RobMoSys

H2020—ICT—732410

RobMoSys

Composable Models and Software for Robotics Systems

Deliverable [D6.9]: [Eclipse Project and Eclipse Project Proposal]

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No. 732410

Project acronym: RobMoSys

Project full title: Composable Models and Software for Robotics Systems

Work Package: [WP 6 - Exploitation]

Document number: [D6.9]

Document title: [Eclipse Project and Eclipse Project Proposal]

Version: [1.0]

*Due date: [*December 31th, 2020]

Delivery date: *2020-12-31*

Nature: [Report (R)]

Dissemination level: Public (PU)

Editor: Marco Jahn (EFE)

Author(s): Marco Jahn (EFE), Maria Teresa Delgado (EFE), Dennis Stampfer (HSU), Matteo Morelli (CEA), Ansgar Radermacher (CEA)

Reviewer: Lorna McKinlay (PAL)

Executive Summary

It is a challenge for any research project to implement a sustainable and successful Open Source strategy for their project results. More specifically it is very difficult to (1) reach the project targets, (2) get the desired visibility and (3) build a community around the project, which will contribute to the sustainability of the code. These elements present a challenge for organisations that are smaller or less well known.

As of May 2019, GitHub reported having over 37 million users and more than 100 million repositories (including at least 28 million public repositories), making it the largest host of source code in the world. While GitHub provides the necessary infrastructure to host open source projects, the chances of creating a vibrant developer community can be rather low for research projects due to their limited life-span. Successful open source requires more than technical infrastructure, especially when aiming for commercial up-take.

Therefore, the RobMoSys consortium decided to work closely with the Eclipse Community to benefit from their developer's network and from their experience in building sustainable communities. The Eclipse Foundation provides vendor-neutral governance, IP management, community building, and infrastructure for open source projects. The clearly defined Eclipse Development Process provides the basis for successful, industry-grade open source software. The Eclipse Public License and thorough IP management and governance pave the way for commercial exploitation.

Two Eclipse Projects have been created during the RobMoSys Project: Eclipse Smart MDSD and Eclipse Papyrus for Robotics.

Eclipse SmartMDSD[1] is an Eclipse-based Integrated Development Environment (IDE) for robotics software development. The SmartMDSD Toolchain provides support and guidance to apply best-practices for the development of individual software building blocks, as well as their composition to robotics applications and systems. Eclipse SmartMDSD went through the Eclipse Development Process, was accepted as an Eclipse Project in December 2019 and had its first full release in July 2020. As of today, the project has four active committers and has a solid foundation for becoming a successful open source project. The project is in the Incubation Phase. Incubation indicates that the Eclipse community is helping this project "learn the ropes" about being a full open source project producing high quality extensible frameworks and exemplary tools. Projects typically stay in the incubation phase for a year or two before graduating to the Mature Phase.

Papyrus for Robotics[2] is a graphical editing tool for robotic applications that complies with the RobMoSys approach. It manages the complexity of robotics development by supporting composition-oriented engineering of robotics systems and separating the task into multiple tiers executed by different roles. It is based on Eclipse Papyrus, an industrial-grade open source Model-Based Engineering tool. Eclipse Papyrus has notably been used successfully in industrial projects and is the base platform for several industrial modeling tools.

[1] https://projects.eclipse.org/projects/modeling.smartmdsd
[2] https://www.eclipse.org/papyrus/components/robotics/

Content

1 Introduction

This deliverable provides an overview of the RobMoSys open source activities and particularly the two major outcomes: Eclipse SmartMDSD and Eclipse Papyrus for Robotics. These projects are governed under the Eclipse Foundation umbrella, providing the ecosystem for long-term maintainability, community-building and exploitation.

Section 2 provides an overview of open source and the Eclipse Development Process, which plays an important role in creating successful and business-friendly open source software. Section 2.1 presents a brief summary of what is open source and why an open source community is important to a project like RobMoSys. We explain what an open source community's best practices are and why they contribute to the sustainability of the project. Section 2.2 presents an overview of the Eclipse Foundation, its developer community and its working groups and Section 2.3 is focused on the Eclipse Development Process, with particular focus on the Project Lifecycle, which are the distinct Phases projects go through before becoming an Eclipse Project.

Finally, in Section 3, we present Eclipse SmartMDSD and Eclipse Papyrus for Robotics.

2 The Eclipse Foundation and the Eclipse Development Process

2.1 What is Open Source?

2.1.1 Elinor Ostrom principles

Elinor Ostrom, Nobel price of Economy in 2009, designed 8 principles for managing stable Common Pool Resource (CPR):

1. **Clearly defined** (clear definition of the contents of the common pool resource and effective exclusion of external un-entitled parties);
2. The appropriation and provision of common resources that are **adapted to local conditions**;
3. **Collective-choice** arrangements that allow most resource appropriators to participate in the decision-making process;
4. **Effective monitoring** by monitors who are part of or accountable to the appropriators;
5. A scale of **graduated sanctions** for resource appropriators who violate community rules;
6. Mechanisms of **conflict resolution** that are cheap and of easy access;
7. **Self-determination** of the community recognized by higher-level authorities; and
8. In the case of larger common-pool resources, organization in the form of **multiple layers of nested enterprises**, with small local CPRs at the base level.

Deciding to contribute to the open source either as a consumer of open source components or producer of open source components or both is a volunteer act to manage **in common** some software code and all the resources attached to this software.

Because the initial investment might feel harder, it is important to feel guided by Elinor Ostrom principles.

2.1.2 What is Open Source Software?

The Open Source Initiative (https://opensource.org) provides a very good definition of Open Source Software (OSS) and defines it in *10 commandments* (https://opensource.org/osd-annotated):

1. **Free redistribution**: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
2. **Include source code**: The program must include source code, and must allow distribution in source code as well as compiled form.

3. **Modifications and derived works**: The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. **Integrity of author's source code**: The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
5. **No discrimination against person and groups**: The license must not discriminate against any person or group of persons.
6. **No discrimination against fields of endeavor**: The license must not restrict anyone from making use of the program in a specific field of endeavor.
7. **Distribution of license**: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
8. **License not specific to a product**: The rights attached to the program must not depend on the program's being part of a particular software distribution.
9. **License not restricting other software**: The license must not place restrictions on other software that is distributed along with the licensed software.
10. **License technology neutral**: No provision of the license may be predicated on any individual technology or style of interface.

2.1.3 Why do we need Open Source?

From Small and Medium-sized Enterprises[3] to large organizations[4], a lot of companies have adopted and contribute to the one or more open source communities like the Apache Software Foundation, the Eclipse Foundation, and the Linux Foundation. Some of them have been involved for several decades already.

Actually, organizations like open source software to proprietary software for many reasons, including:

- **Maturity of the model:** There are numerous examples of projects and products based on the OSS which are more reliable and sustainable than other proprietary solutions. Here are a few of them:
    - The Linux operating system, which is now widely adopted by major private or public organizations.
    - Apache HTTP Server, certainly the most used HTTP server. It played a key role in the initial growth of the World Wide Web.
    - The Mozilla web browser called Firefox, which in 2009 was the most popular web browser with 32% of the market. In 2016, between 9% and 16% of individuals use Firefox as their desktop browser, making it the second most popular web browser, the first one is Google Chrome.
    - In 2001, IBM gave up its Java IDE to the open source community, now known as the Eclipse IDE. After 15 years, this platform supports one of the largest development tool portfolios. Not only Rational, the IBM brand, has built all its desktop workbenches on top of Eclipse, but several SME have based their own development tool on top of the Eclipse Rich Client Platform, the minimal set of Eclipse plug-ins needed to build a rich client application.

[3] Like CodeTrails (http://www.codetrails.com/) in Code Assistance, Micro-EJ (http://www.microej.com/) in IoT and Obéo (https://www.obeo.fr/en/) in Modelling.

[4] Like Amazon, Bosch, Google, IBM, Microsoft, Samsung, and Siemens.

- **Cost of acquisition:** Adopters of OSS obtain a financial gain for each stage of a project. For example:
  - **Free**: it's free to download and use.
  - **Try before buy**: because it is free, companies can try different OSS solutions before making the decision to invest time or resources in a specific one.
  - **Hiring is easier**: because OSS is free, many developers use it and become proficient with the software early on in their career or during their studies. This makes it easier and less expensive to find good developers that have experience with the open source technologies they have adopted for their project.
  - **Training:** it's easier to train a team with the assets produced by the OSS community of developers.
  - **Customizability**: open source software can be tweaked to suit various needs. Since the code is open, it's simply a matter of modifying it to add the functionality needed by the project.
  - **Time to Market is shorter**: products don't have to be built from scratch. Companies can rely on sustainable OSS and build their solution on top of it.
  - **Lower total cost of ownership:** companies can rely on the OSS community for maintenance and, by joining the community, they mutualized maintenance costs.
- **Dependence**: Organizations don't depend on the status of the subcontractor who originally built the software. In open source software, if a contributor ceases working on a project for any particular reason, the source code stays accessible and someone else can take over the work.
- **Quality of the code:** OSS gets closer to what users want because those users can have a hand in improving it.
- **Security**: Some users consider OSS more secure and stable than proprietary software, mainly because they can control the source code and they can identify and fix errors or omissions. The efforts are mutualized with the other community members, which results in secure and stable source code.

The RobMoSys consortium does not only reuse OSS, but is also open sourcing two projects developed in RobMoSys to make them sustainable, even beyond the end of the research project. By experience, when code is not open sourced, it can be challenging to retain interest from all consortium members.

The Eclipse Foundation has presented to the consortium the kind of business models we can put in place with the open source. The RobMoSys consortium should be able to take advantage of the open source on 3 axes:

- Selling services by leveraging expertise,
- Selling hardware by providing open source software, or
- Selling infrastructures.

2.1.4 Why do we need Open Source communities?

An OSS community is the keystone for the sustainability of our project. It is essential to be able to attract and convince people that our code is worth time and resources for testing it, providing feedback, providing patches, and contributing in general, otherwise the intrinsic value of OSS could be lost.

In other words, without Maturity, Quality, Cost of Acquisition, Control, and Security, the Sustainability of our code could present a challenge. And vise-versa, Sustainability is a great indicator demonstrating the Maturity, the Quality, the Control and the Security of the code.

So, an OSS community has four main roles:

- **Sustainability**: This ensures the sustainability of this code. If the initial committers leave the project, the code will continue to be maintained by the community.
- **Mutualization**: By building a community around your project, we are able to mutualize our effort and resources. The community, including few of our resources, will maintain the core of the project while the rest of our team will work on our product, our added value, to make the difference with our competitors.
- **Standardization**: If our code is widely adopted, it will become, de facto, a standard. We have several examples in the industry:
    - In early 1991, Bosch open sourced the specification of the CAN Bus (Controller Area Network). This generated a strong attraction in the car manufacturer community, designating the CAN bus as THE standard for the communication between microcontrollers in a vehicle.
    - In 2011, after 15 year of internal usage, IBM open sourced its Messaging Client for Embedded Devices: MQTT. In less than 5 years, this protocol was so widely adopted that in January 2016 it became an ISO/IEC Standard.

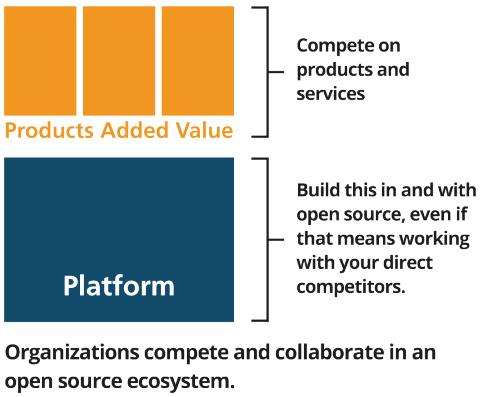

*Figure SEQ Figure * ARABIC 1. A business friendly ecosystem based*

- **Metrics**: Last, but not least, a good way to know if the OSS we want to use is sustainable and viable is to check the activity of the community: number of committers, number of commits, regularity of the release, and the quality and quantity of assets built around the OSS is also a great indicator. In other words, the community is an excellent evaluation metric for our project.

This is the 'snowball' effect: we attract early adopters with the quality of our code, the initial assets attached to the code like the Getting Started guide, documentation, scientific and technical papers, first releases, well defined code infrastructure (bug tracking system, continuous testing, continuous integration…), and the interest of the adopters will do the rest. For this reason, it is very important to take a particular care of our early RobMoSys adopters and the project's community as it grows.

2.2 The Eclipse Foundation

The Eclipse community is for individuals and organizations who wish to collaborate on commercially-friendly open source software. Its projects are focused on building an open development platform consisting of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of complementary products and services.

The Eclipse Project was originally created by IBM, in November 2001, and supported by a consortium of software vendors. Industry leaders Borland, IBM, MERANT, QNX Software Systems, Red Hat, SuSE, TogetherSoft, and Webgain formed the initial eclipse.org Board of Stewards. By the end of 2003, this initial consortium had grown to over 80 members. Today there are over 270 Eclipse members.

Originally, a consortium that formed when IBM released the Eclipse Platform into open source, the Eclipse Project became an independent body that will drive the platform's evolution to benefit the providers of software development offerings and end-users. All technology and source code provided to and developed by this fast-growing community is made available royalty-free via the Eclipse Public License (EPL).

The Eclipse Foundation, Inc. was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community. It was created to allow a vendor neutral and open, transparent community to be established around the Eclipse community. Today, its community consists of individuals and organizations from a cross section of the software industry.

2.2.1 The Eclipse Developer Community

Today, the Eclipse community consists of:

- Over 370 open source projects;
- 130 million lines of code delivered every year;
- Over 1.600 committers;
- Over 5 millions of active users;
- An average of 1,5 million downloads per month;
- An average of 2 million unique website visitors per month;
- A leading IDE on Java, C/C++, PHP…;
- 270 members (including 11 strategic members);
- 75 events organized or co-organized each year, like EclipseCon events, Eclipse Days, Eclipse Summits or Eclipse Demo Camps, etc.

Joining this community brings the necessary visibility to assist the RobMoSys consortium for the sustainability of its project and will help with the best practices of open source.

2.2.2 The Eclipse Industry Working Groups

Companies looking to drive innovation and efficiencies within their own organizations are increasingly looking for external sources to advance new ideas. Commonly referred to as "open innovation," this paradigm encourages collaboration across organizational boundaries, and is often practiced in the open source community. In 2009, the Eclipse Foundation launched the notion of Working Groups to allow organizations to combine the best practices of open source development with a set of services required for open innovation, and in turn enabling organizations to foster industry collaborations.

Eclipse Working Groups (https://www.eclipse.org/org/workinggroups/) provide a **vendor-neutral** governance structure that allow organizations to freely collaborate on new technology development. The Eclipse Foundation, through Eclipse Working Groups, provides five basic services to enable these types of collaborations:

Figure 2. Pillars of open collaborations

- **Governance**: Good governance that controls how decisions are made, policies established and disputes resolved is important for any successful collaboration.
- **Intellectual Property Management and Licensing**: Collaborations among different organizations requires due diligence on the co-developed intellectual property. Eclipse Working Groups are established under the Intellectual Propriety (IP) policies of the Eclipse Foundation. These policies ensure that any open source software created in Eclipse projects is available for use by anyone, including developers of commercial software products.
- **Development Processes**: The Eclipse community has created a successful development process for large-scale distributed development that involves many different organizations.
- **IT Infrastructure**: The Eclipse Foundation manages the IT infrastructure for Eclipse Working Groups, including Git code repositories, Bugzilla databases, Hudson CI servers, development-oriented mailing lists and newsgroups, download sites, and websites.
- **Ecosystem Development**: An important way that the Eclipse Foundation supports the community is through active marketing and promotion of Eclipse Working Groups and the wider Eclipse ecosystem.

Any collaboration needs these services. Eclipse Working Groups make it easy to reuse the services provided by the Eclipse Foundation rather than creating them from scratch.

Today, the Eclipse Foundation has established 14 working groups. In the following we provide an overview of the working groups which are relevant for RobMoSys:

- **Internet of Things:** With over 40 industry-leading member companies, more than 35 IoT projects, and 350+ contributors, the Eclipse IoT Working Group (https://iot.eclipse.org/) is the preeminent open source community for commercial-grade IoT innovation. This initiative enables collaboration on the development of open source implementations of IoT standards and protocols, frameworks and services that will be used by IoT solutions, and tools for IoT developers. IoT and Robotics are overlapping in certain technical areas as robots can benefit from interacting with IoT environments.
- **OpenADx:** The OpenADx Working Group (https://openadx.eclipse.org/) is centered around the autonomous driving toolchain and aims to bring transparency and better integration capabilities into the autonomous driving tool space. This industry-wide initiative is of special interest for OEMs, Tier 1s, and tool and technology vendors. openADx was launched by Bosch and Microsoft and now comprises contributors from the industrial, research, and start-up worlds Eclipse Cyclone DDS, a tier one ROS 2 middleware has found its way into the openADx WG.
- **EdgeNative:** The Eclipse Edge Native Working Group (https://edgenative.eclipse.org/) drives the adoption of Edge Computing technologies. It provides services like vendor-neutral marketing to the Eclipse Edge Native ecosystem and defines licensing and intellectual property flows that encourage the community to open collaboration. Edge processing of information is highly relevant for robotics applications.
- **Cloud Development Tools:** The Eclipse Cloud Development Tools Working Group (https://ecdtools.eclipse.org/) The Eclipse Cloud Development (ECD) Tools Working Group will drive the evolution and broad adoption of de facto standards for cloud development tools, including language support, extensions, and developer workspace definition. Cloud Tools are an emerging topic for robotics development, aka Cloud Robotics.

2.2.3 The Eclipse Intelligent Robotics Working Group

RobMoSys partners are working on establishing an Intelligent Robotics Working Group. The goal of this working group is to harmonize the development of an industry platform for robotics based on open technologies. The starting point are the two RobMoSys projects Eclipse SmartMDSD and Eclipse Papyrus for Robotics. Another candidate is Eclipse Cyclone DDS, a tier one ROS2 middleware.

At the time of writing, the process of creating the working is in the state of definition of a charter and identifying interested parties as core founding members. Typically, this process takes time because Eclipse Working Groups are not loose collaborations without binding. Committing to a working group means committing time, effort, and budget to implement, maintain, and marketize open source technologies in cross-organizational collaborations.

2.3 Eclipse Development Process

The Eclipse community has created a successful development process for large-scale distributed development that involves many different organizations.

In this section, we will summarize some of the principles driving the Eclipse Development Process (https://www.eclipse.org/projects/dev_process/) with focus on the different phases that a project needs to go through to become an Eclipse Project.

2.3.1 Definitions

2.3.1.1 Project Structure and Organization

A project is the main operational unit at the Eclipse Foundation. Specifically, all open source software development occurs within the context of a project. Projects have leaders, developers, code, builds, downloads, websites, and more. Projects are more than just the sum of their many parts, they are the means by which open source work is organized when presented to the communities of developers, adopters, and users. Projects provide structure that helps developers expose their hard work to a broad audience of consumers.

2.3.1.2 Committers

Each project has exactly one set of committers. Each project's set of committers is distinct from that of any other project. All project committers have equal rights and responsibilities within the project. Partitioning of responsibility within a project is managed using social convention.

The committers of a project have the exclusive right to elect new committers to their project; no other group can obligate the project to accept a new committer.

2.3.1.3 Code and Resources

Each project owns and maintains a collection of resources. These resources may include source code, a project website, space on the download servers, access to build resources, and other services provided by the Eclipse Foundation infrastructure. The exact infrastructure provided by the Eclipse Foundation varies over time and is defined outside this process document.

Namespaces are assigned to a project by the Eclipse Management Organization (EMO). All project source code must be organized in the assigned namespaces and projects can only release code under their own namespace (that is, they cannot infringe on another Eclipse project's namespace).

2.3.1.4 Leaders

There are two different types of Eclipse project leadership: The Project Management Committee (PMC) and project leaders. Both forms of leadership are required to:

- ensure that their project is operating effectively by guiding the overall direction and by removing obstacles, solving problems, and resolving any conflicts;
- operate using open source rules of engagement: meritocracy, transparency, and open participation; and
- ensure that the project conforms to the Eclipse Foundation IP policy and procedures.

2.3.1.5 Committers and Contributors

Each project has a development team, led by the project leaders. The development team is composed of committers and contributors. Contributors are individuals who contribute code, fixes, tests, documentation, or other work that is part of the project. Committers have write access to the project's resources (source code repository, bug tracking system, website, build server, downloads, etc.) and are expected to influence the project's development.

Contributors who have the trust of the project's committers can be promoted through the election to become committers for that project. The breadth of a committer's influence corresponds to the breadth of their contribution. A development team's contributors and committers may (and should) come from a diverse set of organizations. A committer gains voting rights allowing them to affect the future of the project. Becoming a committer is earned by contributing and showing discipline and good judgment. It is a responsibility that should be neither given nor taken lightly, nor is it a right based on employment by an Eclipse member company or any company employing existing committers.

The election process begins with an existing committer on the same project nominating the contributor. The project's committers will vote for a period of no less than one week of standard business days. If there are at least three positive votes and no negative votes within the voting period, the contributor is recommended to the project's PMC for commit privileges. If there are three or fewer committers on the project, a unanimous positive vote of all committers is substituted. If the PMC approves, and the contributor signs the appropriate committer legal agreements established by the Eclipse Management Organization (EMO) (wherein, at the very least, the developer agrees to abide by the Eclipse Intellectual Property Policy), the contributor becomes a committer and is given write access to the source code for that project.

At times, committers may become inactive for a variety of reasons. The decision-making process of the project relies on active committers who respond to discussions and vote in a constructive and timely manner. The project leads are responsible for ensuring the smooth operation of the project. A committer who is disruptive, does not participate actively, or has been inactive for an extended period may have his or her commit status revoked by the project leads. Unless otherwise specified, "an extended period" is defined as "no activity for more than six months".

2.3.1.6 Project Plans

Projects are required to make a project plan available to their community at the beginning of the development cycle for each major and minor release. The plan may be as simple as a short description and a list of issues, or more detailed and complex.

Project Plans must be delivered to the community through communication channels approved by the EMO. The exact nature of the project plan varies depending on numerous variables, including the size and expectations of the communities, and requirements specified by the PMC.

2.3.1.7 Mentors

New project proposals are required to have at least one mentor. Mentors must be members of the Architecture Council. The mentors must be listed in the proposal. Mentors are required to monitor and advise the new project during its incubation phase; they are released from that duty once the project graduates to the mature phase.

2.3.2 Development Process

2.3.2.1 Overview

Projects go through distinct phases. The transitions from phase to phase are open and transparent public reviews.

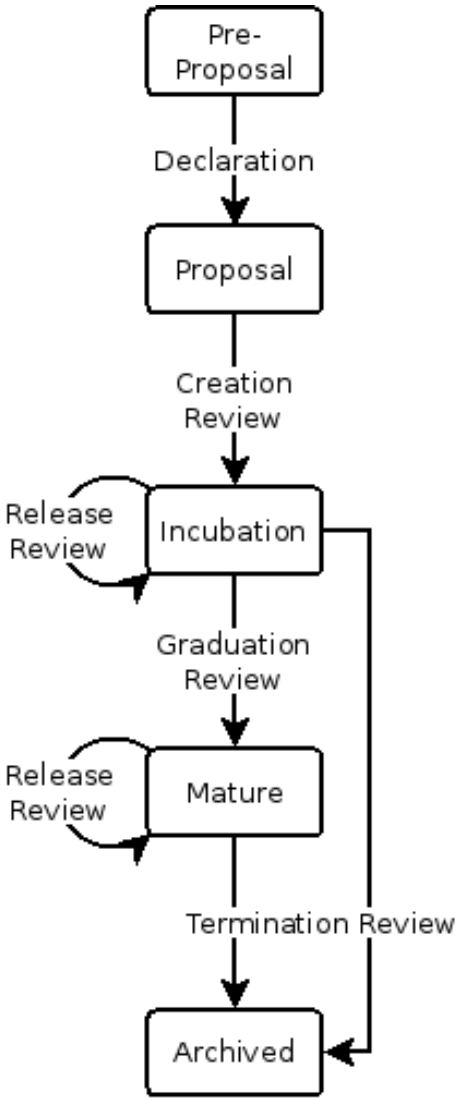

Figure . Eclipse Development Process

Pre-proposal Phase

An individual or group of individuals declares their interest in, and rationale for, establishing a project. The EMO will assist such groups in the preparation of a project proposal. The pre-proposal phase ends when the proposal is published by EMO and announced to the membership by the EMO.

Proposal Phase

The proposers, in conjunction with the destination PMC and the community, collaborate in public to enhance, refine, and clarify the proposal. Mentors for the project must be identified during this phase. The proposal phase ends with a creation review, or withdrawal. The proposal may be withdrawn by the proposers at any point before the start of a creation review. The EMO will withdraw a proposal that has been inactive for more than six months.

Incubation Phase

The purpose of the incubation phase is to establish a fully-functioning open-source project. In this context, incubation is about developing the process, the community, and the technology. Incubation is a phase rather than a place: new projects may be incubated under any existing project. A project in the incubation phase can (and should) make releases and the incubation phase ends with a graduation review or a termination review.

Mature Phase

The project team has demonstrated that they are an open-source project with an open and transparent process; an actively involved and growing community; and Eclipse-quality technology. The project is now a mature member of the Eclipse community. Major releases continue to go through release reviews.

Archived

Projects that become inactive, either through limited resources or by reaching their natural conclusion, are archived. Projects are moved to archived status through a termination review. If there is sufficient community interest in reactivating an archived project, the project can start again with a creation review. Since there must be good reasons to have terminated a project, the creation review provides a sufficiently high bar to prove that those reasons are no longer valid.

2.3.2.2 Eclipse Project Proposal

Any project which wants to join the Eclipse Foundation has to submit a proposal. The proposal has to follow the guidance described in the document[5].

To summarize, the proposal has to define:

- **Project Name**: Naming and branding are challenging issues. In order to provide a consistent brand for Eclipse, projects must follow the project guidelines[6]. As a defensive measure, the Eclipse Foundation holds the trademark to the Project names on behalf of the Projects - this prevents companies from misusing or misrepresenting their products as being the Projects. The EMO will initiate a trademark review prior to scheduling a Creation Review. Existing trademarks must be transferred to the Eclipse Foundation (please see the Trademark Transfer Agreement).
- **Project description**: the description must be clear, concise and understandable. It must use plain non-technical English. It must describe all acronyms. The project description should include the following sections:
    - Background: Describe where the project came from. What is the historical journey of the project; who/what company wrote the project. Did it go through any significant alterations / rewrites / language changes.
    - Scope: Provide an introductory paragraph describing what the project aims to be followed by several bullet points. The scope should allow for some flexibility, but still provide well-defined boundaries for the project.
    - Description: The introductory paragraph should clearly explain what the project is and does, similar to an expanded elevator pitch.
    - Why Here:
        - Why does this project want to be hosted at the Eclipse Foundation?
        - What do you expect to gain by having your project at the Eclipse Foundation?
        - What value does the project provide to the Eclipse community and ecosystem?
        - **Licenses**: Check the licenses that apply to the project.
        - **Legal Issues**: Describe any legal issues around the project and/or code.
        - List the current licenses of the main code.
        - List the 3rd party dependencies and associated licenses.
- **Initial Contribution**: Describe the initial contribution. Where is the code coming from? Current Eclipse project/GitHub repository or other.
- **Future Work**: How is the project going to grow its community (users / adopters / committers)?
- Source Code Section:
    - Repository Source Type
    - Source Repositories
- People:
    - A project needs a project lead

[5] https://wiki.eclipse.org/Development_Resources/HOWTO/Pre-Proposal_Phase
[6] https://wiki.eclipse.org/Development_Resources/HOWTO/Project_Naming_Policy

- Initial Committers
- Mentor(s)
- Interested Parties: Who is interested in this project? This could be individuals or companies.

2.3.2.3 Reviews

The Eclipse Development Process is predicated on open and transparent behavior. All major changes to Eclipse projects must be announced and reviewed by the membership-at-large. Major changes include the project phase transitions as well as the introduction or exclusion of significant new technology or capability. It is a clear requirement that members who are monitoring the appropriate media channels not be surprised by the post-facto actions of the projects.

Projects are responsible for initiating the appropriate reviews. If it is determined to be necessary, the project leadership chain (e.g. the PMC or EMO) may initiate a review on the project's behalf.

2.3.2.4 Creation Review

The purpose of the creation review is to assess the community and membership response to the proposal, to verify that appropriate resources are available for the project to achieve its plan, and to serve as a committer election for the project's initial committers. The Eclipse Foundation strives not to be a repository of unused code and thus projects must be sufficiently staffed for forward progress.

2.3.2.5 Graduation Review

The purpose of the graduation review is to mark a project's change from the incubation phase to the mature phase. The graduation review confirms that the project is/has:

- A working and demonstrable code base of sufficiently high quality.
- Active and sufficiently diverse communities appropriate to the size of the graduating code base: adopters, developers, and users.
- Operating fully in the open following the principles and purposes of the Eclipse Foundation.
- A credit to the Eclipse Foundation and is functioning well within the larger Eclipse community.

A graduation review is generally combined with a release review.

2.3.2.6 Release Review

The purposes of a release review are: to summarize the accomplishments of the release, to verify that the IP Policy has been followed and all approvals have been received, to highlight any remaining quality and/or architectural issues, and to verify that the project is continuing to operate according to the principles and purposes of the Eclipse Foundation.

2.3.2.7 Termination Review

The purpose of a termination review is to provide a final opportunity for the committers and/or Eclipse membership to discuss the proposed archiving of a project. The desired outcome is to find sufficient evidence of renewed interest and resources in keeping the project active.

2.3.2.8 Releases

Any project may make a release. Releases are, by definition, anything that is distributed outside of the committers of a project. If users are being directed to download a build, then that build has been released. All projects and committers must obey the Eclipse Foundation requirements on approving any release. All official releases must have a successful release review before being made available for download.

3 RobMoSys Open Source Projects

Two open source projects hosted at the Eclipse Foundation are a direct outcome of the RobMoSys Project: Eclipse SmartMDSD and Eclipse Papyrus for Robotics.

3.1 Eclipse SmartMDSD

This section gives an overview of Eclipse SmartMDSD. We suggest to visit https://projects.eclipse.org/projects/modeling.smartmdsd to learn more and get in touch with the developers and the community.

Eclipse SmartMDSD went through the Eclipse Development Process, was accepted as Eclipse Project in December 2019 and had its first full release in July 2020. As of today, the project has 4 active committers and has a solid foundation for becoming a successful open source project. The project is in Incubation Phase. Incubation indicates that the Eclipse community is helping this project "learn the ropes" about being a full open source project producing high quality extensible frameworks and exemplary tools. Projects typically stay in the incubation phase for a year or two before graduating to the Mature Phase.

3.1.1 Overview

This Eclipse project provides the "**SmartMDSD Toolchain**", an Eclipse-based Integrated Development Environment (IDE) for robotics software development. The SmartMDSD Toolchain provides support and guidance to apply best-practices for the development of individual software building blocks, as well as their composition to robotics applications and systems. This project will maintain the eclipse-based tooling with its internal implementation (e.g. meta-models, code-generators).

Underlying methodology

The SmartMDSD Toolchain supports various users in applying the necessary robotics structures to enable composition in an overall robotics ecosystem. More precisely, the SmartMDSD Toolchain enables **domain experts** to model shared robotics domain knowledge, **component suppliers** to develop and supply individual software components, and **system builders** to flexibly combine and re-combine (i.e. "compose") these components to new applications considering individual system-requirements.

Main target is the SmartSoft Framework

The SmartMDSD Toolchain will mainly support the "**SmartSoft** Framework", a service-oriented component-based robotics communication framework as one of the main underlying execution environments. However, it also outreaches to other initiatives such as the ROS framework and OPC UA.

The SmartMDSD Toolchain conforms to the structures proposed by the European Union's Horizon 2020 research and innovation programme "**RobMoSys**" (https://robmosys.eu/) and BMWi/PAiCE "**SeRoNet**" (https://www.seronet-projekt.de).

For more information, see the following resources:

- http://robmosys.eu/wiki/baseline:environment_tools:smartsoft:smartmdsd-toolchain:start
- https://www.youtube.com/watch?v=JIYPJXmop3U
- Dennis Stampfer, Alex Lotz, Matthias Lutz, and Christian Schlegel. "The SmartMDSD Toolchain: An Integrated MDSD Workflow and Integrated Development Environment (IDE) for Robotics Software". In: Journal of Software Engineering for Robotics (JOSER): Special Issue on Domain-Specific Languages and Models in Robotics (DSLRob) 7.1 (July 2016). ISSN 2035-3928, pp. 3–19. URL: http://joser.unibg.it/index.php/joser/article/view/91

3.1.2 Licenses

Eclipse Public License 2.0

3.1.3 Latest Release

V3.14, 2020-07-30

3.1.4 Source Repositories

https://github.com/eclipse/smartmdsd

3.2 Papyrus for Robotics

Papyrus for Robotics is a RobMoSys-conformant implementation of Papyrus customized for the Robotics domain. It provides a model-based and graphical development environment. It manages complexity of robotics developments by supporting composition-oriented engineering of robotics systems and separating the task into multiple tiers executed by different roles. Further information at: https://www.eclipse.org/papyrus/components/robotics/

Papyrus for Robotics builds upon an existing ecosystem around the Eclipse project Papyrus: www.eclipse.org/papyrus. Papyrus is part of the Eclipse Polarsys working group, targeting the development of embedded and critical systems by having a focus on safety, reliability and quality.

Papyrus for Robotics uses UML/SysML as underlying realization technology. The platform uses the UML profile mechanism to enable the implementation of DSMLs that assist RobMoSys' ecosystem users in designing robotics systems.

Papyrus for Robotics features a modeling front-end that provides custom architecture viewpoints, diagram notations and property views, including (but not limited to) those for the definition of software components, services, the representation of the system's architecture, and coordination and configuration policies. The main target functionalities of Papyrus for Robotics are:

- **Code-Generation** Papyrus for Robotics includes generators that transform models of software component architectures, platform descriptions and deployment specifications into code. Currently, it supports code generation for ROS2 run-time environments. The tool speeds-up the development of ROS2 systems through the generation of nodes' source code, including their communication interface (topics, services and actions), the generation of build files and launch scripts. See https://wiki.eclipse.org/Papyrus/customizations/robotics/ros2
- **Reverse-Engineering** Papyrus for Robotics can build component and service definitions models from existing systems. A complete library of existing ROS2 service definitions is bundled with the current release. The reverse engineering of components is based either on running nodes or on source code analysis. More information on this functionality is available at https://wiki.eclipse.org/Papyrus/customizations/robotics/reverse
- **Behavior tree execution** Papyrus for Robotics supports graphical modeling of behavior trees. Modeled behaviors are executable in frameworks compatible with BehaviorTree.CPP, such as ROS2 Navigation2. See https://wiki.eclipse.org/Papyrus/customizations/robotics/bt
- **Safety Analysis**. Papyrus for Robotics features a safety module to perform dysfunctional analysis on system architecture's components, including task-based hazard and risk analysis (HARA), see https://wiki.eclipse.org/Papyrus/customizations/robotics/hara

During the last months of the project until month M30, CEA has refined and consolidated Papyrus for Robotics, has supported the tool adoption for different use cases, such as the RobMoSys CEA pilot and other use cases in projects where CEA is involved, and has made links with the following RobMoSys ITPs:

- **Mood2Be** (OC1, completed), https://robmosys.eu/mood2be/. Exporter from Papyrus for Robotics behavior tree models to the XML format supported by BehaviorTree.CPP.
- **eITUS** (OC1, completed), https://robmosys.eu/e-itus/. Components' faults modeling and code-generation to support safety analysis through simulation-based fault injection
- **Miranda** (OC2, completed), https://robmosys.eu/itp-2-2-2/. Modeling and code-generation for a field trial application of an inspection robotic rover in the context of a nuclear site.
- **SafeCC4Robot** (OC2, in progress), https://robmosys.eu/safecc4robot/. Improve the modeling interface to integrate methodological guidance for the development process and supporting contract-based validation approaches.
- **SCOPE** (OC2, in progress), https://robmosys.eu/scope/. Exporter from Papyrus for Robotics models of skills, behaviors and system architecture to the SCOPE toolchain for the analysis of correct task execution.
- **ForSAMARA** (OC2, in progress), https://robmosys.eu/forsamara/. Extension of task-based hazard and risk analysis according to the project needs.

3.2.1 Licenses

Eclipse Public License 2.0

3.2.2 Latest Release

V0.8.0, 2020-09

3.2.3 Source Repositories

https://git.eclipse.org/c/papyrus/org.eclipse.papyrus-robotics.git

4 Conclusion

Two open source projects have been created during the course of the RobMoSys projects: Eclipse SmartMDSD, an Eclipse-based Integrated Development Environment (IDE) for robotics software development and Papyrus for Robotics, a graphical editing tool for robotic applications that complies with the RobMoSys approach.

Both projects successfully are hosted at the Eclipse Foundation, providing a solid basis for further dissemination and exploitation of these projects beyond the lifetime of the RobMoSys project. Both projects have been presented and promoted at various occasions such as yearly EclipseCon, Eclipse Newsletters, etc.

https://projects.eclipse.org/projects/modeling.smartmdsd

https://www.eclipse.org/papyrus/components/robotics/