SZ@FiR - *internetowy system zarządzania firmą, czyli jak wytwarzać oprogramowanie na wynajem*

Robert Zyskowski
Dział Projektów Internetowych PROKOM Software SA
81-319 Gdynia, ul. Śląska 23/25
e-mail: zyskowskir@prokom.pl

Streszczenie

Artykuł przedstawia praktyczne doświadczenia autora zdobyte w trakcie kierowania wytworzeniem internetowego sytemu zarządzania firmą SZ@FiR. Ponieważ usługi ASP są ciągle jeszcze pewnym novum na polskim rynku, we wstępie omówione zostały zasady wynajmu aplikacji oraz wynikające z tego korzyści i problemy. W artykule omówiona jest ogólna koncepcja systemu SZ@Fir, jego przeznaczenie, struktura i zastosowane technologie. Następne rozdziały zawierają opis praktycznej implementacji ramy pracy Microsoft Solution Framework, zastosowanej przy wytwarzaniu systemu - scharakteryzowany został proces wytwórczy oraz właściwa dla MSF koncepcja zarządzania zespołem realizacyjnym o strukturze skorelowanej z tym procesem.

1. Wstęp

Mianem Application Services Provider (ASP) określa się zwykle firmę usługową, która wynajmuje oprogramowanie za pośrednictwem sieci - Internetu lub łączy dedykowanych. Klient korzysta z programów fizycznie zainstalowanych w centrum danych należącym do ASP. Dzięki temu zostaje on uwolniony od obowiązków związanych z utrzymaniem oprogramowania, zarządzaniem i aktualizacją, archiwizacją danych, wsparciem technicznym. Klient i ASP rozliczają się w ustalonym czasie, przy czym w kontrakcie zazwyczaj określony jest minimalny okres użytkowania dzierżawionej aplikacji.

Rozwojowi rynku usług ASP sprzyja rozwój technologii - rośnie złożoność aplikacji, poprawia się przepustowość łączy. Sprzyja też rynek. Przedsiębiorstwa -

potencjalni klienci ASP - odczuwają coraz silniejszą presję w związku z rozwojem handlu elektronicznego. Fuzje i przejęcia, wymagające ustawicznego przekształcania systemów informatycznych, stawiają pod znakiem zapytania sens długofalowych inwestycji. Od pewnego czasu wzrasta więc aprobata modelu, w którym informatyką zarządza firma zewnętrzna.

W modelu ASP udostępniane mogą być te aplikacje, których zadaniem nie jest zaspokajanie podstawowych potrzeb przedsiębiorstwa. W USA, gdzie działają pierwsze firmy ASP, najczęściej wynajmuje się programy wspomagające zarządzanie kadrami, kontaktami z klientami, łańcuchem logistycznym, dystrybucją. Ciekawą ofertą jest wynajmowanie aplikacji biurowych, które wymagają częstych aktualizacji.

Dla klientów największą korzyścią z wynajmu aplikacji może być rozłożenie kosztów w czasie. Kupno i uruchomienie własnego oprogramowania na własnym sprzęcie wymaga wysokich nakładów, co jest szczególnie dotkliwe dla niewielkich firm o ograniczonych zasobach. Stałe koszty, które są ponoszone w związku z amortyzacją sprzętu i oprogramowania niezależnie od stopnia jego użytkowania, mogą być przekształcone w koszty zmieniające się w zależności od rzeczywistych, bieżących potrzeb przedsiębiorstwa. Struktura kosztów staje się przewidywalna.

1.1. Problemy

Istnieje również kilka argumentów, które przemawiają przeciwko korzystaniu z ASP. Przede wszystkim oddanie kontroli nad aplikacjami i danymi zewnętrznej firmie jest powszechnie uważane za rozwiązanie dość ryzykowne. Polscy przedsiębiorcy niechętnie myślą o oddaniu kontroli nad jakąkolwiek częścią firmy zewnętrznemu, wyspecjalizowanemu usługodawcy. Wolą sami sprawować kontrolę nad całością, niezależnie od logiki kosztowej. Przedsiębiorcy decydują się często na założenie przysłowiowej kłódki na swojej serwerowni, która, choć niczego nie zabezpiecza, daje poczucie psychicznego komfortu.

Potencjalni usługodawcy są zdania, że klienci częstokroć nie zdają sobie sprawy, jak wysokie są koszty samodzielnego utrzymywania systemów informatycznych - ciągłych aktualizacji oprogramowania, konfiguracji nowych stanowisk, utrzymywania i wymiany sprzętu, archiwizacji, zatrudnienia zespołu informatyków itd. Dlatego trudno przekonywać do ASP tylko argumentami ekonomicznymi.

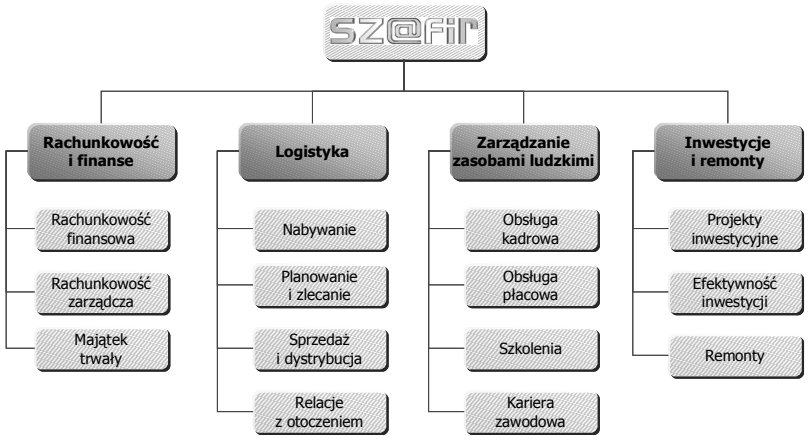
2. Koncepcja systemu SZ@FIR

SZ@FIR jest produktem adresowanym do małych i średnich firm. Firmy te często potrzebują nowoczesnych technologii do skutecznego konkurowania na rynku, natomiast nie są przygotowane do ponoszenia dużych kosztów związanych z informatyzacją. Rozwiązaniem dla firm tego typu może okazać się wynajmowanie aplikacji za pośrednictwem Internetu.

2.1. Struktura systemu

SZ@FIR jest rozwiązaniem standardowym, obsługującym podstawowe procesy biznesowe przedsiębiorstwa. W toku przeprowadzonych prac analitycznych został opracowany model biznesowy przedsiębiorstwa, na podstawie którego wyodrębniono 4 obszary funkcjonalne systemu:

- Finanse i rachunkowość
- Logistyka
- Zarządzanie zasobami ludzkimi
- Inwestycje i remonty.



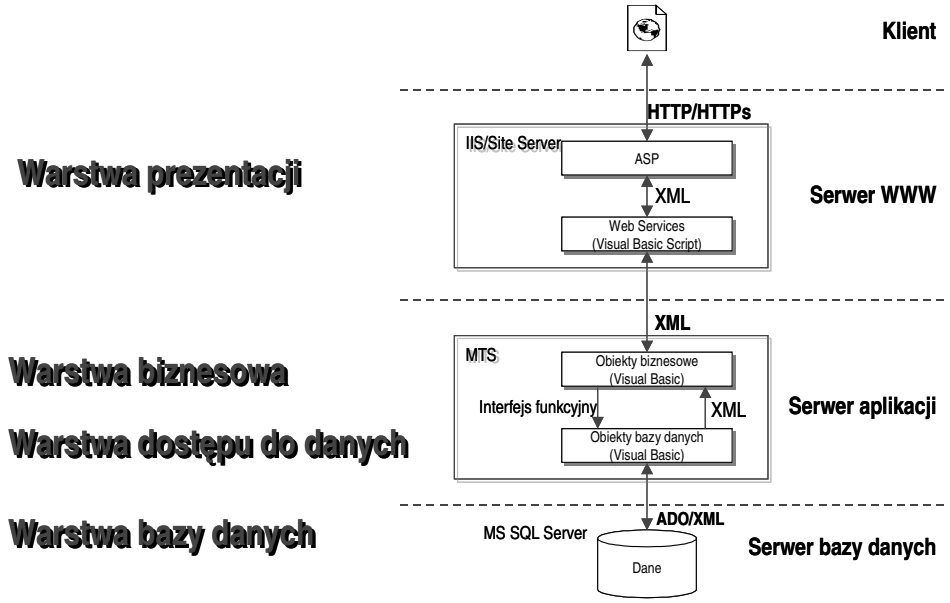
Rysunek 1. Struktura systemu

Wyodrębnienie w systemie obszarów funkcjonalnych (podsystemów, modułów) umożliwia użytkownikowi korzystanie tylko z tych elementów systemu, które są mu niezbędne. Jednocześnie, pomimo zastosowanego podziału, zachowano pełną integrację procesów zachodzących pomiędzy poszczególnymi obszarami. Dzięki możliwości zmiany parametrów konfiguracyjnych system

SZ@FIR może zostać przystosowany do potrzeb konkretnej branży czy odmiennego przebiegu procesu.

2.2. Architektura systemu

System został zaimplementowany w architekturze wielowarstwowej z wykorzystaniem **Microsoft Transaction Server** w połączeniu z **Internet Information Server**. Dostępność systemu na poszczególnych platformach systemowych jest uwarunkowana dostępnością ww. produktów.



Rysunek 2. Warstwy technologiczne

- Warstwa prezentacji to strony HTML, które obsługują całość interakcji z systemem. Za ich pośrednictwem odbywa się wprowadzanie danych, sprawdzanie ich poprawności oraz wyświetlanie żądanych informacji. Do wyświetlania stron HTML służy przeglądarka Internet Explorer w wersji 5.0 lub nowszej. Strony HTML są generowane na serwerze na podstawie plików Active Server Pages. Wygenerowane strony zawierają jedynie opis sposobu wyświetlania oraz sprawdzania poprawności wprowadzanych danych, natomiast nie zawierają samych danych. Dane te są ściągane z serwera po wyświetleniu strony w przeglądarce. Wymiana danych pomiędzy stronami HTML i serwerem odbywa się wyłącznie w formacie XML [5]. Językiem programowania warstwy prezentacji systemu jest Java Script oraz DHTML.

- Warstwa biznesowa systemu powstaje jako oprogramowanie komponentowe w oparciu o specyfikację COM+ dla środowiska Microsoft Windows. Komponenty te są umieszczone w plikach .dll na serwerze. Zadaniem warstwy biznesowej jest powtórne sprawdzenie poprawności otrzymanych danych, sprawdzenie uprawnień użytkownika, zamiana danych z formatu XML do postaci akceptowanej przez komponenty warstwy dostępu do danych, przekazanie danych do komponentów warstwy dostępu do danych, odebranie przetworzonych danych i odesłanie ich do warstwy prezentacji. Wymiana danych między warstwą prezentacyjną i biznesową następuje w formacie XML. Językiem programowania warstwy biznesowej systemu jest Visual Basic.
- Na warstwę dostępu do danych, podobnie jak w wypadku warstwy biznesowej, składają się komponenty umieszczone w plikach .dll na serwerze. Zadaniem tych komponentów jest przekazywanie danych pomiędzy warstwą biznesową i bazą danych. Komponenty te zajmują się wywoływaniem procedur składowanych, obsługą błędów specyficznych dla bazy danych (zamianą komunikatów o błędach bazy danych na komunikaty zrozumiałe dla użytkownika) oraz przekazywaniem przetworzonych danych do warstwy biznesowej.
- Warstwę bazy danych stanowi motor bazy danych. Dostęp do danych umieszczonych w tablicach odbywa się wyłącznie za pośrednictwem procedur składowanych w oparciu o standard ADO. System wykorzystuje motor baz danych MS SQL Server 2000.

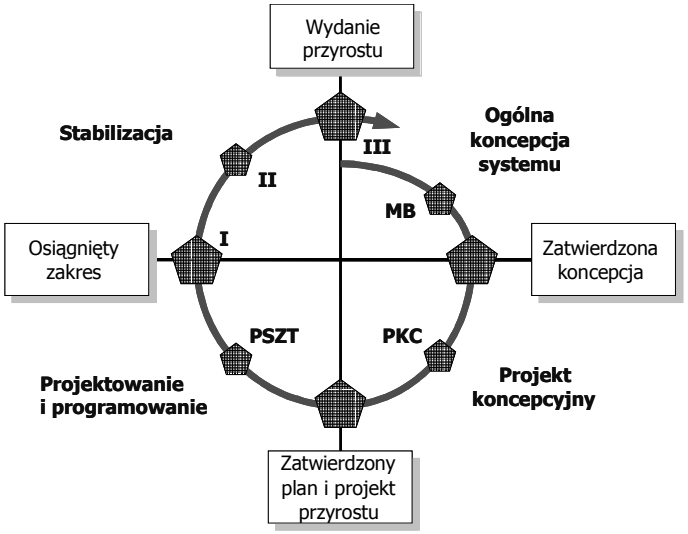
3. Proces wytwórczy

3.1. Cykl życia systemu

Sprawne zarządzanie procesem wytwarzania systemu wymaga zdefiniowania faz i etapów procesu oraz określenia produktów poszczególnych faz i etapów. Z tego względu istotne staje się wprowadzenie formalnych odbiorów produktów poszczególnych faz. Szczególnego znaczenia nabiera wdrożenie stosowania trybu rejestracji zmian w dokumentach projektowych, stosowanie wersjonowania oraz utrzymywanie jawnej listy kontrolnej osób odpowiedzialnych za poszczególne elementy produktów. Oczywiście, stosowanie rozbudowanej formalnej kontroli procesu wytwórczego zwiększa pracochłonność, co jest szczególnie widoczne w początkowej fazie rozwoju systemu. Jednakże wraz z rozwojem systemu formalna kontrola stopniowo traci znaczenie, gdyż powtarzane w kolejnych fazach i etapach czynności wytwórcze zaczynają mieć charakter rutynowy. Stosowanie standardów staje się elementem samodyscypliny zespołu

wytwórczego. Efekt autokorekty procesu wytwórczego jest szczególne widoczny w wypadku stosowania iteracyjnego, addytywnego cyklu życia systemu, zgodnie z którym system wytwarzany jest w kolejnych, kompletnych przyrostach jego funkcjonalności.

Celem pojedynczej iteracji jest zbudowanie modelu i zaimplementowanie kompletnego pod względem funkcjonalnym fragmentu systemu - przyrostu. Tak więc w ramach każdego przyrostu realizowany jest pełny cykl wytwórczy. Zachowanie spójności między przyrostami i ponowne wykorzystanie wyników poprzedniej pracy zachodzi automatycznie, gdyż operuje się ciągle na tym samym modelu, do którego dodawane są szczegóły specyficzne dla danego przyrostu. W różnicy między przyrostami zawarte są zarówno nowe funkcje, jak i efekty korekty błędów wykrytych w produkcie finalnym poprzedniego przyrostu.

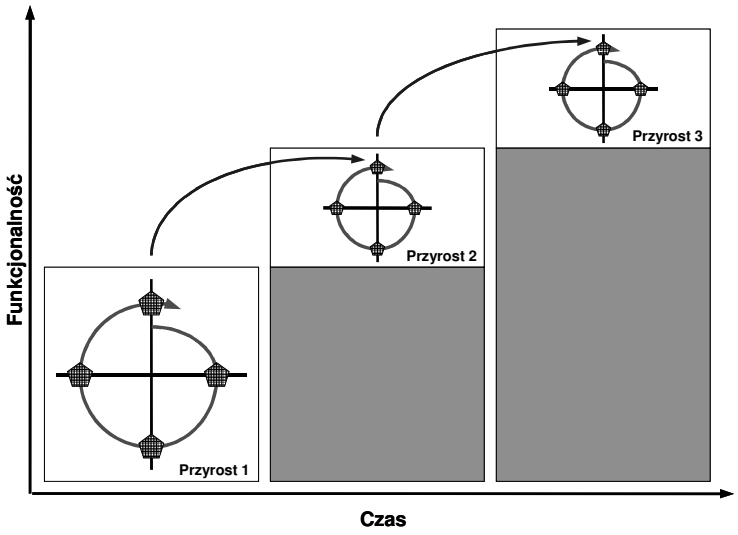


Rysunek 3. Przebieg pojedynczej iteracji[1]

Określenie zakresu przyrostu wymaga pewnej wprawy. W wypadku systemu SZ@FiR jest to ok. 40 przypadków użycia do zrealizowania w ciągu 4 miesięcy przez 20-osobowy zespół realizacyjny (kierownik zespołu analizy + 6 analityków, kierownik zespołu projektowo-implementacyjnego + 12 projektantów-programistów) przy korzystaniu z pomocy 5-osobowego zespołu dokumentacji, zaangażowanego w ok. 50%, oraz 6-osobowego zespołu testów, zaangażowanego w ok. 10%. Zdumiewająca jest także zgodność proporcji czasu trwania fazy wytwarzania do fazy stabilizacji z proporcjami prezentowanymi na kursie MSF (Microsoft Solution Framework). W wypadku zastosowanego do produkcji systemu SZ@Fir języka programowania Visual Basic proporcja ta

[1] MB – model biznesowy, PKC – projekt koncepcyjny, PSZT – projekt szczegółowy techniczo – technologiczny.

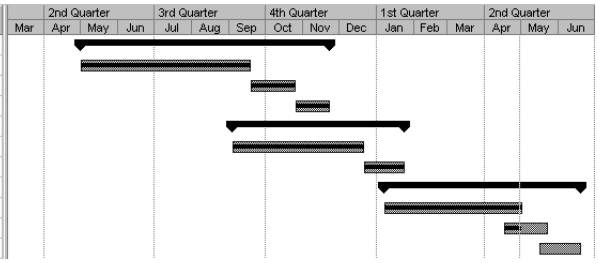
wynosi 2:1 (wytworzenie: stabilizacja). Dla porównania, w wypadku stosowania C++ proporcja ta wynosi 1:2, ale osiągana funkcjonalność biznesowa jest dużo mniejsza w porównaniu z osiąganą z zastosowaniem języka Visual Basic, przy założeniu analogicznych zasobów realizacyjnych.



Rysunek 4. Przyrostowy cykl życia systemu

Generalnie, wytworzenie przyrostu trwa 4 miesiące, stabilizacja 1,5-2 miesięcy, przy czym poszczególne przyrosty mogą się zazębiać. Całość obejmuje ok. 6 miesięcy, co średnio daje wydania co 4-5 miesięcy.

	Task Name	Duration	Start	Finish
1	**Przyrost 1**	**148 days**	**Mon 00-05-01**	**Wed 00-11-22**
2	I-wsze wydanie	101 days	Mon 00-05-01	Mon 00-09-18
3	II-gie wydanie	27 days	Tue 00-09-19	Wed 00-10-25
4	III-cie wydanie	20 days	Thu 00-10-26	Wed 00-11-22
5	**Przyrost 2**	**102 days**	**Mon 00-09-04**	**Tue 01-01-23**
6	I-wsze wydanie	79 days	Mon 00-09-04	Thu 00-12-21
7	II-gie wydanie	23 days	Fri 00-12-22	Tue 01-01-23
8	**Przyrost 3**	**117 days**	**Mon 01-01-08**	**Tue 01-06-19**
9	I-wsze wydanie	82 days	Mon 01-01-08	Tue 01-05-01
10	II-gie wydanie	26 days	Tue 01-04-17	Tue 01-05-22
11	III-cie wydanie	25 days	Wed 01-05-16	Tue 01-06-19



Rysunek 5. Przebieg wytwarzania systemu SZ@FIR

3.2. Modelowanie

Krytyczne znaczenia dla prawidłowego przebiegu wytwarzania systemu ma stosowanie formalnie zdefiniowanej metodyki jego modelowania. Tworzenie

modeli pozwala z jednej strony na precyzyjne wyrażanie zakresu i obowiązków systemów biznesowego i informatycznego z drugiej zaś na prowadzenie uzgodnień specyfikacji wymagań wobec systemu. Stosowanie wspólnego języka opisu z dobrze poznaną notacją i semantyką oraz większa precyzyjność języka formalnego w stosunku do języka naturalnego pozwalają na unikanie wieloznaczności specyfikacji. Stosowanie modeli pozwala na dokumentowanie zmian zarówno specyfikacji jak i projektów. Uzyskiwany efekt to brak zmian nieudokumentowanych oraz możliwość przeglądu rozwoju systemu poprzez kolejne wersje modeli, a także możliwość precyzyjnego planowania rozwoju systemu na poziomie wymagań biznesowych.

Przy wytwarzaniu systemów zarządzania firmą wybór metody analizy procesów biznesowych do opisania organizacji związany jest z potrzebą patrzenia na firmę poprzez pryzmat jej konkurencyjności. Dlatego też w przypadku systemu SZ@FiR zrezygnowano z analizy struktury funkcyjnej firmy, a skoncentrowano się na modelowaniu procesów występujący wewnątrz firmy. W szczególności procesów dostarczających wartość klientowi.

Jako praktyczną implementację metody analizy biznesowej wybrano Lynx 95 [3].

Wybór obiektowej metodyki projektowania [1][2] wynika w dużym stopniu z przyjętej metodyki analizy biznesowej [3] zakładającej dekompozycję procesów w celu modelowania przypadków użycia systemu informatycznego. W metodyce tej modelowanie systemu jest podporządkowane wynikom analizy biznesowej i koncentruje się na wspieraniu procesu przy użyciu technologii informatycznej. W metodyce tej tworzy się modele systemu po to aby stanowiły pomost pomiędzy wspomaganym procesem a wytworzonym, w celu wspomagania procesu, oprogramowaniem. W przeciwieństwie do metodyki strukturalnej powstające modele bardzo dobrze odzwierciedlają modele analityczne i są jednocześnie niezbędne w fazie implementacji.

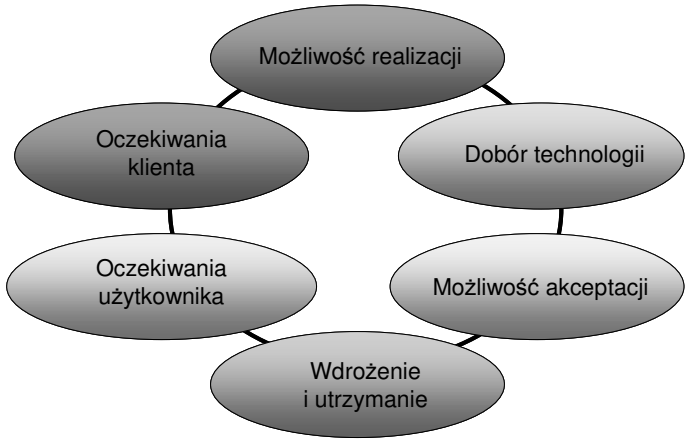
W przypadku systemu SZ@FiR jako praktyczną implementację obiektowego modelowania systemu wybrano UML [6].

3.3. Zarządzanie

Definicja procesu wytwórczego nie byłaby pełna, gdyby zabrakło w niej jawnego określenia ról i funkcji przyporządkowanych poszczególnym osobom/zespołom. Przyporządkowanie to daje szansę na skuteczne egzekwowanie podjętych przez zespół zobowiązań oraz pozwala zwykle na wdrożenie zasad przestrzegania zakresu praw i obowiązków wynikających z ról i funkcji. Tradycyjna, hierarchiczna struktura zespołu, z dobrze zdefiniowanymi rolami i funkcjami, może pozwolić na szybką identyfikację właściciela problemu,

jeśli problem się pojawi. Nie chroni ona jednak przed występowaniem sytuacji eskalacji każdej wątpliwości na poziom kierownika przedsięwzięcia, co często stwarza niebezpieczeństwo powstania paraliżu decyzyjnego.

W procesie realizacji systemu SZ@FIR istotą modelu zarządzania jest symbioza równoważnych, choć często sprzecznych ze sobą ról odpowiedzialnych za poszczególne punkty widzenia na system. Dominacja poszczególnych ról w zespole jest konsekwentnie przenoszona wraz z następowaniem kolejnych faz i etapów pojedynczej iteracji [4]. Zgodnie z tą zasadą w fazie opracowywania ogólnej koncepcji systemu dominuje rola zarządzania produktem, która koncentruje się na sformułowaniu wymagań wobec systemu wynikających wprost z oczekiwań klienta. W następnej fazie projektowania koncepcyjnego i planowania coraz większego znaczenia nabiera rola zarządzania programem, która koncentruje się na określeniu możliwości realizacji sformułowanych w poprzedniej fazie wymagań i zaplanowaniu ich realizacji. Idąc dalej, zgodnie z cyklem pojedynczej iteracji, wkraczamy w fazę projektowania szczegółowego i programowania, w której dominują role implementacji oraz edukacji użytkownika. W tej właśnie fazie następuje dobór technologii i weryfikacja powstającego oprogramowania pod kątem zgodności z oczekiwaniami przyszłych użytkowników. Gdy zostanie już osiągnięty założony w ramach przyrostu zakres funkcjonalny systemu, rozpoczyna się faza stabilizacji oprogramowania, w której dominują role testowanie oraz zarządzanie logistyką. Przy czym pierwsza z nich określa poziom akceptacji oprogramowania druga natomiast koncentruje się nad zapewnieniem wdrożenia i utrzymania systemu po jego wydaniu.



Rysunek 6. Różne punkty widzenia występujące w przedsięwzięciu

W strukturze skorelowanej z przebiegiem procesu wytwórczego, w której nastąpiło sprowadzenie poziomu decyzyjnego do ról odpowiedzialnych za poszczególne fazy procesu, w naturalny sposób kładzie się nacisk na podejmowanie decyzji i eliminowanie postaw nastawionych wyłącznie na

szukanie problemów. Podporządkowanie struktury zespołu modelowi procesu wytwórczego powoduje w efekcie powstanie i funkcjonowanie pewnego rodzaju centrów kompetencyjnych, skupiających się z jednej strony na produktach konkretnej fazy, z drugiej zaś reprezentujących istotny punkt widzenia na system od początku jego wytwarzania. Jednocześnie pozwala to w maksymalnym stopniu zabezpieczyć się przed ryzykiem powstania zagrożeń w różnych obszarach przez jednoznaczne określenie właścicieli-ról odpowiadających poszczególnym obszarom.

Tabela 1: Przechodzenie dominacji ról przez fazy iteracji

Punkt kontrolny	Rola
Zatwierdzona koncepcja	Zarządzanie produktem
Zatwierdzony plan i projekt przyrostu	Zarządzanie programem
Osiągnięty zakres	Implementacja i Edukacja użytkownika
Wydanie	Zarządzanie logistyką i Testowanie

4. Podsumowanie

Za każdym razem, gdy powoływane jest nowe przedsięwzięcie informatyczne, poszukuje się skutecznej recepty na jego przeprowadzenie w taki sposób, aby zakończyło się sukcesem. Wydaje się, że im nowsza technologia oraz im bardziej nietypowy sposób wprowadzenia produktu finalnego na rynek, tym większe ryzyko niepowodzenia przedsięwzięcia. W omówionym przykładzie zarówno technologia, jak i idea udostępnienia produktu były zupełnie nowe. Cóż więc sprawiło, że produkt został dostarczony w założonym czasie przy zachowaniu zasobów i bez ograniczania funkcjonalności?

Gdyby ze wszystkich czynników, które miały wpływ na prawidłowy przebieg przedsięwzięcia, wybrać najważniejsze, to śmiało można stwierdzić, że krytyczne dla jego powodzenia było jednoznaczne zdefiniowanie:

- Cyklu życia
- Metodyki modelowania
- Modelu zarządzania
- Architektury i technologii.

Zdaniem autora, każde z wymienionych wyżej pryncypiów wytwarzania oprogramowania jest jednakowo ważne i tylko jednoznaczne określenie każdego z nich gwarantuje powodzenie przedsięwzięcia.

Bibliografia

[1]. Peter Coad, Edward Yourdon. Analiza Obiektowa. *Oficyna wydawnicza READ ME,* Warszawa, 1994.

[2]. Peter Coad, Edward Yourdon, Projektowanie Obiektowe. *Oficyna wydawnicza READ ME,* Warszawa, 1994.

[3]. Robert Dumnicki, Artur Kasprzyk, Mariusz Kozłowski. Analiza i projektowanie obiektowe. *Wydawnictwo Helion,* Gliwice, 1998

[4]. Principles of application development - Class Pack 1516ACP. *Microsoft Corporation*, 1999.

[5]. Richard Anderson, Mark Birbeck, Michael Kay, Steven Livingstone, Bian Loesgen, Didier Martin, Stephen Mohr, Nikola Ozu, Bruce Peat, Jonatan Pinnock, Peter Stark, Kevin Williams. Professional XML. *Wrox Press Ltd.*®, 2000.

[6]. UML Semantics, UML Notation. *Rational Software Corporation.* 1997.