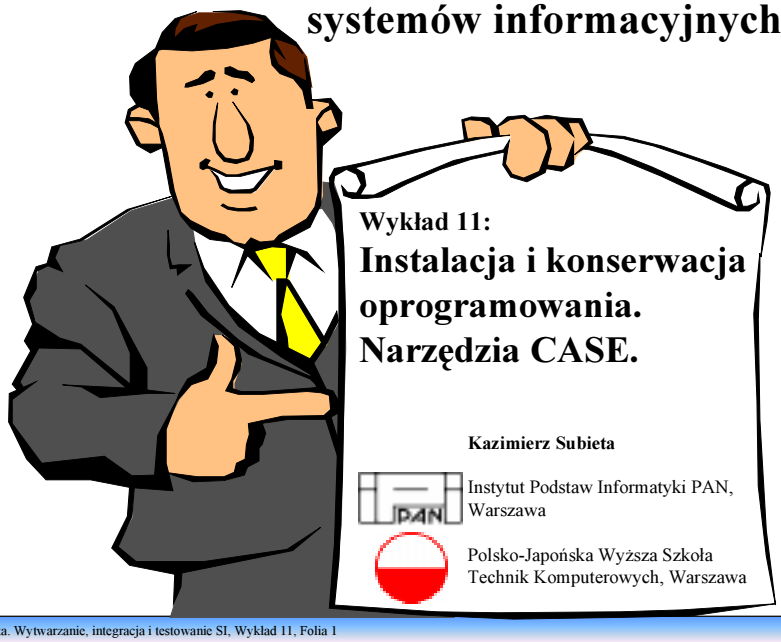
Wytwarzanie, integracja i testowanie systemów informacyjnych



Wykład 11:

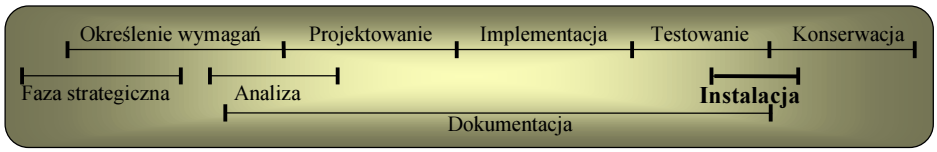
Instalacja i konserwacja oprogramowania. Narzędzia CASE.

Kazimierz Subieta

Instytut Podstaw Informatyki PAN, Warszawa

Polsko-Japońska Wyższa Szkoła Technik Komputerowych, Warszawa

Instalacja



Na fazę instalacji składają się:

- Szkolenie użytkowników końcowych i administratorów systemu
- Instalacja sprzętu i przeniesienie oprogramowania
- Wypełnienie baz danych
- Nadzorowane korzystanie z systemu, często równoległe z tradycyjnym sposobem pracy
- Usuwanie błędów w oprogramowaniu i dokumentacji użytkowej
- Przekazanie systemu klientowi

Problemy podczas instalacji

Szkolenie użytkowników: zaleca się, aby przeprowadzały je osoby, które były zaangażowane w prowadzenie przedsięwzięcia. Osobom tym będzie łatwiej nawiązać kontakt z przyszłymi użytkownikami.

Wypełnienie bazy danych jest często bardzo żmudnym procesem, wymagającym wprowadzenia danych z nośników papierowych. Niekiedy część danych jest w formie elektronicznej - wtedy z reguły potrzebne są specjalne programy konwersji. Konwersja jest łatwiejsza, jeżeli znana jest specyfikacja struktury starej BD.

Ważne jest planowanie i harmonogramowanie prac. W tej fazie pojawia się szereg problemów, np. konieczność usunięcia błędów i wprowadzenia modyfikacji. Z reguły, wykonawcy systemu nie mogą zarezerwować w pełni swojego czasu na prace związane z instalacją. Z drugiej strony, użytkownicy nie mogą zaniechać wykonywania przez nich bieżących prac.

Pewien opór klienta przed zmianą sposobu pracy. Często użytkownicy systemu są to osoby po raz pierwszy stykający się z systemem (inni niż ci, którzy uczestniczyli w poprzednich fazach). Ważne jest uzyskanie ich akceptacji.

Kluczowe czynniki sukcesu w fazie instalacji

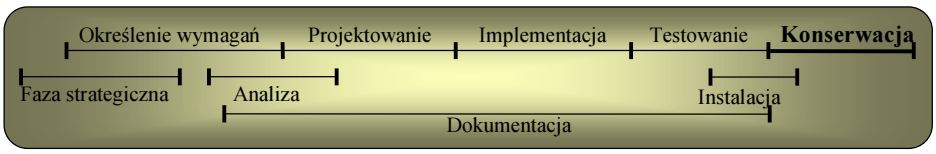
- Właściwe planowanie i harmonogramowanie zadań, które zapewni szybki przebieg instalacji przy jednoczesnym nie zakłócaniu pracy klienta i producenta.
- Uzyskanie wstępnej pozytywnej reakcji użytkowników.

Podstawowe rezultaty fazy implementacji

- Poprawiony kod, projekt, model i specyfikacja wymagań
- Oprogramowanie zainstalowane u użytkownika.

Konserwacja oprogramowania

maintenance



(Używa się również terminów “pielęgnacja” oraz “utrzymanie”.)

Konserwacja oprogramowania polega na wprowadzeniu modyfikacji.
Istnieją trzy główne klasy wprowadzanych w oprogramowaniu modyfikacji:

- **Modyfikacje poprawiające**: polegają na usuwaniu z oprogramowania błędów popełnionych w fazach wymagań, analizy, projektowania i implementacji .
- **Modyfikacje ulepszające**: polegają na poprawie jakości oprogramowania.
- **Modyfikacje dostosowujące**: polegają na dostosowaniu oprogramowania do zmian zachodzących w środowisku jego pracy

Modyfikacje ulepszające

- Poprawa wydajności pewnych funkcji
- Poprawa ergonomii interfejsu użytkownika
- Poprawa przejrzystości raportów

Modyfikacje dostosowujące wynikają z:

- Zmiany wymagań użytkowników
- Zmian przepisów prawnych dotyczacych dziedziny problemu
- Zmian organizacyjnych po stronie klienta
- Zmian sprzętu i oprogramowania systemowego

Zaleca się, aby wprowadzanie modyfikacji polegało na powrocie do najwcześniejszych faz analizy, na której rezultaty wpływa dana zmiana w oprogramowaniu.

Analiza sensowności wprowadzenia modyfikacji

Analiza powinna uwzględniać:

- Znaczenie wprowadzenia zmiany dla użytkowników
- Koszt wprowadzenia zmiany
- Wpływ zmiany na poszczególne składowe systemu
- Wpływ zmiany na poszczególne składowe dokumentacji technicznej.

Dopiero po dokonaniu oceny zmiany podejmowana jest decyzja o jej ewentualnej realizacji. W przypadku bardzo dużych przedsięwzięć może zostać powołana w tym celu specjalna komisja.

Nie wprowadzać każdej zmiany natychmiast. Zaleca się grupowanie zmian, których wykonanie prowadzi do nowej wersji systemu.

Koszty konserwacji oprogramowania

Występuje tendencja do tego, aby niżej oceniać koszt konserwacji niż koszt wytworzenia oprogramowania. Koszty konserwacji są jednak czesto duże. Niedocenianie nakładów pracy na fazę konserwacji jest jedną z głównych przyczyn opóźnień przedsięwzięć programistycznych.

Obiektywne czynniki wpływające na koszty konserwacji:

- **Stabilność środowiska w którym pracuje system.** Zmiany zachodzące w przepisach prawnych, zmiany struktury organizacyjnej i sposobów działania po stronie klienta prowadzą do zmian wymagań wobec systemu.
- **Stabilność platformy sprzętowej i oprogramowania systemowego**
- **Czas użytkowania systemu**. Całkowite koszty konserwacji rosną, gdy system jest eksploatowany przez dłuższy czas.

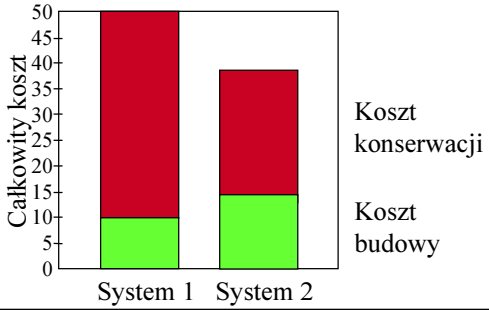
Czynniki redukcji kosztów konserwacji (1)

- **Znajomość dziedziny problemu**. Jeżeli analitycy pracujący nad systemem dobrze znają daną dziedzinę problemu, mają mniej trudności z właściwym zebraniem wymagań oraz budową oddającego rzeczywistość modelu.
- **Wysoka jakość modelu i projektu**, w szczególności jego spójność, stopień powiązania składowych oraz przejrzystość.
- **Wysoka jakość dokumentacji technicznej**. Powinna ona:
  - w pełni odpowiadać systemowi
  - być wystarczająco szczegółowa
  - być zgodna z przyjętymi w firmie standardami.
- **Stabilność personelu**. Niezależnie od jakości dokumentacji, pewne aspekty systemu są znane tylko osobom bezpośrednio uczestniczącym w realizacji. Niekoniecznie muszą one same dokonywać modyfikacji, ale mogą istotnie wspomagać konsultacjami.

Czynniki redukcji kosztów konserwacji (2)

- **Środowisko implementacji**. Zaawansowane środowisko implementacji sprzyja skróceniu czasu niezbędnego na wprowadzenie modyfikacji.
- **Niezawodność oprogramowania**. Wysoka niezawodność oprogramowania przekazanego klientowi zmniejsza liczbę modyfikacji.
- **Inżynieria odwrotna**. Pod tym pojęciem rozumie sie odtwarzanie dokumentacji technicznej na podstawie istniejącego oprogramowania.
- **Zarządzanie wersjami**.

Wiele działań zmierzających do redukcji kosztów konserwacji musi być podjęte już w fazie budowy systemu.



Inżynieria odwrotna

reverse engineering

W wielu przypadkach zachodzi konieczność wprowadzenia zmian w oprogramowaniu, które nie jest opisane dokumentacją techniczną.

Inżynieria odwrotna polega na odtworzeniu dokumentacji technicznej na podstawie istniejącego kodu lub struktury bazy danych.

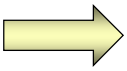
Podczas prób odtworzenia dokumentacji technicznej może się okazać, że wiele konstrukcji zastosowanych w programie nie da się zapisać w przyjętej notacji (jezyki hybrydowe, np. C++)

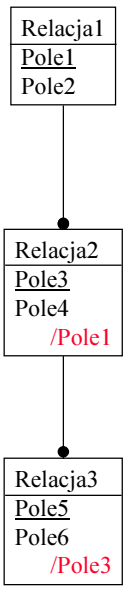
Inżynieria odwrotna daje się automatyzować. Moduły inzynierii odwrotnej są często składowymi narzędzi CASE.

Pewne elementy mogą być jednak niejednoznaczne. Np. implementacja asocjacji może prowadzić do dodatkowych pól, których interpretacja wynika z semantyki danych i nie jest odwzorowana formalnie. W takich sytuacjach automatycznie odtworzoną dokumentację należy poprawić ręcznie.

Odtworzenie diagramu na podstawie SQL

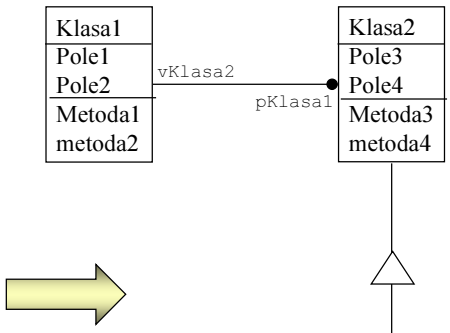
```
CREATE TABLE Relacja1(
         Pole1 TEXT NOT NULL,
         Pole2 TEXT NOT NULL,
         PRIMARY KEY ( Pole1 )
)
CREATE TABLE Relacja2(
         Pole3 TEXT NOT NULL,
         Pole4 DATE NOT NULL,
         Pole1 TEXT NOT NULL,
         PRIMARY KEY ( Pole3 )
)
ALTER TABLE Relacja2 (
         ADD FOREIGN KEY ( Pole1 )
         REFERENCES Relacja1 ( Pole1 )
            ON DELETE CASCADE
)
CREATE TABLE Relacja3(
         Pole5 TEXT NOT NULL,
         Pole6 DECIMAL NOT NULL,
         Pole3 TEXT NOT NULL,
         PRIMARY KEY ( Pole5 )
)
ALTER TABLE Relacja3 (
         ADD FOREIGN KEY ( Pole3 )
         REFERENCES Relacja2 ( Pole3 )
)
```





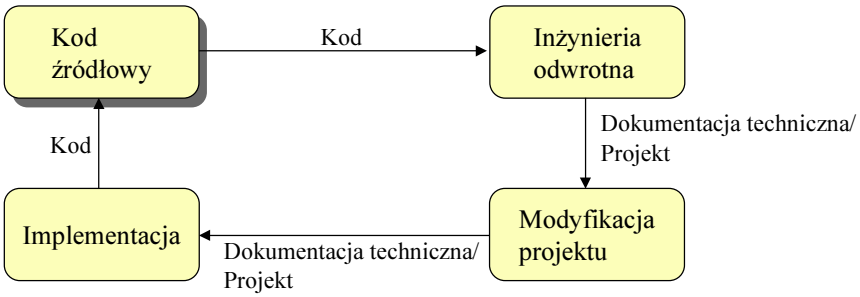
Odtworzenie diagramu na podstawie C++

```
class Klasa1
{
private: long Pole2;
         double Pole2;
public:  void Metoda1();
         void Metoda2();
protected:
         Vector<Klasa2 *> vKlasa2;
};
class Klasa2
{
private: unsigned short Pole3;
         float Pole4;
public:  void Metoda3();
         void Metoda4();
protected:
         Klasa1 * pKlasa1;
};
class Klasa3 : public Klasa2
{
private: long Pole5;
         double Pole6;
public:  void Metoda5();
         void Metoda6();
};
```



W istocie, z podanej specyfikacji w C++ nie wynika, że pKlasa1 jest odwrotnością vKlasa2. C++ nie daje możliwości określenia tego ograniczenia.

Fazy reinżynierii oprogramowania



Faza konserwacji jest ostatnim etapem przedsięwzięcia programistycznego. Jeżeli po pewnym czasie narasta potrzeba dokonania istotnych modyfikacji, wówczas możliwe są dwa wyjścia:

- zaprzestanie korzystania z systemu
- rozpoczęcie prac nad nową wersją (nowe przedsięwzięcie)

Kluczowe czynniki sukcesu fazy konserwacji

- Wysoka jakość definicji wymagań, modelu i projektu
- Dobra znajomość środowiska implementacji
- Właściwa motywacja osób wykonujacych konserwację oprogramowania
- Właściwe oszacowanie kosztów konserwacji

Podstawowy rezultat:

poprawiony kod, projekt, model i specyfikacja wymagań.

Narzędzia CASE

computer assisted/aided software/system engineering

Pod tym hasłem kryją się wszelkie narzędzia wykorzystywane w trakcie prac nad oprogramowaniem: kompilatory, debuggery, edytory tekstu, narzędzia harmonogramowania przedsięwzięć, arkusze kalkulacyjne.

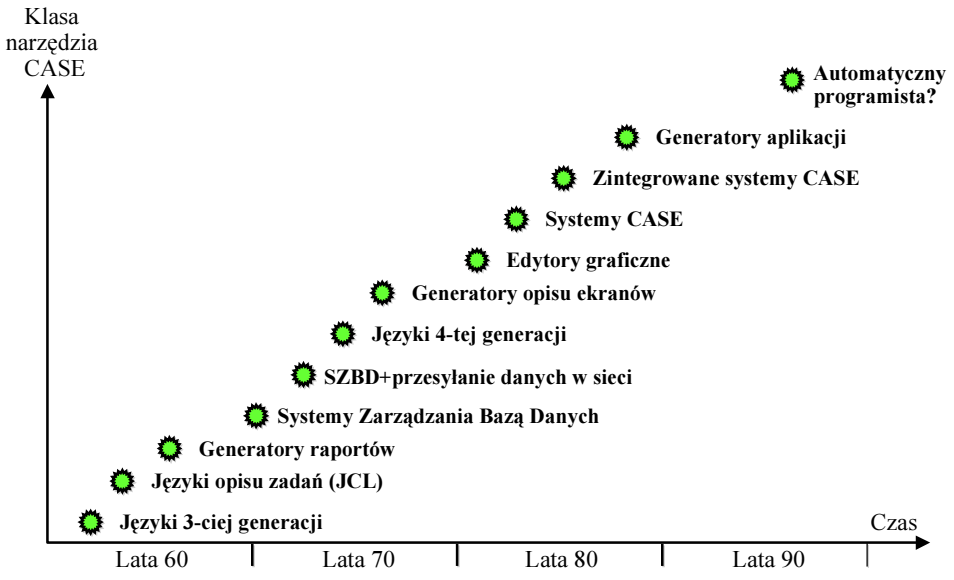
Tradycyjnie pod pojęciem CASE rozumie się narzędzia, które:

- Wspomagają ogólnie rozumiane wytwarzanie oprogramowania
- Koncentrują się na fazach analizy i projektowania oraz bezpośrednim wykorzystaniu wyników tych faz w implementacji.

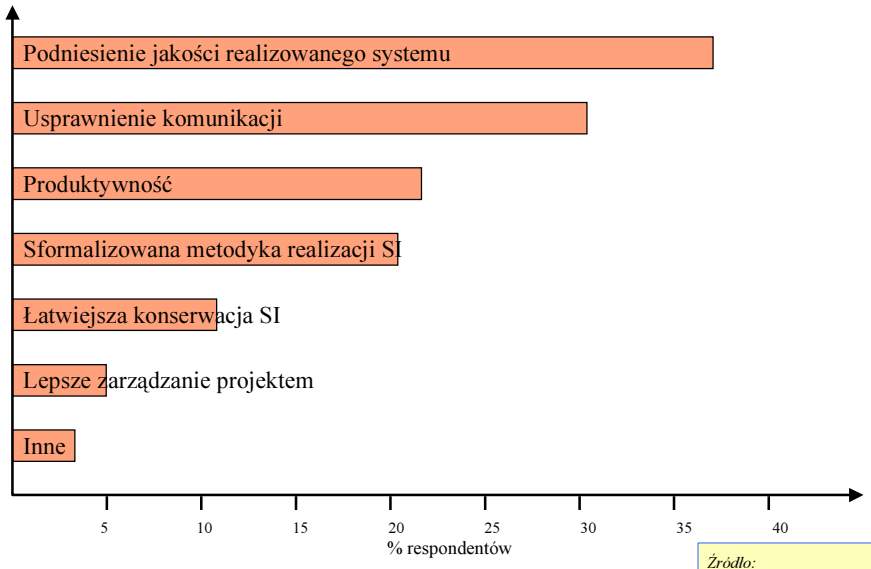
Dwie główne grupy narzędzi CASE:

- **Upper-CASE**: wspomaganie wczesnych faz prac nad oprogramowaniem, w szczególności fazy analizy (potrzeby analityków i projektantów). Narzędzia te nie są związane z konkretnym srodowiskiem implementacyjnym.
- **Lower-CASE**: wspomaganie faz projektowania i implementacji (potrzeby programistów). Narzędzia te są z reguły sciśle związane z konkretnym środowiskiem implementacji.

Historia narzędzi CASE



Korzyści ze stosowania narzędzi CASE



Źródło:
CASE RESEARCH CORP. 1989

Aspekty narzędzi CASE

Narzedzia CASE stosują różne techniki wizualizacji projektów, w szczególności notacje diagramów encja-związek (ER), OMT, UML i inne.

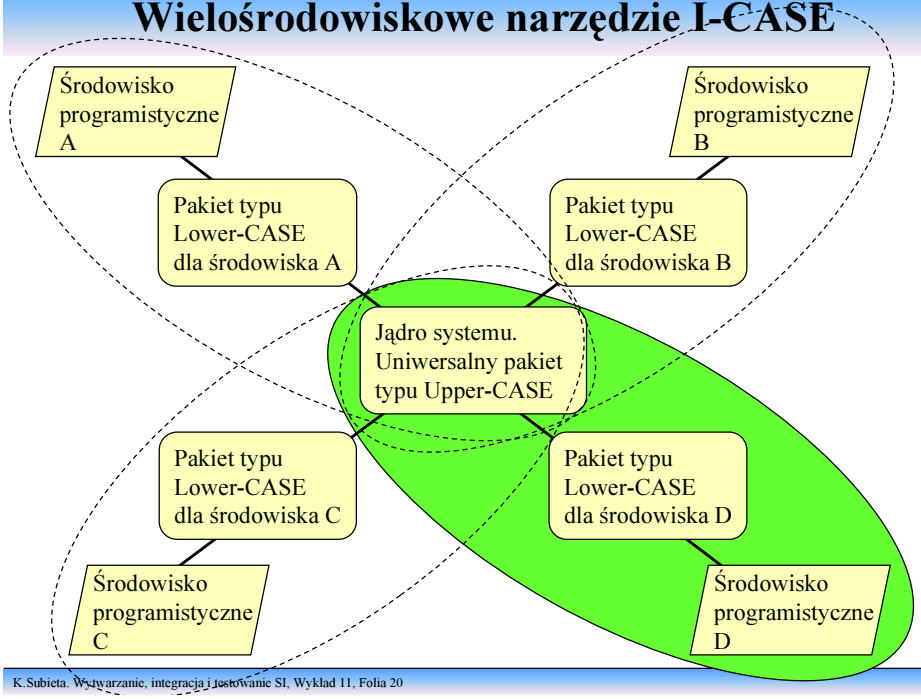
Obecnie większość producentów określa swoje środowiska jako I-CASE (Integrated-CASE). Są to narzędzia łączące w sobie możliwości Lower-CASE i Upper-CASE.

Stosunek narzędzi CASE do konkretnych metodyk i notacji jest dość różny. Istnieje grupa narzędzi uniwersalnych, które umożliwiają pracę z wieloma notacjami i wieloma metodykami. Istnieją również narzędzia CASE przypisane do konkretnych produktów, np. Oracle CASE. Wiele narzędzi CASE łączy elementy znane z wielu metodyk z własnymi pomysłami.

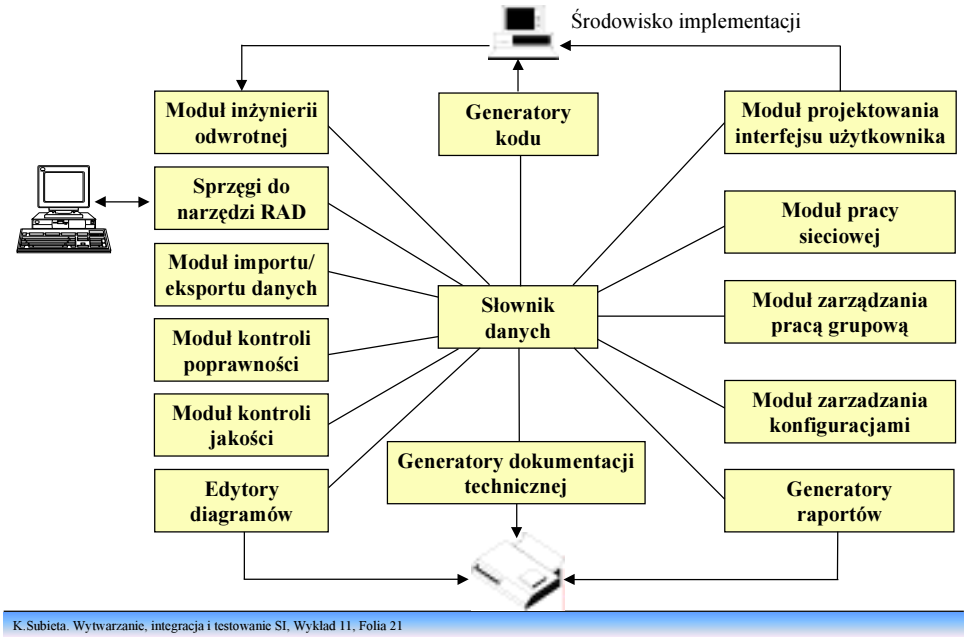
Przykłady narzędzi CASE (są ich dziesiatki):
Oracle CASE, EasyCASE, CASE 4.0, objectiF, Select OMT Professional, System Architect, ObjectTeam, Paradigm Plus

Wiele narzędzi jest niczym więcej niż wyspecjalizowanymi edytorami graficznymi.

Wielośrodowiskowe narzędzie I-CASE



Składowe narzędzi CASE



Funkcje edytorów notacji graficznych

- Tworzenie i edycja diagramów wykorzystywanych w fazach określania wymagań.
- Tworzenie i edycja powiązań pomiędzy poszczególnymi symbolami i diagramami oraz nawigowanie po sieci powiązanych diagramów.
- Wydruk diagramów.

Ocena notacji graficznych:

- Ergonomia pracy. Diagramy graficzne są jednym z podstawowych narzędzi pracy w fazach analizy i projektowania. Powinny one pozwalać analitykom i projektantom skupić się na pracy, a nie na “zmaganiach” z edytorem.
- Możliwość kontrolowania ilości informacji prezentowanej w sposób graficzny.
- Jakość i możliwość formatowania wydruków.
- Wykrywanie na bieżąco konstrukcji niepoprawnych.
- Zapewnienie spójności informacji umieszczonych na różnych diagramach.

Słownik danych

Inaczej repozytorium.
Jest to baza danych o realizowanym projekcie oraz narzędzie służące do jej edycji i przeglądania.

Podstawowe funkcje słownika:

- Wprowadzenie oraz edycja specyfikacji modelu i projektu, a takze innych informacji związanych z przedsięwzięciem.
- Wyszukiwanie pożądanej informacji

W wielu narzędziach CASE słownik jest przechowywany w sposób umożliwiający dostęp z programów zewnętrznych pisanych np. w Visual Basic, SQL, ODBC, itd.

W niektórych narzędziach CASE użytkownik ma możliwość konfigurowania struktury słownika.

Pozostałe moduły narzędzi CASE (1)

- **Moduł kontroli poprawności**
  Pewne błędy mogą być wykrywane na bieżąco w trakcie edycji diagramów i słownika danych., np. uczynienie klasy swoja własną specjalizacją.
- **Moduł kontroli jakości**
  Pewne systemy pozwalają na automatyczną ocenę pewnych miar jakości projektu. Dotyczy to szczególnie złożoności oraz stopnia powiązania składowych.
- **Generator raportów**
  Służy do przygotowania raportów na podstawie zawartości słownika danych. Niektóre raporty są parametryczne. Narzędzia CASE zawierają sporo gotowych generatorów raportów. Niektóre z nich pozwalają na definiowanie własnych.
- **Generator dokumentacji technicznej**
  Moduł służacy do przygotowania dokumentacji technicznej złożonej z szeregu diagramów. Swobodne formatowanie dokumentów. Przykładowe dokumenty. Łatwe i efektywne uaktualnienie dokumentacji po dokonaniu zmian w projekcie.

Pozostałe moduły narzędzi CASE (2)

- **Generatory kodu**
  Narzędzia służące do generacji kodu w rozmaitych jezykach programowania. Często generują szkielety, które muszą byc uzupełnione przez użytkownika dodatkowym kodem. Wygenerowany kod jest uzupełniony o komentarze i inne informacje. Może także zawierać pewne elementy do modyfikacji. Nazwy uzyte w projekcie przechodzą do wynikowego kodu (ewentualnie są skracane).
- **Moduł zarządzania wersjami**
  Umożliwia kontrolę różnych wersji projektu powstających ze względu na konieczność grupowego wprowadzania zmian oraz wskutek wielu środowisk informatycznych (sprzęt i oprogramowania) oraz różnych zastosowań.
- **Moduł projektowania interfejsu użytkownika**
  Dotyczy projektowania dialogów, okien, menu. Zaletą jest wykorzystanie informacji znajdujących się w słowniku danych. Pozwala to np. an automatyczne wygenerowanie dialogu do edycji pewnej struktury danych.
  Integracja z RAD.
- **Moduł inżynierii odwrotnej**

Pozostałe moduły narzędzi CASE (3)

- **Sprzęgi do narzędzi RAD**
  Automatyczne generowanie składowych interfejsu użytkownika, definicja relacji, wiazanie elementów interfejsu uzytkownika ze składowymi bazy danych na podstawie opisu projektu. Możliwy także przepływ informacji w drugą stronę, od pakietu RAD do narzędzia CASE, np. wprowadzenie nowej encji na podstawie relacji zadeklarowanej w pakiecie RAD.
- **Moduł eksportu/importu danych**
  Wymiana informacji pomiędzy różnymi pakietami CASE, np. wg standardu CDIF (*CASE Data Interchange Format*)
- **Moduł zarządzania pracą grupową**,np.
  - dodawanie i usuwanie użytkowników oraz grup uzytkowników z określeniem praw
  - ochronę dostępu do projektu za pomocą haseł
  - określenie praw użytkowników oraz ich grup do odczytu i modyfikacji informacji
  - udostępnienie praw przez uzytkowników dla innych użytkowników
  - zabezpieczanie projektów i ich fragmentów przed przypadkowa zmianą
  - śledzenie pracy poszcególnych użytkowników
- **Moduł pracy sieciowej**
  Umożliwienie pracy w sieci wielu użytkownikom, z podtrzymaniem spójności

Ocena narzędzi CASE

Kryteria:

- Zakres oferowanych funkcji i ich zgodność z potrzebami firmy
- Koszt
- Niezawodność
- Opinia o producencie i dystrybutorze
- Dostępność na rynku pracy specjalistów znających dany pakiet
- Stopień zintegrowania z przyjętym środowiskiem programistycznym
- Wielośrodowiskowość
- Koszt szkoleń
- Koszt dostosowania sprzętu do wymagań pakietu

Obecnie narzędzia oparte o metodyki obiektowe nie ustępują jakością tradycyjnym narzędziom opartym o metodyki strukturalne

Wdrażanie i konfigurowanie pakietu CASE

- Skonfigurowanie pakietu stosownie do potrzeb i zgodnie ze standardami już stosowanymi w firmie. Obejmuje skonfigurowanie słownika danych, zdefiniowanie niezbędnych raportów, zdefiniowanie dokumentów, które będą generowane za pomocą generatora dokumentacji technicznej.
- Dostosowanie standardów firmy do nowej technologii.
- Szkolenie pracowników w zakresie metodyki analizy i projektowania wspomaganej przez pakiet.
- Szkolenie pracowników w zakresie obsługi pakietu.
- Odtworzenie dokumentacji technicznej poprzednich przedsięwzięć. Obejmuje funkcje inżynierii odwrotnej oraz możliwości importu danych ze standardowych formatów.
- Wykonanie pilotowego projektu (projektów) z wykorzystaniem narzędzia CASE, często równolegle do stosowanych wcześniej metod.

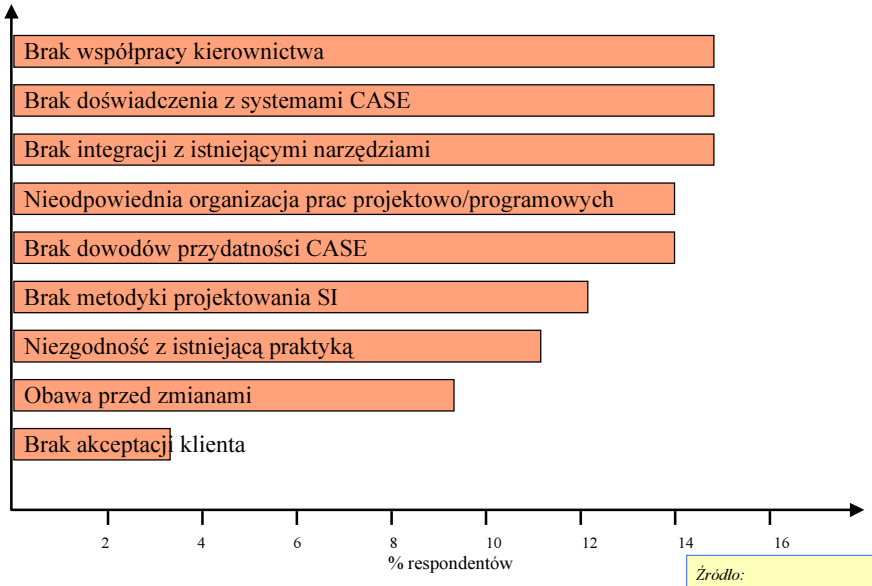
Przyczyny małej efektywności stosowania narzędzi CASE

Narzędzie CASE nie jest cudownym środkiem, który w sposób automagiczny załatwi powodzenie projektu. Zastosowanie obiektowego CASE niekoniecznie oznacza "nowoczesność" projektu. Stosowanie narzędzi CASE często przynosi znikome efekty.

Przyczyny trudności:

- **Traktowanie narzędzi CASE wyłącznie jako generatorów kodu**. Nie jest to efektywne przy braku rzetelnego podejścia do analizy i projektowania:
  - nakłady na implementację stanowią tylko ok. 15-30% całych nakładów
  - koszt błędów popełnionych w fazie implementacji jest stosunkowo niewielki
  - istnieją inne, tańsze narzędzia programistyczne (RAD)
- **Nieznajomość metodyki analizy i projektowania.** Narzędzia CASE nie zwalniają z myślenia, wiedzy i doświadczenia.
- **Niewłaściwa organizacja i zarządzanie przedsięwzięciem**. Nieuporządkowanie prac, brak planu, brak właściwych ocen, brak monitorowania postępu, itd.
- **Zbyt wysokie oczekiwania w stosunku do narzędzia CASE**. Może ono zredukować koszty co najwyżej o 50%, koszt wdrożenia jest wysoki, efekty pojawiają się z pewnym opóźnieniem, wymaga dyscypliny w przedsięwzięciu.

Przeszkody we wdrażaniu CASE



Źródło:
CASE RESEARCH CORP. 1989

Skala trudności zmian

- Metody rozwiązywania problemów
- Organizacje prac projektowo-programowych
- Podejście do projektowania
- Języki programowania
- Oprogramowanie narzędziowe
- System operacyjny
- Sprzęt komputerowy w ramach jednej rodziny
- Standardy dokumentacyjne
- Styl programowania

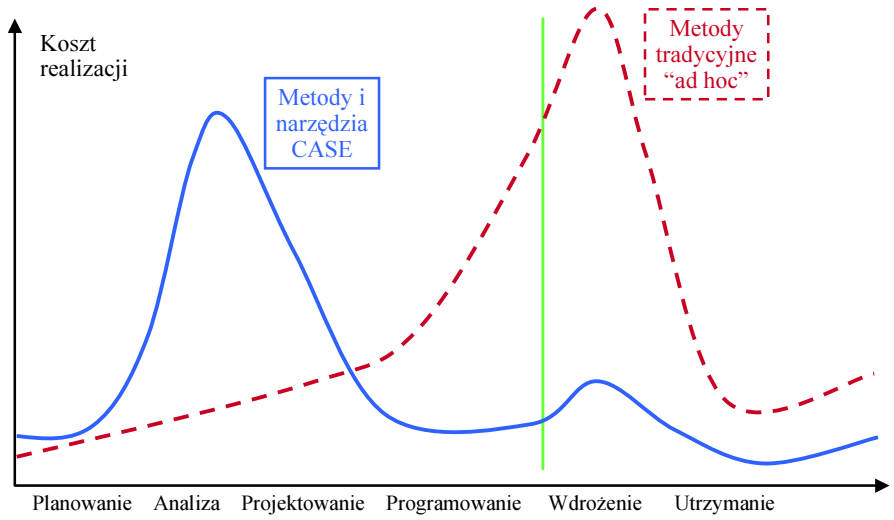
Źródło: Auerbach

Koszty realizacji SI

Rozkład kosztów realizacji SI

Rozkład kosztów naprawy błędów

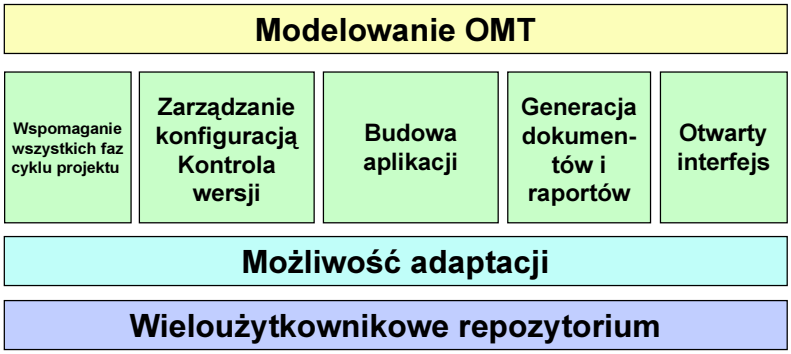
Rozkład kosztów realizacji SI



Kierunki rozwoju narzędzi CASE

- Integracja poszczególnych elementów CASE
- Semantyka modeli wykorzystywanych w CASE
- Interfejs graficzny w CASE
- Inżynieria odwrotna
- Integracja z otoczeniem programistycznym
  - narzędzia, metodyki, zarządzania projektami, ORSZBD, OSZBD
- Projektowanie systemów:
  - Klient-Serwer
  - Obiektowych
  - Komponentowych
  - Multi-medialnych
  - Eksperckich
- Projektowanie rozproszonych baz danych
- Dostosowanie narzędzia CASE do projektu (*customization*)
- Elementy sztucznej inteligencji (?)

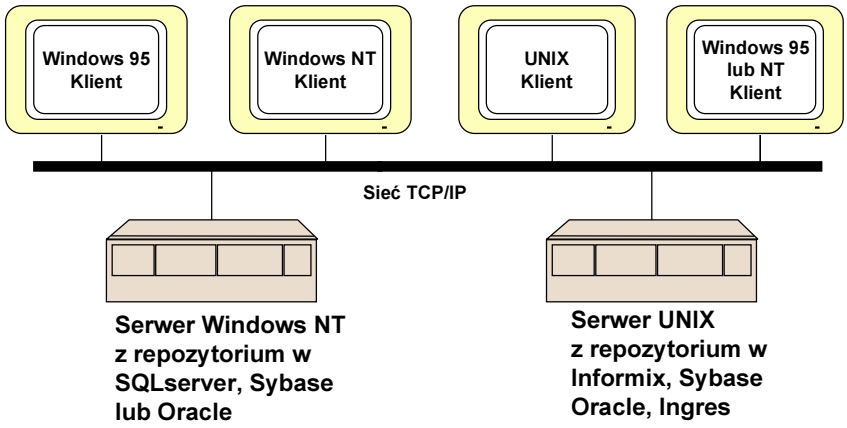
Przykład narzędzia CASE: Architektura ObjectTeam dla OMT



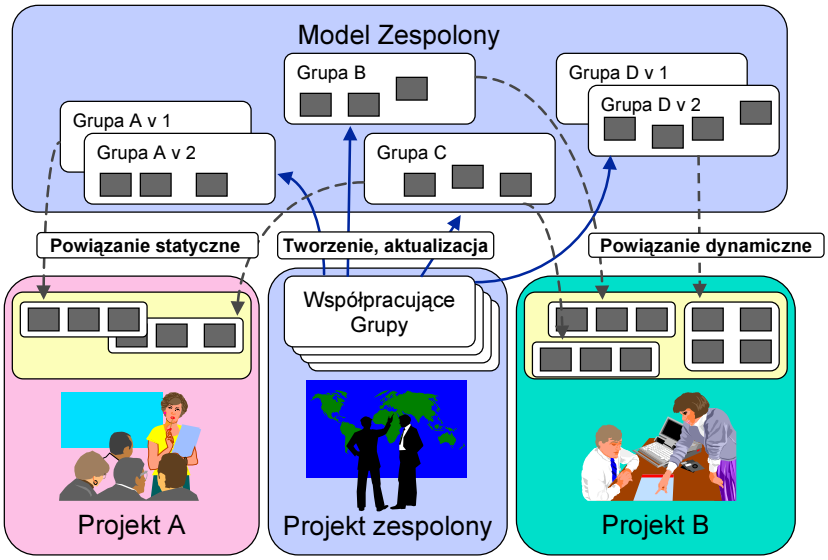
Kluczowe cechy ObjectTeam

- ✓ *Scentralizowane, wieloużytkownikowe repozytorium z kompletnym zarządzaniem konfiguracjami i wersjami*
- ✓ *Konfigurowalne i otwarte środowisko*
- ✓ *Zaimplementowana metodologia OMT z przeznaczeniem do dużych projektów i pracy zespołowej*
- ✓ *Przystosowanie do generowania kodu*
- ✓ *Możliwość ponownego użycia kodu C++ z możliwością zintegrowania z generowanym kodem*
- ✓ *Zintegrowanie z procesorami tekstu*

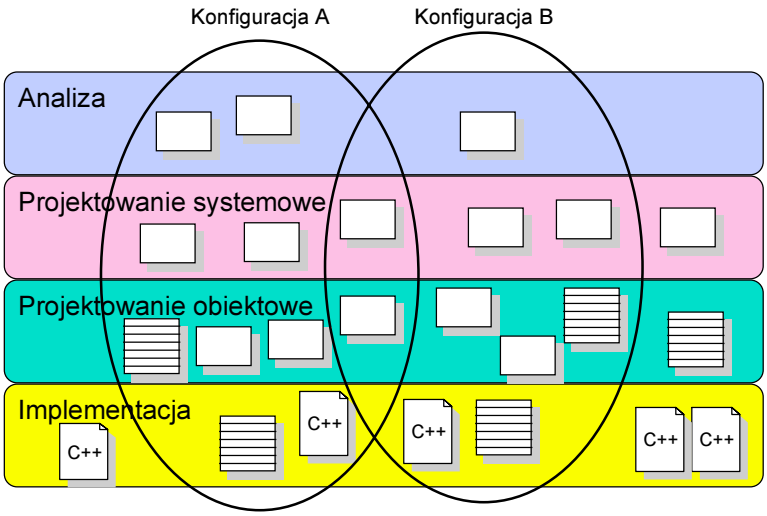
ObjectTeam: Heterogeniczne środowisko klient-serwer



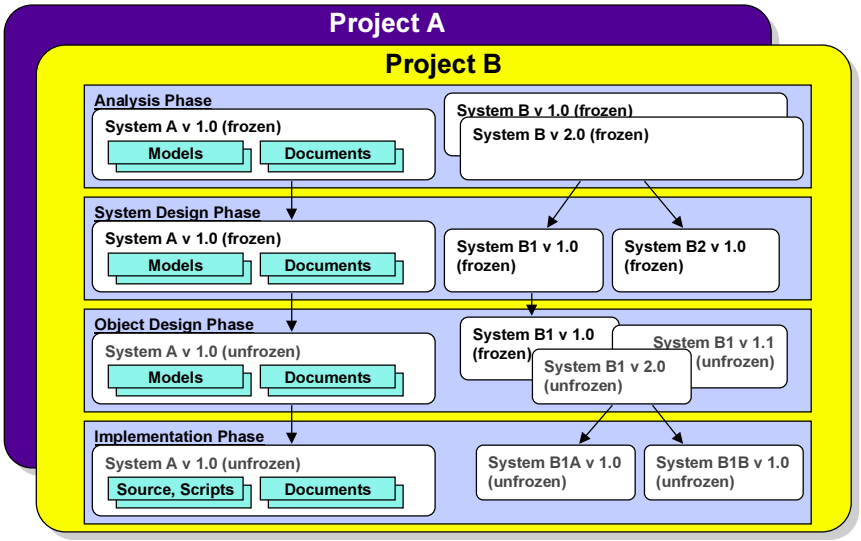
ObjectTeam: Modelowanie pracy zespołowej



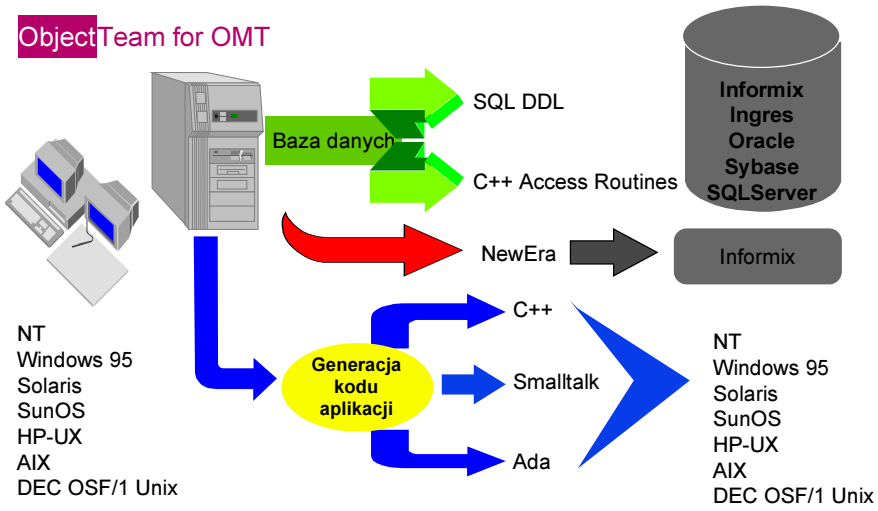
ObjectTeam:Zarządzanie konfiguracjami



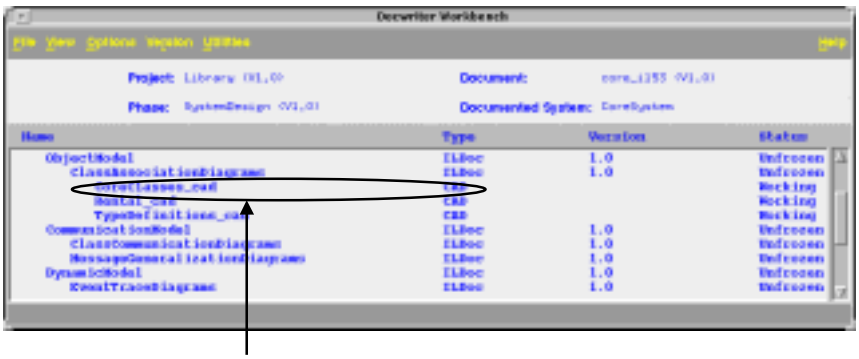
ObjectTeam: Zarządzanie projektami i procesami



ObjectTeam: Wzmocniony generator aplikacji



ObjectTeam: Generowanie dokumentów i zarządzanie dokumentami



Składowe diagramy projektu mogą być identyfikowane w repozytorium