

Podstawy metodyki SCRUM

Nowoczesne sposoby prowadzenia projektów w organizacji

Redakcja:

Romuald Lenarski

Autorzy:

Marta Matylda Kania, Izabela Pięta,
Julia Gutkind, Piotr Marchewka,
Katarzyna Wojciechowska

Słownik pojęć **SCRUM**

Popularne **błędy** projektowe

Praktyczne zastosowanie **SCRUM** w organizacji



Wrocław 2023

Copyright © All rights reserved

Pełny tytuł

Podstawy metodyki Scrum.

Nowoczesny sposób prowadzenia projektów w organizacji

Autorzy

Marta Matylda Kania, Izabela Pięta, Julia Gutkind,

Piotr Marchewka, Katarzyna Wojciechowska

Redakcja

Romuald Lenarski

Zdjęcia

Adobe Stock, system Firmbee

IFIRMA SA

Grabieżyńska 241G,

53-234 Wrocław

NIP: 898-16-47-572

REGON: 931082394

E-mail: contact@firmbee.com

www.firmbee.com

www.ifirma.pl



Firmbee



IFIRMA

Spis treści

08 Czym jest Scrum?



- 08 Filozofia, teoria i struktura
- 09 Wartości Scrum
- 11 Jak wdrożyć Scrum w swojej firmie?

13 Role i obowiązki Scrum Team



- 13 Skład Scrum Team
- 15 Product Owner
- 20 Scrum Master
- 28 Współpraca Scrum Mastera z Product Ownerem
- 30 Zespół Developerski
- 34 Skalowanie Scrum

36 Artefakty Scrum



- 37 Backlog Sprintu
- 39 Backlog Produktu
- 42 User Stories
- 50 Estymacja i Story Points w Scrum
- 55 Przyrost w Scrum

57 Wydarzenia w Scrum



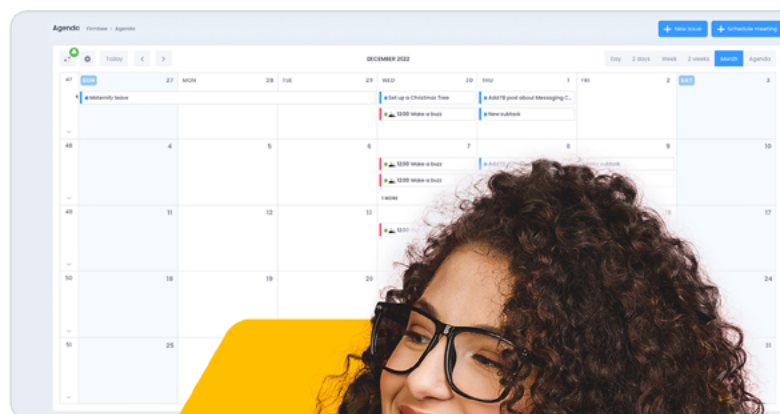
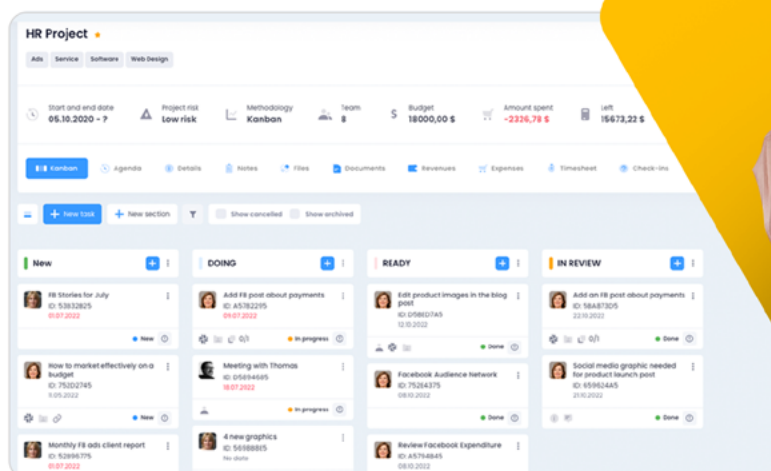
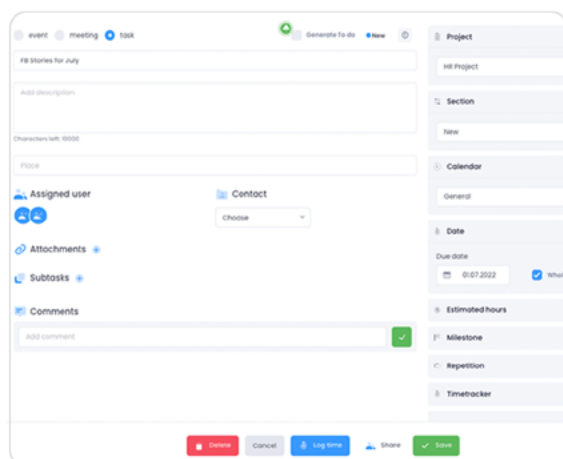
- 59 Sprint w Scrum
- 61 Cel Produktu, Cel Sprintu i Definicja Ukończenia
- 63 Wykres Spalania
- 70 Tablice Kanban w Scrum i Scrumban
- 72 Prędkość Zespołu Developerskiego
- 74 Daily Scrum
- 76 Sprint Planning
- 77 Sprint Review
- 78 Sprint Retrospective

77% zespołów o wysokiej wydajności korzysta z oprogramowania do zarządzania projektami.

Dołącz do nich dzięki Firmbee!

- Planuj i śledź postępy swoich projektów za pomocą **tablic Kanban**
- Współpracuj** z członkami swojego zespołu
- Zarządzaj **czasem i budżetem**

Dowiedz się więcej



Prezentowane treści są chronione prawami autorskimi. Ich kopiowanie, przetwarzanie, publikowanie oraz wykorzystywanie (w całości lub w części) bez zgody ich autora/ów jest zabronione oraz stanowi naruszenie Ustawy o prawie autorskim i prawach pokrewnych.

Scrum jest skuteczny. Pozwala firmom rozwijać się i skalować ponieważ sprawia, że zespoły uczą się na własnych błędach. Daje członkom zespołu poczucie odpowiedzialności za to co robią oraz ustala stały cykl wydarzeń, dzięki któremu prowadzenie biznesu w dowolnej branży staje się bardziej przewidywalne. Zasady Scrum są proste i dostępne, zaś jego działanie przetestowały w praktyce niezliczone firmy - od pięcioosobowych ekip specjalistów po wielkie organizacje. Dowiedz się na czym dokładnie polega ta metodologia z naszego e-booka. Zapraszamy do lektury!

Słowniczek Scrum

Podstawą komunikacji zespołu pracującego wspólnie w Scrum jest jasność i zrozumiałość wszystkich używanych terminów. Dlatego zebraliśmy poniżej definicje najważniejszych pojęć. Zapraszamy do korzystania ze słowniczka szczególnie podczas lektury wpisów wprowadzających nowe terminy Scrum.

Podstawy Scrum

- **Scrum** - sposób pracy Scrum Team, w którym tworzenie produktu podzielone jest na krótkie iteracje - Sprints. Jego celem jest szybkie i elastyczne dostarczanie wartościowych rozwiązań.
- **Wartości Scrum** - 5 wartości leżących u podstaw Scrum to: zaangażowanie w pracę, skupienie na celu, otwartość na zmiany, szacunek oraz odwaga w rozwiązywaniu problemów.
- **Empiryzm** - ograniczanie planowania i założeń do minimum na rzecz oparcia się na doświadczeniu, obserwacji i eksperymentowaniu.
- **Samozarządzanie** - cecha Scrum Team, która oznacza, że jego członkowie samodzielnie podejmują decyzję o tym, kto z nich będzie wykonywał określone zadania, kiedy i jak.

Role w Scrum

- **Scrum Team** - samozarządzający zespół składający się ze Scrum Mastera, Product Ownera i Developerów, zwanych też Zespołem Developerskim.
- **Product Owner** - przedstawiciel klienta wewnątrz Scrum Team. Osoba odpowiedzialna za maksymalizację wartości produktu przez dbałość o Cel i jego realizację przez Zespół Developerski.
- **Scrum Master** - coach i lider Zespołu Developerskiego. Osoba odpowiedzialna za właściwe rozumienie Scrum i działanie według jego zasad.
- **Developer** - każdy członek Zespołu Developerskiego, niezależnie od specjalizacji zawodowej. Osoba odpowiedzialna za współtworzenie użytecznego Przyrostu w każdym Sprincie.
- **Zespół Developerski** - interdyscyplinarna grupa składająca się ze wszystkich Developerów zaangażowanych w tworzenie Produktu.

- 🟡 **Interesariusz** - osoba nienależąca do Scrum Team, w której interesie leży stworzenie jak najlepszego Produktu. W ramach Scrum Team reprezentowany przez Product Ownera. Bierze udział w Sprint Review.

Artefakty Scrum i ich składowe

- 🟡 **Artefakty** - Backlog Produktu, Backlog Sprintu, Przyrost i ich składowe. Stanowią odzwierciedlenie aktualnego stanu pracy nad Produktem w odniesieniu do Celu Produktu, Celu Sprintu i Definicji Ukończenia.
- 🟡 **Backlog Produktu (Product Backlog)** - uporządkowana lista prac potrzebnych do stworzenia określonego Produktu, czyli realizacji Celu Produktu. Jest zarządzany przez Product Ownera.
- 🟡 **Backlog Sprintu (Sprint Backlog)** - uporządkowana lista prac potrzebnych do realizacji funkcjonalności Produktu zdefiniowanej przez Cel Sprintu. Jest zarządzany przez Zespół Developerów.
- 🟡 **Przyrost (Increment)** - kompletna i wartościowa praca wykonana przez Developerów w jednym Sprincie. Suma wszystkich Przyrostów tworzy Produkt.
- 🟡 **User Story** - opis częściowej funkcjonalności Produktu z punktu widzenia Klienta. Przyjmuje formę schematu „Jako [typ użytkownika] chcę [co zrobić?], ponieważ [po co? dlaczego?]”.
- 🟡 **Definicja Ukończenia (Definition of Done)** - zamieszczony w Backlogu Produktu jasny i przejrzysty opis oczekiwanego stanu Produktu po ukończeniu Przyrostu. Opisuje pracę, jaka została wykonana w ramach Przyrostu.

Wydarzenia Scrum i ich składowe

- 🟡 **Wydarzenia** - dotyczą pracy Scrum Team, jej planowania lub refleksji nad nią. Są to: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective i ich składowe.
- 🟡 **Sprint** - cykliczne wydarzenie obejmujące pracę nad nową wersją Produktu. Trwa najczęściej dwa tygodnie, a najdłużej miesiąc. Służy jako „pojemnik” na inne wydarzenia i działania Scrum.
- 🟡 **Sprint Planning** - spotkanie Scrum Team, w ramach którego wybierane są z Backlogu Produktu wszystkie prace do wykonania w kolejnym Sprincie. Trwa maksymalnie 8 godzin.
- 🟡 **Daily Scrum** - codzienne spotkanie Zespołu Developerów, na którym planowane są zadania na dany dzień trwa maksymalnie 15 minut, odbywa się zawsze w tym samym miejscu i czasie.
- 🟡 **Sprint Review** - wydarzenie podsumowujące ukończony Sprint pod kątem Celu Produktu. Dla Scrum Team i Interesariuszy. Jego celem jest ocena Przyrostu i aktualizacja Backlogu Produktu. Trwa 4 godziny lub mniej.

- 🟡 **Sprint Retrospective** - wydarzenie podsumowujące ukończony Sprint pod kątem sposobu pracy Scrum Team. Jego celem jest poprawa funkcjonowania Scrum Team. Trwa maksymalnie 3 godziny.
- 🟡 **Cel Produktu** - opis przyszłego Produktu, nad którego tworzeniem pracuje Scrum Team. Droga prowadząca do osiągnięcia Celu Produktu jest zapisana w Backlogu Produktu.
- 🟡 **Cel Sprintu** - praca do wykonania w ramach jednego Sprintu wyrażona w formie celu biznesowego. Zapewnia spójność pracy Zespołu Developerów.
- 🟡 **Wykres Spalania (Burndown Chart)** - pokazuje ilość pracy zaplanowanej w Backlogu Sprintu lub Produktu w odniesieniu do czasu, jaki pozostał na jej wykonanie.
- 🟡 **Prędkość Zespołu (Velocity)** - wskaźnik pozwalający określić jaka część Backlogu Produktu stała się Przyrostem w trakcie jednego Sprintu.

Czym jest Scrum?

Scrum to najpopularniejszy sposób nowoczesnego, elastycznego zarządzania pracą zespołową.

Jest używany nie tylko przy tworzeniu oprogramowania. Coraz częściej wykorzystują go również zespoły z branż tak odległych od siebie jak finanse i marketing, HR i branże kreatywne. Jego wciąż rosnąca popularność sprawia, że Scrum to w tej chwili najbardziej sprawdzone ramy organizacyjne, z których korzystają zespoły dążące do maksymalnej efektywności.

Scrum z założenia jest prosty. Po pierwsze, pozwala rozłożyć trudne problemy na mniejsze, bardziej zrozumiałe składowe. Po drugie, pozwala podzielić pracę na etapy, które mogą zostać w pełni wykonane w krótkim czasie przynosząc wymierny i satysfakcjonujący efekt. Po trzecie, pozwala zaplanować kolejne etapy pracy korzystając z uzyskanych rezultatów i wniosków wyciągniętych z procesu realizacji.

Filozofia, teoria i struktura



Należy pamiętać, że **Scrum wyznacza jedynie ramy postępowania. Bazując na nich trzeba każdorazowo zbudować szczegółowy sposób pracy dostosowany do potrzeb i możliwości zespołu oraz organizacji.**

Pomimo ogólności, są to ramy świetnie zarysowane. Świadczy o tym ogromna popularność Scrum. Według raportu 15th Annual State Of Agile z 2021 roku, zasady Scrum stosuje aż 66% zespołów pracujących w zgodzie z najnowocześniejszymi metodykami, a odsetek ten wzrasta do ponad 80% zespołów z różnych dziedzin, jeśli dodamy metodyki wywodzące się bezpośrednio ze Scrum.

Scrum jest wszechstronny i służy do optymalizacji pracy zespołowej. Stanowi on dobrze określony punkt wyjścia. Ogólność zasad Scrum sprawia, że nie da się po prostu z dnia na dzień „zastosować Scrum”. Szkicowość tych ram jest jednak celowa i wypływa z praktyki zarządzania projektami. U podstaw filozofii Scrum leży bowiem założenie o potrzebie ciągłego rozwoju opartego na refleksji i własnym doświadczeniu. Odrzucone zostają więc złożone, sztywne systemy organizujące pracę bez uwzględniania konkretnych realiów. Autorzy Scrum, Ken Schwaber i Jeff Sutherland, nazywają tą zasadę empiryzmem w oficjalnym Przewodniku po Scrum.

Teoria Scrum

Dla teorii Scrum kluczowe jest jedno słowo: empiryzm. **Empiryzm oznacza ograniczanie planowania i założeń do minimum na rzecz oparcia się na doświadczeniu, obserwacji i eksperymentowaniu.** Staje się to możliwe i efektywne dzięki podejściu iteracyjnemu, czyli pracy w krótkich cyklach, w których zawiera się nie tylko praca nad produktem, ale także jej planowanie i ocena rezultatów. Najważniejsze dla skuteczności działania Scrum są tutaj trzy filary empiryzmu:

- 🟡 **przejrzystość** - dzięki niej zarówno osoby pracujące, jak i Interesariusze są w stanie łatwo sprawdzić, jaki w danym momencie jest stan prac nad Produktem,
- 🟡 **inspekcja** - oznacza częste i rzetelne aktualizowanie oraz sprawdzanie postępów, dzięki któremu możliwe jest szybkie wykrycie problemów i ich rozwiązanie,
- 🟡 **adaptacja** - polega na korekcie sposobów pracy oraz Celów, jeśli w czasie inspekcji pojawiły się błędy lub rozbieżności.

Empiryzm sprawdza się najlepiej, jeśli zespół działający według jego zasad ma zdolność samzarządzania zgodnie z koncepcją lean. Oznacza ona elastyczność struktury organizacyjnej, która umożliwia adaptację do istniejących warunków, ciągłe doskonalenie się oraz niezależność Scrum Team.

Struktura Scrum

Scrum wyznacza ramy działania zespołowego definiując: skład i role w zespole nazywanym Scrum Team, rytm jego działań i spotkań zwanych Wydarzeniami Scrum oraz sposób planowania zadań i ich organizacji w ramach Artefaktów Scrum.

Scrum Team to niezależny, interdyscyplinarny zespół specjalistów pracujący w Scrum,

wolny od napływu dodatkowych zadań z organizacji. Jest podstawą skutecznej pracy w Scrum.

Scrum Team składa się z Product Ownera, Scrum Mastera oraz Zespołu Developerskiego. Jest to niewielki zespół o możliwie niezmiennym składzie, pracujący nad określonym Celem. Scrum Team powinien nieustannie poprawiać i ulepszać nie tylko produkt, lecz także własny sposób działania.

Pozwala to na zwiększanie efektywności jakości pracy zespołowej. Działania i spotkania Scrum Team nazywane są Wydarzeniami Scrum. Należą do nich Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective i ich składowe. Sposób planowania i realizacji oraz warunki skutecznego osiągania celów opisane są w Artefaktach Scrum, czyli w Backlogu Produktu i Backlogu Sprintu. Są one bardzo często aktualizowanymi dokumentami odzwierciedlającymi aktualny stan pracy nad Produktem.

Wartości Scrum

Wartości Scrum nie zawsze były obecne w oficjalnym Przewodniku po Scrum. Zostały one dodane na prośbę użytkowników do wydania opublikowanego w 2016 roku. Opiszmy po kolei Wartości Scrum, czyli: **zaangażowanie, skupienie, otwartość, szacunek, oraz odwagę**. Aktualizacja Przewodnika po Scrum przebiegła zgodnie z zasadą empiryzmu, czyli została przeprowadzona na podstawie doświadczeń zespołów wdrażających Scrum. Wynikało z nich bowiem, że nawet ścisłe przestrzeganie organizacji pracy w Scrum nie zawsze wywoływało pozytywne zmiany w funkcjonowaniu zespołów. Wprowadzenie Wartości Scrum miało zaś na celu jasne określenie właściwego podejścia sprzyjającego realizacji zasad Scrum.

1. Zaangażowanie

Pierwszą wartością, która leży u podstaw właściwego wdrażania Scrum jest zaangażowanie w pracę. Zespołowi Developerskiemu działającemu według zasad Scrum musi zależeć na realizacji Celu Sprintu oraz Celu Produktu. Jednak jeszcze ważniejsza jest troska o zaangażowanie każdego Developera z osobna. Zapewnia ono nie tylko aktywne dążenie do rozwiązywania problemów pojawiających się podczas wypełniania codziennych obowiązków. Obejmuje również chęć udzielania wsparcia innym członkom Zespołu Developerskiego, kiedy tego potrzebują.

2. Skupienie

Jeśli każdy z Developerów z pełnym zaangażowaniem koncentruje się na zadaniach cząstkowych, łatwo może stracić z oczu główny Cel Produktu, a nawet Cel Sprintu. W rezultacie Przyrost wytworzony w danym Sprincie okaże się niekompletny. Dlatego skupienie na Celu Sprintu jest niezmiennie istotne z punktu widzenia pracy zespołowej. Dzięki skupieniu zyskuje na jakości nie tylko poziom zadań wykonywanych przez poszczególnych Developerów. Także ich rezultaty łączą się w jedną całość realizującą Cel Sprintu. A tym samym zostaje zrealizowana wartość biznesowa: powstała funkcjonalność Produktu, która miała zostać wypracowana w tym przebiegu pracy Scrum Team.

3. Otwartość

Podejście zwinne różni się od tradycyjnego przede wszystkim otwartością na zmiany. Dlatego właśnie znalazła się ona wśród głównych Wartości Scrum. Zarówno Scrum Team jak i Interesariusze zyskują pielęgnując wartość otwartości w odniesieniu do wykonywanej pracy. Zmienność, o której mowa dotyczyć może wielu obszarów projektu, na przykład: zmniejszenia zakresu projektu, redefinicji funkcjonalności Produktu, czy też reorganizacji zadań wykonywanych przez Scrum Team w celu optymalizacji efektywności.

Zmiany mogą wynikać z decyzji Interesariuszy, ograniczeń biznesowych bądź technicznych. Otwartość w obliczu zmiany jest wspierana przez przejrzystość będącą jednym z filarów empiryzmu w Scrum. Dzięki przejrzystości członkowie Scrum Team oraz Interesariusze wiedzą, z czego wynika i do czego ma prowadzić wprowadzana zmiana.

4. Szacunek

Członkowie Scrum Team są niezależnymi, równymi sobie jednostkami. Wszyscy Developerzy, Scrum Master i Product Owner są profesjonalistami, którzy traktują się nawzajem z należnym sobie szacunkiem. W Scrum Team nie powinno być miejsca na hierarchię. Szacunek służy również zapewnieniu psychologicznego bezpieczeństwa. Dotyczy to szczególnie członków Zespołu Developerskiego. Dzięki wzajemnym okazywaniu sobie szacunku mogą oni swobodnie eksperymentować i nie bać się popełniania błędów, które są niezbędną częścią procesu innowacji i prowadzą do poprawy efektywności Scrum Team.

5. Odwaga

Odwaga w rozwiązywaniu problemów jest - obok szacunku - niezbędnym warunkiem udanego eksperymentowania. Sprawia również, że w obliczu trudnych zadań Scrum Team podejmuje i ponawia próby ich rozwiązania, aż do otrzymania satysfakcjonującego rezultatu. Odważne proponowanie rozwiązań biznesowych, technicznych i interpersonalnych pozwala wykorzystać potencjał innowacji tkwiący w Scrum. Odwaga jest także fundamentem samozarządzania Scrum Team, który bierze pełną odpowiedzialność za konsekwencje decyzji podejmowanych przez osoby należące do zespołu.

Jak wdrożyć Scrum w swojej firmie?

Choć ramy Scrum są proste, jego wdrożenie w firmie nie należy do łatwych zadań. Scrum służy do optymalizacji pracy zespołowej, jednak początkowo może przysporzyć wielu kłopotów, a nawet spotęgować problemy istniejące w organizacji. Jak zatem zabrać się do wdrażania Scrum?



Na pomysł zastosowania zasad Scrum w swojej firmie wpada wielu przedsiębiorców. Scrum obiecuje bowiem świetną efektywność zespołu, energetyczną atmosferę i brak hierarchii. Stawia jednak bardzo określone wymagania zarówno zespołowi, jak i poszczególnym osobom wchodzącym w skład Scrum Team. **Podejmując decyzję o wdrożeniu Scrum, warto zastanowić się nad szczegółami technicznymi organizacyjnymi.** Przyjrzyjmy się więc potencjalnym problemom z wdrożeniem, potrzebnym kompetencjom członków zespołu, oraz sposobom na wyodrębnienie Scrum Team z całości organizacji.

1. Wdrożenie. Zadanie dla Scrum Mastera

Podczas wdrażania Scrum szczególnie ważną osobą będzie doświadczony Scrum Master, który jest osobą odpowiedzialną za właściwe rozumienie Scrum przez wszystkich zainteresowanych. Będzie także sprawdzał i korygował działanie według zasad i Wartości Scrum. Zatrudnienie Scrum Mastera z doświadczeniem jest bardzo istotne w przypadku, gdy pracownicy firmy nie pracowali dotychczas według zasad Scrum. Pewnie będą mieli mnóstwo pytań, a Wydarzenia Scrum będą wymagały jego szczegółowego przewodnictwa co najmniej przez kilka pierwszych tygodni.

Zadania Scrum Mastera ograniczą się do ról coacha i lidera, gdy członkowie przyszłego Zespołu Developerskiego poznają już wystarczająco dobrze zasady Scrum. Przed rozpoczęciem innych działań odpowiedz sobie zatem na pytanie: *Czy jesteś w stanie znaleźć i zatrudnić odpowiedniego Scrum Mastera?*

2. Interdyscyplinarność. Jak zbudować Zespół Developerski?

Kolejne pytanie, na jakie musi odpowiedzieć sobie osoba wdrażająca Scrum brzmi:
„Czy w mojej firmie są już zatrudnione osoby zdolne stworzyć samowystarczalny, interdyscyplinarny zespół?”

Szczegółowy opis działania Zespołu Developerskiego w Scrum opisujemy w dalszej części e-booka. Tutaj wspomnimy tylko o dwóch problemach, które mogą pojawić się podczas tworzenia Zespołu: brak wystarczająco dojrzałych (senior) pracowników o uzupełniających się kompetencjach oraz sztywna struktura organizacji.

Wszystkie osoby, które mają wejść w skład Scrum Team powinny być specjalistami w swojej dziedzinie.

Zaś ich kompetencje powinny się uzupełniać. Dobrze skomponowany, interdyscyplinarny zespół nie powinien być więc zależny od pomocy zewnętrznych specjalistów. Jest to szczególnie ważne, jeśli Zespół pracuje z wrażliwymi danymi, które nie powinny być udostępniane osobom spoza organizacji. Korzystanie z zewnętrznej pomocy zaburza także funkcjonowanie jednego z filarów Scrum, przejrzystości. Może także stwarzać zagrożenie stworzenia hierarchii wewnątrz Zespołu. Na przykład wyodrębnienia „Developerów drugiej kategorii”, osób, które nie będą brały pełnego udziału w działaniach Scrum Team.

Problemy z wdrożeniem Scrum mogą pojawić się w przypadku, gdy firma jest podzielona na ściśle wyodrębnione działy. Jeśli każda z osób, które mają tworzyć Zespół Developerski, pracuje w innym dziale - potrzebna będzie spora reorganizacja. Jednym z tematów do przemyślenia jest chociażby wspólne miejsce pracy interdyscyplinarnego zespołu.

3. Rytm Scrum. Wyodrębnienie Scrum Team

Sprawą wartą przemyślenia podczas wdrażania Scrum jest stworzenie pewnego rodzaju „zapory ogniowej” chroniącej świeży Scrum Team przed napływem zadań z zewnątrz.

Zapewne bowiem utworzą go osoby, które pracowały przy innych projektach w twojej firmie. Siłą nawyku osoby, z którymi współpracowali członkowie nowego Scrum Team będą nadal szukać ich pomocy. A to może generować konflikty, powodować napływ dodatkowych zadań, a także zaburzać rytm Wydarzeń Scrum.

Czy na pewno warto wdrażać Scrum?

Jeśli zastanawiasz się poważnie nad wdrożeniem Scrum w swojej firmie i masz świadomość problemów, które mogą się przy tym pojawić, zastanów się raz jeszcze, czy na pewno Scrum jest rozwiązaniem dla Ciebie. **„Tak” powiedziało metodyce Scrum aż 66% zespołów zwinnych.** Statystyki nie pokazują jednak, jak skuteczne są zespoły świeżo po wdrożeniu. Ani jak dużo czasu zajmuje dojście do efektywności podobnej do tej sprzed zmiany, a następnie jej prześcignięcie. Warto zwrócić także uwagę na wielkość Scrum Team i objętość projektów, nad którymi ma on pracować.

Role i obowiązki Scrum Team

Scrum Team to zespół pracujący według zasad Scrum. Jego najważniejszą cechą jest brak wewnętrznej hierarchii. Gdy cel i zakres zadań do wykonania w danym Sprincie zostanie uzgodniony, członkowie zespołu przejmują inicjatywę. Od tego momentu samodzielnie podejmują decyzje o tym kto, jak i kiedy będzie wykonywać poszczególne zadania. Jak działa taki samzarządzający się Scrum Team?



Scrum Team jako całość ponosi odpowiedzialność za swoje działania, dlatego musi posiadać uprawnienia do zarządzania swoją pracą. Jest to szczególnie ważne w większych organizacjach. Jeśli zespół ma działać w zgodzie z zasadami Scrum, rytm i sposób jego działania nie może być zakłócany z zewnątrz. Jednak oczywiście w ramach Scrum Team każda osoba ma określone własne obowiązki.

Głównym zadaniem całego Scrum Team jest działanie prowadzące do realizacji Celu Produktu.

Realizacja tego Celu rozłożona jest na krótkie Sprints. Scrum Team buduje w każdym Sprincie nową, działającą część projektu, czyli Przyrost. Małe zadania realizowane przez Scrum Team są w ramach Sprintu dobrze zdefiniowane i rozdzielone pomiędzy członków zespołu. W realizacji tego zadania przydatne jest narzędzie do zarządzania projektami i zespołem.

[Screen: System Firmbee - Tworzenie nowego zadania](#)

Skład Scrum Team

Scrum Team to zwykle mniej niż dziesięć osób. Powinien składać się z profesjonalistów o otwartych głowach, których kompetencje się uzupełniają. Skład zespołu Developerskiego zmienia się w zależności od realizowanego projektu. Jednak na miejscu pozostają zawsze dwie figury: Product Owner i Scrum Master.

1. Product Owner

Product Owner jest członkiem zespołu równym w hierarchii wszystkim pozostałym. Ale przede wszystkim jest głosem klienta w Scrum Team. **Do podstawowych zadań Product Ownera, czyli Właściciela Produktu, należy komunikacja z klientem, a następnie ustalanie priorytetów pracy zespołu.** Ma to odzwierciedlenie zarówno w Celu Produktu, jak i w zawartości Backlogu Produktu. To właśnie Product Owner odpowiada na pytania dotyczące Produktu zadawane przez pozostałych członków zespołu przyjmując perspektywę klienta. Dbą zatem o otwartość i zrozumiałość kierunku oraz celu działania zespołu Developerskiego. Akceptuje także pracę wykonaną przez Developerki i Developerów oraz zatwierdza przekazanie rezultatu klientowi.

2. Scrum Master

Scrum Master pomaga wszystkim członkom zespołu Developerów w zrozumieniu teorii i praktyki Scrum, a także pełni rolę pośrednika. Często jest bowiem równocześnie członkiem zespołu Developerskiego i osobą wspierającą Product Ownera. To właśnie Scrum Master odpowiada za funkcjonowanie zespołu i jego działanie zgodnie z zasadami Scrum. Do jego zadań należy dbałość o pracę zespołu. Pełni w nim rolę zarówno lidera jak i coacha. Scrum Master koncentruje się na obserwacji niedociągnięć i wdrażaniu ulepszeń. Dbą także o wydajność i efektywność pracy Developerów.

3. Developerzy

Developerami określa się wszystkich członków zespołu, którzy nie pełnią roli Product Ownera albo Scrum Mastera. Nazwa nie powinna was zmylić - nie chodzi tutaj wyłącznie o programistów. Developerami mogą być osoby o bardzo zróżnicowanych umiejętnościach i zakresach obowiązków. Do obowiązków Developerów należy planowanie swoich zadań, czyli tworzenie planu Sprintu zawartej w Backlogu Sprintu, a także codzienna praca nad rozwijaniem produktu w celu wytworzenia w każdym Sprincie użytecznego Przyrostu zgodnie z Definicją Ukończenia.

Działanie Scrum Team

Scrum Team ma charakter interdyscyplinarny. Nie znaczy to oczywiście, że każdy może wykonywać każde zadanie postawione przed Scrum Team. Jego członkowie razem wzięci posiadają wszystkie umiejętności niezbędne do wykonania zadania i zarządzania własną pracą. Dlatego tak ważne jest budowanie zaufania pomiędzy osobami należącymi do Scrum Team, a także budowanie przez organizację zaufania do Scrum Team jako całości i zapewnienie mu niezależności. To właśnie motywuje jego członków do podejmowania zadań trudniejszych, wykraczających poza codzienny zakres obowiązków.

Interdyscyplinarność pomaga nie tylko w zwiększeniu wydajności zespołu. Sprawia również, że specjaliści z różnych dziedzin pracujący jako Scrum Team mogą pomagać sobie wzajemnie w wykonaniu zadań postawionych przed zespołem, uzupełniając swoje umiejętności. To wymaga nauczania się współpracy, a także wymaga od każdego z członków zespołu ogólnej wiedzy na temat tego, czym zajmuje się każdy z nich. Dzięki temu uczestnicy Scrum Team, wiedzą do kogo zwrócić się o specjalistyczną pomoc w razie problemu z wykonaniem zadania.

Product Owner



Zdarza się, że złożony z doświadczonych specjalistów Zespół Developerski działa niewystarczająco sprawnie. Po przeanalizowaniu sytuacji często okazuje się, że dzieje się tak z powodu braku wytyczonego celu działania. Żeby zapobiec takim problemom, we frameworku Scrum przewidziano w każdym zespole miejsce dla osoby pełniącej rolę Product Ownera. **Tylko Product Owner może dokonywać wpisów w Backlogu Produktu.** Do niego należy też ostatnie słowo w razie wątpliwości dotyczących oczekiwań klienta.

Podstawowe obowiązki Product Ownera można streścić w kilku punktach. Należą do nich:

- 🟡 **współpraca z Klientem** - prowadzenie regularnych rozmów z klientem, które prowadzą do określania i do określania cech Produktu tworzonego przez Scrum Team; celem nadrzędnym jest tutaj stworzenie Produktu jak najlepiej odpowiadającego wymaganiom klienta,
- 🟡 **artykułowanie Celu Produktu** - czyli opracowywanie oraz określanie długoterminowego kierunku działań Scrum Team, a także ciągłe upewnianie się, że rozumieją go wszyscy członkowie zespołu,
- 🟡 **stanie na straży Backlogu Produktu** - temu, czym jest Backlog Produktu poświęcimy osobny podrozdział; teraz musi nam wystarczyć stwierdzenie, że jest to jeden z Artefaktów Scrum zdefiniowany w oficjalnym Przewodniku po Scrum jako:

„Ewoluuująca, uporządkowana lista tego, co jest konieczne do ulepszenia produktu. To jedyne źródło pracy podejmowanej przez Scrum Team.”

1. Głos klienta w Scrum Team

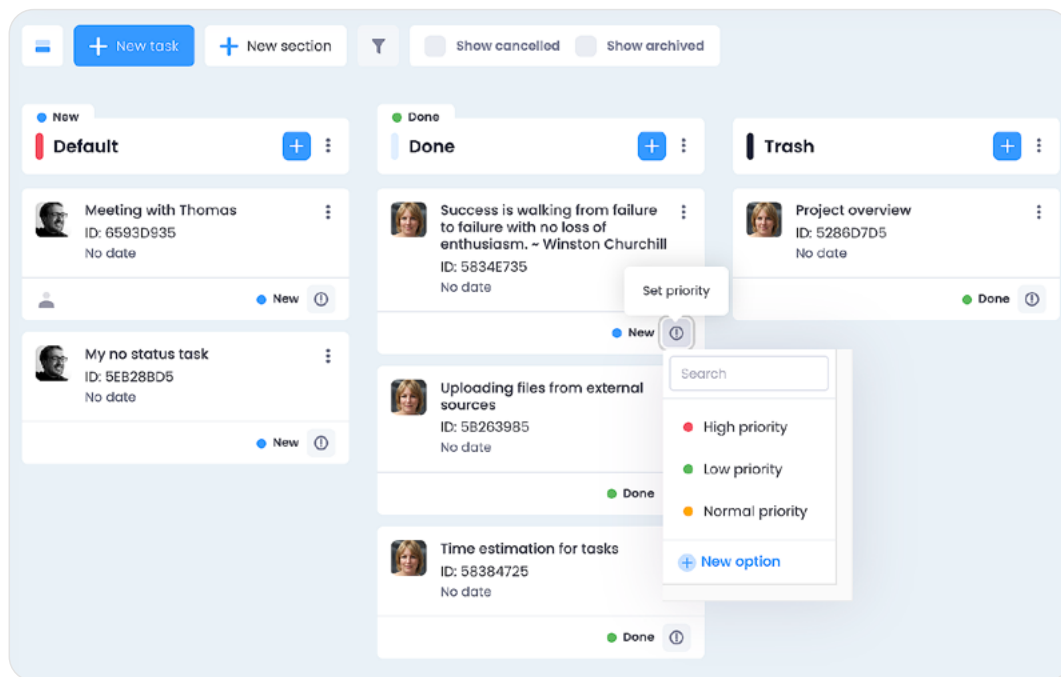
Product Owner jest głosem klienta w Scrum Team. **Dzięki obecności Product Ownera Zespół Developerów może na bieżąco rozwiewać swoje wątpliwości dotyczące wykonania powierzonych mu zadań.** Nie musi na własną rękę interpretować często nie do końca jasnych wytycznych pozostawionych przez klienta. Dlatego Product Owner jest tak ważnym członkiem zespołu. Podejmuje decyzje dotyczące Produktu, a także priorytetów Zespołu Developerskiego. I zgodnie z nimi tworzy oraz porządkuje wpisy w Backlogu Produktu.

Codzienne zaangażowanie Product Ownera i jego kontakt z Zespołem Developerskim i Scrum Masterem są kluczowe z powodu krótkiego czasu trwania Sprintu, czyli wytwarzania Przyrostu produktu. Nie ma tutaj czasu na wysyłanie pytań do osoby pracującej w innym środowisku i czekanie na odpowiedź. Product Owner jest przedstawicielem klienta należącym do zespołu i realnie w nim obecnym. Product Owner bierze także udział we wszystkich spotkaniach Scrum Team. Dzięki temu jest na bieżąco z postępami w pracach Zespołu Developerskiego. Wie też, z jakimi trudnościami zmagają się jego współpracownicy. W ten sposób może udzielać klientowi wiarygodnych informacji, a także sprawnie organizować pracę zespołu.

Rola Product Ownera sytuuje go na styku otoczenia Scrum Team, czyli środowiska biznesowego i Developerów. Organizuje pracę Zespołu Developerów. Również Product Owner decyduje, jakie są kryteria ukończenia pracy nad danym rozwiązaniem, a także zatwierdza moment ich spełnienia.

2. Właściciel produktu i jego cel

Product Owner jest odpowiedzialny za jasne postawienie i bieżące doprecyzowywanie Celu Produktu, czyli definiowanie celu działania zespołu. Innymi słowy, do jego obowiązków należy praca koncepcyjna i organizacyjna. Celem tej pierwszej jest praca z User Stories: tworzenie koncepcji produktu i jego funkcjonalności odpowiadającej potrzebom Klienta. Równie duża część obowiązków Product Ownera to praca organizacyjna - porządkowanie i priorytetyzacja zadań w Backlogu Produktu.



Screen: System Firmbee - ustalanie statusów i priorytetów zadań

Z perspektywy Scrum Team oznacza to dbanie o zrozumiałość horyzontu działań. Celowi Produktu poświęcimy osobny podrozdział.
Dziś natomiast posłużymy się przykładem:

Niech celem Scrum Team będzie stworzenie aplikacji na urządzenia mobilne służącej do organizowania pracy zespołu.

Zadaniem Product Ownera będzie wyjaśnienie członkom Scrum Team, jakie są kluczowe funkcjonalności aplikacji z punktu widzenia klienta. Na przykład - dodawanie nowych członków zespołu według listy kontaktów czy przełączanie widoku kalendarza zespołowego i osobistego. Gdy zespół zrozumie już Cel Produktu, do Product Ownera będzie należało:

- 🟡 **utrzymanie właściwego kursu** - aby zawsze najbardziej wyeksponowaną funkcjonalnością tworzonej aplikacji była zgodna z Celem Produktu, czyli organizowaniem pracy zespołu,
- 🟡 **wyjaśnianie bardziej szczegółowych kwestii** - wspólne z zespołem szukanie, uzgadnianie i doprecyzowanie działania Produktu,
- 🟡 **pilnowanie priorytetów** - w myśl zasady „first things first”, zadaniem Product Ownera będzie na przykład zapobieganie rozwijaniu pobocznych albo dodatkowych funkcjonalności aplikacji, drugorzędnych z punktu widzenia klienta.

Patrząc z perspektywy Klienta bądź jeszcze szerzej - wszystkich interesariuszy - Product Owner jest przede wszystkim osobą odpowiedzialną za tworzenie produktu. Bierze udział w rozmowach z osobami z zewnątrz Scrum Team oraz odpowiada głową za dostarczenie klientowi wartościowego produktu.

3. Strażnik Backlogu

Do codziennych obowiązków Product Ownera należy porządkowanie elementów Backlogu Produktu.

Jest to niekończąca się praca, ponieważ Backlog Produktu zmienia się nieustannie. Zawiera bowiem wszystkie znane kroki prowadzące do realizacji Celu Produktu. Oczywiście są one zaznaczone z różną szczegółowością.

Product Owner doprecyzowuje zadania zawarte w Backlogu Produktu, a także dzieli je na mniejsze części. Następnie decyduje, które z nich są gotowe do wejścia w etap realizacji. Z jednej strony praca z Backlogiem Produktu jest więc pracą administracyjną i organizacyjną, wymagającą rozumienia charakteru, możliwości i ograniczeń pracy Zespołu Developerów, a także zewnętrznych, biznesowych uwarunkowań tworzenia Produktu.

Z drugiej strony natomiast zadaniem Product Ownera jest tworzenie Backlogu Produktu zgodnie z potrzebami Scrum Team. Musi on być jasny, dostępny i zrozumiały dla wszystkich członków zespołu. Zaś zadaniem Product Owner jest odpowiadanie na wszystkie pytania i wątpliwości dotyczące produktu, jakie pojawiają się u Scrum Mastera i Developerów. Dzięki niemu cały zespół wie, co już zostało zrobione, co jest w trakcie realizacji i co jeszcze pozostało do zrobienia.

Najczęstsze błędy popełniane przez Product Ownera

1. Problemy na linii Product Owner - Klient

Product Owner jest osobą, która personalnie odpowiada za niepowodzenia Scrum Team. Z racji swojej pozycji wykraczającej poza działania zespołu mówi się, że jest single wringable neck, czyli pojedynczym karkiem do skręcenia. Innymi słowy, to właśnie Product Owner najbardziej ucierpi w sytuacji, gdy praca Scrum Team nie idzie po myśli Klienta i inwestora. Jak zatem radzić sobie w problematycznych sytuacjach i chronić się przed najczęstszymi błędami? Dla ułatwienia przedstawiliśmy problemy na linii Product Owner - Klient w tabelce. Pod nią znajdziecie szczegółowe omówienie każdego z problemów.

Błąd	Generowanie pomysłów	Sugestie rozwiązania
nieumiejętność ustalania priorytetów	nieoptymalizowany Backlog Produktu, rozmywanie się Celu Produktu	słuchanie, pytanie, negocjowanie Celu Produktu z Klientem, uważne opracowanie wyników negocjacji
brak asertywności	zbyt wiele zadań do wykonania przez Scrum Team	myślenie realistyczne, wiedza i pamięć o możliwościach zespołu
niewystarczające kwalifikacje biznesowe	ryzyko obniżenia wartości biznesowej Produktu wytworzonego przez Scrum Team	nieustanne uczenie się i zdobywanie kompetencji biznesowych

🔷 Nieumiejętność ustalania priorytetów

Błąd polegający na nieumiejętności ustalania priorytetów to zmora wielu Product Ownerów. Dlaczego ustalanie priorytetów zadań jest kluczową kompetencją? Ponieważ w momencie, gdy wszystko staje się tak samo ważne, znika z oczu Cel Produktu, czyli zamierzony efekt działania Scrum Team.

Problem zaczyna się już podczas pierwszych rozmów z klientami dotyczących Celu Produktu. Klient chce bowiem zwykle, aby zrealizować wszystkie jego pomysły jak najszybciej i jak najtaniej. Zadaniem Product Ownera jest więc ustalenie listy priorytetów. Jego zadanie polega na stworzeniu listy jasnych i możliwych do zrealizowania oczekiwań uszeregowanych od najważniejszych do najmniej ważnych. A trzeba to zrobić na podstawie nieustrukturyzowanych oczekiwań klienta.

Problem z priorytetami najczęściej ma źródło w niezrozumieniu oczekiwań klienta.

Pojawia się, kiedy Product Owner nie potrafi wydobyć z klienta informacji o realnym Celu Produktu. Czyli odpowiedzi na pytanie, na jakie potrzeby produkt ma odpowiadać. Jak więc chronić się przed tym błędem? Po pierwsze - słuchać uważnie klienta. Po drugie, nauczyć się zadawać pytania dotyczące Celu i sposobów działania poszczególnych funkcjonalności produktu. Po trzecie - negocjować i ograniczać Cele do zrealizowania. A do tego potrzebna będzie asertywność.

🔷 Brak asertywności Product Ownera

Problemem, który ściśle łączy się z nieumiejętnością priorytetyzacji jest brak asertywności. Wtedy do problemu nieodpowiednio ustawionej kolejki zadań prowadzącej do realizacji Celu Produktu dochodzi problem ich nadmiernej ilości. Dlatego **zdolność powiedzenia „nie” klientowi jest naprawdę kluczowa**. Asertywność Product Ownera powinna być oparta na trzech filarach: wiedzy o możliwościach zespołu, wiedzy o używanych i rozwijanych przez zespół rozwiązaniach, świadomości własnej roli i wartości opartej o miejsce w Scrum Team. Dlatego jednym z najważniejszych sposobów zapobiegania problemom z asertywnością jest codzienna praca Product Ownera ze Scrum Team. Dzięki temu zbuduje on realistyczne przekonania dotyczące czasu i możliwości realizacji pomysłów Klienta.

🔷 Niewystarczające kwalifikacje biznesowe

Następnym błędem, który chcielibyśmy omówić jest brak odpowiednich kwalifikacji biznesowych. Mocnymi stronami tych Product Ownerów są zwykle kwalifikacje specjalistyczne. Ich kompetencje wiążą się ściśle z obszarem działania Zespołu Developerskiego, niż z biznesem. Brakuje zatem ugruntowanej, praktycznej wiedzy dotyczącej konkurencji, zasad działania rynku, a także finalnego odbiorcy produktu tworzonego przez Scrum Team. Nie ma prostego remedium, które zaradziłoby temu problemowi. Szczególnie, że może on pojawiać się w bardzo specyficznych sytuacjach. Z pewnością jednak dobrym kierunkiem działania dla Product Ownera świadomego jego istnienia jest nieustanne uczenie się i zdobywanie doświadczenia i kompetencji biznesowych.

2. Problemy na linii Product Owner - reszta Scrum Team

Zdolność priorytetyzacji zadań, asertywność Product Ownera oraz jego wysokie kwalifikacje biznesowe składają się na umiejętności służące do stworzenia wzorowego Backlogu Produktu. Czyli długoterminowej podstawy działania Scrum Team. Jeśli Backlog nie został zarysowany spójnie i trafnie, problemy z relacji Product Owner - Klient przeniosą się na relację Product Owner - pozostali członkowie Scrum Team. A co za tym idzie, wpłyną bezpośrednio na skuteczność działania Scrum Team. Jakże jeszcze pułapki czekają na Product Ownera w relacjach z pozostałymi członkami Scrum Team? Dla ułatwienia przedstawiliśmy problemy na linii Product Owner - Scrum Team w tabelce. Poniżej znajdziecie szczegółowe omówienie każdego z problemów.

Błąd	Generowanie pomysłów	Sugestie rozwiązania
niewystarczająca charyzma	Zespół Developerski nie wykonuje zadań zawartych w Backlogu, zdanie Product Ownera jest podważane	budowanie autorytetu opartego na kompetencjach miękkich i wiedzy
niewystarczająca kwalifikacje specjalistyczne	niezrozumienie codziennego działania oraz możliwości Zespołu Developerskiego	orientacja w specjalizacjach członków zespołu, a także zdobywanie wiedzy na temat obszaru działania Zespołu
niesamodzielność	rozmywanie odpowiedzialności	wzmocnienie autorytetu

🔷 Niewystarczająca charyzma

W codziennej pracy Product Owner ma za zadanie koordynowanie wytycznych Klienta ze sposobem ich realizacji przez Zespół Developerski. Bez wątpienia wymaga to posiadania odpowiedniego autorytetu, posłuchu i charyzmy. Problemu niewystarczającego autorytetu nie da się rozwiązać z dnia na dzień. Wymaga długotrwałej pracy nad kompetencjami miękkimi. A także zdobywania wiedzy na temat zakresu zadań i umiejętności pozostałych członków zespołu.

🔷 Niewystarczająca kwalifikacje specjalistyczne

Rola Product Ownera nie jest rolą stricte techniczną. Jednak znajomość podstaw specjalistycznych umiejętności członków Zespołu Developerskiego może znacząco podbudować autorytet Product Ownera. Niewystarczające kwalifikacje w obszarze działania zespołu mogą nie tylko generować problemy z charyzmą i autorytetem Product Ownera. Błąd polegający na braku zainteresowania tym, w czym specjalizują się członkowie Zespołu Developerskiego, oraz podstawami ich kompetencji może generować zabawne sytuacje ale również sytuacje o katastrofalnych skutkach biznesowych i interpersonalnych.

Dlatego też, **aby Scrum Team mógł dostarczać produkty najlepszej jakości, Product Owner musi dokładnie zrozumieć ich istotę.** Zdobycie odpowiednich kwalifikacji nie powinno być trudne biorąc pod uwagę, że Product Owner jest częścią zespołu złożonego z profesjonalistów. Mogą oni służyć nie tylko wyjaśnieniami ale również podpowiedzi, z jakich źródeł warto czerpać wiedzę na temat ich dziedziny.

◆ **Niesamodzielność**

Product Owner musi potrafić samodzielnie podejmować decyzje. Oczywiście kluczową kwestią jest znajomość uwarunkowań Scrum Team, oraz nieustanne rozmowy z Zespołem Developerskim. Jednak to właśnie Product Owner odpowiada głową za skuteczność jego działania. Z tego powodu musi budować swój autorytet i brać odpowiedzialność za podejmowane decyzje. Ostatnie słowo dotyczące kierunku działania zespołu, priorytetyzacji i akceptacji zadań należy właśnie do niego.

Scrum Master

Scrum Master jest odpowiedzialny za codzienne funkcjonowanie Zespołu Developerskiego. Jego rola sytuuje się na pograniczu zadań kojarzonych zwykle z rolami lidera oraz coacha. Dlatego jest określany jako przywódca służebny. Na pierwszy rzut oka mogłoby się wydawać, że Scrum Master to tylko dziwna nazwa menadżera projektu. Jest to bowiem osoba, która upewnia się, że członkowie zespołu posiadają wszystkie umiejętności niezbędne do realizacji celu. **Prowadzi i podsumowuje spotkania Scrumowe, a także motywuje i wspiera zespół w jego codziennym dążeniu do wypracowania wartościowego Przyrostu.** Jest jednak kilka kluczowych różnic pomiędzy menadżerem projektu a Scrum Masterem.

1. Przywódca służebny

Przed wszystkim od menadżera projektu w tradycyjnie zarządzanych organizacjach oczekuje się, że wycofa się z działania zespołu po sprawdzeniu, że wszystko jest w porządku. Pojawi się znów po umówionym czasie żeby sprawdzić, czy zadania zostały wykonane. Inaczej rzecz się ma ze **Scrum Masterem. Scrum Master pracuje z zespołem przez cały czas. Tak jak Product Owner, jest on niezastąpionym i zawsze obecnym członkiem Scrum Team.** Jego udział w wykonywaniu codziennych prac jest konieczny. To do zadań Scrum Mastera należy bowiem utrzymywanie morali zespołu.

W wykonaniu tego zadania pomocna może być funkcja check-ins dostępna w systemie do zarządzania projektami. Drugą kwestią mocno odróżniającą postać menadżera projektu od Scrum Mastera jest kwestia rozwiązywania problemów napotkanych podczas zespołowej pracy. W tradycyjnym podejściu zespół powinien poradzić sobie z nimi sam. Natomiast **w Scrum w razie trudności Zespół Developerski powinien zwrócić się o pomoc do Scrum Mastera.** Tutaj jego rola najbardziej zbliża się do roli coacha.

Scrum Master jest coachem również, gdy buduje motywację całego Zespołu Developerskiego i każdego z osobna. Jednym z kluczowych zadań Scrum Mastera jest bowiem zachęcanie poszczególnych członków zespołu do rozwijania się. Może on wskazywać umiejętności, które warto udoskonalić albo proponować nowe, ciekawe ścieżki rozwoju. Przed wszystkim takie, które zwiększają interdyscyplinarność zespołu. Poszerzanie kompetencji ma na celu zwiększanie efektywności i zgrania Scrum Team, co skutkuje większą wewnątrzsterownością oraz lepszymi możliwościami samozarządzania.

Check-ins

What question do you want to ask?

What have you worked on this week?

Characters left: 150

How often do you want to ask?

☐ On selected days
 ☒ Once a week on
 ☐ Once a month on the first

On which day?

Mo Tu We Th **Fr** Sa Su

At what time of day?

☐ Beginning of the day (9:00am)
 ☒ End of the day (4:30pm)
 ☐ Let me pick a time...

Who do you want to ask?

Nothing selected

☐ Add all project members

Cancel

Save

Screen: System Firmbee - Funkcja check-ins

Zadaniem Scrum Mastera, jest także dbanie o koncentrację na wykonywaniu zadań i co ściśle z tym związane, ochrona zespołu przed napływem zadań z zewnątrz. Jeśli bowiem Scrum Team działa jako część większej organizacji, może się zdarzyć że jego praca jest zakłócana przez pilne potrzeby innych jednostek. To właśnie do zadań Scrum Mastera należy wyjaśnienie innym osobom, że zakłócenia rytmu pracy Scrum Team nie jest wskazane.

2. Strażnik Scrum

Kluczową kompetencją Scrum Mastera jest doskonała znajomość zasad Scrum, natomiast kluczowym zadaniem - dbałość o stosowanie się do zaleceń Scrum. Pomaga wszystkim osobom w zrozumieniu teorii i praktyki Scrum. **Ważną częścią obowiązków Scrum Mastera jest przedstawienie sposobu działania Scrum Team Klientowi i innym interesariuszom.** Od tego bowiem często zależy sprawna realizacja Celu Produktu. Klient musi być bowiem świadomy, że w ramach Scrum - tak jak i innych metodyk zwinnych - wyżej ceni się nieustanną współpracę z klientem od negocjacji umów. Oznacza to między innymi konsultowanie priorytetów projektu z Product Ownerem zamiast jednorazowego przygotowania specyfikacji.



Ponieważ Scrum Master stoi na straży Scrum, może być również jego ewangelistą w organizacji.

Jego zadania obejmują jednak przede wszystkim szkolenie osób należących do zespołu. **Scrum Master dba o to, aby zasady Scrum były jasne i zrozumiałe dla wszystkich osób należących do Scrum Team.**

Pilnuje też, by odbywały się wszystkie Wydarzenia Scrumowe. Do zadań Scrum Mastera należy również moderacja tych spotkań, a także dbanie, by mieściły się w wyznaczonych ramach czasowych. Do zadań Scrum Mastera należy też upewnienie się, że spotkania nie były stratą czasu, lecz realnie wspierały dążenie zespołu do osiągnięcia Celu. Innymi słowy, żeby były konstruktywne i produktywne.

Cechy dobrego Scrum Mastera

Dobry Scrum Master - czyli jaki Scrum Master? Na pewno taki, który pomaga zespołowi działać w najlepszy możliwy sposób. Osiąga to przede wszystkim maksymalizując wewnętrzny potencjał zespołu oraz usuwając przeszkody stojące na drodze do sprawnej realizacji Przyrostu. Potrzebne są do tego odpowiednie spojrzenie, podejście do problemów napotykanym przez zespół i miękkie umiejętności.

1. Organizacja i koordynacja

Idealny Scrum Master to osoba, która posiada umiejętność świetnej organizacji pracy. Dzięki temu będzie efektywnie i na bieżąco porządkować Backlog Produktu, czyli jeden z Artefaktów Scrum. Jednak umiejętności organizacyjne Scrum Mastera wykorzystywane są przede wszystkim do koordynacji pracy wykonywanej przez innych. Dlatego też warto, by równie sprawnie radził sobie z organizacją wydarzeń Scrum, jak z ich moderacją. Do jego zadań należy też uzupełnianie brakujących informacji i odpowiadanie na pytania Developerów oraz ułatwianie członkom zespołu dostępu do szkoleń i narzędzi.

Jak pisaliśmy już wcześniej, Scrum Master zajmuje się przede wszystkim pracą z ludźmi. Zarządza on zespołem zapewniając jego sprawne działanie. Dzięki niemu każdy z członków Scrum Team wie, jaka jest jego rola i zakres odpowiedzialności w zespole. Innymi słowy - każdy wie, co ma robić i w jakim celu. Posiadając bazę kontaktów w systemie do zarządzania projektami, do każdego użytkownika można zdefiniować m.in. jego rolę, nazwę obejmowanego stanowiska czy też stawkę wynagrodzenia.

Configuration Firmbee • Configuration • Users					+ New user	
Users					Display only active users	
USER	POSITION	ROLE	HOURLY RATE	ACTIVE		
▼ Beverly Mills	Owner	Admin	150,00 \$	<input checked="" type="checkbox"/>	...	
▼ Connor Jones	Web developer	Default User	90,00 \$	<input checked="" type="checkbox"/>	...	
▼ Donald Carlson	Developer	Default User	50,00 \$	<input checked="" type="checkbox"/>	...	
▼ Dwayne Peck	Developer	Default User	100,00 \$	<input checked="" type="checkbox"/>	...	
▼ Harry Miller	Web developer	Default User	50,00 \$	<input checked="" type="checkbox"/>	...	

Screen: System Firmbee - baza kontaktów

2. Przewodnik, nie kierownik

Scrum Master to specyficzna rola lidera. To rola przewodnika, który zamiast stawiać się ponad innymi, pomaga im w wykonywaniu zadań wskazując sposoby na zwiększenie produktywności i ulepszenie współpracy. Innymi słowy, dobry Scrum Master nie forsuje swojego zdania. Nie szuka też rozwiązań dotyczących tworzonego przez zespół Produktu. **Koncentruje się zupełnie na sposobie, w jaki pracuje zespół, dzięki temu pomaga innym w efektywnym poszukiwaniu rozwiązań i podpowiada, w jaki sposób współpracować.** Dlatego dla Scrum Mastera tak ważna jest umiejętność zadawania pytań. Pytań, które nie są tradycyjnie kojarzone z rolą osoby zarządzającej. Czyli, zamiast sprawdzać zadania i pytać - czy zostały wykonane prawidłowo i na czas? I dlaczego nie? Scrum Master pyta:

- 🟡 Jak mogę ci pomóc?
- 🟡 Co mogłoby pomóc zespołowi zrobić postępy w tym projekcie?
- 🟡 Czy coś nam nie umknęło?
- 🟡 Czy omówiliśmy już wszystkie problemy?
- 🟡 Co jeszcze może się nie udać?
- 🟡 Jak dużo czasu na to potrzeba?
- 🟡 Czy będziemy kontynuować to podejście w następnej iteracji?



Zadawanie pytań to jednak nie wszystko. Równie kluczowe będą tutaj umiejętności słuchania i korzystania z wiedzy, spostrzeżeń i pomocy udzielanej przez zespół.

3. Perswazja zamiast nakazu

Podstawą mocnej pozycji Scrum Mastera w Scrum Team jest równe traktowanie wszystkich członków zespołu. Bardzo ważne jest to, żeby każdy członek Zespołu Developerskiego czuł, że jego wkład w pracę zespołu jest doceniany. Upewnienie się, że tak jest należy do kluczowych obowiązków Scrum Mastera. Kolejnym krokiem jest wzmocnianie samodzielności członków zespołu dzięki wspieraniu ich rozwoju zawodowego i osobistego. Zespół Developerski złożony z równych i pewnych swych umiejętności jednostek będzie dążył do samodzielnego rozwiązywania problemów.

Oznacza to jednak sposób zarządzania znacznie trudniejszy od tradycyjnego. Scrum Master powinien bowiem stawiać dialog i dyskusję ponad forsowaniem własnego stanowiska, a naprowadzanie i stawianie pytań nad szukaniem rozwiązań za Developerów. Wymaga to oczywiście olbrzymiej cierpliwości ale także zdecydowania. Nie chodzi bowiem o to, żeby sporne kwestie pozostawały nierozstrzygnięte, dlatego właśnie Scrum Master musi opanować umiejętność wywierania wpływu na zespół, co wymaga dużej inteligencji emocjonalnej. Zwykle nie pracuje on bowiem na stanowisku, które czyniłoby go w tradycyjnym sensie przełożonym Zespołu Developerskiego.

4. Perspektywa z lotu ptaka

Scrum Master działa właśnie wewnątrz zespołu, a nie spogląda na jego pracę „z góry”. Nie dotyczy to jednak wszystkich sytuacji. Dzięki temu, że Scrum Master nie jest bezpośrednio zaangażowany w tworzenie produktu, może dostrzec prawidłowości i zależności niewidoczne dla członków Zespołu Developerskiego. Odmienność tej perspektywy sprzyja także motywowaniu innych. Pozwala bowiem wytyczyć horyzonty dla rozwoju i konsekwentnie wspierać członków Zespołu Developerskiego w dążeniu do nich.

Umiejętność spojrzenia na pracę zespołu „z lotu ptaka” sprawi, że zamiast wciąż pomagać Developerom w poprawianiu pojedynczych, powtarzających się błędów, Scrum Master może zaproponować systemową zmianę podejścia. Podtrzymywaniu nowej wizji sprzyja także energetyczność - kolejna cecha dobrego Scrum Mastera, który potrafi dodawać zespołowi energii.

Najczęstsze błędy popełnianie przez Scrum Mastera

Pracę dobrego Scrum Mastera poznać można po tym, że w pewnym momencie przestaje być potrzebny w codziennej pracy Zespołu Developerskiego. Jednak nie zawsze tak się dzieje. Jakie są przyczyny błędów popełnianych przy pełnieniu roli Scrum Mastera? Praca Scrum Mastera polega przede wszystkim na wspomaganiu pracy Zespołu Developerskiego. Dlatego najczęstsze popełniane przez niego błędy wpływają zwykle ze sposobu, w jaki uczestniczy on w codziennym funkcjonowaniu Developerów. Błędy te podzieliliśmy na dwie grupy. Pierwsza obejmuje problemy wynikające ze zbyt dużego zaangażowania, zaś druga z niewystarczającej obecności Scrum Mastera w życiu Zespołu Developerskiego.

1. Nadobecność Scrum Mastera

Potrzeba utrzymywania zbyt dużej kontroli nad Zespołem często powoduje błędy w stosowaniu Scrum. Błędy Scrum Mastera najczęściej stają się widoczne w następujących sytuacjach.

- 🟡 **Scrum Master szuka rozwiązania problemu zamiast wspomagać zespół w radzeniu sobie z trudnościami.** Często źródłem problemu jest to, że Scrum Master jest zarazem specjalistą w dziedzinie tego, czym zajmuje się Zespół Developerski. Nieumiejętność wyjścia z roli eksperta sprawia, że nie jest on w stanie skutecznie wspomagać zespołu w samodzielnym znajdowaniu rozwiązań. Takie podejście może także prowadzić do jednoosobowego, autorytarnego podejmowania decyzji - a to chyba największy błąd, jaki może popełniać Scrum Master.
- 🟡 **Scrum Master nie pozwala zespołowi popełniać błędów.** Ten problem wiąże się ściśle z poprzednim. Jeśli zespół będzie skutecznie chroniony przez Scrum Mastera przed popełnianiem błędów, nie nauczy się samodzielnie rozwiązywać problemów, ani brać odpowiedzialności za swoją pracę. Zawsze będzie polegał na radzie i ekspertyzie Scrum Mastera.
- 🟡 **Scrum Master próbuje zmieniać ludzi zamiast pracować nad atmosferą w zespole.** Problem ten dotyczy zarówno zbyt dużego nacisku na zmianę zachowań członka lub członków zespołu, jak i zmian personalnych. Błędem jest zmienianie składu Zespołu Developerskiego podczas pracy nad Celem Produktu jeśli nie jest to absolutnie konieczne. Może to wprowadzić znaczące opóźnienia w jego realizacji, zaburzyć rytm pracy Zespołu Developerskiego, a także zaburzyć rytm kształtowania się Zespołu.
- 🟡 **Scrum Master pełni w organizacji funkcję przełożonego Zespołu Developerskiego.** Jest to błąd, który nie wynika często z decyzji samego Scrum Mastera. Może jednak powodować nasilenie wszystkich błędów wynikających z potrzeby kontroli nad Zespołem.
- 🟡 **Scrum Master nadmiernie angażuje się w działanie Zespołu.** Gdy Zespół składa się z ekspertów znających swoje umiejętności i zakresy obowiązków, oraz funkcjonuje w zgodzie z zasadami Scrum,

Scrum Master nie powinien nieproszony ingerować w sposób jego działania. Jeśli to robi, po prostu przeszkadza w sprawnym działaniu swojego zespołu. Dobry Scrum Master dzięki ugruntowanej pozycji coacha i lidera będzie proszony o radę w sytuacjach nadzwyczajnych lub wymagających świeżego spojrzenia. Dlatego powinien być dostępny na wezwanie Developerów, ale nie narzucać swej obecności.

- 🟡 **Scrum Master zbyt sztywno trzyma się zasad Scrum.** Jeśli któryś z aspektów Scrum w danym Zespole nie działa, Scrum Master powinien spróbować innego podejścia. Każdy Zespół jest inny, a Scrum jest tylko ogólnymi ramami działania.

2. Zbyt małe zaangażowanie Scrum Mastera

Nie tylko zbyt duże, lecz również niewystarczające zaangażowanie Scrum Mastera może prowadzić do pojawienia się wielu błędów. Opisaliśmy poniżej najczęstsze z nich.

- 🟡 **Scrum Master niewystarczająco dobrze zna zasady Scrum.** Błąd ten z dużym prawdopodobieństwem doprowadzi do ich niewłaściwego wdrażania. Zaś praca Zespołu tylko z pozoru będzie pracą w Scrum.
- 🟡 **Scrum Master nie egzekwuje zasad Scrum.** Niewystarczająca obecność Scrum Mastera na co dzień sprawia, że nie chroni zespołu tak, jak powinien. Może to prowadzić do braku ochrony przed napływem zadań z zewnątrz albo do niewywiązywania się Zespołu Developerskiego z realizacji Celu Sprintu.
- 🟡 **Scrum Master nie pilnuje przestrzegania stałego rytmu Scrum.** Niedbałość w kwestii organizacji Wydarzeń Scrum może prowadzić do marnowania czasu. Rezultatem będą za długie lub źle prowadzone Wydarzenia - Sprint Planning, Sprint Retrospective czy Sprint Review. Błędem jest także przekładanie wydarzeń lub zmiana czasu ich trwania.
- 🟡 **Scrum Master nie reaguje na konflikty w Zespole.** Oczekiwanie, że konflikty w Zespole z czasem się rozwiążą, jest błędem Scrum Mastera. Konflikt nie zawsze jest zły, ale Scrum Master powinien być nie tylko świadom jego istnienia i aktualnego stanu, ale również zaangażować się w niego jako negocjator, a także potrafić wykorzystać konflikt do zmiany i usprawnienia działania Zespołu.
- 🟡 **Niewystarczająca obecność Scrum Mastera.** Problem pojawia się, gdy Scrum Master za mało czasu poświęca pracy z Zespołem i angażuje się na przykład w zadania specjalistyczne. Sprawia to, że za mało słucha i zadaje za mało pytań. To zaś jest kluczową umiejętnością Scrum Mastera. W rezultacie Scrum Master nie wie wystarczająco dobrze, jaka jest aktualna sytuacja i atmosfera panująca w Zespole, oraz zadowala się status quo.
- 🟡 **Scrum Master nie kwestionuje status quo.** Żeby Zespół Developerski, a także cały Scrum Team mógł się rozwijać, konieczne jest ciągłe kwestionowanie status quo. Jest to często działanie ryzykowne i potencjalnie konfliktogenne. Scrum Master powinien się go podejmować ze świadomością trudności, jakie może napotkać. Jednak nie ma czegoś takiego, jak „dojrzały Zespół Developerski, który już się nie rozwija”. Pozostawienie go samemu sobie szybko doprowadzi do znaczącego pogorszenia jego działania.

🟡 **Scrum Master nie dzieli się z Zespołem swoimi obserwacjami dotyczącymi jego funkcjonowania.**

Zatrzymywanie tej wiedzy dla siebie utrudnia, a nawet zupełnie zahamowuje rozwój Zespołu. Jest on co prawda zupełnie skupiony na swoich codziennych obowiązkach, jednak nie pracuje nad sposobem, w jaki współpracują ze sobą jego członkowie. Prowadzi to często do nawarstwiania się problemów i konfliktów.

Jakie statystyki i metryki powinien śledzić Scrum Master?

Po co Scrum Masterowi statystyki i metryki? Przede wszystkim po to, aby sprawdzać, czy jego metody pracy nad przewidywalnością rezultatów i poprawą efektywności zespołu są skuteczne. Lecz także po to, aby na bieżąco śledzić, jak jego działania wpływają na Zespół Developerski, czyli jak kształtują doświadczenie użytkownika pracownika (employees user experience, EX). Jakie zatem statystyki i metryki powinien śledzić Scrum Master?

1. Mierzenie rezultatów pracy Zespołu Developerskiego

Najczęściej używanymi wykresami, jakie powinien obserwować Scrum Master, są te opisujące tempo i przepływ wykonywania zadań. Są to Wykres Spalania, Burnup Chart, oraz Cumulative Flow Chart.

Służą one do mierzenia zarówno rozwoju produktu, jak i efektywności zespołu. Każdy z nich pozwala podejść do tych zagadnień nieco inaczej, dlatego warto używać ich łącznie. Mogą one służyć do oceny postępów w różnej skali. Zarówno tych dokonanych podczas Sprintu, jak i całego procesu powstawania produktu.

🟡 Wykres Spalania

Wykres Spalania, pokazuje Scrum Masterowi i Zespołowi Developerskiemu, ile pracy wykonano, a ile zostało jeszcze do wykonania. Na osi X przedstawiony jest czas jaki pozostał na wykonanie pracy. Natomiast na osi Y nanoszona jest ilość pracy pozostałej do wykonania, która została zaplanowana w Backlogu Sprintu lub Backlogu Produktu. Wykres Spalania pozwala określić Prędkość Zespołu Developerskiego. Jest to uśredniona ilość pracy wykonywanej w czasie jednego Sprintu. To proste narzędzie pozwala Scrum Masterowi nie tylko przekonać się, jak wydajnie pracuje zespół. Pomaga też odpowiedzieć na pytania:

- 🟡 Jaka część pracy została już ukończona?
- 🟡 Jak wiele zadań zostało do wykonania?
- 🟡 Jak długo będzie trwała praca nad Produktem?

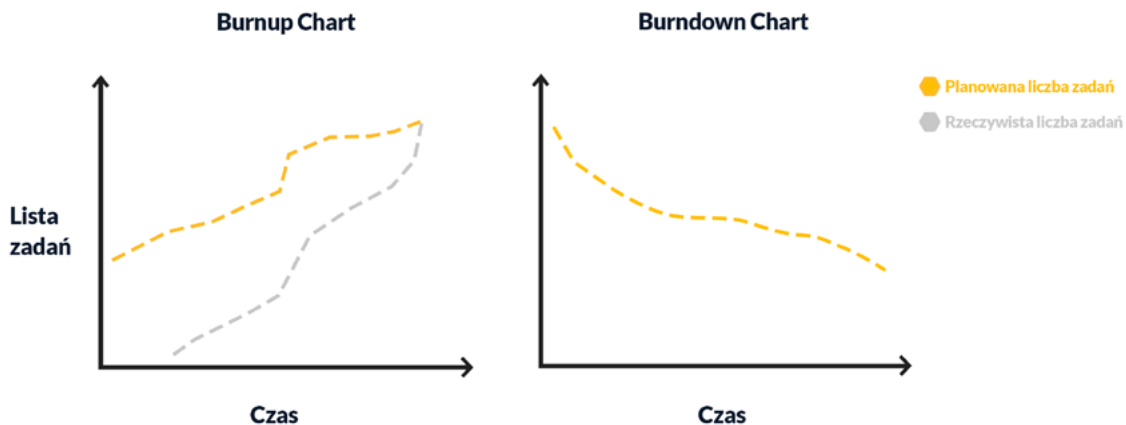


Scrum Master korzystając z Wykresu Spalania musi pamiętać o tym, że nie jest to jedyne narzędzie pozwalające na statystyczną ocenę postępów pracy zespołu. **Warto zapamiętać, że Wykres Spalania sprawdza się on najlepiej w przypadku projektów, w których zakres prac jest stały i znany.** Gorzej sprawdzi się w przypadku tworzenia bardzo innowacyjnych rozwiązań z nowym Klientem. Wtedy ilość pracy do wykonania w całym projekcie - czyli treść Backlogu Produktu - może znacząco zmieniać się w trakcie realizacji utrudniając korzystanie z Wykresu Spalania.

🔷 Burnup chart

Można powiedzieć, że Burnup Chart jest odwróceniem omówionego powyżej Wykresu Spalania (Burndown chart). Tutaj również na osi Y zaznaczona jest ilość pracy pozostałej do wykonania. Natomiast oś X pokazuje czas realizacji wyrażony w ilości Sprintów albo w datach. Scrum Master wykorzystuje jednak Burnup Chart w nieco innym celu. Pozwala on bowiem nie tylko ocenić postęp prac nad produktem i postępy Zespołu.

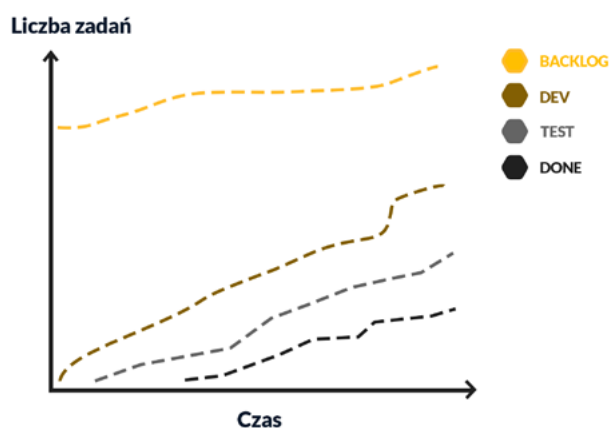
Metryka ta służy także ocenie tego, jak z upływem czasu zmienia się zakres prac w projekcie, dlatego sprawdza się dobrze w projektach o zmiennym zakresie. Burnup Chart jest także narzędziem planowania, które staje się z czasem coraz skuteczniejsze. Pozwala odpowiedzieć na pytanie, ile pracy szacunkowo wykona Zespół Developerski w następnym Sprincie.



🔷 Cumulative Flow Diagram

Trzeci rodzaj wykresu bardzo przydatnego w pracy Scrum Mastera z Zespołem Developerskim to Cumulative Flow Diagram. Pozwala on ocenić, jak stabilne jest tempo i wydajność pracy Zespołu Developerskiego. Układ jego osi jest taki samo jak w Burnup Chart, dlatego często określany jest jako jego bardziej złożona wersja. Cumulative Flow Diagram służy jednak nie tylko do określania ilości zadań wykonanych w danym okresie czasu. Uwzględnia także ilość zadań, które oczekują w kolejce na wykonanie.

Dzięki temu **Cumulative Flow Diagram pozwala na diagnozę tzw. „wąskich gardeł” (bottlenecks), czyli momentów procesu, które spowalniają powstawanie produktu.** Ta właśnie funkcja diagnostyczna czyni go jedną z najbardziej przydatnych metryk w rękach Scrum Mastera. Pozwala bowiem przeorganizować pracę w taki sposób, aby inaczej rozłożyć siły Zespołu Developerskiego i uniknąć przestojów.



2. Monitorowanie employees user experience Developerów

Regularne i skrupulatne prowadzenie oraz analiza statystyk jest bardzo ważną częścią pracy skutecznego Scrum Mastera. Musi on jednak pamiętać przede wszystkim o employees user experience Developerów, czyli sposobie w jaki odbierają oni pracę w Scrum Team. O tym decyduje jednak nie jakość metryk, lecz sposób, w jaki Scrum Master z nich korzysta.

Jeśli statystyki są prowadzone zgodnie z zasadami Scrum - są przejrzyste, jawne i zrozumiałe dla zainteresowanych Developerów - mogą być sposobem na motywowanie zespołu do bardziej wydajnej pracy lub nagradzania go za świetne rezultaty. **Statystyki mogą być jednak używane jako narzędzie służące do wywierania presji na Zespół Developerski.** Wtedy ich wskazania stają się generatorem oskarżeń i pretensji. Mogą przyczyniać się do pogarszania morali zespołu i psucia praktyki pracy zespołowej.

Drugim ważnym czynnikiem employees user experience Developerów, o jaki musi zadbać Scrum Master pracujący z narzędziami statystycznymi, jest sposób wykorzystania jego własnego czasu. Scrum Master musi mieć bowiem wystarczającą ilość czasu, żeby zająć się Zespołem Developerskim. Z tego powodu w przypadku dużego projektu, warto rozważyć włączenie do Scrum Team dodatkowej osoby. Będzie ona pełniła rolę menadżera projektu i zajmowała się prowadzeniem metryk. Dzięki temu odciąży Scrum Mastera - a także do pewnego stopnia Product Ownera - z wykonywania zadań które odciągają go od pracy z Zespołem Developerskim.

Współpraca Scrum Mastera z Product Ownerem

Product Owner stawia przed Zespołem Developerskim jasno określony Cel Produktu i wymaga postępów w jego realizacji. Scrum Master dba o jakość procesu jego powstawania: o dobrą atmosferę towarzyszącą pracy Zespołu, motywację i usuwanie przeszkód. Product Owner i Scrum Master nie są jednak dwoma niezależnymi od siebie siłami działającymi na Zespół Developerski. Choć sposoby działania każdego z nich są bardzo różne, ich interesy są zbieżne: chodzi przede wszystkim o efektywne działanie Scrum Team. Dlatego tak ważne są relacje pomiędzy Scrum Masterem i Product Ownerem, oraz ich efektywna współpraca. Większość zadań Product Ownera i Scrum Mastera koncentruje się wokół ich obowiązków związanych z pracą Zespołu Developerów. Jednak do obowiązków i zadań Scrum Mastera należy również wspieranie pracy Product Ownera.

1. Wspomaganie skutecznej komunikacji z Developerami

Skuteczna komunikacja Product Ownera z Zespołem Developerskim wymaga co najmniej dwóch mocnych podstaw: zrozumiałości i odpowiedniej siły oddziaływania. Scrum Master pomaga Product Ownerowi w ich wzmacnianiu.

Zrozumiałość Backlogu Produktu

Jednym z podstawowych sposobów, w jakie Scrum Master pomaga Product Ownerowi jest upewnienie się, że formułowane przez niego komunikaty są zrozumiałe dla Zespołu Developerskiego. Przegląda on wpisy w Backlogu Produktu i zadaje dodatkowe pytania, które pozwolą poprawić ich zrozumiałość.

Zwraca uwagę przede wszystkim na:

- 🟡 **jasność wpisów** - po to, żeby Deweloperzy dokładnie wiedzieli, w jakim celu rozwijają daną funkcjonalność,
- 🟡 **zwięzłość wpisów** - po to, żeby opisy planowanych funkcjonalności zawierały tylko niezbędne informacje i zapoznanie się z nimi zajmowało jak najmniej czasu.

W ten sposób Scrum Master zapobiega pojawieniu się rozbieżności pomiędzy Celem Produktu, takim, jakim wyobraża go sobie Product Owner, a tym, jak zrozumieli swoje zadanie członkowie Zespołu Developerskiego.

🟢 Siła oddziaływania Product Ownera

Scrum Master pomaga Product Ownerowi wzmacniać skuteczność przekazu i charyzmę. Pełni bowiem rolę coacha, z którym Product Owner może omówić problematyczne kwestie dotyczące Produktu i jego realizacji. Dlatego tak ważne są indywidualne spotkania, podczas których odbywają się pomiędzy nimi dyskusje. To dzięki nim Product Owner może doprecyzować wizję produktu oraz odpowiedzieć na pytania ze strony Scrum Mastera jeszcze zanim przedstawi wizję zespołowi. Scrum Master, udzielając informacji zwrotnej sprawia, że przekaz Product Ownera podczas spotkania z zespołem staje się mocniejszy i jaśniejszy. To niezbędne przygotowanie pomaga Product Ownerowi coraz lepiej komunikować Cel Produktu podczas Wydarzeń Scrum.

2. Wiedza wynikająca z doświadczenia

Scrum Master pomaga także Product Ownerowi w realistycznym planowaniu zadań przewidzianych dla Zespołu Developerów. Może się bowiem zdarzyć, że świetnie przygotowany Backlog Produktu nie będzie przystawał do sposobu działania organizacji, w której realizowany ma być Cel Produktu. Scrum Master będzie zatem wspierał Product Ownera swoją wiedzą wynikającą z doświadczenia. Czerpie ją z obserwacji niepowodzeń i trudności, które pojawiły się przy realizacji poprzednich projektów. Dzięki wiedzy empirycznej Scrum Master może przewidzieć trudności w wykonaniu zadań i realizacji Celu Produktu, które wynikają ze specyfiki organizacji, Zespołu albo jego specjalizacji.

3. Wprowadzanie Interesariuszy w Scrum

Na co dzień Scrum Master współpracuje głównie z Zespołem Developerskim. A czasem także z działem HR - szczególnie podczas budowania Zespołu oraz w tych rzadkich momentach, kiedy wymaga on rozbudowy lub zmiany. Codzienne obowiązki Scrum Mastera zwykle nie obejmują więc współpracy z Interesariuszami - te należą bowiem do zadań Product Ownera. Wyjątek dotyczy momentu rozpoczęcia współpracy z Interesariuszami, którzy nie są zaznajomieni z zasadami działania i rolami w Scrum. Wtedy właśnie Scrum Master współpracuje z Product Ownerem podczas spotkań ze wszystkimi osobami zaangażowanymi w powstanie Produktu. Wyjaśnia kto jest kim w Scrum Team oraz pomaga Product Ownerowi we wdrażaniu dobrych praktyk komunikacyjnych. Należą do nich na przykład aktywna obecność Interesariuszy podczas Sprint Review czy tworzenie dobrych User Stories.

Zespół Developerski

Zespół Developerski pracujący zgodnie z zasadami Scrum to samodzielna grupa specjalistów.

Nie korzysta ze wsparcia zewnętrznych ekspertów ani z pomocy podwykonawców. Co jednak decyduje o tym, że Zespół jest dobrze dobrany do wykonania postawionego przed nim Celu? Jakie obowiązki wchodzą w zakres zadań Zespołu Developerskiego - niezależnie od jego specjalizacji?

Cechy Zespołu Developerskiego

Zespół Developerski, aby działać skutecznie musi posiadać co najmniej trzy cechy: zdolność do samoorganizacji, dążenie do rozwoju, oraz interdyscyplinarność.

1. Samoorganizacja

Gdy mówimy o Scrum Team, którego częścią jest Zespół Developerski, używamy określenia „samozarządzanie”. Oznacza ono samodzielność na poziomie organizacji. Scrum Team jako całość decyduje nie tylko o tym, kto i jak wykona pracę, lecz także o tym, nad czym będzie pracować. W Scrum Team duża część zadań związanych z zarządzaniem spoczywa na barkach Product Ownera i Scrum Mastera. Dlatego w przypadku Zespołu Developerskiego ważniejsza od samozarządzania jest samoorganizacja.

Dotyczy ona planowania obowiązków, czyli samodzielnego decydowania o tym, kto będzie wykonywał określone zadania, kiedy i jak.

2. Dążenie do rozwoju

Kluczową cechą skutecznego Zespołu jest dążenie do rozwoju. **Sposób realizacji postawionych przed nim zadań powinien być ambitny.** Wynika to nie tylko z indywidualnych predyspozycji i nastawienia każdego z członków Zespołu Developerskiego. Do podnoszenia kompetencji i wysiłku skłania także atmosfera panująca w Zespole, która określa go jako całość.

3. Interdyscyplinarność

Interdyscyplinarność Zespołu oznacza, że jego członkowie razem wzięci powinni mieć wszystkie umiejętności niezbędne do stworzenia w każdym Sprincie wartościowego Przyrostu.

Oznacza to również, że każdy z członków Zespołu wykonuje zadania niezbędne w danym Sprincie. Każdy robi to, co niezbędne do osiągnięcia Celu. Nawet jeśli oznacza to podejmowanie nowych zadań wykraczających poza kompetencje Developera. Błędem jest sztywne trzymanie się swoich zawodowych kompetencji czy roli.

Skład Zespołu Developerskiego



Według Scrum Guide maksymalna ilość Developerów to osiem osób. Tak niewielki skład sprzyja komunikacji i otwartości, ponieważ członkowie Zespołu mają możliwość poznania się nawzajem.

Warto zapamiętać, że Zespół Developerski powinien składać się minimalnie z trzech osób.

Musi być bowiem wystarczająco duży, żeby w każdym Sprincie mógł wykonać postęp widoczny z biznesowego punktu widzenia.

Developerami w ramach Scrum nazywane są osoby o bardzo zróżnicowanych umiejętnościach

i zakresach obowiązków. W żadnym przypadku nazwa nie jest zarezerwowana dla osób zajmujących się programowaniem. W skład Zespołu wchodzić mogą zatem programiści i projektanci, badacze i analitycy, testerzy i naukowcy, a także inni specjaliści. Wśród Developerów nie obowiązuje hierarchia, dlatego nie posługują się oni tytułami zawodowymi czy naukowymi. Ważnym założeniem dotyczącym składu zespołu Developerskiego jest stwierdzenie, że stanowi on jedność. Dlatego też nie powinno się z niego wyodrębnić mniejszych zespołów pracujących nad innymi Celami.

Obowiązki Zespołu Developerskiego

Zakres obowiązków Zespołu Developerskiego można podzielić na trzy obszary. Są to: planowanie swoich zadań, praca nad produktem, oraz doskonalenie współpracy w Zespole.

1. Planowanie swoich zadań

Planowanie zadań to obowiązek, który wypełniają wszystkie Zespoły Developerskie działające według zasad Scrum. Polega ono na tworzeniu planu Sprintu i ujęciu go w Backlogu Sprintu. Najważniejsze, aby Zespół Developerski pracował nad nim wspólnie. W ten sposób każdy z Developerów będzie w stanie realistycznie określić ilość zadań do wykonania w danym Sprincie. Pozwoli to w dłuższym horyzoncie na utrzymywanie stałego tempa pracy Zespołu i bardziej trafne planowanie. Równie ważne jest trzymanie ręki na pulsie, czyli codzienne dostosowywanie planu do realiów. Jeśli pojawiają się problemy, może pojawić się potrzeba zmiany: przeorganizowania zadań, innego rozdzielenia pracy, albo rozmowy ze Scrum Masterem na temat wyłaniających się trudności.

2. Praca nad produktem

Formy pracy nad produktem mogą diametralnie się różnić w zależności od obszaru, w jakim działa dany Zespół Developerski. Najogólniej rzecz biorąc, celem do zrealizowania w każdym Sprincie jest stworzenie Przyrostu, czyli wartościowej biznesowo funkcji Produktu. Przydatne jest tutaj wypowiedzenie wprost i stosowanie następującej zasady:

Podejmując pracę nad Produktem, trzeba zostawić go w stanie nie tylko ulepszonym, ale nie mniej wykończonym niż wersja poprzednia.

Stosowanie tej zasady oznacza, że Zespół jako całość bierze odpowiedzialność za Przyrost. Jeśli Developer wykonuje zadania niedbale, co sprawia, że jakość Produktu ulega pogorszeniu, to ktoś inny będzie musiał wykonać pracę za niego. Z drugiej strony, jeśli któryś z Developerów trafia na błędy w Produkcie, powinien naprawić je samodzielnie lub przekazać informację o błędzie osobie, która może to zrobić.

3. Doskonalenie współpracy w Zespole

Praca nad sposobem działania Zespołu to nieustanne doskonalenie wydajności i skuteczności działań poszczególnych Developerów. Jest to jednak również - a może przede wszystkim - praca nad komunikacją pomiędzy Developerami. **Doskonalenie polega na wypracowywaniu rozwiązań umożliwiających sprawny i trafny podział zadań**, a także na ćwiczeniu umiejętności:

- 🟡 **krytykowania rozwiązań, a nie ludzi** - zmiana języka, którym opisujemy pracę prowadzi do zmiany nastawienia i polepszenia współpracy,
- 🟡 **zdystansowania się od swoich pomysłów** - dzięki niej tworzy się miejsce na poczucie humoru i bardziej szczerą informację zwrotną,
- 🟡 **budowania zaufania** - dzięki zaufaniu może pojawić się znacznie więcej innowacyjnych pomysłów proponowanych przez Developerów bez obaw o negatywną reakcję otoczenia.

Doskonalenie współpracy w Zespole dokonuje się dzięki bieżącej refleksji nad sposobem działania Zespołu i udzielaniu informacji zwrotnych podczas Wydarzeń Scrum.

Najczęstsze błędy popełniane przez Developerów

Wiele błędów najczęściej popełnianych przez Developerów pracujących w Scrum ma swoje źródło w ich podejściu do pracy zespołowej. Z jednej strony jest to źle rozumiana niezależność i obrona swoich pomysłów wbrew interesom Zespołu. Z drugiej strony - zdawanie się na innych i brak samodzielności. Kolejnym źródłem problemów może stać się opaczne rozumienie zespołowej odpowiedzialności.

1. Nadmierne przywiązanie do swoich pomysłów

Do codziennych obowiązków Developerów należy znajdowanie innowacyjnych rozwiązań złożonych problemów. Wysiłek wkładany w opracowywanie rozwiązań może powodować, że nadmiernie przywiązują się do swoich pomysłów. To z kolei sprawia, że tracą z oczu Cel Produktu i poświęcają zbyt dużo czasu na rozwijanie pobocznych rozwiązań, które nie są użyteczne z biznesowego punktu widzenia, a także mniej chętnie poszukują alternatywnych rozwiązań, co zagraża zwinności Zespołu.

2. Praca na własny rachunek

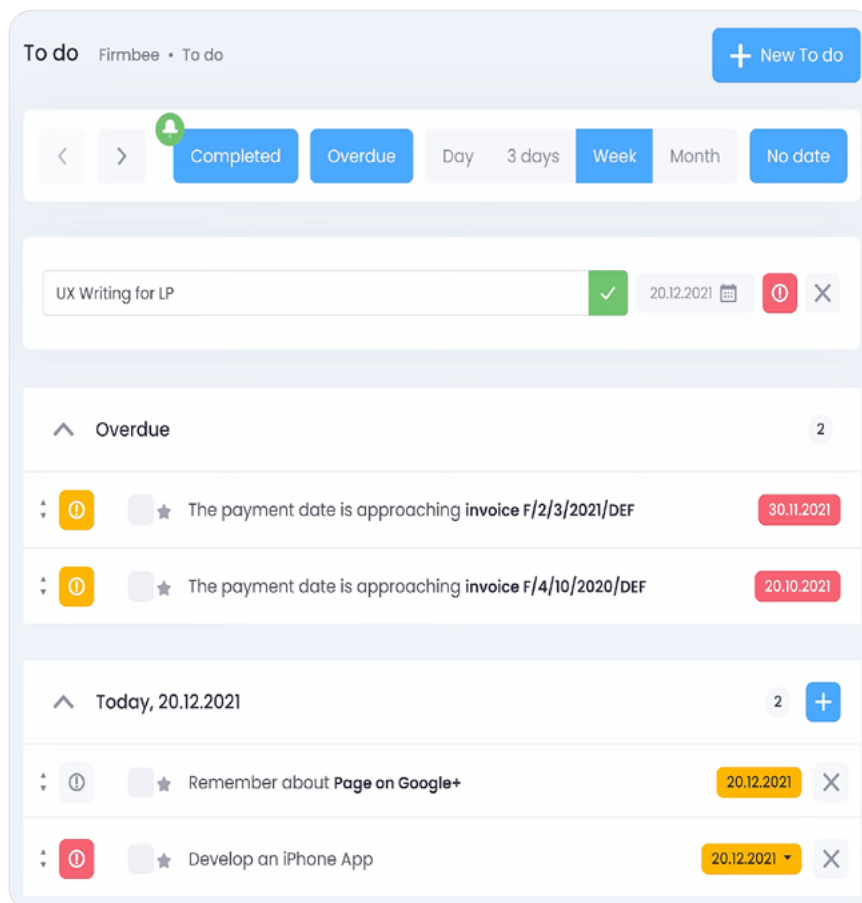
Jeśli Developer ma trudności ze zrozumieniem swojej roli w Zespole, będzie próbował wyodrębnić swoje zadania z Celu Sprintu. Co gorsza, będzie realizować je bez odnoszenia się do pracy reszty Zespołu. Problemem może stać się również samowolne dokonywanie zmian w Backlogu Sprintu. Tak właśnie źle rozumiana niezależność jednego z Developerów może wyplýwać z problemów komunikacyjnych. Nadmierne pragnienie niezależności może mieć źródło w braku uznania dla indywidualnych osiągnięć Developera. Pojawia się, gdy jego wkład w pracę wykonaną przez Zespół jest oceniany nieproporcjonalnie do włożonego wysiłku i trudności zadania. Praca na własny rachunek może być źródłem poważnych konfliktów w Zespole. Dlatego tak ważna jest reakcja Scrum Mastera i jak najszybsze rozwiązanie problemu, który leży u jego podstaw. Może się bowiem okazać, że błąd nie leży po stronie Developera, tylko nietrafnej oceny jego zaangażowania.

3. Wycofanie Developera

Problemem wynikającym z dwóch poprzednich - pracy na własny rachunek i nadmiernego przywiązania się do własnych pomysłów - może być problem braku komunikacji. Mimo, że wykonuje swoje zadania zgodnie z Backlogiem Sprintu, wycofuje się z życia Zespołu. W takiej sytuacji Scrum Master powinien zwrócić szczególną uwagę na wycofanego Developera. Docenić jego wkład w pracę Zespołu oraz zachęcić do przyjęcia proaktywnej postawy.

4. Niesamodzielnosc

Samoorganizacja to cecha dojrzałego, dobrze skomponowanego Zespołu Developerskiego. Oznacza ona, że mimo trudności, Developerzy nie polegają na innych osobach, które mówią im, w jaki sposób rozdzielać między sobą zadania, jak i kiedy je realizować. W celu lepszej organizacji swojej pracy, członkowie Zespołu mogą skorzystać z systemu do zarządzania projektami, a dokładniej z to-do listy, gdzie mogą zapisać swoje obowiązki, ustalić ich termin realizacji, nadawać statusy, czy ustalać priorytety.



Screen: System Firmbee - Tworzenie to-do listy

Samoorganizacja może jednak rodzić interpersonalne nieporozumienia. W takim przypadku konieczna jest stała obecność Scrum Mastera czuwającego nad podziałem zadań, które muszą zostać wykonane do realizacji Celu Sprintu. Wtedy właśnie pojawia się problem niesamodzielnosci Developerów. Z pomocą znowu powinien przyjść Scrum Master, zachęcając członków Zespołu Developerskiego do samostanowienia i brania odpowiedzialności za wykonywane zadania.

5. Ograniczanie obowiązków do zakresu kompetencji

Kolejnym problemem, z jakim muszą mierzyć się Developerzy, szczególnie w formującym się Zespole, jest **niechęć do wykonywania zadań innych niż należących do kluczowych kompetencji Developera**. Błąd ten może prowadzić do znacznego zmniejszenia efektywności Zespołu Developerskiego.

Nie we wszystkich Sprintach wykorzystywane są kluczowe kompetencje każdego z członków Zespołu. Dlatego muszą być oni otwarci na wykonywanie innych, pomocniczych lub porządkujących zadań, które są równie istotne z uwagi na Cel Sprintu.

6. Bałagan w Backlogu Sprintu

Jednym z takich zadań jest dbanie o porządek w Backlogu Sprintu. To zadanie kluczowe dla sprawnego działania Zespołu Developerskiego. Jednak często popełnianym błędem jest przerzucanie się odpowiedzialnością za jego prowadzenie między Developerami. Utrudnia to nie tylko pracę nad Celem Sprintu, lecz również rozwijanie się Zespołu i bieżące usprawnienie jego działania.

Skalowanie Scrum

W miarę jak organizacja rośnie, mogą pojawiać się coraz większe problemy z jej efektywnością. Są one spowodowane skomplikowaną strukturą wewnętrzną, utrudnionym podejmowaniem decyzji, czy wyznaczaniem kierunków działania. Dlatego firmy, które działają zwinnie na poziomie małych zespołów projektowych, często szukają możliwości ich skalowania.

W wielu organizacjach skalowanie Scrum nie jest potrzebne. Nawet jeśli działa w nich równocześnie wiele Scrum Teams, ich działanie nie musi być koordynowane. W organizacji działa wtedy po prostu wiele niezależnych Scrum Teams. Nie oznacza to jednak, że jest to Scrum wielozespołowy. **Potrzeba skalowania pojawia się dopiero, gdy większa część organizacji pracuje nad jednym produktem, natomiast działania wielu Scrum Teams wymagają synchronizacji.** Większość organizacji, które stosują zwinne metody zarządzania na dużą skalę wybiera model SAFe, czyli Scaled Agile Framework. Dziś przyjrzymy się jednak temu, jak skalowany jest Scrum. Według 15th State of Agile report, w 2021 roku drugim w kolejności wyborem był bowiem Scrum@Scale.

Scrum@Scale

W 1996 roku twórcy Scrum, Jeff Sutherland i Ken Schwaber, pracowali nad dużym projektem. W trakcie jego realizacji borykali się z problemem synchronizacji mniejszych zespołów pracujących w Scrum. Wymyślili wtedy sposób jego skalowania, który finalnie nazwali Scrum@Scale. Analogicznie do oficjalnego Przewodnika po Scrum powstał Przewodnik po Scrum@Scale, który definiuje ten sposób skalowania pracy jako:

„Framework w ramach, którego sieci Zespołów Scrumowych działające zgodnie z Przewodnikiem po Scrum mogą rozwiązywać złożone adaptacyjne problemy, kreatywnie dostarczając produkty o możliwie największej wartości.”

Podstawowym założeniem Scrum@Scale jest prostota i wydajność. Dlatego jego działanie oparte jest o bezskalową architekturę. Innymi słowy, wykorzystuje on Scrum do skalowania Scrum. W ten sposób Scrum Team złożony z pojedynczych osób pełniących rolę Product Ownera, Scrum Mastera czy Developera staje się Scrum of Scrums: zespołem złożonym z zespołów.

Scrum of Scrums

Scrum of Scrums jest to Scrum Team, w którym znajdziemy osoby pełniące zwyczajne role Scrum.

Jednak w związku z tym, że zadaniem Scrum of Scrums jest integracja wyników pracy kilku Scrum Teams, potrzebne są w nim dodatkowe role oraz grupy:

- **Product Owner Team** - grupa Product Ownerów spotykająca się w celu uzgodnienia priorytetów i stworzenia spójnej wizji produktu,
- **Chief Product Owner** - może być Product Ownerem jednego ze Scrum Teams, albo osobą zajmującą się wyłącznie Scrum of Scrums,
- **Scrum of Scrums Master** - jego zadaniem jest praca nad efektywnością Scrum of Scrums. Spotykają się oni na takich samych Wydarzeniach Scrumowych i używają zbliżonych Artefaktów.

Dalsze skalowanie i problemy Scrum@Scale

Bezskalowa architektura Scrum@Scale sprawia, że może on być skalowany nie tylko raz.

Jeśli organizacji potrzebna jest koordynacja zespołów na jeszcze większą skalę, może zostać utworzony Scrum of Scrums of Scrums.



Skalowanie Scrum może jednak rodzić problemy, a także zwielokrotnić te, które pojawiały się w podstawowych Scrum Teams. Dlatego zalecane jest dopracowanie szczegółów współpracy wewnątrz każdego Scrum Team przed rozpoczęciem wdrażania Scrum na szerszą skalę oraz podejmowanie decyzji o skalowaniu Scrum jedynie w przypadku doświadczonych zespołów, które dobrze znają i rozumieją sposób działania i Wartości Scrum.

Dla skutecznej organizacji pracy Scrum Team kluczowe są trzy Artefakty Scrum: to dwa Backlogi, czyli listy zadań oraz Przyrost, czyli potencjalnie gotowa do wydania wersja Produktu ulepszona w danym Sprincie. Są one zbiorczo nazywane Artefaktami, ponieważ są tworzone w jednym celu. Mianowicie po to, aby maksymalizować przejrzystość informacji dotyczących pracy nad Produktem. Dzięki dostępności Artefaktów Scrum, każda osoba należąca do Scrum Team lub Interesariusz może w dowolnym momencie uzyskać jasny obraz sytuacji. Dowie się z nich:

- 🟡 jaki Produkt i w jakim Celu jest tworzony,
- 🟡 jakie zadania zostały zaplanowane do wykonania,
- 🟡 nad jakimi zadaniami pracuje aktualnie Zespół Developerski,
- 🟡 jakie zadania zostały już ukończone,
- 🟡 jak wygląda aktualna wersja działającego Produktu.

Backlogi Scrum opisują Produkt zarówno od strony technicznej, jak i biznesowej. Techniczny opis Produktu tworzonego przez Scrum Team zawiera sposób działania Produktu, a także propozycje konkretnych rozwiązań wdrażanych przez Zespół Developerski. Opis biznesowy zawiera natomiast User Stories, które odpowiadają na pytania typu: do czego ma służyć Produkt, jakie funkcje ma pełnić Produkt, jakie oczekiwania Klienta ma spełniać Produkt. Opisują więc cząstkowe funkcjonalności Produktu z punktu widzenia Klienta.

1. Backlog Produktu

Backlog Produktu to lista zadań, nad którymi będzie pracować Scrum Team. Jest wyrażony w języku biznesowym, a jego zakres obejmuje cały czas trwania projektu. Prowadzenie i dostępność Backlogu Produktu jest kluczowa dla przejrzystości pracy Scrum Team. Dzięki temu dokumentowi Zespół Developerski wie, nad rozwiązaniem jakiego problemu biznesowego pracuje, oraz jakie są priorytety Klienta. Co więcej, Backlog Produktu stanowi drogowskaz, na którym można się wesprzeć w razie przytłoczenia mniejszymi zadaniami zacierającymi obraz całości projektu.

Backlog Produktu pozwala śledzić postępy Zespołu Developerskiego na drodze do realizacji Celu Produktu. Jest zarządzany przez Product Ownera i powinien być aktualizowany na bieżąco tak, aby w każdym momencie dawać jasny obraz pracy, jaka została do wykonania. Zadania, których czas wykonania jest najbliższy, są w Backlogu Produktu opisane najbardziej szczegółowo. Te z odległym terminem wykonania bądź opcjonalne mają formę ogólnego zarysu.

2. Backlog Sprintu

O Backlogu Sprintu możemy pomyśleć analogicznie do Backlogu Produktu. Zmienia się jednak sposób opisywania zadań oraz skala czasowa. O ile w Backlogu Produktu nacisk położony był na opis zadań z punktu widzenia Interesariuszy i języka biznesu, to Backlog Sprintu jest domeną Developerów. To oni są odpowiedzialni za prowadzenie i aktualizację Backlogu Sprintu. Stanowi opis pracy Zespołu, dlatego też wyrażony jest w języku technicznym.

Opisuje szczegółowe zadania i rozwiązania tak, jak planują ich wykonanie Developerzy. Backlog Sprintu operuje w skali czasowej odpowiadającej trwaniu jednego Sprintu, czyli zwykle od dwóch tygodni do miesiąca. Pozwala śledzić postępy Zespołu Developerskiego na drodze do realizacji Celu Sprintu.

3. Przyrost

Treść Backlogu Produktu opisuje krok po kroku sposób osiągnięcia Celu Produktu. Zawartość Backlogu Sprintu to opis zadań prowadzących do osiągnięcia Celu Sprintu. Natomiast **Przyrost jest sumą cząstkowych funkcjonalności Produktu zrealizowanych w danym Sprincie dodaną do zastanego stanu Produktu.**

Każdy nowy Przyrost nadbudowuje się na poprzednim. Dlatego wykonana praca powinna zostać dokładnie przetestowana. Testy pozwalają upewnić się, czy nowe rozwiązanie nie koliduje z tymi, które zostały stworzone wcześniej, ani nie zaburza ich działania. **Aby praca mogła stać się Przyrostem, musi zostać włączona w istniejący stan Produktu** i dać w rezultacie jego ulepszoną, działającą wersję. Innymi słowy, Przyrost to zbiór wykonanych w jednym Sprincie zadań, które składają się na nową, działającą wersję Produktu. Zaś jego biznesowe znaczenie opisuje Definicja Ukończenia. Zostaje ona wpisana do Backlogu Produktu podczas Planowania Sprintu.

Backlog Sprintu

Nowy Backlog Sprintu jest akceptowany przez Zespół Developerski podczas Planowania Sprintu (Sprint Planning). Od tej chwili staje się aktualnym zobowiązaniem Developerów, czyli listą nowych funkcjonalności, ulepszeń i modyfikacji Produktu, które powstaną w rozpoczynającym się Sprincie. Po rozpoczęciu Sprintu Backlog ten staje się natomiast obowiązującą kolejką, z której Developerzy wybierają zadania do wykonania.

Backlog Sprintu opisuje pracę Zespołu Developerskiego w czasie trwania jednego Sprintu.

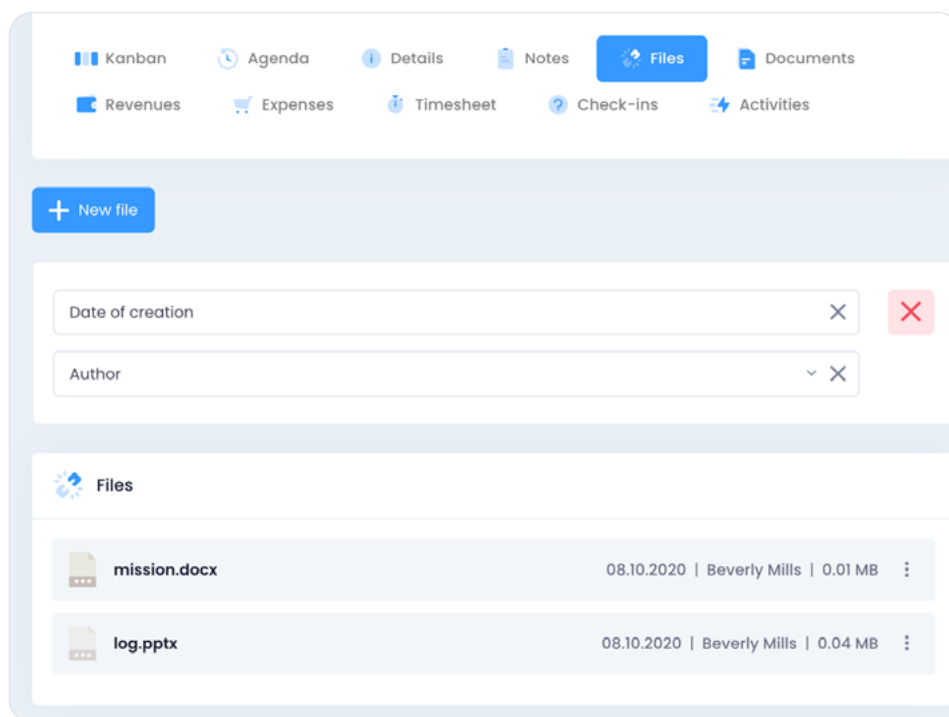
Dlatego też wyrażony jest w języku technicznym. Opisuje szczegółowe zadania i ich planowane rozwiązania. Składa się więc z listy zadań sporządzonej w sposób czytelny dla Developerów. Backlog Sprintu zwykle w niewielkim stopniu uwzględnia język wartości biznesowej Produktu, czyli sposób opisywania właściwy Backlogowi Produktu. Backlog Sprintu jest tworzony: na podstawie Backlogu Produktu, na czas trwania jednego Sprintu, podczas Wydarzenia Scrumowego zwanego Planowaniem Sprintu, przez cały Scrum Team - jednak kluczową rolę w jego powstawaniu odgrywa Zespół Developerski.

Jak powstaje Backlog Sprintu?

Podczas Planowania Sprintu Product Owner proponuje, w jaki sposób w najbliższym Sprincie zwiększyć wartość Produktu. Następnie cały Scrum Team współpracuje nad sformułowaniem Celu Sprintu, czyli wybiera z Backlogu Produktu funkcjonalność, którą będzie realizował. Cel Sprintu określa, w jaki sposób zostanie ulepszony bądź rozbudowany Produkt, aby mógł lepiej spełniać oczekiwania Klienta.

Kolejnym krokiem jest zastanowienie się, jaki zakres prac jest realny do wykonania w najbliższym Sprincie i w jaki sposób ta praca zostanie wykonana. Rezultaty tych ustaleń zostają spisane w formie technicznego opisu zadań do wykonania i ta lista właśnie staje się nowym Backlogiem Sprintu.

Nowo powstały Backlog Sprintu zostaje umieszczony w miejscu łatwo dostępnym dla wszystkich członków Zespołu Developerskiego. W przestrzeni fizycznej zwykle jest to tablica umieszczona w miejscu pracy. Natomiast w przestrzeni cyfrowej - współdzielony dokument, który mogą aktualizować wszyscy Developerzy.



Screen: System Firmbee - dokumenty projektowe

Chociaż właścicielem Backlogu Sprintu jest cały Scrum Team i wszyscy jego członkowie powinni dbać o jego aktualizację, zwykle zadaniem tym zajmuje się Scrum Master lub jeden z Developerów.

Co zawiera Backlog Sprintu?

Backlog Produktu nie określa sposobu wykonywania zadań. Dzieje się tak, ponieważ pozostawienie swobody wyboru postępowania jest bardzo ważne dla samoorganizacji Zespołu Developerskiego. Wolność wyboru kolejności i metod działania jest ważna także dla poczucia samodzielności i odpowiedzialności każdego z Developerów.

Temu samemu celowi służy potraktowanie Backlogu Sprintu jako nieuporządkowanej listy zadań do wykonania. Developer wybiera z niej zadanie, które akurat może lub ma ochotę wykonać (pull). Jest to przeciwieństwo tradycyjnego modelu push, w którym zadania są narzucane Zespołowi lub Developerowi w określonej z góry kolejności. Backlog Sprintu określa:

1. Cel Sprintu - czyli daje odpowiedź na pytanie, po co będą wykonywane zadania przewidziane w tym Sprincie.
2. Listę nowych funkcji Produktu i jego ulepszeń, które powstaną w tym Sprincie. Zawiera bowiem elementy Backlog Produktu wybrane do realizacji w tym Sprincie.
3. Listę zadań do wykonania - czyli techniczny opis tego, jak i przez kogo zostaną wykonane prace, których rezultatem będzie Przyrost.

Korzystanie z Backlogu Sprintu

Postęp prac spisanych w Backlogu Sprintu jest odzwierciedlany przez narzędzia metryczne - najczęściej przez Wykres Spalania. Dzięki takiej wizualizacji Zespół Developerski może łatwo zorientować się, czy prace nad realizacją Celu Sprintu idą zgodnie z przewidywaniami.

Może się zdarzyć, że w czasie trwania Sprintu okazuje się, że plan prac został zakresłony nierealistycznie. Innymi słowy, ilość zadań zapisanych w Backlogu Produktu, które są potrzebne do realizacji Celu Sprintu, jest zbyt duża lub zbyt mała. W takim przypadku Developerzy negocjują z Product Ownerem zmiany w bieżącym Backlogu Sprintu. Możliwe jest bowiem zmniejszenie ilości prac lub też dobranie dodatkowych zadań z Backlogu Produktu, bądź rozbudowanie planowanych już rozwiązań.



Zmiany nie mogą jednak dotyczyć Celu Sprintu.

Backlog Produktu

Backlog Produktu to największy z Artefaktów Scrum. **Odwierciedla stan pracy nad Produktem w odniesieniu do Celu Produktu.** Natomiast po zakończeniu pracy nad Produktem, jego Backlog staje się pełną listą zadań wykonanych przez Scrum Team w celu stworzenia Produktu. Nie zawiera on jednak szczegółowych rozwiązań technicznych. Backlog Produktu powstaje podczas spotkań Product Ownera z Interesariuszami. Wyłącznym właścicielem i osobą odpowiedzialną za Backlog Produktu jest Product Owner. Wpisy w Backlogu Produktu są wyrażane w języku biznesowym. Innymi słowy, opisują one wartość Produktu z punktu widzenia Interesariuszy.

Opisy zadań zawarte w Backlogu Produktu powinny mieć spójny i zrozumiały dla wszystkich charakter. Zawierają one funkcje i udoskonalenia Produktu ujęte najczęściej w formie User Stories. Tutaj wspomnimy tylko, że są to opisy częściowych funkcjonalności produktu odpowiadające na pytania o następujące kwestie: zakres modyfikacji Produktu, cel modyfikacji Produktu, typ użytkownika, dla którego ta modyfikacja jest wprowadzana.

Kształt Backlogu Produktu

Kolejność zadań zawartych w Backlogu Produktu zmienia się w miarę rozwijania Produktu.

Scrum Team pracując nad Produktem rozwija jego funkcjonalności. Sposób ich realizacji oraz napotymane problemy pozwalają przemyśleć i dookreślić kolejne rozwiązania. Ukończone rozwiązania definiują też i zmieniają porządek powstawania kolejnych, które będą z nich czerpać. Dlatego kolejność ich realizacji jest zmienna. Doskonalenie Backlogu Produktu ma na celu jego bieżącą aktualizację oraz przygotowywanie do wykonania kolejnych zadań. Z tego powodu ma ono charakter ciągły.

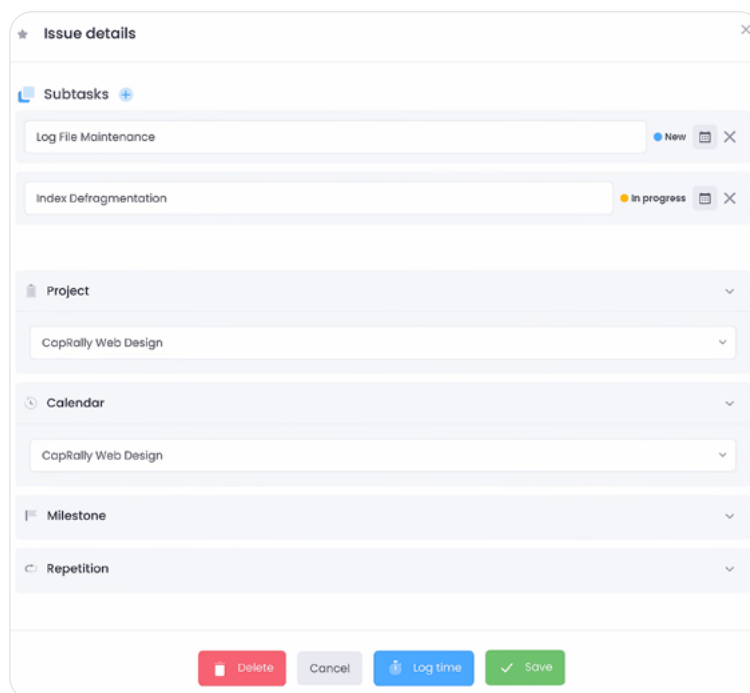
Zadania, których termin wykonania jest odległy, są zazwyczaj dużymi, ogólnymi całościami. Ich opis nie zawiera szczegółów, a jedynie zarys funkcjonalności, która powinna zostać zrealizowana. Można wśród nich także znaleźć zadania, które nigdy nie zostaną zrealizowane. Wpisy w Backlogu Produktu mogą przedstawiać alternatywne rozwiązania, a także pomysły Klienta, które mogą się zdezaktualizować, okazać się nieoptyczne lub z innego powodu nigdy nie wejść w fazę realizacji. Dlatego właśnie Backlog Produktu nazywany jest czasem żartobliwie „listą życzeń Klienta”.

Innym powodem zmian w kształcie Backlogu Produktu jest redefiniowanie rozwiązań. Czasem okazuje się bowiem, że pewien problem znalazł już rozwiązanie przy okazji tworzenia innej funkcjonalności produktu lub oczekiwana funkcjonalność stała się zbędna z powodu zmiany innych rozwiązań.

Jednym z podstawowych działań podczas doskonalenia Backlogu Produktu jest dzielenie zadań zawartych w Backlogu Produktu na części. Dzięki temu ogólny zarys funkcjonalności zostaje

przedstawiony w postaci mniejszych, bardziej dookreślonych i precyzyjnie zdefiniowanych jednostek.

W systemie do zarządzania projektami, każde zadanie można podzielić na podzadania, a także określić ich status, priorytet i czas realizacji.



[Screen: System Firmbee - Subtaski](#)

Zadania przeznaczone do bliższej realizacji stają się bardziej szczegółowe. Stają się także mniejsze, zawierają detale rozwiązań. Szczegóły wyłaniają się w trakcie rozwijania Produktu. Zaś dzięki znajomości aktualnego stanu Produktu oraz bieżących oczekiwań Interesariuszy, Product Owner uzupełnia najbliższe zadania o ich opis, kolejność oraz rozmiar. Oraz wybiera najlepiej opisane zadania do najbliższego Backlogu Sprintu.

Od Backlogu Produktu do Backlogu Sprintu

W toku pracy nad Produktem Product Owner modyfikuje i uszczegóławia Backlog Produktu we współpracy z Zespołem Developerskim. Kierując się sugestiami Product Ownera, podczas Planowania Sprintu Zespół wybiera z Backlogu Produktu funkcjonalności do zrealizowania. Przenosi je wówczas do Backlogu Sprintu oraz dzieli na zadania do wykonania. Zadania przenoszone do Backlogu Sprintu zostają opisane w języku technicznym, czyli w sposób najbardziej przydatny dla Developerów.

Rozmiar zadania jest ważną miarą z punktu widzenia Zespołu Developerskiego. Jego właściwe oszacowanie staje się szczególnie ważne w momencie wybierania User Stories z Backlogu Produktu do Backlogu Sprintu. Zespół Developerski uczy się z czasem prawidłowo oszacować czas i nakład pracy potrzebnej do realizacji określonej User Story. Jest on wyrażany w dniówkach, roboczogodzinach lub Story Points i pozwala oszacować wartość nazywaną Prędkością Zespołu.

Pielęgnacja Backlogu Produktu

Pielęgnacja Backlogu Produktu to jedno z podstawowych zadań Product Ownera. Do zadań związanych z pielęgnacją należy dopisywanie nowych User Stories w miarę ich formułowania i uszczegóławiania. Najważniejszym z zabiegów pielęgnacyjnych jest jednak dbanie o właściwą kolejność wpisów zamieszczonych w Backlogu Produktu, czyli o priorytetyzację zadań. Backlog Produktu to jeden z Artefaktów Scrum, który zawiera uporządkowaną według priorytetów listę prac potrzebnych do stworzenia Produktu. Innymi słowy, jest listą User Stories niezbędnych do realizacji Celu Produktu. Pielęgnacja Backlogu Produktu jest również znana pod nazwami: Priorytetyzacja Backlogu, Doskonalenie Backlogu, a także Skalowanie Backlogu.

1. Cel pielęgnacji Backlogu Produktu

Backlog Produktu jest zarządzany przez Product Ownera, do którego kluczowych umiejętności należy priorytetyzacja zadań w miarę zbliżania się terminu ich realizacji. **Celem pielęgnacji Backlogu Produktu jest bowiem upewnienie się, że na samej górze listy zadań do wykonania znajdują się funkcjonalności Produktu o największej wartości biznesowej**, czyli te najważniejsze z punktu widzenia Klienta oraz że są one opisane jasno i szczegółowo, tak, aby ich realizacja mogła rozpocząć się już w kolejnym Sprincie.



Backlog Produktu może być aktualizowany nawet codziennie, jeśli jest taka potrzeba. Product Owner może dopisywać do Backlogu Produktu nowe User Stories po rozmowie z Interesariuszami, Zespołem Developerskim, lub wyciągając wnioski i formułując na nowo User Stories już wpisane w Backlog Produktu. **Obowiązkowa aktualizacja Backlogu to jedno z zadań wykonywanych podczas Sprint Review.**

Zwykle podczas tego spotkania Scrum Team omawia nie tylko zadania przeznaczone do realizacji w następnym Sprincie. Uszczegóławia także wstępnie User Stories, które będą realizowane w następnych dwóch lub trzech Sprintach. Taki sposób działania pozwala na szersze spojrzenie na długofalowy kierunek rozwoju Scrum Team i jego działań. Dzięki temu może on myśleć o aktualnie wykonywanych zadaniach w perspektywie ich rozwinięcia w kolejnych Sprintach.

2. Błędy w pielęgnacji Backlogu Produktu

Jednym z najczęstszych problemów dotyczących pielęgnacji Backlogu Produktu jest dopuszczenie do niekontrolowanego poszerzania się jego zakresu. Podczas pracy nad Produktem pojawiają się bowiem spontanicznie różne dodatkowe funkcjonalności i zadania proponowane zarówno przez Interesariuszy jak i członków Scrum Team. Dlatego ograniczanie rozrastania się zakresu Backlogu Produktu (tzw. scope creep) to jedno z najważniejszych zadań wykonywanych przez Product Ownera. Do najczęstszych błędów, których poprawieniem zajmuje się Product Owner należą:

- 🟡 **Odbieganie od Celu Produktu** - dopisywanie do Backlogu Produktu zbyt wielu pomysłów wykraczających poza podstawowy Cel Produktu nie jest dobrą praktyką, ponieważ bardzo obniża to jego czytelność. Zdecydowanie lepiej sprawdza się zbieranie pomysłów na dodatkowe funkcjonalności w osobnym dokumencie.
- 🟡 **Zwielokrotnienie treści** - wpisywanie do Backlogu powtarzających się lub bardzo podobnych do siebie idei pochodzących od różnych Interesariuszy - przed dodaniem kolejnego wpisu do Backlogu, Product Owner powinien upewnić się, czy nowy wpis nie duplikuje któregoś z już istniejących.

- 🟡 **Brak szerszej perspektywy** - wpisy w Backlogu Produktu powinny być uporządkowane zgodnie z ich wartością w odniesieniu do Celu Produktu, jednak priorytetyzacja powinna uwzględniać kilka następnych Sprintów, tak aby zadania realizowane w danym Sprincie były płynnie powiązane zarówno z poprzedzającym, jak i Sprincie następującym bezpośrednio po nim.

Błędów tego typu nie sposób uniknąć. Jednak świadomość ich pojawiania się może sprawić, że Product Owner będzie ostrożniej dopisywał nowe User Stories do Backlogu Produktu. Musi on wypracować odpowiednią równowagę. Błędem jest bowiem również zbytnie okrajanie Backlogu i eliminacja wpisów, które zawierają podobne zadania różniące się jednak istotnymi kwestiami. Na przykład opisujące podobne funkcjonalności Produktu, które istotnie różnią się zastosowaniem.

3. Pielęgnacja Backlogu a metryki stosowane w Scrum

Backlog Produktu zawiera opis pracy, która pozostała do wykonania w całym projekcie. Jednak jedynie aktualny i regularnie pielęgnowany Backlog pozwala trafnie oszacować stosunek ilości wykonanej pracy do jej całości. Do obrazowania ilości zakończonej pracy, używany jest Wykres Spalania. Innym wskaźnikiem, często używanym do opisywania pracy Scrum Team jest Prędkość Zespołu. Mierzy się ją przez porównanie ilości wpisów w Backlogu Produktu przekształconych w Przyrost podczas jednego Sprintu.

User Stories

User Story jest to skrótowy opis nowej funkcjonalności Produktu lub jego udoskonalenia. Nie zawiera technicznego rozwiązania. Zwykle odpowiada na pytania opisujące funkcjonalność w kategoriach: Kto robi?, Co robi? I Dlaczego? User Story opisuje więc działanie Produktu w języku potocznym lub biznesowym. Choć bywa też używana do opisu zadań Scrum Team, które mają na celu poprawę funkcjonowania Zespołu.

User Story to najpopularniejszy sposób formułowania zadań realizowanych przez Scrum Team.

Pojedyncza User Story określa niewielką funkcjonalność Produktu. Opisuje bowiem najmniejszy sensowny do wyodrębnienia, cząstkowy Cel Produktu. Z tego powodu User Stories są bardzo krótkie.

User Stories są tworzone przez cały czas pracy nad Produktem. Powstają one nieustannie, od momentu podjęcia decyzji o rozpoczęciu pracy, aż po realizację Celu Produktu.

Tworzenie User Stories to zadanie Product Ownera. Na podstawie rozmowy z Klientem formułuje on odpowiedzi na pytania pozwalające stworzyć User Story i wpisuje je do Backlogu Produktu. Jednak User Stories odzwierciedlają nie tylko potrzeby Klienta.

Mapowanie historyjek użytkowników

1. Czynności :



2. Kroki :



3. Szczegóły :



Czyja to historia?

User Story określa potrzeby użytkownika Produktu tworzonego przez Scrum Team. Dlatego jest wyrażona w języku biznesowym. Innymi słowy, wskazuje na korzyści, jakie jej wprowadzenie przyniesie użytkownikowi produktu. W Backlogu Produktu mogą znaleźć się jednak także User Stories, które dotyczą potrzeb Zespołu Developerskiego, na przykład doskonalenia przepływu pracy między Developerami albo opisujące potrzeby Product Ownera, dotyczące na przykład przeprowadzenia prac porządkujących Backlog Produktu. W takich przypadkach Użytkownikiem w User Story staje się odpowiednio Developer i Product Owner.

User Stories można stworzyć odpowiadając na pytania 3W:

- 🟡 Kto robi? - **Who?** (is doing that)
- 🟡 Co robi? - **What?** (are they doing)
- 🟡 Po co? Dlaczego? Z jakiego powodu tego potrzebuje? - **Why?** (do they need it)

User Story zawiera się wtedy w formule:

Jako [typ użytkownika] chcę [co robić?], ponieważ [po co? dlaczego?].

Przykłady User Stories dotyczące funkcjonalności sklepu internetowego zapisane w takiej formie ilustruje poniższa tabelka:

	Kto? Jaki użytkownik?		Co? Co robić?		Po co? Dlaczego?
Jako	Klient	chcę	kupić magiczną różdżkę za pomocą jednego kliknięcia	ponieważ	potrzebuję jej natychmiast
	Developer		usprawnić wyświetlanie zdjęć magicznych różdek		inskrpcje są nieczytelne na ekranach smartfonów
	Product Owner		zmienić sposób przypinania User Stories do tablicy		spadają

Formuła ta pozwala nie tylko jasno sformułować User Story lecz także w stosunkowo prosty sposób przetłumaczyć język techniczny na język biznesowy i vice versa. Dzięki temu Cel oraz etap pracy nad Produktem jest zrozumiały zarówno dla Developerów, jak i dla Interesariuszy.

Jak korzystać z User Stories?

Stworzenie schematycznej User Story to dopiero początek pracy. Są one bowiem sygnalizatorami, punktami wyjścia do dyskusji nad problemami i ich rozwiązaniem. Dyskusja nad przyjmowanymi do realizacji User Stories ma miejsce podczas Planowania Sprintu. To na ich podstawie do Backlogu Sprintu są wpisywane techniczne zagadnienia rozwiązywane na bieżąco przez Zespół Developerski.

Zwykle w przestrzeni fizycznej User Stories są zapisywane na małych, kolorowych karteczkach przypinanych w miejscu pracy. Natomiast w przestrzeni cyfrowej najlepiej sprawdzają się cyfrowe whiteboardy, współdzielone przez Scrum Team. Zapisanie w ten sposób User Stories ma kilka zalet, ponieważ:

- 🟡 **podkreśla autonomię każdej User Story** - każda z nich ma osobne ramy i może zostać wykonana niezależnie od innych,
- 🟡 **akcentuje dynamikę User Stories** - kolejność ich realizacji jest renegocjowana przez Scrum Team, a aktualny porządek realizacji jest widoczny na tablicy dzięki fizycznemu ułożeniu karteczek z User Stories,
- 🟡 **pełni rolę przypominającą** - dzięki wizualnej reprezentacji User Stories Scrum Team ma w zasięgu wzroku drogowskaz, który przypomina o celu podczas tworzenia szczegółowych rozwiązań.

Nakład pracy potrzebny do realizacji danej User Story szacuje Zespół Developerski używając dniówek, roboczogodzin lub Story Points.

Kryteria akceptacji

User Story musi mieć określone kryteria akceptacji już w momencie przyjęcia jej do realizacji przez Zespół Developerski. Kryteria akceptacji determinują, w którym momencie praca nad User Story może zostać uznana za zakończoną. Dzięki temu zarówno Klient, jak i Developerzy wiedzą, w jaki sposób wykonywana przez nich praca przełoży się na wartość biznesową. Zwykle User Story zostaje uznana za zrealizowaną, gdy określony w niej użytkownik może wykonać opisaną czynność. Korzystając z przykładu powyżej, User Story o treści:

„Klient może kupić magiczną różdżkę za pomocą jednego kliknięcia”

Zostaje zrealizowana w momencie, gdy na stronie sklepu internetowego pojawia się działający przycisk „Kup teraz”, który wykorzystuje domyślne dane płatności i wysyłki dla zalogowanego użytkownika.

INVEST, czyli jak stworzyć dobre User Story

INVEST jest metodą tworzenia dobrych User Stories. Pozwala sprawdzić, czy mają odpowiednio sformułowaną treść, czy odnoszą się do wartości biznesowej Produktu. A także, czy ich wielkość i użyteczność zostały odpowiednio dobrane. INVEST to akronim stworzony przez Billa Wake w 2003 roku. Każda jego litera oznacza początek słowa, które charakteryzuje dobrą User Story. Według zasady INVEST każda User Story powinna być:

1. N jak Niezależna (Independent)

Pierwszą cechą dobrej User Story jest jej niezależność, czyli jej opis i charakterystyka powinny być zrozumiałe bez odniesienia do innych User Stories. Ale przede wszystkim to, że jej realizacja nie powinna być bezpośrednio uzależniona od realizacji innych User Stories. Nie będzie to oczywiście pełna niezależność. Tworzenia Produktu nie da się bowiem podzielić na zupełnie odrębne moduły. Jednak ważne by pamiętać o zachowaniu możliwie dużej samodzielności User Stories. Dzięki temu nawet, gdy jedna z nich nie wejdzie do fazy realizacji, albo zostanie znacząco zmodyfikowana, pozostałe User Stories nie będą musiały być modyfikowane. Co do zasady, User Story powinna więc stanowić odrębną i logiczną całość.

2. N jak Negocjowalna (Negotiable)

User Story powinna być negocjowalna. Oznacza to, że wyznacza ona Cel, a nie sposób dojścia do niego. Innymi słowy, określa oczekiwaną cechę Produktu, a nie techniczne rozwiązanie, które powinno zostać wdrożone. Negocjowanie User Story ma miejsce pomiędzy Product Ownerem a Zespołem Developerskim. Product Owner proponuje realizację pewnej funkcjonalności Produktu, czyli mówi „Co” zostanie zrobione. Natomiast do Developerów należy odpowiedź na pytanie „Jak”. Czyli wynegocjowanie konkretnych sposobów rozwiązania problemu postawionego w User Story.

3. W jak Wartościowa lub Wertykalna (Valuable/Vertical)

W skrótownicy INVEST litera V jest odczytywana dwa sposoby, jako: wartościowa (Valuable) lub wertykalna (Vertical). Oba rozwinięcia dają istotną charakterystykę dobrej User Story. Dlatego zdecydowaliśmy wyjaśnić, co oznacza każde z nich.

- Wartościowa User Story to taka, która jasno uzasadnia biznesowy sens wprowadzanej modyfikacji.

Innymi słowy, trafnie odpowiada na pytanie Po co? dana modyfikacja powinna zostać wprowadzona. I dlaczego jest to ważne z punktu widzenia Interesariuszy.

- Wertykalna User Story zawiera nową cechę Produktu widoczną dla Użytkownika, czyli nie koncentruje się na horyzontalnej „poprawie funkcjonowania” w wybranej warstwie Produktu - przeciwnie, dobudowuje do niego kolejne „piętro”. Innymi słowy, User Story opisuje, w jaki sposób zmodyfikować całość działania Produktu odpowiadając na pytanie Co konkretnie zostanie ulepszone? Oznacza to również, że każda funkcjonalność Produktu nabudowuje się na już istniejących rozwiązaniach. Rozwinięcie litery V jako Wertykalna zostało zaczerpnięte z metodyki Agile.

4. S jak Szacowalna (Estimable)

Dobra User Story powinna być szacowalna, czyli musi jasno określać zakres modyfikacji, które trzeba wprowadzić w produkcie, aby User Story została uznana za zrealizowaną.

Dzięki temu Zespół Developerski jest w stanie określić czas i nakłady pracy konieczne do jej wykonania. Zakres i trudność zadania są najczęściej szacowane w jednostkach nazywanych Story Points. Są one relatywne. A każdy Zespół Developerski wypracowuje w praktyce wartość Story Point bazując na wcześniejszych doświadczeniach.

5. M jak Mała (Small)

User Story przyjęta do realizacji przez Zespół Developerski powinna być krótka, czyli czas jej planowanej realizacji powinien być nie dłuższy niż czas trwania jednego Sprintu.

Jeśli Developerzy podczas Planowania Sprintu odkryją, że User Story zaproponowana przez Product Ownera jest zbyt długa, powinni podzielić ją na możliwie niezależne od siebie części.

6. T jak Testowalna (Testable)

Pod ostatnią literą akronimu INVEST kryje się słowo testowalna. **Testowalna oznacza, że modyfikacja Produktu opisana w User Story musi być konkretna i możliwa do sprawdzenia.** Innymi słowy, powinna istnieć możliwość weryfikacji, czy rozwiązanie zaimplementowane przez Developerów dostarczyło zakładaną wartość określonemu Interesariuszowi.

Najczęstsze błędy popełniane przy pisaniu User Story

User Story może być naprawdę świetnym narzędziem motywującym Zespół do proponowania nowych rozwiązań problemów przedstawionych z perspektywy użytkownika. Skupmy się jednak na tym, co może pójść nie tak podczas pisania i korzystania User Story.

1. Problemy z 3W

Właściwie napisana User Story odpowiada na pytania: Who? (Kto?), What? (Co?), Why? (Dlaczego?). Jednak odpowiedzi na każde z tych pytań mogą towarzyszyć problemy. Najbardziej pojawiającym się problemem są wątpliwości dotyczące tego, co powinno zmienić się w produkcie w odpowiedzi na potrzeby Klienta. Dlatego skoncentrujemy się na problemach dotyczących Who? i Why?

Who? i persona użytkownika

Jednym z najczęstszych błędów popełnianych przy tworzeniu User Stories jest brak wystarczająco precyzyjnej odpowiedzi na pytanie: dla kogo? Innymi słowy, **kto jest użytkownikiem, dla którego przeznaczona jest planowana zmiana?** Często nie wystarczy ogólna odpowiedź wskazująca na Klienta czy też End Usera jako adresata zmiany. Rozwiązaniem tego problemu jest wyobrażenie sobie odbiorcy jako konkretnej osoby. Persona jest to modelowe wyobrażenie docelowego Klienta. Innymi słowy, persona to reprezentacja osoby, która będzie korzystać z Produktu w określony sposób.

Po analizie User Story może okazać się, że opowiada ona historię różnych person równocześnie. Jeśli docelowych użytkowników jest wielu, warto zastanowić się nad rozbiciem User Story na mniejsze fragmenty, żeby nie wykonywać sprzecznych, wykluczających się, albo po prostu nieskutecznych działań.

Why?, czyli źle określony cel

Zdarza się, że źródłem problemów staje się ostatnia sekcja User Story. **Powinna ona określać biznesową wartość, jaką wnosi zmiana dokonywana podczas realizacji User Story.** Przykładem błędu, gdzie miejsce celu zajmuje opis dodatkowej funkcjonalności daje taka User Story:

Jako Klient chcę kupić magiczną różdżkę za pomocą jednego kliknięcia, ponieważ chcę w przyszłym tygodniu kupić latający dywan.

Powód kupienia magicznej różdżki zostaje tutaj zastąpiony dodaniem kolejnego punktu do listy zakupów potencjalnego Klienta. Dlatego przygotowując User Story warto pamiętać o powodach, dla jakich potrzebna jest nowa lub zmieniona funkcjonalność Produktu.

2. Problemy z 3C

Praca z User Stories jest często dzielona na trzy etapy nazywane 3C:

- **Card - Karta** - na której zapisywana jest User Story
- **Conversation - Konwersacja** - rozmowa wewnątrz Scrum Team na temat karty User Story
- **Confirmation - Konfirmacja** - czyli określenie kryteriów akceptacji potwierdzających wykonanie zadania

Karta

Karta, na jakiej zapisana jest User Story ma ograniczoną pojemność, dlatego najczęściej pojawiającymi się problemami są kłopoty z długością i objętością User Story. Z założenia powinna być ona zwięzła i naprawdę zamykać się w jednym, konkretnym zdaniu. Problem karty User Story ma bowiem dwa wymiary. Jeden to sposób jej sformułowania, który musi być zwięzły i zawierać minimalną ilość wyliczeń. Drugi to rzeczywista objętość User Story. Jednym ogólnym zdaniem można bowiem wyrazić ogromną ilość zadań, które nie mogą zostać zrealizowane w trakcie jednego Sprintu.

🔵 Konwersacja

Jednozdaniowe sformułowanie User Story stanowi punkt wyjścia do rozmowy z Zespołem Developerskim. Dlatego **błędem jest traktowanie jej jako opisu zadania do wykonania**. Zamyka to bowiem możliwości negocjacji i dyskusję nad różnymi możliwościami jej realizacji. User Story nie powinna być traktowana jak opis wymagań dotyczących nowej funkcjonalności produktu. User Story jest raczej zaproszeniem do rozpoczęcia konwersacji dotyczącej konkretnych rozwiązań technicznych, które z kolei doprowadzą do realizacji biznesowej wartości określonej przez User Story.

🔵 Konfirmacja

Podczas omawiania tematyki User Stories, pisaliśmy szczegółowo o kryteriach akceptacji.

Jednym z często popełnianych błędów jest brak lub niejasność kryteriów wykonania zadania.

W przypadku dobrze napisanej User Story, ona sama zawiera opis sytuacji, w której zostaje zrealizowana. Jej sprawdzianem jest to, że użytkownik jest w stanie skorzystać z nowej funkcjonalności stworzonej przez Zespół Developerski. Przydatnym narzędziem do sprawdzenia poprawności User Story jest opracowanie testu akceptacyjnego. Zwykle zostaje on zapisany po drugiej stronie karty zawierającej User Story.

Kryteria akceptacji User Story

Kryteria akceptacji User Story służą do sprawdzania, czy nowa funkcjonalność Produktu jest w pełni sprawna i satysfakcjonująca z punktu widzenia użytkownika. **Kryteria akceptacji spełniają następujące warunki: opisują nowe, ulepszone działanie Produktu z punktu widzenia użytkownika oraz są unikatowe dla każdej User Story.**

User Story ani kryteria jej akceptacji nie zostały zdefiniowane w oficjalnym Przewodniku po Scrum. Są one bowiem opcjonalnymi, choć bardzo często używanymi elementami pracy w Scrum. Dlatego na potrzeby e-booka zdefiniujemy kryteria akceptacji jako: *warunki, jakie musi spełniać udoskonalenie Produktu wykonane w danym Sprincie, aby mogło zostać zaakceptowane przez Użytkownika.*

1. Jak sformułować kryteria akceptacji?

Dobrze napisana User Story zawiera jasny opis sytuacji, w której zostaje zrealizowana, czyli zawiera swoje kryteria akceptacji. User Story jest jednak często bardzo krótkim zdaniem, które często można interpretować na różne sposoby.

🔵 Jasność i dostępność kryteriów akceptacji

Aby zapobiec niejasnościom, konieczne jest przeprowadzenie szczegółowej rozmowy z Klientem i ustalenie celu realizowanego rozwiązania. **Ostateczne sformułowanie kryteriów akceptacji należy jednak do Product Ownera.** Muszą one zostać spisane przed włączeniem User Story do Sprint Planning. Każdy z członków Scrum Team musi się z nim zapoznać i potwierdzić, że rozumie i zgadza się z przedstawionymi kryteriami akceptacji User Story. Zwykle kryteria akceptacji zostaną zapisane po drugiej stronie karty z User Story.

Właściwie sformułowane kryteria akceptacji sprawiają, że użytkownik może sprawdzić, czy User Story została zrealizowana po prostu używając produktu w opisany przez nią sposób. Mogą przyjmować formę checklisty, której punkty zostaną oznaczone jako wykonane przy testach Produktu na koniec Sprintu.

Sprawa jest prosta, jeśli działanie Produktu jest przejrzyste dla użytkownika. Jednak efektywne sprawdzenie złożonych produktów może okazać się trudne. Przykładem może być skomplikowane oprogramowanie lub usługi świadczone na szeroką skalę. Dlatego w większości przypadków przydatnym narzędziem do sprawdzenia poprawności User Story jest przygotowanie testu akceptacyjnego.

■ Test akceptacyjny

Jeśli test akceptacyjny zostaje opracowany, jest on zapisywany w miejscu kryteriów akceptacji, czyli na rewersie karty zawierającej User Story. Może on zostać przeprowadzony przez Scrum Team bądź przez zewnętrzny zespół QA. **Test akceptacyjny musi przede wszystkim zawierać jasne określenie, czy Produkt Nie Przeszedł/Przeszedł test.** Nie może zawierać określeń procentowych lub stanów pośrednich.



Warto pamiętać, że jeśli User Story zawiera więcej niż jedno kryterium akceptacji, każde z nich powinno być testowane niezależnie. Dzięki temu znacznie łatwiej określić, która funkcjonalność produktu wymaga poprawy lub dopracowania. Jest to szczególnie istotne, jeśli nowe funkcjonalności zawarte w User Story zazębiają się lub są od siebie zależne.

2. Kryteria akceptacji a Definicja Ukończenia

Definicja Ukończenia jest integralną częścią pracy w Scrum, która jest technicznym odpowiednikiem kryterium akceptacji. Spełniają one jednak odmienne role i nie należy ich ze sobą mylić.

Definicja Ukończenia to zamieszczony w Backlogu Produktu jasny i przejrzysty opis oczekiwanego stanu Produktu po ukończeniu Przyrostu. Opisuje on ulepszenia dokonane w ramach Przyrostu.

W odróżnieniu od kryterium akceptacji odpowiadającemu User Story, które opisuje funkcjonalność Produktu stworzoną podczas ostatniego Sprintu tak, jak jest ona postrzegana przez Klienta.

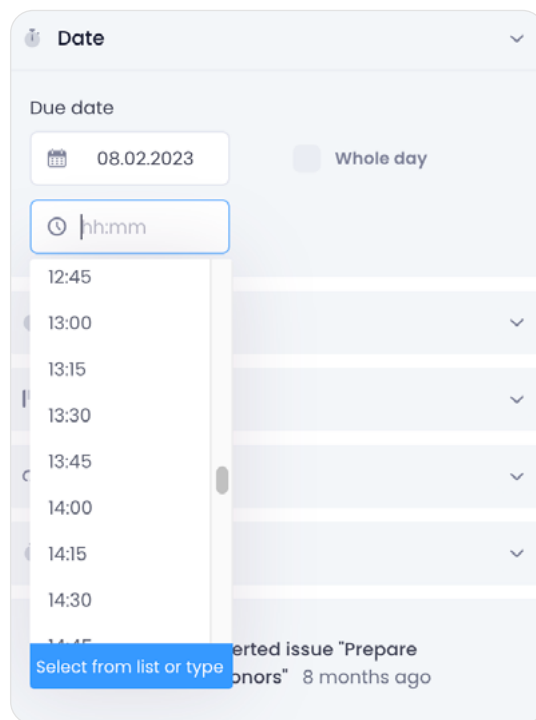
Przykładowo, weźmy User Story o treści: *Jako zalogowany klient sklepu internetowego chcę kupić magiczną różdżkę za pomocą jednego kliknięcia.*

Definicja Ukończenia dla powyższej User Story może zawierać następujące elementy: *stworzenie panelu logowania dla klientów sklepu, integracja systemu płatności, dodanie przycisku natychmiastowej płatności do szablonu strony produktu.*

Natomiast kryterium akceptacji przez Klienta będą następujące: *możliwość zalogowania się do sklepu, możliwość zdefiniowania domyślnej metody płatności, działający przycisk „Kup teraz” przy produkcie „magiczna różdżka”.*

Estymacja i Story Points w Scrum

Tworzenie estymacji w Scrum służy do szacowania stopnia skomplikowania i czasu potrzebnego do wykonania zadań. Oszacowania dokonuje wspólnie Scrum Team na podstawie swojego dotychczasowego doświadczenia. Dlatego też im bardziej doświadczony Scrum Team, tym trafniejsze są jego estymacje. Scrum Team uzgadnia planowany czas wykonania zadań podczas Sprint Planingu pamiętając jednak, że nie jest on traktowany jako zobowiązanie, lecz prognoza, której trafność mogą zaburzyć niespodziewane trudności w realizacji zadań.



Screen: System Firmbee - Ustalenie terminu realizacja zadania

Na każdym Sprint Planingu, Product Owner przedstawia zespołowi nowe User Stories. Product Owner wybiera je z Backlogu Produktu do realizacji w najbliższym Sprincie zaś członkowie Scrum Team wspólnie oceniają ilość pracy potrzebną do wykonania tej nowej porcji zadań. Zadanie to nazywane jest estymacją, wyceną wymagań, bądź szacowaniem.

Wydawać by się mogło, że najprostszym sposobem estymacji jest określenie czasu potrzebnego do wykonania zadania w godzinach lub dniach. Jednak praktyka i badania prowadzone już od lat '40 XX wieku dowodzą czegoś innego. Ludzie nie są w stanie trafnie oszacować czasu potrzebnego na realizację nawet bardzo dobrze zdefiniowanych zadań. Poza tym ilość godzin potrzebnych do realizacji zadania jest zależna od tego, kto to zadanie wykonuje, oraz co zostało - lub nie zostało - zrobione wcześniej. Dlatego też w Scrum stosuje się zazwyczaj jednostki zwane Story Points.

Znaczenie Story Points w Scrum

Każdy Zespół Developerski wypracowuje w praktyce wartość jednostki Story Point bazując na swoich wcześniejszych doświadczeniach oraz porównaniu wielkości poszczególnych zadań, czyli kierując się zasadą empiryzmu. Najczęściej podczas Sprint Planningu Scrum Master wybiera jedno lub kilka przykładowych już zrealizowanych User Stories, które stanowią punkt odniesienia dla określenia wartości User Stories wybranych do realizacji.

Zadaniom nie można więc przypisać wartości w Story Points niezależnie od ich kontekstu. Przykładowo, jeśli pierwszemu zadaniu przypisana zostanie wartość 10, kolejne zostanie oszacowane w odniesieniu do niego jako większe bądź mniejsze. W ten sposób **w ramach jednego projektu realizowanego przez Scrum Team, wszystkie zadania w Backlogu Produktu są odnoszone do siebie nawzajem.**

Oznacza to, że podobne zadania realizowane przez jeden Zespół Developerski otrzymają podobną liczbę punktów.

Story Points są jednostkami relatywnymi. Oznacza to, że:

1. Wartość Story Point odnosi się wyłącznie do zadań wykonywanych przez konkretny Scrum Team.

Story Points opisują prędkość realizacji zadań jednego zespołu. Innymi słowy, User Story oszacowana na 10 Story Points przez zespół A może zostać oszacowana na 50 Story Points przez zespół B.

Dzieje się tak, ponieważ, jak wspomnieliśmy ich wartość estymowana jest na podstawie i w odniesieniu do innych zadań wykonywanych przez ten zespół, a także jego doświadczenia z podobnymi zadaniami.

2. Wartość Story Points zrealizowanych w jednym Sprincie nie może być podstawą porównania wydajności dwóch Scrum Teams.

Aby uniknąć błędów przy zarządzaniu projektami realizowanymi w Scrum należy pamiętać, że Prędkość Zespołu Developerskiego wyrażona w Story Points zrealizowanych w jednym Sprincie nie może służyć do porównywania wydajności dwóch Zespołów. Mogły one bowiem wykonać w równoległych Sprintach identyczną pracę, którą jeden z Zespołów oszacował na 10, a drugi 50 Story Points.

Nie należy również zapominać, że estymacja zawiera wiele elementów niewiadomych i dokonywana jest na bazie niekompletnych danych. Z tego powodu przewidywania nawet bardzo doświadczonego Scrum Team mogą czasem mocno odbiegać od realnego nakładu pracy potrzebnego do realizacji User Story.

Relatywne techniki estymacji

Jakie są zatem najskuteczniejsze techniki estymacji stosowane w Scrum? Nie ma jednej uniwersalnej metody, która sprawdzi się w każdym Scrum Team. Wśród technik estymacji używanych w metodykach zwinnych stosowane są najczęściej:

- **Planning Poker.** Ta najpopularniejsza metoda relatywna polega na użyciu gry w karty do oszacowania ilości pracy potrzebnej do wykonania zadania.
- **Team Estimation Game.** Polega on na przypisaniu User Stories przeznaczonych do realizacji w danym Sprincie odpowiednich wartości liczbowych wybranych z ciągu Fibonacciego.

Natomiast z reguły nie stosuje się w Scrum estymowania klasycznego (Absolute Estimation) używanego w tradycyjnym zarządzaniu projektami. Klasyczny sposób szacowania zadań polega bowiem na określeniu z góry osobomiesięcy, czasu trwania i kosztu całego projektu. Jest to proces długotrwały, trudny w realizacji, oraz wymagający udziału ekspertów, którzy niekoniecznie będą wykonywać zadania, których wartość oszacowali. Innymi słowy, jest on nie tylko żmudny, ale również wysoce nieefektywny.

Planning Poker

Planning Poker jest jedną z technik estymacji używanych w Scrum. Rozgrywa się go podczas Sprint Planningu. Graczami są członkowie Zespołu Developerskiego. Każdy z nich równocześnie wyklada na stół kartę z ilością Story Points, na jaką szacuje opisane przez Product Ownera zadanie. Jakie są zalety i wady Planning Pokera?

Planning Poker, nazywany także Scrum Poker albo Pointing Poker to relatywna technika szacowania ilości pracy potrzebnej do wykonania konkretnego zadania. Stworzył ją w 2022 roku James Grenning. Chciał w ten sposób rozwiązać problem niekończących się sporów w Scrum Team dotyczących oceny trudności zadań stawianych przed Developerami.

1. Jak grać w Planning Pokera?

Celem Planning Pokera jest estymacja trudności i pracochłonności każdej z User Stories wybranych do wykonania w danym Sprincie. Zasady gry w Planning Pokera są proste. Najpierw należy jednak przygotować niezbędne akcesoria.

2. Akcesoria do gry

Akcesoria do gry w Planning Pokera to: jedna talia kart z User Stories - przygotowywana osobno do każdej rozgrywki oraz talie kart z wartościami punktowymi - po jednej talii dla każdego z Developerów, używane wielokrotnie. Karty z punktami zazwyczaj zawierają wartości odpowiadające ciągowi Fibonacciego, czyli sekwencję 0, 1, 3, 5, 8, 13, 20, 40 i 100. Zdarza się również, że są oznaczone kolejnymi potęgami liczby 2, czyli 2, 4, 8, 16, 32, i tak dalej. Dlaczego nie są to kolejne liczby? Ponieważ w Planning Poker chodzi o wyraźne pokazanie różnic pomiędzy trudnością wykonywanych zadań. Zaś zbyt małe różnice pomiędzy wartością kart zaciemniały by obraz estymacji. **Liczby zwykle wyrażają ilość Story Points. Mogą to być jednak także inne jednostki miary używane przez Scrum Team.**

3. Zasady Planning Pokera

Fazy przebiegu Planning Pokera prezentuje następująca tabela:

1. przedstawienie User Story	
2. dyskusja	Fazy 2. oraz 3. są powtarzane aż do momentu osiągnięcia konsensusu
3. rozgrywka	
4. konsensus	
5. przejście do kolejnej User Story	

Gra w Planning Pokera ma miejsce podczas Sprint Planingu. Product Owner trzyma w ręce karty User Stories. Natomiast każdy z Developerów otrzymuje talię kart z punktami. Moderatorem jest Product Owner. Rozpoczyna on grę od przedstawienia jednej User Story pozostałym członkom Scrum Team. Jeśli pojawiają się pytania, powinny zostać zadane od razu po przedstawieniu User Story.

Kolejnym krokiem jest rozpoczęcie dyskusji na temat realizacji User Story. W rozmowie bierze udział cały Scrum Team, jednak głównymi uczestnikami są Developerzy. Dyskusja dotyczy między innymi zagadnień takich jak:

- techniczna strona wykonania zadania,
- umiejętności poszczególnych Developerów, które będą niezbędne do jego realizacji,
- sposobów radzenia sobie ze spodziewanymi trudnościami,
- dodatkowych zadań wiążących się z realizacją User Story.

Gdy Developerzy ustalą odpowiedzi na najważniejsze pytania, każdy z nich samodzielnie wybiera jedną z kart ze swojej talii. Kładzie na stół kartę, której wartość jego zdaniem najlepiej odzwierciedla stopień skomplikowania User Story. Kolejny krok zależy od tego, jakie karty zostały wyłożone.

● **Jeśli Developerzy położyli na stole karty o różnych wartościach, wracają do dyskusji.**

A następnie zabierają ze stołu karty i ponownie szacują wartość User Story. Sytuacja się powtarza, a Developerzy wykładają karty ponownie aż do uzyskania konsensusu.

● **Jeśli Developerzy zgodnie ocenili User Story, przechodzą do kolejnej tury Planning Pokera.**

Product Owner prezentuje kolejną User Story i procedura powtarza się aż do wyczerpania puli User Stories planowanych do realizacji w bieżącym Sprincie.

4. Zalety i wady Planning Pokera

Zaletą Planning Pokera jest bez wątpienia ustandaryzowane pracy z User Stories. Zespół Developerski trzyma w ręce gotowy zestaw kart służący do szacowania ilości pracy. Dzięki temu w każdym Sprincie wartości pozostają niezmiennie, a Zespół uczy się estymacji w określonych jednostkach.

Ważną zaletą jest także równy udział wszystkich Developerów w szacowaniu skomplikowania zadania. Nawet osoby, które nie biorą bezpośrednio udziału w jego wykonaniu mogą wnieść istotny wkład w dyskusję. Na przykład zwracając uwagę na problemy, które nie przyszły do głowy Developerom nastawionym na techniczne aspekty wykonania zadania.

Kolejną korzyścią płynącą z wykorzystania Planning Pokera jest możliwość ustawienia limitów czasowych na dyskusję, a także - w razie potrzeby - ograniczenia ilości rund rozgrywanych do każdej User Story. Czas potrzebny do osiągnięcia konsensusu jest jednak także jedną z najczęściej wymienianych wad Planning Pokera. Jeśli jeden lub kilku Developerów nie chce zgodzić się z estymacją pozostałych, gra może potencjalnie ciągnąć się w nieskończoność.

Team Estimation Game

Team Estimation Game to technika estymacji używana podczas Sprint Planingu w Scrum. Nazywana jest również Swimlanes Estimation. To drugie określenie zostało ukute spontanicznie, a wzięło się z obserwacji ułożonych na stole kart do estymacji, które wyglądają właśnie jak tory pływackie w sportowym basenie.

Team Estimation Game nieustannie zyskuje na popularności, ponieważ Zespoły Developerskie tworzą estymacje z jej użyciem mniej więcej 3 razy szybciej niż używając Planning Pokera. Na czym więc polega Team Estimation Game?

1. Zasady Team Estimation Game

Akcesoria potrzebne do gry w Team Estimation Game to: talia kart z User Stories - przygotowywana osobno do każdej rozgrywki oraz talia kart z wartościami punktowymi - mogą być używane wielokrotnie.

Karty z User Stories powinny być na początku ułożone w stos, w kolejności odpowiadającej wpisom do Backlogu Produktu. Dzięki temu w pierwszej kolejności zostaną oszacowane te, których wykonanie jest najpilniejsze. Karty z punktami zazwyczaj zawierają wartości odpowiadające ciągowi Fibonacciego. Jest to sekwencja następujących liczb: 0, 1, 3, 5, 8, 13, 20, 40 i 100. Mogą być także oznaczone kolejnymi potęgami liczby 2, czyli 2, 4, 8, 16, 32, i tak dalej.

Aby zagrać w Team Estimation Game, członkowie Scrum Team siadają dookoła stołu. Przebieg gry wygląda następująco:

- 🟡 **Grę rozpoczyna Product Owner.** Wyciąga on pierwszą kartę z talii User Story. Opowiada o jej zawartości, po czym kładzie ją na stole. Wyjaśnia pozostałym członkom Scrum Team, że po lewej stronie karty należy umieszczać User Story łatwiejsze, a po prawej - trudniejsze do realizacji. Jeśli któreś z nich mają taki sam stopień trudności, zostają umieszczone jedna pod drugą, poniżej karty znajdującej się na stole. Kolejny ruch wykonuje osoba siedząca obok, zgodnie z ruchem wskazówek zegara.
- 🟡 **Osoba wyciąga kartę z talii User Story.** Czyta jej treść, która jest następnie objaśniana przez Product Ownera. Osoba trzymająca kartę kładzie ją następnie na stole wybierając miejsce zgodnie z własnym zdaniem na temat trudności tej User Story. Następnie uzasadnia swój wybór, a pozostali członkowie Scrum Team mogą zadawać jej pytania. Nie mogą jednak kwestionować jej decyzji.
- 🟡 **Każda następna osoba ma do wyboru dwa ruchy:** powtórzyć punkt 2 lub przesunąć jedną z kart leżących na stole w miejsce, które uzna za najwłaściwsze. W przypadku wyboru drugiej opcji, należy również uzasadnić decyzję o zmianie estymacji. Krok trzeci jest powtarzany aż do momentu wyczerpania talii z User Stories.
- 🟡 **Końcowy etap rozmieszczania kart z User Stories to jedna** - lub kilka, w zależności od praktyki przyjętej przez Scrum Team - runda, w której każdy z graczy ma możliwość przesunięcia jednej z kart na stole w odpowiednie miejsce.
- 🟡 **Dopiero gdy wszystkie karty z User Stories mają już swoje finalne miejsca, Zespół Developerski przechodzi do przypisania ilości Story Points.** Nad każdym z pasów zostaje umieszczona karta z liczbą punktów. Pierwszej karcie z lewej Product Owner przyporządkowuje kartę z najmniejszą liczbę punktów. Natomiast zasada rozmieszczania kolejnych jest analogiczna do punktów 3 i 4. W ten sposób estymacja zostaje zakończona.

2. Team Estimation Game a Planning Poker

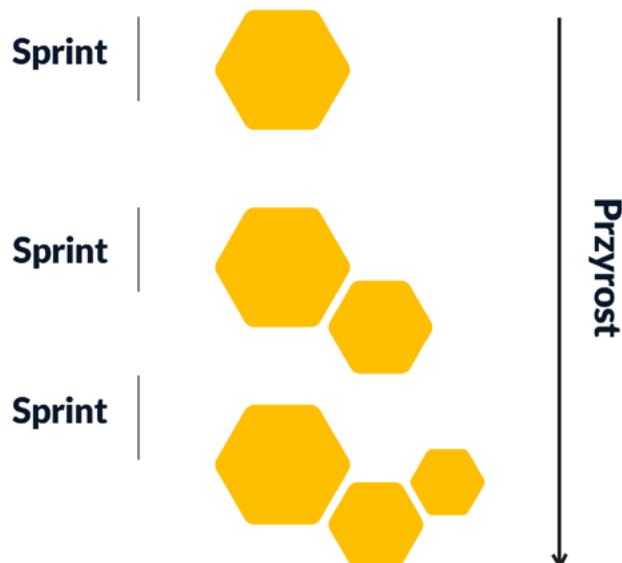
Team Estimation Game jest zwykle bardziej efektywnym narzędziem estymacji niż Planning Poker. Dzieje się tak, z powodu następujących różnic pomiędzy tymi technikami.

- 🟡 **Karta-stół.** W Team Estimation game obowiązuje znana z popularnych gier karcianych „zasada karta-stół”. Oznacza ona, że nie można cofnąć raz położonej karty. Dzięki temu, że User Story szacowana jest przez jedną osobę równocześnie wahanie się między szacunkami i liczba ich zmian jest znacząco ograniczona w stosunku do Planning Pokera.
- 🟡 **Wystarczająco trafna estymacja.** W Planning Pokerze przy każdej User Story powinien zostać osiągnięty pełny konsensus. Natomiast w Team Estimation Game decyzję podejmuje jedna osoba. Nawet jeśli jej estymacja jest nietrafna, z dużym prawdopodobieństwem zostanie skorygowana przez innego Developera. W ten sposób zwykle znacznie szybciej zostaje podana wystarczająco trafna estymacja.
- 🟡 **Wystarczająco długa dyskusja.** Dyskusje często nadmiernie się wydłużają podczas gry w Planning Pokera. Ich czas znacznie się skraca podczas Team Estimation Game, ponieważ koncentrują się na decyzji podjętej przez jednego z Developerów, a nie na charakterze każdej User Story.

Jedną z potencjalnych wad Team Estimation Game jest poczucie niesprawiedliwości. Jeśli Zespół Developerski jest liczniejszy niż ilość User Stories przewidzianych do realizacji w danym Sprincie, niektórzy Developerzy mogą czuć się pominięci.

Przyrost w Scrum

Przyrost jest wykonaną w czasie jednego Sprintu najnowszą, gotową i udoskonaloną wersją Produktu, która ma wartość biznesową i może potencjalnie zostać wydana. Ważną częścią jego definicji jest stwierdzenie, że Przyrost jest sumą wcześniejszego stanu Produktu i tego, co zostało do niego dodane w bieżącym Sprincie. Nie określa więc wyłącznie nowych funkcjonalności dodanych do Produktu, lecz także - lub przede wszystkim - ich integrację z jego zastaną wersją oraz wniesione ulepszenia i poprawki.



Przyrost a Cel Produktu i Cel Sprintu

W każdym Sprincie powinien zostać wytworzony co najmniej jeden nowy Przyrost. Za jego realizację jest odpowiedzialny cały Scrum Team. Jak zatem ma się Przyrost do Celu Sprintu? Można powiedzieć, że **Cel Sprintu jest odpowiedzią na pytanie, po co budowany jest Przyrost**. Należy jednak pamiętać, że w jednym Sprincie może zostać zrealizowanych kilka Przyrostów, których suma będzie pełną realizacją Celu Sprintu.

Przykładowym Celem Sprintu może być zatem zadanie o następującej treści: *Stworzenie funkcjonalności „koszyk” w sklepie internetowym z akcesoriami magicznymi*.

Natomiast składowymi Przyrostu będą funkcjonalności sklepu takie jak: *dodanie przedmiotu do koszyka, usuwanie przedmiotu z koszyka, wyświetlanie sumy należności za zakupy, obliczanie kosztów dostawy dla przedmiotów w koszyku, itp.*

Każdy Przyrost przybliża moment realizacji Celu Produktu i z myślą o nim powinien być realizowany. Dlatego też Przyrostem mogą być ulepszenia takie jak: *uporządkowanie bazy danych klientów, usprawnienie przepływu pracy pomiędzy Developerami*. Dzieje się tak, ponieważ chociaż nie są one bezpośrednio związane z poprawą obecnego stanu Produktu, mają ogromny wpływ na lepszą i bardziej efektywną realizację Celu Produktu.

Kiedy wykonana praca staje się Przyrostem?

Miarą udanego Przyrostu jest to, czy jest użyteczny. Innymi słowy, **wykonana praca staje się Przyrostem dopiero w momencie, w którym spełnia Definicję Ukończenia**. Definicja ta określa jednoznacznie, jakie zmiany i ulepszenia pojawiły się w Produkcie. Na podstawie Definicji Ukończenia można przetestować Przyrost by ocenić, czy rzeczywiście wprowadza ulepszenia do Produktu i czy wnosi deklarowaną wartość biznesową.

Jeśli wykonana praca nie jest zgodna z Definicją Ukończenia, powraca do Backlogu Produktu. Kolejne kroki obejmują następujące opcje:

- 🟡 praca może zostać ukończona w kolejnym Sprincie,
- 🟡 praca może zostać anulowana,
- 🟡 praca może zostać przesunięta w dół Backlogu Produktu do wykonania w przyszłości - po usunięciu przeszkód stojących na drodze do jej realizacji.

Jeśli Przyrost został w danym Sprincie wytworzony z sukcesem, jest on przedstawiany i oceniany podczas Sprint Review. W razie potrzeby może jednak zostać dostarczony Interesariuszom nawet przed ukończeniem Sprintu. Ostateczną decyzję o jego wydaniu podejmuje Product Owner.

Wydarzenia w Scrum

Wydarzenia Scrum to spotkania lub działania organizacyjne podejmowane przez Scrum Team.

Ich wspólną cechą jest to, że służą pielęgnowaniu przejrzystości: stanowią okazję do rozmowy nad planowaniem, metodami pracy, a także refleksji nad osiągnięciami Scrum Team. Każde z Wydarzeń Scrum ma jednak odrębną charakterystykę, zdefiniowany cel i czas trwania. Wydarzenia Scrum, nazywane nieoficjalnie obrzędami Scrum („Scrum ceremonies”) są to: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective. Każde z nich pełni w Scrum ściśle określoną rolę.

1. Sprint

Sprinty wyznaczają cykl pracy Scrum Team, dlatego każdy Sprint ma taką samą długość.

Następują po sobie nieprzerwanie i trwają od pierwszego po ostatni dzień pracy nad Produktem.

Czas trwania jednego Sprintu jest ustalany przez Scrum Team. Zależy on od potrzeb i możliwości zespołu i organizacji, a także od charakteru Produktu, nad którym pracuje Scrum Team. Optymalny czas trwania Sprintu określają zasady empiryzmu, zgodnie z którymi powinien on być:

- **wystarczająco długi** - tak, żeby Zespół Developerski zdążył znacząco udoskonalić Produkt,
- **wystarczająco krótki** - tak, żeby ograniczyć ryzyko, niepotrzebny wzrost złożoności oraz rozmywanie się Celu.

Z dobrych praktyk wynika, że jeden Sprint powinien trwać od jednego do czterech tygodni. Krótsze Sprinty pomagają szybciej diagnozować problemy i ograniczenia, a także oszacować ilość pracy do wykonania w danym Sprincie. Dłuższe pozwalają natomiast dowartościować wyniki pracy dzięki prezentacji podczas Sprint Review większych gotowych całości. W czasie każdego Sprintu odbywają się wszystkie mniejsze Wydarzenia Scrumowe, które szczegółowo opisujemy poniżej.

2. Sprint Planning

Sprint Planning jest spotkaniem Scrum Team, podczas którego planowana jest praca do wykonania

w kolejnym Sprincie. Dlatego odbywa się w pierwszym dniu nowego Sprintu. W Sprint Planningu bierze udział cały Scrum Team, zaś wydarzenie trwa maksymalnie osiem godzin. Jego przebieg można streścić odpowiadając na trzy pytania: jaki będzie Cel Sprintu, co zostanie zrobione, w jaki sposób zostanie zrobione.

3. Daily Scrum

Daily Scrum jest krótkim wydarzeniem, w którym biorą udział sami Developerzy. To codzienne spotkanie trwa nie więcej niż 15 minut. Aby ograniczyć złożoność, Daily Scrum odbywa się zawsze w tym samym miejscu i czasie. Głównym tematem Daily Scrum jest planowanie, dlatego rozmowa na temat aktualnego postępu prac nad Produktem powinna ograniczyć się do ich odniesienia do Celu Sprintu.

Zespół Developerski decyduje wspólnie, jakie jest najważniejsze zadanie, które powinien wykonać każdy z Developerów do następnego Daily Scrum.

4. Sprint Review

Sprint Review, wraz ze Sprint Retrospective, stanowią wydarzenia podsumowujące i mają miejsce ostatniego dnia kończącego się Sprintu. Celem Sprint Review jest podsumowanie Sprintu pod kątem ukończonych zadań przybliżających realizację Celu Produktu. Zwykle zawiera też prezentację aktualnego stanu Produktu i omówienie jego nowych zrealizowanych funkcjonalności. Każdy Sprint Review powinien kończyć się aktualizacją Backlogu Produktu. W Sprint Review jest wydarzeniem otwartym: bierze w nim udział cały Scrum Team, a także Interesariusze, którzy wyrażą taką chęć. Wydarzenie jest zwykle moderowane przez Scrum Mastera.

5. Sprint Retrospective

Sprint Retrospective, choć również jest wydarzeniem podsumowującym Sprint, jego charakter jest inny niż Sprint Review. Koncentruje się bowiem na sposobach pracy Scrum Team, a nie na rozwijanym przez niego Produkcie. Aby stworzyć bezpieczną przestrzeń do komunikacji, **Sprint Retrospective jest wydarzeniem zamkniętym. Mogą w nim uczestniczyć tylko członkowie Scrum Team.** Sprint Retrospective moderuje Scrum Master. Sprint Retrospective trwa maksymalnie trzy godziny i jest ostatnim wydarzeniem kończącego się Sprintu. Jego cele są następujące: wyciągnięcie wniosków z obecnych metod współpracy, wskazanie obszarów, które wymagają poprawy oraz propozycje i dyskusja nad wprowadzeniem ulepszeń.

The screenshot displays the 'Issue details' form in the Firmbee system. The form is titled 'Issue details' with a star icon and a close button. It features several sections for configuring an event:

- Event Type:** Radio buttons for 'event', 'meeting' (selected), and 'task'. There are also buttons for 'Generate To do' and 'New'.
- Title:** A text input field containing 'Sprint planning'.
- Guiding Questions:** A text area with the following questions: 'What will be the Goal of the Sprint?', 'What will be done?', and 'How will it be done?'. Below this, it says 'Characters left: 9905'.
- Room:** A text input field containing 'Room nr 5'.
- Online Meeting:** A section with a blue icon and a button 'Add online meeting'.
- Assigned user:** A section with a blue icon and two user avatars.
- Contact:** A section with a blue icon and a 'Choose' dropdown menu.
- Attachments:** A section with a blue icon and a plus sign.
- Comments:** A section with a blue icon and a text input field 'Add comment' with a green checkmark button.
- Metadata:** A sidebar on the right contains several dropdown menus: 'Project', 'Calendar', 'Scrum team', 'Date' (with 'Start' and 'End' date pickers set to 03.02.2023 and time pickers set to 8:00 and 15:00), 'Estimated hours' (with a 'hh:mm' input), 'Milestone', and 'Repetition'.

At the bottom of the form, there are 'Cancel' and 'Save' buttons.

Screen: System Firmbee - planowanie spotkania

Sprint w Scrum

Sprint to wydarzenie-pojemnik, które zawiera w sobie wszystkie pozostałe typy wydarzeń Scrum. Każdy Sprint może być rozumiany jako krótki projekt wyodrębniony z całości pracy nad Produktem. Jako taki, każdy Sprint ma określony Cel Sprintu i prowadzony przez Zespół Developerski Backlog Sprintu. Sprints to największe z wydarzeń w Scrum, które następują po sobie w nieprzerwanym cyklu, trwającym od początku do końca pracy nad Produktem. Zaś każda iteracja przybliża zespół do osiągnięcia Celu Produktu.

Każdy Sprint ma określony Cel Sprintu zapewniający spójność pracy Zespołu Developerów. Ma on formę celu biznesowego i odpowiada na pytanie „Po co?”, „W jakim celu?”, lub „Dlaczego?”. **Przebieg prac w danym Sprincie dokumentuje Backlog Sprintu, zawierający listę prac potrzebnych do realizacji Celu Sprintu.**

Struktura Sprintu

Każdy ze Sprintów ma określoną strukturę i zawiera w sobie:

- 🟡 **Sprint Planning** - to wydarzenie, od którego rozpoczyna się Sprint. Scrum Team wybiera podczas niego planowane prace z Backlogu Produktu, które są przeznaczone do wykonania w nowym Sprincie.
- 🟡 **Daily Scrum** - to codzienne wydarzenie, podczas którego Developerzy planują zadania na dany dzień.
- 🟡 **Sprint Review** - to otwarte dla Interesariuszy wydarzenie odbywające się w ostatnim dniu trwania Sprintu. Jego celem jest podsumowanie Sprintu pod kątem postępów w pracy nad Produktem.
- 🟡 **Sprint Retrospective** - to wydarzenie kończące Sprint, podczas którego Scrum Team rozmawia o sposobach pracy i pomysłach na jej usprawnienie.

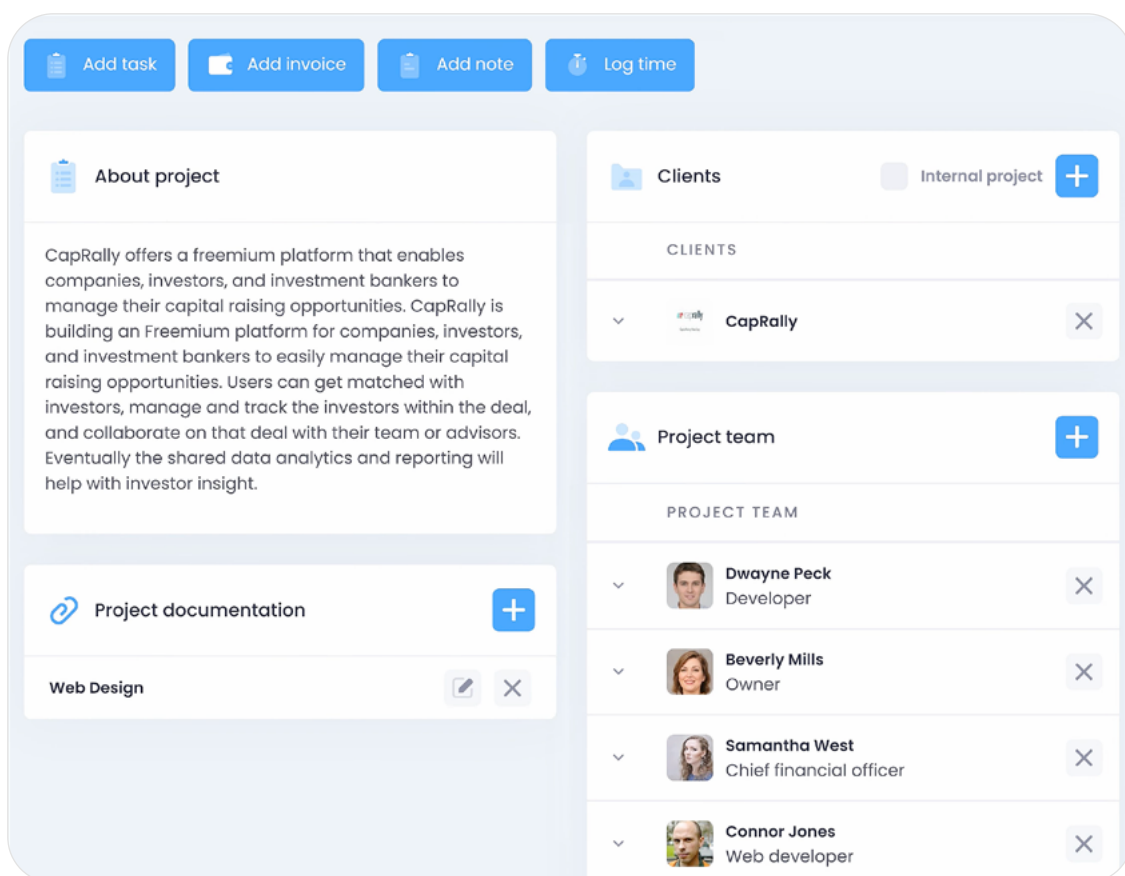
Powtarzalność wydarzeń w Sprincie sprzyja wdrażaniu dobrych praktyk organizacyjnych. Innymi słowy, Scrum Team wdraża się w rutynowe działania niezbędne do efektywnego planowania i podczas pracy zwraca uwagę na problemy, które będą mogły zostać omówione podczas odpowiednich wydarzeń.

Sprint i trzy filary empiryzmu

Dzięki Sprintom Scrum Team zyskuje podział pracy nad Produktem na równe odcinki czasowe trwające nie więcej niż miesiąc. Tak wyznaczone stałe ramy wzmacniają trzy filary empiryzmu: przejrzystość, inspekcję, adaptację. Przyjrzyjmy się, w jaki sposób filary empiryzmu znajdują odzwierciedlenie w Sprincie i jego strukturze.

1. Przejrzystość

Podział pracy na Sprints pozwala na wzmocnienie przejrzystości. Dzieje się tak, ponieważ wszystkie zainteresowane osoby mogą w każdym Sprincie otrzymać wymagane informacje na temat stanu pracy nad Produktem. Sprint Planning oraz Sprint Review, czyli rozpoczęcie i zakończenie Sprintu, łączą się z aktualizacją Backlogu Produktu. Dzięki temu wszyscy zainteresowani zyskują wgląd w bieżący stan prac nad Produktem. Zachowaniu przejrzystości pomagają dedykowane narzędzia do zarządzania projektami. Członkowie zespołu mogą na bieżąco kontrolować postęp prac, zapoznać się z udostępnionymi dokumentami czy też kontrolować budżet.



Screen: System Firmbee - szczegóły projektu

2. Inspekcja

Dzięki podziałowi pracy na Sprints możliwe jest częste sprawdzanie postępów w pracy. Sprzyja to regularnemu identyfikowaniu problemów w dwóch kluczowych obszarach. Są to:

- 🟡 **problemy związane z realizacją Celu Produktu** - w momencie rozpoczęcia i zakończenia Sprintu, czyli podczas Sprint Planningu oraz Sprint Review,
- 🟡 **problemy związane ze sposobami pracy Scrum Team** - podczas codziennych spotkań oraz na zakończenie każdego Sprintu, czyli podczas Daily Scrum i Sprint Retrospective.

3. Adaptacja

Adaptacja jest bardzo ważną częścią pracy Scrum Team, ponieważ pozwala na rozwiązywanie problemów zidentyfikowanych podczas inspekcji. W czasie trwania każdego Sprintu Daily Scrum oraz Sprint Retrospective stwarzają bezpieczną przestrzeń do rozmowy nad poprawą funkcjonowania Scrum Team. Zaproponowane rozwiązania mogą zostać wdrożone od razu bądź od następnego Sprintu. Sprint Planning i Sprint Review stwarzają przestrzeń do dyskusji nad Celami i sposobami realizacji Celu Produktu. Samozarządzający się Scrum Team może podczas tych wydarzeń podjąć decyzję o sposobie realizacji zadań w kolejnym Sprincie.

Jakie zmiany mogą zostać wprowadzone w czasie trwania Sprintu?

Każdy Sprint pozostawia Scrum Team duże pole do doskonalenia sposobów pracy.

Dlatego ważne jest określenie, jakie zmiany mogą zostać dokonane w czasie trwania Sprintu.

Przewodnik po Scrum nie zawiera listy takich zmian. Jednak zgodnie z duchem empiryzmu podaje reguły, które można dostosować do sposobu działania konkretnego Scrum Team.

1. Nie można dokonywać zmian zagrażających osiągnięciu Celu Sprintu. Zgodnie z pierwszą regułą, podczas trwania Sprintu nie można na przykład zmniejszać ilości zadań przewidzianych do wykonania w danym Sprincie lub istotnie zmieniać ich charakterystyki. Sprint jest ściśle związany z Celem Sprintu. Dlatego w momencie zmiany Celu, Sprint powinien zostać przerwany. Taka sytuacja powinna jednak praktycznie się nie zdarzyć. Jedynym powodem uzasadniającym przerwanie Sprintu jest bowiem dezaktualizacja Celu. Decyzję o przerwaniu Sprintu może podjąć tylko Product Owner.

2. Nie można obniżać jakości pracy. Zasada ta ma na celu zapobieganie sytuacji, w której praca wykonana w czasie Sprintu nie może stać się Przyrostem, ponieważ nie spełnia Definicji Ukończenia. Obniżenie jakości pracy może sprawić, że Cel Sprintu zostaje pozornie zrealizowany, jednak sposób wykonania poszczególnych zadań nie spełnia standardów jakościowych określonych przez organizację lub wymaganych przez Interesariuszy.

3. Można uszczegółowić Backlog Produktu. Podczas pracy nad Produktem wzrasta wiedza na jego temat. W naturalny sposób zwiększa się więc szczegółowość zadań do wykonania. Dlatego zmianą dopuszczalną, a nawet wskazaną w czasie trwania Sprintu jest uszczegółowienie Backlogu Produktu.

4. Można doprecyzować lub renegocjować zakres prac. Zmiana ta, tak jak poprzednia, wiąże się z coraz lepszym rozumieniem charakteru wykonywanych prac. Może dokonać jej Zespół Developerski w porozumieniu z Product Ownerem. Jednak podstawowym warunkiem jej wprowadzenia jest brak konfliktu z zasadami 1. i 2.

Cel Produktu, Cel Sprintu i Definicja Ukończenia

Każdy Artefakt Scrum tworzy pewne zobowiązanie dla Scrum Team. Cel Produktu, Cel Sprintu i Definicja Ukończenia opisują przyszły stan Produktu. Są one zamierzeniami lub też zobowiązaniami Scrum Team. Każdy obejmuje jednak inny zakres i skalę czasową. Cel Produktu został dodany do oficjalnego Przewodnika po Scrum dopiero w jego wersji z 2020 roku. Został w nim określony jako zobowiązanie dotyczące finalnego kształtu Produktu. Zostaje ono podjęte przez Scrum Team i zapisane w Backlogu Produktu. Cel Produktu dołączył w ten sposób do wcześniej funkcjonujących w Scrum Celów: Celu Sprintu, który dotyczy Backlogu Sprintu oraz Definicji Ukończenia, która jest zobowiązaniem Scrum Team dotyczącym realizacji Przyrostu według uzgodnionych wytycznych. Twórcy Scrum zdecydowali się w ten sposób na następujące powiązania:

Zobowiązanie	Artefakt Scrum
Cel produktu	Backlog Produktu
Cel Sprintu	Backlog Sprintu
Definicja Ukończenia	Przyrost

Podjęcie zobowiązania ma na celu wzmocnienie skupienia Scrum Team. Dzięki niemu zespół ma jasno określony cel i horyzont działań w perspektywie całości projektu, jednego Sprintu oraz Przyrostu. Wszystkie one powinny mieć na uwadze Wizję Produktu, czyli długoterminowy, strategiczny cel. Wizja nie jest zobowiązaniem Scrum Team, a raczej horyzontem wszystkich jego dążeń. Pomaga ona w odpowiednim formułowaniu Celów. Wizja Produktu może być sformułowana następująco: Zostać światowym liderem handlu i usług magicznych. W treści poniżej, przykłady częściowych Celów będą się odnosić do tego przykładu.

1. Cel Produktu

Cel Produktu jest opisem przyszłego Produktu, nad którego tworzeniem pracuje Scrum Team od rozpoczęcia po zakończenie danego projektu. Droga prowadząca do osiągnięcia Celu Produktu jest zapisana w Backlogu Produktu. Scrum Team może jednocześnie realizować tylko jeden Cel Produktu. Aby go zmienić, Scrum Team musi go zrealizować albo z niego zrezygnować. Dobre sformułowanie Celu Produktu jest zależne od branży w której działa Scrum Team. Jednak we wszystkich przypadkach Cel ten powinien być przekonujący, jasno wyrażony, oraz istotny. Innymi słowy, powinien opisywać kluczowe osiągnięcie przybliżające realizację Wizji Produktu. Może być to na przykład:

Cel Produktu 1: Otwarcie pierwszego na świecie sklepu online z akcesoriami magicznymi.

Cel Produktu 2: Stworzenie działającego 24/7 serwisu latających mioteł.

2. Cel Sprintu

Cel Sprintu jest opisem pracy, która zostanie wykonana w ramach jednego Sprintu.

Jest ona wyrażona w formie celu biznesowego, który pozwala zapewnić spójność pracy Zespołu Developerów. A także zwiększyć Skupienie - jedną z podstawowych Wartości Scrum. Realizacja Celu Produktu 1, który zaproponowaliśmy powyżej, mogłaby zostać podzielona na następujące Cele Sprintów:

Cel Sprintu 1.1: Przeprowadzenie badania rynku magów pod kątem zapotrzebowania na akcesoria magiczne.

Cel Sprintu 1.2: Sporządzenie listy dostawców i dostępności produktów.

Cel Sprintu 1.3: Stworzenie strony internetowej sklepu.

Natomiast dla Celu Produktu 2 Cele Sprintu mogłyby wyglądać następująco:

Cel Sprintu 2.1: Znalezienie dogodnej lokalizacji dla pierwszego punktu serwisowego.

Cel Sprintu 2.2: Przeprowadzenie rekrutacji serwisantów.

Cel Sprintu 2.3: Rozpoczęcie lokalnej akcji marketingowej.

Każdy Cel Sprintu jest określany i omawiany przez Scrum Team podczas Sprint Planningu.

Wydarzenie Sprint Planningu nie może zostać zakończone bez określenia Celu kolejnego Sprintu.

3. Definicja Ukończenia

Definicja Ukończenia jest kolejnym szczegółowym zobowiązaniem podejmowanym przez Scrum Team. W przeciwieństwie do Celów Produktu i Sprintu, ma ona charakter formalny. Zwykle składa się z **listy kryteriów jakościowych, które musi spełnić Produkt, aby Przyrost został uznany za gotowy. Dzięki Definicji Ukończenia członkowie Scrum Team oraz każdy z Interesariuszy może łatwo przekonać się, jakie zmiany i ulepszenia pojawiły się w Produkcie dzięki danemu Przyrostowi.**

W odniesieniu do Celu Sprintu 1.1: *Przeprowadzenie badania rynku magów pod kątem zapotrzebowania na akcesoria magiczne*, kryterium Przyrostu może być: *Opracowanie wyników badań magów aktywnych na forum magicznym*. Definicja ukończenia tego przyrostu mogłyby brzmieć następująco:

Definicja Ukończenia Przyrostu 1.1:

- 🟡 *Każdy z magów udzielił wyczerpujących odpowiedzi na pytania.*
- 🟡 *Co najmniej 80% magów jest czynnych zawodowo.*
- 🟡 *Rezultaty badań zostały wprowadzone do bazy danych magicznej kuli.*

Definicja ukończenia może także odnosić się do standardów jakościowych określonych przez organizację. Wtedy nie jest ona swobodnie ustalana przez Scrum Team. Jej minimalne wymagania wyznaczają zewnętrzne wymagania organizacji. Definicja Ukończenia różni się jednak od kryteriów akceptacji.

Odnosząc się do przykładu Definicji Ukończenia 1.1, kryteria akceptacji można by sformułować następująco:

Kryterium Akceptacji 1.1:

- 🟡 *Oszacowanie odsetka magów, którzy będą skłonni korzystać ze sklepu internetowego.*
- 🟡 *Określenie, jakie akcesoria cieszą się wśród nich największym zainteresowaniem.*
- 🟡 *Określenie jakim budżetem dysponują potencjalni klienci.*

Kryterium akceptacji odnosi się zatem do realizacji zadania, tak jak jest ono postrzegane przez Klienta.

Wykres Spalania

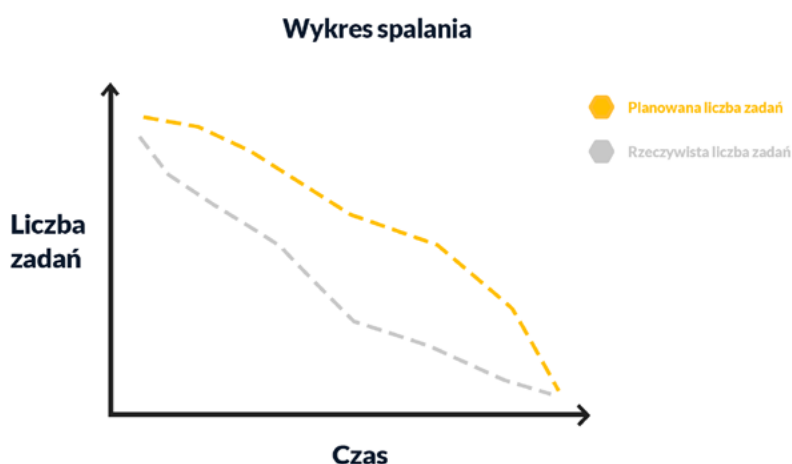
Wykres Spalania pozwala ocenić, na jakim etapie realizacji znajduje się bieżący Sprint lub cały projekt realizowany przez Scrum Team. Przede wszystkim pokazuje jednak, ile czasu potrzeba na realizację Celu Sprintu lub Celu Produktu. Do pewnego stopnia pozwala również mierzyć zmiany efektywności pracy zespołu Developerskiego. A dzięki temu ocenić, jakie jest prawdopodobieństwo ukończenia zadań na czas. Wykres Spalania pokazuje bowiem, ile pracy już wykonano, a ile pozostało jeszcze do wykonania.

Wykres spalania służy do szacowania czasu i nakładu pracy potrzebnych do wykonania zadania.

Jest też cennym narzędziem ostrzegawczym, jeśli praca nie idzie zgodnie z planem. Dzięki niemu za pomocą jednego rzutu oka każdy z członków Scrum Team jest w stanie ocenić stan realizacji zadań. Dlatego też Wykres Spalania powinien być łatwo dostępny dla wszystkich osób zaangażowanych w projekt. Wykres Spalania jest w Scrum wykorzystywany w dwóch wersjach: jako Wykres Spalania Sprintu oraz jako Wykres Spalania Projektu. Będziemy omawiać je łącznie, ponieważ zasada ich działania jest identyczna. Różnią się one jedynie skalą czasową i ilością zadań, które obrazują.

Jak wygląda Wykres Spalania?

Na wykresie spalania na osi X przedstawiony jest czas pozostały na wykonanie pracy. Jeśli wykres dotyczy Sprintu, czas mierzony jest dniami. Natomiast jeśli Wykres Spalania dotyczy całego projektu, czas najczęściej mierzony jest w Sprintach. Oś Y wykresu spalania wyznacza ilość pracy pozostałej do wykonania. Odnosi się więc do zadań zamieszczonych w Backlogu Sprintu lub w Backlogu Produktu. Jednostkami używanymi na pionowej osi wykresu mogą być: Story Points, Dniówki/osobogodziny (Man Days/Hours), Zadania (Tasks). Wykres Spalania zazwyczaj uwzględnia linię spalania przedstawiającą idealny, liniowy spadek ilości zadań, jakie pozostały do wykonania. Dzięki niemu można zorientować się, jaki jest planowany czas zakończenia.



Szacowanie czasu realizacji zadań

Każdy z Developerów z osobna musi oszacować ilość godzin potrzebnych mu na wykonanie konkretnej pracy. Zaś szacowany czas realizacji Celu Sprintu bądź Projektu jest sumą czasu szacowanego przez wszystkich Developerów. Dużą trudność stanowi przełożenie szacowanego czasu wykonania (estimated time) na rzeczywisty czas realizacji zadań. Jest to jeden z większych problemów trapiących nowo utworzone Zespoły Developerskie. Może on zaburzać obraz przedstawiony na wykresie spalania.

Co więcej, nie zawsze udaje się od razu stworzyć optymalny przepływ pracy w Zespole Developerskim. Przez to w wykonywaniu zadań mogą pojawiać się przestoje, które wydłużają czas realizacji zadań. Jednym ze wskaźników dojrzałości Zespołu jest umiejętność trafnego oszacowania czasu pracy. Im większa przewidywalność, tym bardziej dojrzały jest zespół. Znajduje to odzwierciedlenie w wykresie spalania, na którym linia odzwierciedlająca realne spalanie opada równomiernie do zera. Sformułowanie to nie dotyczy jednak wszystkich sytuacji.

Jak wspomnieliśmy wcześniej, podczas omawiania statystyk monitorowanych przez Scrum Mastera, wiele zależy od charakteru samego projektu: jego innowacyjności oraz powtarzalności zadań wykonywanych przez Developerów. **Wykres Spalania sprawdza się bowiem najlepiej w przypadku projektów, w których zakres prac jest stały i znany.** Dlatego też w przypadku innowacyjnych projektów lepiej zastosować inne sposoby pomiaru efektywności zespołu.

Jak tworzyć i jak interpretować Wykres Spalania?

Wykres Spalania jest stosunkowo łatwy do stworzenia. Istnieje wiele narzędzi pozwalających na jego generowanie na podstawie pracy logowanej przez członków Zespołu Developerskiego. Pomimo prostoty, jego interpretacja może być źródłem cennych informacji dla całego Scrum Team.

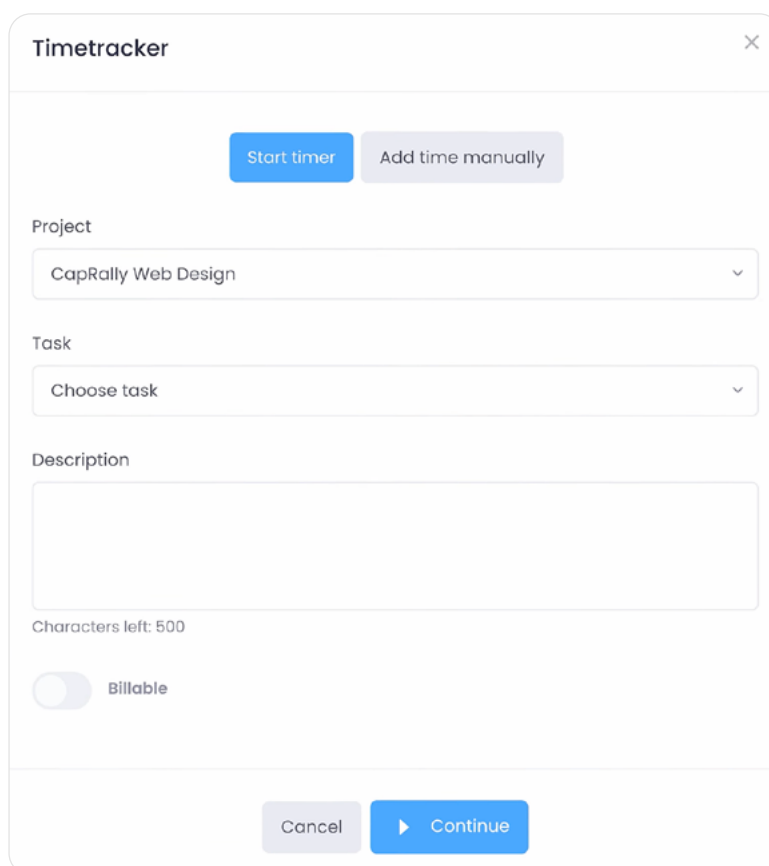
Zespół Developerski powinien monitorować swoją codzienną pracę. Jest to podstawą nie tylko oceny jego efektywności, lecz także jej poprawy zaś jednym z najprostszych i sprawdzonych narzędzi służących do tego celu jest Wykres Spalania. Można stworzyć go ręcznie, rysując na kartce papieru układ współrzędnych. Na osi Y trzeba nanieść ilość pracy wyrażoną w wybranej jednostce, na przykład Story Points. Na osi X wyrysować skalę wyznaczającą kolejne dni Sprintu. Wyrysować linię idealnego spalania, a potem dla każdego dnia oznaczać ilość realnie zrealizowanych zadań. Choć to rozwiązanie urocze i angażujące Zespół, jest niezbyt praktyczne. Niekoniecznie sprawdzi się też w przypadku Zespołu pracującego zdalnie.

Niektóre narzędzia służące do logowania pracy nad zadaniami rozdzielonymi pomiędzy członków Zespołu, jest wyposażonych w opcję automatycznego tworzenia wykresu spalania. Wtedy wystarczy, że Developer oznaczy rozpoczęcie i zakończenie pracy nad daną funkcjonalnością produktu, a jego wkład znajdzie odzwierciedlenie na wykresie spalania. Dzięki użyciu narzędzi możliwe jest też dowolne skalowanie wykresu. Daje to wgląd w spalanie nie tylko na poziomie danego Sprintu, lecz również w skali kwartału czy całego projektu.

Ważnym czynnikiem, jaki trzeba wziąć pod uwagę przy wyborze narzędzia służącego do tworzenia wykresu spalania jest jego dostępność dla wszystkich członków Scrum Team. **Widoczność wykresu spalania dla całego Zespołu Developerskiego jest kluczowym czynnikiem motywacyjnym.**

Równie ważne jest codzienne spojrzenie na linię ukazującą pracę pozostałą do wykonania. Rozmowa na temat spalania podczas Daily Scrum, skłania Developerów do zastanowienia nad sposobami pracy i bieżącym stanem Produktu.

Zespoły zdalne mogą jednak skorzystać z funkcji logowania czasu, która jest dostępna w systemie do zarządzania projektami. Czas poświęcony na wykonanie zadania można zalogować włączając i wyłączając timer (czasomierz) lub wpisując godziny ręcznie.



[Screen: System Firmbee - Logowanie czasu](#)

Kto jest odpowiedzialny za Wykres Spalania?

Kwestia własności wykresu spalania budzi nieco kontrowersji. Z jednej strony powinien on należeć do Scrum Mastera, ponieważ jest narzędziem pozwalającym upewnić się, że Zespół pracuje wydajnie i zgodnie z planem. Z drugiej strony, powinien należeć do Product Ownera, ponieważ odzwierciedla postęp na drodze do Celu Produktu, który może zostać przekazany Klientowi. Z trzeciej natomiast - jest wewnętrznym narzędziem Zespołu Developerskiego.

Wykres Spalania jest bardzo ważną metryką pozwalającą ocenić efektywność Zespołu Developerskiego i korzystają z niego wszyscy członkowie Scrum Team. Dlatego tak istotna jest jego przejrzystość i dostępność. Jednak samo jego prowadzenie ma w założeniu służyć przede wszystkim Zespołowi. Ma wzmacniać jego samoorganizację, poprawiać motywację, oraz dawać realny obraz stanu prac nad powierzonymi mu zadaniami. Dlatego w teorii każdy z członków Zespołu Developerskiego może aktualizować Wykres Spalania.

W praktyce jednak zadanie aktualizowania wykresu spalania przypada zwykle Scrum Masterowi.

Dzieje się tak szczególnie na początku jego pracy z nowym Zespołem Developerskim, kiedy Prędkość Zespołu jest jeszcze zmienna i trudna do oszacowania. Niemniej jednak zaleca się przekazanie tego zadania jednemu z Developerów. Wykres Spalania ma być bowiem szczerą i wewnętrzną miarą postępów w pracy tak, jak oceniają ją sami Developerzy.

Spalanie realne i idealne

Dla interpretacji wykresu spalania bardzo ważnym czynnikiem jest nie tylko regularne nanoszenie realnego „spalania”, czyli realizacji zadań przez Zespół Developerski. Równie istotne dla obrazu sytuacji jest jego porównanie z idealnym spadkiem linii spalania (guideline). Dzięki porównaniu idealnej linii spalania z realnym zmniejszaniem ilości pracy oznaczanym na wykresie spalania, można ocenić dwa bardzo istotne parametry. Po pierwsze sprawdzić, czy jeśli praca będzie przebiegać w dotychczasowym tempie, Zespół Developerski zrealizuje na czas Cel Sprintu lub Cel Produktu. Po drugie, zorientować się, kiedy prace zostaną ukończone przy utrzymaniu obecnego tempa. Innymi słowy, Wykres Spalania obrazuje rzeczywiste tempo realizacji zadań, zaś linia idealna pokazuje w jakim tempie Zespół powinien pracować aby ukończyć założone zadania.

Wykres Spalania pozwala także w dłuższej perspektywie określić wartość nazywaną Prędkością Zespołu Developerskiego. Jest to wartość wyznaczana przez ilość pracy wykonanej w czasie jednego Sprintu. Dzięki temu, że Wykres Spalania obrazuje porównanie idealnej linii spalania z realnym zmniejszaniem ilości zadań, pozwala oszacować tempo pracy. A tym samym przewidywać ryzyko opóźnień w realizacji projektu.

Wybór jednostki pomiaru

Prędkość Zespołu zwykle mierzona jest w jednostkach nazywanych Story Points.

Określa ona ilość zrealizowanych user stories. Te zaś mogą wymagać bardzo różnego nakładu pracy. Dlatego wiele Scrum Teams stosuje miarę czasową. W zależności od skali są to dniówki lub osobogodziny. Każdy z Developerów szacuje, a następnie loguje ilość czasu poświęconego na wykonanie swoich zadań.

Kolejną opcją jest wykorzystanie jako jednostki zadań (tasks). Są to nieco większe całości, którym z kolei może zostać przypisana wartość wyrażona w Story Points, albo w dniówkach czy osobogodzinach. Jest to jednostka pozwalająca przedstawić Klientowi postępy w pracy nad Produktem w bardziej czytelny sposób. Niezależnie od zastosowanej jednostki pomiaru, warto pamiętać o zasadzie obliczania prędkości Zespołu Developerskiego. W danym dniu lub Sprincie liczą się tylko zadania, które zostały realnie ukończone. Oznacza to, że rozpoczęte zostaną zaliczone na poczet następnego dnia bądź Sprintu nawet jeśli zabrakło tylko finalnych testów.

Zalety i wady Wykresu Spalania

Wykres Spalania ma wiele zalet. Nie bez powodu jest jednym z głównych narzędzi metrycznych używanych w Scrum. Jest łatwy do stworzenia, skalowalny i czytelny. Ma jednak także wady, które sprawiają, że nie jest narzędziem uniwersalnym. O tym czym jest, jak tworzyć i interpretować Wykres Spalania pisaliśmy już wcześniej. Teraz skoncentrujemy się na zaletach i wadach tego rozwiązania. Większość z nich nie kryje się jednak w samym prostym wykresie. Wiążą się one raczej ze sposobami wykorzystywania wykresu spalania do motywowania Zespołu Developerskiego, opisywania rezultatów jego pracy i wzmacniania samoorganizacji.

1. Zalety Wykresu Spalania

Wykres Spalania pozwala w czytelny sposób zwizualizować postęp prac w projekcie.

To właśnie czytelność i prostota wykonania decydują o jego ogromnej popularności. Dlatego warto, by Wykres Spalania był nie tylko na bieżąco aktualizowaną metryką ukrytą w cyfrowym narzędziu do zarządzania projektem. Jeśli to tylko możliwe, warto by stał się punktem odniesienia dla Zespołu Developerskiego widocznym w fizycznym miejscu pracy. Czy to w formie wyświetlanej na ekranie wizualizacji, czy ręcznie narysowanego szkicu.

🟡 **Motywacja Zespołu Developerskiego** - Przejrzystość wykresu spalania może sprawić, że stanie się on narzędziem motywującym Zespół Developerski do wydajnej pracy. **Osiągnięcie w każdym Sprincie punktu „zero” może stać się ambicjonalnym celem Zespołu, za który są przyznawane nagrody** - zgodnie z zasadami biznesowej gamifikacji. Widoczność aktualnego i ciekawie prowadzonego wykresu spalania może również budować ducha współpracy i samoorganizacji. Metryka jest bowiem miarą zespołowej pracy. Nie widać na niej, kto dokładnie wykonał - lub nie wykonał - zaplanowane zadania, a jedynie osiągnięte rezultaty.

🟡 **Miara realnie wykonanej pracy** - To Developerzy decydują, ile zadań wykonują w danym Sprincie. Im bardziej doświadczony jest Zespół, z tym większą trafnością powinien przewidywać swoje działania. Zaś Wykres Spalania odzwierciedla realny przebieg Sprintu. **Zaletą wykresu spalania jest więc nie tyle mierzenie obiektywnej ilości wykonanej pracy, ile stosunku zaplanowanych do realnie wykonanych zadań.** Dzięki temu Developerzy stopniowo uczą się ich planowania i mogą coraz dokładniej oszacować swoje możliwości oraz eliminować powtarzalne błędy.

🟡 **Możliwość łączenia z innymi narzędziami** - Jedną z istotnych zalet wykresu spalania są wyniki, jakie można uzyskać dzięki połączeniu go z innymi narzędziami. Mogą być to narzędzia służące do: analizy pracy Zespołu Developerskiego, wizualizacji postępu pracy nad Produktem, czy też szacowania budżetu projektu. Przykładowo, w tym ostatnim przypadku wykorzystanie wykresu spalania w skali Projektu pozwala na porównanie planowanego i rzeczywistego budżetu całego przedsięwzięcia.

NUMBER	TI	CUSTOMER TI	STATUS TI	DATE OF ISSUE TI	DUE DATE TI	TOTAL AMOUNT	AMOUNT DUE
R/3/10/2020/DEF		Bitpay	Paid	07.10.2020	21.10.2020	200,00 \$ 200,00 €	

Screen: System Firmbee - Wpływy i wydatki projektu

2. Wady wykresu spalania

Pomimo wszystkich przedstawionych powyżej zalet, Wykres Spalania może stać się źródłem nieporozumień w Zespole Developerskim. Często to, co nazywamy „wadami” wykresu spalania nie jest jednak spowodowane niedoskonałością narzędzia. Przedstawione poniżej problemy wynikają bowiem raczej ze sposobu używania wykresu spalania, niż z jego konstrukcji. Poniżej przedstawiamy wady, jakie mogą zakłócać obrazowanie w ten sposób postępów w pracy Zespołu Developerskiego.

🟡 „Czynnik ludzki”

Wykresy nie mogą stanowić absolutnej miary postępu prac zespołu. Są to tylko narzędzia, które mogą być stosowane w różny, mniej lub bardziej umiety sposób. Może to zostać uznane za wadę (lub zaletę) nie tylko wykresu spalania, lecz także innych mierników efektywności zespołu. Do stworzenia wykresu spalania potrzebne są dane wprowadzane przez ludzi. Innymi słowy, czas wykonania zadania jest wprowadzany do wykresu przez Developera. Mógł on nieco wydłużyć lub skrócić - przez nieuwagę lub chcąc poprawić sytuację zespołu. Developerom zdarza się też zapomnieć o logowaniu czasu pracy lub pozostawić włączony licznik. Sprawia to, że czas pracy wydłuża się do kilkunastu godzin. A po odkryciu pomyłki trudno jest odtworzyć jego realny przebieg.

🟡 Zmiany w Backlogu Sprintu

Backlog Sprintu nie powinien być modyfikowany po rozpoczęciu Sprintu. Jednak w praktyce dość często zdarzają się takie zmiany. Wynikają one ze zmieniających się wymagań Interesariuszy albo nieprzewidzianych wcześniej problemów, na jakie trafiają Developerzy. Sprawia to, że Wykres Spalania zostaje przeskalowany. Czas wykonania zadań pozostaje bowiem ten sam. Zwiększa się natomiast skala zadań pozostałych do wykonania. Może to sprawić mylne wrażenie, że Zespół Developerski nieprawidłowo zaplanował prace do wykonania w danym Sprincie. Albo też, że pracuje on zbyt wolno.

Zmiany w Backlogu Sprintu mogą też wynikać ze zbyt szybkiego ukończenia zadań, które zostały przewidziane do realizacji. W takiej sytuacji Zespół Developerski decyduje zwykle o zwiększeniu ilości zadań. To z kolei może skutkować nieukończeniem ich na czas, a także konfliktem wynikającym z nawarstwienia się zadań pozostałych z poprzedniego Sprintu z nowymi, zaplanowanymi do wykonania przez Interesariuszy i Product Ownera.

🟡 Zmiany w Backlogu Produktu

Duże zmiany w Backlogu Produktu mogą zaburzać kształt wykresu spalania, a tym samym mocno fałszować obraz postępów prac i efektywności Zespołu. Dzieje się tak, gdy pojawiają się nowe User Stories, zaś te, które zbliżają się do fazy realizacji, są często rozbijane na mniejsze części. Zdarza się również, że Klient rezygnuje z niektórych funkcjonalności Produktu. Dlatego podczas interpretacji wykresu spalania trzeba kierować się wiedzą i doświadczeniem w ocenie pracy Zespołu. A także brać pod uwagę zmienność Backlogu. Jeśli Wykres Spalania nie jest jedyną metryką stosowaną do oceny efektywności, inne wykresy, pozwolą dostrzec pełniejszy obraz postępu prac.

Tablice Kanban w Scrum i Scrumban

Scrum i Kanban to sposoby pracy zespołowej, które łączy wiele podobieństw. Z tego, co najlepsze w obu metodykach powstał Scrumban. Jest on najczęściej stosowany w projektach łączących tworzenie Produktu ze świadczeniem usług, gdzie nie zawsze sprawdzają się długie Sprints i stosunkowo sformalizowane spotkania Scrumowe. Kanban to metoda wywodząca się z Japonii. Powstała w latach '50 dwudziestego wieku i pierwotnie była narzędziem służącym do zarządzania ciągłą produkcją w taki sposób, aby nie tworzyć zapasów i nadwyżek, tylko na bieżąco przetwarzać zasoby. Na początku XXI wieku Kanban został przystosowany do potrzeb tworzenia oprogramowania przez Davida J. Andersona.

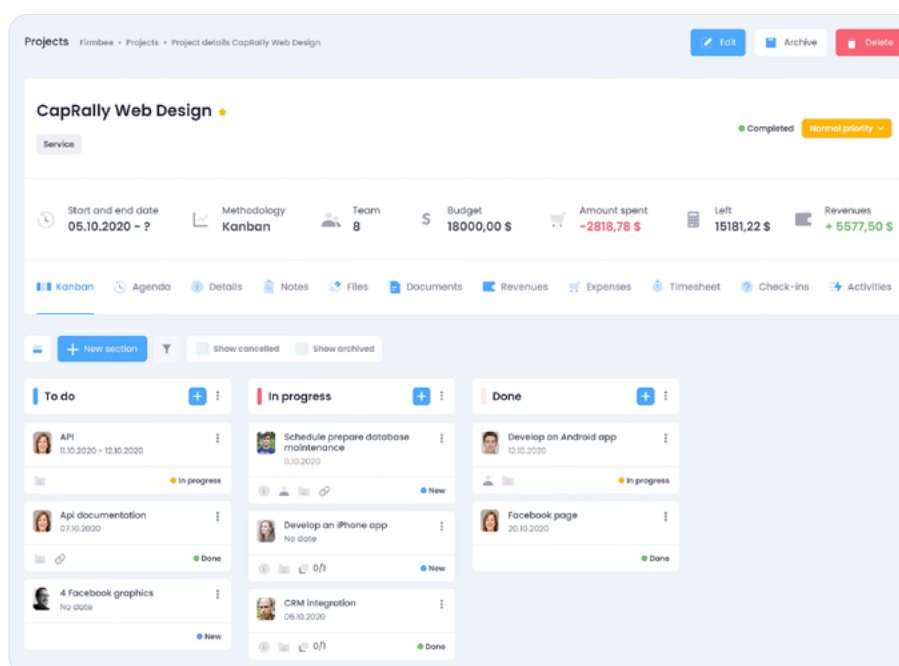
Kanban a Scrum

Całościowy sposób pracy w Kanban różni się od Scrum przede wszystkim mniejszym stopniem formalizacji. W Kanbanie nie ma tak szczegółowych wytycznych dotyczących na przykład pracy w Sprintsach, ról Product Ownera, Scrum Mastera i Zespołu Developerskiego. Jest to możliwe, ponieważ Kanban koncentruje się na ciągłości zadań takich jak świadczenie usług określonego rodzaju, które są bardziej powtarzalne i nie wymagają tak złożonego planowania. Podobne są natomiast cel i sposoby pracy.

Celem Kanbana jest terminowe dostarczenie Klientowi Produktu jak najwyższej jakości.

Natomiast zasady dotyczące wspólnych obu metodom sposobów pracy można sformułować następująco:

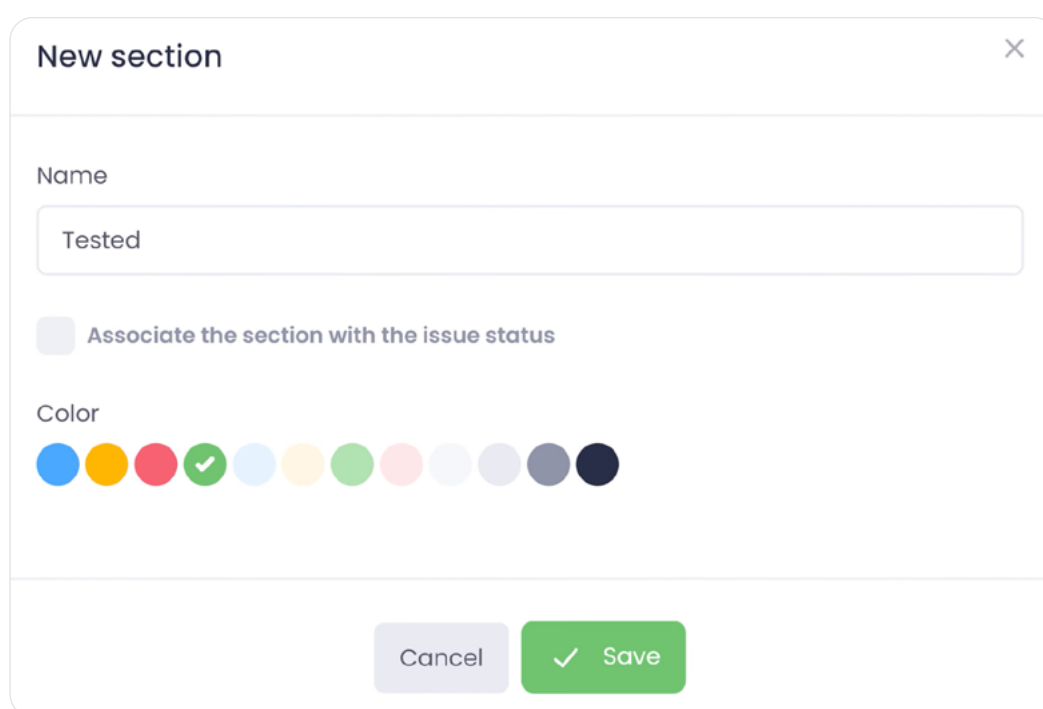
- 🟡 **Praca powinna przebiegać płynnie i bez przestojów** - w Scrum dzieje się to dzięki ciągłemu następowaniu po sobie Sprintów, zaś praca w Kanbanie jest nieustanna dzięki płynnemu napływowi zadań. Tworzą one kolejkę, z której deweloperzy wybierają (pull) kilka zadań do realizacji.
- 🟡 **Zespół powinien koncentrować się wyłącznie na zadaniach wybranych do realizacji** - używając terminologii Kanban, zespół powinien „ograniczać pracę w toku”. W Scrum odpowiednikiem są zadania w formie User Stories wybrane z Backlogu Produktu do umieszczenia w Backlogu Sprintu
- 🟡 **Postępy w wykonywaniu zadań powinny być widoczne dla wszystkich zainteresowanych osób** - w Kanbanie są one wizualizowane za pomocą tablic, które są także często wykorzystywane przez Scrum Teams.



Tablice Kanban w Scrum

Tablica Kanban jest szeroko wykorzystywanym narzędziem służącym do wizualizacji pracy zespołowej. Jest to tabela z kilkoma kolumnami. W każdej z nich znajdują się zadania o określonym statusie. Kategoryzacja zadań opiera się zaś na prostej zasadzie: karteczkę z opisem zadania - lub jej wirtualny odpowiednik - umieszcza się w jednej z kolumn. Wersja minimalna tablic Kanban zawiera trzy kolumny: do wykonania, w trakcie realizacji oraz ukończone - do ostatniej kolumny przesuwane są zadania, które spełniają Definicję Ukończenia. Bardzo często zdarza się, że kolumn jest więcej.

Jeśli zadań do wykonania jest więcej, zwykle pomiędzy kolumnami „do wykonania” i „w trakcie realizacji” pojawia się dodatkowa, zatytułowana „wybrane do realizacji”. O ile kolumna „do wykonania” pełni rolę Backlogu Produktu, kolumna „wybrane do realizacji” pełni funkcję Backlogu Sprintu.



Screen: System Firmbee - Tworzenie nowej sekcji w kanbanie

Drugim częstym uzupełnieniem jest kolumna zatytułowana „w trakcie oceny” lub „do akceptacji”. Jest ona zwykle wpisywana pomiędzy kolumny zawierające zadania „w trakcie realizacji” i te „ukończone”. Znajdują się w niej zadania ukończone przez Zespół Developerski, oczekujące na akceptację ze strony Product Ownera. Zadaniem Product Ownera jest sprawdzenie ich zgodności z kryteriami akceptacji oraz uzyskanie ich ostatecznego zatwierdzenia przez Klienta. W takiej sytuacji, do ostatniej kolumny przenoszone są tylko zadania ostatecznie zaakceptowane.

Scrumban

W związku z ogromną popularnością Scrum i Kanbana, powstała ich hybryda łącząca to, co najlepsze w obu tych sposobach pracy. **Scrumban sprawdza się najlepiej w organizacjach, które łączą tworzenie Produktów ze świadczeniem usług, polegających często na implementacji Produktu u Klienta.**

Ze względu na ograniczenie ilości spotkań i komunikacji, Zespół może być większy. Scrumban kładzie mniejszy nacisk na metryki stosowane powszechnie w Scrum, takie jak na przykład Wykres Spalania. Korzysta jednak z filarów Scrum dotyczących potrzeby ciągłego ulepszania procesu pracy i dostosowywania ich do warunków i potrzeb Klienta. Podczas pracy w Scrumban praca nie jest jednak podzielona na Sprints. Spotkania podsumowujące odbywają się co 3, 6 lub 12 miesięcy.

Planowanie pracy przebiega według zasady „On-Demand”, czyli w miarę jej pojawiania się. User Stories są umieszczane bezpośrednio w pierwszej kolumnie tablicy Kanban zawierającej zadania „do wykonania”. Pełni ona więc rolę Backlogu Sprintu.. Tak jak w Backlogu Sprintu, na górze listy zadań do wykonania znajdują się te najpilniejsze. Przy bardziej złożonych projektach Project Manager może jednak prowadzić osobną listę zadań do wykonania odpowiadającą Backlogowi Produktu, z którego wybiera zadania do umieszczenia w pierwszej kolumnie.

Przy przenoszeniu zadań z pierwszej do drugiej kolumny obowiązuje zasada „Pull”. Oznacza ona, że zadania nie są delegowane konkretnemu Deweloperowi. Każda osoba wybiera z kolejki zadanie do realizacji i samodzielnie je realizuje. Ilość zadań umieszczonych w środkowej kolumnie, „do realizacji” jest zwykle ograniczona w zależności od wielkości zespołu, tak aby w miarę możliwości każdy zajmował się tylko jednym zadaniem na raz.

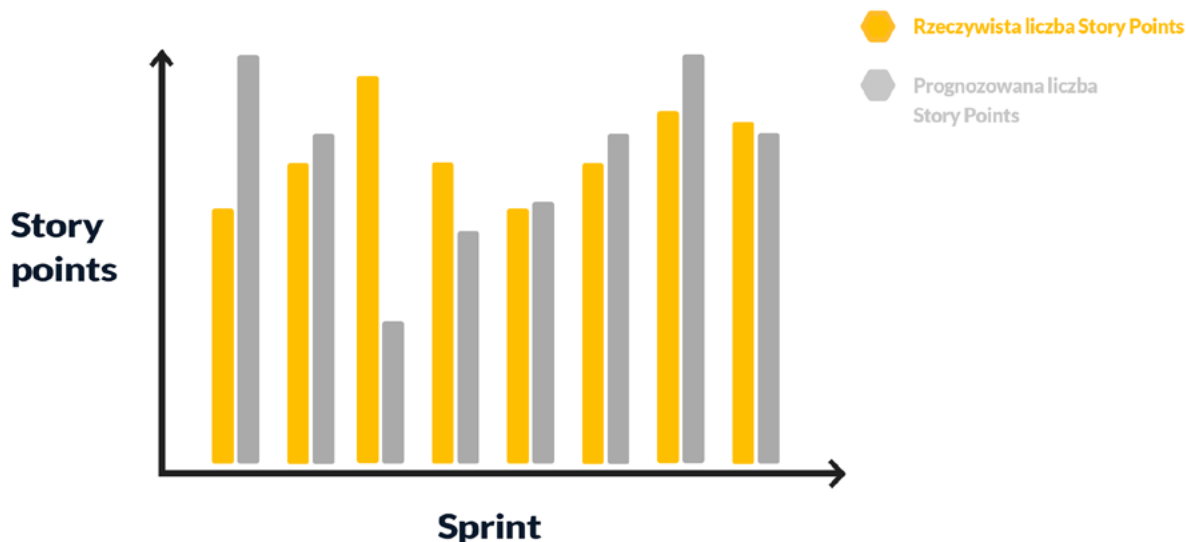
Prędkość Zespołu Developerskiego

Prędkość Zespołu pozwala określić tempo, w jakim Scrum Team realizuje zadania.

Można zdefiniować ją jako średnią ilość Story Points wykonanych w jednym Sprincie. Prędkość Zespołu może służyć także do szacowania czasu trwania projektu na podstawie już zrealizowanych postępów w pracy. Ma to jednak sens jedynie w przypadku dojrzałego zespołu, który pracuje w równym i stabilnym tempie.

Prędkość Zespołu jest opcjonalnym, lecz chętnie używanym sposobem mierzenia tempa pracy Scrum Team. Dzieje się tak, ponieważ **trafnie oszacowana Prędkość Zespołu pozwala w pewnym stopniu prognozować czas potrzebny na realizację projektu**. Jest to jednak miara, która może być stosowana tylko w odniesieniu do danego Zespołu Developerskiego, który będzie wykonywał zadania, które sam „wycenił” używając znanej sobie jednostki, takiej jak na przykład Story Points.

Prędkość Zespołu Developerskiego jest najczęściej przedstawiana w formie Velocity Chart. Na osi X są tutaj oznaczane kolejne Sprints. Natomiast na osi Y znajdziemy ilość Story Points lub innych, odpowiadających im jednostek, które zostały zrealizowane w danym Sprincie. Dzięki użyciu Velocity Chart, Scrum Team zyskuje wizualizację zmian w tempie swojej pracy. Jeśli linia oznaczona na wykresie się wznosi, oznacza to, że Zespół optymalizuje swą efektywność lub zmniejsza wartość Story Points. Zarówno Scrum Master jak i Product Owner powinni więc uważnie śledzić przebieg linii ukazującej Prędkość Zespołu.



Prędkość rzeczywista i planowana

Rzeczywista Prędkość Zespołu Developerskiego opisuje tempo pracy w ukończonym Sprincie i jest obliczana na koniec każdego z nich. Przyjmuje ona wartość sumy Story Points dla wszystkich zrealizowanych User Stories. Rzeczywista Prędkość Zespołu Developerskiego pozwala na planowanie i szacowanie z pewnym prawdopodobieństwem tempa realizacji przyszłych zadań.

Planowana Prędkość Zespołu jest natomiast szacowana na podstawie uśrednionej wartości rzeczywistej Prędkości. Wymaga ona przyjęcia założenia o braku zmian w Zespole Developerskim. Jest ważnym wewnętrznym narzędziem Zespołu Developerskiego, który na jej podstawie może ocenić, czy współpraca w Zespole przebiega właściwie i czy tempo pracy zostaje zachowane.

Prędkość Zespołu planowana pozwala także Product Ownerowi na oszacowanie czasu realizacji dobrze zdefiniowanych User Stories przewidzianych do wykonania w kolejnych Sprintach. Dzięki temu możliwa jest sprawniejsza pielęgnacja Backlogu Produktu. Jednak praktyka stosowania planowanej Prędkości Zespołu do szacowania długości trwania projektów nie jest tak prosta.

Trudności i zagrożenia związane z Prędkością Zespołu

Prędkość Zespołu często przypisuje się zbyt wielkie znaczenie, nie biorąc pod uwagę następujących czynników:

- szacowanie większych całości lub całego projektu** - o ile Zespół Developerski jest w stanie trafnie oszacować ilość Story Points, którą należy przypisać konkretnemu zadaniu, opisanie w tych jednostkach większych całości przeznaczonych do przyszłej realizacji jest bardzo trudne lub niemożliwe,
- zmiany w projekcie** - każda zmiana w projekcie oznacza potencjalnie zmianę ilości Story Points niezbędnych do realizacji Celu Produktu. Może też okazać się, że już wykonane zadania będą wymagały modyfikacji lub nawet nie zostaną wykorzystane w finalnej wersji Produktu,
- nieprzewidziane wydarzenia** - przewidywanie tempa realizacji przyszłych projektów na podstawie tych już zrealizowanych, czyli przekładanie rzeczywistej Prędkości Zespołu na Planowaną, może owocować trafnymi szacunkami. Jednak każdy projekt ma swoją specyfikę i trafne przewidywanie w oparciu o historię jest zwykle niemożliwe.

Daily Scrum

Daily Scrum trwa nie więcej niż piętnaście minut i odbywa się zawsze w tym samym miejscu i o tej samej porze, żeby ograniczać niepotrzebną złożoność. Biorą w nim udział wszyscy Developerzy pracujący wspólnie nad Produktem oraz - opcjonalnie - Scrum Master. Głównym celem tego Wydarzenia Scrum jest zaplanowanie zadań, na których realizacji będą się koncentrować w danym dniu. Daily Scrum to najkrótsze i najczęstsze z Wydarzeń Scrum. Zadaniem Developerów uczestniczących w Daily Scrum jest szybkie wyznaczenie celów pracy na najbliższe 24 godziny. Dzięki temu każdy z nich wie, nad czym pracują inni i w jaki sposób zmierzają do wspólnego osiągnięcia Celu Sprintu.

Formuła Daily Scrum

Nie istnieje jedna właściwa formuła Daily Scrum. Każdy Zespół Developerski wypracowuje działający dla niego format spotkania. Istnieją jednak ogólne ramy ułatwiające jego przeprowadzanie. Dobrze przeprowadzony Daily Scrum powinien dawać każdemu z uczestników możliwość odpowiedzi na dwa pytania:

- 🟡 Jakie jest najważniejsze zadanie, które wykonam dzisiaj?
- 🟡 Jakie są przeszkody na drodze do realizacji tego zadania?



Jednak zadanie ich wprost nie jest obowiązkową formułą. Są to przykładowe pytania określające oś spotkania. **Daily Scrum ma służyć poprawieniu komunikacji w Zespole Developerskim, priorytetyzacji zadań i zmniejszeniu ryzyka zastoju w pracy (bottlenecks).** Daily Scrum jest wydarzeniem odpowiadającym Daily Standup w innych metodach Agile i często przebiega bardzo podobnie do niego - chociaż oficjalny Przewodnik po Scrum nie wymaga od Developerów stania podczas tego krótkiego Wydarzenia. Bardzo często jego uczestnicy po prostu stoją podczas rozmowy w nieformalnej grupie.

Choć może się wydawać że 15 minut dziennie to dużo jak na omówienie codziennych zadań, z praktyki wynika, że takie właśnie spotkanie najlepiej wpływa na efektywność Zespołu Developerskiego. Dzięki częstej i regularnej aktualizacji celów i zobowiązań wszyscy Developerzy skupiają się na priorytetowych zadaniach oraz stawiają wyżej płynny postęp prac zespołowych nad indywidualnymi rezultatami.

Problemy z Daily Scrum

Jednym z problemów związanych z Daily Scrum jest przeciąganie przez Developerów czasu spotkania.

W takim przypadku warto wprowadzić zasadę zapisywania na tablicy - fizycznej bądź wirtualnej - problematycznych kwestii, które nie są kluczowe dla Daily Scrum, ale są istotne dla Zespołu.

W ten sposób będzie można wrócić do problemów, które zostały do omówienia podczas nieformalnych rozmów w ciągu dnia, a także, jeśli to potrzebne, podczas Sprint Retrospective.

Innym problemem często pojawiającym się podczas Daily Scrums jest przekształcenie ich w spotkania podsumowujące poprzedni dzień pracy. Developerzy koncentrują się wówczas na omawianiu osiągniętych już rezultatów. Nie jest to dobra praktyka. Co prawda bieżąca orientacja Developerów w stanie zaawansowania prac prowadzących do realizacji Celu Sprintu jest bardzo istotna. Jednak poświęcanie Daily Scrum już zrealizowanym zadaniom nie sprzyja efektywności.

Pytania pomocnicze

Jeśli Zespół nie czerpie korzyści z Daily Scrum, Scrum Master może pomóc Developerom zidentyfikować problemy obserwując spotkanie pod kątem odpowiedzi na następujące pytania:

OBSZAR PROBLEMOWY	PYTANIE
Rozumienie Celu Sprintu	Czy Cel Sprintu jest jasny dla Developerów?
Uczestnictwo w spotkaniu	Czy każdy z Developerów ma możliwość swobodnej wypowiedzi?
Wzajemny szacunek	Czy Developarzy słuchają siebie nawzajem?
Współpraca zespołowa	Czy Developerzy pomagają sobie nawzajem?
Zaangażowanie w rozwiązywanie problemów	Czy Developerzy proponują różne sposoby rozwiązywania pojawiających się problemów?

5 Whys

Po wstępnej identyfikacji problemu, skuteczną techniką ustalenia przyczyny jego powstania może być metoda 5 Why nazywana też 5 Whys lub 5W autorstwa Sakichi Toyody. Polega ona na zadaniu kilku z rzędu pytań „Dlaczego?”. Pozwala to zdiagnozować głębszą przyczynę problemu, a tym samym łatwiej go rozwiązać. Przykładowo weźmy ostatnią pozycję z tabeli: problem pojawia się w obszarze zaangażowania w rozwiązywanie problemów przez Zespół Developerski. Pięć pytań może wyglądać następująco:

1 x WHY?

Q: *Dlaczego Developerzy nie proponują różnych sposobów rozwiązania pojawiających się problemów?*

A: *Ponieważ Developer Harry zawsze jako pierwszy proponuje jedno rozwiązanie.*

2 x WHY?

Q: *Dlaczego Developer Harry zawsze jako pierwszy proponuje jedno rozwiązanie?*

A: *Ponieważ nikt inny nie zabiera głosu.*

3 x WHY?

Q: *Dlaczego nikt inny nie zabiera głosu?*

A: *Ponieważ inni Developerzy nie mają ochoty szukać lepszych rozwiązań.*

4 x WHY?

Q: Dlaczego inni Developerzy nie mają ochoty szukać lepszych rozwiązań?

A: Ponieważ szukanie rozwiązań wymaga skupienia i łatwiej uznać rozwiązanie Harrego za wystarczająco dobre.

5 x WHY?

Q: Dlaczego uznali rozwiązanie Harrego za wystarczająco dobre?

A: Ponieważ nie są nagradzani za proponowanie alternatywnych rozwiązań, omówili już na początku spotkania swoje plany na dziś i myślą o rozpoczęciu pracy.

W takim przypadku problem braku zaangażowania w rozwiązywanie problemów można rozwiązać zmieniając porządek Daily Scrum i zaczynając od tego zagadnienia albo wymyślając system nagradzania najlepszego rozwiązania, na przykład wprowadzając symboliczną nagrodę dla autora największej ilości rozwiązań zaakceptowanych przez Zespół w danym Sprincie.

Sprint Planning

Sprint Planning jest jednym z Wydarzeń Scrum, które koncentruje się wokół User Stories umieszczonych na samej górze Backlogu Produktu. Innymi słowy, na tych, które są najbardziej szczegółowo opisane i przeznaczone do wykonania w najbliższym Sprincie. Rolę moderatora Sprint Planningu pełni zwykle Scrum Master. Przebieg podręcznikowo przeprowadzonego spotkania można streścić odpowiadając na trzy pytania:

1. Jaki jest nowy Cel Sprintu?
2. Co zostanie zrobione?
3. W jaki sposób zostanie to zrobione?

Przyjrzyjmy się bliżej, co powinny zawierać odpowiedzi na powyższe pytania.

1. Jaki jest nowy Cel Sprintu?

Rolą Product Ownera podczas Sprint Planningu jest przedstawienie uczestnikom spotkania Celu oraz składających się na niego zadań do wykonania. Rozpoczyna on spotkanie od sformułowania Celu Sprintu oraz uzasadnienia, dlaczego jest on wartościowy z punktu widzenia Klienta. Następnie otwiera dyskusję, w której mogą zabrać głos nie tylko członkowie Scrum Team, lecz również Interesariusze.

Podsumowując dyskusję, Product Owner podaje finalne brzmienie Celu Sprintu, do którego osiągnięcia będzie dążył cały Scrum Team oraz upewnia się, czy Cel jest zrozumiały dla wszystkich zainteresowanych osób.

2. Co zostanie zrobione?

Druga część Sprint Planningu koncentruje się na wyborze User Stories, które zostaną zrealizowane w nowym Sprincie, oraz na dyskusji nad ich uszczegółowieniem. **Jednym z najtrudniejszych zadań podczas Sprint Planningu może okazać się trafna estymacja ilości oraz pracochłonności zadań wybranych do wykonania.**

Im bardziej doświadczony jest Scrum Team, tym trafniej szacuje pracę możliwą do wykonania w jednym Sprincie. Zespół stosuje bowiem coraz bardziej skutecznie techniki estymacji. Aby przyspieszyć dojrzewanie Zespołu, ułatwić oraz ustandaryzować proces estymacji, wiele Scrum Teams stosuje specjalne metody. Są to przede wszystkim Planning Poker oraz Team Estimation Game.

3. W jaki sposób zostanie to zrobione?

Trzecia, najbardziej techniczna część Sprint Planningu koncentruje się na udzieleniu odpowiedzi na pytanie „W jaki sposób zostanie to zrobione?”. **Zespół Developerski proponuje w niej sposoby wykonania zadań wybranych do realizacji w drugiej części spotkania.** Nikt oprócz samych Developerów nie powinien narzucać sposobu realizacji zadań od strony technicznej.

Planowanie powinno uwzględniać nie tylko technologię wykonania, ale też przepływ pracy pomiędzy Developerami. Pozwoli to uniknąć zastoju w pracy [bottlenecks], które mogą powodować opóźnienia w realizacji zadań. Tak jak w przypadku sposobów wykonania zadań, również o podziale zadań pomiędzy poszczególnych Developerów decydują oni sami bez zewnętrznej ingerencji. Zwykle Zespół Developerski koncentruje się tutaj na podziale User Stories na mniejsze zadania. Optymalna długość realizacji zadania to jeden dzień roboczy. Rezultatem Sprint Planningu jest jasny i jednoznacznie sformułowany Cel Sprintu, a także szczegółowo opisane User Stories wybrane do realizacji z Backlogu Produktu. Wszystkie te elementy składają się na Backlog Sprintu.

Sprint Review

Sprint Review jest Wydarzeniem Scrum podsumowującym prace nad Produktem, które zostały ukończone podczas bieżącego Sprintu. Odbywa się ostatniego dnia Sprintu i jest otwarty dla Interesariuszy. Jego celem jest ocena Przyrostu, czyli prezentacja najnowszej wersji Produktu. Ważną częścią Sprint Review jest także dyskusja nad wprowadzonymi ulepszeniami i aktualizacjami, a także wprowadzenie niezbędnych zmian do Backlogu Produktu, dzięki czemu wszystkie zainteresowane osoby mogą zapoznać się z aktualnym stanem prac nad Produktem.

Sprint Review poświęcony jest Produktowi, natomiast jego celem jest inspekcja Przyrostu, czyli efektów pracy wykonanej w kończącym się właśnie Sprincie. Wydarzenie trwa maksymalnie cztery godziny. Biorą w nim udział wszyscy członkowie Scrum Team, a także Interesariusze, czyli wszystkie osoby zainteresowane postępami prac nad Produktem.

Rola Interesariuszy podczas Sprint Review

Podczas Sprint Review Scrum Team prezentuje Przyrost Interesariuszom. Podsumowuje przy tym wykonane zadania i odpowiada na szczegółowe pytania:

1. Kto wykonał dane zadanie?
2. Co konkretnie zostało zrobione?
3. W jakim celu zostało to zrobione?



Warto zapamiętać, że najważniejszą częścią Sprint Review jest dyskusja z Interesariuszami dotycząca Przyrostu, a także kierunku, w którym rozwijany jest Produkt. Interesariusze udzielają informacji zwrotnej członkom Scrum Team. Dzięki temu możliwa jest adaptacja, czyli dostosowanie sposobu działania Scrum Team do potrzeb i wizji Klienta. Ma to na celu maksymalne podniesienie wartości biznesowej Produktu. Udzielana na każdym Sprint Review informacja zwrotna jest szczególnie ważna w przypadku tworzenia innowacyjnych Produktów, które muszą być na bieżąco dopasowywane do działań konkurencji i potrzeb rynku.

Wydawanie Przyrostów

Nie powinno się traktować Sprint Review jako jedyne momentu, kiedy Scrum Team przekazuje Przyrost Klientowi. Jeśli jakaś funkcjonalność Produktu spełnia wcześniej Definicję Ukończenia, Product Owner może podjąć decyzję o jego natychmiastowym wydaniu. Możliwa jest również sytuacja, gdy jakiś element Backlogu Produktu nad którym pracował Scrum Team w danym Sprincie nie został ukończony i nie jest zgodny z Definicją Ukończenia. Nie może wtedy zostać wydany ani nawet przedstawiony podczas Sprint Review.

Praca nad Backlogiem Produktu podczas Sprint Review

Aktualizacja Backlogu Produktu jest równie ważną częścią Sprint Review jak prezentacja wyników pracy Interesariuszom. Zwykle aktualizacji Backlogu przeznaczona jest ostatnia część spotkania, więc Interesariusze nie muszą brać w niej udziału. Product Owner aktualizuje Backlog Produktu na podstawie informacji zwrotnych od Interesariuszy oraz wniosków wyciągniętych przez Zespół Developerski. Jest to szczególnie ważne, jeśli uzyskane informacje zwrotne mają wpływ na kształt i Cel kolejnego Sprintu. Aktualizacja Backlogu jest wtedy niezbędnym krokiem przygotowującym kolejny Sprint Planning.

Sprint Retrospective

Sprint Retrospective to wydarzenie podsumowujące Sprint, w którym mogą wziąć udział tylko członkowie Scrum Team. Dzięki temu może ono zostać w pełni poświęcone wewnętrznym sprawom zespołu. Sprint Retrospective służy bowiem przede wszystkim refleksji nad obecnymi metodami pracy, a także dyskusji nad propozycjami ich ulepszenia. Sprint Retrospective to spotkanie kończące każdy Sprint. Zgodnie z oficjalnym Przewodnikiem po Scrum, Sprint Retrospective zajmuje maksymalnie trzy godziny przy miesięcznym Sprincie lub odpowiednio krócej, jeśli Scrum Team pracuje w krótszych cyklach.

Cele i tematyka Sprint Retrospective

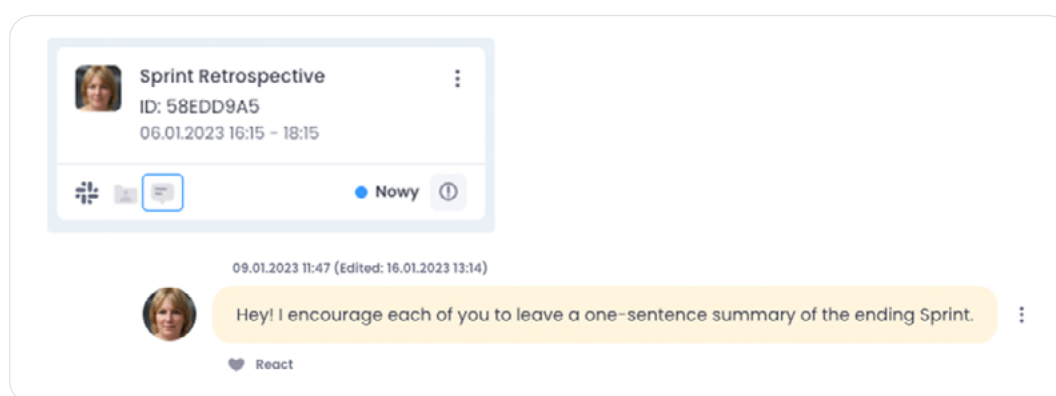
W Sprint Retrospective biorą udział wszyscy członkowie Scrum Team. **Celem spotkania jest dyskusja na temat problemów związanych z pracą Scrum Team i sposobów, w jaki radzi sobie on z ich rozwiązywaniem.** Nie są to jednak problemy dotyczące rozwijanego przez Scrum Team Produktu, tylko kwestie związane ze specyfiką i przebiegiem współpracy pomiędzy członkami Scrum Team. Z powodu tego, że poruszane kwestie są często delikatne i wrażliwe, Sprint Retrospective jest wydarzeniem zamkniętym. Jego cele można sformułować następująco:

- podsumowanie obecnych sposobów współpracy,
- wskazanie problemów i niedoskonałości wymagających do poprawy,
- propozycje rozwiązań i modyfikacji.

Cele Sprint Retrospective są ściśle związane z filarami empiryzmu, na których wspiera się Scrum. Dwa pierwsze punkty związane są z inspekcją. Natomiast ostatni - z adaptacją. Podczas Sprint Retrospective często pojawiają się następujące pytania: co nam się udało, co się sprawdziło, co można poprawić, co postanawiamy poprawić razem jako Zespół, co każdy z osobna postanawia poprawić w następnym Sprincie. Rezultatem odpowiedzi na powyższe spotkania jest nie tylko jasny obraz zasad współpracy Scrum Team dostępny dla wszystkich jego członków. Zespół podejmuje także zobowiązania dotyczące poprawy kooperacji i zachowań zespołowych, które zostaną wdrożone w kolejnym Sprincie.

Jak przeprowadzić skuteczną Sprint Retrospective?

Z powodu tego, że Sprint Retrospective jest trudnym spotkaniem, kluczowa jest w nim rola Scrum Mastera, który moderuje dyskusję. Najlepiej, aby zaproponował on członkom Scrum Team kolejne zabieranie głosu. Może na przykład poprosić każdego o jednozdaniowe podsumowanie kończącego się Sprintu.



Screen: System Firmbee - Komentarz do zadania

1. Problemy do omówienia

Jako że rozmawianie o problemach w zespole może budzić duże emocje, często stosowanym rozwiązaniem jest spisanie kwestii wymagających omówienia na osobnych karteczkach. Dzięki temu wyrażenie swojego zdania staje się łatwiejsze. Łatwiej też zauważyć większe problemowe obszary i kwestie, co do których ma zastrzeżenia większa ilość osób.

Jeśli problemów sformułowanych przez Scrum Team jest zbyt wiele, można rozpocząć od omówienia problemów, które wymieniło kilka osób albo też przegłosować, które kwestie są najważniejsze zdaniem Scrum Team. Problemy, na które nie wystarczyło czasu podczas Sprint Retrospective można przesunąć na kolejną retrospektywę. Oczywiście tylko w przypadku, jeśli wciąż one występują.

2. Dyskusja i podjęcie zobowiązań

Najważniejsze części Sprint Retrospective to jednak dyskusja oraz podjęcie zobowiązań.

Dyskusja powinna dotyczyć przyczyn problemów, momentów gdy się pojawiają, a także ich wpływu na funkcjonowanie Scrum Team. Warto zastanowić się, czy można uniknąć ich występowania i z jakimi osobami należy omówić ich rozwiązanie.

Podjęcie zobowiązań jest równie ważne jak diagnoza problemów, ponieważ sama świadomość ich istnienia oraz przyczyn nie przekłada się jednoznacznie na ich rozwiązanie. **Rezultatem Sprint Retrospective jest zwykle podjęcie kilku zobowiązań.** Jeśli problem dotyczy całego zespołu, często jeden z członków zespołu zostaje zobligowany do zwracania szczególnej uwagi na konkretny problem w kolejnym Sprincie oraz do zaproponowania jego rozwiązania, lub nawet rozwiązania samego problemu. Jeśli natomiast problem dotyczy działania konkretnej osoby, ona sama podejmuje zobowiązanie dotyczące zmiany swojego postępowania już w następnym Sprincie.

Częste błędy w czasie Retrospekcji

Błędy w Sprint Retrospective zdarzają się niestety bardzo często. To jedno bowiem z najtrudniejszych spotkań, którego skuteczne przeprowadzanie wymaga od zespołu dużej dojrzałości. Dlatego warto przyrzeć się problemom najczęściej pojawiającym się w innych zespołach, aby łatwiej dostrzec ich symptomy podczas prowadzenia Sprint Retrospective w swoim Scrum Team.

1. Niewystarczająca transparentność

Według Przewodnika po Scrum każdy z członków Scrum Team jest zobowiązany do szczerości i odważnego wyrażenia wątpliwości, a także do zgłaszania swoich uwag podczas Sprint Retrospective. Jednak w praktyce zobowiązanie do transparentności jest bardzo wymagające. Przez to często członkowie Scrum Team starają się je omijać. **Jednym z trudnych do dostrzeżenia i rozwiązania problemów jest unikanie dyskusji nad zaobserwowanymi niedociągnięciami w pracy Scrum Team.** Może to jednak w dłuższej perspektywie doprowadzić do znacznie poważniejszych problemów. Zadaniem Scrum Mastera jest więc uważne obserwowanie sytuacji w zespole, a także zachęcanie wszystkich jego członków do aktywności już od samego początku Sprint Retrospective.

2. Skupienie się na jednorazowych problemach lub sukcesach

Kolejnym problemem, który może pojawiać się podczas Sprint Retrospective jest przywiązywanie nie wystarczającej wagi do cyklicznych i powtarzalnych zachowań zespołowych, oraz ich wpływu na efektywność zespołu. Zawsze warto pogratulować członkom Scrum Team, jeśli odnieśli wyjątkowy sukces, jednak Sprint Review nie powinien być poświęcony jego świętowaniu. Tak samo rzecz się ma z porażkami. Jeśli coś się nie udało ze względów losowych lub zdiagnozowanego już błędu, nie warto nadmiernie analizować tego zdarzenia podczas Sprint Review.

Zdarza się jednak, że zespół poświęca dużą część Sprint Retrospective takim wydarzeniom. Celem Sprint Retrospective jest jednak szukanie sposobów usprawniania codziennej pracy zespołu. Dlatego spotkanie nie powinno obracać się wokół jednorazowych sukcesów lub problemów, które z dużym prawdopodobieństwem więcej się nie powtórzą.

3. Nadobecność Product Ownera

W wielu organizacjach pozycja Product Owner jest utożsamiana z pozycją Product Managera. Jest on wtedy często uważany za przełożonego Scrum Team. Z tego powodu zdarza się, że Zespół Developerski nie chce rozmawiać w jego obecności na temat problemów z pracą zespołową. Dlatego tak ważną kwestią jest budowanie wzajemnego zaufania pomiędzy Zespołem Developerskim i Product Ownerem.

Jednak proces budowania zaufania jest trudny i długotrwały. Dlatego czasem warto, by Product Owner zrezygnował z udziału w całości lub części Sprint Retrospective, aby zostawić reszcie zespołu przestrzeń na swobodną dyskusję.

4. Problemy z samodzielnym zarządzaniem

Samozarządzanie oznacza, że członkowie Scrum Team samodzielnie podejmują decyzję o tym, kto z nich będzie wykonywał określone zadania, kiedy i jak. Podczas Sprint Retrospective zespół dyskutuje na temat ludzi, ich interakcji i praktyk zespołowych. Decyduje więc, jakie problemy należy rozwiązać w najbliższym Sprincie, w jaki sposób to zrobić i kto będzie odpowiedzialny za podjęcie działania.

Jeśli w samodzielnym zespole pojawiają się poważniejsze problemy, w Scrum Team może pojawić się pokusa zrzeczenia się odpowiedzialności. Członkowie zespołu nie chcą brać udziału w dyskusji i starają się zepchnąć na kogoś odpowiedzialność za zarządzanie. Aby temu zapobiec, ogromnie ważna jest regularność omawiania nawet niewielkich problemów, aby zapobiec ich nagromadzeniu.

5. Zbyt wiele zobowiązań

Aktywny Scrum Team działający zgodnie z trzema filarami empiryzmu, przejrzystością, inspekcją i adaptacją, może napotkać na problem podejmowania zbyt wielu zobowiązań naraz. Jeśli zobowiązań podjętych przez Scrum Team podczas Sprint Retrospective jest zbyt wiele, istnieje spore ryzyko, że żadne zobowiązanie nie zostanie zrealizowane należycie, niektóre ze zobowiązań nie zostaną zrealizowane wcale lub wprowadzone zmiany nie będą stałe. Dlatego **dobrą praktyką jest podejmowanie nie więcej niż czterech usprawnień w każdym Sprincie. Pozwala to na stopniowe, ale skuteczne poprawianie efektywności zespołu.**

Gdzie zdobyć wiedzę i doświadczenie w Scrum?

Zapoznałeś się już z naszym przewodnikiem po Scrum, ale wciąż nurtują cię pytania?

Jeśli chcesz pogłębić swoją wiedzę na temat Scrum, a być może nawet zdobyć certyfikat, przedstawiamy Ci sprawdzone źródła, a także informacje na temat oficjalnej certyfikacji Scrum.

Podstawowym źródłem wiedzy na temat Scrum jest oczywiście *Oficjalny Przewodnik po Scrum* autorstwa Kena Schwabera i Jeffa Sutherlanda. Jest dostępny za darmo [pod tym linkiem](#). Jego ostatnia, najdoskonalsza wersja powstała w 2020 roku i została przetłumaczona na ponad 30 języków. Oficjalny przewodnik zawiera opis wszystkich podstawowych założeń Scrum oraz wyjaśnienie niezbędnych pojęć w zwięzłej, minimalistycznej formie.

1. Wiedza na temat Scrum

Lista przewodników i poradników dotyczących pracy w Scrum powiększa się z roku na rok. Można znaleźć wśród nich zarówno książki przeznaczone dla początkujących, jak i omawiające szczegółowe problemy, na które napotykają osoby pracujące jako Scrum Master czy Product Owner. Zdecydowaliśmy się wspomnieć tutaj o trzech ogólnych i najczęściej polecanych książkach na temat Scrum:

- 🟡 **Scrum: The Art of Doing Twice the Work in Half the Time** - Ta pełna przykładów książka autorstwa Jeffa Sutherlanda koncentruje się wokół tematu zmieniania świata, a w szczególności naszego codziennego otoczenia - miejsca pracy. Podkreśla, że Scrum sprawdza się nie tylko w biznesie, ale również w edukacji czy przy rozwiązywaniu problemów społecznych i osobistych. Innymi słowy, rodzaj realizowanego projektu często nie jest tak ważny, jak sposób myślenia o jego realizacji. Kluczowa jest także umiejętność dostosowywania się do zmiennych warunków i uczenia na błędach.
- 🟡 **A Scrum Book. The Spirit of the Game** - Jednym ze współautorów tej książki jest również Sutherland. Jej głównym celem jest opisanie wzorców Scrum. Dzięki ich znajomości można zarówno łatwiej dostrzec prawidłowości podczas realizacji kolejnych projektów w Scrum, jak i szybciej wprowadzać usprawnienia w projektach.
- 🟡 **Scrum Mastery. From Good to Great Servant Leadership** - Książka Geoffa Wattsa to praktyczny przewodnik opisujący sposoby pracy nad doskonaleniem umiejętności niezbędnych do pełnienia roli Scrum Mastera. Kładzie też duży nacisk na opis trudnych realiów pracy w Scrum w organizacjach, które dotąd stosowały tradycyjne metody zarządzania.

Nieocenione źródło wiedzy, lecz także możliwość dyskusji na temat praktycznych zastosowań Scrum stanowi forum dostępne na stronie [Scrum.org](https://www.scrum.org). Udzielają się na nim osoby, które dopiero się uczą i szukają odpowiedzi na praktyczne pytania, na które nie potrafią udzielić odpowiedzi na podstawie Scrum Guide, a także doświadczeni praktycy Scrum chętni do podzielenia się swoją wiedzą na temat pracy ze Scrum Team, wdrażania Scrum w organizacji, czy szczegółów dotyczących pracy Product Ownera lub Scrum Mastera.

Osoby zainteresowane stosowaniem zasad Scrum w większych organizacjach, powinny zajrzeć do wersji [Scrum@Scale Guide](#), który przygotowali również Kena Schwaber i Jeff Sutherland.

Tak jak Oficjalny Przewodnik po Scrum, ten podręcznik również został przetłumaczony na wiele języków. Można w nim znaleźć nie tylko zasady skalowania Scrum, lecz także szczegółowy opis korzyści płynących z zastosowania Scrum w większych organizacjach.

Jeśli interesuje Cię, jak Scrum wypada na tle innych praktyk Agile, możesz sięgnąć po [State of Agile Report](#). Jest to jeden z największych raportów na temat Agile, który jest przygotowywany corocznie od 2006 roku. Dowiesz się z niego między innymi, jakie techniki i praktyki Agile są stosowane w firmach. A także jak wygląda i jak szybko przebiega rozprzestrzenianie się technik pracy takich jak Scrum z branży IT na cały świat biznesu.

2. Szkolenia i certyfikaty Scrumowe

Zdobywanie wiedzy na temat Scrum może okazać się łatwiejsze dzięki kursom online. Jeśli twoja organizacja chętnie wyśle Cię na szkolenie, są również organizowane zajęcia stacjonarne. Najlepszym źródłem wiedzy są certyfikowane szkolenia Scrum dostępne [tutaj](#). Umożliwiają one zdobycie pogłębionej wiedzy na temat praktykowania w organizacji zasad Scrum.

Można odbyć szkolenie w zakresie wdrażania Scrum, lub wziąć udział w kursie kończącym się profesjonalnym certyfikatem Scrum Mastera lub Product Ownera. Scrum.org prowadzi również szkolenia dotyczące łączenia zasad Scrum z zastosowaniem tablic Kanban, a także dotyczące skalowania Scrum czy jego połączenia z projektowaniem doświadczenia użytkownika (UX).

Po ukończeniu szkolenia można uzyskać cenione certyfikaty, tak istotne w CV osób starających się o wyższe stanowiska. Jako potwierdzenie profesjonalnych, praktycznych umiejętności można uzyskać następujące certyfikaty: Scrum Mastera, Product Ownera, Scrum Developera, a także Profesjonalisty w dziedzinie wdrażania Scrum@Scale. Biorąc pod uwagę efektywność pracy w Scrum, a także przełożenie praktycznych umiejętności jego wdrażania na zarobki, warto rozważyć uzyskanie profesjonalnego certyfikatu.



Poznaj nas bliżej



Firmbee

