

Scaling Software

ITEA2 — SCALARE Project Newsletter No. 2 — June 2015

FEATURE ARTICLE

Scaling the Organization with Crowdsourcing

// Page 3

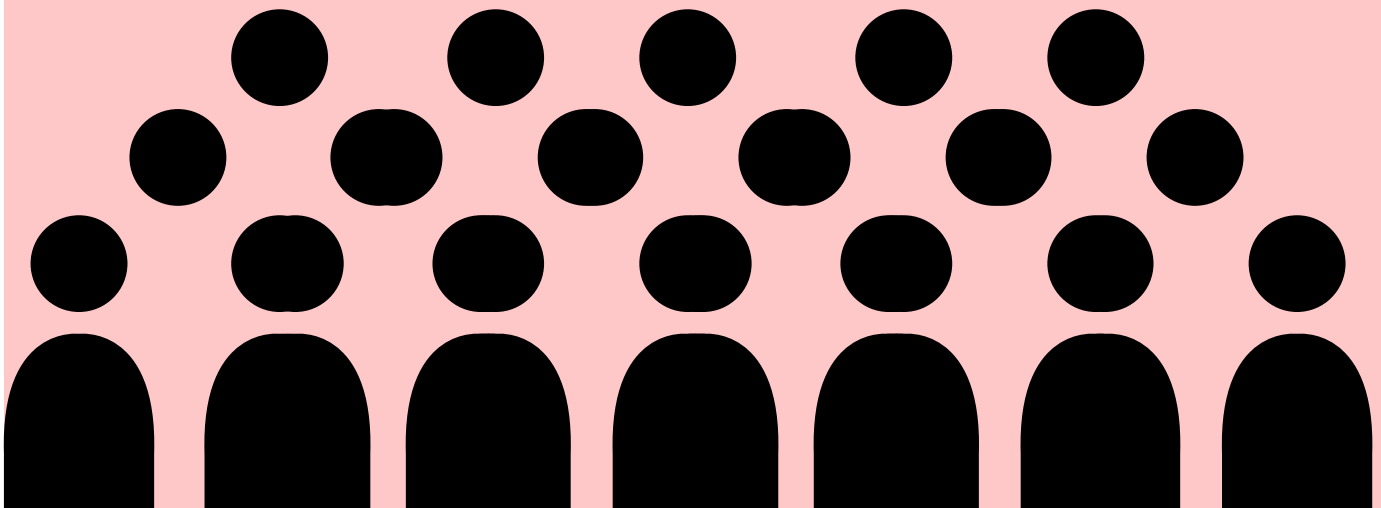


Image by Alex Kwa
from The Noun Project

- SCALARE News // Page 2
- Project Activities // Page 7
- Publications // Page 8

SCALARE News

New Project Leader for SCALARE

The SCALARE project has had a change of leadership. The project was initially led by Jesús Bermejo from Telvent, now part of Schneider Electric; however due to reorganizations within Telvent, Jesús was no longer able to provide his leadership. Since August 2014, the project leader is Miguel Ángel Oltra Rodríguez, also from Telvent.

SCALARE Passed First Review Meeting

The SCALARE project has had its first review meeting on 27 January 2015. The review meeting was held in the offices of Tieto in Stockholm. The review report was very constructive and the project passed. Based on the constructive feedback, the consortium held a strategy meeting after in order to define the activities for the next year.

New Logo for the SCALARE Project

The SCALARE project has a new logo. The new logo now features a

black-green color scheme and depicts the Pterophyllum Scalare, a fish first drawn during the Thayer Expedition, which was led by Louis Argassiz and financed by Nathaniel Thayer.

New Website for SCALARE

With an increasing number of activities and outputs in the SCALARE project, we have revamped the website, hosted at www.scalare.org. The new website now features more information about news, events, and publications.

SCALARE Project Meetings

The SCALARE consortium has had several meetings in the last few months. In January 2015 the project members came together for a strategy meeting to discuss next steps and activities for the months to come after the feedback from the first review by ITEA. In March 2015, several of the partners came together in Seville and met in the offices of Telvent, now part of Schneider Electric.

FROM THE EDITOR

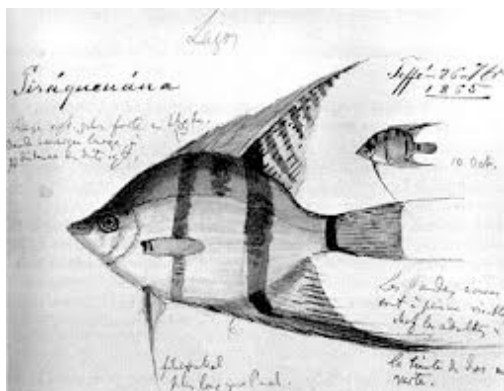
Welcome to the second issue of the Scaling Software Newsletter! We have seen many activities and new developments within the project since the last newsletter. Since August 2014, the project is led by Miguel Ángel Oltra Rodríguez, who has taken over from Jesús Bermejo. We are grateful for Jesús' initial leadership and thank him for his enthusiastic input in the first year. At the same time, we are very happy that Miguel is able to take over the leadership of the project.

In this issue we report on the latest news of the project, the new project website, and a feature article on one organization that scaled up through crowdsourcing. Enjoy!

Klaas-Jan Stol
Editor-in-Chief
Scaling Software Newsletter



Miguel Ángel Oltra Rodríguez, the new SCALARE Project Leader



The new logo resembles the Pterophyllum Scalare more closely

Scaling the Organization with Crowdsourcing

This article is an abridged version of the paper "Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development," by K. Stol and B. Fitzgerald, published in ICSE 2014.

Klaas-Jan Stol and Brian Fitzgerald, Lero, University of Limerick

Software engineering no longer takes place in small, isolated groups of developers, but increasingly takes place in organizations and communities involving many people. There is an increasing trend towards globalization with a focus on collaborative methods and infrastructure. One emerging approach to getting work done is crowdsourcing, a sourcing strategy that emerged in the 1990s. Driven by Web 2.0 technologies, organizations can tap into a workforce consisting of anyone with an Internet connection. Customers, or requesters, can advertise chunks of work, or tasks, on a crowdsourcing platform, where suppliers (i.e., individual workers) select those tasks that match their interests and abilities.

A number of potential benefits have been linked to the use of crowdsourcing in general, and these would also be applicable in the context of software development specifically:

- Cost reduction through lower development costs for developers in certain regions
- Faster time-to-market through accessing a critical mass of necessary technical talent who can achieve follow-the-sun development across time zones;
- Higher quality through broad participation: the ability to get access to a broad and deep pool of development talent who self-select on the basis that they have the necessary expertise;
- Creativity and open innovation: there are many examples of "wisdom of crowds" creativity whereby the variety of expertise available ensures that more creative solutions can be explored.

Crowdsourcing at TechPlatform Inc.

The application which TPI selected for crowdsourcing was Titan, a web application to be used by TPI field engineers when migrating from one platform to another as part of a customer engagement. Within TPI a technical decision was taken that future development should use HTML5, and this was the technology chosen for the front end, which was replacing the desktop application. The back-end services were based on a similar technology set used by the previous

desktop-based solution. Thus, TPI were keen to leverage HTML5 expertise from the large global TC community.

Similarly in the front-end, topics such as migration planning, importing and the scripting engine were retained for development by TPI. The two activities that are part of the TC crowdsourced development are asset modeling and automation testing. Modeling refers to the arrays and switches that need to be migrated and thus have to be modeled (i.e. created and configured) in the Titan application. Automation testing complements unit and integration testing which is designed by TPI developers, and refers to the testing designed by QA to test the front-end GUI interaction with the back-end.

Task Decomposition

The choice as to what parts of the product were appropriate for crowdsourcing was not entirely trivial for TPI. Code and executables which were self-contained would be easier to merge and hence were more suitable for crowdsourcing. However, if code from TC had to be directly merged with code being developed in-house, this would be more problematic. The decision as to what work to crowdsource was primarily based on internal resources (or lack thereof) and the amount of domain knowledge required for a certain task. Tasks that required the least amount of domain knowledge were deemed most suitable.

Crowdsourcing has worked particularly well for such tasks. Examples include tagging images.

TPI divided the project into five development phases. The first dashboard phase was the front-end which involved the high-level dashboard interface pages, e.g., for customer creation, project creation and navigation. The next two development phases involved configuration of TPI's flagship product. Following this, Phase 4 was concerned with the various network devices which also form part of the migration configuration. Finally, Phase 5 dealt with the low-end legacy products and various third party solutions that also need to be migrated. In order to minimize the modifications that would need to be made to the TC code after delivery, TPI made the header and footer browser code available to TC developers. This was to ensure this standard format would be maintained by all TC developers. For the Titan application, TPI's policy was to only use HTML5 where a feature was supported by all platforms to increase portability. Initially, there was an expectation that the TC community would deliver some innovative HTML5 code. However, the TPI requirement that HTML5 features would have to be supported by all browser platforms resulted in a very small proportion of all potential HTML5 features being available for use by TC developers. The expected innovation from the "crowd" was thus precluded by the TPI specification.

In order to minimize integration effort later on, the architect had wanted to let TC developers work against a real back-end core as opposed to stub services. However, by the time development with TC started, the core was not ready and stubs were used during most development contests. Consequently, this integration effort was pushed back to a later stage in the development process, which was not ideal.

For traditional in-house development, TPI developers had internalized a great deal of information in relation to coding standards and templates, and technical specifications. However, many of the coding standards and templates were documented informally and not stored centrally on the internal wiki installation. This scattering of information and URLs prevented it from being packaged as a deliverable for TC developers. A great deal of extra work was necessary to ensure that this information was made explicit in the requirements specification for the external TC developers. Most of the effort was related to the technical specifications.


Coordination and Communication

From the TC perspective, the software development process consists of a number of interrelated phases. While the TC process is essentially a waterfall one, an agile

development process, based on Scrum, was in use at TPI. Synthesizing these different development processes was problematic. TC development had to be assigned to a Scrum team within TPI, and TC contributions needed to be subsequently injected into the appropriate sprints. There were also quite a number of layers in the engagement model between TC and TPI. Firstly at the TC end, a co-pilot liaised between the TC developer community on the one hand, and TPI personnel on the other hand. Furthermore, a platform specialist and the TPI account manager were involved, effectively overseeing the co-pilot and recommending changes at that level. In this case, following some problems, a new co-pilot was selected with a tendency to be more proactive than his predecessor.

Within TPI, the choice of personnel to interact with the TC co-pilot was a difficult decision. While TC would prefer a single point of contact within the customer organization, there were significant management and technical issues involved, thus requiring senior people from TPI on both the management and technical end. A senior TC program manager was appointed specifically for all programs being developed with TC. This manager ensured that management were aware of any scheduling issues that could arise, for example, and also ensured that training was provided. However, there was also a specific Titan program manager, and thus there was inevitably some overlap between both roles. On the technical side, a senior architect was allocated to coordinate the TC development for the Titan project. This role of TC liaison which had daily contact with the TC community was considered to be problematic within TPI, given the considerable pressure to answer questions which was also very time consuming. There was some concern within TPI about allocating such a senior resource to this liaison role given the significant cost. At the initial stage, this liaison role involved answering questions on the TC Forums. There was significant time pressure involved since a time penalty applied if forum questions were not answered in a timely fashion by TPI, which would mean that the original committed delivery date for TC development would be pushed out. Also, the architect estimated the time answering questions on the TC Forums to be at least twice as long as would be the case with internal development.

In contrast to distributed development which typically involves other developers from the same organization, the only relationship which tended to build over time was that with the TC co-pilot. There was no real opportunity to build up a relationship with



A great deal of extra work was necessary to ensure that this information was made explicit in the requirements.

any of the TC developers, as interaction was filtered through a number of layers. Another structural coordination issue arose in that TPI allocate architects to products, and the desire to get the TC project completed resulted in two additional architects working on the project. This was seen as a sub-optimal resource allocation, given that the architect role was a somewhat scarce and extremely valuable resource.

TPI also had a so-called “tactical” Scrum team that could be assigned to different tasks more flexibly in that they were not formally assigned to projects on a long-term basis, as was the case with the normal Scrum teams at TPI. This tactical team could deal with TC contributions when they arrived. However, in some cases a normal Scrum team would also be assigned to the project, and in these cases involvement of the tactical Scrum team would not then be necessary. Overall, there was extra overhead and duplication of work on the project in that two teams had to become familiar with the project and deliverables. These two teams also had to communicate with each other. To address this issue, TPI dropped the use of the tactical team, and instead scheduled time in the project sprints to integrate the deliveries from TC.

Planning and Scheduling

The Titan project comprised more than fifty TC competitions. These competitions involved a total of 695 contest days, with an average length of competition of just over 13 days. The shortest completion time for a competition was 4 days while the longest competition took 32 days to complete. As discussed above, TPI had structured the overall development of the Titan product into five phases. The average duration across these development phases is 80 days, with the longest development duration (90 days) for the front-end HTML5 panels in the first phase, and the shortest development duration (69 days) for the final phase involving the low-end legacy and third-party arrays.

Some of the specific timings and the granularity of possible decisions for TC development were somewhat problematic for TPI. For example, TC allows a customer five days to accept or reject a deliverable. According to the architect, this was often not long enough to analyze and fully test the deliverable, and it was difficult to get these reviews done in time internally. A further difficulty arose in that TC deliverables must be accepted as a whole, or rejected as a whole, with no middle ground. It would be better from TPI’s point of view if more flexible granularity was possible in that certain parts of deliverables could be accepted and partial payment made for these acceptable parts. Because TPI did not want to deter TC developers from bidding on future

There was no real opportunity to build up a relationship with any of the TC developers

competitions, there was a tendency to accept code, even with some defects. There was an additional warranty period of 30 days, but integrating fixes under this warranty would pose considerable overhead in receiving, checking and integrating new code with an active code base which would more than likely have undergone significant further modification internally within TPI in the interim. Furthermore, when issues were escalated within the 30-day warranty, the resolutions were generally not satisfactory to TPI. Overall, a single longer initial acceptance period of 15 days would probably be more beneficial to TPI than the two current periods of five and 30 days, respectively. Another issue related to planning and scheduling arose when TPI had to wait for a contest to finish, while the main application was evolving, causing possible integration issues. TPI’s schedule was also jeopardized by several contests failing due to a lack of submissions. These contests had to be rescheduled thus causing a delay in TPI’s schedule. When rescheduled, there was only a single submission in one case, despite more than 30 registrants indicating an interest.

Quality Assurance

Much research in software engineering has focused on identifying and eliminating errors as early as possible in the development process, on the well established basis that errors cost exponentially more to rectify, the later they are found in the development cycle. However, the structure of the TC development process made it difficult to preserve this, as it shifted QA issues towards the back-end of the development process, after coding has been completed.

The number of defects identified was quite significant. While many issues were of a cosmetic nature, and therefore fairly trivial, the sheer volume of issues required considerable time and attention from developers within TPI. Furthermore, as more contests were finished and software delivered back to TPI, the rate of new issues was increasing as well. There was also a problem with lack of continuity. TC developers do not remain idle at the end of competitions, and may thus not be free to continue with TPI development in subsequent tasks. In fact, TPI experienced problems with bugs which had previously been identified being re-introduced to code after it went back for further development with TC. Partly this was due to how TC developers used the source code control tool. This added to the critical perception expressed by the Divisional CTO, when he contrasted it with the investment one would be prepared to make when using remote development teams for development, in describing crowdsourcing as being “a fleeting relationship.” Given that the combination of technical and specific domain expertise was

considered by TPI to be quite rare (based on experience in recruiting developers), TPI took some initiatives to improve the quality of crowdsourced contributions. For example, a virtual machine with a sample core application was made available as an image that could easily be downloaded and run. This was used by the TC development community both in development and as a final test or demonstrator for code they developed. Prior to this, TC code testing was done with stubbed-out service calls to the back-end, but there was a concern within TPI that TC code would not necessarily run smoothly when connected fully to the back-end. When the code for the initial HTML5 high-level panel applications was produced by TC, there were some quality issues, for instance, the same header was repeated in every file. TPI took this code and further developed it to a “Gold Standard,” at the level required by TPI. This was delivered back to the TC community as a template for future development. This tactic was extended to prepare sample code for a web application that could act as a template for the TC community. This included a parent project object model (build script), source code compliant with all TPI code standards, unit and integration tests, automation tests, and instructions for deployment and setup.

Knowledge and IP

The “fleeting relationship” mentioned earlier also has consequences for knowledge management and IP. According to the architect involved in the project, the lack of depth in the relationship with contestants meant that: *“there is a limited amount of carry-over knowledge. We will get a few contestants that will participate in multiple contests, but they won’t build up domain knowledge in the way that an internal person would.”*

Also, given that there is no single supplier as would be the case in a traditional outsourcing scenario, any intellectual property relating to specifications and product knowledge is more widely exposed simply by virtue of its being viewed by the ‘crowd’ of potential developers. Almost 90% of those registered for a contest did not actually submit anything to that contest. In other words, making detailed product

and specification information available, which is necessary to achieve the benefit of tapping into the crowd’s wisdom and creativity, seems (in this case) not to be as fruitful as one would hope given the limited numbers of submissions.

TPI chose a pseudonym to disguise their participation on the TC platform. This was to obfuscate the fact that the work was for the TPI platform as it was felt that developers from competing organizations might be working for TC in their spare time. TPI took advantage of the standard Competition Confidentiality Agreement (CCA) which TC use with their development community. TPI will not do business with certain countries, for example, and this can be policed through the CCA which identifies the home location of TC developers. TPI were still concerned about the extent to which proprietary information may be exposed in TC competitions. To address this, TPI plan to identify the “Secret Sauce” which should not be shared without very careful consideration. This would include the source code for the flagship and legacy applications, libraries and binaries from other TPI business units, performance calculation formulae, hardware specifications and business rules.

Motivation and Remuneration

Given a potential development community of a half million members, TC would claim to have broad and deep enough expertise to ensure a healthy competition rate. However, TPI have had to cancel some competitions because of a lack of participation and there had been a number of others with just a single contestant. The fact that TPI used a pseudonym does appear to be significant in that well known companies do attract TC developers more readily and TPI would certainly be a very well known company globally. The TC pricing structure was quite complex, At the top level, there was a monthly platform fee to TC. For TPI this was a monthly fee of \$30,000.

The co-pilot who was the principal liaison between TC and TPI typically cost \$600 per contest. There was an

initial specification review before the contest begins, and this cost \$50. The individual contest pricing was also quite complex. In the case of TPI, first prizes for contests ranged from \$200 up to \$2,400, depending on the size and complexity of a contest. A second prize of 50% of the first prize was paid to the runner up in each contest.

There was also a Reliability Bonus which was paid to the winning submission. The calculation of this bonus is quite detailed, but basically it can be up to 20% of the first prize, depending on the past successful track record of the winning contestant (i.e., his/her reliability – does a contestant actually submit after registering?). In addition, there was a cost of 45% of the first prize to support the TC Digital Run, an initiative whereby TC share money with the TC development community based on the monthly contest revenue and proportional to the number of points that TC developers have amassed in contests. The Digital Run is an additional mechanism to motivate potential contestants to participate even if they assess their chance of winning to be low. Following the contests, three reviewers from the TC community evaluated submissions and this cost approximately \$800 on average. Finally, TC charged a 100% commission equal to the total development costs above. Overall, the total average cost per competition so far was approximately \$6,200 (excluding the monthly platform fee).

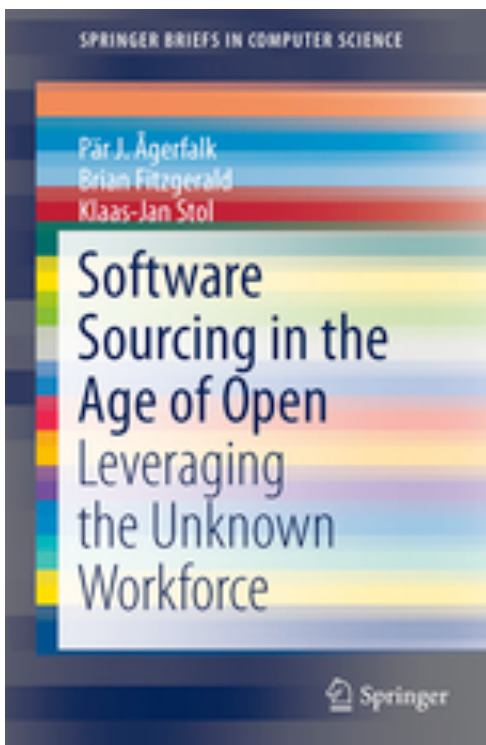
Dr Klaas-Jan Stol is a research fellow at Lero.
Contact him at klaas-jan.stol@lero.ie.

Prof. Brian Fitzgerald is Chief Scientist at Lero.
Contact him at bf@lero.ie

Project Activities

New Book Launched on Sourcing in the Age of Open

SCALARE researchers Prof. Brian Fitzgerald and Dr. Klaas-Jan Stol, together with Prof. Pär J. Ågerfalk from Uppsala University in Sweden have published a book on software sourcing in the age of open. The book is published in the Springer Briefs series, which are short books (up to 100 pages). The book focuses on new and emerging strategies to source software. In particular, the new strategies that are studied are crowdsourcing, opensourcing and innersourcing, all three of which are derived from the open source development paradigm. A sample chapter is available for download on the SCALARE website. The full book is available on Amazon.com and Springer.com.



SCALARE Project Featured at ITEA Co-Summit in Berling

The SCALARE project was presented at the Speakers Corner at the ITEA Co-Summit in Berlin, March 2015. Focus was to present our major result coming out of Scalare, the Scaling Management Framework (SMF). The SMF should provide guidance to SCALARE users to support them in their scaling scenarios. The guidance will come from case studies, patterns and experiences connected to different business drivers initiating a scaling scenario. At our presentation in the speakers corner we presented one case study from Work Package 2 and showed the audience how a future user could get guidance based on this case study as a way to exemplify SMF.

Case Studies and Developing the SMF

All SCALARE partners are currently actively working on reporting case studies. A Case Study in this context is defined as an exemplar scaling scenario based on real-world data. Sony Mobile, QUMAS and Husqvarna are all encountering the scaling phenomenon in their own ways, and the goal is to document these scenarios so that together they can be used as illustrations of the scaling management framework. The SMF itself is derived both from the literature.

To further demonstrate the SMF, the SCALARE consortium is also developing a demonstrator, which can serve as a decision making tool for organizations that need support in their scaling management. The demonstrator takes as input a number of key parameters based on which it will identify appropriate scaling strategies. We aim to have the demonstrator complete before the next review, which will take place in January 2016.

Publications

Book

P.J. Ågerfalk, B. Fitzgerald, K. Stol (2015) *Software Outsourcing in the Age of Open: Leveraging the Unknown Workforce*, Springer.

Journal Publications

K. Wnuk, J. Kabbedijk, S. Brinkkemper, B. Regnell and D. Callele (forthcoming) Exploring Factors Affecting Decision Outcome and Lead-time in Large-Scale Requirements Engineering, *Journal of Software Maintenance and Evolution*.

M. Michlmayr, B. Fitzgerald, and K. Stol (2015) Why and How Should Open Source Projects Adopt Time-Based Releases? *IEEE Software*, vol. 32(2). Special Issue on Release Engineering

K. Stol and B. Fitzgerald (2015) Inner Source—Adopting Open Source Development Practices within Organizations: A Tutorial, *IEEE Software*, vol. 32, no. 4

Conference Publications

P.J. Ågerfalk, B. Fitzgerald and K. Stol (2015) Not so Shore Anymore: The New Imperatives when Sourcing in the Age of Open, *Proceedings of the European Conference on Information Systems (ECIS) Münster, Germany*

B. Fitzgerald and K. Stol (2015) The Dos and Don'ts of Crowdsourcing Software Development, *Proceedings SOFSEM 2015, LNCS 8939*, pp. 58-64

Workshop Publications

K. Stol and B. Fitzgerald (2015) A Holistic Overview of Software Engineering Research Strategies, In *proceedings of the Third International Workshop on Conducting Empirical Studies in Industry (CESI) co-located with ICSE '15, Florence, Italy*.

Write for Us!

Contributions to the Scaling Software Newsletter can be sent to Klaas-Jan Stol (klaas-jan.stol@lero.ie). We welcome experience and lessons learned reports, and case study reports.