The Experience of OSCAR

Cornelia Boldyreff, David Nutter, and Stephen Rank
{cboldyreff,dnutter,srank}@lincoln.ac.uk
University Of Lincoln, UK

Abstract

Evolutionary development of a large software component by a small team within a larger research project has many problems in common with industrial software development as well as giving rise to its own unique problems.

We reflect on these problems based on our experience developing OSCAR within the GENESIS project. Key issues are identified and possible ways to overcome or ameliorate these problems are suggested.

1 Introduction

Software research projects naturally involve significant software development and thus experience many of the problems encountered by industrial software development. The uncertain nature of research compounds these problems and introduce others including short timescales (1–3 years), high staff turnover, chaotic development practices, and changing research goals. The Open Source Component Artefact Repository (OSCAR) is the result of development undertaken in a two-year research project (GENESIS) involving academic and industrial partners. GENESIS gave rise to a number of these problems.

2 Experiences

Failings ocurred in all phases of the development of OSCAR, particularly in collaboration between consortium partners [1]. Firstly requirements failings ocurred, primarily *over-ambition*; the research goals of OSCAR and the industrial partners' requirements were difficult to achieve and evaluate successfully in a short timescale. These ambitious requirements resulted in too many features in the initial design. Many of these features were unnecessary for a proof-of-concept prototype and not required by partners.

Secondly, a lack of agreement on core technology hampered the project. In OSCAR's case, an initially promising database (PostgreSQL on Linux) caused problems when OSCAR was deployed on Windows. The maintenance cost to solve these problems was significant. The fact that the consortium was using two different development environments (Linux and Windows) initially hid these problems.

A lack of communication, particularly between partners, caused problems. Attempts to integrate OSCAR with other GENESIS components or re-use it in other projects were hampered by lack of user support. However, communication needs to be bi-directional; research developers using OSCAR often failed to supply informative bug reports.

Finally, disseminating the work beyond papers is important. Developing an information pack on CD containing the current software, papers and any flyers to distribute at conferences and industrial gatherings is a good strategy that should not be confined to the closing stages of the project. This would foster wider participation as well as aiding internal collaboration.

3 Conclusion

Early agreement on issues such as collaboration and communication support, common platforms, component selection, tool support and the research/development tradeoff is important. During planning, effective risk management including contingency planning and expectation management is vital. Maintaining a balance between the research goals of the project and the needs of the industrial partners is difficult. In our case, due to our funding conditions, the correct decision was to solve the partner's problems first. While problems within a small research team can be readily recognised and addressed, collaboration support, especially communication of work status, between teams is vital for an effective distributed software research and development project.

References

[1] D. Nutter, C. Boldyreff, and S. Rank. Communication and conflict issues in collaborative software research projects. In *Proc. of the 4th Workshop on Open Source Software Engineering*, Edinburgh, May 2004. IEEE.