

Zarządzanie jakością przy produkcji oprogramowania, czyli jakość to będzie (w następnym wydaniu)

Andrzej Sołowiej
ASSECO POLSKA S.A.

e-mail: andrzej.solowiej@assecopl

Abstrakt. Do napisania tego referatu skłoniło mnie zaobserwowane zjawisko, z którym wielokrotnie zetknąłem się uczestnicząc w projektach informatycznych. Okazuje się, iż pomimo, że mamy do dyspozycji coraz lepsze komputery i oprogramowanie, pomimo, że wśród nas jest coraz więcej, coraz lepiej wykształconych i coraz bardziej doświadczonych informatyków, pomimo, że kierują nami coraz lepsi, coraz bardziej doświadczeni kierownicy, i w dodatku wszyscy chcemy dobrze – to tak często wychodzi nam „jak zwykle”. Od dłuższego czasu zastanawia mnie, dlaczego tak się dzieje. Dlaczego tak często dochodzi do sytuacji, w której klient zamawiający wdrożenie systemu informatycznego, za olbrzymie pieniądze otrzymuje oprogramowanie które nie spełnia jego wymagań – a więc niskiej jakości? Czyżbyśmy nie dbali o jakość naszych produktów? Przecież się staramy. A może źle się staramy i dlatego nam nie wychodzi? Stwierdzenie, że produkowanie oprogramowania stałe dobrej jakości wymaga olbrzymiej pracy i zaangażowania ze strony informatyków różnych specjalności, jest trywialne. W swoim referacie, opierając się na literaturze przedmiotu, obserwacjach własnych i wymianie poglądów z kolegami po fachu, próbuję ukazać, jakie są główne przyczyny tego zjawiska oraz jakie działania mogą być podjęte w celu podniesienia jakości produkowanego oprogramowania. Główną tezę mojego referatu jest, że produkowanie oprogramowania stałe dobrej jakości jest sprawą zbyt poważną, aby pozostawić ją informatykom. Jest to wyzwanie, które wymaga podejmowania właściwych działań na wszystkich poziomach zarządzania organizacją, w tym także, a może przede wszystkim, na poziomie strategicznym. Konieczne są właściwe i konsekwentnie realizowane działania ze strony najwyższego kierownictwa, a brak takich działań może prowadzić do sytuacji, w której wysiłek informatyków i kierowników operacyjnych będzie iść na marne.

Informacje o autorze: Autor od prawie dziesięciu lat jest pracownikiem firmy ASSECO POLAND S.A. (dawny SOFTBANK S.A.). W swojej pracy zawodowej zajmuje się produkcją oprogramowania. Pełniąc różne role – programisty, projektanta, architekta, analityka, kierownika – uczestniczył w wielu projektach informatycznych, a że większość z nich była realizowana przy użyciu technologii i narzędzi Oracle stąd jego związki z PLOUG. Jako zarządzany i zarządzający miał okazję osobiście przekonać się, czym jest praca w zespołach projektowych i jakie wyzwania niesie ze sobą bycie włączonym i zaangażowanym (*involved and committed*) w projekty. Od 2006 roku jest uczestnikiem Międzynarodowego Studium Doktoranckiego w zakresie nauki o zarządzaniu organizowanego przez Instytut Organizacji i Zarządzania w Przemśle ORGMASZ w Warszawie.

1. Wstęp

Gdybyśmy mieli własnymi słowami, bez odwoływania się do naukowych definicji, określić, co decyduje o tym, że jakiś produkt oceniamy jako „dobrej jakości”, prawdopodobnie w pierwszej kolejności stwierdzilibyśmy, że musi to być produkt w pełni sprawny, nie może być zepsuty ani nie może posiadać wad produkcyjnych. A na podstawie jakich kryteriów mielibyśmy ocenić, że dany produkt jest „w pełni sprawny”? Prawdopodobnie pierwszą odpowiedzią, która by się nam nasunęła byłoby, że produkt powinien być zgodny ze specyfikacją, a więc posiadać te cechy, które producent zadeklarował, że posiada. Jednakże, czy zgodność ze specyfikacją i brak uszkodzeń wystarczą do stwierdzenia, że dany produkt jest wysokiej jakości? Niekoniecznie. Przede wszystkim oczekiwilibyśmy, że produkt, czy też usługa, będzie spełniać nasze wymagania.

Właśnie ten aspekt, stopień spełnienia przez produkt lub usługę wymagań użytkowników jest przyjmowany jako miara jakości. Norma PN ISO 9000:2006 definiuje, że „jakość” jest to „stopień, w jakim zbiór inherentnych właściwości spełnia wymagania” [Iso9000 s. 25]. Norma wyjaśnia, że „inherentny», w przeciwieństwie do »przypisany«, oznacza tkwiący w istocie czegoś, szczególnie jako stała właściwość” [Iso9000 s. 25], natomiast „wymaganie” jest to „potrzeba lub oczekiwanie, które zostało ustalone, przyjęte zwyczajowo lub jest obowiązujące” [Iso9000 s. 25].

Aby dostawcy byli w stanie dostarczać produkty dobrej jakości, muszą na każdym poziomie funkcjonowania przedsiębiorstwa podejmować działania na rzecz jakości. Takie „skoordynowane działania dotyczące kierowania organizacją i jej nadzorowania w odniesieniu do jakości” norma PN ISO 9000:2006 nazywa „zarządzaniem jakością” [Iso9000 s.29]. W referacie spróbuję przedstawić, w jaki sposób działania składające się na zarządzanie jakością mogą być realizowane przy produkcji oprogramowania.

W swoich rozważaniach skoncentruję się na tych elementach zarządzania jakością przy produkcji oprogramowania, które związane są z organizacją procesu wytwórczego, z położeniem szczególnego nacisku na zadania najwyższego kierownictwa. Z premedytacją pomijam te elementy zarządzania jakością, które wiążą się z technologiczną stroną produkcji oprogramowania, takie jak organizacja testowania i poprawiania błędów. Organizatorzy konferencji uświadomili mi, że na te tematy większość odbiorców wie znacznie więcej ode mnie i nie ma co odbiorców denerwować mówieniem o rzeczach dla Państwa oczywistych.

2. Podejście procesowe do zarządzania jakością

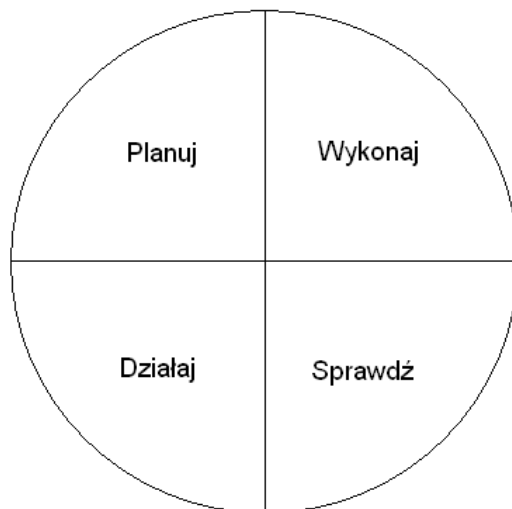
Wspomniana już norma ISO 9000 zaleca aby w celu zwiększenia stopnia spełnienia wymagań klienta, przy wdrażaniu systemu zarządzania jakością stosować podejście procesowe. Według niej, „organizacje, aby funkcjonować skutecznie, powinny identyfikować i zarządzać licznymi wzajemnie powiązаныmi i wzajemnie oddziałującymi procesami” zachodzącymi w organizacjach [Iso9000 s. 13].

W zarządzaniu jakością do podejścia procesowego norma PN ISO 9001:2001 zaleca stosowanie metody znanej jako „Planuj-Wykonaj-Sprawdź-Działaj” (PDCA z ang. Plan-Do-Check-Act). Poszczególne etapy tej metody są opisane następująco:

- „Planuj – ustal cele i procesy niezbędne do ustalenia wyników zgodnych z wymaganiami klienta i polityką organizacji.
- Wykonaj – wdróż procesy.
- Sprawdź – monitoruj i mierz procesy i wyrób w odniesieniu do polityki, celów i wymagań dotyczących wyrobu i przedstawiaj wyniki.

- Działaj – podejmij działania dotyczące ciągłego doskonalenia procesów.” [Iso9001 s. 13]

Metoda ta jest znana powszechnie i wielokrotnie opisywana w literaturze przedmiotu jako „koło Deminga”. [por. HaMa06 s. 93, KiSr05 s. 124] (patrz rysunek 1.)



Rys. 1. Koło Deminga [KiSr05 s. 25]

Z opisu metody PDCA oraz jej ukazania jako koło Deminga wynikają wnioski ważne dla zarządzania jakością:

- zarządzanie jakością musi być procesem iteracyjnym, wszelkie działania na rzecz jakości powinny być regularnie oceniane i doskonalone;
- wejściem dla procesu zarządzania jakością muszą być wymagania klienta, analogicznie – sprawdzanie skuteczności zarządzania jakością musi opierać się na monitorowaniu poziomu zadowolenia klienta, a więc konieczne jest zbieranie danych czy produkt, organizacja, proces wytwórczy spełniają wymagania klienta.

W dalszej części przedstawię, w jaki sposób powyższe wnioski mogą być uwzględnione w procesach produkcji oprogramowania.

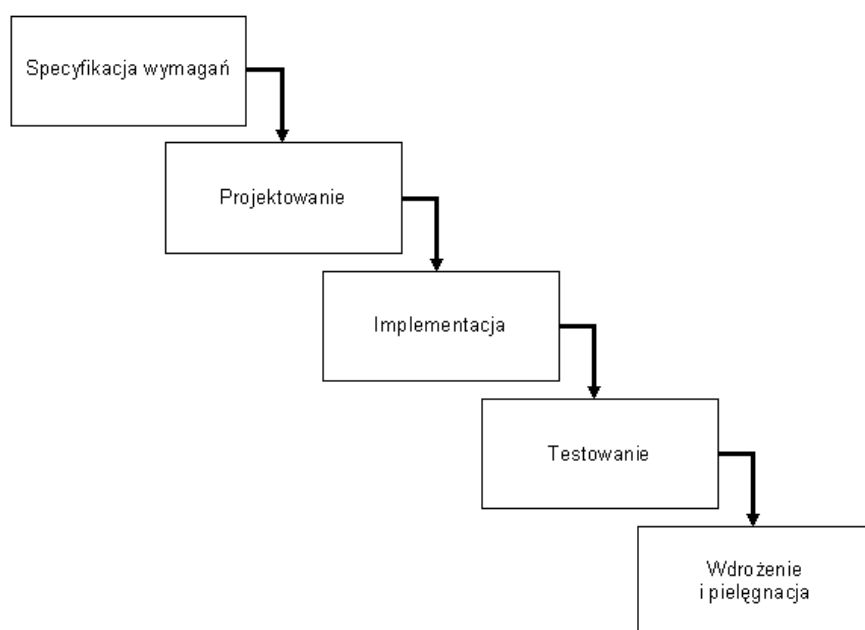
2.1. Proces wytwórczy na poziomie operacyjnym

W wyniku praktyki budowania i wdrażania systemów informacyjnych wykształcił się typowy podział procesu produkcji oprogramowania na etapy. Podział ten jest, z pewnymi odmianami, powszechnie wykorzystywany. Najczęściej spotykane są następujące fazy procesu produkcji oprogramowania:

- Specyfikacja wymagań – sformalizowanie wymagań, stawianych przez odbiorcę wobec budowanego systemu.
- Projektowanie – zdefiniowanie struktury budowanego systemu informacyjnego, tzn. architektury oprogramowania i sprzętu komputerowego, zaprojektowanie modułów systemu oraz połączeń (interfejsów) między nimi, podzielenie modułów na komponenty programowe (procedury, funkcje, metody) i ich zaprojektowanie, zaprojektowanie sposobu komunikowania się (interfejsów) budowanego systemu z innymi systemami.
- Implementacja – zaprogramowanie komponentów programowych w konkretnym języku oprogramowania a następnie ich połączenie.

- Testowanie – przetestowanie na różnym poziomie zarówno poszczególnych modułów systemu jak i całego systemu oraz usuwanie wykrytych błędów.
- Wdrożenie i pielęgnacja – przekazanie systemu do eksploatacji a następnie usuwanie usterek i błędów które mogą pojawić się w trakcie eksploatacji.

W najprostszej organizacji procesu wytwórczego oprogramowania zakłada się, że kolejne fazy następują po sobie w ten sposób, że kolejna rozpoczyna się po zakończeniu poprzedniej. Jednocześnie zakłada się, że nie ma możliwości powrotu do faz już zakończonych. Tak zorganizowany proces wytwórczy jest określany w literaturze przedmiotu jako „kaskadowy”. Kaskadowy model procesu produkcji oprogramowania jest przedstawiony na rysunku 2.



Rys. 2. Kaskadowy proces produkcji oprogramowania
Źródło: opracowane na podstawie [Szyj04 s. 25]

Kaskadowa konstrukcja procesu produkcji oprogramowania została opisana w wielu pozycjach literaturowych, do których odsyłam zainteresowanych (por. [Flas07, s. 69-70], [Górs99, ss. 34-35], [Kan06, s. 37-43] [Szyj04, s. 23-32]). Z punktu widzenia zarządzania jakością ważne jest, że niezależnie od tego, jaki model procesu wytwórczego jest wykorzystywany w konkretnym wdrożeniu, na poziomie operacyjnym prawie zawsze będzie to jakaś odmiana procesu kaskadowego.

W kontekście zarządzania jakością ważne jest, aby do procesu wytwórczego były włączone mechanizmy, które zapewnią, że oprogramowanie wyprodukowane na wyjściu procesu będzie zgodne z ustalonymi wymaganiami. Wdrożenie takich mechanizmów a następnie konsekwentne ich wykorzystywanie jest olbrzymim wyzwaniem. To, że jest to zadanie trudne, a walka o jakość nierówna i często kończąca się niepowodzeniem, wielu z nas miało okazję przekonać się osobiście.

Kiedy przyjrzymy się dokładniej kaskadowej organizacji procesu produkcji oprogramowania, zauważymy, że opiera się on na pewnych optymistycznych założeniach:

1. Analitycy są w stanie na początku procesu ustalić z odbiorcą i zatwierdzić specyfikację wymagań, która pozostanie niezmienną aż do fazy wdrożenia i która będzie podstawą do formalnego odbioru oprogramowania (i zapłaty).
2. Projektanci są w stanie zaprojektować przed rozpoczęciem fazy implementacji rozwiązanie, które będzie w pełni spełniać wymagania zawarte w specyfikacji.

Założenia te, na pierwszy rzut oka, wydają się rozsądne i, co więcej, realistyczne. W końcu od wielu tysięcy lat w taki właśnie sposób buduje domy, drogi i mosty. Niestety, okazuje się, że w realnym świecie oprogramowanie rzadko kiedy można produkować w ten sposób. Przyczyny, dlaczego tak się dzieje opisał Philippe Kruchten (por. [Kruc03] s. 53-59). W tym miejscu pokrótce przytoczę najważniejsze z nich.

W rzeczywistości analitycy tylko w naprawdę małych i krótkotrwałych projektach są w stanie na początku ustalić i zatwierdzić niezmienną specyfikację wymagań, a i to tylko wtedy, gdy obszar, który ma być informatyzowany jest dobrze zorganizowany i jest dobrze znany zarówno dostawcy oprogramowania jak i zamawiającym. Aby można było dobrze ustalić wymagania wobec systemu informacyjnego dodatkowym warunkiem jest, aby użytkownicy i analitycy potrafili rozmawiać tym samym językiem – czyli znać metody analizy wymagań. W większości przypadków warunki te nie są spełnione a to dlatego, że świat się zmienia: użytkownicy się zmieniają, problem się zmienia, technika informacyjna się zmienia oraz sposób prowadzenia biznesu przez odbiorcę oprogramowania się zmienia. Im dłuższy jest czas pomiędzy zatwierdzeniem wymagań a dostarczeniem gotowego oprogramowania do odbioru, tym te zmiany będą większe.

Weźmy na przykład sposób postrzegania oprogramowania przez użytkowników. Użytkownik będzie miał inne wyobrażenie o oprogramowaniu gdy się o nim rozmawia lub czyta, a inne, gdy je zobaczy i zacznie z niego korzystać. Najczęściej, gdy użytkownik zobaczy w jaki sposób jego wymagania zostały zaimplementowane, zmieni swoje wymagania. Poza sytuacjami wyjątkowymi, użytkownicy tak naprawdę nie wiedzą czego chcą, a nawet jeśli już wiedzą, to nie potrafią swoich oczekiwań jednoznacznie wyartykułować, natomiast, kiedy widzą gotowe oprogramowanie, są w stanie określić, co im się w nim nie podoba i czego od początku nie chcieli. Szczególnie, jeżeli czas trwania wdrożenia jest mierzony w miesiącach lub w latach, budowanie, na podstawie wymagań zamrożonych na początku procesu, prowadzi do zbudowania oprogramowania zgodnego z wymaganiami, ale bezużytecznego dla użytkowników.

Biorąc to pod uwagę, trzeba przyznać, że założenie, iż na początku procesu produkcyjnego analitycy będą w stanie zebrać ostateczne wymagania wobec oprogramowania jest kuszące, ale nierealne¹.

Co do projektantów, założenie, że będą oni w stanie zaprojektować przed rozpoczęciem fazy implementacji rozwiązanie, które będzie w pełni spełniać zawarte w specyfikacji wymagania zarówno funkcjonalne, wydajnościowe, ergonomiczne i estetyczne, może być spełnione, ale tylko w naprawdę wyjątkowych sytuacjach. Przede wszystkim oprogramowanie musiałoby być implementowane przy użyciu technologii, która jest dobrze poznana przez zespół wdrożeniowy, sprawdzona i stabilna. A i to prawdopodobnie nie byłoby warunkiem wystarczającym. Przede wszystkim dlatego, że, jak twierdzi Philippe Kruchten, „Inżynieria oprogramowania nie osiągnęła poziomu innych dyscyplin inżynierskich (i prawdopodobnie nigdy nie osiągnie) ponieważ będące jej podstawą »teorie« są słabe i źle rozumiane, a jej metody poznawania faktów poprzez stawianie hipotez i badania są niedojrzałe” [Kruc03 s. 57].

Ograniczenie wpływu wymienionych zagrożeń wymaga podjęcia odpowiednich działań na poziomie organizacji procesu wytwórczego oprogramowania. Skoro ryzyko złego zdefiniowania wymagań jest tym większe im dłuższy jest czas pomiędzy specyfikacją wymagań a dostarczeniem gotowego oprogramowania do wdrożenia, to naturalnym sposobem przeciwdziałania jest takie

¹ Tymczasem prawo o zamówieniach publicznych wymusza, aby wdrażanie systemów informacyjnych w administracji odbywało się za stałą, z góry określoną cenę oraz na podstawie z góry określonych wymagań (słynne SIWZy – Specyfikacja Istotnych Warunków Zamówienia). Można by przytoczyć kilka dużych wdrożeń informatycznych, które doprowadziły do sytuacji, że po zapłaceniu określonej ceny i po określonym czasie zostało wdrożone oprogramowanie ściśle spełniające SIWZ, ale zupełnie bezużyteczne. Natomiast doprowadzenie tego oprogramowania do stanu jako takiej użyteczności wiązało się z dodatkowymi, często olbrzymimi nakładami pracy, czasu i kosztów.

zorganizowanie procesu aby ten czas był jak najkrótszy. Skoro istnieje zagrożenie, że wymagania użytkowników będą się zmieniać i dopiero gdy użytkownicy otrzymają gotowe oprogramowanie, będą wiedzieć, co chcą, to trzeba do procesu wprowadzić mechanizm zarządzania zmianami wymagań. Jednym z możliwych sposobów jest:

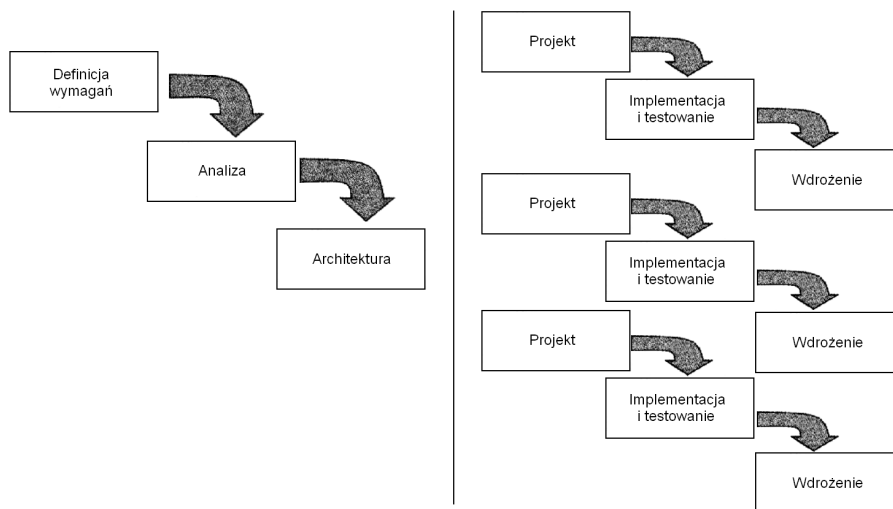
1. podzielenie budowanego oprogramowania na części – pod względem funkcjonalnym lub ze względu na wykorzystywaną technologię;
2. podzielenie wdrożenia na krótsze fazy, z których każda kończy się przekazaniem użytkownikom do oceny działającej części oprogramowania;
3. uwzględnienie zmieniających się wymagań przy rozpoczynaniu kolejnych faz wdrożenia.

Powyższe zalecenia stały się podstawą organizacji różnorodnych modeli procesów wytwórczych oprogramowania. W literaturze przedmiotu można znaleźć modele organizacji procesu wytwórczego oprogramowania, które w swojej konstrukcji uwzględniają wymienione wcześniej sugestie. Poniżej, pokrótce opiszę modele najczęściej spotykane: przyrostowy, spiralny, iteracyjny.

2.2. Przyrostowy model produkcji oprogramowania

Przyrostowy model produkcji oprogramowania dokładnie opisał Zdzisław Szyjewski (por. [Szyj04 ss. 36-39]), w tym miejscu przytoczę jedynie najważniejsze elementy tego modelu.

Schemat modelu przyrostowego jest przedstawia rysunku 3.

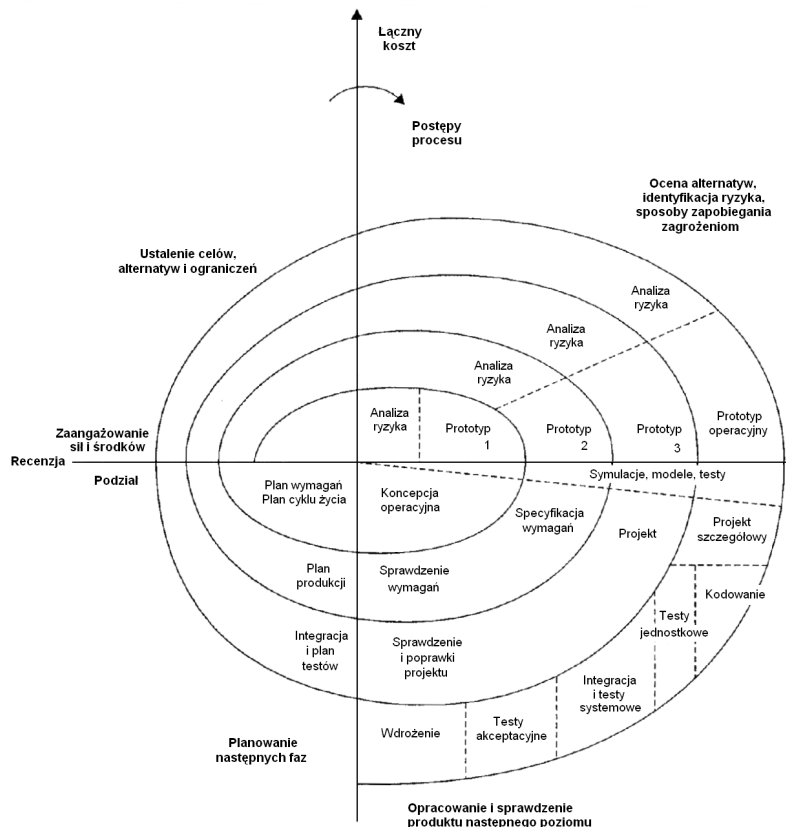


Rys. 3. Model przyrostowy produkcji oprogramowania
Źródło: [Szyj04 s. 37]

W modelu przyrostowym wdrożenie rozpoczyna się od przeprowadzenia analizy, której efektem jest opracowanie architektury dla całości rozwiązania. Natomiast faza realizacji jest dzielona na krótsze etapy, które są kolejno wykonywane. Każdy z tych etapów kończy się wdrożeniem części oprogramowania. Dzięki temu unika się wymienionych wcześniej zagrożeń i uzyskuje się korzyści wynikające z wcześniejszego przekazywania użytkownikowi części wdrażanego rozwiązania. Jak pisze Z. Szyjewski: „Wczesne wdrożenie niektórych elementów projektu pozwala na wcześniejsze uzyskanie korzyści z wprowadzanej zmiany oraz ma istotny wpływ edukacyjny dla użytkowników. Pozwala to również łatwiej akceptować sukcesywnie wprowadzane zmiany oraz elastycznie modyfikować, będące jeszcze w realizacji, elementy projektu. Kolejne, przyrostowo wdrażane moduły systemu informatycznego zaczynają wcześniej przynosić efekty wynikające z zastosowania informatyki” [Szyj04 s. 36].

2.3. Spiralny model produkcji oprogramowania

Spiralny model produkcji oprogramowania, stworzony przez B. W. Boehma w 1988 roku i wielokrotnie cytowany, jest przedstawiony na rysunku 4.



Rys. 4. Spiralny model produkcji oprogramowania

Źródło: [Kan06 s. 46]

Podstawą tego modelu jest założenie, że każdy etap produkcji zawiera tę samą sekwencję kroków. Każda faza produkcji obejmuje jeden cykl spirali. Pierwszym krokiem każdego cyklu jest określenie celów aktualnie budowanej części oprogramowania, alternatywnych sposobów wytworzenia oraz ograniczenia związane z wprowadzeniem tych alternatyw. Następnym krokiem jest analiza ryzyka, a więc identyfikacja możliwych zagrożeń i sposobów zapobiegania. Ważnym elementem modelu spiralnego jest budowanie prototypów. Dzięki zastosowaniu prototypowania unika się, wymienionego wcześniej, ryzyka wyprodukowania oprogramowania zgodnego ze specyfikacją, ale niezgodnego z rzeczywistymi potrzebami użytkownika. Dopiero po zaakceptowaniu prototypu rozpoczyna się właściwą produkcję z przejściem przez wszystkie fazy implementacji. Każdy cykl produkcji kończy się przeglądem, którego wyniki stają się materiałem wejściowym przy rozpoczynaniu kolejnego cyklu spirali.

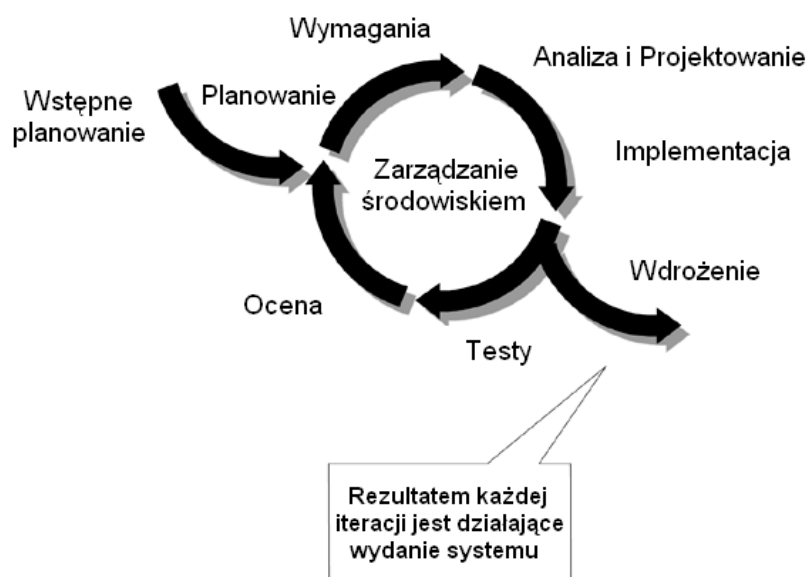
W opisie modelu spiralnego skoncentrowałem się na jego najważniejszych cechach. Model ten został dokładnie opisany w wielu pozycjach literaturowy, do których odsyłam osoby zainteresowane (por. [KiSr05 ss. 110-111], [Kan06 ss. 45-48], [Szy04 ss. 41-43], [Górs99 ss. 36-38]).

2.4. Iteracyjny model produkcji oprogramowania

Model iteracyjny można potraktować jako pewną syntezę modeli przyrostowego i spiralnego. Główną ideą tego modelu jest podzielenie procesu produkcyjnego na kilka iteracji. Każda iteracja składa się z kolejnych wykonywanych po kolei etapów: analizy, projektowania, implementacji,

testowania, a więc tak jak w modelu kaskadowym. Każda iteracja kończy się przekazaniem użytkownikowi kolejnego wydania oprogramowania. Uwagi użytkowników dotyczące oddanej części oprogramowania stają się materiałem wejściowym dla kolejnej iteracji.

Schemat iteracyjnego modelu produkcji oprogramowania jest przedstawiony na rysunku 5.



Rys. 5. Iteracyjny model produkcji oprogramowania
Źródło: [Kruc03 s. 9]

P. Kruchten wymienia szereg korzyści uzyskiwanych dzięki zastosowaniu modelu iteracyjnego. Najważniejsze z nich to (por. [Kruc03 s. 9]):

- Możliwość wczesnego wychwycenia niezrozumienia potrzeb lub wymagań użytkowników, dzięki czemu możliwe jest szybkie skorygowanie błędów analizy.
- Szybkie uzyskanie informacji zwrotnej od użytkowników, co umożliwia odkrycie ich rzeczywistych potrzeb i wymagań.
- Wczesne wykrywanie niezgodności pomiędzy wymaganiami, projektem a implementacją.
- Regularne dostawy oprogramowania dają użytkownikom widoczny dowód postępu prac.

Bodajże najdojrzałą metodyką opartą na modelu iteracyjnym jest Rational Unified Process rozwijany przez firmę IBM. Zainteresowanych tą metodyką oraz sposobem w jaki realizuje ona zasady modelu iteracyjnego odsyłam do cytowanej już książki [Kruc03].

3. Polityka jakości i doskonalenie jakości – zadania najwyższego kierownictwa

Zastosowanie, przez osoby odpowiedzialnie za organizację produkcji oprogramowania, wymienionych sposobów zorganizowania procesu wytwórczego może w skuteczny sposób wyeliminować wiele zagrożeń, szczególnie tych wynikających z niewłaściwej specyfikacji wymagań oraz

niewłaściwej implementacji, będących częstą przyczyną dostarczenia użytkownikowi oprogramowania złej jakości. Tyle, że nie zawsze jest tak się dzieje. Sugestie wynikające z dotychczasowych rozważań mogą zostać zastosowane tylko przy spełnieniu pewnych warunków. Tych warunków, które niedoświadczeni kierownicy projektów przeważnie uważają za oczywiste do tego stopnia, że nawet o nich nie wspominają podejmując się realizacji wdrożenia. Chodzi mi o takie, na przykład, oczekiwania, że osoby biorące udział w produkcji oprogramowania będą mieć odpowiednie kwalifikacje, tzn. programiści będą umieć programować w języku programowania, który jest wykorzystywany w danym wdrożeniu, projektanci będą umieć projektować, a analitycy będą umieć wykryć, zanalizować i jednoznacznie wyspecyfikować wymagania a w dodatku wszyscy będą odpowiednio zmotywowani do pracy.

Niestety, często okazuje się, że te, wydawało by się naturalne, wymagania wcale takie oczywiste nie są. Kierownik mający zapewnić wyprodukowanie oprogramowania dobrej jakości dostaje do dyspozycji zespół składający się z osób nieposiadających odpowiednich kwalifikacji, a w dodatku przedsięwzięcie, które nieopatrznie zobowiązał się poprowadzić jest niedoszacowane, za to ma nierealne terminy. Czemu dochodzi do takiej sytuacji? Z moich obserwacji wynika, że to zjawisko jest ściśle związane z wdrożeniem – lub raczej brakiem właściwego wdrożenia – elementów składowymi zarządzania jakością: polityką jakości i systemem doskonalenia jakości.

Zgodnie z definicją, polityka jakości jest to „ogół zamierzeń i ukierunkowanie organizacji dotyczące jakości, formalnie wyrażone przez najwyższe kierownictwo” [Iso9000 s. 27]. Natomiast doskonalenie jakości jest to „część zarządzania jakością ukierunkowana na zwiększenie zdolności do spełniania wymagań dotyczących jakości” [Iso9000 s. 27].

Oczywistym jest, że samo doskonalenie jakości jest zadaniem wszystkich uczestników produkcji oprogramowania, zarówno pracowników wykonawczych, jak również kierowników wszystkich szczebli. Uważam natomiast, że wdrożenie systemu doskonalenia jakości powinno być częścią polityki jakości, a przez to jednym z zadań najwyższego kierownictwa, a to dlatego, że doskonalenie jakości wiąże się z bardzo ważnym elementem zarządzania jakością – finansowaniem kosztów jakości.

3.1. Koszty jakości

Każdy, kto zetknął się z produkowaniem czegokolwiek, dobrze wie, że jakość kosztuje. Aby wyprodukować wyrób dobrej jakości trzeba poświęcić dużo pracy, czasu i środków na dobre zorganizowanie procesu wytwórczego, dobre zaprojektowanie wyrobu, dokładne wykonanie wyrobu, dokładną kontrolę oraz poprawianie błędów i usterek.

Według literatury przedmiotu, na całość kosztów jakości składają się następujące składniki:

- „koszty prewencji, generowane w związku z działalnością zapobiegającą powstawaniu negatywnych skutków w kształtowaniu jakości produktów,
- koszty oceny, ponoszone w procesach badania, kontroli, weryfikacji i oceny jakości,
- koszty braków wewnętrznych, powstające w związku z występowaniem i usuwaniem negatywnych skutków braków występujących w przedsiębiorstwie,
- koszty braków zewnętrznych, związane z występowaniem i usuwaniem negatywnych skutków braków wykrytych poza przedsiębiorstwem.

Niekiedy powyższy wykaz składników kosztów jakości rozszerza się o koszty zewnętrznego zapewnienia jakości oraz o tzw. utracone korzyści, które wyrażają umniejszenie przychodów i zysków wynikające z nieodpowiedniej jakości produktów.” [HaMa06 s. 154]

Przypisanie wymienionych kategorii kosztów do wspomnianych wcześniej działań związanych z jakością w produkcji oprogramowania wyglądałoby następująco:

- koszty prewencji – koszty organizacji procesu wytwórczego, a więc m.in. organizowanie i wdrożenie procedur, oraz szkolenie pracowników a także ich motywowanie,
- koszty oceny – koszty zorganizowania i przeprowadzenia testów oprogramowania,
- koszty braków wewnętrznych – koszty poprawiania błędów znalezionych w trakcie testowania,
- koszty braków zewnętrznych – koszty analizy i poprawiania błędów zgłoszonych przez użytkowników w trakcie eksploatacji,
- koszty zewnętrznego zapewnienia jakości – koszty ponoszone w związku ze wspieraniem utrzymania oprogramowania będącego w eksploatacji,
- koszty utraconych korzyści – trudne do wyliczenia koszty, będące efektem takich zjawisk jak, m.in., zła renoma firmy, brak wystarczającej ilości odpowiednich zasobów.

Dlaczego piszę o kosztach jakości w rozdziale poświęconym zadaniom najwyższego kierownictwa? Otóż dlatego, że na podstawie doświadczenia i obserwacji dochodzę do wniosku, iż niska jakość produkowanego oprogramowania bardzo często jest efektem braku strategicznego podejścia do jakości ze strony najwyższego kierownictwa przedsiębiorstw informatycznych. Jest to ściśle związane z faktem, że koszty jakości i zyski z jakości, z natury rzeczy, rozkładają się nierównomiernie w zależności od tego w jak długiej perspektywie czasu będziemy je oceniać. W krótkim okresie widać przede wszystkim koszty jakości, a korzyści z jakości uwidaczniają się w perspektywie strategicznej, a więc kilkuletniej.

Pracownicy wykonawczy oraz kierownicy działający na operacyjnym poziomie zarządzania oceniają podejmowane działania w perspektywie od miesiąca do maksymalnie trzech miesięcy, najczęściej w perspektywie otrzymywanych wynagrodzeń i ewentualnych premii za wykonanie zadań. Na poziomie taktycznym perspektywa ta, z punktu widzenia kierownika wdrożenia, wydłuża się – kierownik ocenia działania w perspektywie zakończenia najbliższej fazy wdrożenia. Z takiej perspektywy najwyraźniej widać koszty jakości, mianowicie to, że dokładniejsze i ogólnie lepsze zaprojektowanie, implementacja, przetestowanie i poprawianie błędów wymagałyby: (1) zaangażowania większego zespołu, (2) zatrudnienia lepiej wykształconych, bystrzejszych pracowników, których następnie trzeba by bardziej zmotywować, (3) zarezerwowania dłuższego czasu oraz (4) użycia odpowiedniego oprogramowania narzędziowego wspierającego wytwarzanie i testowanie oprogramowania. To wszystko kosztuje i obciąża bieżący budżet wdrożenia. Oczywiście brak jakości też kosztuje, znacznie więcej niż koszty jakości, tyle, że koszty braku jakości, zresztą tak samo jak korzyści z jakości, są odsunięte w czasie, a koszty jakości trzeba ponosić na bieżąco.

Jest jeszcze jeden aspekt sprawy. Teoria podaje, że przy produkcji oprogramowania główne znaczenie mają trzy czynniki: czas, koszt (budżet) i zakres. Czynniki te są ze sobą powiązane i zmiana każdego z nich wpływa na konieczność zmiany co najmniej jednego z pozostałych, na przykład zwiększenie zakresu spowoduje konieczność zwiększenia budżetu lub wydłużenia czasu produkcji. Tymczasem, w realnym świecie, bardzo często termin oddania produktu, całkowity budżet oraz skład zespołu są narzucone i ani kierownik wdrożenia ani tym bardziej kierownicy operacyjni nie mają na nie wpływu. Co zatem może zrobić kierownik odpowiedzialny za produkcję oprogramowania kiedy termin jest nierealny, budżet niedoszacowany za to zespół zbyt mały i składający się osób nieposiadających odpowiednich kwalifikacji? Skoro nie można sterować ani czasem, ani kosztem, ani zakresem, to często steruje się, jawnie lub nie, czwartym czynnikiem – jakością.

Jeżeli zespół ma w określonym czasie zbudować oprogramowanie posiadające określoną funkcjonalność, a już ze wstępnych szacowań wynika, że nie jest to możliwe, to naturalną tendencją będzie oszczędzanie na tych działaniach, których efekt jest widoczny jak najpóźniej. W praktyce oznacza to realizację z wdrożenia zgodnie z powszechnie stosowaną metodyką F&D, a więc „Fast and Dirty”. Sprowadza się to ograniczania czasu i zakresu testowania oraz poprawiania tylko błędów.

dów spektakularnych, za to łatwych do szybkiego poprawienia. Działanie takie wynika z pozornie racjonalnej kalkulacji. Skoro działania na rzecz wysokiej jakości kosztują, a ich wynik w postaci oprogramowania dobrej jakości będzie widoczny dopiero za jakiś czas, to jeśli zrezygnujemy z tych działań, osiągniemy widoczną oszczędność, a brak jakości uwidoczni się dopiero kiedyś. Pensje i premie programistów i kierowników operacyjnych zależą od wykonania zadań. Jeżeli programista staje przed wyborem: nie dam rady wykonać tego zadania dobrze, więc, albo nie wykonam go wcale, albo wykonam je byle jak, to wybór jest prosty. Jeżeli kierownik zespołu programistów staje przed wyborem: tego się nie da zrobić dobrze w tym czasie, więc, albo nie oddamy tego systemu w ogóle, albo zrobimy go byle jak i oddamy go z błędami, to też nietrudno zgadnąć jaką decyzję podejmie. Wprawdzie wszyscy mają świadomość, że błędy w oddanym oprogramowaniu wyjdą na jaw bardzo szybko, ale zakładają, że stanie się dopiero po tym jak premie za terminowe oddanie oprogramowania znajdą się już w ich kieszeniach.

Podejście takie bierze się także po części z powszechnego przekonania, że oprogramowanie można w każdej chwili poprawiać i łatwo wprowadzać w nim zmiany. Rodzi to pokusę, że jeśli nie ma czasu ani środków aby zrobić coś dobrze od razu, to zrobimy to teraz jak się da, a później to poprawimy. Wprawdzie doświadczenie uczy, że zmiany tylko w pewnym zakresie można wprowadzać łatwo i w miarę tanio, natomiast poprawianie błędów wynikających ze źle zaprojektowanej architektury systemu jest bardzo kosztowne, a czasami wprost niemożliwe, to zawsze pozostaje nadzieja, że, przy odrobinie szczęścia, te koszty poniesie ktoś inny.

Z wymienionych wyżej przyczyn twierdzę, że „jakość oprogramowania jest rzeczą zbyt poważną, aby pozostawiać ją informatykom.” Osiągnięcie wysokiej jakości produkowanego oprogramowania wymaga zaangażowania najwyższego kierownictwa oraz musi być częścią strategii przedsiębiorstwa.

3.2. Zadania najwyższego kierownictwa

Jakość kosztuje, a koszty jakości trzeba ponosić ciągle. Największe korzyści z jakości dla przedsiębiorstwa są odsunięte w czasie. Tak samo, odsunięte w czasie są alternatywne koszty utraconych korzyści wynikające z braku jakości. Dlatego zarządzanie jakością w przedsiębiorstwie musi być realizowane w perspektywie strategicznej, obejmującej co najmniej trzy lata, a gros działań musi być podejmowane przez najwyższe kierownictwo.

Norma PN ISO9001:2001 [Iso9001] wymienia zadania, jakie wdrożenie systemu zarządzania jakością stawia przed najwyższym kierownictwem. Zgodnie z nią: „najwyższe kierownictwo powinno dostarczyć dowód swojego zaangażowania w tworzenie i wdrożenie systemu zarządzania jakością oraz w ciągłe doskonalenie jego skuteczności przez

1. zakomunikowanie w organizacji znaczenia spełnienia wymagań ustawowych i przepisów,
2. ustanowienie polityki jakości,
3. zapewnienie, że ustanowione są cele zapewnienia jakości,
4. zaprowadzenie przeglądów zarządzania,
5. zapewnienie dostępności zasobów.” [Iso9001 s. 25]

Z powyższych zapisów wynika, że to na strategicznym poziomie zarządzania musi być podjęta decyzja, że firma informatyczna chce produkować oprogramowanie dobrej jakości. Następnie to najwyższe kierownictwo musi określić sposób w jaki przedsiębiorstwo ma ten zamiar osiągnąć, czyli określić politykę jakości. To na poziomie strategicznym muszą być określone mierzalne i spójne z polityką jakości „cele dotyczące jakości i wymagania dotyczące wyrobu” [Iso9001 s. 29], jak również *last but not least* muszą zostać zapewnione odpowiednie zasoby do ich realizacji. Dodatkowo to „najwyższe kierownictwo powinno przeprowadzać przegląd systemu zarządzania jakością organizacji w zaplanowanych odstępach czasu, w celu zapewnienia jego stałej przydatności, adekwatności i skuteczności. Przeglądem tym należy objąć ocenianie możliwości dosko-

nalenia i potrzebę zmian w systemie zarządzania jakością, łącznie z polityką jakości i celami dotyczącymi jakości” [Iso9001 s. 27]. Wynika z tego, że system zarządzania jakością organizacji nie ma być ustalony raz na zawsze i niezmienny ale musi być ciągle doskonalony. Tak więc zarządzanie jakością w organizacji oraz utrzymanie systemu zarządzania jakością wymaga wprowadzić ciągłego wysiłku ze strony wszystkich, ale największego ze strony najwyższego kierownictwa przedsiębiorstwa.

Powyższe wymagania, stawiane wobec najwyższego kierownictwa, wzięte pod uwagę wraz ze stwierdzeniem, że „zazwyczaj polityka jakości jest spójna z całościową polityką organizacji” [Iso9000 s. 27] prowadzą do wniosku, że system zarządzania jakością powinien być elementem strategii organizacji. Odnosząc to do branży informatycznej – jakość oprogramowania produkowanego przez firmę informatyczną zależy od strategii najwyższego kierownictwa. Jeżeli strategia istnieje, a najwyższe kierownictwo jest skoncentrowane na realizacji długofalowych planów rozwoju, wtedy można oczekiwać inwestycji w zapewnienie odpowiednich zasobów do produkcji oprogramowania dobrej jakości. Jeżeli natomiast strategii nie ma lub jest ale tylko na papierze, a najwyższe kierownictwo jest skoncentrowane na osiągnięciu jak najlepszego wyniku finansowego w najbliższym kwartale, wtedy trudno się dziwić, że będzie oszczędzać na zasobach niezbędnych do zapewnienia jakości.

Uważam, że to właśnie w wyniku braku długofalowej, konsekwentnie realizowanej przez najwyższe kierownictwo strategii zawierającej politykę jakości, dochodzi do takich sytuacji, w których najwyższe kierownictwo firm informatycznych zobowiązuje się do wyprodukowania i wdrożenia oprogramowania ewidentnie poniżej kosztów i przy nierealnych terminach, a następnie kierownicy średniego szczebla razem ze swoimi zespołami są zmuszani do realizowania wdrożeń typu „Marsz Śmierci” [Your97]. Wprawdzie bezpośrednio przyczyny takich sytuacji bywają różne. Czasami jest to świadome działanie spowodowane trudną sytuacją firmy i nadzieją, że bieżące wpływy z realizacji wdrożenia pozwolą zachować płynność finansową oraz przetrwać firmie do lepszych czasów. Często jest to wynikiem naiwności kierownictwa lub marketingu przejawiającej się w składaniu klientom nierealnych obietnic, konkurencji spowodowanej globalizacją lub nieoczekiwanymi kryzysami [por. Your97 ss. 8-19]. Jednakże efekt jest prawie zawsze taki sam – wyprodukowanie, pomimo wielkiego wysiłku, oprogramowania niskiej jakości.

4. Podsumowanie

Zgodnie z definicją, jakość jest to „stopień, w jakim zbiór inherentnych [stałych, tkwiących w istocie produktu] właściwości spełnia wymagania” [Iso9000 s. 25]. Produkt dobrej jakości ma zaspokajać potrzeby i oczekiwania użytkownika.

Zarządzanie jakością w procesie produkcji oprogramowania jest wyzwaniem wobec wszystkich uczestników procesu, na wszystkich szczeblach zarządzania. Wysiłek osób pracujących na poziomie operacyjnym i taktycznym może zostać zaprzepaszczony, jeżeli nie zostanie poparty właściwymi działaniami strategicznymi na poziomie najwyższego kierownictwa.

Na operacyjnym i taktycznym poziomie zarządzania produkcją oprogramowania zarządzanie jakością koncentruje się, w sferze technicznej, na odpowiednim zorganizowaniu testowania i poprawiania błędów oraz na takim zorganizowaniu procesu wytwórczego aby wdrożony system informacyjny realizował, w możliwie najlepszy sposób, wymagania użytkowników.

Największy wpływ na jakość produkowanego oprogramowania mają decyzje i działania podejmowane na poziomie strategicznym. To właśnie najwyższe kierownictwo podejmując decyzje o znaczeniu strategicznym, decyduje o funkcjonowaniu systemu zarządzania jakością w organizacji. Wśród wielu zadań, które norma PN ISO 9001:2001 stawia przed najwyższym kierownictwem między innymi znajdują się: „ustanowienie polityki jakości”, „zapewnienie, aby wymagania klienta były priorytetem w całej organizacji” oraz *last but not least* „zapewnienie dostępności

niezbędnych zasobów” [Iso9000 s. 17]. Niestety, w realnym świecie szczególnie to ostatnie zadanie najwyższego kierownictwa jest często zaniedbywane.

Na poziomie najwyższego kierownictwa ważnym jest, aby system zarządzania jakością był częścią strategii organizacji. Jeżeli zarządzanie jakością nie jest elementem strategii, to istnieje potencjalnie duże ryzyko, że w sytuacji ograniczoności zasobów w organizacji, inne cele okażą się ważniejsze niż jakość, a w przypadku konieczności szukania oszczędności najłatwiej będzie zaoszczędzić na jakości. Krańcową sytuacją jest gdy kierownictwo firmy produkującej oprogramowanie w ogóle nie posiada strategii. W takiej sytuacji, niezależnie od wysiłku pracowników, szansa na produkowanie przez taką organizację oprogramowania dobrej jakości jest znikoma.

Bibliografia

- [Flas07] Flasiński M.: Zarządzanie projektami informatycznymi. Wydawnictwo Naukowe PWN, Warszawa 2007, ISBN 978-83-01-14592-7
- [Górs99] Górski J. red.: Inżynieria oprogramowania w projekcie informatycznym. Zakład Nauczania Informatyki MIKOM, Warszawa 1999, ISBN 83-7158-161-0
- [HaMa06] Hamrol A., Mantura W.: Zarządzanie jakością – teoria i praktyka, Wydawnictwo Naukowe PWN, Warszawa 2006, ISBN-13: 978-83-01-14994-9, ISBN-10: 83-01-14994-9
- [Iso9000] Polska Norma: PN-EN ISO 9000:2006 Systemy zarządzania jakością – Podstawy i terminologia. Polski Komitet Normalizacyjny, Warszawa 2006, ISBN 83-251-0771-5
- [Iso9001] Polska Norma: PN-EN ISO 9001:2001 Systemy zarządzania jakością – Wymagania. Polski Komitet Normalizacyjny, Warszawa 2001, ISBN 83-236-6786-1
- [Kan06] Kan S.H.: Metryki i modele w inżynierii jakości oprogramowania. Wydawnictwo Naukowe PWN, Warszawa 2006, ISBN-10: 83-01-14829-2 (01), ISBN-13: 978-83-01-14829-04
- [KiSr05] Kisielnicki J., Sroka H.: Systemy informacyjne biznesu. Agencja Wydawnicza PLACET, Warszawa 2005, ISBN 83-85428-94-1
- [Kruc03] Kruchten P.: The Rational Unified Process, An Introduction. Addison Wesley, 2003, ISBN 0-321-19770-4
- [Szyj04] Szyjewski Z.: Metodyki zarządzania projektami informatycznymi. Agencja Wydawnicza PLACET, Warszawa 2004, ISBN 83-85428-84-4
- [Your97] Yourdon E.: Death March – The Complete Software Developer’s Guides to Surviving “Mission Impossible” Projects. Prentice Hall PTR, Upper Saddle River, New Jersey 1997, ISBN 0-13-748310-4