Spis treści

Rozdział 1

Wstęp

Rozwój technologii internetowych w ostatnich latach umożliwił programistom tworzenie wysoce interaktywnych narzędzi przy jednoczenym ograniczeniu wymagań systemowych jakie muszą spełnić klienci, chcący skorzystać z wytworzonego oprogramowania. W szczególności, niedawno wprowadzona na rynek, technologia HTML5 daje nowe, szerokie możliwości przeglądarkom internetowym. Możliwości, które były dotychczas zarezerwowane głównie dla technologii Flash (oraz z trochę gorszymi wynikami MS Silverlight, JavaFX, itp.) stają się wspierane natywnie, w każdej popularnej przeglądarce internetowej.

Z wymaganiami użytkownika stykamy się praktycznie od samego początku pracy nad projektem informatycznym. Z tego powodu potrzebne są skuteczne narzędzia gromadzenia i przetwarzania wymagań użytkownika oraz komunikowania ich wszystkim członkom zespołu. Dzięki wyżej wspomnianej technologii HTML5, istnieje możliwość usprawnienia procesu dokumentowania wymagań przy jednoczesnym zachowaniu wysokiej dostępności aplikacji, włączając do niego nowe narzędzia oparte na metodach graficznych.

Rynek nowych technologii roi się dzisiaj od aplikacji rozwiązujących rozmaite problemy, od prostych list zakupów, po zaawansowane narzędzia wspierające zarządzanie projektami. Pomimo istnienia wielu aplikacji skupiających się na procesach fazy analizy, bardzo ciężko znaleźć rozwiązanie dostosowane do potrzeb i realiów prowadzenia projektów w małych i średnich

przedsiębiorstwach.

W niniejszej pracy zostanie zaprezentowana koncepcja aplikacji wspierającej zarządzanie wymaganiami użytkownika, stworzona z myślą o rzeczywistych problemach jakie wiążą się z tym etapem projektu informatycznego. Zaproponowany system ma za zadanie ułatwić zespołom projektowym komunikację i dostęp do wiedzy przy jednoczesnym skupieniu uwagi na najistotniejszych aspektach definiowania i przetwarzania wymagań.

W niniejszej pracy zaprezentowano nowoczesne narzędzie wspierające proces zbierania wymagań użytkownika. Przyjęte rozwiązania mają za zadanie ułatwić zespołom projektowym komunikację, dostęp do wiedzy oraz wizualizację przypadków użycia w kontekście fazy analizy i zbierania wymagań. Głównymi założeniami przy tworzeniu koncepcji rozwiązania, były m.in. prostota obsługi oraz skupienie uwagi użytkownika na najważniejszych aspektach definiowaVnia wymagania.

1.1 Cel pracy

W niewielkich projektach, proces gromadzenia i dokumentowania wymagań użytkowników jest najczęściej słabo sformalizowany. Wymagania trafiają do zespołu z różnych, heterogenicznych źródeł i w wielu, zwykle niekompatybilnych, formatach. Pomimo faktu, iż na rynku nie brakuje oprogramowania wspierającego zarządzanie wymaganiami, istniejące rozwiązania, często nie są przystosowane do realiów prowadzenia projektów w małych, dynamicznych przedsiębiorstwach. Skomplikowane i trudno dostępne systemy często wymagają instalacji oprogramowania po stronie klienta [!ref] a rozwiązania dostępne online, w formule SaaS (Sofware As A Service) [!ref] często powielają tylko funkcjonalnosći potężnych (i drogich) systemów takich jak IBM DOORS [!ref] czy IBM ReqiusitePro [!ref]. Ponadto niezwykle trudno jest znaleźć rozwiązanie darmowe lub posiadające ogólno dostępną wersję demonstracyjną.

Celem niniejszej pracy jest konstrukcja prototypu systemu usprawniającego zbieranie i przetwarzanie wymagań użytkownika w postaci tekstowej oraz graficznej, dostarczając narzędzi edycji tekstu oraz graficznego mode-

lowania przypadków użycia. W efekcie, użytkownik, po wprowadzeniu początkowych wymagań, ma możliwość automatycznego wygenerowania dokumentu SRS (Software Requirement Specification) [!ref]. Podejście zaprezentowane w tej pracy, opiera się na założeniach, że nowoczesne oprogramowanie wspierające zarządzanie wymaganiami, powinno m.in.: stawnowić centralne repozytorium wiedzy o wymaganiach; być łatwo dostępne w jak największej ilości różnych środowisk; umożliwiać łatwą kolaborację oraz być łatwe w obsłudze, prowokując do kreatywności, zamiast stawiać bariery w postaci skomplikowanych formularzy i tabel.

1.2 Zarys koncepcji

Proces zbierania wymagań jest trudny, ponieważ często jest procesem niesformalizowanym, rozproszonym i udokumentowanym na wiele sposobów (różne nośniki danych, niekomatybilne oprogramowanie). Ponadto wymagania zwykle pochodzą od wielu interesariuszy: od sponsora projektu, po użytkowników końcowych. Niezależnie od poziomu dojrzałości organizacji i stosowanej metodyki zarządzania projektami, źródła pochodzenia wymagań pozostają rozproszone i niekompatybilne.

Na potrzeby ninejszej pracy, został stworzony prototyp systemu pozwalający w łatwy sposób dokumentować gromadzone wymagania użytkownika. W powstałym systemie, głownymi narzędziami definiowania wymagań są pliki tekstowe oraz diagramy przypadków użycia. Osoby odpowiedzialne w projekcie za zarządzanie wymaganiami, otrzymują proste w obsłudze narzędzia, pozwalające na skupienie się nad sednem poroblemu, który poszczególne wymaganie ma na celu rozwiązać. Zaimplementowano system zarządzania projektem, w którym użytkownik ma możliwość zdefiniowania podstawowych parametrów takich jak nazwa projektu, planowany czas zakończenia oraz ogólny opis, zawierający kluczowe informacje o projekcie na etapie analizy. Korzystając z narzędzia definiowania wymagań w projekcie, użytkownik ma do dyspozycji edytor tekstowy, w którym dokumentuje zidentyfikowane wymagania użytkownika. Dzięki wykorzystaniu prostego języka znaczników (Markdown [!ref]), użytkownik skupia się na logicznej strukturze opisu wymagania. Brak

narzędzi "WYSIWYGźnanych ze standardowych edytorów tekstu sprawia, że użytkownik nie jest rozpraszany potrzebą myślenia o graficznej reprezentacji tekstu opisującego problem, przy jednoczesnym zachowaniu czytelnej struktury dokumentu. System załączników i komenatrzy, umożliwia iteracyjną pracę nad wymaganiami przy wykorzystaniu zewnętrzych źródeł informacji.

Dzięki graficznemu edytorowi przypadków użycia standardu UML, użytkownik ma możliwość definiowania kluczowych funkcjonalności systemu i łączenia ich z wybranymi wymaganiami. Tworzone przez użytkowników diagramy, są dostępne do ponownego wykorzystania w innych miejscach w systemie, dzięki czemu wspierana jest kolaboracja i korzystanie z już istniejących rozwiązań. Zarówno opisy wymagań jak i przypisane im przypadki użycia, są integralną częścią dokumentu specyfikacji wymagań użytkownika. Dlatego na każdym etapie fazy analizy istnieje możliwość automatycznego wygenerowania specyfikacji wymagań na podstawie danych wprowadzonych do systemu. Takie podejście sprzyja iteracyjnemu podejściu do tworzenia specyfikacji wymagań i umożliwia prezentację specyfikacji już na wczesnym etapie projektu.

Pozyskiwanie wymagań jest procesem kreatywnym. Koncepcja systemu bazuje na silnym przekonaniu, że brak ograniczeń w postaci skomplikowanych przytłaczających fuknc jonalności pozwala skupić się na procesie twórczym w fazie definiowania wymagań, rozwiązujących rzeczywiste problemy.

Głównym założeniem projektu, jest następująca hipoteza: dostarczenie odpowiednich, przemyślanych i prostych narzędzi, łatwych do natychmiastowego wdrożenia w projekcie, powoduje zmniejszenie czasu pracy nad specyfikacją wymagań użytkownika oraz zwiększa jakość powstałych w tym procesie wymagań.

Głównym założeniem projektu jest następująca hipoteza: ograniczenie możliwości definiowania formatu i struktury opisu wymagań, dostarczenie graficznych metod modelowania przypadków użycia oraz możliwość wygenerowania specyfikacji na podstawie danych wprowadzonych do systemu, ułatwi proces odkrywania i dokumentowania wymagań. Użytkownik nie powinien być jednocześnie przytłoczony ilością funkcjonalności dostępnych w aplikacji, a narzędzia którmi dysponuje powinny być intuicyjne i nie wymagać żadnej

dokumentacji.

1.3 Struktura pracy

W rozdziale 2 zostaną przedstawione, dostępne na rynku, narzędzia wspierające zarządzanie wymaganiami użytkownika. Rodział 3 jest dokładnym opisem dziedziny problemu oraz koncepcji zaimplementowanego prototypu. W rozdziale 4 zaprezentowano najistotniejsze technologie, jakie wykorzystano podczas pracy nad prototypem oraz krótkie omówienie środowiska programistycznego w jakim powstał przedmiot pracy. Rozdział 5 stanowi opis zaimplementowanego prototypu. W rozdziale 6 zawarto podsumowanie wyników pracy oraz propozycje kierunków dlaszego rozwoju.

Rozdział 2

Istniejące rozwiązania

W poprzednim rozdziale nakreślono temat przewodni niniejszej pracy. Poruszono problem dostępności i użyteczności istniejących narzędzi wspomagających zarządzanie wymaganiami. Zaprezentowano także autorską koncepcję rozwiązania, mającego na celu usprawnienie procesu tworzenia specyfikacji wymagań użytkownika.

W tym rozdziale zostanie przeprowadzona analiza rynku oprogramowania wspierającego fazę analizy. Druga część rozdziału zajmie się identyfikacją najistotniejszych problemów jakie posiadają istniejące rozwiązania.

2.1 Analiza rynku systemów wsparcia zbierania wymagań

Na rynku nie brakuje systemów wsparcia procesu zbierania wymagań. Dostępne rozwiązania oferowane są w zasadzie w trzech różnych modelach. Najpopularniejszą ostatnio architekturą jest Software As A Service, czyli aplikacja internetowa zainstalowana na serwerach twórcy oprogramowania. Również klasyczne aplikacje okienkowe, które należy zainstalować na komputerze klienta nadal cieszą się dużą popularnością. Należy jednak zaznaczyć, iż w przeważającej większości są to starsze systemy, nierzadko implementowane jeszcze przed popularyzacją zaawansowanych aplikacji webowych. Niektóre firmy oferują także platformy w architekturze klient-serwer, wymagające ist-

nienia zarówno centralnego serwera - repozytorium w sieci jak i desktopowych aplikacji klienckich. W tej sekcji zostaną przedstawione i porównane wybrane aplikacje z każdej z trzech kategorii.

2.1.1 Model SaaS (Sowtware as a Service)

W ostatnim czasie, wraz z popularyzacją i rozwojem technologii internetowych znacznie wzrosły techniczne możliwości aplikacji dostępnych z poziomu przeglądarki internetowej. Tendencja ta sprzyja powstawaniu licznych aplikacji, adresujących bardzo specyficzne problemy, często w obrębie ściśle określonego segmentu rynku. Jednym z przykładów takiego wąskiego segmentu mogą być np. aplikacje do zarządzania projektami online. Wśród innych popularnych rozwiązań znajdują się takie produkty jak basecamp.com, polski nozbe.com czy unfuddle.com. Innym przykładem mogą być aplikacje wspierające proces rekrutacji (humanway.com, recruiterbox.com, jobvite.com). Na uwagę zasługują także aplikacje wspierające rezerwacje hoteli, niezależnych pokoi i mieszkań jak airbnb.com, booking.com czy b&b.com. Naturalnie powyższe przykłady dotyczą tylko skrawka możliwych do zagospodarowania segmentów. W rzeczywistości, bardzo ciężko znaleźć wertykalny rynek, który jest niezagospodarowany serwisami, oferującymi różne podejścia do rozwiązania problemów danego segmentu.

Powodem takiego stanu rzeczy jest szeroko pojęta popularyzacja internetu oraz przenoszenie się do sieci firm tworzących opgrogramowanie. Dystrybucja oprogramowania w formule SaaS ma wiele zalet w stosunku do klasycznych aplikacji "desktopowych". W szczególności pomięty jest w tym przypadku proces rozprowadzania aplikacji do klientów za pomocą sieci stacjonarnych sklepów z oprogramowaniem. Zaimplementowane rozwiązanie, jest gotowe do użycia z chwilą udostępnienia w internecie. Udostępnienie aplikacji na serwerach twórcy oprogramowania daje nieograniczone możliwości wprowadzania nowych funkcjonalności i poprawek. Z kolei monitorowanie zachowań klientów pozwala niemalże w czasie rzeczywistym odpowiadać na potrzeby użytkowników.

W związku z licznymi zaletami modelu SaaS, w internecie powstało wiele

aplikacji próbujących zaadresować problemy związane z procesem zbierania i dokumentowania wymagań użytkownika. Do najciekawszych rozwiązań można zaliczyć gatherspace (http://gatherspace.com/) [!ref] «««« krótki opis głównych osobliwości gatherspace'a »»»», tracecloud.com «««« krótki opis głównych osobliwości tracecloud'a »»»», accompa.com «««« krótki opis głównych osobliwości accompy »»»»

Gatherspace

Tracecloud

Accompa

2.1.2 Aplikacje desktopowe

Model SaaS, mimo wszystkich swoich zalet, posiada również wady. Głównym powodem kontrowersji jest konieczność powierzenia danych biznesowych firmie udostępniającej narzędzie na swoich serwerach. Dla dużych korporacji, pilnie strzegących swoich tajemnic handlowych, takie rozwiązanie, może być nie do zaakceptowania ze względów bezpieczeństwa. Mimo potencjalnej możliwośći uniknięcia kosztów budowy i utrzymania infrastruktury sprzętowo-software'owej ryzyko związane z brakiem kontroli nad powierzonymi danymi jest często zbyt wielkie.

W klasycznych aplikacjach okienkowych, odpowiedzialność w zakresie zabezpieczenia danych spoczywa na samej korporacji i jej pracowniku. Dzięki temu, nadal istnieje zapotrzebowanie na oprogramowanie instalowane na lokalnym dysku użytkownika. Przykładami takich aplikacji są m.in. IBM DOORS, IBM Rational RequisitePRO oraz seapine.com.

RequisitePRO

DOORS

seapine

2.1.3 Platforma IBM .jazz

Platforma IBM jazz zasługuje na osobną sekcję, ponieważ jest zintegrowanym, kompleksowym zestawem narzędzi i aplikacji dla przedsiębiorstw tworzących oprogramowanie. Centrum zarządzania platformą jazz jest serwer stanowiący repozytorium danych i ośrodek dowodzenia. Platforma składa się zarówno udostępnionych na serwerze usług sieciowych oraz aplikacji instalowanych lokalnie, łączących się ze zdalnym serwerem (tzw. rich-client applications).

2.2 Problemy z istniejącymi rozwiązaniami

W poprzedniej sekcji omówiono wybrane narzędzia z wielu dostępnych na rynku aplikacji wspomagających zarządzanie wymaganiami. !! Analiza zastosowanych rozwiązań zdecydowanie prowokuje do przemyśleń i nasuwa na myśl serię pytań związanych motywacjami jakimi kierują się twórcy tego typu oprogramowania.

Analizując istniejące rozwiązania, nie można oprzeć się wrażeniu, że wszystkie przystosowane są do dużych, korporacyjnych projektów. Większość aplikacji powiela rozwiązania znane i sprawdzone funkcjonalności. W konsekwencji trudnym zadaniem jest znalezienie wyróżniających się elementów wśród oferowanych możliwości tych produktów.

Proces definiowania wymagań w niewielkich (budżet max w granicach 100tyś euro - ?!) projektach jest w dużej mierze procesem twórczym. Często pojawienie się nowego wymagania jest inicjowane z kilku heterogenicznych źródeł. Nierzadko zdarza się również, że to samo wymaganie jest różnie komunikowane przez wiele źródeł. W niewielkich projektach, gdzie często jedna osoba, lub mały zespół odpowiedzialny jest za specyfikację wymagań, wymagania przybierają postać wiadomości email, rozmów telefonicznych, no-

tatek ze spotkań i formalnych dokumentów. Zadaniem osoby lub zespołu odpowiedzialnego za specyfikację wymagań, jest przefiltrowanie cząstkowych informacji oraz ekstrakcja i udokumentowanie wymagań. Żadne narzędzie (przynajmniej na razie) nie zastąpi skutecznie pracy człowieka nad wyłuskaniem wszystkich wymagań.

Rozdział 3

Technologie wykorzystane w implementacji

krótki wstęp

3.1 Z czego korzystałem z lotu ptaka - grails, java, tomcat, heroku, git, postgresql, javascript, biblioteka jsuml2, jquery

3.2 Framework Grails

3.2.1 Język Groovy

3.2.2 Spring Framework i Hibernate ORM

3.2.3 Framework Grails

3.3 Pozostałe technologie

3.3.1 Postgresql

3.3.2 Javascript, jQuery i biblioteka jsUml2

3.3.3 System kontroli wersji git i serwer heroku

3.3.4 Srodowisko programistyczne (linux, vim, tmux)

Rozdział 4

Opis prototypu - System Żeqmanager"

krótki wstęp

4.1 Filozofia i architektura systemu

4.2 Język UML w trakcie fazy analizy

4.3 Zbieranie wymagań w Reqmanager

4.4 Technologie webowe w kontekście RIA (zalety i wady przenoszenia sie do webu)

4.5 Implementacja

4.5.1 Diagramy przypadków użycia

4.5.2 Przetwarzanie wymagań (Mapowanie obiektów w JS na encje w bazie danych)

4.5.3 Generowanie dokumentacji

4.6 Problemy techniczne

4.7 Podsumowanie rozwiązania (zalety i wady)

Rozdział 5

Podsumowanie

krótki wstęp

5.1 Plan rozwoju

5.2 Zakończenie