Spis treści

Uwagi do wydania polskiego

Książka ta została przetłumaczona z uwzględnieniem specyficznych reguł narzuconych tłumaczom. Jedną z nich był wymóg utrzymania oryginalnego brzmienia nazw własnych elementów Scruma, zgodnie z aktualnie obowiązującą wersją Przewodnika po Scrumie. Poniżej zamieszczamy zestawienie nazw własnych z ich polskimi odpowiednikami wykorzystywanymi we wcześniejszych wydawnictwach, w tym również we wcześniejszych wersjach Przewodnika.

Events	**Wydarzenia**
Sprint	Sprint
Sprint Planning	Planowanie Sprintu
Daily Scrum	Codzienny Scrum
Sprint Review	Przegląd Sprintu
Sprint Retrospective	Retrospektywa Sprintu
Roles	**Role**
Scrum Team	Zespół Scrumowy
Scrum Master	Scrum Master
Product Owner	Właściciel Produktu
Developers	Deweloperzy
Artifacts	**Artefakty**
Product Backlog	Backlog Produktu
Sprint Backlog	Backlog Sprintu
Increment	Przyrost

Aktualną wersję *Przewodnika po Scrumie* można znaleźć pod adresem:
https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Polish.pdf

O autorze

RICHARD HUNDHAUSEN jest prezesem Accentient – firmy, która pomaga organizacjom i zespołom zajmującym się tworzeniem oprogramowania w dostarczaniu lepszych produktów dzięki zrozumieniu i wykorzystaniu usług Azure DevOps oraz Scruma. Jest profesjonalnym trenerem Scruma (Professional Scrum Trainer) i współtwórcą platformy Nexus Scaled Scrum.

Jako programista, konsultant i szkoleniowiec z niemal 40-letnim doświadczeniem rozumie, że oprogramowanie jest tworzone i dostarczane przez ludzi, a nie przez procesy lub narzędzia. Można się z nim skontaktować pod adresem *richard@accentient.com*.

Przedmowa

Do roku 2001 branża programistyczna była w tarapatach – więcej projektów kończyło się niepowodzeniem niż sukcesem. Klienci zaczęli domagać się kar umownych i zlecać pracę za granicą. Niektórzy programiści zaczęli jednak odnosić coraz większe sukcesy stosując „lekkie" procesy w rozwijaniu oprogramowania. Niemal zawsze procesy te były oparte na dobrze określonym, iteracyjnym i przyrostowym działaniu.

W lutym 2001 programiści ci wydali manifest programowania zwinnego – Agile Manifesto. Manifest ten wzywał do zwinnego (agile) tworzenia oprogramowania w oparciu o cztery podstawowe wartości i dwanaście zasad. Dwie spośród tych zasad to: 1) zadowalać klientów wczesnym i ciągłym dostarczaniem działającego oprogramowania oraz 2) możliwie często dostarczać działające oprogramowanie (w okresie od kilku tygodni do kilku miesięcy).

Do roku 2009 używany był głównie proces Scrum Agile. Zapewniał proste ramy postępowania dla iteracyjnego, przyrostowego tworzenia oprogramowania. Uwzględniał też wartości i zasady Agile Manifesto. Dwóch autorów Scruma, Jeff Sutherland i ja, byliśmy też wśród autorów Agile Manifesto.

Spodziewałem się pewnych trudności, jakie napotkają organizacje (a nawet zespoły) wybierające Scrum. Wierzyłem jednak, że programiści rozkwitną w środowisku Scruma. Przyduszeni ograniczeniami kaskadowej metodologii wytwarzania oprogramowania, programiści odżyją, stosując zwinne praktyki, współpracę i narzędzia, na które nie było czasu w projektach opracowywanych wcześniej. Ku mojemu zaskoczeniu dotyczyło to może 20 procent wszystkich twórców oprogramowania.

W roku 2009 Martin Fowler określił większość zwinnego tworzenia oprogramowania jako „zwiotczałe":

> *Słyszałem ostatnio o sporym bałaganie w całkiem dużej liczbie projektów. Odbywa się to często następująco:*
>
> - *Chcą wykorzystać proces Agile i wybierają Scrum.*
> - *Przyjmują praktyki Scruma, a może nawet jego zasady.*
> - *Po pewnym czasie postęp działań znacznie spowalnia, ponieważ w bazie kodu jest bałagan.*
>
> *Nie przywiązywali wystarczającej uwagi do wewnętrznej jakości swojego oprogramowania. Popełniając ten błąd, można wkrótce odkryć, że produktywność jest ściągana w dół, ponieważ dużo trudniej jest dodawać nowe funkcje. Ze względu na paraliżujący dług techniczny zasady Scruma ledwo trzymają się na nogach (a jeśli ktoś naprawdę stosował Scrum, to wie, że to bardzo źle)."*
> *http://martinfowler.com/bliki/FlaccidScrum.html*

Opis „zwiotczałego” Scruma dokonany przez Martina zgadzał się z naszym doświadczeniem. Większość programistów była wykwalifikowana, ale nie dość wykwalifikowana w trzech wymiarach wymaganych do szybkiego budowania pełnych przyrostów użytecznej funkcjonalności. Wymiarami tymi są:

- **Ludzie** Umiejętność pracy w małym, wielofunkcyjnym, samozarządzającym się zespole.
- **Praktyki** Wiedza i umiejętność zastosowania nowoczesnych praktyk inżynieryjnych wymaganych przez krótki cykl rozwoju produktu.
- **Narzędzia** Narzędzia, które integrują i automatyzują te praktyki, tak aby kolejne przyrosty mogły być szybko integrowane bez nagromadzania się artefaktów, które trzeba obsługiwać ręcznie.

Zawiesiliśmy naszą działalność, podczas gdy pracowaliśmy przez cały rok 2009, tworząc to, co stało się znane jako program Professional Scrum Developer. Zaproponowaliśmy warsztaty w trzydniowym formacie. Na wejściu byli programiści, których wiedza i zdolności dawały „zwiotczałe” przyrosty. Wyjście stanowiły zespoły programistów, które opracowywały solidne przyrosty oprogramowania wymagane przez manifest Agile oraz nowoczesne, konkurencyjne organizacje.

Richard był z nami od początku. Jego książka *Profesjonalne wytwarzanie oprogramowania z zastosowaniem Scruma i usług Azure DevOps* stanowi dalszy ciąg jego udziału w tym ruchu, rozpoczętym przez nas w roku 2009.

Czytając książkę Richarda, można poznać trzy wymiary potrzebne do zwinnego opracowywania oprogramowania: ludzi, praktyki i narzędzia. Jak podczas tego kursu, Richard przeplata je ze sobą w coś, co łatwo przyswoić. Warto przeczytać książkę Richarda, jeśli ktoś jest członkiem Scrum Teamu. Wymienić wymagane praktyki. Zidentyfikować, które praktyki stanowią największe wyzwania dla zespołu. Uszeregować je według największego wpływu. Następnie kolejno je skorygować.

Wiele osób wydaje pieniądze na konferencje dotyczące Agile. Lepiej je oszczędzić, kupując tę książkę, omawiając ją z innymi i chodząc na spotkania oraz inne nieformalne „konferencje” dla zaawansowanych.

Richard i ja czekamy na wzrost waszych umiejętności. Nasza branża i nasze społeczności tego potrzebują. Oprogramowanie jest największym skalowalnym zasobem potrzebnym naszemu coraz bardziej złożonemu społeczeństwu. Skuteczna i wydajna praca zespołowa grup Agile jest podstawą rozwiązywania problemów, których rozwiązania również oczekuje nasze społeczeństwo.

Bierzmy się za Scrum!

Ken Schwaber
Współtwórca Scruma

Wprowadzenie

Scrum stanowi ramy postępowania do opracowywania i utrzymywania złożonych produktów, takich jak oprogramowanie. Scrum to tylko zestaw zasad zdefiniowanych w *Przewodniku po Scrumie* (*https://scrumguides.org*) i opisuje role, wydarzenia, artefakty oraz reguły wiążące je ze sobą. Gdy są poprawnie używane, te ramy postępowania umożliwiają zespołowi rozwiązywanie złożonych problemów, jednocześnie wydajnie i twórczo dostarczając produkty o najwyższej możliwej wartości. Scrum jest metodą Agile (zwinną). Właściwie jest najpopularniejszą metodą Agile będącą obecnie w użyciu.

Scrum stosuje iteracyjne i przyrostowe podejście do optymalizowania przewidywalności i kontrolowania ryzyka. Wynika to z empirycznego charakteru sterowania procesami w Scrumie. Poprzez właściwe użycie inspekcji, adaptacji i przejrzystości Scrum Team może próbować nowych sposobów wykonania jakiegoś zadania (eksperymenty) i mierzyć przydatność po krótkiej iteracji. Członkowie zespołu mogą następnie kolektywnie podjąć decyzję o przyjęciu, rozszerzeniu lub porzuceniu danej praktyki. Obejmuje to narzędzia wykorzystywane przez zespół oraz sposób ich użycia.

Połączenie Scruma z narzędziami dostępnymi w usługach Microsoft Azure DevOps stanowi silny związek. Celem tej książki jest ustanowienie podwalin do zrozumienia Scruma i sposobu obsługi Scruma w Azure DevOps. Zilustruję też, które praktyki zapewniają większą wartość przy realizacji bez użycia narzędzi. Ponadto wskażę narzędzia, które błędnie są reklamowane jako zwinne i skontrastuję je z bardziej preferowanymi praktykami.

W wytwarzaniu oprogramowania wszystko może się zmienić w mgnieniu oka. Wiedzą o tym zdrowe zespoły. Wiedzą też, że ciągłe sprawdzanie i adaptowanie sposobu wykonywania zadań jest sposobem na życie. Wydajne Scrum Teamy posuwają się nawet o krok dalej. Wiedzą, że w każdej przeszkodzie lub dysfunkcji jest okazja do nauki i doskonalenia. Przeczytanie tej książki jest świetnym pierwszym krokiem.

Kto powinien przeczytać tę książkę

Ta książka przyda się każdemu członkowi zespołu programistycznego, który wykorzystuje lub rozważa wykorzystanie Scruma. Skupiam się głównie na obowiązkach i zadaniach Developera (co w terminologii Scruma obejmuje projektantów, architektów, programistów, testerów, autorów technicznych, itd.). Product Ownerzy i Scrum Masterzy również będą czerpać wartość z tej książki, gdyż będą wykorzystywać wiele tych samych narzędzi Azure DevOps do planowania i zarządzania swoją pracą oraz do oceny postępów. Interesariusze, w tym klienci, użytkownicy, sponsorzy i menedżerowie również mogą uzyskać wartość z tej książki, zwłaszcza ucząc się, co powinni i czego nie powinni robić zgodnie z regułami Scruma oraz które narzędzia w usługach Azure DevOps wspierają te reguły.

Ta książka głównie skupia się na wykorzystaniu Scruma przy tworzeniu produktów programistycznych, gdyż jest to docelowa dziedzina Azure DevOps. Większość tej książki może jednak znaleźć zastosowanie również poza środowiskami programistycznymi i projektami informatycznymi. Ponieważ Scrum stanowi lekką platformę do opracowywania adaptacyjnych rozwiązań dla wszelkiego rodzaju złożonych problemów, wskazówki zawarte w tej książce mogą być stosowane przy opracowywaniu wszelkiego rodzaju produktów, takich jak usługi, produkty fizyczne lub coś bardziej abstrakcyjnego.

Ta książka może nie być odpowiednia, jeśli...

Ta książka jest przeznaczona dla zespołów wykorzystujących razem Scrum i usługi Azure DevOps przy opracowywaniu złożonych produktów, takich jak oprogramowanie. Nie będzie aż tak wartościowa dla zespołów niewykorzystujących Scruma albo zespołów opracowujących mniej skomplikowane produkty przy użyciu Scruma. Nie zapewni żadnej wartości zespołom wykorzystującym formalne, kaskadowe albo sekwencyjne projekty rozwoju oprogramowania, chyba że będzie sposobem na przekonanie takich zespołów do Scruma. Podobnie, jeśli zespół wykorzystuje Scrum, ale nie korzysta jeszcze z Azure DevOps, to duża część tej książki nie będzie zbyt ciekawa poza zdefiniowaniem i podkreśleniem charakteru Professional Scrum oraz wskazaniem, co takie zespoły mogą potencjalnie zyskać. Dotyczy to również zespołów wykorzystujących starsze wersje Team Foundation Server, które nie zawierają najnowszych, wartościowych narzędzi zespołowych do planowania i zarządzania pracą oraz umożliwiających współpracę w zespole.

Jeśli ktoś szuka „najlepszych praktyk”, to nie jest to właściwa książka ani właściwy autor. Staram się nie używać tego terminu, ponieważ oznacza on przyjęcie kilku

błędnych założeń: (1) że dana praktyka jest naprawdę „najlepsza” dla wszystkich zespołów pracujących nad wszystkimi produktami we wszystkich organizacjach oraz (2) że zespół może przestać poszukiwać nowych rozwiązań i eksperymentować, gdy już odnajdzie taką najlepszą praktykę. Wolę zamiast tego określenie „sprawdzona praktyka”. Niezależnie od nazwy, ta książka jest pełna wielu praktyk, które osoby i zespoły mogą wziąć pod uwagę w swoim dążeniu do doskonałości.

Organizacja tej książki

Ta książka jest podzielona na trzy części, z których każda koncentruje się na innym aspekcie związku Professional Scrum z Azure DevOps. Część I „Podstawy Scruma” zapewnia podwaliny wiedzy na temat struktury Scruma, Professional Scrum, Azure DevOps, a w szczególności usługi Azure Boards. Część II „Professional Scrum w praktyce” składa się z kilku rozdziałów opisujących szczegółowo praktyczne zastosowania przez Professional Scrum Team poszczególnych funkcji Azure DevOps do tworzenia i zarządzania Product Backlogiem, Sprintu Planningu, tworzenia Sprint Backlogu i efektywnej współpracy podczas Sprintu. Część III „Doskonalenie” zawiera rozdział definiujący i usprawniający działanie Scrum Teamu, identyfikując typowe wyzwania i dysfunkcje, które należy usuwać oraz wykorzystując techniki ciągłego doskonalenia metod Scruma. Jest tam też rozdział dotyczący poprawy skalowalności poprzez Scaled Professional Scrum przy wykorzystaniu platformy Nexus Scaled Scrum. Czytając kolejno wszystkie części, można będzie zobaczyć, jak w efektywny sposób korzystać jednocześnie z usług Azure DevOps i Scruma i jak Scrum Team może przekształcić się w Professional Scrum Team, a później w wysokowydajny Professional Scrum Team.

W każdym rozdziale proponuję i polecam wiele praktyk i wzorców działania. Wykorzystuję terminy, takie jak *Professional* Scrum Team i *wysokowydajny* Scrum Team, aby odróżniać je od typowych Scrum Teamów, które praktykują Scrum mechanicznie bez zwracania uwagi na inspekcję, adaptację i doskonalenie. Czasami ktoś może odrzucać moje wskazówki jako „myślenie magiczne” i zakładać, że nie żyję w realnym świecie. Można by sądzić, że proponowane przeze mnie pomysły nie zadziałają w danym zespole, przy danej grupie ludzi lub w danej organizacji. Choć prawdą jest, że nie znam specyfiki danej organizacji, to jestem przekonany, że można dążyć do doskonałości, niezależnie od liczby napotykanych przeszkód. Byłem tego świadkiem, tak samo jak wielu moich kolegów prowadzących szkolenia jako Professional Scrum Trainer. Trzeba mieć na uwadze, że moje opisy tych wysokowydajnych zachowań powinny być traktowane jako wizja albo „cel doskonalenia”, do którego zespół powinien dążyć. Będzie to trudne. Będzie wymagać czasu. Będzie wymagać pomocy. Ostatecznie tacy właśnie ludzie będą prowadzić swoje zespoły na drodze do doskonałości.

Od czego najlepiej zacząć czytanie tej książki

Poszczególne części tej książki obejmują różne zakresy zagadnień. W zależności od potrzeb i aktualnej wiedzy na temat Scruma, usług Azure DevOps i powiązanych praktyk, można skupić się na określonych obszarach tej książki. Korzystając z poniższej tabeli, można określić, jak najlepiej korzystać z tej książki.

Jeśli ktoś	To powinien:
Nigdy nie słyszał o Scrumie lub jest jego nowym adeptem	Przeczytać *Przewodnik po Scrumie*, a następnie przeczytać rozdział 1
Nigdy nie słyszał o Professional Scrum lub jest jego nowym adeptem	Przeczytać rozdział 1
Jest nowym użytkownikiem zestawu narzędzi Azure DevOps	Przeczytać rozdział 2
Jest nowym użytkownikiem usługi Azure Boards lub chce wiedzieć więcej na temat tworzenia niestandardowego procesu Professional Scrum	Przeczytać rozdział 3
Jest zaznajomiony ze Scrumem oraz usługami Azure DevOps i jedynie chce dowiedzieć się, jak skonfigurować Azure DevOps dla Scrum Teamu	Przeczytać rozdział 4
Jest zaznajomiony ze Scrumem oraz usługami Azure DevOps i jedynie chce dowiedzieć się, jak planować Sprint i tworzyć Sprint Backlog	Przeczytać rozdział 6
Nie zna pojęcia programowania sterowanego testami akceptacyjnymi i chce dowiedzieć się, jak planować i śledzić Sprint przy użyciu Azure Test Plans	Przeczytać rozdział 7
Nie zna pojęcia przepływu albo sposobów wykorzystania tablicy Kanban do wizualizacji pracy i zarządzania jej przepływem	Przeczytać rozdział 9
Staje przed typowymi wyzwaniami Scruma i jest zainteresowany radzeniem sobie z dysfunkcyjnymi działaniami	Przeczytać rozdział 10
Staje w obliczu sytuacji skalowania, gdzie kilka Scrum Teamów współpracuje nad budowaniem wspólnego produktu	Przeczytać rozdział 11

Konwencje i zasady w tej książce

Ta książka przedstawia informacje, korzystając z konwencji zaprojektowanych tak, aby informacje te były czytelne i łatwe w odbiorze.

- Dla orientacji udostępniane są zrzuty ekranów odpowiednich funkcji Azure DevOps.
- Elementy w ramkach zatytułowanych „Uwaga” lub „Wskazówka” zapewniają dodatkowe informacje i porady związane z danym zagadnieniem.
- Niektóre uwagi i wskazówki są praktycznymi poradami zapewnianymi przez zaprzyjaźnione osoby z tytułami Professional Scrum Developer i Professional Scrum Trainer, które pomagały w pracy nad tą książką.

Dodatkowo zawarłem dwa dodatkowe rodzaje ramek zatytułowanych „Brzydkie zapachy” i „Studium przypadku Fabrikam Fiber”.

Brzydki zapach W tej książce zwracam uwagę na konkretne sytuacje i pułapki, których Scrum Team i jego członkowie powinni unikać. Nazywam je *brzydkimi zapachami*. Te brzydkie zapachy zazwyczaj, ale nie zawsze, oznaczają jakieś zaburzenia lub inne niezdrowe zachowania. Dla zespołów będących nowicjuszami Scruma te brzydkie zapachy mogą być trudne do zidentyfikowania. Gdy jednak zostaną wskazane, powinny zostać wyeliminowane i wykorzystane jako okazja do nauki. W miarę rozwoju zespołu powinien coraz lepiej radzić sobie z rozpoznawaniem i usuwaniem takich zaburzeń. Zespoły Professional Scrum Team potrafią identyfikować potencjalnie szkodliwe elementy lub dysfunkcje, oceniać ryzyka z nimi związane, a nawet eliminować konkretne szkodliwe zachowania.

Studium przypadku Fabrikam Fiber

Na kolejnych stronach tej książki będziemy posługiwać się studium przypadku firmy Fabrikam Fiber. Fabrikam Fiber jest fikcyjnym dostawcą szerokopasmowych usług telekomunikacyjnych. Fabrikam Fiber jest wielką korporacją, która świadczy usługi w wielu stanach USA. Firma ta wykorzystuje wewnętrzną aplikację webową dla swoich przedstawicieli obsługi klienta do tworzenia i zarządzania zgłoszeniami problemów z obsługą klientów. Tworzący ją zespół korzysta ze Scruma od jakiegoś czasu i przeszedł od niedawna na usługi Azure DevOps. Moje opinie na temat zdrowych i niezdrowych zachowań są widoczne poprzez wybory dokonywane przez Scrum Team w firmie Fabrikam Fiber.

Wymagania systemowe

Mimo że ta książka nie zawiera żadnych ćwiczeń praktycznych, zachęcam do zarejestrowania się w usługach Azure DevOps, aby móc z nimi eksperymentować i ćwiczyć podczas czytania tej książki. Utworzenie swojej organizacji zajmuje tylko kilka minut, a pierwszych pięciu użytkowników w planie podstawowym może korzystać z tych usług za darmo – co powinno wystarczyć, aby grupa kolegów mogła korzystać ze wszystkich funkcji wspominanych w tej książce. Usługi Azure DevOps stanowią ofertę SaaS (Software as a Service – oprogramowanie jako usługa) opartą na chmurze i oferującą dostarczanie nowych funkcji co kilka tygodni, co oznacza, że zrzuty ekranów w tej książce mogą nie odpowiadać dokładnie temu, co będzie widoczne w przeglądarce czytelnika za jakiś czas.

Ponadto warto pobrać oprogramowanie Visual Studio Community Edition lub Visual Studio Code, aby zbadać, jak łączy się z Azure DevOps i jak może być wykorzystywane do współpracy przy użyciu usług Azure Boards i Azure Repos. Oba te produkty są darmowe.

Errata, aktualizacje i wsparcie

Dokonaliśmy wszelkich starań, aby zapewnić dokładność informacji w tej książce i jej materiałach towarzyszących. Aktualizacje tej książki mogą być dostępne w formie erraty i listy poprawek pod adresem:

MicrosoftPressStore.com/ProfScrumDevelopment/errata

W przypadku odkrycia błędu, który nie jest już wymieniony na tej liście, prosimy o przesłanie nam informacji, korzystając z formularza na tej samej stronie.

Dodatkowe wsparcie i informacje związane z tą książką można znaleźć pod adresem: *www.MicrosoftPressStore.com/Support.*

Należy zwrócić uwagę, że wsparcie techniczne dla produktów programowych i sprzętowych firmy Microsoft nie jest oferowane pod powyższymi adresami. Pomoc dotyczącą oprogramowania lub sprzętu firmy Microsoft można uzyskać pod adresem *http://support.microsoft.com.*

Bądźmy w kontakcie

Chętnie porozmawiamy! Nasza strona w serwisie Twitter: *http://twitter.com/MicrosoftPress*

Podziękowania

Jest kilka osób, które pomogły mi w napisaniu tej książki. Podziękowania dla: Loretty Yates za kolejną szansę napisania książki dla wydawnictwa Microsoft Press; Charvi Arora, Tracey Croom, Elizabeth Welch, Songlin Qiu, Vaishnavi Venkatesan i Donny Mulder za cierpliwe przeglądanie moich materiałów i pomoc w poprawieniu stylu; Donis Marshall za inspirację do napisania kolejnej książki oraz bezpośrednie (i cenne) uwagi; Dana Hellema za mnóstwo odpowiedzi na pytania dotyczące Azure Boards i przejrzenie rozdziałów tej książki; Phila Japikse, Simona Reindla, Briana Randella, Ognjen Bajić, Ana Roje Ivančić, Martina Kulova, Cory'ego Isaksona, Davida Corbina, Charlesa Revella, Daniela Vacanti i Christiana Hassa za świetne pomysły i pomoc w dopracowaniu mojego przesłania; Kena Schwabera i Jeffa Sutherlanda za zaktualizowanie *Przewodnika po Scrumie*, gdy już niemal kończyłem pisanie tej książki.

CZĘŚĆ I

Podstawy Scruma

Rozdziały w tej części kładą podwaliny pod zrozumienie trzech obszarów, które musi znać każdy praktyk Professional Scrum korzystający z narzędzi DevOps firmy Microsoft:

- Scrum, a dokładniej Professional Scrum
- Microsoft Azure DevOps (ogólnie)
- Microsoft Azure Boards (w szczególności)

Zacznę od przyjrzenia się Scrumowi i zasadom Scruma. Skupię się na tym, jak i kiedy Developer wchodzi w interakcję z Product Ownerem i Scrum Masterem, uczestniczy w różnych wydarzeniach w Scrumie i współdziała z różnymi artefaktami Scruma. Ważne jest, aby wszyscy Developerzy rozumieli reguły Scruma i jakie są oczekiwania wobec nich i ich zespołu, a także kiedy i jak powinni wchodzić w interakcje z Product Ownerem, Scrum Masterem, interesariuszami i różnymi artefaktami.

Pozostałe rozdziały w tej części mają bardziej techniczny charakter i obejmują narzędzia dostępne w usługach Azure DevOps, a w szczególności usługę Azure Boards. Skupiam się na dostępnych w chmurze usługach Azure DevOps, a nie na lokalnym serwerze Azure DevOps. Choć istnieje wiele narzędzi DevOps dostępnych dla Scrum Teamu, staram się wymieniać i omawiać tylko te, które są istotne dla

praktykujących Professional Scrum. Zwracam również uwagę na to, których błyszczących narzędzi lepiej nie wyciągać, pozwalając zespołowi raczej na ćwiczenie bardziej cenionych praktyk związanych ze współpracą. Poza wszystkim cenimy jednostki i interakcje bardziej niż procesy i narzędzia.

Uwaga W Przewodniku po Scrumie z 2020 roku rola Development Teamu została zastąpiona rolą Developera. Celem było wyeliminowanie pojęcia oddzielnego zespołu wewnątrz zespołu, co prowadziło do podziałów „my i oni" pomiędzy Product Ownerem a Development Teamem. Obecnie jest tylko jeden zespół – Scrum Team – i koncentruje się na tym samym celu przy trzech różnych zakresach odpowiedzialności, które stanowią: Product Owner, Scrum Master i Developerzy. Pamiętajmy, że Scrum uznaje testerów, programistów, projektantów, architektów, analityków, specjalistów od baz danych, autorów technicznych, po prostu za... Developerów.

ROZDZIAŁ 1

Professional Scrum

Scrum to uproszczone ramy postępowania, które pomagają poszczególnym osobom, zespołom i organizacjom wytwarzać wartość poprzez adaptacyjne rozwiązywanie złożonych problemów. Oprogramowanie jest złożonym problemem. Dlatego Scrum jest idealny do zarządzania rozwojem oprogramowania w celu znajdowania rozwiązań w sposób adaptacyjny. Opracowywanie oprogramowania nie generuje tego samego wyjścia za każdym razem przy określonym wejściu. Scrum uznaje ten fakt, a ze względu na jego empiryczny charakter, promuje wykorzystanie eksperymentów w celu przeprowadzania inspekcji i adaptacji.

Scrum nie jest metodologią ani procesem. Stanowi tylko ramy postępowania. Innymi słowy, jeśli weźmiemy Scrum i dodamy swoje własne praktyki uzupełniające, takie jak programowanie sterowane testami adaptacyjnymi, uzyskamy *swój* proces. Zespoły mogą wykorzystywać empiryczne atrybuty Scruma, aby regularnie sprawdzać skuteczność swoich praktyk i dokonywać odpowiednich zmian. W tej książce przedstawię wiele praktyk uzupełniających do wzięcia pod uwagę.

Uwaga Fundamentem Scruma jest empiryczna teoria sterowania procesem i koncepcja „lean" (szczupłego zarządzania). Empiryzm stwierdza, że wiedza pochodzi z doświadczenia i podejmowania decyzji w oparciu o to, co jest znane. Koncepcja „lean" ogranicza straty i koncentruje się na tym, co najważniejsze.

Nawet dzisiaj, po ponad 60 latach ewolucji tworzenia oprogramowania, istnieje poważne ryzyko, że projekt programistyczny średniej lub dużej wielkości się nie powiedzie. Na szczęście nasza branża w końcu zauważyła ten problem, lepiej go rozumie i zaczyna na niego reagować. Niektóre organizacje zwiększyły swoje szanse na sukces. Istnieją dowody, że praktyki zwinne, takie jak Scrum, prowadzą do tych sukcesów.

Wskazówka Korzystając z analogii programistycznej, możemy traktować Agile jako *interfejs*. Manifest Agile definiuje cztery abstrakcyjne wartości i 12 abstrakcyjnych zasad (*http://agilemanifesto.org*). Chociaż istnieje wiele sposobów implementacji tych wartości i zasad, Agile Manifesto ich nie opisuje. Scrum natomiast to robi. Możemy traktować Scrum jako *konkretną klasę*, która *implementuje* wartości i zasady Agile poprzez swoje role, wydarzenia, artefakty i reguły.

Zespoły Agile wiedzą, że muszą ciągle dokonywać inspekcji i adaptacji – nie tylko swojego produktu, ale też swojego procesu i praktyk. Dobre poznanie (np. z tej książki) Scruma, DevOps i narzędzi firmy Microsoft stanowi dobry początek. Doświadczenie w korzystaniu z nich razem w praktyce jest jeszcze lepsze. Ideałem jest umiejętność rozpoznawania i wykorzystania okazji do poprawy w ich stosowaniu! Oznacza to bycie profesjonalistą. To powinno być naszym celem. Nie wystarczy zadowalać się projektem, który jedynie nie zawiedzie. Warto dążyć do jak najlepszego ukończenia projektu, z większą wartością i płynącą z niego nauką, niż mogłoby się to wydawać możliwe.

Przewodnik po Scrumie

Scrum istnieje od wczesnych lat 90-tych poprzedniego wieku. Na początku definicja Scruma i związanych z nim praktyk pochodziła z książek, prezentacji i od specjalistów, którzy starali się go jak najlepiej objaśniać. Niestety te informacje nie zawsze były dokładne i prawie nigdy nie były spójne. Nawet dwaj twórcy Scruma – Ken Schwaber i Jeff Sutherland – bywali czasami niespójni. Obecny Scrum nie przypomina tego sprzed 25 lat.

W roku 2009 organizacja Scrum.org skodyfikowała Scrum, tworząc i publikując *Przewodnik po Scrumie*. Ten darmowy przewodnik stanowi oficjalne reguły Scruma i jest opracowywany przez twórców Scruma: Schwabera i Sutherlanda. Jest to bardzo zwięzły dokument. Wersja PDF z roku 2020 zajmuje tylko 14 stron. Jest on dostępny w 30 językach do pobrania pod adresem *https://scrumguides.org*.

Przewodnik po Scrumie jest świetnym źródłem, z którego można korzystać podczas czytania tej książki. Czytając ten przewodnik, można zobaczyć, że Scrum jest prosty i dość łatwy do zrozumienia. Niestety jest niezwykle trudny do pełnego opanowania. *Przewodnik po Scrumie* będzie w przyszłości aktualizowany i może nieco zmienić wskazówki dostępne w tym rozdziale i w pozostałej części książki.

Wskazówka Można potraktować Scrum jak grę w szachy. Jedno i drugie ma swoje reguły. Na przykład Scrum nie pozwala na istnienie dwóch Product Ownerów, tak jak szachy nie pozwalają graczowi mieć dwóch królów. Podczas gry w szachy należy postępować zgodnie z regułami gry. Jeśli nie będzie się tego robić, nie będzie to graniem w szachy. Tak samo jest ze Scrumem. Można by też powiedzieć, że to nie Scrum ani szachy wygrywają lub przegrywają. To gracze wygrywają lub przegrywają. Ci, którzy grają zgodnie z zasadami, będą grać coraz lepiej, choć opanowanie gry może zająć sporo czasu.

Platforma Scrum składa się ze Scrum Teamu (zespołu) i powiązanych z nim ról, wydarzeń, artefaktów i reguł. Każdy z tych elementów służy określonemu celowi, jak zobaczymy w tym rozdziale. Reguły Scruma zdefiniowane w *Przewodniku po Scrumie* wiążą ze sobą te role, wydarzenia i artefakty. Przestrzeganie tych reguł ma zasadnicze

znaczenie dla powodzenia zdolności użycia Scruma przez zespół, a co ważniejsze, dla udanego opracowania i dostarczenia produktu o wysokiej wartości i jakości. Zmiana podstawowych zasad lub koncepcji Scruma, pominięcie pewnych elementów lub nieprzestrzeganie reguł zakrywa problemy i ogranicza korzyści zapewniane przez Scrum, potencjalnie czyniąc go wręcz bezużytecznym.

Scrum jest darmowy i jest oferowany w formie *Przewodnika po Scrumie*. Role, wydarzenia, artefakty i reguły Scruma są niezmienne, a choć możliwa jest implementacja tylko części Scruma, wynikiem nie będzie Scrum. Scrum istnieje tylko w całości i działa dobrze jako pojemnik dla innych praktyk, technik i metodologii. Często opisuję Scrum jako „ramy postępowania, w ramach których zespół może eksperymentować z różnymi praktykami uzupełniającymi".

Filary Scruma

Fundamentem Scruma jest empiryzm i koncepcja „lean". Empiryzm stwierdza, że wiedza pochodzi z doświadczenia i podejmowania decyzji w oparciu o to, co jest obserwowane i znane. Koncepcja „lean" ogranicza straty i koncentruje się na tym, co najważniejsze. Scrum stosuje iteracyjne i przyrostowe podejście do optymalizowania przewidywalności i kontrolowania ryzyka. Scrum angażuje grupy ludzi, którzy wspólnie mają wszystkie umiejętności i wiedzę do wykonania danej pracy i dzielą się nimi lub nabywają takie umiejętności w razie potrzeby.

Scrum łączy cztery formalne wydarzenia umożliwiające inspekcję i adaptację w ramach obejmującego je wydarzenia, jakim jest Sprint. Te wydarzenia działają, ponieważ implementują empiryczne filary Scruma, czyli inspekcję, adaptację i przejrzystość. Wydarzenia omówię później w tym rozdziale, ale chciałbym poświęcić chwilę na zdefiniowanie tych trzech filarów Scruma:

- **Inspekcja** Artefakty Scruma oraz postępy w dążeniu do uzgodnionych celów muszą być poddawane częstej i rzetelnej inspekcji, aby możliwe było wykrycie potencjalnie niepożądanych odstępstw lub problemów. Aby ułatwić inspekcję, Scrum zapewnia stały rytm w postaci swoich pięciu wydarzeń. Inspekcja umożliwia adaptację. Inspekcja bez adaptacji jest uznawana za bezcelową. Celem wydarzeń w Scrumie jest wywoływanie zmian i poprawy.
- **Adaptacja** Jeśli jakikolwiek aspekt procesu wykracza poza dopuszczalne limity lub jeśli uzyskany produkt jest niemożliwy do zaakceptowania, stosowany proces lub wytwarzane materiały należy odpowiednio skorygować. Zmian tych należy dokonać jak najszybciej, aby zminimalizować dalsze odstępstwa. Adaptacja staje się trudniejsza, kiedy osoby w nią zaangażowane nie mają odpowiednich uprawnień lub zdolności samozarządzania. Oczekuje się, że Scrum Team wprowadzi modyfikacje natychmiast po uzyskaniu jakiejkolwiek nowej wiedzy w wyniku inspekcji.

- **Przejrzystość** Kształtujący się proces oraz praca muszą być widoczne dla osób ją wykonujących, jak również dla osób, na rzecz których praca ta jest wykonywana. W Scrumie ważne decyzje podejmowane są na podstawie zaobserwowanego stanu jego trzech formalnych artefaktów. Niedostateczna przejrzystość artefaktów może prowadzić do decyzji, których wynikiem jest zmniejszona wartość i zwiększone ryzyko. Przejrzystość umożliwia inspekcję. Inspekcja i adaptacja bez przejrzystości prowadzą do błędów i strat.

Scrum w działaniu

Czytając *Przewodnik po Scrumie*, można zrozumieć składniki i powiązane z nimi reguły, ale nie koniecznie łatwo dostrzec, jak współdziałają ze sobą. Do tego wymagane jest faktyczne doświadczenie w korzystaniu ze Scruma przy opracowywaniu produktu w zespole. Jako substytut tego doświadczenia rysunek 1-1 został stworzony przez Scrum.org, aby pomóc w zilustrowaniu ram Scruma w działaniu.

W Scrumie produkt jest nośnikiem, który zapewnia wartość. Ten produkt może być usługą, produktem fizycznym lub czymś bardziej abstrakcyjnym. Produkt ma wyraźne granice, znanych interesariuszy oraz dobrze zdefiniowanych użytkowników lub klientów. Oprogramowanie dobrze pasuje do tej definicji, choć Scrum może być używany nie tylko do wytwarzania oprogramowania. Ponieważ jest to też książka dotycząca usług Azure DevOps, będę opisywał Scrum i Professional Scrum w kontekście *oprogramowania* jako produktu.

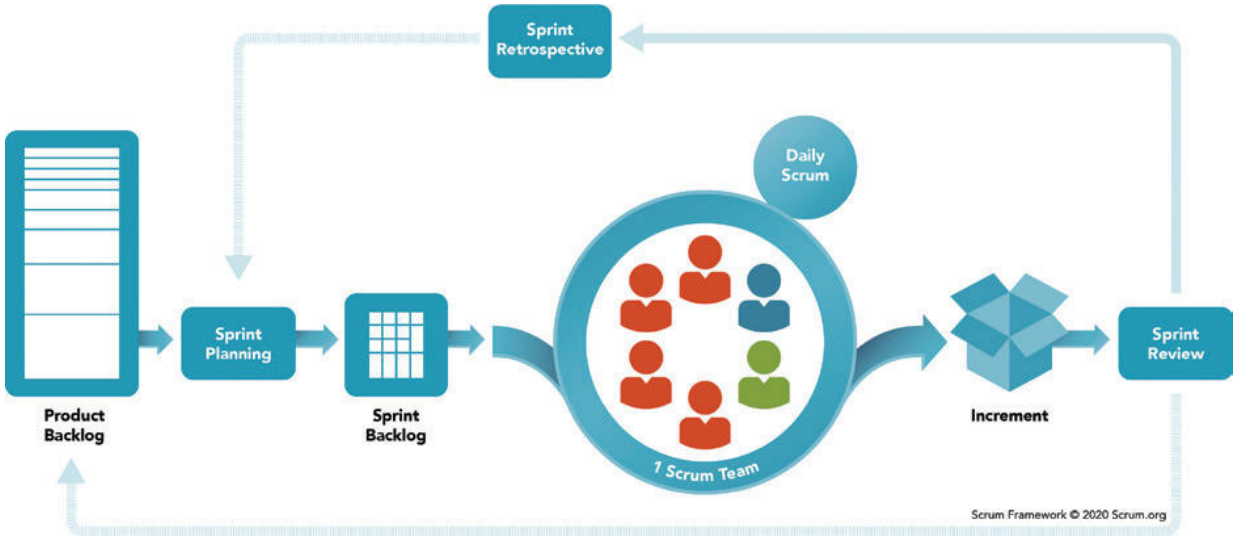
Cel Produktu opisuje przyszły stan produktu, który może służyć jako cel planowania działań przez Scrum Team. Cel Produktu stanowi długoterminowe zamierzenie Scrum Teamu. Zespół musi zrealizować jeden cel (lub z niego zrezygnować), zanim przystąpi do realizacji kolejnego. Przykładem Celu Produktu w przypadku oprogramowania mogłoby być „uzyskać 100 tys. pobrań ze sklepu z aplikacjami".

Product Backlog to uporządkowana lista wszystkiego, co jest konieczne do osiągnięcia Celów Produktu. Jest to jedyne źródło zmian dla produktu, które stale się zmienia, co oznacza, że będzie ciągle ewoluować ze względu na zmiany warunków biznesowych, dziedziny i technologii. Każdy element na tej liście jest określany skrótem PBI (Product Backlog Item). W przypadku oprogramowania Product Backlog obejmuje funkcje do zaimplementowania, błędy do poprawienia i eksperymenty do przeprowadzenia. Product Owner odpowiada za zarządzanie oczekiwaniami, zagrożeniami i wynikami produktu, a także za zapewnianie, aby Product Backlog był uporządkowany (ustawienie priorytetów), przejrzysty (dostępny) i zrozumiały.

Developerzy współpracują z Product Ownerem i innymi osobami w razie potrzeby podczas Sprint Planningu i udoskonalania Product Backlogu, aby rozumieć, szacować i przewidywać elementy PBI. Product Owner uporządkowuje Product Backlog zgodnie z czynnikami, takimi jak zwrot z inwestycji (ROI) dla tych elementów PBI, ryzyko, priorytety biznesowe, zależności i okazja do nauki.

Sprint jest okresem czasu o stałej długości, który obejmuje inne wydarzenia Scruma. Sprint powinien trwać miesiąc lub krócej, aby zmniejszać ryzyko i prowadzić do spójności. Nowy Sprint zaczyna się natychmiast po zakończeniu poprzedniego Sprintu.

Pierwszym wydarzeniem w Sprincie jest Sprint Plannning (planowanie Sprintu). W tym wydarzeniu Scrum Team współpracuje nad prognozowaniem i planowaniem pracy w danym Sprincie. Będą brane pod uwagę elementy PBI z górnej części Product Backlogu, które pozwolą jak najlepiej osiągnąć Cel Produktu. Scrum Team współpracuje nad prognozowaniem tych elementów, które wydają się możliwe do ukończenia przed końcem Sprintu. Formułowany jest Cel Sprintu i powstaje Sprint Backlog. Sprint Backlog zawiera prognozowane elementy oraz plan ich dostarczenia. Sprint Backlog na bieżąco pokazuje pracę pozostałą do wykonania podczas Sprintu.



Rysunek 1-1 Ramy Scruma

Znaczna część Sprintu będzie poświęcona na pracę nad osiągnięciem Celu Sprintu poprzez opracowywanie elementów ze Sprint Backlogu. Reguły Scruma nie określają dokładnie, co dzieje się codziennie podczas opracowywania produktu. Developerzy muszą spotykać się regularnie podczas wydarzenia Daily Scrum, aby synchronizować plan działań na następne 24 godziny.

Jeśli to konieczne, Developerzy powinni też spotykać się z Product Ownerem, aby udoskonalać Product Backlog. Podczas tego udoskonalania elementy w Product Backlogu otrzymują dodatkowe szczegóły i szacunki. To sprawia, że Product Backlog jest aktualny, tak aby Product Owner mógł prowadzić rzeczowe rozmowy z interesariuszami, planować wydania produktu i podejmować lepsze decyzje dotyczące przyszłych działań.

Podczas Sprintu zespół Scrum Team kończy prace nad elementami w Sprint Backlogu zgodnie z Definicją Ukończenia (Definition of Done). Ta definicja wymienia praktyki i standardy, które muszą być spełnione dla każdego elementu, zanim będzie mógł on zostać uznany za ukończony. Jeśli Definicja Ukończenia jeszcze nie istnieje,

to jest tworzona przez Scrum Team. Istotne jest, aby wszyscy rozumieli tę definicję, ponieważ będzie używana w celu ułatwienia kontroli postępów i jakości. Praca, która nie spełnia Definicji Ukończenia… nie jest ukończona i nie może zostać wydana. Nie powinna również podlegać inspekcji podczas wydarzenia Sprint Review (przeglądu Sprintu).

Najlepiej, aby Developerzy współpracowali z Product Ownerem podczas całego Sprintu, żeby zapewnić powstanie świetnego produktu. Jeśli Developerzy wcześnie ukończą pracę nad swoimi prognozami, powinni współpracować z Product Ownerem nad znalezieniem dodatkowych elementów PBI do realizacji. Z kolei przy pierwszych sygnałach, gdy Developerzy podejrzewają, że nie będą w stanie ukończyć prognozowanej pracy, powinni współpracować z Product Ownerem, aby zidentyfikować i omówić kompromisowe rozwiązania i zmodyfikować Sprint Backlog w taki sposób, który nie wpłynie negatywnie na jakość ani nie zmieni Celu Sprintu.

Increment (przyrost) jest użytecznym i dostarczającym wartość fragmentem pracy podlegającej inspekcji, który spełnia Definicję Ukończenia. Increment składa się z jednego lub kilku ukończonych elementów PBI, które zostały prognozowane do wykonania w czasie danego Sprintu. Increment podlega inspekcji podczas Sprint Review. Product Owner może zaprosić różnych interesariuszy na Sprint Review, aby poznać ich zdanie. Te informacje są gromadzone i mogą stać się nowymi elementami w Product Backlogu. Istniejące elementy PBI mogą też zostać zaktualizowane lub usunięte. Informacje zwrotne od interesariuszy mogą wpłynąć na decyzję Product Ownera dotyczącą wydania produktu, kontynuowania jego rozwoju lub zatrzymania w ogóle prac nad produktem. Te ostatnie decyzje powinny być oparte na przesłankach biznesowych, a nie jakościowych. Niezależnie od tego, kiedy Increment zostanie wydany, Scrum Team powinien zawsze opracowywać go tak, jakby miał zostać wydany. Może to obejmować wielokrotne wydanie podczas Sprintu, co jest wspierane przez Scrum.

Ostatnim wydarzeniem w Sprincie jest Sprint Retrospective (retrospektywa Sprintu). Jest to okazja na to, aby Scrum Team dokonał inspekcji siebie i swojej pracy w celu poprawienia swoich praktyk i procesu. Jeśli określone zostaną eksperymenty prowadzące do usprawnienia, zespół powinien utworzyć plan działania dla następnego Sprintu. Aby zapewnić stałe doskonalenie, Scrum Team powinien określić co najmniej jedno priorytetowe ulepszenie procesu i wdrożyć je podczas następnego Sprintu. Nic nie wykracza poza zakres Sprint Retrospective – można omawiać relacje, ludzi, proces, praktyki i narzędzia. Scrum Team może też podjąć decyzję o dostosowaniu swojej Definicji Ukończenia w celu zwiększenia jakości. Po zakończeniu Sprint Retrospective rozpoczyna się nowy Sprint i cały cykl się powtarza.

Role w Scrumie

Grupa osób, która odpowiada za zbudowanie produktu i spełnienie jego celów, jest nazywana Scrum Teamem (Zespołem Scrumowym). Scrum Team składa się z następujących ról:

- Product Owner (Właściciel Produktu)
- Developerzy
- Scrum Master (Mistrz Scruma)

Gdyby potraktować role jako świadczenie usług, to Developerzy służą Product Ownerowi, natomiast Scrum Master służy wszystkim. Dlatego Developerzy mogą wywierać silny wpływ na wybór Scrum Mastera. Product Owner może wywierać silny wpływ na wybór Developerów do zespołu (niezależnie od zasad i procedur kadrowych). Ze względu na ten rozdział obowiązków, role te powinny być odgrywane przez różne osoby. Zmniejsza to ryzyko wystąpienia konfliktu interesów. W mniejszych zespołach może być konieczne łączenie ról.

Developerzy

Developerzy są profesjonalistami w Scrum Teamie, którzy są w stanie zaprojektować, zbudować, przetestować i dostarczyć ukończony produkt. W Scrum Teamie jest od trzech do dziewięciu Developerów – jest to wystarczająco mała liczba, aby mogli się łatwo komunikować i żwawo działać, a przy tym wystarczająco duża, aby zapewniać wielofunkcyjność i współpracę w złożonej przestrzeni, takiej jak tworzenie oprogramowania.

Zespół składający się z tylko dwóch Developerów nie potrzebuje Scruma, ponieważ może się łatwo komunikować bezpośrednio i zachować wydajność. Istnieje też większe ryzyko, że tych dwóch Developerów nie będzie miało wszystkich umiejętności wymaganych do wykonania całej pracy. Z drugiej strony zespoły z większą liczbą Developerów niż dziewięć wymagają zbyt dużej koordynacji. Te większe zespoły zwykle generują zbyt dużo złożoności, żeby mogły czerpać wartość z empiryzmu Scruma. W sytuacjach, gdy mamy więcej niż dziewięciu Developerów, konieczne może być utworzenie kilku Scrum Teamów, które nawet mogą tworzyć Nexus (splot). Omówię dokładniej Nexus i Scaled Professional Scrum w rozdziale 11 „Skalowalny Professional Scrum”.

Uwaga Product Owner i Scrum Master nie wliczają się do tej liczby 3–9 Developerów, o ile sami nie pełnią *również* roli Developera, który będzie pracować nad Sprint Backlogiem podczas Sprintu. Tak, czy owak, Scrum Team zwykle liczy 10 lub mniej osób.

Warto zwrócić uwagę, że to, iż dana osoba pełni rolę Developera w Scrumie, nie oznacza, że jest developerem w klasycznym sensie – czyli kimś, kto pisze kod. W zależności

od zadania, może zajmować się architekturą, interfejsem użytkownika, testami, schematem bazy danych, potokami wdrożeniowymi, wynikami testów, instalatorami albo dokumentacją. Każda z tych osób rozwija jakąś część produktu.

Tabela 1-1 zawiera listę działań wysokiego poziomu, które będą wykonywane przez Developerów w Scrum Teamie.

Tabela 1-1 Działania Developerów w Scrumie

Działanie	Kiedy
Współpraca z Product Ownerem przy prognozowaniu pracy na dany Sprint i opracowywaniu Celu Sprintu.	Sprint Planning
Współpraca z innymi Developerami przy planowaniu implementacji prognozowanej pracy.	Sprint Planning, Daily Scrum lub w razie potrzeby
Uczestnictwo w wydarzeniu Daily Scrum.	Codziennie
Opracowywanie Incrementu zgodnie z Definicją Ukończenia.	Po Sprint Planningu i przed Sprint Review
Współpraca z Product Ownerem nad dopracowaniem Product Backlogu.	Podczas Sprintu, jak określi Scrum Team
Wspólne określanie dodatkowych zadań, gdy prognozowana praca zostanie wcześnie zakończona.	Podczas Sprintu w razie potrzeby
Wspólne omawianie kompromisów i tworzenie planu awaryjnego, gdy prognozowane prace nie mogą zostać ukończone.	Podczas Sprintu w razie potrzeby
Wspieranie interesariuszy przy inspekcji Incrementu i zbieranie informacji zwrotnych.	Sprint Review lub w dowolnym czasie podczas Sprintu w razie potrzeby
Przemyślenie procesu i praktyk w celu ustalenia eksperymentów mających doprowadzić do poprawy.	Sprint Retrospective
Inspekcja, adaptacja, uczenie się i doskonalenie – życie wartościami Scruma.	Zawsze

Nie należy zakładać, że Developer będzie wykonywał tylko te typy zadań, w których jest dobry lub na których się dobrze zna. Na przykład to, że Dieter ma doświadczenie w programowaniu baz danych, nie oznacza, że będzie zajmował się tym rodzajem zadań. Jeśli podczas Sprintu okaże się, że następne logiczne zadanie do wykonania wymaga programowania bazy danych, a Dieter nie będzie dostępny, inny Developer powinien podjąć tę pracę, o ile to możliwe. Podczas opracowywania produktu osoba najlepiej predystynowana do wykonania danego zadania zostanie ustalona na podstawie

wielu czynników, w tym wiedzy i dostępności. To właśnie z tego powodu szacunki dotyczące Sprint Backlogu są dokonywane kolektywnie, a nie indywidualnie przez Developerów – nawet jeśli poszczególne osoby są specjalistami lub nawet ekspertami w danych dziedzinach. Dlatego też Scrum Team powinien mieć po kilku Developerów z potrzebnym zestawem umiejętności. Zajmiemy się tym dokładniej w rozdziale 8, „Skuteczna współpraca".

Wskazówka Znam bardzo niewiele Scrum Teamów, w których członkowie nazywają siebie „Developerami". Nadal istnieje odruch, który zrównuje termin „developer" z programistą lub osobą piszącą kod. Jest to też wzmacniane przez całą naszą branżę. W takich przypadkach tymczasowo odpowiednim substytutem może być użycie terminu „członek zespołu" – żeby wyraźnie obejmował on też testerów, projektantów i inne osoby niebędące programistami.

Jako kolektyw Developerzy muszą być interdyscyplinarni. To znaczy, że musi istnieć co najmniej jeden Developer w Scrum Teamie, który ma potrzebne umiejętności do wykonania każdego typu wymaganego działania. Innymi słowy, Developerzy – jako całość – muszą mieć wszystkie umiejętności wymagane do ukończenia pracy. Ta interdyscyplinarność nie oznacza, że każdy Developer jest interdyscyplinarny – choć byłoby to idealne. Najlepiej, aby zawsze było kilku Developerów, którzy mają wymaganą umiejętność. Jeśli tak nie jest, to zespół powinien dążyć do poprawy tego stanu rzeczy przez pracę w parach i grupach lub wykorzystując jakieś inne techniki instruktażowe podczas opracowywania produktu. Posiadanie w zespole tylko jednego Developera z kluczową umiejętnością jest ryzykowne.

Uwaga Prędkość (velocity) jest historyczną miarą elementów PBI, które udało się dostarczyć Scrum Teamowi. Może być mierzona poprzez liczbę tych elementów, rozmiar/wysiłek (np. punkty) albo ich wartość biznesową. Prędkość pojedynczego Sprintu nie jest przydatną miarą, ale śledzenie jej na przestrzeni kilku Sprintów pokazuje ogólny trend wydajności zespołu. Po znormalizowaniu prędkość może być użyteczna w planowaniu Sprintów i wydaniach produktu. Na przykład, jeśli prędkość zespołu wynosi średnio 20 punktów na Sprint, a Product Backlog pokazuje 12 elementów PBI, które trzeba opracować w celu osiągnięcia Celu Produktu i mających w sumie 96 punktów, to możemy oczekiwać, że wydanie produktu będzie dostępne za mniej więcej 5 Sprintów, czyli za 2 i pół miesiąca przy 2-tygodniowych Sprintach. Prędkość nie jest wymieniana w *Przewodniku po Scrumie* i jest uważana za praktykę uzupełniającą. Zespoły mogą też śledzić i wykorzystywać miary przepływu, takie jak przepustowość, jako miarę przeszłej wydajności. Przepływ i miary przepływu omówię w rozdziale 9 „Poprawianie przepływu".

Skład Scrum Teamu nie zmienia się podczas Sprintu. Jeśli miałby się zmienić, powinno to się odbywać tylko pomiędzy Sprintami. Zwykle jest to wynik decyzji podjętej wspólnie podczas wydarzenia Sprint Retrospective. Zmiany te mogą obejmować dodanie nowego członka zespołu, wymianę członka z innym zespołem, usunięcie członka z zespołu albo zmianę roli członka zespołu. Trzeba pamiętać, że wszelkie zmiany składu zespołu stanowią zakłócenie. Wydajność może początkowo zmniejszyć się na jakiś czas, a następnie powinna (miejmy nadzieję) ponownie wzrosnąć. Jeśli Scrum Team śledzi prędkość lub przepustowość prac, to takie zakłócenie będzie wyraźnie widoczne.

Studium przypadku Fabrikam Fiber

Developerzy w Scrum Teamie Fabrikam Fiber to pięć interdyscyplinarnych osób o różnym doświadczeniu, zestawach umiejętności i poziomach zaawansowania. Są nimi Andy, Dave, Dieter, Toni i Richard (ja sam). Andy i Toni mają doświadczenie w dziedzinie architektury, projektowania i pewną znajomość języka C#. Dave, Dieter i ja mamy solidne doświadczenie w programowaniu w języku C#. Dieter i ja mamy też doświadczenie w programowaniu SQL i Azure, w tym w programowaniu skryptów Windows PowerShell. Jako zespół wszyscy uczestniczyliśmy w szkoleniu Professional Scrum Developer organizowanym przez Scrum.org i uzyskaliśmy po nim pozytywne oceny.

Product Owner

Product Owner reprezentuje głos naszego użytkownika. To oznacza, że Product Owner zna nie tylko produkt, jego domenę, wizję i cele, ale też jego użytkowników. Sama wiedza, jak produkt działa i co w nim naprawić, nie wystarcza, aby być kompetentnym Product Ownerem. Dobry Product Owner jest na bieżąco z potrzebami swoich użytkowników. Dobry Product Owner podziela pasję swoich użytkowników i odczuwa empatię dla ich zmagań i oczekiwań. Product Owner reprezentuje też Developerów przed organizacją – przynajmniej do momentu, gdy Developerzy nie zostaną wprowadzeni do organizacji i nie zaczną współpracować bezpośrednio z innymi osobami w organizacji.

Uwaga Przez lata słyszałem, że Product Owner jest głosem interesariuszy lub klienta. Choć jest to prawda, to wolę myśleć o Product Ownerze głównie jako o głosie *użytkownika*. Jaka jest różnica? Interesariuszem jest każdy, kto jest zainteresowany produktem lub jego rozwojem. Klientem jest zwykle ten, kto sponsoruje lub płaci za produkt. Użytkownikiem jest ten, kto faktycznie go używa. Product Owner w Professional Scrum dąży do zadowolenia wszystkich zainteresowanych stron.

Product Owner musi reprezentować potrzeby użytkownika i przekierowywać wartość w ich kierunku, a nie tylko starać się zadowolić osobę, która płaci. W Scrum Teamie jest tylko jeden Product Owner, co pozwala unikać zamieszania. Gdy Developerzy mają jakieś pytanie dotyczące produktu, które trzeba przedstawić interesariuszowi, powinni przede wszystkim zwrócić się do Product Ownera. Product Owner być może będzie musiał skonsultować się z innymi, aby uzyskać odpowiedzi, zwłaszcza w przypadku dużych lub bardziej złożonych produktów. Product Owner powinien być osobą, do której kierowane są wszystkie pytania dotyczące wizji, celów i funkcjonalności produktu.

Product Owner odpowiada za maksymalizowanie wartości produktu poprzez pracę Developerów. Głównym narzędziem Product Ownera do komunikacji z Developerami jest udoskonalony i uporządkowany Product Backlog. Product Owner współpracuje z Developerami nad tym, co i kiedy ma zostać opracowane. Tabela 1-2 wymienia typowe interakcje Developerów z Product Ownerem.

Częstym nieporozumieniem – które zostało poprawione w *Przewodniku po Scrumie* z 2020 roku – jest to, że Developerzy opracowują produkt. W istocie to Scrum Team opracowuje i rozwija produkt – poprzez współpracę wszystkich członków zespołu.

Wskazówka Product Owner w Professional Scrum powinien znać produkt, znać dziedzinę produktu, znać klientów produktu, znać użytkowników produktu, znać Scrum, mieć uprawnienia do podejmowania decyzji związanych z kierunkiem produktu, być dostępnym dla reszty Scrum Teamu i mieć dobre umiejętności interpersonalne. Nie spotkałem jeszcze Product Ownera, który spełniałby wszystkie te kryteria. Spotkałem wielu Product Ownerów, którzy chcieli się doskonalić w tych obszarach i dążyli do tego celu.

Tabela 1-2 Interakcje Developera z Product Ownerem

Interakcja	Kiedy
Wspólne planowanie Sprintu i prognozowanie elementów PBI.	Sprint Planning
Uzyskiwanie odpowiedzi na pytania dotyczące produktu/dziedziny i przedstawianie ich interesariuszom.	Podczas Sprintu w razie potrzeby
Udoskonalanie Product Backlogu.	Podczas Sprintu, jak określi Scrum Team
Współpraca przy podejmowaniu dodatkowych zadań.	Podczas Sprintu w razie potrzeby
Współpraca przy planowaniu pracy awaryjnej.	Podczas Sprintu w razie potrzeby
Pomoc Product Ownerowi przy inspekcji Incrementu i innych pojawiających się prac.	Podczas Sprintu w razie potrzeby

Tabela 1-2 Interakcje Developera z Product Ownerem

Interakcja	Kiedy
Pomoc Product Ownerowi podczas inspekcji przez interesariuszy i zbieraniu informacji zwrotnych.	Przynajmniej podczas Sprint Review, ale również podczas Sprintu w razie potrzeby
Współpraca przy inspekcji praktyk Scrum Teamu i planowania udoskonalania.	Sprint Retrospective
Współpraca przy tworzeniu Definicji Ukończenia.	Sprint Retrospective

Profesjonalne Scrum Teamy rozumieją podział obowiązków pomiędzy Product Ownera a Developerów i oczekują, że każda ze stron będzie wypełniać swoją rolę. Choć *Przewodnik po Scrumie* nie określa jasno, że Product Owner nie może być Scrum Masterem lub Developerem, to uważam, że dobrze jest utrzymać taki podział. Receptą na sukces jest, aby Product Owner koncentrował się na tym, *co* rozwijać, Developerzy koncentrowali się na tym, *jak* to rozwijać, a Scrum Master skupiał się na tym, aby wszyscy rozumieli i stosowali się do reguł Scruma.

Ponieważ Product Owner może odpowiadać przed organizacją za zyski lub straty produktu, Product Owner powinien stale czuwać nad optymalizowaniem wartości produktu. Product Owner w Professional Scrum jest zaangażowanym Product Ownerem. Stale troszczy się o to, co jest najlepsze dla produktu, a co ważniejsze, co jest najlepsze dla jego użytkowników.

Studium przypadku Fabrikam Fiber

Paula jest Product Ownerem aplikacji webowej Fabrikam Fiber. Zaczynając wcześniej jako technik, zna słabe strony produktu i zmagania jego użytkowników. Ta świadomość inspiruje ją do ciągłej poprawy i ewolucji możliwości produktu. Nawet lubi się chwalić, że jest najintensywniejszym użytkownikiem tej aplikacji. Jej wizja polega na stopniowym usprawnianiu Fabrikam Fiber, aż większość użytkowników będzie mogła uzyskać rozwiązanie swoich problemów w tym samym dniu. Paula jest poinformowanym i zaangażowanym Product Ownerem, który jest dostępny, gdy to konieczne i ma władzę podejmowania potrzebnych decyzji. Paula korzysta ze Scruma od około trzech lat. Przeszła szkolenia Professional Scrum Foundations oraz Professional Scrum Product Owner w Scrum.org.

Scrum Master

Scrum Master promuje wartości, praktyki i reguły Scruma w Scrum Teamie i całej organizacji. Scrum Master zapewnia, że Product Owner i Developerzy są funkcjonalni i wydajni, zapewniając niezbędne wskazówki i wsparcie. Scrum Master ponosi też

odpowiedzialność za zapewnienie, że Scrum jest rozumiany przez wszystkie zainteresowane strony i że każdy stosuje się do jego reguł.

Uwaga Scrum Master nie jest tym samym, co menedżer projektu. *Jest* uznawany za menedżera – ale menedżera Scruma i jego implementacji, a nie menedżera projektu, ludzi lub produktu. Dlatego powinien mieć uprawnienia do wprowadzania zmian i usuwania przeszkód.

Scrum Master musi być czujny, dając jednocześnie organizacji czas na aklimatyzację i zdanie sobie sprawy z korzyści płynących ze Scruma. Oznacza to trzymanie w ryzach wszelkich dysfunkcyjnych przyzwyczajeń (takich, jak stare metody „kaskadowe"). Oznacza to również utrzymywanie w ryzach wszelkich nieoświeconych menedżerów przy ciągłej walce z iluzją, że dowodzenie, kontrola i nieprzejrzystość równają się lepszemu i szybszemu dostarczaniu wartości. Czasami Scrum Master może stać się de facto agentem zmian, prowadząc wysiłki w kierunku przyjęcia Scruma w całej organizacji. Jeśli tak jest, to niezłomność Scrum Mastera musi być w stanie się skalować. To również ilustruje potrzebę posiadania świetnych umiejętności interpersonalnych przez Scrum Mastera.

Scrum Master może działać jako trener zapewniający, że zespół sam się zarządza oraz jest funkcjonalny i wydajny. Ta rola może obejmować ochronę zespołu przed przerwami zakłócającymi skupienie i innymi konfliktami zewnętrznymi oraz usuwanie wszelkich przeszkód na drodze do postępów zespołu. Umiejętność Scrum Mastera do służenia zespołowi przez usuwanie przeszkód na drodze do sukcesu jest istotnym elementem Scruma.

Jako służebny przywódca Scrum Master osiąga wyniki, nadając wyższy priorytet potrzebom zespołu. Scrum Master może również służyć interesariuszom i innym osobom w organizacji, pomagając im w zrozumieniu ram Scruma i oczekiwań ze strony różnych graczy. Służebni przywódcy są często postrzegani jako pokorni opiekunowie ludzi i procesów, w które są zaangażowani. Dzięki postawie „Co mogę dzisiaj dla ciebie zrobić?" Scrum Master sprzyja środowisku współpracy i szacunku, zapewniając żyzną glebę dla wysokowydajnego Scrum Teamu. Lao Tzu, starożytny filozof chiński, wyraził to najlepiej:

> *Gdy rządzi mistrz, ludzie nie są świadomi tego, że istnieje. W dalszej kolejności najlepszym przywódcą jest ten, który jest kochany. Następnie ten, którego się boją. Najgorszy jest ten, którym gardzą. Jeśli nie zaufasz ludziom, sprawisz, że będą niegodni zaufania. Mistrz nie mówi, ale działa. Gdy wykona swoją pracę, ludzie powiedzą „Wspaniale: zrobiliśmy to, całkiem sami!"*

Scrum Master nie jest rolą techniczną. Mocne doświadczenie w danej dziedzinie (takiej jak tworzenie oprogramowania) nie jest konieczne, choć czasem może być pomocne.

Scrum Master musi bardzo dobrze znać Scrum – to jest jego dziedzina. Nie podlega to dyskusji. Dobry Scrum Master będzie też miał dobre zdolności komunikacyjne i interpersonalne. Może ułatwiać interakcje z innymi członkami zespołu lub umożliwiać współpracę pomiędzy rolami, wydarzeniami lub innymi osobami w organizacji. Ważne są te „miękkie" umiejętności. Trzeba mieć to na uwadze, zastanawiając się nad tym, kto mógłby być dobrym Scrum Masterem. Tabela 1-3 wymienia sposoby, na jakie Scrum Master służy Developerom.

Wskazówka Moim zdaniem tradycyjni menedżerowie projektów *nie* stają się dobrymi Scrum Masterami. Niestety jest to typowy odruch w organizacjach przyjmujących Scrum. Na przykład decydenci często wysyłają „Rogera" z certyfikatem PMI, kochającego wykresy Gantta, eksperta od Microsoft Project na szkolenie Professional Scrum Master. Oczekują, że właśnie Roger pokieruje zmianami. Najczęściej obserwuję, że albo nawyki Rogera z zarządzania projektami, albo jego koledzy i menedżerowie negatywnie wpływają na przyjęcie Scruma.

Tabela 1-3 Sposoby służenia Developerom przez Scrum Mastera

Usługa	Kiedy
Pomaga organizować wydarzenia Scruma, gdy Developerzy nie mogą tego zrobić sami.	Podczas Sprintu w razie potrzeby
Identyfikuje, dokumentuje i usuwa przeszkody.	Podczas Sprintu w razie potrzeby
Zapewnia szkolenia, mentoring i motywowanie.	Podczas Sprintu w razie potrzeby
Szkoli Developerów w zakresie samodzielnego zarządzania.	Podczas Sprintu w razie potrzeby
Jest wysłannikiem Developerów przed organizacją.	Podczas Sprintu w razie potrzeby
Bierze udział w nieproduktywnych, ale „obowiązkowych" spotkaniach w imieniu Developerów.	Podczas Sprintu w razie potrzeby
Chroni Developerów przed przerywaniem pracy i zakłóceniami, aby dbać o ich koncentrację.	Podczas Sprintu w razie potrzeby
Staje się coraz mniej potrzebny.	Z czasem, gdy Developerzy się doskonalą

Obowiązki Scrum Mastera nie muszą wymagać pełnoetatowego zaangażowania. Zespół Professional Scrum Team może wybrać Developera, aby odgrywał przez część czasu rolę Scrum Mastera. Tę rolę mogą podejmować rotacyjnie różni Developerzy. Pełnoetatowy Scrum Master może zmieniać się też w Developera albo wspomagać inne Scrum Teamy, które pojawią się w organizacji. Rola Scrum Mastera jest bardziej elastyczna w tym zakresie niż pozostałe role. Wystarczy, że Scrum Team rozumie

i przestrzega zasad Scruma oraz ma dostęp do kogoś, kto może wykonywać obowiązki Scum Mastera w razie potrzeby.

Wskazówka Umiejętności Scrum Mastera są wyjątkowe i ważne. Najlepszym Scrum Masterem, jakiego znałem, był Brian. Nie miał doświadczenia biznesowego ani technicznego. Był wcześniej pracownikiem poradni antynarkotykowej i antyalkoholowej, co oznaczało, że umiał słuchać, zachęcać, motywować, a także rozpoznać, gdy ktoś się obija, nie pracuje na miarę swoich możliwości albo nie mówi prawdy. Dla niektórych bycie Scrum Masterem jest wyborem zawodowym. Z mojego doświadczenia najlepsze są osoby otwarte, zmotywowane i oddane ciągłemu doskonaleniu swoich umiejętności, ponieważ służą one zespołowi. Taki Scrum Master powinien być tylko Scrum Masterem i nie powinien zmieniać swojej roli na inną. Będzie stanowił większą wartość dla zespołu i organizacji jako pełnoetatowy Scrum Master. Tego typu osoby są bardzo cenne.

Studium przypadku Fabrikam Fiber

Scott został sprowadzony w zeszłym roku i miał służyć jako Scrum Master. Najpierw służył innemu zespołowi, zapewniając konieczne szkolenia w celu przekształcenia ich w wysokowydajny Professional Scrum Team. Zarząd zgadza się, że będzie dobrym Scrum Masterem dla zespołu Pauli. Planują też skorzystać z usług Scotta przy pomocy innym zespołom w organizacji w poznawaniu i przyjęciu Scruma. Scott ma wieloletnie, praktyczne doświadczenie w dziedzinie Scruma w różnych firmach i zespołach. Uczestniczył w kilku szkoleniach Professional Scrum i jest aktywnym członkiem społeczności Scrum.org.

Interesariusze

Choć nie jest to rola oficjalnie zdefiniowana w *Przewodniku po Scrumie*, interesariusze obejmują wszystkich innych zainteresowanych rozwojem produktu. Interesariuszami mogą być menedżerowie, dyrektorzy, kierownicy, członkowie zarządu, analitycy, eksperci dziedzinowi, prawnicy, sponsorzy, członkowie innych zespołów, klienci i użytkownicy oprogramowania. Interesariusze są bardzo ważni. Reprezentują konieczność istnienia danego produktu z różnych perspektyw. Wpływają również na wizję, cele i użyteczność produktu, wpływając na Product Ownera. Bez interesariuszy, kto korzystałby z produktu, płacił za jego opracowanie lub czerpał z niego korzyści?

Z mojego doświadczenia Developerzy często mają tendencję do niedoceniania osób nietechnicznych. To dość niefortunne – nie należy ignorować ineresariuszy. Powinni być oni zaangażowani. Z kolei niektórzy interesariusze mogą być *zbytnio* zainteresowani rozwojem produktu i jego stanem, co może powodować rozproszenie uwagi i niszczenie koncentracji. Istnieje wiele nieporozumień na temat tego, kiedy i dlaczego interesariusze i Developerzy powinni wchodzić w interakcję. Czytając *Przewodnik*

po Scrumie, można by sądzić, że interakcja ma miejsce tylko podczas wydarzenia Sprint Review. Jak widać z tabeli 1-4, interesariusze i Developerzy mogą wchodzić w interakcje w wielu miejscach podczas Sprintu.

Tabela 1-4 Interakcje Developerów z interesariuszami

Interakcja	Kiedy
Współpraca nad udoskonalaniem Product Backlogu.	Podczas Sprintu, jak określi Scrum Team
Współpraca nad odpowiadaniem na pytania Developerów dotyczące elementów PBI (szacowanie, planowanie, projektowanie, budowanie, testowanie, itd.).	Podczas Sprintu w razie potrzeby
Współpraca nad inspekcją Incrementu i zbieraniem informacji zwrotnych.	Przynajmniej podczas Sprint Review, ale również podczas Sprintu w razie potrzeby

Inspekcja i zbieranie informacji zwrotnych na temat produktu, na przykład zgłaszanie zapotrzebowania na jakąś funkcję, powinny obejmować Product Ownera. Inspekcja i zbieranie informacji zwrotnych na temat procesu rozwoju produktu, na przykład sprawdzanie stanu postępów, powinny być obsługiwane przez Scrum Mastera. Innymi słowy, interesariusze powinni niemal zawsze być trzymani z dala od procesu rozwoju produktu – o ile nie zostaną zaproszeni przez Developerów. Stosowanie się do tych zaleceń pomoże chronić koncentrację Developerów.

Wskazówka Wykresy burndown, burnup lub inne dane analityczne umieszczane we wspólnym obszarze lub na tablicy są świetnym sposobem na bieżące informowanie interesariuszy. Pozwala to utrzymywać przerwy w działaniu Scrum Teamu do minimum. Jeśli ktoś ma jakieś pytania, powinien zwracać się do Scrum Mastera.

Scrum Master powinien dążyć do utrzymywania interesariuszy poza różnymi wydarzeniami Scruma, poza Sprint Review i być może Sprint Planningiem. Interesariusze nie powinni uczestniczyć w działaniach związanych z planowaniem, rozwijaniem i udoskonalaniem produktu, o ile nie zostaną o to poproszeni. Uczestnictwo w jakimkolwiek wydarzeniu powinno odbywać się tylko na zaproszenie Scrum Teamu. Interesariusze nie powinni być uczestnikami Daily Scrum, ponieważ celem tego wydarzenia jest pozwolenie Developerom na zsynchronizowanie ze sobą najbliższych prac. Nawet obecność Product Ownera podczas Daily Scrum może być traktowana jako zakłócenie dla celu tego wydarzenia.

Studium przypadku Fabrikam Fiber

Firma Fabrikam Fiber istnieje od kilku lat. Zmieniała już raz swoją nazwę i przeszła kilka reorganizacji. Głównymi interesariuszami są jej klienci, którzy są użytkownikami najważniejszej dostarczanej wartości: połączenia internetowego. Wraz z technikami obsługi, innymi jednostkami biznesowymi (w tym sprzedażą i marketingiem) są interesariuszami aplikacji webowej Fabrikam Fiber. Większość grup interesariuszy nie ma zbytniego doświadczenia technicznego, jeśli chodzi o oprogramowanie. Niektórzy wewnętrzni interesariusze mają jednak głęboką wiedzę fachową w dziedzinie sprzętu i sieci. Wszyscy interesariusze są dość otwarci na przekazywanie informacji zwrotnych na temat aplikacji webowej. Paula rozumie znaczenie pozyskiwania informacji zwrotnych od klientów. W tym celu nalegała na ustanowienie adresu e-mail *wish@fabrikam.com* do zbierania informacji zwrotnych przez pocztę elektroniczną. Te wiadomości są przekazywane do osoby z pomocy technicznej, która klasyfikuje zgłoszenia, a następnie współpracuje z Paulą nad dodawaniem elementów PBI do Product Backlogu.

Wydarzenia Scruma

Ramy postępowania Scruma wykorzystują wydarzenia do ustrukturyzowania różnych przepływów zadań w inkrementacyjnym rozwoju produktu. Każde wydarzenie ma określone ramy czasowe, co oznacza, że istnieje stały okres czasu na przeprowadzenie działań w ramach danego wydarzenia. Te ograniczenia czasowe minimalizują straty przez zapewnianie odpowiedniej ilości czasu poświęcanego na skupienie się na określonym działaniu. Te wydarzenia Scruma mają na celu utrzymanie pewnej regularności i rytmu. Mają też minimalizować konieczność niepotrzebnych lub naprędce zwoływanych spotkań, które nie są częścią Scruma.

Wszystkie wydarzenia Scruma stanowią formalną okazję do inspekcji i adaptacji jakiegoś elementu. Inspekcja pozwala zespołowi oceniać postępy na drodze do celu, a także identyfikować zmiany w obecnym planie. Jeśli inspekcja stwierdzi wystąpienie niedopuszczalnego odchylenia, to należy dokonać adaptacji (dostosowania). Te dostosowania powinny być dokonywane jak najszybciej, aby zminimalizować dalsze odstępstwa. Brak udziału w wydarzeniu Scruma prowadzi do zmniejszonej przejrzystości i stanowi utraconą okazję do inspekcji i adaptacji.

Wracając do rysunku 1-1, możemy zobaczyć, że w Scrumie jest pięć wydarzeń:

- **Sprint** Pojemnik na pozostałe cztery wydarzenia. Cała praca potrzebna do osiągnięcia Celu Produktu, obejmująca Sprint Planning, wydarzenia Daily Scrum, Sprint Review oraz Sprint Retrospective odbywa się w ramach Sprintu. Sprinty są wydarzeniami o ustalonym czasie trwania wynoszącym miesiąc lub krócej. Nowy Sprint zaczyna się natychmiast po zakończeniu poprzedniego Sprintu.

- **Sprint Planning** Zapoczątkowuje Sprint poprzez rozplanowanie pracy do wykonania w Sprincie. To planowanie jest przeprowadzane poprzez wspólną pracę całego Scrum Teamu.
- **Daily Scrum** To wydarzenie jest używane do inspekcji postępów w dążeniu do osiągnięcia Celu Sprintu oraz w razie potrzeby adaptacji Sprint Backlogu poprzez dostosowanie planowanej przyszłej pracy. Tylko Developerzy uczestniczą w Daily Scrum.
- **Sprint Review** To wydarzenie jest używane do inspekcji wyników Sprintu oraz wskazania przyszłych adaptacji. Scrum Team przedstawia wyniki swojej pracy kluczowym interesariuszom, Product Backlog jest aktualizowany i omawiane są postępy w dążeniu do Celu Produktu.
- **Sprint Retrospective** To wydarzenie jest używane do planowania sposobów na podnoszenie jakości i efektywności.

Uwaga Istnieje przekonanie, że Sprint jest przedziałem czasu następującym *po* wydarzeniu Sprint Planning i kończącym się przed wydarzeniem Sprint Review, w którym przeprowadzane są faktyczne prace rozwojowe. To nie jest poprawne podejście. Sprint jest w istocie zewnętrznym „pojemnikiem" na cztery pozostałe wydarzenia. To oznacza, że Sprint rozpoczyna się w istocie wcześniej niż początek wydarzenia Sprint Planning. Czas pomiędzy wydarzeniami Sprint Planning i Sprint Review nie ma technicznej nazwy, ale większość odnosi się do niego jako „rozwój produktu". Sprint kończy się po wydarzeniu Sprint Retrospective, a następnie rozpoczyna się ponownie wraz z nowym wydarzeniem Sprint Planning.

Sprint

Sprint jest ustalonym okresem, w którym opracowywany jest Increment produktu. *Sprint* jest scrumowym określeniem iteracji. Sprinty trwają miesiąc lub krócej i następują bezpośrednio po sobie. Częstotliwość informacji zwrotnych, doświadczenie i techniczna doskonałość zespołu oraz zapotrzebowanie organizacji i Product Ownera na elastyczność stanowią kluczowe czynniki w określaniu długości Sprintu. Na przykład, jeśli produktem jest firmowa aplikacja pulpitowa z dość dobrze zdefiniowanymi celami i nie ma dużego odchylenia w planach, dłuższe Sprinty są w porządku. Jeśli aplikacja jest opartym na chmurze produktem programowym jako usługą (SaaS) z kilkoma konkurentami i wymagającymi klientami, bardziej pożądane są krótsze Sprinty. Interesariusze i Scrum Team muszą współpracować nad ustaleniem idealnej długości Sprintu.

Sprint Planning, rozwijanie produktu, wydarzenia Daily Scrum, Sprint Review i Sprint Retrospective odbywają się podczas Sprintu. Gdy zaczniemy korzystać ze Scruma, *zawsze* będziemy wewnątrz Sprintu – przy założeniu, że nadal mamy produkt, interesariuszy i chęć inwestowania w nowe możliwości. Gdy kończy się Sprint

Retrospective, zaczyna się następny Sprint i od nowa powtarzają się jego wewnętrzne wydarzenia. Nie powinno być żadnych przerw pomiędzy Sprintami.

Kiedyś zapytałem Kena Schwabera, jak długi powinien być Sprint. Jego odpowiedź brzmiała „tak krótki, jak to możliwe, ale nie krótszy". Sprinty dłuższe niż cztery tygodnie (jeden miesiąc) mają „brzydki zapach" – zapach działania kaskadowego. Gdy długość Sprintu jest dłuższa niż miesiąc, definicja tego, co jest budowane, może się zmieniać albo złożoność i ryzyko mogą wzrastać. Dzięki ograniczeniu maksymalnej długości Sprintu, co najwyżej jeden miesiąc wysiłków związanych z rozwijaniem produktu zostanie zmarnowany, a nie kilka miesięcy, jak w klasycznym projekcie kaskadowym.

Z drugiej strony Sprinty o długości krótszej niż jeden tydzień są możliwe, ale powinny być stosowane tylko przez wysokowydajne zespoły Professional Scrum Team. Nawet przy bardzo krótkich Sprintach trzeba uwzględniać narzuty wynikające z wydarzeń wewnętrznych, co zostawia jeszcze mniej czasu (procentowo) na rzeczywisty rozwój produktu. Zespoły pracujące w takich „mikro-Sprintach" muszą codziennie działać na najwyższych obrotach.

Idealnie długość Sprintu nie powinna się zmieniać. Jeśli już musi, to może się zmieniać tylko pomiędzy Sprintami jako wynik decyzji podejmowanej podczas Sprint Retrospective. Wszelkie zmiany długości Sprintu będą powodować zakłócenia rytmu Developerów, wpływając na prognozy i plany. Choć z czasem się to skoryguje, to nie jest dobrym pomysłem stałe wprowadzanie takiego chaosu.

Każdy Sprint jest jak mini-projekt. W istocie, gdy przedstawiam Scrum nowej organizacji lub zespołowi, polecam zastąpienie w rozmowach użycia terminu „projekt" terminem „Sprint" lub „Sprinty". Na przykład zamiast odnosić się do „projektu integracji z systemem CRM", można odwoływać się do tej inicjatywy jako Sprinty 17–19, w których prognozowane i opracowywane będą elementy PBI związane z integracją z CRM.

Sprint zawiera definicję tego, „co" ma zostać opracowane. Obejmuje to też elastyczne podejście do tego, „jak" to opracować. Podczas Sprintu wykonywane są wszystkie aspekty pracy nad rozwojem produktu. W przypadku produktu programistycznego, będzie to coś więcej niż tylko projektowanie, kodowanie i testowanie. Zakres prac może być doprecyzowany w miarę wzrostu wiedzy, a Product Owner może współpracować z Developerami nad renegocjacją i dodawaniem nowych elementów albo wymienianiem elementów w Sprint Backlogu – o ile będą pasować do Celu Sprintu. Developerzy nie mogą zmniejszać celów jakościowych w celu ukończenia swojej pracy. Powstaje wynikowy Increment produktu, który podlega inspekcji przez interesariuszy i być może jest nawet wydawany.

Wybór dnia tygodnia, w którym zaczynać (i kończyć) Sprint, zależy całkowicie od Scrum Teamu. Niektórzy praktycy wolą poniedziałki lub piątki. Ja wolę środek tygodnia, tak aby było jak największe prawdopodobieństwo, że obecny będzie cały zespół i wszyscy będą uczestniczyć maksymalnie skoncentrowani. Harmonogram wydarzeń może być nieco zmieniany w okresach świątecznych, ale zalecam ścisłe

trzymanie się terminów Sprintów. Na przykład, jeśli Sprint 26 kończy się podczas święta, należy zostawić jego datę rozpoczęcia i zakończenia bez zmian, aby utrzymać dwutygodniowy rytm i po prostu zaplanować Sprint Review i Sprint Retrospective na koniec pierwszego tygodnia. Drobne korekty tego typu mogą być konieczne na przestrzeni całego roku.

Studium przypadku Fabrikam Fiber

Wcześniej Scrum Team stosował czterotygodniowe Sprinty. Członkowie zespołu uważali, że dłuższy przedział czasu będzie bliższy kwartalnemu harmonogramowi dostaw, do którego byli przyzwyczajeni. Niestety, ponieważ Scrum stanowił nowość dla zespołu, to nadal stosowali sekwencyjne podejście do rozwijania produktu. Poświęcali dużo czasu na analizę i projektowanie na początku Sprintu i odkładali testowanie na sam koniec. Wynikający z tego kryzys w ostatnich dniach Sprintu nie był równoważony i powodowało to powrót do przyzwyczajeń z pracy kaskadowej. Zespół nie doświadczał oczekiwanego wzrostu wydajności. Gdy Scott został Scrum Masterem, zaproponował przejście na dwutygodniowe Sprinty. To spowodowało zwiększone odczuwanie pilności działań przez Developerów i zmianę ich pracy poprzez utrzymywanie odpowiedniego poziomu intensywności przez cały Sprint. Scott zalecił też rozpoczynanie Sprintu w środę. Ta zmiana zwiększyła prawdopodobieństwo, że cały zespół będzie dostępny i będzie działał w maksymalnym skupieniu. Pozwoliło to też interesariuszom na łatwiejsze podróże w celu osobistego wzięcia udziału w wydarzeniu Sprint Review i następującym po nim Sprint Planningu bez konieczności zostawania na weekend. Scrum Team ukończył wiele udanych Sprintów w tym dwutygodniowym rytmie.

Sprint Planning

Sprint Planning służy do wybierania i planowania pracy, która będzie wykonywana podczas Sprintu. Jest to pierwsze wydarzenie, które występuje podczas Sprintu. Bierze w nim udział cały Scrum Team. Jako wejście dla wydarzenia Sprint Planning wymagany jest udoskonalony i uporządkowany (przez priorytety) Product Backlog. Developerzy współpracują z Product Ownerem nad zakresem prac, które mogą zostać wykonane. Jest to zwane *prognozowaniem*. Prognozowana praca wraz z Celem Sprintu i planem wykonania pracy stanowią wyjścia z tego wydarzenia. Wyjścia te są utrzymywane w Sprint Backlogu.

Sprint Planning ma ustalony czas trwania, więc wszyscy muszą być maksymalnie skupieni. Zakłócenia, takie jak rozmowy nie na temat, powinny być minimalizowane. Sprint Planning powinien trwać maksymalnie osiem godzin, ale w praktyce ta długość powinna być funkcją długości Sprintu, jak widać w tabeli 1-5.

Tabela 1-5 Długość wydarzenia Sprint Planning

Długość Sprintu	Długość wydarzenia Sprint Planning
4 tygodnie	~ 8 godzin lub mniej
3 tygodnie	~ 6 godzin lub mniej
2 tygodnie	~ 4 godziny lub mniej
1 tydzień	~ 2 godziny lub mniej
Mniej niż tydzień	Proporcjonalnie do powyższych długości

Sprint Planning składa się z trzech tematów. Każdy z tych tematów odpowiada na pytanie: dlaczego, co i jak. Odpowiedź na każde z tych pytań stanowi wyjście wydarzenia Sprint Planning. Odpowiedź na pytanie *Dlaczego* stanowi cel sprintu. Odpowiedź na pytanie *Co* stanowi prognoza. Odpowiedź na pytanie *Jak* stanowi plan. Na kolejnych stronach szczegółowo omówię każdy z tych tematów.

Temat 1: Dlaczego ten Sprint ma wartość? Product Owner proponuje, jak zwiększyć wartość produktu oraz jego użyteczność w bieżącym Sprincie. Następnie cały Scrum Team współpracuje nad sformułowaniem Celu Sprintu, który opisuje to, dlaczego dany Sprint jest wartościowy dla interesariuszy. Cel Sprintu określa w formie narracyjnej cel, do którego dążą Developerzy podczas rozwoju danego Incrementu. Cel Sprintu zapewnia też interesariuszom możliwość zobaczenia zarysu tego, nad czym pracuje Scrum Team. Cel Sprintu musi zostać określony przed zakończeniem Sprint Planningu.

Choć Product Owner może przynieść Cel Produktu i inne cele biznesowe na Sprint Planning, to ważne jest, aby cały Scrum Team razem sfinalizował Cel Sprintu i zgodził się na jego brzmienie i znaczenie. Każdy członek Scrum Teamu powinien go następnie zapamiętać. Interesariusze również powinni mieć do niego dostęp. Po rozpoczęciu prac nad rozwojem produktu (to znaczy po zakończeniu Sprint Planningu) Cel Sprintu nie powinien być zmieniany. Stanowi on *motyw przewodni*, który zespół będzie się starał osiągnąć. Jeśli Developerzy nie są w stanie osiągnąć Celu Sprintu albo jeśli cel stanie się nieaktualny, Product Owner może podjąć decyzję o anulowaniu Sprintu – co jest kolejną wskazówką, jak duże jest znaczenie Celu Sprintu. Anulowanie Sprintu jest omawiane w rozdziale 6 „Sprint".

Przewodnik po Scrumie nie określa, co ma pierwszeństwo, Cel Sprintu, czy prognoza. Niektóre Scrum Teamy wolą opracować Cel Sprintu najpierw albo przynajmniej równolegle z prognozowaniem pracy. W ten sposób istnieje większa spójność z celem i elementami PBI, które są rozwijane podczas Sprintu. Ta spójność ułatwia zrozumienie wartości Incrementu i tego, jak pasuje on do celów produktu lub wydania. Inne zespoły mogą woleć uwzględniać w prognozie najwyżej uszeregowane elementy Product Backlogu, a następnie tworzyć narrację dla Celu Sprintu wokół tych elementów. Oba podejścia mogą być trudne dla zespołów, które muszą dostarczyć oddzielne funkcje i poprawki błędów w danym Sprincie.

Cel Sprintu daje Developerom pewną elastyczność i wskazówki dotyczące funkcjonalności implementowanej w ramach Sprintu. Nawet jeśli dostarczą mniej elementów PBI, niż zostało prognozowanych w Sprint Planningu, nadal mogą osiągnąć swój Cel Sprintu. Na przykład powiedzmy, że Developerzy prognozowali następujące elementy PBI podczas Sprint Planningu:

1. Dodanie wpisów z serwisu Twitter na stronie głównej.
2. Utworzenie strony na Facebooku dla firmy.
3. Utworzenie i prowadzenie serwisu wiki dla wsparcia produktu.

Mając takie prognozy, Cel Sprintu mógłby brzmieć „Zwiększyć świadomość naszej firmy i jej produktów wśród społeczności" albo po prostu „Uspołecznienie Fabrikam Fiber". Podczas pracy Developerzy powinni mieć ten cel na uwadze. Jeśli zespół nie będzie w stanie ukończyć trzeciego elementu PBI (wiki), to nie będzie porażki, ponieważ nadal będzie w stanie osiągnąć Cel Sprintu poprzez ukończenie pierwszych dwóch elementów PBI.

Jeśli wygląda to tak, jakby Cele Sprintu dawały Developerom pewną swobodę działania, to tak jest w istocie. Pamiętajmy, że to, co robią Developerzy, jest trudne i ryzykowne. Dlatego właśnie powinni *prognozować* poszczególne elementy, jeśli sądzą, że mogą je dostarczyć, ale *angażować* się w Cel Sprintu, który je uosabia.

Uwaga Cel Sprintu nie powinien być zbyt szeroki, jak na przykład „Usprawnić produkt". Cel Sprintu nie powinien być też zbyt specyficzny, taki jak „Zaimplementować PBI #1, PBI #2, PBI #3, PBI #4 i PBI #5..." Może być trudno osiągnąć zbyt ogólny Cel Sprintu, a zbyt złożony Cel Sprintu może zbytnio dzielić uwagę zespołu i nie pozwalać na elastyczność. Cel Sprintu powinien opisywać powód, dla którego podejmujemy dany Sprint, a nie tylko streszczać prognozowane elementy PBI. Dobry Cel Sprintu pomaga zespołowi zrozumieć cel i znaczenie wykonywanej pracy, co jest dobrym motywatorem.

Temat 2: Co może zostać ukończone w tym Sprincie? Podczas Sprint Planningu Developerzy biorą po kolei pod uwagę najwyżej uszeregowane elementy PBI z Product Backlogu. O tej kolejności decyduje Product Owner. Omawiane są szczegóły i kryteria akceptacji dla każdego elementu PBI. Dodatkowe wyjaśnienia są przedstawiane przez Product Ownera i innych interesariuszy, którzy mogą być w stanie zapewnić wiedzę fachową i wyjaśnienia z danej dziedziny.

Po uzyskaniu odpowiedniego poziomu zrozumienia elementu PBI Developerzy wspólnie określają, czy dany element PBI jest wystarczająco mały, aby zmieścił się w pojemności danego Sprintu. Jeśli Developerzy szczerze wierzą, że mogą dostarczyć dany element w tym Sprincie – zgodnie z Definicją Ukończenia – dany element jest dodawany do prognozy. Może to wymagać szacowania lub użycia miary przepływu,

takiej jak Service Level Expectation (oczekiwany poziom usługi). Przepływ i miary przepływu omówię w rozdziale 9.

Jeśli Developerzy nie są zgodni, czy element należy uwzględnić w prognozie, czy nie, może to wymagać dalszej analizy i omówienia danego elementu PBI. Ostatecznie element może zostać podzielony lub odłożony do późniejszego Sprintu, gdy więcej będzie wiadomo na jego temat. Developerzy następnie przechodzą do kolejnego elementu w Product Backlogu. Ponieważ Product Owner uczestniczy w Sprint Planningu, kolejność elementów w Product Backlogu może być w razie potrzeby aktualizowana.

Ten proces prognozowania jest powtarzany, aż Developerzy uznają, że mają odpowiednią ilość pracy na dany Sprint, biorąc pod uwagę swoje możliwości, dostępność, przeszłą wydajność i inne czynniki. Ujęte w prognozie elementy PBI są przenoszone z Product Backlogu do Sprint Backlogu.

Nowe Scrum Teamy, które nie mają jeszcze pojęcia o swojej przeszłej wydajności, mogą skorzystać z intuicji, aby zdecydować, jaka ilość pracy *wydaje się* odpowiednia. Wysokowydajne zespoły Professional Scrum Team mogą postępować podobnie. Jeśli podczas Sprintu Developerzy wcześnie ukończą pracę nad swoimi prognozami, powinni współpracować z Product Ownerem w środku Sprintu nad znalezieniem dodatkowych elementów PBI do realizacji. Najlepiej, aby te elementy również były zgodne z Celem Sprintu. Developerzy nigdy nie powinni prognozować większej ilości pracy niż *wiedzą*, że mogą ukończyć.

Uwaga W roku 2011 *Przewodnik po Scrumie* wprowadził dość kontrowersyjną zmianę do Sprint Planningu. Słowo „zobowiązać" zostało zastąpione słowem „prognozować". Praktycy Scruma mieli od jakiegoś czasu problemy ze słowem *„zobowiązać"*. Problem w tym, że zakładało ono obowiązek dostarczenia przez zespół wszystkich elementów PBI na koniec Sprintu. Było to szczególnie istotne, gdy interesariusze nierozumiejący złożoności rozwijania skomplikowanych produktów, takich jak oprogramowanie, słyszeli to słowo. Ponieważ opracowywanie złożonego produktu jest trudne i pełne ryzyka, dostarczenie wszystkich elementów PBI w każdym Sprincie jest mało realistyczne. Developerzy mogli być zmuszani do obniżania jakości, aby dotrzymać swoich „zobowiązań", a to jest zabronione w Scrumie oraz w etycznym zarządzaniu produktem.

Termin „prognozowanie" jest bardziej realistyczny i łatwiejszy do zrozumienia przez interesariuszy, którzy zwykle słyszeli na przykład o „prognozowaniu sprzedaży". Sugeruje to, że chociaż Developerzy będą się starać, to mogą pojawić się nowe informacje podczas Sprintu, które mogą utrudniać realizację założonych planów. Jeśli organizacja jest przywiązana do terminu „zobowiązanie", to przestawienie się na termin „prognozowanie" może zająć nieco czasu.

Temat 3: Jak (w jaki sposób) zostanie wykonana wybrana praca? Sprint Planning nie kończy się, dopóki Developerzy nie opracują planu, w jaki sposób opracują