Szczyrk, 22-26.06.1998r.

DZIESIĄTA GÓRSKA SZKOŁA PTI

SZCZYRK '98 — 22-26 CZERWCA 1998

DOSKONALENIE PROCESÓW KONSTRUOWANIA I UŻYWANIA OPROGRAMOWANIA

A INTEGRACJA SYSTEMÓW INFORMACYJNYCH

Szanowni Państwo

Oddział Górnośląski Polskiego Towarzystwa Informatycznego jest gospodarzem już dziesiątej Górskiej Szkoły. Po dwóch latach wróciliśmy do Szczyrku, gdzie tak, jak już od paru lat w Noc Świętojańską powitamy lato. W tym roku współorganizatorami szkoły są: Sekcja Inżynierii Oprogramowania PTI, Initiative for Software Process Improvement — Régions Extérieures (INSPIRE) oraz Inicjatywa Poprawy Procesów Programowania (IP^3). Te trzy organizacje uznały, że poszukiwanie kwiatu paproci, jakkolwiek pasjonujące, jest mniej efektywne od mozolnej pracy u podstaw i połączyły swe wysiłki dla szerokiego propagowania poprawnych technik i metod organizacji procesu wytwarzania i użytkowania oprogramowania.

Temat właściwych inżynierskich praktyk w konstruowaniu programów jest obecny w Szczyrku od samego początku. W ciągu ostatniej dekady zostało tu wygłoszonych kilkadziesiąt wykładów poświęconych: narzędziom CASE, metodykom organizacji wdrażania systemów informatycznych oraz narzędziom wspomagającym wspomniane procesy. Górska Szkoła zawsze była miejscem spotkania „praktykujących informatyków". Tym bardziej cieszy nas, jako organizatorów, obecność przedstawicieli czołowych polskich wytwórców oprogramowania oraz integratorów systemów informatycznych wśród wykładowców i uczestników tegorocznej konferencji.

Tytuł konferencji wzbudził pewne zastrzeżenia. Pracując nad nim staraliśmy się jak najprecyzyjniej oddać zamysł, który nam przyświecał. Sądzimy, że widzenie po tuwimowsko-mieszczańsku wszystkiego osobno: analizy, konstruowania, a zwłaszcza integracji jest symptomem pewnej niedojrzałości naszej dziedziny techniki. Niewątpliwie projektowanie i konstruowanie oprogramowania stanowi część procesu integracji, jednak jak się wydaje, analiza potrzeb użytkownika winna sięgać znacznie dalej niż tylko do określenia wymagań na oprogramowanie. Znane są przypadki, ze najtańszą i najskuteczniejszą metodą poprawy działania przedsiębiorstwa byłaby reorganizacja i przegrupowanie istniejących zasobów. Szczęśliwy traf zdarzył, że z ponad dwudziestu wykładowców połowa pracuje w firmach parających się na co dzień wytwarzaniem i wdrażaniem oprogramowania, druga połowa natomiast – w ośrodkach naukowych. Sądzimy, że spotkanie tych dwóch środowisk może być pożyteczne dla obu stron. Praca z klientem zajmuje tyle czasu, że nie starcza go już na śledzenie najnowszych naukowych nowinek. Z drugiej strony naukowcy zajęci konstruowaniem nowych teorii i uogólnień, nie mają czasu sprawdzać, jak działają poprzednie w praktyce. Użytkownik na ogół nie ma na nic czasu, zajęty próbami zrozumienia, czemu dostawca sprzętu nie jest w stanie porozumieć się z dostawcą oprogramowania. Mamy nadzieję, że zgodnie z tytułem konferencja pozwoli na, jak to się teraz modnie nazywa „holistyczne" czyli wszechogarniające spojrzenie na problemy, z którymi spotykamy się w swojej pracy.

W nadziei, że czas w Szczyrku spędzony będzie równie przyjemnie, co pożytecznie

Piotr Fuglewicz

SPIS TREŚCI

Str.

1. **Arek Balcerek** (EDS-Poland): „System Live Cycle3, czyli jak EDS wychodzi naprzeciw potrzebom klientów"..... 99

SLC3 jest metodologią rozwoju systemów informatycznych będącą wynikiem doświadczeń specjalistów zatrudnionych w Electronic Data Systems oraz obserwacji trendów na rynku informatycznym. Prezentacja zawiera opis metodologii SLC3 z uwzględnieniem faz na jakie się dzieli, a także korzyści jakie niesie zarówno klientowi jak i samej firmie EDS. Powszechnie stosowanym metodologiom towarzyszą zwykle narzędzia wspomagające proces projektowania. Dlatego część prezentacji poświęcona będzie również narzędziu Proces Sourcerer które służy do rozwijania systemów informatycznych w oparciu o metodologię SLC3.

2. **Tomasz Byzia** (Inter-Design): „Zarządzanie jakością oprogramowania"..... 61

Seminarium prezentuje zagadnienia jakości oprogramowania w duchu zarządzania przez jakość (TQM). Problemy definicji pojęcia jakości, polityki jakości firmy, systemu jakości, itd. przedstawione będą w kontekście całej firmy i każdego procesu produkcyjnego, a nie tylko etapu testowania. Omówione będą podstawowe zasady zarządzania jakością oraz kroki prowadzące do ich pełnego zastosowania w praktyce działalności firmy.

3. **Tomasz Byzia** (Inter-Design): „SPICE — standard oceny i doskonalenia procesów produkcji oprogramowania"..... 151

Seminarium zapoznaje ze współczesnymi trendami doskonalenia procesów produkcji oprogramowania na przykładzie standardu ISO 15504. Zaprezentowana będzie historia standardu, przyjęty model oceny procesów i jego struktura, a także poszczególne kategorie procesów produkcyjnych. Przedstawione będą też wyniki pierwszych praktycznych zastosowań standardu i plany jego rozwoju.

4. **Christophe Debou** (Q-Labs): „The Capability Maturity Model (CMM): Theory and Practice"..... 47

„All you want to know about the CMM" An introduction to the CMM framework for software process improvement will be given: the theoretical background, the components, how to implement it pragmatically, some experiences and return on investment from several industrial sources. Emphasis will be given in breaking the „wrong" belief that the CMM is only applicable for big organisation.

5. **Christophe Debou** (Q-Labs): „A poor man's guide to Software Project leader"..... 9

Presentation introduces the major steps in planning and managing a software project. It basically structure the job, a software project leader has to perform. Steps covered are: defining project goals, Work breakdown structure, cost/effort estimation, scheduling, resources allocation, risk management, budget, project tracking. More emphasise will be given on the aspects of cost estimation and project tracking.

6. **Prof. Janusz Górski** (Politechnika Gdańska): „Inspekcje oprogramowania"..... 21

W ramach cyklu życia, oprogramowanie istnieje w wielu różnych reprezentacjach. Jedną z nich jest wykonywalny kod programu. Inspekcje dokumentacji to jedna z najbardziej skutecznych metod wykrywania defektów w dokumentach. Inspekcje mogą być stosowane do dowolnej dokumentacji. Dzięki temu stanowią jedną z podstawowych technik projakościowych stosowaną we wczesnych etapach rozwoju oprogramowania, wtedy gdy kod programu jeszcze nie istnieje. Są więc istotnym uzupełnieniem testowania, które jest tradycyjnie stosowane w stosunku do kodu programu. Wykład przedstawia proces inspekcji oraz doświadczenia związane z zastosowaniem inspekcji w rzeczywistym projekcie.

7. **Prof. Janusz Górski** (Politechnika Gdańska): „Polityka bezpieczeństwa"..... 5

W wielu zastosowaniach informatyki bezpieczeństwo informacji jest jednym z podstawowych kryteriów rozstrzygających o praktycznej przydatności systemu. Polityka bezpieczeństwa informacji organizacji określa cele bezpieczeństwa oraz zasady, odpowiedzialność i środki służące osiąganiu postawionych celów. W ramach wykładu przedstawiono zakres zagadnień związanych ze zdefiniowaniem i wdrożeniem polityki bezpieczeństwa informacji w instytucji, której funkcjonowanie jest uzależnione od systemu informatycznego.

8. ***Tomasz Gutt** (ICL Polska, Warszawa): „Doświadczenia ICL we wdrażaniu zintegrowanych systemów zarządzania przedsiębiorstwami — metodologia MAXIM"*..... 75

W referacie omówiono specyfikę wdrażania zintegrowanych systemów wspomagających realizację procesów gospodarczych w przedsiębiorstwach produkcyjnych. Natura tych procesów wymaga od oprogramowania dużej elastyczności w kształtowaniu procedur. Identyfikacja procedur przedsiębiorstwa i modelowanie ich w systemie przez odpowiednią parametryzację to pierwszy z czynników decydujących o sprawności wdrożenia. Uzyskanie przez przedsiębiorstwo zdolności do samodzielnego kształtowania procedur systemu zgodnie ze zmieniającymi się pod wpływem rynku procesami gospodarczymi to drugi z czynników, którego rangę podkreśla się w referacie. ICL wykorzystuje wnioski z kilkudziesięciu wdrożeń w Polsce i kilkuset za granicą w swojej metodologii wdrażania systemów klasy MRP-II pod nazwa MAXIM.

9. ***Bogusław Jackowski** (BOP s.c.): „Mocne marki czyli zarządzanie dokumentacją. Opis techniki"*..... 95

Dokumentacja zdaje się być najmniej docenianą częścią produktu programowego — powszechną praktyką jest przygotowywanie dokumentacji za pomocą pierwszego z brzegu edytora tekstów, bez należytej dbałości o formę i — co gorsza — spójność logiczną dokumentacji. Tymczasem technika pasująca jak ulał do problemu zarządzania dokumentacją istnieje od dawna, a zwie się SGML (Standard Generalized Markup Language) i opisana jest szczegółowo w normie ISO 8879 z 1986 roku. W trakcie wykładu przedstawiona zostanie w ogólnym zarysie technika znaczników (markup), przy czym nacisk położony zostanie na jej uniwersalność i prostotę. Dzięki tym właśnie atrybutom znaczniki świetnie nadają się do tworzenia i pielęgnowania dokumentacji. Na zakończenie wykładu pokrótce omówiony zostanie wzięty z życia przykład zastosowania omawianej tu techniki.

10. **Prof. Piotr Jędrzejowicz** (Wyższa Szkoła Morska, Gdynia): „W jakim celu oraz w jaki sposób mierzyć niezawodność oprogramowania?"..... 69

Przedmiotem wykładu jest próba ukazania w jaki sposób oraz w jakim zakresie wyniki pomiarów niezawodności mogą być wykorzystane do lepszego zarządzania produkcją oprogramowania. Oczekiwane korzyści proponowanego podejścia obejmują lepszą identyfikację potrzeb klienta, skrócenie cyklu rozwoju produktu, zwiększenie produktywności oraz lepsze planowanie zużycia zasobów. W trakcie wykładu omówione będą bezpośrednie i pośrednie cele programu pomiarów niezawodności oraz przedstawione zostaną podstawowe czynności wykonywane w ramach takiego programu na poszczególnych etapach cyklu życia oprogramowania. W szczególności, wykład będzie koncentrował się na metodach identyfikacji profilu funkcjonalnego, definiowania i klasyfikowania uszkodzeń, identyfikacji potrzeb użytkownika w zakresie niezawodności, testowaniu wzrostu niezawodności, certyfikacji niezawodności oraz monitorowania niezawodności eksploatacyjnej. W dalszej części wykładu scharakteryzowany zostanie wymagany format danych o uszkodzeniach oprogramowania oraz procedury ich zbierania. Scharakteryzowane zostaną także zasoby niezbędne do wdrożenia programu pomiarów niezawodności oraz zaproponowane ogólne ramy implementacji takiego programu.

11. **Dr Andrzej Kobylinski** (SGH Warszawa): „Miary oprogramowania".....119

Zawartość: Pomiar oprogramowania. Klasyfikacja miar. Miary produktu i procesu programowania. Podstawowe typy miar oprogramowania. Przykłady miar wykorzystywanych w rożnych fazach cyklu życia. Współzależności miedzy miarami.

12. **Jacek Królik** (InfoVide, Warszawa): „Techniki estymacji – metoda punktów funkcyjnych a szacowania eksperckie".....85

Odwieczną bolączką projektów informatycznych jest konieczność określania kluczowych parametrów projektu — budżetu, pracochłonności i czasu realizacji, bez głębokiej znajomości zakresu prac. Brak powszechnie stosowanych zasad rzetelnego i wiarygodnego kosztorysowania projektów to jeden z czynników stawiających pod znakiem zapytania „inżynierskość" inżynierii oprogramowania. W wykładzie omówione zostaną dające się w praktyce zastosować formalne techniki estymacji. Podstawowe formalne techniki szacowania przydatne w praktyce to formuły oparte na pomiarze złożoności projektu na podstawie jego specyfikacji. Wszelkie takie techniki zawierają jednak istotne elementy oceny subiektywnej. Omówimy zatem również zasady prowadzenia szacowań subiektywnych (opartych o wiedzę ekspercką i zdefiniowaną procedurę oceny) umożliwiające osiąganie za ich pomocą względnie dokładnych oszacowań. Poruszymy tutaj tematy takie jak formalna ocena ryzyka i estymacje oparte o formułę delficką (W.IVE).

13. **Dr Marian Kuraś, Agnieszka Zając** (Akademia Ekonomiczna, Kraków): „Powodzenie i zagrożenia zamierzeń informatycznych".....39

W referacie omówiono zagrożenia i ryzyko związane z wprowadzaniem i wykorzystywaniem techniki informacyjnej do wspomagania zarządzania. Jego celem jest wskazanie zagrożeń oraz możliwości zapobiegania, eliminowania i zmniejszania ryzyka. Rozważania oparto na najnowszej literaturze oraz własnych badaniach. Omówiono sposób rozumienia powodzenia zamierzenia, systematykę zagrożeń i ryzyka zamierzeń oraz przedyskutowano potrzebę sterowania ryzykiem zamierzeń. Przedstawiono w skrócie propozycję metodyki analizy i sterowania ryzykiem zamierzeń.

14. **Dr Marian Kuraś** (AE Kraków), **prof. Zbigniew Nęcki** (Uniwersytet Jagielloński) „Psychologiczne aspekty komunikacji z użytkownikiem".....89

W referacie omówiono podstawy komunikacji między informatykami a użytkownikami. Przedstawiono typowe zakłócenia i ich przyczyny natury psychologicznej. Celem referatu jest uświadomienie i umożliwienie lepszego zrozumienia niektórych błędów popełnianych w kontaktach z użytkownikami a także w środowisku zawodowym. Do przygotowania referatu wykorzystano bogate własne obserwacje oraz literaturę przedmiotu. Przedstawiono proste rady dla informatyków i użytkowników, jednak nacisk położono na potrzebę rozwijania umiejętności sprawnego, skutecznego i efektywnego porozumiewania się.

15. **Dr Marek Maniecki** (Inter-Design): „Zapewnienie jakości w wytwarzaniu oprogramowania — wyniki badań przeprowadzonych w Polsce i w krajach Europy Zachodniej".....1

Celem seminarium jest przedstawienie wyników ankiety przeprowadzonej w ramach projektu INSPIRE wśród polskich firm na temat stosowania metod zapewnienia jakości w produkcji oprogramowania. Wyniki te zostaną porównane z rezultatami analogicznego badania przeprowadzonego w 17 krajach Europy Zachodniej (projekt ESPITI).

16. **Dr Marek Miłosz** (Politechnika Lubelska): „Estymacja zasobów w projektach informatycznych" 79

Wykład prezentuje problemy szacowania zapotrzebowania na zasoby w trakcie realizacji systemów informatycznych. Po klasyfikacji technik estymacji parametrów projektu w różnych jego fazach omówione zostaną podstawowe techniki estymacji pracochłonności projektów informatycznych (metoda punktów funkcyjnych i COCOMO). Przedstawione zostaną także problemy dostosowywania modeli do konkretnych warunków realizacji projektów informatycznych i formowania zespołów ludzkich. Poza estymacją zasobów potrzebnych do wytworzenia metryki oprogramowania mogą spełniać różne funkcje. W wykładzie będzie zwrócona uwaga na te niekonwencjonalne zastosowania.

17. **Dr Jerzy Nawrocki (Politechnika Poznańska):** „PSP — osobisty proces tworzenia oprogramowania" 111

Na jakość organizacji zajmujących się wytwarzaniem oprogramowania decydujący wpływ ma jakość pracujących tam ludzi. Osobisty proces tworzenia oprogramowania (PSP od ang. Personal Software Process) jest to zestaw metod i narzędzi mających wspomóc indywidualnych wytwórców oprogramowania (często będących członkami wieloosobowych zespołów) w doskonaleniu ich procesu tworzenia oprogramowania. PSP wymaga zbierania danych na temat czasu spędzanego na realizacji poszczególnych faz cyklu życia oprogramowania, liczby błędów popełnianych i wykrywanych w poszczególnych fazach itp. W zamian PSP daje możliwość wnioskowania statystycznego o przyszłych projektach (ich czasie, kosztach itp.). W trakcie wystąpienia przedstawiona zostanie idea PSP, zaprezentowany zostanie tzw. podstawowy proces osobisty oraz poruszone zostaną zagadnienia związane z planowaniem.

18. **Grzegorz Prochowski** (Pro-Cons, Warszawa): „Zarządzanie projektami metodą «Earned Value»" 139

Zarządzanie oparte na Earned Value ("wartości zarobionej") jest prostą koncepcyjnie metodą pozwalającą w jednym mierzalnym elemencie skupić główne problemy związane z kosztami, harmonogramem i wydajnością w projekcie. Jednocześnie metoda ta ułatwia stworzenie bazy dla efektywnej komunikacji między zarządem projektu a zespołem projektowym, podwykonawcami i wysokim kierownictwem (sponsorem projektu). Metoda Earned Value zyskuje ostatnio bardzo dużą popularność w świecie. Między innymi jest zalecaną metodą zarządzania wszelkimi projektami (nie tylko informatycznymi) wykonywanymi na zlecenie rządu Stanów Zjednoczonych (standard ANSI). Jest stosowana w projektach podejmowanych w ramach NATO. Referat omawia podstawowe pojęcia metody EV oraz jej zastosowania do zarządzania projektem — między innymi: śledzenie wydajności w projekcie, prognozowanie przyszłych kosztów, prognozy terminu zakończenia.

19. **Prof. Eberhard E. Rudolph** (FB2 Informatik, Hochschule Bremerhaven): „Measuring the software development process and it's result"

This presentation discusses the state of art of commercially used software metrics. Many thousand organisations world-wide have metric programs and use Function Points as an indicator for software size. Increasingly management decisions, cost assessments and legal contracts are based on results derived from this software metrics. With functional size as normalisation factor, indicators such as productivity or defect rates are derived, providing data for a quantitative assessment of software and it is development processes. In general, however, software development is not subjected to any measurement. Most managers base software development planning on intuitive guesswork. Measurement is often regarded as a threat. The academic and scientific world, furthermore, casts great doubts on the validity of such metrics. Many publications point out that, according to measurement theory, the

metrics in place, and particularly Function Points, have no scientific justification and violate fundamental theoretical principals. Naturally software developers, managers, and investors are confused. Can measurement programs be justified? Can the results be trusted for decision making? Worse, isn't it a good excuse to refrain from measurement, since, after all, the results may be of little value? This presentation provides a framework of the key data a software developer and user should collect for management and audit purposes. Some suggestions are made to improve the verification process of software measures and to guard against misinterpretation of results.

20. **Dariusz Samól, Piotr Jankiewicz** (SAP Polska, Warszawa): „Integracja łańcucha dostaw poprzez moduły oprogramowania R3 v.4.0" 31

Tezy: Rozszerzenie zakresu działalności przedsiębiorstwa na obszar dostawców i klientów; Widoczność procesu logistycznego: — aktywne wykorzystanie procedury wdrożeniowej SAP, etapy i sposób realizacji, — zastosowanie części narzędziowej systemu R3 do fazy koncepcyjnej i wdrożenia; Integracja prognozowania i planowania; Wspomaganie podejmowania decyzji w czasie rzeczywistym.

21. **Dr inż. Bożena Skibicka** (MIS, Wrocław): „Procesowe zarządzanie firmą w technologii Lotus Notes" 131

Artykul omawia wykorzystanie pakietu oprogramowania wykonanego w technologii Lotus Notes do opisu i śledzenia procesów, zgodnie z którymi działa przedsiębiorstwo. Przedstawia przykładowe opisy. Wskazuje na szerokie możliwości zastosowania, np. do opisu procedur zgodnych ze standardem ISO 9000.

22. **Piotr Sobolewski** (Digital Equipment Polska, Warszawa): „Architektura rozwiązania — z perspektywy Integracji Systemów" 25

Przedmiotem prezentacji jest określenie roli architektury rozwiązania w projektach integracyjnych. Architekt rozwiązania jest odpowiedzialny za przewodzenie podczas definiowania i tworzenia skomplikowanego systemu informacyjnego, a w szczególności za dopasowanie odpowiednich technologii do potrzeb klienta w celu stworzenia całościowego projektu systemu, utrzymywanie koncepcyjnej spójności architektury w trakcie budowy sytemu oraz zweryfikowanie czy zbudowany system spełnia kryteria zdefiniowane w architekturze.

23. **Borys Stokalski** (InfoVide, Warszawa): „Joint Application Development — techniki efektywnej współpracy z użytkownikiem" 157

Dla większości organizacji tworzących systemy informatyczne dla użytkowników sprawą oczywistą jest potrzeba panowania nad zakresem i jednoznaczna definicja potrzeb użytkowników. Stosowane dziś techniki definiowania wymagań — od formalnych modeli, do prototypowania — tworzą bogaty „język" porozumiewania się twórców systemu z jego przyszłymi beneficjantami. Samo bogactwo środków wyrazu nie gwarantuje jednak efektywnej komunikacji. W trakcie wykładu przedstawiona zostanie koncepcja Współprojektowania (Joint Application Development, JAD). Współprojektowanie to metoda grupowego tworzenia produktów projektowych z intensywnym udziałem użytkowników. Omówione zostaną zasady praktycznego wykorzystania tej koncepcji w przedsięwzięciach informatycznych.

24. **Janusz Wielki** (Akademia Ekonomiczna we Wrocławiu): „Podstawowe elementy krytyki reengineeringu i analiza ich zasadności" 103

Referat jest próbą oceny przydatności powstałej na początku lat dziewięćdziesiątych koncepcji wprowadzania głębokich zmian w funkcjonowaniu organizacji, zwanej reengineeringiem. Przedstawione w nim zostaną najważniejsze zarzuty pod adresem BPR zebrane na podstawie dostępnej literatury jak również dokonana zostanie ocena ich zasadności. Zaprezentowane będą też wyniki badań dotyczących oceny skuteczności tej koncepcji oraz stosunku menedżerów do niej. W końcowej części przedstawione zostaną pewne sugestie i wnioski.

Dr Tadeusz Wilusz (Akademia EK): „Edukacja informatyczna jako ważny kontekst dyskusji o jakości oprogramowania" 125

Wychodząc z jawnie oczywistej zasady, że o jakości realizacji jakiegokolwiek przedsięwzięcia w głównej mierze decydują umiejętności i kwalifikacje jego realizatorów, trudno, w kontekście dyskusji o problemach wytwarzania wysokiej jakości oprogramowania, pominąć pytanie o aktualny kształt edukacji informatycznej w Polsce. Właśnie szeroko rozumianej edukacji a nie wyłącznie specjalistycznego kształcenia kadr informatyki. W intencji autora referat ma stanowić bilans otwarcia do jak najszerszej dyskusji nad zagadnieniem, którego strategiczną rangę dla zasadniczego tematu konferencji trudno jest przecenić.

OŚRODKI

1. AKADEMIA EKONOMICZNA W KRAKOWIE
2. AKADEMIA EKONOMICZNA WE WROCŁAWIU
3. FB2 INFORMATIK, HOCHSCHULE BREMERHAVEN, NIEMCY
4. POLITECHNIKA GDAŃSKA
5. POLITECHNIKA LUBELSKA
6. POLITECHNIKA POZNAŃSKA
7. SZKOŁA GŁÓWNA HANDLOWA
8. UNIWERSYTET JAGIELLOŃSKI
9. WYŻSZA SZKOŁA MORSKA, GDYNIA
10. BOP S.C., GDAŃSK
11. DIGITAL EQUIPMENT POLSKA, WARSZAWA
12. EDS, WARSZAWA
13. ICL WARSZAWA
14. INFOVIDE, WARSZAWA
15. INTER-DESIGN, WARSZAWA
16. MIS, WROCŁAW
17. PRO-CONS, WARSZAWA
18. Q-LABS, KAISERSLAUTER
19. SAP, WARSZAWA

Marek Maniecki

Zapewnienie jakości w wytwarzaniu oprogramowania

porównanie wyników badań przeprowadzonych w Polsce
i w krajach Europy Zachodniej

INSPIRE

Zapewnienie jakości w wytwarzaniu oprogramowania

porównanie wyników badań przeprowadzonych w Polsce i w krajach Europy Zachodniej

dr inż. Marek Maniecki
koordynator projektu INSPIRE
PZ Inter-Design Tessel Systems

INSPIRE Szczyrk, 22 06 1998 Slide 1 INTER-DESIGN TESSEL SYSTEMS

Projekt INSPIRE

- Celem projektu jest udostępnienie małym i średnim firmom w Europie Środkowej i Wschodniej wiedzy, metod i narzędzi zapewnienia jakości w wytwarzaniu oprogramowania, stosowanych w Europie Zachodniej.
- **INSPIRE** jest realizowany w ramach Europejskiej Inicjatywy d/s Systemów i Oprogramowania przy finansowym wsparciu Komisji Europejskiej poprzez projekt Copernicus.
- Projekt jest realizowany w Polsce oraz na Węgrzech, w Rumunii i w Estonii z udziałem firm brytyjskiej i niemieckiej.

INSPIRE Szczyrk, 22 06 1998 Slide 2 INTER-DESIGN TESSEL SYSTEMS

ESSI

- ESSI została powołana 1993 roku dla:
  - koordynacji działań w zakresie poprawy procesu wytwarzania oprogramowania w państwach Unii Europejskiej,
  - zapewnienia skutecznych metod rozpowszechniania wyników prowadzonych prac z tej dziedziny a sponsorowanych przez KE.
- Tworzy grupy robocze - INSPIRE razem z 5 innymi projektami należy do ESSI Training Cluster.
- Współdziała z Europejskim Instytutem Oprogramowania w Bilbao w Hiszpanii.

INSPIRE Szczyrk, 22 06 1998 Slide 3 INTER-DESIGN TESSEL SYSTEMS

Przedsięwzięcia szkoleniowe

- Seminaria uświadamiające - zwiększenie świadomości potrzeby doskonalenia procesu wytwarzania oprogramowania
- Sesje szkoleniowe - przedstawianie konkretnych metod i narzędzi
- Sesje treningowe - dla specjalistów, wymiana doświadczeń
- Spotkania grupy roboczej - Inicjatywa Poprawy Procesów Programowych **IP3**, zawiązana w lipcu 1997 roku.

INSPIRE Szczyrk, 22 06 1998 Slide 4 INTER-DESIGN TESSEL SYSTEMS

Finansowane przez projekt doradztwo

- Regionalny zespół doradców:
  - zgłoszenia od 8 osób,
  - 7 osób spełniających kryteria.
- Pomoc w ocenie stanu firmy pod względem jakości tworzenia oprogramowania
- Pomoc we wprowadzaniu zmian w firmie, zastosowaniu narzędzi
- Doradcy opłacani przez projekt dla 4 polskich firm (po 10 dni roboczych)

INSPIRE Szczyrk, 22 06 1998 Slide 5 INTER-DESIGN TESSEL SYSTEMS

Finansowane przez projekt eksperymenty w firmach

- Cel - zrealizowanie opracowanego przez firmę eksperymentu poprawy procesu wytwarzania oprogramowania z funduszy projektu
- Pomoc dla 3 polskich firm (zgłoszenia od 5 firm)
- Rozpowszechnienie wyników - poszukiwanie rozwiązań wspólnych problemów
- Inicjowanie współpracy między firmami dla wymiany wiedzy i doświadczeń

INSPIRE Szczyrk, 22 06 1998 Slide 6 INTER-DESIGN TESSEL SYSTEMS

Badanie ankietowe

- Badanie dotyczące stosowania metod zapewnienia jakości w wytwarzaniu oprogramowania
- Ankieta zgodna z badaniem przeprowadzonym w 17 krajach Europy Zachodniej w ramach programu ESPITI w 1996 roku:
  - rozesłano ponad 70 000 kwestionariuszy,
  - otrzymano 3 800 odpowiedzi (5.4%):
    - najwięcej z Niemiec (ok. 18%),
    - najmniej z Islandii i Grecji (poniżej 3%).

INSPIRE Szczyrk 22.06.1998 Slide 7 INTER-DESIGN TESSEL SYSTEMS

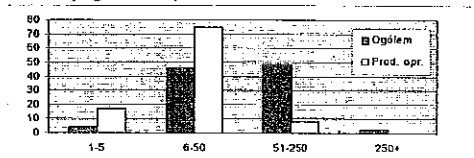
Badanie ankietowe w Polsce

- Badanie w Polsce:
  - polska wersja kwestionariusza,
  - rozesłano do 410 przedsiębiorstw:
    - członków Polskiej Izby Informatyki i Telekomunikacji,
    - członków Stowarzyszenia Polski Rynek Oprogramowania PRO,
    - wybranych klientów Inter-Design,
  - otrzymano wypełniony formularz od 45 firm,
  - współczynnik zwrotu - 11%,
  - odpowiedzi przygotowane głównie przez dyrektora firmy lub kierownika zespołu programistów.

INSPIRE Szczyrk 22.06.1998 Slide 8 INTER-DESIGN TESSEL SYSTEMS

Firmy ankietowane (1)

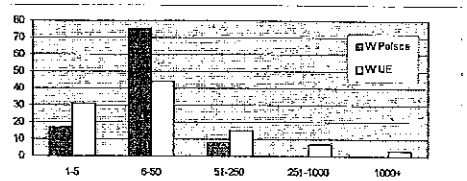
- **Profil firmy:**
  - sektor rynku:
    - produkcja oprogramowania i usługi - 80% (45%),
    - inne (głównie energetyka) - 20% (55%),
  - liczba zatrudnionych (ogółem i w wytwarzaniu oprogramowania):



INSPIRE Szczyrk 22.06.1998 Slide 9 INTER-DESIGN TESSEL SYSTEMS

Firmy ankietowane (2)

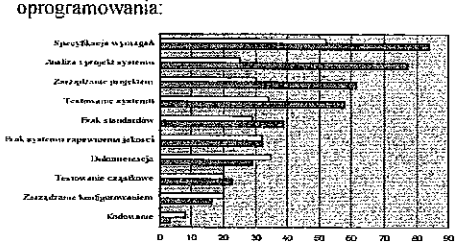
- Porównanie wielkości firm:
  - w sektorze produkcji oprogramowania i związanych z tym usług,
  - według liczby zatrudnionych przy wytwarzaniu oprogramowania:



INSPIRE Szczyrk 22.06.1998 Slide 10 INTER-DESIGN TESSEL SYSTEMS

Problemy

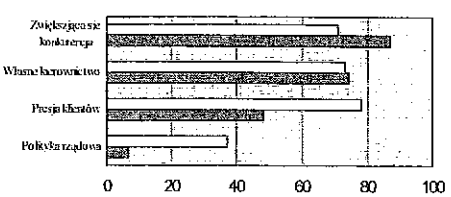
- Główne problemy występujące przy wytwarzaniu oprogramowania:



INSPIRE Szczyrk 22.06.1998 Slide 11 INTER-DESIGN TESSEL SYSTEMS

Certyfikacja (1)

- Czynniki wpływające na chęć zdobywania certyfikatu ISO 9000:



INSPIRE Szczyrk 22.06.1998 Slide 12 INTER-DESIGN TESSEL SYSTEMS

Certyfikacja (2)

- Zainteresowanie kierownictwa i personelu firm problemami zapewnienia jakości i certyfikacją:

	SPI	ISO900	Szkolenie
Kierownictwo handlowe	+ -	+ -	+
Kierownictwo techniczne	+	+	+
Personel techniczny	+ -	-	+

INSPIRE Szczyrk, 22.06.1998 Slide 13 INTER-DESIGN TESSEL SYSTEMS

Sytuacja w firmie

- Obecna sytuacja i plany firmy w zakresie metod zapewniania jakości i certyfikacji:

INSPIRE Szczyrk, 22.06.1998 Slide 14 INTER-DESIGN TESSEL SYSTEMS

Zapotrzebowanie na szkolenia

- Typy szkoleń:
  - głównie szkolenia uświadamiające (podobnie w UE).
- Preferowana długość szkoleń:
  - od 2 do 3 dni (podobnie w UE).
- Średni budżet na szkolenia na jednego pracownika zaangażowanego w wytwarzanie oprogramowania:
  - w Polsce - 5 dni w ciągu roku,
  - w UE - 8 dni w ciągu roku.

INSPIRE Szczyrk, 22.06.1998 Slide 15 INTER-DESIGN TESSEL SYSTEMS

Uwagi końcowe

- Wyniki ankiety nie wykazują dużych różnic w świadomości problemu poprawy jakości wytwarzania oprogramowania pomiędzy firmami polskimi i zachodnimi.
- W polskich firmach:
  - o wiele mniejsze znaczenie przywiązuje się do problemu zdobywania certyfikatów,
  - wskazywana bardzo duża rola szkoleń uświadamiających sygnalizuje mniejszą wiedzę na te tematy, zwłaszcza w kierownictwach firm,
  - firmy deklarują chęć poznawania i zastosowania metod i narzędzi z zakresu poprawy procesów programowych.

INSPIRE Szczyrk, 22.06.1998 Slide 16 INTER-DESIGN TESSEL SYSTEMS

Janusz Górski

Politechnika Gdańska, Katedra Zastosowań Informatyki,
ul. Narutowicza 11/12, 80-952 Gdańsk
e-mail: jango@pg.gda.pl

Polityka bezpieczeństwa informacji

W wielu zastosowaniach informatyki bezpieczeństwo informacji jest jednym z podstawowych kryteriów rozstrzygających o praktycznej przydatności systemu. Polityka bezpieczeństwa informacji organizacji określa cele bezpieczeństwa oraz zasady, odpowiedzialność i środki służące osiąganiu postawionych celów. W ramach artykułu przedstawiono zakres zagadnień związanych ze zdefiniowaniem i wdrożeniem polityki bezpieczeństwa informacji w instytucji, której funkcjonowanie jest uzależnione od systemu informatycznego.

1. Pojęcie bezpieczeństwa informacji

Pod pojęciem systemu komputerowego (lub krócej, systemu) będziemy rozumieli strukturę, na którą składają się węzły generujące, składujące i/lub przetwarzające informację oraz środki komunikacyjne służące przesyłaniu informacji. Informacja stanowi podstawowy zasób takiego systemu. Będziemy posługiwali się pojęciem *podmiotu* dla określenia jakiegokolwiek obiektu (użytkownik, komputer, program aplikacyjny, itp.), który może uzyskać dostęp do informacji przechowywanej, przetwarzanej lub przesyłanej w systemie. Dostęp jest rozumiany szeroko, może on polegać na bezpośrednim odczytaniu, wywnioskowaniu (z innych, posiadanych już informacji), modyfikacji lub zniszczeniu rozważanej informacji. *Podmiot upoważniony* to taki, który ma prawo do dostępu. Zakładamy, że istnieje zbiór kryteriów pozwalających na wyróżnienie podmiotów upoważnionych. Kryteria te reprezentują punkt widzenia właściciela systemu.

Tradycyjnie, *bezpieczeństwo informacji* jest rozumiane w terminach następujących atrybutów:

- *Integralność (*ang. *integrity)* - informacja w systemie nie może zostać zmieniona lub zniszczona przez podmioty nieupoważnione.
- *Dostępność (*ang. *availability)* - informacja jest dostępna (w określonym miejscu, czasie i postaci) podmiotom upoważnionym.
- *Poufność (*ang. *security)* - informacja nie jest udostępniana podmiotom nieupoważnionym.

Nie wszystkie informacje w systemie podlegają ochronie w tym samym stopniu. Tak więc, celowym jest dokonanie klasyfikacji zasobów informacyjnych, a następnie określenie wymaganego poziomu ich zabezpieczenia. Klasyfikacji takiej dokonuje się w odniesieniu do wartości jaką ma dany zasób informacyjny dla wykorzystującej go instytucji. Punktem wyjścia przy rozważaniach dotyczących stopnia zabezpieczenia zasobów jest pojęcie *ochrony niezbędnej*. Ochrona niezbędna wynika z obowiązujących ogólnych aktów prawnych (np. z ustawy o ochronie dóbr osobistych obywateli) lub umów szczegółowych, zawartych np. pomiędzy daną instytucją i jej klientami. Przykładem może tu być np. informacja o adresach, informacja o płaconych rachunkach, informacja o dochodach, itp. Zakres ochrony wykraczający poza ochronę niezbędną wynika z oceny znaczenia rozważanego zasobu informacyjnego dla danej instytucji. Często stosuje się tu podział na kategorie, w ramach których określane są szkody związane z utratą bezpieczeństwa.

2. Zakres polityki bezpieczeństwa

Z każdym systemem (w tym również informatycznym) związany jest jego *cykl życia*, który w najogólniejszym ujęciu obejmuje okres czasu od powzięcia zamiaru wytworzenia tego systemu, poprzez wszystkie etapy związane z konkretyzacją tej idei, projektowaniem, realizacją, instalacją w środowisku docelowym, eksploatacją, dokonywaniem zmian i udoskonaleń, a skończywszy na wycofaniu systemu z użytkowania i jego całkowitej dematerializacji. Bezpieczeństwo jest atrybutem całego systemu i dlatego musi być brane pod uwagę w ramach wszystkich etapów cyklu życia i w odniesieniu do wszystkich jego części składowych (a nie wyłącznie w odniesieniu do segmentu technicznego). Oznacza to w szczególności, że skuteczne zabezpieczenie wymaga podejścia systemowego, obejmującego oprócz części technicznej, również zagadnienia organizacji, zarządzania, personelu i infrastruktury oraz że oprócz zagadnień związanych z projektowaniem, implementacją i użytkowaniem należy również brać pod uwagę problemy pielęgnacji, współpracy z kontrahentami, instalacji, likwidacji czy też zgodności z obowiązującym prawem.

Nie ma systemów absolutnie bezpiecznych. Również zabezpieczanie wszystkiego i wszędzie nie jest działaniem sensownym i może w efekcie doprowadzić do nadmiernych kosztów i sparaliżować normalne funkcjonowanie danej organizacji. Bezpieczeństwo jest w znacznym stopniu uzależnione od ludzi, ich świadomości i woli współdziałania. Ludzie ci funkcjonują w ramach struktur organizacyjnych i realizują zadania wynikające z przydzielonych im funkcji i związanej z tym odpowiedzialności. Dlatego niezbędne jest jawne sformułowanie *polityki bezpieczeństwa* danej organizacji, która określa sposób rozumienia bezpieczeństwa, stanowi podstawę do dalszych analiz w kierunku konkretnych rozwiązań technicznych, ustala odpowiedzialność i jasno wyraża intencje władz organizacji odnośnie wspierania wszelkich działań zmierzających do realizacji tej polityki.

Racjonalny wybór zabezpieczeń musi być oparty o analizy, których wyniki stanowią podstawę do wyznaczenia sensownego poziomu nakładów oraz wskazują kierunki, gdzie nakłady te będą wykorzystane w sposób najbardziej efektywny. Bez przeprowadzenia takich analiz nie jest możliwy uzasadniony wybór środków bezpieczeństwa i konkretnych rozwiązań technicznych. Dla przykładu, inwestycja w szyfrowanie sygnałów przesyłanych liniami komunikacyjnymi może okazać się zupełnie chybiona jeżeli aplikacje końcowych użytkowników, korzystające z deszyfrowanej już informacji, będą narażone na łatwy do przeprowadzenia atak prowadzący do utraty poufności. Działania związane z ustaleniem zagrożeń istniejących w kontekście danej aplikacji i ich potencjalnych skutków są objęte *analizą bezpieczeństwa.*

Zestaw zabezpieczeń wynikających z przeprowadzonych analiz bezpieczeństwa znajduje swój materialny wyraz w postaci *systemu ochrony*, który obejmuje całość środków, struktur organizacyjnych i procedur postępowania służących zapewnieniu bezpieczeństwa informacji w danej organizacji.

W nawet w najlepiej zabezpieczonym systemie trzeba liczyć się z sytuacją, w której dojdzie do utraty bezpieczeństwa chronionych zasobów. Wtedy, wielkość poniesionych strat będzie zależeć od zdolności danej organizacji do zidentyfikowania i ograniczenia skutków takiego incydentu, podjęcia działań zmierzających do przywrócenia poprzedniego poziomu bezpieczeństwa oraz kontynuacji normalnej działalności. Sprawność tych działań zależy od tego czy organizacja ta jest do takiej sytuacji przygotowana. Umożliwia to *plan zapewnienia ciągłości funkcjonowania,* który zawiera scenariusze działań związanych z sytuacjami awaryjnymi. Plan taki jest więc istotnym elementem kompleksowego systemu bezpieczeństwa.

W bezpieczeństwie systemu znaczącą rolę odgrywa czynnik czasu. Nawet bardzo dobrze zabezpieczony system (względem określonej grupy zagrożeń) może utracić bezpieczeństwo na skutek pojawienia się nowych zagrożeń, które nie były poprzednio wzięte pod uwagę (np. nowe postaci wirusów) lub utraty skuteczności istniejących zabezpieczeń (np. poprzez ujawnienie ich sekretów). Może również zaistnieć sytuacja gdzie (dobrze dobrany) zbiór zabezpieczeń staje się nieefektywny ponieważ nie są one w praktyce stosowane. Niezbędne jest więc ciągłe zbieranie danych nt. nowych zagrożeń, inwentaryzacja prób

ataków na system i innych zdarzeń związanych z bezpieczeństwem oraz ponawianie analizy bezpieczeństwa i weryfikacja skuteczności istniejących zabezpieczeń. Oznacza to, że niezbędne jest *zarządzanie bezpieczeństwem* systemu mające na celu ciągłe utrzymywanie (odnawianie) pożądanego poziomu bezpieczeństwa. Istotnym elementem jest tu *audyt* czyli działania kontrolne oceniające stan zabezpieczeń w systemie.

Powyższe rozważania wskazują na szeroki zakres pojęcia bezpieczeństwa systemu, daleko wykraczający poza zagadnienia czysto techniczne. Wynika również z nich, że w wielu wypadkach, przyjęcie konkretnych rozwiązań technicznych powinno być poprzedzone analizami przeprowadzonymi w odniesieniu do całego systemu i związanej z nim aplikacji.

3. Cykl życia bezpieczeństwa systemu

Cykl życia bezpieczeństwa systemu obejmuje całość działań dotyczących bezpieczeństwa i związanych z rozpatrywanym systemem. W szczególności, w jego zakres wchodzą następujące obszary:

- *Wytwarzanie systemu.*
  - Polityka bezpieczeństwa systemu, która jest wykorzystana jako punkt odniesienia przy ocenie bezpieczeństwa.
  - Specyfikacja funkcjonalna, która precyzuje wymagania odnośnie bezpieczeństwa systemu.
  - Analiza funkcjonalna systemu wykazująca w jaki sposób funkcje wymuszające bezpieczeństwo są powiązane z zagrożeniami.
  - Analiza "mocy" mechanizmów bezpieczeństwa, poprzez ocenę tworzących je algorytmów, przyjętych założeń oraz wynikających stąd własności.
  - Identyfikacja poziomu bezpieczeństwa komponentów systemu. W obecnym stanie technologii, system jest często "składany" z istniejących elementów, których własności z punktu widzenia bezpieczeństwa mogą się różnić w znaczący sposób.
  - Analiza integracyjna, która określa zdolność do harmonijnego współdziałania funkcji i mechanizmów bezpieczeństwa obecnych w systemie.
  - Analiza i ocena słabych stron konstrukcji systemu, polegająca na ich identyfikacji, ocenie wynikających stąd skutków oraz propozycji środków zaradczych.
  - Wybór i ocena architektury systemu ochrony określającej dyslokację i sposób dostępu do funkcji i mechanizmów wymuszających bezpieczeństwo.
  - Szczegółowy projekt funkcji i mechanizmów bezpieczeństwa.
  - Weryfikacja, walidacja i testowanie bezpieczeństwa systemu.
- *Środowisko wytwórcze:*
  - Zarządzanie konfiguracją zarówno w aspekcie wytwarzanego systemu jak i w aspekcie środowiska wytwórczego (w szczególności chodzi tu o ochronę i kontrolę dostępu do elementów konfiguracji oraz tworzonych wersji systemu).
  - Bezpieczeństwo użytych narzędzi (np. języki programowania i ich kompilatory).
  - Bezpieczeństwo związane z personelem zaangażowanym w proces wytwórczy.
  - Bezpieczeństwo procesu produkcji systemu.
- *Użytkowanie systemu:*
  - Łatwość użytkowania. Dotyczy to nie tylko sytuacji "normalnych" ale również sytuacji związanych z błędem systemu oraz procedurami powrotu do stanu normalnego i wynikającymi stąd skutkami dla bezpieczeństwa.
  - Niestabilność warunków operacyjnych systemu i wynikające stąd skutki dla bezpieczeństwa oraz możliwe środki zaradcze.
  - Dokumentacja użytkowa w zakresie bezpieczeństwa systemu.
  - Bezpieczne dostarczenie systemu w miejsce przeznaczenia oraz bezpieczeństwo instalacji systemu.
  - Bezpieczeństwo inicjalizacji i konfigurowania systemu.
- *Pielęgnacja:*
  - Stosowane strategie pielęgnacji systemu.
  - Ilość i bezpieczeństwo personelu mającego dostęp do systemu w trakcie jego konserwacji.

- Efekty wynikające z aktualizacji jednego podsystemu i ich wpływ na inne podsystemy z nim współpracujące.
- Bezpieczeństwo testowania po zabiegach konserwatorskich.
- Usuniecie "kanałów dostępu" związanych z testowaniem systemu.

4. Proces zapewniania bezpieczeństwa informacji

Zapewnienie bezpieczeństwa nie jest aktem jednorazowym. Jest to proces, który powinien mieć jasno zdefiniowany zakres i cele, przypisaną odpowiedzialność oraz przydzielane zasoby. Ponieważ proces ten przebiega w zmiennych warunkach, musi on podlegać zarządzaniu.

Uruchomienie procesu zapewniania bezpieczeństwa w organizacji wymaga podjęcia następujących akcji:

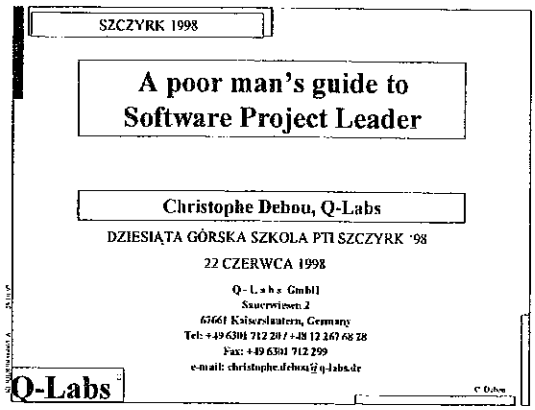
- *Definicja celów bezpieczeństwa.* Cele lub zadania organizacji jak również istniejące uregulowania prawne nakładają określone wymagania dotyczące wykorzystania istniejących w niej systemów informatycznych. Wymagania te są podstawą do sformułowania celów bezpieczeństwa. Tak więc, cele bezpieczeństwa stanowią wymagania, których wypełnienie gwarantuje właściwe i bezpieczne wykorzystanie systemów informatycznych na rzecz osiągania celów i wypełniania zadań stawianych przed całą organizacją.

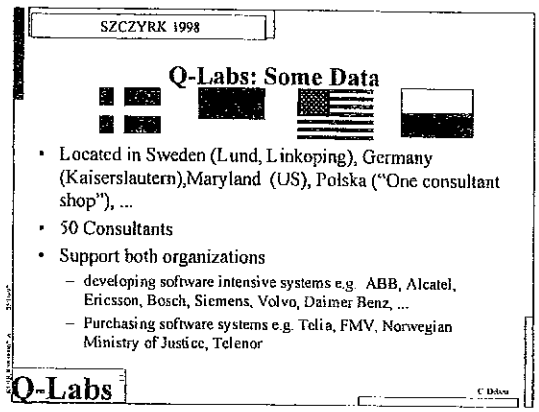
- *Uruchomienie funkcji zarządzania bezpieczeństwem.* Zarządzanie bezpieczeństwem zmierza do osiągnięcia i utrzymywania w czasie poziomu bezpieczeństwa adekwatnego do jawnie sformułowanych celów bezpieczeństwa. Wymaga to określonych działań organizacyjnych, dekompozycji i przydziału zadań i odpowiedzialności oraz koordynacji i nadzoru nad bezpieczeństwem.
- *Opracowanie polityki bezpieczeństwa informacji.* Polityka bezpieczeństwa określa sposoby osiągania bezpieczeństwa, na podstawie analizy zagrożeń i związanego z nimi ryzyka oraz poprzez dobór odpowiedniego zestawu środków ochronnych.
- *Implementacja systemu ochrony wynikającego z przyjętej polityki bezpieczeństwa.* Poprzez ustalanie priorytetów, planowanie zadań i przydzielanie odpowiedzialności związanej z ich realizacją, zestaw zabezpieczeń objęty systemem ochrony podlega wdrożeniu w ramach istniejących ograniczeń zasobów.
- *Zdefiniowanie programu szkoleń.* Program szkoleń musi uwzględniać potrzeby podniesienia kompetencji personelu czynnie zaangażowanego w procesie zapewniania bezpieczeństwa oraz ogólną potrzebę rozbudzenia świadomości i zrozumienia wśród załogi w zakresie przyjętych środków ochrony.
- *Nadzór nad bezpieczeństwem.* Proces zapewniania bezpieczeństwa nie kończy się z momentem wprowadzenia w życie systemu ochrony. Bezpieczeństwo musi podlegać nadzorowi i okresowym sprawdzeniom. W sytuacjach pojawienia się nowych zasobów, zagrożeń, wykrycia luk w systemie ochrony, itp., działania objęte powyższymi punktami są muszą być ponowione.

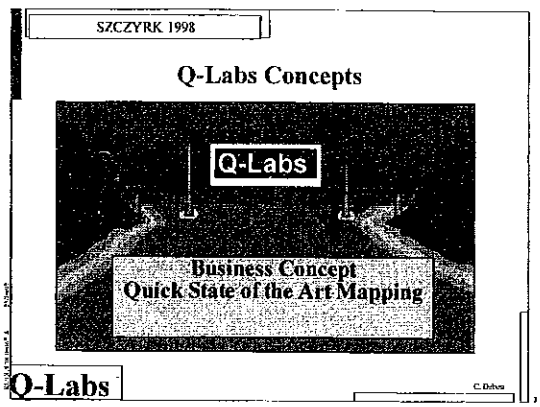
Christophe Debou

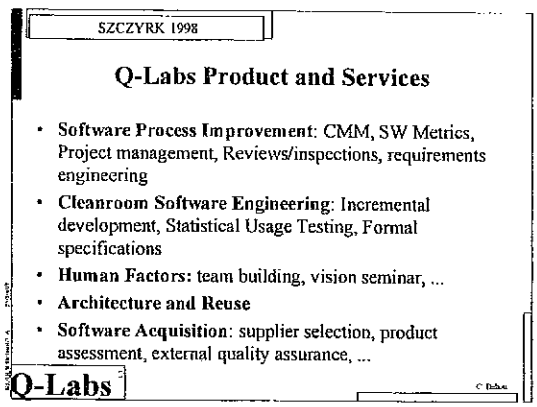
Q - L a b s GmbH
Sauerwiesen 2
67661 Kaiserslautern, Germany
Tel: +49 6301 712 20 / +48 12 267 68 28
Fax: +49 6301 712 299
e-mail: christophe.debou@q-labs.de

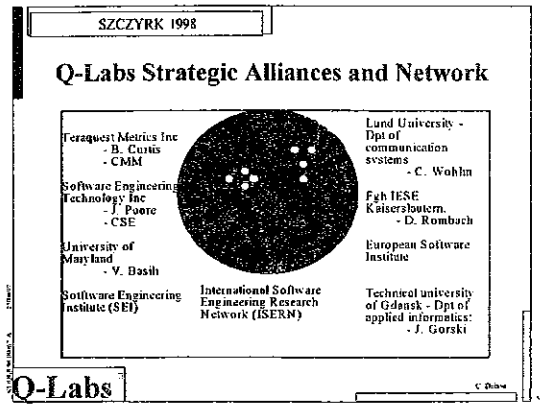
A poor man's guide to Software Project Leader

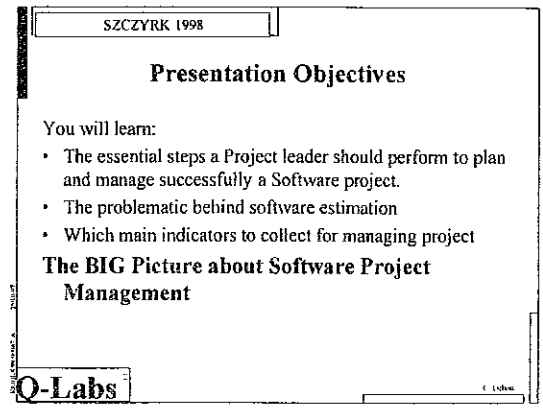


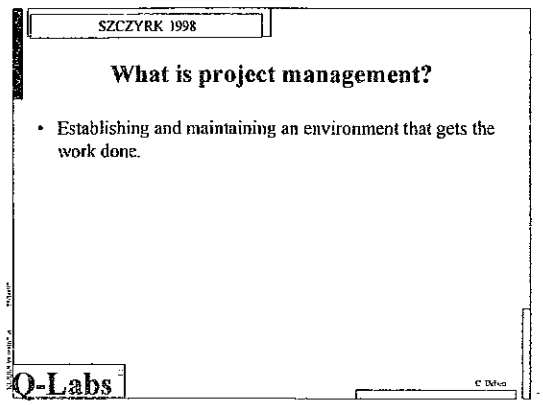


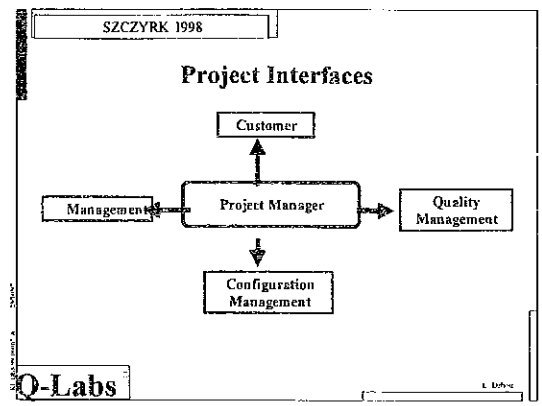


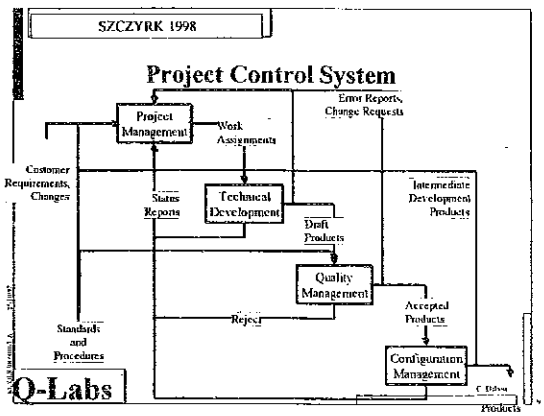


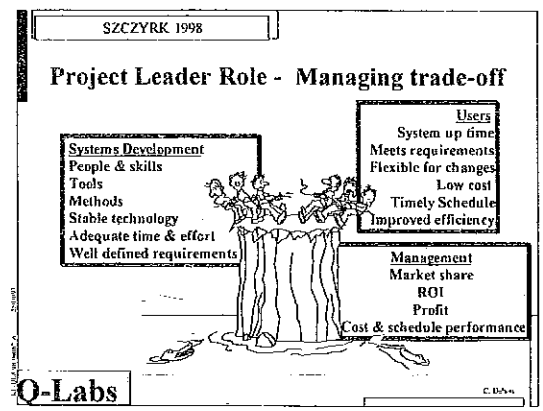

SZCZYRK 1998
Q-Labs Strategic Alliances and Network
Teraquest Metrics Inc
- B. Curtis
- CMM
Software Engineering Technology Inc
- J. Poore
- CSE
University of Maryland
- V. Basili
Software Engineering Institute (SEI)
International Software Engineering Research Network (ISERN)
Lund University - Dpt of communication systems
- C. Wohlin
Fgh IESE Kaiserslautern.
- D. Rombach
European Software Institute
Technical university of Gdansk - Dpt of applied informatics:
- J. Gorski
Q-Labs


SZCZYRK 1998
Presentation Objectives
You will learn:
• The essential steps a Project leader should perform to plan and manage successfully a Software project.
• The problematic behind software estimation
• Which main indicators to collect for managing project
The BIG Picture about Software Project Management
Q-Labs


SZCZYRK 1998
What is project management?
• Establishing and maintaining an environment that gets the work done.
Q-Labs


SZCZYRK 1998
Project Interfaces
Customer
Management
Project Manager
Quality Management
Configuration Management
Q-Labs


SZCZYRK 1998
Project Control System
Project Management
Work Assignments
Error Reports, Change Requests
Customer Requirements, Changes
Status Reports
Technical Development
Intermediate Development Products
Draft Products
Quality Management
Reject
Accepted Products
Standards and Procedures
Configuration Management
Products
Q-Labs


SZCZYRK 1998
Project Leader Role - Managing trade-off
Systems Development
People & skills
Tools
Methods
Stable technology
Adequate time & effort
Well defined requirements
Users
System up time
Meets requirements
Flexible for changes
Low cost
Timely Schedule
Improved efficiency
Management
Market share
ROI
Profit
Cost & schedule performance
Q-Labs

SZCZYRK 1998

Role of the Customer

- The Customer is the Individual or Organization responsible for accepting the product and authorizing payment to the developing Organization.

Q-Labs

SZCZYRK 1998

Role of Customer Representative

- Advantage of an internal customer representative:
  - Can defend the interest of many customers (e.g., generic developments)
  - Necessary in an organization that has not learned to interact positively with the customer
- Define and Approve Requirements
- Provide Support and Information
- Request Changes
- Accept the End Product
- Provide Feedback
- Participate in Project Reviews

Q-Labs

SZCZYRK 1998

Role of Quality Manager

- Ensure the required level of quality is built into all product items by identifying and managing the risks to the project objectives
- Planning the quality activities
  - Involvement with all phases and activities of project
  - Measuring conformance to standards and procedures
  - Report and seek resolution of issues at the appropriate level

Q-Labs

SZCZYRK 1998

Role of Configuration Management

- Establish and maintain the integrity of the products of the software project throughout the project's software life cycle:
  - Define Baselines
  - Identify Configuration Items
  - Manage or control changes to Baselines and Configuration Items
  - Facilitate Requirements Traceability
  - Ensure the reliability of a milestone (in conjunction with quality manager)

Q-Labs

SZCZYRK 1998

Role of Management

- Define long term strategic interests, and ensure that the projects being developed are conform to them:
  - Provide policies
  - Monitor overall status
    - Schedule
    - Costs
    - Risk
    - Functional Contents and Quality
  - Provide resources as appropriate
  - Approve at critical points
  - Ensure correct decisions are made

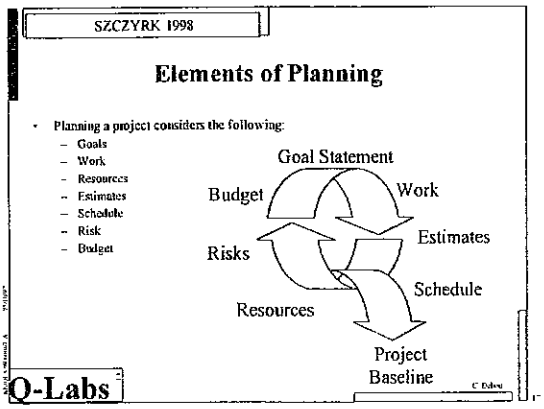
Q-Labs

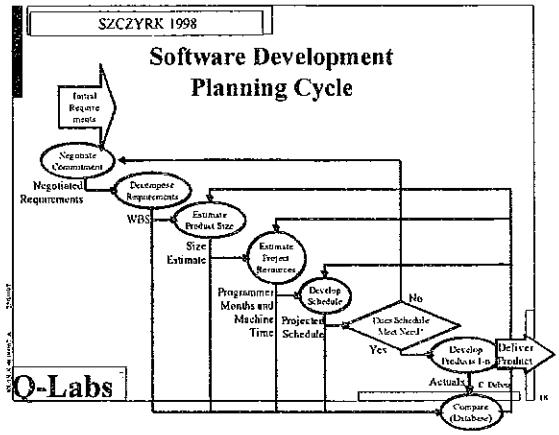
SZCZYRK 1998

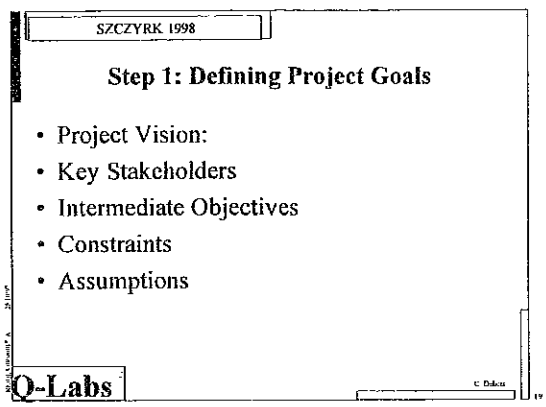
Project Factors

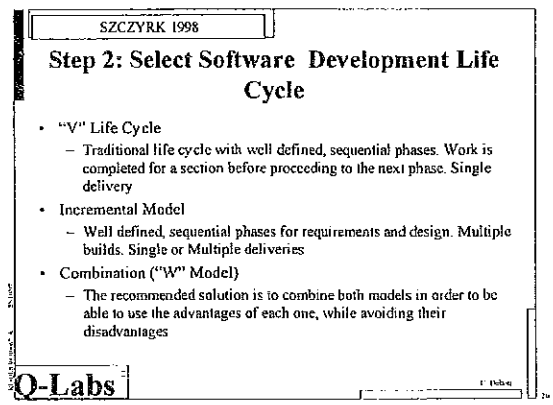
- The project manager is responsible for planning and managing:
  - Scope
  - Resources
  - Time
  - Cost
  - Risk
  - Quality

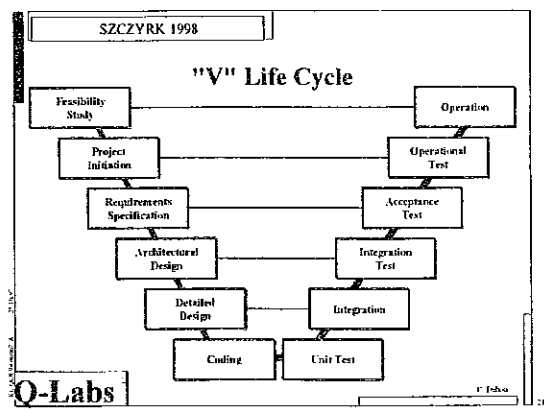
Q-Labs

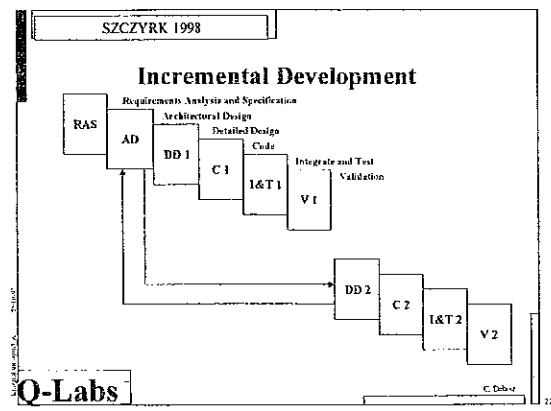

SZCZYRK 1998
Elements of Planning
Planning a project considers the following:
Goals
Work
Resources
Estimates
Schedule
Risk
Budget
Goal Statement
Budget
Work
Estimates
Risks
Schedule
Resources
Project Baseline
Q-Labs


SZCZYRK 1998
Software Development Planning Cycle
Negotiate Commitment
Negotiated Requirements
Decompose Requirements
WBS
Estimate Product Size
Size Estimate
Estimate Project Resources
Programmer Months and Machine Time
Develop Schedule
Projected Schedule
No
Does Schedule Meet Need?
Yes
Develop Product I-n
Deliver Product
Actuals
Compare (Database)
Q-Labs


SZCZYRK 1998
Step 1: Defining Project Goals
Project Vision:
Key Stakeholders
Intermediate Objectives
Constraints
Assumptions
Q-Labs


SZCZYRK 1998
Step 2: Select Software Development Life Cycle
"V" Life Cycle
Traditional life cycle with well defined, sequential phases. Work is completed for a section before proceeding to the next phase. Single delivery
Incremental Model
Well defined, sequential phases for requirements and design. Multiple builds. Single or Multiple deliveries
Combination ("W" Model)
The recommended solution is to combine both models in order to be able to use the advantages of each one, while avoiding their disadvantages
Q-Labs


SZCZYRK 1998
"V" Life Cycle
Feasibility Study
Operation
Project Initiation
Operational Test
Requirements Specification
Acceptance Test
Architectural Design
Integration Test
Detailed Design
Integration
Coding
Unit Test
Q-Labs


SZCZYRK 1998
Incremental Development
Requirements Analysis and Specification
Architectural Design
Detailed Design
Code
Integrate and Test
Validation
RAS
AD
DD 1
C 1
I&T 1
V 1
DD 2
C 2
I&T 2
V 2
Q-Labs

SZCZYRK 1998

Step 3: Define the Work Using

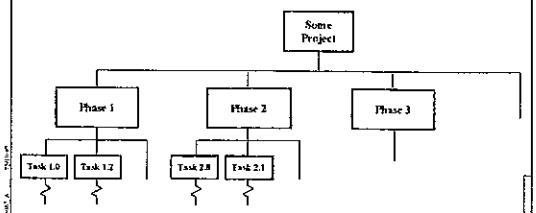
- WBS (Work Breakdown Structure):
  - Top-down chart which represents a hierarchical decomposition of all work to be done in a project
  - Basis for estimating, budgeting, scheduling, tracking,...
- Workpackage:
  - Lowest element of your WBS

Q-Labs

SZCZYRK 1998

WBS Example

Life Cycle Breakdown



Q-Labs

SZCZYRK 1998

Step 4: Estimate Work to be Performed

- Estimate Project Objectives (Effort, Time)
- Using Product objectives
  - Features
  - Reliability, ...
- Through size estimation

Q-Labs

SZCZYRK 1998

Why Estimate?

- The resulting schedule and estimate can be compared with the initial need, and,
  - if they fit, commitment can be made and work can proceed.
  - if, as is more usual, the cost is too high, or the schedule too long, requirements negotiation and re-planning are needed.

Q-Labs

SZCZYRK 1998

Effort

- Estimate the effort required:
  - How long will it take to develop?
  - How many people are needed?
  - What will it cost?
  - What other resources will be required?

Q-Labs

SZCZYRK 1998

Basis for Effort

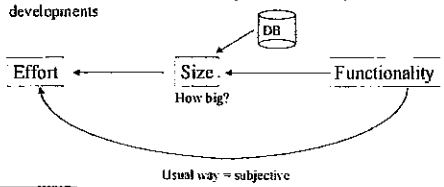
- Effort is based on
  - Size of Development
  - Complexity of Development
  - Environment
  - Experience

Q-Labs

SZCZYRK 1998

Why Estimate Size?

- Size can be Measured Objectively
- Size can be Estimated
- Size can be used as a basis for comparison with completed developments



Q-Labs

SZCZYRK 1998

What is Size?

- The basic size measure is LOC (Lines of Code)
- Alternate possible measures include
  - Function points
  - Feature points
  - Modules
  - Lines of documentation
  - Number of screens
  - Number of test cases
  - ...

Q-Labs

SZCZYRK 1998

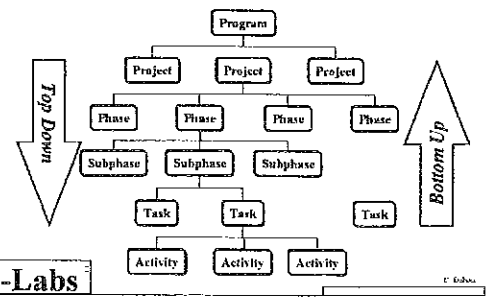
Which Size Measurement?

- A number of different measurements are available and of equivalent usefulness as long as
  - A standard measurement and estimating process is defined
  - The same unit of measurement is used consistently
  - Historical data allows to verify the unit in past projects

Q-Labs

SZCZYRK 1998

Estimating Approaches



Q-Labs

SZCZYRK 1998

Techniques for Estimation: Top-Down

- **Standard method:**
  - Determine an element which can be estimated and which is relevant
  - Apply standard ratio to determine the remainder of the estimate based on historical data.
- **Cost Estimating Tools/Models:**
  - Cocomo, QSM/SLIM
  - Be careful !! Needs calibration.

Q-Labs

SZCZYRK 1998

Top-Down Estimating

- Pros:
  - Global focus
  - If ratios apply, they can be helpful
- Cons:
  - Requires substantial history and experience to compare and differentiate projects
  - Difficulty in identifying "similar" projects
  - Difficult to take into account skill level of delivery staff

Q-Labs

SZCZYRK 1998

Techniques for Estimation - Bottom-up

- **Delphi method:** getting consensus among experts using some statistical method
- **Function Points/Features Points**: Measuring size of the project early in the development by counting standard "functions":
  - Taylor to your own context
  - Requires substantial historical data
- ...

Q-Labs

SZCZYRK 1998

Bottom-Up Estimating

- Pros:
  - Requires use of detail work breakdown structure
  - Minimizes activity omissions
- Cons:
  - Tendency to be lulled into a false sense of security by equating "detailed" with "precise"

Q-Labs

SZCZYRK 1998

Causes of Poor Estimates

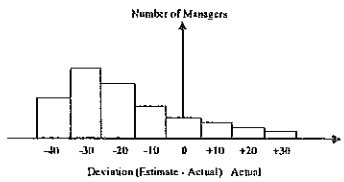
- Fear of measurement
- Lack of time to estimate
- Lack of estimating expertise
- Biases toward underestimating
- Tendency to reduce estimates
- Lack of useful input to the estimating process, such as historical data

Q-Labs

SZCZYRK 1998

Biases in Estimating

"DeMarco" Estimating Experiment

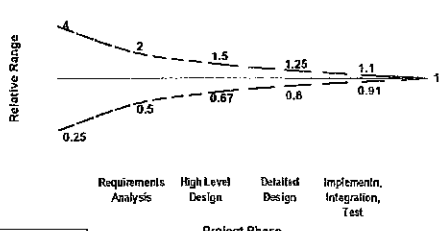


Tom DeMarco, Controlling Software Projects

Q-Labs

SZCZYRK 1998

Expected Software Estimation Accuracy versus Project Phase



Barry Boehm

Q-Labs

SZCZYRK 1998

Contingencies

- Estimates will be inaccurate and these inaccuracies need to be compensated for:
  - Statistical variations can be handled through multiple estimators (Delphi method)
  - Unknown elements of the project can be covered through statistical contingencies

Q-Labs

SZCZYRK 1998

Effort to Duration

- Project Time
  - Determine available time for project work during a work year
- Calculate Project Time Factor:
  - = 180 / 260
  - = .69
- No more than 27½ hours of project time, on average, should be scheduled in a typical 40 hour work week:
  - 5½ hours per day on average

• Maximum available days for project work (5 * 52)	260
• Less non project time	
– Vacation	25
– Holidays	10
– Illness	10
– Training (non-project)	15
– Meetings (non-project)	20
Total available project days	180

Q-Labs

SZCZYRK 1998

Step 5: Scheduling

- Determine Task Dependencies
- Draw a Network Diagram
- Calculate the Critical Path Schedule
- Integrate Milestones into schedule
- Integrate Resources into schedule (levelling according to available resources)
- The first result is probably not the best one: this is an iterative process and needs to be repeated every time the estimates or the development environment changes

Q-Labs

SZCZYRK 1998

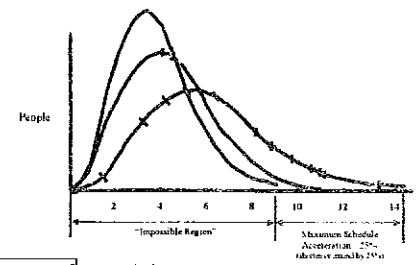
Scheduling tools

- Network Diagram
- Gantt Chat
- Histograms/loading charts
- Pert Diagram (PDM: Precedence Diagramming Method)

Q-Labs

SZCZYRK 1998

Staff Loading Curve: Impossible Region



Barry Boehm
Software Engineering Economics

Q-Labs

SZCZYRK 1998

Step 6: Applying Resources

- OBS (organizational Breakdown Structure):
  - Roles
  - Reporting relationships
- Good practices for organizational structure:
  - Small teams
  - Minimal requirements for communication between teams
  - Cohesive work units
  - Team grouping that minimize effect of work
  - Hierarchical structure

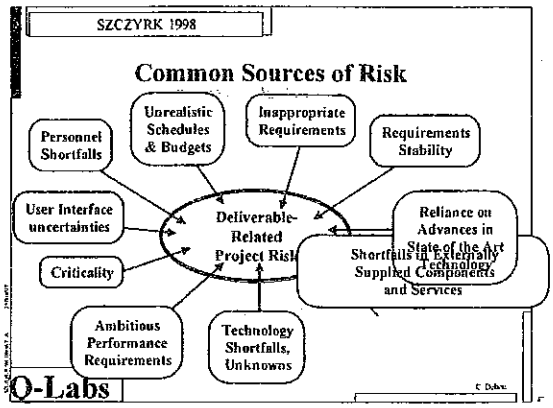
Q-Labs

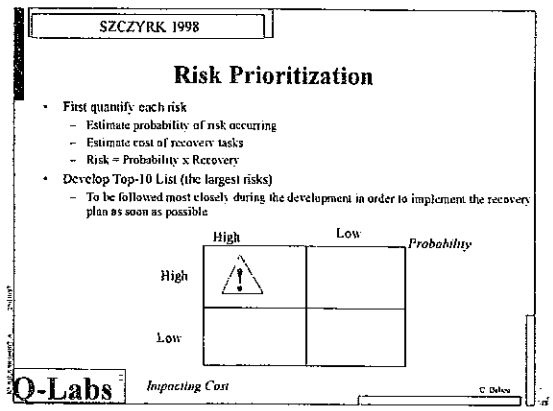
SZCZYRK 1998

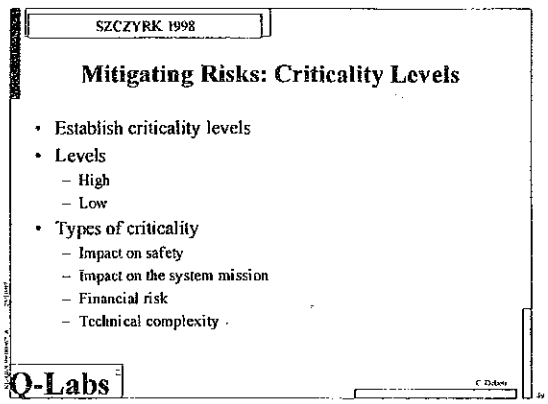
Step 7: Managing Risks

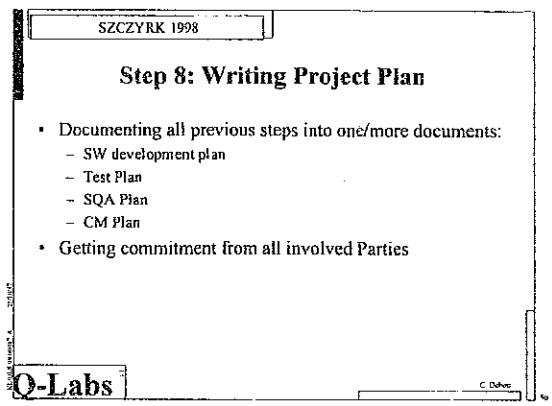
- Risks are any events that COULD impact project
  - cost
  - Schedule
  - Quality
- Risk Management:
  - Risk Assessment (identification, analysis, prioritization)
  - Risk Control (management planning, resolution, monitoring)

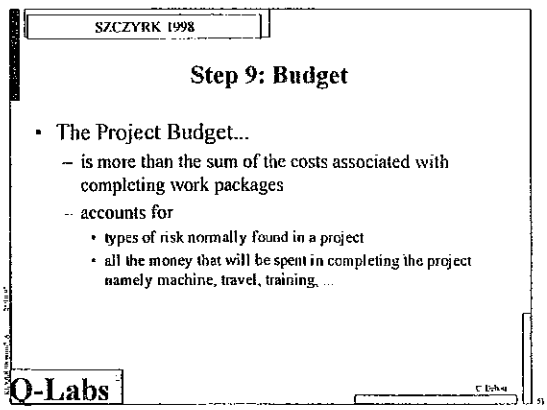
Q-Labs

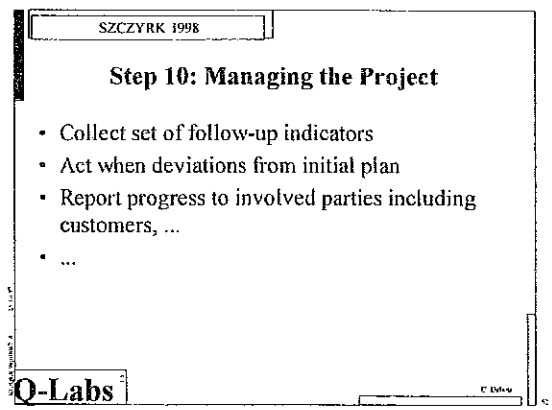

SZCZYRK 1998
Common Sources of Risk
Personnel Shortfalls
Unrealistic Schedules & Budgets
Inappropriate Requirements
Requirements Stability
User Interface uncertainties
Criticality
Deliverable-Related Project Risk
Reliance on Advances in State-of-the-Art Technology
Shortfalls in Externally Supplied Components and Services
Ambitious Performance Requirements
Technology Shortfalls, Unknowns
Q-Labs


SZCZYRK 1998
Risk Prioritization
First quantify each risk
Estimate probability of risk occurring
Estimate cost of recovery tasks
Risk = Probability x Recovery
Develop Top-10 List (the largest risks)
To be followed most closely during the development in order to implement the recovery plan as soon as possible
High
Low
Probability
High
Low
Impacting Cost
Q-Labs


SZCZYRK 1998
Mitigating Risks: Criticality Levels
Establish criticality levels
Levels
High
Low
Types of criticality
Impact on safety
Impact on the system mission
Financial risk
Technical complexity
Q-Labs


SZCZYRK 1998
Step 8: Writing Project Plan
Documenting all previous steps into one/more documents:
SW development plan
Test Plan
SQA Plan
CM Plan
Getting commitment from all involved Parties
Q-Labs


SZCZYRK 1998
Step 9: Budget
The Project Budget...
is more than the sum of the costs associated with completing work packages
accounts for
types of risk normally found in a project
all the money that will be spent in completing the project namely machine, travel, training, ...
Q-Labs


SZCZYRK 1998
Step 10: Managing the Project
Collect set of follow-up indicators
Act when deviations from initial plan
Report progress to involved parties including customers, ...
...
Q-Labs

SZCZYRK 1998

Management Level Indicators

- Progress
  - Actual vs planned task completion (e.g., Gantt chart)
- Effort
  - Actual vs planned staffing profiles
- Cost
  - Actual vs planned costs (cost and schedule variances)
- Computer Resource Use
  - Actual vs planned profiles of computer resource utilization

Q-Labs

SZCZYRK 1998

Management Level Indicators - 2

- SQA Audits
  - Status of non-compliance issues (whether they be product or process related should be registered)
- Reviews
  - Status of Action items
- Trouble Reports
  - Status of trouble reports (number of reports, open, closed, unevaluated, density, comparison between trouble reports and test cases)

Q-Labs

SZCZYRK 1998

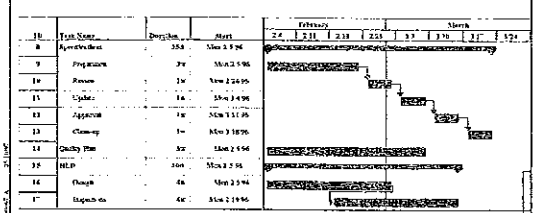
Management Level Indicators - 3

- Requirements Stability
  - Number of requirements changes and clarifications, distribution of requirements over releases
- Size Stability
  - Growth, distribution over releases

Q-Labs

SZCZYRK 1998

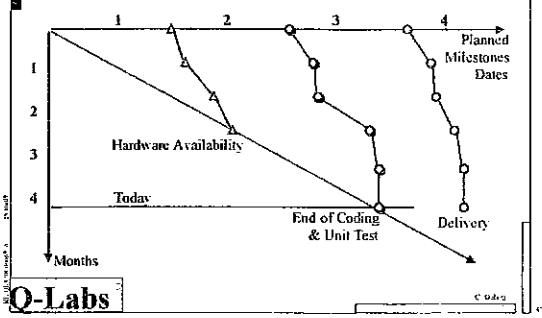
Progress - 1: Gantt Chart



Q-Labs

SZCZYRK 1998

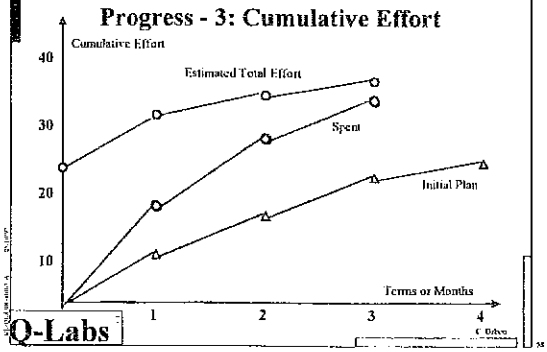
Progress - 2: Milestones



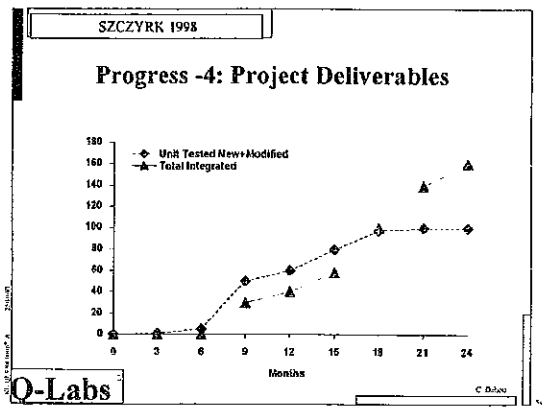
Q-Labs

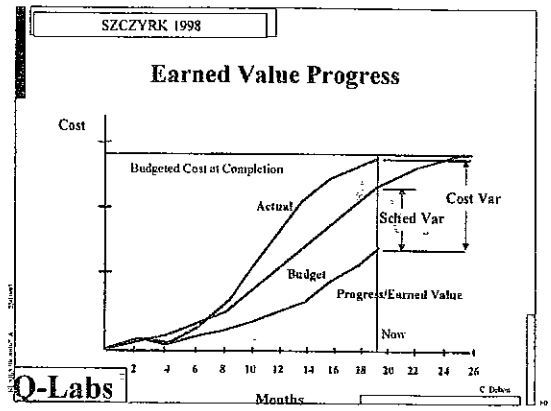
SZCZYRK 1998

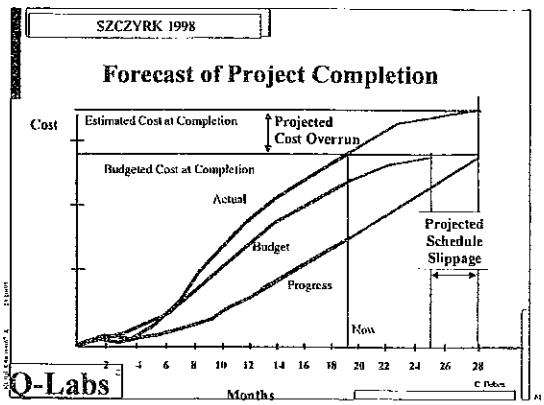
Progress - 3: Cumulative Effort

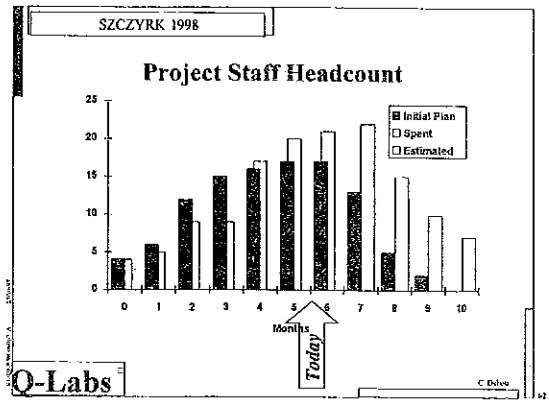


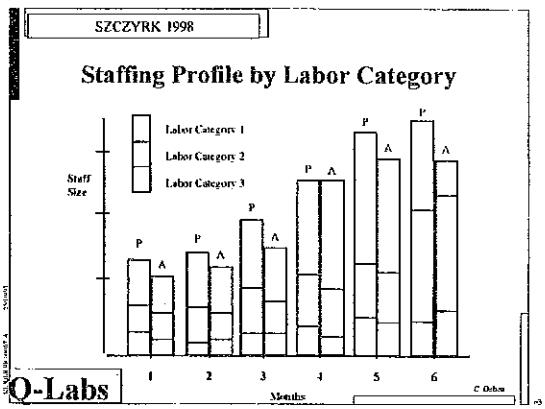
Q-Labs

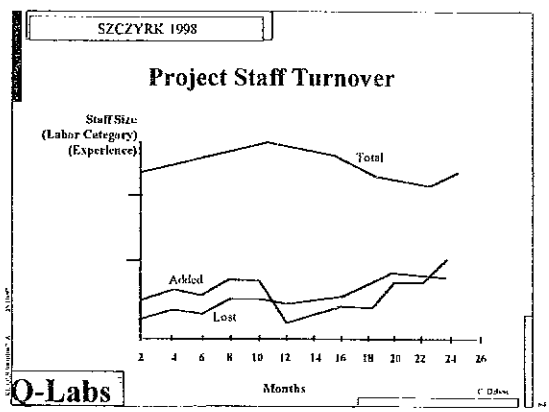

SZCZYRK 1998
Progress -4: Project Deliverables
Unit Tested New+Modified
Total Integrated
Months
Q-Labs


SZCZYRK 1998
Earned Value Progress
Cost
Budgeted Cost at Completion
Actual
Budget
Progress/Earned Value
Sched Var
Cost Var
Now
Months
Q-Labs


SZCZYRK 1998
Forecast of Project Completion
Cost
Estimated Cost at Completion
Projected
Cost Overrun
Budgeted Cost at Completion
Actual
Budget
Progress
Projected
Schedule
Slippage
Now
Months
Q-Labs

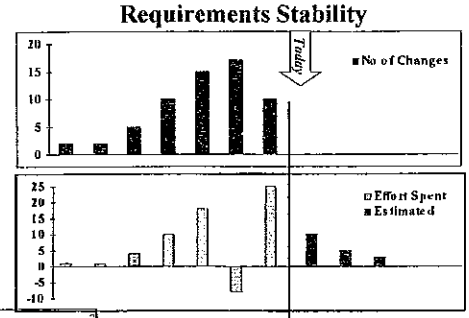

SZCZYRK 1998
Project Staff Headcount
Initial Plan
Spent
Estimated
Months
Today
Q-Labs


SZCZYRK 1998
Staffing Profile by Labor Category
Labor Category 1
Labor Category 2
Labor Category 3
Staff
Size
Months
Q-Labs


SZCZYRK 1998
Project Staff Turnover
Staff Size
(Labor Category)
(Experience)
Total
Added
Lost
Months
Q-Labs

SZCZYRK 1998

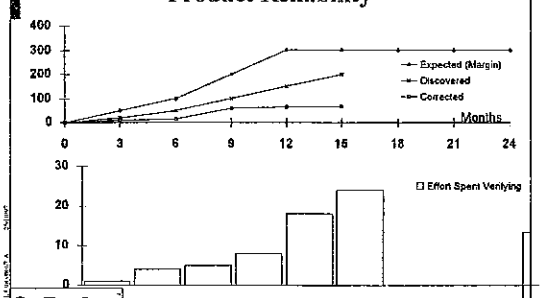
Requirements Stability



Q-Labs

SZCZYRK 1998

Product Reliability



Q-Labs

SZCZYRK 1998

Process Ability to Introduce and Remove Faults

Origin \ Discovery	Req Analysis	TLD Reviews	DD & Code Reviews	Unit Tests	Integ. Tests	Val Tests	Maint
Validation						1	0
Integration					10	20	7
UT				35	15	7	1
DD/Code			85	120	50	20	7
TLD		20	60	2	7	7	0
Req	25	5	55	1	3	4	0

Q-Labs

SZCZYRK 1998

Step 11: Manage Subcontractor

- Subcontracting work = Giving part of the work to an entity that you cannot Directly control (can be internal department, external company)
- Select Subcontractor based on criticality of task and ability to subcontract to achieve it
- Set up contract between prime and subcontractor covering (among others):
  - acceptance criteria
  - Rules when changes of requirements, ...
  - Interface mechanism, ...
- Manage subcontractor with same practices for managing internally your project (continuous follow-up)

Q-Labs

SZCZYRK 1998

Step 12: Collection Lessons Learned

- Need to capture lessons learned in order to
  - collect Data for Historical Data-base
  - Conduct a Project Completion Review
  - Assess the Results of the Review
  - Document the Results
- The data can be used to perform root cause analysis of any problems that occurred and must be stored to help in planning process of future projects

Q-Labs

SZCZYRK 1998

Conclusions

- In addition, Project leaders should be:
  - good negotiator
  - good communicator
  - good decision maker
  - ...

Q-Labs

Janusz Górski

Politechnika Gdańska, Katedra Zastosowań Informatyki,
ul. Narutowicza 11/12, 80-952 Gdańsk
e-mail: jango@pg.gda.pl

Inspekcje oprogramowania

W ramach cyklu życia, oprogramowanie istnieje w wielu różnych reprezentacjach. Jedną z nich jest wykonywalny kod programu. Inspekcje dokumentacji to jedna z najbardziej skutecznych metod wykrywania defektów w dokumentach. Inspekcje mogą być stosowane do dowolnej dokumentacji). Dzięki temu stanowią jedną z podstawowych technik pro-jakościowych stosowaną we wczesnych etapach rozwoju oprogramowania, wtedy gdy kod programu jeszcze nie istnieje. Są więc istotnym uzupełnieniem testowania, które jest tradycyjnie stosowane w stosunku do kodu programu. Artykuł przedstawia proces inspekcji oraz korzyści płynących z stosowania tej techniki.

1. Wprowadzenie

Koszt naprawy defektu wprowadzonego na etapie analizy wymagań, zidentyfikowanego i usuniętego w trakcie projektowania systemu jest 10-krotnie większy niż koszt jego usunięcia na etapie wymagań. Usunięcie takiego błędu w trakcie implementacji systemu może kosztować już 100 razy więcej, a w trakcie eksploatacji - nawet 1000 razy (jest to tzw. reguła 1:10, [1]). Zapewnienie jakości jest ukierunkowane na zapobieganie, wykrywanie i usuwanie defektów we wszystkich dokumentach tworzonych w ramach cyklu życia oprogramowania. Odgrywa zasadniczą rolę w procesie wytwarzania oprogramowania i powinno towarzyszyć jego każdej fazie i aktywności. Wyniki analiz dotyczących jakości oprogramowania wskazują, że większość (ok. 60-70%) defektów zostaje wprowadzona w fazach analizy i projektowania, a jedynie 30-40% - w trakcie kodowania. Jeśli uwzględnimy, że testowanie umożliwia wykrycie jedynie ok. 50% defektów (i to w większości powstałych w procesie kodowania) okazuje się, że ograniczenie się wyłącznie do testowania nie jest skuteczną drogą zapewniania jakości oprogramowania. Należy wprowadzić do użytkowania metody, które umożliwiają eliminację defektów w możliwie najkrótszym czasie po ich wprowadzeniu.

2. Cele inspekcji

Jednym z najbardziej skutecznych sposobów rozwiązania powyższego problemu są inspekcje dokumentacji. Jest to metoda zespołowego wykrywania defektów w produktach procesu wytwarzania oprogramowania. Ze względu na jej wysoką efektywność i uniwersalność jest to jedna z najważniejszych metod zapewnienia jakości.

Rozumiane w sensie potocznym, inspekcje (przeglądy, recenzje) dokumentów stanowią szeroko i od dawna stosowaną technikę pracy z dokumentacją, polegającą na tym, że (inna niż autor) osoba dokonuje krytycznego przeglądu dokumentu, poszukując zawartych w nim defektów. W postaci sformalizowanej oraz w stosunku do oprogramowania metoda inspekcji została zaproponowana przez M. E. Fagana w latach siedemdziesiątych. Metoda ta przyniosła nadspodziewanie dobre wyniki w zakresie poprawy zarówno jakości produktu jak i efektywności procesu jego wytwarzania.

Pośród celów inspekcji wymienić można:

- wykrywanie i identyfikację defektów,
- usunięcie tych defektów,
- formalne potwierdzenie jakości produktu,

- doskonalenie procesu wytwarzania oprogramowania, w oparciu o wyniki analizy przyczyn znalezionych defektów,
- doskonalenie samego procesu inspekcji.

3. Proces inspekcji

Inspekcja obejmuje szereg kroków, prowadzonych w oparciu o zdefiniowane procedury i dokumenty pomocnicze, z udziałem zespołu o jawnie przydzielonych rolach. Każdy krok ma zdefiniowane cele, warunki wejściowe, wyjściowe, wykorzystywane dokumenty, produkty, procedury postępowania oraz uczestników.

Proces inspekcji zostaje zainicjowany w chwili zgłoszenia przez właściciela dokumentu (np. autora) zapotrzebowania na przeprowadzenie inspekcji (*faza rozpoczęcia*). Zgłoszenie to trafia do odpowiedniej komórki, która wyznacza kompetentnego lidera inspekcji. Lider sprawdza, czy produkt oraz produkty źródłowe (czyli te, na podstawie których powstał dokument przewidziany do inspekcji) spełniają określone kryteria wejściowe. Celem tych działań (*faza wejścia*) jest niedopuszczenie do inspekcji produktów źle przygotowanych, których analiza byłaby jedynie stratą czasu. *Faza planowania* obejmuje prace przygotowawcze, np. ustalenie liczby spotkań kontrolnych, wyznaczenie uczestników, przygotowanie harmonogramu prac, itp. *Spotkanie inicjujące* ma na celu wyjaśnienie wszystkim uczestnikom inspekcji jej celów, harmonogramu, oczekiwań w stosunku do nich i ich pracy, rozdanie materiałów, przydział ról, wyjaśnienie procedur inspekcji, określenie parametrów wydajnościowych (np. tempa kontroli indywidualnej). *Kontrola indywidualna* dokumentu jest wykonywana osobno przez każdego z kontrolerów, w oparciu o produkty źródłowe, procedury kontroli, obowiązujące zasady i listy kontrolne. Głównym celem pracy kontrolera jest znalezienie jak największej liczby defektów w badanym dokumencie. *Spotkanie kontrolne* jest kluczową fazą procesu inspekcji. W tej fazie są zbierane i dokumentowane wyniki indywidualnej pracy kontrolerów. Wykryte defekty podlegają klasyfikacji i są przedstawiane w ramach wspólnej struktury danych. Dodatkowym celem spotkania kontrolnego jest znajdowanie i dokumentowanie nowych defektów i zagadnień, które nie zostały wykryte w trakcie pracy indywidualnej. W trakcie spotkania są również zgłaszane i dokumentowane sugestie poprawy samego procesu inspekcji, list kontrolnych, procedur i reguł. Adresatem tych zagadnień jest komórka odpowiedzialna za inspekcje. W ten sposób, w trakcie spotkania kontrolnego są pozyskiwane efekty pracy grupowej. Kolejnym krokiem inspekcji jest *edycja*, w ramach której są rozwiązane problemy udokumentowane w trakcie spotkań kontrolnych. Krok ten najczęściej realizuje autor ocenianego dokumentu. W kolejnym kroku *kontynuacji*, lider inspekcji dokonuje sprawdzenia wyników fazy edycji, rozsyła odpowiednim osobom sugestie zmian produktów źródłowych oraz propozycje poprawy procesu inspekcji. Kolejnym krokiem jest *wyjście*, będące formalnym zatwierdzeniem przez lidera wyników inspekcji. Po zakończeniu tego kroku dokument zostaje oficjalnie udostępniony do dalszego wykorzystania, z podaniem szacunkowej liczby pozostałych w nim defektów.

Dodatkowym etapem inspekcji może być tzw. *burza mózgów*, zwoływana w celu podjęcia próby identyfikacji przyczyn powstania najpoważniejszych defektów wykrytych w badanym dokumencie. Zwykle prowadzi to do sugestii poprawy procesu wytwarzania oprogramowania.

4. Role uczestników

W inspekcji uczestniczy zespół osób pełniących różne role. Możliwe jest spełnianie więcej niż jednej roli przez tę samą osobę. Poniżej krótko omówiono poszczególne role.

Lider inspekcji. Lider jest centralną postacią inspekcji. Od niego w dużej mierze zależy jej jakość i efektywność. Zakres odpowiedzialności lidera obejmuje kontrolę spełnienia warunków wejścia dokumentu, organizację i przeprowadzenie spotkania inicjującego, wybór kontrolerów (w porozumieniu z autorem), przygotowanie spotkania kontrolnego, przeprowadzenie spotkania kontrolnego, klasyfikację i dokumentowanie wykrytych defektów i zagadnień, przekazanie raportu z inspekcji działowi zapewnienia jakości dla celów statystycznych, kontrolę spełnienia warunków wyjścia dokumentu. Lider jest odpowiedzialny za sprawne i skuteczne zrealizowanie całego procesu.

Autor. Autor aktywnie uczestniczy w większości kroków inspekcji. To on podejmuje decyzję o przedłożeniu swojego dokumentu do inspekcji. Bierze udział w spotkaniu inicjującym i najczęściej prowadzi również kontrolę indywidualną swojego produktu oraz uczestniczy w spotkaniach

kontrolnych. Autor pełni też kluczową rolę podczas fazy edycji dokumentu i jest odpowiedzialny za dokonanie odpowiednich zmian.
Sekretarz. Zadaniem sekretarza jest dokumentowanie zagadnień zgłaszanych w trakcie spotkania kontrolnego. Rolę tę może pełnić każdy członek zespołu.
Kontroler. Zwykle w zespole uczestniczy kilku kontrolerów. Zadaniem kontrolera jest znalezienie jak największej liczby defektów, których nie znajdą inni kontrolerzy oraz ich czytelne udokumentowanie. Zakres uwagi różnych kontrolerów może być rozdzielony, np. poprzez wykorzystanie różnych list kontrolnych.

5. Dokumentacja

Proces inspekcji jest wspierany odpowiednią dokumentacją obejmującą *przewodnik inspekcji* zawierający definicję wzorcowego dla danej instytucji procesu inspekcji, w ujęciu ogólnym oraz z punktu widzenia poszczególnych ról, *zasady i listy kontrolne* wykorzystywane w trakcie kontroli indywidualnej oraz różne *formularze* związane poszczególnymi krokami inspekcji (np. formularz zgłoszenia potrzeby inspekcji, formularz planu inspekcji, formularz listy zagadnień/defektów wykrytych podczas inspekcji, formularz podsumowania inspekcji, itp.).

6. Efektywność inspekcji

Efektywność inspekcji należy rozumieć w kategoriach krótko i długoterminowych. W pierwszym wypadku chodzi o wpływ inspekcji na jakość badanego dokumentu w odniesieniu do poniesionych nakładów. W drugim wypadku chodzi o wpływ procesu inspekcji na poprawę praktyk wytwarzania oprogramowania w danej instytucji (łącznie z samym procesem inspekcji). O ile pierwszy z powyższych celów może być osiągnięty stosunkowo łatwo, poprzez zebranie i interpretację danych odnoszących się do indywidualnej inspekcji, to w drugim przypadku niezbędne jest podjęcie dodatkowego wysiłku zbierania, dokumentowania i analizowania dużej ilości informacji nie związanych bezpośrednio z badanym dokumentem. Są to informacje o charakterze jakościowym, np. przyczyny pojawienia się różnych rodzajów defektów oraz ilościowym, np. liczba znalezionych głównych zagadnień na jednostkę czasu kontroli indywidualnej, tempo dokumentowania zagadnień podczas spotkania kontrolnego, pracochłonność całego procesu. Zbieranie tych danych musi być odpowiednio zaplanowane oraz włączone w sferę zarządzania projektem.

Inspekcje nie są darmowe. Wymagają znacznego nakładu prac organizacyjnych i merytorycznych. Pracochłonność pojedynczej inspekcji może łatwo przekroczyć sto roboczogodzin. Dodatkowo należy uwzględnić koszty szkoleń, zarządzania, zbierania danych, analizy metryk, itp. Z drugiej strony, uzysk z inspekcji wyraża się liczbą wykrytych defektów w stosunku do tych, które nie zostały wykryte. Dopiero całościowa ocena uwzględniająca wpływ inspekcji na cały proces wytworzenia produktu informatycznego umożliwia wyciągnięcie właściwych wniosków. Publikowane doświadczenia jednoznacznie oceniają inspekcje jako przedsięwzięcie bardzo opłacalne. Poza zyskiem czysto ekonomicznym, związanym z redukcją całkowitej pracochłonności, wskazywane są inne korzyści ze stosowania tej techniki. Dzięki stosowaniu inspekcji we wszystkich fazach wytwarzania, defekty są usuwane równomiernie, w miarę ich wprowadzania do kolejnych reprezentacji produktu, a nie jedynie w trakcie testowania. Systematyczne stosowanie inspekcji oraz dokumentowanie ich wyników wspomaga doskonalenie procesu wytwarzania poprzez ujawnianie tych jego obszarów które należy poprawić w pierwszej kolejności. Dodatkowe korzyści wynikają z podnoszenia jakości i kwalifikacji wytwórców oprogramowania poprzez zapoznawanie się z wynikami prac innych osób (podczas kontroli indywidualnej), udział w szkoleniach, spotkaniach kontrolnych i „burzach mózgów".

T. Gilb [3] podaje szereg danych ilościowych charakteryzujących wpływ inspekcji na proces wytwarzania oprogramowania: wzrost produktywności o 30% - 100%, skrócenie czasu wytwarzania o 10% - 30%, 5-10-krotna redukcja czasu i kosztu testowania, redukcja kosztów pielęgnacji o rząd wielkości, nieomal „automatyczna" poprawa jakości procesu i produktu, eliminacja poślizgów w dostarczaniu produktów.

7. Podsumowanie

Inspekcja stanowi jedną z technik wykorzystywanych w ramach systemu zapewnienia jakości, który wyznacza właściwy kontekst do jej zastosowania. Muszą służyć osiąganiu jawnie zdefiniowanych celów, a ich zastosowanie musi podlegać normalnym procesom planowania i zarządzania. Sukces w zastosowaniu inspekcji zależy od wielu czynników. Jednym z najważniejszych jest dobre osadzenie procesu inspekcji w ramach systemu zapewniania jakości oraz ścisłe zdefiniowanie kroków, ról, zadań i zasad składających się na ten proces. Wprowadzanie inspekcji do praktyki danej organizacji powinno odbywać się według opracowanego wcześniej scenariusza, uwzględniającego takie czynniki jak: pełne poparcie wyższych szczebli zarządzania, świadomość kadry inżynieryjno-technicznej, szkolenia, tworzenie bazy metryk czy dokumentowanie. Szczególnie ważne są czynniki socjologiczne, psychologiczne, relacje interpersonalne i ogólna atmosfera w organizacji. Inspekcja może łatwo przerodzić się w ocenę autora a nie wytworzonego przez niego dokumentu. Prowadzi to bezpośrednio do konfliktów na gruncie personalnym i stanowi jedno z najpoważniejszych zagrożeń dla skuteczności inspekcji. Podobny efekt dają próby wykorzystania przez kierownictwo metryk zbieranych w trakcie inspekcji do oceny uczestniczących w niej ludzi, a nie do oceny i poprawy samego procesu.

3. Bibliografia

[1] B. W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.
[2] M. E. Fagan, *Design and Code Inspections to Reduce Errors in Program Development*, IBM systems Journal, 15(3).
[3] T. Gilb, D. Graham, *Software Inspection*, Addison-Wesley Publishing Company, 1993.
[4] D. P. Freedman, G.M. Weinberg, *handbook of Walkthroughs, Inspections and Technical Reviews*, Dorset House Pub., New York, 1990.
[5] M. Jurkowlaniec, B. Runge, *Inspekcje produktów informatycznych*, Informatyka, 11, 1997.

Piotr Sobolewski

Systems Integration Services Manager
DIGITAL Equipment Polska

piotr.sobolewski@digital.com

Architektura rozwiązania - z perspektywy Integracji Systemów

digital™

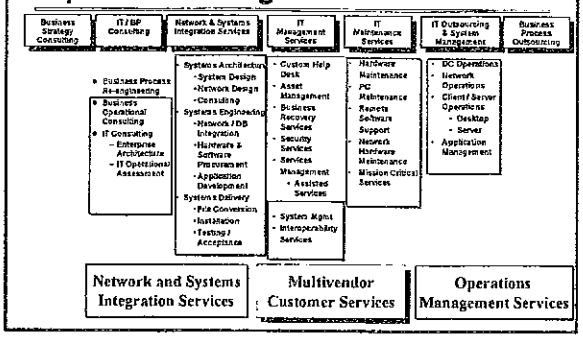
Architektura rozwiązania - z perspektywy Integracji Systemów

Piotr Sobolewski
piotr.sobolewski@digital.com
Systems Integration Services Manager
DIGITAL Equipment Polska

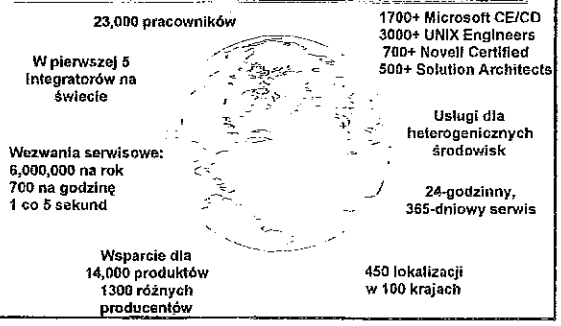
Agenda

- {kilka słów o DIGITALu}
- Architektura rozwiązania
- Rola, cechy i zadania architekta rozwiązania
- Udział architekta rozwiązania w projekcie
- Społeczność architektów w DIGITALu

Spektrum usług



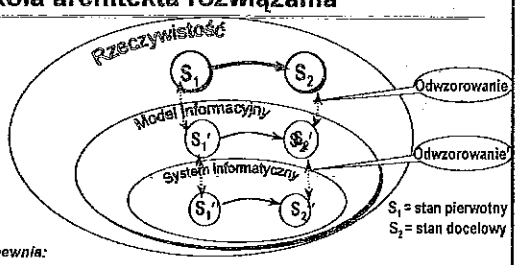
Usługi z DIGITALa - możliwości



Podstawowa terminologia...

- Architektura sugeruje ogólną strukturę i ogólne cechy projektu, jest czuła na wrażenia estetyczne
- Określanie architektury jest częścią inżynierii systemu najbardziej związaną z określaniem celu, formułowaniem idei i struktury systemu, oraz weryfikacją wykorzystania przez użytkownika
- Cel: Dlaczego ktoś chce systemu?
- Idea i struktura: Ogólne właściwości struktury, z różnych punktów widzenia (różne projekcje)
- Weryfikacja: Określenie w jakim stopniu system spełnia potrzeby

Rola architekta rozwiązania



AR zapewnia:

- Wystarczające zrozumienie zmian zachodzących w rzeczywistym świecie
- System informacyjny musi odwzorowywać te zmiany
- Poprawne odwzorowanie bytów rzeczywistych do modelu informacyjnego i systemu informatycznego

Rola architekta rozwiązania

- Architekt rozwiązania jest odpowiedzialny za przewodzenie podczas definiowania i tworzenia skomplikowanego systemu informacyjnego
  - Dopasowanie odpowiednich technologii do potrzeb klienta w celu stworzenia całościowego projektu systemu na wysokim poziomie (architektura)
  - Utrzymywanie koncepcyjnej spójności architektury w trakcie budowy sytemu (współpraca z klientem i wykonawcami systemu)
  - Zweryfikowanie czy zbudowany system spełnia kryteria zdefiniowane w architekturze

Architektura vs inżynieria

Właściwość	Architektura	Arch. i Inż.	Inżynieria
Położenie/ Cele	Słabo zestrukturalizowane /Satysfakcja	Skrępowany /Zgodność	Zrozumiałe/ Optymalizacja
Metody	Heurystyki Synteza Sztuka i Nauka	↔ ↔ Sztuka i Nauka	Równania Analiza Nauka i Sztuka
Integralność systemu poprzez	"Indywidualna wizja"	Jasne cele	Formalna metodyka
Zagadnienia Organizacyjne	Pracuje dla klienta	Pracuje z klientem	Pracuje dla budowniczych
	Koncepcja i weryfikacja	Cały cykl życia projektu	Spełnienie wymagań projektu
	Zaufanie	Konflikt interesów	Zysk v. koszt

zaczerpnięte z Mark W.Maier, *The Art of Systems Architecting*

Cechy architekta rozwiązań

- Pasja do nauki (prawdopodobnie podstawowa cecha)
  - Nadążanie za rozwojem technologii, metodologii, narzędzi
  - Umiejętność działania w niejasnej sytuacji
  - Praca w zmiennym środowisku dopóki nie zostaną określone wymagania i rozwiązania technologiczne będące w stanie je spełnić
- Zrozumienie uwarunkowań politycznych
  - Stworzenie odpowiedniego środowiska dla osiągnięcia sukcesu
- Doświadczenie w komunikacji (rozmowy, słowo pisane)
  - Zrozumienie i wpływanie na innych

Tradycyjne atrybuty architekta

- Holistyczny punkt widzenia
  - Interaktywne określanie form i funkcji
  - Zmienna głębokość technicznego zrozumienia
- Rola w cyklu życia
  - Praca począwszy od określenia koncepcji, mniejsze natężenie podczas implementacji, weryfikacja użyteczności
  - Wybieranie stabilnych pośrednich form w ewolucyjnym, inkrementacyjnym projekcie
- Relacje
  - Często pracuje bezpośrednio dla klienta systemu
- Kluczowy opis:
  - Architekt tworzy satysfakcjonującą i użyteczną koncepcję systemu, zapewnia jej spójność podczas całego cyklu tworzenia oraz weryfikuje czy system został zbudowany dla klienta

Architekt musi...

- Stworzyć opisy architektury (modele) łatwe do zrozumienia dla klientów i budowniczych
  - Zdefiniowanie podstawowych atrybutów systemu, które determinują wartość, koszt, ryzyko i wykonalność
- Przekształcać potrzeby klienta w uwarunkowania techniczne
  - Które można gładko przekształcić w szczegółowe decyzje projektowe
- Syntetyzować koncepcje struktury systemu na podstawie słabo strukturalizowanych przesłanek
- Podejmować decyzje dotyczące problemów technicznych bez pełnej ich znajomości
- Widzieć system z różnych punktów widzenia

Przegląd podstawowych zadań AR

- *PRZYGOTOWANIE*
  - Utrzymanie i rozwijanie umiejętności
  - Utrzymywanie relacji z potencjalnymi klientami
- *WYGRANIE*
  - Pomoc w identyfikacji, oszacowaniu, wygraniu i rozpoczęciu zyskownego projektu
- *TWORZENIE ARCHITEKTURY*
  - Opracowanie proponowanego systemu poprzez analizę, projekt i planowanie
- *KONSTRUOWANIE*
  - Przewodzenie w tworzeniu systemu
- *ZAMKNIĘCIE*
  - Weryfikacja, że system pracuje i jak najszybsze jego wdrożenie

Zadania ER: *Przygotowanie*

- *NAUKA*
  - gromadzenie wiedzy, umiejętności i doświadczeń
  - główne obszary: wiedza ogólna, technologia, rozwój rynku
- *PLANOWANIE*
  - planowanie kariery jako sekwencji wyzwań, doświadczeń z projektów i szkoleń
- *WYKONANIE*
  - praca przy projektach; nauka poprzez obserwacje i poprzez wykonawstwo
- *REFLEKSJE*
  - wyciąganie ważnych lekcji z własnych doświadczeń
- *KONTAKTY*
  - tworzenie i wykorzystanie sieci podobnych sobie profesjonalistów
- *RELACJE*
  - rozwijanie relacji z klientami

Zadania AR: *Wygranie*

- *POŁĄCZENIE*
  - praca ze sprzedawcą przy identyfikacji projektu
- *OCENA*
  - sprawdzenie, że okazja jest warta dalszych inwestycji
- *ZROZUMIENIE*
  - zrozumienie od czego zależy wygrana
- *ZAKRES*
  - określenie granic systemu
- *PROPOZYCJA*
  - współprzygotowanie formalnego opisu sytemu
- *DEMONSTRACJA*
  - dowiedzenie wykonalności systemu dla kluczowych aspektów
- *ZGODA*
  - wzajemne zrozumienie dla kluczowych aspektów projektu
- *ZAMKNIĘCIE*
  - asysta podczas negocjacji kontraktu

Zadania AR: *Architektura*

- *ANALIZA*
  - wydobycie wymagań, zdefiniowanie warunków, przegląd możliwości, identyfikacja i rozwiązanie problemów i ryzyka, określenie standardów
  - odkrycie co jest rzeczywiście potrzebne, a co można pominąć
  - udokumentowanie uwarunkowań i jasne określenie czego nie robić
- *PROJEKT* całości systemu, *SKUPIENIE* uwagi na krytycznych elementach
  - selekcja platformy, komponentów i narzędzi
  - spełnienie wymagań z wykorzystaniem dostępnej technologii
  - maksymalne wykorzystanie gotowych komponentów i platformy
  - zdefiniowanie i wyjaśnienie kryteriów odbioru i innych standardów
- *PLANOWANIE*
  - cykl życia, inkrementy, zadania, sekwencje i kamienie milowe
  - oszacowanie kosztów, planu i zasobów
  - demonstracja wykonalności: technologiczna i czasowa
- *STWORZENIE* i *PRZEWODZENIE* zespołowi architektonicznemu

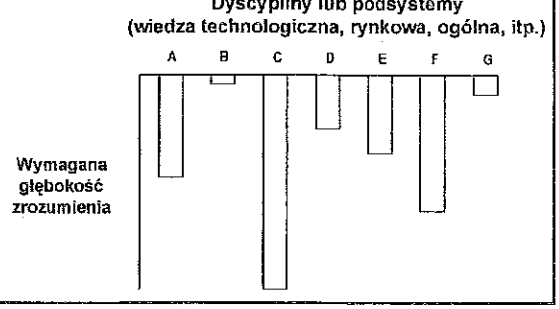
Zadania ER: *Konstruowanie*

- *INTERPRETACJA, DOKUMENTOWANIE i ZATWIERDZANIE projektu*
  - weryfikacja podejmowanych decyzji względem założeń i wymagań
  - nadzór nad spornymi kwestiami, dokumentacja rozwiązań
  - testowanie, zbieranie informacji, reagowanie, zmienianie planów
- *PRZEWODZENIE w tworzeniu* i ZARZĄDZANIE ryzykiem
  - utrzymanie integralności koncepcji I
  - wyłapywanie obszarów ryzyka: źle określonego zakresu, zasobów i skali komplikacji
  - określenie planów w celu redukcji zagrożeń
- *REAKCJA* na napływające informacje i zewnętrzne zmiany
  - identyfikacja pożądanych i niepożądanych zdarzeń i przeszkód
  - definicja możliwych alternatywnych kierunków i podejść
- *ZMIANA PLANÓW* na początku każdego inkrementu
  - skupienie uwagi na iteracyjnym rozwijaniu w oparciu o napływające informacje od klienta, wyniki testów prototypów i instalacji pilotowych
  - ocena zmian w wymaganiach technicznych i biznesowych

Zadania AR: *Zamknięcie*

- *ZATWIERDZANIE*
  - testowanie systemu w oparciu o kryteria produkcji
- *URUCHOMIENIE*
  - szkolenie: dla użytkowników i administratorów
  - przekazanie odpowiedzialności za operacyjne działania
- *AKCEPTACJA* (wiesz kiedy system został zrobiony)
  - QA, testy odbiorcze, regresywne, weryfikacja użyteczności
- *SPRAWDZENIE satysfakcji klienta*
  - upewnienie się, że klient jest usatysfakcjonowany
- *EVALUACJA* i *DOKUMENTACJA* poziomu osiągniętego sukcesu
  - przegląd wszystkich aspektów w celu zrozumienia co działało, a co sprawiło problemy
  - dokumentowanie i dzielenie się naukami
- *PLANOWANIE* utrzymania i ewolucji systemu

Zrozumienie różnych zagadnień przez architekta

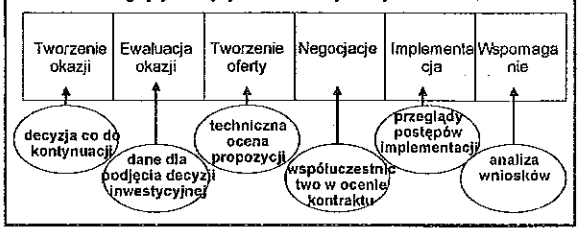


Role w zespole architektonicznym

- Główny architekt
- Architekt
- Ekspert od interfejsu
- Projektant grafiki
- Tester
- Projektant modelu danych
- Ekspert baz danych
- Ekspert TP
- Ekspert od bezpieczeństwa
- Specjalista od sieci
- Ekspert od QA
- Ekspert dziedzinowy
- Ekspert od modelowania OO
- Ekspert od integracji apl.
- Ekspert od middleware
- Ekspert od zarządzania środ.
- Specjalista systemowy
- Ekspert badania wydajności
- Ekspert od zarządzania siecią
- Pisarz dokumentacji
- Szkoleniowiec
- Administrator

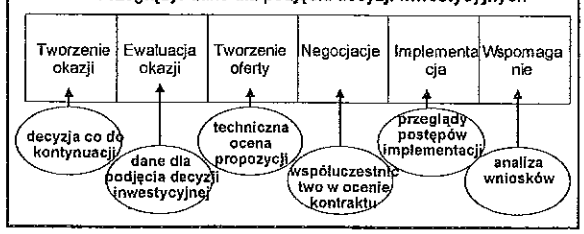
Udział AR w cyklu życia projektu

- Tworzenie okazji
  - Asystowanie w identyfikacji okazji
  - Ewaluacja profilu biznesowego
  - Przeglądy: decyzja co do kontynuacji



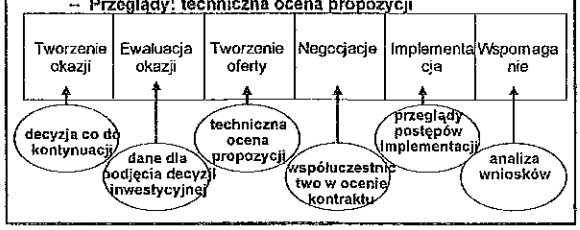
Udział AR w cyklu życia projektu

- Ewaluacja okazji
  - Ewaluacja okazji / zapytanie ofertowe
  - Wstępna definicja architektury rozwiązania
  - Przeglądy: dane dla podjęcia decyzji inwestycyjnych



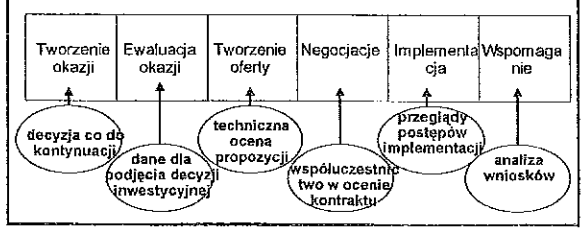
Udział AR w cyklu życia projektu

- Tworzenie oferty
  - Stworzenie propozycji architektury rozwiązania
  - Asystowanie w określaniu listy rzeczy do zrobienia
  - Asystowanie w decyzjach finansowych
  - Przeglądy: techniczna ocena propozycji



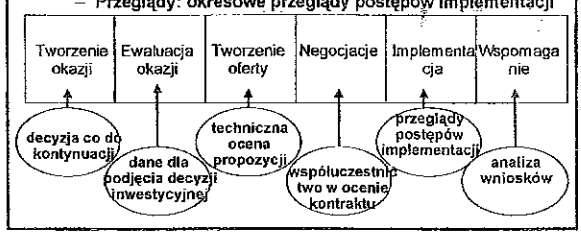
Udział AR w cyklu życia projektu

- Negocjacje
  - Ewaluacja jaki wpływ mają zmiany na koszt i ryzyko
  - Przeglądy: współuczestnictwo w ocenie kontraktu



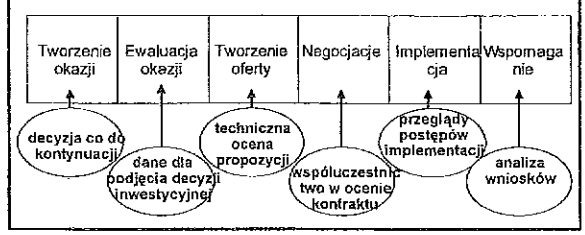
Udział AR w cyklu życia projektu

- Implementacja
  - Tworzenie technicznej architektury i planu implementacji
  - Ocena jaki wpływ na architekturę mają zmiany
  - Zapewnienie jakości dostarczanego rozwiązania
  - Przeglądy: okresowe przeglądy postępów implementacji



Udział AR w cyklu życia projektu

- Wspomaganie
  - Gromadzenie doświadczeń
  - Przeglądy: analiza wniosków



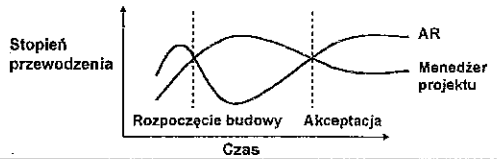
Relacje architekta rozwiązania i menedżera projektu

Architekt:

Odpowiedzialny za zdefiniowanie czym jest system (i jak może zostać zbudowany)

Menedżer projektu:

- *Odpowiedzialny za zbudowanie systemu (i generacje zysku)*



Gdzie jest potrzebny architekt rozwiązania?

Architekci rozwiązań w DIGITALu

- Organizacja Architektów
  - DSAT - DIGITAL Solution Architecture Technique
    - Technika tworzenia architektury rozwiązania
  - Komitet AR
  - Specjalizowane grupy robocze
  - Społeczność AR
    - Metodyka
    - Wymiana doświadczeń
    - Rozwój zawodowy

DSAT - cztery punkty widzenia
uwarunkowania w realizacji projektu

- **Dlaczego** dane przedsięwzięcie jest realizowane, jakie są motywy?
- **Co** będzie robił system, który ma być stworzony?
- **Jaka** będzie struktura systemu?
- **Czym** będziemy tworzyć system i w kontekście jakiej organizacji?

DSAT - cztery dziedziny
zakresy przy określaniu architektury

- Biznes (dlaczego?)
  - Punkt widzenia klienta
- Funkcjonalność (co?)
  - Projekcja całości systemu z punktu widzenia : eksploatacji, właściwości, usług,...
  - Niezależność od technologii, produktów i implementacji
- Technologia (jak?)
  - Projekcja infrastruktury, danych, aplikacji
- Implementacja (czym?)
  - Projekcja obecnych i przyszłych uwarunkowań technicznych i organizacyjnych

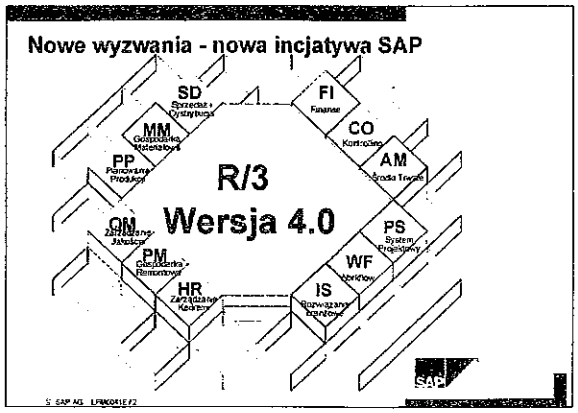
digital™

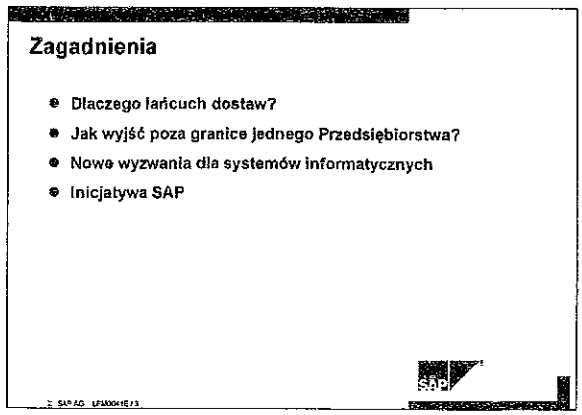
Digital Equipment Corporation
© 1998

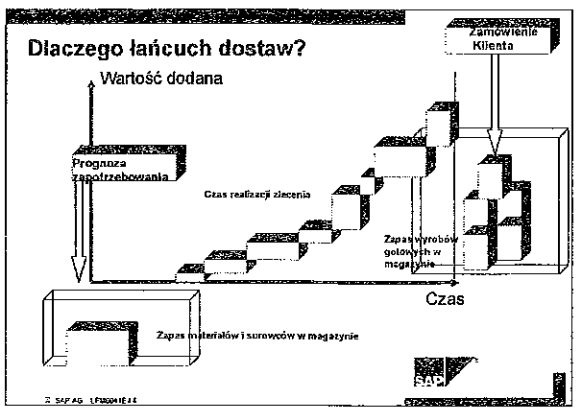
Dariusz Samól

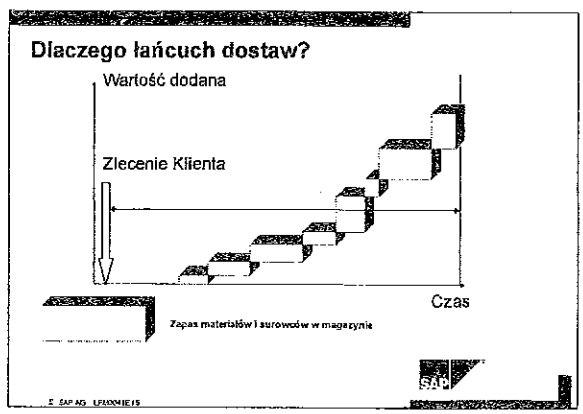
SAP Polska Sp. z o.o.

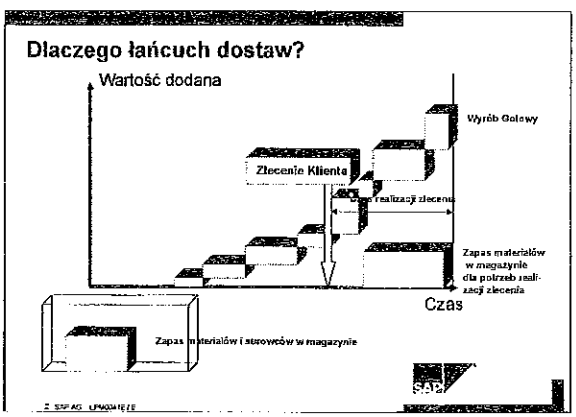
Integracja łańcucha dostaw poprzez moduły oprogramowania R3 v 4.0

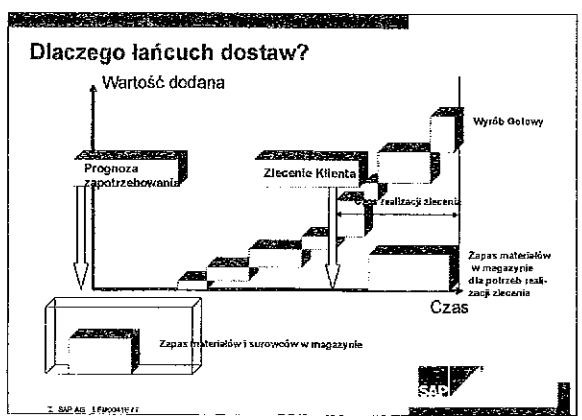

Nowe wyzwania - nowa inicjatywa SAP
SD
FI
MM
CO
PP
AM
R/3
Wersja 4.0
QM
PS
PM
WF
HR
IS

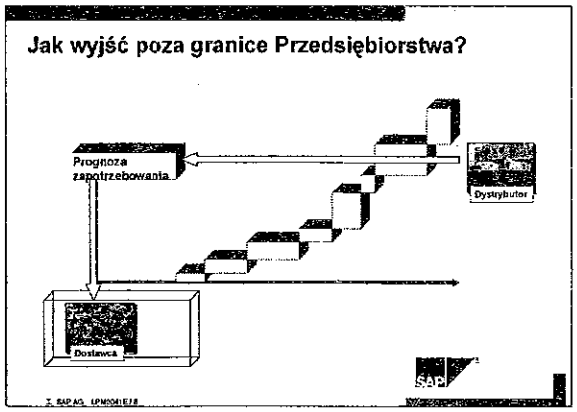

Zagadnienia
Dlaczego łańcuch dostaw?
Jak wyjść poza granice jednego Przedsiębiorstwa?
Nowe wyzwania dla systemów informatycznych
Inicjatywa SAP


Dlaczego łańcuch dostaw?
Zamówienie Klienta
Wartość dodana
Prognoza zapotrzebowania
Czas realizacji zlecenia
Zapas wyrobów gotowych w magazynie
Czas
Zapas materiałów i surowców w magazynie

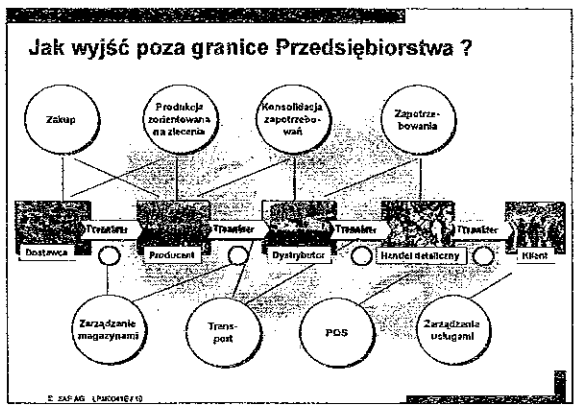

Dlaczego łańcuch dostaw?
Wartość dodana
Zlecenie Klienta
Czas
Zapas materiałów i surowców w magazynie

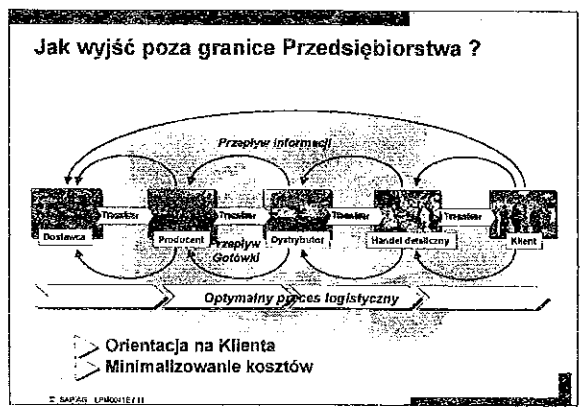

Dlaczego łańcuch dostaw?
Wartość dodana
Wyrób Gotowy
Zlecenie Klienta
Zapas materiałów w magazynie dla potrzeb realizacji zlecenia
Czas
Zapas materiałów i surowców w magazynie

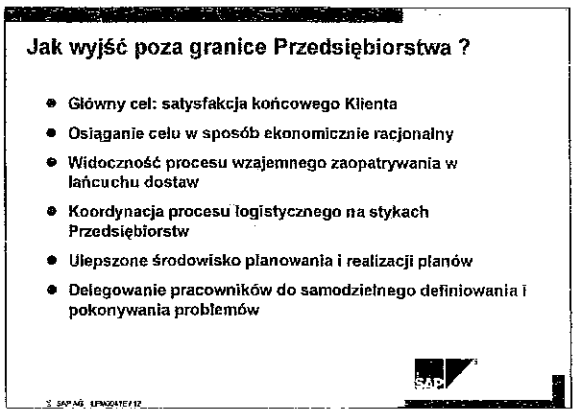

Dlaczego łańcuch dostaw?
Wartość dodana
Wyrób Gotowy
Prognoza zapotrzebowania
Zlecenie Klienta
Zapas materiałów w magazynie dla potrzeb realizacji zlecenia
Czas
Zapas materiałów i surowców w magazynie

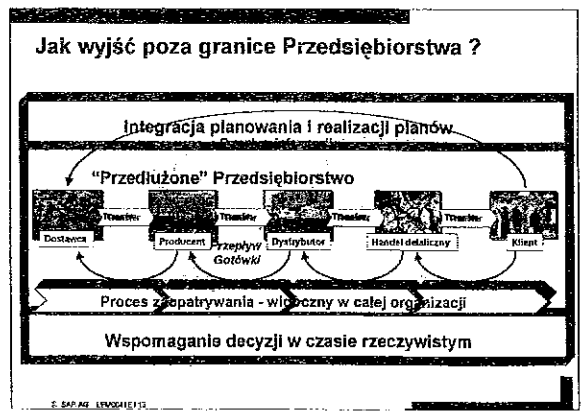

Jak wyjść poza granice Przedsiębiorstwa?
Prognoza zapotrzebowania
Dystrybutor
Dostawca

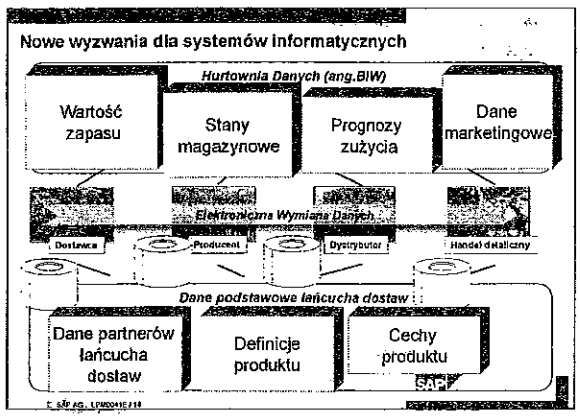
Jak wyjść poza granice Przedsiębiorstwa ?
Transfer
Dostawca
Producent
Dystrybutor
Handel detaliczny
Klient
Optymalny proces logistyczny
Orientacja na Klienta
Minimalizowanie kosztów

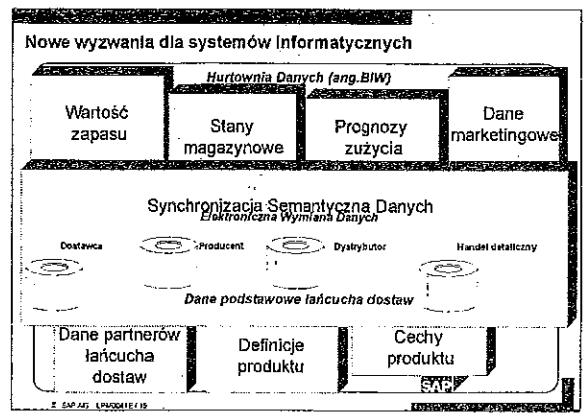

Jak wyjść poza granice Przedsiębiorstwa ?
Zakup
Produkcja zorientowana na zlecenia
Konsolidacja zapotrzebowań
Zapotrzebowania
Transfer
Dostawca
Producent
Dystrybutor
Handel detaliczny
Klient
Zarządzanie magazynami
Transport
POS
Zarządzanie usługami

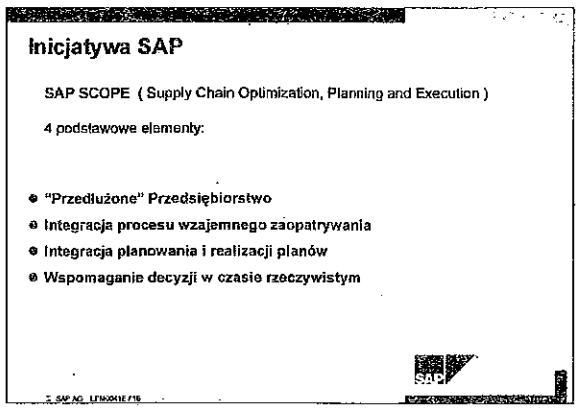

Jak wyjść poza granice Przedsiębiorstwa ?
Przepływ informacji
Transfer
Dostawca
Producent
Dystrybutor
Handel detaliczny
Klient
Przepływ Gotówki
Optymalny proces logistyczny
Orientacja na Klienta
Minimalizowanie kosztów

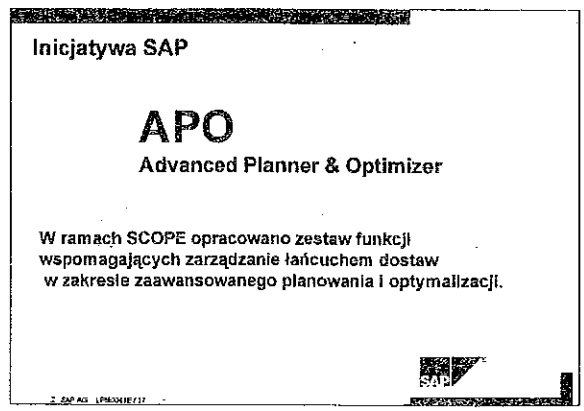

Jak wyjść poza granice Przedsiębiorstwa ?
Główny cel: satysfakcja końcowego Klienta
Osiąganie celu w sposób ekonomicznie racjonalny
Widoczność procesu wzajemnego zaopatrywania w łańcuchu dostaw
Koordynacja procesu logistycznego na stykach Przedsiębiorstw
Ulepszone środowisko planowania i realizacji planów
Delegowanie pracowników do samodzielnego definiowania i pokonywania problemów

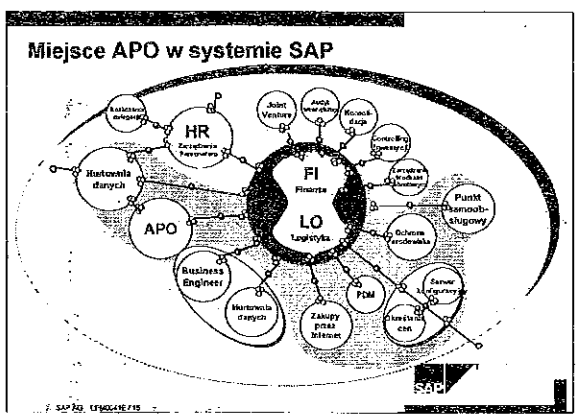

Jak wyjść poza granice Przedsiębiorstwa ?
Integracja planowania i realizacji planów
"Przedłużone" Przedsiębiorstwo
Transfer
Dostawca
Producent
Dystrybutor
Handel detaliczny
Klient
Przepływ Gotówki
Proces zaopatrywania - widoczny w całej organizacji
Wspomaganie decyzji w czasie rzeczywistym

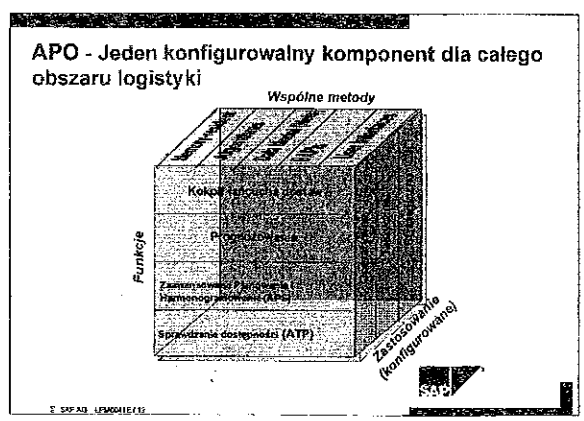

Nowe wyzwania dla systemów informatycznych
Hurtownia Danych (ang.BIW)
Wartość zapasu
Stany magazynowe
Prognozy zużycia
Dane marketingowe
Elektroniczna Wymiana Danych
Dostawca
Producent
Dystrybutor
Handel detaliczny
Dane podstawowe łańcucha dostaw
Dane partnerów łańcucha dostaw
Definicje produktu
Cechy produktu
SAP

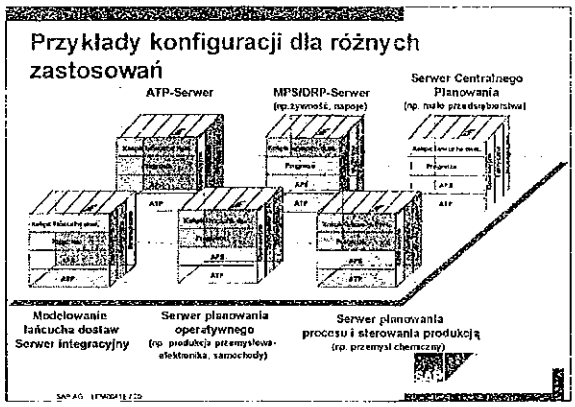

Nowe wyzwania dla systemów informatycznych
Hurtownia Danych (ang.BIW)
Wartość zapasu
Stany magazynowe
Prognozy zużycia
Dane marketingowe
Synchronizacja Semantyczna Danych
Elektroniczna Wymiana Danych
Dostawca
Producent
Dystrybutor
Handel detaliczny
Dane podstawowe łańcucha dostaw
Dane partnerów łańcucha dostaw
Definicje produktu
Cechy produktu
SAP

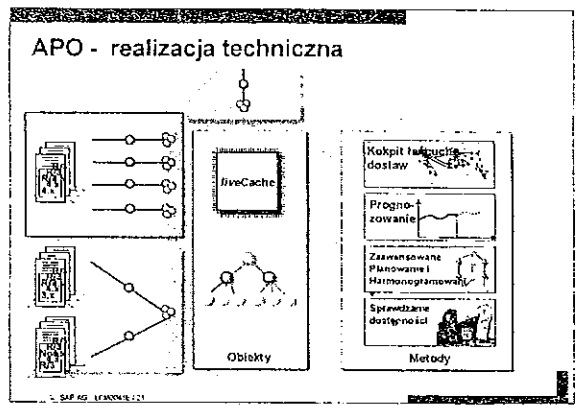

Inicjatywa SAP
SAP SCOPE (Supply Chain Optimization, Planning and Execution)
4 podstawowe elementy:
"Przedłużone" Przedsiębiorstwo
Integracja procesu wzajemnego zaopatrywania
Integracja planowania i realizacji planów
Wspomaganie decyzji w czasie rzeczywistym
SAP

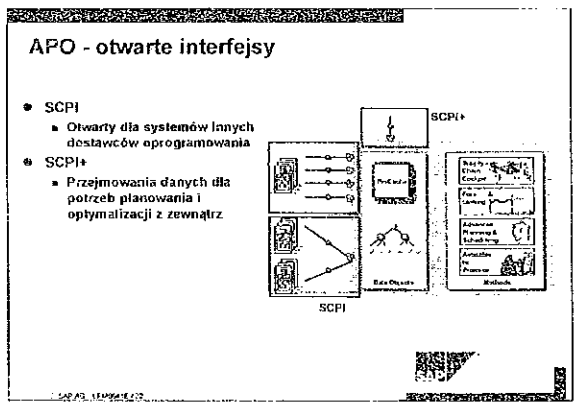

Inicjatywa SAP
APO
Advanced Planner & Optimizer
W ramach SCOPE opracowano zestaw funkcji wspomagających zarządzanie łańcuchem dostaw w zakresie zaawansowanego planowania i optymalizacji.
SAP

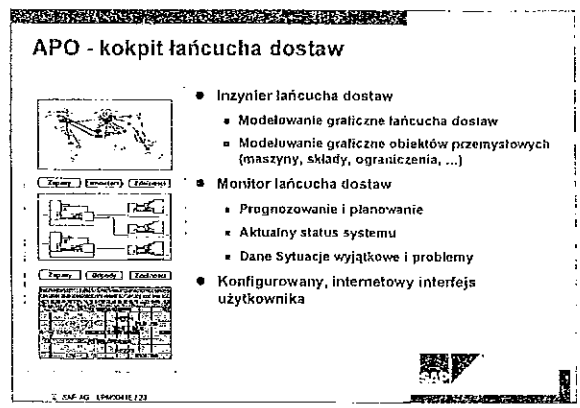

Miejsce APO w systemie SAP
HR
FI
Finanse
LO
Logistyka
APO
Hurtownia danych
Punkt samoobsługowy
Business Engineer
PDM
SAP

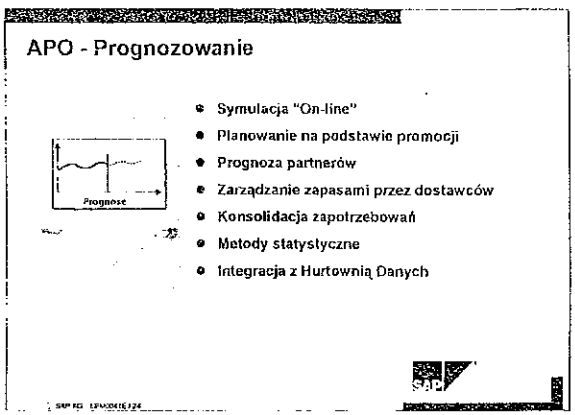

APO - Jeden konfigurowalny komponent dla całego obszaru logistyki
Wspólne metody
Funkcje
Sprawdzanie dostępności (ATP)
Zastosowanie (konfigurowanie)
SAP

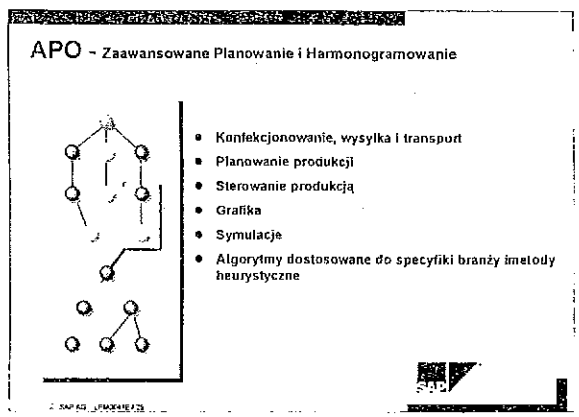

Przykłady konfiguracji dla różnych zastosowań
ATP-Serwer
MPS/DRP-Serwer
Serwer Centralnego Planowania
Modelowanie łańcucha dostaw Serwer integracyjny
Serwer planowania operatywnego
Serwer planowania procesu i sterowania produkcją


APO - realizacja techniczna
liveCache
Obiekty
Kokpit łańcucha dostaw
Prognozowanie
Zaawansowane Planowanie i Harmonogramowanie
Sprawdzanie dostępności
Metody

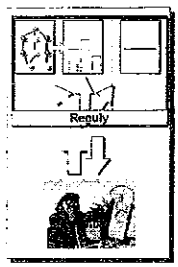

APO - otwarte interfejsy
SCPI
Otwarty dla systemów innych dostawców oprogramowania
SCPI+
Przejmowania danych dla potrzeb planowania i optymalizacji z zewnątrz
SCPI+
SCPI


APO - kokpit łańcucha dostaw
Inżynier łańcucha dostaw
Modelowanie graficzne łańcucha dostaw
Modeluwanie graficzne obiektów przemysłowych (maszyny, składy, ograniczenia, ...)
Monitor łańcucha dostaw
Prognozowanie i planowanie
Aktualny status systemu
Dane Sytuacje wyjątkowe i problemy
Konfigurowany, internetowy interfejs użytkownika


APO - Prognozowanie
Prognose
Symulacja "On-line"
Planowanie na podstawie promocji
Prognoza partnerów
Zarządzanie zapasami przez dostawców
Konsolidacja zapotrzebowań
Metody statystyczne
Integracja z Hurtownią Danych

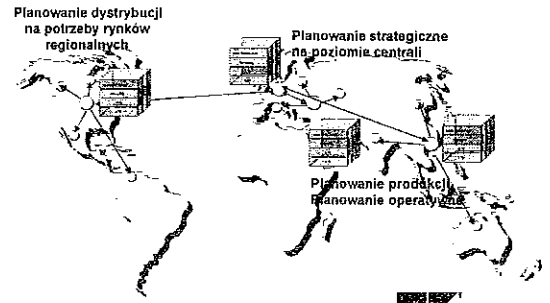

APO - Zaawansowane Planowanie i Harmonogramowanie
Konfekcjonowanie, wysyłka i transport
Planowanie produkcji
Sterowanie produkcją
Grafika
Symulacje
Algorytmy dostosowane do specyfiki branży imetody heurystyczne

APO - sprawdzanie dostępności zapasu

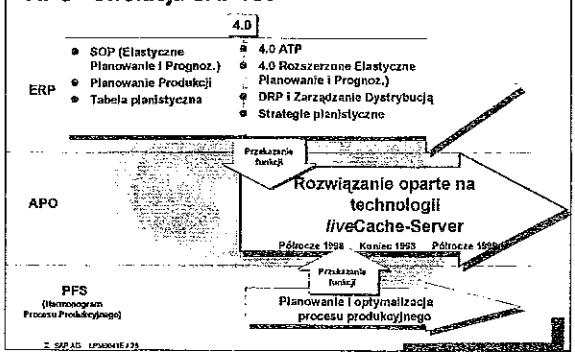


- Wysoka wydajność
- Globalny serwer ATP
- Wielostopniowa procedura
- Strategie definiowane w oparciu o reguły zarządzania zapasem

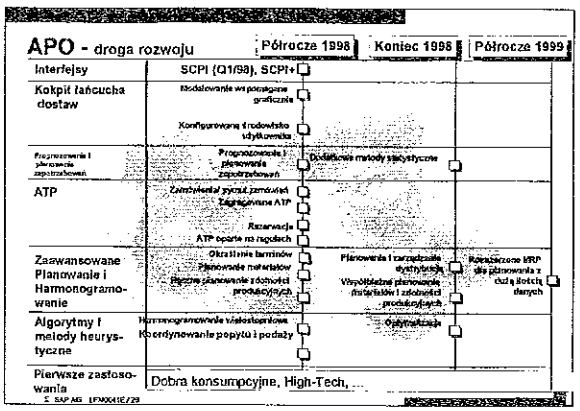
Przykładowe zastosowanie



APO - ewolucja SAP R/3



APO - droga rozwoju



Podsumowanie

- Trendy w obsłudze klientów i rozwoju produktów
- Wymagania dla dostawców komponentów i surowców
- Integracja procesów zamawiania i zaopatrywania
- Przepływ informacji gospodarczej pomiędzy komputerowymi systemami partnerów gospodarczych
- SAP SCOPE

Przewidywane efekty

- Zwiększenie integracji informacji poziomu planowania i wykonywania planów w czasie rzeczywistym
- Otwartość na integrację systemów różnych producentów oprogramowania
- Otwartość na optymalizację współpracy partnerów w łańcuchu logistycznym
- Rozwój metod wspomagania procesu wdrożeniowego (modele branżowe, Business Engineer)
- Standaryzacja - jeden konfigurowalny komponent dla całego obszaru logistyki

Daniel Słobodzian

e-mail: daniels@man.poznan.pl

Zarządzanie transakcjami w architekturach klient-serwer obiektowych baz danych

Celem pracy było przeprowadzenie analizy wydajności czterech algorytmów zarządzania transakcjami i spójnością danych w rozproszonej pamięci cache, w systemie obiektowej bazy danych o architekturze klient-serwer. W specjalnie zaprojektowanym i skonstruowanym symulatorze o nazwie CSDBSim, pracującym w systemie Windows NT, zbadano dwa algorytmy optymistyczne: Optimistic 2-Phase Locking i Optimistic with Loosely Synchronized Clock, algorytm pesymistyczny Callback Locking oraz wielowersyjny algorytm Transient Versioning. Wątki systemowe reprezentowały rozproszone stacje klienckie, natomiast oferowany przez Windows NT system komunikacji pomiędzy wątkami modelował sieć komputerową. Algorytmy zostały tak dobrane, by wszystkie cechowały się pewną odmiennością od pozostałych, ale w podobny sposób wykorzystywały bufory cache, utrzymując w nich dane przez czas trwania wielu transakcji. Różnią się przede wszystkim sposobem uaktualniania replik stron, mechanizmami weryfikacji konfliktów i sposobem unieważniania zdalnych kopii. Zamodelowano system o stronicowej ziarnistości danych (page-server), przy metodzie komunikacji typu data-shipping, w uproszczonej wersji z jednym serwerem.

Dla każdego z algorytmów przeprowadzono szereg przebiegów, przy różnych modelach rozkładu dostępu do danych. Każdy z modeli oferował różny dobór rejonów bazy danych, do których generowane były żądania dostępu typu odczytu lub zapisu. Skupiono się na popularnych rozkładach, najczęściej spotykanych w literaturze: UNIFORM, HOTCOLD, PRIVATE, FEED oraz HICON. Za każdym razem stacje klienckie generowały losowe transakcje, komunikując się z serwerem. W sposób zależny od badanego algorytmu, serwer koordynował spójność zawartości lokalnych buforów utrzymywanych przez poszczególne stacje, doprowadzając do pomyślnego zakończenia transakcji (commit) lub restartując pewne z nich.

Eksperymenty wykazały, że w grupie czterech wybranych algorytmów, podejście optymistyczne sprawdza się najlepiej dla mniejszej liczby stacji klientów. Dla większego obciążenia systemu algorytmy pesymistyczne, a w szczególności Callback Locking, zwiększają swoją relatywną wydajność względem algorytmów optymistycznych. Dla tych samych modeli rozkładu dostępu do danych algorytmy zachowują podobne tendencje, różniąc się przede wszystkim wartościami.

Implementacja symulatora jako wielowątkowej aplikacji pracującej w systemie Windows NT jest podejściem odmiennym od najczęściej spotykanych rozwiązań, które jako środowisko pracy wykorzystują systemy operacyjne oparte na systemie UNIX. Praca miała również na celu sprawdzenie systemu Windows NT, obiektowych cech języka C++ oraz klas biblioteki Microsoft Class Library, pod kątem zastosowania w dziedzinie badań symulacyjnych. W efekcie okazało się, że środki te mogą służyć do prowadzenia symulacji, przy uwzględnieniu pewnych ograniczeń. Podstawową wadą jest tu wykorzystanie systemu komunikatów przesyłanych między oknami systemu operacyjnego jako systemu komunikacji pomiędzy obiektami symulowanego

modelu. Część komunikatów obsługiwała interfejs graficzny, wprowadzając dodatkowe opóźnienia. W praktyce okazało się również, że wyniki symulacji są znacząco zaburzone, gdy w systemie, w trakcie trwania symulacji, działają inne aplikacje użytkowników. Wynika to nie tylko z większej liczby procesów współdzielących czas procesora, ale także z potencjalnie różnych priorytetów nadanych procesom.

Marian Kuraś[1], Agnieszka Zając[2]

Katedra Informatyki Akademii Ekonomicznej w Krakowie
e-mail: [1]eikuras@cyf-kr.edu.pl, 2eizajac@cyf-kr.edu.pl

Powodzenie i zagrożenia zamierzeń informatycznych

Streszczenie: W referacie omówiono zagrożenia i ryzyko związane z wprowadzaniem techniki informacyjnej do wspomagania zarządzania. Celem referatu jest wskazanie zagrożeń oraz możliwości zapobiegania, eliminowania i zmniejszania ryzyka. Rozważania oparto na najnowszej literaturze oraz własnych badaniach. Omówiono sposób rozumienia powodzenia zamierzenia, systematykę zagrożeń i ryzyka oraz przedyskutowano potrzebę i możliwości sterowania ryzykiem zamierzeń. Skrótowo przedstawiono propozycję metodyki analizy i sterowania ryzykiem.

Wstęp

Każde złożone zamierzenie niesie ze sobą zagrożenia, których nie można przewidzieć ani w pełni zapobiegać im. Zamierzenia informatyczne – czego dowodzi praktyka – bardzo często kończą się niepowodzeniem (niepełnym powodzeniem), gdyż w ich przygotowaniu zwykle nie identyfikuje się w systematyczny sposób zagrożeń ani nie analizuje się ryzyka.

Problematyce zagrożeń i wynikającego z nich ryzyka poświęca się wiele uwagi w światowej literaturze z zakresu zarządzania oraz TI/SI (np.: Cash i in.: 1991; Lyytinen: 1988; Martin: 1992; Neumann: 1995; Sage: 1995). Nieporównanie mniej miejsca poświęca się tym kwestiom w piśmiennictwie krajowym (np.: Kuraś, Zając: 1996; 1997; Połap: 1995).

Referat jest adresowany do menedżerów, osób odpowiedzialnych za rozwój zastosowań TI w organizacjach oraz do twórców SI. Jego celem jest wskazanie zagrożeń i związanego z nimi ryzyka oraz wskazanie możliwości jego zmniejszania. Referat jest skróconym omówieniem wstępnych badań i zawiera liczne odwołania do innych publikacji.

1. Powodzenie i ryzyko zamierzeń

Wprowadzanie techniki informacyjnej (TI) do organizacji może i powinno być, choć nie zawsze jest zmianą na lepsze. TI jest narzędziem nowoczesnym (skutecznym, sprawnym, wydajnym i efektywnym) ale jedynie pod warunkiem jej celowego zastosowania. Cele powinny wynikać z celów działalności i strategii firmy. Innymi słowy, najnowsza TI zastosowana do wspomagania niezmienionej źle funkcjonującej organizacji nie przyniesie poprawy sytuacji. Cele zastosowań TI powinny jednoznacznie wskazywać oczekiwane korzyści, które zdefiniowane jako kryteria oceny, powinno się formułować w kategoriach operacyjnych i finansowych.

Głównym celem wszelkich działań podejmowanych w organizacji jest (powinien być) jej sukces[1]. Zależy on od licznych czynników, przy czym główne czynniki są związane z zarządzaniem. Przygotowanie projektu powinno uwzględniać sformułowanie krytycznych czynników powodzenia zmiany organizacyjnej

[1] Pod tym sformułowaniem należy rozumieć trafne określenie celów strategicznych i ich osiąganie. Za sukces organizacji można uznać np. poprawienie pozycji na rynku, zwiększenie udziału w rynku, wejście na (nowy) rynek czy wprowadzenie nowych produktów.

oraz wprowadzania TI (Bullen: 1996; Rockart: 1979, 1982). Za jeden z głównych czynników powodzenia trzeba przyjąć zrozumienie roli i znaczenia TI dostarczającej metod i narzędzi umożliwiających dokonanie potrzebnych zmian organizacyjnych.

Drugą stroną medalu są zagrożenia i ryzyko związane z wprowadzaniem TI. W tablicy 1 przedstawiono definicje ryzyka spotykane w literaturze. Źródła zagrożeń oraz wynikającego z nich ryzyka są bardzo liczne i w ograniczonym stopniu rozpoznane, zinterpretowane, rozumiane i uznawane. Większość zagrożeń ma typowy charakter, ale ujawniają się one i oddziaływają w sposób specyficzny, zależny od warunków, w jakich powstaje i wdraża się SI. Z tej przyczyny konieczne jest rozpoznawanie, interpretowanie, analiza, zrozumienie, akceptowanie oraz przeciwdziałanie zagrożeniom i zmniejszanie ryzyka. W pracach muszą uczestniczyć wszyscy kluczowi uczestnicy zamierzenia: menadżerowie, użytkownicy i twórcy SI. Ich świadomość zagrożeń oraz wiedza i motywacja są koniecznymi czynnikami eliminowania / zapobiegania / zmniejszania ryzyka całego zamierzenia.

Liczność i rozmaitość zagrożeń wymaga ich systematyzacji oraz metodyki analizy. W pracach (Kuraś, Zając: 1996a, b, c; 1997a, b. Kuraś i in.: 1996) przedstawiono propozycje dotyczące systematyki zagrożeń i źródeł ryzyka oraz ich rozpoznawania i analizy. Zgodnie z przyjętymi założeniami systematyki (rysunek 2) stworzono listę zagrożeń elementarnych (generowanych przez każdy z pięciu zidentyfikowanych obszarów[2]) i kompleksowych (powstających w wyniku złożenia oddziaływań zagrożeń elementarnych). Lista ta zawiera obecnie blisko trzysta zagrożeń i czynników ryzyka, jednak nie jest ona jeszcze kompletna. Uzupełnia się ją w miarę dalszych obserwacji i analiz.

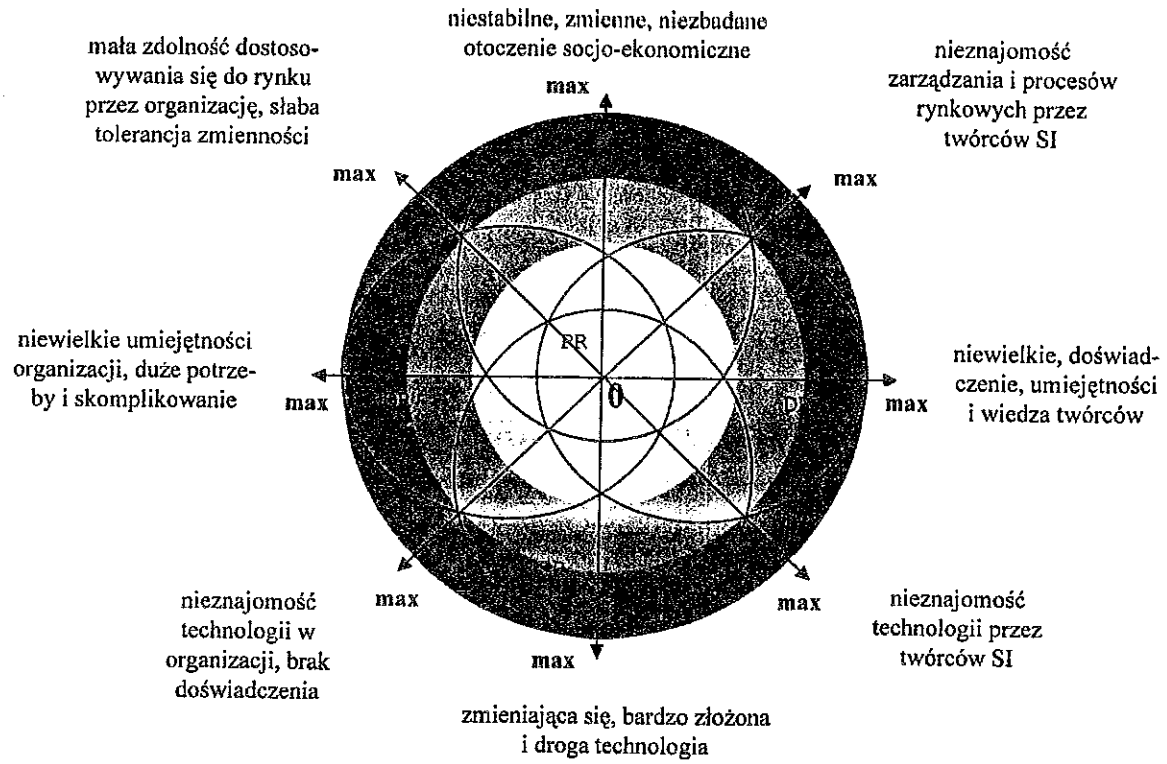
Tablica 1

Definicje ryzyka

Ryzyko definiuje się jako potencjalne źródło istotnego błędu, kosztu lub odchylenia od oczekiwań -- niepożądaną konsekwencję niepewności, nieokreśloności *Cooper, Chapman: 1987*
Ryzyko oznacza niepomyślne konsekwencje, zwłaszcza, zwłaszcza gdy dotyczy przedsięwzięć i działań społecznych (Jennergran, Keeney: 1985)
Ryzyko oznacza zagrożenie (narażenie) negatywnymi skutkami przyszłych zdarzeń. *(Management: 1994)*
Ryzyko jest szkodą, jaką może przynieść urzeczywistnienie się zagrożenia.
Ryzyko jest miarą wielkości szkód rozmiarów zagrożenia jak np. iloczyn prawdopodobieństwa i wielkości skutków. *(Neumann: 1995)*
Ryzyko i niepewność charakteryzuje sytuacje, w których ostateczny skutek określonego zdarzenia lub działania może odchylać się od wielkości prognozowanych, szacowanych. *(Raftery: 1994)*
Ryzyko oznacza prawdopodobieństwo lub możliwość straty bądź szkody w specyficznych warunkach otoczenia i w określonym czasie. *(Sage: 1995)*

Dla wszelkich projektów trudno jest wskazać główne zagrożenia, gdyż mają one charakter sytuacyjny. Oznacza to, że występują one i oddziaływają w określonych warunkach, w jakich przeprowadza się zamierzenie. Trzeba zdawać sobie sprawę z istnienia pewnych klas zagrożeń o zbliżonym i współzależnym charakterze oraz szczególnym znaczeniu (rysunek 1). Główne zagrożenia w większości naszych organizacji wynikają z cech otoczenia socjo-ekonomicznego oraz organizacji. Niedostatki systemu edukacji, niechęć do zmian, archaiczne pojmowanie zarządzania, niski poziom tolerowania ryzyka, niestabilność systemu prawnego i gospodarczego, to tylko niektóre zagrożenia wpływające na podejmowane zamierzenia informatyczne.

[2] Za podstawowe obszary ryzyka uznano otoczenie społeczno-gospodarcze i techniczne, czynniki organizacyjne, twórców SI oraz sam projekt zamierzenia (Kuraś i in.: 1996).



Źródło: (Kuraś, Zając: 1997a)

Rys. 1. Złożone klasy źródeł i czynników ryzyka

2. Konieczność podejmowania ryzyka

Zważywszy na nieuniknioność zagrożeń i niemożność pełnego rozpoznania ryzyka, można byłoby sformułować tezę, że nie powinno się podejmować żadnych zamierzeń. Jest to całkowicie błędny wniosek. Ryzyko należy podejmować, jeśli chce się utrzymać na rynku i poprawić pozycję. Jeśli firma nie podejmie ryzykownego zamierzenia służącego strategicznej zmianie organizacyjnej, zrobią to konkurenci. W tablicy 2 przedstawiono zasady podejmowania zamierzeń. Powinni nimi kierować się wszyscy uczestnicy a w szczególności menedżerowie, którzy podejmują decyzje o zamierzeniu. Wszyscy uczestnicy zamierzenia muszą być zarazem świadomi ryzyka, co oznacza, że muszą uczestniczyć w jego analizie oraz przeciwdziałaniu zagrożeniom.

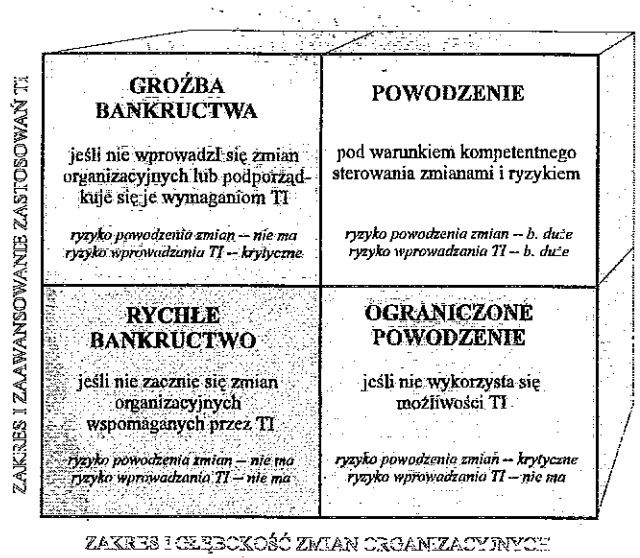
Na rysunkach 2 i 3 przedstawiono zależności między ryzykiem zmian organizacyjnych i technologicznych a sytuacją firmy. Z rysunku 2 wynika, że powodzenie zamierzenia w zasadniczym stopniu zależy od kompleksowości zmiany. Jest wiadomo, że złożoność, szeroki zakres zmiany jest czynnikiem znaczenie zwiększającym ryzyko. Organizacja musi zatem ocenić swoją sytuację strategiczną i dokonać wyboru wariantu działania określając zakres zmiany oraz jej potencjalne skutki (wpływ na sytuację rynkową). Kryteriami wyboru będą możliwości finansowe i kompetencyjne, koszty oraz wysokość ryzyka zamierzenia. Należy zatem oszacować poziom ryzyka oraz podjąć próby jego obniżenia do poziomu akceptowalnego -- dającego widoki na adekwatne korzyści oraz gwarancje panowania nad zagrożeniami.

Tablica 2

Zasady podejmowania zamierzeń informatycznych

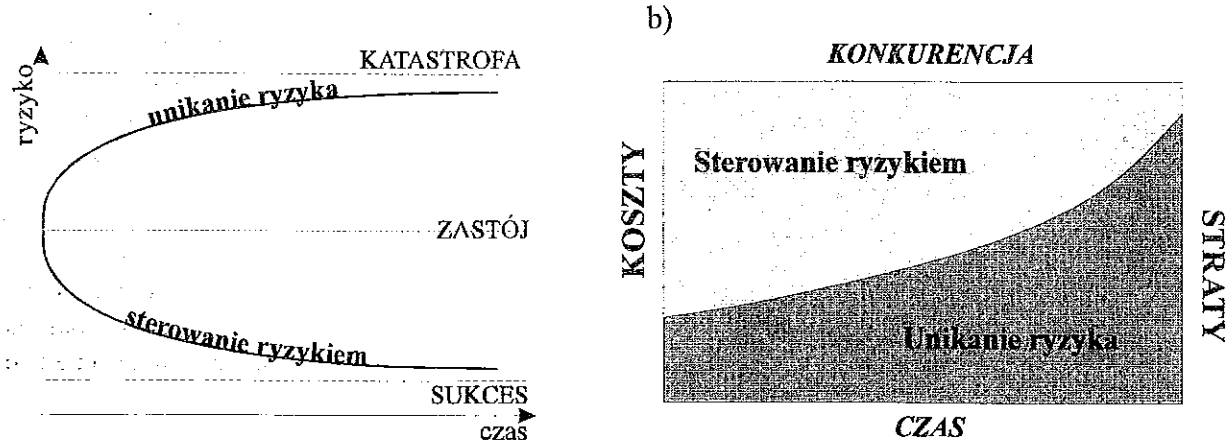
1. Każde zamierzenie niesie ryzyko.
2. Zamierzenia potencjalnie bardzo korzystne wiążą się z wysokim ryzykiem
3. Ryzyka nie można bezkarnie nie dostrzegać, lekceważyć czy też uniknąć.
4. Problemów nie rozwiążą informatycy ani technika informacyjna.
5. Zastosowania TI muszą służyć określonym celom i odpowiadać potrzebom organizacji.
6. Nie opłaca się nie podejmować ryzyka.
7. Od podjęcia i przeprowadzenia ryzykownego zamierzenia może zależeć przyszłość firmy.
8. Warunkiem powodzenia związanego z ryzykiem zamierzenia jest jego rozpoznanie, uświadomienie i zaakceptowanie ryzyka, jego analiza i przyjęcie stosownych sposobów działania czyli sterowanie ryzykiem.

Źródło: opracowanie własne



Źródło: (Kuraś, Zając: 1997e)

Rys. 2. Strategiczne ryzyko zmian organizacyjnych i technologicznych



Źródło: (Kuraś, Zając: 1997e)

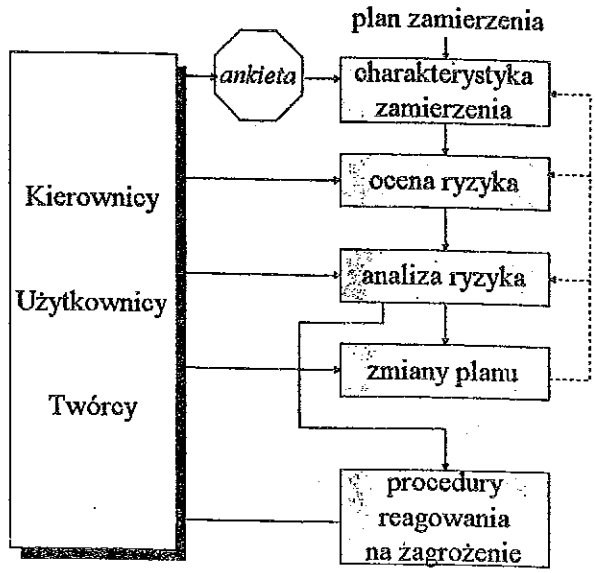
Rys. 3. Wpływ podejścia do ryzyka na powodzenie zamierzenia (a) i na pozycję rynkową (b)

3. Analiza i sterowanie ryzykiem

Do ustalenia dopuszczalnego poziomu ryzyka potrzeba identyfikacji i analizy zagrożeń oraz oceny prawdopodobieństw i skutków ich wystąpienia. *Identyfikacja ryzyka* opiera się na obserwacji i formułowaniu czynników ryzyka, związków między nimi, postrzegania ich i odczuwania zmian powodowanych przez zmiany parametrów ryzyka. *Analizę ryzyka* definiuje się jako cały proces identyfikacji i kwantyfikacji ryzyka oraz oceny jego dopuszczalności (Management: 1994). *Oceny ryzyka* dokonuje się indywidualnie oraz w zespołach i odnosi się do indywidualnych i systemowych wskaźników ryzyka. *Wskaźnik ryzyka* jest miarą określającą narażenie na prawdopodobne negatywne konsekwencje określonego zagrożenia, ich klasy lub wszystkich zagrożeń. W analizie określa się zagrożenia oraz ich powiązania i źródła. *Źródłem zagrożeń (ryzyka)* określa się ogół warunków przyczyniających się (bezpośrednio lub pośrednio) do ich powstania i ewentualnego zaistnienia. W sekcji 1 przedstawiono typologię źródeł zagrożeń (ryzyka). *Czynnikiem ryzyka* określa się prawdopodobieństwo zaistnienia zagrożenia w specyficznych warunkach zamierzenia oraz skalę konsekwencji jego zaistnienia dla organizacji. *Czynnikiem ryzyka* jest zatem nie samo zagrożenie a oszacowania prawdopodobieństwa i wielkości strat będących konsekwencją jego wystąpienia.

Sama analiza ryzyka jest niewystarczająca – konieczne jest sformułowanie planów przeciwdziałania ryzyku wynikających z analizy. *Sterowanie ryzykiem* wiąże się z podejmowaniem i wykonywaniem decyzji dotyczących przyszłych zdarzeń (sytuacji), które mogą powodować niekorzystne, niepożądane efekty oraz z tworzeniem planu strategicznego, działań taktycznych i operacyjnych zapewniających radzenie sobie w sytuacjach zagrożeń, unikanie/eliminowanie niepotrzebnego ryzyka i utrzymywanie ryzyka na akceptowalnym poziomie. Wyróżnia się cztery podejścia do sterowania ryzykiem (por.: Sage: 1995, s. 137)):

- *inaktywne* polegające na nieuznawaniu, niedostrzeganiu, lekceważeniu ryzyka;
- *reaktywne* polegające na reagowaniu na zagrożenia po ich wystąpieniu i podejmowaniu działań mających eliminować skutki;
- *interaktywne*, które zakłada branie pod uwagę zagrożeń i ich ewentualnych skutków we wszystkich fazach przygotowania i wprowadzania TI;
- *proaktywne*, wymagające przewidywania zagrożeń i ich skutków oraz stworzenia rozwiązań sterowania systemami zapewniających uwzględnienie potencjalnego ryzyka na wszystkich poziomach organizacyjnych (marketingu i planowania, badań, tworzenia, testowania i oceny oraz wdrażania i eksploatacji).



Źródło: (Kuraś i in.: 1996)

Rys. 4. Procedura wstępnej analizy i oceny ryzyka zamierzenia

4. Metoda sterowania ryzykiem RAM

Literatura zaleca całościowe, spójne podejścia do ryzyka w ramach systemu sterowania systemami (np.: Cooper, Chapman: 1987; Neumann: 1995; Sage: 1995). System taki powinien zapewniać metody, zasoby i sposoby działania (decyzje i procedury postępowania) umożliwiające rozpoznanie, analizę i interpretację ryzyka we wszystkich fazach przygotowania i tworzenia SI. Stworzenie pełnej metodyki sterowania ryzykiem wymaga badań pozwalających ustalić listę zagrożeń, prawdopodobieństw zaistnienia jak też ich potencjalnych skutków. Powinna ona zawierać zestaw technik i narzędzi umożliwiających planowanie i przeciwdziałanie ryzyku przy trzech podejściach: reaktywnym, interaktywnym i proaktywnym (Sage: 1995). Planowanie obejmuje rozpoznawanie, analizę i interpretację ryzyka, oraz stworzenie programu sterowania ryzykiem. Obejmuje on założenia, zasady, decyzje i procedury mające na celu wykrywanie zagrożeń, diagnozowanie i korygowanie działań we wszystkich fazach tworzenia SI.

Metodyka RAM (Risk Analysis & Management)[3] uwzględnia obydwie fazy sterowania ryzykiem. Ponadto wyodrębniono wstępną fazę, poprzedzającą planowanie (rysunek 4 i 5). Cele metodyki RAM można ująć następująco:

❶ eliminowanie oraz unikanie zagrożeń i ich przyczyn;
❷ zmniejszanie ryzyka, gdy nie można wyeliminować źródeł zagrożeń;
❸ eliminowanie lub ograniczanie negatywnych skutków zaistniałych zagrożeń;
❹ określenie sposobów działania w razie wystąpienia negatywnych skutków zamierzenia, których nie można wyeliminować bądź zmniejszyć.

Metodyka zawiera trzy moduły, z których pierwszy ma na celu wstępne przygotowanie całego zamierzenia oraz jego uczestników, a pozostałe służą systematycznemu sterowaniu ryzykiem:

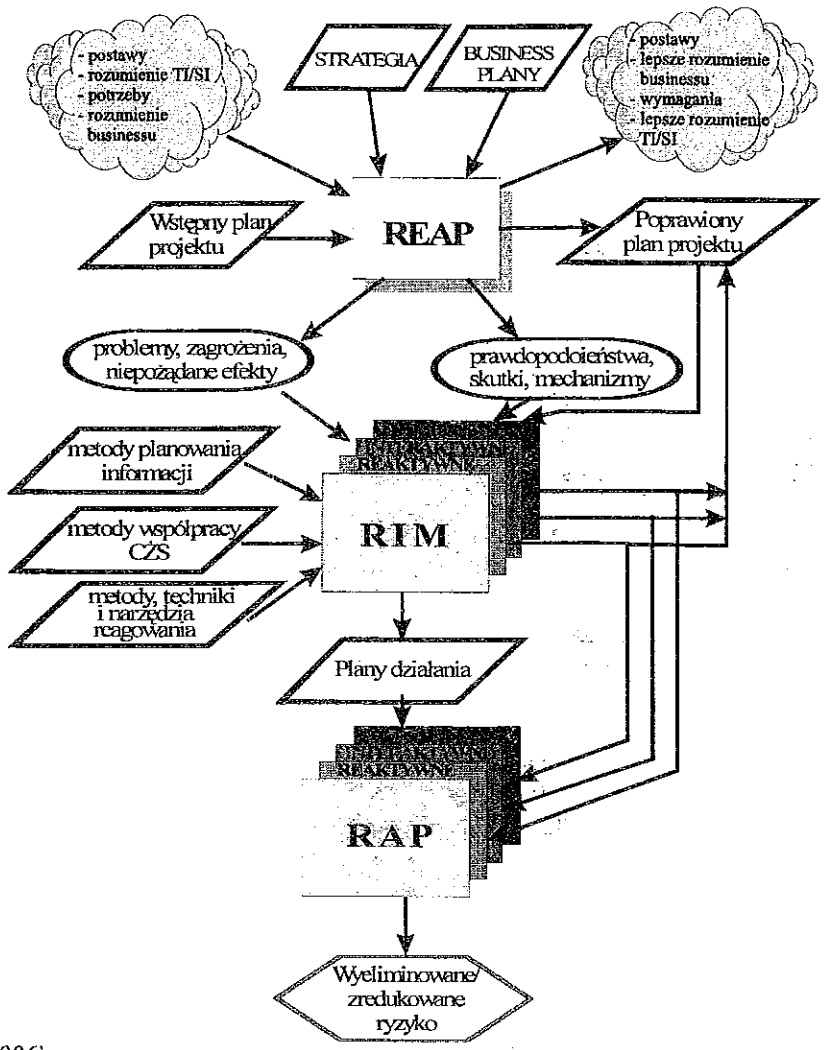
❶ REAP (*Risk Evaluation & Analysis Procedure*) -- procedura wstępnej oceny i analizy ryzyka mająca zapewnić rozpoznanie głównych zagrożeń, uświadomienie związanego z nimi ryzyka, uczenie się użytkowników i analityków oraz modyfikację planu zamierzenia;
❷ RIM (*RIsk Mapping*) -- planowanie działań zapobiegawczych i eliminujących ryzyko przy każdym z trzech podejść: proaktywnym, interaktywnym i reaktywnym, obejmujące szacowanie skutków i prawdopodobieństw zaistnienia zagrożenia, określanie sposobów przeciwdziałania oraz projektowanie reguł podejmowania decyzji i procedur reagowania na ujawniające się zagrożenia;
❸ RAP (*Risk Abatement Performance*) -- wykrywanie, rozpoznawanie, diagnozowanie i redukowanie ryzyka zgodnie z planami sterowania ryzykiem na wymienionych poziomach.

Metodyka w obecnej fazie jest koncepcją, z której opracowano i zastosowano moduł REAP, zaś na pozostałymi trwają prace. Procedura wstępnej oceny i analizy ryzyka -- REAP -- okazuje się przydatna w warunkach większości polskich organizacji. Dostarcza ona sposobów (metod, technik narzędzi i procedur) postępowania w fazie przygotowania zamierzenia. Jest szczególnie przydatna, gdy nie ma planu zamierzenia lub gdy wstępny plan nie uwzględnia zagrożeń i ich skutków. Celem procedury jest raczej uczenie się i poszukiwanie niż uzyskanie prostej odpowiedzi na pytanie: *co robić?*

Wnioski

Wszelkie zamierzenia mające na celu dostosowanie organizacji do warunków otoczenia wiążą się z ryzykiem, którego nie można nie dostrzegać, uniknąć ani lekceważyć. Ryzyko musi się podejmować, gdyż tylko wtedy uzyskuje się przewagę konkurencyjną. Warunkiem koniecznym powodzenia jest pełne rozeznanie zagrożeń, ich zbadanie, stworzenie planu przeciwdziałania i jego wprowadzenie czyli kompleksowe sterowanie ryzykiem.

[3] Założenia metodyki RAM przedstawiono w pracach wymienionych w bibliografii.



Źródło: *(Kuraś i in.: 1996)*

Rys. 5. RAM -- metoda analizy i sterowania ryzykiem

Rozpoznanie zagrożeń oraz analiza ryzyka ma zapewnić, że decyzje dotyczące wprowadzania TI będą podejmowane świadomie z uwzględnieniem rozpoznanych zagrożeń i znajomością sposobów przeciwdziałania ryzyku. Postępowanie proponowane w procedurze REAP metodyki RAM ma ułatwić organizacji uczenie się przeciwdziałania ryzyku zamierzenia. Nacisk kładzie się na zakres analizy i procedurę postępowania a pomija techniki i narzędzia pomiaru ryzyka. Są one zagadnieniem pomocniczym a sam sposób liczenia ryzyka nie może mieć wpływu na jego znajomość ani tym bardziej na zapobieganie.

Analiza ryzyka jest niezbędnym sposobem zapewnienia powodzenia projektu SI, który ma służyć przeprowadzeniu zmiany organizacyjnej. Niepowodzenie strategicznych zmian organizacyjnych, spowodowane nieudanym -- w konsekwencji zaniechania analizy ryzyka -- wdrożeniem TI, mogłoby się stać przyczyną poważnych zakłóceń w działalności firmy.

Bibliografia:

Bullen, C.V. (1996). Reexamining Critical Success Factors [in:] Umbaugh (1996).

Cash, J.I.Jr., McFarlan, F.W., McKenney, J.L., Applegate, L.M. (1991). Corporate Information Systems Management. Text and Cases. 3rd Ed. Homewood, Ill.: Irwin.

Cooper, D., Chapman, C. (1987). Risk Analysis for Large Projects. Chichester: J.Wiley & Sons.

Kuraś, M., (1995). Ryzyko wprowadzania techniki informacyjnej.. [w:] H.Sroka, S.Stanek (red.). Inteligentne systemy wspomagania decyzji w zarządzaniu. Nowe kierunki prac. Katowice A. E.

Kuraś, M., Zając, A. (1996a). Eliminowanie ryzyka zamierzeń informatycznych.. Mat. Konf. INFOGRYF'96. Tom I. Szczecin-Kolobrzeg: TNOiK, Oddz. w Szczecinie (ss.: (128-135).

Kuraś, M., Zając, A. (1996b). Podglądanie ryzyka.. ComputerWorld, 1996/10/21

Kuraś, M., Zając, A. (1996c). Sterowanie ryzykiem przedsięwzięć informatycznych.. Wielodostępne Systemy Informatyczne do zarządzania przedsiębiorstwem Mat.II Konf. Komputerowe Systemy Wielodostępne. Bydgoszcz-Ciechocinek: ATR - BELMA S.A. - BZP

Kuraś, M., Zając, A. (1996d). Zagrożenia dla zamierzeń informatycznych. [w:] Mat. II Konf. Informatyka na wyższych uczelniach dla gospodarki narodowej. Tom I. Gdańsk: Politechnika Gdańska.

Kuraś, M., Zając, A. (1996e). Sterowanie ryzykiem zamierzeń informatycznych. [w:] Mat.konf. Wielodostępne systemy informatyczne do zarządzania przedsiębiorstwem. AT-R Bydgoszcz. Ciechocinek.

Kuraś, M., Zając, A. (1996f). Czynniki powodzenia i ryzyka projektów informatycznych. Zeszyty Naukowe Akademii Ekonomicznej w Krakowie (w druku)

Kuraś, M., Zając, A. (1997a). Czynniki ryzyka tworzenia i wykorzystania systemów wspomagania decyzji. Sroka, H., Stanek, S. (red.) Inteligentne Systemy Wspomagania Decyzji. Rozwój Badań. Katowice: Akademia Ekonomiczna

Kuraś, M., Zając, A. (1997b). Information Systems Pitfalls: How to Reduce IT Risks. Strategic Management and Its Support by Information Systems. International Scientific Conference. Ostrava: VSB - Technical University of Ostrava. Faculty of Economics.

Kuraś, M., Zając, A. (1997c). Istota ryzyka zamierzeń informatycznych. [w:] B. F. Kubiak (ed.), A.Korowicki (ed.) Human Computer Interaction '97. Gdańsk: UG

Kuraś, M., Zając, A. (1997d). Narzędzia i techniki analizy ryzyka projektów SI. Restrukturyzacja systemów informacyjnych przedsiębiorstwa. Mat. III Konferencji Komputerowe Systemy Wielodostępne. Bydgoszcz-Ciechocinek: ATR - BELMA S.A. - BZP

Kuraś, M., Zając, A. (1997e). Ryzykować? Nie ryzykować?. ComputerWorld nr 33/301(1997/09/15)

Kuraś, M., Zając, A., (1997f). Wpływ sterowania ryzykiem na powodzenie zamierzeń informatycznych. Informatyka Stosowana, VI Konf. Klubu Użytkowników Systemu Progress, Kazimierz Dolny.

Kuraś, M., Zając, A., (1997g). IS Project Risk at Polish Organizations. [in:] H.Krallman (ed.). Wirtschaftsinformatik'97. Internationale Geschäftstätigkeit auf der Basis flexibler Organisationsstrukturen and leistungsfähiger Informationssysteme. Heidelberg: Physica-Verlag.

Kuraś, M. Zając, A. (1997h). Powodzenie i zagrożenia dla zamierzeń informatycznych. Zeszyty Naukowe Akademii Ekonomicznej w Krakowie (w druku).

Kuraś, M., Zając, A., (1998). Zagrożenia dla zamierzeń informatycznych. Informatyka, Vol. 34, Nr 3.

Kuraś, M., Zając, A., Stefański, M. (1996). Strategiczne planowanie rozwoju systemu informacyjnego. Analiza ryzyka zamierzeń informatycznych. Raport z pracy badawczej 85/KI/6/95-S. Kraków: A. E.

Neumann, P.G. (1995). Computer Related Risks. New York: ACM Press.

Połap, J. (1995). Zarządzanie ryzykiem systemów informatycznych. Rachunkowość. No.4.

Rockart,, J.F. (1979). Chief Executives define Their Own Data Needs. Harvard Business Review (Sept.-Oct., 1979).

Rockart,, J.F. (1982). The Changing Role of the Information Systems Executive. Sloan Management Review (Fall, 1982).

Sage, A.P. (1995). Systems Management for Information Technology and Software Engineering. N.York-Chichester-Brisbane-Toronto-Singapore: John Wiley & Sons.

Steinmann, S., Schmitgen, S. (1997). Neue Kernprozesse für Versicherer mit Agenturnetz Ein Beispiel für IT-enabled Business Innovation [in:] H.Krallman (ed.). Wirtschaftsinformatik'97. Int. Geschäftstätigkeit auf der Basis flexibler Organisationsstrukturen and leistungsfähiger Informationssysteme. Heidelberg: Physica-Verlag.

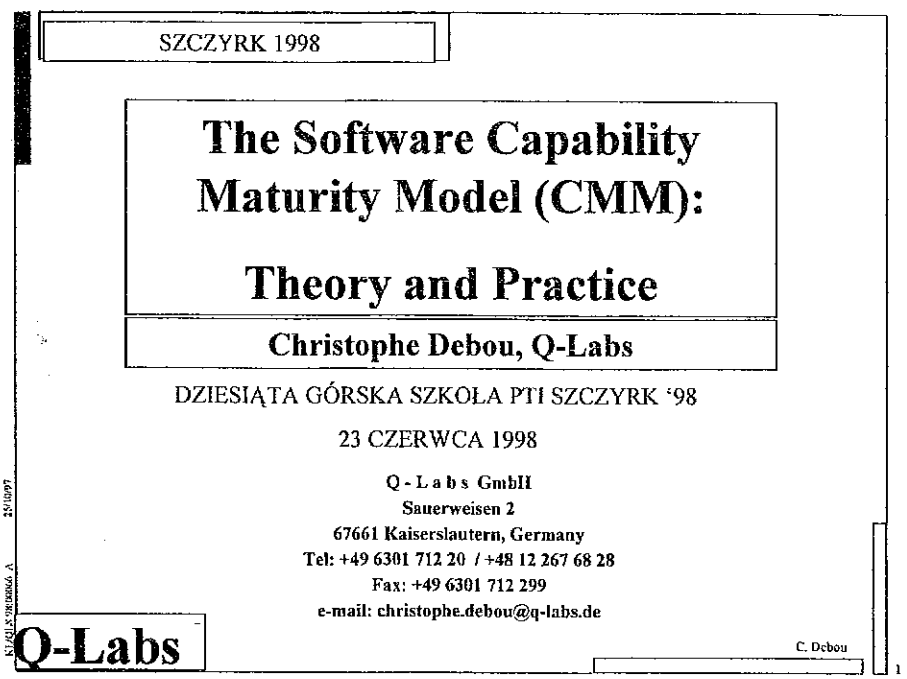
Stokalski, B. (1997). Informatyka w czasach zarazy. Zarządzanie ryzykiem w projektach informatycznych [w:] Trzynaste Jesienne Spotkania PTI 'Informatyka w organizacjach – Stan dzisiejszy i rzut oka w przyszłość. Mrągowo 24-28 listopada.

Christophe Debou

Q - L a b s GmbH
Sauerwiesen 2
67661 Kaiserslautern, Germany
Tel: +49 6301 712 20 / +48 12 267 68 28
Fax: +49 6301 712 299
e-mail: christophe.debou@q-labs.de

The Software Capability Maturity Model (CMM)

Theory and Practice



SZCZYRK 1998

The Software Capability Maturity Model (CMM): Theory and Practice

Christophe Debou, Q-Labs

DZIESIĄTA GÓRSKA SZKOŁA PTI SZCZYRK '98

23 CZERWCA 1998

Q-Labs GmbH
Sauerwiesen 2
67661 Kaiserslautern, Germany
Tel: +49 6301 712 20 / +48 12 267 68 28
Fax: +49 6301 712 299
e-mail: christophe.debou@q-labs.de

Q-Labs

SZCZYRK 1998

Agenda

- who is Q-Labs?
- CMM: why?
- CMM: What?
- CMM: How?
- CMM: How much?

Q-Labs

SZCZYRK 1998

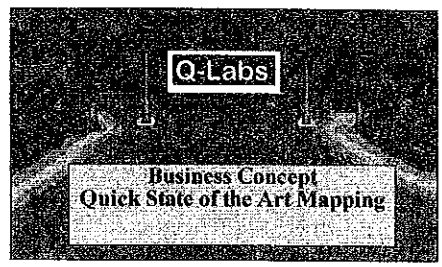
Q-Labs: Some Data

- Located in Sweden (Lund, Linkoping, Gothenburg), Germany (Kaiserslautern), Maryland (US), Polska ("One consultant shop"), ...
- 50 Consultants
- Support both organisations
  - developing software intensive systems e.g. ABB, Alcatel, Eriesson, Bosch, Siemens, Volvo, Daimer Benz,
  - Purchasing software systems e.g. Telia, FMV, Norwegian Ministry of Justice, Telenor

Q-Labs

SZCZYRK 1998

Q-Labs Concepts



Q-Labs

SZCZYRK 1998

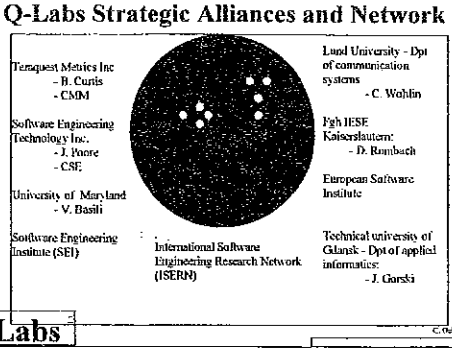
Q-Labs Product and Services

- **Software Process Improvement**: CMM, SW Metrics, Project management, Reviews/inspections, requirements engineering
- **Cleanroom Software Engineering**: Incremental development, Statistical Usage Testing, Formal specifications
- **Human Factors:** team building, vision seminar, ...
- **Architecture and Reuse**
- **Software Acquisition**: supplier selection, product assessment, external quality assurance, ...

Q-Labs

SZCZYRK 1998

Q-Labs Strategic Alliances and Network



Q-Labs

SZCZYRK 1998

Presentation Objectives

By the end of this presentation you should be able to explain:

- The reasons for applying CMM
- The structure of CMM
- The areas CMM addresses in software development
- How you can use CMM for improving your organisation

Q-Labs

SZCZYRK 1998

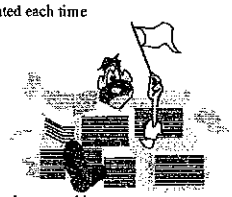
CMM: WHY?

Q-Labs

SZCZYRK 1998

Crisis-Driven Organisations

- Processes:
  - ad hoc - improvised and reinvented each time
  - abandoned under stress
- Schedules and budgets:
  - unpredictable
  - usually exceeded
- Success factors:
  - heroes, overtime, fire fighting



Blaming others: bad subcontractors, hardware problems...

Q-Labs

SZCZYRK 1998

Sources of crisis

- Inaccurate estimates
  - underscoped requirements
  - explosive code growth
  - schedule and resource shortfalls
- Unmanaged work
  - no corrective action
  - unruly subcontractors
- Volatile baselines
  - changing requirements
  - uncontrolled versions

Q-Labs

SZCZYRK 1998

Executive problems in software

- Low visibility into progress and quality
- Unpredictable performance
- Constant surprises
- Excessive costs

Q-Labs

SZCZYRK 1998

What frustrates developers

- Being given half the time they need to do the work
- Continual overtime
- Guessing what requirements means
- Contending with uncontrolled changes in requirements
- Finding they are not working with the current version
- Redoing builds forever
- Delivering products they are not proud of
- Fixing old defects without design documentation
- Constantly patching up old software that has no architecture

Q-Labs

SZCZYRK 1998

The Product

Technology

Product

People

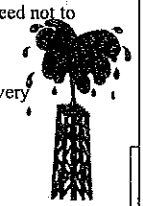
Process

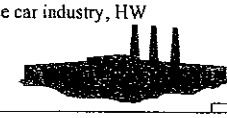
Q-Labs

SZCZYRK 1998

The Process in Industry

- Earlier the production was based on few items and skilled individuals.
- Moving on to mass production there was a need not to check every item produced.
- Control the process, not the (end) product.
- Process is a competitive weapon in almost every industry except software development e.g. Japanese car industry, HW





Q-Labs

SZCZYRK 1998

Definition of Software Process

"...Software Process can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals)."

Definition from SEI technical report SEI-93-TR-24, "Capability Maturity Model for Software, Version 1.1"

Q-Labs

SZCZYRK 1998

Misconceptions about process

- I do not need process if I have
  - really good people
  - advanced technology
  - an experienced manager
- Process interfaces with creativity
- Process = bureaucracy
- Process hinders agility in fast-moving markets
- Process is only useful on large projects

"If this is your view then you do not have the right processes"

Q-Labs

SZCZYRK 1998

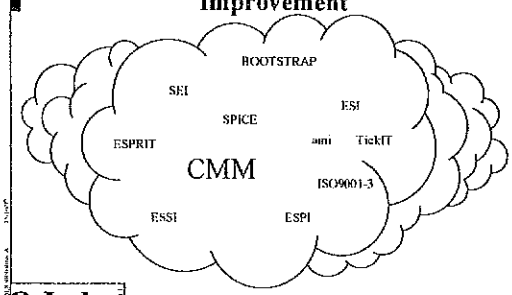
What is an effective Process?

- Defined
- Documented
- Used
- Supported by management (enforced)
- Trained
- Measured/Known in quantitative terms
- Supported by tools and technology

Q-Labs

SZCZYRK 1998

Initiatives for Software Process Improvement



Q-Labs

SZCZYRK 1998

The Start of CMM

- Millions of dollars wasted for US Department of Defence, Ada etc didn't solve all problems. Management problem, not technical!
- Watts Humphrey, then at IBM, had an idea!
- SEI and MITRE Corporation began the development late 1986
- Assessment questionnaire - September 1987
- Managing the Software Process by W. Humphrey- 1989
- CMM Version 1.0 - 1991
- Version 1.1 - 1993
- Version 2.0 - under development

Q-Labs

SZCZYRK 1998

Software Engineering Institute Overview

- Funded by US department of Defence (1987)
- Mission: to provide leadership in advancing the state of the practice of software engineering to improve the quality of systems that depend on software
- Originally established to assess DOD software suppliers (Software Capability Maturity Model)

Q-Labs

SZCZYRK 1998

CMM: What?

Q-Labs

SZCZYRK 1998

The Capability Maturity Model for Software (CMM)

"The Capability Maturity Model for Software (CMM) is a **framework** that describes the elements of an effective software process. The CMM describes an **evolutionary path** from an ad hoc, chaotic process, to a mature disciplined process"

Q-Labs

SZCZYRK 1998

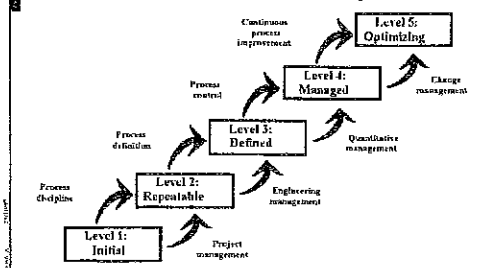
Capability Maturity Model Objectives

- CMM is TQM with a software view, is based on state-of-the-art SW Engineering practices and helps organisations:
  - Characterise the maturity of their process
  - Establish goals for process improvement
  - Set priorities for immediate actions
  - Envision culture of software excellence

Q-Labs

SZCZYRK 1998

The Capability Maturity Model



Q-Labs

SZCZYRK 1998

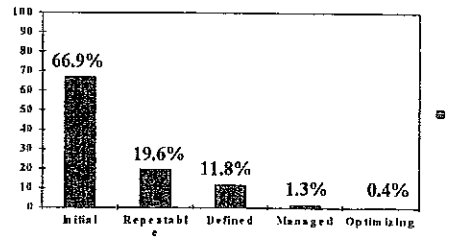
CMM architecture

Level	Focus	Key Process Areas
Optimizing (5)	Continuous process improvement	Defect prevention Technology change mgmt Process change mgmt
Managed (4)	Product and process quality	Quantitative process mgmt Software quality management
Defined (3)	Defined engineering process	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
Repeatable (2)	Project management and commitment process	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software configuration management
Initial (1)	Heroes.	

Q-Labs

SZCZYRK 1998

Organisation Maturity Level - Nov 1996



Based on most recent assessments of 542 organisations

Q-Labs

SZCZYRK 1998

Level 1 - The Initial Process

- Unstable environment lacking sound management practices - commitments not controlled
- Practitioners argue against engineering discipline under the guise of 'individual creativity'
- Success rides on individual talent and heroic effort
- Standards and practices often sacrificed to management priorities - usually schedule
- Process capability is unpredictable - schedule, cost, and quality targets are rarely met

Q-Labs

SZCZYRK 1998

Level 2 - The Repeatable Process

- Basic project management discipline is installed
- Previously successful processes are repeatable in a stable, managed environment
- Reasonable commitments are planned
- Product baselines are established and controlled
- Process capability exists for meeting schedule

Q-Labs

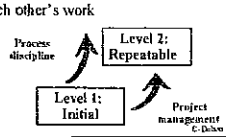
SZCZYRK 1998

Moving to Level 2

Focus — get individual projects to manage themselves

Coming from ad hoc environments:
- each project may develop its own approach
- projects leverage industry examples and standards

Coming from environments with some good practices:
- individual projects may leverage each other's work
- project approaches may be similar



Q-Labs

SZCZYRK 1998

Level 3 - the Defined Process

- Standard software processes are defined, documented, and applied throughout the organisation
- Shared understanding of how the process works and each person's role is established
- Software engineering process group created to focus on process improvement
- Process capability to meet schedule, cost, and functionality targets exists within established product lines

Q-Labs

SZCZYRK 1998

Moving to Level 3

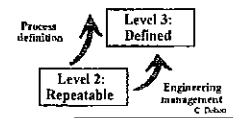
At Level 2 the focus is on projects

At level 3 the emphasis shifts to the organisation:

- Best practices are gathered from across the organisation
- Standard processes are developed for the organisation by integrating these best practices
- Processes are tailored for use on projects

Organisation process capability

- Meet schedule, cost and functionality targets within established product lines



Q-Labs

SZCZYRK 1998

Level 4 - the Managed Process

- Statistical process control principles are used to address special causes of process variation
- Detailed process and product data are available - quality targets are set
- Process is capable of performing within narrowly defined quantitative limits - targets are predictable
- Detailed measures of the software process and product quality are collected
- Both the software process and products are quantitatively understood and controlled using detailed measures

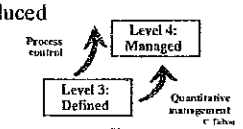
Q-Labs

SZCZYRK 1998

Moving to Level 4

- At Level 3 systematic measures have been defined and collected
- At Level 4 decisions are made based on data analyses

Projects are quantitatively managed and process and quality variation are reduced



Q-Labs

SZCZYRK 1998

Level 5 - the Optimising Process

- Chronic causes of poor performance are identified and eliminated.
- Software process is continually improved
- New technologies are prototyped, piloted, and if successful, introduced into the process
- Process capability is continuously raised
- Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies

Q-Labs

SZCZYRK 1998

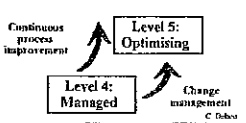
Moving to Level 5

The organisation's focus shifts from special to common causes of poor performance

- At Level 4 process and quality variation are reduced
- At Level 5 the average process and quality levels are continuously improved

Continuous improvement becomes a way of life

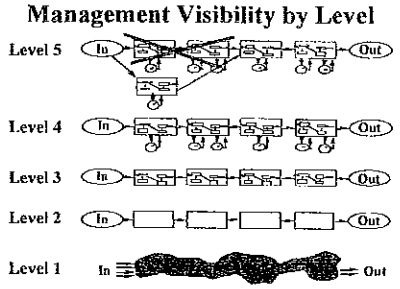
- At the lower maturity levels worker participation on continuous improvement may be on the order of 20-30%
- World-class companies have 70-80% participation in improvement programmes at any given time



Q-Labs

SZCZYRK 1998

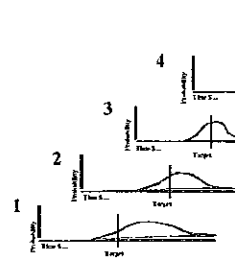
Management Visibility by Level



Q-Labs

SZCZYRK 1998

Process Capability by Level



With increasing maturity:
- accuracy increases
- variance reduces
- performance improves

Q-Labs

SZCZYRK 1998

Process Characteristics by CMM Level

Level 5	Processes improved continuously
Level 4	Processes perform within quantitative bounds
Level 3	Best processes transferred across organisation
Level 2	Processes protected within projects
Level 1	Individuals do their own thing

Q-Labs

SZCZYRK 1998

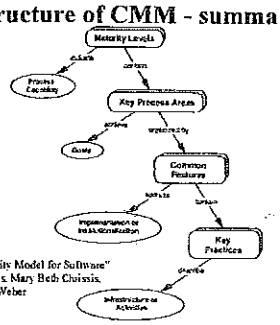
Cultural Changes by CMM Level

Level 5	Culture of empowerment and innovation
Level 4	Culture of performance excellence
Level 3	Culture of engineering
Level 2	Culture of commitment
Level 1	Imported culture

Q-Labs

SZCZYRK 1998

The structure of CMM - summary



From: "The Capability Maturity Model for Software" by Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber

Q-Labs

SZCZYRK 1998

Key Process Area (KPA)

Key process area:
- Important component of behaviour at a maturity level
- Collects a number of related activities
- The activities shall be performed collectively to gain the benefits or avoid the problems
- The Key Process Areas are initiated at one of the maturity levels
- the lower Key Process Areas will evolve as we reach higher maturity levels.

There are 18 Key Process Areas in the CMM

Q-Labs

SZCZYRK 1998

Key Process Area - example

Software Project Planning (a level 2 KPA):

"The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project."

Q-Labs

SZCZYRK 1998

Key Process Area Goals

Key process area goals:

- Conditions that enhance the capability of the processes related to the KPA
- Requirements for the maturity level

Are used to guide:

- Process assessments
- Alternative practices to fulfil a KPA

The most important description of a KPA!

Q-Labs

SZCZYRK 1998

Key Process Area Goals - example

Software Project Planning goals

"Goal 1- Software estimates are documented for use in planning and tracking the software project"

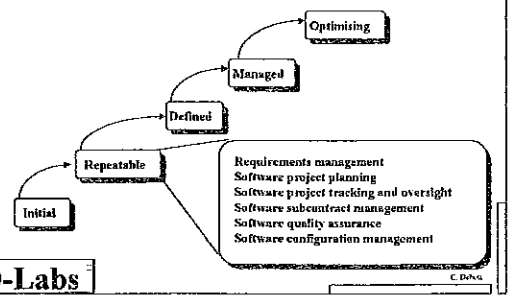
"Goal 2- Software project activities and commitment are planned and documented"

"Goal 3 - Affected groups and individuals agree to their commitments related to the SW project"

Q-Labs

SZCZYRK 1998

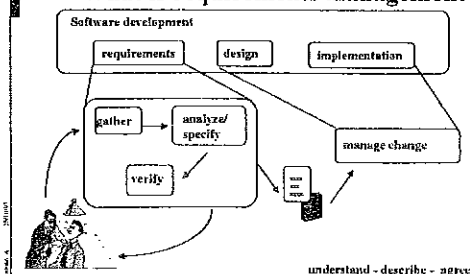
The Repeatable Level



Q-Labs

SZCZYRK 1998

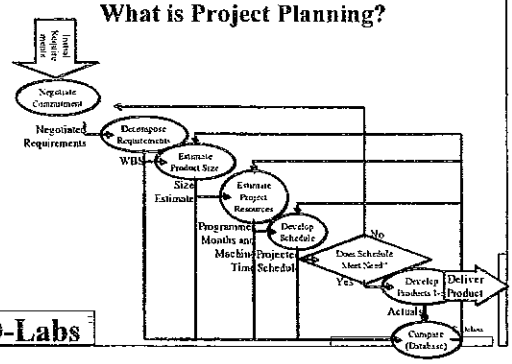
What is requirements management



Q-Labs

SZCZYRK 1998

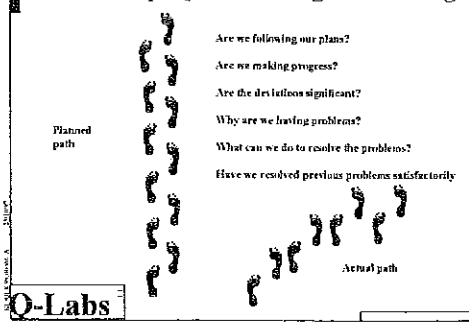
What is Project Planning?



Q-Labs

SZCZYRK 1998

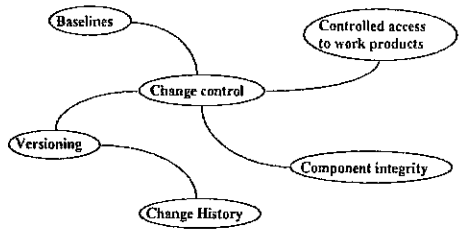
What is project tracking and oversight?



Q-Labs

SZCZYRK 1998

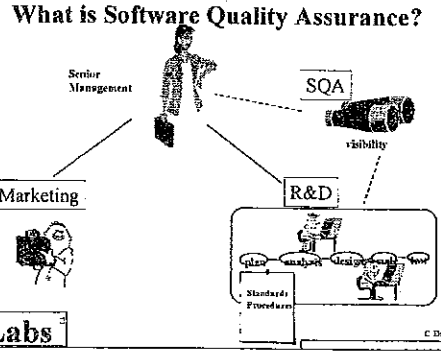
What is Software Configuration Management?



Q-Labs

SZCZYRK 1998

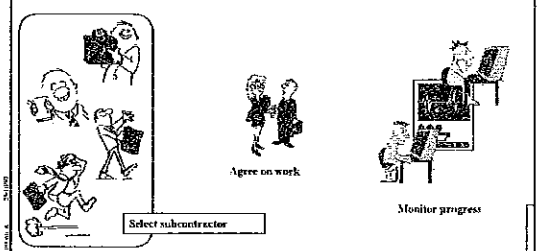
What is Software Quality Assurance?



Q-Labs

SZCZYRK 1998

What is software subcontract management?



Q-Labs

SZCZYRK 1998

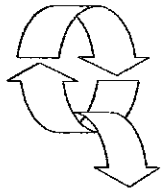
CMM and other models

- Boostrap: It is just an assessment model (questionnaire-based)
- SPICE: Coming ISO Standard:
  - just a framework for process assessment
  - CMM and others will have to be comply to SPICE requirements
- CMM is the ONLY approach with
  - Extensive experiences with Implementation and ROI
  - Detailed framework for Improvement (See book)
- But Remember: SPICE Versus CMM is the WRONG Debate

Q-Labs

SZCZYRK 1998

CMM: HOW?



Q-Labs

SZCZYRK 1998

Misconceptions about CMM

- CMM is only applicable for big organisations?
  - *No because*
    - It is a matter of wise interpretation of the principles (remember it is not a cookbook but a framework) and selection of the suitable parts
    - Big organisations are made up of SMEs
    - there exists some light application of CMM (cf Qlabs)
- CMM is DOD/American based:
  - So what?? (see above)
- CMM takes time until you can quantify benefits:
  - Any improvement initiatives takes time and effort
  - Improvement strategy/business case is key

Q-Labs

SZCZYRK 1998

SMEs as priority targets in Poland

- 1-a few hundreds software engineers
- Business orientation is ESSENTIAL: improvement should impact performance in the short term
- Low capital for investment
- Low resources for Improvement
- SMEs resistance
  - "no time for improvements": customer are expecting for products!
  - "Do not need process. We are small and have good people"

—> BUILD A BUSINESS CASE

Q-Labs

SZCZYRK 1998

Implementation of CMM Step 1: Understand CMM

- Get acquainted/trained with the model (books, experience reports, ...)
- Map CMM terminology to your culture
- Try to initially map organisational problems with what CMM is about
- Start gather information for cost/benefits analysis
- Start awareness session at all level (management, practitioners, ...)
- Get some support from external

IS CMM addressing my major problems?

Q-Labs

SZCZYRK 1998

Implementation of CMM Step 2: Identify where you are

- Understand business context through top level management involvement:
  - Business drivers (quality, costs, schedule)
  - Cost of non-quality
  - Business case
- Obtain an initial management commitment for Improvement budget
- Get your organisation assessed with third-party using CMM framework (focus on L2 issues).

Q-Labs

SZCZYRK 1998

Assessment Style

- Cost efficient (2 days to one week)
- Open questions (Questionnaire Y/N are often useless!!)
- Interview a representative sample of staff from management to practitioners through project leaders
- Taylor CMM to your needs (vocabulary, areas to be considered, ...).
- Objectives is improvement not model compliance
- Use CMM goals

Q-Labs

SZCZYRK 1998

Implementation of CMM Step 3: Produce improvement action plan

- Define software improvement goals based on Business Goals and Assessment results (Management)
- Dedicated workshop (e.g. Metaplan technique) with customer projects (Project leader and key staff) to identify and prioritised improvement actions
- Produce consolidated action plan with budget, timeframe, responsibilities, ...
- Get approval by Management (budget, resources)

Translate CMM "What" into YOUR "how"

Q-Labs

SZCZYRK 1998

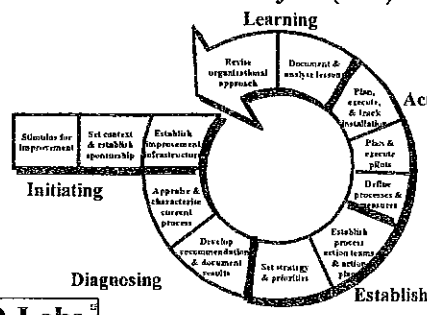
Implementation of the CMM Step 4: Do it!

- Track the improvement project as any SW projects (apply same L2 principles to it)
- Involve your customers if needed
- Continuous management follow-up to keep focus
- Measure benefits

Q-Labs

SZCZYRK 1998

IDEAL Life Cycle (SEI)



Q-Labs

SZCZYRK 1998

CMM: How Much?

Q-Labs

SZCZYRK 1998

Impact - 1

- Many companies have launched CMM software improvement programs:
  - Hughes, Raython, Lockeed, US air force, Boeing, ...
  - Motorola, Ericsson, Alcatel, Siemens Philips, ABB, Schlumberger, HP, Nokia,
  - Lloyds bank, Bank of America,ING, ...
- *This is just a sample of "big" names!*

Q-Labs

SZCZYRK 1998

Raytheon's Equipment Division

- Raytheon's Equipment Division, USA;
  - 400 software engineers
  - real-time embedded radar and air traffic control
- Started with process improvements 1988
- Level 2 1990
- Level 3 1991
- Level 4 1995

Dion (1993), Haley (1995)

Q-Labs

SZCZYRK 1998

Raytheon's Cost Analysis Method

Crosby's cost of quality model:

- Performance - cost of building it right the first time
- Non-conformance - cost of rework
- Appraisal - cost of testing
- Prevention - cost of preventing non-conformance

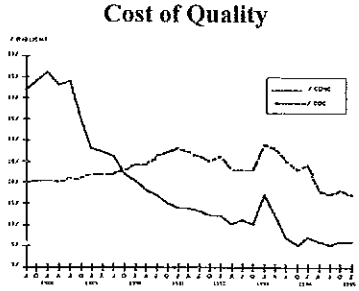
Average% project time by cost type:

	Perform	Nonconf	Appraisal	Prevent
1988	34	41	15	7
1990	55	18	15	12
1992	66	11		
1994	76	6		

Q-Labs

SZCZYRK 1998

Cost of Quality



Q-Labs

SZCZYRK 1998

Return on Investment

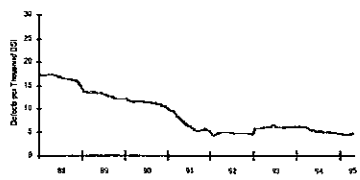
- Invests $1.000.000 annually
- Reports an ROI of 5,7 times per year
- 7.7 times in 1990

Received a $9.600.000 bonus for early delivery on one system (not included in ROI numbers!)

Q-Labs

SZCZYRK 1998

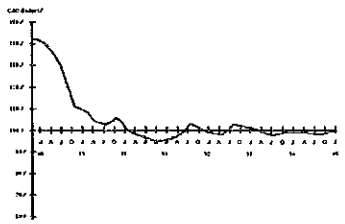
Better quality



Q-Labs

SZCZYRK 1998

Predictability



Q-Labs

SZCZYRK 1998

Costs/benefits

- Costs:
  - One full-time equivalent for 50 SW engineers to co-ordinate the improvement initiative
  - Experts/coach allocated to improvement task force (In average, 50% for one month)
  - Training staff in new processes/...
  - External support
- Benefits:
  - Measure performance baseline at the start-up (rework, #defects, ...)
  - Continuous measurement of project performance
  - Link to financial performance/customer satisfaction

Q-Labs

SZCZYRK 1998

Final Tips to YOU

- Start with a CMM-like (SPI) initiatives before going through ISO 9001 certification (≈ CMM level 3)
- Get first an understanding of what the REAL problems are (assessment of the organisation by independent third party)
- Problems are seldom technical but management-related
- The implementation of a CM system (tool + processes) can be an excellent starting point for improvement
- CMM is not all. Complement Process view with Product view (architecture, ...) + overall business process

Q-Labs

SZCZYRK 1998

Qlabs Experience with CMM

- Over 6 years of practical experiences
- Has supported extensively Ericsson centres in CMM initiative (from assessment, training to improvement actions implementation)
- Qlabs consultants had been world-wide CMM co-ordinator at ABB, Alcatel, Saab, ...
- New experiences collected in other business domains like automotive industry, banking, ...

Training/assessment/improvement package have been developed and tailored also to SMEs.

Q-Labs

Tomasz Byzia

Zarządzanie jakością oprogramowania

Zarządzanie jakością oprogramowania

Tomasz Byzia

1

Zarządzanie jakością oprogramowania

Tomasz Byzia

1

Co to jest jakość?

„Jakość jest to pewien stopień doskonałości"

Platon 427-347 pne

„Jakość jest jedną z dziesięciu kategorii, które umożliwiają podział wszystkich pojęć na grupy logiczne (czas, miejsce, ilość, substancja ...)"

Arystoteles, 384-322 pne

„Jakość to istotne cechy przedmiotu wyróżniające go spośród innych i stanowiące o jego swoistości pod danym względem"

Encyklopedia Popularna PWN

Co to jest jakość?

„Przewidywany stopień jednorodności i niezawodności przy możliwie niskich kosztach i dopasowaniu do wymagań rynku"

Deming

„Zgodność z wymaganiami"

Crosby

„Przydatność użytkowa"

Juran

„Jakość wiąże się ze stopniem spełnienia rzeczywistych wymagań odbiorcy/klienta"

Oakland

Co to jest jakość?

„Zbiorcza charakterystyka produktu i serwisu z uwzględnieniem marketingu, projektu, wykonania i utrzymania, która powoduje, że dany produkt lub usługa spełniają oczekiwania użytkownika"

Feigenbaum

„Jakość wyrobu jest stopniem spełnienia przezeń wymagań odbiorcy. W wyrobach przemysłowych jakość jest wypadkową jakości projektu, jakości wykonania oraz jakości użytkowania"

Kindlarski

Co to jest jakość?

„Jest to stopień w jakim użytkownik wierzy, że produkt lub usługa spełnia jego potrzeby i oczekiwania."

Gitlow

„Jakość jest tym, czego brak oznacza straty dla wszystkich"

Taguchi

„Ogół cech i właściwości produktu decydujących o jego zdolności do zaspokojenia stwierdzonych lub przewidywanych potrzeb"

ISO 8402

Fakty o jakości

- Powszechność
- Intuicyjność
- Kontekstowość
- Subiektywność
- Zmienność

Wniosek: *nie można podać jednoznacznej definicji jakości, jakość rodzi się dynamicznie w chwili wydawania sądu jakościowego, przez konkretną osobę, w konkretnym czasie, miejscu i sytuacji, w odniesieniu do konkretnego przedmiotu.*

Równanie jakości

Problem jakości to w istocie poszukiwanie odpowiedzi na pytanie: **dlaczego klient kupuje określony wyrób?**

Model Van Ettinger'a i Sittig'a

$$V_{jk} = \sum_i w_{ik} c_{ij} - m_k p_j$$

V_{jk} - „wartościowość" wyrobu (j) dla klienta (k)
w_{ik} - „waga" cechy (i) dla klienta (k)
c_{ij} - cecha (i) wyrobu (j)
m_k - „waga" danej jednostki monetarnej dla klienta (k)
p_j - cena wyrobu (j) w jednostkach monetarnych

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 7*

Podejścia do jakości

Podejście „tradycyjne"
Podejście „współczesne"
Podejście „orientalne"

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 8*

Podejście „tradycyjne"

Jakość jest to:
- zgodność produktu z wymaganiami (specyfikacją)

Kto ocenia jakość produktu:
- producent, ekspert

Wada jest to:
- niezgodność ze specyfikacją

Kryteria jakości:
- mierzalne obiektywne kryteria
- techniczne parametry, atrybuty i charakterystyki

Charakterystyka systemu zapewnienia jakości:
- analiza wymagań, system projektowania, kontrola techniczna, standaryzacja, dokumentacja, analiza produktu końcowego, skutków procesu projektowania

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 9*

Podejście „współczesne"

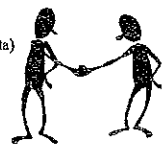
Jakość jest to:
- stopień spełnienia oczekiwań użytkownika (klienta)

Kto ocenia jakość produktu:
- klient, użytkownik, laik

Wada jest to:
- niespełnienie oczekiwań użytkownika (klienta)

Kryteria jakości:
- niemierzalne subiektywne oceny użytkownika
- stopień satysfakcji użytkownika

Charakterystyka systemu zapewnienia jakości:
- analiza wymagań, badania konkurencji i marketingowe, ocena satysfakcji klientów



Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 10*

Podejście „orientalne"

Jakość jest to:
- daleki cel, do którego należy dążyć
- wszystko to, co jeszcze można poprawić

Kto ocenia jakość produktu:
- każdy

Wada jest to:
- każdy mankament, który można poprawić

Kryteria jakości:
- subiektywne i obiektywne oceny i kryteria użytkownika i ekspertów

Charakterystyka systemu zapewnienia jakości:
- system „totalny" rozciągnięty na całą firmę, kultura jakości, koła jakości, kontrola i doskonalenie procesów

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 11*

Zarządzanie jakością - historia

Kontrola jakości

sprawdzanie, mierzenie lub testowanie jednej lub więcej charakterystyk produktu i odnoszenie wyników do wyspecyfikowanych wymagań w celu potwierdzenia zgodności. Zadanie to wykonywane jest zwykle przez wyspecjalizowany personel i nie wchodzi w zakres obowiązków pracowników produkcyjnych. Produkty niezgodne ze specyfikacjami są odrzucane lub przekazywane do poprawienia.

Statystyczna kontrola jakości

przebieg kontroli jakości ustala się na podstawie danych statystycznych i rachunku prawdopodobieństwa. Istotą jej jest ciągłość procesu kontroli.

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 12*

Zarządzanie jakością - historia

Sterowanie jakością

położenie nacisku na kontrolę, ale z włączeniem do systemu jakości pracowników produkcyjnych w celu stworzenia sprzężeń zwrotnych pomiędzy wynikami kontroli, a linią produkcyjną. Na podstawie wyników kontroli proces produkcyjny jest modyfikowany w celu otrzymania produktów zgodnych ze specyfikacjami.

Zarządzanie jakością - historia

Zapewnienie jakości

prowadzenie systematycznych i zaplanowanych działań prowadzących do wytwarzania produktów zgodnych ze specyfikacją. Wykonywanie regularnych inspekcji, przeglądów, auditów i zewnętrznych ocen w celu ciągłego zapewnienia jakości. System zapewnienia jakości jest formalnie opisany i stosowany, a jego skuteczność jest stale monitorowana.

Cechy podejścia tradycyjnego

- Zarządzanie jakością jest oddzielną sferą działalności wyodrębnioną z organizacji
- Za jakość odpowiadają wydzielone komórki organizacji
- Główny nacisk położony jest na wykrywanie wad
- Funkcyjna struktura organizacyjna firmy
- Decyzje operacyjne podejmowane są tylko przez kadrę kierowniczą
- Głównym czynnikiem motywującym jest płaca

Zarządzanie jakością - dziś

TQM *(Total Quality Management - Zarządzanie Przez Jakość)*

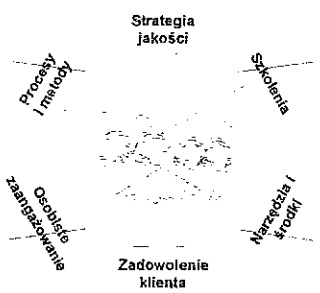
jest to skoncentrowane na jakości zarządzanie przedsiębiorstwem, przy współudziale wszystkich jego członków, w którym długofalowe korzyści przedsiębiorstwa oraz korzyści społeczeństwa osiągane są poprzez spełnianie oczekiwań klientów.

ISO 9000

Cechy podejścia dzisiejszego

- Jakość jest głównym celem przedsiębiorstwa
- Jakość jest sprawą każdego
- Jakość jest wielowymiarowa i obejmuje systemy, procesy, procedury, ludzi, itd.
- Jakość polega na zapobieganiu, a nie wykrywaniu wad
- Jakość trzeba nieustannie doskonalić

Wymiary TQM w firmie



Zasady zarządzania jakością

Zasada 1 - Ukierunkowanie na klienta
Zasada 2 - Przywództwo
Zasada 3 - Zaangażowanie ludzi
Zasada 4 - Podejście procesowe
Zasada 5 - Podejście systemowe do zarządzania
Zasada 6 - Ciągłe doskonalenie
Zasada 7 - Podejmowanie decyzji oparte na faktach
Zasada 8 - Partnerstwo dla jakości

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 19*

Ukierunkowanie na klienta

Organizacja zależy od swoich klientów, dlatego powinna dążyć do poznania i zrozumienia ich bieżących i przyszłych potrzeb, realizować ich wymagania i starać się przekraczać oczekiwania.

- klient to osoba lub organizacja, której przekazuje się wyniki swej pracy
- klient wewnętrzny i zewnętrzny
- praca dla uzyskania pełnej satysfakcji klienta, zamiast jedynie usunięcia przyczyn jego niezadowolenia

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 20*

Przywództwo

Przywódcy ustanawiają jedność celu, kierunku i wewnętrznego środowiska organizacji.

- otwartość na wymianę myśli i budowanie jasnych, długofalowych wizji organizacji
- troska o dostępny potencjał ludzki
- budowanie kultury korporacyjnej, wpływanie na zmianę postaw i zachowań
- budowanie na wartościach

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 21*

Zaangażowanie ludzi

W zapewnienie jakości zaangażowani są wszyscy pracownicy na wszystkich poziomach organizacji.

- twórcami jakości są ludzie
- ludzie stają się w pełni zaangażowani w osiąganie celów organizacji
- zaangażowanie dostarcza satysfakcji z pracy i zwiększa motywację
- szkolenia całej kadry firmy budują atmosferę współodpowiedzialności i wzajemnego zaufania

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 22*

Podejście procesowe

Pożądane rezultaty są osiągane skuteczniej, gdy odpowiednie zasoby i działania są widziane jako elementy dynamicznego procesu.

- podział obowiązków i odpowiedzialności
- łancuchy jakości
- praca bezusterkowa - „zrób dobrze za pierwszym razem"
- miary produktywności na wejściu, wyjściu i wewnątrz procesu
- doskonalenie procesu i akcje korygujące
- statystyczna kontrola procesów

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 23*

Podejście systemowe

Identyfikacja, zrozumienie i zarządzanie systemem wzajemnie powiązanych procesów przyczynia się do zwiększenia skuteczności i wydajności organizacji.

- kompleksowość działań obejmująca wszystkie etapy powstawania produktu
- kompleksowość działań widząca złożone procesy w złożonych relacjach
- doskonalenie oznacza równoległe doskonalenie wszystkich powiązanych ze sobą procesów

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 24*

Ciągłe doskonalenie

Ciągłe doskonalenie jest stałym celem organizacji.

- Kaizen, ciągła poprawa dostrzeżonych niedociągnięć, wad i mankamentów w produkcie, procesie produkcyjnym i zasobach ludzkich
- doskonalenie oparte na systemie metryk produktywności
- systematyczne przeglądy procesów i systemu jakości (audyty)

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 25*

Decyzje oparte na faktach

Skuteczne decyzje i działania są oparte na logicznej i intuicyjnej analizie danych i informacji.

- cykl:
  fakt - metryka - dane - gromadzenie - analiza - informacja - decyzja - działanie
- udokumentowane konkrety
- procedury i instrukcje
- niezgodności i akcje korygujące

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 26*

Partnerstwo dla jakości

Partnerskie powiązania między organizacją i jej dostawcami (klientami) zwiększają zdolność wspólnego tworzenia wartości.

- lepsze rozpoznanie i przystosowanie do potrzeb klienta
- współpraca, doskonalenie, skuteczność
- redukcja niedeterministyczności procesów
- budowanie klimatu zaufania
- międzyorganizacyjny efekt synergiczny

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 27*

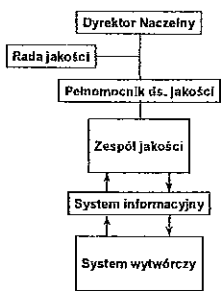
Warunki powodzenia TQM

- Zaangażowanie najwyższego kierownictwa
- Organizacja procesu ciągłego doskonalenia jakości
- Komunikacja wewnątrz przedsiębiorstwa
- Program kształcenia i szkoleń
- Współudział wszystkich pracowników
- System jakości
- Analiza (audyty) jakości

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 28*

Struktura systemu jakości

Polityka jakości
Rada jakości
Pełnomocnik ds. jakości
Zespół ds. zapewnienia jakości
System informacji o jakości



Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 29*

Polityka jakości

"Xerox jest firmą wysokiej jakości. Jakość jest naszą podstawową zasadą. Jakość oznacza dla nas dostarczanie naszym zewnętrznym i wewnętrznym klientom innowacyjnych produktów i usług, które w pełni satysfakcjonują ich wymagania. Doskonalenie jakości jest sprawą każdego pracownika Xerox'a ..."

"Od samego początku istnienia IBM jakość jest jednym z filarów działalności firmy. Znaczenie jakości w polityce personalnej, wytwarzaniu produktów, obsłudze klienta oraz ogólnym sposobie, w jaki podchodzimy do wszelkiego rodzaju zadań, widoczne jest w wielu aspektach działania firmy i zajmuje szczególne miejsce w naszych podstawowych przekonaniach ..."

Zarządzanie jakością oprogramowania © Tomasz Byzia *Strona 30*

Rada jakości

Organ doradczy naczelnego kierownictwa firmy.

- Formułuje strategiczne cele przedsiębiorstwa
- Przygotowuje decyzje programowe
- Organizuje współpracę z jednostkami badawczo-rozwojowymi
- Nadzoruje działanie systemu jakości
- Rozwiązuje problemy międzywydziałowe

Pełnomocnik ds. jakości

Osoba postawiona wysoko w hierarchii zarządzania firmy, np. w randze wicedyrektora.

- Wyznaczona przez kierownictwo firmy
- Wyposażona w duże pełnomocnictwa i kompetencje
- Opowiedzialna za działanie systemu jakości
- Niezależna i obiektywna

Zespół jakości

Opracowywanie
- metod i planów kontroli jakości produktów faz produkcyjnych
- planów badań parametrów technicznych i użytkowych
- analiz jakości wyrobów własnych i konkurencji

Kontrola
- dostaw, komponentów, kodu „cudzego"
- produkcji
- produktów faz projektowych
- produktu końcowego
- stosowania norm

Organizowanie
- systemu jakości
- współpracy z kooperantami

Zespół jakości

Współdziałanie
- w ustaleniu przyczyn niskiej jakości
- wypracowaniu skutecznych metod oddziaływania na jakość
- w analizie kosztów jakości
- w analizie wymagań jakościowych nowych wyrobów
- w zbieraniu informacji od użytkowników i odbiorców
- z zespołem doskonalenia procesów produkcyjnych

Inicjowanie
- szkoleń w zakresie problematyki jakości
- akcji uświadamiających zasady zarządzania przez jakość

Samokontrola

System informacji o jakości

Wewnętrzny system przekazywania informacji o jakości

Dokumentacja systemu jakości
- cele i polityka jakości, opis systemu (księga jakości), plany jakości
- procedury dotyczące głównych działań (procesów) i instrukcje
- kontrakty i dokumenty związane
- zapisy jakości
  - systemowe: wyniki przeglądów, ocen, audytów, akcji korygujących, przeprowadzonych szkoleń, itp.
  - kontraktowe/produktowe: przeglądy kontraktów, weryfikacja projektów, identyfikacja wyrobów, wyniki badań i kontroli, postępowanie z wyrobami niezgodnymi.

Badania rynkowe

Śledzenie opinii o produkcie, firmie, konkurencji

Zbieranie informacji o kliencie

Kroki dochodzenia do TQM

1. Ocena obecnego stanu systemu jakości
2. Szkolenia
3. Identyfikacja klientów zewnętrznych i wewnętrznych
4. Stworzenie zespołu ds. jakości
5. Stworzenie polityki jakości
6. Strategiczne planowanie działań
7. Określenie zadań projakościowych
8. Próbne wdrożenia, weryfikacja, upowszechnienie

Piotr Jędrzejowicz

Katedra Informatyki, Wyższa Szkoła Morska
ul. Morska 83, 81-832 Gdynia
pj@wsm.gdynia.pl

W jakim celu i w jaki sposób mierzyć niezawodność oprogramowania?

1. Wstęp

Rolę produktywności w procesie tworzenia oprogramowania dostrzeżono wiele lat temu. Zainteresowanie jakością oprogramowania pojawiło się później. Jakość stała się istotnym problemem dla inżynierii oprogramowania wraz z dramatycznym wzrostem zapotrzebowania na systemy informacyjne oraz rozwojem technologii informacyjnych. Popyt na systemy o złożonej architekturze sprzętowej i programowej rośnie szybciej niż możliwości ich projektowania, implementacji, testowania i utrzymywania. Nietrudno obronić tezę, że w przeciwieństwie do postępów w zakresie technologii sprzętu, rozwój po stronie oprogramowania nie nadąża w zakresie wszystkich podstawowych aspektów, tj. jakości, produktywności, kosztu i efektywności.

Twórcy oprogramowania uznają dzisiaj, że jakość i produktywność są wzajemnie powiązane. W przeszłości wielu sądziło, że jakość i produktywność są przeciwstawne, a więc, że wzrost wydajności jest możliwy tylko poprzez obniżenie jakości. Współcześni menedżerowie na własnej skórze odczuli, że powyższe założenie jest prawie zawsze błędne. Producent oprogramowania, który wprowadzi skuteczny program podnoszenia jakości może oczekiwać efektów trojakiego rodzaju. Po pierwsze produktywność jego zespołu wzrośnie, po drugie, łączne koszty produkcji oprogramowania zmniejszą się i po trzecie, konkurencyjność produktu i firmy wzrośnie.

Niezawodność jest uznawana za, prawdopodobnie, najważniejszy atrybut jakości oprogramowania [Musa et. al. 1987]. Oprogramowanie o wysokiej niezawodności posiada w większym lub mniejszym stopniu większość spośród własności uznawanych za główne cechy jakości. Dodatkowo, niezawodność okazała się tym szczególnym atrybutem jakości oprogramowania, który najłatwiej poddaje się próbom kwantyfikacji. W istocie norma ISO 9000-3 określa rejestrację uszkodzeń oprogramowania w trakcie jego użytkowania jako jedyną wymaganą miarę jakości [ISO 9000-3, 1991]. Niezawodność oprogramowania rozumiana jest jako prawdopodobieństwo działania oprogramowania bez uszkodzeń przez określony okres czasu i w określonym otoczeniu [ANSI 1991].

Ostatnio do analizy i oceny własności systemów zaczęto używać pojęcia wiarygodności (ang.: *dependability*). Wiarygodność obejmuje, między innymi, takie atrybuty jak niezawodność, gotowość, bezpieczeństwo czy naprawialność. W ogólności, wyróżnia się rozmaite zagrożenia dla wiarygodności (np. błędy i uszkodzenia) oraz środki zapewnienia wiarygodności (tolerowanie uszkodzeń, usuwanie uszkodzeń, prewencja uszkodzeń) [Laprie 1995]. Niezawodność pozostaje podstawowym atrybutem wiarygodności. Może ona być użyteczna do:

- specyfikowania własności, które spełniać powinien system;
- oceny jakości systemu w związku z zagrożeniami i użytymi środkami zapewnienia wiarygodności.

Wytwórcy oprogramowania doskonale rozumieją potrzebę dostarczania swoim klientom niezawodnych produktów zdając sobie jednocześnie sprawę, że poważna część ponoszonych kosztów związana jest z zapewnieniem, na wszystkich etapach cyklu rozwojowego, rozwiązań gwarantujących wymaganą niezawodność. Z powyższego wynika ogromne zapotrzebowanie na systematyczne i skuteczne zastosowania metod inżynierii niezawodności oprogramowania (ang.: *software*

reliability engineering). Inżynieria niezawodności oprogramowania zajmuje się ilościową analizą zachowania się oprogramowania w trakcie eksploatacji w odniesieniu do wymagań użytkownika dotyczących niezawodności [IEEE 1995]. Inżynieria niezawodności oprogramowania (INO) obejmuje pomiary niezawodności, analizę wpływu atrybutów i wielkości charakteryzujących proces projektowania, architekturę oprogramowania, implementację oraz profil eksploatacyjny na niezawodność, a także zastosowanie uzyskanej wiedzy do zarządzania procesem tworzenia oprogramowania.

Przedmiotem niniejszego wykładu jest próba ukazania w jaki sposób oraz w jakim zakresie wyniki pomiarów niezawodności mogą być wykorzystane do lepszego zarządzania produkcją oprogramowania.

2. Cele programu pomiarów niezawodności oprogramowania

Kluczowym celem każdego programu pomiarów niezawodności oprogramowania jest dostarczenie niezbędnych informacji dla dalszych czynności należących do INO, które prowadzą do lepszego zarządzania procesem tworzenia oprogramowania. Oczekiwane korzyści proponowanego podejścia obejmują pełniejszą identyfikację potrzeb klienta, skrócenie cyklu rozwoju produktu, zwiększenie produktywności oraz lepsze planowanie zużycia zasobów. Rezultaty pomiarów niezawodności są krytyczne dla:

- uzyskania pewności, że oprogramowanie osiągnęło wymagany poziom bezpieczeństwa;
- decyzji o zakończeniu testowania oprogramowania;
- analizy i zarządzania ryzykiem związanym z przekazaniem oprogramowania do użytkowania;
- zapewnieniem jakości oprogramowania.

Bezpośrednim celem każdego programu pomiarów niezawodności jest dostarczenie co najmniej jednej spośród poniższych informacji:

- estymacja aktualnej niezawodności oprogramowania;
- predykcja przyszłej niezawodności oprogramowania;
- porównanie wyników estymacji lub predykcji ze zidentyfikowanymi potrzebami klienta.

Określenie celów programu pomiarów niezawodności determinuje dalsze czynności wykonywane w ramach INO (identyfikacja profilu funkcjonalnego i operacyjnego, klasyfikacja uszkodzeń ze względu na ich znaczenie, analiza potrzeb użytkownika w zakresie niezawodności czy specyfikacja poziomu niezawodności oprogramowania). Te czynności, z kolei, wpływają na stosowane metody, modele, zasoby oraz niezbędny budżet.

3. Ramy czasowe oraz metodologia pomiarów niezawodności oprogramowania.

Ramy czasowe pomiarów niezawodności obejmują okres od sformułowania koncepcji produktu aż po jego użytkowanie przez klienta. Czynności związane z mierzeniem niezawodności powinny towarzyszyć wszystkim fazom cyklu życia oprogramowania. [Donnelly et al. 1996] formułuje następującą listę działań podejmowanych w ramach INO (czynności związane z pomiarami niezawodności zostały wyróżnione pogrubioną czcionką):

Tab.1. Działania podejmowane w ramach inżynierii niezawodności oprogramowania

Fazy cyklu życia	Działania
Analiza opłacalności oraz specyfikacja wymagań	**określenie profilu funkcjonalnego** **definiowanie i klasyfikacja uszkodzeń** **identyfikacja potrzeb klienta w zakresie niezawodności** analizy alternatywnych rozwiązań (ang.: *trade-off studies*) **określenie celów w zakresie niezawodności**
Projektowanie oraz implementacja	alokacja niezawodności do modułów poszukiwanie rozwiązania prowadzącego do osiągnięcia celów w zakresie niezawodności koncentracja zasobów w zgodzie z profilem funkcjonalnym kontrola powstawania i propagacji błędów **pomiary niezawodności oprogramowania pochodzącego ze źródeł zewnętrznych**

Testowanie i próbna eksploatacja	**określenie profilu operacyjnego** **testowanie wzrostu niezawodności** **monitorowanie postępów procesu testowania** alokacja dodatkowych zasobów na potrzeby testowania (jeśli wystąpi taka konieczność) **certyfikacja stopnia osiągnięcia celów w zakresie niezawodności**
Utrzymanie i obsługa	planowanie niezbędnych zasobów **monitorowanie niezawodności eksploatacyjnej na tle postawionych celów w zakresie niezawodności** monitorowanie satysfakcji klienta harmonogramowanie nowych rozwiązań i ulepszeń poprzez monitorowanie niezawodności wykorzystanie miar niezawodności do lepszego sterowania procesem tworzenia oprogramowania

4. Format danych o uszkodzeniach i procedury ich zbierania.

Zaprojektowanie i wdrożenie procedur zbierania danych o niezawodności wymaga podjęcia następujących kroków [Nikora, Lyu 1996]:

- opracowanie planu obejmującego cele przedsięwzięcia oraz definiującego sposoby zbierania danych. W planie tym należy określić jak często, przez kogo i w jakim zakresie będą zbierane dane, jak zebrane dane będą przetwarzane i przechowywane, jakie narzędzia programowe będą używane, jak proces zbierania danych będzie monitorowany oraz jaki wysiłek i zasoby są niezbędne do realizacji całego projektu;
- ocena dostępności, dojrzałości i użyteczności narzędzi proponowanych narzędzi programowych;
- szkolenie wszystkich uczestników projektu (analitycy systemu - projektanci, programiści, personel testujący, pracownicy pionu zapewnienia jakości, menedżerowie projektu oraz użytkownicy);
- implementacja planu oraz zapewnienie dostępu do niezbędnych zasobów;
- regularne monitorowanie procesu zbierania danych;
- zapewnienie informacji zwrotnych wszystkim uczestnikom projektu (włącznie z osobami bezpośrednio zaangażowanymi w zbieranie danych).

W ogólności, dla potrzeb inżynierii niezawodności oprogramowania zbierane mogą być dwa rodzaje danych o uszkodzeniach:

- dane o liczbie uszkodzeń (lub dane o liczbie uszkodzeń na jednostkę czasu)
- dane o czasach między uszkodzeniami.

Wykorzystanie modeli inżynierii niezawodności oprogramowania w trakcie testowania opiera się na założeniu pomiarów czasów obliczeń, w trakcie których występują uszkodzenia. Czas pracy CPU jest bezpośrednią miarą czasu obliczeń. Oprócz danych o uszkodzeniach potrzebne są dodatkowe informacje. Mogą one obejmować m.in.: klasę dotkliwości danego uszkodzenia, informacje o wykorzystywanych zasobach systemu komputerowego, liczbę nowych lub zmodyfikowanych instrukcji kodu źródłowego, średni czas wykonywania takiej instrukcji, wielkość programu mierzona liczbą instrukcji kodu wynikowego, identyfikator testowanej funkcji, czas trwania oraz data aktualnej sesji testowania. Dalsze dane będą potrzebne dla analiz dotyczących alternatywnych rozwiązań programowych.

Wykorzystanie modeli INO w trakcie użytkowania, obok rejestracji uszkodzeń oraz czasów ich napraw, wymaga dodatkowych informacji [IEEE 1988]. Zbierane dane mogą obejmować m.in. czasy wystąpienia uszkodzeń, datę otrzymania informacji o uszkodzeniu, identyfikator klienta oraz instalacji, klasę dotkliwości uszkodzenia oraz okres degradacji, przyczyny uszkodzenia - bezpośrednie i pierwotne, poziom, na którym uszkodzenie wystąpiło (system, podsystem), aktualny status problemu oraz procedury jego rozwiązywania. Dodatkowo, zbierane mogą być informacje o wersji oprogramowania, typie uszkodzenia (błąd danych lub błąd programu), opis uszkodzenia lub dane wejściowe reprodukujące uszkodzenie.

W przypadku oprogramowania użytkowanego równolegle w wielu wersjach oraz w przypadku programów "z półki" lub, po prostu w przypadkach kiedy kontekst użytkowania programu przez klienta tak dyktuje, dane o uszkodzeniach rejestrowane są tylko w dziedzinie czasu kalendarzowego. Zazwyczaj bezpośrednie monitorowanie intensywności użytkowania takiego oprogramowania jest niemożliwe bądź mało praktyczne.

5. Wymagane zasoby oraz koszty pomiarów niezawodności.

Programy pomiarów niezawodności oraz niektóre pozostałe czynności charakterystyczne dla inżynierii niezawodności oprogramowania, mogą być włączane do procesów rozwoju oprogramowania stopniowo, na bazie kolejnych projektów lub mogą być traktowane jako całościowe rozwiązanie strategiczne obejmujące pełne spektrum działań podejmowanych w ramach INO. Obydwa podejścia wymagają odpowiedniego zaplanowania i alokacji potrzebnych zasobów. Zasoby, o których mowa obejmują, z zasady, wiedzę (ang.: *know-how*), pracę i czas. Obydwa podejścia do wdrażania metod inżynierii niezawodności oprogramowania wymagają także przygotowań, zaangażowania oraz motywacji na różnych szczeblach organizacyjnych. Pogląd na niezbędną wielkość angażowanych środków dają informacje przedstawione w tablicy 2.

Table 2. Koszty wybranych projektów INO

Projekt	Czynność	Koszt
Projekt realizowany przez 10-osobowy zespół, około 100000 wierszy kodu źródłowego, czasokres trwania projektu - 18 miesięcy [Donnely et al. 1996]	• transfer wiedzy z zakresu INO • identyfikacja profilu operacyjnego • zbieranie i analiza danych testowych	• 1 osoba/miesiąc (konsultant zewnętrzny) • około 2 osobo/miesięcy • około 0.3 osobo/miesięcy
Bardzo duży projekt (zespół ponad 1000 osób, wiele różnych dużych profili operacyjnych) [Donnely et al. 1996]	• pełne spektrum działań INO	• jedna osoba, pełen etat
Średni projekt "System informacyjny dla małej firmy", 80,000 linii kodu źródłowego, czasokres trwania projektu 12 miesięcy [Jędrzejowicz et al. 1997]	• szkolenie i przygotowania • identyfikacja profilu funkcjonalnego i operacyjnego • zbieranie i analiza danych o uszkodzeniach w trakcie testowania • zbieranie i analiza danych o uszkodzeniach w trakcie użytkowania przez 1 rok kalendarzowy	- 1 osoba/miesiąc • 1 osoba/miesiąc • około 2 osobo/miesięcy • 2 około osobo/miesięcy
Duży projekt (system operacyjny czasu rzeczywistego, użytkowany w wielu wersjach) [Jędrzejowicz et al. 1997]	• zbieranie danych o uszkodzeniach w trakcie użytkowania przez 3 lata kalendarzowe • analiza danych eksploatacyjnych	• około 6 osobo/miesięcy • 1.5 osobo miesięcy (konsultant zewnętrzny)

Bezpiecznym założeniem dla większości projektów jest przyjęcie, że koszty działań związanych z inżynierią niezawodności oprogramowania nie przekroczą 1.5% - 2% wszystkich pozostałych kosztów.

6. Wdrażanie programu mierzenia niezawodności oprogramowania.

Mierzenie niezawodności oprogramowania jest logicznym pierwszym krokiem wdrażania w organizacji działań w ramach inżynierii niezawodności oprogramowania. Sukces programu mierzenia niezawodności zależy od zaangażowania i motywacji kierownictwa najwyższego szczebla. Program początkuje zwykle przygotowanie i przedstawienie kierownictwu firmy syntetycznej informacji o celach i założeniach przedsięwzięcia. Z kolei, przeprowadzana jest seria szkoleń. Początkuje ją krótki kurs o charakterze przeglądowym dla kierowników projektów softwarowych oraz pogłębiony kurs z zakresu INO dla członków zespołu odpowiedzialnego za po-

miary. Równolegle, lub bezpośrednio po zakończeniu tych kursów, celowe jest przeprowadzenie warsztatu planistycznego, na którym formułuje się wstępny plan działań wdrożeniowych. W tej fazie implementacji organizacja może odnieść korzyści z zatrudnienia i wykorzystania wiedzy konsultanta zewnętrznego.

Dalsze przygotowania i planowanie obejmują podjęcie decyzji o sposobie zbierania danych o uszkodzeniach (automatyczne lub ręczne). Konsekwencją powyższego jest decyzja o alokacji odpowiedniego personelu i zasobów niezbędnych do wykonania postawionych zadań. Na tym etapie dokonywana jest wstępna analiza danych o uszkodzeniach, która pozwala wyeliminować informacje nadmiarowe i nieistotne. W efekcie projektowane są procedury filtracji i wstępnego przetwarzania danych o uszkodzeniach, które pozwalają na uzyskanie istotnych informacji w formacie umożliwiającym dalsze analizy

W ramach czynności związanych z wdrażaniem programu mierzenia niezawodności podejmowane są także decyzje co do wyboru narzędzi analitycznych dla całego projektu. Decyzje w powyższym zakresie uwzględniają dostępność, koszt przygotowania, zakupu oraz instalacji i utrzymania, liczbę projektów softwarowych, dla których będą prowadzone pomiary, rodzaje tworzonych systemów, przyjazność narzędzi, harmonogram projektu oraz rodzaj zbieranych danych jak również możliwości analityczne i inne cechy narzędzi INO.

W ramach działań wdrożeniowych podejmowana jest także decyzja o wstępnym wyborze modeli oraz sposobach ich wykorzystania. Po wyborze modelu jego jakość i przydatność mogą być oceniane metodami analitycznymi. Wstępny wybór modeli opiera się raczej na innych kryteriach (doświadczenia z podobnych projektów mogą tu być dostateczną przesłanką).

7. Weryfikacja i walidacja programu mierzenia niezawodności oprogramowania.

Weryfikacja programu mierzenia niezawodności oprogramowania polega na zapewnieniu, że metody i techniki używane do zbierania danych o uszkodzeniach, a następnie do analizy tych danych w celu otrzymania predykcji i estymacji, zostały zastosowane w poprawny sposób. Weryfikacja dotyczy oceny procedur i działań w aspekcie zapewnienia, że otrzymane rezultaty są prawdziwe, zaś wnioski zostały wysnute w sposób właściwy. Kroki związane z weryfikacją pomiarów niezawodności powinny zostać precyzyjnie zaplanowane już na etapie przygotowania projektu. Realizacja tych kroków musi być monitorowana na poziomie kierownictwa projektu.

Walidacja, z kolei, obejmuje zapewnienie, że wybrano odpowiednią metodologię i wdrożono efektywne rozwiązanie. Jest to zadanie złożone. W przypadku niezawodności oprogramowania należy nie tylko zademonstrować empirycznie spełnienie warunku reprezentatywności [Fenton 1995] oraz wiarygodności otrzymanych predykcji, ale także przekonać, że proporcje pomiędzy korzyściami, a kosztami projektu nie mogły być bardziej korzystne. Walidacja obejmuje, między innymi, ocenę dokładności stosowanych modeli, błędów predykcji oraz stacjonarności analizowanych danych o uszkodzeniach.

8. Wnioski.

Chociaż działania wymagane w trakcie implementacji, a potem realizacji programu mierzenia niezawodności oprogramowania wydają się względnie proste i dobrze zdefiniowane, to jednak istnieje wiele czynników, które mogą zmniejszyć szanse powodzenia całego programu. Pośród nieprzyjaznych czynników egzogenicznych, których wystąpienie powinno być brane pod uwagę, wymienić należy:

- niestacjonarność danych dotyczących niezawodności oprogramowania;
- zmienne schematy użytkowania oprogramowania;
- zmienność profilu funkcjonalnego aplikacji;
- zmienność otoczenia programowego.

Pośród nieprzyjaznych czynników endogenicznych do istotnych należą następujące:

- brak motywacji i zaangażowania ze strony personelu odpowiedzialnego za rozwijanie oprogramowania. Przyczyną powyższego może być obawa, że estymacje niezawodności posłużą do oceny jakości pracy twórców oprogramowania. Inną przyczyną jest nieznajomość metodologii INO;

- niechęć, ze strony kierownictwa firmy do rozpowszechniania informacji o niezawodności własnych produktów softwarowych, a w szczególności do udostępniania takich informacji konkurencji

Pomimo obecności wspomnianych czynników inżynieria niezawodności oprogramowania obejmująca mierzenie niezawodności oprogramowania, jest obecnie dojrzałą dziedziną inżynierską, która dysponuje sprawdzonymi metodami, narzędziami oraz schematami działań.

9. Literatura

[ANSI 1991] *Standard Glossary of Software Engineering Terminology,* STD-729-1991, ANSI/IEEE, 1991.

[Donnelly et al. 1996] Donnelly M., B.Everett, J.Musa, G.Wilson: *Best Current Practice of SRE,* in: M.R. Lyu (ed.), Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw - Hill, New York, 1996, pp. 219 - 254.

[Fenton 1995] Fenton N.E.: *Software Measurement: A Necessary Scientific Basis,* in: in: B.Randell, J.-C.Laprie, H.Kopetz, B.Littlewood (eds.), Predictably Dependable Computing Systems, Springer, Berlin, 1995, pp. 67 - 84.

[IEEE 1988] *Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software,* IEEE Std-982.2-1988, IEEE, New York, 1988.

[IEEE 1995] *Charter and Organization of the Software Reliability Engineering Committee,* IEEE, New York, 1995.

[ISO 9000-3 1991] *Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software,* ISO 9000-3, 1991.

[Jędrzejowicz et al. 1997] Jędrzejowicz P., E,Rudolph: *Experiences in software product assessment,* Proc. 8th European Software Control and Metrics Conference, Berlin, 1997, pp. 150 – 155.

[Laprie 1995] Laprie J.-C.: *Dependability - Its Attributes, Impairments and Means,* in: B.Randell, J.-C.Laprie, H.Kopetz, B.Littlewood (eds.), Predictably Dependable Computing Systems, Springer, Berlin, 1995, pp. 3 - 24.

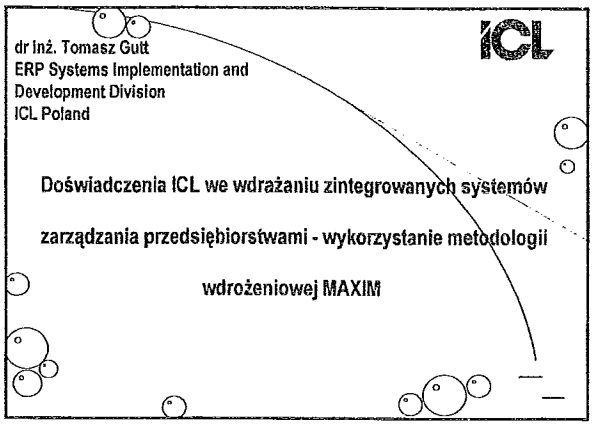
[Musa et al. 1987] Musa J.D., Iannino A., Okumoto K.: *Software Reliability. Measurement, Prediction, Application,* McGraw-Hill, New York,1987.

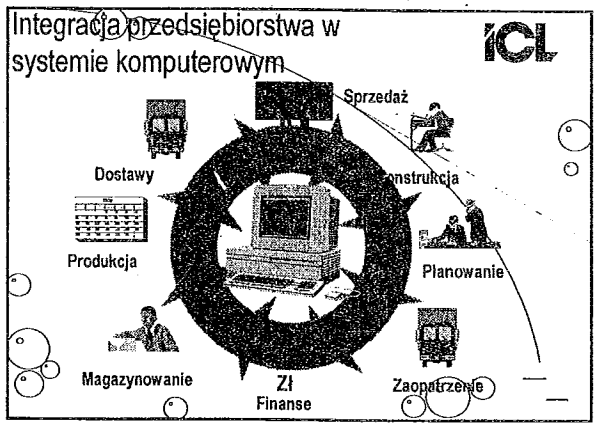
[Nikora, Lyu 1996] Nikora A.P., M.R.Lyu: *Software Reliability Measurement Experience,* , in: M.R. Lyu (ed.), Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw - Hill, New York, 1996, pp. 255 - 302.

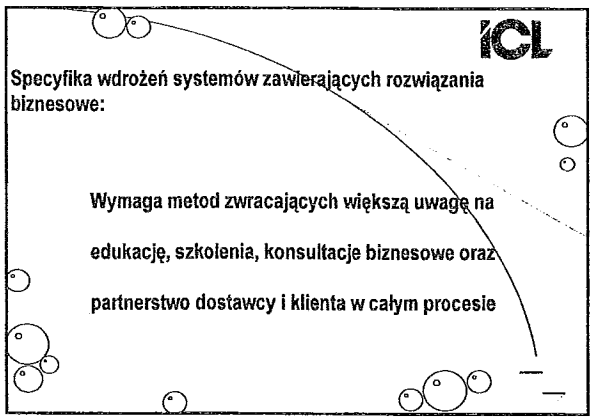
Tomasz Gutt

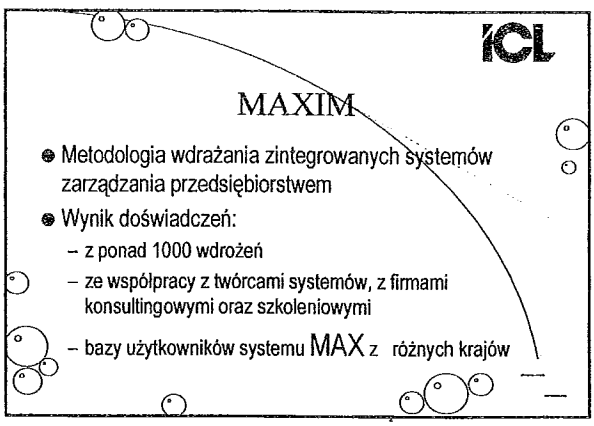
ERP Systems Implementation and Development Division ICL Poland

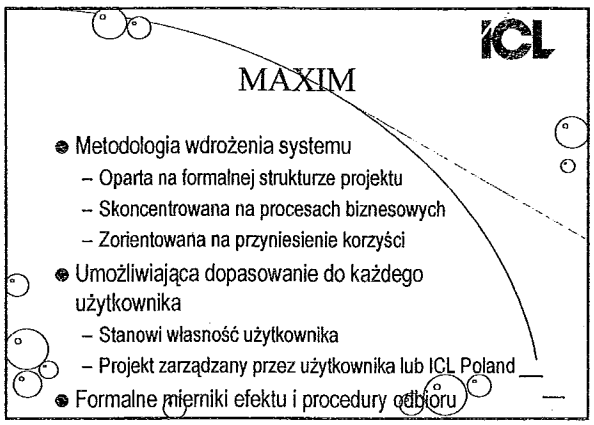
Doświadczenia ICL we wdrażaniu zintegrowanych systemów zarządzania przedsiębiorstwami - wykorzystanie metodologii wdrożeniowej MAXIM

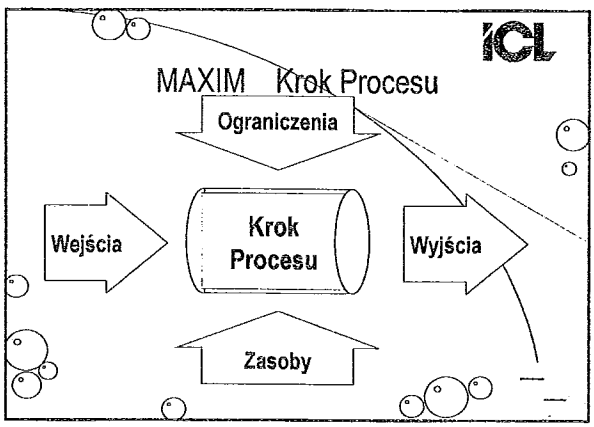


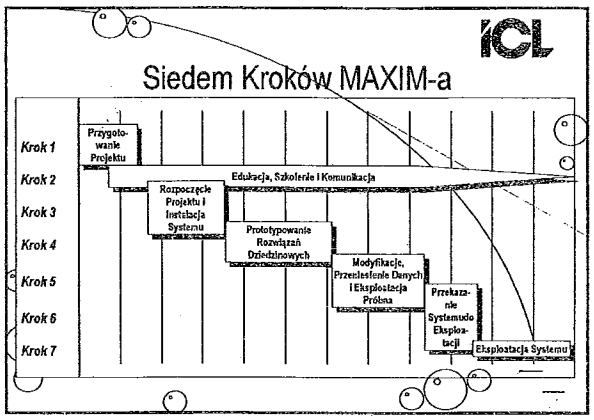


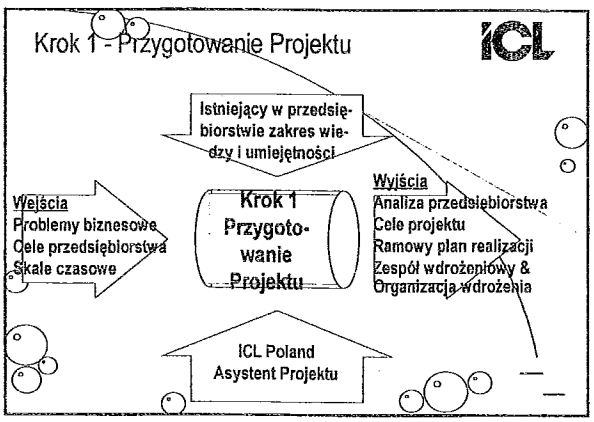





ICL
MAXIM
Metodologia wdrożenia systemu
– Oparta na formalnej strukturze projektu
– Skoncentrowana na procesach biznesowych
– Zorientowana na przyniesienie korzyści
Umożliwiająca dopasowanie do każdego użytkownika
– Stanowi własność użytkownika
– Projekt zarządzany przez użytkownika lub ICL Poland
Formalne mierniki efektu i procedury odbioru

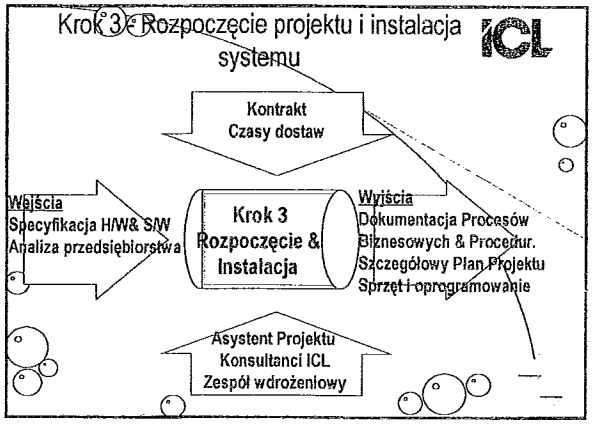

ICL
MAXIM Krok Procesu
Ograniczenia
Wejścia
Krok Procesu
Wyjścia
Zasoby

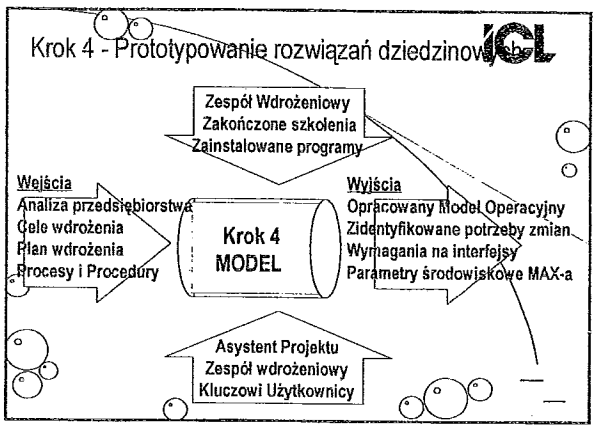

ICL
Siedem Kroków MAXIM-a
Krok 1
Krok 2
Krok 3
Krok 4
Krok 5
Krok 6
Krok 7
Przygotowanie Projektu
Edukacja, Szkolenie i Komunikacja
Rozpoczęcie Projektu i Instalacja Systemu
Prototypowanie Rozwiązań Dziedzinowych
Modyfikacje, Przeniesienie Danych i Eksploatacja Próbna
Przekazanie Systemu do Eksploatacji
Eksploatacja Systemu

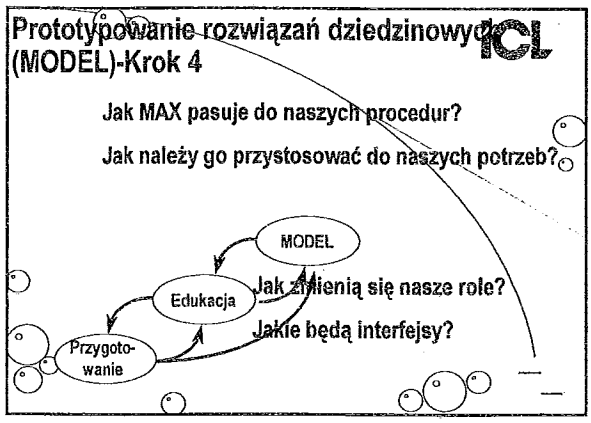

ICL
Krok 1 - Przygotowanie Projektu
Istniejący w przedsiębiorstwie zakres wiedzy i umiejętności
Wejścia
Problemy biznesowe
Cele przedsiębiorstwa
Skale czasowe
Krok 1 Przygotowanie Projektu
Wyjścia
Analiza przedsiębiorstwa
Cele projektu
Ramowy plan realizacji
Zespół wdrożeniowy &
Organizacja wdrożenia
ICL Poland
Asystent Projektu

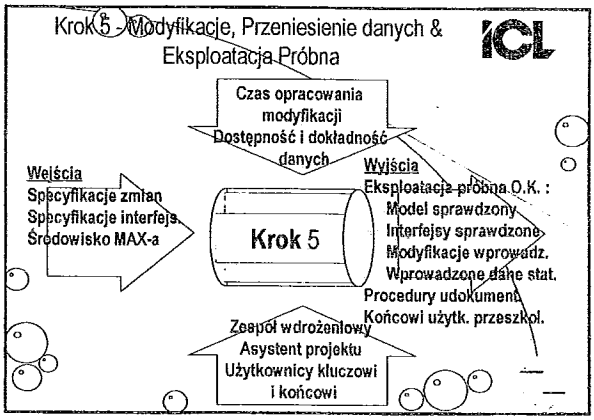
ICL
Przygotowanie Projektu - Krok 1
Jakie są cele wdrożenia?
Wdrożenie
Które procesy są krytyczne?
Kogo obejmie wdrożenie?
Jak musimy się zorganizować?
Przygotowanie

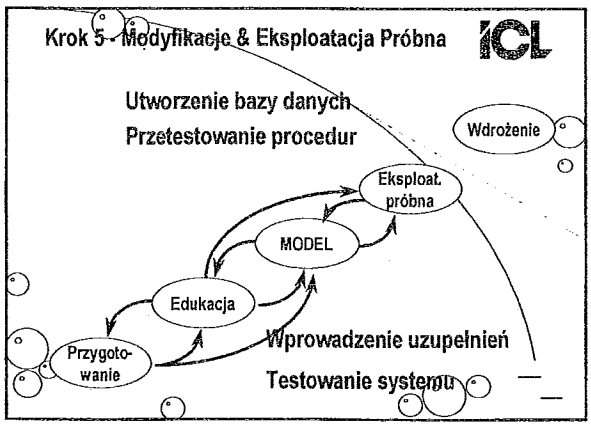
ICL
MAXIM Edukacja - Krok 2
Jak uzyskać korzyści?
Wdrożenie
Komu potrzebna jest edukacja?
Co muszą wiedzieć?
Edukacja
Przygotowanie

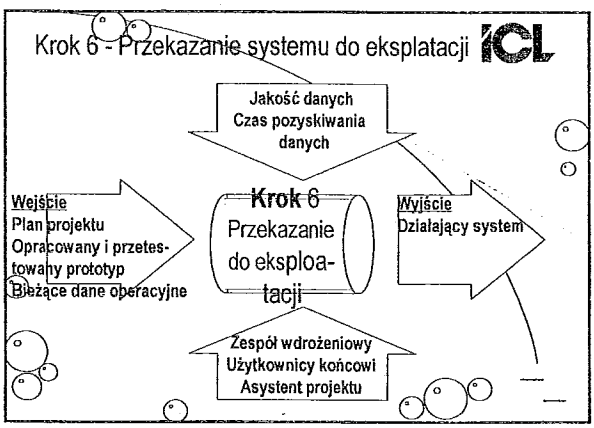

Krok 3 - Rozpoczęcie projektu i instalacja systemu
ICL
Kontrakt
Czasy dostaw
Wejścia
Specyfikacja H/W& S/W
Analiza przedsiębiorstwa
Krok 3
Rozpoczęcie & Instalacja
Wyjścia
Dokumentacja Procesów Biznesowych & Procedur.
Szczegółowy Plan Projektu
Sprzęt i oprogramowanie
Asystent Projektu
Konsultanci ICL
Zespół wdrożeniowy

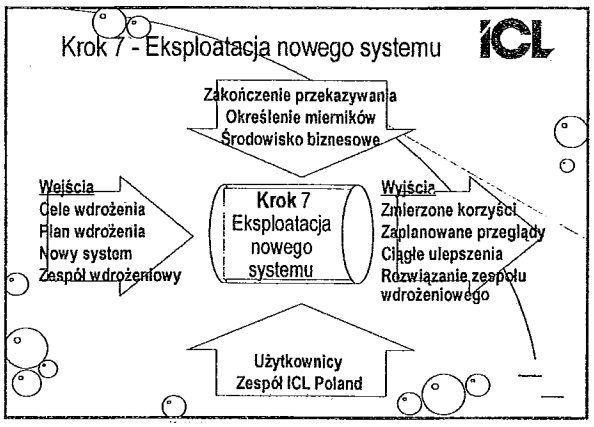

Krok 4 - Prototypowanie rozwiązań dziedzinowych
ICL
Zespół Wdrożeniowy
Zakończone szkolenia
Zainstalowane programy
Wejścia
Analiza przedsiębiorstwa
Cele wdrożenia
Plan wdrożenia
Procesy i Procedury
Krok 4
MODEL
Wyjścia
Opracowany Model Operacyjny
Zidentyfikowane potrzeby zmian
Wymagania na interfejsy
Parametry środowiskowe MAX-a
Asystent Projektu
Zespół wdrożeniowy
Kluczowi Użytkownicy

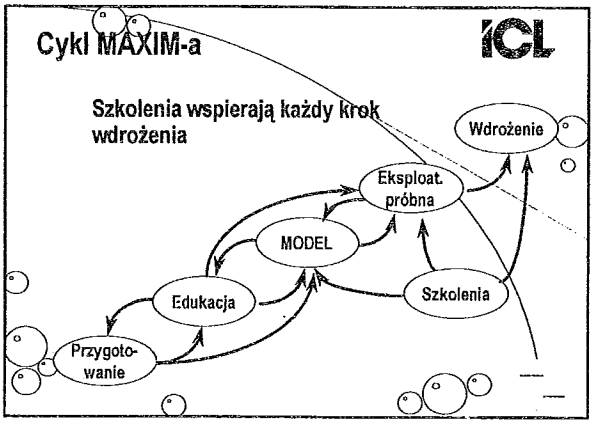

Prototypowanie rozwiązań dziedzinowych (MODEL)-Krok 4
ICL
Jak MAX pasuje do naszych procedur?
Jak należy go przystosować do naszych potrzeb?
MODEL
Edukacja
Przygoto-wanie
Jak zmienią się nasze role?
Jakie będą interfejsy?

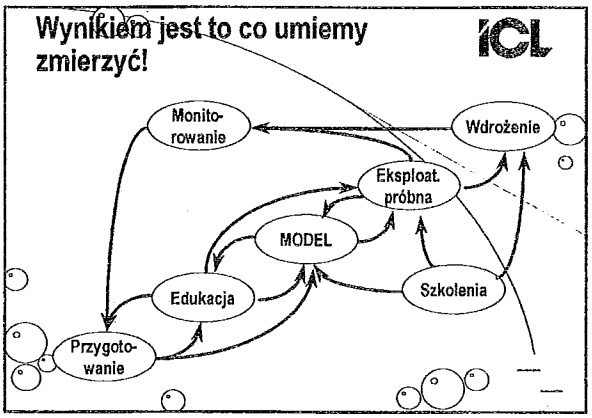

Krok 5 - Modyfikacje, Przeniesienie danych & Eksploatacja Próbna
ICL
Czas opracowania modyfikacji
Dostępność i dokładność danych
Wejścia
Specyfikacje zmian
Specyfikacje interfejs.
Środowisko MAX-a
Krok 5
Wyjścia
Eksploatacja próbna O.K. :
Model sprawdzony
Interfejsy sprawdzone
Modyfikacje wprowadz.
Wprowadzone dane stat.
Procedury udokument.
Końcowi użytk. przeszkol.
Zespół wdrożeniowy
Asystent projektu
Użytkownicy kluczowi i końcowi

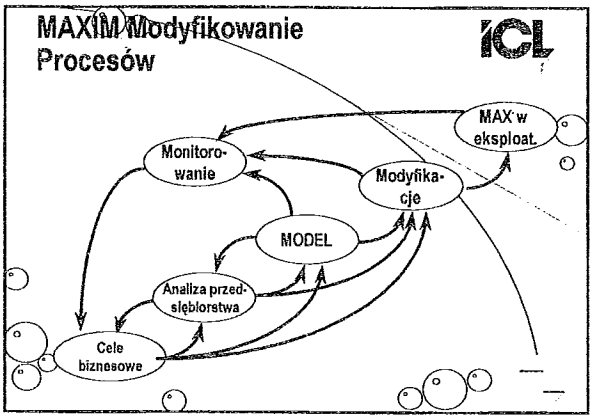

Krok 5 - Modyfikacje & Eksploatacja Próbna
ICL
Utworzenie bazy danych
Przetestowanie procedur
Wdrożenie
Eksploat. próbna
MODEL
Edukacja
Przygoto-wanie
Wprowadzenie uzupełnień
Testowanie systemu

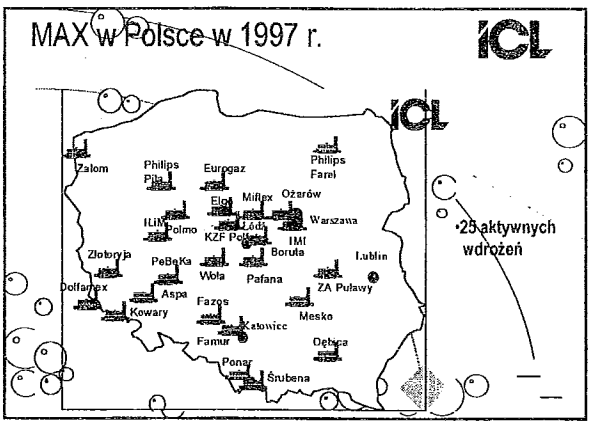

Krok 6 - Przekazanie systemu do eksplatacji
ICL
Jakość danych
Czas pozyskiwania danych
Wejście
Plan projektu
Opracowany i przetes-towany prototyp
Bieżące dane operacyjne
Krok 6
Przekazanie do eksploa-tacji
Wyjście
Działający system
Zespół wdrożeniowy
Użytkownicy końcowi
Asystent projektu

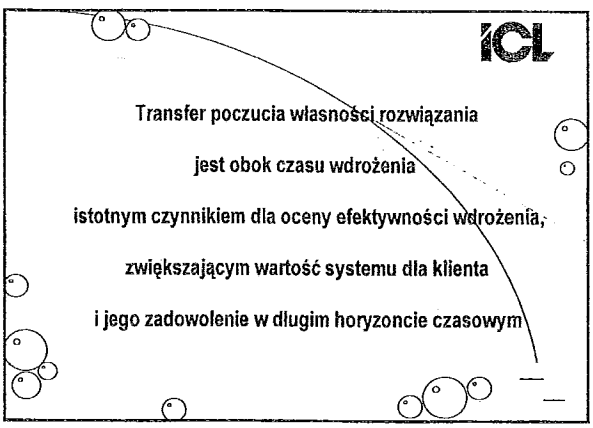

Krok 7 - Eksploatacja nowego systemu
ICL
Zakończenie przekazywania
Określenie mierników
Środowisko biznesowe
Wejścia
Cele wdrożenia
Plan wdrożenia
Nowy system
Zespół wdrożeniowy
Krok 7
Eksploatacja
nowego
systemu
Wyjścia
Zmierzone korzyści
Zaplanowane przeglądy
Ciągłe ulepszenia
Rozwiązanie zespołu
wdrożeniowego
Użytkownicy
Zespół ICL Poland


Cykl MAXIM-a
ICL
Szkolenia wspierają każdy krok
wdrożenia
Wdrożenie
Eksploat.
próbna
MODEL
Edukacja
Szkolenia
Przygoto-
wanie


Wynikiem jest to co umiemy
zmierzyć!
ICL
Monito-
rowanie
Wdrożenie
Eksploat.
próbna
MODEL
Edukacja
Szkolenia
Przygoto-
wanie


MAXIM Modyfikowanie
Procesów
ICL
MAX w
eksploat.
Monitoro-
wanie
Modyfika-
cje
MODEL
Analiza przed-
siębiorstwa
Cele
biznesowe


MAX w Polsce w 1997 r.
ICL
ICL
Zalom
Philips
Pila
Eurogaz
Philips
Farel
Elgo
Miflex
Ożarów
ILIM
Polmo
Warszawa
KZF Polska
Łódź
IMI
Boruta
Złotoryja
PeBeKa
Woła
Pafana
Lublin
Dolfamex
Aspa
ZA Puławy
Kowary
Fazos
Mesko
Katowice
Famur
Dębica
Ponar
Śrubena
•25 aktywnych
wdrożeń


ICL
Transfer poczucia własności rozwiązania
jest obok czasu wdrożenia
istotnym czynnikiem dla oceny efektywności wdrożenia,
zwiększającym wartość systemu dla klienta
i jego zadowolenie w długim horyzoncie czasowym

Marek Miłosz

Katedra Informatyki, Politechnika Lubelska
e-mail: marekm@pluton.pol.lublin.pl
tel.: 0-601-334 934

Estymacja zasobów w projektach informatycznych

Wykład prezentuje problemy szacowania zapotrzebowania na zasoby w trakcie realizacji systemów informatycznych. Po klasyfikacji technik estymacji parametrów projektu w różnych jego fazach omówione zostaną podstawowe techniki estymacji pracochłonności projektów informatycznych (metoda punktów funkcyjnych i COCOMO). Przedstawione zostaną także problemy dostosowywania modeli do konkretnych warunków realizacji projektów informatycznych i formowania zespołów ludzkich. Poza estymacją zasobów potrzebnych do wytworzenia metryki oprogramowania mogą spełniać różne funkcje. W wykładzie będzie zwrócona uwaga na te niekonwencjonalne zastosowania.

1. Wstęp

Oprogramowanie charakteryzuje się całym szeregiem cech, które odróżniają go od dowolnego innego, materialnego produktu. Jego niewidzialność, abstrakcyjność i duża złożoność powoduje znaczne trudności na wszystkich etapach realizacji projektu informatycznego. Projekty informatyczne są więc „ciężkimi" w planowaniu i realizacji. Występuje w nich tzw. negatywny efekt skali. Dla wielu projektów „materialnych" (np. budowlanych) dodatni efekt skali powoduje, że jednostkowe koszty produkcji maleją przy wzroście jej rozmiarów. W przedsięwzięciach informatycznych jest na ogół na odwrót. Przy wzroście złożoności systemu koszt jego wytworzenia wzrasta nieproporcjonalnie. Efekt ten jest związany z lawinowym wzrostem stopnia komplikacji systemu informatycznego (mierzonego np. liczbą związków pomiędzy jego poszczególnymi elementami systemu) przy wzroście jego wielkości. Wzrost ten jest tylko w niewielkim stopniu kompensowany poprzez zdobywanie doświadczenia przez zespół, specjalizacja pracowników, ponowne użycie tych samych komponentów, redukcja wpływu stałych kosztów projektu itp.

W projektach informatycznych mamy do czynienia z paradoksalną, zdawało by się, sytuacją. Częstokroć rozpoczynając projekt informatyczny nie wiemy dokładnie w jakiej technologii i przy pomocy jakich narzędzi będzie on zrealizowany. Dzieje się tak w wyniku naturalnego dążenia twórców systemu do utrzymania nowoczesności zrealizowanych rozwiązań. A od inicjalizacji projektu do jego ukończenia rzadko upływa mniej niż rok czy dwa. Jest to okres, w którym zmieniają się nie tylko wersje produktów informatycznych a nawet całe technologie.

Projekty informatyczne są zwykle źle uwarunkowane. Nie stoją za nimi setki i tysiące lat doświadczeń ludzkości (jak np. w budownictwie czy medycynie) lub też duże zbiorowe doświadczenie i dobrze sprecyzowane wyniki (jak np. przy programowaniu gier w systemie Windows). Projekt informatyczny oznacza zawsze nowość sytuacji, losowość realizacji, ograniczony zakres i przydatność posiadanych doświadczeń.

Tymczasem zwykle po wstępnym zdefiniowaniu potrzeb, a co za tym idzie zakresu projektu zwykle padają pytania: kiedy? za ile? przy pomocy czego? Odpowiedzi na te i inne tego typu pytania wymagają przeprowadzenia szeregu oszacowań czyli estymacji.

Tendencje rozwojowe systemów informatycznych wykazują jednoznacznie wzrost udziału kosztów wytworzenia oprogramowania (a więc analizy, projektowania, kodowania, testowania, wdrożenia, utrzymania i rozwoju) w ogólnych kosztach. Czynnik o dużym stopniu przewidywalności jakim jest zwykle koszt sprzętu staje się coraz mniejszy. Proces ten utrudnia szacowanie pracochłonności, a zatem i kosztów projektów informatycznych.

2. Estymacja i jej zakres w projekcie informatycznym

W projektach szacuje się te nieznane wielkości, których nie można wyliczyć, wyznaczyć lub wartości które nie wynikają w sposób jednoznaczny z zewnętrznych uwarunkowań. Są to wielkości niederministyczne. Wpływa na nie wiele różnych zmiennych elementów natury ludzkiej, technicznej i zewnętrznej - z otoczenia projektu informatycznego. Rezultatem procesu oszacowania jest więc probabilistyczna wielkość - szacunek, estymator. Estymacja (oszacowanie) uogólnia ciąg doświadczeń pojedynczej osoby, zbiór jednostkowych doświadczeń wielu ludzi lub też oba elementy łącznie. Estymację otrzymuje się przy pewnych założeniach, mniej lub bardziej oczywistych.

Oszacowaniu w projekcie informatycznym podlegają wszystkie zasoby. Począwszy od najbardziej naturalnych, takich jak: pracochłonność, wydajność pracownika, wielkość zatrudnienia w każdej fazie projektu, koszty w różnych przekrojach i czasie, zasoby potrzebne do realizacji projektu (urządzenia i ich obciążenie, materiały itp.), aż po mniej oczywiste: straty czasu na działania nieprodukcyjne, kwalifikacje pracowników, jakość obsługi, zagrożenia realizacji projektu. Nawet takie elementy jak założenia przy jakich jest realizowany projekt są w wielu przypadkach jedynie szacunkiem. Szacuje się bowiem czas pracy w firmie (nie zawsze jest to przepisowe osiem godzin), liczbę podróży służbowych do klienta, czas dojazdu itp.

Estymacji podlegają wszystkie elementy, których koszty mogą w istotny sposób wpływać na koszt realizacji projektu. Dokładność szacunków zależy od wielkości tych kosztów, a w zasadzie od stosunku kosztów błędu w oszacowaniu do kosztu procesu oszacowania. Stopień ważności i rola oszacowania rośnie wraz ze wzrostem zakresu, złożoności i kosztów projektu informatycznego. Podobnie jak konsekwencje błędu w szacunkach.

W trakcie realizacji projektu wiele oszacowań podlega ponownym ocenom. Związane jest to z następującymi procesami:

- stopniowy wzrost dokładności ocen w miarę realizacji projektu;
- oddziaływanie różnorodnych czynników (wewnętrznych i zewnętrznych) na proces realizacji projektu, prowadzące do konieczności modyfikacji założenia ocen.

3. Problemy i zagrożenia związane z szacowaniem projektów

Yourdon wskazuje na szereg zagrożeń związanych z szacowaniem w projektach informatycznych:

- różnica pomiędzy szacowaniem (realizm) a negocjacjami (kompromis);
- konieczność uwzględnienia nie średnich a realnych zdolności zespołu realizującego projekt (w tym, bardzo często, i zdolności uczenia się - „krzywa uczenia");
- duża niepewność samooceny twórców systemów informatycznych;
- brak dokładnych danych historycznych;
- nacisk kierownictwa na zbyt szczegółowe i przedwczesne oszacowania;
- optymistyczne założenia dotyczące pracy w nadgodzinach, „szturmowania terminów";
- trudności w ustanowieniu miary produktywności analityków, projektantów i programistów.

W trakcie planowania projektów informatycznych kierownicy powinni być świadomi tych zagrożeń.

4. Metody szacowania projektów

Szacowanie projektów informatycznych jest kombinacją danych historycznych (doświadczeń) i metod. Metody są różne i zwykle wykorzystywane są łącznie. Można wyróżnić następujące potencjalne opcje szacowania pracochłonności projektów informatycznych:

- opóźnianie oszacowań,
- analogia,
- metody dekompozycyjne,
- ekstrapolacja przy pomocy różnych empirycznych modeli parametrycznych.

Można wyróżnić także niekonwencjonalne metody szacowania:

- szacowanie zgodnie z prawem Parkinsona: „projekty informatyczne praktycznie zawsze są wykonywane przy założonych zasadach";
- szacowanie dla wygranej: „koszt jest określany na podstawie oceny możliwości klienta i działań konkurentów".

Opóźnianie procesu oszacowania zasobów w projekcie i przenoszenie go na późniejsze etapy projektu ma swoje uzasadnienie w zmniejszaniu się pola niepewności w miarę realizacji projektu. W praktyce jednak nie zawsze udaje się stosować tą „metodę". Prowadzi ona bowiem do wykonania kosztorysu powykonawczego.

Szacowanie przez analogię bardzo często jest porównywane do „metody sufitu" (w krajach anglosaskich: „reguła kciuka"). Wymaga ono posiadania dużego doświadczenia w realizacji podobnych projektów. Można w tej metodzie opierać się także na doświadczeniach innych projektów informatycznych. Doświadczenia te, ujęte w wykresy i tabele pracochłonności poszczególnych etapów projektów umożliwiają, po oszacowaniu nakładów na jeden etap, wyznaczenie nakładów na cały projekt.

Metody dekompozycyjne polegają na połączeniu metody top-down i bottom-up. Ich idea zakłada podział projektu informatycznego na szereg drobnych zadań (dekompozycja), oszacowanie pracochłonności każdego z nich i wyznaczenie na tej podstawie (synteza) pracochłonności całego projektu. Realizują one następujący schemat: **dekompozycja ⇒ estymacja ⇒ synteza**.

Metoda dekompozycji działań opiera swoją poprawność na dwóch przesłankach:

- statystycznym wyrównywaniu błędów
- znacznie większej potencjalnej dokładności estymacji małych zadań w porównaniu z dużymi.

Jest ona powszechnie stosowana w procesie tworzenia harmonogramów projektów. Metody dekompozycyjne wspierane są całym szeregiem technik: strukturyzacja prac, harmonogramowanie (wykresy Gantta) i sieciowe (CPM, PERT, GAN). Większość technik jest przełożona na odpowiednie narzędzia komputerowe. Narzędzia te jednak wymagają danych wejściowych: struktury prac (i ich szeregowanie) i estymacji parametrów poszczególnych zadań. Wspomagają więc pracę tylko instrumentalnie. Problem szacowania przenoszony jest tylko na niższy poziom.

Metody ekstrapolacyjne polegają na budowie empirycznych modeli, pozwalających oszacować różnorodne parametry projektów informatycznych na podstawie atrybutów produktywności. Modele są opracowywane na podstawie danych empirycznych. Uogólniają więc doświadczenia zbierane w trakcie realizacji wielu projektów.

Do najbardziej znanych modeli parametrycznych należą:

- modele „linii kodu";
- metoda punktów funkcyjnych (ang. *FP - Function Points*) Allana Albrechta (Albrecht: 1979);
- model COCOMO (ang. *COnstructive Cost Model*) Barrego Boehma (Boehm: 1981).

Metody ekstrapolacyjne są zwykle stosowane na początkowych etapach projektu i wykorzystują różne modyfikacje wzoru:

$$Z = Z_1 W^{\beta} + Z_0$$

gdzie:

W - miara wielkości projektu („linia kodu", punkt funkcyjny itp.);
Z_1 - jednostkowe zużycie zasobu;
Z_0 - stałe zużycie zasobu;
β - współczynnik skali.

5. Metoda Punktów Funkcyjnych

Metoda Punktów Funkcyjnych została opracowana poprzez uogólnienie badań wielu projektów informatycznych przez długoletniego pracownika firmy IBM Allana J. Albrechta w 1979 roku. Została ona zmodyfikowana przez twórcę w 1983 i rozwinięta przez International Function Point Users Group (IFPUG) w latach 1986, 1988, 1990 i 1994.

Allan Albrecht wydzielił pewne parametry systemu informatycznego i uznał je za wielkości charakteryzujące wielkość (złożoność realizacyjną) systemu. Parametry te były wybrane w taki sposób, by można je było określić i oszacować dla nieistniejącego, projektowanego systemu. Metoda Punktów Funkcyjnych szacuje wpływ każdego

z parametrów na wielkość systemu i nakłada na tę ocenę szacunek wpływu warunków realizacji projektu informatycznego. Wyznaczone w ten sposób punkty funkcyjne są traktowane jako miara produktywności zespołu realizacyjnego i równocześnie jako miara złożoności systemu informatycznego, a w konsekwencji - pracochłonności projektu. Należy podkreślić fakt, że projekt jest oceniany na wstępnych etapach z uwzględnieniem warunków jego późniejszej realizacji. Metoda Punktów Funkcyjnych jest zatem bardzo cenną z punktu widzenia szacowania pracochłonności projektów. Jej idea może zostać w prosty sposób rozszerzona na szacowanie innych, niż pracochłonność, zasobów.

Metoda FPs umożliwia oszacowanie złożoności projektu informatycznego z uwzględnieniem dwóch różnych grup czynników:

- Elementów przetwarzania informacji w systemie (w aspekcie funkcjonalnym i informacyjnym). Elementy te to: wejścia i wyjścia z systemu, zbiory danych wewnętrznych i zewnętrznych i zapytania do bazy danych. Łączy się je w grupy o oszacowanych rozdzielnie wagach i sumuje ich liczności z uwzględnieniem tych wag. Ta ważona suma jest tzw. nieskorygowanymi punktami funkcyjnymi (ang. *UFP - Unadjusted Function Points*).
- Całościowej oceny stopnia złożoności systemu informatycznego i warunków jego realizacji w postaci różnorodnych czynników korygujących składających się na kompleksowy współczynnik korygujący (ang. *VAF - Value Adjustment Factor*). Czynniki korygujące określa się dla systemu jako całości, bez wnikania w jego wewnętrzną strukturę.

Do elementów przetwarzania A. Albrecht zaliczył:

- wejścia użytkownika,
- wyjścia użytkownika,
- zbiory danych wewnętrzne,
- zbiory danych zewnętrzne,
- zapytania zewnętrzne do systemu.

Ich wpływ na wielkość projektu informatycznego różni się znacznie: od 3 (proste wejście lub proste zapytanie) punktów do 15 (zbiór danych o skomplikowanej strukturze wewnętrznej i licznych powiązaniach w ramach systemu).

Wyznaczone w powyższy sposób nieskorygowane punkty funkcyjne podlegają modyfikacjom, uwzględniającym wpływ innych parametrów projektu. Wpływ każdego z czternastu czynników może być różny w danym projekcie. Jest on subiektywnie szacowany w skali sześciostopniowej: 0, 1, 2, 3, 4, 5. Im większy, bardziej znaczący wpływ danego czynnika tym wyższa jego ocena punktowa. Łączny wpływ czynników korygujących uwzględnia ***kompleksowy współczynnik korygujący***, który jest prostą sumą oszacowań wpływu każdego czynnika.

Punkty funkcyjne pierwotnie służyły do wyznaczania pracochłonności projektów informatycznych wg empirycznie uzyskanych zależności.

Na bazie idei punktów funkcyjnych opracowywane są inne metody. Jedną z nich jest Metoda **Punktów Charakterystycznych** (ang. *FP - Feature Points*). Metoda ta powstała dla systemów operacyjnych, systemów czasu rzeczywistego, oprogramowania telekomunikacyjnego, przełączającego, sterującego, symulacyjnego itp. Zakłada ona dołączenie algorytmów do listy elementów charakteryzujących przetwarzanie danych w systemie i modyfikację wag poszczególnych parametrów. Dla systemów informatycznych zarządzania (SIZ) obie metody dają praktycznie te same rezultaty. Wynika to ze znikomego wpływu złożoności algorytmów na pracochłonność projektu SIZ. Rozbieżności pojawiają się w przypadku systemów, w których algorytmizacja odgrywa dużą rolę (tab.).

Współzależność punktów funkcyjnych i charakterystycznych

Typ aplikacji	Punkty funkcyjne	Punkty charakterystyczne
Wsadowe SIZ	1	0,80
Interaktywne SIZ	1	1,00
Systemy telekomunikacyjne	1	1,20
Systemy kontroli	1	1,28

Wbudowane systemy czasu rzeczywistego	1	1,35
Automatyka przemysłowa	1	1,50

Z innych metod pokrewnych FPs należy zwrócić uwagę na: MKII i 3D FPs. Ta ostatnia metoda zakłada trójwymiarowy przekrój przez projekt informatyczny: dane, funkcje i sterowanie.

6. Niekonwencjonalne zastosowanie metod estymacji zasobów

W trakcie opracowania metod estymacji zasobów powstały różnego typu metryki umożliwiające pomiar wielkości systemów informatycznych. Metryki te znajdują szerokie zastosowanie w:

- ocenie produktywności (a więc w pewnym sensie jakości narzędziowej) poszczególnych języków programowania;
- auditing projektów;
- wybór systemów informatycznych, funkcjonujących w przedsiębiorstwie do re-engineeringu;
- szacowanie liczby (i objętości próbek) testów;
- ocenę i porównanie jakości pracy i wydajności zespołów ludzkich;
- ocenę wielkości zmian, wprowadzanych przez użytkownika na poszczególnych etapach budowy systemu informatycznego (wymaga ciągłego przeliczania punktów funkcyjnych, ale dokładnie ocenia zakres i, zatem i koszt, modyfikacji);
- prognozowanie kosztów pielęgnacji i rozwoju systemów;
- porównanie i ocenę różnych ofert dostawców oprogramowania pod kątem merytorycznym i kosztowym (np. koszt jednego punktu).

7. Podsumowanie

Oszacowanie pracochłonności projektu informatycznego jest w gruncie rzeczy sztuką, której uczy się kierownik projektu w trakcie długoletniej praktyki. Wspomaga go w tej sztuce znajomość metod i technik, uświadomienie sobie zagrożeń i błędów innych, a także opracowanie, wdrożenie i eksploatacja systemu zbierania danych o parametrach zrealizowanych projektów.

Sztukę tą należy stosować w praktyce. Umożliwia ona bowiem zmniejszenie ryzyka związanego z realizacją projektu informatycznego. Jednocześnie, za jednym z twórców inżynierii oprogramowania Tomem DeMarco należy podkreślić, że „...nie ma przenośnych modeli kosztów...". By zwiększyć dokładność szacowania różnych parametrów projektu (w tym i pracochłonności) niezbędne jest poniesienie nakładów na stworzenie systemu rejestrowania danych o zrealizowanych projektach i wykorzystywania tej informacji w praktycznej działalności.

Literatura

1. Abran, P.N.Robillard, Function Points Analysis: An Empirical Sudy of its Measurement Processes, IEEE Transactions on Software Engineering, vol. 22, no. 12, 895-909, 1996.

2. Baran, A., P.N. Robillard, Function Point Analysis: An Empirical Study of its Measurement Processes, Montreal, Univ. of Quebec, 1996. (http://saturne.info.uqam.ca/labo-recheche/Lrgl/publi/rjournal

3. A.J. Albrecht, Measuring Application Development Productivity, Proceedings of the Joint SHARE/GUIDE Application Development Symposium, Chicago, GUIDE International Corp., 1979.

4. Czarnocka-Chrobot, Metoda FUNCTION POINTS, Informatyka, 5, 1994, 19-24.

5. Miłosz, M. Miłosz: Estymacja pracochłonności w projektach informatycznych. Informatyka Stosowana, numer specjalny S3/97, str. 37-54. VI Konferencja Polskiego Klubu Użytkowników Systemu Progress. Kazimierz Dolny, 19-22.10.97

6. Yourdon, Współczesna analiza strukturalna, Warszawa, WNT, 1996.

7. F.P., Jr. Brooks, The mythical man-month: Essays of Software Engineering; NY, Addison-Wesley Pub. Co., 1995.

8. H. Kerzner, Project Management: A System Approach To Planning, Scheduling and Controlling; NY, Van Nostrand Reinhold Company, 1984.

9. Capers, What Are Function Points?, Software Productivity Research, 1995 (http://www.spr.com)

10. Putman, A. Fitzsimmons, Estimation Software Costs, Datamation, v. 9, 1979, 89-98.

11. Miłosz: Szacowanie zasobów w projektach informatycznych Informatyka, nr 11, 1997, str. 15-22

12. M.Maya, A.Abran: Function Points Adapted to Small Maintenance Requests. IFPUG Fall Conference, Toronto, 1995.

13. Programming Languages Table, Software Productivity Research, 1996 (http://www.spr.com/library/langtbl.html).

14. Software Measurement Guidebook. Revision 1, June 1995; Software Engineering Laboratory Series, Greenbelt, Goddard Space Flight Center, NASA, 1995.

15. T. DeMarco, Controlling Software Projects, NY, YOURDON Press, 1982.

Jacek Królik

Techniki estymacji - metoda punktów funkcyjnych a szacowania eksperckie

Metryki oprogramowania, w szczególności metody szacowania pracochłonności oraz czasu realizacji przedsięwzięć informatycznych, to temat nienowy i nieustająco modny od kilkudziesięciu lat.

Moda ta ma wiele przyczyn. Pierwsza jest taka, iż przyjemnie jest wymyślać metryki oraz metody szacowania, a następnie poddawać je weryfikacji eksperymentalnej. Jest to stosunkowo wdzięczny i niewyczerpany temat badawczy, toteż wielu badaczy zajmuje się nim z ochotą. Inna przyczyna jest taka, że jest to zagadnienie niewątpliwie ważne.

Studiując literaturę przedmiotu oraz czasopisma poświęcone zarządzaniu projektami odnoszę czasem wrażenie, iż mam do czynienia z „operacją poszukiwania świętego Graala ", czyli Uniwersalnej Metody Szacowania Wszystkiego. Poszukiwaniem owego przedmiotu kultu zajmują się głównie zawodowi poszukiwacze, niemniej jednak istnieje pokaźna grupa wiernych, którzy uważają, iż Metoda została już znaleziona lub jesteśmy o krok od Jej znalezienia.

Rzeczywiście prawdą jest, iż opracowano kilka metod szacowania, które wyglądają całkiem „ładnie" i dają nadzieje na uporządkowanie zagadnienia szacowania raz na zawsze. Metody te są nierzadko bardzo już zabytkowe: COCOMO, WICOMO, Bang; są też i nowocześniejsze, znajdujące coraz więcej wyznawców, jak metoda Punktów Funkcyjnych (FPA). W przypadku tej ostatniej doszło nawet już do schizmy, toteż występuje ona w co najmniej dwóch wersjach: standardowej, głoszonej przez IFPUG oraz heretyckiej, zaproponowanej przez Symonsa (FPA Mk II).

Dla przykładu spróbujmy się zatem przyjrzeć metodzie Punktów Funkcyjnych, bez wnikania w szczegóły, czyli określania dokładnie, o który wariant tej metody chodzi.

Metoda szacowania pracochłonności oparta na FPA jest szalenie prosta i składa się z kilku kroków:

1. Obliczenie wielkości systemu wyrażonej w umownych jednostkach, zwanych punktami funkcyjnymi. Obliczenie to odbywa się według pewnej zdefiniowanej procedury. Procedura polega na tym, że analizuje się specyfikację programu, którego pracochłonność wykonania szacujemy, a następnie poszczególnym elementom tej specyfikacji przyporządkowuje się punkty funkcyjne, zgodnie z zaleceniami metody. Po przyporządkowaniu miar wszystkim istotnym elementom specyfikacji, należy te miary zsumować, w ten sposób otrzymuje się wielkość systemu wyrażoną w punktach funkcyjnych.
2. Następnie tak obliczoną wielkość systemu należy skorygować o ± 35%, co wykonuje się stosując kolejną technikę. Polega ona na przejrzeniu listy kilkunastu czynników modyfikujących wstępne oszacowanie i obliczeniu na ich podstawie współczynnika złożoności technicznej, który jest liczbą z zakresu 0,65...1,35. Po przemnożeniu tego współczynnika przez liczbę punktów funkcyjnych otrzymaną w poprzednim kroku otrzymujemy finalne oszacowanie wielkości systemu wyrażone w punktach funkcyjnych.
3. Dysponując liczbą punktów funkcyjnych musimy dokonać teraz przełożenia wielkości systemu na pracochłonność jego wykonania. Zazwyczaj wykonuje się to

znając przeciętną produktywność personelu, określającą ilość punktów funkcyjnych wykonywanych miesięcznie przez jednego pracownika, np. może to być 0,5 p.f. miesięcznie.

W ten sposób dzieląc liczbę punktów funkcyjnych przez przeciętną produktywność otrzymujemy oszacowanie pracochłonności wykonania systemu.

Proste? Bardzo. Przyjrzyjmy się jednak bliżej tej metodzie. Zwłaszcza zastanówmy się, czy jest ona rzeczywiście obiektywna oraz czy jest ona w zupełności wystarczająca.

Po pierwsze zatem zastanówmy się nad obiektywizmem tak otrzymanego szacowania. Gołym okiem widać, że w tak przedstawionej metodzie istnieją trzy miejsca, które opierają się na ocenie subiektywnej dokonywanej przez osobę szacującą:

- Przydział punktów funkcyjnych elementom specyfikacji.
- Obliczanie współczynnika korygującego.
- Przyjęcie współczynnika produktywności, na podstawie którego dokonuje się oszacowania pracochłonności.

Jaki stąd wniosek? Otóż taki, że metoda FPA nie nadaje się do użycia! Ściślej mówiąc nadaje się, tylko po jej wdrożeniu, czyli zrozumieniu dostosowaniu do norm produktywności zespołu pracującego nad różnymi projektami w różnych sytuacjach.

Ponadto zauważmy, że szacowanie metodą FPA, to tylko część problemu. A co z oszacowaniem ryzyka? Jaki jest wpływ dziedziny problemu na pracochłonność przedsięwzięcia? Jaka jest produktywność naszego zespołu w „takich" przedsięwzięciach, dla „takich" klientów, w „takim" środowisku technicznym? Jakie były pracochłonności realizacji innych „podobnych" przedsięwzięć? Czy oszacowania wykonywane innymi metodami dają podobne wyniki? Okazuje się, że szukanie odpowiedzi na te pytania może doprowadzić do skorygowania oszacowań uzyskanych metodą FPA o 50 lub nawet o 100 procent.

Stąd pytanie: czy oszacowanie uzyskane w taki sposób jest oszacowaniem obiektywnym, czy też subiektywnym? Wydaje się, że podpieranie się „pełnym obiektywizmem" „formalnej" metody jest tu zwykłym nadużyciem.

Kolejną rzeczą, którą chcę uświadomić uczestnikom mojego wykładu jest fakt, że każdy zespół musi sobie swoją metodę szacowania „wyhodować". Hodowla powinna wziąć swój początek od jakiejś metody standardowej, np. FPA, niemniej jednak to dopiero początek. Dalsze kroki polegają na zastosowaniu klasycznego podejścia Process Management'owego, zorientowanego na doskonalenie metody szacowania, opartego na doświadczeniach. Doświadczenie owo to zarówno wiedza ekspertów, którzy realizowali przedsięwzięcia informatyczne oraz brali udział w ich szacowaniu. Jest ono także rezultatem analizy danych o pracochłonności zgromadzonych w przedsięwzięciach już zrealizowanych.

W takim ujęciu na pierwszą pozycję wysuwają się techniki szacowania oparte na doświadczeniu ekspertów oraz na uśrednianiu wyników przedsięwzięć poprzednich, np. metodą wyroczni delfickiej WAVE (ang. weighted average). Oczywiście metody te muszą by stosowane w powiązaniu z metodami formalnymi, takimi jak opisana wcześniej FPA. Wydaje się, że dopiero powiązanie tych podejść pozwala na uzyskiwanie wyników wiarygodnych, opartych zarówno na najlepszych praktykach inżynierii oprogramowania, jak również na doświadczeniu i wiedzy zespołu dokonującego oszacowania.

Podsumowując powyższe rozważania możemy zatem stwierdzić:

- Metody formalne, takie jak wspomniana metoda punktów funkcyjnych, nie są wolne od subiektywizmu. Ponadto są one zazwyczaj tylko częścią procesu szacowania, na który składają się różne techniki szacowania, a tylko niektóre z nich są sformalizowane i to nie zawsze w całości.

- Realistyczne szacowanie to zawsze proces polegający na weryfikacji tego, co obiektywnie wiemy o przedsięwzięciu z tym, co subiektywnie wiemy o jego otoczeniu i możliwościach wykonawczych zespołu, technologii produkcji, kliencie, temacie, etc.
- WAVE tylko na pierwszy rzut oka wygląda jak metoda wygrywania w ruletkę. W rzeczywistości może to być kluczowy element wiarygodnego procesu szacowania przedsięwzięcia.
- Stosowanie WAVE wychodzi poza kontekst pojedynczego przedsięwzięcia. Metoda ta może być również jedną z kluczowych praktyk doskonalenia metod szacowania.

Sceptyków i Wyznawców UMSW zapraszam do dyskusji na moim wystąpieniu w Szczyrku.

Marian Kuraś, Zbigniew Nęcki

Zakład Zastosowań Informatyki
Akademii Ekonomicznej w Krakowie
ul. Rakowicka 27, PL 31-510 Kraków
e-mail: eikuras@cyf-kr.edu.pl

Instytut Psychologii
Uniwersytetu Jagiellońskiego
ul. Gołębia 4; Kraków
e-mail: zbign@apple.phils.uj.edu.pl

Psychologiczne aspekty komunikacji z użytkownikiem

Streszczenie: W referacie omówiono podstawy komunikacji między użytkownikami a informatykami. Przedstawiono typowe zakłócenia i ich przyczyny natury psychologicznej. Celem referatu jest uświadomienie i umożliwienie lepszego zrozumienia niektórych błędów popełnianych w kontaktach z użytkownikami a także w środowisku zawodowym. Do przygotowania referatu wykorzystano bogate własne obserwacje oraz literaturę przedmiotu. Przedstawiono proste rady dla informatyków i użytkowników, jednak nacisk położono na potrzebę rozwijania umiejętności sprawnego, skutecznego i efektywnego porozumiewania się.

Wstęp

Umiejętność komunikowania się, jakkolwiek często lekceważona lub niedostrzegana, jest podstawową umiejętnością nieodzowną do nawiązania i utrzymania współpracy między twórcami systemów informacyjnych (SI) a ich użytkownikami. Tworzone nowe rozwiązanie ma w większości przypadków umożliwiać usprawnianie komunikacji, zatem musi odpowiadać potrzebom użytkowników. Warunkiem tego jest sprawna i efektywna komunikacja między twórcami SI i jego użytkownikami.

Problemów komunikacji przez lata nie dostrzegano, gdyż żyliśmy w 'dobrze ustrukturyzowanym świecie', gdzie określono a często skodyfikowano wszelkie zachowania organizacyjne w tym również komunikacyjne. Formalizacja więzi międzyludzkich była przyczyną zakłóceń komunikacji, co z kolei powodowało spadek sprawności organizacji. Oczekiwania poprawienia komunikacji organizacyjnej w wyniku wprowadzania techniki informacyjnej były całkowicie bezpodstawne. TI a szczególnie komputery opierają się na formalizacji i służą jej, jeśli w organizacji oczekuje się jej. Obserwując praktykę organizacyjnych zastosowań TI łatwo jest zauważyć, iż tak użytkownicy (menedżerowie) jak i informatycy nie rozróżniają komunikacji i informacji. Obydwa pojęcia, jakkolwiek mające wspólne obszary, są odmienne i nie mogą być traktowane jako synonimy (Sanderlin: 1982).

Dostrzeżenie tematyki komunikacji należy wiązać ze zmianami politycznymi i gospodarczymi. W warunkach ustroju demokratycznego i gospodarki rynkowej sprawne komunikowanie się przez jednostki i grupy społeczne staje się warunkiem funkcjonowania w życiu społecznym i w gospodarce. Potrzeby te ujawniły się w zarządzaniu i w zastosowaniach TI (np. Kuraś: 1991; Kuraś, Zaliwski: 1993). Na rynku pojawiły się niezliczone podręczniki i poradniki, które dobrze sprzedają się, jednak nie zawsze pozwalają dowiedzieć się, czym jest komunikacja i jak lepiej komunikować się w sytuacjach zawodowych. Nieliczne są opracowania porządkujące i wyjaśniające istotę problemów komunikowania się (np. Nęcki: 1996).

Celem referatu adresowanego do informatyków jest zwrócenie uwagi na psychologiczne tło i uwarunkowania komunikacji oraz możliwości poprawy komunikacji w procesie tworzenia SI.

1. Komunikacja międzyludzka (jako czynnik powodzenia zamierzeń informatycznych)

Rozumienie pojęcia komunikacji jest bardzo zróżnicowane. Zależnie od zawodu i pełnionych funkcji inaczej widzi się proces komunikowania się i inne przypisuje się jej znaczenie. Spośród wielu definicji komunikacji chcemy zwrócić uwagę na trzy, które najlepiej odpowiadają potrzebom rozważań.

Pierwszą, najpowszechniej uznawaną, jest definicja cybernetyczna. Zgodnie z nią komunikowanie jest **wymianą (przepływem) informacji od nadawcy do odbiorcy**. Proces komunikacji w tym ujęciu uwzględnia *KTO CO JAK KOMU* i z jakim *SKUTKIEM* przekazuje. Ważną cechą jest uwzględnienie sprzężenia zwrotnego, zaś istotą jest redukcja niepewności i uzgodnienie danych.

Definicja cybernetyczna dobrze opisuje komunikacje komputerów ale nie ludzi. Słowa i sekwencje wypowiadane przez ludzi mają odmienne znaczenia dla każdej jednostki i tylko dla komputerów są jednoznaczne. W modelu tym nie uwzględnia się kontekstu (subtekstu), odrywa się akt komunikacji od kontekstu, w którym zachodzi.

Przykład: zarobiłem bardzo dużo (ile to jest: dużo?), *to jest tani produkt* (co to znaczy dla: małej firmy, wielkiego przedsiębiorstwa, osoby indywidualnej no i twórcy / sprzedawcy)

Podstawy drugiej definicji daje teoria relewancji. Komunikacja jest określana jako **intencjonalna modyfikacja przestrzeni w polu percepcyjnym odbiorcy**. Zakłada się, że każdy akt komunikacyjny jest zagadką, którą trzeba rozwikłać, przy czym są zagadki łatwe (np. *jutro przyjdę do pracy*) i trudne (np. *to piękne dzieło myśli ludzkiej*). Informatyk, menedżer, klient i jego rodzina tworzą świat zagadkowych zachowań, które odbiorcy muszą rozwikłać.

Przykłady: Każdy przepływ prądu szkodzi zdrowiu! Szkodzi czy nie? Czemu szkodzi? W jakim stopniu? Jakie są skutki? Jak można im zapobiegać? *Komputery zawsze usprawniają pracę!*. Co to znaczy, że usprawniają? Jakie komputery? Po co do nich programy? Jaki jest tego koszt?

Definicja pragmalingwistyczna (Awdiejewa-Nęckiego) odwołuje się do procesów wymiany symboli w celu koordynacji działań. To nie informację wymienia się a symbole, znaki, słowa, gesty. Trzy główne rodzaje to symbole werbalne (*zwykłe słowa*), wokalne (*donośność, intonacja*) oraz symbole niewerbalne (*przedmioty, gesty, postawa*). Sprawdzianem końcowym efektywnej komunikacji jest określone lepsze współdziałanie uczestników procesu komunikacji (nadawcy i odbiorcy) a nie enigmatyczna zgodność poglądów i przekonań. Można to obserwować, mierzyć i weryfikować w codziennej praktyce działania.

Ostatnie z wymienionych podejść jest podejściem praktycznym. Zwraca uwagę na praktyczny efekt – lepsze dopasowanie działań stron – co jest celem każdego aktu komunikacyjnego. Resztę uważa się za uzupełnienie tworzące tło i obszar pod przyszłe procesy komunikacyjne, których celem oczywiście będzie sprawniejsze i bardziej efektywne działanie.

Przedstawione definicje komunikacji w zasadzie wyczerpują nasze potrzeby związane z potrzebami oceny praktyki, poznania błędów i ich przyczyn oraz przemyślenia możliwości poprawy praktyk komunikowania się. Osobom pragnącym zgłębić tajniki komunikowania się można doradzić liczne poradniki oraz poważniejsze opracowania jak np. (Nęcki: 1995).

2. Komunikacja między twórcami systemu informacyjnego a użytkownikami

Dostrzeganie i uznanie komunikacji jako kluczowego czynnika powodzenia w tworzeniu i wykorzystaniu techniki informacyjnej jest efektem zmian w zarządzaniu i wymagań użytkowników a także wzrostu konkurencji na rynku usług informatycznych. Mimo wyraźnych zmian można twierdzić, że wciąż niesprawna i nieskuteczna komunikacja jest przyczyną niepowodzeń zamierzeń informatycznych. Niesprawna komunikacja wynika z braku warsztatu i wiedzy, przekonania o swojej wyższości bądź z nie w pełni uczciwego traktowania partnera.

Zrozumienie sytuacji i problemów użytkownika, własnej roli jako promotora innowacji oraz uczenia się organizacji jest warunkiem skutecznej współpracy. Dodajmy do tego właśnie

umiejętność komunikowania się i uzyskamy warunki powodzenia zamierzenia zależne właśnie od twórców SI. Sytuacje, w jakich dochodzi do kontaktów między informatykami a użytkownikami są niestabilne i potencjalnie konfliktowe. Takie sytuacje wymagają nasilenia komunikacji i jej usprawnienia. Unikanie komunikowania się, jej formalizowanie i redukowanie kontaktów jak też ich zastępowanie np. przez lekturę przepisów uniemożliwia uczenie się tak przez informatyków jak i użytkowników. Do powodzenia zamierzenia konieczne jest wzajemne zaufanie obydwu grup uczestników. W szczególności informatycy potrzebują zaufania użytkowników. Można je uzyskać jedynie poprzez udaną komunikację. Informatycy w szczególności powinni nauczyć się słuchać użytkowników, by rozumieć sytuacje i problemy użytkowników. Muszą zrozumieć, że bez słuchania mogą narazić się na śmieszność występując z kategorycznymi sądami na tematy lepiej znane przez użytkowników. Powinni też zrozumieć, że samo przekazanie i trzymanie informacji od użytkownika nie jest komunikowaniem się (Sanderlin: 1982).

Proces projektowania oparty na tak ograniczonych informacjach wyklucza skuteczne rozwiązanie problemów. Jeśli problemów nie rozpozna się bądź nie uświadomi poprzez celową komunikację nie można ich rozwiązać. Istnieje ryzyko, że stworzony system informatyczny będzie postrzegany jako narzędzie informatyka a nie potrzebne i przystosowane do potrzeb wspomaganie użytkownika. Jeśli twórcy systemu nie komunikują się (nie poznają potrzeb, nie potrafią się wczuć, zrozumieć sytuacji użytkownika), nie mogą liczyć na pełną akceptację swojego rozwiązania. Bardzo często nie dowiedzą się nawet, że użytkownicy nie zaakceptowali go.

3. Powszechne trudności i błędy w komunikowaniu się

Podstawowe błędy w komunikowaniu się dotyczą wszystkich relacji informatyka z otoczeniem społecznym czyli w kontaktach z klientem zewnętrznym, klientem wewnętrznym (współpracownikami z innych komórek) i w kontaktach hierarchicznych (z przełożonymi i podwładnymi). (s.183) Trzeba podzielić je na cztery typy: Typ pierwszy najbardziej oczywisty jest związany z aktami komunikacyjnymi, których celem jest koordynacja działań rozmówców – nadawcy, odbiorcy. Chodzi tu o strategie pragmatyczne takie jak: *polecenie, rada, prośba, obietnica* i *propozycja*. Każdy z tych pięciu aktów ma liczne obwarowania: kulturowe i interpersonalne. Przykładowo *polecenie* nie może być kierowane przez podwładnego do przełożonego.

Rada niespecjalisty kierowana do specjalisty wywołuje co najwyżej uśmiech politowania. Jakże często uśmiech ten pojawia się w środowisku informatyków. Również *prośba* jest 'niebezpiecznym' komunikatem, gdyż w wypadku odmowy jej spełnienia następuje kłopotliwy kryzys (np.: *pożycz mi swoje auto*). *Prośba* o poświęcenie czasu jest niezręczna w tak zabieganym świecie, jaki tworzą informatycy (np. *czy możesz poświęcić mi 'chwilę' i pokazać jak działa* ...). Należy zwracać uwagę na to, co obiecujemy zarówno kolegom jak i klientom. Nierealistyczne *obietnice*[1], których nie sposób dotrzymać wywołują uzasadnione pretensje (obiecałeś to zrobić na dziś ...). Ostatni problem to komunikowanie *odmowy*. Jest ona przykrym doświadczeniem dla obydwu stron i dlatego powinno się odmawiać szczególnie delikatnie, rzeczowo i z wyczuciem. Przykrość usłyszenia '*to się nie da*' trzeba dla dobra relacji uzasadnić racjami obiektywnymi, systemowi i normatywnymi przedstawionymi w sposób osobisty. Najbardziej drastyczna forma *odmowy* to stwierdzenie, że rozmówca nie ma ochoty tego zrobić, choć ma możliwości. Nawet podejrzenie, iż ktoś kieruje się w odmowie niechęcią a nie niemożliwością, budzi spore napięcia (*nie ma podwyżki, nagrody, bo nie*).

Druga grupa trudności to problem uzgadniania wartości i ocen np.: co jest dobre a co złe dla firmy i dla pracownika? Co to jest sukces informatyka / firmy? Co ważniejsze -- zysk czy realizacja twórczych nowatorskich rozwiązań? Jak ocenić (wycenić) szkodliwość pracy na skomputeryzowanym stanowisku? *Rozwiązanie ma być przede wszystkim skuteczne, efektywne czy eleganckie? Kto jest ważniejszy informatyk czy klient, programista czy projektant ... wdrożeniowiec czy sprzedawca?* Jak wyceniać rożne prace? Wiele transakcji komunikacyjnych ma na celu ustalenie hierarchii potrzeb, ważności, przydatności, użyteczności, opłacalności.

[1] Obietnice sprawdzalne (*zrobię to na jutro rano*) i niesprawdzalne (*zrobię to możliwie szybko*).

Drastycznym przykładem rozbieżności w jest próba ustalenia, która religia czy opcja polityczna jest lepsza. Możne to prowadzić do wojen, których celem jest ustanowienie wyższości jednej grupy i jej kultury nad drugą. Przyczyną rozpadu firmy mogą być rozbieżności w ocenie wartości takich jak np. uczciwość, solidność, traktowanie klientów. Rozdźwięki powstają na tle wykorzystania narzędzi, platform i zasad współpracy. Nie do pojęcia bywa akceptacja a wręcz afirmacja działań hackerów i całkowite pomijanie jej skutków.

Trzeci typ trudności dotyczy uzgodnień obrazu świata – wiedzy i przekonań: co istnieje a co nie. *Czy istnieje diabeł komputerowy. Co to jest przestrzeń wirtualna. Czy kosmici bywają na ziemi? Czy jest możliwe 'biuro bez papieru', totalna automatyzacja. Czy roboty mogą się samo-programować uczyć? Czy istnieje sztuczna inteligencja i czym w ogóle jest?* Realnym problemem jest uzgodnienie prawdopodobieństwa zdarzeń – czy zamierzenie ma szanse powodzenia? Czy użytkownicy potrafią wykorzystać (zaawansowaną) TI? Wobec trudności komunikacyjnych zdarza się ukrywanie ról i specjalizacja w komunikowaniu się -- informatyka, który wie wszystko, musi reprezentować wobec klienta sprzedawca i wdrożeniowiec. W początkowych fazach tworzenia SI -- wobec trudności i braku kompetencji -- rezygnuje się z uzgadniania wiedzy o organizacji, jej możliwościach i ograniczeniach oraz perspektyw wykorzystania sprzedawanej technologii.

Czwarty i ostatni blok podstawowych trudności – być może najciekawszy – dotyczy organizacji samego procesu porozumiewania się. Organizacja procesu wymaga dwu rodzajów działań: (1) dotyczących przebiegu kontaktu (*kto ma mówić, a kto słuchać*) oraz (2) niezgodność kodu komunikacyjnego. W kontakcie komunikacyjnym generalnie zakłada się, że wypowiedzi powinny następować naprzemiennie. W praktyce osoby ważne (np. informatycy :-) w ogóle nie lubią słuchać. Zabieranie głosu innym (*inwazja*) lub nazbyt długie milczenie to częste przykłady zakłóceń komunikacyjnych natury organizacyjnej.

Jeszcze ważniejszym problemem metakomunikacyjnym jest rozbieżność kodu językowego. To, co jest oczywiste dla informatyka, dla osób spoza branży jest niezrozumiałe.

Przykłady: ZN w sklepie komputerowym 'O' dowiedział się tylko tyle że sprzedawca jest bardzo mądry i wszystko wie o komputerach i ich konfigurowaniu.

Prezes N usłyszał od informatyków z firmy S wszystko oprócz tego, czego chciał się dowiedzieć na temat wspomagania pracy zespołowej. Wie teraz dokładnie, jaki jest mu potrzebny sprzęt i oprogramowanie oraz jakie są wymagania wobec sieci. Wciąż jednak nie rozumie na czym ma polegać zmiana po wprowadzeniu nowego wyposażenia i jakie osiągnie korzyści.

Wykorzystanie specjalizowanego kodu zdarza, wtedy gdy jedna z grup uczestników nie zna problemów partnerów i chce zyskać przewagę. Język analityka/projektanta czy technologa może być wykorzystywany do zrobienia wrażenia i wykazania wyższości nad użytkownikami. W kontaktach należy też zwracać uwagę na poziom wypowiedzi, który może być poważny, żartobliwy lub ironiczny. Zadaniem rozmówcy jest rozróżnienie tonu i stosowne reagowanie. Żartobliwe przezwisko, na przykład, wymaga uśmiechu ale wypowiedziane na serio staje się poważną obrazą. Ironiczna reakcja informatyka na poważną a nieporadną wypowiedź użytkownika może spowodować wycofanie tego ostatniego i unikanie w dalszych kontaktach dzielenia się swoją wiedzą i wątpliwościami.

W zakończeniu omówienia źródeł trudności i zakłóceń w komunikacji należy wspomnieć takie nieprawidłowości jak: brak spójności między wypowiedziami rozmówców (A: *Kiedy zrobicie poprawki w programie ...?* B: *Musicie koniecznie zmienić wersję sterownika*), wypowiedzi uwłaczające, poniżające, wypowiedzi wewnętrznie sprzeczne (*szkolenie powinno być angażujące, barwne, ciekawe, emocjonujące, inspirujące, interesujące, praktyczne, przekonywujące, urozmaicone, zajmujące, żywe ...*), pomijanie kontekstu wypowiedzi. Proces komunikacji może być zakłócany przez takie czynniki jak: nieprawidłowa globalna i lokalna organizacja rozmowy, niewerbalne zachowania rozmówcy, nieporozumienia słowne i próby rekonstrukcji rozmowy, wieloznaczne akty komunikacyjne i niepoprawne używanie zaimków oraz zakłócenia środowiskowe. Szczegółowe omówienie źródeł trudności i zakłóceń komunikacji można znaleźć w pracy (Nęcki: 1996). W licznych poradnikach

i opracowaniach można znaleźć również omówienie barier komunikacji np. (Kuraś: 1991).

4. Potrzeba i możliwości zwiększania skuteczności komunikowania się

Informatycy, którzy podejmują się misji usprawnienia procesów informacyjnych i komunikacji w organizacji, muszą w tym celu poprawić własne umiejętności komunikowania się. Jest to warunek konieczny skutecznej współpracy twórców i użytkowników SI. Brak tej komunikacji, jej zakłócanie i nieskuteczność są źródłami zagrożeń dla zamierzenia. Warunkiem jego powodzenia jest znajomość problemów i sytuacji użytkownika, który nie znając sposobu ich rozwiązania ani narzędzia, nie mogą być partnerami w tworzeniu nowych rozwiązań. Nowe rozwiązanie nie polega ponadto na napisaniu programów, zaprojektowaniu nieco innych dokumentów, ich obiegu oraz stworzenia oprogramowania i instrukcji posługiwania się nim. Informatycy często nie zdają sobie z tego sprawy zaabsorbowani misją dostarczenia oprogramowania. Nie rozumieją, że oprogramowanie może zastąpić pracownika, ale tylko w wykonywaniu prostych i dobrze ustrukturyzowanych prac. Zarządzanie jest jednak bardzo złożonym procesem, który ma na celu powodowanie ludźmi, by przyczyniali się do osiągania celów.

Nie uczyni tego zamiast ludzi żadna, nawet najnowocześniejsza, technologia. Jest ona konieczna do osiągania celów, jednak cele te – formułowane przez zarządzających -- należy upowszechnić i uzgodnić. Konieczna do tego jest sprawna komunikacja. Jeśli narzędzia mają służyć świadomym użytkownikom, informatycy muszą być świadomi celów, jakie wyznaczą menedżerowie i użytkownicy systemowi informacyjnemu. Do ich rozpoznania a także zinterpretowania potrzeb użytkowników konieczna jest komunikacja. Nie wystarczą odpowiedzi na pytania w ankiecie czy przejrzenie a nawet przestudiowanie dokumentacji i przepisów. Coraz częściej do zaproponowania satysfakcjonującego rozwiązania nie wystarcza udział samego bezpośredniego użytkownika a konieczny jest doradca mogący zasugerować nowe, możliwe (potrzebne) rozwiązanie organizacyjne. Ma ono prymat nad oprogramowaniem, którego zadaniem jest umożliwić dokonanie zmiany organizacyjnej.

Sprawność i efektywność komunikowania się uczestników procesu jest krytycznym czynnikiem powodzenia. Należy brać pod uwagę, że proces ten nigdy nie jest podobny jak opisy w literaturze. Zmianę, jakiej ma się dokonać wyobrażają sobie tylko nieliczni, podczas gdy pozostali obawiają się jej. Jeśli nie istnieje wizja stanu docelowego większość działań ma charakter przypadkowy i niekoniecznie przyczynia się do poprawy sytuacji. Stworzenie jej i upowszechnienie wymaga otwartej, intensywnej, sprawnej i skutecznej komunikacji.

W celu usprawnienia komunikacji w procesie tworzenia i wykorzystywania SI konieczne jest przede wszystkim zwiększanie świadomości potrzeb komunikowania. Trzeba tworzyć warunki do sprawnego komunikowania się. Pierwszym warunkiem jest podnoszenia umiejętności komunikowania się. Pozostałe dotyczą form organizacyjnych współpracy uczestników procesu oraz metodyki, techniki i narzędzia wspomagające komunikację.

W tablicy 1 przedstawiono podstawowe wskazówki komunikacji doskonałej. Można wykorzystać je do rozpoczęcia pracy nad sobą a wcześniej sprawdzenia i oceny własnych zdolności komunikowania się.

Tablica 1

Komunikacja doskonała

1. Zacznij obserwować swoich współpracowników przyjąwszy wcześniej założenie, że mogą oni wiedzieć coś ciekawego, czego warto od nich się nauczyć.
2. Poznaj samego siebie – swoje wady i zalety oraz silne i słabe strony.
3. Bądź uważny (i powolny) w formułowaniu ocen.
4. Pamiętaj o selektywności ludzkiej uwagi.
5. Bądź wrażliwy na uczucia innych ludzi, delikatnie odmawiaj ich prośbom.
6. Szanuj każde otrzymane pytanie – każde jest ważne.

6. Szanuj każde otrzymane pytanie – każde jest ważne.
7. Doceniaj korzyści płynące z różnicy zdań.
8. Zmieniaj się rolami ze swoim rozmówcą, postaraj się patrzeć na sprawę inaczej.
9. Unikaj udzielania rad, postawy znawcy wszystkiego.
10. Nie krytykuj, a jeśli już musisz, nie krytykuj ludzi a sprawy.
11. Bądź aktywnym słuchaczem i obserwatorem.
12. Mów krótko, jasno i rzeczowo nawet o skomplikowanych sprawach.
13. Okazuj szacunek dla partnera – to buduje partnerstwo, motywuje, aktywizuje konstruktywne nastawienie.

Źródło: opracowanie własne

Zakończenie

Komunikacja międzyludzka jest środkiem transferu wiedzy, przy czym wiedzy rozumianej jako umiejętność wykorzystania informacji do skutecznego i efektywnego działania (Senge i in.: 1994). Współczesna organizacja może przetrwać jedynie pod warunkiem uczenia się czyli zdobywania umiejętności przystosowywania się do zmian w otoczeniu. Z tej właśnie przyczyny oczekuje się od pracowników umiejętności sprawnego komunikowania się. Komunikacji nie można traktować jako synonimu informacji, nie można zatem oczekiwać, że tzw. komputeryzacja rozwiąże problemy organizacyjne firmy poprzez dostarczenie danych nawet w najdoskonalszej postaci.

Bibliografia:

Berne, E. (1987). W co grają ludzie? Psychologia stosunków międzyludzkich. Warszawa: PWN.

Kuraś, M.(1991). Problemy komunikacji z użytkownikami i wewnątrz zespołu. Informatyka. Nr 10 i 11

Kuraś, M., Zaliwski, A. (1993). Komunikacyjne aspekty tworzenia systemów informacyjnych. [w:] V Jubileuszowa Wyższa Górska Międzynarodowa Szkoła PTI. PTI-Oddział Górnośląski, Katowice 1993.

Myers, D.G. (1993). Social Psychology. 4-th Ed. New York: McGraw-Hill, Inc.

Nęcki, Z. (1996). Komunikacja międzyludzka. Kraków: Wydawnictwo Profesjonalnej Szkoły Biznesu.

Sanderlin. R. (1982). Information Is Not Communication. Business Horizons (March-Apr.).

Senge, P., Roberts, C., Ross, R.B., Smith, B.J., Kleiner, A. (1994). The Fifth Discipline Fieldbook. Strategies and Tools for Building a Learning Organization. New York: A Currency Book.

Sperber, D., Wilson, D. (1986). Relevance. Oxford.

Bogusław Jackowski

BOP s.c., Gdańsk
B.Jackowski@gust.org.pl

Mocne marki czyli zarządzanie dokumentacją: opis techniki

Robert M. Slade w recenzji książki Dale'a Dougherty i Arnolda Robbinsa, zatytułowanej „sed & awk", pisze:

> *Co jest Wielkim Problemem naszych dni? Rok 2000, rzecz jasna. Oznacza to, że należy wszystkie pola dwucyfrowe zamienić na pola czterocyfrowe – do tego wystarczy* `sed`.
>
> *A co oferują nam Wielkie Firmy Software'owe od lat piętnastu czy dwudziestu? Języki czwartej generacji. Cóż to takiego? Programy piszące raporty, tzn. programy, które czytają dane, wyszukują odpowiednie informacje i formatują je w sensowny sposób – coś jakby* `awk`, *nieprawdaż?*

Powyższy cytat w żartobliwy sposób sygnalizuje problem, który – jak widać – nie tylko w Polsce coraz bardziej daje się we znaki: w informatyce rośnie popularność technologii bazujących na wyważaniu otwartych drzwi w świetle reflektorów i przy dźwiękach fanfar. Co gorsza, często bywa tak, że w ten sposób łatane są usterki oprogramowania w wersji poprzedniej ewidentnie sknoconego. Przykłady pominę...

Jednym z najbardziej spektakularnych przykładów tego rodzaju „nowości" jest kariera języka HTML (*HyperText Markup Language*). Nie mam zamiaru deprecjonować tego, co uczynił dla świata Timothy Berners-Lee, twórca idei Pajęczyny. Nie on też odegrał rolę wyważającego otwarte drzwi, tylko jego zamożni a hałaśliwi następcy. Jedyne, z czym bym polemizował, to mniemanie, jakoby rewolucyjność idei tkwiła w zastosowanym języku.

Języki znacznikowe

Sama idea znaczników (*mark*) jest tak stara, jak przetwarzanie tekstów, a języki operujące znacznikami (*markup languages*) były znane, zanim powstał HTML. Zasługa Bernersa-Lee polega na tym, iż zauważył on, że system znaczników pasuje jak ulał do przesyłania informacji poprzez sieć, czyli że docenił potęgę tkwiącą w znacznikach. Ale bynajmniej ich nie wymyślił. Można wręcz zaryzykować stwierdzenie, że Pajęczyna by nie powstała, gdyby wcześniej nie istniały języki znacznikowe, w szczególności SGML – *Standard Generalized Markup Language*.

Języki znacznikowe są jakby językami programowania „na opak" – o ile w „zwykłych" językach programowania podstawę stanowią instrukcje dla komputera, a fragmenty tekstu, zwane łańcuchami (*string*) lub napisami, wyróżniane są za pomocą specjalnych separatorów, o tyle w językach znacznikowych podstawowym tworzywem jest tekst, mniej lub bardziej gęsto poprzetykany instrukcjami odróżniającymi się jednoznacznie od tekstu (np. `<head>`, `\begin{section}`, itp.), przeznaczonymi m.in. dla komputera.

Niewinne określenie „m.in." stanowi o sile metody znaczników.

Rzecz bowiem w tym, że chociaż podstawowym przeznaczeniem języków znacznikowych jest wizualizacja tekstu, to jednak same znaczniki w pierwszym rzędzie służą do wyróżniania

logicznego określonych fragmentów tekstu. Wbrew pozorom nie musi się to wiązać koniecznie z odróżnianiem wizualnym – możemy chcieć odróżnić imiona od nazwisk, chociaż w wynikowej postaci tekstu (wydruk, ekran) i imiona, i nazwiska wyglądać będą tak samo. Natomiast z tekstu źródłowego można w razie potrzeby bez trudu wyłuskać osobno spis użytych imion, a osobno spis użytych nazwisk.

Oczywiście nie ma nic za darmo – żeby z upstrzonego znacznikami tekstu w łatwy sposób wydobyć sensowną informację, trzeba się nieco napracować, zwłaszcza koncepcyjnie, na etapie wymyślania systemu znaczników. W tym kontekście – kto wie, czy nie częściej niż gdzie indziej – pojawić się może sposób przetwarzania danych określany jako *gigo* (*garbage in, garbage out* – śmieci na wejściu, śmieci na wyjściu). Zły system znaczników to po prostu zmarnowany czas, zaś stworzenie dobrego systemu wymaga fachowości. Czyżby tu się kryło źródło niezasłużenie małej popularności języków znacznikowych?

Języki znacznikowe nadają się nie tylko do przetwarzania sieciowego.

Przykładem innego rodzaju języka znacznikowego może być system składu tekstu TeX, stworzony przez Donalda E. Knutha. TeX został na tyle starannie zaprojektowany, że dane TeX-owe znacznie łatwiej przenieść pomiędzy platformami sprzętowo-systemowymi, niż programy napisane w takich, zdawałoby się reprezentujących ogólnoświatowy standard, językach programowania jak C czy PostScript. Decyzja wyboru języka znacznikowego odegrała przy tym istotną rolę. System znaczników TeX-a był tworzony głównie pod kątem definiowania formy graficznej tekstu. Jednakże doświadczeni użytkownicy TeX-a z reguły podporządkowują się zasadzie używania znaczników w taki sposób, by niosły one nie tylko informację formatującą, ale również – a może nawet przede wszystkim – logiczną.

Jako ciekawostkę można odnotować, że twórcy formatu RTF (*Rich Text Format*), czyli języka znacznikowego wykorzystywanego przez niektóre programy edytujące środowiska Windows, najwyraźniej czerpali natchnienie z systemu TeX, o czym świadczy notacja, bliźniaczo podobna do notacji wymyślonej przez Knutha.

SGML

Jak sama nazwa wskazuje, SGML jest uogólnionym językiem znacznikowym, metajęzykiem, definiuje bowiem całą klasę języków, a nie – jak na przykład TeX – jeden konkretny język (chociaż dzięki makrodefinicjom TeX może być traktowany jako język w pewnym sensie rozszerzalny). Tym samym zachęca do wprowadzania swoich własnych konwencji notacyjnych, adekwatnych do danego zadania. Jednocześnie zostaje zachowana daleko idąca zgodność pomiędzy różnymi instancjami języków definiowalnych za pomocą SGML-a, co było, rzecz jasna, intencją jego twórców.

Język HTML do pewnego etapu był w istocie instancją SGML, co oznaczało wysoki stopień standaryzacji, a w konsekwencji przenośność danych HTML-owych. Nie jest jasne, czy w przyszłości stan rzeczy nie ulegnie zmianie. Majsterkowicze spod znaku www wykazują bowiem niespożytą inwencję twórczą i – mimo energicznych wysiłków ze strony *World Wide Web Consortium*, organizacji, która dba o spójność światowej Pajęczyny i w której udziela się również Berners-Lee – każdy na własnej skórze zapewne doświadczył, jak się owa inwencja odbija na przenośności danych HTML-owych.

Ściśle rzecz biorąc, dane SGML-owe składają się z dwóch części: z tekstu z powstawianymi znacznikami oraz formalnej definicji użytego języka znacznikowego, czyli pliku DTD (*Document Type Definition*). O danych zawartych w plikach DTD można myśleć jako o produkcjach gramatycznych, zapisanych w specyficzny dla SGML-a sposób.

W przypadku HTML-a definicja DTD oczywiście została sformułowana, aczkolwiek większość użytkowników Pajęczyny zapewne nie zdaje sobie sprawy z jej istnienia, gdyż zwyczajnie jej nie potrzebuje. Pliki DTD są wykorzystywane głównie w celu sprawdzenia formalnej poprawności danych zapisanych w konkretnym języku znacznikowym. Robią z nich użytek programy przetwarzające dane SGML-owe.

Warto odnotować, że istnieje znakomity, uniwersalny parser plików SGML-owych, autorstwa Jamesa Clarka, dostępny bezpłatnie wraz ze źródłami na zasadach *public domain*. Istnieją też edytory plików SGML-owych, sprawdzające automatycznie poprawność wprowadzanego tekstu na podstawie pliku DTD, ale w większości przypadków są to bardzo drogie produkty

komercyjne i – jak na razie – nic nie słychać na temat wersji spolonizowanych.

Nie miejsce tu na drobiazgowe rozważania na temat SGML-a. Jako niezłe wprowadzenie zainteresowanym tę tematyką można polecić publikację internetową „Gentle introduction to SGML", dostępną na przykład pod adresem `http://etext.virginia.edu/tei-tocs1.html`

Znaczniki dla „ubogich"

W niektórych przypadkach okazuje się, że wymagany jest bardzo prosty system znaczników. Wówczas rygorystyczne przestrzeganie zasad tworzenia plików SGML-owych jest zbytecznym utrudnieniem, a zatem mija się z celem. W szczególności nie ma potrzeby tworzenia pliku DTD i posługiwania się zaawansowanymi narzędziami edycji plików.

Aby poradzić sobie z tworzeniem i analizą danych, wystarcza zwykły edytor tekstowy (im mocniejszy, tym lepiej – godny polecenia w tym kontekście jest edytor `emacs`) oraz użyte *ad hoc* standardowe narzędzia systemowe, typu wspomnianych na wstępie programów `sed` czy `awk`.

Przykładem może być prywatny notatnik adresowy: kilka znaczników, takich jak `<nazwisko>`, `<imię>`, `<adres>`, `<firma>`, `<telefony>`, powinno w zupełności wystarczyć, zwłaszcza jeśli dopuścić znaczniki z parametrami, np. `<firma nr=1>`, `<firma nr=2>`, itp.

Pielęgnacja „baz danych" tego rodzaju jest niemalże trywialna, a ich zalety – ogromne.

Podstawową zaletą jest właśnie łatwość pielęgnacji – najprostszy edytor tekstowy wystarcza, by zachować spójność i aktualność danych. Dzięki zastosowaniu postaci tekstowej danych unika się problemu, trapiącego niemal wszystkich, którym przychodzi korzystać z danych binarnych, generowanych przez tzw. profesjonalne oprogramowanie; należy dziękować losowi, jeśli wyższe wersje programów w sposób mniej więcej poprawny odczytują dane binarne zapisane przez wersje niższe. Do tego, że na odwrót rzekomo być nie może, a także do tego, że binarny format zapisu jest ściśle tajny, producentom udało się przyzwyczaić użytkowników.

Drugą istotną zaletą jest łatwość wydobywania żądanej informacji. Takim zabiegiem w wypadku bazy adresowej może być znalezienie wszystkich znajomych mieszkających w danym mieście.

Trzecią zaletą, być może najważniejszą, jest łatwość dokonywania automatycznych przekształceń danych za pomocą prostych narzędzi. Na przykład napisanie w języku `awk` programu po prostu usuwającego znaczniki, tzn. Pozostawiającego „goły" tekst, jest zadaniem banalnym nawet dla kogoś, kto zna ten język jedynie pobieżnie. Nieco więcej wprawy – ale naprawdę tylko *nieco* – wymaga zamiana takich danych na postać drukowaną. W tym celu można się posłużyć systemem TeX albo też dokonać bezpośredniej konwersji na język urządzenia drukującego (praktycznie jedynym wchodzącym w rachubę kandydatem jest w tym wypadku PostScript).

Na tym możliwości konwersji się nie kończą – jeśli mamy plik zapisany w języku PostScript, to zamiana na postać elektroniczną PDF (*Portable Data Format*), rozumianą przez dostępny bezpłatnie program Acrobat Reader, sprowadza się do nabycia za kilkaset złotych programu Acrobat Distiller. Formalnie bezpłatnie, ale za to dość boleśnie z powodu zwyczajowo kiepskiej dokumentacji (a więc niezupełnie za darmo), można dokonać konwersji danych do postaci akceptowanej przez program WinHelp. W tym celu należy przetłumaczyć dane na format RTF (szczególnej postaci), z którego korzysta Microsoftowy program o nazwie *Help Compiler*.

Znaczniki a zarządzanie dokumentacją

Ostatnie dwa przykłady konwersji nie są szczególnie przydatne w przypadku danych w rodzaju prywatnego spisu adresów.

Ale jeśli chodzi o dokumentację, a zwłaszcza dokumentację oprogramowania, to zarówno postać drukowana jak i elektroniczna są nader interesujące. Warunkiem niezbędnym jest wszakże przygotowanie dokumentacji w formie znacznikowej i posiadanie odpowiednich narzędzi do jej pielęgnowania, w szczególności do konwersji.

Okazuje się, że nawet obszerna dokumentacja nie musi wcale wymagać skomplikowanego systemu znaczników. Warunkiem jest narzucenie (na etapie projektu dokumentacji) zwartości i spójności logicznej, zarówno w stosunku do formy, jak i do treści. Wymaga to oczywiście starannego przestrzegania dyscypliny w trakcie

tworzenia i pielęgnowania dokumentacji. Jeśli sprostanie tym wymaganiom miałoby być kłopotliwe dla producenta software'u, to można by przypuszczać, że na analogiczne kłopoty natrafia on podczas tworzenia oprogramowania, a do takiego oprogramowania należy podchodzić z rezerwą.

Praktyka pokazuje, że analiza języka znaczników nadającego się do tworzenia dokumentacji nie jest już wprawdzie tak prosta, jak analiza prywatnej bazy adresowej, ale nie jest też nadmiernie skomplikowana. Stopień trudności jest mniej więcej taki, jak stopień trudności analizy wyrażeń arytmetycznych – prosty rekursywny parser sterowany składnią BNF w zupełności wystarcza. Zadanie to należy do kanonu elementarnych umiejętności programistycznych i można oczekiwać, że w profesjonalnym zespole programistów co najmniej jedna osoba powinna umieć taki parser w krótkim czasie (kilku dni) w swoim ulubionym języku napisać.

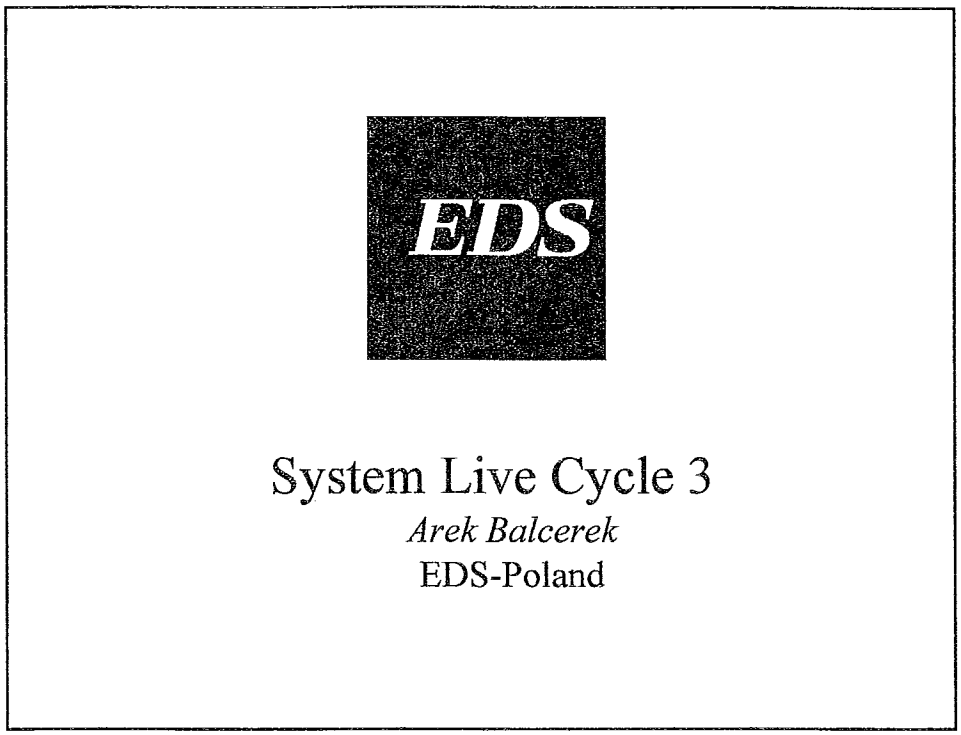
Niestety, praktyka pokazuje również, że producenci oprogramowania nie darzą dokumentacji szczególną troską. Powody są co najmniej dwa: po pierwsze, wymagania użytkowników w tym względzie nie są zbyt wielkie, co zresztą nie najlepiej świadczy o średnim poziomie kultury informatycznej; po drugie, sami producenci nie zdają sobie sprawy, że stworzenie lepszej dokumentacji nie oznacza wyłącznie dodatkowych kosztów. Starannie przygotowana dokumentacja nie jest jedynie elementem podnoszącym w sposób niemierzalny rynkową atrakcyjność produktu. Pozwala również na sprawdzenie wewnętrznej spójności samego oprogramowania, wyszukanie niejednorodności, a nawet błędów, dzięki możliwości łatwego analizowania danych źródłowych pod różnym kątem, tworzeniu różnego rodzaju roboczych skorowidzów (np. nazw zmiennych czy procedur) itp.

Wypada mieć nadzieję, że sytuacja w tej dziedzinie ulegnie jednak kiedyś zmianie na lepsze, z obopólną korzyścią – i dla użytkowników, i dla producentów oprogramowania.

Arek Balcerek

EDS-Poland

System Live Cycle 3



Korporacja EDS

- g Istnieje od 1962 roku, notowana na Giełdzie
- g 110,000 pracowników w 44 krajach
- g Ponad 9000 klientów
- g 15,2 mld $ przychodów w 1997 roku
- g 84 mld $ w zamówieniach

Copyright EDS-Poland, PTI, Szczyrk 1998 2 EDS

EDS w Polsce

- g EDS-Poland Sp. z o.o
- g Istnieje od 1993 roku
- g Centrala w Warszawie
- g Biuro w Gliwicach
- g Ponad 80 pracowników
  - Lokalni specjaliści
  - Zagraniczni konsultanci

Copyright EDS-Poland, PTI, Szczyrk 1998 3 EDS

System Live Cycle 3

1. Wprowadzenie
2. Fazy metodologii SLC 3
3. "Process Sourcerer"
4. Korzyści dla klienta
5. Korzyści dla EDS
6. Podsumowanie

Copyright EDS-Poland, PTI, Szczyrk 1998 4 EDS

Fazy metodologii SLC 3

- g Faza Definicji (*Define Phase*)
- g Faza Analizy (*Analyse Phase*)
- g Faza Projektu (*Design Phase*)
- g Faza Wykonania (*Produce Phase*)
- g Faza Optymalizacji (*Optimize Phase*)
- g Faza Wdrożenia (*Implement Phase*)

Copyright EDS-Poland, PTI, Szczyrk 1998 5 EDS

Faza Definicji

- g Identyfikacja potrzeb biznesowych
- g Określenie ogólnych wymagań klienta
- g Ustalenie zakresu projektu
- g Stworzenie ogólnego planu projektu
- g Określenie liczby iteracji oraz ich zakresu

Copyright EDS-Poland, PTI, Szczyrk 1998 6 EDS

Faza Analizy

- g Aktualne aspekty biznesowe i techniczne środowiska klienta
- g Uszczegółowienie wymagań klienta
- g Identyfikacja komponentów logicznych systemu

Copyright EDS-Poland, PTI, Szczyrk 1998 7 EDS

Faza Projektu

- g Tworzenie, ulepszanie lub eliminowanie procesów biznesowych
- g Ustalenie komponentów technicznych
  - sprzęt
  - oprogramowanie
  - architektura sieciowa
- g Stworzenie procedur konwersji i migracji

Copyright EDS-Poland, PTI, Szczyrk 1998 8

Faza Wykonania

- g Stworzenie lub zakup niezbędnego oprogramowania
- g Zaprogramowanie procesów biznesowych
- g Nabycie oraz instalacja komponentów sprzętowych i sieciowych
- g Integracja i testowanie komponentów
- g Przygotowanie materiałów szkoleniowych

Copyright EDS-Poland, PTI, Szczyrk 1998 9

Faza Optymalizacji

- g Testowanie w środowisku klienta oraz dostrajanie komponentów systemu
- g Udoskonalanie krytycznych elementów systemu pod kątem niezawodności i wydajności
- g Uzyskanie ostatecznej akceptacji systemu ze strony klienta

Copyright EDS-Poland, PTI, Szczyrk 1998 10

Faza Wdrożenia

- g Trening użytkowników
- g Instalacja systemu w środowisku klienta
- g Migracja aktualnych procesów i danych do nowego rozwiązania

Copyright EDS-Poland, PTI, Szczyrk 1998 11

"Process Sourcerer"

- g Narzędzie do wdrażania metodologii SLC 3
- g Stworzone przez *Electronic Data Systems*
- g Uwzględnia trzy główne komponenty procesów:
  - Element (*Work Elements*)
  - Produkt (*Work Products*)
  - Uczestnik (*Agents*)

Copyright EDS-Poland, PTI, Szczyrk 1998 12

EDS and the EDS logo are registered marks of Electronic Data Systems Corporation
EDS is an equal opportunity employer m/f/d
Copyright ©1997 Electronic Data Systems Corporation All rights reserved

EDS jest oficjalnym dostawcą usług informatycznych dla Mistrzostw Świata W Piłce Nożnej we Francji 1998

www.eds.com
www.eds.com.pl

Copyright EDS-Poland, PTI, Szczyrk 1998 13

Janusz Wielki

Akademia Ekonomiczna Wrocław
jwielki@polo.po.opole.pl

Podstawowe elementy krytyki reengineeringu i analiza ich zasadności

1. Wprowadzenie

Niemal od samego początku reengineering zbierał różne opinie krytyczne pod swoim adresem. Ukazywały się one najczęściej w specjalistycznych publikacjach i dotyczyły różnych aspektów wiążących się z tą koncepcją. W czerwcu 1997 roku bardzo spektakularny „atak" na reengineering przypuścił wpływowy i poczytny Business Week piórem swego analityka Johna A. Byrna. Opierając się mianowicie na wynikach badań Bain & Co. Consultants stwierdził on wszem i wobec, iż reengineering jest koncepcją odnoszącą zdecydowanie najmniejsze sukcesy spośród wszystkich najpopularniejszych koncepcji zarządzania związanych z wprowadzaniem zmian w funkcjonowaniu organizacji. Dodatkowym „smaczkiem" całej sprawy jest fakt, iż ten sam Business Week parę lat wcześniej nazwał współtwórcę reengineeringu, Michaela Hammera, *„jednym z czterech najwybitniejszych guru zarządzania lat dziewięćdziesiątych"* (patrz [CAHM95]). Ciekawe jest wobec tego co takiego zdarzyło się podczas tego bądź co bądź krótkiego okresu, iż ten wpływowy magazyn zmienił w tak radykalny sposób opinię o reengineeringu i jego współtwórcy. Oczywiste jest też, iż jeśli opinia wyrażona przez Business Week byłaby rzeczywiście prawdziwa to nie byłoby specjalnego sensu zajmować się tą koncepcją w przyszłości. Wydaje się też, iż ocena wartości reengineeringu byłaby bardzo istotna dla polskich menedżerów ze względu na fakt, iż wiele krajowych firm wymaga przeprowadzenia głębokich i radykalnych zmian, a reengineering wskazywany jest w niektórych publikacjach jako właśnie ta koncepcja która jest najbardziej użyteczną do tego celu. Przykładem może tu być artykuł S.Trautmana w którym twierdzi on, iż umiejętność wykorzystania reengineeringu w praktyce jest niezbędna dla krajowych organizacji w warunkach globalnej konkurencji i że stanowi on *„skok jakościowy i wyzwanie dla dyrekcji i zarządów większości polskich firm"*[1] [TRAUT96]. Dlatego też w dalszej części spróbujemy, opierając się na dostępnych źródłach, dokonać przeglądu podstawowych zarzutów pod adresem reengineeringu i ocenić ich zasadność. Na zakończenie zaś przedstawione zostaną nasuwające się wnioski i sugestie oraz dokonana zostanie próba oceny użyteczności tej koncepcji

2. Najczęstsze zarzuty pod adresem reengineeringu

Tak jak wspomniano wcześniej spora ilość krytycznych uwag w odniesieniu do reengineeringu pojawiła się w literaturze tematu. Poniżej przedstawionych zostało sześć najczęściej powtarzających się spośród nich i kolejno zostaną one omówione.

Bardzo duży procent projektów związanych z reengineeringiem kończy się niepowodzeniem

Jest to zdecydowanie najpowszechniejszy i chyba najbardziej znaczący zarzut w stosunku do BPR. Krytycy reengineeringu wskazują na fakt, iż około 70 % prób implementacji reengineeringu kończy się niepowodzeniem (patrz np. [COV96], [MALH96]). Tak naprawdę to zarzut ten nie jest specjalnie zaskakujący, gdyż już Hammer i Champy pisali w swej książce: *„nasze amatorskie szacunki wskazują, iż około 50% do 70%, organizacji które podjęły wysiłek związany z reengineeringiem nie osiągnęły takich rezultatów jak to zakładały"* [CAHM95]. Wydaje się zresztą, iż wyjaśnienie tego faktu jest stosunkowo proste i oczywiste. Ryzyko wiążące się z implementacją

[1] Pierwsze kroki zostały poczynione w Polsce dość wcześnie bo w roku 1994. Wtedy to zakończony został w Stomilu Sanok S.A. pionierski jeżeli chodzi o kraje byłego bloku wschodniego projekt reengineeringu [WIZ97].

BPR jest znaczne i na fakt ten wskazywano wiele razy (patrz [DAV93], str. 189).

Ryzykowność reengineeringu wynika głównie z zakresu działań jakie muszą być podjęte. Obejmują one zmiany w bardzo wielu obszarach [GUKETE97]:

- podsystemy zarządzania (wartości, wskaźniki, styl pracy),
- czynniki ludzkie (zawody, aspekty kulturowe, umiejętności),
- technologia informatyczna,
- struktura organizacyjna (kompresja pionowa).

Ten szeroki zakres koniecznych zmian jak również ich różnorodność powodują, iż zwiększa się prawdopodobieństwo popełnienia błędu czy też niedoceniania wpływu pewnych istotnych elementów, czego rezultatem może być niepowodzenie całego przedsięwzięcia. Oczywiste jest, iż tak wysoki poziom ryzyka kompensowany jest korzyściami, które można osiągnąć w przypadku powodzenia projektu. Potencjalne korzyści są znacznie większe niż te będące rezultatem zastosowania koncepcji takich jak *business process improvement* czy TQM. W obydwu z nich ryzyko niepowodzenia jest znacznie niższe z uwagi na fakt, iż ich celem jest jedynie doskonalenie procesów w istniejącej postaci, a nie ich dogłębne przeprojektowanie. Spróbujmy się przyjrzeć jakiego rodzaju elementy decydują najczęściej o niepowodzeniu reengineeringu. Wyniki badań przeprowadzonych w roku 1995 przez *CFO Magazine* wspólnie z *Ernst & Young* pośród menedżerów amerykańskich wskazują, iż za elementy w największym stopniu decydujące o niepowodzeniu całego projektu uważają oni [CFO95]:

- niewłaściwe kierowanie przebiegiem projektu - 17,4%,
- czynniki natury kulturowej - 15,5%,
- brak wsparcia dla projektu ze strony kierownictwa - 14,7%.

Z kolei według badań *„1996 CIO Survey"* przeprowadzonych rok później przez Deloitte & Touche Consulting Group najważniejszymi przyczynami porażek są [DETO96]:

- opór wewnątrz organizacji - 82%,
- nieodpowiednie wspomaganie ze strony kierownictwa - 72%,
- nierealistyczne oczekiwania - 66%,
- nieodpowiednie kierowanie projektem - 60%.

Warto do tego dodać, iż szczególnie we wczesnym okresie BPR był po prostu modny i wielu szefów firm decydowało się na przeprowadzenie reengineeringu bez dokonania oceny rzeczywistej jego potrzeby, podążając za ogólnym trendem [BPR-L97]. Natknąć się również można na stwierdzenie, iż tak wysoki poziom niepowodzeń jest rezultatem włączania do szacunków projektów, które określane co prawda były mianem reengineeringu, ale tak naprawdę nic wspólnego z nim nie miały [2].

Obok powszechnie cytowanych danych o niepowodzeniach dotykających 70% projektów reengineeringu, znaleźć można również dane wskazujące na całkiem znaczny poziom ogólnego zadowolenia menedżerów z rezultatów osiąganych przy wykorzystaniu BPR. Przykładem mogą być wyniki cytowanych wcześniej badań *„1996 CIO Survey"* (patrz tabela 1).

Również wyniki badań przeprowadzonych pośród 30 firm meksykańskich (należących do 500 największych w tym kraju), które wprowadzały w życie program reengineeringu, wskazują na znaczny poziom zadowolenia z osiągniętych rezultatów (patrz tabela 2).

Tabela 1. *Ocena rezultatów osiągniętych dzięki wykorzystaniu reengineeringu*

Źródło: [DETO96]

Satysfakcjonujące	76 %
Trudno powiedzieć	13 %
Niezadowalające	11 %

[2] Bliżej odniesiemy się do tego zagadnienia w podsumowaniu.

Tabela 2. *Ocena rezultatów reengineeringu w 30 firmach meksykańskich*

Źródło: [BPR-L96]

Wartościowe rezultaty	69 %
Przeciętne rezultaty	21 %
Nie czas jeszcze na ocenę rezultatów	10 %

Niedocenianie roli pracowników w transformacji procesów

Jest to poważny i generalnie rzecz biorąc słuszny zarzut. Reengineering dość powszechnie zresztą zyskał sobie opinię koncepcji, która koncentruje się głównie na technologii informatycznej, traktujące pracowników jako drugorzędny element, którym można dowolnie manipulować. Bardzo dosadnie ujął tą sytuację T. Davenport pisząc, że *„BPR traktował ludzi pracujących wewnątrz firmy tak, jakby byli oni po prostu kolejnymi bitami i bajtami, wymiennymi częściami, które muszą zostać poddane reengineeringowi"* [DAV96]. Prawda jest natomiast zupełnie inna. Ludzie stanowią kluczowy element każdego programu reengineeringu, ponieważ to właśnie oni wprowadzać będą zmiany, a następnie będą musieli funkcjonować w nowych dla siebie rolach jak również warunkach. Dlatego też docenienie znaczenia powyższego faktu przez menedżerów oraz ich pomoc wydają się być niezwykle istotnymi czynnikami.

Konsekwencją niedoceniania roli pracowników w reengineeringu stało się również niedoszacowanie poziomu ich obaw związanych z nową dla nich sytuacją. Obawy te spowodowane są wieloma problemami. Jako, że projekty BPR są zwykle dosyć złożonej natury, pracownicy muszą w związku z tym stawić czoła wielu nowym problemom. Pierwszym z nich jest odmienny styl pracy. W organizacjach, w których przeprowadzono reengineering mają oni zdecydowanie większy zakres odpowiedzialności, muszą podejmować decyzje jak również oczekuje się od nich elastyczności w działaniu. Wydaje się, iż w początkowym okresie może to być dla niektórych z nich dosyć trudne.

Kolejnym trudnym do przezwyciężenia elementem są zmiany w odniesieniu do preferowanych w firmie wartości i poglądów. Ich tworzenie może być tak naprawdę najtrudniejszym aspektem do przezwyciężenia, ponieważ powstają one przez lata i w związku z tym trudno oczekiwać, iż można je będzie zmienić jednym posunięciem. Również zmiana nastawienia do technologii informatycznych, szczególnie w przypadku starszych pracowników, jest ważnym i trudnym elementem wprowadzanych zmian.

Kolejnym elementem, który może być poważnym źródłem obaw pracowników jest konieczność opanowania przez nich nowych umiejętności i kompetencji. Będą one oczywiście różne w zależności od pozycji zajmowanej w hierarchii organizacyjnej.

Zaliczyć do nich można między innymi [CORR96]:

- umiejętność zarządzania ryzykiem,
- umiejętność rozwiązywania problemów,
- umiejętność pracy grupowej,
- ukierunkowanie na potrzeby klientów,
- umiejętność myślenia strategicznego,
- umiejętność planowania.

I na koniec jeszcze jeden element powiększający ich obawy względem reengineeringu. Jest nim obawa przed zwolnieniami. Jej źródłem są głównie przekazywane z ust do ust opinie, z których wynika, iż typową konsekwencją projektów BPR są masowe zwolnienia. W ten sposób możemy przejść do kolejnego zarzutu.

Reengineering jest tym samym co downsizing

Kolejna grupa krytyków reengineeringu twierdzi, iż prawie każdy projekt BPR prowadzi do zwolnień na dużą skalę. Niektórzy posuwają się o krok dalej twierdząc, że reengineering równoważny jest *downsizingowi*, czyli „odchudzaniu" organizacji. Jest to skojarzenie zdecydowanie mylne [BUSS98]. Bywa jednak tak głęboko zakorzenione w świadomości pracowników, iż np. w firmie Corning świadomie unikano używania słowa reengineering w stosunku do przeprowadzanych zmian [ACNE96].

Dostępne dane wskazują, iż typowym rezultatem implementacji reengineeringu jest utrata pracy przez około 21% pracowników (patrz [DAV96], [FRMAC95]). Bywa jednak, iż procent ten jest znacznie większy. Np. w wyniku wspomnianego wcześniej reengineeringu w Stomilu Sanok S.A. pracę utraciło 2500 ludzi z 6000 załogi, czyli 41% [WIZ97].

Konieczne jest jednak podkreślenie, iż zwolnienia same w sobie nie są absolutnie celem reengineeringu, bywają one raczej jego rezultatem [LEE96]. Kwestię tę zresztą wyjaśnił parę lat temu sam M. Hammer w artykule opublikowanym na łamach The Economist (patrz [HAMM94]).

Wydaje się jednak sprawą dosyć oczywistą, że wprowadzanie w życie głębokich i radykalnych zmian przy szerokim wsparciu technologii informatycznej może prowadzić, i często prowadzi do eliminacji części stanowisk. Również kompresja pionowa dokonująca się w organizacjach oraz rozszerzenie zakresu uprawnień i odpowiedzialności pracowników prowadzi do eliminacji głównie menedżerów średniego szczebla. Warto jednak w tym miejscu przypomnieć, iż zmniejszanie się ilości menedżerów średniego szczebla nie jest zjawiskiem wcale nowym. Trzeba bowiem przypomnieć, iż w rezultacie wprowadzanie w życie programów TQM jedynie w USA w okresie dziesięciu lat ubyło około miliona tego typu stanowisk [LUCE95].

Również pewien wpływ na opinię o częstych zwolnieniach związanych z implementacją reengineeringu miało ścisłe stosowanie w praktyce koncepcji „czystej kartki" (*clean slate*), szczególnie w we wczesnym stadium rozwoju BPR, chociaż powszechność tego zjawiska nie była zbyt znaczna [GUKETE97].

Reengineering nie jest niczym istotnym, a jedynie krótkotrwałą modą

Następna liczna grupa krytyków twierdzi, iż trudno jest traktować reengineering jako koncepcję mającą przed sobą przyszłość, gdyż jest to tylko przejściowa moda. Tego typu opinie pojawiały się w licznych publikacjach prawie od zarania reengineeringu i wciąż można się natknąć na nowe (np. [BYRN97]). Przyjrzymy się czy taką samą opinię wyrażają na temat reengineeringu menedżerowie czyli ludzie, którzy nie piszą na jego temat, ale mają z nim do czynienia w praktyce. Przywołajmy po raz kolejny "*1996 CIO Survey*". Jego wyniki wskazują jasno, iż opinie menedżerów różnią się zasadniczo od tych jakie pojawią się w publikacjach prasowych. Około 84% ankietowanych menedżerów stwierdziło bowiem, iż reengineering jest „ *ważnym narzędziem zarządzania* " (patrz tabela 13).

Tabela 13. *Co menedżerowie sądzą na temat reengineeringu ?*

Źródło: [DATO96]

Procent respondentów wyrażających określoną opinię	
Ważne narzędzie zarządzania jeżeli jest właściwie użyte	84 %
Jeszcze jedna przejściowa moda która prawdopodobnie zniknie za kilka lat	13 %
Głównie powód dla ludzi ze środowisk akademickich oraz konsultantów dla powiększania swych dochodów	3 %

Trzeba wyraźnie podkreślić, iż żaden z podstawowych komponentów reengineeringu tj.:

- technologia informatyczna,
- orientacja procesowa,
- koncepcja „czystej kartki "

nie ma cech krótkotrwałej mody, pozostają wciąż aktualne i ważne.

Znaczenie technologii informatycznych we wprowadzaniu zmian zwiększa się wciąż w kontekście gwałtownego rozwoju Internetu.

Jeszcze jedna uwaga na koniec. Do grona ludzi twierdzących, iż reengineering nabrał cech przejściowej mody należy obecnie również jeden ze współtwórców tej koncepcji - Thomas Davenport. Chociaż docenia on znaczenie wymienionych wcześniej składników reengineringu, to jednak twierdzi, iż szczególnie w USA koncepcja ta straciła na znaczeniu. Dodaje on tego, iż BPR w zasadzie od swego początku posiadało atrybuty teorii będącej tylko przejściową modą. Według niego cechy te obejmują [DAV97]:

- przesadne oczekiwania co do rezultatów,
- powierzchowne zrozumienie przez rzesze entuzjastów,
- posiadanie i czczenie guru,
- implementacja w firmach w uproszczony sposób.

Reengineering nie jest niczym nowym

Zarzut ten jest również dosyć często powtarzany. P. Strassmann twierdzi nawet, iż reengineering jest niczym więcej niż po prostu chwytliwą nazwą (*clever buzz word*) [STRASS94]. Tak naprawdę to rzeczywiście składniki reengineeringu, jak je uprzednio wymieniliśmy, były wcześniej wykorzystywane jednak zawsze osobno. Pomysł w koncepcji BPR jest odmienny. Łączy ona wszystkie te trzy komponenty razem po to, aby w ten sposób osiągnąć znaczące rezultaty [DAV96]. Prawdą jest również, iż reengineering posługuje się głównie technikami oraz narzędziami wykorzystywanymi już wcześniej (np. w TQM), jednak z drugiej strony coraz więcej pojawia się aplikacji przeznaczonych specjalnie dla BPR.

Jeżeli chodzi o techniki wykorzystywane w reengineeringu to zaliczyć do nich można np.: benchmarking, analizę „ości ryby", diagram Pareto, budżetowanie, diagramy przepływów, analizę łańcucha wartości czy burzę mózgów. Z kolei przykładami wykorzystywanych narzędzi mogą być następujące aplikacje: ABC Flow Charter, Microsoft Project, PACE, Oracle Process Manager. W [GUKETE97] zidentyfikowane zostały 72 techniki i 68 narzędzi wykorzystywanych podczas reengineeringu.

Teoria i praktyka reengineeringu pełna jest rewolucji i przemocy

Zarzut ten został sformułowany przez wspomnianego wcześniej P. Strassmana (patrz [STRASS95]) i jest on w pewien sposób zasadny, jednak tylko w odniesieniu do reengineeringu w ujęciu Michaela Hammera przedstawionym w artykule *„Don't automate, obliterate"* i biblii reengineeringu, napisanej wspólnie z Champym książce *„Reengineering the Corporation"*. Akcentował on w nich „radykalność" reengineeringu jako jego najważniejszą cechę. Jednak dosyć szybko i Hammer zaczął przesuwać akcenty w swym spojrzeniu na BPR. Już w drugim wydaniu *„Reengineering the Corporation"* z 1995 oraz w wydanej rok później *„Beyond Reengineering"* Hammer wskazuje, iż orientacja procesowa a nie radykalność, powinna być widziana jako najważniejsza cecha reengineeringu [HAMM97].

Zupełnie odmienną wizję BPR prezentował w swych artykułach, a przede wszystkim w książce *„Process Innovation"* Thomas Davenport. Sposób przedstawienia koncepcji reengineeringu jest w jego wydaniu pozbawiony jakichkolwiek elementów agresji i jest bardzo rzeczowy. Kładzie on nacisk również na inne elementy stymulujące reengineering poza technologią informatyczną, takie jak czynnik ludzki. Niestety, spojrzenie Hammera stało się dominującym tak, że większość menedżerów buduje w oparciu o nie swe wyobrażenie o reengieeringu.

3. Wnioski

Wydaje się, iż powyższy przegląd najważniejszych obszarów krytyki reengineeringu prowadzi do wielu istotnych wniosków płynących na przyszłość, jednak tak naprawdę żaden z nich nie podważa znaczenia koncepcji BPR. Reengineering jest ciągle młodą, rozwijającą się i ewoluującą metodą wprowadzania radykalnych zmian w funkcjonowaniu organizacji. Pomimo, iż niektórzy z jego współtwórców odsunęli się od niego (Davenport), z pewnością nie jest on koncepcją martwą jak chcieliby niektórzy z krytyków, a raczej jak pisze D.Coleman *„jesteśmy w środku rewolucji związanej z reengineeringiem (...)"* [COL97]. Najlepszym dowodem jest wzrastająca ilość nowych metodologii jak również aplikacji opracowywanych specjalnie na potrzeby reengineeringu [GUKETE97]. Również ilość państw w których BPR jest wykorzystywany dowodzi rosnącej jego popularności na świecie. W przeprowadzonych na początku 1997 przez ProSci badaniach dotyczących BPR uczestniczyły organizacje z 26 krajów znajdujących się na sześciu kontynentach [PROSCI97]. Z pewnością mniej jest obecnie wokół reengineeringu wrzawy i emocji, których było sporo w początkowym okresie, za to więcej rzeczowego czy wręcz naukowego podejścia. Mówi się też o wejściu BPR w drugą fazę rozwoju, oczekując, iż przyniesie ona wiele zmian na lepsze wynikających z analizy błędów pierwszego etapu, w tym również dużej ilości zaistniałych niepowodzeń [NISS96].

Niewątpliwie nasuwa się wiele istotnych wniosków. Wydaje się, iż dla zmniejszenia prawdopodobieństwa niepowodzenia projektu należy zwrócić uwagę na następujące elementy:

- właściwy wybór procesu (procesów) w których wprowadzane mają być zmiany,
- szczegółowe przygotowanie całego projektu,
- wybór sprawdzonej metodologii.

Również szczególnie istotnym elementem każdego projektu reengineeringu powinno być właściwe docenienie roli uczestniczących w nim ludzi i przygotowanie całego systemu motywacyjnego zachęcającego do aktywnego uczestnictwa we wprowadzanych zmianach oraz minimalizującego obawy związane z możliwą utratą pracy.

Następnym istotnym nasuwającym się wnioskiem jest konieczność traktowania reengineeringu jako formy głębokich i raczej długotrwałych zmian organizacyjnych, a nie jako koncepcji mającej służyć uzyskiwaniu znaczących rezultatów w krótkim czasie (*quick fix*) [GUKETE97]. Potwierdzają to wspomniane wcześniej badania przeprowadzone przez ProSci wskazujące, iż w chwili obecnej średni czas projektu reengineeringu wynosi 19,7 miesięcy, a może on być znacznie dłuższy [PROSCI97]. Np. w GTE Telephone Operations implementacja zmian trwała 5 lat, a ocenia się, iż w niektórych przypadkach może to być nawet 8 do 10 lat [LEE96], [BERZ96].

Jako, że reengineering bywa mylony z innymi koncepcjami spróbujmy na koniec sprecyzować kiedy właściwie mamy z nim do czynienia. Tak jak wspomniano wcześniej bardzo często jest on błędnie utożsamiany z *downsiziningiem* z którym tak naprawdę BPR nie ma nic wspólnego. Generalnie rzecz biorąc działania podejmowane w ramach *downsiziningu* sprowadzają się jedynie do zmniejszenia kosztów funkcjonowania organizacji poprzez zwolnienie części pracowników, natomiast w reengineeringu chodzi jednak o coś więcej.

Jest on też często mylony jest z *business process improvement* (patrz [BERTZ96]) pomimo, iż różnica jest między nimi wyraźna. Obie koncepcje oparte są co prawda o orientację procesową jednak reprezentują one zupełnie inną filozofię. Podczas gdy BPI koncentruje się na doskonaleniu istniejących procesów to reengineering stara się dokonać głębokiej ich transformacji szeroko wykorzystując do tego celu technologię informatyczną. Jednym z istotnych elementów rozróżniającym dotychczas obie koncepcje był osiągany stopień poprawy podstawowych wskaźników funkcjonowania organizacji. W przypadku *business process improvement* oczekiwano ich polepszenia o około 10% podczas gdy rezultatem reengineeringu miał być ich wzrost o *„50%, 100% lub nawet więcej"* [DAV93]. Natknąć się można nawet na stwierdzenie, iż z prawdziwym BPR mamy do czynienie wtedy gdy w wyniku przeprowadzonych zmian efektywność działania organizacji zostaje podwojona [BERZ96]. Jednak przeprowadzone w 1997 roku badania wskazują, iż obecne oczekiwania menedżerów w stosunku do BPR odnośnie poprawy podstawowych wskaźników (koszty, czas trwania procesu, wydajność) znacznie się obniżyły i kształtują się na poziomie 10% do 30% [PROSCI97]. Dlatego też identyfikacja reengineeringu jedynie na podstawie osiąganych wyników może być myląca. Podstawowe znaczenie ma stosowana filozofia działania.

Na koniec dwie jeszcze uwagi. Po pierwsze wydaje się, iż jeszcze dużo pracy zostało przed reengineeringiem. Biorąc pod uwagę jedynie proces marketingu, szacuje się, iż wkrótce około 70% organizacji będzie musiało poddać go reengineeringowi, przede wszystkim ze względu na rozwój Internetu i wynikające z tego konsekwencje [KOM97]. A co do cytowanego na początku artykułu z Business Week kwestionującego użyteczność tej koncepcji, warto dodać, iż wyrażona tam opinia oparta została na wypowiedzi 17,3% ankietowanych twierdzących, iż są niezadowolenia z rezultatów osiągniętych przy zastosowaniu BPR.

Literatura

[ACNE96] Ackerman R., Neilson G.; Partnering for Results: A Case Study of Re-engineering, the Corning Way; [http://www.strategy-business.com/casestudy/96205/page1.html]; 1996

[BERT96] Berztiss A.; Software Methods for Business Reengineering; New York: Springer-Verlag; 1996.

[BPR-L96] Business Process Re-engineering Mailing List BPR-L; BPR-L Digest # 883; [BPR-L@TWI.TUDELFT.NL];1996

[BPR-L97] Business Process Re-engineering Mailing List BPR-L; BPR-L Digest # 1162; [BPR-L@TWI.TUDELFT.NL];1997

[BUSS98] Business Architects; What Is Reengineering ?; [http://busarch.com/papers/]; 1998

[BYRN97] Byrne J.; Management theory - or fad of the month?; Business Week; June 23 1997

[CAHM95] Champy J., Hammer M.; Reengineering the corporation; Nicholas Brealey Publishing; 1995

[CFO95] CFO Magazine and The Ernst & Young; What you really think about Reengineering; [http://www.mediapool.com/offtherecord/cfo-bpr.html]1995.

[COL97] Coleman D.; Is re-engineering still relevant ?; Computerworld 21 kwiecień; [http://cwlive.cw.com:8080/home/print9497.nsf/All/SL0421dc] 1997

[CORR96] Corrigan S.; Human and organisational aspects of business process reengineering; [http://bprc.warwick.ac.uk./shef-summ.html]; 1996

[COV96] Covert M.; Successfully performing business process reengineering; [http://www.ozemail.com.au/~ieinfo/BPR.htm]; 1996

[DAV93] Davenport T.H.; Process innovation - reengineering work through information technology; Harvard Business School Press; 1993

[DAV96] Davenport T.H.; The fad that forgot people; [http://www.fastcompany.com/01/reengin.html]; 1996

[DAV97] Davenport T.H.; Prywatna korespondencja z prof. Davenportem pocztą elektroniczną; 29 wrzesień 1997

[DETO96] Deloitte & Touche Consulting Group; 1996 CIO Survey - Business Process Reengineering; [http://www.dttus.com/dttus/publish/cio/bpr.htm]; 1996

[FRMAC96] Francis A., MacIntosh R.; The market, technological and industry contexts of business process re-engineering in UK business; [http://bpcr.warwick.ac.uk/glasgow1.html]; 1996

[GUKETE97] Guha S., Kettinger W., Teng J.; Business Process Change - A study of methodologies, techniques and tools; [http://theweb.badm.sc.edu/bpr/]; 1997

[HAMM94] Hammer M.; Hammer defends re-engineering; The Economist; November 5th 1994

[HAM97] Hammer M.; Beyond Reengineering: How the Process-Centered Organization is Changing Our Work and Our Lives; HarperCollins; 1997

[KOM97] Komenar M.; Electronic Marketing; John Wiley & Sons, Inc.; 1997

[LEE96] Lee C.; Process Re-engineering at GTE: Milestones on a Journey Not Yet Completed; [http://www.strategy-business.com/casestudy/96406/page1.html]; 1996

[LUCE95] Lucey T.; Management Information Systems; DP Publications; 1995

[MALH96] Malhorta Y.; Business Process Redesign: An Overview; [http://www.brint.com/papers/bpr.htm]; 1996

[NISS96] Nissen M.; A Focused Review of the Reengineering Literature: Expert Frequently Asked Questions (FAQs); „Quality Management Journal" 3:3 1996

[PROSCI97] ProSci; Best Practices Report for Reengineering and Business Process Design Teams - 1997 Benchmarking Study; [http://www.prosci.com]; 1997

[STRASS94] Strassmann P.; The Hocus-Pocus of Reengineering; [http://www.strassmann.com/pubs/hocus-pocus.html]; 1994

[STRASS95] Strassmann P.: Reengineering; [http://www.strassmann.com/pubs/reengineering.html]; 1995

[TRAUT96] Trautman S.; Reengineering wyzwanie dla menedżerów ; Computerworld; 7 Październik 1996

[WIZ97] Wizdom Systems, Inc; BPR in Poland: How Wizdom Lead a Polish Company on its First BPR Project; [http://wizdomsystems.com/successpoland.htm] 1997

Jerzy R. Nawrocki

Instytut Informatyki, Politechnika Poznańska,
ul. Piotrowo 3a, 60-965 Poznań
fax: 0-61 8782375; e-mail: nawrocki@put.poznan.pl;
http://www.cs.put.poznan.pl/~nawrocki

PSP - osobisty proces tworzenia oprogramowania[1]

1. Wprowadzenie

PSP (ang. *Personal Software Process*) jest to zestaw metod i narzędzi ukierunkowanych na doskonalenie osobistego procesu tworzenia oprogramowania. PSP koncentruje się na podstawowych problemach inżynierii oprogramowania dotyczących zarządzania czasem projektu programistycznego oraz jakością oprogramowania. Strategia PSP została opracowana na początku lat 90-tych przez Wattsa Humphrey'a, znanego specjalistę z zakresu doskonalenia procesów tworzenia oprogramowania i byłego dyrektora programu badawczego "*Software Process Program*" w *Software Engineering Institute* (*Carnegie Mellon University*, USA). Najlepszym źródłem informacji na temat PSP jest monografia [1].

Problem zarządzania czasem jest rozwiązywany w PSP poprzez:

- planowanie terminu zakończenia projektu na podstawie danych dotyczących wcześniejszych projektów i wnioskowania statystycznego,
- kontrolę postępów w projekcie na podstawie metody Earned Value (metoda ta jest przedmiotem osobnego wystąpienia w ramach tej konferencji [2]).

Aby programista mógł uporać się z problemem jakości, PSP proponuje analizę błędów, analizę artefaktów (tj. specyfikacji wymagań, specyfikacji modułów, kodu itp.) oraz odpowiednie wzorce specyfikacji (ang. *specification templates*).

Często PSP jest widziana poprzez pryzmat formularzy, które Humphrey proponuje programiście (niektóre osoby, które uczęszczały na kursy PSP wręcz utożsamiają PSP z tymi formularzami). W moim odczuciu najbardziej oryginalnym i najciekawszym elementem PSP są nie formularze, lecz wnioskowanie statystyczne, które pozwala określić wartość oczekiwaną czasu zakończenia projektu i, co więcej, również przedział tej wartości dla założonego prawdopodobieństwa. Technika formularzy jest dość powszechnie stosowana i proponowana przez różnych autorów zajmujących się problemem zarządzania czasem (patrz np. [3,4]), jednakże żadna ze znanych mi prac z tego zakresu nie proponuje wykorzystania tak spójnej i silnej „klamry", jaką jest wnioskowanie statystyczne.

PSP opiera się na filozofii samodoskonalenia się i samozarządzania się programisty. PSP zakłada, że przedmiotem

[1] Praca wykonana w ramach grantu BW 43-290

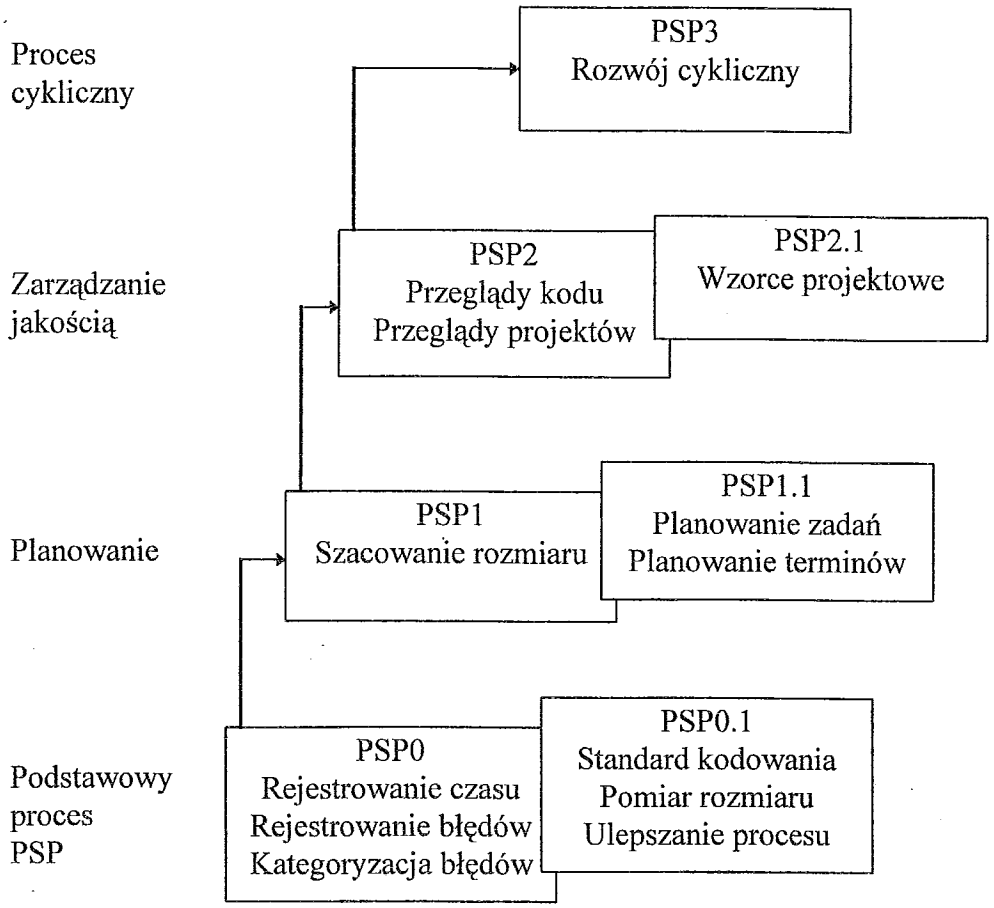
zarządzania są nie tyle zespoły ludzkie co procesy tworzenia oprogramowania. Tymi procesami trzeba zarządzać na wszystkich szczeblach organizacyjnych firmy, także (a raczej „przede wszystkim") na szczeblu pojedynczego programisty. W konsekwencji PSP zajmuje się procesami realizowanymi przez pojedynczego programistę. Z tego też względu z zestawu funkcji zarządzania wymienionych przez H. Fayola i rozwijanych później przez innych autorów (patrz np. [5,6]), PSP odnosi się do dwóch: planowania (zasobów, wyników i działań) oraz kontroli (ustalanie stanu, wyciąganie wniosków i podejmowanie działań korygujących).

Referat stanowi wprowadzenie do PSP. Krótko omówiono procesy zbierania danych, planowania zadań i terminów oraz kontrolę postępu prac w projekcie.

2. Stopnie wtajemniczenia w PSP

W ramach PSP można wyróżnić cztery stopnie wtajemniczenia: do poziomu 0 do 3 (patrz rys. 1). **Podstawowy proces PSP** (poziom 0) dotyczy rejestracji czasu pracy i popełnianych błędów. Rejestracja błędów, aby mogła być sprawnie prowadzona i by zgromadzone w ten sposób dane mogły podlegać łatwej analizie, wymaga opracowania kategoryzacji błędów. Nieco wyższy poziom wtajemniczenia (0.1) obejmuje przyjęcie standardu kodowania, metody pomiaru rozmiaru oprogramowania i przygotowanie się na potrzebę ciągłego ulepszania procesu tworzenia oprogramowania. **Planowanie** wymaga umiejętności szacowania rozmiaru oprogramowania, szacowania czasochłonności poszczególnych zadań wyodrębnionych w projekcie oraz zbudowania wiarygodnego harmonogramu. Uważam, że w zakresie tych dwóch poziomów (zbierania danych i planowania) PSP oferuje najwięcej. **Zarządzanie jakością** jest bardzo ważnym ale i bardzo trudnym problemem. PSP stawia tutaj na przeglądy (przegląd, w odróżnieniu od inspekcji, jest wykonywany przez autora modułu oprogramowania) i listy rzeczy do sprawdzenia (ang. *checklists*). Humphrey proponuje także korzystanie z wzorców projektowych (ang. *templates*) obejmujących specyfikację funkcjonalną, specyfikację stanów, specyfikację logiki oraz scenariusze operacyjne. **Proces cykliczny** dotyczy dużego oprogramowania i polega na podzieleniu go na mniejsze fragmenty, do których można stosować techniki rozwinięte w ramach PSP0, PSP1 i PSP2.

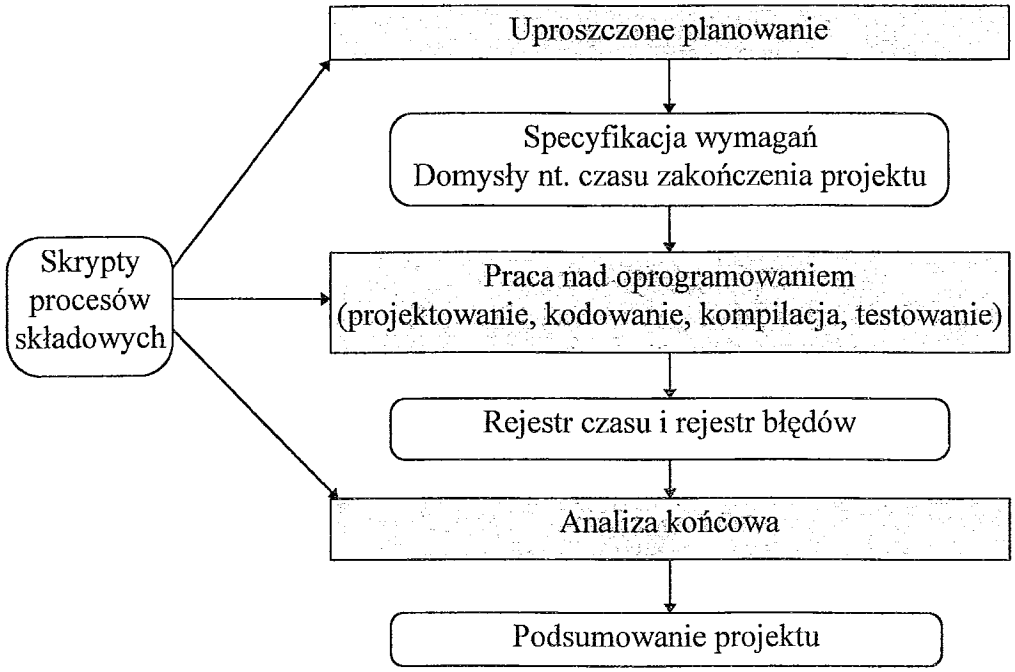
PSP odnosi się do prawie wszystkich elementów występujących w ramach CMM [7] oprócz zarządzania wymaganiami, zarządzania zleceniami, zapewnienia jakości, zarządzania konfiguracjami (poziom 2), programu szkoleń i koordynacji międzygrupowej (poziom 3).



Rys. 1. Ewolucja umiejętności w zakresie PSP.

3. Podstawowy proces PSP

Z punktu widzenia doskonalenia procesu tworzenia oprogramowania, głównym składnikiem podstawowego procesu PSP jest rejestracja czasu i błędów. Proces ten został zilustrowany na rys. 2. Na tym poziomie wtajemniczenia planowanie jest bardzo uproszczone. Na podstawie analizy wymagań programista bardziej zgaduje niż szacuje czas zakończenia projektu. W czasie pracy nad oprogramowaniem programista zbiera dane dotyczące swojego czasu pracy i popełnionych przez siebie błędach, które okażą się bardzo pomocne na wyższych stopniach wtajemniczenia. Każdy projekt kończy się analizą końcową (ang. *postmortem*), w trakcie której zebrane dane zostają wstępnie przetworzone i powstaje krótkie podsumowanie projektu. Każdy z procesów składowych jest opisany w odpowiednim skrypcie.



Rys. 2. Podstawowy proces PSP.

Na poziomie podstawowego procesu PSP najważniejszymi „narzędziami" są rejestry czasu i błędów. Na rys. 3 przedstawiono przykładowy **rejestr czasu pracy**. Każde „zabranie się do pracy" nad projektem programistycznym jest odnotowane w postaci osobnego wiersza. Jeśli w nieprzerwany sposób przechodzimy od jednej fazy projektu do następnej, to wówczas będzie tyle wierszy, ile faz przeszliśmy „za jednym zamachem". **Start** określa godzinę, gdy po raz któryś zabraliśmy się za nasz projekt a **Stop** podaje godzinę zakończenia pracy nad projektem (godzinę zakończenia pracy a nie ukończenia projektu - projekt może być kontynuowany np. następnego dnia). Jeśli w czasie pracy mieliśmy jakieś przerwania (np. telefon od szefa), to na bieżąco je odnotowujemy w rejestrze (ich „czas obsługi" liczony w minutach zapisujemy w kolumnie **Przerwania** a przyczynę w kolumnie **Komentarz**). **Delta** określa efektywny czas spędzony nad projektem liczony w minutach, czyli Stop - Start - Przerwania.

Przykładowy **rejestr błędów** pokazano na rys. 4. Dla każdego błędu zapisujemy datę jego wykrycia, kolejny numer (1, 2, ..) służący jako identyfikator błędu, typ błędu (o tym szerzej za chwilę), fazę pracy nad projektem, w trakcie której dany został popełniony, fazę, w której błąd usunięto i czas spędzony na usuwaniu tego błędu.

Program *Task Sche*

Data	Start	Stop	Przerwania	Delta	Faza	Komentarz
5.06	11:15	11:25	0	10	Planowanie	
	11:25	12:25	2+3	55	Projekt	telefony od szefa

Rys. 3. Przykładowy rejestr czasu pracy.

Program Task Sche

Data	Numer	Typ	Popełniono	Usunięto	Czas	Defekt źródłowy
6.06	10	80	Projekt	Test	52	
Opis: Wskaznik ElemPtr byl powiekszany o 1 raz za duzo w petli while						
	11	20	Test	Test	1	10
Opis: Blad w nazwie zmiennej popelniony przy korekcie bledu 10						

Rys. 4. Przykładowy rejestr błędów.

Może się zdarzyć, że poprawiając jeden błąd popełnimy kolejny. Aby umożliwić rejestrację takich przypadków, wprowadzono kolumnę **Defekt źródłowy**, w której zapisuje się numer błędów, którego naprawa spowodowała powstanie danego błędu.

Kategoryzacja błędów proponowana w ramach PSP jest oparta na pracy Chillarege'a i innych [8]. Wyodrębnia się tutaj 10 rodzajów błędów numerowanych co 10: błędy dokumentacji (10), błędy składniowe (20), błędy zarządzania wersjami i bibliotekami (30), błędy przypisania (40), błędy interfejsu (50), błędy dotyczące sprawdzania poprawności i informowania o błędach w programie (60), błędy danych (70), błędy w logice oprogramowania (80), błędy systemowe (90) i błędy środowiska programistycznego (100).

4. Proces planowania

Proponowany przez Humphrey'a proces planowania można w skrócie opisać następująco: po przeprowadzonej analizie wymagań i opracowaniu wstępnego projektu oprogramowania szacuje się jego rozmiar. Rozmiar ten służy do oszacowania czasu potrzebnego do wytworzenia oprogramowania. Z drugiej strony, w oparciu o dane historyczne i zaplanowane zdarzenia (takie jak np. urlop) obliczamy w sposób narastający ilość czasu jaką będziemy mogli poświęcić na wytworzenia oprogramowania. Mając z jednej strony oszacowaną ilość potrzebnego czasu a z drugiej czas, jaki mamy do dyspozycji możemy określić, kiedy zakończymy prace nad oprogramowaniem, a co więcej, możemy określić czas związany z „kamieniami milowymi" naszego projektu.

Do **szacowania rozmiaru oprogramowania** Humphrey proponuje metodę **PROBE** (*PROxy-Based Estimating*). W metodzie tej samo szacowanie jest poprzedzone przygotowaniem konceptualnego projektu oprogramowania. Projekt taki zawiera wykaz obiektów i ich metod (czyli podprogramów) oraz charakteryzuje ich funkcję w systemie. Następnie określa się dla każdej metody jej rozmiar w kategoriach: bardzo mały, mały, średni, duży, bardzo duży. Wiedząc jaki język programowania zostanie użyty do implementacji, jakiego rodzaju jest obiekt (np. przetwarzający tekst albo wykonujący obliczenia numeryczne), jaka jest kategoria jego rozmiaru (np. średni) i ile metod zawiera, można na podstawie danych historycznych oszacować jego rozmiar w sensie liczby linii kodu. Metoda PROBE nadaje się do szacowania zakresu prac także w przypadku re-użycia (ang. *reuseability*). Szacuje się wówczas łączną liczbę nowych i zmienianych linii kodu. Na podstawie regresji liniowej i danych historycznych otrzymuje się spodziewaną liczbę nowych i zmodyfikowanych linii kodu, a w oparciu o rozkład t Studenta i odchylenie standardowe oblicza się dolną i górną granicę rozmiaru dla założonego poziomu prawdopodobieństwa (np. jeśli dla prawdopodobieństwa 0.9 otrzymamy granice rozmiaru od 303 do 773 linii, oznacza to, że faktyczny rozmiar oprogramowania będzie należał do przedziału [303, 773] z prawdopodobieństwem 0.9). Oczywiście, aby korzystać z dobrodziejstw wnioskowania statystycznego programista nie musi się znać na statystyce matematycznej - te obliczenia można łatwo zaprogramować np. w arkuszu kalkulacyjnym.

Przejdźmy zatem do **szacowania czasu realizacji projektu**. W idealnym przypadku,

gdy mamy zgromadzone odpowiednie dane historyczne, powinniśmy mieć „w dorobku" przynajmniej 3 programy, dla których kwadrat współczynnika korelacji, r^2, między oszacowanym rozmiarem oprogramowania a czasem jego realizacji jest nie mniejszy niż 0.5. Możemy wówczas zastosować regresję liniową i obliczyć wartość oczekiwaną czasu realizacji projektu, jak też odpowiedni przedział ufności. Jeśli nie mamy do czynienia z idealnym przypadkiem ale mamy dane na temat korelacji między faktycznym rozmiarem oprogramowania a czasem potrzebnym na jego wytworzenie i $r^2 \geq 0.5$, to można oszacować czas projektu obliczając parametry regresji liniowej między faktycznym rozmiarem oprogramowania a czasem realizacji projektu. Również i w tym przypadku można policzyć przedział ufności, ale nie będzie on uwzględniał błędów wynikających z szacowania rozmiaru oprogramowania (wykorzystaliśmy związek między FAKTYCZNYM rozmiarem oprogramowania a czasem oprogramowania). W skrajnym przypadku możemy nie mieć nawet i takich danych. Jedyne co można w takiej sytuacji zrobić, to oszacować na podstawie danych historycznych wydajność liczoną w liniach kodu na godzinę i pomnożyć przez nią oszacowany rozmiar oprogramowania. Aby mieć pojęcie nt. błędu takiego oszacowania można oszacowany rozmiar pomnożyć raz przez najmniejszą, a drugi raz przez największą wydajność, jaką pokazują dane historyczne.

Przedstawioną metodę szacowania czasu można zastosować do różnych innych rodzajów zadań, np. do tworzenia dokumentacji użytkowej.

Przejdźmy teraz do **planowania zadań i terminów**. Na rys. 5 przedstawiono przykładowy formularz planowania terminów. Dla każdego tygodnia obliczamy liczbę efektywnych godzin, jakie możemy przeznaczyć na realizację projektu w wersji bezpośredniej oraz skumulowanej. Ważne jest by wziąć tutaj pod uwagę różnego rodzaju obciążenia dodatkowe (jak np. konieczność obsługi poczty elektronicznej) oraz urlopy i planowane wyjazdy i realistycznie oszacować czas, jaki mamy do dyspozycji. Ponadto, szeregujemy zadania według prawdopodobnego porządku ich realizacji i wpisujemy ich nazwy do formularza planowania zadań przedstawionego na rys. 6. Dla każdego zadania szacujemy czas

Projekt Task Sche Data: 30.05.1998

	Planowane			Faktyczne			
Tydzień	Godziny bezpośred.	Godziny skumulow.	Skumulow. wartość planowana	Godziny bezpośred.	Godziny skumulow.	Skumulow. wartość zarobiona	Skorygow. wartość zarobiona
1.06.98	20	20	5				
8.06.98	25	45	13.2				
15.06.98	15	60	20				

Rys. 5. Przykładowy formularz planowania terminów.

Projekt Task Sche Data:
30.05.1998

Zadanie	Plany					Fakty	
	Godziny	Skumulow. godziny	Planowana wartość	Skumulow. planowana wartość	Tydzień	Data	Zarobiona wartość
Projekt szczegół.	15	15	5	5	1.06		
Kod. modułu I/O	25	40	8.2	13.2	8.06		
Kod. modułu Graf	20	60	6.8	20	15.06		

Rys. 6. Przykładowy formularz planowania zadań.

jego wykonania i wpisujemy go do drugiej kolumny tego formularza. Do kolumny trzeciej wstawiamy skumulowane godziny. Porównując teraz trzecie kolumny obu formularzy jesteśmy w stanie wyznaczyć spodziewany termin zakończenia poszczególnych zadań (z dokładnością do tygodnia) i identyfikator tygodnia (zazwyczaj jest to data odpowiadająca poniedziałkowi) wpisujemy do kolumny **Tydzień** formularza planowania zadań.

5. Kontrola postępu prac

Mogłoby się wydawać, że mając przypisany dla każdego zadania spodziewany termin jego zakończenia sprawa kontroli jest trywialna: wystarczy sprawdzić pod koniec odpowiedniego tygodnia, czy zaplanowane zadanie zostało wykonane. Problem polega na tym, że kolejność wykonywania zadań może z różnych względów ulegać zmianie. Aby w tych warunkach móc śledzić postęp prac w projekcie PSP wykorzystuje znaną skądinąd metodę wartości zarobionej (ang. *Earned Value*). W metodzie tej każdemu zadaniu przypisuje się pewną liczbę punktów, jakie można zarobić realizując to zadanie. Punkty te powinny odzwierciedlać złożoność czasową zadania. Punkty otrzymuje się dopiero po zakończeniu realizacji danego zadania. Przykładowe wartości podano na rys. 6, w czwartej kolumnie. Przyjmując, że tygodnie są reprezentowane na osi x, a skumulowana wartość planowana i zarobiona na osi y, można graficznie obrazować postęp w pracach i w razie potrzeby przydzielać dodatkowe zasoby (patrz [2]).

Literatura

[1] W.S. Humphrey, A Discipline for Software Engineering, Addison-Wesley, Reading, MA, 1995.

[2] G. Prochowski, Zarządzanie projektami metodą Earned Value, X Górska Szkoła PTI Szczyrk'98, PTI - Oddział Górnośląski, 1998.

[3] B.C. Mc Rae, Praktyczne gospodarowanie czasem, M&A Communications Polska, Lublin, 1994.

[4] J. Rogers, Gospodarowanie czasem, **w:** D.M. Stewart (red.), Praktyka kierowania, PWE, Warszawa, 1996.

[5] L.R. Bittel, Krótki kurs zarządzania, PWN, Warszawa, 1994.

[6] T. Oleksyn, Sztuka kierowania, Wyższa Szkoła Zarządzania i Przedsiębiorczości, Warszawa, 1997.

[7] M.C. Paulk, B. Curtis, M.B. Chrisis, Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR, February, 1993.

[8] R. Chillarege *et al.*, Orthogonal Defect Classification - A Concept for In-Process Measurements, IEEE Trans. on Soft. Eng., vol. 18, no. 11 (1992), 943-956.

Andrzej Kobyliński

Katedra Informatyki Gospodarczej
Szkoła Główna Handlowa
Warszawa
e-mail: kobyl@sgh.waw.pl

Miary oprogramowania

1. Pomiar oprogramowania

W naukach przyrodniczych pomiary mają tradycje sięgające tysiącleci. W czasach nowożytnych standardowa miara długości - metr - ustanowiona została w 1799 r, chociaż nadal w wielu krajach powszechnie wykorzystywane są inne jednostki. Podobnie, wiele problemów związanych było z ujednoliceniem pomiaru temperatury. G. D. Fahrenheit wprowadził swoją skalę w 1725 r, A. Celsius w 1742, pamiętać też trzeba o skali Kelvina (1848). Z wieku XIX, okresu bujnego rozwoju nauk przyrodniczych, pochodzą często cytowane wypowiedzi koryfeuszy ówczesnej nauki:

"Nauka zaczyna się wtedy, kiedy zaczyna się mierzenie" (D. I. Mendelejew),

czy też:

"... kiedy można zmierzyć to, o czym się mówi i wyrazić to przy pomocy liczb, to wie się coś tym; lecz jeśli nie można tego zmierzyć, kiedy nie można tego wyrazić przy pomocy liczb, wiedza o tym jest uboga i niewystarczająca..." (W. Thomson (Lord Kelvin)).

Efektowne, chociaż złośliwe jest stwierdzenie Robertsa [ROBE79], który w swej książce o teorii pomiarów napisał:

"Podstawową różnicę między "dobrze rozwiniętymi" naukami, takimi jak fizyka a nieco słabiej "dobrze rozwiniętymi" naukami, takimi jak psychologia i socjologia, stanowi stopień, w jakim wykorzystywane są pomiary".

Pomiar oprogramowania jest ważnym tematem, zaliczanym do inżynierii oprogramowania. Inżynieria oprogramowania, z założenia, łączy w sobie dyscypliny informatyczne z naukami ekonomicznymi, społecznymi, o zarządzaniu. Mimo tego, że łączy w sobie tak odległe dziedziny chcielibyśmy, by (zgodnie z przyjętą dla tej dyscypliny nazwą) stosować w niej metody inżynierskie, a nie czystą intuicję. Dlatego nie ulega wątpliwości, że stosując pomiary, zarówno procesu tworzenia programowania, jak i produktów programowych, można przybliżyć tę dyscyplinę do grupy nauk ścisłych.

Tak więc pomiar oprogramowania może dostarczyć nam danych do ilościowych porównań i ocen zarówno samego oprogramowania, jak i procesu jego wytwarzania.

2. Miary (metryki) a modele oprogramowania

Miary (metryki) oprogramowania stosowane są już od 30 lat. Prawdopodobnie pierwsza praca dotycząca miar (brak w niej odwołań do wcześniejszych prac) ukazała się w 1968 r i dotyczyła próby pomiaru jakości [RUBE68]. W rzeczywistości tradycje stosowania miar są jeszcze dłuższe. W omawianym okresie powstało wiele miar, odnoszących się do różnych faz życia produktu.

Niektóre z proponowanych w literaturze przedmiotu miar są obiektywne (algorytmiczne), tzn. wartości tych miar dadzą się obliczyć dokładnie, zgodnie z przyjętym algorytmem, a uzyskiwane wartości są takie same, niezależnie od czasu, miejsca i osoby dokonującej pomiaru.

Nie wszystkie opisywane miary dadzą się obliczyć w tak precyzyjny sposób. Niektóre miary mają charakter subiektywny. Przede wszystkim są to te miary, które otrzymywane są w wyniku osobistej oceny pewnej cechy produktu, dokonywanej przez autora, jego zwierzchnika, eksperta, klienta czy użytkownika. Miary tego typu obarczone są wieloma wadami, ale jak inaczej wyrazić choćby "przyjazność oprogramowania", jeśli nie w sposób subiektywny ?

Bardzo często miary oprogramowania nie są używane w czystej postaci, lecz wiążąc ze sobą kilka miar tworzy się pewien model interesującej nas charakterystyki programu lub aspektu procesu programowania. W podobny sposób robi, przykładowo, importer gazu płynnego, który konstruuje sobie matematyczny model wyjaśniający (lub prognozujący) zużycie gazu, w którym to modelu zmiennymi objaśniającymi mogą być takie dane, jak stan i tempo przyrostu samochodów napędzanych gazem płynnym, pora roku (wahania sezonowe związane ze zmianami zapotrzebowania na gaz do butli turystycznych), akcje promocyjne zachęcające do ogrzewania gazowego itp. W przypadku tworzenia modeli związanych z rozwojem oprogramowania parametrami, które są używane jako zmienne objaśniające, są miary (metryki) oprogramowania.

W literaturze dotyczącej pomiarów oprogramowania pojęcia "miary" i "metryki" traktowane są jako synonimy i używane są wymiennie. Nie jest to całkiem ścisłe podejście. Z teorii pomiarów wynika, że aby używać terminu "metryka" wymagane jest zaistnienie określonych warunków, które nie zawsze są spełnione [ZUSE91]. Stosowanie terminu "metryka" w dziedzinie pomiarów oprogramowania stanowi zatem nadużycie. Dlatego też w dalszej części systematycznie stosowany będzie termin "miara".

3. Miary produktu i miary procesu

Dokonując ogólnej klasyfikacji miar, zwykle dzieli się miary oprogramowania na dwie podstawowe grupy. Są to miary produktu i miary procesu programowania.

Do pierwszej grupy zaliczane są te wszystkie miary, które odnoszą się do dzieła, będącego efektem pracy analityków, projektantów i programistów.

Miary procesu próbują opisać sam proces powstawania produktu, a także otoczenie, w jakim produkt powstaje. Do miar tego typu można zaliczyć, przykładowo, doświadczenie zespołu w posługiwaniu się pewnym narzędziem (mierzone latami lub miesiącami), długość stażu pracy programisty w danej firmie itp.

Przykłady kilku innych miar produktu i procesu zostaną krótko przedstawione w dalszej części.

4. Główne obszary zastosowań pomiarów oprogramowania

Zazwyczaj wyróżnia się 6 podstawowych obszarów zastosowań pomiarów oprogramowania. Zanim zostaną one wymienione trzeba podkreślić, że ich zakresy niekiedy pokrywają się. Miary produktywności, jakości, a szczególnie miary złożoności są do siebie bardzo zbliżone.

1. Modele estymacji kosztu.
2. Miary produktywności.
3. Modele jakości.
4. Modele niezawodności.
5. Miary złożoności.
6. Złożoność obliczeniowa (algorytmiczna)

Przedstawmy krótkie objaśnienie wymienionych obszarów zastosowań.

ad. 1. Modele te stosuje się celem predykcji kosztu produkcji oprogramowania już na wstępnych etapach cyklu życia systemu. Najlepiej znanym modelem estymacji kosztu jest COCOMO Boehma [BOEH81].

ad. 2. O ile koszt jest kategorią ekonomiczną, której jednostka miary jest dobrze zdefiniowana, to próba pomiaru produktywności jest trudniejsza. Produktywność oblicza się zgodnie z ogólnym wzorem:

$$\text{produktywność} = \frac{\text{wielkość oprogramowania}}{\text{nakład pracy [osobomiesiące]}}$$

przy czym wielkość oprogramowania może być mierzona na wiele różnych sposobów, na przykład jako liczba linii kodu źródłowego, liczba punktów funkcyjnych [ALBR79] lub, jak zaproponował Halstead [HALS77], jako pewna kombinacja operatorów i operandów w programie.

ad. 3. Osiąganie produktów wysokiej jakości jest głównym celem inżynierii oprogramowania. Nie istnieje i nie wydaje się, by była kiedykolwiek możliwa do wprowadzenia, jednostka miary jakości. Na jakość składa się, wg Boehma [BOEH76], nieco zmodyfikowana w normie ISO/IEC 9126 [ISO9126]:

- niezawodność,
- przenośność,
- efektywność,
- przyjazność,
- łatwość testowania,
- zrozumiałość,
- modyfikowalność.

Idealnie byłoby oczywiście, gdyby udało się osiągnąć maksymalne wartości w każdej z wymienionych kategorii, ale jest to niemożliwe do uzyskania ze względu na ich wzajemne uzależnienia. Dość oczywiste jest, że na przykład zwiększenie przenośności i przyjazności spowoduje zmniejszenie efektywności. Ale trzeba sobie zdawać sprawę z tego, że zwiększenie każdego z parametrów jakości powoduje zwiększenie kosztu produkcji.

ad. 4. Modele niezawodności oprogramowania są statycznymi modelami, służącymi do prognozowania średniego czasu między-awaryjnego lub oczekiwanego czasu do kolejnego uszkodzenia.

ad. 5. Złożoność programu definiuje się jako stopień trudności zrozumienia działania programu, określany wyłącznie na podstawie jego kodu źródłowego. Miary złożoności mają wiele wspólnego ze złożonością psychologiczną. Złożoność psychologiczna uwzględnia, obok złożoności kodu programu źródłowego, również czynnik ludzki oraz dostępny materiał informacyjny inny, niż sam program.

ad. 6. Złożoność obliczeniowa (algorytmiczna) odnoszona jest nie tyle do złożoności programu, co do efektywności algorytmu. Oceniany jest sam algorytm w konfrontacji z innymi algorytmami stanowiącymi rozwiązanie tego samego problemu.

5. Cel dokonywania pomiarów oprogramowania

Rozważając problem pomiarów oprogramowania, a dotyczy to szczególnie środowiska praktyków, odpowiedzieć trzeba na podstawowe pytanie:

Jakie mogą być korzyści ze stosowania pomiarów oprogramowania?

Miary oprogramowania mogą znaleźć zastosowanie:

1. jako podstawa do dokonywania różnego rodzaju estymacji,
2. do określania złożoności,
3. do śledzenia postępu w pracach nad systemem,
4. do analizy błędów,
5. przy podejmowaniu decyzji o tym, czy udało się nam już osiągnąć pożądany stopień jakości systemu,
6. w eksperymentalnym uwiarygodnianiu najlepszych praktyk programowania.

6. Przykłady miar i modeli, mogących mieć zastosowanie w różnych fazach cyklu życia oprogramowania

W czasie produkcji oprogramowania przez cały czas trzeba mieć na uwadze i kontrolować (oczywiście w miarę możliwości) 5 podstawowych parametrów:

- wielkość (mierzoną liczbą linii kodu, punktami funkcyjnymi lub innymi metodami),
- koszt (wyrażony w jednostkach pieniężnych),
- czas trwania (w miesiącach),
- pracochłonność (w osobomiesiącach),
- jakość (najtrudniejszą do zmierzenia i nie dającą wyrazić się pojedynczą wartością liczbową).

Ale oprócz tego, w każdej z faz cyklu życia systemu można stosować inne, specyficzne dla danego etapu, miary. Z upływem czasu wykorzystywane miary mogą być coraz bardziej szczegółowe, gdyż coraz bogatszy jest materiał, którym dysponujemy. Pokrótce przedstawimy niektóre z nich.

6.1. Faza określania wymagań

- Miarą tego, w jaki sposób pracuje zespół określający wymagania, jest liczba wymagań, które uległy zmianie na dalszych etapach rozwoju systemu. Jeśli duża jest liczba wymagań, jakie musiały ulec zmianie przy określaniu specyfikacji, projektowaniu i później, to nie ulega wątpliwości, że trzeba przyjrzeć się pracy tego zespołu.
- Jeśli w fazie określania wymagań powstał prototyp systemu, dobrze byłoby zmierzyć, które części prototypu są szczególnie często próbowane, a które z kolei są ignorowane przez wszystkich użytkowników. Jeśli na przykład okaże się, że wszyscy eksperymentatorzy co najmniej raz wypróbowali opcję D z menu głównego, a żaden z nich nie wszedł do opcji A, to oczywiste jest, że opcja D ma podstawowe znaczenie, należy poświęcić jej wiele uwagi i starać się dopracować ją i zoptymalizować; należy też zrobić badanie, czy opcja A jest naprawdę potrzebna, czy też można usunąć ją z ostatecznej wersji produktu.
- Miarą wielkości, możliwą do zastosowania w fazie określania specyfikacji, jest liczba stron dokumentu specyfikacji. Jeśli tę samą metodę pomiaru organizacja zastosuje do więcej, niż jednego produktu, różnice w wielkości specyfikacji mogą stanowić ważny wskaźnik w kolejnych fazach rozwoju produktu.
- Miarą, która może być przydatna w prognozowaniu wielkości przyszłego produktu, jest liczba pozycji w słowniku danych. Można tu zastosować kilka wskaźników, takich jak liczba plików, procesów itp.

6.2. Faza planowania

Mając ustaloną specyfikację systemu, trzeba zaplanować koszt i czas jego produkcji. Klient chce i musi wiedzieć, ile produkt będzie go kosztować i ile czasu będzie trwało jego opracowywanie. Niestety, nie jest to prosta czynność. Zbyt wiele zmiennych należy uwzględnić, przy czym podstawową trudność stanowi konieczność uwzględnienia czynnika ludzkiego. Z badań wynika, że różnice w produktywności programistów są ogromne, a w ekstremalnych stwierdzonych przypadkach stosunek ten może wynosić 1:28 [SCHA93]. Trzeba też mieć na uwadze, że kluczowe osoby zaangażowane w projekcie mogą chcieć zmienić pracę lub mogą się im przytrafić różne przypadki losowe.

U postaw zagadnienia estymacji kosztu i czasu trwania projektu leży problem: Jak mierzyć wielkość produktu? Ponieważ jest to obszerne odrębne zagadnienie, wspomnimy tylko w tym miejscu, że do miar wielkości zaliczamy:

- liczbę linii kodu,
- rodzinę miar *software science* Halstead'a [HALS77],
- punkty funkcyjne (oryginalne [ALBR79] oraz w wielu różnych mutacjach, np. MARK II, CHECKPOINT, ASSERT-R, SPQR/20, punkty charakterystyczne).

Dwie pierwsze metody są stosunkowo mało użyteczne, gdyż ich realne wartości możliwe są do uzyskania dopiero na stosunkowo późnych etapach w cyklu życia, stąd ogromna popularność trzeciej z wymienionych tu metod.

Próbując prognozować koszt rzeczywisty produktu zwykle stosuje się jedną z metod:

- opierając się na ocenie ekspertów, którzy w oparciu o swoje doświadczenie próbują, często bardzo trafnie, dokonać oszacowania,
- oceniając przez analogię, w oparciu o informacje o uprzednio zrealizowanych przedsięwzięciach, i to zarówno w firmie podejmującej zadanie, jak i w innych firmach o podobnym charakterze,
- w oparciu o algorytmiczne modele estymacji kosztu - przewidywana wielkość produktu (mierzona liczbą linii lub punktów funkcyjnych) stanowi podstawę do budowy modelu prognozującego koszt produkcji systemu. Modeli tego typu powstało wiele, ale najbardziej znany jest wspominany już model COCOMO.
- bottom-up (wstępującą), gdy cały projekt dzieli się na mniejsze zadania, których koszt oszacować jest łatwiej.

6.3. Faza projektowania

Nie zostały do tej pory opracowane i powszechnie zaakceptowane miary, standardowo wykorzystywane w tej fazie. Należy oczywiście

kontrolować 5 miar podstawowych, można też posłużyć się następującymi wskaźnikami:

- bardzo ogólną miarą wielkości produktu docelowego jest liczba modułów w systemie,
- siła więzi modułowych i mocy modułów jest miarą jakości projektu,
- w projekcie szczegółowym jako miara złożoności modułu może być użyta tzw. miara złożoności cyklomatycznej McCabe'a, wyrażana przez liczbę dróg w module i łatwa do obliczenia jako liczba podejmowanych w module decyzji binarnych + 1,
- jeżeli cały projektowany system przedstawić w formie grafu, w którym wierzchołki odpowiadają modułom systemu, a krawędzie symbolizują przepływ informacji pomiędzy modułami, to Henry i Kafura [HENR81] proponują, by złożoność modułu obliczać zgodnie z wzorem:

$$długość \times (fan\_in \times fan\_out)^2,$$

gdzie:

długość - dowolna miara wielkości modułu,
fan_in - liczba danych przekazywanych do modułu,
fan_out - liczba danych wysyłanych z modułu.

6.4. Fazy implementacji i integracji

Począwszy od fazy implementacji dostępny staje się kod źródłowy programów. W decydujący sposób ułatwia to stosowanie miar, gdyż od tej chwili jesteśmy w stanie dokonywać pomiarów obiektywnych. Miarami, które mogą być stosowane począwszy od tego etapu są wszystkie miary złożoności. Są to miary obiektywne, obliczane w sposób algorytmiczny na podstawie kodu źródłowego oprogramowania. Im poświęcony będzie cały kolejny rozdział.

7. Miary złożoności

Na złożoność programów komputerowych mają wpływ cztery podstawowe czynniki:

- wielkość programu,
- struktura i przepływ sterowania w programie,
- sposób użycia danych,
- styl kodowania.

Zaprojektowane w minionych latach miary w większości uwzględniają wyłącznie jeden aspekt całego problemu złożoności. Oto lista najpopularniejszych miar złożoności.

⇒ **Miary wielkości**

- liczba
  - linii kodu źródłowego,
  - linii kodu źródłowego bez pustych linii i komentarzy,
  - instrukcji programu,
- miary *software science* Halstead'a (rodzina kilkunastu miar)[HALS76].

⇒ **Miary sterowania**

- miara złożoności cyklomatycznej McCabe'a [MCCA76],
- absolutna i względna złożoność logiczna Gilba,
- miara złożoności MIN Chena,
- miara złożoności zakresu SCOPE Harrisona i Magela,
- miara złożoności poziomu zagnieżdżenia ℵ Piwowarskiego,
- miara liczby przecięć w tekście programu KNOTS Woodwarda, Hennella, Hedleya.

⇒ **Miary danych**

- rozpiętość między odwołaniami do danych - SPAN Elshoff'a,
- średnia liczba odwołań do danych w instrukcji,
- para: segment - zmienna globalna,
- miara złożoności Q Chapina.

Literatura

[ALBR79] Albrecht A. J., "Measuring Application Development Productivity", *Proceedings of the IBM SHARE/GUIDE Applications Development Symposium,* Monterrey (CA), October 1979, str. 83-92.

[BOEH76] Boehm B. W., "Software Engineering", *IEEE Transactions on Computers* **C-25** (December 1976), str.1226-1241.

[BOEH81] Boehm B. W., *Software Engineering Economics*, Prentice-Hall, Englewoods Cliffs, NJ, 1981.

[HALS77] Halstead M. H., *Elements of Software Science*, Elsevier North-Holland, New York, 1977.

[HENR81] Henry S. M., Kafura D., "Software Structure Metrics Based on Information Flow", *IEEE Transactions no Software Engineering* **SE-7** (September 1981), str. 510-518.

[ISO9126] *International Standard ISO/IEC 9126: Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for Their Use*, Geneva, 1991.

[MCCA76] McCabe T. J., "A Complexity Measure", *IEEE Transactions no Software Engineering* **SE-2** (December 1976), str. 308-320.

[ROBE79] Roberts F. S., *Measurement Theory with Applications do Decisionmaking, Utility and the Social Sciences*, Addison Wesley, 1979.

[RUBE68] Rubey R. J., Hartwick R. D., "Quantitative Measurement Program Quality", *Proceeding of the National Computer Conference,* ACM, 1968, str. 671-677.

[SCHA93] Schach S. R., *Software Engineering (Second Edition)*, Irwin and Aksen, Homewood (IL) and Boston (MA), 1993.

[ZUSE91] Zuse H., *Software Complexity. Measures and Methods*, Walter de Gruyter, Berlin, New York, 1991.

Tadeusz Wilusz

Katedra Informatyki Akademii Ekonomicznej w Krakowie

Edukacja informatyczna jako ważny kontekst dyskusji o jakości oprogramowania

STRESZCZENIE

Wychodząc z jawnie oczywistej zasady, że o jakości realizacji jakiegokolwiek przedsięwzięcia w głównej mierze decydują umiejętności i kwalifikacje jego realizatorów trudno, w kontekście dyskusji o problemach wytwarzania wysokiej jakości oprogramowania, pominąć pytanie o aktualny kształt edukacji informatycznej w Polsce. Właśnie szeroko rozumianej edukacji a nie wyłącznie specjalistycznego kształcenia kadr informatyki. W intencji autora referat ma stanowić bilans otwarcia do jak najszerszej dyskusji nad zagadnieniem, którego strategiczną rangę dla zasadniczego tematu konferencji trudno jest przecenić

Wprowadzenie

Ranga i znaczenie systemu edukacji zarówno dla rozwoju każdego kraju jak i warunków życia jego społeczeństwa jest bezsporna i bezdyskusyjna. Wielce dyskusyjnym natomiast zawsze był, jest i niewątpliwie nadal będzie kształt tegoż systemu zarówno w aspekcie organizacji procesu nauczania jak i sformułowania celów, doboru treści, form oraz technologii kształcenia. Warto sobie w tym miejscu uświadomić, że jest to stan normalny i prawidłowy, który jedynie dowodzi troski zabierających głos w dyskusji o dalsze losy kraju[1] Wątpliwości zaczynają się pojawiać dopiero wtedy, gdy zaczynamy szukać realnych efektów prowadzonych dyskusji.

Zwykle ze zdumieniem stwierdzamy, że proces dostosowywania systemu kształcenia do zmian zachodzących w otoczeniu i realizacji zweryfikowanych w toku wielostronnych dyskusji postulatów zmian wykazuje nieprawdopodobną wręcz inercję, która czyni zachodzące zmiany trudno dostrzegalnymi i może najzagorzalszych zwolenników zmian na lepsze doprowadzić do głębokiej frustracji. Czy zatem warto kontynuować jawiący się jako syzyfowy trud poszukiwania lepszej formuły edukacyjnej i czy warto poświęcać temu czas na spotkaniu ludzi zainteresowanych bardzo konkretnym problemem, w tym przypadku jakością oprogramowania? Część pierwsza pytania jest ewidentnie retoryczna a uwypuklona sprzeczność - pozorna. Wystarczy bowiem sformułować cel dyskusji o kształt edukacji jako poszukiwanie optymalnych rozwiązań w jednej z najważniejszych dziedzin dla prawidłowego rozwoju kraju aby sformułowaną wątpliwość zamienić na poczucie obowiązku. Część druga równie łatwo da się rozstrzygnąć na „tak", gdyż oprogramowanie jest wytworem ludzi, a ludzi ... trzeba wykształcić w kulturze społeczeństwa dbającego o jakość efektów swojej pracy. Spory o optymalny kształt segmentu informatycznego systemu edukacji mają tak bogaty dorobek, że powtarzanie już zgłoszonych i dokładnie uzasadnionych propozycji nie ma większego sensu. Ma natomiast głęboki sens sięgnięcie do tego

[1] O czym wiedział już Stanisław Staszic pisząc słowa „Takie będą Rzeczpospolite, jakie ich młodzieży chowanie".

dorobku[2] i zadanie sobie trudu udzielenia odpowiedzi na pytanie, które z proponowanych rozwiązań, zgłoszonych zastrzeżeń i postulatów zmian w stosunku do obowiązującej praktyki mają najistotniejsze znaczenie dla osiągnięcia celu sformułowanego jako „podniesienie jakości oprogramowania". Tak sformułowane zadanie tylko z pozoru wydaje się trywialnym. Problem w tym, że rzeczy „najważniejszych" jest zwykle niewiele a bogactwo proponowanych rozwiązań olbrzymie. Dlatego też uzgodnienie możliwie krótkiej listy priorytetowych uwarunkowań podnoszenia jakości oprogramowania tkwiących korzeniami w obowiązującym systemie edukacji ośmielam się uważać za ważne i odpowiedzialne zadanie tejże Konferencji. Poniżej pozwalam sobie jako pierwszy zabrać głos w tej sprawie.

Edukacja informatyczna czy kształcenie informatyków?

Uważam, że w kontekście dyskusji nad warunkami decydującymi o jakości oprogramowania (użytkowego) kapitalne znaczenie ma nie utożsamianie tych dwóch pojęć. W przeciwnym przypadku łatwo o efekt marnotrawstwa, na który zwraca uwagę między innymi pan Sienkiewicz [13] pisząc: "próby dostosowywania się do nowej sytuacji prowadzą niekiedy do powielania przez wyższe uczelnie programów nauczania szkół średnich". Gdyby tylko o takie marnotrawstwo chodziło! Nie od dzisiaj wiadomo, że „czym skorupka za młodu nasiąknie tym na starość trąci" co przekłada się na powszechnie znaną konstatację, że dużo trudniejszym (kosztowniejszym), a czasami w pełni niemożliwym, jest pozbycie się złych nawyków aniżeli opanowanie nowych umiejętności. Jeśli potraktujemy system edukacji informatycznej w kategoriach układu z jednym wejściem (aktualnie to chyba już poziom przedszkola) i jednym wyjściem (kursy wąsko specjalizowanego doskonalenia zawodowego) to wniosek iż kształcenie informatyków znajduje się „w środku" jest ewidentny". Podobnie zresztą jak ogólnie znany wniosek, że im wcześniejsza faza popełnienia błędu tym większy koszt jego usunięcia. Zwraca na to uwagę fundamentalny dokument jak „Raport I Kongresu Informatyki Polskiej"[3], ale w tym dokumencie zebrano tyle ważkich problemów, że decyzja, który z nich jest najważniejszy należy do odbiorcy. Tymczasem u odbiorcy istnieje naturalny odruch projekcji na obszar jego aktywności zawodowej i dlatego od lat dyskusje szczegółowe toczą się na równoległych płaszczyznach poszczególnych stopni systemu edukacji: odrębnie na poziomie szkolnictwa podstawowego, oddzielnie średniego i ,konsekwentnie, oddzielnie na poziomie szkół wyższych. W efekcie możemy oczekiwać, że mogą powstać na każdym z tych systemów dobre rozwiązania, które łącznie nie będą stanowić niczego, co zasłuży na miano „systemu edukacji informatycznego" we właściwym tego słowa znaczeniu.

Problem kształcenia kadr informatyki pojawia się dopiero na poziomie szkolnictwa średniego (profilowane szkoły średnie, zawodowe studia policealne) i rozkwita na poziomie szkolnictwa wyższego. Załóżmy na chwilę, że mamy „idealny" program i realizację kształcenia informatyków. A co z użytkownikami, których wiedza o technologii informacji została zakonserwowana na poziomie szkoły podstawowej[3].? Czy ma szansę powstać wysokiej jakości oprogramowanie w zespole złożonym ze świetnych informatyków i świetnych specjalistów dziedzinowych (np. bankowców), którzy nie są w stanie się porozumieć a co gorsza, ci ostatni mogą mieć „w genach" lęk przed wszechpotężną technologią, którego nabawili się „po drodze" do doskonałości w swoim zawodzie, bo nauczyciel w szkole średniej był „domorosłym" informatykiem i jedyne co po sobie pozostawił to przekonanie iż wystarczy kupić

[2] Bogactwo opublikowanych głosów w dyskusji na temat szeroko rozumianej edukacji informatycznej stało się swoistym problemem przy powstawaniu tego tekstu. Był to oczywiście klasyczny problem wyboru, gdyż pełna bibliografia prawdopodobnie nie zmieściłaby się w dopuszczalnej dla całego referatu objętości. W rezultacie w zamieszczonym wykazie źródeł znalazły się jedynie: a) opracowania kompleksowe o fundamentalnym dla tematu znaczeniu jak [3] b) pozycje zawierające rozwinięcie prezentowanych poglądów (np. [7]) c)pozycje łatwo dostępne (czyli „internetowe" z podanymi w nawiasach kwadratowych URL'ami). Wybór był w pełni arbitralny i niewątpliwie można zaproponować lepszy.

[3] Z jakimi problemami boryka się edukacja informatyczna można zapoznać się zaglądając za pośrednictwem internetu do dorobku „I Konferencji IdS" [2].

komputer aby świat uczynić lepszym?

Konkluzja nasuwa się sama: **Istnieje z dnia na dzień coraz bardziej pilna potrzeba zaprojektowania systemu edukacji informatycznej da capo al fine jeśli chcemy w określonej perspektywie wytworzyć stabilne warunki do opracowywania rozwiązań zasługujących na etykietkę „High Quality Product" respektowaną pod każdą szerokością geograficzną.**

Sam projekt zwykle nie wystarcza. Potrzebni są jeszcze fachowi jego wykonawcy. I tak dochodzimy do problemu, któremu, zgodnie z zasadami elementarnej logiki jestem skłonny przyznać drugą lokatę na liście rozważanych priorytetów. Jest nim oczywiście problem kadry dydaktycznej:

Dydaktycy przedmiotów informatycznych pilnie poszukiwani

I trzeba dodać: w aktualnych warunkach prawie, że nie do znalezienia. Ponieważ sam jestem dydaktykiem w dziedzinie informatyki i mogę być posądzany o stronniczość w wygłaszanych poglądach postanowiłem posłużyć się cytatem ze „wstępniaka" Informatyki nr 4/96 [4] autorstwa pani Krystyny Karwickiej. Autorka pisze w nim:

„ (...) Wszystkie wyższe szkoły, zwłaszcza państwowe, cierpią na niedobór kadry dydaktycznej. W „Raporcie kongresowym" zostało to określone w sposób jasny: Najpoważniejszym zagrożeniem dla przyszłości rozwoju informatyzacji w Polsce jest zmniejszenie się liczebności kadr w uczelniach i obniżenie poziomu nauczania oraz brak motywacji finansowych dla nowych kadr. Wszystkie uczelnie alarmują, że liczebność kadry dydaktycznej w zastraszającym tempie ubywa. Przyczyna jest bardzo prosta: absolwent, a nawet student kierunku informatycznego „na dzień dobry" otrzymuje w firmie pensję nieraz i trzykrotnie przewyższającą apanaże pracowników naukowych. Wśród osób, które decydują się podjąć pracę dydaktyczną, ze świeczką trzeba szukać takiej, która by nie dorabiała poza uczelnią różnymi sposobami do akademickiej pensyjki. W niektórych specjalnościach najbardziej na rynku pracy deficytowych, takich jak analitycy systemów, integratorzy, administratorzy sieci - uczelnie nie są w stanie znaleźć chętnych do prowadzenia zajęć dydaktycznych. Zresztą i w firmach są oni na wagę złota. Chociaż coraz powszechniej informatyką interesuje się młodzież szkół średnich, a także podstawowych, coraz trudniej jest o nauczycieli informatyki. Powody są takie same jak w przypadku wyższych uczelni; śmiesznie małe, w porównaniu z oferowanymi przez firmy, płace."

Nic dodać, nic ująć. Jeśli program nauczania jest świetny a wykonawców brak, to po co nam program? I czy w takiej sytuacji coś co nie ma szans działać można nazwać systemem (edukacji w tym przypadku)?

Konkluzja. **Jeśli chcemy mieć wysokiej jakości produkty to musimy mieć sprawny i wysokiej jakości system edukacji. Ergo, nie ruszymy z miejsca jeśli nie podniesiemy konkurencyjności profesji dydaktycznej na rynku pracy.**

Załóżmy na chwilę, że mamy świetny program i rozwiązaliśmy palący problem kadry dydaktycznej. Co z tych dwóch podstawowych elementów jest w stanie uczynić sprawnie działający system o założonych i oczekiwanych cechach? Odpowiedź znów wydaje się być oczywista: to co elementy każdego systemu wprawia w ruch, czyli zasady działania, alias reguły. Jeśli ma to być jeden system to musi być oparty o spójny zespół reguł. W ten sposób docieramy do pozycji numer trzy na mojej liście priorytetów, czyli standaryzacji organizacji procesu nauczania.

Reguły i standardy

Rzecz ciekawa, że ten problem jawi się często jako najłatwiejszy a w praktyce częstokroć stanowi przysłowiowe ziarnko piasku zdolne zatrzymać całą maszynerię. Rzecz równie ciekawa, że termin"

standaryzacja" znacznie częściej jest rozumiana jako postulat ujednolicenia dokumentów i programów (vide „minima programowe") aniżeli reguł działania. Tymczasem tego co jest podstawą każdego i niezawodnego systemu jest prosty, dający się powszechnie rozumieć, a co za tym idzie i kontrolować, system reguł. Najłatwiej będzie mi tę tezę zilustrować odwołując się znowu do przykładu, tym razem do podsumowania wyników badań nad stanem edukacji informatycznej w uczelniach ekonomicznych opublikowanych w pracy[10].

> *„Stwierdzenie, że dyskusja na temat edukacji informatycznej może być tylko wtedy efektywna a kształtujące ją decyzje racjonalne gdy dysponuje się rzetelnym i pełnym obrazem aktualnego stanu zarówno jej jak i otoczenia jest teoretycznie oczywistością. Jednakże, nawet pobieżna, analiza zebranych danych nasuwa poważną wątpliwość, czy teoria pokrywa się z praktyką. Jeśli bowiem potraktować studia wyższe w kategoriach procesu, który absolwenta szkoły średniej ma wyposażyć w wiedzę i umiejętności poszukiwane na rynku pracy, to rzucające się w oczy zróżnicowanie programów edukacji informatycznej w poszczególnych uczelniach jest zbyt duże. Zatem zakładając, że programy te na poszczególnych uczelniach zostały skonstruowane i są dostosowane do otoczenia można dojść do absurdalnego wniosku, że poszczególne uczelnie funkcjonują w różnych rzeczywistościach. Dużo bardziej prawdopodobnym wyjaśnieniem jest przypuszczenie, że decyzje programowe były podejmowane w oparciu o nieaktualne lub niekompletne dane.*
>
> *Ogólnie rzecz ujmując brak spójności programów kształcenia informatycznego w obrębie segmentu rynku edukacyjnego jaki stanowią uczelnie ekonomiczne ma duży, negatywny wpływ na efektywność procesu kształcenia komplikując, wiele typowych procedur i podnosząc tym samym koszty kształcenia i to, zarówno materialne jak i społeczne. Na przykład - w dość typowej sytuacji zmiany uczelni przez studenta, decyzja, które przedmioty z poprzedniej uczelni mogą mu być zaliczone, a które student musi uzupełnić, tak aby być w zgodzie z wymogami nowego środowiska jest aktualnie, delikatnie mówiąc, bardzo trudna.*
>
> *Powyższe uwagi w żadnym wypadku nie mogą być traktowane jako tęsknota za uniformizacją, ale są wskazówką, że do całego zagadnienia należy podchodzić w sposób systemowy w tym sensie, że zmieniając określone reguły (np. odchodząc od scentralizowanego systemu kształtowania programu studiów) należy je zastąpić funkcjonalnie pełnym zestawem nowych mechanizmów, które są w stanie sprostać nowym, nie istniejącym uprzednio wyzwaniom"*

Badania były przeprowadzone w 1994 roku ale w międzyczasie nic istotnego w materii podmiotu badań nie zaszło i sygnalizowany problem nadal istnieje i jest „tolerowany".

Przechodząc do konkluzji można ją sformułować w postaci warunku koniecznego, że potrzebne jest rozwiązanie, w którym obowiązują cztery (w znaczeniu nieliczne) reguły, które pozwalają rozstrzygać setki przypadków, które zastąpi „działający system" oparty na tysiącach reguł, po jednej oddzielnie dla każdego przypadku.

Zamiast podsumowania

Intencją przedstawionych wyżej rozważań było uzasadnienie tezy postawionej w tytule i rozwiniętej w treści referatu. Jeśli tak sformułowany cel został osiągnięty to pozostaje zaproszenie do dyskusji w toku konferencji, po której będzie właściwy czas na podsumowanie.

Wybrane źródła:

[1] Abt S.: „Kształcenie menedżerów - użytkowników systemów informatycznych", Materiały konferencji „Informatyka w wyższych uczelniach dla gospodarki narodowej", Politechnika Gdańska, Gdańsk 1994.

[2] I Konferencja IdS we Wloclawku [http://rhea.ids.pl/ids/konferencja/]

[3] I Kongres Informatyki Polskiej,[http://www.agh.edu.pl/PTI/kongres.html]

[4] Karwicka Krystyna. „Kadra naukowa pilnie poszukiwana",INFORMATYKA, nr 4, 1996. [http://www.pol.pl/news._p/informa/4-96.htm]

[5] Kolbusz E., Olejniczak W.: „Problemy informatyki ekonomicznej", Informatyka, 7/1994.

[6] Kuraś M., Wiluszt T.: „Cele i treści kształcenia w zakresie informatyki w uczelniach ekonomicznych", Informatyka, Zeszyty metodyczne nr 3, COMSE-SGPiS, Warszawa 1987.

[7] Kuraś M., Wilusz T.: „Kilka uwag na temat kształcenia informatyków", INFOGRYF'96, Materiały konferencyjne, tom II, Kołobrzeg październik 1996.

[8] Kuraś M.: „Kształcenie analityków systemów informacyjnych", Informatyka, 1993/7.

[9] Motylewski Tomasz: "dyskusja nad minimum programowym" ,http://www.ch.uj.edu.pl/plchem/chinf/mp/mp1.html

[10] Nowarska B., Wiluszt T. „Informatyka w programach studiów ekonomicznych w Polsce", kwartalnik „Komputer w edukacji", Instytut Informatyki Uniwersytetu Wrocławskiego, nr 3-4/95, Wrocław, 1996.

[11] Postępski Dawid: "Edukacyjne wykorzystanie Internetu z pozycji ucznia",http://rhea.ids.pl/ids/konferencja/referaty/ref2_2.html

[12] Rożniatowski Stanisław: "Co nauczyciele wiedzą o Internecie?",http://rhea.ids.pl/ids/konferencja/referaty/ref2_1.html

[13] Sienkiewicz P.: „Dylematy informatycznego kształcenia", Informatyka, 7/1994.

[14] „Strategia Rozwoju Informatyki w Polsce. Stan, Perspektywy, Zalecenia",Raport Kongresowy opracowany na podstawie wyników prac I Kongresu Informatyki Polskiej, który odbył się w dniach 1-3 grudnia 1994 roku w Poznaniu, Warszawa 1995 [http://www.agh.edu.pl/PTI/tekst.html]

[15] Wilusz T.: „Intranet w uczelni", INFOGRYF'96, Materiały konferencyjne, tom II, Kołobrzeg październik 1996.

[16] Wilusz T.: „Usługi sieciowe w dydaktyce", Zeszyty Naukowe Akademii Ekonomicznej w Krakowie, Prace z zakresu informatyki, nr 474, Kraków 1996.

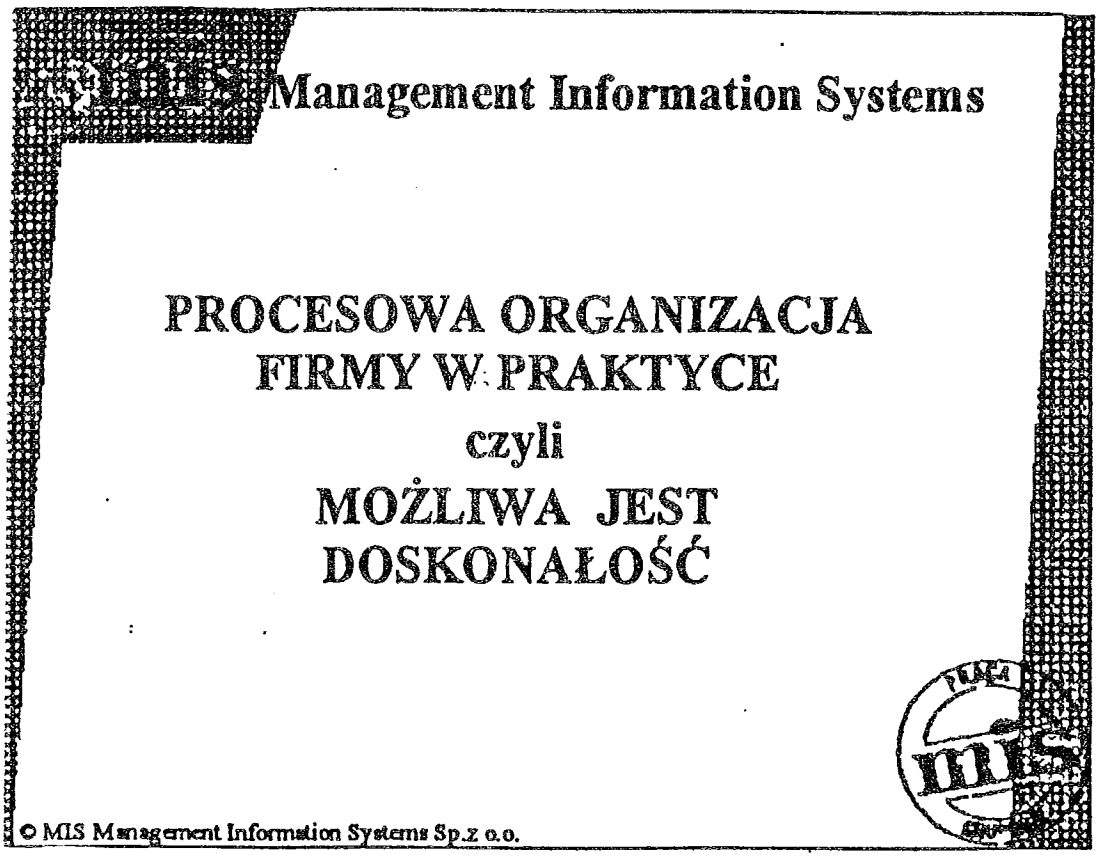
[17] Witryna dedykowana praktycznym aspektom budowy systemu zarządzania jakością [http://clover.com.au/ISO9000/]

Bożena Skibicka

MIS

Management Information Systems
Spółka z o.o.

Procesowa organizacja firmy w praktyce



Management Information Systems

PROCESOWA ORGANIZACJA FIRMY W PRAKTYCE
czyli
MOŻLIWA JEST DOSKONAŁOŚĆ

© MIS Management Information Systems Sp.z o.o.

Agenda:

1. **Nowe trendy w zarządzaniu**
2. **Zasady działania i korzyści płynące z korzystania z mis-Partnera**
3. **Pokaz praktyczny**

© MIS Management Information Systems Sp.z o.o.

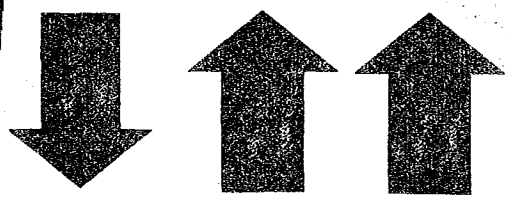
REENGINEERING -
- WORKFLOW -
- CLIENTING -**

nowe - magiczne - niemagiczne trendy
czyli gdzie szukać rezerw poprawy efektywności?
Procesowe zarządzanie firmą:

- **Odejście od tradycyjnych schematów organizacyjnych;**
- **Kiedyś tylko w nauce - obecnie wkracza do biznesu;**
- **Możliwe dzięki upowszechnieniu technik teleinformatycznych;**

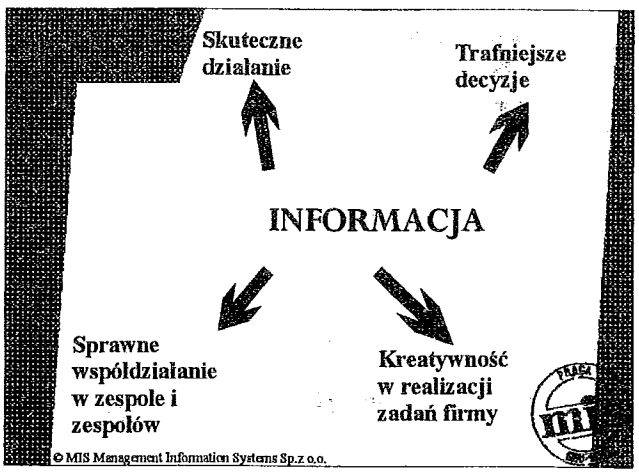
**** Przedruk artykułu z Informatyki 09/97 w materiałach.**

Tendencje w inwestycjach informatycznych



Infrastruktura informatyczna | Wydajność | Zwrot nakładów inwestycyjnych

© MIS Management Information Systems Sp.z o.o.



© MIS Management Information Systems Sp.z o.o.

Oczekiwania użytkowników wobec systemów informatycznych

- **mają działać szybciej**
- **mają być tańsze**
- **mają być przyjazne i niezawodne**
- **mają wspierać osiąganie doskonałości**

© MIS Management Information Systems Sp.z o.o.

Te oczekiwania mogą zostać zaspokojone

Wszystko staje się możliwe dzięki nowym technologiom informatycznym. Takim jak Lotus Notes i dedykowane oprogramowanie (aplikacja), które rozwiązują kluczowe problemy i wspomagają zachodzące w firmie procesy.

© MIS Management Information Systems Sp.z o.o.

Zjawiska, z którymi żyjemy na co dzień:

W firmie małej lub dużej, nowej lub z tradycjami zdarza się, że:

- ▷ każdy ma inne dane o kontaktach z tymi samymi klientami, często nieaktualne;
- ▷ pracownik dowiaduje się od klienta o nieznanych mu działaniach swojej firmy;
- ▷ coś nie zostało zrobione, bo wszyscy sądzili, że zrobił to ktoś inny;
- ▷ nieznany jest stan realizacji określonego zadania, a odpowiednie działania podejmowane są zbyt późno;
- ▷ obsługa zdarzenia jest niemożliwa z powodu nieobecności pracownika;

Potrzeba: sprawny proces obsługi klienta w relacji wielu klientów <--------> wielu pracowników firmy

© MIS Management Information Systems Sp.z o.o.

Spróbujmy pomarzyć (wszakże chcemy działać doskonale):

Wyobraźmy sobie, że można:

- ▷ spowodować, by zadania przydzielone pracownikom zostały wykonywane w terminie;
- ▷ kontrolować na bieżąco realizację zleconych zadań;
- ▷ sprawdzić status określonego problemu, przedsięwzięcia, dokumentu;
- ▷ spowodować by potrzebna informacja dotarła jednocześnie do wszystkich zainteresowanych

© MIS Management Information Systems Sp.z o.o.

Marzymy dalej...

Możemy:

- ▷ zapewnić analizę kontaktów z klientami;
- ▷ zapewnić monitorowanie aktywności firmy i poszczególnych pracowników;
- ▷ spowodować, by kontakty handlowe stały się własnością firmy, a nie pojedyńczego pracownika;
- ▷ zapewnić załatwienie każdej sprawy mimo nieobecności prowadzącego ją pracownika;
- ▷ zopobiegać ginięciu w firmie informacji, dokumentów, faksów, itd.

© MIS Management Information Systems Sp.z o.o.

Partner4

Sposób na pracę Twojego biura

Struktura Aplikacji

Hierarchiczna archiwizacja dokumentów i załączników

© MIS Management Information Systems Sp.z o.o.

Firmy i pracownicy (kontrahenci)

- Hierarchiczna baza firm, oddziałów, pracowników i osób prywatnych
- Wspólne dane o kontrahentach dla wszystkich użytkowników
- Szerokie możliwości grupowania i wyszukiwania

© MIS Management Information Systems Sp.z o.o.

Przedsięwzięcia

- Planowanie przedsięwzięć i kontrola ich realizacji
- Określanie kontrahentów i ich ról w przedsięwzięciu
- Przydzielanie użytkownikom opieki nad przedsięwzięciem

© MIS Management Information Systems Sp.z o.o.

Kontakty

- Rejestracja przeprowadzanych rozmów telefonicznych i spotkań
- Możliwość odtworzenia historii kontaktów z kontrahentem
- Powiadamianie użytkowników o ustaleniach dokonanych podczas rozmów z kontrahentami

© MIS Management Information Systems Sp.z o.o.

Korespondencje Wychodzące

- Tworzenie i przechowywanie dokumentów we wspólnej bazie danych
- Łatwy dostęp do dokumentów tworzonych przez innych użytkowników
- Wysyłanie faksów bezpośrednio z aplikacji
- Indywidualne określanie praw modyfikacji i czytania

© MIS Management Information Systems Sp.z o.o.

Korespondencje Przychodzące

- Przechowywanie dokumentów faksowanych lub skanowanych
- Dostarczanie korespondencji bezpośrednio na "biurka" użytkowników
- Łatwy dostęp do korespondencji z dowolnego okresu

© MIS Management Information Systems Sp.z o.o.

Korespondencje Seryjne

- Łatwe tworzenie i wysyłanie korespondencji i faksów seryjnych
- Elastyczny wybór adresatów
- Rejestr korespondencji wysyłanych

© MIS Management Information Systems Sp.z o.o.

Wewnętrzne

- Tworzenie, przechowywanie i obieg dokumentów tworzonych w biurze
- Tworzenie bazy wiedzy przedsiębiorstwa
- Dostępność do wyników pracy innych użytkowników
- Biblioteka obiektów i plików

© MIS Management Information Systems Sp.z o.o.

Zadania

- Przydzielanie zadań do wykonania dla użytkowników
- Powiązanie wykonywanych prac z przedsięwzięciami i kontrahentami
- Kontrola wykonania zadań
- Kontrola terminów wykonania zadań

© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - indywidualny

- Określanie trasy i typu obiegu dla dokumentu
- Możliwość przechowywania wszystkich wersji dokumentu
- System powiadamiania i ponaglania pocztą elektroniczną
- Kontrola terminów

© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - określony

- Korzystanie z biblioteki marszrut dokumentów określonych w firmie
- Przestrzeganie określonego schematu decyzyjnego przedsiębiorstwa
- Zatwierdzanie i odrzucanie wersji dokumentów
- Biblioteka procedur postępowania

© MIS Management Information Systems Sp.z o.o.

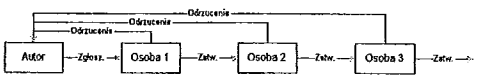
Obieg dokumentów - typy obiegu

- obieg szeregowy
- obieg równoległy

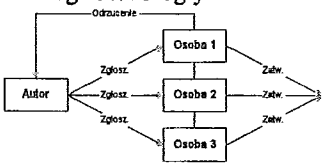
- odrzucenie/zatwierdzenie dokumentu
- praca nad dokumentem
  - modyfikacje z historią zmian
  - modyfikacje bez historii zmian

Obieg dokumentów - odrzucenie/zatwierdzenie

- Obieg szeregowy



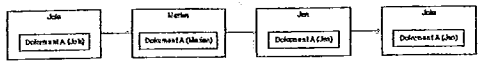
- Obieg równoległy



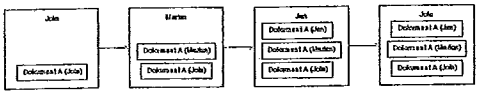
© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - historia zmian

- Szeregowa modyfikacja oryginału (bez historii)



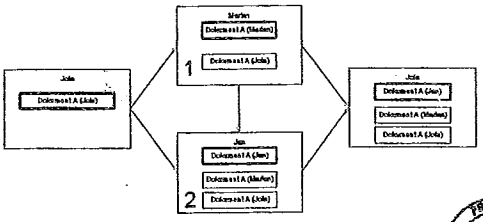
- Szeregowa modyfikacja oryginału (z historią zmian)



© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - historia zmian

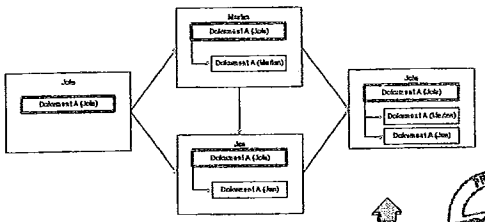
- Równoległa modyfikacja z historią zmian wg zasady "pierwszy lepszy"



© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - historia zmian

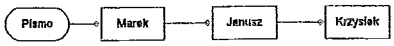
- Równoległa modyfikacja z zachowaniem oryginału (zmiany - odpowiedzi)



© MIS Management Information Systems Sp.z o.o.

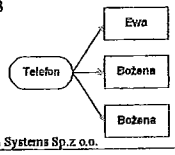
Obieg dokumentów - możliwości

- Obieg dla danego typu dokumentów:
  - dokument 1



  - dokument 2

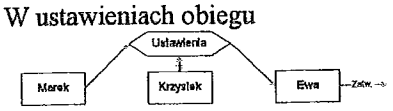
  - dokument 3



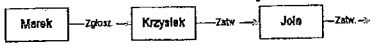
© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - określanie osób

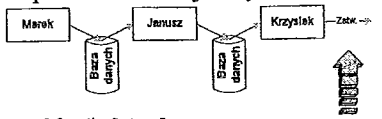
- W ustawieniach obiegu



- Po dostarczeniu do osoby



- Na podstawie innej bazy

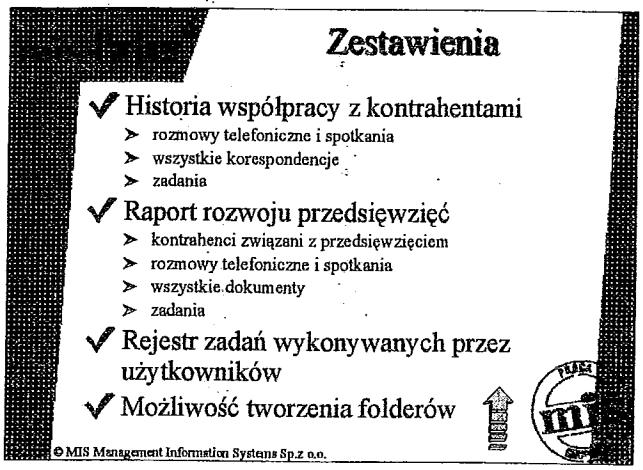


© MIS Management Information Systems Sp.z o.o.

Obieg dokumentów - ustawienia

- Określenie czasu ważności
- Automatyczna akcja po terminie:
  - automatyczna akceptacja,
  - automatyczne odrzucenie,
  - ponaglenie
- Terminologia

© MIS Management Information Systems Sp.z o.o.

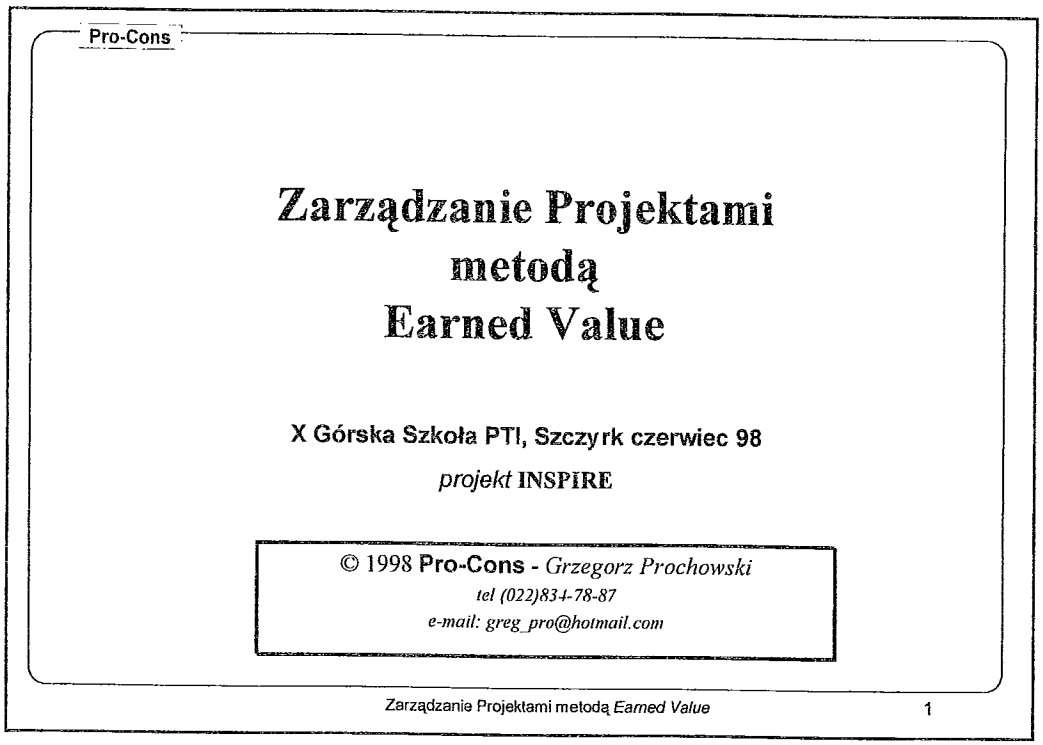

Zestawienia
Historia współpracy z kontrahentami
rozmowy telefoniczne i spotkania
wszystkie korespondencje
zadania
Raport rozwoju przedsięwzięć
kontrahenci związani z przedsięwzięciem
rozmowy telefoniczne i spotkania
wszystkie dokumenty
zadania
Rejestr zadań wykonywanych przez użytkowników
Możliwość tworzenia folderów
© MIS Management Information Systems Sp.z o.o.

Grzegorz Prochowski

Pro-Cons

tel. (022) 834-78-87
e-mail: greg_pro@hotmail.com

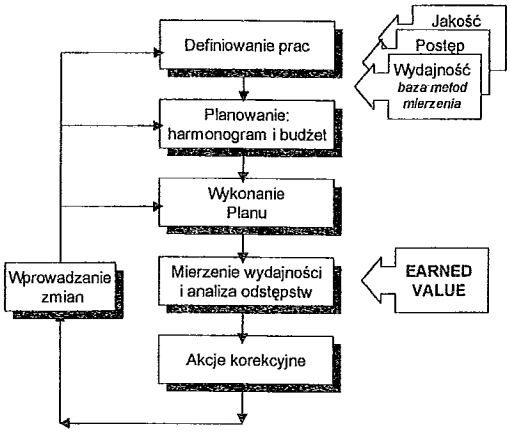
Zarządzanie Projektami metodą Earned Value



Pro-Cons

Metoda Earned Value

- Metoda EV jest obiektywną metodą mierzenia wydajności i postępu prac w projekcie. EV jest znacznym usprawnieniem w stosunku do tradycyjnych metod śledzenia kosztów, w których porównuje się budżet i koszty bieżące.
- Earned Value ("wartość zarobiona") jest pojedynczym elementem skupiającym koszty, harmonogram i techniczną wydajność stanowiąc podstawę dla komunikacji między klientem produktów projektu i zespołem wykonującym.
- Metoda EV wspomaga zarządzanie projektem:
  - wszystkie elementy zakresu projektu są planowane i ich realizacja śledzona
  - cele określone w zakresie,harmonogramie i kosztach są zintegrowane w planie bazowym i podlegają obiektywnemu mierzeniu
  - osiągane wyniki są obiektywnie oceniane wraz z określeniem sprawności wykonawczej
  - znaczące odchylenia od planu są analizowane. a ich wpływ na przyszły przebieg projektu jest przewidywalny co pozwala przedsięwziąć akcje korekcyjne dostatecznie wcześnie.



Zarządzanie Projektami metodą *Earned Value* 2

Pro-Cons

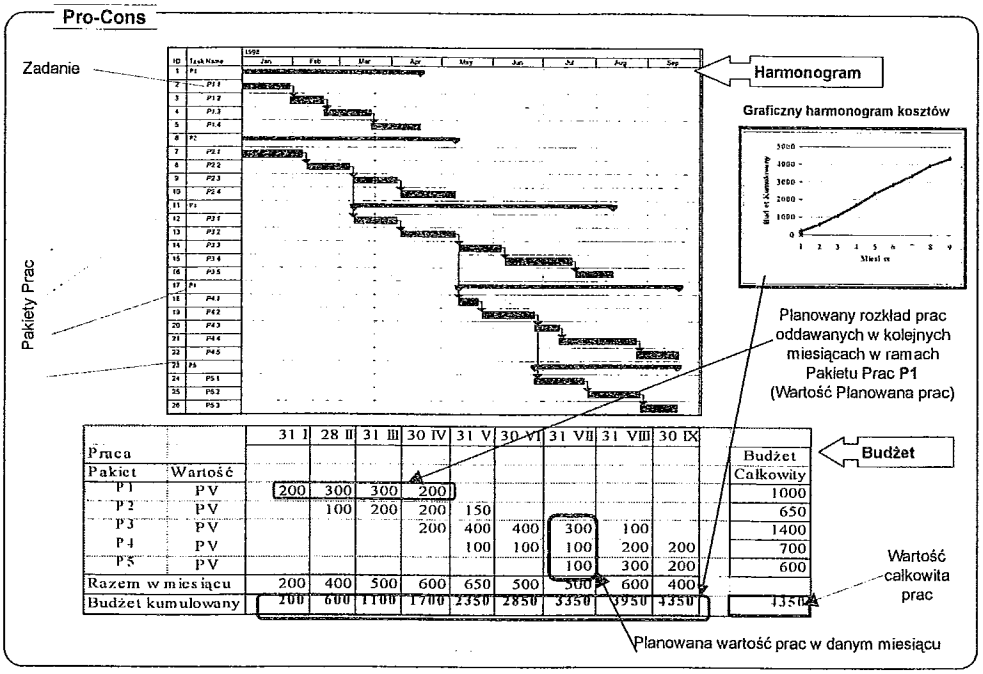
Metoda Earned Value

Metoda Earned Value opiera się na trzech mierzalnych atrybutach planowania związanych z harmonogramem prac:

- Wartości Prac Planowanych (**PV** - *Planned Value*)
- Kosztach Rzeczywistych (**AC** - *Actual Costs*)
- Wartości Prac Wykonanych lub "zarobionych" (**EV** - *Earned Value*)

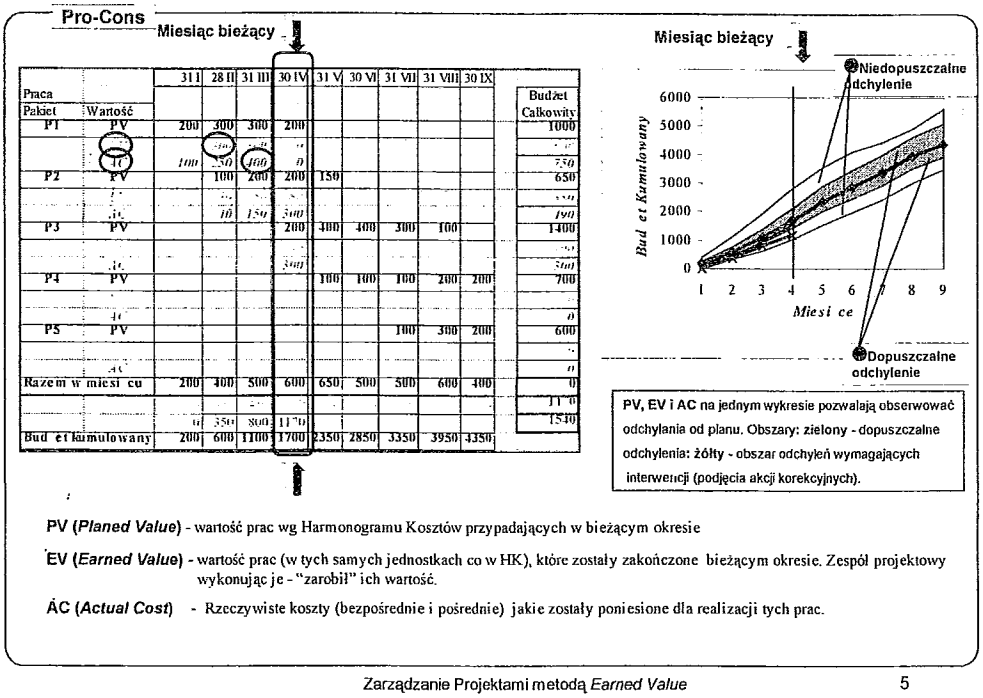
Rozważmy przykład ... ⇨

Zarządzanie Projektami metodą *Earned Value* 3



Praca		31 I	28 II	31 III	30 IV	31 V	30 VI	31 VII	31 VIII	30 IX	Budżet
Pakiet	Wartość										Całkowity
P1	PV	200	300	300	200						1000
P2	PV		100	200	200	150					650
P3	PV				200	400	400	300	100		1400
P4	PV					100	100	100	200	200	700
P5	PV							100	300	200	600
Razem w miesiącu		200	400	500	600	650	500	500	600	400	
Budżet kumulowany		200	600	1100	1700	2350	2850	3350	3950	4350	4350

Zarządzanie Projektami metodą *Earned Value* 4



Praca		31 I	28 II	31 III	30 IV	31 V	30 VI	31 VII	31 VIII	30 IX	Budżet
Pakiet	Wartość										Całkowity
P1	PV	200	300	300	200						1000
P2	PV		100	200	200	150					650
P3	PV				200	400	400	300	100		1400
P4	PV					100	100	100	200	200	700
P5	PV							100	300	200	600
Razem w miesiącu		200	400	500	600	650	500	500	600	400	0
Budżet kumulowany		200	600	1100	1700	2350	2850	3350	3950	4350	4350

PV (*Planed Value*) - wartość prac wg Harmonogramu Kosztów przypadających w bieżącym okresie

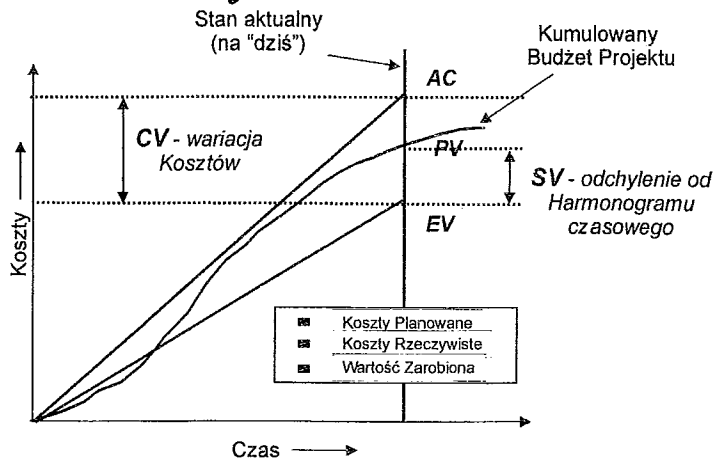
EV (*Earned Value*) - wartość prac (w tych samych jednostkach co w HK), które zostały zakończone bieżącym okresie. Zespół projektowy wykonując je - "zarobił" ich wartość.

AC (*Actual Cost*) - Rzeczywiste koszty (bezpośrednie i pośrednie) jakie zostały poniesione dla realizacji tych prac.

Zarządzanie Projektami metodą *Earned Value* 5

Pro-Cons

Miary w metodzie EV



Wariacje

SV=EV - PV Odchylenie od Harmonogramu (czasowego) do daty bieżącej. Wartość ujemna oznacza opóźnienie w stosunku do planu.

CV=EV - AC Wariacja kosztów do daty bieżącej. Wartość ujemna oznacza, że wykonanie prac konsumuje więcej zasobów niż przewiduje plan.

Zarządzanie Projektami metodą *Earned Value* 6

Pro-Cons

Kontynuacja przykładu 1

Odchylenie od harmonogramu:

	P1	P2	P3	razem	
PV	200	200	200	600	
EV	-	180	200	380	
SV=EV-PV	-200	-20	0	-220	-37%

Wariacja kosztów:

	P1	P2	P3	razem	
EV	-	180	200	380	
AC	-	300	250	600	
CV=EV-AC	-	-120	-50	-170	-45%

Porównanie wydatków?

	P1	P2	P3	razem	
PV	200	200	200	600	
AC	-	300	250	550	
CV=PV-AC	-	-120	-50	+50	+8%

Porównanie wydatków planowanych i aktualnych nie jest w jawnym związku z pracą wykonaną. Dowiadując się, że wydatki w okresie sprawozdawczym było mniejsze o 8% od zaplanowanego, nie wiemy nic o pracach wykonanych

Zarządzanie Projektami metodą *Earned Value* 7

Pro-Cons

Wskaźniki sprawności (1)

- Wskaźniki sprawności pokazują jak sprawnie przebiega realizacja Projektu. Wartości większe niż 1 (100%) oznaczają sprawność większą niż wymagana. Wartości mniejsze niż 1 (100%) oznaczają sprawność małą i zagrożenie niewykonania prac zgodnie z planem.

Przykład:

BAC - *Budget At Completion*
Budżet Końcowy Projektu

Kumulowane dane Projektu X w dniu bieżącym:

PVc - Wart. Planowana	EVc - Wart. Zarobiona	ACc - Koszty Rzecz.	BAC
45 000 zł	48 200 zł	50 100 zł	60 000 zł

- Wskaźnik Sprawności Wykonania Harmonogramu (czasowego) (**SPI** - Schedule Performance Index) - jest to stosunek tego co zostało zrobione do tego co powinno być zrobione:

$$SPI = \frac{EVc}{PVc}$$

EVc - Kumulowana Wartość Zarobiona
PVc - Kumulowana Wartość Planowana (harmonogram kosztów prac do dnia bieżącego).

Przykład	SPI	SPI%
48 200 zł / 45 000 zł	1.07	107 %

W przykładzie realizacja Projektu wyprzedza plan.

Jednakże wskaźnik SPI niekoniecznie daje prawdziwy obraz sytuacji. Niektóre prace mogły być wykonane w innej sekwencji. Potrzebne są inne wskaźniki...

Zarządzanie Projektami metodą *Earned Value* 8

Pro-Cons

Wskaźniki sprawności (2)

- Wskaźnik Sprawności Kosztów (**CPI** - *Cost Performance Index*) pokazuje on wartość prac wykonanych na każdą wydaną złotówkę:

$$CPI = \frac{EVc}{ACc}$$

EVc - Kumulowana Wartość Zarobiona
ACc - Kumulowana Wartość Kosztów poniesionych do dnia bieżącego

Przykład	CPI	CPI%
48 200 zł / 50 100 zł	0.96	96%

Za każdą wydaną złotówkę zrobiona została praca wartości 96 groszy.

W Projekcie X - wydajność nie jest najlepsza albo wykorzystywane są zasoby droższe niż przewidywał plan.

- Wymagany Wskaźnik Sprawności (**TCPI** - *To Complete Perfor-mance Index*) jest stosunkiem pracy pozostałej do wykonania do pozostałej części budżetu. Pokazuje jaki CPI powinniśmy utrzymać by zakończyć Projekt w ramach Budżetu.

$$TCPI = \frac{BAC - EVc}{BAC - ACc}$$

EVc - Kumulowana Wartość Zarobiona
ACc - Kumulowana Wartość Kosztów poniesionych do dnia bieżącego
BAC - Budżet Końcowy Projektu

Przykład	TCPI	TCPI%
11 800 zł / 9 900 zł	1.19	119%

Aby utrzymać się w budżecie należy podnieść wydajność do poziomu TCPI=1.19.

Ponieważ jednak różnica TCPI - CPI > 0.2 (20%) jest słaba nadzieja by mógł nastąpić taki wielki wzrost wydajności. Raczej trzeba wynegocjować nowy Budżet Końcowy...

Zarządzanie Projektami metodą *Earned Value* 9

Pro-Cons

Koszty Końcowe

Estymacja Kosztów Końcowych (*EAC -Estimate at Completion*) -- przewidywane koszty całego Projektu dokonywane na podstawie aktualnej wydajności. Najczęściej stosowane przybliżenia:

- *I.* **EAC = (koszty aktualne) + (pozostały budżet zmodyfikowany przez współczynnik wydajności)**

$$EAC_1 = ACc + \frac{BAC - EVc}{CPI}$$

lub jeśli przypuszczamy, że wariacje przyszłe będą podobne do obecnych (proces jest stabilny statystycznie), to można stosować aproksymacje:

$$EAC_2 = ACc + \frac{BAC - EVc}{0.8 \times CPI + 0.2 \times SPI}$$

$$EAC_3 = ACc + \frac{BAC - EVc}{CPI \times SPI}$$

- *II.* **EAC = (koszty aktualne) + (nowe estymacje dla wszystkich pozostałych prac)**
  Taką aproksymację stosuje się, jeśli jesteśmy pewni (na podstawie analizy wydajności), że dotychczasowe estymacje były całkowicie błędne lub nie można ich dłużej utrzymać na skutek poważnych zmian w Projekcie.

- *III.* **EAC = (koszty aktualne) + (pozostały budżet)**

$$EAC = ACc + (BAC - EVc)$$

Taką aproksymację stosuje się, kiedy uznajemy, że obecne wariacje są wyjątkowe i nie wystąpią w przyszłości.

Zarządzanie Projektami metodą *Earned Value* 10

Pro-Cons

... dalszy ciąg przykładu

PVc	EVc	ACc	BAC
45 000 zł	48 200 zł	50 100 zł	60 000 zł

SPI	CPI	TCPI
1.07	0.96	1.19

- Nowe estymacje Budżetu Końcowego (wg wzorów na poprzednich slajdach) oraz nowe wartości Wymaganego Wskaźnika Wydajności.

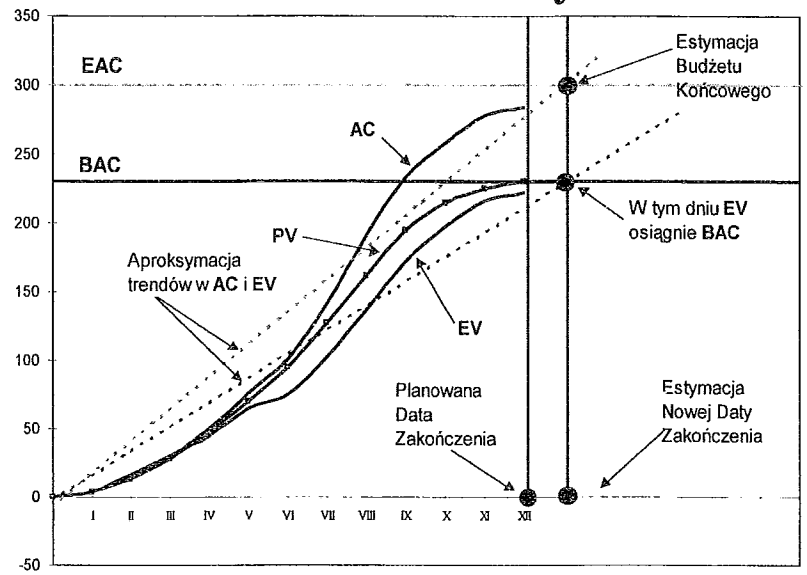
	EAC	nowy TCPI
EAC_1	62 392 zł	0.96
EAC_2	62 116 zł	0.98
EAC_3	61 587 zł	1.03

$$TCPI_{EAC} = \frac{BAC - EVc}{EAC - ACc}$$

Zarządzanie Projektami metodą *Earned Value* 11

Pro-Cons

Podsumowanie metody *Earned Value*



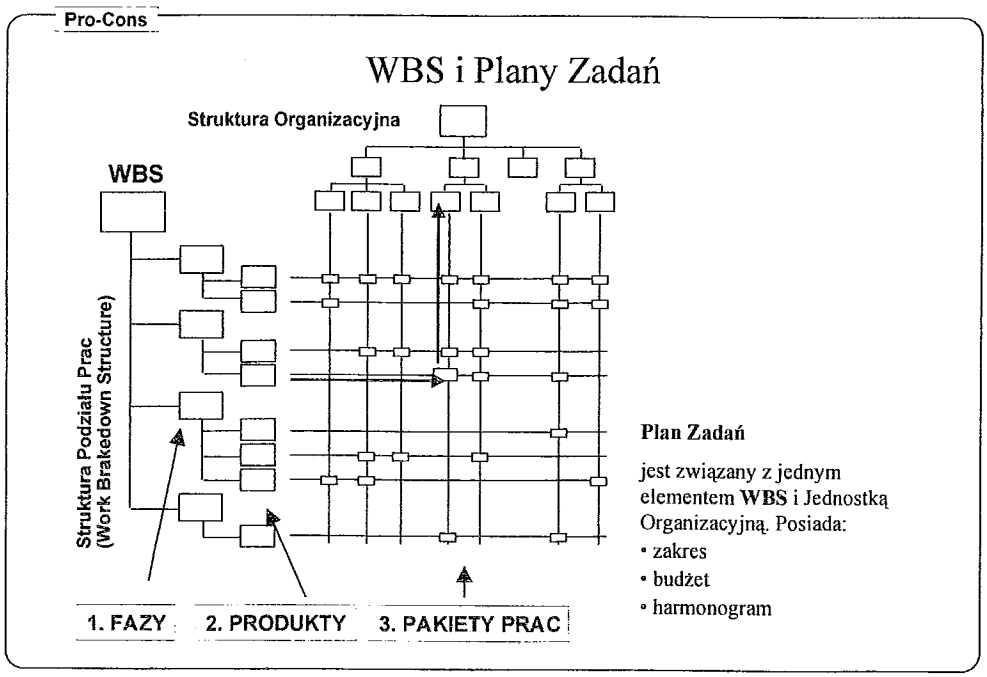
Zarządzanie Projektami metodą *Earned Value* 12

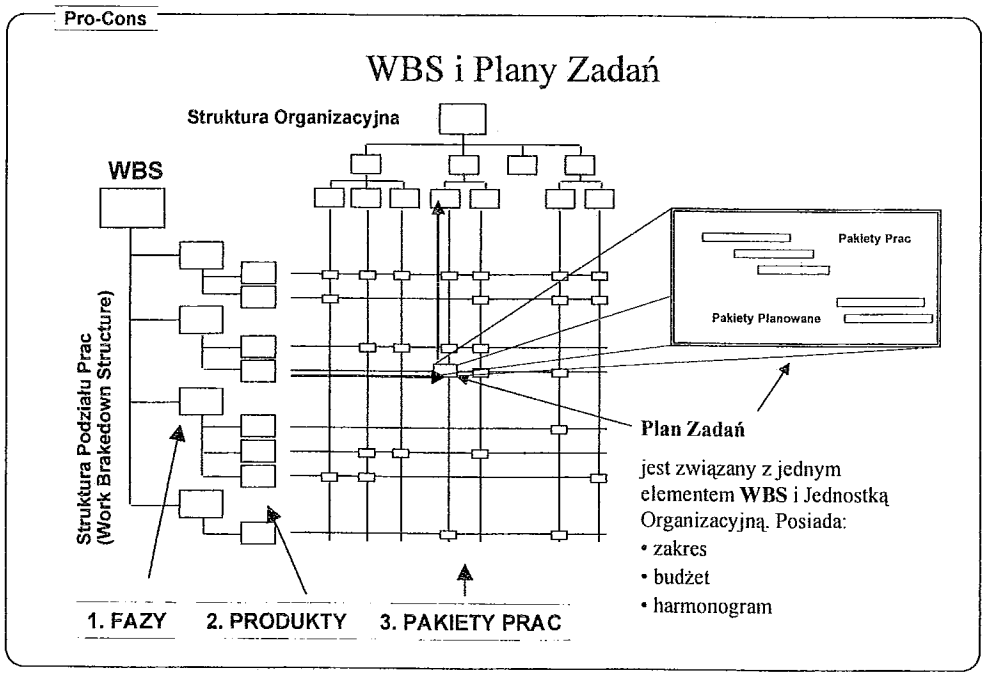
Pro-Cons

Co jest potrzebne by można było wdrożyć Zarządzanie metodą *Earned Value*?

- Zdefiniuj wszystkie elementy pracy, które muszą być wykonane
- Zbuduj Strukturę Podziału prac (WBS - Work Breakdown Structure) - hierarchiczną strukturę, w której najwyższy poziom określa projekt, niższy główne fazy, następny produkty faz, następny pakiety prac itd....
- Oszacuj pracochłonność każdej czynności.
- Zdefiniuj strukturę organizacyjną uczestników projektu.
- Zbuduj harmonogram.
- Oszacuj koszty.
- Dokonaj integracji WBS i struktury organizacyjnej (macierz przypisania obowiązków)
- Ustal system kamieni milowych dla mierzenia postępu prac.
- Wprowadź praktykę zarządzania metodą Earned Value...

Zarządzanie Projektami metodą *Earned Value* 13

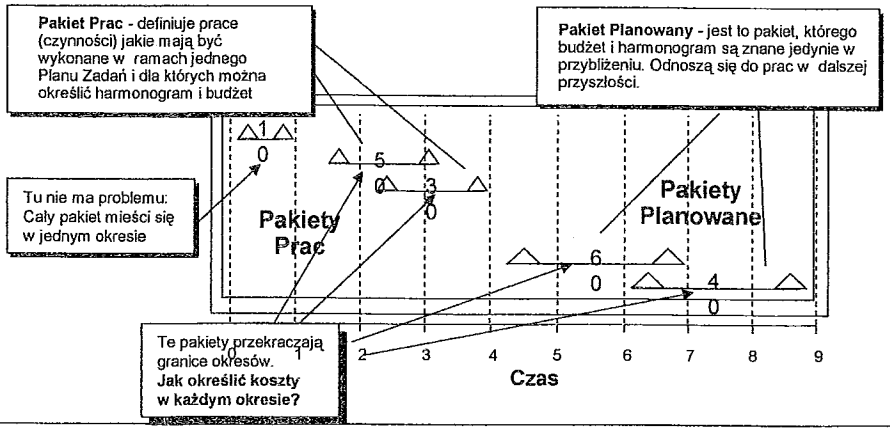

Pro-Cons
WBS i Plany Zadań
Struktura Organizacyjna
WBS
Struktura Podziału Prac (Work Brakedown Structure)
1. FAZY
2. PRODUKTY
3. PAKIETY PRAC
Plan Zadań
jest związany z jednym elementem WBS i Jednostką Organizacyjną. Posiada:
• zakres
• budżet
• harmonogram
Zarządzanie Projektami metodą Earned Value
14


Pro-Cons
WBS i Plany Zadań
Struktura Organizacyjna
WBS
Struktura Podziału Prac (Work Brakedown Structure)
Pakiety Prac
Pakiety Planowane
1. FAZY
2. PRODUKTY
3. PAKIETY PRAC
Plan Zadań
jest związany z jednym elementem WBS i Jednostką Organizacyjną. Posiada:
• zakres
• budżet
• harmonogram
Zarządzanie Projektami metodą Earned Value
15

Pro-Cons

Jak tworzy się Harmonogram Kosztów (HK)?

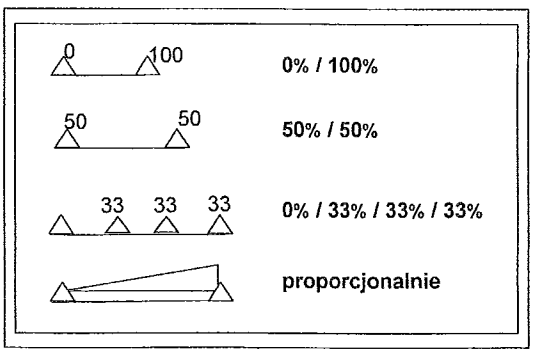
- Plan Zadań zbudowany jest z Pakietów Prac
- Przyrosty kosztów w HK muszą odpowiadać ustalonym odstępom czasu (okresom sprawozdawczym)



Pro-Cons

Przypisywanie kosztów do Kamieni Milowych (KM)

- Oto niektóre techniki przypisywania kosztów do Kamieni Milowych. Jeśli w danym okresie sprawozdawczym powinien być osiągnięty (wewnętrzny) KM - do HK dodaje się odpowiedni przyczynek (procent) kosztów.



- Inne techniki to:
  - standardy "inżynierskie" - koszty KM zależą od metody
  - ocena podwykonawcy
  - ocena kierownictwa Projektu.

Pro-Cons

Warunki powodzenia metody *Earned Value*

- Koszty i harmonogram powinny być oparte na danych empirycznych.
- Planowanie oparte na metrykach wymaga dostatecznie wczesnego określenia rozmiarów przedsięwzięcia, projekcji kosztów i harmonogramu z danych wzorcowych - czyli danych z ukończonych z sukcesem projektów o podobnym charakterze i podobnych celach.
- W trakcie trwania projektu należy systematycznie rejestrować dane pomiarowe.
- Celem jest wczesne uchwycenie i identyfikacja problemów.
- Istotną sprawą jest rozstrzygnięcie czy przyczyna opóźnień leży w zespole czy w złym, zbyt agresywnym planie....

Zarządzanie Projektami metodą *Earned Value* 18

Pro-Cons

Jak należy postępować?

- Opracowanie dobrej WBS (Struktury Podziału Prac) jest sprawą kluczową.
- Należy planować krótko-terminowe zadania z mierzalnymi rezultatami (produktami), których ukończenie można określać binarnie (tak/nie).
- Często przeglądać metryki:
  - ✓ Earned Value <--> Rzeczywiste wydatki
  - ✓ EAC Oszacowane koszty pozostałe (włączając oszacowanie nierozstrzygniętych ryzyk) <--> Koszty planowane - BAC
  - ✓ CPI i TCPI

Zarządzanie Projektami metodą *Earned Value* 19

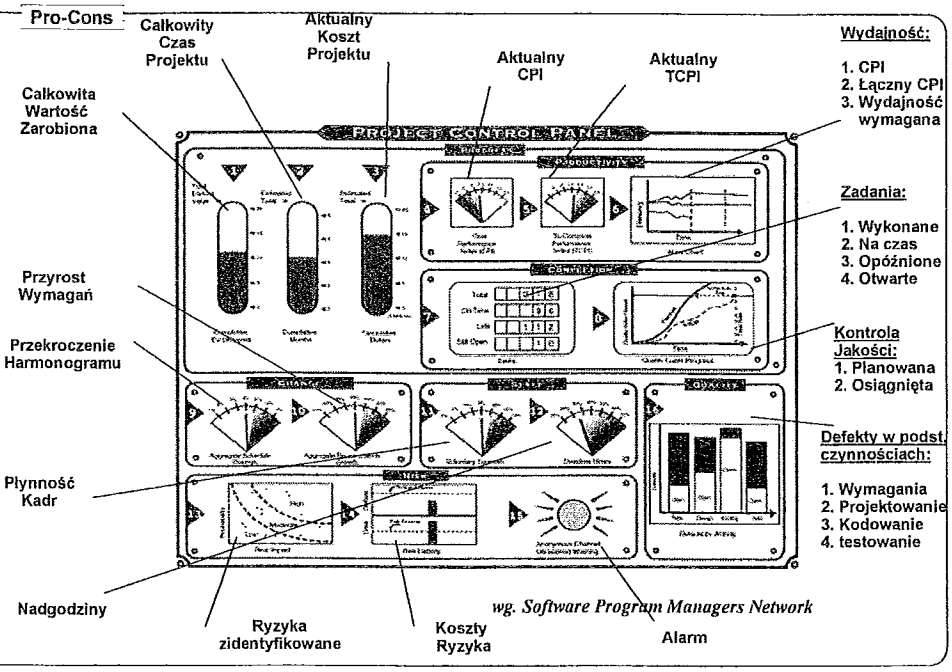
Pro-Cons

O czym nie mówiliśmy?

- Jak tworzyć harmonogramy warunkowe - t.zn. - w których pewne prace są podejmowane i ich zakres jest określony w zależności od wykonania innych prac ujętych w harmonogramie głównym?
- Harmonogramy typu Master-Slave?

Zarządzanie Projektami metodą *Earned Value* 20

Pro-Cons



Zarządzanie Projektami metodą *Earned Value* 21

Pro-Cons

Słownik terminów, synonimy

- **BCWS** -Budgeted Cost for Work Scheduled - to samo co Wartość Planowana (PV -Planned Value)
- **BCWP** - Budgeted Cost for Work Performed - to samo co Earned Value (EV)
- **ACWP** - Actual Cost of Work Performed - to samo co Koszty Poniesione (AC - Actual Costs)
- **BAC** - Budget At Completion - Budżet końcowy całkowity (planowany)
- **EAC** - Estimate At Completion - Budżet końcowy estymowany
- **CPI** - Cost Performance Index - Sprawność (wg kosztów)
- **SPI** - Schedule Performance Index - Sprawność (realizacji harmonogramu)
- **WBS** - Work Breakdown Structure - Struktura podziału prac.

Zarządzanie Projektami metodą *Earned Value* 22

SPICE

standard oceny i doskonalenia procesów produkcji oprogramowania

Tomasz Byzia

Wprowadzenie

Rozwój przemysłowy Europy będzie oparty na trzech fundamentalnych strukturach sieciowych:

- transport
- energetyka
- informacja

Budowanie skutecznych, efektywnych i konkurencyjnych rozwiązań na ich bazie wymaga wysokiej jakości systemów informacyjnych

Raport Bangemann'a, 1994

Growth, competitiveness, employment - The challenges and the ways forward into the 21st century White Paper, European Commission, Brussels 1994

Wyzwania

Cechy współczesnego rynku oprogramowania

- Wzrost skali i zakresu systemów
- Większe i bardziej złożone projekty
- Złożone architektury systemów
- Funkcjonalność oparta głównie na oprogramowaniu
- Krótszy czas produkcji i dostarczenia
- Większe wymagania zmian, modyfikacji, poprawek
- Wymaganie wysokiej niezawodności
- Większa konkurencja na rynku

Wyzwanie = Problemy

Jakość oprogramowania

Współczesne oprogramowanie i proces jego produkcji poddawane są krytyce przez użytkowników oraz jego producentów. Uwagi dotyczą:

- problemów z rozpoznaniem rzeczywistych wymagań
- niedotrzymywania terminów dostaw
- przekraczania budżetu projektów
- nieefektywnych metod zarządzania projektami
- niskiej niezwodności oprogramowania
- braku metodyk i narzędzi

Problemy = Trzeba je rozwiązać

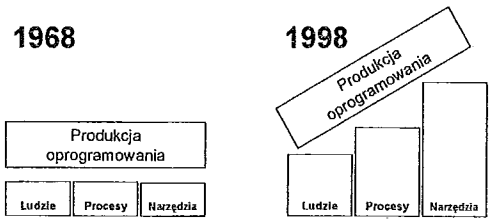
Inżynieria oprogramowania

Konferencja "Software engineering", NATO 1968

- Wyraz frustracji użytkowników i producentów
- Analiza stanu rzeczy i impuls do poprawy
- Rzeczywista inspiracja "dekady wynalazków"
  - zauważenie cyklu życia oprogramowania
  - zasady pracy zespołu programistów
  - paradygmat programowania strukturalnego
  - język Pascal
  - dojrzewanie koncepcji modularyzacji oprogramowania
  - abstrakcyjne typy danych
  - rozwój metrologii oprogramowania (modele estymacyjne)
  - formalna weryfikacja programów
  - nowoczesne narzędzia

Rzeczywista od-nowa?

Przeświadczenie o poprawie procesów produkcji oprogramowania przez doskonalenie metod i narzędzi okazało się mitem



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 6

Nowy kierunek doskonalenia

Badania pokazują, że 80% niepowodzeń ma źródło w przyczynach nietechnicznych

Niewielkie nakłady na zmiany w sferze organizacyjnej i zarządzania firm mogą przynieść większe korzyści niż duże nakłady na doskonalenie narzędzi

Korzyści z powyższego podejścia to:

- wzrost produktywności
- kontrola czasu i budżetu projektów
- krótrzy czas dostarczenia
- wczesna identyfikacja problemów i defektów
- stopa zwrotu rzędu 4 do 9

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 7

Nowy kierunek - stare pytania

Co odpowiadamy na następujące pytania klientów:

- czy wasze rozwiązania pasują do moich potrzeb?
- czy dysponujecie nowoczesną technologią?
- czy możecie zagwarantować jakość?
- czy jesteście zdolni zrealizować ten system dla nas?
- czy jesteście wiarygodni?

5 x TAK

Co zrobić, gdy klient zapyta:

Jak przekonacie nas, że mówicie prawdę?

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 8

Czego potrzebujemy?

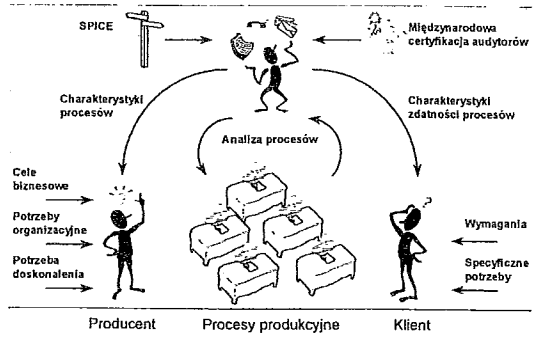
Potrzebujemy systematycznej metody badania procesów produkcyjnych i kryteriów ich oceny pozwalających na zebranie wyników, które są:

- obiektywne
- powtarzalne
- porównywalne
- użyteczne do doskonalenia procesu
- użyteczne do oceny zdatności procesu

SPICE (Software Process Improvement and Capability dEtermination)
Celem projektu jest dostarczenie światowego standardu spełniającego powyższe postulaty

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 9

Kontekst standardu



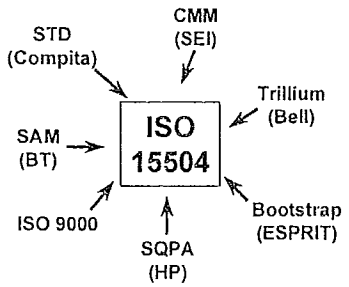
SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 10

Trochę historii

1987 - raport SEI: metoda przeprowadzania oceny firm
1989 - pierwszy kwestionariusz oceny SEI
1991 - model CMM 1.0
1991 - powstaje TRILLIUM: model Bell Northern Research/Northern Telecom
1991 - powstaje Bootstrap - Europejski model oceny ESPRIT
1991 - studium UK MOD (analiza 25 istniejących schematów oceny)
1993 - pierwsze spotkanie grupy projektu SPICE - ISO/IEC JTC1/SC7/WG10
1993 - model CMM 1.1
1994 - zakończenie fazy opracowywania standardu SPICE
1995 - zakończenie 1 fazy prób
1996 - rozpoczecie 2 fazy prób

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 11

Źródła standardu



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 12

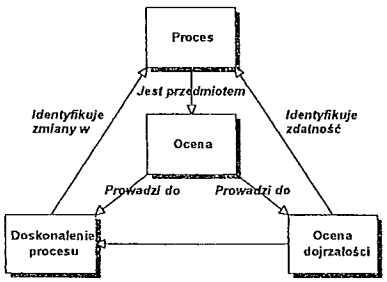
Cele standardu

Dostarczenie powtarzalnej metody oceny procesów produkcyjnych, która pozwala na:

- Identyfikację silnych i słabych stron producenta oprogramowania
- Lepszą definicję procesów produkcyjnych
- Przygotowanie skutecznego planu doskonalenia procesów produkcyjnych
- Wybór najlepszego kooperanta na podstawie oceny poziomu zdatności jego procesów

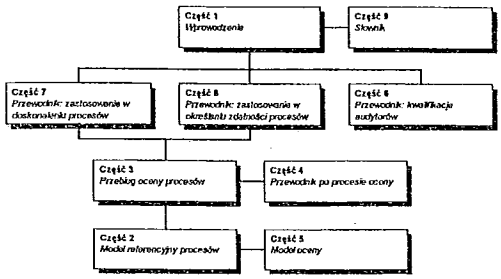
SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 13

Schemat metody



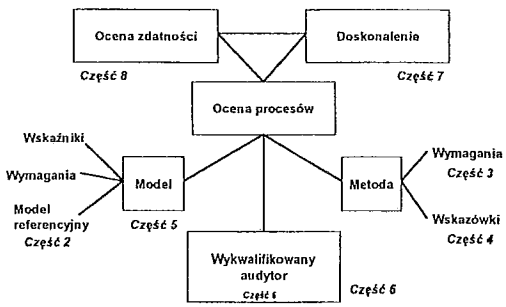
SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 14

Struktura standardu



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 15

Zależność między elementami



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 16

Model

Wybrane procesy ocenianej organizacji są porównywane z modelem odniesienia SPICE

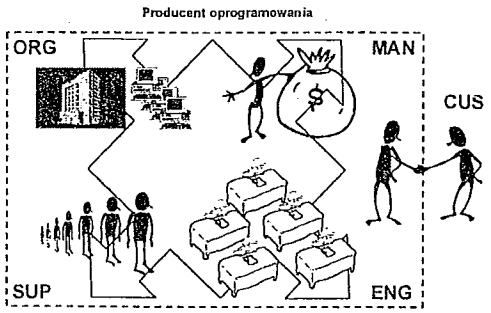
Model ma dwa wymiary:

- wymiar procesu
  - określone są cele, których osiągnięciu służy proces
  - procesy są zgrupowane w kategorie
- wymiar zdatności
  - określone są atrybuty, które charakteryzują proces
  - atrybuty są zgrupowane w poziomy dojrzałości
  - te same atrybuty stosują się do każdego procesu

Atrybuty procesów podlegają ocenie stopnia realizacji w skali 4 stopniowej: wszystko, większość, część, nic

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 17

Kategorie procesów



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 18

Kategoria: Klient-Producent

CUS (*Customer-Supplier*)

CUS.1 Pozyskiwanie oprogramowania
CUS.2 Zarządzanie potrzebami klienta
CUS.3 Dostarczanie oprogramowania
CUS.4 Obsługa oprogramowania
CUS.5 Obsługa klienta

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 19

Kategoria: Inżynieria

ENG (*Engineering*)

ENG.1 Opracowanie wymagań i projektu systemu
ENG.2 Opracowanie wymagań oprogramowania
ENG.3 Opracowanie projektu oprogramowania
ENG.4 Implementacja projektu oprogramowania
ENG.5 Integracja i testy oprogramowania
ENG.6 Integracja i testy systemu
ENG.7 Pielęgnacja systemu i oprogramowania

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 20

Kategoria: Wsparcie

SUP (*Support*)

SUP.1 Opracowanie dokumentacji
SUP.2 Zarządzanie konfiguracją
SUP.3 Zapewnienie jakości
SUP.4 Weryfikacja produktów
SUP.5 Walidacja produktów
SUP.6 Wspólne przeglądy
SUP.7 Audyty
SUP.8 Rozwiązywanie problemów

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 21

Kategoria: Zarządzanie

MAN (*Management*)

MAN.1 Zarządzanie projektem
MAN.2 Zarządzanie jakością
MAN.3 Zarządzanie ryzykiem
MAN.4 Zarządzanie podwykonawcami

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 22

Kategoria: Organizacja

ORG (*Organization*)

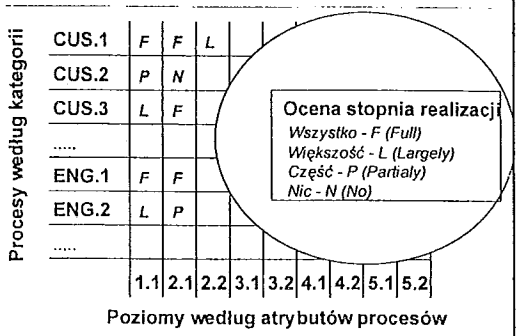
ORG.1 Budowanie kultury organizacyjnej
ORG.2 Definicja procesu
ORG.3 Doskonalenie procesu
ORG.4 Zarządzanie zasobami ludzkimi
ORG.5 Dostarczanie infrastruktury

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 23

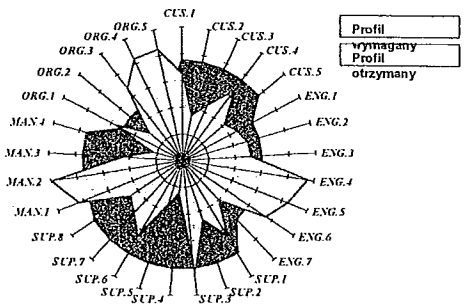
Poziomy zdatności

Poziom 5 Proces optymalizowany	
5.1 Dostosowywanie procesu	*Proces dostosowywany do bieżących i przyszłych potrzeb*
5.2 Doskonalenie procesu	
Poziom 4 Proces przewidywalny	
4.1 Metryki procesu	*Proces stabilny i przewidywalny w osiąganiu celów*
4.2 Sterowanie procesem	
Poziom 3 Proces zdefiniowany	
3.1 Definicja procesu	*Proces realizowany według sprawdzonych procedur i praktyk*
3.2 Zarządzanie zasobami	
Poziom 2 Proces kierowany	
2.1 Zarządzanie realizacją	*Proces dostarcza produktów o założonej jakości przy określonych zasobach*
2.2 Zarządzanie produktem	
Poziom 1 Proces realizowany	
1.1 Realizacja	*Proces osiąga swój cel*

Analiza procesów



Prezentacja wyników analizy



Certyfikacja audytorów

SUGaR - oficjalna grupa użytkowników standardu opracowała program autoryzacji audytorów SPICE:

- Szkolenie audytorów SPICE
- Egzamin
- Aplikacja: *Certified Provisional SPICE Assessor*
- Zdobywanie doświadczenia w analizie wg SPICE
- Aplikacja: *Certified SPICE Assessor*
- Pogłębianie doświadczeń w analizie wg SPICE
- Odnowienie aplikacji: *Certified SPICE Assessor*

Standard SPICE precyzyjnie definiuje wymagania i kryteria oceny kandydatów na audytorów

Próby aplikacyjne standardu

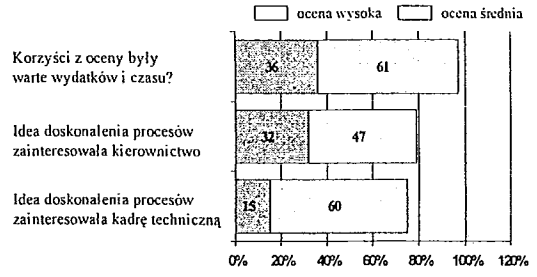
SPICE jest tworzony przyspieszoną procedurą standaryzacyjną i opiera się na wczesnej praktycznej weryfikacji modelu i metod oceny

Ustalono 3 fazy prób (badań aplikacyjnych):

- Faza 1 (-1995)
  weryfikacja użyteczności i jakości dokumentów standardu (30 firm)
- Faza 2 (1996-1998)
  weryfikacja jakości dokumentów oraz powtarzalności i porównywalności wyników oceny (200+ firm)
- Faza 3 (1998-)
  zebranie danych potwierdzających i oceniających wartość biznesową standardu wraz z jego szerokim upowszechnieniem

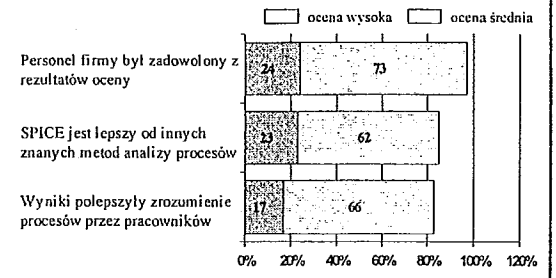
Wyniki 1 fazy prób

Ogólna ocena SPICE przez badane firmy



Wyniki 1 fazy prób

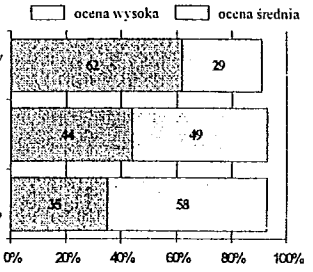
Ogólna ocena SPICE przez audytorów



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 30

Wyniki 1 fazy prób

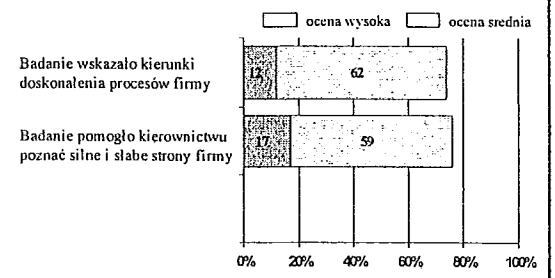
Ocena trafności wyników wg badanych firm



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 31

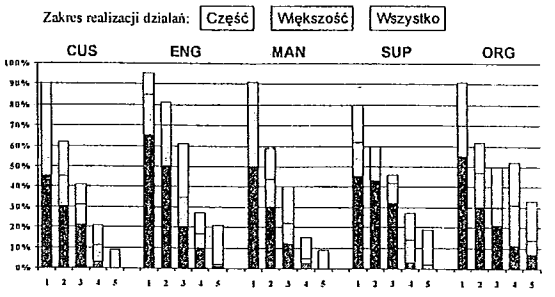
Wyniki 1 fazy prób

Ocena trafności wyników wg audytorów



SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 32

Wyniki w kategoriach



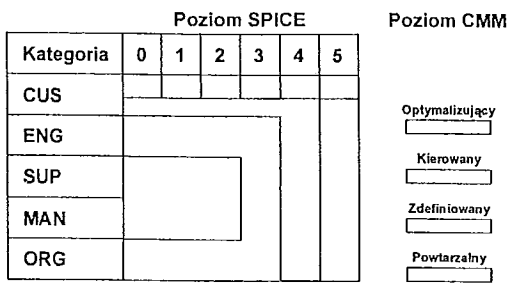
SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 33

Korzyści stosowania SPICE

- Pomoc w ocenie zdatności procesów produkcyjnych dostawcy - wiarygodna ocena ryzyka współpracy
- Korzystanie ze standardowego, powtarzalnego i porównywalnego schematu oceny
- Narzędzie zainicjowania i ciągłego doskonalenia procesów produkcyjnych
- Przygotowanie organizacji do wdrożenia innych standardów (np. ISO 9000)
- Lepsze powiązanie działań doskonalących z rzeczywistymi celami organizacji
- Lepsze dostosowanie procesów do rzeczywistych celów organizacji

SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 34

SPICE vs CMM



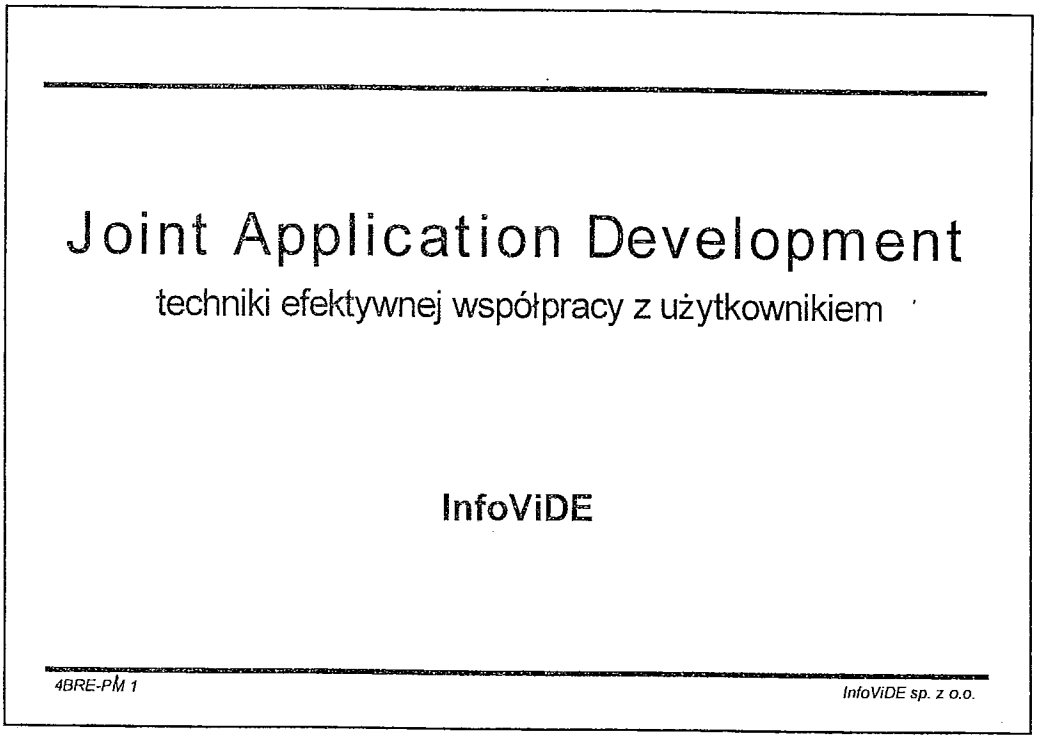
SPICE standard oceny i doskonalenia procesów © Tomasz Byzia Strona 35

Borys Stokalski

InfoViDE

Joint Application Development

techniki efektywnej współpracy z użytkownikiem



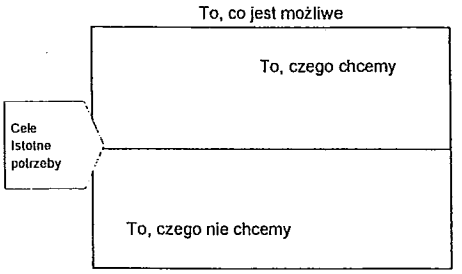
Joint Application Development

techniki efektywnej współpracy z użytkownikiem

InfoViDE

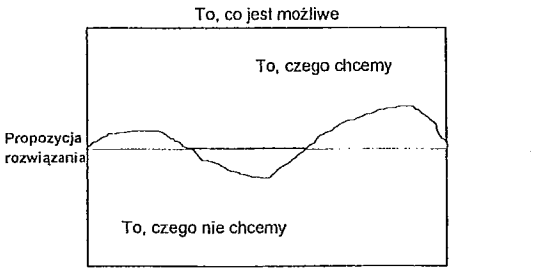
4BRE-PM 1 InfoViDE sp. z o.o.

Idealne rozwiązanie



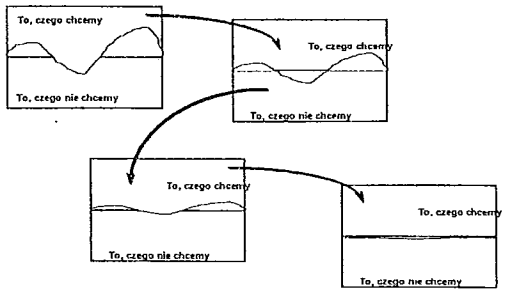
4BRE-PM 2 InfoViDE sp. z o.o.

Definiowanie zakresu



4BRE-PM 3 InfoViDE sp. z o.o.

Analiza wymagań - precyzowanie zakresu



4BRE-PM 4 InfoViDE sp. z o.o.

Definiowania i analiza potrzeb

Jakie są konkretne (mierzalne) cele przedsięwzięcia?

Co charakteryzuje rozwiązanie właściwie realizujące zakładane cele?

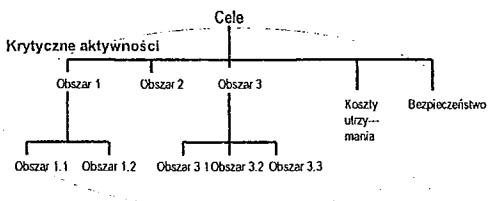
Jak oddzielić rozwiązania „lepsze" od „gorszych"?

Jak określić optymalny zakres przedsięwzięcia

- priorytety
- ocena kosztów/korzyści

4BRE-PM 5 InfoViDE sp. z o.o.

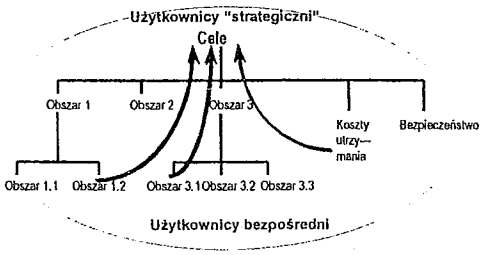
Model wymagań



Wymagania - miary i wymagane poziomy jakości rozwiązań
Problemy - sytuacje w których aktualny poziom jakości jest gorszy od wymaganego

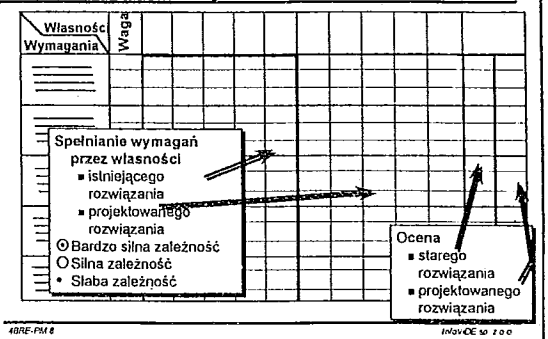
4BRE-PM 6 InfoViDE sp. z o.o.

Wymagania, użytkownicy i beneficjenci



4BRE-PM 7 InfoViDE sp. z o.o.

Techniki jakościowe - QFD



4BRE-PM 8 InfoViDE sp. z o.o.

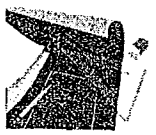
Modelowanie

Specyfikacja wymagań zawierać może dodatkowe modele uszczegóławiające wymagania:

- model procesów Specyfikacja
- model danych
- model obiektów
- wyniki prototypowania.

Zalety modelowania

- standardowa notacja
- dobrze określone kryteria jakości
- „obraz wart tysiąca słów"



4BRE-PM 9 InfoViDE sp. z o.o.

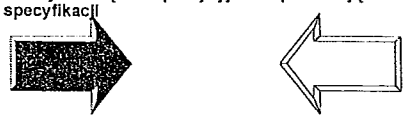
Producent/konsument

Użytkownik

- precyzyjnie określa swoje potrzeby
- weryfikuje zgodność produktu z założeniami

Wykonawca

- tworzy rozwiązanie precyzyjnie odpowiadające specyfikacji



4BRE-PM 10 InfoViDE sp. z o.o.

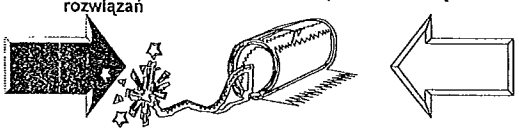
Punkt widzenia...

Dostawca

- utrzymanie rentowności
- płynność finansowa
- stabilizacja zakresu
- pragmatyzm rozwiązań

Klient

- utrzymanie budżetu
- finansowanie za sukces
- elastyczność zakresu
- doskonałość i pełność rozwiązań



4BRE-PM 11 InfoViDE sp. z o.o.

Praca grupowa

Interdyscyplinarny zespół, grupujący przedstawicieli użytkownika i wykonawcy

- Identyfikacja i analiza potrzeb
- wypracowanie koncepcji rozwiązań
- określanie zakresu
- ocena i wybór wariantów rozwiązań

Wykonawca

- prowadzi dokumentację
- organizuje przebieg sesji

4BRE-PM 12 InfoViDE sp. z o.o.

Praca grupowa a odpowiedzialność

Po stronie użytkownika

- komunikowanie potrzeb
- autoryzacja decyzji
- przestrzeganie reguł pracy grupowej
- konstruktywna krytyka

Po stronie wykonawcy (zwykle)

- proces (plan, produkty, standardy)
- kontrola postępu prac i jakości rezultatów

4BRE-PM 13 InfoVIDE sp. z o.o.

Współprojektowanie (JAD)

Współprojektowanie (JAD - Joint Application Development) polega na warsztatowym tworzeniu wybranych produktów w zespołach skupiających zarówno użytkowników jak i wykonawców systemu.

Istotnym elementem jest łączenie dyscypliny i koncentracji działań z wykorzystaniem „psychologicznych" technik pobudzania twórczego myślenia.

4BRE-PM 14 InfoVIDE sp. z o.o.

Współprojektowanie — trzy etapy:

4BRE-PM 15 InfoVIDE sp. z o.o.

JAD - opracowanie rezultatów

4BRE-PM 16 InfoVIDE sp. z o.o.

JAD a definiowanie potrzeb

Sesje JAD doskonale nadają się do

- planowania projektu
- budowanie modeli
- opracowania koncepcji zakresu
- identyfikacji i rozstrzyganie dobrze zdefiniowanych problemów i kwestii spornych

4BRE-PM 17 InfoVIDE sp. z o.o.

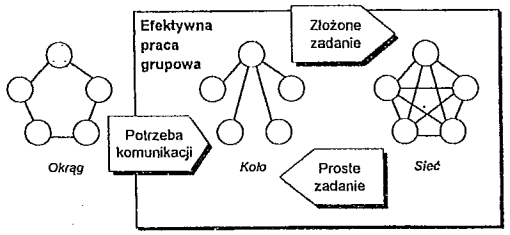
JAD a definiowanie rozwiązań

Sesje JAD znajdują zastosowanie przy:

- identyfikacji i ocenie potencjalnych rozwiązań
- projektowaniu koncepcji rozwiązań
- ocenie prototypów rozwiązań

4BRE-PM 18 InfoVIDE sp. z o.o.

JAD - organizacja pracy grupy



4BRE-PM 19 InfoViDE sp. z o.o.

JAD - Instrumenty

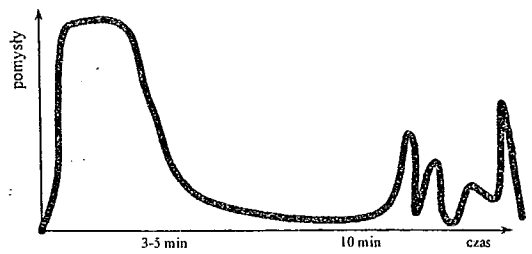
Generowanie koncepcji
- warianty „burzy mózgów"

Definiowanie, precyzowanie i ograniczanie koncepcji
- klasyfikacja i usuwanie redundancji
- techniki określania priorytetów
- ustalanie kryteriów wyboru

Ocena postępu prac
- listy kontrolne
- plan sesji

4BRE-PM 20 InfoViDE sp. z o.o.

Jak się mózgi burzą



4BRE-PM 21 InfoViDE sp. z o.o.

Wszystko się może zdarzyć

Jak poprowadzić twórczą sesję zamiast:
- narady produkcyjnego
- występu iluzjonisty
- wojny biurowej
- .....

czyli:

problemy z ludźmi

4BRE-PM 22 InfoViDE sp. z o.o.

Jawne role w zespole

Prowadzący
- Obsługuje tablicę i flipchart
- Prowadzi dyskusję
- Ogłasza przerwy

Uczestnicy
- Są właściwymi autorami produktu spotkania
- Wymyślają wszystko
- Dyskutują i decydują

Dokumentalista
- Przygotowuje wstępną wersję produktu spotkania
- Może prosić o więcej precyzji

Obserwatorzy
- Nie odzywają się i uważają
- Ich obecność ma ułatwić pracę im, a nie grupie

4BRE-PM 23 InfoViDE sp. z o.o.

Prowadzenie spotkania

- Ustalenie reguł
- Podstawowe zadania
- Generowanie pomysłów
- Utrzymanie dyscypliny

4BRE-PM 24 InfoViDE sp. z o.o.

Podstawowe reguły na początek

- Tylko jedna rozmowa
- W jednej chwili jeden temat
- Nowe problemy odkładamy na później
- Milczenie oznacza zgodę
- Bez krasomówstwa

4BRE-PM 25 InfoVIDE sp. z o.o.

Właściwa postawa

- Słuchaj innych uczestników
- Wszyscy uczestnicy są równouprawnieni
- Bierz pod uwagę opinie innych uczestników
- Zapomnij o swojej zwykłej pracy
- Zapomnij o swoim ego

4BRE-PM 26 InfoVIDE sp. z o.o.

Reguły dla prowadzącego

- Rozpocznij sesję o czasie
- Nie toleruj spóźnień
- Zapewnij, że sesja skończy się w planowanym czasie (albo wcześniej)
- Prowadzący ogłasza przerwy
- Wykorzystaj sesję do oporu
- Nie narzucaj swojego zdania
- Bądź zdecydowany

4BRE-PM 27 InfoVIDE sp. z o.o.

Kontrola dyskusji

Regulowanie

- Przerywaj krasomówcom, nie pozwól na nadmierną komplikację

Koncentrowanie

- Aktualny temat powinien być widoczny dla grupy

Interpretowanie

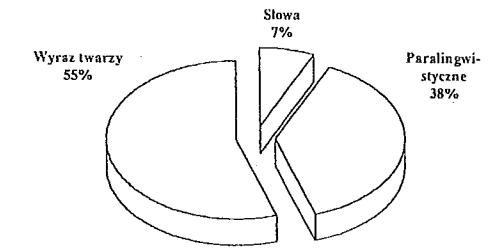
- Przeformułowuj trudne zdania

Podsumowania

- Pokazuj, jak rozwija się dyskusja i do czego doszła grupa

4BRE-PM 28 InfoVIDE sp. z o.o.

Składniki komunikacji



4BRE-PM 29 InfoVIDE sp. z o.o.

Utrzymanie tempa

Przyśpieszanie

- Mów szybciej
- Bądź bardziej dyrektywny i formalny
- Ucinaj dyskusje
- Zadawaj zamknięte pytania

Zwalnianie

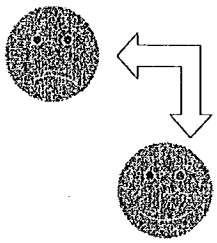
- Mów wolniej
- Więcej przykładów, anegdot, opowieści z życia
- Zadawaj otwarte pytania
- Wywołuj dyskusje

4BRE-PM 30 InfoVIDE sp. z o.o.

Role ukryte

Role ukryte są wyrazem procesów grupowych, a więc:

- odgrywanie roli ukrytej nie jest złośliwością uczestnika, a ponad to
- możemy wykorzystać ten mechanizm dla usprawnienia pracy grupy



4BRE-PM 31 InfoViDE sp. z o.o.

„Psychologiczne judo"

Warchoł, krytykant

- Zatrudnij go jako „adwokata diabła"

Wszystkowiedzący mówca

- Traktuj jak eksperta w dziedzinie. Pozwól poprowadzić fragment spotkania

4BRE-PM 32 InfoViDE sp. z o.o.

„Psychologiczne judo" w praktyce

Narzekacz

- Poproś o prowadzenie listy zarzutów i wątpliwości.
- Przejrzyj listę na zakończenie spotkania (uprzedź o tym!!)

Szeptacz

- Przyjmij zasadę, że dla oszczędności czasu, ci którzy nie rozumieją, będą pytać sąsiadów, zamiast przerywania sesji.

4BRE-PM 33 InfoViDE sp. z o.o.

„Psychologiczne judo" w praktyce

Milczek

- Przyjmij zasadę, że niektórzy ludzie nie życzą sobie brać aktywnego udziału w sesji. To nie znaczy, że nie rozumieją. To znaczy tylko, że nie biorą udziału.

Szef

- Zaproponuj przejście na pozycję obserwatora. Poproś o przygotowanie podsumowania spotkania i udział w podjęciu ostatecznej decyzji.

4BRE-PM 34 InfoViDE sp. z o.o.

JAD w praktyce

Opracowanie strategicznej architektury SI dla dużej firmy handlowej (inicjatywa Zarządu)

- logistyka, zaopatrzenie, zbyt, sprzedaż, marketing

Proces

- JAD - podstawowa metoda pracy (6 tygodni z 3 miesięcy)
- sesje z zarządem (2 x 10 osób)
- sesje z użytkownikami (10 x 15 osób)

Produkt

- bardzo szczegółowa specyfikacja konstruktywnie oceniona przez użytkowników
- decyzja zrobić/kupić
- utrzymanie specyfikacji i jej bieżące wykorzystanie w rozwoju SI

4BRE-PM 35 InfoViDE sp. z o.o.

Joint Application Development

techniki efektywnej współpracy z użytkownikiem

InfoViDE

4BRE-PM 36 InfoViDE sp. z o.o.

Wydawca:

Polskie Towarzystwo Informatyczne
Oddział Górnośląski
40-014 Katowice
ul. Św. Jana 10

Druk i oprawa:

Zakłady Mechaniczne „BUMAR-ŁABĘDY" S.A.
Drukarnia Centrum Informatyki
44-109 Gliwice 9, ul. Mechaników 9

Przygotowanie do druku i projekt okładki:

Mirosława Minich

DZIESIĄTA GÓRSKA SZKOŁA PTI SZCZYRK '98 -- 22-26 CZERWCA 1998

DOSKONALENIE PROCESÓW KONSTRUOWANIA I UŻYWANIA OPROGRAMOWANIA A INTEGRACJA SYSTEMÓW INFORMACYJNYCH

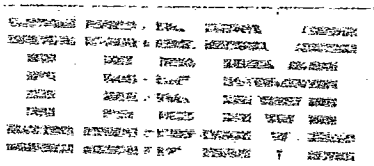
DATA	SESJA INSPIRE				PREZENTACJE INTEGRATORÓW	DYSKUSJE PANELOWE	
	9:00-9:45	10:00-10:45	11:15-12:45	15:00-15:45	16:00-16:45	17:15-18:45	
PONIEDZIAŁEK 22.06.98	Dr Marek Maniecki *Zapewnienie jakości w wytwarzaniu oprogramowania*	Prof. Janusz Górski *Polityka bezpieczeństwa*	Christophe Debou *A poor man's guide to Software Project leader*	Prof. Janusz Górski *Inspekcje oprogramowania*	Piotr Sobolewski *Architektura rozwiązania z perspektywy integracji systemów*	Dariusz Samól, Piotr Jankiewicz *Integracja łańcucha dostaw poprzez moduły oprogramowania R3 v.4.0*	Wykłady Laureatów konkursu PTI na najlepszą pracę dyplomową: Daniel Słobodzian
WTOREK 23.06.98	Dr Marian Kuraś, Agnieszka Zając *Powodzenie i zagrożenia zamierzeń informatycznych*	Christophe Debou *The Capability Maturity Model (CMM): Theory and Practice*	Tomasz Byzia *Zarządzanie jakością oprogramowania*	Prof. Piotr Jędrzejowicz *W jakim celu oraz w jaki sposób mierzyć niezawodność oprogramowania?*	Tomasz Gutt *Doświadczenia ICL we wdrażaniu zintegrowanych systemów zarządzania przedsiębiorstwami*	**Co to jest integracja Systemów informatycznych?** PP. Byzia, Gutt, Jankiewicz, Sobolewski	
ŚRODA 24.06.98	Dr Marek Miłosz *Estymacja zasobów w projektach informatycznych*	Jacek Królik *Techniki estymacji – metoda punktów funkcyjnych a szacowania eksperckie*	Dr Marian Kuraś, Prof. Zbigniew Nęcki *Psychologiczne aspekty komunikacji z użytkownikiem*	Bogusław Jackowski *Mocne marki czyli zarządzanie dokumentacją. Opis techniki*	Arek Balcerek *System Live Cycle 3, czyli jak EDS wychodzi naprzeciw potrzebom klientów*	**Psychologiczne aspekty informatyki** PP. Stokalski, Kuraś, Fuglewicz	
CZWARTEK 25.06.98	Janusz Wielki *Podstawowe elementy krytyki reengineeringu i analiza ich zasadności*	Dr Jerzy Nawrocki *PSP – osobisty proces tworzenia oprogramowania*	Dr Andrzej Kobylinski *Miary oprogramowania* / Prof. Eberhard E. Rudolph *Measuring the software development process and it's result*	Dr Tadeusz Wilusz *Edukacja informatyczna jako ważny kontekst dyskusji o jakości oprogramowania*	Dr Bożena Skibicka *Procesowa organizacja firmy w praktyce*	**Edukacja informatyczna** PP. Kuraś, Miłosz, Nawrocki, Wilusz	
PIĄTEK 26.06.98	Grzegorz Prochowski *Zarządzanie projektami metodą «Earned Value»*	Tomasz Byzia *SPICE standard oceny i doskonalenia procesów produkcji oprogramowania*	Borys Stokalski *Joint Application Development – techniki efektywnej współpracy z użytkownikiem*	**W ŚRODĘ O GODZINIE 20:00 NOC – ŚWIĘTO JACKA KACZMARSKIEGO W NOWYM PROGRAMIE „MIĘDZY NAMI"**			

PTI

Zarząd Główny
ul. Żelazna 87
00-879 Warszawa
tel./fax (0-48-22) 552-32-59
pti_zg@organ.waw.pl

Oddział Górnośląski
ul. Św. Jana 10
40-012 Katowice
tel./fax (048-32) 253-96-78
ptiog@netpol.pl

CZŁONKOWIE WSPIERAJĄCY POLSKIE TOWARZYSTWO INFORMATYCZNE



IBM Polska Sp. z o.o.
02-652 Warszawa, Wiśniowy Business Park, ul. 1 Sierpnia 2a

OCÉ Poland
02-232 Warszawa, ul.Lopuszańska 53

oraz

Centralny Ośrodek Informatyki Górnictwa S.A., 40-065 Katowice, ul Mikołowska 100

Fabryka Wtryskarek „PONAR-ŻYWIEC" S.A., 34-300 Żywiec, ul;. Stolarska 21

ICL Poland Sp. z o.o., 01-199 Warszawa ul. Leszno 21

InfoVide Sp. z o.o., 02-119 Warszawa, ul. Pruszkowska 17

Ster-Projekt Sp. z o.o., 02-652 Warszawa, ul. Magazynowa 1

Zakłady Przemysłu Barwników „BORUTA" S.A., 95-100 Zgierz, ul. Struga 30

International Business Systems, 51-124 Wrocław, ul. H Kamieńskiego 57

OPIEKA PRASOWA

Informatyka