Propozycja modelu dojrzałości dla Programowania Ekstremalnego*

Jerzy R. Nawrocki
Politechnika Poznańska, Instytut Informatyki

Bartosz Walter
Politechnika Poznańska, Instytut Informatyki

Adam Wojciechowski
Politechnika Poznańska, Instytut Informatyki

Streszczenie

Rosnąca popularność lekkich metodyk rozwoju oprogramowania, takich jak Programowanie Ekstremalne (XP od ang. *eXtreme Programming*) skupia uwagę wielu zespołów programistycznych. Atrakcyjność tego stylu pracy płynie zarówno z ciekawego doboru praktyk programistycznych, jak i rezygnacji z wielu procedur formalnego dokumentowania postępów. Z naszych doświadczeń wynika jednak, że wiele osób traktuje XP jako wymówkę dla zaniedbań w realizacji tradycyjnych procedur zapewniania jakości w budowie oprogramowania. Pełna realizacja praktyk proponowanych przez XP jest bardzo trudna. Potrzebny byłby więc model dojrzałości, który umożliwiłby ocenę stopnia zaawansowania zespołu wykonawców w stosowaniu praktyk XP. W pracy zaproponowany został prosty 4-poziomowy model dojrzałości dla zespołów stosujących XP w procesie wytwarzania oprogramowania.

1. Wprowadzenie

Metodyki wytwarzania oprogramowania określane mianem *lekkich*, reprezentowane m.in. przez Programowanie Ekstremalne (XP od ang. *eXtreme Pro-*

* Praca wykonana w ramach projektu badawczego 43-364/01-BW.

gramming), szybko zdobywają popularność wśród zespołów programistów. Atrakcyjność takiego stylu pracy płynie częściowo z ciekawego doboru praktyk programistycznych. Praktyki XP zostały bowiem tak zdefiniowane, aby tworzyć zbiór wzajemnie uzupełniających się procedur wytwarzania oprogramowania oraz zapewnienie jakości i osiągnięcie satysfakcji klienta [2, 3]. Silnym argumentem przemawiającym za stosowaniem w praktyce podejścia XP jest rezygnacja z formalnych inspekcji oprogramowania oraz zaniechanie obowiązku gromadzenia szczegółowej i zwykle bardzo obszernej dokumentacji wymagań, projektu, raportów z testowania itp.

Z naszych obserwacji wynika, że dla wielu programistów XP stanowi wymówkę dla zaniedbań w zakresie realizacji tradycyjnych procedur zapewnienia jakości w procesie budowy oprogramowania. Porzucając klasyczny model dokumentowania prac nad budową oprogramowania, zespoły te nie realizują w pełni praktyk wymaganych w podejściu XP. Niestety, część z praktyk XP bywa trudna w realizacji. Dobrym przykładem może być tutaj wymaganie stałej obecności przedstawiciela klienta władnego podejmować decyzje o charakterze biznesowym odnoszące się do projektu. Ważne jest aby reprezentant klienta, stale współpracujący z zespołem programistów, odporny był na konflikty o charakterze politycznym w swej macierzystej organizacji [1]. Rola osób o takich kompetencjach i osobowości bywa jednak na tyle istotna dla firmy klienta, że pracownik taki jest trudno dostępny dla zespołu programistów.

Należy zatem rozróżnić stopnie zaawansowania w realizacji praktyk XP. Poziomy dojrzałości mogą stanowić miarę ryzyka związanego z realizacją projektu przez dany zespół programistów. Warto również, aby podobnie jak w modelu CMM(I) [5], poziomy dojrzałości wskazywały kierunek doskonalenia procesów budowy oprogramowania i wzrost satysfakcji klienta.

W pracy zaproponowany został prosty 4-poziomowy model dojrzałości dla zespołów realizujących projekty informatyczne wg metodyki XP. W rozdziale drugim przedstawione zostały zasadnicze praktyki modelu XP. Rozdział trzeci stanowi krótki przegląd uznanych modeli dojrzałości dotyczących wytwarzania oprogramowania. Z tego przeglądu wyciągnęliśmy wniosek, iż najbardziej odpowiedni będzie model przypominający CMMI i PSP. W czwartym rozdziale przedstawiona została propozycja modelu dojrzałości dla metodyki XP. Poszczególne poziomy proponowanego modelu zostały szerzej opisane w rozdziale piątym. Problem oceny dojrzałości zespołu programistów został poruszony w rozdziale szóstym.

2. Krótka charakterystyka Programowania Ekstremalnego

Programowanie Ekstremalne jest *lekką* metodyką budowy oprogramowania [2]. XP jest zorientowane na szybką budowę oprogramowania poprzez przyrostowe dodawanie własności funkcjonalnych poczynając od tych, które mają istotne znaczenie dla klienta. Elastyczny charakter tego podejścia do budowy oprogramowania płynie z doświadczeń kierowników projektów, programistów i osób odpowiedzialnych za logikę biznesową w projektach informatycznych. Model ten zakłada preferencję nieformalnej, ustnej komunikacji wewnątrz zespołu nad metodami formalnego dokumentowania pracy w postaci zapisów tekstowych. Stała obecność przedstawiciela klienta, który w czasie prac nad projektem staje się członkiem zespołu wykonawców daje szansę podporządkowania decyzji projektowych potrzebom i możliwościom (finansowym) klienta. XP jest podejściem proponowanym dla zespołów kilku- lub kilkunastoosobowych i opiera się na pracy par programistów (często XP jest utożsamiane z programowaniem parami).

Przystępując do budowy modelu dojrzałości konieczna jest identyfikacja praktyk stosowanych w XP. Podobnie jak w [3], podzieliliśmy praktyki XP na pięć kategorii: planowanie, projektowanie, kodowanie, testowanie oraz warsztat pracy.

Planowanie:

P0. Plan przedsięwzięcia jest rezultatem gry planistycznej.

P1. Dokumentacja projektowa ograniczona do kodu i przypadków testowych.

P2. Dialog z klientem udokumentowany w formie opisów zastosowań.

P3. Podstawowym okresem planowania jest wydanie.

P4. Częste wydania przyrostowo zwiększające funkcjonalność systemu.

P5. Podstawową miarą z punktu widzenia planowania jest prędkość realizacji przedsięwzięcia.

P6. Realizacja projektu podzielona na iteracje.

P7. Realizacja iteracji otwierana sesją planowania.

P8. Przemieszczanie osób między parami programistów.

P9. Krótka narada „na stojąco” na początku każdego dnia pracy.

P10. Proponowanie zmian praktyk XP, kiedy dotychczasowy model zawodzi.

Projektowanie:

D0. Dążenie do maksymalnej prostoty projektu.

D1. Metafora – prosty opis projektu, oparty na analogiach do innych systemów lub obiektów znanych klientowi i producentowi.

D2. Stosowanie kart CRC (ang. Class, Responsibility, Communication) w czasie sesji projektowania.

D3. Eksperymentalne pozyskiwanie wiedzy nt. technologii i związanych z nimi czynnikami ryzyka.

D4. Unikanie rozszerzania własności funkcjonalnych nie zwiększających wartości systemu z punktu widzenia klienta.

D5. Częsta refaktoryzacja, czyli optymalizacja i modyfikacja kodu podnosząca wartość systemu, bez zmniejszenia realizowanej funkcjonalności.

Kodowanie:

C0. Reprezentant klienta stale współpracujący z zespołem programistów.

C1. Konsekwentnie stosowany standard kodowania i dokumentowania kodu.

C2. Kodowanie jest poprzedzone konstrukcją testów jednostkowych.

C3. Programowanie realizowane w parach programistów.

C4. Tylko jedna para programistów może w danym momencie integrować nowy fragment kodu z systemem.

C5. Ciągła integracja - każda para programistów może dowolnie często integrować swój kod z systemem dodając tym samym nową funkcjonalność.

C6. Wszyscy członkowie zespołu programistów współdzielą cały kod.

C7. Optymalizacja kodu w końcowej fazie pracy nad systemem.

C8. Ośmiogodzinny dzień pracy.

C9. Stosowanie narzędzi i metod systematycznego zarządzania wersjami.

Testowanie:

T0. Cały kod musi posiadać testy jednostkowe.

T1. Przed kolejnym wydaniem każdy fragment kodu musi pomyślnie przejść przez testy jednostkowe.

T2. Kompletność testów – znalezienie błędu nie wykrywanego przez testy jednostkowe implikuje konieczność uzupełnienia przypadków testowych.

T3. Testy akceptacyjne są wykonywane często, a ich rezultaty są dostępne dla członków zespołu.

T4. Stosowanie algorytmów i narzędzi automatycznego testowania dla realizacji testów integracyjnych.

Warsztat pracy:

F0. Laboratorium zespołu programistów. Centralny sektor pracy programistycznej otoczony stanowiskami do pracy indywidualnej [2].

3. Modele dojrzałości

Capability Maturity Model Integration (CMMI^{SM}) zaproponowany przez Software Engineering Institute (SEI) stanowi schemat reprezentujący ścieżki poprawy procesów wytwarzania oprogramowania [5]. Model CMMI jest zorientowany na doskonalenie pracy organizacji i jest niezależny od dziedziny działalności. Hierarchiczna, 5-cio poziomowa struktura CMM(I) stała się szeroko akceptowanym wyznacznikiem dojrzałości procesów stosowanych wewnątrz firmy. Z wyjątkiem poziomu pierwszego (ang. *Initial*), każdy z poziomów jest dekomponowany na obszary (ang. *Key Process Areas*, KPA). Każdy KPA definiuje zestaw czynności, których łączne wykonanie pozwala osiągnąć cele istotne dla dojrzałości procesu wytwórczego w firmie.

W świetle prac nad modelem dojrzałości dla XP, z naszego punktu widzenia do istotnych charakterystyk CMMI(I) należą:

- Model CMM(I) składa się z pięciu poziomów dojrzałości. Z każdym poziomem związany jest zbiór praktyk (rozumianych jako sposób realizacji zadań w poszczególnych KPA) i wszystkie praktyki związane z danym poziomem CMM(I) muszą być wdrożone w celu uznania stopnia dojrzałości organizacji. Proponowany model dojrzałości w sensie XP, zakłada 4-poziomową strukturę obowiązkowych praktyk.
- Firmy, których sposób działania nie przystaje do modelu CMM(I) klasyfikowane są na poziomie 1. Również w proponowanym modelu dojrzałości dla XP przyjęliśmy zasadę zaklasyfikowania na poziom 1 wszystkich tych organizacji, których działalność nie przystaje do obowiązkowych praktyk wymaganych na wyższych poziomach tego modelu.
- Poziom 2 modelu CMM(I) jest zorientowany na projekt, podczas gdy na wyższych poziomach dojrzałości (3, 4, i 5) nacisk kładziony jest cele organizacyjne. Według naszej oceny, podejście XP jest zorientowane na projekty, nie zaś na cele organizacyjne. Kent Beck w swojej książce [2] jasno stwierdza, że metodologia XP najlepiej nadaje się dla małych zespołów programistów (do 10 osób). Zatem nasz model, w przeciwieństwie do CMM(I), będzie zorientowany na zespoły związane z projektami budowy oprogramowania.
- W modelu CMM(I), zorientowanym na doskonalenie procesów wewnątrz firmy, dojrzałość organizacji oceniana jest na podstawie zestawu praktyk, które są realizowane. Nie ma tu bezpośredniej oceny wyników pracy. Założeniami XP są m.in. satysfakcja klienta i brak pracy ponadnormatywnej.

Zatem celem naszym jest, aby model dojrzałości dla XP wyraźnie rozgraniczał projekty „udane", czyli zrealizowane w założonym czasie i z pełną satysfakcją klienta od projektów zakończonych częściowym sukcesem lub porażką.

Model PSP zorientowany na doskonalenie pracy indywidualnego programisty [7] jest strukturalnie podobny do modelu CMMI. Różna jest tu liczba poziomów dojrzałości i inne niż w CMMI są praktyki wymagane od programisty w celu zaklasyfikowania jego stylu pracy na kolejne poziomy. Celem realizacji coraz bardziej zaawansowanych praktyk jest wzrost jakości wytwarzanego oprogramowania, doskonalenie metod planowania pracy i przewidywania efektów oraz ulepszanie procesu wytwarzania oprogramowania.

Celem procesu organizacji doskonalącej swój proces wytwarzania oprogramowania jest zrozumienie i powielanie sukcesów oraz poznanie i unikanie błędów [8]. Odrębne podejście do doskonalenia procesu wytwarzania oprogramowania zostało przedstawione przez NASA. Model ten zorientowany jest na końcowy produkt i zakłada cykliczne doskonalenie procesów wytwórczych wewnątrz organizacji. W modelu tym nie ma zdefiniowanych poziomów dojrzałości, ani uniwersalnych zestawów praktyk. Każdy etap poprawy procesów jest podzielony na trzy fazy: poznanie (ang. *understanding*), ocena (ang. *assessing*) i opracowanie (ang. *packaging*). Dobre praktyki, które mają być wdrożone przez daną organizację są identyfikowane w fazie poznania. Następnie ich pilotażowe wdrożenie podlega ocenie. Jeśli uzyskane wyniki okazują się pozytywne opracowywane są procedury poprawy procesów, które następnie są dystrybuowane do poszczególnych działów firmy w celu wdrożenia gotowych procedur poprawy procesów. Ponieważ praktyki XP są ściśle określone i opisane, uznaliśmy, że model zaproponowany w NASA nie jest właściwym sposobem doskonalenia kompetencji w zakresie wytwarzania oprogramowania wg metodologii Programowania Ekstremalnego.

4. Proponowany model dojrzałości dla Programowania Ekstremalnego, XPMM

Trudno jest jednoznacznie ocenić, czy dany zespół pracuje zgodnie z regułami XP. Zbiór praktyk XP został przedstawiony m.in. w pracach [2, 3]. Nie ma tam jednak wskazówek jaka część wymienionych praktyk musi zostać wdrożona, aby można było mówić o pracy zgodnej z XP. Jednoznaczne ocenianie pracy jako niezgodnej z duchem Programowania Ekstremalnego w sytuacji niespełnienia wymogów niektórych praktyk może być krzywdzące dla zespołu programistów. Brak pełnej realizacji niektórych praktyk bywa konsekwencją czynników zewnętrznych i nie musi oznaczać braku zaangażowania zespołu w realizację projektu wg zasad XP. Rozważmy następujące sytuacje:

- W wielu przypadkach programiści nie mają fizycznej możliwości, aby cały kod powstawał w czasie pracy w parach.
- Przedstawiciel klienta nie zawsze będzie mógł pozostawać z zespołem programistów przez cały czas pracy nad systemem.
- Zarządzanie wersjami oprogramowania powinno być w pełni zautomatyzowane poprzez zastosowanie narzędzi zarządzania wersjami. Obowiązek ten zespół może jednak powierzyć osobie, która będzie odpowiedzialna za administrowanie wersjami i integrację kodu?
- Ze względu na wewnętrzne i zewnętrzne ograniczenia, zespół programistów może stanąć przed koniecznością pracy w godzinach nadliczbowych. Niestety, praktyka ta nie należy do rzadkości w wielu firmach.

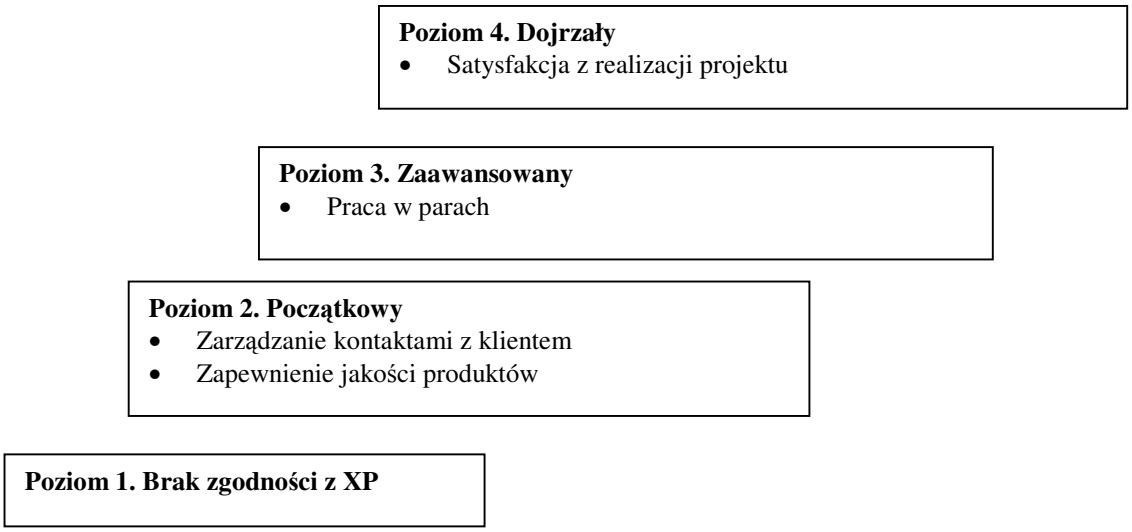
Wszystkie przedstawione sytuacje stanowią pewne odstępstwa od praktyk XP. Należy jednak zauważyć, że spełniając pozostałe warunki określane zestawem praktyk, taki zespół programistów byłby bardzo bliski ideałowi pracy zgodnej z XP. W ocenę taką wkrada się jednak pewien relatywizm i subiektywne odczucia odnośnie znaczenia poszczególnych praktyk dla XP. Rozwiązaniem obiektywnej oceny stopnia realizacji XP byłby hierarchiczny model dojrzałości określający zakres obowiązkowych praktyk wymaganych na poszczególnych poziomach.

Konstruując model dojrzałości dla XP, który miałby być stosowany w firmach produkujących oprogramowanie warto wziąć pod uwagę wymiar praktycznej użyteczności modelu dla firmy i jej klientów. Do istotnych czynników wpływających na możliwość zaakceptowania modelu w środowisku przemysłowym należą:

- Niski koszt wdrożenia modelu i oceny pracy zespołu.
- Jasna wizja związku pomiędzy poziomem dojrzałości zespołu (w sensie przyjętego modelu) a wytwarzanymi produktami.
- Czytelna hierarchiczna struktura obowiązkowych praktyk XP związanych z każdym poziomem dojrzałości.
- Osiągnięcie wyższego poziomu dojrzałości powinno wiązać się z uzyskaniem korzyści płynących z poprawy procesu i lepszego wykorzystania dostępnych technologii (wzrost kompetencji).
- Poprawa procesu wytwarzania oprogramowania musi iść w parze ze wzrostem jakości produktów (gwarancja jakości dla klienta).
- Model dojrzałości powinien być równie *lekki* jak metodologia XP.

Struktura proponowanego modelu stanowi nawiązanie do CMM(I) i PSP. Składa się on z czterech poziomów: 1. Brak zgodności z XP, 2. Początkowy, 3. Zaawansowany, 4. Dojrzały. Osiągnięcie wyższego poziomu dojrzałości idzie w parze ze wzrostem kompetencji zespołu programistów. Zaklasyfikowanie ze-

społu na danym poziomie oznacza pełną realizację wszystkich praktyk wymaganych na tym poziomie oraz wszystkich praktyk z poziomów niższych.



Rys. 1. 4-poziomowa struktura proponowanego model dojrzałości dla XP

5. Poziomy dojrzałości w proponowanym modelu

Zespoły, które w trakcje realizacji projektów nie stosują się do praktyk XP zostają, w proponowanym modelu, automatycznie zaklasyfikowane na poziomi 1 – Brak zgodności z XP.

5.1. Poziom 2 – Początkowy

Choć metodyka XP różni się od podejścia proponowanego w CMM(I) należy zauważyć, oba modele mają również cechy wspólne. Na drugim poziomie obu tych modeli nacisk położony jest na pracę zespołu programistów. Interesujące jest zatem porównanie w jakim zakresie wybrane praktyki z modelu CMMI [5] znajdują przełożenie na praktyki XP. Naszym zdaniem, w szczególnym stopniu dwie spośród praktyk XP znajdują odbicie w praktykach modelu CMMI:T3 oraz P0.

Obowiązkowe praktyki XP na drugim poziomie dojrzałości proponowanego modelu obejmują:

- W zakresie zarządzania kontaktami z klientem: P0, P2, P3, P4, P5, P6, P7, D1, D3, D4, C0a (warunek osłabiony do czynnego udziału klienta w konstruowaniu przypadków użycia, testów akceptacyjnych i w grze planistycznej; współpraca może się dobywać na zasadzie telekonferencji lub wymiany korespondencji elektronicznej).
- W zakresie zapewnienia jakości produktu: C2, C5a (na tym poziomie zakładamy, że integracja oprogramowania powinna odbywać się co najmniej raz w tygodniu), C7, T0, T1a (każdy fragment kodu musi zostać poddany testom jednostkowym; osłabienie warunku 100% poprawnej realizacji testów jednostkowych), T2, T3.

5.2. Poziom 3 –Zaawansowany

Poziom zaawansowany jest zorientowany na pracę w parach. Jedna osoba pełni rolę lidera kodowania, a druga – lidera testowania. W czasie pracy programiści zamieniają się rolami i wspólnie biorą odpowiedzialność za opracowywany kod. Silną stroną takiej formy pracy jest forma ciągłego przeglądu kodu oraz fakt, że każdy fragment kodu ma dwóch autorów, z których każdy może zająć się pielęgnacją kodu po jego wdrożeniu.

Obowiązkowy zestaw praktyk na poziomie 3 obejmuje:

- W zakresie pracy parami: P8, C0b (wizyta reprezentanta klienta co najmniej raz w tygodniu; nawiązanie osobistych kontaktów z programistami), C1, C3, C4, C5b (integracja kodu co najmniej raz na dzień [2]), C6, C8b (zbieranie i publikowanie informacji o pracy w godzinach nadliczbowych), C9, T4, F0.

5.3. Poziom 4 – Dojrzały

Na poziomie 4 – dojrzałym, nacisk położony jest na satysfakcję programistów i klienta. W przeciwieństwie do CMMI i PSP, w modelu dojrzałości dla XP wzięte zostały pod uwagę korzyści płynące dla zespołu wykonawców i klienta będące efektem realizacji zdefiniowanego procesu wytwarzania oprogramowania.

Obowiązkowy zestaw praktyk na poziomie 4 obejmuje:

- W zakresie satysfakcji z realizacji projektu: C0, C8, T1.

6. Problem oceny dojrzałości

Lekki charakter metodyki XP, zakładający m.in. minimalizację dokumentacji towarzyszącej pracom nad projektem budowy oprogramowania, oraz nieostre sformułowanie części praktyk (np. D0, D3, D4), stanowią pewną trudność w obiektywnej ocenie zespołów. Naszym zdaniem, obserwacja, rozmowy i towarzyszenie programistom w ich pracy stanowią najlepszą technikę poznania stopnia realizacji praktyk XP. Informacje dotyczące realizacji poszczególnych praktyk mogą być zbierane w formie ankiety obejmującej wszystkie praktyki, których realizacja jest przewidziana na poszczególnych poziomach dojrzałości.

7. Podsumowanie

W pracy przedstawiona została propozycja modelu dojrzałości dla Programowania Ekstremalnego. Charakter modelu zakłada *lekkość* typową dla podejścia XP. 4-poziomowa konstrukcja modelu dojrzałości wzorowana jest na CMM(I). Najwyższy poziom odnosi się bezpośrednio do jakości pracy (brak godzin nadliczbowych, pełna realizacja testów modułowych, satysfakcja klienta). Zastosowanie proponowanego modelu dojrzałości pozwala na ocenę i rozpoznanie projektów realizowanych zgodnie z metodyką XP i tych, w których pseudo-XP stanowi jedynie wymówkę dla braku spisanej dokumentacji.

Bibliografia

[1]. E. Yourdon, Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects, Prentice Hall, 1999.

[2]. K. Beck, Extreme Programming Explained: Embrace Change, Addison Wesley, Boston, 2000.

[3]. J. Donovan Wells, The Rules and Practices of Extreme Programming, http://www.extremeprogramming.org/rules. html, February 2001.

[4]. L. Williams, R. Kessler, W. Cunningham, R. Jeffries Strengthening the Case for Pair Programming, IEEE Software, vol. 17 (2000), No. 4, 19-25.

[5]. CMMI^{SM} for Systems Engineering/Software Engineering/ Integrated Product and Process Development, Version 1.02, CMU/SEI-2000-TR-030 ESC-TR-2000-095, Software Engineering Institute, Carnegie Mellon University, http://www.sei.cmu.edu/pub/documents/00.reports/pdf/ 00tr030.pdf, 12.12.2000.

[6]. I. Sommerville, P. Sawyer, Requirements Engineering. A good practice guide, John Wiley & Sons, 1998.

[7]. W.S. Humphrey, Discipline for Software Engineering, Addison-Wesley, Reading, 1995.

[8]. http://www.ivv.nasa.gov/SWG/docs/process/, Feb. 2001.

[9]. J.T. Nosek, The case for collaborative programming, Communications of the ACM, vol. 41 (1998), No. 3, 105-108.

[10]. J. Nawrocki, A. Wojciechowski, Experimental evaluation of Pair Programming, **w**: K. Maxwell, S. Oligny, R. Kusters, E. Van Veenendaal (red.), Project Control: Satisfying the Customer, Proceedings of the 12th European Software Control and Metrics Conference ESCOM 2001, Shaker Publishing, Maastricht, 2001, 269 – 276.

[11]. J.M. Carroll et al., Requirements development in scenario-based design, IEEE Transactions on Software Engineering, vol. 24 (1998), No. 12, 1156-1170.