Inżynieria Oprogramowania 1

Inżynieria wymagań

Arkadiusz Chrobot

Katedra Systemów Informatycznych, Politechnika Świętokrzyska

Kielce, 13 listopada 2024

Plan

Motto

„Dział sprzedaży cybernetycznej korporacji Syriusza definiuje robota jako «przyjaciela, z którym przyjemnie przebywać». *Autostopem przez galaktykę* definiuje dział sprzedaży cybernetycznej korporacji Syriusza jako «sforę bezmózgich szaleńców, którzy pierwsi pójdą pod ścianę, gdy nadejdzie rewolucja»…”

— Douglas Adams, *„Autostopem przez galaktykę”*

Definicje (tłumaczenia własne)

Inżynieria wymagań

Inżynieria wymagań jest jedną z pierwszych czynności w inżynierii oprogramowania, związaną ze zbieraniem, analizowaniem, zatwierdzaniem i zarządzaniem wymaganiami.

Wymaganie

Wymaganie jest orzeczeniem określającym charakterystykę lub ograniczenie funkcyjne, operacyjne lub projektowe produktu lub procesu, które jest jednoznaczne, weryfikowalne (ang. *testable*) lub mierzalne i konieczne, aby produkt lub proces został przyjęty (przez klienta lub wewnętrzny dział jakości) [1].

Interesariusz

Interesariusz (ang. *stakeholder*) jest osobą, grupą osób, organizacją lub innym podmiotem, który ma bezpośredni lub pośredni wpływ (udział) na oprogramowanie [1].

Źródła wymagań

Istnieje wiele źródeł wymagań:

dziedzina zastosowania to docelowy obszar przeznaczenia oprogramowania,

regulacje mogą być wewnętrzne lub zewnętrzne (np. prawne),

interesariusze (włączając w to wytwórców oprogramowania) są głównym źródłem wymagań.

Klasyfikacja wymagań

Wymagania mogą być klasyfikowane na kilka sposobów. Najbardziej podstawowy podział to:

Wymagania funkcjonalne to te wymagania, które określają jakiego rodzaju funkcje lub usługi będzie dostarczało oprogramowanie (innymi słowy *co* będzie robiło).

Wymagania niefunkcjonale nazywane również *pozafunkcjonalnymi*, definiują ograniczania dla rozwiązań (innymi słowy określają *jak* oprogramowanie powinno dostarczać usługi). Wymagania niefunkcjonalne mają bezpośredni wpływ na jakość oprogramowania.

Wymaganie funkcjonalne — przykład

System multimedialny musi umożliwiać użytkownikowi dostrojenie jasności obrazu (skala: 1–100).

Wymaganie niefunkcjonalne — przykład

Czas reakcji systemu Anti-Lock Braking System (ABS) musi być poniżej 200 ms.

Klasyfikacja wymagań

Wymagania mogą być także klasyfikowane ze względu na poziom szczegółowości:

Wymagania biznesowe są najbardziej istotne i ogólne (wysokopoziomowe). Ich źródłem jest dziedzina zastosowania oprogramowania. Rzadko ulegają zmianom.

Wymagania użytkowników definiują własności i usługi oprogramowania, które są niezbędne dla jego użytkowników. Są one bardziej szczegółowe niż wymagania biznesowe.

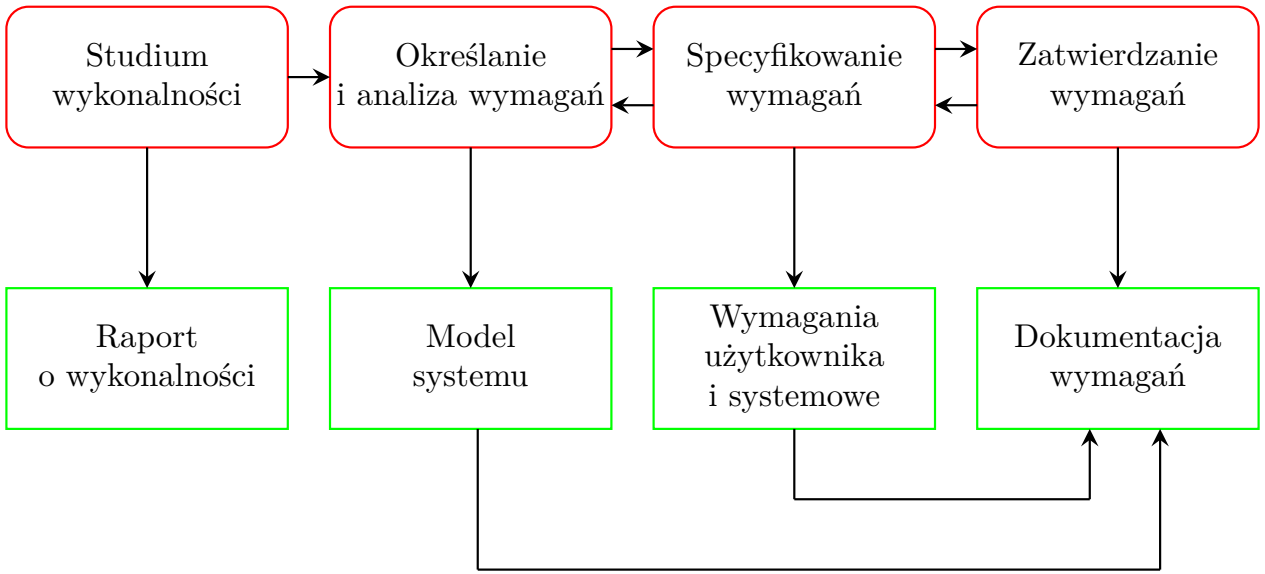
Wymagania systemowe są najbardziej niskopoziomowymi wymaganiami. Definiują szczegóły niezbędne dla interesariuszy do zrozumienia i zaaprobowania tego, co oprogramowanie powinno dostarczać.

Złożoność inżynierii wymagań

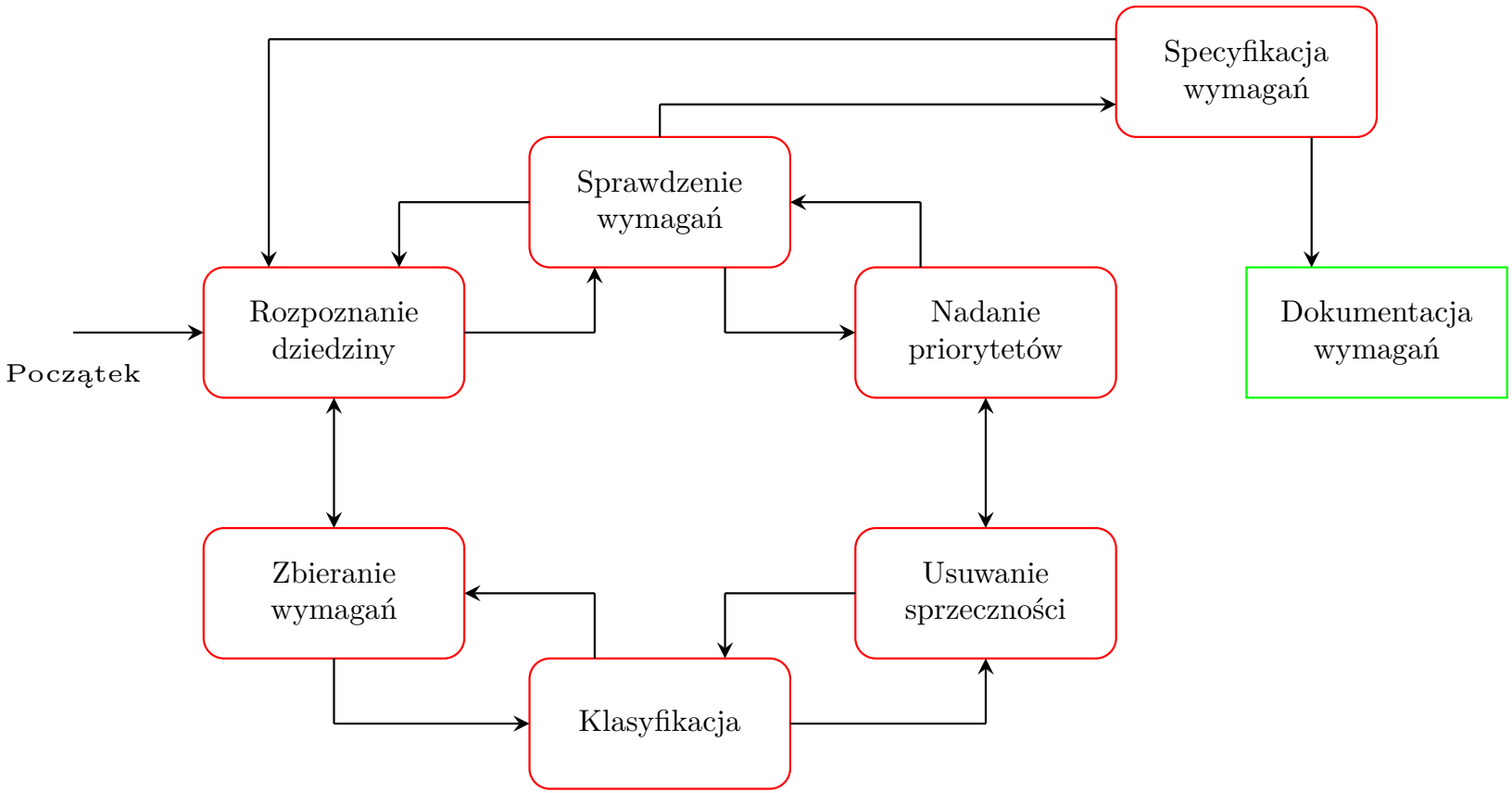
Inżynieria wymagań jest złożonym zagadnieniem, ponieważ [2]:

- interesariusze często nie wiedzą, co oprogramowanie powinno robić;
- interesariusze mogą mieć nierealne oczekiwania;
- interesariusze definiują wymagania używając słów i określeń z języka swojej dziedziny, które mogą mieć ukryte znaczenia, albo nie być wstępnie zrozumiałe;
- różni interesariusze mogą mieć różne oczekiwania, co może prowadzić do sprzeczności wymagań.

Ogólny proces inżynierii wymagań



Określanie i analiza wymagań



User stories

User stories są głównym narzędziem określania i analizy wymagań stosowanym w metodach zwinnych. Całość tego procesu zaczyna się jednak od *inicjatywy* (ang. *initiative*) (odpowiadającej *celowi produktu*), która jest dzielona na *epiki* (ang. *epics*), a te na *user stories*. One, z kolei, są źródłem *zadań* (ang. *tasks*), które muszą koniecznie być wykonane, aby te *user stories* zostały zaimplementowane [3].
W ujęciu formalny, *user story* jest uchwyceniem opisu własności oprogramowania z perspektywy użytkownika końcowego [4]. Jednakże, *user story* nie jest kompletnym opisem wymagania. To raczej punkty wyjściowy do dyskusji z *product ownerem* (lub innym przedstawicielem interesariuszy) o szczegółach wymagania i do określenia *kryterium akceptacji* (ang. *acceptance criteria*) pozwalającego sprawdzić wytwórcom i interesariuszom, czy wymaganie zostało poprawnie zaimplementowane.
Kolejny slajd jest podsumowaniem całościowego opisu pojedynczego wymagania w metodach zwinnych. User stories zostały opracowane na potrzeby programowania ekstremalnego, ale obecnie są stosowane w Scrum oraz innych metodach zwinnych.

User stories

Karta	Konwersacja	Potwierdzenie
User stories są zapisane na kartach zawierających oszacowania, notatki, itd.	Szczegóły user story są odkrywane i określane podczas rozmowy z *product ownerem*.	Testy akceptacyjne weryfikują implementację.

Wymaganie

User story

Struktura user story

Jako <określony użytkownik, persona, rola>, **chcę** <potrzeba>, **aby** <problem do rozwiązania, cel do osiągnięcia>.
Warunki akceptacji/spełnienia

Alternatywna struktura user story (5W)

Jako <kto> <kiedy> <gdzie>, **chcę** <co>, **ponieważ** <dlaczego>.
Warunki akceptacji/spełnienia

Alternatywna struktura user story (BDD)

Jako <kto> <kiedy> <gdzie>, **chcę** <co>, **ponieważ** <dlaczego>.
Scenariusz 1: <nazwa scenariusza>
Given <pierwszy warunek początkowy>
(**And** <drugi warunek początkowy>)
When <wyzwalacz scenariusza>
Then <expected result>

Scenariusze

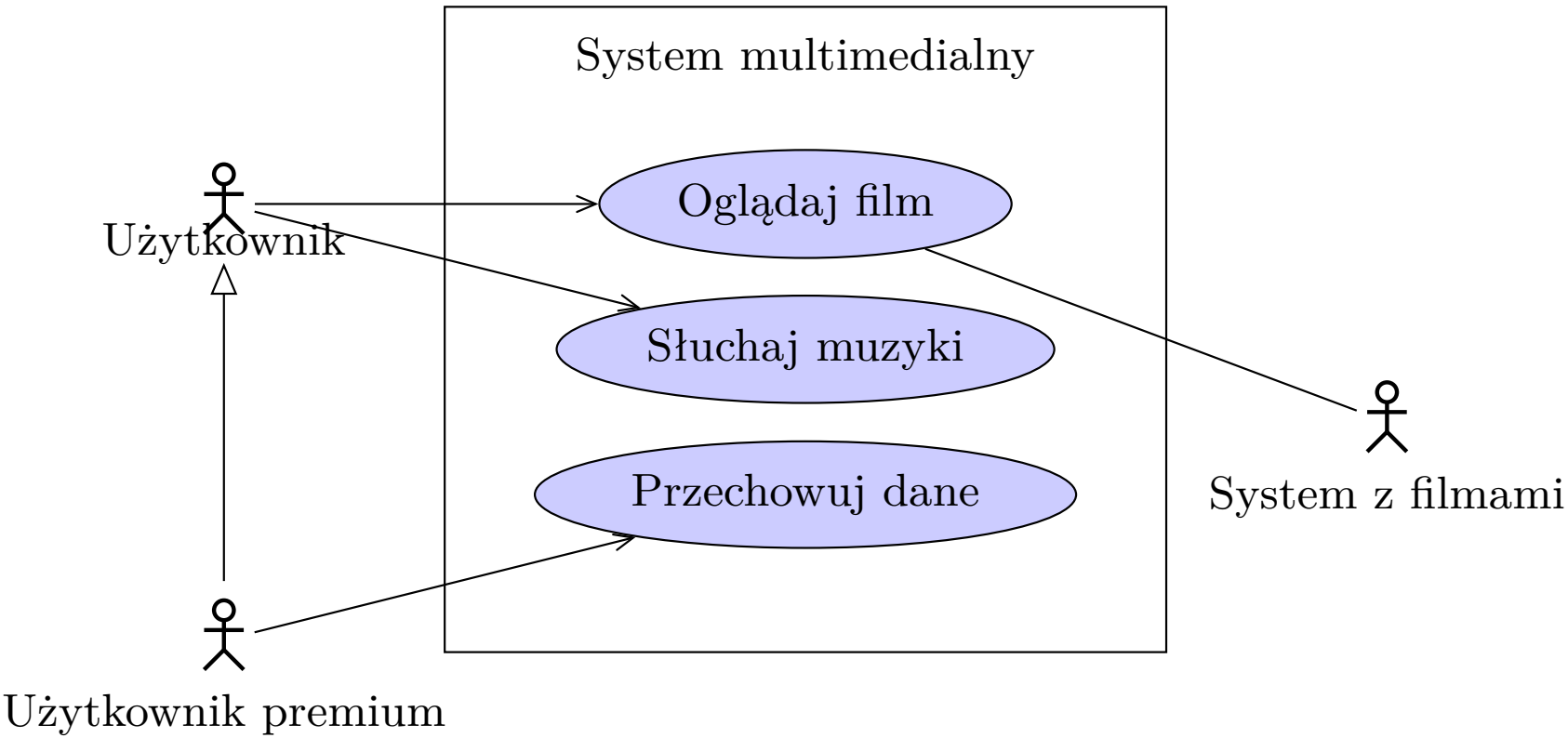
Scenariusze są innym narzędziem do określania i analizowania wymagań, które ma zastosowanie zarówno w tradycyjnych, jaki i zwinnym metodach wytwarzania oprogramowania. Scenariusz opisuje interakcję użytkownika z oprogramowaniem, przy czym użytkownik niekoniecznie musi być człowiekiem. Może on być zewnętrznym systemem lub organizacją. Każdy scenariusz musi spełniać następujące warunki:

1. musi określać stan systemu przed jego rozpoczęciem,
2. musi opisywać typowy przebieg czynności,
3. musi określać wyjątki i sposób ich obsługi,
4. powinien zawierać informacje o innych czynnościach, które mogą być wykonane w tym samym czasie,
5. musi określać stan systemu po jego zakończeniu.

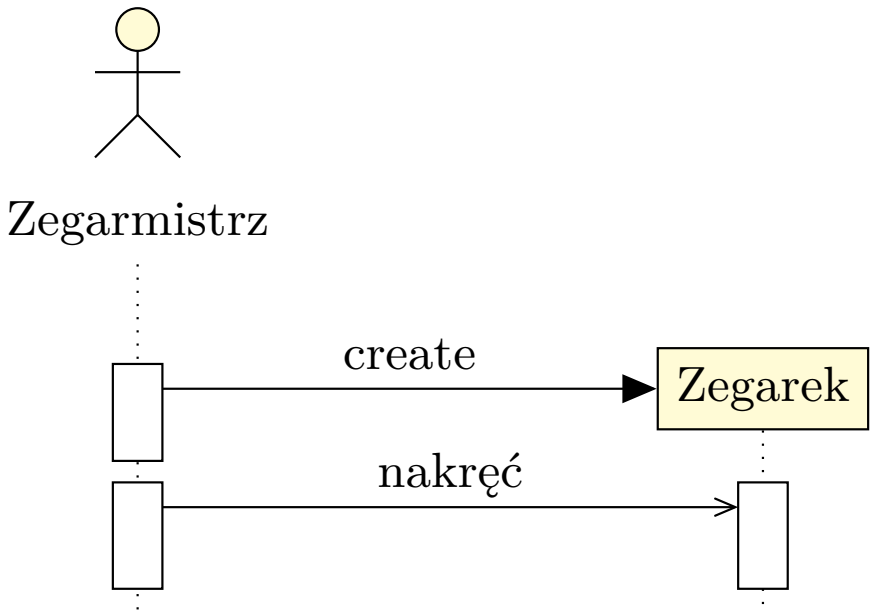
Przypadki użycia (ang. *use cases*)

W tradycyjnych metodach wytwarzania oprogramowania scenariusze mogą być opisane przy użyciu *diagramów przypadków użycia* UML (ang. *UML Use Case Diagrams*). Przykład takiego diagramu znajduje się na następnym slajdzie. Patykowate figury to *aktorzy*, którzy reprezentują *role* obejmujące ludzi-użytkowników, inne oprogramowanie, sprzęt lub inne systemy. Elipsy to *przypadki użycia* określające interakcje z oprogramowaniem, inaczej usługi, które to oprogramowanie dostarcza. Każdy przypadek użycia jest raczej zbiorem scenariuszy niż pojedynczym scenariuszem. Niektóre z nich opisują alternatywne sposoby wykonania tego samego przypadku użycia. Diagramy przypadków użycia nie są wystarczające, aby ująć wszystkie te scenariusze, dlatego powstały inne diagramy UML, które określają ich szczegóły. Należy do nich *diagram sekwencji*, którego przykład został zaprezentowany na slajdzie nr 17.

Use Cases


System multimedialny
Użytkownik
Oglądaj film
Słuchaj muzyki
Przechowuj dane
System z filmami
Użytkownik premium

Przypadki użycia

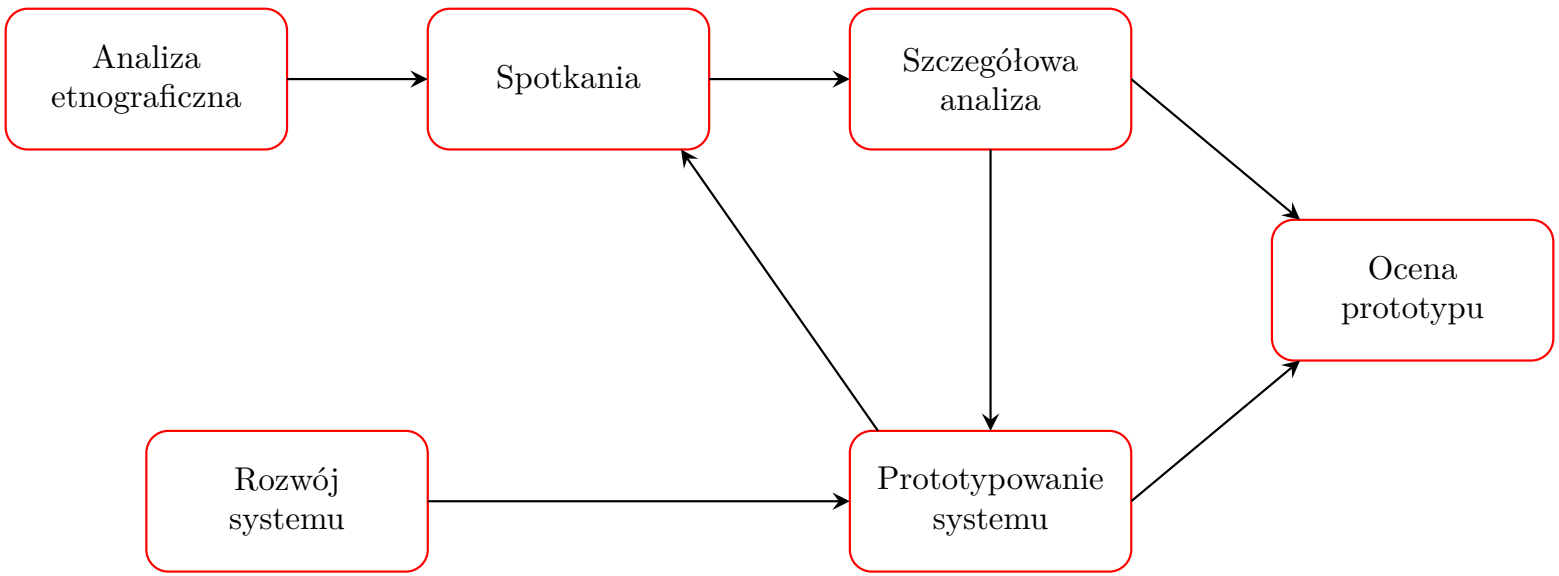


Etnografia i prototypowanie

Prototypowanie jest stosowane zarówno w tradycyjnych, jaki i zwinnym metodach wytwarzania oprogramowania do analizy złożonych lub wieloznacznych wymagań. Polega na opracowaniu i dostarczeniu użytkownikom prototypowej wersji oprogramowania, która implementuje tylko te usługi, które są definiowane przez takie wymagania. To pozwala wytwórcom otrzymać informację zwrotną na temat tego jak dobrze te wymagania zostały rozpoznane.

Prototypowanie może zostać połączone z etnografią. W kontekście inżynierii oprogramowania *etnografia* oznacza obserwację zwyczajów pracy osób, które będą użytkownikami opracowywanego oprogramowania. Analityk przeprowadzający etnografię może odkryć zwyczaje zawodowe przyszłych użytkowników oprogramowania, które nie są ujęte w oficjalnych dokumentach, ale powinny mieć wpływ na to, jak będzie działać to oprogramowanie. Wprowadzenie prototypu do środowiska pracy tych ludzi może mieć konsekwencje, które ponownie mogą być odkryte przez analityka. Po kilku takich rundach wytworzy się równowaga, która pozwoli wytwórcom zakończyć analizę i badanie wymagań. Proszę zwrócić uwagę, że etnografia nie może być użyta do określania i analizy formalnie zdefiniowanych wymagań.

Etnografia i prototypowanie



Zatwierdzanie wymagań

Według opracowania pt. *Chaos Report*, problemy z wymaganiami są najczęstszą przyczyną niepowodzeń projektów informatycznych. Defekty w wymaganiach są bardziej kosztowne w naprawie, niż defekty w kodzie. Ważnym jest więc, aby wytwórcy oprogramowania mieli pewność, że wymagania:

- są klarownie zdefiniowane,
- istotne,
- kompletne,
- realizowalne,
- weryfikowalne.

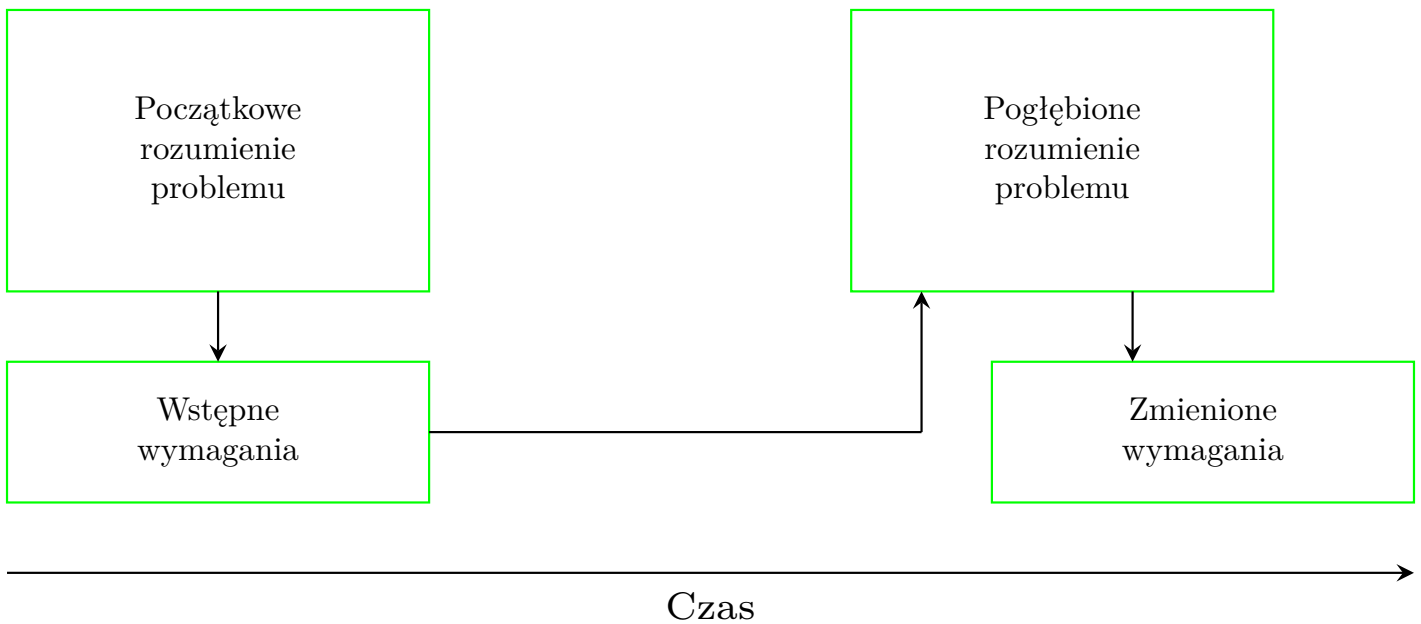
Te własności są sprawdzane w procesie *zatwierdzania wymagań* (ang. *requirements validation*). Istnieje kilka metod jego przeprowadzania:

- Przeglądy wymagań (ang. *requirements reviews*),
- prototypowanie,
- testowanie akceptacyjne (ang. *acceptance testing*),
- automatyzacja zatwierdzania (jeśli są używane metody formalne).

Zarządzanie wymaganiami

W miarę postępów prac w projekcie informatycznym, wymagania mogą i najczęściej się zmieniają, choć nie wszystkie w takim samym tempie. Zmiana tych, które bezpośrednio wynikają z dziedziny zastosowania jest mniej prawdopodobna, niż tych, na które mają wpływ czynniki zewnętrzne, takie jak regulacje prawne. Najważniejszą przyczyną zmian jest wiedza o produkcie, której nabywają inżynierowie oprogramowania wraz z upływem czasu (następny slajd).

Zarządzanie wymaganiami



Zarządzanie wymaganiami

Aby zarządzać zmianami wymagań inżynierowie oprogramowania muszą mieć metodę *śledzenia* tych zmian i *zarządzania* nimi. Śledzenie zmian w wymaganiach jest względnie proste w przypadku metod zwinnych, gdzie iteracje kończące się wydaniami są krótkie, a informacja zwrotna jest obfita. W metodach tradycyjnych może ono wymagać dodatkowego wysiłku. Jednak w obu przypadkach wymaga ono znajomości pochodzenia wymagań, zależności między nimi oraz tego, jaki wpływ na projekt będzie miała ich zmiana. Schemat procedury obsługi zmiany wymagania znajduje się na następnym slajdzie.

Zarządzanie wymaganiami

Bibliografia


Jeremy Dick, Elizabeth Hull i Ken Jackson. *Requirements Engineering*. Cham, Switzerland: Springer, 2017.


Gerard O'Regan. *Concise Guide to Software Engineering*. Cham, Switzerland: Springer, 2017.


User stories with examples and a template. 2023. URL: https://www.atlassian.com/agile/project-management/user-stories.


What is User Story? 2023. URL: https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!