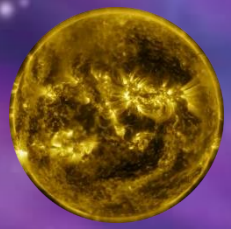
PROJEKT ZESPOŁOWY

TEAM PROJECT & TEAMWARE



NARZĘDZIA PRACY GRUPOWEJ

Adrian Horzyk

Akademia Górniczo-Hutnicza

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Katedra Automatyki i Inżynierii Biomedycznej, Laboratorium Biocybernetyki

30-059 Kraków, al. Mickiewicza 30, paw. C3/205

horzyk@agh.edu.pl, Google: Adrian Horzyk

TEAM PROJECT

CELE PROJEKTÓW ZESPOŁOWYCH

- ✓ **Opanowanie umiejętności pracy w grupie poprzez:**
  - **Rozwinięcie inteligencji społecznej**
  - **Poznanie technik ułatwiających kooperację**
  - **Nabycie umiejętności rozpoznawania charakteru i potrzeb adwersarzy**
  - **Poznanie narzędzi informatycznych wspomagających pracę w grupie oraz realizację wspólnych projektów zespołowych**
- ✓ **Opanowanie technik związanych z prowadzeniem projektu, tj.: zbieranie informacji, projektowanie, planowanie prac, kontroli błędów i dokumentowania (tworzenie specyfikacji wymagań, zarysu architektury, specyfikacji technicznej, instrukcji wdrożeniowej, scenariuszy testów automatycznych i jednostkowych itp.)**
- ✓ **Przyswojenie dobrych praktyk programowania zapewniających wykonanie powierzonych zadań w ograniczonym czasie przez harmonogram projektu.**
- ✓ **Zdobycie umiejętności pracy w grupie poprzez udział w wybranym projekcie.**
- ✓ **Umiejętność wykorzystywania narzędzi pracy grupowej (TeamWare)**

TEAM PROJECT

PRACA GRUPOWA

Praca w grupie cechuje się wieloma zaletami i polega na:

- ✓ **Wspólnym gromadzeniu wiedzy z zakresu realizowanego projektu**
- ✓ **Współpracy w tworzeniu projektu i jego realizacji**
- ✓ **Możliwości dzielenia się swoimi umiejętnościami i specjalizacją**
- ✓ **Możliwości wymiany doświadczeń**
- ✓ **Prowadzeniu konstruktywnych dyskusji i burzy mózgów**
- ✓ **Udostępnianiu członkom zespołu wyników oraz narzędzi**
- ✓ **Wzajemnym motywowaniu się i wspieraniu**
- ✓ **Tworzeniu standardów porozumienia się i narzędzi ułatwiających kooperację członków grupy, jak również wymianę doświadczeń.**

TEAM PROJECT

ZŁOTE ZASADY PRACY W GRUPIE

1. Mówimy zawsze w swoim imieniu.
2. Uważnie słuchamy wypowiedzi innych osób.
3. Dopytujemy o szczegóły i parafrazujemy w celu upewnienia się, iż właściwie zrozumieliśmy innych.
4. Staramy się nie przerywać innym w trakcie ich wypowiedzi.
5. Szanujemy czas swój i innych, nie wygłaszając zbyt długich monologów.
6. Nie oceniamy wypowiedzi innych osób, lecz próbujemy zrozumieć ich punkt widzenia.
7. Nie krytykujemy innych osób, za to kim są, jak wyglądają itp.
8. Wszelkie działania podejmujemy z nastawieniem na współpracę i osiągnięcie wspólnych korzyści, a nie rywalizację, dominację lub walkę.
9. Osobiste uprzedzenia do innych zostawiamy za drzwiami i dajemy wszystkim szansę na współpracę.
10. Staramy się zachowywać asertywnie, dbając o poszanowanie swoich praw, lecz również nie raniąc innych. Pamiętając też o poczuciu humoru.
11. Staramy się być tolerancyjni i życzliwi wobec siebie i innych, dając każdemu prawo do popełnienia błędu i osiągnięcia sukcesu. Próbujemy się jednak zabezpieczyć przed powtarzaniem tych samych błędów.

TEAM PROJECT

POZYTYWNE ZACHOWANIA

1. Bądź życzliwy, miły, cierpliwy i pozytywnie nastawiony do innych.
2. Ceń własne pomysły, lecz doceniaj innych i próbuj ich włączyć do dyskusji.
3. Dbaj o atmosferę pracy w grupie oraz pamiętaj o humorze i dobrym słowie.
4. Jeśli nie umiesz, nie krytykuj, lecz staraj się doceniać i chwalić pozytywne cechy.
5. Liczy się praca wszystkich, więc spróbuj wykorzystać indywidulane zdolności i zalety.
6. Pracuj w małych zespołach oraz dziel duże grupy na mniejsze.
7. Pytaj i proś, a następnie wyraź swoją wdzięczność i uznanie.
8. Szukaj mocnych stron partnerów, pozwól na specjalizację i nawiązuj współpracę.
9. Jeśli umiesz już mówić, naucz się słuchać i brać pod uwagę zdanie innych.
10. Jeśli umiesz już słuchać, naucz się mówić i wyrażać swoje zdanie.
11. Mów, jeśli chcesz, by inni uwzględniali Twoje zdania.
12. Uzgadniaj, a nie uśredniaj.
13. Nie uogólniaj negatywnych postaw ani zdarzeń.

TEAM PROJECT

FUNKCJONALNOŚĆ TEAMWARE

Pracę grupową możemy obecnie wspomagać wykorzystując różne narzędzia informatyczne, które pozwalają na:

- ✓ **Wspólne projektowanie i planowanie.**
- ✓ **Tworzenie listy zadań oraz przydzielanie i rozliczanie zadań.**
- ✓ **Śledzenie wykonania zadań, procesów oraz ich zależności.**
- ✓ **Określanie kamieni milowych.**
- ✓ **Szacowanie czasu wykonania projektu oraz poszczególnych jego części**
- ✓ **Określanie ścieżki krytycznej decydującej o terminie zakończenia projektu oraz działania interwencyjne.**
- ✓ **Tworzenie wspólnych zbiorów danych, tekstów, baz danych, repozytoriów programistycznych, wersjonowania kodu źródłowego.**
- ✓ **Gromadzenie wiedzy i lepsze wykorzystanie zasobów grupy.**
- ✓ **Współpracę i pracę asynchroniczną.**

TEAM PROJECT

TYPOWE PROBLEMY PROGRAMISTYCZNE

- ✓ 10% czasu zajmuje zrobienie 90%, 90% czasu dokończenie 10%.
- ✓ Nie dokładne określenie, co program ma robić: „Przynieś mi kamień.”
- ✓ Zebrać wszystkie osoby klienta związane z przedmiotem projektu w celu określenia, co każdy z nich oczekuje od systemu.
- ✓ Określenie priorytetów.
- ✓ Dokumentacja wstępna – określająca możliwie wszystkie funkcje systemu.
- ✓ Zaangażowanie osób klienta w realizację praktyczną projektu.
- ✓ Nie pisze, że ma robić, ale nie pisze, że nie ma robić – problem funkcjonalności.
- ✓ Wprowadzanie zmian w trakcie realizacji projektu, a nawet zmiana założeń – problem podnoszenia kosztów realizacji przedsięwzięcia.
- ✓ Budowa zespołu realizującego projekt – stawianie na doświadczenie – problem kosztów komunikacji dużego zespołu programistów (lepiej mniejsza grupa bardziej droższych i doświadczonych niż duża grupa tanich i niedoświadczonych).
- ✓ Raportowanie i kontrola realizacji projektu.
- ✓ Przydzielanie strategicznych części projektu (uzależniających realizację terminową – miejsca gardłowe ścieżki krytycznej realizacji projektu) najbardziej doświadczonym.
- ✓ Metodologie: wstępująca, zstępująca, mieszana.

TEAM PROJECT

PROGRAMOWANIE EKSTREMALNE

- ✓ Nowoczesny kierunek określający reguły programowania.
- ✓ Stawianie na efektywność i jakość kodu
- ✓ Brak nadmiarowości (unikanie zbędnej funkcjonalności)
- ✓ Praca w parach (wykrywanie błędów już na etapie pisania, unikanie żmudnej analizy błędów)
- ✓ Opowiadanie klienta (*User Stories*) – służy zbudowaniu aplikacji
- ✓ Zbiory testów zbudowanych na bazie przykładów opracowanych przez użytkowników (*User Cases*) – służą sprawdzaniu założeń zdefiniowanych przez klienta
- ✓ Testy pisze się najpierw i piszą je inni programiści niż ci, którzy realizują projekt
- ✓ UML – przygotowanie projektu systemu (automatyczna generacja interfejsów przekazywanych grupom roboczym)
- ✓ Wybór technologii dla projektu – zależna od realizowanego zadania, np.: C++ x Java: C++ 4x szybszy, zaś w Javie 2x szybciej się pisze aplikacje + przenaszalność
- ✓ Zarządzanie wersjami: CVS, BitKeeper – repozytorium plików i zmian, porównania, łączenie
- ✓ Określanie ryzyka opóźnień (*Trouble Ticketing*)
- ✓ Poprawianie wydajności (*Profilling*) – odnajdywanie procedur często wykonywanych i ich optymalizacja
- ✓ Ponowne wykorzystanie części kodu napisanego w przeszłości (*Refactoring*)

TEAM PROJECT

PROGRAMOWANIE EKSTREMALNE

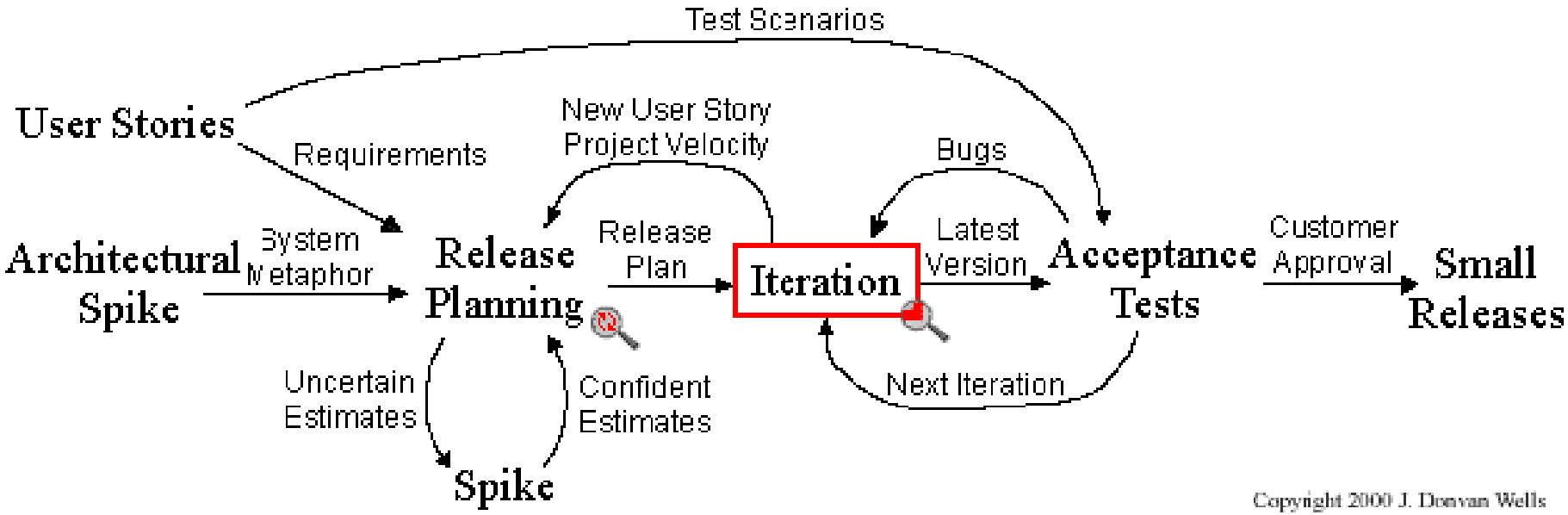
- ✓ Moment weryfikacji systemu to moment prezentacji klientowi – „Co Pan to przyniósł?”
- ✓ Zmiany wymagań w trakcie realizacji systemu
- ✓ Modyfikacje „ad-hoc” produkują często inne błędy lub powodują rozbieżność z dokumentacją
- ✓ Problemy z dostawcami (np. sprzętu)
- ✓ Eskalacja konfliktów – umiejętność reagowania na problemy – Extreeme Committee
- ✓ Istota poprawności systemu w zależności od jego zastosowań: telekomunikacja, gra, finansowy, ...
- ✓ Cykl życia produktu: Założenia, Refactoring, Development, Profiling, Testy, Uruchomienie
  - Plany rozwojowe już na etapie założeń (ważne dla wyboru technologii)
  - Prototypy
  - Kiedy warto zacząć od początku? – Inna technologia, ...
  - Reuse kodu
  - Nowy system, nowe problemy – jeśli nie trzeba – nie poprawiaj
- ✓ Z programowaniem jak z medycyną – wszyscy się na niej znają.
- ✓ Nie istnieją rozwiązania idealne
- ✓ Wdrożenie systemu – to interes obopólny
- ✓ Umiejętność szacowania kosztów
- ✓ Tworzenie dokumentacji, np. w DOC++

TEAM PROJECT

PROGRAMOWANIE EKSTREMALNE

- ✓ **Schemat przebiegu procesu programowania ekstremalnego:**

Extreme Programming Project



TEAM PROJECT

ZASADY PROGRAMOWANIA EKSTREMALNEGO

1. Programowanie Ekstremalne bazuje na prostocie, komunikacji, sprzężeniu zwrotnym i agresywności.
2. Ważnym aspektem jest zastosowanie konwencji kodowania w celu ułatwienia komunikacji pomiędzy programistami. Można również łatwiej zrozumieć, co dany kod realizuje.
3. Dobra metoda komentowania zachęca najpierw do poprawy kodu. Jeżeli jakaś metoda nie działa klarownie, lepiej jest ją poprawić tak, żeby działała klarownie niż dodać komentarz wyjaśniający sposób jej działania. Taki system ma za zadanie ułatwienie komunikacji. Stosujemy komentarze
   dla kawałków kodu, żeby zakomunikować, gdzie wszędzie można go wykorzystać.
4. Preferujemy Kolektywną własność kodu niż Indywidualną własność kodu. To poprawia szybkość wykonania czegoś przez ominięcie negocjacji z właścicielem kodu. To jest część agresywności XP.
5. Preferowana jest częste produkowanie kodu wynikowego (raz dziennie lub częściej). To jest możliwe
   i celowe ze względu na Testy Jednostkowe i Funkcjonalne. Praktyka redukuje czas integracji kodu, poprawia komunikację ponieważ pozostaje w stanie zsynchronizowania i jest kluczowym czynnikiem agresywności.
6. Dopuszczamy nadmierne wykorzystanie czasu i kodu w przyszłości dla możliwych przyszłych zastosowań. Taka praktyka utrzymuje kod prosty.
7. Na początku atakowane są problemy najłatwiejsze. Tworzymy system tak prosty, jak to możliwe. Nie rozbudowujemy go o elementy, które „mogą przydać się w przyszłości”.

TEAM PROJECT

ZASADY PROGRAMOWANIA EKSTREMALNEGO

8. Praktykujemy programowanie parami. Poprawia to komunikację, szczególnie w parze, sprawia,
   że kod jest prostszy. Co 2 głowy to nie jedna – przy rozwiązywaniu problemów.
   Stajemy się bardziej uważni, gdy ktoś patrzy, co piszemy.
9. Rozpoczynamy od napisania pierwszego modelu i bardzo spartańskiego interfejsu.
   To pozwala skupić się na bardziej wartościowych i ryzykownych działaniach.
10. Programowanie ekstremalne jest systemem myśli na pracę w celu szybkiego napisania programów. Programowanie ekstremalne to przede wszystkim praktyka.
11. Rozpoczynamy realizację projektu od napisania Testów Jednostkowych.
    Testy jednostkowe umożliwiają szybkie odnajdywanie błędów zaraz na początku.
12. Należy dostrzec możliwość przerobienia starego kodu.
13. Jeżeli jest taka możliwość, trzeba wziąć kawałek podobnego kodu I przerobić go. Kiedyś działający kod poprawnie prościej przerobić niż napisać nowy poprawny od początku.
14. 1-2 programistów pracuje we ścisłej współpracy razem. ALBO 5-10 programistów pracuje w bliskości geograficznej. ALBO 100 programistów jest rozsianych po całym świecie.
15. Nigdy nie piszemy kodu na wyrost, bo mogłaby się kiedyś przydać. Jeśli będzie potrzebne, przerobimy istniejący kod.
16. Należy robić rzeczy tak prosto, jak się tylko da.
17. Bardziej skomplikowany kryje więcej błędów, które trudniej znaleźć i usunąć.

TEAM PROJECT

CO MA ISTOTNE ZNACZENIE?

What really matters? Software is too damned hard to spend time on things that don't matter. So, starting over from scratch, what are we absolutely certain matters?

- ✓ **Coding.** At the end of the day, if the program doesn't run and make money for the client, you haven't done anything.
- ✓ **Testing.** You have to know when you're done. The tests tell you this. If you're smart, you'll write them first so you'll know the instant you're done. Otherwise, you're stuck thinking you maybe might be done, but knowing you're probably not, but you're not sure how close you are.
- ✓ **Listening**. You have to learn what the problem is in the first place, then you have to learn what numbers to put in the tests. You probably won't know this yourself, so you have to get good at listening to clients - users, managers, and business people.
- ✓ **Designing.** You have to take what your program tells you about how it wants to be structured and feed it back into the program. Otherwise, you'll sink under the weight of your own guesses.

TEAM PROJECT

PLANOWANIE – *PLANNING*

- ✓ **Spisanie opowiadania klienta – czego on chce (User stories are written)**
- ✓ **Ostateczne planowanie harmonogramu realizacji projektu (Release planning creates the schedule)**
- ✓ **Częste tworzenie małych gotowców (Make frequent small realeases) -** *The development team needs to release iterative versions of the system to the customers often. The release planning meeting is used to discover small units of functionality that make good business sense and can be released into the customer's environment early in the project. This is critical to getting valuable feedback in time to have an impact on the system's development. The longer you wait to introduce an important feature to the system's users the less time you will have to fix it.*
- ✓ **Pomiar szybkości realizacji projektu (The Project Velocity is measured)**
- ✓ **Podział projektu na etapy (The project is divided into iterations)**
- ✓ **Dobranie wykonawców (Move people around)**

TEAM PROJECT

PROJEKTOWANIE – *DESIGNING*

- ✓ Prostota (Simplicity)
- ✓ Stosowanie odpowiedniego, klarownego i właściwie kojarzącego się nazewnictwa (Choose a system metaphor): obiektów, zmiennych itp.
- ✓ Projektowanie systemu zespołowo (Use CRC - Class, Responsibilities, and Collaboration cards for design sessions) – większa ilość dobrych pomysłów.
- ✓ Tworzenie bardzo prostych rozwiązań pozwalających zbadać potencjalne rozwiązania. (Create spike solutions to reduce risk) Rozwiązania takie często stają się ostatecznymi.
- ✓ Nie dodawaj zbędnej funkcjonalności zbyt wcześnie (No functionality is added early)
- ✓ Wykorzystaj ponownie cokolwiek i gdziekolwiek możliwe (Refactor whenever and wherever posssible)

TEAM PROJECT

IMPLEMENTACJA - *CODING*

- ✓ Klient musi współuczestniczyć w projekcie (The customer is always available)
- ✓ Kod musi być zgodny z obowiązujacymi standardami (Code must be written to agreed standards)
- ✓ Najpierw piszemy testy (Code the unit test first)
- ✓ Wszystkie części kodu piszemy parami (All production code is pair programmed)
- ✓ Tylko jedna para zajmuje się integracją kodu w danym czasie (Only one pair integrates code at a time)
- ✓ Częste łączenie kodu (Integrate often)
- ✓ Wykorzystaj kolektywną własność kodu (Use collective code ownership)
- ✓ Optymalizację kodu pozostaw na koniec (Leave optimization till last)
- ✓ Nie przekraczaj wyznaczonego czasu na poszczególne zadania (No overtime)

TEAM PROJECT

TESTY I TESTOWANIE - *TESTING*

- ✓ Każda część kodu musi posiadać testy (All code must have unit tests)
- ✓ Każda część kodu musi zostać sprawdzona zanim trafi do integracji/łączenia z pozostałym kodem (All code must pass all unit tests before it can be released)
- ✓ Kiedy odnaleziony jest błąd testy są tworzone (When a bug is found tests are created)
- ✓ Testy akceptacyjne są wykonywane często i publikowane są ich wyniki (Acceptance tests are run often and the score is published)

TESTY MOGĄ BYĆ:

- ✓ jednostkowe (unitarne) – uważać na dobre pokrycie danych wejściowych, wykonywane zawsze najpierw
- ✓ funkcjonalne – sprawdzanie poszczególnych funkcji systemy, czy działają poprawnie, stosowane w szczególności przy zmianie wersji
- ✓ regresyjne – stosowane pomiędzy wersjami, spr. czy nie tracą system nie stracił funkcjonalności
- ✓ obciążeniowe (stress testing) – badają ile konstruowany system jest w stanie wytrzymać
- ✓ beta testing – siada człowiek i próbuje doprowadzić system do błędu
- ✓ scenariusze – badają funkcjonalność systemu
- ✓ automatyczne – sprawdzają funkcjonalność i poprawność działania aplikacji wykorzystując zbiory danych lub reguł, dla których znane są wyniki ich działania.

TEAM PROJECT

LITERATURA

- ✓ James Shore, Shane Warden: Agile Development. Filozofia programowania zwinnego, Helion.
- ✓ Kena Schwaber: Sprawne zarządzanie projektami metodą Scrum, Microsoft.
- ✓ Esther Derby, Diana Larsen, Ken Schwaber: Agile Retrospectives. Making Good Teams Great, Pragmatic Bookshelf

TEAM PROJECTS

OPEN ALD LEAD TO THE FUTURE AND COOPERATION