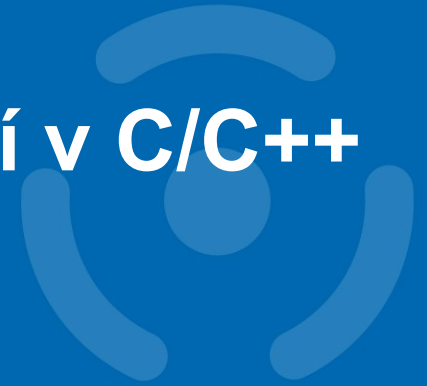


PB173 - Tématický vývoj aplikací v C/C++ (jaro 2016)



Skupina: [Aplikovaná kryptografie a bezpečné programování](#)

Petr Švenda svenda@fi.muni.cz

Konzultace: A.406, Pondělí 15-15:50

CRCS

Centre for Research on
Cryptography and Security

Refactoring

- Refactoring is process of restructuralization of source code to make it **easier to understand** and **modify in future** without changing its observable behaviour.
- No new functionality is added
- Existing code is rewritten, split, erased and otherwise modified to improve code quality

Refactoring (2)

- Coding can be divided into two parts
 - adding code for new functionality
 - refactoring existing code
- Refactoring is necessary to keep code maintainable
 - spaghetti code will sooner or later consume more time to maintain than to rewrite it
- Be aware – code rewrite might introduce new bugs
- Proper (automated) testing is required
 - that's why we have unit tests!
 - run these tests during refactoring

Refactoring – refactoring techniques

- (De-)Composing methods properly
- Moving code between modules/classes
- Change data organization
- Making conditional expressions simpler
- Making API clearer
- ...
- See <http://sourcemaking.com/refactoring>
 - detailed explanation of many techniques with examples
 - principles and practical tips how to solve problems

Code extraction into separate function

- <http://sourcemaking.com/refactoring/extract-method>
- Locate function doing multiple functionalities
- Identify logical blocks of functionality
- Create new function
 - with name describing *What* not *How*
 - move code there, replace by function call
 - think about others who also use old/new function
- Take care of local variables
 - pass them as function arguments

Additional explanation variable

- <http://sourcemaking.com/refactoring/introduce-explaining-variable>
- Add additional well-named variable to hold intermediate value
 - even when such variable is not necessary in principle
- Improve readability of code
- Increase possibility for debugging
 - you can watch and conditionally break on variable

Separate work done by single module/class

- <http://sourcemaking.com/refactoring/extract-class>
- Over the time, your module/class will grow
 - one module/class is doing multiple functionalities
- Violation of several design principles
- Identify distinct functionalities
 - usually set of methods and attributes responsible for single functionality
- Create new class(es) and move functions there
 - separate interfaces for separate functionalities
 - use multiple inheritance or aggregation to glue together

Refactoring - tools

- Most of the work with refactoring is “manual”
 - find out how to refactor and write simpler code
- Tools can still help
 - identify problematic areas (Code metrics, SourceMonitor)
 - provide call graph and data flow (Doxygen, VS Profiler)
 - apply transformation consistently in all project files

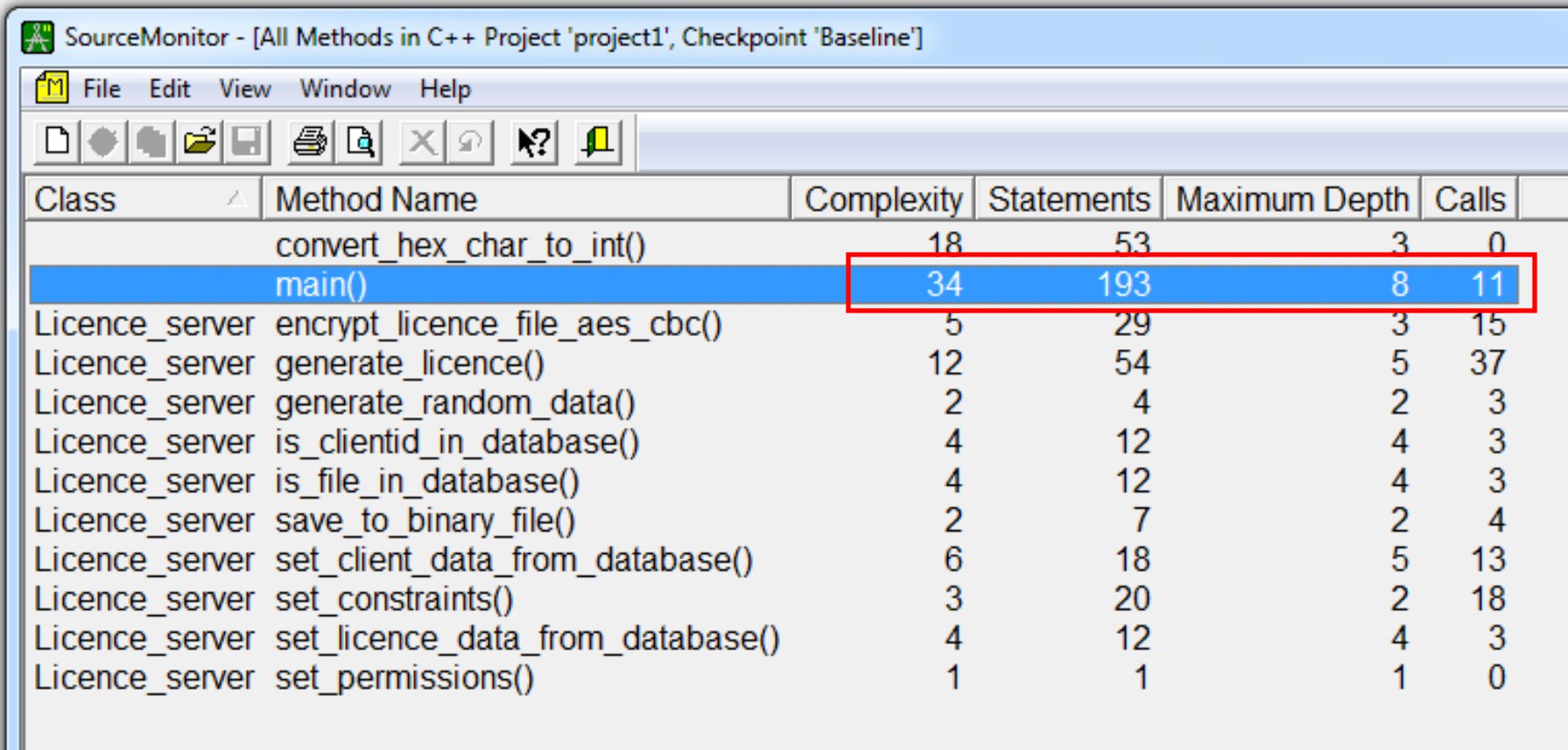
Refactoring – tools (2)

- Refactoring support for C/C++ in VS 2015
 - <https://visualstudiogallery.msdn.microsoft.com/164904b2-3b47-417f-9b6b-fdd35757d194>
- 3rd party add-ons like Visual Assist X / VSCommands
 - not only refactoring support, but also code completion...
 - <http://www.agile-code.com/blog/list-of-visual-studio-code-refactoring-tools/>
- NetBeans (and others) have refactoring support
 - <http://wiki.netbeans.org/Refactoring>
 - variable renaming, code extraction...

Source monitor

- Create new project
 - File → New project
 - language, directory with sources *.c / *.cpp
 - initial 'Baseline'
- After code update
 - Checkpoint → New checkpoint
- Details on particular checkpoint and file
 - RClick → Display Function Metrics Details...

Source monitor – example outputs



SourceMonitor - [All Methods in C++ Project 'project1', Checkpoint 'Baseline']

File Edit View Window Help

Class	Method Name	Complexity	Statements	Maximum Depth	Calls
	convert_hex_char_to_int()	18	53	3	0
	main()	34	193	8	11
Licence_server	encrypt_licence_file_aes_cbc()	5	29	3	15
Licence_server	generate_licence()	12	54	5	37
Licence_server	generate_random_data()	2	4	2	3
Licence_server	is_clientid_in_database()	4	12	4	3
Licence_server	is_file_in_database()	4	12	4	3
Licence_server	save_to_binary_file()	2	7	2	4
Licence_server	set_client_data_from_database()	6	18	5	13
Licence_server	set_constraints()	3	20	2	18
Licence_server	set_licence_data_from_database()	4	12	4	3
Licence_server	set_permissions()	1	1	1	0

- Complexity: 1-10 (OK), 11-20 (sometimes), > 20 (BAD)

Antipatterns

- Common defective process and implementation within organization
- Opposite to design patterns
 - see http://sourcemaking.com/design_patterns
- Read <http://sourcemaking.com/antipatterns>
 - good description, examples and how to solve
- Not limited to object oriented programming!
- Software development antipatterns
 - <http://sourcemaking.com/antipatterns/software-development-antipatterns>

Practical assignment - refactoring

- Use code metric tool to analyze your sources
 - <http://www.campwoodsw.com/sourcemonitor.html>
- Do it NOW! Discuss
- Find and refactor all functions
 - with complexity more than 15
 - with Maximum Depth more than 4
- Read and use refactoring techniques
 - <http://sourcemaking.com/refactoring>
- Make your code nice, readable and tidy!

Assignment – prepare for presentation

- Prepare refactored code and presentation
 - 15 minutes presentation
 - Design, actual implementation
 - Live demo of implementation
 - Single zip for download (for other teams for code review)
- Deadline 9.5.2016 12:00
 - HW Vault: Crypto - 9. homework (Presentation)