Praktické aspekty vývoje software

Jaroslav Dytrych

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole
dytrych@fit.vut.cz

11. února 2025

Obsah přednášky

- Úvod do předmětu
- Praktická pravidla pro psaní udržovatelného kódu
- Refaktorizace
- Efektivní využití IDE (Integrated Development Environment)

Motivace

- V rámci předmětů bakalářského programu budete zpracovávat řadu samostatných i týmových projektů.
- V praxi se můžete stát členy malého týmu či pracovat zcela samostatně (např. tvorba webů).
- Lze vystačit i bez znalostí IVS (v r. 2011 první běh předmětu).
- Ale nástroje a postupy prezentované v IVS Vám mohou pomoci zefektivnit týmovou spolupráci, usnadnit Vám jednotlivé kroky při vývoji SW (od analýzy požadavků až po doručení zákazníkovi) a podpořit vytváření systémů udržovatelných po delší době, nikoliv jen v měsíci, kdy vznikly.

Motivace (2)

- Jinou možností je, že se v rámci nového zaměstnání setkáte s vágně pojatým procesem vývoje software, volnými či neexistujícími normami a změtí nekompatibilních nástrojů.
- Předmět IVS by vás měl naučit, které nastroje a postupy se vyplatí zavést a které nikoliv.
- Je také možné, že se v praxi setkáte s dobře definovaným procesem vývoje, normami a sdílenou sadou nástrojů. V takovém prostředí musíte být schopní si rychle osvojit využité nástroje a danou koncepci vývoje.
- V IVS byste měli získat zkušenosti s typickými postupy, což by mělo pomoci ke snazšímu pochopení specifik nových nástrojů.

Co se v předmětu nedozvíte

- Jak se naučit programovat v konkrétním programovacím jazyce / používat vývojové nastroje specifické pro konkrétní jazyk (viz např. předmět ISJ).
- Teoretické základy, na nichž stojí různé nástroje (např. pro odhalování chyb) – na to jsou jiné specializované předměty.
- Vyčerpávající přehled dostupných nástrojů, s nimiž se lze v praxi setkat.
- Vyčerpávající popis možností určitého nástroje – ten lze dohledat v dokumentaci.

- Řízení projektu (MS Project, ProjectLibre, Notion, Trello, Freelo, Jira, ClickUp, . . .)
- Testování (GoogleTest, JUnit, GCOV, . . .)
- Správa verzí (GIT, Mercurial, Bazaar, . . .)
- Generování dokumentace (Doxygen, Javadoc, phpDocumentor, . . .)
- Sestavování částí (Make, CMake, QMake, . . .)
- Odhalování chyb (debugger)
- Evidování chyb a námětů (Redmine, Trac, Bugzilla, . . .)
- Profiling (GProf, cProfile, JProfiler, . . .)
- Integrace (Jenkins)
- Nasazení (Windows Installer, Debian Package Manager, ...)
- . . .

Přehled přednášek

1. Úvod, praktická pravidla pro psaní udržovatelného kódu, refaktorizace a efektivní využití IDE
2. Testování software, TDD (Test-Driven Development) a jeho použití při vývoji v týmu
3. Týmová spolupráce, komunikace, sdílení dat v týmu, základy návrhu a plánování projektu
4. Systémy pro distribuovanou správu verzí, GIT
5. Typy dokumentace, generování programové dokumentace z kódu, identifikace existujících komponent a využívání knihoven dostupných na různých platformách
6. Sestavení programů, Make, Cmake a Qmake
7. Uživatelská rozhraní
8. Debugging, bugtracking a QA
9. Demo na pokročilý GIT (Red Hat)
10. Nasazení programů a půlsemestrální zkouška
11. Kontejnerování aplikací
12. Marek Milkovič (Gen Digital): Programovací jazyky a paradigmata, FFI (Foreign Function Interface)
13. Petr Zemek (Gen Digital) / Paralelizace a profiling

- Z předchozích let máme záznamy – např.:
  - Tvorba grafického uživatelského rozhraní
  - GIT (vylepšený skript viz `sftp://ivs.fit.vutbr.cz:22/ivs/git.sh`)
- Letos bude stejně jako loni od firmy Red Hat na téma:
  - Pokročilý GIT (Interactive rebase, git bisect apod.)
    - Co se v základech neučí, ale ve firmách používá.
  - Dříve zpětná vazba studentů, že cvičení bylo příliš krátké a rychlé, pak že bylo příliš pozdě na využití v projektech
    - Loni a předloni bylo prodloužené na 3 hod. a mělo úspěch, letos bude také
    - Kvůli absenci místa v rozvrhu **bude rozdělené na 2 části, přičemž druhá bude v pátek**

Návštěva odborníka z praxe

- Letos si přednášku na Programovací jazyky a paradigmata, FFI (Foreign Function Interface) vezme Marek Milkovič, Engineering Manager z firmy Gen Digital
- Na poslední přednášku se pokusím domluvit Petra Zemka (Lead Software Engineer at Gen Digital)
- Na přednášky z předchozích let se můžete podívat ze záznamů.

- 1. projekt: Definice testů (18 bodů)
  - Zadání na 2. přednášce.
  - Individuální – vypracuje každý sám (**kontrola plagiátorství**).
  - **Pozor na zveřejnění řešení** (Facebook, Pastebin, Github)!
  - **Pozor na prohlížení zveřejněných řešení** (podvědomě napodobíte – málo kdo vymyslí jiné řešení, když před prohlédnutím toho cizího nevěděl, jak na to).
  - **Pozor na využití umělé inteligence** (v podstatě jen kombinuje existující řešení – na kom asi trénovala projekty z IVS? – na studentech, co to loni a předloni dali na GitHub!).
    - V případě, že využijete umělou inteligenci, nejste zbaveni povinnosti vytvořenému řešení rozumět.
  - Kdo bude podezřelý z plagiátorství či využijí umělé inteligence, bude pozván na obhajoby.
    - Při podezření z plagiátorství povede neomluvená absence u obhajoby k předání disciplinární komisi.
    - V případě využití umělé inteligence je nutno prokázat, že odevzdanému řešení rozumíte – **za část řešení, které nerozumíte, dostanete 0 bodů**.
    - Chcete-li zabránit chybnému vyhodnocení využití umělé inteligence jako plagiátu, doporučujeme v komentáři explicitně uvést, jak a kterou jste využili.

- 2. Projekt zaměřený na spolupráci v týmu (52 bodů)
  - Zadání oficiálně na 3. přednášce (shlédněte před započetím práce – alespoň záznam z loňska).
  - Průběžná týmová práce (odevzdává se postupně, pozor na **termíny – jsou v zadání a nejsou o půlnoci!**).
  - Instalace (předvedení) v posledním týdnu výuky, případně i v 1. týdnu zkouškového období.
  - Veřejné obhajoby v první polovině zkouškového období.
  - Z obhajob je pořizován záznam a veřejně zpřístupněn.
  - V závěru dostanete detailní zpětnou vazbu (ne jen čísla).
  - **Není o programování, ale o procesu vývoje!**
    - Když celý projekt naprogramuje 1 student, je to na nic.
    - Soustřeďte se na zadání – za funkční produkt je jen 1/2 bodů.
- Půlsemestrální písemná zkouška (30 bodů).
  - Mimo přednášku, v pondělí 13. 4. 2026 **18:00-19:50**
- Úkoly k procvičení (bez bodů, dle možností zpětná vazba).

- Bonusové body
  - Za půlsemestrální test lze získat až 36 bodů s tím, že na max. 30 se hodnocení ořízne. Nadbytečné body mohou být (částečně) přidány do bonusových bodů.
  - Na přednášce či demonstračním cvičení hosta z praxe může být (při malé účasti) prezence za 1 bod.
  - Po obhajobě 2. projektu se lze přihlásit na úkol za max. 3 bonusové body v IS a napsat si e-mailem o zadání zaměřené na Vaše chyby/neznalosti z předchozích aktivit v předmětu. Pracnost lze očekávat v rozsahu cca 5–10 hod./1 bod.

Server ivs.fit.vutbr.cz

- Server ivs.fit.vutbr.cz je určen pro řešení projektů.
- Připojit se lze přes ssh z IP adres v doménách .cz a .sk (vyžaduje funkční reverzní DNS – ověřit lze např. tak, že na `https://www.whatismyip.com/` zjistíte svoji IP a pak ověříte, že má reverzní záznam, na `https://www.whatismyip.com/reverse-dns-lookup/`).
- Login je stejný jako na ostatní servery FIT, heslo Vám bude zasláno e-mailem nejpozději koncem 2. týdne semestru.
- Chovejte se na něm prosím slušně a neomezujte své kolegy jeho přetěžováním.
- Pokud se nemůžete připojit, zašlete mi svoji IP adresu e-mailem na dytrych@fit.vut.cz (**posílejte prosím veřejnou a nikoliv lokální IP**). Pokud se Vám adresa často mění, pošlete název a rozsah svého poskytovatele připojení (můžete hledat např. na `https://www.nirsoft.net/countryip/`).

Praktická pravidla pro psaní udržovatelného kódu

- Unix vznikl v roce 1969 a od té doby je stále v provozu. Počítače se systémem Unix mají za sebou již více než stovky tisíc provozních hodin.
- Programátoři v systému Unix mají za sebou desítky let zkušeností. To, co museli v historii řešit, my dnes pokládáme za samozřejmé.
- Porozumění základním konceptům je užitečné, abychom „znovu neobjevovali kolo“ a byli schopni vytvářet SW, který bude rovněž moci přečkat desítky let.

Unixová filosofie na 1 slajdu

- Navrhujte programy odspodu nahoru a snažte se, aby byly co nejjednodušší.
- Vytvářejte jednoduché nástroje, které vzájemnou spoluprací dokáží plnit složité úkoly (program by měl dělat jednu věc a to opravdu dobře).
- Počítejte s tím, že výstup každého programu může být využit jako vstup jiného programu (i u GUI – např. kopírovat do schránky).
- Předpokládejte určitou inteligenci uživatele (vyhněte se zbytečným dotazům, dialogům apod.).
- Při procedurálním programování pište malé funkce pro řešení specifických problémů.
- Při objektově orientovaném programování by objekt měl zastřešovat právě jednu entitu a to úplně. Obdobně by jedna metoda měla provádět jednu činnost.
- Navrhujte SW tak, aby mohl být rychle prototypován a zkoušen (ideálně v řádu týdnů).
- ...

- Pravidlo srozumitelnosti (srozumitelnost je lepší než chytrost)

Správně

```
int Sum = 0;
for (int i = 1; i < N; ++i) {
  Sum = Sum ^ i;
}
```

Špatně

```
int Sum = (N & (N % 2 ? 0 : ~0) |
(((N & 2)>>1) ^ (N & 1))) ^ N;
```

- Pravidlo kompozice (navrhujte programy tak, aby je bylo možné připojit k dalším programům)

Správně

```
int c;
while ((c = getc(stdin)) != EOF) {
  putchar(toupper(c));
}
```

Špatně – zbytečně ze souboru a ještě fixní název

```
int c;
FILE *f;
f = fopen("input.txt", "r");
while ((c = getc(file)) != EOF) {
  putchar(toupper(c));
}
fclose(f);
```

- Pravidlo transparentnosti (navrhujte programy tak, aby bylo co nejsnazší jejich ladění a kontrola)

Správně

```
int maxCount = 50;
int sum = 0;
for (int i = 1; i < maxCount && buf[i] > 0; i++) {
  sum = sum + buf[i];
}
```

Špatně

```
int sum = 0;
for (int i = 1; i < 50 && buf[i] > 0; i++)
  sum = sum + buf[i];
```

(„magické číslo" + když přidáte řádek (např. s ladícím výpisem), nesmíte zapomenout doplnit závorky)

- Pravidlo robustnosti (robustnost je dítětem transparentnosti a jednoduchosti)

Správně

```
char *line = NULL;
size_t len = 0;
printf("Enter a value:");
read = getline(&line, &len, stdin);
...
free(buff)
```

Špatně (takto se tvoří chyby „buffer overflow“)

```
char str[100];
printf("Enter a value:");
gets(str);
```

- Pravidlo reprezentace (zahalte znalosti do dat tak, aby logika programu mohla byt hloupá a robustní)

Správně

```
char* messages[7];
strings[0] = "you typed zero";
strings[1] = "n is a perfect square";
strings[2] = "n is an even number";
strings[3] = "n is a prime number";
strings[4] = "n is a perfect square";
...
printf("%s\n",strings[n]);
```

Špatně

```
if (n == 0) printf("you typed zero");
else if (n == 1 || n == 4 || n == 9)
  printf("n is a perfect square");
else if (n == 2 || n == 6 || n == 8)
  printf("n is an even number");
else if (n == 3 || n == 5 || n == 7)
  print "n is a prime number";
```

- Pravidlo nejmenšího překvapení (při návrhu rozhraní se snažte, aby dělalo vždy co nejméně překvapující věci)

Správně

```
int multiply(int a, int b) {
  return a * b;
}
```

Špatně

```
int multiply(int a, int b) {
  return a + b;
}
```

- Pravidlo hospodárnosti (programátorův čas je drahý, šetřete svůj čas na úkor času počítače)

Program v C

```
int wordCount(char *line) {
  int count = 0;
  while(isspace(*line)) line++;
  int len = strlen(line);
  bool onWord = true;
  for(int i = 0; i < len; i++) {
    if(isspace(line[i])) {
      if(onWord) {
        count++;
      }
      onWord = false;
// ... ještě 10 řádků
```

Program v Pythonu

```
def wordCount(line):
  words = line.strip().split()
  return len(words)
```

- Pravidlo optimalizace (nejprve vytvořte prototyp, zprovozněte, a pak se teprve pusťte do jeho optimalizace)

Správně

```
float f[100],sum;
...
sum = 0;
for(i=0;i<100;i++) {
  sum += f[i]; }
```

Špatně

```
float f[100],sum0,sum1,sum2,sum3,sum;
...
sum0 = sum1 = sum2 = sum3 = 0;
for(i=0; i<100; i+=4) {
  sum0 += f[i];
  sum1 += f[i+1];
  sum2 += f[i+2];
  sum3 += f[i+3];
}
sum = (sum0 + sum1) + (sum2 + sum3);
```

- Pravidlo rozmanitosti
  - Nedůvěřujte všem tvrzením typu „existuje jediný správný způsob“.
- Pravidlo rozšiřitelnosti
  - Navrhujte pro budoucnost, protože ta nastane dříve, než si myslíte.
- ...

- Máte-li dobrý důvod, klidně kterékoliv pravidlo porušte, ale:
  - nejprve je třeba pochopit, proč dané pravidlo máme a jaké bude mít jeho nedodržení důsledky,
  - udržujte kód konzistentní a přehledný,
  - vždy myslete na ty, co po Vás budou pokračovat v práci.
- Příkladem může být optimalizační „finta", která upřednostní chytrost a rychlost před srozumitelností – pak je ale nutné dobře komentovat, případně uvést neoptimální kód jako komentář.

Formátování kódu

- Formátování musí být takové, aby byl kód co nejpřehlednější.
  - Kód bez komentářů je pro ostatní často nečitelný (např. regulární výrazy jsou tzv. „write-only“).
  - Kódování znaků a konce řádků musí být jednotné.
  - Odsazování musí být důsledné.
  - Odsazování o hodně znaků je nepřehledné (je nutné skrolovat vertikálně i horizontálně). Časté jsou 2 nebo 4 mezery nebo 1 tabulátor.
  - I volné řádky mají svůj význam (za funkcí, mezi bloky, . . .).
  - . . .
- Je vhodné využívat obvyklé vzory a postupy, které jsou často podpořené v IDE.
- Formátování musí být v celém programu jednotné.
- Při vývoji v týmu je na začátku potřeba stanovit pravidla.
- S dodržováním jednotného nastavení v rámci týmů a projektů Vám může pomoci EditorConfig `http://editorconfig.org/`

- Konfigurační soubor pro přizpůsobení nastavení editoru (`.editorconfig`).
- Některá IDE a editory jej přímo podporují, pro ostatní je potřeba zásuvný modul (Code::Blocks, NetBeans, Eclipse, ...).
- Nejdůležitější jsou kódování, konce řádků a odsazování.

Ukázka

```
root = true
[*]
charset = utf-8
end_of_line = lf
insert_final_newline = true
[Makefile]
indent_style = tab
[*.{c,cpp,h}]
indent_style = space
indent_size = 2
```

Použitelnost programů

- Pravidlo mlčenlivosti (nemá-li program co překvapivého říci, neměl by říkat nic)

Správně

```
$ ls empty_dir
$
```

Špatně

```
$ ls empty_dir
$ Vítejte v programu ls. Vámi požadovaný
adresář je prázdný, chcete vypsat obsah
jiného adresáře? [Y|N]: N
$
```

- Pravidlo opravy (pokud se program musí zhroutit, ať to udělá hlasitě a co nejdříve)

Správně

```
$ wget www.google,com
Překládám www.google,com (www.google,com)...
nezdařilo se: Neznámé jméno nebo služba.
wget: adresu počítače www.google,com nelze přeložit
$
```

Špatně

```
$ wget www.google,com
Překládám www.google,com (www.google,com)...
nezdařilo se: Neznámé jméno nebo služba.
Zkouším znovu...
Překládám www.google,com (www.google,com)...
nezdařilo se: Neznámé jméno nebo služba.
Zkouším znovu...
...
```

- Nezavádějte zbytečně mnoho vyžadovaných pozičních parametrů.
- Nenabízejte konfigurační přepínače proto, co může být spolehlivým způsobem zjistitelné automaticky (např. autodetekce formátu vstupního souboru).
- Uživatelé by neměli mít přístup k optimalizačním přepínačům.
- Nesnažte se přidat přepínačem to, čehož by se dalo dosáhnout zapouzdřením programu do skriptu nebo jeho začleněním do zřetězení (např. přepínač pro stránkování výstupu).
- Zamyslete se nad použitím:

```
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... -t DIRECTORY SOURCE...
```

```
find /ivs/test/ -type f | head -5 | xargs cp -t /ivs/t2
```

Pojmenovaní distribučních souborů

1. předpona projektu
2. spojovník nebo podtržítko
3. číslo verze
4. tečka
5. src nebo bin (nepovinné)
6. tečka, spojovník nebo podtržítko
7. binární typ a možnosti (nepovinné)
8. přípona archivu nebo kompresního formátu

```
lcov-2.4-0.noarch.rpm
lcov-2.4-0.src.rpm
git_2.51.0.orig.tar.xz
git_2.51.0-1ubuntu1_amd64.deb
mercurial-3.2.3.win-amd64-py2.7.exe
mercurial-7.0.1-x64.msi
```

- Raymond, E.S.: *The art of unix programming*, ISBN-0131429019, Pearson Education, 2003
- Balzarotti, D.: *Software Development The UNIX Philosophy*, Eurecom – Sophia Antipolis, France, 2015
- Cantin, C.: *Basic Introduction to UNIX/linux*, 2010
- Lande, J.: *Unix Tutorial*, 2010

Refaktorizace

Refaktorizace

- Každá část programu musí mít jednu jednoznačnou a spolehlivou podobu.
- Opakovaní vede k nekonzistencím – kdykoliv spatříte duplicitní kód, je to známka nebezpečí.
- Refaktorizace je proces zabývající se vnitřním preuspořádáním kódu, aniž by se změnila jeho funkčnost. Zahrnuje především:
  - odstraňování duplicit,
  - přejmenování částí kódu (pro sjednocení, zpřehlednění apod.),
  - zpřehlednění (např. rozdělením na menší části).

Cíle refaktorizace

- Zlepšit interní kvalitu aplikace.
- Lepší škálovatelnost.
- Vylepšení čitelnosti a udržovatelnosti kódu.
- Zjednodušení struktury kódu.

Základní techniky refaktorizace

- Zjednodušení podmínek cyklů a podmíněných výrazů.
- Úpravy řetězců (kódování, překlad).
- Přesunutí opakujícího se kódu do nové funkce.
- Zkracování funkcí vytvářením nových
  - Kód dlouhé funkce přesuneme do více krátkých, které z dané funkce voláme.
  - Ve funkci na nejvyšší úrovni je dobře vidět posloupnost operací.
  - Funkce na nižší úrovni dělají právě jednu činnost – dojde-li k chybě, je jasnější, co se zrovna provádělo.
- Vhodná přejmenování funkcí a proměnných.
- Vytknutí společných vlastností do třídy, ze které se bude dědit (tedy např. třídy Malíř a Sochař nebudou mít duplicitní vlastnosti a metody, ale vše společné bude vytknuto do třídy Umělec, ze které budou dědit).
- ...

- Nikdy neměňte funkcionalitu i strukturu současně.
- Refaktorizujte průběžně a často.
- Neponechávejte původní kód (nekopírujte ale přesouvejte) – pro vrácení zpět využijete GIT.
- Po refaktorizaci vždy otestujte, zda nedošlo ke změně chování programu.
  - Před refaktorizací je vhodné napsat a spustit testy.

- https://refactoring.com/
- https://sourcemaking.com/refactoring
- http://www.integralist.co.uk/posts/refactoring-techniques/

Efektivní využití IDE

Motivace k využití IDE

- Uživatelská přívětivost
  - Zvýrazňování syntaxe
  - Doplňování kódu
  - Refactoring
  - Rychlejší navigace po projektu
- Funkčnost
  - Verzování kódu
  - Debugging
  - Automatická kompilace a linkování
  - Spuštění programu

- Automatické
  - Autoindentace
  - Automatická tvorba závorek
  - Zvýraznění odpovídajících si závorek
  - Doplňování jmen proměnných a metod objektů
  - Kontrola názvů proměnných
- Z nabídky
  - Vyhledání chyby
  - Pokročilé vyhledávání
  - Přepínání mezi definicí a deklarací
  - Přeložit & Spustit
  - ...

Existující vývojová prostředí

- Linux
  - KDevelop
- Windows
  - Visual Studio
- Mac
  - Xcode
- Multiplatformní
  - Visual Studio Code
  - QtCreator
  - Code::Blocks
  - Bluefish
  - Eclipse
  - NetBeans

Části vývojových prostředí

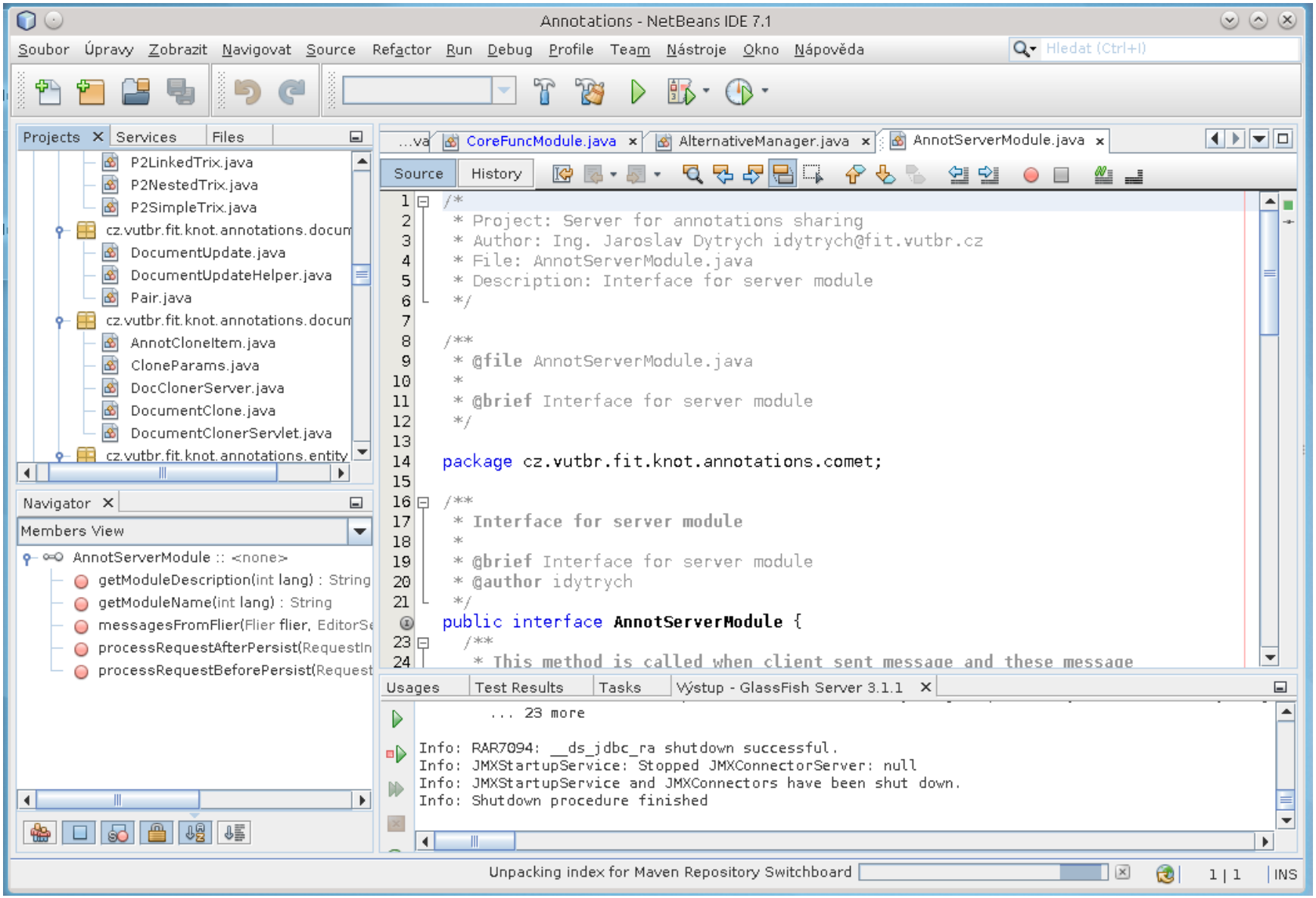
Grafická část

- Správce projektů
- GUI Designer
- Editor zdrojového kódu
- Zobrazení výstupu
- Vestavěná nápověda

Funkce

- Debugger
- Kompilátor
- Linker
- Interpret

Příklad IDE


Annotations - NetBeans IDE 7.1
Soubor Úpravy Zobrazit Navigovat Source Refactor Run Debug Profile Team Nástroje Okno Nápověda
Hledat (Ctrl+I)
Projects Services Files
P2LinkedTrix.java
P2NestedTrix.java
P2SimpleTrix.java
cz.vutbr.fit.knot.annotations.docun
DocumentUpdate.java
DocumentUpdateHelper.java
Pair.java
cz.vutbr.fit.knot.annotations.docun
AnnotCloneItem.java
CloneParams.java
DocClonerServer.java
DocumentClone.java
DocumentClonerServlet.java
cz.vutbr.fit.knot.annotations.entity
Navigator
Members View
AnnotServerModule :: <none>
getModuleDescription(int lang) : String
getModuleName(int lang) : String
messagesFromFlier(Flier flier, EditorSe
processRequestAfterPersist(RequestIn
processRequestBeforePersist(Request
CoreFuncModule.java
AlternativeManager.java
AnnotServerModule.java
Source History
1 /*
2 * Project: Server for annotations sharing
3 * Author: Ing. Jaroslav Dytrych idytrych@fit.vutbr.cz
4 * File: AnnotServerModule.java
5 * Description: Interface for server module
6 */
7
8 /**
9 * @file AnnotServerModule.java
10 *
11 * @brief Interface for server module
12 */
13
14 package cz.vutbr.fit.knot.annotations.comet;
15
16 /**
17 * Interface for server module
18 *
19 * @brief Interface for server module
20 * @author idytrych
21 */
public interface AnnotServerModule {
23 /**
24 * This method is called when client sent message and these message
Usages Test Results Tasks Výstup - GlassFish Server 3.1.1
... 23 more
Info: RAR7094: __ds_jdbc_ra shutdown successful.
Info: JMXStartupService: Stopped JMXConnectorServer: null
Info: JMXStartupService and JMXConnectors have been shut down.
Info: Shutdown procedure finished
Unpacking index for Maven Repository Switchboard
1 | 1
INS

Zásuvné moduly (plugins)

- Přídavná funkčnost
  - Podpora dalších programovacích jazyků
  - Vývoj mobilních aplikací
  - Nástroje pro testování
  - Knihovny
  - Správa verzí
  - ...
- Obvykle se instalují ve správci doplňků

Volba IDE a tipy

- Počáteční volbě IDE pro nový projekt věnujte dostatečnou pozornost.
  - Úspora času dosažená vhodnými funkcemi IDE může být velká.
  - Migrace existujícího projektu může být pracná.
- IDE za Vás může udělat spoustu věcí, ale také skrýt chyby.
  - Projekt v Mavenu může být IDE zkompilován a spuštěn i přes nedostatky v konfiguračním souboru – spuštění bez IDE je pak problém (ale je nutné u zákazníka).
  - IDE na server nasadí i projekt s chybami v konfiguraci, ale distribuční balíček pak nefunguje (nasazení na vzdáleném serveru pomocí NetBeans? – to nechcete!).
  - **Program pravidelně testujte i jinak než kliknutím na tlačítko v IDE.**
- Využití různých IDE různými členy týmu může komplikovat vývoj.
  - Může se projevit až v průběhu vývoje, když začnete využívat pokročilé funkce.

- https://www.kdevelop.org/
- https://code.visualstudio.com/
- https://visualstudio.microsoft.com/cs/
- https://developer.apple.com/xcode/
- https://www.qt.io/development-tools
- https://www.codeblocks.org/
- http://bluefish.openoffice.nl/index.html
- https://www.eclipse.org/ide/
- https://netbeans.org/
- https://www.jetbrains.com/

Využití cizího kódu

Využití cizího kódu

- Bude probráno na 5. přednášce.
  - Logicky na správném místě, ale pro některé studenty pozdě.
- Hlavní zásady:
  - **Zkontrolujte licenci**, zda kód smíte převzít (nenaleznete-li, silně zvažujte rizika).
  - **Správně citujte** – musí být zcela jasné, která část kódu, v jakém rozsahu a odkud je převzata.
  - Stack overflow (Stack Exchange Network) apod. mají jasné podmínky využití obsahu – uvedení původu (Stack Exchange Network), URL otázky, jmen a URL autora otázky a autorů využitých odpovědí apod.
  - Ve škole se doslovně citovaná část typicky nepočítá do rozsahu Vaší práce, nebo je její rozsah omezen.
- Využitý kód může ovlivnit licenci/využití výsledku Vaší práce – využijete-li knihovnu s licencí GPL, Váš program musí být distribuován s GPL (nemůžete jej uzavřít a prodat).

Citování je důležité

- **Jaký je rozdíl mezi plagiátem a nesprávnou/zapomenutou citací?**
  - **Často žádný, protože výsledek vypadá stejně!**
- Hodně špatně udělaný projekt vede na 0 bodů z projektu (třeba z 5). Opsaný projekt vede na neudělení zápočtu (často anulování bodů ze všech aktivit) a disciplinární řízení.
- Odevzdání bakalářské práce plné nesmyslů vede na její opakování. Odevzdání bakalářské práce s převzatou kapitolou bez řádné citace vede na disciplinární řízení nutně vedoucí k vyloučení ze studia!
- Když plagiátorství nezjistí vyučující hned, neznamená to, že prošlo!
  - Kolik ministrů a vysokých politiků v ČR přišlo o funkci kvůli dodatečně zjištěnému plagiátorství v závěrečné práci? – zkuste hledat „ministr plagiátorství" na Google.

- https://choosealicense.com/licenses/
- https://opensource.org/licenses/category
- https://stackexchange.com/legal

Děkuji za pozornost!