9. rész: Korszerű Implementációs Technikák

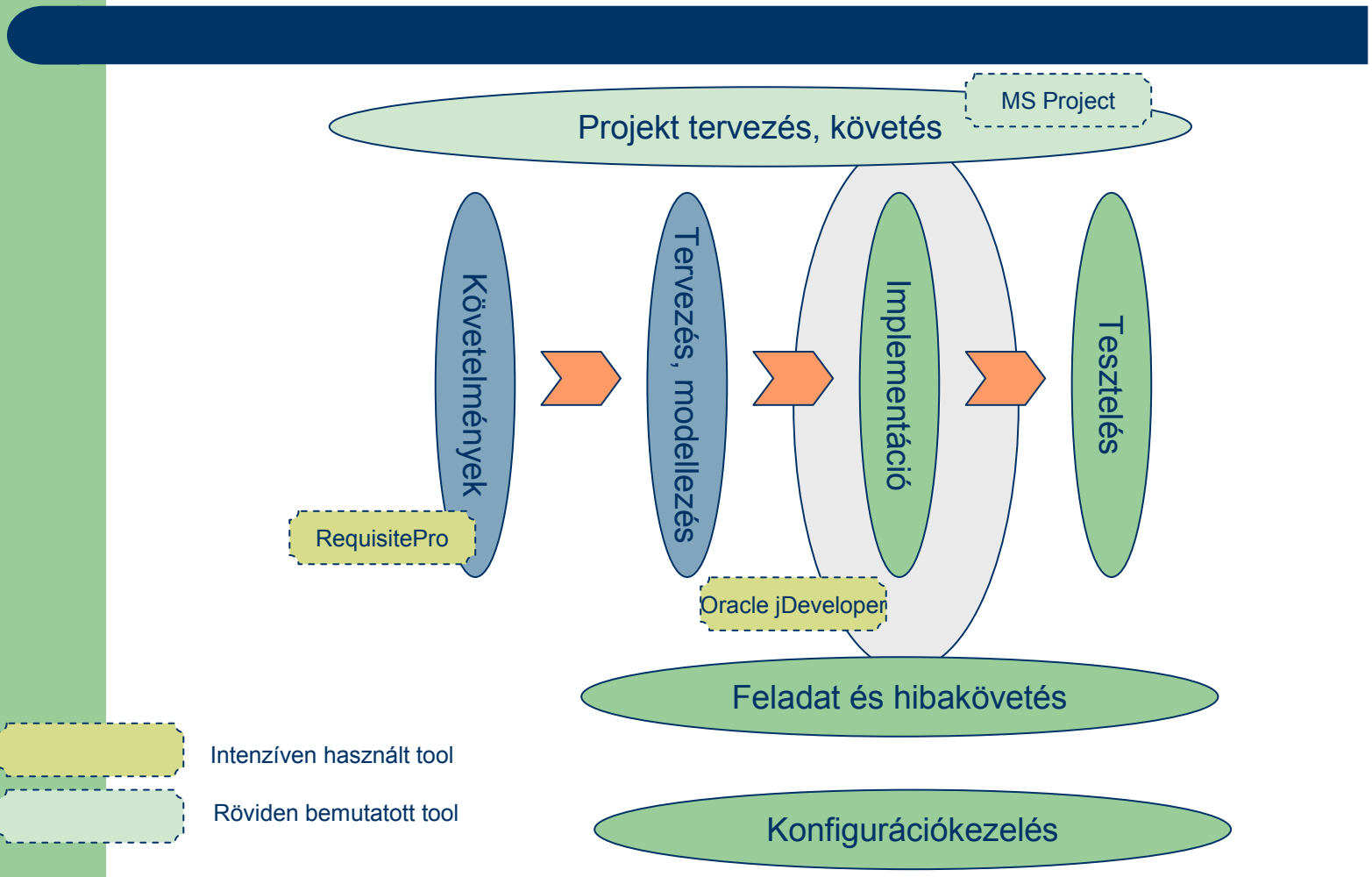
Bakay Árpád dr.
NETvisor kft
(30) 385 1711
arpad.bakay@netvisor.hu

A tananyag készült az
ELTE-IKKK
projekt támogatásával

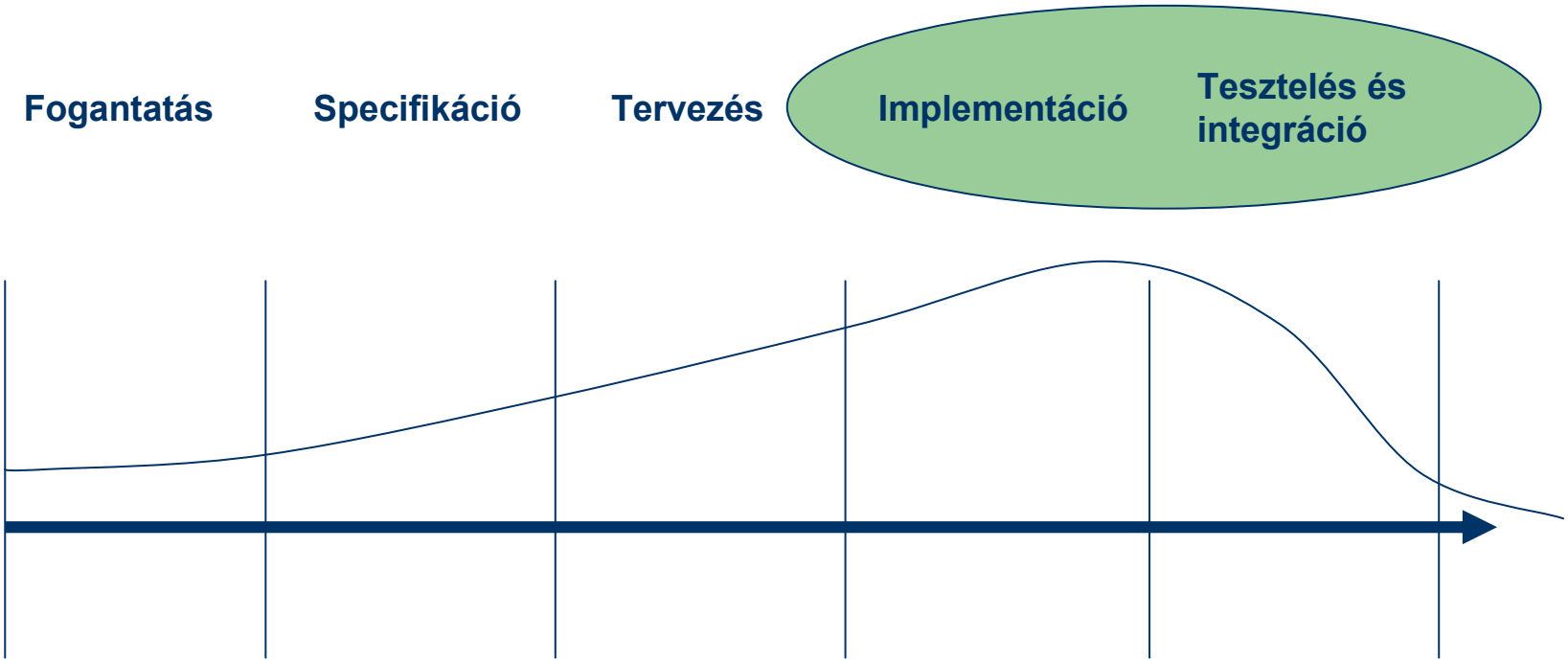
Mi kéne még (2007/04/24)

- JMeter
- (Terracotta - APP server clustering)
- Apache Derby database
- https://glassfish.dev.java.net/javaee5/persistence/persistence-example.html
- Query.setparameter

Hol tartunk?


Projekt tervezés, követés
MS Project
Követelmények
RequisitePro
Tervezés, modellezés
Implementáció
Oracle jDeveloper
Tesztelés
Feladat és hibakövetés
Konfigurációkezelés
Intenzíven használt tool
Röviden bemutatott tool

Az munka utolsó fázisai



A mai anyag:

- Házi feladat 2. fázis ismertetése
- Fejlesztői környezetek és debuggerek
- Kódolási konvenciók
- Kódolvasás, véleményezés

1. Házi feladat 2. fázis

- Tudor/Szeráj stb. EJB réteg elkészítése a JDeveloperrel
  - A web-en található Ursula mintapélda alapján
  - Adatmodellezés
  - 3-4 Entity készítése
  - 1-2 Stateless Session EJB (Facade)
  - Egyszerű teszt kliens osztály

2. Fejlesztői környezetek

- IDE: Integrated Development Environment

  „Programers' workbench"
- Cél növelni a
  - Hatékonyságot (line, funkció/nap)
  - Pontosságot (frequent errors and bugs)
  - Tiszta kódot (coding style and documentation)
  - Csoportmunkát (source control)
  - Dokumentáltságot

IDE Szokásos Részei

- Komponensek és források menedzselése
- Kód editor
- Navigációs és munkagyorsító technikák
  - plug-inek és wizardok
- Build manager
- Debugger
- Verziókezelés integráció

2/a Komponensek és források menedzselése

- „Workspace”
  - egy egész (akár nem is összetartozó) szoftver-rendszer összes komponense
- „Project” -> egyetlen építendő komponens
  - jar, war, ear (vagy .exe, .dll, .lib stb.)
  - forrás-fileok a projektekhez rendelve
  - project settings: library-k, directory-k, compile & run options
- Projektek közti függőségek, fordítási sorrend
- Futtatási konfigurációk
  - építés és futtatás többféle lehetséges változatban
    - Debug / release opciók
    - Profiler
    - (C++: Különféle optimalizációk, futtató op.rsz. verziók, stb.)

2/b Kód editor

- Szintaxis ellenőrzés és kiemelés
- Célirányos, környezet-érzékeny help
- Automatikus kiegészítés (autocomplete)
- Keresztreferenciák
- Editor műveletek, blokkok, keres-cserél stb.
- Beautifier: kód rendezés, tördelés

- Speciális resource editor interfészek
  - Grafikus UI (dialog box) editor
  - HTML editor
  - Modellezési funkciók

2/c Navigáció

- Osztály-nézet, structure view
- Keresztreferenciák
  - Show declaration
  - Find usages
- Keresés
  - Sok fileban v. reguláris kifejezés szerint
  - Hibákra, todo-kra, stb.
  - Kiemelés
  - Keres-cserél (-> refactoring)
- Context-sensitive help system
  - Az IDE-re, és a használt nyelvre is!

2/d Félautomatikus munkafolyamatok, „wizard”-ok

- Projekt, osztály, metódus stb. készítése
- Tömeges változtatás: Refactoring
- Integráció más programokkal
  - pl. Modellező, feladatkezelő, CVS stb.
- További toolok,
  - pl. data modelling, server admin, etc.

- Saját wizardok készíthetők
  - Plugin/addon API
  - Pl. dokumentáció készítés, report generálás

2/e Build manager

- Automatikus build
  - Lehetőleg csak a változott részekre
- Cél a fordítás gyorsítása gyorsaság
  - Pl. C++:
    - Egymenetes fordítás
    - Előfordított headerek
    - Inkrementális linkelés
    - Runtime update
- Szakosodott build eszközök speciális fordítási feladatokra:
  - Command-line build tools: Ant , make
  - Code preprocessor/precompiler
    - XDoclet, AspectJ
  - EJB compiler
  - XML eszközök
- Hibák jelzése, megjelenítése
  - Lehetőleg már editálás közben is!!
  - Quick fix: egyszerűbb hibák automatikus javítása (a user felügyeletével!)

2/f Debugger

- Lépésenkénti végrehajtás
- Breakpointok / feltételes breakpointok
- Data watch/edit
- C/C++:
  - Assembly view
  - Spec. nézetek: stack, memória, regiszterek
- Hibák detektálása
  - Memory leak, (szivárgás)
  - Nem kezelt kivételek

… debugger folyt

- A debugger használata alapvető a C/C++ világban

de

- Korszerű nyelveknél és fejlesztési módszereknél némileg csökkent debugging a jelentősége.Ennek okai:
  - Elosztott rendszerek
  - Automatikus memória és referencia-kezelés
  - Testsuite-ok
  - Logging frameworkok

- További eszközök:
  - Teljesítmény analízis: Profiler
  - Rendszerhívások analízise

JDeveloper CodeCoach

- Egy speciális Java Virtual Machine kiterjesztést használva, path elemzési és statisztikai alapon tanácsokat ad a futó Java kódról
  - Collection-ok kihasználtsága
  - Felesleges kód, pl.: `new String("abc");`
- Pragma commentekkel megszelidíthető:
  - `//@codecoach:disable NVCT,LALL`

Köszönöm a figyelmet!

- arpad.bakay@netvisor.hu