Communication, coordination and control in distributed development: an OSS case study

Anna Persson, Brian Lings, Björn Lundell
University of Skövde
Skövde, Sweden
{ anna.persson | brian.lings | bjorn.lundell }@his.se

Anders Mattsson, Ulf Ärlig
Combitech Systems AB
Jönköping, Sweden
{ anders.mattsson | ulf.arlig }@combitechsystems.com

***Abstract* – It has been claimed that distributed development practices in OSS development may be a model for enterprise development practices of the future. With this in mind, we have conducted a study of one OSS project, namely ArgoUML, with a view to understanding development practice within the project, and specifically to considering possible differences from traditional (non-OSS) distributed development conducted in a commercial project. We do this by explicitly considering issues of communication, coordination and control. Our findings suggest that primary differences lie in control and resulting project structures, motivated through differing goals. We comment on the open question of how the advantages of one development context can be realised in the other.**

I. INTRODUCTION

Open Source Software (OSS) is a multifaceted phenomenon which can be analysed from a variety of perspectives [1]. A central characteristic of the development model used in Open Source projects is that it relies on contributions from a large number of geographically distributed developers. In recent years, the community has witnessed several Open Source projects which have produced successful and widely used products (such as, for example, Linux, Apache, MySQL) using this development model. Based on reported success for this development model, proponents have claimed that the Open Source community may be a model for enterprise development practices of the future (see, for example, [2, p. 562]). At the same time, [3, p. 1281] claims that research into the phenomenon has ignored "what may be a key independent variable: the size of the user-programmer community", and that "the major reason open source projects fail is a lack of user-contributors to do the work."

Our primary interest in the research project out of which this work has evolved, is in understanding the phenomenon of distributed development (DD), and specifically the differences between DD in open source projects and in commercial projects[1]. To this end we report on a case study of DD within one Open Source project, namely ArgoUML. In one sense our case study is different from most previous studies, in that the project under investigation is a development tool and there is likely to be less of a consensus on requirements (most previous case studies on OSS projects are concerned with projects dealing with systems/infrastructure level where there is more common agreement on requirements). We have a secondary interest in the role of Open Standards in DD, and have used this to add more focus to this study. This is because much emphasis is placed on Open Standards by the OSS community, relating to an ongoing debate about the problems of vendor-dependence (tool and systems lock-in) [4] [5].

We study the available sources of information from the Argo project, and draw out a number of themes which emerged during our analysis of the processes in this DD project. We briefly contrast these with DD experience in one commercial company. Overall, we find that although there are some similarities between DD in OSS and commercial projects, key differences in goals lead to some interesting differences in coordination, communication and control. We postulate that each type of distributed activity has much to learn from the other in order to better harness individual enthusiasms dynamically whilst meeting broad project goals.

II. BACKGROUND

ArgoUML started life in 1995 as a university research project by Jason Robbins [6] [7] [8]. The main goal of the Argo project was to develop the modelling tool ArgoUML (hereafter simply referred to as Argo). Argo was made publicly available in the middle of 1998 and turned into an open-source project in the beginning of 1999. In 2001 the company Gentleware developed a non-open source extension of the Argo code base called "Poseidon for UML". The ArgoUML project itself is the focus of this paper.

Argo is an object-oriented design tool using the Unified Modeling Language (UML) design notation. It supports the creation and saving of eight out of nine of the standard UML version 1.3 diagrams (not the object diagram). Forward engineering of Java, PHP and C++ code, and reverse engineering of Java code are also supported. The tool is implemented in Java and consists of approximately 1100 classes (version 0.16). It is made available without charge under the BSD Open Source License.

The Argo project attaches great importance to following open standards [6] [9]. Robbins and Redmiles [8] state that this is because "Standardization produces an economy of scale that leads to more and better tools, better interoperability between tools, more developers who are skilled in using the standard notation, and lower overall training costs". The use of open standards guides development and reduces the need to develop and document new approaches, and helps the team to avoid producing an idiosyncratic product [8]. In the project homepage it is stated that: "The key advantage of open standards is that it permits easy inter-working between applications, and the ability to move from one application to another as necessary".

[1] Whilst acknowledging that OSS-development may be undertaken in a commercial project, we will use the phrase only when referring to projects undertaken in a commercial enterprise using 'traditional' (non-OSS) development models.

Proceedings of the First International Conference on Open Source Systems
Genova, 11th-15th July 2005
Marco Scotto and Giancarlo Succi (Eds.), pp. 88-92

A. Argo project participants

Fig. 1 shows an overview of participant roles in the Argo project. Recursively, a participant with a role in one rectangle can (and often does) act in the role of the enclosing rectangle. This is a common characteristic in OSS projects [10] [11].

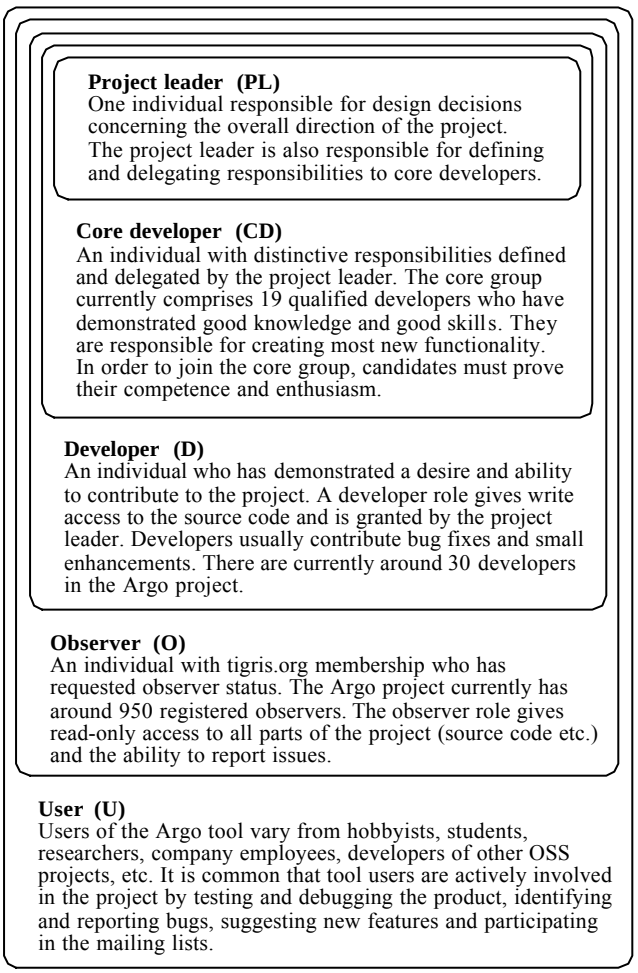

Fig. 1. Argo project participants

B. Argo project infrastructure

The Argo project infrastructure mainly comprises mailing lists, an issue tracking system and a homepage.

Mailing lists:

The Argo project has nine mailing lists for various types of discussions. Most discussions take place via the developers' mailing list, which has more than 17,000 posts (January 2005). This list is used by Argo developers to discuss directions for the project, choices of technical solution, documentation contents etc.

Issuezilla:

When it comes to specific problems related to the topics discussed in the developers' mailing list, the Issuezilla system is used. Issuezilla (based on Bugzilla, www.bugzilla.org) is an issue database and tracking system. Issuezilla make it is possible for project members to enter new issues, browse existing issues, modify issue parameters and comment on issues. Issues could be defects (bugs), feature or enhancement requests, tasks and patches. Issuezilla is used to handle all kinds of issue, not just code-based issues.

Homepage:

The project web site contains a large amount of information intended for users as well as developers. For developers there is extensive information available through different kinds of project documentation. This documentation communicates information to developers.

III. RESEARCH METHOD

It has been claimed for DD that "primarily, processes of communication, coordination and control are affected by distance, with direct consequences on how software is defined, constructed, tested and delivered to customers, as well as how its development is managed." [12, p. 793] This has influenced the perspective from which we analyse the Argo (OSS) project.

Data for the study were collected from publicly available electronic archives of project communications. In particular, we used the following data sources:

- Developer mailing list (hereafter referred to as "dev_list")
- Module development mailing list (hereafter referred to as "mod_list")
- Issuezilla
- Project home page.

On the project home page facts, news, history and other valuable information related to the project are published.

The contents of these data sources were analyzed with a specific focus on issues related to work on the open standards: PHP, UML, XMI and OCL.

IV. PROJECT ISSUES

In the data sources, a number of people feature, in a number of roles. To make sense of issue development we use role abbreviations (see Fig. 1) together with a chronology indicator to signify individual contributors. Thus, O1 is an observer – the first chronologically from the portions of the sources related to our focus.

A. Intra-project issues

Argo supports PHP code generation through a plug-in module. Towards the end of 2002 the PHP module worked with the fresh development release. On testing PHP with this release, O1 noticed that erroneous PHP code was generated and so created a bug issue in Issuezilla.

CD3, who had previously been involved in PHP issues, noticed the reported bug and committed source code to the CVS that solved the problem – within 24 hours of the report. CD3 informed about the patch and requested O1, as the reporter of the issue, to verify that the bug was eliminated. O1 validated the patch – changing its issue status to "verified" – but informed that a new problem had been introduced as a side effect. When CD3 noticed this he declared his dissatisfaction with the solution and changed

the issue status to "reopened", but could not reproduce the error so asked O1 about the steps needed to trigger it. This request was not answered by O1 and no further effort was made related to the issue for more than a year.

The lack on activity around the issue was noticed by PL2, who requested CD3 to take another look at it. CD3 did not react, but O1 did – with a suggested approach to the problem. There were no reactions to this suggestion, but O1 started to work on his idea anyway. Three weeks later O1 requested that someone inspect his new PHP module for commit to CVS if approved. This task was accepted by PL2. Obviously PL2 liked the contribution from O1 as within two weeks O1 was allotted a core developer role with PHP responsibility.

B. Inter-project issues

Software created in other OSS projects is used in several Argo components. For example, Argo relies on the Novosoft meta-model library (NSUML[2]) for the UML meta-model and XMI implementation. NSUML was introduced in Argo in the beginning of 2000 as a substitute for Argo's own meta-model. Further, its OCL implementation is based on the Dresden OCL Toolkit[3].

XMI from NSUML:

XMI support in Argo has been a subject of much discussion. Here we pick up on the format of Universally Unique Identifiers (UUIDs) in exported XMI files. A UUID is a globally unique element identification attribute that is mandatory in a valid XMI document [13]. UUID format is not included in the XMI specification, but is said to typically be DCE [13]. DCE (Distributed Computing Environment) is an open standard for distributed computing provided by the Open Group.

The UUID format became a subject of discussion in the dev_list when D1 reported that UUIDs in XMI files exported from Argo looked strange. At once NPL[4] made clear that this is not a NSUML problem since UUIDs are not generated by NSUML. CD1, who first implemented XMI in Argo, explained that the format resulted because he could not implement DCE compliant UUIDs because of limitations in the Java language. CD1 welcomed suggested improvements, and other developers responded.

D1 quoted from the XMI specification about DCE UUID and compared this with the format used in Argo. He informed the other developers that he will try to work with the problem. CD1 expressed his opinion that DCE compliant UUIDs would be very hard to realize. D2 partly disagreed, suggesting a workaround, but CD1 pointed out that the suggested workaround was the approach currently used in Argo. CD1 also expressed discomfort at trying to match DCE UUID format when it wasn't true DCE. D2 concurred and shortly presented a patch that produced UUIDs that explicitly expressed non-DCE compliance. This patch, however, seems never to have been included in any Argo release.

NPL had watched the ongoing discussion and explained a problem with the UUID approach used in Argo. This was not seen as the real problem by D2, who explained what he thought was a more serious problem but doubted that the whole issue was of major importance. NPL disagreed, and expressed an interest in working on it.

Once again D1 expressed the importance of DCE UUIDs in allowing Argo's XMI documents to be used in large repositories – something CD2 disagreed was a major problem. To avoid the UUID problem in Argo he asked NPL if UUID support could be added to NSUML. NPL declared his reluctance to do this. This impasse means that today UUIDs in Argo's XMI files are still not DCE compliant.

UML from NSUML:

In the end of 2002 NPL declared that NSUML was currently in "very deep hibernation" with "little hope of unfreezing". From this time the extensive dependency on NSUML was shown to be hampering the Argo project. One of many illustrative cases is work on support for concurrent composite states in activity diagrams.

O2 noticed that such support was currently inadequate. He created an enhancement issue to draw attention to the situation, and started to work on it. However, during this work it became apparent that concurrent composite states were not being properly saved. O2 reported this as a problem in NSUML. CD4 advised O2 to try the latest release of NSUML (Argo was lagging one release behind), but O2 confirmed that he was already using the latest release. He also reported that he had raised the problem in the NSUML project but received no answer. CD5 pursued this, creating a bug report (including a patch) at the NSUML homepage in the name of the Argo project. CD5 also took the opportunity to suggest that the Argo project should get administrator rights in order to make it possible for them to do NSUML bug fixes. NPL answered the he had no time to work with NSUML and is no longer working at Novosoft. He declared willingness to give administrator rights to a member of the Argo project, but warned that there may be a remote possibility that Novosoft would claim the project back in the future. CD5 reported this problem in Issuezilla, noting the apparent death of the NSUML project and lack of Argo rights in the project.

Issue activity now stopped for some time until CD4 requested an updated status report. He also noted that the NSUML situation might be a problem for the Debian project, a suspicion confirmed by U1. PL2 explained that although the NSUML project were willing to give administrator rights, CD5 didn't wanted that responsibility. This led to debate over whether NSUML should be created as a sub-project of Argo called ArgoNSUML. Eventually PL2 expressed a strong opinion that the project should be kept on Sourceforge instead of moving it to Tigris, to minimize workload. CD4 concurred, but noted the problem that nobody knows how the NSUML code generator works.

A long term goal in the Argo project is to replace NSUML with an as yet undetermined alternative. The objective is to have the first non-NSUML version of Argo released in March 2005, but as stated in the release plan "Who is working with this?"

[2] Founded by the Russian company Novosoft and developed at the OSS community Sourceforge [14].

[3] Founded in a project at the Dresden University of Technology, http://dresden-ocl.sourceforge.net/

[4] The NSUML project leader.

OCL Toolkit from Dresden:

A number of contributions to Argo related to OCL have come from the Dresden OCL Toolkit project, several from one member of that project – hereafter referred to as DOT.

In an e-mail DOT informed that he had started to work on enhancements to the code generator to allow it to generate OCL constraint-related tags into documentation comments. He explained that these tags can be evaluated by the toolkit's OCL injector. This would enhance the source code to allow run time checking of OCL constraints. He attached source code and asked for comments and CVS commitment if the code was approved. After about a week DOT asked why the code he sent to the list hadn't been committed yet. PL1 excused himself for overlooking it, and informed that the code was now committed. He also requested that others take a look at it.

DOT then sent improved code to the dev_list together with a request for CVS commitment. The code was inspected by D3 who did not support a commit due to an error found in the code. Shortly after, D3 retracted this statement, confirmed that the code was correct, and committed it to CVS.

A few days later DOT posted further improvements to the code on the dev_list. Again there was no response, so after some days DOT again asked about the absent CVS commit. He also argued that his lack of CVS commit rights hampered his progress on improving the integrating between Argo and the toolkit. PL1 responded that the code worked fine and he would check it in. When a further refinement met a similar lack of response, PL1 intervened and gave DOT CVS commit rights. D4 expressed strong support for DOT's later committed code, and after a month PL1 granted DOT a core developer role with responsibility for the OCL module.

V. ANALYSIS

In Argo, coordination and control is served by a system of role allocation supported by publicly accessible information channels by which participants communicate and can monitor issues. Role allocation is dynamic, controlled by the project leader and based on participant interest and demonstrated contribution. If a role is not being actively filled, then a keen and able participant is drafted in to fill the vacuum.

Monitoring is non-automated, and in the absence of detailed schedules an issue can potentially lie dormant for extended periods of time. When issues are cross-project, as occurs when Argo relies on components from other OSS projects, cross-posting of reports occurs and dormancy can result if the issue is not taken up within that project.

However, this is unlikely to be the case with issues of current import as the numerous users and observers, some of whom will be members of several projects, continually feed back with questions when technical problems affect them, raising the profile of the associated issues.

Issue dormancy primarily occurs when a link in the chain becomes inactive: either a developer or another project. In such circumstances the project leader is the final monitoring backstop; but as observed, this may allow a (non-topical) issue to be ignored for a year, or a suggested solution to a problem to be left out of releases.

The project leader role is core to the success of the Argo project, and demands great commitment and a clear sense of what is happening project-wide. The project leader became involved in all issues reported here, and was responsible for noting long-term inactivity in one, and directing the attention of an alternative core developer to it.

There seems to be a clear protocol for the use of the different communication channels available. For example, inter-project issues (such as XMI and OCL functionality) are discussed via the developer mailing list only: interested members of other projects tend to subscribe to this list rather than following wider-ranging (though structured) discussion in Issuezilla. However, mailing lists, being a push technology, involve high overheads for subscribers, and issues can get lost in the volume of mail. Code changes may be picked up by anyone or no one – as happened with OCL, to the frustration of the observer developing the code.

Issuezilla, however, supports coordination and communication by grouping information by issue, and logging progress and status. Monitoring is therefore simpler. However, active browsing is required. The project leader seems to check for unsolved issues periodically, attempting to promote activity by drawing attention to dormant issues; but at the end of the day, task assignment is not part of the consensual nature of "community software development" which is one of the goals of Argo.

In fact, the Argo project has several other stated goals, including: cognitive support for design; adherence to the UML specification; comprehensive support for OCL and XMI; and modularity and extensibility. However, contributors have their own goals – for example a well-functioning PHP generation for their own development activities. The distributed nature of the project, and the diverse nature of the project participants, make it hard to agree upon more than general common goals – and priorities are implicit only, reflecting who is willing and able to contribute to which issues.

Real issues have arisen around dependencies with other OSS projects, and particularly when a project goes dormant. The reliance on NSUML has become of project-wide concern, exercising many developers and requiring a position to be taken. Discussions ensued about Argo developers reviving the project or taking it over as an Argo sub-project. Ultimately, such strategic decisions are made by the project leader. In this case, a new goal was created of making Argo independent of NSUML, and this new goal was even given a deadline. However, task allocation and scheduling are alien to the development philosophy of Argo, which relies on pressure (as well as solutions) coming from the grass roots.

VI. REFLECTIONS AND CONCLUSIONS

A number of differences can be observed between distributed development approaches observable in Argo and those in a typical commercial systems development enterprise. Specifically, control and structure seem to be weaker in Argo, as "predictable time scales" is not an Argo project goal. There is an industry view that you pay people to meet this goal, as it requires developers to sacrifice the freedom to work on what they find most interesting, with the potential drawback of reducing enthusiasm. Roles may change in a commercial development environment, but they are more strictly under management control and

largely fixed. This again has the potential for being misaligned with the interests of individuals, and so may lead to under-use of latent skills and enthusiasms.

In a commercial enterprise, a developer typically has read only rights to modules other than their own. Issues are tracked in a similar way to Argo, using both developer mailing lists and issue control software, and with similar advantages and drawbacks. Commercial channels will be more focussed, however, reflecting the development structure. This has the advantage of shielding developers from "irrelevant" issues, related to other components; and the disadvantage of preventing potential contributions from outside a fixed team. However, the Argo channels will be supplemented in a commercial context by technical review meetings, teleconferencing and NetMeetings.

It could be conjectured that development paradigms which encompass the goals of both OSS and commercial projects is the holy grail of distributed development. How commercial distributed development efforts can effectively tap the creativity, enthusiasm and validation potential evident in OSS development is one open question. A corresponding one for the OSS community is how increased activity can be motivated in order to more predictably meet project as well as individual goals.

VII. ACKNOWLEDGMENT

This research has been financially supported by the European Commission via FP6 Co-ordinated Action Project 004337 in priority IST-2002-2.3.2.3 'Calibre' (http://www.calibre.ie).

VIII. REFERENCES

[l] J. Feller and B. Fitzgerald, *Understanding Open Source Software Development*, Addison Wesley, London, 2002.

[2] L. Augustin, D. Bressler and G. Smith, "Accelerating Software Development Through Collaboration," in *Proceedings of the 24th International Conference on Software Engineering*, ACM, New York, 2002, pp. 559-563.

[3] J. West, "How open is open enough? Melding proprietary and open source platform strategies," *Research Policy*, Vol. 32, No. 7, 2003, pp. 1259-1285.

[4] ITEA, "ITEA Report on Open Source Software", ITEA Office Association, January 2004, Available online: www.itea-office.org.

[5] Statskontoret, "Free and Open Source Software – a feasibility study," 2003:8a, Statskontoret, Stockholm, 2003, Available online: http://www.statskontoret.se/upload/Publikationer/2003/200308A.pdf.

[6] J.E. Robbins, "Cognitive Support Features for Software Development Tools," Ph.D. Dissertation, University of California, Irvine, 1999, Available online http://argouml.tigris.org/docs/robbins_dissertation/.

[7] J.E. Robbins, "Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools," in Feller, J., Fitzgerald, B., Hissam, S.A. and Lakhani, K.R. (Eds.) *Perspectives on Open Source and Free Software*, The MIT Press, Cambridge, MA, 2004.

[8] J.E. Robbins and D.F. Redmiles, "Cognitive support, UML adherence, and XMI interchange in Argo/UML," *Information and Software Technology*, Vol. 42, No. 2, 2000, pp. 79-89.

[9] ArgoUML, "The ArgoUML project," Available online http://argouml.tigris.org, Accessed 15 January 2005.

[10] A. Mockus and J.D. Herbsleb, "Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?," in *Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering*, held in conjunction with the 24th International Conference on Software Engineering, Orlando, FL, May 2002.

[11] C. Gacek and B. Arief, "The Many Meanings of Open Source," *IEEE Software*, Vol. 21, No. 1, 2004, pp. 34-40.

[12] D. Damian, F. Lanubile and H.L. Oppenheimer, "Addressing the Challenges of Software Industry Globalization: The Workshop on Global Software Development," in *Proceedings 25th International Conference on Software Engineering*, IEEE Computer Society, Los Alamitos, 2003, pp. 793-794.

[13] OMG, "OMG-XML Metadata Interchange (XMI) Specification," version 1.0, 2000, http://www.omg.org/docs/formal/00-06-01.pdf.

[14] NSUML, "Novosoft metadata framework and UML library," Available online http://nsuml.sourceforge.net, Accessed 15 January 2005.