Katastrofa

Katastrofa

Nie bójmy się tego słowa. To, co zdarzyło się z systemem informatycznym Instytucji, to była – patrząc na konsekwencje – nie usterka, nie awaria, ale właśnie k a t a s t r o f a : zdarzenie, którego zawsze bardzo trzeba unikać. Dlatego oprogramowanie sterujące urządzeniami, gdzie *awaria* łatwo może stać się *katastrofą*: samolotami, sprzętem medycznym, hamulcami w samochodach i elektrowniami atomowymi, buduje się z wielką dbałością, według ściśle określonych procedur. Niestety, wobec systemów IT nie ma ani woli, ani wiedzy, ani obyczaju postępowania z n a l e ż y t ą s t a r a n n o ś c i ą , przeciwnie – powszechne jest niechlujstwo, niewiedza i dezynwoltura.

Katastrofa – czy nie przesadzam? W sensie technicznym, ten program przecież, jak mawiają często informatycy, „prawie działał", kulawo, ale jednak coś tam liczył. Owszem, ale konsekwencje społeczne i polityczne tej akurat kulawości były takie, że słowo „katastrofa" jest najzupełniej odpowiednie.

Siedem punktów opisujących, jak to można było zrobić dobrze:

1. Jasny cel przedsięwzięcia
2. Poprawne, pełne i precyzyjne wymagania
3. Zastosowanie metodyki przyrostowej
4. Informatyczne kompetencje zleceniodawcy
5. Testowanie, głupcze!
6. Koniec z manufakturą, czas na taśmę produkcyjną
7. Architektura, BPR, DevOps i inne takie

Kto zawinił?

Winny jest ten sam złoczyńca, co zawsze, we wszystkich projektach informatycznych, od lat. Znany jest on doskonale z listów gończych, wystawianych przez Instytut Gartnera i przez *The Standish Group*: „n i e k o m p l e t n e w y m a g a n i a , b r a k z a a n g a ż o w a n i a u ż y t k o w n i k ó w , b r a k z a s o b ó w i w s p a r c i a k i e r o w n i c t w a , n i e r e a l i s t y c z n e o c z e k i w a n i a o r a z p r o b l e m y z p l a n o w a n i e m ". Przyczyny i skutki tej afery nie różniły się niczym od setek, tysięcy podobnych wpadek i pół-wpadek, zwykle nieznanych nikomu, prócz uczestników, zapomnianych lub zatuszowanych, czasem wlokących się po sądach. Nie różniły się niczym, oprócz politycznego wymiaru, które z tej właśnie, dość typowej wpadki, uczyniły spektakularną katastrofę.

Kto jest winny, jeśli właściciel pięciopokojowego pokojowego mieszkania, niemający pojęcia o naprawianiu i malowaniu, usiłuje zamówić gruntowny remont trzy dni przed planowaną wizytą ważnych gości, o której wiedział już kilka miesięcy wcześniej, ale sprawę odwlekał? W dodatku, nie

postarał się, by mieć na remont dość pieniędzy? Firma budowlana, postawiona przed takim niemal niewykonalnym zadaniem, może odmówić, albo zgoła próbować nagłośnić sprawę w mediach „uważajcie na tego szaleńca, nie przyjmujcie jego obłąkanych zleceń!". Może, ale pieniądze za wykonanie zlecenia kuszą, jest nadzieja, że może się uda tak, jak udało się poprzednio. Firma, mająca w tej sprawie ten grzech na swoim sumieniu, jak widać - polecam wizytę na jej stronie - wyspecjalizowała się w zamówieniach publicznych, stwarzających chronicznie przesłanki do identycznych wpadek. Gdyby, zamiast mieć nadzieję, że i tym razem się uda, Firma pobiegła do ABW lub do gazet z doniesieniem, że Instytucja oszalała, to wprawdzie poszłaby do nieba, ale pewnie żadnego więcej zamówienia od żadnej instytucji już by nie dostała. Nie tylko od instytucji zresztą - prywatni też niechętnie widzieliby, żeby firmy informatyczne, na zlecenia wykonujące programy wspierające ich kulawe i źle skonstruowane procesy biznesowe, chwaliły się publicznie tym, jaki nieudolny klient im się trafił…

Jak unikać katastrof

Wiedza o tym, jak dobrze realizować przedsięwzięcia IT i unikać podobnych katastrof, jest łatwo dostępna: jest w licznych opracowaniach, w książkach i w głowach wielu mądrych ludzi, specjalistów inżynierii oprogramowania. Niestety, często nie ma jej tam, gdzie zapadają decyzje, ani tam, gdzie realizuje się projekty. Bywa, że nie ma jej ani w głowach biznesowych zleceniodawców, ani w głowach wykonawców projektów. Instytucja, składając SIWZ w tej formie i w tym terminie, wydała sobie świadectwo sobie zupełnego braku wiedzy. Co do stanu świadomości wykonawcy, wymowna jest zarówno treść komentarzy na temat błędów programistycznych jego kodu, jak i zawartość ich ogłoszeń rekrutacyjnych: nie szukają analityka ani projektanta, lecz… kolejnego programisty! Sprawdźcie sami na stronie Firma/praca.

Siedem punktów opisujących, jak to należało zrobić dobrze…

… i o czym trzeba odtąd pamiętać we wszystkich przyszłych projektach, aby unikać katastrof, a czego nie zrobił ani zleceniodawca, ani wykonawca systemu wyborczego.

1. Jasny cel przedsięwzięcia

Konieczne jest, aby zleceniodawca miał jasno określony cel, w firmach zwany „celem biznesowym". Gdyby celem Instytucji było naprawdę dobre zrealizowanie zadania, sprawne i minimalizujące prawdopodobieństwo niepowodzenia, wtedy ryzyko całej afery zmalałoby dramatycznie. Na pewno nie doszłoby do aż tak karygodnie późnego ogłoszenia przetargu, bo niepokój przed wpadką nie dawałby jej członkom zasnąć już dwa lata wcześniej. Nie mam oczywiście pewności, muszę domniemywać, ale zapewne naczelnym celem osób w Instytucji – jak jest w wielu organizacjach publicznych – jest zupełnie co innego: święty spokój i szacowne stanowisko, więc nie trudzą się formułowaniem innych celów. Decyzja o zamówieniu systemu IT była więc najpierw – dla świętego spokoju – odwlekana, a potem podjęta bez planów awaryjnych, na szybko, bo „przecież musimy mieć jakieś komputery". Myśląc kryteriami celu nadrzędnego, zamiast prestiżem, zapewne Instytucja jaśniej i bez wewnętrznego oporu widziałaby też własną niekompetencję i np. zamówiła stosowne ekspertyzy – zawczasu.

2. Poprawne, pełne i precyzyjne wymagania

Algorytm nie jest zbyt zawiły i jego realizacja była podobno poprawna, co wykazały m.in. testy algorytmu, wykonane przez niezależną firmę.

Natomiast, co powinien wiedzieć każdy liczący się interesariusz projektu IT, algorytm to nie wszystko; równie ważne lub nawet ważniejsze są tak zwane wymagania poza-funkcjonalne, takie jak stabilność, wydajność pod obciążeniem, łatwość obsługi, niezawodność i bezpieczeństwo systemu. Tego nie testowano, mimo doświadczeń sprzed 12 lat (wybory samorządowe 2002), co znaczy, że nikomu nawet nie przyszło do głowy określić potrzebnych wymagań.

Krok pomiędzy celem i wizją z jednej strony, a szczegółowymi wymaganiami z drugiej strony, to określenie – wybaczcie mi szczególną terminologię inżynierii oprogramowania – *interesariuszy* oraz *kontekstu* systemu. Z ważnymi interesariuszami trzeba porozmawiać, aby poznać ich wymagania dotyczące systemu. Taka praca, nawet pobieżnie zrobiona, pozwoliłaby nieszczęsnemu tandemowi Instytucja / Firma dowiedzieć się tego, czego sami nie wiedzieli, lub nie chcieli wiedzieć, między innymi od fachowców, jak i od członków komisji wyborczych, od prawników, och polityków, od wyborców wreszcie.

> W y m a g a n i a , to pięta achillesowa wszystkich projektów wykonywanych zgodnie z prawem o zamówieniach publicznych. Podkreślam, co już wcześniej pisałem na moim blogu, że nie jest to żadna wada tego prawa, które przecież nie ma być alternatywnym dla PRINCE 2 czy PMI standardem prowadzenia projektów, tylko metodą utrudniania korupcji. Ani to prawo, ani żadne inne, nie jest w stanie przeciwdziałać niekompetencji, cwaniactwu i niechlujstwu.
>
> Powtarzany bezmyślnie i do znużenia, zarzut wobec tego prawa, że premiuje kryterium ceny, nie wytrzymuje krytyki. Po pierwsze, ostatnia aktualizacja tego prawa sugeruje stosowanie również kryterium jakości, a po drugie, jeśliby specyfikacja wymagań – SIWZ – była dostatecznie precyzyjna, określając dokładnie i przedmiot zamówienia, i warunki jego odbioru, w tym jakość, wówczas kryterium ceny byłoby zupełnie wystarczające! Oczywiście, to stawia wymagania wobec zleceniodawcy, którym zwykle nie jest on w stanie podołać, ale ten problem pojawia się równie często w umowach na budowę systemów, których prawo o zamówieniach publicznych nie dotyczy.

Ani Instytucja, ani Firma, nie są, oczywiście, kompetentne w zakresie inżynierii wymagań. Prostym, ale niestety bardzo rzadko stosowanym rozwiązaniem, jest zdobycie potrzebnych kompetencji poprzez szkolenia, lub powierzenie tego zadania – napisania SIWZ-u – zewnętrznej firmie, mającej potrzebne kwalifikacje.

> Z ostatniej chwili: udało mi się znaleźć informację o tym, że począwszy od lutego 2013, prawo zamówień publicznych, zezwala na stosowanie procedury tak zwanego dialogu technicznego *„zwracając się o doradztwo lub udzielenie informacji w zakresie niezbędnym do przygotowania opisu przedmiotu zamówienia, specyfikacji istotnych warunków zamówienia (SIWZ) lub określenia warunków umowy.”* (artykuł 31a Pzp). To znakomita wiadomość, umożliwiająca – wreszcie! – stosowanie najlepszych praktyk inżynierii wymagań już w trakcie przygotowywania SIWZ! Oczywiście, to wymaga woli, wiedzy i czasu, no i prawnicy powinni o tej możliwości masowo nas informować, a nie chować ją w BIP-ach i innych miejscach, gdzie nikt poza nimi raczej nie zagląda…

3. Zastosowanie metodyki przyrostowej

Co najmniej od lat 70-ych, kiedy Tom Gilb opublikował zasady swojej metodyki ewolucyjnego wytwarzania oprogramowania (zob. wywiad z Tomem Gilbem „Projekt bez zbędnych opóźnień", Computerworld 22.03.2011), wiadomo, że oprogramowanie warto, kiedy tylko jest to możliwe, tworzyć po kawałku, stopniowo, zamiast usiłować w dramatycznych okolicznościach wodować całość, niczym jakiś wielki okręt. Już w latach 90-ych to podejście znacznie się rozpowszechniło (m.in. znany proces RUP i technika programowania ekstremalnego), a dzisiaj, wobec wzrostu popularności agile, staje się wręcz większościowe.

Dzięki podejściu przyrostowemu (ang. „incremental"), system można testować – po kawałku – od samego początku, więc wszelkie jego braki poznawać znacznie wcześniej, kiedy jeszcze jest czas im przeciwdziałać. Jednak w sytuacji krytycznej, kiedy strach przed spóźnieniem się tłumi rozsądek, często wygrywa pokusa sposobu pracy w stylu „big bang" w nadziei, że się uda.

Zastosowanie podejścia przyrostowego w opisywanym przypadku nie spowodowałoby wprawdzie magicznej przemiany trzech miesięcy w potrzebny, dłuższy czas, ale zapewne pozwoliłoby wcześniej zidentyfikować słabości tego oprogramowania i wdrożenie rozmaitych działań zaradczych, ograniczających szkody.

Spotkałem się z pytaniem, czy metoda agile nie byłaby czasem ratunkiem przed katastrofą? Zanim odpowiem, muszę wyjaśnić, czym naprawdę jest agile, bo to słowo stało się w IT ostatnio tak modne, że zatraciło ostrość znaczenia.

Agile ma cztery cechy:

(1) tworzenie przyrostowe – już opisane powyżej;

(2) podejście iteracyjne – stopniowe doprecyzowywanie niepełnych i niejasnych z początku wymagań, w ścisłej współpracy z aktywnie współdziałającym klientem;

(3) wykonywanie pracy w małych, samoorganizujących się zespołach;

(4) dynamiczne planowanie – szczegóły harmonogramu poznaje się dopiero w trakcie projektu.

Korzyść – częściową - z zastosowania podejścia przyrostowego już potwierdziłem powyżej. Podejście iteracyjne nie było potrzebne, gdyż nie było potrzeby doprecyzowania niejasnych wymagań, a jedynie ich uzyskanie, do czego iteracje nie byłyby pomocne. Ani stosowanie samoorganizujących się zespołów, ani dynamiczne harmonogramowanie, też niczego by nie zmieniły. Agile, to dobra metodyka, ale nie jest w stanie powstrzymać zdeterminowanych samobójców. Na dobitkę, agile zakłada dobrą współpracę wykonawcy z aktywnym klientem – zleceniodawcą, czyli z Instytucją – tym razem to było marzenie ściętej głowy.

4. Informatyczne kompetencje zleceniodawcy

W swojej genialnej, zupełnie niedocenionej przez naszych szefów i dyrektorów, książce „Skokowy wzrost wydajności" (CeDeWu 2011), Adam Kolawa, twórca firmy Parasoft, napisał, że warunkiem sukcesu biznesu, korzystającego z IT, jest zrozumienie możliwości IT przez kadrę zarządzającą.

Nie chodzi o to, by dyrektor zarządzający i dyrektor marketingu każdej firmy byli programistami, lecz o to, by bardzo dobrze zdawali sobie sprawę z możliwości IT, oraz wymogów, jakie firmie stawia zamiar jej skutecznego wykorzystania.

Tak jest – dobiegają, na szczęście, końca czasy aroganckich, wąsatych kapitalistów-sarmatów, osiągających sukcesy samym cwaniactwem, oraz leśnych dziadków-prawników, dla których komputer „to, panie, takie nowomodne dziwactwo, ale trzeba je mieć". Szkoda, że konwulsje tej odchodzącej formacji akurat dotknęły obszar tak newralgiczny dla naszej demokracji, jak wybory samorządowe.

5. Testowanie, głupcze!

Od początku XXI wieku, kiedy autor tego artykułu przyniósł do Polski (2002) nowopowstały system certyfikacji dla testerów ISEB/ISTQB, testowanie przeżywa prawdziwy renesans. Niestety, dzieje się on głównie na terenach wewnątrz IT, natomiast zleceniodawcy, biznes, nadal wydają się nie bardzo rozumieć, o co w tym wszystkim chodzi. Rzeczywistość szkoleń w tej dziedzinie potwierdza to: uczniami są zwykle bardzo młodzi w y k o n a w c y , rzadziej projektanci testów, natomiast ich szefowie w tym czasie robią certyfikacje ITIL, PMI czy PRINCE 2, w zakresie testów, mówiąc delikatnie, nie do końca staranne. Zaś capo di tutti capi, absolwenci MBA i guru zarządzania, nadal – jak czterdzieści lat temu! – widzą testowanie, jako jakieś dziwactwo nadmiernie ostrożnych cieniasów. Brednie, które na temat testów, zwanych naprzemiennie „audytem" albo „sprawdzaniem", przy okazji tej afery, czytaliśmy w prasie, potwierdzają tę smutną diagnozę. Żeby sygnał był jeszcze wyraźniejszy, afera zbiegła się w czasie z żałosnymi wysiłkami PKP, aby zapanować nad rozjeżdżaniem się swego systemu pod wpływem kilku zmian. Może taki zdublowany sygnał do kogoś dotrze, i w przyszłości testowanie będzie się częściej wykonywać *przed* wdrożeniem, a nie dopiero po nim?

6. Koniec z chaosem, czas na taśmę produkcyjną

Wiem, to niepopularne twierdzenie w czasach gwałtownej mody na kreatywność i samorealizację, ale t a ś m a p r o d u k c y j n a n a p r a w d ę j e s t ś w i e t n y m w y n a l a z k i e m . Jej pojawienie się na przełomie XVIII i XIX wieku stało się cezurą, oddzielającą szczęśliwe czasy powszechnego ubóstwa od smutnej współczesności masowego materialnego dostatku. Dobrze stosowana, nie stoi na przeszkodzie kreatywności, przeciwnie – jeśli podstawowy proces produkcyjny toczy się bezproblemowo, mamy więcej czasu na twórcze uniesienia, zamiast marnować go na gaszenie pożarów i naprawianie licznych, głupich defektów.

IT ma szczególne problemy z zaakceptowaniem tej prostej prawdy. Dyskusje na temat jakości kodu, jakie afera systemu Instytucji gwałtownie rozpaliła, kręcą się wokół tematyki, którą doskonale pamiętam z lat studiów – stylu programowania. To bardzo ważny aspekt – mieć kiepski kod źródłowy programu, to jakby budować wielką konstrukcję ze spróchniałego drewna – tyle, że są na to doskonale fachowcom znane metody, chronicznie zupełnie lekceważone przez przemysł IT: analiza statyczna, testy jednostkowe, pomiary pokrycia kodu… Redakcja prosiła, żeby nie pisać zbyt technicznie, więc muszę przerwać – zapraszam na korepetycje!

7. Architektura, BPR, DevOps i inne takie

W dyskusji wokół katastrofy razi, choć już nie dziwi, nagminne, błędne używanie słowa „informatyk" w znaczeniu „programista" oraz przekonanie zarówno Instytucji, jak i setek anonimowych dyskutantów na forach, że to „programiści" budują oprogramowanie i odpowiadają za jego jakość. Prawda jest inna: programiści tworzą oprogramowanie na tyle, na ile murarze, układający cegły, budują domy. Dom, zbudowany wyłącznie przez murarzy, nazywa się lepianką, a oprogramowanie zbudowane bez udziału architektów, przypomina swoją konstrukcją taką właśnie ogromną,

nadmuchaną i niezbyt stabilną, lepiankę. Nie znam architektury programów, który zawiódł, ale nie słyszę w dyskusji żadnych na ten temat głosów, co źle świadczy o informatycznej dojrzałości ogółu.

Zamykam więc punkt siódmy – stosowanie i odpowiednie projektowanie architektury oprogramowania.

Zapewniam, że stosowanie – solidne stosowanie - opisanych tutaj pokrótce s i e d m i u z a s a d praktycznie gwarantuje brak tego typu afer w przyszłości.