The Experience Of OSCAR

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

Introduction

The problems encountered during the development of the Open Source Component Artefact Repository (OSCAR)[3] as a component of the GENESIS[4] platform are relevant to distributed development of academic research prototypes[2] especially where research products are to be reused in further projects. GENESIS is an Open Source software engineering environment that provides process, resource and software artefact management support to distributed software teams.

Background

OSCAR was designed at the University of Durham to store large collections of software artefacts, either generated by the other GENESIS components or by developers accessing OSCAR directly. It provided services to index (with a Self Organising Map[1]), version control and annotate artefacts with extensible metadata.
Problems were encountered in all phases of the project, leading to slowly increasing delay. Risk-management resulted in a reduction of functionality in the final OSCAR deliverables including the removal of dependability and promised plug-in features.

Project's Vital Statistics

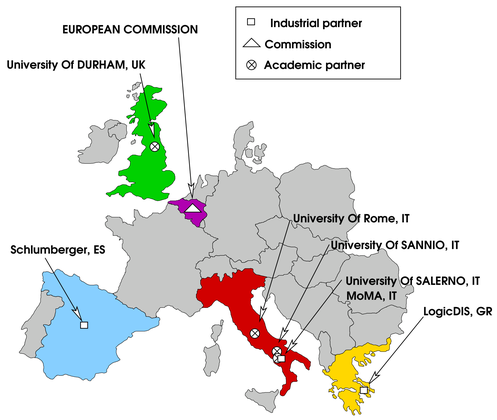
The table below compares the rest of the GENESIS project with OSCAR. The OSCAR code is obviously the largest single component of the system.

GENESIS project (excluding OSCAR)	
Total LoC	119,846
Mean class size	239
No. of classes	501
No. of packages	69
Of which subpackages	60
No. of support files	683
OSCAR sub-project	
Total LoC	53,718
Mean class size	191
No. of classes	281
No. of packages	30
Of which subpackages	19
No. of support files	665

OSCAR was the largest single component in the system, unsurprising given its important role.

Consortium Map

This shows the location across Europe of the various consortium partners. There is a large concentration of partners in Italy.



Poor Collaboration

Agree a process for software release and information distribution. Stick to it!
Additionally, the choice of tools and platform are important when arranging collaboration. In particular, a centralised repository of all project data should be established at project commencement, rather than relying on per-component or per-developer repositories.

Undocumented Meetings

Partners who are unable to attend meetings should be kept fully informed.
This is especially important if decisions impacting the work of the absent partners are taken at that meeting. Full minutes should be produced and circulated of all such meetings, with the important decisions highlighted. Unbalanced geographic distribution can exacerbate collaboration and communication problems.

Short Timescale

Concentrate on effective risk management
Due to lack of recovery time after mistakes, a risk management plan must be prepared for a variety of unfavourable situations. Our usual response to a bad risk was to discard that functionality in favour of something more simple.

Overambition

Manage expectations: promise little and over-deliver if you can.
Obviously this must be balanced with the need to do novel research, if that is a goal of the project. A simplistic project may not be impressive from a research perspective.

Poor component choice

Take care that selected components satisfy all stakeholders' needs before proceeding
This was our greatest design failing in terms of time wasted, so be careful! The risk management plan should address situations where components need to be abandoned.

Balancing Needs

Solve partner's problems first. Any blue-sky research is a bonus.
Much research including GENESIS is funded on condition that the academic partners collaborate closely with their industrial counterparts to solve a real problem. It is therefore churlish to use the time to do purely speculative research. A process, not necessarily formal, for exchanging software and management information between sites is necessary. For example, academic partners should address the installation support needs of industrial partners.

Documentation

Balancing the need for documentation with ongoing work is hard. Impress upon collaborators the need to be adventurous. Provide examples!
While examples are vital for researchers learning the new software, complete reference guides and manuals are not. Additionally, examples are quick to write whereas comprehensive manuals are not.

Dissemination

An "information pack" with lots of documents and if possible software is an invaluable aid when talking to curious potential users
Industrial dissemination is somewhat different, but for academics collected papers and source code on CD is a convenient format for distribution at conferences.

Bugtracking

Agree the information required in a useful bug report and train researchers developing with the system to be active participants in maintenance where possible.
For a lightly staffed research project, merely telling the developers "This does not work" is insufficient. Completing bug reports is necessary although the time developers can spend tracking down problems is necessarily limited.

Conclusions

Despite the problems above the final release of OSCAR was usable, though feature-reduced. The problem areas discussed have been noted and will be used to redirect research development to avoid them in the future.
The issues peculiar to the GENESIS project:

- Geography exacerbated communication & collaboration problems
- Failure to recognise poor quality of a key component

Issues common to all academic projects, on which agreement should be reached early in the project:

- Common platform, component selection and appropriate tool support
- Balancing research and software development
- Dealing with collaboration issues
- High staff turnover
- Documentation and data organisation

The short-term nature of modern research projects appears to be the factor exacerbating all these problems as there is little or no time to recover from mistakes. Consequently, a strong risk-management strategy appears to be the most important preventative measure.

References

[1] C. Boldyreff and J. Brittle. Self-organizing maps applied in visualising large software collections. In A. V. Deursen, C. Knight, J. I. Maletic, and M.-A. Storey, editors, *Proceedings of the 2nd IEEE Workshop on Visualising Software for Understanding and Analysis*, pages 99–104. IEEE, September 2003.
[2] 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.
[3] D. Nutter, S. Rank, and C. Boldyreff. Architectural requirements for an Open Source Component and Artefact Repository System within GENESIS. In *Proceedings of the Open Source Software Development Workshop*, pages 176–196. University Of Newcastle, February 2002.
[4] P. Ritrovato. Generalised enviroment for process management in cooperative software engineering. In *Workshop on Cooperative Supports for Distributed Software Engineering Processes, Proceedings of COMPSAC2002*, pages 1049–1053. IEEE, August 2002.