

WP

PJATK – kurs GO



WP

Podstawy pracy z wymaganiami



Aleksandra Kankowska
27-28.03.2023r.



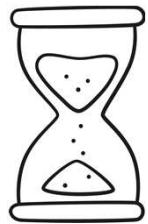
Agenda

1. Wymagania i potrzeby
2. Jak możemy zrozumieć drugą stronę - chemia rozmowy
3. Zbieranie i formułowanie wymagań
4. User Stories, czyli Historie Użytkownika
5. Jak to podzielić - techniki dzielenia wymagań na mniejsze części

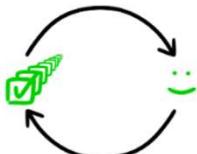


Dlaczego warto się tego nauczyć ?

Rozumiejąc wymagania i potrafiąc podzielić je na mniejsze:



- Dokładniej oszacujesz swoją pracę,
- Otrzymasz wcześniejszy feedback na temat tego co zrobiłeś/aś,
- Stworzysz rozwiązanie spełniające oczekiwania - coś co będzie wykorzystywane, a nie leżało w szufladzie lub trafi do kosza,
- Kod który powstanie będzie czystszy – łatwiejszy do czytania i do zmiany w przyszłości.





WYWIWAGANIA

Zacznijmy od początku – czym jest wymaganie?



1.

Warunek lub zdolność potrzebna interesariuszowi do rozwiązania problemu lub osiągnięcia celu.

2.

Warunek lub zdolność, które musi spełniać lub posiadać system lub komponent systemu, spełnienie warunków umowy, normy, specyfikacji lub innych formalnie narzuconych dokumentów.

3.

Udokumentowane przedstawienie stanu lub zdolności, jak w punkcie (1) lub (2).

Zacznijmy od początku – czym jest wymaganie?

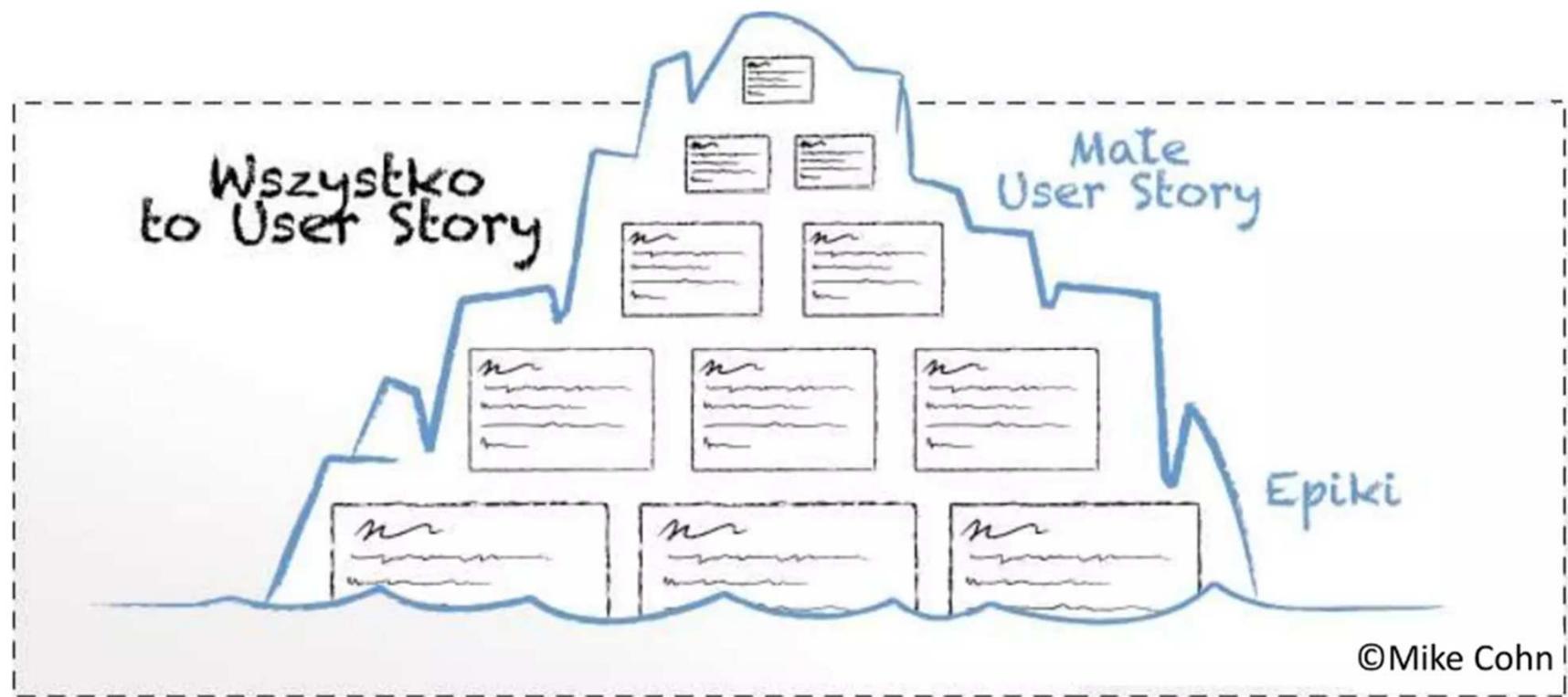


"**Wymagania** stanowią specyfikację tego, co powinno zostać zaimplementowane. Opisują, jak powinien zachowywać się system, albo określają jego właściwości lub atrybuty. Mogą nakładać ograniczenia na proces tworzenia systemu."

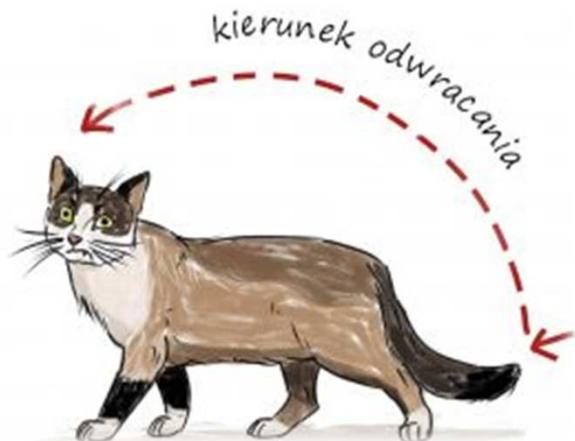
RODO
W
IT

Wymagania w Agile

Praca nad wymaganiami w Agile odbywa się zgodnie z zasadą „**Just In Time**” zaczerpniętą z Lean i następuje interakcyjnie.



Wymagania w Agile



- **Metodyki zwinne są otwarte na zmiany** - są odpowiedzią na obserwację, że wymagania klienta często zmieniają się w trakcie trwania projektu.
- **Nie można „odwracać kota ogonem”** - zmiany wymagań nie mogą być zupełnie nieograniczone i zachodzić w dowolnym momencie, możliwa jest natomiast negocjacja zakresu, kiedy zespół lub klient dowiedzą się czegoś nowego.

Co różni te wypowiedzi?



”

"Jestem odpowiedzialny za zwiększenie liczby obsługiwanych umów do sześciuset, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

”

"Jeśli liczba obsługiwanych umów pozostanie na poziomie dwustu miesięcznie, to zamkną nasz departament, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

Co różni te wypowiedzi?



“

"Jestem odpowiedzialny za zwiększenie liczby obsługiwanych umów do sześciuset, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

”

"Jeśli liczba obsługiwanych umów pozostanie na poziomie dwustu miesięcznie, to zamkną nasz departament, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

POWÓD - POTRZEBA BIZNESOWA



POTRZEBĄ

Czym jest potrzeba?



Potrzeba (np. klienta, interesariusza) jest to oczekiwana korzyść do osiągnięcia albo spodziewany problem, którego należy uniknąć.

Grupy potrzeb biznesowych



Dwie grupy potrzeb biznesowych:

- korzyści do osiągnięcia,
- problemy od uniknięcia.

Wróćmy do wcześniejszego przykładu



“

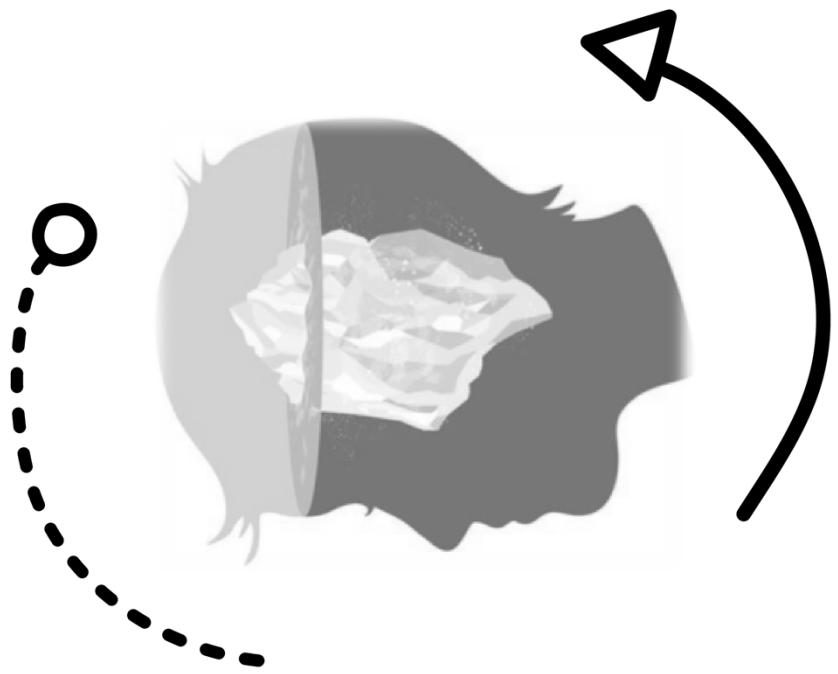
"Jestem odpowiedzialny za zwiększenie liczby obsługiwanych umów do sześciuset, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

”

"Jeśli liczba obsługiwanych umów pozostanie na poziomie dwustu miesięcznie, to zamkną nasz departament, WIĘC chcę zobaczyć miesięczny wykaz zawartych umów."

Korzyść do osiągnięcia

Problem do uniknięcia



**JAK ZROZUMIEĆ
DRUGĄ STRONĘ**

WP

Mózg wybucha, nie?





Klient nie wie czego chce?

- **Klient zawsze wie, czego chce** — wie, że chce rozwiązać jakiś problem, osiągnąć jakiś cel, coś usprawnić.
- **Klient nie zawsze wie, czego potrzebuje** — ponieważ jest skupiony przede wszystkim na swoich procesach biznesowych, wizji.
- **Klient często nie zdaje sobie sprawy z konsekwencji swoich oczekiwaniń** — gdyż jest ekspertem w obrębie swojej działalności biznesowej, a nie w obszarze technologii informatycznych.

Odkrywanie potrzeb [9]





Oczekiwania vs budżet Klienta



Źródło: <https://in.pinterest.com/pin/829014243888693605/>

<https://ahseeit.com/?qa=49875/client-expectations-vs-clients-budget-be-like-meme>

Gdy coś nie zagra z wymaganiami...



Gdy coś nie zagra z wymaganiami...

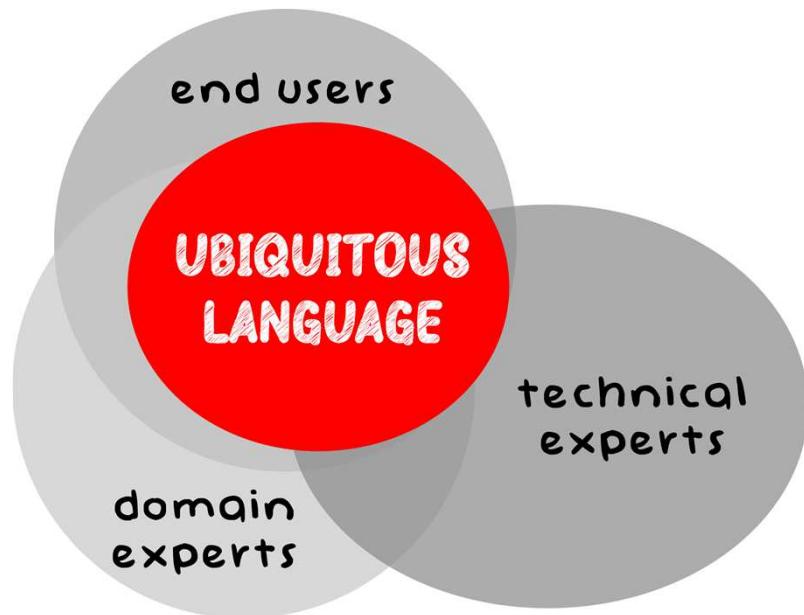


Gdy coś nie zagra z wymaganiami...





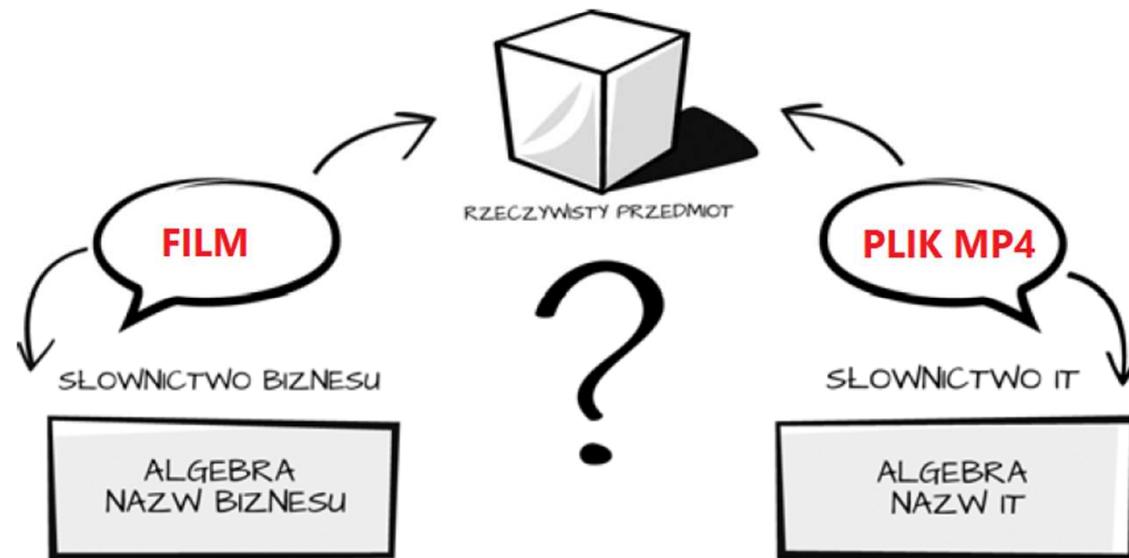
Ubiquitous language



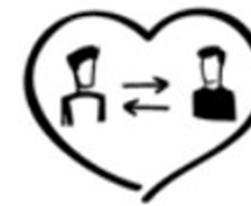
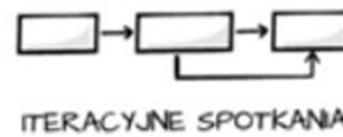
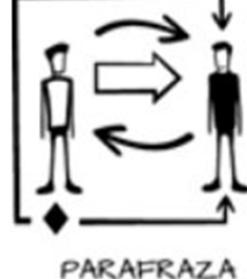
"**Ubiquitous Language** is the term Eric Evans uses in Domain Driven Design for the practice of building up a common, rigorous language between developers and users. This language should be based on the Domain Model used in the software - hence the need for it to be rigorous, since software doesn't cope well with ambiguity."

Za słowami kryje się znaczenie

Świat pojęć używanych przez Biznes jest zupełnie różny od tego, w którym funkcjonuje IT.



Techniki zbierania wymagań



MODEL NVC

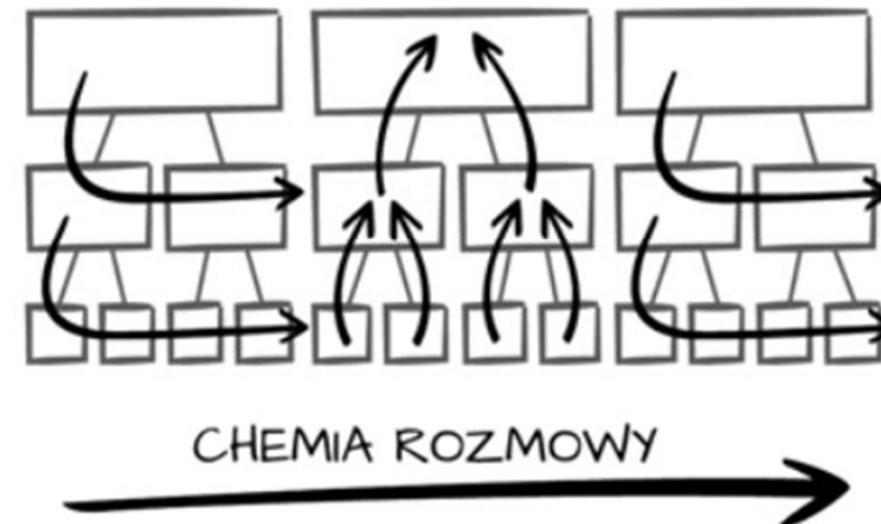


POZYTYWNA INTENCJA

JAK INACZEJ?

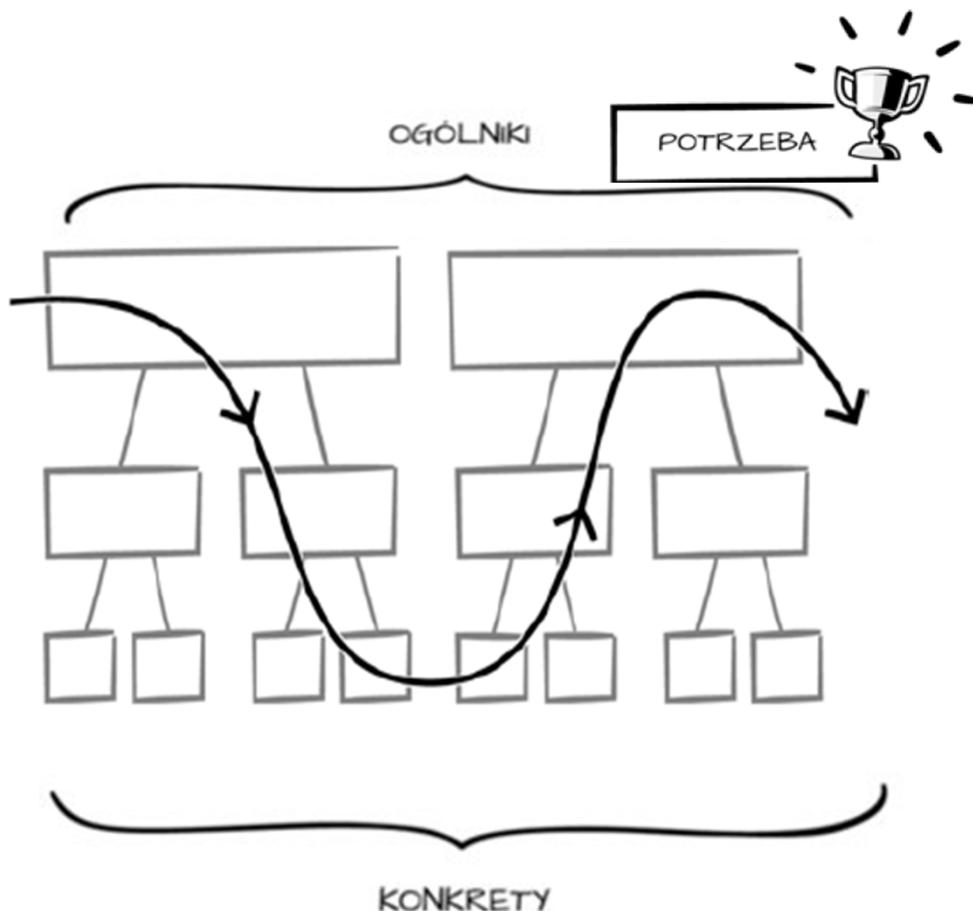


WIZJA



RAMY ODNIESIENIA

Konkretyzowanie i uogólnianie wymagań



→ **Konkretyzowanie** - rozbijanie informacji ogólnych na coraz większe konkrety.

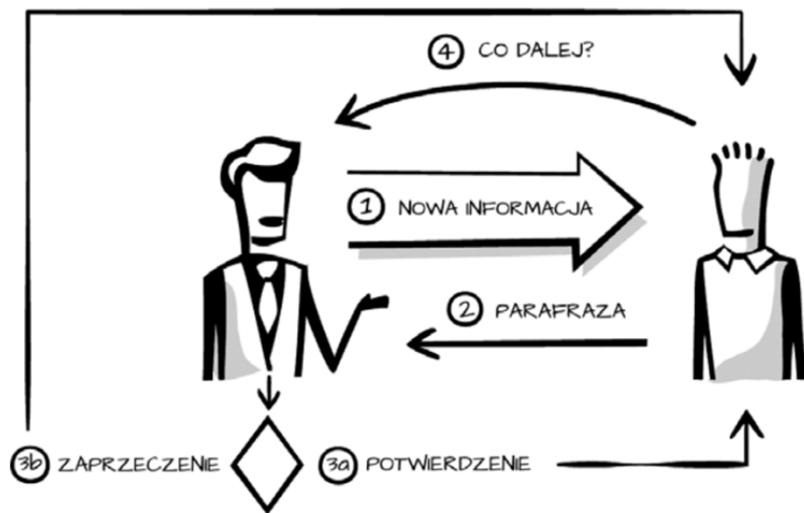
- Z czego się składa...?
- Jakie są kryteria...?
- Po czym poznasz, że...?

→ **Uogólnianie** - odnajdywanie powodów, dla których klient oczekuje określonych rozwiązań.

- Co się stanie, jeśli to zrealizujemy?
- Przed jakimi stratami może cię to uchronić?
- Dlaczego? Po co?

Parafrazowanie

Celem prafracowania jest budowanie zrozumienia potrzeb klienta.



Struktura komunikatu parafrazy:

→ Jeśli dobrze cię zrozumiałem,
to <WYPOWIEDŹ Klienta UJĘTA WŁASNYMI SŁOWAMI>. Czy pominąłem coś istotnego?



Parafrazowanie

“

"Przyszłoroczne koszty są tylko w 70% pokryte przychodami.

**Mamy do wyboru albo automatyzację i redukcję kosztów,
albo zamykanie całych oddziałów."**

A

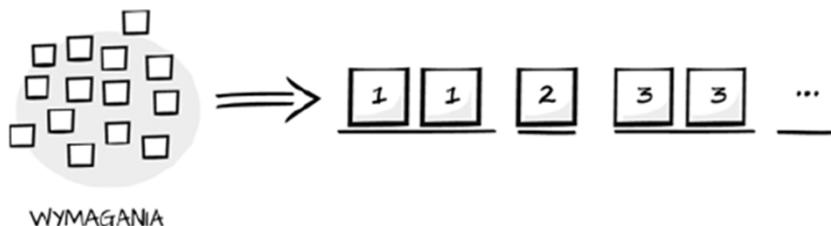
"Jeśli dobrze cię zrozumiałem, to przyszłoroczne koszty są tylko w 70% pokryte przychodami wobec tego mamy do wyboru albo automatyzację i redukcję kosztów, albo zamykanie całych oddziałów. Czy pominąłem coś istotnego?"

B

"Jeśli dobrze cię zrozumiałem, to oczekujesz, że zautomatyzowanie niektórych procesów w organizacji zapewni wzrost przychodów/redukcję kosztów o 30 %. Czy pominąłem coś istotnego?"

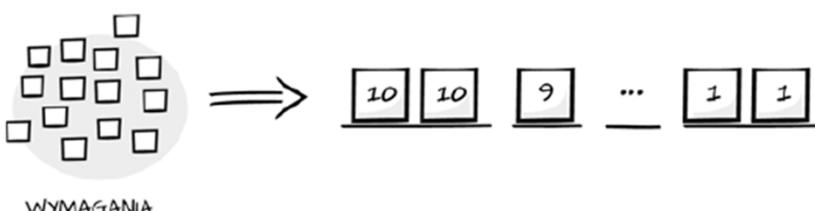
Określenie priorytetów wymagań

Wymaga poprowadzenia konwersacji, tak aby Klient spojrzał na system z wielu różnych perspektyw, dzięki czemu uzyska dostatecznie wiele informacji i będzie mógł podjąć dobrą decyzję.



Eliminowanie „poprzez korzyść”

→ Jeśli miałbyś zacząć używać tego systemu już jutro, to z których funkcjonalności chciałbyś skorzystać najpierw?



Eliminowanie „od problemu”

→ Gdyby zabrakło Ci teraz czasu/budżetu, to z której funkcjonalności zrezygnowałbyś na tę chwilę?



Wyobraźmy sobie parasol...

- Ma chronić przed deszczem
- Można go trzymać jedną ręką
- Rozkłada się
- Mieści się w torbie





Narzędzia do spisywania wymagań

Istnieją dwa popularne narzędzia do spisywania wymagań:

User Stories

(z ang. historie użytkownika)

wraz ze scenariuszami

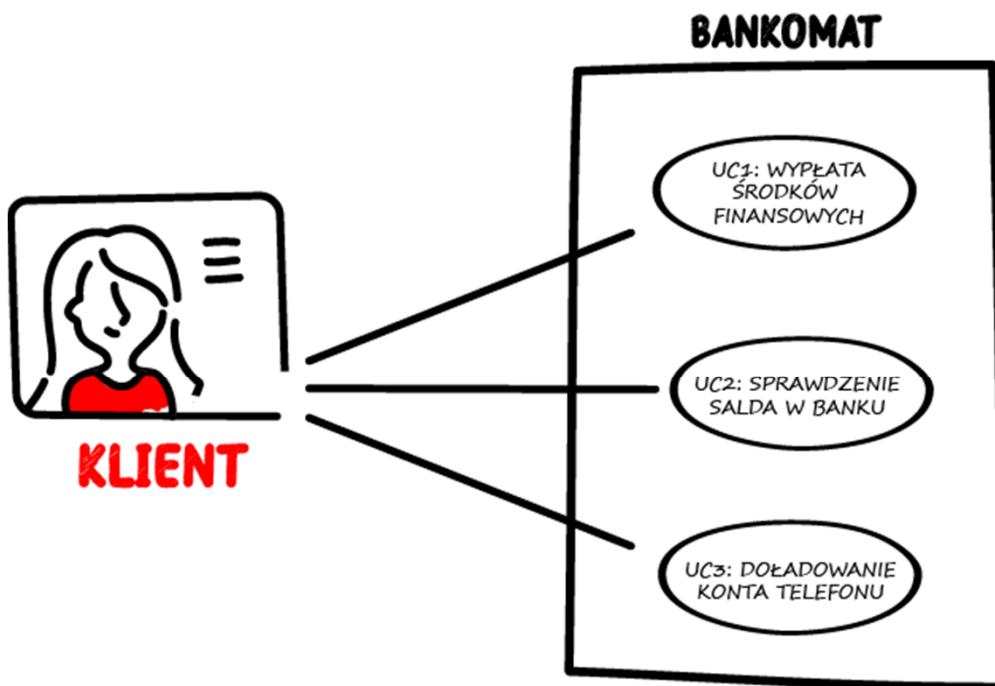
albo kryteriami akceptacyjnymi

Use Cases

(z ang. przypadki użycia).

Use case

Przypadek użycia reprezentuje ciąg czynności wykonywanych przez aktora i system w dążeniu do określonego celu.

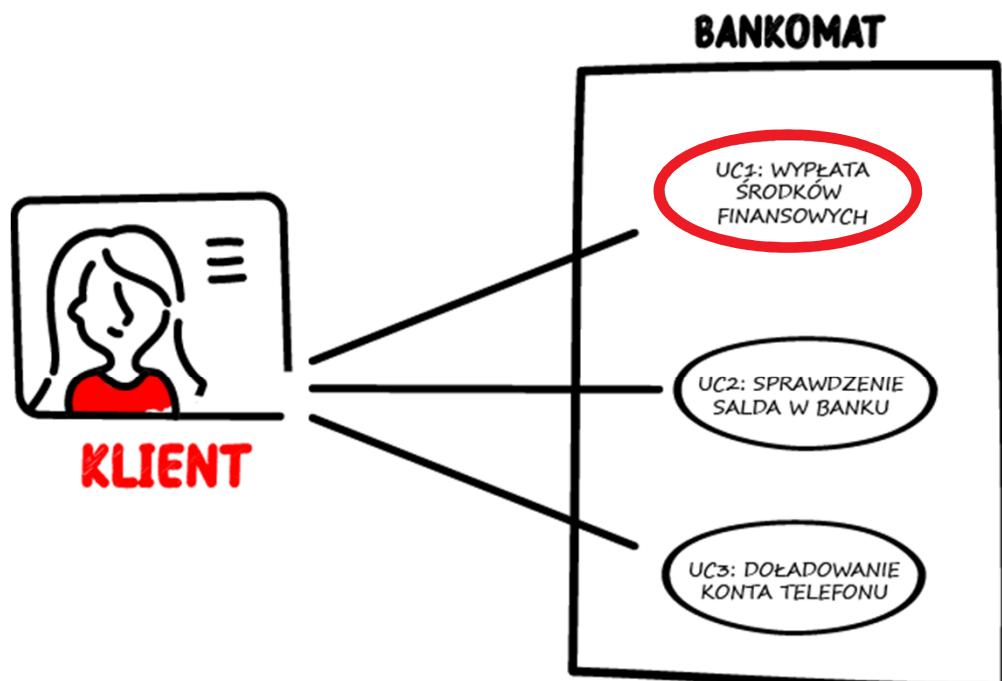


Przypadek użycia powinien:

- opisywać w jaki sposób system powinien być używany przez aktora w celu osiągnięcia konkretnego celu.
- być pozbawiony szczegółów dotyczących implementacji oraz interfejsu użytkownika.
- opisywać system na właściwym poziomie szczegółowości.

Scenariusze przypadków użycia

Najważniejszym elementem w opisie każdego przypadku użycia jest specyfikacja przepływu zdarzeń między aktorem a systemem.



- Przypadek użycia rozpoczyna się, gdy klient wkłada kartę do bankomatu.
System czyta informacje na karcie i bada ich poprawność.
- System pyta o PIN. Klient wprowadza PIN. System sprawdza poprawność.
- System pyta o rodzaj operacji do wykonania. Klient wybiera: "Wypłacić pieniądze".
- System pyta o wielkość kwoty. Klient wprowadza kwotę.
- System komunikuje się z bankiem, żeby sprawdzić poprawność ID konta, PIN i dostępność kwoty.
- ...



User Stories

Historyjki użytkownika spisywane są najczęściej według jednego z następujących schematów:



- JAKO <ROLA> CHCĘ <FUNKCJONALNOŚĆ>,
PO TO, ABY <EFEKT BIZNESOWY>.

- ABY <EFEKT BIZNESOWY>, JAKO <ROLA>
CHCĘ <FUNKCJONALNOŚĆ>.



User Stories

Kolejna równie dobra możliwość tworzenia US polega na zastosowaniu struktury Pięciu W, czyli formy kto, co, kiedy, gdzie, dlaczego.



→ JAKO <KTO> <KIEDY><GDZIE>CHCĘ <CO>,
PONIĘWAŻ <DLACZEGO>.

Używanie User Stories do odkrywania potrzeb



Rodzaje potrzeb	Wypowiedź klienta pasuje do schematu	Schemat User Stories
Problem do uniknięcia	<ul style="list-style-type: none">• Chcę uniknąć <WYPowiedź Klienta>.• Nie chcę, aby <WYPowiedź Klienta>.• To trudne, ponieważ <WYPowiedź Klienta>.• Jeśli tego nie zrobimy, to <WYPowiedź Klienta>.	ABY UNIKNAĆ <problem do uniknięcia>, JAKO <rola> CHCĘ <funkcjonalność>
Korzyść do osiągnięcia	<ul style="list-style-type: none">• Chcę osiągnąć <WYPowiedź Klienta>.• To sprawi, że <WYPowiedź Klienta>.• To będzie oznaczać, że <WYPowiedź Klienta>.	ABY OSIĄGNAĆ <korzyść do osiągnięcia>, JAKO <rola> CHCĘ <funkcjonalność>.

User Stories - czego brakuje w tym przypadku?



”

"Jako kierowca chcę widzieć zaplanowaną
trasę na kokpicie mojej ciężarówki."

Możemy się tylko domyślać...



Kierowca chce:

- poznać czas przejazdu między punktami trasy
- zobaczyć natężenie ruchu na trasie
- zobaczyć tylko listę punktów, do których dostarcza towar
- otrzymać informację o spodziewanym i aktualnym czasie przejazdu, żeby wiedzieć, czy wyrabia plan

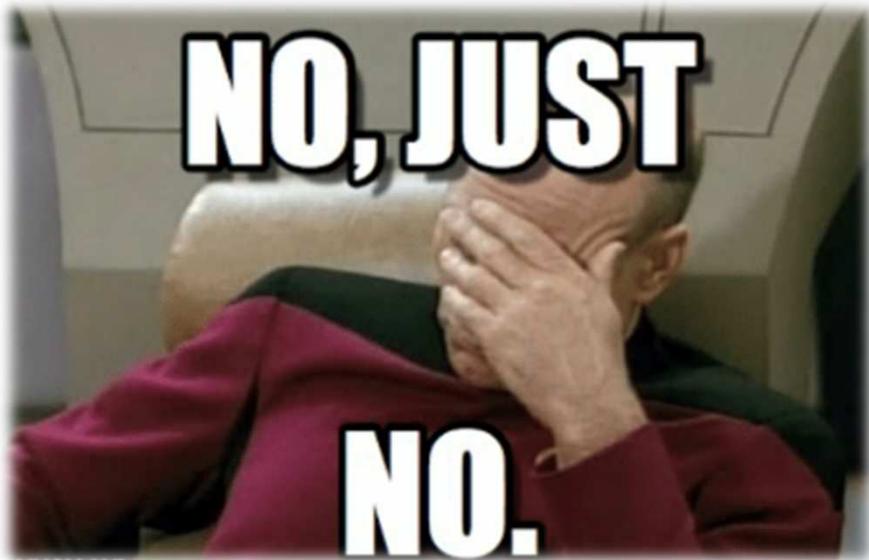
- A może chce...?

Oj nie tak...





Nie pchaj wszystkiego w User Stories

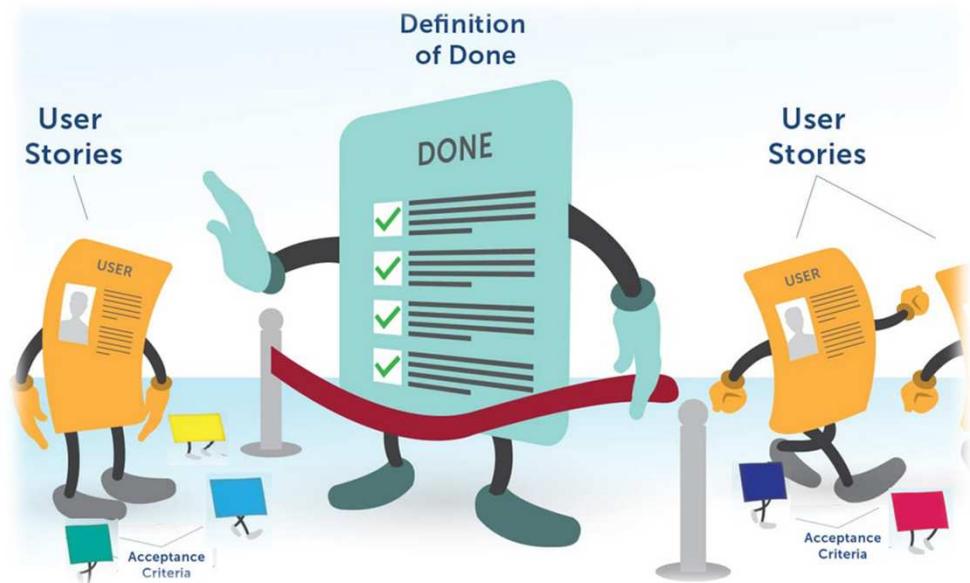


- JAKO Operator chcę się zalogować, PO TO,
ABY się zalogować.
- JAKO Doradca chcę złożyć nową ofertę,
PONIEWAŻ Product Owner tak chce.
- JAKO System chcę podbić wersję GO, PO TO,
ABY była najnowsza.

Kryteria akceptacji vs. Definition of Done



Kryteria akceptacji vs. Definition of Done

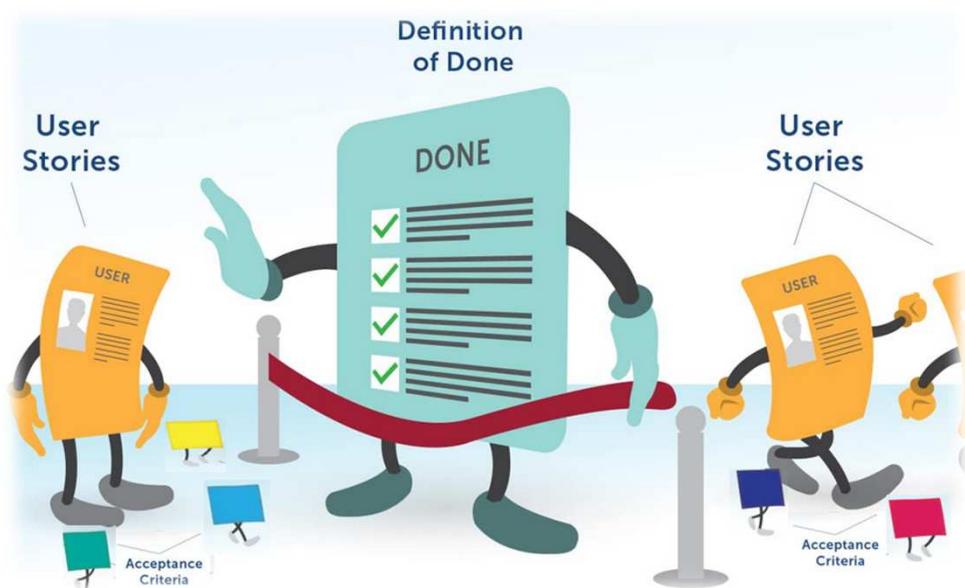


→ **Kryteria akceptacji** to z góry ustalone warunki, które produkt albo projekt (lub ich fragmenty) musi spełnić, by zostać zaakceptowanym przez użytkownika, klienta albo interesariusza.

Źródło:<https://agilepainrelief.com/blog/definition-of-done-user-stories-acceptance-criteria.html>

<https://porzadnyagile.pl/094-kryteria-akceptacji/>

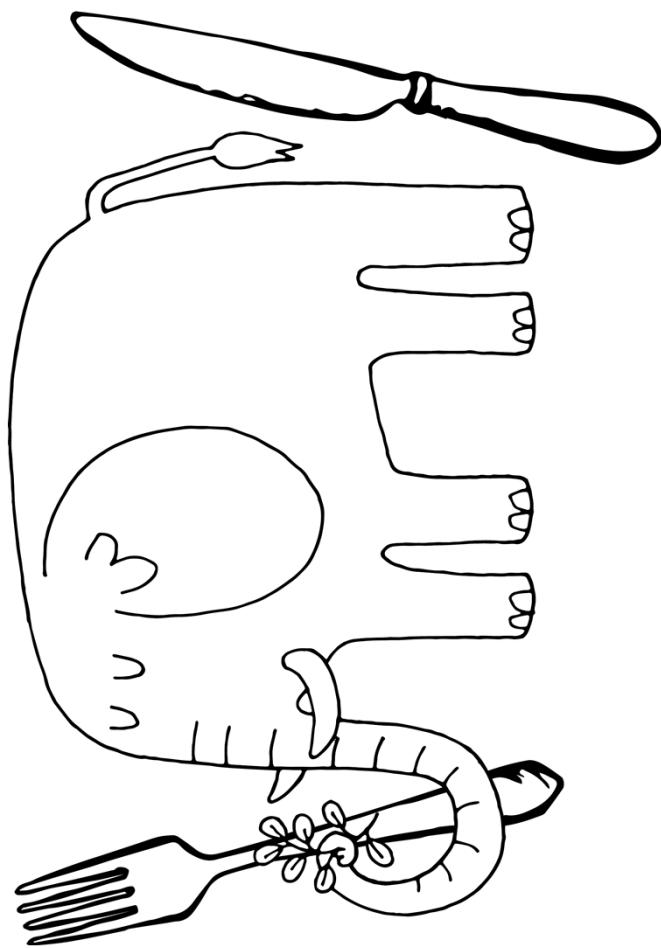
Kryteria akceptacji vs. Definition of Done



→ „**Gotowe**” nie zmienia się z jednej Historii użytkownika na drugą, podczas gdy Kryteria akceptacji są napisane specjalnie i niepowtarzalnie dla każdej indywidualnej funkcji lub Historii użytkownika.

Źródło:<https://agilepainrelief.com/blog/definition-of-done-user-stories-acceptance-criteria.html>

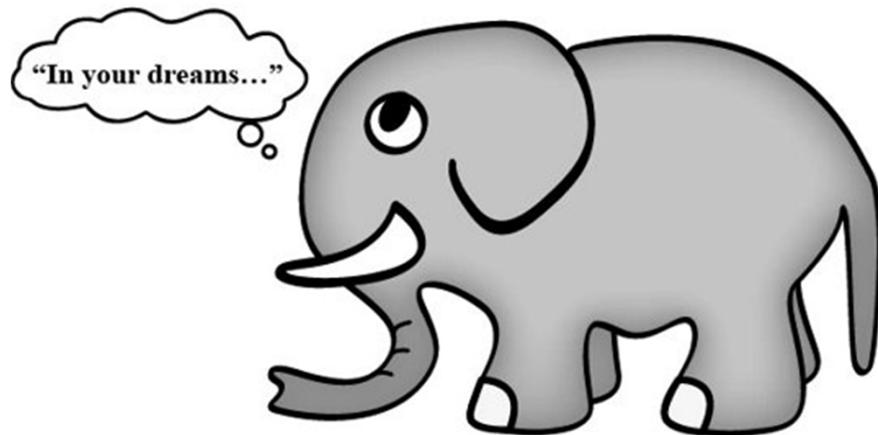
<https://porzadnyagile.pl/094-kryteria-akceptacji/>



JAK ZJĘŚĆ
STONIA?

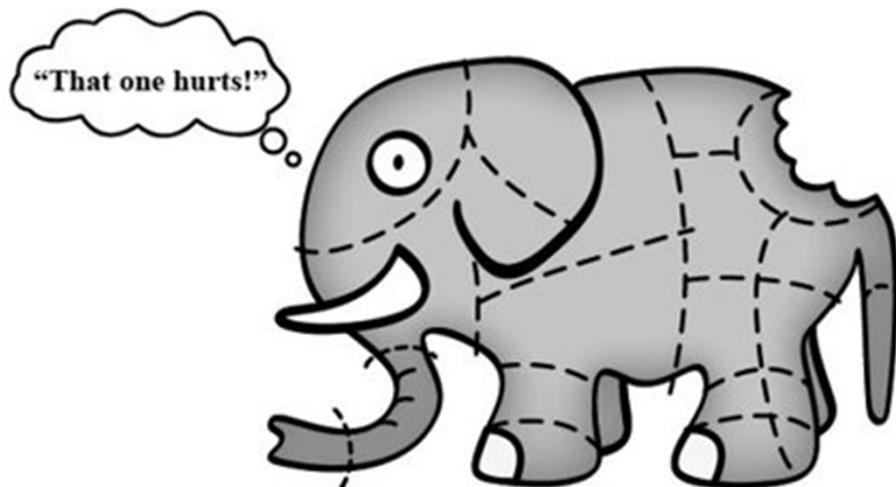
Jak zjeść słonia?

How to Eat an Elephant?



Słoń jest metaforą dla dużych projektów, zadań, tematów, których nie da się zrobić w ciągu kliku godzin, czy jednego dnia, a ich realizacja trwa w czasie.

Podobno, najlepiej po kawałku...



One bite at a time.

Słonia w jednym kawałku nie da się zjeść, natomiast jeśli się go pokroi na drobne kawałki, to stopniowo można to zrobić.

Dzielenie czego?



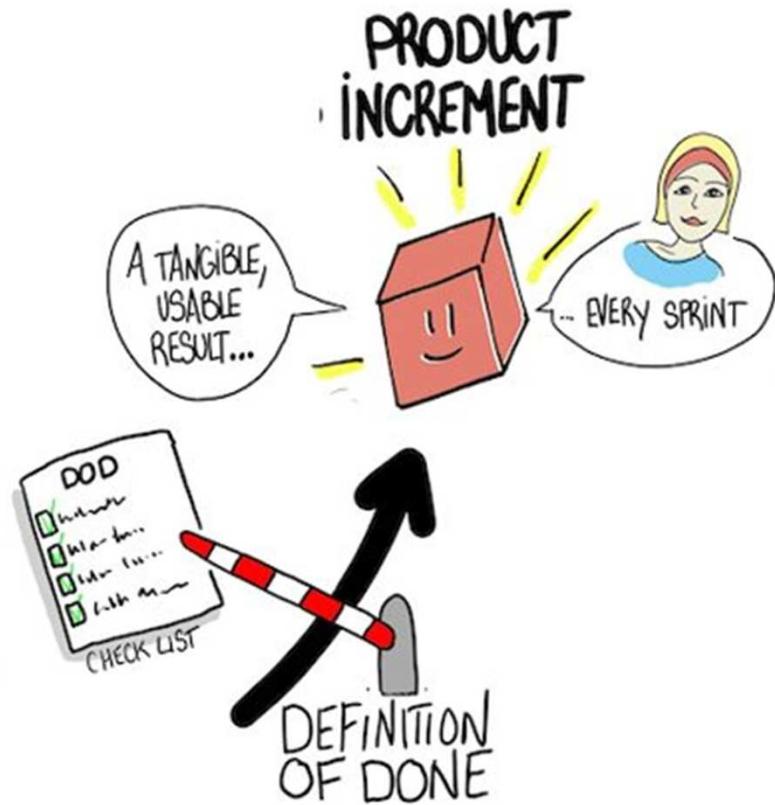
Product Breakdown

Dzielenie dużych kawałków produktu, na mniejsze elementy, które na wykonać zespół.

Work Breakdown

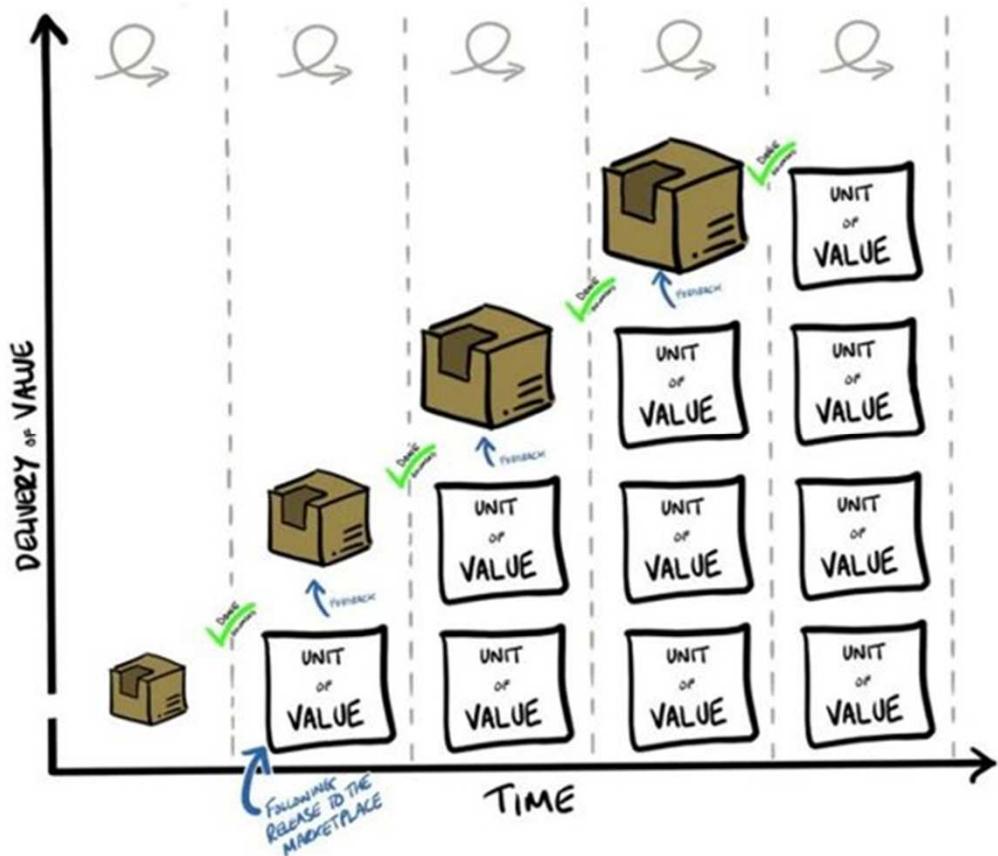
Dzielenie pracy w rozumieniu rozdzielania jej na mniejsze elementy.
Na przykładzie programowania osobno testów, osobno jakieś pracy na frontendzie, na backendzie, co jest kojarzone raczej z taką pracą w środku Sprintu.

Po co dzielić wymagania?



Efektem każdego Sprintu w Scrumie powinien być działający **przyrost produktu**.

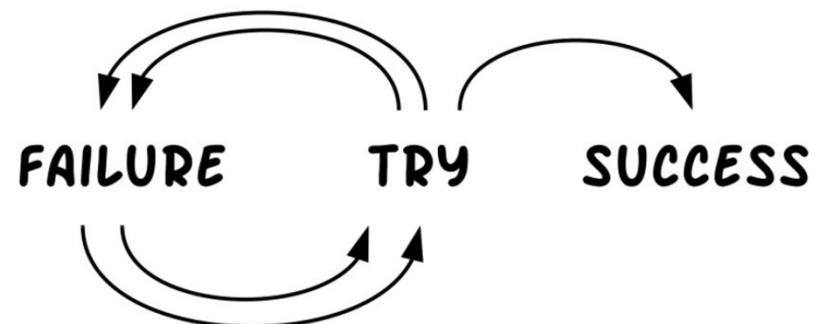
Po co dzielić wymagania?



"Remember, if you are doing something **only once**, you are **not iterating**. If you are **not adding** to what you already have, you are **not incrementing**."



Po co dzielić wymagania?



"Fail early, fail often, but always fail forward..."

Korzyści wynikające z dzielenia



- Mniejsze historie pozwalają zespołowi na **wczesną porażkę**,
- Mniejsze historie pozwalają zespołowi **na szybką porażkę**,
- Mniejsze historie umożliwiają zespołowi **szymbką naukę** (zarówno techniczną, jak i doświadczenia użytkownika)
- Mniejsze historie **robią miejsce na innowacje**
- Mniejsze historie **szyciej przechodzą przez przepływ pracy**
- Mniejsze historie mają **większą pewność i przejrzystość**
- Mniejsze historie **przyspieszają pracę** zespołu
- Mniejsze historie **zwiększą zwrot z inwestycji**

Po czym poznać, że można coś podzielić?



- Ma **spójniki** (i, albo, lub, oraz, a także) w tytule lub/i opisie
- Ma conajmniej **kilka kryteriów akceptacji**
- Ma bardzo **rozległy opis**
- **Estymacja** jaka przychodzi nam do głowy **jest wysoka**
- Jest chociaż **jedna osoba w zespole**, która sygnalizuje/uważa, że **da się podzielić**



"Niedasizm" dekpozycji wymagań

Najczęściej u źródłowego z problemów z dekompozycją wymagań znajdziemy brak zrozumienia jednej lub kilku kluczowych dla zwinności kwestii:

- **Brak wiedzy o produkcie lub zrozumienia wymagań** - bardzo ciężko przecież podzielić na kilka sensownych, wartościowych kawałków coś, czego nie ogarniamy jako całości.
- **Brak zrozumienia, czym jest potencjalnie używalny produkt** - nowa wersja, obejmująca wszystkie wcześniej istniejące cechy i funkcjonalności produktu, nad którą prace developerskie w pełni się ukończyły i której potencjalnie można zacząć używać.
- **Brak zrozumienia na czym polega inkrementalny rozwój produktu** - dużego produktu nie da się wytworzyć od razu w jego docelowym kształcie z co najmniej z dwóch powodów:
 - bo będzie to trwać niezmiernie długo
 - i nie jesteśmy w stanie z góry przewidzieć, jaki ten docelowy kształt powinien być.

Podział horyzontalny i inne kiepskie pomysły

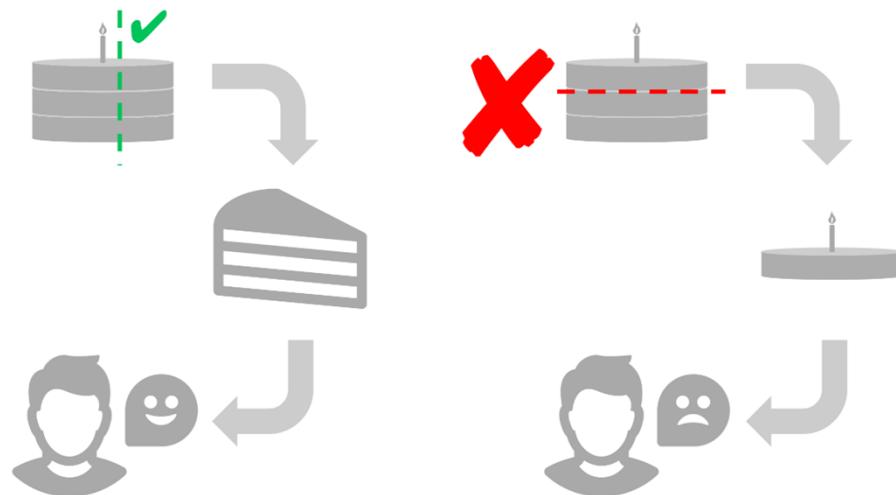


Dekomponując duże wymagania w ten sposób, że dopiero po zrealizowaniu ich wszystkich w określonej kolejności, uzyska się wartościowe rozwiązanie w istocie...

nie dokonujemy żadnej dekompozycji

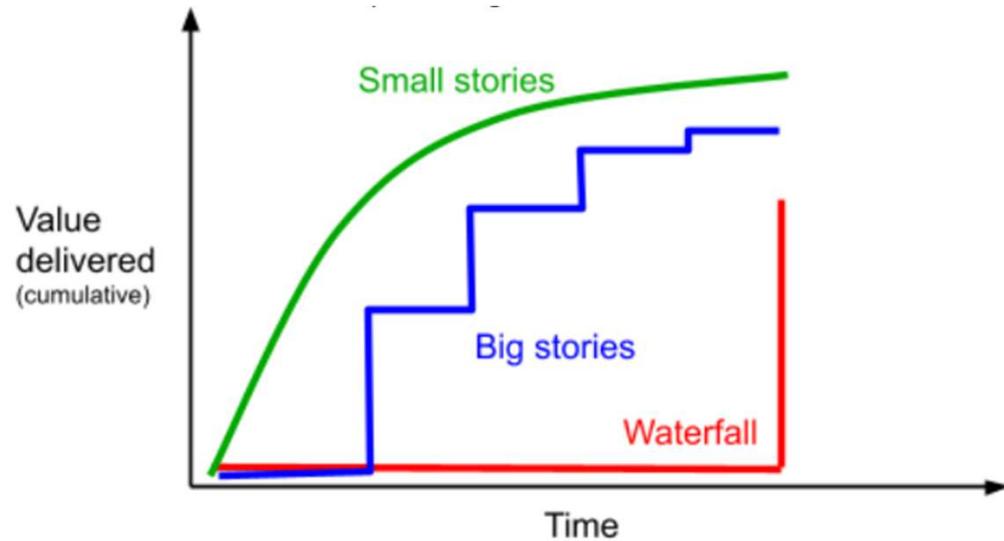
- jest to wyłącznie „**zabieg księgowy**” w backlogu produktu.

Podział horyzontalny i inne kiepskie pomysły



→ Kolejny mało wartościowy sposób podziału wymagań to poszatkowanie produktu na **warstwy technologiczne** (np. frontend/backend) i opisanie zmian w każdej z nich osobnym wymaganiem - pozornie dokonujemy tu zmiany w całym produkcie, a nie w jednym komponencie... ale wciąż, aby spełnić wymaganie biznesowe, **zrealizować trzeba prace we wszystkich takich warstwach.**

Przyrost wartości biznesowej

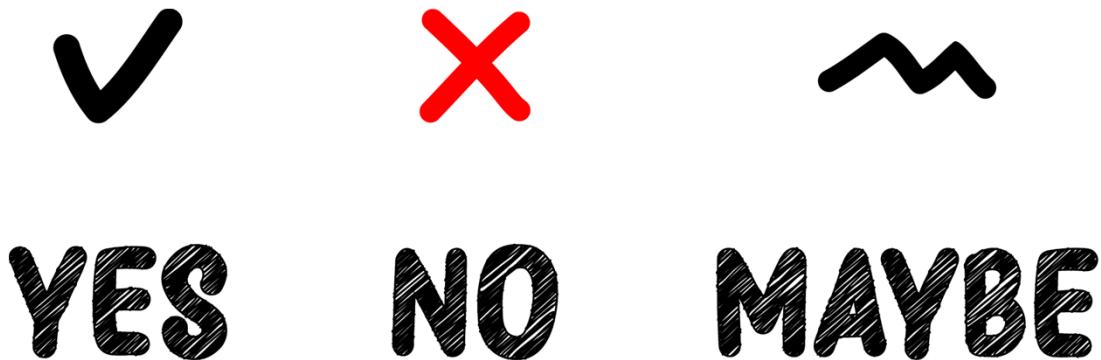


- Na przedstawionym rysunku możemy zauważyć, że wraz z upływem czasu **wartość biznesowa** jest zależna od przyjętej metody wdrażania i przygotowania wymagań.
- Im **mniejsze wymagania**, tym szybciej możemy wdrażać i tym szybciej możemy z klientami **weryfikować, czy podążamy w dobrym kierunku.**

Dzielenie wymagań w trakcie Sprintu



Jeżeli naszą intencją od początku była możliwość dopuszczenia do tego, że wykonamy tylko część danego zadania, czy powinniśmy w trakcie Sprintu podzielić dane wymaganie na część, którą zrobiliśmy i to, co zostało?





Dzielenie wymagań w trakcie Sprintu



- **Dokonajmy tego podziału najpóźniej na Planowaniu Sprintu.**
- Nasz plan jest sugestią dla innych zespołów, interesariuszy, Product Ownera odnośnie tego, **czego mogą się spodziewać**.
- Jeżeli później zaczniemy z niego „wykrajać” te kawałki, które nie działają i robić z nich jakieś „osobne User Stories”, to trochę tak jakbyśmy gościowi, który spodziewa się dwudniowego obiadu postawili przed nosem zupę i powiedzieli:
„zmieniliśmy pańskie zamówienie, po schabowegoproszę przyjść jutro”.

Elephant carpaccio



- **Elephant carpaccio** ćwiczy dzielenie wymagań poprzez przesadę. Dzięki tej metodzie sprawiamy, że historie są tak małe, że wszystko, co robimy dzisiaj, wydaje się w porównaniu z nimi wielkie.
- Metoda ta uczy co to znaczy projektować wymagania tak, aby skupić się na jak **najszybszym dostarczeniu wartości** dla użytkownika końcowego.



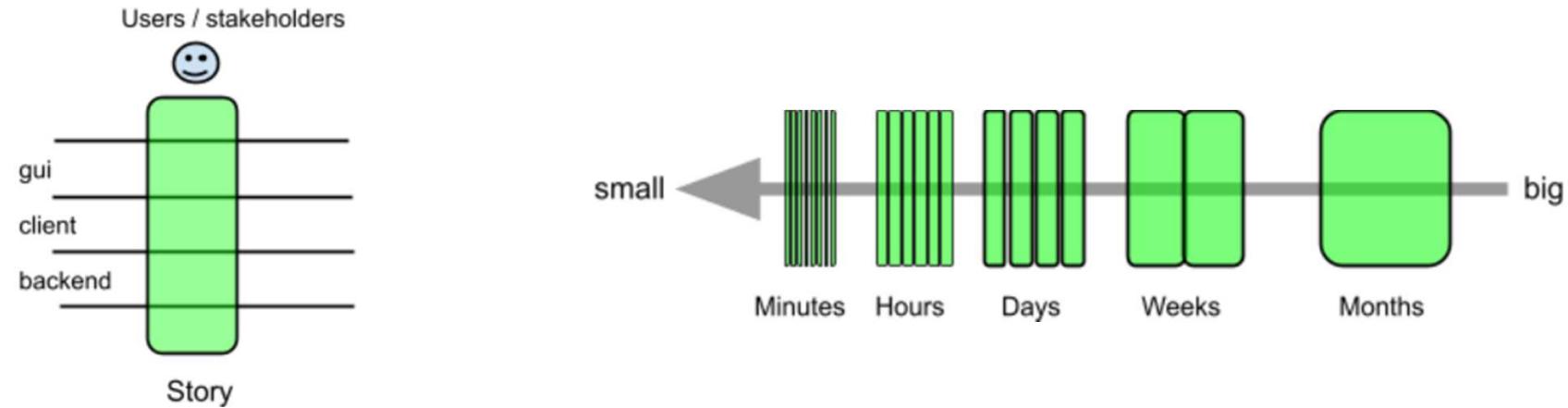
Elephant carpaccio



Kwintesencja tej metody to:

- Skupienie się na wizji produktu,
- Przemyślenie wymagań (zbudowanie backlogu) w taki sposób, aby implementacja była możliwa do wykonania określonym czasie (timebox).

„Wertykalny” podział



Story = vertical, testable, user-valuable

Story slicing = making thinner stories (but still vertical)

Zamiast kroić produkt na plasterki, dzielmy go pionowymi „paskami” przechodzącymi przez wszystkie warstwy technologiczne tak, by zrealizować jakąś – niekoniecznie kompletną biznesowo – funkcjonalność.

Hamburger method – transformacja sposobu myślenia



- **Metoda Hamburgera** została stworzona przez Gojko Adzic dzięki inspiracji zaczerpniętych z mapowania historii użytkowników Jeffa Pattona i pomysłów opisanych przez Craiga Larmana i Basa Vodde w książce *Practices for Scaling Lean & Agile Development*.
- Jest to metoda wizualna, która sprawia, że **myślmy o alternatywach**, pozostając jednocześnie w swojej **strefie komfortu**.
- Sprawdza się szczególnie w sytuacjach, gdy Zespół nalega na techniczne podziały, bo nie może znaleźć dobrego sposobu na rozbicie historyjki na mniejsze części z perspektywy dającej wartość biznesową, a jest ona zbyt duża by ją oszacować.

Hamburger method – transformacja sposobu myślenia



BAZA DANYCH



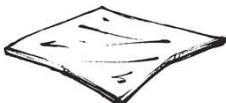
BACKEND



API



FROTEND



...



- **Krok 1:** Zidentyfikuj zadania
- **Krok 2:** Zidentyfikuj opcje dla każdego zadania
(alternatywne scenariusze w tym aspekt jakości)
- **Krok 3:** Połącz wyniki (posortuj opcje na podstawie wartości biznesowej, usuń duplikaty)
- **Krok 4:** Pokrój hamburgera
- **Krok 5:** Weź pierwszy kęs
(napisz pierwszą historię użytkownika)
- **Krok 6:** Weź kolejny kęs... i kontynuuj





Hamburger method – transformacja sposobu myślenia

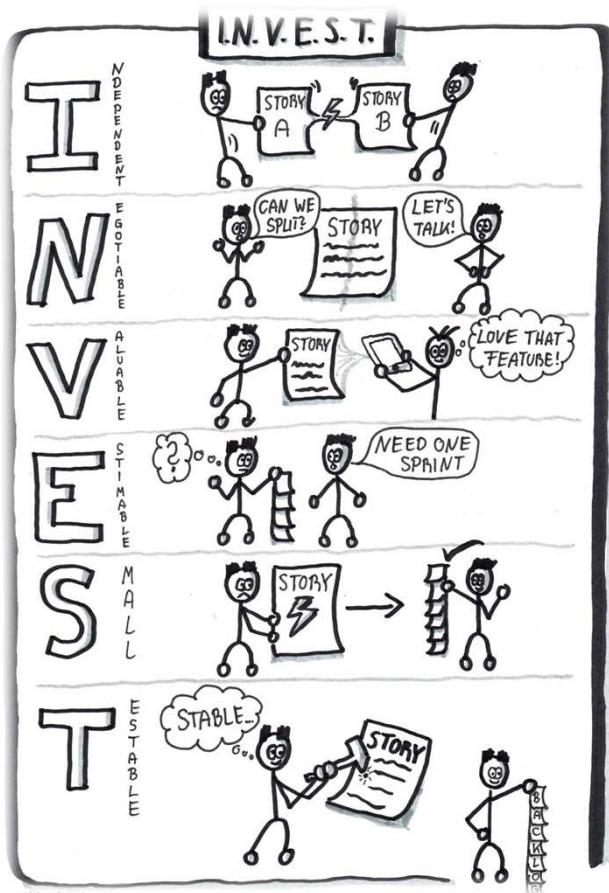
Kluczowe w tej metodzie jest **spojrzenie z kilku perspektyw** i próba odpowiedzi na szereg w gruncie rzeczy prostych pytań np.



- Czy zanim wydajność będzie satysfakcjonująca, funkcjonalność może po prostu zacząć działać?
- Czy musimy od razu umożliwiąć obsługę 100 tys. żądań na sekundę?
- Czy możemy póki co poczekać 5 sekund na wynik zapytania i upewnić się, że wynik (dane) jest dla nas satysfakcjonujący?



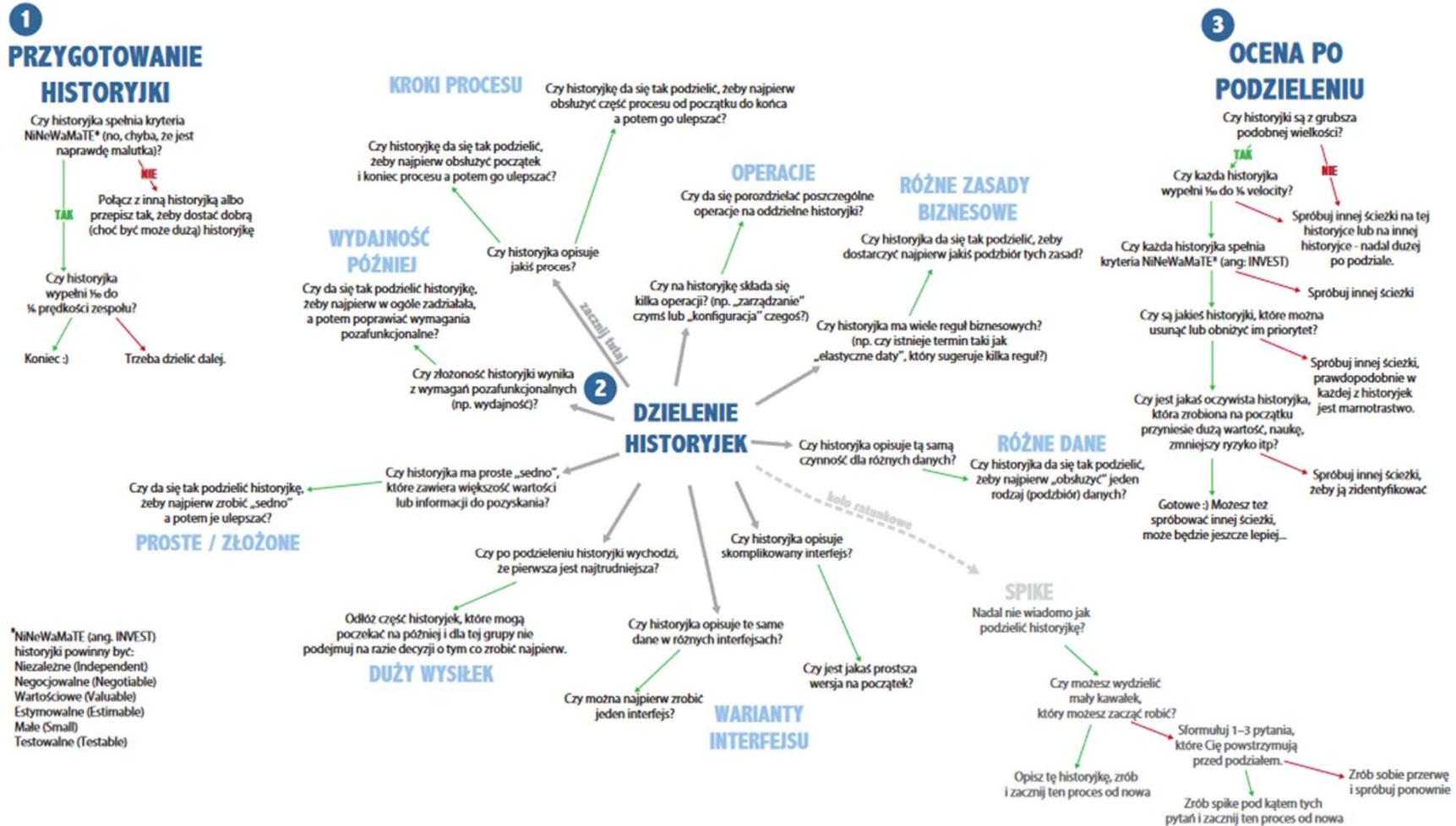
"Humanizujący przewodnik" dzielenia User Stories



Richard Lawrence oraz Peter Green opracowali **schemat podziału historii użytkownika**.

- Wejściem do procesu jest spełnienie przez historyjkę kryteriów **I.N.V.E.S.T.**
- Drugim krokiem jest **zastosowanie wzorców** poprzez odpowiedzenie sobie na szereg pytań dotyczących wymagania np.
 - Czy historyjka opisuje skomplikowany interfejs?
 - Czy jest jakaś prostsza wersja na początek?
 - Ostatnim elementem jest sprawdzenie, czy realizowane przez nas czynności doprowadziły do **odpowiedniej wielkości wymagania**.

"Humanizujący przewodnik" dzielenia User Stories





Ocena podziału

Wybierz podział, który pozwala obniżyć priorytet lub odrzucić historię.

Zasada **80/20** mówi, że większość wartości historii użytkownika pochodzi z niewielkiej części funkcjonalności.

Wybierz podział, który zapewni Ci więcej równych rozmiarów małych historii.

Podział, który zamienia 8-punktową historię na cztery 2-punktowe, jest bardziej użyteczny niż ten, który daje 5 i 3. Daje Właścicielowi Produktu większą swobodę w ustalaniu priorytetów poszczególnych części funkcjonalności



Weź ze sobą

1. Klient wie czego chce, lecz nie zawsze wie czego potrzebuje
2. Problemy do uniknięcia i korzyści do osiągnięcia są motywatorami działania
3. Kieruj się wartością biznesową
4. Dzielenie na mniejsze kawałki zawsze jest możliwe

Bibliografia



- [1] K.Zmitrowicz, *Analityk systemów. Przygotowanie do egzaminu z inżynierii wymagań*, Wydawnictwo Naukowe PWN, 2016
- [2] S.Sobótka, *Domain Driven Design krok po kroku. Część II: Zaawansowane modelowanie DDD – techniki strategiczne: konteksty i architektura zdarzeniowa*, 2012, <https://bottega.com.pl/pdf/materiały/ddd/ddd2.pdf>
- [3] M.Perendyk, *Definicja wymagania i rodzaje wymagań*, 2021, <https://analizawymagan.pl/definicja-wymagania-rodzaje-wymagan-czyli-o-czym-powinien-pamietac-kazdy-analityk/>
- [4] J.Sammy, *Needs and Solutions, Requirements and Designs*, 2013, <https://www.iiba.org/professional-development/knowledge-centre/articles/needs-and-solutions/>
- [5] K.Kaczor, *SCRUM i nie tylko. Teoria i praktyka w metodach Agile*, PWN, 2017
- [6] M.Robinson, *Naming Content Types Using a Ubiquitous Language*, 2020, <https://www.lullabot.com/articles/naming-content-types-using-ubiquitous-language>
- [7] T.Tomaszewski, *Jak napisać dobrą historyjkę użytkownika (user stories)?*, 2017, <https://productvision.pl/2017/napisac-dobra-historyjke-uzytkownika-user-stories/>
- [8] J. Gadek et al., *Product Guide. Podręcznik Product Managera*, 2016
- [9] M. Bartyzel, *Oprogramowanie szyte na miarę. Jak rozmawiać klientem, który nie wie, czego chce?* Helion , 2015
- [10] H.Kniberg, A.Cockburn, *Elephant Carpaccio exercise. Facilitation guide*, <https://docs.google.com/document/d/1TCuuu-8Mm14oxsOnlk8DqfZAA1cvtYu9WGv67YjsSk/pub>, 2013
- [11] M.Cohn, *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional, 2004
- [12] R.Markowicz, *O dekompozycji wymagań*, 2018, https://codesprinters.pl/blog/o-dekompozycji-wymagan/?lng=pl_PL
- [13] J. Wieczorek, *Labirynty Scruma. Sprawdzone sposoby na najczęstsze pułapki*, Jacek Wieczorek Consulting, 2017