Problemy Zarządzania, vol. 10, nr 3 (38): 25–40
ISSN 1644-9584, © Wydział Zarządzania UW
DOI 10.7172.1644-9584.38.2

Kryteria wyboru metod zarządzania projektami informatycznymi

Witold Chmielarz

Zasadniczym celem niniejszego artykułu jest identyfikacja metod zarządzania projektami informatycznymi wśród tradycyjnych i nowoczesnych. W ostatnich latach podejście do zarządzania projektami systemów informatycznych ulega szybkim i wielokierunkowym przemianom. Klasyczna teoria projektowania i obecna pragmatyka coraz bardziej się rozmijają. Wydaje się, że zastosowanie lekkich metod projektowania przynajmniej częściowo minimalizuje powstające różnice. Przeprowadzona analiza stara się wyjaśnić zaistniałe w tym zakresie wątpliwości oraz pokazuje, jak wybrać właściwą metodę dla określonego przypadku zarządzania projektem.

1. Wprowadzenie

Zarządzanie projektami informatycznymi jest istotną, wiodącą częścią zarządzania projektami i najczęściej utożsamiane jest z analizą i projektowaniem systemów informatycznych oraz tzw. inżynierią oprogramowania. Kategorią najszerszą – z teoretycznego punktu widzenia – jest tu zarządzanie projektami utożsamiane z określoną dziedziną naukową, opartą na rozwiązaniach teoretycznych problemów praktycznych wynikających „z konieczności zaspokojenia zanalizowanych wymogów zleceniodawcy przy pomocy dostępnych umiejętności, wiedzy, metod, technik i narzędzi realizacyjnych" (Mingus 2009: 21).

Jest to więc nauka o skutecznym osiąganiu zakładanych celów przy pomocy racjonalnego użycia zasobów (ludzkich, finansowych, materialnych itp. i relacji między nimi) w zakładanym czasie. Podmiotem zarządzania projektami jest niewątpliwie projekt, który może dotyczyć unikalnych przedsięwzięć realizowanych w różnych branżach, sektorach, w różnym czasie, zakresie, za pomocą różnych środków czy zespołów, w określonej strukturze organizacyjnej, instytucjonalnej itd. Na rozwój tej dziedziny naukowej największy jednak wpływ miały projekty informatyczne – ze względu na ich wielkość, złożoność, innowacyjność, ryzyko, wysokie koszty realizacji i związaną z tym konieczność zarządzania nie tylko wymienionymi elementami, ale również relacjami pomiędzy nimi. Zarządzanie projektami informatycznymi jest pojęciem najszerszym, obejmującym wszystkie aspekty

tworzenia, wdrożenia i wykorzystania systemów informacyjnych wspomagających zarządzanie, zarówno od strony praktycznej, jak i teoretycznej. Inżynieria systemów informatycznych koncentruje się na technologicznych aspektach tych procesów. Analiza i projektowanie systemów informatycznych – złożony proces organizacyjny – skupiony jest na pierwszych, głównie informacyjnych, ale kto wie, czy nie najważniejszych fazach cyklu życia systemu informatycznego. Cykl życia systemu informatycznego jest tu, jak się wydaje, kategorią kluczową, występującą we wszystkich rozpatrywanych obszarach metodycznych.

Cykl życia systemu informatycznego jest w nich traktowany jako sekwencja etapów (faz) czynności wzorowanych na życiu organizmów, którego ostatecznym celem jest budowa, wdrożenie, modyfikacje i wycofanie (lub zastąpienie) systemu informatycznego, począwszy od pierwszego pomysłu na ten temat (narodziny) po jego likwidację (śmierć). Idea cyklu życia (projektu, systemu, procesów inżynierii) jest w tych wszystkich koncepcjach tożsama, różnice występują w liczbie i nazewnictwie etapów (faz), ich kolejności logicznej lub zrównolegleniu i następstwie oraz relacjach z otoczeniem, szczególnie z użytkownikiem finalnym. Różnice te nasiliły się w ostatnim dziesięcioleciu, wraz z momentem rozpoczęcia szerokiego stosowania tzw. metodyk lekkich (zwinnych – *agile*). Identyfikacja tych różnic wraz z wynikającymi z nich konsekwencjami jest głównym celem niniejszego artykułu.

2. Ogólna charakterystyka metod ciężkich (tradycyjnych) projektowania systemów informatycznych

W cyklu życia systemu informatycznego wyróżnia się przeważnie poszczególne etapy (fazy), z których część może występować sekwencyjnie lub równolegle, jednokrotnie bądź wielokrotnie. W tworzeniu modeli cyklu życia projektu wyróżnia się metodyki:

– ciężkie (klasyczne, tradycyjne), takie jak kaskadowa (liniowa), przyrostowa, ewolucyjna, baz danych, prototypowa, spiralna;
– lekkie (nowoczesne, zwinne – *agile*), takie jak XP (eXtreme Programming), Scrum, Feature Driven Development (FDD), Dynamic System Development Method (DSDM) czy Adaptive Software Development (ASD)[1].

W *kanonicznej* (podstawowej, funkcjonującej jako punkt odniesienia) wersji projektowania systemów informatycznych cykl życia składa się z następujących etapów[2]:

– inicjacji, wstępnego rozpoznania systemu (wykonalności, strategia rozwoju organizacji, identyfikacja systemu, analiza możliwości informatyzacji, postawienie problemu) – polegającego na: identyfikacji celu, problemów, stanu obecnego i perspektyw (w tym ryzyka wykonalności) rozwoju systemu informacyjnego lub jego zmian;

– analizy informacyjnej systemu – na którą składa się identyfikacja otoczenia, łączności z otoczeniem, podsystemów, składowych systemu oraz obecnych i przyszłych warunków jego działania;
– projektowania (technicznego) systemu – koncentrującego się na uszczegółowieniu założeń analitycznych, stworzeniu modelu logicznego i fizycznego przyszłego systemu lub ewentualnych zmian w funkcjonowaniu obecnego systemu;
– oprogramowania systemu (kodowania, implementacji) – polegającego na oprogramowaniu modelu fizycznego, konstrukcji programu lub zestawu współpracujących ze sobą programów (systemu informatycznego);
– testowania – czyli sprawdzenia pod względem semantycznym, technicznym i merytorycznym prawidłowości funkcjonowania systemu;
– wdrożenia (zastosowania, aplikacji) – tj. uruchomienia (instalacji) i zainstalowania systemu u użytkownika końcowego, załadowania systemu parametrami niezbędnymi do jego funkcjonowania, podłączenia do aktualnych źródeł danych, ostatecznego szkolenia użytkownika końcowego;
– eksploatacji (realizacji) – sprowadzającej się do uruchomienia użytkowego, monitoringu, oceny i modyfikacji systemu;
– wycofania (zakończenia działalności) systemu – po badaniu opłacalności opracowania procedury przejścia na nowy system informatyczny zachowujący maksymalnie możliwą ilość pozytywnych cech starego systemu.

Etapy i fazy modelu kanonicznego występują w różnych postaciach, wariantach i rozwinięciach zarówno w metodach tradycyjnych, jak i nowoczesnych. Podstawowa równica pomiędzy nimi polega – zgodnie z jednym podejściem (Chmielarz 2010: 359–402) – w zasadzie na tym, że:
– w metodach tradycyjnych realizowane są kolejne etapy cyklu życia, w tym planowanie funkcjonalności (zakresu) wraz z niezbędnymi dla nich zasobami (materiałowymi, ludzkimi, informacyjnymi) oraz przypisanym im budżetem;
– w metodach nowoczesnych następuje koncentracja na funkcjonalnościach, w trakcie realizacji których na bieżąco ustalane są (na ogół w porozumieniu z użytkownikiem) zasoby i budżet niezbędne do ich realizacji.

Inni autorzy twierdzą (Mannaro 2008), że różnice te wyrażają się w podejściu przez metodyki tradycyjne i nowoczesne do problematyki zarządzania zmianą. W metodyce tradycyjnej wyraża się ono (na ogół) przyjęciem ukrytego założenia, że możliwe jest osiągnięcia założonego celu w trakcie jednego podejścia, którego parametry są ustalane a priori na początku cyklu. Dlatego metodyka ta jest najlepsza w tych projektach, które są stabilne, tzn. takie, w których nie są przewidywane zmiany wymagań czy technologii w trakcie realizacji projektu. Alternatywnie, nowoczesne metodyki są najbardziej odpowiednie dla projektów, w których przewiduje się wiele zmian podczas realizacji. Nie skupiają się one na rozpoznaniu wymagań i planowaniu działań, głownie na początku projektu dla całego jego przebiegu, ale na rozpoznaniu wymagań i potrzebnych zasobów w trakcie realizacji zadań szczegółowych.

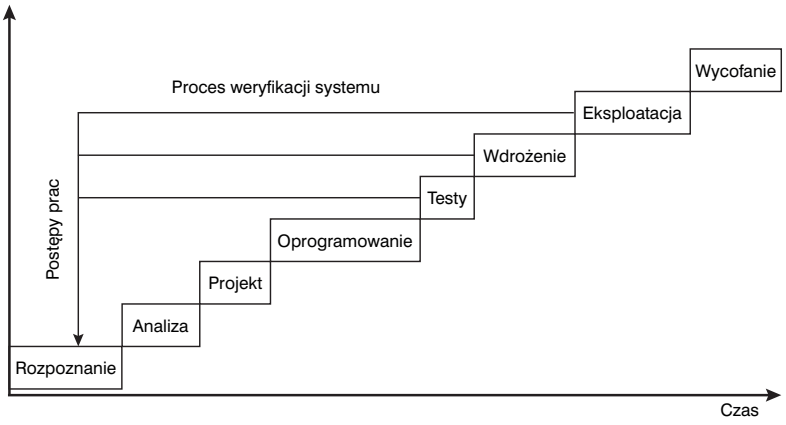
Niemniej jednak kanon postaci cyklu życia projektowania systemów informatycznych wywodzi się z historycznego rozwoju metod zarządzania projektem informatycznym (szkół strukturalnych, operacyjnych, obiektowych i społecznych) z jednej strony i rozwoju technologii systemów z drugiej. Były to procesy długotrwałe, dynamiczne i radykalne – dlatego w teorii lepiej i bardziej szczegółowo są zaprezentowane i opisane tradycyjne metody projektowania.

Najbardziej znanym, najczęściej używanym i najbardziej charakterystycznym z modeli postępowania w metodach tradycyjnych był *model kaskadowy* (rysunek 1). Przyjmowano w nim następujące założenia:
- na początku każdego projektu istnieje stabilny zestaw potrzeb i wymagań informacyjnych użytkownika i celów, do których on dąży;
- potrzeby nie zmieniają się w trakcie życia systemu;
- proces budowy systemu odbywa się stopniowo, dopiero po skończeniu jednego etapu zaczynany jest następny;
- każdy kolejny etap oznacza uszczegółowienie i przybliżenie do rzeczywistości;
- powoduje to powrót do poprzednich etapów w momencie, gdy zostanie wykryty błąd i powstanie konieczność jego korekty.

Niestety w rzeczywistości już założenie pierwsze na ogół się nie sprawdza (użytkownicy nie mają sprecyzowanych na wstępie potrzeb), może więc szybko nastąpić rozminięcie się z potrzebami użytkownika. Ponadto jest to bardzo droga kombinacja, ponieważ w przypadku realizacji przez firmę informatyczną tylko jednego projektu w danym czasie oznacza konieczność utrzymywania ekipy realizacyjnej częściowo bezrobotnej podczas realizacji następnych etapów. Ponadto w rzeczywistości często występuje nakładanie się poszczególnych etapów na siebie. W tej sytuacji istnieją liczne modyfikacje tego podejścia.

Model kaskadowy nie jest jedynym, a tylko najprostszym modelem realizacji cyklu życia projektu. Nie uwzględnia wielu problemów, np. zarządzania ryzykiem. Jego zaletą, dzięki której jest lubiany i akceptowany przez użytkowników, jest prostota rozwiązań i fakt, że stał się podstawą innych metodyk, np. ewolucyjnej czy przyrostowej.

Założenia *modelu ewolucyjnego* były następujące:
- cały system jest dzielony na moduły;
- każdy z nich odbywa przejście przez kolejne fazy cyklu budowy systemu;
- na końcu działań projektowych przystępuje się do specjalnego etapu polegającego na integracji całego systemu i przeprowadzeniu testów;
- w systemie podzielonym na części, których realizacja jest przesunięta w czasie, łatwiej nadążać za zmieniającym się celem działania;
- ponieważ każdy moduł stanowi początkowo organicznie odrębną część, należy zwrócić uwagę na niebezpieczeństwo związane z koniecznością integracji modułów w całość (bardzo często staje się to główną przyczyną niepowodzeń realizacji projektu).



Rys. 1. Schemat ideowy modelu kaskadowego. Źródło: opracowanie własne.

Jest to rozwiązanie tańsze w realizacji od poprzedniego, ponieważ może – po przesunięciu realizacji poszczególnych modułów w czasie – być obsługiwane przez mniejszą liczbę personelu. Kiedy analitycy kończą pracę nad jednym modułem, przechodzą do następnego, a ich miejsce zajmują programiści. Model ewolucyjny niestety nie zawsze spełniał pokładane w nim nadzieje. Poszczególne podsystemy – dla dużych projektów budowane przez różne zespoły – nawet pomimo tworzenia specjalnych procedur integracyjnych, nie zawsze w pełni dawały się zintegrować ze względu np. na różną wizualizację mechanizmów funkcjonowania. Brakowało w nich wyraźnie spójnych założeń dla całego systemu informatycznego. Dlatego właśnie model kaskadowy ewoluował dalej – do stworzenia modelu przyrostowego.

Model przyrostowy charakteryzował się następującymi założeniami.

– przeprowadzane jest rozpoznanie i analiza dla całości systemu, dzięki której powstaje całościowa koncepcja wstępna systemu, poparta ogólną analizą całego systemu;
– następnie system podzielony na moduły realizacyjne jest projektowany, programowany i testowany kolejno dla każdego z nich przez dziedzinowe zespoły robocze wykonujące projekty techniczne dla każdego modułu i je testujące;
– spójność systemu zapewniają pierwotne założenia systemu oraz wspólne końcowe etapy instalacji i wdrożenia, w których przeprowadzana jest też pełna integracja systemu.

Jest to procedura, podobnie jak poprzednia, najlepiej sprawdzająca się przy ograniczonych środkach, którymi dysponuje zespół realizujący system. Każdy z podsystemów może być realizowany oddzielnie, cały zespół pracuje jedynie na początku, podczas tworzenia całościowej koncepcji rozwiązania systemu, oraz na końcu, w trakcie integracji poszczególnych podsystemów

w jedną organiczną całość. Procedura przyrostowa jest lepsza od ewolucyjnej, ponieważ mechanizmy integracyjne działają zarówno na początku, jak i na końcu procesu tworzenia systemu. Na początku cyklu, oprócz rozpoznania i analizy, tworzone są także wspólne normatywy i zasady wykonania projektu każdego modułu. Ułatwia to późniejszą integrację na etapach końcowych. W cyklu ewolucyjnym oraz przyrostowym można też w ograniczony sposób reagować na zmiany wymagań użytkownika.

Zupełnie odrębną systematykę zakłada *model tworzenia struktury baz danych* (Freya). W cyklu życia opartego na tworzeniu struktury bazy danych wyróżnia się na ogół następujące fazy:

- rozpoznania i analizy – gdzie następuje rozpoznanie i zebranie potrzeb informacyjnych użytkowników;
- projektu technicznego, składającego się z dwóch etapów, dotyczących tworzenia projektu logicznego, czyli opisu modelu danych i przyszłych procesów w systemie, oraz projektu fizycznego, czyli projektu struktury bazy danych (zbiorów i relacji pomiędzy nimi), wzorców dokumentów, technologii przetwarzania, specyfikacji wewnętrznych itp.;
- oprogramowania i testowania – polegającego na stworzeniu bazy danych i oprogramowanie zastosowań (aplikacji) opartych na danych zawartych w bazie danych oraz testowaniu oprogramowania;
- wdrożenia – zainstalowania oprogramowania w określonej platformie i konfiguracji sprzętowej i wprowadzenia parametrów określonych przez użytkownika w celu kastomizacji i przygotowania systemu do eksploatacji;
- eksploatacji użytkowej i kontroli – polegającej na roboczym użytkowaniu systemu oraz zapewnieniu poprawności z ustalonymi normatywami i wymogami użytkownika;
- potencjalnych modyfikacji i adaptacji – czyli udoskonalenia funkcjonowania w wyniku pojawienia się nowych potrzeb; w razie potrzeby powrót do etapów początkowych.

Zasadnicza idea tego cyklu życia polega na stworzeniu na początku projektu struktury bazy danych systemu i oprogramowania bazy danych, co umożliwia rejestrację podstawowych faktów z życia organizacji, a następnie obudowanie bazy danych oprogramowaniem aplikacyjnym, które wykorzystuje dla różnych aplikacji te same dane zawarte w bazie danych do obsługi sprzedaży, dostaw, kontaktów z kontrahentami itp. W ten sposób powstawało i jest wykorzystywanych do dziś wiele złożonych systemów opartych na bazie danych. Jest to jednak metoda, gdzie użytkownik końcowy rezultaty działania projektanta i programisty ogląda dopiero po uruchomieniu bazy danych oraz pierwszych aplikacji. Dopiero wtedy może też sprawdzić ich zgodność z założeniami ustalanymi z zespołem projektowym. Znacznie szybciej uwidacznia się to podczas zastosowania modelu prototypowego.

Istotą *modelu prototypowego* jest oprogramowanie na początku działań projektowych – schematu działania systemu, najlepiej razem z przyszłym użytkownikiem. Cel takiego postępowania jest następujący:

– redukcja czasu oczekiwania na rezultaty programowe i wykazanie się wobec użytkownika końcowego uchwytnymi (dla niego) efektami realizacyjnymi;
– szybkie „sprzężenie" użytkownika z zespołem projektowym;
– ograniczenie liczby błędów oraz iteracyjnych (potencjalnych) rozmów z użytkownikiem, np. na temat wizualizacji systemu;
– większe zaangażowanie użytkownika w analizę i projekt.

Model ten składa się z następujących etapów:
– rozpoznania i analizy potrzeb użytkownika;
– projektu technicznego wraz z generacją oprogramowania – w trakcie tego etapu powstaje prototyp, który następnie jest sukcesywnie poprawiany w interakcji z użytkownikiem;
– po testach kolejnych wersji prototypu i jego udoskonaleniach oraz instalacji system jest przekształcany w wersję ostateczną, która przekazywana jest do eksploatacji i – w miarę potrzeb – modyfikowana z użyciem narzędzia, za pomocą którego została stworzona.

Sprawna realizacji systemu prototypowego wymaga zastosowania dobrego oprogramowania klasy CASE (Computer Aided System Engineering), co niestety powoduje, że jego bezpośrednie i pełne zastosowanie bywa ograniczone poprzez własności tego narzędzia. A dodatkowo zwiększa koszty projektu. Dlatego, pomimo niewątpliwych zalet tego podejścia, wielu projektantów i programistów woli stosować jedno z podejść klasycznych.

Najbardziej rozwiniętą formą tradycyjnego modelu cyklu życia jest natomiast *model spiralny*. Fazy realizacji modelu spiralnego są następujące:
– ustalenie celów działania – dotyczy definicji konkretnego celu systemu oraz uzasadniania koncepcji i ustalania planów jego realizacji; budowa modelu rozpoczyna się od inicjacji i definiowania projektu (ustalenia wstępnych wymagań), a następnie polega na analizie wstępnej ryzyka realizacji projektu; na tej podstawie buduje się pierwszy prototyp i tworzy konceptualny plan (harmonogram) całości projektu;
– rozpoznanie i redukcja zagrożeń – po kolejnej fazie analizy ryzyka pierwszego prototypu (identyfikacja najważniejszych zagrożeń, źródeł i sposobów zapobiegania zagrożeniom) oraz zbadania ewentualnych alternatyw budowany jest następny prototyp projektu i tworzy się wymagania dotyczące ograniczeń i zasobów projektu;
– tworzenie i zatwierdzanie – w tej fazie następuje proces rozwoju kolejnej części produktu, oparty na najbardziej odpowiednim modelu, wybranym na podstawie oceny zagrożeń; kolejno powstaje plan realizacji projektu i odbywa się kolejny etap zakończony planem całości projektu;
– ocena i planowanie – to faza, w której recenzowany jest postęp prac oraz planowana jest kolejna bądź ostateczna faza projektu; ostatni obieg cyklu przynosi projekt szczegółowy, testy i realizację projektu oraz jego zamknięcie.

Model ten różni się od pozostałych wielokrotnym powtarzaniem poszczególnych faz podstawowego modelu cyklu życia oraz – po każdym „podcyklu"

– następuje dalsze uszczegółowienie modelu oraz analiza ryzyka zakończonej sukcesem wykonalności całego systemu w danym momencie rozwoju. Jest to najbardziej skomplikowany i najdroższy w realizacji model cyklu życia systemu informatycznego, stosowany do rozwiązywania najbardziej skomplikowanych i trudnych problemów tworzenia systemów, np. militarnych. Tak jak inne modele klasyczne oparte na prototypowaniu i analizie ryzyka najbardziej zbliżony jest do podejścia nowoczesnego.

Grupa metodyk tradycyjnych istnieje od dawna. Reżim, który nakładają na rozwój projektu, dyscyplinuje w pewnym sensie sposób postępowania w trakcie realizacji projektu. Nie daje to jednak gwarancji, że projekt zakończy się sukcesem. Metody te są tak bardzo „sztywne" i ustrukturyzowane, że przestrzeganie wszystkich kroków, formuł i procedur znacząco spowalnia cały proces rozwoju projektu. Charakteryzują się one następującymi cechami:

– *Przewidywalne i powtarzalne podeście do procesu rozwoju projektu*. W klasycznych metodykach zakłada się zwiększającą się szczegółowość w miarę realizacji poszczególnych faz i etapów rozwoju projektu i obejmowanie całego okresu od początku do końca projektu. Do analiz przeprowadzanych na bardzo niskim stopniu abstrakcji stosuje się na ogół deterministyczne techniki szczegółowe. Wynik uzyskany za ich pomocą jest prawdziwy dopóki przynajmniej jedno z założeń początkowych nie zostanie zmienione. Zakłada się więc deterministyczny, niezmieniany, nieadaptowany i mało elastyczny harmonogram, budżet i zasoby oraz budowany na tej podstawie pełen podział zadań dla każdego zespołu tworzącego produkt lub usługę.
– *Obszerna dokumentacja*. Tradycyjne podeście do realizacji projektu zakłada, że dokumentacja jest tworzona po każdej fazie, każdym etapie cyklu życia projektu. W najbardziej konwencjonalnej formie modelu kaskadowego przyjmuje się, że zakres oraz wymagania dotyczące użytkownika są zbierane i ustalane w pierwszej fazie cyklu, a następnie na ich podstawie tworzony jest produkt lub realizowana usługa. Nie zmienia się tych założeń do końca cyklu życia projektu; część zebranych i przekształconych w zalecenia wykonawcze informacji i dokumentów wymaga przez to zmian w trakcie realizacji projektu.
– *Zorientowanie na proces*. Celem klasycznych metodyk jest w zasadzie określenie procesu czy procesów, które będą uniwersalne i powtarzalne, czyli będą funkcjonowały w prawidłowy sposób i będą przydatne dla każdego, kto będzie ich używał, w każdej sytuacji, w której wydadzą się przydatne. Każdy proces składający się z zadań/czynności powinien być wykonywany według określonych z góry procedur przez określoną, przypisaną do niego i odpowiedzialną za niego grupę pracowników/wykonawcę.
– *Zorientowanie na narzędzia i techniki wspomagające realizację.* Do wykonania każdego określonego w projekcie zadania powinny być dostarczone odpowiednie narzędzia wspomagające zarządzania.

3. Przegląd lekkich (nowoczesnych) metod projektowania systemów informatycznych

Na drugim biegunie znajdują się metody lekkie (zwinne – *agile*) projektowania. Ich zasadniczym celem jest maksymalne zwiększenie wartości produktu w porównaniu z jego aktualnym stanem, przy założonych kosztach i w określonym czasie. W *Manifeście rozwoju zwinnego oprogramowania* (Beck i in. 2010) zwraca się uwagę na konieczność cenienia sobie bardziej ludzi i interakcji pomiędzy nimi niż procesów i narzędzi, działający rezultat projektu ponad obszerną dokumentację, współpracę z klientem ponad formalne ustalenia oraz reagowanie na zmiany ponad podążanie za planem. Odpowiada to postulatom wymienionych dotychczas zmian, które wynikają z ewolucji poglądów na same projekty i zarządzanie nimi. Głównym czynnikiem różniącym metody zwinne od tradycyjnych jest uznanie ludzi jako podstawowego czynnika sukcesu projektu, który połączony jest z intensywnym naciskiem na skuteczność i uwzględnienie zmian (Kendall i Kendall 1999). Jest też to swoista odpowiedź na wyzwania biznesowe powstałe z wyniku ukształtowania się szybko zmieniających się globalnych rynków.

Poniżej przestawione są założenia metodyk zwinnych, które traktować można również jako główne różnice pomiędzy tradycyjnymi metodykami zarządzania projektami (Awad 2005; Koszlajda 2010):

– *Zorientowanie na interesariuszy projektu.* Jest to według tych metodyk najważniejszy czynnik związany z rozwojem projektu (co zgadza się z obserwacjami m.in. Standish Group) – najważniejszym zadaniem dla kierowników zespołów „zwinnych" projektów jest to, żeby kłaść większy nacisk na ludzi wraz z ich umiejętnościami, takimi jak: ambicje, zdolności, oraz na wzajemną komunikację (Highsmith i Cockburn 2004). Jeżeli ludzie zaangażowani w projekt nie są, wtedy żaden proces nie naprawi ich nieadekwatności.
– *Adaptacyjność* (nieosiągalna idée fixe metod tradycyjnych). W tym podejściu kładzie się nacisk na zarządzanie zmianą. Sprzyja to przekazaniu użytkownikowi własnej wiedzy większej niż minimalna zakładana projektem (Sully 1993). Zarządzanie zmianą zakłada ciągłą reakcję na ciągłe zmiany zachodzące w projekcie. Najtrudniejsze do oceny i reakcji są zewnętrzne zmiany środowiskowe. Ponieważ nie jest możliwa ich eliminacja, należy dążyć do minimalizacji kosztów z nimi związanych.
– *Zgodność z rzeczywistością.* Zwraca się bardziej uwagę na zgodność otrzymanych rezultatów z uzyskanymi wynikami projektu niż na zgodność z wynikami początkowo zakładanymi (zgodność nie z planem, ale z założeniami biznesowymi).
– *Elastyczność planowania.* Podstawowym problemem planowania projektu jest brak możliwości przewidzenia implikacji rozwoju planów przedsięwzięć innowacyjnych (wszystkie projekty powinny należeć do tej kategorii), ponieważ środowisko, w którym powstają, jest wysoce

dynamiczne. W „zwinnych" projektach wykonawcy muszą zastanowić się, jak mogą uniknąć nieodwracalności swoich decyzji wymuszonych przyzwyczajeniem do praktyki szczegółowego projektowania tradycyjnego, które prowadzi do daleko posuniętej szczegółowości. Zamiast próbować podjąć właściwe decyzje na początku każdego cyklu (projektowanie tradycyjne), lepiej je podjąć w taki sposób, żeby w następnych etapach można było je odwrócić.

- *Oparcie się na procesach empirycznych.* W metodach tradycyjnych procesy występują jako deterministyczne i liniowe, w metodach „zwinnych" jako proces empiryczny (probabilistyczny, źle lub słabo ustrukturalizowany) bądź nieliniowy. Proces deterministyczny to taki, w którym od rozpoczęcia do zakończenia można za każdym razem spodziewać się takich samych wyników. Projektów, poprzez cechę wyjątkowości, jednorazowości itp., nie można zdefiniować jako procesów deterministycznych, ponieważ w czasie ich realizacji może się rozwijać i produkt, i zespół projektowy. Jest bardzo mało prawdopodobne, aby jakikolwiek zestaw predefiniowanych kroków doprowadził do pożądanego, przewidywalnego wyniku, ponieważ zmieniają się wymagania technologiczne oraz ludzie wewnątrz zespołu projektowego.
- *Wykorzystanie podejścia zdecentralizowanego.* Zdecentralizowany styl zarządzania może znacząco wpłynąć na projekt, ponieważ może on zaoszczędzić więcej czasu niż podeście autokratyczne. W metodykach lekkich deleguje się część zadań związanych z podejmowaniem decyzji do wszystkich członków zespołu.
- *Prostota.* W projektowaniu prowadzonym metodykami lekkimi zawsze wybierana jest najprostsza droga prowadząca do celu. Zakłada się łatwe zmiany modelowe, które dostosowywane są do bieżących potrzeb i mogą następować w różnych terminach. Nie wytwarza się większej funkcjonalności niż ta, która jest w danym momencie konieczna, ani dokumentacji próbującej przewidzieć przyszłość projektu. To zmniejsza koncentrację na znalezienie informacji potrzebnych do tych predykcji (Gibson i Hughes 1994).
- *Oparcie się na współpracy z odbiorcą (użytkownikiem) i współpracy wewnętrznej.* Klient projektu powinien ściśle współpracować z zespołem projektowym, zapewniając mu wszelkie potrzebne informacje, oraz zgłaszać bieżące uwagi i komentarze do projektu. Ze względu na decentralizację zespół wykonawczy w metodykach „zwinnych" powinien się w sposób ciągły komunikować ze sobą.
- *Funkcjonowanie poprzez małe, samoorganizujące się zespoły*. W zespołach takich obowiązki i zadania przekazywane są do zespołu jako całości, a zespół, rozdzielając je, zapewnia najlepszy sposób ich realizacji. W małych zespołach najlepiej sprawdza się idea ciągłej komunikacji. Struktura procesu i konkretne praktyki tworzą minimalne, elastyczne ramy strukturalne dla zespołów samoorganizujących. Odpowiednie

ich wykorzystanie znacznie zmniejsza ryzyko związane z czynnikiem ludzkim.

Jako przykład metodyki projektowania lekkiego przedstawiono *metodykę Scrum*. Jest to iteracyjna metodyka oparta na zarządzaniu zmianą, koncentrująca się na ciągłych, twardych negocjacjach pomiędzy graczami: użytkownikiem maksymalnie możliwego elastycznego systemu informatycznego a zespołem projektowym. Nazwa nawiązuje do sytuacji występującej podczas gry w rugby. Termin *scrum* (młyn) oznacza w niej rzut wolny, w którym spleceni ze sobą gracze przeciwnych drużyn przepychają się (negocjują) nad piłką (celem) (Jaszkiewicz 1997). Scrum jest jednocześnie metodyką prototypowo-przyrostową – wartość dodana projektu powstaje w wyniku zastosowania w kolejnych iteracjach faz cyklu życia systemu określonych narzędzi (przeciwdziałających potencjalnemu ryzyku realizacyjnemu), wynikających z wypracowanych dobrych praktyk dla małych i średnich, ale za to złożonych projektów (Rising i Janoff 2000: 26–32).

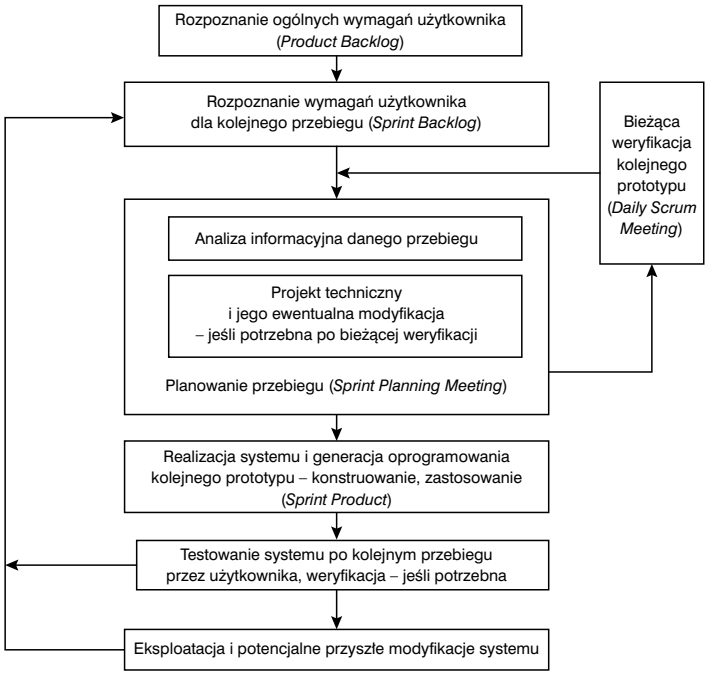
Podstawowe używane w metodzie Scrum praktyki projektowania można sprowadzić do następujących:

– tworzenie rejestru zamówień (*product backlog*) – czyli listy wszystkich wymagań użytkownika: funkcji i ustalonych z realizatorem zmian wraz z priorytetami, czekających na realizację (odpowiedzialny użytkownik końcowy – właściciel produktu, *product owner*);
– cykl pracy – przebieg (*sprint*) – etap pracy zespołu projektowego (od jednego do sześciu tygodni, z zaleceniem regularności i jednolitości długości trwania każdej iteracji, np. miesiąc); w każdym cyklu jest dostarczana użytkownikowi do przetestowania i oceny następna działająca wersja prototypu produktu;
– planowanie cyklu pracy – przebiegu (*sprint planning meeting*) – składa się z dwóch części: analizy i projektu realizacji; w pierwszej wraz ze wszystkimi użytkownikami zespół projektowy ustala kompletny (w danym momencie) zbiór celów i funkcji systemu, w drugiej zaś kierownik projektu (*scrum master*) z zespołem uzgadnia najlepszy sposób realizacji produktu podczas danego przebiegu;
– tworzenie rejestru zamówień dotyczącego konkretnego przebiegu (*sprint backlog*) – listy nowych lub zmienionych funkcjonalności przypisanej do kolejnego przebiegu; w momencie jej realizacji powstaje nowa wersja prototypu;
– weryfikacja postępu prac (*daily scrum meeting*) – odbywa się w trakcie obowiązkowych codziennych spotkań zespołu projektowego; polega na identyfikacji niezbędnych zmian i określenia warunków ich realizacji przez zespół (na zasadach samorealizacji).

Cykl życia w metodzie Scrum (rysunek 2) przebiega według następującego schematu:

– rozpoznanie ogólnych wymagań i wstępna analiza informacyjna całości systemu;

– rozpoznanie i analiza dla kolejnego, bieżącego przebiegu;
– planowanie przebiegu (analiza i projekt techniczny);
– bieżąca weryfikacja założeń w trakcie codziennych spotkań;
– realizacja systemu i generacja oprogramowania kolejnego prototypu – konstruowanie i zastosowanie;
– testowanie systemu po kolejnym przebiegu;
– eksploatacja i potencjalne przyszłe modyfikacje systemu.



Rys. 2. Cykl życia systemu informatycznego w metodyce Scrum. Źródło: opracowanie własne na podstawie K. Schwaber i M. Beedle 2001. Agile Software Development with Scrum, Prentice Hall; K. Schwaber 2005. Sprawne zarządzanie projektami metodą Scrum, Warszawa: Promise.

Przed każdym przebiegiem odbywa się spotkanie zespołu realizacyjnego z użytkownikami, identyfikujące priorytetowe zadania (określenie zakresu, zawartości i intencji użytkownika) oraz konwertujące je w funkcjonalności przyszłego systemu. Na tej podstawie tworzony jest plan wykonania oprogramowania w bieżącej iteracji (priorytety, podział obowiązków, szczegółowe działania). Natomiast na początku każdego dnia przebiegu (w niektórych interpretacjach tej metodyki – na zakończenie dnia) w trakcie spotkania zamkniętego zespołu projektowego odbywa się bieżąca weryfikacja realizacji

zadań (stan wykonania, problemy ogólne i szczegółowe realizacji oraz jednostkowe sposoby ich rozwiązania). Ma na celu koordynacje i synchronizację dziennej pracy członków zespołu. Po każdej iteracji odbywa się natomiast spotkanie z użytkownikiem w celu prezentacji produktu kolejnej iteracji i określenia, czy realizuje on kierunek zmian przez niego oczekiwany. Ma ono pomóc klientowi w określeniu, czy i co w następnej kolejności powinno być wykonywane. Na podstawie wniosków z tego spotkania produkt zostaje przekazany do testowania i użytkowania lub, w kolejnej iteracji, do dalszych modyfikacji.

4. Kryteria wyboru pomiędzy metodykami tradycyjnymi i nowoczesnymi

Przeprowadzona powyżej analiza wykazała, że istnieje wiele metodyk zarządzania projektami i nie ma jednej właściwej dla wszystkich przypadków metodyki uniwersalnej. Metodyki są tylko pewnym zbiorem wzorców, zasad oraz formuł, które pomagają w uniknięciu błędów, lecz zupełnie ich nie usuwają. Pomimo to konsekwentnie wdrożona metodyka daje poczucie kontroli nad przedsięwzięciem, utrzymywania pełnego zaangażowania wszystkich zainteresowanych stron oraz uzasadnia poczucie bezpieczeństwa realizacji strategii biznesu sponsora.

W związku z powyższym każdy projekt informatyczny wymaga zdefiniowania metodyki, według której będzie prowadzony. W najbardziej ogólnym sposób – opierając się na wcześniej zdefiniowanych cechach charakterystycznych metodyk tradycyjnych i nowoczesnych – możemy stwierdzić, że wybór ten określa szereg czynników związanych z konkretnym projektem (jego wielkością, branżą, w której jest realizowany, czynnikiem ludzkim w projekcie i dodatkowymi wymogami realizacyjnymi itd.).

Niemniej jednak wstępną listę kryteriów pomocniczych wyboru można oszacować na podstawie takich zmiennych, jak:

– *Rodzaj i harmonogram finansowania prac zawarty w umowie z klientem.* Jeżeli zakłada się wykładniczy, to powinny to być metodyki zwinne. W tych metodykach przyjmuje się, że czynnikami ważniejszymi niż negocjowanie kontraktów są zaufanie i współpraca partnerów biznesowych (przynajmniej deklaratywnie). W trakcie podpisywania umowy i wstępnego ustalania harmonogramu albo przyjmuje się realizację budżetu od początku korzystną dla klienta (powoli rosnącą na początku projektu, wykładniczo wraz z przekazywaniem uzgodnionych, przyjmowanych przez użytkownika produktów), albo na kontraktach o niewielkim stopniu ryzyka dla realizatora projektu (np. umowy z refundowanym kosztem według cen stałych). Jeżeli natomiast zakłada się logarytmiczną funkcję realizacji budżetu (wysoką w stadiach początkowych, stabilizującą się w późniejszych fazach projektu), to – zwłaszcza dla firm o ugruntowanej pozycji na rynku – opłaca się przyjąć jedną z metodyk klasycznych.

- *Typ budżetu.* Nierównomiernie rozłożony w czasie budżet projektu, wymuszający różną intensywność prac oraz wysoki stopień niepewności, wskazuje na efektywniejsze użycie metodyki zwinnej. Zaś budżet finansujący projekt w sposób równomierny i stały w czasie sugeruje wykorzystanie metodyki klasycznej.
- *Charakter ustaleń harmonogramu.* Narzucone w harmonogramie sztywne założenia czasowe w stosunku do zaplanowanych zadań przemawiają na korzyść zastosowania metodyk klasycznych. Przybliżone lub relatywne terminy realizacji sugerują użycie metod zwinnych.
- *Ilość i poziom wymaganej dokumentacji oraz poziom jakości oprogramowania.* Spełnienie złożonych wymagań związanych z formalnymi standardami dokumentacji, licencjami czy certyfikacji wskazuje na konieczność zastosowania klasycznej metodyki zarządzania projektem. Natomiast projekt, w którym nacisk kładziony jest na jakość oprogramowania i możliwości jego rozwoju, jest zgodny z filozofią metodyk zwinnych.
- *Podejście do ryzyka projektowego* – niskie ryzyko projektowe obsługują lepiej metodyki zwinne. Wysokie ryzyko w projekcie wymusza stosowanie metodyk klasycznych, w których zakłada się tworzenie planów minimalizacji ryzyka lub sytuacji awaryjnych.
- *Komunikacja z klientem.* Metodyki zwinne zakładają ciągłą oraz bezpośrednią komunikację z klientem. Zmniejszane są więc wszelkie czynniki związane z niezrozumieniem realizowanych zadań. W metodykach klasycznych preferuje się rzadszy kontakt z klientem, co prowadzi bardzo często do nieporozumień w momencie prezentacji i przekazania zrealizowanego systemu.
- *Struktura organizacyjna konieczna do realizacji projektu.* Przedsiębiorstwa o strukturze hierarchicznej lub pokrewnych, zawierające w swoich strukturach ściśle wyspecjalizowane jednostki, będą preferować metodyki klasyczne. Przedsiębiorstwa o strukturze np. macierzowej, projektowej lub innej, w której mogą sobie pozwolić na delegowanie zadań projektowych do mniejszych zespołów, w których nie ma ściśle zdefiniowanej hierarchii organizacyjnej, powinny zdecydować się na wybór metodyki klasycznej.
- *Sektor/branża realizacji projektu* związana często ze strukturami organizacyjnymi też znacząco wpływa na realizację projektu. Branże, które na wejściu mają niewielką liczbę surowców i materiałów, a na wyjściu gotowych produktów, lepiej nadają się do użycia w trakcie projektowania systemu metod klasycznych.
- *Wielkość projektu.* Część środowisk programistycznych związanych z wdrożeniami wielkich systemów uważa, że metody klasyczne lepiej nadają się do realizacji tego typu projektów. W metodykach zwinnych (a zwłaszcza omawianej Scrum) trudno przy wielkich projektach, nawet powtarzalnych, skoordynować działania wielu małych grup realizujących projekt.
- *Rodzaj systemu, dla którego projekt jest realizowany.* Systemy eksperckie lub oparte na bazie wiedzy wymagają bardzo często dużej wiedzy o reali-

zowanym zagadnieniu. Jeśli są to systemy z założenia konstruowane jako samouczące, zaleca się stosowanie metodyk nowoczesnych, w przeciwnym przypadku – dla systemów opartych na ustalonych wzorcach postępowania – metodyk klasycznych.

– *Czynniki psychologiczne*, np. doświadczenie zespołu realizacyjnego w stosowaniu metodyk zwinnych lub klasycznych, zaufanie organizacji, w której projekt jest realizowany do zespołu projektowego, wysoki stopień wykorzystania w zespołach mieszanych specjalistów ze strony przyszłego użytkownika itp.

Powyższa lista czynników nie jest ani kompletna, ani wystarczająca. Bardzo trudno jest bowiem ex ante ocenić, która ze znanych metodyk w przedstawionym układzie jest najlepsza dla realizowanego projektu. Tak więc coraz więcej organizacji utrzymujących się z realizacji projektów jest zainteresowanych zarządzaniem przez projekt, polegającym w praktyce (upraszczając sytuację) głównie na równoległym prowadzeniu jak największej liczby projektów. Umożliwia to nie tylko porównanie ich cech charakterystycznych (harmonogramów, kosztów, zasobów) realizowanych dla różnej wielkości firm, w wielu branżach, w wielu lokalizacjach, ale również skuteczności metody prowadzenia projektów, aż do zaproponowania własnej. Pozwala to również przecież na wielowymiarową ocenę (w tym porównawczą, efektywności itd.) prowadzonych projektów i dostarcza kierownikom projektów wzorców dobrych praktyk zarządzania oraz ocenę efektywności poszczególnych jednostek organizacyjnych prowadzących projekty.

Analizy wykonania projektów dostarczają też przesłanek do współdzielenia zasobów projektów (w tym technologii, know-how itp.), rozliczenia czasu i kompetencji wykonawców oraz wyceny generowanych w projekcie produktów lub usług. Zarządzanie przez projekty jest stylem zarządzania doskonalącym przedsiębiorczość i będącym swoistym „złotym środkiem" pomiędzy rzeczywistymi potrzebami firmy zleceniodawcy (określonej przez jej specyfikę) a wiedzą (teoretyczną i praktyczną) i metodami zarządzania projektami (Stabryła 2006). Ta pragmatyka podejścia umożliwia tworzenie wzorców postępowania podczas realizacji poszczególnych rodzajów projektów, co ułatwia tworzenie strategii nie tylko organizacji utrzymujących się z tworzenia i wdrażania projektów (i coraz częściej – nie tylko informatycznych) oraz niewątpliwie w ostatecznym rozrachunku wybór metodyk zarządzania projektem.

Informacje o autorze
Prof. dr hab. Witold Chmielarz – Katedra Systemów Informacyjnych Zarządzania, Wydział Zarządzania Uniwersytetu Warszawskiego. E-mail: vitec@post.pl.

Przypisy

1 Więcej na ten temat np. w: Chmielarz 2010: 359–402.

2 Patrz też: Chmielarz i Klincewicz 2010: 238–252.

Bibliografia

Awad, M.A. 2005. *A Comparison between Agile and Traditional Software Development Methodologies*, The University of Western Australia, http://www.scribd.com/doc/55475-190/A-ion-Between-Agile-and-Traditional-SW-Development-Methodologies, odczyt: marzec 2012.

Beck, K. i in. 2010. *Manifesto for Agile Software Development*, Agile Alliance, 2001, Retrieved 14 June 2010, http://www.pmbriefcase.com/methodologies/50-software-development/55-agile-software, odczyt: kwiecień 2012.

Chmielarz, W. 2010. Projektowanie systemów informatycznych, w: J. Zawiła-Niedźwiecki, K. Rostek i A. Gąsiorkiewicz (red.) *Informatyka gospodarcza*, t. 1, s. 359–402. Warszawa: C.H. Beck.

Chmielarz, W. i K. Klincewicz 2010. Zarządzanie w kontekście zmian, w: J. Bogdanienko (red.) *Organizacja i zarządzanie w zarysie*, s. 238–252. Warszawa: Wydawnictwo Naukowe WZ UW, Dom Wydawniczy Elipsa.

Gibson, M. i G. Hughes 1994. *Systems Analysis and Design. A Comprehensive Methodology with CASE*, Boyd & Fraser.

Highsmith, J. i A. Cockburn 2004. Agile Software Development: The Business of Innovation. *IEEE Computer*, nr 6 (36), http://www.jimhighsmith.com/articles/IEEEArticle1Final.pdf, odczyt: kwiecień 2012, DOI: 10.1109/2.947100.

Jaszkiewicz, A. 1997. *Inżynieria oprogramowania*, Gliwice: Helion.

Kendall, K. i J. Kendall 1999. *Systems Analysis and Design*, Upper Saddle River: Prentice Hall.

Koszlajda, A. 2010. *Zarządzanie projektami IT. Przewodnik po metodykach*, Gliwice: Helion.

Mannaro, K. 2008. *Adopting Agile Methodologies in Distributed Software Development*, Wydział Inżynierii Elektrycznej i Elektroniki Uniwersytetu Cagliari, http://www.diee.unica.it/DRIEI/tesi/19_mannaro.pdf.

Mingus, N. 2009. *Zarzadzanie projektami*, Gliwice: Helion.

Project Management Institute 2004. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, Third Edition.

Rising, L. i N.S. Janoff 2000. The Scrum Software Development Process for Small Teams. *IEEE Software*, nr 17, s. 26–32, DOI: 10.1109/52.854065.

Schwaber, K. 2005. *Sprawne zarządzanie projektami metodą Scrum*, Warszawa: Promise.

Schwaber, K. i M. Beedle 2001. *Agile Software Development with Scrum*, New Jersey: Prentice Hall.

Stabryła, A. 2006. *Zarządzanie projektami ekonomicznymi i organizacyjnymi*, Warszawa: Wydawnictwo Naukowe PWN.

Sully, P. 1993. *Modelling the World with Objects*, New Jersey: Prentice Hall.