Teaching Global Software Development through Game Design

John Noll
Lero, the Irish Software Research Centre
University of Limerick, Ireland
John.Noll@lero.ie

Andrew Butterfield
Trinity College Dublin
Ireland
Andrew.Butterfield@scss.tcd.ie

***Abstract*—In order to be prepared for careers in todays global economy, software engineering students need to understand the issues, methods, and practices associated with Global Software Development (GSD). One approach to teaching GSD is to conduct a GSD project class involving student teams from different institutions in different countries. This approach has the advantage of giving students first-hand experience with the barriers to collaboration and other issues faced by software development teams engaged in GSD. However, this approach is resource-intensive and requires cooperation among institutions.**

This paper presents an alternate approach based on game design, where students learn GSD concepts by developing a GSD simulation game. Following this approach, students learn about GSD through implementing a game engine that simulates the effects of global distance on a distributed software project. The experience shows that students seem to grasp the concepts and issues as a side effect of implementing the game.

***Index Terms*—Global Software Development; Software Engineer; Empirical Software Engineering**

I. Introduction

Global Software Development is increasingly becoming the norm for companies developing software. Even small teams can be highly distributed, with multiple team members distributed across national, timezone, and cultural boundaries [9]. As such, to be prepared to contribute in a globally distributed context, software engineering students need to understand the issues, methods, and practices associated with GSD.

One approach to teaching Global Software Development is to recreate the GSD context by conducting a GSD class involving student teams from different institutions in different countries [5]. This approach has the advantage of giving students first-hand experience with the barriers to collaboration and other issues faced by software development teams engaged in GSD. However, this approach is resource-intensive and requires cooperation among institutions [5]. Also, it usually requires that the entire term be devoted to the GSD project [5].

This paper presents a novel approach to teaching Global Software Development through *game design*. In this approach, rather than teaching Global Software Development through first-hand experience in a GSD project, students are introduced to the issues associated with GSD by designing a GSD simulation game that uses knowledge about Global Software Development to shape the game play. Students in a final-year software design class were tasked with developing a Global Software Development game (*GSD Sim* [10]) that casts the player in the role of project manager of a Global Software Development project, who must distribute tasks among multiple teams around the world, and cope with problems such as schedule slips, misunderstood requirements, integration failures, etc. The game engine simulates these negative events using a probability calculation proposed by Avritzer and colleagues [1]. Consequently, in order to implement the simulation engine correctly, students must understand the probability calculation, and in turn the values used to parameterize the calculation [8]. As such, students learn about Global Software Development as a side effect of developing the game.

The GSD Sim game has been a popular component of public events in Ireland such as the "Celebrate Science" event (http://celebratescience.eu/) and the Lero Research Centre launch (http://lero.ie/lerolaunch2015/). The success of the game itself is evidence that the developers understood GSD issues well enough to create a credible simulated GSD experience.

A software engineering class involving multiple teams in different locations is likely the best way for students to learn how to deal with Global Software Development. But when resources or time prohibit this approach, developing a GSD simulation game can introduce the issues without the overhead involved in actual Global Software Development.

The rest of this paper is organized as follows: in Section II we introduce the background to the problem, and define our research questions. Section III describes the method used. In Section IV we present the results, and discuss their implications and limitations; Section V presents conclusions and future directions.

II. Background

One of the most commonly reported approaches to Global Software Development education is to give students first-hand knowledge GSD by conducting a GSD class in cooperation with other institutions around the world [5]. This approach has the advantage that students get to experience the difficulties of GSD directly, by attempting a GSD project. One of the most ambitious examples of this approach was the DOSE project, that involved twelve institutions in eleven countries across three continents [11]; the authors note that, in addition to the overhead of setting up and running the course, *coupling* between teams at different locations was an issue, in that the

TABLE I
GSD SIM FEATURE LIST.

Name	Description
Master config	Master configuration (file) that specifies certain global values
Process simulator	Process simulator that calculates progress on each task for each module for each simulated day in the game.
Status display	Map-based status display showing which sites are making normal progress, which are behind, which are failing.
Default scenarios	Default game scenarios including pre-specified product and site configuration.
End of game report	End of game report comparing estimates to actual performance; report can be saved.
Nominal schedule calculator	The "nominal schedule" is just the sum of all the efforts estimated for each module, divided by a default developer-period effort value.
Game score calc.	Calculate game score as a function of budget and revenue.
Module Completion calc.	Module-task completion calculator that determines how much effort each task for each module actually takes, based on random 25% variation.
Problem simulator	Problem simulator that occasionally selects a site or module to experience a problem, with probability determined by game parameters.
Intervention interface	Intervention interface that allows the player to spend resources on interventions to correct or prevent problems.
Waterfall Process simulator	Process simulator enhanced to include *waterfall* development method.
Module location interface	Interface for specifying where module tasks will be carried out.
Clickable map	Clickable map for specifying location and size of distributed teams, and location of "home" site. Only major cities need by selectable (we won't allow putting a site in the middle of the Sahara, for example).
Culture-influenced reporting	Culture-influenced reporting that causes a site's status to be displayed according to the site's culture.
Inquiry interface	Interface to submit inquiries to a site about their progress.
Module specification interface	Interface for specifying modules and subsystems, the estimate of effort required to complete each module, and the expected monthly revenue when the product is released.
FTS simulator	Follow-the-sun simulator that takes into account the "hand-off" time for each module at the end of the day, until the module is completed.
Save-resume	Save-state feature to stop simulation and save the game state, for resumption later.
Real-time mode	Real-time mode that requires player to make interventions without stopping the clock.
Master config editor	Master configuration editing interface that allows the player or admin to change game master parameters without editing the file directly.
Problem KB	Knowledge base of known problems and solutions derived from the literature.
User accounts	User accounts that allow game state, game parameters, and results to be saved under a specific user's identity.
Budget by difficulty	Game budget calculated according to desired game difficulty mode (novice, advanced, expert).
Enhanced problem set	Enhanced problem set that included external events such as sudden release of competitive product (impact on revenue) or natural disasters (severe delays result).
Enhanced intervention set	Enhanced intervention set that allows hiring and firing developers.
Internal cultures	Specify different internal cultures for different sites at project setup.
Sub-tasks	Allow tasks to be subdivided and handed-off to different sites.
Module dependencies	Enhanced product specification incorporating dependencies among modules.
Critical path display	Enhanced product schedule display showing critical path in PERT or GANTT chart fashion.

success of a given team depends on their remote colleagues as well as their own work; thus, keeping the remote teams engaged was seen to be important.

A variation on this approach has co-located student teams engaging with a remote customer [2, 6]. This approach has the advantage of not requiring coordination across multiple institutions while still giving students experience with communication barriers introduced by global distance. This approach still requires recruitment of industrial partners to serve as customers, which adds administrative overhead [3, 4].

Another approach uses simulation to give students the GSD experience without the administrative challenges of organizing a project-based course involving multiple institutions. In fact, this is the intent of the GSD Sim game itself: to simulate aspects of Global Software Development to give the player a sense of the issues, without requiring an actual GSD project [10]. One advantage of simulation is that time can be compressed. For example, GSD Sim's engine simulates a full week every second; this means a player can experience an entire project in a matter of minutes.

Monasor and colleagues created a different kind of simulator designed to give students experience communicating with colleagues from different cultures; their VENTURE simulator uses chatbots to simulate a remote colleague, and "virtual guide" who helps the student interact in a culturally appropriate manner [7].

III. APPROACH

The GSD Sim game was developed as a class project for the fourth year Software Design module at Trinity College Dublin. Students formed groups of three to five developers, which were to implement the game in two "releases" comprising four one-week iterations; thus the total development time was nine weeks, including an initial "zero-velocity" release to allow groups to become familiar with their chosen development platform.

The class was given a list of "features" at the beginning of the term, which comprised the requirements for the game

TABLE II
PROBLEMS THAT CAN OCCUR AT A SITE.

Problem	Context where occurs	Impact	Delay
a site falls behind more than 25% on a task	Design	Repeat Design	15%
a site falls behind more than 25% on a task	Implementation	Repeat Implementation	15%
module fails unit tests	Unit Test	Go back to beginning of Implementation task	25%
module fails to integrate properly	Integration	Go back to beginning of Implementation task	40%
module fails system tests	System test	Go back to beginning of Integration task	55%
module fails to deploy correctly	Deployment	Go back to beginning of System test task	70%
module or product fails to meet real requirements	Acceptance test	Go back to beginning of Design task	100%
team reports progress inaccurately, according to cultural norms	any	If site is located in Russia or Asia, status will be green regardless	n/a

TABLE III
GLOBAL DISTANCE FACTORS

Qualitative measure of geographic distance	Value	D_{geo}
Same region (two hour drive)	Low	1
Less than three hour flight	Medium low	2
Transcontinental flight	Medium high	3
Intercontinental flight	High	4
Qualitative measure of temporal distance	**Value**	D_{time}
Transcontinental (five hour overlap)	Low	1
Intercontinental (three or more hour overlap)	Medium low	2
Global (less than three hour overlap)	Medium high	3
No overlap	High	4
Factor that contributes to cultural distance	**Value**	$d_j^{culture}$
Lack of a common language	High	4
Uneven language skills	Medium high	3
East/West divide in culture	Medium high	3
High versus Low context cultures	Medium high	3
Different national culture	Medium Low	2
Different organizational culture	Low	1

TABLE IV
INTERVENTIONS TO REDUCE GLOBAL DISTANCE.

Interventions that reduces geographic distance			i_j^{geo}
Intervention	Cost	Impact	
Face-to-face meetings	25,000	High	4
Exchange program	125,000	High	4
Synchronous communication possibilities	5,000	Med high	3
Support for video conference at all sites (depending on the project size)	5,000	Med low	2
Suitable selection of communication tools (multiple communication modes)	5,000	Med low	2
Interventions that reduces temporal distance			i_j^{time}
Intervention	Cost	Impact	
Relocate to adjacent time zone	500,000	High	4
Adopt Follow the Sun development	125,000	High	4
Create bridging team	500,000	Med high	3
Interventions that reduces cultural distance			$i_j^{culture}$
Intervention	Cost	Impact	
Face-to-face meeting	25,000	High	4
Cultural Training	25,000	Med high	3
Cultural Liaison/Ambassador	125,000	Med high	3
Adopt low-context communication style	5,000	Med low	2
Reduce interaction between teams from different cultures	5,000	Low	1

(see Table I). Each group was required to estimate the effort required to implement each feature; based on these estimates, a target set of features was set for each four-week release.

The majority of features are related to general aspects of the game, such as configuration and user interface features. Other features are about simulating software development in general; these include the *Process Simulator*, *Nominal Schedule Calculator*, *Game Score Calculator*, *Module Completion Calculator*, *Waterfall Process Simulator*, and *Module Specification Interface*. The implementation of these features would be largely the same if the game was simulating co-located development.

The GSD-specific features are indicated in italics in Table I; these are discussed in detail below.

A. Inquiry Interface

The Inquiry Interface provides a means for the player to inquire about the status of development at a given site. The response includes the status (on-time, late, finished, etc.) of each module under development at the site. Each inquiry has a cost, which is proportional to the effort required on the part of the site to which the inquiry is directed, ranging from 0 for an inquiry via email, to 7 developer-days for a site visit.

B. Culture-influenced Reporting

When the player inquires about the status of development, the response is filtered through cultural phenomena such as the "mum effect," which inhibits certain cultures from reporting bad news [12].

C. Problem Simulator and Knowledge Base

The Problem Simulator feature triggers problem events that the player must handle; the probability of a problem event occurring at a site is determined by the *global distance* of

that site from the home office. The requirements for this feature were based on a model proposed by Avritzer and colleagues [1].

Global Distance is a measure of how far away one site is from another. It has three components:

1) Geographic Distance, which is simply the actual separation between sites that prevents them from meeting face-to-face.
2) Temporal Distance, a measure of how different are the two site's time zones. Temporal Distance limits the times when synchronous meetings can be held.
3) Cultural Distance, which measures the difference in organizational, national, and regional culture that can affect how people interpret what is said.

Values for Geographic Distance (D_{geo}) and Temporal Distance (D_{time}) are found by looking up the separation in the upper and middle sections respectively of Table III.

Cultural Distance ($D_{culture}$) is somewhat more complicated, as multiple factors may contribute. Instead of a simple table lookup, Cultural Distance is computed by adding the distance values for each characteristic present between two sites:

$$D_{culture} = \sum_j d_j^{culture}$$

Values for $d_j^{culture}$ are shown in the lower section of Table III. All of the values in Table III are based on empirical data [8].

1) Global Distance and Probability of Failure: Given values for D_{geo}, D_{time}, and $D_{culture}$, Global Distance can now be computed as:

$$D_{global} = D_{geo} + D_{time} + D_{culture}$$

The probability of failure can then be obtained using the following formula:

$$P_{fail} = C(\frac{D_{global}}{1 + D_{global}})$$

This yields a value between 0 and 1 (C is a constant, to allow adjusting the probability to increase or decrease the game difficulty).

2) When does a site experience problems?: The problem simulator used the following procedure to determine when a site has a failure:

1) identify the "home" site; this is the site where the game player would be located.
2) for each site other than the home site,
   a) Compute D_{global} between that site and the home site.
   b) Compute P_{fail}
   c) Generate a random number between 0 and 1. If the number is less than or equal to P_{fail}, the site should experience a problem; select one from the following list that matches the current lifecycle phase (task):

When the impact is "Go back to beginning of… task," the effect is to add effort to the *actual* effort required, proportional to where the new task is in the lifecycle, in relation to the task where the problem occurred. So, for example, if the impact is "Go back to the beginning of Implementation task," the effect is to add the effort to redo Implementation, Unit test, and Integration to the actual effort (as calculated by Feature 8):

$$effort_{new} = effort_{actual}(1 + (.15 + .10 + .15)) \quad (1)$$

D. Intervention Interface

A player can buy various interventions (Table IV) to reduce the probability of failure; the cost is deducted from the player's budget.

Interventions reduce global distance, according to the following equation:

$$P_{fail} = C(\frac{D_{global}}{(1 + D_{global})(1 - I)})$$

Where

$$I = \frac{\sum_j i_j}{1 + \sum_j i_j}$$

The i_j are given in Table IV.

E. FTS Simulator

The FTS Simulator feature simulates "Follow the Sun" development by transferring work on a single module from one site to the next at the end of the work day. This proved to be challenging and no team implemented it successfully.

F. Internal Cultures

Internal cultures are organization-specific cultural factors, such as flat vs. hierarchical organization.

IV. Results

Five teams attempted to create a game with the features described in Table I. All teams managed to create games that demonstrated an understanding of the Global Software Development issues that inform the Problem Simulator. Two teams went beyond the stated requirements for the Problem Simulator, incorporating additional features accounting for morale at different sites. For example, the player can opt to require a site to work overtime in order to make up schedule slips; this will have a negative impact on morale, however, and thereby increase the probability of failure in the future. Conversely, a player can make certain interventions (including buying pizza for the remote team) that improve morale, and thereby productivity.

One of these allowed the player to choose a management style by choosing values for various manager characteristics, such as sensitivity, perception, charisma, intelligence, and assertiveness.

The other was of sufficient quality to serve as a "demonstrator" for GSD research conducted at Lero (see Fig. 1).

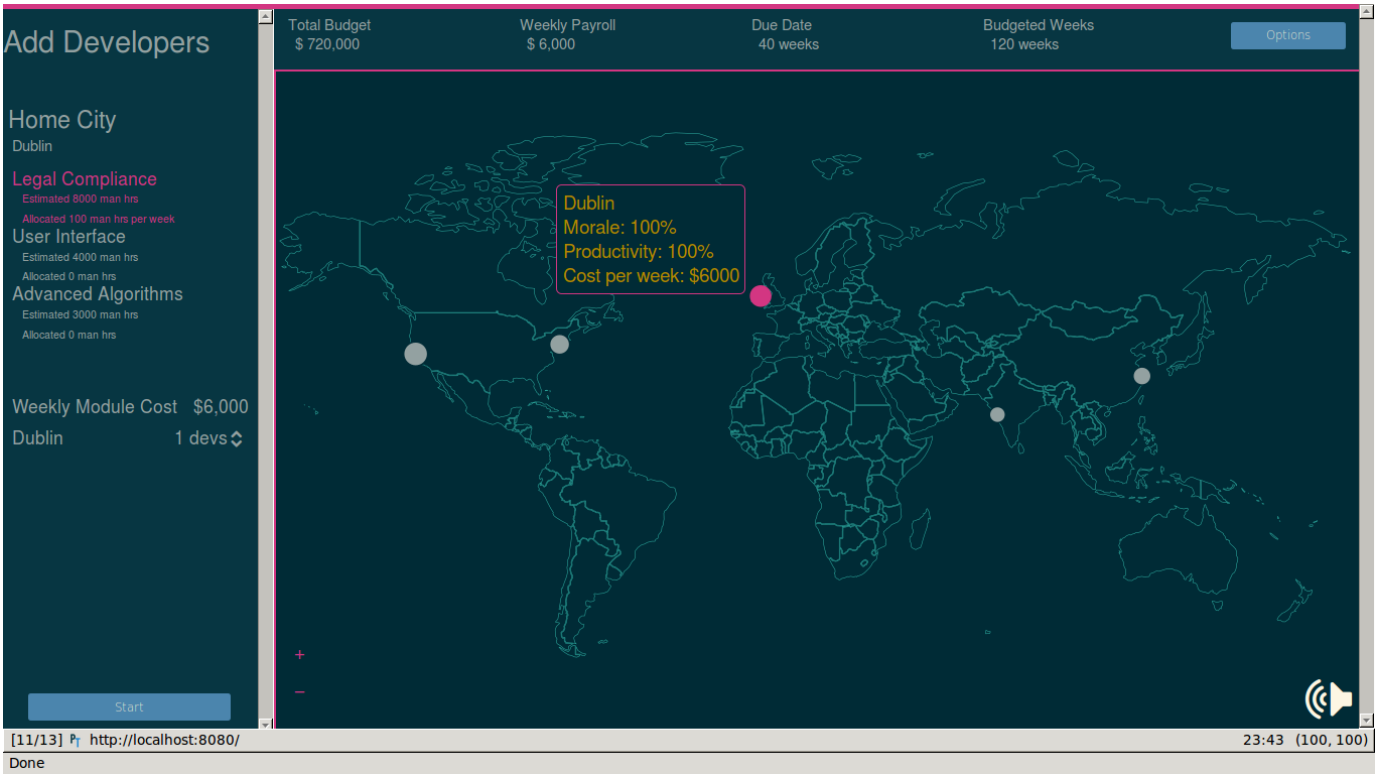

Fig. 1. GSD Sim screenshot.

V. Conclusions

A software engineering class where students are required to work on a significant project involving teams at other locations is one of most realistic ways to introduce students to the issues arising from Global Software Development, and the techniques used to deal with those issues. However, organizing and running such a class is a significant undertaking that requires a great deal of effort on the part of the instructor, over and above what is normally required for a project class. Also, GSD becomes the focus of such a class, which may not always be practical.

In situations where resources or curricula don't permit a multi-site class on Global Software Development, tasking students with creating a GSD simulation game can be a way to introduce GSD concepts without the overhead of a dedicated GSD project class.

Acknowledgments We would like to thank our students at Trinity College Dublin for their enthusiastic participation.

This work was supported, in part, by Enterprise Ireland and the European Development Fund through grant IR20130022, and by Science Foundation Ireland grants 10/CE/I1855 and 13/RC/2094 to Lero - the Irish Software Research Centre (www.lero.ie).

References

[1] Alberto Avritzer, Sarah Beecham, Josiane Kroll, Daniel Sadoc Menasché, John Noll, and Maria Paasivaara. Survivability models for global software engineering. In *Proceedings of the IEEE 9th International Conference on Global Software Engineering (ICGSE 2014)*, Shanghai, China, August 2014.

[2] Ivana Bosnić, Igor Čavrak, Martin Zagar, Rikard Land, and Ivica Crnković. Customers' role in teaching distributed software development. In *23rd IEEE Conference on Software Engineering Education and Training (CSEE&T 2010)*, pages 73–80. IEEE, 2010.

[3] Ivana Bosnić, Federico Ciccozzit, Igor Čavrak, Raffaela Mirandola, and Marin Orlic. Multi-dimensional assessment of risks in a distributed software development course. In *3rd International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD 2013)*, pages 6–10. IEEE, 2013.

[4] Bemd Bruegge, Allen H Dutoit, Rafael Kobylinski, and Günter Teubner. Transatlantic project courses in a university environment. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC 2000)*, pages 30–37. IEEE, 2000.

[5] Tony Clear, Sarah Beecham, John Barr, Matts Daniels, Roger McDermott, Michael Oudshoorn, Airina Savickaite, and John Noll. Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education – Working Group Reports (ITICSE-WGR '15)*, Vilnius, July 2015.

[6] Tony Clear and Mats Daniels. Using groupware for

international collaborative learning. In *30th Annual Frontiers in Education Conference (FIE 2000)*, volume 1, pages F1C–18. IEEE, 2000.

[7] Miguel J. Monasor, Aurora Vizcaíno, Mario Piattini, John Noll, and Sarah Beecham. Simulating global software development processes for use in education: A feasibility study. In *EuroSPI 2013*, Dundalk, Ireland, June 2013.

[8] John Noll and Sarah Beecham. Measuring global distance: A survey of distance factors and interventions. In *International SPICE Conference on Process Improvement and Capability Determination in Software, Systems Engineering and Service Management*, Dublin, Ireland, June 2016 (forthcoming).

[9] John Noll, Sarah Beecham, and Ita Richardson. Global teaming for global software development governance: A case study. In *International Conference on Global Software Engineering*, Irvine, California, USA, August 2016 (forthcoming).

[10] John Noll, Andrew Butterfield, Kevin Farrell, Tom Mason, Miles McGuire, and Ross McKinley. GSD Sim: A global software development game. In *Workshop on Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects (PARIS '14) (co-located with ICGSE 2014)*, Shanghai, China, August 2014.

[11] Martin Nordio, Carlo Ghezzi, Bertrand Meyer, Elisabetta Di Nitto, Giordano Tamburrelli, Julian Tschannen, Nazareno Aguirre, and Vidya Kulkarni. Teaching software engineering using globally distributed projects: the dose course. In *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, pages 36–40. ACM, 2011.

[12] Sakgasit Ramingwong and Sitthichok Snansieng. A survey on mum effect and its influencing factors. *Procedia Technology*, 9:618 – 626, 2013. CENTERIS 2013 - Conference on ENTERprise Information Systems / ProjMAN 2013 - International Conference on Project MANagement/ HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies.