Uniwersytet Ekonomiczny w Katowicach

Studia Ekonomiczne. Zeszyty Naukowe
Uniwersytetu Ekonomicznego w Katowicach
ISSN 2083-8611 Nr 235 · 2015

Jakub Synowiec

Uniwersytet Ekonomiczny w Katowicach
Wydział Informatyki i Komunikacji
Katedra Badań Operacyjnych
jakub@jakubsynowiec.info

Tomasz Błaszczyk

Uniwersytet Ekonomiczny w Katowicach
Wydział Informatyki i Komunikacji
Katedra Badań Operacyjnych
tomasz.blaszczyk@ue.katowice.pl

PROPOZYCJA MODELOWYCH CECH NARZĘDZIA WSPOMAGAJĄCEGO ZWINNE ZARZĄDZANIE PROJEKTAMI INŻYNIERII OPROGRAMOWANIA

Streszczenie: W artykule opisano próbę analizy zasadności wykorzystania informatycznych narzędzi wspierających zwinne zarządzanie projektami oraz określenia modelowych cech, którymi powinno charakteryzować się takie oprogramowanie, aby zaspokoić oczekiwania konkretnych członków zespołów projektowych w każdej fazie ich realizacji (koncepcyjnej, planowania, produkcji, testowania w fazie operacyjnej). Rozważania prowadzono, opierając się na wynikach pogłębionych wywiadów, przeprowadzonych wśród członków zespołów projektowych realizujących zadania w sektorze inżynierii oprogramowania.

Słowa kluczowe: zarządzanie projektami, metodyki zwinne, inżynieria oprogramowania, narzędzia informatyczne.

Wprowadzenie

W ciągu minionych czterech dekad oprogramowanie komputerowe zmieniło się z narzędzi używanych do analizy i przekształcania danych lub rozwiązywania problemów w produkt sam w sobie. Dostrzegając potencjał tej zmiany, na rynku usług IT powstało wiele firm wytwarzających oprogramowanie służące do pracy, ale także do rozrywki. Globalny dostęp do sieci internet dodatkowo wzmógł te procesy i dziś już prawie każda branża opiera swój model biznesowy na jakimś rodzaju oprogramowania komputerowego. Oprogramowania, które należy wytworzyć, wdrożyć i pielęgnować.

Wczesne realizacje i wdrożenia produktów programistycznych ujawniły wiele przeszkód i problemów, których sposoby przezwyciężania przekształciły

się w dziedzinę inżynierii systemów – inżynierię oprogramowania. Ma ona wspierać definiowanie i nadzorowanie wszystkich aspektów pracy wykonywanej w ramach wytwarzania oprogramowania w celu zapewnienia wysokiej jakości i zgodności z wymaganiami klienta. Osiągane jest to m.in. poprzez zestawy narzędzi i technik, które pozwalają na sprawne monitorowanie i zarządzanie procesami zachodzącymi w trakcie wytwarzania oprogramowania. Takich narzędzi i technik dostarczają np. metodyki opisujące gotowe, ustandaryzowane metody realizacji zadań i rozwiązywania napotykanych problemów. Avison i Fitzgerald definiują metodykę jako zbiór procedur, technik, narzędzi oraz dokumentacji wspomagających zespół projektowy w implementacji nowego systemu informacyjnego [Avison i in., 1995]. Metodyka pomaga ściśle kontrolować procesy rozwoju oprogramowania, aby były one bardziej wydajne i przewidywalne (Avison i in., 1995; Awad, 2005].

Podwaliny metodyk zwinnych powstawały i były promowane w latach 80. XX w. [Weinberg, 1982; Brooks, 1987] jako próba spojrzenia na proces tworzenia oprogramowania w sposób odwrotny do klasycznego, m.in. dopuszczenie, że projekty są podatne na błędy, uwzględnienie klienta we wszystkich fazach projektu oraz promowanie indywidualizmu. Na rozwój metodyk zwinnych miało również wpływ powstanie w latach 90. XX w. ideologii i technologii z nią związanych – *Rapid Application Development* (RAD)[1]. Kolejne lata były czasem, w którym grupy praktyków próbowały przekuć tę ideę i ograniczenia klasycznego podejścia do projektów na nowe narzędzia i techniki – metodyka Scrum[2] (1986), *Dynamic System Development Methodology* (DSDM)[3] (1994), *eXtreme Programming* (XP)[4] (1996), *Feature-Driven Development* (FDD)[5] (1998). Ideologia Agile rozwijała się dynamicznie i była stosowana z coraz większym rozmachem. Punktem przełomowym był luty 2001 r., kiedy w USA został opracowany i podpisany przez 15 sygnatariuszy tzw. Manifest zwinnego tworzenia oprogramowania (*Manifesto for Agile Software Development*), stanowiący deklarację podstawowych wartości i zasad Agile, które w sposób prosty oddają filozofię „zwinności" [Manifesto…, 2001].

---

[1] RAD (*Rapid Application Development*), szybkie wytwarzanie aplikacji, metodyka wytwarzania oprogramowania faworyzująca szybkie prototypowanie i wykorzystywanie gotowych komponentów.

[2] Scrum – zwinna metodyka zarządzania projektami skupiająca się na procesie zarządzania oraz śledzenia realizacji projektu.

[3] DSDM (*Dynamic System Development Method*), zwinna metodyka zarządzania projektami zakładająca stały koszt, czas i jakość oraz wykorzystująca metodę MoSCoW do ustalania priorytetów.

[4] XP (*eXtreme Programming*), programowanie ekstremalne – paradygmat i zwinna metodyka programowania stosowana w małych i średnich projektach wysokiego ryzyka (o dużej złożoności).

[5] FDD (*Feature Driven Development*), zwinna metodyka wytwarzania oprogramowania, skupiająca się na ocenie wartości funkcji z perspektywy klienta.

W niniejszej pracy podjęta została próba analizy na podstawie danych zebranych w wywiadach pogłębionych, przeprowadzonych z członkami zespołów projektowych, dotyczących zasadności stosowania narzędzi do wspomagania zarządzania projektami informatycznymi oraz określenia modelowych cech, które powinny te narzędzia posiadać, aby spełnić oczekiwania poszczególnych członków zespołów projektowych na każdym poziomie realizacji (planowanie, projektowanie, wytwarzanie, testy i zarządzanie).

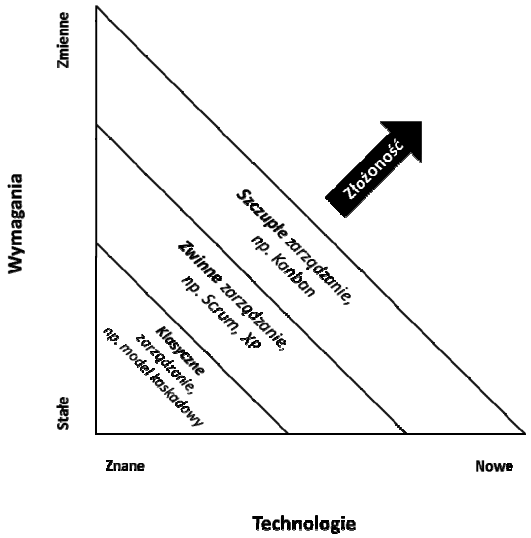
1. Metodologie zarządzania projektami – podejście klasyczne a podejście „zwinne"

W dzisiejszych czasach procesy biznesowe przekształcają się i powstają bardzo szybko. Wraz z tymi zmianami pojawiają się złożone wymagania stawiane oprogramowaniu komputowemu, które ma te procesy wspomagać lub realizować. Ponadto duża i ciągle rosnąca konkurencja między przedsiębiorstwami oraz zwiększające się oczekiwania klienta końcowego wpływają na częste zmiany wymagań oraz priorytetów firm zlecających wykonanie konkretnego oprogramowania. Taka ewolucja rynku spowodowała, że tradycyjne modele zarządzania cyklem życia oprogramowania (*Software Development Life Cycle*, SDLC[6]), zwane również ciężkimi (*heavyweight*), zaczęły przeszkadzać w realizacji projektów i doprowadziły do potrzeby opracowania nowych praktyk, które pozwoliłyby zażegnać powstały kryzys. Przykład klasycznego (metoda kaskadowa) SDLC przedstawia rys. 1. Współczesne podejście, zwane zwinnym lub lekkim (*lightweight*), ma dostarczać rozwiązania dla większości problemów podejścia tradycyjnego [Awad, 2005; Munassar i in., 2010; Brooks, 1987; Lehman i in., 2011].

[6] SDLC (*Software Development Life Cycle*), cykl życia oprogramowania, termin z inżynierii oprogramowania. Struktury SDLC dostarczają i opisują czynności wykonywane na każdym etapie cyklu życia projektu. Istnieje wiele modeli SDLC, m.in. kaskadowy, spiralny, RAD, iteracyjny, metodyki zwinne.

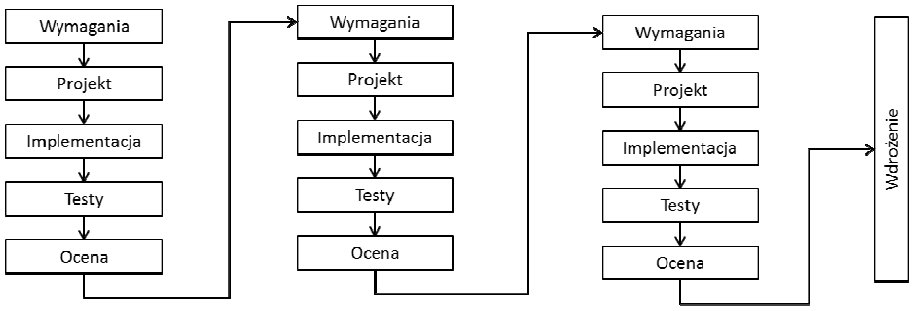
Rys. 1. Cykl życia projektu oprogramowania w ujęciu klasycznym

Projekty informatyczne mające na celu produkcję oprogramowania są projektami twórczymi z pogranicza nauki i sztuki – wykorzystują zaawansowane techniki i języki programowania do urzeczywistniania wizji i marzeń klienta. Taki wysiłek twórczy wymaga poświęcenia większości czasu na myślenie, a tylko znikomej jego części na rzeczywiste programowanie. Wynika to z dużego poziomu złożoności, który, im mniej sprecyzowane oczekiwania klienta i znane technologie, tym jest większy. Trudność polega również na tym, że wymagania ewoluują w trakcie projektu. Klient nie zawsze zdaje sobie sprawę lub potrafi od razu określić i zdefiniować wszystkie swoje potrzeby. Sposób wyboru metodyki odpowiedniej do realizacji projektu ze względu na jego złożoność przedstawia rys. 2. Zauważyć można, że im spodziewana złożoność projektu jest większa, tym wybierana metodyka powinna w mniejszym stopniu skupiać się na zdefiniowanych procesach, dokumentacji i klasycznym SDLC.



Rys. 2. Koncepcja doboru metodyk zarządzania projektem w zależności od jego złożoności

Metodyki zwinne wykorzystują krótkie, powtarzalne cykle składające się z etapów zbierania wymagań, analizy, projektowania, implementacji, testowania i dostarczania częściowo gotowego oprogramowania. Schemat takiej realizacji przedstawia rys. 3. Następnie oczekuje się na akceptację etapu pracy przez klienta i modyfikuje założenia na podstawie jego opinii. Pozwala to na skupienie się na satysfakcji odbiorcy poprzez włączenie go do procesu ciągłej oceny i planowania oprogramowania [Boehm i in., 2004]. Przekłada się to na szybsze dostarczenie gotowego, kompletnego i zgodnego z wymaganiami, czyli **zorientowanego na potrzeby biznesu**, produktu [Awad, 2005; Lehman i in., 2011; Sharama i in.; 2012; Boehm i in., 2004). Według 70% ankietowanych w badaniu Version One [Version One, 2007, 2008, 2012] zespołów projektowych projekty realizowane z wykorzystaniem metodyk Agile charakteryzują się krótszym czasem realizacji.



Rys. 3. Cykl życia projektu z koncepcji zwinnego zarządzania

Zespoły projektowe, które skutecznie wdrożyły zwinne metodyki wskazują, że m.in. aspekty takie jak czas potrzebny na reakcję na zmieniające się wymagania oraz priorytety, produktywność i morale zespołu, jakość wytwarzanego kodu lub ryzyko niepowodzenia projektu uległy poprawie. Większość doświadczonych użytkowników Agile wskazuje zdolność reakcji na zmiany, przejrzystość projektu oraz produktywność jako najistotniejsze zyski wynikające z wdrożenia zwinnych metodyk zarządzania [Version One, 2012]. Istnieje wiele metodyk, za pomocą których można zwinnie realizować projekty informatyczne. Metodyki te koncentrują się na innych aspektach SDLC, ale wszystkie zwinne metodyki (np. XP, ASD, DSDM, Scrum, Crystal, FDD) zarządzania projektami wykształciły się na gruncie tych samych wartości i zasad, które zostały opisane w manifeście Agile [Manifesto…, 2001].

Tabela 1. Porównanie modelu zwinnego i klasycznego z perspektywy relacji pomiędzy wytwórcą a klientem

	Podejście zwinne	Podejście klasyczne
Rozpoczęcie	Określenie tylko ogólnych założeń	Konieczność wyspecyfikowania wszystkich wymagań
Wytwarzanie	Aktywna współpraca, zapewnianie ciągłych informacji zwrotnych, opiniowanie i szczegółowe specyfikowanie wybranych wymagań	Ewentualnie odbiory cząstkowe
Zakończenie	Odbiór gotowego produktu, brak niespodzianek i niezgodności, brak konieczności walidacji	Walidacja i zatwierdzenie końcowego produktu pod względem zgodności z określonymi wymaganiami

Tabela 2. Porównanie głównych cech modelu zwinnego i klasycznego

Podejście zwinne	Podejście klasyczne
Nieformalna i przyrostowa architektura	Architektura zostaje stworzona i udokumentowana przed pracami programistycznymi
Współdzielona własność kodu wśród zespołu	Każdy programista jest odpowiedzialny za przydzielony fragment
Ciągła integracja	Integracja realizowana na koniec każdego kamienia milowego
Koncentracja na realizacji wybranych funkcji w krótkich iteracjach	Koncentracja na realizacji modułów (części architektury) w kamieniach milowych
Wykorzystanie praktyk programistycznych (TDD, refaktoryzacja, wzorce itp.)	Nie wymusza dobrych praktyk programistycznych
„Lekkie" procesy i dokumentacja	„Ciężkie" procesy i dokumentacja
Wymaga szerokiej wiedzy dotyczącej wykorzystywanych technologii i jej współdzielenia	Bazuje na niewielkiej grupie specjalistów, którzy nadzorują kompletny projekt. Reszta zespołu nie musi posiadać rozległej wiedzy i doświadczenia
Główna rola: programiści	Główne role: architekci i projektanci
Polityka otwartych drzwi: uczestnicy projektu są namawiani do bezpośredniego kontaktu z biznesem, QA i kierownictwem w dowolnym momencie. Motywacja do dzielenia się opiniami	Tylko wyznaczone osoby mogą kontaktować się z biznesem. Wymiana informacji następuje głównie na początku projektu i przed czasami po zrealizowaniu kamieni milowych

2. Przebieg i wyniki badania

Grupa badawcza składała się z osób odpowiedzialnych za różne poziomy realizacji zadań projektowych (planowanie, projektowanie, implementacja, testy, wdrożenie, zarządzanie i wsparcie) w organizacjach o różnym profilu działalności. W trakcie przeprowadzanych rozmów zadawane pytania miały na celu uzyskanie informacji takich jak:

- czy organizacja zna i stosuje jakieś metodyki zarządcze w swoich projektach, a jeśli tak, to jakie i czy preferuje dla tych projektów podejście klasyczne czy zwinne,
- czy zespoły realizujące projekty korzystają z narzędzi do wspomagania procesów zarządczych oraz jakie cechy, funkcje i zastosowania są wymagane przez kadrę zarządzającą, a jakie przez samych użytkowników,
- czy osoby odpowiedzialne za realizację projektów odczuwają jakieś utrudnienia i braki w stosowanych metodach oraz narzędziach, czy organizacja planuje im jakoś przeciwdziałać,
- jakich informacji oczekuje kierownictwo odnośnie do prowadzonych projektów, jakie informacje są niezbędne do działania innych obszarów organizacji,
- w jaki sposób można usprawnić istniejące procesy i wykorzystywane metodyki.

Zebrane informacje miały na celu określenie poziomu wiedzy badanej osoby odnośnie do metodyk i narzędzi, świadomości zespołu, organizacji i klientów oraz specyfiki pracy w środowisku, w którym dana osoba realizuje projekty. Oprócz tego konieczne było uzyskanie informacji określających kontekst (charakter działalności) przedsiębiorstwa, tj.:

- jaki charakter mają projekty realizowane w organizacji,
- kto jest typowym klientem w realizowanych projektach,
- jakie są oczekiwania klienta wobec realizowanych projektów,
- jak jest pozyskiwany klient,
- jak są pozyskiwane projekty do realizacji,
- czy są odgórnie narzucone w organizacji sposoby realizacji projektów (metodyki, procedury)

oraz kontekst pracownika:

- zakres odpowiedzialności,
- stanowisko,
- posiadaną wiedzę (jawną i cichą),
- znajomość metodyk, narzędzi i praktyk
- preferencje odnośnie do sposobu pracy.

Razem dane te pozwoliły na umocowanie informacji odnośnie do preferencji metodyk i cech narzędzi względem potrzeb organizacji, sposobu pracy, rodzaju i pochodzenia klienta. Pozwoliły również na stworzenie profilu oczekiwań odnośnie do metodyk i narzędzi ze względu na pełnioną w organizacji funkcję i zakres odpowiedzialności.

Przeprowadzone wywiady pogłębione pozwoliły na zebranie opinii osób, które pełnią różne funkcje w organizacjach lub pracują samodzielnie. Informacje dotyczące zakresu odpowiedzialności i miejsca pracy pozwalają na spojrzenie z szerszej perspektywy na odpowiedzi respondentów dotyczące cech i funkcji narzędzi wspomagających ich pracę oraz oczekiwań wobec realizacji samych procesów. Uzyskane dane pozwalają również ocenić poziom wiedzy rozmówców odnośnie do narzędzi i metodyk wspomagających realizację projektów. Wskazują również na różnice w oczekiwaniach i postrzeganej wartości różnorakich danych, gromadzonych w trakcie realizacji projektu. Gdy dokonuje się analizy udzielanych odpowiedzi, możliwe jest pogrupowanie wymienianych przez respondentów funkcji i cech aplikacji w zbiory. Dotyczą one uniwersalnych aspektów takich narzędzi i mogą stanowić wskazówkę przy budowie potencjalnego systemu oceny dostępnych na rynku narzędzi. Na podstawie przeprowadzonych rozmów wszystkie wskazywane wady, zalety oraz istotne funkcje wykorzystywanych narzędzi można przypisać do jednej z kategorii:

Dostosowalność – cecha ta oznacza możliwość dostosowania zaimplementowanych w narzędziu metodyk (lub zdefiniowania własnych) do wariantów wykorzystywanych w przedsiębiorstwie. Aplikacja musi pozwalać na zmianę domyślnych stanów zadań, definiowanie nowych, określanie zależności pomiędzy nimi lub tworzenie całych definicji procesów zachodzących podczas realizacji projektów. Podejście takie spotykane jest w narzędziach do modelowania procesów biznesowych wykorzystywanych do wspierania zarządzania procesami biznesowymi (*business process management*, BPM). Zawiera się tutaj również funkcjonalność dostosowania widoków aplikacji do potrzeb konkretnej osoby (stanowiska – zestawu odpowiedzialności biznesowych), ukrywania lub wyświetlania pewnych danych itd. Aplikacja powinna być „świadoma" kontekstu pracy i oferować widok zoptymalizowany dla konkretnej akcji w konkretnej chwili.

Rozszerzalność – oznacza, że w aplikacji został przewidziany system modułów pozwalających na zmianę i rozszerzenie funkcjonalności narzędzia. Przekłada się to na możliwość tworzenia interfejsów komunikacji z innymi narzędziami, agregowania lub udostępniania danych, udostępniania konektorów dla innych usług i aplikacji oraz łatwego dostosowywania i wzbogacania istniejących funkcji aplikacji.

Dostęp do danych – rozumiany jako łatwość w dostępie do zgromadzonych w systemie informacji, ich eksport i przetwarzanie. Badane osoby na stanowiskach kierowniczych wskazywały, że, bez względu na ilość zaimplementowanych w narzędziu funkcji raportowania, nic nie jest w stanie skutecznie zastąpić możliwości samodzielnego przetwarzania „surowych" danych.

Modularność – związana z rozszerzalnością i dostosowalnością, ale rozumiana jako możliwość dowolnego konfigurowania działania systemu poprzez wybór i konfigurację działających modułów. Oczekiwanie respondentów dotyczy również rozwoju przez producenta funkcji i naprawy błędów na poziomie pojedynczych modułów, co ma gwarantować szybszą reakcję i brak konieczności aktualizacji całego narzędzia.

Szybkość działania, skalowalność – narzędzie musi działać sprawnie i wydajnie oraz poprawnie skalować się wraz ze wzrostem ilości użytkowników i wolumenu danych. Wiele dostępnych na rynku aplikacji nie jest przewidzianych do pracy z ilością informacji generowanych przez kilka współpracujących zespołów, a jest to kluczowe dla uniknięcia frustracji wynikającej z obsługi takiego narzędzia.

Wielkość dodatkowej pracy (narzut) – cecha wskazywana przez prawie każdego respondenta. „Czas to pieniądz" i nikt nie chce wykonywać dodatkowej pracy, która nie przybliża go do osiągnięcia założonego celu. Procesy realizowane przez systemy wspomagające nie powinny absorbować uwagi użytkownika i wymagać jedynie minimalnej interakcji. Duża część czynności powinna być automatyzowana.

Zdolność do komunikacji z innymi systemami – grupa cech i funkcji, która jest związana z wyżej wymienionymi modularnością i rozszerzalnością. Narzędzie wspomagające realizację projektów najczęściej ma postać systemu zarządzania zadaniami. Wokół tych zadań gromadzone są dodatkowe informacje, wykorzystywane zazwyczaj na innych poziomach hierarchii zarządczej, niż są zbierane. Respondenci wskazują konieczność rozszerzania już wykorzystywanych narzędzi, tj. IDE, klientów pocztowych, komunikatorów, aplikacji biurowych, systemów wiki i innych o zdolność wymiany informacji z narzędziem do wspomagania realizacji projektu.

„Doświadczenie użytkownika" (*User eXpiriences*, UX) – najobszerniejsza grupa, która zawiera wszystkie cechy i funkcje dotyczące łatwości i przyjemności korzystania z narzędzia. Zaklasyfikować można tutaj wszystkie życzenia respondentów dotyczące czytelności i przejrzystości interfejsów, łatwości wprowadzania danych, ergonomii, użyteczności i poziomu satysfakcji z użytkowania aplikacji.

Badane osoby oczekują więc, aby aplikacja była prosta i przyjemna w użytkowaniu, działała wydajnie i dobrze się skalowała wraz ze wzrostem ilości użyt-

kowników i wprowadzanych informacji, wymagała od użytkownika niewielkiej uwagi, a większość procesów była zautomatyzowana. Dane powinny łatwo agregować się z wykorzystywanych narzędzi za pośrednictwem konektorów już istniejących lub możliwych do stworzenia samodzielnie przez użytkowników. Moduł raportowania może być pominięty, ale musi istnieć możliwość eksportu wszelkich gromadzonych w narzędziu danych. Aplikacja musi również „inteligentnie" rozpoznawać aktualny kontekst pracy i dostarczać tylko niezbędnych informacji (interfejs nie może być przeładowany danymi). Zastanawiające jest to, że większość badanych osób nie ma preferencji odnośnie do potencjalnej ceny lub modelu dostarczonej aplikacji. Tylko w jednej z rozmów pojawiły się jasno określone wytyczne, które były wynikiem polityki przedsiębiorstwa. Na tej podstawie można wnioskować, że potencjalni użytkownicy za wygodę i jakość byliby w stanie zapłacić. Istotność poszczególnych cech przedstawia rys. 1. Istotność poszczególnych cech w „idealnej aplikacji" do wspomagania realizacji projektów informatycznych wskazują uśrednione wartości powyższych wag. Dane te zawiera tabela 3. Największa wartość odchylenia standardowego występuje dla dostępu do danych. Potwierdza to wcześniejsze wnioski dotyczące zmiany wagi przykładanej do gromadzenia i analizy danych wraz ze wzrostem zakresu obowiązków i funkcją w zespole realizującym projekt. Najmniejsza zmienność dotyczy szybkości działania i skalowalności aplikacji. Oznacza to, że oczekiwania wszystkich respondentów są zbliżone – aplikacja musi działać szybko i dobrze się skalować wraz ze wzrostem liczby osób z niej korzystających oraz wolumenem obsługiwanych danych.

Tabela 3. Istotność cech narzędzia – średnie z ocen respondentów

Cecha	średnia	σ
Dostosowalność	2,0	0,63
Rozszerzalność	1,8	0,75
Dostęp do danych	2,7	1,03
Modularność	2,2	0,75
Szybkość działania, skalowalność	3,7	0,52
Wielkość dodatkowej pracy (narzut)	4,5	0,55
Zdolność do komunikacji z innymi systemami	3,0	0,63
UX	4,2	0,75

Kolejnym z wniosków powstałych na podstawie przeprowadzonych rozmów jest wskazanie dominującego typu wykorzystywanych narzędzi – systemu obsługi zadań. Każda z ankietowanych osób wykorzystuje jakiś rodzaj aplikacji pozwalającej na gromadzenie i zarządzanie listami zadań. Większość respondentów ma styczność z systemami, które wywodzą się z narzędzi do obsługi zgłoszeń lub błędów wzbogaconych o funkcje i mechanizmy pomocne w zarządzaniu projektami. Nie jest jednak pewne, czy wykorzystywanie takiego systemu jako rdzenia aplikacji jest pożądane. Być może powinna być to tylko jedna z funkcji. Istotna, ale nadal tylko funkcja dodatkowa. Ciekawym pomysłem zdaje się narzędzie opisane przez jedną z osób – rozbudowany klient poczty elektronicznej. Narzędzie takie jako podstawę swojego działania wykorzystuje komunikację – wymianę wiadomości elektronicznych.

Rys. 4. Uśredniona istotność cech

Rozmiar projektów również ma znaczenie dla potrzeby wykorzystywania narzędzi, metodyk i praktyk. Im projekt mniejszy, tym łatwiej jest go obsługiwać „na papierze". Wraz ze wzrostem liczby wymagań, złożoności, liczby członków zespołu trudniejsze staje się zarządzanie takim projektem przy wykorzystaniu klasycznych metod i narzędzi. Respondenci, którzy realizowali skomplikowane i rozbudowane projekty, każdorazowo dostrzegali konieczność dokładniejszej kontroli nad ich realizacją, szczegółowego monitorowania i gromadzenia

danych oraz weryfikowania postępu prac. Można więc uznać, że im większa złożoność i niepewność projektu, tych większa potrzeba sprawowania kontroli. Ma to zapewniać w przyszłości możliwość odniesienia się do już posiadanych doświadczeń – czy to zdobytych, czy też zgromadzonej wiedzy. Według badanych osób przekładać się to będzie na zwiększenie dokładności i szybkości szacunków oraz łatwości rozwiązywania potencjalnych problemów. Potwierdza to więc informacje zgromadzone na podstawie studiów literatury. Pozyskane informacje pozwalają określić preferencje co do wykorzystywanych w projektach metodyk. Wszystkie badane osoby korzystają z jakiegoś wariantu metodyki zwinnej. Respondenci, którzy uczestniczą w projektach koncentrujących się na wytwarzaniu nowych produktów, skupiają się na wariantach metodyki Scrum. Najczęściej są to wybiórczo stosowane praktyki kilku metodyk, rzadko pełna metodyka. Być może w przypadku przedsiębiorstw posiadających przeszkolone zespoły metodyki są wykorzystywane w pełni. W przypadku tego badania próba była zbyt mała, aby to określić. Druga grupa badanych ma kontakt z projektami polegającymi na utrzymaniu i pielęgnacji istniejącego oprogramowania – wsparciu. Jest to główna odpowiedzialność respondentów lub ich zespołów, a wytwarzanie nowego oprogramowania następuje epizodycznie i ma niższy priorytet niż poprawianie błędów lub rozwój istniejących funkcji. W tych przypadkach popularnością cieszyły się warianty metodyk szczupłych. Znacznie częściej stosowane były pełne metodyki (Kanban) lub ich warianty (Scrum-ban). Wybór podejścia Lean może być podyktowany koniecznością ciągłej reakcji na zgłoszenia od innych pracowników, które dotyczą systemów produkcyjnych. Na aplikacjach tych niejednokrotnie opiera się model biznesowy przedsiębiorstw i musi być zagwarantowane ich nieprzerwane, poprawne działanie. Nie da się więc zastosować podejścia iteracyjnego i produkować nowych wersji co ustalone interwały czasu – konieczne jest działanie „z doskoku", a tutaj świetnie sprawdza się właśnie filozofia Lean.

Podsumowanie

Zrealizowane badania pozwoliły na potwierdzenie zasadności stosowania narzędzi do wspomagania zarządzania projektami informatycznymi oraz na określenie cech modelowej aplikacji do wspomagania zarządzania projektami informatycznymi. Zebrane dane pozwoliły również na określenie preferencji odnośnie do metodyk zarządczych, które są wybierane przez zespoły realizujące projekty informatyczne. Pomimo dostępności wielu narzędzi wspomagających,

których część została wymieniona w tej pracy, żadne z nich zdaje się w pełni nie zadowalać osób, z którymi zostały przeprowadzone rozmowy. Być może wynika to z genezy większości testowanych aplikacji, które wywodzą się z systemów zarządzania zgłoszeniami. Możliwe, że narzędzia kolejnej generacji, które od podstaw projektowane będą nie jako systemy zarządzania zgłoszeniami i błędami, ale jako aplikacje do wspomagania projektów, spełnią oczekiwania członków zespołów projektowych.

Literatura

Avison D.E., Fitzgerald G. (1995), *Information Systems Development: Methodologies, Techniques and Tools*, Blackwell Scientific Publications, Oxford.

Awad M.A. (2005), *A Comparasion between Agile and Traditional Software Development Methodologies. Report for the Honours Programme of the School of Computer Science and software Engineering*, The University of Western Australia.

Boehm B., Turner R. (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley Professional.

Brooks F.P.Jr. (1987), *No Silver Bullet – Essence and Accidents of Software Engineering*, „Computer", Vol. 20, Iss. 4.

Lehman T. J., Sharma A. (2011), *Software Development as a service: Agile Experiences*, SRII Global Conference.

Manifesto for Agile Software Development (2001), http://agilemanifesto.org.

Munassar N.M.A., Govardhan A. (2010), *A Comparasion between Five Models of Software Engineering* , „International Journal of Computer Science Issues", Vol. 7, Iss. 5.

Sharama S., Sarkar D., Gupta D. (2012), *Agile Processes and Methodologies: A Conceptual Study*, „International Journal of Computational Science and Engineering", Vol. 4, Iss. 5.

Version One, 3rd Annual Survey: The State of Agile Development, 2008, http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf.

Version One, 7th Annual Survey, 2012, http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf.

Version One, Agile development: Result Delivered, 2007, http://wwwversionone.com/pdf/AgileDevelopment_ResultDelivered.pdf.

Weinberg G.M. (1982), *Overstructured Management of Software Engineering*, ICSE, Proceedings of the 6th International Conference on Software Engineering, IEEE.

THE PROPOSAL FOR MODEL FEATURES OF THE TOOL SUPPORTING AN AGILE PROJECT MANAGEMENT IN SOFTWARE ENGINEERING

Summary: In this paper we described an attempt of the analysis the legitimacy of the use of tools to support project management and to determine the characteristics of the model, which should have the tools to meet the needs of post-specific members project teams at every level of implementation (planning, design, production, testing and management). The data we used was collected by in-depth interviews conducted with members of the software engineering project teams.

Keywords: project management, agile methods, software engineering, software tools.