Essay Proposal

COMP160 - Software Engineering Essay

KG197307

March 16, 2017

Topic

My essay will be on: The economic impact of making code reusable and reusing code between computer platforms and environments including robotics.

Paper 1

Title: On device abstractions for portable, reusable robot code

Citation: [?]

Abstract: "We seek to make robot programming more efficient by developing a standard abstract interface for robot hardware, based on familiar techniques from operating systems and network engineering. This paper describes the application of three well known abstractions, the character device model, the interface/driver model, and the client/server model to this purpose. These abstractions underlie Player/Stage, our open source project for rapid development of robot control systems. One product of this project is the Player Abstract Device Interface (PADI) specification, which defines a set of interfaces that capture the functionality of logically similar sensors and actuators. This specification is the central abstraction that enables Player-based controllers to run unchanged on a variety of real and simulated devices. We propose that PADI could be a starting point for development of a standard platform for robot interfacing, independent of Player, to enable code portability and re-use, while still providing access to the unique capabilities of individual devices."

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/1249233/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=1249233`

Comments: Interesting main point of the paper focuses on the goal of making a standard abstract interface for programming for robot hardware which covers many purposes using such a wide variety of hardware specs.

Paper 2

Title: The Scalability-Efficiency/Maintainability-Portability Trade-Off in Simulation Software Engineering: Examples and a Preliminary Systematic Literature Review

Citation: [?]

Abstract: Large-scale simulations play a central role in science and the industry. Several challenges occur when building simulation software, because simulations require complex software developed in a dynamic construction process. That is why simulation software engineering (SSE) is emerging lately as a research focus. The dichotomous trade-off between scalability and efficiency (SE) on the one hand and maintainability and portability (MP) on the other hand is one of the core challenges. We report on the SE/MP trade-off in the context of an ongoing systematic literature review (SLR). After characterizing the issue of the SE/MP trade-off using two examples from our own research, we (1) review the 33 identified articles that assess the trade-off, (2) summarize the proposed solutions for the tradeoff, and (3) discuss the findings for SSE and future work. Overall, we see evidence for the SE/MP trade-off and first solution approaches. However, a strong empirical foundation has yet to be established; general quantitative metrics and methods supporting software developers in addressing the trade-off have to be developed. We foresee considerable future work in SSE across scientific communities.

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/7839468/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=7839468

Comments: I found how the paper focuses on trying to balance Scalability and Efficiency "SE" with Maintainability and Portability "MP" to achieve high hardware efficiency. However to get this high hardware efficiency it states code needs tailoring to the specific hardware, therefore losing the maintainability and portability aspects. This links very well with my other research papers but on a larger and more broad scale.

Paper 3

Title: Object-oriented and classical software engineering

Citation: [?]

Full text link: https://worayoot.files.wordpress.com/2012/10/object-oriented-and-classical-software-engine

Comments: Goes through basics of code reuse and obstacles programmers may come across. Explaining the benefits of portable code. A broad source like this helps cover the basics.

Paper 4

Title: Software engineering with reusable components

Citation: [?]

Abstract: The book provides a clear understanding of what software reuse is, where the problems are, what benefits to expect, the activities, and its different forms. The reader is also given an overview of what sofware components are, different kinds of components and compositions, a taxonomy thereof, and examples of successful component reuse. An introduction to software engineering and software process models is also provided.

Web link: http://dl.acm.org/citation.cfm?id=1965498

Full text link: http://www.hostemostel.com/software/41.pdf

Comments: Links with Schach's book of code re-purposing covering requirements with code between interfaces and the economical aspects of reusing code and what the impacts are of making code reusable. Covering another aspect of my essay question while still relating to others.

Paper 5

Title: Portable software for multiprocessor systems

Citation: [?]

Abstract: The authors describe Prelude, a programming language and accompanying system support for writing portable parallel programs for multiprocessor architectures. Prelude allows the programmer to separate the description of the computation to be performed by a program from the description of how that computation is to be mapped onto a machine. This makes it easier to tune the performance of a program on a particular machine and also simplifies porting a program to new architectures.

Web link: http://ieeexplore.ieee.org/document/180499

Full text link: http://www.waldspurger.org/carl/papers/portable-mp-compctl-nov92.pdf

Comments: Different approach rather than making code reusable similar to other papers. Instead this explores how parallel programming can make code optimized for different hardware. Using the language Prelude.

Paper 6

Title: Application Portability in Cloud Computing: An Abstraction-Driven Perspective

Citation: [?]

Abstract: Cloud computing has changed the way organizations create, manage, and evolve their applications. While the abundance of computing resources at low cost opens up many possibilities for migrating applications to the cloud, this migration also comes at a price. Cloud applications, in many cases, depend on certain provider specific features or services. In moving applications to the cloud, application developers face the challenge of balancing these dependencies to avoid vendor lock-in. We present an abstraction-driven approach to address the application portability issues and focus on the application development process. We also present our theoretical basis and experience in two practical projects where we have applied the abstraction-driven approach.

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/6497434/?part=1

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=6497434

Comments: This paper describes issues faced when transferring application to a cloud gaming service. Another way programming may be faced with portability problems.

Paper 7

Title: Practices and Technologies in Computer Game Software Engineering

Citation: [?]

Abstract: Computer games are rich, complex, and often large-scale software applications. They're a significant, interesting, and often compelling domain for innovative research in software engineering techniques and technologies. Computer games are progressively changing the everyday world in many positive ways. Game developers, whether focusing on entertainment market opportunities or game-based applications in nonentertainment domains such as education, healthcare, defense, or scientific research (that is, serious games), thus share a common interest in how best to engineer game software. This article examines techniques and technologies that inform contemporary computer game software engineering.

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/7819395/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=7819395

Comments: A more broad paper on industry programming techniques can help me explain why some problems may not be that easily solved. This paper also cover code repurposing.

Paper 8

Title: Software reuse in robotics: Enabling portability in the face of diversity

Citation: [?]

Abstract: Computer games are rich, complex, and often large-scale software applications. They're a significant, interesting, and often compelling domain for innovative research in software engineering techniques and technologies. Computer games are progressively changing the everyday world in many positive ways. Game developers, whether focusing on entertainment market opportunities or game-based applications in nonentertainment domains such as education, healthcare, defense, or scientific research (that is, serious games), thus share a common interest in how best to engineer game software. This article examines techniques and technologies that inform contemporary computer game software engineering.

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/1438043/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=1438043

Comments: Describes the portability of code throughout robots with different purposes and hardware limitations. The paper explores the limitations between using code specific for hardware and coding for abstract purposes.

Paper 9

Title: Efficient kernel synthesis for performance portable programming

Citation: [?]

Abstract: The diversity of microarchitecture designs in heterogeneous computing systems allows programs to achieve high performance and energy efficiency, but results in substantial software re-development cost for each type or generation of hardware. To mitigate this cost, a performance portable programming system is required. One fundamental difference between architectures that makes performance portability challenging is the hierarchical organization of their computing elements. To address this challenge, we introduce TANGRAM, a kernel synthesis framework that composes architecture-neutral computations and composition rules into high-performance kernels customized for different architectural hierarchies. TANGRAM is based on an extensible architectural model that can be used to specify a variety of architectures. This model is coupled with a generic design space exploration and composition algorithm that can generate multiple composition plans for any specified architecture. A custom code generator then compiles these plans for the target architecture while performing various optimizations such as data placement and tuning. We show that code synthesized by TANGRAM for different types and generations of devices achieves no less than 70 percent of the performance of highly optimized vendor libraries such as Intel MKL and NVIDIA CUBLAS/CUSPARSE.

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/7783715/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=7783715

Comments: Paper goes over software trying to keep up with hardware advances and how software has to be redeveloped. This relates to the economic aspect i wanted to cover as part of my question. It goes over a program that can ”generate multiple composition plans for any specified architecture”

Paper 10

Title: Software reuse across robotic platforms: limiting the effects of diversity

Citation: [?]

Abstract:

Web link: Give the URL of the paper in IEEE Xplore, ACM Digital Library, or similar

Full text link: Give the URL of a downloadable PDF of the paper, if you can find one

Comments: Write a few sentences on how you found the article and why you believe it is relevant and/or important.