

# IO - Plan testów

M.Jałmużna

T.Jurkiewicz

P.Kasprzyk

M.Robak

5 czerwca 2006

## Spis treści

<b>1</b>	<b>Historia zmian</b>	<b>3</b>
<b>2</b>	<b>Zakres testów</b>	<b>3</b>
2.1	Integration testing - Testy spójności . . . . .	3
2.2	Facility testing - Spełnienie założeń . . . . .	3
2.3	Volume testing - Testowanie objętościowe . . . . .	3
2.4	Stress testing - Testowanie obciążeniowe . . . . .	4
2.5	Endurance testing - Testy wytrzymałościowe . . . . .	4
2.6	Usability testing - Łatwość użycia . . . . .	4
2.7	Security testing - Testy bezpieczeństwa . . . . .	4
2.8	Performance testing - Testy wydajności . . . . .	5
2.9	Storage testing - Testowanie składowania danych . . . . .	5
2.10	Configuration testing - Testowanie konfiguracji . . . . .	5
2.11	installability testing - Testowanie instalacyjności . . . . .	6
2.12	Reliability testing - Testowanie niezawodności . . . . .	6
2.13	Recovery testing - Testowanie odporności . . . . .	6
2.14	Documentation testing - Testowanie dokumentacji . . . . .	6
<b>3</b>	<b>Zasoby potrzebne do testów</b>	<b>7</b>
3.1	Sprzęt i oprogramowanie . . . . .	7
<b>4</b>	<b>Plan testów</b>	<b>7</b>

## 1 Historia zmian

Wersja	Data	Opis	Autor
0.5	27.05.2006	Wzór dokumentu	
0.9	31.05.2006	Przygotowanie wstępnej wersji	
0.99	1.06.2006	Przygotowanie (prawie) ostatecznej wersji	

## 2 Zakres testów

### 2.1 Integration testing - Testy spójności

- Przeprowadzenie testów na spójność komponentów
- Wykorzystanie wszystkich składowych systemów jednocześnie

Test ma być wykonany na wszystkich komponentach aplikacji, po wykonaniu każdej iteracji oraz po wykonaniu całego systemu.

### 2.2 Facility testing - Spełnienie założeń

- Zbudowanie schematu uruchomienia wszystkich funkcji systemu opisanych w dokumentacji
- Opisanie sposobu działania każdej funkcji
- Przetestowanie każdej funkcji systemu zgodnie z planem

Test ma być wykonany na każdym komponencie systemu pod kątem zgodności z dokumentacją.

### 2.3 Volume testing - Testowanie objętościowe

- Generowanie dużej ilości danych, w szczególności
  - Wygenerowanie dużej liczby działów, wątków i postów na forum
  - Wygenerowanie dużej ilości użytkowników systemu
  - Wygenerowanie dużej bazy rozwiązanych problemów
  - Wygenerowanie dużej liczby materiałów dydaktycznych

Testowanie objętościowe dotyczy bazy danych aplikacji. Wymaga się przeprowadzenie tego testu po każdej iteracji, oraz po zakończeniu fazy projektowania aplikacji.

## 2.4 Stress testing - Testowanie obciążeniowe

- Symulacja korzystania z systemu przez dużą liczbę użytkowników
  - Jednoczesna edycja, dodawanie postów na forum
  - Jednoczesne korzystanie z chatu przez wiele osób
  - Jednoczesna próba pobrania tego samego zasobu
  - Próba logowania wszystkich użytkowników w jednym momencie

Test obciążeniowy dotyczy modułu bazy danych aplikacji, oraz modułu chat. Moduły mają być testowane po każdej iteracji, przy czym decydujący wpływ ma test po zakończeniu fazy implementacji, gdyż nie zakłada się wysokiej wydajności aplikacji w pierwszych fazach jej budowy.

## 2.5 Endurance testing - Testy wytrzymałościowe

Uruchomienie skryptów testujących system, wykonujących ciągle operacje (co pozwoli na symulację długookresowego działania systemu w krótkim okresie czasu rzeczywistego), typowe dla korzystania z podobnych systemów.

Testom wytrzymałościowym musi być poddana baza danych aplikacji po każdej iteracji (w ramach udostępnianych funkcjonalności), oraz baza danych w połączeniu z resztą komponentów po zakończeniu fazy projektowania aplikacji.

## 2.6 Usability testing - Łatwość użycia

Zmierzenie czasu potrzebnego do wykonania podstawowych operacji, wyselekcjonowanych na podstawie ankiet. Wydajność systemu będzie mierzona czasem potrzebnym do wykonania sekwencji operacji przez użytkowników:

- zaznajomionych z obsługą systemu
- mających pierwszy kontakt z testowaną aplikacją.

Test łatwości użycia dotyczy interfejsu użytkownika powiązanego z modułami chat oraz forum www. Testowane będą wszystkie udostępniane funkcjonalności w ramach każdej iteracji. Testy muszą być przeprowadzone przez użytkowników znających aplikację, a także przez nowych użytkowników, pod kątem wykonania zleconych operacji.

## 2.7 Security testing - Testy bezpieczeństwa

Sprawdzenie odporności aplikacji na próby:

- Włamania do bazy danych metodą SQL-injection

- Przeciżenia aplikacji jednoczesnym pobieraniem i wysyłaniem ponadnaturalnej ilości danych
- Włamania do bazy danych poprzez próbę uzyskania nieuprawnionego dostępu

Wykazanie błędów w działaniu aplikacji pod tym kątem zostanie uznane jako błąd krytyczny. Testowany ma być dostęp do bazy danych przez udostępniane interfejsy użytkownika.

## 2.8 Performance testing - Testy wydajności

Czas reakcji systemu będzie mierzony czasem, jaki upłynął od zadania zapytania do bazy danych aplikacji, lub czasem, jaki upływa między kolejnymi wypowiedziami na chacie aplikacji. Testy wydajnościowe muszą być powiązane z testami łatwości użycia aplikacji. Mają one być wykonywane po każdej fazie iteracji, dzięki czemu uzyskana zostanie informacja o tym, które elementy aplikacji należy poprawić.

## 2.9 Storage testing - Testowanie składowania danych

Należy przeprowadzić testy związane z odpornością bazy danych aplikacji na przeciążenie zbyt dużą ilością danych. W tym celu przeprowadzi się testy (za pomocą skryptów testujących), które będą zapisywać duże ilości danych każdego typu. Test należy przeprowadzać na bazie danych po zakończeniu każdej fazy iteracji aplikacji.

## 2.10 Configuration testing - Testowanie konfiguracji

Założeniem projektowym aplikacji jest prawidłowe działanie aplikacji na komputerach klienckich (niekoniecznie identycznych). Należy zapewnić prawidłowe działanie:

- interfejsu WWW
- chata
- platform Java

Przetestowanie Interfejsu WWW obejmować będzie:

- Sprawdzenie zgodności wszystkich podstron ze standardem W3 (Xhtml 1.0 / CSS)
- Wykonanie zrzutów ekranowych wyrenderowanych podstron każdego typu na każdej założonej przeglądarce, a następnie porównanie zrzutów
- Odporność interfejsu WWW na błędy użytkownika - wybieranie podstron w niewłaściwej kolejności, próby manipulacji adresem URI

Przetestowanie chatu oraz platformy Java zakłada przetestowanie wszystkich funkcji komunikatora na każdej konfiguracji (OS x Platforma Java)

### **2.11 installability testing - Testowanie instalacyjności**

Próba instalacji serwera aplikacji oraz aplikacji klienckich na dostępnych konfiguracjach sprzętowych.

### **2.12 Reliability testing - Testowanie niezawodności**

Jedyną metodą sprawdzenia niezawodności systemu w czasie jest jego próbne uruchomienie wraz ze skryptami testującymi symulującymi długotrwałe działanie systemu.

### **2.13 Recovery testing - Testowanie odporności**

Baza danych systemu powinna być odporna na awarie. Stąd niezbędne jest dokonywanie zautomatyzowanych czynności archiwizujących. Sama aplikacja powinna być odporna na wszelkie awarie związane z infrastrukturą. W przypadku awarii bazy danych, wszystkie zarchiwizowane dane powinny zostać odtworzone.

- Próbne awaryjne wyłączenie systemu w różnych stanach
- Podnoszenie bazy danych

Stwierdzenie braku możliwości przywrócenia stanu bazy danych do zarchiwizowanego zostanie zakwalifikowane jako usterka krytyczna.

### **2.14 Documentation testing - Testowanie dokumentacji**

Niebędne jest przetestowanie dokumentacji i aplikacji pod kątem wzajemnej zgodności. Aplikacja musi posiadać zaimplementowane wszystkie funkcje opisane w dokumentacji projektowej. Wszystkie zaimplementowane funkcje muszą działać według specyfikacji opisanej w dokumentacji projektowej.

- Sprawdzenie zaimplementowania wszystkich funkcji aplikacji według listy
- Sprawdzenie działania każdej funkcji aplikacji według dokumentacji

### 3 Zasoby potrzebne do testów

#### 3.1 Sprzęt i oprogramowanie

Typ	Opis	Dlaczego niezbędne
PC	Komputer klasy PC spełniający standardowe wymagania sprzętowe z zainstalowanymi systemami: Vista, Windows XP, Windows ME, Windows 98, Linux (dowolna dystrybucja na jądrze 2.6.x).	Pomimo niezależności platformy JAVA od systemu operacyjnego należy sprawdzić wygląd i zachowanie aplikacji pod dwoma najpopularniejszymi środowiskami.
PC	Komputer klasy PC o minimalnej specyfikacji sprzętowej	Niezbędne jest sprawdzenie ewentualnych problemów wynikających z małej wydajności końcówki roboczej
Serwer	Serwer o standardowych parametrach, najlepiej z dwoma systemami (Linux, Windows)	Rozsądne byłoby oddzielenie środowiska testowego od produkcyjnego
Przeglądarki internetowe	Darmowe wersje najpopularniejszych przeglądarek internetowych: Mozilli Firefox, Internet Explorera (najlepiej w różnych wersjach np 5 i 6), Opera, Konqueror, Safari)	Niezbędne aby przetestować i zapewnić w ten sposób prawidłowe działanie pod wieloma przeglądarkami

### 4 Plan testów

Testy będą planowane przed wykonaniem każdej iteracji budowy aplikacji. Testy będą obejmować sprawdzenie każdego modułu z osobna, oraz przetestowanie spójności modułów po zakończeniu iteracji. Kompletna aplikacja zostanie poddana całościowym testom po zakończeniu fazy budowy aplikacji.

Lista zadań do wykonania w trakcie testowania:

- Wykonanie zestawu testów do modułów pierwszej iteracji
- Przeprowadzenie testów dla każdego zakończonego modułu w pierwszej iteracji iteracji
- Przeprowadzenie zestawu testów dla aplikacji po wykonaniu iteracji

- Wykonanie zestawu testów do modułów drugiej iteracji
- Przeprowadzenie testów dla każdego zakończonego modułu w drugiej iteracji iteracji
- Przeprowadzenie zestawu testów dla aplikacji po wykonaniu iteracji
- Wykonanie zestawu testów do modułów pierwszej iteracji
- Przeprowadzenie testów dla każdego zakończonego modułu w pierwszej iteracji iteracji
- Przeprowadzenie zestawu testów dla aplikacji po wykonaniu iteracji
- Przeprowadzenie testów dla całej aplikacji