

ECSEL Research and Innovation actions (RIA)



AMASS

Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

AMASS open source platform project proposal D7.3

Work Package:	WP7 Industrial Impact and Community Building
Dissemination level:	PU = Public
Status:	Final
Date:	31 January 2017
Responsible partner:	Gaël Blondelle (Eclipse Foundation Europe GmbH)
Contact information:	gael.blondelle@eclipse.org
Document reference:	AMASS_D7.3_WP7_EFE_V1.0

PROPRIETARY RIGHTS STATEMENT

This document contains information, which is proprietary to the AMASS Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the AMASS consortium.

This deliverable is part of a project that has received funding from the ECSEL JU under grant agreement No 692474. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and from Spain, Czech Republic, Germany, Sweden, Italy, United Kingdom and France.

Contributors

Names	Organisation
Gaël Blondelle, Philippe Krief	Eclipse Foundation Europe GmbH
Huascar Espinoza	TECNALIA Research & Innovation
Barbara Galina	Mälardalens Hoegskola

Reviewers

Names	Organisation
Barbara Galina (Peer Reviewer)	Mälardalens Hoegskola
Stefano Puri (Peer Reviewer)	Intecs SPA
Silvia Mazzini	Intecs SPA
Cristina Martínez	TECNALIA Research & Innovation
Jose Luis de la Vara	Universidad Carlos III de Madrid

TABLE OF CONTENTS

Executive Summary.....	6
1. Introduction	7
2. Introduction to Open Source Software	8
2.1 How to manage Common Pool Resources?	8
2.2 What is Open Source Software?.....	8
2.3 Why do we need Open Source?	9
2.4 Why do we need Open Source Communities?.....	10
3. The Eclipse Foundation and PolarSys	12
3.1 The Eclipse Developer Community.....	12
3.2 Eclipse Working Groups.....	12
3.3 PolarSys	14
4. The Eclipse Development Process.....	16
4.1 Project Structure and Organization.....	16
4.2 Committers.....	16
4.3 Code and Resources	16
4.4 Leaders	16
4.5 Committers and Contributors	17
4.6 Project Plans	17
4.7 Mentors	17
4.8 Development Process.....	18
4.9 The Eclipse IP Process.....	19
4.10 Proposal.....	19
4.11 Reviews.....	20
4.12 Creation Review	21
4.13 Graduation Review	21
4.14 Release Review	21
4.15 Termination Review	21
4.16 Releases.....	21
5. The PolarSys AMASS Proposal	22
5.1 An Ecosystem of Open Source Projects.....	22
5.2 OpenCert as the Core AMASS Open Source Platform.....	24
5.2.1 Creation of OpenCert	24
5.2.2 OpenCert IP Analysis	25
5.3 The CHES Toolset	26
5.4 The Eclipse Process Framework Project	27
5.5 Next Steps for the AMASS Open Platform	28
5.5.1 Integrate the contributions of the AMASS core prototype	28
5.5.2 Elect new committers	28
5.5.3 Releases and release reviews.....	28
6. Conclusions	30
Abbreviations and Definitions	31
References.....	32
Appendix A: OpenCert Proposal	33
Appendix B: OpenCert Proposal Progress Tracking	37

List of Figures

Figure 1.	A business friendly ecosystem based on extensible platforms	10
Figure 2.	Pillars of open collaborations	13
Figure 3.	The PolarSys ecosystem	14
Figure 4.	Eclipse Development Process	18
Figure 5.	AMASS Tangible Outcomes	22
Figure 6.	AMASS Reference Tool Architecture	23
Figure 7.	AMASS Open Source platform architecture	23
Figure 8.	OpenCert page	25
Figure 9.	OpenCert mailing list	25
Figure 10.	The Eclipse Parallel IP Process	26
Figure 11.	CHESS project page	26
Figure 12.	CHESS download page	27
Figure 13.	EPF project page	27
Figure 14.	Release Iteration process.....	29

List of Tables

Table 1.	Next steps for the AMASS Open Platform.....	28
-----------------	---	----

Executive Summary

This document first introduces Open Source principles. Then, it presents thoroughly the Eclipse Foundation and the PolarSys Working Group. They are good examples of Business Friendly Open Source ecosystems, PolarSys being devoted to open source tools for the development of embedded systems.

Several sections describe the Eclipse Development process and the Eclipse IP due diligence process that are the two pillars of this business friendly open source approach as they ensure that the code hosted by Eclipse and the Eclipse Working Groups can be reused in different contexts (other open source projects, proprietary products, service offers, ...) without taking legal risks.

Since its inception, and following the work started in previous projects like OPENCROSS [1], CHESS [2], and SafeCer [3], the AMASS consortium decided to publish a significant part of its results as Open Source Software in the framework of the PolarSys Working Group at the Eclipse Foundation. The goal is to make adoption easier, and to establish a de-facto standard. The end of the first year of AMASS, and the second year of the project are the moment when the AMASS consortium are turning a large part of the development in public by directly contributing to AMASS Open Platform composed of OpenCert [4], CHESS [18] and EPF [5] for new developments when they can be licensed in Open Source.

AMASS is a unique opportunity for the OpenCert and CHESS contributors to create traction about their projects by showing activity, by demonstrating that the platform implements more and more features along the three years of AMASS, and by using the open source part of the platform as a widely accessible demonstrator.

1. Introduction

AMASS will **create and consolidate a de-facto European-wide assurance and certification open tool platform, ecosystem and self-sustainable community** spanning the largest CPS vertical markets. The ultimate aim is to lower certification costs in face of rapidly changing product features and market needs. This will be achieved by establishing a novel holistic and reuse-oriented approach for **architecture-driven assurance** (architecture represents a major aspect for ensuring dependability and for meeting assurance and certification needs and requirements), **multi-concern assurance** (compliance demonstration, impact analyses, and compositional assurance of **dependability** aspects), and for **seamless interoperability** between assurance/certification and engineering activities along with third-party activities (external assessments, supplier assurance).

AMASS will publish a significant part of its results, the AMASS Open Platform, as Open Source Software under the Eclipse Public License, in **PolarSys** [6], the **Working Group of the Eclipse Foundation** dedicated to open source solutions for embedded systems. Such open source projects follow a precise process that ensures the application of best practices including Intellectual Property analysis for the code and dependencies, configuration management, continuous integration and release management.

Note that these activities are not pure dissemination activities but rather technical activities that are necessary to ensure the success of the AMASS Open Platform as an open source platform.

Task 7.3 (Creation and Coordination of AMASS open source community) will support the publication of the main results of the AMASS project as an open source platform hosted by Eclipse to implement the AMASS approach. This task will be responsible to guarantee the awareness of the AMASS project as a PolarSys project, its acceptance and incubation as a project hosted by the Eclipse Foundation. This task will actively coordinate and contribute to the creation and development of the open source community.

In April 2016, GitHub reported having more than 14 million users and more than 35 million repositories, making it the largest host of source code in the world. Based on such huge numbers, how can a project that opens source code on GitHub expect to (1) reach their targets, (2) get the desired attraction and (3) build a community that will contribute to the sustainability of the code? There is almost no chance unless you are named Google, Microsoft or Facebook.

For this reason, the AMASS project consortium decided to work closely with the Eclipse Community to collaborate with the PolarSys Working Group and benefit from their developer network and from their experience in building sustainable communities.

This document explains the benefits in joining an open source community like the Eclipse Foundation. After a brief reminder of what *open source software* stands for and why an open source community is important to a project like AMASS, this document explains what the best practices of open source communities (governance, IP management, collaborative infrastructure, and recommended development process) are and why they contribute to the sustainability of the project. Finally, the document describes how the AMASS project decided to leverage several Eclipse and PolarSys projects mainly OpenCert [4], CHESS [2] and EPF [5] to put together the AMASS Open Platform. It then describes the next steps necessary to create a sustainable open source project.

2. Introduction to Open Source Software

Deciding to contribute to the open source, as a consumer or a 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 is significant, it is important to feel guided.

2.1 How to manage Common Pool Resources?

Elinor Ostrom [7][8], Nobel Prize of Economy in 2009, designed 8 principles to manage stable Common Pool Resources (CPR):

1. **Clearly defined boundaries** (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.

2.2 What is Open Source Software?

The Open Source Initiative [9] provides a very good definition of Open Source Software (OSS) and defines it in *10 commandments* [10]:

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 in compiled form.
3. **Modifications and derived works:** The license must allow modifications and derived works, and must allow to distribute them 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 endeavour.
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.

2.3 Why do we need Open Source?

From Small and Medium-sized Enterprises to large organizations, lots of companies have adopted and contributed for decades to one or more open source communities like the Apache Software Foundation, the Eclipse Foundation, the Linux Foundation or the OpenStack Foundation. This choice has nothing to do with altruism: it is a business strategy.

Actually, organizations adopt open source software for many reasons, including:

- **Maturity of the model:** Even if it is not systematic, there are numerous examples of projects and products based on Open Source Software (OSS) that are more reliable and sustainable than 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 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 donated 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 SMEs 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 is 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:** because the software is freely accessible and because there is a network of skilled experts able to give trainings, it is 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 is simply a matter of modifying it to add the functionality needed by the project.
 - **Time to Market is shorter:** products do not 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 OSS community for maintenance and, by joining the community, they mutualize maintenance costs.

- **No Vendor Lock-In:** Organizations do not depend on the status of the subcontractor, who originally built the software. In OSS, 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.
- **Auditability:** Some users consider OSS more secure and stable than proprietary software, mainly because they can check the source code, identify and fix bugs. The efforts are mutualized with the other community members, which results in better and stable source code.

2.4 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 and Control, the Sustainability of our code is nearly impossible. And vice-versa, Sustainability is a great indicator demonstrating the Maturity, the Quality and the Control of the code.

Therefore, an OSS community has four main roles:

- **Sustainability:** This ensures the sustainability of the 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.

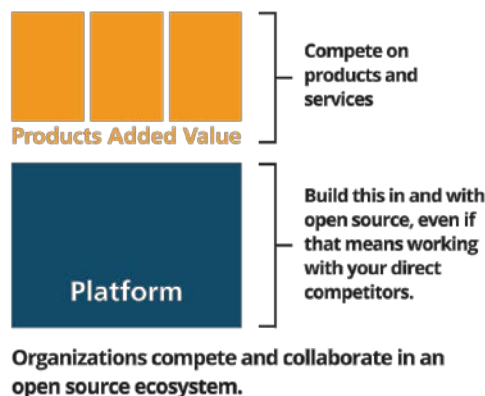


Figure 1. A business friendly ecosystem based on extensible platforms

- **Standardization:** Open Source has proved to be a very good vector of standardization. Here is an outstanding example: 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 a project.

This is the 'snowball' effect: a project attracts early adopters with the quality of 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, etc.), and the interest of the adopters will do the rest. For this reason, it is very important to provide specific support to AMASS early adopters and the project's community as it grows.

3. The Eclipse Foundation and PolarSys

The Eclipse community gathers individuals and organizations that wish to collaborate on business-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 like 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 250 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, open and transparent community to be established around the Eclipse projects. At the end of 2016, the community consists of individuals and organizations from a cross section of the software industry.

3.1 The Eclipse Developer Community

Today, the Eclipse community consists of:

- Over 300 open source projects;
- 130 million lines of code delivered every year;
- Over 1.400 individual committers;
- Over 5 million 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, etc.;
- More than 250 members (including 12 strategic members);
- 75 events organized or co-organized each year, like EclipseCon events, Eclipse Days, Eclipse Summits or Eclipse DemoCamps, etc.

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

3.2 Eclipse 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 provide a **vendor-neutral** governance structure that allows 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 require due diligences 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 five active working groups:

- **openMDM:** The openMDM Working Group (<http://www.openmdm.org/>) wants to foster and support an open and innovative eco-system providing tools and systems, qualification kits and adapters for standardized and vendor independent **management of measurement data** in accordance with the ASAM ODS standard.
- **Internet of Things:** The IoT Working Group (<http://iot.eclipse.org>) is the place to learn about Eclipse technologies developed to make Internet of Things (IoT) development simpler.

- **LocationTech:** The LocationTech Working Group (<https://www.locationtech.org/>) is focusing on open source geospatial technologies.
- **Science:** The Science Working Group (<http://science.eclipse.org/>) is a collaboration of scientists developing software components used for basic **scientific research**.
- **PolarSys:** The PolarSys Working Group (<https://www.polarsys.org/>) was created by large industry players and by tool providers to collaborate on the creation and support of Open Source tools for the **development of embedded systems**.

Due to its focus, the AMASS project collaborates in the **PolarSys working group**.

3.3 PolarSys

The Eclipse PolarSys Working Group is a collaboration of large end-user companies and open source tools providers dedicated to providing industrial grade open source tools for the development of embedded systems.

The goals of PolarSys are:

- Provide means of collaboration between end user companies;
- Organize sustainable commercial services and ecosystems around open source components;
- Foster exchanges between academics and industry partners;
- Manage the quality and maturity of tools and components from early research prototypes through to obsolescence;
- Provide the documents and qualification kits required for certification;
- Recognize project maturity and company know-how and commitment through a branding process;
- Ensure long-term longevity of tools since they must last for a very long time, in some industries up to 30-80 years.

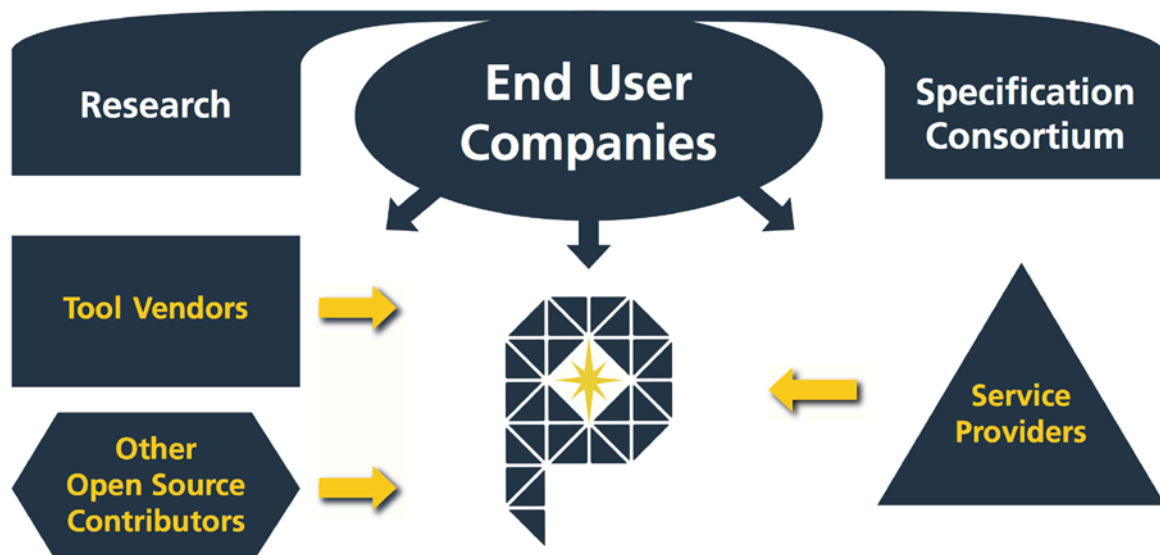


Figure 3. The PolarSys ecosystem

PolarSys projects address the key phases of development in domains such as aeronautics, aerospace, automotive, defense and security, energy, health care, railway, and telecommunications:

- Modelling;
- Compiling, debugging, tracing;

- Application Lifecycle Management process and tools – configuration management, change management, issue tracking, requirement management, project reporting;
- Test and verification frameworks and tools targeting embedded software methods, static analysis, simulation, and early validation Embedded components (RTOS, runtime, middleware, etc.).

All of the PolarSys solutions are based on technology and tools that have been deployed by large-scale systems engineering and embedded systems development teams. All the solutions are open source and are provided free of charge under the Eclipse Public License (EPL).

PolarSys does not intend to systematically re-develop tool components. A lot of very good solutions answering some industrial needs already exist in open source. But most of the time, specific issues like durability or certification are not taken into account. In this case, PolarSys plays its part by completing tool components assets, setting up specific support, and coordinating development and support. This is why PolarSys is open to manage existing tools that satisfy the most important industrial needs.

The PolarSys community selects industrial grade open source solutions that support key aspects of embedded system development, including Systems and Software Engineering, Software Development, and Verification and Validation. Each PolarSys Solution is led by an industry leader who is responsible for managing the development and evolution of the Solution to meet user needs, and, in particular, the requirements for scalability and usability. Because development is provided by an ecosystem of developers, end-user organizations are not dependent on any specific tool supplier. This allows the elimination of vendor lock-in and achieving the required business flexibility.

In addition, the PolarSys Long Term Support (LTS) aims to support PolarSys projects and solutions for up to 10 years, and the Very Long Term Support (VLTS), is designed to provide support as long as users need it.

LTS is a need shared with the rest of the Eclipse Community. For example, proprietary software vendors, which rely on the Eclipse platform for their tools, need to be able to provide support to their customers for ten years in some cases. The implementation of LTS relies on a Common Build Infrastructure [11] (CBI), shared between all the stakeholders, that guarantees predictable and reproducible builds by the application of build and test best practices on LTS releases. LTS releases are chosen as a consensus between stakeholders.

VLTS is specific to PolarSys and targets longer-term support compatible with the lifetime of the products in which systems are embedded. Such systems can last for 25 years for a space mission, for 35 years for a telecommunication infrastructure, or for even more for an aircraft system.

LTS constraints include the capacity to build and test a patched version of the projects in one week, whereas in VLTS, we have lower expectations in terms of reactivity (one month can be acceptable), but have higher expectation in terms of durability.

4. The Eclipse Development Process

As mentioned in the previous chapter, the Eclipse community has created a successful development process for large-scale distributed development that involves many different organizations. Joining the PolarSys Working Group means for the AMASS project to follow this process.

In this section, we will summarize some of the principles driving the Eclipse Development Process (https://www.eclipse.org/projects/dev_process/).

4.1 Project Structure and Organization

A project is the main operational unit at the Eclipse Foundation. Specifically, all the 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.

4.2 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.

4.3 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 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).

4.4 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.

4.5 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, through election, be promoted 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 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 leaders 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 leaders. Unless otherwise specified, "an extended period" is defined as "no activity for more than six months".

4.6 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.

4.7 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.

4.8 Development Process

As indicated on the following figure, projects go through distinct phases. The transitions from phase to phase are open and transparent public reviews.

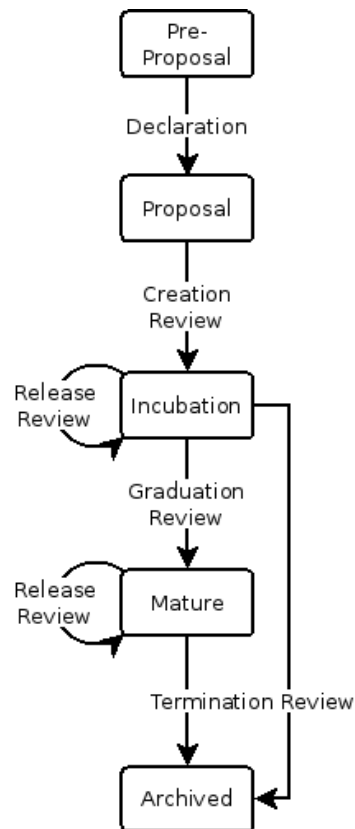


Figure 4. Eclipse Development Process

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.

4.9 The Eclipse IP Process

Eclipse projects are expected to take necessary precautions to mitigate Intellectual Property (IP) risk to adopters. The Eclipse Intellectual Property Policy [12], and the process that implements it, are important so that companies wanting to integrate the open source code into their project can do so knowing that the project can legally be distributed under the agreed-to terms. The IP Due Diligence Process [13][14], managed by the Eclipse IP Team (commonly referred to as the IP Team), is in place to support this.

Code provenance tracking is critical; the Eclipse Foundation needs to know the source of all code that ends up in its repositories. To that end, all new projects are required to make an **initial contribution** before any code is committed to a project's source code repository.

This code originates from three sources at Eclipse:

1. Eclipse Committers
2. Community Contributions (typically in the form of bug fixes)
3. Content developed and maintained somewhere other than Eclipse (other open source projects)

The first two cases are addressed by asking committers and contributors to sign agreements before they contribute code and by making sure that the code has not been copied inappropriately, that licenses are being used correctly, and so forth.

The third case is the most complex one as it implies that the Eclipse Foundation checks all dependencies used in a project, as well as dependencies of dependencies, in order to make sure that proper IP management process is applied to these libraries.

Projects are not be able to make a release until the due diligence on the IP contained in that release—including project code contributions and third-party libraries—is complete.

This IP process applies also to projects hosted by Eclipse Working Groups, so it applies to OpenCert as part of PolarSys.

The Eclipse Foundation uses a special tracker called IPZilla in order to manage Intellectual Property due diligences according the Eclipse IP process.

4.10 Proposal

Any project that wants to join the Eclipse Foundation has to submit a proposal. The proposal has to follow the guidance described in the document (https://wiki.eclipse.org/Development_Resources/HOWTO/Pre-Proposal_Phase).

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

(https://wiki.eclipse.org/Development_Resources/HOWTO/Project_Naming_Policy).

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 [15]).

- **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)?
- **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.

4.11 Reviews

The Eclipse Development Process is predicated on open and transparent behaviour. 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.

4.12 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.

4.13 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.

4.14 Release Review

The purposes of a release review are: (1) to summarize the accomplishments of the release, (2) to verify that the IP Policy has been followed and all approvals have been received, (3) to highlight any remaining quality and/or architectural issues, and (4) to verify that the project is continuing to operate according to the principles and purposes of the Eclipse Foundation.

4.15 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.

4.16 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.

5. The PolarSys AMASS Proposal

5.1 An Ecosystem of Open Source Projects

As illustrated in Figure 5, the AMASS tangible results shall be:

- **AMASS Reference Tool Architecture.** This is the conceptual, modeling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms.
- **AMASS Tool Platform.** It corresponds to a collaborative tool environment that supports CPS assurance and certification. The AMASS Tool Platform is a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project.
- **OpenCert Community at Polarsys.** It manages the project outcomes, for maintenance, evolution and industrialization. The OpenCert Community will be supported by governance board, rules, policies, and quality models.

Since the inception of the AMASS project, the consortium agreed to deliver the AMASS Tool Platform results in open source. This ensures both the usability of the platform to fulfill AMASS objectives and the interoperability between potential specializations of AMASS.

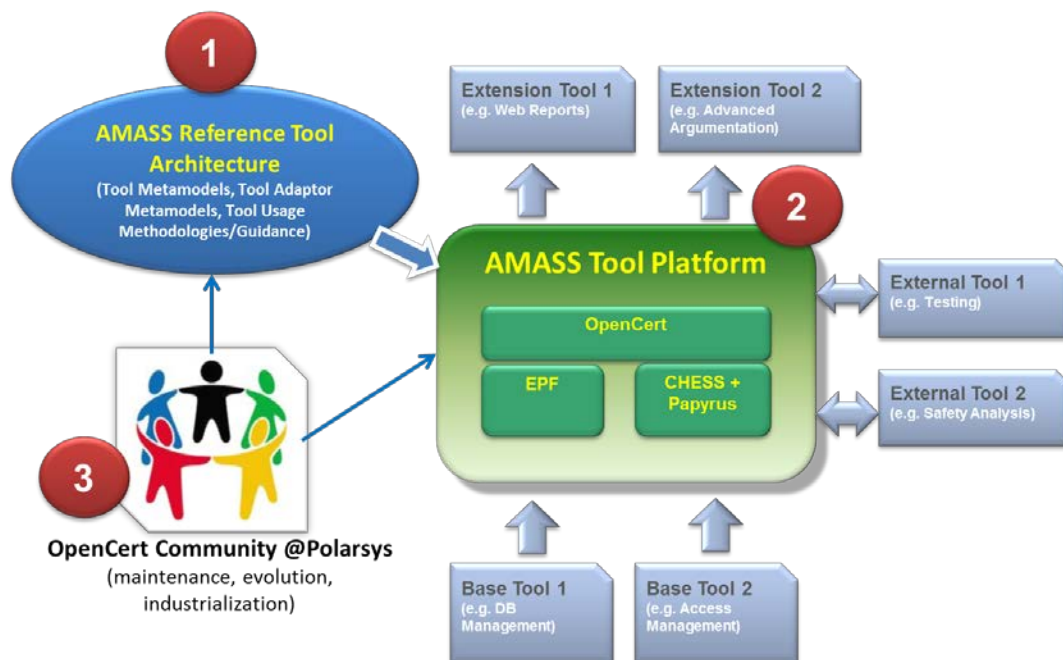


Figure 5. AMASS Tangible Outcomes

The AMASS Reference Tool Architecture represents a virtual entity that embodies a common set of tool interfaces/adaptors, working methods, tool usage methodologies and protocols that will allow any stakeholder of the assurance and certification/qualification activities to seamlessly integrate their activities (e.g., product engineering, external/independent assessment, component/parts supply) into tool chains adapted to the specific needs of the targeted CPS markets. Figure 6 provides a high-level picture of the AMASS Reference Tool Architecture.

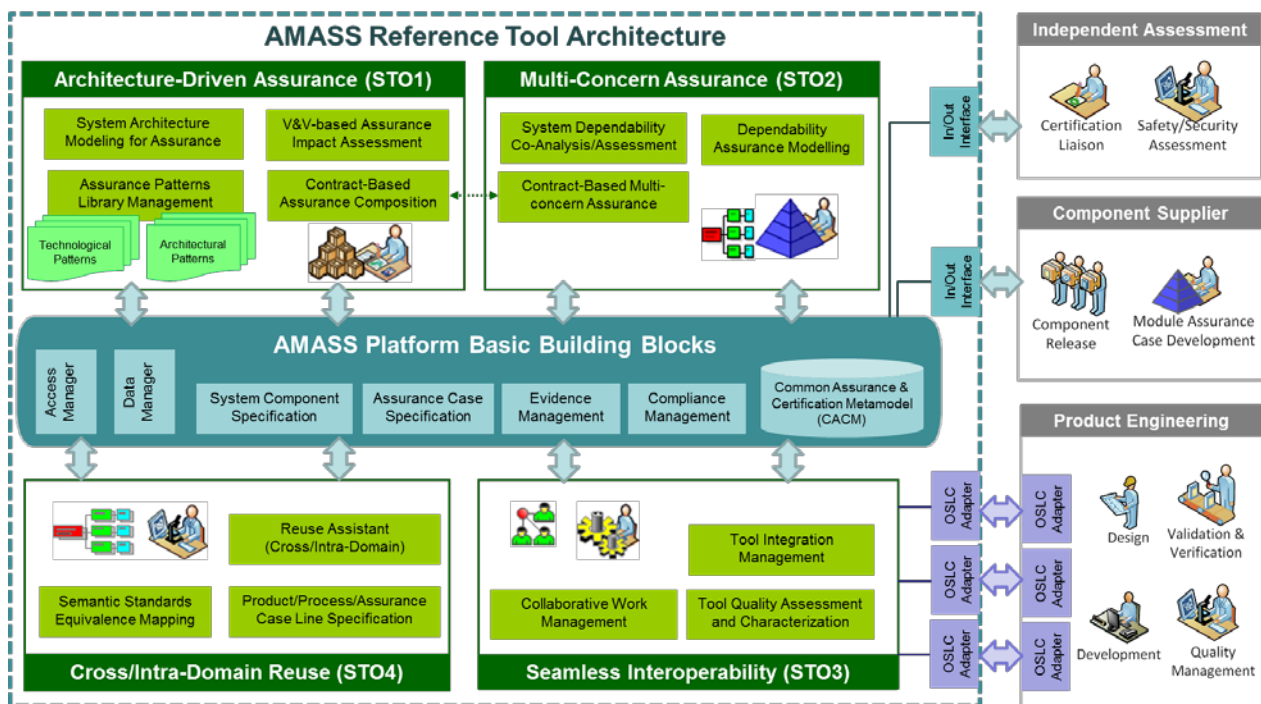


Figure 6. AMASS Reference Tool Architecture

AMASS is not a monolithic platform: the AMASS Tool Platform is composed of several projects that cover the activities of the different work packages as described in Figure 7:

- **OpenCert** is the core of the AMASS open source platform. OpenCert, which was created by the members of the OPENCROSS research project, supports evidence management, assurance case specification and part of the compliance management functionalities from the Basic Building Blocks. It will also include new functionalities implemented during the AMASS project.
- The **CHES** toolset, which was created by the CHES research project and continued by SafeCer, adds support for Architecture-Driven Assurance. The CHES toolset leverages another important PolarSys project, the Papyrus platform for UML design and profiles.
- **EPF Composer**, a pre-existing Eclipse project created by IBM some years ago, is a key component for WP6. The tool is used to describe and support the processes for Cross-domain and Intra-Domain reuse.

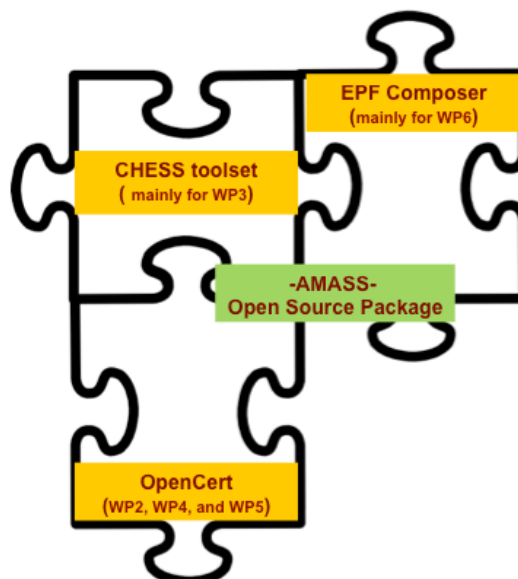


Figure 7. AMASS Open Source platform architecture

The reuse of existing open source projects shows the pragmatism of the team, and the application of a good practice of open source communities that is to “join and contribute to existing communities” instead of “reinventing the wheel”. As such, the AMASS open source platform is a good example of the advantages of open source software that can be combined and extended more easily as all APIs and formats are open.

The Eclipse Foundation tries to enforce a rule that “open source projects must have different names than funded research projects”. Indeed the open source project is supposed to be still active after the end of the research project. AMASS confirms the relevance of this rule as the AMASS project supports development of several open source projects.

In the following sections, we present the PolarSys OpenCert project proposal in depth, give an overview of the PolarSys CHES proposal and a short status about the EPF project.

5.2 OpenCert as the Core AMASS Open Source Platform

5.2.1 Creation of OpenCert

At the end of the OPENCROSS project, one core OPENCROSS project partner (TECNALIA, in its role of OPENCROSS coordinator) submitted the open source licensable results of the project as a PolarSys project: **OpenCert**.

As the project leader, TECNALIA played a key role by:

- Leading the team of committers;
- Submitting the OpenCert project proposal to PolarSys;
- Preparing the OpenCert code, together with Eclipse and PolarSys members, to be released/hosted in open source;
- Operating dedicated code repositories, build chains, test facilities, etc.;
- Fostering exchanges between OpenCert partners and PolarSys industry partners;
- Proposing OpenCert tool enhancements (industry-friendly functionalities, new features, security and reliability features, tool connectors with other PolarSys tools, among others);
- Managing the quality and maturity of OpenCert tools.

In November 2015, TECNALIA submitted the OpenCert proposal to PolarSys. After the recruitment of two mentors, the approval of the name after an analysis of potential trademarks issues, and a period of community review, the proposal was approved in December 2015. In the days following the proposal approval, the Eclipse Foundation created the corresponding PolarSys project with a set of handlers to gather all the assets related to the OpenCert project:

- An overview of the project (see Figure 8) drafted according to the description presented in the proposal;
- A download page that will list all the OpenCert milestones and releases;
- The people initially involved in the project (project leads, committers and mentors);
- Developer resources: documentation and other assets built to assist developers including links to the bug tracker and the project source code repositories;
- The governance: which Working group it is part of;
- How to contact to and participate to the project.



Figure 8. OpenCert page

A dedicated mailing list was also created (opencert-dev@polarsys.org) to facilitate the communication between the developer community and the project team.

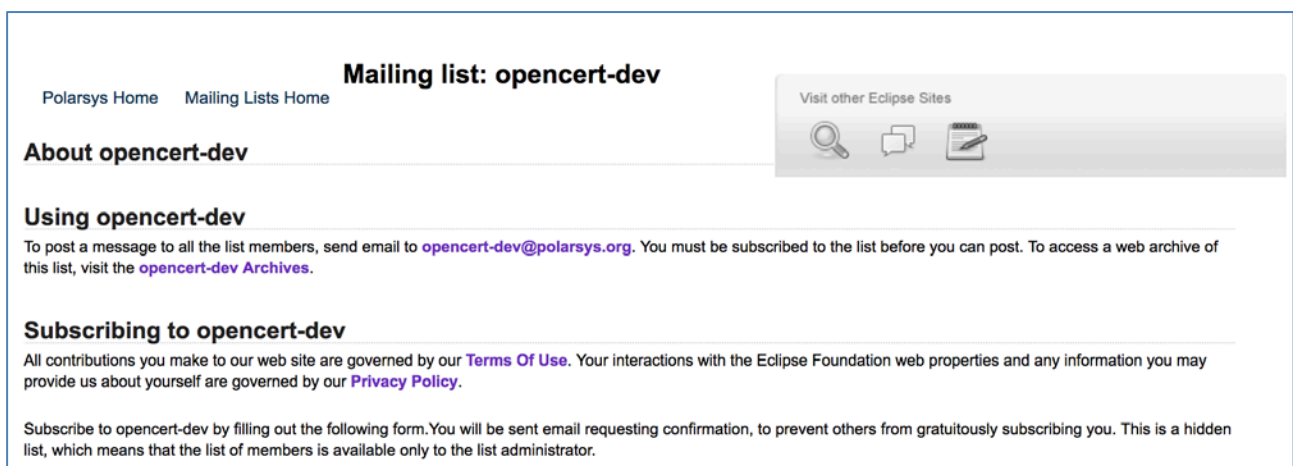


Figure 9. OpenCert mailing list

5.2.2 OpenCert IP Analysis

TECNALIA started the initial contribution for OpenCert in March 2016 [16]. After one month, with the help of the project mentors, the Eclipse IP team authorized the project committers to start working, according to the Parallel IP Process [17].

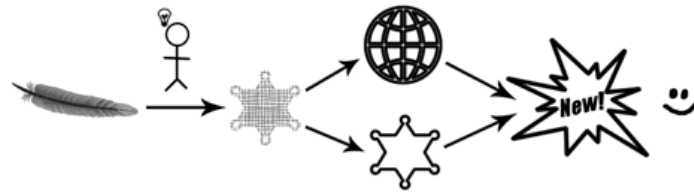


Figure 10. The Eclipse Parallel IP Process

“The parallel IP process allows the check-in to occur before the legal review is complete (but after the preliminary legal approval) - the legal review must be completed before the code is included in a release.”

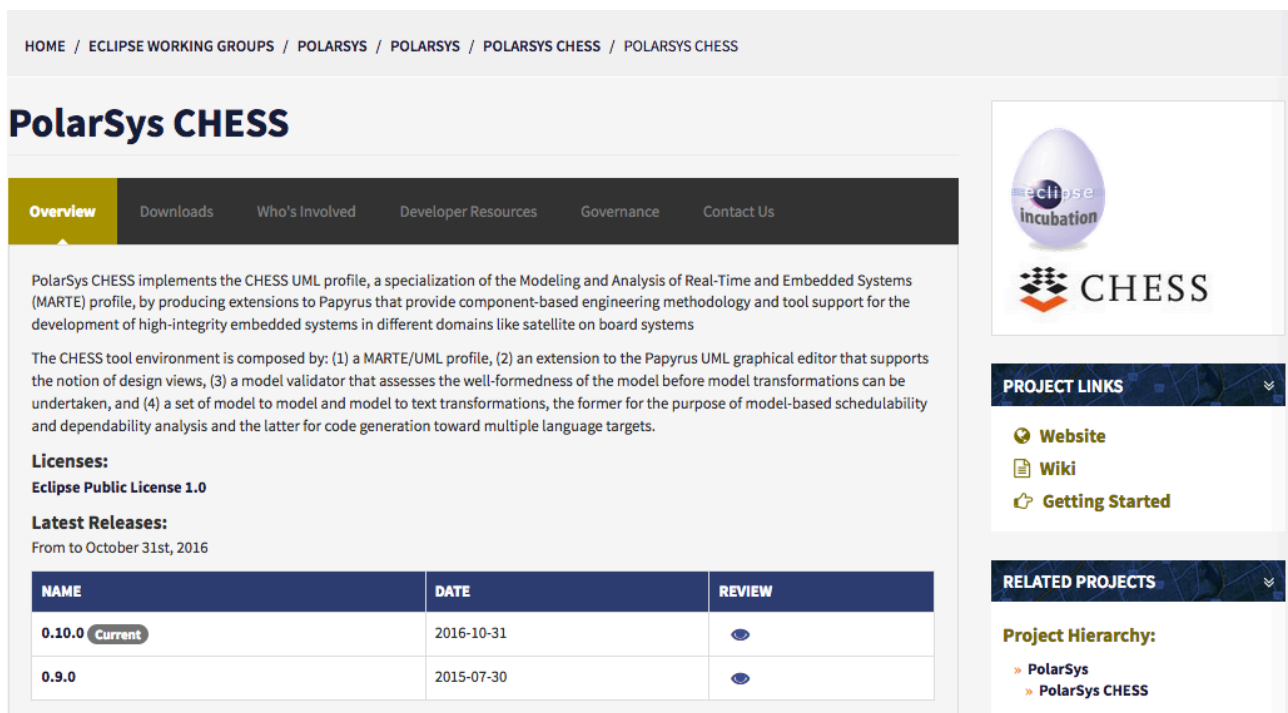
TECNALIA then submitted the initial code developed in the context of OPENCROSS to the OpenCert git repository.

5.3 The CHESSE Toolset

Intecs posted the CHESSE project proposal in October 2013. Intecs played a similar role for CHESSE than TECNALIA played for OpenCert. The project was accepted in Feb 2014 and the development activities started in the following months.

As CHESSE started earlier, the project has been able to go through the full IP due diligence process, and already made few releases (see Figure 11). The release naming (0.9, 0.10) indicates that the project team keeps the right to change the APIs before releasing a 1.0 version. The convention is that every time a project team breaks external APIs, they have to increase the major version number.

CHESSE also provides a page for downloading the current version (see Figure 12).



HOME / ECLIPSE WORKING GROUPS / POLARSYS / POLARSYS / POLARSYS CHESSE / POLARSYS CHESSE

PolarSys CHESSE



Overview Downloads Who's Involved Developer Resources Governance Contact Us

PolarSys CHESSE implements the CHESSE UML profile, a specialization of the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) profile, by producing extensions to Papyrus that provide component-based engineering methodology and tool support for the development of high-integrity embedded systems in different domains like satellite on board systems




The CHESSE tool environment is composed by: (1) a MARTE/UML profile, (2) an extension to the Papyrus UML graphical editor that supports the notion of design views, (3) a model validator that assesses the well-formedness of the model before model transformations can be undertaken, and (4) a set of model to model and model to text transformations, the former for the purpose of model-based schedulability and dependability analysis and the latter for code generation toward multiple language targets.

Licenses:
Eclipse Public License 1.0

Latest Releases:
From to October 31st, 2016

NAME	DATE	REVIEW
0.10.0 Current	2016-10-31	
0.9.0	2015-07-30	

PROJECT LINKS

-  [Website](#)
-  [Wiki](#)
-  [Getting Started](#)

RELATED PROJECTS

Project Hierarchy:

- » PolarSys
- » PolarSys CHESSE

Figure 11. CHESSE project page

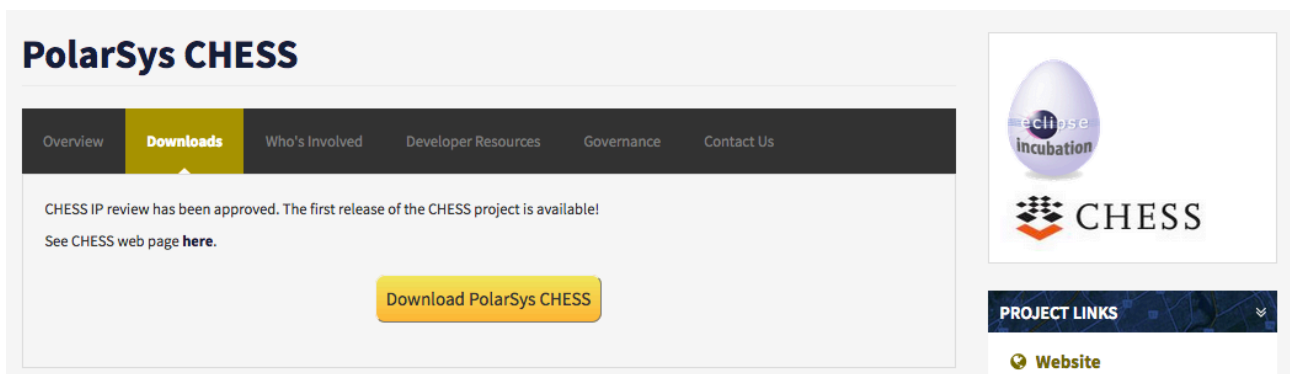


Figure 12. CHESS download page

5.4 The Eclipse Process Framework Project

The Eclipse Process framework (EPF) is one of the historic projects hosted by the Eclipse Foundation. Again, it illustrates the paradigm presented in Figure 1 of an open platform that is extensible to create products. One of the main products based on EPF is Rational Method Composer (RMC).

As EPF is a very mature project, there is not so much activity on it, but as the project fulfils a large part of what AMASS needs in this domain, the AMASS consortium studies the opportunity to step up in the EPF project and potentially proceed to the development or adaptation of EPF to fit better in the AMASS Open Platform Architecture. One of the main actions to be done is the migration of EPF to the current version of Eclipse so that the project can be seamlessly integrated in the tool chain.

By design, the Eclipse license, and the Eclipse community allow newcomers to become active in the project and to implement such changes. That is another interesting feature of Open Source ecosystems: if you can dedicate the time and skills, you are able to make any evolution needed to the projects.

Concerning EPF in AMASS, the final decision about a potential migration will be taken in the coming months.

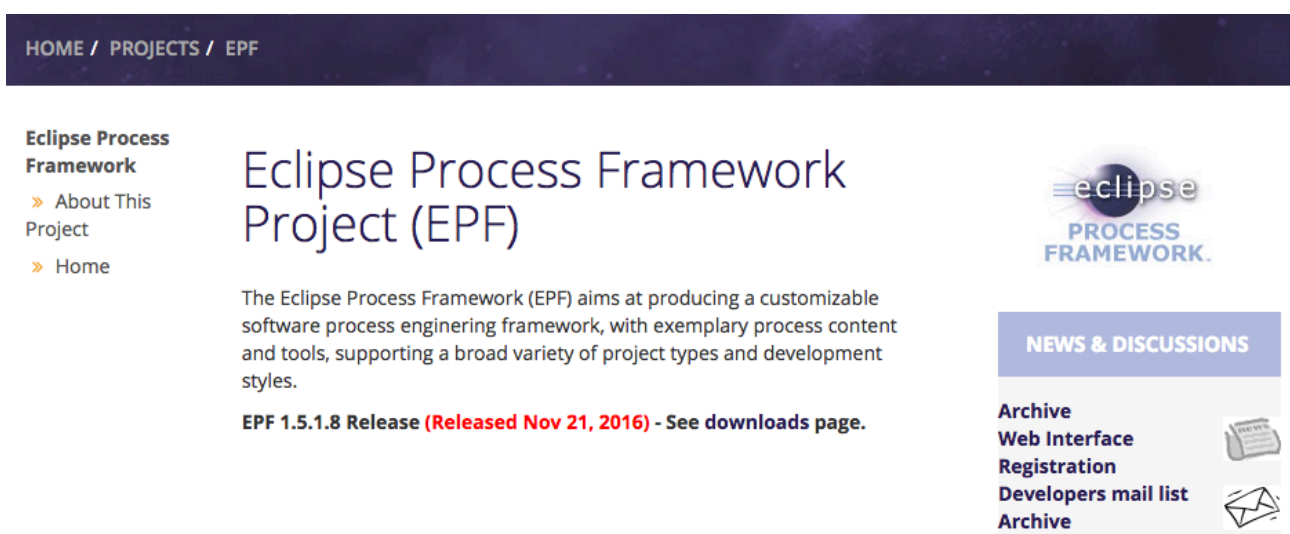


Figure 13. EPF project page

5.5 Next Steps for the AMASS Open Platform

The next steps that are necessary to create a sustainable open source project for the AMASS platform are shown in Table 1 and described in the following sections.

Table 1. Next steps for the AMASS Open Platform

Step	Estimated Date
Integrate the contributions of the AMASS core prototype	Along 2017
Elect new committers	Mid 2017
Releases and release reviews	In sync with the second AMASS prototype

5.5.1 Integrate the contributions of the AMASS core prototype

AMASS has reached its full speed, and a lot of partners from AMASS plan to contribute code to OpenCert and CHESS as the AMASS open source platform.

The next most important step will be to integrate those new contributions in the current code base, and this should be done earlier rather than later. Indeed, in order to be successful, an open source project must work in the public and with transparency so that it attracts users, adopters and contributors.

5.5.2 Elect new committers

Open Source projects are meritocracy. In order to become a committer, a new contributor must be elected by the existing committers. This election must take into account previous contributions that demonstrate the capability of the contributor to understand the project architecture and to contribute useful code to the project.

After the integration of the contributions of the AMASS core prototype to OpenCert and CHESS, it will be possible to elect as new committers the developers of those contributions.

5.5.3 Releases and release reviews

The CHESS team has already done several releases, but for the OpenCert team, it will be the time to plan for the first one.

Releases are formal for every Eclipse project. They start with planning and end with a community review. We can capture as many future releases as we would like. It is common practice to specify releases three to six months in advance.

Releases are broadly categorized as:

- Major releases include API changes (potential for downstream breakage);
- Minor releases add new functionality, but are API compatible with previous versions; and
- Service releases include bug fixes only and include no significant new functionality.

For all major and minor releases, the project must engage in a **release review**. Note: Release reviews are not required for bug-fix/service releases.

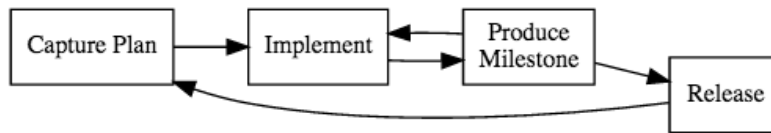


Figure 14. Release Iteration process

A project plan (alias Development Plan) is required for each major and minor project release. The plan should lay-out in broad terms what the goals are for the release. As plans are a valuable means for the community to get involved with open source projects, the plan should be created at the beginning of the release cycle. Establishing the plan early gives prospective contributors help in determining how they can most usefully contribute, and adopters can prepare their own development schedule and themes. Plans can change during the release cycle.

6. Conclusions

Deliverable D7.3 presents the report about open source platform project creation in the AMASS project.

First, we gave a short introduction to Open Source principles. Then, we presented thoroughly the Eclipse Foundation and the PolarSys Working Group, that are good examples of Business Friendly Open Source ecosystems, PolarSys being devoted to open source tools for the development of embedded systems.

We also spent several sections describing the Eclipse Development process and the Eclipse IP due diligence process that are the two pillars of this business friendly open source approach as they ensure that the code hosted by Eclipse and the Eclipse Working Groups can be reused in different contexts (other open source projects, proprietary products, service offers, ...) without taking legal risks.

From the beginning, the AMASS consortium decided to publish a part of the results of the project in open source in order to make adoption easier, and to establish a de-facto standard. The end of the first year of AMASS, and the second year of the project are the moment when the AMASS consortium are turning a large part of the development in public by directly contributing to AMASS Open Platform composed of OpenCert, CHESS and EPF for new developments when they can be licensed in Open Source.

AMASS is a unique opportunity for the OpenCert and CHESS contributors to create traction about their projects by showing activity, by demonstrating that the platform implements more and more features along the three years of AMASS, and by using the open source part of the platform as a widely accessible demonstrator.

By working in the context of PolarSys and by adopting the best practices of the Eclipse platform and Eclipse Modelling, AMASS partners expect that the project solutions will evolve in pace with the more challenging requirements of modern engineering teams and will provide more flexible extensibility and customization that makes it easier to adopt the tools to the methods and processes of industrial engineering teams.

In deliverable D7.4 "AMASS open source platform marketing and outreach plan", we will describe the actions planned to promote AMASS Open Source platform (OpenCert, Chess and EPF) inside the Eclipse and PolarSys ecosystem, as well as how we will leverage open source to promote AMASS results to the worldwide communities of Systems Engineering and Embedded Systems.

Abbreviations and Definitions

Acronym	Meaning
API	Application Programming Interface
ASAM	Association for Standardisation of Automation and Measuring Systems
AUTOSAR	AUTomotive Open System ARchitecture
CBI	Common Build Infrastructure
CI	Continuous integration
CPR	Common Pool Resources
CPS	Cyber Physical System
EMO	Eclipse Management Organization
EPF	Eclipse Process Framework
EPL	Eclipse Public License
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IMA	Integrated Modular Avionics
IoT	Internet Of Things
IP	Intellectual Property
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
LTS	Long Term Support
MQTT	MQ Telemetry Transport
ODS	Open Data Services
OpenMDM	Open Measurement Data Management
OSS	Open Source Software
PHP	Pre Hypertext -processor
PMC	Project Management Committee
RTOS	Real-time operating system
SME	Small and Medium Enterprise
STO	Scientific and Technical Objective
VLTS	Very Long Term Support
WG	Working Group

References

- [1] OPENCROSS project <http://www.opencross-project.eu/>
- [2] CHESS project <http://www.chess-project.org/>
- [3] SafeCer project <http://www.safecer.eu/>
- [4] OpenCert proposal <https://www.polarsys.org/proposals/opencert>
- [5] Eclipse Process Framework Project (EPF) <https://eclipse.org/epf/>
- [6] PolarSys – Open Source Solutions for Embedded Systems <https://www.polarsys.org/>
- [7] See Elinor Ostrom Wikipedia page: https://en.wikipedia.org/wiki/Elinor_Ostrom
- [8] Governing the Commons: The Evolution of Institutions for Collective Action by Elinor Ostrom, 1991
- [9] The Open Source Initiative WebSite is available at <https://opensource.org>)
- [10] Annotated version of the definition of Open Source Software: <https://opensource.org/osd-annotated>
- [11] Common Build Infrastructure (CBI): <http://wiki.eclipse.org/CBI>
- [12] The Eclipse Intellectual IP Policy: http://www.eclipse.org/org/documents/Eclipse_IP_Policy.pdf
- [13] The Eclipse due diligence process: <http://www.eclipse.org/legal/EclipseLegalProcessPoster.pdf>
- [14] A more digest version of the Eclipse IP Process in cartoons:
http://www.eclipse.org/projects/dev_process/ip-process-in-cartoons.php
- [15] The Eclipse Trademark Transfer Agreement: http://www.eclipse.org/legal/Trademark_Transfer_Agreement.pdf
- [16] OpenCert initial contribution tracker: https://dev.eclipse.org/ipzilla/show_bug.cgi?id=11104
- [17] Parallel IP Process: http://wiki.eclipse.org/Development_Resources/HOWTO/Parallel_IP_Process
- [18] CHESS Project proposal: <http://www.eclipse.org/proposals/polarsys.chess>

Appendix A: OpenCert Proposal

We provide more details about the OpenCert project creation process in these appendixes. Similar details are easily available for CHES on the Eclipse servers.

See <https://www.polarsys.org/proposals/opencert>

This proposal has been approved and the **OpenCert** project has been created. ✕

OpenCert

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.

Parent Project:
PolarSys

Background:

This project proposal will issue the source code resulting from the OPENCROSS project (www.opencross-project.eu). The OPENCROSS project developed a common certification tool framework that spans different vertical markets for railway, avionics and automotive industries.

The OPENCROSS plan was always to make project results publicly available, and open beyond the project's life. The OPENCROSS consortium implemented a fully-fledged version of the OPENCROSS tool platform during the project and has defined a plan for its further development and maintenance to an industrial open-source community. After negotiations with Polarsys, OPENCROSS partners decided to use Polarsys as the open-source community. Code contributors decided to use the name "OpenCert" for the publicly available and open-source code resulting from the OPENCROSS project. For further details see [here](#).

Scope:

OpenCert is a product and process assurance/certification management tool to support the compliance assessment and certification of safety-critical systems in sectors such as aerospace, railway and automotive. The main tool features include:

Knowledge Management

- Capture information from Standards (e.g. safety functional standards such as ISO 26262, IEC 61508, etc.).
- Specify company specific Processes or Rules.
- Map Knowledge from different Standards to know better their equivalences and reuse opportunities.

Assurance Project Management

- Create Safety Assurance Project.
- Define Safety Assurance Project Baseline.
- Define access permission for users.

Argumentation Management

- Define modular assurance structure.
- Develop claims and links to evidence.
- Specify argumentation module assumptions.
- Validate argumentation module assumptions.

Evidence Management

- Determine the evidence to provide.
- Collect and characterise evidence items information.
- Specify traceability between evidence items.
- Perform evidence change impact analysis.

Process Management

- Check process compliance against Standards (e.g. functional safety standards)
- Measure and estimate safety metrics.
- Specify traceability between process items.

Description:

OpenCert is a customizable safety assurance and certification tool environment integrated into existing manufacturers' development and safety assurance processes and tooling. The OpenCert tools support the following activities of safety-critical product development:

1. **Standards & Regulations Information Management:** This activity group supports knowledge management about standards (e.g. DO178C, ISO26262, EN 50128/50126/50129, etc.), regulations and interpretations, in a form that can be stored, retrieved, categorized, associated, searched and browsed.
2. **Assurance "Project" Management:** This is the core set of functionalities concerned with the development of assurance cases, evidence management, assurance process management, and global monitoring of the compliance with standards and regulations. The most relevant services of the OpenCert tools are to provide functionality that supports guidance and re-use of assurance artefacts. In addition, these services offer an evolutionary and transparent product and process assurance and certification with the ability to automate the most labour-intensive activities (e.g., traceability, compliance checking, assurance process planning, and metrics management, among others), as well as providing facilities to integrate the engineering activities with the certification activities from early stages.
3. **Compliance Management:** The OpenCert tools help "engineers" to assess where they are with respect to their duties to conform to safety practices and standards, and still to motivate them to see the effective progress of the work and level of compliance.
4. **Modular and Incremental Certification:** OpenCert supports a modular safety assurance and certification approach to enable cost-effective reuse of pre-qualified building blocks in different contexts (e.g., systems, configurations, upgrades).

Why Here?:

The innovation required to advance OpenCert tools needs to be driven by the key industrial companies. Polarsys Working Group is a perfect environment for open innovation and industrial feedback.

More concretely, there is a shared set of specific goals between Polarsys and OpenCert, which motivates us to join Polarsys:

- **Open Innovation:** Ensuring the highest levels of productivity, reliability, safety, service, and performance implies a continuous effort of research and development in software tools.
- **Computer Assistance and Automation:** The numerous and complex operations required to develop and maintain industrial systems imply a high level of automation based on software tools.
- **Certification (e.g. DO178C, ISO26262, EN 50128/50126/50129, etc.):** The development of safety-critical and embedded systems must comply with strict regulations impacting both the final product and the development process and tools used to build them.
- **Very Long Term Support:** The tool chain needs to remain operational for the life cycle of the products; many industries need more than 10 years, and some need up to 80 years.

By joining Polarsys, we expect that OpenCert will evolve in pace with the more challenging requirements of modern engineering teams and will provide more flexible extensibility and customization that makes it easier to adopt the tools to the methods and processes of industrial engineering teams. Other initiatives, such as OpenECTS (<http://www.openetcs.org/>) working on the safety-critical railway domain and with plans to join Polarsys, will also create new opportunities to evolve OpenCert.

Licenses:

Eclipse Public License 1.0

Legal Issues:

We use a new name for the project (instead of using OPENCROSS), so that there is no conflict with European Commission.

It has been decided among the OPENCROSS partners that a **weak copyleft license** was to be applied to OpenCert, in particular to allow proprietary plugins to be connected to OpenCert. This choice has been made to allow the distribution of OpenCert together with proprietary products integrated with it, such as evidence tools (e.g., Medini Analize).

Icons provider (Icons8) provided icons license for free for open source projects: SVNKit (<http://svnkit.com/licensing.html>). While using SVNKit in Open Source projects is completely free of charge.

Project Scheduling:

Upload of the initial contribution and first build: December/2015

Future Work:

By request from the European Commission (OPENCROSS project officer), in January/2016, the OpenCert project will be presented at HiPEAC Conference, jointly between Tecnia and Polarsys.

The OpenCert code will continue its development and maintenance in the framework of the AMASS project (ECSEL programme). AMASS will start in April 2016 and has a duration of 3 years. Total budget is 20 Million Euro and there are 30 partners from 8 European countries.

During the first year of the AMASS project, we will mature OpenCert tools to TRL4 together with tools from the SafeCer project. This should be released in April 2017.

A plan to mature OpenCert to TRL 5 will be implemented in the context of AMASS until April 2019.

PEOPLE**Project Leads:**

Huascar Espinoza

Committers:

Huáscar Espinoza (Tecnalia)

Angel Lopez (TECNALIA)

Xabier Larrucea (Tecnalia)

Jan Mauersberger (KPIT)

Marc Born (KPIT)

Tim Kelly (University of York)

Mark van den Brand (Eindhoven University of Technology)

Mentors:

Wayne Beaton

Benoit Langlois

Interested Parties:

- KPIT

- University of York

- Eindhoven University of Technology

- Intecs

- Honeywell

- Airbus Helicopters

- Tecnalia

SOURCE CODE**Initial Contribution:**

The OpenCert architecture has some major components:

- CDO server and client
- Eclipse environment with various plugins
- Tomcat for the web services

Below is a non-comprehensive list of the dependencies for OpenCert:

- Eclipse Public License 1.0 licensed dependencies
 - Eclipse
 - EMF
 - EEF
 - GMF
 - Ecoretools
 - Epsilon
 - Xtext
 - CDO
- Apache License 2.0
 - Ant
 - Tomcat
 - Vaadin
 - Spring
- GNU Lesser General Public License v2.1
 - Hibernate
- Tmate Open Source License
 - SVNKit

The other dependencies: PostgreSQL, REST clients are integrated via standard network protocols and are not released with the software platform, and thus do not add constraints on the license of the OpenCert release.

Due to the strong link between the OpenCert architecture and the Eclipse environment, and also because the OpenCert dependencies are compatible with the Eclipse Public License (EPL-1.0), and this license meets the objective of releasing the OpenCert under a free software license with weak copyleft, it has been decided that the platform will use the EPL-1.0 globally, for all core modules.

Hence, will be released under EPL-1.0 the following modules:

- The webapp modules: process manager, evidence manager, reports
- The eclipse editors
- The CDO server

Source Repository Type:

Git

SIGN IN to post comments.

Appendix B: OpenCert Proposal Progress Tracking

See https://bugs.eclipse.org/bugs/show_bug.cgi?id=483200

Wayne Beaton ✓ ECA	2015-11-27 15:11:18 EST	Description
We'll use this bug to track the progress of the proposal. I've opened the proposal for community review.		
Wayne Beaton ✓ ECA	2015-11-27 15:12:58 EST	Comment 1
Mike has expressed concern that the name, OpenCert, is suggestive of digital certificates. Do you share this concern?		
Gael Blondelle ✓ ECA	2015-12-02 10:58:56 EST	Comment 2
Embedded systems people (the target of PolarSys), think certification when they read "cert". So I think that as the project will be hosted by PolarSys, it will be clear that it is about certification and not about certificates.		
Wayne Beaton ✓ ECA	2015-12-02 11:31:16 EST	Comment 3
I still require a mentor. In anticipation of being successful in my search for a mentor, I've scheduled the creation review to conclude on December 16/2015. Please continue to monitor communication channels. Following the creation review, we will initiate the provisioning process. As part of this process, we will bring committers on board. To gain committer status, some paperwork must be completed. The exact nature of that paperwork depends on several factors, including the employment status on the individual and the Eclipse Foundation membership status of the employer. More information: https://www.eclipse.org/projects/handbook/polarsys.html#paperwork If you can be ready with the paperwork in time for the completion of the creation review, then we can move quickly through the provisioning process. When we initiate provisioning, committers will be sent an email with instructions; please don't send any paperwork in until after you receive those instructions.		
Wayne Beaton ✓ ECA	2015-12-16 10:52:26 EST	Comment 4
I declare this review successful! We will initiate the project resources provisioning process shortly. Please tell your project committers to carefully monitor their email for a message from The Eclipse Foundation with instructions for providing committer paperwork [1]. Our IT team cannot allocate project resources until after we have processed the paperwork for at least one committer, so your attention in this matter will keep the process moving forward. Be advised that the paperwork process will time out after 120 days; any committers who are unable to complete their paperwork requirements in this timeframe will have to be elected to the project (your project mentors can provide assistance with this). Immediately following the provisioning process, your next step will be to submit an initial contribution [2] for review by the IP Team. Please do not commit any code to an Eclipse Foundation Git repository until after you receive the IP Team's approval. In anticipation of this step, you may consider ensuring that your code has the required copyright headers and namespace (if applicable). If you have any questions, please send a message to emo@eclipse.org and we will provide assistance. Please encourage all project committers to join the incubation mailing list [3]. We use this list to connect committers from new projects to their peers in other projects in the incubation phase and to mentors who can help answer questions and discuss issues related to the project onboarding process. An overview of the complete project creation process is in the Project Handbook [4]. [1] https://www.eclipse.org/projects/handbook/#paperwork [2] https://www.eclipse.org/projects/handbook/#ip-initial-contribution [3] https://dev.eclipse.org/mailman/listinfo/incubation [4] https://www.eclipse.org/projects/handbook/#starting		