Simulating Cardinal Movements of Human Labour Using Finite Elements

Zelimkhan Gerikhanov

Supervisor: Dr. Joost Noppen

School of Computing Sciences

University of East Anglia

Upgrade Report

January 2015

Contents

List of Figures

List of Tables

Statement of Originality

Chapter 1

Introduction

Computer based simulations find numerous applications in medicine. Such applications include training of medical personell, diagnosing patients based on digital data and scientific research to gain better understanding of physiological phenomena. One of the most challenging applications of computer based simulation in medicine is bio-mechanical simuations. The underlying principles are very complex and thus require sophisticated theoretical formulations and considerable software development efforts.

Computer simulations provide a good tool for representing real world phenomena, but they are only capable of representing the simulated objects to a certain degree of approximation. Better approximations are predominantly much more expensive in terms of computational power. With the increased processing power of modern computers, it is possible to perform simuations with a higher degree of fidelity. However, even the most performant machines can struggle with certain types of high-cost simulations. In such cases, we have to utilize the underlying hardware to the highest degree possible. This can be achieved by a number optimization techniques. One of the most effective techniques is using parallel processing in order to speed up the computation.

It is desired to achieve the highest fidelity of the simulaitons with the as little latency as possible. Therefore, performance optimizations and GPGPU utilization for Finite Element Analysis is one of the main focuses of this thesis. Several available application programming interfaces (API's) will be overviewed as the candidates for the implementation. It is then shown how the chosen API is used to achieve highly efficient implementation of FEA for soft-tissue simulation.

Another important aspect of creating a computer based simulation of childbirth is acquiring realistic 3D models of the underlying physiological structures. Namely, the fetal body and maternal lower body geometries are required. The possible ways of constructing the required meshes will be covered in this thesis. The approach is not yet decided on.

The remainder of this report is arranged as follows. In chapter 2 a literature review is conducted presenting the body of already existing relevant research. Chapter 3 narrates the .(a paper covering work performed in benchmarking reverse engineering is also included in Appendix **??**, with details of the benchmark process and tooling in Appendix **??**). Future work is identified in chapter 4 in-

cluding a work plan and Gantt chart. Finally chapter 5 presents a proposed thesis structure for the final write-up.

Chapter 2

Literature Review

2.1 Introduction to Literature Review

The literature review in this report contains an overview of the key literature related to work conducted so far in the Traceability Forensics Project. Specifically this is a brief overview of traceability and requirements traceability (section **??**), reverse engineering of source code (section **??**) and mining of source code repository data (section **??**). Clustering techniques potentially offer the ability to determine semantic relationships and links between software components as an aid to comprehension and so is covered in section **??**. Some of these sections are limited in scope for brevity (traceability) or because the topic is a relatively newly investigated part of the project (clustering and to a lesser extent repository mining).

Literature was primarily found through searches of related keywords in Google scholar and DBLP, along with forwards and backwards citation analysis in addition to targeting specific conference proceedings. It is foreseen that the literature review for the PhD thesis will be significantly more detailed within the sections included in this report as well as containing a number of additional topics.

Chapter 3

Methodology

3.1 Overview of Work to Date

Chapter 4

Future Work

This chapter outlines future work to be performed as part of the project, split into different areas of focus. Specific research questions (RQs) are identified along with associated sub-research questions (SRQs). Later areas of research do not yet have specifically defined research questions, which will be determined nearer to the time.

4.1 Reverse Engineering Output

Complete integration of reverse engineering work within D-UEA-ST framework with respect to re-projection of UML within UMLet, specifically integrate relationships into the projection. Also investigate the possibility of using time-based evolution with reverse engineering. Look further into the gaps found in poor-performing reverse engineering software.

RQ. Is there any advantage to providing a sequential series of reverse engineering "snapshots" linked to the development of the software over time (reconstructed at points from source code repositories)?

RQ. For "missing information" in poor reverse engineering, is this the same common information between different reverse engineering applications?

SRQ. Can we build a more complete structural picture from multiple poor pictures e.g. can we combine output from more than one tool and form a more complete output?

4.2 Co-Commit Information

Refine co-commit toolchain to replace IBDOOS inclusion for efficiency. Test this technique with repositories other than git. Analyse the output generated from Co-Comm data to quantify its usefulness as applied to traceability reconstruction.

RQ. Does co-commit data show "sensible" (and useful) relationships?

RQ. What is the correlation between relationships identified through co-commit information and those through reverse engineering?

SRQ. Can relationships identified through reverse engineering be augmented through co-commit relationships?

4.3 Clustering of Related Components

Continue to investigate clustering techniques and their application to the data generated from different sources. Specifically work to refine the EM model and find appropriate settings to cluster software components, ideally at different levels such as sub-clusters.

RQ. What clustering techniques are most suited to software components?

RQ. Can clustering provide a more useful picture of relationships between components than purely similarity/dissimilarity numbers?

RQ. Can different data sources be combined to better inform clustering either pre-cluster (combined similarity matrices) or post-cluster (average of cluster memberships)?

RQ. Can clustering be used to determine parts of software architectural styles such as model-view-controller?

4.4 Other Sources of Information

Investigate other potential sources of information and methods to integrate them into existing toolchains.

4.4.1 Requirements Documentation

In association with an MSc project in 2013/14 perform textual analysis of requirements documentation (using techniques such as natural language processing). Link documentation to components within software (possibly performing transformation to allow source-code to be textually examined).

4.4.2 Technical Documentation and Code Comments

Investigate integration of more specific technical documentation (API documentation etc), most likely in a similar method to requirement documentation.

Work to integrate source code comments (such as Javadoc but also more generalised comments) into visualisation and/or description of software at a model level, and as potential indicators to relationship between requirements and implementation.

4.4.3 Runtime Relationship/Dependency Information

Investigate the possibility of incorporating run-time information such as call traces into augmenting relationship information.

RQ. Is effective runtime analysis possible in larger applications?

SRQ. Can collection of runtime information be automated in terms of collection of state types?

RQ. Does runtime relationship analysis show different relationships to other information sources such as reverse engineering?

SRQ. What is the overlap between runtime and other information sources relationships?

RQ. Can runtime information be usefully integrated to provide additional information?

4.4.4 Existing Software Library

Investigate the possibility of building a library of existing software for which certain information is known, for example for which previous manual analysis has been performed. This library would consist of not just source code and resultant analysis but also a set of abstracted structural representations at different levels which may be used for pattern matching.

4.5 Reasoning Component

Work towards the development of a reasoning component to augment decision making in terms of detection of components and links between levels of information. This will be a semi-manual process involving a weighted set of data sources being provided (possibly analysed for accuracy through a visualisation application). All types of data input will be supported e.g. reverse engineering, co-commit data, clustering output etc. The infrastructure will be extensible allowing for inclusion of more data sources as they become available. The reasoning component will then aim to identify certain attributes of the system under analysis for example through architectural style detection and classification. A series of outputs would be generated each with associated justification and traceability documentation demonstrating why a particular feature or style was detected and based on what input information. Weights could then be changed and further data sources added to increase accuracy. The conceptual process of the reasoning component is shown in figure 4.1.

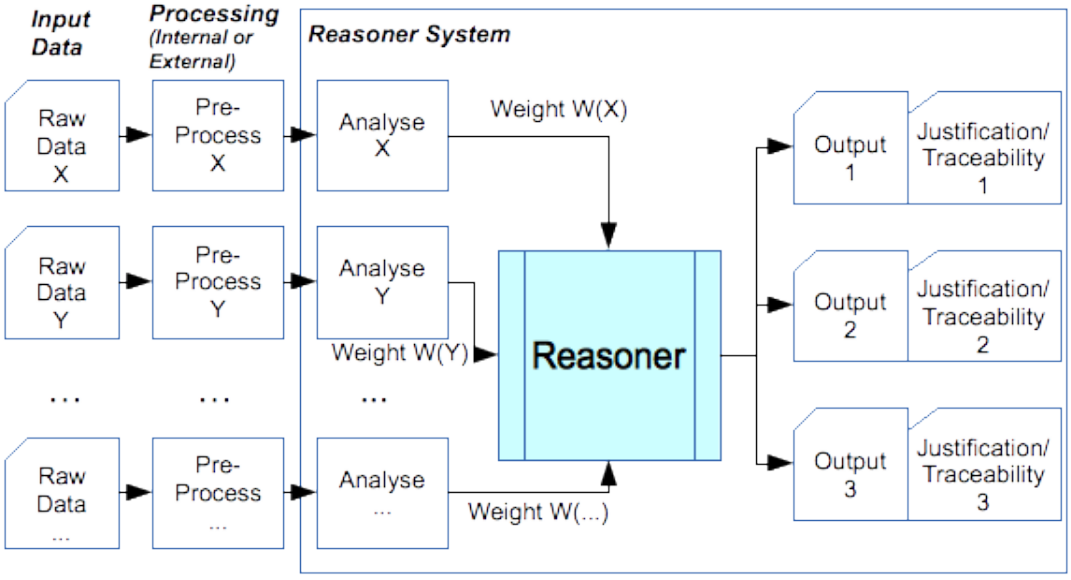

Figure 4.1: Conceptual Diagram of Reasoning Component

4.6 Visualisation and Developer Information

Work towards an integrated application aimed at:

- Visualising software structures through utilisation of many different sources of information
- Provide information to developers focused on change-impact e.g. for component A what other components are shown to be closely related and on what basis?

It is envisioned this will be an application (either standalone or more likely integrated within the D-UEA-ST framework) with which a developer can add multiple sources of information and may include the reasoning component (or at least output from the component). After an analysis has been automatically performed the developer will then be able to explore the software through an interface incorporating UML and textual representations, with more information on dependencies, relationships, and links to documentation provided.

4.7 Requirements Traceability Reconstruction

Using the above techniques investigate methods of rebuilding requirement links within the software (and feed this back into the process for example including requirement links within the visualiser).

4.8 Work Plan

The work plan is presented as a Gantt chart in figure 4.2, including a one month contingency period and one month holiday in August 2014.

4.9 Possible Future Papers

The following is a list of *possible* potential papers from the future work. Whether they will be prepared (represent worthwhile findings) and the exact scope and

size of the work will be decided as the project progresses. Potential targets are shown in *(bracketed italics)* with the definitions of the conferences below.

- Commonalities Between Reverse Engineering and Repository Mining in Software Component Relationships *(SANER, MSR)*
- Building Complete Reverse Engineering Pictures from Multiple Partial or Failed Reverse Engineering Output *(SANER, ECOOP, FASE, ACE)*
- Augmenting Reverse Engineering with Additional Data Sources for Comprehensive Understanding *(SANER, ECOOP, FASE, ACE)*
- Visualising Software Relationships using Multiple Information Sources *(FASE, ACE)*
- Reasoning Software Structure in a Semi-Automated Fashion *(FASE, ACE)*
- Comparison of Structural Relationships Detected by Reverse Engineering, Repository Mining, and Runtime Analysis *(SANER, MSR, FASE, ACE)*
- Rebuilding Traceability Links from Multiple Sources of Information *(FASE, ACE, ECOOP)*
- Traceability Between Requirements Documentation, Technical Documentation, and Source Code Analysis *(FASE, ACE, ECOOP)*

Targets for these potential papers would include:

- SANER: Merge of Working Conference on Reverence Engineering (WCRE) and European Conference on Software Maintenance and Re-engineering (CSMR)
- ECOOP: European Conference on Object-Orientated Programming
- FASE: Conference on the Fundamental Approaches to Software Engineering
- ACE: ACE! Conference on Software Development
- MSR: Conference on the Mining of Software Repositories

Figure 4.2: Project Gantt Chart

Chapter 5

Proposed Thesis Structure

1. Introduction
2. Literature Review
   (a) Literature Review Introduction
   (b) Traceability
   (c) Reverse Engineering
   (d) Repository Mining
   (e) Clustering
   (f) Software Visualisation
   (g) Natural Language Processing
   (h) Information Retrieval and Match of Multi-Source Data
      i. Textual Conversion of Source Code
      ii. Natural Language Processing
      iii. Pipeline for Comparison
3. Possible Sources of Information to Inform Traceability Forensics
   An analysis of possible sources of information drawing from the literature review; feeding into the concrete sources detailed in the following chapters
4. Framework for Comparison of Multiple Information Sources
   Detailing the specification and creation of a framework allowing for analysis using the multiple sources of information identified in the following chapters. How the workflow will be based, what tools are required/developed, what was found during their development.
5. Reverse Engineering
   (a) Benchmarking of Reverse Engineering Tools
   (b) Working with Reverse Engineering Output
   (c) Failure of Reverse Engineering
   (d) Constructing from Multiple Imperfect Sources

6. Source Code Mining
    (a) Clustering of Semantic Relationships
    (b) Evaluation of Approach
    (c) Working Further with Repository Information
7. Requirement Specification Analysis and Matching
8. Further Information Sources for Software Understanding
    (a) Dynamic Analysis (Stack Trace)
    (b) ... (more sources here as available)
9. Multi-Variate Integration, Normalisation, Clustering and Comparison of Information Sources
10. Identification of Software Architectural Styles and Component Groups
11. Aiding Comprehension of Software from Multiple Information Sources
    (a) Reasoning Component (Automated Analysis)
    (b) User-Informed Decision Making
    (c) Visualisation of Multi-Variate Software Information
12. Rebuilding Traceability Links from Multiple Information Sources
13. Evaluation of Methods
14. Conclusion

Bibliography

Chapter 6

Appendices

6.1 BirthView application screenshots