Wprowadzenie do inżynierii wymagań

Adam Wojciechowski*

Poznan University of Technology, Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
Adam.Wojciechowski@put.poznan.pl

Streszczenie. Niniejsza praca stanowi wstęp do podręcznika z zakresu inżynierii wymagań. Omówione tu zostały podstawowe cechy procesu definiowania wymagań dla systemów informatycznych jak również własności, które wymagania powinny posiadać, aby we właściwy sposób mogły być interpretowane przez zespoły twórców oprogramowania. Wskazano również podstawowe zagrożenia wynikające z zaniedbań w zakresie gromadzenia i pielęgnacji wymagań.

1 Wstęp

Dzisiejsza rzeczywistość to świat, w którym człowiek często korzysta z systemów komputerowych. Dobrym przykładem na potwierdzenie tej opinii są systemy bankowe, systemy rezerwacji miejsc w samolotach, a także telefony komórkowe i komputery osobiste wspomagające zarządzanie i organizację pracy. Jednak komputery mogą mieć jeszcze większy wpływ na ludzkość – od nich niejednokrotnie może zależeć nasze życie. Na dowód tego, można wspomnieć o systemach wspomagających pilotów oraz kierowców, systemach utrzymujących czynności życiowe organizmu człowieka. Za jakość realizowanych usług oraz sposób, w jaki komputery mają „służyć" ludzkości, odpowiadają twórcy programów, które sterują ich działaniem. Oczywiście, działanie to musi być w pełni kontrolowane i określone dla wszystkich możliwych przypadków. Widać więc, że kluczową sprawą jest budowa dobrego oprogramowania, które spełnia oczekiwania użytkowników.

Obecnie, wiele zespołów programistów na całym świecie pracuje nad konstruowaniem systemów informatycznych. Pomimo, że dziedziny zastosowań są różne, a programiści porozumiewają się różnymi językami, to podczas wytwarzania oprogramowania często korzysta się z tych samych narzędzi, z tych samych technologii. Najważniejszy jest oczywiście cel, jaki każdy z zespołów chce i musi zrealizować: „*wytworzyć oprogramowanie dobrej jakości, na czas i w ramach wyznaczonego budżetu, zaspokajające rzeczywiste potrzeby klienta*" [8]. Istotnym stwierdzeniem w powyższym zdaniu są słowa „potrzeby klienta", a więc wymagania.

O tym, jak ważne są wymagania świadczyć może również fakt, iż w każdym z zaproponowanych modeli wytwarzania oprogramowania (np. model kaskadowy, cy-

* We współpracy z Tomaszem Dąbrowskim i Krzysztofem Szczech

kliczny, prototypowanie) [3, 6], zdefiniowano etap specyfikacji wymagań. Etap ten jest jednym z pierwszych w cyklu życia oprogramowania i poprzedza on między innymi takie fazy jak: projektowanie, produkcja, testowanie i wdrażanie. Pozyskiwanie „potrzeb klienta" stanowi zatem podstawę całego procesu budowy systemów informatycznych i od tego etapu uzależniony jest dalszy sposób realizacji projektu. Dobrze określone wymagania zapewniają lepszą jakość przyszłego oprogramowania. Z kolei wyższa jakość produktu końcowego to większa satysfakcja użytkownika.

Wiele problemów występujących podczas tworzenia oprogramowania, jest powiązanych z wymaganiami. Do podstawowych czynników, które sprawiają, że przedsięwzięcia informatyczne nie kończą się sukcesem można zaliczyć [8]:

- brak danych wejściowych od użytkownika – 13% wszystkich przedsięwzięć,
- niepełne wymagania i specyfikacje – 12% wszystkich przedsięwzięć,
- zmiany wymagań i specyfikacji – 12% wszystkich przedsięwzięć.

Z danych liczbowych wynika więc, że źle określone wymagania, błędna ich dokumentacja lub nieodpowiednie zarządzanie wymaganiami *stanowią najczęstszą przyczynę defektów systemów komputerowych. Ich liczba to prawie 1/3 wszystkich dostarczonych wad* [8].

Istotnym problemem realizacji przedsięwzięć informatycznych są również wysokie koszty naprawiania błędów wymagań w następnych cyklach życia oprogramowania. Niepełne i niespójne wymagania mogą stanowić przyczynę sprzeczności podczas projektowania, kodowania, testowania oraz pielęgnowania systemu. Koszty naprawiania błędów popełnionych w fazie pozyskiwania wymagań rosną wykładniczo wraz z etapem, na którym je wykryto. W celu redukcji tych wydatków, należy dążyć do szybkiej identyfikacji defektów i ich naprawy.

Problemy związane z zarządzaniem wymaganiami wskazują na potrzebę istnienia procesu, którego zadaniem jest usystematyzowanie wszelkich działań dotyczących wymagań. Proces *inżynierii wymagań* musi być ściśle powiązany z cyklem życia oprogramowania.

„Termin inżynieria (…) oznacza projektowanie i konstruowanie obiektów i urządzeń technicznych (…) W informatyce rolę budowli, obok sprzętu, odgrywa oprogramowanie. Inżynieria to praktyczne stosowanie praw i zasad pewnej dziedziny" [1]Inżynieria wymagań opisuje schemat pozyskiwania, dokumentacji i zarządzania wymaganiami. Jej wdrożenie w ramach konkretnego przedsięwzięcia ma na celu zminimalizowanie negatywnych konsekwencji opóźnień w realizacji przedsięwzięcia związanych z niedoskonałością procesu gromadzenia wymagań. Poprzez redukcję problemów na etapie opisu przyszłego sytemu, można zmniejszyć prawdopodobieństwo przekroczenia budżetu projektu. Stosowanie inżynierii wymagań może w znacznym stopniu przybliżyć zespół projektowy do osiągnięcia celu, jakim jest dostarczenie użytkownikom systemu komputerowego wysokiej jakości.

Proces specyfikacji i zarządzania wymaganiami nie jest łatwy. Temat ten jest przedmiotem wielu prac badawczych podejmowanych m.in. na Politechnice Poznańskiej [5, 7, 9, 10]. W literaturze znaleźć można wiele podejść do realizacji tego procesu [3, 8, 11, 14], a wiodące firmy produkujące oprogramowanie wykreowały własne modele lub zaadaptowały najlepsze dla nich schematy procedur pozyskiwania i zarządzania wymaganiami. Dobry materiał dydaktyczny a jednocześnie przewodnik dla organizacji definiujących i wdrażających procedury zarządzania wymaganiami jest

praca [12], w której autorzy przedstawili 66 dobrych praktyk inżynierii wymagań. W zaproponowanym modelu praktyki zostały podzielone na trzy grupy: podstawowe, pośrednie, zaawansowane, a w zależności od stopnia ich realizacji w danej organizacji, zespoły otrzymują od 0 do 3 punktów. Suma punktów uzyskiwanych za realizację praktyk z poszczególnych grup stanowi podstawę dla oceny dojrzałości procesów inżynierii wymagań w sensie trójwarstwowego modelu. Mechanizm ten pozwala na wzajemne porównywanie organizacji w sensie dojrzałości ich procesów zarządzania wymaganiami i jednocześnie wytycza kierunek poprawy procesów programistycznych.

2 Podstawy inżynierii wymagań

2.1 Wymaganie

Organizacja Institute of Electrical and Electronics Engineers (w skrócie IEEE) w dokumencie [4] definiuje wymaganie jako:

1. Warunek lub możliwość potrzebną przez użytkownika do rozwiązania lub osiągnięcia konkretnego celu.
2. Warunek lub możliwość, która musi mieć system lub komponent systemu, aby spełnić warunki kontraktu, standardu, specyfikacji lub innego narzuconego dokumentu.
3. Udokumentowana reprezentacja warunku lub możliwości określonej w punkcie 1 lub 2.

Z kolei Sommerville i Sawyer definiują wymaganie jako: „Wymagania są specyfikacją tego, co powinno zostać zaimplementowane w systemie. Są one opisem tego jak system powinien się zachować, lub opisem właściwości systemu. Mogą one być ograniczeniem podczas procesu wytwarzania systemu" [12].

Można zatem stwierdzić, że nie ma uniwersalnej definicji, czym jest wymaganie, a każda z definicji przedstawia nieco odmienna wizję. Wydaje się jednak, że na wyższym poziomie za wymaganie można uznać własność, która jest cenna z punktu widzenia użytkownika.

Poniżej przedstawiono kilka przykładowych wymagań.

„System ma generować ostrzeżenia w kolorze czerwonym."

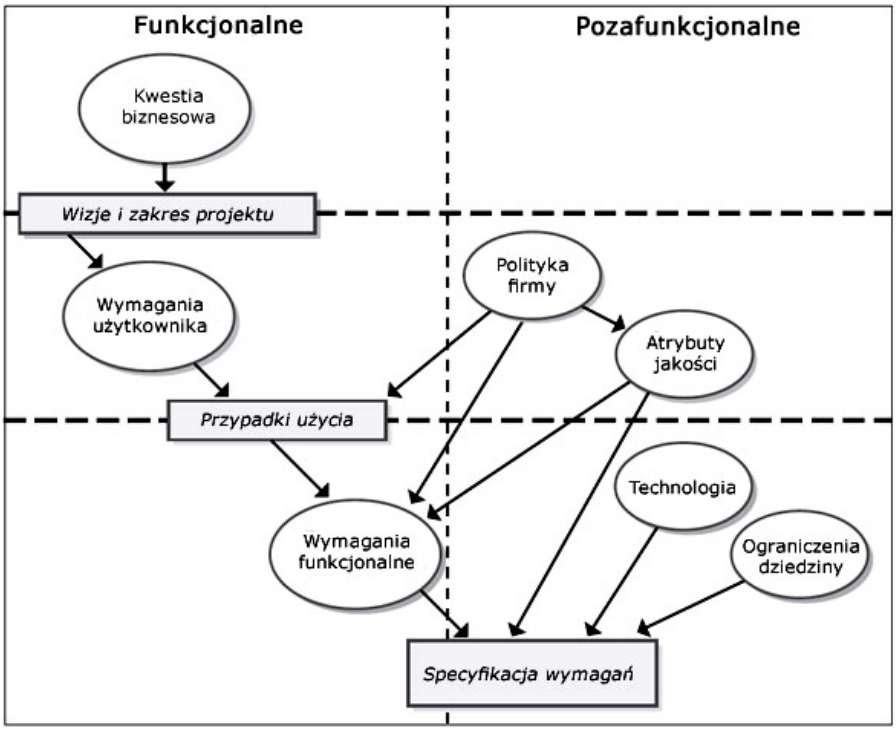
„Aby móc dodać nowego użytkownika, administrator musi być zalogowany."

„Użytkownik może wyszukać produkty w sklepie internetowym według atrybutów."

2.2 Poziomy wymagań

Dyscyplina związana ze specyfikacją wymagań definiuje dwa podstawowe typy wymagań: funkcjonalne (ang. *funcional*) i pozafunkcjonalne (ang. *nonfunctional*). Dodatkowo, osoba odpowiedzialna za pozyskiwanie wymagań od klienta powinna być

świadoma istnienia wymagań biznesowych lub też inaczej mówiąc kwestii biznesowej (ang. *business case*) oraz wymagań użytkownika (scenariusze użytkownika, ang. *use cases*). Na rysunku 2-1 zaprezentowano schematycznie zależności pomiędzy wspomnianymi wyżej rodzajami wymagań. W elipsach przedstawiono informację dotyczącą wymagań, w prostokątach dotyczącą dokumentów powiązanych z daną informacją.



Rysunek 2-1. Schemat zależności między wymaganiami (źródło [14]).

Kwestia biznesowa opisuje na dość ogólnym poziomie cel, który chce osiągnąć firma lub klient, poprzez realizacje danego przedsięwzięcia. Cel ten jest w pewnym stopniu odpowiedzią na pytanie, dlaczego klient lub organizacja potrzebuje danego systemu. Należy poznać aktualną sytuacje w firmie oraz zrozumieć, w jakim stopniu ta sytuacja zmieni się poprzez wykorzystanie projektowanego systemu. Wymagania biznesowe pochodzą przede wszystkim od sponsora danego projektu, od reprezentanta użytkowników końcowych. Dobrym pomysłem jest spisanie wizji (wraz z proponowanymi przez klienta rozwiązaniami) oraz rozmiaru projektu, w dokumencie „Wizje i zakres projektu". Jest on podstawowym źródłem, podczas określania scenariuszy użytkownika.

Wymagania użytkownika opisują to, co użytkownicy będą mogli wykonać w systemie. Scenariusza użytkownika przedstawione są w formie opisu kolejnych kroków działań podejmowanych przez użytkownika i wariantów odpowiedzi systemu. Przy-

kładem może być scenariusz użytkownika „Dodanie produktu”, w ramach którego opisane są kolejne działania związane z dodaniem produktu, np. zalogowanie się do systemu jako administrator, wpisanie atrybutów nowego produktu, weryfikacja poprawności wprowadzonych danych, dodanie produktu. Scenariusze użytkownika mogą być spisane w dokumencie „Przypadki użycia”. Dokument ten stanowi podstawę do określenia szczegółowych wymagań w systemie.

Wymagania funkcjonalne specyfikują dokładnie to, co system powinien robić, jego funkcjonalność, którą zespół programistów powinien zrealizować w formie programu komputerowego. Funkcjonalność ta jest niezbędna dla użytkownika w celu realizacji konkretnego zadania, oraz jest niezbędna w celu zaspokojenia kwestii biznesowej. Przykładem wymagania funkcjonalnego może być zdanie: „System powinien umożliwiać dodanie nowego produktu”.

Wymagania pozafunkcjonalne nie mają wpływu na funkcjonalność systemu. Nakładają one ograniczenie na sposób, w jaki sposób wymagania funkcjonalne będą zrealizowane. Wymagania te mogą wynikać między innymi bezpośrednio:

- z technologii, w której ma zostać zrealizowany system,
- z ograniczeń dziedziny systemu,
- z zasad, według których działa klient lub firma (polityka firmy),
- z polityki jakości.

Przykładem wymagania pozafunkcjonalnego może być zdanie: „Dodanie nowego produktu ma trwać nie dłużej niż 5 sekund”.

Wymagania funkcjonalne i pozafunkcjonalne powinny zostać spisane w dokumencie „Specyfikacja wymagań” (ang. s*oftware requirements specification*). Dokument ten jest oficjalna listą wymagań projektowanego systemu dla klienta, użytkowników końcowych oraz zespołu programistów

Wymagania nie mogą zawierać same w sobie sposobu zaprojektowania lub też metody implementacyjnej. Ich treść nie może również odnosić się do informacji dotyczącej planowania bądź też testowania. Wszystkie wymienione tutaj kwestie powinny być zapisane w oddzielnym dokumencie, tak, aby treść wymagań dotyczyła tylko i wyłącznie tego, co ma zostać wytworzone w ramach danego projektu.

2.3 Dobre wymaganie

Wiedząc już, co to jest wymaganie, znając ich rodzaje, warto skupić uwagę na zagadnienia dobrego formułowania treści wymagań. Okazuje się, że nie ma jednego, najlepszego sposobu napisania wymagania. Wiele firm stosuje własne praktyki, notacje i schematy do ich opisu. Również język użyty do opisania wymagania może być na różnym poziomie i zależy od osoby, która jest jego źródłem. I tak na przykład, jeżeli dane wymagania zostało pozyskane od osoby posiadającej wiedzę inżynierską, bardzo prawdopodobne jest, że w treści wymagania wystąpi wiele charakterystycznych zwrotów dla danej dziedziny. Z kolei, jeżeli źródłem wymagania był „zwykły” użytkownik końcowy, wymaganie będzie opisane za pomocą języka naturalnego, prostego. Wiele wymagań ma właśnie postać tych ostatnich. Dobrym pomysłem jest uzupełnienie tego typu opisu przez diagramy, schematy za pomocą, których lepiej jest zrozumieć dane wymaganie.

Trudno jest arbitralnie stwierdzić, na jakim poziomie szczegółowości powinny być opisane wymagania. Jest to kwestia indywidualna, która w dużej mierze zależy od modelu przyjętego w danej firmie. Szczegółowość wymagań zależy także od tego, czy podstawą kontraktu z klientem będzie dokument wymagań i system, który powstanie na podstawie tego dokumentu. Przykładowo, jeżeli firma realizuje wewnętrzny projekt, to wymagania mogą być spisane ogólnikowo (np. same idee), a następnie mogą być one uszczegółowiane podczas etapu implementacji. Z drugiej strony natomiast, jeżeli realizuje się projekt dla firmy zewnętrznej, zalecane jest dokładne określenie wymagań, tak, aby wiadomo było, jakie funkcje przyszły system ma zawierać.

Z każdym wymaganiem wiąże się oczywiście szereg atrybutów. Atrybuty te, mogą być podstawą do określenia cech, jakim powinno charakteryzować się wymaganie. W pracy [14] przedstawiono kilka właściwości, które mogą być dobrą miarą sprawdzenia jakości naszego wymagania. Właściwości te są przedstawione poniżej.

Kompletność

Każde wymaganie musi dokładnie i w pełni określać funkcjonalność, która powinna być dostarczona. Musi ono zawierać niezbędne informacje dla programisty, umożliwiające mu zaprojektowanie, a następnie zaimplementowanie tej funkcjonalności.

Poprawność

Każde wymaganie musi odpowiednio definiować funkcjonalność, która ma zostać stworzona. Poprawność wymagania może być sprawdzona między innymi poprzez weryfikację jego treści przez osobę, która była jego autorem.

Wykonalność

Musi istnieć możliwość zaimplementowania każdego wymagania przy wykorzystaniu dostępnych narzędzi, technologii uwzględniając przy tym ograniczenia systemu i jego środowiska. Aby uniknąć zdefiniowania niewykonalnych wymagań, zalecana jest współpraca programisty z analitykiem na etapie pozyskiwania wymagań. Programista posiada wiedzę pozwalająca stwierdzić, co może być zrobione w sensie technologii, a co może wymagać nakładu większych kosztów lub być niewykonalne przy przyjętych ograniczeniach. Podejście przyrostowe, bądź też prototypowanie jest dobrym sposobem, aby skutecznie ocenić wykonalność konkretnych wymagań.

Konieczność

Każde wymaganie powinno dokumentować czynniki, które wpływają na fakt, że klient potrzebuje danej funkcji systemu, że wymaganie musi być zgodne z zewnętrznymi wymaganiami systemowymi lub standardami. Wymaganie powinno być pozyskane od źródła, które ma uprawnienia do specyfikowania tego wymagania.

Priorytet

Każde wymaganie powinno mieć przypisany priorytet. Dzięki temu, można ocenić jak ważna dla klienta jest funkcjonalność związana z danym wymaganiem w projekcie. Oczywiście priorytety mogą być również ustalone w obrębie kolejnego inkrementu, co umożliwia ocenienie, które wymagania powinny być zrealizowane w następnym przyroście. Podkreślmy, że jeżeli wszystkie wymagania mają określony priorytet jako wysoki, trudno jest podążać według budżet projektu, należy brać pod uwagę opóźnienia w realizacji, dodanie nowych wymagań, itp.

Jednoznaczność

Treść wymagania powinna być napisana w taki sposób, iż wszyscy czytelnicy mają taką samą jego interpretacje. Dlatego też wymaganie powinno być napisane w sposób

prosty, zwięzły, bezpośredni, uwzględniając przy tym dziedzinę problemu. Po przeczytaniu wymagania, powinniśmy móc powiedzieć, czego dane wymaganie dotyczy oraz jaką funkcjonalność ma realizować. Pomocne może być wyspecyfikowanie wszystkich charakterystycznych dla danej dziedziny zwrotów w dokumencie „Słownik" (bądź też w sekcji słownikowej dokumentu wymagań).

Weryfikowalność

Do każdego wymagania powinna istnieć możliwość ułożenia kilku testów lub użycia innych technik weryfikacji (inspekcje, demonstracje). Ma to umożliwić sprawdzenie, czy produkt poprawnie implementuje dane wymaganie. Jeżeli wymaganie nie jest weryfikowalne, należy sprawdzić czy pozyskano wszystkie niezbędne szczegóły wymagania.

Dobrze zdefiniowane wymagania, które posiadają wszystkie wyżej wymienione cechy nie zapewnią nam jednak od razu końcowego sukcesu, w postaci zakończenia projektu. Należy zauważyć, że zbiór naszych wymagań powinien charakteryzować się między innymi takimi cechami: kompletność, spójność, łatwość modyfikacji, zdolność do śledzenia powiązań (ang. traceability).

Kompletność oznacza to, iż nasze wymagania w pełni definiują system. Żadna z istotnych informacji, bądź też żadne z wymagań nie zostały pominięte.

Spójność świadczy z kolei o tym, że określone przez nas wymagania nie są ze sobą konflikcie. Konflikt ten nie występuje również pomiędzy wymaganiami a sprawą biznesowa (ang. business case), systemem i jego środowiskiem. Dobrze zdefiniowane wymagania wzajemnie się uzupełniają tworząc tym samym spójną całość, czyli projektowany system.

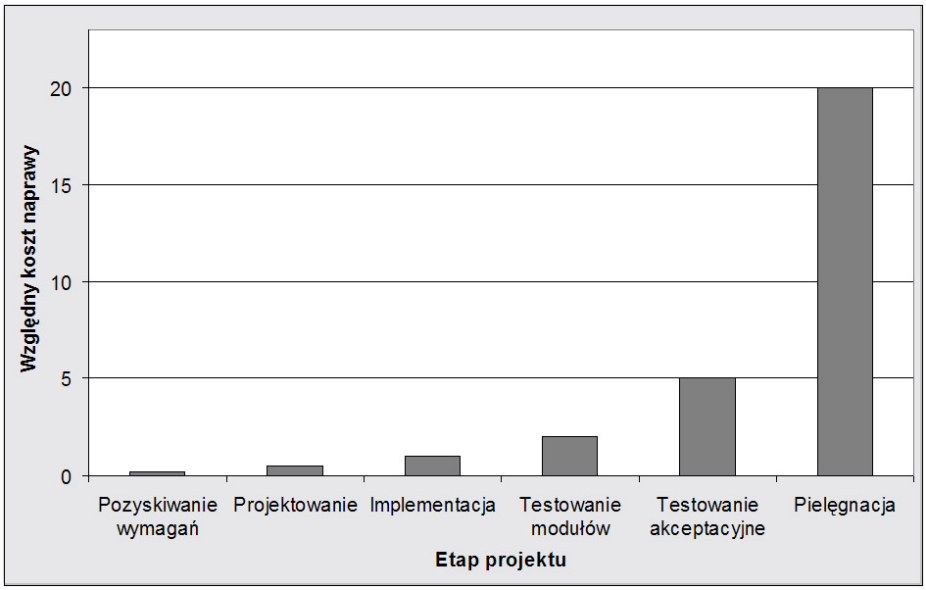
Łatwość modyfikacji zapewnia, że treść dokumentu wymagań może być w prosty sposób zmieniona, a historia zmian jest zapisana. Każde z wymagań powinno posiadać unikalny identyfikator, dzięki czemu możemy się w łatwy sposób odwoływać się do niego. Dobrym pomysłem jest również koncepcja „jedno wymaganie, jedna kartka", to znaczy treść każdego z wymagań powinna być spisana na odrębnej kartce.

Dzięki śledzeniu powiązań między wymaganiami, oraz wymaganiami a komponentami systemu, możemy w szybki sposób określić, jaki wpływ na system ma zmiana konkretnego wymagania. Powiązania umożliwiają pogrupowanie w zbiory wymagań, które kooperują ze sobą, zapewniając tym samym określoną funkcjonalność. W pracy [12] zaproponowano stworzenie macierzy powiązań, która ukazuje odpowiednie zależności. Macierz ta posiada tyle wierszy i kolumn, ile jest wymagań. W najprostszej formie, jeżeli wymaganie w wierszu W1 zależy od wymagań w kolumnach K3, K4, to na przecięciu wiersza z tymi kolumnami stawiamy wcześniej ustalony znak (np. *). Dobrym rozwiązaniem jest przechowywanie wszystkich wymagań w bazie danych lub innym narzędziu wspomagającym zarządzanie wymaganiami (np. Rational RequisitePro).

2.4 Koszty błędów wymagań

Konsekwencją źle ułożonych wymagań jest ponowna praca nad czymś, co zostało już raz zrobione. Według doświadczeń opisanych w pracy [2], przerabianie pochłania zwykle od 30% do 70% budżetu typowego przedsięwzięcia. Na sumę tą w dużym

stopniu wpływają błędy wymagań, które stanowią między 70% a 85% jej wartości [8]. Nakład pracy poświęcony na przerabianie może być zredukowany, dzięki unikaniu i wykrywaniu błędów wymagań w jak najwcześniejszym etapie projektu. Jak przedstawiono na rysunku 3-2, naprawienie defektów wymagań w późniejszym okresie, kosztuje znacznie więcej niż usunięcie ich już na etapie pozyskiwania oczekiwań od klienta.



Rysunek 2-2. Względne koszty naprawy błędów.

Potwierdzeniem powyższej tezy mogą być badania, jakie zostały przeprowadzone w takich firmach jak IBM, Hewlett Packard. Ich celem było oszacowanie kosztów naprawy błędów, który wystąpiły na różnych etapach cyklu życia przedsięwzięcia. Wnioskiem eksperymentu było stwierdzenie, że koszt wykrycia i naprawienia błędu podczas etapu określania wymagań jest pięcio-, dziesięciokrotnie mniejszy [8].

Jest wiele czynników, które mają wpływ na jakość zebranych przez nas wymagań. Nie sposób wymienić jest wszystkich, lecz można zwrócić czytelnikowi uwagę na parę kluczowych.

Niewystarczające zaangażowanie klienta

Często zdarza się, że klient nie zdaje sobie sprawy, z wagi etapu pozyskiwania wymagań. Efektem tego może być, na przykład, brak możliwości komunikacji z użytkownikami końcowymi, czyli z osobami, które faktycznie mają używać przyszłego systemu. Zdarza się również, iż reprezentant klienta tak naprawdę nie wie, czego oczekuje od systemu, a tym samym jego wymagania nie są pełne i spójne.

Podatność do zmian

W prawie w każdym z projektów, jest rzeczą oczywistą, że wymagania się zmieniają. Wraz z ich modyfikacjami, bardzo często zmienia się plan projektu, tym samym powodując, że termin oddania systemu zostaję przesunięty do przodu w czasie.

Jako przyczyny takiego stanu, można tu wskazać ciągłość zmian wymagań przez klienta oraz złe zrozumienie wymagań przez programistów, co w konsekwencji prowadzi do błędnego oszacowania czasu wykonania.

Niejednoznaczność wymagań

Niejednoznaczność wymagań jest wynikiem braku precyzji oraz szczegółów podczas etapu pozyskiwania. Jedną z jej oznak jest fakt, że różne osoby czytające wymagania różnie je rozumieją i interpretują. Niejednoznaczność prowadzi do rozczarowań klienta podczas prezentowania utworzonego do tej pory systemu – powstały produkt jest inny od jego wizji. Niejednoznaczność to również stracony czas przez programistów, którzy zaimplementowali błędne rozwiązanie. Pewnym sposobem na jej wyeliminowanie jest etap walidacji wymagań. Podczas jego trwania, osoby reprezentujące różne dziedziny wiedzy (a więc posiadające odmienne spojrzenie), przeglądają dokumenty wymagań. Ich wizja systemu, wynikająca z tych dokumentów, powinna być zbieżna.

Dodanie funkcjonalności

Sytuacja ta ma miejsce, jeżeli programista dodaje funkcjonalności do wymagań, tzn. zaimplementowane wymaganie zawiera więcej niż to zakładano wcześniej. Często zdarza się, że klient nie zwraca uwagi na tą „dodatkową" funkcjonalność, a więc czas spędzony nad jej implementacją jest czasem straconym. Zalecane jest jednak zasugerowanie pewnych idei klientowi, pewnych dodatkowych rozwiązań. Jeżeli uzna on, że warto je wdrożyć, wtedy też to powinno mieć miejsce. Podczas prac implementacyjnych programiści powinni wykazywać się „prostotą" i „łatwością rozwiązań".

Minimalna specyfikacja

Czasami zdarza się, że klient przedstawia tylko wizję systemu, a następnie oczekuje od analityków jej rozwinięcia poprzez przedstawienie kompletnych wymagań. Niestety, nie jest to korzystne, ani dla programistów (mogą pracować przy niewłaściwych założeniach), ani dla klienta (otrzyma system, który jest inny od jego wizji).

Przeoczenie grupy użytkowników końcowych

Podczas realizacji projektu ważnym elementem jest zdefiniowanie wszystkich typów użytkowników końcowych. Pominięcie pewnej klasy użytkowników może spowodować, zmniejszenie liczby źródeł wymagań. Tym samym, może to doprowadzić do braku określonej funkcjonalności niezbędnej dla prawidłowego działania systemu (z punktu widzenia pominiętych użytkowników).

Błędne planowanie

Błędne (np. zbyt optymistyczne) oszacowanie kosztów realizacji wymagania może prowadzić do wielu niepożądanych sytuacji. Może być ono wynikiem częstych zmiany wymagań, przeoczenia jakiegoś wymagania, niewystarczającej komunikacji z klientem. Podczas oszacowywania zalecane jest podanie najlepszego, najbardziej prawdopodobnego, najgorszego przypadku.

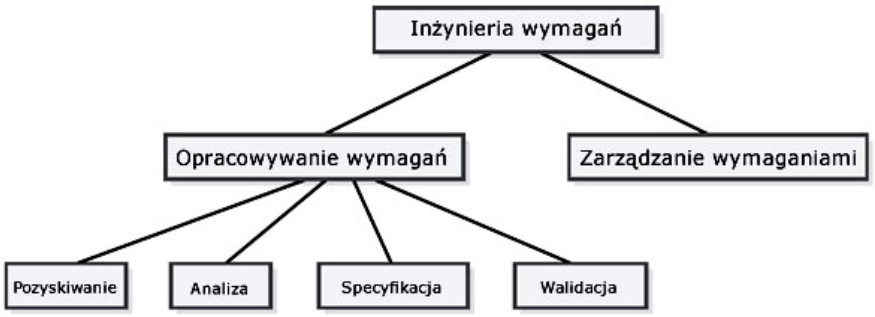
2.5 Inżynieria wymagań

Jedną z definicji terminu „inżynieria wymagań" możemy znaleźć w opracowaniu [12]. Według autorów „inżynieria wymagań jest relatywnie nowym terminem, który

został wymyślony w celu nazwania wszystkich działań związanych z pozyskiwaniem, dokumentacją i utrzymywaniem zbioru wymagań dla systemu komputerowego."

Inżynieria wymagań jest to, więc, proces w ramach, którego zostają zidentyfikowane wszystkie ograniczenia oraz wymagania, które ma zawierać przyszły system. Ponadto, w ramach tego procesu prowadzona zostaje dokumentacja zbioru tych wymagań. Kontekście powyższych definicji, słowo inżynieria powinno być rozumiane jako systematyczna i powtarzalna technika, która zapewnia nam, że wymagania są kompletne, spójne, itd.

Graficzny opis procesu inżynierii wymagań został zaprezentowany na rysunku 2-3.



Rysunek 2-3. Komponenty inżynierii oprogramowania (źródło [14]).

Na inżynierię wymagań składają się dwa podstawowe komponenty: opracowywanie wymagań i zarządzanie wymaganiami. Oczywiście ten drugi jest bezpośrednio uzależniony od istnienia tego pierwszego.

Podczas etapu *pozyskiwania* wymagań, wymagania są definiowane na podstawie rozmowy z użytkownikami końcowymi, dokumentów systemu, wiedzy dziedzinowej przyszłego systemu oraz rynku.

Kolejnym etapem opracowywania wymagań jest ich szczegółowa *analiza* i ewentualna negocjacja z różnymi użytkownikami końcowymi, tak, aby zdecydować, które wymagania mają zostać zrealizowane.

Po określeniu zbioru wymagań, proces inżynierii wymagań zaleca dokładne wyspecyfikowanie każdego z nich, a następnie spisanie ich w dokumencie (etap *specyfikacji*).

Walidacja kończy proces opracowywania wymagań. Podczas tego etapu, należy dokładnie sprawdzić wymagania pod kątem ich spójności i kompletności.

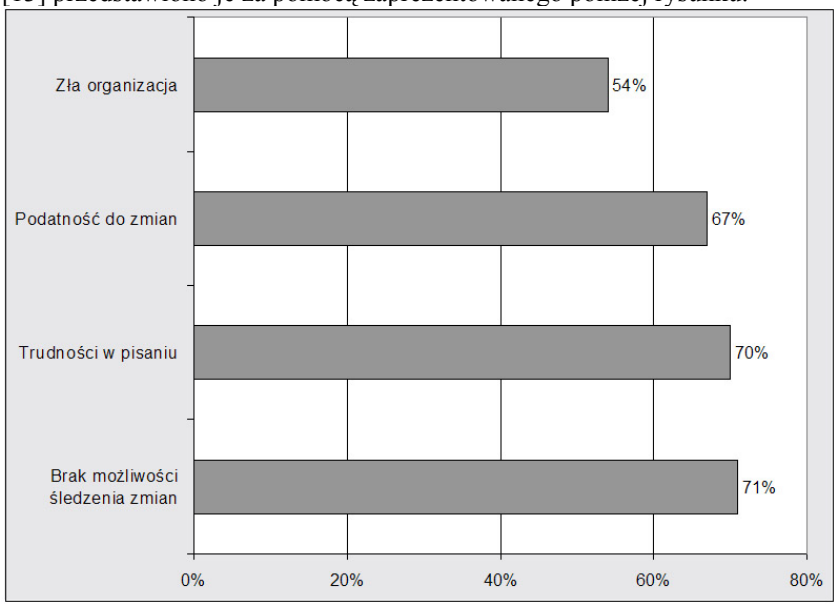
Aby wspierać obszar opracowywania wymagań, należy wdrożyć komponent zarządzania wymaganiami, który będzie odpowiedzialny za zmiany i modyfikacje wymagań. Ponadto, w ramach zarządzania wymaganiami przeprowadzane są między innymi takie czynności jak:

- przegląd proponowanych zmian wymagań i ocena wpływu tych zmian na cały projekt przed ich zatwierdzeniem,
- kontrolowanie wprowadzenia do projektu zaproponowanych zmian wymagań,

- śledzenie każdego wymagania pod kątem jego zgodności z projektem, kodem, przypadkami testowymi,
- śledzenie statusu wymagania podczas projektu.

2.6 Problemy inżynierii wymagań

Wydaje się, że proces, jakim, jest inżynieria wymagań, powinien usystematyzować prace podczas projektu. W rzeczywistości jest jednak trochę inaczej. Wdrażając tą praktykę, na przykład w ramach swojego projektu, często pojawiają się nowe komplikacje, które mogą stwarzać problem w jej pomyślnym wykorzystaniu. W pracy [13] przedstawiono je za pomocą zaprezentowanego poniżej rysunku.



Rysunek 2-4. Najpowszechniejsze problemy inżynierii wymagań (źródło [13]).

Jeżeli problemy te wystąpią wraz z nieodpowiednim procesem i niewłaściwymi umiejętnościami zarządzania wymaganiami oraz brakiem łatwych w użyciu narzędzi wspomagających, wiele zespołów programistycznych nie ma możliwości zarządzać dobrze wymaganiami.

Literatura

[1] Begier B., *Inżynieria oprogramowania – problematyka jakości*. Wydawnictwo Politechniki Poznańskiej, 1999; ISBN 83-7143-408-1.

[2] Boehm B. W., Papaccio P. N., *Understanding and controling Software Cost*. IEEE Press, 1988; ISSN 0098-5589.

[3] Górski J., *Inżynieria oprogramowania w projekcie informatycznym*. Wydawnictwo Informatyki Mikom, 2000; ISBN 83-7279-028-0.
[4] *IEEE standard of Software Engineering Terminology*. IEEE Std 610. IEEE press, 1990.
[5] Jasiński M., Nawrocki J., *Planowanie zakresu przedsięwzięcia na etapie inżynierii wymagań*, III Krajowa Konferencja Inżynierii Oprogramowania KKIO 2001. Wydawnictwo Informatyki Mikom, Warszawa, 2001, 213-224.
[6] Jaszkiewicz A., *Inżynieria oprogramowania*. Wydawnictwo Helion, 1997; ISBN 83-7197-007-2.
[7] Kowalczykiewicz K., Nawrocki J., *Dokumentowanie wymagań w języku XML*, III Krajowa Konferencja Inżynierii Oprogramowania KKIO 2001. Wydawnictwo Informatyki Mikom, Warszawa, 2001, 83-92.
[8] Leffingwell D., Widrig D., *Zarządzanie wymaganiami*. Wydawnictwo Naukowo-Techniczne, 2003; ISBN 83-204-2805-5.
[9] Nawrocki J., Jasiński M., Walter B., Wojciechowski A., *Extreme Programming Modified: Embrace Requirements Engineering Practices*, 10th IEEE Joint International Requirements Engineering Conference, RE'02, Essen, Los Alamitos, 303-310.
[10] Nawrocki J., Walter B., Jasiński M., Wojciechowski A., *Programowanie Ekstremalne i inżynieria wymagań*, IV Krajowa Konferencja Inżynierii Oprogramowania KKIO 2002. Wydawnictwo Nakom, Poznań 2002, 201-212.
[11] Sommerville I., *Inżynieria oprogramowania*. Wydawnictwo Naukowo-Techniczne, 2003; ISBN 83-204-2795-9.
[12] Sommerville I., Sawyer P., *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons Ltd., 1997; ISBN 0-471-97444-7.
[13] *Using Rational RequistePro*, Version 2001. Rational Software Corporation, 2001.
[14] Wiegers K. E., *Software Requirements*, 2nd Edition. Microsoft Press, 2003; ISBN 0-7356-1879-8.