PRENTICE HALL

Agile

Metodyki zwinne w planowaniu projektów

Mike Cohn

Helion

Tytuł oryginału: Agile Estimating and Planning

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-4486-0

Authorized translation from the English language edition, entitled:
AGILE ESTIMATING AND PLANNING; ISBN 0131479415; by Mike Cohn;
published by Pearson Education, Inc., publishing as Prentice Hall PTR.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/agilem
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

CZĘŚĆ IV TWORZENIE HARMONOGRAMÓW

ROZDZIAŁ 3.

Zwinne podejście

Dobry plan bezwzględnie zrealizowany dziś jest lepszy
niż idealny plan zrealizowany w przyszłym tygodniu.
— generał George S. Patton

Choć ruch Agile rozpoczął się wiele lat temu, to oficjalnym jego początkiem było stworzenie Manifestu Agile w lutym 2001 roku (Beck i in.). Ten manifest został napisany i podpisany przez siedemnastu zwolenników „lekkich metod", jak ich wówczas nazywano. Ich dokument zarówno dał nazwę stosowanej metodzie wytwarzania oprogramowania, jak i dostarczył listę wartościowych twierdzeń. Autorzy Manifestu Agile napisali:

- ludzie i interakcje między nimi są ważniejsze niż procesy i narzędzia;
- działające oprogramowanie jest bardziej wartościowe niż obszerna dokumentacja;
- współpraca z klientami jest ważniejsza od negocjowania kontraktów;
- reagowanie na zmiany jest ważniejsze niż trzymanie się planu.

Zespoły Agile cenią bardziej osoby i interakcje od procesów i narzędzi, ponieważ wiedzą, że dobrze funkcjonujący zespół złożony z doskonałych jednostek dysponujących przeciętnymi narzędziami zawsze będzie działał wydajniej niż zespół złożony z przeciętnych osób posługujących się doskonałymi narzędziami i procesami. Świetne oprogramowanie jest tworzone przez świetnych programistów. W naszej branży podejmowaliśmy zbyt wiele wysiłków, które przynosiły zbyt mało sukcesów, aby zdefiniować proces rozwoju, w którym ludzie są tak traktowani jak wymienne tryby w maszynie. W procesach Agile uznajemy wyjątkowe zalety (i wady) indywidualnych osób i staramy się je wykorzystać. Nie traktujemy wszystkiego tak, jakby były to jednorodne elementy.

Zespoły Agile bardziej cenią działające oprogramowanie od kompleksowej dokumentacji, ponieważ dzięki temu na końcu każdej iteracji otrzymują coraz lepszą wersję produktu. To pozwala wcześnie i często zbierać opinie dotyczące produktu i procesu. W miarę rozwoju opracowywanego oprogramowania w każdej iteracji można je prezentować potencjalnym lub rzeczywistym użytkownikom. Opinie tych użytkowników trafiają z powrotem do procesu projektowania. W ten sposób można uzyskać pewność, że zespół zawsze pracuje nad najbardziej wartościowymi funkcjonalnościami oraz że te funkcjonalności spełniają oczekiwania użytkowników.

Współpracę z klientem cenimy bardziej niż negocjacje kontraktów, ponieważ zespoły Agile dążą do tego, aby wszyscy uczestnicy projektu starali się osiągnąć taki sam zbiór celów. Negocjacje kontraktu czasami powodują, że zespół programistów i klient od początku projektu stoją po przeciwnych stronach barykady. Uwielbiam grać w większość gier. Gdy moja starsza córka miała cztery lata, kupiłem jej grę kooperacyjną, ponieważ wydawało mi się, że jej się spodoba, oraz ponieważ nie miałem pojęcia, jak zabawne mogą być tego rodzaju kooperacyjne gry. W grze, którą jej kupiłem, na księżniczkę zostało rzucone zaklęcie, a gracze muszą pokonać przeszkody (fosę, zamknięte drzwi i tym podobne) znajdujące się między nimi a księżniczką. Gracze wykonują ruchy na zmianę tak, jak w większości gier, ale celem jest wspólne usunięcie przeszkód i ocalenie księżniczki. Wszyscy gracze wygrywają albo wszyscy przegrywają. Gra w tę grę to wspaniała zabawa. Chcielibyśmy, aby zespoły pracujące nad oprogramowaniem i klienci podchodzili do projektów z taką samą wolą współpracy i poczuciem wspólnych celów. To prawda, że kontrakty często są potrzebne, ale warunki i szczegóły zawarte w kontraktach mogą mieć olbrzymi wpływ na to, czy obie strony będą koncentrowały się na współpracy, czy na rywalizacji.

Zespoły Agile bardziej cenią reagowanie na zmiany od podążania za planem, ponieważ ich ostatecznym celem jest dostarczenie jak najwięcej wartości dla użytkowników i klientów projektu. W żadnych projektach, z wyjątkiem najprostszych, nie ma możliwości, by użytkownicy znali wszystkie szczegóły wszystkich żądanych funkcjonalności. Z całą pewnością użytkownicy będą zgłaszać nowe pomysły i prawie tak samo nieuniknione jest to, że niektóre funkcjonalności, które dziś są pożądane, jutro będą miały niższy priorytet. Dla zespołu Agile plan jest jedną z wizji przyszłości, ale możliwych jest ich wiele. Zespół powinien uwzględniać w planie nową wiedzę i doświadczenia. Możliwe, że zespół pracuje szybciej lub wolniej niż pierwotnie zakładano; być może użytkownikom jeden zbiór funkcjonalności podoba się bardziej niż oczekiwano, a nie podoba się inna funkcjonalność, która początkowo była uznawana za kluczową.

Mając na uwadze cztery stwierdzenia dotyczące wartości z Manifestu Agile, w tym rozdziale przyjrzymy się temu, co to znaczy stosować zwinne podejście do projektu, a także co to znaczy mieć zwinne podejście do oceny i planowania.

Zwinne podejście do projektów

Pamiętając o czterech głównych wartościach Agile, możemy zwrócić uwagę na to, jak zespół Agile wygląda w praktyce. Stosowanie czterech wymienionych wartości łącznie prowadzi do procesów wytwarzania oprogramowania, które są w wysokim stopniu iteracyjne i przyrostowe. W wyniku ich stosowania na koniec każdej iteracji otrzymujemy zakodowane i przetestowane oprogramowanie. W poniższych punktach opisano niektóre ważniejsze cechy pracy zespołów Agile. Zespół Agile:

- działa jako jeden kolektyw;
- pracuje w krótkich iteracjach;
- w każdej iteracji coś dostarcza;
- koncentruje się na priorytetach biznesowych;
- sprawdza i doskonali swoją pracę.

Zespół Agile działa jako jeden kolektyw

Dla powodzenia projektu kluczowe znaczenie ma to, aby wszyscy uczestnicy projektu postrzegali siebie jako jeden zespół, który realizuje wspólny cel. W projekcie Agile nie ma miejsca na mentalność typu „zrobiłem swoje i reszta mnie nie interesuje". Analitycy nie rzucają projektantom opracowanych wymagań. Projektanci i architekci nie rzucają projektów przez ścianę programistom, a programiści nie rzucają przez ścianę w połowie przetestowanego kodu testerom. Skuteczny zespół Agile musi prezentować myślenie w rodzaju „uczestniczymy w tym razem". Chociaż zespół Agile musi pracować razem, jako jedna drużyna, to można w nim wyróżnić kilka konkretnych ról. Warto zidentyfikować i objaśnić te role, które mają znaczenie dla zwinnego szacowania i planowania.

Pierwsza rola to *właściciel produktu* (ang. *product owner*). Do najważniejszych obowiązków właściciela produktu należy dbanie o to, aby wszyscy członkowie zespołu dążyli do wspólnej wizji projektu, ustalanie priorytetów, tak aby prace zawsze dotyczyły funkcjonalności o największej wartości, oraz podejmowanie decyzji prowadzących do wysokiej stopy zwrotu z inwestycji w projekt. W rozwoju komercyjnego oprogramowania właściciel produktu często jest osobą z działu marketingu lub przedstawicielem działu zarządzania produktami w firmie. W przypadku opracowywania oprogramowania do użytku wewnętrznego właściciel produktu może być użytkownikiem, menedżerem użytkowników, analitykiem lub osobą finansującą projekt.

Druga rola to *klient*. Klient jest osobą, która podjęła decyzję o finansowaniu projektu lub zakupie oprogramowania. W projekcie rozwoju oprogramowania do użytku wewnętrznego klient zazwyczaj jest przedstawicielem innego zespołu lub działu. W takich projektach role właściciela produktu i klienta są często połączone. W przypadku produktu rozprowadzanego komercyjnie klientem jest osoba, która kupuje oprogramowanie. W każdym przypadku klient może, lecz nie musi być *użytkownikiem* oprogramowania, co oczywiście jest kolejną, ważną rolą.

Inną rolą wartą podkreślenia jest *deweloper* (ang. *developer*). Określenia *deweloper* używam w odniesieniu do wszystkich osób uczestniczących w procesie rozwoju oprogramowania. Obejmuje to programistów, testerów, analityków, inżynierów bazy danych, specjalistów w dziedzinie użyteczności (ang. *usability*), dokumentalistów, architektów, projektantów i tak dalej. Jeśli korzysta się z tej definicji, jako deweloper w wielu projektach może być traktowany nawet właściciel produktu.

Ostatnia rola to *menedżer projektu* (ang. *project manager*). Jak opisuje Highsmith (2004a), rola menedżera projektu w projektach Agile zmieniła się. Menedżerowie projektów Agile koncentrują się bardziej na roli liderów niż specjalistów w dziedzinie zarządzania. W niektórych projektach Agile osoba pełniąca rolę menedżera projektu pełni również inną rolę — często pracuje jako deweloper, ale od czasu do czasu także jako właściciel produktu.

Zespół Agile pracuje w krótkich iteracjach

W projekcie Agile nie ma wyraźnego podziału na etapy — nie istnieje etap opracowywania wstępnych wymagań, po którym następuje faza analizy, projektu architektury i tak dalej. W zależności od konkretnego zwinnego procesu, który wybierzemy lub ustalimy, na początku projektu możemy przeprowadzić krótką fazę projektowania, modelowania lub inną. Ale kiedy projekt rozpocznie się na poważnie, to wszystkie prace (analiza, projektowanie, kodowanie, testowanie i tym podobne) w każdej iteracji będą odbywać się równolegle.

Czas iteracji jest ściśle określony w *oknach czasowych*. To oznacza, że iteracje kończą się na czas nawet wtedy, gdy decydujemy się na usunięcie funkcjonalności. Okna czasowe iteracji często są bardzo krótkie. Większość zespołów Agile pracuje w iteracjach trwających od dwóch do czterech tygodni, choć niektóre zespoły wydłużają czas ich trwania nawet do trzech miesięcy. Większość zespołów ustala względnie stałą długość iteracji. Niektóre jednak wybierają odpowiedni czas trwania na początku każdej iteracji.

Zespół Agile w każdej iteracji coś dostarcza

Ważniejszy od wybierania przez zespół konkretnej długości iteracji jest fakt, że podczas iteracji następuje przekształcenie jednego lub kilku nieprecyzyjne określonych twierdzeń w specyfikacji wymagań w zakodowane, przetestowane i potencjalnie gotowe do dostarczenia oprogramowanie. Oczywiście wiele zespołów nie dostarcza wyników każdej iteracji użytkownikom. Celem jest jednak to, że mogłyby to zrobić. To oznacza, że w każdej iteracji zespoły robią postępy poprzez dodawanie jednej bądź kilku małych funkcjonalności, ale każda dodana funkcjonalność jest kodowana, testowana i ma jakość pozwalającą na wdrożenie.

Kluczowe znaczenie ma to, aby na końcu każdej iteracji produkt został doprowadzony do stanu pozwalającego na dostarczenie go użytkownikom. W praktyce nie wynika z tego, że zespół musi zrobić absolutnie wszystko, co jest konieczne do opublikowania oprogramowania, ponieważ często się zdarza, że nie w każdej iteracji następuje publikacja. Na przykład jeden z zespołów, z którym pracowałem, przed opublikowaniem swojego produktu obejmującego zarówno sprzęt, jak i oprogramowanie potrzebował dwóch miesięcy testów średniego czasu między awariami (ang. *mean time between failure* — **MTBF**). Nie można było skrócić tego dwumiesięcznego okresu ze względu na wymogi kontraktu z klientem. Taka ilość czasu jest często konieczna do sprawdzenia, czy nie nastąpiły awarie sprzętu. Zespół pracował w czterotygodniowych iteracjach i pomimo uruchamiania dwumiesięcznych testów MTBF na koniec każdej iteracji jego produkt był w stanie nadającym się do wydania.

Ponieważ pojedyncza iteracja zwykle nie zapewnia wystarczająco dużo czasu na zrealizowanie tylu nowych funkcjonalności, aby zaspokoić potrzeby lub życzenia użytkownika, wprowadzono szersze pojęcie *wydania* (ang. *release*). Wydanie obejmuje jedną lub więcej (zwykle więcej) iteracji, które bazują na sobie i mają na celu zrealizowanie kompletnego zbioru powiązanych ze sobą funkcjonalności. Chociaż iteracje zazwyczaj trwają od dwóch do czterech tygodni, wydanie zwykle trwa od dwóch do sześciu miesięcy. Na przykład w systemie zarządzania inwestycjami jedno wydanie może obejmować wszystkie funkcjonalności związane z kupowaniem i sprzedażą funduszy powierniczych i funduszy kapitałowych. Realizacja tych funkcjonalności może zająć sześć dwutygodniowych iteracji (około trzech miesięcy). W kolejnym wydaniu mogą zostać dodane funkcjonalności związane z obrotem akcjami i obligacjami, które mogą zająć dodatkowe dwutygodniowe iteracje. Wydania mogą odbywać się z różną częstotliwością. Opracowanie pierwszego wydania może zająć sześć miesięcy. Kolejne wydanie może nastąpić trzy miesiące później i tak dalej.

Zespół Agile koncentruje się na priorytetach biznesowych

Zespół Agile wykazuje zaangażowanie w priorytety biznesowe na dwa sposoby. Przede wszystkim dostarcza funkcjonalności w kolejności określonej przez właściciela produktu, od którego oczekuje się ustalenia priorytetów i połączenia funkcjonalności w wydania w taki sposób, aby zoptymalizować zwrot z inwestycji w projekt dla firmy. Aby osiągnąć ten cel, tworzony jest plan wydań uporządkowany

według priorytetów. Podczas jego tworzenia uwzględniane są możliwości zespołu oraz lista pożądanych nowych funkcjonalności. Aby właściciel produktu mógł uzyskać maksymalną elastyczność w ustalaniu priorytetów, tworzone funkcjonalności muszą być pisane w taki sposób, aby występowało między nimi jak najmniej technicznych zależności. Jeśli wybór jednej funkcjonalności wymaga wcześniejszego zrealizowania trzech innych, to właściciel produktu będzie miał trudności z ustaleniem priorytetów funkcjonalności w taki sposób, aby powstał sensowny plan wydań. Zespół raczej nie osiągnie celu w postaci całkowitego braku zależności, jednak utrzymanie zależności na minimalnym poziomie często jest zupełnie realne.

Poza tym zespoły Agile koncentrują się na realizowaniu i dostarczaniu funkcjonalności wartościowych dla użytkowników, a nie na wykonywaniu odizolowanych zadań (które ostatecznie łączą się w funkcjonalność przynoszącą wartość dla użytkownika). Jednym z najskuteczniejszych sposobów osiągnięcia tego celu jest wykorzystywanie historyjek użytkownika — nieskomplikowanej techniki wyrażania wymagań dla oprogramowania (Cohn 2004). *Historyjka użytkownika* to krótki opis funkcjonalności z punktu widzenia użytkownika lub klienta systemu. Historyjki użytkownika mają swobodną formę. Nie istnieje obowiązkowa składnia. Jednak ogólnie rzecz biorąc, warto zadbać o to, aby historyjki użytkownika przyjęły następującą ogólną formę: „Jako <typ użytkownika> chcę <możliwość>, tak aby uzyskać <wartość dla biznesu>”. W przypadku tego przykładowego szablonu historyjka użytkownika może mieć następujące brzmienie: „Jako nabywca książki chcę mieć możliwość znalezienia książki po numerze ISBN tak, aby móc szybko znaleźć odpowiednią książkę”.

Historyjki użytkownika są nieskomplikowane, ponieważ prace wymagane do ich zebrania i udokumentowania nie są wykonywane wszystkie jednocześnie. Zamiast pisać obszerne specyfikacje wymagań, zespoły Agile odkryły, że lepsze będzie podejście do opracowywania wymagań typu „dokładnie na czas”. Zazwyczaj proces powstawania historyjki użytkownika zaczyna się od ręcznego zapisania historyjki na karteczce z notesu albo — w przypadku dużego zespołu lub zespołu rozproszonego geograficznie — napisania jej na komputerze. Karta historyjki to jednak dopiero początek. Tworzeniu każdej historyjki użytkownika towarzyszy tyle rozmów między deweloperami a właścicielem produktu, ile jest potrzebne. Rozmowy te odbywają się tak często, jak to konieczne. Uczestniczą w nich wszystkie niezbędne osoby. Stosowanie podejścia do wymagań bazującego na historyjkach użytkowników nie wyklucza istnienia pisemnej dokumentacji. Jednak akcent wyraźnie przesuwa się z komunikacji pisemnej na werbalną.

Zespół Agile sprawdza i doskonali swoją pracę

Plan stworzony na początku dowolnego projektu nie jest gwarancją tego, co się wydarzy. Jest to po prostu próba przewidzenia efektów w określonym punkcie w czasie. Plan może zdezaktualizować się z powodu wielu czynników — członkowie projektu przychodzą i odchodzą, technologie sprawdzają się lepiej lub gorzej niż oczekiwano, użytkownicy zmieniają zdanie, konkurencja zmusza nas do reakcji innej, szybszej i tak dalej. Zespoły Agile postrzegają każdą taką zmianę zarówno jako okazję, jak i potrzebę zaktualizowania planu tak, aby lepiej odzwierciedlał rzeczywistość bieżącej sytuacji.

Na początku każdej nowej iteracji zespół Agile wykorzystuje całą nową wiedzę zdobytą w poprzedniej iteracji i odpowiednio się dostosowuje do warunków. Jeśli zespół dowiedział się czegoś, co może mieć wpływ na dokładność lub wartość planu, to wprowadza odpowiednie korekty. Na dokładność planu może mieć wpływ odkrycie, że zespół źle ocenił tempo swojej pracy — tzn. w rzeczywistości pracuje szybciej lub wolniej. Mogło się również okazać, że określony rodzaj pracy jest bardziej czasochłonny niż wcześniej przypuszczano.

Na wartość planu może wpłynąć zdobycie przez właściciela produktu wiedzy na temat funkcjonalności pożądanych przez potencjalnych użytkowników. Może się zdarzyć, że na podstawie obserwacji oprogramowania z wcześniejszej iteracji właściciel produktu dowiedział się, że użytkownicy chcieliby jedną z funkcjonalności rozwinąć bardziej oraz że inna funkcjonalność nie podoba im się tak, jak wcześniej sądzono. W takim przypadku zespół może poprawić wartość planu przez przesunięcie do wydania więcej pożądanych funkcjonalności kosztem funkcjonalności postrzeganych jako mniej cenne.

Nie oznacza to, że zespoły Agile gwałtownie zmieniają ustalone priorytety w wyniku obserwacji bieżącej sytuacji. Priorytety są stosunkowo stabilne między kolejnymi iteracjami. Jednak możliwość zmiany priorytetów między iteracjami istotnie przyczynia się do maksymalizacji zwrotu z inwestycji w projekt.

Zwinne podejście do planowania

Szacowanie i planowanie rozwoju nowego produktu jest żmudnym zadaniem, które dodatkowo utrudnia niewłaściwe zrozumienie istoty projektów. Macomber (2004) twierdzi, że nie powinniśmy postrzegać projektu wyłącznie jako realizacji szeregu etapów. Zamiast tego należy raczej postrzegać projekt jako proces, w wyniku którego szybko i wiarygodnie jest generowany strumień nowych przydatnych możliwości i wiedzy. Nowe możliwości są dostarczane w produkcie. Nowa wiedza jest używana do tego, aby produkt stał się możliwie jak najlepszy.

W projekcie Agile ten strumień nowych możliwości i wiedzy wykorzystujemy jako siłę napędową do wykonywania bieżącej pracy. Nowa wiedza wygenerowana przez projekt może dotyczyć zarówno produktu, jak i projektu. Nowa *wiedza o produkcie* pomaga nam dowiedzieć się więcej o tym, czym powinien być produkt. Nowa *wiedza o projekcie* jest informacją o zespole, stosowanych technologiach czy zagrożeniach.

Często nie uznajemy tej wiedzy i jej nie planujemy. Nieuwzględnianie nowej wiedzy w planie prowadzi do budowania planów z założeniem, że wiemy wszystko, co jest konieczne do stworzenia dokładnego planu. W świecie wytwarzania oprogramowania taka sytuacja zdarza się rzadko, jeśli w ogóle kiedykolwiek się zdarza. Ward Cunningham powiedział: *it's more planning what you want to learn, not what it will be in the end*[1] (Van Schooenderwoert 2004).

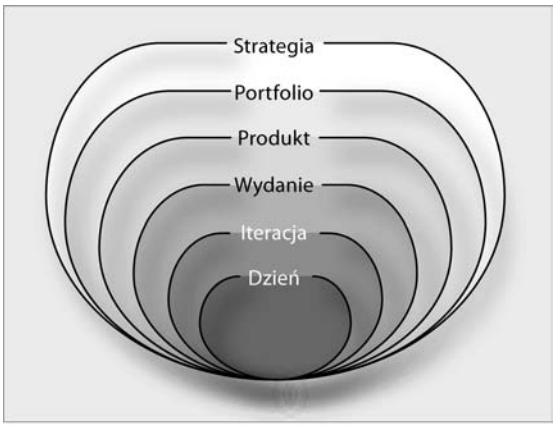
Często porównuję tradycyjne spojrzenie na projekt do biegu na dystansie dziesięciu kilometrów. Wiesz dokładnie, jak jest daleko do mety, a Twoim celem jest dotarcie do niej możliwie jak najszybciej. W projekcie Agile nie wiemy dokładnie, gdzie jest meta, ale często wiemy, że musimy do niej dotrzeć albo jak najbardziej się do niej zbliżyć przed określoną datą. Lepszym porównaniem dla projektu Agile jest wyścig trwający określony czas niż wyścig na ustalonym dystansie dziesięciu kilometrów: przebiegnij tyle, ile zdołasz w ciągu sześćdziesięciu minut. W ten sposób zespół Agile wie, kiedy skończy, ale nie wie, co do tego czasu zdoła dostarczyć. Kiedy uświadomimy sobie zarówno to, że efekt końcowy jest w pewnym sensie nieznany, jak i to, że nie można go z góry poznać, to planowanie staje się procesem ustalania i weryfikowania celów, które prowadzą do celu długoterminowego.

[1] Dosł. Interesuje nas bardziej nauczenie się planowania niż to, czym to [produkt] ostatecznie będzie — *przyp. tłum.*

Wiele poziomów planowania

Podczas ustalania i weryfikowania celów należy pamiętać, że nie jesteśmy w stanie zobaczyć tego, co jest za horyzontem, oraz że dokładność planu gwałtownie spada, im bardziej staramy się planować poza to, co możemy zobaczyć. Wyobraź sobie, że stoisz na małej łodzi, a Twoje oczy są na poziomie trzech metrów nad wodą. W tym przypadku odległość do horyzontu wynosi nieco ponad sześć kilometrów[2]. Jeśli planujesz trzydziestokilometrową podróż, to powinieneś planować ją, patrząc do przodu co najmniej pięć razy — raz na sześć kilometrów. Ponieważ nie możesz zobaczyć tego, co jest za horyzontem, musisz spoglądać na horyzont co jakiś czas i korygować swój plan.

Projekt jest zagrożony, jeśli planowanie mocno wykracza poza horyzont planisty i nie uwzględnia czasu na to, by planista podniósł głowę, spojrzał na nowy horyzont i dokonał korekt. Plan powinien być opracowywany progresywnie. Zespoły Agile osiągają ten cel poprzez planowanie dla trzech różnych horyzontów. Są nimi wydanie, iteracja i bieżący dzień. Relacje między nimi (i innymi) horyzontami planowania zostały zilustrowane na „cebuli planowania" na rysunku 3.1.



RYSUNEK 3.1. *Cebula planowania. Zespoły Agile planują co najmniej trzy razy — na poziomie wydań, iteracji i dni*

Większość zespołów Agile jest zainteresowana tylko trzema, wewnętrznymi poziomami cebuli planowania. W planowaniu wydania analizowane są historyjki użytkownika lub motywy, które będą opracowywane w nowej wersji produktu lub systemu. Celem planowania wydania jest udzielenie właściwej odpowiedzi na pytania dotyczące zakresu, harmonogramu i zasobów do projektu. Planowanie wydań odbywa się na początku projektu, ale nie jest to pojedyncze przedsięwzięcie. Dobry plan wydań jest aktualizowany w trakcie realizacji projektu (zwykle na początku każdej iteracji). Dzięki temu stale odzwierciedla aktualne oczekiwania co do tego, jakie funkcjonalności zostaną uwzględnione w kolejnym wydaniu.

Na następnym poziomie jest planowanie iteracji, które jest przeprowadzane na początku każdej iteracji. Na podstawie pracy wykonanej w właśnie zakończonej iteracji właściciel produktu identyfikuje

[2] Aby obliczyć odległość do horyzontu w kilometrach, należy pomnożyć pierwiastek kwadratowy z wysokości, na której znajdują się oczy, przez 3,57.

prace o wysokim priorytecie, które zespół powinien zrealizować w nowej iteracji. Ze względu na to, że patrzymy na bliższy horyzont w porównaniu z planowaniem wydania, komponenty planu iteracji mogą być mniejsze. Podczas planowania iteracji mówimy o zadaniach, które będą wymagane do tego, aby przekształcić żądanie funkcjonalności w działające i przetestowane oprogramowanie.

Na końcu mamy do czynienia z planowaniem dnia. Większość zespołów Agile stosuje jakąś formę codziennych spotkań *stand-up* w celu skoordynowania pracy i zsynchronizowania codziennych wysiłków. Chociaż uwzględnianie takiego planowania w sensie formalnym może wydawać się przesadą, to w czasie trwania tych spotkań zespoły z całą pewnością tworzą, oceniają i weryfikują swoje plany. Podczas codziennych spotkań zespoły ograniczają horyzont planowania do poziomu co najwyżej następnego dnia, kiedy odbywa się kolejne spotkanie. Z tego powodu koncentrują się na planowaniu i koordynacji pojedynczych działań, które prowadzą do realizacji zadań.

Dzięki planowaniu z uwzględnieniem wymienionych trzech horyzontów — wydania, iteracji i dnia — zespoły Agile koncentrują się na tym, co jest widoczne i ważne z punktu widzenia tworzonego planu.

W większości przypadków pojedyncze zespoły Agile nie koncentrują się na planowaniu produktu, portfolio i strategii (ta tematyka nie jest także poruszana w niniejszej książce). Planowanie produktu wymaga od właściciela produktu patrzenia w przyszłość poza bieżące wydanie oraz planowania ewolucji wydawanego produktu lub systemu. Planowanie portfolio zakłada dobór produktów, które będą jak najlepiej realizowały wizję ustaloną podczas planowania strategii organizacji.

Warunki satysfakcji

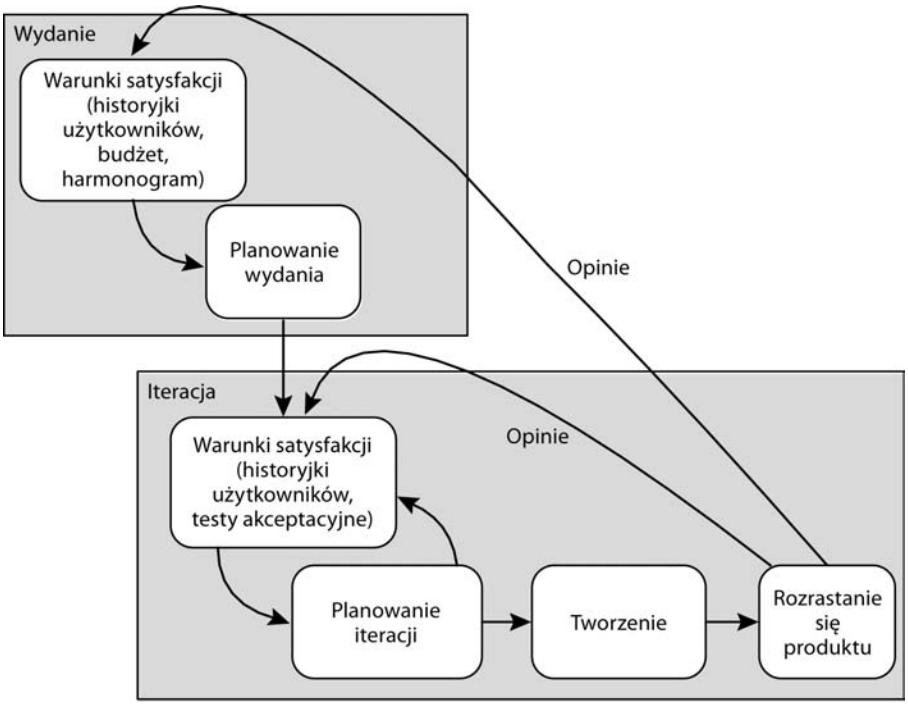
Każdy projekt jest inicjowany począwszy od ustalenia zbioru celów. Naszym bieżącym projektem może być stworzenie najlepszego na świecie edytora tekstu. Utworzenie takiego właśnie edytora tekstu zazwyczaj jest tylko jednym z celów tego projektu. Prawie na pewno istnieją dodatkowe cele związane z harmonogramem, budżetem i jakością. Te cele mogą być rozumiane jako *warunki satysfakcji* klienta lub właściciela produktu. Innymi słowy, warunki satysfakcji określają kryteria wykorzystywane do zmierzenia sukcesu projektu.

Kiedy byłem w szkole średniej i otrzymywałem zadanie domowe polegające na napisaniu wypracowania na temat jakiejś książki, na przykład *Moby Dicka*, zawsze pytałem nauczyciela, jaką objętość ma mieć praca. Odpowiadał coś w stylu „na pięć stron" i wtedy wiedziałem, jakie są jego podstawowe kryteria satysfakcji. Oczywiście istniał szereg dodatkowych, niepisanych kryteriów satysfakcji — na przykład że wypracowanie ma być starannie napisane, że ma to być moja samodzielna praca i napisana zrozumiałym językiem.

Na początku planowania wydania zespół i właściciel produktu wspólnie analizują kryteria satysfakcji właściciela produktu. Są to typowe elementy — zakres, harmonogram, budżet i jakość — choć zespoły Agile zwykle wolą traktować jakość jako niepodlegającą negocjacjom. Zespół i właściciel produktu szukają sposobów spełnienia wszystkich kryteriów satysfakcji. Na przykład właściciel produktu może być tak samo zadowolony z wydania za pięć miesięcy, które będzie obejmowało implementację jednego zestawu historyjek użytkownika, jak z wydania o miesiąc później, które będzie obejmowało implementację dodatkowych historyjek użytkownika.

Czasami jednak nie można spełnić wszystkich kryteriów satysfakcji właściciela produktu. Zespół może zbudować najlepszy na świecie edytor tekstu, ale nie uda mu się go stworzyć do końca przyszłego miesiąca. Jeśli nie można znaleźć wykonalnego rozwiązania, to trzeba zmienić kryteria satysfakcji.

Z tego powodu planowanie wydań i analiza kryteriów satysfakcji właściciela produktu to zadania w wysokim stopniu iteracyjne, co pokazano na rysunku 3.2.



RYSUNEK 3.2. *Warunki satysfakcji sterują zarówno planowaniem wydania, jak i iteracji*

Po ustaleniu planu obejmującego w przybliżeniu kolejne trzy do sześciu miesięcy pracy plan ten jest wykorzystywany w roli danych wejściowych w procesie planowania pierwszej iteracji. Planowanie iteracji, podobnie jak planowanie wydania, rozpoczyna się od analizy kryteriów satysfakcji właściciela produktu. W przypadku iteracji kryteriami satysfakcji właściciela produktu są zazwyczaj funkcjonalności, które według niego powinny być zrealizowane w następnej kolejności, oraz pewne wysokopoziomowe testy poszczególnych funkcjonalności.

Jako przykład rozważmy serwis internetowy biura podróży, w którym zawarto następującą historyjkę użytkownika: „Jako użytkownik chcę mieć możliwość anulowania rezerwacji". Podczas omawiania tej historyjki z właścicielem produktu deweloperzy dowiedzieli się, że kryteria satysfakcji dla tej historyjki są następujące:

- Użytkownik, który anuluje rezerwację wcześniej niż na dwadzieścia cztery godziny przed planowaną podróżą, otrzymuje pełny zwrot pieniędzy.
- Użytkownik, który anuluje rezerwację później niż 24 godziny przed podróżą, otrzymuje zwrot kosztów pomniejszony o opłatę 25 ¤.
- Kod anulowania rezerwacji jest wyświetlany w serwisie oraz wysyłany e-mailem do użytkownika.

Podobnie jak planowanie wydania, także planowanie iteracji jest procesem iteracyjnym. Właściciel produktu wraz z zespołem omawiają różne sposoby najlepszego spełnienia warunków satysfakcji dla iteracji.

Na rysunku 3.2. zaprezentowano pętle sprzężenia zwrotnego, które prowadzą od pola wynikowego powiększonego produktu do pól warunków satysfakcji na początku planowania zarówno wydania, jak i iteracji. Na podstawie doświadczeń z rozwoju produktu w czasie iteracji zespół może uzyskać wiedzę lub umiejętności, które wpływają na planowanie na jednym bądź kilku poziomach. Na podobnej zasadzie zaprezentowanie przyrostu produktu istniejącym lub prawdopodobnym użytkownikom może wygenerować nową wiedzę, która stanie się powodem wprowadzenia zmian do planów. Zespół Agile uwzględnia te zmiany w swoich planach, co prowadzi do powstania produktu o wyższej wartości.

Podsumowanie

Zespoły Agile pracują wspólnie jako kolektyw, ale w ramach zespołu istnieją funkcje, które są przypisane do konkretnych osób. Pierwsza to właściciel produktu, który jest odpowiedzialny za wizję produktu oraz za ustalenie priorytetów, zgodnie z którymi zespół będzie pracować. Kolejna to klient — osoba, która płaci za projekt lub kupuje oprogramowanie, gdy tylko stanie się dostępne. Kolejne funkcje w projekcie Agile to użytkownicy, deweloperzy i menedżerowie.

Zespoły Agile pracują w krótkich oknach czasowych, w których pod koniec każdej iteracji dostarczane jest działające oprogramowanie. Funkcjonalności stworzone podczas tych iteracji są wybierane na podstawie priorytetów ustalanych dla biznesu. To gwarantuje, że najpierw są opracowywane funkcjonalności najważniejsze. Powszechnie stosowanym przez zespoły Agile sposobem na wyrażanie wymagań użytkowników są historyjki użytkowników. Zespoły Agile mają świadomość, że plany mogą szybko stać się nieaktualne. Z tego powodu w razie potrzeby dostosowują swoje plany do zmieniającej się sytuacji.

Projekty powinny być postrzegane jako przedsięwzięcia, które szybko i wiarygodnie generują strumień nowych przydatnych funkcjonalności oraz nowej wiedzy, a nie tylko jako wykonywanie ciągu czynności. Projekty generują dwa typy nowej wiedzy: wiedzę o produkcie oraz wiedzę o projekcie. Każda z nich przydaje się do udoskonalania planu produktu po to, aby osiągnąć jak największą wartość dla organizacji.

Zespoły Agile stosują planowanie na trzech poziomach: planowanie wydania, planowanie iteracji oraz planowanie codzienne. W planie wydania uwzględnia się przyszłość równą czasowi trwania wydania — zwykle od trzech do sześciu miesięcy. W planie iteracji ważny jest wyłącznie czas trwania pojedynczej iteracji — zazwyczaj od dwóch do czterech tygodni. Plan dzienny to rezultat podjęcia zobowiązania przez członka zespołu przed innymi członkami zespołu podczas codziennego spotkania *stand-up*.

Zrozumienie kryteriów satysfakcji właściciela produktu ma kluczowe znaczenie zarówno podczas planowania wydania, jak i w czasie planowania iteracji. W czasie planowania wydania cały zespół identyfikuje sposób spełnienia kryteriów satysfakcji dla wydania. Kryteria te obejmują zakres, harmonogram i zasoby. Aby to osiągnąć, może być konieczne, żeby właściciel produktu zrezygnował z jednego lub kilku kryteriów satysfakcji. Podobny proces odbywa się podczas planowania iteracji, gdy kryteriami satysfakcji są nowe funkcjonalności, które zostaną zaimplementowane, oraz wysokopoziomowe przypadki testowe, które pokazują, czy funkcjonalności zostały prawidłowo zaimplementowane.

Pytania do dyskusji

1. W jaki sposób praca w jednolitym kolektywie — jako cały zespół — wpłynęła na Twój bieżący lub poprzedni projekt?
2. Jakie są kryteria satysfakcji w Twoim obecnym projekcie? Czy wszyscy interesariusze i uczestnicy projektu zgadzają się ze wszystkimi tymi kryteriami? Jakie zagrożenia wiążą się z kontynuowaniem projektu, w którym nie ma zgody na wszystkie kryteria satysfakcji?
3. Dlaczego budżet i harmonogram zostały wyszczególnione na rysunku 3.2 jako kryteria satysfakcji do uwzględniania podczas planowania wydania, a nie podczas planowania iteracji?

Skorowidz

Agile w planowaniu oznacza sukces projektu!

Projekty związane z tworzeniem oprogramowania zdecydowanie różnią się od innych, bardziej tradycyjnych przedsięwzięć. Jeśli nie weźmiesz tego pod uwagę i uznasz, że konwencjonalne podejście wystarczy, ryzykujesz, że Twój projekt programistyczny stanie się źródłem frustracji i pasmem niepowodzeń. Efektem źle zbudowanego planu będzie niedotrzymywanie terminów, niedoszacowanie kosztów czy słaba jakość produktu. Projekt programistyczny wymaga takiej metodyki planowania, aby częste zmiany i wysoki stopień niepewności nie obróciły wniwecz wysiłku zespołu. Dlatego właśnie należy zastosować Agile.

Ta książka jest wyczerpującym przewodnikiem po planowaniu projektów programistycznych. Znajdziesz tu dokładne omówienie filozofii metodyk zwinnych i dowiesz się, w jaki sposób je zastosować, aby w efekcie otrzymać dobry plan. Poszczególne zagadnienia zostały zaprezentowane czytelnie i dokładnie, a przy tym zilustrowane za pomocą rzeczywistych przykładów i studiów przypadków. Dzięki opisanym w książce technikom łatwo zachowasz zwinność od początku do końca projektu. Jeśli Twój zespół stosuje którąś z licznych metod Agile, ta książka będzie nieocenionym źródłem wiedzy, niezależnie od tego, czy jesteś menedżerem, liderem, czy szeregowym programistą.

Z książki dowiesz się:

- co sprawia, że plan jest dobry, i czym jest zwinne podejście do planowania
- w jaki sposób dokonywać oceny rozmiaru i czasu trwania projektu
- jak ustalać priorytety właściwości funkcjonalnych produktu
- dlaczego warto modelować finansowy zwrot funkcjonalności produktu
- w jaki sposób tworzyć harmonogramy projektu
- jak monitorować postępy realizacji przyjętego planu

Mike Cohn jest jednym z twórców metody Scrum. Pierwszy raz zastosował ją w praktyce w 1995 roku i od tej pory ozostaje gorącym zwolennikiem Agile. Cohn ma ponad 30 lat doświadczenia w programowaniu i zarządzaniu projektami. Był dyrektorem technicznym w wielu firmach — począwszy od startupów, a skończywszy na firmach z listy Fortune 40. Jest członkiem założycielem stowarzyszenia Agile Alliance. Pisze książki, publikuje artykuły w magazynach branżowych i regularnie wygłasza referaty na konferencjach.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
WWW.SZKOLENIA.HELION.PL
INFORMATYKA W NAJLEPSZYM WYDANIU
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-4486-0
9 788328 344860
Cena: 59,00 zł
PRENTICE HALL
PEARSON EDUCATION