Inżynieria Oprogramowania modele procesu produkcji

Dr hab. inż. Ilona Bluemke

Plan wykładu

- Model wodospadowy
- Model ewolucyjny
- Prototypowanie
- Formalne transformacje
- Metoda iteracyjna
- Metoda reuse
- Model spiralny
- Czynniki nie-techniczne w inżynierii oprogramowania

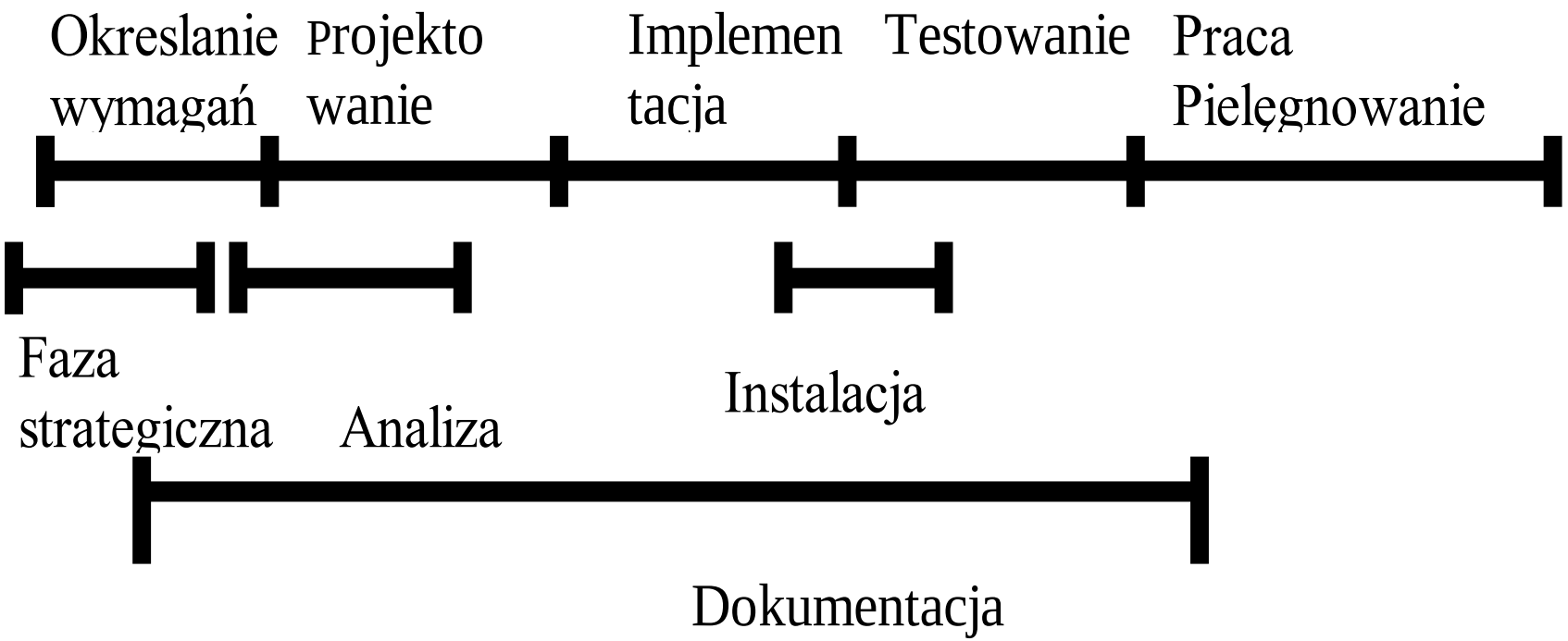
Model wodospadowy (waterfall) (Royce 1970)

Inaczej kaskadowy, liniowy

Analogia do prowadzenia prac inżynieryjnych.
Wprowadzono fazy:

- **specyfikacji wymagań**
- **projektowania** oprogramowania (design)
- i**mplementacji** (implementation, coding)
- **testowania**
- **użytkowania i pielęgnowania**

Nakładanie się faz



Estymacja kosztów

	Wymagania/ projekt	Implement.	testy
s. sterow.	46	20	34
s. operac.	33	17	50
s. nauk.	44	26	30
s. biznes.	44	28	28

Zalety i wady modelu wodospadowego

Zalety:

- łatwość zarządzania przedsięwzięciem (planowanie, harmonogramowanie, monitorowanie)
- narzucenie kolejności wykonywania prac

Wady modelu:

- narzucenie kolejności wykonywania prac
- wysoki koszt błędów popełnionych we wczesnych fazach
- długa przerwa w kontaktach z klientem

Rezultaty faz modelu wodospadowego - 1

- Analiza wymagań - studium wykonalności, „zgrubne” wymagania
- Definicja wymagań - dokument opisujący wymagania
- Specyfikacja systemu - funkcjonalna specyfikacja systemu, plan testów akceptacyjnych, szkic podręcznika użytkownika
- Projektowanie architektury - specyfikacja architektury, testy systemowe
- Projektowanie interfejsów - specyfikacja interfejsów, testy integracyjne

Rezultaty faz modelu wodospadowego - 2

- Projektowanie jednostek - projekt szczegółowy, testy jednostkowe
- Kodowanie - kod programu
- Testowanie jednostek - raport testowania
- Testowanie modułów - raport testowania
- Testowanie integracyjne - raport testowania integracyjnego, podręcznik użytkownika
- Testowanie systemowe - raport testowania
- Testowanie akceptacyjne - system i dokumentacja

W & W

Weryfikacja

- Czy właściwie budujemy produkt ?
- Czy spełnia wymagania ?

Walidacja

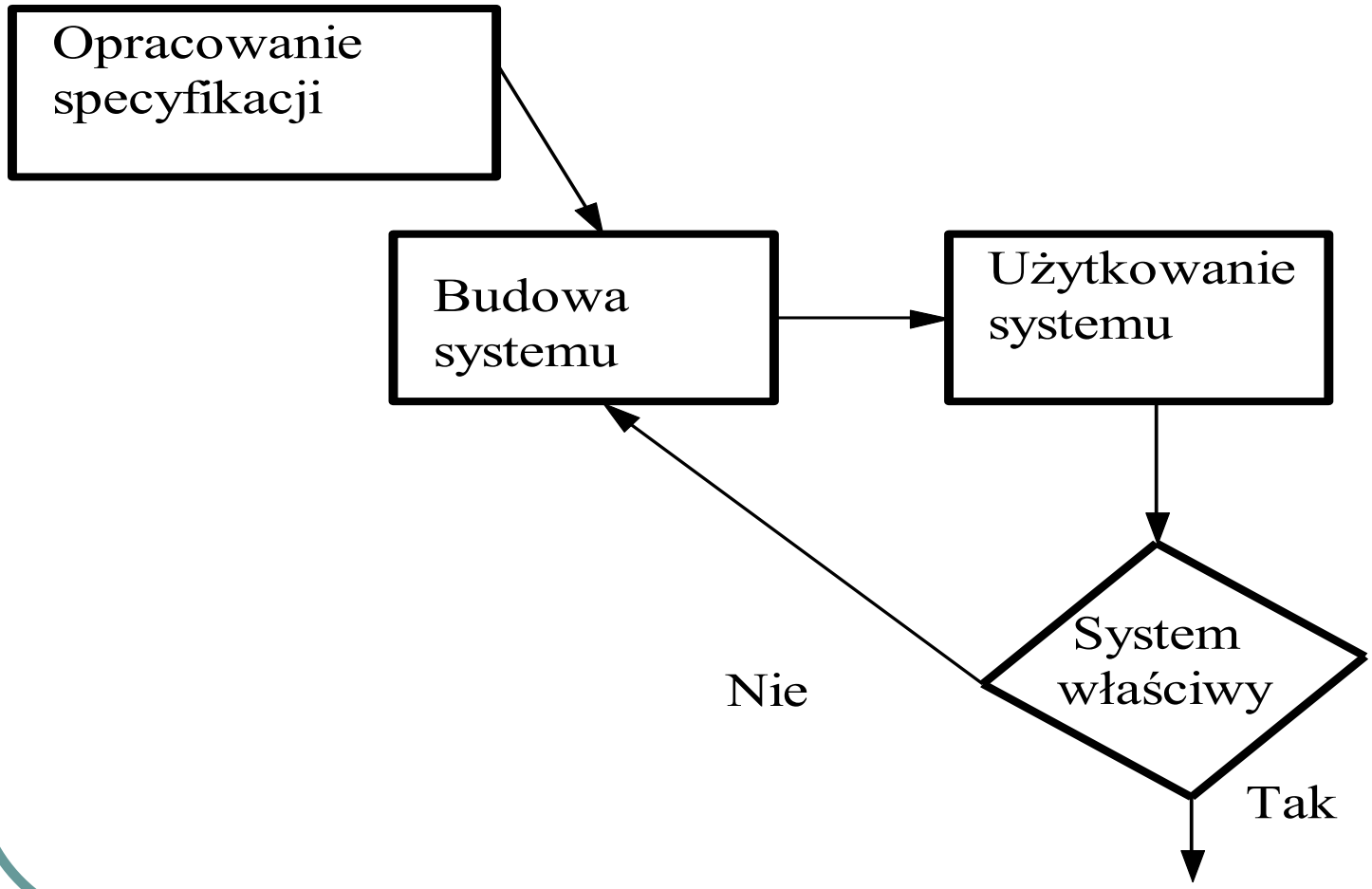
- Czy budujemy właściwy produkt ?
- Czy funkcje produktu są takie jak klient naprawdę oczekiwał ?

Model ewolucyjny

(exploratory programming), odkrywczy

Stosowany w przypadkach, gdy określenie dokładnych wymagań klienta nie jest możliwe (np. systemy sztucznej inteligencji).

Model ewolucyjny - 2



Zalety i wady m. ewolucyjnego

Zalety:

- możliwość stosowania nawet w przypadkach kłopotów z określeniem wymagań klienta.

Wady:

- struktura systemu b. zagmatwana
- konieczność bardzo szybkiej produkcji stąd użycie języków takich jak Lisp, Prolog, środowisk produkcji
- nie ma weryfikacji (sprawdzenia czy spełnia wymogi)
- nie gwarantuje możliwości pielęgnowania systemu

Prototypowanie

Fazy:

- ogólne określenie wymagań
- opracowanie szybko działającego prototypu
- walidacja prototypu przez klienta
- określenie szczegółowych wymagań
- opracowanie pełnego systemu

Prototyp – cele, możliwości

Celem prototypu jest wykrycie:

- trudnych usług
- braków w specyfikacji
- nieporozumień między klientem a projektantami

Prototyp pozwala na:

- demonstrację pracującego systemu
- daje możliwości szkolenia zanim zostanie zbudowany pełen system

Budowa prototypu

- programowanie ewolucyjne
- wykorzystanie gotowych komponentów
- niepełna realizacja
- języki wysokiego poziomu
- generatory np. interfejsów użytkownika

Formalne transformacje

Wymagania wobec systemu są zapisywane w języku formalnym. Podlegają one automatycznym przekształceniom do programu.

Zaleta:

- wysoka niezawodność (brak błędów przy transformacjach)

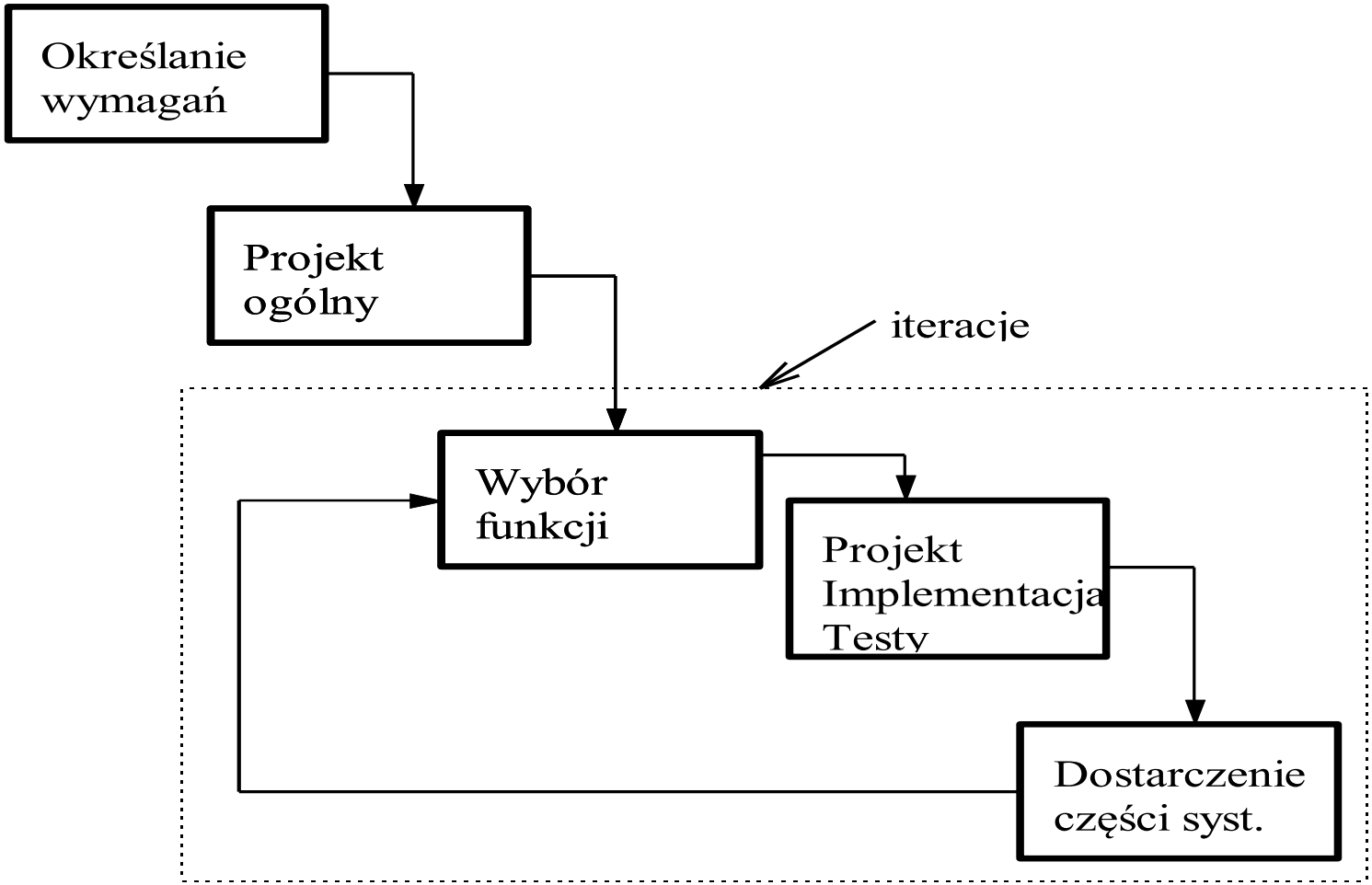
Wady:

- trudności formalnego specyfikowania
- mała efektywność kodu

Formalne specyfikacje

- Stosuje się ją do produkcji systemów wymagających dużej niezawodności, bezpieczeństwa.
- Formalne specyfikacje wymagają dużych umiejętności zespołu.
- Nie wszystkie wymagania można formalnie wyspecyfikować np. nie można formalnie wyspecyfikować interfejsu użytkownika.

Realizacja przyrostowa (incremental development)



Kryteria wyboru funkcji do realizacji

- Priorytet dla klienta
- Łatwość realizacji
- Przydatność dla dalszych iteracji

Wady i zalety metody iteracyjnej

Zalety:

- częsty kontakt z klientem
- możliwość wczesnego wykorzystywania części systemu

Wady:

- - dodatkowy koszt związany z realizacją fragmentów systemu (pisanie szkieletów modułów)

Montaż z gotowych elementów (off-shell programming, reuse)

W montażu z gotowych elementów (ang. reuse, off-shell programming) korzysta się z gotowych, dostępnych komponentów i tworzy się kod integrujący je.

COTS ang. Commercial Off The Shelf

Stosowanie :

- bibliotek
- języków czwartej generacji
- pełnych aplikacji

Reuse - Proces produkcji - 1

- Specyfikacja wymagań.
- Analiza komponentów - poszukiwanie komponentów spełniających funkcje systemu, zwykle komponenty spełniają tylko część wymagań.
- Modyfikacja wymagań – dostosowanie wymagań do znalezionych komponentów, w przypadku wymagań bardzo istotnych dla systemu, dla których nie znaleziono komponentów, poszukiwanie rozwiązań alternatywnych.

Reuse – proces produkcji - 2

- Projektowanie systemu z komponentami – zaprojektowanie „połączeń” między komponentami, ewentualne zaprojektowanie kodu realizującego wymagania, dla których komponenty nie były dostępne.
- Realizacja systemu i integracja – implementacja i testowanie zaprojektowanego kodu i integracja komponentów.

Wady i zalety metody reuse

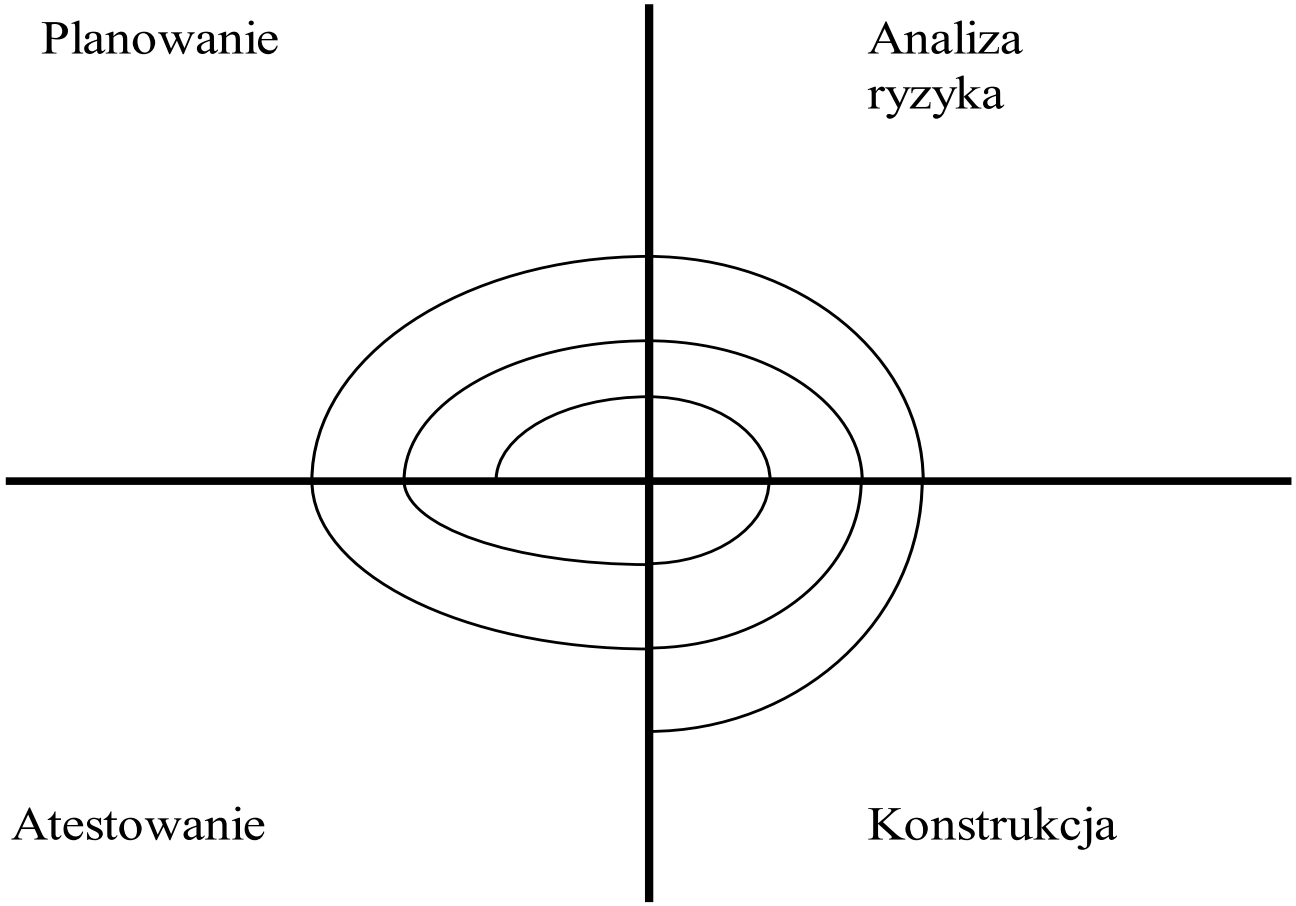
Zalety:

- wysoka niezawodność
- narzucenie standardów
- małe koszty, szybko

Wady:

- dodatkowy koszt przygotowania elementów do ponownego użycia
- ryzyko uzależnienia od dostawcy komponentu
- wysokie koszty pielęgnowania
- nie w pełni realizowane wymagania klienta.

Model spiralny (Boehm)



Sektory spirali

Każda spirala składa się z czterech sektorów:

1. Określenia celów, identyfikacja ograniczeń, poszukiwanie alternatywnych rozwiązań.
2. Analiza ryzyka związanego z proponowanymi rozwiązaniami, redukcja ryzyka np. w przypadku ryzyka związanego z wymaganiami budowa prototypu.
3. Po określeniu ryzyka wybiera się najlepsze rozwiązanie (o najmniejszym ryzyku).
4. Planowanie następnej spirali, podjęcie decyzji dotyczącej kontynuacji.

Model spiralny

W modelu spiralnym można włączać inne modele. Np. prototypowanie można zastosować do wykrycia braków w wymaganiach, formalne specyfikacje do budowy fragmentów systemu, dla których są wymagane bardzo wysokie parametry niezawodnościowe, a model wodospadowy dla podsystemów o dobrze określonych wymaganiach.

Możliwość obserwowania postępu wykonywanych prac

Modele o **dobrej obserwowalności** :

- wodospadowy ,
- formalnych transformacji (każda transformacja kończy się dokumentem, raportem),
- spiralny (w każdym segmencie spirali powstaje dokument).

Modele o **słabej obserwowalności**

- Montaż z gotowych komponentów
- Najmniejsza możliwość obserwowania procesu produkcji jest w modelu ewolucyjnym

Przykładowe pytania

- Wady i zalety modelu wodospadowego.
- Jakie są „deliverable” w modelu wodospadowym ?
- Kiedy można stosować model wodospadowy ?
- Kiedy stosuje się prototypowanie ?
- Dla jakiego typu oprogramowania stosuje się formalne specyfikacje ?
- W jakich modelach procesu produkcji oprogramowania zajmujemy się analizą ryzyka ?
- Na czym polega walidacja ? Na czym polega weryfikacja ?
- Jakie kryteria wyboru funkcji do realizacji możemy stosować w modelu iteracyjnym ?

Czynniki nie-techniczne w inżynierii oprogramowania

- Zarządzający projektem powinien rozumieć zespół projektowy, osoby indywidualne, ich interakcje, lepsze zrozumienie pozwala na stawianie realnych celów
- Systemy są użytkowane przez ludzi i ich możliwości i ograniczenia powinny być brane pod uwagę podczas projektowania systemu.
- Znajomość czynników społecznych pozwala zwiększyć produktywność pracowników

Proporcje czasu

- praca własna
- czynności nieprodukcyjne
- interakcje

(30% , 20% , 50%)

Zespoły których członkowie znają się (wady, zalety) mają większą wydajność niż takie, w których członkowie mają niewielki kontakt ze sobą.

Osobowości w grupie

zorientowany na:

- **na zadania** - motywacją jest sama praca
- **na siebie** - motywacją jest dążenie do sukcesu, praca jest środkiem
- **na interakcje** - motywacją jest obecność w grupie

Najlepsze efekty - zespoły zróżnicowane, przywódcą osoba zorientowana na pracę.

ego -less programming

- Program uważany jako "własny", trudno przyjmowana krytyka, traktowany jako bezbłędny
- Błędy - normalne, muszą wystąpić
- Program traktowany jako "wspólny" - **ego -less**, przyjmowana krytyka

Przywódca grup

- **wyznaczony** może pełnić funkcje administracyjne
- **wyłaniany** mający największy wpływ na zespół, osobowość dominująca, często na każdym etapie inny

Integracja zespołu

Lojalność w grupie, integracja

- + współpraca, pomoc
- - utrata krytycyzmu

Interakcje w grupie

Czynniki wpływające na efektywność interakcji:

- wielkość grupy
- struktura
- status i osobowość członków
- środowisko pracy

n (n -1) liczba potencjalnych dróg komunikacyjnych ***n*** członków grupy