Architecture of an electronic student services system and its implementation

Ivan Chorbev[1], Marjan Gusev[1], Dejan Gjorgjevikj[1], Ana Madevska-Bogdanova[1]

[1] Faculty of computer science and engineering, University of Ss Cyril and Methodius, "Rugjer Boshkovikj" 16, P.O. Box 393, 1000 Skopje,R. of Macedonia
{ivan.chorbev, marjan.gushev, dejan.gjorgjevikj, ana.madevska.bogdanova}@finki.ukim.mk

Abstract. A new university information system that provides electronic services for both university management and students was developed for iKnow project and implemented at Universities in Macedonia. It is an eStudent system enabling complete electronic functioning of a University, rendering paper documents obsolete. The system is web based and implements state of the art modular service oriented technologies. This paper presents the modules and functionalities, software and database architecture.

Keywords: iKnow, electronic student services, university information system, student information systems, LMS, eStudent, MVP, MVC.

1 Introduction

The eStudent Information System iKnow is designed to store and administer student's records and personal files, as well as related university data. It is developed using innovative approach and knowledge management techniques. The system provides exchange of electronic information among all stakeholders: students, professors, administration, university management and the Ministry of Education.

The system consists of two main components, each consisting of modules: the new students Enrolment Student Services (ESS) and the Core Student Services (CSS). Special software modules have been developed for data migration from legacy software applications into iKnow using Excel template files.

The Enrolment component consists of the candidate's enrolment wizard, the enrolment forms for manual entry of candidate's data, candidate's data processing, ranking module and the enrolment results publishing.

The Core student services component includes:

- Module for administration (university and faculty), which encompasses users management, exams sessions, semesters and the integration with the learning management system (LMS) Moodle;
- Module for study programs and schedules, study programs management, courses management, equivalency of courses, connections between courses and programs

S. Markovski, M. Gusev (Editors): ICT Innovations 2012, Web Proceedings, ISSN 1857-7288

- Student activities module, the students online services, with its components: semester enrolment, exam application, courses selection, documents and certificates request, grades overview, diploma thesis management, the student services department for overall student data processing;
- Module for personal identification and access control
- Module for electronic payment and use of resources
- Migration of old data, interfaces to other systems

This paper is organized as follows: Chapter 2 gives an overview of other systems similar to the one presented in this paper. Details of the implementation of the system are presented in chapter 3, including the software architecture, database specifics and performance tests. Technical data about the hardware server infrastructure and security issues can also be found here. The customization and fine tuning of the user interface along with evaluation of the functionalities implemented in the system are presented in the chapter 4. Evaluation analyses of the software implementation are presented in chapter 5, followed by the concluding remarks in the chapter 6.

2 Related work

Information systems used for storing and organizing data about students and related university concepts and activities have been around ever since computers became available in the university environment. With the advancement and increased availability of computers as well as software platforms, such systems grew in complexity and features offered. Lately, with the rise of the Internet and its ubiquitous presence these systems became ever more student oriented transferring increasing number of tasks as well as opportunities to the students themselves. Operating such systems becomes increasingly the student's job offering them control over their educational choices as well as timely and transparent information they ought to have.

Today's members of e-Society expect each service they need to be available online. Young, IT era educated students are the potential primary users of such online services, rather than an exception to the rule. The availability of omnipresent Internet service through mobile devices anytime and everywhere increases expectations for higher education institutions to transform into 24 / 7 service providers. These expectations can only be met by deploying state of the art online student services that provide plethora of correct and timely information and enable instant feedback from students into the systems. Students expect to be allowed to apply for exams, read their exam results, choose elective courses or enroll in a semester by only scrolling their fingers on the touch screen of their mobile device anytime, everywhere. In the same time, the stored information must be safe, guarded from privacy breaches, consistent, readily available on demand and with permission only. On the other hand, educational institutions are expected to derive evermore complex statistical reports and archive data, provide more complex combinations of elective study programs, making their information systems increasingly strained and in need of features like extensibility, flexibility, modularity.

Currently there are multiple student information systems in use in universities across the world [15], [16], ranging from in-house developed solutions, up to from-

market of-the-shelf products sometimes adapted to the specific needs of the university in question. Also, they provide a variable range of online services for students, but very few offer complete electronic functioning and eliminated paperwork. In some instances CRM systems are adapted to serve as academic student info systems [17],[18],[19]. Also ERP solutions are sometime adapted for roles of student data storage [20], [21]. Systems use both open source and licensed databases with corresponding interfaces.

3 Implementation

3.1 Server infrastructure and security

In order to increase the performance and secure the stability of the system in face of the increasing number of concurrent users, a reorganization of the server architecture was performed engaging virtualization. Two servers and an independent storage system were employed at University Sts Cyril and Methodius, Macedonia. The first server is used as SQL server with Microsoft SQL Server installed and the second is used as an application server. The system is upgraded by installing 4 virtual Windows servers using VMware. Both servers have an intel xeon e5630 2.56 GHz cpu and 16GB of RAM, 1 TB storage.

Significant improvement of the responsiveness is achieved since virtualization and the multiple Internet Information Servers (IIS) that resulted avoid deadlocks and blockings often experienced in single, overused IISs.

Additional backup Internet link was installed in order to improve system availability in case of failures of the primary link. A third server on a remote location is installed that ought to be fully synchronized and take over in case of failure of the primary server farm or its Internet links.

The SSL protocol is used when accessing all system functionalities; both web forms and web services. A digital certificate was installed to encrypt all the communication.

3.2 Database architecture

The system is based on an SQL Server 2010 R2 database. Other database platforms are also integrated because of usage of external Learning Management Systems (LMS) like Moodle. The SQL database contains 145 tables so far, including several ASP.NET membership tables. The tables are connected with relations following the rules of a normal database form. However, certain redundancies were allowed in order to optimize query speeds and reporting efficiencies. Although the student's grade average can be calculated in real time, to avoid reporting delays and increased the efficiencies, the pre-calculated averages are stored for each student. There are also

other pre-calculated redundant data if they are often retrieved. Special attention is set to keep the consistency of these pre-calculated values. Recalculations are performed whenever the possibility arises for change of a pre-calculated value, and therefore the data consistency is guaranteed in the same time severely increasing efficiency. The database also contains 203 stored procedures used to increase efficiency and code transparency. **Fig. 1** gives an illustrative part of the database diagram.

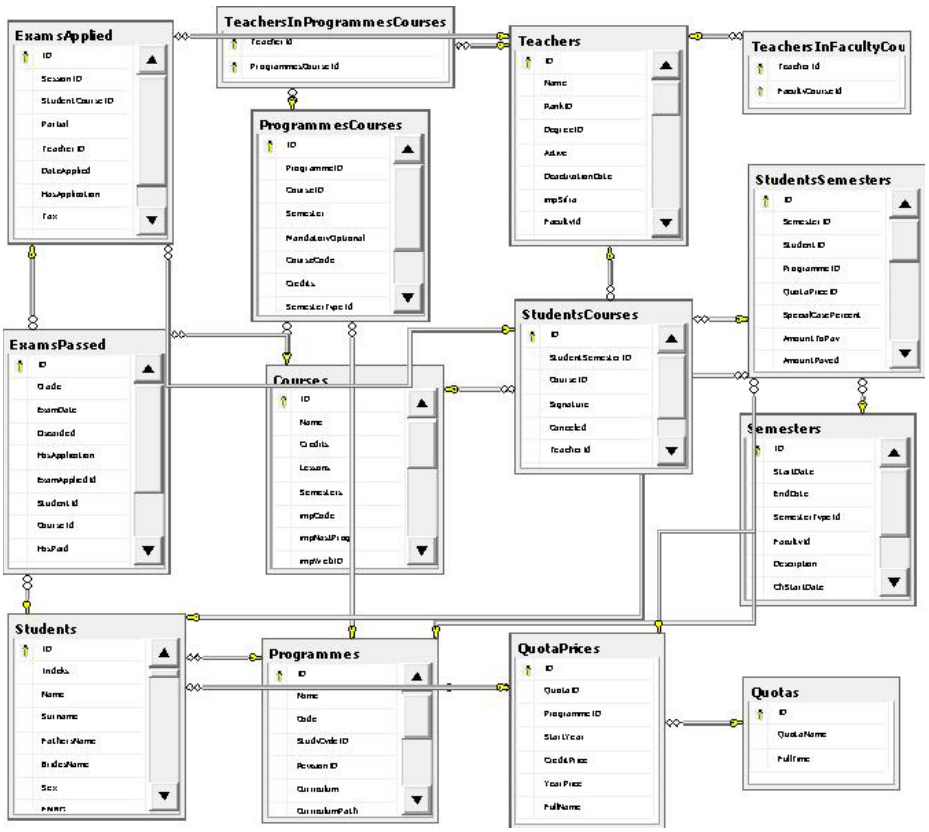

Fig. 1 An illustrative part of the database diagram

3.3. The MVP pattern used

The software architecture of the system strictly follows the principles of the MVP software design pattern.

Both the Model-View-Controller (MVC) and Model-View-Presenter (MVP) patterns have been use for several years [13], [14]. They both address the key principal of separation of concerns between the User Interface and the business layers.

There are several frameworks that are based on these patterns including: JAVA Struts, several PHP libraries, ROR, Microsoft Smart Client Software Factory (CAB),

Microsoft Web Client Software Factory, ASP.Net MVC framework etc. The ASP.NET Web Forms MVP [12] implementation of the pattern was used for iKnow.

The MVC pattern aims at separating the user interface - UI (View) from its business layer (Model). The MVP is a presentation pattern based on the concepts of the MVC pattern. In the iKnow pattern implementation of MVP the presenter interacts with a service (controller) layer to retrieve/update the model. One of the major advantages of the MVP pattern is the easy integration of unit tests. In the development of iKnow the advantage was used developing several unit tests in the view interface and service layer for testing the presenter and the model.

The clear separation of concerns and responsibilities between layers and components imposed by the pattern in the development of the iKnow system benefited the project in many ways. The major benefit was in the testing phase which showed that software bugs were relatively low in number. Also, the bugs were rather easy to correct since functionalities were separated and the errors were easy to locate.

Another benefit of using the pattern was the increased productivity due to code reuse. The project development speed increased as the system grew bigger since each new form or functionality was composed of controls that were previously developed for the preceding modules. Similarly, the flexibility of the used data layer enabled easily integrating different interfaces, ranging from the web forms for users to the web services for other third party systems.

One of the drawbacks of using the MVP pattern is the added complexity of the project. Since iKnow is a relatively big project, the complexity of the used pattern added at the initial stages resulted in an eventually cleaner, easily maintainable and upgradable project in the later stages. While unnecessary in smaller application, MVP is useful in bigger projects producing a high quality code structure that is relatively bug-proof if the pattern is strictly followed. The complexity of the pattern paid out later in the agile development phase when multiple functionalities changes were easily implemented without introducing instabilities in the system.

Another drawback of the pattern is the longer learning period even for relatively experienced developers that have not previously faced this architecture. This resulted in harder recruitment of developers in the team and inability to easily enlarge the team in face of incoming deadlines. However, after a learning period, different in length for every developer, the benefits of the high quality code produced compensate for the late engagement.

The ASP.NET Web Forms MVP [12] implementation of the pattern was used for iKnow. WebFormsMVP is an open source project.

The framework supports AJAX. Our experiments show that the speed and responsiveness of the application is more than satisfactory.

Performance tests were performed over the system using a tool capable to simulate multiple users accessing the software (WAPT 7.1 http://www.softlogica.com/). **Fig. 2** shows the average number of http errors depending on the number of consecutive users using the system. It is obvious that the system is stable in the sense that the number of errors is constant and independent of the number of users. The test lasted for 25 minutes, starting from 10 up to 90 users, increasing the number of users every 5 minutes with a rate of 20. The right axis and the black lines of the graph show the number of users, and the horizontal axis shows time. The red line shows the average number of errors with labels shown on the left axis.

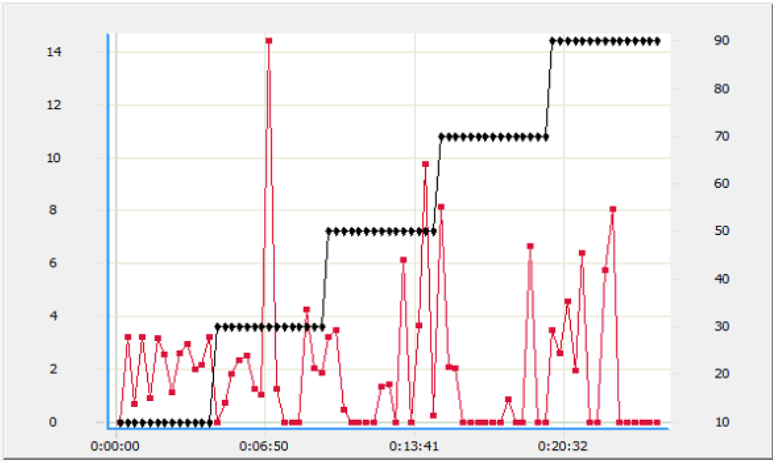

Fig. 2 HTTP errors in percent with increasing number of users

3.4 Software project architecture

All logical components in the system comply with the NTier concept, meaning division of the application in Layers and Tiers. The architecture of the system is presented in **Fig. 3**.

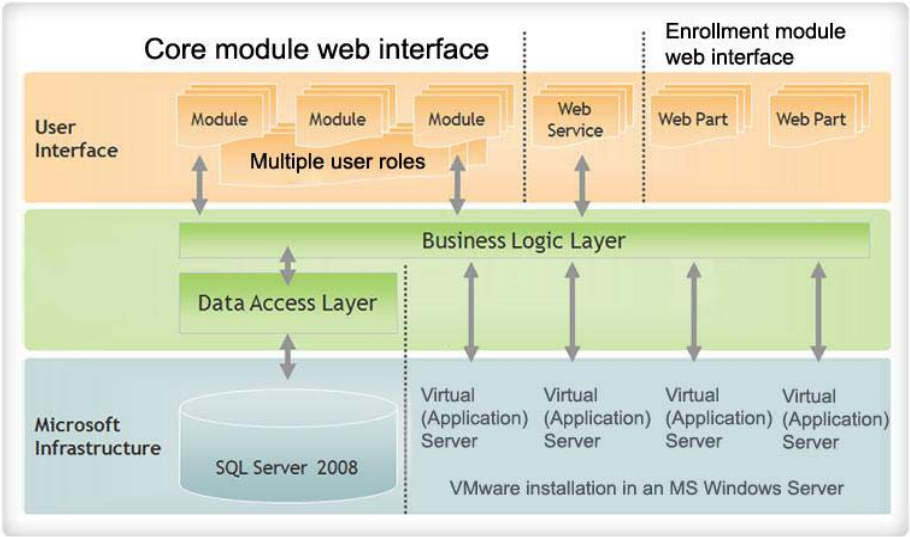

Fig. 3 Software architecture of the system

Several of the projects included in the complete iKnow solution are shortly described in the remaining of this chapter:

UniSS.DataModel is a Class library project containing the Entity framework model of the data. The model maps all tables, procedures and functions in classes and class methods. The model takes care of the overall communications and data flow from the database to the application and vice versa.

The audit log component is implemented in this project as well, logging all transactions from and to the database. The implementation is placed in the UniversityEntitiesCreator.cs class. Because the entity framework model is a partial class, and it is an extension to the model, it makes an addition to the OnSave event, by processing the XML of the changed data based on the old and new state of an entity.

UniSS.Repositories is a Class library project storing the classes for implementation of the Repository template. The classes of the implementation are in the subdomain (folder) Repos. The refactored entities that exist only in the business layer are in the subdomain (folder) Domain. For instance, if a list of students containing their index number, name and surname is to be rendered on the interface, it is not the complete Student entity with 50 columns that is retrieved, but rather a subdomain named StudentsRefactored, containing the aforementioned three columns. It is an optimization that decreases the operational memory used by the application as well as the time needed to process the data.

UniSS.Logic is a Class library project containing the implementation of the MVP template. The subdomain Models contains the models, Views contains the interfaces, and Presenters contains the presenters.

UniSS.Logic is a Web Forms application project in which all forms are separated according to their functionalities. The structure is based on the following hierarchy: Master Page, Web Page and UserControl. The user controls usually contain the entire business logic of the form and a Web Page implements the appropriate control. The Master contains the main layout.

The form named StudentsListReport.aspx is an example taken for illustrative presentation in this paper

Every form has a distinct Model, View and Presenter. All data that the form can manipulate is stored in the StudentsListReportModel model. The data consists of all control variables, the records needed for the form filters retrieved from the database, results of the search etc.

The IStudentsListReportView View component implements all the events in the form, for instance, when changing certain filter values the appropriate filtered data are retrieved and shown. When the selected value in a DropDownList component is changed, an event is triggered that refreshes the data in the grid. When clicking the Export button, the appropriate event is triggered, etc.

The presenter makes a connection of the view and the model, making an implementation of the event handlers of the view. When an event is fired, the appropriate repository calls a certain data retrieval function. The data is stored in an appropriate model variable

The function StudentsListReport defind in the repository is called, using LinqToEntities to query the database and retrieve data that is placed in the variable Students of the model.

In the user control named StudentsListReportControl in the UniSS.WebSite project, the appropriate events of the View are triggered, and after the successful

execution, the variables in the model are filled with data and bonded to the DataSource of the grid.

4 The User Interface

4.1 User interface specifics and implementation details

Systems offering electronic student services are complex information systems. They encompass multitude of functionalities provided to numerous users with different access privileges. Also, the users have different levels of familiarity to the system and the underlying processes. The user interface and it's performance is essential for the system's adoption as much as the systems reliability and scope of functionalities. [5],[6].

Agile development of a user interface is necessary due to multiple reasons:

- changing requirements [4]
- user satisfaction (constructive remarks by experienced users, habits gained by using legacy applications, trial period experiences , upgrade of technologies during development.

Users demanded several important features :

- grouping of functionalities on fewer forms for easy access and decreased need for navigation (**Fig. 4**))
- Screen size and resolution (conflicting) [2], [3]
- Simplicity and impossible to make an error [1]
- Automation of processes

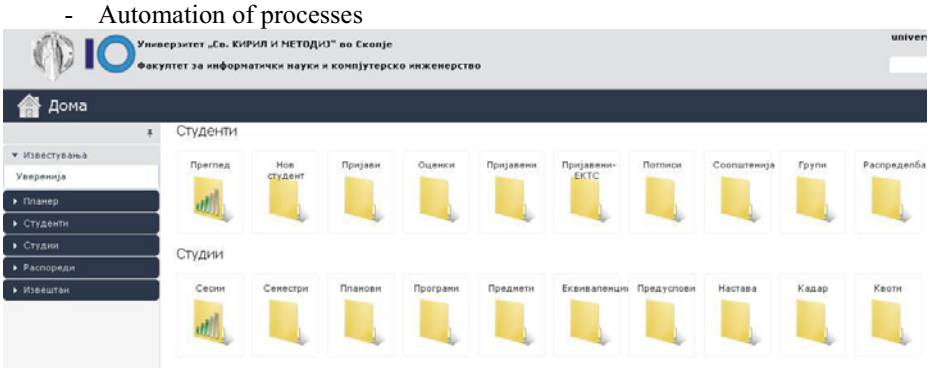

Fig. 4 Homepage – navigation through standard top and side menus or main part with folders

The system was developed as a web application with multiple user profiles and privileges. Users tend to avoid leaving the current page that they are working on, due to time-consuming navigation. Therefore the forms were heavily enriched with popups that could present additional information or provide means for inserting/updating data [7].

Standard web page popups were avoided as deprecated, and instead, hidden forms existing on the same web page were used (**Fig 6**, **Fig. 7**). Although effective in terms of work efficiency and user satisfaction, such forms tend to exponentially grow in complexity and size. The side-effects of such policies are complex forms prone to higher number of errors and extreme difficulty to test and validate.

Server side preparation of such forms presents a serious load on both the application and database server. Additionally, the size of the content itself is a burden to the Internet link since it generates a significant traffic. Since the demands are conflicting, a balance had to be reached for effective, but still lightweight forms that could be easily navigable. The advantages of AJAX and caching were heavily used to achieve the desired goals. For example, **Fig. 5** shows a form in which students choose the courses they will enroll during the semester. Multiple calculations are performed in real time generating the multiple selection list containing only courses that the student is allowed to take, the courses the student must repeat, the ECTS credit limitations in the semester, the financial implications of the selections, etc.

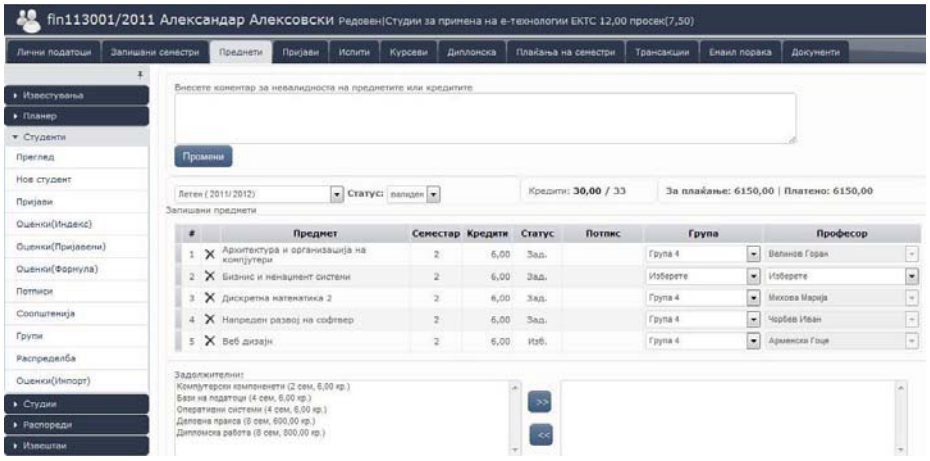

Fig. 5 Courses selection per semester, a backend complex form rendered easy to use

The screen size of the monitors used by the employees in the student services department varies from 15" up to 22". Therefore an adaptive interface had to be developed that is capable of using the benefits of large screens and resolutions, in the same time avoiding rendering small monitors useless. The content had to be visible and the interface usable in all screen sizes. Additionally, management used pads mainly for reporting, adding another layer of complexity, making the interface workable on touch screens and appropriate screen resolutions. Similarly students use Tablet computers or smartphones with variable, usually small resolution, demanding the interface to be robust and adaptable.

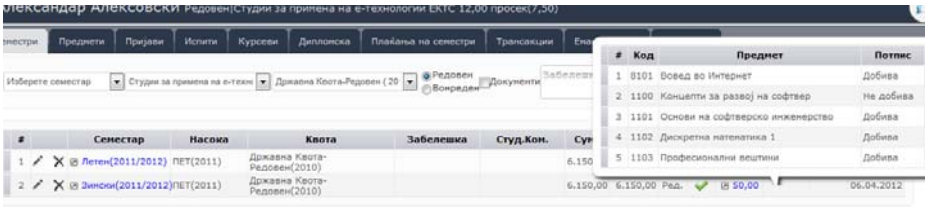

Fig 6 Overview of student semesters, expandable as needed with modal popups with additional data for SMS payments, courses enrolled in the semester, etc.

Although training is always thorough and unavoidable when deploying a complex information system such as this one, users tend to learn the system by "trial and error" and intuition. The user interface had to provide clear and short labels and messages, prevent from accidental deletions, and lead the user through the business processes. We have followed the design principle that simplicity is the key issue (**Fig. 7**).

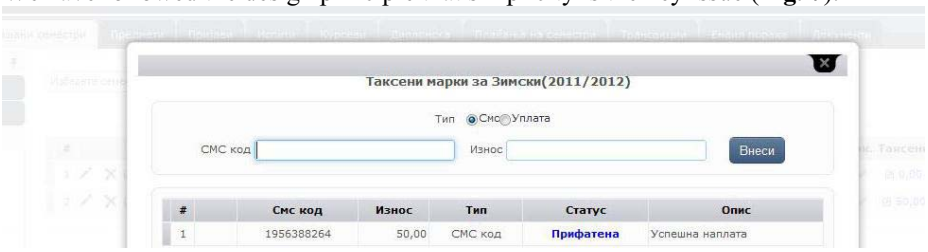

Fig. 7 Modal popup for SMS payment automation

Efficient use of the workforce in the student services department is only possible if the processes are as automated as possible, therefore bulk importing and fast input of grades for exams passed was developed.

4.2 User interface evaluation

In order to evaluate the usability and quality of the system, a questionnaire was developed and given to the end users to assess their opinions. The questionnaire consisted of 33 questions. Each question was answered by the users with a grade of 1-5. The user could answer with a 0 if he/she had no opinion on the matter.

The questions were organized in seven parts including: Accessibility; Layout; Navigation; Exception and status handling; User guidelines and online help; and Learning; and Content and Efficiency.

The results of the evaluation questionnaire were statistically analyzed and average grades were calculated for each of the questions. The users that participated belonged to 5 different faculties in the university bringing diversity of user backgrounds. The number of users questioned was 12. Out of the users questioned, 83.3% were female. Religious and cultural background of the users was also diverse. The answers provided were mostly positive.

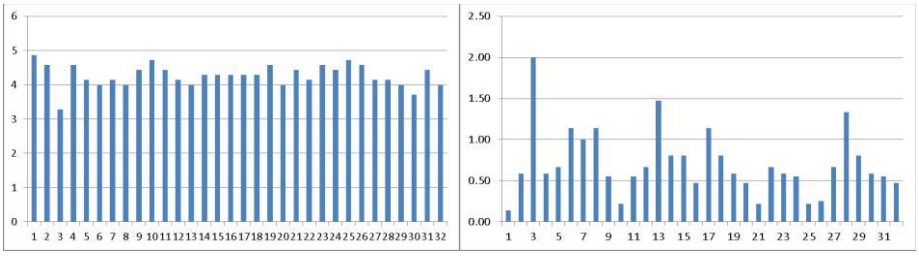

Fig. 8 a) Average grades given by the users in the questionnaire, b) Variance of average grades given by the users in the questionnaire

Fig. 8a presents the average grade awarded to each question by the users, while **Fig. 8**b presents the variance of the average grades.

The user satisfaction is evidently high, due to their participation in the user interface fine tuning and the efficient interface that resulted.

The Core component has generally been accepted as quite usable, easy to learn and work on. User's requirements have been implemented using agile development. Procedures and the user interface have been remodeled according to interviews with the initial users of the system. Adaptations of the interface have been made based on feedback from both teachers and student services employees. Students have confirmed the high usability of the application when asked.

The enrollment component is in the process of adaptation and customization based on the experiences during the first use of the system in August and September 2011. Problems in its use were noted, passed to the developers and the updated version is expected for the next term of enrollments (August 2012).

5 Evaluation of the software implementation

5.1 Detailed evaluation of the enrolment component

For the enrolment term in August-September 2011 the enrolment module was implemented with all planned functionalities except for the automatic transfer of the data for the enrolled students in the core module. Some faculties at the university completely relied on the iKnow system for enrolment (FCSE), while others used it in parallel with legacy systems, achieving identical results (Faculty of Natural Sciences, etc.) The faculties successfully completed the enrolment process using the iKnow system. Some smaller bugs noted during the enrolment process were addressed promptly by the developer that provided very agile response during the enrolment period. **Fig. 9** gives detailed statistical overview of the number of bugs in the Enrollment component.

Status	Total
Closed	333
In Progress	5
New	17
Pending	14
Rejected	20
Resolved	2
TOTAL	391

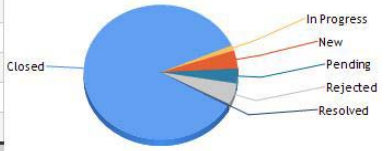

Fig. 9 Bug statistics of the Enrollment component

Student-candidates are generally satisfied with the solution. It's use is straightforward resulting in very few mails sent from students to the support team. The mails were often targeting data inconsistencies entered by the system administrators, and almost no mails were targeting issues of the functional system features and usability.

The processing of candidate applications by the enrollment committee was optimized for performance and accuracy. Two interfaces were developed to process applications: a wizard similar to the candidates interface for detailed analysis, and a short form for quick processing and updating. The later short form enhanced usability and significantly reduced the applications processing time for the enrollment committee. The experiences from the initial use are being implemented into the new version aiming at further enhancement of usability, reduction of complexity, discarding unnecessary steps and stages of each application. Automation of certain checks are also implemented (client side validators, consistencies of specific data types)

The main ranking of candidates for enrolment in different study programs was very rigorously tested prior to system roll-out, therefore no inconsistencies were detected.

At the focus of our interest in the nonfunctional requirements were the usability and the response times. The response time was measured during the testing phase and several optimizations were demanded. By the time the enrolment process started, the application had satisfactory response times. The response time averaged below 5 seconds for the most complex forms for fewer than 30 simultaneous requests, and below 10 seconds for the same forms for fewer than 50 simultaneous requests. The most frequently used forms by the students and the enrolment committee were optimized for its best usability. The main focus of optimization and design was aimed at the interface for student candidates, since the number of such users was to be measured in thousands. The strain of the servers was reduced to minimum, and the user satisfaction had to be high. Also, these users could receive no training; therefore they ought to face a trivially simple, yet fully functional error proof interface. Last minute optimizations were also made in the interface for the enrollment committees. Although of secondary importance, the speed of operation and the user satisfaction among the enrollment personnel was essential for complete adoption of the system in the entire university.

The measure of success of the implemented system is a very important issue in order to improve its functionalities in the future. We are using social media as Facebook and twitter to reach to the end users and gather their experience of their

eventual difficulties when using the system as well as gathering ideas for further improvements. In order to better explain the enrollment process to the candidates, social networks were used as a communication channel [23]. The channels were:

- Facebook page (Statistics shown on **Fig. 10**, **Fig. 11**)
- Twitter profile
- YouTube channel (**Fig. 12**)
- website (FAQ, Contact & Facebook live-chat)

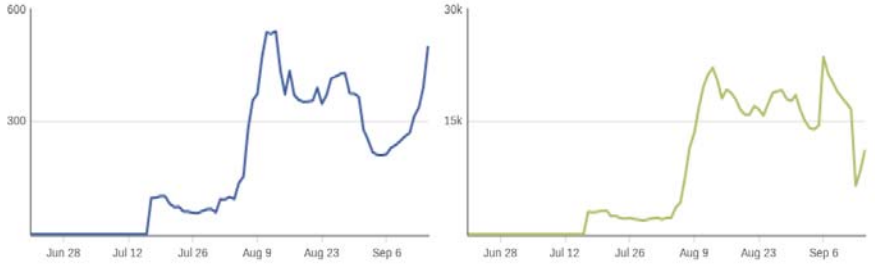

Fig. 10 Facebook - All stories, talking about stories (left) and viral reach (right) [23]

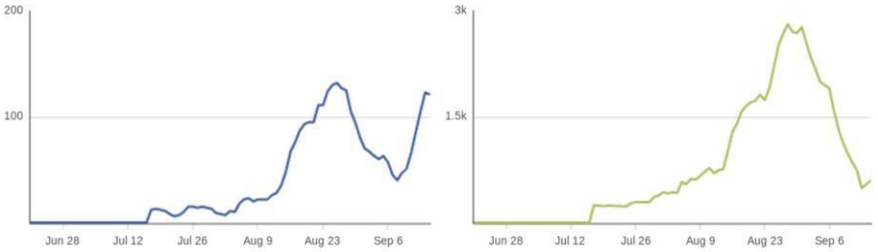

Fig. 11 Facebook – post by others, talking about posts (left) and viral reach (right) [23]

Fig. 12 Enrollment training video seen 803 times on the Youtube channel [23]

5.2 Detailed evaluation of the Core Student Services component

Most of the functionalities in the e-student services are fully implemented. Few of them are dropped out when implementing the system as unnecessary burden in this phase and several are pending - mostly reporting modules.

According to the specification document [9], there are 76 functionalities. Fully implemented are 46 functionalities, 24 are going to be implemented in the near future, 5 are not implemented in this phase, because other external systems should be established first, and 1 of them is discarded as irrelevant and an unnecessary burden to the system (administration of seminar thesis and reports, because mandatory seminar thesis are treated as any other course in the e-learning system). **Fig. 13** gives detailed statistical overview of the number of bugs in the Core component. The average time spent working on resolving software bugs is presented in **Fig. 14**.

Status	Total
Closed	774
In Progress	111
New	51
Pending	4
Rejected	2
Resolved	207
TOTAL	1149

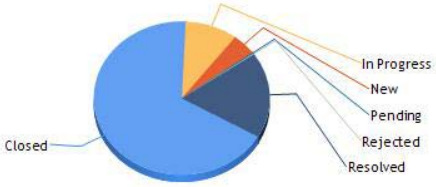

Fig. 13 Bug statistics of the Core component

AVERAGE TIME

Opening reported bugs	27.65 hours
Closing resolved bugs	23.71 hours
Resolving reported bugs	25.96 hours
Resolving rejected bugs	25.72 hours
Completing reported bugs	103.04 hours

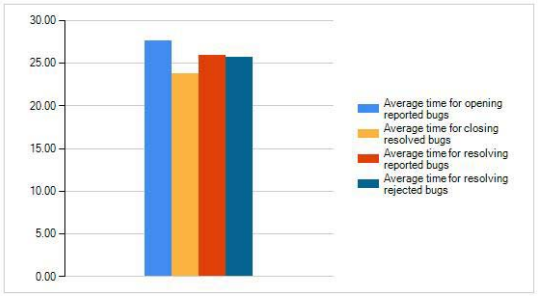

Fig. 14 Average time working od software bugs

Module for study programs and schedules

The question of status and quality of the system's functionality regarding the module for study programs and schedules in particular defining student programs, courses, prerequisites and rules for studies, is very important. These functionalities are implemented with special concern focused on making this interface easy to use with minimal steps in achieving results. Excel import with simple predefined templates was provided. Also, there are multiple different forms with the same functionality, offering options to various users to insert the data using several different methods, depending on the structure of the available data, user habits, the amount of data etc.

The functionalities of mapping of faculty staff to courses are also implemented. Teachers can be mapped in the courses menu, one teacher for all occurrences of the course in programs, or in the programs menu, with a different teacher for each course-

program link. The functionality of connecting equivalent courses is also implemented. Courses with different names or from different programs can be connected as equivalent so that students that change study programs during their studies can still keep their record of appropriate exams passed.

Student activities module

System's functionality regarding the student activities module with respect to the functionality of enrolment in a semester and selection of courses is implemented. Both students can do it by themselves or the student services employees can do that for them. The functioning is successful for both first time students that started using the system from the first moment of enrollment in the university as well as students that were migrated from legacy applications.

Forming groups at the beginning of each semester is a very important functionality in order to begin the course activities on time. This functionality is implemented. Students can be assigned to groups for each course they have selected, both individually, or multiple student at once, using excel import or multiple selection.

Module for administration

The system's functionality regarding the module of administration with respect to the administration of faculties and accredited study programs is implemented. Administration of members of the faculty is also implemented. Creation of users in the system is implemented in parallel. The administration of classrooms, rooms and laboratories is implemented. The administration of exams is part of the LMS, but the administration of earned ECTS credits and grades from exams passed is fully implemented. The functionality for storing and administration of personal records for students is implemented

Module for personal identification and access control

This module is implemented regarding the access control - currently, teachers can record the attendance of each student in the system. MS .NET membership access control is implemented. Based on special demands by the Faculty of computer Science and Engineering (FCSE), authentication was integrated with the Central Authentication Service used only by the FCSE users (both students and staff).

Module for presence monitoring and student activities is planned. The university is working in parallel on a distinct system for attendance control using RFID cards. Integration of both systems is planned.

Module for electronic payment and use of resources

Administration of payments by the students is implemented by automatic retrieval of SMS payments for administrative tax for students. Other payments are entered manually by student services department so far.

The administration of the use of learning management systems (LMS) is implemented by Moodle integration. In the iKnow system, teachers can demand creation of a Moodle course, and all students enrolled in the course in iKnow are automatically enrolled in the Moodle course as well.

Migration of old data, interfaces to other systems

The system supports easy migration of legacy data using imports of Excel templates filled with the old data. The imports are limited to students, exams passed, courses, study programs etc. However, FSCE being the first faculty that uses the system and having access to the database structure, achieved more thorough migration of enrolled semesters for legacy students and the courses taken in each semester.

The system incorporates web services that enable integration with external Learning Management Systems, Ministry of Education high school graduation storage systems, Biro of statistics. Also web services are available for developing external modules (mobile apps, etc).

6 Conclusion

This paper presents the details of the implementation of an advanced student oriented student services system that provides extensive list of functionalities to both students and staff using state of the art web software technologies. The integrated systems enables university level real time reporting and management. Students are provided with timely and secure information and opportunities to change information and their choices in the system by themselves anytime, anywhere. An interactive and intuitive user interface guaranties ease of use, reduced help and support effort and prolonged life of the system.

Systems like the one presented never stop to grow and change. Legislation changes along with new ideas to improve high education and provide students with new services and choices. Therefore the modularity and extensibility of the system in question is a key issue that ought to guarantee the systems future and growth. The build-in web services that await their use in native mobile applications are an example of such forward thinking. Every important piece of information in the system can be retrieved or changed using the secured web services that can easily be integrated in Android, iPhone or Windows Mobile apps. Such examples are being developed and will be readily available. Similar web services are developed to enable integration with external Learning Management Systems, Ministry of Education high school graduation storage systems, Biro of statistics etc.

Another forward thinking feature of the system is the idea of providing software as a service. In the era of evermore growing popularity of the concept of cloud computing, this architecture hosted on a cloud can be adapted with relatively minor effort to server multiple universities bringing all the known benefits.

References

1. Anthony Vance, Braden Molyneux,Paul Benjamin Lowry , "Reducing Unauthorized Access by Insiders through User Interface Design: Making End Users Accountable", Hawaii International Conference on System Sciences, January 2012, pp. 4623-4632
2. Jörg H. Mayer,Timm Weitzel , "Appropriate Interface Designs for Mobile End-User Devices--Up Close and Personalized Executive Information Systems as an Example", Hawaii International Conference on System Sciences, January 2012, pp. 1677-1686
3. Gang Huang,Daimeng Wang, "Adapting user interface of service-oriented rich client to mobile phones", Service-Oriented System Engineering, IEEE International Workshop on December 2011, pp. 140-145
4. Lu Xudong, Wan Jiancheng, "User Interface Design Model", Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Eighth ACIS International Conference on, July 30 2007-Aug. 1 2007, Volume: 3, Page(s): 538- 543
5. Massila Kamalrudin,John Grundy, Generating essential user interface prototypes to validate requirements, "Automated Software Engineering, International Conference on", November 2011, pp. 564-567
6. Shinichi Inenaga,Kaoru Sugita,Tetsushi Oka,Masao Yokota, "Performance Evaluation of User Interfaces According to User Computer Skill and Computer Specifications", Intelligent Networking and Collaborative Systems, International Conference on, December 2011, pp. 446-449
7. Shinichi Inenaga,Kaoru Sugita,Tetsushi Oka,Masao Yokota , "A Preliminary Evaluation for User Interfaces According to User Computer Skill and Computer Specifications", P2P, Parallel, Grid, Cloud, and Internet Computing, International Conference on ,October 2011, pp. 295-298
8. David R. Karger, "Creating user interfaces that entice people to manage better information", Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM '11), October 2011, pp. 1-2
9. Project Tempus JPGR 511342 – iKnow http://iknow.ii.edu.mk
10. Ivan Chorbev, Marjan Gusev, 'iKnow Student services" System documentation of Project Tempus JPGR 511342, 2010-2011, http://iknow.ii.edu.mk
11. Dejan Gjorgjevik, Marjan Gusev, "iKnow Enrolment module", documentation of Project Tempus JPGR 511342, 2010 – 2011, http://iknow.ii.edu.mk
12. http://wiki.webformsmvp.com/index.php?title=Main_Page (1.8.2012)
13. http://www.infragistics.com/community/blogs/todd_snyder/archive/2007/10/17/mvc-or-mvp-pattern-whats-the-difference.aspx (1.8.2012)
14. http://msdn.microsoft.com/en-us/magazine/cc188690.aspx (1.8.2012)
15. http://schoolcomputing.wikia.com/wiki/Student_Information_Systems (1.8.2012)
16. http://www.dmoz.org/Computers/Software/Educational/Administration_and_School_Management/ (1.8.2012)
17. http://www.microsoft.com/australia/education/schools/solutions-for-schools/customer-relationship-management.aspx (3.8.2012)
18. http://crm.dynamics.com/en-us/education (3.8.2012)
19. http://civicrm.org/node/586 (3.8.2012)
20. http://www.jenzabar.com/products.aspx?id=102 (3.8.2012)

21. http://www.sap.com/industries/higher-education-research/index.epx (3.8.2012)
22. Milos Jovanovik, Vladimir Zdraveski and Marjan Gusev, Social Networks for Customer Relationship Management, Molika, Bitola, CIIT 2012
23. Dejan Gjorgjevikj, Ana Madevska Bogdanova, Ivan Chorbev, Marjan Gusev, Implementation of electronic student services at UKIM, Molika, Bitola, CIIT 2012
24. Ivan Chorbev, Marjan Gusev, Dejan Gjorgjevikj, Sasko Ristov, User interface agile development and evaluation of the iKnow student services system, Molika, Bitola, CIIT 2012

Appendix A: MVP and MVC pattern detailed elaboration

There are several frameworks that are based on these patterns including: JAVA Struts, several PHP libraries, ROR, Microsoft Smart Client Software Factory (CAB), Microsoft Web Client Software Factory, ASP.Net MVC framework etc.

The MVC pattern aims at separating the user interface - UI (View) from its business layer (Model). The pattern separates responsibilities within three components:

- the view is responsible for rending UI elements,
- the controller is responsible for responding to UI actions,
- the model is responsible for business behaviors and state management.

In most implementation of the pattern, all three components can directly interact amongst each other. In some implementations the controller is even responsible for determining which view should be displayed (Front Controller Pattern).

The MVP is a presentation pattern based on the concepts of the MVC pattern. The pattern separates responsibilities within four components:

- the view is responsible for rending UI elements,
- the view interface is used to loosely couple the presenter from its view,
- the presenter is responsible for interacting between the view/model,
- the model is responsible for business behaviors and state management.

In some pattern implementations the presenter interacts with a service (controller) layer to retrieve/update the model. The view interface and service layer are often used to write unit tests for the presenter and the model.

There are a lot of benefits as well as drawbacks when using either the MVC or MVP pattern. The biggest drawbacks include the additional complexity and learning curve. While the patterns may not be appropriate for simple projects; advanced software systems can greatly benefit from using a pattern.

Several major benefits from using design patterns are presented in the list below:

- The presenter or controller serves as an intermediary between the UI code and the model allowing the view and the model to evolve independently of each other.
- Clear separation of concerns and responsibilities
- Testing - by isolating each major component it is easier to write unit tests. This is especially true when using the MVP pattern which only interacts with the view using an interface.

- Code Reuse - especially true when using a complete domain model and keeping all the business logic and state management procedures in appropriate modules.
- Hide Data Access - Separating and distancing from the data access enables replacing the database platform in future implementations and other independent changes in the database.
- Flexibility and Adaptability - A properly design solution using MVC or MVP can support multi UI and data access technologies at the same time.

The key advantages of the MVP pattern include:

- The view is more loosely connected to the model. The presenter is concerned with binding the model to the view.
- Easier to unit test because the interaction with the view is channeled through an interface
- Usually one view is mapped to one presenter; however, complex views may have multiple presenters.

The ASP.NET Web Forms MVP [12] implementation of the pattern was used. WebFormsMVP is an open source project created by Tatham Oddie and Damian Edwards.

The WebFormsMvp open source framework utilizes the following components [12]:

- Presenter: This is a class that inherits from Presenter or Presenter<T>.
- View interface: This is an interface that inherits from IView or IView<T>.
- View Implementation: The page or user control that implements the view interface.
- Event Arguments: A custom event argument class is the way to pass data from the view to the presenter.
- Model: This is a class that contains the data needed

Appendix B: Questionnaire

The questions were as follows:

Accessibility

1. How do you judge the information of the launch and the training?
2. How do you access the process of registration?
3. Does your browser display all information correctly?
4. Is site load time appropriate to content and response?

Layout

5. Text-to-background contrast
6. Is the font size and style easy to read?
7. Does the site have a consistent look and feel?
8. If you have a disability regarding your eyesight: Is the content readable?
9. Is the label location and format consistent?

Navigation

10. Are the major parts/menus of the site directly accessible from the main page

11. Are the navigation labels clear and descriptive?
12. Is the workflow navigation consistent and easy to identify?
13. Is the respective location within the process (site) transparent?
14. Is the site search easy to access?
15. Is the exit point clear on each page?
16. Does it require minimal steps in sequential menu selection?

Exception and status handling

17. Are the messages regarding status clear and descriptive?
18. Are the messages regarding exceptions/errors clear and descriptive?
19. Position of messages on screen is good

User guidelines and online help

20. Is the site designed to require minimal help and instructions?
21. Is the help and instruction information easily accessible?
22. Is there an easy channel available to communicate with an administrator?

Learning

23. Easy to learn to operate the system
24. Easy to explore new features by trial and error
25. Easy to remember names and use of commands

Content and Efficiency (refers to all information i.e. field explanations)

26. Is the content understandable?
27. Is the content well-structured and correlates to your requirements?
28. How do you evaluate the support of the system?
29. Do you observe an increase in efficiency?
30. Does the system provide a sufficient number and quality of reports?
31. It is easy to use.
32. What is your overall evaluation of the system?