

GEM — Plan zatwierdzania projektu

wersja 1.0

Maria Donten
Marek Grabowski
Piotr Hofman
Kuba Pochrybniak

Spis treści

1. Wprowadzenie	3
1.1. Cel	3
1.2. Zakres	3
1.3. Definicje i skróty	3
1.4. Załączniki	3
1.5. Omówienie reszty dokumentu	3
2. Zakres odpowiedzialności	3
3. Kroki odbioru projektu	4
3.1. Kryteria odbioru	4
3.2. Wstępny audyt konfiguracji	4
3.3. Audyt funkcjonalności komponentów	4
3.3.1. Edytor graficzny	4
3.3.2. Edytor tekstowy	4
3.3.3. Przeglądarka	5
3.3.4. Kompilator i konwerter	5
3.3.5. Moduł sieciowy	5
3.4. Plan	5
4. Potrzebne zasoby	6
4.1. Sprzęt	6
4.2. Oprogramowanie	6
4.2.1. Potrzebne	6
4.3. Dokumentacja	6
4.4. Personel	6
4.5. Dane testowe	6
5. Rozpoznawanie problemów i reakcja na problemy	7
6. Środowisko odbioru produktu	8
7. Wymagane kroki kontroli	8
7.1. Edytor graficzny	8
7.2. Edytor tekstowy	8
7.3. Przeglądarka	8
7.4. Kompilator i konwerter	9
7.5. Moduł sieciowy	9
7.6. Dokumentacja	9
8. Historia zmian	9

1. Wprowadzenie

1.1. Cel

Celem tego dokumentu jest rozplanowanie przebiegu procesu akceptacji programu GEM oraz sprecyzowanie warunków określających, kiedy projekt może zostać uznany za zakończony z sukcesem.

1.2. Zakres

Projekt GEM jest wykonywany w ramach zajęć Zespołowego Projektu Programistycznego na wydziale MIM UW. Proces akceptacji programu GEM jest podporządkowany celowi tych zajęć, czyli przygotowaniu pracy licencjackiej z informatyki — program i jego dokumentacja muszą spełniać wymagania pracy licencjackiej.

1.3. Definicje i skróty

Podane w załączniku „Słownik projektu GEM”, uaktualnianym na bieżąco, aby zawierał definicje wymagane przez powstające kolejno dokumenty.

1.4. Załączniki

— Dokument „Słownik projektu GEM”, wersja 4.1.

1.5. Omówienie reszty dokumentu

Rozdział drugi dokumentu określa, za jakie elementy procesu akceptacji odpowiedzialne są poszczególne osoby w nim uczestniczące. Rozdział trzeci zawiera plan przebiegu procesu akceptacji, a także precyzuje, na jakie właściwości programu należy zwrócić szczególną uwagę podczas prowadzenia testów. Bardziej szczegółowe opracowanie testów znajduje się w rozdziale siódmym. Rozdziały czwarty i szósty prezentują potrzeby dotyczące zasobów oraz środowiska przeprowadzania procesu akceptacji. Natomiast rozdział piąty zawiera plany procedur postępowania w przypadku wystąpienia błędów podczas testów programu.

2. Zakres odpowiedzialności

Zespół programistyczny Drużyna A jest odpowiedzialny za następujące kwestie:

1. implementację programu GEM;
2. przygotowanie i przedstawienie kompletnej dokumentacji projektu GEM;
3. przygotowanie planu testów, danych i programów koniecznych do testowania programu GEM;
4. przeprowadzenie testów wewnętrznych (prowadzą je członkowie Zespołu) przy obu wydaniach;
5. przygotowanie sprawozdań z przebiegu testów;
6. przedyskutowanie z prowadzącą zajęcia panią Agatą Janowską wszystkich poważnych błędów, które wystąpią podczas testowania programów;
7. znalezienie kilku (przynajmniej trzech) osób spoza Zespołu, które zgodzą się testować program jako użytkownicy, nie według scenariuszy testów (przy drugim wydaniu);
8. przygotowanie prezentacji programu GEM.

Prowadząca zajęcia pani Agata Janowska jest odpowiedzialna za:

1. dostarczenie jasnych kryteriów oceny projektu;
2. zapoznanie się z dostarczoną przez Zespół dokumentacją;
3. rozmowę z członkami Zespołu w przypadku wystąpienia sytuacji kryzysowej dla projektu;
4. dokonanie oceny projektu na podstawie dostarczonego programu, dokumentacji i prezentacji przygotowanej przez Zespół na zakończenie zajęć.

3. Kroki odbioru projektu

Przed ukazaniem się w ostatecznej wersji, program zostanie poddany intensywnym testom. Poniżej znajduje się ramowy opis testów.

3.1. Kryteria odbioru

Program będziemy testować pod wieloma kątami. Najważniejsze rzeczy, które będziemy brać pod uwagę, to:

- zgodność ze specyfikacją;
- niezawodność i reagowanie na sytuacje wyjątkowe;
- szybkość;
- ergonomia i prostota obsługi.

3.2. Wstępny audyt konfiguracji

Produkt końcowy będzie składał się z następujących elementów:

1. Program:
 - a) edytor graficzny,
 - b) edytor tekstowy,
 - c) przeglądarka,
 - d) kompilator i konwerter,
 - e) moduł sieciowy.
2. Dokumentacja:
 - a) komplet dokumentów technicznych,
 - b) podręcznik użytkownika.

3.3. Audyt funkcjonalności komponentów

Każda z wymienionych w punkcie 3.2 składowych programu będzie dokładnie testowana.

3.3.1. Edytor graficzny

Testowany będzie pod wieloma kątami:

- odporność na tworzenie dużej liczby obiektów graficznych;
- odporność na skomplikowane zależności dotyczące względnego położenia obiektów;
- funkcjonalność przy dużym zagęszczeniu obiektów;
- podobieństwo między szkicowym rysunkiem w edytorze a efektem w przeglądarce.

3.3.2. Edytor tekstowy

Jako dużo mniej złożony komponent, będzie testowany mniej intensywnie niż edytor graficzny. Weźmiemy pod uwagę głównie odporność na bardzo duże pliki tekstowe.

3.3.3. Przeglądarka

Testując przeglądarkę, szczególną uwagę zwrócimy na:

- odporność na duże, skomplikowane rysunki;
- jakość wyświetlanych rysunków;
- szybkość odświeżania obrazu;
- wygodę skalowania i przesuwania.

3.3.4. Kompilator i konwerter

Przy testowaniu kompilatora pod uwagę weźmiemy przede wszystkim reagowanie na błędy. Kompilator będzie korzystał z zewnętrznego programu, który jest przez środowisko informatyczne uznawany za pozbawiony błędów.

Przy konwerterze zwrócimy uwagę przede wszystkim na poprawność konwersji: obrazek wykonany w edytorze graficznym musi być przekształcony na pozbawiony błędów kod. Testować będziemy go zarówno na małych, jak i dużych plikach.

3.3.5. Moduł sieciowy

Testować moduł sieciowy będziemy pod kątem:

- odporności na połączenia o bardzo niskiej przepustowości;
- odporności na okresowe przerywanie połączenia;
- szybkości odświeżania ekranu;
- odporności na dużą liczbę zmian w jednostce czasu.

3.4. Plan

Zespół planuje dwie fazy testów: przed wydaniem wersji β oraz 1.0. Testy zaczną się odpowiednio 10 kwietnia oraz 14 maja 2007.

Oprócz członków Zespołu testować będzie grupa wybranych wcześniej osób, mających różne doświadczenie w tworzeniu ilustracji naukowych.

Każda z obu faz będzie wyglądała następująco:

1. Dwa tygodnie przed wydaniem program zostanie udostępniony grupie wybranych testerów. System będzie testowany na różnych systemach (na Windows i Linuksie), na sprzęcie o zróżnicowanych parametrach.
2. Testerzy będą mieli tydzień na przetestowanie programu. W czasie tego tygodnia Zespół będzie w kontakcie ze wszystkimi testującymi, aby na bieżąco wyłapywać usterki oraz by służyć pomocą w wątpliwych sytuacjach. To drugie będzie ważne zwłaszcza przed pierwszym wydaniem, kiedy duży nacisk będzie położony na testowanie ergonometrii i intuicyjności użytkownika edytora.
3. Każdy z testujących będzie przypisany do konkretnego członka Zespołu, któremu po danej fazie testów przekaże wszystkie swoje uwagi.
4. Na tydzień przed wydaniem zakończą się testy. Podczas spotkania Zespół omówi ujawnione przez testy problemy oraz sposoby ich rozwiązania.
5. Produkt nie powinien być zmieniany przez ostatnie dwa dni przed prezentacją.
6. Przy okazji prezentacji Zespół zbierze jak najwięcej, nie tylko konstruktywnych, uwag, które rozważy w dalszych pracach nad projektem.

Jeżeli Zespół oraz prowadząca zajęcia uznają, że produkt w wersji 1.0 spełnia oczekiwania i jest funkcjonalnym i wygodnym narzędziem, program zostanie rozreklamowany w środowisku naukowym oraz „około \TeX owym” (np. na grupach dyskusyjnych). W takim wypadku powstanie specjalna strona internetowa projektu, razem z kontem e-mailowym,

na które użytkownicy będą mogli przysyłać swoje uwagi, a projekt będzie dalej rozwijany (choć już nie w ramach zajęć z ZPP).

4. Potrzebne zasoby

4.1. Sprzęt

Projekt wymaga komputera PC z procesorem taktowanym co najmniej 1GHz i 256 MB pamięci RAM. Do używania aplikacji sieciowej wymagane jest szerokopasmowe połączenie do internetu.

4.2. Oprogramowanie

4.2.1. Potrzebne

- Do instalacji i działania systemu potrzebna jest poprawnie zainstalowane i działające:
- wirtualna maszyna Javy — JRE 1.5 lub nowsze,
 - biblioteka Apache-XML-RPC 3.0 lub nowsza,
 - kompilator języka METAPOST,
 - kompilator języka L^AT_EX.

4.3. Dokumentacja

W trakcie pracy nad projektem zespół będzie korzystał z dokumentacji technicznej używanych języków i narzędzi programistycznych. Będzie to między innymi:

- SUN Java API,
- dokumentacja języka L^AT_EX,
- dokumentacja języka METAPOST,
- dokumentacja techniczna konkretnych rozwiązań Javowych (np. dokumentacja Java Swinga).

4.4. Personel

W skład zespołu pracującego nad projektem wchodzi następujące osoby:

- Maria Donten — szefowa zespołu
- Marek Grabowski — członek zespołu
- Piotr Hofman — członek zespołu
- Jakub Pochrybniak — członek zespołu

Osobą odpowiedzialną za specyfikacje projektu i nadzorującą pracę jest Agata Janowska

Ponadto, w trakcie pracy nad projektem, będą uczestniczyć osoby trzecie, wybrane przez Piotra Hofmana, jako testerzy programu.

4.5. Dane testowe

W ramach testów, program zostanie udostępniony osobom trzecim, których zadaniem będzie używanie programu i zgłaszanie wszelkich uwag. Tym samym niemożliwe jest określenie wszystkich cech danych testowych.

Jednakowoż zespół, w ramach testów wewnętrznych, będzie starał się wprowadzić jak najbardziej zróżnicowane dane — zarówno pod względem rozmiaru jak i typu. W związku

z tym wygenerowane zostanie dużo rysunków, zarówno jako kod METAPOST, jak i rysunki zrobione w programie.

Większa część testów odbędzie się w formie udostępnienia wersji beta, dzięki czemu będzie możliwe wykrycie błędów najbardziej uciążliwych dla użytkownika, ale uniemożliwia konkretne zaplanowanie konkretnych danych w fazie projektowania programu.

5. Rozpoznawanie problemów i reakcja na problemy

Jeśli jakiś test zakończy się niepowodzeniem, należy wykryć ogólną przyczynę błędu. Najczęściej spotykane to:

- błąd w kodzie, brak rozpatrzenia jakiegoś przypadku, sekwencji decyzji użytkownika;
- błąd na styku części kodu stworzonych przez dwie osoby;
- błąd w testach;
- błąd koncepcyjny w założeniach programu (takie błędy są najgroźniejsze, powinny zostać ostatecznie wyeliminowane podczas implementacji).

Aby wykryć przyczynę błędu, postępujemy w następujący sposób:

1. Najpierw należy dokładnie sprawdzić program testujący.
2. Gdy zostanie ustalone, że błąd znajduje się w kodzie programu, a nie w testach, cały Zespół jest informowany o wykryciu błędu.
3. Kierowniczka grupy w porozumieniu z resztą Zespołu wyznacza osobę lub grupę osób, które zajmą się znalezieniem, dokładnym zbadaniem i w miarę możliwości poprawieniem błędu. Osoby są wyznaczane w zależności od części programu, w której dany błąd wystąpił — jeśli to możliwe, powinni być to autorzy tej części.
4. Najpierw osoby wyznaczone do zbadania błędu przeglądają i testują funkcje uruchamiane podczas nieprawidłowo wykonanego testu, aby wykryć ewentualne błędy w implementacji poszczególnych metod.
5. Jeśli to nie skutkuje, te osoby analizują ciąg funkcji wywoływanych podczas źle wykonanego testu i badają zależności między nimi. Z dużym prawdopodobieństwem przyczyną błędu jest brak porozumienia co do szczegółów działania i zwracanych wartości funkcji pomiędzy programistami wykonującymi różne elementy programu.
6. Jeśli nadal program nie zostanie poprawiony, następuje ponowna, tym razem drobiazgowa analiza działania poszczególnych funkcji programu i zależności między nimi, aż do skutku. W pracy mogą włączyć się pozostali członkowie Zespołu.
7. W czasie wykonywania powyższych czynności może się okazać że nastąpiły błędy na etapie projektowym. W takim przypadku należy poinformować resztę zespołu, celem ustalenia zmian w specyfikacji, bądź wystąpienia do prowadzącego o zmianę funkcjonalności projektu (w przypadku krytycznego błędu projektowego). Dalsze procedury w tym przypadku zostały opisane w dokumencie „Plan zarządzania projektem”.

Po znalezieniu błędu, w zależności od jego rodzaju, należy podjąć odpowiednią akcję:

1. Dla błędu w testach, należy poinformować o nim innych członków zespołu, poprawić test i wykonać od go początku.
2. Dla błędów małych, wynikających z wadliwego napisania procedury, należy poprawić błędny kod i poinformować o poprawce resztę grupy. Następnie należy wykonać test od początku. A najlepiej powtórzyć wszystkie testy, jeśli starczy czasu.
3. W przypadku błędów wykrytych na styku części pisanych przez różne osoby, należy przedyskutować poprawki, poinformować o błędzie budowy programu resztę zespołu.

Następnie trzeba poprawić błąd, wybrać najważniejsze testy do powtórzenia i wykonać je ponownie. Najlepiej powtórzyć wszystkie.

4. W przypadku błędu projektowego stosuje się procedurę opisaną w dokumencie „Plan zarządzania projektem”.

6. Środowisko odbioru produktu

Do oddania projektu potrzebujemy komputera klasy PC, na którym działa z maszyną wirtualna Javy.

Do prezentacji aplikacji sieciowej potrzebujemy dwóch komputerów spełniających powyższe wymagania, dodatkowo podłączonych do sieci.

7. Wymagane kroki kontroli

7.1. Edytor graficzny

Przed wszystkim przetestowany zostanie interfejs graficzny służący do tworzenia rysunków — na początku przeprowadzone zostaną testy poprawnościowe. Zespół wykona testy oparte na scenariuszach zamieszczonych w dokumencie „Plan testów”. Na powstałych w ten sposób rysunkach zostaną wykonane testy zapisu do plików i odczytu zapisanych rysunków, a także testy tłumaczenia wewnętrznego formatu rysunku na język METAPOST. Te rysunki posłużą również do testowania przeglądarki. Scenariusze testów poprawności będą uwzględniały możliwie skomplikowane zależności między obiektami graficznymi.

Po zakończeniu testów poprawnościowych Zespół przetestuje działanie edytora przy dużym obciążeniu. W tym celu powstanie kilka rysunków zawierających po kilkaset obiektów. Osoby testujące będą obserwowały szybkość reakcji programu, pobieżnie sprawdzając poprawność wyników (dokładne sprawdzenie poprawności mogłoby w tym przypadku zająć bardzo dużo czasu).

Dla dalszego użytkowania systemu bardzo ważne będą testy wykonywane przez osoby spoza Zespołu, które sprawdzą, czy obsługa programu jest wystarczająco intuicyjna i czy w programie nie ma błędów przeoczonych przez Zespół, a także czy program pracuje dobrze przy standardowym obciążeniu wymaganym do stworzenia ilustracji do pracy matematycznej.

7.2. Edytor tekstowy

Jako dużo mniej złożony komponent, będzie testowany mniej intensywnie niż edytor graficzny.

Według scenariuszy z „Planu testów” zostaną sprawdzone podstawowe funkcje edytora: pisanie tekstu, wycinanie, wklejanie, zapisywanie do pliku i wczytywanie z pliku kodu METAPOST.

Ponadto zostanie przetestowane działanie przeglądarki podczas edycji pliku tekstowego.

7.3. Przeglądarka

Podczas testów obu edytorów osoby przeprowadzające testy będą stale kontrolować współpracę przeglądarki z całością programu i zgodność wyświetlanych rysunków z oczekiwaniem. W ten sposób zostanie sprawdzona również wydajność pracy przeglądarki.

Ponadto w „Planie testów” pojawiają się scenariusze, według których będzie przebiegało sprawdzanie takich własności, jak przesuwanie i skalowanie rysunku, oraz koordynacja wyświetlanego obszaru rysunku pomiędzy przeglądarką a edytorem graficznym.

7.4. Kompilator i konwerter

Testy kompilatora i konwertera formatów będą wykorzystywać posiadanych przez Zespół rysunków w kodzie METAPOST. Zespół nie ma wpływu na czas działania procesów kompilacji kodu METAPOST i konwersji formatów, ponieważ te czynności są wykonywane przez zewnętrzne narzędzia dołączone do programu GEM.

7.5. Moduł sieciowy

Działanie modułu sieciowego zostanie sprawdzona przy użyciu scenariuszy zamieszczonych w „Planie testów”. Zespół kładzie szczególny nacisk na poprawność przekazywanych danych oraz na przejrzystość interfejsu tej części programu.

Natomiast nie będzie testowane bezpieczeństwo, ponieważ Zespół nie widzi konieczności zabezpieczania przesyłanych danych (przynajmniej w pierwotnej wersji programu).

7.6. Dokumentacja

Dokumentacja techniczna projektu zostanie dokładnie przeczytana przez cały Zespół. Wszystkie wątpliwości zostaną przedyskutowane.

Podręcznik użytkownika, po przeczytaniu przez wszystkich członków Zespołu, zostanie przekazany osobom spoza Zespołu testującym program, aby określiły, jak dobrze spełnia on swoje cele i czy na jego podstawie można dobrze zapoznać się z obsługą programu.

8. Historia zmian

Wersja	Data	Autorzy zmian	Zmiany
0.5	22.02.2006	cały Zespół	pierwotna, niepełna wersja
0.6	26.02.2006	Maria Donten	poprawki merytoryczne
1.0	2.03.2006	Kuba Pochrybniak	poprawki merytoryczne i językowe