RobMoSys

H2020—ICT—732410

RobMoSys

COMPOSABLE MODELS AND SOFTWARE FOR ROBOTICS SYSTEMS

DELIVERABLE D6.5: 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 7

Document number: D6.5

Document title: Eclipse Project and Eclipse Project Proposal

Version: 1.0

Due date: June 30th, 2019

Delivery date: 2019-07-01

Nature: Report (R)

Dissemination level: **Public**

Editor: Maria Teresa Delgado (EFE)

Author(s): Maria Teresa Delgado (EFE)

Reviewer: Huascar Espinoza (CEA)

Executive Summary

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.

It is a challenge for any project to feel comfortable when open sourcing its code on such platform. 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. There is almost no chance unless you are a big industrial player.

It is for this reason that 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.

In this document, we explain the benefits of joining an open source community like the Eclipse Foundation. Section 2 provides an overview of Open Source and the Eclipse Development Process. More specifically, Section 1.1 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 1.2 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 0, the two assets that the project has currently selected to release as Open Source: are presented: the SmartMDSD tool and the Papyrus4Robotics extension. A detailed description of the current status of these is provided in this section.

Content

1 The Eclipse Foundation and the Eclipse Development Process

1.1 What is Open Source?

1.1.1 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.

1.1.2 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.

1.1.3 Why do we need Open Source?

From Small and Medium-sized Enterprises[1] to large organizations[2], 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. This choice has nothing to do with altruism: it is a business strategy.

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.

- **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 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.

[1] 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.

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

  - **Lower total cost of ownership:** companies can rely on 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 will not only reuse OSS, but is also open sourcing some parts of its own code to make it sustainable, even beyond the end of the research project. By experience, when code is not open sourced, it is really difficult to keep enough interest from a large part of 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.

We might be able to identify other development axes later on in during the project.

1.1.4 Why do we need Open Source communities?

An OSS community is the keystone for the sustainability of our project. If we are not able to attract and convince people that our code is worth spending time and resources on testing it, providing feedback, providing patches, and contributing in general, then all the intrinsic value of OSS is lost.

In other words, without Maturity, Quality, Cost of Acquisition, Control, and Security, the Sustainability of our code is nearly impossible. 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:

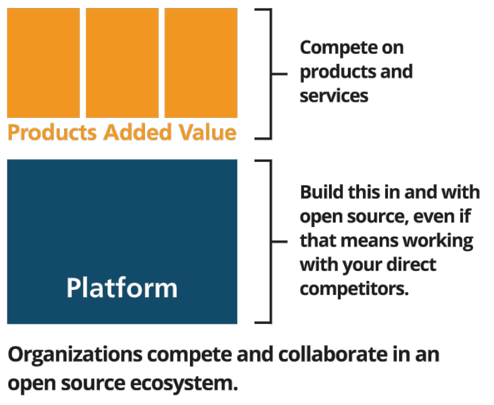
Products Added Value
Compete on products and services
Platform
Build this in and with open source, even if that means working with your direct competitors.
Organizations compete and collaborate in an open source ecosystem.

Figure 1. A business friendly ecosystem based on extensible platforms

  - 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 has become an ISO/IEC Standard.

- **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 its community: number of committers, number of commits, regularity of the release, and the quality and quantity of assets built around the OSS is 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.

1.2 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 comprised 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.

1.2.1 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 assists RobMoSys consortium for the sustainability of its project and will help with the best practices of open source.

1.2.2 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:

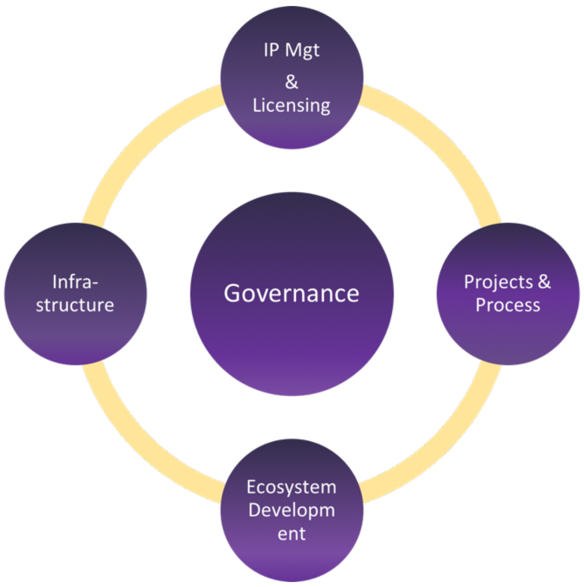

Figure 2. Pillars of open collaborations

- **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 9 active working groups:

- **GEMOC RC** The GEMOC Research Consortium (http://gemoc.org/) is an open and collaborative initiative which supports the development, coordination, and dissemination of the research efforts on the use and globalization of modeling languages. GEMOC develops techniques, frameworks, and environments to facilitate the creation, integration, and automated processing of heterogeneous modeling languages.
- **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 preminent 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.
- **Jakarta EE:** The Jakarta EE Working Group (https://jakarta.ee/) enables Java ecosystem players to collaborate on advancing Java EE and Jakarta EE to support moving mission-critical applications and workloads to the cloud. This initiative focuses on cultivating the business interests associated with Eclipse Enterprise for Java (EE4J) technologies, which includes managing the specifications process related to EE4J, and the Jakarta EE brand.
- **LocationTech:** The LocationTech Working Group (https://www.locationtech.org/) is dedicated to developing advanced location-aware technologies by providing a collaborative environment for stakeholders to jointly develop Big Geospatial Data processing and visualization components, tools, and frameworks. LocationTech manages the quality and maturity of tools and components and establishes sustainable commercial services and ecosystems around them.
- **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.
- **OpenMDM:** The openMDM Working Group (http://www.openmdm.org/) fosters and supports an open, collaborative and innovative ecosystem by providing tools and systems, qualification kits and adapters for standardized and vendor-agnostic measurement data management in accordance with the ASAM Open Data Services (ODS) standard. The openMDM framework facilitates the further improvement and international distribution of openMDM®
- **openMobility**: (https://openmobility.eclipse.org/) Mobility modelling and traffic simulation technologies are important tools for testing autonomous driving functions, developing new mobility solutions and for creating digital twins of urban areas. This working group shapes and fosters the development of required software tools and frameworks based on validated mobility models in order to provide a common platform for industrial applications and academic research.
- **openPASS:** The openPASS Working Group (https://wiki.eclipse.org/OpenPASS-WG) promotes a collaborative and innovative ecosystem by offering tools, systems, and adapters for a standardized, openly-available and vendor-neutral platform for traffic scenario simulation. Together with members like BMW, Daimler, and Volkswagen Group of America, openPASS develops core frameworks and modules for the safety assessment of driving assistance and automated driving systems.
- **Science:** The Science Working Group (https://science.eclipse.org/) represents a collaborative effort involving industry, academia, and government with the purpose of developing reusable

open source software for scientific research. This Working Group seeks to create a set of tools and frameworks to help accelerate scientific research while enabling a collaborative approach to producing technologies used for the interdisciplinary analysis of scientific data.

1.2.3 The Eclipse Industrial Robotics Working Group

The Eclipse Industrial Robotics Working Group is an initiative that is still under discussion at the Eclipse Foundation and will hopefully become a reality by the end of year 2020. This working group will be established by an ecosystem of companies and individuals that are working together to establish an industry platform for robotics based on open technologies. It provides open source implementations of standards, services and frameworks that enable an open ecosystem for Robotics.

Due to its nature, the RobMoSys project is highly interested in the creation of this WG. In fact, RobMoSys is actively participating in the creation process and expects to benefit from its creation. Hopefully, it will not only be a great landing zone to promote the RobMoSys technology and to gather early and involved adopters, but would also be a great open source community to collaborate with.

1.3 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.

1.3.1 Definitions

1.3.1.1 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.

1.3.1.2 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 force a project to accept a new committer.

1.3.1.3 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).

1.3.1.4 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 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.

1.3.1.5 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 election committer 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 a privilege that 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".

1.3.1.6 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.

1.3.1.7 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.

1.3.2 Development Process

1.3.2.1 Overview

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

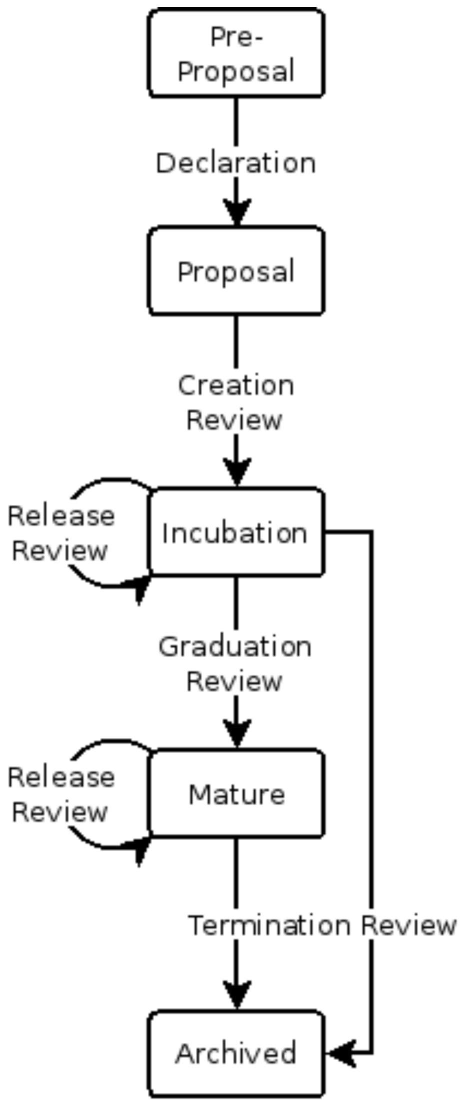

Figure 3. Eclipse Development Process

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 dwindling 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.

1.3.2.2 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[3].

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[4]. 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. Think of this as 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)?

[3] https://wiki.eclipse.org/Development_Resources/HOWTO/Pre-Proposal_Phase
[4] https://wiki.eclipse.org/Development_Resources/HOWTO/Project_Naming_Policy

- Source Code Section:
  - Repository Source Type
  - Source Repositories
- People:
  - A project needs a project lead
  - Initial Committers
  - Mentor(s)
  - Interested Parties: Who is interested in this project? This could be individuals or companies.

1.3.2.3 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.

1.3.2.4 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 "code dumps" and thus projects must be sufficiently staffed for forward progress.

1.3.2.5 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.

1.3.2.6 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.

1.3.2.7 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.

1.3.2.8 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.

2 Eclipse RobMoSys project proposal

The project has defined the architecture of the solution, and specified the core parts and the reference implementation which will be delivered as an open source project in the Eclipse Foundation forge.

We are encouraging the consortium to deliver their initial contribution sooner than later giving like that more chance to the early adopters to try the code, provide feedback and, we hope, adopt RobMoSys technologies.

At the moment., the project is working on an Eclipse Project proposal related to the SmartMDSD tool, lead by ULM, and an extension for the Papyrus modeling tool (Papyrus4Robotics).lead by CEA. In the following sections an overview of the current status of these two assets will be provided.

2.1 SmartMDSD Eclipse Project Proposal (draft)

In this section, the draft proposal of the SmartMDSD Eclipse Project Propsal is presented. Currently the proposal misses license(s), project id, and mentors, that will hopefully be decided soon in order to move forward to the next development phase.

2.1.1 Basics

This proposal is in the Project Proposal Phase (as defined in the Eclipse Development Process) and is written to declare its intent and scope. We solicit additional participation and input from the community. Please login and add your feedback in the comments section.

2.1.1.1 Parent Project

Eclipse Modeling Project

2.1.1.2 Proposal State

Draft

2.1.2 Background

The SmartMDSD and its main development asset, the SmartMDSD Toolchain, is being developed and maintained by Service Robotics Research Center at Ulm University of Applied Sciences. The initial contributors who will kickstart and actively maintain SmartMDSD are working at this organization.

The SmartMDSD Toolchain, the main development of the proposed eclipse project, has a long history (see http://robmosys.eu/wiki/baseline:environment_tools:smartsoft:start#smartmdsd_toolchain_and_the_smartsoft_framework). The SmartMDSD Toolchain started in 2009 using the Itemis Open-Architecture Ware (OAW) and Papyrus UML, then between 2013 and 2016 used Xtext, Xtend and Papyrus UML. It has currently moved towards using the latest Eclipse Modeling technologies based on latest Xtext, Xtend and Sirius plugins. This major rework enabled additional internal flexibility and conformance to the European Union's Horizon 2020 research and innovation programme "RobMoSys" (https://robmosys.eu/). This is initial version that will be included in the SmartMDSD eclipse project is the fourth generation of the tooling.

Since 2008, several research projects on German national level and the European level have contributed to the background of this eclipse project. All these projects have contributed knowledge, structures and implementations for robotics software development that will be included in this eclipse project: ZAFH Servicerobotik (Germany, state of Baden Württemberg), FIONA (European level), iserveU (German national level), etc. (see slide 6, http://www.servicerobotik-ulm.de/drupal/sites/default/files/2017-07-11-What-is-SmartSoft.pdf)

The underlying structures for robotics software development that become accessible through the tooling provided by SmartMDSD date back to 1998. They have been constantly refined and enhanced and are now being formalized on the European level via the European Union's Horizon 2020 research and innovation programme "RobMoSys" (https://robmosys.eu/) with the support of the Eclipse Foundation as a consortium partner.

A lot of robotics software components exist and are being used. The initial code-base of SmartMDSD builds upon the third generation of the SmartMDSD Toolchain. It is very mature as it is being used in operational environments (technology readiness level 6). Its development artifacts (software components and systems) are shipped worldwide within products of FESTO and REC.

2.1.3 Scope

SmartMDSD will focus on model-driven tooling for service-oriented and component-based robotics software development (hence the project name). The main outcome in this project is the SmartMDSD Toolchain, an IDE for robotics software development.

- SmartMDSD will mainly support the "SmartSoft Framework", a service-oriented component-based framework for software development in service robotics.
- SmartMDSD is planned to support the "SeRoNet approach", an approach for system composition based on OPC UA.
- SmartMDSD will conform to composition structures defined by the European Project RobMoSys
- Other infrastructure (e.g. meta-models, examples, etc.) will be hosted here as well.
- The SmartSoft Framework itself is not part of the proposed eclipse project. Tooling provided within the proposed eclipse project is independent of the underlying framework. The frameworks that currently exist are hosted externally (vendor specific repositories) and support the CORBA (open source), ACE (open source), DDS, and OPC UA (planned open source) communication infrastructure.
- Apart from examples, the project will not host robotics software components for composition. Due to the robotics ecosystem organization and separation of roles, these (open or closed-source) software components originate from vendor-specific repositories. The project focuses on (open) tooling for (open) structures to enable composition.

2.1.4 Description

SmartMDSD provides tooling for robotics software development to support system composition in an robotics software business ecosystem supporting different roles. This project will maintain the eclipse-based tooling and necessary infrastructure (e.g. meta-models, code-generators).

Model-driven tooling
Tooling developed in this project is the "SmartMDSD Toolchain", an Eclipse-based Integrated Development Environment (IDE). The model-driven SmartMDSD Toolchain provides support and guidance to apply structures for system composition in an software ecosystem.

Structures
The SmartMDSD Toolchain supports users in applying the necessary structures to enable composition in an robotics ecosystem. The tooling enables domain experts to model robotics domain structures, enables component suppliers to provide software components and enables system builders to flexibly combine and re-combine ("compose") them to new applications according to their needs.

Main target is the SmartSoft Framework

SmartMDSD covers the complete workflow from domain structures to component development to system composition. SmartMDSD will mainly support the "SmartSoft Framework", a service-oriented component-based framework for software development in service robotics.

SmartMDSD 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
- http://www.servicerobotik-ulm.de/drupal/?q=node/20
- 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

2.1.5 Why Here?

SmartMDSD is contributing to a composition-oriented approach for robotics software development in an software business ecosystem. A key enabler for this vision are structures that need to be open in the form of meta-models. These structures must be open and need to be maintained by an active community. An eclipse project would support in kickstarting this open community and in providing a frame for further evolvement.

The members of the proposed project have a strong background in the needs of robotics and how to address them. Both the robotics and the eclipse community can benefit from fostering a close link via this eclipse project:

- Model-driven engineering demonstrated to be a key driver to manage system complexity. Robotics can benefit from knowledge in tooling and model-driven engineering which is both well established in the eclipse community.
- Through the proposed project, the Eclipse community can extend its scope to the robotics domain. The main outcome of "SmartMDSD" will be the "SmartMDSD Toolchain", a matured robotics software development IDE that is well known in the service robotics domain.
- The very same arguments hold true for choosing "Eclipse Modeling Project" as the parent project: SmartMDSD heavily applies tools from the Eclipse Modeling Project.

2.1.6 Legal Issues

Additional comments to licensing: The current code is all BSD 3-clause. We own IP of all parts. If it is still an option to have our background code and our future commits BSD, this would be our favorite solution. But I cannot choose this in the form. If this is not possible, we are interested in dual licensing BSD+EPL.

2.1.7 Project Scheduling

An initial contribution is to be expected very soon after the infrastructure is available.

We will now contribute the fourth generation, the SmartMDSD Toolchain "v3" as an initial contribution which is already quite advanced and will be usable by day 1.

According to that, we expect that the incubation phase can be kept short as the project has a history since 2009, is in active use by a broad user basis, and the development supported by two publicly funded projects (RobMoSys and SeRoNet, see Background and Interested Parties).

2.1.8 Future Work

Previous generations of the code have been demonstrated in operational environments (TRL 6). Since the version to be committed initially is a new generation of tooling in that existing line, we expect its usability and stability to improve to TRL 6 in the first half year. We expect that the existing user basis will support us with valuable feedback.

The project is conformant to the composition structures of the H2020 RobMoSys project. RobMoSys runs until 12/2020 and the project is continuously updated to continue being conformant to RobMoSys. RobMoSys offers cascaded funding through open calls which will broaden the community around this Eclipse project.

2.1.9 People

2.1.9.1 Project Leads

Dennis Stampfer

stampfer@hs-ulm.de

2.1.9.2 Committers

Dennis Stampfer (stampfer@hs-ulm.de)

2.1.9.3 Interested Parties

It is expected that the research and development projects RobMoSys and SeRoNet will contribute to the proposed "SmartMDSD" Eclipse project.

RobMoSys ("Composable Models and Software for Robotics Systems", https://robmosys.eu), a project funded in the European Union's Horizon 2020 research and innovation programme. RobMoSys is working towards an EU digital platform for industrial robotics. RobMoSys brings to the robotics software community a model-driven support, specifically targeting the autonomous nature of robotics systems and their validation. RobMoSys adopts a composition-oriented approach that manages, maintains and assures system-level properties, while preserving modularity and independence of existing robotics frameworks and code bases.

SeRoNet ("Service Robotics Network", https://www.seronet-projekt.de) is a German national project funded by the Federal Ministry for Economic Affairs and Energy (Bundesministerium für Wirtschaft und Energie, BMWi). SeRoNet is creating an IT platform for robotics users, component suppliers, and system integrators. SeRoNet aims at reducing the overall costs to service robotics software development.

The members of the proposed eclipse project are active in the euRobotics aisbl Topic Group on "Software Engineering, System Integration, System Engineering". They thus maintain a link to the service robotics community.

2.1.10 Source Code

2.1.10.1 Initial Contribution

The initial contribution will be the SmartMDSD Toolchain. It is fully functional in modeling the complete composition workflow and code-generation to the "SmartSoft Framework". See section "Background" for more detailed information.

The initial contribution will consist of the (at that time) current release of the SmartMDSD Toolchain: http://www.servicerobotik-ulm.de/files/SmartMDSD_Toolchain/releases/

2.1.10.2 Source Repository Type

GitHub

2.2 Papyrus4Robotics

Papyrus4Robotics is a RobMoSys-conformant implementation of Papyrus that provides a Robotics-customized model-based development graphical environment. It manages complexity of robotics development 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/

Papyrus4Robotics builds upon an existing ecosystem around Papyrus which is an Eclipse project: www.eclipse.org/papyrus. Papyrus is part of the Eclipse Polarsys working group where a Papyrus Industrial Consortium has been created to federate Papyrus users (industrials and academics) and Papyrus tool/service providers: https://www.polarsys.org/papyrus-ic/about

Papyrus4Robotics 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.

Papyrus4Robotics 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 Papyrus4Robotics are:

- **Safety Analysis**. Papyrus4Robotics features a safety module to perform dysfunctional analysis on system architecture's components, including (but not limited to) Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA). Model-based safety analysis is enabled by a dedicated DSML, modeling views and a set of analysis and report generation modules.
- **Performance Analysis** (integration on-going). Papyrus4Robotics supports the Architectural Pattern for Stepwise Management of Extra-Functional Properties, with a focus on the timing properties of software component architectures. Concretely, it enables the analysis of end-to-end response times and resource utilization of component compositions by leveraging concepts in Compositional Performance Analysis (CPA), such as paths and cause-effect chains.
- **Code-Generation** (integration on-going). Papyrus4Robotics includes generators that transform models of software component architectures, platform descriptions and deployment specifications into code. Currently, we support code generation for OROCOS (C++) and we are working on ROS 2 code generators, to be fully available in October 2019.

2.2.1 Current Developments

During the last months of the project until month M30, the Papyrus4Robotics strategy has been refined to cover the following aspects:

- ***Modular and Role-Based Design***. Different robotics stakeholders play their role at the adequate abstraction level in the tool workflow. Further information at:
  - https://wiki.eclipse.org/Papyrus/customizations/robotics/getstarted
  - https://wiki.eclipse.org/Papyrus/customizations/robotics/modular
- ***Agile Risk Assessment***. Robot behavior can be specified by using Behavior Trees (BT) and configured by defining task models using skill definitions. Risk assessment is performed assessing operational hazard situations and mitigation measures. Further information at:
  - https://wiki.eclipse.org/Papyrus/customizations/robotics/hara
- ***Compositional Safety Analysis***. Component suppliers specify potential faults and their propagation through component ports. The system builder uses component fault models to analyze system-level propagation and fault trees. Further information at:
  - https://wiki.eclipse.org/Papyrus/customizations/robotics/fta

- ***Robustness Simulation***. Robustness of robotics systems can be assessed against faults injected in controlled experiments. We can then generate the code of controllers, including the fragments of the injected faults. Generated code enables evaluation of safety properties by means of normal and faulty simulations. The eITUS Integrated Technical Project (ITP) used the Papyrus4Robotics code-generation capabilities to target Orocos-RTT and Gazebo to enable early safety assessment of robotics systems using fault injection simulations. Further information at:
  - https://robmosys.eu/wiki/community:safety-analysis:start

2.2.2 Release Scheduling

Figure 4 shows the full roadmap (until June 2020) of Papyrus4Robotics. During the next period (June 30 to September 30), modelling capabilities are being added for world modelling as well as analysis capabilities for performance and FTA cause-effect chain. AT the same time a first full version of ROS 2 code generation will be released. Until January 31, these functionalities will be tested and improved by using case studies. In the last period until June 30 2010, runtime safety monitoring capabilities will be added as well as more complex functionalities for code generation.

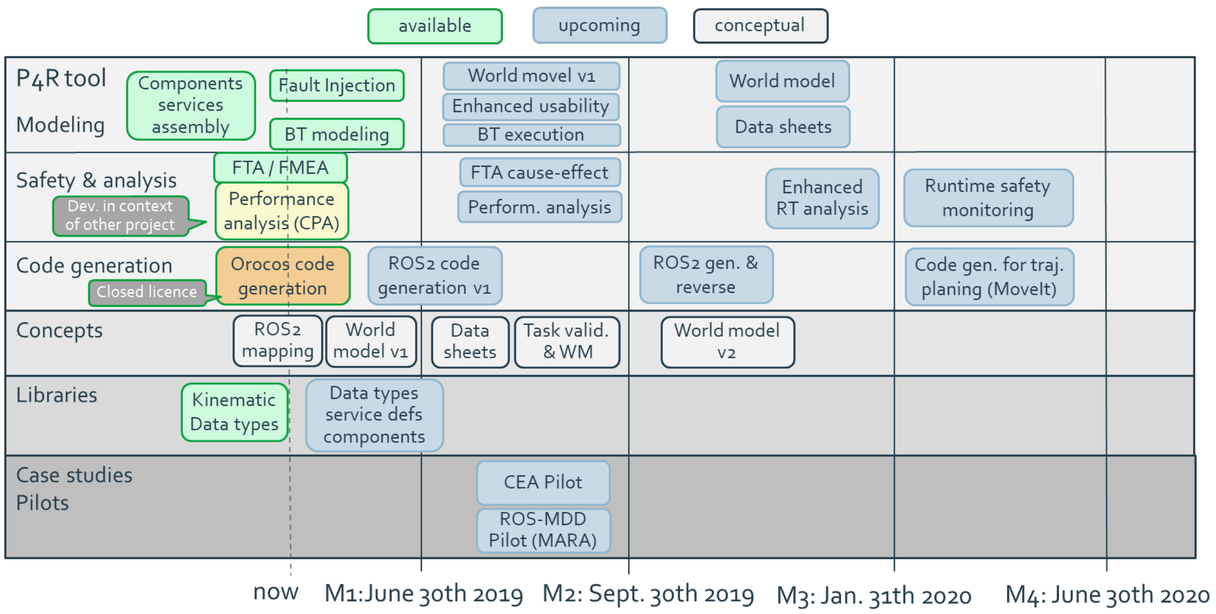

Figure 4. Papyrus4Robotics Roadmap

3 Conclusions

The RobMoSys projects has invested a great deal of effort into transforming its results in open and available tools for the community. So far two main outputs in this regard have been defined: the SmartMDSD Eclipse Project Proposal and the Papyrus4Robotics extension.

We will continue to work as a consortium to deliver more results like these in the future, also in collaboration with the ITPs.

Our hope is that this assets will become sustainable after the RobMoSys project lifecycle comes to an end, and for these reason the consortium is leveraging the know-how, infrastructure and services offered by the Eclipse Foundation, which is a partner in the consortium.