Andras Keri

E-governing in Social Aid

Diplomarbeit

TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

DIPLOMARBEIT

E-GOVERNANCE AND SOCIAL AID

AUSGEFÜHRT AM

INSTITUT FÜR GESTALTUNGS- UND WIRKUNGSFORSCHUNG

DER TECHNISCHEN UNIVERSITÄT WIEN

UNTER DER ANLEITUNG VON

O.UNIV.PROF.I.R. UNIV.DOZ. DIPL.ING.

DR. PETER FLEISSNER

DURCH

ANDRÁS KÉRI

(1020 WIEN, TABORSTRASSE 44/39)

WIEN, MAI 2007

Keywords: E-government, Social aid, Equal opportunities, Open source software

Contact to the Author: keria@tarstud.hu or nessie@metacortex.hu

Table of Contents

I. Introduction

It was a long time ago when this work has started and even longer when the idea to develop this system has came to the mind of the author.

The author has been responsible for four years for distributing social aid at the Technical University of Budapest, Faculty of Electrical Engineering and Informatics. During this time it became obvious that processing meters of paper documents every year without real support from an automated software system was really fatiguing: on one side hearing a family tragedy is always very exhausting, hearing hundreds of them is more than that. On the other hand considerable amount of staff was required every year to handle the demands, to examine them and to attribute a fair social score.

The goal of this work was to find an ideal solution to this problem, by defining the best possible software solution, based on other examples. It was not the goal to wipe out humanity, but to evaluate the demands as fair as possible.

This work has three major parts: first e-governance is presented to have a good overview. This includes definition, prerequisites, typical use and a few important questions to answer with possible answers and as a future example, on-line voting is presented.

Next the principles of social aid are presented: how it works generally in Hungary, at the TU Budapest and the current method to score the demands at the faculty. After this a few design issues are addressed and decisions are given a ground, and the functional specification is also included. As the last chapter of this part the technologies used are presented.

An important part of the work is in the appendices: the technical documentation. It includes description of the database, of the software layer and of the presentation layer, in three different chapters. Completing this, a CD-ROM contains the complete software, the technical documentation and a list of changes (or revisions) of the software.

If you have any questions reading through this work, please do not hesitate to contact the author under the following e-mail address: keria@tarstud.hu.

II. E-governance in general

A. What is e-governance?

First of all, we should try to give a definition of the term *e-governance*. To centre our field of work, we define it the following way: *E-governance is the exchange of information between the state and businesses (G2B) and the state and citizens (G2C), using the Internet.*

Let's make our definition clearer!

The term *exchange of information* stands for supplying information in both directions to the other party. It can be standard government information about everyday matters (regulations, official affairs), standard civic information, like tax declaration and different requests. It can also be on the surface non-standard information, like various individual correspondences, which have to be canalised following a procedure to processing. This means that (almost completely) *structured information* is transmitted on forms, data sheets or on any other carrier.

The term *state* is used in a broader sense, instead of the term *government* because governance does not only imply that the central government ruling the country provides these services, but also independent national and international authorities and local governments give access to this mean of communication. That is the term state will be used to sum up every possible state organs and agencies, big and small, central and distributed (decentralized), country-level and settlement-level and even international level.

Businesses and citizens are the two major target groups of e-government services. The two only differ in their content, not in their nature: both parties have a position as client to these services. We will keep this distinction while enumerating the services, but will speak about them generally as *clients*, if not mentioned otherwise.

While other access methods beside the Internet like phone, fax, SMS etc. exist, we consider them as other platforms to access the same content that is available on-line. The terms *on-line* and *off-line* will be used often in this work; to make it clear *on-line* will always mean "on the Internet, without personal presence of the two parties", while *off-line* will always mean the contrary, "in personal presence of the two concerned parties, the government representative and the client, which can be business or private person".

It has to be made clear at this point that on-line and off-line administration should

not differ in their content.

Having given a detailed definition of e-governance, let's try to define what it is not!

B. What e-governance is not

Another service, offered by governmental agencies to other governmental agencies (also called G2G for Government-To-Government) is not presented in this work as it is more a complex, internal and well protected system that creates a formal interface between state organs without interaction with external parties.

Thus said e-governance *is not* a central government network that allows communication between these organs.

E-governance is also not T*he Software That Can Everything*. It is not single piece of software, but more a combination of hundreds and thousands of different standard and non-standard software, which support its activities.

It is not a *website* where clients can download form sheets to be handled in different offices, nor can they write e-mails to "the government" as there is no "one government above all", but different state organs and agencies.

It is also not *The Solution* to the inefficient administration that the state has been financing forever. It does not require recruiting more working force, but demands the retraining and restructuring of the currently available, while increasing its efficiency and reducing the costs of operating.

Having defined what our work is and is not about, we will give a brief summary of the advantages of e-governance and show a few prerequisites of its implementation.

C. Key: on the Internet

A few characteristics of e-government systems have to be made clear in order to be able to design such a system.

First of all this system has to be available on the Internet. This is not as trivial as it sounds: there have been many trials to create systems where the computer was only used as a word processor instead of a complete system.

Take it literally: a first system called "e-governing" was certainly an electronic typewriter, which helped to get forms completed with clearly readable characters. A more evolved version of this typewriter method is completing electronic forms, printing them and sending them by post. Then the data gets entered to the system and it becomes than information. A more tricky way is to print the information on the sheet in bar codes on the bottom of the page: this makes reading in these pages more efficient, but it is still paper.

Another solution was (and unfortunately still is) a software which exports the data onto a floppy disk (can you still remember this thing?), which had to be sent in to the authority. It might have been a good solution once, but by now it must disappear.

The way to solve these problems is to use the Internet as transmitting medium between the clients and the authorities. It is wide-spread, cost-effective and is getting widely accepted. The traditional way still exists, but more and more people should use it: either for its advantages to traditional ways (quicker, easier, less time-consuming and cheaper) or because of legal obligation.

D. Why is it better?

We would like to claim that e-governance on the whole is better than traditional governance, however we are aware of its drawbacks and will try to give solutions to avoid them.

First of all, clients do not have to go to offices, which saves time. As the American saying goes: time is money and money is time, therefore saved time is potential money. Here is an example to illustrate this: while calculating the returns of an important investment on national level, like building a new subway line in Budapest, Hungary, the time saved by not travelling because the journey is faster is estimated at a very low 600 forints (approximately €2,40) per hour, without taking into account the potential production increase.

A 24/7 opening is also a great advantage: while in the off-line system most people have to take a day or a few hours off work for administrative purposes (especially in Austria), the same thing can be done on-line by using electronic services at any time, especially off-work.

While processing off-line information requires registering data and then taking a decision, doing this on-line the first part of the work can be simply saved: the user that used to fill in forms by hand that later got entered into a complex system now enters directly the information into the system. This method allows automatic checking of the data, following different criteria.

The question of archiving and filing has to be mentioned: storing official documents has always been a problem, but what was always an even bigger problem, was retrieving a given piece of information from the stored data. E-governance entails storage in a database that holds all the necessary information that can be retrieved by the authorized persons. Costs of archiving electronic files compared to paper files is also relatively low, for instance there is no need to build new national archives as it happens once in a while in most countries.

E-governance reduces disparity caused by distance: smaller regions and regions that are further off from administrative centres have thus the same level of access than bigger and closer regions; it diminishes the divide between urban and rural people.

E-governance assures equal opportunities to everyone accessing the services, as the interface can be made accessible to the visually, the physically impaired, to the elderly or to citizens with language difficulties. It can facilitate the access even if a personal presence would be an almost insoluble problem.

Not only has e-governance its advantages in the quality of service, but it is relatively easy to show a few elements which prove its cost-effectiveness. Instead of entering data written on forms, entering it directly the system can save us the time and the potential error of this task. Using remote administration instead of personal presence can save very important front office space and personal costs. Electronic archiving can save lot of back office space, as well as the task to order paper files.

Sure, developing e-governance services cost money and naturally this complex system is not cheap. While accepting real costs involved in the development, we will try to show that these costs do not have to be as heavy as most think. Since most government services are already centralized, only a new interface has to be developed that links the concerned service to the e-governance framework.

E. Prerequisites

As our goal is to discuss e-governance without trying to give a complete solution to each of the problems it arises, we only list up the prerequisites without an in-depth discussion. Each of these questions needs an own discussion which goes far beyond the limits of this work.

The government has to create on-line content that is accessible to all of its citizens, without regard to age, gender, education level, computer literacy and literacy. These parameters have to be taken into account while in the planning phase. Of course a full penetration of e-governance services cannot be attained, as even off-line government services do not reach every citizen.

i. Computer literacy

The lack of computer literacy is the main impediment in the use of e-governance services. It can be fought starting among the young and occasionally among the elderly; it is not easy to teach the use of computers to those who work beside and do not use computers yet, thus such programs should be initiated.

ii. Internet access

The use of e-governance services requires some kind of Internet access, which is measured mostly by the rate people of a certain region use the internet, also called *Internet penetration*. It has to be made clear that this rate is very difficult to obtain, as there no complete census on internet usage exists, only (very) rough estimates come up once in a while based on the number of broadband access and different (so-called) representative surveys.

As it can be seen on the charts of the Internet World Stats homepage[1] as of the data updated on 11th January 2007, only 16,6% of the population has access to the Internet, with a penetration rate of 69% in North America, 54% in Australia and Oceania, 39% in Europe, 16% in South America, 10% in Asia and the Middle East and as low as 4% in Africa. It has to be added, that even if the access rate is as high as 75% like in Sweden,

1 http://www.internetworldstats.com

New Zealand and Portugal, a considerable part of the population does not have access, which means he or she is an outcast.

It is a more specific issue to examine household penetration, which measures the ratio of households that have Internet. Even if the home-use of Internet is mostly a matter of money, if the willingness to manage administrative affairs on-line exists, access provided elsewhere will allow the use of e-governance services.

Businesses in the highly developed part of the world are mostly networked, but being the engine of development, this will be a very exciting question in the developing world.

iii. Technical aspects

There are both hardware and software prerequisites to use the Internet and more specifically e-governance services.

As far as the hardware is concerned, it can be said that a computer bought for around 300€ is powerful enough to access the Internet. We do not want to enter into details on ways to support developing countries with computer hardware, as this would lead us to a very different way from our current goals. Despite this the author wants to express his concerns about the possible difficulties to use the computers provided by the One Laptop Per Child program[2] to use e-governance services by the parents of these children.

Supplying software to people with low budget is not a difficult point. In most countries unlicensed private use of computer software is not persecuted very strongly, thus being illegal; only resellers of illegal software are pursued by authorities. (This is also a major concern when such countries want to join the World Trade Organisation.) As a country develops, the rate of legal software use increases and attains a decent level. It has to be added that for the use of e-governance services most of the time only an operating system is needed, thus the cost of software is decreased.

The author casts his vote for the use of libre software (libre meaning both free as a beer and free as freedom), but this does not really influence the use of computers, only provides an easier way to use legal computer software.

2 http://laptop.media.mit.edu/ [Accessed: 22th March 2007]

In case the above listed technical criteria cannot be fulfilled by somebody, the state has to make an effort in providing a computer with Internet access. Special groups of the society can be chosen to facilitate their computer use by providing them with cheap or free computers for home use, but this must be limited to groups where it will create externalities. This is a policy question but we could imagine groups like state employees (getting to a higher degree of computer literacy that helps them while working), teachers (using computers to teach better the future generations) or students (the future generation that will use computers in most moments of their life).

iv. Access points

An access point is where the user goes on-line to administer. Principally Government2Business use is from the office and Government2Private use is either in or outside the domicile.

All other disadvantaged groups should be able to use public access points all around the country, where they can access the Internet generally and especially e-government services for very low costs or for free. This process has to be supported with means provided by the state, thus creating a network of such access points all over the country.

A telecentre good solution to create access points which offer Internet access and also other services: it "is a public place where people can access computers, the Internet, and other technologies that help them gather information and communicate with others at the same time as they develop digital skills."[3] They can be combined as also planned in Great Britain with traditional government services like post offices, social or agricultural help points or even with a doctor's surgeries, thus telecentres become a real centre for rural life and an important meeting point in larger settlements.

F. Typical use

In this chapter we will try to give a few typical uses of e-governance services. They are not structurally different if provided to private persons or businesses, but can be grouped for a better understanding into these two categories: Government2Private and Government2Business (G2B).

3 Definition from www.telecentre.org

i. Government2Private

a) Tax declaration

Computer programs can make it much easier to understand the labyrinth of rows and columns, they can provide assistance to complete the forms interactively and can correct smaller – and often the most common – mistakes. Moreover the completed form can be handed in electronically instead of postal way thus saving the work entering the data.

b) Domicile registration

This registration process is usually a sheet of paper handled in at the office after queuing. Electronic registration can accelerate this process and save time for clients.

c) Car registration

If a car is bought or sold, in most cases the car does not need to be controlled, only the information concerning the owner is updated. In this case both the buyer and the seller should enter the data in the database and the two sides should match. If it does, no manual work is required, the new vehicle license can be automagically delivered.

d) Social security

Listing of different treatments, detailed information about delivered services and expected payments can be recalled by using on-line services without much effort. This guarantees transparency in this matter.

e) School or university enrolment

Creating an on-line service to enrol pupils and students relieves educational institution from administrative tasks, as their main field of activity is education and not administration.

f) Demands, declarations

There are pretty many declarations and demands that have to be handled in regularly where personal presence of the demander is not required. These can be for example a demand for authorization of marriage, demands for family aid, appointment reservations, local tax

declaration etc.

g) Building permission

Handling in plans required to obtain a building permission can also be simplified: instead of printing all the huge sheets, plans can be handled in in electronic format, digitally signed by the architect responsible for the plans. It makes administration of plans easier, archives can then be better searched and construction sites can be more easily controlled while the plans are on the computer of the controller.

ii. Government2Business

a) Regular declarations (tax declaration, social security, statistical authority)
Saving the basic data (e.g. data of employees) on the server. Up-to-date database available to analysts without manual data entry

b) Permits

Permits for different activities: sheet of paper with stamp and a signature is the same as a digitally signed declaration. Easier administration and archiving.

c) Registry court declarations (Firmenbuch)

As it is expansive, companies do not update their records as often as they should, which results in an imprecise database. With less paperwork and also less costs for every party, the database would be kept up-to-date much easier.

d) Procurements: tender announcements and offer handing-in on-line, which makes it more transparent and easier to control.

G. Similar use in business

In order to show that e-governance has nothing extraordinary compared to the business use of information technology, two examples of important on-line services will be mentioned in this chapter. They are Business2Business or Business2Private solutions and they are not typical e-commerce applications as – similarly to the state – they are not selling anything. We took two very common and widely accepted examples of every day life: electronic banking and electronic study administration in colleges and universities.

i. E-banking

E-banking (or as often described for private use: home banking) is not the technology of the second millennium but more the reality of the 90s: already at this time did banks allow major clients to have a PC connected through a modem to access account information and to transfer money. It was a very well functioning system as it is still in use in many companies as they did not want to change to the more modern on-line e-banking systems, as the "Internet is evil".

What are these systems able to do?

The answer is trivial to most of us, who use the services on a daily basis. The basic functions are the same: one can access account balance, account statements (both incoming and outgoing transfers) and start new transfers. A couple of other things were added to create an extended functionality: transfer templates (for frequent partners), regular transfers, direct debit, saving accounts, multiple accounts, investment management, savings management, and lastly electronic invoices management. To sum up it can be said that e-banking can everything, except for personal counselling and cash withdrawal, available only at branches.

Why this area became such a success?

Banks have very important costs bound to office space, working force and cash management. It can be understood without much explanation that a centralized call centre or administration centre is much more efficient than the employees sitting in a branch office playing solitaire during an important part of the day, which leads to save on office space and working force. It is also trivial that banks prefer transfers between accounts than cash transfer as it is a burden to handle with cash: buy and repair money dispenser and than regularly refill them, count money again and again, check for forgery, secure the transport etc. That is why banks try to persuade each client to use electronic services to such a big extent as possible. The development of the financial market, the willingness of people to use banks made the bank system much larger and denser.

Which might be a more import question in our discussion is *why people like e-banking?* First of all, as a consequence of the previous example, banks offer important rebate on electronic payments. It can go from 1% off of everything bought with a certain credit

card to making electronic transfers much cheaper than "regular" – paper-based – ones, or even free. Beside cost savings, home banking users might prefer to have time to take decisions in money matters, to have a constant overview over their account and they probably do not want to talk to other people about money matters.

E-banking system shall be seen as probably the most successful e-governance-like application where effectiveness from both sides and need for privacy from the clients' side met and created in my opinion the best B2C framework ever. It increased effectiveness on both sides, where this effectiveness is much money worth. Its ease of use is not a problem, a regular e-banking user would never like to have to old system back and that is the most important metric on the acceptance of a new technology.

ii. Study administration at colleges and universities

As the number of students enrolled at Austrian and European universities has been increasing over the past decade, universities could not cope with the increasing administrative tasks. In the years where there were 50 students in a "class" one person could be responsible for their study matters. With hundreds of students taking courses at their own rhythm, students taking different specialisation courses, students going abroad or coming from abroad, administrative tasks of universities have largely increased. As the job of universities is not to administer but to teach and to do research, some kind of solution had to be found.

This was the main reason for introducing study administration software at colleges and universities. It helped to overcome these administration problems and changed checked paper for taking courses to a well-administered electronic system and created an electronic register of marks instead of different paper-based registers.

The solution that came forward is quite different from university to university. In most places special software is used that has been developed for that institution, its capabilities are totally based on local needs. Another solution is to create country-wide accepted software that can be parametrised for every need. The latter is much more powerful, the question is to know if we need that power. For sure, universities investing on these systems got a very good return while improving services they offer.

Principally this solution broke personal contact between students and university administration, the question is if it is worth the price: do students have anything better to do, as to go to the university and get their name on a sheet of paper? Does university administration have better to do as to write names and marks on different sheets of paper?

The answer is clear and the balance is also obvious: despite the missing face-to-face contact, university administration could cope with a much larger number of students without increasing the administration personal by the same amount, to the contentment of both students and teachers.

H. Questions to answer

In this chapter a few general questions will be answered. The first and most detailed one is about authentication.

i. Authentication

Some kind of authentication is required, to be able to attach a virtual identity – that is an unknown person with an IP address in front of an unknown computer – to a real-world person. User name and password, ID and fingerprint, iris and retina scans and voice recognition, it has to be controlled and verified. While picking one of the possibilities, the threat and the possible losses have to be compared with the costs of the method. Not only the software is to be taken into account, but every access point has to be created: how many people would use electronic tax declaration if an iris scanner at home would be required?

A certain registration process is required to create the digital identities of citizens and companies, as there is no official register in any country containing a unique natural identifier (like fingerprint or iris image) of every citizens that could be used as identifier, thus artificial identifiers must be created or natural identifiers must be recorded.

Another problem arises with companies and other legal entities as they cannot have a digital personality as they do not even have a natural one. Legal entities are represented by natural persons as CEO, chairperson, two of the board members together etc. thus it can

be connected to natural persons. That leads us to the conclusion that legal entities are represented by one or more natural persons who are allowed to represent the company.

A widely accepted classification of authorization methods are the so-called authorization factors, where the most common three are the following: **something you know** (PIN, password, codes), **something you have** (credit or other identification card, security token, mobile phone, one-time password generator etc.) or **something you are** (fingerprint, retinal scan, voice recognition). We will follow this classification but will omit less standard methods and only present those that are common or might become common in the close future.

The standard method for user authentication is a **user name and a password**, which is a one-factor authentication (what the user knows). The advantage of this method is that neither special hardware nor special software is needed to use it and a secure – encrypted – connection between client and server guarantees maximal security of the transmission. Citizens get a password any time they want to register or they might get the passwords delivered by mail. The problems with its use are multiple: a password can be forgotten, or what most do, they scribe it on a sheet of paper, thus it can be stolen, or the password can be stolen on phishing websites. Some use more than one password for authentication like user ID, account number and password, but as these data are so complicated that every user certainly writes them down, it does not improve security, only makes the login process more complicated.

Banks usually extend the previous method with **one-time transaction codes (TANs)** for transfer authorization. A list of such codes is given to the client once in a while and that way a physical object needed to make transfers is created: the list of codes. This is much better than the password-only method, but as most user stores the list of codes together with password and user ID, it does not really increase security.

Another possibility is when banks send out such a code by SMS or e-mail and the user has to enter it on the site to approve transaction. A small problem with this solution is that e-mails are not secure and sending SMS has a certain cost, but this seems to resolve the previous problem, thus creating a **real two-factor authentication**.

As a further solution installed digital certificates are also available: the user becomes his/her **digital certificate on a floppy disk** (!) or a memory stick and has to install it in the computers certificate repository. When needed the certificate is used automatically to grant rights. The problem of this solution is that the certificate may get compromised if the computer gets compromised by physical access or by software attacks (trojan horse), thus it is better not to store the certificate on the computer.

These were the software only devices; let's have a look at those where some hardware is required. When using external hardware it has to be taken into account that this hardware has to be bought on one side and that his hardware is not available everywhere, the device has to be always carried with.

A solution not storing the certificate constantly is used in Austria for electronic signature of tax declarations: the user has to obtain a **digital certificate, stored on a smart card**: either a general-purpose commercial digital certificate (available from major banks for a yearly fee, installed on standard debit cards) or a free certificate installed on the "**e-card**" (electronic health insurance card), available for free, but only usable for e-government authentication. This physical object, combined with a password is needed to authenticate. Such an object is easier to secure than a mental object like a password and if it gets lost, the user notices it. This is a good solution for the moment, but it has to be added that if someone is not willing to buy a certified digital signature at a bank, he has to used his e-card, while the registration process is quite lengthy. The certificate on a chip card is used to verify the identity of the person using the computer and also used to produce a tamper-proof document, handled in to the authority in question.

Another possibility of the kind would be the use of **RFID** (radio-frequency identification) **tags**, installed on every new passport in the EU. This technology is quite wide-spread as it has been used for years: this chip contains the basic information about the person (name, gender, date of birth), readable to all with a special device. The inconvenience is that if someone can read the data contained on the tag, it can be also copied.

Another solution is to use **fingerprint identification**: new-generation fingerprint scanners are almost 100% reliable and are tamper-proof. To make the situation better, many IBM Thinkpad (now Lenovo Thinkpad) and Sony Vaio laptops have an integrated

fingerprint scanner, which gives the impression that the technology is quite mature and well-tested. However it has an important drawback: people – the citizens – regard the fingerprint register as part of the criminal register; no one would agree to give on purpose a fingerprint to the database.

There exist other possibilities in the authentication factor "what you are" beside fingerprints, namely iris and retina scan and facial recognition. All possibilities use a to our knowledge unique personal identifier, where iris and retina recognition is used only in high-security premises, facial recognition despite a few years long trial period has not proved to be very efficient. These are technologies of the future or they might not even get realised.

To sum up the above technologies in my opinion the following ways are actually available to e-governance authentication:

- simple user name and password pair
- user name, password with one-time transaction code (SMS/e-mail): real two-factor authentication, but changing contact data (phone number, e-mail address) must be administered
- smart card with digital certificate, PIN: reader is required, provides extra security through digitally signed documents
- fingerprint scan: reader is required, not available to sign documents

They have to be reviewed carefully and the best method has to be chosen for every purpose. If a general framework is created for authentication, this process should be even longer as it might have important financial effects. The author thinks that using smart cards is the best solution (especially with smart card readers integrated into computers), because beside its high-level of security (two authentication factors), it can be used to digitally sign documents and it has the capacity to be extended for other services like on-line payments.

ii. Lack of personal contacts

After this chapter about the authentication process a few other problems arisen while introducing e-governance have to be discussed.

First among them is the impersonal treatment of clients: not a person any more, only some bits of information. But what could be more effective and transparent than such an impersonal and bureaucratic way of dealing with affairs? What could be more similar to Max Weber's notion of neutral bureaucratic organisation[4]?

E-governance is nothing more than series of forms to be completed, which helps to automate the processing.

> As white-collar workers, the "off-line database administrators" got superfluous as computers with huge databases became available, the same process is happening now with more and more people doing the work of typing in the information free-willingly instead of the paid employees. According to Parkinson's Law "***work expands so as to fill the time available for its completion***", it will not cause drastic increase in unemployment, thus reassuring politicians of all continents.
>
> *C. Northcote Parkinson: Parkinson's Law, The Economist, November 1955*

The loss of personal contacts, the disappearance of the possibility to ask question is also very fearful for many. This can be fought with the use of on-line information databases, lists of frequently asked questions and context-sensitive forms accompanied with step-by-step instructions answering all possible questions. If this does not help, a free or local rate number can be called where most questions can be answered. And the move to e-governance does not mean that all offices will be closed, only working force will be distributed more to do active work, help clients instead of typing in data.

iii. Storage, archiving

E-governance and electronic commerce restructures the need for archiving: what used to be thousands of running meters of documents in a dark, wet cellar of an old building is now a few gigabytes of data on a shiny computer hard disk.

This can save uncountable office space and makes data retrieval much easier and quicker, but it had to be secured for the eternity: paper could be kept – if no special disaster occurred – for 400-500 years that has to be the minimum for digital data, including all modification of course. Beside electronic data traditional documents handled

4Weber, M.; *Wirtschaft und Gesellschaft* Tübingen 1922 (1. Teil, Kapitel III., 2. Die legale Herrschaft mit bureaukratischem Verwaltungsstab) [Full available available at http://www.textlog.de/weber_wirtschaft.html, accessed: 22th March 2007]

in in paper form has to be kept according to local regulation, which may require decades of secure storage.

It is quite evident that even if only a small part of data comes in electronic formats, the rest of data should be digitalised: scanning, tagging and character recognition are very important parts of the migration to electronic administration.

Another question is the digitalisation of already existing paper data: how much effort and money is it worth to get old papers and type every single line into a computer database or at least scan them and tag them with appropriate keywords?

Often the answer is obvious: as much as needed. There exist databases which are partly available only off-line and they are needed for everyday life. Such examples are land registers and company registers, which are often available partially only on paper. These data – the current data, without changes – have to get digitalised (if not already done) as soon as possible for reasons of legal security. Other data – such as historic birth and death registers – should get some funding to get the data entered into a database, and others again – as general data from national archives – should stay on paper for the eternity.

iv. Data protection

Data protection is an important issue, as a massive amount of sensible personal data can be easily stolen from a digital database. There exist methods to control data access already used in tax authority and police data centres: every access is logged and then analysed and unusual access is automatically subject to investigation. Hopefully this automatic investigation and a very strict internal regulation can control unauthorized data access.

I. Common framework or local solutions?

It has to be cleared if a common central framework or local solutions should prevail, each has its advantages.

With central solutions the user gets a common look, a common user database, but as different authorities do not share their databases, interfaces have to be created to ensure

communication. Creating such a system is very demanding in project management, thus might lead to a system that will never be finished.

Contrary to this local solutions can escape all these drawbacks: no need to create interfaces, no need to create a never-ending project, just a small task like a website for a municipality and it is ready. On the other hand the problem arises here with missing competence, no way to get secure authentication.

In the authors view the ideal way to overcome the problems is to mix these two: the central government should be responsible for creating its own web site with plenty of links to other e-governing websites and besides it has to create a **framework for secure authentication** as defined centrally and provide basic information and links to authorized users. With some basic supervision of services developed by different authorities, this can provide a secure and light-weight method to create more and more e-government services.

J. An example for the future – On-line voting

On-line voting is probably the most interesting and most challenging application in the field of e-governance. It must not only be secure for identification, but it must provide a reliability of 100%, store the individual data completely securely and in an untraceable way. One must trust this voting machine, as there is no possibility to recount the votes as it is possible in paper-based systems.

It has to be added that existing implementation of voting systems have been compromised many times, just recall the Florida scandal in year 2000 on recounting voting cards. Another interesting story comes from the Netherlands and Ireland where the electronic voting machine has been attacked and in less than a month of time the researcher could play chess with the computer or could tell from a distance of up to 10 meters if someone voted for the Greens (the only party with the *Umlauts*) and this was only the first step before modifying the results of the elections. A detailed report on *Nedap* voting computers is available on the web site of a civic rights organisation.[5]

5 Report: www.wijvertrouwenstemcomputersniet.nl/images/9/91/Es3b-en.pdf, full web site available under: www.wijvertrouwenstemcomputersniet.nl/Nedap-en [Accessed: 14th February 2007]

The key to a successful electronic voting system is the so-called blind signing, which can be seen as a parallel to signing the closure of an envelope to ensure its integrity by a person who does not read the content of the envelope. The technology consists of a double usage of public key cryptography: a message is first blinded, that is encrypted and this blinded message is given to the signer party, who attest the authenticity of the person who created the message.

If one wants to create such a system the voter has to encrypt his or her vote with the public key of the voting committee (of course together with some random noise in order not to create identical encrypted files). The secret key from which this public key has been generated has to remain secret until the end of the elections, in order not to leak any preliminary results before the official counting.

Finally the encrypted vote has to be signed by the voter to verify his or her identity, but this signature has to be removed before counting to ensure secrecy. It is a very important task to ensure that the vote gets counted correctly and the importance of the transparency of such a system is above all priorities. Such a pilot project has been conducted for example by an international consortium and the complete project documentation is available under www.eucybervote.org and in various publications[6,7,8].

6 www.eucybervote.org/TUE-WP2-D6V1v1.0.pdf [Accessed: 14th February 2007]

7 Shoenmakers, B.: Fully Auditable Electronic Secret-Ballot Elections www.eucybervote.org/xootic2000.pdf [Accessed: 14th February 2007]

8 Canard, S.; Gaud, M.; Traore, J.: Electronic Voting Process Fair Blind Signature (Patent application: http://v3.espacenet.com/textdoc?CY=ep&LG=EN&F=4&IDX=EP1721408&DB=EPODOC) [Accessed: 14th February 2007]

III. Social aid at the TU Budapest

A. Introduction

In the next chapters a particular e-government system will be discussed, which has been created in the framework of this work. The system was designed to demonstrate how the steady work of one single student can lead to a functioning example, which is the system helping to distribute social aid at the Technical University of Budapest, Faculty for Electrical and Computer Engineering.

First the legal basics are presented: what is social aid in Hungarian higher education, what are the legal requirements. In a second part the distribution mechanism will be described more in detail, together with the description of a new system that has been introduced in order to give more meaningful information on the social situation of university students.

B. Legal basis

The legal framework of social aid is governed by different acts at four different levels: the Higher Education Act lays the principles of financial aid, while a government act defines its distribution. More precise regulations are contained in the university rules, while the actual announcement is created by the student commission of the faculty.

Lately the trend goes in the direction of repealing the fourth level and thus creating a unified announcement and unified application procedures for the whole Technical University of Budapest.

i. The Higher Education Act is the most important basis for our point: it allocates important funds for scholarships, lecture note subsidy, social aid and residence allowance, which is quite socialistic in the context of higher education. It reinforces the European view of higher education (being a public service) against the American view, where it is considered a private investment. As this sum became an important point in election campaigns, it has started to increase from around €300 p.a. in 2002 to €500 in 2007, to the great joy of students.

A recently changed[9] government act describes precisely the distribution principles. Beside many other points including detailed regulation about who is entitled to state financing and who is not, it defines the way this sum should be distributed: this used to be 75% for scholarships and 25% for other goals including social aid and it became from September 2008 40% for social aid and 60% for all other goals. An important shift in the focal point is to be underlined: social aid gained importance not only in percentage but by declaring its importance above scholarships. A few other points have been changed to unify public social efforts: those categorized by the state as "in need" (bad social situation, underprivileged, minorities) automatically get more money.

ii. An additional point to be mentioned is the student loan system with a monthly sum of up to 160 euros, being a state guaranteed loan with relatively low interest rate. The instalments are 6% of the monthly loan and they fall off while on maternity leave and the other debtors stand for if someone gets retired. This system gets almost no state funding (except for a basic subvention for administration) and is fully self-supporting.

iii. To make this system a bit more complicated the very special rules have to be created at university level or even at faculty level. They do not change the system, but define deadlines, ways to apply, application forms etc. A very important point here is that university rules at the TU Budapest give very wide rights to the students' self-governing body to distribute these funds; the dean of each faculty has only a legal supervision right (a veto right in case of unlawfulness).

C. Principles

After this brief description of the framework of social aid, the distribution mechanism used at the Technical University of Budapest, Faculty for Electrical and Computer Engineering is presented in details, as the goal of this work was to create a computer assisted solution to award social aid.

9 Enacted on 24th February 2007.

i. The old method

Of course distributing social aid has nothing new: it has always existed, or at least in the last 50 years. Until 1990 the distribution was not that complicated: the appliers were asked to present a single document for each family member to attest the revenue and the awarded social aid was calculated depending on the income per family member, plus a few political factors (students of proletarian descent got more, of intellectual descent got less).

The changes of the 90s implied huge changes about how people earned money and how much money they earned. This, together with the very important expansion of the higher education made distributing social aid a much more complex task. The possibilities of making money became much broader (both in legal and grey or black sectors of the economy), and the changes made people more inclined to lie when it went about their social situation. In order to avoid abuse, more and more official documents have been asked.

The goal was always to determine the income per person, but due to differences between loan and income this became more and more difficult. Official declarations from tax office could help the work to great extent by giving one single certificate, but it does not contain tax-free income, like pension, local social aid, unemployment benefit and family allowance. Beside this, black market income or tax fraud is also a problem: how could a physician earn only the minimal wage of around 250€?

To overcome the problem the evaluators used fuzzy methods: with years of routine an approximate income based on certified income, profession and other factors could be estimated, more or less precisely. Also it had to be taken into account that a few students lived in a very special situation that could not be justified by the official declarations presented. In these cases the evaluators turned again to these fuzzy methods and tried to evaluate the situation in comparison with the other applicants.

ii. Reasons for the new method

Of course this method lacked the very important principle of transparency. Beside most people that were satisfied with the result, a few very unsatisfied turned up every

time. To most of them the results could be explained in detail, but a few remained unhappy with the social aid they got awarded.

The transparency was also required by superior organs like by the dean of the faculty as a few students unhappy with the results complained. He could understand the points of the evaluators, but he gave his mind that more transparency in the method would improve it.

It was in every case very important to be able to justify the decisions, both to the demander and to the evaluator him- or herself. The lack of such a justification led to an important psychic burden of the evaluators. It was important to give points of references which could lead to an easier decision, while taking into account special conditions.

iii. Points, a key to transparency

To overcome these problems a brand new system has been introduced: instead of calculating revenue per family member a table with different categories has been designed, where each category had a certain value.

One of these categories, of course, depends on the income per family member and it represents an important part of the total score, but while calculating this, no other factors are taken into account and above a certain limit a negative score is attributed.

Categories have been created for parameters that can be objectively measured: orphans, brothers and sisters still studying, unemployed, disabled or chronic sick family member, distance from university. The total score in these categories now represent a part around 50% of the total score.

An extra category remained, thou: the one with freely attributable scores. These may not exceed 15% of the total score and are generally not attributed, but give the evaluator a possibility to deal with special situations.

The three categories are then added up and depending on the total score a monthly sum (which can also be nothing) is attributed.

D. Changes with the new system

Of course plenty of problems arose during the implementation of this new system.

Though attention was paid in creating the categories and the relative scores, a regular review of the scores was needed in each semester since. The scoring table has been reviewed every semester and also modified to some extent, in order to better suit the changes in the income structure. In five semesters one new category has been created and four has been deleted in order to simplify the process.

As many factors had to be taken into account while calculating the points, the processing of the demands became much more complicated, it took much more time then with the former method. Sure, transparency (as democracy) has a price to pay, but taking into account that all the evaluators worked voluntarily, the price was possibly a bit too high.

But beside that some help was needed to make the work easier, the introduction of scoring led the author to an important conclusion: this new system could be automated, where automatism means a computer working instead of a human. Of course not every task can be simplified, but calculating the scores can be done by a computer, while humans can concentrate on tasks that require personal attention like the evaluation of special cases.

E. E-social aid

For now the problems are quite well encircled. The goal was to create a method that helps to decide about the demands by making decisions as far as possible automatic. This led to a more transparent system where the applicants could calculate their score. The next task was to give this possibility to the evaluators as well.

The main points were defined: with the correct data fed into the system, it should be able to get the scores right and then give the possibility to the evaluators to reconsider the special cases.

But then the question came: how to get this data fed into the system? Entering this data would take generally more time than to get the results calculated on the traditional way, so how could such a system help?

The answer seems then again clear: the users of the system – the applicants – should be asked to enter the data instead of writing them down on a form. Then the presented

documents has to be verified by the evaluators, but as only GO – NO GO possibilities exist, this does not take very long and the mental and psychical effort of the evaluators got reduced to the necessary minimum. In fact with this extension it became possible to bind in external people (even other students not concerned in the matter with some minimal salary) to make the evaluation phase as quick as possible.

i. Extension: Dormitory administration

There is also a possibility to extend this process with the dormitory administration. Dormitories belong to the university and as they represent a scarce resource, some mechanism to attribute dormitory placement is also required. As they are let out to students for a minimal fee, a triple system has been introduced a couple of years ago to ensure its proper distribution.

One third of the score comes from the social situation, as described above. One third comes from school results where also negative scores can be achieved. The third part comes from work for the community of the students of the dormitory and of the faculty which score is attributed by a hierarchy of elected persons, representing the whole community.

The social situation is already described in previous chapters, information can be found there. The marks can be accessed at the study administration office so the study score can be calculated based on this. The scores for the work for the community represent one single data that is also available in a simple data sheet.

Thus extending the system to handle this task is not impossible, though not easy, due to various requirements of the students to get a place at a given dormitory with the given room-mates, especially when not every room-mate gets a dormitory placement.

To summarize this extension could be created in a later version, but while designing the software, the possibility of this later extension should be taken into account.

IV. Design principles

A. On the theory of brick-laying

Software design is just like theoretical brick-laying: if learned at university level, although theoretical bases have been thoroughly taught (static, chemistry of material, technical drawing etc.), this does not mean that the student has ever seen a brick or a trowel, or have used them!

The author of this work passed all the necessary exams with no below-average marks to become an engineer including different lectures and laboratories in both imperative and declarative programming and, of course, software design. He can draw fancy UML diagrams and is conscious of the mechanism of inheritance and operator overloading, for example, but has never really used them before this exercise in practice.

He has heard about plenty of different technologies, has a rough overview of possibilities, but until one has not tried a bunch of them, these are no more than theoretical knowledge on brick-laying.

i. Scalability

The system to be designed has to be scalable to provide services to the estimated 1500 users, where they usually want to use it in the days before the deadline. Despite this no extra-scalable systems need to be created, though it is important to separate different server processes as far as possible.

As we are designing a system where the data is stored in a relational database management system (RDBMS) and the content and the business intelligence is serviced by a special server process, these two services could be easily separated to two different machines.

If the database to be used is getting bigger and performance problems arise, the capacity of the database management system to use a distributed database could be exploited, though the author does not see its necessity at this point of the time.

As no data is stored on the content servicing server, they could be replicated also easily where load-balancer software could distribute requests among them. To do so, it is an important point to see that the served content has a static and a dynamic part. Static content are images, documents, static pages, while dynamic content represent the business intelligence behind the system that is "the software". Once again, static and dynamic content can be served from different servers, again helping to distribute the load. This technology is very often used for computers serving busy websites and the software and procedures to use are available freely and well documented, for example on Wikipedia website[10].

In most cases it is very difficult to calculate the load a server can serve; only rough comparisons have been done to ensure the server can handle it. Of course sometimes failures may happen when most users try to access the server at the same time and this will make it inaccessible. We are not creating a real-time system and although we should avoid failures, if they happen, we can handle them. If the required capacity is not present, with very little work the resources can be multiplied by setting up new servers to distribute the tasks: creating a separated database server and splitting static and dynamic content will not be a problem.

ii. The database engine

It was quite obvious to choose a database back-end already at the beginning. After some calculations we arrived at the result, that our software does not have *any special needs* at all. Even calculating with twice as much applicants as realistic (3000), with twice as many family members (an average of 7 instead of around 3.5), the total number of records for every data we collected (family members, dormitory administration, personal details), plus twice (!) this amount estimated to be created during the evaluation we got to around 90,000 records which is simply ridiculously few for nowadays database management systems.

That is why it was not really important to get the most scalable database engine. To make the decision easier, the author chose the one, he already had some experience with,

10 http://meta.wikimedia.org/wiki/Cache_strategy

which was MySQL. The other real possibility would have been PostgreSQL, both are available free of charge.

Later on during a re-implementation phase the author opted for the most professional solution that exists: Oracle 10g. At first the free (as free beer) Express Edition has been used, later on the more complete Enterprise Edition has been used. For the latter the cost factor was void, as the University got unlimited Oracle license from the company, for university-internal use.

However it was a goal during the implementation to allow easy migration between databases on the software side. At first a separate module was created to manage database connections and database operations. Later on, as described in the following chapter about persistent connections, special software called persistence manager has been used to make database connections completely vendor-independent and much more efficient. More about this topic can be found in the referenced chapter.

iii. Backup and data security

It is an important point to ensure that the data is kept securely. This issue has two parts: one is about backing up data regularly that is ensuring minimal losses because of a server failure or a destructive attack and the other is about setting different access rights and securing data from unauthorized access.

As there is no data contained in the software only the database has to be backed up regularly, that means – taking into consideration that the applicants are mostly computer scientists – daily or even more frequent backups are required. To do these the included mechanisms of the database management systems can be used in form of automated backups to CD or to another computer on the network. It is also important to ensure that the backup data is handled with care and no unauthorized person can access it. It is also important to test if data can be restored from such a backup: there is nothing worse than false security.

Despite the trustworthiness of the evaluators it has to be ensured that only the application in the hands of the evaluator is being accessed. It is also clear that if someone has the rights only to register demands, he or she should not have the right to view all

demands at a glance. Generally evaluators may not see a list of all results, except for the leader of the process and his or her deputy.

To conclude: the database has to be backed up regularly using an automatic procedure and access rights have to be managed carefully.

iv. Authentication and session handling

In the design phase the authentication was an exciting question.

As already discussed in a previous chapter, a binding between a virtual and a real identity had to be created. As we had to manage an important number of people (around 2000) compared to our resources (maximum 10 people), it was hardly possible to get all these people come personally to register in this system.

First we had the idea to use the RFID card that every student has to access computer labs, but there was a point we could not solve: we wanted to have students access the system from everywhere, not only from computer labs and students in need could certainly not afford an RFID-reader.

Every student has also got a contact-only card, the student's ID, which is a regular chip card. But even the university could not afford buying readers, as the company owning the rights to sell the readers charged an incredible high amount of money for it, so this possible solution also failed.

As at the university there exist also a study administration system like the system TUWIS at Technical University of Vienna, we wanted to have an interface to the database to check the credentials or have this central system check them. This latter is realized at the TU Vienna for example, where for authentication of students departments can request a website of this central server that returns basic data about the student if the identity check was successful and this way no authentication data is stored by the department's website, which increase security. The problem that came forth was that the company, developing the study administration system, was not willing to create this interface and direct database access was not a good solution: no-one would commit his or her password to the trust of a system operated by students.

Finally the design opted for a simple but efficient solution: as every student has an e-mail address provided by the university and upon request, they can set up a different one and give in the details in this study administration system, it was decided, that this list of email addresses will be used for authentication. The list gets exported from the study administration system and is then read in to our system. When a user wants to register, the student identification code (*Matrikelnummer*) is given in and a random generated password is sent to the user's e-mail addresses (to all of them). Upon retried registration, the password is sent again to the user. The user may change the password and the e-mail address and these data will be stored in the database and will be kept for the next semester.

Of course, if the developer of the study administration system decides to create an interface that allows authentication, the whole authentication becomes simpler, but the author doubts that this will be realised soon.

B. Functional specification of the user side

To get the idea about how the process of handing in the demands, we took a look at the already existing forms (which have been designed – accidentally of course – by the author of this work). Forms are in general guidelines of processes and a well-designed form can help very much processing forms. Now the task was to make this processing really automatic!

There were the two forms that have been examined: one for social aid and another one for dormitory application. Both had a common part which was about personal data. It was decided to keep the electronic forms quite similar to the paper-based ones as there is no reason to change them radically.

After logging in the user interface was imagined to contain the following points:

- View information about application process (available without login)
- Frequently Asked Questions (FAQ, available without login)
- Edit personal information
- Edit social demand

- Edit dormitory demand
- List certificates and statements
- Feedback
- Print and hand in demand
- Log out

The *Change password* point was omitted as it was not considered crucial for a system that a user will use twice a semester and getting the password resent is really trivial.

i. General information, FAQ

General information have to be given to applicants and not only the announcement stuffed with legal terms, but also answers to the most frequently asked questions about deadlines, where the statements can be obtained and what to do in certain special cases that are in reality not so special.

If a good list of such questions is compiled, people can be directed to check the answers there, such saving human resource.

ii. Feedback

The feedback form is only available to logged-in users. This form takes over the role of e-mails of the current systems and helps tracing back the problems. In every case the student identification number is transferred (very often it is almost impossible to answer a question without knowing this) and directs the question to the person in charge by e-mail.

iii. Personal information

Editing the personal information is quite a trivial task. Mobile phone number, postal address and e-mail address are exported from the study administration system and then imported here. The user can change these data, if required and these data will then be stored in the database. Basic syntax checking is necessary, but one can trust the users to give in usable information when money is at stake.

iv. Social aid

This is the most complex and also the key part of the whole process. It has to be ensured that the user interface is as obvious as possible and numeric values get controlled for validity to make automatic evaluation possible.

First of all the applicant has to decide if he or she really wants to apply for social score. If not, the following is ignored.

A few pieces of information about the applicant are required: personal income, different scholarships, student loan. As a principle these incomes are not included at all in the family's total income, but help to describe the social situation as for example in case of self-supporting students. A detailed explanation of special conditions should included in the appropriate field,

Other family members have to be added one-by-one, name, profession and different financial data have to completed, activity type must be selected from a drop-down list. Again an explanation can be included in case of non-evident conditions.

It is important to mention that the existing data can be partly copied from the existing database to the new semester. Not everything, of course, as social situation changes from semester to semester, bur for example family members usually remain the same and this can accelerate data entry and the verification of statements.

v. Dormitory application

For the application for dormitory placement it has been decided to create a form where the applicant can enter his or her study average (before getting it officially stated by the study administration), the preferences for dormitory buildings and for room-mates. In the mean time extra functional requests came to the list: as the main dormitory building is going to be completely restored from April, the applicants have to give their preferences on dormitory building and room-mates twice: once for the period February–April and once for the period April–June. Of course, such last minute requests from the client's side are not unrealistic in real life, but they mean most of the time to unadvised changes in design, obscure complexity and important delay in delivery.

Checking room-mates was included as extra feature during the development: instead of letting the evaluators to check if the given room-mate really exist, it has been decided that the applicants have to give in the study administration ID of their future room-mates which is than automatically checked against the database and only valid IDs are accepted.

vi. List of certificates and statements

Most problems came always from the fact that demands were incomplete. If one of the main certificates is missing, no social score can be attributed. To avoid this, a list of necessary certificates should be compiled, based on the data entered by the applicant. This would avoid most cases with incomplete certificates, the resulting complaints and the bad feeling about people not getting social aid, despite their real need.

vii. Handing in demands

As the authentication mechanism used in this system is not sufficient to process the completed demand as a legally binding document (which would require an advanced digital signature with a time stamp), it is necessary to hand in the documents physically. So the applicant is required to print out the complete application form and sign it. The required certificates should also be handed in at this time. In case of dormitory application it would be just a simple sheet of paper with a signature and that is it.

To make processing easier and more tolerant to typing errors, a bar code is printed on the cover sheet of the application form, thus ensuring a quick read-in later. Bar code scanners are available at low price in the commerce (under 50€) and the reader behaves exactly like a keyboard, it can be even set up to "hit" the enter key after reading the code.

C. Functional specification of the administrator side

i. Registering demands

Registering the demands is not a complicated procedure, though important. This is the point when the applicants cannot modify any more the data of the application.

First of all, every single sheet of paper concerning the demand has to be stapled together, in order not to loose or misplace any of them. It is recommended to order the

certificates the way they are requested in the next step, which makes evaluation much easier.

Next the bar-code is read with the reader and the complete application is put into a folder.

It is important to keep precisely in evidence the applications, in order not to loose them, which is why it is recommended to register the incoming applications regularly (on a daily basis or even more often).

ii. Checking statements and certificates

A procedure had to be created to check the statements the applicants presented to support their demands. Taking into account that the about 1000 demands contained such a social part this task was crucial. Just an example: a model family with two parents, the applicant and a brother/sister still going to school has to present 6 different sheets of paper, and this was only the most simple case: beside declarations for the parents from the employer *and* the tax agency, a statement of the residents of the household and the statement from the brother or sister attending school. That totals to around 20 meters of paper and as checking these statements is the most demanding task of all, its efficiency has to be as high as possible.

A list of certificates is presented to the evaluator, ordered by family members. The certificates are checked one-by-one, and green, blue or red buttons have to be pressed. Green means: "Everything is OK", red means "Certificate is missing", while blue means "Certificate present, value different!" and in this latter case the new value has to be entered in the field.

If the certificates are controlled the process ends by hitting the respective button. A possibility to have a second round of examination must exist for special cases when a more experienced evaluator has to take a look at the demand but this is only optional.

iii. Study score

Data to calculate the study score comes from to sources: the study averages from the previous semester come from the study administration and are imported into the

database, and the data for the current semester come from the student. These latter values are checked then later in the semester.

An important remark has to be added here: marks in Hungary are not like in Austria or Germany. They are simple the opposite; the system was reversed after the dissolution of the Austro-Hungarian Empire, what makes 5 being the best and 1 being the worse mark.

Study average in this case is not a simple average, but the marks are weighted with the credit point (ECTS) of each subject. Then the total is divided by 30 what is the number of credit points a student *should ideally obtain* per semester. Subjects with Fail (F or "nicht genügend") are not taken into account – the credit point is multiplied with 0. This "average" can be under 2.0 (a theoretical *pass* – "genügend" – in every subjects) or over 5.0 (which means more than 30 credit points collected in the semester).

First the average of this "average" for the last two semesters is calculated.

Then the students are distributed into different "homogeneous student groups". These groups represent students that are in the same stage of their studies and should have fulfilled the same requirements. That is for example "students of the old 5-years program of computer engineering, before the specialisation" or "B.Sc. students in electrical engineering". There are in total 7 different groups for the moment.

The scores are then distributed: 100 points means among the best 5% of the homogeneous group, 0 means 2.0 (theoretical minimum to pass every exams) and it can also go below 0: a student having such an average of 1.0 get -20 points, that is a penalty for not doing much in the semester.

iv. Community score

The score awarded for the work for the community is the simplest one for our point: despite it is calculated in a complicated way, it is none of our business: a special committee sets these scores every semester. All we get is a table with student IDs and scores and it gets imported into the table. That's it.

v. Social score

It seems to be the most complicated one, yet it is not as complicated as one would expect.

If certificates are missing the score is automatically zero. This is a rule that is made clear to applicants prior to evaluation.

First the income per family member is calculated (revenue of family members except for the students, divided by the number of family members) and a certain point is attributed, following a simple formula.

After this, points are added for special conditions. For example a brother or sister studying is 3, 5 or 7 points "worth", depending on the level of education (primary, secondary school, or college/university). A pensioner living in the same household gets 3 points. If the student is semi-orphan, he or she gets 6 points, if orphan, he or she gets automatically the maximal score. Disease of the student or family members is scored by the evaluator depending on its seriousness 0 to 3 points etc. If needed, experts are consulted. Disabled students and students having a child automatically get the maximal score.

The scores are than summed up and the total social score is then fixed.

vi. Dormitory placement (future)

While trying to define the specific algorithm to distribute people between dormitories quite a few "dead bodies" have been discovered. Despite the strict regulations the students responsible for this task did not take every single detail into account and dealt with dormitory placements in a few cases a bit superficially. This meant a group of people who got a placement without being entitled to it or people getting a place in a better-situated dorm only for historic reasons: they have ever been there, why change it?

The main point here is that the three factors (social situation, study average, work for community) have to be taken into account when giving placement for someone, and it has to be decided not only whether he or she gets a place or not, but also in which dormitory the place will be available: in a well-equipped one in the city centre or in a lower-class one far from the university.

As the work begun partly new rules have been defined while distributing places. It was defined, that social need is the most important criteria on the places distributed (note that places for students with scholarships or doctoral students are guaranteed), the two other parameters come only after that, in order to ensure that everyone in real need gets placement. However, it was also made clear that achievements in studies and work for the community are the most important factors while choosing the dormitory building, thus rewarding those students.

This led to a complete redesign of the existing algorithm: first the principles were designed including every possible subgroup (around 30 of them, depending on when the student started his or her studies and different other factors like scholarships etc.) on paper. Then a decree text was created to get the announcement ready, which was followed by a debate in the students' self-government about the too rapid change of this algorithm. Because of this discussion it was decided that these changes will be introduced gradually and a more solid version has been accepted for the first year, thus creating a continuous need to further development of the software and the algorithms.

In this later version the algorithm will be as follows.

First students getting for sure a dormitory placement are selected. First those who are entitled by law or by an agreement to get a placement, then those above a certain social score having a positive study score, then those with excellent study results and those with outstanding work for the community. For the rest of the applicants, the three sums are added up and by decreasing order students are selected.

As soon as we know who will get a dormitory place, we can start distributing people among dormitories.

This time only study score and work for community scores are taken into account. Those with outstanding score get a placement in the dormitory they want. Then all the others get a placement in a decreasing order.

This way a list of occupants in each dormitory is created.

vii. Room arrangement in dormitories (future)

This process cannot be fully automated as communities living on the same floor might want to stay together and this data cannot be collected from every applicant.

To help this, a report should be created with a list of every applicant, next to them all the desired room-mates who got placement in the same dormitory. Such way complete or partly complete rooms are enlisted which can be then easily ordered – manually – on the floor plan of the dormitory.

Following the assignment, the decisions must be entered into the system to ensure data are stored.

viii. Warden module for dormitories (future)

A future module has been designed partly to help the work of dormitory wardens.

Wardens are responsible for moving into and moving out from dormitories. They have to trace the current residents of the dormitory. Beside that they are responsible for room inventories and this task can be supported by this system. Instead of checking forms and storing paper sheets, the mouse is clicked a few times.

ix. Feedback (future)

It is also planned that the feedback form filled in by the user should not be sent as a simple email message but should be inserted into a support request tracking database. It would be important to have all the requests answered without missing any and to have the complete history of the applicants available. As this is quite a complex task, it was decided to include it only in a future version.

x. Send information about current state

It was a primordial goal to ensure the transparency of the system.

To achieve this goal a simple, but useful feature should be implemented: applicants should get notification about every step happening with their application. They should receive an email when the demand gets registered, when the social or study score is

calculated, when the community score gets calculated. They should be informed of every single step taking place in order to increase the trust in this system.

V. Implementation

Let us have a brief review of every technology used during the development process which gives enough reasons to justify, why it took so long to come to an approximate end.

It was originally planned to create a two-tier software, that is the first tier – the database – holding all the necessary data and a second tier – the software – doing everything.

A. MySQL

The business model of MySQL AB, the producer of MySQL, is quite common: the company together with voluntary developers create an open-source product that can be used free of charge. Beside that the company delivers more complex products for large enterprises, embedded systems etc. together with printed documentation and valuable technical support. Despite this most users only take the "community edition", which makes it the most popular database server of the world.

The other possibility – PostgreSQL – is somewhat different, because it is completely open-source. The development is coordinated by a group of people, paid by the sponsors. Of course, commercial support is also available by companies around the globe, but they do not have any official contact to the developers, thus assuring more independence and transparency.

As already said, MySQL was chosen in this phase to be the back-end database. It has support to most common programming languages, so that did not limit our possibilities. While having the first attempt, it was also quite evident to use something that is known to the author.

B. PHP: Hypertext Preprocessor

Well, the first programming language the author used was the well-known PHP, *the programming language that anyone can learn*. It is more a scripting language by default, where the output – a web page in HTML – is generated on the fly. Of course, using PHP is not

very efficient as the source code has to be recompiled every time the page is loaded which is quite a big overhead for systems under heavy load, but based on the number of applicants in previous years, we did not expect such a high load.

PHP can seamlessly access data from databases and can present them in a readable form: in fact PHP program is just a simple web page with dynamic content.

No much afterwards problems arose as the complexity a certain – not very high – level: as standard HTML-formatting was mixed with generating dynamic content, the source code became more and more obscure. Something had to be done!

C. Smarty: a template engine for PHP

The solution at this point was to use of a so-called template engine. It is used to facilitate the separation of application code from the presentation code, thus a new architecture, the so called three-tier architecture is created.

It is a kind of a rule of thumb: in most cases it is a good choice to separate business logic and presentation. This way the work of programmers and designers do not interfere as much and plenty of otherwise programmatic tasks can be solved automatically: the compiler for the template engine replaces the tags with the corresponding PHP code.

The template engine used for this purpose was Smarty[11], an extension to the standard PHP. Despite an advanced caching mechanism that compile templates and program code together, it has to be added that it do not improve final execution time as the software stays in PHP that has to be compiled at run-time.

D. OpenLaszlo

OpenLaszlo is a platform created by Laszlo Systems Inc. that has been recently open sourced which has gained in importance due to that. The platform is ideal for creating *"zero-install web applications with the user interface capabilities of desktop client software.*

OpenLaszlo programs are written in XML and JavaScript and transparently compiled to Flash and, with OpenLaszlo 4, DHTML. The OpenLaszlo A[pplication] P[rogramming] I[nterface]s provide animation, layout, data binding, server communication, and declarative U[ser] I[nterface]. An

11 http://smarty.php.net

OpenLaszlo application can be as short as a single source file, or factored into multiple files that define reusable classes and libraries.

OpenLaszlo is write once, run everywhere. An OpenLaszlo application developed on one machine will run on all leading Web browsers on all leading desktop operating systems."[12]

That is it basically: a simple page with HTML-like tags and simple JavaScript codes allows us to create a multimedia-featured, modern-looking user interface that is available in the almost universally available Flash format.

E. XML

XML, that is Extended Mark-up Language is in fact a meta-language created by the World Wide Web Consortium (W3C), the international standardisation body of the Internet. It is a descendant of SGML (Standard Generalized Mark-up Language) just like HTML, but XML has as goal to store structured data in a standard format, thus facilitating the sharing of data across different information systems. It is reasonably human-legible, terseness was not considered essential in its structure. Different standard XMLs have been defined like MathML, Scalable Vector Graphics (SVG) etc. in order to allow reliable transfer between different software systems in a standardised way.

F. The Eclipse Platform

The main reason to abandon PHP with the Smarty template engine and to use Java technologies was the impressive support for Java technologies in the Eclipse Integrated Development Environment (IDE).

An IDE is a complete tool kit that supports software development and Eclipse is very probably the best of the existing ones.

Eclipse is a Foundation that develops the initially by IBM created widely extendible platform. Its founders and industry partners like IBM, Rational Software (now part of IBM), Borland, HP, Ericsson, Intel, SAP etc. spend money and

12 http://www.openlaszlo.org

human resources on developing this unified platform, which can be very well extended using a well-conceived plug-in system.

"The Eclipse platform itself is a sort of universal tool platform – it is an IDE for anything and nothing in particular. It can deal with any type of resource (Java files, C files, Word files, HTML files, JSP files, etc.) in a generic manner, but doesn't know how to do anything that is specific to a particular file type. [...] The real value comes from tool plug-ins for eclipse that teach the platform how to work with these different kinds of resources. This pluggable architecture allows a more seamless experience for the end user when moving between different tools than ever before possible."

Eclipse make it possible to get the technical documentation of functions used at the flick of a finger, it does syntax checking, it offers a list of functions and variables to use and as a plug-in a database explorer (and many others) can be included.

G. Java Server Pages

It would have been possible to continue developing the software in PHP under Eclipse, but seeing the mess that was created in the first steps, it was decided to abandon it and to change to Java.

First the Java Server Pages (JSP) technology has been tried. In fact JSPs are HTML-like pages like PHP pages, the difference is in the programming language. JSP pages are then compiled into a Java class which is then executed by a special server, called Apache Tomcat, also available free of charge.

No we got something similar to PHP, why is it better then? Because Java is a technology that is used by many, it has a huge community supporting it and it can be extended to do every possible thing one can imagine.

H. Subversion version control system

If a new bug appears in a system, it has to be traced back to when it appeared, and who created it. It is also useful to have a kind of archive to be able to correct accidental changes in the software code. It is also useful to be able to keep track of different versions of documents (specifications etc.).

That is why we use version control systems.

A version control system is a document repository. These documents (text documents, plans, drawing or source code) can be checked in to the repository (put into the archive) and checked out from the repository (borrowed from the library). The mechanisms to ensure locking and distributed editing are present and as users have to describe the changes they made, it is easy to get a list of changes between versions, as it is also simple to compare different versions of the same document.

That is why it has been decided to use a version control system for this project. There are two standard implementations, both open-source, that came into question: CVS and Subversion. The latter has been chosen, as it is the more modern one which had as goal to improve and extend CVS. A graphical user interface of Subversion is available also, it is called TortoiseSVN[13].

I. Oracle

To tell the truth, there was not much reason to change to the Oracle database engine. It is much more expensive (licensing costs can come to hundred thousand euro) and it is usually a bit slower with low load then MySQL.

It had to be added, though, that the TU Budapest got free Oracle licenses for internal use for every student and teacher at the university and Oracle provides a complete solution with a graphical user interface. The author is not claiming that Oracle is better than MySQL, it is only said that at the time creating the software it was easier to use it.

Altogether it is not an important difference to change the database engine in the software code does not take much effort as the interfaces are very similar.

J. Java Servlet Technology

As JSP pages are automatically compiled to Java servlets, this change was not a big one at first sight. What was more important is the complete change in system architecture. Before simple web pages were created using scripts. Now the complete object-oriented architecture can be used with different objects instantiated, inherited etc. This makes

13 http://turtoisesvn.tigris.org

software design more transparent and more in line with modern software design technologies.

K. Persistence, POJOs, Entity Managers

In spite of the advice of different IT professionals, the use of relation persistence has been refused until this step. Not because of mistrust or disagreement, but because of lack of experience. Finally it has been decided to introduce Hibernate, another open-source software, for relational persistence.

As described on the project homepage[14], *"Hibernate's goal is to relieve the developer from 95 percent of common data persistence related programming tasks, compared to manual coding with SQL and the JDBC API."* Hibernate generates automatically the queries, provides fetching and caching and object conversion. Another important advantage came with the standard Java Persistence API which allows a complete change between supported databases by changing two lines in a configuration file.

Database rows are accessed as simple Java objects, so-called Plain Old Java Objects (POJOs). This is a term that describes instead of using some complicated data access mechanism the way "as simply as possible": each row is the *instance* of the same class, where columns or cells are *attributes* of this instance, which can be modified and read by *setter* and *getter* functions. This approach is just simply the standard Java approach.

Beside making programming much easier, Hibernate also makes it quicker and more secure: instead of creating a new connection every time a user wants to access the database, Hibernate opens at the beginning a few connections and distributes it upon requests. This can save the costly operation of building up a new connection every time. Of course, the results are first checked in the cache and a database access only happens if the data is not found in the cache.

It has to be added that Hibernate is secure against SQL injection attacks: SQL queries cannot be cheated by inserting quotation marks into form fields or any similar way.

Hibernate is just the current state of the art.

14 http://www.hibernate.org

VI. Conclusion

What are the conclusions drawn from this work? First, it has to be stated that this complicated e-governing project was not impossible to solve. Sure, the goals were **overestimated** at the beginning of this project and the way was long, had dead ends, was not easy, **but far from impossible**, for someone working alone and being inexperienced in programming.

And yes, it had to be realised that **programming can be fun** and an interesting task.

We know by now that such a complex system could be created by using **only open-source software** (including the development tools) and this did not lead to any insurmountable difficulty.

We also noticed that for quite a while, goals were getting further and further; this was due to the lack of experience of the author. Finally we came to an end.

And what's next?

Probably the system will **go live** for the demands presented in **June 2007** and that is when the conclusions can be drawn by practice. It is sure that the graphical user interface will need some redesign due to user feedback.

As to draw some general conclusions from this project, it can be stated that developing simple **e-government** projects **is a solvable problem**. With some effort and financial support any organisation could create such a system, which would help citizens access the information and get in touch with the state. **No magic**, no piles of money, **just some thinking**, design and programming is necessary and this could lead to a brand new way of dealing with authorities in the future.

VII. Appendixes

A. Database definition

A short description of the database tables is required to have a good overview on data structures, despite most of the information has been already provided in the previous parts.

For each of the tables the field name will be stated, then the type of the field (**C**haracter, **B**oolean or **N**umeric) followed by a a short description of the content.

i. Table AKTIV

This table contains all the basic data about enrolled students at the faculty.

Frequency of update: once in a semester (theoretically), data received from students' administration office.

Neptun	C	Student's ID (matriculation code)
Nev	C	Name of the student
Email	C	E-mail address
Mobil	C	Mobile phone number (original data may contain occasionally character symbols)
Kepzes	C	Subject of the studies
P_statusz	C	Financial status (self-financed, state-financed, international scholarship, Hungarian living abroad etc.)
Tan_statusz	C	Active or passive (currently enrolled or not)
Sex	N	1 for men, 2 for women (following the data of the national registers)

ii. Table ATLAG

This table contains study averages from previous semesters.

Frequency of update: once in a semester, at the beginning of a new semester the data entered by the students for the last semester is checked against the official records.

Semester	N	Semester the current record concerns

Neptun	C	Student's ID (matriculation code)
Kepzes	C	Subject of the studies
Szakirany	B	Speicalization
Arch_atlag	N	Study average
Id	N	ID

iii. Table BESOROLAS

This table contains the final results for dormitory placement.

Frequency of update: record created while registering demand, data updated regularly as the process runs.

Neptun	C	Student's ID (matriculation code)
Osszpontszam	N	Total score
Felv_csoport	N	Grouping
Besorolasi_kor	N	Round (e.g. PhD-students and very good students get a placement in first round, not that good students after and bad students only in third round)
Kolesz	N	Dormitory
Szoba	C	Room number

iv. Table CSALADTAG

This table contains the data about family members, as entered by the applicant.

Frequency of update: each time the students logs in, as long as his/her demand is not registered (writes to this table is blocked after registration of the demand).

Id	N	Id
Neptun	C	Student's ID (matriculation code)
Nev	C	Name of family member
Rokonsag	N	Degree of relationship
Tevekenyseg	N	Activity as given in table TEVEKENYSEG
Foglalkozas	C	Profession
Apeh_brutto	N	Gross salary as on tax office statement
Apeh_elkul	N	Special income as on tax office statement
Apeh_ado	N	Tax as on tax office statement.

Fizetes	N	Salary
Vallalkozo	N	Income from business
Munkanelk	N	Unemployment aid
Munkanelk_ido	N	Unemployed aid
Gyes	N	Maternity grant
Nyugdij	N	Pension
Rokkant_fok	N	Degree of disability
Fogyatakos	B	Handicapped?
Egyeb	N	Other revenue
Szoveges	C	Written explanation

v. Table DORMS

This table contains a list of dormitories.

Frequency of update: once in a semester.

Id	N	Id
Name	C	Full name of the dormitory
Capacity	N	Capacity of the dormitory in the first period
Address_street	C	Street address of the dormitory
Address_zip	N	Postcode of the dormitory
Address_town	C	Town of the dormitory
Shortname	C	Short name of the dormitory
Capacity2	N	Capacity of the dormitory in the second period (optinal)

vi. Table FAMILYCHECKED

This table contains data about family members of the applicant, as registered after verification. The table is completed by fields about certificates to be presented, where a boolean value stores if it was OK or missing.

Frequency of update: during the verification of certificates.

Id	N	Id, identical to the one stored in table CSALADTAG
Neptun	C	Student's ID (matriculation code)
Nev	C	Name of family member
Rokonsag	N	Degree of relationship

Tevekenyseg	N	Activity as given in table TEVEKENYSEG
Foglalkozas	C	Profession
Apeh_brutto	N	Gross salary as on tax office statement
Apeh_elkul	N	Special income as on tax office statement
Apeh_ado	N	Tax as on tax office statement.
Fizetes	N	Salary
Vallalkozo	N	Income from business
Munkanelk	N	Unemployment aid
Munkanelk_ido	N	Unemployed aid
Gyes	N	Maternity grant
Nyugdij	N	Pension
Rokkant_fok	N	Degree of disability
Fogyatekos	B	Handicapped?
Egyeb	N	Other revenue
Elhunyt	N	Deceased
Diak	N	Student
Kolesz	N	Brother or sister in dorm
Szulakvikiv	N	Birth certificate
Elvalt	N	Divorced
Indoklas	N	Score for special conditions
Jovedelem	N	Total income
Apeh	N	Total income based on tax office statements

vii. Table HHCS

This table contains the homogeneous student groups.

Frequency of update: once in a semester (or even rarely)

Id	N	ID
Text	C	Description
Kepzes	C	Subject of the studies
Szakirany	B	Specialisation
Koltseg	B	Self-financed?
Szoctam	B	Entitled to social aid (e.g. self-finances and doctoral students are not)

Kolesz	B	Entitled to dormitory placement (will be removed, as a new rule changed this)

viii. Table JOGOK

This table contains the different rights administrators have.

Frequency of update: as new users are added or rights are changes (a few times in each semester)

Usernam	C	User name of administrator
Nev	C	Name of administrator
Iktat	B	Right to register demands?
Elsokor	B	Right to check demands in first round?
Masodikkor	B	Right to check demands in second round?
Szocmodosit	B	Right to modify demands later on?
Szocref	B	Chief
Login	B	Right to log in?
Import	B	Right to import tables (a used in FileUpload.java)

ix. Table KOLESZ

This table contains data entered by the student about his request for dormitory placement.

Frequency of update: entered by applicants regularly.

Neptun	C	Student's ID (matriculation code)
Atlag	N	Study average for last semester, as entered by the applicant.
Szobatars1, szobatars2, szobatars3	C	Requested room-mates
Kolesz1, kolesz2, kolesz3, kolesz4, kolesz5	N	Requested dormitories for first period
KoleszA, koleszB, koleszC, koleszD, koleszE	N	Requested dormitories for second period (optional)

x. Table KOZOSSEGI

This table contains the score for the work for the community.

Frequency of update: once in a semester (theoretically)

Neptun	C	Student's ID (matriculation code)
Felev	N	Semester
Kozossegi_pont	N	Score for the work for community

xi. Table OPTIONS

This table stores different settings and options (current semester, information if logging in is allowed etc.).

Frequency of update: when needed (a few times in a semester)

Key	C	Key of the option
Value	C	Value of the option

xii. Table PALYAZAT

This table is a basic table where all demands and users are registered.

Frequency of update: upon login or registration of applicants.

Neptun	C	Student's ID (matriculation code)
Password	C	Login password hash, created with MD5 algorithm
Lastlogin	D	Date and time of last login
Szocialis_ker	B	Requests social aid?
Kollegium_ker	B	Requests dormitory placement?
Mobil	C	Mobile phone number, as entered by the user
Email	C	E-mail address, as entered by the user
Allow_login	B	Login is disallowed after registering the demand.

xiii. Table RESZEREDMENY

This table contains partial results of each demand.

Frequency of update: always when the demands gets into a following step or any partial score is calculated.

Neptun	C	Student's ID (matriculation code)
Szocialis	N	Social score
Kozossegi	N	Community score
Tanulmanyi	N	Study score
Szobatars1, szobatars2, szobatars3	C	Requested room-mates
Kolesz1, kolesz2, kolesz3, kolesz4, kolesz5	N	Requested dormitories for first period
KoleszA, koleszB, koleszC, koleszD, koleszE	N	Requested dormitories for second period (optional)
Megjegyzes	C	Comment entered during the evaluation
Status	N	Status as described in the table STATUS
Sorszam	N	ID attributed to the demand while filing the papers. This is the key to retrieve the demand from the folders.

xiv. Table STATUS

This table contains all possible status.

Frequency of update: never.

Value	N	Status value
Text	C	Long description of status
Text_short	C	Short description of status

xv. Table STRINGSEQ

This table contains all strings to be printed.

Frequency of update: never

Shorttext	C	Key of the text to be printed
Longtext	C	Long text to be printed.

xvi. Table SZOCIALIS

This table contains data about the applicant if s/he asks for social aid.

Frequency of update: every time the applicant modifies data.

Neptun	C	Student's ID (matriculation code)
Diakhitel	N	Study loan revenue
Esely	N	Scholarship of the *Chance to Study* foundation (governmental fund)
Sajat	N	Own salary
Bursa	N	Scholarship from *Bursa Hungarica* communal scholarship fund
Arvaellatas	N	Orphan's aid
Szoveges	C	Written demand
Tavolsag	N	Distance from school in km

xvii. Table SZOCIALISCHECKED

This table contains the data created during evaluation for the social score.

Frequency of update: each time a data is updated during the evaluation.

Neptun	C	Student's ID (matriculation code)
Sajat	N	Own salary
Arvaellatas	N	Orphan's aid
Egyhazt	C	Number of people in the household
Tavolsag	N	Distance from school in km
Palyazoindoklas	N	Score attributed due to special condition
Megjegyzes	C	Comment entered during the evaluation

xviii. Table TEVEKENYSEG

This table contains a list of all possible economic activities a family member can carry out.

Frequency of update: never.

Id	N	ID
Text	C	Description of activity

B. Business Intelligence

The technical documentation is more or less the documentation bound to the source code. A description about each of the files used is included, partly in a standard format, automatically generated by JavaDoc, the documentation formatter of Java from the formatted comment of the author, and partly using a non-standard style to present the most important information.

i. Database access classes (POJOs)

Database fields are accessed through the Java Persistence Layer which is then connected to the database through the Hibernate relational persistence suite. To describe it shortly: after finding the record in question (by giving the primary key or by entering a more complex select-like statement as criteria), an instance of the class describing the row is created and each field can be then queried by using the related GET-method (for example getName() to get the NAME field). To modify such a field, the SET-method has to be used, like setName("John Doe"), in both cases of course by giving the name of the object like o.getName(). These are called generally POJOs or Plain Old Java Objects to make it clear that this is the standard way in Java to access objects and instead of a shiny new feature, this very simple and traditional method is used to access the data.

All these objects are automatically generated by Hibernate from the database schema, or these files could be used to let the underlying schema modified. As each file contains apart from the constructor methods only setters and getters, only the names are enlisted, as a more detailed description of each field has already been included in the previous chapter.

Unfortunately here again the mixed Hungarian-English naming is still present, which will disappear in a later version.

ii. Business intelligence

The documentation of the kernel of the software can be found in appendix, as it has been included in the source code and then generated automatically by the JavaDoc tool.

C. Presentation layer

First the presentation layer is presented. Basically the files are split up into different directories in order to create a logical structure, which could be also a basis for splitting up dynamic and static content.

All the files with .lzx extension contain an XML-definition later compiled into a flash animation describing the content of a page. A few other files are also included, like data stored outside of the database, images, small animations, Cascading Style Sheets (CSS) and Java Server Pages (JSP) files, both responsible for one part of the presentation of the whole site.

File or directory	Description
/	Root directory to store all the files required for the presentation layer. It has to be placed to the working directory of the web server or the server has to be configured to use the files from a different directory.
explore-nav.lzx	Navigation pane on the left side. Original idea from OpenLaszlo example, later transformed to include login pane. Structure is read from nav.xml file, different menus are shown if the user is logged in or not. By clicking any button the content described in the nav.xml is loaded into the right pane. Two fields are available for login (user name and password) with two buttons: then hitting the *Log in* button, the user's credentials are checked against its password stored in a scrambled way in the database, if the match, the log in is completed. If a failure occurs, the user gets an error message. If the user hits the *Register* button after entering a user name (which is identical to the matriculation number), he/she gets a password by e-mail to the central e-mail

	address and to his or her e-mail address stored in the database of the study administration.
index.jsp	This simple Java Server Pages site is responsible for loading the navigation pane (explore-nav.lzx) to the left side and the requested page to the right side. This page was also included in the OpenLaszlo examples.
koleszfelveteli.lzx	Page where the users should enter the required data if they request a place in a dormitory. First it must be announced if a placement is required, and if yes, a few other details have to be entered: study average, order of requested dormitories and room-mates.
nav.xml	Data file containing the menu, displayed by explore-nav.lzx. Two complete menus are described, one for not logged in users (menustandard) and one for logged in users (menufull).
szocpont.lzx	This is the place where the user can request social aid (and of course social score for the dormitory placement). If he/she requests social scores, a bunch of details has to be entered about social situation. First the data about the applicant should be entered, study loan, different scholarships, orphan aid, own income, distance of permanent residence and detailed written application. Then the persons of the household have to be added, one by one, giving for each the name, the degree of relationship, the occupation, income as stated by the tax office, income stated by employer, pension's office, labour office etc. A detailed application can be included, if any special conditions are present.
torzsadat.lzx	This is the simple page where the user can enter its personal contact details (actual e-mail address and mobile

	phone number).
static/	This folder is used to store all the static pages, which are seldom modified.
static/faq.lzx	This page was the first page created while using OpenLaszlo. It uses an external XML data file (faq.xml) to store the Frequently Asked Questions and the related answers. Not very complicated, it only displays the data hold in the file in a given format.
static/faq.xml	Very simple XML file containing the questions and the answers, as server by faq.lzx.
static/kapcsolat.lzx	Contact form, which allows sending emails in a given format to the reviewer. Its advantage is that it can include details which help tracking back the demand and while categorizing the problem, it can be directly processed by the person in charge.
static/kiiras.lzx	This page holds the text of the official announcement. Only static content is available.
static/kolesz.lzx	This page hold important information for the dormitory application. It can be considered as an extract from the official announcement.
static/login.lzx	This page gives information about the login process and helps with advice, if an error occurs.
static/palyazat.lzx	This page gives general information about the demands.
static/register.lzx	This page gives information about the registration process and helps with advice, if an error occurs.
static/szocialis.lzx	This page gives important information about the social part and describes the whole process.
static/welcome.lzx	This is just a dummy page to welcome the users.
lib/	This directory contains libraries, which are used to perform basic functions of the presentation layer.
lib/basic.lzx	Two basic modules are contained in this file: one for users not logged in and users logged in but not having the

	necessary permissions. They are show in every screens of the right side if the necessary event is evoked.
lib/explore.css	A Cascading Style Sheet page containing information about the look of the left-side menu. Also included from OpenLaszlo example.
lib/menu.lzx	Important modules to have the left-pane menu working. Also included from the OpenLaszlo example code.
lib/assets/	Small Flash animations which appear when the user clicks on a button of the left-side menu, like the sub-menu dropping down etc. Included from OpenLaszlo example code.
lib/images/	Images included in the menu, like different arrows. From the OpenLaszlo example.
admin/	The admin/ directory contains most files required by the administration interface. Users have to get a special authentication for this section and other methods are incorporated to limit access to this site.
admin/checkDemand.lzx	This is the key part of the evaluation interface. Simplicity was defined as a goal, that is why only a simple field is displayed where the matriculation number of the student has to be entered, and an OK button are on the screen. Beside entering the key on the keyboard, it was conceived the way that a bar-code scanner can also be used to read in the code and automatically hit the OK button at once. When registering the demand, only the attributed ID is printed on the screen, which have to be written on the demand. While checking the demands for social score, a list of certificates hast to be verified; for each certificate the person, a short description and the amount is presented, while giving the evaluator the possibility to decide by hitting the appropriate button if its OK (green), it it is

	missing (red) or the certificate is there but the value the user entered is incorrect (blue). For each demand there can be as many as 30 lines, that is why it was important to make this process as simple as possible. To make this task easier, there is no need to check the demands for dormitory placements, as all students enter the details in a form where the entries are checked syntactically. As these were the data the user entered, no more verification is required.
admin/explore-nav.lzx	Navigation pane on the left side, similar to the previous one. Two fields are available for login (user name and password) with only one button: then hitting the *Log in* button, the user's credentials are checked against its password stored in a scrambled way in the database, if the match, the log in is completed. If a failure occurs, the user gets an error message. Optionally a verification of the user's IP address can be included.
admin/import.lzx	This is a tool that provides an interface to import data to the database. Data exported from Excel to Comma-Separated-Values format can be imported, containing data about students (name, address, email etc.), the study averages for previous semesters, the scores for work for the community and the list of character strings used in the program. The main purpose for this import tool was to overcome important character conversion problems, caused by the special Hungarian character set.
admin/index.jsp	This simple Java Server Pages site is responsible for loading the navigation pane (explore-nav.lzx) to the left side and the requested page to the right side. This page was also included in the OpenLaszlo examples.

admin/nav.xml	Data file containing the menu, displayed by explore-nav.lzx. Two menus are described, one containing nothing, for users that have nothing to do there (menustandard) and one for logged in and authenticated users (menufull), who participate in the evaluation.
admin/search.lzx	This is an interface to a tool which makes it possible to find demands. Data from the demands can be also modified on this interface.
admin/stringseq.lzx	This is an interface to create new character strings to be printed on the screen, by using the character set required.
admin/static/welcome.lzx	This is a welcome page, where important actual information may be placed to the attention of all the evaluators.