

On Software Engineering Methodologies for eScience

Andrew Martin and Lee Momtahan
Oxford eScience Centre

Abstract

eScience represents a significant convergence of many disciplines in order to promote a change in the dynamic of the way science is undertaken (Taylor). It follows that good use of results from those disciplines must be made, if good progress is to be achieved. Our interest is in the application of Software Engineering disciplines to the eScience endeavour.

An earlier paper (Momtahan and Martin, 2002) explored something of a mismatch between the present conduct of eScience and industrial best practice in Software Engineering. Some aspects of this are inevitable, because science is conducted differently from business; other features arise simply because those engaged in scientific research are simply not educated in the core disciplines of software engineering, or seek to argue that those ideas are in fact inapplicable to their research.

A significant problem arises in moving from the kinds of software which are small and self-contained, to the genuinely large components which are envisaged by many eScience projects and Grid applications. The former can usually be comprehended in their totality by one individual, are either short-lived (relevant to just one experiment) or 'handed-down' from one doctoral student to another and lovingly curated, and have only very simple interfaces or dependencies on other programs and data.

It has long been recognised in most sections of the Software Industry that moving from such projects to genuinely large, long-lived software, with complex interfaces, requires a complete change of process. A major discontinuity must be addressed; good engineering practice is needed. This comes as no surprise to practitioners of other engineering disciplines.

Decisions about the adoption of particular technologies and processes will be determined by the context in which work is undertaken — we previously observed such peculiarities of the academic context as low staff attrition and 'square pulse' funding regimes. Some problems are profound and may be expected to be hard to accommodate: communication overheads may rise exponentially with the size of the project, and multi-site projects exasperate this complexity.

Based upon our observation of a number of eScience projects we make some suggestions for approaches which help to maximise the scope for delivery of useful systems and middleware (the full paper will expand and justify these):

1. Consider lifecycle models appropriate for the project, and adopt one. Many eScience applications seem to fit well with iterative development. Document requirements, designs, and test plans — or have a clear rationale for not doing so.
2. Start with a small project team; create a project culture that respects the adopted lifecycle model and processes, and shares an interest in the project and institution's reputation.

3. Make good use of tools, both software tools and conceptual tools. Train staff in UML. Use techniques such as CRC cards to gain a shared understanding of a distributed system's interactions.
4. Use change management, source code control and bug tracking from the outset.
5. Security planning and analysis should be integrated with the design lifecycle. The requirements activity should capture the high-level security goals. As designs are elaborated, their performance against security goals should be explored.
6. If open source is a feasible outcome, adopt an open source *process* too: publish often; build frequently. Make basic functionality available quickly.
7. Ensure minimal geographical separation of development effort. Where separation is necessary, apportion the tasks *after* design work has been carried out.
8. Adopt a component-based development scheme, with crisply-defined interfaces that are as narrow as possible. Realise that it is not necessary for all of the developers to understand all the code; it's not even a good idea. Do not split the development of a single component across multiple sites.

These expedients are not by any means advanced techniques, but they have much potential to improve the quality, focus, and outcomes of eScience projects. Many of these ideas will be familiar to members of the project staff. The key to success is appropriate commitment from the senior staff and project leaders (see point 2). The development of an eScience Open Middleware Infrastructure Institute, and a Software Engineering group to oversee it, has much potential to raise the profile of these disciplines within eScience projects.

Projects are now being encouraged/forced to undertake security planning, and have long been expected to engage in some project milestone planning. It may be that in future they should also be asked, if they propose to produce software, to explain the techniques and methodologies to be employed.

References

- Martin, A. and Momtahan, L. (2002). e-Science: A Software Engineering Challenge, *UK eScience All Hands Meeting*.
- Momtahan, L. and Martin, A. (2002). e-Science experiences: Software Engineering practice and the EU DataGrid, *Proc. Asia-Pacific Software Engineering Conference*, IEEE Press, pp. 269–275.