

Usprawnianie projektowania oprogramowania i usprawnianie pracy działu IT z wykorzystaniem najlepszych praktyk przemysłowych

Andrzej Góralczyk
Portal Dyrekcja.pl

portal.dyrekcja@businessdialog.pl

Abstrakt. Jakość oprogramowania w wielu obszarach zastosowań często wzbudza poważne zastrzeżenia, a efektywność pracy działów IT w przedsiębiorstwach często nie odpowiada oczekiwaniom. Wśród samych zainteresowanych pojawiają się opinie, że specyficzne metodyki i rozwiązania organizacyjne wypracowane w sektorze IT w zbyt małym stopniu uwzględniają wymagania użytkowników i klientów. W pierwszej części referatu pokazano na kilku przykładach, że problemy powstają w sytuacjach, gdy te specyficzne rozwiązania nie uwzględniają także podstawowych reguł inżynierii i organizacji pracy wypracowanych w innych, tradycyjnych dziedzinach przemysłu i usług. Na tym tle przedstawiono propozycje transferu najlepszych praktyk przemysłowych do trzech obszarów: organizacji projektowania software'u, modelowania procesów biznesowych i organizacji pracy działu IT w przedsiębiorstwie.

1. Wprowadzenie

Nie jest moim celem zasypywanie Czytelnika gradem zarzutów pod adresem informatyki. Jednakże nie mogę tego uniknąć, gdyż propozycje rozwiązań, z jakimi występuję od pewnego czasu, wyrastają właśnie z analizy przyczyn jej licznych niedostatków.

W roku 1994 korzystałem ze swojego pierwszego peceta i mogłem już skompletować wszystkie potrzebne mi programy. Miałem procesor tekstu i system finansowo-księgowy dla swojej firmy, darmowy program DTP, właśnie ukazała się pierwsza graficzna przeglądarka WWW i pierwsza wyszukiwarka, nabyłem także Excela na miękkiej dyskietce i wersję alfa bazy danych Access. Dzisiaj mam komputer o pamięci operacyjnej 250 razy większej, 1000 razy większej pojemności dysku twardego i 200 razy szybszym procesorze. Używam nieco innego zestawu programów, ale nie robią one nic więcej, niż 13 lat temu i nie robią tego szybciej. Toteż trudno mi przyjmować bez sceptycyzmu hasła o innowacyjności informatyki bądź o jej wyjątkowym zaawansowaniu technicznym.

Ten zdumiewający wynik jest niewątpliwie wynikiem historycznego błędu – koncepcji rozwoju sektora dokonującego się wedle zasady stałego utrzymywania przewagi technicznej dostawcy nad Użytkownikiem. Pod tym względem informatyka jest wyjątkiem pośród innych dziedzin techniki i przemysłu.

Informatyka izoluje się także od innych dziedzin przemysłu i inżynierii. Być może dominującym tego powodem jest przekonanie o jej wyjątkowości. Nie pierwszy to przypadek w historii – pamiętam, że podobne przekonanie żywili w latach 1950-1970-tych pionierzy przemysłu tworzywo sztucznych. Pamiętam artykuły o niezwyklej złożoności technologii, wizje nowego stylu życia z płaszczem ortalionowym dla każdego, marzenia o lekkich konstrukcjach sięgających stratosfery itd. Informatyka powtarza dziecięcą chorobę każdej chyba młodej dziedziny twórczości inżynierskiej i praktyki.

Tradycyjne dziedziny inżynierii także czasem cierpią na chorobę izolacjonizmu i braku pokry. Tyle tylko, że dopracowały się recept na pokonanie tej słabości, a informatyka jeszcze nie. Toteż zbyt wiele jej produktów ma wady koncepcji i defekty wykonania – zwłaszcza oprogramowanie dla przedsiębiorstw. Organizacja prac inżynierskich jest tak „specyficzna”, że olbrzymia większość projektów kończy się po terminie i nie mieści się w budżecie. Praktycy często dodają „...jeśli w ogóle się kończy”.

Podobnie jest z utrzymaniem wyposażenia. Dostawcy prześcigają się w złożoności swoich produktów, gdyż wydaje im się, że jest to doskonały sposób na utrzymywanie przewagi konkurencyjnej. Z podobną motywacją walczą na standardy, koniecznie zamknięte, więc nie udokumentowane dla Użytkownika. To jest pierwsze i najważniejsze źródło permanentnych problemów dotyczących prawie każdą firmę korzystającą z oprogramowania dla przedsiębiorstw. Nie da się tych urządzeń i programów zintegrować siłami własnych inżynierów, więc Klient musi korzystać z szalenie kosztownych usług tzw. integratorów. Musi słono płacić za historyczne błędy swoich dostawców, gdyż nie ma innego wyjścia.

Zagadnienie integracji systemów informatycznych jest zbyt obszerne, aby się nim zajmować w krótkim z konieczności artykule. Natomiast zajmę się bardziej przyziemnymi i codziennymi problemami, na jakie napotyka zapewnienie wymaganej dostępności wyposażenia – sprzętu i oprogramowania.

Patrzę na rozwój informatyki oczyma inżyniera przemysłowego i dostrzegam zagadnienia podobne do tych, jakie pojawiały się w historii różnych dziedzin praktyki, a także szanse rozwiązania wielu z nich przy pomocy podejść i technik dobrze sprawdzonych gdzie indziej. O tym traktuje dalszy ciąg niniejszego artykułu.

2. Projektowanie oprogramowania

Skoro proces projektowania oprogramowania daje wyniki o znacznie gorszej jakości, niż projektowanie innych produktów w przemysłach tradycyjnych, zasadne jest poszukiwanie przyczyn tego stanu rzeczy w samym procesie. Czym różni się proces projektowania w IT od projektowania w innych przemysłach?

2.1. Obraz ogólny procesu projektowania

Jeden ze znanych „guru” oprogramowania dla przedsiębiorstw przedstawił na wrocławskiej konferencji dane o typowym rozkładzie czasu, jaki poświęca się na kolejne etapy projektowania: 20% na opracowanie wizji, założeń, koncepcji i planu produktu, 60% na projektowanie prototypu i 20% na końcowe testowanie. W „tradycyjnym” przemyśle rozkład jest inny, co ilustruje poniższe zestawienie:

Tabela 1. Podział czasu projektowania pomiędzy jego etapy oraz wyniki projektowania w IT i w przemyśle samochodowym

Etap	IT	Przemysł samochodowy (Europa)
Wizja, założenia i plan	20%	40%
Projektowanie prototypu	60%	49%
Testowanie	20%	11%
Wynik: w budżecie	ok. 30%	ok. 80%
Wynik: wymagana jakość	ok. 30%	ok. 95%

Obliczenia odnoszące się do przemysłu samochodowego w Tabeli 1. nie obejmują projektowania produkcji, gdyż taki etap nie występuje w przypadku oprogramowania dla przedsiębiorstw. Istotna różnica występuje także na etapie projektowania. W przypadku oprogramowania prawie cały ten proces to kolejne modyfikacje i rozbudowa prototypu, który powstaje we wczesnej fazie procesu. Natomiast w przemyśle samochodowym projektowanie odbywa się „na sucho”, tzn. w postaci dokumentacji elektronicznej i dzieli wyraźnie na 2 podprocesy: projektowanie specjalne (elementów istotnie nowych) oraz projektowanie produktu jako całości, które można opisać jako montaż całości projektu z elementów nowych i standardowych oraz nadawanie tej całości właściwej formy. Prototyp zaś, w kilku egzemplarzach, powstaje dopiero w fazie testowania.

Wartości przeciętne ukazane w Tabeli 1., a w odniesieniu do IT wręcz orientacyjne, ze swej natury nie ujmują zróżnicowania, o którym należy powiedzieć parę słów. W przemyśle tradycyjnym zdarzają się – aczkolwiek rzadko – projekty z bardzo przekroczonym budżetem oraz z wadami koncepcji. Charakterystyczne jest to, że większość z nich to „szczytowe osiągnięcia sztuki inżynierskiej”, na przykład dedykowane automaty o niezwykle wysokiej wydajności. Podstawowe ich mankamenty wynikają z wysokiej złożoności oraz z tego, że muszą pracować w bardzo stabilnych warunkach. Zatem często się „zacinają”, a naprawy i przebrojenia trwają wieki! Z kolei w projektowaniu software'u spotykamy metodyki, w których testowanie zajmuje o wiele więcej czasu, niż to pokazano w Tabeli 1., a także metodyki „napędzane testowaniem”, np. Test-driven Development.

2.2. Analiza przeglądowa procesu projektowania

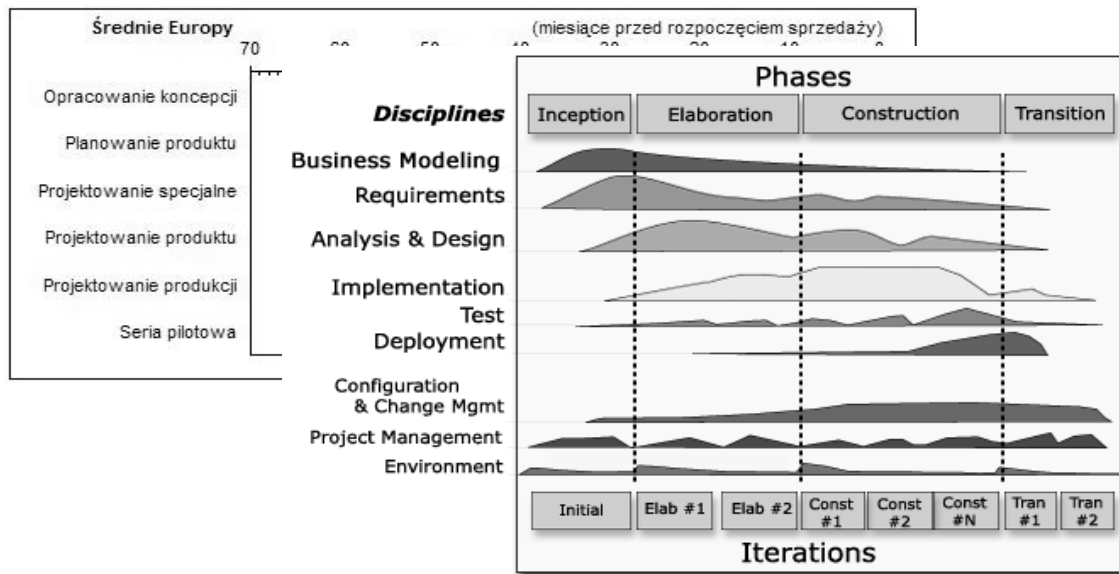
Jeśli dobrze rozumiem logikę rozwoju inżynierii oprogramowania, jest ona po części wyrazem dążenia do przełamania ograniczeń tzw. modelu kaskadowego (*ang. waterfall model*). Jednakże sam proces projektowania od początku natrafia na poważną przeszkodę, mianowicie na barierę komunikacyjną pomiędzy Klientem i projektantami. Podobna bariera występuje niekiedy w źle

zarządzanych firmach tradycyjnego przemysłu, lecz w informatyce jest ona o wiele trudniejsza do pokonania, z racji wspomnianego na wstępie izolacjonizmu. Można powiedzieć, że informatyka sama potęguje tę trudność – na własne życzenie. Trudność zaś objawia się tym, że Klient nie jest w stanie ani zrozumieć możliwości i ograniczeń technologii, ani sformułować swoich wymagań w języku zrozumiałym dla informatyków.

Typowe rozwiązanie omawianego problemu komunikacyjnego jest niewłaściwe i paradoksalne. Mianowicie fazę formułowania założeń skraca się do minimum, aby później, w następnych fazach, projektować częściowo „na wycucie” oraz uzupełniać i zmieniać owe założenia, co dezorganizuje cały proces. W drastycznych przypadkach dochodzi nawet do porzucenia wykonanych już prac i rozpoczynania projektowania niemal od początku. W przypadkach mniej drastycznych mamy do czynienia z tzw. iteracjami, w toku których następuje wzajemne dopasowywanie projektu i modyfikowanych wielokrotnie założeń. W ten sposób zamiast jednej kaskady mamy wiele kaskad, ułożonych w iteracje. Iteracje zaś można postrzegać jako oscylowanie pomiędzy niedopracowanymi założeniami a niedopracowaną ich realizacją.

Niezwykle skrócona faza formułowania założeń ma także źródło w awersji projektantów oprogramowania do planowania i dokumentowania. Literatura na ten temat jest pokaźna, więc nie będę przytaczał przykładów ani opinii, tym bardziej że moim celem nie jest krytyka, lecz szukanie rozwiązań. Dodam natomiast uwagę o testowaniu. Jak wspomniano we wstępie, testowanie rozwijanego prototypu odbywa się przez prawie cały czas opracowania produktu i projektowania. Patrząc na to z dystansu można powiedzieć, że nie bardzo wiedząc co ma zostać zrobione, a nawet nie chcąc wiedzieć, zaczynamy od kodowania, a później prawie do końca projektu zajmujemy się usuwaniem błędów.

Zarówno mnogość iteracji, jak i bezustanne testowanie są w znacznej części podyktowane koniecznością naprawy tego, co zostało popsute na skutek złych praktyk komunikacji z Klientem i Użytkownikiem oraz złych praktyk we wczesnych fazach projektu. Mimo to zostały usankcjonowane jako dobre praktyki projektowania oprogramowania! Istnieje mnóstwo przykładów takich „metodyk”, „standardów” a nawet „paradygmatów” programowania, które nawołują do „nie marnowania czasu” na planowanie i dokumentowanie. Przykłady można mnożyć, ale przytoczę tylko jeden: metodykę zwaną Rational Unified Process. Zapewne powstała w najlepszej wierze, i jej niewątpliwą zaletą jest swoisty realizm – zgoda na to, że projektanci i tak nie zrobią dobrze tego, czego nie lubią robić. Dla porównania zamieszczam na Rys. 1. także wykres ilustrujący przebieg projektowania samochodu.



Rys. 1. Porównanie procesów projektowania samochodu z procesem projektowania oprogramowania (Rational Unified Process). Źródła: pierwszy wykres – opracowanie własne autora na podstawie Clark K. B., Fujimoto T., *Product Development Performance*, Harvard Business School Press, Boston, Massachusetts, 1991 (wydanie 5, 1995); drugi wykres – copyright Rational Software Corporation, obecnie oddział IBM.

Trzecie zagadnienie, na które zwróciłem uwagę analizując niedostatki inżynierii oprogramowania dotyczy błędów w organizacji pracy. Zauważyłem mianowicie, że znaczna część opóźnień i zaległości wynika z tego, iż programiści bezskutecznie poszukują wśród założeń i planów produktu wytycznych, których akurat potrzebują. Kilku szefów projektów potwierdziło, że założenia i plany formułują na podstawie uzgodnionej z Klientem specyfikacji produktu i lecz nie uwzględniają w tych dokumentach wymagań klienta w procesie, tzn. Programisty. Oznacza to, że w omawianych przypadkach nie jest respektowana procesowa natura procesu projektowania.

2.3. W kierunku projektowania solidnego (Robust Design)

Problem trudności komunikacji z Klientem, czy też Użytkownikiem produktu rozwiązuje się w tradycyjnym przemyśle na różne sposoby. Sposobem dobrze sprawdzonym jest mieszany zespół projektowy, w którym uczestniczą osoby z różnych działów przedsiębiorstwa, jeśli jest to projekt firmowy albo specjaliści badań użytkowych, jeśli produkt jest projektowany dla Klienta zewnętrznego. Trzeba powiedzieć, że w niektórych przedsiębiorstwach produkcyjnych w Polsce takie mieszane zespoły realizują projekty różnorodnej natury, w tym także informatyczne. Bywa i tak, że szefem projektu informatycznego nie jest informatyk. Tego rodzaju doświadczenia wyrastają z dobrych tradycji inżynierskich i warto je przyswajać jako najlepsze praktyki zarządzania projektami.

Istotną barierą komunikacji w większych firmach jest brak wspólnego języka, zrozumiałego dla różnych społeczności. Liczne przykłady dowodzą, że dobry wspólny język rozmowy o biznesie jest zawarty w technikach organizatorskich. Projekt informatyczny może być świetną okazją do poprawy komunikacji z biznesem, jeśli specyfikacja produktu zostanie opracowana wspólnie z wykorzystaniem technik organizatorskich specjalnie do tego przeznaczonych i sprawdzonych, jak na przykład drzewo problemowe bądź Rozwinięcie Funkcji Jakości (*ang. QFD – Quality Function Deployment*).

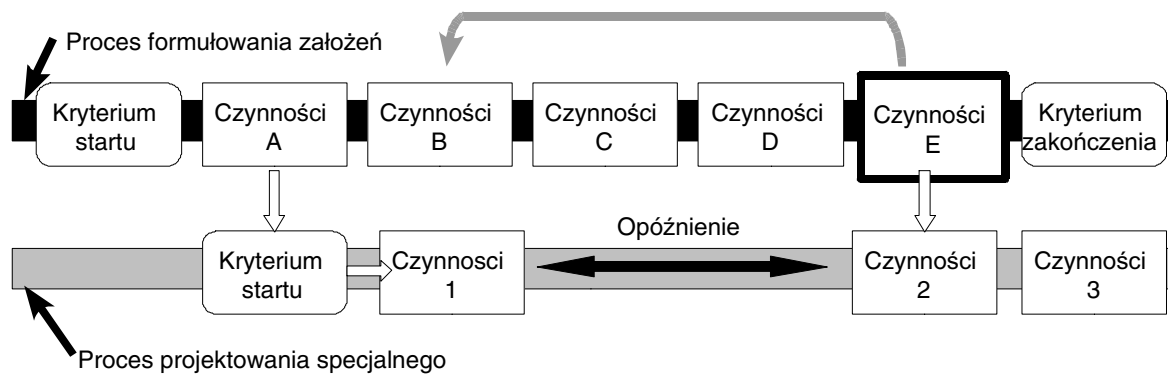
Każda z dobrze w przemyśle zdomowionych technik organizatorskich niesie ze sobą dodatkowe, nietechniczne wartości. Na przykład opracowanie drzewa problemowego jest zadaniem niełatwym dlatego, że wymaga całkowitego odejścia od kryteriów inżynierskich na rzecz kryte-

riów Klienta. Pomaga zatem pokonać problem, o którym pewien dyrektor informatyki powiedział – „myśląc o nowym produkcie nie jesteśmy w stanie oderwać się od myślenia w jaki sposób go zrealizujemy”. Dobrze opanowanie metodologii QFD pomaga natomiast uniknąć problemów określanych często w zdaniu „Klient nie wie dokładnie czego chce, więc ma tendencję do mnożenia wymagań, które potem okazują się nieistotne, a ich realizacja kosztuje wiele niepotrzebnej pracy”.

Idealem w dziedzinie organizacji prac inżynierskich jest projektowanie zintegrowane – rozwinięta forma dawniejszego projektowania współbieżnego (*ang. concurrent engineering*). Zapewne ideał ten przyświeca twórcom wielu metodyk takich, jak wspomniany Rational Unified Process. Jednakże w praktyce zamiast częściowego nakładania się faz procesu projektowania metodyki te prowadzą do nakładania się iteracji, co wyraźnie ukazuje Rys. 1.

Droga do projektowania zintegrowanego nie musi być trudna, chociaż na pewno nie może być krótka. Rozpoczyna się ona od analizy procesu takiego, jakim jest on w praktyce, aby wychwycić typowe miejsca, w których rozpoczynają się opóźnienia. Na Rys. 2. pokazano przykładowo, że takie opóźnienie pojawia się w fazie projektowania specjalnego. Jeśli już analizujemy dokumentację opóźnionego projektu, to zbadajmy także, których wytycznych potrzebował projektant, aby wznowić swą pracę. Na rysunku pokazano, że są to wytyczne opracowane w wyniku czynności należących do fazy formułowania założeń (np. planu produktu). Zatem faza formułowania założeń wymaga niewielkiego przeprojektowania w taki sposób, aby czynność E zakończyła się przed rozpoczęciem przez projektanta czynności 2. W ten sposób, przeprojektowując krok po kroku wczesne fazy procesu (w tym przypadku fazę formułowania założeń) możemy wyeliminować niepożądane opóźnienia i zaległości. Takie stopniowe przeprojektowanie procesu jest jednym z przykładów Ciągłego Ulepszania (*jap. KAIZEN*).

Wreszcie końcowy punkt programu dojścia do projektowania solidnego – zapewnienie jakości. Eliminowanie błędów wykrytych podczas testowania nie jest zapewnieniem jakości – jest utrwalaniem niskiej jakości! Lecz spróbujmy przeprowadzić statystykę, aby zorientować się które przyczyny powodują najczęściej błędów i wyeliminujmy te przyczyny tak, aby już nigdy te najczęstsze błędy się nie pojawiały. Gdy już nabierzemy wprawę w „statystycznym spojrzeniu” na nasz proces i gdy poprawimy znacznie jego jakość, spójrzmy w ten sam sposób na koncepcję i plan naszego produktu. Najlepiej wspólnie z Użytkownikiem. Opierając się na doświadczeniu naszym i Użytkownika spytajmy, które elementy produktu mogą okazać się krytyczne w fazie eksploatacji i jak wielkie odstępstwa od wymagań mogą one powodować. Wybierzmy tylko 2 albo 3 czynniki krytyczne i oszacujmy prawdopodobieństwa lub częstości odstępstw osobno dla każdego z tych czynników. Następnie przeprowadźmy analizę wariancji i na tej podstawie zidentyfikujmy fragment projektu, który musi zostać szczególnie dobrze dopracowany. Postępowanie takie nazywa się projektowaniem eksperymentalnym i jest główną techniką organizatorską projektowania solidnego.



Rys. 2. Schemat przykładowego przeprojektowania prac inżynierskich w celu eliminacji opóźnienia. Przeprojektowanie polega w tym przykładzie na przesunięciu czynności E w taki sposób, aby została ukończona przed rozpoczęciem przez projektanta czynności 2.

2.4. Podsumowanie rozdziału 2

W dwóch etapach analizy procesu projektowania oprogramowania zwróciliśmy uwagę na 3 przyczyny jego mankamentów: nie dość jasne i niekompletne formułowanie koncepcji i planu produktu, praktykę nadmiernej liczby iteracji i testowania oraz opóźnienia i zaległości pojawiające się w procesie projektowania. Zarysowaliśmy program praktycznych przedsięwzięć mających na celu poprawę organizacji prac inżynierskich i prowadzących poprzez projektowanie zintegrowane do projektowania solidnego (Robust Design).

3. Komputer w procesie biznesowym

Jeszcze w minionym dziesięcioleciu projektowanie procesów biznesowych było domeną inżynierów przemysłowych. W ostatnich latach zostało niemal zupełnie zawłaszczone przez inżynierów oprogramowania i informatyków w przedsiębiorstwach. Wśród licznych pozytywnych tego skutków można wymienić przede wszystkim upowszechnienie przekonania, że ulepszanie procesów biznesowych to przedsięwzięcie potrzebne i korzystne. Dzisiaj projekt usprawniający nie jest już przywilejem firm najbogatszych i można nawet „zamówić sobie proces” u wyspecjalizowanych dostawców. Jednakże pojawiły się także skutki negatywne, na przykład:

- nadmierne usztywnienie działania firmy procedurami („terror procedur”),
- zbyt dużo danych o niskiej wartości (niepotrzebnych) i jakości,
- zakłócenia procesu przez nieoptymalne działania sterujące (np. tzw. eskalacje).

Analityczne spojrzenie na pracę projektantów procesów biznesowych pozwala odkryć rzeczy zdumiewające, wypływające z tego, o czym już wielokrotnie tutaj mówiliśmy – informatycy ignorowali wiele podejść i zasad wypracowanych przez dziesięciolecia i poszli własnymi drogami. Dzisiaj niektórzy, po licznych niepowodzeniach projektów, apelują o „powrót do podstaw, do starych sprawdzonych praktyk i elementarnych definicji”. Niniejszy rozdział jest jedną z prób odpowiedzi na te apele.

3.1. Ontologia procesu

Kiedyś napisałem w Wikipedii artykuł przeglądowy o zarządzaniu. Ktoś dodał na końcu króciutki rozdział pt. „Techniki zarządzania”, a w nim sporą listę takich rzeczy jak programowanie dynamiczne czy programowanie liniowe. Pomylił techniki ekonometryczne i programistyczne

z technikami zarządzania. Nie wiem, co pozwala fanowi informatyki na tego rodzaju anarchię pojęciową, przypuszczam jednak że przekonanie o tym, iż zarządzanie to domena rozważań dowolnych, nie poddanych rygorom ścisłości.

Dzisiaj mylenie procesu z programem to plaga powszechna oraz źródło wielu nonsensów i błędów koncepcyjnych popełnianych przy konstruowaniu procesów biznesowych.

Wróćmy zatem do pojęć podstawowych. Proces to ciąg operacji przetwarzania (*ang. processing*), a program to ciąg instrukcji. Mamy zatem 2 istotne różnice pomiędzy jednym a drugim.

Po pierwsze – przedmiot. Przedmiotem „działania” procesu jest tworzywo: materiał, dane, „sprawa do załatwienia”. Przedmiotem „działania” programu jest natomiast proces przetwarzania danych oraz sam program.

Po drugie – modalność. Proces należy do „świata rzeczywistego”, stanowi obraz tego, co JEST robione z tworzywem pod określonymi warunkami. Program zaś należy do „świata postulowanego”, określa to co MA BYĆ robione pod określonymi warunkami, niekoniecznie z tworzywem.

Oba te rozróżnienia można ująć w jednym stwierdzeniu: proces należy do sfery działania rzeczywistego, a program do sfery sterowania działaniem. Myląc jedno z drugim tracimy mnóstwo możliwości, które pojawiają się, jeśli te dwie sfery wyraźnie i jednoznacznie oddzielamy.

Przejawem łączenia dwóch omawianych sfer – procesu i sterowania – jest włączenie zdarzeń do definicji procesu biznesowego i notacji służących do jego opisów (w grupie języków BPEL). Zdarzenia należą bowiem również do sfery sterowania. W ten sposób mylenie procesu i programu zostało usankcjonowane w standardzie modelowania procesów biznesowych.

3.2. Marzenie informatyki i dekonstrukcja procesu

Dyskusje i spory na poruszony przed chwilą temat toczą od wielu miesięcy, usiłując dociec co jest pierwszą przyczyną owego swoistego mieszania definicji. Od jednego z architektów procesów biznesowych usłyszałem, że proces jest „współdziałaniem człowieka i komputera”. Zafrapowany przyjrzałem się kilku konkretnym „mapom” procesów sporządzanych przez informatyków oraz interpretacjom owych map. Wnioskiem z tych obserwacji jest hipoteza, że źródłem całego zamieszania jest marzenie informatyków, aby komputer uczynić partnerem człowieka w pracy, a nie tylko użytecznym narzędziem. W innych dziedzinach inżynierii oprogramowania także marzenia sterują koncepcjami. Na przykład u źródeł koncepcji Sieci Semantycznej leży marzenie o tym, aby komputer zastąpił człowieka w rozumieniu, wnioskowaniu, a nawet decydowaniu.

Lecz marzenia nie ziszczają się, jeśli nie podchodzi się do ich urzeczywistniania w sposób systematyczny. Marzenie o komputerze jako koniecznym aktorze procesu biznesowego wymaga najpierw cierpliwej pracy nad określaniem jego rzeczywistej roli i zadań w tym procesie. Krok po kroku. Niekoniecznie od razu domagając się, aby zastępował człowieka w myśleniu, ponieważ to przede wszystkim człowiek ma myśleć. Podobnie w procesie biznesowym – niekoniecznie od razu domagając się, aby zastępował człowieka w decydowaniu, lecz aby pomagał mu w tym. Potrzebna jest refleksja! Na przykład „Co to znaczy, że komputer wspomaga decydowanie?”. Może to być doskonałe ćwiczenie techniki drzewa problemowego.

Przechodząc do konkretnego warto zwrócić uwagę na przykład na stan zaawansowania koncepcji udziału komputera w sterowaniu procesem biznesowym. Chyba najbardziej rozpowszechniona obecnie jest w tej dziedzinie koncepcja przepływu pracy (*ang. workflow*). W istocie rolą komputera jest tutaj „porządne” układanie człowiekowi kolejki zadań, które człowiek ma wykonać. Jednakże większość szefów informatyki w firmach, które wdrożyły workflow donosi, że to nie działa – że ludzie i tak robią to, co sami uważają za pilne i ważne, a niekoniecznie to, co ma wysoki priorytet nadany przez workflow. Działa to natomiast tylko w tych firmach, które zastosowały dodatkowy system „kijów i marchewek”, na przykład redukowały premie pracowników za ignorowanie

takich priorytetów. To jest ważna nauka – komputer nie zarządza procesem biznesowym, zarządzają nim ludzie!

Podobnie ze sterowaniem. Typowe rozwiązanie dla przypadków opóźnień i zaległości w wykonywaniu procesu biznesowego proponuje się rozwiązywać przy pomocy tzw. eskalacji, czyli przekazania przełożonemu informacji o opóźnieniu na stanowisku podwładnego. Skutkiem tej informacji ma być interwencja ze strony przełożonego. W praktyce interwencja powoduje na ogół jeszcze większy bałagan i zakłócenie pracy, niż owo opóźnienie. Interwencja bowiem nie jest optymalnym sposobem sterowania procesem. W ten sposób komputer w roli strażnika terminowości wykonania zadań inicjuje zakłócenie procesu podobnie, jak strażnik źle wyszkolony i nieświadomy swej roli.

4. Komputer pod opieką

Obserwowałem kiedyś przez otwarte drzwi, łącznie przez 3 godziny i 40 minut, co się dzieje w pokoju informatyków w średniej firmie. Przy biurku jednego z nich nie działo się prawie nic, gdyż jego zadaniem było krążenie po piętrach biurowca i rozwiązywanie drobnych problemów zgłoszonych przez Użytkowników komputerów i telefonów. W ciągu tych 3 godzin i 40 minut pracownik ów wrócił do pokoju 9 razy:

- 3 razy – po toner do drukarki,
- 2 razy – po zasilacz do laptopa,
- 1 raz – po ładowarkę do telefonu komórkowego,
- 1 raz – po śrubokręt,
- 1 raz – po dłuższy kabel do sieci,
- 1 raz – po kartę sieciową do laptopa.

Nietrudno zorientować się, że mógłby zaoszczędzić sobie czasu i wysiłku, gdyby jego praca została lepiej zorganizowana. Na przykład, skoro zużywa się dużo toneru, to nie ma sensu trzymać zapasu w jednym miejscu, lepiej ulokować podręczne zapasy na każdym piętrze. Pracownik mógłby mieć także torbę z podręcznymi narzędziami i drobnymi częściami najczęściej potrzebnymi, jak śrubokręt czy ładowarka do telefonu.

Ten prosty przykład ukazuje tylko wierzchołek góry lodowej licznych i trudnych problemów utrzymania infrastruktury informatycznej w przedsiębiorstwach. Zaczynają się one od konfliktu wymagań. Z jednej strony wymagamy, aby urządzenia były dostępne przez cały czas, z drugiej – chcemy taki stan osiągać tanio. Konflikt typowy dla nazbyt technicznego podejścia do zagadnień zapewnienia niezawodności.

Trzeba od razu powiedzieć, że elektronika i informatyka wyróżniają się pozytywnie na tle innych dziedzin techniki szczególnie wysoką dostępnością urządzeń. Jednakże z grubych oszacowań wynika, że osiągnięte jest to kosztem ok. 20 razy za wysokim. To też niedużo w porównaniu z niektórymi innymi dziedzinami przemysłu, ale skro można byłoby taniej...

Według współczesnego podejścia do utrzymania ruchu celem zapewnienia niezawodności nie jest osiągnięcie maksymalnej niezawodności! Celem jest biznes – minimum sumy strat z tytułu obniżonej dostępności oraz kosztów zapewnienia wymaganej dostępności w ciągu całego cyklu życia urządzenia. Inaczej mówiąc, celem jest „wyciśnięcie” z niego maksymalnie wiele.

Świadomość ekonomicznej natury dostępności szybko rośnie, wraz z gromadzeniem doświadczeń dotyczących zawierania i realizacji tzw. SLA – umów o minimalnym dopuszczalnym poziomie obsługi. Po prostu rośnie świadomość że wysoki poziom obsługi kosztuje, tym więcej im ma

być wyższy. Przy bardzo wysokich poziomach obsługi koszty rosną w przybliżeniu eksponencjalnie, więc rozsądnym podejściem jest łagodzenie wymagań gdzie tylko się da.

Druga kwestia to optymalizacja planów obsługi. Nie sposób tego wymyślić, więc trzeba zawsze zaczynać od statystyki strat, aby wychwycić te awarie i defekty, które są najbardziej dotkliwe ekonomicznie. Wówczas będzie na przykład wiadomo które urządzenia można zdublować, gdyż są tanie, ale awaria powoduje duże straty czasu pracy. Albo które części zamienne i wymienne opłaca się kupować w większych partiach, a które tylko w razie absolutnej konieczności. Które urządzenia wymagają obsługi zapobiegawczej, aby przedłużyć ich resurs, a które wystarczy wstawiać do planu przeglądów „na najbliższą okazję”. Optymalny dobór planów obsługi może być źródłem istotnych oszczędności.

Trzeci punkt programu kompleksowego produktywnego utrzymania to podział zadań. Nie wszystkie zabiegi konserwatorskie muszą być wykonywane przez wykwalifikowany personel obsługi. Na przykład toner w drukarce może wymienić sam Użytkownik, jeśli będzie miał zapas pod ręką i w razie potrzeby zostanie przeszkolony. Może także sam sobie poradzić z zawieszoną aplikacją, jeśli polecenie ALT zasugeruje mu konsultant przez telefon :).

Wreszcie kwestia organizacji pracy. Chodzi nie tylko o wyposażenie informatyka krążącego po piętrach biurowca. Chodzi także o optymalizację jego marszrut, aby zaoszczędzić mu wysiłku. Najlepiej kilka podstawowych marszrut zestandaryzować i wtedy łatwiejsze będzie planowanie dobrego wykorzystania jego czasu pracy. Standaryzacja pozwala także na równoważenie pracy i wyrównywanie obciążeń poszczególnych pracowników. Jeśli Użytkownicy narzekają na zbyt długi czas do naprawy, warto pomyśleć o wyrównywaniu procesu. Można zacząć od wytworzenia nawyku, że w pierwszej kolejności wykonujemy zadania najkrótsze, a zadania zabierające najwięcej czasu wykonujemy „w międzyczasie” - odwrotnie niż nakazuje intuicja!

Zdaję sobie sprawę, że powyższy program produktywnego utrzymania brzmi zupełnie inaczej, niż zalecenia zawarte w zbiorach najlepszych praktyk w rodzaju ITIL. No cóż, tradycyjna inżynieria różni się zarówno terminologią jak i doświadczeniami od inżynierii najnowszej. Ale sprawdza się w praktyce, od ponad stu lat.