Delficka gra planistyczna typu XP

Michał Jasiński
Instytut Informatyki, Politechnika Poznańska
e-mail: vitric@fanthom.math.put.poznan.pl

Jerzy R. Nawrocki
Instytut Informatyki, Politechnika Poznańska
e-mail: Jerzy.Nawrocki@cs.put.poznan.pl

Streszczenie

Planowanie, a planowanie zakresu działań w szczególności, jest jednym z najistotniejszych etapów w procesie rozwoju oprogramowania. W pracy krótko przedstawiono metodę delficką (ang. *Wideband-Delphi Method*) w ujęciu klasycznym oraz zaproponowano podejście wykorzystujące tę metodę w grze planistycznej typu XP, mającej na celu zaplanowanie zakresu działań w przedsięwzięciu informatycznym.

1. Wprowadzenie

Planowanie jest jednym z kluczowych czynników sukcesu każdego przedsięwzięcia, w tym przedsięwzięcia informatycznego. Klasyczne podejście do planowania polega na oszacowaniu, na podstawie specyfikacji wymagań, pracochłonności przedsięwzięcia i opracowaniu, na tej podstawie, jego harmonogramu. Ten typ myślenia jest charakterystyczny m.in. dla CMM(I) [3] i PSP [7]. Inne podejście do planowania jest proponowane w metodologii XP (ang. *eXtreme Programming*) [1]. Proces wytwarzania oprogramowania jest dzielony na wydania (ang. *releases*), a te składają się z przyrostów (ang. *increments*). Liczba wydań nie jest z góry znana. Planowanie nie wykracza poza horyzont pojedynczego wydania. Czas realizacji wydania jest z góry określony i zadaniem planowania jest zdefiniowanie wymagań mających postać historyjek użytkownika

(ang. *user stories*), które będą zrealizowane w tym okresie. Planowanie przyrostów odbywa się na podobnej zasadzie. W XP planowanie ma postać gry między przedstawicielem klienta a informatykami, w której:

- Przedstawiciel klienta sortuje zebrane historyjki od najważniejszej do najmniej ważnej.
- Informatycy identyfikują czynniki ryzyka związane z poszczególnymi historyjkami użytkownika i określają ich pracochłonność.
- Przedstawiciel klienta dokonuje wyboru historyjek, które mają być zrealizowane w przyjętym okresie.

Jest to bardzo atrakcyjna koncepcja, niemniej jednak ma pewne wady:

- W XP zakłada się, że przedstawiciel klienta jest jeden. Nie jest jasne, jak wspomniana gra planistyczna ma wyglądać, gdy przedstawicieli klienta jest wielu i każdy z nich ma inny pogląd na ważność historyjek i zakres najbliższego wydania.
- XP jest bardzo „lekką" metodyką wytwarzania oprogramowania. Autorzy XP przyjęli, że dokumentacja obejmuje jedynie kod i przypadki testowe. Powstaje zatem pytanie, jak można podejście do planowania proponowane przez XP dostosować do sytuacji, gdy tworzenie oprogramowania obejmuje scenariusze użycia i specyfikację wymagań (np. zgodnie ze standardem IEEE 830).

Celem pracy jest zaproponowanie modyfikacji gry planistycznej wykorzystywanej w XP, która umożliwiałaby udział w procesie decyzyjnym wielu przedstawicieli klienta i obejmowałaby szeroko rozumianą inżynierię wymagań (wraz z opracowywaniem scenariuszy użycia). Główny pomysł polega na połączeniu gry planistycznej wykorzystywanej w XP z metodą delficką [11]. Idea metody delfickiej jest bardzo ogólna, opiera się na idei „burzy mózgów" (ang. *brain storm*) i wymaga zawsze pewnego dostosowania. W artykule przestawiamy wersję metody wkomponowaną w grę planistyczną typu XP i dostosowaną do scenariuszy użycia oraz inżynierii wymagań.

2. Metoda delficka

Metoda delficka została opracowana w latach czterdziestych XX wieku w firmie RAND Corporation przez O. Helmera i N. Dalkey'a. Początkowo była to technika przewidywania przyszłych wydarzeń. Z biegiem czasu metoda ta ewoluowała i ostatecznie została zdefiniowana, jako struktura komunikacji grupowej, umożliwiająca rozwiązywanie określonych zadań [11]. Jest ona przede wszystkim wycelowana w szczegółową, krytyczną analizę i dyskusję, nie zaś w dążenie za wszelką cenę do osiągnięcia szybkiego kompromisu.

Podstawą metody jest stworzenie elastycznych ram służących grupowemu procesowi komunikacji. Struktura taka, angażująca moderatora, opowiadającego za przebieg dyskusji, oraz recenzentów, obejmuje kolejno kroki:

- Rozpoznanie i zdefiniowanie problemu.
- Zmianę reprezentacji.
- Identyfikację i opracowanie celów, które mają zostać osiągnięte poprzez rozwiązanie danego zagadnienia.
- Określenie strategii tworzenia możliwych rozwiązań.
- Wybór strategii.
- Stworzenie kryterium oceny znalezionych rozwiązań.
- Ocenę kryterium.
- Wypracowanie rozwiązań.
- Ocenę rozwiązań.

Należy podkreślić, iż ewaluacja omawianych rozwiązań jest realizowana w sposób anonimowy, podczas gdy pozostałe etapy mają charakter jawny. Do pozostałych cech charakterystycznych metody delfickiej należą: asynchroniczna interakcja, brak presji, brak zniekształceń oszacowań oraz uwzględnianie wielu aspektów danego problemu.

Jednak w klasycznym ujęciu, metoda delficka jest w pewnym sensie ograniczona, ze względu na „jednowymiarowość" wygłaszanych opinii. Uczestnicy dyskusji, to zwykle eksperci dysponujący zbliżonym poziomem wiedzy i porozumiewający się tym samym, fachowym językiem. W niniejszej pracy prezentujemy zastosowanie tej metody w postaci niejednorodnej (wielowymiarowej), tj. mającej formę dyskusji osób myślących w zindywidualizowanych kategoriach oraz posiadających wiedzę o zróżnicowanym poziomie szczegółowości i dotyczących różnych domen. Sytuacja taka ma najczęściej miejsce w przypadku rozmów pomiędzy informatykami, reprezentującymi producenta systemu, a przedstawicielami klienta, na którego zamówienie oprogramowanie to powstaje. Podejście to znacząco zwiększa wpływ przyszłych użytkowników rozwiązania informatycznego na jego ostateczny kształt. Uważamy przy tym, iż wszelkie decyzje, co do strony użytkowej powstającego systemu powinien podejmować klient (tak, jak proponuje to XP). Informatycy pełnią w tym procesie jedynie role doradców.

3. Planowanie w cyklu życia oprogramowania

Złożony proces produkcji i eksploatacji oprogramowania powinien być realizowany w systematyczny i zorganizowany sposób. Pozwala to porządkować

przebieg prac, ułatwia planowanie zadań, a także umożliwia kontrolę przebiegu ich wykonywania. Proponuje się więc usystematyzowane metodologie postępowania, definiujące szczegółowy kształt tych procesów, nazywane *modelami cyklu życia oprogramowania*. Wśród opisywanych w literaturze podejść, dominują: model kaskadowy (ang. *waterfall*), model spirali, prototypowanie (ang. *prototyping*), programowanie odkrywcze (ang. *exploratory programming*), realizacja przyrostowa (ang. *incremental development*).

W niniejszej pracy zależało nam na zaproponowaniu takiego modelu cyklu życia oprogramowania, który:

- Podobnie jak w podejściu XP, zapewniałby krótkie „sprzężenie zwrotne" (ang. *feedback*), pomiędzy zespołem projektowym, a przedstawicielami klienta.
- W przeciwieństwie do XP, uwzględniałby konieczność udokumentowania wymagań użytkownika (tak w postaci scenariuszy użycia, jak i specyfikacji wymagań, zgodnej np. ze standardem ANSI/IEEE 830-1984).

W ramach tak skonstruowanego modelu, przyjęliśmy następujące założenia:

- Proces tworzenia oprogramowania podzielony jest na wydania. Każde wydanie kończy się wdrożeniem opracowanej wersji systemu u klienta. Czas realizacji wydania, mierzony w tygodniach lub miesiącach kalendarzowych, służy jako parametr wejściowy dla procesu planowania. W dalszych rozważaniach przyjmujemy, że wynosi on trzy miesiące (wartość ta jest akceptowana przez autorów metodologii XP).
- W ramach pojedynczego wydania wyróżnia się fazę inżynierii wymagań oraz kilka przyrostów. Uważamy, iż w przypadku wydań trzymiesięcznych, najlepszym rozwiązaniem są dwa przyrosty 1-miesięczne. Zatem na inżynierię wymagań zostaje miesiąc.
- Inżynieria wymagań obejmuje opracowanie scenariuszy użycia (dwa tygodnie) oraz specyfikację wymagań (kolejne dwa tygodnie). Pod koniec każdego tygodnia ma miejsce gra planistyczna, której celem jest skupienie działań następnego tygodnia na najważniejszych, z punktu widzenia klienta, obszarach tworzonego systemu.
- Przyrost tworzy funkcjonalnie spójną wersję systemu, z tym, że trafia ona jedynie do tych przyszłych użytkowników oprogramowania, którzy są zaangażowani w projekt po stronie klienta (wydanie jest przeznaczone dla szerszego grona przyszłych użytkowników systemu). Realizacja każdego przyrostu trwa około jednego miesiąca i obejmuje: projektowanie, kodowanie, integrację oraz testowanie danej wersji systemu.

W przedstawionym podejściu planowanie odbywa się na etapie inżynierii wymagań i przyrostów. Po zakończeniu pierwszego etapu znany jest zakres funkcjonalny danego wydania. W odróżnieniu od inżynierii wymagań planowanie na etapie

przyrostów odbywa się na początku każdego okresu (początek piątego i dziewiątego tygodnia), a nie pod koniec.

4. Planowanie zakresu na poziomie scenariuszy użycia

Faza opracowywania scenariuszy użycia [16] trwa dwa tygodnie. Kluczową rolę odgrywa w niej klient oraz jego reprezentanci, podczas gdy informatycy głównie kontrolują i weryfikują dostarczane informacje. Jeżeli założylibyśmy, że przez ten czas analitycy zajmują się zbieraniem scenariuszy użycia od przyszłych użytkowników systemu, zaś pod koniec tego okresu organizowane jest spotkanie planistyczne, w trakcie którego zostanie opracowany harmonogram prac nad wydaniem, to niesie to pewne niebezpieczeństwo. Istnieje bowiem groźba, iż informatycy będą wówczas dysponowali zbiorem scenariuszy, które dotyczyć będą bardzo ”implementacyjnie” odległych od siebie cech funkcjonalnych tworzonego rozwiązania informatycznego. W rezultacie, liczba scenariuszy opisujących kolejne wydania może okazać się zbyt mała.

Aby temu zapobiec, proponujemy dwuetapowe podejście do opracowywania scenariuszy, stopniowo uszczegółowiające najistotniejsze aspekty systemu. Każda z przedstawionych dalej faz powinna obejmować jeden tydzień pracy. Pierwsze trzy dni obejmują proces bezpośredniego opracowywania scenariuszy użycia na drodze rozmów pomiędzy analitykami systemowymi, a reprezentantami klienta. Pod koniec tygodnia (np. w czwartek) informatycy przygotowują krótki raport zawierający zebrane scenariusze oraz oceniają powiązania między nimi, pracochłonność i czynniki ryzyka.

Trójstopniowa analiza scenariuszy realizowana jest w trakcie, zorganizowanego na bazie metody delfickiej, spotkania analityków systemowych oraz programistów, którzy będą realizować opracowaną koncepcję rozwiązania informatycznego. Punktem wyjścia do szacowania pracochłonności jest średnia złożoność scenariusza wyznaczona na podstawie bazy danych historycznych organizacji tworzącej oprogramowanie. Zadaniem projektantów jest „jedynie” ocena, jak dalece rozważane scenariusze użycia odbiegają od przypadku średniego, przy czym możliwe jest uwzględnienie pewnego rozmycia przedziału (w zakresie np. ±10%). Jednak w tym celu należy najpierw oszacować, jaki jest stopień ich wzajemnej zależności. Czynność tę można zrealizować, przy pomocy proponowanego przez nas formularza powiązań przedstawionego na rysunku 1. Odpowiedni znak w polu leżącym na skrzyżowaniu odpowiedniego wiersza świadczy o istnieniu powiązania lub też jego braku. Ostatni wiersz zawiera sumę wszystkich zależności, reprezentując tym samym złożoność scenariusza, którego identyfikator znajduje się w danej kolumnie.

Proces ustalania zależności pomiędzy scenariuszami ma charakter dyskusji nad architekturą tworzonego rozwiązania informatycznego. Uczestnicy spotkania głosują anonimowo nad powiązaniami, a następnie, ponownie dyskutują nad uzyskanymi wynikami w kontekście architektury systemu. Proces ten powinien być kontynuowany, aż do momentu, w którym zostanie osiągnięty akceptowalny konsensus. Jeżeli nie uda się go osiągnąć w rozsądnym czasie, proces jest kończony, a rozbieżne opinie są uwzględniane na kolejnych etapach dotyczących szacowania pracochłonności oraz identyfikacji czynników ryzyka. Jeżeli jednak istnieją duże wątpliwości, a zależy nam na „bezpieczeństwie" procesu planowania, można założyć, iż pomiędzy spornymi scenariuszami nie ma żadnych zależności.

		Id Scenariusza					
		S_1	S_2	S_3	...	S_{n-1}	S_n
Id Scenariusza	S_1						
	S_2						
	S_3						
	...						
	S_{n-1}						
	S_n						
Złożoność:							

Rysunek 1. Formularz powiązań scenariuszy użycia.

Kolejnym etapem jest oszacowanie pracochłonności indywidualnych scenariuszy użycia. Proponujemy, aby również ta faza była zorganizowana według zasad metody delfickiej. Posiadając udostępnione przez informatora informacje dotyczące złożoności implementacji historii użytkownika, każdy z informatyków uczestniczących w spotkaniu, indywidualnie szacuje pracochłonność poszczególnych scenariuszy. Szacunki te są zbierane przez moderatora dyskusji, który tworzy raport końcowy. Jego postać przedstawiono na rysunku 2. Pierwsza kolumna zawiera identyfikator scenariusza, wraz tymi, które pozostają z nim w ewentualnej korelacji. Kolejne kolumny zawierają estymacje nakładu pracy, wykonane przez każdego z uczestników spotkania (z zaznaczonymi wartościami dla

każdego z uczestników), podczas gdy w ostatniej z nich podane jest oszacowanie średnie.

Scenariusz	**Oszacowanie $_1$**	**Oszacowanie $_2$**	**...**	**Oszacowanie $_n$**	**Średnia**
S_1					
S_2: S_1, S_3					
...					
S_m					

Rysunek 2. Formularz raportu szacowania pracochłonności scenariuszy.

Podobnie, jak w przypadku procesu ustalania zależności pomiędzy scenariuszami, także na tym etapie sugerujemy, aby po zebraniu wszystkich ocen nawiązać dyskusję dotyczącą architektury tworzonego oprogramowania, zaś sam proces delficki powtórzyć, co najmniej raz.

Ostateczna pracochłonność E pojedynczego scenariusza może być wyznaczona przy pomocy poniższego równania:

$$E = \frac{E_{MIN} + 4 \cdot E_{AVG} + E_{MAX}}{6} \quad (1)$$

gdzie E_{MIN} jest minimalną, E_{AVG} – średnią, zaś E_{MAX} – największą szacowaną pracochłonnością dla danego scenariusza.

Ostatnią fazą, w której bierze udział grupa ekspertów informatyków, jest ocena czynników ryzyka dotyczących tworzonego systemu. Każda z osób (proponujemy, aby nie było ich więcej niż cztery) biorących udział w spotkaniu, zgłasza 10 do 15 czynników ryzyka, wybierając je ze standardowej listy. Następnie, korzystając z metody budżetowej (każdy z uczestników procesu decyzyjnego ma do dyspozycji pewną liczbę punktów, które rozdziela między czynniki ryzyka), uczestnicy spotkania anonimowo starają się ograniczyć liczbę możliwych zagrożeń do 20 najistotniejszych, biorąc pod uwagę prawdopodobieństwo wystąpienia oraz wpływ danego czynnika na powodzenie projektu. Iloczyn tych dwóch wartości określa wagę danego zagrożenia, a ta jest ujawniana w raporcie przez moderatora. W przypadku znacznych różnic w ocenie, następuje dyskusja o ich wpływie i prawdopodobieństwie, a w konsekwencji – kolejne głosowanie. Proces ten jest powtarzany, aż do momentu uzyskania zadowalająco zbliżonych ocen.

Na tym kończy się faza analizy scenariuszy użycia. W ostatnim dniu tygodnia, raport opracowany na podstawie danych uzyskanych podczas dyskusji między analitykami, a programistami, jest prezentowany przedstawicielom

użytkownika. Będąc w posiadaniu danych determinujących złożoność i wykonalność systemu, każdy z reprezentantów tej ostatniej grupy, umieszcza w swoim lokalnym „plecaku" reprezentującym funkcjonalność wydania, scenariusze o łącznym koszcie nie większym niż maksymalna i nie mniejszym niż minimalna pracochłonność wydania. Proponujemy dwa podejścia do pakowania „plecaka" globalnego:

- Każdy scenariusz otrzymuje 1 punkt za umieszczenie go w „plecaku", zaś 0 w przeciwnym wypadku. Wadą takiego rozwiązania jest to, iż jeśli pewien scenariusz nie został uwzględniony przez żadnego z uczestników spotkania, ale był tego bardzo bliski, to dostanie on mniej punktów, niż scenariusz, który został umieszczony tylko w jednym „plecaku" lokalnym, choć z punktu widzenia wszystkich pozostałych osób był on plasowany na samym końcu.
- Podejście wykorzystujące metodę budżetową pozbawione jest wad poprzedniej propozycji. Każdy z reprezentantów użytkownika otrzymuje pewną pulę punktów, które może rozdzielić między scenariusze i jego przypisanie jest właściwe o tyle, o ile elementy o niezerowej liczbie punktów wymagają przynajmniej minimalnej pracochłonności wydania.

Punkty rozdzielone przez reprezentantów klienta są sumowane, co pozwala na ich uporządkowanie pod względem ważności. Aby zaplanować zakres wydania, należy wybierać kolejne pozycje z utworzonej w ten sposób klasyfikacji i umieszczać je w „plecaku" globalnym, wraz z powiązanymi z nimi elementami, o ile nie została przekroczona jego dopuszczalna pojemność.

Ostateczny kształt „plecaka" powinien zostać poddany ocenie na drodze procesu delfickiego angażującego decydentów ze strony przedstawiciela klienta. Jeśli zostanie on zaakceptowany, możliwe jest przejście do etapu specyfikacji wymagań. W przeciwnym wypadku konieczna jest ponowna dyskusja o kształcie tworzonego rozwiązania informatycznego i kolejne głosowanie.

5. Planowanie zakresu na poziomie wymagań

Punktem wyjścia do planowania na poziomie wymagań jest, uporządkowana pod względem istotności, lista scenariuszy użycia, które w na poprzednim etapie zostały umieszczone w „plecaku" globalnym. To na ich podstawie analitycy opracowują poszczególne wymagania funkcjonalne i niefunkcjonalne. Następnie, według analogicznego schematu, jak w przypadku planowania na poziomie scenariuszy użycia, realizowana jest trójetapowa analiza i ewaluacja wyspecyfikowanych wymagań, na którą składa się ocena powiązań pomiędzy nimi, oszacowanie ich pracochłonności oraz identyfikacja czynników ryzyka. Podstawą do tego procesu jest baza danych historycznych, zawierająca informacje dotyczące średniej zależności pomiędzy nakładem pracy na implementację pojedynczego

wymagania, a liczbą wymagań funkcjonalnych (wymagania niefunkcjonalne nie są rozpatrywane, ze względu na brak jednoznacznej korelacji między nimi, a szacowaną pracochłonnością).

Wynikiem tego procesu jest specyfikacja, zawierająca uporządkowany zestaw wymagań, wraz z szacowanym nakładem pracy oraz szczegółową analizą czynników ryzyka. Ponadto, dysponując stworzoną hierarchią logicznych funkcji systemu, informatycy tworzą projekt konceptualny, definiujący zakres realizacji dla pojedynczego wydania.

6. Planowanie zakresu na poziomie przyrostów

Głównym celem planowania na tym etapie jest opracowanie harmonogramu zadań, które należy zrealizować w ramach danego przyrostu. Model konceptualny, wraz ze scenariuszami użycia i specyfikacją wymagań jest dobrym punktem wyjścia, lecz nie obejmuje wszystkich istotnych aspektów problemu. Podstawową kwestią, jaką muszą rozpatrzyć decydenci są *zadania stowarzyszone*, tzn. takie, którym należy poświęcić pewien czas, pomimo, iż nie dotyczą bezpośrednio tworzenia oprogramowania (np. rozpoznanie technologii, narzędzi). Mogą one mieć znaczy wpływ na „realizowalność" wymagań, zwłaszcza w przypadku początkowych przyrostów. Rozważyć należy również strukturę zespołu zajmującego się realizacją systemu informatycznego. Proponujemy, aby model współpracy informatyków tworzących system kładł nacisk nie na znalezieniu konkretnego rozwiązania, lecz na pozyskiwanie wiedzy przez wszystkich członków zespołu. Oznacza to, iż powinna istnieć pewna grupa osób kompetentna w zakresie pielęgnacji całego oprogramowania, podczas gdy część rozwiązań może być realizowana przez pracowników pełniących rolę *konsultantów*, zajmujących się projektem jedynie przez pewien czas i w ograniczonym zakresie. Kolejnym problemem, z jakim należy się zmierzyć, to monitorowanie, zarządzanie i łagodzenie wpływu czynników ryzyka (ang. *Risk Monitoring, Management and Mitigation*). Wreszcie, w przypadku, gdy tworzone plany dotyczą drugiego lub kolejnych przyrostów, powinno się uwzględnić postępy w realizacji poprzednich etapów.

Delficki proces decyzyjny w tej fazie ma na celu określenie, jakie wymaganie mają zostać zrealizowane w ramach danego przyrostu. Przedstawiciele klienta oraz informatycy rozstrzygają o tym, dyskutując nad danymi dotyczącymi nałożonych ograniczeń (organizacja zespołu i liczba oraz charakter zadań stowarzyszonych, czynniki ryzyka). W przypadku braku jednoznacznej decyzji, ostatnie słowo, co do wybranego zestawu wymagań mają przedstawiciele klienta.

Rezultatem tego etapu, rozpatrywanego dla każdego zaplanowanego przyrostu, powinien być zbiór zadań do realizacji w ciągu najbliższego miesiąca

oraz ocena czynników ryzyka, wykonana pod kątem minimalizacji prawdopodobieństwa zajścia danego zdarzenia oraz łagodzenia jego ewentualnych skutków.

7. Podsumowanie

W pracy zaproponowano koncepcję cyklu życia oprogramowania, który łączy krótkie „sprzężenie zwrotne" pomiędzy zespołem projektowym, a przedstawicielami klienta z podejścia XP, z koniecznością dokumentowania wymagań. Przedstawiono również trójpoziomowe podejście do planowania zakresu przedsięwzięć z wykorzystaniem metody delfickiej. Zaproponowana metoda planowania przedsięwzięć będzie wykorzystana w następnej edycji Studia Rozwoju Oprogramowania na Politechnice Poznańskiej.

Bibliografia

[1]. K. Beck. *Extreme Programming Explained.* Addison-Wesley, 2000.

[2]. B. Boehm. *Software Engineering Economics.* Prentice-Hall, 1981.

[3]. Carnegie Mellon University, Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley, 1994.

[4]. N.E. Fenton. *Software Metrics – A Rigorous Approach.* Chapman & Hall, 1991.

[5]. N.E. Fenton, S.L. Pfleeger. *Software Metrics – A Rigorous & Practical Approach.* International Thompson Computer Press, UK, 1996.

[6]. W.S. Humphrey. *A Discipline for Software Engineering.* Addison-Wesley, 1995.

[7]. W.S. Humphrey. *A Discipline for Software Engineering.* Addison-Wesley, 1989.

[8]. W.S. Humphrey. *Managing the Software Process.* Addison-Wesley, 1989.

[9]. M. Jasiński. *Delphi-based Effort Estimation for Software Projects.* Praca magisterska, Politechnika Poznańska, 2002 (w przygotowaniu).

[10].C. Jones. *Estimating Software Costs.* McGraw-Hill, New York, 1998.

[11].H.A. Linstone, M. Turoff. *The Delphi Method: Techniques and Applications.* Addison-Wesley, 1975.

[12].E.E. Mills. *Software Metrics – SEI Curriculum Module SEI-CM-12-1.1.* Carnegie Mellon University, 1988.

[13].R.S. Pressman. *A Manager's Guide to Software Engineering.* McGraw-Hill, 1993.

[14].R.S. Pressman. *Software Engineering – A Practitioner's Approach.* McGraw-Hill, Singapore, 1992.

[15].I. Sommerville. *Software Engineering.* Addison-Wesley, 1995.

[16].F. Woźniak. *Scenario-Based Requirements Engineering.* Praca magisterska, Politechnika Poznańska, 2000.