SESE – an Experiment Support Environment for Evaluating Software Engineering Technologies

Erik Arisholm[#], Dag I.K. Sjøberg[#], Gunnar J. Carelius[#] and Yngve Lindsjørn[*]

[#] Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway
{erika;dagsj;carelius}@simula.no

[*] 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.

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 *et al.* 1993, Rombach *et al.* 1993, Basili 1996, Tichy 1998, Zelkowitz & 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 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 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 and describes ongoing and future work.

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):

- *Realistic tasks*. This challenge is concerned with the size, complexity and duration of the involved tasks. Most experiments in software engineering seem simplified and short-term in which "the experimental variable must yield an observable effect in a matter of hours rather than six months or a year" (Harrison 2000). This is hardly realistic given the tasks of building and maintaining real, industrial software, particularly since many of the factors we wish to study require a significant time period before we can obtain meaningful results.
- *Realistic subjects*. This challenge is concerned with the selection of subjects to perform the experimental tasks, that is, to what extent do the selected subjects represent the population that we wish to make claims about? Even though there are some preliminary indications that students can be used for certain tasks instead of professionals under certain conditions (Höst *et al.* 2000), it is still unclear how well results from student-based experiments generalize to professional software engineers (Harrison 2000). It is worrying, therefore, that most of these studies attempt to generalize their results to an industrial environment.
- *Realistic environment*. Even when realistic subjects perform realistic tasks, they may be carried out in an unrealistic manner. The challenge is to configure the experimental environment with an infrastructure of supporting technology (processes, methods, tools, etc.) that resembles an industrial development environment. Traditional pen and paper based exercises used in a classroom setting are hardly realistic for dealing with relevant problems of the size and complexity of most contemporary software systems.

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.

3. Developing an Experiment Support Environment

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 all 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.

3.1. 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 provide the following functionality:

- real-time monitoring of the experiment
- flexibility of defining new kinds of questions and measurement scales
- automatic recovery of experiment sessions
- automatic backup of experimental data
- multi-platform support for download and upload of experimental materials and task solutions

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.

---

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

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 1. 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

More specifically, an overview of how the abovementioned tools support our requirements is given in Table 1. 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. The remainder of this section describes our collaboration with a software company in developing SESE and our strategy for its further development.

3.2. Collaboration with a Software Company

When conducting experiments where up to (so far) 130 professionals take part, 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. For the current version of SESE (developed from July 2001 to June 2002) SRL paid approximately 400 000 NOK (35 000 US$).

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.

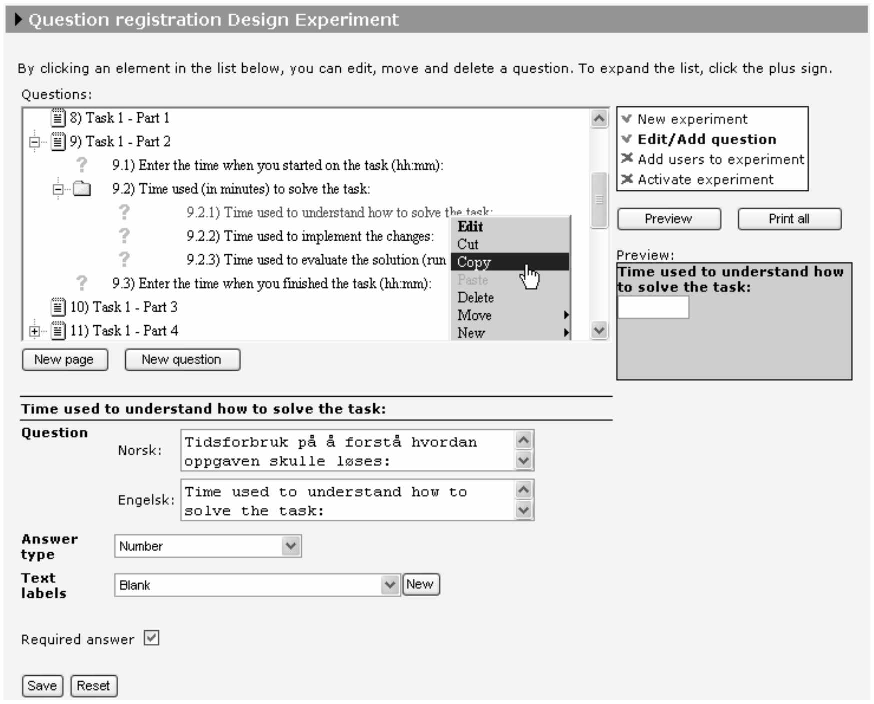
The development project used the OO design experiment (Section 5) as a proof of concept milestone: the first version of SESE had to support the functionality required to conduct that particular experiment. Since February 2002, SESE was further improved to support the needs of another experiment, on Design Patterns, which was run during a three day period in May 2002 with 44 professionals. Thereafter, experiments on use cases, estimation and other software engineering issues are planned, which in turn will lead to other sets of requirements to SESE. For example, we plan to include logging functionality for window operations, keystrokes, mouse operations and movements logged with timestamps (Karahasanovic *et al.* 2001). Thus, SESE is developed using an evolutionary process in which the version used in the OO design experiment can be viewed as the first operational prototype.

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.



Question registration Design Experiment

By clicking an element in the list below, you can edit, move and delete a question. To expand the list, click the plus sign.

Questions:

8) Task 1 - Part 1
9) Task 1 - Part 2
9.1) Enter the time when you started on the task (hh:mm):
9.2) Time used (in minutes) to solve the task:
9.2.1) Time used to understand how to solve the task:
9.2.2) Time used to implement the changes:
9.2.3) Time used to evaluate the solution (run
9.3) Enter the time when you finished the task (hh:mm):
10) Task 1 - Part 3
11) Task 1 - Part 4

Preview:
Time used to understand how to solve the task:

Time used to understand how to solve the task:

Question
Norsk: Tidsforbruk på å forstå hvordan oppgaven skulle løses:
Engelsk: Time used to understand how to solve the task:

Answer type: Number

Text labels: Blank

Required answer ☑

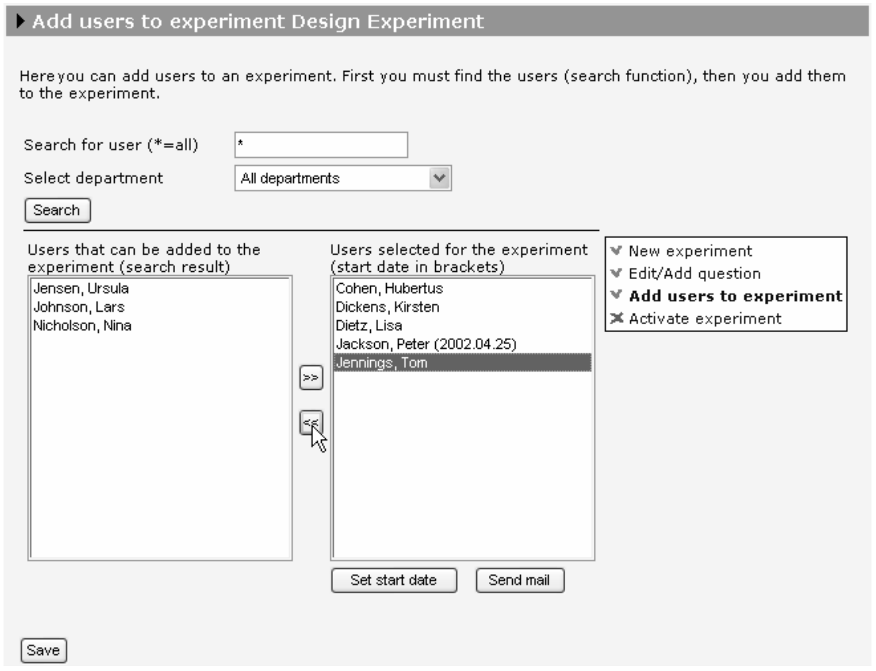
Figure 1. *Question registration window*

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 (Figure 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 been recruited, they are assigned to an experiment in the *Add users to experiment*


Add users to experiment Design Experiment

Here you can add users to an experiment. First you must find the users (search function), then you add them to the experiment.

Search for user (*=all): *

Select department: All departments

Users that can be added to the experiment (search result):
Jensen, Ursula
Johnson, Lars
Nicholson, Nina

Users selected for the experiment (start date in brackets):
Cohen, Hubertus
Dickens, Kirsten
Dietz, Lisa
Jackson, Peter (2002.04.25)
Jennings, Tom

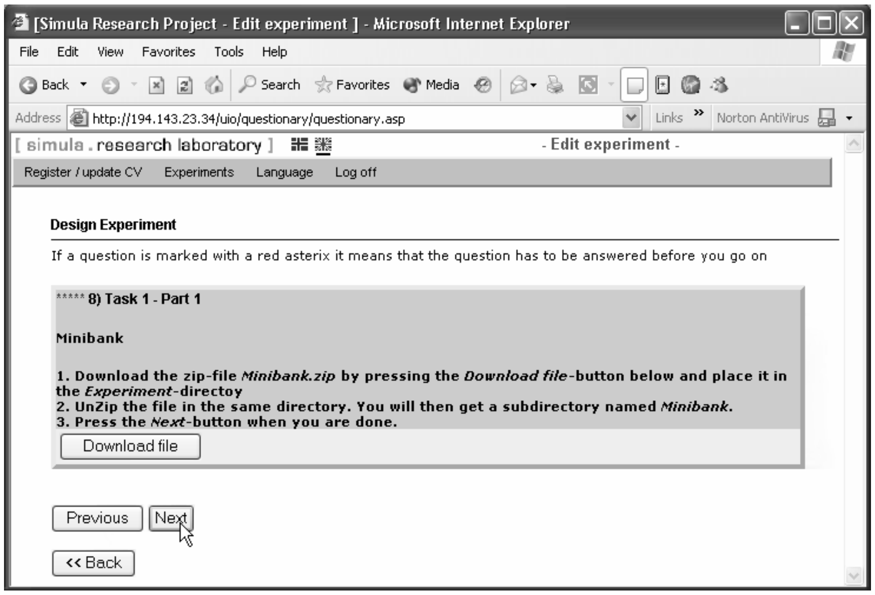
Figure 2. *Add users to experiment window*

window (Figure 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 are sent to the subjects, including user name and password.

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.
5. 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.


[Simula Research Project - Edit experiment] - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://194.143.23.34/uio/questionary/questionary.asp
[simula . research laboratory]
- Edit experiment -
Register / update CV Experiments Language Log off

Design Experiment

If a question is marked with a red asterix it means that the question has to be answered before you go on

***** 8) Task 1 - Part 1

Minibank

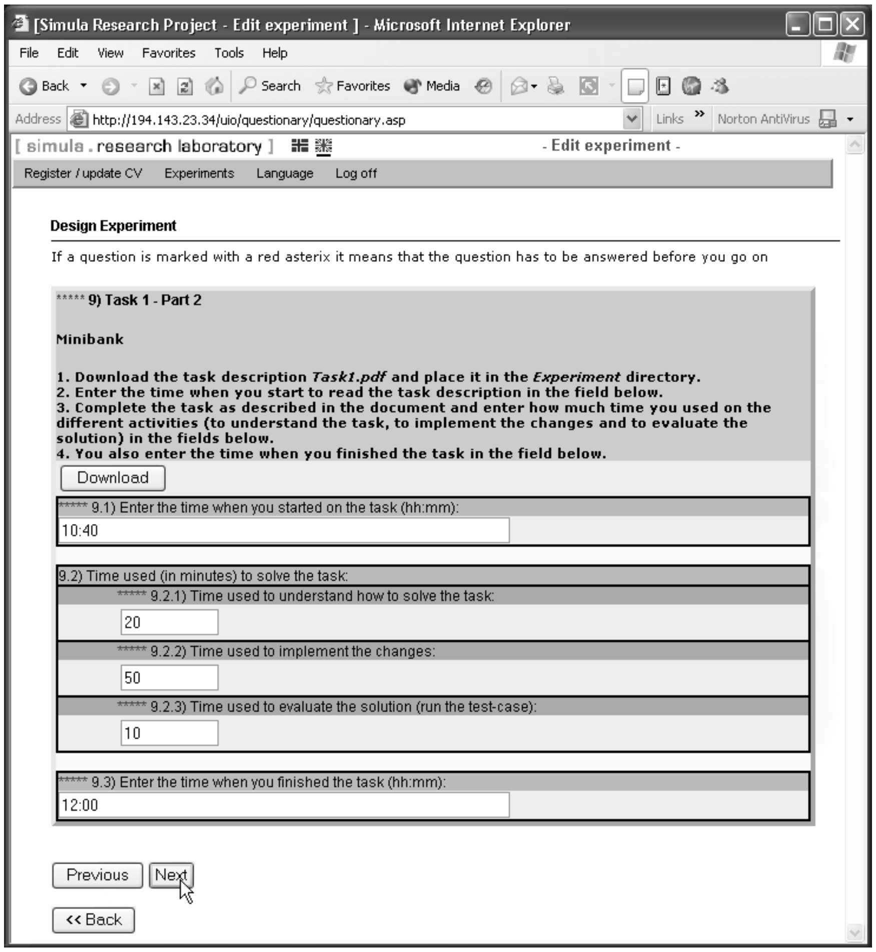
1. Download the zip-file *Minibank.zip* by pressing the *Download file*-button below and place it in the *Experiment*-directoy
2. UnZip the file in the same directory. You will then get a subdirectory named *Minibank*.
3. Press the *Next*-button when you are done.

Figure 3. *Question 8*, *Task 1 – Part 1*

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 (Figure 3). Then the subject must download a PDF-file containing a detailed description of the task to be solved (Figure 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 (Figure 5). Finally, the subject fills in a post-mortem questionnaire related to the change task.


[Simula Research Project - Edit experiment] - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://194.143.23.34/uio/questionary/questionary.asp
[simula . research laboratory]
- Edit experiment -
Register / update CV Experiments Language Log off

Design Experiment

If a question is marked with a red asterix it means that the question has to be answered before you go on

***** 9) Task 1 - Part 2

Minibank

1. Download the task description *Task1.pdf* and place it in the *Experiment* directory.
2. Enter the time when you start to read the task description in the field below.
3. Complete the task as described in the document and enter how much time you used on the different activities (to understand the task, to implement the changes and to evaluate the solution) in the fields below.
4. You also enter the time when you finished the task in the field below.

***** 9.1) Enter the time when you started on the task (hh:mm): 10:40

9.2) Time used (in minutes) to solve the task:

***** 9.2.1) Time used to understand how to solve the task: 20

***** 9.2.2) Time used to implement the changes: 50

***** 9.2.3) Time used to evaluate the solution (run the test-case): 10

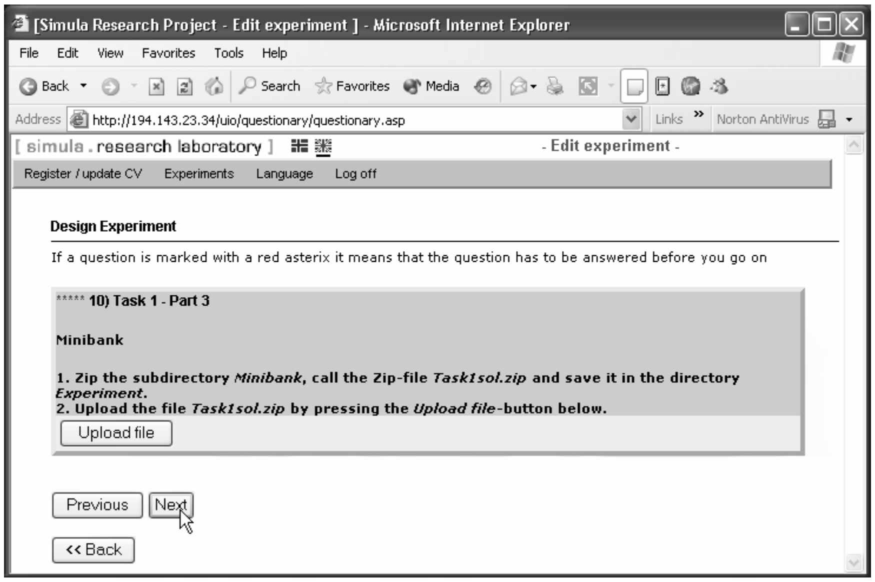
***** 9.3) Enter the time when you finished the task (hh:mm): 12:00

Figure 4. *Question 9*, *Task 1 – Part 2*

4.1.4. Step 4 – Monitor Experiment

The researcher can monitor experiments in real time in the *User status* window (Figure 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.


[Simula Research Project - Edit experiment] - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://194.143.23.34/uio/questionary/questionary.asp
Links Norton AntiVirus
[simula . research laboratory]
- Edit experiment -
Register / update CV Experiments Language Log off
Upload file
Previous Next
<< Back

Design Experiment

If a question is marked with a red asterix it means that the question has to be answered before you go on

***** **10) Task 1 - Part 3**

Minibank

1. Zip the subdirectory *Minibank*, call the Zip-file *Task1sol.zip* and save it in the directory *Experiment*.
2. Upload the file *Task1sol.zip* by pressing the *Upload file*-button below.

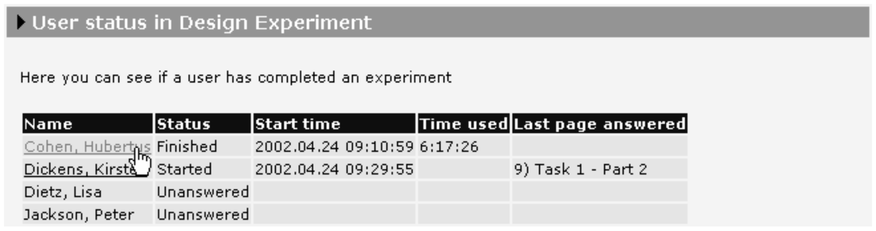
Figure 5. *Question 10*, *Task 1 – Part 3*

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 as a Microsoft Access 2000 database table and then be copied into a statistical analysis tool, for example.

4.2. Technical Architecture

SESE is deployed on an n-tier client/server architecture, built on Microsoft COM technology (Figure 7). The SESE application layer runs on one computer and the database on another. Users communicate with the application through a standard web-browser (e.g., Netscape and Internet Explorer).



User status in Design Experiment

Here you can see if a user has completed an experiment

Name	Status	Start time	Time used	Last page answered
Cohen, Hubertus	Finished	2002.04.24 09:10:59	6:17:26	
Dickens, Kirste	Started	2002.04.24 09:29:55		9) Task 1 - Part 2
Dietz, Lisa	Unanswered			
Jackson, Peter	Unanswered			

Figure 6. *Monitoring the experiment*

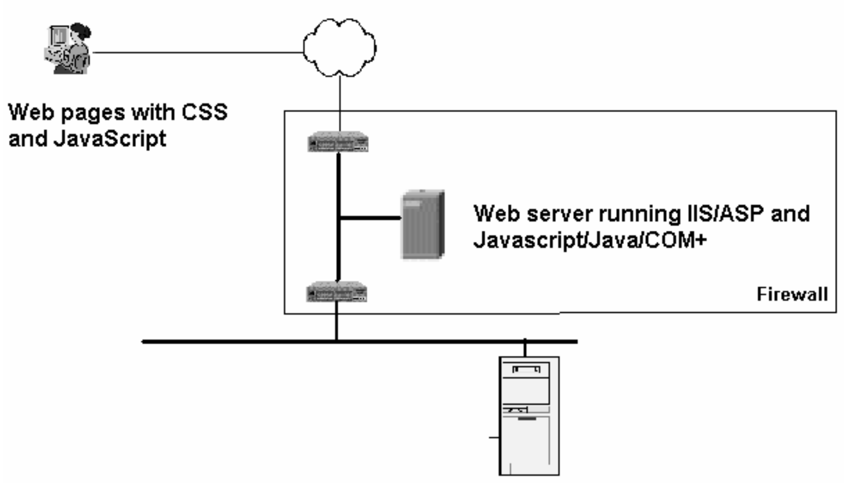

Web pages with CSS and JavaScript
Web server running IIS/ASP and Javascript/Java/COM+
Firewall

Figure 7. *The SESE client/server architecture*

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.

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.

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 class-room 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 (Accenture, Cap Gemini Ernst & Young, Ementa, Ementor, Genera, Objectnet, Software Innovation, Software Innovation Technology and TietoEnator). The remaining 60 subjects were students from the University of Oslo. All subjects were paid to participate. The students were given an honorarium of 1000 NOK each, whereas the consultancy companies were paid slightly less than normal consultancy fees (from 500 to 700 NOK per hour per developer depending on the seniority level of each developer).

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 & 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 Tasks

Important infrastructure needs to be in place to conduct administrative tasks:

- The researcher must be physically present during the whole experiment session, to assist in problems or answer questions and, in general, to monitor and control the experiment. While SESE’s monitoring functionality is an important and useful tool, it is insufficient to ensure that the experiment runs smoothly.
- 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. 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.

5.2.3. 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.

5.2.4. 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.

5.2.5. 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 and Future Work

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. For example, future extensions of SESE will include detailed logging of the way a task is performed or a technology is used.

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 and Kirsten Ribu 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. & Sjøberg, D.I.K., Assessing the Changeability of two Object-Oriented Design Alternatives – a Controlled Experiment with Professionals in Realistic Environments, 2002 (in preparation).

Arisholm, E., Sjøberg, D.I.K. & Jørgensen, M. Assessing the Changeability of two Object-Oriented Design Alternatives – a Controlled Experiment. Empirical Software Engineering, (6):231–277, Sep. 2001.

Basili, V.R. The Role of Experimentation in Software Engineering: Past, Current, and Future, Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, March 25-29, pp. 442–449, 1996.

Basili, Victor R., Rombach, Dieter & Selby, Richard. The Experimental Paradigm in Software Engineering. Experimental Engineering Issues: Critical Assessment and Future Directions, International Workshop, Dagstuhl, Germany, 1992, Springer Verlag, LNCS, No. 706, 1993.

Basili, Victor R., Selby, Richard & Hutchens, David. Experimentation in Software Engineering. IEEE Transactions on Software Engineering (invited paper), July 1986.

Briand, L.C., Bunse, C. & Daly, J.W., A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs, IEEE Transactions on Software Engineering, Vol. 27, No. 6, pp. 513–530, 2001.

Glass, R.L. The Software-Research Crisis, IEEE Software, vol. 11, no. 6, pp. 42–47, 1994.

Harrison, W. 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. & Jørgensen, M. Data Collection in Software Engineering Experiments, IRMA2001, Toronto, Canada, May 20–23, 2001, pp. 1027–1028.

Potts, C. Software-Engineering Research Revisited, IEEE Software, vol. 10, no. 5, pp. 19–28, 1993.

Rombach *et al.* Experimental Software Engineering Issues: Critical Assessment and Future Directions, Dagstuhl Workshop, Germany, September, 1992, LNCS 706, Springer Verlag, 1993.

Sjøberg, D.I.K., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Koren, E.F., and Vokac M. Conducting Realistic Experiments in Software Engineering. To appear at ISESE'2002, Nara, Japan, 3–4 October 2002.

Tichy, W.F., Should Computer Scientists Experiment More? 16 Reasons to Avoid Experimentation, IEEE Computer Vol. 31, No. 5, pp. 32–40, May 1998.

Zelkowitz, M.V. & Wallace, D.R., Experimental Models for Validating Technology", IEEE Computer, Vol. 31, No. 5; pp. 23–31, May 1998.