SCORE: the first student contest on software engineering

Dino Mandrioli[1], Stephen Fickas[2], Carlo A. Furia[3], Mehdi Jazayeri[4], Matteo Rossi[1], and Michal Young[2]

[1]Politecnico di Milano, Italy [2]University of Oregon, USA
[3]ETH Zurich, Switzerland [4]University of Lugano, Switzerland

Abstract

The Student Contest on Software Engineering (SCORE), organized for the first time in conjunction with the International Conference on Software Engineering (ICSE) 2009, attracted 50 student teams from around the world, produced an impressive and varied set of projects, and earned appreciative comments from participants and even from teams who chose not to submit their results to the competition. It was a remarkable success, but not without problems and setbacks. In this article we explain the objectives, constraints, and design philosophy of SCORE, particularly as they distinguish it from the tradition of computer science contests focused more narrowly on programming. We also recount key approaches taken to design and management of this novel kind of contest, the difficulties we met (some still outstanding), and the lessons learned.

Most of us are familiar with contests in which engineering students construct bridges from toothpicks, competing to see how many coins each bridge can support, or contests to build protective enclosures for eggs dropped from the upper floors of a building. Many have watched television documentaries of robot battles held at MIT, and thousands of schoolchildren compete each year in solving challenges with robots constructed from Lego building blocks [1]. These student contests inspire excitement, teach problem-solving skills, and communicate the nature of engineering disciplines. The Student Contest on Software Engineering (SCORE) [9], a world-wide competition for graduate and undergraduate college students, was likewise created to promote knowledge and appreciation of software engineering.

Many contests in the field of computing preceded SCORE. Among contests focused on the activity of programming, the annual ACM programming competition [2], which takes place over a period of a few hours, and the ICFP Programming Contest [3], which takes place over a period of a few days, are well known. A variety of project competitions, often industry sponsored, have also been established, such as Microsoft's Imagine Cup [5] and Google's Android Developer Challenge [6]. These are typically tied to particular development or delivery platforms, and at most only loosely tied to university curricula. While taking inspiration from these prior efforts, the SCORE organizers conceived of a contest that emphasizes the engineering aspects of software development, encompassing much more than programming alone, and including the challenges of a realistic, complete software project. They sought inclusiveness, with fair competition among students from all varieties of universities worldwide, and without limitation to a single application domain or development environment. Most of all, they sought to foster

among students and teachers an understanding and appreciation of the creative challenge of software engineering.

We regard the first offering of SCORE to have been quite successful, although not without challenges. This brief report of our experiences tries to distill the main lessons we learned; we believe they can be useful suggestions for those who may consider organizing similar contests, as well as for teachers of project-oriented courses in software engineering.

The present paper is structured as follows. In the next section we report on the whole process of SCORE, starting from the initial idea up to its conclusion in Vancouver with the selection of the winning teams, and the presentation of awards. The exposition follows a fairly "historical" style, but the focus is on the main problems we faced, the solutions we adopted, and the alternatives and risks we weighed. A *post mortem* analysis follows, reporting on the success we believe to have achieved and the critical aspects that need more examination. The last section summarizes the main lessons we learned and draws a few suggestions for future contest organizers and teachers.

The history of the first SCORE

The idea of a student competition on Software Engineering (SE) was originated in 2006 by Steve Fickas, the ICSE 2009 general chair, and Paola Inverardi, the ICSE 2009 program co-chair. It immediately found a set of enthusiastic volunteers to make it real by the Vancouver event. The team had roughly three years to accomplish the task.

As a first step, we defined a number of features that would distinguish SCORE, and justify its creation. First, and most importantly, the contest should challenge student teams through a *realistic and complete SE project*. Second, the contest should be inclusive and provide a fair chance to students from small and large schools. As far as possible, the rules should enable the teams to be evaluated, not their schools or programs.

Several consequences immediately derive from these requirements, most notably that the contest must have a *considerable duration*. Thus, contestants should have the chance to work off-line, rather than meet at a contest site to perform for a short period of time. Contestants could then be teams that originate from SE classes of any university. This further emphasized the main difference with respect to other competitions launched in the academic world, typically in the context of scientific conferences or during ad-hoc events (mostly algorithms and quick coding), but also the multi-day contests held by the functional programming or computer security communities [3, 4]. In addition, the challenge should touch all facets of software development, from requirements analysis and elicitation to implementation and verification.

To realize the desired features of the contest, several decisions regarding project selection, management, evaluation, and team composition had to be made. These decisions are discussed in the next sections.

Devising contest projects

Determining what kind of projects SCORE participants would pursue was probably the most controversial issue in the preparation of SCORE, since it involves some clear and critical trade-offs. The range of possibilities we evaluated was very wide: at one extreme, teams could have been left free to choose their own projects to develop; at the other end of the spectrum (as in many other competitions) all participants could be given the same project to develop. Both approaches have their pros and cons.

Giving teams free rein to devise their own projects would simplify some parts of contest organization. It could be attractive to some participants, imposing little overhead or deviation from existing practices at a participating school. But it entails risks to inclusiveness and even-handedness, favoring those who can exploit existing strong research projects or industrial ties to create impressive projects, and potentially turning SCORE into more of a school exhibition and less of a *student* contest.

The other extreme is to pose a single project for all participants. This would make evaluation simpler, but perhaps not fairer, as any single project tilts the competition toward some particular domain, technology, or development issue (e.g., real-time systems). Moreover, it is difficult to create a single project that is attractive to a wide array of student teams.

We decided on a balanced solution between the two extremes: The program committee (PC) defined *several* projects to choose among. The projects proposed by the SCORE PC collectively fulfilled the following major requirements: They had to be well-balanced in terms of application fields, covering topics ranging from embedded and critical systems to web-based services; they had to allow exploiting different development methodologies, so that no single SE approach (agile development, formal methods, etc.) was favored or disadvantaged. In keeping with the long-standing goal of the software engineering research community to engage with industry, we also wanted to offer a mix of academically-flavored and industrially-flavored projects..

Managing the projects

To comply with the above requirements, nine projects were defined and proposed by nine different members of the PC. They were posted on the SCORE site [7] and are listed in Table 1.

Project title	Project domain	PC member(s)/proponent(s)
Diogene (Digital I/O GENerator Engine)	Hardware control and simulation	Giovanni De Toni
Distributed Decision in a Mobile Context	Distributed voting system	Miguel Felder, Xiaoping Jia, and Stuart Faulk
BTW: if you go, my advice to you	Interactive trip planner	Stephen Fickas
Personnel Access Control System (PACS)	Access control system	Constance Heitmeyer
Global Studio Project (GSP)	Distributed software development	Daniel Paulish
Awareness Tool for Distributed Software Team	Distributed software development	Gina Venolia
Design Rationale Investigation Tool	Software design tool support	Gina Venolia
A Simple Pacemaker Implementation	Embedded device control	Alan Wassyng
GPXCleaner: GPS Path Editing and Simplification	Data visualization	Michal Young

Table 1. List of SCORE projects

We designated a *stakeholder* for each project in the context of SCORE. The stakeholder was intended to play the role of the customer, and typically was the project proponent him/herself; however, exceptions were allowed and did in fact occur. The rules of interaction between the stakeholder and the team were part of the project description, i.e., the rules could vary for each different project. In some cases teams were urged to find their own stakeholders in their environment; this was intended to mimic the situation where the project consists of developing a product to be commercialized in an open market, and there is no specific *a priori* customer. In such case, the project description just outlined the general theme and rationale of the project, whereas the requirements had to be gathered directly by the developers interacting with (presumably) real, potential customers.

Important dates, including publication of the projects, submission and delivery deadlines, were set to allow any university to encourage their students to participate during their academic year. Given the diversity of academic calendars throughout the globe, it was clear that a wide time window was necessary. In concrete terms, the requirements had to include students associated with university courses whether those courses were held in the spring or in the fall, in the northern or southern hemisphere.

From the point of view of the project management on the teams' side, we mandated that each team enact and document all typical aspects of a real SE project. Apart from this, we left much freedom in organizing the process, and we gave only a minimal set of guidelines, which are summarized in the following.

- Team size had to be between two and seven, though we suggested that teams be limited to no more than five members (with allowed exceptions).
- Geographically distributed teams were encouraged. In practice, some geographically distributed teams exceeded the suggested limit of five members.
- Teams were left free to choose not only the preferred project among the nine published, but also the preferred development approach (waterfall, agile, etc.),

provided they satisfied the major requisites stated by the project definition. This included not only requisites about the *product* to be delivered, but also, in some cases, about the process to be followed. For example, some project proponents explicitly asked for intermediate delivery of work in progress documents and/or for appropriate interaction with the stakeholder.

- Teams were free to emphasize particular aspects of SE, provided that they gave enough coverage to all relevant aspects. For instance, a team could devote particular attention to requirements elicitation and user interaction, provided that a running implementation that was suitably validated and verified was delivered; another team could instead emphasize validation and verification, maybe by using different complementary techniques, while somewhat reducing – but not skipping! – the early phases. Quality was explicitly recommended as a higher-priority goal over quantity.

- Given that it was impossible to state a fixed time to devote to project development (this depends on the school calendar), we decided to ask teams to document the effort done in terms of person-time and calendar time, and then to evaluate the value of the project also relative to the applied effort.

- Teachers and schools were allowed, and in fact encouraged, to provide support to teams in terms of motivation, integration with official SE courses, help in managing contacts with SCORE organization, and generic advice. However, participation of any member of the teaching staff or of some "external consultant" in the actual technical development of the project was clearly excluded.

- Teams were not allowed to select a project proposed by their teacher or a member of their own school for reasons of fairness and avoiding conflict of interest. This is in sharp contrast with normal practices of project assignment within SE classes, and may have discouraged potential participants, but we deemed it necessary for fairness of the competition.

No specific action was taken to enforce these rules, and their strict enforcement would have been difficult. Substantial good faith was assumed on the side of the participants, as is common and necessary in most scientific events. We do not have reason to suspect any abuse by any of the participants.

Targeting the contestants

The SCORE competition was limited to teams of undergraduate and master's students, possibly mixed. PhD students were ineligible, in consideration of the advantage they would have in research experience and length and depth of software engineering study. The evaluation process

took into account the composition of participating teams, and we describe below how this happened.

Much effort was spent to publicize the event, given its novelty. All the typical advertising means were exploited, though, as discussed later in the paper, this has been a critical issue.

Special tracks within SCORE

Even though one of the foremost goals of the competition was to cover all main aspects of SE, we were also open to give special emphasis to particular fields of the discipline. Thus, we launched the idea of special "tracks" and awards devoted to subfields of our discipline, even looking for communities willing to "sponsor" such special tracks and awards. In principle this could be applied to any particular field. In the case of SCORE 2009 we contacted the Formal Methods Europe (FME) group, which happily accepted the proposal, and sponsored a special prize for the team that best exploited formal methods in their project. A member of the FME group was enrolled in the PC to act as a co-chair for the special track.

Though we were open to organize other special tracks in addition to the one sponsored by the FME group, in this first edition only the one on formal methods has been realized.

Organizing the evaluation process

The scrutiny and evaluation of full SE projects is a formidable task, whether it occurs within an industrial environment, in the market, or in a classroom. The challenge for SCORE was increased by the diversity of projects as well as diversity in approaches. There could be no simple check-list or score-sheet for comparison of project submissions. And yet, criteria and a pragmatic process for evaluation arise quite naturally from considering the goal of SCORE to emphasize the engineering aspects of software development. While participating teams were free to choose among a wide range of methods, or to make their own, we expected a *reasoned* choice that participants could explain and justify. Likewise for other aspects of the project, freedom is not chaos; we expected participants to be able to explain the choices they made, the rationale for those choices, and the outcomes of their choices. Thus, in an attempt to combine freedom with efficiency and thoroughness, we approached the task of evaluation incrementally in a three-phase process, beginning not with the executable programs but with a written report.

Teams had to complete their project within the assigned deadline (January 15, 2009), but they were asked to deliver only a summary report initially. We did not impose a standard format for the summary report, though some suggestions were posted on the SCORE website [8]. The summary report – approximately twenty pages long – was intended to give an overall description of the whole developed project so as to allow the PC to perform a first global evaluation of the team submissions.

A preliminary filtering of the submissions was based on the summary report. It was conducted according to the typical reviewing process of scientific conferences. A number of reviewers were assigned to each team through a standard bidding process, except that one proponent or stakeholder of each project was asked to review all submissions of that project. Teams who passed the first filter were invited to submit the full documentation of their projects, including high-level specifications, architectural design, running code, installation and configuration instructions, validation and verification documentation, user manuals, etc., for a second round of evaluation.

Using summary reports for the first round of evaluation posed a risk of two possible undesirable outcomes: A poor project could be evaluated positively because of a well-written report, or a well-executed project could be judged negatively because of a poorly written report. We considered the second case to be unlikely, particularly considering the value of clearly written design rationale as an essential component of a software system that outlives its initial developers. In the first case, we assumed that the poorly executed project would be caught during the second evaluation phase.

All teams selected for the second round delivered the full documentation within the stated deadline. They also used the assigned time to comply with reviewers' requests, when these were part of the first evaluation. The second, more thorough evaluation included experimenting with the running systems. To further improve this critical phase, a second stakeholder was assigned to each team.

For both rounds of review, the evaluators did not use any detailed evaluation sheet. Instead, they were given general qualitative guidelines, in accordance with the overall philosophy of the contest: seek quality over quantity; require breadth but allow for different focus on a particular aspect or phase of the development. Evaluators produced a review text, a confidence score, and a summary acceptance grade (from strong reject to strong accept). The evaluators were allowed to modify their reviews in response to the discussion among themselves.

At the end of this second phase, the finalist teams were selected and invited to participate at ICSE. The conference organization substantially contributed to the travel and accommodation expenses for the invited teams. Table 2 lists the number of selected teams per project and evaluation phase.

Project	# registr.	# subm.	# selected	# ICSE finalists
Diogene (Digital I/O GENerator Engine)	1	0	0	0
Distributed Decision in a Mobile Context	12	8	4	2
BTW: if you go, my advice to you	15	6	3	2
Personnel Access Control System (PACS)	5	2	0	0
Global Studio Project (GSP)	1	0	0	0
Awareness Tool for Distributed Software Team	2	0	0	0
Design Rationale Investigation Tool	0	0	0	0
A Simple Pacemaker Implementation	5	4	1	1
GPXCleaner: GPS Path Editing and Simplification	9	6	2	1
Total	**50**	**26**	**10**	**6**

Table 2. For each project: number of registered teams (2nd column), received submissions (3rd column), teams accepted to the second round of reviews (4th column), and finalist teams invited to ICSE.

Each finalist team gave a formal presentation of their project during regular sessions of the conference. Furthermore, they presented demos open to all conference participants, and they showed a poster of their work during the conference poster session. A mentor – typically a former stakeholder – was assigned to each finalist team to help them organize their final presentation at the conference.

Finally, an intense PC meeting held during the conference chose the winners. Although we emphasized that all finalists were in some sense winners, and have been given an official certificate, we delivered a cup to the overall winning team (Figure 1), and two "special" cups: one for the team that best exploited formal methods and one for the best all-undergraduate team. Table 3 lists the members of the winning teams.

	Team members	Project
Overall winner	Nikola Tankovic, Sonja Milicic, Danijel Zovic (University of Zagreb, Croatia), Gianluigi Ciambriello, Savino Ordine, and Zafar Bhatti Ahmad (Mälardalen University, Sweden)	*BTW: if you go, my advice to you*
FME award	Valerio Panzica La Manna, Andrea Tommaso Bonanno, and Alfredo Motta (Politecnico di Milano, Italy)	*A Simple Pacemaker Implementation*
Best undergraduate team	Kaan Yucer, Mahmud Resid Cizmeci, Ilkay Ozan Kaya, Elif Akan, Fatma Ekici, and Ali Karasu (Bogaziçi University, Turkey)	*Distributed Decision in a Mobile Context*

Table 3. Winning teams.

Figure 1. Left to right: Steve Fickas, Gianluigi Ciambriello, Nikola Tankovic, Mehdi Jazayeri, Dino Mandrioli.

Post mortem analysis

In this section we analyze the achievements of this first edition of SCORE, and also the difficulties encountered along the way. Both our successes and challenges may provide useful guidance to organizers of other contests, as well as future editions of SCORE, and perhaps also to instructors who face related challenges in organizing project-oriented classes.

Successes

The quality of submitted projects, and particularly of the finalists, exceeded the most optimistic predictions of the organizers. The written documentation, the oral presentations, and the demos arranged during the conference were comparable to professional presentations. During oral presentations and demos at ICSE 2009, PC members posed tough questions to teams, challenging them to explain the rationale of their choices, to which all teams responded in a knowledgeable and appropriate way. Certainly teams put a lot of effort in their projects, including the preparation of the finals, with the considerable and enthusiastic help of their mentors. It is indeed encouraging to see the level of engagement and maturity of software engineering students around the globe.

Although the off-site selection phases were not as deep and well-documented as the final evaluations, the quality of non-finalist projects was also more than satisfactory with very few exceptions. In fact, it appears that teams and their teachers have been quite conservative in submitting their material, and probably applied some careful self-selection, as evidenced by the fact that, of 50 teams that registered to participate in the contest, only 26 actually submitted summary reports by January 2009.

The PC was also pleasantly surprised with the overall correspondence between the quality of the first summary reports and the final complete delivery. No team that passed the first selection failed to submit the full material, and in most cases they responded well to reviewers' requests despite the short interval (roughly four weeks) between notification of the first selection and deadline for the submission of the complete deliverables.

Enthusiasm of all participants was another source of great satisfaction for the organizers. This was evident during the teams working on the projects and their final participation in the conference. Teams in general paid special attention to the processes they followed, they interacted regularly with stakeholders, worked incrementally and used modern communication technologies for keeping in touch within the team and with the stakeholders.

The competition was also successful in that there was considerable interest and participation by many conference attendees, including those not directly involved (as organizers or as teachers of competing teams) in the contest. All of them reported unanimous appreciation and encouragement to further pursue the endeavor.

While numbers are not large enough to allow for a statistically sound analysis, and more experience is needed, it does appear that interest has been fairly well distributed among the various aspects of software development. In particular, all finalists devoted considerable attention to user needs and requirements elicitation (to some extent also as a result of pressure from the stakeholders).

An unexpected success story in the first edition of SCORE was the significant participation and overall excellent results of geographically distributed teams (two of the six finalists, including the overall winner, were in fact distributed teams), despite the objective difficulties that hamper even professional distributed development. Certainly the supporting universities helped much in this regard, but there have been also cases of failures due to this difficulty.

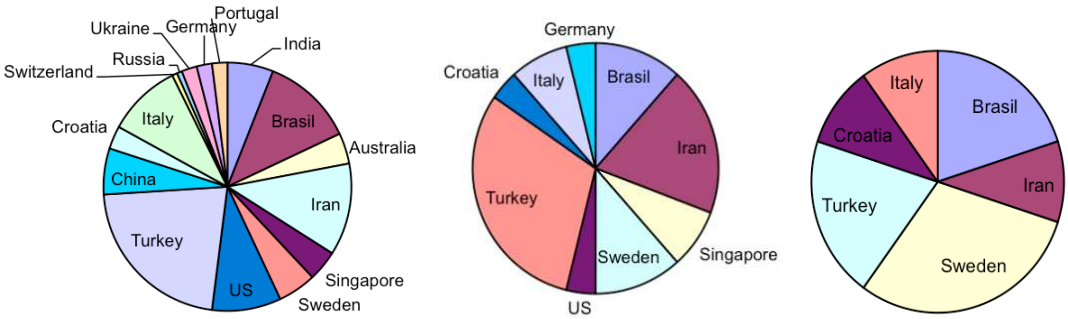

Figure 2. Countries of origin of registered teams (left), of submitted projects (center), and of projects selected for the second round of evaluation (right).

We were pleased to notice significant involvement of countries that traditionally do not have a large presence at SE events, and ICSE in particular, both in terms of attendance and in terms of

papers submitted and accepted (see Figure 2). This was actually one of the major goals of the whole endeavor. However, participation was just “good enough” and remains fairly uneven from a geographical point of view, as Figure 2 demonstrates.

We learned after the conclusion of the competition that several schools used SCORE projects in their courses, but without competing, just to “see what happens”. We also talked to professors who said that they used the SCORE projects from the website, or slight variations of them, to assign to their students, but in the end they decided not to submit the projects because of the perceived overhead of participation. They saw the benefit of the projects in their careful definition by an international panel. This was among the indirect impacts of the contest.

Difficulties

At the inception of the contest, we assumed that it would be useful to involve industry both in proposing projects and in helping during the supervision and evaluation of the projects. We assumed that some students would be attracted to projects proposed by large companies. We underestimated the challenge of cooperation with industry in devising and guiding projects. The extended period of the contest and the unpredictability of time demands can make it hard to maintain support for participating teams as priorities and interests shift in a large organization. Three of the nine published projects were proposed by large companies, but these were not among the most successful in terms of attracting participants nor in terms of providing continuity in support of student teams. On the other hand, a project devised by a PC member from a smaller company proved so popular that we recruited additional stakeholders to interact with student teams.

The limited selection of projects did not suit everyone. Some schools were reluctant to participate given that they could not choose their own projects, and this included schools with a strong tradition of participation in software engineering research. We have sympathy for this point given the effort involved in defining good problems for a university SE program. Schools may wish to recoup this effort in a larger contest, and not start over with pre-selected projects. Nevertheless, we continue to believe the pre-selection of a set of projects for the reasons we have given in prior discussion.

One of the major problems we faced rested in the fact that, even in this Internet era, advertising the contest to potential participants (and especially to teachers of potential teams), and explaining the novelty of the endeavor was most effective only when accompanied by face to face discussions with colleagues, less so when supported only by electronic means.

Finally, the experiment with special tracks was not particularly successful, as we were only able to organize one, on formal methods.

The reception of project topics by teams

The attractiveness of the projects on offer is a considerable factor in the success of a contest like SCORE, so it is useful to consider which of the SCORE 2009 projects were taken up by significant numbers of student teams. By looking at the number of registrations and submissions listed in Table 2, we can group proposed projects in roughly three categories:

- *Group W* ("well attended"): three projects (Distributed Decision, BTW, and GPXCleaner) that each were the target of between 20% and 30% of the registrations and also of the submissions.
- *Group M* ("moderately attended"): two projects (PACS and Pacemaker) that were the target of 10% of the registrations and between 5% and 15% of the submissions.
- *Group S* ("sparsely attended"): four projects that were the target of less than 4% of the registrations, and had no submissions.

Many heterogeneous factors likely contributed to create the picture reported in Table 2, most probable the preferences of students and of their teachers. Let us put forth some conjectures on possible causes.

For various reasons three of the projects in *group S* mentioned above could not be supported throughout the competition by their proposers, and at some point had to be withdrawn from the list of available projects (at the time of the withdrawals, none of the teams that had registered for these projects were actively developing them). This of course had an impact on the number of registrations and submissions received for these projects. However, as Table 2 shows, at the time of their withdrawal, which occurred after July 2008, the projects had been targeted by few teams.

In general, the projects of *group S* correspond to applications that seem focused on fairly specific user bases (software developers in distributed teams, testers of embedded systems), and moreover those user audiences were primarily within the domain of software development. We think this factor could have weakened their appeal. Projects of *group S* addressed real industrial and commercial concerns of companies; however, it is possible that undergraduate and master's students who have not yet had extensive experience in software might find it hard to relate to the aforementioned concerns. Projects with a more direct and easily grasped application outside the world of software development fared better.

Another factor that may have hindered some projects is their reliance on particular hardware. Some of the projects of *groups M* and *S*, to be implemented, needed either a particular hardware platform (e.g., a digital I/O board), or an emulation thereof. This might have created an "entry barrier" for teams, especially if the hardware needed to be bought from a third party.

Exploiting the legacy of SCORE

Based on the analysis of the outcome of the first edition of SCORE, let us elucidate the main lessons learned in the form of suggestions for organizers of future similar contests and teachers of project-related courses. The success of the first edition of SCORE gives us hope that such contests can realistically expect to build on an existing base of interest.

How contest organizers can benefit

Set the scope of the contest

One of the first decisions to be taken when designing and launching a student design contest such as SCORE is its scope. Here the term "scope" has several dimensions: geographical, application field, methodological. Such dimensions are typical (although not exclusive) of the SE world and should be clearly selected from the very beginning.

SCORE selected the scope as widely as possible: every country, school, SE sector, and methodology was equally welcome. On the other hand, it is conceivable to organize smaller contests by narrowing the scope along any of its dimensions. For instance, local contests would restrict the geographical area of interest; this would correspondingly narrow down the focus even along other related dimensions (such as interaction with the industrial environment and duration of the process).

Our experiment with having a special track for formal methods was positive overall. We do not know how far this idea can pushed. Certainly the tradeoff between a general contest and fragmentation among different SE communities must be considered carefully. Within a broad scope contest, however, even the choice of special tracks should have some variety over the years, to avoid giving a biased, humdrum picture of what is "hot" and "interesting" within SE.

Involve industry partners

Because software engineering is not a purely academic discipline and is heavily impacted by technology and industrial practice, software engineering conferences such as ICSE always attempt to involve industry. We started with the assumption that an active and visible role of industry partners in a design contest can motivate the students by offering a wider and more complete picture of SE.

In the previous section, we briefly illustrated the difficulties we had in fully achieving the potential benefits of industrial involvement. While we still believe that industry participation in future contests should be tried, realistic guarantees of support will need to be worked out prior to teams taking on industry-sponsored projects. In particular, choosing industry people with prior experience of cooperation with schools and scientific communities is likely to help. We also suggest picking people from both small and large companies, as this is likely to ameliorate the heterogeneity of both the technical features of the projects they can propose, and the

development process they require. Finally, be careful and detailed when defining and assigning roles: project proposer, stakeholder, reviewer, etc.

We believe that the professional societies might wish to join this discussion, given that it touches on building relationships between academic and industrial software engineering, something paid lip service to by most of us, but perhaps much harder to obtain than we would like to think.

Select the project offering

Selecting, designing, and publicizing the projects is another key and controversial issue. For a wide-scope contest such as SCORE, we are fairly happy with the main choice we took: cover different application fields, leave much freedom in the choice of the methodological approach, avoid self-proposed projects, set a conflict for students in the same school as the project proponent (a departure from normal practice with obvious and serious drawbacks in terms of potential participation).

We also maintain that, in a competition such as SCORE, proposing some projects addressed to a narrow user base or targeting specific hardware is surely not a problem; in fact, these features might actually be desirable in some cases, to achieve some level of diversity in the set of the proposed projects. The possible drawbacks of these features should, however, be kept in mind to achieve a good balance in the project offering.

The positive outcome of our choices does not imply that some variations may not be also successful or even more so. For example, the second edition of SCORE allows greater flexibility in the negotiation of the requirements between the contestants and the stakeholders. This way, contestants can play a more active role even in the definition of the project content itself.

A more ambitious choice could even be to launch a “call for project proposals” open to anyone beyond the PC board. This would help general diffusion of knowledge and may widen the scope of the competition by attracting schools not only in the development of third-party projects but also in the publicity of their own favorite projects. It would also obviously make the whole process longer and more challenging to manage. In summary, this ambitious approach would likely require an already well established and widely recognized event to be adequately supported.

Organize the timeline

A distinguishing feature of contests such as SCORE is that participants must work off line for many weeks, usually encompassing attendance at a SE university course. Thus, to be sure to give a chance to all interested students and schools worldwide to participate, we offered a year-long submission window. This combined with the management of the other components of the whole process – setting up the PC, designing and publicizing the projects, evaluating the deliverables – almost automatically lead to a global duration of two years.

Within contests with narrower scope than SCORE, however, the timeline can be shortened considerably. As worldwide Olympic games and the ACM Programming Contest are paralleled by national and regional competitions, local SE competition could be organized within much more restricted time spans and employing more agile processes.

Advertise the contest

We hope that the difficulties encountered in properly publicizing SCORE will not be an issue in future editions, as the contest develops its own history. However, the problem remains of how to select appropriate means to effectively advertise new endeavors. The proliferation of student contests has caused confusion among potential participants and there is an increased need for each contest to distinguish its goals and intended audience clearly. The primary challenge is to develop a community for the contest. In the case of SCORE, the community leaders to be targeted are the professors of software engineering.

Publicize the evaluation criteria

The evaluation of full SE projects within an international contest lies at the intersection of the practices of peer-reviewed scientific conferences and university courses. As we discussed previously, the first edition of SCORE was quite liberal in letting the evaluators grade and rank the submissions, with the only constraint of a three-phase evaluation process; the choice turned out to be effective.

This should not discourage trying stricter guidelines within different scenarios. We would probably recommend against a strictly numerical evaluation based on assigning a score to several pre-defined categories (e.g., precision and adaptability of the requirements, quality of the test suites, look and feel of the GUI, etc.). In our opinion, this may distract the focus of the participants away from the goals (i.e., quality) to the means (i.e., the numerical scores).

What really matters is that evaluation criteria are made publically known well in advance, preferably in the official Call for Participation of the contest, so that potential participants can better take their decisions and plan their work ahead.

Maintain a project repository

Most scientific events publish their proceedings. Well-established conferences often organize their proceedings in a series which contributes to strengthen the tradition and knowledge of the event. We are quite convinced that contests such as SCORE – not necessarily restricted to SE – deserve this type of documentation in the same line as conference proceedings and, even more, other competitions such as the ACM programming contest or the Math Olympics.

In response to an extensive demand, we have arranged an archive for SCORE [10] which will evolve from the present web sites of SCORE 2009 and SCORE 2011. In the long term, it will contain all major information about the event, and especially data about the projects: their description, who signed up for them, the evaluation of the submitted projects, etc. Among other

things, this will allow us to grant the winning projects the status of an official publication, a well-deserved reward for both the students and their schools and teachers.

More generally, the archive may have a significant impact, not only in increasing interest in the contest, but also as an independent didactic tool, similar to other on-line applications, that teachers can exploit to increase the quality and effectiveness of their courses.

How SE teachers can benefit

While we targeted our suggestions mostly at contest organizers, we also have a few tips for teachers and schools that would like to participate in future contests.

First, you can reap benefits even without competing in the contest by re-using contest projects as your course projects. If you select projects from the currently running edition of the contest, you can additionally decide, after the course is over, if it is worth supporting some of the students to actually enter the contest with their work.

Second, while you may already have strong and interesting projects for your students, and you cannot currently use them to participate in SCORE, contest projects are still worth considering in your class. You can use them to complement your offering to the students and make it even more varied and appealing. You may find that many students find competition among a global student base motivating.

How SE teachers can participate

Try out contest projects in your courses, perhaps as extensions to the projects you already use. Provide feedback to contest organizers on your experience.

Suggest projects that you use as possible contest projects to the organizers. Under SCORE rules, this would prohibit your students from selecting these projects. But you will be contributing to the larger goal of generating a set of interesting projects for the contest and the SE community.

Consider organizing your own regional (e.g., state-wide) contest. Given a more local scope, it may be possible to rework (and shorten) timelines to closely fit the schedule of universities in your region.

Conclusions

SCORE was conceived as a contest to communicate the excitement and creative challenge of software engineering while emphasizing its nature as an engineering discipline that encompasses much more than programming alone. Distinctive features of SCORE included its scope and also its integration with computer science curricula. These distinctive features entail significant

challenges, including devising a set of projects and contest procedures that are suitable for use in software engineering project courses in many different university systems with diverse curricula and calendars. Overall, we have been delighted with the enthusiasm of student contestants and their instructors from many parts of the world, and by the very high quality of contest entries, though there remains significant room for improvement as noted above. SCORE will be repeated in the context of ICSE 2011, and we hope that our experience with SCORE will also serve as a useful model to others who may wish to develop their own SE contests, whether more closely focused on region or special SE topic.

Acknowledgements.

We are grateful to all teams that enthusiastically participated to SCORE and to their teachers. In particular we wish to mention the exceptional contribution by Mälardalen University in terms of number of teams and quality of their work.

References

[1] http://www.usfirst.org/

[2] http://cm.baylor.edu

[3] http://www.icfpcontest.org/

[4] http://ictf.cs.ucsb.edu/

[5] http://imaginecup.com/About/WhatIs.aspx

[6] http://code.google.com/android/adc/

[7] http://score.elet.polimi.it/Projects.html

[8] http://score.elet.polimi.it/media/report_guidelines.pdf

[9] Mehdi Jazayeri and Dino Mandrioli. SCORE: The first student Contest in Software Engineering. *ICSE Companion 2009*, pp. 487-488.

[10] http://score-contest.org