Poprawa procesów programowych w małej organizacji programistycznej

Grzegorz Wieczerzak
Politechnika Poznańska

Streszczenie

Program poprawy procesów programistycznych nie jest domeną dużych korporacji ani organizacji administracyjno-rządowych. Uniwersalna definicja CMM(I) sprawia, że często po wyższy poziom dojrzałości sięgają kolejne firmy programistyczne na świecie. Kwestią czasu jest wzrost zapotrzebowania w Polsce na systemy informatyczne wytwarzane w warunkach zapewniających jakość samego produktu, jak też atrakcyjność oferty. Niniejsze opracowanie próbuje wykazać słuszność wdrażania programów poprawy w małych organizacjach programistycznych na podstawie działań prowadzonych w Dziale Oprogramowania jednej z firm informatycznych oraz tendencji panujących na świecie.

1. Wgląd w procesy wytwarzania oprogramowania

Celem programu poprawy procesów w dowolnej organizacji programistycznej jest usprawnienie metod zarządzania i uzyskanie lepszej kontroli na poszczególnych etapach wytwarzania oprogramowania.

Wyobraźmy sobie organizację, w której istnieją zespoły programistów bezpośrednio zaangażowane w tworzenie nowego systemu informatycznego. Rozwój projektu nadzorują kierownicy poszczególnych zespołów. Z kolei ci ostatni podlegają kierownictwu wyższego szczebla.

Im dana osoba odpowiada za wyższy poziom zarządzania projektem, tym bardziej syntetyczny obraz całego systemu powinna posiadać. Kierownictwo odpowiada bowiem za istotne aspekty procesu produkcyjnego, jakimi są budżet oraz harmonogram. Jeśli zapewni się właściwą kulturę na poziomie zarządzania

produkcją oprogramowania, rozwój systemu informatycznego stanie się bardziej przewidywalny. Oprócz tego, uzbrojeni w realne dane i statystyki, będziemy mogli na życzenie klienta prezentować osiągnięcia w ustalonych terminach, a jeśli zajdzie potrzeba modyfikacji lub poprawy opracowywanego systemu, będziemy w stanie reagować na bieżąco.

W pięciostopniowym modelu dojrzałości CMM (Capability Maturity Model) organizacja programistyczna ma zdefiniowane obszary w ramach produkcji oprogramowania, które potrafi kontrolować i monitorować. Już na poziomie 2 CMM posiada wdrożone mechanizmy obsługi wymagań napływających od klienta. Poprawa metod zarządzania prowadzi do poznania procesów, które wspomagają lub utrudniają osiągnięcie pożądanych efektów pracy. Proste wydzielenie etapów oraz kamieni milowych znacząco ułatwia podejmowanie dalszych decyzji. Przestaje w pewnym momencie działać reguła 90-90, czyli opieranie się wyłącznie na intuicji, a mianowicie, że skoro przypuszczamy, iż nasz system pochłonął już 90% planowanego czasu lub innych zasobów, jest zrealizowany w 90%. Idąc dalej możemy wyobrazić sobie sytuację, wpatrując się w firmę będącą na 4 czy 5 poziomie CMM, że sterujemy procesem wytwarzania w sposób ilościowy. Baza danych towarzysząca od początku programu poprawy procesów zdążyła zapełnić się danymi które dla każdego kierownika czy dyrektora działu jest cennym źródłem przy podejmowaniu strategicznych decyzji. Na razie trudno w warunkach rodzimego przemysłu produkcji oprogramowania mówić o zoptymalizowanych liniach produkcyjnych (przyp. autora), gdyż oficjalnie nie istnieją na naszym rynku organizacje na najwyższym poziomie CMM, z drugiej jednak strony nie wydaje się, aby prezentowany przez zachodnioeuropejskie czy amerykańskie firmy (nie zapominajmy też, że duży rozwój obserwuje się na Wschodzie, czyli rynku azjatyckim, np. w Indiach) był przesadzony bądź utopijny. Problem zaczyna się od odpowiedzi na pytanie: Co z jakością naszych produktów? Jak możemy poprawić nasz wizerunek w oczach naszych klientów?

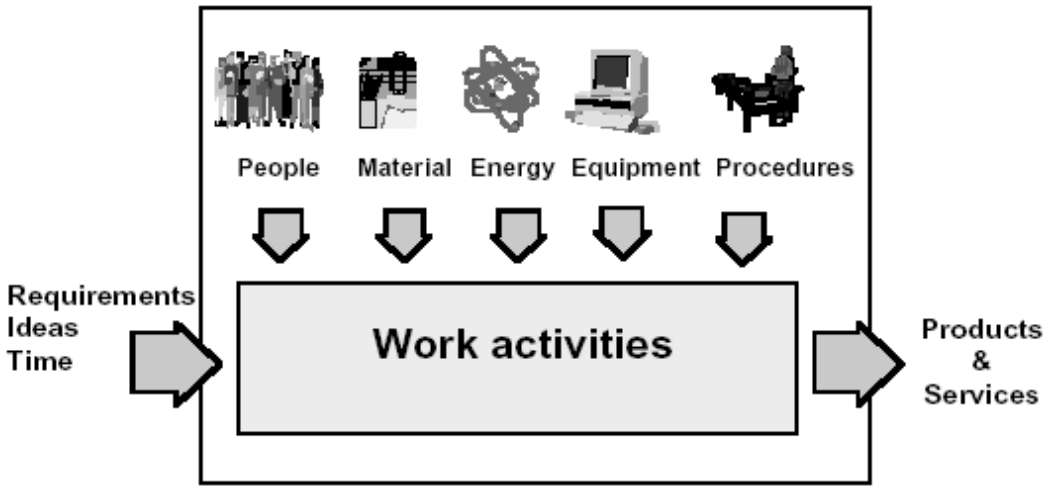
Nie można odpowiedzieć na powyższe pytania, dopóki nie zrozumiemy istoty procesów programowych w życiu organizacji programistycznej.

1.1. Model procesu

Model dojrzałości wytwórczej opracowany w instytucie Software Engineering Institute (SEI) nie jest gotową receptą na sukces organizacji programistycznej. Chociaż znacznie ułatwia określenie drogi rozwoju, a towarzyszące standardowi CMM dokumenty i raporty wspomagają definiowanie procedur [7], to jednak część pracy, jaka stoi przed firmą.

Jeśli nie znajdą się odpowiednie bodźce do zmian, jeśli zgadzamy się ze status quo, jakie towarzyszy nam w codziennej pracy programisty, projektanta czy

kierownika działu, to żadna próba wprowadzenia reguł nie ma szans powodzenia. Proces bowiem to nie tylko martwa recepta na przeprowadzenie danej operacji. To żywy instrument wykorzystywany każdego dnia w pracy przed komputerem czy podczas negocjacji z klientem. Oczywiście proces jako obowiązująca zasada powinien być udokumentowany, ponieważ czasami okazuje się potrzebne dokonanie jego korekty bądź odrzucenia. Dzięki takiej selekcji wdraża się jak najbardziej pożyteczne procedury.



Rys. 1 Zasada funkcjonowania procesu [6].

Proces jest to sekwencja kroków wykonywanych w określonym celu. Proces integruje ze sobą ludzi, narzędzia, energię, procedury i wiele innych czynników charakterystycznych dla danego środowiska pracy (rys. 1). Dla prostej organizacji programistycznej model możemy uprościć do ludzi, komputerów, oprogramowania oraz norm obowiązujących w danej firmie. Procesy programowe, czyli związane z produkcją oprogramowania mogą być zdefiniowane jako zestaw czynności, metod, praktyk i transformacji, które są stosowane przez określone zespoły lub jednostki, w celu wytworzenia i pielęgnowania oprogramowania i produktów pośrednich, takich jak plany projektu, dokumentacja architektury systemu i wymagań, kod źródłowy, przypadki testowe, dokumentacja użytkownika, etc. Integracja elementów jest niezbędna, gdy stoi przed nami zadanie napisania modułu realizującego pewną funkcję złożonego systemu informatycznego. Przypuśćmy, że jedno ogniwo źle funkcjonuje. Mogą pojawić się problemy w komunikacji między zespołami, szczególnie na etapach projektu powiązanych ze sobą. Może okazać się, że nie potrafimy wykorzystać wszystkich możliwości narzędzi na które zakupiliśmy drogie licencje. Bardzo często problemem staje się nieuporządkowana komunikacja z klientem stawiająca część zespołu przed koniecznością „ugaszenia nagłego pożaru", np. zgłaszane usterki, dodatkowe wymagania. Wówczas należy

zrewidować istniejące metody postępowania w takiej sytuacji i zaproponować lepsze, skuteczniejsze rozwiązanie.

Problem pojawia się, gdy istnieje lęk przed uświadomieniem sobie, że pewne procesy nie działają, są nieskuteczne, nie prowadzą do zamierzonych efektów. To jest właśnie ta część pracy, jaką kierownictwo wraz z inżynierami oprogramowania powinni wykonać w programie poprawy procesów – przezwyciężenie strachu przed zmianą istniejącego stanu rzeczy.

Procesy programistyczne prawidłowo wdrożone powinny dawać przewidywane rezultaty. Jeśli zauważy się rażącą niezgodność lub nieprzydatność procesu, należy podjąć kroki, które usprawnią działanie organizacji. Takie decyzje niekiedy musi podejmować wyższe kierownictwo, szczególnie jeśli w grę wchodzi zakup nowego oprogramowania albo dodatkowych licencji.

Przy ustalaniu procedur należy pamiętać, by nie odgradzać się od klienta. Niekiedy taka sztuczna tama stwarza organizacji wizerunek firmy „nieelastycznej", nie idącej na kompromis. Mimo, że zagadnienie współpracy z klientem wykracza poza zakres programu poprawy procesów, jednak należy się zastanowić czy popularne ostatnio inicjatywy CRM nie są już wdrażane w naszej organizacji. Jeśli tak, należy rozważyć współpracę działu wytwarzającego oprogramowanie z działem odpowiedzialnym za marketing i współpracę z klientem. Synergii można szukać więcej w firmie. Warto też mieć świadomość istnienia zagrożeń ze strony niektórych inicjatyw dla programu poprawy.

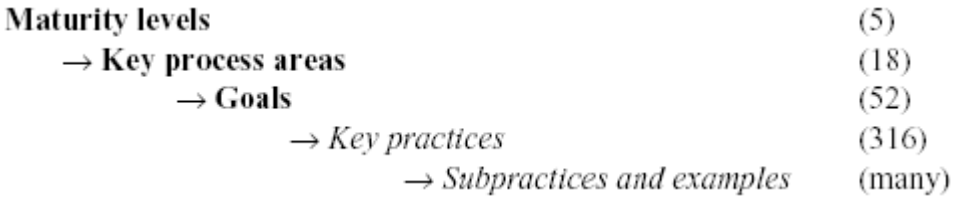
1.2. Powtarzalność procesów

Rozwój firmy informatycznej jest w dzisiejszym świecie nieunikniony. Coraz częściej obserwuje się potrzebę podkreślenia jakości swoich produktów. Brak profesjonalizmu w działaniu może być nieprzewidywalny, łącznie z utratą klientów.

To klienci decydują o kształcie rynku. Dom, który został zbudowany w trzy tygodnie, może budzić obawy w potencjalnym lokatorze. Dlatego budownictwo objęte jest rygorystycznymi przepisami i normami. Według określonych standardów architekt zaczyna od szkicu obiektu, który następnie stopniowo uszczegółowia, mając na względzie życzenia i wymagania klienta, znając wymiary działki oraz zasobność portfela nabywcy.

Gotowy plan jest końcowym produktem procesu projektowania. Z tym planem możemy zrobić wiele rzeczy. Udać się z nim do firmy budowlanej, lub schować go do szuflady dla swoich dzieci. Ponadto, jeśli pomysł zbudowania domu spodoba się naszym znajomym, cały proces powtórzy się według podobnych zasad.

Hierarchiczna struktura modelu dojrzałości wytwórczej (ang. *Capability Maturity* Model, CMM), ułożona w kluczowe obszary procesów, które definiują cele, zwyczaje (ang. *key practices*) i praktyki podrzędne (ang. *subpractices*), jest łatwa w interpretacji i pozwala na dostosowanie modelu dojrzałości do potrzeb organizacji (rys. 2).



Rys. 2 Hierarchiczna struktura CMM v. 1.1 [1].

Model dojrzałości wytwarzania oprogramowania (CMM-SW) definiuje pięć poziomów które są wyznacznikiem dojrzałości danej organizacji programistycznej. Poczynając od poziomu pierwszego, początkowego, który charakteryzuje się brakiem procesów i chaotycznym działaniem przy poszczególnych etapach przedsięwzięcia, przechodzimy na poziom powtarzalny, ukierunkowany na procesy związane z zarządzaniem projektami. Na poziomie drugim poszczególne czynności wykonywane są w sposób podobny. Pojawia się w związku z tym szansa na powtórzenie sukcesu przy realizacji projektu o podobnym charakterze.

2. Zapewnienie ciągłej poprawy

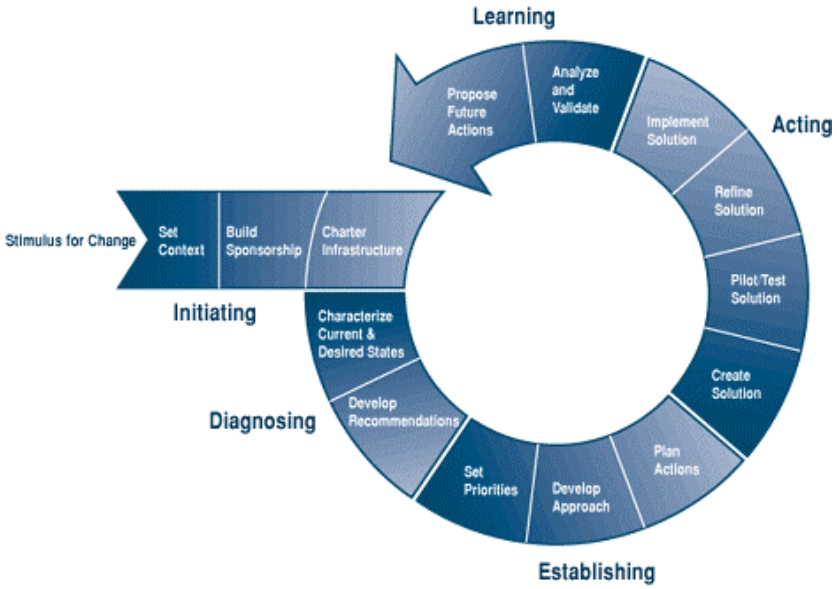
Istotą programu poprawy jest osiągnięcie większej wydajności, zmniejszenie liczby usterek w systemie, trafniejsze zarządzanie budżetem projektu, stworzenie lepszych warunków na osobisty rozwój pracowników – jednym słowem doskonalenie działania organizacji programistycznej wszędzie tam, gdzie dostrzegamy niezadowolenie z istniejącym stanem. Jest to swego rodzaju dążenie do doskonałości.

Aby zapewnić trwałość programu, bo tylko wtedy on ma sens, należy traktować jego wdrażanie na równi z innymi przedsięwzięciami mającymi miejsce w firmie. Bardzo ważne jest działanie przynoszące widoczne efekty. Oczywiście na początku nie są one tak oczywiste jakby można się było spodziewać. Jednakże która z organizacji nie podejmowała się szczytnej idei, a po kilku spotkaniach cała energia po prostu zanikła? Dlatego ważne jest zaczynanie poprawy od znalezienia jak najszerszego zbioru istotnych kwestii i stworzenie odpowiedniej infrastruktury. Działania nie mogą rozwiązać od razu wszystkich założeń, gdyż w przeciwnym wypadku łatwo rozproszyć siły na sprawach nie aż tak istotnych.

CMM może być skutecznym narzędziem przy prowadzeniu poprawy procesów programowych. Powinno być używane w kontekście systematycznego podejścia. Odpowiednim i uniwersalnym modelem wydaje się być IDEAL (od nazwy pięciu etapów następujących po sobie: *Initiating*, *Diagnosing*, *Establishing*, *Acting*, *Learning*).

2.1. Cykliczne działanie

Poprawa oparta na modelu CMM jest najefektywniejszym rozwiązaniem, choć sam model nie powinien być traktowany jako metodologia postępowania czy planowania projektu poprawy. Instytut SEI (CMU) proponuje w tym celu systematyczne podejście do projektu poprawy, jakim jest IDEAL^{SM}. Model IDEAL (rys. 3) jest pięciofazowym cyklem wyznaczającym ścieżkę realizacji projektu SPI. Możemy go również interpretować jako model składający się z dwóch etapów: strategicznego, bazującego na potrzebach organizacji oraz taktycznego, w którym procesy są definiowane, modyfikowane i uruchamiane. Po zakończeniu obu faz, następuje etap pozyskiwania wiedzy *Learning*, podczas którego kierownictwo definiuje nowe cele i strategie.



Rys. 3 Model IDEAL [3].

2.2. Zarządzanie projektem SPI

Poprawa procesów programowych jest dla organizacji znaczącym przedsięwzięciem. Aby zapewnić koordynację wielu czynności podejmowanych podczas programu SPI należy stworzyć efektywną infrastrukturę. Dodatkowo, owa infrastruktura powinna w porę reagować na żądania wynikające z zaistniałej sytuacji, tzn. wspomagać zarządzanie organizacją w miarę rozwoju programu.

Realizacja projektu SPI jest dosyć skomplikowana, ponieważ dotyczy konkretnego środowiska. Oto najczęściej pojawiające się problemy które należy wziąć pod uwagę jeszcze przed rozpoczęciem systematycznego działania:

- Od czego zacząć?
- Jak to zorganizować?
- Jak zaplanować działania?
- Kto się tym ma zająć?
- Jak śledzić postęp?
- Jak pokierować rozwojem?

Zarządzanie projektem poprawy procesów ma kluczowe znaczenie dla jego rozwoju. W pewnym momencie działania wykroczą poza obszar kompetencji inicjatorów, a wtedy, bez wsparcia i motywacji wyższych szczebli kierownictwa, program poprawy należy przerwać.

3. CMM(I) w małej organizacji programistycznej

„All models are wrong; some models are useful."

George Box

Nadeszła pora aby rozwiać wątpliwości budzące się po przeczytaniu tytułu niniejszego opracowania. Jakie czynniki decydują o tym, że jedną firmę określa się mianem dużej? Czy w grę wchodzi rozmiar infrastruktury placówek, czy może ilość pracujących w nich ludzi? Według definicji ekonomicznych można wskazać kilku liderów na polskim rynku dających pracę w branży informatycznej kilkuset programistom. Nie jest jednak zamiarem autora prezentowanie rankingu spółek informatycznych na podstawie osiąganych obrotów. Kiedy zorientujemy się z przykładów publikowanych na stronach Internetu, jakie organizacje wdrożyły program SPI (ang. *Software Process Improvement*), gdy w nagłówkach *success stories* ujrzymy nazwy dużych światowych koncernów lub organizacji administracyjno-rządowych, zaczniemy zastanawiać się nad trafnością pomysłu wdrożenia procesów w naszej organizacji.

Model CMM(I) (CMM *Integrated*, 2000), jak zapewniają jego twórcy, nie został celowo stworzony na potrzeby konkretnego środowiska. Jest on jedynie

zestawem wskazówek i praktyk, które należy dostosować do potrzeb danej organizacji. Nie ma znaczenia, w jaki sposób zespół dochodzi do pewnego celu, gdyż być może w naszych realiach nie dysponujemy wystarczającym budżetem na wprowadzenie pewnych metodologii. Co więcej, wdrożenie projektu o charakterze inwestycji długoterminowej może przysparzać różnym organizacjom odmienne trudności.

Ciekawostką jest jednak to, że zarówno same raporty źródłowe powstające na Uniwersytecie Carnegie Mellon, jak też *success stories* są ogólnodostępne. Aby zrozumieć dążenie do rozpowszechniania na szeroką skalę osiągnięć innych organizacji, należy wrócić do początków kryzysu systemów informatycznych kilkadziesiąt lat temu. Nie do pominięcia jest też promowanie własnego wizerunku firmy programistycznej i jakości produktów końcowych.

Skoro CMM(I) można wdrożyć w działach oprogramowania zatrudniających kilkanaście do kilkudziesięciu programistów, a znaczenie „mała organizacja" może być rozumiana na wiele sposobów [1], rozpatrzmy czynniki, które najczęściej decydują o sukcesie programu poprawy procesów tak w małych jak i dużych firmach programistycznych. Sukcesem programu SPI jest jego poprawność i skuteczność.

3.1. Infrastruktura

Wiele programów poprawy zakończyło się tylko na dobrych intencjach. Poprawa jest następstwem planowania czynności, przydzielania zasobów do zadań i ciągłej kontroli programu. Czasem wskazane jest wydzielenie zespołów odpowiedzialnych za kontrolę nad przebiegiem programu (SEPG, MSG), grup odkrywczych (ang. *discovery teams*) [3], zespoły do zarządzania zmianami. Ostatecznie jednak należy się kierować właściwymi proporcjami w liczebności zespołów. Należy stworzyć warunki, aby wdrażane procesy były skuteczne i pożyteczne. Poprawnie działająca infrastruktura powinna zapewniać [2]:

- **dokumentowanie** procesów prostym, jednoznacznym językiem;
- **szkolenie** w zakresie danego procesu odpowiednich osób;
- **stosowanie** procesów podczas codziennych czynności związanych z produkcją oprogramowania, nawet w chwilach kryzysu;
- **mierzenie** procesów za pomocą prostych miar;
- **utrzymywanie** i **wspieranie** procesów przez wyznaczone osoby;
- **kontrolowanie** zmian procesów;
- **weryfikowanie** i **akceptowanie**.

3.2. Cele zbierania danych

Wszelka analiza, którą przeprowadzamy na danych, powinna być użyteczna. Aby osiągnąć zamierzony rezultat, należy zrozumieć, jakie informacje są nam przekazywane i w jaki sposób można je interpretować. Proces definiowania miar rozpoczyna się od zebrania prostego zbioru użytecznych danych [8]. Następnie wprowadza się standardowe definicje dla typowych miar, co pozwala na agregowanie i porównywanie ich wartości z różnych środowisk.

Do interpretacji miar należy podchodzić z dużą czujnością. Zła interpretacja danych, może przyczynić się do obrania niewłaściwej drogi poprawy. Kierownictwo wyznaczając pewne cele i kierunki, tym samym przekazuje pracownikom do wiadomości, co jest w danej chwili najważniejsze. Właściwe zarządzanie i kierowanie dowolnym przedsięwzięciem, powinno opierać się na miarach. Dzięki temu można uniknąć sytuacji, w których skupiamy cały wysiłek na harmonogramowaniu, spychając jakość na drugi plan. Miary określają, co jest ważne. Oczywiście nie wszystko można w prosty sposób śledzić. I właśnie na te elementy trzeba zwracać szczególną uwagę.

3.3. Wsparcie kadry kierowniczej

Wsparcie kierownictwa polega na zdefiniowaniu misji, polityki działania oraz dostarczaniu zasobów niezbędnych do realizacji procesów programowych [2]. Nieocenione także jest przekonanie i wiara w strategiczne znaczenie poprawy procesów. Wskazane jest aby szczeble kierownicze firmy okazywały poparcie dla działań zespołów zaangażowanych w projekt oraz zwracały uwagę i wyrażały zainteresowanie zachodzącymi przemianami.

Dodatkowym czynnikiem motywującym wszystkich pracowników, jest podkreślanie przez zarząd znaczenia jakości dla firmy i jej klientów. Okazją do tego mogą być wspólne spotkania świąteczne, noworoczne, wewnętrzne konferencje w obecności przedstawicieli klientów, itp. Dzięki otwartej postawie, kierownictwo zyskuje respekt w oczach pracowników, a wszelkie kryzysy spowodowane skróceniem terminu realizacji projektu, zmniejszeniem budżetu, pogorszeniem warunków pracy, które mają prawo zdarzyć się w każdej organizacji z różnych przyczyn, nie będą odbierane wyłącznie jako problemy zarządu.

Wspieranie projektu poprawy w perspektywie długoterminowej powinno iść w parze z celami ekonomicznymi organizacji. Program doskonalenia zdolności procesów powinien mieć wysoki priorytet. Dzięki temu kadra kierownicza będzie w stanie monitorować ciągłe, systematyczne zmiany. Zdobywanie wyższych poziomów dojrzałości jest działaniem przyrostowym i wymaga długoterminowego zaangażowania. Ważna jest umiejętność stawiania jasnych

celów i prowadzenia proaktywnej polityki. Do poprawy procesów powinno podchodzić się jak do normalnego projektu. Przygotować plan, scharakteryzować kamienie milowe, wyznaczyć miary postępu.

Bibliografia

[1]. M.C. Paulk. *Using the Software CMM® in Small Organizations*. Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, 1998.

[2]. M.C. Paulk. *Effective CMM-Based Process Improvement*. Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, 1996.

[3]. B. McFeeley. *IDEAL^{SM}: A User's Guide for Software Process Improvement,* (CMU/SEI-96-HB-001). Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, February 1996.

[4]. M.C. Paulk, B. Curtis, M.B. Chrissis, C.V. Weber. *Capability Maturity Model^{SM} for Software, Version 1.1,* (CMU/SEI-93-TR-024). Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, February 1993.

[5]. M.C. Paulk, C.V. Weber, S.M. Garcia, M.B. Chrissis, M. Bush. *Key practices of the Capability Maturity Model^{SM}, Version 1.1,* (CMU/SEI-93-TR-025). Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, February 1993.

[6]. W.A. Florac, R.E. Park, A.D. Carleton. *Practical Software Measurment: Measuring for Process Management and Improvement,* (CMU/SEI-97-HB-003). Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, April 1997.

[7]. T.G. Olson, N.R. Reizer, J.W. Over. *A Software Process Framework for the SEI Capability Maturity Model^{sm},* (CMU/SEI-94-HB-01). Pittsburg, PA: Software Engineering Institute, Carnegie Mellon University, September 1994.

[8]. J. Nawrocki. *Pomiary procesów programistycznych* (prezentacja). Politechnika Poznańska, 2001.