5 Years of "Papyrusing" – Migrating Industrial Development from Proprietary Commercial Tool to Papyrus

Ronan Barrett[1], Francis Bordeleau[2]

[1]Ericsson AB, Stockholm, Sweden
Ronan.barrett@ericsson.com
[2]Ericsson Canada, Town of Mount Royal, Canada
francis.bordeleau@ericsson.com

Abstract. Five years ago, Ericsson decided to investigate the possibility of replacing a proprietary commercial UML modeling tool, used at the core of one of its internal toolchains, with an open source alternative based on Papyrus. The motivation for this switch was manifold, including cost, technology integration and community activity. It was clear from the outset that significant effort would be required to make Papyrus into a tool usable in large-scale industrial projects. This journey took time and dedication from both the Ericsson and the Papyrus development teams, but we have now reached the point where deployment has begun. In this paper we provide an experience report from the last five years.

Keywords. MDE, open source, Eclipse, Papyrus, experience report

1 Introduction

Ericsson relies on software development technologies and tools to maximize developers' productivity and to reduce product development time and cost, and get to market faster. The use of leading-edge technologies and tools are the keys to maximizing the capability to innovate and develop real product differentiators. Model-Driven Engineering (MDE) technology is used at the core of Ericsson development processes to address many different aspects, including system design, software design, network architecture design, and information/data design.

Like most companies, Ericsson's development processes have traditionally been based on proprietary commercial tools. With the emergence of open source software development tools, the possibility to re-engineer processes on open source tooling becomes an attractive alternative. Open source tools offer many key benefits, including cost reduction, elimination of vendor lock-in, and increased agility and ability to get required product features and improvements faster. Ultimately, the use of open source solutions allows to better control our destiny.

However, while the benefits are significant, migrating to open source technologies is not free. It requires financial investment and it requires involvement and commitment to work within the community. It is important to highlight the fact that the required level of involvement/investment differs greatly depending of the maturity of the open source technology. For example, today companies can use open source technologies like Linux [1] and CDT [3] without getting directly involved in the development. However, because no mature open source modeling tool existed when our project started, a major involvement/investment was needed to bring Papyrus and the related ecosystem to the required level of maturity and scalability. Papyrus had never been used before in a large-scale industrial context and Ericsson projects involve hundreds of geographically distributed engineers.

This paper provides an experience report of the involvement of Ericsson in the collaborative development of Papyrus, over the last five years. The paper is structured as follows. Section 2 describes the experience of the Ericsson team involved on the development of a toolchain using Papyrus from a technical perspective. Section 3 discusses the Ericsson experience from a project management perspective. Finally, Section 4 concludes by summarizing the main learnings of the collaboration.

2 Tool Migration Experience

The Papyrus open source journey within Ericsson began five years ago when a new Operations & Maintenance (O&M) toolchain development effort began. This section provides a description of the O&M context, the tool requirements, the journey in migrating to Papyrus and ongoing activities. We also summarize the journey so far and identify lessons learned.

2.1 Operations & Maintenance Modeling

O&M in the context of telecommunications is the operation and management of telecommunication devices by setting and receiving configuring parameters, reading alarms, clearing alarms as well as retrieving performance counters [12]. The models are hierarchies of classes with attributes, relationships and operations. These models can scale up to thousands of elements for complex devices. An initial configuration, which is an instantiation of a subset of these classes, may also be defined.

Of course, the basic UML doesn't provide for the hundreds of, sometimes interrelated, metadata properties required for O&M modeling. These metadata properties are modeled in a UML Profile, and constrained by Object Constraint Language (OCL), which is applied to all O&M models. This profile can be considered an O&M Domain Specific Language (DSL).

The models are the input to model transformations which generate artifacts to drive the O&M interface allowing the operator, or a higher level management system, to manipulate the system using different protocols.

The O&M models are created by software engineers worldwide, who are rarely modeling experts. Each network device is responsible for their own model and often

many engineers collaborate to define a given O&M model. The O&M models themselves are considered to be source code and are stored in a version control system.

2.2 Operations & Maintenance Modeling Tool Requirements

Given the background of our O&M modeling context, here we state a set of tooling assumptions as follows:

- A UML platform is required as our O&M models are object orientated
- Diagrams are needed to communicate models to users
- The cost of building our own UML tool cannot be justified
- The Eclipse platform is desirable due to the tooling ecosystem available there e.g. File management, version control integration, Compare/Merge, etc.
- Open source solutions should be used if possible

The following mandatory requirements must be supported by the modeling platform:

- Class and instance diagram support
- DSL/UML profile support
- Use the Eclipse UML2 API
- Model validation
- Transformation language support
- Collaboration tools (merge/compare)
- Version control system integration (SVN/Git/Clearcase)
- CI integration (headless mode)

The following optional requirements should be supported by the modeling platform to make the tool desirable to users. These requirements were not met by the existing proprietary modeling tool and so can be considered as motivating factors for the migration. The optional requirements are as follows:

- Low or zero license cost per user
- Textual modeling integration
- Simple installation (local and centralized)
- Two way tool platform collaborative development process
- Tool co-location in same Eclipse workspace
- Active research & support community

2.3 The New Operations & Maintenance Tool

In 2010 when the new tool was being developed a state of the art study was performed to see what UML tools and associated ecosystems were available. At this time there were only two UML modeling tools capable of potentially meeting our base requirements. A proprietary modeling tool used in a legacy O&M tool or Papyrus. It was decided early that we would use the proprietary modeling tool until Papyrus was stable.

In the following sections we provide an experience report whilst following the Eclipse release train lifecycle where releases are made once a year with an alphabetically incremented release name. [15]

Eclipse Indigo

The first Papyrus evaluation was done during the Eclipse Indigo release train timeline. An evaluation of Papyrus was performed but it was not in a state, from a feature or usability perspective, to be considered as an alternative to the existing proprietary modeling solution. We decided to get involved with the Papyrus effort and together build a UML tool that could replace the commercial proprietary one in the medium term.

Our first engagement with Papyrus was to sponsor a "statement of work" (SoW). We defined the set of features we wanted implemented so we could use Papyrus. It was a given that stability must also be improved as crashes and model corruption were very common at this stage. The SoW contained requests to provide some of the following features; instance modeling, OCL integration and usability including themes and preferences.Eclipse Juno

The SoW was submitted and the status was assessed against the Juno release train. Little or no communication, back or forth, was had between the time of filing the SoW and the release of the Juno version. Of course, this led to disappointment from the Ericsson side when our features were either not implemented or not implemented in the way we required.

Two fundamental problem areas became clear at this time. Papyrus had an ethos of providing a truly generic UML tool that did not hide any of the complexities of the UML and that all modeling should be done via the diagram. This was done with good intensions but it has a huge usability impact for users who wanted to be productive. It was clear the lack of direct, frequent communication must be resolved if we are to get a tool that meets our requirements in the medium term.

The stability of Papyrus at this time was of great concern. The developers of Papyrus were not having significant problems but we in Ericsson were. It became clear we were doing something very different to them, we were working collaboratively! Papyrus had never been tested in an industrial environment where many engineers work on a model simultaneously and use a version control system to manage the code base. Scaling issues were also noted in that model fragmentation wasn't supported and then when it was it wouldn't work correctly with profiles. It seemed the testing being done with Papyrus was often on small models with no profiles, as profile evolution was not supported properly either.

Eclipse Kepler

The Kepler release was the first version of Papyrus to implement all the items from the original SoW. Unfortunately the OCL integration into Papyrus was completely broken. At this stage other groups in Ericsson were becoming interested in the Papyrus initiative. We joined forces and started writing considerable numbers of Bugzilla

reports to try and get Papyrus up to the stability we required. We categorized the bugs into specific focus areas to enable us to work in a focused manner.

A key feature which had previously not been investigated was also started, Model/Profile Import. This feature would support the import of the models built on the existing propriety UML tool. The objective was that both model and diagram information would both be imported faithfully.

The OCL integration issues that had meant Kepler was unusable in the O&M modeling context would be resolved by Ericsson and Papyrus engineers working directly on the features piece by piece. This collaboration required a large amount of testing of builds and direct communication between the teams.

Eclipse Luna

The Luna Papyrus release came with significant quality improvements. The focused categorization of bugs, and resolution of these bugs, meant specific areas that lacked quality were improved drastically.

Although the Luna release provided a much-needed boost to quality, a significant bug was identified soon after its release. The model/diagram context was lost when users switched between models. This bug was another example of where Papyrus was built without an industrial perspective. The issue did however provide an ideal platform for collaboration where our engineers set out what context needed to be saved before a model switch occurs. Papyrus engineers then implemented the fix and a number of test and feedback cycles between engineers took place.

Unfortunately, from the O&M modeling perspective we were still not in a position to move to Papyrus as the OCL integration was still broken. A series of bugs caused service release after service release in Luna to be unusable. Finally, after the fifth Luna service release we had OCL integration in place and we were ready to launch the O&M tool on Papyrus!

Eclipse Mars and Beyond

Whilst the Mars release has been available for some time we have not yet migrated to it yet. The process of migrating users from the propriety tool to Papyrus is taking place. It is essential that users are met with a robust, thoroughly tested environment to ensure user acceptance. It cannot be underestimated how important a user's first impression of a tool is. Users cannot be used as a testing community for a tool, it is neither cost effective nor practical. The cost of many users hitting the same bug can be very significant and the loss of confidence permanent.

An important consideration of having finally reached the desired level of quality for a tool is at least maintaining that standard and hopefully improving it. The tool must only get better, it cannot get worse, or else users will not upgrade to it. Automated tests and maintaining backwards compatibility are key factors in maintaining the significant investment made in bringing Papyrus to an industrial standard.

3 Project Management Experience

This section discusses the Ericsson experience of direct involvement in Papyrus development from project management perspective.

3.1 The Ericsson Papyrus Open Source Modeling Project

The Ericsson Papyrus Open Source Modeling project involves a group of different Ericsson stakeholders and several suppliers. Stakeholders come from Ericsson teams involved in different products, or product aspects, and have different needs and priorities. Their modeling needs are quite diverse, and include information/data modeling, network architecture modeling, system modeling, and software design modeling. The case described in Section 2 is from one of the key project stakeholders, but many other stakeholders are also involved. The suppliers are external companies providing key technical expertise on the different open source technologies used in the project, including Papyrus [4, 13], Papyrus-RT [5], EMF Compare [7], EGit [8], OCL [9, 14], and GMF Tooling [10]. These companies, which are typically leaders in the Eclipse ecosystem, are essentially responsible for fixing defect and enhancing the different Eclipse open source components.

Over the last 3 years, the total number of people directly involved in the project has varied from about 15 to 40 people. Participants come from different organizations, located in different countries, and have different work cultures. As opposed to the more conventional case where all project members are part of the same organization, in this case, we have to deal with the fact that people are part of different organizations each having their own rules, legal obligations, and working process. To succeed, it is crucial to get everybody aligned on the same overall project objectives and working together in a collaborative manner.

3.2 Agile Project Management

The successful management of a project involving many stakeholders, development teams, and external suppliers, simultaneously working on different project aspects requires the establishment of a project management process that provides sufficient flexibility to allow the different parties involved to work efficiently without unnecessary management overhead while ensuring meeting the overall project objectives in terms of time, cost, and quality. Also, because we are dealing with a set of requirements and priorities that are changing/evolving over time with the needs of the stakeholders, it is fundamental that the project management process has the inherent flexibility to cope with those changes. Besides the technical challenges described in Section 2, our main objectives regarding the definition of the project management process were to provide clear governance for the different aspects of the project, provide transparency regarding technical and management decisions, and ensure that key information is communicated to all of the different parties involved in a clear and timely manner. Another key objective was to adapt and leverage, as much as possible, the existing processes of our internal stakeholders and suppliers.

In the current case, as there were no off-the-shelf process model we could use for the management of this open source project, we worked together with the different stakeholders and suppliers to put in place an agile process that fits our needs. The goal was to start with something simple and efficient whilst improving the process as the project evolved. Thanks to the constructive collaborative environment we have been able to establish, we have been able to make continuous improvements, and we are continuing to do so.

3.3 New Way of Working

Working with open source technologies is significantly different than working with traditional proprietary commercial technologies in which customers buy licenses for products developed and owned by commercial vendors. One of the key differences is that the relationship with suppliers is no longer a conventional customer/vendor relationship, but a peer-to-peer relationship in which users and suppliers need to be aligned on the same overall objectives and work together to achieve success. Such a development context allows users and suppliers to work closely together and both contribute, with their respective expertise, to the development of the end solution. This has the great advantage of getting people from both sides working as a single team to ensure that the resulting product meets the user needs.

This way-of-working can only be successful if respect, trust and openness exist between the different parties. It is key to recognize that different people contribute in different manners to the overall project. Typically, engineers from the supplier side provide the technical expertise with the open source components, which is essentially why they are selected for the project, while engineers on the user side provide expertise on the industrial usage of the technologies. However, in many cases, like ours, several people on the user side have been closely involved in Eclipse projects for many years and so they can directly contribute to the technical development of the solution.

3.4 Best Practices

During the last years of involvement in the Papyrus project, we have identified several practices that we consider key in the success of an open source project.

When working in open source feature requests should be generic. It is important that engineers involved, especially the users, think in terms of the global user community and not only of their own specific needs. The open source technologies/components shall be developed to serve different use cases and be usable in different development contexts. If customization is required for a specific context, then the users shall be responsible for doing the customization locally.

One of the main contributions that users can make is to provide concrete use cases. Open source committers often don't have a very good understanding of industrial use cases. By contributing concrete use cases, users can help committers better understand the different usage contexts and make the technology better addressing some of the

key industrial development issues. This contributes to save development time, cost and get better technology faster.

To ensure clear understanding of user needs, direct communication between users and committers, using simple mechanisms like screen sharing sessions and phone calls, is much more effective that writing long emails and documents. In this project, for each new work item (defect or enhancement), we identified two engineers, one on each side, who are responsible for working together to develop a solution. Typically, the Ericsson engineer is responsible for providing a description of the use case, providing required clarifications, and for testing and validating the solution as it is developed. The Papyrus engineer is responsible for implementing the solution taking into account the user needs and the different aspects of the overall open source project. Our experience is that the closer these persons work together, the faster the solution is developed and the better it is.

Bugzilla is used to report all technical issues, defects and enhancements, and to follow progress on reported issues. When reporting issues, users are asked to provide repeatable test case descriptions, with associated models when appropriate, so that the developers can reproduce the problem and develop an appropriate solution. It is very important that all technical communications, unless proprietary information is involved, are done using Bugzilla to make the work visible to the global Eclipse community and to avoid doing the work in a parallel Ericsson project community.

4 Summary

Here we have reported on the Ericsson experience in the development of Papyrus, over the last five years, to provide an open source alternative to a proprietary commercial UML modeling tool. The paper discussed the main motivation for why we wanted to move to Papyrus and how we worked together with the Papyrus development teams to achieve the required key milestones, both from a technical perspective and project management perspective.

The importance of industry contributing to open source projects cannot be underestimated. By providing sets of use-cases and testing time, as well as of course contributing code directly, the quality of open source projects can be greatly enhanced. In the context of our collaboration with Papyrus and the associated ecosystem, we filed hundreds of bugs and worked directly with the committers.

Our migration to Papyrus was not helped by previous false starts, unrelated to our effort, in which Papyrus had been evaluated internally and had fallen short of the required quality. It is an important lesson to learn for any software development effort, manage the expectation of your users as you might only get one chance to sell it to them. Papyrus could have mitigated the problems by being clear what features worked in what releases and in what contexts.

Papyrus is part of an ecosystem where many components collaborate to provide a modeling environment. A pain point in our tool migration was the integration between Papyrus and the Eclipse OCL project. As the OCL support comes from a supplier it became apparent the supplier and integrator needed to work closely together to ensure

a seamless solution. This supplier and integrator relationship is very common in open source modeling contexts and should be encouraged but managed carefully.

It is clear from the Papyrus Luna release experience that we had reached a key milestone. At this key point, it is essential that suppliers are willing to be flexible in making extra service releases to fix critical bugs. We must have a good stable version of the modeling platform and migrate our users to this version before we can consider moving onto the next release, in this case Mars. Of course once the migration is complete and users are happy we must then quickly get back on the release train.

From a project management perspective, we have defined and implemented an agile project management process adapted to our project environment in which many internal stakeholders and external suppliers are involved. This process is continuously improved to cope with emerging issues and reduce unnecessary overhead.

Working in an open source environment has led us to adapt a new way-of-working based on a peer-to-peer relationship with our external suppliers as oppose to the conventional client/vendor relationship. Our experience so far is that this model can be very successful when respect, trust and openness is built between the different parties, and when everybody work together towards the same global objectives. The experience of the last years of work in the Papyrus Open Source Modeling project has allowed us to assemble a set of best practices that we are now using as part of our process.

In conclusion, after five years of involvement in Papyrus development, our position is that Papyrus and open source Eclipse modeling technologies are a main alternative to existing proprietary commercial solutions, but success can only be achieved with proper involvement. This result allows us to envision the future with great enthusiasm.

5 Acknowledgments

We would like to thank CEA and other suppliers, Eclipse project committers, and the development teams that have been involved in the project over the last five years for their technical contributions and the fruitful collaboration context they have establish in this project.

References

1. Linux, https://www.linux.com
2. Eclipse Foundation, https://www.eclipse.org
3. Eclipse, Eclipse CDT, https://eclipse.org/cdt/
4. Eclipse, Eclipse Papyrus, http://www.eclipse.org/papyrus/
5. Eclipse, Eclipse Papyrus for Real Time (Papyrus-RT), https://projects.eclipse.org/projects/modeling.papyrus-rt
6. Eclipse, Eclipse EMF, http://www.eclipse.org/modeling/emf/
7. Eclipse, Eclipse EMF Compare project, http://www.eclipse.org/emf/compare/
8. Eclipse, Eclipse EGit, http://www.eclipse.org/egit

9. Eclipse, Eclipse OCL https://projects.eclipse.org/projects/modeling.mdt.ocl
10. Eclipse, Eclipse GMF Tooling, https://www.eclipse.org/gmf-tooling/
11. Grandite, Open ModelSphere, http://www.modelsphere.org
12. Andersson, L. et al. Guidelines for the Use of the "OAM" Acronym in the IETF. https://tools.ietf.org/html/rfc6291
13. S. Gérard, C. Dumoulin, P. Tessier, B. Selic: Papyrus: A UML2 Tool for Domain-Specific Language Modeling. In H. Giese, G. Karsai, E. Lee, B. Rumpe, B. Schätz: Model-Based Engineering of Embedded Real-Time Systems International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers. Springer, 2010
14. Object Constraint Language (OCL), Version 2.4. http://www.omg.org/spec/OCL/2.4/
15. Eclipse Simultaneous Release. https://wiki.eclipse.org/Simultaneous_Release