Na styku biznesu i IT: zakres projektu informatycznego i zakres oprogramowania

Dział ZP

Brzeska Ewa

# Wstęp

W [pierwszym numerze REQMagazyn](http://req.wymagania.org.pl/magazyn/pobierz-magazyn/) ukazał się mój artykuł "*Na styku biznesu i IT: od potrzeby biznesowej do gotowego produktu – rozważania w kontekście inżynierii oprogramowania*". Poruszyłam w nim kilka aspektów, które są istotne, aby proces wytwórczy oprogramowania realizowany przez IT zakończył się pełnym sukcesem, czyli dostarczył oczekiwaną wartość biznesowi. Zainteresowanych zapraszam serdecznie do przeczytania całego artykułu. Dziś chciałabym kontynuować poruszanie się po styku biznesu i IT i zająć się szerzej pojęciem "zakres". Pojęcie to występuje w kontekście tworzonego oprogramowania - produktu, który IT ma dostarczyć biznesowi oraz w kontekście projektu informatycznego, w ramach którego oprogramowanie jest wytwarzane. Zakres oprogramowania definiuje konkretne funkcje, jakie mają być zawarte w produkcie, zakres projektu natomiast – prace, które mają być wykonane, aby powstał produkt posiadający zdefiniowane funkcje. Zasadne jest zatem rozważanie pojęcia „zakres” zarówno z perspektywy zarządzania projektem informatycznym, jak i z perspektywy wytwórczej - inżynierii oprogramowania.

Zakres systemu informatycznego, bądź w ujęciu projektowym - zakres prac, jakie mają być zrealizowane, jest pojęciem niezmiernie istotnym. Definiuje dokładnie **co** ma być wykonane oraz stanowi podstawę do odbioru wykonanych prac przez biznes. Dla IT precyzyjnie określony zakres projektu jest podstawą do szacowania jego pracochłonności oraz terminów i kosztów realizacji. Tymczasem, jak pokazuje praktyka, zakres projektu z różnych powodów bywa nieprecyzyjnie ustalony we wstępnych fazach prac nad projektem oraz zmienia się nagminnie w fazie implementacji oprogramowania. Jakie to niesie za sobą reperkusje? Na co należy zwrócić uwagę podczas realizacji projektu? Na czym polega zarządzanie zakresem? Spójrzmy na to, co jest istotne zarówno z punktu widzenia biznesu, jak i IT, ze wspólnej perspektywy prowadzącej do powodzenia w realizacji projektu, w rozumieniu dostarczenia przez IT produktu oczekiwanego przez biznes, zgodnie z postawionym na wstępie celem biznesowym przedsięwzięcia.

# Zakres jako jeden z parametrów projektu informatycznego

Wszyscy pewnie znają pojęcie trójkąta ograniczeń projektu. W ujęciu klasycznym tzw. żelazny trójkąt projektu przedstawiany jest jako trójkąt o trzech wierzchołkach: zakres, czas, koszty. Jeszcze lepiej tradycyjną rzeczywistość projektową odzwierciedla trójkąt obrazujący zrównoważony system, w którym występuje pięć wzajemnie zależnych zmiennych:

* zakres
* koszt
* czas
* jakość
* zasoby

Budowa tego trójkąta jest w zasadzie dowolna. Ważna jest odpowiednia interpretacja pomiędzy poszczególnymi zmiennymi, które budują trójkąt oraz brak elastyczności wzajemnych powiązań. Do poniższych rozważań przyjęłam model, w którym czas, budżet i zakres umieściłam po bokach trójkąta, natomiast jakość i zasoby znalazły się w jego wnętrzu (Rys. 1)

{Rys. 1 Trójkąt ograniczeń projektu}

Wzajemna zależność w/w zmiennych – parametrów projektu informatycznego - oznacza, iż zmiana jednego z nich pociąga za sobą konieczność zmiany innych zależnych parametrów tak, aby cały układ wrócił do równowagi. Ponieważ te właśnie parametry mają decydujący wpływ na sukces lub porażkę projektu, myślę że warto poświęcić im chwilę uwagi.

**Zakres** definiuje dokładnie granice projektu, czyli co w projekcie ma być wykonane oraz to, co w ramach tego projektu zrealizowane nie będzie. Określanie zakresu projektu odbywa się na podstawie wymagań klienta w stosunku do produktu – oprogramowania, które ma zostać wytworzone w ramach projektu. Jasno i precyzyjnie zdefiniowane wymagania są podstawą definicji projektu informatycznego. Zbiór kompletnych wymagań, zarówno funkcjonalnych, jak i niefunkcjonalnych, definiuje również pełen zakres produktu. Wymagania określane są we wstępnej fazie prac projektowych, co nie znaczy, że pozostają stałe w całym okresie realizacji projektu. Im projekt jest bardziej rozbudowany, a czas jego realizacji dłuższy, tym bardziej prawdopodobne, że wymagania w stosunku do wytwarzanego oprogramowania ulegną zmianie, co pociągnie za sobą zmianę zakresu samego produktu, jak i projektu, w ramach którego produkt jest wytwarzany.

**Koszt**, czy budżet projektu, to parametr wyrażony w jednostkach monetarnych, który ma decydujący wpływ na cały projekt. Często zdarza się, że zakres projektu, a także funkcjonalność oprogramowania, które ma w ramach tego projektu powstać, są wręcz dopasowywane do środków (górnej granicy budżetu), które klient biznesowy może przeznaczyć na ten projekt. A chyba nie znam klienta, który podczas prezentacji oferty nie zapytałby „dlaczego tak drogo i czy da się zrealizować ten projekt taniej?”. Warto pamiętać, że koszt jest dla klienta często najważniejszym, bo z jego punktu widzenia najbardziej odczuwalnym, parametrem projektu. Z własnej praktyki projektowej wiem, że jeśli już zostanie ustalony budżet, w ramach którego ma powstać produkt informatyczny, nagminnie zdarza się, że zamawiający chce zwiększać ustalony razem z budżetem zakres projektu, przy zachowaniu pozostałych warunków (czas wykonania oraz jakość produktu) i jednocześnie zupełnie nie widzi potrzeby zwiększenia budżetu na to przedsięwzięcie.

**Czas –** zmienna określająca ramy czasowe projektu, które najczęściej wyznacza termin zakończenia realizacji zadań projektowych. Czas jest zazwyczaj mocno związany z kosztami, co oznacza, że np. jeśli w trakcie realizacji zadań projektu znacznie zwiększy się jego zakres, to da się dotrzymać wcześniej wyznaczonego terminu realizacji jedynie wówczas, jeśli znacząco zostanie zwiększony budżet przedsięwzięcia.

**Jakość –** w kontekście produktów i projektów informatycznych parametr jakości występuje w dwóch kategoriach:

* Jakość produktu końcowego, rezultatu projektu
* Jakość procesu realizacji tego projektu

Jeśli chodzi o jakość produktu końcowego - przyjmuje się, że produkt ma właściwą jakość, jeśli spełnione są kryteria jakościowe określone w dokumentacji projektu, w ramach którego produkt jest wytwarzany. W informatycznych projektach wytwórczych produktem jest oprogramowanie. Z racji różnorodności typów oprogramowania i jego przeznaczenia, nie można mówić o jednoznacznej definicji jakości, uniwersalnej dla wszystkich produktów. Inne normy jakościowe będzie miało oprogramowanie sterujące maszynami lub procesami, inne – oprogramowanie czasu rzeczywistego wspomagające procesy podejmowania decyzji czy systemy bankowe, a całkowicie inne – proste programy i aplikacje desktopowe czy mobilne służące rozrywce użytkownika. Kryteria jakościowe produktu powinny zatem każdorazowo zostać jednoznacznie zdefiniowane w fazie zbierania wymagań. Kryteria te mają bezpośredni wpływ na zakres produktu i projektu.

**Zasoby**, czyli przede wszystkim ludzie – wykonawcy zadań związanych z realizacją projektu oraz sprzęt i infrastruktura, dzięki którym realizacja tych zadań staje się możliwa. Zapewnienie właściwych zasobów i zarządzanie nimi jest zazwyczaj kwestią strony wykonawczej projektu. Klienta biznesowego nie interesuje zwykle, kto personalnie wytwarza dla niego produkt, na jakim sprzęcie produkt jest wytwarzany i testowany oraz jaka infrastruktura jest do tego potrzebna. Nieco inaczej sprawa może wyglądać jeśli produkt wytwarzany jest w metodyce zwinnej, która z definicji zakłada ścisłą współpracę wykonawcy z zamawiającym, ale chwilowo zostawmy na boku tę kwestię – do metodyki realizacji projektu i wytwarzania oprogramowania wrócę jeszcze w dalszej części artykułu.

Powyżej omówiłam krótko każdy z pięciu parametrów projektu, parametrów które mają bezpośredni wpływ na produkt informatyczny wytwarzany w ramach danego projektu. Przy okazji omawiania w/w zmiennych myślę, że warto poruszyć jeszcze jedną kwestię. Otóż **szybko, tanio i jednocześnie dobrze to mission impossible projektu informatycznego**. Nie da się wytworzyć oprogramowania wysokiej jakości, jeśli budżet będzie niewystarczający, a czas zbyt krótki. Warto o tym pamiętać, gdyż przy zdefiniowanym zakresie projektu i z góry narzuconym przez zamawiającego terminie jego realizacji, zbyt często jedynym kryterium wyboru oferenta jest cena za jaką oferuje on swoje usługi. Rażąco niska wycena zamiast cieszyć klienta, powinna tak naprawdę od razu budzić jego niepokój co do przyszłej jakości produktu za tak niską cenę wytworzonego.

# Zmiana zakresu produktu w trakcie realizacji projektu

Układ wzajemnie zależnych parametrów projektu pozostaje w równowadze na początku prac nad projektem, pod koniec fazy planowania, kiedy to zatwierdzany jest budżet, harmonogram realizacji projektu i planowane są zasoby. Praktyka pokazuje jednak, że w trakcie realizacji projektu, dawno po zakończeniu fazy planowania oraz zatwierdzeniu budżetu i harmonogramu, zamawiający bardzo często prosi o wprowadzenie różnego rodzaju zmian we wcześniej określonym zakresie projektu. Najczęściej są to dodatkowe funkcje oprogramowania, które nie były brane pod uwagę na etapie planowania, a które w mniejszym lub większym stopniu zwiększają zakres produktu i projektu. Powody chęci zmian mogą być różne: najczęściej jest to zmiana oczekiwań zamawiającego w stosunku do realizowanego produktu, która może być spowodowana np. nowymi uwarunkowaniami rynkowymi, korektą celu biznesowego przedsięwzięcia, działaniami konkurencji, czy nawet zmianami technologicznymi, jakie zaszły w międzyczasie. Nie ma nic pewniejszego podczas realizacji projektu informatycznego niż to, że w trakcie wytwarzania oprogramowania trzeba będzie reagować na różnego rodzaju zmiany zakresu. Przy czym najczęściej zamawiający sądzi, że rozszerzenie zakresu, którego właśnie chce dokonać jest tak znikome, że nie powinno mieć wpływu na pozostałe parametry projektu, czyli przede wszystkim na budżet oraz czas realizacji (termin oddania projektu). Spójrzmy tymczasem na Rysunek 2 i zobaczmy co dzieje się z układem, który uprzednio był w równowadze:

{Rys. 2 Trójkąt ograniczeń projektu - niezrównoważony}

Jak widać, nie da się zwiększać zakresu produktu i projektu bez wpływu na pozostałe parametry. Co prawda można sobie wyobrazić, że jeśli zmiany zakresu są niewielkie i dotyczą na przykład kilku dodatkowych funkcji oprogramowania, którego funkcjonalność jest bardzo złożona, zmiana taka będzie na tyle mało istotna, że wykonawca zrealizuje ją na własny koszt, w imię powodzenia całego projektu. A co robić, jeśli na skutek kolejnych żądań klienta zmiany zakresu stanowią 10 lub 20 % zakresu pierwotnego? Wtedy na pewno nie da się już zrealizować przedsięwzięcia z zachowaniem stałości pozostałych parametrów projektu. Co zatem możemy zrobić, aby wówczas cały projekt zakończył się sukcesem, klient był zadowolony z produktu, który otrzyma, a wykonawca nie poniósł straty realizując projekt?

# Zarządzanie zmianą zakresu w projekcie

Klient zamawiający potrzebne mu oprogramowanie nie musi absolutnie znać zasad zarządzania projektem, w tym projektem IT. Nie musi także wiedzieć o zależnościach pięciu parametrów projektu pozostających ze sobą w ścisłym związku, o których pisałam wyżej. Dlatego aby ułatwić późniejsze zarządzanie zmianami zakresu w projekcie, warto na etapie planowania projektu uświadomić zamawiającemu te zależności oraz uzgodnić z nim, jakie priorytety mają dla niego poszczególne parametry, a następnie umieścić te informacje w specyfikacji wymagań projektu. Przykładową macierz możliwych kompromisów projektowych - tabelę zawierającą zmienne projektu oraz nadane im priorytety umieściłam w Tabeli 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Priorytet  Parametr | Najwyższy (krytyczny, ustalony) | Optymalizowany | Optymalizowany | Optymalizowany | Najniższy (elastyczny, negocjowalny) |
| 5 | 4 | 3 | 2 | 1 |
| Zakres |  |  | X |  |  |
| Koszt |  |  |  | X |  |
| Czas | X |  |  |  |  |
| Jakość |  | X |  |  |  |
| Zasoby |  |  |  |  | X |

Tabela 1 Przykładowa macierz kompromisów projektowych

Dzięki zdefiniowaniu priorytetów odnośnie poszczególnych parametrów projektu, każdorazowo gdy zamawiający zgłasza potrzebę zmiany zakresu projektu, należy przygotować dokument, w którym zawarty jest opis wpływu oczekiwanych zmian na projekt oraz co najmniej jedna propozycja praktycznego wprowadzenia zmian w projekcie, uwzględniająca wcześniej nadane priorytety. Dla przykładu priorytetów zawartego w Tabeli 1, ze względu na to, że projekt jest mocno zorientowany na czas i jakość, można zaproponować klientowi wprowadzenie zmian kosztem zwiększenia budżetu na jego realizację oraz zrealizowanie zmiany za pomocą większej ilości zasobów. Takie podejście pozwala podążać wytwórcy oprogramowania za oczekiwaniami klienta, jednocześnie uzmysławiając mu, że oczekiwane przez niego zmiany zakresu projektu generują najczęściej konieczność zmian pozostałych zmiennych definiujących projekt.

Jak może skutkować w projekcie zmiana proponowana przez klienta? Możliwych jest kilka wariantów, z których najczęściej występują poniższe:

* zmiana mała - nie pociąga za sobą żadnych skutków w pozostałych zmiennych projektu
* zmiana możliwa do przeprowadzenia, lecz konieczne jest wydłużenie czasu realizacji projektu, aby można ją było zrealizować przy obecnych zasobach
* zmiana możliwa do realizacji, lecz żeby projekt mógł zostać zrealizowany we wcześniej ustalonym terminie, konieczne będą dodatkowe zasoby.
* zmiany można dokonać jedynie wówczas, jeśli zostanie wydłużony czas realizacji projektu oraz zostaną przydzielone dodatkowe zasoby.
* proponowana zmiana ma na tyle silny wpływ na projekt, że nie można jej dokonać bez znacznej modyfikacji założeń całego projektu.

Jak to wygląda w praktyce? Wyobraźmy sobie, że w trakcie realizacji projektu klient silnie nalega na dołożenie kilku czy kilkunastu funkcji do wcześniej ustalonej funkcjonalności. Zamawiający chce przy tym (co jest bardzo częste), aby zmieścić się w ustalonym wcześniej budżecie i dotrzymać wcześniej ustalonego terminu realizacji, natomiast wykonawca z pewnych przyczyn chce sprostać tym żądaniom. Czy jest to w ogóle możliwe?

Teoretycznie tak – można na przykład skierować do pracy nad tym projektem większy zespół ludzi przy zachowaniu tego samego budżetu. Oznacza to, że przy realizacji zadań projektu będzie pracować więcej, ale za to mniej opłacanych pracowników. Być może uda się tą drogą rzeczywiście zrealizować większy zakres projektu mieszcząc się we wcześniej ustalonym budżecie i dotrzymując uprzednio ustalonego terminu realizacji, lecz jakość oddanego produktu z dużym prawdopodobieństwem niestety spadnie, gdyż mniej opłacani pracownicy to z reguły pracownicy o  niższych kwalifikacjach zawodowych. Jeśli ta niższa jakość będzie nadal spełniała wcześniej ustalone kryteria jakościowe produktu, to wszyscy powinni być zadowoleni: klient, ponieważ bez zwiększenia budżetu i w oczekiwanym czasie otrzyma produkt o zakresie większym niż pierwotnie ustalony, ze strony wykonawcy zaś nadal takie przedsięwzięcie będzie opłacalne, pomimo zmian wprowadzonych do pierwotnego kształtu produktu i projektu.

Inna sytuacja powstanie, jeśli uzyskana jakość produktu nie będzie spełniała wcześniej ustalonych kryteriów jakościowych – produkt będzie wymagał poprawek, które mogą być zarówno czasochłonne, jak i kosztowne, co w rezultacie z perspektywy wykonawcy uczyni projekt nieopłacalnym. Natomiast produkt nie spełniający kryteriów jakościowych zamawiającego, zostanie przez niego odrzucony i wówczas projekt zakończy się niestety porażką.

Jest jeszcze jedna sytuacja, kiedy zmiana zakresu projektu, przy założeniu zachowania wcześniej ustalonego budżetu oraz czasu może być bardzo trudna od strony wykonawczej. Wyobraźmy sobie, że zamawiający chce dodać do zakresu produktu tylko jedną czy dwie funkcje, które z perspektywy ilościowej całego projektu wydają się pomijalnie małe (częsty argument zamawiającego: cóż znaczą dwie kolejne funkcje, w stosunku do tak dużego systemu, który zamówiliśmy). Z perspektywy jedynie ilościowej tak może to rzeczywiście wyglądać. Lecz dodatkowe funkcje, bez względu na ich ilość, zawsze mogą mieć znaczący wpływ na całą, lub dużą część architektury oprogramowania, które jest już obecnie w zaawansowanej fazie wytwarzania. Z drugiej strony, realizacja tych funkcji może wymagać pewnych specyficznych zasobów, które są aktualnie trudno dostępne. Na przykład zmiany może zrealizować tylko jeden z dwóch programistów systemowych, którzy aktualnie pracują nad innym projektem i pierwszy możliwy termin ich pozyskania dla tego projektu nie gwarantuje, że zmiana zakresu będzie mogła być przeprowadzona w pierwotnie zakładanym terminie.

Tak więc nawet z pozoru niewielkie zmiany zakresu projektu mogą w praktyce nieść za sobą poważne konsekwencje organizacyjne lub finansowe i dlatego powinny być świadomie zaakceptowane przez obie strony: zarówno zamawiającą, jak i wykonawczą. Jeśli zaś wykonawca projektu zdecyduje się zrealizować zmianę na własny koszt, powinien w jasny sposób poinformować zamawiającego, że właśnie robi mu prezent. Klient zwykle nie docenia tego, co łatwo mu przychodzi i jest całkowicie za darmo, a to może rodzić niebezpieczeństwo, że zmiany w tak realizowanym projekcie staną się regułą i de facto mogą bardzo opóźnić lub nawet uniemożliwić jego zakończenie.

# Zarządzanie zmianą zakresu w procesie wytwórczym oprogramowania

A więc stało się – klient zgłosił w projekcie chęć zmiany zakresu produktu, zmiana została zatwierdzona na szczeblu kierownictwa projektu i przychodzi ten dzień, kiedy ze zmianą musi zmierzyć się zespół deweloperski: projektanci i architekci, którzy muszą uwzględnić zmianę w dotychczasowej architekturze rozwiązania i programiści, których zadaniem będzie zmianę tę zaimplementować w kodzie źródłowym. Zwykle nie są oni zachwyceni koniecznością takich zmian. Jeśli praca nad oprogramowaniem trwa już jakiś czas, to najczęściej zmiany zakresu powodują nie tylko konieczność napisania nowych linijek kodu, aby oprogramować nowe funkcje, lecz także generują konieczność modyfikacji kodu już napisanego i dokładnie przetestowanego. Należy przy tym zwrócić uwagę na to, że wiele projektów informatycznych jest dużych i skomplikowanych, czas ich realizacji jest długi, a w ich wyniku powstaje często oprogramowanie liczące dziesiątki i setki tysięcy linii kodu, składające się z wielu bibliotek, modułów i plików, realizowane w zależności od potrzeb, przez mniejsze i większe zespoły specjalistów.

Żądanie zmiany w projekcie nie trafia zwykle w realizacyjną pustkę. Najczęściej deweloperzy ścigają się z czasem (czy ktoś kiedykolwiek widział, żeby harmonogram realizacji projektu był stworzony z jakimkolwiek zbędnym zapasem czasowym?) i w pocie czoła implementują kolejne funkcje oprogramowania. W atmosferze gorączkowego kodowania, aby zdążyć zrealizować projekt na czas, w zależności od estymowanej pracochłonności realizacji nowego zadania, jeden lub kilku programistów dostaje od swojego przełożonego zlecenie zaimplementowania zmiany. Jaki to ma wpływ na pozostałe prace programistyczne? Czy prace te muszą być przerwane, aby mogły zostać wprowadzone zmiany? Jako, że z procesem wytwórczym oprogramowania związana jestem od lat, mogę z czystym sumieniem powiedzieć, że absolutnie nie wyobrażam sobie takiej sytuacji. Prace uprzednio zaplanowane zgodnie z harmonogramem powinny być kontynuowane, natomiast oprócz nich muszą być zrealizowane nowe zadania, związane z implementacją zmiany. Owszem, mogą być potrzebne pewne korekty w stosunku do poprzednio założonego harmonogramu, mogą być konieczne przesunięcia deweloperów do innych zadań niż te, które były poprzednio przydzielone, może być wreszcie konieczne pozyskanie nowych zasobów, czyli rozszerzenie zespołu deweloperskiego – nie wyobrażam sobie jednak całkowitego wstrzymania wszelkich prac programistycznych tylko po to, aby wprowadzić do kodu konieczne zmiany. Tym bardziej, że jak pokazuje praktyka, konieczność zmian pojawia się bardzo często.

{Rys. 3 } [Bez podpisu]

Jak zatem pracować na wspólnym kodzie w wieloosobowym zespole programistycznym, w którym każdy z deweloperów może mieć przydzielone odrębne zadania? Pracę taką, a przede wszystkim zarządzanie zmianami w kodzie oprogramowania oraz jakością kodu źródłowego, umożliwiają systemy kontroli wersji.

Narzędzie do kontroli wersji oprogramowania pozwala blokować poszczególne pliki zawierające kod źródłowy, czy inne zasoby oprogramowania (pliki tekstowe, dźwiękowe, graficzne, dokumentację, itp.) na wyłączność tego członka zespołu deweloperskiego, który ma zrealizować na nich swoje zadanie. W tak zablokowanych plikach nie mogą być przeprowadzane równolegle inne modyfikacje. System kontroli wersji śledzi ponadto wszelkie zmiany dokonane przez wszystkich programistów i inne uprawnione do zmian osoby, a także w razie potrzeby (np. stwierdzenia trudnego do znalezienia i usunięcia błędu w kodzie źródłowym) umożliwia powrót do wcześniejszych wersji oprogramowania.

Korzyści z wdrożenia w zespole deweloperskim narzędzi do kontroli wersji jest znacznie więcej, lecz ich omawianie przekracza ramy tego artykułu. Ważnym jest to, że narzędzie kontroli wersji wdrożone w zespole wytwarzającym oprogramowanie pozwala właściwie zarządzać procesem wytwórczym, poszczególnymi wymaganiami, zmianami zakresu i kodu źródłowego oprogramowania oraz jakością kodu, a przez to jakością wytwarzanego produktu.

# Zakres a metodyka prowadzenia projektu

W moich powyższych rozważaniach skupiłam się przede wszystkim na założeniu, że oczekiwany zakres produktu i projektu są określone we wstępnej fazie realizacji projektu. Oznacza to, że po fazie zbierania wymagań wszystkie wymagania w odniesieniu do produktu - oprogramowania, które ma powstać w wyniku realizacji właśnie rozpoczętego projektu informatycznego są zdefiniowane i zebrane w dokumencie specyfikacji wymagań oprogramowania (ang. *Software Requirements Specification*, SRS), który stanowi podstawę realizacji projektu oraz podstawę prowadzenia prac wytwórczych prowadzących do powstania oprogramowania.

Nie zawsze jednak tak jest. Nie zawsze także projekt informatyczny powinien być aż tak sformalizowany, nie zawsze ma określony z góry zakres, a nawet nie zawsze już na wstępie zdefiniowany precyzyjny cel biznesowy. Do takich nieokreślonych projektów nie pasują tradycyjne metodyki zarządzania projektami informatycznymi, ani tradycyjne metodyki wytwarzania oprogramowania. Tym bardziej, że z perspektywy lat wiadomo, iż metodyki te okazały się w wielu przypadkach zbyt mało elastyczne, za bardzo sformalizowane i w rezultacie za „ciężkie”. W wielu projektach, aby zmaksymalizować wartość biznesową wynikającą z realizowania projektu, warto postawić na ścisłą współpracę zamawiającego i wytwórcy oprogramowania, na szybkie i naturalne reagowanie na zmiany kodem oprogramowania, bez potrzeby tworzenia rozbudowanej dokumentacji tych zmian zanim zmiany zostaną zaimplementowane, na częste przedstawianie klientowi kolejnych iteracji oprogramowania lub też oprogramowania, które jest wynikiem realizacji kolejnego cyklu wytwórczego. Warto wówczas stosować iteracyjny model wytwarzania lub zwinne metodyki wytwarzania oprogramowania i zarządzania projektem.

Ważnym jest, aby zarówno metodykę zarządzania projektem, jak i metodykę wytwarzania oprogramowania każdorazowo dobierać właściwie do projektu, który ma być realizowany. W zależności od celu tworzonego rozwiązania, stopnia zdefiniowania wymagań (pełnego lub tylko częściowego określenia zakresu rozwiązania), a także rodzaju samego oprogramowania, które ma zostać wytworzone w trakcie realizacji prac projektowych, powinna być dobrana odpowiednia metodyka: tradycyjna, iteracyjna, czy zwinna, z pełną świadomością zalet i ograniczeń każdej z nich.

Metodyka tradycyjna jest niewygodna i zbyt sformalizowana do prowadzenia wielu projektów, które z natury swojej muszą być prowadzone dynamicznie. Na przykład oprogramowanie mobilne, które musi nadążać bardzo szybko za zmianami rynkowymi i poczynaniami konkurencji, w związku z czym proces jego wytwarzania powinien być ukierunkowany na ciągłą zamianę zakresu projektu, jest wprost stworzone do tego, aby wytwarzać go w metodykach „lekkich”, zwinnych. Natomiast skomplikowane oprogramowanie np. do sterowania procesem produkcji energii elektrycznej w elektrowni ma tak wiele sztywnych, z góry ustalonych wymagań wstępnych, które bezwzględnie muszą być zdefiniowane, a następnie zaimplementowane w oprogramowaniu, że wręcz predestynuje to go do wytwarzania w metodyce klasycznej. W praktyce przy tak dużych projektach informatycznych często stosuje się metodyki mieszane: duże części oprogramowania, które zależne są od różnego rodzaju praktycznie stałych uwarunkowań i ograniczeń tworzy się metodyką klasyczną, natomiast części, które powinny szybko reagować na zmiany wymagań (np. obsługa interfejsu użytkownika), mogą i nawet powinny być realizowane w metodyce zwinnej, przy założeniu ciągłej interakcji zespołu wytwórczego z przyszłymi użytkownikami oprogramowania.

# Zakończenie

W projektach informatycznych, których celem jest wytworzenie oprogramowania zakres projektu definiuje co należy wykonać, aby zrealizować specyfikowane wymagania klienta. Jednocześnie zakres określa także czego w projekcie realizować nie trzeba, a także jakie są kryteria odbioru gotowego produktu (specyfikacja tych wymagań, które muszą być zrealizowane, aby produkt mógł być zaakceptowany przez klienta). Zakres wyznacza granice projektu. Nie są to jednak granice stałe podczas całego czasu realizacji projektu. Częstotliwość i skala zmian silnie zależą od tego jakiego rodzaju oprogramowania projekt dotyczy.

Niepoprawnie zdefiniowany zakres projektu, duża częstotliwość zmian zakresu oraz niewłaściwe zarządzanie tymi zmianami, to częste przyczyny niepowodzeń projektów informatycznych. Dlatego tak ważne jest skuteczne zebranie wymagań, zaakceptowanie przez cały zespół projektowy nieuchronności zmian oraz odpowiednie zarządzanie tymi zmianami i ich skutkami, zarówno w sferze zarządzania projektem, jak i w sferze wytwórczej.

Bardzo ważne jest również dobranie odpowiednich metodyk zarządzania projektem i wytwarzania oprogramowania. W zależności od stopnia wstępnego określenia celu i zakresu projektu oraz rodzaju oprogramowania, jakie ma zostać wytworzone w ramach tego projektu, można stosować metodyki tradycyjne, lekkie / zwinne lub mieszane.

Najważniejszym jest jednak, aby w całym cyklu wytwórczym oraz cyklu zarządzania projektem informatycznym nie stracić z oczu celu biznesowego tego projektu. Koncentracja na kliencie i jego potrzebach daje właściwe ramy realizowanym procesom i wydatnie zwiększa prawdopodobieństwo dostarczenia klientowi satysfakcjonującego go produktu.

# Bibliografia:

**Książki:**

Frączkowski Kazimierz, Zarządzanie projektem informatycznym; Projekty w środowisku wirtualnym; Czynniki sukcesu i niepowodzeń projektów, Wrocław, Oficyna Wydawnicza Politechniki Wrocławskiej, 2003, ISBN: 83-7085-731-0

Wysocki K. Robert, dr, Efektywne zarządzanie projektami; Tradycyjne, zwinne, ekstremalne, Wydanie 6, Gliwice, Helion, 2013, ISBN: 978-83-246-6891-5

**Praca zbiorowa:**

A Guide to the Project Management Body of Knowledge: (PMBOK**®** Guide), Project Management Institute Staff, Edition: 5th, Newtown Square; PA USA, Project Management Institute, 2013, ISBN 13: 978-1-935589-67-9; ISBN: 1-935589-67-9

**Słowa kluczowe artykułu: zarządzanie projektem, zakres projektu, zakres oprogramowania, zarządzanie zmianą, parametry projektu**

**Cytaty:**

**Cytat 1:**

**„Szybko, tanio i jednocześnie dobrze to mission impossible projektu informatycznego**. Nie da się wytworzyć oprogramowania wysokiej jakości, jeśli budżet będzie niewystarczający, a czas zbyt krótki”

[Lub, jeśli 170 znaków to zdecydowanie za dużo – w zamian cytat poniższy:]

**„Zakres** definiuje dokładnie granice projektu, czyli co w projekcie ma być wykonane oraz to, co w ramach tego projektu zrealizowane nie będzie”

**Cytat 2:**

„Koncentracja na kliencie i jego potrzebach daje właściwe ramy realizowanym procesom i wydatnie zwiększa prawdopodobieństwo dostarczenia klientowi satysfakcjonującego go produktu”

**Informacje o autorze**

**Ewa Brzeska**

Doświadczony Manager i Konsultant IT. Od 25 lat związana czynnie z branżą informatyczną. W tym czasie uczestniczyła w realizacji wielu projektów, pełniąc rolę analityka, projektanta, architekta i programisty oraz kierownika projektu. Wśród zrealizowanych przez nią projektów są rozwiązania wspomagające zarządzanie różnymi sferami biznesu i produkcji. Ma ponad 15 lat doświadczenia w tworzeniu systemów klasy MES (ang. *Manufacturing Execution System*) dla energetyki i przemysłu.

Zarządza projektami i produktami informatycznymi oraz wytwarzaniem oprogramowania. Zajmuje się konsultingiem i doradztwem IT, wykorzystując przy tym bogate doświadczenie praktyczne oraz wiedzę teoretyczną, zdobytą m.in. w Szkole Głównej Handlowej w Warszawie.

W 2009 roku wraz ze wspólnikiem założyła firmę [Ebitech](http://www.ebitech.pl), która zajmuje się głównie tworzeniem oprogramowania biznesowego na platformy systemowe Apple: iOS i Mac OS X.

Współzałożycielka oraz Członek Zarządu Stowarzyszenia Inżynierii Wymagań.