# Awareness, autocompletion and resource management for a cloud-based IDE

Project plan

Brian Bullins

supervised by
H.-Christian Estler
Martin Nordio
Bertrand Meyer

2012

# 1 Introduction

## 1.1 Motivation

Global software development (GSD) is a reality of today's software development. Companies cross the barriers produced by distance, cultural differences, and time zones looking for talented personnel. To succeed in this journey, a new set of challenging problems have to be solved. The challenges include developing software requirements specifications [7], API design [12], project management, and communication and collaboration challenges [5, 9].

The effects of GSD have been studied from different angles by researchers in the field [1, 9, 3, 6, 4]. Focusing on communication and the effect of time zones, Herbsleb et al. [5] and Espinosa et al. [2] tried to answer an interesting question: how distribution and time zones affect distributed software development, specially when the teams are distributed in several countries and continents. Nordio et al. [9], we studied the effect of time zones and locations on communication within distributed teams; they performed the study as part of the DOSE [11, 10] university course. Global software development has also been studied from a quality point of view. The goal is to identify whether the quality and productivity of distributed projects is comparable to the one produced in co-located projects.

Tools in distributed software development play a key role, however, they are rare. An example is Jazz: a tool for distributed software development that improves communication and collaboration between developers. Jazz does not support project management, so other tools such as Microsoft Project have to be used. Environments for software development, known as Integrated Development Environment (IDE), have been widely developed in the last decade; for example Eclipse and EiffelStudio. While there are tools to support software development and project management, these tools are not integrated in a common environment, and they do not satisfy all the requirements of distributed software development.

## 1.2 CloudStudio

The Integrated Development Environment is the software developer's central tool. IDEs have undergone considerable advances. While Internet development has benefitted from IDEs, the IDE has not benefitted from the Internet; it remains an essentially personal tool, requiring every member of a project to work on a different copy of the software under development and periodically to undergo a painful process of reconciliation.

CloudStudio [8] brings software development to the cloud. In recent years ever more human activities, from banking to text processing, have been "moved to the cloud". CloudStudio does the same for software engineering by introducing a new paradigm of software development, where all the products of a software project are shared in a common web-based repository.

Moving software development to the cloud is not just a matter of following general trends, but a response to critical software engineering needs, which current technology does not meet: supporting today's distributed developments, which often involve teams spread over many locations, and iterative development practices such as pair programming and online code reviews; maintaining compatibility between software elements developed by different team members; avoiding potentially catastrophic version incompatibility problems; drastically simplifying configuration management.

CloudStudio brings flexibility to several new facets of software development, most importantly configuration management (CM): to replace the traditional and painful update-modify-commit-reconcile cycle, CloudStudio tracks changes at every location in real time and displays only the selected users' changes in the integrated editor. The compiler and other tools are aware of the current user preferences, and target the version of the code coinciding with the current view. CloudStudio also integrates communication tools (a chat box and Skype), and includes a fully automated verification component, including both static (proof) and dynamic (testing) tools. This array of tightly integrated tools makes CloudStudio an innovative IDE, which can improve the quality and speed of projects involving distributed teams, and

support highly collaborative development practices.

## 2 Goals

The goals of this project is to extend the current implementation of CloudStudio improving its awareness system, and providing important IDE functionalities such as autocompletion and resource management. In the setup of CloudStudio, autocompletion has to extend the traditional techniques. For example, if a developer $a$ adds a new routine $r$ without performing a commit, CloudStudio should include or exclude this routine depending of the user settings. If developer $b$ decides to *monitor* and compile the changes of developer $a$, then the autocompletion should include the new feature. The GUI has to be implemented in a way that the developer is aware that this routine has not be committed, and a change in his/her settings would break the system.

The awareness system will extend the current functionalities of CloudStudio, which allows developer to see and compile the code of other developers. The new version of the awareness system would monitor the developers changes, and based on the developers *annotations* (preconditions, postconditions, invariants, and even code comments), the system would inform other developers about these changes.

## References

[1] E. Carmel. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[2] J. A. Espinosa, N. Nan, and E. Carmel. Do gradations of time zone separation make a difference in performance? a first laboratory study. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE 2007)*, pages 12–22. IEEE, Aug. 2007.

[3] H.-C. Estler, M. Nordio, C. A. Furia, B. Meyer, and J. Schneider. Agile vs. structured distributed software development: A case study, 2012. ETH technical report.

[4] J. Herbsleb and D. Moitra. Global software development. *Software, IEEE*, 18(2), 2001.

[5] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 319–328, New York, NY, USA, 2000. ACM.

[6] H. Holmstrom, E. O. Conchuir, P. J. Agerfalk, and B. Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In *Global Software Engineering, 2006. ICGSE '06. International Conference on*, pages 3 –11, 2006.

[7] B. Meyer. On formalism in specifications. *Software, IEEE*, 2(1):6 –26, 1985.

[8] M. Nordio, H.-C. Estler, C. A. Furia, and B. Meyer. Collaborative software development on the web, 2011. arXiv:1105.0768v3.

[9] M. Nordio, H.-C. Estler, B. Meyer, J. Tschannen, C. Ghezzi, and E. D. Nitto. How do distribution and time zones affect software development? a case study on communication. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE 2011)*. IEEE, 2011.

[10] M. Nordio, C. Ghezzi, B. Meyer, E. D. Nitto, G. Tamburrelli, J. Tschannen, N. Aguirre, and V. Kulkarni. Teaching software engineering using globally distributed projects: the DOSE course. In *Collaborative Teaching of Globally Distributed Software Development - Community Building Workshop (CTGDSD)*, New York, USA, 2011. ACM.

[11] M. Nordio, R. Mitin, and B. Meyer. Advanced hands-on training for distributed and outsourced software engineering. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE)*, 2010.

[12] M. Nordio, R. Mitin, B. Meyer, C. Ghezzi, E. D. Nitto, and G. Tamburrelli. The role of contracts in distributed development. In *Proceedings of Software Engineering Approaches for Offshore and Outsourced Development*, 2009.