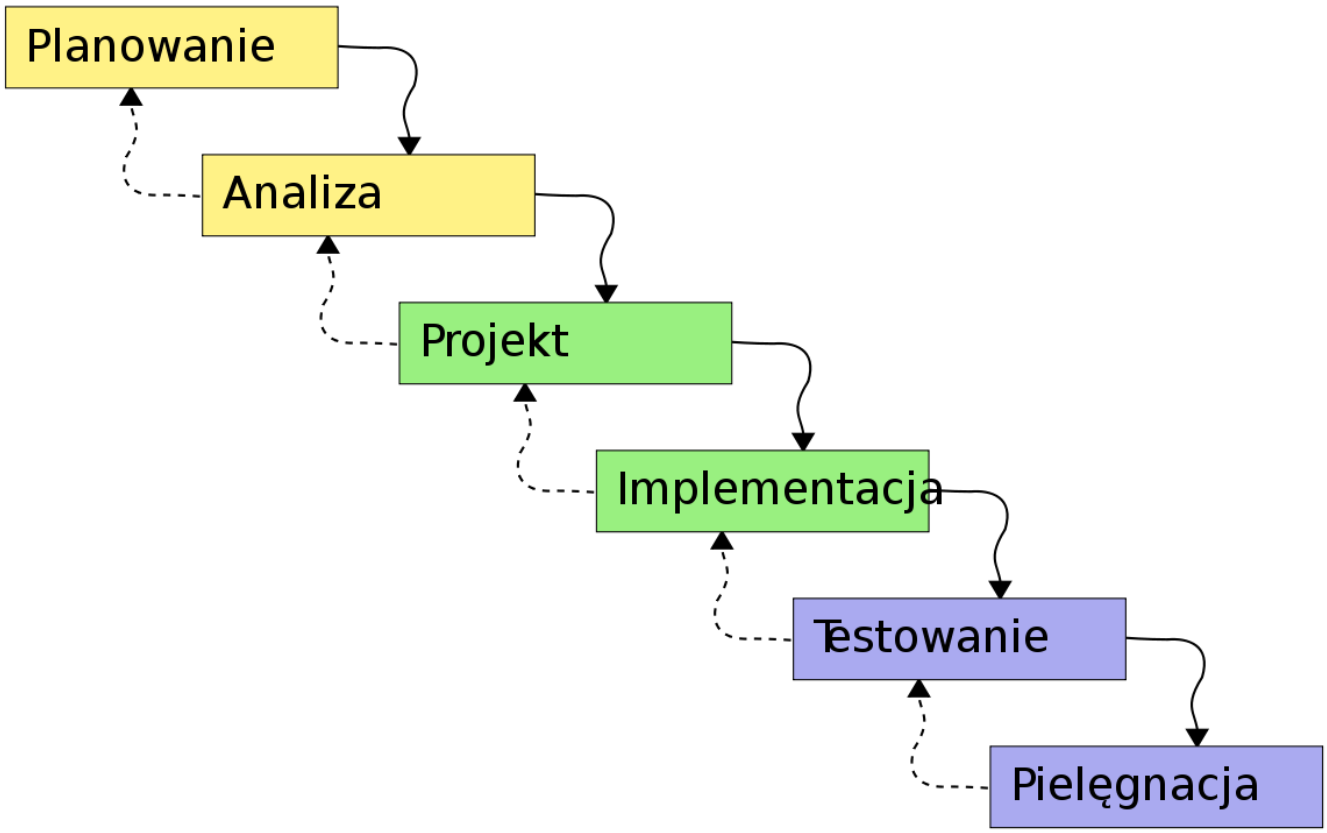
PODEJŚCIE KASKADOWE

Budowa sklepu w tym modelu jest podzielona na kilka faz, z czego każda odpowiedzialna jest za dostarczanie innego rodzaju rezultatów. Aby zbudować finalny produkt, konieczne jest przejście przez kilka etapów:

- analizy (lista wymagań do zrealizowania)
- projektowania (architektury, interfejsu użytkownika oraz konkretnych funkcjonalności)
- implementacji (na podstawie danych zebranych w fazie analizy i projektowania)
- testów (sprawdzenie, czy implementacja nie zawiera błędów)
- wdrożenia (sklep staje się dostępny dla użytkowników)

Każdy etap jest restrykcyjnie sprawdzany i weryfikowany. W przypadku jakiegokolwiek niepowodzenia zespół uczestniczący w pracach cofa się i tak długo poprawia etap, aż będzie on w pełni gotowy.



KONSEKWENCJE PODEJŚCIA KASKADOWEGO

Pomimo, iż na pierwszy rzut oka podejście to wydaje się logiczne oraz poukładane, w kontekście budowania produktów posiada bardzo dużo wad, takich jak:

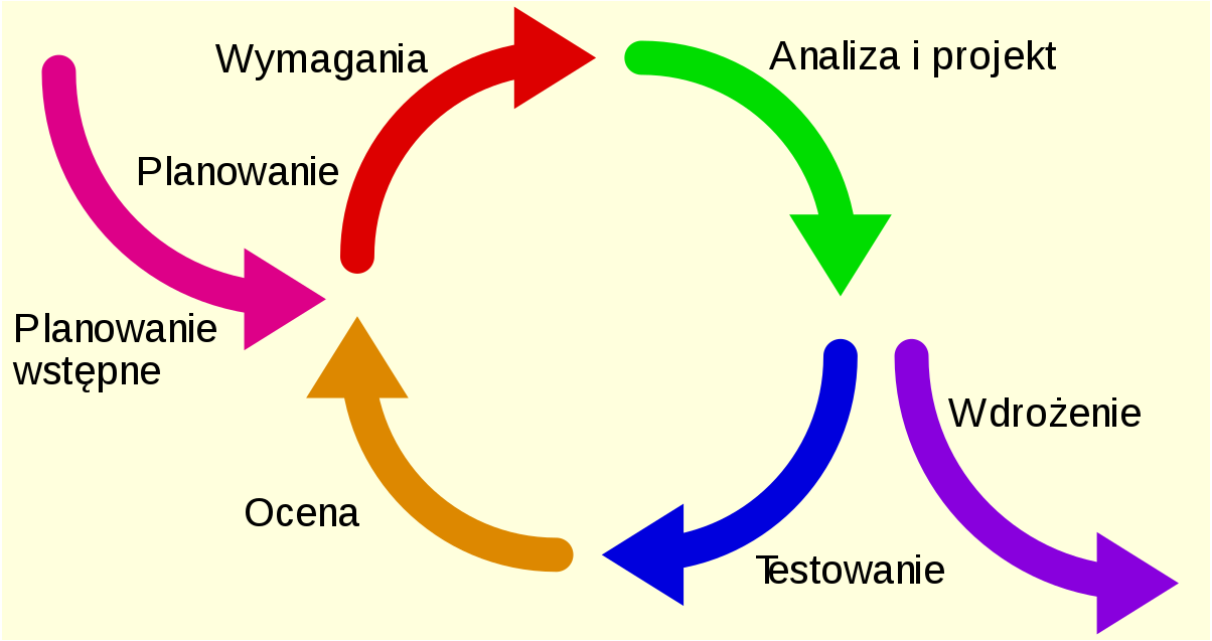
- **brak możliwości zebrania informacji zwrotnej na temat naszego produktu –** możemy wprawdzie pokazać użytkownikom efekt analizy (np. w formie dokumentu tekstowego), czy projektowania (np. makiety), jednak aby usłyszeć wartościową informację zwrotną, trzeba by pokazać chociażby "klikalną" namiastkę działającego sklepu. Taką możliwość mamy dopiero pod koniec fazy rozwoju (faza testów), co powoduje, że ryzyko zbudowania produktu, który nie spełnia oczekiwań użytkowników, jest wysokie,
- **brak możliwości otrzymania produktu wcześniej niż planowano –** wyobraźmy sobie sytuację, w której z jakiegoś powodu musimy zredukować budżet projektowy i chcielibyśmy zakończyć rozwój produktu wcześniej, zadowalając się uboższą wersją naszego sklepu. Zakładając, że fazy mają podobną długość, gdybyśmy chcieli zredukować budżet o 50%, to zatrzymamy się w okolicach środka fazy implementacji. Powoduje to, że produkt znajduje się w stanie niezdatnym do przekazania go do użytkowników końcowych (trwająca implementacja, brak testów),
- **możliwość zarabiania na produkcie dostępna dopiero po wdrożeniu –** ze względu na fakt, iż rezultatem konkretnej fazy nie jest działające oprogramowanie, a tylko pewien pół-produkt, nasz sklep pozwoli nam zarobić pierwsze pieniądze dopiero po wdrożeniu całościowego efektu, będącego sumą wszystkich zrealizowanych już faz. Musimy więc wydać cały zaplanowany budżet, aby otrzymać produkt, który jest gotowy do komercyjnego uruchomienia,
- **wysokie ryzyko techniczne –** pierwsze fazy w modelu kaskadowym są mocno teoretyczne (analiza, projektowanie), co powoduje, że pewne ryzyka mogą zmaterializować się dopiero w późniejszych, bardziej praktycznych fazach (implementacja, testowanie). Nie wszystkie problemy jesteśmy w stanie przewidzieć oraz odpowiednio obsłużyć z wyprzedzeniem, co wynika ze złożoności materii tworzenia oprogramowania,
- **wysokie ryzyka biznesowe –** walidacja założeń biznesowych staje się możliwa, gdy produkt nabierze faktycznego kształtu, co jest wykonalne dopiero po wdrożeniu. W przypadku projektów, które trwają wiele miesięcy, a czasem nawet i lat, dodatkowym problemem jest starzenie się rozwiązania, które w skrajnych przypadkach może okazać się przestarzałe i nie adekwatne do rynku, kiedy już zostanie wdrożone produkcyjnie.

Zalety to między innymi pewność otrzymania oczekiwanych wyników, stały i dokładnie określony rytm pracy oraz dokładność i rygor podczas prac.

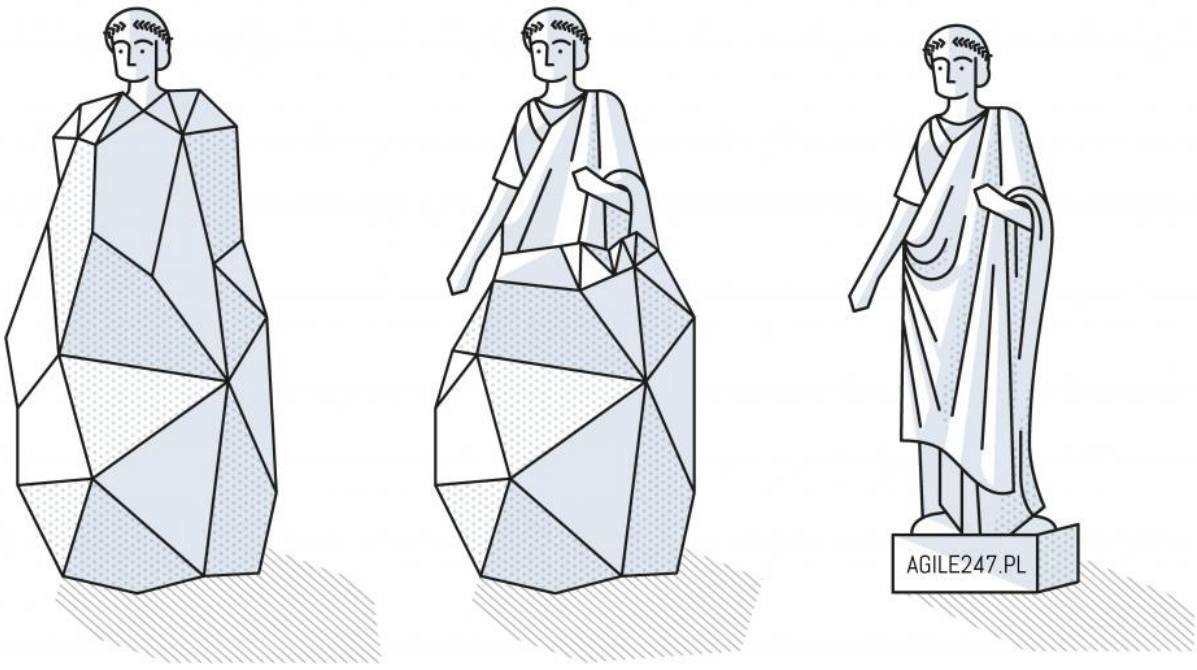
Do wad należy zaliczyć niską elastyczność, brak możliwości wprowadzania modyfikacji podczas prac a testy poprawności wykonuje się na samym końcu.

PODEJŚCIE PRZYROSTOWE

Przyrostowe wytwarzanie oprogramowania opiera się na zdecydowanie odmiennych założeniach w porównaniu do modelu kaskadowego. Podejście to wymaga posiadania **jasnej wizji produktu**, której realizacja prowadzona jest w powtarzalnych, krótkich krokach o stałej długości, trwających najczęściej 1 lub 2 tygodnie. W dużym uproszczeniu, każdy taki krok składa się z małego kawałka analizy, projektowania, właściwej implementacji oraz testów – oczywiście w bardzo okrojonym zakresie, dotyczącym konkretnej, niedużej funkcjonalności. Efektem pojedynczego kroku jest drobny, działający i skończony przyrost produktu.



Pomocna w zrozumieniu modelu przyrostowego może być analogia do pracy rzeźbiarza: gdyby pracował przyrostowo, podzieliłby swoje dzieło (np. postać boga greckiego) na wiele małych przyrostów (np. twarz, dłoń, klatka piersiowa) i rzeźbiłby każdy z nich po kolei, aż do osiągnięcia jego wyglądu ostatecznego. Raz przygotowany przyrost dzieła (np. twarz) jest już jego "ostateczną wersją" i rzeźbiarz nie wraca już do niego w kolejnych krokach realizacji.



Przykład wykonania rzeźby w modelu przyrostowym

W przypadku naszego sklepu, takim przyrostem może to być np. strona logowania lub fragment widoku szczegółów produktu. Tak przygotowany przyrost jest kompletny i zespół nie wraca już do niego, chyba że w trakcie rozwoju produktu pojawi się wiedza, która znacząco wpłynie na stworzoną już funkcjonalność.

KONSEKWENCJE PODEJŚCIA PRZYROSTOWEGO

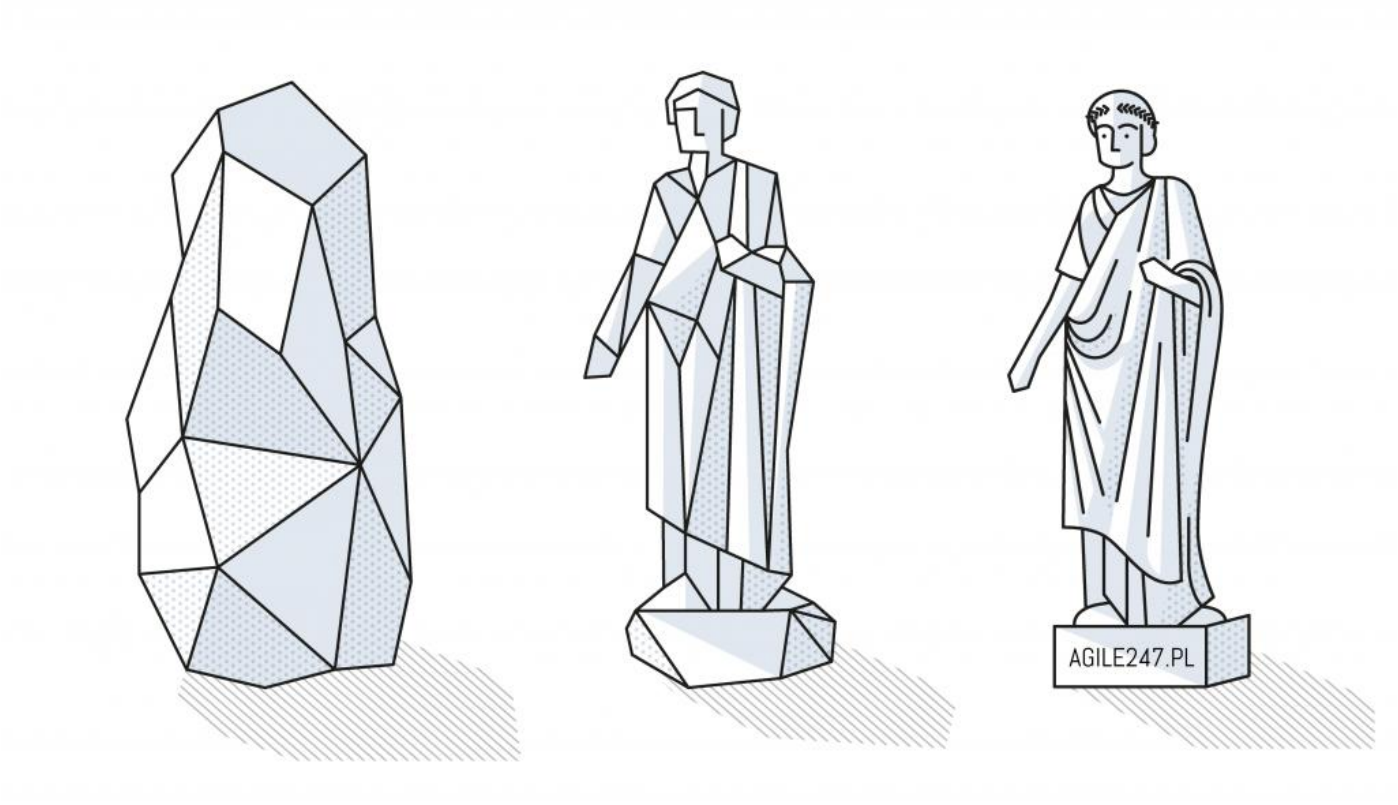
Pomimo iż model ten posiada zdecydowanie więcej zalet niż model kaskadowy, posiada pewne ograniczenia:

- **częściowa możliwość zebrania informacji zwrotnej na temat naszego produktu –** jest lepiej niż w przypadku modelu kaskadowego, ponieważ efektem 1-2 tygodniowej pracy jest działające oprogramowanie, a nie tylko efekty analizy czy projektowania. Pozwala to pokazać działający kawałek produktu interesariuszom, zaprzyjaźnionym partnerom lub użytkownikom końcowym. Co ważne, możliwość taka występuje już na wczesnym etapie rozwoju produktu. W przypadku sklepu, moglibyśmy przykładowo pokazać skończoną, gotową stronę logowania z zastrzeżeniem, że póki co to wszystko, co jest na ten moment dostępne,
- **częściowa możliwość otrzymania produktu wcześniej niż planowano –** w zależności od budowanego produktu oraz podjętych decyzji związanych z podziałem rozwoju produktu na odpowiednie kroki, możemy mieć lub nie mieć możliwość wcześniejszego otrzymania czegoś działającego. Przykładowo, jeżeli – nawiązując do przykładu naszego sklepu – wyprodukowane przyrosty produktowe to strona logowania oraz możliwość przeglądnięcia dostępnej oferty, może to oznaczać, że produkt nie jest na tyle funkcjonalny (nie można kupić produktu) aby zdecydować się na komercyjne wdrożenie aplikacji. Natomiast jeśli naszym produktem byłby edytor tekstowy (jak np. Microsoft Word), wypuszczenie aplikacji bez skomplikowanych, rzadko używanych funkcji mogłoby mieć sens,
- **częściowa możliwość wydania mniej pieniędzy, niż planowano –** pomimo, iż zawsze istnieje pewna elastyczność jeśli chodzi o zakres, to co do zasady nasz sklep musi składać się z wszystkich zaplanowanych elementów: strony logowania, strony z listą produktów oraz strony umożliwiającej wykonanie płatności i sfinalizowanie zamówienia. Jeśli chcielibyśmy zakończyć pracę wykorzystując wyłącznie część budżetu, mogłoby się okazać, że wprawdzie strona logowania i lista produktów są wykonane perfekcyjnie i są skończone, to nadal nie dokończyliśmy kluczowej strony, a mianowicie tej, która umożliwia zapłatę za zakupione produkty. Przykład jest oczywiście lekko przerysowany – dużo w tym przypadku zależy bowiem od tego jak Product Owner poukłada wraz z zespołem roadmapę produktu. Zakładam, że świadomy, doświadczony biznesowiec ma aspekty związane z kolejnością i ważnością prac pod kontrolą, pamiętajmy jednak, że każdy kiedyś zaczyna, stąd sygnalizuję to jako potencjalne ryzyko,
- **częściowa możliwość obniżenia ryzyk projektowych –** w tym przypadku jest o niebo lepiej niż w przypadku modelu kaskadowego. Mamy bowiem do czynienia z namacalnym, działającym oprogramowaniem, dostępnym po każdym kroku. Duża część ryzyk, dotychczas dostępnych na papierze, ma szansę się zmaterializować, dużą wartością jest również możliwość odkrycia ryzyk, o istnieniu których nie mieliśmy wcześniej pojęcia. W przypadku naszego sklepu, może się przykładowo okazać, że nakład prac dostosowujących wygląd sklepu do wyświetlaczy urządzeń mobilnych jest zdecydowanie większy, niż planowaliśmy. Jeżeli dowiemy się o tym na wczesnym etapie działań, mamy przestrzeń, aby odpowiednio zareagować.

PODEJŚCIE ITERACYJNE

W podejściu tym zespół posiada **ogólną wizję** produktu, która w odróżnieniu od modelu przyrostowego, nie musi być bardzo szczegółowa oraz dopracowana. Pracuje na zasadzie "od ogółu do szczegółu": w pierwszym kroku powstaje bardzo ogólny szkielet całego produktu, który w kolejnych krokach jest doprowadzany do pożądanego poziomu szczegółowości. W trakcie pracy zespół wielokrotnie wraca do wcześniej napisanych fragmentów i stopniowo je rozbudowuje, dodając kolejne detale oraz funkcjonalności. To, co moim zdaniem wyróżnia to podejście i jest jednocześnie największym wyzwaniem, to całościowe myślenie o produkcie.

Używając analogii wcześniej wspomnianego artysty rzeźbiarza: gdyby pracował iteracyjnie, to w pierwszym kroku wyrzeźbiłby bardzo ogólną, surową i pozbawioną detali postać boga greckiego. W każdym kolejnym kroku dopracowywałby i ubogacał całościowo swoje dzieło dodając detale, aż do uzyskania satysfakcjonującego efektu. Każdy element rzeźby (np. twarz, dłoń czy klatka piersiowa) po każdym kroku wygląda nieco lepiej i zbliża artystę do stanu docelowego.



Przykład wykonania rzeźby w modelu iteracyjnym

Wracając do przykładu naszego sklepu internetowego, w podejściu iteracyjnym chciałbym umożliwić zakup produktu **już po pierwszym kroku**. Aby do tego doprowadzić, zbudowałbym bardzo mocno uproszczoną wersję sklepu, która:

- nie wymagałaby od użytkownika zalogowania się,
- lista oferowanych produktów byłaby prostą tabelką (produkt + cena) bez żadnych dodatkowych funkcji,
- zakup byłby możliwy tylko gotówką podczas odbioru osobistego.

Następne kroki ubogacałyby mój sklep o kolejne funkcjonalności, takie jak możliwość zalogowania zwykłego lub przy użyciu Facebook'a czy integrację z serwisami umożliwiającymi dokonanie płatności takimi jak PayPal czy PayU.

KONSEKWENCJE PODEJŚCIA ITERACYJNEGO

Pomimo, iż jest to dosyć wymagające podejście, charakteryzuje się wieloma pozytywnymi cechami, których trudno szukać w innych podejściach.

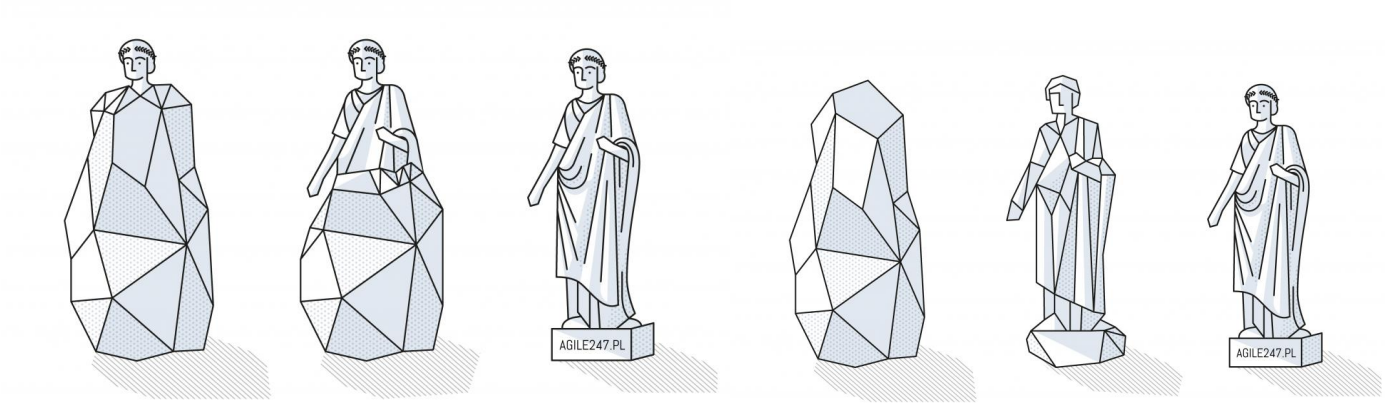
- **możliwość zebrania informacji zwrotnej na temat naszego produktu –** właściwie co każdy krok jesteśmy w stanie otrzymać feedback dotyczący realizowanej inicjatywy. O ile w modelu przyrostowym jesteśmy w stanie co krok pokazać skończony, gotowy fragment produktu, tak w modeli iteracyjnym pokazujemy z kolei całościowy produkt, zbudowany jednak w dużym uproszczeniu. To, co zyskujemy, to możliwość otrzymania informacji zwrotnej na temat tego, jaki jest ogólny odbiór użytkowników, oraz wiedza, czy interakcje wewnątrz produktu są logiczne i zrozumiałe dla odbiorców. Każda kolejna iteracja jest zatem cenną lekcją dla zespołu z zakresu budowania rozumienia potrzeb użytkowników.
- **możliwość otrzymania produktu wcześniej niż planowano –** Leonardo Da Vinci powiedział, że "wielkie dzieła nigdy nie są kończone, tylko porzucane". Ostatecznie, wyłącznie artyści tworzący konkretne dzieło wiedzą, z czego zrezygnowali oraz co nie zostało zrobione, a nadal z perspektywy odbiorców to dzieło może być odbierane jako finalne i kompletne. Dzięki stosowaniu podejścia iteracyjnego mamy podobną jak artyści możliwość "porzucenia" inicjatywy niemal w dowolnym momencie. Wynika to z faktu, że od początku budujemy kompletne rozwiązanie (ang. end-to-end), czyli pewną uproszczoną ścieżkę podróży użytkownika, najczęściej bez przypadków brzegowych, zapewniającą jednak realizację celu, dla którego korzysta się z aplikacji. Z każdym krokiem rozwoju dodajemy kolejne elementy, funkcje oraz detale, które wpływają na całościową jakość rozwiązania. Operując na przykładzie sklepu internetowego, możemy zdecydować się udostępnić produkt użytkownikom bez możliwości logowania przy użyciu Facebooka czy też oferując wyłącznie jeden sposób płatności za produkt.
- **możliwość niewykorzystania całego budżetu –** korzyść ta jest powiązana z poprzednim punktem: jeżeli możemy udostępnić użytkownikom produkt wcześniej, najprawdopodobniej będziemy w stanie oszczędzić pieniądze i zagospodarować je w inny sposób. Pomocny może być tutaj popularny wskaźnik ROI (ang. Return on Investment) – jeżeli kolejna iteracja zespołu nad produktem ma nas kosztować kilkadziesiąt tysięcy złotych a jej efektem miały by być mało znaczące funkcje, warto zastanowić się, czy jest sens wydawać na to pieniądze.
- **wczesne obniżenie ryzyk projektowych –** całościowe, iteracyjne myślenie o rozwoju aplikacji powoduje, że bardzo wcześnie dotykamy różnych obszarów aplikacji, co ma wpływ na fakt, że bardzo szybko jesteśmy rozpoznać ryzyka i ocenić, jaki mogą mieć wpływ na rozwój produktu. Przykładowo, wczesne rozpoznanie tematu integracji z API dostawcy płatności w naszym sklepie może dostarczyć nam informacji (np. o bardzo słabej jakość dokumentacji API dostawcy), które mogą mieć wpływ na dalsze plany produktowe. Co ważne, jest szansa, że dowiemy się tego na początku rozwoju produktu, kiedy jeszcze możemy tę informację uwzględnić w naszych planach. Jeśli integrację z płatnościami zostawiliśmy na końcową fazę – możemy mieć kłopoty.

Z KTÓREGO MODELU ZATEM KORZYSTAĆ?

W dużym skrócie – z obu, w zależności od potrzeb oraz preferencji. Ogranicza nas wyłącznie potrzeba oraz wyobraźnia.

Przykładowo, po kilku krokach rozwoju produktu wykonanych w modelu iteracyjnym, można skupić się wyłącznie na wybranym fragmencie i doprowadzić go do perfekcji. W przypadku naszego sklepu, może to być decyzja o skupieniu się na idealnym dopracowaniu strony płatności, zostawiając na dalszym planie rozwój pozostałych elementów sklepu.

Z drugiej strony wyobrażam sobie scenariusz, w którym po kilku krokach przyrostowych, decydujemy się na dalsze rozwijanie iteracyjne. Czyli mając idealnie przygotowaną stronę logowania oraz szczegółów produktu, w kolejnych krokach rozwijamy symultanicznie całość sklepu, a efektem każdego kroku jest po prostu trochę lepsza wersja produktu.



Zadania

Zadanie 1: Porównaj i skontrastuj metodykę kaskadową i przyrostową pod kątem ich zalet i wad. Następnie przedstaw przykład projektu, który lepiej nadaje się do jednej z tych metodyk i uzasadnij dlaczego.

Zadanie 2: Zidentyfikuj główne różnice pomiędzy podejściem iteracyjnym a podejściem przyrostowym podczas realizacji projektu oprogramowania. Następnie przedstaw, jakie są główne wyzwania, z którymi może się spotkać zespół programistów podczas pracy w obu tych metodykach.