Student Management Information System for Polish Universities

Janina Mincer-Daszkiewicz

Faculty of Mathematics, Informatics, and Mechanics,
Warsaw University, Poland
jmd@mimuw.edu.pl

Abstract

The purpose of this paper is to report the results of the New Educational Tools Tempus JEP project (JEP-14461-1999), which started November 1999, ended December 2001, involved all 17 Polish universities, and — in our opinion — was a great success. The project objective was to design and introduce an integrated Student Management Information System (SMIS) for academic structures and student affairs at Polish universities. In two years we managed to gather information on study programs and administration procedures in use at the participating universities, to produce fully functional software, and deploy it on some of the faculties (others are now in various stages of this process).

What makes this success more significant is that the software was produced in-house, by the Faculty of Mathematics, Informatics, and Mechanics, Warsaw University, by students supervised by academic teachers and Ph.D. students. Software was designed and developed on software engineering and database courses, and on master seminars. This can be considered a very successful experiment in the didactics of software engineering at the academic level.

Polish universities are now in process of creating consortium (similar to Swedish LADOK [4]) responsible for further development and future support of the system (as well as other common activities of the participants).

Keywords: Student Management Information System (SMIS); University *SOS* (USOS); Software engineering in education

1 Introduction

The Tempus JEP project called New Educational Tools started November 1999 and ended December 2001. The greatest strength of the project was the active participation of 17 Polish universities[1], highly determined to undertake an initiative to unify university administration procedures and develop common software.

EC partners from Swedish LADOK consortium and universities from Belgium, Germany, Portugal, and United Kingdom also participated in the project and offered their advice and expertise to Polish partners.

The main project objective was to design and introduce an integrated Student Management Information System (SMIS) for academic structures and student affairs at Polish universities. In 1999 only some Polish universities had computerised databases for that purpose, mainly in Desktop database technology (Clipper, Access or Foxpro), functioning at the faculty level. The greatest challenge of the project was to conduct requirements engineering at the participating universities and come up with the software package which would be accepted by all partners, used at the university level, and possibly approved by the Ministry of Higher Education as standard.

One of the very first conclusions was that none of the software packages currently in use can be easily modified to fulfill the needs of all participants. It also quickly became clear that no Polish software company is able to provide the application software needed for the national level university-wide database with functionality requested for SMIS, mostly due to the limited budget of the project, but also because of the lack of expertise on study programs and procedures involved in university management.

The help came from the authorities of the Faculty of Mathematics, Informatics, and Mechanics (MIM) at Warsaw University (UW), who decided that the needed software would be designed and developed in-house, by the teachers and students of the Faculty. By engaging students as programmers the problem of the low budget could be solved. And of course no software company could compete on the ground of expertise in academic procedures with the team of teachers, students, and authorities settled in university environment.

The development team was lead and supervised by the vice-dean of the Faculty. Academic teachers designed, co-ordinated and supervised students' job, students worked as programmers and leaders of sub-projects. Software was designed and developed on software engineering and database courses, and on master seminars. This can be considered a very successful experiment in the didactics of software engineering at the academic level.

At the end of the Tempus project (December 2001) the production version of SMIS, called USOS (the term comes from "*Uniwersytecki System Obslugi Studiow*" — in English: "*University Study-Oriented System*" — or, according to some university officials, "*University SOS*") was in use at the Faculty of MIM UW, and few other faculties from four different universities. In Warsaw the USOS package had been used since the beginning of the academic year 2000/2001. In particular at the end of the academic year students from the 2000 admission had been evaluated by built-in automatic procedures (only for those students the data kept in USOS is complete). It is the best

[1] In Poland there are 17 state higher education institutions called *universities*. All of them participated in the project. Detailed information on the Polish system of higher education is available in [3].

evidence that the developed software is fully functional.

In January 2002 we started deployment of USOS at three other faculties of Warsaw University. We hope to start the academic year 2002/03 with USOS being fully operational as university-wide software.

2 USOS — Requirements Analysis

The requirements analysis was an especially important stage of the software development process. In Poland faculties of the same university may differ substantially in the study programs, examination procedures, financial rules, and academic activities. The problem is even more significant when different universities are considered. Some follow more traditional approach in which programs do not support interdisciplinary studies and give little flexibility, as the large majority of courses are compulsory. Other encourage student-driven education and allow students to select individual courses to form their profiles. Only few universities use ECTS to monitor students' progress, although most of them mention credit points in study guides.

The software designers started with the thorough analysis of university study guides. A special Internet application was implemented for delivering requirements through forms accessible by web browsers. We advertised this application during one of the workshops and encouraged contact persons from other universities to fill it with the detailed requirements which could not be identified otherwise.

The news group was formed with the purpose to create the forum for discussion and exchange of ideas. Also the project web page ([5]) was posted on the net to suport easy access to information.

It has been decided that USOS should support management of student affairs (with exclusion of payroll and human resources), in particular handling of the following data and activities:

- students' and teachers' personal data,
- study programs and requirements of degree certificates, along with the ECTS standards,
- course catalog (complete list with details; some of them are persistent, like syllabus, prerequisites, credit points, others depend on course edition, like course instructors or weekly class schedule),
- course registration,
- class schedules,
- dormitories,
- tuition and financial aid,
- issuing documents, gathering statistics, producing reports, etc.

Many non-functional requrements had been recognized. The system should be:

- **university-wide**: there should be one global database for storing data concerning all organizational units of the university;
- **flexible**: the software should be configurable, which means adaptability to programs and procedures of all participating universities. It should not only deal with standard administration procedures but also allow for non-standard solutions;
- **open**: the system should be easily adaptable to (possibly frequent) changes in study programs;
- **secure**: the multilevel hierarchical system of priviledges should be supported;
- **compatible with ECTS standards**;
- **user-friendly**: services offered to various groups of users should be coherent and available through an easy and intuitive interface (this is especially important for users who are not professionals in Information Technology).

It had also been recognized that various access methods should be supported for various groups of users, e.g.:

- administration officers and university authorities need reliable and secure access to highly confidential data,
- students and teachers need easy, but personalized and authorized access to personal data, e.g. students want to check their grades and progress within the study, browse course catalog and register for courses, whereas academic teachers should be allowed to browse or modify course descriptions and fill up examination sheets;
- the public (e.g. prospects) need an easy and unlimited access to information on university program offer.

3 USOS — Software Design and Implementation

The main goals of the project seemed almost contradictory. On the one hand the software should support university-wide repositorium of data with easy access for students, faculty and administration officers, whereas on the other it should ensure security of data and lower administrative expenses. The average faculty at Warsaw University has about 200–300 academic teachers and 1500–3000 students, and there are about 18 faculties and 25 other organizational units. The solution with one central database accessible for all kinds of users would mean high licensing costs, possibly poor performance at rush hours, and endangered privacy of data.

To meet the stated goals we designed the following architecture for USOS:

1. the main database is implemented in Oracle technology; it is accessed only by administration staff via Oracle forms and reports. Oracle technology ensures effective handling of huge amounts of data accessed simultaneously from distributed client sites. The EU funding allowed for the purchase of some number of Oracle licences to start the project: for software designers and developers (Oracle Designer, Oracle Forms, Oracle Reports), and for end clients;

2. there are also a few Internet databases with massive numbers of clients accessing them via web browsers; this part of software is made in open-source technology. Internet databases are partial copies of the main Oracle database. In particular highly confidential personal data are not stored on the web, data are available mostly for reading, and modifiable data comprise only a small subset. All data transfers are encrypted. The Internet part of USOS (called USOSweb [6]) can be integrated with departamental portals, making the whole service coherent and easy to use.

Such architecture allows to avoid high cost of licensing which would arose from permitting massive access to the central database, It also increases scalability of the solution (e.g. we may install one web database per faculty) and the security (confidential data are only available to limited number of users and are not endangered by web access).

On the other hand this architecture is more complex from the technological point of view since databases have to be synchronized by special algorithms and mechanisms (cf. Sect. 4).

The project was developed in accordance with the CDM Fast Track methodology ([2]), designed by Oracle, which is a standard for enterprises conducted in RAD technology. We also applied some practices of Extreme Programming ([1]). The important techniques described in the paragraphs to follow were used in the project.

For each task in the project its **priority** had been determined and **strict time limits** were specified for conducting this task. This was particularly important due to the needs of the end-users expecting deliverables on time, and due to psychological aspect of keeping active involvement of participants and developers.

Workshops were very effective technique for information gathering and decision making. Their particular importance implied from the scale of the project involving universities spread all over Poland. We organized workshops at least every few months to keep universities informed about the progress made and to involve them in decision making about the possible direction of software development.

Prototyping was used from a very early stage in the project. The character of the project allowed us to deliver prototypes quickly, soon after having agreed on the first functional requirements. Prototypes were delivered to end-users and helped in getting quick feedback on functionality and usability of the product. Early prototypes offered limited possibilities for automatization of various procedures for groups of students (students could only be handled individually), supported only few most important reporting facilities and almost non global statistics. Nevertheless they were used by administration officers who could comment on interface, test correctness of implemented routines, or point out missing options. The close contacts between designers, programmers, and customers were one of the most valuable aspects of developing software in-house.

Iterative development was driven by feedback from end-users. They worked with prototypes and came up with ideas for modifications. Requirements were not fixed at the start point of the project, but were developed along the way.

Since various tasks of the project were conducted by various groups of students we separated those tasks on the basis of the size of the group and time deadlines for the expected functionality. **Partitioning** of the overall project into smaller tasks was an important activity, driven by deliverables expected at each milestone.

The project was lead by the vice-dean of the faculty, who had the **authority** to state goals, prioritize requirements, make decisions and enforce their implementation. Being an academic teacher in computer science she could also take personal responsibility for didactic aspect of the project.

From the very beginning we were aware that the final success of the project depends heavily on our ability to offer the functioning software at the end of the Tempus project. Tempus helped to get all universities involved in a common task. However, such involvement usually does not last long. From the psychological point of view it was critical to deliver the useful product before the obligation and the will to work together ends.

The RAD approach helped to achieve this goal. The contact persons from participating universities played an active role in communicating and exploring requirements throughout the project, and participated in decision making. Prototypes delivered for use and testing helped to deploy the software at some faculties before the end of the Tempus project.

4 USOS — Some Implementation Details

In this Section we describe some implementation details of USOS, which might be interesting for wider audience.

4.1 Databases Synchroniser

As it has already been explained, to limit costs of Oracle licenses and ensure security, we decided to separate the main USOS Oracle database from the web and make it accessible only to authorised members of university administration. Simultaneously, however, we keep some auxiliary databases which can be accessed by students, faculty members, and general public using web browers. In those web databases we keep only subset of data stored in the main database and in case of failure we can rebuild them quickly. However, since data may

also be entered through the web interface to web databases (when students register to courses and academic teachers fill examination sheets) we need some tools for database synchronization.

In the prototype version of USOS we implemented very simple way of transferring data by files. The needed data from the Oracle database was written into a script file in form of SQL insert statements and then this script was run to fill in web databases.

Student registration and assessment data were transferred from web to Oracle by a text file as raw data. This file then was read into temporary tables at the Oracle site. Then the temporary tables were checked for consistency with original Oracle tables. The synchronisation consisted in tranferring consistent data to original tables and delivering inconsistent data to human operator for decision.

This procedure worked satisfactorily at the early stage of USOS deployment. However it had some drawbacks:

- Oracle to web data migration always involved all data needed at the website (even if only a small piece of data had changed at the Oraclesite). Its amount grew very quickly over time;
- every time the Oracle database structure had changed code of the migration procedures had to be updated. It happened a few times that programmers changed database structure but forgot to make necessary corrections in the migration procedure. This solution does not comply with the software engineering rules such as information hiding or modularisation and makes software difficult to maintain.

Finally we designed and implemented much more flexible and generic tool which adapts automatically to the changing structure of the Oracle database. This tool, called Synchroniser, connects remotly to both databases, reinstalls chosen tables at the website in case their structure had changed at the Oracle site and then synchronizes tables contents. It uses auxiliary tables at the Oracle site to keep track of changes made since last synchronisation. Only changed records are exchanged between databases. Again, in case of conflict (changes have been made simultaneously at both sites and they are not consistent), Synchroniser stops automatic data transfers and asks the operator for the decision.

4.2 Authorised access to subsets of data

USOS has been designed as university-wide application, in particular data concerning all faculties are being stored in its database. On the one hand it gives the opportunity to deliver various global level statistics and reports. On the other hand the new problems emerge of unauthorised access to data or access to excessible amount of data. The solution to the latter problem is discussed in the next section. Here we consider the problem of authorisation.

Of course only authorised users can log into the Oracle database and access data stored there. However, various users may have various access priviledges, according to the roles they play in the real life. Even at the faculty level we may recognise various such roles, e.g. the role of a clerk responsible for first grade students of Computer Science, who can select, insert, delete, and update records concerning those students, or the role of a faculty dean who can select records concerning all faculty students (for example to generate global report). Of course one of the possible solutions was to deliver separate interface for each role. That would, however, mean the necessity to maintain all those interfaces and to program it for each new role which may come out in the future. We have chosen more flexible but also more technically sophisticated solution. Roles, meant as privileges to select, insert, delete, and/or update rows and columns of database tables (determined by SQL statements) can be freely defined in USOS and then assigned to USOS users. So we may define a role of a ”faculty dean”, a role of a ”clerk responsible for first grade Computer Science students”, or a role of a ”a clerk responsible for financial aid for students”. Every user can have a few roles assigned but only one of them is active at a time. Only Roles Administrator can define roles and assign them to other users, but every user can by himself activate one of his roles.

The whole USOS interface is transparent to the system of roles. A user cannot read data it is not authorised to by his role (database select statement delivers empty set of records). When a user tries to modify records he can select but not update, the “insufficient priviledges” message is being displayed and the operation is aborted. The interface (Oracle forms) had to be designed and implemented in compliance with this requirement. The solution is very flexible since no Oracle programming is involved in defining a new role, and it can be done any time the need arises.

4.3 Data filtering

The system of roles solves the problem of authorised access to subsets of data, but does not solve the problem of having to deal with excessible amount of data. For example, clerks at the faculty of Mathematics are, most of the time, interested in data concerning math students. They don’t want to see records concerning students of biology or chemistry. On the other hand they have to be allowed to read these records when needed (e.g. when a biology student decides to study math). What is needed is the flexible mechanism of filtering the data. By default a clerk at our faculty wants to see only math students, but from time to time he also wants to get access to other students as well. In USOS we deliver such functionality by a system of filters. A filter has its name and can be defined by some SQL statement (e.g. a filter called ”math student” or ”biology student”). Filters can be freely made available to users, and various sets of filters can be accessible on various Oracle forms. On each such form one of the filters is defined as default. A user can by himself define new filters, determine their availability on various forms and choose default filters. No Oracle

programming is involved in defining and configuring filters.

4.4 Checking degree requirements

The module for student assessment is one of the most sophisticated parts of USOS. This is mostly due to the student-centric character of modern universities. The software for handling study programs at such universities should support flexible study systems. In particular it should allow students to select individual courses to form their profiles, as well as should encourage interdisciplinary studies.

This means that the software should allow not only to state degree requirements for some predefined profiles but also for individual students and their dynamically built-in profiles. USOS supports such flexible requirements definition while simultaneously delivering automatic procedures for checking whether those requirements have been met. The set of degree requirements can be copied for individual students and modified accordingly. The requirements can be stated by pointing a fixed set of courses or a given number of compulsory courses picked up from some group. Those groups can also be freely redefined according to changing criteria, e.g. we may state that to be promoted from a second to third grade a student should take eight courses from group *A*, but not more than three from group *B*. We can also state requirements for so called conditional promotion. All those requirements can be checked automatically and a status of a student's progress can be updated accordingly. The automatic system decisions can be verified at each stage and also updated by the operator (who is, of course, the ultimate authority).

5 USOS — Software Deployment

At the end of the Tempus project USOS had been deployed at pilot faculties of the participating universities. The process of software development and deployment proceeded in the following steps:

- after a new functionality had been developed, the necessary changes had been made to the copy of the main Oracle database and newly implemented Oracle forms had been installed to be used by few testers and the project leader;
- after this testing phase, the structure of the main faculty Oracle database had been updated and the new forms had been delivered to administration staff. They used the prototyped version of software giving quick feedback on its functionality and usability to software developers;
- only after the prototype had been in use at MIM UW for some time, the new distribution package of USOS was prepared and offered to other universities. They also delivered quick feedback.

In January 2002 we started to deploy USOS at other faculties of Warsaw University. University-wide deployment involves many important decisions, steps and activities:

- USOS is not the first and only database application used at Warsaw University. There is another application (called HMS) for managing payroll and human resources. This application had been designed a few years ago and is still supported by a software company which owns the source code. It has not been our intention to build HMS functionality into USOS. However, there are some data with are stored in both databases (e.g. information on university staff), and thus to maintain their consistency we had to agree on using common database dictionaries. It was a good opportunity to spend some extra time on these dictionaries and come out with the better design, more suitable for future needs. In USOS there will be a special role for a user authorised to define or change content of system dictionaries.
- To keep data consistent we had to decide that it is entered into one database and exported from it to the other. A special data export procedure had to be designed and implemented, to transfer data from HMS to USOS. Such data transfer has to be performed for each new faculty deploying USOS, and will have to be done periodically in the future for data updates.
- Some faculties have partial data concerning study programs and courses, gathered in various desktop databases or spread sheets. On the one hand we want to transfer automatically as much data from those sources as possible, but on the other we have to check carefully for data consistency (it is very easy to enter inconsistent data into spread sheets) and cross-check data shared by various faculties (e.g. students majoring in two programs).
- At this stage new roles have to be defined and tested for representatives of administration staff from different faculties comprising various access rights to shared data repository.

We do not rush at this stage since we consider it very important to unify dictionaries, coding rules, and administration procedures employed at various faculties at as early stage of software deployment as possible. We hope to start the academic year 2002/03 with USOS being fully functional as university-wide software.

6 USOS — Education in Software Engineering

The main goal of software engineering courses offered at universities as part of Computer Science curriculum is to make students practice "programming in the large" (cf. [9]). The emerging problems are that programming in the large means taking part in real life projects, aiming at the production of commodity software. Students rarely have an opportunity to participate in all stages of the software process: specification, verification, design, implementation, testing, and maintenance.

They are not motivated to meet the requirements, work according to time limits, bother for quality of the final product, take part in its deployment and maintenance (cf. [7]). For example, how to make them interested in designing for change and software reuse if the developed products live no longer then until being graded by the course instructor?

The Tempus project gave us, responsible for software engineering education at the Faculty of Mathematics, Informatics, and Mechanics, a rare opportunity to integrate education with production of commodity software, to involve students in a real project, teaching them software engineering methods and tools along the way.

This is not the first time students at university take part in producing commodity software. What makes USOS project different from the others is the scale and multilevel integration of design, development and deployment activities with education.

After two years of experience we may summarize the advantages and disadvantages of that idea (e.g. [8]). Among the advantages are:

1. students take part in all stages of a real software development process, including software deployment and maintenance;
2. students learn how to be team members, how to collaborate with colleagues from various groups, study programs, and even faculties;
3. students learn how to use professional tools, meet quality and performance requirements, apply standards, feel responsibility for the product, confidentiality and integrity of the handled data;
4. students get involved in the project even not participating in it personally (every student of the faculty is aware that software is developed in-house by the faculty members and computer science students). They offer advice concerning interface and functionality of the system, take part in testing;
5. last but not least the academic community obtaines the high quality software.

The disadvantages are the following:

1. such project demands much higher involvement of the academic staff, substantially exceeding the ordinary academic obligations;
2. the necessity of giving priority to academic goals may be hard to reconsile with project goals, e.g. students are not fully productive during summer breaks or exams.

7 Summary and Future Plans

We succeeded in getting the running software at the end of the Tempus project. This, however, should be regarded as the very first step in a long process.

Every participant of the Tempus project is entitled to get USOS distribution package and a few Oracle licences financed by Tempus. Polish universities have also understood advantage of standardized information system. These are sufficient conditions to start deployment.

USOS delivers main functionality needed for handling student affairs and study programs. Many useful functions are still missing and this is our task for the future. The professional support and maintenance group has now to be formed and take over the software from the development team. The main task of this group will be to deliver help-desk, support software deployment, gather bug reports, debug and maintain software, prepare documentation, and collect new requirements.

Of course we need funding for that task. Further development and support of the system should be one of the obligations of the future consortium of Polish universities.

Students of the Faculty of Mathematics, Informatics, and Mechanics will be engaged in developing new modules. Currently we work on modules for handling students dormitories, financial aid and scholarships, class schedules, diploma supplement (a document established by the European Commission, Council of Europe and UNESCO/CEPES in order to improve the recognition of educational credential). Many faculties ask for the module for tuition fee processing. The central university administration needs access to global interdisciplinary statistics. We also want to deliver more information through the web, e.g. detailed information on degree requirements, individual requirements of student profiles, student progress, personalised class schedules. USOS should expend in the direction of Virtual Students Office, offering integrated set of services.

In particular, one of the main goals for the near future is the design and development of the web-based student admission system. It would constitute the basis for the new admission procedures, which will be introduced with the reform of secondary school graduation excepted in three years. This project has been undertaken by a separate group of academic teachers and students of the faculty. Our faculty was supporting computerisation of the central admission procedure for Warsaw University for a few years so it was relatively easy to define system requirements. The simple prototype is expected in a short time and the fully functional version should be ready for admission for the 2002/03 academic year. The guide of study programs of all Warsaw University faculties will be available on-line, easily browsable. Printable version of the guide will be prepared from the same source of data, delivered as XML documents. A special agreement with one of banks have been worked out. Prospects will transfer registration fees to bank accounts, and information about these bank transfers will be send automatically to the admission system. Basing on this information the system will authorise applications delivered through web forms. The admission software is designed to have a separate database but its structure is consistent with the USOS database thus it will be easy to transfer records of the accepted prospects from one database to the other.

Last but not least, the academic society should not only be delivered a new more sophisticated software but also should be instructed how to use it, how to find needed information, enter data, print reports, etc. The reality is that members of this society — students, academic teachers, and administration staff — vary in their computer literacy and they need support and encouragement to fully benefit from the new opportunities.

This is also a challange for the Faculty of Mathematics, Informatics, and Mechanics, for the following reasons: we know technology, we have experienced instructors, and we have laboratories with computers which can be used for training. Two years ago we started offering courses for academic teachers from all university faculties on using Internet, searching for information, filling up forms, preparing and delivering course details. At the beginning of an academic year new students are instructed how to browse course catalog, register for courses, find teachers' home pages (e.g. to check office hours), get in touch with attendends of the same course, lookup grades and progress within the studies.

The impact of the new university and nation-wide software should be observed at the administration side as well as at the academic side.

On the administration side it should improve management practices, reduce cost of administration processes and promote business rules, e.g. by giving access to various statistical data which could support decision making by educational authorities. New self-service mechanisms available remotly for students and teachers should take care of some administrative tasks, resulting in shorter waiting lines in front of student offices and reduced paperwork.

On the academic side it should allow for introduction of flexible study systems and flow of students between faculties, support national and international mobility, provide coherent services to students. Students can get some advice on determining their profiles, since all necessary information is easily available at their finger tips. Especially those who indent to study more than one subject should benefit from this new approach.

References

[1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesle, 2000.

[2] S. Gylseth, *Using CDM Fast Track, Oracle's DSDM Compliant RAD Approach*, Oracle Corporation, 2000.

[3] Home page of Bureau for Academic Recognition and International Exchange, Warsaw, Poland, `http://www.buwiwm.edu.pl`.

[4] Home page of LADOK consortium, Umea, Sweden, `http://www.ladok.umu.se`.

[5] Home page of USOS, Warsaw, Poland, `http://usos.mimuw.edu.pl`.

[6] Home page of USOSweb, Warsaw, Poland, `http://usosweb.mimuw.edu.pl`.

[7] P. Klint, J.R. Nawrocki, editor. *Proc. Software Engineering Education Symposium SEES'98*, Scientific Publishers OWN, Poznan, 1998.

[8] J. Mincer-Daszkiewicz, *Developing Commodity Software in Academic Environment* (in Polish), III Krajowa Konferencja Inzynierii Oprogramowania — KKIO'2001, Otwock, 2001, pp. 225–236.

[9] I. Sommerville, *Software Engineering*, 6th ed., Addison-Wesley, 2000.