2006-2484: ENGINEERING A NATIONWIDE ENGINEERING DESIGN CONTEST

Eugene Ressler, U.S. Military Academy

COL Eugene Ressler is Professor and Deputy Head of the Department of Electrical Engineering and Computer Science at the U.S. Military Academy. He teaches computer science and has also served as the Academy's Associate Dean for Information and Educational Technology. He is a recipient of the AAES Norman Augustine Award for Outstanding Achievement in Engineering Communications.

Stephen Ressler, U.S. Military Academy

Colonel Stephen J. Ressler, P.E., is Professor and Vice Dean for Education at the U.S. Military Academy, West Point, NY. He earned a B.S. degree from USMA in 1979 and M.S. and Ph.D. degrees in Civil Engineering from Lehigh University in 1989 and 1991. He is a past Chairman of the ASEE CE Division and is a recipient of the ASEE Mid-Atlantic Section Distinguished Educator Award, the Premier Award for Excellence in Engineering Education Courseware, and the EDUCOM Medal for application of information technology in education.

Catherine Bale, U.S. Military Academy

Catherine Bale is an adjunct professor at Mount Saint Mary College in Newburgh, New York. She teaches English and Communications classes and has coordinated the West Point Bridge Design Contest since 1997.

Engineering a Nationwide Engineering Design Contest

Abstract

This paper concerns problems solved and lessons learned while conducting the West Point Bridge Design Contest,[1] with a focus on the design of technology support and operations behind the scenes. The contest is a nationwide, Internet-based competition for teams of one or two students, age 13 through grade 12, culminating in a final round with large cash prizes. In 2006 the contest is in its fifth year. We have previously reported it as a means of engineering outreach.[2] This work, on the other hand, is technical, concerning the engineering behind the contest that allows it to be run by a half-time administrator and two college faculty members working in their spare time. The design has successfully dealt with challenges including large service demand fluctuations, tied contest entries, participation by ineligible persons "masquerading" as true contestants, hackers, an extortionist, hardware failure, Internet outages, an artificially intelligent bridge optimizer, and other interesting tribulations, all of which were managed without mishap. Hence the goal of this paper is to pass on information useful to anyone contemplating related work, where similar occurrences are likely.

Introduction

The intent of this paper is to document our experience in designing and operating the West Point Bridge Design Contest (WPBDC), a nationwide Internet-based competition that has involved some 70,000 K-12 students over a five-year period. Careful design of the contest rules, the supporting technology, and the roles of support personnel has produced an effective and efficient operation. The original goals for the contest have been met. Moreover, two college faculty members working in their spare time plus a half-time coordinator have administered the contest with only modest additional institutional support and no serious mishap. Accordingly, we will discuss our design methodology, some particular design solutions, and the roles of support personnel that have evolved over time. While these are necessarily tailored to the unique goals and constraints of the WPBDC, many are likely to transfer well and therefore to benefit other, related efforts. We also provide some anecdotes to give the flavor of unexpected challenges that inevitably arose during contest operations and how the contest's design allowed them to be met.

The overarching goal of the WPBDC is to increase *awareness of* and *interest in* engineering among a large, diverse population of middle and high school students. As described in our earlier work,[2] its motivation is to attract young students of the United States to careers in engineering, math, and science in order to mitigate projected national shortfalls in the future. This leads to more specific goals, which are that each contestant should:

- Learn about engineering through a realistic, hands-on problem-solving experience.
- Learn about the engineering design process—the application of math, science, and technology to create devices and systems that meet human needs.
- Learn about truss bridges and how they work.
- Learn how engineers use the computer as a problem-solving tool.
- Have some fun pitting individual problem-solving skills against those of other virtual bridge designers worldwide.

A goal implied in "learning about the engineering design process" is to encourage work in collaborative teams while also allowing individuals to compete successfully.

Technology supports

To achieve our goals, we established the central principle for design of WPBDC: to exploit computer and Internet technology to provide an *engaging* engineering design-build-test experience with *high learning value* at *no cost* to a *large number of participants* and with *low costs of administration*. With due consideration, this principle led directly to four broad categories of technology support for the contest.

Web site. The contest web site provides potential and actual contestants and their teachers with all information necessary to prepare for competition, produce successful designs, and submit them for judging. The current site includes information on purpose, rules, prizes and eligibility to win, schedule, supporting lesson materials for teachers, and results of previous contest rounds. Over time, analysis of questions emailed to the Webmaster has guided additions and refinements including a Frequently Asked Questions page. The contest web site also provides free downloads of the *client software* for the contest, the second technology support.

Client software. The West Point Bridge Designer client software is provided at no cost. It runs on any Windows computer, presenting a virtual design problem in a graphical form that resembles an engineering drawing of a real job site where a pin truss bridge is to be erected across a river gap. The contestant "builds" a virtual bridge, placing joints and members by manipulating a simplified Computer-Aided Design (CAD) user interface. Finally the contestant determines whether her design is *successful* with a single button press that causes a simulated truck to pass over the bridge, presenting it with a realistic load. Designs can be saved as files, which can be re-opened by the client for further design work and submitted to the contest web site for judging.

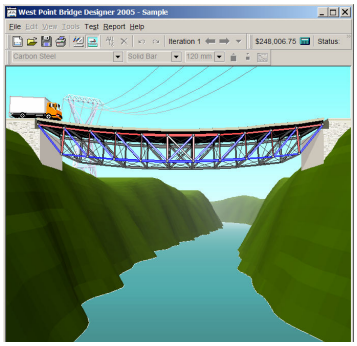

Figure 1—Client simulation.

During the simulation, a lifelike three-dimensional display, shown in Figure 1, depicts the forces in each bridge member with color. Red indicates compression (crushing force) and blue, tension (stretching force). Color intensity shows the fraction of a member's capacity being demanded. Dull red or blue means the member is lightly stressed, while bright color means it is near failure. Thus colors change dynamically with member forces as the truck advances. If a member fails, the simulator approximates the motion of the broken bridge, and the ill-fated truck appears to tumble into the gap. Thus, the animation is an attractive and intuitive display of forces in a truss.

The client employs a simple but realistic cost model to continuously indicate the cost of the structure assembled thus far. The model includes the cost of materials, which depends on cross-section, length, and type of metal used in structural elements. It includes fabrication cost, which depends on the number of joints and joined elements. It also includes site preparation costs determined by shore abutment and pier configurations, which are chosen at the outset by the

contestant from a menu of 56 possibilities. The most *efficient* design is one that is both successful (passes the truck load) and has the least possible cost. The contest is to produce the most efficient design.

The client software captures several important aspects of engineering in an appealing way.

- It shows the connection between the abstract concept of member forces and a real consequence, the truck passing or falling into the gap.
- It requires the contestant to experience the iterative nature of design. The software handily supports design, build, test, and redesign in rapid cycles, and it records the number of such cycles. Top entries normally result from thousands of iterations.
- It reveals the relative ease of creating a successful design versus an efficient one. Nearly anyone who can use a computer can design a bridge that supports the truck load. Top designs can result only from a detailed understanding of structures and the cost model.

In survey studies, these three qualities of the client software appear to be responsible for reports of high learning value.

Automatic judging. Another technology support, intended to make the WPBDC *engaging* by appealing to the competitive instincts of contestants, is the automatic judging feature of the web site. To qualify for prizes, competing teams must register for the contest. In a series of simple web forms, the system establishes eligibility for prizes, gathers team information, and finally provides a home page where the team may log in at any time to submit bridges for judging and see instantly how the team's best design is faring in competition.

Administrator interface. The contest administrator interface is a separate, secure way to access the web site in order to retrieve contest management information, record judging decisions, and post current official standings. Details of the administrator interface are discussed further below. Its intent is to provide for administration with a minimum investment in hours of effort.

Design of the contest

We employed *use cases* as the primary means for collaboratively envisioning the final system.[3] For our purposes, a use case is a narrative describing the interaction of *actors* with the contest technology. We considered interaction to be series of *events*, each consisting of an *action* by some actor followed by a *response* of the technology. We initially considered the following actors:

- Competing teams
- Supporters of competing teams (teachers, mentors, parents, etc.)
- Client software author/maintainer
- Judging system software author/administrator
- Contest coordinator
- Contest judge
- Database administrator
- General system administrator/technician
- Webmaster
- "Bad guy" (malicious Internet entity)

There is no strict relationship between actors and people; an actor in the system may be zero or more people and *vice versa*. The list of actors became longer as design proceeded.

Our methodology was to develop a use case narrative while noting its implications for both contest rules and support technology requirements. We expected technology requirements to follow from decisions about rules. Yet we found that the opposite occurred nearly as often. Requirements for rules followed from decisions about technology. The narrative form of use cases led naturally to "what if?" reasoning about alternatives so that most use cases developed a conditional, branching structure. It was quickly apparent that our most difficult task was to anticipate all possible contingencies. In general, each use case branch fell into one of three categories:

Normal branches described routine interactions of actors with the support technology. An example would be a contest team registering for the contest and viewing its team home page for the first time. Mistakes by users were also considered normal.

Failure mode branches described the experience of actors attempting to use the support technology while some part of it was failed or failing. An example would be an actor attempting to submit a bridge for judging while some part of the system was inoperable.

Malicious branches considered attacks by "bad guys" intending to disrupt the competition or gain unfair advantage. One example we considered was a "denial of service" attack, where a "hacker" would employ nefarious technology to bombard the contest web site with so many requests for service that *bona fide* contestants could not gain access. There were many others.

Use case : Register and submit design		
Action	**Response**	**Notes**
Select "Register and log in."	Show "register and log in page."	* Register and "log in" dialogs must appear simultaneously with good instructions.
Press "register" button.	Show initial registration form.	* Need best practices for form layout. * How to handle multiple team members?
Fill in form correctly and press "submit."	Determine and show eligibility for prizes. Allow user to verify correctness.	* What team data are required? * What are eligibility rules? * Need branches for bad form entries.
User verifies correctness.	Register the team with given eligibility. *Present the modifiable data form.*	** Data entered so far cannot be modified!* ** Need separate form for modifiable team information.*
Enter modifiable data correctly and press "submit."	Present team home page.	* What are modifiable data? All that do not determine eligibility (ex: email address, school, home town). * Need branches for bad form entries.
Browse for bridge design file and press "upload."	Analyze bridge design, verify the truck load passes, compute cost, show home page including results of analysis and contest standing of the design.	* What if the team later submits a bridge not as good as this one? * Need branches for failed load tests, files other than bridge designs including extremely large files that would disable the web server. * What is biggest possible bridge design file? * Standing can only be "unofficial" pending judging.

Table 1—Use case example.

A simplified example, taken from the author's design notes, suffices to illustrate. It is presented in Table 1.

One can see that this was an early, rough use case that led to many branches, questions, and refinements. For example, it made obvious the need for a rule on maximum team size. We settled on teams of only one or two members because we reasoned that young people were unlikely to be productive in larger groups. The rule instantly became part of the software design.

Another important point in the use case development is shown *in italics.* The need for a separate form to collect modifiable team data followed from the need to prevent future changes to data on team eligibility. This became apparent only as the use case was being discussed. The use case narrative developed in parallel with decisions on requirements. Such tight coupling of discovery and consequences within use cases was common. In addition, many use cases implied changes to rules or technology, which affected other use cases. Thus the overall design process was strongly connected and highly iterative. A hypothetical example is shown schematically in Figure 2, where the arrows indicate how one event implies a necessary change to another, either within the same use case or in another.

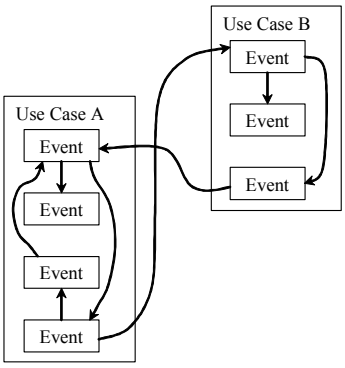

Figure 2—Inter-relatedness of events in use cases.

A partial list of contest rules and software features that resulted from our use-case analysis is as follows:

- A standard annual cycle based on the average US K-12 school year.
- A three-round structure for the contest where each successive round is more closely observed and controlled than the last, while the number of competing teams is geometrically smaller. See Table 2. This arrangement ensures that final winners are deserving, while the highest possible level of qualifying round participation is also achieved. The latter served the goals of maximum learning and broad participation.

Round	Number of teams	Technology supports used	Competition site	Observer
Qualifying	Thousands or millions	All	Any Internet computer	None
Semi-final	Hundreds	All	Mutually agreed observed locations	Teachers and volunteers
Final	Ten or fewer	Client only	Arnold Hall, West Point, NY	WPBDC administrators

Table 2—Three-round structure of contest.

- Mass emailing infrastructure for communication with teams.
- Tied submissions to be avoided by 1) disallowing geometrically identical bridge submissions and 2) by assigning unique sequence number to each successful submitted bridge. If two bridges of identical cost are submitted, the lowest sequence number wins. Rejecting identical bridges creates an interesting technical challenge, discussed below.
- An "open competition" category for curious but ineligible people to try their hand without impersonating a K-12 student by entering fraudulent personal data. Hence we added a new actor to the list, "Curious, ineligible competitor."

- Placing the contest coordinator "in the loop"—reviewing team personal data before posting to official standings pages for "top 30" teams. This avoids offensive information from being posted automatically to the contest web site. It adds a significant administrative burden, but is important to the credibility of the contest.
- A difference between real-time "unofficial" standings, which (for algorithmic reasons) include all reviewed *and unreviewed* teams versus "official" top 30 listings, which include only reviewed teams.
- 100% logging of all web server activity with detail sufficient to "replay" the contest from the logs if necessary.
- Encouragements for teams to log in throughout the contest, which assures sponsors that their investments are paying off in contest activity. These include "bridge design tips" updated weekly and available to teams only through their home pages. In addition, bridge costs are normally *not* listed in official standings so that teams below the top 30 will need to check their home pages to see how their designs are faring.
- "Load dumping." Should publicity cause an unmanageable usage load spike, the administrative interface feature allows easy posting of official standings that *include* costs for the top 30. This would immediately discourage logins by the large majority of teams that do not have highly competitive bridge designs.

Risk analysis

The existence of failure mode and malicious branches in our use cases led us to a systematic consideration of risks in the design, implementation, or operation of the contest and its supports.

Participation risk recognizes that problems with the system can lead participants to give up, subverting the goal of attracting large numbers.

Learning risk is defined as the danger that system problems might interfere with learning about the engineering design process, truss bridges, and computer design tools.

Disruption risk is the possibility that an unrecoverable technology problem can prevent a fair conclusion of the contest with the selection of final winners.

Embarrassment risk is entailed with the association between the WPBDC and the U.S. Military Academy. Should there be even the perception of a less-than-successful outcome for the contest, there would follow an institutional price to be paid.

Failure/risk crosswalk

With risks enumerated, we set out to analyze the failure mode and malicious use case branches with respect to each kind of risk. Conceptually, we constructed a matrix with one axis representing possible problems and the other the kinds of risk along with its likelihood. Each cell was filled with a *risk management decision*. For our purposes, a risk management decision is a (possibly empty) list of mitigation measures that trade off risk for implementation cost. A few rows of the table are shown in Table 3.

Failure mode/ malicious branch	Likelihood	Risk			
		Participation	Learning	Disruption	Embarrassment
Offensive team data entered for a top 30 team	Very high	Low risk; no action			High risk; follow up team data with school personnel
Hacker intrusion	Very high	High risk; take defensive action			
Client bug	Very low	Low risk; redistribute repaired client		Moderate risk; make strong integrity checks on uploaded files	Moderate risk follows from disruption; same action
Spiking participation	Low	Moderate risk; make services rapidly scalable			
Health failure of admin team	Low	Moderate risk; no action			
Solution clustering	Unknown	High risk; use 56 cost-comparable design cases.	High risk follows from participation; same action	Very low risk; no action	Low risk; no action

Table 3—Risk crosswalk matrix.

Solution clustering occurs if the bridge design problem inadvertently leads to a relatively small and obvious set of solutions that are all near-optimal. In this case, many teams quickly arrive at similar solutions, the leader board becomes static, and there is less incentive to participate. Mitigation consisted of offering 56 different shore abutment and pier configurations and then taking the greatest possible care that near-optimal designs for each configuration would all have similar costs.

Specific design decisions taken as a result of risk analysis but not shown above include:

- Use of fully redundant hardware with real-time backup of the contest database.
- Use of the institution-standard enterprise database engine for all team and uploaded design data and "borrowing" of a skilled database administrator for setup.
- Stationing server computers in power and atmosphere-controlled machine rooms and borrowing an expert technician to maintain their basic operating systems.

Unforeseen requirements

Despite our care with use case and risk analysis, several unforeseen requirements appeared during the first two contest years. A discussion of these illustrates how the initial design was changed on-the-fly to meet them. In several cases, responding to participant requests in this manner substantially improved the contest.

Annual contests. In fact, the WPBDC was initially intended to be a single event rather than an annual one. The year 2002 was the Bicentennial Year of the Military Academy, and the WPBDC was conceived as a fitting celebration of the Academy's engineering heritage. Successive years were added only in response to requests from teachers and students and the willingness of financial supporters to continue. To redesign the system for additional contest years, we reconsidered existing use cases in the new light. New ones were added to describe the work necessary between the finals at the end of one contest year and the next year's qualifying round. These included creating a new design problem by making changes to the truck load and cost

model, changing the client and server software to suit, archiving the completed year's data, and resetting the contest database.

Archive analysis. To minimize risk from solution clustering, the completed year's bridge submissions were searched for the minimum cost bridge in each of the 56 shore abutment and pier configurations. These were used to ensure that a winning bridge could not be obtained using the same shore/abutment configuration in the following year and to make the other 55 configurations equally likely to produce winning designs. This approach was successful. In the two most recent contest years, several different configurations were represented among qualifying round winners, who advanced to the semi-finals.

The COPA. Two months before the first qualifying round, legal review by a prospective contest supporter made us aware of the Children's Online Protective Act[4] (COPA) and its provisions. Our widely distributed advertisements had already promised that all U.S. K-12 students would be eligible for prizes. Yet the COPA required written permission from a parent or guardian for children less than thirteen years old before personal data could be collected via our electronic registration forms. We responded by adding use cases for children of this age. The registration system was modified to provide the COPA permission form and ask the contestant to certify that the form had been signed and mailed prior to finishing registration. Modifications to the server software were relatively simple. To the contest coordinator's list of duties was added the retrieving and storing the COPA forms that accumulated in our post office box rented to receive them. After the first year, contest rules were changed so that children younger than 13 were no longer eligible for prizes.

Special reports. Several groups including state engineering societies and school districts requested custom reports of participation in their geographical areas. Since the system was based on an enterprise database engine, it was straightforward to generate a daily report, accessible through the web site, showing the numbers of competing teams by zip code. This satisfied nearly all the individual requests and was implemented in about 24 hours.

Local contests. One request for special information could not be met by the zip code report. This was to provide the standings of teams participating in a statewide bridge design contest that had been scheduled to "piggyback" on our national one. Without our help, the state would be faced with a cumbersome manual method of deciding winners. We determined that such requests for local contest standings could be met if each participating team entered a unique code word in an optional registration form field (we chose the name of the team's teacher or volunteer mentor). On the server side, we began generating hourly local contest standings pages with web addresses based on the code word. We informed the local coordinator of this address. Thus we found that we could support a virtually unlimited number of local contests with the only administrative burden being to issue local contest codes through e-mail to the local coordinators. This simple idea proved very successful. Over 200 local contest codes have been issued. Server records show that approximately three-fourths of these have had three or more participating teams, the largest over 1000. Groups including home-school clubs, classrooms, schools, school districts, professional society chapters, states, and foreign countries have conducted local contests. In following years, the administrator interface of the judging system has been augmented to manage codes and coordinator information.

Bridge data obfuscation. The 2002 and 2003 client software saved bridge data in a readable format, which was easy to modify with a text editor or generate with a separate computer program. By design, the client software made few checks of data integrity as it read these data files. The server, on the other hand, carefully checked submitted files to ensure with perfect certainty that each successful submission could have been produced by the client. This eliminated some kinds of risk and avoided arcane and unverifiable rules about how submitted files must be produced.

After two contest years, there was strong evidence that several groups were constructing automatic bridge designers—heuristic search algorithms using artificial intelligence techniques. All groups known to us were pursuing legitimate research, and none were finding success. Nonetheless, there was high risk of contest disruption if any such effort, legitimate or not, succeeded. Therefore, as a precaution, bridge files for the 2004 contest and beyond have been stored in a scrambled form that would require a high level of technical sophistication to decipher.

Design of support technology

Our use case and risk analyses provided clear requirements for support technology. We list them here for reference.

Correctness. All client and server software needed to function in accordance with use case requirements and the contest rules. While the client was already mature in 2002 and had been in daily use by hundreds of people for some years prior, the server software was new. Hence in addition to best practices in implementation, a comprehensive server software testing program was added to mitigate risk.

Robustness and reliability. Software, hardware, and network equipment had to provide adequate service consistently to all participants and administrators.

Availability. The contest web site had to be consistently available except during scheduled maintenance hours, which were timed to be outside school hours in all U.S. time zones.

Response times. In accordance with best practices for user interface design, the web site had to respond to user interaction in less than ½ second. We deemed Internet-induced delays to be unavoidable and ignored them.

Simplicity of administration. Due to constraints on administrative support personnel, administration had to be simple and possible from any Internet computer. Indeed, the fourth year of the contest took place while the judging system administrator was in Afghanistan, performing his tasks remotely.

Moderate hardware and network costs. We sought to keep equipment and communication costs low. On the other hand, where additional or more expensive equipment could reduce administrator hours or mitigate high and moderate risk, the best decision was usually to purchase.

Skill environment. Development languages and tools employed were those familiar to the software authors at the time the project first started in 1999. This had consequences, as will be discussed below.

Usage load estimation

Nearly all of these requirements hinged on one independent variable— the rate of requests to the web server. Finding no help in the literature, we proceeded with an educated guess. According to the 2000 census, there were approximately 51.5 million K-12-age children in the U.S. and about 92,000 primary and secondary schools. Earlier downloads of the pre-contest client software numbered about 67,000. We settled on the following estimates:

- 100,000 teams would register.
- 1,000,000 bridges would be submitted,
- 4,000,000 registration and login interactions would occur.

We assumed interaction would be spread over 8 hours of each contest day. Using a rudimentary M/M/1 queuing model, we determined that a service time per interaction of 0.3 second would result in an average queue wait of 0.2 seconds, providing the desired 0.5 second response. However, we suspected that spikes would occur when the contest was advertised in metropolitan newspapers and other media with large audiences as planned for the Bicentennial. Some further back-of-the-envelope calculation indicated that a 0.03 second service time provided an acceptable performance margin. The same calculations indicated that an inexpensive 0.4 megabit per second Internet uplink would serve all purposes except downloads of the client software. The client has therefore been distributed through volunteer educational institutions, including ours, through their high-bandwidth connections to the Internet.

Special technical requirements

A few fascinating problems in software design are inherent in the rules of the contest. One is the need to reject bridges that are duplicates of previous submissions. It is not sufficient to check that bridge file contents are identical. These files are necessarily based on an arbitrary numbering of truss joints. Member ends are specified with these joint numbers, and members may also be listed in any order. Thus a bridge with n joints and m members has at least $n!m!$ possible bridge file representations, a large number. Moreover, a new bridge must be checked against the existing database of up to one-million others in approximately 0.02 seconds to meet service time requirements.

To achieve adequate performance, we used two well-known tools of computer science. We first implemented a function to compute a *canonical variant* of any given bridge. A canonical variant in our case is a numbering of joints and an ordering of members unique for a given bridge geometry. We chose left-to-right, bottom-to-top joint ordering and then ordered the members by the smallest of its two joint numbers ascending. Hence to compare two bridges for identical geometry, we first convert them to canonical form and then compare the variants for exact equality.

The second technique needed for rapid duplicate checks is a *hash function*. In our case, the hash function translates a bridge into a short string of characters such that two unequal bridges are very likely to produce different strings.

With these in hand the algorithm for duplicate checking is as follows:

1. Convert the new bridge B to its canonical variant $C(B)$.
2. Compute $H(C(B))$, the hash string for the canonical variant.
3. Search the database for all bridges M_i with stored hash string equal to $H(C(B))$.
4. If no such bridge is found, go to 6.
5. Otherwise convert each bridge M_i to its canonical variant $C(M_i)$ and check whether $C(B) = C(M_i)$ for any i. If so, a duplicate has been found, otherwise continue.
6. There is no duplicate. Store the pair B and $H(C(B))$ in the database.

Since a standard database engine can look up a hash string very rapidly, and canonical variants and hash strings are also quick to compute, this algorithm successfully met the performance requirement.

A second challenge was determining the unofficial standings of any team in a population of 100,000, also in less than 0.02 seconds. Our enterprise database was inadequate for this task, since its relational engine needed a linear scan of 100,000 records in the worst case. A well-known balanced tree algorithm with node numbering was well-suited, but implementation presented some arcane technical problems. Help came from the Open Source software community in the form of a production-quality embeddable database system with the required node-numbering feature.[5]

Bearing in mind that our usage load estimates were rough, we set out to implement the server software for *scalability*. We chose an architecture of communicating services that each provided a separate function. In the system's original configuration, all services were located on the same server computer. If load grew beyond estimates and performance suffered, it would be possible to quickly distribute services on separate computers. Some could also be replicated on any number of computers to further share and balance loads. A diagram of the server organization is presented here.

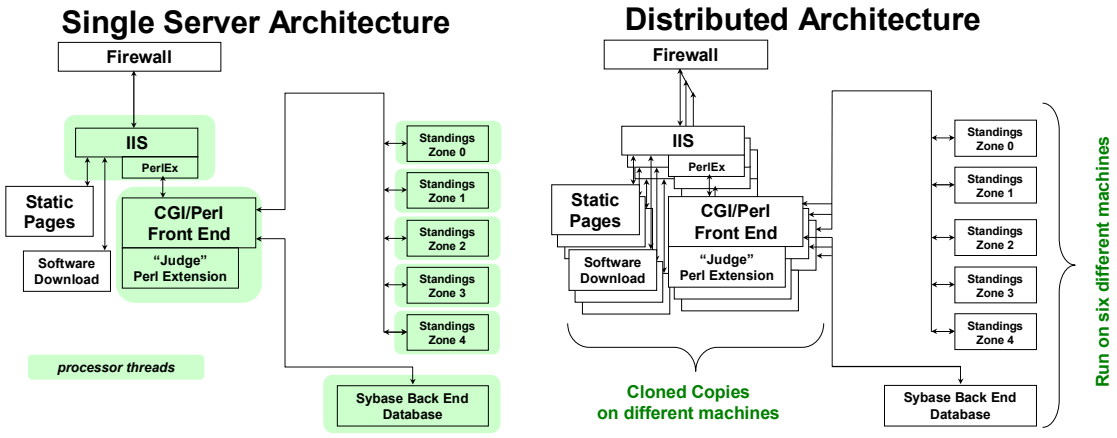

Figure 3—Scalable services architecture for the contest server.

Today, service-oriented systems are common due to the wide acceptance of industry standards such as CORBA, XML, and SOAP.[6] This was not true when the WPBDC was designed. The choice to use services has proven a good one. Though scaling of the system by distributing and replicating them has not been necessary to date, the capability to do so is powerfully reassuring. In addition, though our original implementation used only two Open Source software components—BerkelyDB[5] and perl,[7] the Open Source movement now provides versions of all the WPBDC service components. Were we beginning today, we could choose Linux[8] rather than Microsoft Windows 2000,[9] the Apache[10] web server rather than Internet Information Server,[11] modperl[12] rather than ActiveState PerlEx,[13] and PostgreSQL[14] rather than Sybase Enterprise Server[15] to duplicate the current architecture at no cost for software licenses. In addition, we could replace the hand-written communications code in the standings servers with a SOAP service provider for a simpler implementation.

Administrator support

The administrator interface of the contest web site is secured by password and provides various supports to the administrative team, which are also depicted in the typical screen shown here.

- Server status and consistency checks.
- Verification that the server can be accessed from a third-party Internet location.
- Review of "top 30" team information for offensive content and other issues; approval or disapproval of eligibility for prizes.
- Preview and posting of official standings for approved, eligible teams.
- Viewing of currently posted standings.
- Simple queries to find arbitrary teams by team name.
- Viewing sketches of the best bridges of any set of teams.
- Adding, removing, and searching for local contest codes and associated coordinator data.
- Producing e-mail distribution lists for top 30 teams.

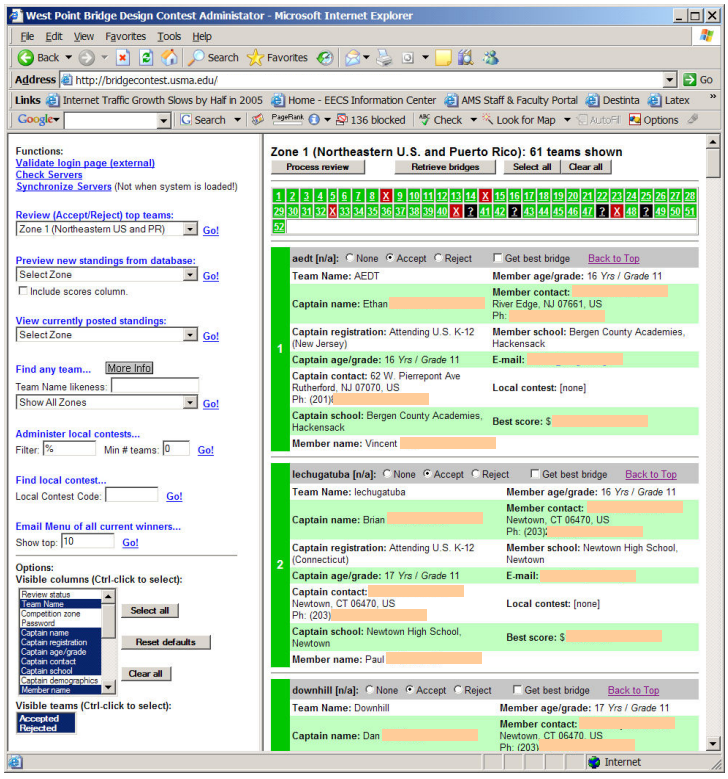

Figure 4—Administrator interface.

These functions have not changed since the second contest year, when local contests were added. A typical administrator screen is shown here. Personal data have been elided.

Administrative support team roles

The division of labor and authority over the administrative support team has evolved slowly to a specific set of roles. These are filled by three people as already explained plus modest institutional and volunteer support. In Table 4, the main support personnel are denoted by A, B, and C. The reader should take careful note that the routine time estimates are for the contest's fifth year of operation, after much learning and reorganization of work. At the outset, they were roughly three times higher.

		Time estimates	
Role	**Personnel**	**Routine**	**Task**
Webmaster	A	2 hr/wk	100 hrs
Client software author	A	—	800 hrs
Client software maintainer	A	—	20 hrs/yr
Judging system software author	B	—	500 hrs
Judging system software administrator	B	2 hr/wk	*varies*
Contest coordinator	C	20 hr/wk	80 hrs/yr
Chief judge	A	1 hr/wk	*varies*
Database administrator	B	2 hr/wk	—
General system administrator/technician	Institutional support	2 hr/wk	—
Local contest coordinator	Volunteers	—	*varies*

Table 4—Administrative support team roles.

The *webmaster* is a conventional author and maintainer of the static information portion of the contest web site. The *client software author* independently created the West Point Bridge Designer. Annual design changes and bug fixes fall to the *client maintainer*. Similarly, the judging system author and administrator respectively created and continuously operate the judging system. The *contest coordinator* is the human voice and face of the WPBDC. She telephonically verifies the administrative data of each top 30 team. At the start of the qualifying round, this is a large daily task. She makes decisions to qualify or disqualify teams, referring those that are not clear-cut to the chief judge. She arranges semi-final round sites and monitor personnel at locations throughout the U.S., on ships afloat, and in foreign countries. She plans, organizes, and executes the contest finals including travel of finals teams to West Point, live competition in an arena-like venue, distribution of prizes during an awards banquet or luncheon, and reimbursements for travel. To the *chief judge* falls the final adjudication of decisions not within the coordinator's purview. He interprets rules and officiates at the finals. The *database administrator* is a standard support role; he performs routine monitoring and preventative maintenance on the enterprise database engine of the contest support system. The *general system administrator* is another standard role; he keeps server and network hardware and operating system software in good repair and up to date.

As shown in the rightmost two columns of Table 4, time spent by contest administrators may be divided into routine and task-oriented work that may be scheduled or unscheduled. Routine work occurs each week from the start of the qualifying round through the completion of finals. Scheduled tasks are generally aimed at preparation for the next contest round. Exceptions are the

tasks of the webmaster and software authors, which reflect the effort of initial development. Unscheduled tasks result from unpredictable events such as software bugs and misbehaviors of contestants.

Observations, episodes, and lessons learned

We close with a few anecdotes and observations having the flavor of out-of-the-ordinary challenges that seem inevitable as each contest unfolds, beginning with most dramatic.

The extortionist. Among the various misbehaviors of young contestants, one stands out. During the closing days of one contest year, the coordinator received an articulate email message from one of the current contest leaders, call him L, explaining that another person, let us say E, was asking, via pseudonym email, to be sent a copy of L's winning design. If L did not comply, then E would arrange to have L disqualified. L found that he was unable to log into his home page. E had guessed his password, logged in, and changed it. E promised to make good his threat by changing L's team information to include offensive language. Fortunately, the contest coordinator already knew L's school principal and verified that L was an honest competitor. Though E took some measures to conceal his true identity, information provided by L along with the contest server logs were sufficient to identify E with high certainty. The case was turned over to E's principal, and E was permanently disqualified from the contest.

The hardware failure. Another episode occurred in the contest's second year when, despite all precautions, a hardware failure led to corruption of the contest database, and the backup system failed. Fortunately, a skilled database administrator was able to recover about three-fourths of the database using specialized techniques. It was then possible to rescue all but a handful of bridge design submissions by "replaying" the system logs, repeating earlier interactions between teams and the server. In all, this intense effort required 14 hours mostly weekend hours. We saw no measurable impact on the contest. We knew we were lucky. In the following year, we upgraded hardware, improved the backup system, and changed log formats to support easier replaying in the future. No similar incident has occurred. In fact, for the most recent contest year, there was 100% service availability with no errors.

Other events have included software bugs manifest by non-US character sets in both the client and server software (the authors were initially ill-acquainted with international software development), offense taken in the wording of registration forms, Internet worms and outages, and many others that had straightforward resolutions, but nonetheless have constituted the press and roar of contest operations.

Annual software changes. Finally, we relate that, in hindsight, our worst design decision has been the choice of different implementation languages for the client and judge portion of the server, which duplicates load, member force, and cost calculations. Recall that our choice stemmed from the expertise of the implementers. It was made when the contest was planned as a one-time event. The result has been that load and cost model changes between contest years have been implemented twice, once in each computer language. More importantly, it has been necessary to test the two implementations extensively to verify that they produce identical results

in all circumstances. In retrospect, a common language implementation would have repaid the time investment for one of the authors to learn a new language many-fold over the years.

Conclusion

We have presented information about the design of the West Point Bridge Design Contest that ought to be helpful to people engaged in similar work. We described the goals of the contest and how they were translated to a design principle. The principle led us to an overall organization of technology supports. We set out to design these supports and found a mutual dependency between them and the contest rules. We settled on use case methodology as a way to envision both the contest rules and technology requirements simultaneously through iteration. We performed a risk analysis because our use cases indicated substantial dangers inherent in the contest, and we addressed risks systematically as management decisions arranged in a matrix.

With requirements in hand, we determined a key unknown in software design: usage load. We made educated guesses on usage load to guide software design and hardware selection; these proved to be relatively accurate. We elected to use a service-based implementation so that capacity could be rapidly scaled should participation grow beyond estimates, though this has not occurred. We described the algorithms needed to provide real-time feedback on contest standings and to reject duplicate contest entries. We described the administrator support interface of the web site and how the small contest administrative support team divided responsibilities. Finally, we related some stories with the flavor of operating challenges that similar efforts should expect.

Withal, the design and implementation of the WPBDC has itself been an exciting and enlightening engineering experience.

1. http://bridgecontest.usma.edu is the contest web site.
2. Stephen J. Ressler and Eugene K. Ressler, "Using a Nationwide Internet-Based Bridge Design Contest as a Vehicle for Engineering Outreach," *Journal of Engineering Education*, vol. 93, no. 2. April 2004.
3. An original reference is *The Unified Software Development Process*, Ivar Jacobson, Grady Booch, James Rumbaugh, Addison-Wesley. A more readable guide is *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition*, Martin Fowler, Addison Wesley Professional, 2004.
4. *Children's Online Privacy Protection Act*, Title XIII, U.S. Code, Children's Online Privacy Protection, Sec. 1301–1308, 1998.
5. BerkeleyDB, available at http://www.sleepycat.com.
6. A good, modern summary text is Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, *Web Services*, Springer-Verlag, New York, 2004.
7. http://www.perl.org.
8. http://www.linux.org.
9. http://www.microsoft.com/windows2000.
10. http://www.apache.org.
11. http://www.microsoft.com/iis.
12. http://perl.apache.org.
13. http://www.activestate.com.
14. http://www.postgresql.org.
15. http://www.sybase.com/products/informationmanagement/adaptiveserverenterprise.