Simulating Awareness in Global Software Engineering: A Comparative Analysis of Scrum and Agile Service Networks

Damian A. Tamburri, Ivan S. Razo-Zapata  
VU University Amsterdam  
Amsterdam, The Netherlands  
[d.a.tamburri, i.s.razozapata]@vu.nl

Héctor Fernández, Cédric Tedeschi  
IRISA. University of Rennes 1 / INRIA  
Rennes, France  
[hector.fernandez, cedric.tedeschi]@inria.fr

Abstract—Global software engineering (GSE) is a business strategy to realize a business idea (i.e., the development project) faster, through round-the-clock productivity. However, GSE creates a volatile and unstable process in which many actors interact together against unpredictable premises (e.g., cultural or time differences), often producing unexpected outcomes (e.g., compacting effects of distance and time). So far, Scrum has been used extensively for embarking in global software engineering, but many of the problems in Scrum-based GSE could still benefit from the usage of ad-hoc supporting tools (e.g., information continuity between timezones, cultural differences, developers awareness, etc.). Agile Service Networks (ASNs) are networks of service-oriented applications (nodes) that collaborate adaptively towards a common goal. ASNs offer a way to represent GSE professionals through service-oriented “social” nodes in a “small-world” network (much like a Facebook for a specific GSE project). This paper presents a comparison between the two approaches, namely Scrum and ASNs, to determine ASN’s potentials as mechanisms to maintain awareness in GSE.

Keywords—Global Software Engineering; Agile Service Networks; Chemical Engine;

I. INTRODUCTION

Global software engineering (GSE) is a business tactic to deliver a product faster by using round-the-clock productivity. GSE exponentially increases complexity of the effort and impact of bugs (e.g., identical bugs may be discovered in many locations without knowledge of discovery in other sites). Many compacting factors (e.g., changing requirements from changing stakeholders, dynamic and unpredictable environments, cultural interactions, information discontinuity between timezones, etc.) have been studied in literature to explain the exponential increase in complexity [20], [21], [14], [24]. One of these factors is that engineers are rarely made (explicitly) aware of overall project status [8]. So far Scrum [17] has been increasing in usage to embark on the GSE business tactic, as shown for example in [18]. Nevertheless, Scrum practices could benefit from the usage of ad-hoc support for awareness between GSE professionals [2], [22]. Agile Service Networks are dynamic networks of service-oriented applications (nodes) that collaborate together on a common practice or endeavor (edges) [5]. Many similarities bring together ASNs and GSE, e.g., (a) in both, the nodes (services in ASNs and teams in GSE) must collaborate to achieve goals, (b) both are generated through a business decision, since ASNs derive from (business-)service value networks [19], while GSE is a business tactic to maximize productivity (through round-the-clock production) and (c) both are dynamic organisms that react to context change (or need to, in the case of GSE).

Our hypothesis is that ASNs can be used to reduce latency times of “wicked-problems” (e.g., bugs, errors in documentation, mistaken design decisions, employee turnovers, etc.) by supporting status awareness among GSE professionals.

Our argument is that ASNs can be used to represent GSE developers in a project-specific awareness network (which is by definition a small-world network [25]). Through this network (which is social since it represents people and their relations), GSE professionals can automatically receive status updates from any of their peers in the project-specific ASN (so that, for example, relevant design issues can be discovered by one and immediately disclosed to many).

In this paper we sustain this argument, by modeling and analyzing prototypes for Scrum- and ASNs-supported awareness in a GSE “wicked-problem” scenario. Simulating the two prototypes and comparing results, we found that indeed ASNs show promise as awareness support mechanisms in GSE.

The rest of the article is structured as follows: Section II explains the related works; Section III introduces the GSE case scenario we consider to conduct our simulation, as well as the wicked-problem it exhibits; Section IV explains how the prototypes for both Scrum and ASNs were developed and simulated; Section V presents the results of our experiments and finally, Section VI concludes the paper, hinting to future work.

II. RELATED WORK

Our work in supporting communication in GSE (and more specifically, status awareness between professionals) is much similar to approaches reported in [22]. In this paper, the authors produce a series of approaches in GSE which have been proposed to support the 3C collaboration model (Communication, Coordination, Cooperation). More
in particular, three studies are very close to ours, for goals and intended uses. They are presented in [3], [10] and [15]. In [15], authors present the evaluation of a tool to timeline activities (collaboratively) so that different sites can coordinate their effort in a collaborative manner. It is similar to our approach since we argue that activity awareness is central in GSE. On the other hand, we are struggling to deliver a way to pro-actively disseminate activities and their schedules across sites, without the explicit intervention of GSE professionals.

In [10], the authors discuss an approach and a tool to collaboratively build calendars with activities, artifacts and people related together in a web of dependencies. Such a system is again a passive retrieval information system, following classical “publish-subscribe” patterns. Our approach focuses on ASNs, dynamic and pro-active service networks which can deliver relevant information (e.g. status updates, such as we are simulating in this work) to professionals without their explicit intervention or expressed need.

Finally, in [3], authors present a case study in which they discuss mechanisms to tackle cross-timezone coordination of teams, with different day-by-day routines, different temporal rhythms and working culture. This study provides evidence of the usefulness of pro-active and social-network based approaches to meet (dynamically) the daily (changing) needs of GSE professionals (e.g. awareness).

III. MATERIALS AND METHODS

A. Case Scenario

The case scenario we used was refined from [6], and inspired by a real-life situation. It can be summarized as follows:

The LOOSE (Large Overly-Occupied Software Engineering) company develops product Y, using 5 teams (of various specialty and members’ number) in 4 different time zones, namely: team 1 is in UTC -8, Los Angeles; team 2 is in UTC -5, Washington, DC; team 3 and 4 are in UTC +1, Berlin and Rome respectively; Team 5 is in UTC +5, New Delhi.

Product Y is under stringent time-to-market (rather than budget) constraints since it is a revolutionary product. Y cannot miss its delivery deadline. Before kick-off, management in LOOSE, decides that Y needs 8 tasks:

1) requirements engineering
2) requirements validation with stakeholder
3) service-oriented analysis and design
4) service development and testing
5) service integration
6) product quality estimation and monitoring
7) process monitoring
8) integration verification

To speed up production, all tasks are started at once, upon project kick-off at day D. As a consequence, timing is critical. We have used a value model to graphically represent the GSE business idea. The value model in Figure 1 describes not only the tasks to be performed and the teams in charge, but also the value exchanges among the tasks and their dependencies. In this way, we can have a better understanding of the contribution that each task brings to the whole business idea, i.e. what is the value of performing a given task.

Since a deep explanation about value modeling is out of the scope, we only describe the main concepts in Figure 1. At the top, a service bundle (the GSE solution) provides the required value objects to cope with the stakeholders need, i.e., the progress report, the quality estimation document and the integrated system solution, tested and verified against the requirements. At the bottom, the tasks 1 to 5 support the functioning of the tasks 6, 7 and 8. Making this distinction provide insights on which part of the GSE development network might be more sensitive to changes.

In order to provide a valuable GSE solution (both customers and teams meet their expectations), all the value exchanges in Figure 1 must be realized. Within the GSE context, Scrum approaches are used to achieve this objective. Nonetheless, we propose a novel approach based on Agile Service Networks that is explained in the following sections.

B. Wicked Problem

At day D + 21, an unavoidable technical constraint discovered by a programmer in team 5 causes design to be reworked and some requirements to be changed (updated). Designers (in teams 1 and 2) are immediately alerted through phone calls, but time-distance forces 1 day of overhead. At day D + 22, designers immediately negotiate an adjustment in requirements with system analysts from team 2 and modify their designs appropriately. Subsequently, at day D + 23, the same technical constraint is raised repeatedly by other programmers in team 4 who were not informed of the design changes (or were using designs that may have been unrelated to the adjusted design, but related to the affected requirements). At D + 24, team 4 is able to contact teams 1, 2 and 3 and is able to limit damages to an additional 1 day overhead.

The concern exposed in the scenario is that, quoting [6]:

“the participants must be kept aware of the development process, while striking a balance between providing information germane to their current work responsibilities and limiting extraneous, redundant information that overwhelms developers or desensitizes them to awareness mechanisms”

1For a deep explanation on value modelling the reader is referred to [12].
C. Scenario Implementations

The value network from Figure 1 represents value objects being exchanged in the above scenario. We prototyped the awareness mechanisms in this scenario in two flavors: one features regular Scrum teams, and the other features ASNs.

Scrum Alternative: The set of goals for each task is devised following Scrum guidelines (i.e. loosely dependent increments). Essentially, Scrum proposes agile and adaptable control. Intermediate milestones and frequent meetings in close time-zones, are planned so that all teams can rendezvous, evaluate and merge progress so far, as well as to plan next sprints. Communication tools used are commonly e-mails, dedicated forums, dedicated phone lines, corporate cell-phones as well as dedicated mailing lists for Scrum Masters, and micro-blogging (e.g. skype). No live status updates are shared. Scrum Masters are able to contact directly, through mailing lists and planned meetings. Team-specific awareness information (i.e. status updates of single Scrum members) is exchanged (locally) at daily debriefings. Overall project awareness is exchanged daily between Scrum Masters at planned end-shift debriefings. In the worst-case scenario (such as the one we are considering), the awareness information is disseminated in a quasi-hierarchical way, since commonly Scrum Masters are the ones which directly transmit awareness information between Scrum teams in different sites.

ASNs Alternative: Nodes in the ASN represent each developer. Developers working on the same task (or on related tasks) are linked by an edge and automated status updates are fed reciprocally. More specifically, our hypothesis is that this layout, in addition to status self-update, limits the awareness information only to the strictly needed information (i.e. only developers that need to receive status updates, actually do receive it), therefore complying to the concern expressed at the end of Section III-B. Also, the dynamicity and emergence mechanisms of ASNs make it possible to dynamically add services as needed [24], to reach developers in many emergency circumstances, e.g. when they are off-line, adding SMS messaging systems can increase the probability of reaching them.

IV. SOFTWARE PROTOTYPE

In this section, we discuss the implementation of software prototype for both the alternatives described in section III, namely ASNs-based and Scrum-based. Both prototypes are written in Java.
A. ASN-based Awareness Prototype

In order to accommodate the need for more dynamic and decentralized coordination mechanism in ASNs, we designed a fully decentralized P2P architecture in conjunction with a chemistry-inspired coordination model. Our design complies with ASNs characteristics [23], [24], as well as SOA principles presented in [13].

The ASN nodes represent the work-units within the network; their relations are defined by affinity (collaboration on same tasks and data dependencies). Likewise, the interactions among nodes are made possible by using a Gossip-based [16] message propagation mechanism: messages are transferred across neighbors until they reach the destination node. Such a communication mechanism allows to share the knowledge among the nodes in the network, as well as improving the status awareness of each node, as we detail in Section V.

On the other hand, the proposed decentralized architecture solves the scalability issue since there is no need for a single coordinator node: each ASN node is autonomous. To provide this autonomic behavior, we decided to use a chemistry-inspired coordination model, in which molecules of data react among them until a stable state is reached. In this chemical model, the information is seen as molecules moving across the nodes (membranes), and the computation as chemical reactions between these molecules. More precisely, we use a higher-order chemical language called HOCL [1]. HOCL is a rule-based execution language that considers the reaction rules as molecules, so that reactions can be applied on other reactions, in other words, programs modifying other programs, what gives the dynamicity of our model. Following the HOCL syntax, a program is a solution of molecules, formally a multiset of atoms, denoted \( A_1, A_2, \ldots, A_n \), "\( \)" being the associative and commutative operator of construction of compound molecules. Atoms can be constants (integers, booleans, etc.), reaction rules, tuples of \( n \) atoms, denoted \( A_1;A_2;\ldots;A_n \), or sub-solutions, denoted \( \langle M_i \rangle \), where \( M_i \) is the molecule content of the sub-solution. A reaction involves a reaction rule replace \( P \) by \( M \) if \( V \) and a molecule \( N \) satisfying the pattern \( P \) and the reaction condition \( V \). The reaction consumes the molecule \( N \), to produce a new molecule \( M \). This rule can react as long as some molecule satisfying the pattern \( P \) exists in the solution.

Algorithm 1

<table>
<thead>
<tr>
<th>Line</th>
<th>Rule Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>let createBug = replace bugId, description</td>
</tr>
<tr>
<td>02</td>
<td>by BUG:bugId:desc</td>
</tr>
<tr>
<td>03</td>
<td>let propagateBug = replace BUG:id:desc, BUG:TO:⟨ dest1, ..., destn ⟩</td>
</tr>
<tr>
<td>04</td>
<td>by transferMolecule(BUG:id:desc, dest1), BUG:TO:⟨ destn ⟩</td>
</tr>
<tr>
<td>05</td>
<td>let timezone = replace-one TIMEZONE:value</td>
</tr>
<tr>
<td>06</td>
<td>by TIMEZONE:getCurrentLocation()</td>
</tr>
</tbody>
</table>

The low layer of each engine is an HOCL interpreter based on the on-the-fly compilation of HOCL specifications. This HOCL interpreter processes the content of the multiset in which a set of rules expose the different services provided by each node. For instance, we defined a set of rules for the management of the project status information into the network. We now review two of these rules illustrated in Algorithm 1. The propagateBug rule is in charge of the propagation of molecules representing a wicked-problem which was detected in the system. When reacting, this rule consumes two molecules: a molecule \( \text{BUG:id:desc} \) giving information about a bug with an identifier and a short description of the problem, and another molecule with the form \( \text{BUG:TO:⟨ dest1, ..., destn ⟩} \) specifying the neighbours that have to receive this information. Rule createBug produces a bug molecule from an identifier and a short description of a wicked problem. Consequently, the resulting molecule will trigger the propagateBug rule.

This prototype has some similarities with the decentralized workflow management system presented in [9], in which the chemical coordination model was adopted for the execution of workflows in a decentralized way.

B. Scrum-based Awareness Prototype

A similar implementation, inspired by descriptions in [18], [17], [7] is used for the Scrum alternative. Two key factors make the Scrum alternative different to the ASN-based alternative: 1) propagation of status reports between Scrum teams in our “wicked-problem” scenario takes place daily, only between Scrum Leaders (e.g. the companies in the scenario are Small-Medium-Enterprises new to Scrum and GSE, therefore they do not know each other); 2) there exists working-time overlaps between teams during which, propagation time of status reports is identical to ASNs (since developers can access immediately to mails or forums, etc.). This prototype uses the similar rules for message propagation and another rule to obtain the current local time from the geo-location of each node. Thus, the timezone rule produces a molecule with the form \( \text{TIMEZONE:value} \) representing the current time-zone of a node. This molecule will be consumed during the propagation process to provoke a degradation to those teams with a different local time and no working-time overlapping.
V. RESULTS AND DISCUSSION

Our objective is here to show how the ASNs are able to augment the "error" awareness of a project in the sense that the project status information is perceived sooner by their participants than using the traditional Scrum systems. Thus, Scrum notification systems can cause delays for the perception of such information depending on its tree-like structure and different team locations on Earth.

Experiments were conducted over the French National Grid platform called Grid’5000 [4]. More specifically, these experiments were conducted on the parapide, paramount and paradent clusters, located in Rennes. The parapide cluster is composed of nodes equipped with two quad-core Intel Xeon X5570, 24 GB of RAM; the paramount cluster provides nodes with two quad-core Intel Xeon L5148 LV processors, 30 GB of RAM, and the paradent cluster is equipped with two quad-core Intel Xeon L5420 processors. All three clusters are furnished with 40GB InfiniBand Ethernet cards.

A. Scrum vs ASNs

In a first experiment, we measured the time spent to propagate a set of errors all over the two network prototypes, namely ASNs (which are small-world by definition) and Scrum (which creates a hierarchical message passing mechanisms between Scrum Masters).

- **Scrum - Hierarchical** organizes the different teams in a tree-like structure. This structure in conjunction with the different team locations can cause delays from 1 hour to 8 hours for the perception of a project status change. In such a way, three assumptions have been made during the execution of our experiment. First, a delay of one second is introduced, during the propagation process, per hour of difference and per transaction between teams linked and located in different timezones. Second, there is no delay when there are working-time overlaps between teams during the transaction. Third, teams located in the same region have zero delay.

- **ASNs - Small-world** organizes the teams depending valuable data exchanges between them. With this topology, the project status information is propagated to all the participants into the network, even if most of them are not neighbors. Similarly, small-world topology minimizes wiring costs and supports the dynamic nature of the ASNs (e.g. allows/eases adaptation or emergence mechanisms, etc.). Therefore, there is no delay depending on its structure or on team location.

Figure 3 plots the time of reception for an error notification, against the number of network nodes. In Figure 3, we can appreciate the high-awareness (i.e. lower time values) of a GSE system with a small-world topology (two curves in the bottom). This topology enables to propagate the errors to all the participants in a shorter period of time, while in systems with a hierarchical topology, the team organization and the mentioned delay produce a performance degradation which is independent of the quantity of errors to be propagated. Scrum Masters, in the tree-like organization, can provoke small bottlenecks to the whole system during the propagation. Moreover, we can make two observations.

On one hand, we show the acceptable behavior of both alternatives when scaling out and increasing the number of...
concurrent status-reports. This suggests that Scrum Masters, in the way they are set to communicate with the rest of the development network, constitute bottlenecks but allow handling part of the errors locally, rather than diffusing them network-wide. On the other hand, it is noticeable how ASNs’ time tends to increase exponentially when the number of nodes increase (it can be seen by confronting the slope of the curve at the bottom of the graph in Figure 3, and the one immediately above it). This suggests that the dynamity behind ASNs should be studied further to tune it for GSE. An additional simulation could be conducted to compare the two prototypes at the increase of concurrent nodes as well as concurrent notifications.

As a result however, our system represents a proof of concept in the sense that ASNs are a well-suited model for the management of awareness in GSE “wicked-problem” scenarios.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a comparative simulation of ASNs and Scrum, to understand ASNs’ feasibility as awareness support mechanisms in GSE “wicked-problem” scenarios. For this simulation we implemented two prototypes and evaluated them in a chemical-reaction based test engine.

On one hand, our results show that ASNs are indeed feasible to support awareness in GSE. ASNs dynamicity allows them to reach nodes, withstanding increased concurrent status notifications.

On the other hand, our results also uncover that ASNs may not be able to scale with the increase of concurrent nodes.

In this exploratory study, simulations are based on a number of assumptions and simplifications. In our future work, we plan to address more realistic scenarios, refining the ASN prototype with more pessimistic assumptions on ASNs as well as more optimistic assumptions on Scrum. We plan to perform a literature review on communication latency in software engineering using (un-)conventional means and use average values for ASNs. Also, we plan to gather data from industry by performing industrial interviews determining mean Scrum delays.

Moreover, while in this work we assumed absence of information overload and instantaneous error-detection, the wicked problem we simulated could include many variants, e.g. to analyze how ASNs behave with information overload or how they would help to discover errors.

Additional research is also needed in ASNs’ emergence and self-coordination mechanisms to understand if and how scalability problems can be circumvented. An ASN-based awareness mechanism should be tested in practice (e.g. through action research) to verify their practical uses beyond this simulation. To further understand ASNs’ benefits to GSE they could be used in combination to Scrum industrial practices.

Finally, since the value model which captures our scenario represents the business idea triggering GSE, future research can also focus on the role played by the technical implementation (e.g. ASNs) on providing meaningful information to allow adaptation in the value model, i.e. adapting the business idea. Finally, as shown in previous research, value modeling can also provide some insights to better understand the GSE context and dynamically adapt to it [11].

ACKNOWLEDGMENT

This work received partial funding from the European Community’s Seventh Programme FP7/2007- 2013, grant agreement 215483 (S-Cube). The authors would like to thank Prof. Patricia Lago for her contribution to the manuscript and in designing the study.

REFERENCES


