**Lean software development**

Programowanie odchudzone

Określenie Lean Software Development zostało zaprezentowane po raz pierwszy jako tytuł konferencji zorganizowanej przez inicjatywę ESPRIT Unii Europejskiej w Stuttgarcie, w Niemczech, w październiku 1992 r. Niezależnie w następnym roku Robert „Bob” Charette w 1993 r. zasugerował koncepcję „odchudzonego tworzenia oprogramowania” jako część jego prac dotyczących szukania lepszych sposobów zarządzania ryzykiem w projektach tworzenia oprogramowania. Termin „Chudy” datuje się na rok 1991. Został zaproponowany przez Jamesa Womacka, Daniela Jonesa i Daniela Roosa w książce The Machine That Changed the World: The Story of Lean Production, jako termin języka angielskiego, używany do określania podejścia do zarządzania w firmie Toyota. Pomysł zastosowania metodologii Lean w rozwoju oprogramowania powstał bardzo wcześnie, bo już po 1 lub 2 latach od pierwszego zastosowania jej w odniesieniu do procesów produkcji i inżynierii przemysłowej.

Womack i Jones zdefiniowali pięciu filarów metodyki odchudzonego myślenia. Były to:

* **Wartość**
* **Strumień wartości**
* **Przepływ**
* **Ciągłość**
* **Doskonałość**

Stało się to domyślną definicją roboczą odchudzonego myślenia na większą część następnej dekady. Dążenie do doskonałości, jak to zasugerowano, zostało osiągnięte poprzez wyeliminowanie odpadów. Wśród 5 filarów tylko ten piąty, polegający na dążeniu do doskonałości poprzez identyfikację systemową działań nieekonomicznych i ich eliminacji, naprawdę zyskał na popularności wśród szerokiego grona odbiorców. Metodyka odchudzona stała się prawie wyłącznie kojarzona z praktyką eliminacji strat pod koniec lat 90 i na początku 21 wieku.

Doskonałość jest osiągana poprzez ograniczenie działalności bez wartości dodanej, lecz również poprzez wygładzanie przepływu i wyeliminowanie nadmiernego obciążenia. Ponadto podejście Toyoty bazowało na fundamentalnym szacunku dla ludzi i było pod ogromnym wpływem XX-wiecznych teorii specjalistów od zapewniania jakości oraz statystycznej kontroli procesów, takich jak W. Edwards Deming.

Odchudzone wytwarzanie oprogramowania jest adaptacją Systemu Produkcyjnego Toyoty dla środowiska informatycznego i opiera się na wartościach i zasadach adaptacyjnego zarządzania projektami . Za twórców określenia „Lean software development” uważa się Mary Poppendieck i Toma Poppendiecka, którzy w swojej książce „Lean Software Development: An Agile Toolkit” przedstawili m.in. 7 głównych zasad odchudzonego zarządzania i zestaw 22 technik wspierających podejście. Niestety istnieje prawie tyle definicji programowania odchudzonego, ilu jest autorów.

Lean software development opiera się na 7 podstawowych zasadach. Są to wytyczne, które wspomagają wypracowanie praktyk pasujących do konkretnych środowisk programistycznych.

* **Eliminacja Strat**

**Stratą** lub marnotrawstwem jest wszystko, co w żaden sposób nie dodaje produktowi wartości rozumianej, jako wartość z punktu widzenia klienta ( Muda). Definicja Womack i Jones dla metodyki Lean nie jest powszechnie udostępniana. Zasady zarządzania w firmie Toyota są znacznie bardziej subtelne. Pojedynczy wyraz „odpady” jest opisany bardziej szczegółowo trzema japońskich określeniami:

* Muda — oznacza dosłownie „odpady”, ale sugeruje działania niedające wartości dodanej
* Mura — oznacza „nierównomierność” i jest interpretowane jako „zmienność przepływu”
* Muri — oznacza „nadmierne obciążenie” lub „nieracjonalność”.

W szczupłym myśleniu pojęcie **straty** jest niezwykle istotne. Jeśli program posiada niewymagane przez klienta funkcje, jest to **strata**. Stratą jest również przekazywanie pracy nad oprogramowaniem innemu zespołowi czy nadmierne testowanie. Poniżej przedstawiono **czynniki generujące straty** w przemyśle przełożone na straty w środowisku programistycznym.

|  |  |
| --- | --- |
| **Straty w przemyśle** | **Straty przy tworzeniu oprogramowania** |
| Zapasy | Gotowe części kodu |
| Nadmierne przetwarzanie | Dodatkowe procesy |
| Nadprodukcja | Niepotrzebne funkcje |
| Transport | Zmienianie zadań |
| Oczekiwanie | Oczekiwanie |
| Zbędny ruch | Zbędny ruch |
| Błędy i wady jakościowe | Błędy i wady jakościowe |

Idea **eliminacji strat** opiera się o zrozumienie potrzeb klienta i dostarczenie mu rozwiązania realizującego dokładnie te potrzeby, jak najszybciej. Każdy czynnik opóźniający pracę jest marnotrawstwem.

* **Wzmaganie procesu uczenia**

Rozwój i produkcja oprogramowania to proces wymagający ciągłego uczenia i dostosowywania do nowych wyzwań. Wspomaganie tych procesów znacznie poprawia wydajność pracy. Aby to osiągnąć można stosować krótkie cykle przyrostowe oprogramowania – w skład, których wchodzi integracja z dotychczasowym kodem i niezbędne testy, oraz konsultacje z klientem po każdym cyklu.

* **Opóźnianie decyzji**

Tworzenie oprogramowania zawsze związane jest z niepewnością i ryzykiem. Ich redukcji sprzyja podejmowanie decyzji oparte na faktach a nie spekulacjach, co w początkowych stadiach prac jest wręcz niemożliwe. Z tego powodu należy dążyć do podejmowania decyzji jak najpóźniej to możliwe posiadając już konkretne dane. Ważnym elementem realizowania tej zasady jest budowa systemu z myślą o jego otwartości na zmiany, w przeciwieństwie do dostarczania wielu niemodyfikowalnych części.

* **Jak najszybsze rezultaty**

Dostarczanie produktów jak najszybciej zapewnia, że klient otrzyma rozwiązanie którego potrzebuje, a nie którego potrzebował kiedyś. Idea opóźniania decyzji opiera się o podejmowanie szybkich decyzji bazując na dostępnej aktualnie wiedzy i wytwarzanie potrzebnej części oprogramowania. Bez szybkiego tempa dostarczania strategia traci na wartości. Pozwala ona również klientowi na bieżąco oceniać wdrożone funkcjonalności.

* **Upoważniony zespół**

W związku z tym, że decyzje podejmowane są jak najpóźniej a wykonywanie poszczególnych elementów systemu następuje szybko, nie jest możliwe pełne kontrolowanie pracowników przez kierownictwo. Podejście „odchudzone” wspiera zasadę „znajdź dobrych ludzi i pozwól im pracować”. Jej realizacja polega na przyznaniu odpowiednich uprawnień, codziennych spotkaniach, wyłapywaniu błędów oraz odstępstw od planu. Nie należy natomiast dążyć do kontrolowania każdej czynności pracownika.

* **Zapewnienie spójności**

Spójność rozumiana jest, jako zgodność oprogramowania z wymaganiami klienta oraz współdziałanie elementów systemu ze sobą. Ponadto program powinien zachowywać swoją użyteczność z biegiem czasu, a więc otwarty na modyfikacje.

* **Spojrzenie całościowe**

Zasada ma na celu zwalczanie zjawiska suboptymalizacji. Rozbicie projektu na części i osobne mierzenie postępów każdego z nich może działać negatywnie na projekt. Nadmierne skupienie się na jednym elemencie powoduje jednocześnie zaniedbanie integralności systemu i przeszkadza w postrzeganiu go jako jednego produktu.

**Praktyki**

Metodyka odchudzonego tworzenia oprogramowania nie przepisuje żadnych praktyk. Ważniejsze jest wykazanie, że rzeczywiste definicje procesów są dopasowane do zasad i wartości. Jednakże pewna liczba praktyk jest powszechnie przyjmowana. Ta sekcja zawiera krótkie omówienie niektórych z nich.

* **Zbiorcze diagramy przepływu**

Zbiorcze diagramy przepływu są standardową częścią raportowania w programie Team Foundation Server od wersji 2005. Zbiorcze diagramy przepływu to wykresy powierzchniowe łącznie wszystkich elementów pracy na każdym etapie przepływu pracy. Są bogate w informacje i mogą być użyte do uzyskania średniego czasu cyklu między krokami w procesie, jak również prędkości przepływności (zwanej "prędkością"). Różne procesy tworzenia oprogramowania generują różne wizualne podpisy na zbiorczych diagramach przepływu. Praktykujący mają szansę nauczyć się rozpoznawać wzorce dysfunkcji w procesie przedstawionym na wykresie obszaru. W prawdziwie w odchudzonym procesie będzie widać równomiernie rozłożone obszary koloru, o płynnie rosnącym nasyceniu w stałym tempie. Obraz będzie wyświetlany bez postrzępionych linii i widocznych bloków koloru.

W najprostszej postaci zbiorcze diagramy przepływu są używane do wizualizacji ilość pracy w toku na każdy etap w cyklu życia elementu pracy. Można tego używać do wykrywania wąskich gardeł i obserwowania skutków "mura" (zmienności przepływu).

* **Formanty wizualne**

Oprócz stosowania zbiorczych diagramów przepływu zespoły korzystające ze zwinnego tworzenia oprogramowania używają fizycznych tablic lub projekcji z elektronicznych systemów wizualizacji do wizualizacji pracy i obserwowania jej przepływu. Takie wizualizacje pomagają członkom zespołu obserwować prowadzone prace i umożliwiają im skupienie się na wąskim gardle oraz efektach „mura”. Formanty wizualne pozwalają również członkom zespołu na organizację pracy i współpracę bez planowania, określonych wskazówek kierowniczych ani interwencji. Te formanty wizualne są często określane jako "ściany z karteczkami" lub czasem (niepoprawnie) jako "płyty kanban".

* **Wirtualne systemy Kanban**

System kanban to praktyka przyjęta z metodyki produkcji odchudzonej. Używa systemu fizycznych kart do ograniczenia ilości pracy w toku na każdym etapie przepływu pracy. Takie systemy ograniczonego nakładu pracy w toku tworzą „wciąganie”, dzięki któremu zaczyna się nowe zadanie, gdy istnieją wolne kanban wskazujące, że nowe zadanie może zostać „wciągnięte” do określonego stanu, a praca zostanie wykonana w większym stopniu zaawansowania.

W odchudzonym wytwarzaniu oprogramowania obiekty kanban są wirtualne i często śledzone przez ustawienie maksymalnej liczby dla danego etapu w przepływie pracy dla typu elementu pracy. W niektórych implementacjach elektroniczne systemy śledzą wirtualne obiekty kanban i wysyłają sygnał, gdy można rozpocząć nową pracę. Sygnał może być wizualny lub w formie alertu, takiego jak wiadomość e-mail.

Wirtualne systemy Kanban często są łączone z formantami wizualnymi, tworząc wizualny wirtualny system Kanban reprezentujący jeden lub kilka typów elementów pracy. Takie systemy są często określane jako „karty kanban” lub „elektroniczne systemy kanban”. Wizualny system wirtualny kanban jest udostępniany jako wtyczka dla Team Foundation Server o nazwie PWT Visual[20]. Ten projekt został opracowany jako open source przez Hakana Forssa w Szwecji.

* **Małe rozmiary partii/mały przepływ pojedynczego elementu**

Metodyka odchudzonego tworzenia oprogramowania wymaga, aby praca była podejmowana w małych porcjach, często określanych jako „iteracje” lub „przyrostach”, lub aby elementy pracy przepływały niezależnie, co jest określane jako „przepływ jednoelementowy”. Przepływ jednoelementowy wymaga zaawansowanej strategii zarządzania konfiguracją, aby umożliwić wykonanej dostarczanie wykonanej pracy oraz zapobiec przypadkowemu zwalnianiu niekompletnej pracy. Zazwyczaj jest to osiągane przy użyciu strategii rozgałęzienia w systemie kontroli wersji. Za małą partię pracy zazwyczaj uważana się partię, którą może wykonać niewielki zespół maksymalnie 8 osób w terminie poniżej 2 tygodni.

Małe partie oraz przepływ pojedynczego elementu wymagają interakcji z właścicielami działalności biznesowej, aby zniwelować zaległości, kolejkę lub wykonać pracę. Wymagają one także możliwości częstych wydań. Aby włączyć częste interakcje z ludźmi biznesu i częste dostawy, należy zmniejszyć koszty transakcji i koordynacji obu działań. Powszechny sposób osiągnięcia tego to wykorzystanie automatyzacji.

* **Automatyzacja**

Metodyka odchudzonego tworzenia oprogramowania oczekuje wysokiego poziomu automatyzacji, aby ekonomicznie uzasadnić przepływ jednoelementowy i wspierać wysoką jakość oraz zmniejszenie i zmniejszenia żądań po usterkowych. Użycie automatycznego testowania zautomatyzowanego wdrażania i fabryk oprogramowania do zautomatyzowania wdrażania wzorców projektowania oraz tworzenie powtarzalnych sekcji kodu źródłowego o niskiej zmienności będzie powszechne w procesach programowania oprogramowania typu Lean.

* **Zdarzenia kaizen**

W literaturze poświęconej metodyce odchudzonej termin kaizen oznacza „ciągłą poprawę”, a zdarzenie kaizen to czynność wprowadzania zmiany w procesie lub narzędziu, która, mamy nadzieję, przyniesie poprawę.

Procesy odchudzonego tworzenia oprogramowania używają kilku różnych działań do generowania zdarzeń kaizen. Są one wymienione poniżej. Każde z tych działań jest przeznaczone do stymulowania konwersacji o problemach, które szkodliwie wpływają na możliwości i w konsekwencji na zdolność realizacji żądań. Istota kaizen w pracy wymagającej wiedzy polega na tym, że należy prowokować rozmowy na temat problemów w grupie osób z różnych zespołów i o różnych umiejętnościach.

* **Codzienne spotkanie**

Zespoły programistów, często składające się z maksymalnie 50 osób, zazwyczaj spotykają się przy wizualnym systemie kontroli, takim jak biała tablica wyświetlająca wizualizację ich aktualnej pracy. Omawiają dynamikę przepływu oraz czynniki wpływające na przepływ pracy. Szczególna uwaga jest poświęcona pracy blokowanej zewnętrznie i pracy opóźnianej przez usterki. Problemy z procesu często stają się oczywiste po serii spotkań wprowadzających. Wynik oznacza, że mniejsza grupa może pozostać po spotkaniu w celu omówienia problemu i zaproponowania rozwiązania lub zmiany procesu. Nastąpi zdarzenie kaizen. Te spontaniczne posiedzenia są nazywane często spontanicznymi kręgami jakości w starszych pozycjach literatury fachowej. Takie spontaniczne spotkania są podstawą kultury kaizen. Menedżerowie będą zachęcać do powstania zdarzeń kaizen po codziennych spotkaniach w celu promowania metodyki odchudzonej w swojej organizacji.

* **Kolekcje**

Zespoły pracujące nad projektem mogą planować regularne spotkania, aby dyskutować o postępach. Są one często wykonywane po ukończeniu produktów cząstkowych projektu lub po przyrostach programistycznych w wyznaczonym czasie, znanych jako iteracje lub sprinty w programowaniu oprogramowania Agile.

Kolekcje zazwyczaj używają podejścia anegdotycznego do odczuć, zadając pytania typu: „co poszło dobrze?”, „co mogliśmy zrobić inaczej?” czy „co powinniśmy przestać robić?". Kolekcje tworzą zazwyczaj dzienniki z sugestiami dotyczącymi zdarzeń kaizen. Zespół może następnie priorytetować niektóre z nich w celu implementacji.

* **Przeglądy operacji**

Przegląd operacji jest zazwyczaj większy niż retrospektywa i obejmuje przedstawicieli z całego strumienia wartości. Jest typowe, że nawet 12 działów prezentuje obiektywne, ilościowe dane pokazujące zgłoszony popyt i odzwierciedlające możliwość jego zaspokojenia. Przeglądy operacji są zazwyczaj przechowywane co miesiąc. Główne różnice między przeglądem działań i retrospektywą polegają na tym, że przeglądy działań dotyczą szerszego zestawu funkcji, zwykle obejmują portfolio projektów i innych działań, a ponadto korzystają z obiektywnych danych ilościowych. Kolekcje, dla porównania, zwykle obejmują zakresem pojedynczy projekt; obejmują kilka zespołów, jak zespół analizy, rozwoju czy badań; i mają ogólnie anegdotyczny charakter.

Przegląd operacji będzie powodować dyskusje na temat wzajemnej dynamiki wpływającej na funkcjonowanie zespołów. Prawdopodobnie jeden zespół generuje żądanie po usterkowe, które jest przetwarzane przez inny zespół. Może to żądanie po usterkowe jest szkodliwe i powoduje, że drugi zespół nie wywiązuje się z zobowiązań oraz nie spełnia oczekiwań? Przegląd operacji dostarcza możliwości omówienia takich problemów i zaproponowania zmian. Przeglądy operacji zazwyczaj tworzą małe zaległości zdarzeń kaizen, które mogą być ustalone jako priorytet i zaplanowane do przyszłej implementacji.

Nie istnieje nic takiego jak pojedynczy proces programowania oprogramowania Lean. Proces można uznać za odchudzony, jeżeli jest wyraźnie dopasowany do wartości i zasad metodyki odchudzonego tworzenia oprogramowania. Metodyka odchudzonego tworzenia oprogramowania nie przepisuje żadnych praktyk, ale niektóre działania stały się popularne. Organizacje stosujące metodykę odchudzoną zachęcają do stosowania metodyki kaizen przez wizualizację przepływu pracy i pracy w toku oraz zrozumienie dynamiki przepływu i czynników wpływające na niego (np wąskie gardła, brak natychmiastowej dostępności, zmienność, straty). Udoskonalenia procesu są sugerowane i uzasadnione jako sposoby zmniejszenia źródeł zmienności, wyeliminowania odpadów, poprawy przepływu lub zwiększenia wartości dostawy lub zarządzania ryzykiem. W związku z tym procesu odchudzonego tworzenia oprogramowania zawsze będą ewoluowały i specyficznie dostosowywały się do organizacji, w której ewoluują. Nie będzie naturalne po prostu skopiować definicję procesu z jednej organizacji do innej i spodziewać się, że będzie działała w innym kontekście. Jest również mało prawdopodobne, aby po powrocie do organizacji po kilka tygodniach lub miesiącach okazało się, że używany proces jest taki sam, jak zaobserwowano wcześniej. Zawsze będzie on ewoluował.

Można powiedzieć, że organizacja korzystająca z procesu rozwoju oprogramowania Lean jest Lean, jeśli wynikiem jej działania jest tylko niewielka ilość odpadów we wszystkich trzech formularzach ("mura", "muri" i "muda") i może wykazać się optymalizacją dostarczania wartości poprzez skuteczne zarządzanie ryzykiem. Dążenie do doskonałości w Lean to zawsze wyzwanie. Brak obiektu docelowego. Prawdziwie chude organizacje zawsze szukają możliwości dalszej poprawy.

Odchudzone tworzenie oprogramowania to wciąż nowa dziedzina i spodziewamy się jej ewoluowania w ciągu następnej dekady.