Galton4m — A Survey System

Seminar work

Jakub Holý

Departement of Computer Science

FEE CTU in Prague

March 2, 2005

Chapter 1

Introduction

The purpose of my work was to create a web-based application for the design and application of on-line questionnaries that should be used by sociologists and similar specialists with high requirements. In addition to it I considered it essential to apply the best practices of software engineering and modern methods of software development.

1.1 Draft of the structure

Methodology Nowadays, there're several categories of software engineering methodologies and many methods and their variations inside each category. Thus it's necessary to consider carefully what method is the best for a given project with respect to overhead it induces and benefits it offers. I'll present some of the

most widespread methodologies, discuss their advantages and disadvantages, and select the most suitable one for this web-based one-person project.

Analysis of the problem domain First I have to analyse the problem to learn what I'm requested to do, what else needs to be done and what are the possible problems I could encounter. The analysis has to deal with user requirements, constraintes given by the area of application and available software, the relation of the application to the current "business processes", use cases and more.

Essential design decisions, overview and analysis of existing solutions Based on the analysis I've to choose an implementation environment and make some essential design decisions regarding the architecture and frameworks to be used. First of all I need to select the implementation platform, in other words a programming language and perhaps some other platform-determining factors such as target web server and database engine unless I decide for a platform independent solution.

Second, I have to choose what architecture and frameworks will I use. This question is closely related to two of the keywords of modern software engineering — *reusability* and *component architecture*. They imply among other things that we should not "reinvent the weel" but rather we should profit from the work of others and use already existing pieces of code, libraries and stand-alone compo-

nents to speed up the application development. On the other hand sometimes it might be faster and better to implement a functionality by yourself than trying to understand and fit to your needs a — possibly buggy — code of somebody else. So I have to analyse similar existing (open-source) projects and related solutions such as database access and to study frameworks and architectures used in web development.

1. To find components or even an entire solution that could be reused in my project.

2. To learn different ways to do the same thing and select the best fitting one.

I'd like to stress that my primary goal is not to create my own, original solution but to deliever as functional and as usable product as possible as soon as possible. The art of software engineering is to make a product that meets all requirements including deadline and budget limits and satisfies fully both the customer and the users. The desires and criteria of developers are secondary to that. Why do I say that? Because we can see very well the opposite in the open-source community: there are many ambitious projects based perhaps on great ideas but never finished, left half-developed. I'd be actually content to find a project that is very close to what I need and that only requires some refinement and extensions.

Design Finally, a detailed design based on the software engineering methodology and previous decisions has to be done.

1.2 Why does software engineering matter?

On the importance of software engineering. Problems of software projects and benefits of SW engineering.

Chapter 2

Analysis of the problem domain

2.1 The task

Since it's expensive and time-consuming to produce, distribute, collect and evaluate paper forms, researches prefer on-line surveys in certain cases. And since often they haven't IT specialists at their disposal they need an easy to use and easy to deploy system to help them with it.

Therefore the task of this project is to produce a web-based system for creation and administration of on-line surveys. The system should allow users to assemble a form, publish it and obtain the collected data. It should allow for a certain level of customization of appearance. Prime users of the system will be scientists with high requirements and, probably, a low knowledge of the invovled web technologies.

The system will be used at the Faculty of Humanities of Charles University and has to run on the available software platform.

2.2 Introductory study

As the system aims to be used by social and other scientists it must support a wide range of question types appearing frequently in scientific surveys. Moreover, it could profit from the dynamic-content capabilities of the web and implement the behaviour that is often required but hard to do with a paper form such as conditioned questions or sections that only appear if a particular response is selected.

The system must make possible all common operations with surveys — creation, edition, publication, termination, removal.

Since the system may be used by more questionary authors and we need to protect a questionary from unauthorized modifications a certain form of authentication is required. In addition, the system could optionally support identification or authentication of respondents that could be combined with unique form URLs to control who has (not) filled it or with additional data to personalize the form.

The users are not expected to be very experienced with computers and to have a good support from their IT departement. This results in two requirements: first, the system must require minimal effort to install including its dependencies such as a particular database software and web server; second, the user interface must

be simple and intuitive while being powerful enough to create "any" questionary desired.

The system is expected to be used only by a low number of researches and by tens or at maximum hundreds of concurrent respondents and thus performance mesures are not very important.

The system will be deployed at the Faculty of Humanities and has to be compatible with the software currently used there which is either MS SQL & IIS or PostgreSQL & Apache.

A detailed list of requirements follows.

2.3 Requirements

This section presents requirements on the functionality and various characteristics of the product. They've been obtained from multiple sources — my own knowledge of the problem area, observations of business processes at work at present, user requirements, experiences from an on-line survey, and the available software platform.

First I shall present a list of requirements sorted by subject and than a list of requirements sorted by priorities. Only some of the requirements are strict and has to be met, most of them are a kind of suggestions or wishes.

2.3.1 Subject-grouped requirements

2.3.1.1 Questions — form elements

- Logical type
  - Closed — selection from a list of given answers
  - Half-open — same as closed except for one choice that let the user to fill in a text (such as "Other - write it down")
  - Open — the respondent can write down whatever s/he wants
- Presentation — visual form of a question
  - Selection from a list of answers; either single or multiple (perhaps with bottom or/and upper bounds on the number of choices)
  - Entry of a text
  - Other — select or modify a picture, semantic differential, ... — it’s not required that the system supports all but it should make it possible to add easily new types of questions
  - Multi-page and single-page forms
- Question flow
  - Conditioned questions or sections (branching) — they only appear if a particular answer is (not) selected

  - Randomized order of questions in a given section or randomized order of (selected) sections

- Validation — the data are validated before being commited and the respondent is prompted to correct any problems; it checks that obligatory questions are answered and the format of some input data (such as "an integer" or "an e-mail address")

- Other — Connection with a data source: it's possible to create answers to select from for a question based on data from a database, for example to offer a selection from courses the respondent has taken

2.3.1.2 Platform

The system shall be as much platform independent as possible and it must support the platform available at the Faculty of Humanities. The reason for the platform independence is given by the requirement for an easy deployment — the user shall not be forced to install any new software to be able to run the application because that could be unmanagable without an IT support. And since the application is targeted at people without sufficient IT resources with the principal aim to overcome the missing IT expertise it is essential that the deployment requires minimal effort. At least the platforms that are the most common in low-profit sectors such as universities and research institutes shall be supported.

2.3.1.3 User interface

The user interface should be simple and easy enough to use for the inexperienced.

2.3.1.4 Internationalization

It's imperative that the system handles the encoding of non-latin1 texts correctly, both for the form texts and user typed texts, above all regarding Czech. Moreover, it would be good if the system supported full internationalization of elements of forms such as button labels, in other words if the texts could be replaced by translated ones.

2.3.1.5 Customization

The system should allow for a certain level of customization of forms such as custom styles (CSS) assignable to a form or event to individual elements. And it would be perfect if the system supported the customization of layout of forms or questions, either via CSS or via templates.

2.3.1.6 Documentation

A sufficient user documentation should be available.

2.3.1.7 Security

The system shall satisfy basic requirements of security — at least it should be protected against SQL injection to protect data in the database from a disclosure.

2.3.1.8 Developer requirements

- Maintainability — clear code, modular, good documentation
- Extensibility — easy to add new features, question types etc.
- Granularity — the system should be ready for different levels of users ranging from inexperienced computer users to programmers offering them an appropriately increasing level of control

2.3.1.9 Other

It must be possible to download the collected data in a form that can be read by other programs, ideally CVS (semicolon separated data fields).

The resulting form as seen by respondents may not depend upon JavaScript being enabled. The reason is that there isn't any way to ensure that all or almost all respondents will have it switched on. JavaScript still can be used to improve the user experience and for additional, not necessary features — in general everywhere where it doesn't matter whether it works for all or only for some users.

2.3.2 Priority-ordered requirements

1. Support of all common types of questions
2. Ready to deploy at the Faculty of Humanities
3. Support of the Czech language
4. High platform independence.
5. Easy & intuitive user interface
6. Easy installation of the application and its dependencies.
7. Question flow — Conditioned questions
8. Connection to a data source
9. Customization of the appearance
10. Easy download and upload of data
11. Validation of form data
12. Security
13. Authentication, identification
14. Documentation

15. The rest — a progress bar, running statistics, text formating of questions' description, randomized order of questions etc.

Chapter 3

Essential design decisions, overview and analysis of existing solutions

3.1 Implementation environment

It has already been said that the solution is going to be a web-based one. The reason is simple — it creates forms that must be located on a server anyway and that need the same technology for data validation, database connectivity etc.

A server-side technology can be integrated with the (web) server either as its module communicating to it via its Application Programming Interface (API) or it can be a stand-alone command-line executable communicating to the server via a standard interface called “Common Gateway Interface” (CGI) — they communicate by means of environmental variables and standard input/output. According

to [10] “The main drawback of the CGI technique is overall inefficiency in handling concurrent client requests”. The author adds that “The use of modules improves performance by adding multithreading and persistent database connection features.”

Regarding the API way, there are three widespread server-side technologies for web projects — Active Server Pages (ASP), J2EE including JavaServer Pages (JSP), and PHP. All are based on the principle of embedding a special code in an ordinary web page that is processed by an appropriate interpreter to produce pure HTML before being sent to a web browser. Regarding the CGI way, the most used technology is Perl. I’ll briefly introduce each of them together with its advantages and disadvantages. Then a discussion will follow that will decide which one fits best to this project.

3.1.1 Active Server Pages (ASP)

Active Server Pages (ASP) is a technology created originally by Microsoft for its *Internet Information Services* or *Server* (IIS) that runs under Windows. The classic ASP (up to and including the version 3.0) is similar to PHP except for that it doesn’t define a new language. Instead, Microsoft’s VBScript or its implementation of JavaScript called JScript can be used as the scripting language and third-party plugins for other languages are available as well. The latest ASP.NET

is a part of the new .NET platform and resembles much more the Java approach (compiled code, virtual machine, lot of pre-made components, object-oriented). ASP.NET can be scripted in any .NET language (e.g. C#) and requires the .NET framework to be installed.

Microsoft delievers also an integrated development environment (IDE) for ASP.NET that is either free of charge or reasonably cheap and that is a great help in the development and debugging. Of course it is only available under Windows.

The main disadvantage of ASP and ASP.NET is that they're system dependant though there're attemps to port them to other operating systems.

When compared to PHP, "ASP.NET is expensive with respect to memory usage and execution time, which is due in large part to a longer code path.", at least according to [6]. And that's undoubtably important for a web server. It'd be interesting to know how does it perform in comparison to J2EE. There is a couple of benchmarks comparing the two but, as with all benchmarking, there's also many questions and criticism. Thus it's hard to conclude anything reliable.

3.1.2 J2EE and JavaServer Pages (JSP)

SUN Microsystems created a number of technologies to ease web development that are based on its programming language Java. The whole of them is usually reffered to as J2EE. Java is a mature, powerful object-oriented language with a lot

of libraries covering all possible areas including database connectivity, mailing and security, all of them well documented.

Java 2 Enterprise Edition (J2EE) "is a standard (albeit with no ISO or ECMA standard) for developing distributed Multi-tier architecture applications, based on modular components running on an application server" [9]. Some of the key components are XML, Enterprise Java Beans (EJB), Java Servlets and Java Server Pages. The architecture usually consists of a presentation tier employing JSP and servlets, a business tier containing all the logic and based on the Enterprise Java Beans, and a physical tier — a database. To run a J2EE application you need an EJB container and a JSP container (also known as a servlet container) that are usually a part of an application server such as jBoss. The objective of the EJB container is to simplify the job of a programmer and to perform a lot of common tasks on behalf of her. For example, it can handle the persistance of an object without the need to write a single line of code. Thus the programmer can concentrate on mission-specific tasks and the development is much faster. Of course conditions are not always favorable and it may happen that services provided by the container are not sufficient and something must be implemented. The disadvantage of an EJB container is that it consumes additional resources and that it may be difficult to configure and to learn how to use it.

There is a number of third-party frameworks for web applications that further accelerate the development. The most important ones are Hibernate, which pro-

vides an object-relational mapping and a database abstraction layer, Struts that simplifies many tasks of the presentation tier such as data validation and unification of page appearance and enforces the Model-View-Controller architecture, and Spring, a powerful application framework based on the inversion-of-control (IoC), supporting aspect-oriented programming (AOP) and offering, among others, a generic abstraction layer for transaction management.

JavaServer Pages is a technology closely related to servlets. A servlet is a Java class residing on a server and processing a client request to return a response to it. For the most part it processes HTTP requests from a web browser and responds by a HTML page in return — such a servlet normally extends the class javax.servlet.http.HttpServlet. For instance, the request for "http://example.com/news" could be forwarded to a servlet that would read the latest news from a database, create a page with their headlines and send it to the client. JavaServer Pages were introduced to ease the development of dynamic web pages that do not require much code and to make the strength of Servlets available to those who aren't Java programmers. JSP is similar to PHP — it's Java code embedded in a HTML page. But, unlike PHP, a JSP page is translated into a Servlet; it's only another way to write a Servlet. And the Servlet, as well as any other, is then compiled into binary bytecode (.class file) — both these transformation happen when the page is accessed for the first time. The advantage is that any further accesses to the page are much faster as it doesn't need to be interpreted again, as opposed to PHP.

And they can be compiled in advance to avoid the initial slow processing. Another advantage is that it gives you accesse to the whole power of Java. To use JSP and Servlets you need a JSP container, such as Tomcat — it performs all the processing described above, and Java Virtual Machine.

The J2EE techologies may be used in different combinations with respect to requirements and (software, time, ...) resources. The most common is likely a full-blown J2EE application based on EJB. An equally powerful alternative is an application based on Spring with Hibernate that, moreover, doesn't require an EJB container. And finally, the simplest but least powerful way is to use only JSP and Servlets.

3.1.3 PHP

PHP (which is a recursive acronym for "PHP: Hypertext Preprocessor", according to [9]) is an open-source scripting language. It is not object-oriented but provides a certain level of support for object-oriented programming (OOP). A code in PHP can be embedded into an HTML page, which is then run through a PHP preprocessor to produce pure HTML. PHP modules exist for most major web servers and it can be used through the common gateway interface (CGI) too. (CGI enables a web server to run any command-line executable and to display its output.) It's best for small- and middle-size projects. Contrary to a JSP container, the PHP pre-

processor is stateless — it doesn't preserve any information between successive calls. The first consequence of this is a lesser effectivity — a page is parsed and interpreted everytime it's requested even if it is the same page as the last time. On the other hand, commercial and free tools improving the performance by means of caching, compilation etc. are available if the performance is important. The second consequence is that it's much harder for a programmer to preserve data between requests as he must do all the work by himself with a little help of the global variable $_SESSION.

There exist a lot of libraries and components for PHP but they are all produced by third-parties and thus they're much less uniform than in the case of JSP. Moreover, as they're mostly produced by the open-source community, they may exhibit common problems of such products — missing or incomplete documentation, unsure quality, and so on. On the other hand, there is an initiative to build a set of high-quality components called PEAR — "the PHP Extension and Application Repository". Additional extensions enhancing the capabilities of PHP can be obtained from PECL — "the PHP Extension Community Library ". An extensions is written in C and needs to be compiled in the PHP executable.

PHP is likely to be the most widely used of the discussed technologies. According to [9], "One major part of PHP which has helped it become popular is that it is a very loose language; in particular, it is dynamically typed." It is used for example by Yahoo!.

The latest version of PHP is PHP 5 that adds more support for the object-oriented programming. But since it's quite recent, PHP 4 is still used much more often.

3.1.4 CGI/Perl

Perl is an interpreted language for general text processing with a very rich set of modules and libraries in its CPAN repository. It is, as well as PHP, a loosely typed language. Perhaps it's usually used via CGI but modules exist for some servers too. According to [8] Perl has the following disadvantages:

- There's More Than One Way To Do It
- poor sandboxing, easy to screw up server
- wasn't designed as web scripting

3.1.5 Making the choice

For what platform should I decide depends on the criteria of their evaluation. From the analysis in the chapter §2 follow these criteria, ordered according to the priority:

1. Deployment costs — the time and effort needed to install the final product and all its dependencies plus any financial costs. If different target platforms

(O.S., server, database) are taken into account, the platform independence counts here too because it is difficult or even impossible to install an application not targeted for the platform at hand.

2. Development costs — the time it takes to create the application together with the time needed to learn the technology if it's new for the developer.

Performance is not very important as it has been explained earlier. Security is secondary too though it may not be neglected.

We can divide the technologies into two groups: "lightweight" and "complex" ones. The first including the classic ASP, PHP and CGI or module Perl, the second including J2EE and ASP.NET. The lightweight technologies are less powerful but perhaps easier to learn and use, have not such a clear and nice design and are not trully object-oriented. The complex technologies are harder to set up, have more dependencies, require more configuration and knowledge, but provide a lot of ways to speed the development, encourage component-based development and clear, reusable code.

The lightweight technologies ASP, contrary to the other two, is highly platform dependant as it's only destined for Windows and IIS and that's the reason to reject it. PHP and Perl are quite similar and the choice is a matter of taste. I prefer PHP because it is designed for web development and it seems to be superior to CGI/Perl

as explained above. Furthermore, a PHP application is ready to be deployed at many sites thanks to wide spread of PHP in the low-profit sector of my interest.

The complex technologies ASP.NET shall be rejected for the same reason as ASP. J2EE is very good but it rather addresses enterprise applications and is perhaps an "overkill" for my task. More importantly, it's time-consuming to set up the appropriate environment (JVM, application server) and not very likely that it's already installed.

Finally I've decided to use PHP because it is the most widespread technology, well suited for small- and middle-size applications, with a lot of documentation and helpful user community and, last but not least, there's already a couple of applications similar to the mine that can be used as a base for my development. This choice doesn't pose any or nearly any limitations to the server and database to use.

3.2 Overview of existing solutions

3.2.1 Criteria of evaluation of the projects

The criteria for evaluation of the existing projects are derived directly from the priority-ordered requirements that I shall not repeat here. An additional criterium I find important is whether the project is actively maintained and further developed.

And that because it promises that new requirements that may emerge in the future could be satisfied and because it's easier to reuse or extend a project if you can communicate with its developer(s).

3.2.2 The Unit Command Climate Assessment and Survey System (UCCASS)

`http://UCCASS.sf.net/`. Demo avalable.

UCCASS is one of the best survey systems available, originally designed for the government. It is flexible and portable thanks to applying the famous template engine Smarty for the presentation layer and ADOdb for the database abstraction layer. And it's easy to install, at least if the target database is MySQL.

It supports all the common types of questions and a "matrix question" — rows of radio buttons, typically used for semantic differentials. A half-opened question cannot be created directly but still it's possible to combine a selection question and a conditioned text input but it's not a perfect solution. A question description can be either plain text, limited HTML or full HTML including JavaScript, depending on the survey settings. The same holds for respondent-supplied input but full HTML is discouraged there for security reasons. It's also possible to insert a page break at any point splitting the survey into multiple pages. Arbitrary number of dependencies can be set for any question to make it to (dis)appear or to become

required or optional if a particular response has been selected for a previous question this one depends upon. A very nice feature is the ability to define and reuse "answer types" (such as yes/no/maybe or a scale from 1 to 5) as it's often the case that a set of questions share the same type of answer.

The user interface is pleasant and easy to use. A sufficient user documentation in English is available.

UCCASS has some nice additional features — a graphical result view, start and end dates for a survey, various layouts of check boxes/radio butons (horizontal, vertical, matrix)

Nonetheless, there are some weak points too. First of all, the system as is — or at least the demonstration version — doesn't handle some accented Czech characters correctly. Perhaps that could be corrected by providing a template that sets the encoding of all pages to either either ISO-8859-2 (latin2) or UTF-8, but it's unsure. Next, it doesn't provide any form of validation, neither server- nor client-side. Currently it also doesn't support user identification and/or authentication except for "private surveys" protected by a single password but it's planned for the next release. Finally, it is not possible to use external data to construct a set of answers or for other purposes. And it's a question how easy or difficult it would be to localize the application — the template system could make it easy but there may be some areas it doesn't cover.

The conclusion to draw is that UCCASS is pretty close to what I need but it

requires some work to solve the problems stated above.

3.2.3 Marketing Survey Tool

`http://surveyworld.org/`. Demo avalable.

MST is a one-man work that si intended for research specialists. It works with two most widespread open-source databases, namely with MySQL and PostgreSQL. It has pretty good support for various question types and excellent question flow management allowing for conditioned author-defined blocks instead of single questions and for randomization of question order inside a block. It's even possible to rotate the order of answers to a question.

Other positive characteristics are that MST handles Czech texts in question description correctly and that it has a partial support for authentication, there're even two types of it (internal, external).

A weak point of MST is the user interface for survey authors — it's split into many pages. This problem shall be addressed by the next release if there will be any (the last is from april 2004). The only possible customization is by assigning a new style (perhaps a CSS stylesheet) to a survey. It's possible to set any block (or question?) as required but no other input data validation can be performed.

Unfortunately I couldn't have finished exploring the application because the demo version that is on-line stopped working.

3.2.4 phpESP - php Easy Survey Package

`http://phpesp.sf.net/`. Demo avalable.

phpESP is weak in many regards, for instance it doesn't enable any change of the question flow, it only works with MySQL (though it's being ported to ADOdb) and a survey cannot be modified after it the transition to the testing state. But it excells in two regards too.

First, it has a very well worked-out authentication based on groups and roles — there's a superuser role assigned to all members of the superuser group, a group manager role that has superuser rights limited to a given group, a designer role (for survey authors) and a respondent role. Respondents are grouped to allow them the right to fill a form and designers are grouped to be managed by a group manager.

Second, it doesn't only handle Czech characters without problems but it also supports full internationalization of all strings by means of the library gettext.

Some other positive features are the ability to send filled forms by e-mail, HTML tags in question description and postponed surveys — a respondent can stop after filling a half of a survey and come back later to continue.

3.2.5 PHPSurveyor

`http://phpsurveyor.sourceforge.net/`. Demo avalable.

PHP Surveyor is an actively developed product of 12 people. It is a very

good application though it has some weak points too; it has most of the required features. For example, it's quite good in internacionalization and customization — it's possible to use Czech without any modifications and its appearance can be changed by means of templates. An outstanding feature is that it supports "tokens" to make it possible to allow and track access to/of selected individuals and send them invitation and reminder emails. Its qualities are proved by the fact that it has once been used in the US elections for some task. Some of other nice features are the ability to stop filling a survey and resume it later on, the "expires" date of a survey, the possibility to copy (reuse) branching conditions, an optional "previous" button, a progress bar and the fact that input validation (whether required answers have been given) is made both client-side (faster) and server-side (doesn't depend upon JavaScript).

PHPSurveyor supports all basic types of questions and a number of predefined ones (gender, some scales,...), but it isn't possible to define new answer types. Questions are grouped into groups to allow paging (single page, one group per page or a question per page) and to assign them a common text. A unique name not to be displayed is required for every question and group for they're sorted on base of them. The user interface is nice on one hand, with icons etc., on the other hand there're some things that could be improved. First, it's less flexible as it doesn't allow for questions to be moved between groups and thus it requires the survey to be designed in advance. Second, it only display's one element at a time

while creating a survey as opposed to those that display the whole form created so far — I find this less comfortable. Third, it is not as user-friendly as it should be — for instance, if a mistake is made when a new survey is being defined, all the data filled is lost and you have to start from scratch again.

An important disadvantage of PHP Surveyor is that it only works with MySQL and probably only with Apache (at least the authors do not mention any other server). And the installation may be a bit difficult because it requires manual editing of some configuration files.

PHP Surveyor would be a good choice if it was multi-platform and preferably had a better user interface.

3.2.6 Conclusion

There're other survey systems but those described are the best I've found and that are worth of being considered.

The application that best met the requirements is UCCASS. But it doesn't satisfy all of them — above all it doesn't handle Czech correctly and it has no support for internacionalization. And it can be improved in several ways inspired by those other applications described here.

Bibliography

[1] AuthorName, FirstName Initial., “Article Title,” *Journal Name, Book Title, or Proceedings Name,* **volume**(number), (City, State, Country: Publisher, day mon. year): page–page.

[2] Armstrong, E., et all., *The J2EE 1.4 Tutorial,* 2004, on-line: `http://java.sun.com/j2ee/1.4/docs/tutorial-update2/doc/index.html`; accessed February 17, 2005.

[3] Březina, V. *WWW prezentace počítačové firmy s formulářem na objednávku zboží,* Nepublikovaná diplomová práce, Jihočeská univerzita (Pedagogická fakulta), Praha 1998; on-line: `http://home.pf.jcu.cz/~pepe/Diplomky/brezina.doc`; accessed February 18, 2005.

[4] Burd, B., *JSP: JavaServer Pages : Podrobný průvodce,* Computer Press, Praha 2003.

[5] Heng, H. “PHP vs CGI,” on-line: `http://www.javascriptkit.com/howto/phpcgi.shtml`; accessed February 18, 2005.

[6] Hull, S. “PHP and ASP.NET Go Head-to-Head,” on-line: `http://www.oracle.com/technology/pub/articles/hull\_asp.html`; accessed February 18, 2005.

[7] Literák, L. “Rozhovor: Proč je NAVRCHOLU.cz v Javě?,” *ABC Linuxu*, on-line: `http://www.abclinuxu.cz/clanky/show/69723`; accessed February 17, 2005.

[8] Radwin, M. J. “Making the Case for PHP at Yahoo!,” Unpublished talk presented PHPCon 2002, on-line: `http://public.yahoo.com/~radwin/talks/yahoo-phpcon2002.pdf`; accessed February 17, 2005.

[9] *Wikipedia — The Free Encyclopedia*, on-line: `http://wikipedia.org/`; accessed February 17, 2005.

[10] Wu, A. W. “Performance Comparison Of Alternative Solutions For Web-To-Database Applications,” Unpublished paper, on-line: `http://rain.vislab.olemiss.edu/~ww1/homepage/project/mypaper.htm`; accessed February 17, 2005. Written 2000.