ISSN 1349-1113
JAXA-RR-16-002E

JAXA Research and Development Report

CODA: Ticket Management System to Support JSS2 Operation and Assistance to Users
- Redmine Implementation and Hints of Its Usage -

Kazuhiro KIMOTO

September 2016

Japan Aerospace Exploration Agency

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 issue-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.

Note to English translation of the paper:

Redmine uses term of "issue" for items (tasks, incidents etc.) to be managed.

In Japanese translation of Redmine software localization, "issue" is translated into "チケット (ticket)". Original Japanese edition of this paper uses "ticket" in accordance with Redmine software Japanese localization.

In English translation of this paper, the term "issue" is used instead of "ticket", so that people around the world could understand what "issue" is, without interpreting "ticket" to "issue".

Exception of using "ticket" is only in the title of this paper. The title remains the same as Japanese original version so that it is identified that the papers of both Japanese and English translation have the same content.

There is a corrigendum which was published on May 31, 2016 to the original Japanese edition. The corrigendum is applied in this English edition.

Keywords: Redmine, JSS2, CODA, ticket-management-system, project management software, issue management system

1. Introduction

The Japan Aerospace Exploration Agency (hereafter, JAXA) began operating the JSS2 (JAXA Supercomputer System Generation 2) – SORA (Supercomputer for Earth Observation, Rockets, and Aeronautics)[1)], JAXA's second-generation supercomputer system, in October 2014. With the addition of the primary computational resource SORA-MA (Main System) in April 2015 for stage-two operations, full-scale operation is underway.

JAXA's Supercomputer Division is working on JSS2 operations and various activities related to user assistance. At the same time as the installation of JSS2, the Supercomputer Division implemented a issue management system called CODA (acronym for "CODA is the Operation and Development Assistant"). CODA is now an integral part of the Supercomputer Division's activities, and is used for information sharing and progress management in operations and support. CODA is a business management application based on Redmine, an open-source software program for issue and project management.

The objective of this paper is to provide useful information to people considering introducing, or implementing, an issue management system such as Redmine in the future. The paper first overviews the characteristics of Redmine, then discusses the experiences and issues the Supercomputer Division had with the issue management system, the introduction and usage status of CODA. It also discusses the benefits of the system and its utilization.

Best-practice hints and tips for introducing and setting up Redmine are also presented. These were found out while building and operating CODA, and are a practical resource for those are already using Redmine as well. Finally, the future outlook for CODA and Redmine is discussed.

doi: 10.20637/JAXA-RR-16-002E/0001

[*] Original Japanese Edition: Received on October 6th, 2015
English Translation: Received on August 19th, 2016

[*1] Supercomputer Division, Security and Information Systems Department

2. Overview of Redmine

Redmine is an open-source software (OSS) application developed and published at http://www.redmine.org/. In general, it is classified as project management software, but is sometimes also classified as issue management software.

2.1 Environment of Redmine

Redmine is a web-based client server application developed with Ruby on Rails. It can be installed on a server running Unix variants, MS Windows or Mac OS X. Major prerequisite software on the server end are an RDB (MySQL, etc.), an HTTP server (Apache, etc.), Ruby, and Ruby on Rails. The client end uses a web browser to access the system. Web browsers that support JavaScript, such as Firefox, Chrome, Safari, and Internet Explorer, can be used.

2.2 Development and Usage of Redmine

Redmine is being actively developed. In general, version upgrades are provided every four to five month. New versions actively incorporate issue reports and requests for additional features from users all over the world. Version 3.0, with many enhancements, was released in February 2015 (version 3.1.0 is the latest release at the time of writing).

One well-known example of Redmine being used for development and bug tracking is Ruby's development management system (Ruby Issue Tracking system, https://bugs.ruby-lang.org/)[2]. Moreover, Redmine development, release of new versions, and related discussions are also managed with Redmine at http://www.redmine.org.

Redmine have been coming to the forefront in Japan primarily in the IT development and management areas. Although accurate number of users and market share information are not available because Redmine is an OSS, according to a survey of development support tools in the June 2013 issue of Nikkei Systems[3], it is second in market share for project management tool installations at 22.3%, following the first-place Microsoft Project (26.9%). When only the two-year period closest to the time the survey was conducted is considered, it is in first place at 15.3%, more than double the share of Microsoft Project (6.2%). In addition, many Japanese-language books and book-style magazines specialized for Redmine are being published.

Rich and accurate Japanese localization is another characteristic of Redmine. The meaning of texts displayed in Japanese, such as in on-screen messages, is clear, and there is rarely confusion over how to use the software. This has also contributed to Redmine's popularity in Japan.

2.3 Wide Range of Applications and High Adaptability

It is stated above that Redmine is usually classified as a project management software program. However, Redmine has characteristics that are different from conventional project management-dedicated software such as Microsoft Project. These characteristics relate to Redmine's wide range of applications and its high level of adaptability. The following three characteristics contribute significantly to the active use of Redmine with CODA:

(1) Card Image of multipurpose issue structure.
(2) Various features geared toward team and collaborative work.
(3) Web-based settings and definitions that take effect immediately.

2.3.1 Card Image of Multipurpose Issue Structure

Redmine offers a variety of features; however, the issue management feature is the centerpiece of the program. Its essential characteristics can be summed up as follows: a "management system for multipurpose cards that offers status management features." Figure 1 shows the structure of a Redmine issue.

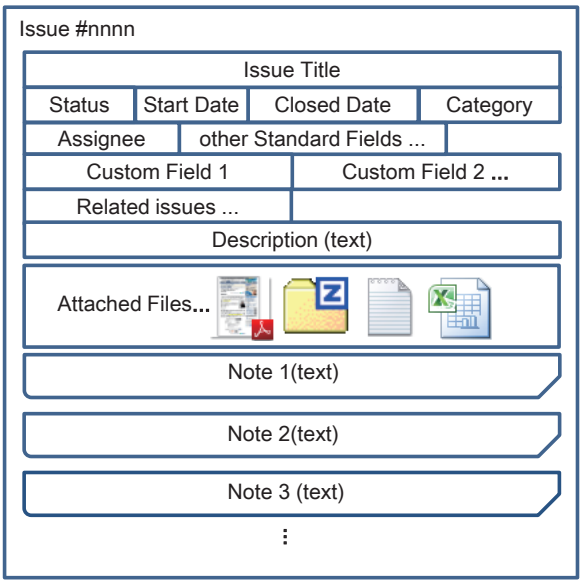

Figure 1 Structure of a Redmine Issue

A Redmine issue contains information such as issue number, issue title, status, assignee, start date, and closed date. Description column is for adding detailed information about what the issue deals with. These standard fields of Redmine are equivalent to preprinted fields on a work slip (card). Notes sections are used to add work records and investigation findings related to the items dealt with by the issue. Notes are like sticky notes attached to the card and are recorded chronologically. Files can also be attached to the issue. This makes it possible to bundle work records together with their outcomes and reference materials. Furthermore, it is also possible to associate multiple related issues together for reference. In Redmine, it is possible to define multiple unique "custom fields" to suit the purpose of the issue, and these fields can be handled in the same way as standard fields.

Thus, the issues can be handled like work slips (cards) that bundle the following items as a whole:

(1) Preprinted fields (standard fields, custom fields).
(2) Summary descriptions (description).
(3) Memos and sticky notes such as work records (notes).
(4) Related materials (attached files).

If we consider work records and related materials being grouped together, a "folder" may be a more appropriate metaphor than a work slip.

Description and notes sections of an issue support the markup notation used on Wikis. In addition to easy formatting such as itemization or bold text, it is also simple to put in external URLs and links to information within Redmine such as other issues, Wikis, or Documents (Section 2.3.2).

Multiple combinations of fields can be defined to the trackers (one combination for one tracker) and combined with the workflows (Section 4.1).

Moreover, generation of the issue lists are supported, using field values as query criteria. Full text search is available. CSV and PDF export of generated lists and PDF export of issues are also supported.

2.3.2 Various Features Geared Toward Team and Collaborative Work

Redmine is a client-server web application and it means more than just "using web browsers". It also offers a variety of features that make it easy to work collaboratively with multiple users. Resolution of conflict of adding notes and field updates on issues by multiple users are taken into account. Moreover, Redmine offers features such as those below, which can be selected to be used or not, depending on the nature of the business which Redmine is applied to:

(1) Wikis.
(2) Forums (bulletin board feature).
(3) Notification of updates by email.
(4) News.
(5) Gantt and Calendar.
(6) Repository of the documents.

Based on the characteristics outlined in this and the previous section, Redmine can be considered as an assistance tool that supports team operations centered on issues, rather than a standard project management software such as Microsoft Project, which is aimed to resource and value management. Such a view is appropriate for Redmine. It makes it easier to adapt Redmine in various businesses and enjoy its benefits.

Redmine also offers features such as the ones below that are useful and valuable when used for team development work as well.

(1) Cooperation with version control system software (Subversion, Git, etc.).
(2) Roadmap (target version management).
(3) Distribution of files.
(4) Recording of labor hours (time spent on work).

2.3.3 Web-based Settings and Definitions that Take Effect Immediately

Virtually all Redmine settings and definitions can be made via a web-based console, and changes take effect immediately without the need for a restart. For instance, the format of a custom field can be defined from multiple format types. In the “List” format type, it is common to add possible values or change the order of the possible values. Such changes take only a few minutes in Redmine. This makes it possible to reflect the business changes to Redmine and to improve it quickly, without delay.

3. Usage in the Supercomputer Division

The Supercomputer Division uses the Redmine-based CODA system for information sharing and process management in operations of JSS2 and user support activities. This section describes the introduction of Redmine to the division, usage statistics, and examples of utilization for CODA. Also, the factors which make proactive use of CODA is discussed.

3.1 Experiences and Issues of the former Issue Management System, and Introduction of CODA

The JAXA Supercomputer organization, which is now the Supercomputer Division, had been developing and using a custom-made incident management system called NSIM (Numerical Simulator Incident Manager) approximately for 10 years[4]. Like CODA, NISM was a web application that uses web browsers. As time went by, issues such as those described below began to surface over time:

(1) Insufficient document and troubleshooting difficulties.
(2) Difficulty to adapt to changing needs and improvements for newly introduced supercomputer systems to be used. For example, option values for system classification were scattered within the source codes, and the format of reports to be generated was fixed.
(3) XUL (XML User Interface Language), which has been used for web browser screen control, assumes the use of a specific web browser, and incompatibilities has been getting apparent as many release-ups of the browser software.

Because of these issues, the Supercomputer Division began to study for alternative software to replace NSIM. In the study, multiple candidates as issue management systems or incident management systems were considered. During the process, the following viewpoints were focused.

(1) The software has no assumption or reliance on a specific methodology or development style.
(2) Multiple definitions of issue type and work are allowed, to make it easy to use the software for different ways according to the details of the business or the person in charge.
(3) The software does not require users to learn new technologies very much when installing and setting-up.
(4) New system can likely be built in a short period of time so that the preliminary system can be started before the introduction of JSS2 begins.
(5) Rich information about the software in books or on the internet is available, particularly in Japanese.
(6) Development activities are active for such as bug fixes and additional features, and the discussion is open to public.

In parallel with searching for the software, the Supercomputer Division also considered to develop an in-house tool to succeed NSIM, Redmine was finally adopted. The reason is that it would provide the organization with a high-quality tool in a shorter period of time and with fewer effort than developing an in-house tool. Another driver was that it was expected to keep sustainability to be compatible with the business over the long term.

The following points also contributed to the decision to adopt Redmine.

(1) Packages are available on the net that allow users to install and set up Redmine and its required software as a bundle (Bitnami [https://bitnami.com/stack/redmine], etc.). By using these packages, functions could be evaluated on the PCs running Windows without difficulty. Although CODA is built on Linux, it was a major advantage that evaluation could be performed on the commodity PCs during the consideration process.
(2) As a result of the evaluation, we got certain that adding or changing settings of Redmine was

simple, and that we would be able to make changes in order to suit Redmine to our operations in a moderate and flexible manner.

Redmine based CODA system started production run, synchronized with the start of the installation of new JSS2 supercomputer system. Prior to the production run, works focusing on investigating the settings and use of CODA itself was managed with CODA as a "test drive". Over this period, we became relatively proficient in Redmine's operation and functions, and improved the settings while using it. It was helpful in launching production run.

We decided not to use the existing old NSIM system for matters related to JSS2. NSIM was stopped when JSS, the predecessor of JSS2, was retired.

3.2 CODA Usage

CODA is used for most of the Supercomputer Division's operations. This section summarizes CODA usage from the perspectives below.

(1) Number of users: CODA has 46 registered users, with approximately 35 of them using CODA for daily operations at the time of writing.

(2) Organization of Users: All members of the Supercomputer Division are registered in CODA. Moreover, vendor members of JSS2 system (i.e. System Engineers and Customer Engineers) are also registered.

(3) Number of issues: From the start of CODA's test drive in January 2014 to the time of writing, there have been approximately 3,100 issues. Figure 2 shows the accumulated number of issues registered and the trend per month up to July 2015. Following the start of JSS2 stage-two operations in April 2015, approximately 300 new issues have been registered every month.

(4) Projects: Major business scope of the Supercomputer Division is the following: operations and management of Supercomputers and IT facilities, ISO-9001 based quality management [5], and user support including visualization of computational outputs. In addition, there is also organizational and business management. CODA has several different projects since the primary persons in charge differ depending on the business, and fields of the issues corresponding to the nature of the business are different. There are currently seven projects. Users participate in single or multiple projects according to their business responsibilities.

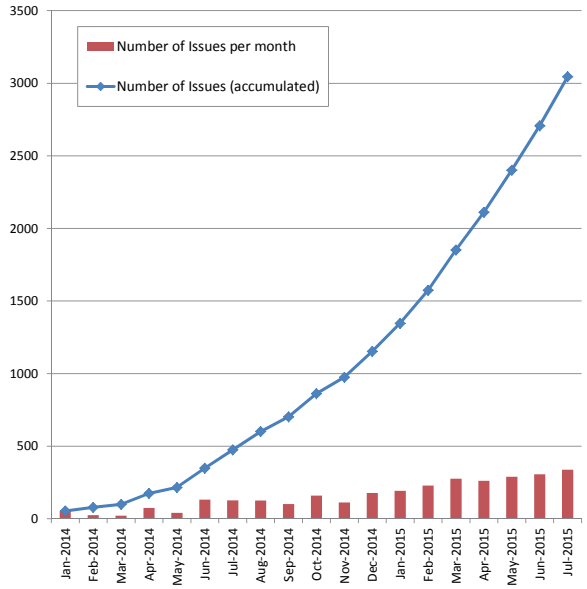

Figure 2 Number of Issues Registered in CODA

3.3 CODA Application Examples

This section presents several specific examples of CODA applications that can serve as a reference for those thinking of using Redmine.

3.3.1 Daily Events, Work Records, Communications

CODA is most frequently used for service requests, incidents, and event management of service failure and so on. An issue is filled out for each matter, including inquiries from users about using the system, change requests, and malfunctions. In addition, CODA is used to record jobs which are related to operations.

It is easier to search for necessary information in centralized management system such as CODA, compared with conventional email communication or file-based management systems such as Excel because (1) the most recent information can be shared in real time and (2) information is not scattered around as it is in emails and files.

Creating statistical reports on events such as hardware and software malfunctions is also simple. For example, a user can obtain a list of issues and number of occurrences by narrowing down query results based on the period when the malfunction

occurred, and exporting the list as a PDF which is to be a report of "Summary of Malfunctions in the Previous Month, organized by Manufacturers and Machine types." In this way, CODA can also be used as a source of data when looking back on past events to identify what to improve. Another example is that turnaround time of user Q&As and service requests are recorded in issues so that CODA issues are used to analyze the number and distributions of turnaround time.

3.3.2 Management Records

In CODA system, Redmine's flexible method of dealing with issues are used for management records. The following are some examples of management records.

(1) Customer Property: In accordance with the ISO-9001 certification that the Supercomputer Division has implemented and received, Customer Property must be managed appropriately. More specifically, Customer Property includes items such as source codes with which are entrusted by users. Statuses of Customer Property is recorded in CODA, for example, where it is kept, where its copies are made to for evaluation by whom, number of copies made, when the copy is deleted by whom.

(2) Corrective and Preventive Action: Corrective and Preventive Action is defined in "continual improvement" in ISO-9001 clause. It must be recorded and reviewed. Records, related documents made, and record of review and approval are managed in CODA, making it possible to examine its progress with just one click.

(3) User needs: Requests from users regarding service improvement and expansion are managed in CODA in a standardized form.

(4) Priority Usage (special usage): Priority usage is the program of JSS2 usage, which allows the user's jobs to be scheduled with high priority. The users of the program could use JSS2 system resources more than normal users could. It is necessary to record the approval process of the nominees, period of priority usage start and end, and the amount of resources that can be assigned to the priority usage. Related information such as communication with users is also bundled and managed within an issue.

How they are recorded in CODA is standardized, for example, what to input and how the detail information is described so that it is easily retrieved with queries.

3.3.3 Preparation and Minutes of meetings, To-do List Items

CODA is also used in a variety of ways to manage meetings.

In addition to meeting announcements (date, location, etc.) description, the materials used in meetings (files) can be attached to issues for meeting minutes. After a meeting ends, the minutes are added to the description. This makes it possible to, among other things, view the status of meeting preparations or look over the minutes of past meetings along with the materials used.

At meetings which are related to operations, or projects such as the introduction of JSS2, the creation and progress of to-do list items are major topics on the agenda. These ones are recorded in CODA, meetings can be conducted while showing the CODA information on a projector. This has many advantages compared to summarizing using tables in Excel, which is likely still widely used: no time nor effort needs to be spent for creating summaries, the latest information can be shared, and there are no discrepancies between versions.

3.3.4 Deliverables plus Their Catalog

At the time when JSS2 is delivered, there are a variety of deliverables such as specifications or test result reports from vendors along with computer systems.

Such deliverables generally arrive in the form of physical documents organized in a binder, but the original documents are usually digital files of word processors and the like. It means that it is more convenient if they are viewed and referred online. As such, we prepared a dedicated project to collect deliverable documents, added categories to issues to catalog deliverables, and began attaching deliverables in file form to issues. In this way, CODA is used as a "library card compilation and search system" that allows users to access the library documents.

3.4 Factors behind the wide use of CODA

Luckily, issues are registered and updated in CODA every day, and CODA became an indispensable tool in the Supercomputer Division. However, there are a fair number of examples of issue management systems like Redmine being introduced but barely (or never) used. Such comments often appear in online blogs and the like, and magazines[6] and books[7] have also presented strategies for avoiding such situations.

As a reference for those dealing with Redmine and other issue management systems, the following points can be cited as factors behind the wide use of CODA.

(1) A policy of centralizing accumulated information: We synchronized the timing of the full-scale production of CODA with the beginning of JSS2 introduction, it was easy to consistently follow the policy to "always register work and materials related to the new system into CODA." In addition, a source of motivation that encouraged us to use CODA was the standardization of operations, including issues of meeting minutes, meeting to-do list items, and hardware malfunctions.

(2) Management's support: In the Supercomputer Division, management strongly encourages the use of CODA. Even with the knowledge that past records are helpful when a similar malfunction or question arises in the future, recording daily work in issues can sometimes be a hassle. For addressing this, proactive CODA usage policy, comments on CODA issues, and approval of issues from management are one of driving forces behind CODA taking root. Management's major expectations regarding CODA are as follows: to increase the efficiency of work, thus improving the quality of user services and creating more time for creative work; to share and inherit work experiences and skills; and to support ISO-9001.

(3) Experiences of the previous system: As described in Section 3.1, the Supercomputer Division had been using an in-house incident management system for approximately 10 years. When use of CODA began, it received a major boost from the fact that, to a certain extent, there was already a shared awareness of the benefits of ticket issuing, recording and immediate sharing of the most up-to-date information using a web application.

(4) Focusing on bringing in users: When CODA was initially introduced, users were first encouraged to use it by being asked to do so. There was not very strict rules about granularity of issues, how summaries are to be made when the issues are completed, or how precisely work and/or communications with users was recorded. Enforcing overly detailed rules would discourage users, it possibly leads abandoning of the system. As described in the previous section, many users already had experience with the old system and, luckily, there were few major disruptions. Best-practice methods for using CODA is now being formed as users become more familiar with it. Now we are currently using the methods as baselines to develop rules for CODA usage.

(5) Gradual expansion and modification of features: Since the issue management system is directly related to the business of the organization, the manner in which it is used has a major impact on work efficiency. However, it is difficult to appropriately decide on the necessary settings and usage rules from the very beginning, and fixating on those things means that more time will be required to get the system starting to operate. When we decided to synchronize the start of full-scale CODA production with the introduction of the JSS2 system, we also decided to have a certain level of tolerance about changing system settings and how the system was used after it was up and running. Adding or changing fields in Redmine is easy. Adding projects or trackers to deal with additions and changes of the business are simple as well. It is also possible to move existing issues to a different project or a tracker. As such, we were able to implement improvements to usage and settings in a relatively smooth manner, which came to mind as we became proficient in the features of Redmine.

(6) Selection of an easy-to-use tool: When using a tool such as an issue management system daily, it is important to give users the feeling that the tool is helpful and easy-to-use. This paper has already given several examples, Redmine has convenient features such as the following: descriptions and notes that use a Wiki format and allow for easy reference to other issues in CODA or external URLs; queries and full-text search; and customizable listing of issues. JQuery UI is used

as the browser User Interface (UI). It offers a UI that is appropriate for contemporary web applications. In addition, appropriately installing plugins published on the internet creates the system which is even easier to use.

4. Hints on Definitions and Settings of Redmine

As indicated in the previous section, it is easy to reflect changes to the definitions in Redmine. It is software to be set up easily. However, there are a few points to consider so that the program can be used with appropriate definitions. This section presents hints on installing and setting up Redmine that were discovered through our experiences implementing and using CODA.

First, this section clarifies and organizes the structure of Redmine definitions, which might be slightly difficult to understand. Then, it describes tips to efficiently make definitions in Redmine, to create a system that is easy for users, and the criteria of project division. Furthermore, the use of plugins are discussed.

4.1 Clarifying and Organizing the Structure of Redmine Definitions

Redmine allows virtually all the definitions to be made from the administration screen of the browser. However, although the individual screens are easy to be understood, determining how various definitions are related to each other may be difficult, which tends to generate confusion when settings are being made. This can be particularly confusing for users who are unfamiliar with Redmine.

Figure 6 shows the top screen of Administration. On this screen, individual setting categories are displayed in a flat list, making it difficult to understand what should be defined first, and how each definition is related to which of other categories.

As an example of individual administration screen, Figure 7 shows the settings screen for custom fields. In addition to Format, Name, Description, and Possible values, this screen has checkboxes at the bottom right for “Trackers” and “Projects.” “Trackers” and “Projects” at the bottom right are also displayed in the list of Administration top screen, but there is no explanation of the relationship between the top screen and the custom fields screen, the relationship is rather unclear. (The reason for displaying “Trackers” and “Projects” at the bottom right of custom fields is discussed later.)

Although guide books of Redmine that are sold in stores do offer explanations of individual settings, unfortunately, the relationships between settings are rarely explained. To address this, Figure 8 presents an overall structure showing the relationships between major definitions of Redmine. This paper will clarify and organize Redmine structure based on the picture.

First, Redmine definitions are divided into Logical Component Definitions and Actual Entities Definitions. Both of these types of definitions are indicated in the figure by arrows on the left-hand side.

In the figure, “(1) Role Definition Layer” defines the roles of users. A “role” is defined for each functional role. For each of the roles, various actions which users can perform in Redmine is permitted or prohibited using checkboxes. Roles model the authority of actual users. In regard to this, definitions for actual individual users are made in “(5) User Definition Layer” at the very bottom of the figure. Details on this are provided later.

Below “(1) Role Definition Layer” is “(2) Issues Definition Layer.” Here, definitions related to things such as issue content, status transitions, and actions for each role are created. In “Issue statuses,” which is on the left in this layer, definitions are created for status names of issues and attributes. Here, it is rather difficult to understand that status definitions are not status transitions. The definition being given here is each status that is used as input for workflow definitions. Status transitions are defined in “Workflow,” which is described later. “Custom fields” and “Standard fields” are discussed here, which are shown on the far right of the figure. Redmine is able to define not only standard fields such as assignee, start date, and due date, but also original custom fields. “Duration of Resolution,” shown in Figure 7, is an example of the custom field which holds a selected value from a predefined list of values. On this screen, an administrator can set possible values and default one, and whether the field is required or not, and so on. “Tracker” definitions appear to the left of “Custom fields” and “Standard fields” in Figure 8. Lists of standard fields and previously defined custom fields are shown in the “Tracker” settings screen, which

designate what fields are used or not in the specific tracker by checkboxes. We now have all the definitions required to set actions of issues.

Actions of issues in Redmine are determined by "Workflow," the final category in "(2) Issues Definition Layer." Workflow combines previously defined Roles, Issue statuses, trackers and, for each tracker, sets what roles are allowed to transition an issue from what status to what other statuses. Additionally, for each field defined in the tracker, settings are done that determines whether the field is read only, required, or neither when a certain role manipulates an issue with a certain status. For example, it is possible to specify that "Assignee" and "Due Date" are not required for the status "New," but are required for statuses other than "New."

Let us now look at Actual Entities Definitions. A Project is an entity that contains data such as issues, Wikis, documents, and forums. It is possible to make multiple projects based on the needs of business. Projects are differentiated from each other by their names and summary descriptions. In each project, things such as Redmine features (issue tracking, Wikis, Forums, Time tracking, etc.) being used, trackers being used in the project, and custom fields are defined. Although it is not shown in the figure, when working with a repository, the repository is defined. When performing time tracking, names of activities are also defined.

"(5) User Definition Layer," at the very bottom of the figure, defines each user to Redmine. It can also gather multiple users together to be handled as one group. The purpose of groups is to perform actions such as addition of users to a project as a whole, and optionally set the assignee of an issue to a group of users rather than to an individual user.

To get users to participate in a project, roles are assigned to users (or groups) and users are getting members of the project, as shown in "(4) Role Assignment Layer." Permissions defined for roles in "(1) Role Definition Layer" becomes the permissions of those users or groups in the project, and settings for the status transitions and field attributes defined for those roles in Workflow are applied.

The roles of "Non member" (person who is registered as a Redmine user but not a member of the project) and "Anonymous" (person who is not registered as a Redmine user, or a user that is not logged in) are also defined in Redmine projects. However, these are roles that are envisioned for projects which are public on the internet mainly. (For example, you could browse issues and documents at http://www.redmine.org/ without registering as a member because it is allowed to do so for "Anonymous" role). These roles are not explained in this paper any further. It is assumed that users are participating as members of a project.

During the installation of Redmine, a sample series of definitions is automatically created, including roles, trackers, workflows, and projects. Examining the definition samples at installation while referring the above explanation and Figure 8 will be helpful in understanding Redmine's overall structure during customization.

Finally, here is the explanation why "Trackers" and "Projects" checkboxes are displayed at the bottom right of the custom fields settings screen. According to the explanation thus far, definitions are considered to be made in the following order: first, custom fields are defined; next, the custom fields are designated to be used with a tracker; then, the custom fields are designated for use with a project. However, when adding a custom field to an existing tracker or a project, it is convenient to be able to designate trackers and projects to be used, when the custom field is defined. In addition, when changing the definition of a custom field, it is also useful to be able to add or delete trackers and projects at the same time. There is no clear documentation about this, but it appears that the "Trackers" and "Projects" settings at the bottom right of custom fields are shown as a shortcut to allow users to quickly make such changes.

4.2 Hints and Tips for Definitions and Settings

This section provides useful tips, discovered from experience of operating CODA, for creating a user-friendly system, while definitions can be made in efficient manner of Redmine.

4.2.1 The "OR Rule of Role Settings"

As shown in Section 4.1, Redmine definitions are structured systematically and are related to each other. Although their well-established structure is a strong advantage, the number of definitions can grow

extremely large and become difficult to maintain if definitions are created without foresight. In particular, increases of roles and trackers must be carefully observed. For instance, if four kinds of roles and trackers are each defined, workflows must be defined for every tracker and for each role, resulting in a total of 4 × 4 = 16 workflows. If the number of roles increases by one and the number of trackers increases by two, the number of workflow definitions will be (4 + 1) × (4 + 2) = 5 × 6 = 30, nearly doubling. If many workflows need to be completely changed, the burden of maintenance will be larger and it will make more mistakes to occur.

In the interest of simplifying this to a certain extent, the below points are very helpful when assigning users (or groups) to projects:

(1) It is possible to assign multiple roles to one user (or one group).
(2) Permissions, which are assigned to a user (or a group) of the multiple roles, are "OR" operated.

In this paper, this is referred to as the "OR rule of role settings".

A detailed example is given below.

Assume that an issue's workflow is a simple transition of "New" → "In Progress" → "Completed" → "Approved", and the transition to go back from "Completed" or "Approved" to "In Progress." Members of the project are general user A, manager B, and X who is in charge of project maintenance. All members are able to perform non-privileged issue operations (creation of new issues, transitions of statuses to "In Progress" and "Completed," addition of notes, field updates, etc.). However, we would like B to be the only one who can change the status of issues to "Approved." We would also like to give permissions of maintenance related operations (category management, document deletion, addition of news, etc.) to X only.

Figure 3 shows the registration of users as members to a certain project. In the figure, members are registered in the project as follows: A as Role 1 of "General;" B as Role 2 of "Approver;" and X as Role 3 of "Maintainer." They have the authority to manage issues as stated above.

Now, suppose we add the "Under Inquiry" status to indicate that, for example, an expert outside the organization is being consulted. This addition would require that the all workflows of Roles 1, 2, and 3 to be changed. Another example is that, in order to make changes such as assigning "deletion of documents" authority to all members, we must grant that authority to Roles 1 and 2.

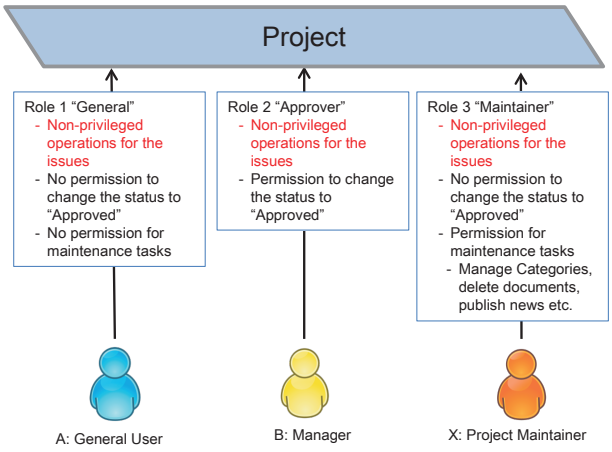

Figure 3 Role Settings and Member Assignment (1)

Conversely, Figure 4 shows settings that uses "OR rule of role settings", with a single user (or a single group) being assigned to multiple roles when registered as a member. In this scenario, although A has only Role 1 of "General," while B has both Role 1 of "General" and Role 2 of "Approver." In addition to the authority of Role 1 of "General," B can also change issues to "Approved" using the authority of Role 2 of "Approver." Similarly, X participates in both Role 1 of "General," and Role 3 of "Maintainer," X can perform actions such as category management using the authority of Role 3 of "Maintainer" in addition to the authority of Role 1 of "General." In this scenario, workflow changes related to non-privileged issue operations, such as the addition of the status "Under Inquiry," can simply be accomplished with changes only to Role 1 of "General."

Moreover, "deletion of documents" authority can be granted to all members by giving the authority only to Role 1. Although leaving the authority of "deletion of documents" in Role 3 intact does no harm, it is more appropriate to delete it because that authority is covered by Role 1 of "General," through "OR rule of role settings."

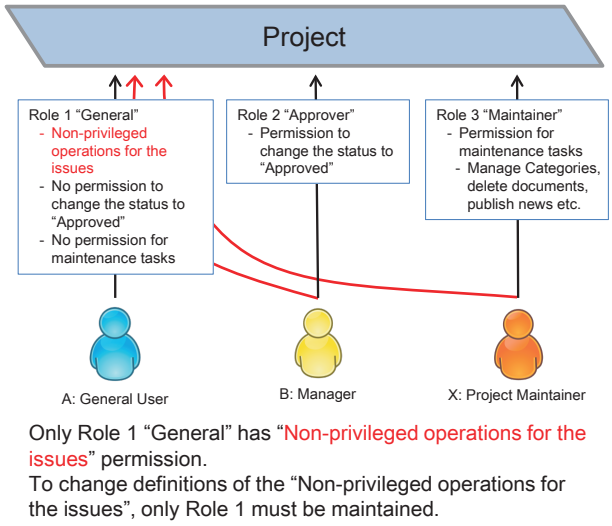

Figure 4 Role Settings and Member Assignment (2)

This is how "OR rule of role settings" makes it possible to reduce the maintenance burden for roles, permissions, and workflows. When CODA was initially introduced, definitions were created as in Figure 3. However, role and workflow maintenance became so complicated as the scope of system usage widened that definitions were changed to the method using "OR rule of role settings" shown in Figure 4, approximately a year after production run of CODA.

Table 1 gives a summary of the current roles in CODA, which were established using the methods presented here. Among the roles in the table, "Regular" role is applied to users participating in a project normally. It is the most basic role and allows users to perform actions such as issue creation and update, and status changes (excluding some statuses). It has read-only permission for documents in Redmine.

"File Clerk" role has only the authority to manage documents (adding, editing, deleting) in Redmine, and is not granted for other authorities nor make status transitions. This was established because of a requirement to limit the ability to manage documents to only limited number of designated persons in some projects. The designated persons participate in these projects in both "Regular" and "File Clerk" roles, allowing them to both read and manage documents with "OR" operation of both roles. Contrarily, other users participate only in "Regular" role, they are able to read documents but not manage them. For other projects, participants in "Regular" role are simultaneously set up to participate under "File Clerk" role, allowing them to manipulate documents with no impediments.

"Approver" role is assigned only to managers. It has only permission of status transition to make issues "Approved" and return "Approved" issues to "Return." On the other hand, it has no "Roles and permissions" authority. It means that this role has very limited function. But managers are also assigned "Regular" role at the same time, they are able to make all status transitions with "OR rule of role settings." Moreover, "OR rule of role settings" makes it possible to allow managers to perform all the "Roles and permissions" authority of "Regular" role.

"Maintainer" role is for carrying out project maintenance work. It has the authority to manage project related changes such as project description, activation and deactivation of trackers; management of public queries and news; deletion of Wiki pages. These types of authority are not assigned to other roles than "Maintainer." While "Maintainer" role has such authority, it does not have any status transition in workflow. It is a role that is assigned to only those who are in charge of maintenance in addition to "Regular" role.

These four roles above have authority to change data or settings of CODA. There is an "Observer" role that differs from these, and it is a read-only role. On some projects, there is a requirement to give some persons an ability only to read through issues, Wikis, documents, and the like. "Observer role" is used in these situations.

Some tasks, such as creating projects, managing members, and deleting issues, are not assigned to any of the roles listed above. Redmine administrators are the ones who handle such operations because they are able to perform all operations without being limited by the role settings. How it is appropriate to give these authorities to whom varies according to the organization. Possible ideas are; structures such as giving authority to Redmine administrators only (centralized structure), giving authority to maintainers of each project (decentralized structure), and giving authority to all members (flat structure) etc. CODA is operated under the centralized structure style.

4.2.2 Project Participation using Groups

As shown in Figure 8, there are two possible methods of assigning users to projects in Redmine: assigning each individual user to a project as a direct member, or creating groups and assigning users to a project in group units.

CODA uses the group assignment method in principle. When making changes such as personnel rotation, individual user assignment requires changes to be made to each project in which the individual participates. On the other hand, with group units we can simply remove the person transferring out from the group they belonged to and replace them with the person transferring in, eliminating the need to change the project participation status for each individual. Particularly for situations in which there is a large number of projects and roles, the group assignment method keeps the burden of maintenance work low and reduces the possibility of making mistakes.

4.2.3 The "AND Rule of Field Settings"

One thing to consider when creating trackers is the field settings that will be used with those trackers. Status transitions constitute workflows and they are often consistent in an organization (for instance, the transitions shown in Section 4.2.1: "New" → "In Progress" → "Completed" → "Approved.") in general. Conversely, the fields which are used often differ based on the nature of the business in the organization.

A straightforward implementation method is to prepare multiple custom fields tailored to each situation, then individually define appropriate trackers for them. The drawback of this method is that multiple trackers are created for fields in use that only differ slightly. Workflows must be defined for the number of trackers times the number of roles, thereby increasing the maintenance burden. The best way to address this issue is to avoid increasing the number of trackers as much as possible and using the same ones in many situations. In Redmine, dividing things up using multiple projects makes it possible to achieve the appropriate use of fields shown here. This paper refers to this method as the "AND rule of field settings." Figure 5 shows a detailed example of this method.

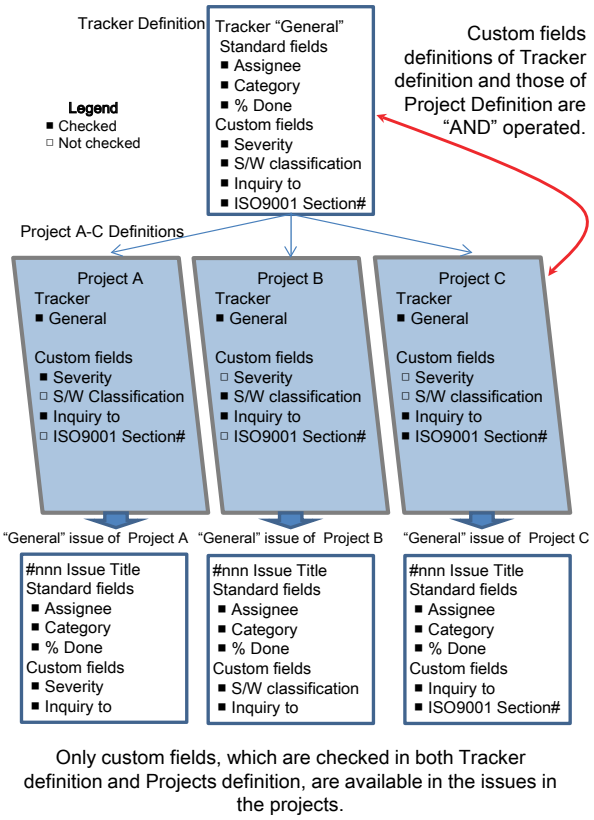

Figure 5 Example of the AND Rule of Field Settings

"General" tracker is a multipurpose tracker handling general issues. It defines the organization's basic workflow: "New" → "In Progress" → "Completed" → "Approved". On the other hand, the details of business vary, they are for system operation, user support, and quality management system (QMS) activities based on ISO-9001. Persons who are in charge and the meetings to be held are different between the businesses. However, workflow is the same within the organization, it is better to use the same "General" tracker for all of the jobs. To make this possible, the following definition is valuable to utilize only the fields that are relevant to the business by using the "AND rule of field settings."

First, custom fields that "can be" used are defined to "General" tracker. Three projects that actually store issues are to be defined for each subject area of business: A (system operation), B (user support), and C (QMS activities). In each project, the custom fields that will be used "actually" for the business are checked, and the fields that will not be used are not checked. Using Project A of Figure 5 as an example, "Severity" and "Inquiry to" fields are checked, these

fields appear on Project A's "General" issue. Conversely, "S/W classification" nor "ISO9001 Section#" are not checked, they don't appear. This is because only fields for which both the tracker field definition and the project field definition are checked (i.e., those that fulfill the AND condition) can actually be used. This same setup causes "Inquiry to" and "ISO9001 Section#" appear on Project C's "General" issue, but "Severity" nor "S/W classification" don't appear.

Utilizing this method makes it possible to use one tracker to deal with situations in which the workflows are the same but the contents to be managed (fields) are different. This makes maintenance easier, and gives the following benefits to users.

(1) Only fields that are relevant to the subject area of business are shown in the issue. There is not any field displayed on the screen which are not for the business. Hence, it is easy to focus on the business.
(2) When a person is in charge of multiple subject areas of business, he/she can use the same tracker (for example, "General") even when working on different projects. This lowers the number of tracker choices and reduces confusion.

4.3 Project Division Criteria

Although multiple projects can be used in Redmine, not much information is provided on the internet or in existing publications regarding when it is appropriate to divide projects. However, based on our experience with constructing and operating CODA, project division is a major factor to effective use of Redmine. The principles of project division likely differ in accordance with the culture and policies of the organization which use Redmine, whether or not to divide projects is an important issue to consider when the scope of business is expanded. To this end, this paper introduces project division criteria based on our experiences implementing CODA.

(1) Divide projects when participating members are different for security reasons and the like. As discussed in Section 3.2, vendor members also participate in CODA. Matters that are being handled with vendors are managed in the projects that they participate in, while other matters are managed in projects that they do not participate. Redmine has a feature called "subproject" which makes projects to be connected in a "parent-child" relationship, and that feature is effective in such cases. Projects that vendor members participate in are "sub-projects" of projects that vendors do not participate in, users of the Supercomputer Division can search issues, work with the documents of both projects at once. Meanwhile, vendor members are only participating in the sub-project, they cannot see content of the parent project.
(2) Divide projects when, although participating members are the same, their roles need to be changed between the projects. The roles currently used in CODA are shown in Table 1. The role of Regular are assigned to users in the normal case. But, default role is Observer for projects that store deliverables. Read-only role of Observer is assigned in order to protect issues and attached files in a secure way. Adding new deliverables or updating deliverables to new revisions is only performed at the time of delivery. Consequently, only during this period alone, Regular role is added for specified users and deliverable issues are added or updated. During this time there is a possibility that an issue could be modified incorrectly or attached files could be deleted by mistake. Redmine records changes to issues, it would be possible to track such mistakes.
(3) Divide projects when the required fields are different, or the frequency of updates significantly differs, in accordance with the scope of business. CODA is used to manage a variety of businesses in the Supercomputer Division. Major portion of issues are related to the operation of JSS2. In the organization, there is a group of persons who is mainly in charge of user support such as visualization of the computational results. Projects are divided between the operations of JSS2 and visualization support, with the "AND rule of field settings" outlined in Section 4.2.3. Redmine has a feature that notifies members by email of issues updates etc.. User can select options of mail notification settings such as "For any event on all my projects", or "For any event on the selected projects only" and so on. By dividing projects based on the scope of business in this way, users can set things up such that they receive all notifications for just the projects

within the primary scope and receive notifications from other projects only “you watch or you're involved in (eg. issues you're the author or assignee)”. This allows them to appropriately limit their email notification flood.

(4) Divide the project related to Redmine maintenance. The reasons behind this are predominantly the same as those for scope of business in the previous paragraph; dedicated project for CODA development and maintenance is held in order to separate tasks from other operation-related projects. It is convenient that features and changes which are specific to the project can be tested here before being applied to the production in other projects.

(5) Divide projects when the options of “Category” field are to be split based on the subject area of business, and make selecting options easy when editing issues and set the criteria for query. This is discussed in 4.3.1 on this aspect.

4.3.1 Using the Standard “Category” Field

One of the benefits of project division is active use of the standard “Category” field that Redmine offers. “Category” works in a slightly different manner than other fields and it is strongly related to projects, it is worth to discuss here.

Value ranges or options of standard fields are common in a Redmine installation. Custom fields are fields that can be defined in an installation and their attributes, and options settings are the same across the multiple projects in an installation. In comparison, “Category” is a list form field, which means that users select a string from the predefined list of strings, and what is unique is that the list of available strings is defined independently in every project. This makes it possible to prepare options (list of strings) to suit the nature of the business handled by each project. This is useful to make finer classifications of business or add specific keywords for queries. For example, in the projects presented in Section 4.2.3, the Maintainers can provide options that are suited to the business of Project A (system operation), Project B (user support), and Project C (quality management system), respectively. Although this is not a major benefit when there is only one project in an installation, it is highly recommended making use of, when multiple projects are held.

During initial construction of CODA, there were only 1 or 2 two projects, so we did not prepare unique options of Category for individual projects. As usage of CODA expanded, the number of projects increased and the jobs being handled also became more varied. As such, it is now planned to set category options which are unique for each project.

The Categories field has two more characteristics that other fields do not. The first is a feature that automatically assigns an issue assignee according to the option value. This feature is very convenient when used in a well prepared manner, at present there is no plan to implement it in CODA. The second is the ability for users to directly add options while editing an issue. This feature may make number of options of Category much larger without control in exchange for its convenience. In CODA, updating options of Category are allowed only to Maintainer role. We are currently considering whether to allow it for general users.

4.4 Use of Plugins

Various plugins are available in the internet which expand features of Redmine. Many plugins are free open source software. These plugins can make Redmine usage more pleasant or maintenance and management easier. At present, CODA is using six of them.

Although consideration for using plugins are likely the same basically as those for using any OSS, this paper introduces a brief presentation about what is considered when installing plugins in CODA below:

(1) Whether updating of information and addition of features are done regularly or not. There is a risk that plugins are not working correctly because of the version upgrades of Redmine. This is why it is important to have ongoing maintenance.

(2) Whether explanations and documentation are rich and sufficient or not. Although it depends on the features of the plugin, when documentation is insufficient it is safer not to move forward with using it.

(3) Check users’ experiences and reviews on the internet. These information could reveal how developers respond to things and whether maintenance is ongoing.

(4) Choose simple plugins whenever possible. Considering the risk that a plugin may stop

working when Redmine is updated, it is safer not to be too reliant on plugins. As such, more care is needed when considering a sophisticated, complex plugin as opposed to a simple one.

In the experience of CODA, there were two cases in the past when the Supercomputer Division considered introducing a plugin, but decided against it. In one of these instances, the primary reason was lack of information; documentation was insufficient and there were various questions being ignored in past question and answers. The other instance was because of version incompatibility: the plugin had not been maintained and was not compatible with recent Redmine upgrades, there was number of blog articles that the plugin started working after correcting the incompatible issues through trial and error.

5. Future Outlook

This paper concludes in this section with a discussion of the future outlook for CODA and Redmine.

5.1 Future Expansion of CODA

Approximately one year after beginning full-scale production run of CODA system, the expansion of its use is generally going smoothly, and it has become an indispensable tool in the Supercomputer Division. Moving forward, we would like to further perfect its use, mainly focusing on the points below.

(1) Updating to the new Redmine 3.1 system: At the time of writing, we have been using the Redmine 2.5x system, the version available when full production of CODA began. Since then, version 3 of Redmine was released (at the time of writing, the most recent version is 3.1.0). Many features that improve usability were added in version 3. We are planning to upgrade to that version quickly and use it to improve productivity of work.

(2) Promoting standardization in areas such as issue creation and completion: While users are more familiar with using the software than when it was first running, we are still on the way of standardization of rules such as granularity of issues, how detail description is needed and conditions of issue completion. This is true not only for the efficient use of the tool, but also for quality management of organization, namely, the standardization and transparency/visualization of business. In the Supercomputer Division, promotion and revision of workflow standardization from the perspective of ISO-9001 is underway. We would like to coordinate this with the manner how CODA is used and apply CODA as a tool to better support the organization's activities.

(3) Working with a version control system: Redmine can be used with version control systems such as Git. In CODA, this feature is only used as a trial use for modification of source program of Redmine and creation and maintenance of themes (CSS (Cascading Style Sheet) used for browser look and feel.) With JSS2 operation, commands for users and operational utilities are developed and maintained, as well as modifications of system parameter files are managed. We would like to enhance CODA to work with a version control system so that they can work together in order to relate the modifications of programs and parameters to CODA issues which describe system events or malfunctions.

5.2 Expectations to Redmine

As discussed thus far in this paper, Redmine is a versatile issue management system. It could be a very satisfying software program overall for using it as such as CODA, based on our experiences of improvement of settings and usage. This section presents views of current status and future expectations regarding Redmine itself and its ecosystem.

When thinking of using OSS, it is very important to consider whether development is active and whether there is a substantial supporters' community. Redmine is being actively developed, there are many publications about it, and active supporters' community is working. Therefore it is expected that the addition of new features will be going well and more information on how to use Redmine better will be provided in the future.

When considering to use it as the project management for development, there is a number of areas in which it is inferior to programs such as Microsoft Project in terms of features and ease of use.

However, fare-paying plugins that compensate them are becoming available in the market. In addition, there are companies that offer Redmine ASP (SaaS) services, or offer designing, installing, and maintenance of Redmine implementation. Having such a comprehensive ecosystem like these is important for an OSS to get popularity and take root. This is incredibly encouraging.

6. Conclusion

This paper discussed the characteristics of Redmine; experiences and issues the Supercomputer Division had with the prior issue management system, and the introduction process of CODA; unique hints to set up and use Redmine better, which derived from our experiences of constructing and using CODA; and the future outlook for CODA and Redmine.

I would like to appreciate all the people of the Redmine community, including those who work on the development and maintenance of Redmine itself and its plugins; the members of communities who spend their time to such as disseminating information on the internet, providing articles on books and magazines, and holding conference events; and the companies that offer SaaS, installation and maintenance services.

In addition, I would like to appreciate all the users who are providing many thoughts and much support throughout the planning and use of CODA.

References

1) Naoyuki FUJITA, " *A purpose of JAXA next generation supercomputer installation and its composition outlines*", The 46th Fluid Dynamics Conference / 32nd Aerospace Numerical Simulation Symposium, 2014, https://repository.exst.jaxa.jp/dspace/handle/a-is/459711 (last visited August 21, 2015). (Japanese)
2) "*Who uses Redmine?*", http://www.redmine.org/projects/redmine/wiki/WeAreUsingRedmine (last visited July 26, 2015).
3) Yoshinobu KATO, "*An In-depth Look at Development Support Tools 2013*," Nikkei Systems, June 2013 issue, pp. 49-51. (Japanese)
4) Yuichi MATSUO and Masako TSUCHIYA, "*Issues and Tips on the Operation of the JAXA Large-scale SMP Cluster, Large-scale SMP Operation Working Group Report: Attachment 39-1*," Scientific Systems Association. 2006, http://www.ssken.gr.jp/MAINSITE/download/wg_report/smpo/t39-1.pdf (last visited July 26, 2015). (Japanese)
5) "*ISO 9001:2008 Quality management systems—Requirements*," ISO (International Organization for Standardization). 2008 , http://www.iso.org/iso/catalogue_detail?csnumber=46486 (last visited August 24, 2015).
6) Ryutaro YAGUCHI, "*Success Guaranteed ! The Vital Points of Redmine Introduction,*" Nikkei Systems, September 2013 issue, pp. 50-55. (Japanese)
7) Yoshihito KURAKI et al, "*Redmine: Project Management Made Easy!*", 2009, Impress Japan.(Japanese)

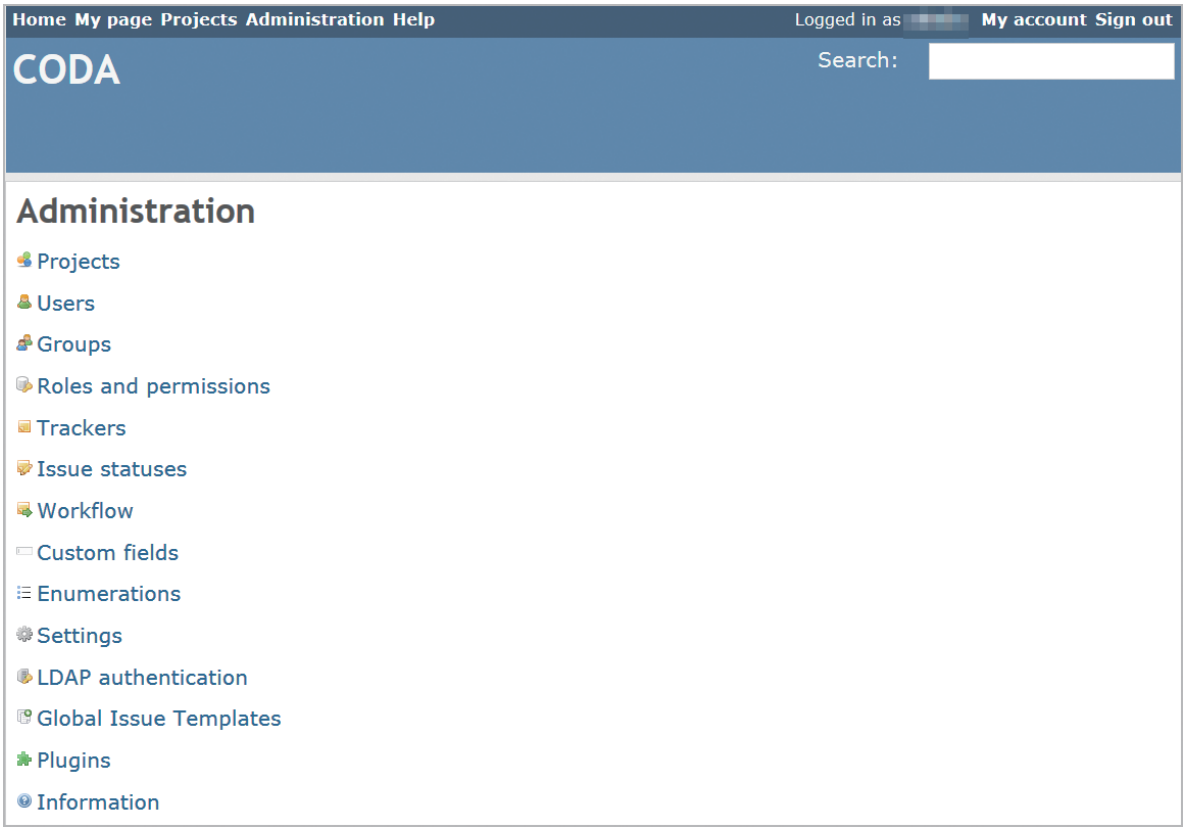

Figure 6 Administration Top Screen of Redmine

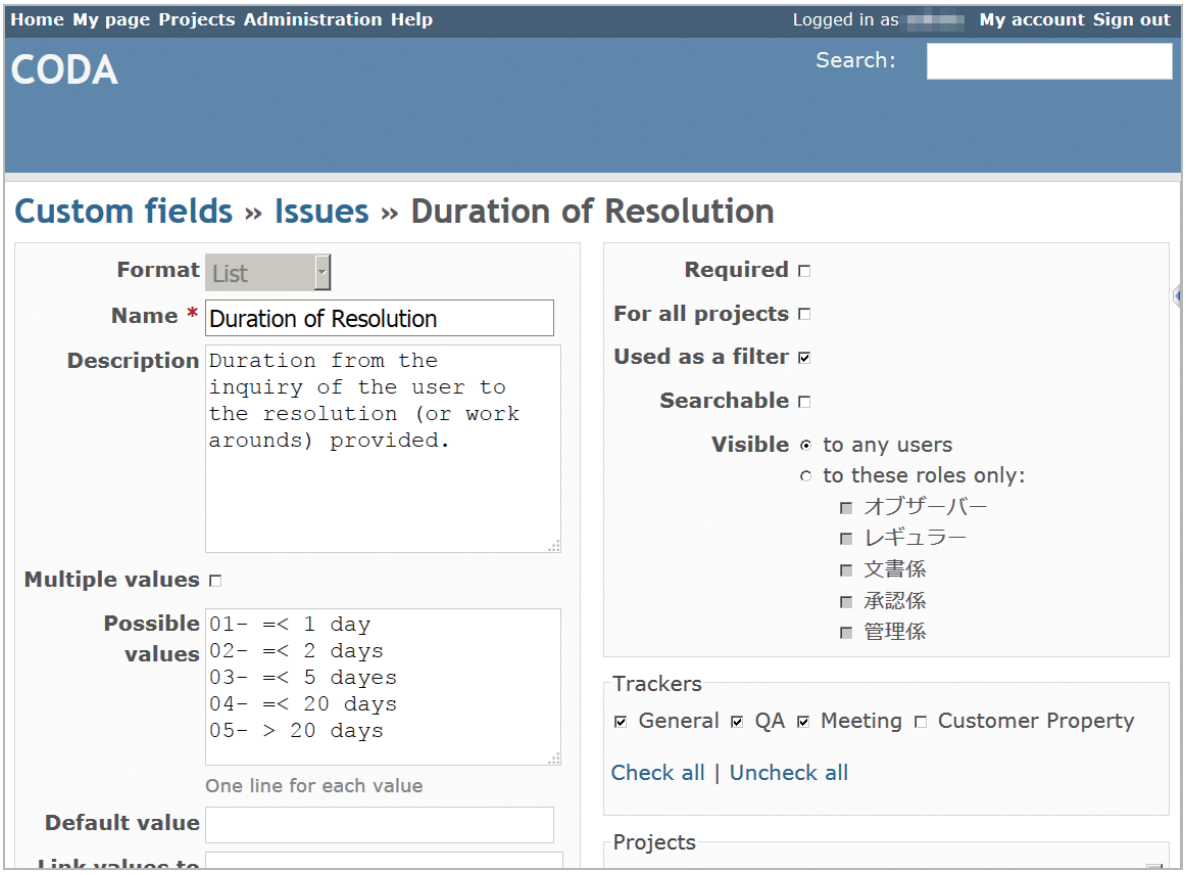

Figure 7 Sample of Administration Screen: Custom fields

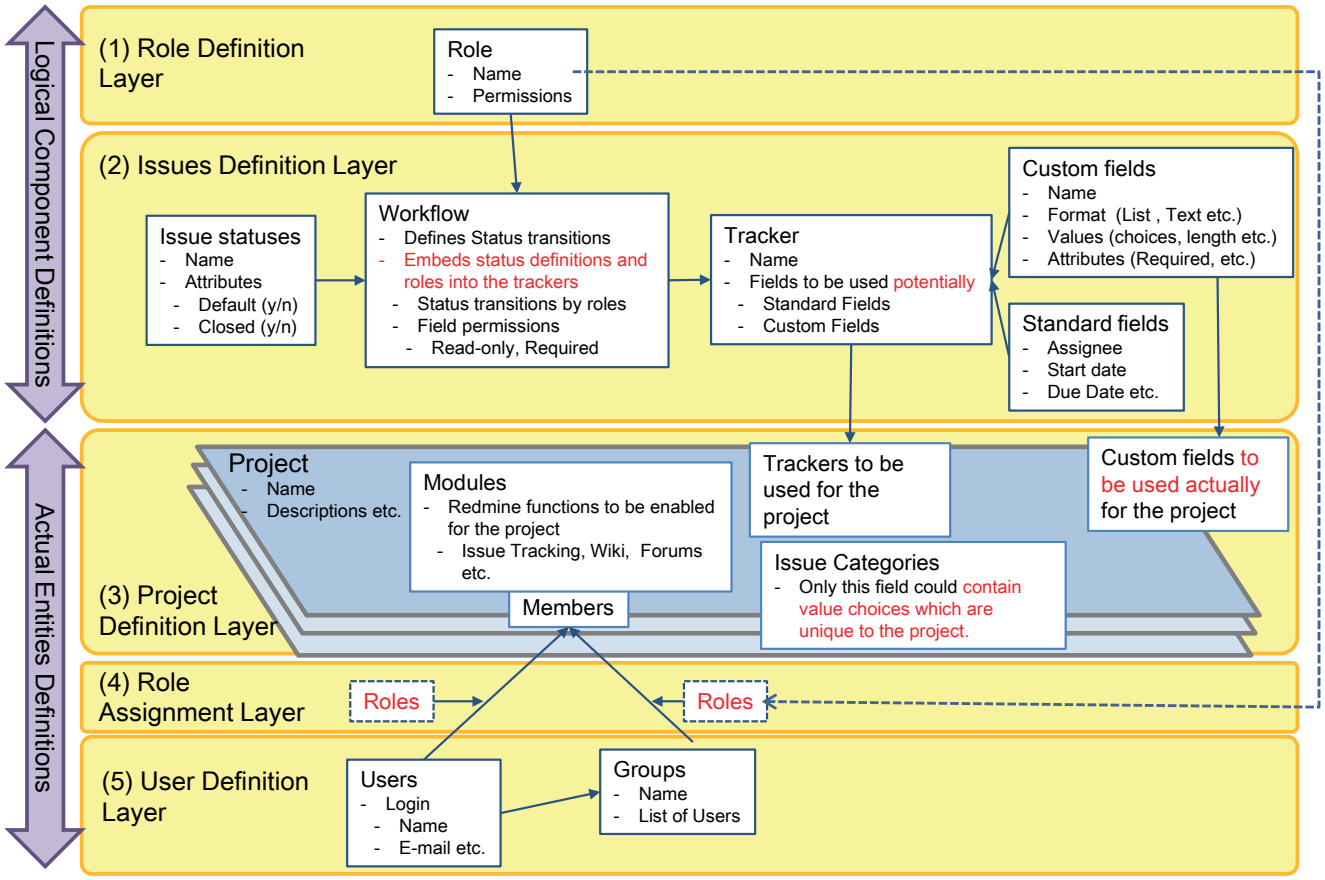

Figure 8 Structure of Major Definitions of Redmine

Table 1 Roles defined in CODA

Role Name	Description	Status Transition	Roles and permissions
Regular	Performs regular tasks such as handling of Issues, editing Wiki. General users typically work with this role.	All transitions available except for the ones which are allowed only to Approver role.	Permissions available which are needed to regular tasks except for the ones only for File Clerk role and Maintainer role.
File clerk	Performs management of documents.	None	Only add, edit or delete documents.
Approver	Gives approval as a manager.	Transition to change issues "Approved", and "Approved" back to "Return" ONLY.	None (see Note 2 below)
Maintainer	Performs maintenance of the project	None	Only maintain Project related Settings.
Observer	Performs only read contents (issues, documents etc.)	None	Only view Issues, Wikis, documents etc.

As of August, 2015

Note 1: In Roles and permissions, no role has permissions to add projects, manage members, delete issues, and other administrative tasks. Only Redmine administrator can perform such tasks in CODA.

Note 2: for English translation update: "Add Issues" or "Edit Issues" permission is needed to make "Approver" to be visible in Workflow definition of Administration. (In recent versions of Redmine, it is needed, although it was not needed with Redmine 2.5.0.)

JAXA Research and Development Report JAXA-RR-16-002E

CODA: Ticket Management System to Support JSS2 Operation and Assistance to Users
- Redmine Implementation and Hints of Its Usage -

Edited and Published by: Japan Aerospace Exploration Agency
7-44-1 Jindaiji-higashimachi, Chofu-shi, Tokyo 182-8522 Japan
URL: http://www.jaxa.jp/
Date of Issue: September 26, 2016
Produced by: Matsueda Printing Inc.