Scrum nie jest metodologią – jest drogą.

Ken Schwaber (Boulder, Co, Listopad 2005)

5.1. Pochodzenie i podstawowe mechanizmy

„W dzisiejszym pędzącym, niezwykle konkurencyjnym świecie komercyjnego wytwarzania nowych produktów, szybkość i elastyczność są najistotniejsze. Firmy coraz częściej zdają sobie sprawę, że stare, sekwencyjne podejście do wytwarzania nowych produktów po prostu nie przynosi efektów. Zamiast tego firmy w Japonii i Stanach Zjednoczonych używają metody holistycznej – tak jak w rugby, gdzie piłka jest ciągle podawana w zespole, kiedy ten pokonuje pole gry jako całość."

Nie, Drogi Czytelniku, to nie jest fragment nowej strategii jednej z przodujących korporacji, która powstała podczas warsztatów innowacyjności w trakcie dwudziestoczterogodzinnej medytacji do góry nogami na kole podbiegunowym. Te pomysły nie są niczym nowym ani odkrywczym. Jest to fragment artykułu „The new new product development game" autorstwa Japończyków Hirotaka Takeuchi i Ikujiro Nonaka, który został opublikowany na łamach Harvard Business Review, w numerze Styczeń-Luty 1986, czyli ponad dwadzieścia siedem lat temu.

W przeciwieństwie do tradycyjnych metod wytwarzania oprogramowania Scrum nie jest wynikiem rozważań teoretycznych i przewidywania wszystkich możliwych sytuacji. Nie jest on też instrukcją, która zawiera odpowiedzi na pytanie „jak dostarczyć idealny projekt?" lub „mam problem z X, co mam teraz zrobić?". Scrum jest wypadkową najlepiej działających i sprawdzonych metod przynoszących efekty. Jest to sposób prowadzenia projektu nastawiony na efekty i panowanie nad zmianą, w przeciwieństwie do metod promujących aktywności i sztywne trzymanie się oryginalnych planów.

Scrum od początku wydawał mi się zupełnie naturalny, prosty i praktyczny, więc zacząłem go stosować i zainteresowałem się jego źródłami zgodnie z zasadą Shu-Ha-Ri, która zostanie przedstawiona w rozdziale 6. W tym rozdziale przedstawię te źródła, wyjaśnię również mechanizmy stojące za skutecznością Scrum oraz powody, dla których każda z praktyk jest ważna dla działania tego frameworku.

5.1.1. Nowa gra wytwarzania nowych produktów

W 1993 roku artykuł „The new new product development game" trafił w ręce Jeffa Sutherlanda razem z rezultatami projektu Pasteur Project w ATT Bell Labs oraz praktykami wypracowanymi przez Toyota (Toyota Production System). To właśnie on opracował podstawy Scrum, równocześnie nadając nowej metodzie nazwę.

Do zilustrowania różnic w procesach wytwarzania produktów w artykule użyto metafory sportowej. To tutaj po raz pierwszy usystematyzowane podejście do wytwarzania (w odniesieniu do oprogramowania jest to model Waterfall) porównano do biegu sztafetowego, w którym produkt jest przekazywany z rąk jednej grupy ekspertów do rąk kolejnej. Z kolei nowe podejście, znacznie bardziej skuteczne zwłaszcza przy wytwarzaniu nowych produktów, porównane zostało do gry w rugby, podczas której zespół, podając piłkę do przodu i do tyłu między członkami drużyny, musi przejść całe pole wspólnie, jako jednostka, żeby osiągnąć sukces. Po przerwaniu gry, podczas jej ponownego rozpoczęcia tworzona jest formacja, w której członkowie zespołów stają naprzeciw siebie zwróceni twarzami w stronę przeciwnika i trzymając się w zwartej grupie, prą do przodu. Ta formacja nazywa się Scrum i stąd właśnie pochodzi nazwa frameworku.

Wnioski opublikowane w artykule zostały wyciągnięte na podstawie badań wykonanych podczas tworzenia nowych produktów, które nie były nawet produktami programowymi.

Projekty służące jako obiekt badań to:
- średniej wielkości kopiarka FX-3500 wprowadzona przez Fuji-Xerox w 1978;
- kopiarka do użytku osobistego PC-10 (Canon, 1982);
- samochód Honda City z silnikiem 1200 cc (Honda, 1981);
- komputer osobisty PC 8000 (NEC,1979);
- lustrzanka AE-1 (Canon, 1976);
- aparat kompaktowy Auto Boy, znany w Stanach Zjednoczonych jako Sure Shot (Canon, 1979).

Każdy produkt został wybrany ze względu na jego znaczenie, postrzeganie przez wytwórcę jako przełomowy w procesie rozwoju, nowoczesność funkcjonalności produktu w swoim czasie, sukces, jaki produkt odniósł na rynku, i dostęp do dokumentacji.

W wyniku badań wyznaczono sześć charakterystycznych elementów procesu wytwarzania nowych produktów, które osobno nie dają wymaganej szybkości i elastyczności, ale wspólnie stanowią receptę na uzyskanie przewagi nad konkurencją. Oto one:
1. Wbudowana niestabilność.
2. Samoorganizujące się zespoły projektowe.
3. Nachodzące na siebie fazy procesu wytwarzania.
4. Wielopłaszczyznowe uczenie.
5. Subtelna kontrola.
6. Transfer wiedzy w organizacji.

Wbudowana niestabilność

Kadra zarządzająca firmy wyznacza zespołowi ambitny cel sformułowany w postaci luźnej wizji, na przykład „w ciągu dwóch lat zbudować kopiarkę, której produkcja będzie o połowę tańsza niż wyprodukowanie modelu z najwyższej półki, a będzie działała tak samo dobrze" lub „stworzyć samochód, którym będą chcieli jeździć młodzi ludzie". Zespół dostaje zupełną swobodę odnośnie do sposobu wykonania projektu. Angielskie określnie tego zjawiska to *Built-in instability*.

Samoorganizujące się zespoły projektowe (ang. *Self-organizing project teams*)

Pozostawiony sam sobie zespół nieposiadający doświadczenia, które może wykorzystać, po pewnym czasie zaczyna działać podobnie do startupu i tworzyć własną agendę. Zespół przejmuje inicjatywę, podejmuje ryzyko i uczy się na błędach.

Kadra kierownicza ogranicza swoją aktywność do oferowania wskazówek, pieniędzy i wsparcia. Honda, organizując swój zespół, zbudowała go z członków, których średnia wieku wynosiła 27 lat, bo przecież mieli zbudować samochód dla młodych ludzi. Chociaż kuszące było stworzenie kolejnej wersji istniejącego na rynku modelu Civic, zespół postanowił postąpić na przekór normom przyjętym wtedy w branży i zamiast długiego, niskiego samochodu stworzył pojazd krótki i wysoki. Z kolei grupa, która budowała PC 8000 dla NECa, nie miała pojęcia o komputerach osobistych, byli to inżynierowie, którzy sprzedawali mikroprocesory. Wyniki takiego podejścia były zdumiewające. Zespół Canona stwierdził, że było to naprawdę trudne wyzwanie, ponieważ jego członkowie musieli zakwestionować swój tradycyjny sposób myślenia. Inżynierowie zredukowali liczbę części w aparacie o 30–40 procent i zmodularyzowali produkcję.

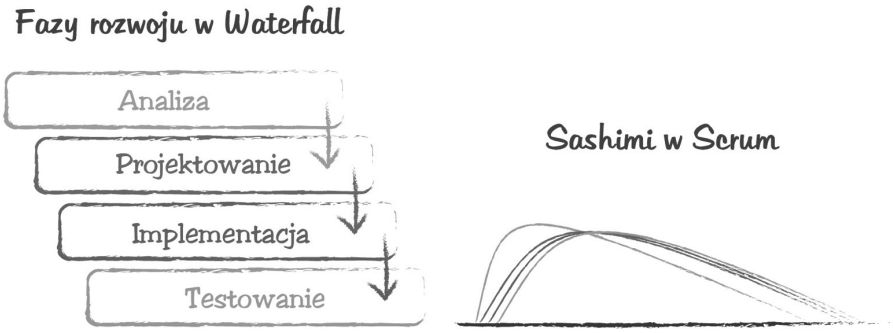
Starannie wyselekcjonowane zespoły składały się z przedstawicieli różnych działów firmy, na przykład badań i rozwoju, produkcji, sprzedaży, ale też z ludzi o różnych charakterach. W ten sposób w zespole pojawiały się pomysły wynikające z różnych punktów widzenia. Fuji-Xerox umieściło zespół w jednym, dużym pokoju, żeby wymusić obieg informacji i stworzyć jak najwięcej okazji do rozpoczynania dyskusji.

Czynniki pozwalające na działanie samoorganizujących się zespołów zostały określone jako: **autonomia**, **samo-transcendencja** (wychodzenie poza dotychczasowe ramy poznawcze) i **samozapłon** (generowanie nowych pomysłów wewnątrz grupy).

Nachodzące fazy procesu wytwarzania

Oryginalne określenie tej charakterystyki to *Overlapping development phases*. Zamiast tradycyjnego procesu fazowego, w którym najwięcej problemów pojawia się podczas przekazywania produktu z jednej fazy do drugiej, wykorzystywano coś, co przez

firmę Fuji-Xerox zostało nazwane systemem sashimi. Sashimi to japońska potrawa z ryb i owoców morza podawana w taki sposób, że kawałki ryby nachodzą na siebie. Innowacyjne zespoły składały się z członków o różnej specjalności pracujących wspólnie, czego skutkiem ubocznym było nałożenie faz procesu tworzenia produktu. Została też stworzona „wspólna wizja pracy", zatem członkowie zespołów czuli się współodpowiedzialni za wykonanie zadania. Specjalizacje i role zostały rozmyte, zeszły na drugi plan.



Rysunek 16. Sashimi - nachodzące fazy tworzenia oprogramowania

Uczenie wielopłaszczyznowe

Korzystając z zewnętrznych źródeł informacji, zespoły stale się uczą, dzięki czemu mogą szybko reagować na zmieniające się warunki. Mieszanina specjalizacji powoduje też, że zespół jest w stanie rozwiązać bardzo różnorodne problemy. Zespół szybko uczy się przez serie prób i błędów, szybko ograniczając liczbę możliwych, działających rozwiązań. Ponieważ uczenie zachodzi na wielu poziomach (**indywidualnym, grupowym, organizacyjnym**) oraz w wielu dziedzinach wiedzy, ten proces został określony jako uczenie wielopłaszczyznowe (ang. *Multilearning*).

Subtelna kontrola

Ścisła kontrola zabija spontaniczność i kreatywność. W jaki sposób można zatem wprowadzić kontrolę wystarczającą do tego, aby nie dopuścić do przeistoczenia projektu w chaos? Odpowiedzią jest skorzystanie z połączenia „samokontroli", „kontroli przez presję koleżeńską" i „kontroli przez miłość".

Wyróżniono siedem sposobów realizowania subtelnej kontroli (ang. *Subtle control*):

1. Wybranie odpowiednich ludzi do projektu i reagowanie na zmiany w dynamice grupy przez dodawanie lub usuwanie jej członków.
2. Stworzenie otwartego środowiska pracy.
3. Zachęcanie inżynierów do rozmów z klientami i sprzedawcami.
4. Ustanowienie procesu ewaluacji i nagradzania zależnego od wyników grupy.
5. Zarządzanie różnicami w rytmie pracy w trakcie procesu wytwarzania.
6. Tolerowanie pomyłek.
7. Zachęcanie dostawców (współpracowników) do samoorganizacji. Zaangażowanie ich we wczesnej fazie projektowania.

Wszystkie wymienione praktyki powinny zostać wdrożone w każdym zespole pracującym w środowisku metod zwinnych, nie tylko Scrum.

Transfer wiedzy w organizacji

Wspomniano już potrzebę przekazywaniu wiedzy między różnymi poziomami i funkcjami w zespole. Co ciekawe, zauważono równie silny nurt przekazywania wiedzy na zewnątrz. Regularnie występował transfer wiedzy do nowych projektów oraz innych oddziałów firmy. W kilku firmach wiedzę transferowano w sposób, który przez Hirotaka Takeuchi i Ikujiro Nonaka został określony mianem osmozy. Praktyka osmozy polega na przesuwaniu kilku najważniejszych pracowników między projektami. Firma zdaje sobie sprawę z tego, kto posiada większe umiejętności niż standardowe i przenosi te osoby do kolejnych projektów natychmiast po ich rozpoczęciu i obraniu dobrego kierunku. Kilka najważniejszych osób pozostaje, a reszta zespołu jest wymieniana, co w naturalny sposób sprzyja transferowi wiedzy w organizacji.

Pomimo powszechnych w firmach praktyk gromadzenia wiedzy i doświadczenia projektowego (ang. *lessons learned*), firmy zauważyły także potrzebę oduczania się. Czasem łatwiej jest zacząć od zera i zapomnieć to, co do tej pory uznawało się za pewnik. Wiedza i przekonania mogą ograniczać dalszy rozwój.

5.1.2. Mechanizmy Scrum

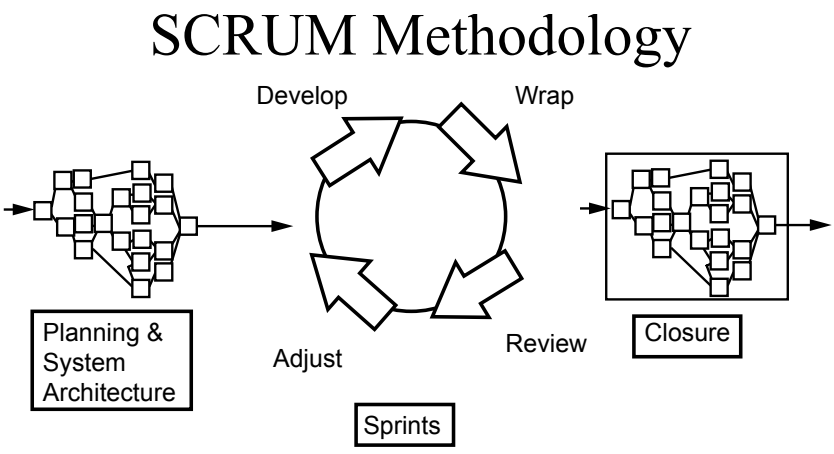
Zanim Jeff Sutherland zajął się rozwojem oprogramowania, badał złożone systemy adaptacyjne. Pracując przy pierwszym projekcie, zauważył, że oprogramowanie było zawsze dostarczane zbyt późno, programiści byli pod ciągła presją, a im bardziej kierownictwo naciskało, tym projekt bardziej się opóźniał. Nie byłoby w tym niczego odkrywczego, gdyby uwagi Jeffa nie przykuł proces stosowany w rozwoju oprogramowania, który stanowił źródło problemów. Czytając artykuły i książki Jeffa lub słuchając jego wypowiedzi, zauważymy, że często odnosi się do nieaktualnych wykresów Gantta i wypomina praktykę skupiania się na aktywnościach zamiast na wynikach pracy zespołu. Z lektury poprzednich rozdziałów już wiesz, że twórcy innych metod mieli podobne poglądy.

Scrum został opracowany w odpowiedzi na pytanie: „Czy można stworzyć hiperwydajny zespół, który dostarczałby oprogramowanie pięć do dziesięciu razy szybciej niż zwykle?”. Jeff Sutherland doskonale i dobitnie ujął sens powstania tej metody, wyrażając wizję w następujący sposób: „Celem Scrum jest osiągnąć «efekt Toyoty», dostarczając czterokrotnie więcej oprogramowania, dwunastokrotnie lepszej jakości, w szeregu krótkich ram czasowych nazwanych «Sprintami», które trwają 30 dni lub krócej”.

Drugim pytaniem, które wymagało odpowiedzi, było: „Jaki proces może skutecznie wspierać szybkie reagowanie na zmiany i dostarczanie oprogramowania, kiedy wymagania nie są pewne, a technologia dla zespołu jest nowa?”. Z góry określony,

predefiniowany proces nie będzie w tym przypadku działał, trzeba więc skorzystać z procesu empirycznego, opartego na bieżącym doświadczeniu i pętli **Inspect & Adapt**, czyli Sprawdź i Dostosuj. Dane na wejściu i na wyjściu procesu powinny być dobrze zdefiniowane, natomiast sposób wykonywania pracy będzie kształtowany w jej trakcie, w miarę, jak Zespół będzie się uczył i pozyskiwał nowe informacje na temat produktu i technologii. Zaczynamy z Wizją Produktu i celem iteracji, tworzymy plan, rozpoczynamy pracę i jak najczęściej sprawdzamy, czy podążamy w kierunku celu. Jeżeli tak nie jest, reagujemy i dostosowujemy plan do obecnych warunków. Ten mechanizm został opisany przez Deminga jako PDCA, czyli Plan-Do-Check-Act, który został omówiony w rozdziale 4.

Jak później napisze Ken Schwaber, „SCRUM zakłada, że proces wytwarzania systemów jest nieprzewidywalnym, złożonym mechanizmem, który może być tylko z grubsza opisany jako ogólny postęp".



Rysunek 17. Pierwszy diagram Scrum ©Ken Schwaber

Zespół

Wspomniane już prace w Bell Labs pokazały, że zespoły są wydajniejsze, kiedy mają mniej ról. Spytasz, jak bardzo były wydajne? Lider zespołu Quattro for Windows zapisał: „Kodowanie było wykonywane przez nie więcej niż ośmiu ludzi w tym samym czasie, co oznacza, że indywidualna produktywność kodowania była większa niż tysiąc wierszy kodu na osobo-tydzień". Zatem powstało rozróżnienie jedynie na członków i lidera zespołu. Pod wpływem artykułu „The new new product development game" lider zespołu został nazwany **Scrum Master**. Potrzebna była też osoba odpowiedzialna za tworzenie listy rzeczy do zrobienia, czyli **Rejestru Produktu** (ang. *Product Backlog*), i utrzymywanie jej w odpowiedniej kolejności, aby projekt, trafiając na rynek, jak najlepiej odpowiadał potrzebie klientów. Ta osoba została nazwana **Właścicielem Produktu** (ang. *Product Owner*). Mamy więc trzy role – Scrum Master, Właściciel Produktu i członek Zespołu. Rola Menedżera Projektu została rozproszona pomię-

dzy Scrum Mastera, Właściciela Produktu i Zespół. Komunikacja w Zespole i poza nim stała się otwarta, bez hierarchii i ścieżek eskalacji. Takie rozwiązanie znacznie przyspiesza pracę i usuwa niepotrzebne przekłamania informacji, likwidując efekt „głuchego telefonu".

Spotkania

Następnie uwaga została skupiona na spotkaniach. Nie można szybko budować oprogramowania, jeżeli Zespół ciągle uczestniczy w spotkaniach. Jakie spotkania są więc potrzebne i niemożliwe do wyeliminowania? Zespół musiał dostarczać produkt w cyklach nie dłuższych niż cztery tygodnie, które zostały określone jako **Sprint**. Potrzebne było spotkanie, w którego trakcie Zespół wybierze element, nad którym będzie pracował podczas kolejnego cyklu, i sposób, w jaki go zaimplementuje. I tak powstało **Planowanie Sprintu** (ang. *Sprint Planning*) składające się z dwóch części. Pierwsza część obejmuje wybór elementu do osiągnięcia w kolejnej iteracji, a druga – stworzenie planu jego wykonania.

Wysoce wydajne zespoły w Bell Labs były napędzane przez codzienne spotkania, zatem rytuał ten wprowadzono również do frameworku Scrum. „Projekt zyskiwał na swoim małym rozmiarze przez skupianie aktywności programistycznych wokół codziennych spotkań, podczas których architektura, projekt i interfejs były przekazywane w grupie". Czas trwania tego spotkania ograniczono do piętnastu minut i nadano mu nazwę **Codzienny Scrum** (ang. *Daily Scrum*). W jego trakcie każdy członek zespołu powinien dowiedzieć się, co zostało już wykonane, czym będzie się zajmował i jak zespół może sobie pomóc w szybkim wykonaniu pracy. Jest to wykorzystanie pętli **Sprawdź i Dostosuj** do tworzenia planu na kolejny dzień.

Ze swojej poprzedniej firmy Jeff zapożyczył podejście „demo or die", czyli dążenie za wszelką cenę do pokazania działającego oprogramowania na koniec iteracji. Zatem na koniec każdej iteracji potrzebne było spotkanie, podczas którego oprogramowanie jest prezentowane ludziom, którzy będą z niego korzystali, w celu otrzymania informacji zwrotnej. Cykl informacji zwrotnej jest kolejnym z czynników powodujących szybsze wytwarzanie oprogramowania. Spotkanie podsumowujące Sprint jest dziś określane nazwą **Przegląd Sprintu** (ang. *Sprint Review*). Nie ma ono na celu jedynie prezentacji wersji demonstracyjnej produktu, ale także inspekcję rozwoju produktu i określenie dalszych kroków. Dokładnie rzecz ujmując, jest to pętla **Inspect & Adapt** zastosowana do produktu.

Raportowanie

Kolejnym elementem wymagającym weryfikacji było raportowanie. Projekty informatyczne skutkowały wyprodukowaniem dużej liczby raportów, głównie w postaci wykresów Gantta. Zwykle jednak raporty były nieaktualne, co utrudniało planowanie i ustalenie statusu projektu. Jeff Sutherland, nawiązując do swojego doświadczenia lotniczego, zauważył, że dostarczenie produktu w trakcie iteracji jest podobne do lądowania samolotu, ponieważ należy śledzić wysokość względem ziemi i prędkość,

z jaką się do niej zbliżamy. Metafora ta doprowadziła do powstania **Wykresów Burndown** (ang. *Burndown Charts*). Ponieważ w myśl minimalizowania strat zgodnie z **Lean** Scrum powinien być samoraportujący, wykres jest umieszczany na ścianie. Na tej samej ścianie ostatecznie umieszczono Rejestr Produktu, wykres Burndown i zadania w kolumnach statusu, w jakim obecnie się znajdują, na przykład: zadania do wykonania, w trakcie i wykonane. Taki sposób śledzenia postępu i raportowania w jednym określamy mianem **Agile Wall**.

Jeff Sutherland przeprowadził pierwsze wdrożenie Scrum w firmie Easel Corporation w 1993 roku. Jak wspomina, ciężko było przekonać prezesa firmy do wprowadzenia nowej metody pracy, zwłaszcza że zmiana dotyczyła najważniejszego projektu. Nie było podobnych przypadków referencyjnych ani dokumentacji wcześniejszych doświadczeń pracy tą metodą. Możesz sobie wyobrazić taką rozmowę z własnym szefem.

Zespół osiągnął stan hiperproduktywny, ale nie było to jedynie zasługą wprowadzenia wzorca organizacyjnego Scrum. Jednocześnie wykonywano testy modułowe, integrację pakietów, refactoring wybranych części systemu i wiele kompilacji (buildów) dziennie. Czytając wcześniejsze rozdziały, wiesz, że te aktywności stały się najważniejszymi elementami Programowania Ekstremalnego. Niewiele osób wie, że Scrum był pierwszy.

5.1.3. Rozwój Scrum i Organizacje Certyfikujące

Omawiając Scrum, należy oczywiście wspomnieć o różnych certyfikatach, których posiadanie jest często wymagane w ofertach pracy dla członków Zespołów Scrum. Obecnie sytuacja nieco się skomplikowała, ponieważ istnieją dwie oficjalne organizacje certyfikujące – Scrum Alliance i scrum.org. Jest też kilka mniejszych, wymyślających własne certyfikaty, ale nie są one oficjalnie uznawane ani akceptowane przez pracodawców.

Zacznijmy od kilku ważnych dat. W 1995 roku Ken Schwaber został poproszony przez Jeffa Sutherlanda o przygotowanie opisu Scrum na potrzeby warsztatu odbywającego się podczas konferencji Object-Oriented Programming, Systems, Languages & Applications. Jego opracowanie z późniejszymi modyfikacjami nadal stanowi źródło teorii tego frameworku. W 2001 roku odbyło się pierwsze oficjalne szkolenie Scrum pod nazwą Certified Scrum Master. Trzy lata później Ken Schwaber, Esther Derby i Mike Cohn stworzyli Scrum Alliance. W 2006 roku została opublikowana kultowa, mała, czarna książka „Software development with Scrum" napisana przez Kena Schwabera i Mike'a Beedle'a, która stała się podstawowym źródłem wiedzy na temat Scrum. W 2009 roku Ken Schwaber odszedł ze Scrum Alliance i stworzył scrum.org. Powody jego decyzji i różnice między organizacjami można znaleźć na stronie https://www.scrum.org/About/Origins. Scrum nadal jest rozwijany, a jego opis jest publikowany na stronie scrum.org pod nazwą Scrum Guide. W chwili pisania tej książki najnowsza wersja frameworku pochodzi z lipca 2013 roku i właśnie do niej odnoszę się w tej książce.

Obie organizacje oferują podobne trzy stopnie certyfikatów, ale mają inne wymagania odnośnie do sposobu otrzymania takiego dokumentu. Zestawienie odpowiedników certyfikatów znajduje się w tabeli 2.

Tabela 2. Zestawienie odpowiedników certyfikatów przyznawanych przez Scrum Alliance i scrum.org

Scrum Alliance	**scrum.org**
Certyfikaty pierwszego poziomu	
Certified Scrum Master (CSM)	Professional Scrum Master I (PSM I)
Certified Scrum Product Owner (CSPO)	Professional Scrum Product Owner I (PSPO I)
Certified Scrum Developer (CSD)	Professional Scrum Developer I (PSD I)
Certyfikaty drugiego poziomu	
Certified Scrum Professional (CSP)	Professional Scrum Master II (PSM II) Professional Scrum Product Owner II (PSPO II)
Certyfikaty trzeciego poziomu	
Certified Scrum Trainer (CST)	Professional Scrum Trainer (PST)
Certified Scrum Coach (CSC)	

Różnica w certyfikacji widoczna jest już na pierwszym poziomie. W Scrum Alliance głównym wymaganiem jest odbycie odpowiedniego szkolenia prowadzonego przez certyfikowanego trenera. Wprowadzony niedawno, prosty egzamin online sprawdza naprawdę podstawową wiedzę. W scrum.org wystarczy zdać egzamin online weryfikujący wiedzę. Jest on trudniejszy od egzaminu Scrum Alliance, a do otrzymania certyfikatu wymagany jest poziom 85% dobrych odpowiedzi, a jeżeli kandydat myśli o ścieżce trenera, poprzeczka zostaje podniesiona do 95%. Do otrzymania kolejnych poziomów certyfikacji wymagane jest posiadanie certyfikatu pierwszego stopnia i zdanie kolejnego egzaminu. W obu organizacjach proces wspięcia się na poziom trenerski jest dosyć trudny i kosztowny.

Ponieważ zasady i stawki ulegają zmianom, po więcej szczegółowych informacji odsyłam do stron scrumalliance.org i scrum.org.

5.2. Framework Scrum

Scrum jest frameworkiem, w ramach którego można rozwiązywać złożone problemy adaptacyjne, jednocześnie produktywnie i kreatywnie dostarczając produkty o możliwie największej wartości.

Definicja Scrum jasno określa, że nie jest to metoda ani metodologia.