Lekkie metodyki

tworzenia oprogramowania

Programowanie zwinne (Agile software development)

- grupa metodyk wytwarzania oprogramowania opartego o programowanie iteracyjne (model przyrostowy). Wymagania oraz rozwiązania ewoluują przy współpracy samozarządzalnych zespołów, których celem jest przeprowadzanie procesów wytwarzania oprogramowania.

Model przyrostowy (incremental development)

- jedna z technik pisania oprogramowania, stosowana w przypadkach, w których dopuszczalna jest okrojona funkcjonalność systemu.
- Analiza + implementacja + testowanie = integracja kolejnych funkcjonalnych części systemu

Generalnie metodyka oparta jest o zdyscyplinowane zarządzanie projektem, które zakłada częste inspekcje wymagań i rozwiązań wraz z procesami adaptacji (zarówno specyfikacji jak i oprogramowania). Metodyka ta najczęściej znajduje zastosowanie w małych zespołach programistycznych, w których nie występuje problem komunikacji, przez co nie trzeba tworzyć rozbudowanej dokumentacji kodu. Kolejne etapy wytwarzania oprogramowania zamknięte są w iteracjach, w których za każdym razem przeprowadza się testowanie wytworzonego kodu, zebranie wymagań, planowanie rozwiązań itd. Metoda nastawiona jest na szybkie wytwarzanie oprogramowania wysokiej jakości.

Skład zespołów jest zazwyczaj wielofunkcyjny oraz samozarządzalny, bez zastosowania jakiejkolwiek hierarchii korporacyjnej. Członkowie zespołu biorą odpowiedzialność za zadania postawione w każdej iteracji. Sami decydują jak osiągnąć postawione cele.

Metoda nastawiona jest na bezpośrednią komunikację pomiędzy członkami zespołu, minimalizując potrzebę tworzenia dokumentacji. Jeśli członkowie zespołu są w różnych lokalizacjach, to planuje się codzienne kontakty za pośrednictwem dostępnych kanałów komunikacji (wideokonferencja, e-mail itp.).

Podstawowe zasady Manifest Agile (Agile Manifesto) – założenia:

- ludzie i ich wzajemne oddziaływania - interakcje są ważniejsze niż procedury i narzędzia
- przedkłada się działające oprogramowanie nad wyczerpującą dokumentację
- ważniejsza jest współpraca z klientem od negocjacji umów
- istotniejsze jest reagowanie na zmiany niż ścisłe trzymanie się planu

Większość metodyk zwinnych ma swoje zastosowanie w zarządzaniu projektami IT, które są innowacyjne i bardzo kreatywne. Postępowanie według zasad powyższej deklaracji znacznie przyczyni się do poprawienia efektywności oraz skuteczności działania członków projektu.

Najbardziej znane zwinne metodyki to:

- Metodyka Scrum
- Extreme Programming
- Lean Development
- Metodyka Crystal
- Feature Driven Development

Scrum (z ang. „młyn” w grze rugby)

- W tej iteracyjnej metodyce rozwój produktu podzielony jest na mniejsze, trwające od jednego do czterech tygodni, fazy zwane sprintami następującymi bezpośrednio po sobie. Po każdym sprincie zespół pracujący nad rozwojem projektu jest w stanie dostarczyć działający prototyp projektu.

Scrum skupia się na:

- dostarczaniu kolejnych, coraz bardziej dopracowanych wyników projektu,
- włączaniu się przyszłych użytkowników w proces wytwórczy,
- samoorganizacji zespołu projektowego.

Programowanie ekstremalne (eXtreme Programming, XP)

- metodyka programowania mające na celu wydajne tworzenie małych i średnich "projektów wysokiego ryzyka", czyli takich, w których nie wiadomo do końca, co się tak naprawdę robi i jak to prawidłowo zrobić.
- Koncepcja prowadzenia projektu, wywodząca się z obserwacji innych projektów, które odniosły sukces.

Zalecenia w XP

- Iteracyjność
- Nie projektować z góry
- Testy jednostkowe
- Ciągłe modyfikacje architektury
- Programowanie parami
- Stały kontakt z klientem

Lean Software Development

- "Odchudzone" czy inaczej "wyszczuplone" zarządzanie (lean management). Odchudzone zarządzanie ma swe źródło w koncepcji odchudzonej produkcji (ang. Lean production), która została wymyślona i po raz pierwszy zastosowana w japońskim koncernie samochodowym Toyota, przez szefa produkcji tego koncernu Taiichi Ohno. Ta bardzo popularna koncepcja znalazła swoje zastosowanie także przy produkcji oprogramowania, znana jest pod nazwą Lean Software Development, a została zaadoptowana przez Mary Poppendieck i Toma Poppendiecka. W szczupłym wytwarzaniu oprogramowania wyróżnia się 7 zasad wspomaganych przez 22 narzędzia.

- Eliminacja strat (Eliminate Waste)
- Tworzenie jakości i spójności (Build Quality In)
- Wzmocnienie pozyskiwania wiedzy (Create Knowledge)
- Podejmowanie decyzji najpóźniej, jak to możliwe (Defer Commitment)
- Wdrażanie najwcześniej, jak to możliwe (Deliver Fast)
- Respektowanie zespołu (Respect People)
- Spojrzenie na całość (Optimize the Whole)

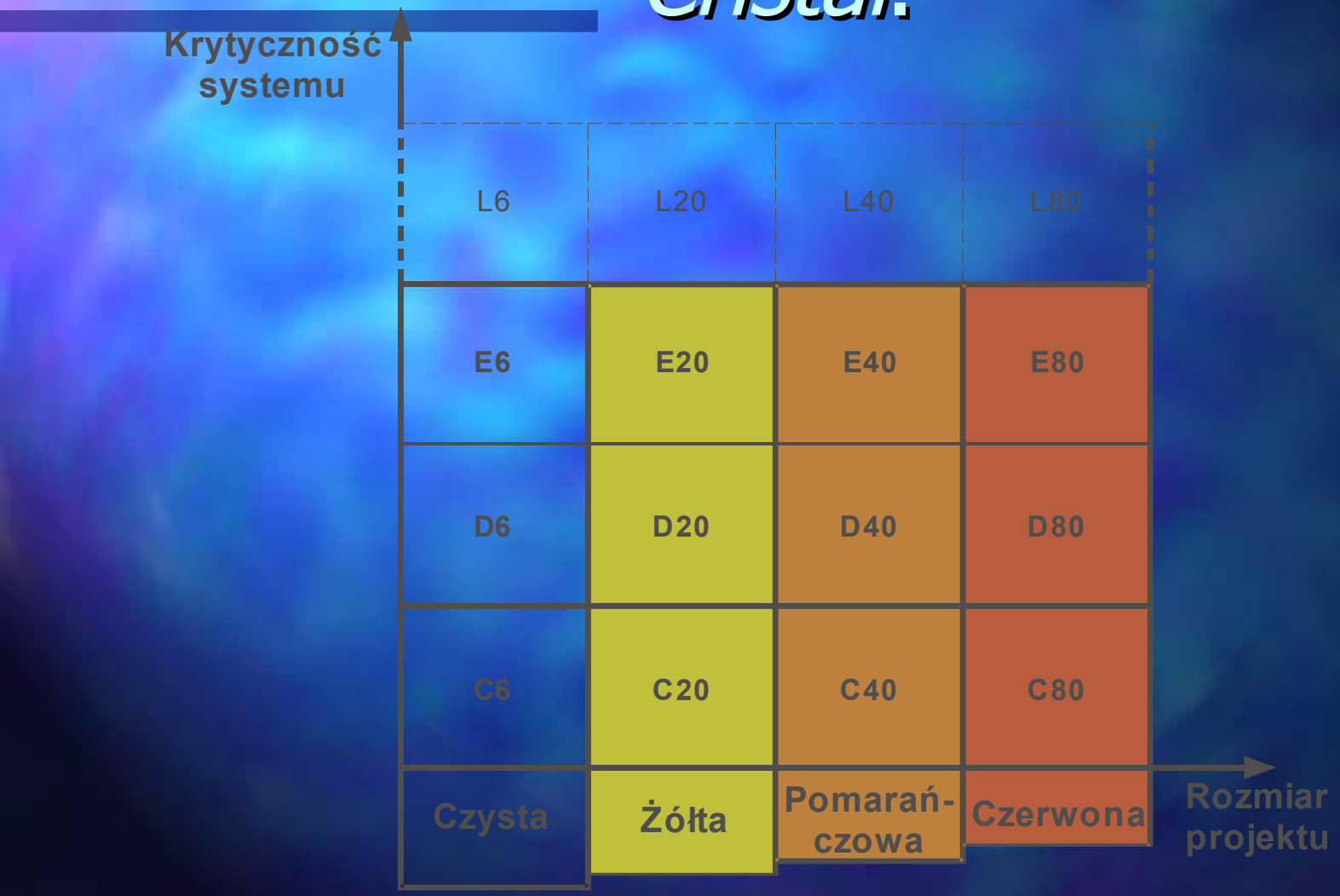
Crystal

- Metodyka *Cristal* ma odmiany w zależności od stopnia **krytyczności** projektu (kategoryzowanego poprzez litery **C**, **D**, **E**, **L** – objaśnione dalej) i w zależności od jego **rozmiaru**, mierzonego liczbą projektantów zaangażowanych w tworzenie projektu.

Kategorie krytyczności projektowanego systemu:

- **C** - **K**omfortowe (Comfort),
- **D** - **Z**arządzające **F**inansami (Discretionary Money),
- **E** - **F**inansowo **i**stotne (Essential Money)
- **L** - **K**rytyczne dla **Ż**ycia (Life Critical)

Na tej podstawie proponowana jest cała rodzina metod typu *Cristal*.



Feature Driven Development

- Jej głównymi celami jest umożliwienie wytwarzania użytecznego oprogramowania w powtarzalny i efektywny sposób, zapewniając wiarygodne informacje o stanie projektu informatycznego do wszystkich jego uczestników, z minimalnym narzutem na pracę programistyczną.

Założenia

- Zapewnia dostateczną strukturę dla prac większych zespołów
- Nacisk na jakość wytwarzanego oprogramowania
- Kolejne wersje oprogramowania powstają często i zawierają nowe użyteczne funkcje
- Zapewnia mechanizmy do wiarygodnego śledzenia postępu prac
- W FDD są używane testy jednostkowe
- FDD zakłada przypisanie kodu (klas) do właścicieli (programistów)
- Podczas implementacji wykonywane są inspekcje kodu

Fazy projektu:

- Budowa ogólnego modelu,
- Budowa listy cech,
- Planowanie według cech,
- Projekt według cech i Implementacja według cech,

dwie ostatnie są powtarzane wielokrotnie podczas projektu.

Inne lekkie metodyki to:

- Dynamic Systems Development Method,
- Adaptive Software Development,
- Pragmatic Programming,

KUNIEC :)