Configuration Management support for Distributed Software Development

Lars Bendix
Department of Computer Science
Lund University, Sweden
bendix@cs.lth.se

Jan Magnusson
Sony Mobile Communications
Lund, Sweden
Jan.Magnusson@sonymobile.com

Christian Pendleton
Softhouse Consulting
Malmö, Sweden
Christian.Pendleton@softhouse.com

ABSTRACT

In an ever more globalized world, there are many advantages from doing distributed development. However, geographically distributed development is generally recognized as being much more challenging than traditional co-located development and therefore companies are often hesitant to "go distributed". Configuration management is a basic service that provides the infrastructure for projects and organizations. In this paper, we analyse the role that configuration management can play in the context of distributed software development. From a configuration management point of view a good part of the challenges from distributed development can be dealt with by applying traditional configuration management concepts and techniques, many challenges can be alleviated by extended support from configuration management – and then there is a group of challenges left that will have to be dealt with in other ways.

Keywords

Configuration Management, distributed development, challenges, experience.

1. INTRODUCTION

Distributed software development is becoming more and more popular and for many good reasons. It gives a larger pool from which to recruit talents and specialists, allows cooperation between companies and/or departments, facilitates integration for mergers and acquisition, allows around the clock work, and gives more flexibility in scaling up and down projects. However, the coin also has a flipside and companies that do practice distributed development report problems of many different kinds.

So apparently something bad happens when we move from co-located development to distributed development. Something that means we lose control over our projects when we do "business as usual" like the distributed project was a co-located one. Why is that so and what is it that goes wrong?

We, as configuration management experts and practitioners, were rather puzzled with that fact. In our view, configuration management (CM) was put into the world exactly to handle certain aspects of distribution on traditional projects. On traditional waterfall projects developers are often distributed. Rarely requirements engineers, designers, testers and programmers are sitting at the same place at the same time. CM principles and techniques make sure that all the "handovers" between the different groups are controlled and done in an orderly way [6]. Even programmers are rarely at the same place at the same time and often work in parallel – and CM principles and techniques keep the team productive all the same [1]. CM even handles when development and maintenance makes work distributed in time. So, what is the big deal about this distribution?

Was there something that we did not understand – or was there something that others had overlooked [2]? We decided to look into the reported challenges by reviewing existing literature and to investigate to what degree they could be handled by using traditional CM principles and techniques. For this analysis we exploited our own expertise in CM as well as a number of practical cases from the project [2]. Finally, we wanted to explore if, extending and building on the concepts of what CM is, we could provide further help to distributed projects. This led to the categorization of challenges into three groups of strongly, weakly and not related to CM. For the latter category of challenges CM cannot help, whereas for the other two CM is able to solve or alleviate the challenge.

In the following, we first detail our research method and discuss related work. We then present the results of our analysis and discuss the motivations before finally drawing our conclusions.

2. RESEARCH DESIGN AND RELATED WORK

Our primary goal was not to look for challenges in general for distributed software development, but rather to identify how CM can help – directly or indirectly – to support distributed development efforts. For that reason we looked to literature for establishing what challenges are particular for distributed efforts in comparison to co-located efforts. We ended up with da Silva et al. [8] and Jiménez et al. [5] as the most substantial studies. There are many others, most of them reporting on single cases, but they could not add anything significant to the compound list from [8] and [5].

Jimenez et al. base their compilation on the review of 78 other studies and come up with 10 different challenges that are rather broad-sweeping in nature and therefore more difficult to handle when going into details. da Silva et al., on the other hand, come up with a rather detailed list of 30 specific challenges compiled from a review of 54 other studies. Obviously there is some overlap between the two reviews, but they also complement each other well. In addition da Silva et al. also present mappings between best practices and challenges, and between tools and challenges. However, we find it problematic (from a CM point of view) that they can only map the "*deploy a configuration management system*" best practice to the "*effective communication*" and "*trust*" challenges – and that the "*change management tool*" (apparently they do not consider configuration management tools) only maps to the "*cooperation*" and "*scope and change management*" challenges. The challenges per se, though, seem to fit nicely with those reported by others.

We also looked for prior art about CM in the context of distributed software development, but apart from reports from singular cases we were able to find only the studies by Pilatti et al. [7] and by Fauzi et al. [4]. Pilatti et al. report on 4 cases they have studied, whereas Fauzi et al. review 24 different reported studies. Both studies add more – potentially configuration management related – challenges to the compound list from [5] and [8]. However, we find problems in their claimed scope and in how they handle CM concepts and principles. We find that "*all configuration items required for a build should be put under CM*" [7] and "*lack of a planned baseline*" [4] should not be challenges since they are CM fundamentals that, if ignored, will leave any project – distributed or co-located – challenged. Furthermore, some of the reported challenges are more closely related to CM than others – while other challenges (like "*lack of coding standards*" [4]) are not related at all.

The compound list of distributed development challenges was then analysed from a configuration management point of view to find which challenges were strongly related to CM, which were weakly related to CM and which were not related to CM at all. Each of the authors has more than a decade of experience from CM. We also had 6 practical cases to refer to so we were able to discuss most challenges in practical contexts. Our primary focus was on the categorization of the challenges and not on finding solutions.

3. RESULTS AND DISCUSSION

In this section, we present the categorization of their relationship to CM that resulted from our analysis of the compound list of 61 challenges we found in literature. For the categorization we decided on three categories (strongly, weakly and not related to CM), though there can be no clear-cut division of the challenges since their degree of relationship to CM is not binary, but rather a continuous scale. So to get the detailed picture you need to read also the fine print [3]. However, the overall picture from the categorization will give a good idea of which challenges can to a large part be solved by CM, which you can get more or less help for – and which are outside the domain of CM (but may be solvable by others).

In the following, we will present a couple of examples of challenges from each of the three categories, motivate why they belong to a certain category and briefly discuss the CM principles and techniques involved. More examples and more details can be found in [3], but this brief overview should demonstrate how it works.

3.1 Strongly Related to CM

The main message of this sub-section is that CM is there and for some of the challenges it can actually solve the problem – you just have to use it. That a challenge is strongly related to CM means that it is mostly or for large parts a CM responsibility and already known to CM. The solution to the strongly related challenges is the more or less straightforward application of well-known CM principles and techniques.

Effective communication [8]: In order to communicate effectively, a group must share common concepts and definitions. If every piece of information must be built up from first principles, the signal will contain a lot of noise. CM can help in many areas by establishing a system for identities, names, versions, terms and hierarchies (taxonomies), etc, in order to enable precise communication. It is also the foundation for many other activities. For example, change management is not really possible unless you can describe what the change should be applied to. Likewise, how do you allocate tasks without a well-defined way of describing what item the task concerns. A well-designed framework for managing changes to the items defined can also help to reduce the amount of data that must be transmitted in every message. If you find yourself having problems describing tasks, changes or if a release requires long descriptions of what to send where, you are having CM issues.

Dispersed software teams do not get information on what other teams are doing [4]: In a distributed team setup, information about on-going activities will not naturally be passed from developer to developer across sites. If the only interactions with remote developers happen through the code repository when artefacts are retrieved or stored or the repository is queried, developers will be quite reliant on real time communication in order to avoid or resolve conflicts. Strategies with well defined tasks and exclusive areas of responsibility, are only valid if the architecture of the software worked on is such, that there are well-defined components with a clear and shared understanding in the organization of their scope and functionality and when adding two pieces of seemingly unrelated functionality, the probability for them to be dependent is low. Even if those two architectural requirements are fulfilled, the result is not mainly improved awareness of remote developers activities, but instead a way to reduce the risk that the communication deficit resulting from a large and/or distributed development organization affects on-going development. Another commonly used strategy to tackle the risk of unnoticed dependencies interfering with on-going work, is simply to limit the amount of work-in-progress, where continuous integration would be an example. Strategies aimed at improving awareness should encourage sharing rather than isolation.

3.2 Weakly Related to CM

This sub-section is probably the most interesting and promising since this category of challenges has most potential for getting supportive help from CM – something that apparently has been overlooked so far. Weakly related challenges are not really a CM responsibility, but since CM takes care of the repository many aspects of information and visualization can be supported by CM if planned and requested. Alleviating such challenges will in many cases require creativity and out-of-the-box thinking from CM in the implementation.

Different knowledge levels or knowledge transfer [8]: Great differences in knowledge levels make collaboration difficult whether a project is distributed or co-located. It means that there is the need to transfer knowledge from one party to another and in the distributed case there are many obstacles to the quick and easy transfer. These obstacles can be caused physical distance between sites or the timely distance between the development effort and the maintenance effort. CM does not deal directly with knowledge or knowledge transfer, but in the absence of face-to-face meetings one possibility is to capture the knowledge and make it persistent so it can be shared. CM routinely captures certain data related to the configuration items they manage and if needed this data (and much more) could be elaborated and put together to turn it into knowledge and made shareable through the repository. One example could to elaborate the dependency and history information already present in the CM repository and through that make program comprehension much easier.

Intellectual property issues/confidentiality and privacy [8]: It may seem farfetched that intellectual property issues could be related to CM. However, if a company should get sued for improperly using code or stops paying the licensing fee, then we want to be able to identify the code in question and find all the places it is used. Traceability in general is a core CM principle and even though traceability for intellectual property issues is not a standard in CM it can easily be provided if needed. Confidentiality and privacy issues are closer to CM. Setting up and handling access rights is a CM responsibility and many different schemes can cater for situations where, for example, a remote site can change only certain parts of the items and maybe not even see other parts. However, setting up firewalls and other measures to make sure that code and data do not get outside the company is not a CM responsibility.

Quality and measurement [5]: CM is not directly responsible for quality even if CM in many companies has been initiated by request from the quality assurance people. Responsibility remains with the quality assurance group, but CM can capture and provide much of the data they need to evaluate the quality. To protect certain or all configuration items in the repository, CM can set up quality gates that will have to be passed before changes can be submitted. CM will always be heavily involved in measurements and metrics. CM responsibility covers all activities related to the repository and the change management process – which means that large parts of what happens on a project. CM must be capable of catching data and setting up any required metric. It is, however, the responsibility of others to decide exactly what metrics are needed.

3.3 Not Related to CM

In this sub-section we will give a few of examples of distributed challenges that we do not consider as being related to CM. More examples and extended analyses can be found in [3]. For the challenges in this category, the line or project manager will know that he should not bother the configuration manager for help, but look somewhere else. The configuration manager will know that the solution to these challenges lie outside of his competence/expertise and can politely tell the manager to go away.

Risk management [5]: Risk management can be interpreted as many things and some people claim that part of what goes on in the change management is handling of risk. We do not agree with that and see risk management as a much more formal thing that deals with analysis of critical paths and consequences of delays with respect to missing market windows. CM may supply a part of the data used as input for risk management, but has no part in it besides that.

Lack of coding standards [4]: The definition of coding standards is completely outside the scope of CM. The fact that checking for compliance with coding standards often happens through setting up quality gates when code is checked into the repository does not mean that it becomes related to CM. CM can, in general, set up check-in quality gates, but is not involved in the specific contents.

4. CONCLUSIONS

In a certain sense our initial suspicion was confirmed – distributed development does not necessarily have to be that particular. In fact, we found that quite a few of the challenges reported in literature are due to bad or missing CM. For these challenges there is an easy fix – as line or project manager, be aware that this should be the responsibility of the configuration manager and hold him accountable for it (and if you are a configuration manager, you now know what you will be held responsible for – so ask your boss for the resources and support in doing so). For many other challenges, they are not a direct responsibility of the configuration manager, but because the configuration manager sits on the repository where all the important assets of a project are kept, he can provide you (the manager or the developer) with just about any kind of information if it is already there or collect it if it is not. Ask him to do so if you experience such a challenge and be prepared to possibly pay for it. Finally, there is a limit to what configuration management principles and techniques can solve – and for such challenges you know that you will have to ask for help elsewhere than the configuration manager.

5. ACKNOWLEDGMENTS

We would like to thank Marc Girod, Ericsson, Ireland, Ulf Steen ABB, Sweden and Torben Poulsen, Thy:data, Denmark for comments, ideas and provision of cases.

6. REFERENCES

[1] Babich, W. A. *Software Configuration Management – Coordination for Team Productivity*, Addison-Wesley Publishing Company, 1986.

[2] Bendix, L., Magnusson, J. and Pendleton, C. *Configuration Management Stories from the Distributed Software Development Trenches*, in Proceedings of the 7th International Conference on Global Software Engineering, Porto Alegre, Brazil, August 27-30, 2012.

[3] Bendix, L., Girod, M., Magnusson, J. and Pendleton, C. *Configuration Management in the Context of Distributed Software Development*, Technical report, Department of Computer Science, Lund University, Sweden, forthcoming.

[4] Fauzi, S. S. M., Bannerman, P. L. and Staples, M. *Software Configuration Management: A Systematic Map*, in Proceedings of the 17th Asia Pacific Software Engineering Conference, Sydney, Australia, November 30 – December 3, 2010.

[5] Jiménez, M., Piattini, M. and Vizcaíno, A. *Challenges and Improvements in Distributed Software Development: A Systematic Review*, Advances in Software Engineering, Volume 2009, 2009.

[6] A. Leon, A. *Software Configuration Management Handbook*, (second edition), Artech House, 2005.

[7] Pilatti, L., Audy, J. and Prikladnicki, R. *Software Configuration Management over a Global Software Development Environment: Lessons Learned from a Case Study*, in Proceedings of the International Workshop on Global software development for the practitioner, Shanghai, China, May 23, 2006.

[8] da Silva, F. Q. B., Costa, C., França, A. C. C., and Prikladnicki, R. *Challenges and Solutions in Distributed Software Development Project Management: A Systematic Literature Review*, in Proceedings of the 5th International Conference on Global Software Engineering, Princeton, New Jersey, August 23-26, 2010.