Be social, be agile: team engagement with Redmine

Runa Briguglio, Marco Xompero, Marco Riva, Chiara Selmi, Cristiano Urban, et al.

Runa Briguglio, Marco Xompero, Marco Riva, Chiara Selmi, Cristiano Urban, Nicolò Azzaroli, Luca Oggioni, Marcello Scalera, Armando Riccardi, Andrea Balestra, Marco Landoni, Davide Fierro, Letizia Caito, Riccardo Smareglia, Massimo Sponza, Cristina Knapic, Giorgio Pariani, Sergio Poppi, "Be social, be agile: team engagement with Redmine," Proc. SPIE 12187, Modeling, Systems Engineering, and Project Management for Astronomy X, 121870R (25 August 2022); doi: 10.1117/12.2628729

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2022, Montréal, Québec, Canada

Be social, be agile: team engagement with Redmine.

Runa Briguglio[a], Marco Xompero[a], Marco Riva[a], Chiara Selmi[a], Cristiano Urban[a], Nicolò Azzaroli[a], Luca Oggioni[a], Marcello Scalera[a], Armando Riccardi[a], Andrea Balestra[a], Marco Landoni[a], Davide Fierro[a], Letizia Caito[a], Riccardo Smareglia[a], Massimo Sponza[a], Cristina Knapic[a], Giorgio Pariani[a], and Sergio Poppi[a]

[a]INAF - Istituto Nazionale di Astrofisica

ABSTRACT

System engineering and project-team management are essential tools to ensure the project success and the Redmine is a valuable platform for the work organization and for a system engineered approach. We review in this work the management needs related to our project, and suggest the possibility that they fit to many research activities with a similar scenario: small team, technical difficulties (or unknowns), intense activity sprints and long pauses due to external schedule management, a large degree of shared leadership. We will then present our implementation with the Redmine, showing that the use of the platform resulted in a strong engagement and commitment of the team. The explicit goal of this work is also to rise, at least internally, the awareness about team needs and available organizational tools and methods; and to highlight a shareable approach to team management and small scale system engineering.

Keywords: Project management, system engineering, redmine, task tracking, test plan.

1. INTRODUCTION

1.1 Forewords

The Redmine (RM) is a valuable tool for team and project management (PM) and includes also some useful features for the System Engineering (SE) of the project. Here we use the term *Redmine* to indicate a class of software products, in cloud or server based, all sharing some common functionalities such as the issue tracking, the ticketing, the creation of Gantt charts and the resources allocation. The RM in its original version is open source and may be freely downloaded and installed. Starting from this, many commercial versions were developed including additional plug-ins and features, an help desk system, the customer management, the agile board and even the pages branding with the company colors. The RM is then a multi-user tool to be exploited by large companies or small teams. It is a matter of facts that most of the available features are specifically intended for software development, and for large companies with complex internal organization and management.
The authors work at INAF, the Italian Institute for Astrophysics. Just to give the context, INAF is a public research institute with approximately 1000 staff, pursuing both scientific and technological research through many active projects (or activities). Excluding very large programs with world-wide consortia, a typical project in this context is a 2,3 years activity, carried out by 3 to 10 persons, with 4,5 milestones and deliverables. For instance, it may be the Preliminary Design or Final Design Review of an instrument, or the development of a numerical code, or the preparation and analysis of an observing program. We try now to summarize for such a scenario some common situations from a team management perspective:

- very small project teams means many projects and many project leaders, plus we shall consider also the institutional (e.g. outreach) or administrative activities; each of them come with its own specific needs in terms of activities organization. Each team want its own tool or customization.

Further author information:
Runa Briguglio: E-mail: runa.briguglio@inaf.it, Telephone: +39 055 2752200

Modeling, Systems Engineering, and Project Management for Astronomy X, edited by George Z. Angeli, Philippe Dierickx, Proc. of SPIE Vol. 12187, 121870R
© 2022 SPIE · 0277-786X · doi: 10.1117/12.2628729

- the interface between projects or teams shall be addressed, for instance for those (many) projects when the informations shall be passed outside the team;
- this point implies that having a common tool for the entire institute is attractive (e.g. to limit the management cost or to ease the excanges) but complex. For instance, the tailoring of the tool shall be distributed to intercept everyone's needs.
- a system engineered approach would certainly improve the projects output; if the platform is also a *showcase* of system engineering tools or best practices, sooner or later people will implement them in the project.
- Getting a free RM from the web is easy (especially if limited to 5 people), but you will get an issue tracker or little more: no system engineering tools, for instance.
- a research institute carries out laboratory activities, by definition; then a test management tool is extremely useful.

We guess that such a scenario is applicable to many universities or research institutes.

1.2 Goal of this paper

This work has two main purposes. On the one side, we are presenting here an analysis of (our) needs and working procedures as a team. The scheme might be useful for other teams to identify common requirements or methods. The second goal is to rise (at least internally) the awareness about team needs and available organizational tools and methods; to highlight a shareable approach to team management and small scale system engineering; to clarify and propose a roadmap for future developments.

2. A CASE STUDY: THE M4 PROJECT AND THE OPTICAL TEAM

As a starting point, we will describe in this section our very specific context, related to the M4 project.

2.1 The M4 project: general context

M4 is the adaptive mirror (the 4th in the optical train) of the ESO Extremely Large Telescope. The mirror, which is currently under construction, is a flat, 2.5 m diameter, segmented into 6 petals, each controlled by 892 actuators. ESO is the customer; the supplier is a consortium named AdOptica; our team in INAF has two main packages, namely to provide system-wide adaptive optics (AO) expertise and to design and carry out the optical calibration of the deformable mirror in its test tower OTT.
Our activities may then be un-packed as follows: during the PDR phase, acting as a consultant for the design and budgetting of the mirror and to solve AO-specific issues; during the FDR phase, designing the OTT and organizing and analyzing the calibration activities; during the MAIV phase (the current one), validating the OTT and calibrating and verifying the M4 mirror. Some additional informations may be retrieved in the references.

2.2 How are things in real life

The MAIV project duration (the one implemented in the RM) is nominally 5 years, from the delivery of the critical design review documentation to the end of the optical test activity. The immediate feeling is that such a long time gap may lead to loss of crucial informations: procedures, results, know-how. This is particularly concerning when considering the mixed nature of the project: fully technological from the point of view of the MAIT, and largely R&D when looking at very specific metrological aspects. This second point implies that often the progress is not a straight line and all the elements that converged to success shall be recorded. This is for instance the reason why the largest fraction of our work in the last years was basically risk mitigation activities, carried out in the laboratory with the goal to assess key-elements of the test procedures. For instance we created laboratory tests to validate specific measurement procedures. Such block of activities was not fully formalized and organized at an early stage of the project so that we were forced to re-create our Work Breakdown Structure (WBS): very often, a post-facto WBS. In this view, each brick in the WBS is rather a container of collected

information, than a plan of to-do activities.
Working within an industrial consortium, with many external suppliers, deadlines and milestones may be subjected to delays or de-scoping, and you will be the last to be informed. Working with a Gantt chart or with a stringent Resources Management (RM) is unuseful. An Agile approach, on the contrary, fits much better. At last, the work is often super-concentrated in a burst of activities (or *sprint* from the agile terminology), followed by long pause periods.

2.3 The team

The team is composed by people from the Arcetri Observatory (Florence) and the Merate Observatory (Milan); we are not considering here the people from the industrial consortium: they shall however be not forgotten, since their specific request, from a management and reporting point of view, are very different from ours. The territorial separation often reflected a work organization: optics in Milan, SW, analysis and simulation in Florence. We tried however not to keep strict specialization, also in order to share as much as possible the informations and the knowledge about the project.
Since a significant fraction of the activity happened during the Covid-19 pandemic, the personal situations reflect the management scheme (or vice-versa): the team is composed by single-living people, a young post-graduate, a young post-doc, two have childrens. All of us are structurally involved in other projects, also due to the long pauses and delays in the projects.

2.4 Desiderata

Starting from the considerations given above, we searched for a PM tool that helped us is our work. In particular we needed an information management system, such as a wiki page container, plus a way to wrap up the relevant informations in an easier way. This was particularly important since, with the working rhythms described above, we needed to reduce the warm-up phase and increase the team efficiency. Additionally, we wanted also to raise the project awareness, especially in the youngest components in the team.
As a very last point, we considered an institutional perspective: which are the other needs of other research teams at INAF? For instance, we are not involved in subsystems procurement, or requirements management, or RAMS analysis. Nevertheless, these aspects of a project life cycle should be kept in mind.
In the next section we try to describe the specific requirements we had in mind before setting up our RM system.

3. OUR NEEDS, OR WHAT WE ASK TO A REDMINE SUITE

3.1 At least, archiving

We need a tool to store, organize and recover the informations. This last point comes easily after the second. What do we really need to archive? Activities logging as first: they can be day-by-day, unorganized summaries of what happened in the lab, or, on the contrary, a detailed report of an experiment in its final form, ready to be circulated amongst colleagues. As second, we need to store the results of a large amount of tests, each with specific conditions or devices involved. In this regard, the possibility to store automatically some side-informations is very attractive: for instance, the script run to acquire a given dataset. Then we need to store conveniently procedures and how-to's; this point is particularly important to allow everyone in the teams to collaborate actively in the project, for instance in the data acquisition or analysis.
As a last point, the archive shall include a notification mechanism so that when someone works on something, anyone else is automatically informed.

3.2 Management of experimental data

A very desired feature is the archiving and management of experimental data, i.e. the results of laboratory tests. In this case the simple storage of data (in the form of numerical values) is not enough and we list in the following our top-desired features:

- algebraic operations on result values: from a SE point of view, laboratory tests are often carried out as verification procedures on the instruments, so that the results shall at a later stage be summed-up (e.g.) to assemble an error budget table; also comparison of values against the requirements is of interest;

- storage and association of metadata, such as environmental conditions (e.g. temperature, time, experimental setup) relevant for the specific acquisition logged;
- data presentation and discussion, for instance in the form of a plot: this is particularly attractive to have a quick data availability for reports or presentations; last but not least, this *obliges* the user to prepare a data presentation in some automatic way upon the data acquisition;
- record of the acquisition and processing software together with the data;
- linking data with requirements and vice-versa: this is of paramount importance when the compliance of the requirements shall be assessed.
- management of very miscellaneous experimental data: from a SE perspective, a requirement may be demonstrated by test, inspection, analysis, etc: so, be ready to set-up test plans (with associated data) in the form of inspections, measurements, analysis.

The ensemble of all these features shall not translate for the final user into a large, additional logging fatigue; this implies that the logging interface (i.e. additional fields to be filled in) is created in advance and is not intended to be modified during the logging period. Or: clarify in advance what do you exactly want to log and customize the RM accordingly.

3.3 Planning, organizing

A project workflow may be easy prepared with a WBS (or work breakdown structure) or a PBS (Products breakdown structure). In our case the WBS is rather an empty container to be filled with activities. At the end of the project, when all the activities have been logged diligently, the WBS will appear as the table of content of the project final report.
Another point is that our activity is concentrated in burst of homogeneous work, with all members collaborating to the same goal. This behaviour fits well to the *scrum*, from the agile management.

3.4 Engage!

Since the team is also involved in other project to fill the temporal gaps in the activity, we need a tool to engage people. In our mind, engagement means to urge co-workers to act indipendently, to take a part of the work and to carry it autonomously, to share results and difficulties for a collaborative, cross-competence response.
In order to realize it, some pre-requisites are to be considered.

- Fast sharing of news: new data acquired, new procedures released, new results.
- Start of new activities shall be planned together and communicated in time, and everybody shall easily recognize from the RM what's happening right now and where we are.
- tasks shall be well defined, with clear inputs and outputs; this applies also to the *visualization* of inputs and outputs, so that the scope of the task is immediately digested.
- the same applies for the intermediate steps of an activity or its current status: "I'm stuck", or "I've done", or "I need some more data" shall be informations code immediately available by everybody.
- instructions and procedure shall be found easily to allow a fast warm-up and to foster the autonomy.
- a playful organization and presentation of the work is not necessary, but may help. Let's be social!

3.5 Case-specific tailoring

Such a RM does not exist in reality. So our need is converted into the real possibility for customization of the platform. Again, some additional considerations apply:

- customization means distributed administrative rights, while on the contrary you would prefer to have the platform managed by a dedicated IT team;
- such a customization shall allow creating user-defined fields, or identifier, tags and views, so that to intercept specific needs on a project-to-project basis;
- special visibility, or the possibility to have custom fields visible to specific user types or projects, so that each project has its own visualization;
- general or specific purpose templates, which are projects with minimal informations entered but fully structured; the utility of such templates is to allow non-skilled users to take advantage of the RM platform with minimal development effort.

3.6 System engineering, or chapter two

System engineering (SE) is an interdisciplinary approach to enable the realization of successful systems. The reader may find more details in the references. Within a project the system engineer is a key-person; however, based on our experience, we point out that SE knowledge shall not be concentrated in a single person, rather it shall be a common attitude within the working group, regardless the individual roles. We focused then on those *good practices* that are enablers for success. For instance thinking global-wide, considering the life-cycle, operating trade-offs, pursuing the effectiveness; these are some of the SE pillars.
Good practices are something in the form of a check-list or a tool, so that when they are presented by the RM platform, each user is naturally *invited* to keep them in mind or use them throughout the project. In this regard, the RM shall be constructed to sketch out the project life cycle. Such approach implements the concept that the entire team is committed to consider a SE view on the project, including young post-docs, graduate students and even a passer-by collaborator.
Also, it could be adopted even on larger scale projects, when we consider their smaller working package or working groups. In these contexts we consider valuable the integration between professional SE tools and software (used by *official* system engineers only) and the RM used by the rest of the crew. Such integration may be implemented by proper interface to load-retrieve data from/to RM and those specialty softwares.
The next step of such concept is the implementation of model based SE (MBSE) with the RM, and some proof of concepts may be found in the references,[6],[7].[8]

4. THE M4 REDMINE PLATFORM

4.1 The home page

In our RM platform, the starting page is a highly customizable *home page*. In particular, the platform allows to create as many *tabs* as you want and to populate them with standard modules such as text fields, calendars, news and panels to visualize common queries to the database. We customized the home page with the following tabs and the result is presented in the figure.

Now on air It's a tab to show at a quick glance what's going on. It is not specifically referring to a sprint or some tasks, it is rather intended to keep the team informed and engaged. This is particularly important when people are off for a while (e.g. busy on other projects) and need to get updated; also it is used as a gentle reminder for those long lasting tasks: the internal linking brings you immediately at the core of the activity.

Get connected This a panel to summarize all the technical specifications to access the project network. This include the VPN, the workstations in the Observatory and at the OTT laboratory, the interferometer and the thermocamera.

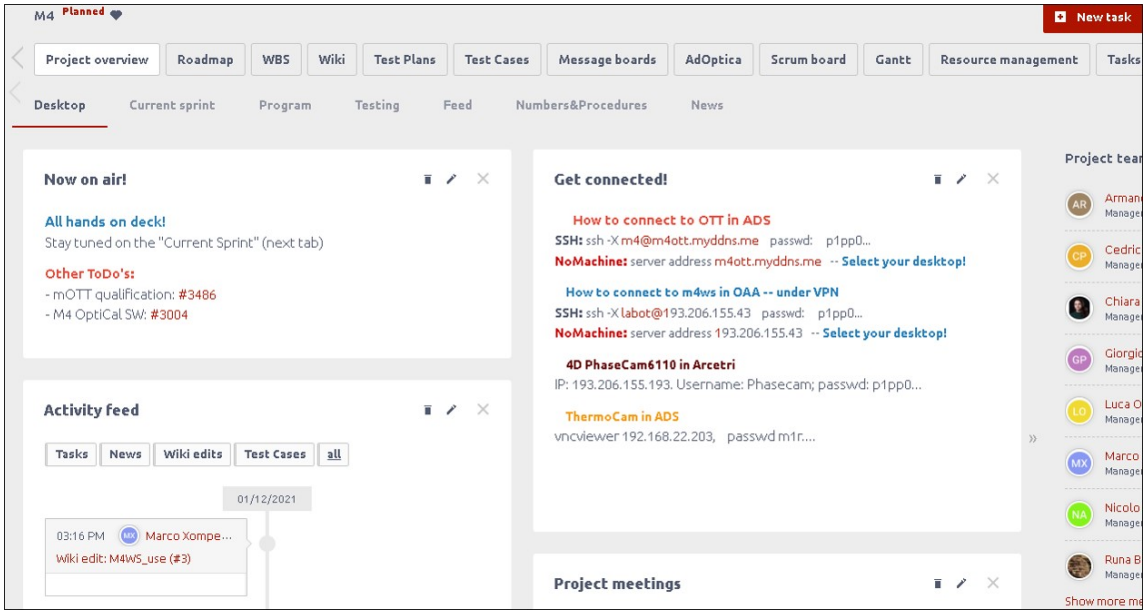

Figure 1. The M4 Redmine homepage.

Activity feed The panel is a quick-view of who has done what and allows monitoring the project activity flow.

Current print It is basically a verbose summary of the past sprints. It allows the passer-by to read the status of the activities.

Numbers and procedures It is a panel with static, ready to use informations such as startup/acquisition/processing procedures, home positions of devices, conversion and calibration factors.

Requirement status A table of the project requirements to summarize their current compliance status, including the last measured value and who performed the measurement.

4.2 WBS and task

The WBS page is a tree-diagram with all the tasks arranged according their mutual relationship. Tasks can be viewed on a milestone, assignee, status or tracker basis, thus allowing an organized view of the project. In principle, the WBS is created together with the project and during such initial phase it contains the top-level work packages only: as long as the scope and targets of the projects are identified, the WBS is hierarchically expanded to include smaller and smaller actions or tasks. For the specific case of the M4 project, where the RM platform has been implemented post-facto, the original WBS contained a few global tasks linked to milestones; then it has been populated with all the activities to be performed.
What is the content of each task? We have foreseen 2 task types (or tracker in the RM notation), Actions and Requirements. Action tasks contain the summary of the activity, ready to be exported as an internal report; Requirement tasks contain the requirement values and the current verification status.

4.3 Wiki

This is a very traditional wiki platform. Within the scope of the M4 project, we used the Wiki to store daily notes, laboratory logs, miscellaneous informations. The point here is that we considered the task in the WBS as the official container for report and deliverable informations, while the Wiki is the place for the rest of the stuff. Such approach implies that a daily, or periodic, effort shall be put by everybody to sum-up the relevant results, lessons-learned and informations in general from the Wiki to the specific Task page. In this way, an updated status of the activities is always available entering the WBS.

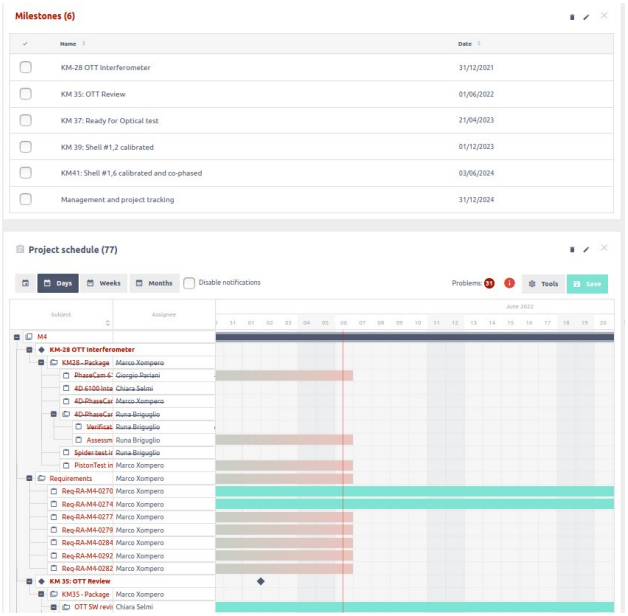

Figure 2. The program tab, with milestones and Gantt chart (that we did not really used....)

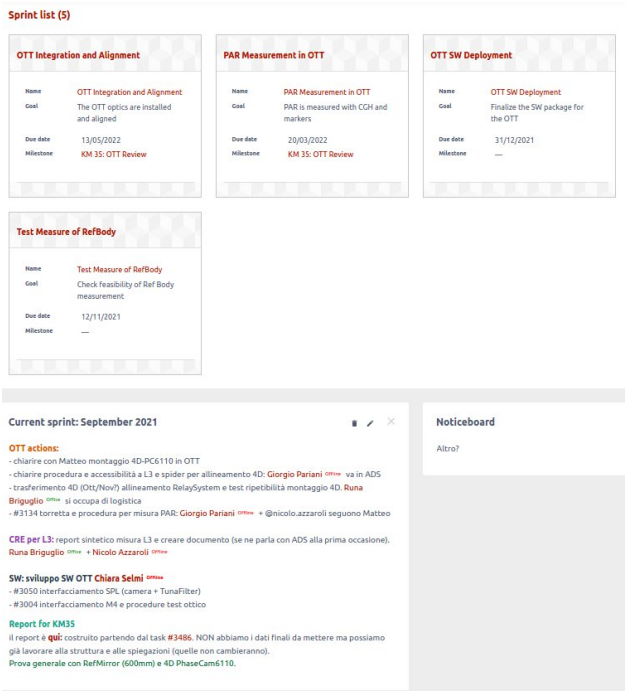

Figure 3. The verbose Sprint panel, with a detailed, historical description of the sprints (including the one related to the SPIE2022 papers.)

4.4 Test management

Test management is what we found more useful for our laboratory and measurement campaigns. The system is based on three levels: test plans, test cases, test runs, described below.

Test plans A test plan is the description of a specific measurement, regardless the test conditions, or the test equipment. The features of a test plan are the test procedure (the sequence of action to produce the measurement up to the result) and the data-to-deliver. This last point is what differentiates test plans: for instance *full aperture WFE* or *local curvature error*. Test plans act then as containers for miscellaneous test conditions to deliver a well defined datum.

Test cases Each possible flavour a test plan may be executed is a test case. We identified scenarios to be selected as environment conditions (day-night, season, noisy-quite, devices on-off), system-under-test (e.g. OTT or dummy-OTT, interferometer) and also sampling and processing software, to take into account the variability obtained with different SW versions or parameters.

Test runs Test runs are the atomic realizations of a measurement in given conditions and are useful to take into account the intrinsic variability of that sampling although in the same conditions. We customized such tabs with adding fields such as the data-identifier (a tracking number indicating the saved folder), the link to the wiki page where the sampling is described and of course the measurement result.

The day-after evaluation of such logging structure is, despite its complexity, pretty positive. It allows in facts to monitor globally both the intrinsic and the intentional variability of measurement. To this end, although the *test result* is naturally an outcome of a test run, we included a result field also for test cases and plans. The duty of filling in the field is demanded to the user that shall summarize the results from runs to cases and to plans. In this regard, for the test plans it is more a test report than a mere copy-paste of the numerical value, also to allow discussing the results across the variability of the test case conditions.

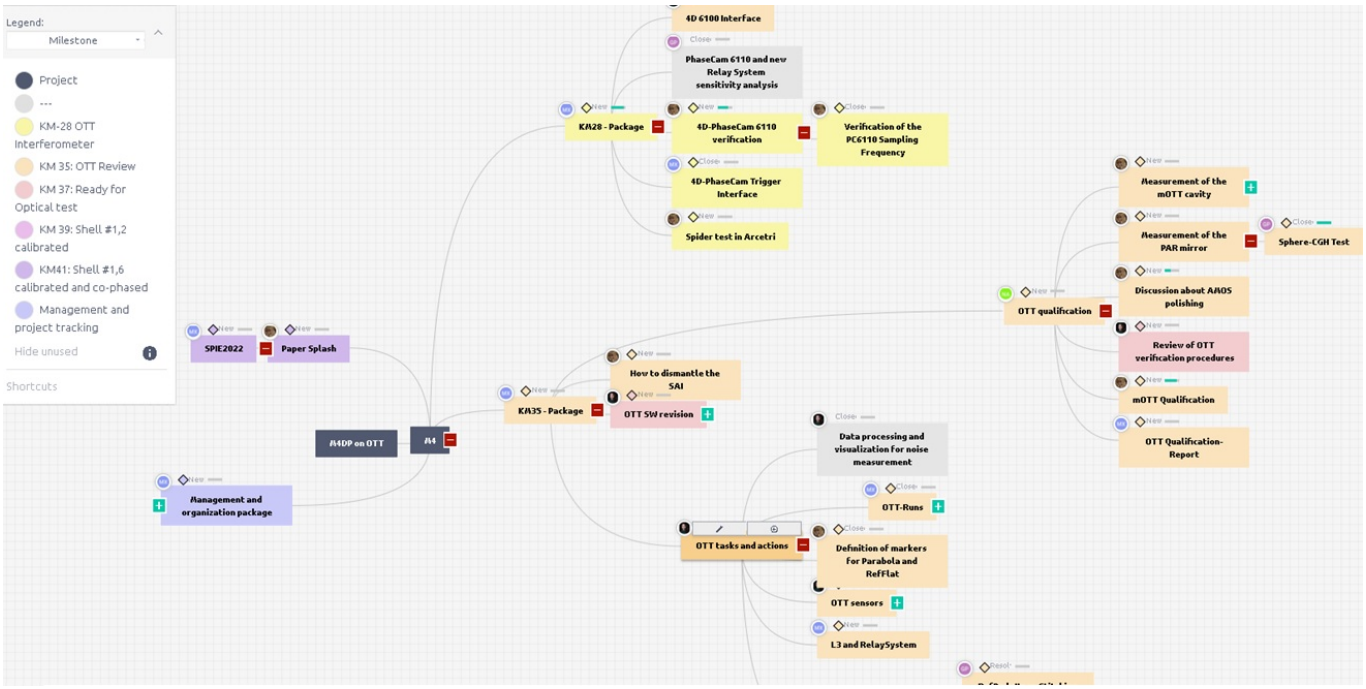

Figure 4. The Work Breakdown Structure tree-diagram of the M4 project. The color-code highlights the tasks associated with each milestone.

4.5 Scrum board

Given the peculiarities of the M4 project (intense bursts of activities, then long pauses), we formalized such approach adopting an agile method. In this case, sprints don't happen on a weekly basis, rather are set-up when a block of activities is in view: optics acceptance, drop of documents, actions on the test tower.
The scrum board allows then to follow the status of the activities during their duration. To this end we coded our own statuses to trap, for instance, a task requiring some feedback from the supervisor (status: NeedFeed in the chart) or a concluded task that shall be reported to the team to be offically closed (status: WrapUp in the chart).

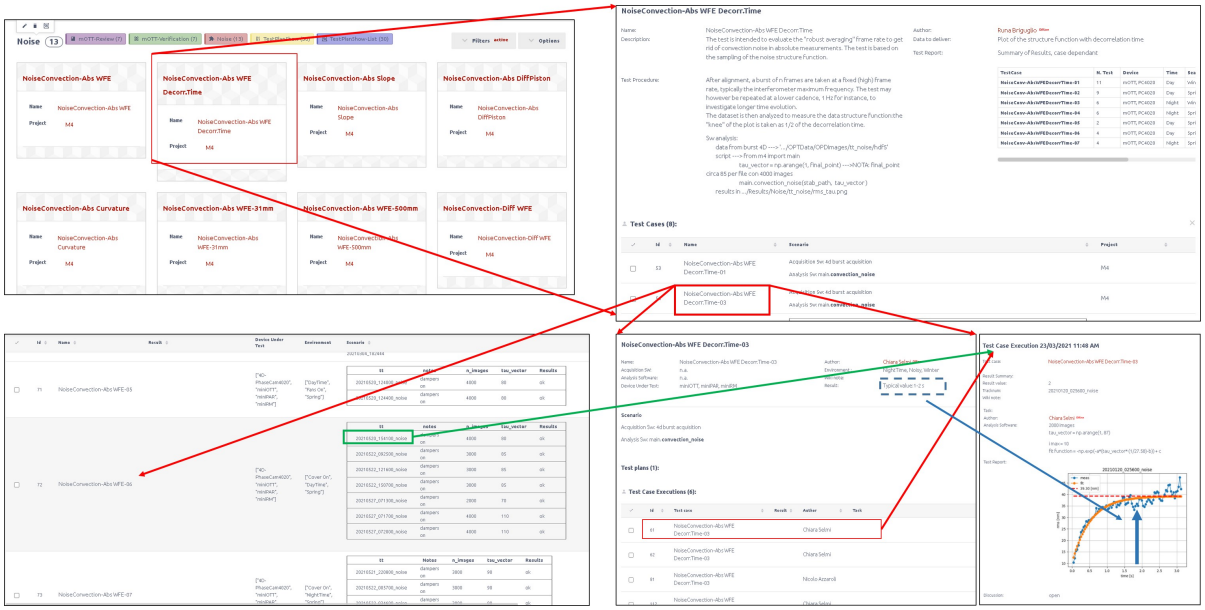

Figure 5. Schematic overview of the test management architecture. Top-left: test plans; top-right: a single test plan with procedure and summary report and associated test cases, differentiated by test conditions or device under test; bottom-left: test cases list for a given plan, and associated atomic test realizations; bottom-right: a single test realization with the specific result.

Figure 6. A *sprint* board, where the tasks associated with a given sprint are organized according their status.

5. THINKING BIG: A ” FOUR SEASONS TEMPLATE”

After the successful experience of the RM for the M4 project, we focused on the realization of templates to be adopted for projects in INAF. A *template* is basically an empty project with a structured organization of pages modules, graphs and so on. A few activities (tasks) are added to implement the visualization and tracking of things. The goal is to prepare a tool to be distributed to the INAF community and help in carrying on the projects.

5.1 Use cases

We focused on three main use cases to intercept the most common work needs within the community: *Project life cycle, Technical services, Administration.* From the perspective of a research institution, the first is the one

that most matches with the typical activities; we will describe it in details in the following sections. An efficient technical-administrative structure is however fundamental to complete successfully a project, that is the reason why we prepared those templates also.
The administrative one is basically built on *workflows* based on documents. Such approach is based on formalizing-standardizing typical activities such as management of orders, contracts, procurement documentation, official communications. The main effort has been drawing the information flow and the identification of the key-documents for each phase. The workflow is then built upon transitions amongst specific roles in the project (or department/office).
The template for the technical services is intended to maintain facilities and carry out scheduled operations. The project doesn't really need those project management tools such as Gantt charts, resources management or milestones; instead, we based it on 3 custom elements: checklists, bugs and products. Checklists are intended to formalize routinary operations such as scheduled maintenance and repetitive tasks. The Bugs section is used as help-desk, issue-tracking: the users may flag malfunctionings, warnings or other situations requiring a technical intervention. The Product section has two tasks: to serve as an inventory for instruments and facilities and to manage the department supply chain (e.g. consumables). As an inventory, such section allows tracking and keeping up-to-dated the status and locations of instruments and laboratories, to store conveniently user documentation, to share informations about usage, training, known bugs, drivers and interfaces.

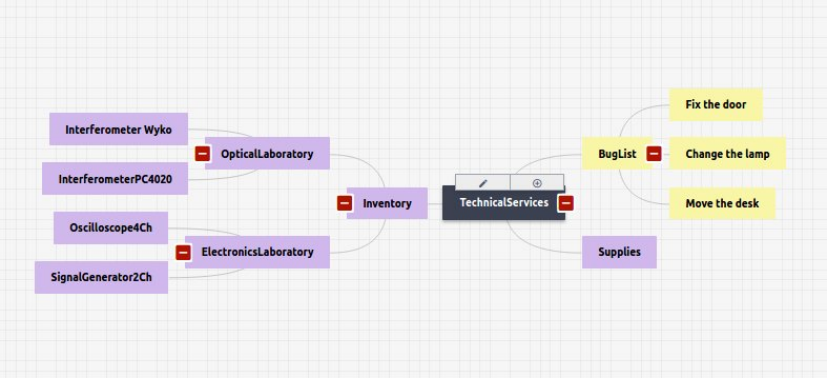

Figure 7. The WBS of the template for technical services. Yellow tasks are for *bugs*, purple tasks are for hardware, supplies.

5.2 Life cycle: a common scheme

A project is formally divided into 3 conceptual steps:

NEED Acquisition of the stakeholder's needs to identify use cases and performance requirements: Phases A-B

VISION Allocation of performance requirements to systems, subsystems and functionalities. Flow down of system requirements. System design: Phase C

DELIVERY Procurement, Manufacturing, Integration, Verification, Validation, Commissioning and Support: Phases D-E

All of these phases share some common elements:

- actions, that someone should complete;
- requirements, in the form of desired performance or feature, or technical specification, or functional requirement;
- elements, from notional architecture to parts, subsystems and fully assembled systems;

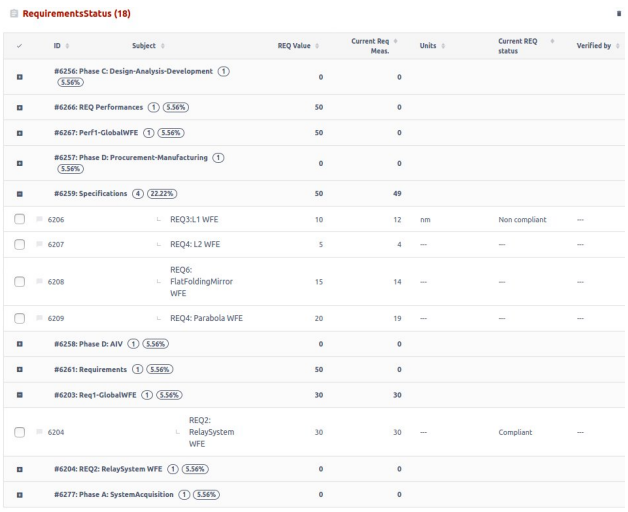

Figure 8. The Requirements overview for the Life-Cycle template.

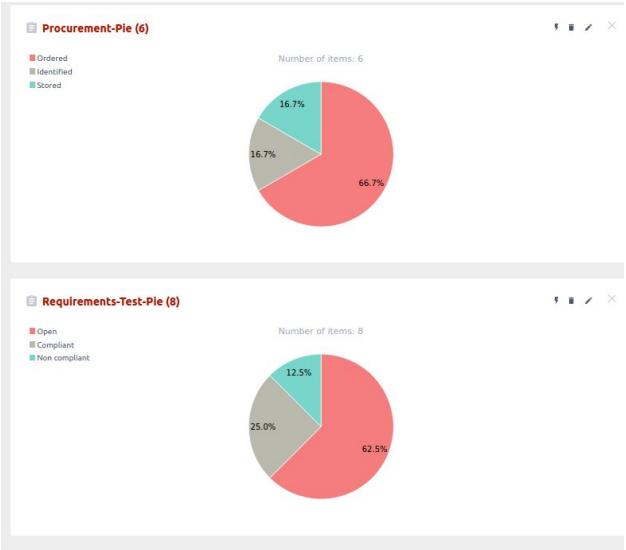

Figure 9. A status update for the Life-Cycle template: Procurement and Requirements.

- test, intending realizations from analytical/numerical models, or incoming inspection, or subsystem testing or system functional/performance verification.

Such a schematization is useful for two reason: on the one hand (for us) to identify a template structure; on the other hand to *force* the user to take into account all the elements since the beginning. For instance, linking requirements to tests is common during system integration and verification while it is often forgotten during a conceptual design. In this way, the template is asking the project team to focus early, even during a project Phase A, on the desired features or performances and associated functions or tests.
The template acts then also a as a tutorial offering the user a clear view of the project phases and needs, ready to be filled up before starting.

5.3 Life cycle: elements

Moving from the concepts described above we organized the *Life cycle* template according to the following elements.

Roadmap It is the temporal sequence of the project milestones, namely: White Book, End of Phase A, System Architecture Review (SAR), Preliminary Design Review (PDR), Critical Design Review (CDR), Final Design Review (FDR), Ready for Integration, Provisional Acceptance, End of commissioning. According to the specific phase of the project, the manager will take only the relevant milestones.

Products It is a custom tracker to indicate everything that is (or will be) a physical item, either a component or an assembled system. Products have custom database fields such as quantity, vendor, cost, associated requirements, links to documentation (such as quotations or datasheets).

Architecture&LifeStatus They are two custom fields for Products to indicate the architectural level of the item (is it a component or an assembled system?) and its current status as an element (in-design, ordered, in-test, stored, in-integration, etc). The values of status depends on the architecture. Such fields allow a more complex yet useful indexing of the architecture or product tree of the project.

Requirements It is a tracker to identify requirements or desired features. The custom fields we added are: REQ value, current measurement, verified by, verification method (Similarity, Design, Analysis, Inspection, Test).

Test Plans The testing suite works exactly as described for the M4 project. Here we differentiated by name the testing activity throughout the life cycle: Analysis during Design phase, Inspection-Verification during procurement/manufacturing phase, REQVerification during system verification and validation.

The agile approach is implemented so that Action or Product tasks may be moved across specific transitions to visualize the work/procurement/testing progress, the failure state, the request for feedback, the closure of actions or testing.

5.4 How to use the templates

Now the templates are ready to be offered to the community. We prepared a checklist to guide new project managers through the RM.

1. As first, do a preliminary sketch of you project on the paper and answer the following questions: what should I deliver? what is the system architecture? Which are the relevant requirements?
2. Check what is your project phase within the life cycle (Phase A? MAIT?) and remove all the non-pertinent milestones, tasks, tests.
3. Build the WBS (all the main action tasks), the PBS (product breakdown), the list of requirements, the Test Plans.
4. Starting from top-level items, create all the internal links: Products to Requirements, REQ to Tests, Actions to Products, everything to milestones.

In the end, the user may customize the Desktop with graphs and charts starting from examples.

6. CONCLUSIONS

Finding the most suitable management tool is a work in the work. In this paper, we summarized our experience with a commercial Redmine platform to organize and manage the M4 project. We customized the database to meet our needs, in particular the organized, incremental logging of activities, the tracking of the laboratory tests to the associated requirements, an agile team management versus a discontinuous workflow. We also tried the implementation of the seven pillars of system engineering. As a result we obtained a tool that helped significantly our work, while allowing a constant engagement and update of the team.
Starting from such experience, we created customized project templates to be offered to our community at INAF. The goal is to provide an easy-to-use tool for the widest (internal) public, from project supervisors to young post docs. The next step will be to monitor the response from the community, to improve and wrap up such experience.

REFERENCES

[1] Biasi, R., Andrighettoni, M., Angerer, G., & al.: ELT M4 adaptive unit ready for integration, Proc. of SPIE, **11445** (2020)

[2] Briguglio, R., Pariani, G., Xompero, M., &al.: The crystal ball, the spider and other stories: a journey around the test tower of the M4 adaptive mirror, Proc. of SPIE, **10703** (2018)

[3] Biasi, R., Manetti, M., Andrighettoni, M., &al.: E-ELT M4 adaptive unit final design and construction: a progress report, Proc. of SPIE, **9909** (2016)

[4] B.E. Goldberg, K. Everhart, R. Stevens, N. Babbitt III, P. Clemens, and L. Stout, System Engineering Toolbox for Design-Oriented Engineers, NASA Reference Publication 1358, (1994)

[5] David D. Walden, Garry J. Roedler, Kevin J. Forsberg, &al.: Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, INCOSE, (2015)

[6] Jacinto Javier Vaz-Cedillo: PORÍS: practical-oriented representation for instrument systems, Proc. of SPIE, **7738** (2010)

[7] Jacinto Javier Vaz-Cedillo: The open-source path to model based enterprise: proof of concept , Proc. of SPIE, **12187**, (2022)

[8] Jacinto Javier Vaz-Cedillo: cosmoSys-Req: a free open-source requirements management tool, Proc. of SPIE, **12187**, (2022)