



Magazine

# Zapewnianie jakości w projekcie informatycznym - Sztuka? Rzemiosło? A może inżynieria? Czy potrzebne są nam modele procesów? (cz.1)

---

**Autor:** Rafał Dobrosielski

**O autorze:** Absolwent Politechniki Gdańskiej, wydziału Elektroniki Telekomunikacji i Informatyki, ukończył również studia podyplomowe z zakresu Inżynierii Oprogramowania i Zarządzania Projektami. Zawodowo, na co dzień, zajmuje się czynnie planowaniem, zapewnianiem i sprawdzaniem jakości w projektach informatycznych. Jest autorem szkoleń i warsztatów z zakresu zarządzania projektem informatycznym, zarządzania i zapewniania jakości produktów informatycznych oraz modeli procesów inżynierii oprogramowania. Interesuje się analizą i modelowaniem procesów biznesowych w organizacjach jak również psychologią biznesu.  
Email: rdobrosielski@paop.com.pl

**Intermediate**

Level

5

Magazine Number

**Quality in project**

Section in the magazine

*„Proste i jasne zasady i reguły dają początek złożonemu i inteligentnemu zachowaniu.*

*Złożone reguły i regulacje dają początek prostemu i głupiemu zachowaniu”*

Dee Hook

## Wstęp

W artykule „Jakość produktów informatycznych” publikowanym w numerze 2 magazynu CORE rozważaliśmy między innymi pojęcie sukcesu projektu informatycznego. Kryterium, że sukces odniesiemy wtedy, kiedy produkt uda się zrealizować w zaplanowanym budżecie i czasie uznaliśmy za niewystarczające.

Przypomnę, że rzeczywisty sukces projektów, nie tylko informatycznych, polega na uzyskaniu przez zamawiającego oczekiwanych korzyści biznesowych, które mogą nastąpić dopiero wiele miesięcy po zamknięciu projektu [10].

Zatem krytycznym czynnikiem sukcesu staje się jasne, przejrzyste i mierzalne zdefiniowanie celów biznesowych zleceniodawcy i jednakowe ich zrozumienie przez najważniejszych interesariuszy projektu. Pozwala na to rzeczywiste zrozumienie potrzeb i wymagań zleceniodawcy, użytkowników, administratorów.

Cele te nie są formułowane wyłącznie w kategoriach funkcji, które system ma wykonywać, a ich spektrum jest znacznie szersze i dotyczy niezawodności systemu, użyteczności, bezpieczeństwa, pielęgnowalności...

Oczywiście osiągnięcie tych celów na „zadowalającym” poziomie w produkcie końcowym odbywa się w warunkach różnorodnych ograniczeń (jakimi są np. wspomniany budżet i czas) oraz w zmieniającym się środowisku.

Projekt informatyczny i zarządzanie nim to nie sekwencja poszczególnych sprawdzonych działań, ale skomplikowana sieć aktywności, które prowadzą do powstania wielu produktów. Skupienie się na zarządzaniu procesem i czynnościami zająłoby się z jakością produktu i bardzo często ogranicza ryzyko powstania produktu złej jakości. Od początku rozwoju informatyki organizacje poszukiwały i poszukują wciąż dobrego przepisu na realizację tego typu projektów, gwarantującego sukces.

## Sztuka?

Alistar Cockburn w pozycji „Agile Software Development. Gra zespołowa” [14] porównuje wytwarzanie oprogramowania do poezji pisanej przez wiele osób.

W tamtejszym rozważaniu tematu pisania dramatu na zamówienie, z jednej strony uczestniczą osoby, które zamówiły wiersz. Chcą one czegoś wyjątkowego, co wprawi w zachwyt ich samych i ich przyjaciół. Dramat ma jednak niewiele kosztować i potrzebny jest, jak zwykle, bardzo szybko.

Pomimo, że tego typu dzieło powinno powstawać, w naturalny sposób, w wyniku pracy jednej osoby, tamtejszy poeta przyjął zobowiązanie, że dostarczy dzieło szybciej, niż jest w stanie je wytworzyć. Prosi on o pomoc kilka dodatkowych osób. Zaproszone osoby uważają zleceniodawcę za głównego projektanta dramatu, który ustala tematykę i kolejność wydarzeń. Ponieważ wciąż istnieje ryzyko, że nie dostarczą dzieła na czas, proszą o pomoc swoich kolegów i sąsiadów. Ci ostatni nie są poetami, ale po nakreśleniu ogólnej treści otrzymują do wykonania opisy, które nie wymagają zbyt wiele talentu.

Na początku tej historii pojawiły się dobre wiadomości. Ujawniło się wiele indywidualnych talentów, wyodrębniono osoby, które dobre są w opisach przyrody, inne w opisach i charakterystyce osób, jeszcze inne były niezastąpione w podawaniu drastycznych szczegółów.

Wszyscy bardzo się starali, ale z czasem napotykały na wiele naturalnych problemów. Dwie z osób napisały wielostronicowy materiał opisujący mało ważne szczegóły, a główny poeta nie umiał przekonać ich do skrócenia tekstu. Inne osoby wciąż dokonywały zmian w tekście, gdyż same nie były zadowolone ze swojej pracy. Główny poeta chciał, by zajęli się kolejnymi tematami, ale oni nie potrafili przestać doskonalić pierwszych fragmentów.

Czas mijał, grupa stawała się coraz bardziej zdesperowana, o pomoc proszono kolejne osoby. Główny poeta, jedyny, który był dobry w opisach i wyrażaniu emocji rwał sobie włosy z głowy, gdyż nie miał czasu na pisanie swojej części poezji. Był przecież zbyt zajęty koordynacją, sprawdzaniem i dzieleniem zadań.

Zaczęło też brakować pieniędzy. Komunikacja wyglądała strasznie, nikt nie miał aktualnej kopii wiersza i nikt nie znał jej aktualnego stanu.

Czy przytoczona historia nie brzmi nam znajomo?

Alistar Cockburn kończy historię „szczęśliwie”. Dzięki niesamowitemu szczęściu zatrudniono utalentowanego administratora, który po namyśle przedstawił plan całego wiersza, zebrał dane o umiejętnościach każdej z zaangażowanych w pisanie dramatu osób, określił przedziały czasu i harmonogram komunikacji, wskazał standardy wersjonowania i łączenia poszczególnych części poematu w całość, a także zajął się sprawami technicznymi.

Wreszcie dramat udało się dostarczyć zadowolonemu klientowi. Oczywiście znacznie przekroczono zakładany budżet. Główny poeta musiał wyjechać na długie wakacje zarzekając się, że „już nigdy tego nie powtórzy” [;-)].

Opisana grupa natknęła się na te same problemy, z którymi borykają się twórcy oprogramowania. Ludzie pracowali nad czymś, co było unikatowe, czego nie potrafili w pełni zrozumieć.

Dzieło wieńczące projekt informatyczny musi się nie tylko rymować, ale także zachowywać według określonej, oczekiwanej logiki „wystarczająco dobrze”.

Tworzenie oprogramowania bazujące na indywidualnej inspiracji i logice wymusza również inżynierię grupową. Jest aktywnością poznawczą i ekspresyjną, wykonywaną dzięki komunikacji, myślącym ludziom, którzy pracują efektywnie pomimo barier ekonomicznych, ograniczeń kulturowych i niezwykle dużej zależności wyników pracy od wszystkich osób pracujących w zespole.

## **Rzemiosło?**

Rzemieślnik często pracuje metodą prób i błędów. Wiele razy koryguje, dopasowuje i poprawia poszczególne części składowe budowanego przez niego dzieła, do momentu aż będą poprawnie działały razem. Często w wyniku tych działań powstają unikatowe, wartościowe dzieła sztuki czy nawet innowacyjne rozwiązania, jeśli rzemieślnik ma talent i włoży w dzieło wiele serca. Powstają wówczas rzeczy niezwykle piękne i funkcjonalne.

Metoda rzemieślnicza nie sprawdza się w przypadku bardziej złożonych obiektów. Jeśli części jest zbyt wiele, to powinien powstać plan inżynierski stanowiący podstawę prac mechanicznych.

Jedynie mając szczęście i pracując bez większych zewnętrznych ograniczeń, metoda prób i błędów może być skuteczna. Jest ona wówczas dużo bardziej nieefektywna w porównaniu z inżynierią. Istnieje zbyt duże ryzyko, że jeśli końcowe części nie będą do siebie pasować, to zbyt wiele z nich trzeba będzie zacząć od początku [15].

Próba zastosowań inżynierskich do prostych obiektów jest również nieefektywna w stosunku do rzemiosła, ponieważ tracimy mnóstwo czasu na dodatkowe prace projektowe, które tylko mogą niebezpiecznie ograniczyć „artystę”.

To samo odnosi się do oprogramowania. Proste problemy można próbować rozwiązać bezpośrednio przez programowanie. Wówczas cykl życia będzie składał się z programowania i testowania jako procesów sprzężonych. Upraszczając, inżynieria oprogramowania jest dla problemów, których próba rozwiązania metodami rzemieślniczymi prowadzi do nadmiernego powtarzania programowania, co utrudnia osiągnięcie celu w ramach przyjętych zewnętrznych ograniczeń.

Oczywiście problem jest bardziej skomplikowany, gdyż uważam, że stosowanie inżynierii również owocuje zwiększonym stopniem zaufania do produktu końcowego, wpływa na kompleksowość ostatecznego rozwiązania i jego użyteczność.

Pamiętajmy, że cechą charakterystyczną rzemiosła jest niewielka skala i rozmiar.

## **Zarządzanie a inżynieria oprogramowania.**

Z powyższych rozważań wynika jasno, że proces realizacji systemu informatycznego musi być poddany zarządzaniu, w przeciwnym razie jego przebieg i rezultaty byłyby zupełnie nieprzewidywalne.

Poddany zarządzaniu proces realizujący wiele zadań, zmierzających do osiągnięcia wspólnych celów jest nazywany projektem informatycznym.

Z inżynierią oprogramowania mamy do czynienia wtedy, gdy w ramach projektu, działania zmierzają do identyfikacji i rozwiązania problemów o charakterze **technicznym**<sup>[kzm1]</sup>, a rozwiązanie tych problemów następuje przy wykorzystaniu technologii informatycznych poprzez wytworzenie odpowiedniego (nowego lub zmodyfikowanego) oprogramowania.

Pojęcia inżynierii i zarządzania wg profesora Górskiego [7] nie są wyraźnie oddzielone. Realizacja projektu informatycznego prowadzi zwykle do jego dekompozycji na projekty mniejsze, bardziej techniczne, które też podlegają zarządzaniu. Im bardziej techniczne są cele czy zadania projektu, tym większe kompetencje techniczne musi posiadać kierownik (tym bardziej musi być on inżynierem, będąc jednocześnie menedżerem).

## Zapewnianie jakości a model procesów w inżynierii oprogramowania

Model procesu jest usystematyzowanym zbiorem praktyk, które opisują cechy efektywnego procesu, a ich efektywność została potwierdzona przez doświadczenie.

Celem dotychczasowych rozważań było nakreślenie, jak ważne jest odpowiednie dobranie modelu procesu do skali przedsięwzięcia informatycznego. Od modelu procesu będzie zależało nie tylko jakich narzędzi będziemy mogli użyć, aby zapewnić wysoką jakość produktów informatycznych, odpowiedni poziom ich użyteczności i przydatności dla naszego klienta, ale również, jak efektywnie projekt będzie mógł być zrealizowany.

## Czy model procesów jest nam potrzebny?

Zmiana modelu procesów stosowana dla danego typu projektów w organizacji (lub ich ponowna adaptacja) może być konieczna, jeśli obserwujemy następujące symptomy wad procesowych (lub braku procesów):

- uzgodnienia/zobowiązania często nie są dotrzymane
  - spóźnienia w dostarczaniu na rynek,
  - kryzysowe sytuacje w ostatniej chwili,
  - rosnące dramatycznie koszty;
- uwidaczniające się potrzeby doskonalenia procesów zarządzania
  - coraz częściej bywamy zaskoczeni;

- problemy z jakością
  - praca zbyt często ponawiana,
  - funkcjonalność nie działa zgodnie z założeniami,
  - klienci narzekają po dostarczeniu na rynek;
- niskie morale pracowników
  - frustracja pracowników, czy jest coś pod kontrolą? Niewystarczająca motywacja!

Pomimo jawnej obserwacji powyższych symptomów, wiele organizacji zbyt późno dochodzi do wniosku, że wina leży w stosowanym modelu procesu wytwarzania lub w jego braku. Wówczas często spotykamy następujące, błędne stwierdzenia:

- nie potrzebujemy procesów, gdyż mamy:
  - najlepszych ludzi,
  - zaawansowaną technologię i wiedzę,
  - doświadczonych menedżerów;
- zdefiniowany (w domyśle ciężki) proces:
  - ogranicza/przeszkadza w kreatywności,
  - równa się biurokracja i reżim,
  - nie jest potrzebny, gdyż zwykle budujemy najpierw prototypy,
  - przydaje się tylko w dużych projektach,
  - utrudnia elastyczność na wymagającym rynku,
  - pociąga za sobą niebotyczne koszty;

Rozważmy, czy potrzebujemy modeli procesów i po co są one używane w dojrzałych organizacjach produkujących oprogramowanie wysokiej jakości. Modele procesu są najczęściej używane, aby:

- pomóc ustanowić cele i priorytety doskonalenia procesu,
- pomóc zapewnić stabilność, skuteczność i dojrzałość procesów,
- jako przewodnik dla ulepszania procesów projektowych i organizacyjnych,
- stanowić model odniesienia podczas szacowania bieżących praktyk w organizacji;

i dostarczają one:

- powodów do przemyśleń, aby rozpocząć doskonalenie,
- korzyści płynących z czerpania ze wspólnego źródła doświadczeń,
- pola i środowiska do priorytetyzacji działań,
- sposoby zdefiniowania, czym jest doskonalenie dla organizacji;

„Wszystkie modele procesów są złe,  
ale niektóre są użyteczne”

*(George Box, Quality and statistics engineer)*

## **Konieczna jest adaptacja!**

Reasumując, modele procesów mogą nam posłużyć jako wzorzec do opracowania naszej własnej metodologii. Metodologia to konstrukcja społeczna (Ralph Hodgson). To wszystkie kroki, jakie są podejmowane, aby „wypuścić” oprogramowania na rynek. Odnosi się ona do tego, jak, dlaczego i kogo zatrudniasz, w jaki sposób ludzie razem pracują, komunikują się. To zbiór zasad i reguł (również tych niepisanych), którymi kierują się poszczególne zespoły. Zwykle metodologia jest unikatowa dla każdej firmy i stanowi o jej wartości. Modeli procesów nie należy wdrażać ortodoksyjnie – na podstawie danych statystycznych (takich jak: wielkość, złożoność projektu, czas trwania, ilość osób). Należy je adaptować i personalizować ze szczególnym uwzględnieniem specyfiki firmy, różnic kulturowych, kompetencji i doświadczenia personelu.

„Metodologia to konwencje, na które zgodził się zespół. Konwencje, na które zgodził się cały zespół to nic innego jak konstrukcja społeczna. To również konstrukcja, którą od czasu do czasu należy poddawać ocenie i niezbędnym poprawkom.” [„Agile Software Development. Gra zespołowa”]

Podczas adaptacji czy projektowania metodologii należy pamiętać o zmienności ludzi. Ludzie i ich zachowania w różnych projektach nie są pewnym komponentem. Wynika to między innymi ze specyfiki projektów informatycznych, które charakteryzują się stosunkową małą powtarzalnością działań.

Najczęściej popełniane błędy podczas wdrażania metodologii:

Nie należy szukać jednej, wspólnej metodologii dla projektów różnej wielkości, tworzonych w różnych technologiach, środowisku i kulturze. Tego typu desperackie poszukiwania uniwersalnego przepisu, który sprawdzi się w każdych warunkach kończą się zwykle fiaskiem, nawet w ramach jednej, większej organizacji.

W procesie poszukiwania/adaptowania metodologii należy pamiętać o tym, że różni ludzie preferują różne metody pracy. Najważniejsza jest dobra, akceptowana komunikacja i skuteczne rozwiązywanie problemów. Nie znaczy to, że wdrażana metodologia ma adaptować wszystkie złe nawyki popełniane do tej pory. Na zmiany, które metodologia ma zamiar wprowadzić, powinny mieć jednak wpływ wszystkie strony. Alistair Cockburn porównuje metodologię do obcisłej kurtki. Nie powinna ona być za ciasna, by nie krępować zbytnio naszych ruchów, a jej uszycie powinno nastąpić po wielu przymiarkach. Aby spełniała ona swoją rolę, musi być chętnie noszona.

Ważnym pytaniem jest, jak wysoki stopień formalizacji metodologia powinna wprowadzać. Długo uważano, że wdrożenie tak zwanych metodologii ciężkich, z dokładnym śledzeniem i większą liczbą elementów jest bezpieczniejsze dla nas i dla projektu. Nakładanie większego stopnia formalizacji na działania zespołu, dodatkowych punktów kontroli i podsumowań, tworzenie dodatkowych raportów nie musi być wcale dobrym rozwiązaniem. Mały, 4 czy 6 osobowy zespół, pracujący ze sobą w tym samym pomieszczeniu nad skomplikowanym problemem, pod presją czasu, raczej nie ukończy pracy szybciej poprzez tworzenie zbędnych, nadmiarowych pośrednich produktów pracy, diagramów Gantt'a i szczegółowych planów iteracji. W tym przypadku, nasze poczucie bezpieczeństwa osiągnięte poprzez wdrożenie ciężkiej metodyki będzie tylko pozorne.

Jedną z najtrudniejszych spraw w zarządzaniu projektem informatycznym jest właśnie znalezienie kompromisu pomiędzy wyłącznie takim zwiększaniem nakładu pracy, który zwiększa nasze poczucie bezpieczeństwa, szansę osiągnięcia sukcesu (ale i koszt projektu), a zaufaniem do ludzi.

## Literatura

- [1] James Bach: „Good Enough Quality: Beyond the Buzzword”, IEEE Computer, Sierpień 1997
- [2] Anna Bobkowska: „Inżynieria Oprogramowania”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
- [3] Grady Booch: „Leaving Kansas” IEEE Software. 15(1), Styczeń/Luty 1998
- [4] Victor R. Basili: „The goal question metric approach”, ( Advanced Computer Studies, Department of Computer Science, University Of Maryland)
- [5] John Fodeh: „What's on Your Dashboard” Better Software, October 2007
- [6] Janusz Górski: „Zarządzanie projektem informatycznym”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
- [7] Janusz Górski: „Inżynieria oprogramowania w projekcie informatycznym”, Zakład Nauczania Informatyki MIKOM, Warszawa 1999
- [8] Jerzy Kaczmarek: „Metryki i zapewnianie jakości”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
- [9] Stephen H. Kan: „Metryki i modele w inżynierii jakości oprogramowania”, Wydawnictwo Naukowe PWN SA, 2006
- [10] Adam Korczowski: „Zarządzanie ryzykiem w projektach informatycznych. Teoria i praktyka”, Wydawnictwo Helion 2010
- [11] Per Kroll, Philippe Kruchten: „Rational Unified Process od strony praktycznej”, Wydawnictwo Naukowo-Techniczne, Warszawa 2007

[12] Philippe Kruchten: „Rational Unified Process od strony teoretycznej”, Wydawnictwo Naukowo-Techniczne, Warszawa 2007

[13] Zdzisław Szyjewski: „Metodyki zarządzania projektami informatycznymi”, Wyd. PLACET, Warszawa 2004

[14] Alistair Cockburn: „Agile Software Development. The Cooperative Game”, Second edition

[15] Dick Hamlet, Joe Maybee „Podstawy techniczne inżynierii oprogramowania”, Wydawnictwo Naukowo-Techniczne, Warszawa