

Autoreferat

1. Obszar badań, uzasadnienie i cel pracy

W przedłożonym autoreferacie nakreślono wkład autorki w rozwój **inżynierii oprogramowania** w zakresie **doskonalenia jakości wyrobu programowego i procesu** jego wytwarzania. Uzyskanie wysokiej jakości oprogramowania należy do kluczowych zadań uprawianej przez autorkę dyscypliny. Szokująco niski procent udanych i wdrożonych systemów informatycznych (w Wielkiej Brytanii w granicach 16–34% [British Computer Society]) ukazuje, że stosowane w praktyce metody nie są zadowalające.

Od lat trwają poszukiwania skutecznych metodyk wytwarzania oprogramowania. Jednak inżynieria oprogramowania rozwija się głównie na zamówienia wojskowe, a opracowane metody stosuje się także w budowie oprogramowania do celów cywilnych. Zdaniem autorki, występuje potrzeba przeorientowania procesu i strategii wytwarzania oprogramowania przez przypisanie najwyższego priorytetu spełnieniu wymogów jakości wyrobu programowego z punktu widzenia jego rzeczywistych użytkowników.

Ideą przewodnią przedłożonych propozycji jest **upodmiotowienie użytkowników** przez ich aktywne włączenie do współpracy z autorami oprogramowania. Innymi słowy, określając proponowaną strategię i umożliwiając jej wdrożenie narzędzia, autorka określa wpływ człowieka na informatyzowane procesy. Podejmowane prace odniesiono głównie, choć nie ograniczono, do interakcyjnych systemów programowych budowanych dla instytucji publicznych (jak urzędy, szpitale, sądy, etc.) [4, 5] oraz do systemów eksperckich kierowanych do określonych grup specjalistów, nie-informatyków [5, 10]. Wspólne cechy uwzględnianej kategorii programów obejmują: długi czas życia, rozwój aż do wycofania wyrobu, interakcyjność (docelowo interaktywność), korzystanie z utrzymywanych latami danych, wymagany wysoki poziom bezpieczeństwa oraz poprawności danych. Przyjęte rozwiązania informatyczne wpływają decydująco na zakres i poziom jakości usług świadczonych wobec interesantów danej organizacji. Uwzględniono, że użytkownicy rozważanej klasy systemów zgłaszają nowe wymagania po zaspokojeniu potrzeb niższego poziomu, zgodnie z hierarchią potrzeb według Masłowa.

Z systemami IT wkraczamy na teren *ekonomii behawioralnej* – zgodnie z jej zasadami firmy producentów troskliwie śledzą, w jaki sposób ludzie stosują jej wyroby. Tę tendencję proponuje się przenieść do firm informatycznych. Jeszcze na przełomie XX i XXI wieku lansowano wygodny dla firm programowych pogląd, że produkt IT to rodzaj infrastruktury/towaru (*commodity*), przydatnej w *back office* do generowania, a następnie analizowania raportów. Innymi słowy uważano, że odbiorca kupi gotowy system i będzie z niego korzystać, jak z telefonów lub energii elektrycznej. Ten pogląd podzielali zwolennicy metod twardych – za pomocą oprogramowania zarządza się wszelkimi zasobami: informacjami, ludźmi, maszynami, sklepami itd. Dziś uważa się, że produkty IT, ludzie, talenty nie są tylko zasobami czy towarem, umożliwiają bowiem generowanie nowych pomysłów.

Użyteczność systemów informatycznych sprowadza się głównie do *efektywnej obsługi procesów* realizowanych w organizacji nabywcy. Kreowanie wiedzy o tych procesach jest domeną nie tylko dostawców oprogramowania, ale także użytkowników. Przykłady na to znajdują się w wywiadach M. S. Hopkinsa zamieszczonych w *MIT Sloan Management Review* w lutym 2010 [Hopkins_a, Hopkins_b]. W [Hopkins_b] podany jest przykład innowacyjnego podejścia Tesco do wdrażania systemów informatycznych (s. 4). Otóż proces i system IT ma być tańszy, ułatwiać życie klientom, a także być łatwiejszy do obsługi dla pracowników (ten ostatni czynnik zwykle nie jest na liście priorytetów dostawców). Z kolei w [Hopkins_a] pokazana jest siła świadomych użytkowników. Kiedy nowy system Cisco nie wspierał platformy MacIntosh 10000 użytkowników stworzyło własne Wiki, na którym rozpowszechniano opracowane sposoby pokonywania trudności instalacji i współdziałania z oprogramowaniem linuksowym (s. 4). Takie małe usprawnienia – mikro-innowacje (a mogą ich być setki i tysiące) napędzają wzrost ekonomiczny i są domeną, przede wszystkim, użytkowników.

Produkt IT ma nie tylko działać, ale **generować wartość dodaną** dla firmy, minimalizować frustrację jej pracowników i wspomagać ich kreatywność (w przeciwieństwie do bezdusznego wykonywania procedur). Trzeba odróżniać *koszt* od *wartości* wyrobu programowego [Ojala].

Celem prowadzonych prac, a zarazem podjętym zadaniem badawczym, jest opracowanie kompleksowej **strategii doskonalenia wyrobu programowego przez włączenie użytkowników** w proces jego wytwarzania. Kontakty z odbiorcami nie powinny ograniczać się do fazy uściślenia wymagań, aczkol-

wiek już takie zaangażowanie przynosi korzystne rezultaty. Praca nad przyjaznym dla użytkownika interfejsem to stanowczo za mało. Do procesu wytwarzania innowacyjnego wyrobu programowego włączani będą jego przyszli użytkownicy, współtworząc rozwiązania i usługi z użyciem nowych technologii. Wytwarzanie oprogramowania zmierza do personalizacji udostępnianej informacji i częściowo cech wyrobu programowego. Poziom satysfakcji użytkowników staje się priorytetowy dla producenta oprogramowania. Rozwinięcie zaproponowanej strategii polega na określeniu zbioru wymaganych procesów składowych, zdefiniowaniu wymaganych procedur i pożądanym do ich prawidłowego wykonania kompetencji pracowników oraz zapewnieniu niezbędnej infrastruktury wspomagającej.

W rozprawie habilitacyjnej podjęto próbę wypośrodkowania między metodykami twardymi (o wojskowym rodowodzie, ale zapewniającymi opanowanie procesu wytwarzania) a zwinnymi (inaczej miękkimi) przez określenie wymaganych procesów składowych i stosowanie zasad metod zwinnych do sprawdzonych modeli procesu wytwórczego. Opracowana strategia wykazuje charakterystyczne dla metod zwinnych ukierunkowanie na produkt i wytwarzających go ludzi. W rozprawie [5] opisano zbudowane **prototypy** elementów składowych **infrastruktury wspomagającej** wytwarzanie oprogramowania zgodnie z opracowaną strategią. Ocenę procesu wytwarzania oprogramowania uzależniono od poziomu **dojrzałości współpracy z użytkownikami**.

Zakładana współpraca z użytkownikami jest także odpowiedzią na pojawiające się zagrożenia natury etycznej, dehumanizację pracy biurowej, makdonaldyzację wielu sfer działalności człowieka wspomaganą przez informatyzację w skali masowej. Zgodnie z podejściem ekologicznym [Arbaoui, Wastell i inni] przyjęto, że głównymi elementami otoczenia systemu programowego są ludzie, a zatem w projektowaniu oprogramowania powinno się uwzględniać wartości ogólnoludzkie [Friedman]. Uznano konieczność włączenia etyki do zarządzania przedsięwzięciem programowym [Gotterbarn; Rogerson i Gotterbarn].

Problematyka roli użytkowników w przedsięwzięciu programowym wpisuje się w nurt prowadzonych współcześnie badań, prezentowanych na łamach czołowych czasopism informatycznych z listy filadelfijskiej, w tym: ACM Computing Surveys, Advanced Engineering Informatics, Behaviour & Information Technology, Communications of the ACM, Decision Support Systems, IBM Systems Journal, Information & Management, IEEE Transactions on Engineering Management, International Journal of Human-Computer Studies, Journal of Management Information Systems, MIS Quarterly, MIT Sloan Management Review, Science and Engineering Ethics.

2. Określenie niektórych pojęć stosowanych w pracy (w porządku alfabetycznym)

Inżynieria oprogramowania (*software engineering*) – dyscyplina inżynierska skoncentrowana na efektywnym rozwoju wysokiej jakości systemów programowych [Sommerville s. 4]; dyscyplina, która adaptuje inżynierskie podejście do tworzenia, rozwoju i konserwacji oprogramowania, koncentruje wysiłki na usatysfakcjonowaniu użytkowników przez spełnienie wymagań szerokiego ich spektrum, określa niezbędne procesy oraz dostarcza metodologie, narzędzia i standardy (techniczne oraz prowadzenia przedsięwzięć programowych) z uwzględnieniem aspektów etycznych [definicja własna].

Jakość (*quality*) – stopień, w jakim zbiór inherentnych właściwości spełnia wymagania [PN-ISO 9000 p. 3.1.1], inaczej zdolność zbioru nieodłącznych charakterystyk wyrobu, systemu lub procesu do spełnienia wymagań klientów lub innych zainteresowanych stron.

Jakość oprogramowania (*software quality*) – całość właściwości (charakterystyk) danej jednostki programowej, decydujących o jej zdolności do spełnienia stwierdzonych i implikowanych potrzeb [ISO/IEC 9126]; zdolność wyrobu programowego do satysfakcjonowania stwierdzonych i implikowanych potrzeb w określonych warunkach [ISO/IEC 25000].

Jakość użytkowa (*quality in use*) – stopień, w jakim produkt używany przez określonych użytkowników spełnia ich potrzeby, aby osiągnąć określone cele w zakresie efektywności, produktywności i satysfakcji w określonym kontekście stosowania [ISO/IEC 9126 p. B.23, ISO/IEC 25000].

Jakość wewnętrzna (*internal quality*) – całość atrybutów wyrobu, które określają jego zdolność do spełnienia stwierdzonych i implikowanych potrzeb podczas jego stosowania w określonych warunkach [ISO 14598-1 p. 4.15, ISO/IEC 9126 p. B.15]; postrzegana i stosowana przez projektantów programistów.

Jakość zewnętrzna (*external quality*) – całość charakterystyk wyrobu programowego widziana z zewnątrz, to jest podczas testów; określona poziomem, w jakim wyrób spełnia stwierdzone i implikowane potrzeby w określonych warunkach [ISO 14598-1 p. 4.7 i p. 4.15, ISO 9126/IEC p. B.7].

Klient (*customer*) – organizacja nabywcy oprogramowania, reprezentowana zwykle przez menadżera podpisującego umowę i faktury.

Metodyka – zbiór zasad dotyczących sposobów wykonywania jakiejś pracy lub trybu postępowania prowadzącego do określonego celu [Słownik wyrazów obcych]; w kontekście rozważanej pracy pojęcie metodyki obejmuje zbiór metod stosowanych łącznie do wytwarzania oprogramowania.

Model jakości (quality model) – zbiór właściwości oraz powiązań między nimi, które razem stanowią podstawę do określenia wymagań jakościowych oraz ewaluacji jakości [ISO 9126/IEC p. B24].

Oprogramowanie (software) – zbiór programów, procedur, reguł stosowania oraz dokumentów związanych systemem przetwarzania informacji [ISO 9126/IEC p. B.28].

Proces wytwórczy oprogramowania (software process) – zbiór procesów, które są stosowane w organizacji dostawcy do planowania, zarządzania, wykonywania, monitorowania, nadzoru i ulepszania czynności odniesionych do oprogramowania [ISO/IEC 15504 p. 3.45].

Użytkownik (user) – osoba korzystająca bezpośrednio lub pośrednio z wyrobu programowego.

Użytkownik bezpośredni (direct user) – pracownik korzystający bezpośrednio z wyrobu programowego podczas swej pracy.

Użytkownik pośredni (indirect user) – klient lub interesant organizacji stosującej oprogramowanie.

Wyrób programowy (software product) – dostarczany użytkownikowi zestaw programów komputerowych, procedur oraz dokumentów związanych i danych [ISO 14598-1, ISO/IEC 25000].

3. Tło prowadzonych prac z odniesieniem do innych metodyk inżynierii oprogramowania

W produkcji oprogramowania dominują **metodyki twarde**, opracowane do budowy oprogramowania dla potrzeb armii i przemysłu zbrojeniowego, oparte na specyfikacji procesu i szczegółowych planach wytwarzania, a następnie rygorystycznym ich przestrzeganiu. Stosowanie tych metodyk, a często także budowę oprogramowania w warunkach akademickich, charakteryzuje ukierunkowanie na wykorzystanie możliwości obranego narzędzia oraz izolacja autorów oprogramowania od jego użytkowników. Z metodyk twardych w prezentowanej przez autorkę strategii przejęto konieczność określenia zbioru wymaganych procesów, definiowanie wybranych procedur, stosowanie miar oceny procesu oraz wprowadzenie modelu dojrzałości relacji z użytkownikami wzorowanego na CMMI® [CMMI].

Oprogramowanie tworzone metodami twardymi często nie spełnia potrzeb użytkowników. Obok rozmaitych niedogodności i wad lekceważone są wymogi odbiorcy w zakresie wystarczającego poziomu weryfikacji danych. Kwoty wydawane w USA na poprawę funkcjonalności i danych sięgają setek miliardów dolarów [Tiwana i Keil]. Błędne dane są rezultatem programów, których nie wyposażono w wystarczające mechanizmy testowania i korekty danych, tak pożądane przez użytkowników. Powierzchnie testowanie wykonywane na przypadkowych danych nie wykrywa tych wad.

Pewne elementy zwinnego podejścia wykazuje brytyjski standard PRINCE2 [PRINCE2] do zarządzania przedsięwzięciem programowym (rozwijany od 1996 roku przez rządową agencję Central Computing and Telecommunication Agency), który reprezentuje podejście procesowe. Zakłada się w nim zaangażowanie w projekt najwyższego kierownictwa strony zamawiającej, które ustanawia i wybiera Komitet Sterujący (*Project Board*). Zadaniem tego Komitetu jest nadzorowanie prowadzonego projektu i wybór jego kierownika (*Project Manager*). W metodyce PRINCE2 projekt i jego etapy muszą być zamykane w sposób uporządkowany i kontrolowany. Komitet Sterujący jest informowany w raportach o stanie projektu i postępie prac, a na tej podstawie decyduje o kontynuowaniu prac i przejściu do kolejnego etapu. Oczekiwania jakościowe są określone w zleceniu i założeniach (opisie) projektu. Wykonywane przeglądy jakości polegają na sprawdzeniu, czy produkt spełnia kryteria jakości zadane w opisie projektu i nie wykazuje braków lub niezgodności. W skład Komitetu Sterującego wchodzi Główny Użytkownik (*Senior User*), który niejako reprezentuje wszystkich użytkowników. Choć w metodyce PRINCE2 podkreśla się potrzebę utrzymywania komunikacji z reprezentantami tzw. zainteresowanych stron (*interested parties*), to jednak zwoływane spotkania informacyjne nie mają mocy wiążącej i bywają odbierane jako bezproduktywne. Zwykli użytkownicy nie mają realnego wpływu na kształt projektu. Metodyka ta uważana jest za zbiurokratyzowaną i czasochłonną, jak inne metodyki twarde.

Bardziej zwinną, nawiązującą do PRINCE2 w zakresie prowadzenia projektu jest opracowana w Politechnice Poznańskiej metodyka XPRINCE [XPRINCE]. Cechuje ją identyfikacja i podział ról w procesie. Do XPRINCE włączono wybrane praktyki z XP: testy jednostkowe i akceptacyjne, projektowanie przypadków testowych przed kodowaniem, założenie wielu iteracji i przyrostów funkcjonalności oraz ciągłą integrację. Etapy są zbliżone do zalecanych w RUP (*Rational Unified Process* [RUP]).

Na kształt stosowanych metodyk wpływa upowszechnianie **standardów jakości**. Konkurencja firm wytwarzających oprogramowanie wymaga wykazania dbałości o jakość wyrobów programowych. W dużych i średnich firmach programistycznych stopniowo następuje certyfikacja systemów jakości, zgodnie z rodziną standardów ISO 9000. Z zaleceń standardów ISO przejęto w tej pracy założenie dzia-

łania w pętli jakości, to jest ustawiczne doskonalenie procesu i wyrobu, zasadę zaangażowania ludzi (wszyscy współtworzą jakość) oraz imienną odpowiedzialność za powierzone zadania. Rekomendowaną w treści ISO 9000 zasadę ukierunkowania na klienta przełożono na ukierunkowanie na potrzeby i oczekiwania szerokiego grona użytkowników. Rozwinięto zarządzanie relacjami z klientem (CRM), obejmując nim systematyczne relacje z użytkownikami. Wprowadzono w życie zalecenia pomiaru zadowolenia klienta i podejmowania decyzji na podstawie analizy gromadzonych danych, ustanawiając w cyklu życia oprogramowania proces systematycznej oceny wyrobu programowego przez jego użytkowników, prowadzony na podstawie zbioru opracowanych miar jakości oraz podejmując działania w kolejnej iteracji na podstawie otrzymanych wyników. Wzorując się z kolei na zintegrowanym zbiorze modeli dojrzałości procesów CMMI[®] [CMMI], autorka opracowała model dojrzałości relacji z użytkownikami UR-CMM.

Obserwuje się rozwój **metodyk zwinnych** (*agile methodologies*) i sukcesy stosujących je firm skandynawskich. Wspólną cechą tych metodyk jest zapewniana komunikacja w zespole i z odbiorcami, którą można rozwinąć do ich współuczestnictwa w tworzeniu wyrobu. Autorka podziela wartości respektowane w metodykach zwinnych [Manifesto, Martin i Martin], stawiając:

- **osoby i interakcje** ponad procesy i narzędzia,
- **działające oprogramowanie** ponad opracowane dokumenty (w tym plany),
- **współpracę z klientem** ponad renegocjowanie kontraktu,
- **bieżące reagowanie na zmiany** ponad postępowanie według planu.

Z metodyk zwinnych przejęto: iteracyjno-przyrostowy model wytwarzania, krótkie iteracje, konieczność dobrej samoorganizacji i współpracy w zespołach, unikanie zbędnych działań administracyjnych na rzecz zapewnienia oczekiwanej jakości wyrobu oraz publiczny dostęp do tworzonych artefaktów. Satisfakcja klienta jest najwyższym priorytetem, bezpośrednia konwersacja najwydajniejszą formą przekazywania informacji, zmiany wymagań uwzględniane w procesie wytwarzania, działające oprogramowanie rzeczywistą miarą postępu prac oraz częste dostawy działającego oprogramowania – to niektóre wprowadzone w prezentowanej strategii zasady, zgodne z wyrażonymi w manifestie współautorów i zwolenników metodyk zwinnych [Manifesto].

Konieczność odejścia od klasycznego modelu cyklu rozwojowego oprogramowania deklaruje się już w firmie IBM [Ambler_a]. Krótkie iteracje i częste testy są charakterystyczne dla metodyk XP (*eXtreme Programming*) [Beck_a] oraz wywodzącej się z niej TDD (*Test Driven Development*) [Astels, Beck_b]. Opracowanie wizji przyszłego systemu programowego zapożyczono z metodyki FDD (*Feature Driven Development*) [Coad & DeLuca], w której tworzy się tę wizję w postaci zbioru modeli obiektowych. Dostrzeżono możliwość tworzenia innowacyjnych rozwiązań po stronie klienta przy stosowaniu metodyk zwinnych [Highsmith], w przeciwieństwie do spotykanego w praktyce odtwarzania wczorajszych rozwiązań w nowym systemie. Przystudiowano wykorzystany w projekcie Memorial kooperacyjny model wytwarzania oparty na wirtualnym środowisku DDW (*Digital Document Workbench*) [Drozdowski, Krawczyk i inni], zapewniający zdalną współpracę wielu użytkowników w celu prawidłowej transformacji istniejących dokumentów historycznych do postaci cyfrowej oraz umożliwienia ich przetwarzania. Tak jak w zwinnym podejściu do tworzenia oprogramowania na podstawie modelu (*Agile Model Driven Development*) [Ambler_b] zakłada się tworzenie modeli w formie czytelnej dla odbiorcy, warunkującej współpracę z użytkownikami.

Propagatorzy standardów jakości oraz zwinnych metod wytwarzania oprogramowania zgodnie deklarują swoje **ukierunkowanie na użytkownika** (*user-centeredness*). W praktyce to pojęcie bywa nadużywane, ma rozmaite znaczenia [Iivari i Iivari] i jest rozumiane jako: określenie celów działania z punktu widzenia przeciętnego (fikcyjnego) użytkownika (*user focus*), skupianie się na wykonywaniu przez abstrakcyjnego pracownika czynności wspomaganych tworzonym systemem (*work-centeredness*), uczestnictwo użytkownika (*user participation*) ograniczone na ogół do małych projektów oraz do osób działających w imieniu użytkowników podczas testowania interfejsu (zwykle są to informatycy lub studenci informatyki), po założeniu adaptacyjnych możliwości projektowanego systemu zapewniającego personalizację (*personalization*) jego określonych cech odpowiednio do modelu konkretnego użytkownika. Podsumowując, ukierunkowanie na użytkownika nie obejmuje obecnie wszystkich etapów cyklu rozwojowego. Stwierdzono jednak poprawę jakości oprogramowania, uzyskaną już dzięki współpracy z użytkownikami w fazie uściślenia wymagań [Kujala].

Z uznaniem spotyka się pomysł *User-centred development process*. Ideę partycypacyjnego projektowania (*participative design*) stanowiska pracy z udziałem użytkownika wprowadził w latach 1971–1973 zasłużony norweski informatyk Kristen Nygaard. Spotyka się opinię, że satysfakcja klienta może być zagwarantowana jedynie wtedy, gdy organizacja producenta oprogramowania zaimplementowała tego

rodzaju proces [Scholtz i Morse] i podaje długoterminowe korzyści ze stosowania takiego procesu: wzrost sprzedaży wyrobów programowych, obniżenie kosztów szkoleń, zapewnienie lojalności klienta, redukcja fluktuacji kadry u klienta z uwagi na wzrost satysfakcji z pracy po wdrożeniu systemu. Stwierdza się jednak brak narzędzi dla deweloperów i słabe wehikuły komunikacyjne. Autorka podjęła próbę opracowania strategii realizacji tego typu procesu.

Podejście *User-centred design* (UCD) stosuje się także do implementacji systemu klasy ERP. W pracy [Vilpola] opisano metodykę U-CEI (*User-Centred ERP Implementation*). Wszystkie podane przez Vilpolę czynniki sukcesu wiążą się z czynnikiem ludzkim. Vilpola podkreśla konieczność włączenia *rzeczywistych użytkowników*, aby spełnić wymagania całej organizacji oraz użytkowników końcowych, gdyż tylko oni zapewnią pożądane sprzężenie zwrotne. Autorka uczyniła to wcześniej.

W literaturze stwierdza się brak narzędzi, notacji i metod dotyczących włączenia problematyki użyteczności (*usability*) do cyklu rozwojowego oprogramowania [Kazman i Bass]. Jedynym lekiem na kłopoty z zapewnieniem użyteczności jest *komunikacja*, która jest ciągle wyzwaniem. [Link et al.] proponuje rozszerzenia standardu UML 2, wprowadzając nowe stereotypy ułatwiające modelowanie interakcji z użytkownikami. Uważa przy tym, że przekraczanie istniejącej przepaści między inżynierią oprogramowania a inżynierią użyteczności wymaga zmian w edukacji informatyków. Autorka uważa, że zawężanie się do cech użyteczności nie wystarczy. Uwzględnia w rozprawie problematykę relacji z użytkownikami i kwestię miękkich kompetencji wymaganych we współpracy informatyków z użytkownikami oraz podaje swoje sugestie zmian w zakresie kształcenia informatyków i kształtowania ich postaw wobec użytkowników.

Za punkt wyjścia do opracowania **kryteriów i miar oceny jakości oprogramowania** przyjęto zbiór atrybutów jakości podany w treści standardu ISO/IEC 9126. Następnie rozwijano go o miary jakości opracowane dla potrzeb oceny kilku rodzajów badanego oprogramowania oraz dodano do zbioru miar podzbiór obejmujący odczucia bezpośrednich i pośrednich użytkowników. Poznano i adaptowano podczas budowy drzewa jakości oprogramowania model EUCS (*End User Computing Satisfaction*) [Doll i Torkzadeh, McHaney, Liang] do określenia i pomiaru satysfakcji użytkownika oprogramowania.

Produkcja oprogramowania nie jest jedynie tworzeniem wyrobu, ale jednocześnie usług dostarczanych z jego użyciem i konserwacją. Brak dostrzeżenia tego faktu stwarza dodatkowe ryzyko niepowodzenia [Tiwana i Keil]. Według autorki, nie wystarcza ocena jakości samego wyrobu programowego, ale konieczna jest ocena poziomu świadczonych usług. Stąd w zbiorze kryteriów i miar jakości wyrobu programowego zamieszczono podzbiór miar z punktu widzenia pośrednich użytkowników.

Wśród wspólnych dla rozmaitych przedsięwzięć programowych źródeł ryzyka wymienia się między innymi: brak spełnienia oczekiwań użytkowników końcowych, nieporozumienia w zakresie wymagań oraz załogę dobraną nieodpowiednio do prowadzonego projektu. Autorka podziela te poglądy i proponuje poszerzenie zbioru źródeł ryzyka o ryzyko słabej współpracy z użytkownikami.

W przeglądowej pracy na temat zarządzania oczekiwaniami użytkowników [Petter] zidentyfikowano na poziomie ogólnym trzy rodzaje taktyk uznanych za pomyślne w tym obszarze:

- włączenie użytkowników (współpraca z rzeczywistymi użytkownikami, słuchanie ich w przeciwieństwie do przekonywania),
- przywództwo gwarantujące prawidłowe opracowanie docelowej wizji informatyzacji,
- zaufanie zdobywane przez uczciwe informowanie użytkowników o faktycznym stanie projektu i wszystkich pojawiających się problemach.

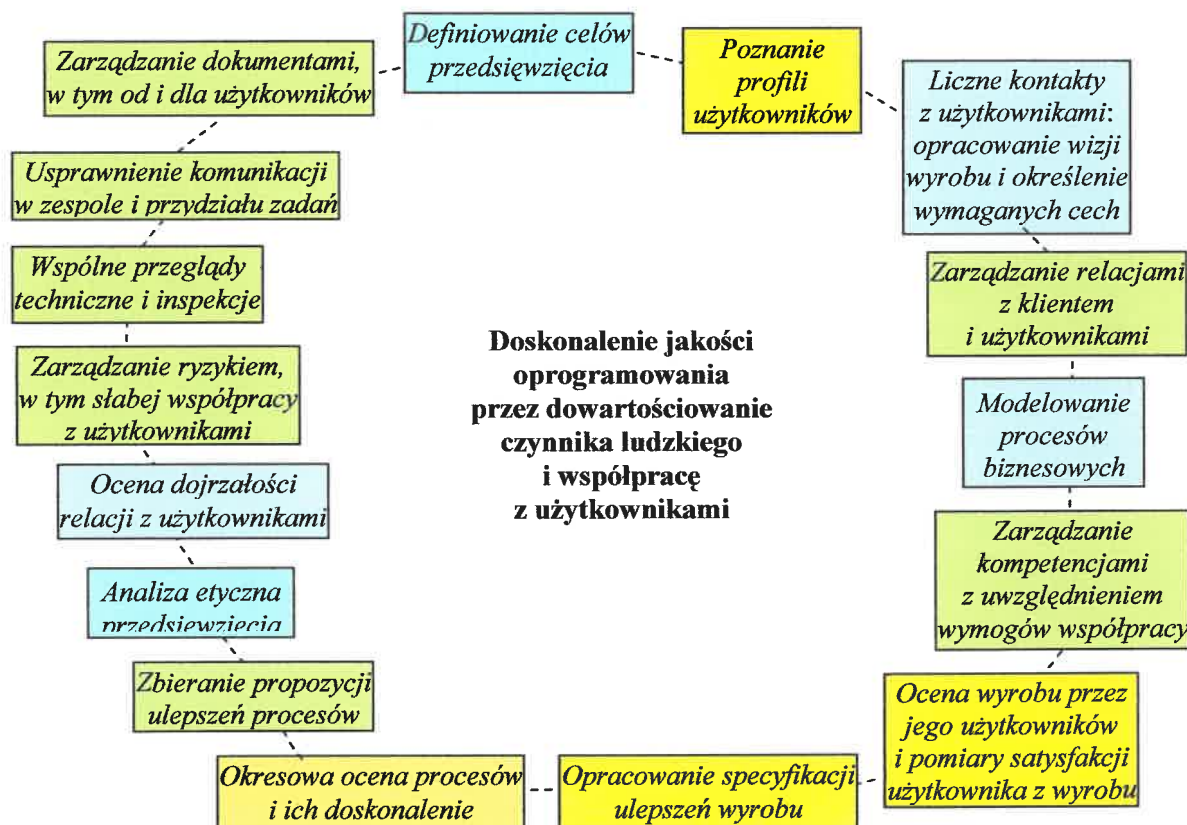
Autorka koncentruje się na pierwszej i ostatniej z wymienionych wyżej taktyk, rozwijając rolę użytkownika jako bezpośredniego uczestnika procesu wytwórczego.

Zalecenie włączenia użytkowników w proces wytwarzania oprogramowania wymaga potwierdzenia, czy i w jakim zakresie to uczestnictwo przynosi korzyści. W pracy z marca 2010 [Subramanyam] przedstawiono wyniki badań, przeprowadzonych wśród 746 respondentów w stu amerykańskich firmach programistycznych, a dotyczących poziomu satysfakcji deweloperów oraz użytkowników oprogramowania w odniesieniu do 117 przedsięwzięć programowych, dotyczących zapewnienia sprawnego łańcucha dostaw w procesach produkcyjnych. Stwierdzono najwyższy poziom satysfakcji autorów oprogramowania w przypadku wysokiego zaangażowania użytkowników w nowych projektach, choć użytkownicy wykazywali wtedy niski poziom satysfakcji, wynikający prawdopodobnie z powstałych nadmiernych ich oczekiwań. Z kolei w projektach polegających na modyfikowaniu istniejącego wyrobu programowego największa satysfakcja po obu stronach ma miejsce, gdy poziom zaangażowania użytkowników (liczony dniami roboczymi) jest średni. Autorka wykazała [10], że ocena każdej wersji oprogramowania przynosi wymierną poprawę jakości i wyższy poziom satysfakcji użytkowników.

4. Opracowana strategia

Doskonalenie jakości oprogramowania przez dowartościowanie czynnika ludzkiego i współpracę z użytkownikami wymaga zagospodarowania wielu obszarów, pokazanych na rys. 1. Uwzględniając pokazane elementy konieczne, zdaniem autorki, do ustawicznego doskonalenia jakości wyrobu programowego i procesu jego wytwarzania, opracowano opisaną w rozprawie oraz pracy [3] strategię wytwarzania nazwaną UID (*User Involved Development*). Autorka określiła wytyczne dla tej strategii:

- Ukierunkowanie na produkt.
- Ewolucyjny rozwój wyrobu, wiele iteracji (niewielkie przyrosty w każdej iteracji).
- Innowacje wprowadzane po stronie klienta.
- Zorientowanie na ludzi (u dostawcy i odbiorcy).
- Uwzględnianie potrzeb rzeczywistych użytkowników.
- Odpowiedniość produktu do oczekiwań biznesu i użytkowników.
- Nadażanie za zmianami.
- Wysoki poziom satysfakcji użytkowników.
- Ciągła współpraca z reprezentantami użytkowników.
- Zapewnienie etyki w zarządzaniu przedsięwzięciem programowym.
- Zarządzanie kompetencjami personelu pod kątem potrzeb współpracy z użytkownikami.
- Ustawiczne ulepszanie wyrobu odpowiednio do sugestii i ocen podanych przez użytkowników.
- Ulepszanie procesu oparte na wynikach oceny wyrobu.

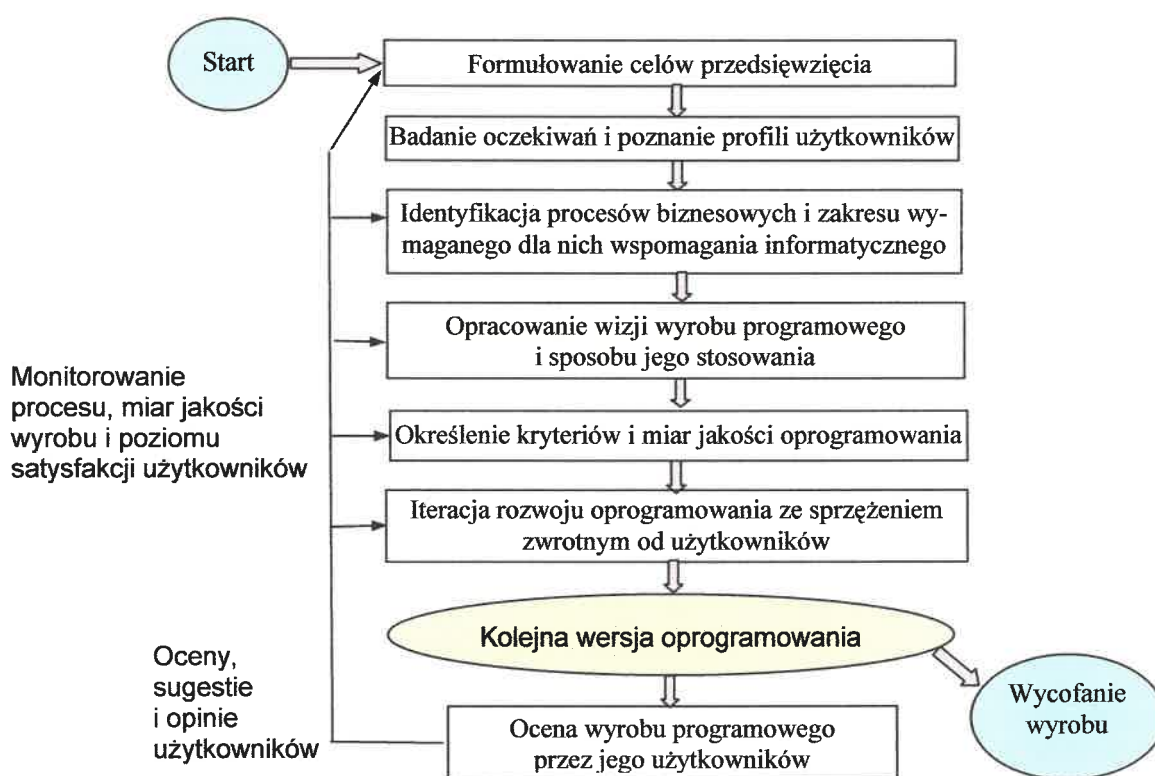


Rys. 1. Elementy doskonalenia jakości wyrobu programowego i procesu jego wytwarzania dobrane pod kątem usatysfakcjonowania użytkowników

W opracowanej strategii jest uwzględniany rzeczywisty użytkownik, a nie abstrakcyjny lub hipotetyczny. Z założenia odgrywa nie tylko rolę dostawcy informacji (w procesie analizy wymagań, wstępnego projektowania i projektowania przypadków testowych) i/lub konsultanta proponowanych rozwiązań (modeli, prototypów, procedur obsługi klienta, wzorów dokumentów). Jest również konstruktywnym krytykiem kolejnych wersji wyrobu programowego podczas jego oceny i źródłem sugestii jego ulepszeń. Bywa także współprojektantem tworzonych rozwiązań.

Opracowana strategia bazuje na sprzężeniu zwrotnym od użytkowników oprogramowania, zgodnie z założeniem ich upodmiotowienia. Strategia dotyczy budowy oprogramowania dla organizacji publicznej lub systemu eksperckiego. Zakładana innowacyjność oznacza, że w niedalekiej przyszłości oprogramowanie dla organizacji publicznej będzie także zaliczane do klasy systemów eksperckich. Cykl życia oprogramowania w tej strategii, z zaznaczoną zewnętrzną pętlą sprzężenia zwrotnego w postaci oceny dostarczonej wersji oprogramowania przez jej użytkowników, przedstawiono na rys. 2.

Wewnętrzne pętle sprzężenia zwrotnego są związane z różnymi formami zaangażowania użytkowników. Proponuje się włączenie reprezentanta użytkowników do każdego zespołu, w tym uczestnictwo w planowaniu i nadzorowaniu testów. Organizowane po stronie klienta sesje burzy mózgów oraz okresowe spotkania w mieszanych zespołach w proporcji pół na pół służą znalezieniu optymalnych rozwiązań. Obecność użytkowników eliminuje ukrywanie przed klientem rzeczywistego stanu prac i projektów. Przedstawiciele użytkowników są zaangażowani w planowane przeglądy techniczne, wprowadzając istotne dla nich tematy do planu przeglądu. Uczestniczą w tworzeniu list kontrolnych i wnoszą swoje propozycje, aby te działania także uwzględniały punkt widzenia użytkowników. Reprezentanci użytkowników mają wgląd w przyjęte ustalenia i wnioski. Reprezentanci mają uprawnienia do wiążących ustaleń, choć nie są przedstawicielami jedynie kadry kierowniczej. Jako fachowcy mogą określić uprawnienia wykonywania swoich ról zawodowych za pomocą tworzonego systemu.



Rys. 2. Cykl życia oprogramowania w proponowanej strategii

Współpraca z reprezentantami użytkowników wymaga zapisów w umowie, które określą między innymi: liczbę reprezentantów obecnych u dostawcy, częstość spotkań mieszanych zespołów, intencję zwołania sesji typu burza mózgów, przyznane uprawnienia merytoryczne i finansowe. Dostawca zastrzega z kolei wymóg prezentowania uzgodnionego stanowiska przez reprezentantów użytkowników.

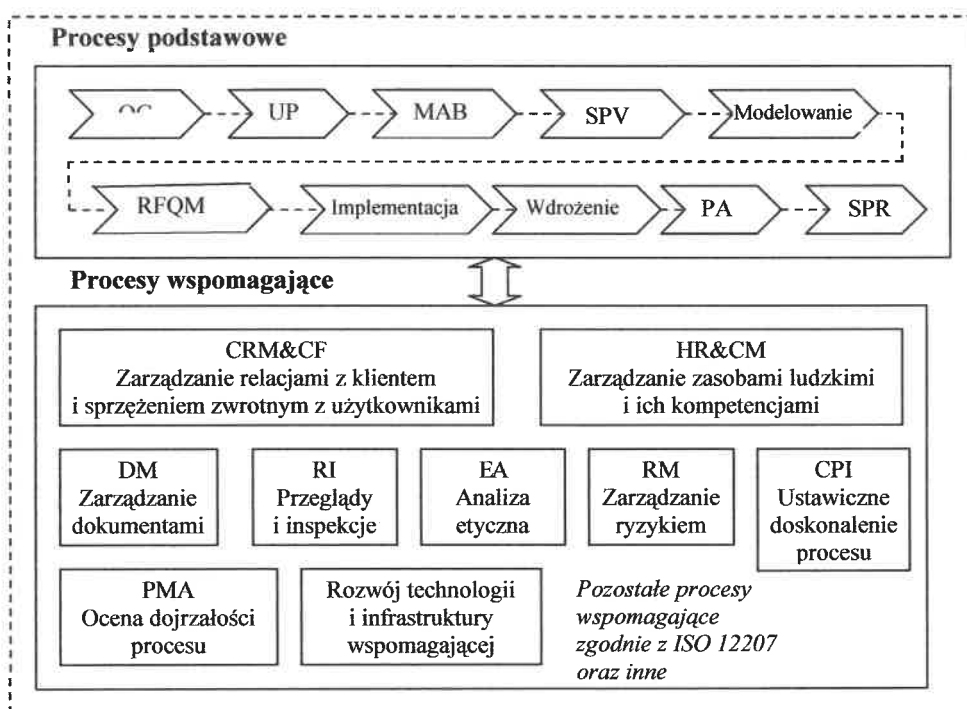
5. Narzędzia proponowane do realizacji przedstawionej strategii

5.1. Zbiór wymaganych procesów

Realizacja wymienionych wyżej wytycznych oraz form włączenia użytkowników w celu uzyskania i wykorzystania sprzężenia zwrotnego z użytkownikami wymaga ustanowienia procesów podstawowych i wspomagających, przedstawionych na rys. 3 (vide rozdz. 4.4.2 rozprawy oraz [9]).

W podejściu UID rozbudowano zbiór **procesów podstawowych** (analiza i modelowanie, implementacja, wdrożenie) o pokazane na rys. 3 dodatkowe procesy i/lub zmieniono ich znaczenie:

- QG (*Quality of Goals*) – analiza jakości celów przedsięwzięcia,
- UP (*User Profiles*) – poznanie profili użytkowników,
- MAB (*Modeling & Analysis of Business processes*) – modelowanie i analiza procesów biznesowych w organizacji klienta,
- SPV (*Software Product Vision*) – opracowanie wizji wyrobu programowego oraz sposobu jego stosowania,
- Modelowanie powiązane z określeniem zbioru wymaganych cech wyrobu; reprezentanci użytkowników współuczestniczą w tym procesie,
- RFQM (*Required Features and their Quality Measures*) – określenie podzbioru wymaganych cech oprogramowania (do realizacji w danej iteracji) oraz kryteriów i miar ich jakości,
- Implementacja, podczas której planowanie i projektowanie przypadków testowych odbywa się z udziałem użytkowników, a testowanie przebiega pod ich nadzorem,
- Wdrożenie – dobranie zakresu szkoleń do profili użytkowników,
- PA (*Product Assessment*) – ocena każdej wdrożonej wersji wyrobu przez użytkowników i analiza potencjalnych zagrożeń jakości użytkowej wyrobu,
- SPR (*Specification of Product Refinement*) – określenie zmian i ulepszeń wyrobu ukierunkowane na zbiór jego cech z uwzględnieniem wyników oceny.



Rys. 3. Procesy podstawowe i wspomagające w proponowanej strategii UID

Nazwy wymaganych procesów wspomagających rozwinięto na rys. 3. Prowadzenie podstawowych procesów zawężono tu do jednego cyklu rozwojowego. Zaznaczono ich wymaganą obecność, a nie rangę ani częstość występowania. Linia przerywana określająca przejścia na mapie procesów symbolizuje przepływ wypracowanej wartości dodanej, ale nie determinuje kolejności wykonywania – występują nawroty po ocenie wyników etapu.

5.2. Problematyka społeczno-etyczna przedsięwzięcia programowego

W artykule [1] oraz w rozdziale 4.5 rozprawy opisano postulowaną *analizę jakości celów* prowadzonego przedsięwzięcia. Jej celem jest zbadanie, czy korzystne rezultaty lokalne w postaci zwiększenia efektywności i wydajności pracy w danej organizacji są pozytywne w szerszej skali. Interes danej organizacji nie zawsze pokrywa się z interesem społecznym – system może sprzyjać uprzedmiotowieniu pracowników, postępującej makdonaldyzacji społeczeństwa, zrywaniu więzi między ludźmi, nieetycznym zachowaniom, homogenizacji rynkowej i kulturowej w skali globalnej¹. Poprawa wyników danej

¹ N. Klein, *No logo*, Izabelin, Świat Literacki 2004, s. 136.

organizacji może wiązać się z przerzuceniem kosztów na społeczeństwo. Koszty społeczne przybierają postać: straty czasu patentów, mitręgi interesantów, perturbacji w załatwianiu spraw urzędowych, czasochłonnej obsługi systemu, narażenia zdrowia pracowników, wzrostu bezrobocia, braku dostępu do określonych usług i związanego z tym społecznego wykluczenia, frustracji obywateli wynikającej z poczucia, że nic nie działa, jak należy, itp. Autorka proponuje poszerzyć cykl rozwojowy oprogramowania o prowadzoną na początku cyklu analizę celów systemu, potencjalnych korzyści oraz dających się przewidzieć perturbacji i zagrożeń społecznych, a na końcu cyklu dodać analizę jakości wyrobu w odbiorze społecznym. W rozprawie podano zbiór pytań (*vide s. 76–77*), na które powinno się znaleźć odpowiedź podczas sugerowanej analizy. Analiza celów przedsięwzięcia ze społecznego punktu widzenia i analiza potrzeb biznesu pozwolą prawidłowo określić wymagania wobec systemu, w szczególności finansowanego ze środków publicznych. W dobie globalizacji należy dodatkowo brać pod uwagę istniejące różnice kulturowe, które mogą dotyczyć samej istoty planowanego systemu. Na przykład, Mohammed Begg podczas konferencji ETHICOMP 2005 wyraził opinię, iż muzułmanie nie będą korzystać ze zdalnego nauczania przez Internet, gdyż w ich kulturze do nauczania konieczny jest autorytet nauczyciela, potwierdzany w bezpośrednich relacjach między mistrzem a uczniem.

Rozwój inżynierii oprogramowania odbywa się na użytek firm programistycznych, jednak ich działania nie zawsze są zgodne z etyką – nie jest etyczne czerpanie profitów ze słabego systemu. Pracownicy informatyzowanej organizacji, a pośrednio obywatele, są zmuszani do zachowań proponowanych przez projektanta – człowiek ma dostosować się do systemu, a nie odwrotnie. Zwykle nie zapewnia się równorzędnych sposobów postępowania, wdrażając *de facto* monopolistyczne praktyki, a nawet lekceważąc użytkownika, na przykład zmuszając go do korygowania danych w niedogodny sposób, jak ma to miejsce w przypadku deklaracji ubezpieczeń społecznych.

Kwestiom etycznym w budowie i stosowaniu systemów informatycznych autorka poświęciła kilka publikacji, w tym zamieszczonych w materiałach konferencji serii ETHICOMP, które odbywają się w odstępie półtora roku. I tak w artykule na ETHICOMP 2001 przedstawiono dostrzegane mocne i słabe strony stosowania produktów IT, przestrzegając przed kolejnymi krokami w kierunku makdonaldyzacji społeczeństwa oraz proponując kierowanie się kryteriami ogólnoludzkimi i etycznymi. Podczas ETHICOMP 2002 cytata z artykułu autorki „*Nowoczesne społeczeństwo potrzebuje kreatywnych ludzi o otwartych umysłach, a jednocześnie w skali globalnej przekształca ludzi w pasywne indywidua, zdolne jedynie do klikania myszą lub naciśnięcia przycisku telewizyjnego pilota (Modern society needs creative and open-minded people but the same society on a global scale is transforming human beings into passive individuals who are able just to click a mouse or to push a button of the TV remote control)*” uznano za motto konferencji wyświetlane na telebimie. Referat w 2005 poświęcono wymaganej reorientacji autorów oprogramowania na rzecz spełnienia oczekiwań szerokiego spektrum rzeczywistych użytkowników. W roku 2008 włączenie użytkowników związane z eliminowaniem dostrzeganych zagrożeń wnoszonych przez informatyzację. Praca przyjęta na ETHICOMP 2010 dotyczy uwzględnienia zaangażowania użytkowników w procesie zarządzania ryzykiem [11].

5.3. Układ opisu wymaganego procesu

Prowadzenie wymienionych w autoreferacie na rys. 3 procesów zmienia przebieg pozostałych. Czynności składające się na procesy podstawowe i wspomagające przeplatają się i nakładają. Każdy przyjęty proces wymaga zdefiniowania, wdrożenia, nadzorowania, oceny i doskonalenia. W rozdziale piątym rozprawy opisano 16 wymaganych dodatkowo lub zmienionych w opisywanej strategii procesów/ procedur postępowania w następującym układzie:

- Cele wprowadzenia
- Przedmiot działań
- Zalecana/proponowana metoda i/lub narzędzie
- Źródło zalecanej metody
- Stosowane pojęcia
- Dane wejściowe i inne wymagania
- Wymagane kompetencje wykonawców
- Zakładane rezultaty
- Schemat postępowania (model graficzny)
- Kryteria i miary oceny
- Uwagi.

Dla każdego procesu trzeba określić i zapewnić niezbędne zasoby: ludzkie, czasowe, finansowe, informacyjne, narzędziowe (sprzęt, oprogramowanie, metody) oraz materiałowe (nośniki, papier).

5.4. Infrastruktura wspomagająca realizację wymaganych procesów. Zbudowane prototypy

Realizacja procesów podstawowych i wspomagających wymaga zapewnienia infrastruktury. Współczesne środowiska wytwarzania oprogramowania oferują narzędzia wspomagające głównie jego konstruowanie (modelowanie, projektowanie, kodowanie itd.) oraz częściowo zarządzanie przedsięwzięciem. Proponowana **rozbudowa środowiska** wytwarzania ma na celu zapewnienie **infrastruktury wspomagającej współdziałanie użytkowników** w procesie. Opracowanie i wdrożenie proponowanych narzędzi urzeczywistnia ideę upodmiotowienia użytkowników oprogramowania, umożliwiając realizację procesów wymaganych w prezentowanej strategii. Proponowana rozbudowa infrastruktury obejmuje:

- zarządzanie relacjami z klientem i współpracą z użytkownikami,
- zarządzanie kompetencjami wykonawców i ich dobór do ról z uwzględnieniem potrzeb współpracy z użytkownikami,
- zarządzanie dokumentami i ich obiegiem, w szczególności na etapie testowania,
- zarządzanie inspekcjami projektów z możliwością wpływu użytkowników na zawartość list kontrolnych,
- zarządzanie ryzykiem w prowadzonym przedsięwzięciu z uwzględnieniem ryzyka słabej bądź porzucanej (deklarowanej jedynie) współpracy z użytkownikami,
- włączenie pracowników do zgłaszania propozycji ulepszeń procesu.

Przy opisie poszczególnych procesów w rozdziale piątym rozprawy podano wytyczne do budowy wymienionych podsystemów, a w szóstym przedstawiono ich prototypy. Poniżej zamieszczono zwięzłą charakterystykę każdego z opracowanych, wyżej wymienionych prototypowych rozwiązań.

Zbudowano prototyp systemu CRM (*Customer Relationship Management*) zarządzania relacjami z klientem i współpracą z użytkownikami, przeznaczony dla firmy programistycznej z Bydgoszczy (*vide* rozdział 6.4 rozprawy oraz [2]). Zadaniem systemu jest utrzymywanie i analiza kanałów wymiany informacji między firmą a reprezentantami jej klientów i użytkowników zgodnie ze strategią UID, gromadzenie zapisów dotyczących kontaktów z przedstawicielami klienta – od zgłoszenia sprawy do dostarczenia wyrobu, rejestrowanie zdarzeń rozmaitych kategorii, które określają tok realizacji zlecenia, ocenę i analizę satysfakcji klientów z wyrobów oraz optymalizowanie procesów pracy w obszarze CRM. W celu realizacji tych zadań, obok danych gromadzonych rutynowo w każdym systemie klasy CRM (jak Klient, Grupa Klientów, Oferta, Produkt, Pracownik, Zespół), wprowadzono jednostki danych: *Przedstawiciel* (reprezentant użytkowników), *Ankieta*, *Sprawa* (wniesiona przez klienta, zwykle zlecenie), *Zdarzenie* (zmienia stan sprawy), *Realizacja* (etap sprawy), *Dokument*, *Notatka* (z kontaktu). Prototyp zapewnia profilowanie klientów i ich reprezentantów, utrzymywanie historii wszelkich kontaktów, kojarzenie dokumentów ze zdarzeniami oraz śledzenie podziału pracy w firmie pod kątem bezpośrednich relacji z reprezentantami klienta, w tym z użytkownikami.

Przyjęto, że w gospodarce opartej na wiedzy firmy konkurują ze sobą na dwóch rynkach: produktów i talentów potrzebnych do ich wytworzenia. Kompetencje załogi są elementem krytycznym metodyk zwinnych, mając zapewnić nadążanie za zmianami otoczenia, które obejmują: rosnące oczekiwania użytkowników, postęp technologiczny i zmieniające się warunki biznesowe. Zaproponowano systematykę kompetencji oczekiwanych u pracowników firmy programistycznej ukierunkowanej na współpracę z użytkownikami [6]. Określono sposób definiowania profili kompetencji dla zidentyfikowanych ról zawodowych i zbudowano profile kompetencji dla kilku wybranych ról. Do identyfikacji wymaganych kompetencji przeprowadzono badania ankietowe. Po przeanalizowaniu metod badania rozmaitego typu kompetencji pracownika wskazano metodę 360° jako przydatną i stosunkowo tanią. Zbudowano dwa prototypy systemu zarządzania kompetencjami z uwzględnieniem potrzeb współpracy z użytkownikami (*vide* rozdział 6.5 rozprawy). Opracowano i przetestowano algorytm doboru wykonawcy do określonej roli. Zaprezentowano wnioski dotyczące stosowania tego algorytmu.

Zbudowano prototypowy system zarządzania dokumentami i ich obiegiem na etapie testowania (*vide* rozdział 6.6 rozprawy), przeznaczony dla firmy ComArch. Zdefiniowano typy używanych dokumentów, wytypowano role osób zaangażowanych w tworzenie ich zawartości i obieg, określono atrybuty z podziałem na część stałą i zmienną dokumentu oraz relacje między dokumentami, a także trasę dokumentu wytyczoną przez jego stany (punkty trasy są opisane rolami uczestników). Prototyp zapewnia łatwość zmiany struktury dokumentu (dzięki jej podziałowi na część stałą i zmienną) oraz możliwość rozbudowy zbioru ról o określonych uprawnieniach, do którego może dołączyć reprezentant użytkowników, na przykład zaangażowany do budowy planu testów lub do oceny raportów przeprowadzonych testów. Przyjęte założenia kwalifikują system do zarządzania dowolnego typu dokumentami, w tym związanymi ze współpracą z użytkownikami w prowadzonym przedsięwzięciu.

Zbudowany prototypowy system zarządzania inspekcjami projektów (*vide* rozdział 6.7 rozprawy) zapewnia możliwość wpływu użytkowników na zawartość listy kontrolnej oraz ustalenie dokumentów podlegających ocenie inspektora. Proponowaną listę kontrolną inspekcji zobrazowano w układzie dwukolumnowym, ilustrującym podział kryteriów oceny na walory techniczne oraz użytkowe.

Zarządzanie ryzykiem w procesie wytwarzania oprogramowania także staje się czynnikiem doskonalenia tego procesu i wyrobu programowego. W opracowanym systemie wspomagającym zarządzanie ryzykiem do atrybutów ryzyka należą: warunek, kontekst, prawdopodobieństwo i ewentualny termin wystąpienia oraz jego konsekwencje. Dla każdego rodzaju ryzyka utrzymuje się zapisy dotyczące jego stanu, strategii łagodzenia skutków wystąpienia oraz dostępnych akcji. Prototyp systemu opisany w rozdziale 6.9 rozprawy pozwala na poszerzenie zbioru uwzględnianych rodzajów ryzyka, w tym o ryzyko słabej współpracy z użytkownikami. Niektóre przykłady ryzyka w tym obszarze to [5, 11]:

- Budowa oprogramowania dla nieokreślonej kategorii użytkowników.
- Brak wizji informatyzacji danej organizacji.
- Lakonicznie sformułowane wymagania.
- Chaos w kontaktach z użytkownikami.
- Brak testów istotnych dla użytkownika przypadków stosowania wyrobu.
- Udostępnienie użytkownikom dopiero gotowego systemu.
- Brak sprzężenia zwrotnego od użytkowników po wdrożeniu określonej wersji wyrobu.
- Słaba motywacja projektantów programistów do współpracy z użytkownikami.
- Użytkownicy pośredni są zmuszani do wypełniania wielkich ilości dokumentów i rubryk.
- Zaimplementowane procedury mogą naruszać prywatność niektórych osób.
- Niska reprezentatywność delegowanych do współpracy reprezentantów użytkowników.
- Słaba motywacja reprezentantów użytkowników do współpracy.

Do zbioru użytkowników systemu wprowadzono kategorię *gości*, którzy są uprawnieni do określania źródeł ryzyka, mają dostęp do określonych kategorii ryzyka i mogą wymieniać komunikaty z autorami oprogramowania. Prototyp wykorzystano podczas budowy i stosowania oprogramowania *Election* do obsługi wyborów na szczeblu wydziału, a w charakterze gości wystąpiły pracownice dziekanatu.

W celu upodmiotowienia pracowników firmy w zakresie ich wpływu na sposób realizacji procesów zbudowano system zgłaszania propozycji ulepszeń procesu wskazanego przez pracownika (*vide* rozdział 6.10 rozprawy). W celu włączenia pracowników do ustawicznego ulepszania procesów utrzymywana jest baza procesów. Załozde udostępnia się opracowane drzewo ustanowionych procesów, łącznie ze strukturą graficzną i opisem każdego z nich. Propozycja ulepszenia odnoszona jest do całego procesu lub jego wskazanego elementu (obiektu składowego). Prototyp umożliwia wygenerowanie graficznego schematu procesu ze zgłoszonymi zmianami w postaci pliku SVG. Utrzymuje się historię zgłoszonych propozycji ulepszeń, jednak decyzje o ich wdrożeniu podejmuje upoważnione gremium.

5.5. Automatyzowanie niektórych zadań przez włączenie agentów programowych

Zwiększanie liczby prowadzonych procesów oraz wprowadzenie nowych uczestników przedsięwzięcia programowego może hipotetycznie powodować rozrost biurokracji oraz zagrażać sprawnemu prowadzeniu prac. Zaangażowanie reprezentantów użytkowników nie oznacza rezygnacji z automatyzowania niektórych czynności, w szczególności działań natury organizacyjnej – wręcz przeciwnie, automatyzacja ma na celu niedopuszczenie do chaosu i zagubienia danych. W rozprawie przedłożono propozycję automatyzowania niektórych zadań przez wprowadzenie agentów programowych. W szczególności dotyczy to zapewnienia sprawnej komunikacji, w tym informowania uczestników o zgłoszonych defektach i uwagach, wprowadzanych zmianach, organizowanych spotkaniach, nowych zadaniach itp.

W rozdziale 5.16 uzasadniono wprowadzanie agentów programowych. Zgodnie z metodologią Gaia, identyfikacja przydatnych agentów dokonuje się drogą analizy ról. Przedstawiono przykłady agentów. Podano schemat postępowania podczas wprowadzania określonego typu agentów. Zaproponowano modelowanie systemu wieloagentowego na diagramach UML, z przedstawieniem ich asynchronicznych i wielowątkowych działań na diagramie czynności z użyciem pasm prezentujących podział odpowiedzialności między poszczególne agenty. Zaproponowano sposoby wymiany informacji między agentami.

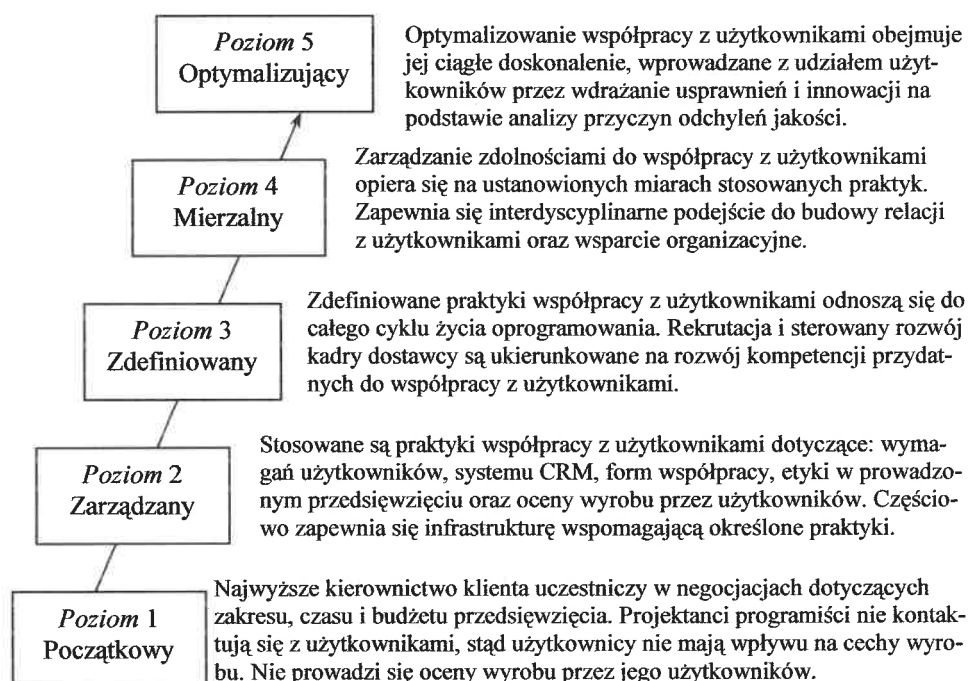
Przedstawiono dwa prototypowe systemy agentów programowych stosowanych do: wprowadzenia zmiany na skutek zgłoszenia defektu przez użytkownika oraz wspomagania przydziału zadań między wykonawców drogą organizowanych przetargów w celu optymalnego wykorzystania czynnika ludzkiego, a jednocześnie zapewnienia wpływu pracownika na rodzaj powierzanych mu zadań. W drugim prototypie role współpracujących agentów obejmują: agenta zadania, agenta zarządcy zadań oraz agenta spotkań. Do każdego nowego zadania dołącza się listę kompetencji wymaganych do jego wykonania.

Wiedza agentów obejmuje: listy zadań do wykonania, opisy zdolności, umiejętności i doświadczeń poszczególnych programistów oraz wykaz zajętych i wolnych terminów każdego z nich. Na tej podstawie dokonuje się wstępnej oceny przydatności pracownika do wykonania określonego zadania.

W rozdziale 6.12 opisano zbudowane prototypowe rozwiązania systemów agentowych. Do przedstawienia i przechowywania kompetencji pracownika służą deskrypcyjne listy zawierające fakty zapisane w postaci wyrażen *Knowledge Elements*. Wspomaganie określonych działań przez poszczególnych agentów programowych daje efekt synergii – poprawiając zarządzanie czasem projektantów programistów i obieg informacji oraz prawidłowy dobór wykonawcy zadania, wpływa na poprawę prac.

5.6. Problem dojrzałości relacji z użytkownikami

W proponowanej strategii wysoką rangę nadano relacjom z użytkownikami. Uznano, że mają one istotne znaczenie procesie wytwarzania oprogramowania i powinny być brane pod uwagę podczas oceny tego procesu. Opracowano wzorowany na rodzinie modeli CMMI pięciostopniowy model dojrzałości relacji z użytkownikami, nazwany UR-CMM (*User-Relationship Capability Maturity Model*), przedstawiony w rozdziale 5.17 rozprawy oraz w pracy [8]. Jego warstwy pokazano na rys. 4.



Rys. 4. Poziomy dojrzałości współpracy z użytkownikami w modelu UR-CMM

Wprowadzono kluczowe obszary dla poszczególnych poziomów. Dla przykładu, na poziomie trzecim są to:

- Profilowanie użytkowników.
- Zarządzanie formami współpracy z użytkownikami.
- Analiza wymaganych kompetencji.
- Zarządzanie rozwojem kompetencji wymaganych do współpracy z użytkownikami.
- Rozwój pracy grupowej.

Zaproponowano praktyki postępowania w każdym zaproponowanym obszarze. Miary poziomu wykonywania poszczególnych praktyk służą do oceny dojrzałości współpracy z użytkownikami.

6. Badania jakości oprogramowania i poziomu satysfakcji jego użytkowników

Przed podjęciem decyzji o wprowadzeniu proponowanej strategii mogą pojawić się wątpliwości:

- Czy użytkownicy są skłonni współpracować z autorami oprogramowania?
- Czy współudział z użytkownikami poprawia jakość tworzonego oprogramowania?
- Czy dotyczy to także systemów eksperckich, przeznaczonych dla specjalistów innych profesji?

- Czy współpraca z użytkownikami wymaga dodatkowej infrastruktury (wsparcia informatycznego)?

Wykonane badania pozwoliły udzielić odpowiedzi twierdzącej na wszystkie wyżej przedstawione pytania. Sześciokrotnie przeprowadzono ocenę jakości eksperckiego systemu programowego obliczeń wytrzymałości konstrukcji budynków wysokich z nadprożami, stosowanego przez inżynierów konstruktorów budownictwa. Ocenę oprogramowania prowadzono z punktu widzenia jego użytkowników. Badano poziom satysfakcji użytkowników z tego oprogramowania. Opracowano i sprawdzono w praktyce schemat postępowania podczas oceny jakości oprogramowania. Obejmuje on:

- Określenie celu badania.
- Poznanie badanego oprogramowania.
- Budowę drzewa jakości oprogramowania – przyjęcie kryteriów i miar oceny jego jakości.
- Określenie grupy ankietowanych osób i budowę profili użytkowników.
- Uzyskanie zgody kierownictwa na przeprowadzenie oceny.
- Projektowanie struktury kwestionariusza ankiety.
- Dobór skali ocen i form akceptowanych odpowiedzi.
- Projektowanie szczegółowe kwestionariusza i próby jego wypełnienia przez dobrane osoby.
- Przeprowadzenie badania w uzgodnionym terminie.
- Zebranie otrzymanych odpowiedzi i obliczenie procentowego udziału odpowiedzi udzielonych dla poszczególnych pozycji kwestionariusza.
- Statystyczne przetworzenie otrzymanych rezultatów.
- Analizę wybranych ocen.
- Porównanie z wynikami poprzedniej oceny.

Zbudowano drzewo jakości badanego oprogramowania (*vide* rozdz. 3.2 rozprawy), a na jego podstawie opracowano kwestionariusz jej oceny (rozdz 5.7.2). Rezultaty kolejnych ocen opisano szczegółowo w rozdziale 6.2 rozprawy, ilustrując wykresami rozrzut podanych ocen (dla przypadków, w których wystąpił). Przedstawiono tryb oceny oraz rezultaty i wnioski z czterokrotnych badań jakości tego oprogramowania w latach 2002–2005. Po każdej ocenie uzyskano wartościowy materiał pod kątem poprawy jakości wyrobu w postaci podanych przez użytkowników ocen poszczególnych miar oraz sugestii zmian wyrobu. Co drugi ankietowany podał sugestie jego poprawy i ulepszeń. **Wykazano, że współpraca z użytkownikami przynosi poprawę jakości oprogramowania** – w przypadku 25 miar jakości badanego wyrobu stwierdzono wzrost średnich wartości w 2005 w stosunku do wyników poprzedniej oceny.

W pracy [10] podano wytyczne do budowy drzewa jakości systemu eksperckiego, przeprowadzania oceny jakości przez użytkowników oraz wykorzystania rezultatów tej oceny. Przedstawiono rezultaty kolejnych ocen oprogramowania używanego przez konstruktorów budownictwa – w 2009 roku aż 35 miar (na 41) uzyskało wartość średnią powyżej 4,0 w stosowanej szkolnej skali ocen od 1 do 5 (32 w 2007, a 30 w 2005 r.), a poziom satysfakcji jego użytkowników osiągnął średnią wartość 4,64 (4,55 w 2007, a 4,08 w 2005 r.).

Przeprowadzono ponadto badania ankietowe oceny jakości następujących wyrobów programowych przez ich użytkowników (*vide* rozdziały 5.7.3 i 6.3 rozprawy):

- modułów *Ruch Chorych* i *Apteka Szpitalna* systemu programowego Eskulap dla szpitali,
- systemu PROMIS w Zakładach Produkcji betonów Komórkowych,
- programów EWID, Siły i Środki, Podział Bojowy oraz Woda2000 używanych w Państwowej Straży Pożarnej,
- systemu Info-Ekspert stosowanego przez agentów ubezpieczeniowych,
- oprogramowania dwóch sklepów internetowych.

Przeprowadzone badania wykazały, że użytkownicy oprogramowania są chętni do współpracy, jeśli widzą sens wspólnych działań. Około jedna czwarta użytkowników podanych wyżej programów określiła konkretne potrzeby zmian w oprogramowaniu.

W pracy [7] przedstawiono, wywodzące się z metodyk zwinnych oraz oparte na prezentowanej w rozprawie strategii, formy współpracy z odbiorcami energii elektrycznej, którzy są pośrednimi użytkownikami oprogramowania stosowanego w energetyce. Zaproponowano ocenę satysfakcji odbiorców, dobierając 6 kryteriów (zdekomponowanych na 35 miar) tej oceny:

- Zawartość dokumentów, w tym umów (5 miar).
- Dokładność zamieszczonych danych (4 miary).
- Format i zrozumiałość stosowanych dokumentów (3 miary).

- Łatwość kontaktów i procedur (10 miar).
- Terminowość usług (7 miar).
- Poziom relacji z klientem (6 miar).

Pierwszych pięć kryteriów nawiązuje do modelu EUCS (*End User Computing Satisfaction*), natomiast ostatnie kryterium zostało dodane przez autorkę. Wszystkie wymienione wyżej kryteria i miary odnoszą się pośrednio do jakości stosowanego oprogramowania, które zapewnia posługiwanie się określonymi dokumentami i warunkuje poziom usług świadczonych przez Biura Obsługi Klienta. Zaproponowano formę ankiety bezpośredniej, w której odbiorcy określają swój profil (autorka zaproponowała strukturę *Profilu indywidualnego odbiorcy*) oraz przypisują wartości zawartym w kwestionariuszu miarom, składającym się na ocenę satysfakcji odbiorcy.

7. Badanie świadomości zagrożeń w wyniku postępującej informatyzacji

Kompleksowa informatyzacja, obok korzyści, niesie pewne zagrożenia, na przykład: osamotnienie jednostki, społeczny autyzm, dehumanizację pracy, bierność obserwatorów ekranu, bezmyślność klientów informacyjnego supermarketu, zanik wzorców zachowań w wyniku zastępowania bezpośrednich relacji międzyludzkich środkami elektronicznymi, brak poszanowania prywatności, strach przed wykorzystaniem osiągnięć technologii przez nieprawie jednostki itd. Informatyzacja nie jest jedynym źródłem tych zagrożeń, nie pojawiłyby się one jednak bez środków informatyki. Pojawia się kwestia, czy informatycy, zwłaszcza przyszli projektanci programiści, mają świadomość zagrożeń, w skali indywidualnej oraz globalnej, związanych z postępującą informatyzacją? Mając na uwadze szeroko pojmowane aspekty etyczne prowadzonych przedsięwzięć programowych, uznano za celowe podjęcie badań nad poziomem świadomości zagrożeń wśród przyszłych projektantów programistów.

W maju 2008 przeprowadzono wśród studentów Wydziału Elektrycznego PP ankietę, której celem było poznanie świadomości zagrożeń w wyniku postępującej informatyzacji. Dostrzegane zagrożenia respondenci mieli odnieść do skali indywidualnej, krajowej oraz globalnej. Poproszono respondentów o zaznaczenie, czy doświadczyli podanych przez siebie zagrożeń osobiście, czy też opierają się na informacji z drugiej ręki. Ankietę i jej rezultaty przedstawiono w pracy [9]. Respondenci wymienili 269 zagrożeń, średnio po 3,64, przy czym wiele wymienianych zagrożeń pokrywało się. Po przeliczeniu wyników opracowano ranking dostrzeganych zagrożeń. W skali indywidualnej na pierwszy plan wysuwają się zagrożenia zdrowia, głównie chorób oczu i kręgosłupa. W skali krajowej na czele listy są zagrożenia prywatności, które dominują także w skali globalnej, obok nieautoryzowanego dostępu do danych osobowych i informacji prywatnych.

Uświadomienie istnienia zagrożeń w wyniku postępującej informatyzacji, zarówno po stronie autorów, jak i użytkowników oprogramowania, pozwala zmniejszyć prawdopodobieństwo ich urzeczywistnienia przez zachowanie wymaganej ostrożności i procedur bezpieczeństwa. Części dostrzeganych zagrożeń można uniknąć dzięki włączeniu użytkowników w proces wytwarzania oprogramowania.

8. Podsumowanie

W prezentowanych pracach całościowo ujęto problematykę doskonalenia jakości wyrobu programowego – opracowano strategię opartą na współpracy z użytkownikami. Współpraca wbudowana jest w cały proces i obejmuje wszystkie etapy wytwarzania oprogramowania. Reprezentanci użytkowników nie są figurantami w tym procesie – opracowane podejście UID (*User-Involved Development*) zapewnia sprzężenie zwrotne z użytkownikami i jego faktyczne wykorzystanie w rozwoju i ulepszaniu wyrobu programowego.

W celu urzeczywistnienia idei upodmiotowienia użytkowników oraz realizacji proponowanej strategii poszerzono zbiór procesów podstawowych i wspomagających tworzenia oprogramowania. Opisano, według przyjętego schematu, procedury zalecane do wykonywania procesów składowych.

Szczególne znaczenie przypisano ocenie każdej wdrożonej wersji oprogramowania – powtarzane cyklicznie oceny i opracowywane na ich podstawie propozycje zmian wyrobu są wyróżnikami proponowanej strategii. Zebrano kryteria i zaproponowano miary oceny jakości oprogramowania przez jego użytkowników, a w przypadku wyrobów programowych dla organizacji publicznych także użytkowników pośrednich. Wielokrotnie wykonano badania jakości oprogramowania i satysfakcji użytkowników. Wykazano, że sprzężenie zwrotne od użytkowników prowadzi do poprawy jakości oprogramowania i podnoszenia poziomu satysfakcji jego użytkowników.

Zapewniono infrastrukturę ułatwiającą realizację proponowanych procesów wspomagających. Zbudowano prototypy systemów: zarządzania relacjami z klientem i jego reprezentantami (CRM), zarzą-

dzania dokumentami (DM), zarządzania ryzykiem (RM), wspomaganie inspekcji (RI), zarządzania kompetencjami (CM) oraz zgłaszania ulepszeń przez pracowników w ramach ustawicznego doskonalenia procesu (CPI). Zaproponowano częściową automatyzację prac z użyciem agentów programowych. Zbudowano prototypowe systemy wieloagentowe.

Uwypuklono aspekty etyczne w wytwarzaniu oprogramowania. Zebrano przykłady nieetycznych praktyk postępowania. Przeprowadzono według własnego pomysłu badanie świadomości zagrożeń wynikających z postępującej informatyzacji.

Uznając, że relacje z użytkownikami mają istotne znaczenie w procesie wytwarzania oprogramowania, zaproponowano badanie poziomu tych relacji i ich stopniowy rozwój. Przedłożono wzorowaną na zbiorze modeli CMMI propozycję modelu dojrzałości współpracy firmy programistycznej z użytkownikami, w postaci modelu UR-CMM, do oceny i planowania poprawy współpracy z użytkownikami.

Wykaz publikacji autorki powoływanych w autoreferacie

- [1] Begier B., Quality of Goals – A Key to the Human-Oriented Technology, *Australian Journal of Information Systems*, vol. 9, Number 2, May 2002, s. 148–154.
- [2] Begier B., Zarządzanie relacjami z klientem w firmie informatycznej, Rozdział XVI w: *Inżynieria oprogramowania. Nowe wyzwania* (red. J. Górski, A. Wardziński), ISBN 83-204-3051-8, Wydawnictwa Naukowo-Techniczne, Warszawa 2004, s. 213–226.
- [3] Begier B., The UID Approach – the Balance between Hard and Soft Methodologies. W: *Software Engineering: Evolution and Emerging Technologies*, ISBN 1-58603-559-2, IOS Press, Amsterdam 2005, s. 15–26.
- [4] Begier B., Involving Users to Improve the Level of Their Satisfaction from a Software Product Designed for Public Organization. W: *Technologies for Business Information Systems*, Springer Verlag, Dordrecht (Netherlands) 2007, ISBN 978-1-4020-5633-8, s. 365–377.
- [5] Begier B., *Doskonalenie jakości oprogramowania przez włączenie użytkowników w proces jego wytwarzania*, Wyd. Politechniki Pozn., nr 417 w serii Rozprawy, Poznań 2007 (wydrukowana w styczniu 2008), ISBN 978-83-7143-363-4, ISSN 0551-6528, 211 stron.
- [6] Begier B., Zarządzanie kompetencjami elementem jakości procesu wytwarzania oprogramowania. W: *Inżynieria oprogramowania – metody wytwarzania i wybrane zastosowania* (red. B. Hnatkowska, Z. Huzar), Wydawnictwo Naukowe PWN, ISBN 978-83-01-15573-5, Warszawa 2008, s. 89–102.
- [7] Begier B., Współpraca z odbiorcami podczas wprowadzania nowych rozwiązań w energetyce, *Rynek Energii*, Nr II (IV), marzec 2009, s. 152–157.
- [8] Begier B., Ocena dojrzałości relacji z użytkownikami w procesie wytwarzania oprogramowania. W: *Od modelu do wdrożenia. Kierunki badań i zastosowań inżynierii oprogramowania* (red. W. Dąbrowski, A. Stasiak), Wydawnictwa Komunikacji i Łączności, Warszawa 2009, s. 35–45.
- [9] Begier B., Users' Involvement May Help Respect Social and Ethical Values and Improve Software Quality, *Information Systems Frontiers*, Springer US, 2009 (w druku, po korekcie autorskiej w lipcu 2009 r.).
- [10] Begier B., Evolutionally Improved Quality of Intelligent Systems Following Their Users' Point of View. W: *Advances in Intelligent Information and Database Systems*, (Eds. N. T. Nguyen et al), Studies in Computational Intelligence (SCI) 283, Springer-Verlag, Berlin Heidelberg, 2010, s. 191–203.
- [11] Begier B., Enhancing risk management in a software process to cover risks referred to software users. W: 11th International Conference ETHICOMP 2010, April 14–16 2010, Tarragona, Spain (w druku).

Wykaz prac innych autorów powoływanych autoreferacie

- [Ambler_a] Ambler S., Active Stakeholder Participation, <http://www.agilemodeling.com/essays/> od 2003.
- [Ambler_b] Ambler S. W., *Agile Model Driven Development (AMDD)*, 2006, <http://www.agilemodeling.com>.
- [Arbaoui, Wastell i inni, 1999] Arbaoui S., Lonchamp J., Montangero C., The Human Dimension of the Software Process, w: *Software Process: Principles, Methodology, Technology*, eds. J.-C. Dermiane, B. A. Kaba, D. G. Wastell, London, Springer 1999, s. 165–200.
- [Astels] Astels D., *Test-Driven Development: A Practical Guide*, Addison-Wesley 2003.
- [Beck_a, 2000] Beck K., *Extreme Programming*, Boston, USA, Addison-Wesley 2000.
- [Beck_b, 2003] Beck K., *Test-Driven Development: By Example*, Addison-Wesley 2003.
- [British Computer Society] British Computer Society, *The challenges of complex IT products*, The report of a working group from The Royal Academy of Engineering and the British Computer Society, April 2004, <http://www.bcs.org/BCS/News/positionsandresponses/positions>.
- [CMMI] CMMI[®] Models and Modules, The Software Engineering Institute, Carnegie Mellon University, 2005, <http://www.sei.cmu.edu/cmmi/models/models.html>.
- [Coad i DeLuca] Coad P., Lefebvre E., DeLuca J., *Java Modeling in Color with UML: Enterprise Components and Process*, Prentice-Hall 1999,
także: FDD Process, <http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf>.

- [Doll i Torkzadeh] Doll W. J., Torkzadeh G.: The measurement of end-user computing satisfaction, *MIS Quarterly* 12 (2), 1988, s. 259–274.
- [Drozdowski, Krawczyk i inni] Drozdowski K., Jarzemski J., Krawczyk H., Melzer M., Smółka M., Wiszniewski B., Wirtualne środowisko wspomaganie realizacji złożonych przedsięwzięć informatycznych, w: *Inżynieria oprogramowania. Nowe wyzwania*, red. J. Górski, A. Wardziński, Warszawa, WNT 2004, s. 169–182.
- [Friedman] Friedman B., *Human Values and the Design of Computer Technology*, Cambridge, Cambridge University Press 1997.
- [Gotterbarn] Gotterbarn D., Informatics and professional responsibility, *Science and Engineering Ethics*, vol. 7, Number 2, June 2001, s. 221–230.
- [Highsmith], Highsmith J., *Agile Project Management*, Boston, Addison-Wesley 2004.
- [Hopkins_a] Hopkins M. S., The 4 Ways IT is Driving Innovation. An Interview with Eric Brynjolfsson, *MIT Sloan Management Review*, February 2010 (Reprint nr 51330).
- [Hopkins_b] Hopkins M. S., Value-Creation, Experiments, And Why It Does Matter. An interview with Michael Schrage, *MIT Sloan Management Review*, February 2010 (Reprint nr 51331).
- [Iivari J. i Iivari N.] Juhani Iivari, Netta Iivari, Varieties of User-Centeredness, *Proceedings of the 39th Hawaii Conference on System Sciences*, IEEE 2006 (otrzymany reprint nie zawiera numerów stron).
- [ISO/IEC 9126] International Standard ISO/IEC 9126-1:2001 *Software engineering – Product quality, Part 1: Quality model*, Geneva, ISO Copyright Office 2001.
- [ISO/IEC 14598-1] ISO/IEC 14598-1:1999(E) *Information technology – Software product evaluation. Part 1: General overview*, Geneva, ISO/IEC Copyright Office 1999.
- [ISO/IEC 15504 p. 3.45] ISO/IEC TR 15504 *Information Technology – Software Process Assessment. Parts 1–9*, Geneva, ISO/IEC Copyright Office 1998.
- [ISO/IEC 25000] *Software engineering – Software product Quality Requirements and Evaluation (SQuARE) – Guide to SQuARE*, ISO/IEC Copyright Office, 2005.
- [Kazman I Bass] Kazman R., Bass L. (Guests Editors), Editorial: Special Issue on Bridging the Process and Practice Gaps between Software Engineering and Human–Computer Interaction, *Software Process: Improvement and Practice*, vol. 8 (2003), s. 63–65.
- [Kujala] Kujala S., Effective user involvement in product development by improving the analysis of user needs, *Behaviour & Information Technology*, vol. 27, 2008, No. 6, s. 457–473.
- [Liang] Liang T.-P., Lai H.-J., Ku Y.-C.: Personalized Content Recommendation and User Satisfaction: Theoretical Synthesis and Empirical Findings, *Journal of Management Information Systems*, vol. 23 (2007), No. 3, s. 45–70.
- [Link i inni] Link S., Hoyer Ph., Kopp T., Abeck S., A Model-Driven Development Approach Focusing Human Interaction, *Software Process: Improvement and Practice*, vol. 14 (2009), s. 90–139.
- [Manifesto] *Manifesto for Agile Software Development*, autorzy: K. Beck, A. Cockburn, M. Fowler, J. Highsmith, R. C. Martin i inni, <http://agilemanifesto.org/>, Agile Alliance February 2001.
- [Martin i Martin] Martin R. C., Martin M.: *Agile. Programowanie zwinne. Zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C#*, Helion, Gliwice 2008 (w oryginale: *Agile Principles, Patterns, and Practices in C#*, Indianapolis: Pearson Education and Prentice Hall, 2007).
- [McHaney] McHaney R., Hightower R., Pearson J.: A validation of the end-user computing satisfaction instrument in Taiwan, *Information & Management*, vol. 39 (2002), s. 503–511.
- Ojala P., Experiences of a Value Assessment for Products, *Software Process: Improvement and Practice*, vol. 14 (2009), s. 31–37.
- [Petter, 2008] Petter S., Managing user expectations on software project: Lessons from the trenches, *International Journal of Project Management*, vol. 26, 2008, s. 700–712.
- [PN-ISO 9000] PN-EN ISO 9000:2001 *Systemy zarządzania jakością. Podstawy i terminologia* (tłum. EN ISO 9000:2000), Warszawa, Polski Komitet Normalizacyjny 2001.
- [PRINCE2] PRINCE2™, <http://www.prince2.com/>, <http://pl.wikipedia.org/wiki/PRINCE2>.
- [Rogerson i Gotterbarn] Rogerson S., Gotterbarn D., The Ethics of Software Project Management, in: *Ethics and information technology*, ed. G. Collste, Delhi, New Academic Publishers 1998, s. 137–154.
- [RUP] RUP. IBM Rational Unified Process®, *Best Practices for Software Development Teams*, <http://www-306.ibm.com/software/awdtools/rup/index.html>.
- [Scholtz i Morse] Scholtz J., Morse E., Using Customer Demands to Bridge the Gap between Software Engineering and Usability Engineering, *Software Process: Improvement and Practice*, vol. 8 (2003), s. 89–98.

- [Słownik wyrazów obcych,] Słownik wyrazów obcych, Warszawa, PWN 1997.
- [Sommerville] Sommerville I., *Software Engineering*, 7th Edition, Pearson & Addison-Wesley 2004.
- [Subramanyam i inni] Subramanyam R., Weisstein F. L., Krishnan M. S., User Participation in Software Development Projects, *Communications of the ACM*, vol. 53, March 2010, s. 137–141.
- [Tiwana i Keil] Tiwana A., Keil M., Functionality Risk In Information Systems Development: An Empirical Investigation, *IEEE Trans. on Engineering Management*, vol. 53 (2006), s. 412–425.
- [Vilpola] Vilpola I. H., A method for improving ERP implementation success by the principles and process of user-centred design, *Enterprise Information Systems*, vol. 2, February 2008, s. 47–76.
- [XPRINCE] Równowaga między zwinnością a dyscypliną w projektach informatycznych, http://xprince.net/xprince_folder (dostępne w lipcu 2009).