Krzysztof (Kris) Jusiak
http://kris.jusiak.net
kris@jusiak.net
(0) 791-384-1386

Software Engineer

Education

2005 - 2010 **Wroclaw University of Technology** **Wroclaw (Poland)**

MSc in Computer Science, specialised in Software Engineering (Top grade)

Employment

2013 – Present **King** **London (United Kingdom)**

Game/Software Developer (Mobile)

King is a worldwide leader in casual games with more than 30 billion games played per month globally. We are a leading interactive entertainment company for the mobile world. Our mission is to provide highly engaging content to our audience to match their mobile lifestyles: anywhere, anytime, through any platform and on any device.

Software Developer in a scrum team responsible for development and releases of mobile games played by millions active daily players. 90% of my job is focused on developing software using C++14 standard for different platforms such as, iOS, Android and Facebook (HTML5 - Emscripten). My core role is to provide high quality features and review/refactor already implemented. I am also involved in recruitment process by being a technical expert on interviews. I have done a lot of improvements to the projects I have been involved in, such as, performance/compile times optimizations, introduction of static analysis tools as well as a replacement of Service Locator pattern by Dependency Injection.

- Implementing and releasing multi-platform games played by more than 100 millions active daily players
- Reduced time to render the textures by 10% by changing loading files caching mechanism
- Implemented an integration test framework which eliminated commonly reoccurring issues

2009 – 2013 **Nokia Networks** **Wroclaw (Poland)**

Software Engineer

Nokia Networks is the world's specialist in mobile broadband, which helps enable end users to do more than ever before with the world's most efficient mobile networks, the intelligence to maximize their value and the services to make it all work together.

Software Engineer in a scrum team (tens of small - 6-8 members – teams around the world) responsible for development of management system for Long Therm Evolution 3G technology (LTE) base station. I am involved in developing of Control Plane part of the base station which is written in multi-threaded C++ using Standard Template Library and Boost libraries. There are multiple target platforms using embedded Linux. GNU GCC is the main compiler and Git/Subversion are used as a version control systems. My core role is to provide high quality features according to the 3GPP specification. In order to accomplish this effectively agile methodology – Scrum – is used as well as eXtreme programming techniques such as test-driven development, pair programming, pair review and continuous integration (Jenkins). To avoid memory and performance issues I am using tools for source code dynamic and static analysis (Valgrind, Klockwork). Part of my job is to test my code, therefore all code I do write I start from the test. Besides unit-testing (Google Test for tests and Google Mock for mocking) I am responsible of providing black bock testing (System Component Tests) using TTCN-3 language. Outside my core role I am involved in estimation of features/items, their design - using agile modeling and Unified Modeling Language - and code review after they are implemented. Since Nokia Siemens Networks is focused on clients and quality, part of my job is also focused on that, therefore I am

involved in fixing of critical customer issues such as performance degradation and memory consumption problems. Besides that I do prepare internal training from template meta-programming, Standard Template Library and Boost libraries as well as I am involved in recruiting system by being a technical expert on interviews. I have done a lot of improvements to the project, not only in the main source code, but also in the build system and in source code generation. I have also taken the advantage of my previous experience in W-CDMA project and adopt improved version of test scripts using Python language.

- Improved base station links capacity by 200% by optimizing - performance critical - links set-up code
- Fixed blocking issues during software trial (Christmas time) by implementing base station memory remote tracker
- Reduced event dispatching execution time by 30% by implementing declarative/compile-time state machine

2009 **Nokia Networks** **Wroclaw (Poland)**

Software Engineer (Test Automation), Intern

Nokia Networks is the world's specialist in mobile broadband. From the first ever call on GSM, to the first call on LTE, Nokia Siemens Networks operate at the forefront of each generation of mobile technology and provide the world's most efficient mobile networks, the intelligence to maximize the value of those networks, and the services to make it all work seamlessly.

My first 'half-time' job during studies. I was responsible for analysis and automation of test scripts in Wideband Code Division Multiple Access project (W-CDMA). I was working on redesigning of legacy testing framework, which was written in Perl. We worked on that within a small team of four people. There were few attempts before to improve the framework, but they hadn't succeeded. After few weeks of studding requirements and existing framework myself and one of my colleagues finally found out the way framework was working. After that, decision about redesign was made and I was involved in improving the framework, since serial approach was replaced by a parallel one. An improved version was written in Python instead of Perl. Within 3 months regression was downgraded from 8 hours to 40 minutes. Afterwords I was responsible for creating the documentation of improved version of test framework as well as for integration it with continuous integration tool (Cruise Control).

- Reduced regression testing time from 8 hours to 40 minutes by implementing parallel execution test framework

Talks

2016 **C++ Now** **Aspen, Colorado**

C++14 Dependency Injection Library

- http://boost-experimental.github.io/di/cppnow-2016
- https://www.youtube.com/watch?v=comZthFv3PM

2016 **C++ Now** **Aspen, Colorado**

C++14 Meta State Machine Library

- http://boost-experimental.github.io/msm-lite/cppnow-2016

2016 **C++ Now** **Aspen, Colorado**

Let's make a web match-3 game in C++14

- http://modern-cpp-examples.github.io/match3/cppnow-2016

- http://modern-cpp-examples.github.io/match3

2016 **Meeting C++** **Berlin, Germany**

Implementing a web game in C++14

- https://github.com/modern-cpp-examples/match3

Projects

2012 - Present **Boost.DI - C++ Dependency Injection Framework**

https://github.com/boost-experimental/di

Open source framework in which I am responsible for all aspects of software engineering. Project idea is taken from Google Guice framework which is a dependency injection framework for Java language. I have done widely research, but I couldn't found any C++ framework for dependency injection, which would meet my requirements and wouldn't use XML file as a configuration. Therefore I have decided to write a new one, which will not be just a copy of Google Guice for C++. My main goal from the very beginning was to use all of the C++ potential, Therefore meta-programming techniques were highly used within the project as well as C++14 standard features in order to provide the highest interface flexibility (variadic templates, template aliases, move semantic, rvalue references, lambda, auto, smart pointers). The goal is to add the framework to the Boost libraries. I am responsible for all aspects of software engineering, code is stored using Git repository and was successfully tested on variety of C++ compilers (GNU GCC, Clang, Visual C++, Intel C++) and platforms (Linux, Windows, MacOs).

2016 - Present **Boost.MSM-lite - C++ Meta State Machine Library**

https://github.com/boost-experimental/msm-lite

C++14, header only eUML-like Meta State Machine library with no dependencies. Proposed to boost in early 2016 library is focused on providing an efficient and elegant way to specify state machine definition using a Domain Specific Language. I am responsible for all aspects of software engineering, code is stored using Git repository and was successfully tested on variety of C++ compilers (GNU GCC, Clang, Visual C++, Intel C++) and platforms (Linux, Windows, MacOs).

2016 **Match3**

https://github.com/modern-cpp-examples/match3

Open source game written in C++ using C++14 standard. I am responsible for all aspects of software engineering. Game is written in modern C++ style using model view controller pattern (MVC), meta state machine and dependency injection framework. Simple DirectMedia Layer (SDL 2) is used as a graphics library. Tests are written using Google Test and Google Mock, wherein mocks are generated using 'Google Mock' mocks generator. Game is stored using Git repository and supports multiple platforms (Linux, Windows).

2013 - Present **Automatic Mocks Injector**

https://github.com/krzysztof-jusiak/mocks_injector

C++14, header only library providing automatic, type safe mocks injection. Library is based on Dependency Injection and provides support for Unit Testing as well as for Integration Testing. Code is stored using Git repository and works on newest versions of Clang, GCC and Visual Studio compilers.

2013 - 2014 **Extension for Boost Meta State Machine**

https://github.com/krzysztof-jusiak/msm

Fork of Boost Meta State Machine (MSM) library (http://www.boost.org/doc/libs) in which I am responsible for writing extensions to the original library. Project allows having non default constructors within actions and guards as well as process non typed events. Additionally integration with the C++

Dependency Injection Framework was introduced. The goal is to merge the functionality into the Boost MSM library. Project is stored using Git repository and written according to the boost coding guideline using C++03 standard. Tests are written using Boost Test and verified on variety of platforms and compilers.

2013 **'Google Mock' mocks generator**

https://github.com/krzysztof-jusiak/gmock

Open source script written in Python for generating 'Google Mock' like mocks using Clang compiler tools. Project idea was to write a script which could be easily integrated with the build system in order to avoid hand written mocks. I was responsible for all aspects of software engineering. Code is stored using Git repository and works on any Python (Python from 2.7) and Clang compiler compliant platform.

2013 - Present **MaxCad**

Software Architect, Software Engineer

Commercial project within a small international team. I am responsible for developing an application which improves projecting of printed circuit boards. Project is written in C++, stored in Git repository and using agile methodologies. Application uses Graphical User Interface written in Qt library (wxWidgets library before) as a communication with a client and is working on many platforms (Windows, Linux, Android).

2011 - 2012 **C++ Quick Finite State Machine**

https://github.com/krzysztof-jusiak/qfsm

Open source framework, which idea was born on daily work as a Software Engineer in Nokia Siemens Networks. Since designing of state machines was huge part of my work, I have been looking for a good finite-state machine – Unified Modeling Language compliant - framework. The best project I have found was the Boost Meta State Machine, unfortunately it couldn't be used in embedded environment (C++Finite State Machine frameworks comparison: https://github.com/krzysztof-jusiak/doc). Therefore I have decided to provide lighter version of it, especially if it comes to compile times. One of the main priorities was to keep Meta State Machine performance (meta-programming techniques) and, at the same time, improve the compile times. Additionally logging feature was added. Since project was designed from scratch I have decided to use more of modern design patterns. In order to make testing easier, dependency injection was hugely used. Beside the code, additional translator was written (Python) in order to make designing of the state machines easier. If design is done using UML tools it can be easily transformed into high efficient C++ code. I was responsible for all aspects of software engineering, code was stored using Git repository and successfully tested on variety of C++ compilers (GNU GCC, Clang, Visual C++, Intel C++) and platforms (Linux, Windows).

2009 - 2011 **C++ Template Unit Test Framework**

http://tut-framework.sourceforge.net

Open source project I was involved in the past by being responsible for implementation of architecture independent stubbing/wrapping method for C++, including template functions. TUT is a great unit test framework, since it's really lightweight (header files approach) and macros free. Unfortunately there is no a good mocking framework which would be able to wrap template methods and could be easily integrated with a unit test framework. Therefore the whole idea was to make stubbing/wrapping methods possible, especially including template functions. In order to achieve that instrumentation of functions was used (instrument-functions). Approach was really successful, but some part of the code had to be written in plain assembler. In the end framework was working on x86 architecture (Linux) and because of differences in Architecture Binary Interface porting to 64 bits wasn't successful. Code was written in C++ using template meta-programming techniques and stored in Subversion version control system.

Training

2015 **Advanced GitHub**

Mastering Git and GitHub Enterprise by GitHub-Training.

2014 **Effective Modern C++**

Internal seminar with Scott Meyers about C++11/14.

2014 **Modern OpenGL with QT**

Practical training by KDAB about usage of modern OpenGL with QT library.

2013 **Concurrent and distributed programming in C++**

Practical training by Infotraining about multi-threading and distributed programming in C++ using Boost and Poco libraries.

2013 **Root Cause Analysis and Escaped Defect Analysis**

Training by Nokia Siemens Networks about how to avoid the same issues in the future by using analysis techniques.

2012 **Design Patterns**

Practical training by Infotraining about modern design patterns in C++.

2012 **Coaching**

Practical training by TNM Coaching about coaching techniques.

2012 **Team Communication**

Training by Grupa Set about how to make a good team spirit, how to improve team communication within a team and how to improve the perception of a team from outside.

2011 **Agile Modeling**

Practical training by Nokia Siemens Networks about agile modeling.

2010 **Test Driven Development**

Practical training by Craig Larman about eXtreme programming techniques, mostly focused on test-driven development.

2010 **Signalling in E-UTRAN/LTE**

Training by Leliwa about basic knowledge of Long Therm Evolution (LTE).

Skills

- ✓ Experience with modern C++ design and design patterns
- ✓ Experience with C++14 standard, standard template library (STL), Boost libraries and Ranges
- ✓ Experience of using scripting languages (Python) and shell scripting
- ✓ Experience of working in a team and with coaching techniques
- ✓ Experience with agile methodologies (Scrum) and all aspects of software engineering
- ✓ Experience with agile modelling including design using unified modelling language (UML)
- ✓ Experience with eXtreme programming techniques such as test-driven development (TDD), acceptance test-driven development (TDD), behaviour driven development (BDD) pair programming, pair review, continuous integration
- ✓ Experience with embedded programming in Linux/Unix environment
- ✓ Experience with mobile development (Android/iOS)
- ✓ Experience of writing high efficient libraries/frameworks using template meta-programming techniques
- ✓ Experience with variety operating systems such as Linux (Gentoo, RedHat), Unix (FreeBSD), Android, iOS and Windows
- ✓ Experience of working with version control systems (Git, Mercurial)
- ✓ Experience with unit test/mocking/dependency injection C++ frameworks (Google Mock, Google Test, Template Unit Test Framework)
- ✓ Experience of writing domain specific languages
- ✓ Strong analytical and problem solving skills
- ✓ Knowledge of C++ graphical user interface / multimedia libraries (wxWidgets, Qt, SDL2)
- ✓ Knowledge of modern web technologies (JavaScript, ASP.Net, PHP, XML, CSS)
- ✓ Knowledge of managed programming languages (Java, C#), system programming languages (C/C++, D), assembly languages (x86, x86-64) and structured query language (SQL)
- ✓ Knowledge of artificial intelligence techniques such as neural networks, genetic algorithms, genetic programming, expert systems

Key Focus

- ✓ Modern C++ (including meta-programming)
- ✓ Performance/Optimization
- ✓ Quality (eXreme Programming techniques, TDD/BDD)
- ✓ Linux/Embedded/Mobile
- ✓ Team work/Agile environment

Languages

- ✓ English - full professional proficiency (IELTS: band 8)
- ✓ German - elementary proficiency
- ✓ Polish - native proficiency

Organizations

- ✓ ACCU

Interests

- ✓ travelling, basketball, cycling, micro-controllers