Okładka

Doskonalenie zaawansowanego Scruma

Zaawansowane techniki dla zespołów Scruma, ról, artefaktów, zdarzeń, metryk, porozumień roboczych, zaawansowanych praktyk inżynierskich i zwinności technicznej

Rituraj Patil

Doskonalenie zaawansowanego Scruma

Zaawansowane techniki dla zespołów Scruma, ról, artefaktów, zdarzeń, metryk, porozumień roboczych, zaawansowanych praktyk inżynierskich i zwinności technicznej

Rituraj Patil

APN Promise 2024

Doskonalenie zaawansowanego Scruma

Authorized translation from the English language edition, entitled „Mastering Advanced Scrum" by Rituraj Patil, published by BPB Publications, ISBN: 978-93-91030-308

Polish language edition published by APN PROMISE SA

Autoryzowany przekład z wydania w języku angielskim, zatytułowanego „Web Data Mining with Python" by Dr. Ranjana Rajnish & Dr. Meenakshi Srivastava, opublikowanego przez BPB Publications, ISBN: 978-93-5551-363-2

APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: wydawnictwo@promise.pl

ISBN: 978-83-7541-532-2 (druk), 978-83-7541-533-9 (ebook)

Przekład: Mariusz Rogulski i Witold Sikorski
Fotografia na okładce: © Tomasz Niesłuchowski
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

[Książkę] **dedykuję**

Chhatrapatiemu Shivaji Maharajowi,
wszystkim moim guru, nauczycielom i mentorom
oraz tym wszystkim,
którzy robią najwięcej dla innych…

Spis treści

O autorze

Rituraj Patil jest doświadczonym Scrum Masterem i trenerem Agile, który przez ponad 10 lat pracował w różnych wielonarodowych organizacjach, tworząc i dostarczając oprogramowanie, usługi i rozwiązania wysokiej jakości. Jest entuzjastą techniki lean agile (przywództwa, kultury i zarządzania), przywódcą służebnym i zapalonym uczniem. W swojej karierze zawodowej odgrywał wiele ról, działając w różnych organizacjach. Na uniwersytecie w Bombaju (Maharashtra, Indie) uzyskał stopień magistra zarządzania ze specjalizacją w systemach. Ma także stopień magistra nauk technicznych w zakresie rozwoju i zarządzania oprogramowaniem uzyskany w Vellore Institute of Technology (Tamilandu, Indie) oraz stopień inżyniera nauk technicznych w dziedzinie technologii informatycznych uzyskany w Kolhapur Institute of Technology's College of Engineering (Maharashtra, Indie).

Z powodzeniem ukończył kilka uznanych na całym świecie szkoleń i certyfikacji. Występuje jako prelegent na różnych i spotkaniach konferencjach branżowych, a także jako gościnny wykładowca w różnych czołowych indyjskich instytucjach edukacyjnych, w tym IMM, Symbiosis, IIT, VIT Velore i wielu innych. Otrzymał wiele nagród, wyróżnień i wyrazów uznania od wielu organizacji, z którymi dotąd współpracował.

Ma ogromne doświadczenie w zarządzaniu wieloma globalnie dostępnymi i kolokowanymi zespołami programistycznymi Agile opracowującymi aplikacje wykorzystujące różne ramy postępowania, narzędzia i techniki agile. Pracuje też jako ekspert branżowy w radzie studiów Uniwersytetu Sanjay Ghodawat (Department of Computer Science Engineering na Uniwersytecie Sanjay Ghodawat, Maharashtra, Indie) oraz na uczelni technicznej w Kolaphur (Kolhapur Institute of Technology's College of Engineering (Autonomous), Maharashtra, Indie).

Poza pracą wolny czas poświęca różnym działaniom jako woluntariusz dla CSR (*Corporate Social Responsibility*). Jest też aktywnym członkiem wielu globalnych społeczności pracujących na rzecz rozwoju różnych technik, przywództwa, edukacji, zrównoważonego rozwoju i inicjatyw związanych z CSR. Lubi grać na tabli (klasycznym instrumencie indyjskim), śpiewa karaoke, gra w piłkę nożną i poświęca wolny czas na pomoc i udzielanie wskazówek początkującym specjalistom IT.

O recenzencie

Amit Mahulikar jest praktykiem agile i od roku 2012 pracuje nad transformacjami agile. Pomógł kilku bankom, instytucjom finansowym i domom sprzedażowym w przyjmowaniu zwinności i różnych ram postępowania, takich jak Scrum, XP i SAFe (Scaled Agile). Odgrywając rolę agenta zmian agile (Agile Change Agent,) ściśle współpracował z organizacjami klienckimi z siedzibą poza Indiami. Będąc absolwentem MBA, jest także konsultantem programowym SAFe (SPC 5.0) i pracuje na stanowisku starszego menedżera w MCN z siedzibą w Pune.

Podziękowania

Jest kilka osób, którym chciałbym podziękować za ciągłe i nieustające wsparcie, które od nich otrzymywałem podczas pisania tej książki. Przede wszystkim chciałbym podziękować mojej ukochanej matce – Aai oraz mojej lepszej połowie – Shwetambari za tolerowanie mnie, gdy byłem bardzo zajęty i spędzałem wiele weekendów i wieczorów nad tą książką. Nigdy bym jej nie skończył bez ich wsparcia, motywacji i zrozumienia z ich strony.

Książka ta nie powstałaby, gdybym nie otrzymał wsparcia i wskazówek od kilku kluczowych praktyków agile i Scruma, należących do globalnej społeczności zwinnego tworzenia oprogramowania. Moja wdzięczność należy się wszystkim praktykom, agentom zmian, przywódców służebnych, moim nauczycielom, przewodnikom, mentorom, liderom, kolegom, przyjaciołom, krewnym, zwolennikom i wszystkim organizacjom, z którymi do tej pory pracowałem i którym służyłem, za wspaniałe możliwości, pomoc, wsparcie, porady, zachęty i motywacje, jakie zawsze od nich otrzymywałem.

Chciałbym też szczególnie podziękować Amitowi Mahulikarowi i Priyance Deshpande za zrecenzowanie tej książki i przekazanie niezbędnych uwag, aby ją poprawić. Na koniec chciałbym podziękować BPB Publications za danie mi tej niezwykłej szansy na napisanie dla nich pierwszej książki w mojej karierze.

Wprowadzenie

Zwinny rozwój i dostarczanie oprogramowania (Agile Software Product Development and Delivery, ASPDD) to podejście, które zawsze powinno być realizowane w środowiskach opartych na współpracy, wspólnym działaniu i złożonych środowiskach adaptacyjnych. Zapewnia dostarczenie oprogramowania wysokiej jakości w różnym zakresie, przy efektywnych kosztach i zwinnym iteracyjnym sposobie pracy w odcinkach czasu, aby spełnić wciąż zmieniające się wymagania biznesu, klientów i interesariuszy. Scrum jest jedną z najpopularniejszych i najmocniejszych zwinnych ram postępowania służącym temu celowi. Stanowi uporządkowany zbiór cennych pojęć, metod i praktyk, wokół których muszą być budowane i modyfikowane procesy w celu zwiększenia sposobu zdolności zespołów deweloperskich do reagowania na zmiany. Zachęca też zespoły do zwinnego tworzenia i dostarczania oprogramowania w sposób iteracyjny, przyrostowy, szybki i częsty.

Scrum Guide (opracowany przez Kena Schwabera i Jeffa Sutherlanda, twórców Scruma) zawiera definicję ram postępowania Scrum. Jest tam też przegląd ról, artefaktów, zdarzeń i reguł Scruma, ale nie ma tam wystarczająco wielu szczegółów, aby wykorzystać zalety w pełni funkcjonującego zwinnego sposobu pracy opartego na Scrumie. Praktycy i zespoły Scruma muszą wykorzystywać dodatkowe rozszerzenia i techniki, które nie są opisane w ich przewodniku.

Te rozszerzenia są zwykle związane z ogólną strukturą, współpracą i komunikacją w ramach zespołów Scruma, rolami Scruma i porozumieniami roboczymi w ramach zespołów Scruma, efektywnym i wydajnym zarządzaniem rejestrem produktu, szacowaniem względnym, zdarzenia Scruma, zaawansowanymi praktykami inżynierskimi i zwinnością techniczną, metrykami Scruma, skalowaniem Scruma i kilkoma innymi dodatkowymi aspektami zwinnej metody działania zespołu Scruma.

W tej książce staram się położyć nacisk na takie rozszerzenia i techniki, które powinny być efektywnie wykorzystywane przez praktyków i zespoły Scruma, aby ustanowić i poprawić w pełni funkcjonalny zwinny sposób pracy oparty na Scrumie. Książka ta nie tylko pomaga zachęcić ich do odpowiedzialności, rozliczania się i własności, lecz także prowadzi czytelników tak, aby stali się samozarządzający, samoorganizujący się i osiągnęli wysoką wydajność.

Kluczową korzyścią z przeczytania tej książki jest osiągnięcie ogólnego zrozumienia ASPDD oraz sposobu pracy zespołów rozwoju oprogramowania w celu ogólnego zrozumienia ram postępowania Scrum (stworzonych przez Kena Schwabera i Jeffa Sutherlanda) i ich powiązania z wartościami i zasadami agile (podanymi na stronie *https://agilemanifesto.org/),* rozpowszechnienie wiedzy o rozszerzeniach i technikach zaawansowanego Scruma wśród jego praktyków (dzięki czemu oni i ich zespoły Scruma mogą poczuć się wzmocnieni i będą mogli się stać bardziej wydajni, samozarządzający i samoorganizujący się) i uruchomić proces myślowy praktyków Scruma, za pomocą którego mogą ocenić przydatne i odpowiednie zaawansowane rozszerzenia i techniki Scruma właściwe dla ich zespołów Scruma.

Czytelnicy tej książki będą się czuli bardziej kompetentni do implementacji ze swoimi zespołami Scruma w pełni funkcjonalnych ram postępowania opartych na zwinnym sposobie pracy. Książka ta pomoże im zwiększyć ogólną produktywność i efektywność w ramach ich zespołów Scruma, dzięki wykorzystaniu większości korzyści z zaawansowanych rozszerzeń i technik Scruma w celu dostarczenia potencjalnie gotowych do wysyłki przyrostów swoich produktów programistycznych o wysokiej jakości.

W kolejnych 15 rozdziałach tej książki czytelnik dowie się następujących rzeczy:

- Rozdział 1 wprowadza w podstawy *zwinnego tworzenia oprogramowania, dostarczania i sposobu pracy* zespołów deweloperskich.
- Rozdział 2 zawiera ogólne porównanie *ram postepowania agile*, takich jak Scrum, Kanban, Scrumban, XP (Extreme Programming), DSDM (Dynamic Systems Development Method), FDD (Feature-Driven Development), Crystal itd.
- Rozdział 3 daje ogólny przegląd ram postępowania Scrum, najpopularniejszych i najszerzej stosowanych w całej globalnej społeczności twórców oprogramowania.
- Rozdział 4 to jeden z kluczowych rozdziałów, w którym dogłębnie omówiono wiele niepoprawnych działań, reakcji, zachowania, wzorców i jednostek pojawiających się w zespołach Scruma, z powodu których zwinny sposób pracy oparty na Scrumie zaczyna źle działać.
- Rozdział 5 wprowadza zaawansowane rozszerzenia i techniki Scruma, które są ostatecznymi rozwiązaniami pozwalającymi na ustanowienie i ulepszenie opartego na Scrumie zwinnego sposobu pracy w zespołach Scruma.
- Rozdział 6 opisuje zaawansowane rozszerzenia i techniki Scruma dla zespołów Scruma, które borykają się z zagadnieniami związanymi z ich strukturą i ustawieniem.
- Rozdział 7 opisuje zaawansowane rozszerzenia i techniki Scruma dla trzech ról Scruma – właściciela produktu, Scrum Mastera i deweloperów.

- Rozdział 8 opisuje zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w efektywnym i wydajnym zarządzaniu rejestrem produktu.
- Rozdział 9 opisuje zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w efektywnym i wydajnym szacowaniu względnym elementów ich pracy.
- Rozdział 10 opisuje zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma bardziej efektywnie i wydajnie wykorzystywać zdarzenia Scruma.
- Rozdział 11 opisuje zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w odpowiednim wykorzystaniu zaawansowanych praktyk inżynierskich do poprawy ich zwinności technicznej.
- Rozdział 12 opisuje zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w ustanowieniu i wykorzystaniu efektywnych i wydajnych metryk Scruma.
- Rozdział 13 opisuje dodatkowe zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w skalowaniu Scruma.
- Rozdział 14 opisuje dodatkowe zaawansowane rozszerzenia i techniki Scruma pomagające zespołom Scruma w ulepszaniu i usprawnianiu ich zwinnego sposobu pracy opartego na Scrumie.
- Rozdział 15 zawiera krótki opis najnowszych zaktualizowanych wersji podręcznika Scruma (tj. *Scrum Guide 2020*).

Rozdział 1

Podstawy zwinnego rozwoju oprogramowania, dostarczania i sposób pracy

Wprowadzenie

W tym rozdziale czytelnicy poznają podstawy zwinnego rozwoju oprogramowania, dostarczaniu go i sposobu pracy. Ponadto dowiedzą się, w jaki sposób zespoły deweloperskie mogą poznawać wartości i zasady agile zawarte w Manifeście Agile (*Manifesto for Agile Software Development*) i wykorzystując je, mogą ustanowić i ulepszyć swój zwinny sposób pracy.

Cel

Po przeczytaniu tego rozdziału Czytelnicy powinni:

- zrozumieć podstawy zwinnego rozwoju oprogramowania, dostarczania go i sposobu pracy zespołów deweloperskich;
- zrozumieć wartości i zasady zawarte w *Manifesto for Agile Software Development*;
- zrozumieć potrzebę cyfrowej i zwinnej transformacji w organizacjach;
- zrozumieć pojęcie zwinności organizacyjnej, zwinnego sposobu pracy i etapów rozwoju grupy.

Alex „menedżer programu" (na spotkaniu podczas rozmowy z Paulem „nowym Scrum Masterem" i pozostałymi Scrum Masterami):

Cześć! Chciałbym powitać wszystkich na dzisiejszym Scrumie Scrumów. Jest to pierwsze spotkanie tego rodzaju, na którym spotykają się wszyscy Scrum Masterzy reprezentujący swoje zespoły Scruma w ramach naszego programu.

Szczerze mówiąc, zamiast nazywania tego spotkaniem, wolałbym określić to jako zwykłą cotygodniową rozmowę, aby podzielić się opiniami, dyskutować i przeprowadzić burzę mózgów na temat wczorajszej pogody, rodzaju wyzwań, przed którymi stoimy, sposobu ich pokonania i postępów, jakie robimy jako „jeden zespół programowy".

Mam nadzieję, że wszyscy są nastawieni entuzjastycznie do tej nowej podróży w celu powiększenia skali naszego zwinnego sposobu pracy na poziomie programu. Jest to dopiero początek i z pewnością przed nami długa droga!

Paul „nowy Scrum Master" (głęboko zamyślony, mówi do siebie):

Scrum Scrumów? Cotygodniowe rozmowy? Wczorajsza pogoda? Jeden zespół programowy? Skalowanie zwinnego sposobu pracy? Co to za żargon? Czy jestem tu jedynym, który czuje się zagubiony?

A Alex kontynuuje:

Mam głębokie przekonanie, że „agile nie oznacza anarchii!". Musimy mieć pod ręką jakiś mechanizm, który zapewni, że robimy co w naszej mocy, podczas gdy „bycie zwinnym to nie tylko postępowanie agile". Musimy mieć pewność, że porażka nastąpi szybko i pozwoli bezpiecznie nauczyć się, oduczyć się i nauczyć na nowo, jednocześnie uwzględniając zmiany i stale dostarczając oczekiwane wartości naszym klientom.

Ustalmy razem, że nasze zespoły Scruma muszą się same organizować, same zarządzać, być wielofunkcyjne i mieć wysoką wydajność. Zastanówmy się, jak stać się bardziej efektywnymi i wydajnymi tak, abyśmy wszyscy współdzielili zrozumienie i współodpowiedzialność w celu ciągłego uczenia się i doskonalenia.

Sprawmy, aby nasi klienci i interesariusze byli zadowoleni! Jesteśmy tu po to, aby współpracować i współdziałać. Skupmy się więc na różnych aspektach poprawy naszego ogólnego zwinnego sposobu pracy. Stwórzmy elementy działań zorientowanych na wyniki, aby wprowadzić ulepszenia, jakie chcielibyśmy widzieć u siebie.

Paul (ponownie zastanawiając się i mówiąc do siebie):

O Boże! Przykro mi, ale nadal cię nie rozumiem, Alex! Dlaczego agile nie oznacza anarchii? Czy bycie zwinnym to nie po prostu działanie agile? Szybka porażka? Bezpieczna porażka? Nauczyć się, oduczyć się i nauczyć na nowo? Zespoły Scruma

samoorganizujące się, samozarządzające się, wielofunkcyjne i o wysokiej wydajności? Współdzielenie zrozumienia i współodpowiedzialność, aby stale uczyć się i wciąż się doskonalić? Elementy działań zorientowane na wyniki?

Po co w ogóle tego potrzebujemy? Uważam, że powinienem teraz zabrać głos! Czy na pewno?

Każdy, kto jest częścią zespołu deweloperskiego (entuzjastą agile, trenerem agile, praktykiem Scruma, Scrum Masterem, właścicielem produktu lub menedżerem produktu/projektu/programu/wydania/dostarczania lub po prostu członkiem zespołu deweloperskiego lub kimkolwiek innym, kto ma podstawową wiedzę o Scrumie) lub wykorzystuje ramy postępowania Scrum, może być uczestnikiem takich rozmów.

Tego rodzaju rozmowy mogą się od siebie różnić pod względem ról i odpowiedzialności uczestników oraz w kwestii ogólnego kontekstu i treści prowadzonej dyskusji. Mogą też różnić się pod względem poziomu organizacyjnego, na jakim się odbywają. Może to być poziom zespołu lub nawet kilka poziomów wyżej.

Słowa w żargonie brzmią tak,
jakbyśmy powiedzieli coś ważniejszego od ich znaczenia.
– Theodor Adorno

Alex, „Menedżer programu", rzuca żargonem i próbuje wyjaśnić innym Scrum Masterom, czym jest „być zwinnym, a nie tylko działać zwinnie". Jednak dla Paula, „nowego Scrum Mastera" (lub każdego innego, kto stanowi część programistycznego zespołu deweloperskiego i jest nowicjuszem ramach postępowania agile i Scrum), staje się to wielkim znakiem zapytania. Jest więc bardzo ważne, aby wszyscy członkowie zespołu deweloperskiego najpierw zrozumieli zasadnicze postawy rozwoju, dostarczania i sposobu pracy agile nad oprogramowaniem, co pozwoli im zrozumieć kolejne słowa Aleksa:

Twoje przekonania stają się myślami,
Twoje myśli stają się Twoimi słowami,
Twoje słowa stają się Twoimi czynami,
Twoje czyny stają się Twoimi zwyczajami,
Twoje zwyczaje stają się Twoimi wartościami,
A Twoje wartości stają się Twoim przeznaczeniem.
*– Mahatma Gandhi**

* Według Facebooka: *https://d.facebook.com/obudzmoc/photos/a.287912843135432/371071211486261/?type=3&source=48* (przyp. tłum.).

Zwinne tworzenie oprogramowania, dostarczanie i taki sposób pracy to niezwykłe połącznie podejścia uwzględniające zbiór zwinnych ram postępowania i związanymi z nimi procedurami, który trzeba ustanowić wykorzystując podstawowe wartości i zasady wymienione w *Manifeście zwinnego tworzenia programowania*. Te podstawowe wartości i zasady działają jako ostateczne wskazówki dla całego zwinnego sposobu pracy dla zespołów tworzących oprogramowanie, uwzględniając wszystkie realizowane przez nie procedury w celu odkrycia stosowalnych i odpowiednich sposobów analizy i oceny oraz zastosowania w ich własnym kontekście.

Przestrzegaj zasad nie będąc nimi związany.
– Bruce Lee

Manifest (*Manifesto for Agile Software Development*, dostępny pod adresem *https://agilemanifesto.org/*) rzuca światło na 4 podstawowe wartości i 12 wspomagających je zasad.

Manifest ten pomaga w zwinnym tworzeniu oprogramowania, dostarczaniu i sposobie pracy, gdzie najwyższa waga przykładana jest zawsze do dostarczenia działającego oprogramowania wysokiej jakości.

W przeszłości 17 wymienionych poniżej niezwykłych osób ze świata tworzenia oprogramowania spotkało się, aby omówić przyszłość rozwoju. Podkreślali różne problemy związane z różnymi praktykami rozwoju oprogramowania, które napotykali w tamtym okresie. Były dyskusje i sprzeczne poglądy.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Wspólny problem, który stanowił clou ich dyskusji, wiązał się z organizacjami i związanymi z nimi ludźmi, którzy zbyt dużą wagę przywiązywali do standaryzacji procesu tworzenia oprogramowania, gdzie na czele zawsze stało planowanie i dokumentowanie cyklu życia rozwoju oprogramowania (SDLC, Software Development Life Cycle). Z tego względu ostateczne dostarczenie wartości i satysfakcji dla biznesu i klientów organizacji było traktowane z najniższym/najmniejszym priorytetem. Czasami ten konkretny aspekt był całkowicie ignorowany.

Aby rozwiązać ten problem, tych 17 osób opracowało manifest, znany jako *Manifesto for Agile Software Development*. Zawierał on wskazówki wprowadzające nowe podejście

do zwinnego sposobu pracy przy tworzeniu i dostarczaniu oprogramowania. Manifest ten całkowicie zmienił świat rozwoju oprogramowania, przewidując przyspieszenie procesu rozwoju i dostarczania działającego oprogramowania wysokiej jakości. Odkryli oni lepsze sposoby wdrażania oprogramowania, oczekując, że zrobią, pomagając w tym innym i skupiając się na czterech wartościach i 12 zasadach podanych przez nich w Manifeście. Poniższe cztery wartości są łatwe do zrozumienia.

- Ludzie i interakcje ponad procesy i narzędzia
- Działające oprogramowanie ponad szczegółową dokumentację
- Współpraca z klientem ponad negocjacje umów
- Reagowanie na zmiany ponad realizację założonego planu*

Te cztery wartości z Manifestu zawsze muszą być przestrzegane przez zespoły tworzące oprogramowanie, mając cały czas na uwadze, że elementy wymienione po lewej stronie muzą być bardziej cenione niż te wymienione po prawej.

Wartość *Ludzie i interakcje ponad procesy i narzędzia* kładzie większy nacisk na docenianie ludzi, którzy tworzą i dostarczają działające oprogramowanie, niż na docenianie procesów lub narzędzi, które są przez nich w tym celu wykorzystywane.

> *Wszystkie interakcje między ludźmi są okazją do uczenia się lub nauczania.*
> *– Stephen Covey*

Dzięki temu zwracamy uwagę na fakt, że choć właściwe procesy i narzędzia są niezbędne do tworzenia oprogramowania, sensowna grupa deweloperów jest niezbędna, aby zrealizować rozwój i dostarczenie oprogramowania. Gdy mamy taką grupę, działa jako zespół deweloperski, który musi wykonać określone działania związane z tworzeniem i dostarczaniem oprogramowania, ustanawiając i ulepszając swój odpowiedni sposób zwinnej pracy, zgodnie z ich potrzebami i oczekiwaniami.

> *Technika jest niczym. Ważna jest wiara w ludzi, że są oni zasadniczo dobrzy i mądrzy, a jeśli damy im narzędzia, zrobią za ich pomocą wspaniałe rzeczy.*
> *– Steve Jobs*

To zespół musi reagować na potrzeby i wymagania biznesu oraz klientów lub użytkowników końcowych, aby realizować działania związane z tworzeniem oprogramowania. Zespół musi wchodzić w interakcje (zarówno wewnętrzne w ramach zespołu, jak i zewnętrzne z interesariuszami), aby mieć pewność, że są na właściwej drodze, dzięki unikaniu

---

* Przekład wg Wikipedii – tłumacz starał się zachować sposób przekładu utrwalony już w sieci. Podobnie jest dalej w przypadku 12 zasad (przyp. tłum.).

i przezwyciężaniu nieprawidłowości, jeśli takie wystąpią. Jeśli taki proces myślowy jest przeprowadzony poprzez nadawanie większego znaczenia procesom i narzędziom, to zespól stanie się miej elastyczny na wciąż zmieniające się wymagania biznesu i klientów. Powoduje to duże ryzyko, że nie trafimy w potrzeby klienta. Ma to także negatywny wpływ na wynikowe oprogramowanie wychodzące do sieci, powodując marnowanie wysiłku, czasu i pieniędzy.

Nie można nie zgodzić się z faktem, że procesy i narzędzia są naprawdę potrzebne przy implementacji oprogramowania dobrej jakości. Działają jak aktywatory i nie można pominąć ich wykorzystania. Jednak te procesy i narzędzia nie mogą działać same. To ludzie umożliwiają im działanie. Ludzie to zwierzęta społeczne i mogą pracować wspólnie, aby zrobić to, co trzeba z istniejącymi procesami i dostępnymi narzędziami. Są one bardziej skuteczne, gdy ludzie współpracują jako jeden ZESPÓŁ zgodnie z filozofią TEAM (Together Everyone Achives More – razem każdy osiąga więcej).

Wartość *Działające oprogramowanie ponad szczegółową dokumentację* stwierdza, że od zespołu deweloperskiego oczekuje się tworzenia dokumentacji dostarczającej pewnej wartości. Zespoły tworzące oprogramowania powinny zawsze przedkładać działające oprogramowanie nad obszerną dokumentacją. Każda obszerna dokumentacja może potencjalnie utrudnić postęp zespołu, pochłaniając ich cenny czas i wysiłek, które można wykorzystać tak, gdzie przyniesie to większą wartość.

Jeśli zespół zajmie się przez dłuższy czas tworzeniem dokumentacji wymagań, analizy, projektowania i przypadków testowych, istnieje możliwość, że dokumentacja stanie się przestarzała w momencie zakończenia pracy. Działające oprogramowanie musi stale uwzględniać zmieniające się potrzeby klientów. Członkowie zespołu deweloperskiego spędzający swój cenny czas nad wyczerpującą dokumentacją wprowadzą opóźnienia w całkowitym postępie prac i dlatego nie będą w stanie dostosować się do wciąż zmieniających się wymagań klientów.

Nie potrzebujemy dokładnego dokumentu.
Potrzebne jest nam wzajemne zrozumienie.
– Jeff Patton

Inną możliwością wystąpienia problemu w zespołach jest to, że analizują one, projektują i dokumentują składniki oprogramowania, które mogą nie być potrzebne. Jest to zwykle marnowanie wysiłku, pieniędzy i czasu na dokumentowanie elementów, które mogą w ogóle nie zostać wykorzystane. Zespoły czasem spędzają znaczącą ilość czasu, tworząc dokumentację przed skonstruowaniem oprogramowania, co powoduje opóźnienia w dostarczaniu większej wartości klientom.

Zespół musi postępować zgodnie z filozofią, że tworzy dokumentację, która przynosi wartość bez opóźniania postępów, i unikając opóźnień lub odchyleń. Jednym ze

sposobów spojrzenia na ten problem jest wprowadzenie wielu testów oprogramowania. Te przypadki testowe mogą zostać wykorzystane jako dokumentacja, która może być traktowana jako jedyne źródło prawdy do weryfikowania przewidywanego zachowania oprogramowania.

Jest to potrzebne, aby sprawdzić, czy oprogramowanie działa zgodnie z oczekiwaniami klienta lub użytkownika końcowego. Testy funkcjonalne i niefunkcjonalne, testy ręczne, testy automatyczne do wykonywania testów jednostkowych, testy integracyjne, testy integracji systemu i testy akceptacyjne muszą zostać przeprowadzone, aby poprawić ogólną jakość oprogramowania i sprawdzić, czy oprogramowanie jest stabilne i jest w nim mniej usterek.

Jeden test jest wart tysiąca opinii ekspertów.
– Bill Nye

Zespół otrzymuje pomoc, wykorzystując te wszystkie testy, co pozwala mu sprawdzić, czy opracowane przez nich oprogramowanie jest integrowane, weryfikowane i wdrażane tak, aby użytkownicy końcowi mogli z niego korzystać.

Jednocześnie muszą oni sprawdzać, czy tworzą tylko najbardziej wartościową dokumentację dla tworzonego oprogramowania. Celem powinna być zawsze budowa działającego oprogramowania i to powinien być główny parametr ich postępów. Klienci bardziej martwią się o najlepszy sposób rozwiązania swoich problemów. To zespół deweloperski jest odpowiedzialny za to, aby ich wspomagać, stale prezentując im działające oprogramowanie.

Wartość *Współpraca z klientem ponad negocjacje umów* uwypukla ważny aspekt związku między zespołem tworzącym oprogramowanie a ich klientami, którzy będą z niego korzystać. Klienci mają swoje potrzeby, które muszą zostać zaspokojone przez działające oprogramowanie.

Zespół deweloperski musi działać w ścisłej współpracy ze swoimi klientami, aby znać ich prawdziwe potrzeby i problemy. Może to być także wspólny wysiłek zespołu deweloperskiego i odpowiednich ludzi biznesu, gdzie może mieć miejsce interakcja, analiza, badanie i ocena wymagań klienta we współpracy. W tym przypadku kontrakt podpisany przez organizację i jej klientów nie powinien mieć nawet znaczenia.

Jeśli wszyscy podążają razem do przodu, sukces przychodzi sam.
– Henry Ford

Jeśli interesariusze i klienci, którzy podpisali umowę (aby mieć pewność, że klienci będą dostawać to, czego potrzebują), są zadowoleni i usatysfakcjonowani ogólnym postępem działań w zakresie rozwoju oprogramowania i mogą zobaczyć bieżący postęp

w kategoriach działającego oprogramowania, ten rodzaj współpracy sam w sobie automatycznie dokona cudów.

Nikt nie potrafi zagwizdać symfonii. Aby ją zagrać, potrzebna jest cała orkiestra.
– H. Luccock

Zespoły tworzące oprogramowanie mogą nadal go dostarczać zgodnie z pierwotnymi oczekiwaniami klientów. W momencie, w którym klienci zmienią zdanie, potrzeby i oczekiwania (które także wpływają na priorytety opracowywanych i dostarczanych funkcjonalności), dobrze jest, aby zespoły były elastyczne. Powinny one uwzględnić taki rodzaj zmian, aby dążyć do zaspokojenia stale zmieniających się potrzeb klientów, zamiast utknąć przy początkowo zdefiniowanych oczekiwaniach.

Wartość *Reagowanie na zmiany ponad realizację założonego planu* zachęca do wprowadzania zmian zamiast ścisłego przestrzegania ustalonego planu. Wszelkie zmiany pochodzące od klientów wiążą się z kosztami, wysiłkiem i czasem, a także wieloma innymi czynnikami, które są nieuniknione. Przestrzeganie określonych standardowych działań przy tworzeniu oprogramowania pozwala zespołom deweloperskim na redukcję kosztów zmian. Zespoły muszą być samowystarczalne, aby zmierzyć wpływ zmian i zareagować na nie.

Przyjmij podejście, że ciągłe planowanie jest czymś dobrym. W każdej iteracji należy oczekiwać zmiany planów (choć w niewielkim stopniu, jeśli planowanie jest skuteczne). Nie wpadaj w pułapkę myślenia, że plan jest bez wad.
– Ian Spence i Kurt Bittner

Zaufanie i przejrzystość to dwa czynniki pomagające zespołowi, za pomocą których może on przekazywać informacje o swoich postępach niezależnie od zachodzących zmian powodujących odchylenia. Dzieje się tak, gdyż zmiany są ciągłe i nieuniknione. Mając właściwe oczekiwania wraz z otwartością i zaufaniem między zespołami a interesariuszami, zespoły muszą informować interesariuszy o swoich szczerych opiniach, aby mieć pewność, że podejmowanie decyzji jest zawsze działaniem opartym na konsensusie.

Manifest (*Manifesto for Agile Software Development*) wspomina także o 12 zasadach (których powinny przestrzegać zespoły deweloperskie), działających na rzecz czterech wymienionych wcześniej wartości. Poniższych 12 zasad jest łatwych do zrozumienia.

- Najwyższy priorytet ma dla nas zadowolenie klienta dzięki wczesnemu i ciągłemu wdrażaniu wartościowego oprogramowania.
- Bądźcie gotowi na zmiany wymagań nawet na późnym etapie jego rozwoju. Procesy zwinne wykorzystują zmiany dla zapewnienia klientowi konkurencyjności.

- Dostarczajcie funkcjonujące oprogramowanie często, w kilkutygodniowych lub kilkumiesięcznych odstępach. Im częściej, tym lepiej.
- Zespoły biznesowe i deweloperskie muszą ściśle ze sobą współpracować w codziennej pracy przez cały czas trwania projektu.
- Twórzcie projekty wokół zmotywowanych ludzi. Zapewnijcie im potrzebne środowisko oraz wsparcie i zaufajcie, że wykonają powierzone zadanie.
- Najbardziej efektywnym i wydajnym sposobem przekazywania informacji zespołowi deweloperskiemu i wewnątrz niego jest rozmowa twarzą w twarz.
- Działające oprogramowanie jest podstawową miarą postępu.
- Procesy zwinne umożliwiają zrównoważony rozwój. Sponsorzy, deweloperzy oraz użytkownicy powinni być w stanie utrzymywać równe tempo pracy.
- Ciągłe skupienie na technicznej doskonałości i dobrym projektowaniu zwiększa zwinność.
- Prostota – sztuka minimalizowania ilości koniecznej pracy – jest kluczowa.
- Najlepsze rozwiązania architektoniczne, wymagania i projekty pochodzą od samoorganizujących się zespołów.
- W regularnych odstępach czasu zespół analizuje możliwości poprawy swojej wydajności, a następnie dostraja i dostosowuje swoje działania do wyciągniętych wniosków.

Zasadę *„Najwyższy priorytet ma dla nas zadowolenie klienta dzięki wczesnemu i ciągłemu wdrażaniu wartościowego oprogramowania”* można podsumować w dwóch słowach jako *zadowolenie klienta*. Zespoły tworzące oprogramowanie powinny zawsze mieć na uwadze, że dla nich najwyższym priorytetem jest zadowolenie klientów poprzez sprawdzanie, czy zawsze dostarczają im wysokiej jakości działające oprogramowanie, zarówno wcześnie, jak i w sposób ciągły.

Podczas dostarczania wartościowego oprogramowania ostateczną drogą do zapewnienia zadowolenia klienta jest dostarczenie oprogramowania wcześnie i zgodnie z podejściem „im szybciej, tym lepiej”. Należy też dostarczać iteracyjne, czyli wielokrotnie w sekwencji wyników, i przyrostowo, czyli przy każdej dostawie należy dodawać wartość. Zespoły deweloperskie muszą to robić, wsłuchując się stale w wymagania swoich klientów. Dzięki stosowaniu tego podejścia przyrostowe i iteracyjne dostarczanie działającego oprogramowania daje klientom wartość na czas w znacznie szybszy sposób. Pomaga też zespołowi deweloperskiemu poznać rzeczywiste potrzeby i oczekiwania klientów, jednocześnie uwzględniając ewentualne zmiany.