ABSTRACT
It is important to identify and act upon coordination problems within and across distributed software development teams in the context of global software development (GSD). Such problems jeopardize the quality of the system under development, and their identification is paramount to mitigate quality related risks. In this paper, we introduce a visualization tool, namely *error-handling graph*, that shows the interactions of development teams in terms of the reported and fixed errors during software development, and guides project managers to focus their efforts towards locating root problem causes. We present an example based on the data collected from the issue repository of an industrial project and discuss error-handling graph’s potential implications. The initial feedback from the piloting industrial team is positive, though a thorough evaluation is still needed. In practice, error graphs could be helpful in identifying the sources of problems and the opportunities for improvement, by creating awareness through data-driven insights for enhancing the way of working within and across software development teams.

Author Keywords
Global software development; distributed teams; communication; coordination; dependency; visualization.

ACM Classification Keywords
Information systems~Decision support systems • Software and its engineering~Programming teams

INTRODUCTION
Coordination of software development activities and resource management are significant challenges, especially within multi-site and large-scale project contexts [2, 6]. One important issue is the recognition and management of dependencies between the parts of software that is to be developed by different teams, and correspondingly tools, approaches and principles have been developed for the purpose, like configuration management and issue-tracking systems. It is obvious that the dependencies in the object of work will lead to communication and cooperation issues between the people involved, and this has led to several attempts to facilitate these issues by means of support tools. Probably best known among such tools is ARIADNE developed by de Souza et al. [8] where the dependencies analyzed in a software system are reflected against authorship information in configuration management repository to produce graphs that suggest social dependencies between the members of the development team. Another dependency graph tool that can be mentioned is SYSIPHUS [7].

This paper discusses about an idea of a potentially useful dependency-tracing graph tool with some similarity to the ideas behind ARIADNE. It is a spin-off of a project in an industrial setting [6] attempting to find ways to help development coordination from software analytics perspective with a focus on predictive analysis of errors. While the goal of software analytics is to propose actionable changes to the way the projects are run [1], our study revealed that the way the results are communicated to practitioners should indeed take precedence over the accuracy of the results in order to take actions [6]. The feedback from practitioners indicated that performance figures were not useful to have an impact in their daily work and pointing out error-prone sections within files was regarded as stating the obvious. Therefore, we were motivated to shift our perspective from technical details of software analytics to computer supported cooperative work (CSCW) for enabling technology transfer to industry.

Consequently, we developed multiple ways of communicating our analysis to practitioners and *error-handling graphs*, which visualize the interactions among teams based on the errors introduced and fixed, turned out to be the most helpful representation as it helped pinpointing communication related issues within and across teams. This paper makes use of edited text from our previous study [6] and extends it by exploring error-handling graphs from CSCW perspective [5], as visualization tools that support coordination in GSD.

Tracing Latent Dependencies Across Software Teams Through Error-handling Graphs

Burak Turhan  
University of Oulu  
90014, Oulu, Finland  
burak.turhan@oulu.fi

Kari Kuutti  
University of Oulu  
90014, Oulu, Finland  
kari.kuutti@oulu.fi
DESCRIPTION OF THE ERROR-HANDLING GRAPH
An error-handling graph shows the interactions among software development teams in terms of errors that are reported and fixed. The errors are linked to the teams that have reported or fixed the error. Figure 1 shows a sample graph based on the pilot project data (please see next section for details on the pilot).

There are two types of nodes in Figure 1. The nodes with text labels correspond to the teams involved in the project. Text labels start either with the prefix DevTeam or TestTeam, meaning that the corresponding team is a development or testing team, respectively. The nodes between the text labeled nodes represents errors extracted from the issue management tool, and they are color coded by the level of testing at which they are detected. A green edge from a team node to an error node means that the corresponding team has fixed the reported error. On the other hand, a red edge means that the corresponding team has reported that error. Finally, blue lines represent the errors that are reported and fixed by the same team [6].

In an ideal scenario, one would expect to see patterns in an error-handling graph, where all errors are: (a) either reported by a testing team and fixed by the responsible development team; (b) or reported and fixed by the same development team. Moreover, an ideal scenario would assume that there would be no issues originating from the interactions among the teams, i.e. an anti-pattern. Such errors are likely to have been missed in team level testing, propagated to higher levels causing re-work, and should have been detected earlier within the team.

DISCUSSION OF THE PILOT CASE
In this section, we first present the context of the pilot case study and then discuss how the error-handling graph can be interpreted to provide actionable insights to practitioners.

Context of the Pilot Case
The data used to construct the sample error-handling graph in this paper come from the software component of a mission-critical embedded system project developed by Elektrobit Wireless Segment in Finland. The software component is implemented in C++ language with over
100,000 lines of code. The development activities are carried out in two distinct geographical locations by multiple development and testing teams, adding up to about 60 developers in total. Data collected from the issue management tool reflects the development history of the project spanning a period of two-years. In the study, a development team volunteered to pilot the developed software analytics models and modes of information representation, providing feedback through personal communication with the R&D team during project meetings [6].

**Interpretation and Discussion**

Analysis of Figure 1 in terms of (anti-) patterns reveals the following examples:

- **Pattern (a):** It can be seen that the **testing** team named ‘Aarne’ is reporting most of the errors, e.g. the density of the red edges are very high around that team’s node.
- **Pattern (b):** Development teams named ‘Oili’ and ‘Niilo’ are following the good practice of transparency by sharing the errors discovered and fixed within the team with the rest of the project teams.
- **Anti-pattern:** There exists a cluster of errors between the development teams ‘Terttu’ and ‘Kristiina’. This indicates that the development works of the teams are most likely to be dependent on each other and errors propagate out of either team’s internal quality assurance process, impacting the work of the other.

Based on the initial findings listed above, the following actionable insights can be taken into account:

- Depending on the amount of errors detected by the testing teams in each testing level, an analysis on the nature of errors can be conducted to understand which level of testing needs to be improved (checking color codes in the graph) in order to identify the errors earlier in the process.
- All development teams can be encouraged to report the errors found within their internal processes in the global issue repository in order to improve the transparency of the development process. This would enable the collaborating teams to share insights about the problems encountered in the shared part of the code.
- Internal quality processes of the teams can be improved, especially when errors are propagating across development teams. This would save time and effort spent on re-work activities. In addition, the communication and coordination practices between such teams can be improved in order to reduce the number of errors reported across teams.

To elaborate more on the last item: there are two potential reasons for the conflict between ‘Terttu’ and ‘Kristiina’ teams. One reason could be that the teams’ way of working may be lacking quality gates for checking how their changes affect the work of others. Alternatively, there are latent dependencies between the parts the teams are concurrently working on. Such dependencies may have not been noticed and flagged when the division of labor between the teams is planned. So, the error-handling graph can in principle indicate problems in two levels: programming and testing practices/tools inside the teams, and in planning and dependency identification practices/tools. Further, the nature of those latent dependencies may not necessarily be code-based. In other words, code-based dependencies may just be a symptom of dependencies not detected at higher levels of abstraction, e.g. conflicting requirements from different stakeholders that are missed, or changes in requirements that were not communicated to the both teams properly. In any case, the root cause is a communication/coordination related issue that can be further worked on, yet can easily be identified by a visual inspection of the error-handling graph.

**CONCLUSION**

In this paper, we introduced error-handling graphs, which are potentially helpful in visually identifying error leakage across teams, as well as the need for better interaction/communication across teams that introduce and fix those errors. An error-handling graph shows the communication channels through the error documentation process. Accordingly, it is expected to help in identifying and handling the bottlenecks of project communication with respect to quality in terms of errors.

In practice, error-handling graphs could be helpful CSCW tools in identifying the sources of problems and the opportunities for improvement, by creating awareness [3, 4] through data-driven insights for enhancing the way of working within and across software development teams. We have demonstrated our arguments with a pilot study conducted within a multinational organization in Finland, where “the global is in the local” [9], since the development activities are: (i) distributed in two geographical sites (ii) and relevant from GSD perspective as they depend on “… coordination between local teams and their local practices” [9].

A formal evaluation of the tool is still pending and our future work will focus on contacting the teams of the pilot project, in which (anti-) patterns are observed, in order to gain more insights about the arguments discussed in this paper. Specifically, communication issues will be investigated by focusing on the errors between the teams, as they are likely be the key errors that slow down the development process by introducing re-work. Nevertheless, the initial feedback for the tool has been positive by the piloting team. Team members stated that the error-handling graph gives an interesting point of view into the project and holds a lot of implicit information.

**ACKNOWLEDGMENTS**

This research is supported in part by the Finnish Funding Agency for Technology and Innovation (TEKES) under Cloud Software Program, and the Academy of Finland with
Grant Decision No. 260871. We would like to thank Jari Partanen, Timo Räty, Jesse Pasuri and Taneli Taipale for their invaluable feedback and ideas for making this research possible in Elektrobit. We would also like to acknowledge Taneli Taipale for his implementation of the prototype of the tool presented.

REFERENCES
2. Smite, D. and Turhan, B. Validity of research on large-scale agile projects. Workshop on Research Challenges in Large-Scale Agile Development (XP 2013 Workshop), 2013.