Nordic Journal of Computing 9(2002), 231–247.

A WEB-BASED SUPPORT ENVIRONMENT FOR SOFTWARE ENGINEERING EXPERIMENTS

ERIK ARISHOLM DAG I. K. SJØBERG
GUNNAR J. CARELIUS
Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway
{erika|dagsj|carelius}@simula.no

YNGVE LINDSJØRN
KompetanseWeb AS
Tullinsgate 6, NO-0166 Oslo, Norway
yngve@kompetanseweb.no

Abstract. The software engineering communities frequently propose new software engineering technologies, such as new development techniques, programming languages and tools, without rigorous scientific evaluation. One way to evaluate software engineering technologies is through controlled experiments where the effects of the technology can be isolated from confounding factors, i.e., establishing cause-effect relationships. For practical and financial reasons, however, such experiments are often quite unrealistic, typically involving students in a class-room environment solving small pen-and-paper tasks. A common criticism of the results of the experiments is their lack of external validity, i.e., that the results are not valid outside the experimental conditions. To increase the external validity of the experimental results, the experiments need to be more realistic. The realism can be increased using professional developers as subjects who conduct larger experimental tasks in their normal work environment. However, the logistics involved in running such experiments are tremendous. More specifically, the experimental materials (e.g., questionnaires, task descriptions, code and tools) must be distributed to each programmer, the progress of the experiment needs to be controlled and monitored, and the results of the experiment need to be collected and analyzed. To support this logistics for large-scale, controlled experiments, we have developed a web-based experiment support environment called SESE. This paper describes SESE, its development and the experiences from using it to conduct a large controlled experiment in industry.

CR Classification:

Key words:

1. Introduction

There is an increasing understanding in the software engineering community that empirical studies are needed to develop or improve processes, methods and tools for software development and maintenance [Basili *et al.* 1986, Basili 1993, Rombach *et al.* 1993, Basili 1996, Tichy 1998, Zelkowitz and Wallace 1998]. The classical method for identifying cause-effect relationships is to conduct controlled experiments where only a few variables vary. Controlled experiments in software

Received August, 2002; accepted October, 2002.

engineering often involve students solving small pen and paper tasks in a classroom setting. A major criticism of such experiments is their lack of realism [Potts 1993, Glass 1994], which may deter technology transfer from the research community to industry. The experiments would be more realistic if it is run on real tasks, on real systems, with software professionals who are representative of the target population of the technology, using their usual development technology in their usual working environment [Sjøberg *et al.* 2002].

For example, in an object-oriented design experiment conducted by some of the authors, a total of 190 subjects participated. Among the subjects, 130 were professional Java developers from nine different consultancy companies; 60 subjects were students. The experiment took place during a two-month period and was organized as 12 separate one-day sessions (each individual participated in only one of the sessions). During the one-day session, each subject had to solve six Java programming tasks on their computer using their usual Java development tool. While this experiment was of larger scale and in many ways more realistic than most software engineering experiments, it posed new challenges:

- The experimental material (e.g., questionnaires, task descriptions, code and tools) had to be distributed to each subject in a timely fashion.
- The experimental design was such that not all the material could be distributed at once. Furthermore, once a subject had solved a task, the solution had to be collected immediately.
- Because each (professional) developer was sitting at his or her usual office location while participating in the experiment, it was crucial to monitor the progress of each individual.
- The results (e.g., answers to questionnaires and Java program solutions) had to be stored for future analyses.

The logistics involved in running experiments such as the one exemplified above motivated the need for a tool that could automate some of that logistics. Consequently, we developed the web-based Simula Experiment Support Environment (SESE) in collaboration with an external development company. SESE allows us to define experiments, including all the detailed questionnaires, task descriptions and necessary code, assign subjects to a given experiment session, run and monitor each experiment session and collect the results from each subject for analyses.

The remainder of this paper is organized as follows. Section 2 gives an overview of the activities and logistical challenges of conducting realistic experiments. Section 3 motivates and describes the SESE development project. Section 4 presents the functionality and architecture of SESE. Section 5 describes the experiences from using SESE. Section 6 concludes.

2. Logistics of conducting software engineering experiments

Our research group aims to make software engineering experiments resemble real world situations and thus possibly generalize the results to industrial practice. An experiment is realistic if the situation presented to the subjects is realistic and the

subjects react to the situation in the same way as they would do in their usual work environment. In particular, it is a challenge to achieve realism regarding experimental tasks, subjects and environment [Harrison 2000]. Conducting realistic experiments requires good management of the necessary activities. A typical experimental procedure is as follows.

Step 1: *Define experiment:* Design a new experiment with the required

- questionnaires to collect background information (name, affiliation, address, email address, bank account if the subjects are paid individually, education, work experience, etc.),
- PC and tool environment,
- task descriptions, and
- files to be down-loaded, etc.

Step 2: *Define, gather and assign subjects:* Define the kind and number of subjects that should take part in the experiment, and recruit them. Typically, a controlled experiment consists of two or more alternative experimental treatments. The appropriate treatment should be assigned to the respective groups.

Step 3: *Each subject runs the experiment:* Distribute the questionnaires and other relevant documents defined under step 1 to the subjects and ensure that they start the experiment. In many experiments, we need timestamps of when a subject starts read a task description and when the task solution is finished.

Step 4: *Monitor experiment:* To ensure that the subjects perform correctly and that the appropriate data is collected, the researcher will monitor the progress of each subject.

Step 5: *Collect results:* When a subject has finished the tasks, his or her results are collected and stored in a safe place. When all the subjects have finished, the researcher can start the analysis.

The experience from the controlled experiments within our research group, which have involved a total of about 750 students and 300 professionals as subjects[1], is that the logistics around the experiments are work intensive and error prone. General information and specific task documents must be printed and distributed, personal information (bank account, etc.) and background information must be collected, all solution documents must be collected and then punched into an electronic form, etc. This may in turn lead to typing errors, lost data [Briand *et al.* 2001], etc.

3. Developing an experiment support environment

To address the problems described in the previous section, we developed the web-based Simula Experiment Support Environment (SESE). This section describes related work, our collaboration with a software company in developing SESE and our strategy for its further development.

[1] Information about most of these experiments can be found at `www.ifi.uio.no/forskning/grupper/isu/forskerbasen`.

3.1 Related tools

We realized that if we were to scale up our experiments and particularly run experiments with professionals in industry using professional development tools, that is, make our experiments more realistic, we would need a tool that could (amongst others) provide the following functionality:

- Real-time monitoring of the experiment: The researcher must be able to monitor the experiment progress, e.g., to view the current status and preliminary answers given by each subject. In this way, technical problems, misunderstandings or other issues related to the experiment conduct can be resolved quickly.
- Flexible definition of questions and measurement scales: The tool must support the definition of legal values, default values, required and optional fields, text fields, date/time fields, numeric fields and multiple choice fields. To ensure consistency within an experiment and between experiments, it should be possible to base a new questionnaire on a questionnaire "template".
- Automatic recovery of experiment sessions: If a subject experiences technical problems, e.g., a network failure that terminates the connection to the experiment environment, the subject must be able to continue the experiment from the point of which it was interrupted without loss of data.
- Automatic backup of experimental data: The data must be stored on a central server enabling reliable backup-routines.
- Multi-platform support for download and upload of experimental materials and task solutions: The tool must enable reliable download and upload of large files, supporting the various web browsers and, to some extent, utilizing operating system specific functionality for searching, naming and storing files.

We searched for suitable tools and found several web tools developed to support surveys, most of them designed by psychologists (e-Experiment[2], PsychExperiments[3], Survey Pro 3[4], S-Ware WWW Survey Assistant[5], Wextor[6]). Those tools basically distribute questionnaires to the respondents who fill them in online. Then the results are stored in a local database or are sent via emails to the researchers.

More specifically, an overview of how the abovementioned tools support our requirements is given in Table I. Note that the table does not represent a comprehensive evaluation of these tools. Some of them have advanced features that are not supported in SESE, for example, functionality for automated random assignment of subjects to questionnaires and for defining hierarchical questionnaires where the next given question depends on the answer of the previous question.

Another category of related work is supporting tools for distance learning. Such tools allow teachers to organize exam sessions and students to consult the results of

[2] `http://www-personal.umich.edu/ ederosia/e-exp/`
[3] `http://www.olemiss.edu/PsychExps/`
[4] `http://apian.com/survey/spspec.htm`
[5] `http://or.psychology.dal.ca/ wcs/hidden/home.html`
[6] `http://www.genpsylab.unizh.ch/wextor/index.html`

TABLE I: Overview of how existing tools support our most important requirements.

	Real-time monitoring of the experiment	Flexibility of defining new kinds of questions and measurement scales	Automatic recovery of experiment sessions	Automatic backup of experiment data	Multi-platform support for download and upload
e-Experiment	No	Yes	No	No (Data sent by Email)	No
PsychExperiments	No	Yes	No	Yes (Data collected in SQL-server)	No
Survey Pro-3	No	Yes	Partial (duplicate cleanup)	Yes (Data collected in SQL-server)	No
S-Ware WWW Survey Assistant	No	Yes	Partial (resubmit control)	Yes (Data file on web-server)	No
Wextor	No	No	Partial (resubmit control)	No	No

their tests by a web interface [Bagnoli *et al.* 2002]. A more general description of web-based teaching activities can be found in [Hansen and Salter 2001]. The distance learning category of tools, although not specifically designed for supporting controlled experiments, could probably have been used as a basis for developing SESE. The fact that we based the development of SESE on a different platform is mainly a result of practical and strategic considerations, as explained in the following sections.

3.2 Collaboration with a software company

When conducting experiments where up to (so far) 130 professionals take part in the same experiment, the quality of a support tool must be better than what can be expected from prototype research tools. Implementing a tool with the needed functional and nonfunctional requirements is obviously very time-consuming and difficult. Furthermore, a tool needs to be maintained, backup routines need to be in place, it must be reliable, etc. Consequently, we initiated collaboration with a software company that develops solutions for human resource management, KompetanseWeb AS, to develop SESE.

SESE is built on top of KompetanseWeb's standard commercial product, which is used by several large Norwegian organizations. SESE was (and still is) developed through close contact between Simula Research Laboratory (SRL) and KompetanseWeb. The development of the extra functionality required in SESE compared with the standard commercial system is paid by SRL.

Another concern is the ownership of SESE. We ended up with an agreement where SRL is allowed unlimited use and support of SESE (including the necessary human resource management technology). In return, KompetanseWeb is allowed to resell the SESE-module to other companies and research institutes. That is, SRL gets the basic human resource management technology for free; KompetanseWeb gets the SESE-module for free. In the contract between SRL and KompetanseWeb are also agreements to ensure that SRL still can use SESE if KompetanseWeb for various reasons cannot support SESE, e.g., if KompetanseWeb is merged into another company or goes bankrupt.

3.3 Experiment-driven development

Like any sophisticated tool that is actively used, SESE will never be "finished". We continuously suggest improvements and discover new possibilities. The requirements are driven by the actual experiments where SESE is used.

The development project used the OO design experiment (Section 5) as a proof of concept milestone: the first version of SESE had to support the functionality required to conduct that particular experiment. After that experiment, SESE was further improved to support the needs of two other experiments, one on Design Patterns, which was run during a three day period in May 2002 with 44 professionals, and another one on UML with 60 students using a commercial UML tool, which was run during half a day in November 2002. Further experiments on use cases, estimation and other software engineering issues are planned, which in turn will lead to other sets of requirements to SESE.

4. Simula experiment support environment

This section gives an overview of the functionality and technical architecture of SESE.

4.1 Functionality

The following sections elaborate on how SESE provides (partial) support for the five steps of a typical experimental procedure, as described in Section 2. Detailed descriptions and screenshots are provided to illustrate how such a tool can be built.

4.1.1 Step 1 – Define experiment

An experiment consists of a sequence of questions presented in a browser window. In the *Question registration* window (Fig. 1), each browser window is defined as a numbered *Page*, for example (9). On a page, questions are numbered in a hierarchical tree, for example (9.1) and (9.2.1). Questions on a certain level can be grouped as in (9.2). Each page/group/question may have a Norwegian *(Norsk)* and English *(Engelsk)* version. A question has a certain *Answer type*. For the types *Combo Box*, *Check Boxes*, and *Option Buttons*, *Text Labels* are assigned to the question. For the *Date/Time* type, a date/time format is selected (*yy, mm, dd, hh, mm, ss*). A question can be indicated as *Required*. When right-clicking on a page/group/question, SESE displays a menu related to the selected line: *Edit*, *Cut*, *Copy*, *Move* (*Up* or *Down*). *New* (*New Page*, *New Group* or *New Question*) is selected to insert a new page/group/question below the selected line. The left side of the window previews the selected page/group/question.

4.1.2 Step 2 – Define, gather and assign subjects

SESE only supports assigning subjects to experimental treatments. Defining and recruiting subjects are still completely manual operations. Once the subjects have

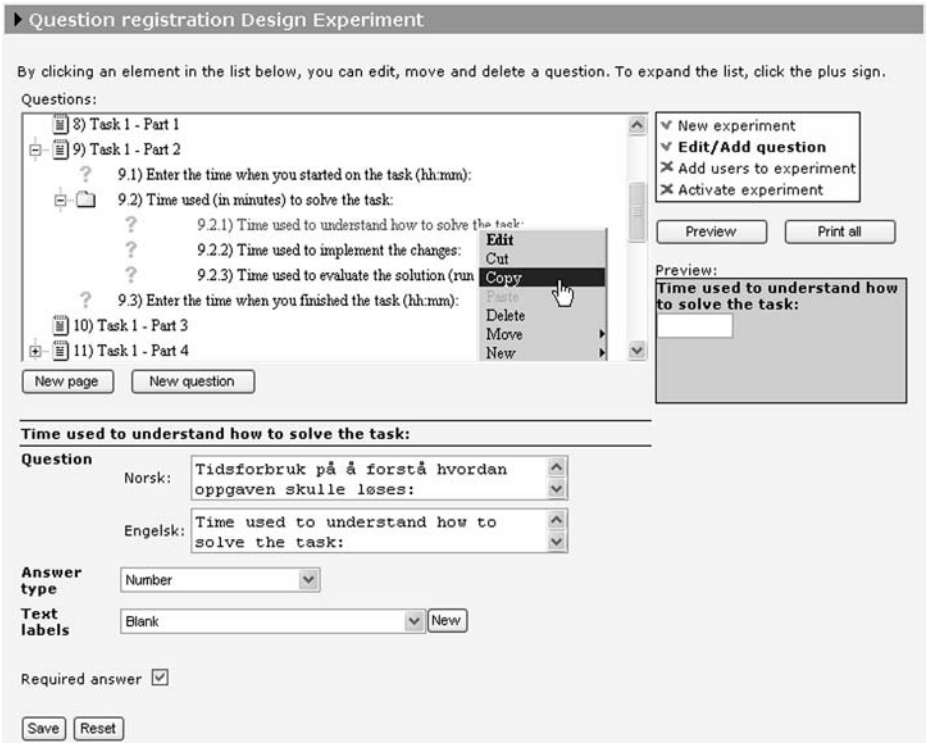

Fig. 1: Question registration window.

been recruited, they are assigned to an experiment in the *Add users to experiment* window (Fig. 2). First, the subjects are found with the search function. A search is done within a selected department or in *All departments*. With the arrows buttons (>> and <<) subjects can be moved in and out of the *Users selected for the experiment* list to the right. The Set start date button is used to prevent selected subjects from accessing the experiment before a certain date. Pressing the *Send mail* button, predefined e-mails containing, amongst others, user name and password are sent to the subjects. The e-mails are generated using a simple ASCII-based email template.

4.1.3 Step 3 – Each subject runs the experiment

The general procedure for running an experiment with SESE is as follows:

1. The subject opens the SESE login window with a web browser and logs onto SESE with the username and password provided by SESE.
2. The subject registers required personal information.
3. The subject starts the experiment that he or she has been assigned to.
4. The subject answers the questions and solves the tasks presented in the browser window.

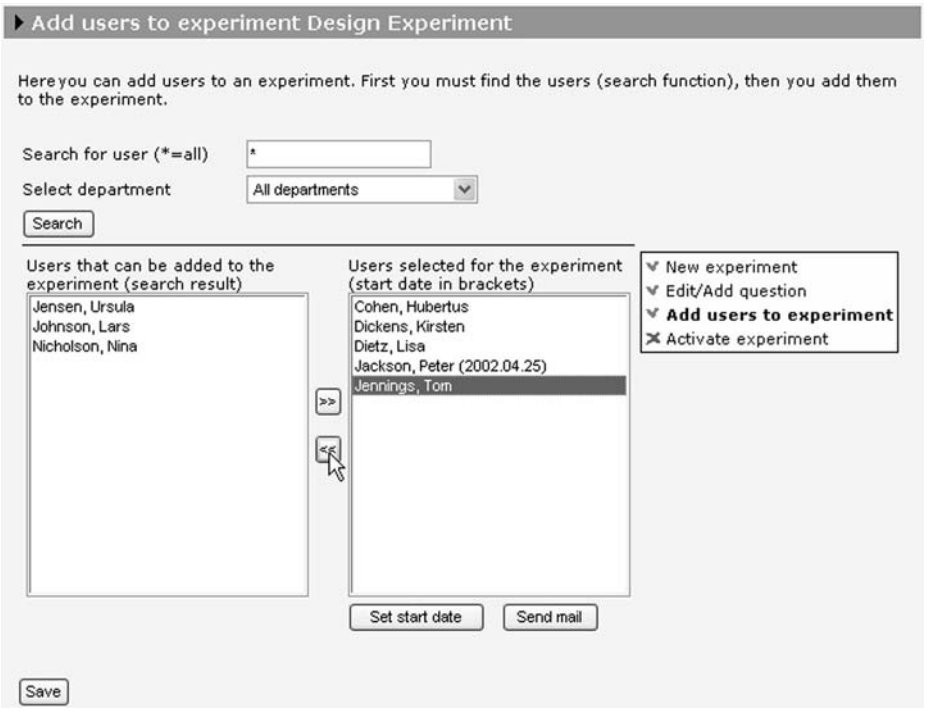

Fig. 2: Add users to experiment window.

If the experiment is interrupted (deliberately by the subject or due to technical problems), the subject will automatically return to the last uncompleted window when the experiment is restarted. At present, SESE enforces the following experimental rules:

- The subject cannot go back to edit answers in a previous window.
- Questions marked with red asterisks (*****) must be answered by the subject.
- An experiment cannot be repeated by the subject once the experiment is completed.

We will illustrate how an experiment is conducted in SESE using one change task of the experiment described in Section 5. The change task proceeds as follows. The subject is asked to download the zipped source code for a program and to unzip the file (Fig. 3). Then the subject must download a PDF-file containing a detailed description of the task to be solved (Fig. 4). The start time, time finished and time used on the different activities of the task must be entered into the appropriate fields. The subject is then asked to zip the subdirectory containing the solution files for the solved task and to upload the zip-file (Fig. 5). Finally, the subject fills in a post-mortem questionnaire related to the change task.

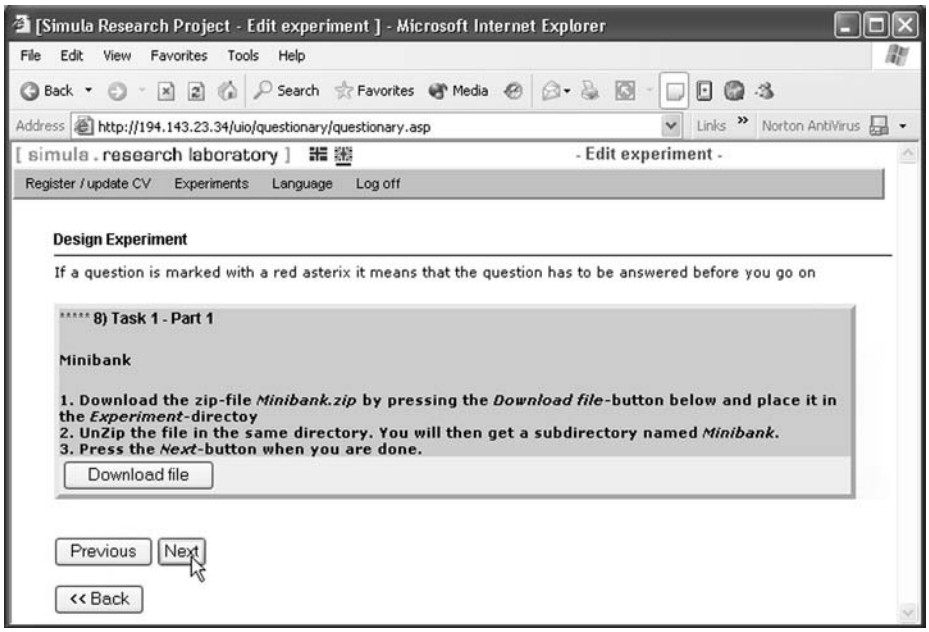

Fig. 3: Question 8, Task 1 – Part 1.

4.1.4 Step 4 – Monitor experiment

The researcher can monitor experiments in real time in the *User status* window (Fig. 6). The *Status* column displays the experiment status of each subject (*Unanswered*, *Started* or *Finished*). The start time for *Finished* and *Started* experiments is found in the *Start time* column. The total time used on *Finished* experiments is displayed in the*Time used* column.

For a *Started* experiment, the number and name of the last page that was answered by the subject are shown in the *Last page answered* column. The researcher can view the answers given by a subject by clicking on the name of the subject in the *Name* column.

4.1.5 Step 5 – Collect results

The results of an experiment can be presented graphically. More importantly, all the raw data for a certain experiment may also be downloaded for further analyses. The questionnaire data are downloaded as a Microsoft Access 2000 database. The task solution files are downloaded as a compressed directory structure identifying each solution file by an experiment-ID, person-ID and task-ID.

4.2 Technical architecture

SESE is deployed on an n-tier client/server architecture, built on Microsoft COM technology (Fig. 7). The SESE application layer runs on one computer and the

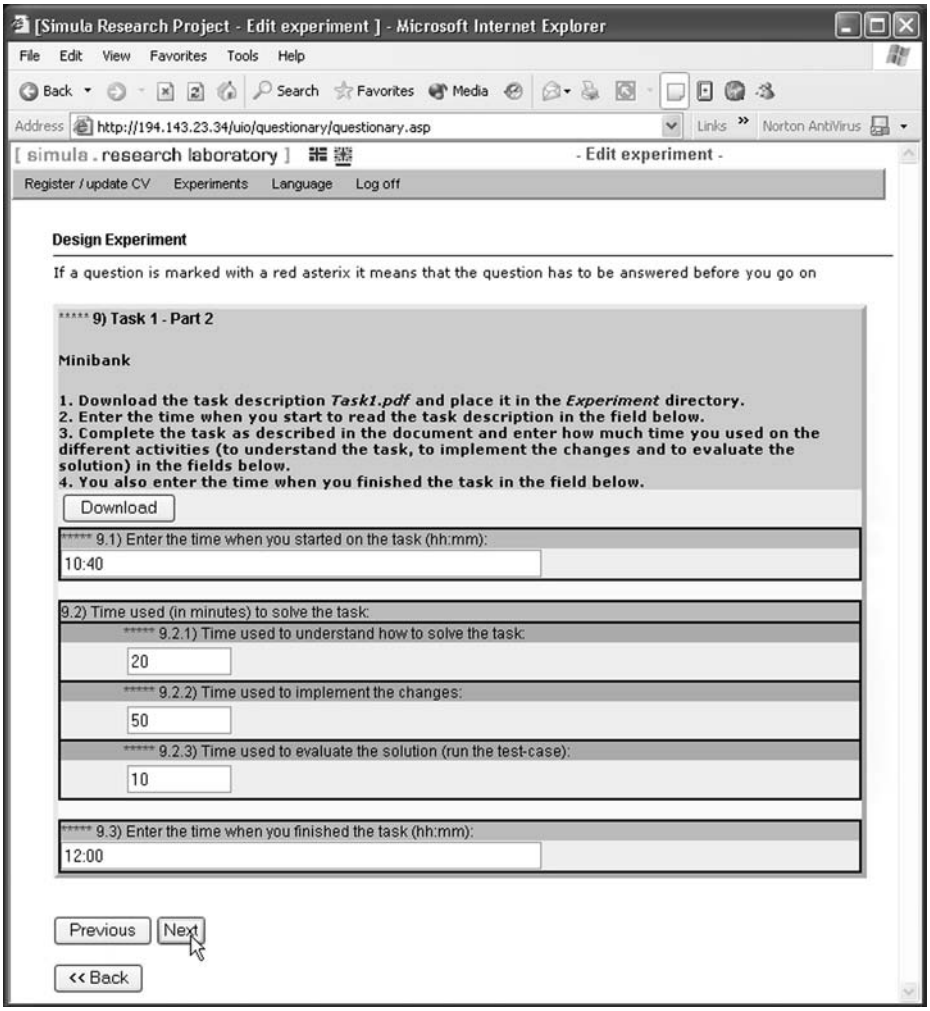

Fig. 4: Question 9, Task 1 – Part 2.

database on another. Users communicate with the application through a standard web-browser (e.g., Netscape and Internet Explorer).

The web pages are built using HTML, CSS (Cascading Style Sheets) and Javascript, and are presented with Microsoft ASP (Active Server Pages). The application/business layer is implemented in Java and contains an object model with methods supporting the operations of the software. The Data Access layer contains classes and methods supporting O/R mapping (Object Read/Write in a standard relational database). The interface with the business layer supports COM+. This layer uses Microsoft's data access technology ADO (ActiveX Data Object). The persistent layer uses an MS SQL-server. The communication with the database is managed by a COM+ component where transactions are initiated by MTS (Microsoft Transaction Server). Scalability in the application layer is configured in COM+ using Load balancing and Object pooling.

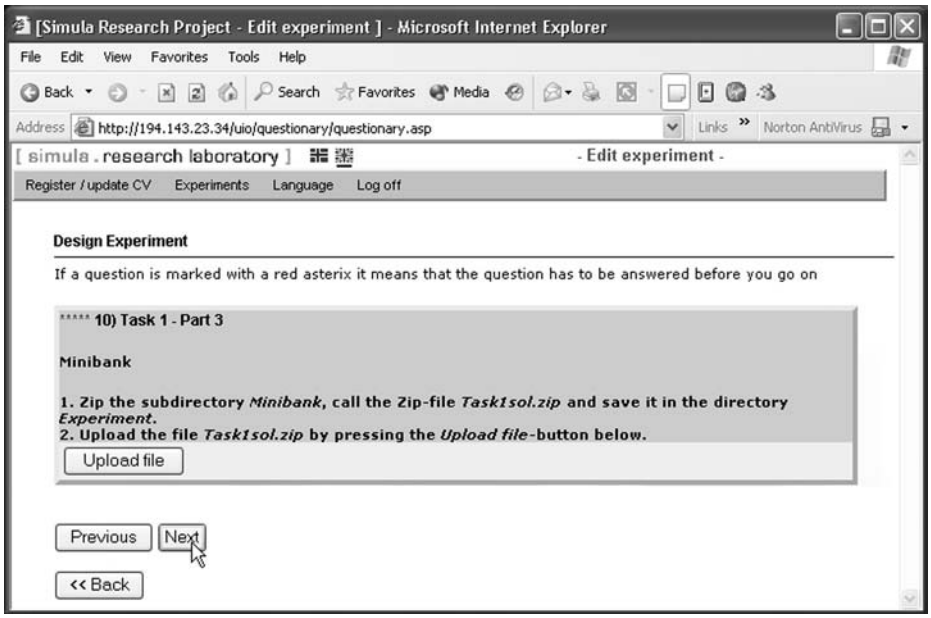

Fig. 5: Question 10, Task 1 – Part 3.

4.2.1 Security

SESE is generally accessible on the Internet. The user ID is verified when the user logs onto the system. Access rights depend on the role of the user. All traffic between the web server and the client may be encrypted using standard SSL with HTTPS. Hence, the data is protected by a firewall, i.e., no other services apart from HTTP (HTTPS) may be accessed from outside.

SESE uses "form-based" authentication. The user fills in username and password in a HTML Form, which is sent to the server with "HTTP Post". If HTTPS is used, the data is encrypted. The user name and password are verified against separate tables.

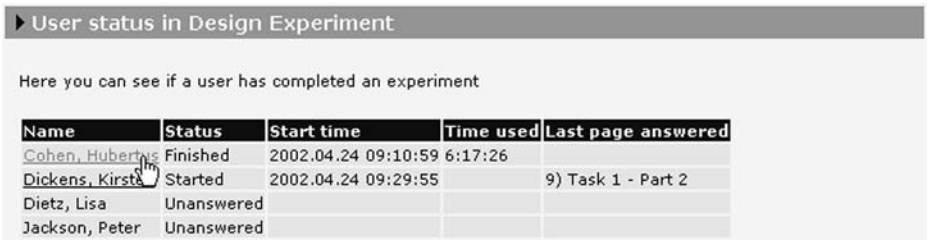

Fig. 6: Monitoring the experiment.

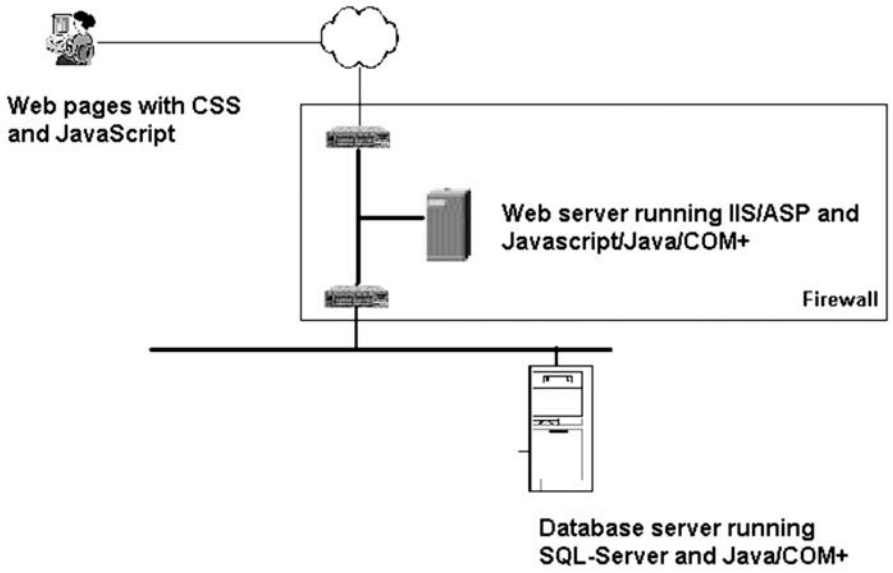

Fig. 7: The SESE client/server architecture.

4.2.2 Sessions

The Internet Information Server (IIS) assigns each user a session object. The session object is deleted either when the user logs off, or after five hours of inactivity. This timeout length is relatively long to support large tasks. The session object stores information such as the user ID and experiment status of each subject in a cache.

4.2.3 Roles

The application applies the notion of roles. All users are assigned one or more roles. All links into the system are assigned a list of roles, stating which users have access to the link. The menus are a collection of links, and will therefore vary according to role of the user who is logged onto the system.

5. Evaluation of SESE

This section describes the experiences of using SESE to conduct a large controlled experiment evaluating how object-oriented design principles may affect changeability. The experiment was a replication of an earlier pen-and-paper experiment using 40 students as subjects [Arisholm *et al.* 2001]. A common critique of pen-and-paper experiments with students is that the results are not valid outside the rather unrealistic experimental conditions; in real development projects, the programmers are professionals, using real development tools in a more familiar

environment. In the replicated experiment with SESE, the goal was to assess whether the external validity of the results would be affected by using

- professional developers instead of (in addition to) students,
- professional development tools and real Java code instead of pen-and-paper exercises, and
- normal work environments (offices or office landscapes) instead of classroom settings.

The remainder of this section focuses on how SESE supported the logistics of conducting the replicated experiment.

5.1 Conducting the replicated experiment using SESE

The experimental materials (e.g., skill level questionnaires, task descriptions, post-mortem questionnaires and Java code) were defined in SESE. In total, 190 subjects participated. Among the subjects, 130 were professional Java developers from nine different consultancy companies. The remaining 60 subjects were students from the University of Oslo. All subjects were paid to participate.

The experiment took place during a two-month period and was organized as 12 separate one-day sessions. The 60 students participated in one common experiment session at a computer terminal facility at the University of Oslo. For the experiment sessions involving professional developers, a local project manager (in each company) was assigned to the "experiment project". He or she ensured that the subjects assigned to a given experiment session actually attended, that PCs and office spaces were available, that meeting rooms had been booked, etc. The project manager also prepared a list of the names and email addresses of each subject that was assigned to a given experiment session. When we received the list from the project manager, the subjects were given a user-id and password in SESE and assigned to one of the two design alternatives. Randomization and blocking were used to avoid uneven group assignments. Then, SESE sent an email to the subjects informing them about their user name, password, how to log on to SESE, and the time of the experiment. Each experiment session started with a short introduction meeting, where the procedure of the experiment was explained to the subjects by the first author. After the meeting, the developers proceeded to their usual office or workstation, logged on to SESE and started the experiment. For all of the experiment sessions, at least one researcher was present.

During the one-day session, each subject had to solve six Java programming tasks on their computer using their usual Java development tool. Most subjects spent somewhere between 5 to 8 hours to complete the experiment. Further details of the tasks and the design alternatives are explained in [Arisholm *et al.* 2001, Arisholm and Sjøberg 2002].

5.2 Lessons learned

This section summarizes what we perceive as the most important experiences and the consequential guidelines for conducting large-scale, controlled experiments with SESE.

5.2.1 Administrative task

Important infrastructure needs to be in place to conduct administrative tasks. Depending on the number of subjects that take part in an experimental session, one or more researchers must be physically present during the whole session to assist in problems or answer questions that relate to both the particular tasks and the experimental support environment. The presence of a researcher also helps ensure that the subjects focus on the actual experiment without external interruptions, or that interruptions are organized in a satisfactory way (lunch breaks, etc.).

The company should use a technical support person to ensure that the PCs have been configured with the required tools and network connections. This is particularly important for those cases where the programmers did not use their "own" PC for running the experiment.

5.2.2 Monitoring

SESE's monitoring functionality was an important and useful tool to assess the progress of each subject during the experiment. However, we would have liked more detailed information about what the subjects were thinking and doing during the experiments, that is, more qualitative information about the individual processes that led to the recorded results. Consequently, we are currently implementing a "write-aloud" screen [Karahasanovic *et al.* 2001], which is a pop-up dialogue asking the subjects to write down what they have been doing since they started on the current task (or since the screen popped up last time) and, in general, give their viewpoints on the experimental situation. The research goal is to get more information that explains the results, in particular unexpected results. The screen can also be used to remind the subjects to follow the procedures of the experiment. The frequency of the pop-up screen can be based on given time intervals or it can be event-driven (e.g., "no user activity for the past 10 minutes"). The leading text of the screen and the pop-up time interval may vary from task to task within the same experiment.

To get even more understanding of the actual behavior of the subjects, we also plan to include logging functionality for window operations, keystrokes, mouse operations and movements logged with timestamps [Karahasanovic *et al.* 2001].

5.2.3 Importance of including a training task

In our experience, professional developers constitute a more heterogeneous group than students. Our results suggest that the variation in skills amongst professionals is *considerably* larger than within a group of second or third-year students. Furthermore, conducting experiments with real development tools instead of pen-and-paper poses additional technical challenges. Consequently, our experiences suggest that, when using SESE to conduct experiments with professionals using realistic development environments, it is crucial to have a training task as a first exercise before initiating the "real" experimental tasks.

During the training task, the subjects familiarized themselves with the experimental procedure (e.g., answering questionnaires, downloading task descriptions and code from SESE, uploading task solutions to SESE, using Acrobat Reader to read task descriptions, using PkZip to uncompress and compress Java code, and coding and compiling the source code). Furthermore, most technical or user-related problems (e.g., having the wrong version of JDK, having an expired license of JBuilder, having an outdated version of Acrobat Reader or incorrect use of PkZip) were resolved before they could have a negative impact on the reliability of the results of the experiment. Most of the technical and user-related problems occurred during the training exercise. In the rare cases where technical or user-related problems occurred after the training task was completed, they were mostly of simple nature and resolved quickly.

5.2.4 Personal interruptions

The professional developers were located in their usual work offices while running the experiment. Consequently, using SESE to support the logistics of such geographically distributed experiments enabled us to increase the realism. However, this increase in realism means that each subject potentially can be interrupted (phone calls, lunch break, etc.). Such interruptions should of course be kept to a minimum to ensure reliable results. To reduce the negative impact of such interruptions, we requested the subjects to limit interruptions to times *between* each change task and explained to them that such interruptions otherwise could threaten the validity of the results of the experiment. We observed that the vast majority of the subjects respected this request as far as practically possible. In cases where interruptions were unavoidable, the subjects used a special "comment" field in SESE to inform us about the nature and time span of the interruption. In summary, based on our experiences from this experiment, we believe that it is possible to ensure that personal interruptions will be kept below the level in which the results of the experiment would be threatened.

5.2.5 Firewalls and virus scanners

Before the experiment, we were worried about whether network security software such as firewalls and virus scanning software would prevent the subjects from downloading and uploading tasks and questionnaires. This turned out to be no problem except for one case, in which a company had a firewall that refused to accept zip-files from external web-sites. However, this issue was resolved during the training exercise so it did not impact the results of the experiment.

5.2.6 Response times, network traffic and server load

Clearly, slow response times or interruptions caused by too high network traffic or SESE server load could threaten the results of the experiment. In particular, it could make the subjects frustrated, which in turn could affect their performance.

Fortunately, with one notable exception discussed below, we did not experience problems related to increased network traffic or SESE server load during the experiment sessions. For the OO design experiment, the change tasks and code were quite small (resulting in a total of approximately 1MB to be downloaded and uploaded per subject). For the sessions in industry, this load caused no problems regarding the response times of SESE. To reduce the risks of network and server-related problems at the SESE server site, we had a technical administrator from KompetanseWeb on call during all the experiment sessions.

For the student experiment session, consisting of 60 students starting the experiment at the same time, we *did* experience a serious server problem: As the 60 students logged onto SESE and started the experiment, the server crashed. Fortunately, the administrator at KompetanseWeb managed to get the server up-and-running after a few minutes. No data was lost (SESE remembers the state of the experiment for each subject), and the remainder of the experiment was conducted as planned. This incidence points out that SESE introduces new risks of a technical nature. Consequently, it may be necessary to have a technical administrator on call at all times to deal with such issues.

6. Conclusions

This paper motivated the need for tool support to run realistic controlled experiments to empirically evaluate software engineering technologies. Realism can, for example, be increased using professionals in addition to students, real development tools instead of pen-and-paper, larger tasks and a typical work environment instead of a classroom. The logistics of running realistic experiments are much more complex than for simple pen-and-paper student experiments.

This paper gave an overview of the functionality and technical architecture of an experiment support tool, SESE. This tool was developed and evaluated in conjunction with a large OO design experiment. Running such large experiments introduces new organizational and technical challenges and risks. If these issues are dealt with properly, our experiences suggest that SESE is an invaluable tool. In fact, without SESE, we believe it would have been infeasible to conduct the OO design experiment.

Several new software engineering experiments are underway in which researchers in our group will use SESE as a backbone experiment support environment. SESE is continually being improved based on the experiences from the OO design experiment and on the requirements of new planned experiments.

Acknowledgements

We are grateful to the students at University of Oslo and the developers from Accenture, Cap Gemini Ernst & Young, Ementa, Ementor, Genera, Software Innovation, Software Innovation Technology and Tietoenator who participated in the design experiment. We thank the anonymous referees for their very constructive comments. We also thank Dag M. Solvoll, Wiggo Bowitz, Kirsten Ribu and

Amela Karahasanovic for their contributions to this paper. This research is partially funded by The Research Council of Norway through the research project INCO (Incremental and component-based software development, project number 140398/431).

References

ARISHOLM, E. AND SJØBERG, D. I. K. 2002. A Controlled Experiment in Industry to Evaluate the Effect of Responsibility-Driven Design Principles on Software Changeability for Different Categories of Professionals (in preparation).

ARISHOLM, E., SJØBERG, D. I. K., AND JØRGENSEN, M. 2001. Assessing the Changeability of two Object-Oriented Design Alternatives - a Controlled Experiment. *Empirical Software Engineering 6*, 3, 231–277.

BAGNOLI, F., FRANCI, F., AND STERBINI, A. 2002. WebTeach: an Integrated Web-based Cooperative Environment for Distance Teaching. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, July 15-19, 2002, Ischia, Italy. ACM, 519–520.

BASILI, V. R. 1996. The Role of Experimentation in Software Engineering: Past, Current, and Future. In *Proceedings of the 18th International Conference on Software Engineering (ICSE)*, Berlin, Germany, March 25-29, 1996. IEEE Computer Society, 442–449.

BASILI, V. R. 1993. The Experimental Paradigm in Software Engineering. In *Experimental Software Engineering Issues: Critical Assessment and Future Directives*, Rombach, H. D., Basili, V. R., and Selby, R., Editors. Proceedings of Dagstuhl-Workshop, September 1992. Volume 706 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 3–12.

BASILI, V. R., ROMBACH, D., AND HUTCHENS, D. 1986. Experimentation in Software Engineering. *IEEE Transactions on Software Engineering 12*, 7, 733–743.

BRIAND, L. C., BUNSE, C., AND DALY, J. W. 2001. A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. *IEEE Transactions on Software Engineering 27*, 6, 513–530.

GLASS, R. L. 1994. The Software-Research Crisis. *IEEE Software 11*, 6, 42–47.

HANSEN, S. AND SALTER, G. 2001. The Adoption and Diffusion of Web Technologies into Mainstream Teaching. *Journal of Interactive Learning Research 12*, 2/3, 281–299.

HARRISON, W. 2000. N = 1: An Alternative for Software Engineering Research?, Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research. Workshop, 5 June, 2000 at 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, 2000.

KARAHASANOVIC, A., SJØBERG, D., AND JØRGENSEN, M. 2001. Data Collection in Software Engineering Experiments. In *Managing Information Technology in a Global Economy*, Information Resources Management Association International Conference IRMA 2001, Software Engineering Track, Toronto, Ontario Canada, May 20-23, 2001. Idea Group Publishing, 1027–1028.

POTTS, C. 1993. Software-Engineering Research Revisited. *IEEE Software 10*, 5, 19–28.

ROMBACH, H. D., BASILI, V. R., AND SELBY, R., EDITORS. 1993. Experimental Software Engineering Issues: Critical Assessment and Future Directions. Proceedings of Dagstuhl-Workshop, September 1992. Volume 706 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.

SJØBERG, D. I. K., ANDA, B., ARISHOLM, E., DYBÅ, T., JØRGENSEN, M., KARAHASANOVIC, A., KOREN, E. F., AND VOKAC, M. 2002. Conducting Realistic Experiments in Software Engineering. In *Proceedings of the First International Symposium on Empirical Software Engineering (ISESE'2002)*, Nara, Japan, October 3-4, 2002. IEEE Computer Society, 17–26.

TICHY, W. F. 1998. Should Computer Scientists Experiment More? 16 Reasons to Avoid Experimentation. *IEEE Computer 31*, 5, 32–40.

ZELKOWITZ, M. V. AND WALLACE, D. R. 1998. Experimental Models for Validating Technology. *IEEE Computer 31*, 5, 23–31.