux 上に構築されているが，導入決定前の機能確認を普段使用している PC で行えたことは検討を進める上で大きな利点であった．
(2) 機能確認の結果，設定の追加や変更が容易で，Redmine を使いながら，運用に合わせた変更を柔軟に行えると判断できたこと．

Redmine をベースとした CODA の本運用は，新たなスーパーコンピュータシステムである JSS2 の導入開始に合わせて開始した．本運用に先立つ 2-3 か月の間は，CODA 自身の設定や運用検討を中心とした作業を CODA で管理した．この間，Redmine の操作・機能にある程度習熟し，使いながら設定を見直すなどの作業を行えたことは，本運用の立ち上げに役立った．

従来の NSIM は，JSS2 に関わる案件には使用しないこととし，JSS2 の前身である JSS の運用停止に合わせて利用を停止した．

3.2 CODA の利用状況

CODA は，スーパーコンピュータ活用課の業務の多くで利用されている．ここでは以下のような側面から利用状況を概括する．

(1) 利用者数：執筆現在の CODA 登録利用者数は 46 名である．このうち，約 35 名が日々の業務で CODA を使用している．
(2) 利用者層：スーパーコンピュータ活用課の全員が CODA に登録されている．また，スーパーコンピュータ活用課以外に，JSS2 システムのベンダーの SE, CE も CODA を利用している．
(3) チケット数：2014 年 1 月の CODA の試験的な開始以降，現在までのチケット数は約 3100 である．2015 年 7 月までの月毎の発生数の累計と月間発生数の推移を Figure 2 に示す．2015 年 4 月の JSS2 第 2 期運用開始後は月々約 300 件程度のチケットが新規に起票されている．
(4) プロジェクト：スーパーコンピュータ活用課の大まかな業務区分としては，運用，可視化などの利用支援，ISO-9001 品質管理活動 [5] に加えて組織・業務のマネジメントがある．業務により主たる担当者が異なり，業務内容に即したチケットのフィールドを設けるようにしているので，CODA ではプロジェクトを使い分けている．現在は 7 個のプロジェクトがあり，利用者は業務上の必要に応じて 1 個または複数のプロジェクトに参加している．

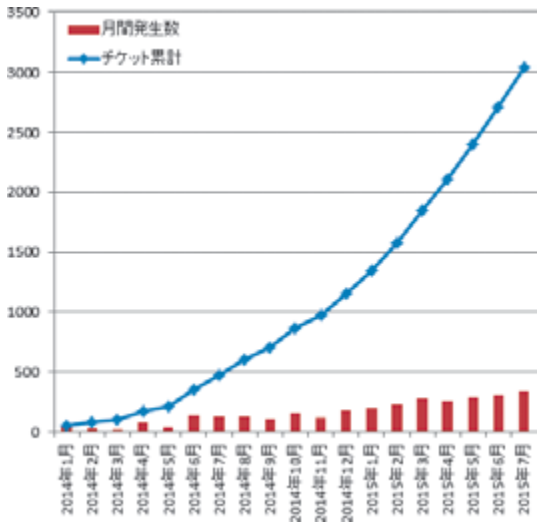

Figure 2 CODA のチケット発生数の推移

3.3 CODA の活用例

ここでは，Redmine の利用を検討する際の一助になるよう，CODA の具体的な活用例を幾つか紹介する．

3.3.1 日々のイベント・作業の記録・連絡

CODA が最も多く利用されているケースは，サービス要求やインシデント，障害などのイベント管理としての利用である．システムの利用に関するユーザからの照会，利用変更申請，障害などの案件毎にチケットが起票されている．また，運用に関わる作業の記録でも CODA が利用されている．

CODA での集中管理は，従来のメールでの連絡や Excel などのファイルでの管理に比べて，(1) 最新情報をリアルタイムに共有出来る，(2) メールやファイルのように情報が散逸しないので必要な情報を探しやすい．

また，ハードウェアやソフトウェア障害などは，例えば発生時期を用いて絞り込んでチケット一覧・件数を取得して PDF 化すれば，「先月の障害発生の機種別サマリー」といったレポートを容易に作成できる．このように，後から振り返って改善を図るための材料としても利用されている．また，ユーザ Q&A や申請の回答期間をチケットに記録するようにしており，月毎の発生件数や回答期間別の割合を分析する，といった作業にも CODA の記録を利用している．

3.3.2 管理帳票

CODA では，Redmine の柔軟性のあるチケットの使い方を管理帳票に利用している．以下に幾つか例を挙げる．

(1) 顧客所有物：顧客所有物は，スーパーコンピュータ活用課が実施・認証を受けている ISO-9001 に従い適切に管理しなくてはならない．具体的には，顧客所有物とはユーザから預かるソースコードなどである．顧客所有物は CODA でその保管場所，ソースなどの機械可読資料であれば複製の作成・抹消の状況などを記録する．

(2) 是正・予防処置：是正処置及び予防処置は ISO-9001 の「継続的改善」に規定のある項目で，記録やレビューが必要である．記録や作成した文書，レビュー・承認手続きは CODA で管理されており，進捗を一目で確認できる．

(3) ユーザニーズ：ユーザからのサービス改善・拡充に関する要望は定型化して管理されている．

(4) 優先実行（特別利用）：JSS2 システム資源を優先的に利用できるユーザ業務の管理を行う．審査の経過，開始・終了時期や優先使用できる資源量などの管理が必要である．ユーザとの連絡記録なども一括してチケットで管理している．

これらを CODA で管理するに当たっては記入項目・記入方法を標準化した．このため，検索では必要な内容を容易に一覧できる．

3.3.3 会議の準備，議事録，To Do 項目

CODA は会議の運営にもさまざまに利用されている．

会議議事録用のチケットには，会議の告知（日程・場所等）や議題に加えて，会議で使用する資料（ファイル）を添付する．会議終了後は議事録を説明に追記する．これにより，会議の準備状況を確認したり，過去の議事録を資料とともに振り返ることができる．

JSS2 導入のようなプロジェクトや運用関係のミーティングでは，To Do 項目の発生・進捗状況が大きな議題となるが，これらも CODA に記録されているので，プロジェクターで CODA の情報を見ながら会議を行うことができる．今日でも恐らく多く使用されているであろう Excel の表などでまとめる方法に比べ，まとめを作成する手間・時間が不要，最新情報を共有可能，版の枝分かれが起きない，など，の利点がある．

3.3.4 納品物本体 + 目録

JSS2 はその納品の際に仕様書・試験結果報告書などのさまざまな資料をベンダーから受領している．

こういった納品物はバインダーに綴じられた紙の書類の形態を取るのが一般的であるが，多くはオリジナルがワープロなどの電子的なファイルであり，オンラインで一覧・参照できる方が便利である．そこで，納品物ドキュメントを収める専用のプロジェクトを用意して，チケットに納品物の目録となる項目を記し，ファイル形式の納品物を添付するようにした．これは，いわば蔵書本体の閲覧が可能な「蔵書カード編集・検索システム」としての利用である．

3.4 CODA の普及要因

幸い，CODA は日々チケットが登録・更新されており，スーパーコンピュータ活用課の不可欠のツールとなった．しかし，Redmine のようなチケット管理システムを導入したもののあまり（もしくは全く）利用されない，というケースは少なくない．インターネット上のブログ等でもこのようなコメントは少なくないし，雑誌[6]や書籍[7]でもこうならないための対策が取り上げられている．

Redmine 及び他のチケット管理システムに取り組む際の参考になるよう CODA の普及要因を振り返ると，以下が挙げられよう．

(1) 情報蓄積を集中させる方針：JSS2 の導入作業の開始にタイミングを合わせて利用を本格化させたため，「新しいシステムに関する作業や資料は必ず CODA に登録」という方針を徹底しやすかった．また，会議の議事録や会議の To Do 項目，ハードウェア障害チケットなどの運用の標準化が CODA の利用を促す動機付けになった．

(2) マネジメントの支持：スーパーコンピュータ活用課では，マネジメントが CODA の利用を強力に推進している．将来，類似の障害や問い合わせがあったときに過去の記録が役立つと分かってはいても，チケットに日々の作業の記録を残すのは億劫なこともある．これに対して，マネジメントの積極的な CODA の活用方針や CODA チケットへのコメント，承認などの実践は，CODA 利用の定着の原動力の一つである．マネジメント側の CODA への主な期待は，作業効率化，それによるユーザサービスの品質向上・より創造的な業務のための時間創出，業務経験やスキルの共有・継承，ISO-9001 を支援するツールである．

(3) 前システムの利用経験:3.1 に記したとおり，スーパーコンピュータ活用課では約 10 年前から独自開発したインシデント管理システムを運用していた．チケットの起票・記録，Web アプリケーションによる即座の最新情報の共有の利点に関して既にある程度共通認識ができていたことは，CODA 運用開始時に大いに追い風となった．

(4) 利用者の取り込みに注力：CODA の運用当初は，まず利用者に「使って貰う」ことを奨励した．チケット作成の粒度や対応完了時のまとめ，作業やユーザとのメールの記録をどの程度行うかなどについてはあまり厳密ではなかったが，あまり細かなルールを徹底すると利用者が萎縮してしまい，利用されないことになりかねない．前項に述べたように前システムの経験者が多かったこともあり，幸いあまり大きな混乱はなかった．その後は利用に慣れるに従いベストプラクティス的な利用方法に収斂しつつあるので，現在はそれを下敷きに運用のルール化を図っている．

(5) 徐々に機能を拡充・変更：チケット管理システムは組織の活動に直結するので，その利用方法が作業効率に大きく影響する．しかし，最初からあるべき運用ルールや設計を的確に定めることは難しく，それにこだわっているとシステム稼働に時間を要してしまう．CODA の本運用開始時期は JSS2 システムの導入に合わせることとしたため，稼働後に使い方やシステム設定の変更をある程度容認することとした．Redmine ではフィールドの追加や変更が容易で，対象業務の追加・変更に合わせてプロジェクトやトラッカーを追加するのも簡単である．既存のチケットの別プロジェクトや別トラッカーへの移動も可能である．これらにより，Redmine の機能に習熟するにつれて気づいた運用や設計の改善を比較的円滑に実施できた．

(6) 使いやすいツールの選択：チケット管理システムのようなツールを日々利用していく上で大切なことは，利用者にとって使いやすく便利と実感して貰えることである．本稿でも既に幾つか紹介しているが，Redmine は，CODA 内の他のチケットや外部の URL の参照も容易に記述可能な Wiki 形式を利用した説明や注記，条件検索や全文検索，カスタマイズ可能な一覧表示などの便利な機能を備えている．ブラウザ上の UI では JQuery UI が使用されており，今日的な Web アプリケーションに相応しい UI を実現している．また，インターネットで公開されているプラグインを適宜導入することで，より使いやすいシステムに仕立てていくことができる．

4. Redmine の定義・設定のヒント

前章までで紹介したとおり，Redmine は定義の変更・反映が容易で，設定変更しやすいソフトウェアである．しかしながら，適切な定義をして運用していくには幾つか考慮点がある．本章では，CODA の構築・運用経験から見いだされた Redmine の導入・設定のヒントを紹介する．

まず，やや分かりにくい Redmine の定義の構造の明確化・整理を行う．次いで，Redmine の定義を効率的に行い, 利用者にも使いやすいシステムを作るための工夫，プロジェクト分割の目安を述べる．更に，プラグインの利用についても触れる．

4.1 Redmine の定義の構造の明確化・整理

Redmine は定義のほとんどをブラウザの管理画面から行うことができる．しかし，個別の画面のわかりやすさに反して，さまざまな定義がどう関係しているかが分かりにくく，設定の際に混乱しがちである．特に Redmine に馴染みがない場合は困惑することも少なくない．

管理画面のトップ画面を Figure 6 に示す．この画面では個別の設定項目がフラットに並んで表示されており，まず何を定義すべきでそれが他のどの項目にどのように

関連するかが分かりにくい.

個別の管理画面の例として，Figure 7 にカスタムフィールドの設定画面を示す. この画面の場合, 書式･名称･説明･選択肢などの他に, 右下に「トラッカー」や「プロジェクト」のチェックボックスがある．右下の「トラッカー」や「プロジェクト」という項目は管理画面のトップ画面の選択肢にも表示されているが，トップ画面の表示とカスタムフィールド画面の表示がどのように関係するのかは特に説明がないので，全体の見通しが良いとは言えない．（なお，カスタムフィールドの右下に「トラッカー」や「プロジェクト」が表示されている理由は後述する.）

市販の Redmine の解説書籍では，個別の設定の解説は見られるが，設定相互の関係を示しているものは残念ながら見当たらない．そこで，Redmine の主な定義の関係を示す全体構造を Figure 8 に示し，これを元に Redmine の構造を明確化・整理する.

まず，Redmine の定義は論理的な定義と物理的な実体の定義に大きく分かれている．図では，両者を左側の矢印で示した.

図内の「(1) ロール定義層」は利用者の役割を定義する部分である．役割毎に「ロール」を定義する．各々のロールでは Redmine で行える数十個の操作権限の可否をチェックボックスで設定する．ロールは，実際の利用者の権限をモデル化するものである．これに対して，実際の個々の利用者の定義は図の一番下の「(5) 利用者定義層」で行うが，詳細は後述する.

「(1) ロール定義層」の下には，「(2) チケット定義層」がある. ここでは, チケットの内容, ステータス遷移, ロール毎のチケット操作などに関わる定義を行う．この層内の左にある「ステータス」ではチケットに使用するステータス名とその属性を定義する．ここでやや分かりにくいのは，ステータスで定義するのはステータスの遷移ではないことである．ここで定義するのはワークフロー定義で材料として使用するステータス 1 個 1 個の定義である．ステータス遷移は後述の「ワークフロー」で定義する．次に，図の右端に示した「カスタムフィールド」と「標準フィールド」を紹介する．Redmine では，担当者や開始日・期日といった標準フィールドに加えて，独自にカスタムフィールドを定義できる．Figure 7 に示したのは「回答期間」というカスタムフィールドで，リストから 1 個選択する形式である．この画面では選択肢の設定や初期値，必須か否かなどを設定する．Figure 8 で「カスタムフィールド」と「標準フィールド」の左にあるのが，「トラッカー」の定義である.「トラッカー」の設定画面では標準フィールドや定義済のカスタムフィールドの一覧が示され，チェックボックスのオン・オフで使用するフィールドを指定する．ここまでで，チケットの動きを設定するために必要な項目が揃ったことになる.

Redmine のチケットの動きは，「(2) チケット定義層」の最後の項目である「ワークフロー」で決定する．ワークフローでは，これまでに定義したロールとステータスとトラッカーを組み合わせ，トラッカー毎に，どのロールであればどのステータスからどのステータスにチケットを遷移できるかを設定する．また，トラッカーに定義されている各々のフィールドについて，どのロールがどのステータスのチケットを操作するときにはフィールドが読み取り専用か，必須か，いずれでもないかといった設定を行う．例えば，「担当者」や「期日」は「新規」ステータスでは必須としないが，「新規」以外のステータスでは必須とするような指定ができる.

次に，「(3) プロジェクト定義層」以下の物理的な実体の定義について説明する．プロジェクトとは，チケットや Wiki，文書，フォーラムなどのデータが収容される実体で, 運用の都合などによって複数作成できる．プロジェクトには名前や概要説明を付けて識別する．各プロジェクトでは，そのプロジェクトで使用する Redmine の機能 (チケットトラッキング，Wiki，フォーラム，時間トラッキング等)，そのプロジェクトで使用するトラッカー，カスタムフィールドなどを定義する．図には示していないが，リポジトリ連携を行う場合はリポジトリの定義，時間トラッキングを行う場合は作業分類なども定義する.

図の一番下の「(5) 利用者定義層」では，Redmine の利用者をひとりずつ定義する．また，複数の利用者をグループにまとめて扱うこともできる. グループの用途は, プロジェクトへの利用者の追加をまとめて行ったり，チケットの担当者を利用者個人ではなくグループに割り当てたりすることである.

利用者をプロジェクトに参加させるには，「(4) ロール割り当て層」に示すように, 利用者（またはグループ）にロールを割り当てて，プロジェクトのメンバーに指定する.「(1) ロール定義層」で設定したロールの権限がそのプロジェクトでのその利用者またはグループの権限となり，ワークフローでそのロールに定義したステータス遷移やフィールドの設定が適用される.

なお，Redmine のプロジェクトでは「非メンバー (Redmine 利用者として登録されているがそのプロジェクトのメンバーにはなっていない)」や「匿名ユーザ (Redmine 利用者として登録されていない，ログインしていないユーザ)」というロールも定義されているが，これらは主としてインターネット上でのプロジェクト公開を想定したロール（例えば，http://www.redmine.org/ では，メンバー登録なしにチケットや文書を閲覧できる

が，これは匿名ユーザのロールで許可されているからである）なので，本稿では説明を省略し，利用者はプロジェクトにメンバーとして参加することを前提とする．

Redmine を導入すると，ロールやトラッカー，ワークフロー，プロジェクトなどの一連の定義のサンプルが自動的に作成される．カストマイズを行う際に，上記の説明と Figure 8 を参照しながら導入時の定義のサンプルを確認し，Redmine 全体の構造の理解に役立てて頂きたい．

最後に，カスタムフィールド設定画面の右下に「トラッカー」や「プロジェクト」が表示されている理由について補足する．これまでの説明によれば，まず，カスタムフィールドを定義し，次いでカスタムフィールドをトラッカーで使用するように指定し，更にプロジェクト定義でそのカスタムフィールドを使用するように指定する，というように順序立てて定義をしていくことになる．しかし，既存のトラッカー，既存のプロジェクトにカスタムフィールドを追加する場合には，カスタムフィールドを定義する画面で使用するトラッカーやプロジェクトを一緒に指定できると便利である．また，カスタムフィールドの定義を変更する際に，使用するトラッカーやプロジェクトの追加・削除を合わせて行えると便利である．この点について明確な記述のドキュメントはないが，カスタムフィールドの右下にある「トラッカー」や「プロジェクト」の設定はこのような定義の変更を迅速に行うためのショートカットとして表示されていると考えられる．

4.2 定義・設定の工夫・ヒント

ここでは，CODA を運用してきた経験から見いだされた，Redmine の定義を効率的に行い，利用者にも使いやすいシステムを作るための工夫について述べる．

4.2.1 ロール設定の OR ルール

4.1 に示したように，Redmine の定義は組織だった構造になっていて，相互に関係している．構造がしっかりしているのは大きな利点であるが，予め見通しを立てた上で定義を進めていかないと定義が非常に多くなって保守しにくい事態になる．特に，ロールとトラッカーの増加には注意が必要である．例えば，ロールとトラッカーを各々 4 種類定義すると，ロール毎に各トラッカーのワークフローを定義するので全部で $4 \times 4 = 16$ とおりのワークフローになる．もしロールを 1 つ増やし，トラッカーを 2 つ増やすと，ワークフローの定義は $(4+1) \times (4+2) = 5 \times 6 = 30$ とおりと，ほぼ倍加してしまう．多くのワークフローを全部変更しなくてはならないとすれば，保守の負担は大きくなり，誤りも発生しやすくなる．

これをいくらかでも単純化するには，利用者（またはグループ）をプロジェクトに割り当てるときに，

(1) 一人の利用者（または一つのグループ）に複数のロールを割り当てられる

(2) 利用者（またはグループ）が使用できる権限は，複数のロールの権限の OR となる

ことを利用するのが有用である．本稿ではこれを「ロール設定の OR ルール」と呼ぶ．

以下に，具体的な例を示す．

チケットのワークフローは単純な「新規」→「対応中」→「対応完了」→「承認済」の遷移と「対応完了」及び「承認済」から「対応中」に差し戻す遷移が定義されているとする．プロジェクトのメンバーになるのは，一般利用者の A さん，マネージャーの B さん，このプロジェクトの保守を担当する X さんである．通常のチケットの操作（新規チケット作成，「対応中」や「対応完了」などへのステータス遷移，注記の記入やフィールドの更新など）は全員が行えるようにする．ただし，チケットのステータスを「承認済」に変更できるのは B さんだけにしたい．また，保守系の操作（カテゴリの管理や文書の削除，ニュースの追加など）権限は X さんだけに付与するようにしたい．

Figure 3 は，あるプロジェクトへの利用者のメンバー登録を図示したものである．この図では，A さんはロール 1「一般」，B さんはロール 2「承認者」，X さんはロール 3「保守担当」でプロジェクトにメンバー登録されていて，上記のような権限管理ができていることになる．

ここで，例えば組織外の専門家に問い合わせている状態を示す「問合せ中」ステータスを追加するとする．このためには，ロール 1，ロール 2，ロール 3 のすべてのワークフローを変更しなくてはならない．また，文書の削除権限を全員に割り当てるような変更では，ロール 1 とロール 2 にその権限を付与しなくてはならない．

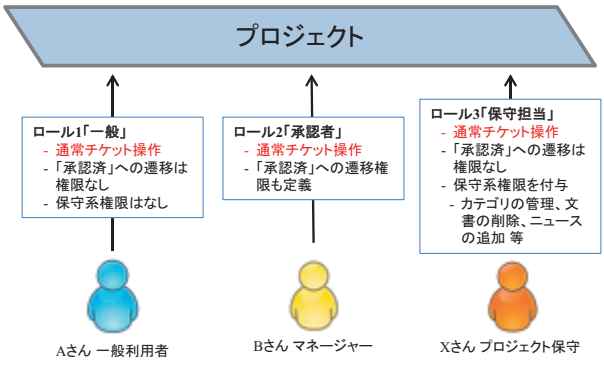

Figure 3 ロール設定とメンバー割り当て（1）

一方，Figure 4 は，ロール設定の OR ルールを利用した設定で，一人の利用者（または一つのグループ）に複数のロールを割り当ててメンバー登録することを示している．この場合，A さんはロール 1「一般」のみだが，B さんはロール 1「一般」とロール 2「承認者」の両方のロールで参加している．B さんは，ロール 1「一般」の権限に加えて，ロール 2「承認者」の権限によりチケットを「承認済」に変更できる．同様に，X さんはロール 1「一般」とロール 3「保守担当」の両方のロールで参加しているので，ロール 1「一般」の権限に加えてロール 3「保守担当」の権限でカテゴリの管理などを行うことができる．この場合は，「問合せ中」ステータスを追加するなどの通常のチケット操作に関わるワークフロー変更はロール 1「一般」にだけ行えばよい．

また，文書の削除権限を全員に付与するならば，ロール 1 に権限を付与すればよい．ロール 3 にある文書の削除権限は残しておいても実害はないが，ロール設定の OR ルールによってロール 1「一般」でカバーできるので削除するのが適切である．

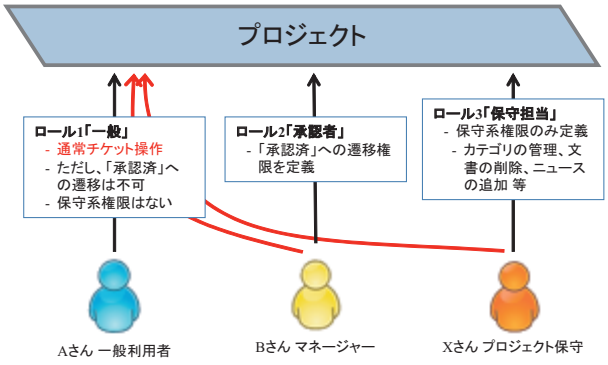

Figure 4　ロール設定とメンバー割り当て（2）

このように，ロール設定の OR ルールを利用することでロールと権限及びワークフローの保守負担を削減できる．CODA では，導入当初は Figure 3 のような定義をしていたが，システムの利用範囲が広がるにつれてロールやワークフローの保守が複雑になってきたため，運用開始から約 1 年経過したところで Figure 4 に示したロール設定の OR ルールを利用する方法に変更した．

ここで述べたような工夫を施しながら設定した CODA の現在のロールの一覧を Table 1 に示す．表内のロールの内，利用者がプロジェクトに参加する際は通常「レギュラー」ロールを適用する．これが最も基本的なロールで，チケットの作成・更新やチケットのステータス変更（一部を除く）などを行うことができる．Redmine の文書については閲覧のみ許可している．

「文書係」ロールは，Redmine の文書の保守権限（追加・編集・削除）のみを許可しており，他の権限やステータス遷移は許可がない．これは，一部のプロジェクトでは「文書を保守できるのは特定の担当者のみに制限したい」という要望から設けたものである．このプロジェクトでは，特定の担当者は「レギュラー」と「文書係」の二つのロールで参加しているので両者のロールが OR されて文書の閲覧と保守ができる．一方，他の利用者は「レギュラー」でのみ参加しているので，文書の閲覧はできるが保守はできない．これ以外のプロジェクトでは，「レギュラー」ロールの参加者は同時に「文書係」ロールでも参加するようにしているので，支障なく文書の操作を行うことができる．

「承認係」ロールは，マネージャーにのみセットするロールである．ステータス遷移の内「承認済」に遷移させる動きと「承認済」を「差戻」に変更する機能のみを許可している．一方，権限に関しては指定がない．これだけだとほとんど何も操作できないことになるが，マネージャーには「レギュラー」ロールも同時に割り当てるので，OR ルールによってすべてのステータス遷移が可能になる．一方，権限については「承認係」に指定がなくとも「レギュラー」ロールでの許可項目が OR ルールで適用される．

「管理係」ロールは，プロジェクトの保守作業を行うためのロールである．プロジェクトの概要やトラッカーの使用のオン・オフといったプロジェクトの設定変更や公開クエリ（予め設定しておき，メンバーがクリック一つで利用できる検索条件）の管理，ニュースの発行，Wiki ページの削除などのプロジェクトの保守に関する権限を許可している．これらの権限は他のロールには割り当てていない．「管理係」はこのような権限を許可する一方，ステータス遷移の定義は行っていない．「管理係」は，一部の保守担当者にのみ「レギュラー」ロールに追加して割り当てている．

上記の 4 つのロールはすべて CODA のデータや設定を変更するロールであるが，「オブザーバー」ロールはこれらと異なり，閲覧専用のロールである．一部のプロジェクトでは，利用者の一部に，チケットや Wiki，文書などを読み取り専用で閲覧のみ許可したいケースがあるが，この際には「オブザーバー」ロールを利用する．

なお，プロジェクトの追加，メンバーの変更，チケットの削除などの一部の権限は上記のロールのいずれにも割り当てていない．Redmine 管理者はロールの制限を受けることなく全操作が可能なので，これらの操作を行う場合は Redmine 管理者で実行する運用としている．こ

れらの権限をどの範囲で許可するのが適切かは組織によって異なる．Redmine 管理者のみ（集中管理型），プロジェクト毎の管理者に委譲(分散委譲型)，全員可能(フラット型）などが考えられるが，スーパーコンピュータ活用課では集中管理型で運用している．

4.2.2 グループ単位でのプロジェクトへの参加

Redmine では，Figure 8 に示したように，個々の利用者を直接メンバーとしてプロジェクトに割り当てる方法と，グループを作成してグループ単位でプロジェクトに割り当てる方法のいずれも可能である．

CODA では原則としてグループでの割り当てを採用している．人事異動などの利用者の入れ替えの際に，利用者個別の割り当てだと参加するプロジェクト個々の変更が必要となるが，グループ単位だと転出者をその利用者が所属していたグループから外し転入者と入れ替えるだけでよく，個々のプロジェクトへの参加の状態を変更する必要がない．特にプロジェクトやロールの数が多くなるとグループ単位での割当の方が保守の作業負担が少なく設定ミスが起きにくい．

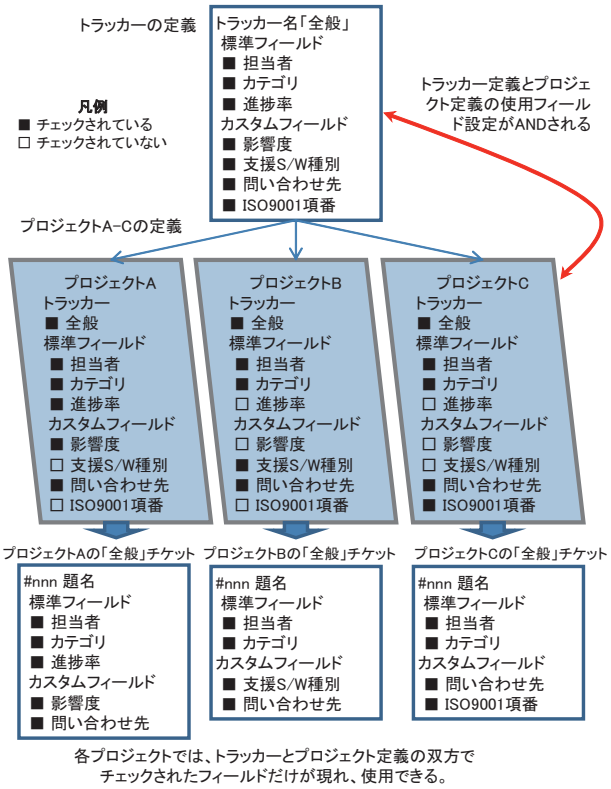

Figure 5 フィールド設定の AND ルール例

4.2.3 フィールド設定の AND ルール

トラッカーを作成する際の考慮点の一つに，そのトラッカーで利用するフィールドの設定がある．ステータスの遷移は業務の流れなので組織毎に概ね安定していることが多いと思われる（例えば 4.2.1 の例に示した，「新規」→「対応中」→「対応完了」→「承認済」）．一方，業務の内容によって使用するフィールドは異なるというケースは多い．

素直な実装方法は，それぞれに応じたカスタムフィールドを複数用意し，それを使い分けるトラッカーを個別に定義することである．この方法の欠点は，使用するフィールドが少しずつ異なるだけのトラッカーが複数できることである．トラッカーの数×ロールの数だけワークフローを定義しなくてはいけないので保守の負担が大きくなる．この点を考えるとトラッカーはできるだけ増やさずに，同じものを使い回すのが得策である．Redmine では，プロジェクトを分けることで，ここに示したフィールドの使い分けを実現出来る．本稿ではこれを「フィールド設定の AND ルール」と呼ぶ．具体的な例を Figure 5 に示す．

「全般」トラッカーは一般的なチケットを扱う汎用的なトラッカーで，組織の基本的な業務フローである，「新規」→「対応中」→「対応完了」→「承認済」（→「差戻」）がワークフローとして定義されている．一方，組織内の業務内容は，システム運用，利用支援，ISO-9001 に基づく品質管理活動に大きく分かれていて，主担当者や会議が異なる．しかし，業務フローは組織内では同一なので，どの業務にも同じ「全般」トラッカーを使用したい．このとき，以下のようにすると，「フィールド設定の AND ルール」を利用して業務に関係のあるフィールドだけを使用することができる．

まず，「全般」というトラッカーには，このトラッカーで使用する”可能性のある”標準フィールドとカスタムフィールドを定義する．実際にチケットを収容するプロジェクトは業務分野毎に A（システム運用），B（利用支援），C（品質管理活動）の 3 個を設ける．各プロジェクトでは，その業務で”実際に使用する”フィールドをチェックし，使用しないフィールドはチェックしない．Figure 5 のプロジェクト A を例に取ると，進捗率，影響度，問い合わせ先がチェックされているので，プロジェクト A の全般チケットにはこれらのフィールドが現れる．逆に，支援 S/W 種別や ISO9001 項番はチェックされていないので現れない．これはトラッカーのフィールド定義とプロジェクトのフィールド定義の両方でチェックされている（AND 条件を満たす）フィールドだけを実際に使用できるからである．同じ仕組みにより，プロジェクト C では問い合わせ先と ISO9001 項番は全般チケットに現れるが，進捗率，影響度，支援 S/W 種別は現れない．

この機能を利用すると，1個のトラッカーで，業務のフローは同じだが業務ごとに管理すべき内容（フィールド）が異なるケースに対応できる．これは，保守が容易になる以外に，利用者にとっても以下のような利点がある．

(1) 業務分野への関係があるフィールドのみが表示されるので，画面表示の煩わしさが少なく，業務に集中しやすい．

(2) 利用者が複数の業務分野を担当している場合，プロジェクトが異なっても同じトラッカー（例では「全般」）を利用できるので，トラッカーの選択肢が少なくなり，選択に迷うことが少なくなる．

4.3 プロジェクトの分割の目安

Redmineでは複数のプロジェクトを使用できるが，どのような場合にプロジェクトを分けるのが適切かに関しては既存の書籍やインターネット上にはあまり情報がないようである．しかし，CODAの構築・運用経験から考えるとプロジェクトの分割はRedmine活用の大きな要素である．プロジェクト分割の指針はRedmineを使用する組織の性格や考え方により異なると思われるが，業務範囲が広がったときにプロジェクトを分けるか否かは重要な検討課題になる．そこで，CODAの実装経験やこれまでの説明に基づいてプロジェクトの分割の目安を整理し，紹介する．

(1) セキュリティなどの理由により参加メンバーが異なるときは分割する．3.2で述べたように，CODAにはベンダーも参加している．ベンダーと共通に管理する案件はベンダーも参加するプロジェクトで管理し，それ以外の案件はベンダーが参加しないプロジェクトで管理している．Redmineにはプロジェクトを親子関係にする「サブプロジェクト」という機能があり，このようなケースでは有効である．ベンダーが参加するプロジェクトは，ベンダーが参加しないプロジェクトの「サブプロジェクト」としているので，スーパーコンピュータ活用課側の利用者は両方のチケットを一括して検索したり，文書を扱うことができる．一方，ベンダー側利用者はサブプロジェクトにしか参加していないので，親プロジェクトの内容を見ることはできない．

(2) 参加メンバーは同じでもロールを変更する必要があるときは分割する．CODAで使用している現在のロールはTable 1に記したとおりで，利用者には基本的にレギュラーというロールを割り当てる．しかし，納品物を格納するプロジェクトのみはオブザーバーというロールを基本にしている．納品物は読み取り専用とするのが安全なので，閲覧のみ可能なオブザーバーを割り当て，チケットや添付ファイルを保護している．納品物の新規追加や版の更新があるのは納品時期のみなので，この期間だけ特定の利用者にレギュラーのロールを追加して，納品物チケットの追加・変更を行う．なお，この際にチケットを誤って変更したり添付ファイルを削除してしまう可能性はあるが，Redmineではチケット変更の記録が残るので，追跡が可能である．

(3) 業務範囲により，フィールドの要・不要が異なったり，更新頻度が大きく異なるときは分割する．CODAはスーパーコンピュータ活用課のさまざまな業務の管理に使用されているが，チケットの多くはJSS2の運用に関わるものである．一方，組織内には可視化などの利用支援を主に扱うグループなどもある．このため，4.2.3で示した「フィールド設定のANDルール」を活用してプロジェクトを分けている．Redmineにはチケットの更新などをメンバーにメールで通知する機能がある．参加している全プロジェクトの全変更を通知，指定したプロジェクトのみ全変更を通知，などを選択できる．このような業務範囲による分割をすると，自分の主な業務範囲のプロジェクトだけは全通知を受け取り，それ以外のプロジェクトは自分が関係している変更のみ通知を受け取るといった設定ができるので，メール通知を適切に絞ることもできる．

(4) Redmineの保守関連作業のプロジェクトは分割する．これは前項の業務範囲とほぼ同じ要因だが，CODAでは，CODAの開発・保守用に専用のプロジェクトを設けてそれ以外の運用系のプロジェクトとは分けている．プロジェクトに固有の変更は業務に適用する前に予めこのプロジェクトで試験することもできて便利である．

(5)「カテゴリ」フィールドの選択肢を業務分野毎に分けて，選択・検索しやすくするときは分割する．これについては4.3.1で紹介する．

4.3.1「カテゴリ」標準フィールドの活用

プロジェクトの分割のメリットの一つに，Redmineが提供する「カテゴリ」標準フィールドの活用がある．「カテゴリ」は一般のフィールドとは働きがやや異なり，プロジェクトに大きく関係するのでここで概説する．

通常の標準フィールドは予め値の範囲や選択肢が決

まっている．また，カスタムフィールドは独自に設定できるフィールドだが，属性や選択肢はどのプロジェクトでも同じものが用いられる．これに対して，「カテゴリ」は文字列をリストから選択する形式だが，選択肢文字列はプロジェクト毎に独立している．このため，各プロジェクトで扱う業務内容に合わせた選択肢を用意し，業務を細かく分類したり，チケットに特定のキーワードを設定して検索での絞り込みに利用できる．4.2.3 で示した例に基づいて考えると，プロジェクト A（システム運用），B（利用支援），C（品質管理活動）でそれぞれの業務に即した選択肢を設けることができる．プロジェクトが 1 個だけの時はあまり大きな利点にはならないが，プロジェクトを分割した際は是非活用したい機能である．

CODA の構築の場合は，構築当初はプロジェクトが 1-2 個だったので，プロジェクト個別のカテゴリの選択肢は用意しなかった．利用が広がるにつれてプロジェクトが増加し，扱う業務も多彩になってきたので，現在はカテゴリの選択肢をプロジェクト毎に個別に設定するよう計画している．

カテゴリには，他のフィールドにはない特徴が更に二つある．一つは，選択肢に応じてチケットの担当者を自動割り当てする機能である．この機能は運用次第では非常に便利だが，現在のところ CODA で利用する計画はない．もう一つは，チケットの編集時に利用者が選択肢を直接追加できることである．これは便利な反面カテゴリの選択肢が無規則に多くなってしまう可能性もあるので，CODA では今のところロールと権限の設定で管理係にのみ許可している．こちらは一般の利用者に許可するか否かを検討しているところである．

4.4 プラグインの利用

インターネット上には Redmine の機能を拡張する各種のプラグインが多数公開されている．ほとんどは無償の OSS である．これらを活用することで Redmine をより快適に利用できたり，保守・管理が容易になる．CODA でも現時点で 6 個使用している．

プラグインの利用に当たっての基本的な注意は一般の OSS の利用と変わらないと思われるが，ここでは CODA でプラグインを導入する際に注意している点を簡単に紹介する．

(1) 情報や機能追加などの更新が定常的にされているかチェックする．本体のバージョンアップによりプラグインが動作しなくなる畏れもあるので，保守が継続されているかは重要である．
(2) 説明文やドキュメントが充実しているかチェックする．プラグインの機能にもよるが，ドキュメントが不十分な場合は無理に使用しない方が安全である．
(3) インターネットなどで利用例や評判をチェックする．過去の経緯などからも開発者の対応や保守の継続性を推測できる．
(4) できるだけシンプルなものを選ぶ．本体のバージョンアップなどで動かなくなる危険を考えると過度にプラグインに依存しない方が安全である．このため，高機能で複雑なプラグインを検討する際はシンプルなものよりも慎重に行う．

CODA の場合，実際にプラグインの導入を検討したが，最終的に利用しないという判断を過去 2 回ほど行った．そのうちの 1 個は，ドキュメンテーションが不足しており，過去の Q&A でも質問が放置されているケースが散見されたことが主な理由である．もう 1 個は，最近保守がされておらず Redmine 本体のバージョンアップに対応していないので試行錯誤して修正して使用している，という情報が複数あったためである．

5. 今後の展望

本稿の最後として，CODA 及び Redmine に関する今後の展望について触れる．

5.1 CODA の今後の拡充

CODA システムは本格稼働からほぼ 1 年以上が経過して概ね順調に利用を拡大し，スパコン活用課の活動に不可欠のツールになった．今後は以下のような点を中心にその活用を一層充実したいと考えている．

(1) 最新の Redmine 3.1 系列への更新：CODA は本格的な利用が始まった当時のバージョンである Redmine 2.5.x 系列を現在までそのまま使用している．その後 Redmine はバージョン 3 がリリースされた（本稿執筆時の最新は 3.1.0）．バージョン 3 では使い勝手を改善する機能が多数追加されたので，早期にバージョンアップを図り，作業の効率性向上に役立てたい．
(2) チケットの作成や終了などの標準化促進：稼働当初と比べて利用者が操作に慣れてきている一方，新規チケットの粒度，記述内容や対応終了の条件といったルールの標準化はまだ道半ばである．これは単にツールの使いこなしだけではなく，業務の標準化・見える化という組織の品質管理の側面もある．現在，ISO-9001 の視点から業務フローの標準化促進や見直しを図っているので，これと CODA の使い方を整合させて，より良く組織の

活動を支援するツールとして活用できるようにしたい.

(3) バージョン管理システムとの連携：Redmine は Git などのバージョン管理システムとの連携ができるが，CODA では試験的に Redmine のソースプログラムの一部修正やテーマ（CSS（Cascading Style Sheet）によるブラウザ画面の修正）の作成・保守でのみ使用している．JSS2 ではユーザ向けのコマンドや運用ユーティリティなどが開発・保守されている他，OS 等のシステムパラメータ・ファイルの保守なども行われているので，これらの保守内容と関連する事象・障害などの CODA チケットを連携できるようにバージョン管理システムとの連携を強化したい.

5.2 Redmine への期待

本稿で今まで述べたように，Redmine は高い汎用性を持つチケット管理システムである．試行錯誤により設定を見直して改善してきた面もあるが，CODA のような用途を考えると，全体としては非常に満足できるソフトウェアと判断している．ここでは，Redmine 本体及びエコシステムという観点から現状と今後の期待を述べる.

OSS の利用を考えるとき，開発が活発か，コミュニティが充実しているか，は非常に重要である．この点，Redmine は活発な開発が行われており，書籍の発行やコミュニティ活動も多い．今後も機能追加や使いこなしの情報が順調に増えると期待できる.

開発案件などのプロジェクト管理に本格的に活用する場合には Microsoft Project などに比べて機能面や操作性の面で劣っている部分も少なくないが，これらを補う有償のプラグインが提供され始めている．また，Redmine を ASP（SaaS）として提供したり，設計・導入・保守を行う会社も複数存在する．OSS の普及・定着にはこのようなエコシステムの充実は重要であり，非常に心強い存在である.

6. おわりに

本稿では，Redmine の特徴，スーパーコンピュータ活用課でのチケット管理ステムの経験と課題及び CODA の導入経緯，利用状況，CODA の構築・運用経験から見いだされた Redmine の導入・設定での独自の工夫やヒント，今後の CODA と Redmine の展望について述べた.

Redmine 本体やプラグインの開発・保守に関わっている方々や，インターネットでの情報発信・書籍や雑誌での紹介・オフラインミーティングの開催などの活動を行っているコミュニティの方々，SaaS や導入・保守サービスなどを提供されている企業・団体など，Redmine のコミュニティの皆様に感謝の意を表します.

また，CODA の企画や利用を通じて多くのご意見・ご支援を頂いた利用者各位に感謝の意を表します.

参考文献

1) 藤田直行，JAXA 新スパコン JSS2 の導入目的と構成概要，第 46 回流体力学講演会 / 第 32 回航空宇宙数値シミュレーション技術シンポジウム，2014，https://repository.exst.jaxa.jp/dspace/handle/a-is/459711（閲覧日：2015/08/21）

2) “Who uses Redmine?”，http://www.redmine.org/projects/redmine/wiki/WeAreUsingRedmine，（閲覧日：2015/07/26）

3) 加藤慶信，開発支援ツール徹底調査 2013，日経 SYSTEMS，2013 年 6 月号，pp.49-51

4) 松尾裕一，土屋雅子，JAXA 大規模 SMP クラスタにおける運用の課題と工夫，大規模 SMP 運用 WG 成果報告書：添付資料 39-1，サイエンティフィック・システム研究会，2006，http://www.ssken.gr.jp/MAINSITE/download/wg_report/smpo/t39-1.pdf（閲覧日：2015/07/26）

5) “ISO 9001:2008 Quality management systems – Requirements”，ISO（International Organization for Standardization），2008，http://www.iso.org/iso/catalogue_detail?csnumber=46486（閲覧日：2015/08/24）．なお，ISO-9001:2008 と同等の JIS 規格は，JIS Q 9001:2008「品質マネジメントシステム－要求事項」である．JIS Q 9001:2008 は，日本工業標準調査会ウェブサイト（http://www.jisc.go.jp/app/JPS/JPSO0020.html）で“Q9001”を指定して検索すれば閲覧出来る.

6) 矢口竜太郎，こうすれば必ず成功！Redmine 導入の勘所，2013，日経 SYSTEMS，2013 年 9 月号，pp.50-55

7) 倉貫義人 et. al.，Redmine - もっと手軽にプロジェクト管理！，2009，インプレスジャパン

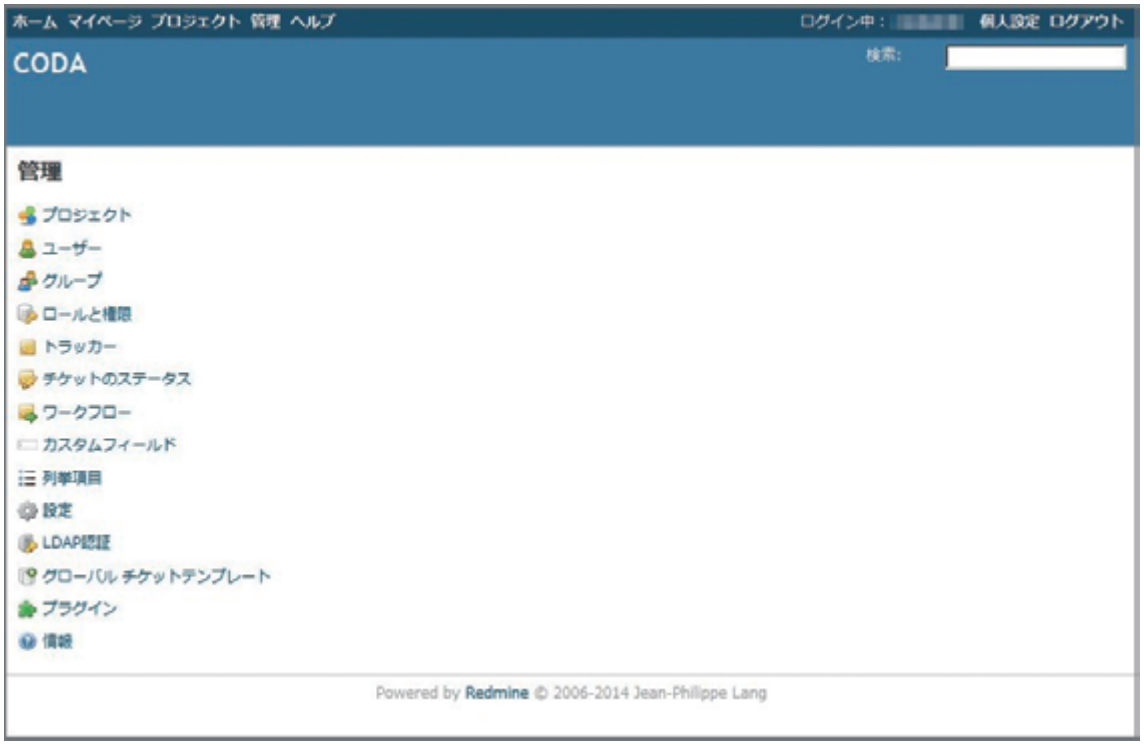

Figure 6 Redmine の管理のトップ画面

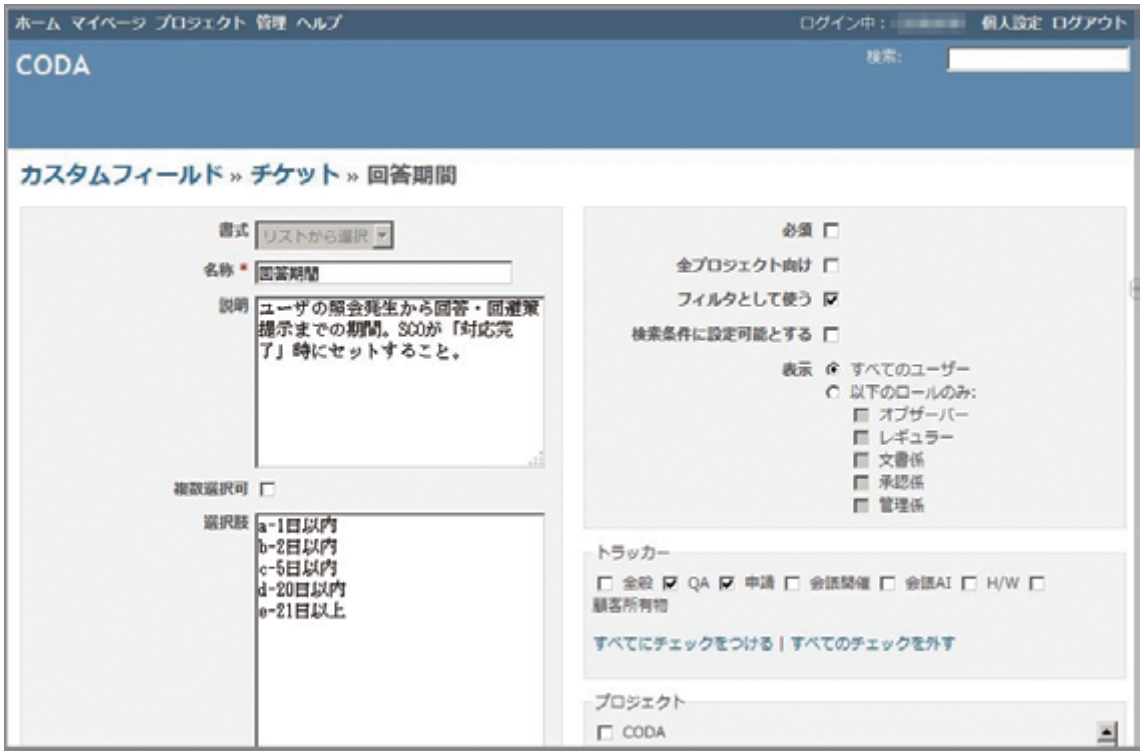

Figure 7 管理画面の例 カスタムフィールド

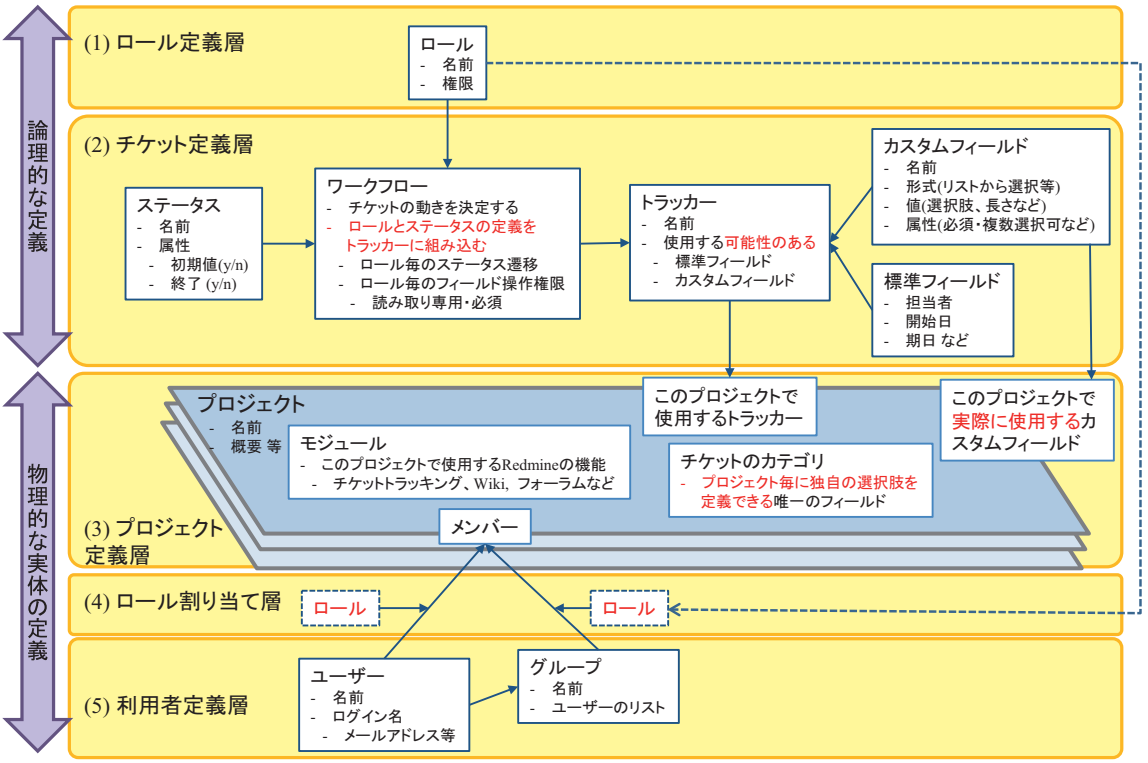

Figure 8　Redmine の主な定義の構造

Table 1　CODA で設定しているロール

ロール名	説明	ステータス遷移の設定	ロールと権限の設定
レギュラー	通常（チケット操作，Wiki 編集等）の役割．一般の利用者は普段この役割で活動する．	承認係のみが可能な遷移を除くステータス遷移が可能．	文書係と管理係にのみ割り当てている権限以外の通常の操作は許可．
文書係	文書の管理を行う．	なし	文書の追加・編集・削除のみ許可．
承認係	マネージャーとしての承認を行う．	「承認済」への変更及び「承認済」を「差戻」に変更することのみ可能．	なし
管理係	プロジェクトの保守作業を行う．	なし	プロジェクトの設定変更やチケット・Wiki ページ等の削除を行う権限のみ許可．
オブザーバー	チケット等の閲覧のみできる．	なし	チケットや Wiki ページ等の閲覧権限のみ許可．

2015/08 現在

注：ロールと権限では，プロジェクトの追加，メンバーの管理，チケットの削除などの権限は上のロールのいずれにも割り当てていない．これらの操作が必要な際は Redmine 管理者で行うこととしている．

「CODA: JSS2 の運用・ユーザ支援を支えるチケット管理システム」への正誤表

Corrigendum to “CODA: Ticket Management System to Support JSS2 Operation and Assistance to Users”

以下は、JAXA-RR-15-003「CODA: JSS2 の運用・ユーザ支援を支えるチケット管理システム」の正誤表である。

pp.10 Figure 5

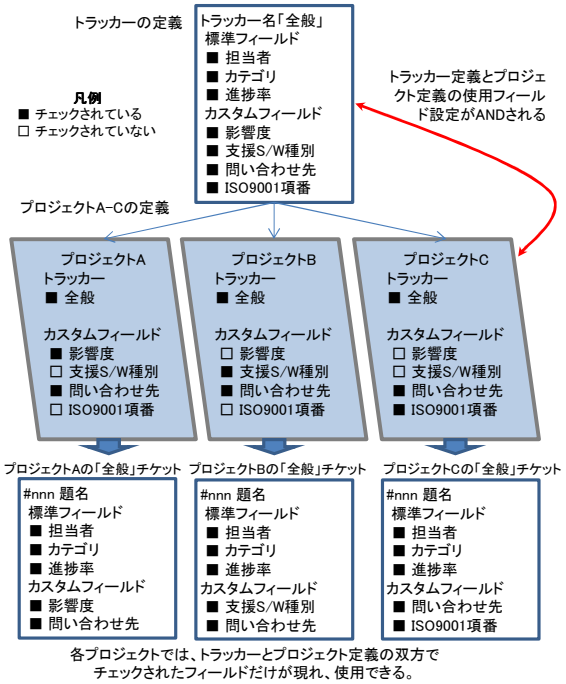

Figure 1 フィールド設定の AND ルール例

pp.10　51 行目

誤) まず,「全般」というトラッカーには, このトラッカーで使用する”可能性のある”標準フィールドとカスタムフィールドを定義する.

正) まず,「全般」というトラッカーには, このトラッカーで使用する”可能性のある”カスタムフィールドを定義する.

pp.10　58 行目

誤) Figure 1 のプロジェクト A を例に取ると, 進捗率, 影響度, 問い合わせ先がチェックされている

正) Figure 1 のプロジェクト A を例に取ると,影響度, 問い合わせ先がチェックされている

pp.10　65 行目

誤) プロジェクト C では問い合わせ先と ISO9001 項番は全般チケットに現れるが, 進捗率, 影響度, 支援 S/W 種別は現れない.

正) プロジェクト C では問い合わせ先と ISO9001 項番は全般チケットに現れるが,影響度, 支援 S/W 種別は現れない.

2016/05/31