Zmiana prosta jak ameba, a jednak milion kroków – krótki apel IT do biznesu

Współczynnik niepowodzenia projektów IT, w szczególności projektów dostosowywanych do wymagań zamawiającego, w większości organizacji wynosi około 70%. Znaczna część z nich kończy się niepowodzeniem lub opóźnieniami czy niższą rentownością. Następstwem tego jest przekraczany budżet projektowy o ok. 27%, a w co szóstym projekcie koszty rosną nawet o 200% (Źródło:www.learn.g2.com). Chcąc uniknąć tego typu problemów, dostawcy oprogramowania korzystają z gotowych komponentów oraz platform narzędziowych zwiększających efektywność prac.

Jakub GŁĄB
VSoft

Zdjęcia: VSoft

Michał JURKOWSKI
VSoft

Na przestrzeni lat wiedza dotycząca planowania i implementacji, a także doświadczenie dostawców wzrosły. Wypracowane zostały nowe metodologie mające pomóc w skutecznym realizowaniu projektów. W tym samym czasie rozmiar systemów informatycznych oraz poziom ich skomplikowania – zarówno z punktu technicznego, merytorycznego, jak i organizacyjnego – nieporównywalnie wzrósł. Systemy o charakterze technicznie monolitycznym, dziś, dzięki łączeniu kilkudziesięciu technologii oraz milionom linii kodu, stają się systemami o charakterze rozporoszonym. Kiedyś rozwiązania technologiczne tworzone były przez kilka osób o bardzo zbliżonym zakresie kompetencji, dzisiaj realizowane są przez zespoły inżynierów specjalizujących się w poszczególnych dziedzinach.
Następujące wraz z rozwojem technologii zmiany wpływają bezpośrednio na metodologie realizacji projektów, sprawiając, że zespoły odpowiedzialne za ich prowadzenie podejmują szereg działań zapobiegających ich niepowodzeniu. Niestety, często, nawet pomimo tych działań, wciąż nie udaje się znacząco poprawić statystyk. Co leży u podstaw problemu? Rozważmy hipotetyczny przykład.

Wskazówka godzin

Czy zegarek z prostą tarczą, dwiema wskazówkami i dwunastoma kropkami jest tak prostym mechanizmem, na jaki wygląda? Czy zamiana wskazówki godzin na podobną, różniącą się jedynie stylistką, na pewno okaże się trywialnym zadaniem?
Przyjmijmy, iż w systemie klasy enterprise, działającym w dowolnej domenie biznesowej, pojawia się nowe wymaganie sugerujące, by użytkownik mógł – w określonym kontekście – wprowadzić i zapisać dodatkową, pojedynczą informację biznesową (np. NIP). Z punktu widzenia użytkownika zmiana jest prosta: na jednym z formularzy pojawia się nowe pole do uzupełnienia w określony sposób. Proste! Dla zespołu inżynierów, oczywiście w uproszczeniu, proces implementacji nowej funkcjonalności wygląda następująco:

- W warstwie bazy danych, w jednej z setek lub tysięcy tabel musi zostać dodana kolumna z określonym typem. Z każdą kolumną związany jest szereg elementów, które potencjalnie również powinny zostać zmodyfikowane (indeksy, widoki etc.).
- Musimy pamiętać, że w wielu bazach danych przechowywane są także tzw. procedury składowane – kod źródłowy reprezentujący określoną logikę. W wielu przypadkach nawet dla tak prostej zmiany musi on zostać zmieniony.
- We współczesnych systemach struktura bazy danych bardzo często reprezentowana jest w postaci kodu obiektowego w wyższej warstwie systemu (inna technologia, inny język programowania, inne narzędzia). Musi ona zostać ręcznie lub automatycznie uzupełniona.
- Za odpowiednią interpretację nowej informacji w systemie najczęściej odpowiada warstwa logiki biznesowej (np. ustawienie statusu oferty adekwatnie do wprowadzonego pola NIP), która również musi zostać zmodyfikowana lub uzupełniona o nowe algorytmy.
- Pamiętajmy, że dokonanie powyższych zmian powinno zostać poprzedzone implementacją tzw. testów jednostkowych (logika odpowiedzialna za weryfikację nowej logiki).
- Nowa funkcjonalność wymaga także zmian w interfejsie użytkownika. W wielu przypadkach projektant UX zadecyduje o konieczności dokonania szerszych modyfikacji formularza (przesunięcie, przegrupowanie itp. pozostałych elementów interfejsu).
- Implementacja zmian w interfejsie użytkownika to kolejny zestaw modyfikacji kodu źródłowego, najczęściej w zupełnie innej technologii i narzędziach. Wymagana będzie co najmniej logika weryfikująca prawidłowość wprowadzonych danych oraz zachowania zależnych ele-


Fot. Gorodenkoff/stock.adobe.com

mentów interfejsu (np. część innych pól pozostaje nieaktywne do momentu wprowadzenia wartości NIP).

- Do tego momentu w nasze działania zaangażowaliśmy już: programistę baz danych, programistę backendu, frontendu oraz projektanta UX. Pamiętajmy, ta **prosta zmiana** dotyczy jedynie dodania nowego okna w formularzu.
- Nie zapominajmy, że większość systemów posiada wersję mobilną. Szereg zmian (nierzadko w kilku warstwach systemu) musi zostać dokonanych dodatkowo, by spełnić wymagania innej platformy. Wykonanie nawet takiej zmiany angażuje programistę specjalizującego się w systemie typu Android czy iOS.
- Kolejną osobą zaangażowaną w projekt jest tester automatyzujący. Wszystkie opisane powyżej zmiany muszą zostać uwzględnione w kodzie odpowiedzialnym za automatyzację testów.
- Implementacja nowej, choćby najmniejszej, zmiany w systemie musi zostać odpowiednio „opakowana", by pozwolić na bezawaryjne wdrożenie systemu w środowisku produkcyjnym. Wdrożenie zmian polega na instalacji różnych komponentów systemu na różnych, często fizycznie oddzielnych serwerach.
- Na koniec pozostaje aktualizacja dokumentacji, które są częścią paczki wdrożeniowej.

Pomimo przemyślanej architektury, opisana wyżej, uproszczona implementacja trywialnego wymagania biznesowego oznacza konieczność dokonania kilkudziesięciu zmian w systemie naczyń połączonych, realizowanych najczęściej przez co najmniej kilku inżynierów różnych specjalizacji. Każda z nich generuje potencjalne ryzyko wystąpienia przyszłego błędu oraz napotkania na nieprzewidziane problemy, które wpłyną na czas lub\i proces jej implementacji.

Rzeczywistość pokazuje, iż pomimo doświadczenia i wiedzy zespoły projektowe posiadają bardzo silną tendencję do przeceniania swych możliwości, prostoty rozwiązania i wynikającego z nich nadmiernego optymizmu i presji czasu, bo przecież musi być szybki time to market. Bez względu na stosowaną metodologię prowadzenia projektów, implementacji zmian w systemie, rodzaju podpisanej umowy, ograniczeń biznesowych oraz pozostałych czynników otoczenia, kluczowym wydaje się przestrzeganie kilku zasad ogólnych.

Festina lente

Łacińskie *„śpiesz się powoli"* bardzo trafnie oddaje to, o czym zdajemy się często zapominać, planując i prowadząc projekty informatyczne. Nie ulega wątpliwości, że poziom złożoności systemów IT już dawno przerósł możliwości objęcia całości przez pojedynczego człowieka, a nawet proste zmiany wiążą się z niemałym ryzykiem wystąpienie defektów.

Wydaje się więc intuicyjnie uzasadnione, by implementując kolejne zmiany w systemach informatycznych, robić to ostrożnie – minimalizować zmiany, nie nawarstwiać ich i ciągle weryfikować ich zasadność i powodzenie. **Mniej znaczy więcej!** Złożoność nas przeraża i kosztuje na start. Dlatego nie tylko powinniśmy dążyć do implementowania nowych funkcjonalności w weryfikowalnych i wdrażanych na bieżąco porcjach, lecz także do redukowania ich liczby i złożoności. Zwykle rzeczywistość weryfikuje zawiłe rozwiązania już na początku procesu implementacji pokazując, że dalsze kroki nie mogą zostać podjęte zgodnie z oryginalnymi założeniami, nie mówiąc o MVP (ang. minimum viable product).

Wbrew pozorom – powinniśmy przyjąć postawę dokładnie odwrotną – starać się szukać minimalistycznych i możliwie najmniej złożonych, szybko wdrażanych rozwiązań. Powinniśmy uczulać analityków, że w dużym stopniu ich rola nie polega na dogłębnym zrozumienia każdego szczegółu procesu biznesowego i stworzeniu lustrzanego odbicia w systemie. Równie lub bardziej istotne jest dążenie do prostoty i zapewnienia szybkiej „konsumpcji" wartości dodanej (wdrożenia produkcyjnego). Dodatkowo okazuje się, iż w przeważającej liczbie systemów informatycznych znaczna część funkcjonalności nigdy nie jest używana. Zamawiający nie wierzą dostawcy, który twierdzi że tzw. fajerwerki nie są sexy. A przecież dostawca też patrzy na wartość biznesową – sukces klienta to też nasz sukces!

A może „co" – nie „jak"?

Definiując wymagania dla systemu informatycznego, zastanawiamy się, jak dany proces biznesowy należy „przełożyć" na nowy system lub zmianę w istniejącym. Bardzo rzadko natomiast poważnie rozważamy zmianę samego procesu, by zminimalizować lub uniknąć modyfikacji w systemie. To błąd. Ostatecznie chodzi przecież, o maksymalne „zgranie" możliwości systemu z procesami biznesowymi – mamy do dyspozycji dwa czynniki zmienne, nie jeden. Możemy zmieniać system lub proces. Niezwykle często relatywnie prostą zmianę w procesie biznesowym zamieniamy na skomplikowaną, drogą i ryzykowną zmianę w systemie informatycznym.

A co na to rzeczywistość?

Zapotrzebowanie na systemy informatyczne rośnie systematycznie z roku na rok. Zdolności wytwórcze również rosną, jednak wolno, stając się jednocześnie coraz droższe. Rozwiązaniem są platformy narzędziowe **low code** pozwalające szybciej reagować na potrzeby rynku. Decydując się na nie, należy mieć na uwadze wszystkie ograniczenia opisane powyżej: architektura, presja czasu, oderwane od rzeczywistości wymagania i ciągły brak rąk do pracy. Należy także zwrócić uwagę, że za pomocą nowoczesnych narzędzi nie da się zrobić wszystkiego, natomiast można szybko dostarczyć wartość biznesową, uruchomić projekt produkcyjnie, zarabiać na nim, analizować dalsze potrzeby rozwojowe i specyficzne wymagania, dobrze planować i realizować w zaplanowanym harmonogramie.

Manifest

Specjalizacje IT są coraz bardziej popularne i coraz lepiej wynagradzane, pamiętajmy jednak, by nie dać się zwariować. Rynek stale rośnie, a ludzi do pracy nad projektami brakuje. Szacuje się że w 2020 r. na rynku w Europie będzie brakowało nawet 600 tys. programistów różnej specjalizacji. Rozwiązaniem może być ciągłe zatrudnianie – kto obecnie rekrutuje, ten wie, z jakimi problemami się to wiąże. Innym sposobem może być wparcie przez proste w obsłudze i pozwalające na szybką integrację narzędzia ułatwiające tworzenie systemów nie tylko programistom, ale także osobom nieposiadającym wiedzy technicznej, np. projektantom procesów. ■