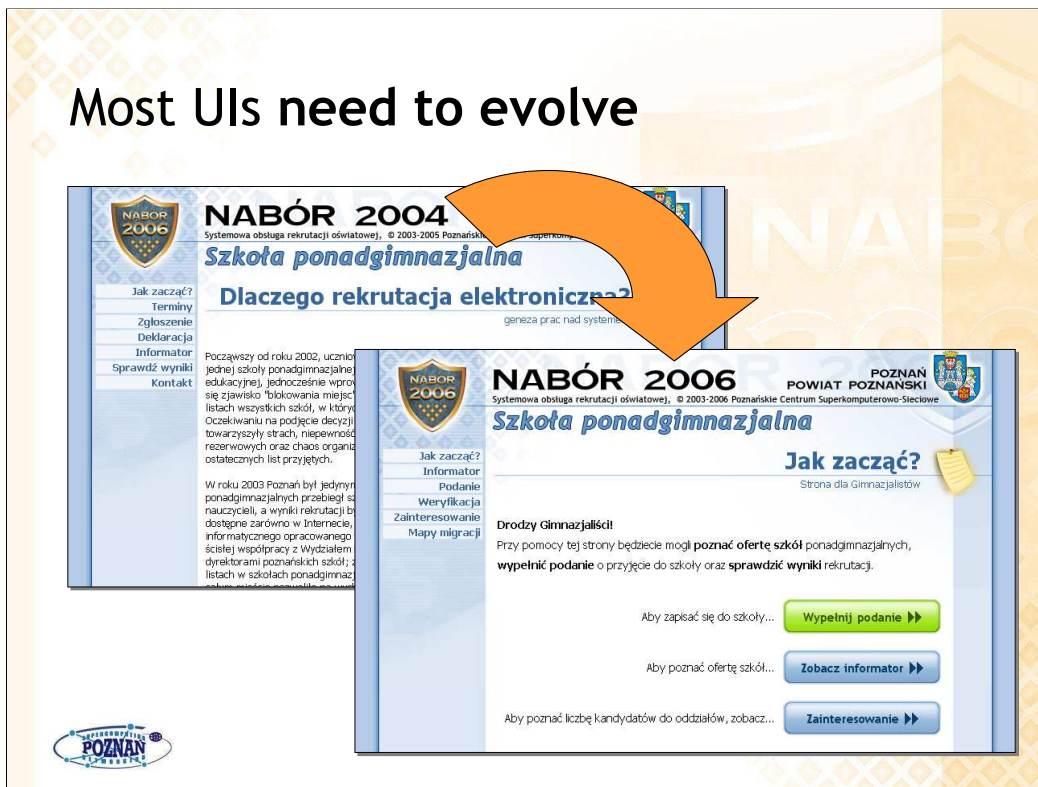


Using historical data to guide user interface evolution

Stanislaw Osinski, stanislaw.osinski@man.poznan.pl
Poznan Supercomputing and Networking Center, Poznan, Poland



Most UIs need to evolve

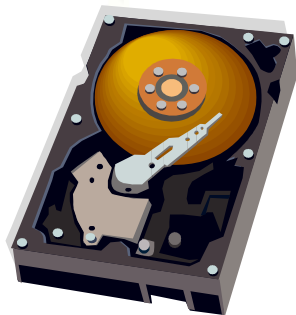


Most UIs need to evolve and we'd rather have them **evolve in the right direction**. This presentation shows how historical data, such as databases or access logs, can be used to support the user interface redesign process.

Examples in this presentation come from an **on-line high school admission system called Nabor** that PSNC has been working on since 2003. Nabor is used each year by over 40,000 16-year-olds to apply to senior high schools in various Polish cities. The 2004 edition was the first one to offer a web-based system for the candidates to fill-in their high school applications. Unfortunately, **the initial design wasn't built with usability and learnability in mind**, which made it quite difficult to, for example, guess how to start filling-in the application. One **goal for the 2006 edition was to redesign the UI** to make it easily learnable by 16-year olds.

The most important input for the redesign was usability testing of the original software and prototypes of the new version. However, we also used the fact that the the **Nabor system had gathered a lot of historical data, that could potentially support the redesign process**.

Historical data can be helpful



```

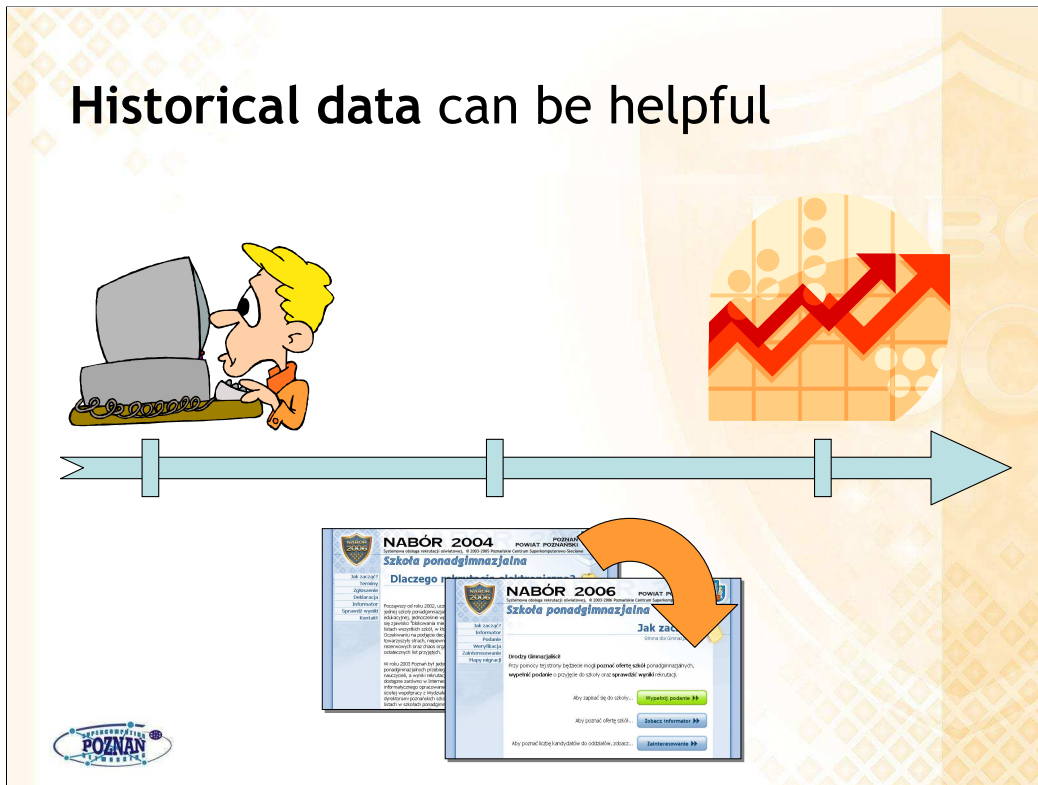
127 0 0 1 -- [28/Jul/2006:15:30:49 +0200] "GET /demo/flush-test HTTP/1.1" 405 1112
127 0 0 1 -- [28/Jul/2006:15:30:56 +0200] "GET /demo/flush-test HTTP/1.1" 405 1112
127 0 0 1 -- [28/Jul/2006:15:30:59 +0200] "GET /demo/flush-test HTTP/1.1" 405 1112
127 0 0 1 -- [28/Jul/2006:15:34:01 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:34:39 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:34:55 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:34:56 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:34:56 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:36:42 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:37:55 +0200] "GET /demo/flush-test HTTP/1.1" 200 284
127 0 0 1 -- [28/Jul/2006:15:40:00 +0200] "GET /demo/flush-test HTTP/1.1" 200 1907
127 0 0 1 -- [28/Jul/2006:15:40:19 +0200] "GET /demo/flush-test HTTP/1.1" 200 3179
127 0 0 1 -- [28/Jul/2006:15:40:35 +0200] "GET /demo/flush-test HTTP/1.1" 200 3179
127 0 0 1 -- [28/Jul/2006:15:45:49 +0200] "GET /demo/flush-test HTTP/1.1" 200 111
127 0 0 1 -- [28/Jul/2006:15:46:07 +0200] "GET /demo/flush-test HTTP/1.1" 200 111
127 0 0 1 -- [28/Jul/2006:15:47:41 +0200] "GET /demo/flush-test HTTP/1.1" 200 3179
127 0 0 1 -- [28/Jul/2006:15:48:35 +0200] "GET /demo/flush-test HTTP/1.1" 200 111
127 0 0 1 -- [28/Jul/2006:15:56:20 +0200] "GET /demo/flush-test HTTP/1.1" 200 145
127 0 0 1 -- [28/Jul/2006:15:56:48 +0200] "GET /demo/flush-test HTTP/1.1" 200 90
    
```

DIRECT SERVICE TIMES								
Page name	Avg [s]	Min [s]	Max [s]	Std dev [s]	Sum [s]	Count	Sum/day [s]	Count/day
edu.Config.ShowTimes / \$Form	0.124	0.109	0.136	0.014	0.372	3	0.048	0.0
edu.Prime.CandidateEdit / \$DirectButtonLink	0.249	0.218	0.269	0.027	0.746	3	0.095	0.0
edu.Prime.CandidateEdit / \$Form	0.347	0.347	0.347	0.000	0.347	1	0.044	0.0
edu.Prime.CandidateEnter / \$Form	0.190	0.127	0.354	0.082	1.142	6	0.146	0.0
edu.Prime.CandidateEnterStart / \$Form	0.225	0.019	0.633	0.248	2.251	10	0.288	1.0
edu.Prime.CandidateList / \$Form	0.146	0.067	0.231	0.082	0.437	3	0.056	0.0
edu.Prime.CandidateList / \$PrimeSchoolCandidateListFlow.candidat	0.366	0.056	0.608	0.226	1.831	5	0.234	0.0
edu.Prime.CandidateList / \$PrimeSchoolCandidateListFlow.candidat	0.185	0.015	0.787	0.228	2.033	11	0.260	1.0
edu.Prime.CandidateView / \$PrimeSchoolParticipantDetailLink.link	0.255	0.059	0.423	0.150	1.020	4	0.130	0.0
edu.Prime.GeneratePerel / \$Form	0.272	0.272	0.272	0.000	0.272	1	0.035	0.0
edu.Prime.HomeInPB / newReportMenu.grouplink	0.100	0.100	0.100	0.000	0.100	1	0.013	0.0



The historical data produced by the software being redesigned can include the **domain-specific database, usage or access logs**. The domain-specific databases can give valuable information about the general characteristics, **trends and distribution of data** processed by the system. In case of Nabor, we used the old databases to find out, for example, how many high school applications were filled in on-line by the candidates. The usage logs and access logs, on the other hand, can give a rough estimation on **how frequently certain functions and elements of the UI are accessed** by the users.

Historical data can be helpful

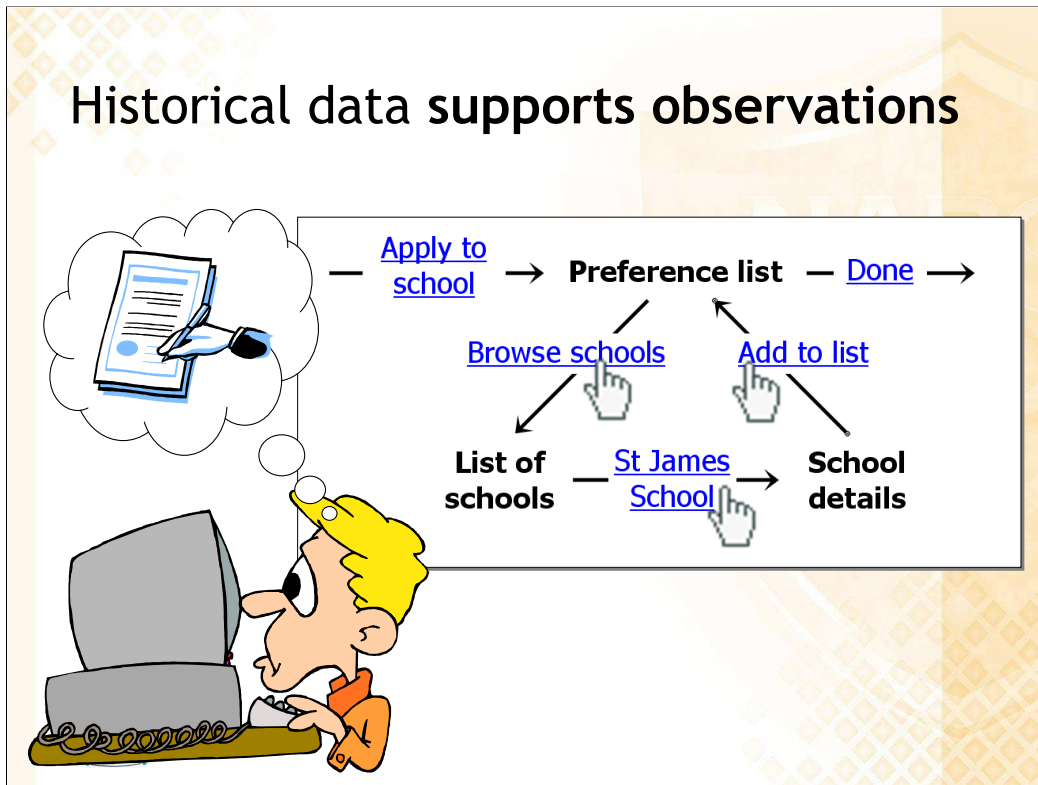


Historical data can be used to support the three main stages of the redesign project:

- during **analyses of the original UI** historical data can support usability observations,
- during **prototyping of new UIs** historical data can guide decisions that are hard to make based only on usability testing
- historical data can help to **verify how the redesigned UIs worked in practice** (e.g. in the production system)

The next three slides give examples of how we used historical data on these three phases in case of the Nabor redesign project.

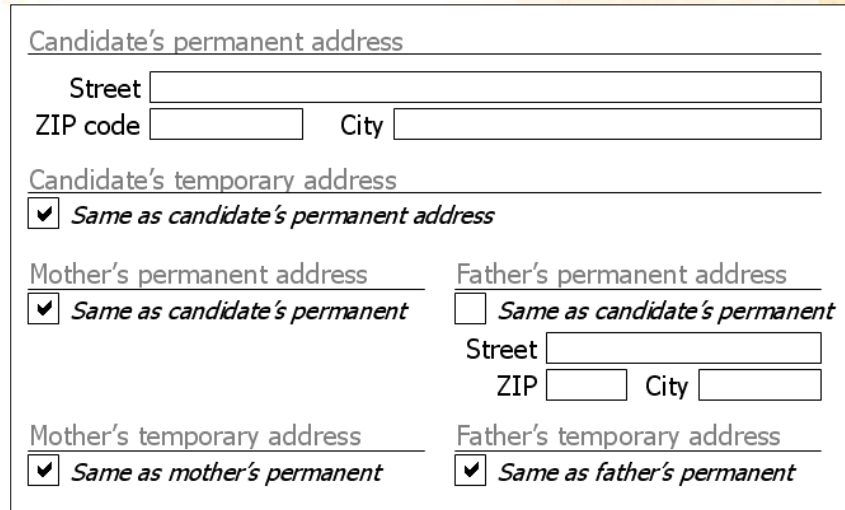
Historical data supports observations



One element of Nabor's on-line application process requires that candidates build a list of schools to which they would like to get admitted (in the decreasing order of preference). The original design assumed a model based on the **shopping basket metaphor**, in which adding a new school to the list required: 1) browsing through the list of all schools in a city, 2) opening a description page of the school to be added, 3) clicking the *Add to list* link. In theory, this model would not necessarily have to cause problems, but user feedback and usability tests with a number of 16-year-olds showed that it could cause many candidates to fail to fill-in their applications on-line. We felt that supplementing the shopping basket model with a simpler direct drop-down-based direct school choice could be beneficial, which was also confirmed by informal usability tests of a prototype we carried out.

Unfortunately, no access logs from that time were available to further support our observations. If we had such data, we could e.g. calculate the percentage of successful on-line applications (which was about 25% according to school administration estimations), which would further support the results of our usability tests.

Historical data helps to make decisions



Candidate's permanent address

Street

ZIP code City

Candidate's temporary address

Same as candidate's permanent address

Mother's permanent address Father's permanent address

Same as candidate's permanent *Same as candidate's permanent*

Street

ZIP City

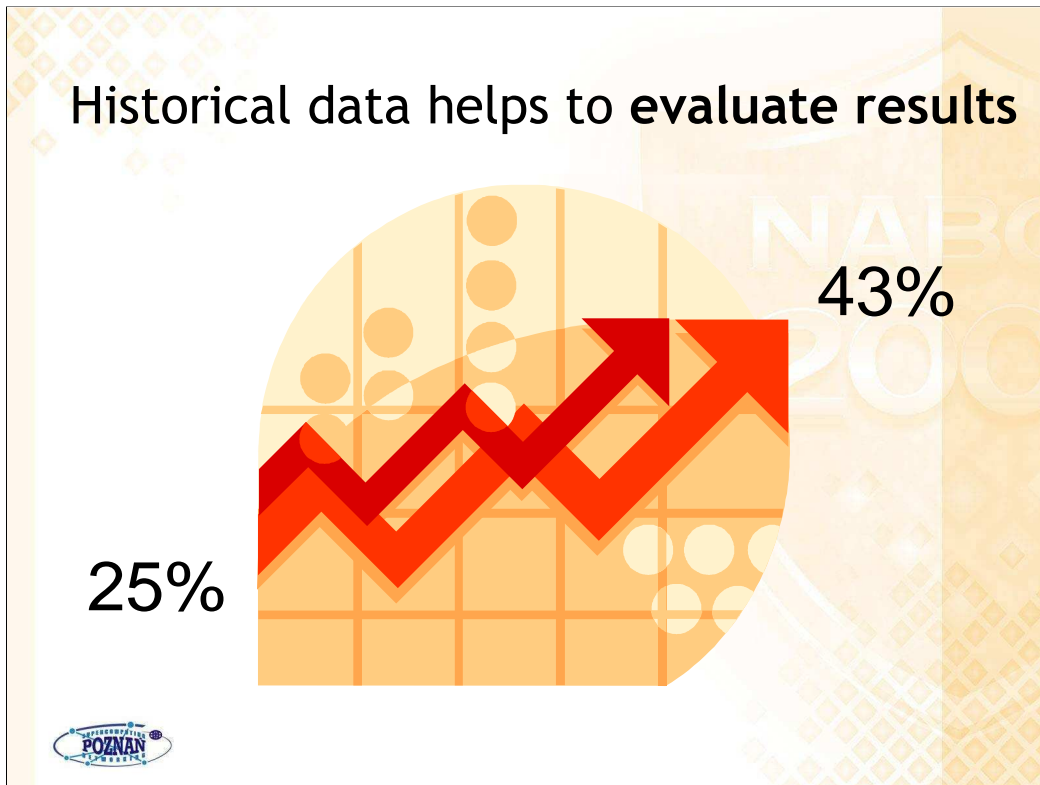
Mother's temporary address Father's temporary address

Same as mother's permanent *Same as father's permanent*



During the course of the detailed design of the new user interface we stumbled upon a number of questions, which could be answered with more confidence after analysing the data found in historical databases of the *Nabor* system. One problem was that the application process required that candidates provide up to 6 different addresses: their permanent and temporary address, their mother's and father's permanent and temporary addresses. The previous year's *Nabor* database provided a helpful design hint for this case: out of 5357 candidates, 4591 (85%) had all six addresses equal. This supported our initial intuition that a variant of the **responsive disclosure pattern (with all addresses equal by default)** would be most appropriate here.

We also used historical data while designing a printable version of the high school application form to make the form fit on one A4 page for the majority of candidates. According to the database, the maximum length of the preference list was 26, but for 99% of candidates, the length did not exceed 18. We therefore ensured that a page break would not occur when printing application forms with up to 18 preferences.



To additionally confirm our design-time decisions and gather data for future interface improvement, we ensured that the potentially useful data, such as the percentage of application forms completed on-line, is retained in the system's database.

According to the content of the statistics we gathered, after the UI change 43% of high school applications were completed on-line, which is a noticeable improvement over the 25% for the original design. Additionally, page access statistics show that less than 20% of the candidates used the original model to build their school lists and over 80% preferred the redesigned model.

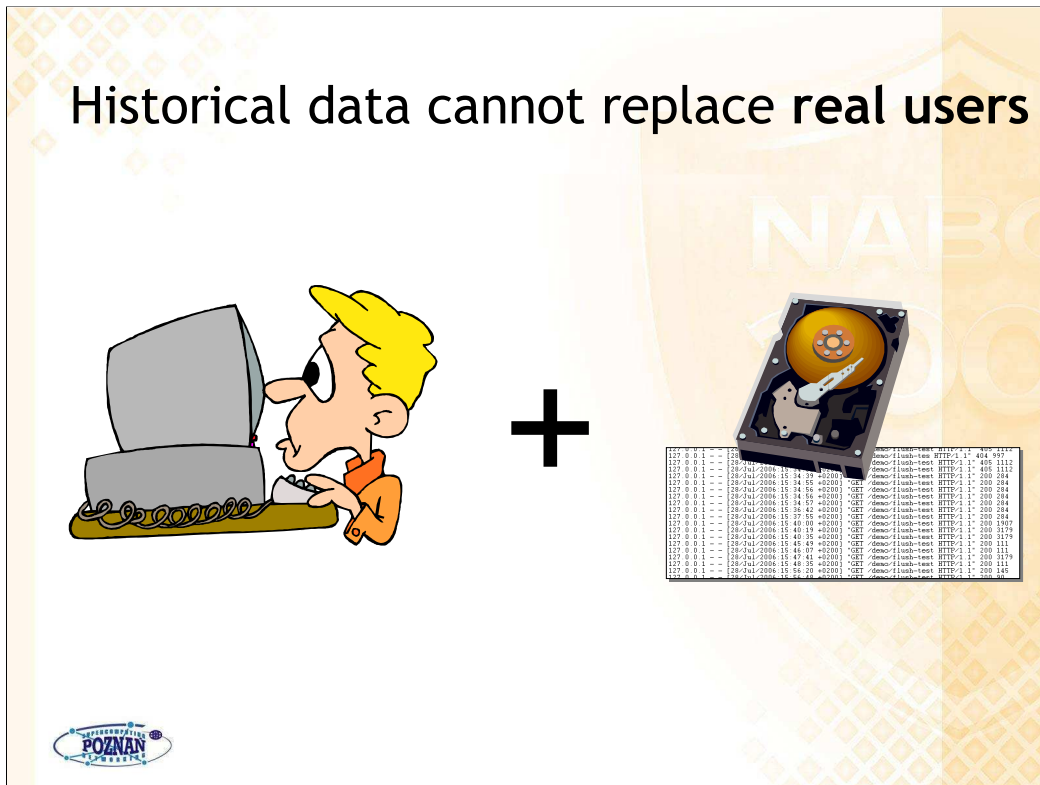
Extra work is required

DIRECT SERVICE TIMES									
Page name	Avg [s]	Min [s]	Max [s]	Std dev [s]	Sum [s]	Count	Sum/day [s]	Count/day	
eduConfig:ShowTimes / \$Form	0,124	0,109	0,136	0,014	0,372	3	0,048	0,0	
eduPrime:CandidateEdit / \$DirectButtonLink	0,249	0,218	0,269	0,027	0,747	3	0,095	0,0	
eduPrime:CandidateEdit / \$Form	0,347	0,347	0,347	0,000	0,347	1	0,044	0,0	
eduPrime:CandidateEnter / \$Form	0,190	0,127	0,354	0,082	1,142	6	0,146	0,0	
eduPrime:CandidateEnterStart / \$Form	0,225	0,019	0,633	0,248	2,250	10	0,288	1,0	
eduPrime:CandidateList / \$Form	0,146	0,067	0,231	0,082	0,438	3	0,056	0,0	
eduPrime:CandidateList / \$PrimeSchoolCandidateListRow.candidat	0,366	0,056	0,608	0,226	1,830	5	0,234	0,0	
eduPrime:CandidateList / \$PrimeSchoolCandidateListRow.candidat	0,185	0,015	0,787	0,228	2,030	11	0,260	1,0	
eduPrime:CandidateView / \$PrimeSchoolParticipantDetailsLink.link	0,255	0,059	0,423	0,150	1,020	4	0,130	0,0	
eduPrime:GeneratePesel / \$Form	0,272	0,272	0,272	0,000	0,272	1	0,035	0,0	
eduPrime:HomeInPB / newReportMenu.groupLink	0,100	0,100	0,100	0,000	0,100	1	0,013	0,0	

In order to make the historical data most useful in the context of UI redesign, it is important to ensure (at the software design/development stages) that the potentially useful data is **retained in the database**. Furthermore, during the deployment phase it is important to ensure that the vital usage/access logs do not get discarded after some period of time.

More specific **tools for gathering simple UI usage statistics** can also be built into the software. In case of the Nabor system, we developed a simple mechanism for counting and storing the number of times particular elements of the web-based user interface (links, form submission) were accessed. This kind of information will let us more effectively optimize Nabor's user interface in the future.

Historical data cannot replace real users



Using historical data in the UI evolution process is most **effective when**:

- There is access to a large database containing the application's historical domain-specific data (in this case it is not important whether the application was implemented in the web or fat client model)
- There is access to usage logs covering a large period of time (most practical for web applications)
- The software development process allows iterative delivery
- The user base is large enough to generate significant amounts of historical data

Using historical data in the UI evolution process **may not be effective when**:

- The historical data gathered in the domain-specific database is not large enough for the conclusions to be statistically significant
- The usage logs are not easily interpreted and there is no purpose-built application for tracking UI element usage

Most importantly, historical data should be treated as a measure that **complements but not replaces** usability testing with real users.

Thank you

Using historical data to guide
user interface evolution

Stanislaw Osinski, stanislaw.osinski@man.poznan.pl
Poznan Supercomputing and Networking Center, Poznan, Poland

