Software Development Plan

Łukasz Bieniasz-Krzywiec
Dariusz Leniowski
Jakub Łącki

30 maja 2007

Spis treści

1 Wprowadzenie

1.1 Cel

Celem Software Development Plan jest przedstawienie planów rozwoju projektu *YapS* w postaci, która przyspieszy wykonanie projektu i ułatwi zarządzanie nim.

1.2 Zakres

Software Development Plan obejmuje najważniejsze czynności i procedury w poszczególnych fazach budowy aplikacji. Uwzględnia on też strukturę i podział prac w zespole, który pracuje nad systemem.

1.3 Definicje

Definicje używanych terminów i skrótów znajdują się w załączonym słowniku.

1.4 Załączniki

Załączniki:

1. Wizja
2. Przypadki Użycia
3. Plan Testów
4. Słownik

2 Omówienie projektu

2.1 Cel i zakres projektu

Celem projektu *YapS* jest stworzenie systemu ułatwiającego przeprowadzanie wszelkiego rodzaju testów oraz ankiet i nieodpłatne dostarczenie go szerokiemu gronu odbiorców.

Głównymi zaletami *YapS* będą jego wsparcie do przeprowadzania egzaminów, bezpieczeństwo, w tym kontrola dostępu do poszczególnych formularzy, a przede wszystkim łatwość obsługi oraz obróbki zgromadzonych danych.

2.2 Założenia i zależności

1. *YapS* jest projektem darmowym,
2. *YapS* będzie rozpowszechniany na zasadach licencji wolnego oprogramowania,
3. *YapS* będzie realizowany przez zespół studentów WMIM (szczegóły w rozdziale 3.1),
4. praca nad dokumentacją *YapS* ma zostać ukończona do 30.06.2007 (szczegóły w rozdziale 4.3.4),
5. dokumentacja *YapS* jest pisana przy założeniu, że praca nad systemem będzie kontynuowana po ukończeniu drugiego roku studiów przez członków zespołu.

2.3 Ramowy harmonogram

Proces tworzenia *YapS* zostanie podzielony na 5 faz: Zbieranie wymagań, Planowanie, Tworzenie Dokumentacji, Implementację, Testowanie i poprawianie oraz Wdrażanie. Pierwsze trzy fazy odbywać będą wodospadowo. Związane jest to z tym, że każdy produkt tworzony podczas tych trzech faz w istotny sposób zależał od materiału przerobionego na zajęciach z Inżynierii Oprogramowania.

Począwszy od Implementacji praca zacznie się odbywać iteracyjnie. System zostanie podzielony na niezależne części (pakiety) i praca nad każdym z nich będzie mogła odbywać się równolegle. W kolejnych iteracjach poszczególne warstwy systemu coraz bardziej rozwijane.

- Iteracja 1 — wyświetlanie i wypełnianie formularzy.
- Iteracja 2 — tworzenie formularza (podstawowe komponenty); prawa dostępu do formularzy.
- Iteracja 3 — wyszukiwanie; wiadomości systemowe; tworzenie formularza (kolejne komponenty).
- Iteracja 4 — statystyki; ostateczna lista komponentów, zarządzanie grupami.

2.4 Produkty projektu

produkt	data wydania
Wizja	14.03.2007
Przypadki Użycia	21.03.2007
Software Architecture Document	23.05.2007
Software Development Plan	30.05.2007
Plan testów	30.05.2007
YapS DEMO	26.11.2007
YapS ALFA	07.01.2008
YapS BETA	03.03.2008
YapS 1.0	21.04.2008

2.5 Procedura zmian w planie projektu

Zmiany w tym dokumencie powinny zostać zatwierdzone przez większość członków zespołu. W przypadku nierozsądzonych kwestii spornych ostateczny głos ma kierownik projektu, a jego decyzje są ostateczne. Wszystkie dokumenty muszą zostać uaktualnione, a zmiany w nich dokonane uwzględnione w tych dokumentach.

3 Organizacja projektu

3.1 Struktura organizacyjna

W skład zespołu tworzącego *YapS* wchodzą:

- Jakub Łącki - kierownik zespołu, analityk/programista
  (e-mail: J.Lacki@yaps.pl),

- Łukasz Bieniasz-Krzywiec -
  zastępca kierownika, kontroler jakości, analityk/programista, tester
  (e-mail: L.Bieniasz-Krzywiec@yaps.pl),
- Dariusz Leniowski -
  specjalista od technologii *Ruby On Rails*, szef programistów, analityk/programista, tester
  (e-mail: D.Leniowski@yaps.pl)

Pozostali pracownicy:

- Adam Czepialski - tester

3.2 Kontakty zewnętrzne

- Jacek Sroka - konsultant, służy wszelką pomocą w każdej sprawie, wie wszystko o tworzeniu dokumentacji

3.3 Role i zadania

Rola i Cel	Obszary działalności	Odpowiedzialność
Kierownik zespołu — stworzenie *YapS* wraz z kompletną dokumentacją	• kontrola i podział pracy zespołu • prowadzenie repozytorium projektu • ocena biznesowa	• doprecyzowanie wymagań stawianych systemowi • przepływ informacji pomiędzy członkami zespołu • przebieg prac zgodny z przyjętym harmonogramem
Zastępca kierownika zespołu — dostarczenie aplikacji w terminie	• wybór i testowanie technologii • zarządzanie budżetem i programem szkoleń	• odpowiedzialny za komplet dokumentacji • raportuje aktualny stan prac • kieruje szacowaniem ryzyka
Szef zespołu programistów — nadzorowanie pracy programistów	• rozdzielanie zadań • kontrola jakości kodu • rozwiązywanie problemów związanych z *Ruby On Rails*	• integracja składowych systemów • wykrywanie błędów implementacji • zgodność tworzonego projektu ze specyfikacją
Kontroler jakości — zaakceptowanie aplikacji zgodnej ze specyfikacją i przyjętymi standardami	• korekta tworzonych dokumentów • planowanie, przeprowadzanie i raportowanie testów	• spójność powstających dokumentów • zgodność tworzonego projektu ze specyfikacją
analityk / programista — tworzenie dokumentacji, ustalanie i implementacja funkcji systemu	• tworzenie komponentów • rozwój aplikacji	• implementacja i działanie przydzielonej mu części aplikacji

4 Zarządzanie projektem

4.1 Oszacowania

YapS jest projektem darmowym realizowanym nieodpłatnie przez członków zespołu. Stan ten może ulec zmianie w przypadku zdobycia sponsora lub pozyskaniu grantu rozwojowego.

Docelowym terminem ukończenia *YapS* jest czerwiec 2008 roku.

4.2 Punkty funkcyjne

4.2.1 Nieznormalizowane punkty funkcyjne

ExternalInput		
Login/Hasło	Low	3
Edytowanie danych użytkownika	Average	4
Personalizowanie systemu	Low	3
Wysyłanie wiadomości	Low	3
Wyszukiwanie formularza	Low	3
Tworzenie grupy	Low	3
Dodawanie członków do grupy	Average	4
Nadawanie/odbieranie grupie dostępu do formularza	Average	4
Nadawanie/odbieranie osobie dostępu do formularza	Average	4
Utworzenie formularza	Low	3
Ustawienie właściwości formularza	Average	4
Wstawienie komponentu prostego	Average	4
Wstawienie komponentu rozszerzonego	High	6
Ustawienie własności komponentu prostego	Average	4
Ustawienie własności komponentu rozszerzonego	High	6
Podpięcie dodatkowych danych pod komponent rozszerzony	Average	4
Pobranie danych do komponentu	Average	4
Pobranie danych do komponent rozszerzonego	High	6
SUMA		72

ExternalOutput		
Powiadomienie o niepoprawnym haśle	Low	4
Powiadomienie o zmianie danych użytkownika	Low	4
Informowanie o nieprzeczytanych wiadomościach	Low	4
Czytanie wiadomości	Average	5
Wyświetlenie znalezionych formularzy	Low	4
Potwierdzenie utworzenia grupy lub komunikat błędu	Low	4
Potwierdzenie dodania osoby do grupy	Low	4
Potwierdzenie nadania grupie dostępu	Low	4
Potwierdzenie nadania osobie dostępu	Low	4
Wyświetlenie ramki formularza	Low	4
Wyświetlenie właściwości formularza	Average	5
Wyświetlenie komponentu prostego	Average	5
Wyświetlenie komponentu rozszerzonego	High	7

ExternalOutput		
Wyświetlenie dodatkowych danych komponentu rozszerzonego	Average	5
Odebranie danych od komponentu	Average	5
Odebranie danych od komponentu rozszerzonego	High	7
Export statystyk	Average	5
Export danych egzaminacyjnych	Average	5
SUMA		85

ExternalInquiry		
Informacje o użytkowniku	Low	3
Informacje o grupie	Low	3
Informacje o formularzu	Low	3
Wybór rodzaju wstawianego komponentu	Low	3
Usuwanie formularza	Average	4
Usuwanie komponentu	Low	3
Usuwanie grupy	Low	3
Usuwanie użytkownika	Low	3
SUMA		85

Internal Logical File		
Baza użytkowników	Average	10
Baza grup	Average	10
Baza formularzy	Average	10
Baza wypełnień formularzy	High	15
Baza danych komponentów	Low	7
Baza wypełnień komponentów	Low	7
Baza wiadomości	Low	7
SUMA		66

External Interface Files		
System pomocy	High	10
Szablony instalacyjne	Average	7
SUMA		17

4.2.2 Podstawowe cechy systemu

Data Communications	5
Distribted Data Processing	4
Performance	3
Heavily Used Configuration	2
Transaction Rate	4
Online Data Entry	5
End-User Efficeincy	3
Online Update	0
Complex Processing	1
Reusability	1

Installation Ease	3
Operational Ease	2
Multiple Sites	2
Faciliate Change	2
SUMA	37

4.2.3 Znormalizowane punkty funkcyjne

$$\sum \text{punktów} = 265$$

$$\sum \text{stopni skomplikowania} = 37$$

$$\text{współczynnik} = 0.65 + 0.01 * 37 = 1.02$$

$$\text{punkty funkcyjne} = 265 * \text{współczynnik} = 270.3 \simeq 270$$

4.3 Plan projektu

4.3.1 Plan faz projektu

Nazwa	Harmonogram	Czynności
Wymagania	**Rozpoczęcie:** 21.02.2007 **Zakończenie:** 07.03.2007	• postawienie i doprecyzowanie wymagań stawianych systemowi • przygotowanie dokumentu *Wizja Systemu* • planowanie przypadków użycia
Planowanie	**Rozpoczęcie:** 07.03.2007 **Zakończenie:** 21.03.2007	• podział obowiązków i zakresu odpowiedzialności

Nazwa	Harmonogram	Czynności
Tworzenie Dokumentacji	**Rozpoczęcie:** 21.03.2007 **Końce iteracji:** 1) 30.05.2007 2) 15.06.2007	• wykonanie wszystkich niezbędnych dokumentów (dwie iteracje): – *Przypadki Użycia* – *Plan Wykonania Systemu* – *Projekt Architektury Systemu* – *Plan testów*
Implementacja	**Rozpoczęcie:** 03.11.2006 **Końce iteracji:** 1) 26.11.2007 2) 07.01.2008 3) 03.03.2008 4) 21.04.2008	• stworzenie wszystkich komponentów systemu *YapS* • testowanie poszczególnych fragmentów • wyszczególnienie wszystkich komponentów systemu
Testowanie i poprawianie gotowego systemu	**Rozpoczęcie:** 02.05.2008 **Zakończenie:** 03.03.2008	• przygotowanie zestawu automatycznych testów (tam gdzie możliwe) • sprawdzenie integralności komponentów • nanoszenie ewentualnych poprawek
Odbiór systemu	**Rozpoczęcie:** 03.06.2008 **Zakończenie:** 15.06.2008	• ustalenie dokładnej daty prezentacji systemu • prezentacja ostatecznej wersji *YapS*

4.3.2 Cele poszczególnych iteracji

Poniższa tabela obejmuje fazy składające sie z więcej niż jednej iteracji. Opis wyników poszczególnych faz pokrywa się z zawartymi w punkcie 4.3.1.

Faza	Iteracja	Wynik
Tworzenie Dokumentacji	1	Przygotowanie wszystkich dokumentów.
Tworzenie Dokumentacji	2	Poprawienie i skonsolidowanie wszystkich dokumentów.
Implementacja	1	Wersja demonstracyjna (DEMO) systemu
Implementacja	2	Wersja ALFA systemu
Implementacja	3	Wersja BETA systemu
Implementacja	4	Wersja 1.0 systemu

4.3.3 Wydania

wersja	data wydania	przeznaczenie
DEMO	26.11.2007	pierwsza, demonstracyjna wersja *YapS*; służy przede wszystkim prezentacji architektury systemu, szaty graficznej oraz GUI
ALFA	07.01.2008	druga wersja *YapS*; ukazuje podstawowe funkcjonalności, czyli te które w dokumencie *„Wizja”* zostały opatrzone wysokim priorytetem
BETA	03.03.2008	wersja testowa *YapS*; przedstawia praktycznie pełną funkcjonalność systemu, celem tej wersji jest przetestowanie systemu w praktyce oraz wykrycie i poprawienie błędów
1.0	21.04.2008	wersja finalna, gotowa do publikacji

4.3.4 Harmonogram projektu

Produkt	Początek	Koniec	Osoby zaangażowane
Wizja	21.02.2007	08.03.2007	Zespół twórców
Przypadki Użycia	08.03.2007	21.03.2007	Zespół twórców
Model systemu	21.03.2007	18.04.2007	Zespół twórców
Projekt Architektury Systemu	11.04.2007	23.05.2007	Zespół twórców
Plan Wykonania Systemu	16.05.2007	30.05.2007	Kierownik projektu
Plan Testów	16.05.2007	30.05.2007	Zastępca kierownika projektu
YapS DEMO	01.10.2007	27.11.2007	Zespół twórców
YapS ALFA	27.11.2007	07.01.2008	Zespół twórców
YapS BETA	08.01.2008	03.03.2008	Zespół twórców
YapS 1.0	04.03.2008	21.04.2008	Zespół twórców

4.3.5 Zasoby

Plan zatrudnienia: System *YapS* od początku do końca będzie wykonany przez zespół w składzie:

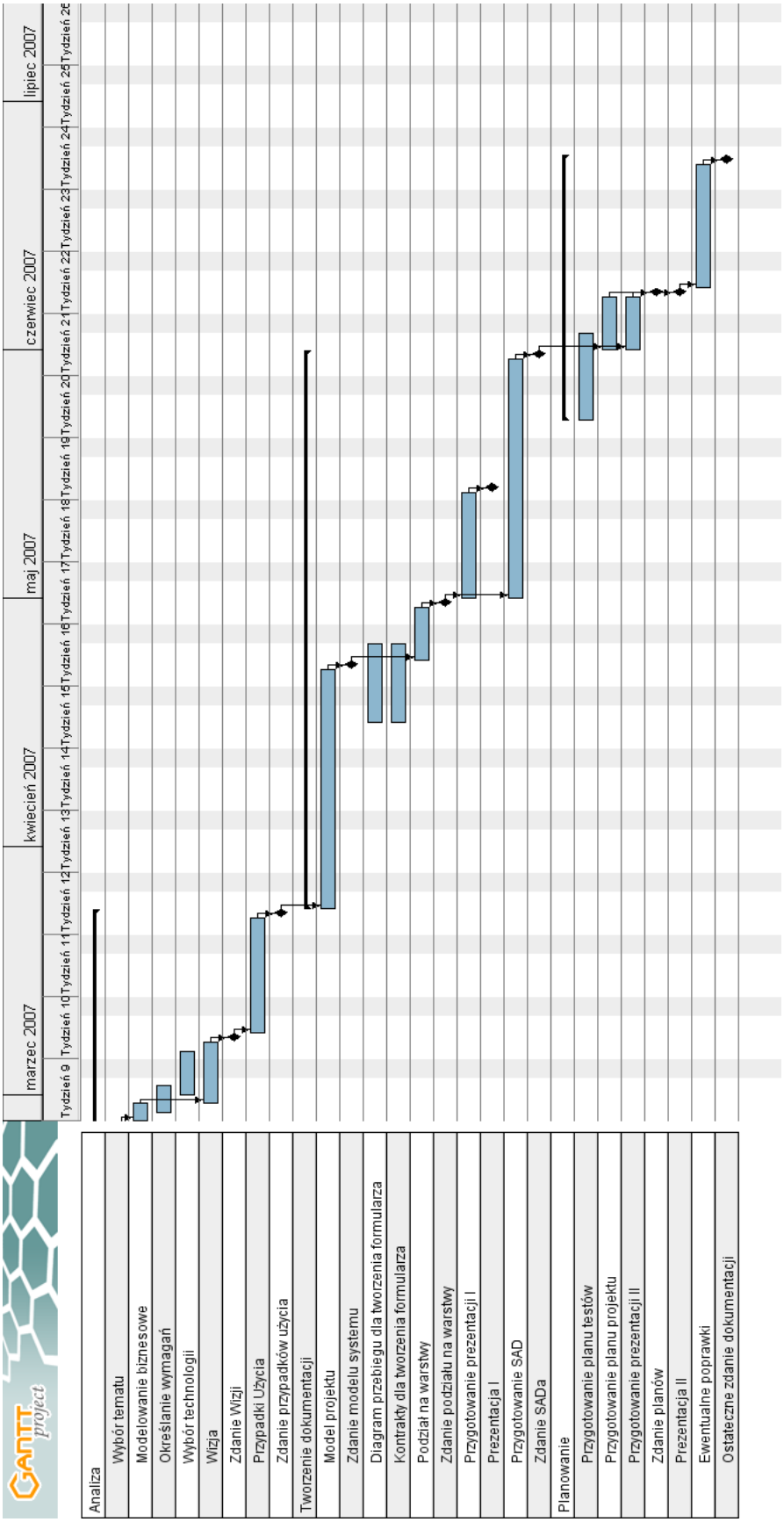
- Łukasz Bieniasz-Krzywiec
- Dariusz Leniowski
- Jakub Łącki

Plan zatrudniania pracowników: Nie planuje się zatrudniania dodatkowych pracowników.

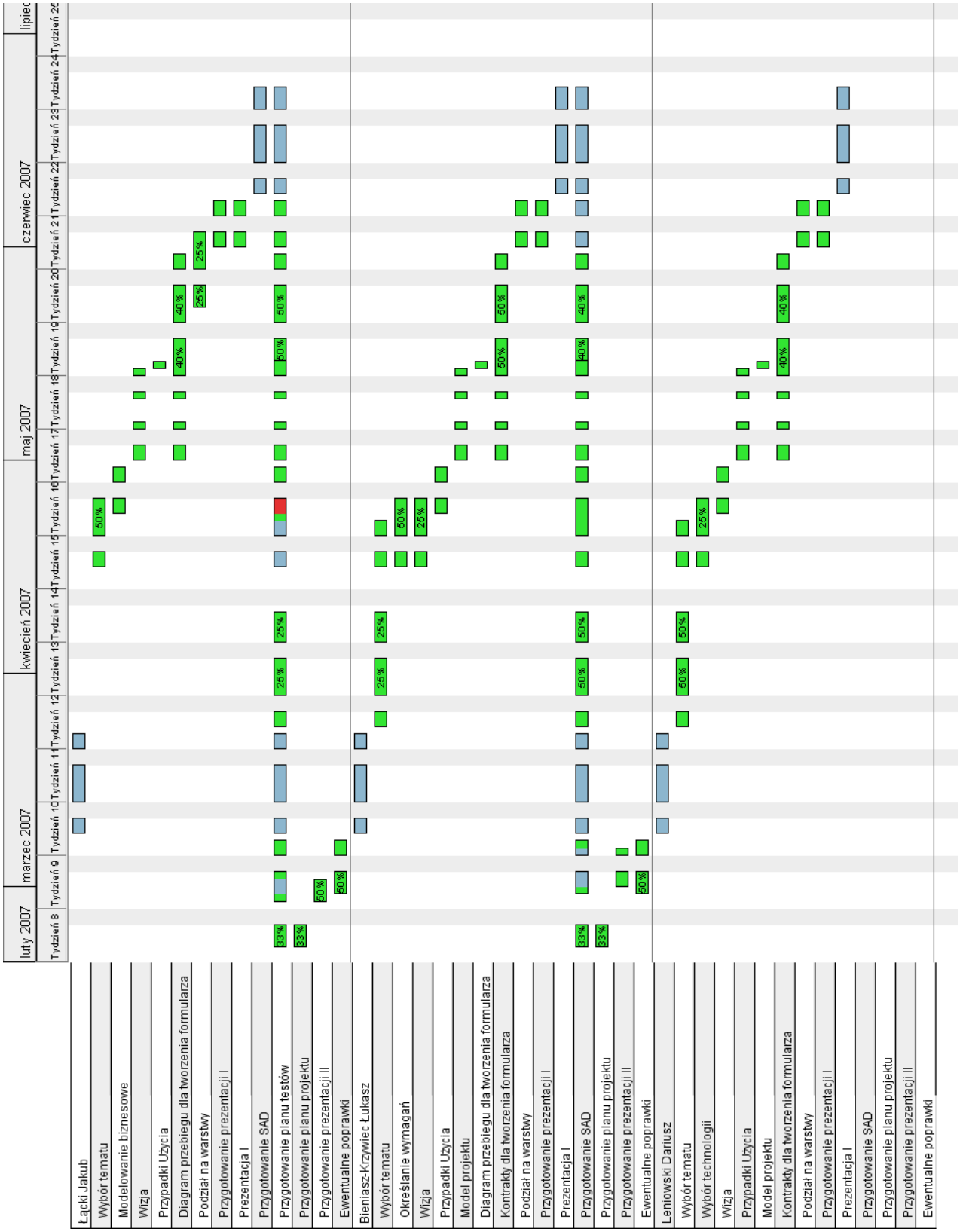
Plan szkoleń:

- LATEX – 21.02.2007 - 08.03.2007
- SVN – 28.02.200 - 08.03.2007
- UML – 08.03.2007 - 30.04.2007
- *Ruby On Rails* - 01.09.2007 – 30.09.2007
- Nowe technologie – na bieżąco

4.3.6 Diagram Gantta (tworzenie dokumentacji)



4.3.7 Diagram podziału pracy (tworzenie dokumentacji)



4.3.8 Budżet

YapS jest projektem darmowym, wykonywanym nieodpłatnie. Członkowie zespołu nie ponoszą żadnych kosztów w związku z projektem, ich jedynym wkładem jest poświęcony czas i praca. Nie posiadamy, ani nie planujemy posiadać budżetu dla projektu.

4.4 Plany iteracji

4.4.1 Cel

Celem drugiej iteracji jest dokończenie implementacji podstawowych funkcjonalności systemu, to znaczy tworzenia, wyświetlania i wypełniania formularza, a także dodanie częściowej kontroli praw dostępu. Po drugiej iteracji wydana zostanie wersja DEMO.

4.4.2 Podział obowiązków

Druga iteracja jest zaplanowana do realizacji od 26.11.2007 do 05.01.2008, według następującego planu:

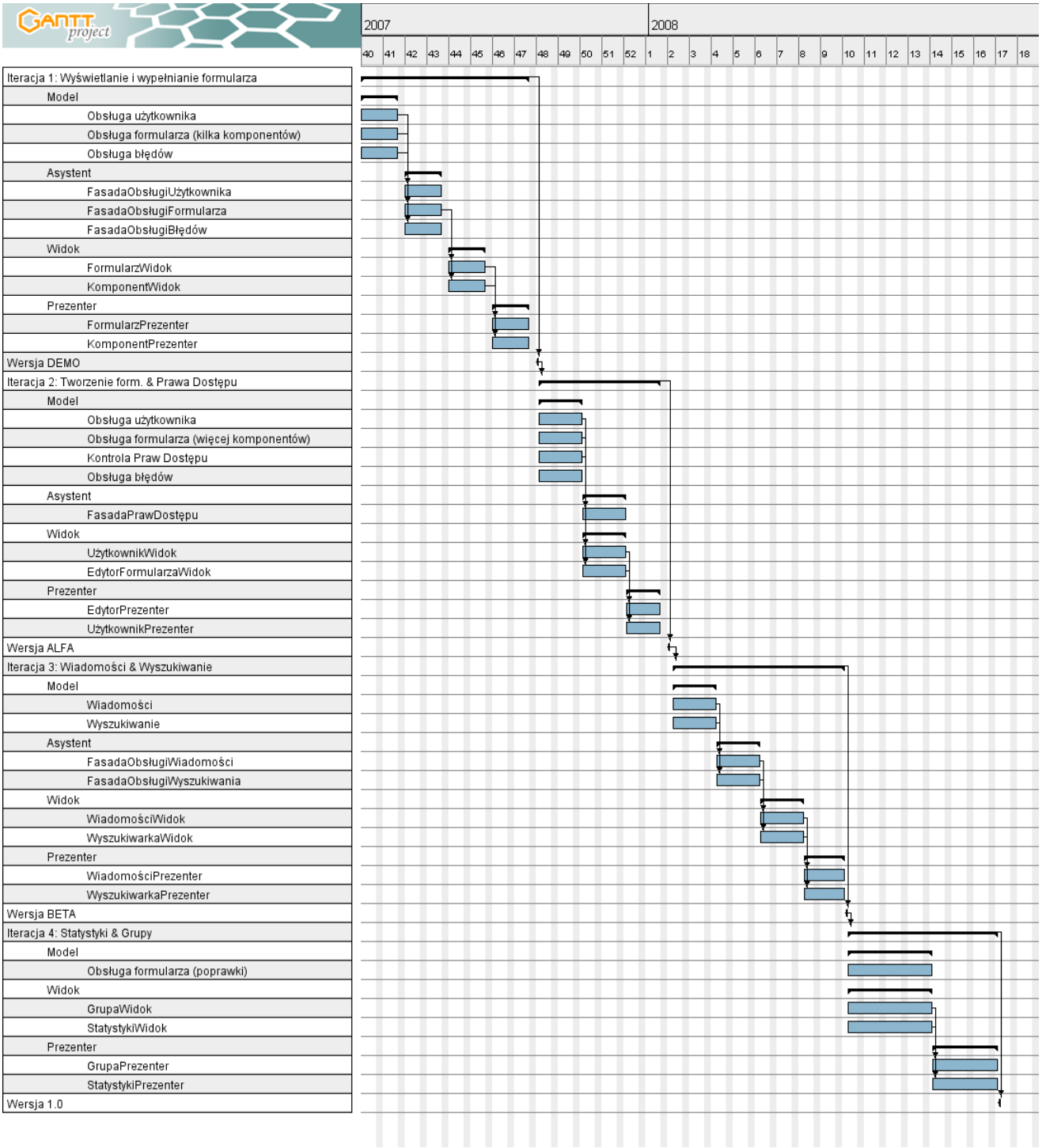
- Rozszerzenie Modelu (27.11.2007 - 11.12.2007):
  - Obsługa użytkownika — Jakub Łącki
  - Obsługa formularza (podstawowa lista komponentów) — Łukasz Bieniasz–Krzywiec
  - Kontrola Praw Dostępu — Dariusz Leniowski
  - Obsługa Błędów — Dariusz Leniowski
- Obsługa praw dostępu w Asystencie (11.12.2007 - 23.12.2007):
  - FasadaPrawDostępu — Dariusz Leniowski
- Rozszerzenie Widoku (11.12.2007 - 23.12.2007):
  - UżytkownikWidok — Jakub Łącki
  - EdytorFormularzaWidok — Dariusz Leniowski
- Dodanie obsługi rozszerzonego Widoku (27.12.2007 - 07.01.2008):
  - EdytorPrezenter — Łukasz Bieniasz-Krzywiec, Dariusz Leniowski
  - UżytkownikPrezenter — Jakub Łącki

4.4.3 Przypadki użycia

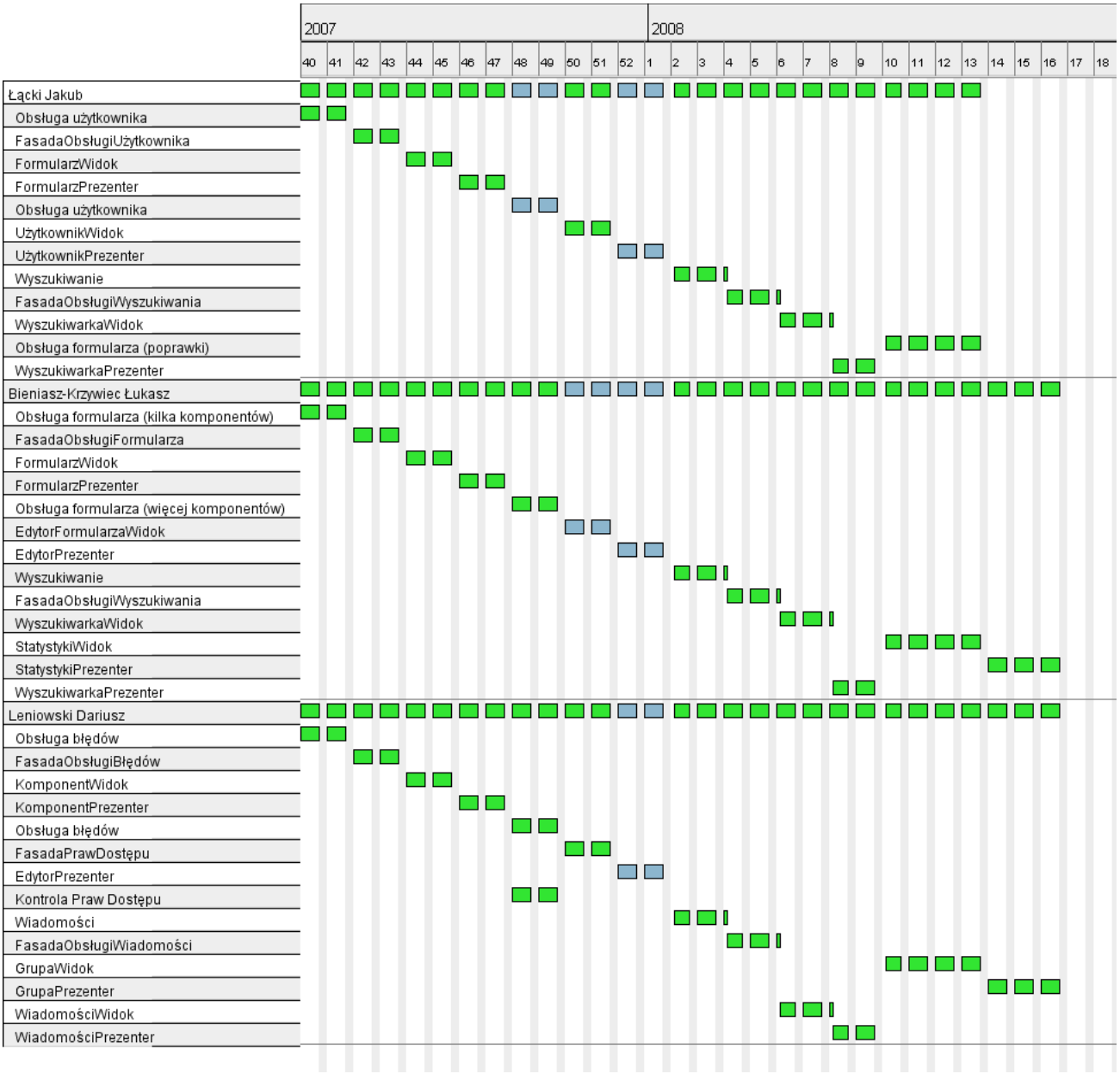
Następujące przypadki użycia będą rozwijane w tej iteracji:

1. logowanie,
2. tworzenie formularza,
3. wypełnianie formularza.

4.4.4 Diagram Gantta (implementacja)



4.4.5 Diagram podziału pracy (implementacja)



4.5 Nadzór i kontrola projektu

4.5.1 Plan zarządzania wymaganiami

Wymagania systemu zostały stworzone przez autorów projektu i spisane w Przypadkach Użycia oraz Wizji. Zakładamy, że nie będą one ulegać poważnym zmianom podczas implementacji projektu. Wątpliwości co do sensowności i pełności wymagań mogą pojawić się podczas testowania systemu pod kątem funkcjonalności. Powstałe problemy należy rozwiązać, o ile nie zrodzi to ryzyka wykroczenia poza harmonogram projektu.

4.5.2 Plan zarządzania harmonogramem

Za pilnowanie harmonogramu odpowiedzialny jest kierownik zespołu. Harmonogram projektu został szczegółowo opisany w rozdziale 4.3.1. Dopuszczamy co najwyżej dwutygodniowe opóźnienie w stosunku do harmonogramu, jednakże każdy dzień zwłoki będzie się wiązał z wyciągnięciem konsekwencji od osoby odpowiedzialnej za spóźnienie. Rodzaj konsekwencji zostanie ustalony przez *Kontrolera jakości*. Jeśli opóźnienie przekroczy jeden tydzień, konieczne będzie zwiększenie zaangażowania uczestników projektu, w celu powrotu do pierwotnego harmonogramu.

4.5.3 Plan pomiarów

W celu ułatwienia zarządzania harmonogramem, kierownik zespołu będzie mierzył postęp pracy nad *YapS*, poprzez zliczanie osiągniętych punktów kontrolnych. Punktem takim może być wykonanie obsługi pewnego przypadku użycia, napisanie jakiegoś modułu, a nawet napisanie pojedynczej funkcji. Punkty kontrolne będą wyznaczane na początku każdej iteracji.

4.5.4 Plan kontroli jakości

Jakość systemu z perspektywy użytkownika, jak i jakość samej implementacji będzie mierzona na bieżąco poprzez jego testowanie. Więcej informacji na ten temat znaleźć można w *"Planie Testów"*.

4.6 Plan zarządzania ryzykiem

Poniżej przedstawiamy działania jakie należy podjąć w przypadku zajścia sytuacji wyjątkowych. Ostateczne decyzje na temat podejmowanych działań podejmuje cały zespół.

Nagła niedyspozycja jednego z twórców projektu Jeśli jeden z autorów *YapS* nie będzie mógł przez pewien czas uczestniczyć w projekcie, należy zwiększyć przydział pracy pozostałych członków zespołu. Zespół składa się jedynie z trzech osób, przez co w przypadku dłuższej niedyspozycji, konieczne będzie wydłużenie harmonogramu projektu.

Nieoczekiwane pojawienie się konkurencji W przypadku, gdy na rynku pojawi się system o podobnych możliwościach do *YapS* wydany na jednej z licencji open source, konieczne będzie obiektywne porównanie go z *YapS*. Jeśli *YapS* nie będzie miał szans konkurować z tym systemem, należy przerwać projekt.

Źle dobrana technologia Jeśli okaże się, że technologia wybrana do stworzenia projektu nie spełnia naszych założeń — jest zbyt mało efektywna lub niewygodna w użyciu do naszych celów, należy zmienić technologię. Zakładamy przy tym, że odpowiedniość technologii da się stwierdzić już na wczesnym etapie implementacji. W innym przypadku, konieczne będzie kontynuowanie projektu przy użyciu obecnej technologii, co może doprowadzić do ograniczenia jakości i funkcjonalności systemu.

Problemy z zasobami W przypadku awarii lub utraty zasobów koniecznych do tworzenia projektu (komputerów, oprogramowania), rozwiązanie powstałego problemu jest obowiązkiem osoby poszkodowanej. Gdyby doszło do takiej sytuacji, osoba poszkodowana może bez konsekwencji opóźnić nieco wykonanie swoich zadań.

Utrata danych Aby zapobiec utracie danych, będą one przechowywane w systemie kontroli wersji SVN. Ponadto, kopie robocze całego repozytorium znajdować się będą na komputerach autorów, dlatego utrata dużej części danych jest mało prawdopodobna.

5 Plany procesów technicznych

5.1 Metody, narzędzia i stosowane technologie

- modelowanie:
  - Rational Unified Process
  - Unified Modeling Language
  - Violet
  - UMLet
  - Gantt project
  - Visual Paradigm
  - LaTeX
  - LaTeX Beamer Class
- kontrola wersji i narzędzia ogólne:
  - darcs
  - SVN
  - Trac
  - Ruby
  - GNU Make
  - VIM — Vi IMproved
- implementacja
  - system operacyjny zgodny z POSIX (Debian, Fedora Core i FreeBSD)
  - serwer HTTP/HTTPS z obsługą CGI, FCGI oraz TLS 1.0 (Apache 2)
  - PostgreSQL
  - Ruby on Rails
  - ImageMagick

6 Plany pomocnicze

6.1 Plan zarządzania zmianami

Kontrola wersji za pomocą SVN'a będzie zawierała całą dokumentację (pliki .tex, .pdf) oraz implementację systemu (kod źródłowy).

6.2 Plan oceny

Ocenę za dokumentację wystawi prowadzący zajęcia Inżynierii Oprogramowania po oddaniu jej w terminie (30.05.2007).

Ocenę za realizację projektu wystawi prowadzący zajęcia Zespołowego Projektu Programistycznego w czerwcu przyszłego roku.

6.3 Plan dokumentacji

Cała dokumentacja *YapS* jest oparta o szablony RUP. W jej skład wejdą:

- Przypadki Użycia
- Plan Wykonania Systemu
- Projekt Architektury Systemu
- Plan testów

6.4 Plan zapewnienia jakości

6.4.1 Zapewnienie jakości dokumentacji

W czasie prac nad projektem *Kontroler jakości* jest na bieżąco informowany o przebiegu prac. Pojedyncze rozdziały trafiają do niego nie później niż na 12 godzin przed terminem oddania dokumentu. *Kontroler jakości* sprawdza dokumenty pod względem merytorycznym, zgodności z tematem i spójności z wcześniej wykonanymi. W przypadki odnalezienia drobnych błędów sam dokonuje poprawek. Jeśli natomiast znalezione usterki okażą się poważne (według uznania *Kontrolera jakości*) informuje o nich autora dokumentu, który jest zobowiązany (w ciągu 2 dni roboczych) nanieść odpowiednie poprawki i przekazać dokument do ponownej kontroli.

Dodatkowo przy dokumencie *Plan Architektury Systemu Kontroler jakości* otrzymuje dwa razy w tygodniu raporty z postępu prac i kopię powstającego dokumentu, tak aby mógł na bieżąco sprawdzać jakość powstającej dokumentacji.

Ponadto *Kontroler jakości* ustala kary za nie dotrzymanie terminu oddania dokumentu. Osoba odpowiedzialna za opóźnienie jest zobowiązana zakupić skrzynkę napoju *Tymbark* do wspólnej puli wszystkich członków zespołu.

6.4.2 Zapewnienie jakości kodu

Szef programistów ustala terminy spotkań podczas których programiści będą przedstawiali swoje fragmenty kodu, po czym będzie następowała ich inspekcja. Ewentualne poprawki są nanoszone przez autorów kodu źródłowego.

6.5 Plan rozwiązywania problemów

Rozwiązania podstawowych problemów są opisane w punkcie 4.5. W razie wystąpienia problemu nie zawartego w punkcie 4.5 jest on zgłaszany bezpośrednio do kierownika zespołu, który decyduje o sposobie rozwiązania zaistniałego problemu.

7 Historia zmian

0.1 - dostosowanie szablonu do potrzeb projektu YapS
0.2 - rozdzielenie pracy nad dokumentem
0.3 - pierwsze wersje rozdziałów 2 i 6
0.4 - pierwsza wersja rozdziału 1 i podrozdziałów rozdziału 4
0.5 - pierwsza wersja rozdziału 5
0.6 - drugie wersje wszystkich rozdziałów
0.7 - usunięcie indeksu
0.8 - poprawienie błędów
0.9 - wstawienie obrazków
0.10 - przeformatowanie tabelek
0.11 - więcej poprawiania błędów
0.12 - spellchecking
0.13 - dodanie punktów funkcyjnych
0.14 - poprawki rozdziałów
0.15 - drobne korekty -> wersja finalna