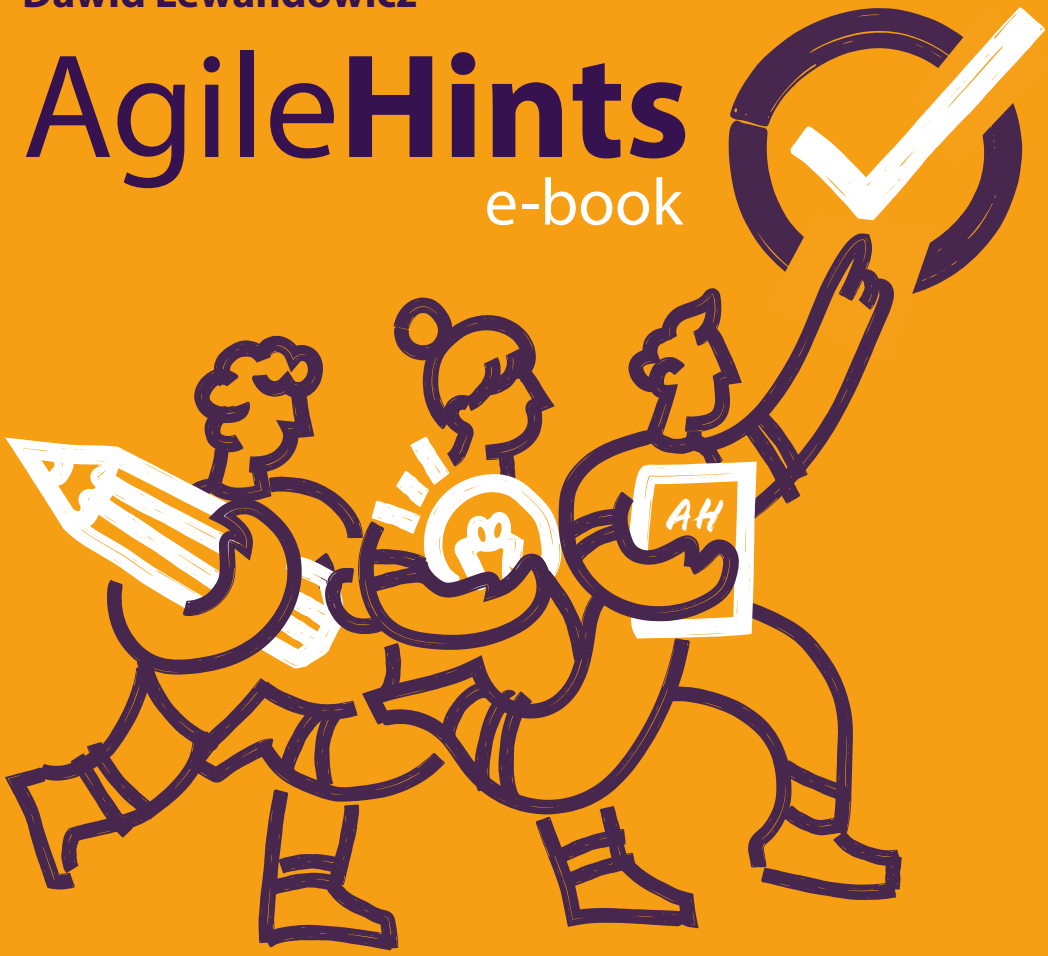
1
CZĘŚĆ

Dawid Lewandowicz

AgileHints
e-book



Twarde aspekty pracy zwinnej

100 sprawdzonych w praktyce hintów, jak podnieść efektywność i zadowolenie zespołów deweloperskich oraz osób bezpośrednio z nimi współpracujących.

Współautorzy:

Joanna Gosk
Monika Braun
Krystyna Abraham-Walasiak
Krystian Mrozek
Dariusz Knopiński
Konrad Synoradzki
Marcin Kliks

Edycja pierwsza

Spis treści

Słowo wstępu

Obserwując rynek wydawniczy i szkoleń w Polsce, w większości przypadków nie znajduję informacji o tym, z jakimi sytuacjami faktycznie mamy do czynienia w bieżącej pracy.

Nie brakuje natomiast pozycji albo interakcji w formie szkoleń, stanowiących o górnolotnych ideach. Przykład? Framework Scruma, którego trenerzy skupiają się na tym, jak framework wygląda, z jakich elementów się składa. Pomijane są wątki, które mogą występować w Waszej codziennej pracy.

To pchnęło mnie w stronę zapełnienia tej luki i stworzenia czegoś, co pokaże przykłady konkretnych sytuacji z życia oraz podpowie, jak można te sytuacje wesprzeć.

W niniejszej książce starałem się pokazać najistotniejsze, moim zdaniem, elementy pracy zwinnej. Nie jest to rozprawa naukowa, ale zbiór elementów dotyczących szeroko rozumianej pracy operacyjnej, które najbardziej wpływają na efektywność zespołu.

Książka jest zbiorem informacji dotyczących pracy zwinnej. Pozycją w której skupiam się (razem ze współautorami) na przekazaniu obserwacji i doświadczeń wynikających z kilkunastu lat pracy z produktami i zespołami deweloperskimi.

Elementy w niej zawarte są niskopoziomowe - dotyczą konkretnych zachowań lub rzeczy, które zespół powinien zrobić/zmienić, aby efektywnie dostarczać oprogramowanie i mieć przy tym wysoki poziom zadowolenia i zaangażowania.

Około 20% tekstu to przeredagowany i uzupełniony materiał, pochodzą-

cy z portalu agile247.pl. Pozostała część książki (ca 80%) to nowe treści.

Niniejsza książka została przygotowana z myślą o bardziej zaawansowanych czytelnikach lub tych na „fast-tracku". Nie będę wyjaśniał w niej podstawowych pojęć, jak na przykład framework Scruma.

Książkę razem ze mną napisało siedem osób, które są jej współautorami:

Asia ,

Krysia ,

Monika ,

Darek ,

Marcin ,

Konrad ,

Krystian .

To specjaliści w swojej dziedzinie, którzy dzielą się z Wami wiedzą i doświadczeniem.

Zapraszam!

2.1. Uszczegółowienie wymagań (ang. Refinement, Grooming)

Dawid Lewandowicz

Czas czytania: 11 min

Refinement jest jednym z najważniejszych wydarzeń we frameworku Scrumowym. Nie znajduje się on jednak bezpośrednio na cyklu. Bez dobrego Refinementu nie jesteśmy w stanie prawidłowo zarządzać naszym Rejestrem Produktu. Bez dobrego Rejestru Produktu, Planowanie, zamiast skupiać się na omówieniu kwestii, które są konieczne, aby dostarczyć przyrost jak najszybciej, zamienia się w dyskutowanie o tym „co autor miał na myśli" i do czego służy dane wymaganie. Brak dobrego Refinementu prowadzi do niepotrzebnych dyskusji, skupianiu się na rzeczach, które powinny być zrobione wcześniej (szybciej). Brakuje również możliwości wykorzystania w pełni danego wydarzenia, w tym wypadku - Planowania.

Jednak zacznijmy od początku. Czym jest i czemu w ogóle służy to wydarzenie? Planowanie to, najprościej mówiąc, spotkanie zespołu deweloperskiego wraz z Product Ownerem/Managerem - osobą odpowiedzialną za „co". Powinno ono dotyczyć uszczegółowienia zebranych wymagań. Product Owner (tak będę nazywał dalej tę osobę, choć nie zawsze ta rola w rzeczywistości tak się nazywa), zbiera wszystkie wymagania - te, pochodzące od interesariuszy, od klientów zewnętrznych lub wewnętrznych, czy też od samego zespołu. Na tym wydarzeniu zespół deweloperski po raz pierwszy widzi wszystkie zebrane wymagania. Celem, jakim nam przyświeca, jest zapoznanie się z nimi, przyswojenie ich, uszczegółowienie. Co oznacza uszczegółowienie? To nic innego jak omówienie każdego funkcjonalnego wymagania, dla którego powinno się:

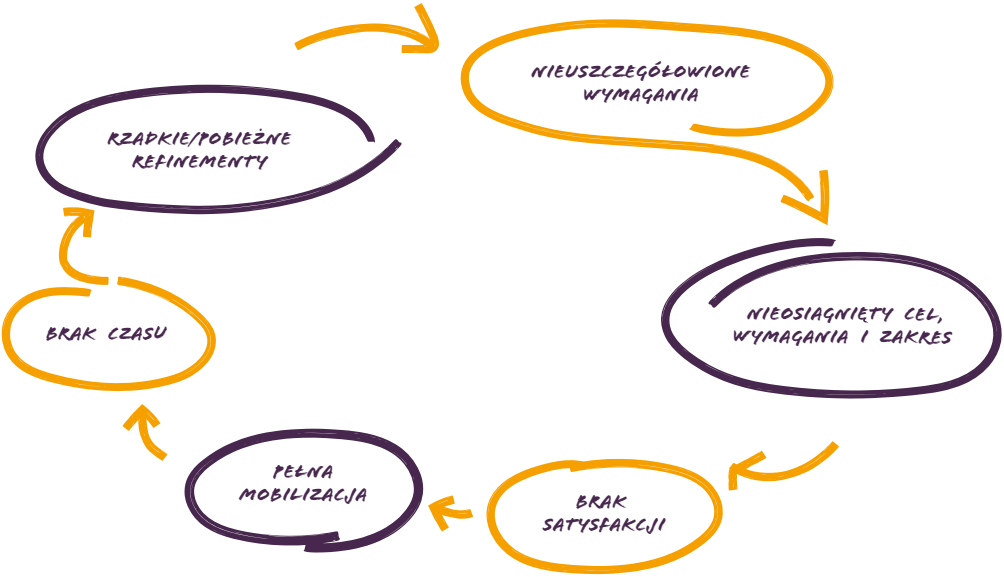
- **przemyśleć sprawy, które nadal trzeba wyjaśnić ze zleceniodawcą,**
- **zastanowić się, w jaki sposób dane wymaganie zostanie sprawdzone, czyli jakie są tak zwane kryteria akceptacyjne,**
- **zastanowić się, jak zespół deweloperski zamierza przetestować dane wymaganie.**

Kiedy te wszystkie rzeczy (aczkolwiek powyższa lista może nie być wyczerpująca) są już omówione, zespół deweloperski bardzo dobrze rozumie kwestie biznesowe albo, inaczej rzecz ujmując, rozumie już to, „co autor miał na myśli".

Ostatnią rzeczą powinno być dokonanie wyceny, która jest otwarciem panelu dyskusyjnego pomiędzy deweloperami. Ale o tym szerzej napisałem w rozdziale dotyczącym wycen i szacowania relatywnego ➳.

Szacuj złożoność zadań na Refinemencie.

Kiedy myślę o Refinemencie, przychodzi mi na myśl błędne koło, które możecie zobaczyć poniżej.



Zaczyna się ono od rzadkich i pobieżnych Refinementów, ponieważ zespół deweloperski, albo ktoś inny, wychodzi z założenia, że żadnego omówienia, uszczegółowienia nie potrzebujemy. To w oczywisty sposób prowadzi do sytuacji, kiedy w naszym Rejestrze Produktu znajdują się rzeczy, które są niegotowe. Kolejnym krokiem w tym kole są niezrealizowane: cel, wymagania, zakres. Innymi słowy, zespół deweloperski bierze na Sprint rzeczy, które wymagają jeszcze pracy „koncepcyjnej". W tej sytuacji, nie może się skupić tylko i wyłącznie na programowaniu, do czego w uproszczeniu jest powołany. Zajmuje się bowiem rzeczami, które powinny być omówione na dedykowanym, wcześniejszym spotkaniu. Nie jest w stanie zrealizować zakresu, do którego się zobowiązał na Planowaniu, ponieważ poświęca czas na coś innego. To z kolei prowadzi do braku satysfakcji, wzajemnych oskarżeń. Mamy do czynienia z sytuacją, w której zespół nie realizuje pracy w ramach danej iteracji, spadają jego morale, a na kolejnej Retrospektywie pojawiają się pytania, co można by

było zrobić lepiej. Retrospektywa kończy się mocnym postanowieniem poprawy i pełnej mobilizacji. Oznacza to, że zespół zrobi wszystko, aby jednak skończyć założony na poprzednim Planowaniu zakres (cel). Jeżeli ktoś mu w tym momencie powie, że powinien skupić się na uszczegółowieniu wymagań, odpowiada, że nie ma na to teraz czasu, ponieważ robi zadania, które miały być ukończone w poprzedniej iteracji. Dochodzimy w tym momencie z powrotem do początku naszego koła, czyli braku lub pobieżnych Refirementach spowodowanych brakiem czasu.

Ustanów Refinement jako stałe spotkanie w kalendarzu zespołu.

Zespół deweloperski nie uzmysławia sobie faktu, że znajduje się w tym błędnym kole. Nie zdaje sobie sprawy z tego, że będzie rozwiązał wciąż te same problemy, które tak naprawdę nie są problemami faktycznymi.

Wyjściem z tego błędnego koła może być sytuacja, w której lider zespołu, Scrum Master lub inna osoba, uświadomi zespołowi, że znajduje się w tym błędnym kole. Może to zrobić poprzez pokazanie nieprawidłowo sformułowanych wymagań, których realizacja zajmuje zespołowi zbyt wiele czasu.

Od czasu do czasu zaproś kogoś spoza zespołu, kto przez pewien okres (1-2 iteracje) będzie przyglądał się temu, co robi zespół. Ważne, by na końcu swojej pracy taka osoba podzieliła się na forum swoimi obserwacjami.

Aby to zrobić, wymagana jest analiza wziętych na Sprint zadań oraz przeprowadzenie ćwiczenia w trakcie Sprintu. Polega ono na odpowiedzeniu sobie na pytanie, ile razy dane wymaganie angażowało deweloperów w jego uszczegóławianie (nie w programowanie). Prosta wizualizacja dwóch, trzech iteracji może już prowadzić do stwierdzeń, że jako zespół znajdujemy się w tym „zaklętym" kręgu. Jeżeli tak się stanie, zespół zacznie myśleć, o prawdziwych problemach i nie będzie zastanawiać się na Retrospektywie, dlaczego nie realizuje danego Sprintu.

Oczywiście napisałem tutaj bardziej o pobieżnym uszczegóławianiu wymagań. Taka sama sytuacja, w sumie w jeszcze bardziej dotkliwy sposób, dotyczy braku danego wydarzenia w ogóle, w tym przypadku Refinementu.

Jeżeli zespół dojdzie do wniosku, że jednak wydarzenia typu Refinement powinny się odbywać w sposób regularny, musi zrobić wszystko, aby były one jak najbardziej efektywne. Innymi słowy, jeżeli nie znamy celu wydarzenia i agendy, najlepiej wcześniej wysłanej, nie możemy oczekiwać, że dane spotkanie przebiegnie w sposób prawidłowy. Ta sama kwestia dotyczy moderatora, którego rola jest tutaj kluczowa. Mamy bowiem dużą tendencję do odchodzenia w rozmowie od głównego tematu. Zadaniem moderatora (rola stała lub przechodnia w zespole, może ją pełnić także np. Scrum Master/Product Owner) powinno być dbanie o to, by rozmowa zawsze dotyczyła sedna sprawy. Ważnym jest, żeby skupić się na omówieniu danego wymagania, odrzucając niepotrzebna dygresje.

Dobry Refinement wymaga moderatora,
wyznaczonej w zespole osoby,
która odpowiednio poprowadzi spotkanie.

W przypadku Refinementu chcę Wam zwrócić uwagę na jeszcze jedną istotną rzecz. Proces dewelopmentu to faza Delivery. Wcześniejszą fazą jest faza Discovery. Jeżeli myślimy o całościowym doprowadzeniu produktu od momentu jego wymyślenia, po dostarczenie do klienta końcowego, musimy mieć na uwadze te dwie fazy. Discovery w tym przypadku dotyczy odpowiedzenia na potrzeby użytkownika. Może się ona realizować w bardzo różny sposób - poprzez zbieranie danych, pogłębione wywiady z użytkownikami, czasami zaproponowanie produktu i tworzenie dla niego rynku. Natomiast w przypadku fazy Delivery mówimy o dostarczaniu pewnej koncepcji na rynek. Koncepcji, która została zidentyfikowana i zbadana w poprzedniej fazie. Czasami jednak, moim zdaniem za rzadko, zespół, który jest odpowiedzialny za utrzymanie danego produktu, nie jest angażowany w fazę Discovery.

Zaangażuj zespół w fazę Discovery
- przykładem jest rozdział 8.1.
- Od idei do wdrożenia w 1 tydzień.

W konsekwencji do zespołu są przesyłane, przez inne jednostki biznesowe przedsiębiorstwa, rozwiązania gotowe do implementacji. Prawdą jest, że nie każdy zespół ma ochotę na uczestniczenie w ich wypracowywaniu, czyli w fazie Discovery. Jednak jako liderzy czy Product Ownerzy musimy mieć świadomość, że podniesienie motywacji zespołu następuje m.in. poprzez angażowanie go do takich prac. Nie poprzez traktowanie jak siły do implementacji, kiedy ma bardzo mało do powiedzenia, ale jako siły do opracowania danej koncepcji. Nie zapominajmy tutaj o jeszcze jednym elemencie - to zespół najlepiej zna produkt, którego wymagania/koncepcje dotyczą. I nie chodzi tu o np. dług techniczny, ale o znajomość biznesową. Trzeba tylko te kwestie w umiejętny sposób od zespołu pozyskać.

Wracając do fazy Discovery (to tak a propos moderatora i robienia dygresji). Jeżeli zespół chciałby jednak wejść w fazę początkową i być odpowiedzialnym za rozwój danego produktu, dyktować innym jednostkom biznesowym, jak ten produkt będzie wyglądał, wtedy Refinementy powinny dotyczyć właśnie fazy Discovery. Nie chodzi już tylko o uszczegółowienia danego wymagania, które zostało do nas, jako zespołu, skierowane, ale wymyślenie tego wymagania. Takie postępowanie bardzo często dotyczy zaangażowania specjalisty od User Experience, który powinien pomóc zespołowi wymagania poszukać .

Zaangażuj projektanta UX, który będzie przewodził zespołowi w poszukiwaniu odpowiedzi i rozwiązań na pytania i problemy użytkownika.

Innymi słowy, UX powinien pomóc tak przeprowadzić warsztaty (Refinement) i wyszukiwanie problemów użytkowników, aby pojawiły się odpowiednie wymagania. Dobrym przykładem takiego zachowania jest to, co napisałem wspólnie z Krystianem - Story mapping w praktyce . Mieliśmy na pokładzie UX'a, który pomógł nam przeprowadzić warsztaty prowadzące do zidentyfikowania elementów, nad którymi jako zespół deweloperski musimy się pochylić. Jeżeli te elementy zidentyfikujemy, w kolejnym kroku mamy już do czynienia ze zwykłym Refinementem, który dotyczy uszczegółowienia tych znalezionych/potwierdzonych wcześniej wymagań.

5.3. Pętle informacji zwrotnej (ang. feedback loop)

Dawid Lewandowicz

Czas czytania: 6 min

Jeśli spojrzymy na definicję (są ich trzy!), którą podaje Cambridge Dictionary, możemy być zaskoczeni, jak bardzo dotyczą naszej pracy . Feedback loop to „system, służący do udoskonalania produktów i/lub procesów, bazujący na zbieraniu i reagowaniu na pojawiające się komentarze użytkowników".

Kiedy natomiast zapytacie zespół, czym jest feedback loop, pojawią się (zapewne) różne odpowiedzi:

- **kluczowe elementy Agile,**
- **inspekcja i adaptacja,**
- **„pętle" są elementami, które pojawiają się na każdym poziomie nowego wdrożenia kodu/produktu,**
- **krótkie pętle zwrotne są kluczowe w kontekście wdrażania i stosowania Agile, ponieważ pozwalają zespołom szybko się uczyć.**

Problemem jednak w tym przypadku nie jest zdefiniowanie czym są „pętle informacji zwrotnej", ale z jakich elementów się składają.

David Lowe w artykule „Feedback loops" podaje listę rzeczy, które można uznać za elementy feedback loop. Lista ta, skategoryzowana i wyczerpująca, jest dla mnie rozwinięciem definicji:

- **pętle informacji zwrotnej na poziomie zespołu/organizacji**

  Wszystkie niżej wymienione elementy znajdziesz w rozdziałach tej książki:

  - Codzienny Scrum,
  - Retrospektywy,
  - Scrum of Scrums (skalowanie Scruma),
  - Podsumowania Sprintów z udziałem innych zespołów lub wspólne pomiędzy zespołami.

- **pętle informacji zwrotnej na poziomie dewelopmentu (tworzenia nowego kodu),**
  - Pair Programming (programowanie w parach),
  - Code Reviews ,
  - testy automatyczne - w tym testy jednostkowe (ang. unit tests), TDD , BDD ,
  - informacja zwrotna pochodząca z testów wykonywanych manualnie,
  - Continuous Integration (CI) ,
  - Continuous Delivery (CD) ,
  - wnioski z wdrożenia,

- **pętle informacji zwrotnej na poziomie wizualizacji/raportowania,**
  - Agile boards

  Są to wszelkie tablice (elektroniczne i manualne), które w założeniu mają przekazać informację dotyczącą stanu naszego Agile'a, w tym na przykład: happiness metric , velocity i capacity, dług techniczny, przerywacze Sprintu .

- statusy red-amber-green (RAG)

  Jest to prosty status pokazujący kondycję inicjatyw/projektów. Każdy przykładowy projekt jest oznaczany kolorem, odpowiadającym jego kondycji (stanu realizacji zadań w nim zawartych, zidentyfikowanych ryzyk). R - red, czyli bardzo źle, A - amber (pol. dosłownie bursztynowy, ale bardziej stosuje się pojęcie żółty), czyli trudne w określeniu. Dlaczego? Ponieważ ten stan może być użyty dla projektów, których realizacja jest w przedziale 10% do 90%. Wymaga on dodatkowego wyjaśnienia. G - green, czyli jest w porządku.

- OKR

Stan realizacji celów wyrażonych w postaci Objectives and key results (pol. celów i ich głównych rezultatów).

- KPIs,

  Główne wskaźniki jednostki/departamentu/obszaru/organizacji. Są to miary pokazujące zwłaszcza bieżącą kondycję (tzw. health checki; wariacją w tym temacie są health checki stosowane w modelu Spotify, które skupiają się na zespołach.

- **pętle informacji zwrotnej na poziomie użytkownika**

  Niżej wymienione wskaźniki zostały w opisane przez Konrada w rozdziale dotyczącym współpracy z UX :

  - raporty analityczne,
  - testy A/B,
  - liczba błędów pokazujących się użytkownikom ,
  - różnego rodzaju badania wśród użytkowników,
  - liczba transakcji,
  - komentarze, oceny, podsumowania produktu,

- zaangażowanie użytkowników w mediach społecznościowych

- **osobiste, międzyludzkie pętle informacji zwrotnej**
  - happiness metrics  ,
  - men
  - rozmowy face-2-face (pol. twarzą w twarz, bezpośrednie) przełożonego i pracownika,
  - ocena roczna.

Wymienione wyżej elementy składają się na feedback loop. Nie zawsze wszystkie istnieją w danym przedsiębiorstwie. Proponuję pomyśleć o nich w kontekście:

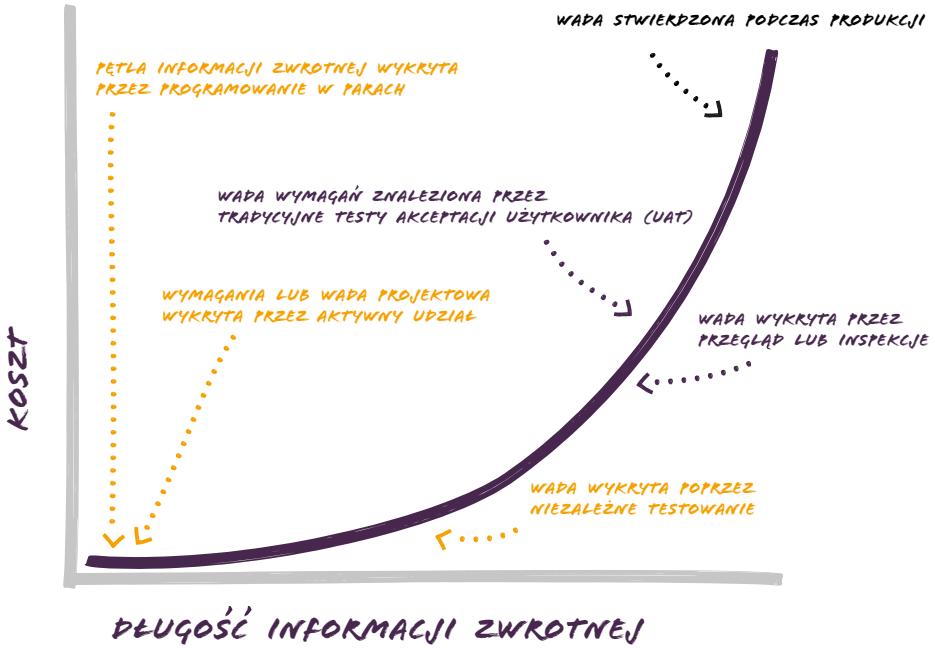
- **podniesienia efektywności procesu wytwórczego,**
- **podniesienia zadowolenia użytkowników z produktu,**
- **podniesienia zadowolenia wśród pracowników.**

Następnie wybrać z powyższej listy te, które możemy i chcielibyśmy sprawdzać.

Istnieje jeszcze inny kontekst feedback loop. Nie jest tak, że wprowadzamy jakąś pętlę informacji zwrotnej tylko dla jej wprowadzenia. Każda pętla ma czemuś służyć i dostarczać informacji. Weźmy jedną przykładową grupę - „pętle informacji zwrotnej na poziomie dewelopmentu". W ramach niej skupmy się na błędach. Będą się one pojawiały w ramach różnych elementów wymienionych w tej grupie: od Pair Programmingu po samo wdrożenie. Koszt wykrycia błędu, w zależności od czasu, jakiego potrzeba, by błąd znaleźć, możecie zobaczyć poniżej.

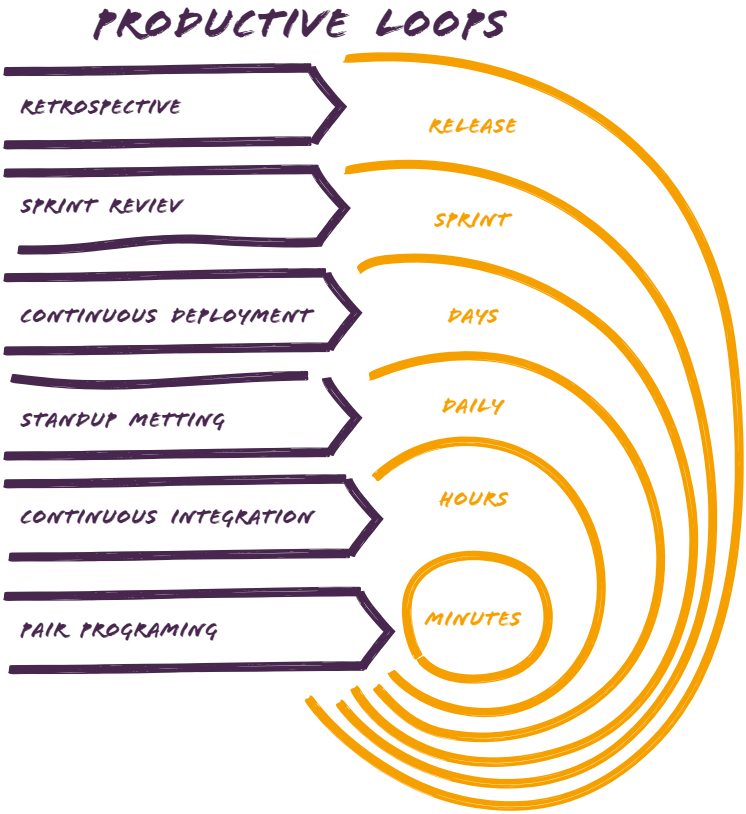
Zaprojektuj feedback loop tak, aby jak najszybciej wykrywać błędy w tworzonym oprogramowaniu.

Wykres pochodzi z artykułu „Independent testing and Agile teams" Scotta Ambler.



Jak zauważycie, im później dany błąd wykryjemy, tym jego koszt będzie większy. I to dużo większy, jeśli wykrycie nastąpi na końcowych etapach produkcji oprogramowania lub już po jego wdrożeniu.

I tak właśnie patrzę na feedback loop. Poszczególne elementy w ramach odrębnych grup dają informację zwrotną (która jest bardzo cenna), a jeżeli spojrzymy na kilka elementów na raz (jak np. na błędy), zauważymy, że powinniśmy, oprócz indywidualnego spojrzenia, patrzeć również całościowo, ponieważ może nam to pokazać trochę inną perspektywę.



Na powyższym rysunku wymienione są poszczególne elementy informacji zwrotnej na każdym poziomie tworzenia oprogramowania. Każdy z nich może dać nam dużą wartość. Dodatkową, zupełnie inną wartość otrzymamy, jak spojrzymy na proces produkcji całościowo (w tym przypadku możemy patrzeć już np. na pętle dotyczące produktu, a nie samego procesu wytwórczego).