

Open Source in the Software Product Line: An Inevitable Trajectory?

Pär J Ågerfalk¹, Brian Fitzgerald¹, Brian Lings², Björn Lundell^{2,1},
Liam O'Brien¹, and Steffen Thiel¹

¹ Lero – The Irish Software Engineering Research Centre, University of Limerick, Ireland.

² School of Humanities and Informatics, University of Skövde, Sweden.

1. Introduction

The open source software (OSS) landscape has changed dramatically in recent years. While OSS and its Free Software antecedent were largely driven by ideology and individual commitment, the main driving force of OSS today is commercialization and opportunities for inter-organizational collaboration (Fitzgerald, 2006). OSS is no longer primarily about enthusiasts contributing to SourceForge projects but increasingly about commercial organizations developing software in “co-competitive ecosystems” (Ågerfalk et al., 2006), and many companies are now actively involved in Open Source (Lundell et al., 2006).

Commercial involvement in OSS projects is often based on a dual licensing model. In such cases, a free version of a product is typically released under an OSS licence while a possibly more advanced version with support and other bundled value adding services is released under a proprietary licence (IONA’s Celtix/Artix and MySQL are good examples). Another approach is to limit OSS engagement to development and maintenance of “commodity” software components, while developing business critical components under a proprietary licence to build on top of the OSS foundation. The latter approach is particularly interesting as it allows for organizations to collaborate on a common core and focus attention on value adding activities. This way, companies can share the cost of developing and maintaining common commodity components while still compete in the marketplace with respect to the complete end-product. In many embedded software domains, such as automotive and medical devices, this is a viable approach since software, although critical, is only a component in a larger mechanical or mechatronic system (Cosi, 2006).

Operating in a product oriented context, many organizations in these embedded software domains have successfully employed software product line (SPL) technology. With this backdrop, the aim of this paper is to pinpoint some of the issues in using OSS in software product lines by raising a set of questions as input to the sketching of an initial research agenda in this area. As an illustration, we will use a brief example from Certus Technology, a UK based company specializing in database solutions for, amongst others, the biological and medical domains.

2. Software Product Lines and OSS

The basic philosophy of SPL engineering is intra-organizational reuse through the explicitly planned exploitation of similarities between related products. This philosophy has been adopted by a wide variety of organizations and has proved to yield remarkable improvements in productivity, time-to-market, cost reduction, and product quality (Clements and Northrop, 2001).

A SPL is set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way (Clements and Northrop, 2001). SPLs are typically developed in a staged process: a domain engineering and concurrent application engineering processes. During domain engineering a reusable platform is established. The platform consists of artefacts such as the

requirements specification, architecture documentation, design specification, implementation, and test cases. Domain engineering defines the commonality and variability between products in the product line and includes the implementation of an adequate product line architecture from which the commonalities can be exploited economically while retaining the ability to vary the products. In application engineering the product line products are then derived from the platform. Application engineering makes use of the pre-planned product line variability and ensures the correct binding of variation points to the specific needs of the customer products (Pohl, Böckle and van der Linden, 2005).

Nonetheless, introducing OSS components as core input assets in an SPL is far from straightforward. For example, even with “commercially friendly” OSS licences (such as LGPL, the “Library/Lesser GNU Public Licence”) available, how OSS components are demarcated and integrated in a product depends no longer only on traditional software engineering criteria but also on legal issues. The inclusion of an OSS component may, for example, require that the complete product be released under the same OSS licence. Also, SPL is often associated with model-based development while OSS components are often not well documented and reuse typically happens on a source-code level.

3. Brief Case Example

Certus Technology Associates is a small company specialising in the development of biomedical information systems, and technical and business IT (Pumphrey, 2006). It uses an SPL business model and an agile MDD development model. One product line supports QA in laboratories offering genetic testing for diseases. There are many such schemes around Europe, all with significant overlap in core functional requirements but all with customer-specific requirements also. One advantage of SPL is that a customer can perceive the advantage of sharing core functionality, as continual enhancement occurs because of the widened customer base.

Consistent with the agile philosophy, the company’s development is split over two sites in the North and South of England, for close-to-customer operation. Its tool base is largely OS, with all development done in a Linux environment. Core tools and technologies include CVS, AndroMDA, JBoss, PostgreSQL, Maven, and Forrest.

Critical to the company’s development model is the utilization of existing components wherever possible, and this increasingly means that OSS infrastructure solutions are looked at competitively with in-house solutions. All components are wrapped in a service layer, which is the target of the MDD transforms developed within the company. This degree of isolation future-proofs development investment, not only offering platform independence, but also reducing the potential for component lock-in. This is considered important whether the wrapped component is OSS or internal.

Choosing an OSS tool or component requires careful research, of product licensing and the quality of both the product and of the community developing it. The profusion of licensing agreements potentially makes this more of a problem than perhaps is necessary. However, Certus has not encountered any serious licensing issues to date. In fact, most of its investment is in tailoring its MDD infrastructure (so-called agile tools) and in customer services other than code delivery. Hence, opening up the code of a developed system would not necessarily be an issue; in fact, in some of its international collaborations with research groups it has offered to do just that.

For most commodity components there are still many competitors in the OS market place even after considering licensing, but it has been found from experience that these can quickly be reduced to a handful with fairly crude quality criteria. The support of a significant organization is one such criterion, and no dependence on other components which do not meet the criteria. Further, the use of an OSS component is clearly facilitated by its adherence to open, or at least transparent standards, so other criteria are based around standards. Adherence to relevant open data standards are of particular importance.

One open issue is whether, as OS components are incorporated, different architectural models can be easily accommodated. Clearly, with MDD it is a relatively orthogonal task to change architectures – particularly as a service model is used already to incorporate components. However, more formal architectural descriptions would protect some of the investment in developing transforms; and clearer OSS architectures are major advantages, as witnessed through the Eclipse project.

4. Open Research Questions

4.1 Licence forms and engineering vs legal decisions.

Several licence/legal questions arise in the context of using OSS components under a LGPL:

- What are the legal issues in using LGPL software components in commercial product lines?
- What would the situation be for the case of an LGPL component that is a core asset? Would all products developed using this core asset have to be under an LGPL licence?
- What would the situation be if the LGPL OSS component was a product-specific component? Would this limit the need for an LGPL licence to just that product?

4.2 Model-based development vs “code is king”

Several questions arise related to use of OSS components in product line development:

- What are the implications of using components within product line development where only the source code is available?
- What happens in cases where there is little or no documentation or high-level models of the components in existence?
- If product line engineering is based on models is there a need to develop models of OSS components and if so who develops and verifies the models?
- What control does an organization have over OSS components that they use in their product line development? Who is responsible for controlling changes and updates to OSS components?

4.3 Benefits of OSS components use in SPL:

There are some benefits to applying OSS in the context of product line development. OSS promotes open standards and spread of reference architectures, which could be important to facilitate software product lines. There are several questions that have to be addressed to fully understand the implications of using OSS:

- What areas within the software product line community would benefit from the use of OSS approaches?
- Would OSS-based software product line frameworks which allow people to build products specific to their needs be useful?

4.4 Issues in using OSS in SPL:

There are many issues related to quality requirements and trust that have to be addressed:

- What are the implications of using OSS components in the automotive domain and similar domains that have high safety, security and reliability requirements?
- Who would guarantee these requirements for OSS components? What level of analysis and testing is required to ensure that the components meet the requirements?
- Are organizations going to trust OSS components that are developed outside of their organizations? Currently trust is built up between organizations over many years of working together. If components can be developed and added to by anyone then with whom does an organization build trust? Who has ownership of the OSS components?
- Are there any organizations currently using OSS components in the development of business critical applications that have these requirements? What has their experience been in doing this and what lessons can be learned?

5. Conclusions

While there is potential for the use of OSS in software product lines there are many questions and issues that have to be addressed. Currently little research is being undertaken in this area, but if the OSS and SPL trajectories are to meet then many of these issues will have to be addressed.

References

- Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Ó Conchúir, O. (2006) Open-Sourcing as Offshore Outsourcing Strategy, Proceedings of the 29th Information Systems Research Seminar in Scandinavia (IRIS 2006), Helsingør, Denmark, 12–15 August 2006.
- Clements, P., Northrop, L. (2001) *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2001.
- Cosi (2006), ITEA COSI Project, www.itea-cosi.org, accessed 19 May 2006.
- Fitzgerald, B. (2006) The Transformation of Open Source Software, *MIS Quarterly*, Vol. 30, No. 3.
- Lundell, B., Lings, B. and Lindqvist, E. (2006) Perceptions and Uptake of Open Source in Swedish Organisations, In *The Second International Conference on Open Source Systems*, 8-10 June, Springer (to appear).
- Meyer, M. H., Lehnerd, A. P. (1997) *The Power of Product Platforms*. New York, NY: Free Press, 1997.
- Pohl, K., Böckle, G., van der Linden, F. (2005) *Software Product Line Engineering: Foundations, Principles, and Techniques*, 1st ed. New York, NY: Springer, 2005.
- Pumphrey, R. (2006) Distributed model-driven development in a small company using Open Source tools, In *An International Research and Practice Workshop: Distributed Development, Open Source & Industry – Opportunities, Suspicions & Strategies*, International CALIBRE Workshop, 3rd March 2006, University of Skövde, Skövde, Sweden.