Kanban i Scrum - jak uzyskać najlepsze z obu

Henrik Kniberg i Mattias Skarin
Wstęp: Mary Poppendieck i David Anderson

C4Media, wydawca InfoQ.com.

Książka należy do serii InfoQ Enterprise Software Development.

W celu uzyskania informacji o nabyciu tej lub innych książek wydanych przez InfoQ, skontaktuj się z: books@c4media.com.

Redaktor Naczelny: Diana Plesa
Okładka: Bistrian IOSIP
Skład: Accurance

Numer katalogowy:
ISBN: 978-0-557-13832-6

Przekład na język polski:

Spis treści

Wstęp Mary Poppendieck

Henrik Kniberg jest jedną z tych nielicznych osób, które potrafią wyodrębnić esencję bardzo skomplikowanej sytuacji, odróżnić główne idee od szumu informacyjnego oraz przedstawić jasne i łatwe do zrozumienia wyjaśnienie. W tej książce Henrik doskonale objaśnia różnice między Scrumem i Kanbanem. Uzmysławia czytelnikom, że są to tylko narzędzia i że najważniejsze to mieć pod ręką pełny zestaw narzędzi, zrozumieć mocne i słabe strony każdego z nich i znać sposoby ich wykorzystywania.

Dzięki lekturze tej książki dowiesz się, czym jest Kanban, poznasz jego zalety i ograniczenia oraz nauczysz się z niego korzystać. Uzyskasz również wiedzę na temat tego, jak i kiedy zmodyfikować Scrum albo dowolne inne narzędzie, z którego przyjdzie ci korzystać. Henrik pokazuje, że najważniejsze nie jest narzędzie, od którego zaczynasz, lecz ciągłe ulepszanie sposobów wykorzystywania tego narzędzia i powiększanie zasobu posiadanych narzędzi.

W drugiej części książki Mattias Skarin sprawia, że książka jest jeszcze bardziej efektywna, opowiadając o wdrożeniu Scruma i Kanbana w rzeczywistej firmie. Zapoznasz się w niej z przykładem wykorzystania tych narzędzi, zarówno osobno, jak i łącznie, do udoskonalenia procesu tworzenia oprogramowania. Zauważysz, że nie ma jednego najlepszego sposobu ich implementacji – musisz samodzielnie dojść do odpowiedniego rozwiązania, w zależności od twojej sytuacji. Niech będzie to twój następny krok w kierunku lepszego procesu tworzenia oprogramowania.

Mary Poppendieck

Wstęp Davida Andersona

Kanban bazuje na bardzo prostym pomyśle: ilość pracy w toku (Work in Progress – WIP) należy ograniczać, a nową pracę można rozpoczynać dopiero wówczas, kiedy aktualnie wykonywana praca zostanie ukończona lub anulowana. Kanban (lub karta sygnałowa) sprawia, że pojawia się wizualny sygnał oznajmiający, iż można rozpocząć nową pracę, gdy aktualna liczba zadań nie jest równa limitowi. Nie brzmi to zbyt rewolucyjnie i z pozoru nie ma znaczącego wpływu na wydajność, kulturę, możliwości i dojrzałość zespołu i organizacji. Ale to tylko pozory! Kanban wydaje się być niewielką zmianą, a jednak zmienia wszystko w biznesie.

Uświadomiliśmy sobie, że Kanban jest podejściem do zmiany sposobu zarządzania. Nie jest cyklem życia oprogramowania ani procesem zarządzania projektami. Kanban jest podejściem do wprowadzania zmian w istniejącym cyklu życia oprogramowania lub procesie zarządzania projektami. Główną zasadą Kanbana jest to, że zaczynasz od tego, co masz w danej chwili. Poprzez mapowanie łańcucha wartości i ustalenie limitów WIP dla każdego stadium procesu będziesz w stanie zrozumieć proces. Potem prace zaczynają płynąć przez system – są wciągane, kiedy wygenerowane zostaną wizualne sygnały Kanban.

Kanban jest użyteczny dla zespołów pracujących w metodyce Agile, ostatnio jednak zyskuje również popularność wśród zespołów wykorzystujących bardziej tradycyjne podejście. Kanban jest przedstawiany jako część inicjatywy Lean pozwalająca zmieniać kulturę organizacji i zachęcać do ciągłego usprawniania procesu.

Ponieważ w systemie Kanban WIP jest ograniczony, prace zablokowane będą powodowały, że system również będzie się zapełniał. Jeżeli zablokowanych zostanie wystarczająco dużo prac, cały proces zostanie

zatrzymany. Dzięki temu cały zespół skupi się na rozwiązaniu problemu, aby odblokować zadanie i wznowić przepływ.

Kanban wykorzystuje mechanizm kontroli wizualnej pozwalający śledzić przepływ pracy poprzez poszczególne fazy. Zazwyczaj jest to tablica z samoprzylepnymi karteczkami lub system elektroniczny. Najlepiej jest korzystać z obu. Przejrzystość tego podejścia sprzyja zmianom w kulturze organizacyjnej. Metody Agile są dobre w obrazowaniu WIP, zakończonej pracy i pomiarów raportowych takich jak szybkość (ilość pracy wykonanej w trakcie iteracji). Kanban idzie jednak o krok dalej i zapewnia przejrzystość procesu i przepływu. Ujawnia wąskie gardła, kolejki, odchylenia i marnotrawstwo – wszystko to ma wpływ na wydajność organizacji, ilość dostarczonej wartościowej pracy i czas cyklu potrzebny do jej dostarczenia. Kanban zapewnia członkom zespołu i osobom decyzyjnym widoczność efektu ich pracy (lub interakcji). Studia przypadku pokazują, że Kanban zmienia zachowania i zachęca do współpracy. Widoczność i wpływ na wąskie gardła, marnotrawstwo i odchylenia zachęca również do dyskusji o usprawnieniach, a zespoły szybko zaczynają te usprawnienia wdrażać.

W rezultacie Kanban zachęca do stopniowych zmian istniejącego procesu i do ewolucji pasującej do założeń Lean i Agile. Kanban nie wprowadza rewolucji w pracy ludzi, lecz skłania do ewolucyjnej zmiany. Jest to zmiana wynikająca ze zrozumienia jej potrzeby i oparta na porozumieniu pracowników i ich współpracowników.

Dzięki naturze systemu opartego na wciąganiu pracy Kanban zachęca także do priorytetyzacji nowych prac i kończenia istniejących zadań. Zazwyczaj zespoły ustalają kadencję spotkań z osobami decyzyjnymi w sprawie ustalenia priorytetów i kolejności wykonywania prac. Spotkania te mogą być organizowane często, ponieważ zwykle są bardzo krótkie. Wystarczy odpowiedzieć na bardzo proste pytanie, na przykład: „Od ostatniego spotkania zwolniły się dwa sloty. Nasz aktualny czas cyklu dostarczenia wynosi sześć tygodni. Jakie dwie rzeczy należy dostarczyć za sześć tygodni?”. Ma to dwojaki efekt: przeważnie pozwala uzyskać szybką odpowiedź i sprawić, aby spotkanie było krótkie. Natura pytania sprawia, że zobowiązanie dotyczące tego, nad czym zespół będzie pracował, odroczone jest do ostatniej chwili. Dzięki zarządzaniu oczekiwaniami zespół zwiększa elastyczność, skraca czas cyklu od

deklaracji do dostarczenia i eliminuje konieczność wielokrotnego wykonywania tej samej pracy wskutek zmiany priorytetów.

Wysiłek ograniczania WIP owocuje przewidywalnością czasu cyklu i większą regularnością wdrożeń. Podejście „zatrzymać linię" wobec przeszkód i błędów wprowadza wyższą jakość i brak konieczności powtórnego wykonywania tej samej pracy.

To wszystko stanie się jasne za sprawą tej wspaniałej książki i prostych wyjaśnień w niej zawartych, niejasne jednak pozostanie to, jak do tego doszliśmy. Kanban nie został wymyślony w jedno popołudnie dzięki temu, że ktoś doznał olśnienia – powstawał przez kilka lat. Wielu spośród skutków psychologicznych i socjologicznych zmieniających kulturę, zdolność i dojrzałość organizacji nawet nie przewidywano – zostały po prostu odkryte. Wiele spośród skutków powstania Kanbana jest sprzecznych z intuicją. To, co wydaje się podejściem bardzo mechanicznym – ograniczanie WIP i wciąganie prac – w rzeczywistości ma ogromny wpływ na ludzi oraz ich interakcje i współpracę. Ani ja, ani nikt inny, kto zajmował się Kanbanem we wczesnym etapie jego powstawania, nie przewidywaliśmy takich rezultatów.

Zajmowałem się tym, co później nazwane zostało Kanbanem, jako sposobem wprowadzania zmian napotykającym jak najmniejszy opór. Było to dla mnie jasne już w 2003 roku. Zająłem się nim również ze względu na korzyści mechaniczne. Przekonałem się wówczas dzięki wdrażaniu technik Lean, że jeżeli zarządzanie WIP ma sens, to jego ograniczanie tym bardziej, ponieważ w ten sposób staje się ono łatwiejsze. Dlatego w 2004 roku zdecydowałem się zaimplementować system polegający na wciąganiu prac. Miałem ku temu okazję, kiedy menadżer w firmie Microsoft poprosił mnie o pomoc w zarządzaniu zmianą w ich zespole zajmującym się organizacją aktualizacji dla aplikacji wewnętrznych. Pierwsza implementacja bazowała na systemie o nazwie Drum–Buffer–Rope (Bęben–Bufor–Lina), będącym odpowiedzią na teorię ograniczeń. Rozwiązanie okazało się wielkim sukcesem: czas cyklu skrócił się o 92%, przepustowość zwiększyła się prawie trzykrotnie, a przewidywalność (daty dostarczenia) na poziomie 98% była bardzo akceptowalna.

W 2005 roku Donald Reinertsen przekonał mnie do implementacji pełnowartościowego systemu Kanban. Miałem taką szansę w 2006 roku, kiedy przejąłem zespół tworzący oprogramowanie w Corbis w Seattle. W 2007 roku zacząłem raportować rezultaty. Pierwsza prezentacja odbyła się na konferencji Lean New Product Development Summit w Chicago w maju 2007 roku. Kolejna prezentacja miała miejsce na spotkaniu open space na konferencji Agile 2007 w Waszyngtonie w sierpniu tego samego roku. Wzięło w nim udział dwadzieścia pięć osób, a trzy z nich były z Yahoo!: Aaron Sanders, Karl Scotland i Joe Arnold. Wrócili do Kalifornii, Indii i Anglii i zaimplementowali Kanban w swoich zespołach, które miały pewne problemy ze Scrumem. Uruchomili również grupę dyskusyjną poświęconą Kanbanowi, która w chwili, gdy piszę ten tekst, ma prawie ośmiuset członków. Kanban zaczął się rozprzestrzeniać, a pierwsze osoby, które go wdrażały, dzieliły się swoimi doświadczeniami.

Teraz, w roku 2009, stosowanie Kanbana jest coraz powszechniejsze i pojawia się coraz więcej informacji od praktyków. W ciągu ostatnich pięciu lat dowiedzieliśmy się o Kanbanie bardzo dużo i dowiadujemy się więcej każdego dnia. W mojej pracy skupiam się na korzystaniu z Kanbana, na pisaniu o Kanbanie, na mówieniu o Kanbanie i na myśleniu o Kanbanie, aby móc go lepiej zrozumieć i łatwiej tłumaczyć innym. Specjalnie powstrzymałem się od porównywania Kanbana do innych metod Agile, chociaż w 2008 roku poświęciłem trochę czasu na wyjaśnienie, dlaczego Kanban powinien być uznany za podejście kompatybilne z Agile.

Innym, posiadającym większe doświadczenie osobom pozostawiam znalezienie odpowiedzi na pytania takie jak: czym się różni Kanban od Scruma? Bardzo się cieszę, że Henrik Kniberg i Mattias Skarin podjęli się próby odpowiedzi na to pytanie. Ty, będąc osobą pracująca na tym polu, potrzebujesz informacji, które pomogą ci podejmować decyzje i robić postępy w pracy. Henrik i Mattias odpowiadają na te potrzeby w sposób dla mnie nieosiągalny. Szczególne wrażenie zrobiło na mnie wnikliwe, obiektywne i rzeczowe porównanie tych dwóch metod. Ilustracje Henrika są szczególnie obrazowe i często oszczędzają czytania wielu stron tekstu. Studium przypadku Mattiasa jest ważne, ponieważ dowodzi, że Kanban to znacznie więcej niż tylko teoria, i na przykładzie pokazuje, jak Kanban może się przydać w twojej organizacji.

Mam nadzieję, że spodoba ci się to porównanie Scruma i Kanbana i że dzięki tej książce uzyskasz pełniejszą wiedzę o Agile, a zwłaszcza o Scrumie i Kanbanie. Jeżeli chcesz dowiedzieć się więcej, odwiedź stronę naszej społeczności, The Limited WIP Society: http://www.limitedwipsociety.org/.

David J. Anderson

Sequim, Washington, USA
8 lipca 2009

Wprowadzenie

Normalnie nie piszemy książek. Wolimy spędzać czas w okopach, pomagając klientom optymalizować, debugować i refaktoryzować ich proces i organizację. Zauważyliśmy ostatnio wyraźny trend i chcielibyśmy podzielić się pewnymi przemyśleniami. Oto typowy przykład:

- **Kuba:** „Wreszcie udało nam się wdrożyć Scruma!”
- **Franek:** „I jak?”
- **Kuba:** „Cóż, jest znacznie lepiej niż wcześniej…”
- **Franek:** „…ale? ”
- **Kuba:** „…ale wiesz, jesteśmy zespołem zajmującym się wsparciem i utrzymaniem. ”
- **Franek:** „No i? ”
- **Kuba:** „No i podoba nam się to całe sortowane priorytetów w rejestrze, samoorganizujące się zespoły, codzienne spotkania, retrospektywy itepe. ”
- **Franek:** „To w czym problem? ”
- **Kuba:** „Nasze sprinty wciąż kończą się porażką. ”
- **Franek:** „Dlaczego? ”
- **Kuba:** „Bo ciężko nam zadeklarować dwutygodniowy plan. Iteracje nie mają u nas zbyt wielkiego sensu, pracujemy po prostu

nad tym, co jest dla nas najbardziej pilne w danej chwili. Może powinniśmy mieć jednotygodniowe iteracje? ”

- **Franek:** „A będziecie w stanie zadeklarować jednotygodniową pracę? Czy będziecie mogli się skupić i pracować w spokoju przez tydzień? ”
- **Kuba:** „Raczej nie, są rzeczy, które wyskakują z dnia na dzień. Może powinniśmy mieć jednodniowe sprinty…”
- **Franek:** „Czy wasze prace trwają mniej niż dzień? ”
- **Kuba:** „Nie, czasami trwają nawet kilka dni. ”
- **Franek:** „Więc jednodniowe sprinty też nie będą dobre. Nie myśleliście o całkowitym porzuceniu sprintów? ”
- **Kuba:** „Chętnie byśmy to zrobili, ale czy nie kłóci się to z ideą Scruma? ”
- **Franek:** „Scrum to tylko narzędzie. To ty decydujesz, kiedy i jak go używać. Nie bądź jego niewolnikiem! ”
- **Kuba:** „No to co powinniśmy zrobić? ”
- **Franek:** „Słyszałeś o Kanbanie? ”
- **Kuba:** „A co to jest? Czym się różni od Scruma? ”
- **Franek:** „Masz, przeczytaj tę książkę! ”
- **Kuba:** „Ale cała reszta elementów Scruma mi się podoba. Czy muszę wszystko zmieniać? ”
- **Franek:** „Nie, możesz połączyć obie techniki! ”
- **Kuba:** „Poważnie? A jak? ”
- **Franek:** „Poczytaj…”

Cel tej książki

Jeżeli interesujesz się wytwarzaniem oprogramowania w metodykach Agile, prawdopodobnie słyszałeś o Scrumie, a być może również o Kanbanie. Pytanie, które słyszymy coraz częściej, brzmi: co to jest ten Kanban i czym się różni od Scruma? Ludzie chcą wiedzieć, czy są one kompatybilne i czy występują między nimi jakieś konflikty?

Celem tej książki jest rozjaśnienie sytuacji i ułatwienie zrozumienia jak Kanban i Scrum mogą się przydać w twoim środowisku.

Daj znać, czy nam się udało!

Część I – Porównanie

Pierwsza część tej książki to próba stworzenia obiektywnego i praktycznego porównania Scruma i Kanbana. Jest to nieco zmodyfikowana wersja artykułu Kanban vs. Scrum z kwietnia 2009 roku. Artykuł cieszył się popularnością, dlatego postanowiłem zamienić go w książkę i poprosiłem mojego przyjaciela Mattiasa, aby urozmaicił ją studium przypadku jednego z naszych klientów. To kawał świetnej roboty! Jeżeli chcesz rozpocząć od studium przypadku, przejdź od razu do części drugiej – nie obrażę się.

No może trochę.

/Henrik Kniberg

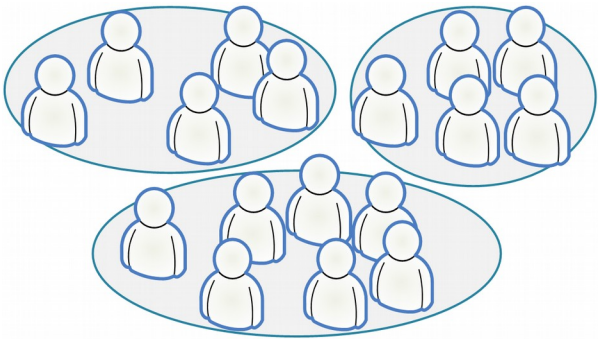
1

Czym właściwie są Scrum i Kanban?

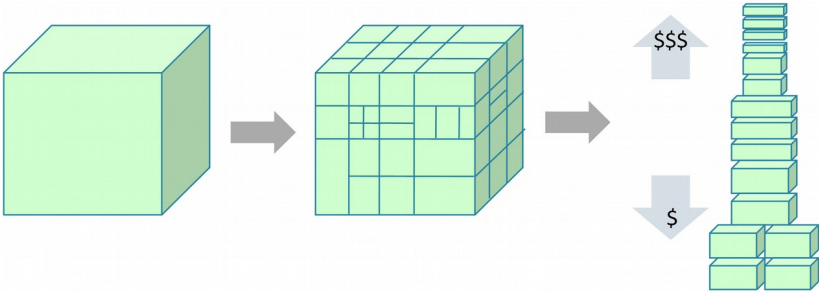
Spróbujmy podsumować Scrum i Kanban w mniej niż stu słowach każdy.

Scrum w skrócie

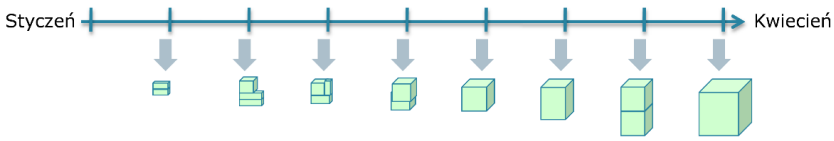
- **Podziel swoją organizację** na małe, wielofunkcyjne, samoorganizujące się zespoły.



- **Podziel swoją pracę na małe, konkretne podzadania**, a następnie posortuj listę według priorytetu i oszacuj relatywny nakład pracy potrzebny do ukończenia każdego zadania.



- **Podziel czas** na krótkie iteracje o ustalonej długości (zazwyczaj 1 – 4 tygodnie) zakończone powstaniem kodu gotowego do wdrożenia.



- **Zoptymalizuj plan dostarczenia** i we współpracy z klientem zaktualizuj priorytety na podstawie informacji uzyskanych w wyniku analizy oprogramowania dostarczonego w poprzednich iteracjach.

- **Zoptymalizuj proces** poprzez zorganizowanie retrospektywy po każdej iteracji.

Zamiast więc kazać **dużej grupie** poświęcić **dużo czasu** na budowę **dużej rzeczy** mamy **mały zespół**, który spędzi **mało czasu** na budowaniu **małej rzeczy**. Kluczem jest jednak **regularna integracja**, dzięki której nie zatracisz oglądu całości.

119 słów – nie najgorzej.

Więcej informacji znajdziesz w książce Scrum and XP from the Trenches, która dostępna jest za darmo w internecie pod adresem http://www.crisp.se/ScrumAndXpFromTheTrenches.html. Znam autora, to świetny facet :o)

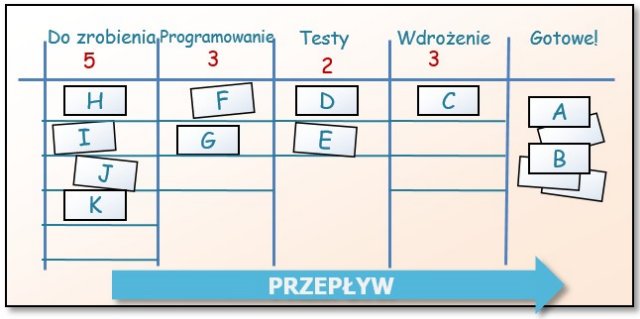
Więcej materiałów dotyczących Scruma znajdziesz pod adresem http://www.crisp.se/scrum.

Kanban w skrócie

- **Zwizualizuj przepływ pracy**
    - Podziel pracę na części, zapisz każdą część na karteczce i przyklej na tablicy.
    - Wykorzystaj odpowiednio nazwane kolumny do zilustrowania, na jakim etapie przepływu znajdują się poszczególne zadania.

- **Ogranicz pracę w toku (Work in Progress – WIP)**, przypisując konkretne ograniczenia liczby zadań, które mogą znajdować się w poszczególnych stanach przepływu.

- **Zmierz czas dostarczenia** (średni czas ukończenia jednego zadania nazywany jest również czasem cyklu), zoptymalizuj proces tak, aby czas dostarczenia był jak najkrótszy i jak najbardziej przewidywalny.



Użyteczne materiały dotyczące Kanbana zbieramy pod adresem http://www.crisp.se/kanban.

2

Jak Scrum i Kanban mają się do siebie?

Scrum i Kanban to narzędzia procesu

Narzędzie = coś, co jest wykorzystywane do wykonania zadania lub osiągnięcia celu.

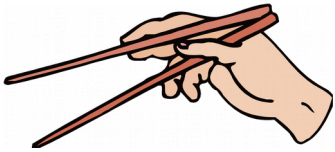
Proces = sposób wykonywania pracy.

Scrum i Kanban to narzędzia procesowe, ponieważ pomagają ci pracować bardziej efektywnie, instruując cię – do pewnego stopnia – co masz robić. Java to również narzędzie – ułatwia ci pisanie programów. Szczoteczka do zębów to także narzędzie – ułatwia ci czyszczenie zębów.

Porównuj narzędzia, aby zrozumieć, a nie oceniać

Nóż czy widelec – które narzędzie jest lepsze?

Bezsensowne pytanie, prawda? Odpowiedź zależy od sytuacji. Do jedzenia pulpetów prawdopodobnie bardziej przyda się widelec. Aby pokroić pieczarki, raczej wybierzesz nóż. Do wybijania rytmu na stole dobry będzie i jeden, i drugi. Jedząc stek, będziesz jednak potrzebować obu. Do ryżu… no cóż, jedni wolą widelec, a inni pałeczki.



Porównując narzędzia, musimy być ostrożni. Porównuj, aby zrozumieć, a nie oceniać.

Żadne narzędzie nie jest kompletne i doskonałe

Jak wszystkie narzędzia, Scrum i Kanban nie są ani doskonałe, ani kompletne. Nie mówią wszystkiego, co musisz zrobić, zapewniają tylko pewne ramy i wskazówki. Na przykład Scrum wprowadza iteracje ograniczone czasowo i zespoły wielofunkcyjne, a Kanban wnosi widoczne tablice i ogranicza rozmiar kolejek.

Co ciekawe, wartość narzędzia tkwi w tym, że ogranicza twoje możliwości. Narzędzie procesowe, które pozwala ci robić wszystko, nie jest użyteczne. Możemy nazwać ten proces „rób, co chcesz", albo lepiej „rób, co trzeba". Proces „rób, co trzeba" zawsze zadziała, jest idealny! Bo jeżeli nie zadziałał, to znaczy, że ewidentnie nie trzymałeś się procesu :o)

Wykorzystanie odpowiednich narzędzi pomoże ci w osiągnięciu sukcesu, ale go nie zagwarantuje. Łatwo pomylić sukces/porażkę projektu z sukcesem/porażką narzędzia.

- Projekt może zakończyć się sukcesem dzięki dobremu narzędziu.
- Projekt może zakończyć się sukcesem pomimo złego narzędzia.
- Projekt może zakończyć się porażką z powodu złego narzędzia.
- Projekt może zakończyć się porażką pomimo dobrego narzędzia.

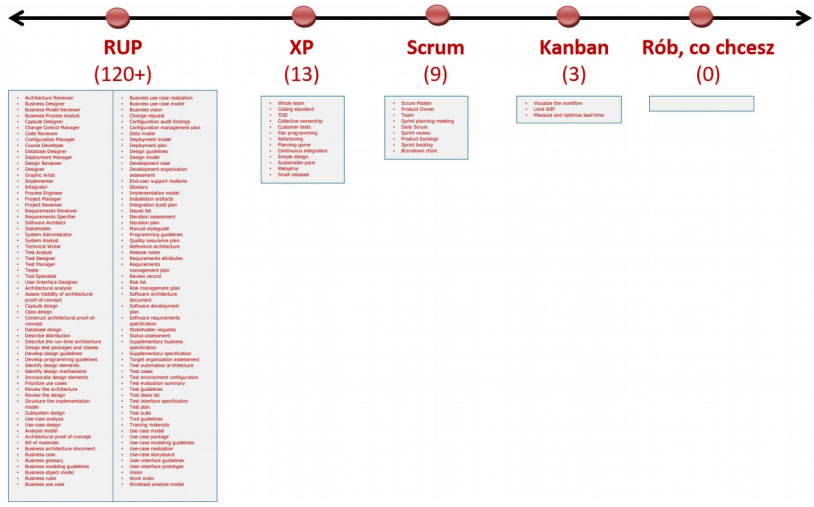
Scrum jest bardziej nakazowy niż Kanban

Możemy porównać narzędzia poprzez pryzmat tego, ile reguł narzucają. Nakazowy oznacza „więcej reguł, których trzeba przestrzegać", a adaptacyjny oznacza „mniej reguł, których trzeba przestrzegać". Proces w stu procentach nakazowy nie pozwala ci korzystać z mózgu, natomiast proces w stu procentach adaptacyjny pozwala ci robić, co chcesz, gdyż nie ma w nim żadnych reguł ani ograniczeń. Jak widzisz, oba ekstrema są raczej absurdalne.

Metody zwinne (Agile) czasami nazywa się metodami lekkimi, ponieważ są mniej nakazowe niż metody tradycyjne. Pierwszy punkt manifestu Agile brzmi: „Ludzie i interakcje ponad procesami i narzędziami”.

Scrum i Kanban są wysoce adaptacyjne, ale Scrum jest relatywnie bardziej nakazowy niż Kanban. Scrum narzuca więcej ograniczeń i tym samym daje mniej możliwości. Na przykład Scrum nakazuje wykorzystanie iteracji ograniczonych czasowo, Kanban nie.

Porównajmy jeszcze kilka narzędzi procesowych z punktu widzenia stopnia ich nakazowości:



RUP jest bardzo nakazowy: opisuje ponad 30 ról, ponad 20 działań i ponad 70 artefaktów – ogromna liczba rzeczy do nauczenia się. Nie masz jednak korzystać z nich wszystkich, lecz musisz wybrać podzbiór elementów odpowiedni dla twojego projektu. Niestety w praktyce jest to trudne: „Hmmm… czy będziemy potrzebowali artefaktu ‘Wnioski z audytu konfiguracji’? Czy będziemy potrzebowali roli Menadżer historii zmian? Nie jestem pewien, więc zachowajmy je na wszelki wypadek”. Może to być jeden z powodów, dla których implementacje RUP są zazwyczaj znacznie bardziej rozbudowane od metod zwinnych takich jak Scrum i XP.

XP (eXtreme Programming – programowanie ekstremalne) jest bardziej nakazowe niż Scrum. Zawiera większość elementów Scruma i kilka dość szczegółowych praktyk tworzenia oprogramowania, takich jak TDD (test-driven development – programowanie sterowane testami) i programowanie w parach.

Scrum jest mniej nakazowy niż XP, ponieważ nie określa żadnych praktyk tworzenia oprogramowania. Jest jednak bardziej nakazowy od Kanbana, nakazuje bowiem takie elementy jak iteracje i zespoły wielofunkcyjne.

Jedną z głównych różnic między Scrumem i RUP jest to, że w RUP otrzymujesz zbyt dużo i powinieneś usunąć elementy, których nie potrzebujesz, a w Scrumie dostajesz zbyt mało i musisz dodać elementy, których brakuje.

Kanban pozostawia prawie wszystko w twoich rękach. Jedyne zasady to konieczność wizualizacji przepływu pracy i ograniczania WIP. Bardzo blisko do „rób, co chcesz", ale metoda ta jest zadziwiająco potężna.

Nie ograniczaj się do jednego narzędzia!

Łącz narzędzia zgodnie z własnymi potrzebami! Nie mogę sobie wyobrazić sprawnie działającego zespołu Scruma, który nie wykorzystuje większości elementów Scruma. Wiele zespołów Kanbana korzysta z codziennych spotkań (będących praktyką Scruma). Niektóre zespoły Scruma zapisują swoje elementy w rejestrze produktu w postaci przypadków użycia (praktyka RUP) lub ograniczają wielkości kolejek (praktyka Kanbana). Wykorzystaj to, co działa w twoim przypadku.

Bardzo dobrze to ujął Miyamoto Musashi, samuraj żyjący w XVII wieku, znany ze swojej techniki walki dwoma mieczami:

> Nie przywiązuj się do jednej broni ani do jednej szkoły walki.
>
> – Miyamoto Musashi

Musisz jednak zwracać uwagę na ograniczenia każdego z narzędzi. Na przykład jeżeli używasz Scruma i postanowisz przestać korzystać z iteracji (lub z innego podstawowego aspektu Scruma), nie możesz mówić, że stosujesz metodykę Scrum. Scrum sam w sobie jest wystarczająco minimalistyczny, więc jeżeli zaczniesz usuwać jego elementy i nadal

będziesz nazywać to Scrumem, nazwa ta stanie się myląca i nieadekwatna. Nazywaj taką metodę „inspirowaną Scrumem" lub „podzbiorem Scruma" albo chociaż „scrumopodobną" :o)

3

Scrum nakazuje role

Scrum nakazuje trzy role: właściciel produktu (Product Owner; określa wizję produktu i priorytety), zespół (Team; implementuje produkt) i mistrz Scruma (Scrum Master; usuwa przeszkody i jest liderem procesu).

Kanban nie określa żadnych ról.

Nie oznacza to, że korzystając z Kanbana, nie możesz albo nie powinieneś mieć właściciela produktu! Znaczy to jedynie, że nie musisz. Zarówno w Scrumie, jak i w Kanbanie możesz ustanawiać takie role, jakich potrzebujesz.

Dodając role, bądź jednak ostrożny, musisz bowiem mieć pewność, że nowe role mają wartość i nie wchodzą w konflikt z innymi elementami procesu. Czy jesteś pewien, że potrzebujesz roli menadżera projektu? W przypadku dużego projektu może to być świetny pomysł; być może jest to człowiek, który pomaga synchronizować wiele zespołów i właścicieli produktu. W małym projekcie ta rola może być niepotrzebna lub, co gorsza, może prowadzić do suboptymalizacji lub mikrozarządzania.

Podstawowym założeniem Scruma i Kanbana jest „mniej znaczy więcej". Jeżeli więc masz wątpliwości, zacznij od „mniej".

W dalszej części książki będę stosował określenie „właściciel produktu" w odniesieniu do osoby ustalającej priorytety w zespole, niezależnie od wykorzystywanego procesu.

4

Scrum nakazuje stosowanie iteracji ograniczonych czasowo

Scrum oparty jest na iteracjach ograniczonych czasowo. Możesz wybrać długość iteracji, ale poszczególne iteracje muszą mieć tę samą długość, dzięki czemu ustanawiana jest *kadencja*.

- **Początek iteracji –** tworzony jest plan iteracji, na przykład zespół, na podstawie priorytetów właściciela produktu, wyciąga z rejestru produktu zadania, które uważa za możliwe do zrealizowania w iteracji.

- **Podczas iteracji –** zespół skupia się na zrealizowaniu zadeklarowanych zadań. Zakres iteracji jest ustalony.

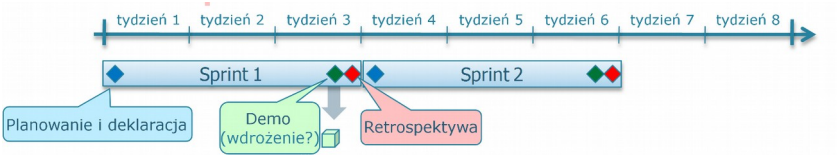
- **Koniec iteracji –** zespół demonstruje działający kod osobom decyzyjnym, najlepiej, gdyby ten kod był gotowy do wdrożenia (przetestowany i kompletny). Następnie zespół organizuje retrospektywę, aby omówić i usprawnić proces.

Tak więc iteracja Scruma to jedna, określona czasowo kadencja łącząca trzy różne aktywności: planowanie, usprawnianie procesu i (idealnie) wdrożenie.

Kanban nie nakazuje iteracji. Ty wybierasz, kiedy wykonywać planowanie, usprawnienia procesu i wdrożenie. Możesz wykonywać te czynności w regularnych odstępach czasu (na przykład: „wdrożenie co poniedziałek") lub na żądanie („wdrożenie, gdy jest coś do wdrożenia").

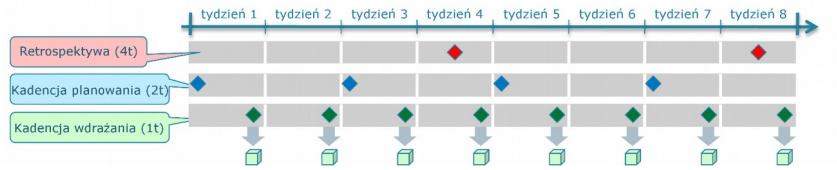
Zespół nr 1 (pojedyncza kadencja)

“Stosujemy iteracje Scruma”



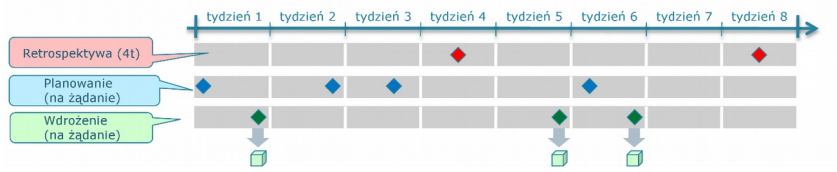
Zespół nr 2 (trzy kadencje)

“Mamy trzy różne kadencje. Co tydzień wdrażamy cały gotowy kod. Co drugi tydzień mamy spotkania, na których planujemy i aktualizujemy priorytety oraz plany wdrożenia. Co czwarty tydzień mamy retrospektywę, na której poprawiamy i modyfikujemy proces”



Zespół nr 3 (sterowany zdarzeniami)

“Sesje planowania organizujemy, kiedy zaczyna nam brakować pracy. Wdrożenie następuje, gdy osiągamy minimalny zestaw cech kwalifikujący do sprzedaży (Minimum Marketable Features – MMF). Jeśli po raz drugi spotykamy się z danym problemem, uruchamiamy spontaniczny krąg jakości. Wykonujemy również wnikliwsze retrospektywy, które odbywają się co 4 tygodnie.”

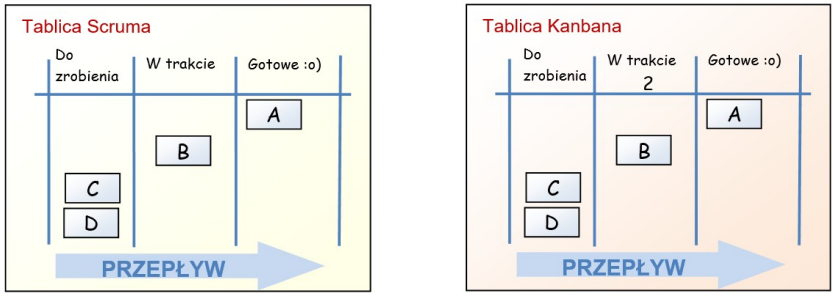


5

Kanban ogranicza WIP dla stanu przepływu, Scrum ogranicza WIP dla iteracjii

W Scrumie rejestr sprintu pokazuje, jakie zadania mają zostać wykonane podczas bieżącej iteracji (w Scrumie iteracja nazywana jest sprintem). Zazwyczaj odzwierciedlają to karty na tablicy nazywanej tablicą Scruma lub tablicą zadań.

Jaka jest zatem różnica między tablicą Scruma a tablicą Kanbana? Zacznijmy od banalnie prostego projektu i porównajmy je:



In W obu przypadkach śledzimy kilka zadań przechodzących przez przepływ. Wybraliśmy trzy stany: do zrobienia, w trakcie i gotowe. Możesz wprowadzać dowolne stany – niektóre zespoły wprowadzają stany takie jak integracja, testy, wdrożenie itd. Nie zapominaj jednak o zasadzie mniej znaczy więcej.

Jaka zatem jest różnica między dwiema przykładowymi tablicami? Tak, mała liczba 2 w środkowej kolumnie tablicy Kanbana. To wszystko. Ta dwójka oznacza: w tej kolumnie w danej chwili mogą się znajdować tylko 2 zadania.

W Scrumie nie ma żadnej zasady zabraniającej zespołowi umieszczenia jednocześnie wszystkich zadań w kolumnie W trakcie! Jest jednak niebezpośrednie ograniczenie, ponieważ cała iteracja ma określony

zakres. W tym przypadku limit dla kolumny będzie wynosił 4, bo na całej tablicy znajdują się tylko 4 zadania. Tak więc Scrum ogranicza WIP niebezpośrednio, podczas gdy Kanban ogranicza go bezpośrednio.

Większość zespołów Scruma z czasem się przekonuje, że nie warto rozpoczynać zbyt wielu zadań, i stara się wprowadzić zasadę, by kończyć zadania przed rozpoczęciem kolejnych. Niektóre zespoły nawet decydują się na bezpośrednie ograniczenie liczby zadań w realizacji, a wtedy – bum! – tablica Scruma staje się tablicą Kanbana.

I Scrum, i Kanban ograniczają liczbę bieżących zadań, ale w odmienny sposób. Zespoły Scruma z reguły mierzą szybkość – ile zadań (lub odpowiednich jednostek, takich jak punkty) wykonywanych jest podczas iteracji. Kiedy zespół pozna swoją szybkość, staje się ona jego limitem WIP (lub przynajmniej wskazówką). Zespół mający średnią szybkość wynoszącą 10 zazwyczaj nie zobowiąże się do wykonania większej liczby zadań (lub punktów) w sprincie.

W Scrumie *WIP jest ograniczany dla jednostki czasu*.

W Kanbanie *WIP jest ograniczany dla stanu przepływu*.

W powyższym przykładzie w stanie ‘W trakcie’ mogą się znajdować maksymalnie 2 zadania, niezależnie od długości kadencji. Musisz wybrać, jakie limity przypisać do poszczególnych stanów przepływu, ale limity powinny być przypisane do wszystkich stanów, zaczynając od jak najwcześniejszego a kończąc na jak najpóźniejszym. Tak więc w powyższym przykładzie powinniśmy również rozważyć dodanie limitu na kolumnie ‘Do zrobienia’ (czyli na kolumnie wsadowej, jakkolwiek ją nazwiesz). Kiedy już będziemy mieć ustanowione limity WIP, będziemy mogli zacząć mierzenie i prognozowanie czasu dostarczenia, na przykład średniego czasu, w jakim zadanie przechodzi przez całą tablicę. Mając przewidywalne czasy dostarczenia, możemy ustanowić ramy SLA (Service Level Agreement) dotyczące poziomu świadczonych usług i tworzyć realistyczne plany wdrożenia.

Jeżeli rozmiary zadań są od siebie drastycznie różne, możesz zdefiniować limity WIP w postaci punktów lub dowolnej innej jednostki. Niektóre zespoły starają się rozbijać prace do mniej więcej tego samego rozmiaru, co pozwala uniknąć takich problemów i zredukować czas potrzebny na szacowanie (niektórzy nawet uważają szacowanie za stratę czasu). Łatwiej jest stworzyć system o swobodnym przepływie, jeżeli prace mają zbliżony rozmiar.

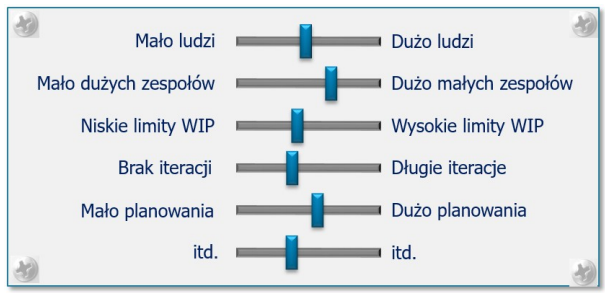
6

Oba są empiryczne

Wyobraź sobie, że przy tych wskaźnikach są pokrętła i możesz konfigurować proces poprzez ich regulowanie. „Chcę wysokiej wydajności, niskich czasów dostarczenia, wysokiej jakości i wysokiej przewidywalności, więc ustawię pokrętła odpowiednio na 10, 1, 10 i 10”.

Czyż to nie byłoby wspaniałe? Niestety takie pokrętła nie istnieją. Ja przynajmniej nic o tym nie wiem. Daj znać, jeżeli je znajdziesz.

Zamiast tego mamy kilka niebezpośrednich ustawień.



Scrum i Kanban są empiryczne, co oznacza, że powinieneś eksperymentować z procesem i przystosowywać go do swojego środowiska. Tak naprawdę musisz eksperymentować. Ani Scrum, ani Kanban nie zapewniają wszystkich odpowiedzi – dają tylko podstawowy zestaw ograniczeń pozwalających sterować usprawnieniami twojego procesu.

- Scrum mówi, że należy mieć wielofunkcyjne zespoły. Kto zatem powinien być w zespole? Nie wiem, eksperymentuj.

- Scrum mówi, że zespół decyduje, ile pracy powinien zadeklarować w sprincie. Ile więc powinien zadeklarować? Nie wiem, eksperymentuj.

- Kanban mówi, że należy ograniczać WIP. Jaki powinien być limit? Nie wiem, eksperymentuj.

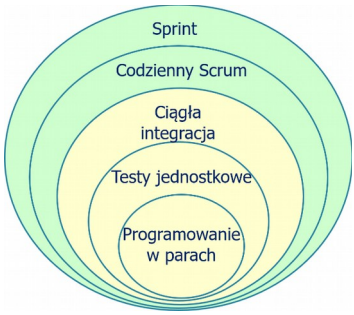
Jak już wcześniej wspomniałem, Kanban zawiera mniej ograniczeń niż Scrum. Oznacza to, że musisz myśleć o większej liczbie parametrów, masz więcej pokręteł do obsługi. Zależnie od kontekstu może to być zarówno zaleta, jak i wada. Kiedy otwierasz okno dialogowe z konfiguracją jakiegoś programu, to wolisz mieć trzy opcje ustawień czy sto? Prawdopodobnie gdzieś pośrodku – w zależności od tego, ile rzeczy chcesz dostosować i na ile rozumiesz narzędzie.

Powiedzmy, że redukujemy limit WIP, zakładając, iż usprawni to nasz proces. Następnie obserwujemy, jak zmieniają się parametry takie jak pojemność, czas dostarczenia, jakość i przewidywalność. Z rezultatów wyciągamy wnioski, a potem zmieniamy kolejne rzeczy, ciągle ulepszając proces.

Procedura ta nosi wiele nazw: kaizen (ciągłe ulepszanie w terminologii Lean), sprawdzaj i dostosowuj (w terminologii Scruma), empiryczna kontrola procesu, a nawet metoda naukowa.

Najważniejszym elementem jest tutaj pętla informacji zwrotnych. Zmień coś => zobacz, jak to działa => wyciągnij wnioski => znów coś zmień. Powinieneś dążyć do tego, aby pętla była jak najkrótsza, dzięki temu bowiem będziesz w stanie szybko dostosowywać swój proces.

W Scrumie podstawową pętlą informacji zwrotnych jest sprint. Pętli jest jednak więcej, zwłaszcza kiedy korzystasz również z XP:



Jeżeli zrobić to dobrze, połączenie Scrum + XP daje kilka bardzo wartościowych pętli informacji zwrotnych.

Wewnętrzna pętla, programowanie w parach, ma długość zaledwie kilku sekund. Błędy są znajdowane i naprawiane w ciągu kilku sekund od ich powstania („Ej, czy ta zmienna nie powinna mieć wartości 3?"). Ten cykl odpowiada na pytanie: czy dobrze budujemy produkt?

Zewnętrzna pętla, sprint, ma długość kilku tygodni. Ten cykl odpowiada na pytanie: czy budujemy dobry produkt?

A jak jest w przypadku Kanbana? Przede wszystkim możesz (i prawdopodobnie powinieneś) wprowadzić wszystkie powyższe pętle informacji zwrotnych do swojego procesu, niezależnie od tego, czy korzystasz z Kanbana, czy nie. Kanban daje natomiast kilka użytecznych pomiarów w czasie rzeczywistym:

- Średni czas dostarczenia. Aktualizowany za każdym razem, kiedy zadanie dotrze do ostatniej kolumny.

- Wąskie gardła. Typowym symptomem jest to, że kolumna X jest zapełniona zadaniami, podczas gdy kolumna X + 1 jest pusta. Szukaj bąbelków powietrza na tablicy.

Zaletą takich pomiarów jest to, że sam możesz wybrać długość pętli informacji zwrotnych, zależnie od tego, jak często chcesz je analizować i wprowadzać zmiany. Długa pętla informacji zwrotnych oznacza, że usprawnienia procesu będą wprowadzane powoli. Zbyt krótka pętla może spowodować, że proces nie zdąży się ustabilizować pomiędzy poszczególnymi zmianami, co z kolei może prowadzić do zaśmiecenia procesu.

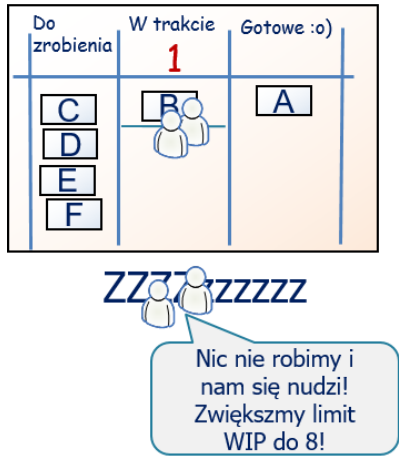
Tak naprawdę długość pętli to kolejna rzecz, z którą możesz eksperymentować… Taka jakby metapętla informacji zwrotnych.

OK, wystarczy.

Przykład: eksperymentowanie z limitami WIP w Kanbanie

Jednym z typowych miejsc, które podlegają modyfikacjom w Kanbanie, jest limit WIP. A skąd mamy wiedzieć, że nasz limit jest odpowiedni?

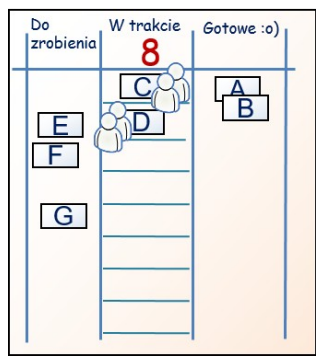
Załóżmy, że mamy zespół złożony z 4 osób i chcemy zacząć z limitem WIP równym 1.



Kiedy zaczniemy pracę nad jednym zadaniem, nie możemy zacząć nowego, dopóki nie skończymy tego zaczętego. Zostanie ono więc wykonane naprawdę szybko.

Super! Ale okazuje się, że wszystkie 4 osoby raczej nie mogą pracować nad tym samym zadaniem (w naszym przykładzie), więc część osób nie robi nic. Jeżeli jest tak tylko raz na jakiś czas, nie ma problemu, jeżeli jednak dzieje się to często, w konsekwencji średni czas dostarczenia wzrośnie. Limit WIP równy 1 oznacza, że zadanie, kiedy już dotrze do kolumny ‘W trakcie’, przejdzie przez nią bardzo szybko, ale spędzi w kolumnie ‘Do zrobienia’ więcej czasu niż to konieczne, dlatego całkowity czas przepływu zadania przez cały proces będzie wysoki.

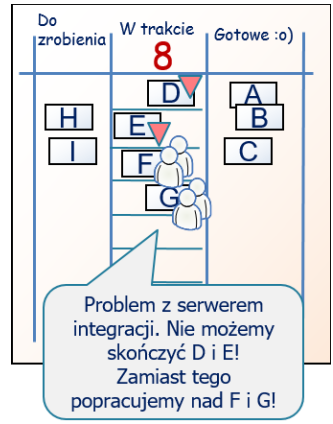
Jeżeli zatem WIP równy 1 był zbyt niski, może podnieść go do 8?



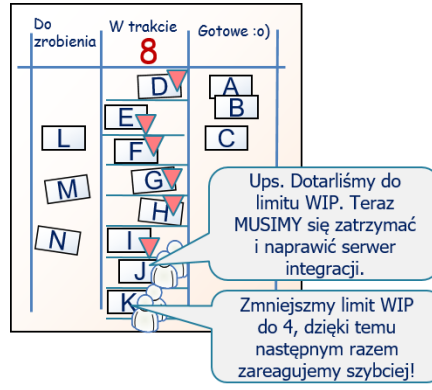
Przez jakiś czas jest lepiej. Odkrywamy, że przeważnie praca w parach sprawia, iż zadania wykonywane są szybciej. Tak więc w 4-osobowym zespole zazwyczaj jednocześnie trwają prace nad 2 zadaniami. Limit WIP

jest tylko górnym limitem, więc mniejsza liczba zadań w kolumnie jest dopuszczalna.

Wyobraź sobie jednak, że napotykamy problem z serwerem integracji i nie możemy w pełni skończyć zadań (nasze kryteria ukończenia zadania obejmują integrację). Takie rzeczy się zdarzają, prawda?

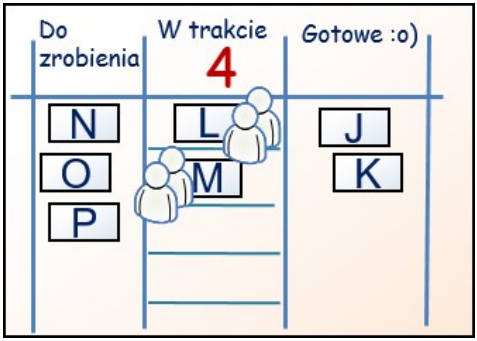


Skoro nie możemy ukończyć zadań D i E, zaczniemy pracę nad zadaniem F. Jego też nie możemy zintegrować, więc zajmiemy się zadaniem G. Po jakimś czasie zapełnimy kolumnę – natrafimy na ograniczenie: „8 zadań”:



Nie możemy już wciągać nowych zadań. Musimy w końcu naprawić ten przeklęty serwer integracji! Limit WIP zmusił nas do podjęcia działań zamiast pobierania kolejnych zadań.

To dobrze. Gdyby jednak limit WIP wynosił 4, zareagowalibyśmy znacznie szybciej, przez co średni czas dostarczenia byłby niższy. Chodzi więc o równowagę. Mierzymy średni czas dostarczenia i optymalizujemy nasze limity WIP, aby poprawiać czas dostarczenia:



Po jakimś czasie może się okazać, że prace nawarstwiają się w kolumnie Do zrobienia. Może pora i tam ustawić limit WIP.

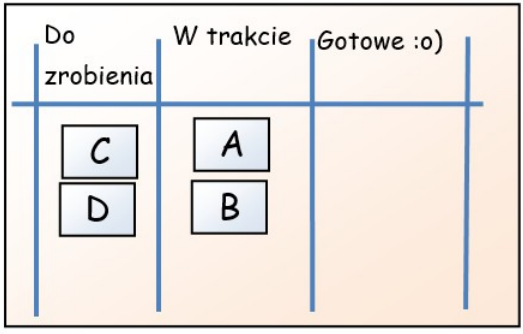
Po co nam w ogóle kolumna Do zrobienia? Otóż gdyby klient był zawsze dostępny i mógł na bieżąco mówić zespołowi, co robić, kolumna ta nie byłaby potrzebna. W tym przypadku jednak klient nie jest zawsze dostępny, dlatego kolumna Do zrobienia daje zespołowi niewielki bufor, z którego może pobierać zadania.

Eksperymentuj! Albo – jak mawiają scrumolodzy – sprawdzaj i dostosowuj!

7

Scrum opiera się zmianom w trakcie iteracji

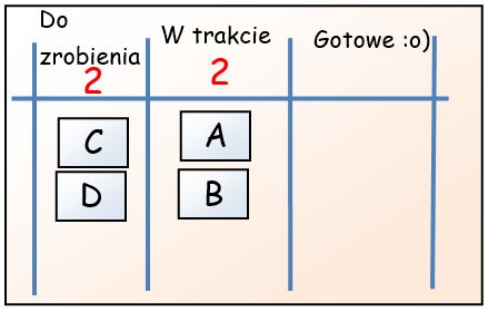
Załóżmy, że nasza tablica Scruma wygląda następująco:



Co się stanie, jeżeli pojawi się ktoś, kto będzie chciał dodać na tablicę zadanie E?

Zespół Scruma powie zazwyczaj coś w stylu: „Przykro nam, zobowiązaliśmy się zrobić A+B+C+D w trakcie tego sprintu. Możesz jednak dodać E do rejestru produktu. Jeżeli właściciel produktu uzna, że to zadanie ma wysoki priorytet, zrobimy to w następnym sprincie". Sprinty o odpowiedniej długości dają zespołowi odpowiednio dużo czasu, aby zrobić pewną porcję funkcjonalności, a właścicielowi produktu możliwość regularnego zarządzania priorytetami.

Co by powiedział zespół Kanbana?



Mógłby powiedzieć: „Możesz dodać to zadanie do kolumny Do zrobienia. Ponieważ jednak limit wynosi 2, musisz najpierw zdjąć zadanie C lub D. Aktualnie pracujemy nad A i B, ale jak tylko zwolni się miejsce w kolumnie W trakcie, wciągniemy pierwsze zadanie z kolumny Do zrobienia".

Tak więc czas reakcji (czyli jak długo trwa reakcja na zmianę priorytetów) zespołu Kanbana zależy od czasu potrzebnego na zwolnienie pojemności przy spełnieniu zasady „jedno zadanie wychodzi = jedno zadanie wchodzi" (narzucanej przez limit WIP).

W Scrumie czas reakcji średnio wynosi połowę czasu trwania sprintu.

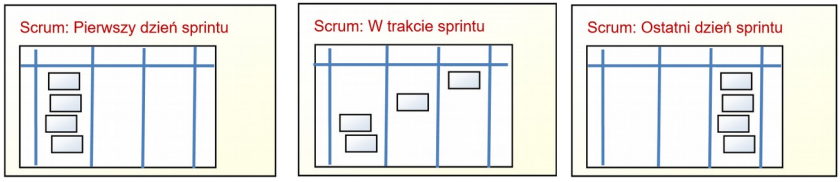
W Scrumie właściciel produktu nie może dotykać tablicy, ponieważ zespół zobowiązał się do wykonania pewnej liczby zadań w iteracji. W Kanbanie musisz wyznaczyć swoje własne zasady dotyczące tego, kto może wprowadzać zmiany na tablicy i w jakim zakresie. Zazwyczaj właściciel produktu dostaje po lewej stronie tablicy kolumnę w stylu Do zrobienia, Oczekujące, Rejestr lub Proponowane, w której może wprowadzać takie zmiany, jakie uważa za stosowne.

Te dwa podejścia nie są jednak jedynymi. Zespół Scruma może pozwolić właścicielowi produktu na zmienianie priorytetów w trakcie sprintu (chociaż normalnie zostałoby to uznane za wyjątek). A zespół Kanbana może wprowadzić restrykcje co do tego, kiedy mogą być zmieniane priorytety. Zespół Kanbana może również zdecydować się na wykorzystanie ograniczonych czasowo iteracji o ustalonym zakresie zadań, podobnych jak w Scrumie.

8

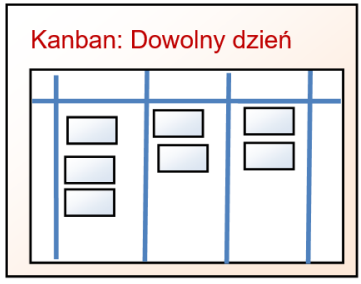
Tablica Scruma jest czyszczona pomiędzy iteracjami

Tablica Scruma w poszczególnych fazach sprintu wygląda mniej więcej tak:



Kiedy sprint się kończy, tablica jest czyszczona – wszystkie zadania są usuwane. Rozpoczynany jest nowy sprint, a po spotkaniu dotyczącym planowania sprintu mamy nową tablicę z nowymi zadaniami w lewej kolumnie. Technicznie jest to stracony czas, w doświadczonych zespołach Scruma zazwyczaj nie trwa to zbyt długo, a proces przygotowywania tablicy daje przyjemne odczucie spełnienia i domknięcia pewnego etapu. Coś jak zmywanie po obiedzie: robienie tego nie jest przyjemne, ale uczucie po – owszem.

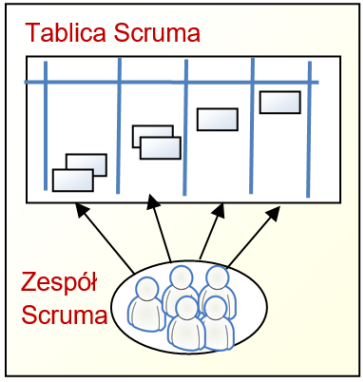
W Kanbanie tablica używana jest cały czas – nie ma potrzeby czyszczenia jej i przygotowywania od nowa. Wygląda ona tak:



9

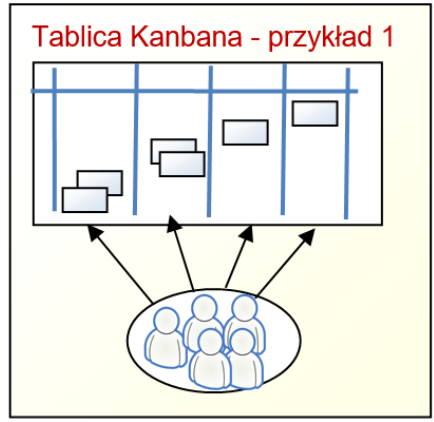
Scrum nakazuje zespoły wielofunkcyjne

Tablica Scruma jest własnością dokładnie jednego zespołu. Zespół Scruma jest wielofunkcyjny, zawiera wszystkie umiejętności potrzebne do ukończenia wszystkich zadań w iteracji. Tablica Scruma jest zazwyczaj widoczna dla wszystkich zainteresowanych, ale tylko zespół, którego jest własnością, może ją zmieniać – jest to jego narzędzie do zarządzania zadaniami bieżącej iteracji.

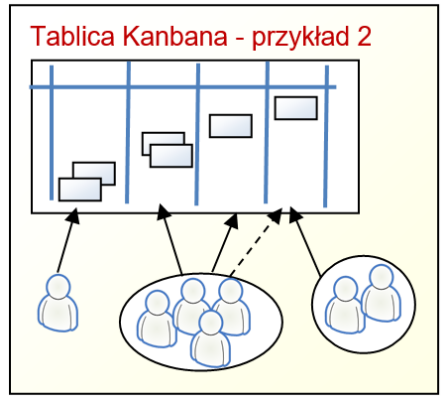


W Kanbanie zespoły wielofunkcyjne są opcjonalne, a tablica nie musi być własnością jednego zespołu. Tablica związana jest z przepływem pracy, niekoniecznie z jednym zespołem.

Oto dwa przykłady:



Przykład 1. Cała tablica jest obsługiwana przez jeden wielofunkcyjny zespół. Zupełnie jak w Scrumie.



Przykład 2. Właściciel produktu ustala priorytety w kolumnie 1. Wielofunkcyjny zespół zajmuje się programowaniem (kolumna 2) i testowaniem (kolumna 3). Wdrożenie (kolumna 4) wykonywane jest przez dedykowany zespół. Występuje pewne nakładanie się kompetencji, jeżeli więc zespół zajmujący się wdrożeniem stanie się wąskim gardłem, jeden z programistów będzie mógł mu pomóc.

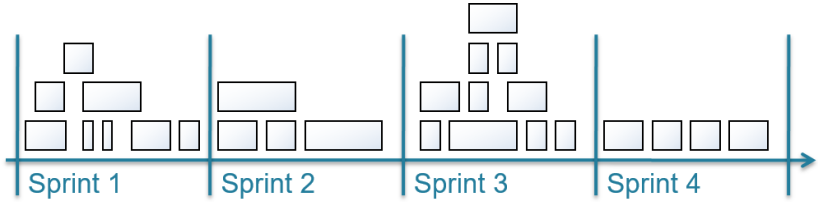
W Kanbanie musisz ustanowić pewne podstawowe zasady dotyczące tego, kto i jak wykorzystuje tablicę, następnie eksperymentować z tymi zasadami i optymalizować przepływ.

10

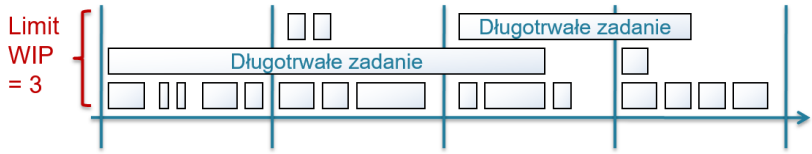
W Scrumie zadania z rejestru muszą mieścić się w sprincie

Zarówno Scrum, jak i Kanban są wykorzystywane przy inkrementacyjnym budowaniu produktu, na przykład przy rozbiciu pracy na mniejsze zadania.

Zespół Scruma deklaruje wykonanie takiej liczby zadań, jaką ma szansę ukończyć w trakcie jednej iteracji (zgodnie z kryteriami ukończenia zadania). Jeżeli zadanie jest zbyt długie, aby zmieścić się w sprincie, zespół lub właściciel produktu próbują dzielić je na mniejsze zadania dopóty, dopóki nie będzie można wykonać ich w ramach jednej iteracji. Jeżeli zadania przeważnie są duże, iteracje będą dłuższe (zazwyczaj nie będą jednak przekraczały 4 tygodni).



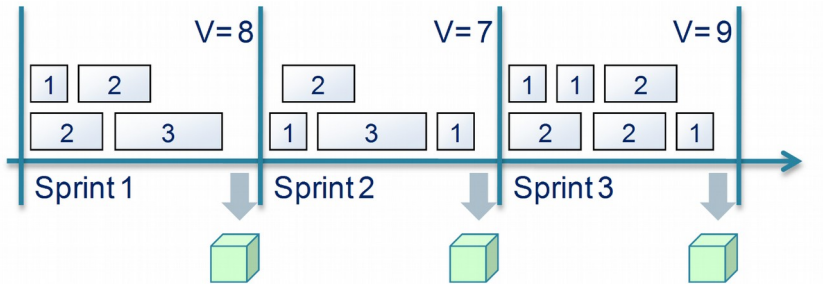
Zespoły Kanbana starają się minimalizować czas dostarczenia i wyrównywać przepływ, co stanowi pośrednią motywację do dzielenia zadań na relatywnie małe zadania. Nie ma jednak żadnej bezpośredniej zasady stwierdzającej, że zadania muszą być na tyle małe, aby zmieściły się w określonych ramach czasowych. Na tej samej tablicy możemy mieć jedno zadanie, które trwa 1 miesiąc oraz inne, trwające 1 dzień.



11

Scrum nakazuje szacowanie i określanie szybkości

W Scrumie zespoły powinny szacować relatywny rozmiar (ilość pracy) każdego z zadań, którego się podejmują. Sumując rozmiary wszystkich ukończonych zadań na końcu sprintu, otrzymujemy szybkość. Szybkość jest miarą pojemności – ile zadań możemy dostarczyć w ramach jednego sprintu. Oto przykład zespołu ze średnią szybkością równą 8.



Dobrze jest wiedzieć, że średnia szybkość zespołu wynosi 8, ponieważ dzięki temu możemy wykonywać realistyczne prognozy na temat tego, które zadania jesteśmy w stanie wykonać w nadchodzących sprintach, a tym samym tworzyć realistyczne plany.

Kanban nie nakazuje żadnej metody szacowania. Jeżeli więc musisz wyznaczać terminy, musisz sam zdecydować, jak zapewnić przewidywalność.

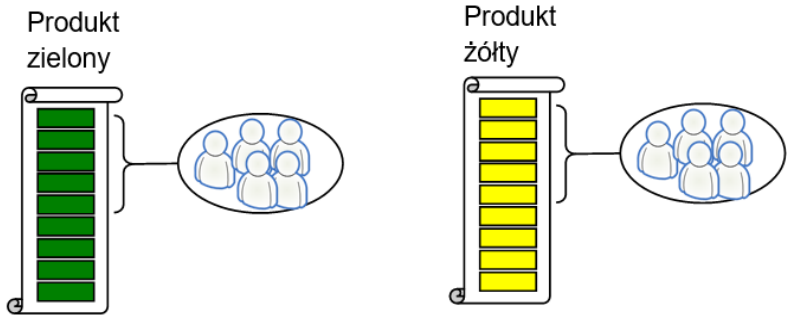
Niektóre zespoły wybierają szacowanie i określanie szybkości analogicznie jak w Scrumie. Inne rezygnują z szacowania i próbują rozbijać prace na zadania o zbliżonej wielkości, dzięki czemu mogą mierzyć szybkość na podstawie liczby wykonanych zadań w jednostce czasu (na przykład liczba funkcjonalności tygodniowo). Jeszcze inne zespoły łączą zadania w grupy MMF, mierzą średni czas dostarczenia dla MMF i otrzymaną wartość wykorzystują do ustalania SLA – na przykład „kiedy zobowiązujemy się do wydania MMF, zakończymy prace w ciągu 15 dni”.

Istnieje mnóstwo interesujących technik planowania wdrożeń i zarządzania zobowiązaniami dla Kanbana, aczkolwiek nie nakazuje on żadnej konkretnej techniki, możesz zatem uruchomić Google'a i wypróbowywać różne techniki, dopóki nie znajdziesz odpowiedniej dla siebie. Z czasem na pewno powstaną jakieś najlepsze praktyki.

12

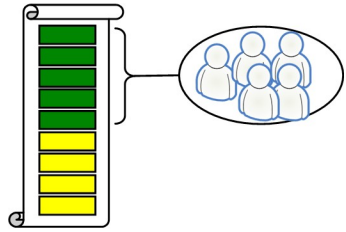
Oba pozwalają pracować jednocześnie nad wieloma produktami

W Scrumie „rejestr produktu” jest dość niefortunną nazwą, ponieważ sugeruje, że wszystkie zadania muszą dotyczyć tego samego produktu. Oto dwa produkty, zielony i żółty, każdy z nich ma swój własny rejestr produktu i swój własny zespół:

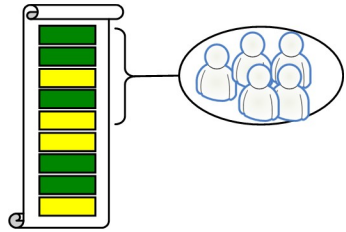


Co, jeżeli masz tylko jeden zespół? Wówczas o rejestrze produktu powinieneś myśleć raczej w kategoriach rejestru zespołu. Zawiera on priorytety na nadchodzące iteracje dla jednego konkretnego zespołu (lub grupy zespołów). Jeśli więc zespół ten pracuje nad wieloma produktami, połącz oba produkty w jedną listę. Takie podejście zmusi do priorytetyzowania produktów, co w niektórych przypadkach jest użyteczne.

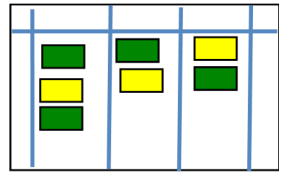
W praktyce można to robić na kilka sposobów. Jedna strategia to zmuszenie zespołu do skupienia się na jednym produkcie w ramach jednego sprintu:



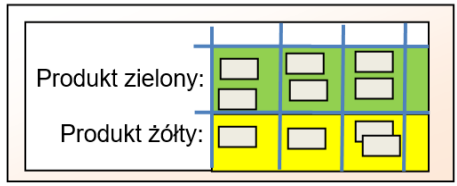
Inna strategia to praca nad obydwoma produktami w tym samym sprincie:



W Kanbanie jest tak samo. Na tej samej tablicy możemy mieć wiele produktów. Możemy rozróżnić je za pomocą kolorów karteczek:



…lub za pomocą odrębnych torów:



13

Oba są Lean i Agile

Nie zamierzam tutaj omawiać filozofii Lean i manifestu Agile, ale ogólnie mówiąc, zarówno Scrum, jak i Kanban dobrze wpisują się w te zasady i wartości. Na przykład:

- Zarówno Scrum jak i Kanban są systemami harmonogramowania opartymi na wciąganiu, co wiąże się z zasadą zarządzania magazynem w Lean – JIT (Just in Time). Oznacza to, że zespół wybiera, kiedy i ile pracy chce zadeklarować a następnie „wciąga" pracę, gdy jest gotowy – praca nie jest „wpychana" z zewnątrz. Tak jak drukarka wciąga kolejną stronę dopiero, gdy jest gotowa na niej drukować (chociaż zapas papieru, z którego może czerpać, jest niewielki i ograniczony).

- Scrum i Kanban oparte są na ciągłym i empirycznym procesie optymalizacji, co związane jest z zasadą kaizen w Lean.

- Scrum i Kanban przedkładają reagowanie na zmiany ponad przestrzeganie planu (chociaż w Kanbanie zazwyczaj czas reakcji jest krótszy) i jest to jedna z czterech wartości manifestu Agile.

…i tak dalej.

Z pewnej perspektywy Scrum może być postrzegany jako niespecjalnie leanowy, ponieważ nakazuje łączenie zadań w iteracje ograniczone czasowo. Jest to jednak uzależnione od długości iteracji i od tego, do czego porównujesz. W porównaniu z bardziej tradycyjnym procesem, gdzie prawdopodobnie integracja i wdrożenie mają miejsce 2 – 4 razy do roku, zespół Scruma, produkujący kod gotowy do wdrożenia co 2 tygodnie, jest bardzo leanowy.

Ale jeżeli będziesz coraz bardziej skracać iteracje, w zasadzie będziesz się zbliżać do Kanbana. Kiedy zaczynasz myśleć o wdrożeniu iteracji krótszych niż tydzień, możesz rozważyć zupełne pozbycie się iteracji ograniczonych czasowo.

Już to mówiłem i będę to powtarzał: eksperymentuj, dopóki nie znajdziesz czegoś, co będzie działać w twoim przypadku. A potem eksperymentuj dalej :o)

14

Drobne różnice

Poniżej opisuję kilka różnic, które w porównaniu do poprzednio omówionych zdają się mieć mniejsze znaczenie. Dobrze jednak być świadomym ich istnienia.

Scrum nakazuje prowadzenie rejestru produktu definiującego priorytety

Priorytety w Scrumie są zawsze ustalane poprzez sortowanie rejestru produktu, a zmiana priorytetów odnosi skutek w kolejnym sprincie (nie w bieżącym). W Kanbanie możesz wybrać dowolny sposób określania priorytetów (lub nawet nie używać żadnego), a zmiany wchodzą w życie, jak tylko zwolni się pojemność (nie o ustalonym czasie). Rejestr produktu może istnieć lub nie i może on odzwierciedlać priorytety lub nie.

W praktyce różnica jest niewielka. Na tablicy Kanbana skrajna lewa kolumna spełnia zazwyczaj to samo zadanie co rejestr produktu w Scrumie. Niezależnie od tego, czy lista jest sortowana według priorytetów, zespół potrzebuje jakiegoś określonego sposobu podejmowania decyzji co do tego, jakie zadanie należy wciągnąć do pracy. Takie reguły to na przykład:

- Zawsze bierz zadanie z góry.
- Zawsze bierz najstarsze zadanie (każde zadanie musi mieć stempel czasowy).
- Weź zadanie.
- Spędzaj około 20% czasu na zadaniach konserwacyjnych i 80% na nowych funkcjonalnościach.
- Podziel pojemność zespołu pomiędzy produkty A i B.
- Jeżeli istnieją, zawsze pobieraj najpierw czerwone zadania.

W Scrumie rejestr produktu może być również wykorzystywany w sposób podobny jak w Kanbanie. Możemy ograniczyć jego rozmiar i określić reguły ustalania priorytetów.

Scrum nakazuje codzienne spotkania

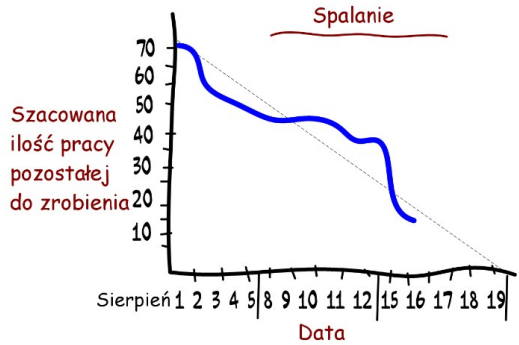
Zespoły Scruma odbywają krótkie codzienne spotkania (trwające do 15 minut) zawsze o tej samej godzinie i w tym samym miejscu. Celem tych spotkań jest dzielenie się wiedzą o bieżących pracach, planowanie pracy na dany dzień i identyfikacja znaczących problemów. Spotkanie to jest nazywane codziennym standupem, ponieważ zazwyczaj odbywa się na stojąco (aby nie trwało długo i aby nie nastąpił spadek energii u uczestników).

Kanban nie mówi nic o codziennych spotkaniach, ale większość zespołów Kanbana i tak z nich korzysta. To świetna technika, niezależnie od procesu, z którego korzystasz.

W Scrumie spotkanie jest zorientowane na ludzi — każda osoba raportuje po kolei. W wielu zespołach Kanbana spotkania są bardziej zorientowane na tablicę i skupiają się na wąskich gardłach i innych widocznych problemach. To podejście jest bardziej skalowalne. Jeżeli cztery zespoły będą współdzielić tę samą tablicę i codzienne spotkania będą organizować wspólnie, nie będzie potrzeby, aby wypowiadali się wszyscy uczestnicy, o ile skupimy się na wąskich gardłach i problemach.

Scrum nakazuje rysowanie diagramów spalania

Diagram spalania pokazuje, ile pracy pozostało do zrobienia w danym dniu sprintu.



Jednostki na osi Y są takie same jak jednostki wykorzystywane do szacowania zadań w sprincie. Są to zazwyczaj godziny lub dni (jeżeli zespół dzieli funkcjonalności z rejestru na zadania) albo punkty (jeżeli tego nie robi). Istnieje jednak wiele wariacji.

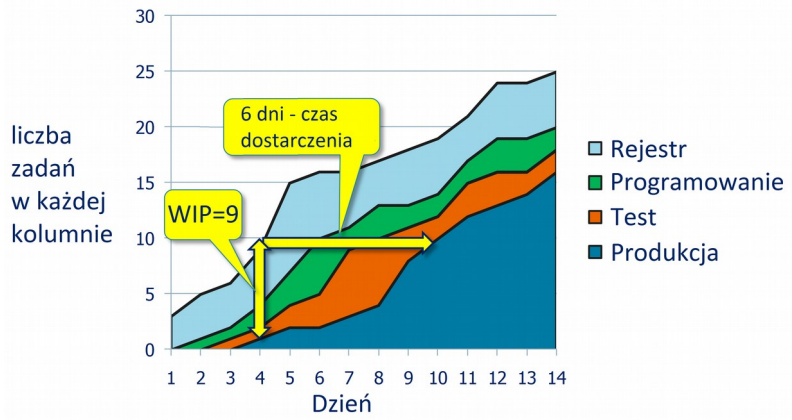
W Scrumie diagramy spalania wykorzystywane są jako jedno z podstawowych narzędzi śledzenia postępu w iteracji.

Niektóre zespoły wykorzystują również diagramy spalania wdrożenia, które pełnią taką samą rolę, ale na poziomie wdrożeń – pokazują zwykle, ile punktów pozostało w rejestrze produktu po zakończeniu poszczególnych sprintów.

Głównym celem diagramu spalania jest jak najszybsze wykrycie, że zespół nie wyrabia się z harmonogramem, dzięki czemu można podjąć stosowne działania.

Kanban nie mówi nic o diagramach spalania. Kanban nie mówi nic o żadnym diagramie. Możesz jednak wykorzystywać diagramy, jakie tylko chcesz (włącznie z diagramem spalania).

Oto przykład kumulatywnego diagramu przepływu. Ten rodzaj wykresu ilustruje, jak płynny jest twój przepływ i jak WIP wpływa na czas dostarczenia.



Zasada działania jest bardzo prosta. Codziennie dodawana jest liczba zadań w każdej kolumnie tablicy Kanbana, a otrzymany wynik oznaczany jest na osi Y. Tak więc w 4. dniu na tablicy znajdowało się 8 zadań. Zaczynając od prawej strony, na tablicy było 1 zadanie w kolumnie Produkcja, 1 zadanie w kolumnie Test, 2 zadania w kolumnie Programowanie i 5 zadań w kolumnie Rejestr. Jeżeli będziemy zapisywać te punkty codziennie i połączymy kropki, otrzymamy ładny diagram, podobny do tego powyżej. Strzałki pozioma i pionowa przedstawiają relację między WIP a czasem dostarczenia.

Strzałka pozioma pokazuje, że zadania dodane do rejestru w 4. dniu były uruchamiane w systemie produkcyjnym średnio po 6 dniach. Około

połowę tego czasu stanowiły testy. Widzimy, że gdybyśmy ograniczyli WIP w kolumnach Test i Rejestr, znacząco zredukowalibyśmy czas dostarczenia.

Nachylenie granatowego obszaru obrazuje szybkość (na przykład liczbę zadań zakończonych dziennie). Z czasem możemy zaobserwować, jak większa szybkość prowadzi do redukcji czasu dostarczenia, podczas gdy wyższy WIP powoduje jego wydłużenie.

Większość organizacji chce pracować szybciej (czyli redukować czas dostarczenia). Niestety wiele z nich, aby to osiągnąć, wpada w pułapkę zatrudniania kolejnych osób lub pracy po godzinach. Zazwyczaj najbardziej efektywnym sposobem na szybsze dostarczanie produktu jest wygładzenie przepływu pracy i ograniczenie pojemności, a nie zatrudnianie większej ilości osób czy też cięższa praca. Ten diagram pokazuje dlaczego tak jest i tym samym zwiększa prawdopodobieństwo, że współpraca zespołu i osób zarządzających projektem będzie owocna.

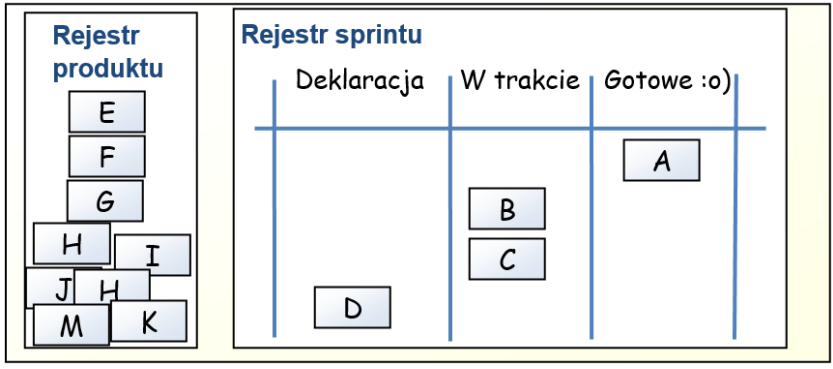
Wszystko staje się jeszcze jaśniejsze, jeżeli rozróżnimy stany kolejki (takie jak „czeka na testy”) i stany w trakcie pracy (takie jak „w trakcie testu”). Chcemy jak najbardziej zminimalizować liczbę elementów czekających w kolejkach, a kumulatywny diagram przepływu może pomóc zapewnić odpowiednią motywację.

15

Tablica Scruma kontra tablica Kanbana – bardziej rozbudowany przykład

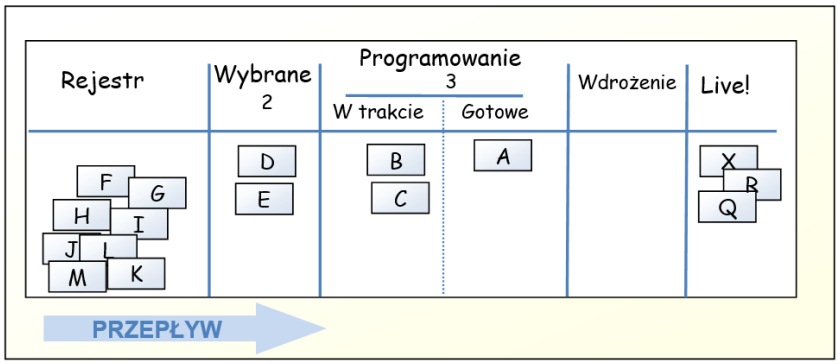
W Scrumie rejestr sprintu to tylko część prac – pokazuje, co zespół zobowiązał się zrobić w trakcie bieżącego sprintu. Reszta prac czeka w rejestrze produktu – czyli na liście wszystkich zadań, które mają być wykonane w kolejnych sprintach.

Właściciel produktu ma wgląd do rejestru sprintu, ale nie może go dotykać. Może zmieniać rejestr produktu, kiedy tylko chce, ale zmiany nie mają żadnego efektu (w sensie zmiany wykonywanej pracy) aż do kolejnego sprintu.



Kiedy sprint się skończy, zespół dostarcza gotowy do wdrożenia kod właścicielowi produktu. Tak zespół kończy sprint, robi przegląd sprintu i z dumą prezentuje funkcjonalności A, B, C i D właścicielowi produktu. Właściciel produktu może teraz podjąć decyzję o wdrożeniu funkcjonalności. Ta ostatnia część – właściwe wdrożenie produktu – nie jest zazwyczaj częścią sprintu i tym samym nie jest widoczna w rejestrze sprintu.

W tym scenariuszu tablica Kanbana mogłaby wyglądać następująco:



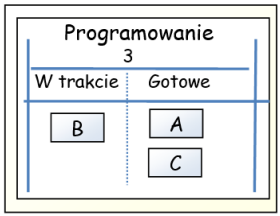
Tutaj cały przepływ jest na jednej tablicy – nie patrzymy tylko na to, czym zajmuje się zespół Scruma w danej iteracji.

W powyższym przykładzie kolumna „Rejestr” jest ogólną listą życzeń, bez żadnej konkretnej kolejności. Kolumna „Wybrane” zawiera zadania o wysokim priorytecie i jest ograniczona do 2 zadań, co oznacza, że w danej chwili mogą znajdować się w niej 2 zadania o wysokim priorytecie. Kiedy zespół jest gotowy do podjęcia pracy nad nowym zadaniem, wciąga pierwsze zadanie z kolumny „Wybrane”. Właściciel produktu może wprowadzać zmiany w kolumnach „Rejestr” i „Wybrane”, kiedy tylko chce, nie może jednak zmieniać żadnych innych kolumn.

Kolumna „Programowanie” (podzielona na dwie części) zawiera te zadania, nad którymi aktualnie trwają prace, a jej limit WIP jest równy 3. W terminologii sieciowej limit odpowiadałby *przepustowości*, a czas dostarczenia byłby odpowiednikiem *pingu* (lub czasu odpowiedzi).

Dlaczego podzieliliśmy kolumnę „Programowanie” na „W trakcie” i „Gotowe”? Aby dać zespołowi wdrażającemu informację o tym, które zadania może wciągnąć do kolumny „Wdrożenie”.

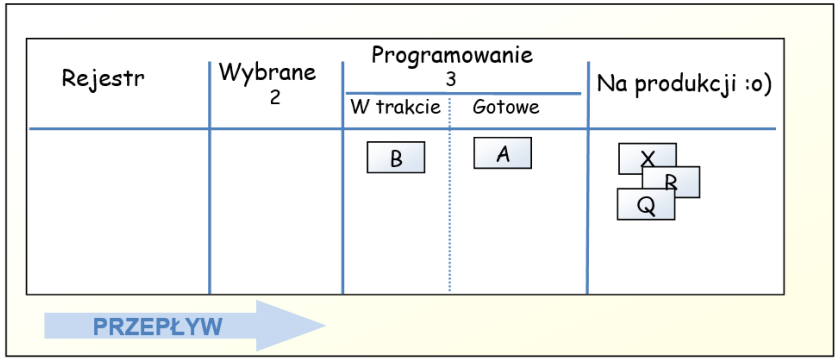
Limit kolumny „Programowanie” wynosi 3 i jest współdzielony przez dwie kolumny. Dlaczego? Załóżmy, że w kolumnie „Gotowe” znajdują się 2 zadania:



Oznacza to, że w kolumnie „W trakcie” może się znajdować tylko 1 zadanie. W związku z tym wystąpi nadmiarowa pojemność – są programiści, którzy mogliby rozpocząć pracę nad nowym zadaniem, ale nie pozwala im na to limit. Daje im to motywację do pomocy we wdrażaniu funkcjonalności, aby wyczyścić kolumnę „Gotowe” i zmaksymalizować przepływ. Efekt ten jest stopniowy – im więcej zadań znajduje się w kolumnie „Gotowe”, tym mniej zadań może znaleźć się w kolumnie „W trakcie” – co pomaga zespołowi skupić się na tym, co jest ważne.

Przepływ jednoelementowy

Przepływ jednoelementowy jest czymś w rodzaju „idealnego przepływu”, w którym zadanie przechodzi przez całą tablicę, nigdy nie pozostając w kolejce. Oznacza to, że w każdej chwili ktoś pracuje nad tym zadaniem. Oto, jak mogłaby wyglądać tablica w takim przypadku:

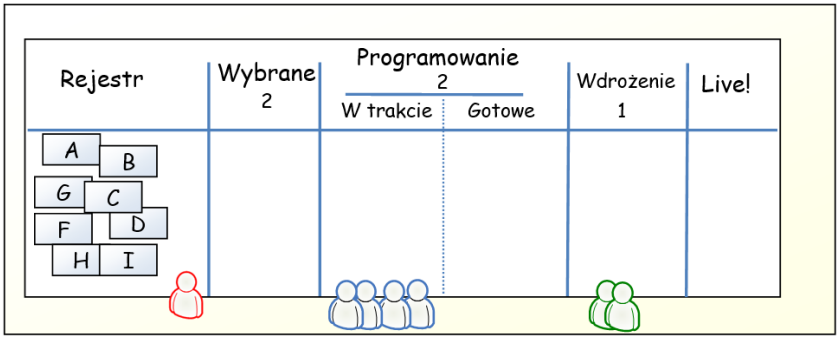


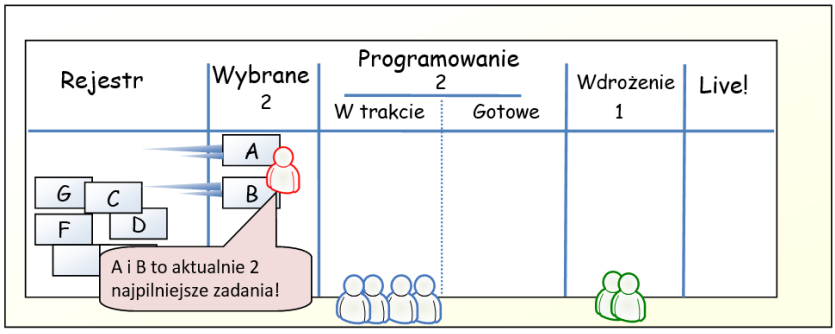
B znajduje się w trakcie programowania, A jest aktualnie wdrażane w środowisku produkcyjnym. Kiedy zespół będzie gotowy do rozpoczęcia kolejnego zadania, pyta właściciela produktu, co jest najważniejsze, i otrzymuje natychmiastową odpowiedź. Jeżeli ten idealny scenariusz się utrzyma, będziemy mogli pozbyć się dwóch kolejek – „Rejestr” i „Wybrane” – i uzyskać naprawdę krótki czas dostarczenia.

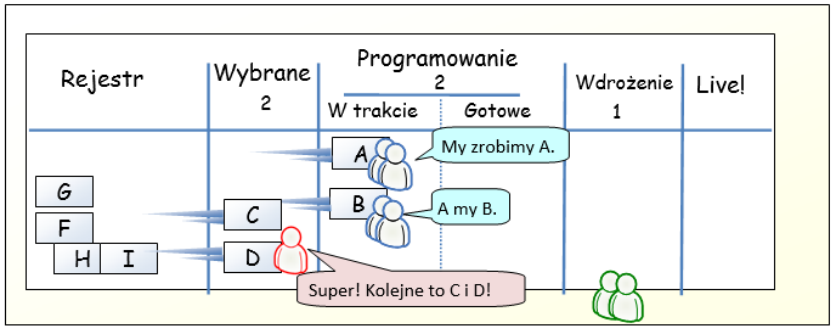
Świetnie to ujął Corey Ladas: „Idealny proces planowania powinien zapewnić zespołowi najlepsze kolejne zadanie do pracy, nie mniej i nie więcej”.

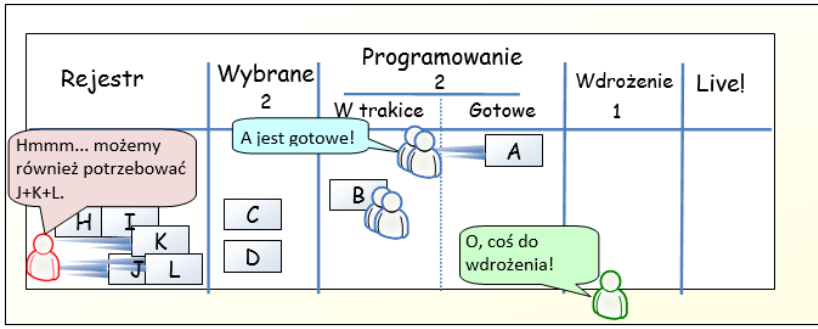
Limity WIP znajdują się na tablicy, aby zapobiec wymknięciu się problemów spod kontroli, jeżeli więc prace przebiegają płynnie, limity WIP nie są wykorzystywane.

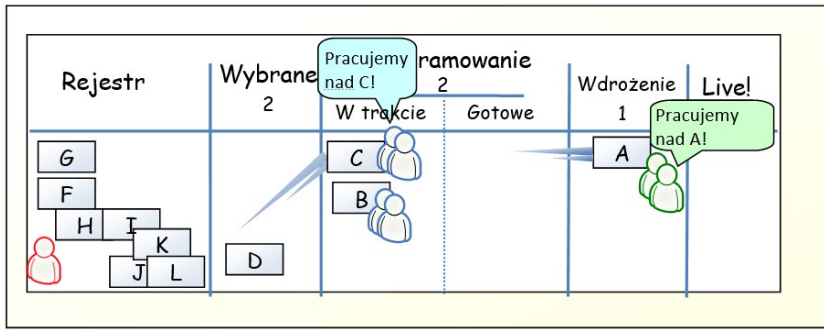
Jeden dzień w świecie Kanbana

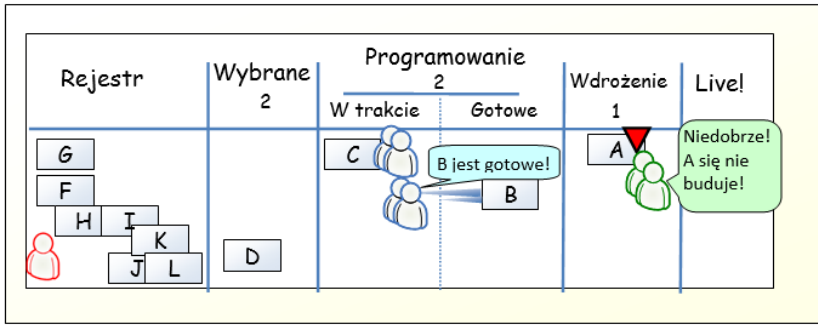


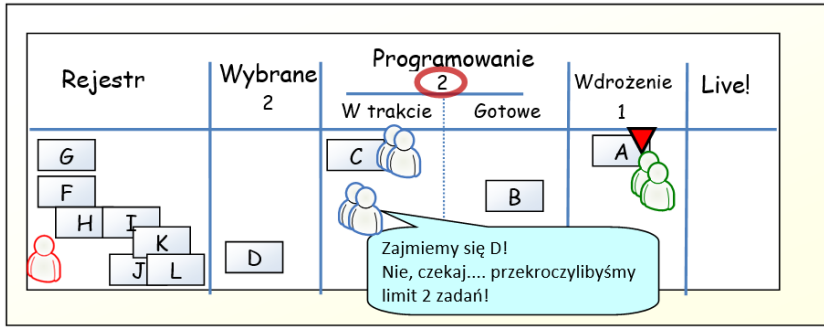


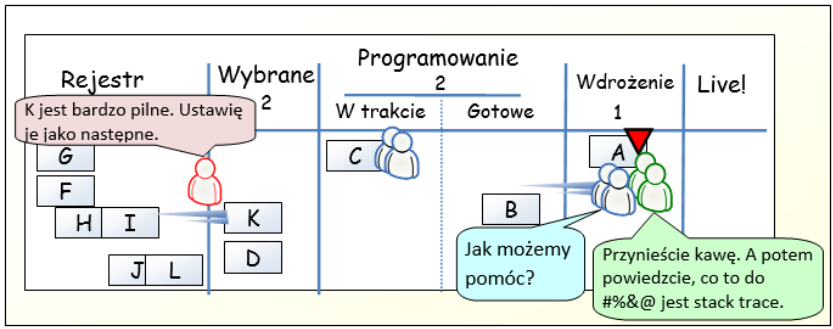


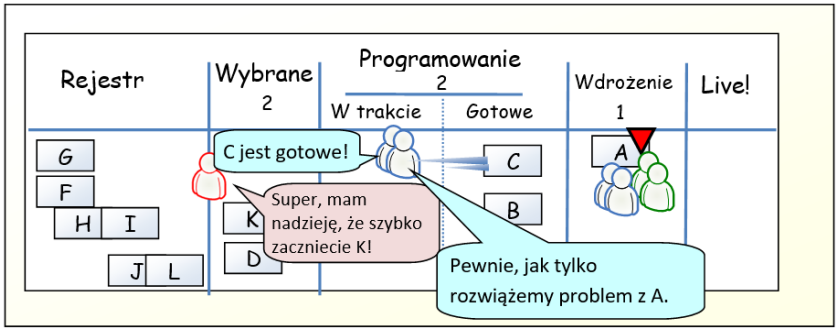

Rejestr
Wybrane 2
Programowanie 2
W trakice
Gotowe
Wdrożenie 1
Live!
Hmmm... możemy również potrzebować J+K+L.
A jest gotowe!
A
B
C
D
H
I
K
J
L
O, coś do wdrożenia!

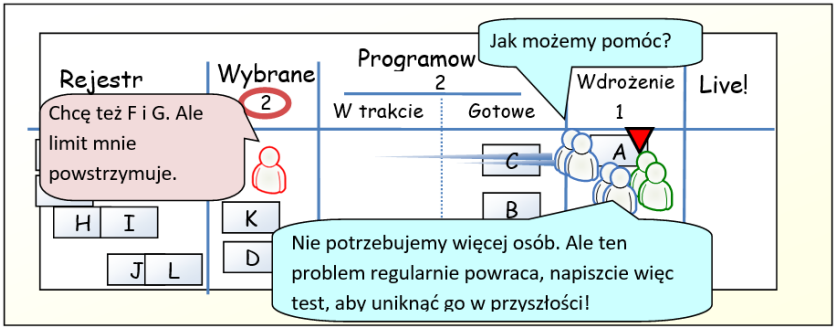

Rejestr
Wybrane 2
Pracujemy nad C!
ramowanie 2
W trakcie
Gotowe
Wdrożenie 1
Live!
Pracujemy nad A!
G
F
H
I
K
J
L
C
B
A
D

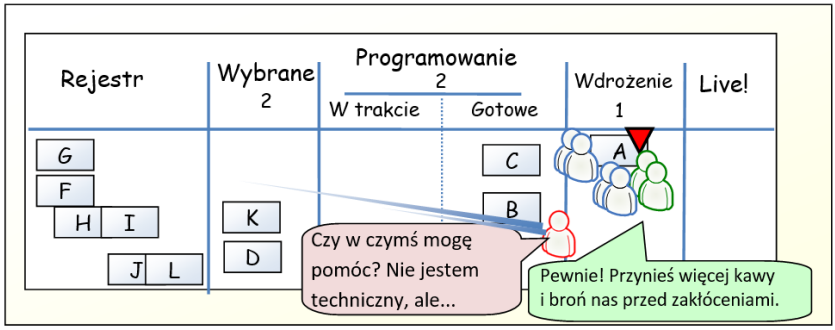

Rejestr
Wybrane 2
Programowanie 2
W trakcie
Gotowe
Wdrożenie 1
Live!
Niedobrze! A się nie buduje!
G
F
H
I
K
J
L
C
B is gotowe!
B
A
D

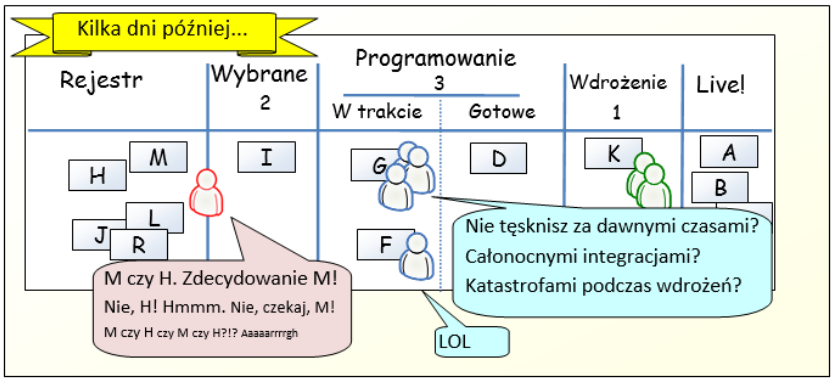

Rejestr
Wybrane 2
Programowanie 2
W trakcie
Gotowe
Wdrożenie 1
Live!
G
F
H
I
K
J
L
C
B
A
D
Zajmiemy się D!
Nie, czekaj.... przekroczylibyśmy limit 2 zadań!


Rejestr
Wybrane
2
Programowanie
2
W trakcie
Gotowe
Wdrożenie
1
Live!
K jest bardzo pilne. Ustawię je jako następne.
G
F
H
I
J
L
K
D
C
B
A
Jak możemy pomóc?
Przynieście kawę. A potem powiedzcie, co to do #%&@ jest stack trace.


Rejestr
Wybrane
2
Programowanie
2
W trakcie
Gotowe
Wdrożenie
1
Live!
G
F
H
I
J
L
K
D
C
B
A
C jest gotowe!
Super, mam nadzieję, że szybko zaczniecie K!
Pewnie, jak tylko rozwiążemy problem z A.


Rejestr
Wybrane
2
Programow
2
W trakcie
Gotowe
Wdrożenie
1
Live!
Jak możemy pomóc?
Chcę też F i G. Ale limit mnie powstrzymuje.
H
I
J
L
K
D
C
B
A
Nie potrzebujemy więcej osób. Ale ten problem regularnie powraca, napiszcie więc test, aby uniknąć go w przyszłości!


Rejestr
Wybrane
2
Programowanie
2
W trakcie
Gotowe
Wdrożenie
1
Live!
G
F
H
I
J
L
K
D
C
B
A
Czy w czymś mogę pomóc? Nie jestem techniczny, ale...
Pewnie! Przynieś więcej kawy i broń nas przed zakłóceniami.


Kilka dni później...
Rejestr
Wybrane
2
Programowanie
3
W trakcie
Gotowe
Wdrożenie
1
Live!
H
M
I
G
D
K
A
B
J
L
R
F
M czy H. Zdecydowanie M!
Nie, H! Hmmm. Nie, czekaj, M!
M czy H czy M czy H?!? Aaaaarrrrgh
Nie tęsknisz za dawnymi czasami?
Całonocnymi integracjami?
Katastrofami podczas wdrożeń?
LOL

Czy tablica Kanbana musi tak wyglądać?

Nie, powyższa tablica to tylko przykład!

Jedyne, co zaleca Kanban, to to, że przepływ pracy powinien być widoczny i że WIP powinien być ograniczony. Celem jest stworzenie płynnego przepływu i minimalizacja czasu dostarczenia. Musisz więc regularnie zadawać takie oto pytania:

Jakie powinniśmy mieć kolumny?
Każda kolumna reprezentuje stan w przepływie pracy lub kolejkę (bufor) pomiędzy dwoma stanami. Zacznij od najprostszej wizualizacji i dodawaj kolumny w miarę potrzeb.

Jakie powinny być limity Kanbana?
Kiedy limit dla „twojej" kolumny zostanie osiągnięty i nie będziesz mieć nic do zrobienia, zacznij się rozglądać za wąskimi gardłami w późniejszych stanach (na przykład nawarstwianie się prac z prawej strony tablicy) i pomóż w ich likwidacji. Jeżeli nie ma wąskiego gardła, oznacza to, że limit może być zbyt niski, ponieważ powodem zastosowania limitu była redukcja ryzyka zasilania wąskich gardeł.

Jeżeli zauważysz, że wiele zadań pozostaje w bezruchu, jest to wskazówka, iż limit jest zbyt wysoki.

- Zbyt niski limit => ludzie bez zadań => niska produktywność
- Zbyt wysoki limit = > zadania bez ludzi => wysoki czas dostarczenia

Jak rygorystyczne są limity Kanbana?
Niektóre zespoły traktują je jako rygorystyczne zasady (zespół nie może przekraczać limitu), inne traktują je jako wskazówki lub przyczynek do dyskusji (złamanie limitu jest dozwolone, ale powinno być wewnętrzną decyzją popartą argumentami). Tak więc raz jeszcze: to zależy od ciebie. Mówiłem, że Kanban nie jest specjalnie nakazowy?

16

Scrum kontra Kanban – podsumowanie

Podobieństwa

- Oba są Lean i Agile.
- Oba stanowią narzędzie harmonogramowania oparte na wciąganiu.
- Oba ograniczają WIP.
- Oba wykorzystują transparentność do sterowania ulepszaniem procesu.
- Oba skupiają się na częstym i szybkim dostarczaniu funkcjonalności.
- Oba oparte są na samoorganizujących się zespołach.
- Oba wymagają rozbicia pracy na mniejsze fragmenty.
- W obu plan wdrożeń jest wciąż optymalizowany na podstawie danych empirycznych (szybkość / czas dostarczenia).

Różnice

Scrum	Kanban
Nakazuje iteracje ograniczone czasowo.	**Iteracje ograniczone czasowo są opcjonalne.** Może mieć odrębne kadencje dla planowania, wdrażania i usprawniania procesu. Iteracje mogą być sterowane zdarzeniami zamiast określania ram czasowych.
Zespół zobowiązuje się do wykonania określonej liczby zadań podczas iteracji.	**Zobowiązania są opcjonalne.**
Wykorzystuje **szybkość** jako domyślny pomiar podczas planowania i usprawniania procesu.	Wykorzystuje **czas dostarczenia** jako domyślny pomiar podczas planowania i usprawniania procesu.
Nakazuje wykorzystanie zespołów **wielofunkcyjnych**.	**Zespoły wielofunkcyjne są opcjonalne**. Dozwolone są zespoły specjalistyczne.
Zadania muszą być rozbite, aby mogły zmieścić się w ramach jednego sprintu.	Nie nakazuje konkretnego rozmiaru zadań.
Nakazuje diagram spalania.	Nie nakazuje konkretnego typu diagramu.
WIP ograniczony pośrednio (per sprint).	**WIP ograniczony bezpośrednio** (per stan przepływu).
Nakazuje szacowanie.	**Szacowanie jest opcjonalne.**
Nie można dodawać zadań w trakcie iteracji.	**Można dodawać zadania, kiedy dostępna jest pojemność.**
Rejestr sprintu jest własnością jednego zespołu.	Tablica może być **współdzielona przez wiele zespołów** lub osób.
Nakazuje trzy role (PO/SM/T).	**Nie nakazuje żadnych ról.**

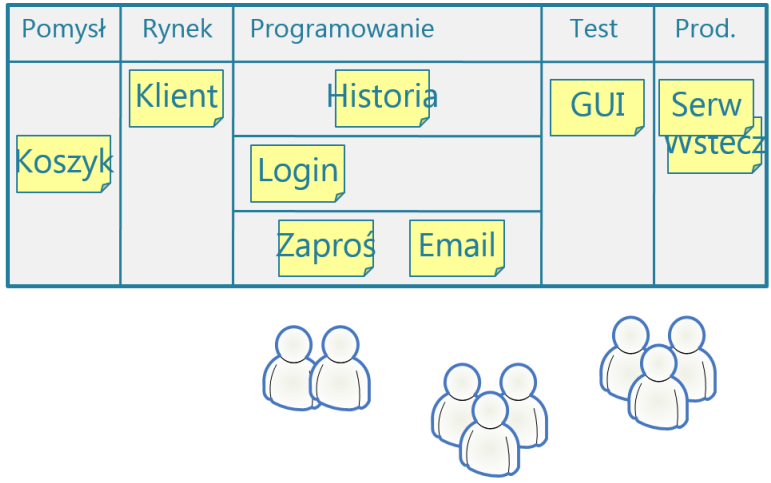
Tablica jest czyszczona pomiędzy sprintami.	Tablica jest stale aktualna.
Nakazuje stosowanie rejestru produktu uporządkowanego według priorytetu.	**Priorytetyzacja jest opcjonalna.**

To wszystko. Teraz znasz różnice.

To jednak jeszcze nie koniec – najlepsze dopiero przed tobą. Zakładaj buty, czas wskoczyć do okopów wraz z Mattiasem i zobaczyć, jak to wszystko wygląda w praktyce!

Część II – Studium przypadku

Kanban w prawdziwym życiu



Jest to historia o tym, jak uczyliśmy się usprawniać naszą pracę za pomocą Kanbana. Kiedy zaczynaliśmy, dostępnych było niewiele informacji, a wujek Google tym razem pozostawił nas z niczym. Dzisiaj Kanban rozwija się i dostępna jest ogromna ilość wiedzy na jego temat. Polecam przyjrzeć się pracy Davida Andersona, na przykład klasom usług. To pierwsze (i ostatnie) zrzeczenie się odpowiedzialności z mojej strony (obiecuję!). Niezależnie od tego, jakie rozwiązanie wdrożysz, upewnij się, że rozwiązuje ono twoje problemy. No to zaczynamy. Oto nasza historia.

/Mattias Skarin

17

Specyfika obsługi technicznej

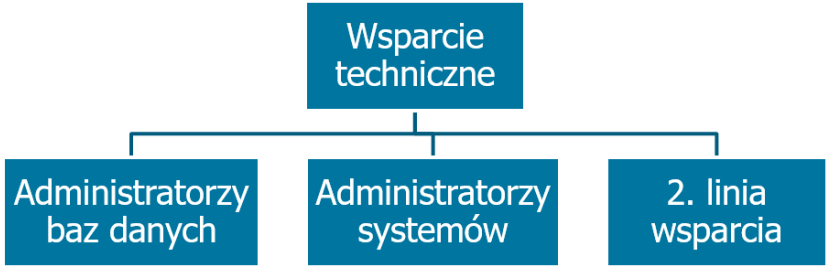
Jeżeli kiedykolwiek byłeś dostępny pod telefonem 7 dni w tygodniu przez 24 godziny na dobę, masz jakieś pojęcie na temat odpowiedzialności związanej z zarządzaniem środowiskiem produkcyjnym. Powinieneś być w stanie rozwiązywać problemy w środku nocy, niezależnie od tego, czy wyniknęły z twojej winy, czy nie. Nikt nic nie wie, dlatego zadzwonili do ciebie. To spore wyzwanie, ponieważ to nie ty stworzyłeś sprzęt, sterowniki, system operacyjny czy niestandardowe oprogramowanie. Twoje możliwości są zazwyczaj ograniczone do zidentyfikowania problemu, zminimalizowania jego wpływu na inne elementy systemu, zapisania dowodów pozwalających na powtórne odtworzenie problemu i czekania, aż osoba odpowiedzialna za spowodowanie awarii będzie w stanie odtworzyć i naprawić problem, którego właśnie byłeś świadkiem.

Dla zespołu wsparcia technicznego szybkość i precyzja rozwiązywania problemów to kluczowe kwestie.

18

Po kiego licha cokolwiek zmieniać?

W 2008 roku jeden z naszych klientów, skandynawska firma tworząca gry, przeszedł całą serię usprawnień procesu. Jednym z tych usprawnień było wdrożenie Scruma i stopniowe usuwanie blokad przeszkadzających zespołom programistycznym w dostarczaniu oprogramowania. Kiedy wydajność tworzenia oprogramowania wzrosła, zwiększyło się również obciążenie zespołów wsparcia technicznego. Wcześniej zespoły wsparcia technicznego głównie przyglądały się z boku, teraz były coraz bardziej angażowane w proces powstawania gotowych produktów.



Rysunek 1. Wsparcie techniczne złożone było z trzech zespołów: administratorzy baz danych (DBA), administratorzy systemów operacyjnych i druga linii wsparcia.

W rezultacie pomoc dla zespołów programistycznych była niewystarczająca. Gdybyśmy skupiali się tylko na programistach, opóźnienia występowałyby podczas pracy nad niezbędną infrastrukturą wykonywaną przez zespoły wsparcia technicznego. Konieczne były usprawnienia w obu obszarach.

Dodatkowo postępy w zespołach programistycznych sprawiały, że menadżerowie byli wciąż proszeni o nowe analizy i opinie na temat pomysłów. Oznaczało to, że mieli mniej czasu na priorytetyzację zadań i rozwiązywanie problemów. Zespół menadżerów zdał sobie sprawę, że musi działać, zanim sytuacja wymknie się spod kontroli.

19

Od czego zacząć?

Dobrym punktem wyjścia było zapytanie zespołów programistycznych, kto jest klientem wsparcia technicznego.

Wsparcie techniczne z punktu widzenia programistów

Zapytałem: „Jakie trzy pierwsze rzeczy przychodzą wam do głowy, gdy pomyślicie o wsparciu technicznym?" Najczęściej padały następujące odpowiedzi:

"Mają wiedzę z różnych dziedzin"

" Ich przepływ pracy jest do bani"

" Bardzo kompetentni w kwestiach związanych z infrastrukturą"

" Czym oni się zajmują?"

"Chcą pomagać, ale uzyskanie pomocy jest trudne"

" Aby osiągnąć prosty cel, trzeba wielu e-maili"

" Projekty trwają zbyt długo"

"Ciężcy w kontaktach"

Tak mniej więcej programiści postrzegają wsparcie techniczne. Teraz zobaczmy, jak wsparcie techniczne postrzega programistów..

Programiści z punktu widzenia wsparcia technicznego

„Dlaczego nie wykorzystujecie zalet istniejących platform?”

„Sprawmy, aby wdrożenia były mniej pracochłonne!”

„Naszą bolączką jest wasza kiepska jakość!”

„Powinni się zmienić” – słychać było po obu stronach. Oczywiście to nastawienie musiało się zmienić, jeżeli mieliśmy być w stanie naprawić problemy. Na szczęście dało się usłyszeć też coś pozytywnego: „Bardzo kompetentni w kwestiach związanych z infrastrukturą” (wiara w kompetencje), co budziło nadzieję, że podejście „my kontra oni” można wyeliminować, jeżeli zostaną stworzone odpowiednie warunki pracy. Likwidacja przeciążenia pracą i skupienie się na jakości – to była jedyna rozsądna opcja.

20

Do dzieła!

Musieliśmy zacząć – ale od czego? Jedyne, co wiedzieliśmy na pewno, to to, że miejsce, w którym zaczniemy, nie będzie miejscem, w którym skończymy.

Swoją karierę zawodową zaczynałem jako programista, więc niewiele wiedziałem o charakterze pracy zespołu wsparcia technicznego. Nie chciałem wparować i zacząć wszystko zmieniać. Potrzebowałem mniej konfrontacyjnego podejścia, które umożliwiłoby nauczenie się potrzebnych rzeczy, pominięcie rzeczy niepotrzebnych i byłoby wystarczająco proste.

Mogłem skorzystać z następujących narzędzi:

1. Scrum – dobrze się sprawdzał w zespołach programistycznych.

2. Kanban – był nowy i nieprzetestowany, ale dobrze pasował do zasad Lean, których tak bardzo brakowało.

W dyskusjach z menadżerami zasady Kanbana i Lean zdawały się pasować do problemów, które chcieliśmy rozwiązać. Z ich punktu widzenia sprinty nie pasowałyby do sposobu pracy, ponieważ zmiany priorytetów odbywały się praktycznie codziennie. Kanban był więc logicznym punktem wyjścia, nawet jeżeli był nowością dla nas wszystkich.

21

Przygotowanie zespołów

Jak przygotować zespoły? Nie było żadnego dostępnego podręcznika, który by mówił, jak zacząć. Zrobienie tego źle byłoby dużym ryzykiem. Poza problemami z ulepszaniem procesu mieliśmy do czynienia z ludźmi o bardzo wyspecjalizowanych umiejętnościach, których ciężko byłoby zastąpić. Odsunięcie ich byłoby baaardzo złym pomysłem.

- Czy powinniśmy po prostu wprowadzać zmiany i radzić sobie z konsekwencjami, kiedy się pojawią?

- Czy powinniśmy może najpierw zorganizować warsztaty?

Dla nas było to oczywiste: powinniśmy najpierw zorganizować warsztaty. Ale jak? Wyzwaniem było osiągnąć, aby cały zespół wsparcia technicznego mógł wziąć udział w warsztatach (no bo kto odbierze telefon?). Ostatecznie zdecydowaliśmy się zrobić warsztaty i postarać się, aby były jak najprostsze i oparte na ćwiczeniach.

Warsztaty

Jedną z zalet warsztatów było to, że pomogły na wczesnym etapie zidentyfikować problemy. Zapewniały również środowisko o wysokim stopniu zaufania, w którym implikacje mogły być omawiane bezpośrednio z członkami zespołu. Spójrzmy prawdzie w oczy: nie wszyscy byli entuzjastycznie nastawieni do zmian, jakie chcieliśmy wprowadzić w ich sposobie pracy. Na szczęście jednak większość członków zespołu była otwarta i naprawdę się starała. Na warsztatach zaprezentowaliśmy więc najważniejsze zasady i wykonaliśmy niewielką symulację Kanbana.

Nauka podstawowych zasad	Demo Kanbana
• Ograniczenie pracy do poziomu pojemności • Rozmiar paczki kontra czas cyklu • Praca w toku kontra przepustowość • Teoria ograniczeń	• 3 rodzaje pracy: odpowiedzi na pytania budowa samochodu z lego projekt i budowa domu • 3 iteracje mierzenie szybkości dla poszczególnych typów pracy eksperymenty, dostosowanie WIP • Odprawa

Na koniec warsztatów zorganizowaliśmy głosowanie „pięść pięciu palców", aby sprawdzić, czy zespół jest gotowy wypróbować ten sposób pracy w rzeczywistości. Nie było żadnych obiekcji, mogliśmy więc kontynuować. *(Pięść pięciu palców to technika osiągania konsensusu, w której uczestnicy za pomocą palców jednej ręki określają swoją zgodę w skali od 1 do 5, gdzie 5 oznacza całkowitą zgodę, 1 – brak zgody, 3 – zgodę z zastrzeżeniami; konsensus ma miejsce, kiedy wszyscy członkowie grupy pokażą przynajmniej 3 palce).*

22

Osoby poboczne

Prawdopodobne było, że osoby poboczne też odczują skutki wdrożenia Kanbana. Byłyby to jednak zmiany na lepsze – oznaczałyby, że zespół mówiłby „nie” w przypadku zadań, których nie mógłby ukończyć, zacząłby dbać o jakość i usuwałby z rejestru zadania o niskim priorytecie. Oprócz tego podjęcie dyskusji przed deklaracją wykonania zadania to zawsze dobry pomysł.

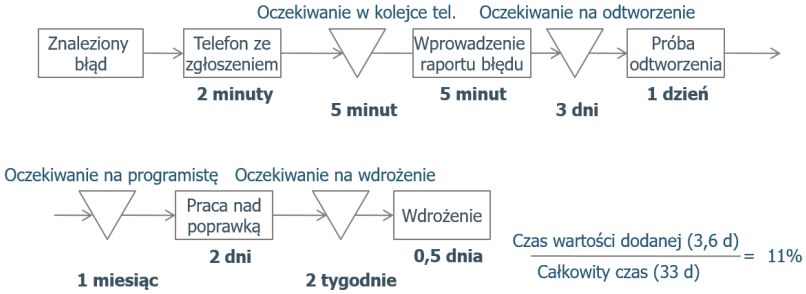
Najbliższe takie osoby to pierwsza linia wsparcia i menadżerowie działu. Ponieważ brali udział w warsztatach, byli pozytywnie nastawieni do wprowadzanych zmian. Podobnie było w przypadku zespołów programistycznych (które i tak spodziewały się usprawnień). Jednak w przypadku jednego zespołu – zespołu wsparcia – rzeczy miały się inaczej. Ich największym problemem było przeciążenie pracą. Ponadto zajmowali się problemami klientów, a firma była zobowiązana odpowiedzieć na wszystkie problemy. Miało się to zmienić, kiedy wdrożymy Kanbana i ograniczymy WIP.

Spotkaliśmy się więc z kluczowymi osobami i przedstawiliśmy nasze intencje, spodziewane zyski i prawdopodobne konsekwencje. Ulżyło mi, gdy się okazało, że nasze pomysły zostały dobrze przyjęte, czasami z uwagami w stylu: „będzie świetnie, jeżeli wreszcie pozbędziemy się tych problemów”.

23

Tworzenie pierwszej tablicy

Dobrym sposobem na tworzenie pierwszej tablicy Kanbana jest wykonanie mapy przepływu wartości. Jest to wizualizacja łańcucha wartości i daje wgląd w stany pracy, przepływ i czas w systemie (czas cyklu).



Zaczęliśmy jednak o wiele prościej, mianowicie z przykładową tablicą Kanbana, którą wraz z menadżerem narysowaliśmy na papierze. Omówiliśmy ją kilka razy i zaczęliśmy. Padły następujące pytania:

- Jakie mamy rodzaje pracy?
- Kto się nią zajmuje?
- Powinniśmy współdzielić odpowiedzialność pomiędzy różnymi rodzajami pracy?
- Jak radzimy sobie ze współdzieloną odpowiedzialnością w przypadku wyspecjalizowanych umiejętności?

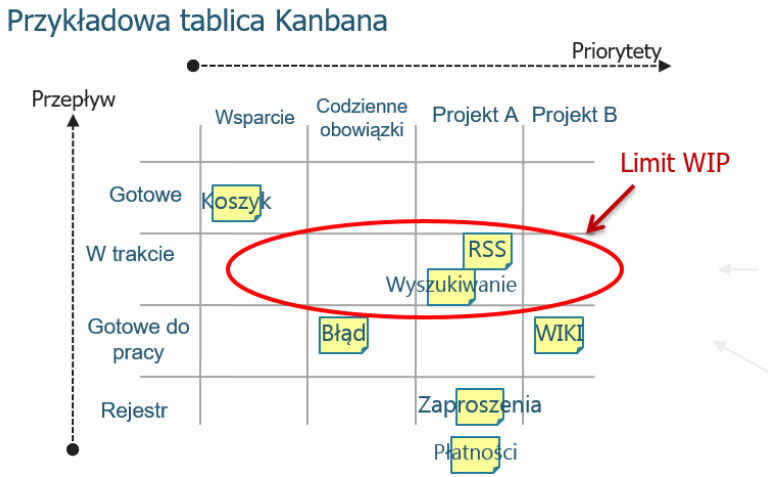
Ponieważ różne rodzaje pracy objęte są różnymi umowami SLA, naturalne wydawało się pozwolić każdemu zespołowi, by opracował własną tablicę. Sami stworzyli wiersze i kolumny.

Kolejną znaczącą decyzją było to, czy odpowiedzialność powinna być współdzielona pomiędzy różnymi typami pracy. „Czy pewna wydzielona

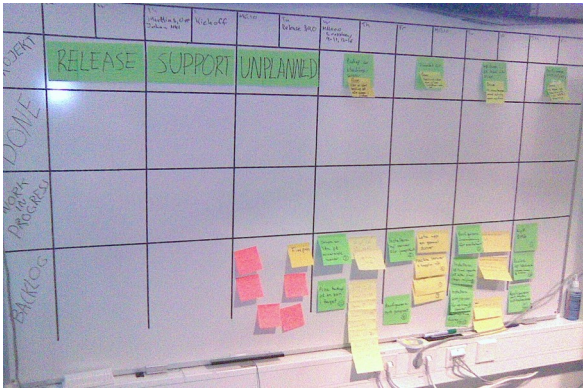
część zespołu powinna zajmować się bezpośrednimi pytaniami (praca reaktywna), a reszcie zespołu pozwolić skupić się na projektach (praca proaktywna)?". Zdecydowaliśmy, że na początku spróbujemy wdrożyć odpowiedzialność współdzieloną. Głównym tego powodem było to, że uznaliśmy samoorganizację, ciągłą naukę i transfer wiedzy za nieodzowne elementy potrzebne do utrzymania ciągłego rozwoju. Minusem tego podejścia były potencjalne zakłócenia dla wszystkich, ale było to najlepsze rozwiązanie, jakie wtedy przychodziło nam do głowy. W tym miejscu muszę poczynić drobną uwagę: podczas warsztatów zespoły same zorganizowały się w związku z tym problemem. Pozwoliły jednej osobie zajmować się krótkimi i pilnymi zgłoszeniami, a reszta zajmowała się większymi problemami.

Pierwszy model Kanbana

Poniżej znajduje się podstawowy model, jaki wykorzystaliśmy. Zauważ, że zespół zdecydował się, aby zadania płynęły w górę (jak bąbelki w wodzie) zamiast bardziej typowego sposobu przepływu od lewej do prawej.



Rysunek 2. Jest to pierwszy model tablicy Kanbana. Priorytety przebiegają od lewej do prawej, przepływ biegnie w górę. WIP liczony jest jako całkowita liczba zadań w wierszu „W trakcie" (zakreślone elipsą). Model został zdefiniowany na podstawie doświadczeń zgłoszonych przez Lindę Cook.



Rysunek 3. Pierwsza tablica Kanbana zespołu zarządzania systemami operacyjnymi.

Wykorzystane wiersze

Stan przepływu (wiersz)	Jak został zdefiniowany
Rejestr	Historyjki, które według menadżera są potrzebne.
Gotowe do pracy	Historyjki, które są oszacowane i rozbite na zadania zajmujące nie więcej niż 8 godzin.
W trakcie	Wiersz z limitem WIP. Zaczęliśmy od 2 x wielkość zespołu – 1 (– 1 jest na potrzebny współpracy). Zatem zespół liczący 4 osoby miałby limit równy 7.
Gotowe	Gotowe do wykorzystania przez użytkownika.

Wykorzystane kolumny

Rodzaj pracy	Jak został zdefiniowany
Wdrożenie	Pomoc zespołom programistycznym we wdrażaniu funkcjonalności.
Wsparcie	Mniejsze zgłoszenia od innych zespołów.
Nieplanowane	Nieprzewidziane prace, które należy wykonać, ale nie mają jasno określonego właściciela, np. drobne usprawnienia infrastruktury.
Project A	Większe projekty techniczne, np. zmiana sprzętu środowiska przejściowego.
Project B	Inny większy projekt.

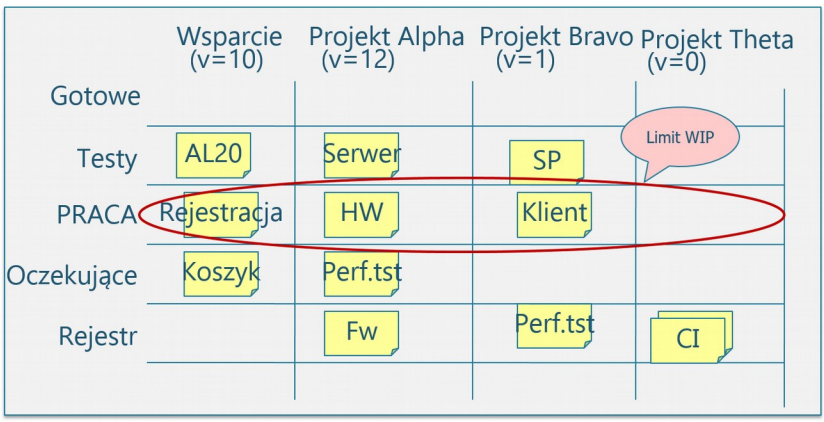
Nie wszystkie tablice Kanbana wyglądały tak samo. Wszystkie zaczynały jako prosty szkic i ewoluowały w trakcie pracy.

24

Ustawienie pierwszego limitu WIP

Nasz pierwszy limit WIP był bardzo szczodry. Tłumaczyliśmy to tym, że wizualizując przepływ, powinniśmy zobaczyć i doświadczyć, co dzieje się w procesie, a to, że będziemy w stanie zgadnąć najlepszy limit już na początku, było bardzo mało prawdopodobne. W miarę upływu czasu dostosowywalibyśmy limity WIP za każdym razem, kiedy znaleźlibyśmy ku temu powód (musielibyśmy tylko spojrzeć na tablicę).

Pierwszy limit został określony za pomocą wzoru 2n – 1 (n to liczba członków zespołu, – 1 miało na celu zachęcanie do współpracy). Dlaczego? Nie mieliśmy po prostu żadnego lepszego pomysłu ☺. Taki początkowy limit wydawał się również mało kontrowersyjny. Wzór gwarantował proste i logiczne wytłumaczenie dla każdego, kto próbował obarczyć zespół jakąś pracą: „Biorąc pod uwagę, że każdy członek zespołu może pracować maksymalnie nad dwoma zadaniami jednocześnie, jednym aktywnym i jednym oczekującym, dlaczego sądzicie, że podejmą się większej liczby zadań?". Patrząc z perspektywy czasu, każdy wysoki limit byłby odpowiedni na początek. Monitorując tablicę Kanbana, łatwo jest określić odpowiednie limity na późniejszym etapie.



Rysunek 4. Oto, jak ustaliliśmy limit WIP dla administratorów systemów i administratorów baz danych – jeden limit dla wszystkich typów pracy.

Zaobserwowaliśmy, że definiowanie limitu WIP w punktach historyjkowych byłoby bezużyteczne. Byłoby to zbyt trudne do śledzenia. Jedynym wystarczająco prostym do śledzenia sposobem ograniczania WIP było po prostu zliczanie liczby zadań (czyli jednocześnie wykonywanych prac).

Dla zespołu wsparcia wykorzystaliśmy limit WIP zdefiniowany dla poszczególnych kolumn. Zrobiliśmy tak dlatego, że potrzebowaliśmy szybszej reakcji, jeżeli limit był łamany.

25

Przestrzeganie limitu WIP

Chociaż przestrzeganie limitu WIP w teorii wydaje się proste, w praktyce jest to dość trudne. Wiąże się z mówieniem „nie”. Próbowaliśmy sobie z tym poradzić na różne sposoby.

Dyskusje przy tablicy

Jeżeli odkryliśmy próbę złamania limitu WIP, braliśmy delikwenta, który się tego dopuścił, i pytaliśmy, co chce osiągnąć. Na początku najczęstszym powodem był brak doświadczenia. W niektórych przypadkach odkryliśmy różnice w postrzeganiu priorytetów, gdzie typowym przypadkiem był członek zespołu specjalistycznego pracujący w ramach określonego obszaru. Były to jedyne przypadki, w których napotkaliśmy problemy, a większość tych problemów była rozwiązywana na bieżąco przy tablicy.

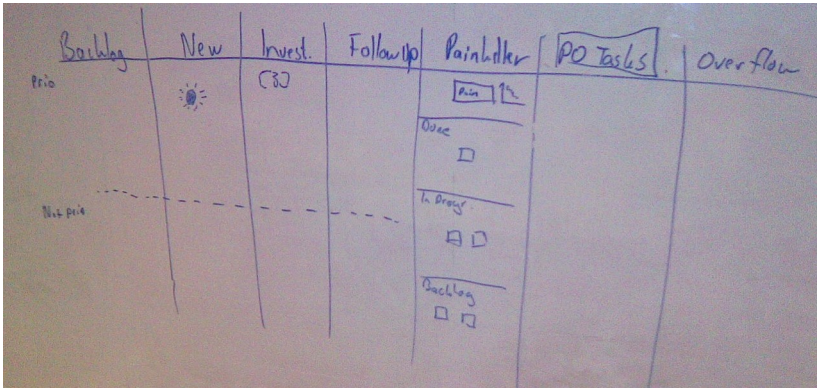
Przyznanie sekcji przepełnienia

Jeżeli powiedzenie „nie” było zbyt konfrontacyjne, a usunięcie zadań z tablicy zbyt trudne, to kiedy limit WIP był przekraczany, przesuwaliśmy zadania o niskim priorytecie do sekcji przepełnienia. Dla takich zadań określiliśmy dwie reguły:

1. Nie można o nich zapominać – należy się nimi zająć, gdy tylko będzie ku temu okazja.

2. Jeśli ich zaniechamy, zostaniesz o tym poinformowany.

Po zaledwie dwóch tygodniach stało się jasne, że te zadania nigdy nie byłby kończone, więc wraz z menadżerem mogliśmy się ich ostatecznie pozbyć.

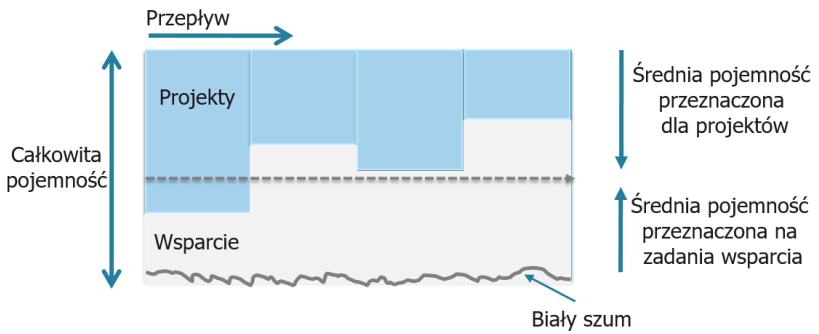


Rysunek 5. Szkic tablicy dla zespołu wsparcia; sekcja przepełnienia to skrajna prawa kolumna.

26

Które zadanie trafi na tablicę?

Na wczesnym etapie zdecydowaliśmy, że nie wszystkie zadania wykonywane przez zespół będziemy dodawać na tablicę. Monitorowanie takich rzeczy jak rozmowy telefoniczne czy parzenie kawy zmieniłoby tablicę w zaśmieconego potworka. Mieliśmy rozwiązywać problemy, a nie je tworzyć ☺. Postanowiliśmy więc umieszczać na tablicy tylko zadania zajmujące więcej niż godzinę, a wszystko, co trwało krócej, było uznawane za „biały szum". To ograniczenie działało dość dobrze i było jednym z niewielu ustaleń, które pozostały niezmienione. (Musieliśmy rozważyć założenie, jaki wpływ ma biały szum na całość pracy, ale więcej informacji o tym znajdziesz w dalszej części).



Rysunek 6. Zaczęliśmy od założenia, że całkowita pojemność jest wykorzystywana głównie przez dwa typy pracy: większe (projekty) i mniejsze (wsparcie). Śledzenie szybkości projektów mogło dać nam informację na temat potencjalnej daty zakończenia. Biały szum (małe zadania wsparcia trwające krócej niż 1 godzinę, spotkania, przerwy na kawę, pomoc kolegom) był stałym elementem codziennych zadań.

27

Jak szacować?

Jest to stały temat rozważań i zdecydowanie jest na to pytanie więcej niż jedna odpowiedź:

- Szacować regularnie.
- Szacować, kiedy zajdzie taka potrzeba.
- Wykorzystywać do szacowania idealne dni i punkty historyjkowe.
- Szacowanie jest niepewne, stosować rozmiary koszulek (S – małe, M – średnie, L – duże).
- Nie szacować lub szacować tylko wtedy, kiedy z opóźnieniem związany jest koszt usprawiedliwiający czas poświęcony na szacowanie.

Będąc pod pewnym wpływem Scruma (w końcu była to metodyka, w której najwięcej pracowaliśmy), postanowiliśmy rozpocząć szacowanie za pomocą punktów historyjkowych. W praktyce jednak zespoły traktowały punkty jako odpowiednik osobogodzin (było to dla nich bardziej naturalne). Z czasem menadżerowie nauczyli się, że jeżeli liczba współbieżnych projektów będzie niska, projekty będą wykonywane szybciej. Nauczyli się również, że w przypadku nagłej zmiany mogą zmienić priorytety i poradzić sobie z ewentualnym problemem.

Potrzeba szacowania daty dostarczenia nie była już problemem. W związku z tym menadżerowie przestali prosić o te szacunki. Robili to tylko wtedy, kiedy obawiali się, że ktoś będzie czekał na zakończenie projektu.

W pewnym momencie, na wczesnym etapie, jeden z menadżerów, zestresowany odebranym telefonem, obiecał zakończyć projekt „do końca tygodnia". Był to projekt z tablicy Kanbana, więc łatwo było oszacować stan zaawansowania (policzyć zakończone historyjki) i dojść do wniosku,

że po tygodniu pracy stan zaawansowania wynosił 25%. Oznaczało to, że potrzebne były jeszcze trzy tygodnie. Znając fakty, menadżer zmienił priorytety projektu, zatrzymał równoczesne prace i sprawił, że szybsze dostarczenie było możliwe. Zawsze należy sprawdzać z tablicą ☺.

Co oznacza czas szacowany? Czas dostarczenia czy czas pracy?

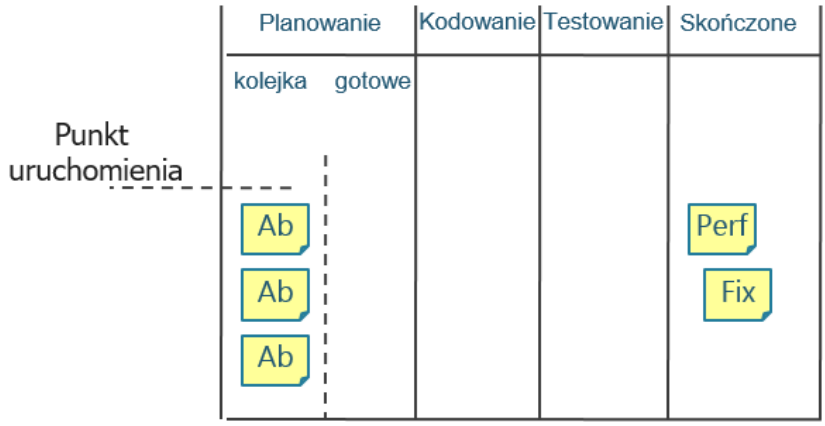
Nasze punkty historyjkowe odzwierciedlają czas pracy, na przykład ile godzin nieprzerwanej pracy zajmie dana historyjka; nie jest to czas dostarczenia (ani czas kalendarzowy, ani liczba godzin oczekiwania). Mierząc liczbę punktów ukończonych każdego tygodnia (szybkość), możemy określić czas dostarczenia.

Szacowaliśmy każdą nową historyjkę tylko raz, nie poprawialiśmy szacunków w trakcie pracy. To pozwoliło nam zminimalizować czas spędzony na szacowaniu.

28

Jak więc pracowaliśmy?

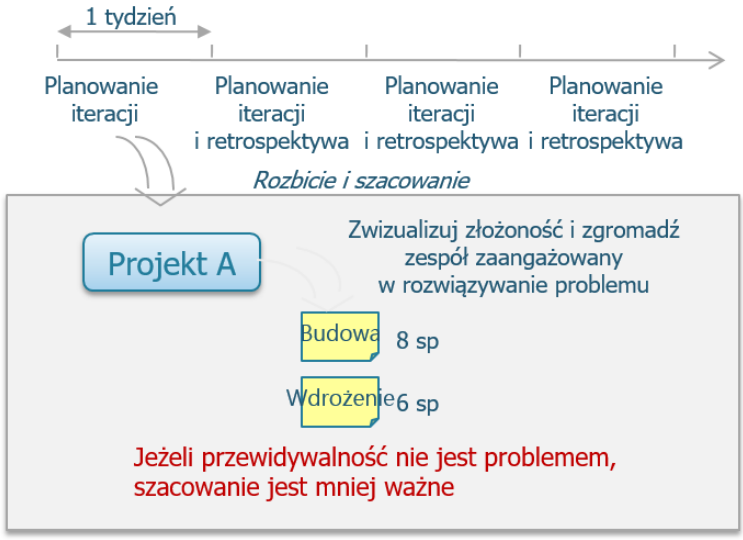
W Kanbanie naprawdę nie ma wielu ograniczeń, można pracować na różne sposoby. Możesz pozwolić zespołowi pracować zgodnie z aktywnościami bazującymi na czasie lub możesz zdecydować, aby zadania były wykonywane, kiedy staną się wystarczająco ważne.



Rysunek 7. Pojawienie się w rejestrze trzech zadań wywołuje zorganizowanie sesji planowania/szacowania.

Postanowiliśmy zaplanować dwa powtarzające się spotkania:

- Codzienne spotkanie – z zespołem przy tablicy, mające na celu wykrycie problemów i zrozumienie zadań pozostałych członków zespołu.
- Cotygodniowe planowanie iteracji – mające na celu planowanie i omawianie usprawnień procesu.



Takie rozwiązanie w naszym przypadku sprawdzało się dobrze.

Codzienne spotkanie

Codzienne spotkanie było podobne do tak zwanego „codziennego scruma". Było organizowane po codziennym „scrumie scrumów", czyli spotkaniu z udziałem wszystkich zespołów (programistycznych, testowych i wsparcia). Scrum scrumów dawał zespołom Kanbana ważne informacje, takie jak to, którymi problemami należy zająć się w pierwszej kolejności albo który zespół programistyczny cierpi aktualnie najbardziej. Na początku menadżerowie często brali udział w tych codziennych spotkaniach, proponowali rozwiązania i ustalali priorytety. Z czasem, kiedy zespoły stawały się coraz lepsze w samoorganizacji, menadżerowie pojawiali się na nich coraz rzadziej (byli jednak niedaleko na wypadek, gdyby byli potrzebni).

Planowanie iteracji

Raz na tydzień organizowaliśmy planowanie iteracji. Spotkanie odbywało się zawsze o tej samej porze, ponieważ przekonaliśmy się, że jeżeli go nie zaplanowaliśmy, czas ten był przeznaczany na inne priorytety ☺. Potrzebowaliśmy również więcej czasu na rozmowę wewnątrz zespołu. Zwyczajowa agenda była następująca:

- Aktualizacja wykresów i tablicy (zakończone projekty były przenoszone na „ścianę zakończonych").

- Spojrzenie wstecz na poprzedni tydzień. Co się stało? Dlaczego? Co możemy zrobić, aby to poprawić?
- Dostosowanie limitów WIP [jeżeli była taka konieczność].
- Rozbicie zadań i szacowanie nowego projektu [jeżeli była taka konieczność].

Planowanie iteracji było tak naprawdę jednocześnie spotkaniem szacunkowym i rytmem ciągłego ulepszania. Małe i średnie problemy były rozwiązywane natychmiast przy wsparciu ze strony menadżerów pierwszej linii. Jednak śledzenie skomplikowanych problemów związanych z infrastrukturą było trudniejsze. Aby sobie z tym poradzić, wprowadziliśmy możliwość przypisania menadżerom do dwóch problemów zespołu.

Zasady były następujące::

1. Menadżer w danej chwili może pracować nad dwoma problemami.
2. Jeżeli pracuje nad dwoma problemami, można dodać następny tylko wówczas, gdy zostanie usunięty ten mniej ważny.
3. Zespół decyduje, kiedy problem jest rozwiązany.

Przyniosło to zmianę na lepsze. Nagle zespoły widziały, że menadżerowie próbują pomóc w rozwiązywaniu nawet najtrudniejszych problemów. Mogły wskazać problem i zapytać: „Jak idzie?”. Problemy nie były zapominane ani pomijane z powodu nowych bardzo ważnych zadań.

Jednym z przykładów takiego problemu była niemożność uzyskania pomocy od programistów, kiedy wsparcie podejrzewało błąd programu. Potrzebny był programista, który pomoże zidentyfikować element systemu powodujący problem, ale ponieważ programiści zajęci byli w sprintach, tworząc nowe funkcjonalności, problemy się nawarstwiały.

Zespoły wsparcia uważały, że programiści niewystarczająco dbali o jakość.

Kiedy ten problem wypłynął, został przekazany najpierw do menadżera linii, a potem do menadżera działu. Podczas dyskusji z szefem działu menadżerowie zgodzili się, że jakość jest najważniejsza. Wymyślili rozwiązanie, w którym w każdym sprincie jeden z zespołów byłby dostępny dla wsparcia. Po zapewnieniu sobie wsparcia po stronie menadżerów szef działu przekazał zespołom wsparcia listę osób kontaktowych wśród programistów. Rozwiązanie zostało natychmiast przetestowane, z podejrzeniem, że nie zadziała. Tym razem było jednak inaczej: niezbędne przygotowania zostały poczynione, a problem został uznany za rozwiązany. Przyniosło to wielką ulgę zespołom wsparcia.

29

W poszukiwaniu działającego schematu planowania

Historia

Pamiętam punkt zwrotny w pracy jednego z zespołów. Siedziałem z nimi na ich drugiej sesji szacunkowej. Utknęli z projektem, którego nie byli w stanie oszacować. Było zbyt wiele niewiadomych i szacowanie się zatrzymało. Zamiast wkroczyć i przejąć kontrolę poprosiłem, aby zmienili proces tak, by mogli znaleźć rozwiązanie. Pod przewodnictwem swojego menadżera podjęli wyzwanie i zaczęli projektować swoje własne rozwiązanie. Zdarzenie to było ważnym punktem zwrotnym, ważnym zwycięstwem, dzięki któremu stali się pewnym siebie zespołem. Potem zaczęli rozwijać się tak dynamicznie, że musieliśmy schodzić im z drogi.

Dwa miesiące później, po retrospektywie, ich menadżer podszedł do mnie i, pokazując na ich tablicę Kanbana, powiedział: „Mam problem: nie mamy żadnych prawdziwych problemów. Co robić?”.

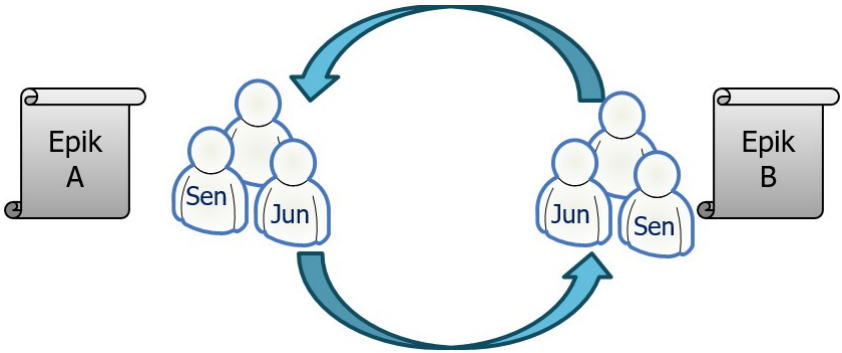
Nowe podejście do planowania

Sesje z pokerem planistycznym nie sprawdziły się w żadnym z zespołów wsparcia. Oto kilka powodów:

1. Wiedza w ramach zespołu była zbyt nierówno rozłożona.
2. Zazwyczaj tylko jedna osoba mówiła.
3. Członkowie zespołu chcieli wracać do ważnych spraw, od których zostali oderwani.

Jednak, dzięki eksperymentom, zespoły niezależnie opracowały dwa różne procesy szacowania. Każdy dobrze działał w danym zespole.

Podejście 1: zmiana i kontrola



- Dla każdego projektu/historyjki przypisywano parę seniora i juniora mającą wykonać szacunek (na przykład jedna osoba dobrze znająca daną historyjkę i jedna nieznająca jej). To pomagało w dzieleniu się wiedzą.

- Pozostali członkowie zespołu decydowali, przy której historyjce chcą pomóc w szacowaniu (ale z ograniczeniem do czterech osób na historyjkę, dzięki czemu dyskusja była bardziej efektywna).

- Każdy z zespołów szacunkowych rozbijał swoją historyjkę na zadania i, jeżeli była taka konieczność, szacował je.

- Następnie zespoły zamieniały się historyjkami i sprawdzały wzajemnie swoją pracę (jedna osoba z każdego zespołu pozostawała przy historyjce i tłumaczyła pracę osobom kontrolującym).

- Gotowe!

Przeważnie cała sesja szacunkowa trwała około 45 minut, a poziom energii utrzymywał się na wysokim poziomie przez cały czas trwania spotkania. Kiedy historyjki były zamieniane i oceniane świeżym spojrzeniem, zazwyczaj wprowadzano jedną lub dwie zmiany.

Podejście 2: wysokopoziomowa ocena seniora, potem szacowanie

Dwóch seniorów w zespole wykonywało wysokopoziomową analizę historyjki/projektu. Analizowali oni rozwiązania architektoniczne i wybierali odpowiednią architekturę dla danego projektu. Po jej ustaleniu zespół rozbijał historyjkę na zadania, korzystając z zaproponowanej architektury jako punktu początkowego.

Rysunek 8. Rozbicie na zadania z kontrolą innego zespołu podczas planowania iteracji.

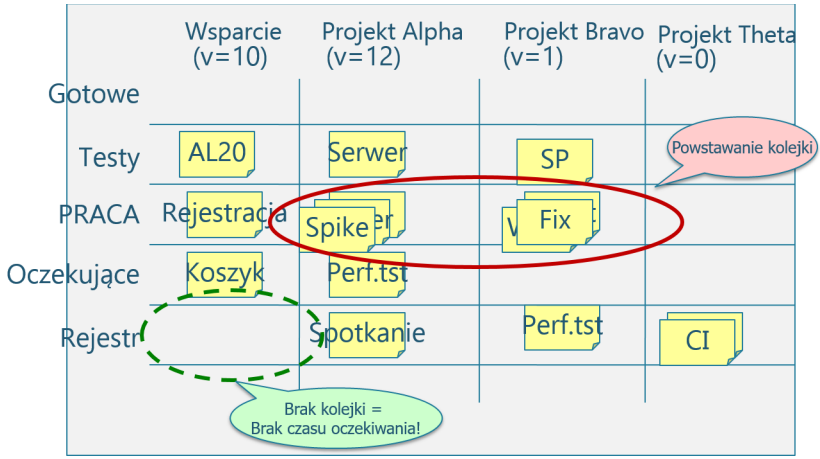
30

Co mierzyć?

Zmierzyć można wiele rzeczy: czas cyklu (od momentu ujawnienia potrzeby do momentu jej zaspokojenia), szybkość, kolejki, spalanie… Należy się jednak zastanowić, które pomiary mogą być wykorzystane do usprawnienia procesu. Radzę eksperymentować i sprawdzać, co działa w twoim przypadku. My stwierdziliśmy, że diagramy spalania były przesadą dla każdego projektu dłuższego niż 4 tygodnie. Ogólne postępy można było ocenić, spoglądając na tablicę Kanbana (ile historyjek znajdowało się w rejestrze, a ile w kolumnie z zadaniami zakończonymi).

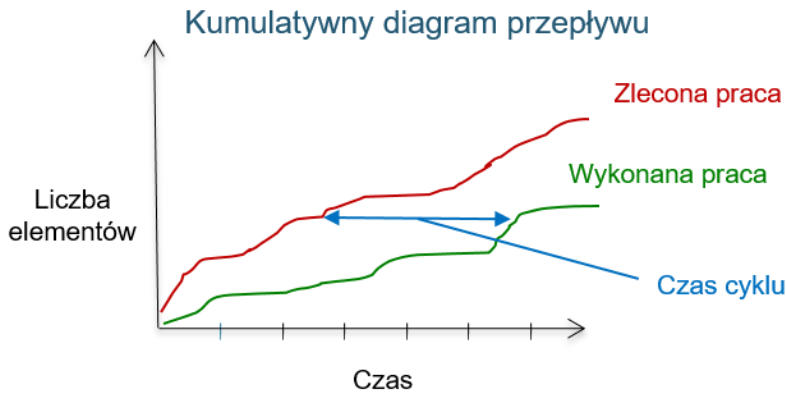
Potencjalny pomiar	**Zalety**	**Wady**
Czas cyklu	Łatwy do zmierzenia. Nie wymaga szacowania. Zaczyna się i kończy na kliencie.	Nie uwzględnia rozmiaru.
Całkowita szybkość (zagregowana dla wszystkich typów prac)	Zgrubny, ale łatwy wskaźnik kierunku usprawnień i odchyleń.	Nie pomaga w określaniu dat dostarczenia dla danych typów pracy.
Szybkość dla typu pracy	Bardziej dokładna niż szybkość całkowita.	Aby była użyteczna, musi zaczynać się od potrzeby klienta i trwać do dostarczenia. Śledzenie, w porównaniu z szybkością całkowitą, trwa dłużej.
Długości kolejek	Szybki wskaźnik fluktuacji zapotrzebowania. Łatwe do zwizualizowania.	Nie mówi, czy przyczyną jest nierówne zapotrzebowanie czy nierówna pojemność. Pusta kolejka może wskazywać na nadmierną pojemność.

Zaczęliśmy od mierzenia szybkości dla typu pracy i długości kolejek. Szybkość dla typu pracy jest łatwa do zmierzenia i spełnia swoje zadanie. Długości kolejek są dobrym wskaźnikiem, ponieważ widać je od razu (kiedy już wiesz, gdzie ich szukać).



Rysunek 9. Wąskie gardła i okazje. Obszar zakreślony na czerwono pokazuje, jak buduje się kolejka, ujawniając wąskie gardło w kolumnie testów. Brak kolejek w kolumnie wsparcia wskazuje, że dla nowej pracy nie ma żadnego czasu oczekiwania. Jest to dobry znak, zwiastujący wysoki poziom wsparcia klienta.

Nie wykorzystywaliśmy kumulatywnego diagramu przepływu, ale mogłoby to być interesujące.



Nie wykorzystywaliśmy kumulatywnego diagramu przepływu, ponieważ tablica Kanbana i diagram szybkości dawały nam wystarczająco dużo informacji, przynajmniej w początkowych fazach pracy. Wąskie gardła,

nierówne rozłożenie pracy i nadmiar zadań mogły być łatwo identyfikowane, a rozwiązywanie tych problemów zajmowało nas przez pierwsze 6 miesięcy.

31

Jak wszystko zaczęło się zmieniać

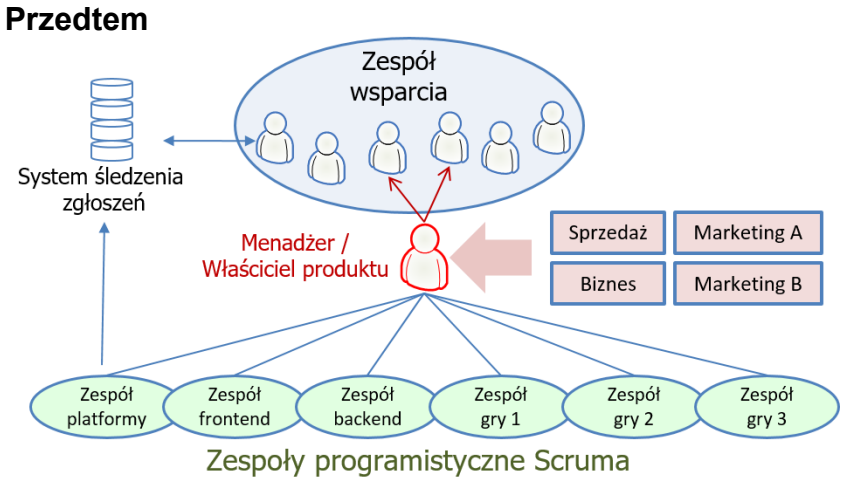
Trzy miesiące po wprowadzeniu Kanbana zespół zarządzający systemami otrzymał od zarządu tytuł „najbardziej wydajnego zespołu" w dziale IT. W tym samym czasie zespół ten został uznany za jedno z trzech „najlepszych doświadczeń" w firmowej retrospektywie. Firmowa retrospektywa to wydarzenie obejmujące całą firmę, które ma miejsce co 6 tygodni, i był to pierwszy raz, kiedy zespół stanął na podium! A zaledwie trzy miesiące wcześniej te zespoły były wąskimi gardłami, na które narzekała większość ludzi.

Jakość usług ewidentnie się poprawiła. Jak to się stało?

Przełomem był moment, w którym wszyscy zaczęli współpracować. Menadżerowie zapewnili jasne cele i chronili zespół przed obowiązkami, które do niego nie należały, a zespoły przyjęły odpowiedzialność za jakość i terminy. Zanim to się stało, minęło około 3 – 4 miesięcy, ale potem wszystko ruszyło płynnie. Nie wszystkie problemy zniknęły (wtedy wszyscy stracilibyśmy pracę ☹), ale napotkaliśmy również nowe wyzwania, na przykład: jak utrzymać w zespole wysoką motywację do usprawniania procesu (skoro nie jest już wąskim gardłem)?

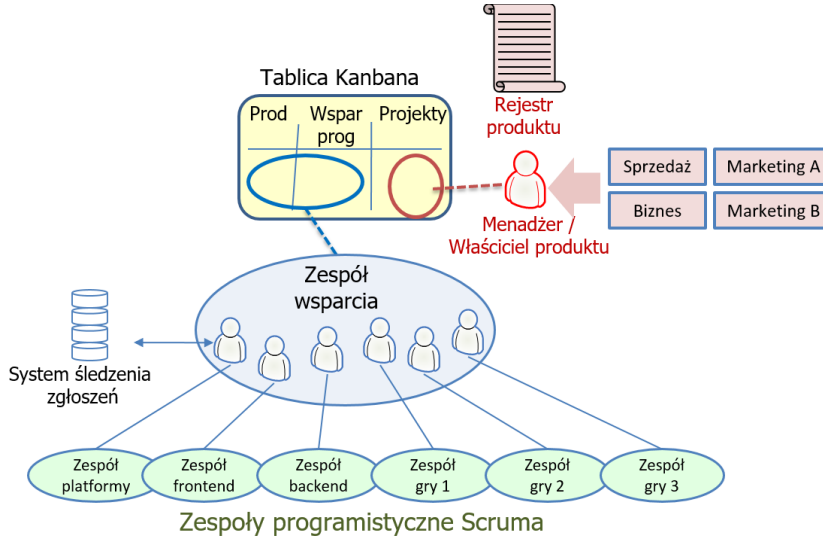
Ważnym elementem samoukładającej się układanki było wprowadzenie koncepcji „jednej osoby kontaktowej na zespół". Oznaczało to, że każdy zespół programistów otrzymał przypisaną do siebie osobę do kontaktów w zespołach wsparcia. Było to możliwe dzięki Kanbanowi, ponieważ członkowie zespołów mogli sami organizować swoją pracę, zapobiegając przepracowaniu i ułatwiając ciągłe udoskonalanie. Wcześniej było tak, że losowo wybrana osoba pobierała pracę z kolejki, wykonywała ją najlepiej jak umiała, a potem zajmowała się następnym zadaniem. Wszelkie błędy w komunikacji powodowały konieczność rozpoczęcia pracy od nowa, od nowego zgłoszenia zespołowi wsparcia. Kiedy wprowadzona została koncepcja „jeden do jednego", zespół wsparcia uzyskał możliwość szybkiego reagowania, gdy błędne dane lub problemy z jakością zagrażały systemowi.

Szybko powstały protokoły komunikacji. Pracownicy wsparcia zaczęli korzystać z komunikatorów w kontaktach z programistami, których dobrze znali; pisać e-maile do osób, które lepiej pisały, niż mówiły; dzwonić, jeżeli była to najszybsza droga do rozwiązania problemów ☺.



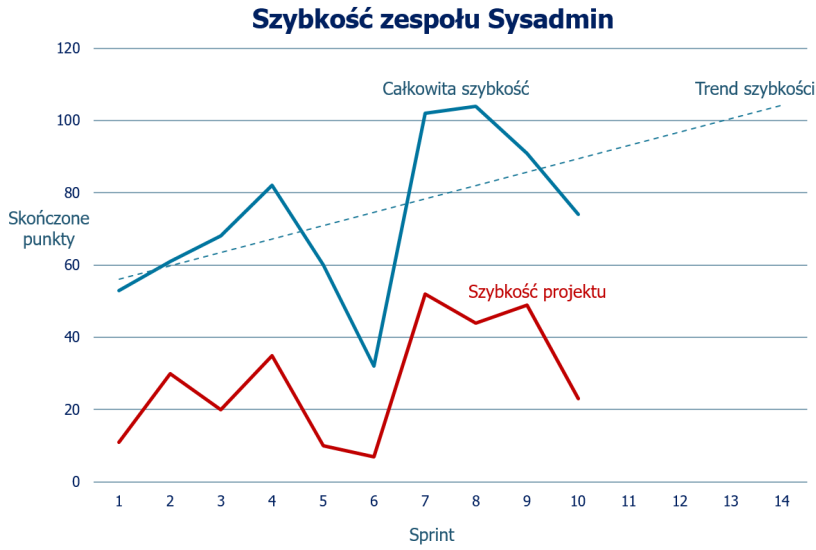
Rysunek 10. Przedtem: Menadżer pierwszej linii jest punktem kontaktowym dla zespołu. Wszystkie ważne zadania przechodzą przez niego. Mniejsze problemy, typowe problemy programistów, są przekazywane przez system śledzenia zgłoszeń. Występowało niewiele bezpośrednich interakcji pomiędzy poszczególnymi osobami.

Potem



Rysunek 11. Potem: Wprowadzenie zasadny „jednej osoby kontaktowej na zespół". Zespoły programistyczne rozmawiają bezpośrednio z wyznaczoną osobą kontaktową. Występuje wiele interakcji między poszczególnymi osobami. Członkowie zespołu wsparcia sami organizują swoją pracę za pomocą tablicy Kanbana. Menadżer skupia się na priorytetyzowaniu większych projektów i udzielaniu wsparcia w razie powstania trudnych problemów.

Jaki to wszystko miało wpływ na wydajność zespołu?



Rysunek 12. Całkowita szybkość i szybkość projektu mierzone za pomocą punktów historyjkowych skończonych w danym tygodniu. Wartość całkowita to suma wszystkich kolumn, szybkość projektu to część związana z projektami (większymi pracami, takimi jak na przykład rozbudowanie platformy sprzętowej). Dwa spadki to: 1) tydzień, w którym prawie wszyscy członkowie zespołu byli na wyjeździe, i 2) duże uruchomienie projektu programistycznego.

Szybkość zespołu wykazywała trend rosnący. W tym samym czasie zespół bardzo inwestował w dzielenie się wiedzą i pracę w parach.

Skoro już o tym mówimy, przyjrzyjmy się wydajności zespołu DBA;

Rysunek 13. Całkowita szybkość i małe zadania wsparcia. Spadek w środkowej części wykresu to okres świąteczny.

Trend jest rosnący, chociaż występuje duże odchylenie. Rozmiar odchylenia zmotywował zespół do monitorowania liczby niewielkich zadań wsparcia (zadań zbyt małych, aby trafiły na tablicę Kanbana). Jak widzisz, wykres pokazuje wyraźny związek między liczbą niewielkich zadań wsparcia a całkowitą szybkością.

Zespół zajmujący się wsparciem dla klientów zaczął przygodę z Kanbanem nieco później niż pozostałe dwa zespoły, więc nie mamy jeszcze żadnych wiarygodnych danych.

Dojrzewanie

Kiedy zaczynaliśmy, odnajdywanie problemów było łatwe. Jednak zlokalizowanie największej okazji do udoskonalenia było trudne. Tablica Kanbana dała nam zupełnie nowy poziom przejrzystości. Nie tylko łatwiej było zlokalizować problemy, ale podnoszone były ważne pytania dotyczące przepływu, odchyleń i kolejek. Zaczęliśmy używać kolejek jako narzędzia do rozwiązywania problemów. Cztery miesiące po stworzeniu tablicy menadżerowie polowali na odchylenie szkodzące zespołom.

Kiedy zespół ewoluował od zbioru indywidualistów do samoorganizujących się jednostek, menadżerowie spostrzegli, że stoją przed zupełnie nowymi wyzwaniami. Musieli więcej zajmować się problemami ludzi – skargami, definiowaniem wspólnych celów,

rozwiązywaniem konfliktów i negocjowaniem ustaleń. Nie była to bezbolesna zmiana – otwarcie przyznawali, że wymagało to umiejętności i energii. Podjęli jednak wyzwanie i ostatecznie stali się lepszymi liderami.

32

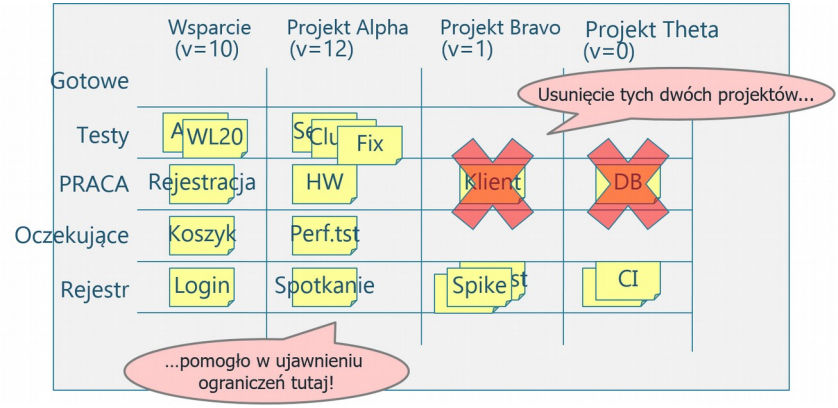
Wnioski

Kiedy zmniejsza się ilość pracy w toku, pojawiają się ograniczenia

Wszystkie zespoły zaczęły z dość wysokimi limitami WIP. Wtedy większość energii była poświęcana na próbę stworzenia przepływu i sprawienia, aby organizacja otrzymała wsparcie, którego potrzebowała.

Na początku menadżerowie chcieli, aby wiele projektów było wykonywanych jednocześnie, ale w ciągu kilku tygodni stało się jasne, że nie było wystarczająco dużo pojemności, aby pracować również nad projektami o niskim priorytecie. Wystarczył szybki rzut oka na tablicę, aby dostrzec, że zadania o niskim priorytecie nie były nawet rozpoczynane. Było to dla menadżerów sygnałem do zmniejszenia liczby projektów na zespół.

Z czasem, w miarę jak przepływ był coraz płynniejszy dla zadań o wysokim priorytecie, zabraliśmy się za dostosowywanie limitów WIP. Zrobiliśmy to, redukując liczbę jednocześnie prowadzonych projektów (kolumn) z trzech do dwóch, a potem do jednego. Kiedy to się stało, zaczęły się ujawniać ograniczenia spoza zespołu. Członkowie zespołu zaczęli raportować, że nie otrzymują na czas wymaganej pomocy, więc menadżerowie zajęli się rozwiązywaniem tego problemu.



Inne problemy, jakie się pojawiły, to na przykład błędne dane przychodzące z innych zespołów. Ciężko było utrzymać płynny i szybki przepływ, kiedy przychodzące zadania ciągle potrzebowały korygowania.

Te problemy nie były niewidzialne, zanim zaczęliśmy. Chodziło raczej o to, którym problemem powinniśmy zająć się najpierw – i o osiągnięcie porozumienia w tej kwestii. Dzięki tablicy Kanbana wszyscy mogli zobaczyć, jak dany problem wpływa na przepływ, co pozwoliło uzyskać motywację do rozprawienia się z problemem.

Tablica będzie się zmieniać, nie ustalaj nic na stałe

Wszystkie tablice Kanbana zmieniały się w czasie. Zazwyczaj konieczne było dwukrotne lub trzykrotne przeprojektowanie, zanim zespół uzyskiwał właściwy układ tablicy. Dlatego inwestowanie dużej ilości czasu w pierwszą wersję tablicy jest raczej błędem. Postaraj się, aby można było łatwo zmieniać układ tablicy. My wykorzystaliśmy paski czarnej taśmy, bo bez problemu można je było poprzekładać, a poza tym można je umieszczać na ścianach i tablicach. Inny sposób to narysowanie linii grubym markerem (upewnij się jednak, czy będzie je można zmazać! ☺).

Poniżej znajdziesz typowy przykład optymalizacji. Na początku priorytety zmieniały się często, aby więc nie musieć przekładać całej kolumny z karteczkami w tę i z powrotem, zespół nad każdą kolumną umieścił karteczkę z numerem priorytetu.

Rysunek 14. Wczesna tablica Kanban z karteczkami oznaczającymi priorytety.

Nie bój się eksperymentów i porażek

Dzięki tej przygodzie przekonałem się, że nie ma punktu końcowego. Przegrywamy w momencie, kiedy uznajemy, że taki punkt istnieje. Jest tylko niekończące się eksperymentowanie i nieustanna nauka. Nie popełnia błędów ten, kto się nie rozwija. My kilkakrotnie popełniliśmy błędy (złe projekty tablic, niewłaściwe szacunki, nadmierne spalanie itp.), ale za każdym razem uczyliśmy się czegoś nowego i ważnego. Gdybyśmy przestali próbować, jak moglibyśmy się uczyć?

Sukces metody Kanban zainspirował również zespoły zarządzające i zespoły Scruma do rozpoczęcia eksperymentów z tablicami Kanbana. Być może ta książka im w tym pomoże!

Czego nauczyła Cię ta książka?

Zacznij od retrospektyw!

Sporo opcji i kwestii do przemyślenia, prawda? Mamy nadzieję, że ta książka rozjaśniła ci trochę sytuację. Bo nam zawarta w niej wiedza bardzo pomogła :o)

Jeżeli chcesz wprowadzić zmiany i udoskonalenia do swojego procesu, pozwól, że podejmiemy jedną decyzję za ciebie: jeśli regularnie nie robisz retrospektyw, zacznij od nich! I upewnij się, że prowadzą do prawdziwych zmian. W razie potrzeby zaangażuj kogoś z zewnątrz, kto będzie w stanie je poprowadzić.

Kiedy już będziesz mieć zorganizowane efektywne retrospektywy, będzie to oznaczało, że rozpocząłeś swoją podróż w kierunku stworzenia procesu odpowiedniego w twoim przypadku — niezależnie od tego, czy będzie on bazował na Scrumie, XP, Kanbanie, ich kombinacji czy czymkolwiek innym.

Nigdy nie przestawaj eksperymentować!

Kanban czy Scrum nie są celem – celem jest nieustanna nauka. Jedną ze wspaniałych cech oprogramowania jest krótka pętla informacji zwrotnych, która jest kluczem do nauki. Korzystaj z niej! Kwestionuj wszystko, eksperymentuj, ponoś porażki, ucz się i znowu eksperymentuj. Nie próbuj zrobić wszystkiego dobrze od początku, bo to nie możliwe! Zacznij gdzieś i zmieniaj proces.

Jedyną *prawdziwą* porażką jest porażka w wyciąganiu wniosków z porażek.

Ale i z tego można wyciągnąć naukę.

Powodzenia i miłej podróży!

/Henrik i Mattias, Sztokholm, 24 czerwca 2009

H: To już wszystko, co mamy?

M: Chyba tak, poprzestańmy na tym.

H.: Może powinniśmy powiedzieć im, kim jesteśmy?

M.: Racja, jeżeli pokażemy się z dobrej strony, może złapiemy jakieś zlecenia jako konsultanci.

H.: Zróbmy to, a potem koniec.

M.: Tak, mamy inne zajęcia i czytelnicy również.

H.: Właściwie to właśnie zaczynam wakacje :o)

M: Eh, nie przechwalaj się.

O autorach

Henrik Kniberg i Mattias Skarin są konsultantami w firmie Crisp w Sztokholmie. Lubią pomagać firmom w odnoszeniu sukcesów zarówno technicznych, jak i personalnych i pomogli wielu zespołom we wdrożeniu zasad Lean i Agile.

Henrik Kniberg

Przez ostatnią dekadę Henrik był dyrektorem technicznym w trzech szwedzkich firmach z branży IT, a wielu innym pomógł usprawnić procesy. Jest certyfikowanym trenerem Scruma i na co dzień pracuje z pionierami Lean i Agile takimi jak Jeff Sutherland, Mary Poppendieck i David Anderson.

Poprzednia książka Henrika, Scrum and XP from the Trenches, zyskała prawie 150 000 czytelników i jest jedną z najpopularniejszych pozycji o tej tematyce. Henrik kilkakrotnie zdobywał tytuł najlepszego mówcy na międzynarodowych konferencjach.

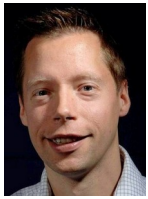
Henrik dorastał w Tokio, a teraz mieszka w Sztokholmie z żoną Sophią i trójką dzieci. W czasie wolnym jest aktywnym muzykiem – komponuje i gra na basie i klawiszach z lokalnymi zespołami.

henrik.kniberg@crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin



Mattias pracuje jako trener Lean i pomaga firmom korzystać z dobrodziejstw Lean i Agile. Uczy personel wszystkich szczebli, od programistów po zarząd. Pomógł firmie tworzącej gry skrócić czas tworzenia gry z 24 do 4 miesięcy, odbudował zaufanie do zespołów programistycznych i był jednym z pionierów Kanbana.

Jako przedsiębiorca założył i prowadził dwie firmy.

Mattias ma tytuł magistra w dziedzinie zarządzania jakością i przez 10 lat pracował jako programista.

Mieszka w Sztokholmie i lubi rock and rolla, taniec, wyścigi i narciarstwo.

mattias.skarin@crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin

Przekład na język polski

Tłumaczenie książki przygotowane zostało przez zespół Shore Labs - twórców serwisu Kanban Tool ®

Kanban Tool to internetowy serwis do organizacji pracy oparty o metodę Kanban, który w prosty sposób pozwala zwizualizować i na bieżąco śledzić postępy pracy, realizację zadań i przebieg projektów w każdej firmie. Od 2009 roku nasza aplikacja wspomaga pracę ponad 40 000 zespołów na całym świecie i dostępna jest już w 7 językach, w tym polskim.

Kanban Tool pozwala na:

- dostosowanie wirtualnej tablicy Kanban do dowolnego procesu i Twoich indywidualnych potrzeb,
- monitorowanie i raportowanie indywidualnego czasu pracy nad każdym zadaniem, co może stanowić podstawę fakturowania,
- współpracę w czasie rzeczywistym z całym zespołem i kontrahentami,
- włączenie kilkudziesięciu dodatkowych funkcji znacznie rozszerzających możliwości serwisu np. o planowanie terminów,
- przechowywanie danych w chmurze, dzięki czemu tablice Kanban dostępne są o każdej porze i w każdym miejscu.

Dzięki tym cechom jesteś w stanie włączyć klientów, pracowników i kontrahentów do współpracy nad zadaniami, niezależnie od ich lokalizacji i godzin pracy. W każdym momencie wiesz co dzieje się teraz, kto nad czym pracuje, co jest w planach, jakie są problemy i kto jest dostępny dla nowych zadań. Zyskujesz też automatyczne analizy, które pomogą zoptymalizować pracę w twoim zespole i firmie.

Sprawdź możliwości Kanban Tool za darmo, rejestrując się na http://kanbantool.com i zapraszając cały zespół do współpracy.

support@kanbantool.com

https://kanbantool.com