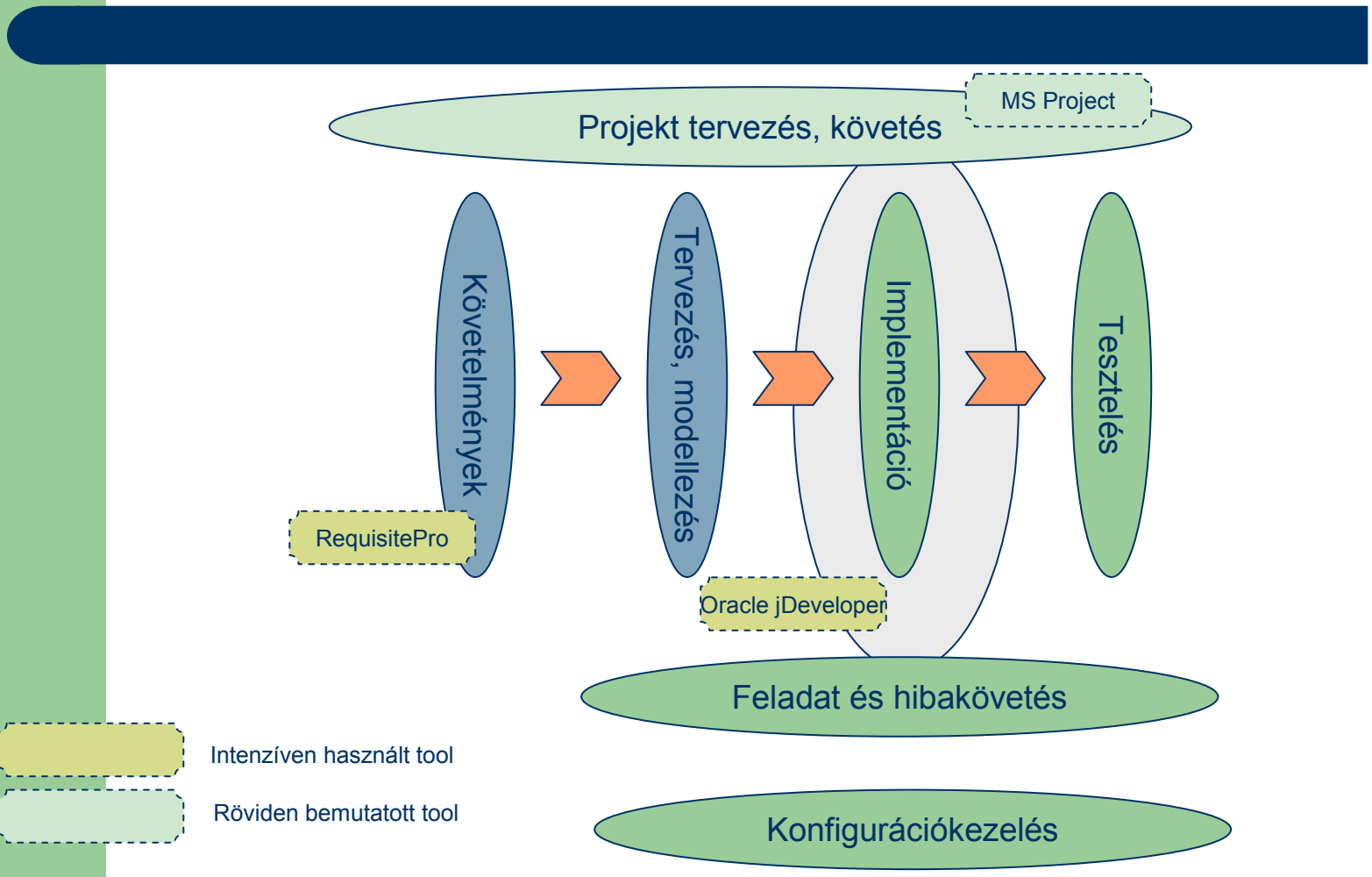
9. rész: Korszerű Implementációs Technikák

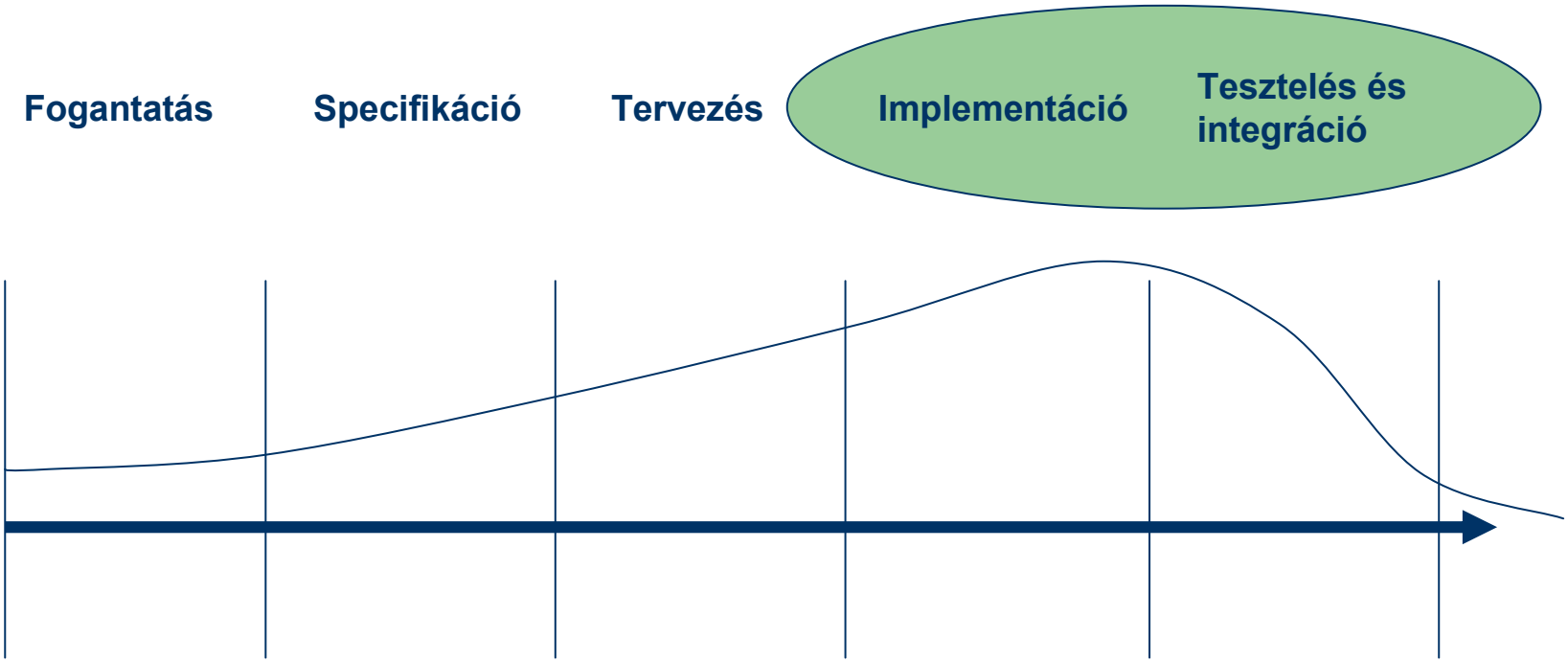
Bakay Árpád dr.
NETvisor kft
(30) 385 1711
arpad.bakay@netvisor.hu

A tananyag készült az
ELTE-IKKK
projekt támogatásával

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)
- Editor műveletek, blokkok, keres-cserél stb.
  - Becsuk/kinyit: metódusokra / import deklarációkra
- Formatter/beautifier: kód rendezés, tördelés
- Kód template-ek

- Nem szokásos forrásokhoz: 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
  - Magára az IDE-re, és a fejlesztett nyelvre is!
    - JDeveloper extra: Quick Javadoc - csak a paraméterek, exceptionok

2/d Félautomatikus munkafolyamatok, „wizard”-ok

- Projekt, osztály, metódus stb. készítése
- Tömeges változtatás: refactoring
  - Rename/move
  - Extract interface, move members up/down, use supertype
  - Encapsulate fields (getter/setter method)
- Integráció más programokkal
    - pl. Modellező, feladatkezelő, verziókezelő stb.
- További toolok,
    - Láttunk ilyent: pl. data modelling, server admin, etc.

- Saját wizardok készíthetők
    - Plugin/addon API
    - Pl. dokumentáció készítés, riport 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 (Java) , make (C/C++)
  - Code preprocessor/precompiler
    - XDoclet, AspectJ
  - Generáló eszközök
    - Lexikai + szintax analízis: Lex/Yacc (C/C++Ö), JAX/JAY (Java)
    - XML eszközök (pl. XSLT)
- 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 debuggolása nehézkes
    - *Azért tudunk majd ilyet is!*
  - Automatikus memória és referencia-kezelés
  - Testsuite-ok
  - Logging frameworkok

2/g Futtatás közbeni analízis

- Teljesítmény analízis
  - Profiler (JProf, JProfiler, JMP, JMemProf)
  - Vagy: Ctrl-Break a JVM-nek
- Hálózati kommunikáció, rendszerhívások
  - TCPdump, Truss / Strace (C/C++)
- Új eszköz: JDeveloper CodeCoach

JDeveloper CodeCoach

- Egy speciális Java Virtual Machine verziót (OJVM) 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`

3. Kódolási és dokumentációs konvenciók/szabványok

- Miért kell?
  - Logikai egységesség → Karbantarthatóság
  - Dokumentáció → Használhatóság
  - Gyakran vétett hibák elkerülése

3/a. Kódolási konvenciók

- Standard Java C. C. http://java.sun.com/docs/codeconv/
  - Vagy egy cég-specifikus változat
- Témakörök
  - File nevezés, könyvtárstruktúra
    - (metódusok logikai csoportosítása)
  - File-on belüli szintaktikai elemek sorrendje, elkülönítése
  - Sorhossz és tördelés
    - Hosszú kifejezések
    - Utasítások (if, for, switch, try-catch, stb.), zárójelezés

… folyt

- Kommentezés
- Névadási konvenciók
  - Változó divatok, pl.:
    - Régebben a Microsoftnál: „Hungarian notation”:
      psz*TitleStringPointer*: „pointer to a zero-terminated string”
    - „Camel notation”: titleStringPointer
- Olvasható kód-logika
    - Deklarációk
    - Javasolt metódus méret-korlátok
    - Problémás, zavaros megoldások kerülése
      - „switch” fall-through („átejtés”)
      - return, break veszélyei

3/b. Kommentezés - fajtái

- Dokumentációs -- a modul használói számára
  - Blokk komment
  - Kiemelhető a kódból → Kész az API specifikáció!!!
    - Pl. JavaDoc-kal dokumentáció generálható
    - Szabad formátumú ismertető és standard tag-ek

„A bőséges dokumentációs komment az igényes fejlesztés tükre'

- Implementációs – a továbbfejlesztők számára
  - Blokk-komment, egysoros: beszúrt, vagy trailing v, sorvégi

„Sok implementációs komment a kód szegénységét tükrözi"

3/c. JavaDoc

- Kódba ágyazott megfelelő dokumentációs kommentekből API dokumentáció generálható
- Standard dokumentációs struktúra standard tag-ek alapján:
  - ***@param [argument name] [argument description]*** - describes an argument of method or constructor.
  - ***@return [description of return]*** - describes data returned by method (unnecessary for constructors and void methods).
  - ***@throws [exception thrown] [exception description]*** - describes exception thrown by method.
  - ***@author [author name]*** - identifies author(s) of a class or interface.
  - ***@version [version]*** - version info of a class or interface.
- Generálás a javadoc pranccsal
- http://java.sun.com/j2se/javadoc/index.html

Köszönöm a figyelmet!

- arpad.bakay@netvisor.hu