Szkoła Główna Handlowa

Piotr Kucharski

Nr albumu: 7431

Zintegrowana zmiana haseł

Praca zaliczeniowa z przedmiotu
PROJEKTOWANIE SYSTEMÓW INFORMACYJNYCH
u prof. dr hab. Jana Golińskiego

Styczeń 2007

Spis treści

1. Wstęp

W wielośrodowiskowym systemie, przy istnieniu różnych baz zawierających dane o użytkownikach, zawsze występują mniejsze i większe problemy z synchronizacją danych. Opiszę tu rozwiązanie jednego z problemów: zmiany hasła.

Systemy informatyczne Szkoły Głównej Handlowej od samego początku były zdywersyfikowane. Na początku lat 90. jako podstawowy serwer plików i serwer dostępowy służył Netware 3.10, a maszyna z Solarisem 2.3 była serwerem poczty. Na przestrzeni lat Novell oddał pole na rzecz serwerów Windows, które obecnie są podstawowym systemem dostępowym na terenie uczelni, zaś maszyny z różnymi wersjami UNIX-a (FreeBSD oraz Solaris) umocniły się w zastosowaniach internetowych i pocztowych.

Od samego początku każdy z tych systemów miał swoje oddzielne bazy użytkowników, własne metody uwierzytelniania i zarządzania użytkownikami oraz – co z punktu widzenia klientów usług najważniejsze – odmienne procedury zmiany haseł. Najważniejsze, a jednak od samego początku przysparzało najwięcej kłopotów.

Dyskomfort klientów spowodowany tymi problemami wymusił wyprodukowanie rozwiązania, które pozwala na łatwą zmianę hasła na wszystkich systemach – zarówno samodzielnie przez użytkownika, jak i użytkownikowi przez obsługę.

2. Istniejący system

Mimo heterogeniczności środowiska dobra synchronizacja informacji o użytkownikach pomiędzy systemami umożliwiła założenie, że dany użytkownik posiada na każdym z systemów konto o takim samym identyfikatorze.

2.1. UNIX

Konta i hasła użytkowników trzymane były w plikach systemowych `/etc/passwd` i `/etc/shadow`. Przy istniejącej wielkości systemu (prawie 40 tys. klientów) czasem dochodziło do blokowania jednoczesnych zapisów, a w warunkach wysokiego obciążenia doprowadzało wręcz do uszkodzenia listy użytkowników. Dodatkowo system zmiany haseł nie był zbyt wygodny dla obsługi (działał w powłoce uniksa, nie pozwalał na wyszukiwanie użytkowników).

Na wiosnę 2005 r. użytkownicy UNIX-a zostali zmigrowani do LDAP-a, co wyeliminowało problemy z blokowaniem przy jednoczesnych zapisach i jednocześnie umożliwiło tworzenie aplikacji zarządzających danymi użytkowników z poziomu WWW[1].

2.2. Windows

Konta w systemie Windows NT były trzymane w plikach systemowych na kontrolerach domeny. Użytkownicy mogli zmieniać swoje hasła w standardowy dla Windows sposób („Zmień hasło" po Ctrl-Alt-Delete), ale tylko po zalogowaniu na komputerze, który był włączony do domeny. Obsługa sieci mogła zmieniać hasła użytkownikom przy pomocy programu usermgr. Na wiosnę 2006 r. został w końcu wdrożony Windows 2000 i system ActiveDirectory oparty na

[1] Czego przykładem jest obecna Książka Adresowa SGH
`http://akson.sgh.waw.pl/ksiazka/`

LDAP. Użytkownicy mogli zmieniać sobie hasło jak dotychczas, natomiast obsługa sieci musiała korzystać z nowego programu: AdminTool.

2.3. Netware

W wersji Netware 3.xx dane o kontach i hasłach przechowywane były w bindery, użytkownicy mogli zmieniać swoje hasła DOS-owym programem setpass (oczywiście tylko wtedy, gdy korzystali z klienta VLM). Ze względu na zbyt duże uprawnienia wymagane do zmiany haseł innym użytkownikom przy pomocy programu setpass, obsługa sieci dostała jeszcze inne narzędzie: program syscon. Z powodu ograniczeń bindery na liczbę użytkowników, na wiosnę 1999 r. nastąpiła migracja do NDS[2], który także jest oparty na LDAP. Użytkownicy w dalszym ciągu korzystali z setpass, a obsługa sieci dostała kolejne narzędzie: chpass. Z czasem pojawiły się w eDirectory inne rodzaje haseł i żeby je zintegrować, wprowadzono hasło uniwersalne, do którego nie wystarczała dotychczasowa aplikacja chpass. Została wprowadzona zmiana haseł przez WWW, a obsługa sieci dostała program NWadmn (NetWare Administrator), w wyglądzie podobny do usermgr/AdminTool dla Windows.

3. Cel, zakres i kontekst systemu

Celem systemu jest dostarczenie możliwości zmiany hasła *w jednym miejscu* na wszystkich systemach (UNIX, Windows, Netware), zarówno samodzielnie przez użytkownika, jak i użytkownikom przez obsługę.

Zakres systemu obejmuje niewielki wycinek zarządzania kontem, czyli zmianę hasła.

Klientów systemu można podzielić na dwie kategorie: obsługę sieci (ok. 20 osób) i użytkowników sieci (ok. 40 tys. osób). Użytkownicy sieci będą zmieniać swoje hasła, zaś obsługa sieci będzie zmieniać hasła użytkowników sieci. Istnieje jeszcze (choć bardzo rzadko występuje) administrator systemu (jedna osoba), który nadaje uprawnienia obsłudze systemu.

4. Spis wymagań

4.1. Wymagania funkcjonalne

1. Klient musi się prawidłowo uwierzytelnić (czyli podać prawidłowe login i hasło).
2. Obsługa sieci musi mieć możliwość wyszukania użytkownika według różnych kryteriów.
3. Obsługa sieci nie może sama wybierać haseł dla użytkowników.
4. Obsługa sieci musi mieć opisane uprawnienia (kto i komu może zmieniać hasło).
5. Klient musi wybrać systemy, na których chce zmienić hasło.
6. System musi zapisywać dziennik z pracy obsługi sieci (kto, komu i kiedy zmieniał hasło).
7. System musi zweryfikować hasło użytkownika pod kątem jego bezpieczeństwa.
8. Użytkownik może zażyczyć sobie zaproponowania mu bezpiecznego hasła.
9. Klient musi otrzymać komunikat informujący o poprawnej zmianie hasła lub komunikat błędu wyjaśniający (lub choć wskazujący) przyczyny błędu.
10. System musi być bezpieczny, bez możliwości podsłuchu lub ujawnienia hasła.

[2] NetWare Directory Services, potem zwane Novell Directory Services, a teraz znane jako eDirectory

Przypadki użycia

1. Zmiana hasła (przez użytkownika)
   wejście Pytanie o aktualne login i hasło, pytanie o nowe hasło. Wybór systemów, na których użytkownik chce zmienić hasło.
   wyjście Komunikat o udanej zmianie hasła lub komunikat błędu.
2. Podpowiedź hasła
   wejście Brak
   wyjście Hasło wygenerowane przez system, które spełnia wymagania polityki bezpieczeństwa.
3. Zalogowanie się do systemu (przez obsługę)
   wejście Pytanie o aktualne login i hasło.
   wyjście Formularz początkowy do zmian haseł innym użytkownikom.
4. Wyszukanie użytkownika (przez obsługę)
   wejście Pytanie o login lub nazwisko i imię.
   wyjście Tabela z listą użytkowników, ich imiona, nazwiska, aliasy pocztowe, typ użytkownika, ewentualnie zdjęcie, graficzna reprezentacja zajętości miejsca (quota), miniformularze do zmiany hasła.
5. Zmiana hasła (przez obsługę)
   wejście Login (pole ukryte, obsługa musi tylko wybrać odpowiednią linię w tabeli, nie trzeba nic wpisywać), lista systemów, na których należy zmienić hasło (checkbox).
   wyjście Strona z loginem i hasłem wygenerowanym przez system.

4.2. Wymagania niefunkcjonalne

— Wymagania dotyczące sprzętu: korzystamy z istniejącej infrastruktury, system będzie bardzo niewielki.
— Wymagania dotyczące przepustowości łącz: ze względu na usługowy i bardzo prosty charakter systemu, nie powinno być problemów z korzystaniem z systemu na dowolnym łączu.
— Wymagania dotyczące technologii: język programowania PHP, bazy danych z użytkownikami LDAP, baza danych na dziennik systemu PostgreSQL, serwer aplikacji: Apache z modułem PHP
— Dane będą pobierane z serwera LDAP (ldap.sgh.waw.pl); dane będą zapisywane na serwerach: LDAP (ldap.sgh.waw.pl), Active Directory (aqua.sgh.waw.pl) oraz eDirectory (hikari.sgh.waw.pl). Dodatkowo będzie tworzony dziennik systemu w bazie danych PostgreSQL. Nie przewiduje się żadnych importów ani eksportów danych.
— Ogólna charakterystyka systemu: cienki klient (WWW), aplikacja na serwerze.

5. Model danych

System w zasadzie nie korzysta z własnych danych, tylko manipuluje danymi zawartymi w innych systemach. W związku z tym zostaną opisane tylko fragmenty wykorzystywanych encji.

5.1. Dane użytkownika

Jako podstawowe źródło danych został uznany serwer LDAP, w którym uwierzytelniają się użytkownicy UNIX-a. Przykład danych, z których korzystamy:

```
dn: uid=jagol,ou=People,dc=sgh,dc=waw,dc=pl
userPassword: {crypt}jakieśhasło
shadowMax: 60
shadowWarning: 7
shadowLastChange: 13479
personalTitle: prof. dr hab.
givenname: Jan
sn: GOLIŃSKI
employeetype: Pracownik dydaktyczny
departmentNumber: cn=Profesor zwyczajny@ou=Zakład Projektowania Systemów,
 ou=Katedra Informatyki Gospodarczej,ou=Kolegium Analiz Ekonomicznych,ou=SGH
departmentNumber: cn=Kierownik@ou=Katedra Informatyki Gospodarczej,
 ou=Kolegium Analiz Ekonomicznych,ou=SGH
```

Na pozostałych systemach tylko zmieniamy hasło, więc korzystamy z bardzo ograniczonego zestawu danych, których nawet nie czytamy, tylko od razu zapisujemy.

```
dn: cn=jagol,ou=USERS,o=SGH
userPassword: jakieśhasło                          Netware
passwordExpirationTime: 19920102000000Z

dn: CN=jagol,CN=Users,DC=nt,DC=sgh,DC=waw,DC=pl
pwdLastSet: 128133316194375000                     Windows
unicodePwd: "jakieśhasło"
```

5.2. Dziennik wydarzeń

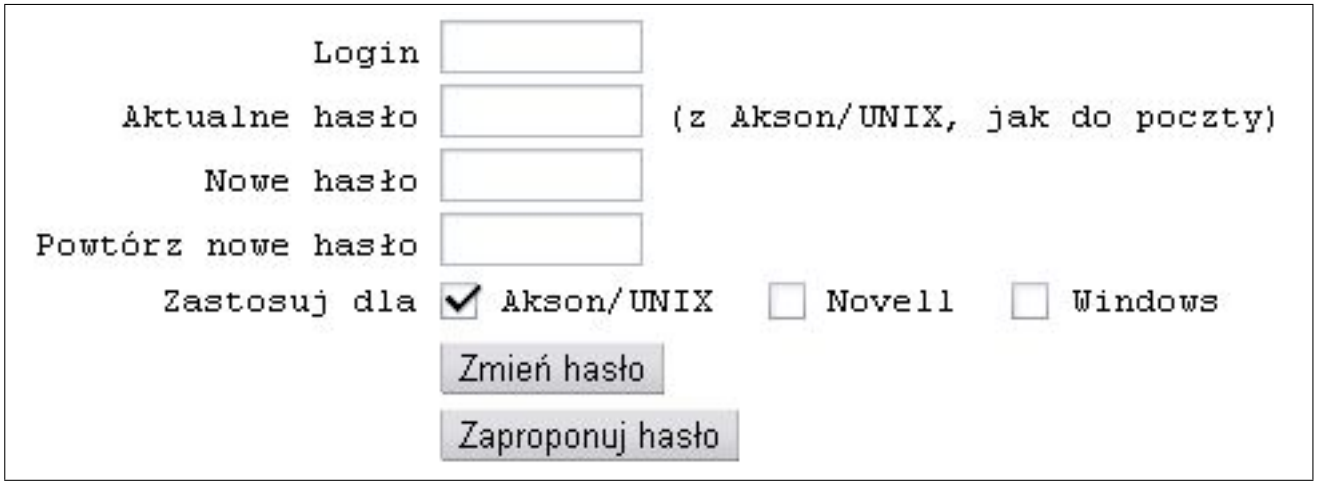
Dziennik wydarzeń trzymany jest w postaci pojedynczej, niepowiązanej tabeli w bazie SQL. Opis pól tabeli:

kto identyfikator obsługującego, który przeprowadził transakcję (text)
kiedy data transakcji w formacie UNIX timestamp (integer)
komu identyfikator użytkownika, któremu zmieniono hasło
proba lista (mapa bitowa: 1 – UNIX, 2 – Windows, 4 – Novell) systemów, na których próbowano zmienić hasło
bledy lista (mapa bitowa, jw.) systemów, na których nie udało się zmienić hasła

6. Projekt interfejsu użytkownika

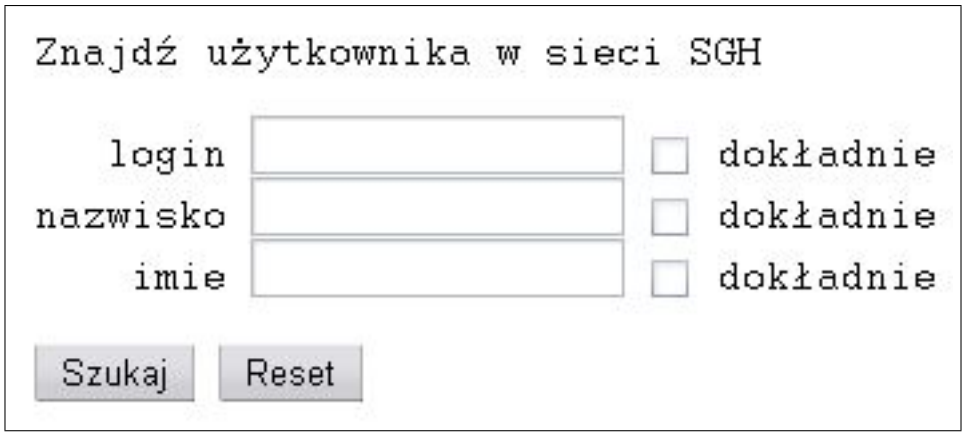
Ze względu na prostotę obsługi dla użytkowników najlepiej będzie, żeby system dla obsługi w ogóle nie był przez nich widziany – pomimo że wewnątrz to są te same funkcje i interfejsy.

Użytkownicy dostają maksymalnie prosty interfejs (rys. 1) – tylko wymagane dane, kilka podpowiedzi, co gdzie wpisać. Dodatkowo, jeżeli użytkownik już był zalogowany w systemie, nie będą pokazywane pierwsze dwa pytania – w końcu jego login i hasło już znamy. Przy wyborze propozycji hasła pojawi się ono obok nowego hasła, użytkownik będzie musiał je przepisać dwukrotnie (chodzi o wyeliminowanie pomyłek przy wpisywaniu).

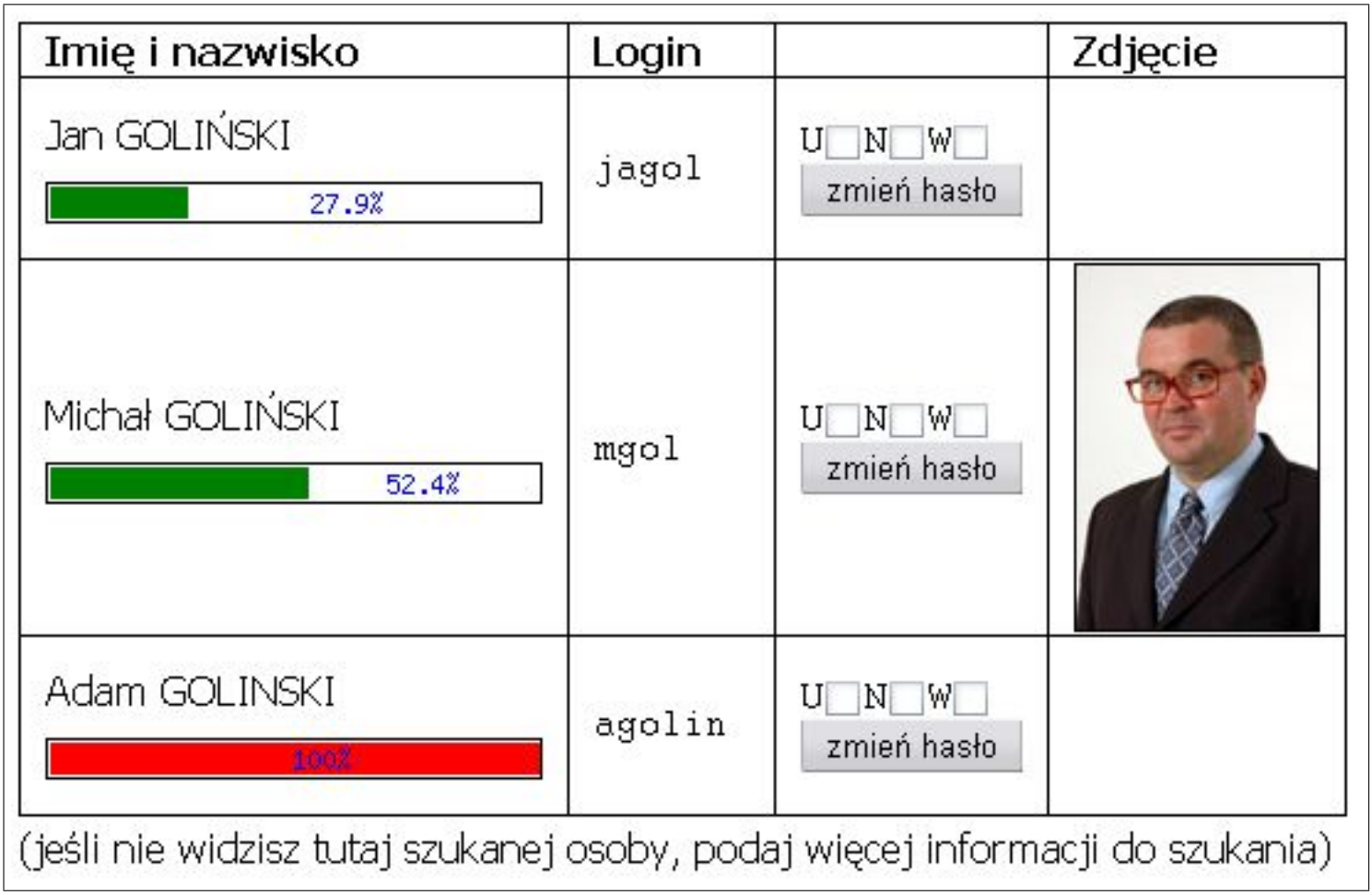


Rysunek 1. Ekran dialogowy do zmiany hasła przez użytkownika

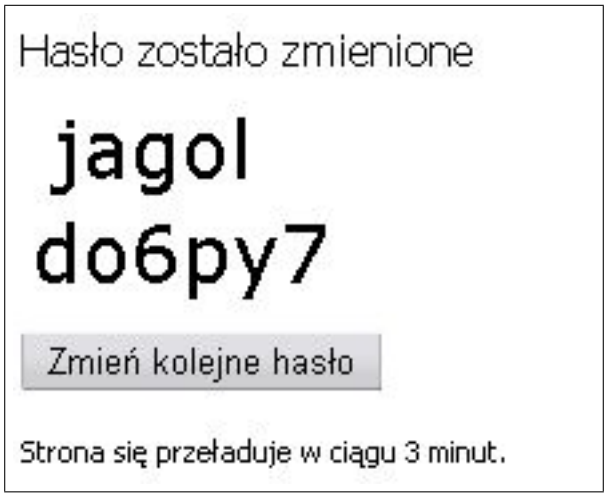
Obsługa musi najpierw uwierzytelnić się swoimi loginem i hasłem, żeby otrzymać podstawowy interfejs. W kroku pierwszym (rys. 2) obsługa podaje informacje o użytkowniku do wyszukania spośród innych, w kroku drugim (rys. 3) wybiera odpowiedniego użytkownika, zaznacza, na którym systemie należy mu zmienić hasło, i wysyła żądanie zmiany. Na końcu otrzymuje (rys. 4) wygenerowane hasło, które jest przepisywane na karteczkę i wręczane użytkownikowi, który jest zobowiązany jak najszybciej zmienić to hasło na własne.



Rysunek 2. Krok 1: szukanie użytkownika



Rysunek 3. Lista użytkowników



Rysunek 4. Wygenerowane hasło

7. Analiza kosztów

Projekt jest niewielki i raczej prosty, tworzony w modelu buduj-i-poprawiaj; szacowana wielkość kodu nie powinna przekroczyć 2000 linii.

7.1. Modele algorytmiczne oparte o KLOC

Algorytmiczne oszacowanie nakładów według modeli zależnych od wielkości kodu:

$$\begin{aligned} \text{Watson \& Felix} &\quad E = 5.2 \cdot \text{KLOC}^{0.91} \approx 9.77 \\ \text{Bailey-Basili} &\quad E = 5.5 + 0.73 \cdot \text{KLOC}^{1.16} \approx 7.13 \\ \text{Boehm} &\quad E = 3.2 \cdot \text{KLOC}^{1.05} \approx 6.63 \\ \text{Doty} &\quad \text{ma zastosowanie dla KLOC} > 9 \end{aligned}$$

7.2. Modele algorytmiczne oparte o FP

Szacunki dotyczące punktów funkcyjnych:

	złożoność komponentu		
komponent	proste	przeciętne	złożone
wejścia	1	2	0
wyjścia	4	0	0
zapytania	4	1	0
pliki główne	3	0	0
interfejsy	4	0	0

Tabela 1. Punkty funkcyjne

Suma ważona[3] niezmodyfikowanych punktów funkcyjnych:

$$\begin{aligned} \text{UFP} = {} & 1 \cdot 3 + 2 \cdot 4 + 0 \cdot 6 + 4 \cdot 4 + 0 \cdot 5 + 0 \cdot 7 \\ & + 4 \cdot 3 + 1 \cdot 4 + 0 \cdot 6 + 3 \cdot 7 + 0 \cdot 10 + 0 \cdot 15 \\ & + 4 \cdot 5 + 0 \cdot 7 + 0 \cdot 10 = 84 \end{aligned}$$

W tabeli 2 został podsumowany wpływ 14 czynników technicznych, co pozwala nam obliczyć zmodyfikowaną wielkość punktów funkcyjnych:

$$\text{FP} = (0,65 + 0,01 \cdot 40) \cdot \text{UFP} = 88.2$$

Algorytmiczne oszacowanie nakładów według modeli opartych na punktach funkcyjnych:

$$\begin{aligned} \text{Albrecht \& Gafney} &\quad E = -13.39 + 0.0545 \cdot \text{FP} \approx -8,59 \\ \text{Kemerer} &\quad E = 60.62 \cdot 7.728 \cdot 10^{-8} \cdot \text{FP}^3 \approx 3,19 \\ \text{Matson, Barnett \& Mellichamp} &\quad E = 585.7 + 15.12 \cdot \text{FP} \approx 1916,26 \end{aligned}$$

Z powyższych oszacowań najbliższy rzeczywistości okazał się model Kemerera.

[3] wagi zgodnie z zaleceniami International Function Point User Group (IFPUG)

czynniki techniczne	stopień wpływu (0-5)
szybkość przesyłania danych	3
przetwarzanie rozproszone	0
częstotliwość wykonywania transakcji	3
kryteria wydajnościowe	3
stopień obciążenia sprzętu	2
wprowadzanie danych online	5
umiejętności użytkownika	3
natychmiastowa aktualizacja	5
złożoność wykonywanych obliczeń	1
reusability	3
łatwość instalacji	1
łatwość obsługi	5
przenośność	3
konserwowalność systemu	3
$\sum$	40

Tabela 2. Czynniki techniczne (DI, degree of influence)

7.3. Podstawowy model COCOMO

Oszacowanie według podstawowego modelu COCOMO dla małego zespołu i znanego środowiska, innymi słowy dla organicznej klasy przedsięwzięcia:

$$E = 2.4 \cdot \mathrm{KLOC}^{1.05} = 4.9693 \approx 5$$
$$D = 2.5 \cdot E^{0.38} = 4.6084 \approx 4.61$$
$$P = E/D = 1.0845 \approx 1$$

gdzie E jest wyrażone w osobomiesiącach, D to czas rozwijania oprogramowania w miesiącach, a P – wymagana do projektu liczba osób. Przy założeniu średniej płacy w branży IT na poziomie 3500 zł miesięcznie[4], przybliżony koszt systemu wynosi:

$$C = 5 \cdot 3500 = 17500\,\mathrm{zł}$$

Kwota ta może wydawać się duża, ale należy pamiętać, że obejmuje koszt projektowania, kodowania, wdrożenia, testowania i tworzenia dokumentacji (zarówno użytkownika, jak i programowej).

Przy dodatkowym narzucie kosztów korporacyjnych (pomieszczenia, wyposażenie, księgowość itd.) ostatecznie szacowany koszt systemu wyniósłby:

$$C_K = 2.40 \cdot 17500 = 42000\,\mathrm{zł}$$

[4] Źródło: Internetowe Badanie Wynagrodzeń za rok 2006.

7.4. Pośredni model COCOMO

W pośrednim modelu COCOMO dla organicznej klasy przedsięwzięcia mamy wzór:

$$E = 3.2 \cdot \mathrm{KLOC}^{1.05} \cdot \prod_{i=1}^{15} a_i$$

gdzie a_i to współczynniki nośników kosztów z tabeli 3.

Nośniki kosztów		Ocena	Współczynnik
Atrybuty produktu			
RELY	wymagana niezawodność	przeciętna	1.00
DATA	wielkość bazy danych	przeciętna	1.00
CPLX	złożoność produktu	przeciętna	1.00
Atrybuty komputera			
TIME	ograniczenia czasu wykonania	przeciętne	1.00
STOR	ograniczenia pamięci operacyjnej	przeciętne	1.00
VIRT	niestabilność pamięci wirtualnej	mała	0.87
TURN	czas obrotu zadania	przeciętny	1.00
Atrybuty personelu			
ACAP	możliwości analityków	małe	1.19
AEPX	doświadczenia w danym typie aplikacji	przeciętne	1.00
PCAP	możliwości programistów	przeciętne	1.00
VEXP	doświadczenie w stosowaniu pamięci wirtualnej	bardzo małe	1.21
LEXP	doświadczenie w danym języku programowania	wysokie	0.95
Atrybuty przedsięwzięcia			
MODP	nowoczesne praktyki programistyczne	brak	1.24
TOOL	zastosowanie narzędzi programowych	bardzo małe	1.24
SCHED	ograniczenia w czasie wytwarzania	bardzo małe	1.23
		$\prod$	2.25

Tabela 3. Nośniki kosztów modelu COCOMO

Czas rozwijania i osobochłonność liczy się tak samo jak w modelu podstawowym; ostatecznie mamy:

$$E = 3.2 \cdot 2^{1.05} \cdot 2.25 = 14.907 \approx 15\,\text{osobomiesięcy}$$
$$D = 2.5 \cdot E^{0.38} \approx 7\,\text{miesięcy}$$
$$P = E/D \approx 2\,\text{osoby}$$

Wygląda na to, że zbyt pesymistycznie oceniłem jakość pracy i doświadczenie analityków i programistów; jeśli przyjąć iloczyn współczynników równy jeden, otrzymujemy $E = 6.62$.

8. Podsumowanie

Wszystkie fazy budowania systemu łącznie z programowaniem przeprowadziłem samodzielnie, a powyższy opis jest próbą podsumowania i oszacowania kosztu takiego projektu już post factum. Ponieważ podczas tworzenia tego systemu powstawały potrzeby realizacji innych projektów pomocniczych (np. aplikacja serwująca zdjęcia w zależności od ustawień użytkownika) oraz ze względu na wieloetapowość budowania-i-poprawiania (np. moduł zmieniający hasła na Windows został dodany bardzo późno), ogólny czas napisania i wdrożenia tego systemu wynosił około pół roku, z czego projektowanie i kodowanie trwało ok. miesiąca.

Można zwrócić uwagę, że czas realizacji pokrywa się z oszacowaniem w modelu COCOMO. O ile całkowity szacunek zgadza się tylko przypadkiem, o tyle w części dotyczącej kodowania, jeśli uwzględnić specyficzny charakter projektu (nie miał on dodatkowych kosztów korporacyjnych, nie ma dokumentacji, zaś projektowanie i wdrożenie były przeprowadzane jednocześnie z kodowaniem), zgadza się wręcz idealnie. Według Boehma[5] dla małego projektu (rzędu 2KLOC) samo kodowanie i testy modułów to 42% wyliczanych wartości. Nie podaje on niestety, ile z tego przypada na testy modułów, ale przy założeniu, że około połowa, to część przeznaczoną na kodowanie razem z poprawianiem błędów można przyjąć ∼20%, co ostatecznie daje wyżej wspomniany $5 * 20\% = 1$ osobomiesiąc.

Organizacja (SGH) nie poniosła w związku z tworzeniem tego systemu żadnych kosztów.

Z systemu można skorzystać pod adresami:
`https://akson.sgh.waw.pl/passwd/` (od tamtej pory został znacznie rozbudowany i przekształcił się w aplikację do zarządzania kontem)
`https://akson.sgh.waw.pl/chpass/` (dostęp tylko dla obsługi)

[5] Barry Boehm, *Software Engineering Economics*, str. 65, tabela 5-2.