THE DISTRIBUTED OPEN SOURCE SOFTWARE DEVELOPMENT MODEL: OBSERVATIONS ON COMMUNICATION, COORDINATION AND CONTROL[1]

Björn Lundell and Brian Lings, University of Skövde, P.O. Box 408, SE-541 28 Skövde, Sweden, {bjorn.lundell | brian.lings}@his.se

Pär J Ågerfalk and Brian Fitzgerald, University of Limerick, Limerick, Ireland, {par.agerfalk | bf}@ul.ie

Abstract

There are many reasons why an organisation should consider adopting distributed development of software systems and applications, including access to a larger labour pool and a broader skills base, cost advantages, and round the clock working. However, distributed development presents many challenges stemming from the complexity of maintaining good communication, coordination and control when teams are dispersed in time (e.g. across time zones) and space, as well as socio-culturally. The open source software (OSS) development model is distributed by nature, and many OSS developments are considered success stories. The question therefore arises of whether traditional distributed development models can be improved by transfer of successful practice from OSS development models. In this paper we compare OSS with traditional distributed development models using a framework-based analysis of the extant literature. From our analysis we find that the advantages of temporal and geographical distance dominate in OSS, rather than their associated problems. Further, socio-cultural distance is lowered by active developer selection. However, there is a challenge to satisfying project goals when personal goals dominate.

Keywords: Industrial Open Source, Open Source Development, Development Models, Global Software Development, Distributed Development.

[1] 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), and also by the Science Foundation Ireland Principal Investigator projects B4-STEP and Lero.

1 INTRODUCTION

Distributed development of software systems and applications (DD) is an issue of increasing significance for organisations today, all the more so given the current trend towards outsourcing and globalisation. However, as well as involvement in conventional DD many companies are getting involved in open source software (OSS) development projects, which have their own development models of DD.

The core challenges of DD seem to lie in the complexity of maintaining good communication, coordination and control when teams are dispersed in time (e.g. across time zones) and space, as well as socio-culturally. Since the OSS development model is distributed by nature, and many OSS developments are considered success stories, the question arises naturally as to whether lessons can be transferred between OSS and distributed development. However, proven methods for successful DD have not yet been formulated, and there is a need for a better understanding and transfer of lessons in relation to all distributed software development. As put by Crowston et al. (2005): "Understanding the work practices of teams of independent knowledge workers working in a distributed environment is important to improve the effectiveness of distributed teams and of the traditional and non-traditional organizations within which they exist." (p. 7)

In line with this sentiment, proponents of the Tigris.org development platform (www.tigris.org) claim that there is much to learn for traditional[2] DD projects by taking a closer look at OSS projects. As stated on the Tigris.org website, the "software engineering field can learn much from the way that successful open source projects gather requirements, make design decisions, achieve quality, and support users."

To facilitate such learning, this paper compares OSS development models with traditional DD models by means of a framework-based analysis of the extant literature on case studies in OSS development. From our analysis we derive the underlying characterisations of OSS development models in the literature. These characterisations can help to inform anyone involved in any kind of DD. As the analysis is based on a previously established framework for DD, our analysis will allow broad comparisons to be made.

2 BACKGROUND

For the purpose of this research, we take the position of Ågerfalk et al. (2005) in defining DD. A development team is distributed if its team members are not co-located, but geographically spread out. Here, *development* is interpreted broadly as *any software development lifecycle activity*. This thus extends beyond *pure* development activities and includes, for example, deployment and maintenance.

For a number of years the international workshop on Global Software Development (GSD) has highlighted the impact of distribution on *communication*, *coordination* and *control* within DD lifecycle activities (see, for example, Damian et al., 2003). This view is consistent with the position taken by a number of authors who have focused on one or more of these three fundamental processes (e.g. Carmel and Agarwal, 2001; Evaristo et al., 2004; McChesney and Gallagher, 2004; Nurmi et al., 2005; Sutanto et al., 2004). In particular, the communication, coordination and control activities are affected over a number of *dimensions*, which have been well elaborated in the literature (e.g. Battin et al., 2001; Espinosa and Carmel, 2003; Ghosh et al., 2004; Nicholson and Sahay, 2001; Sutanto et al.,

[2] We use the phrase "traditional distributed development models" when referring to approaches and models for conducting (non-OSS) distributed development in commercial contexts.

2004). These relate to temporal, geographic and socio-cultural distance. These processes and dimensions have been incorporated into a framework of issues in DD (Ågerfalk et al., 2005).

We will use this framework to present the results of our own study on Open Source development, and so introduce it briefly here (see table 1 below). Basically, *communication* refers to exchange of information. For communication to be successful, exchanged information should be complete and unambiguous so that sender and receiver can reach a common understanding (Carmel and Agarwal, 2001). The communication process concerns the transfer of knowledge and information between actors, and the tools used to facilitate such interaction. *Coordination* is “the act of integrating each task with each organisational unit, so the unit contributes to the overall objective.” (Carmel and Agarwal, 2001, 23) The coordination process concerns how this interaction makes actors interdependent on each other. *Control* is “the process of adhering to goals, policies, standards, or quality levels.” (Carmel and Agarwal, 2001, p. 23) The control process concerns the management and reporting mechanisms put in place to make sure a development activity is progressing. *Temporal distance* is a directional measure of the dislocation in time experienced by two actors wishing to interact. Temporal distance can be caused by time zone difference or time shifting work patterns. In general, low temporal distance improves opportunities for timely synchronous communication but may reduce management options. *Geographical distance* is a directional measure of the effort required for one actor to visit another at the latter's home site. Geographical distance is best measured in ease of relocating rather than in kilometres. In general, low geographical distance offers greater scope for periods of co-located, inter-team working. *Socio-cultural distance* is a directional measure of an actor's understanding of another actor's values and normative practices. As a consequence, it is possible for actor A to be socio-culturally closer to actor B than B is to A. It is a complex dimension, involving organisational culture, national culture and language, politics, and individual motivations and work ethics. In general, low socio-cultural distance improves communication and lowers risk.

A development context is considered distributed if it exhibits significant distance in the geographical dimension. We would consider a development team comprising members in two different offices in different cities within the same country to be distributed, even if they exhibit low temporal and socio-cultural distance. The key feature is that the cost (not necessarily monetary) to bring dispersed team members together is a significant inhibiter to spontaneous face-to-face meetings. When a DD project exhibits high distance in all dimensions, it is commonly referred to as a GSD project.

The complete framework, presented as table 1, forms a matrix in which each cell represents the impact of one dimension on one process. The table has been populated with an overview of the DD issues relating each process to each dimension (from Ågerfalk et al., 2005). This is the basis for our later comparisons.

3 RESEARCH METHOD

In this paper, we use the framework of table 1 as a basis for analysing the extant literature on case studies in OSS development. Our goal is to compare OSS development models with traditional DD models.

Based on an analysis of the published literature, we first review documented case studies in OSS which focus on process. Our initial goal is to characterise the contexts in which OSS development takes place. Following this, we consider the broader literature on reported OSS development experience. We look specifically at how reported experience relates to the framework of Ågerfalk et al. (2005), thereby systematically considering threats to communication, coordination and control in OSS development caused by Temporal Distance, Geographical Distance, and Socio-Cultural Distance. This results in populating the framework with the characteristics of, and work practices used in OSS development.

We then use the two populated frameworks to make observations on how OSS work practices relate to the issues in DD. For the literature analysis, systematic searches of the literature were made using keyword and author searches, and searches of tables of contents of Journals, and Conference and Workshop proceedings. Bibliographic databases were used to assist in forwards and backwards referencing. Papers were included if they had a core focus on DD in OSS, or were considered highly relevant for understanding core issues raised in the literature. An extensive note file was also compiled, with quoted sections from papers which contained their major import. This allowed faster filtering in the later stages of analysis, but context was always checked against the full text. We illustrate the research process in figure 1, which is annotated to show the structure of the paper.

Process	Dimension		
	Temporal Distance	Geographical Distance	Socio-Cultural Distance
Communication	Reduced opportunities for synchronous communication, introducing delayed feedback. Improved record of communications.	Potential for closer proximity to market, and utilisation of remote skilled workforces. Increased cost and logistics of holding face to face meetings.	Potential for stimulating innovation and sharing best practice, but also for misunderstandings.
Coordination	With appropriate division of work, coordination needs can be minimised. Coordination costs typically increase with distance.	Increase in size and skills of labour pool can offer more flexible coordination planning. Reduced informal contact can lead to reduced trust and a lack of critical task awareness.	Potential for learning and access to richer skill set. Inconsistency in work practices can impinge on effective coordination, as can reduced cooperation through misunderstandings.
Control	Time zone effectiveness can be utilised for gaining efficient 24x7 working. Management of project artefacts may be subject to delays.	Difficult to convey vision and strategy. Communication channels often leave an audit trail, but can be threatened at key times.	Perceived threat from training low-cost ‘rivals’. Different perceptions of authority/hierarchy can undermine morale. Managers must adapt to local regulations.

Table 1: An Overview of the Framework for Analysing DD (after Ågerfalk et al., 2005)

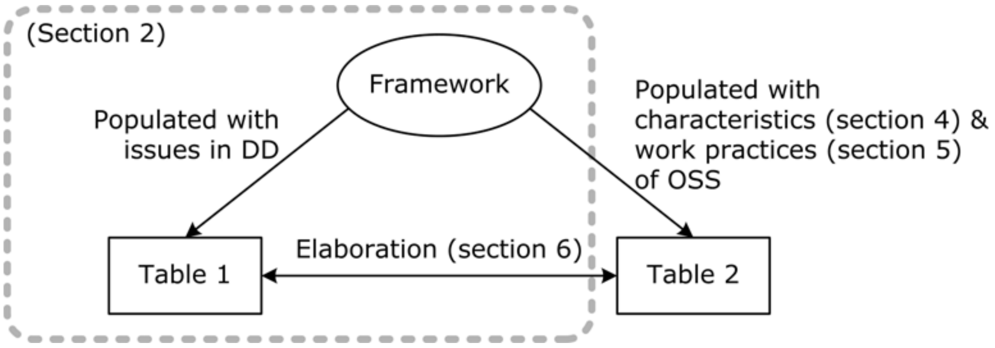
(Section 2)
Framework
Populated with issues in DD
Populated with characteristics (section 4) & work practices (section 5) of OSS
Table 1
Elaboration (section 6)
Table 2

Figure 1: An Overview of the Research Process (with reference to paper structure)

4 VARIETY IN OSS DISTRIBUTED DEVELOPMENT CONTEXTS

OSS development is often characterised as occurring in a very different context from traditional development as managed in commercial companies. This is because the motivational factors can be very different (Bergquist and Ljungberg, 2001; Feller and Fitzgerald, 2002; Haruvy et al., 2003). However, it should be noted that the majority of contributors to OSS projects are now employed in commercial companies (Ghosh et al., 2002). In fact, the relationship between companies and OSS projects is a complex and dynamic one, which must be considered in any attempt to reflect on OSS development – whether or not the DD aspects are of primary importance. Dahlander and Magnusson (2005) characterise three types of relationship: parasitic (in which the commercial interest is indifferent to its effect on OSS) – of great concern to the OSS community[3] as over-exploitation can threaten the "OSS ecosystem"; symbiotic (in which each gains advantage); and commensalistic (referring to a commercial interest not harming the OSS project). Apart from benefiting by improving an OSS product on which a company relies, a symbiotic relationship may result from less obvious benefits. For example, Lussier (2004) details an instance of process enhancement in a company brought about through the experience of its developers in an OSS development project.

To expand on this theme, we consider three case studies chosen because of their direct concern with issues related to communication, coordination and control in specific OSS projects. Although other case study papers have been published (see, for example, Dinh-Trong and Bieman (2004), Franke and von Hippel (2003), Koch and Schneider (2002), Mockus et al. (2002), Moon and Sproull (2000)) only the three studies described directly address the above issues.

Interestingly, the three studies address three very different contexts of OSS development, giving three different perspectives on contrasting models for OSS development with traditional DD. The first contrasts OSS development activities in a project with limited commercial input, with those of a commercial software development organisation (Persson et al., 2005). The second considers the phenomenon of employees of commercial companies paid to support OSS projects (German, 2003, 2003b). The third reports on the experience of an intra-company OSS project (Gurbani et al., 2005).

Process discovery in an OSS project has been identified as a challenging problem (Jensen and Scacchi, 2005b), but is as least made possible by the existence of detailed project information provided over the Internet. The ArgoUML project (argouml.tigris.org/) aims to develop a UML modelling tool with a good interface. Persson et al. (2005) analysed the process dimension of ArgoUML by tracking specific issues raised in developer mailing lists, module development mailing lists, Issuezilla and the project home page. They noted a lack of progress on certain stated goals, concluding that control and structure are weaker in ArgoUML than in a commercial context, for which "predictable time scales" is an important 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, but with the potential drawback of reducing enthusiasm. Roles within the ArgoUML project change unpredictably, changes being made by the project leader in response to core developer inactivity. Roles may also change in a commercial development environment, but they are more strictly under management control and largely fixed. This again has the potential for roles being misaligned with the interests of individuals, and so may lead to under-use of latent skills and enthusiasms. ArgoUML channels of feedback are open, allowing all interested contributors to monitor change. This is in contrast to commercial channels of feedback, which are usually more focussed, reflecting development structure. The latter 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.

The GNOME project (www.gnome.org) aims to develop a free desktop for Unix. German (2003, 2003b) relates a case study of paid employees of companies contributing to the GNOME project. These developers are generally paid to contribute because their employer sees benefit in progressing the project at a steady pace. Hence, they pay for work which is less attractive to volunteers, such as project design, coordination, documentation and bug fixing. Such developers may even become dominant in terms of lines of code committed to the project CVS. They may also be active beyond code contribution, for example adding accessibility features. Volunteers still contribute as bug hunters, contributors and in specialised contributions such as internationalisation. However, straightforward

[3] A recurring concern raised by practitioners during the EU FP6 Calibre series of industrial conferences/workshops.