Typy dokumentace, generování programové dokumentace z kódu, identifikace existujících komponent a využívání knihoven dostupných na různých platformách

Jaroslav Dytrych

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

8. března 2023

Obsah přednášky

- Generování programové dokumentace z kódu.
- Identifikace existujících komponent a využívání knihoven dostupných na různých platformách.

Typy dokumentace

- Uživatelská příručka
  - určena pro neznalého uživatele (vyhýbáme se odborným termínům, implementačním detailům apod.),
  - stručná (jinak ji nikdo nebude číst).
- Referenční manuál
  - detailní popis programu pro uživatele a správce,
  - nepopisuje implementaci ale instalaci a použití.
- Programová dokumentace
  - popisuje zdrojové texty programu,
  - lze generovat i ze zdrojových textů programu.
- Projektová dokumentace
  - popisuje vývoj programu (teoretické základy, návrh, postup implementace, testování apod.).

- Uživatelskou příručku si uživatel čte dříve, než si program nainstaluje.
  - Umí program to, co potřebuji?
  - Co budu potřebovat, aby mi program fungoval (HW a SW)?
  - Jak s programem začít pracovat?
- Uživatelská příručka by měla obsahovat:
  - závislosti programu a HW nároky,
  - návod k instalaci programu,
  - stručný popis použití doplněný vhodnými snímky obrazovek,
  - informace o autorech aplikace.
- Uživatelská příručka by neměla obsahovat:
  - to, co uživatele nezajímá – informace o návrhu programu, implementaci, testování apod.

- Nápověda slouží jako uživatelská příručka, ale je integrována přímo v programu, abychom ji měli vždy snadno a rychle k dispozici.
  - Může být i kontextová nápověda při najetí myší na prvek apod.
  - Může být implementována tak, že se pouze otevře PDF s uživatelskou příručkou, webový prohlížeč s online dokumentací, . . .
- Na rozdíl od uživatelské příručky nápověda neobsahuje postup instalace programu.

- slouží ke generování programové dokumentace ze zdrojových textů programu,
- dostupný pod licencí GNU GPL pro UNIX/Linux i Microsoft Windows,
- podpora celé řady programovacích jazyků (**C, C++, Java, Python, VHDL, PHP, Fortran, . . .**),
- možnost využít filtry pro podporu dalších programovacích jazyků (JavaScript, Object Pascal, Visual Basic, . . .),
- **výstup** může být **v HTML, RTF, Latexu, PostScriptu, PDF či unixových manuálových stránkách**,
- s využitím nástroje dot (součást Graphviz) může generovat i grafy.

Další nástroje

- Javadoc
  - de facto standardní nástroj pro jazyk Java,
  - podpora pro angličtinu a japonštinu,
  - pro češtinu či jiný jazyk nutno vytvořit překlad (asi 300 řádků),
  - komentáře pro Javadoc zpracuje i Doxygen, ale Javadoc některé příkazy pro Doxygen nezná a ignoruje.
- phpDocumentor
  - pro PHP
  - podobné příkazy jako Doxygen
- PyDoc (Python)
- JSDoc, ESDoc (JavaScript) vs.
- YUIDoc (JavaScript, Node.js)
- ROBODoc – komentáře nekompatibilní s jinými nástroji
- ...

Komentáře

- Programová dokumentace se generuje nejenom z programových struktur, ale především z komentářů.
- Pokud je kód špatně komentovaný, dokumentace není využitelná.
- Komentáře je třeba přizpůsobit zvolenému nástroji pro generování dokumentace (v našem případě Doxygen).

- je třeba zvolit vhodné množství (raději více)
- musejí být smysluplné a srozumitelné
  - `i++; // přičte jedničku`
    `nebo`
    `i++; // navýší počítadlo vzorků`
- komentář by měl mít:
  - každý soubor (hlavička souboru, konec souboru),
  - každá třída,
  - každá procedura či funkce,
  - každá proměnná, jejíž název není samopopisný,
    - `int i; // počítadlo průběhů`
    - `int v; // počítadlo nalezených položek`
    - `int s; // počítadlo shodných položek`
  - každý blok kódu, jehož funkce není zcela zřejmá,
  - každý řádek kódu, jehož činnost není zcela zřejmá,

- komentář by měla mít:
  - každá uzavírací závorka, u které není na běžném monitoru vidět otevírací závorka
    - `    } // zpracuje 2. - N-tý řádek`
    - `  } // pokud vybral alespoň 1 řádek`
    - `} // zpracujRadky()`
  - `...`

- Komentáře pro Doxygen se umisťují do tzv. dokumentačních bloků (pouze blokové komentáře).
- Doxygen podporuje různé syntaxe bloku (viz `http://www.doxygen.nl/manual/docblocks.html`), nejběžnější je styl využívaný v C a Javě:

```
/**
 * Toto je komentář pro Doxygen
 */
```

- V bloku mohou být obsaženy příkazy pro Doxygen, např.: `@param`, `@return`, `@brief`, `@file`, `@package`, `@class`, `@todo`, `@bug`, `@author`, `@see`, ...
- Pro proměnné či složky struktur lze využít jednořádkový komentář se zkrácenou syntaxí:

```
int proměnná; /**< Popis proměnné */
```

  - Ne pro lokální proměnné v rámci bloku – ty jsou pro dokumentaci nepodstatné.

Komentování souborů (1/2)

- Hlavička souboru může být např.:

```
/*****************************************************************
 * Název projektu: Applet pro demonstraci Boyerova-Mooreova algori
 * Balíček: boyermooredemo
 * Soubor: Konstanty.java
 * Datum:  11.4.2008
 * Poslední změna: 18.4.2008
 * Autor:  Jaroslav Dytrych dytrych(at)fit.vutbr.cz
 *
 * Popis: Třída s konstantami
 *
 *****************************************************************/
/**
 * @file Konstanty.java
 *
 * @brief Třída s konstantami
 * @author Jaroslav Dytrych (dytrych)
 */
```

- Hlavička je v tomto příkladu složena ze dvou částí. První část není korektním dokumentačním blokem a Doxygen ji tedy ignoruje.
- Ne vždy potřebujeme všechny informace – někdy stačí jen hlavička pro Doxygen se stručným popisem souboru (GIT + Java s vhodnými balíčky).

Komentování souborů (2/2)

- konec souboru se označuje komentářem:

```
/*** Konec souboru nazev_souboru.java ***/
```

  - **Nemá smysl např. když je v souboru 1 třída a jeho konec je jasně daný uzavírací závorkou.**

```
/**
 * Funkce pro zjištění aktuálního obsahu buňky
 *
 * @bug Při neexistujících souřadnicích dojde
 *      k neoprávněnému přístupu do paměti
 * @todo Doplnit návratovou hodnotu pro neexistující
 *       buňku tabulky
 *
 * @param radek Řádek, na kterém se buňka nachází
 * @param sloupec Sloupec, na kterém se buňka nachází
 * @return Vrací obsah požadované buňky
 */
public String vratObsah(int radek, int sloupec)
{
  ...
}  // vratObsah()
```

Další příkazy

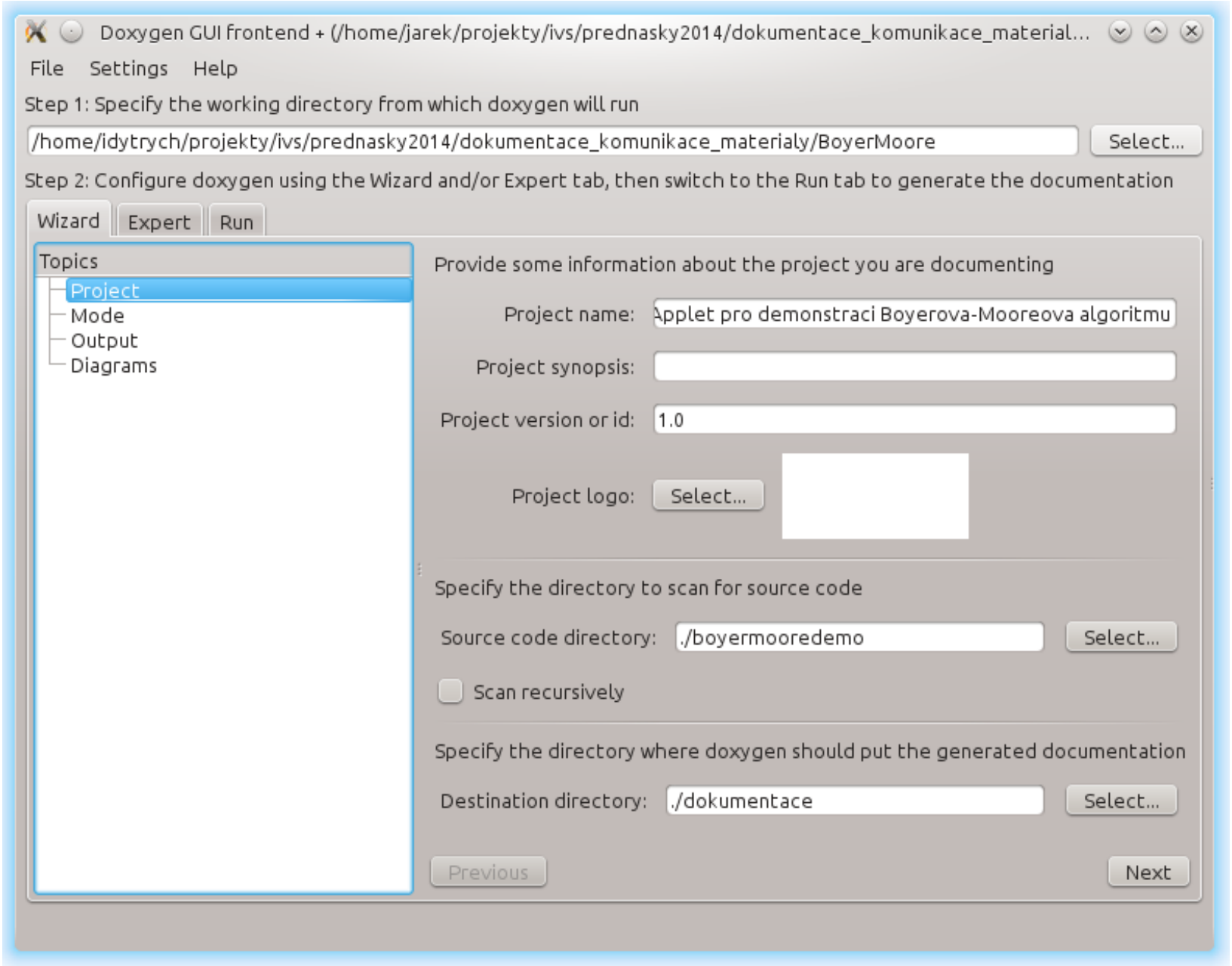
`@package`	komentování balíčku (obdobně jako u souboru; píše se do jednoho souboru v balíčku)
`@class`	komentování třídy
`@see`	umožňuje vytváření odkazů v rámci dokumentace, např. pokud jsou parametry detailněji vysvětleny u jiné varianty funkce

- více viz manuál (`http://www.doxygen.nl/manual/index.html`)

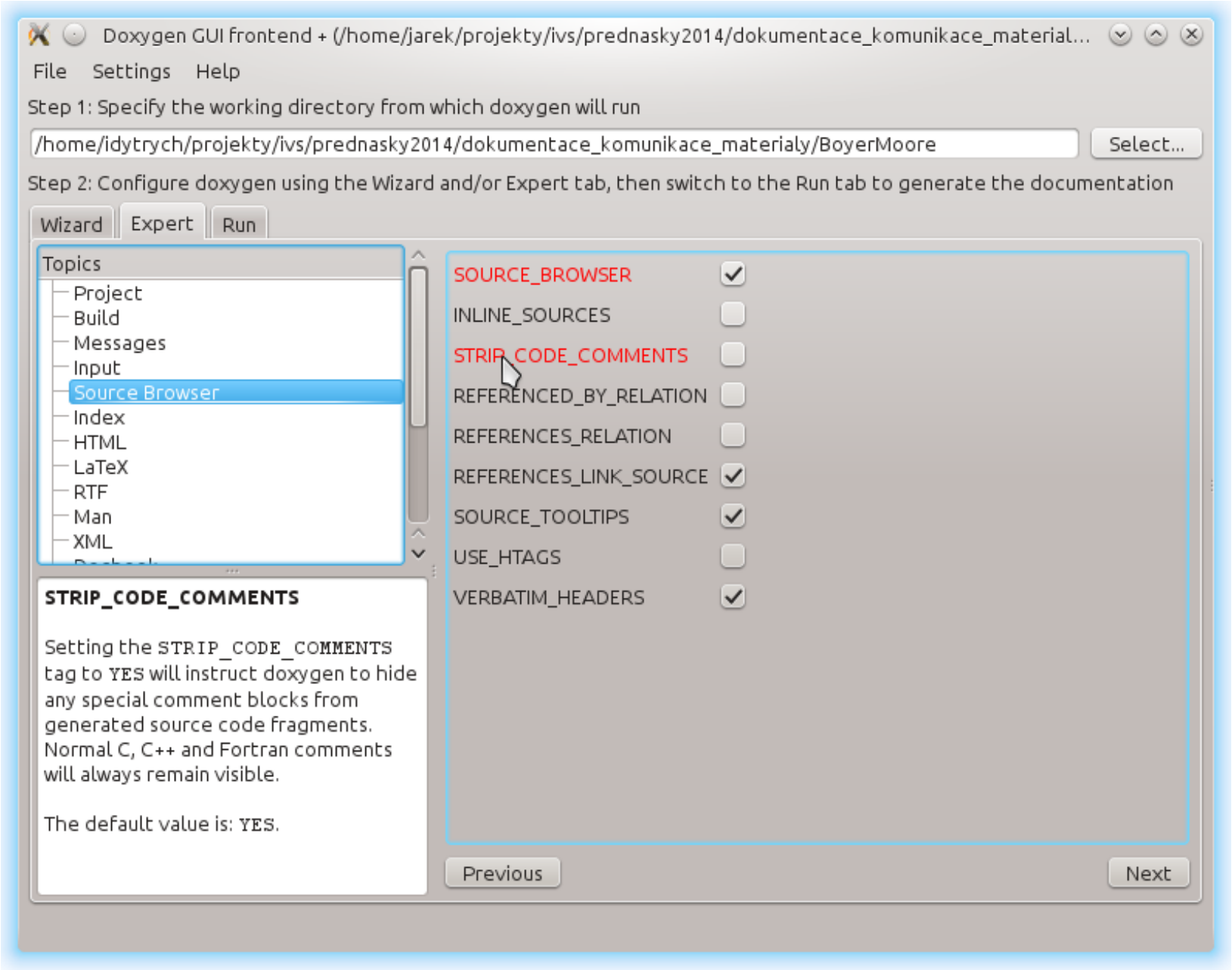
- Doxygen se využívá z příkazové řádky.
- Je třeba vytvořit soubor s konfigurací nazvaný **Doxyfile** a následně spustit Doxygen příkazem „`doxygen Doxyfile`“ či pouze „`doxygen`“.

- lze manuálně vytvořit ze šablony s komentáři, kterou vygeneruje Doxygen.
- lze vygenerovat pomocí GUR doxywizard, kde lze využít průvodce pro základní nastavení a následně doladit v záložce „Expert“.
  - Oproti manuální editaci Doxyfile je tento postup pomalejší a bez podrobného doladění nastavení generovaný Doxyfile často neposkytuje kvalitní výsledky. Výhodou je, že při využití GUR neuděláme syntaktickou chybu v Doxyfile a výběr souborů a složek je pohodlnější.

Doxywizard


Doxygen GUI frontend + (/home/jarek/projekty/ivs/prednasky2014/dokumentace_komunikace_material...
File
Settings
Help
Step 1: Specify the working directory from which doxygen will run
/home/idytrych/projekty/ivs/prednasky2014/dokumentace_komunikace_materialy/BoyerMoore
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Mode
Output
Diagrams
Provide some information about the project you are documenting
Project name:
Applet pro demonstraci Boyerova-Mooreova algoritmu
Project synopsis:
Project version or id:
1.0
Project logo:
Select...
Specify the directory to scan for source code
Source code directory:
./boyermooredemo
Select...
Scan recursively
Specify the directory where doxygen should put the generated documentation
Destination directory:
./dokumentace
Select...
Previous
Next

Doxywizard


Doxygen GUI frontend + (/home/jarek/projekty/ivs/prednasky2014/dokumentace_komunikace_material...
File
Settings
Help
Step 1: Specify the working directory from which doxygen will run
/home/idytrych/projekty/ivs/prednasky2014/dokumentace_komunikace_materialy/BoyerMoore
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Build
Messages
Input
Source Browser
Index
HTML
LaTeX
RTF
Man
XML
SOURCE_BROWSER
INLINE_SOURCES
STRIP_CODE_COMMENTS
REFERENCED_BY_RELATION
REFERENCES_RELATION
REFERENCES_LINK_SOURCE
SOURCE_TOOLTIPS
USE_HTAGS
VERBATIM_HEADERS
STRIP_CODE_COMMENTS
Setting the STRIP_CODE_COMMENTS tag to YES will instruct doxygen to hide any special comment blocks from generated source code fragments. Normal C, C++ and Fortran comments will always remain visible.
The default value is: YES.
Previous
Next

Sekce Doxyfile

- Sekce se základními nastaveními:
  - konfigurační volby spojené s projektem,
  - konfigurační volby spojené se sestavením dokumentace (build),
  - konfigurační volby spojené s varováními a zprávami o zpracování,
  - konfigurační volby spojené se vstupními soubory.
- Sekce pro nastavení výstupu:
  - konfigurační volby spojené s procházením zdrojového kódu,
  - konfigurační volby spojené s abecedním indexem tříd,
  - konfigurační volby spojené s výstupem v HTML (LaTeXu, RTF, XML, . . .),
  - konfigurační volby spojené s výstupem definic pro AutoGen (viz. `http://autogen.sourceforge.net/`),
  - konfigurační volby spojené s přidáváním externích referencí,
  - konfigurační volby spojené s nástrojem dot (generování diagramů).
- Další sekce:
  - konfigurační volby spojené s preprocessorem – předzpracování zdrojových textů.

Základní nastavení (1/3)

- Je třeba nastavit kódování, které využíváme, a to jak k vytvoření Doxyfile, tak k psaní zdrojových textů (dnes obvykle UTF-8):

```
DOXYFILE_ENCODING = UTF-8
INPUT_ENCODING    = UTF-8
```

- Z konfiguračních voleb spojených s projektem je třeba nastavit:
  - název projektu, resp. programu (`PROJECT_NAME`),
  - verzi dokumentovaného programu (`PROJECT_NUMBER`),
  - adresář pro umístění vygenerované dokumentace (`OUTPUT_DIRECTORY`),
  - jazyk, který by se měl shodovat s jazykem, ve kterém jsou psány komentáře (`OUTPUT_LANGUAGE`),
  - případně další volby dle programovacího jazyka a požadavků na dokumentaci.

Základní nastavení (2/3)

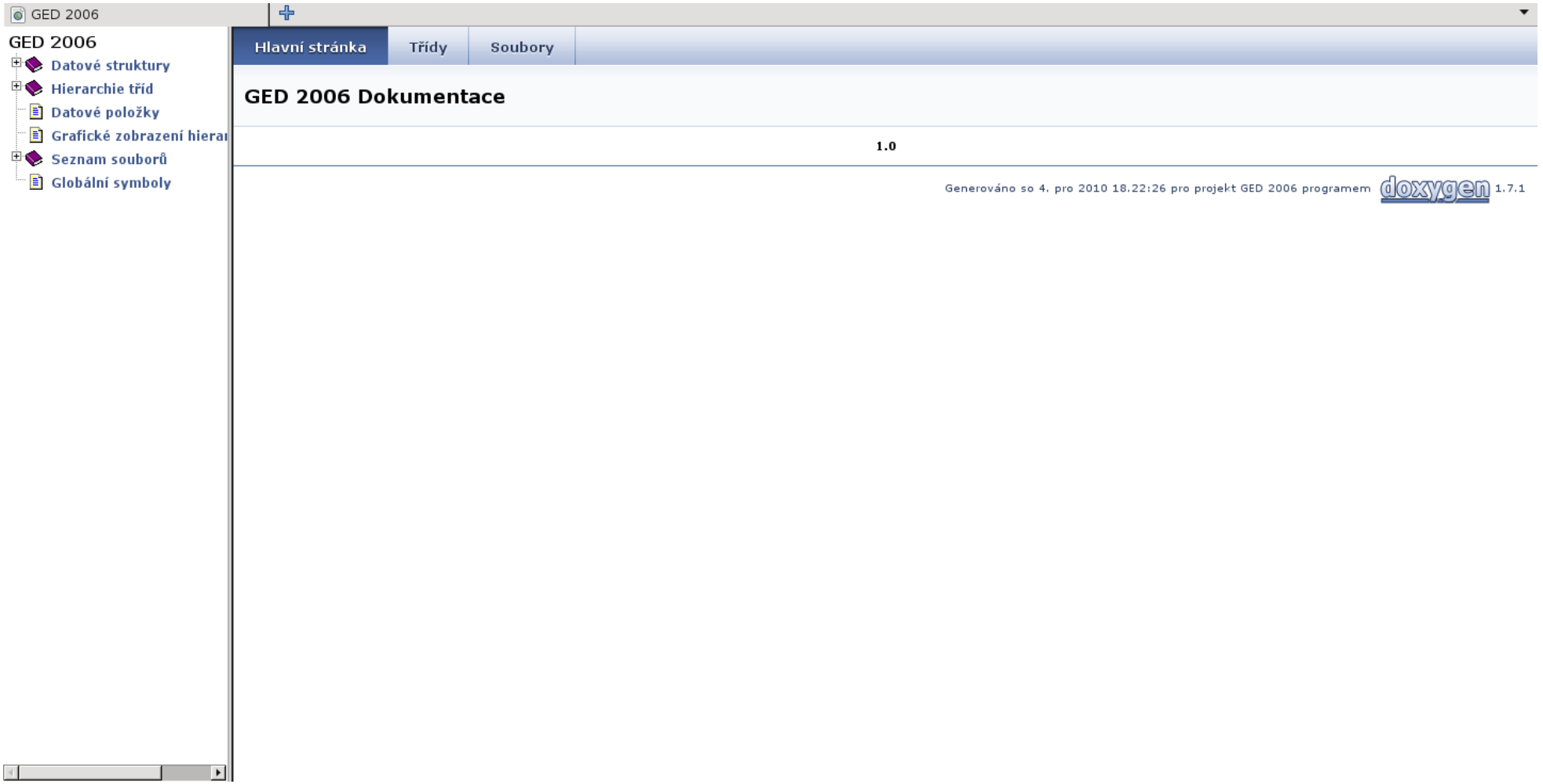
- Konfigurační volby spojené se sestavením (build) umožňují zvolit:
  - co bude ve zdrojových textech zkoumáno,
  - které části dokumentace budou generovány.
- Konfigurační volby spojené s varováními a zprávami o zpracování
  - umožňují vygenerovat i soubor se záznamem o chybách a varováních při generování.
  - Pokud vygenerovaný soubor není prázdný, v komentářích je pravděpodobně chyba, která může způsobit chybu v dokumentaci.
  - Pokud je vygenerovaný soubor prázdný, neznamená to, že je vše v pořádku (některé chyby se automaticky neodhalí).

- Konfigurační volby spojené se vstupními soubory slouží k nastavení:
  - v jakém adresáři jsou dokumentované soubory (`INPUT`),
  - které soubory má doxygen dokumentovat (`FILE_PATTERNS`),
  - zda procházet i podadresáře (`RECURSIVE`),
  - co vynechat (`EXCLUDE`),
  - kde jsou umístěny obrázky (`IMAGE_PATH`),
  - apod.

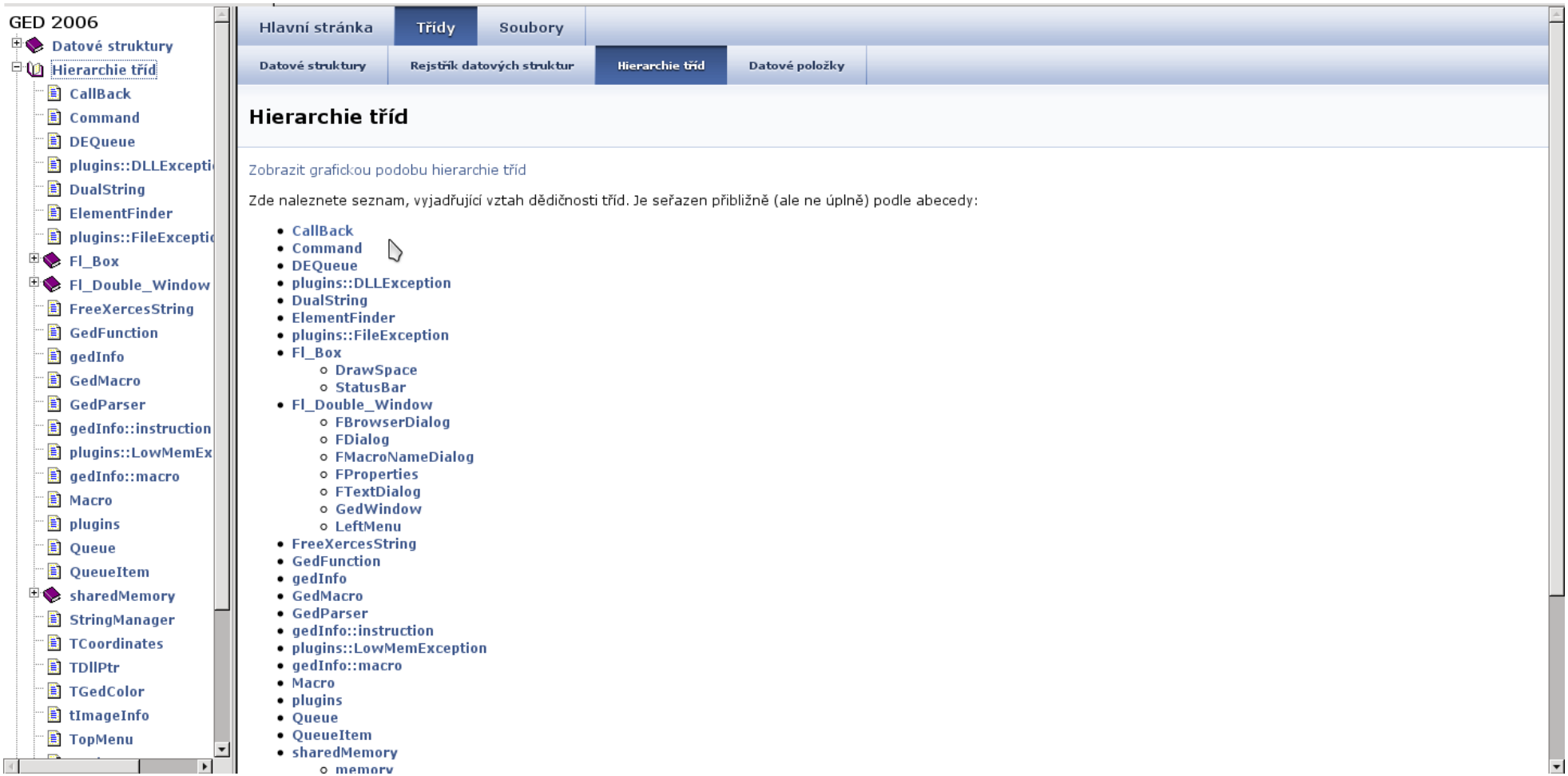
Nastavení výstupu

- Doxygen může ze zdrojových kódů vytvořit součást dokumentace (např. stránka v HTML se zdrojovým kódem se zvýrazněním syntaxe).
- Lze vytvořit abecední rejstřík tříd.
- Podle požadovaných výstupních formátů povolíme a případně upravíme nastavení výstupních sekcí týkajících se formátů.
- Můžeme povolit i generování grafů a zvolit si požadované typy grafů.
  - Pozor, při volbě nevhodného formátu obrázků dojde k výraznému nárůstu velikosti dokumentace (doporučuji obrázky v gif).

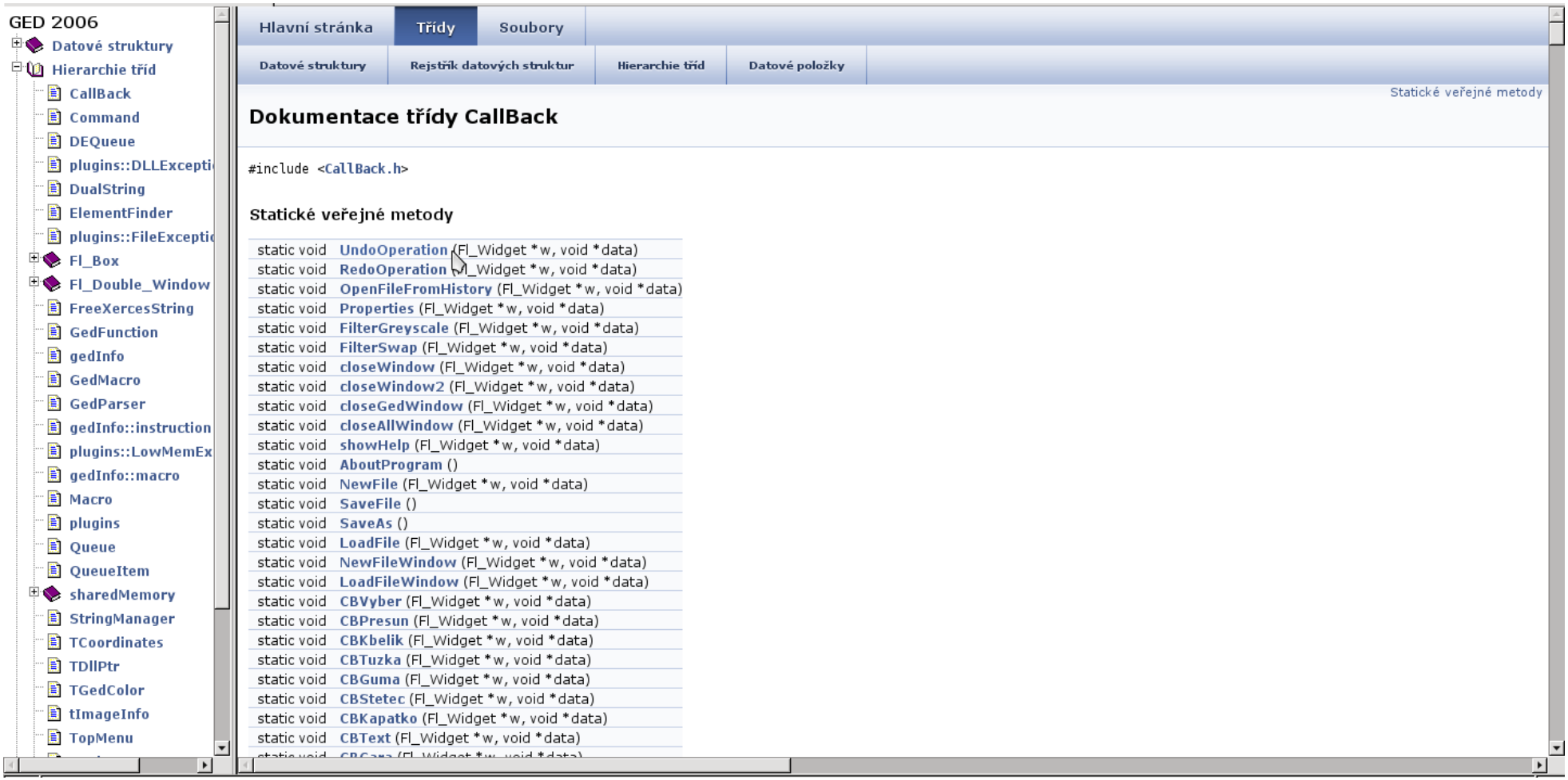
Ukázka výstupu



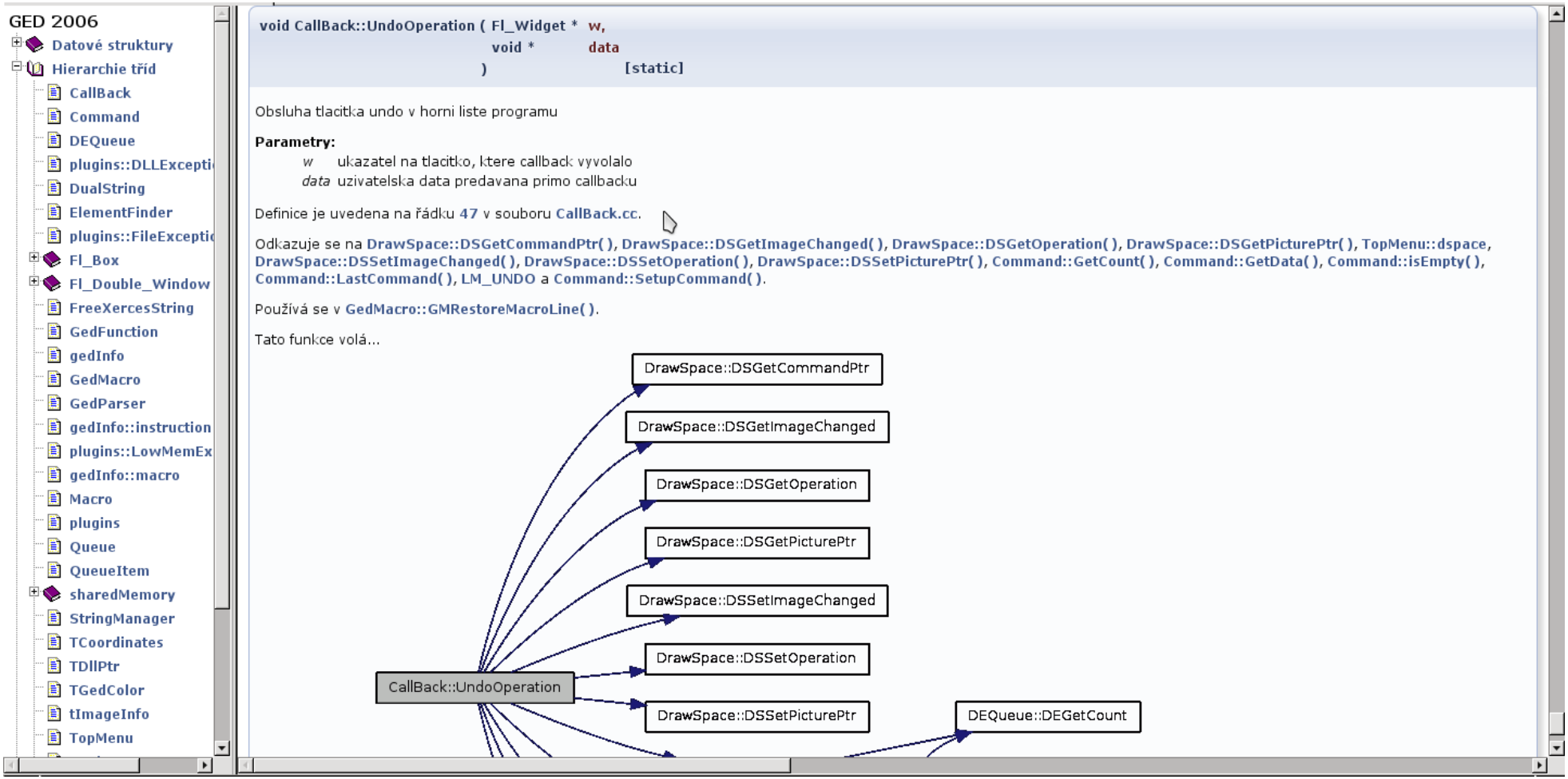
Ukázka výstupu



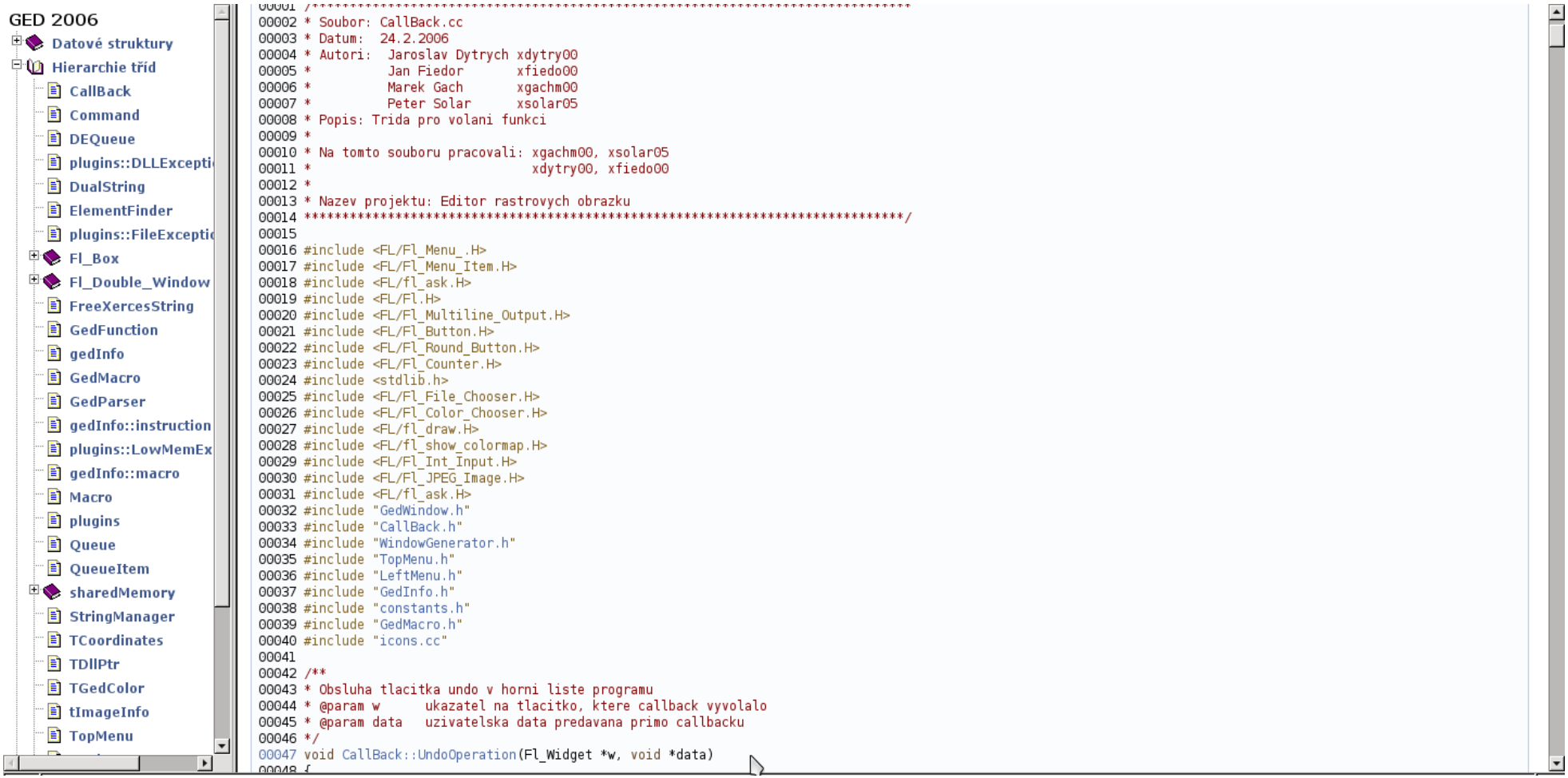
Ukázka výstupu



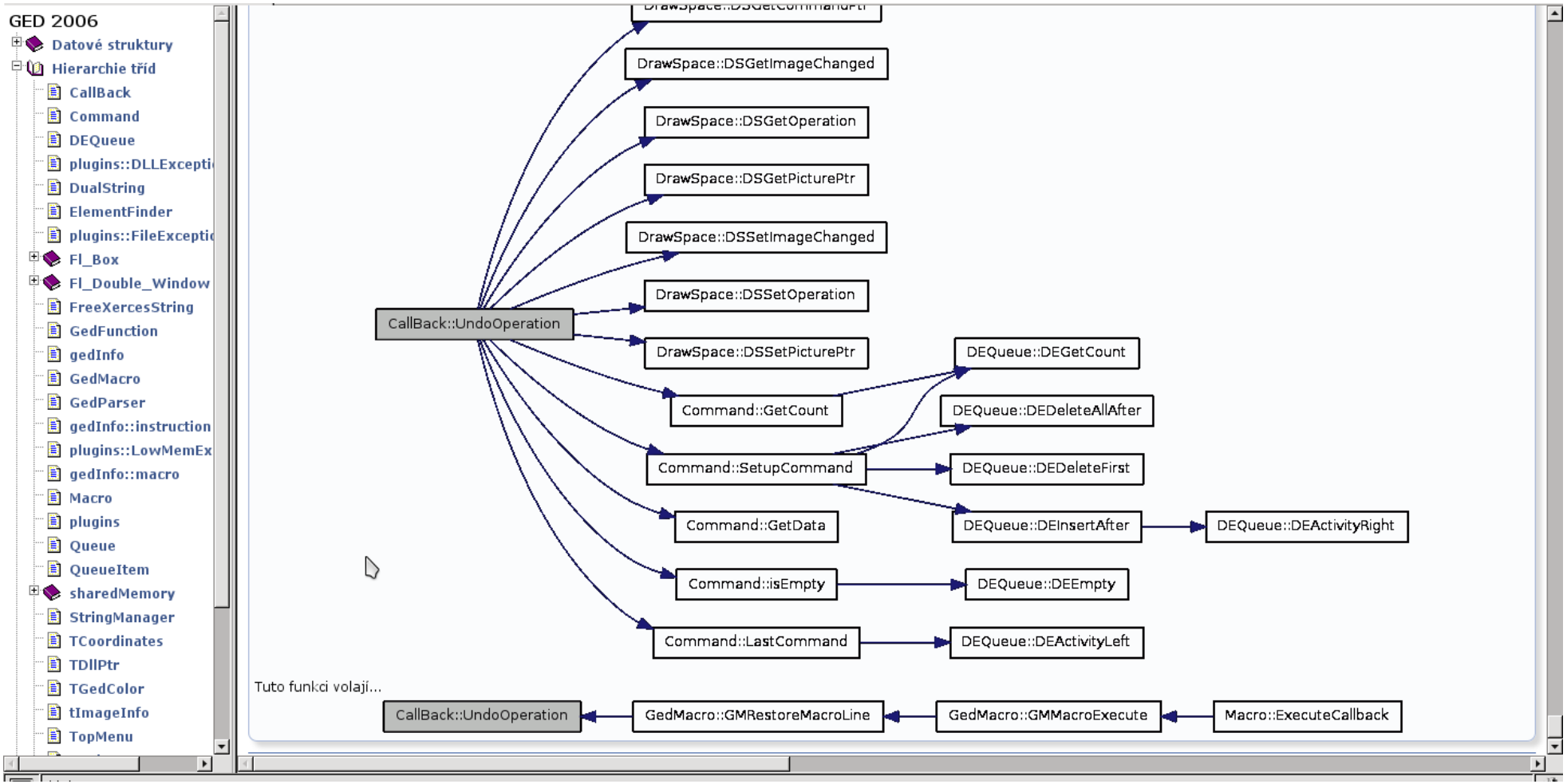
Ukázka výstupu



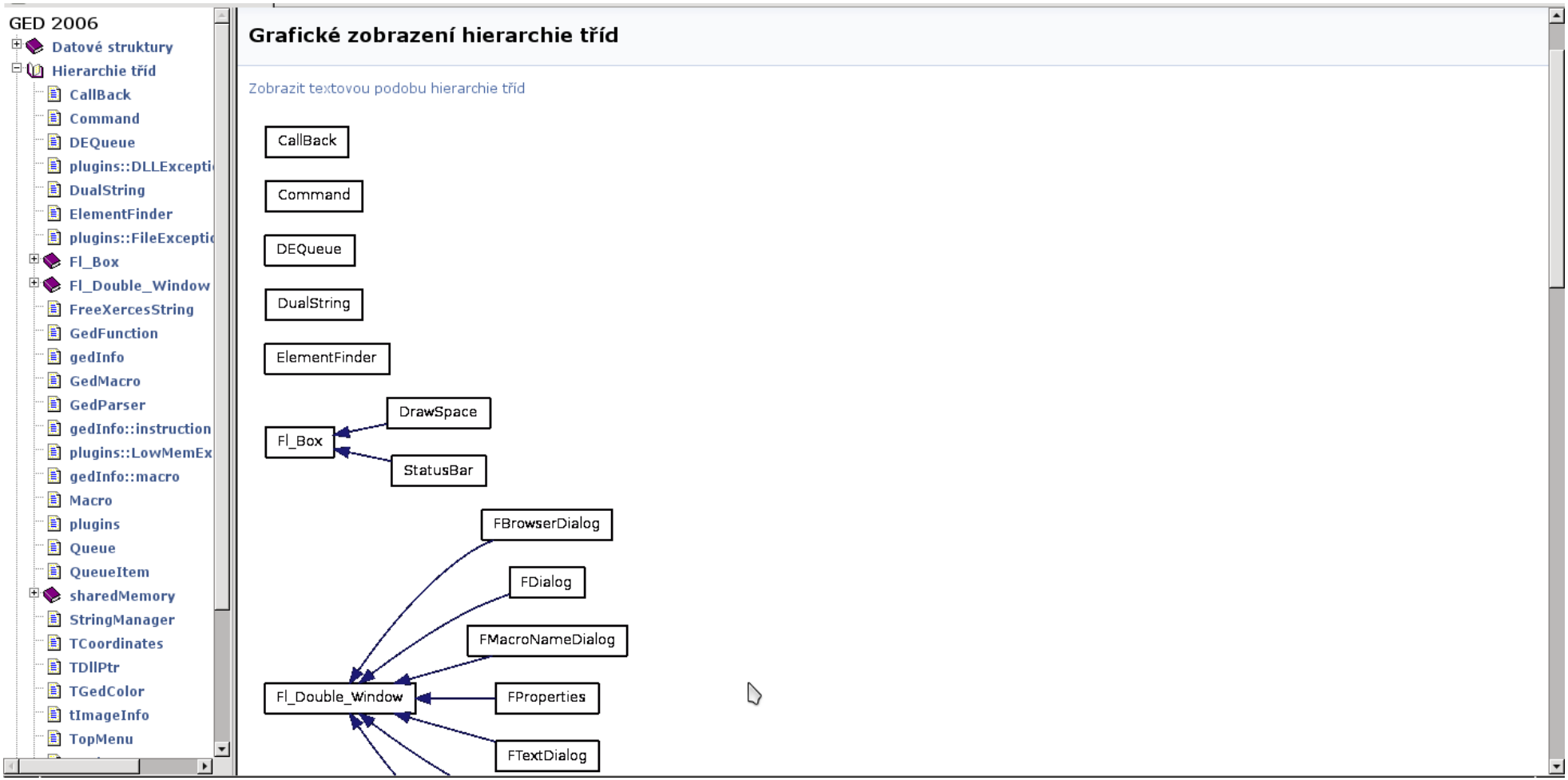
Ukázka výstupu

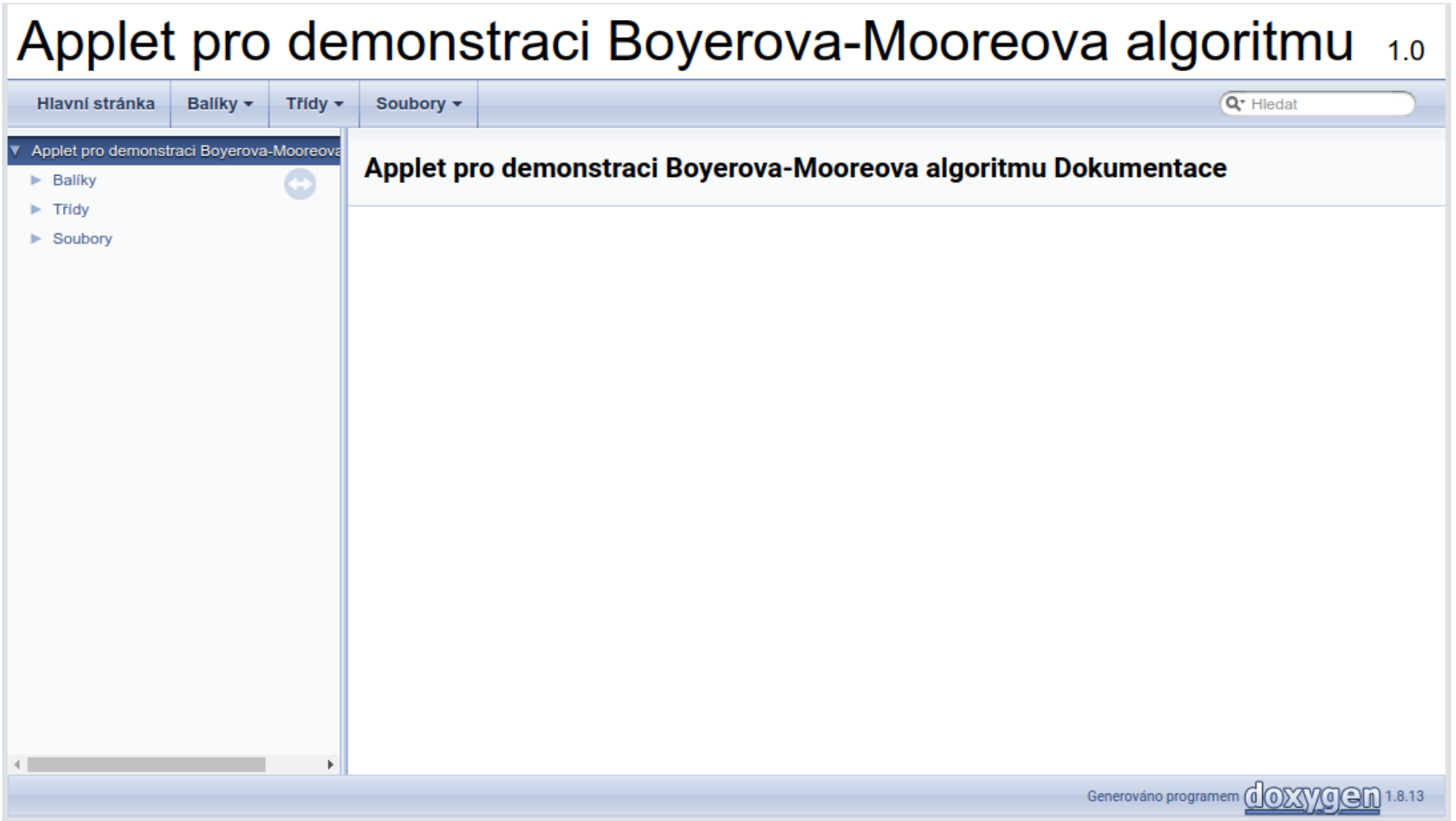


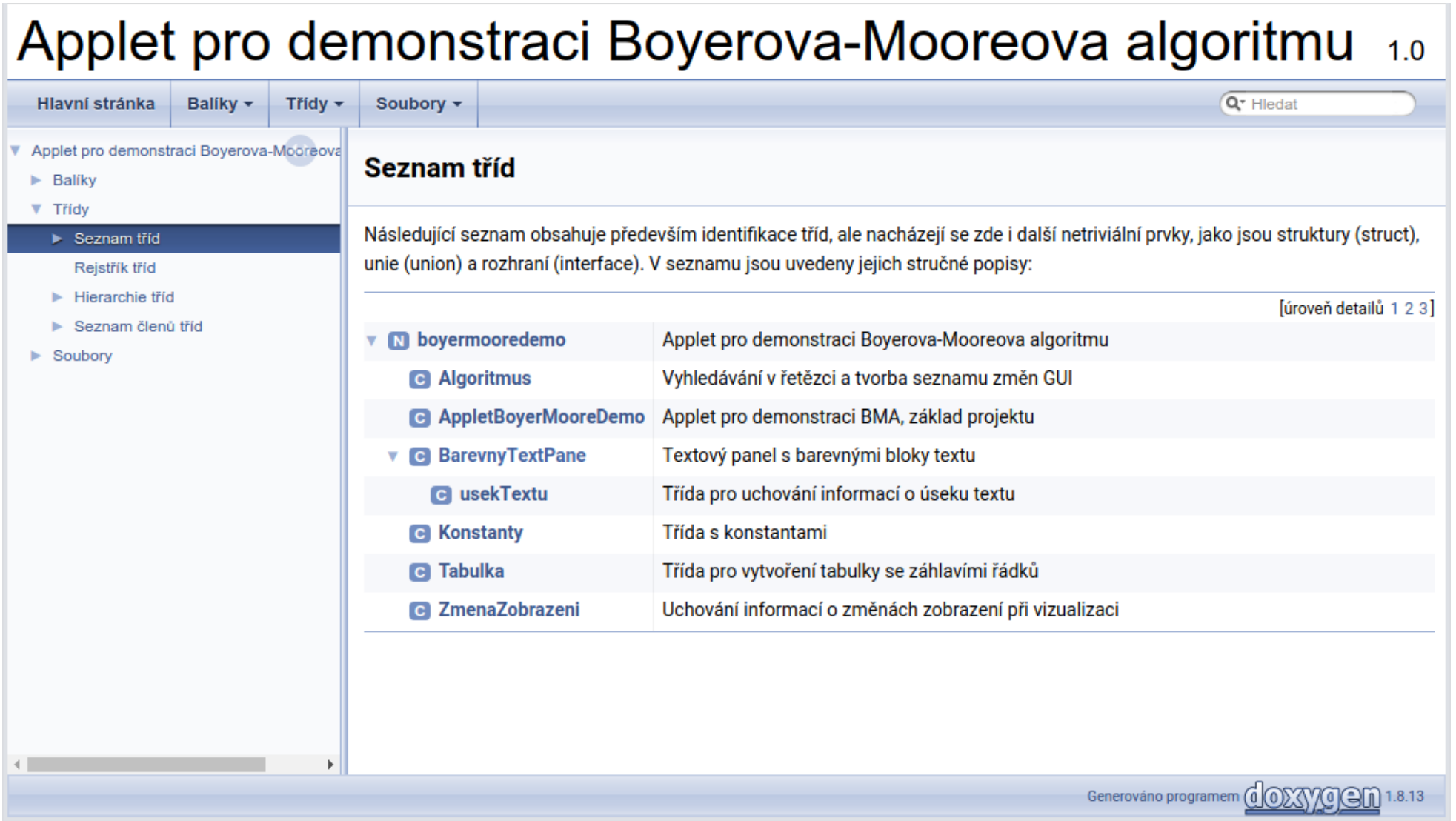
Ukázka výstupu



Ukázka výstupu




Applet pro demonstraci Boyerova-Mooreova algoritmu 1.0
Hlavní stránka
Balíky
Třídy
Soubory
Hledat
Applet pro demonstraci Boyerova-Mooreova
Balíky
Třídy
Soubory
Applet pro demonstraci Boyerova-Mooreova algoritmu Dokumentace
Generováno programem doxygen 1.8.13



Applet pro demonstraci Boyerova-Mooreova algoritmu 1.0

Hlavní stránka | Balíky | Třídy | Soubory | Hledat

- Applet pro demonstraci Boyerova-Mooreova
  - Balíky
  - Třídy
    - Seznam tříd
    - Rejstřík tříd
    - Hierarchie tříd
    - Seznam členů tříd
  - Soubory

Seznam tříd

Následující seznam obsahuje především identifikace tříd, ale nacházejí se zde i další netriviální prvky, jako jsou struktury (struct), unie (union) a rozhraní (interface). V seznamu jsou uvedeny jejich stručné popisy:

[úroveň detailů 1 2 3]

N boyermooredemo	Applet pro demonstraci Boyerova-Mooreova algoritmu
C Algoritmus	Vyhledávání v řetězci a tvorba seznamu změn GUI
C AppletBoyerMooreDemo	Applet pro demonstraci BMA, základ projektu
C BarevnyTextPane	Textový panel s barevnými bloky textu
C usekTextu	Třída pro uchování informací o úseku textu
C Konstanty	Třída s konstantami
C Tabulka	Třída pro vytvoření tabulky se záhlavími řádků
C ZmenaZobrazeni	Uchování informací o změnách zobrazení při vizualizaci

Generováno programem doxygen 1.8.13

Ukázka výstupu

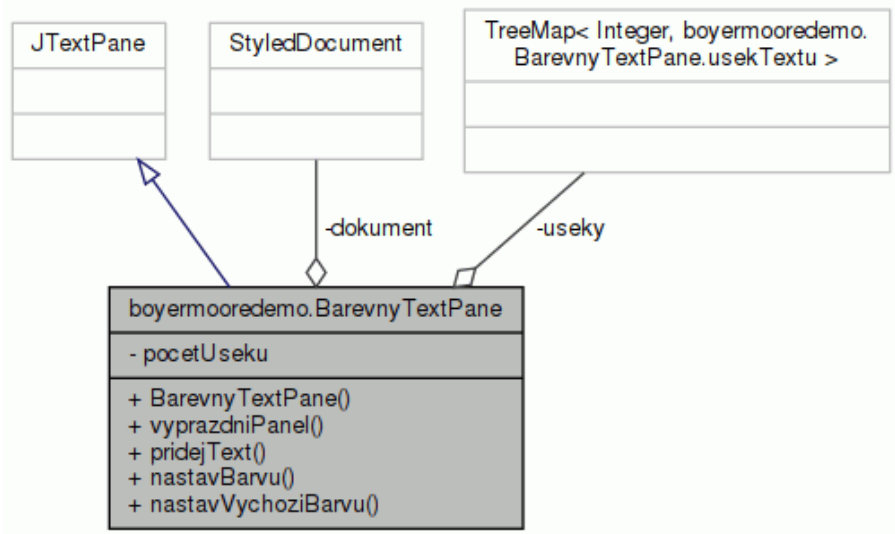
Applet pro demonstraci Boyerova-Mooreova algoritmu 1.0

Hlavní stránka | Balíky | Třídy | Soubory | Hledat

- Applet pro demonstraci Boyerova-Mooreova
  - Balíky
  - Třídy
    - Seznam tříd
      - boyermooredemo
        - Algoritmus
        - AppletBoyerMooreDemo
        - BarevnyTextPane
        - Konstanty
        - Tabulka
        - ZmenaZobrazeni
    - Rejstřík tříd
    - Hierarchie tříd
    - Seznam členů tříd
  - Soubory

+ nastavBarvu()
+ nastavVychoziBarvu()

Diagram tříd pro boyermooredemo.BarevnyTextPane:



Třídy

class **usekTextu**

Třída pro uchování informací o úseku textu. ...

boyermooredemo | BarevnyTextPane

Generováno programem doxygen 1.8.13

Více konfigurací

- V některých případech vytvoříme i více souborů Doxyfile a vygenerujeme více dokumentací s různou mírou detailu a pokrytí kódu dle účelu:
  - pro uživatele naší knihovny (zákazníky),
  - pro vývojáře (zaměstnance).

Identifikace existujících komponent . . .

Dostupné komponenty

- Hotové produkty
  - Přidání požadovaných funkcí do existujících produktů (moduly, rozšíření).
  - Přístup přes API (databáze, webové služby).
- Frameworky
  - Určují chování aplikace, které programátor pouze upravuje.
  - Většinou GUI, webové aplikace.
- Knihovny
  - Chování aplikace určuje programátor a volá funkce z knihovny.
  - Nelze vždy určit, zda se jedná o knihovnu či framework.

Identifikace existujících komponent

- Je třeba zohlednit:
  - Licence – dostupný zdrojový kód, možnosti přizpůsobení, cena, licence výsledného produktu, . . .
  - Výkon a využití zdrojů
  - Dokumentace
  - Přenositelnost – operační systémy, 32 nebo 64 bitů, ARM, . . .
  - Znalost knihovny vs. požadovaná funkcionalita
  - Volba programovacího jazyka (vysokoúrovňové programovací jazyky)
- Existující kompilátory
  - Unix - GCC, CLANG, . . .
  - Windows - MinGW, MSVC, . . .

Vybrané open source licence

Licence	Linkování s uzavřenou aplikací	Distribuce výsledku	Distribuce upraveného kódu
GPL	Ne	GPL komp.	GPL
LGPL	Ano	s omezením	GPL, LGPL
Apache	Ano	Ano	Ano
BSD	Ano	Ano	Ano

- Vlastnosti frameworku Qt
  - GUI, sítě, XML, databáze . . .
  - Napsáno v C++
  - Možno použít i v Pythonu, Ruby, . . .
  - Linux, Mac OS X, MS Windows, Android, iOS, UWP, WebAssembly, . . .
  - Komerční licence, GNU GPLv3 LGPLv3
- Moduly
  - QtCore, QtGui, QtWidgets
  - QtMultimedia, QtNetwork, Qt QML, Qt Quick
  - Qt SQL, Qt Test, Qt Webkit

- Vlastnosti knihovny
  - GUI
  - Napsáno v C
  - Možno použít i v C++, Pythonu, . . .
  - Linux, Unix, Mac OS X, MS Windows
  - GNU LGPL 2.1
- Příklady programů
  - Gimp
  - Inkscape
  - Firefox
  - Pidgin
  - Meld
  - Google Chrome (do verze 35, pak vlastní s názvem Aura)

Další knihovny

- curl a libcurl
  - přenos dat přes HTTP, HTTPS, FTP, SFTP, IMAP, ...
- OpenCV
  - zpracování obrazu, počítačové vidění, načítaní různých obrazových formátů, ...
- SDL
  - nízkoúrovňový přístup k HW pro tvorbu A/V přehrávačů
- Libxml2
  - zpracování XML
- ...

Statické × dynamické linkování

- Statické linkování
  - Použité části knihovny se při linkování stávají součástí programu.
  - K provozu stačí přeložený program.
- Dynamické linkování
  - Knihovna je přeložena zvlášť.
  - Při linkování se ukládají pouze odkazy na symboly definované v dynamické knihovně.
  - K provozu je třeba přeložený program a knihovna.
  - Šetří paměť a místo na disku.
  - Knihovna je v paměti jen jednou a může ji využívat více programů.
  - Problémy s verzemi knihoven a programů.
  - `*.dll` ve Windows, `*.so` v Linuxu, `*.dylib` na MacOS X.

Využití cizího kódu

- Ne vždy použijeme celé cizí dílo či produkt – můžeme potřebovat jen několik řádků kódu.
- Nutno najít licenci!
  - Využití kódu bez jasné licence je značně rizikové.
- Nutno správně citovat!
  - Způsob citování může být nařízen licencí.

Využití cizího kódu – licence

- Jedná-li se o kód z knihovny či nějakého produktu, licence musí umožnit využití části díla.
- Jedná-li se o kód z webu, vztahuje se na něj min. licence daného webu.
  - 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.
- Využitý kód může ovlivnit licenci/využití výsledku Vaší práce – využijete-li kód s licencí GPL, Váš program musí být distribuován s GPL (nemůžete jej uzavřít a prodat).
  - Je to přijatelná cena za několik řádků kódu?

Využití cizího kódu – citace

- Musí být zcela jasné, která část kódu, v jakém rozsahu a odkud je převzata.
  - Hlavička souboru
    - Je-li převzat (téměř) celý.
    - Vyžaduje-li to licence.
  - Počáteční komentář s popisem zdroje, autorem apod.
  - Ukončovací komentář.
- Citace musí být v souladu s licencí
  - Licence může omezit využití kódu (kde a jak jej lze použít).
  - Licence může explicitně vyžadovat, jaké informace musí být uvedeny v citaci.

Využití cizího kódu – na co dát pozor

- Ve škole se doslovně citovaná část typicky nepočítá do rozsahu Vaší práce, nebo je její rozsah omezen.
  - U bakalářské či diplomové práce se ale trestá i to, když „znovu objevujete kolo".
  - V povolené a přiměřené míře jsou citace důležitou součástí práce.
- Využijete-li něčí kód ke tvorbě úspěšného produktu, lze očekávat touhu po podílu na Vašem úspěchu.
  - Pokud má kód vhodnou licenci, do značné míry Vás chrání.
  - Ignorování licence se nemusí projevit ihned.
  - Nedostupná licence neznamená, že neexistuje.
- Nebojte se zbytečně!
  - NIHS („Not Invented Here" Syndrome) je syndrom postihující jednotlivé vývojáře i celé firmy. Jeho projevem je preferování vlastního kódu před (často lepším i cenově efektivnějším) externím řešením. To, že někdo „znovu objevuje kolo", však zákazník neocení a dochází ke ztrátě času, peněz, příležitostí na trhu apod.

- https://choosealicense.com/licenses/
- https://en.wikipedia.org/wiki/Not_invented_here
- https://www.developer.com/design/article.php/3338791/Overcoming-quotNot-Invented-Herequot-Syndrome.htm
- https://opensource.org/licenses/category
- https://stackexchange.com/legal
- https://stackoverflow.blog/2009/06/25/attribution-required/
- https://www.qt.io/
- https://www.gtk.org/

Shrnutí

- Dobré zvyklosti při psaní komentářů nám umožní vygenerovat programovou dokumentaci ze zdrojových textů.
- Vhodná volba knihoven nám může ušetřit spoustu práce a peněz.
- Při využití cizího kódu je vždy potřeba řádně prostudovat licenci a správně citovat.