Developing dynamic Web applications: "Exam scheduler"

S. Maržić, I. Jugo, M. Radovan
University of Rijeka, Department of Informatics,
Omladinska 14, 51000 Rijeka, Croatia
sanjin.marzic@gmail.com; ijugo@inf.uniri.hr; mradovan@inf.uniri.hr

The paper presents and analyzes the methodology of the development of dynamic web applications, illustrated by the example of the web application for exam scheduling and managing, which has been developed for the use at the University of Rijeka, Department of Informatics. The main drawback of the previous application regarded the need for "manual" search for free exam terms, because it did not contain the possibility of automatic insight into the taken (already inserted) exam terms. The new application for the thorough evidence of exam terms has been created by using the script languages PHP and JavaScript, together with the relational database system MySQL and the language HTML, as the basic means for defining the structure of web pages. In the first part of the paper we describe the methodology of the development of web applications, while in the rest of the paper we illustrate the use of this methodology on the example of the development of the dynamic web application "Exam scheduler".

I. INTRODUCTION

Dynamic web applications, like classic desktop applications, ought to be developed by following a series of formally defined steps and phases determined by a methodology. There are several methodologies for software development but they all have a common basic structure [1]:

1. Requirements analysis and specification – what is the object/purpose of the application;
2. Design – how will the application fulfill its purpose;
3. Construction/implementation – implementing the designs created in the previous phase;
4. Verification and validation – does the application fulfill its purpose and work correctly;
5. Deployment – installation, user education, test phase;
6. Maintenance – support, change-test-redeployment loop.

Since this web application is not so complex as various large applications can be, some of the sub phases of the methodology have not been necessary. This regards mainly the strategic planning phase, which deals mostly with general preparation activities and problem analysis. The development of this application actually started as a new iteration from the Maintenance phase.

We developed the "Exam Scheduler" application, which was then added to our existing web content management system. A similar application that was used as an exam scheduler existed before this one but it had several flaws and only basic functions. Recent changes introduced by the Bologna education system required that we rebuild the existing application. The main drawbacks of the existing application where the lack of different exam schedule views as well as the absence of exam overlap control (which had to be done manually). The removal of these drawbacks has been the main goal of this process.

All phases of the methodology must be repeated when making a change in the existing system, especially in the case of developing a new system which completely replace the existing one.

II. MODELING AND DEVELOPING THE WEB APPLICATION "EXAM SCHEDULER"

A. Requirements

The object of this process is to build a web application that will enable the staff of the Department of informatics to manage their exam schedules online. This web application will be added to the existing content management system of the Department.

The goal of the main project phase is a detailed analysis of the system and its decomposition into modules which will be realized as separate project designs. The main product of this phase is a process model of the selected system.

The main requirements are:

- web application will enable the users to manage their exams (insert, update, delete)
- web application will detect and prevent all possible exam terms overlaps by notifying the user and offering possible correct inputs
- web application will have multiple views of the data: full academic year exam schedule,

monthly view, weekly view, per employee view, per course view

- web application will enable data manipulation (exam submission, update, delete) for Department employees only (identity will be checked via a SOAP request [2] to the AAI@EduHr[3] infrastructure)
- exam scheduler views will have an option to be made public or visible only to Department students (identity will also be checked using AAI@EduHr infrastructure)

B. Design

In the next phase we created the process model for our application. The process model describes the process of the observed system on several levels (each one going one more level of decomposition deeper, until we reach the most basic processes that cannot be decomposed further). The process model is created based on interviewing the experts who know and work on the real system. When working on a large scale information system many interviews have to be conducted in order to create a correct and precise process model. The process models for our web application can be seen in Figure I and II.

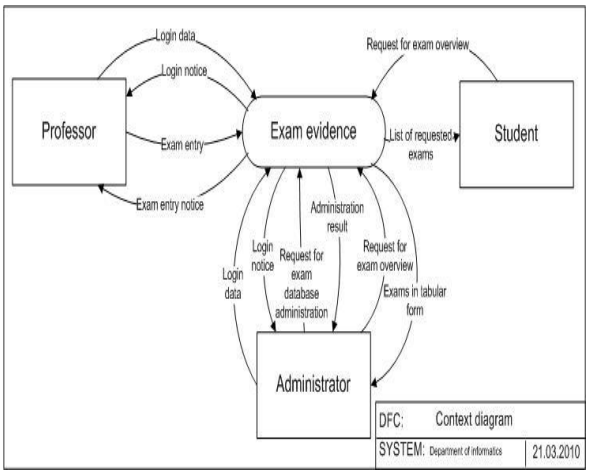

FIGURE I. CONTEXT DIAGRAM FOR THE EXAM SCHEDULER

The context diagram (Figure I) models the whole observed system as one process and its data relations with the outside systems. The first level diagram (Figure II) models its sub processes that cannot be decomposed further.

Processes are presented as rounded shapes, while outside systems are presented as rectangles. Data relations (flows) are displayed as arrows between two processes or the outside system and the process. Outside systems in this model are professors, students and management.

The next step is to develop a data model. The model is based on the previous data model and new data requirements that follow from the new set of requirements. The data model (or entity-relationship model) depicts all entities, aggregations and their relationships as shown in Figure III.

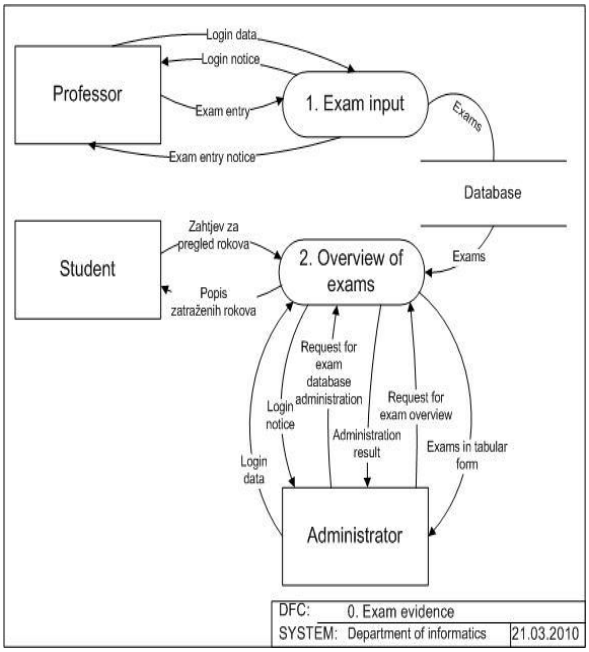

FIGURE II. PROCESS MODEL FOR EXAM SCHEDULER

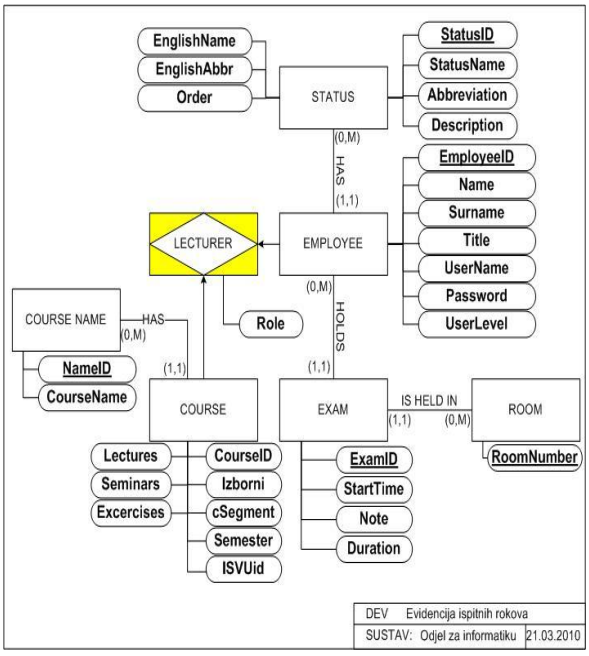

FIGURE III. ER DIAGRAM FOR EXAM SCHEDULER

The entities on the data model must all have a unique identifier (primary key) which is displayed in strong underlined style. An exception from this rule is the aggregation concept whose primary key is a set

consisting of primary keys of its connected entities. Relationships between entities must be named (except relationships between aggregations) and have stated cardinalities (0,1,M) which are later used for translation to the relational model.

When the data model has been completed and reviewed it is ready to be translated into the database relational model. The translation is performed following a set of rules which define the translation procedures for each concept shown in the data model. The most important rules used in this example are:

Each entity from the Entity-Relationship (ER) model becomes a relation in the relational model in such way that the entity attributes become relation attributes, the primary key of the entity becomes the primary key of the relation.

Relationships with cardinalities (1,1):(0,M) are not translated into a relationships, but the key of the entity that enters the relationship with the cardinality M becomes an attribute (foreign key) of the relation that enters the relationship with the cardinality 1.

Any relationship with the cardinality (0,M):(0,M) is called an aggregation and becomes a relation in the relational model, with a primary key made of the primary keys of the connected entities. This relationship type can have more attributes and they become attributes of the relation.

The resulting relational model is displayed in Table I.

TABLE I. RELATION MODEL

Status(**IdStatus**, NazivStatusa, Kratica, Opis, EngleskiNaziv, EngleskaKratica, Poredak)
Zaposlenik(**IdZaposlenika**, Ime, Prezime, Titula, KorisnickoIme, Zaporka, RazinaPrava, fk_*IdStatus*)
Nositelj(***IdZaposlenik***, ***IdKolegij***, Uloga)
NazivKolegija(**IdNaziva**, NazivKolegija)
Kolegij(**IdKolegija**, Izborni, cSegment, Semestar, ISVUId, Predavanja, Seminari, Vježbe, fk_*IdNaziva*)
Ispit(IdIspita, Vrijeme, Napomena, Trajanje, *IdZaposlenik*, *BrojProstorije*)
Prostorija(**BrojProstorije,** Number of seats, Projector, Comments)

The following step in the design phase is to make drafts of the application user interfaces (also known as "UX model"). As this application will be added to the existing Department website (CMS) the drafts had all the basic design guidelines (such as background color, text and link colors etc.). All basic forms for data manipulation and all data views were wireframed and then refined to detailed images. An example for a weekly data view can be seen in Figure IV.

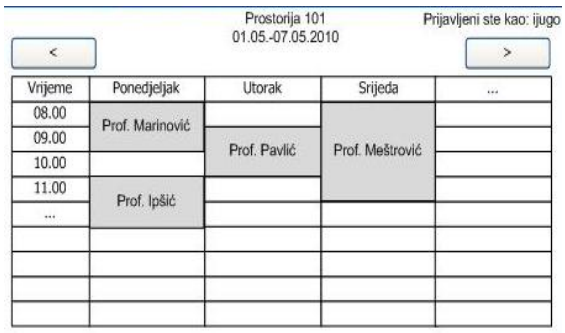

FIGURE 1. A WIREFRAME OF THE WEEKLY EXAM OVERVIEW

The final part of the design phase is to define all operations over the database schema. This includes writing all queries that will later be used in the application. Our web application uses MySQL [4], a relation database management system. There are two basic parts of the SQL language [5]: (1) the Data Definition Language, which is used to define relations, keys, attributes and data types, and (2) the Data Manipulation Language which is used to define queries/operation over the database relations such as inserting, updating, deleting data, and most importantly querying data from the database. The most important queries are described below:

- course exam schedule overview

```
SELECT naziviKolegija.naziv,
      zaposlenik.prezime,
      zaposlenik.ime,
      nositelj.uloga,
      ispit.prostorija,
      ispit.vrijeme
FROM zaposlenik, kolegij,
naziviKolegija, ispit, nositelj
WHERE ispit.zaposlenik=zaposlenik.id
AND
zaposlenik.id=nositelj.zaposlenik
AND nositelj.kolegij=kolegij.id
AND kolegij.naziv=naziviKolegija.id
[AND kolegij.id=$kolegij]
[AND
naziviKolegija.naziv=$nazivKolegija]
ORDER BY ispit.vrijeme
```

This query collects data from five relations in order to display all the data required for this view (exam schedule for the selected course and the name of the professor).

- employee exams overview

```
SELECT * FROM ispiti
WHERE zaposlenik=$sifra ORDER BY
vrijeme DESC
```

- exams in a selected week

```
SELECT * FROM ispit
```

```
WHERE vrijeme BETWEEN $pocetak AND
$kraj ORDER BY vrijeme
```

C. *Construction*

The construction phase is the most time consuming phase during which the code of the application is being written by programmers.

The first activity in this phase was to build the database on the MySQL server. To speed up the process we used the common PhpMyAdmin [6] web application that serves as a user interface to the MySQL server.

The second activity was to create all the basic HTML [7] files needed for forms and data views. After that they were styled using CSS [8]. Furthermore the form fields were given unique id's to facilitate JavaScript [9] manipulation. Lastly, all the necessary JavaScript functions (for input validation and visual presentation of the data (class schedule table)) as well as XmlHttpRequest[10] object invocations were written and tested.

The final activity was to write the necessary PHP code [11]. In order to simplify the application structure and help performance, we decided to write all the code in one PHP file; we obtained code separation by using procedural programming. The main structure of the application is based on a switch structure that receives the action instruction requests from the user (via query string) and calls the required functions. The query string is the main channel for transferring user requests for action as well as starting or ending data identifiers among subsequent HTTP requests.

An example of the query string usage can be seen below:

http://www.adresa.hr/index.php?varijabla1=vrijednost1&varijabla2=vrijednost2

A query string starts with the question mark (?) which is followed by a number of "key=value" pairs separated by the ampersand character (&).

The main functions followed from the process model: exam_overlap_check() and display_schedule(), with a number of auxiliary functions (such as basic input control, database connection, page header/footer, etc.). On a more general basis we can divide the application functionality into three parts: (a) inserting, updating and deleting exam terms, (b) data views and (c) administrative functions. We will describe their development process in the following paragraphs.

To submit exams and update or delete them the user has to authenticate his identity via a login screen that sends the user's credentials to an AAI server through a SOAP request and returns the identification result. When submitting a new, or updating previously submitted exams, a function is called that checks for overlaps. This function gets the starting date and time of the exam, duration and classroom number and then checks for all possible overlap situations as depicted in Figure V.

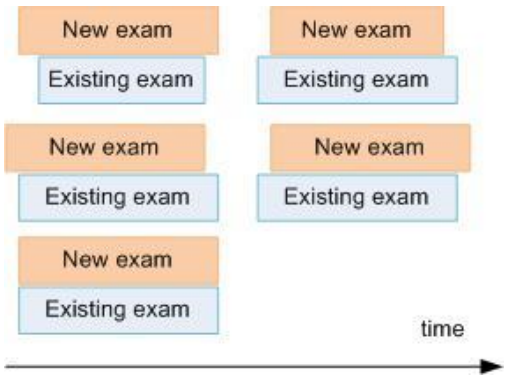

FIGURE V. PROCESS MODEL FOR EXAM SCHEDULER

The SQL query is written as follows (parameters sent to the function begin with the $ sign as mandatory for the PHP language):

```
SELECT * FROM ispit, zaposlenik
WHERE ispit.zaposlenik=zaposlenik.id
AND prostorija=$prostorija
AND ((ispit.vrijeme<=$pocetak AND
(ispit.vrijeme+ispit.trajanje)>$poce
tak)
OR (vrijeme<$kraj AND
(vrijeme+trajanje)>=$kraj)
OR (vrijeme>=$pocetak AND
vrijeme<$kraj)
OR ((vrijeme+trajanje)>$pocetak AND
(vrijeme+trajanje)<=$kraj))
ORDER BY vrijeme
```

The exam submission form requires from the user to insert starting date and time of the exam, duration, classroom number and comments. The starting date field has a JavaScript calendar function attached, which facilitates the insertion of the date. The time is inserted using drop down lists of hours and minutes. The classroom number is selected from another dropdown list populated by the data stored in the Prostorija relation. Another JavaScript function is used to validate all of the inserted or selected data, prior to form submission. If all the data are formally correct, an XmlHttpRequest (short: XHR) object is instantiated and through it a request is made to the PHP script, which then checks for overlaps in the database and returns a true/false result. If the result is *false* the form is submitted and the data is stored in the database.

The XHR object is one of the keystones of the AJAX [12] technology and the connected Web 2.0 web application usage paradigm change. It enables asynchronous communication between the client (web browser) and the server, and is integrated with the JavaScript interpreters of the most contemporary web browsers. When an XHR request is made the *onreadystatechange* event listener is automatically invoked for each change in the *readyState* property of the XHR object. When the response has finished

loading the *readyState* changes to the value 4 and then the response can be displayed on the page the user is currently viewing using JavaScript (for example by displaying the content of the *responseText* property as the contents of a paragraph).

In our web application this is implemented in the following way: the value of the variable *a* located in the query string determines the required action – *p* for exam submission, and *pa* for exam update. Other variables in the query string contain the rest of the required data. Depending on the value of *a* an XHR request is made to the PHP script which returns a true/false response. Depending on the response the form Submit button can be disabled and an alert window displayed.

While the user is logged in any of the defined data views, the view will show the update and delete icons next to his exams. To delete an exam the user has to click the delete icon/link and confirm his action before the exam is deleted.

All data views have a similar basic functionality – submit a query to the database, retrieve and display data on the screen. What changes is the display of data on the screen for each data view.

Monthly overview – the exam schedule for all exams for the current month is displayed in the classic table form. The PHP functions *date()* and *mktime()* are used to format the date output and to calculate the ending date for the data view, respectively. The following query is sent to the database ($pocetak and $kraj are timestamps marking the starting and ending dates for the selected month):

```
SELECT ispit.id AS id, zaposlenik.id
AS zaposlenik, ime, prezime,
vrijeme, napomena, trajanje,
prostorija
FROM ispit, zaposlenik
WHERE ispit.zaposlenik=zaposlenik.id
AND vrijeme BETWEEN $pocetak AND
$kraj
ORDER BY vrijeme ASC
```

The results of the query are all exams that fall into the defined time span ordered by starting time. The output is in the simple form of a HTML table.

Academic year overview – the data for this view is obtained using the same query as the previous view but the display is different. The HTML table columns represent months while each row displays exams for one employee.

Weekly overview – the data for this view is obtained using the same query as the previous view. This is the most complex view of the application. This view has the classic time organizer display with columns as days and rows as 30 minutes time blocks. The exams are displayed as blocks that span multiple rows filled with exam information. The problem is in the fact that HTML tables are written in start row-add columns manner, and this view requires that the table gets generated in the opposite way. This requires extra checks in order to get the right HTML syntax.

FIGURE VI. WEEKLY OVERVIEW OF THE FINISHED WEB APPLICATION

The administrator panel enables the users that have administrator privileges to get various data statistics and delete or change multiple exams at once (such as delete all exams from the finished academic year). Database administrators can also check the status of data tables of the MySQL server, and make repairs and backups if necessary.

D. Verification

After the application was finished, its functioning and performances were tested and validated in the real time usage. Some weaknesses which have been seen in this process, have been removed.

E. Deployment

After the application was tested using test data, it was uploaded to the Department web server and installed as a component of our CMS. The existing database schema was also updated with new and/or changed tables. The application and the procedures of its use were presented to the employees of the Department. The design of forms and views was tested and made self-explanatory as much as possible. In addition, all forms and data views have "help buttons" which facilitated the use of the application, especially during the initial period.

F. Maintenance

The last (and lasting) phase in the life-cycle of an application is the continuous process of monitoring the performances of the application, with the identification of its malfunctioning or weaknesses. Finally, if/when new administrative procedures pose new requirements to the application, the process of its maintenance ought to start with a new iteration of logical and physical modeling, development and implementation.

III. Conclusion

In this paper we presented a complete process of developing a web application, based on a general methodology of software development, which consists of several phases, from the requirements analysis to the application validation and maintenance. It is important to emphasize the specific differences of web applications, such as the simpler ways of update rollouts (the updated files replace the older ones on only one computer – the web server). This application demonstrates the synergetic effects of the "classic" web development technologies (such as HTML, PHP and MySQL) and new web technologies (such as AJAX). JavaScript was used to control user inputs, while XmlHttpRequest object was used to change the usage paradigm of the web application in order to increase application interactivity. The methodology we used ensured the success of the process through the fulfillment of all requirements and the elimination of possible functional errors. Other components of the Departments management system will be rewritten in the same way.

References

[1] SWEBOK - http://www.computer.org/portal/web/swebok/htmlformat (10.01.2011)

[2] SOAP - http://www.w3.org/TR/soap/ (10.01.2011)

[3] AAI - http://www.aaiedu.hr/ (10.01.2011)

[4] MySQL - http://www.mysql.com/ (10.01.2011)

[5] SQL - http://www.w3schools.com/sql (10.01.2011)

[6] PhpMyAdmin - http://www.phpmyadmin.net/ (10.01.2011)

[7] HTML - http://www.w3.org/MarkUp/ (10.01.2011)

[8] CSS - http://www.w3.org/Style/CSS/ (10.01.2011)

[9] Javascript - http://www.javascript.com/ (10.01.2011)

[10] XmlHttpRequest - http://www.w3.org/TR/XMLHttpRequest/ (10.01.2011)

[11] PHP - http://www.php.net/ (10.01.2011)

[12] Garrett, J.J: "Ajax: A New Approach to Web Applications". AdaptivePath.com (10.01.2011)