# SPHERES Zero Robotics Software Development: Lessons on Crowdsourcing and Collaborative Competition

**Sreeja Nag**
**Massachusetts Institute of Technology,**
**Cambridge, MA 02139**
**Email: sreeja_n@mit.edu**

**Ira Heffan**
**TopCoder Inc.,**
**Glastonbury, CT 06033**
**Email:**
**IHeffan@topcoder.com**

**Alvar Saenz-Otero**
**Massachusetts Institute of Technology,**
**Cambridge, MA 02139**
**Email: alvarso@mit.edu**

**Mike Lydon**
**TopCoder Inc.,**
**Glastonbury, CT 06033**
**Email:**
**MLydon@topcoder.com**

## Abstract

Crowdsourcing is the art of constructively organizing crowds of people to work toward a common objective. Collaborative competition is a specific kind of crowdsourcing that can be used for problems that require a collaborative or cooperative effort to be successful, but also use competition as a motivator for participation or performance. The DARPA InSPIRE program is using crowdsourcing to develop spaceflight software for small satellites under a sub-program called SPHERES Zero Robotics - a space robotics programming competition. The robots are miniature satellites, called SPHERES, that operate inside the International Space Station (ISS). The idea is to allow thousands of amateur participants to program using the SPHERES simulator and eventually test their algorithms in microgravity. The entire software framework for the program, to provide the ability for thousands to collaboratively use the SPHERES simulator and create algorithms, is also built by crowdsourcing. This paper describes the process of building the software framework for crowdsourcing SPHERES development in collaboration with a commercial crowdsourcing company called TopCoder. It discusses the applicability of crowdsourcing and collaborative competition in the design of the Zero Robotics software infrastructure, metrics of success and achievement of objectives.

## TABLE OF CONTENTS



## 1. INTRODUCTION

The term 'crowdsourcing' was introduced by Jeff Howe in 2006 in a Wired magazine article. He later went on to define the term as: 'Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call[1]. Most generally, a person or organization with a problem invites a crowd to come up with solutions and offers incentives for contribution. Crowdsourcing has been classified into various typologies based on the aims of practice [2]: Problem solving (crowd wisdom), creative input (crowd creation), opinion polling (crowdvoting), outsourcing tasks (crowd production) and raising money (crowdfunding). The effort described here focused on creative input, problem solving and outsourcing for spaceflight software development in the context of the SPHERES Zero Robotics Program – a joint effort between MIT, TopCoder, and Aurora Flight Sciences, supported by NASA and DARPA.

SPHERES Zero Robotics is a DARPA-initiated endeavor under the umbrella program called InSPIRE to develop spaceflight software by crowdsourcing. It is a robotics programming competition where students learn to write programs that control a satellite in space using a web browser. The robots are miniature satellites called SPHERES (Synchronized Position Hold Engage Reorient Experimental Satellites) – an experimental testbed developed by the MIT Space Systems Laboratory (SSL) operating on the International Space Station (ISS) to test control and navigation algorithms in microgravity. The participants compete to win a technically challenging game by programming their strategies into the SPHERES

978-1-4577-0557-1/12/$26.00 ©2012 IEEE

satellites. The game includes command and control problems of interest to MIT, DARPA and NASA. Students use either a graphical editor or a C editor to write code, and then simulate their program and see the results in a flash animation. The simulation uses a high-fidelity 3D model of the SPHERES satellites. The astronauts run the final robotics competition on the ISS and interact with participating students via a live video broadcast in a large event at MIT, webcast live to all participants so that remote viewing is possible. The structure of the tournament includes both competition and collaboration in order to meet the educational goals of the program. It is this mix that enables participants -- amateur developers assisted by mentors -- to create competitive solutions to the specific tasks outlined in the game. The software framework to enable the above process, i.e. allow crowds of students to use the SPHERES simulator, write spaceflight-capable programs and interact/collaborate with each other is built using TopCoder crowdsourcing contests that also use a mix of competition and collaboration. TopCoder is a commercial company that uses a mix of competition and collaboration within their online community of, over 300,000 developers, who voluntarily register on their website, to make scalable, cloud-based software systems (described in detail in Sections 7 and 8).

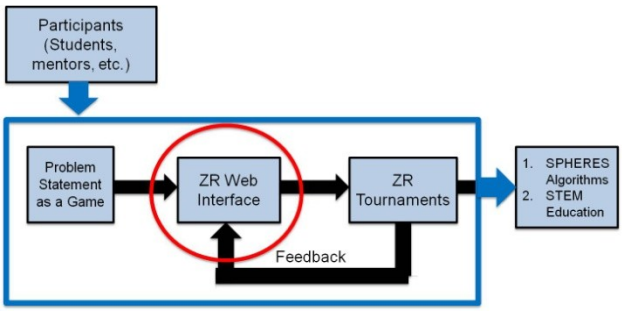



**Figure 1: Zero Robotics Architecture.**

Spaceflight software development via Zero Robotics, therefore, occurs for existing spaceflight hardware and in two stages, as shown in Figure 1: (1) Building the web-based development environment for the programming competitions – circled in red - (by leveraging a crowd of thousands of software developers) and (2) the programming competitions themselves – within the blue box - (when thousands of amateur participants contribute to writing SPHERES software). Both stages are demonstrations of crowdsourcing using different classes of participants and with different objectives. This paper describes stage 1 of the process.

The primary goal of the Zero Robotics tournaments is Science, Technology, Engineering, and Math (STEM) education and the secondary goal is to develop spaceflight algorithms (specifically for SPHERES) at the same time. In Figure 1, the students who participate in the tournaments are the input into the Zero Robotics 'system' and the output are the mentioned objectives, STEM education and satellite software. The 'system' includes a game which is available through the ZR Web Interface, which in turn serves as tools for students to achieve the Zero Robotics objectives. Thousands of developers competed in TopCoder contests to creatively design a web interface for these students (crowd creation) and assemble the software components to build a robust framework to allow satellite control (crowd production). This paper discusses the *process of developing the platform* to organize the tournaments and release the games, referred to as the framework development effort, with the intent of making MIT's SPHERES simulator available and accessible and providing a community platform for crowds to interact and write spaceflight-capable software for SPHERES. It uses data from the development effort to discuss the role of competitions in enabling space research amateurs, from non-technical personnel to software developers, to create space mission software, and the subsequent value gained by the space community.

## 2. SPHERES

The SPHERES program began in 1999 as part of an MIT Aero/Astro undergraduate class. Prototypes were built by the student class in 2000, flight satellites were delivered in 2003, and launched to the ISS in 2006 [3]. SPHERES became one of the first educational programs that launched student-designed hardware to the ISS. SPHERES consists of a set of tools and hardware developed for use aboard the ISS and in ground-based tests: three nano-satellites, a custom metrology system (based on ultrasound time-of-flight measurements), communications hardware, consumables (tanks and batteries), and an astronaut interface. They operate aboard the ISS under the supervision of a crew member (Figure 2).


**Figure 2: SPHERES operates 3 satellites aboard the ISS (astronaut and MIT alum Gregory Chamitoff)**

The ground-based setup consists of a set of hardware analogous to what is in the Station: three nano-satellites, a metrology system with the same geometry as that on the ISS, a research oriented GUI, and replenishable



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

consumables. The SPHERES satellites implement all the features of a standard thruster-based satellite bus. The satellites have fully functional propulsion, guidance, communications, and power sub-systems. These enable the satellites to maneuver in six degrees of freedom (6-DOF), communicate with each other and with the laptop control station, and identify their position with respect to each other and to the reference frame. The laptop control station (an ISS supplied standard laptop) is used to collect and store data and to upload new algorithms. SPHERES uploads new algorithms (ahead of time) and downloads data (after the session) using the ISS communications system.

Figure 2 shows a picture of a SPHERES satellite and identifies its main components. Physical properties of the satellites are listed in Table 1.

| | |
|---|---|
| Diameter | 0.22 m |
| Mass (w/tank & batteries) | 4.3 kg |
| Max linear acceleration | 0.17 m/s$^2$ |
| Max angular acceleration | 3.5 rad/s$^2$ |
| Power consumption | 13 W |
| Battery lifetime (replaceable) | 2 hours |

**Table 1: SPHERES Physical Properties**

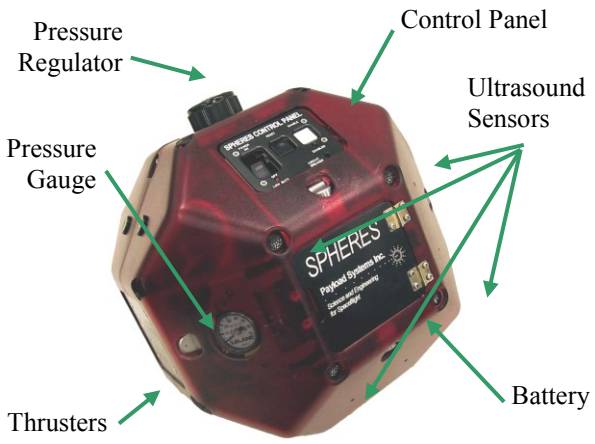



**Figure 3: A SPHERES Satellite**

SPHERES was designed to be a *facility* aboard the ISS, not just a single experiment, by following a set of design principles learned from previous MIT SSL experience [3]. To provide the ability to involve multiple scientists in a simple manner, a SPHERES Guest Scientist Program was created [4]. This program consists of a test development framework, a robust and flexible interface to the SPHERES flight software, a portable high-fidelity simulation, two laboratory test beds and data analysis utilities, and supports the efforts of geographically distributed researchers in the development of algorithms. The Zero-Robotics program expands the Guest Scientist Program with a simplified interface so that students at many different grade and skill levels can program the satellites.

## 3. ZERO ROBOTICS TOURNAMENTS

The Zero Robotics (ZR) competitions draw significant inspiration from FIRST Robotics [5] and shares common goals including building lifelong skills and interest in science, technology, engineering, and math through project-based learning. FIRST Robotics concentrates heavily on the development of hardware, has a registration fee and does not have any space-related components. Since SPHERES concentrates on the development of software, Zero-Robotics complements FIRST Robotics by providing students an avenue to further develop their software skills, with the incentive that the software they develop will be tested by robots and astronauts in space at no cost to participants.

In fall 2009, the SSL conducted a pilot program of the Zero Robotics competition with two schools/10 students from northern Idaho [6]. In 2010, Zero Robotics was a component of NASA's Summer of Innovation, a nationwide program targeted at encouraging STEM education for middle school students. During this competition, 10 teams and over 150 students from schools in the Boston area worked for five weeks to program the SPHERES to compete in an obstacle course race. In the fall of 2010, Zero Robotics conducted a nationwide pilot tournament for high school students named the Zero Robotics SPHERES Challenge 2010. Over 200 students from 19 US states participated as part of 24 teams. The objective of the game was to complete the assembly of a solar power station by maneuvering a satellite to dock with a floating solar panel and then bring it back to the station to finish the mission before the opponent does.

In the fall of 2011, the ZR tournament grew again and had 145 teams participating from all over the USA and select countries in Europe. The objective of the 2011 game was to navigate the satellite to collect a variety of tools, mine asteroids by spinning on it or revolving around it and depositing the collected ore in mining stations. Twenty-seven of the teams that participated will be able to see their code run on the ISS. While previous competitions used a prototype web interface, 2011 used a web interface and the integrated development environment to support this growth in participation, and for the first time the infrastructure for the competitions was itself developed using crowdsourcing. We expect that this infrastructure will enable to the program to scale to many more teams in the future. While this paper deals with the development of the infrastructure only, there is separate literature available that reviews the Zero Robotics tournaments and their impact on crowdsourcing and STEM education [7].



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

## 4. ZERO ROBOTICS WEB ENVIRONMENT

Each Zero Robotics tournament has included several competition rounds in which students play the same or different games against each other. Student teams submit an application on http://zerorobotics.mit.edu/. Upon acceptance, they can create, edit, share, save, simulate and submit code using the ZR website. The components of the web environment available to the students will be explained briefly in this section; more detail can be found in previously published Zero Robotics literature [7]. Apart from the game, animation and the graphical editor, all the components described below were built using TopCoder crowdsourcing contests.

### *4.1. The Zero Robotics Game*

For each tournament, the Zero Robotics development team designs a different game. The game has the following goals, developed from the lessons learned during previous instantiations of Zero Robotics tournaments and constraints of the SPHERES hardware and software.

- A game with relevance to state-of-the-art research with SPHERES, so that the work of students can contribute to future research at MIT, NASA, DARPA, and other research centers.
- Each team controls one SPHERES satellite during the game which involves two teams.

Each Zero Robotics game is designed, balanced, tested, programmed into the SPHERES Zero Robotics API by MIT and made available on the Integrated Development Environment on the ZR website. The game design and testing was not developed through crowdsourcing.

### *4.2. Software Architecture*

In the past, programming the SPHERES satellites required users to have access to the compilers for the SPHERES processor and familiarity with the Guest Scientist Program. This was not practical to engage large numbers of students of high school age and below. Instead, MIT and TopCoder have developed a web-based interface to program the satellites which makes use of the same SPHERES high-fidelity simulation that is used to develop flight software.

The programming takes place via a web-based GUI, which provides a simplified interface to the Guest Scientist API functions and enforces constraints that guarantee compatibility with the SPHERES compilers. Students have access to a text based editor as well as a graphical editor, for those with little or no prior programming experience. A distributed computation engine, hosted on Amazon EC2 virtual machines compiles the user code, links it with the core SPHERES software, and performs a full simulation of the program.. An Adobe Flash-based visualization creates an animated representation of the results. The code programmed by the students via the web interface can be executed in the SPHERES hardware. The flow of information in the ZR software infrastructure is shown in Figure 4. The user code is transmitted to the web app which launches a simulation instance on the 'Farm' which on completion returns the results to the web app and finally the browser, then rendered in the form of an animation.

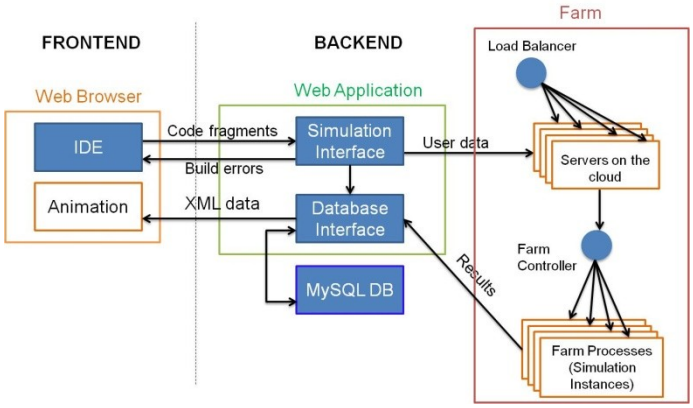



**Figure 4: ZR Software Architecture**

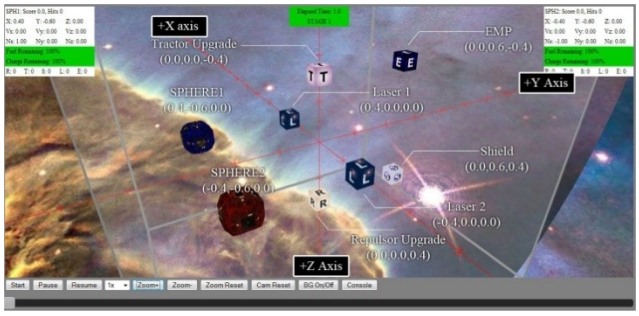



**Figure 5: Example of a ZR Animation**

Users write code inside a main function called *'ZRUser()'* available in each project. *ZRUser()* is called at every iteration of the satellite control cycle (approximately once per second). User defined procedures are all called inside this main which has as its inputs, the position, velocity, attitude and attitude rates of each of the satellites and the time since the game begun. The code within and called by *ZRUser()* is inserted into a pre-defined template and called by the ZR simulation engine to model control of the SPHERES satellites.

### *4.3. Graphical Editor*

The ZR graphical editor allows users with little or no C experience to write code using drag-and-drop programming. It is currently possible to see and generate C-code from the diagram view so that users can initiate their code with diagrams but can move on to more complicated code using the C editor. The graphical editor



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

was built by Aurora Flight Sciences and integrated into the overall software framework of Zero Robotics using TopCoder crowdsourcing contests.

### 4.4. Team and Project Management Tools

In the ZR tournament, teams are organized into two types of members: team leads and team members, with different permissions for each role. The ZR website provides users with the tools that they need to create, edit, share with others, compile, simulate and save all their projects and results. The ZR simulation allows users to tweak different game parameters and choose simulation settings so that they can test different parts of their code independently. They can simulate an individual project, race against another member of their team or race against standard players provided by MIT. The simulation also allows students to control the speed of the game to show the motion in real time or up to 10 times faster. In a formal competition, these settings are fixed by MIT and the purpose of the simulation is to provide ample opportunities to test different strategies and finalize a robust submission.

During the tournaments, teams are given the opportunity to challenge other teams for informal scrimmages. The website provides the ability to select a user project and invite other teams to race their projects against the selected one – called a 'challenge'. Teams can accept or reject challenges using the provided UI and view the results, animations and leader boards for each challenge that they participated in. A simple interface is available to teams for submitting a project as an entry into a formal competition. MIT runs automated simulated competitions using these submitted projects as elimination rounds. Teams that reach the final round have their programs run on the SPHERES satellites aboard the ISS with the help of astronauts.

The Zero Robotics website, the IDE and the all the management tools were and are being developed using crowdsourcing contests supervised by MIT and TopCoder. More detail about the contests and quantitative results is provided later in the paper.

## 5. INNOVATION USING COMPETITIONS

### 5.1. Historical Usage

Challenging crowds to compete to achieve a difficult goal by providing the incentives of prizes has a long history and has led to many successful competition solutions (hence, the terms 'challenges' and 'competitions' will often be used interchangeably). In 1714, the English parliament, seeking to solve the difficult problem of accurately determining ships' longitude at sea, created a Board of Longitude to oversee the offer of a prize of 20,000 pounds to anyone who could solve the problem. Parliament could have directly funded astronomical research efforts, however, instead they chose to offer a prize to anyone who could solve the problem. John Harrison, a self-taught clock maker developed an improved clock design that would be accurate at sea. [8]

In 1775, a prize of 100,000 francs was offered by the French Academy of Sciences for the production of alkali soda ash (sodium carbonate) from salt (sodium chloride) [9]. A surgeon, Nicholas Leblanc, developed a process that some have since characterized as the beginnings of the modern chemical industry[1]. In 1919, a $25,000 prize was offered by hotel magnate Raymond Orteig to the first person to fly non-stop between New York and Paris. In 1927, Charles Lindbergh won that prize, landing 2½ hours ahead of schedule [10].

### 5.2. Recent Usage

A more recent example of the use of large-scale innovation tournaments in aerospace include the X-Prize competition. On October 4, 2004, the X PRIZE Foundation awarded a $10 million prize to Scaled Composites for their craft SpaceShipOne [11]. Aerospace designer Burt Rutan and financier Paul Allen led the first private team to build and launch a spacecraft capable of carrying three people to 100 kilometers above the earth's surface, twice within two weeks, the first humans to achieve this feat.

U.S. Government agencies can use challenges to reach out to thousands of citizens, which is why the White House has been encouraging agencies to consider the use of challenges as a policy tool. At the outset of his Administration, President Barack Obama signed the Memorandum on Transparency and Open Government, committing the Administration to creating a more transparent, participatory, and collaborative government. In Sept. 2009, the President released his "Strategy for American Innovation" calling for agencies to increase their ability to promote and harness innovation by using policy tools such as prizes and challenges [12]. On Dec. 8, 2009, the Director of the Office of Management and Budget (OMB) issued the Open Government Directive, which required executive departments and agencies to take specific actions to further the principles established by the President's memorandum, including to develop an Open Government Plan that should "include innovative methods, such as prizes and competitions, to obtain ideas from and to increase collaboration with those in the

[1] It is interesting to note, however, that both Harrison and Leblanc had trouble collecting on their prizes, Harrison due to the resistance of the astronomical establishment that was holding out for an astronomical solution and Leblanc due to the French Revolution.



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

private sector, non-profit, and academic communities [13]. In January 2011, the America COMPETES Act [14] was reenacted, which authorized all government agencies to conduct challenges and competitions.

Challenges must be designed to meet their intended goals. There is no single type of challenge that can fulfill all needs. A program that is solely intended to educate the public about a topic will be designed differently than a challenge that is created to obtain an innovative solution. To explore these differences, NASA created the NASA Tournament Lab (NTL) in collaboration with Harvard Business School and TopCoder to use open innovation challenges to solve problems within the NASA scientific and research community, and to reach beyond the walls of the research centers and engage the world to help solve its challenging and complex problems [15]. Some examples of successfully crowdsourced (crowd wisdom) NTL problems are:

- NASA required the development of a robust software algorithm that would efficiently recognize vehicles in aerial images [16]. A set of 1000 images containing vehicles and 3000 images containing only background were provided as test cases. The algorithm submissions were tested against a larger set of data. After the problem had been selected and framed, a three-week competition was held on the TopCoder platform. During the competition, 139 programmers from around the world participated by submitting 549 total submissions. The preliminary data analysis by the NASA team showed that the top five solutions were a significant improvement over their current algorithms, employing "state of the art computer vision methods." NASA is currently working on integrating the winning submissions into their own solution.

- NASA's Space Life Sciences Directorate required the development of a software algorithm that would solve a "backpack problem," of recommending the ideal components of the space medical kit included in each manned space mission [17]. As mass and volume are restricted in space flight, the medical kit has to be designed in a way such that both expected and unexpected medical contingencies can be met through the resources in the kit as well as be attuned to the characteristics of the space flight and crew. The challenge was to develop a software algorithm that, based on mission characteristics, would minimize mass and volume and provide the resources necessary to minimize poor health outcomes or mission abruption. After the problem had been selected and framed, a 10 day competition was held on the TopCoder platform. During those 10 days, 439 programmers from around the world participated by submitting 5994 program submissions. The preliminary data analysis by the NASA team is that the solutions developed by the leading entries far surpass the current state of the art internal to NASA in terms of computation time (30 seconds as compared to 3 hours), diversity of technical approaches and robustness. After the competition was done, NASA researchers reviewed the top 5 highest scoring code submissions by looking at the actual code and documentation and said that "The amount of useful code developed in such a short amount of time really made us reconsider some of the ways that we write software" [18]. The NASA team was not able to directly import the code into their software because their model was created with the SAS software analytics package, but they converted elements from the winning submissions to develop a new algorithm to design the medical kids used in space missions.

- NASA wanted to generate ideas for new applications to allow exploration and analysis of the NASA Planetary Data System (PDS) databases - http://pds.nasa.gov/. While rich in depth and breadth of data, the PDS databases have developed in a disparate fashion over the years with different architectures and formats; thereby making the integrated use of the data sets difficult. Consequently, a challenge faced by NASA and the research community is to maximize the usefulness of the enormous amounts of PDS data and identify ways to combine the data that is available to generate interesting applications (e.g., visualizations, analysis tools, educational applications, mash-ups). The goal of this challenge was to generate ideas for these applications. Submissions included a description of the overall idea, a description of the target audience, the benefits of the application for the target audience, the nature of the application (how should the application be implemented? Overall, submissions were expected to be around 2-3 pages of text including figures and tables and images. No code or software was necessary. Prizes included a $1000 grand prize and 3 $500 runners-up prizes. A $750 "community choice" selected by the community also was awarded. There were over 40 submissions received, with the winner proposing an application concept that was focused on a PDS documents parser, processor and validation tool that could be used to identify what areas, parameters, and objects of the planetary systems are well researched and what objects are "white spots," meaning that the data is sparse and more research is needed [18]. Future competitions will include implementing the winning idea.

To summarize, competitions have had a long history to spur innovation and solve problems creatively (crowd wisdom) and in large numbers (crowd production). The government and NASA have only recently tapped into the power of challenges to organize their enormous amounts



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

of information available, identify and solve complex problems and to democratize the innovation process.

### 5.3. Collaborative Competition

Competitions can organize individuals to work toward a common objective with the incentive of a monetary or non-monetary reward. Individuals with a diversity of skills can participate in the task, with a self-allocation of individuals to tasks in which they believe that can be successful. Collaboration allows individuals towork together to achieve larger goals. Development through competitions requires a careful balance of competition and collaboration to achieve its goals.

While big competitions 'challenge' the public with a difficult objective, a series of smaller challenges can be used to engage multiple participants if the challenge structure includes collaboration. Collaboration among the participants allows for the accomplishment of larger tasks by multiple people, and for the performance of each participant to be improved by learning from others. There are a number of ways to bring collaboration into a competitive model, but it is important to retain the benefits of competition.

## 6. ZERO ROBOTICS WEB INTERFACE DEVELOPMENT METHODOLOGY

The Zero Robotics software is being developed using TopCoder's methodology of crowdsourcing contests with the intent of improving the accessibility of MIT's SPHERES simulator and providing a community platform for crowds to interact and write spaceflight-capable software for SPHERES. TopCoder conducts competitions among members of its world-wide technologist community to create software and technology solutions. Problems are posed in an "open call" for solution submissions of a specific type, size, and approximate complexity, and submissions are judged to determine the winner, typically with monetary prizes awarded to the best solutions. For each of these contests, a specification for the desired deliverables is published along with the price to be paid for the "best" solution that meets minimum criteria, and in response developers submit the actual deliverables. Contestants can compete to develop the best algorithm to solve a particular problem, to develop a user interface design, the code for a software component, or to conceive of the best approach to a business or operational problem or opportunity using technology. Solution submissions can range from documents containing ideas, workflow, schematics to graphic design assets such as user interface designs, wireframes and story boards to files containing software code, test data and technical documentation. For many solutions, standard competition types and deliverables formats reduce the learning curve for participants.

### 6.1. Evaluation Criteria

Competition judging methods depend on the type of competition. For most types of deliverables that can be reviewed objectively, submissions are peer-reviewed by historically top-performing reviewers from the community with a rigorous scorecard, and the winner selected based on those scores. However, not all deliverables can be judged objectively. Some other examples are:

- Sponsor of the challenges selects the submission they believe to be most valuable and most closely meets the criteria set forth in the challenge.
- Client and reviewers select the winner based on their preferred submission (subjective); e.g. business requirements contests.
- Automated testing and scoring is used to evaluate; e.g. algorithm development contests can be judged based on the performance and/or accuracy of the algorithm using a specified test data and scoring method focused on the desired results.

In each of these scenarios, the evaluation method needs to be clear and objective, and the results transparent for all participants.

### 6.2. Incentive Structure

The TopCoder web site is designed to identify, promote, and reward the best participants in each category of competition. Cash prizes are awarded to winners and runners-up, and competitor results are posted on the site for public recognition of outstanding performance. A member's username is displayed on the site in a color that reflects their rating, so that their rating becomes a part of their online identity [21]. Detailed, publicly-available statistics above are kept on the web site so that all participants can see how they compare to others such as biography, TC contest statistics, reliability rating, performance and scores from all categories of contests participated in. This allows each member to judge the level of competition in a potential contest and determine the amount of effort he will put in accordingly. For each contest type, there are both short-term prizes and long-term incentives. Competitions typically include prizes for 1st place and at least one runner-up. Some contests also include milestone prizes that are paid based on mid-competition deliverables. In addition, there may be incentives for submission reliability over time and for continued participation, like the "Digital Run" prize pool. These are all in addition to opportunities for additional participation as a reviewer or co-pilot based on historic competition success.

### 6.3. Benefits of Competition in Development

The competition-based development model is successful for a number of reasons. Some of them are that:



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

1. The development conducted through competitions does not depend on the knowledge or availability of any particular individual as a single point of failure.
2. There are innovation benefits that come from reaching out to a global pool of solvers who have a diversity of skills and experience, and bring their creativity to a particular task at hand.
3. The contest judging process inherently includes a detailed review process for assuring the quality of work.
4. Individuals self-select the tasks on which they choose to perform, and for which they are motivated and believe that they have the ability to be successful.
5. Winning submitters are paid a fixed price for the deliverables, and are paid only if their deliverables meet minimum criteria and are delivered by the deadline.

TopCoder's platform has hundreds of new registrants each week and thousands of active participants. The platform is therefore likely to have individuals with the necessary skills and willingness to participate in a given technology-related task. Of course, these significant benefits come with some requirements. Problems must be presented in a format that is suitable for competition. TopCoder has had to develop expertise in developing the formulation of problems and presenting them to a community so that they can be solved in a systematic manner. Also development environments and test data must be provided in a way that is accessible to the community.

### *6.4. Benefits of Collaborative Competition in Development*

The collaboratively competitive development of Zero Robotics' platform, as per the TopCoder methodology, is based on competition, in that there are competitions for each design and development task. These competitions offer both monetary and non-monetary incentives for the participants. Participation in competitions are entirely voluntary and allows the participant complete flexibility and control over their choice of projects. Incentive structures for crowdsourcing challenges in the form of prizes can achieve societal influence in seven different ways [19]:

1. Identifying excellence
2. Influencing public perception
3. Focusing communities on specific problems
4. Mobilizing new talent
5. Strengthening problem-solving communities
6. Educating individuals
7. Mobilizing capital

While each challenge is inherently competitive, the overall effort also includes a significant amount of collaboration, both structured and unstructured.

- Much of the collaboration in TopCoder is structured collaboration, i.e. the TopCoder process dictates how that collaboration takes place. Portions or all of the deliverables created in one competition (e.g., software architecture designs) are used as specifications for another competition. The deliverables are created in a predetermined format to make the communication of information as seamless as possible. In addition, the architects and reviewers in a competition work with the developers during the competition to answer questions and to finalize the deliverables. A "final fix" stage of the competition requires a developer to make changes in response to minor errors or omissions identified by the reviewers. This is similar to code reviews conducted by many development organizations, but takes place at each stage of the software creation lifecycle, not just coding.
- With respect to unstructured collaboration, discussion forums enable participants to ask questions and discuss the requirements with the architects, clients, and each other. This discussion often adds additional detail or relieves ambiguity in the contest specification. It also provides a record of the reasoning for the design and implementation decisions that are discussed. Even while members compete against one another, their interests in algorithms and software brings them to common ground and members are typically willing to help each other as well as teach and advise beginners. The general discussion forums are home to a very active level of interaction about topics of interest to this community.

The structured collaboration in the TopCoder model is important because it enables individuals with different skill sets to address different parts of the problem to be solved and enables distributed development. In other words, it allows a "team" to form in order to solve a complex problem without requiring the team members to establish relationships with each other. It allows team members to pick their contribution based on their interests and skills. Additionally, the structure of the collaboration process makes each team member's contribution and interaction transparent to the other participants. The documentation developed at each stage is critical because the members of the team can keep changing, so the combined knowledge exists not in the experience of the individuals alone but in the documentation and process.

On the other hand, this collaboration structure does add overhead. Since communication is limited to the written documentation and the forums, the interface definitions and documentation are required at every stage. Collaboration with another individual requires at least



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

some written specification of the task, and evaluation of results. At times, particularly when a small, fast change is needed, this overhead seems to take longer than it would if one could just call up the developer on a team and request the change. However, there is not just one developer who can make the change, and so the availability of 'the' developer on the team is not determinative of whether the change can be made.

### 6.5. Development of Complex Systems

Crowdsourcing is not just for a single problem using a single contest to solve it. Large problems can also be broken down into smaller sub-problems that each can be solved by a contest. For example, a computational problem might require an algorithm competition to obtain an algorithm that would solve a problem, and a software component design competition and a software component development competition after that to implement the result of the algorithm competition in a useful framework for use by NASA.

On the TopCoder platform, development projects typically are planned out in "Game Plan" schedules that show the series of competitions scheduled and estimated costs for delivering them. The game plans do not have particular individuals associated with each task, rather the competitors decide whether to participate in the contest for each set of deliverables. Predictions can be made based on past history and the competition parameters (e.g., competition type, pricing, timing) what the likelihood of successful completion will be during the competition lifespan.

For a large, complex project such as the Zero Robotics competition and development environment, we divided the project into several modules and used the traditional Software Development Life Cycle (SDLC) for each module. Each phase of the SDLC loop is a crowdsourcing contest and its outputs are fed into the next phase as input to the next crowdsourcing contest, as shown in Figure 6. Parallel development is therefore possible and interface requirements are very strict (defined by the TC project manager – the only managerially hired position in the entire process) to prevent misfits later.

The top level phases of the lifecycle are:

1. Conceptualization and Specification
2. Architecture
3. Component Production
4. Application Assembly
5. Certification
6. Deployment

A large project is broken into multiple modules that need to be developed; each module is developed through the above phases and each phase has one or more contests.

*Conceptualization* competitions develop Business Requirements documents and High-Level Use cases as solutions. These are then provided as inputs to *Specification* competitions, which develop Application Requirements Documents, Use Cases, Activity Diagrams, and Storyboard and/or Prototypes. These design specification deliverables are then used in *Architecture* competitions to develop Module and System Design Specifications, Sequence Diagrams, Interface Diagrams, and Component Design Specifications. Test cases also may be developed at this time, by conducting testing competitions. The Component Design Specifications are used in competitions to design and develop reusable software components that implement the design. In *Application Assembly,* the components are assembled and the deployment requirements documented. In *Certification*, the assembled software is thoroughly tested through testing competitions and the application is deployed on a staging server for a final integrated set of tests. After the completion of all the phases, the solution is ready for deployment. Figure 6 does not show all of the competitions currently offered by TopCoder. Neither is this the only way that crowdsourcing can be used to develop large, complex systems. Other contests that have not been shown include algorithmic problem-solving, graphic design, user interface design, idea generation, wireframes, prototyping, etc. that might be employed in the development of a technology solution. Zero Robotics development included many such contests.

## 7. CROWDSOURCING CONTEST INFRASTRUCTURE

Conducting a competition is much more involved than simply posting the challenge to a web site. Important elements of the collaborative competitive infrastructure provided by the TopCoder competition platform used to develop the ZR web interface are:

1. A web interface to make competitions structured, organized, compelling and interesting. TopCoder performs these functions using its website: www.topcoder.com
2. A web interface that allows easy problem disambiguation, formulation, communication, validation, recognition and rewards.
3. Behind-the-scenes infrastructure for handling competiton participants' paperwork and inquiries, generating and assuring assent with competition rules, and for legal compliance.
4. Intellectual property rules and documents in place to enable the conduct of competitions to develop assets for enterprise or government clients.
5. Infrastructure to allow customers to create and launch their own contests and follow a workflow to administer the challenge to completion and transfer of assets.



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

6. A centralized web location for participants to obtain problems, submit solutions, judge submissions, view results, scores, statistics, and so on.
7. A central web location for discussion and interaction, providing the community with a "town square" with discussion boards and a wiki to share information.
8. Profiles of and information about the different competitors - all of a member's activities are tracked in real-time and statistics on performance made publicly available.
9. Collaborative software development infrastructure such as source code control, wiki content management, etc.
10. Quick fix mechanisms to make time critical and small corrections to software developed during regular contests. At TopCoder, Short stint challenges called "Bug Hunt" and "Bug Race" competitions are specifically designed to elicit a working solution to a small problem. These challenges are used to update content, to develop quick fixes to technology assets and documentation where the contest ends once a demonstrable solution is submitted, often in a matter of hours.

TopCoder's clients can identify the problem to solve and even contribute to picking and choosing what parts of the process to use. This approach is particularly well-suited for the development of new systems, where the integration points with existing systems are well-defined and can be tested by the community or accurately simulated. Bugs in existing systems can also be fixed using the same types of development environment made available to the community.

Development of upgrades to existing systems where integration points with other systems are not available to the community, and are not easily mocked or simulated, can be more challenging because this typically requires additional client personnel to help identify the pieces that can be developed by the community and to integrate and deploy them into the client environment.

Over the past three years, TopCoder has run over 4500 challenges with 91% completing successfully. Among other factors, TopCoder attributes the high rate of success to the methodology of breaking down a task and honing in the key elements, the large size of the community covering a variety of technology disciplines, and the ability to use of historical data to design and price the challenges in a way that they will be successful. Additionally, TopCoder has over the past ten years developed and refined these contests, attracting hundreds of thousands of technologists and the infrastructure to support them.

With the respect to 9% of challenges that are not successful, TopCoder's view is that a number of factors contribute. Most typically, a competition does not complete successfully because the specification is unclear or is too complicated and is asking for more than is typically requested for that competition type. The main indicator of this is the activity – or lack thereof – in competition registration and in the discussion forums. Sometimes the market is changing, or TopCoder is testing the market, or the prize amounts are set too low to encourage sufficient participation on a particular problem. Usually, in these cases TopCoder can achieve a successful result by dividing the contest specification into multiple parts, and reposting as separate competitions, or by just raising the prizes. Of course, when TopCoder experiments with pricing, changes competition types or deliverables, or adds a new competition type, there is an expectation that some competitions may not complete successfully as the market adjusts to the change.

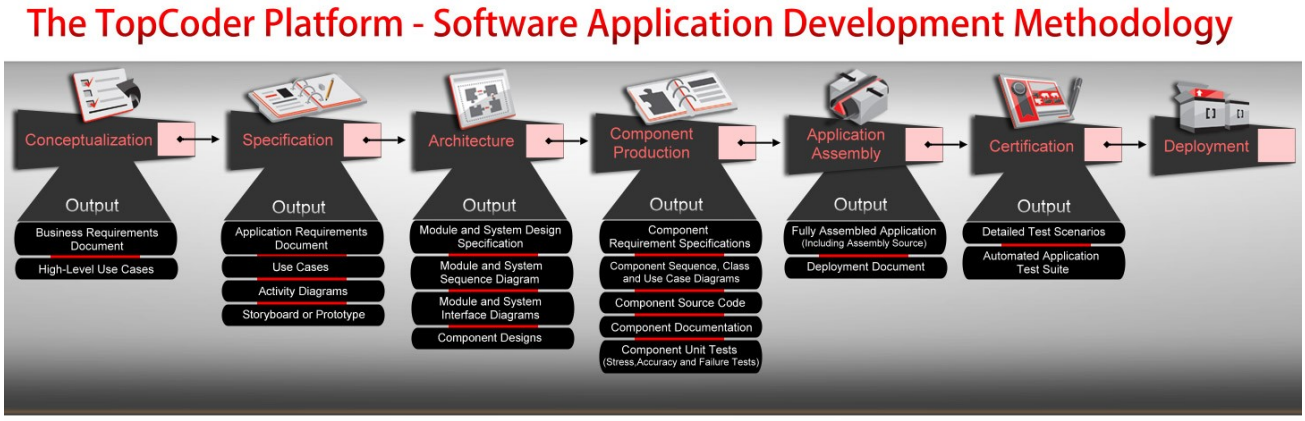



**Figure 6 – Example Software Application Development Methodology**



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

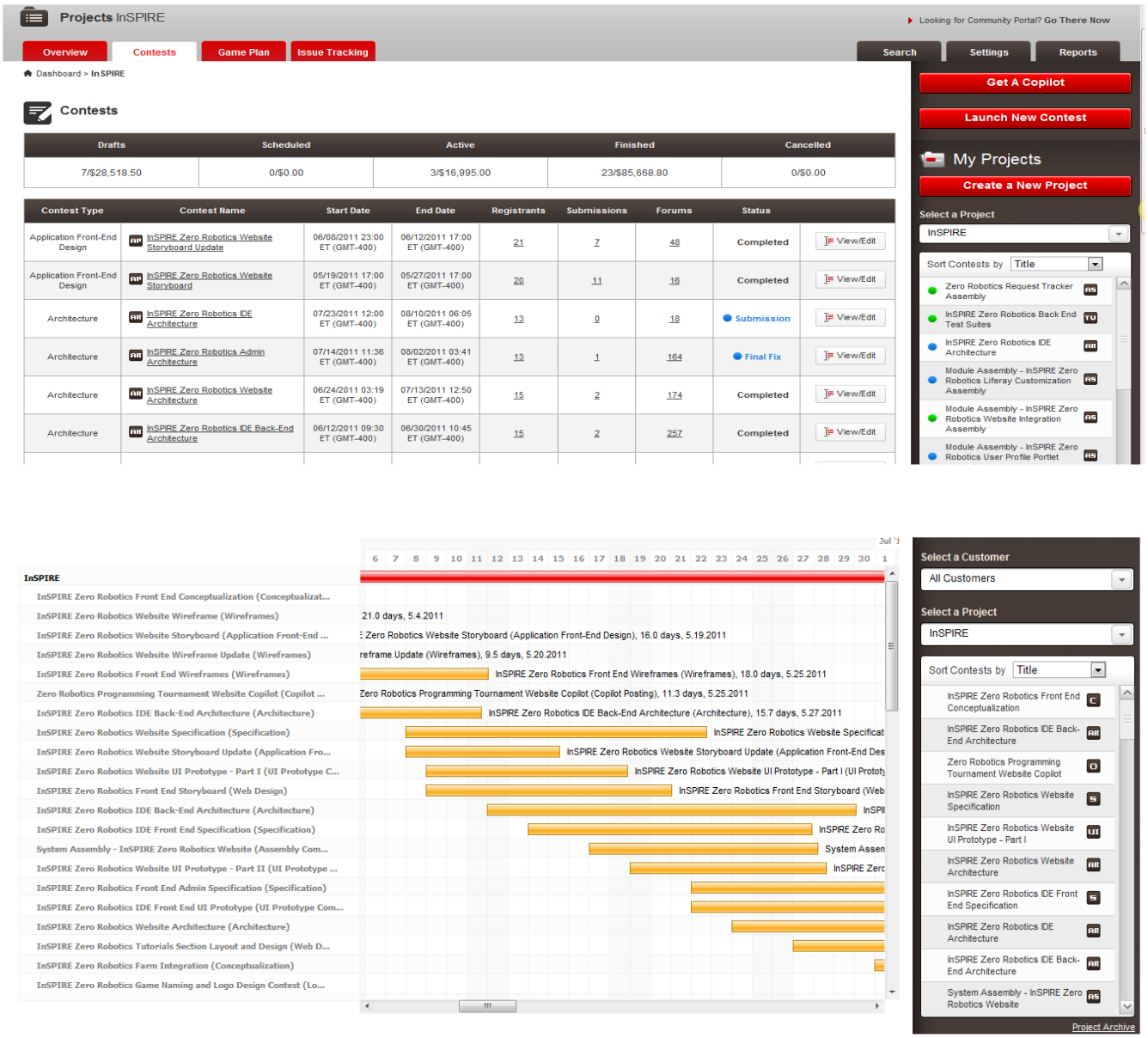



**Figure 7: List of contest details and schedule of the InSPIRE program to develop the Zero Robotics Web Interface**

## 8. CROWDSOURCING CONTEST RESULTS

The Zero Robotics infrastructure was built using the 2010 Zero Robotics web site as a prototype via TopCoder crowdsourcing contests. The program has a TopCoder co-pilot who interacts regularly with TopCoder and MIT and provides technical support to the competition participants. MIT's role was to answer technical questions relating to the requirements in each of the contests and provide detailed feedback to the co-pilot and members. As mentioned in Section 7, there are online tools available to track the ongoing contests. Figure 7 shows a screenshot of the TopCoder Cockpit tool displaying the list of contests, present and past, statistics, and timeline. At a high level, the development tasks undertaken using collaborative competition were:

1. Integration of the Graphical Editor being built separately by Aurora Flight Sciences.
2. Development of the Zero Robotics community website.
3. Development of the SPHERES integrated programming environment using the 2010 version as a prototype
4. Integration of the SPHERES high-fidelity simulation into the TopCoder server compilation and testing 'Farm', which is the robust back-end handling and implementing the ZR simulation requests.



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

A Game Plan schedule was developed for each high-level task, divided into the following phases: Conceptualization, Wireframe (to design the look), Storyboard (to design the feel), Architecture, Assembly, Testing and Deployment. For each task and each phase, a list of required contests were made and recorded within the Game Plan. Part of the Game Plan for the front end task is shown in Figure 8. The horizontal blocks represent each phase and the rows represent an individual contest. The columns are the timeline; the pink regions mark off the period when a specific contest is scheduled to take place.

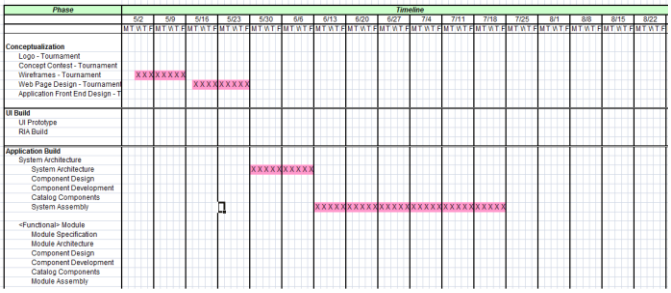



**Figure 8: Front End game plan**

Each individual contest lasted between 5-21days and awarded prizes between $100-$2500 depending on its requirements and scope of the contest, and were defined based on TopCoder's historical experience in the market for each type of deliverable. The crowdsourcing contests included 3 types: graphic design Studio contests (which have been described earlier; evaluated by MIT and TopCoder), software contests (which have the milestone and submission phases but are evaluated by reviewers selected from within the TopCoder community by the program manager) and bug race contests (where the first member of the TopCoder community to submit a solution wins).

Each Studio contest began with the release of a set of requirements and the inputs needed by the participants. Members of the community registered to participate in the contest during the 'Registration phase'. Once the contest launched, participants could review the requirements and work on the problem. For some competitions, such as the conceptualization and wireframe competitions, half-way through the contest participants were required to submit a "milestone" submission. Reviewers and/or the client team reviewed the milestone submissions and provided feedback to participants, awarding small prizes to up to five participants. Participants integrated the milestone feedback into their work, improved upon it and submitted their full solution by the contest deadline. All the entries were then evaluated and first and second place awarded prizes. The winners are responsible for improving their submission according to the reviewer's final comments in the post-contest 'Final Fix' phase.

An example of such a contest is the Front End Storyboard Challenge. The purpose of this challenge was to generate ideas for a look and feel for the web-based integrated development environment to be used by students to program satellites. The prizes for this competition were $1500 for first place and $500 for second place. There were 5 milestone prizes of $75/each. Participants were provided with a description of the solution needed, along with the conceptualization document and wireframes that had been developed in previous competitions. In response, the participants provided a series of graphic images that showed creative examples of how the screens might appear. The competition began June 9, 2011 at 9am Eastern. Milestone submissions were due June 12, 2011 at 9am Eastern, and the final submissions due June 15, 2011. The winners were announced on June 21, 2011. The milestone submissions allowed the solvers to get feedback about their submissions, opening lines of communication. It also helped the competition sponsors determine whether there was sufficient participation in the competition. In this competition, there were 18 registrants, with 10 submissions at the milestone and 4 final submissions. The "best" storyboard as determined by MIT and TopCoder (Figure 9) was selected from these 4 submissions and served as an input into the architecture group of contests for the website.

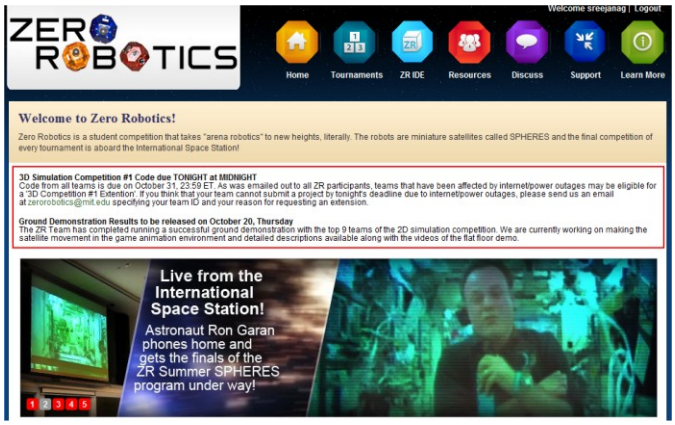



**Figure 9: Zero Robotics Website, look designed by the storyboard contest**

The contests to design the look and feel of the website (Website wireframe and storyboard contests) as well as contests to design the name and logo for the Zero Robotics games highlights the 'creative input' benefit of the crowdsourcing model. Evaluation was done and prizes were awarded based on MIT's judgment with input from TopCoder. While the storyboard competition did very well, the design of the logo did not yield an integrated result satisfactory to MIT, in spite of 12 final submissions. MIT was able to finalize a logo by putting together contributions from 2 winning submissions. Had MIT not been able to do that, TopCoder could have run



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

another logo contest using the winning submissions as inputs, and so conducted an interative development cycle.

While the Studio and software development contests were the main development tools used to further development, Zero Robotics used Top Coder Bug Race contests for fix quick, time-critical bugs. A short problem statement and the appropriate section of design or code was released for each competition and the first competitor to satisfactorily submit a fix is awarded a prize. Bug Race competitions can be used for quick changes, .short tasks that didn't get done during a contest, and integration of solutions from parallel contests – they are essentially Studio contests where the first member to submit an acceptable solution wins. The Bug-Race tracking system allows clients and reviewers to easily create request into order to obtain the specific fixes required. These competitions typically range from about one day to a week long and by design have significantly less participation than the development contests. The 'Bug Race' competitions have takers because the task is very specific and needs quickly available, specific skills. The participant works closely with the person who submitted the ticket and resolves the problem. This capability highlights the 'crowd production' benefit of the crowdsourcing model.

It is worth mentioning that the crowdsourcing model used by TopCoder for Zero Robotics is different from other online staffing outsourcing resource sites that are availabile such as oDesk or eLance in that those sites allow their customers to hire a specific person for a job, follow up with him and pay him after completion. the focus is on selecting an individual, and the competition is in the candidate selection process rather than the solution selection process. Also, in those models every contractor typically gets paid rather than only the winners. The Bug Race competitions differ from this model in that they are a request for a deliverables, rather than a specific person, even though the result is that a small number of individuals completed most of the tasks.

## *8.1. Contest Participation*

The participation in the contests for the development of Zero Robotics was generally what would be expected. There were 54 Studio and software contests in 12 broad categories held among members of the TopCoder community between April 2011 to December 2011. These contests cumulatively received 857 registrations (notice of intent to participate), 149 full submissions and 57 prizes for these contests were awarded. There have been a total of 239 unique participants in the 54 contests.

Figure 10 shows data from the 54 contests. The contests have been sorted in the order of occurrence in the development cycle shown in Figure 6. Registration represents the amount of initial interest in the contest and submissions represent the final output from the contest, of which one is chosen to move forward per contest. Specification contests that include making wireframes, storyboards, web design and application front end design as well as the assembly contests attracted the highest number of registrants possibly due to the large number of people who possess the required design and software skills. Component production contests include prototyping tasks. On the submissions side, conceptualization is lowest, possibly due to the specificity of the task (abstraction of the given project required rather than execution of a defined task using pre-existing skills such as design). It will be shown later using Figure 12 that the submissions number and prize value turns out to be correlated because the prize values are determined by the market, to induce the desired levels of participation.

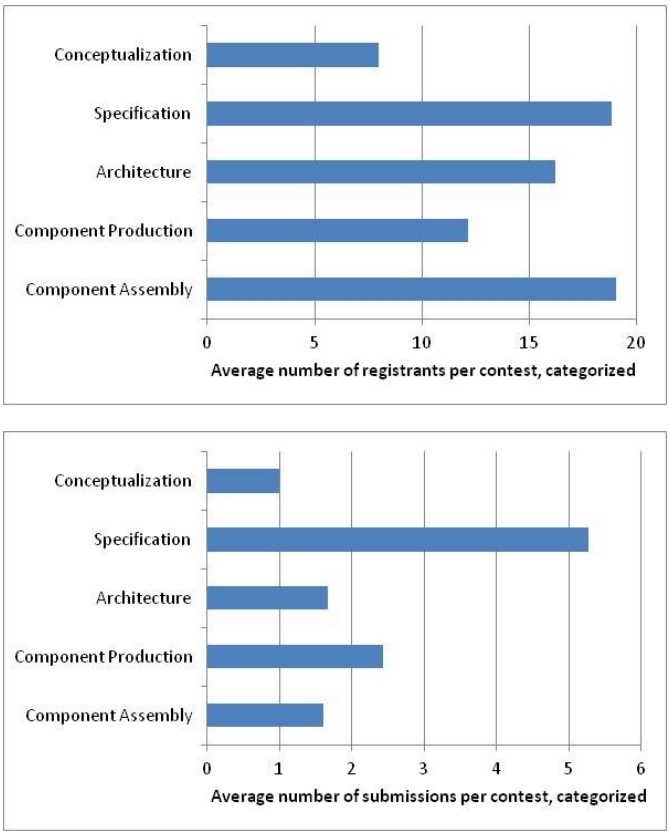



**Figure 10: The average number of users that registered (top) and submitted valid solutions (bottom) per contest, arranged by broad contest category**

Architecture contests, which involve discussing the software requirements with the client and reviewers, documenting them in detail and making test suites and test scenarios, had the most discussion threads on the forums. Architecture contests are also the critical point for technical design, and there were occasions where MIT rejected the winning entries because they did not meet the specifications. The back-end conceptualization and architecture contest was conducted 3 times, and ultimately the community member who won the architecture contest not only designed, but also assembled and supported the back end all through.



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

We also noticed that module assembly had a skewed number of registrations vs. submissions. A disproportionately large number of people registered for these contests. It appears that they gauged their probability of winning by the discussion forum content; and a small subset of the participants ultimately followed through to submit a solution. For example, the User Profile Portlet Assembly contest had 45 unique registrants but was dominated by the community member who won the most assembly contests in the InSPIRE project. This phenomenon was seen in multiple assembly contests – many registrants but eventually 3-4 submissions.

As mentioned above, fixing bugs that are identified in the production software, small changes and integration tasks are performed using Bug Race competitions. by MIT or the ZR website/IDE users are documented in the TopCoder system in the form of a issue report. Unlike, the Studio contests, there is no competition for the best solution of a Bug Race. Instead, community members contact the ZR TC program manager or co-pilot with the request to take up the Bug Race competition and the first acceptable solution is selected to fix the bug. The fixed piece of software is then merged into the existing framework. There were 163 Bug Race competitions between September 2011 and December 2011 with 32 winners.

## 8.2. Contest Prizes

MIT and TopCoder spent $187,260 on prizes for the 54 Studio and software contests and the 163 Bug Race competitions, including reviewer payments and co-pilot support. This is only the payments to the community, and does not include costs for the 2 MIT graduate students, the TopCoder platform, or the TopCoder platform manager. Given that 239 unique members of the community participated in the contests and bug races, from one viewpoint, we were able to 'buy' diversity in participation at the rate of $800 per user over a period of about half a year. However, among the participants (counted as those who registered for a crowdsourcing contest or a reviewer), there were 90 individuals who won prize money. TopCoder therefore paid an average of $2000 per winning competition member over the 6 month period, although the payments were skewed toward larger amounts to a smaller group. Therefore, the number of people working on our problems was far greater than the number of people we paid. This does raise the concern of retention since making any money is based on a probability of success. However, since all participants have access to the discussion forums and members' histories, they are expected to make educated predictions on their win and participate accordingly. As shown in previous literature, access to complete information actually encourages the participation of the strongest contenders.

Figure 11 captures the 54 Studio contests run over a period of 7 months in terms of the number of unique members who registered to participate i.e. expressed interest to compete and the number of complete solutions submitted at the end of the contest. The contests have been arranged in decreasing order of efficiency, defined as the ratio of submissions to registrants. The overall efficiency over all the contests was ~ 15% and the figure visually indicates a large number of contests that have an abnormally low efficiency, which can be due to a variety of reasons. The user profile portlet assembly contest and back-end architecture contests have low numbers because the pool of potential participants contained a member (different for each of the 2 contests) who was known to have a nearly 100% winning streak in Zero Robotics contests. As a result, the other participants backed out after gauging a lowered chance of winning. On the other hand, the highly efficient contests like the game name and logo design contest was a very creative one that did not require very specific skills and all the participants competing in the category had no prior history with ZR. Low efficiency can be a source of concern since it potentially indicates failure to retain the captured interest in a contest and additional effort is required to increase active participation such as increasing the prize money, advertising on the TC website or actively reaching out to skilled members. This is especially required for contests where there are no strong competitors in the participant pool.

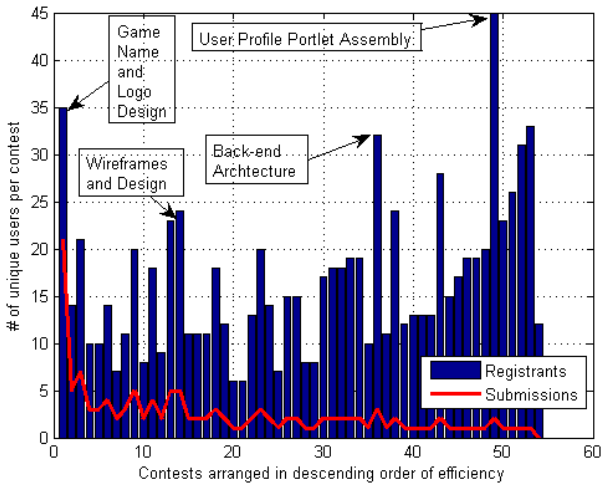



**Figure 11: Number of users per contest for the Zero Robotics Development Program**

Figure 12 shows the prize money distributed for the development of products in each of the categories listed. The vertical blue line marks the average money paid per payment, which is $356 (525 payments were made, including co-pilot and reviewer payments. Contest categories as conceptualization or conceptualization are rewarded much higher than the average prize money in order to attract members to participate in them, in a market based determination of awards. Contests that



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

appeal to a broader skillset (as seen earlier by the number of registrants in Figure 11) such as prototyping i.e. component production and deployment did not require as high a prize for gaining potential interest. The number of contests for conceptualization and architecture is also far lesser than, say, assembly. Correlation with Figure 10 shows that contests that had the lower number of submissions (e.g. Conceptualization) required the highest value of prizes and those that had higher number of submissions (e.g. specification and component producation) had lower amount of prizes.

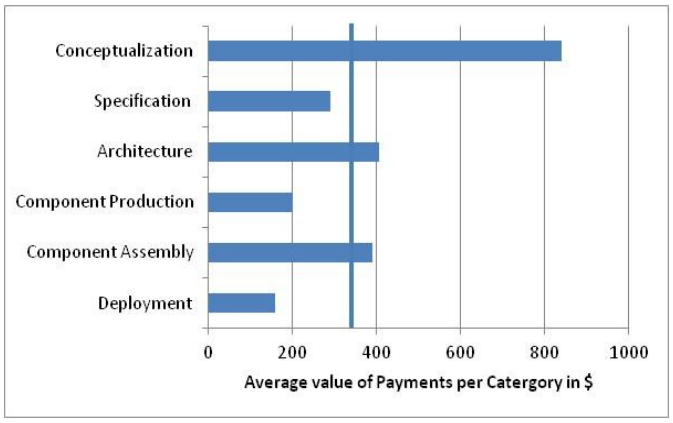
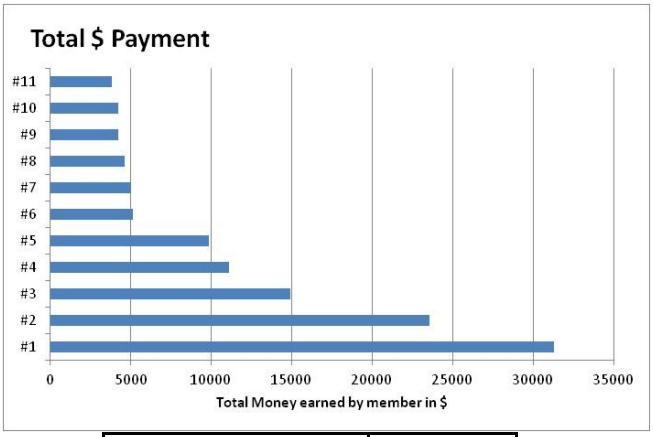



**Figure 12: Dollars spent as prize money for each contest category The blue vertical line is the mean of all the contest prizes run through December 2012.**

From the participants' point of view, a participant dominating the contests can find a good source of income, no matter which category he chooses to dominate in. This leads to loyalty that is very useful because not only does it retain the good quality participants but also provides a field for Bug Race competitors. Figure 13 shows the cumulative earnings of the top 11 earners in the ZR crowdsourcing contests. These 11 highest earners among the 90 total winners claimed 62% of the total money spent on all the payments. The individual who dominated the assembly contests (maximum in number and average in prizes) claimed nearly 26% of the total prize and reviewer money in assembly contests. Since the number of assembly contests is high, there was opportunity for other participants to compete for the dominating position and make significant prize money. Moreover, 4 of the top 11 winners are those who dominated the assembly contests where in the lower 3 have claimed upto 5% of the assembly prize money. The member who won the initial architecture contest for designing the back-end of the IDE was also went on to architect the entire back-end and, since the back-end is the heart of the ZR simulator, he monopolized all subsequent back-end contests as well. As a result, 100% of the back-end prizes were awarded to that individual. The individual who dominated the architecture contests claimed nearly 45% of the architecture prizes. This appears to be a direct result of the fact that architects need to clearly understand the client requirements and document them precisely in order to do well in the contests. The table in Figure 13 shows three of the highest earning categories (as established in Figure 12) and the percentage of the total earnings in that category that was claimed by the participant who claimed the highest in that category. Very obviously, it pays very well to be a "loyal" participant.




| **Category** | % of total payment category |
|---|---|
| Conceptualization | 68% |
| Architecture | 45% |
| Assembly | 26% |

**Figure 13: Prize money in $ of the top 12 community members in terms of total earnings and the % of monopoly in each category**

The "loyal" participants have been consistently conversing with MIT on the TopCoder forums over many contests and are well versed with the ZR framework, increasing their chances of winning contests due to their subject matter expertise. From the perspective of the customer, the phenomenon of "loyal" participants reduces the effort of educating new participants on the background of the ZR framework. For this reason, TopCoder provides incentives in order to encourage member loyalty. Apart from domination opportunities in contests as seen in the statistics above, loyal members (as evaluated by their 'reliability rating' and contest participation) are given an extra payments in addition to the per-contest prize money. While this seems to favor partial monopolization of a market that is inherently supposed to be competitive in order to produce quality, the caveat is that the groups of people who dominate the contests are self-chosen from all around the globe, who have competitively established their position through the process of crowdsourcing. It would been much harder, if at all possible, to find such a match by looking locally for such a candidate, hiring him full-time and managerially requiring that he keep up his standards of work – all



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

possible because of the modular crowdsourcing framework.

*"Competition is exciting, but it always contains the seeds of failure. Contests have both winners and losers, with the losers always outnumbering the winners. The extrinsic motivation of competition provides a positive incentive only if a student stands a reasonable chance of success"* - Woolnough, 1994

Figure 14 is a pie chart showing the distribution of Bug Race competition winners. The top 7 Bug Race competition winners correspond to the top 12 members in all the contests put together, as shown in Figure 13. This is different from traditional managerial assignment in that members volunteer to participate in the races as and when they are available.

Overall, we saw reduced participation of TC members from the thousands available in the community to a few dozen that regularly submitted to the ZR contests, this trend of survival of the most powerful contestants in the presence of complete information is predicted in theoretical crowdsourcing literature [22]. Here it attracted the best of the pool to compete as well as benefited, although relatively little compared to the winners, the newcomers. The incentive of very high rewards combined with detailed feedback is expected to motivate newcomers to climb the learning curve, if they think it possible, after which the TC loyalty benefits keep them engaged and involved in their area of expertise. The availability of detailed member performance records on the TopCoder website provides the community with the advantage of transparent information to make an informed decision on what works best for them.

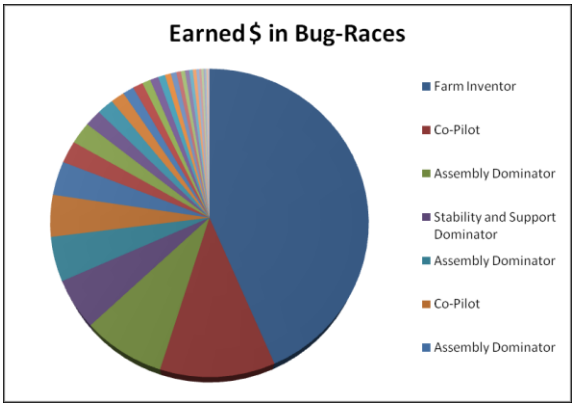



**Figure 14: Prizes earned by members in the Bug Race contests. The earnings have been sorted in descending order before plotting and the top 7 highest earning members listed using their aliases**

### *8.3. Product Quality*

The quality of the ZR web interface can be judged by its ability to perform load tests successfully and by the satisfaction of the students who are using it to participate in ZR tournaments.

Since the first tournament on the new web interface launched in September 2011, the website has seen >480,000 page views where over 70% are returning users. There are 1800 account holders who have among themselves created and saved >254,000 project revisions and run 100,000 simulations on the IDE (all data as of January 2012). The 'Farm', designed and developed by a TopCoder member and shown in Figure 1, has yielded a robust framework for handling and managing multiple requests to the SPHERES simulator from clients simultaneously, which is key to managing crowds of users writing and simulating SPHERES software online. The Farm controller has the ability to relay the requests to 24 available processors currently, a number that can be scaled by adding more virtual machines one the cloud. The website initially had stability issues which irked the users (57% of 109 polled users called it their biggest complaint) however the issue was resolved within 3 weeks using Bug Race competitions and dedicated member support. Currently, the website is supported by multiple servers, with the ability of adding more. Traffic can be directed to different servers by a load balancer. The numbers indicate that crowdsourcing has indeed yielded a stable web environment that successfully supported the tripling of ZR's web usage from 2010 to 2011.

An online survey was conducted at the end of the 2011 tournament season to access student and mentor feedback after they used the developed web interface. Of the 30 alumni students who responded to the survey, i.e. those who had participated in 2010 on the prototype web interface and returned to participate in 2011 on the web interface developed through crowdsourcing, 63% rated their website satisfaction and 75% their IDE satisfaction more in 2011. More specifically, website satisfaction in comparison with 2010 scored a mean of 3.64 with a standard deviation of 1.24, where 3 was 'no difference'. IDE satisfaction scored a mean of 3.94 with a standard deviation of 0.8 – which implies that, even considering that a fraction of alumni responded to the survey, we can be more than 68% confidant that the alumni population preferred the crowdsourced website. These numbers already indicate improvement and more improvement is expected in 2012 using the lessons learned from the 2011 pilot program.

## 9. CONCLUSION

The paper successfully demonstrates the applicability of crowdsourcing to the development of the SPHERES Zero Robotics program software infrastructure using



978-1-4577-0557-1/12/$26.00 ©2012 IEEE

TopCoder's crowdsourcing methodology. Crowdsourcing was conducted along with established techniques of collaborative competition among TC's community of members. Members developed components of a large software system in stages, incentivized by prizes. Within this competitive framework, collaboration was mandatory for certain aspects such as supporting future contests and encouraged for other aspects such as help within community forums. We introduced further collaboration by sometimes combining multiple winning entries of contests into one and working one-on-one with community members.

Key crowdsourcing benefits identified in literature were revisited and their pros and cons identified for smoother future operations. For example, development does not depend on the knowledge or availability of any particular individual reducing single point of failures. However, when individuals begin to dominate in particular tasks, they may get indispensible for critically related tasks (e.g. farm development, stability resolution issues and critical assemblies). Access to a global pool of solvers is possible providing diversity innovation benefits however, getting a point across to people from different cultures and language can be time-consuming and carry a high risk of miscommunication and possibly faulty submissions. The judging process involves several experienced reviewers gloabbly to ensure quality but the process is long and makes it difficult to meet critical deadlines. Individuals self-select their tasks so are motivated however, on the other hand, if the potential participants have more lucrative opportunities, participation will drop. The quality standards of time and quality are best met with the number of active participants is at a healthy number.

Large crowds of amateur users are currently using the developed software to program real satellites on the International Space Station and contributing to developing spaceflight algorithms of use to MIT, NASA and DARPA. The project described in the paper therefore demonstrates the ability to develop spaceflight algorithms using the 'wisdom of crowds' such that even the platform to develop the algorithms can be developed by the crowds themselves. Statistics from the program, deliverables from the contests and the overall lessons have contributed to the learning process to design better crowdsourcing contests to develop more complex software for opening up spaceflight simulations to large crowds of contributors.

## 10. REFERENCES


[1] J. Howe, Crowdsourcing: A Definition, June 2, 2006: http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html

[2] F. Kleemann, G. Vob, K. Reider, Un(der)paid Innovators: The Commercial Utilization of Consumer Work through Crowdsourcing, Science, Technology and Innovation Studies, 4(1), 5-26

[3] Alvar Saenz-Otero, "Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station", Doctoral Thesis, MIT, June 2005

[4] J. Enright, MO Hilstad, et al, "The SPHERES Guest Scientist Program: Collaborative Science on the ISS", 2004 IEEE Aerospace Conference (Paper #1296), Big Sky, Montana, March 7-12, 2004

[5] A. Melchoir et al, "More than Robots: An Evaluation of the FIRST Robotics Competition Participant and Institutional Impacts", Brandeis University, Waltham, MA, April 2005

[6] Alvar Saenz-Otero et. al, "ZERO-Robotics: A Student Competition aboard the International Space Station", Proc. IEEE Aerospace Conference. paper #11263, Big Sky, MT 2010

[7] S. Nag, J.G. Katz, A. Saenz-Otero, "The SPHERES Zero Robotics Program: Education using Games", International Astronautical Congress, Cape Town, South Africa, October 2011.

[8] Dava Sobel, Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time, Walker & Co., 1995

[9] F. Aftalion, A History of the International Chemical Industry: From the "Early Days" to 2000, Chemical Heritage Foundation, 2005, pp. 11-13.

[10] C.A. Lindburgh, The Spirit of St. Louis, 1953

[11] XPrize Foundation, Ansari X Prize, http://space.xprize.org/ansari-x-prize

[12] Office of the Press Secretary, President Obama Lays Out Strategy for American Innovation, Sept. 21, 2009, http://www.whitehouse.gov/the_press_office/president-obama-lays-out-strategy-for-american-innovation/; Office of Science and Technology Policy, A Strategy for American Innovation: Driving Towards Sustainable Growth and Quality Jobs, Sept. 2009

[13] P. Orszag, Memorandum for the Heads of Executive Departments and Agencies, Dec. 8, 2009, http://www.whitehouse.gov/sites/default/files/omb/assets/memoranda_2010/m10-06.pdf

[14] America Creating Opportunities to Meaningfully Promote Excellence in Technology, Education, and Science Reauthorization Act of 2010, 111 H.R. 5116 (Jan. 5, 2010).

[15] TopCoder website: http://www.topcoder.com/nasa




978-1-4577-0557-1/12/$26.00 ©2012 IEEE

[16] TopCoder problem statement: http://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=14481&pm=11313

[17] TopCoder forum posting, June 24, 2010, http://apps.topcoder.com/forums/?module=Thread&threadID=678649&mc=8&view=threaded

[18] J.Ford, Exclusive Interview with PDS Idea Challenge Winner: elenashutova, July 13, 2011, http://community.topcoder.com/ntl/?p=493

[19] McKinsey & Co., And the Winner is…, July 2009, http://mckinseyonsociety.com/capturing-the-promise-of-philanthropic-prizes/

[20] Sadler Enginering 2000, citing Woolnough, B. E. (1994). Effective science teaching. Philadelphia: Open University Press

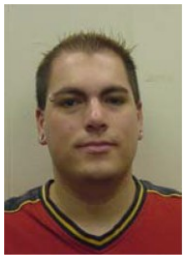
[21] TopCoder Reliability Rating Reference: http://apps.topcoder.com/wiki/display/tc/Component+Development+Ratings

[22] G. V. Ranade and L. R. Varshney, "To Crowdsource or Not toCrowdsource?," submitted to 2012 Collective Intelligence Conference, Cambridge, MA, Apr. 2012.

## 13. BIOGRAPHY

Sreeja Nag is a graduate research assistant in the Space Systems Laboratory (SSL) at the Massachusetts Institute of Technology and leads the SPHERES Zero Robotics Program. She is a dual SM Candidate in Aeronautics & Astronautics Engineering along with Technology & Policy at MIT. She has summer research experience with NASA JPL in 2008 and the European Space Agency (ESTEC) in 2010. Email: sreeja_n@mit.edu

Ira Heffan: As General Counsel of TopCoder, Ira Heffan is responsible for administration of TopCoder's government programs, including the Zero Robotics effort. He hopes that everyone at the conference who reads this will consider mentoring a Zero Robotics team next season. (zerorobotics.mit.edu) Ira received a BSEE from Union College, an MS in Computer Science from Boston University, and a JD from Stanford University.



Alvar Saenz Otero is a Research Scientist at the MIT Aeronautics and Astronautics Department / Space Systems Laboratory. He is the lead research scientist of the SPHERES program. His research concentrates on development of facilities for technology maturation, space systems engineering, and avionics research. He has PhD in AeroSpace Systems, and MEng and BS degrees in Electrical Engineering and a BS degree in Aero/Astro, all from MIT.

Mike Lydon joined TopCoder in April of 2001 as the Chief Technology Officer. Mike is responsible for development of TopCoder's community development and competition infrastructure. Mike holds a business degree from the University of Connecticut where his focus was on Information Systems.



978-1-4577-0557-1/12/$26.00 ©2012 IEEE