**Organizacja prac projektowych - Microsoft Solutions Framework (MSF)**

Przekroczenie terminu uruchomienia systemu lub zwiększenie kosztów jego budowy powoduje, że często przedsięwzięcia informatyczne kończą się niepowodzeniem. Dlatego szefowie wielkich przedsięwzięć informatycznych wiążą duże nadzieje z metodologią prac projektowych. Liczą na to, że  postępowanie zgodnie z zasadami pozwoli wypracować strategię dającą szansę na sukces.

Microsoft Solution Framework (MSF) jest zbiorem zasad projektowych i wybranych najlepszych praktyk wypracowanych w trakcie realizacji wielu projektów przez firmę Microsoft.

Zbiór wskazówek postępowania  i procedur  MSF  pogrupowany jest wokół podstawowych aspektów prowadzenia prac projektowych - w postaci następujących modeli:

1. *model zarządzania ryzykiem* (Risk Management Model)
2. *model procesów projektu* (Process Model)
3. *model zespołu projektowego* (Team Model)
4. *model architektury przedsiębiorstwa* (Enterprise Architecture Model)
5. *model procesu projektowania* (Design Process Model)
6. *model aplikacji* (Application Model)

Konstrukcja powyższych modeli ma przyczynić się do skrócenia czasu wytwarzania systemu informatycznego oraz redukcji kosztów i ryzyka przy zapewnieniu wysokiego  poziomu jakości.

**Model zarządzania ryzykiem**

Ideą przewodnią omawianego procesu jest minimalizacja prawdopodobieństwa przekroczenia założonych terminów i kosztów a w konsekwencji prawdopodobieństwo niepowodzenia projektu.

Ryzyko - to problem, zagrożenie, które może się urzeczywistnić. *Aktywne zarządzanie ryzykiem* (proactive risk management) polega na  identyfikowaniu potencjalnych przyczyn niepowodzenia projektu i podejmowaniu działań w celu zapobiegania lub minimalizowania wpływu ryzyka na projekt. Aktywne zarządzanie ryzykiemobejmuje następujące etapy:

1. Identyfikacja i wyrażenie ryzyka (zagrożenia).
2. Analiza ryzyka.
3. Planowanie obsługi zagrożenia.
4. Śledzenie sytuacji zagrożenia.
5. Kontrola, przeciwdziałanie – konkretne ryzyko może zostać zażegnane całkowicie lub opanowane w danej chwili.

|  |
| --- |
|  |

**Model procesów projektu**

Model procesów projektu to model procesu wytwarzania oprogramowania, określający kolejność działań w projekcie od początku do zakończenia projektu. Podstawowe modele procesów projektowych to:

1. Model kaskadowy (watefall model) – każdy zestaw zadań musi zostać zakończony przed rozpoczęciem kolejnej fazy.
   * Używane są punkty kontrolne nazywane  *kamieniami milowymi (milestones)* jako punkty przejścia i oceny. Są podstawą dla cyklu życia prowadzenia projektu opartego na zadaniach (*task-driven development life cycle*),
   * zwykle różne zespoły projektowe prowadzą różne fazy,
   * każda faza musi zostać dokładnie udokumentowana, aby następny zespół miał pełną informację,
   * krytyczne decyzje zapadają wcześnie,
   * testowanie odbywa się w ostatniej fazie projektu,
   * komunikacja pomiędzy ludźmi jest ograniczona do dokumentacji pisanej – można utracić krytyczną informację, istotny kontekst może zostać nie przekazany,
   * informacja o potrzebach klienta może być utracona w szeregu dokumentów,
   * na początku koncentracja na potrzebach klienta a nie na wizji co dostępna technologia może dać użytkownikowi.
2. Model spiralny (spiral model) – ciągła potrzeba poprawiania wymagań i oszacowań projektowych.
   * Oparty na więzi pomiędzy zespołem projektowym a klientem.
   * Opiera się na kolejnych iteracjach w celu wprowadzania ulepszeń. Nie ma wyraźnie określonych kamieni milowych.

Model  postulowany przez MSF, jest połączeniem modelu kaskadowego i spiralnego przy podstawowym założeniu przyrostowego (iteracyjnego) budowania funkcjonalności wzbogaconego o  częste punkty kontrolne - kamienie milowe.

Cykl życia projektu oparty na częstych punktach kontrolnych (kamieniach milowych):

* Wprowadza dyscyplinę realizowania zadań projektowych pozwalając jednak na zachowanie elastyczności i iteracyjności (do wprowadzania zmian).
* Kamienie milowe to w MSF punkty kontrolne, przeglądowe, synchronizujące, a nie punkty martwe, zamrożone.
* Pozwalają zespołowi oceniać postęp w pracach i dokonywać poprawek.

Struktura wytwarzania projektu oparta jest na 4 fazach - opisanych dalej. Każda faza kończy się zewnętrznie widocznym kamieniem milowym.

|  |
| --- |
|  |

Wyróżniamy dwa rodzaje kamieni milowych: *główne kamienie milowe* i *pośrednie kamienie milowe*.

* Główne kamienie milowe reprezentują osiągnięcia zespołu potwierdzone zgodą klienta na kontynuację prac.
* Produkty prac projektowych nazywane *dostawami* (*deliverables*) są fizycznym dowodem na to, że zespół osiągnął kolejny kamień milowy. ( Oto przykładowy pakiet dostaw  - zbiór dokumentów projektowych wytwarzanych w czasie powstawania systemu informatycznego).

Oto zapowiadane cztery fazy prac projektowych.

Faza 1 (*Faza określania wizji*) Zespół projektowy razem z klientem określają wymagania biznesowe i ogólne cele projektu. Kończy się kamieniem milowym akceptacji wizji co wskazuje, że zespół projektowy i klient zgadzają się na opracowany kierunek projektu.

Faza 2 (*Faza planowania*) Kończy się kamieniem milowym akceptacji planu.

Faza 3 (*Faza tworzenia*) Kończy się gdy produkt projektu (np. oprogramowanie) jest gotowy i może zostać poddany wdrożeniu (deployment).

Faza 4 (*Faza stabilizacji*)  Projekt kończy się akceptacją produktu,  wypuszczeniem go na rynek lub wdrożeniem.

Wersjonowanie zwalnianych "wypuszczanych” produktów (versioned releases)

W przypadku dużego, złożonego projektu jest sugerowane dokonanie podziału całego zadania na wiele wersjonowanych wydań produktu, gdzie:

* pierwsze wydanie dostarcza głównego produktu;
* następne wersje dodają kolejne cechy aż produkt spełnia opracowaną w pierwszej fazie wizję projektu;
* to co najważniejsze jest dostarczane w pierwszej kolejności;
* wersjonowane wydania umożliwiają zespołowi projektowemu szybsze reagowanie na zmiany zakresu, planu i ryzyka podczas tworzenia produktu.

|  |
| --- |
|  |

Zarządzanie trójkątem zależności

Wytworzenie systemu informatycznego wymaga umiejętnego zarządzania zmiennymi projektowymi Podstawowe zmienne projektowe to :

1. *zasoby* (resources) jak ludzie, pieniądze,
2. *harmonogram* (schedule) i
3. *cechy produktu, funkcje* (features).

* Zachodzą między nimi wzajemne zależności (tradeoff). Gdy jedna zmienna ulega zmianie, inne muszą być uaktualnione.
* Kluczem do sukcesu projektu jest wyważenie proporcji między nimi.
* Projekt jest udany gdy klient uważa, że zespół projektowy dokonał prawidłowych wyborów. Zespół projektowy powinien pytać klienta o priorytety jak najwcześniej i jak najczęściej.

|  |
| --- |
|  |

W celu pełnej kontroli nad przebiegiem wytwarzania systemu informatycznego zalecane jest utrzymywanie „żywych dokumentów projektowych”

* Dokument żywy to taki, który odzwierciedla aktualny stan prac projektowych.
* Dzięki nim mamy strukturalny proces kontroli zmian (zarządzania zmianami).
* Zaczynamy prace nad nimi jak najwcześniej (*baseline early*), zamykamy jak najpóźniej (*freeze late*).

**Model zespołu projektowego**

Istotnym elementem metody MSF jest podział zespołu wytwarzającego system informatyczny na podzespoły, które mają precyzyjnie określone role i zadania. Każdy zespół  ma kierownika zespołu. - Zespół kierowników stanowi ścisłe kierownictwo projektu:

* kierownik produktu
* kierownik programu (administrator projektu)
* wytwórca
* tester
* kierownik edukacji użytkownika
* kierownik logistyki

|  |
| --- |
|  |

Każdy kierownik wraz ze swoim zespołem odpowiada za dokładnie określone zadania:

1. Kierownik produktu, zarządzanie produktem (*product management*)
   * prowadzenie zespołu w kierunku realizacji oczekiwań klienta,
   * reprezentowanie zespołu przed klientem,
   * określanie zakresu projektu,
   * opracowywanie i realizowanie planu komunikacji z klientem i użytkownikami.
2. Kierownik programu prac projektowych, zarządzanie pracami  projektowymi (*program management*)
   * kierowanie, koordynacja ogólnym procesem projektowym,
   * zarządzanie harmonogramem, zasobami, ograniczeniami projektu,
   * odpowiedzialność za dostarczenie właściwego produktu we właściwym czasie,
   * odpowiedzialność za zakres produktu i specyfikację,
   * tworzenie raportów o stanie prac projektowych,
   * zarządzanie alokacją zasobów,
   * organizacja komunikacji wewnątrz zespołu projektowego, łagodzenie konfliktów,
   * kierowanie podejmowaniem krytycznych decyzji.
3. Wytwórca, budowa produktu (*development*)
   * budowa i testowanie produktu spełniającego wymagania i oczekiwania klienta,
   * uczestnictwo w projektowaniu produktu,
   * szacowanie czasu i pracy do wykonania produktu,
   * konsultacja w sprawach technologii,
   * pomoc przy instalacji i wdrożeniu  produktu,
   * tworzenie, konfigurowanie i dostosowywanie produktu do klienta (*customization*).
4. Tester, Testowanie (*Testing*)
   * opracowanie strategii, planów i skryptów testowania,
   * kontrola procesu i stanu budowy produktu: co jest źle, co jest już dobrze,
   * przeprowadzanie testów,
   * uczestniczenie w określaniu kryteriów jakości.
5. Edukacja, szkolenie użytkowników (*User Education*)
   * uczenie użytkowników sprawnego posługiwania się produktem,
   * zarządzanie oczekiwaniami użytkowników,
   * przygotowanie materiałów szkoleniowych/instrukcji sprawnego posługiwania się systemem (w tym system pomocy bezpośredniej),
   * reprezentowanie oczekiwań użytkowników przed zespołem projektowym,
   * uczestniczenie w zbieraniu wymagań użytkowników.
6. Kierownik logistyki, logistyka (*Logistics Management*)
   * kontakt ze służbami wspomagającymi (działem operacyjnym),
   * zapewnienie że produkt jest wdrażalny, zarządzalny i że w firmie będzie wspomagana jego eksploatacja,
   * uczestnictwo przy projektowaniu, wspomaganie testów beta produktu,
   * wspomaganie działu operacyjnego przy konfigurowaniu środowiska i samego produktu.

Zasady zespołu projektowego MSF (*team of peers*) - kierującego projektem

1. Wspólna wizja projektu.
2. Pełna współpraca.
3. Uczenie się na doświadczeniach zdobytych w poprzednich projektach.
4. Wspólne zarządzanie projektem oraz podejmowanie decyzji. Bez jednego, głównego kierownika projektu.(Wszyscy muszą uzgodnić wspólne stanowisko - w przypadku różnicy poglądów - decyduje klient)
5. Wspólna praca członków projektu  w jednym miejscu.
6. Podział pracy w dużych zespołach na mniejsze.

 Podsumowanie:

|  |  |
| --- | --- |
| **Rola** | **Cel** |
| Kierownik produktu | Zadowolony klient |
| Kierownik programu | Dostarczenie produktu w ramach ograniczeń projektowych |
| Wytwórca | Dostarczenie produktu zgodnego ze specyfikacją |
| Tester | Sprawdzenie wszystkich potencjalnych problemów |
| Edukator użytkownika | Sprawne użycie systemu przez użytkowników |
| Kierownik logistyki | Sprawne wdrożenie produktu |

**Model architektury przedsiębiorstwa (Enterprise Architecture Model) - BAIT**

|  |
| --- |
|  |

Obraz działalności firmy, tworzony w celu dokładnej identyfikacji zadania,  jest podzielony na cztery aspekty według zasady "jedna architektura, ale cztery perspektywy":

* *biznes* - napęd działania przedsiębiorstwa,
* *aplikacje,*
* *informacje* (środki do osiągnięcia celów i zadań biznesu) i
* *technologia* (fundament, baza).

Biznes

1. Cele i zadania: na czym polega biznes w organizacji?
2. Struktura organizacyjna firmy: kto jest odpowiedzialny za co?
3. Procesy biznesowe: jak organizacja wykonuje swój biznes, jak są dostarczane produkty i usługi?
4. Klient: kim jest klient?
5. Dostawcy/sprzedawcy: z kim współpracuje organizacja?

Aplikacje

* zautomatyzowane usługi, które wspierają procesy biznesowe,
* identyfikacja redundancji między departamentami.

Informacja

* co organizacja musi wiedzieć (dane, informacje, wiedza) aby prowadzić procesy i operacje biznesowe,
* zasady zarządzania danymi,
* opisy wzorców konsumpcji i produkcji informacji w organizacji.

Technologia

* wspomaganie sprzętowe, programistyczne i techniczne,
* standardy, wskazówki (guidelines) potrzebne aby osiągnąć misję  biznesu,
* określa usługi technologiczne potrzebne do realizacji misji biznesu:
  + środowiska budowy systemów,
  + specyfikacje techniczne,
  + warstwy sprzętowe,
  + systemy operacyjne,
  + platformy sieciowe,
  + biblioteki kodu,
  + dokumenty ze standardami,
  + wskazówki projektowe.

**Model procesu projektowania (*Design Process Model*)**

|  |
| --- |
|  |

W procesie projektowania można wyróżnić trzy perspektywy  ustawione w sekwencję wielokrotnie iterowaną:

1. *koncepcyjne projektowanie*:
   * perspektywa użytkownika: co robi i co potrzebuje aby to robić,
   * identyfikacja potrzeb biznesowych,
   * modele które są łatwo zrozumiałe zarówno dla klienta jak i projektanta (jak zgrubne szkice i scenariusze tworzone przy projektowaniu domu);
2. *logiczne projektowanie*:
   * perspektywa użytkownika zastosowana do budowy produktu, który spełnia potrzeby biznesowe,
   * organizuje szczegóły budowanej aplikacji, która ma spełnić potrzeby biznesu i wymagania użytkowników,
   * struktura rozwiązania i ścieżki komunikacyjne między elementami (jak plan pięter, elewacji, związki między nimi).
3. *projektowanie fizyczne*:
   * perspektywa technologii, której będzie używać użytkownik systemu,
   * celem jest zastosowanie ograniczeń technologicznych świata rzeczywistego do rezultatów projektowania logicznego takich jak rozważania implementacyjne i dotyczące efektywności działania systemu (jak plany fizycznych elementów struktury jak odrutowanie, hydraulika, system ogrzewania, wentylacji).

**Model aplikacji (*Application Model*)**

* *Aplikacja* - logiczna sieć współpracujących, rozproszonych *serwisów*.
* *Serwis* to niezależny moduł, który ma dokładnie określony interfejs programistyczny, który może być wielokrotnie używany przez inne serwisy, który może być używany w różnych konfiguracjach uruchomieniowych: na tym samym komputerze, w komputerze w sieci lokalnej, w sieci Internet.
* *Wielokrotne użycie serwisu* oznacza, że pojedyncze rozwiązanie biznesowe może wchodzić w skład wielu faktycznych aplikacji.

Funkcje modelu aplikacji:

1. Ustalenie definicji, reguł i związków, które będą tworzyć strukturę aplikacji.
   * Służy jako baza wymiany idei podczas logicznego projektowania aplikacji.
   * Nacisk jest na poziom logiczny nie fizyczny - jaką strukturę ma aplikacja a nie jak będzie implementowana.
2. Stanowi/umożliwia spójne podejście do projektowania i budowy aplikacji.
   * Model buduje wspólne rozumienie aplikacji i określa robocze słownictwo do tworzenia aplikacji.
3. Używa projektowania komponentowego opartego na serwisach.

**Model aplikacji oparty o serwisy**

|  |
| --- |
|  |

3-warstwowy logiczny model projektowania wielowarstwowych, rozproszonych aplikacji, które obejmują trzy kategorie serwisów:

* *użytkownika*,
* *biznesowe* i
* *dotyczące danych*.

1. *Serwis użytkownika* – dotyczący interfejsu użytkownika:
   * wizualny bądź programowy interfejs – użytkownik może być albo osobą albo aplikacją,
   * prezentacja informacji i zbieranie danych od użytkownika.
2. *Serwis biznesowy*:
   * sterowanie operacjami biznesowymi, realizacja reguł biznesowych, zapewnienie integralności transakcji biznesowych,
   * transformacja danych na informacje przez zastosowanie reguł biznesowych.
3. *Serwis danych* – najniższy widoczny poziom szczegółów używanych do operowania na danych:
   * utrzymywanie trwałych danych aplikacji,
   * dostarczenie możliwości definiowania, tworzenia, odczytywania, aktualizacji i usuwania danych.

[**Przykładowy Pakiet Dostaw:**](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/MPD.htm)

Produkty prac projektowych nazywane *dostawami* stanowią fizyczny dowód na to, że zespół osiągnął kolejny kamień milowy. Proszę przeanalizować przykładowy [spis](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/MPD.htm) dokumentów powstających w czasie tworzenia systemy informatycznego.

**Podsumowanie**

Organizacja prac projektowych jest nietrywialnym zadaniem. Większość metodyk prowadzenia prac projektowych obejmuje fazy:

1. Strategia (analiza wstępna problemu)
2. Analiza szczegółowa problemu
3. Projektowanie systemu
4. Implementacja systemu (z tworzeniem dokumentacji i testowaniem)
5. Wdrożenie systemu

W ramach każdej fazy powstaje określona dokumentacja projektowa – na ogół uzyskiwana i przechowywana w specjalnej bazie danych nazywanej *repozytorium projektowym*. Repozytorium projektowe jest częścią narzędzi wspomagających prowadzenia projektu – CASE.

**MSF** czyli *Microsoft Solutions Framework* jest to zbiór wskazówek postępowania pogrupowanych wokół podstawowych aspektów prowadzenia prac projektowych - w postaci następujących modeli:

1. model zarządzania ryzykiem (*Risk Management Model*)
2. model procesów prowadzenia projektu (*Process Model*)
3. model zespołu projektowego (*Team Model*)
4. model architektury przedsiębiorstwa (*Enterprise Architecture Model*)
5. model procesu projektowania (*Design Process Model*)
6. model aplikacji (*Application Model*)

**Słownik**

[analiza - faza w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Analiza) - początkowy etap prac projektowych polegający na identyfikacji istotnych składników potrzebnych do utworzenia systemu informacyjnego obsługującego pewną dziedzinę działalności firmy, organizacji lub innego działającego układu. Na tym etapie skupiamy naszą uwagę na odpowiedziach na pytania "co", pozostawiając na później pytania "jak".  
[projektowanie - faza w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Proj) - etap prac projektowych następujący po etapie analizy polegający na transformacji jej wyników na postać układu składników docelowego systemu informacyjnego pokazującego jego schemat i zasady działania.  
[dokumentacja projektowa](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Dokumentacja) - opis wyników prac projektowych w postaci ustandaryzowanych dokumentów tekstowych lub multimedialnych.  
[repozytorium projektowe](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#repozytorium) - baza danych zawierająca wszystkie obiekty (w tym dokumenty) używane wspólnie przez cały zespół projektowy przez cały okres działalności projektowej.  
[MSF (Microsoft Solutions Framework)](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Organizacja) - metodyka projektowa w postaci zbioru wskazówek postępowania pogrupowanych wokół podstawowych aspektów prowadzenia prac projektowych takich jak zarządzanie ryzykiem, organizacja zespołu projektowego, podział na fazy i poziomy, model aplikacji oparty na serwisach.  
[ryzyko w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Ryzyko) - aktywne zarządzanie ryzykiem polega na  identyfikowaniu potencjalnych przyczyn niepowodzenia projektu i podejmowaniu działań w celu zapobiegania lub minimalizowania wpływu ryzyka na projekt.  
[kamień milowy w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#kamien) - punkt kontrolnyw pracach projektowych służący do oceny wyników (ich akceptacji lub odrzucenia) i przejścia do kolejnego etapu prac.  
[serwis w aplikacji](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Apl) - podstawa modelu aplikacji jako logicznej sieci współpracujących, rozproszonych serwisów - podlegających zasadzie wielokrotnego użycia. Są trzy rodzaje serwisów: użytkownika,  biznesowe i  dotyczące danych.

**Literatura do wykładu**

* Microsoft Solutions Framework, materiały do kursu firmy Microsoft.
* Materiały do wykładów z przedmiotu "Analiza systemowa" prowadzanego w PJWSTK przez dr Andrzeja Jankowskiego w 1999r. (Przykładowy Pakiet Dostaw.)

**Organizacja prac projektowych - Microsoft Solutions Framework (MSF)**

Przekroczenie terminu uruchomienia systemu lub zwiększenie kosztów jego budowy powoduje, że często przedsięwzięcia informatyczne kończą się niepowodzeniem. Dlatego szefowie wielkich przedsięwzięć informatycznych wiążą duże nadzieje z metodologią prac projektowych. Liczą na to, że  postępowanie zgodnie z zasadami pozwoli wypracować strategię dającą szansę na sukces.

Microsoft Solution Framework (MSF) jest zbiorem zasad projektowych i wybranych najlepszych praktyk wypracowanych w trakcie realizacji wielu projektów przez firmę Microsoft.

Zbiór wskazówek postępowania  i procedur  MSF  pogrupowany jest wokół podstawowych aspektów prowadzenia prac projektowych - w postaci następujących modeli:

1. *model zarządzania ryzykiem* (Risk Management Model)
2. *model procesów projektu* (Process Model)
3. *model zespołu projektowego* (Team Model)
4. *model architektury przedsiębiorstwa* (Enterprise Architecture Model)
5. *model procesu projektowania* (Design Process Model)
6. *model aplikacji* (Application Model)

Konstrukcja powyższych modeli ma przyczynić się do skrócenia czasu wytwarzania systemu informatycznego oraz redukcji kosztów i ryzyka przy zapewnieniu wysokiego  poziomu jakości.

**Model zarządzania ryzykiem**

Ideą przewodnią omawianego procesu jest minimalizacja prawdopodobieństwa przekroczenia założonych terminów i kosztów a w konsekwencji prawdopodobieństwo niepowodzenia projektu.

Ryzyko - to problem, zagrożenie, które może się urzeczywistnić. *Aktywne zarządzanie ryzykiem* (proactive risk management) polega na  identyfikowaniu potencjalnych przyczyn niepowodzenia projektu i podejmowaniu działań w celu zapobiegania lub minimalizowania wpływu ryzyka na projekt. Aktywne zarządzanie ryzykiemobejmuje następujące etapy:

1. Identyfikacja i wyrażenie ryzyka (zagrożenia).
2. Analiza ryzyka.
3. Planowanie obsługi zagrożenia.
4. Śledzenie sytuacji zagrożenia.
5. Kontrola, przeciwdziałanie – konkretne ryzyko może zostać zażegnane całkowicie lub opanowane w danej chwili.

|  |
| --- |
|  |

**Model procesów projektu**

Model procesów projektu to model procesu wytwarzania oprogramowania, określający kolejność działań w projekcie od początku do zakończenia projektu. Podstawowe modele procesów projektowych to:

1. Model kaskadowy (watefall model) – każdy zestaw zadań musi zostać zakończony przed rozpoczęciem kolejnej fazy.
   * Używane są punkty kontrolne nazywane  *kamieniami milowymi (milestones)* jako punkty przejścia i oceny. Są podstawą dla cyklu życia prowadzenia projektu opartego na zadaniach (*task-driven development life cycle*),
   * zwykle różne zespoły projektowe prowadzą różne fazy,
   * każda faza musi zostać dokładnie udokumentowana, aby następny zespół miał pełną informację,
   * krytyczne decyzje zapadają wcześnie,
   * testowanie odbywa się w ostatniej fazie projektu,
   * komunikacja pomiędzy ludźmi jest ograniczona do dokumentacji pisanej – można utracić krytyczną informację, istotny kontekst może zostać nie przekazany,
   * informacja o potrzebach klienta może być utracona w szeregu dokumentów,
   * na początku koncentracja na potrzebach klienta a nie na wizji co dostępna technologia może dać użytkownikowi.
2. Model spiralny (spiral model) – ciągła potrzeba poprawiania wymagań i oszacowań projektowych.
   * Oparty na więzi pomiędzy zespołem projektowym a klientem.
   * Opiera się na kolejnych iteracjach w celu wprowadzania ulepszeń. Nie ma wyraźnie określonych kamieni milowych.

Model  postulowany przez MSF, jest połączeniem modelu kaskadowego i spiralnego przy podstawowym założeniu przyrostowego (iteracyjnego) budowania funkcjonalności wzbogaconego o  częste punkty kontrolne - kamienie milowe.

Cykl życia projektu oparty na częstych punktach kontrolnych (kamieniach milowych):

* Wprowadza dyscyplinę realizowania zadań projektowych pozwalając jednak na zachowanie elastyczności i iteracyjności (do wprowadzania zmian).
* Kamienie milowe to w MSF punkty kontrolne, przeglądowe, synchronizujące, a nie punkty martwe, zamrożone.
* Pozwalają zespołowi oceniać postęp w pracach i dokonywać poprawek.

Struktura wytwarzania projektu oparta jest na 4 fazach - opisanych dalej. Każda faza kończy się zewnętrznie widocznym kamieniem milowym.

|  |
| --- |
|  |

Wyróżniamy dwa rodzaje kamieni milowych: *główne kamienie milowe* i *pośrednie kamienie milowe*.

* Główne kamienie milowe reprezentują osiągnięcia zespołu potwierdzone zgodą klienta na kontynuację prac.
* Produkty prac projektowych nazywane *dostawami* (*deliverables*) są fizycznym dowodem na to, że zespół osiągnął kolejny kamień milowy. ( Oto przykładowy pakiet dostaw  - zbiór dokumentów projektowych wytwarzanych w czasie powstawania systemu informatycznego).

Oto zapowiadane cztery fazy prac projektowych.

Faza 1 (*Faza określania wizji*) Zespół projektowy razem z klientem określają wymagania biznesowe i ogólne cele projektu. Kończy się kamieniem milowym akceptacji wizji co wskazuje, że zespół projektowy i klient zgadzają się na opracowany kierunek projektu.

Faza 2 (*Faza planowania*) Kończy się kamieniem milowym akceptacji planu.

Faza 3 (*Faza tworzenia*) Kończy się gdy produkt projektu (np. oprogramowanie) jest gotowy i może zostać poddany wdrożeniu (deployment).

Faza 4 (*Faza stabilizacji*)  Projekt kończy się akceptacją produktu,  wypuszczeniem go na rynek lub wdrożeniem.

Wersjonowanie zwalnianych "wypuszczanych” produktów (versioned releases)

W przypadku dużego, złożonego projektu jest sugerowane dokonanie podziału całego zadania na wiele wersjonowanych wydań produktu, gdzie:

* pierwsze wydanie dostarcza głównego produktu;
* następne wersje dodają kolejne cechy aż produkt spełnia opracowaną w pierwszej fazie wizję projektu;
* to co najważniejsze jest dostarczane w pierwszej kolejności;
* wersjonowane wydania umożliwiają zespołowi projektowemu szybsze reagowanie na zmiany zakresu, planu i ryzyka podczas tworzenia produktu.

|  |
| --- |
|  |

Zarządzanie trójkątem zależności

Wytworzenie systemu informatycznego wymaga umiejętnego zarządzania zmiennymi projektowymi Podstawowe zmienne projektowe to :

1. *zasoby* (resources) jak ludzie, pieniądze,
2. *harmonogram* (schedule) i
3. *cechy produktu, funkcje* (features).

* Zachodzą między nimi wzajemne zależności (tradeoff). Gdy jedna zmienna ulega zmianie, inne muszą być uaktualnione.
* Kluczem do sukcesu projektu jest wyważenie proporcji między nimi.
* Projekt jest udany gdy klient uważa, że zespół projektowy dokonał prawidłowych wyborów. Zespół projektowy powinien pytać klienta o priorytety jak najwcześniej i jak najczęściej.

|  |
| --- |
|  |

W celu pełnej kontroli nad przebiegiem wytwarzania systemu informatycznego zalecane jest utrzymywanie „żywych dokumentów projektowych”

* Dokument żywy to taki, który odzwierciedla aktualny stan prac projektowych.
* Dzięki nim mamy strukturalny proces kontroli zmian (zarządzania zmianami).
* Zaczynamy prace nad nimi jak najwcześniej (*baseline early*), zamykamy jak najpóźniej (*freeze late*).

**Model zespołu projektowego**

Istotnym elementem metody MSF jest podział zespołu wytwarzającego system informatyczny na podzespoły, które mają precyzyjnie określone role i zadania. Każdy zespół  ma kierownika zespołu. - Zespół kierowników stanowi ścisłe kierownictwo projektu:

* kierownik produktu
* kierownik programu (administrator projektu)
* wytwórca
* tester
* kierownik edukacji użytkownika
* kierownik logistyki

|  |
| --- |
|  |

Każdy kierownik wraz ze swoim zespołem odpowiada za dokładnie określone zadania:

1. Kierownik produktu, zarządzanie produktem (*product management*)
   * prowadzenie zespołu w kierunku realizacji oczekiwań klienta,
   * reprezentowanie zespołu przed klientem,
   * określanie zakresu projektu,
   * opracowywanie i realizowanie planu komunikacji z klientem i użytkownikami.
2. Kierownik programu prac projektowych, zarządzanie pracami  projektowymi (*program management*)
   * kierowanie, koordynacja ogólnym procesem projektowym,
   * zarządzanie harmonogramem, zasobami, ograniczeniami projektu,
   * odpowiedzialność za dostarczenie właściwego produktu we właściwym czasie,
   * odpowiedzialność za zakres produktu i specyfikację,
   * tworzenie raportów o stanie prac projektowych,
   * zarządzanie alokacją zasobów,
   * organizacja komunikacji wewnątrz zespołu projektowego, łagodzenie konfliktów,
   * kierowanie podejmowaniem krytycznych decyzji.
3. Wytwórca, budowa produktu (*development*)
   * budowa i testowanie produktu spełniającego wymagania i oczekiwania klienta,
   * uczestnictwo w projektowaniu produktu,
   * szacowanie czasu i pracy do wykonania produktu,
   * konsultacja w sprawach technologii,
   * pomoc przy instalacji i wdrożeniu  produktu,
   * tworzenie, konfigurowanie i dostosowywanie produktu do klienta (*customization*).
4. Tester, Testowanie (*Testing*)
   * opracowanie strategii, planów i skryptów testowania,
   * kontrola procesu i stanu budowy produktu: co jest źle, co jest już dobrze,
   * przeprowadzanie testów,
   * uczestniczenie w określaniu kryteriów jakości.
5. Edukacja, szkolenie użytkowników (*User Education*)
   * uczenie użytkowników sprawnego posługiwania się produktem,
   * zarządzanie oczekiwaniami użytkowników,
   * przygotowanie materiałów szkoleniowych/instrukcji sprawnego posługiwania się systemem (w tym system pomocy bezpośredniej),
   * reprezentowanie oczekiwań użytkowników przed zespołem projektowym,
   * uczestniczenie w zbieraniu wymagań użytkowników.
6. Kierownik logistyki, logistyka (*Logistics Management*)
   * kontakt ze służbami wspomagającymi (działem operacyjnym),
   * zapewnienie że produkt jest wdrażalny, zarządzalny i że w firmie będzie wspomagana jego eksploatacja,
   * uczestnictwo przy projektowaniu, wspomaganie testów beta produktu,
   * wspomaganie działu operacyjnego przy konfigurowaniu środowiska i samego produktu.

Zasady zespołu projektowego MSF (*team of peers*) - kierującego projektem

1. Wspólna wizja projektu.
2. Pełna współpraca.
3. Uczenie się na doświadczeniach zdobytych w poprzednich projektach.
4. Wspólne zarządzanie projektem oraz podejmowanie decyzji. Bez jednego, głównego kierownika projektu.(Wszyscy muszą uzgodnić wspólne stanowisko - w przypadku różnicy poglądów - decyduje klient)
5. Wspólna praca członków projektu  w jednym miejscu.
6. Podział pracy w dużych zespołach na mniejsze.

 Podsumowanie:

|  |  |
| --- | --- |
| **Rola** | **Cel** |
| Kierownik produktu | Zadowolony klient |
| Kierownik programu | Dostarczenie produktu w ramach ograniczeń projektowych |
| Wytwórca | Dostarczenie produktu zgodnego ze specyfikacją |
| Tester | Sprawdzenie wszystkich potencjalnych problemów |
| Edukator użytkownika | Sprawne użycie systemu przez użytkowników |
| Kierownik logistyki | Sprawne wdrożenie produktu |

**Model architektury przedsiębiorstwa (Enterprise Architecture Model) - BAIT**

|  |
| --- |
|  |

Obraz działalności firmy, tworzony w celu dokładnej identyfikacji zadania,  jest podzielony na cztery aspekty według zasady "jedna architektura, ale cztery perspektywy":

* *biznes* - napęd działania przedsiębiorstwa,
* *aplikacje,*
* *informacje* (środki do osiągnięcia celów i zadań biznesu) i
* *technologia* (fundament, baza).

Biznes

1. Cele i zadania: na czym polega biznes w organizacji?
2. Struktura organizacyjna firmy: kto jest odpowiedzialny za co?
3. Procesy biznesowe: jak organizacja wykonuje swój biznes, jak są dostarczane produkty i usługi?
4. Klient: kim jest klient?
5. Dostawcy/sprzedawcy: z kim współpracuje organizacja?

Aplikacje

* zautomatyzowane usługi, które wspierają procesy biznesowe,
* identyfikacja redundancji między departamentami.

Informacja

* co organizacja musi wiedzieć (dane, informacje, wiedza) aby prowadzić procesy i operacje biznesowe,
* zasady zarządzania danymi,
* opisy wzorców konsumpcji i produkcji informacji w organizacji.

Technologia

* wspomaganie sprzętowe, programistyczne i techniczne,
* standardy, wskazówki (guidelines) potrzebne aby osiągnąć misję  biznesu,
* określa usługi technologiczne potrzebne do realizacji misji biznesu:
  + środowiska budowy systemów,
  + specyfikacje techniczne,
  + warstwy sprzętowe,
  + systemy operacyjne,
  + platformy sieciowe,
  + biblioteki kodu,
  + dokumenty ze standardami,
  + wskazówki projektowe.

**Model procesu projektowania (*Design Process Model*)**

|  |
| --- |
|  |

W procesie projektowania można wyróżnić trzy perspektywy  ustawione w sekwencję wielokrotnie iterowaną:

1. *koncepcyjne projektowanie*:
   * perspektywa użytkownika: co robi i co potrzebuje aby to robić,
   * identyfikacja potrzeb biznesowych,
   * modele które są łatwo zrozumiałe zarówno dla klienta jak i projektanta (jak zgrubne szkice i scenariusze tworzone przy projektowaniu domu);
2. *logiczne projektowanie*:
   * perspektywa użytkownika zastosowana do budowy produktu, który spełnia potrzeby biznesowe,
   * organizuje szczegóły budowanej aplikacji, która ma spełnić potrzeby biznesu i wymagania użytkowników,
   * struktura rozwiązania i ścieżki komunikacyjne między elementami (jak plan pięter, elewacji, związki między nimi).
3. *projektowanie fizyczne*:
   * perspektywa technologii, której będzie używać użytkownik systemu,
   * celem jest zastosowanie ograniczeń technologicznych świata rzeczywistego do rezultatów projektowania logicznego takich jak rozważania implementacyjne i dotyczące efektywności działania systemu (jak plany fizycznych elementów struktury jak odrutowanie, hydraulika, system ogrzewania, wentylacji).

**Model aplikacji (*Application Model*)**

* *Aplikacja* - logiczna sieć współpracujących, rozproszonych *serwisów*.
* *Serwis* to niezależny moduł, który ma dokładnie określony interfejs programistyczny, który może być wielokrotnie używany przez inne serwisy, który może być używany w różnych konfiguracjach uruchomieniowych: na tym samym komputerze, w komputerze w sieci lokalnej, w sieci Internet.
* *Wielokrotne użycie serwisu* oznacza, że pojedyncze rozwiązanie biznesowe może wchodzić w skład wielu faktycznych aplikacji.

Funkcje modelu aplikacji:

1. Ustalenie definicji, reguł i związków, które będą tworzyć strukturę aplikacji.
   * Służy jako baza wymiany idei podczas logicznego projektowania aplikacji.
   * Nacisk jest na poziom logiczny nie fizyczny - jaką strukturę ma aplikacja a nie jak będzie implementowana.
2. Stanowi/umożliwia spójne podejście do projektowania i budowy aplikacji.
   * Model buduje wspólne rozumienie aplikacji i określa robocze słownictwo do tworzenia aplikacji.
3. Używa projektowania komponentowego opartego na serwisach.

**Model aplikacji oparty o serwisy**

|  |
| --- |
|  |

3-warstwowy logiczny model projektowania wielowarstwowych, rozproszonych aplikacji, które obejmują trzy kategorie serwisów:

* *użytkownika*,
* *biznesowe* i
* *dotyczące danych*.

1. *Serwis użytkownika* – dotyczący interfejsu użytkownika:
   * wizualny bądź programowy interfejs – użytkownik może być albo osobą albo aplikacją,
   * prezentacja informacji i zbieranie danych od użytkownika.
2. *Serwis biznesowy*:
   * sterowanie operacjami biznesowymi, realizacja reguł biznesowych, zapewnienie integralności transakcji biznesowych,
   * transformacja danych na informacje przez zastosowanie reguł biznesowych.
3. *Serwis danych* – najniższy widoczny poziom szczegółów używanych do operowania na danych:
   * utrzymywanie trwałych danych aplikacji,
   * dostarczenie możliwości definiowania, tworzenia, odczytywania, aktualizacji i usuwania danych.

[**Przykładowy Pakiet Dostaw:**](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/MPD.htm)

Produkty prac projektowych nazywane *dostawami* stanowią fizyczny dowód na to, że zespół osiągnął kolejny kamień milowy. Proszę przeanalizować przykładowy [spis](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/MPD.htm) dokumentów powstających w czasie tworzenia systemy informatycznego.

**Podsumowanie**

Organizacja prac projektowych jest nietrywialnym zadaniem. Większość metodyk prowadzenia prac projektowych obejmuje fazy:

1. Strategia (analiza wstępna problemu)
2. Analiza szczegółowa problemu
3. Projektowanie systemu
4. Implementacja systemu (z tworzeniem dokumentacji i testowaniem)
5. Wdrożenie systemu

W ramach każdej fazy powstaje określona dokumentacja projektowa – na ogół uzyskiwana i przechowywana w specjalnej bazie danych nazywanej *repozytorium projektowym*. Repozytorium projektowe jest częścią narzędzi wspomagających prowadzenia projektu – CASE.

**MSF** czyli *Microsoft Solutions Framework* jest to zbiór wskazówek postępowania pogrupowanych wokół podstawowych aspektów prowadzenia prac projektowych - w postaci następujących modeli:

1. model zarządzania ryzykiem (*Risk Management Model*)
2. model procesów prowadzenia projektu (*Process Model*)
3. model zespołu projektowego (*Team Model*)
4. model architektury przedsiębiorstwa (*Enterprise Architecture Model*)
5. model procesu projektowania (*Design Process Model*)
6. model aplikacji (*Application Model*)

**Słownik**

[analiza - faza w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Analiza) - początkowy etap prac projektowych polegający na identyfikacji istotnych składników potrzebnych do utworzenia systemu informacyjnego obsługującego pewną dziedzinę działalności firmy, organizacji lub innego działającego układu. Na tym etapie skupiamy naszą uwagę na odpowiedziach na pytania "co", pozostawiając na później pytania "jak".  
[projektowanie - faza w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Proj) - etap prac projektowych następujący po etapie analizy polegający na transformacji jej wyników na postać układu składników docelowego systemu informacyjnego pokazującego jego schemat i zasady działania.  
[dokumentacja projektowa](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#Dokumentacja) - opis wyników prac projektowych w postaci ustandaryzowanych dokumentów tekstowych lub multimedialnych.  
[repozytorium projektowe](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/wyklad8/w8a.htm#repozytorium) - baza danych zawierająca wszystkie obiekty (w tym dokumenty) używane wspólnie przez cały zespół projektowy przez cały okres działalności projektowej.  
[MSF (Microsoft Solutions Framework)](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Organizacja) - metodyka projektowa w postaci zbioru wskazówek postępowania pogrupowanych wokół podstawowych aspektów prowadzenia prac projektowych takich jak zarządzanie ryzykiem, organizacja zespołu projektowego, podział na fazy i poziomy, model aplikacji oparty na serwisach.  
[ryzyko w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Ryzyko) - aktywne zarządzanie ryzykiem polega na  identyfikowaniu potencjalnych przyczyn niepowodzenia projektu i podejmowaniu działań w celu zapobiegania lub minimalizowania wpływu ryzyka na projekt.  
[kamień milowy w projekcie](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#kamien) - punkt kontrolnyw pracach projektowych służący do oceny wyników (ich akceptacji lub odrzucenia) i przejścia do kolejnego etapu prac.  
[serwis w aplikacji](https://edu.pjwstk.edu.pl/wyklady/wspbd/scb/W1.htm#Apl) - podstawa modelu aplikacji jako logicznej sieci współpracujących, rozproszonych serwisów - podlegających zasadzie wielokrotnego użycia. Są trzy rodzaje serwisów: użytkownika,  biznesowe i  dotyczące danych.

**Literatura do wykładu**

* Microsoft Solutions Framework, materiały do kursu firmy Microsoft.
* Materiały do wykładów z przedmiotu "Analiza systemowa" prowadzanego w PJWSTK przez dr Andrzeja Jankowskiego w 1999r. (Przykładowy Pakiet Dostaw.)

# SQL

W oprogramowaniu Microsoft SQL Server zaimplementowano język SQL w oparciu o standard SQL-92. W związku z tym, zdecydowana większość instrukcji omówionych w sykładzie "Systemy baz danych", będzie działać również na serwerze SQL Microsoftu.

Instrukcja SELECT

Podstawowe różnice:

* Nie ma operatora MINUS i EXCEPT
* Nie ma operatora INTERSECT
* Operatorem konkatenacji jest + a nie ||

CREATE TABLE

Polecenie CREATE TABLE opisane w kursie SBD będzie działać poprawnie w MS SQL Server. Istotną różnicą jest możliwość używania właściwości IDENTITY dla kolumn. Dzięki temu, serwer będzie automatycznie generował kolejne numery (np. wartości klucza głównego). Np.:

CREATE TABLE Osoby (IdOsoby INT PRIMARY KEY IDENTITY, Imie VARCHAR(20), Nazwisko VARCHAR(30))

W tak utworzonej tabeli, wartości klucza głównego IdOsoby będą generowane automatycznie.

Tabele tymczasowe

W Microsoft SQL Server istnieją 2 rodzaje tabel tymczasowych. Pierwsze z nich to lokalne tabele tymczasowe. Nazwa takiej tabeli powinna mieć postać "#nazwa\_tabeli". Jest ona widoczna tylko w sesji, w której została utworzona. Po zakończeniu sesji tabela jest usuwana. Globalne tabele tymczasowe różnią się tym, że są widoczne również spoza sesji, w której zostały utworzone. Nazwa takiej tabeli ma postać "##nazwa\_tabeli". Do tworzenia tabel tymczasowych używamy polecenia CREATE TABLE w sposób analogiczny do normalnych tabel.

Transakcje

Do używania transakcji w podstawowym zakresie służą następujące instrukcje:

BEGIN TRANSACTION nazwa\_transakcji - rozpoczęcie transakcji

COMMIT TRANSACTION nazwa\_transakcji - zatwierdzenie transakcji

ROLLBACK TRANSACTION nazwa\_transakcji - wycofanie transakcji

Ustawianie poziomów izolacji działa analogicznie do Oracle. Dostępne są 4 standardowe poziomy izolacji: READ COMMITTED, READ UNCOMMITTED, REPEATABLE READ, SERIALIZABLE.

Domyślnie ustawiona jest opcja IMPLICIT\_TRANSACTIONS na OFF. Przy takim ustawieniu system zakłada transakcję na każdą instrukcję DML i zatwierdza ją (autocommit). Aby wyłączyć automatyczne zakładanie i zatwierdzanie transakcji, należy użyć:

SET IMPLICIT\_TRANSACTIONS ON

Istotną opcją jest również XACT\_ABORT. Przy domyślnym ustawieniu, gdy w trakcie wykonywania transakcji pojawi się błąd, transakcja kontynuuje działanie, a wycofywana jest jedynie błędna instrukcja. Przy ustawieniu SET XACT\_ABORT ON, przy wystąpieniu błędu, wycofywana jest cała transakcja.

# TransactSQL - podstawy

Proceduralnym rozszerzeniem języka SQL w Microsoft SQL Server jest TransactSQL. Jest to odpowiednik PL/SQL w Oracle'u.

W Transact-SQL nie istnieje określona konstrukcja bloku anonimowego. Pisząc program, umieszczamy kolejne linie kodu jedna po drugiej. Tak utworzony program jest uruchamiany, gdy napotkana zostanie komenda GO. Wszystkie zadeklarowane lokalnie zmienne są widoczne tylko do najbliższego wystąpienia GO.

W Transact-SQL nie musimy kończyć każdej linii średnikiem. Instrukcje mogą znajdować się jedna po drugiej. Dla większej czytelności kodu zalecane jest jednak umieszczanie każdej instrukcji w oddzielnej linii i używanie wcięć.

Deklaracje zmiennych

Zmienne użytkownika są zawsze poprzedzone znakiem "@". Deklaracje zmiennych mają następującą postać:

DECLARE @zmienna TYP

W jednej instrukcji DECLARE możemy zadeklarować wiele zmiennych. W odróżnieniu od PL/SQL, deklaracje mogą występować w dowolnym miejscu. Instrukcja DECLARE może występować kilkakrotnie. Przykład:

DECLARE @Imie VARCHAR(20), @Nazwisko VARCHAR(50)

Zmienne systemowe

Zmienne systemowe oznaczone są dwoma znakami "@". Poniżej omówionych zostało kilka często używanych zmiennych systemowych:

@@ERROR - numer ostatniego błędu  
@@FETCH\_STATUS - czy kursor pobrał wiersz (0 gdy pobrał)  
@@IDENTITY - zawiera ostatnio wygenerowaną wartość IDENTITY (bardzo przydatne, gdy chcemy sprawdzić ID wstawionego przez INSERT wiersza)  
@@ROWCOUNT - zwraca liczbę wierszy, na których operowała ostatnia instrukcja  
@@VERSION - zwraca informację o serwerze SQL

SELECT w Transact SQL

Odpowiednikiem konstrukcji "SELECT wyrażenie INTO zmienna ..." używanej w PL/SQL, a zarazem konstrukcją umożliwiającą wykonanie przypisania wartości do zmiennej jest:

SELECT @zmienna = wyrażenie

Tej konstrukcji możemy używać jako przypisania pewnych wartości do zmiennych. Nie używamy wtedy FROM np.:

SELECT @nazwisko = 'Kowalski', @imie = 'Jan'

Ten sam efekt możemy uzyskać przy pomocy instrukcji SET, z tym że w jednej instrukcji możemy dokonać tylko jednego przypisania:

SET @nazwisko = 'Kowalski'

SET @imie = 'Jan'

Możemy również przypisać zmiennym wartości zwrócone przez zapytanie SQL np.:

SELECT @nazwisko = Nazwisko, @imie = Imie FROM Osoby WHERE IdOsoby = 1234

Należy pamiętać, że zapytanie powinno zwracać jeden wiersz. Gdy jednak zapytanie zwróci więcej wierszy, do zmiennych zostaną przypisane wartości z ostatniego rekordu.

# TransactSQL - Instrukcje sterujące

Instrukcje warunkowe

W Transact SQL instrukcją warunkową jest podobnie jak w wielu innych językach instrukcja IF. Ma ona następującą składnię:

IF warunek

ciąg instrukcji

ELSE

ciąg instrukcji

Ciągiem instrukcji może być pojedyncza instrukcja, lub kilka instrukcji znajdujących się pomiędzy słowami BEGIN i END, np.:

IF @pensja > 0

BEGIN

INSERT INTO Emp (Ename, Sal) VALUES (@nazwisko, @pensja)

PRINT 'Wstawiono produkt'

END

ELSE

PRINT 'Niepoprawna cena'

Instrukcje pętli

Instrukcją pętli w Transact SQL jest WHILE. Ma ona następującą składnię:

WHILE warunek

ciąg instrukcji

Istnieje możliwość wyjścia z pętli używając instrukcji BREAK. Instrukcja CONTINUE powoduje, że reszta instrukcji w pętli jest ignorowana i następuje wykonanie kolejnej iteracji, np.:

WHILE (SELECT AVG(Sal) FROM Emp) < 200

BEGIN

UPDATE Emp

SET Sal = Sal \* 1,2

IF (SELECT MAX(Sal) FROM Emp) > 400

BREAK

ELSE

CONTINUE

END

W powyższym przykładzie użycie instrukcji CONTINUE nie jest konieczne.

# TransactSQL - kursory

Aby użyć kursora w Transact SQL, należy wykonać następujące kroki:

Deklaracja kursora:

DECLARE nazwa\_kursora CURSOR FOR instrukcja\_SELECT

Otwarcie kursora:

OPEN nazwa\_kursora

Pobranie kolejnego wiersza (w Transact SQL nie istnieją zmienne typu wierszowego, więc musimy wcześniej zadeklarować tyle zmiennych, ile zwróci pól instrukcja SELECT kursora):

FETCH NEXT FROM nazwa\_kursora INTO zmienne

Do sprawdzenia, czy instrukcja FETCH zwróciła wiersz, służy zmienna systemowa @@FETCH\_STATUS. Zwykle używamy jej w warunku pętli:

WHILE @@FETCH\_STATUS = 0

Zamknięcie kursora i zwolnienie zasobów:

CLOSE nazwa\_kursora

DEALLOCATE nazwa\_kursora

Instrukcja CLOSE zamyka kursor i zwalnia blokady, które zakładał kursor na danych, na których operował, jednak jest on jeszcze dostępny po ponownym otwarciu. DEALLOCATE powoduje całkowite usunięcie kursora, przykład:

DECLARE kursor CURSOR FOR SELECT Ename, Sal FROM Emp WHERE Sal > 200

DECLARE @nazwisko VARCHAR(50), @pensja INT

PRINT 'Pracownicy o pensji wyższej niż 200:'

OPEN kursor

FETCH NEXT FROM kursor INTO @nazwisko, @pensja

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT @nazwisko + ' ' + @pensja

FETCH NEXT FROM kursor INTO @nazwisko, @pensja

END

CLOSE kursor

DEALLOCATE kursor

# TransactSQL - procedury

W Microsoft SQL Server najwygodniej jest tworzyć i modyfikować procedury używając programu Enterprise Manager, rozwijając w drzewie obiektów bazy danych gałąź "Stored procedures". Możemy również wpisywać je ręcznie przy pomocy np. Query Analyzer. Definicja procedury ma następującą składnię:

CREATE PROCEDURE nazwa\_procedury

lista parametrów

AS

instrukcje Transact SQL

Definiując listę parametrów podajemy nazwę poprzedzoną znakiem "@" oraz typ danych. Możemy podać wartość domyślną oraz określić, że jest to parametr wyjściowy (OUTPUT). Definicja parametru procedury ma postać:

@nazwa\_parametru TYP [=wartość\_domyślna] [OUTPUT]

Przykład. Dla podanego Empno, procedura modyfikuje zarobki o zadany procent (domyślnie 20) oraz zwraca poprzez parametr wyjściowy nową wartość:

CREATE PROCEDURE pensja

@Empno INT,

@Nowe\_zarobki DECIMAL(6,2) OUTPUT,

@Procent INT = 20

AS

SELECT @Nowe\_zarobki = Sal + Sal \* @Procent / 100 FROM Emp WHERE Empno = @Empno

UPDATE Emp SET Sal = @Nowe\_zarobki WHERE Empno = @Empno

Aby wywołać procedurę, pobrać i wyświetlić wartość z parametru wyjściowego, piszemy np.:

DECLARE @Nowe\_zarobki DECIMAL(6,2)

EXECUTE pensja 1234, @Nowe\_zarobki OUTPUT

PRINT @Nowe\_zarobki

Używając procedur możemy przekazać wartość do aplikacji, która wywołała procedurę, w inny sposób, używając instrukcji RETURN. Instrukcja ta powoduje natychmiastowe wyjście z procedury. Może przekazać wartość typu INT. Rozważmy przykład:

CREATE PROCEDURE Ile\_pracownikow AS

DECLARE @ile INT

SELECT @ile = COUNT(\*) FROM Emp

RETURN @ile

Procedura zliczy wszystkie wiersze z tabeli "Emp" i zwróci wartość poprzez RETURN. Do wartości zwróconej w ten sposób możemy się odwołać w następujący sposób:

EXECUTE @zmienna = Ile\_pracownikow

Trzecim sposobem, w jaki procedura może zwracać dane jest użycie tzw. "result set", czyli zestawu rekordów będących wynikiem działania instrukcji SELECT. Należy pamiętać, że przekazywany jest wynik ostatniej instrukcji SELECT w procedurze.

# TransactSQL - wyzwalacze

Składnia i zasada działania wyzwalaczy w Oracle i MS SQL Server dość znacznie się różnią. W serwerze Microsoftu wyzwalacz jest zawsze wywoływany po wykonaniu instrukcji. Nie istnieją wyzwalacze typu "before". Nie istnieją również wyzwalacze typu "FOR EACH ROW". Wyzwalacz jest zawsze uruchamiany dla całej instrukcji DML. Odwołania do starych i nowych wartości modyfikowanych danych są możliwe dzięki wirtualnym tabelom o nazwach "inserted" i "deleted", zawierających odpowiednio nowe i stare wartości. Aby się do nich odwołać, musimy użyć instrukcji SELECT tak, jakby były to normalne tabele bazy danych. Modyfikacja danych bezpośrednio w tych tabelach nie jest możliwa. Istnieje możliwość sprawdzenia, jaka kolumna była modyfikowana używając IF UPDATE(nazwa\_kolumny). Aby utworzyć wyzwalacz, piszemy:

CREATE TRIGGER nazwa\_wyzwalacza

ON tabela

FOR instrukcje

AS

instrukcje Transact-SQL

Instrukcje, które mogą wywołać wyzwalacz to: INSERT, UPDATE i DELETE. Po słowie FOR możemy napisać jedną, dwie lub wszystkie trzy (oddzielone przecinkami).

W poniższym przykładzie wyzwalacz nie pozwoli usunąć wiersza z tabeli. Ponieważ jednak wyzwalacz jest uruchamiany po wykonaniu operacji DELETE, musimy przy pomocy instrukcji ROLLBACK wycofać wprowadzone zmiany:

CREATE TRIGGER wyzw ON Emp

FOR DELETE

AS

ROLLBACK

W kolejnym przykładzie, chcielibyśmy aby nowo wstawiani pracownicy nie mogli mieć zarobków równych zero. Gdy taki pracownik zostanie wprowadzony, wyzwalacz usunie go i zgłosi błąd:

CREATE TRIGGER wyzw ON Emp

FOR INSERT

AS

DECLARE @sal DECIMAL(6,2)

SELECT @sal = sal FROM inserted

IF @sal = 0

BEGIN

ROLLBACK

RAISERROR('Nie można wstawić pracownika z pensją równą zero!',1,2)

END

Składnia instrukcji RAISERROR wygląda następująco:

RAISERROR(message, severity, state)  
  
Gdzie:  
Message - dowolny tekst (komunikat błędu)  
Severity - liczba z przedziału 0-25 (przy czym użytkownik może używać wartości z przedziału 0-18  
State - liczba z przedziału 1-127

Wartości parametrów "severity" i "state" są przekazywane do aplikacji klienta, dzięki czemu różne błędy mogą być obsługiwane w różny sposób.

Rozważmy kolejny przykład. Chcielibyśmy aby przy zwiększeniu pensji, była zerowana prowizja:

CREATE TRIGGER wyzw ON Emp

FOR UPDATE

AS

DECLARE @sal\_old DECIMAL(6,2), @sal\_new DECIMAL(6,2), @id INT

SELECT @sal\_old = sal, @id = Empno FROM deleted

SELECT @sal\_new = sal FROM inserted

IF @sal\_old < @sal\_new

UPDATE Emp SET Comm = 0 WHERE Empno = @id

# Funkcje wbudowane i procedury systemowe

Funkcje wbudowane

Poniżej przedstawiono kilka wybranych funkcji, z których możemy korzystać w zapytaniach SQL i programach Transact-SQL:

CONVERT(typ\_danych, wyrażenie [, styl]) - dokonuje konwersji typów danych (styl jest używany przy konwersji do daty, typów walutowych itp.)  
DATEDIFF(część\_daty, data1, data2) - oblicza różnicę między datami (częścią daty może być np. year, month, day, hour, minute, second)  
GETDATE() - zwraca aktualną datę systemową  
LEFT(string, ile\_znaków) - zwraca określoną ilość znaków tekstu rozpoczynając od lewej  
LEN(string) - zwraca długość tekstu  
REPLACE(string, string\_szukany, string\_do\_zamiany) - wyszukuje i zamienia fragment tekstu  
RIGHT(string, ile\_znaków) - zwraca określoną ilość znaków tekstu rozpoczynając od prawej  
SUBSTRING(string, od, do) - zwraca określoną część tekstu

Procedury systemowe

Procedury systemowe znajdują się w bazach master i msdb. Można ich używać z poziomu każdej bazy danych. Do ich wykonania nie jest wymagane użycie polecenia EXECUTE. Przy pomocy procedur systemowych możemy łatwo wyciągnąć informację z tabel systemowych. Niektóre z nich to:

Sp\_tables - wypisuje tabele bazy danych  
Sp\_help nazwa\_obiektu - wypisuje informację na temat obiektu (np. tabeli, perspektywy, procedury)  
Sp\_helptext nazwa\_obiektu - wypisuje treść obiektu (np. procedury)  
Sp\_helpdb nazwa\_bazy - wypisuje informacje na temat bazy danych  
Sp\_helpindex nazwa\_tabeli - wypisuje informacje na temat indeksów założonych na tabeli  
Sp\_spaceused nazwa\_obiektu - wypisuje przestrzeń wykorzystywaną przez obiekt