Thesis Project Ideas

Fabrizio Montesi
⟨fmontesi@itu.dk⟩
IT University of Copenhagen, Denmark
Office 4C14, Rued Langgaards Vej 7, 2300 Copenhagen

October 22, 2013

Contents

1 Introduction **2**

2 Jolie **2**
- 2.1 The Jolie Public Service Repository (JPSR) 2
- 2.2 JSOA 2
- 2.3 Crash Recovery 3
- 2.4 New communication protocols 3
- 2.5 Use case - Oncology workflow 3
- 2.6 Use case - Democratic Elections 3

3 π-calculus **4**
- 3.1 FreePi 4

4 Choreographic Programming **4**
- 4.1 Local Computations 4
- 4.2 Use cases 5

1 Introduction

In this document I report some project ideas that I am interested in following as supervisor of B.Sc. or M.Sc. theses. They are mostly related to the area of *distributed computing* and *programming languages*. Each project is described only briefly; please contact me if you would like to have more information.

I am also open to hearing new suggestions, so if you think that you have a cool idea near to my research area and would like my opinion on that, feel free to contact me.

Some projects are listed as suitable for both B.Sc. or M.Sc. theses. This means that the project has tasks of different difficulties, to be selected depending on the level of the thesis.

2 Jolie

Jolie is a general-purpose programming language for programming distributed systems. You can find more information about Jolie at the Jolie website [6]. A more academic description of Jolie is given in [10]. An interesting feature of Jolie is that its language model is mathematically formalised (see [9]), enabling the direct application of formal methods.

Jolie is used in 5 different universities for teaching and research; it is also adopted in the industry for the development of distributed systems [1].

2.1 The Jolie Public Service Repository (JPSR)

Level:	B.Sc. or M.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

The objective is to develop a public repository of Jolie services that can be accessed for free by registered users, through API keys. This would help new Jolie adopters to easily write powerful programs by composing such already available online services. The repository should offer a software platform that allows maintainers to create, publish and manage new online services through the official Jolie website [6].

The results of the project would be published on the Jolie website and then used for teaching (in academia and industry).

2.2 JSOA

Level:	B.Sc. or M.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

The objective is to extend JSOA, an open-source project for cloud computing based on the Jolie language. JSOA offers a web interface for uploading new services, running them, and monitoring them. There are many extensions needed, e.g., a distributed architecture for authenticating users, multi-tenant database management, and an extension to the Jolie language itself for better handling communications between services in a same virtual machine.

2.3 Crash Recovery

Level:	M.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

Sometimes, a Jolie program may be unexpectedly closed for external reasons, such as an operating system malfunction or a hardware problem. In such cases, the execution state of the program is lost. This is rather inconvenient in a distributed setting, since the program may have been in the middle of processing the responses for some client invocations.

The aim of this project is to develop a crash recovery mechanism for Jolie. If a program crashes, then the next time such program is executed it should be made aware of the fact and offered a way to recover from the situation. Concretely, this project would require modifying the Jolie interpreter and possibly the programming language itself.

2.4 New communication protocols

Level:	B.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

Jolie comes with an API for extending the language to support new communication protocols and data formats. Examples of protocols that we would be interested in implementing are Google SPDY and WebSockets. We are particularly open to new suggestions.

2.5 Use case - Oncology workflow

Level:	B.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

The aim of the project is to implement a use case from the Danish healthcare system, in which some hospital employees need to coordinate for assigning medications to patients. The main task would be to translate a formal description of the use case, given in Dynamic Condition Response graphs (DCR graphs) [7], to Jolie. Optionally, the thesis may also explore automatic ways of translating DCR graphs to Jolie code.

2.6 Use case - Democratic Elections

Level:	B.Sc. or M.Sc.
Project Type:	Software Development
Suitable for a group:	Yes

The aim of the project is to implement a use case on democratic elections, where a distributed system takes care of managing ballots and their secure transmission. This project will involve the usage of high-end cryptographic techniques for securing data while preserving its verifiability.

3 π-calculus

The π-calculus is a seminal formal model for describing distributed systems [8]. I am particularly interested in projects that mix the π-calculus with session types, a theory for the specification of network protocols [4, 5].

3.1 FreePi

Level:	M.Sc.
Project Type:	Formal Models and Software Development
Suitable for a group:	No

The aim of this project is to extend a toy implementation of an Eclipse-based IDE for programming π-calculus programs. The tool, to be developed in Scala, should support type checking programs with session types, allowing developers to ensure that their programs follow some given protocol specifications. We are particularly interested in implementing a language inspired by the calculus proposed in [11].

An optional milestone for the project would be to integrate the IDE with existing tools for deadlock-freedom analysis.

The developed tool would be used for teaching and research. The π-calculus is a very abstract language, which makes it ideal for designing the communication flows inside distributed systems.

4 Choreographic Programming

Choreographic Programming is an emerging paradigm for designing distributed systems, where programmers write a *choreography* that describes how two or more processes interact by exchanging messages during execution, as opposed to a collection of programs that define individually the actions performed by each process [12]. It is an exciting new paradigm for two main reasons: (i) programs are easier to understand, due to a more succint syntactic representation of communications; and (ii) choreographies guarantee important safety properties by design, e.g., freedom from deadlocks and race conditions, which are usually very hard to provide.

A paper describing the theory behind choreographic programming is [2]. We are particularly interested in Chor, a choreographic programming language equipped with an Eclipse-based IDE [3].

4.1 Local Computations

Level:	B.Sc. or M.Sc.
Project Type:	Formal Models and Software Development
Suitable for a group:	No

Chor is still limited to very simple local computations, e.g., variable assignments supporting integers or strings. The aim of this project is to extend Chor to the usage of more complex data structures, such as arrays and trees. Moreover, we wish to develop new language primitives to integrate Chor programs with other languages, such as Jolie, so to allow choreographies to build on existing programs given as external services (web or otherwise).

4.2 Use cases

Level:	B.Sc.
Project Type:	Software Development
Suitable for a group:	No

We are interested in the development of use cases with the Chor programming languages. The aim of the project would be to evaluate the Chor language by adopting it for implementing a complex distributed system for, e.g., distributed authentication or parallel algorithms.

References

[1] italianaSoftware s.r.l. `http://www.italianasoftware.com/`.

[2] M. Carbone and F. Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. In *POPL*, pages 263–274, 2013. Available online at `http://www.itu.dk/people/fabr/papers/multichor`.

[3] Chor. Programming Language. `http://www.chor-lang.org/`.

[4] K. Honda, V. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138, Heidelberg, Germany, 1998. Springer-Verlag.

[5] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, volume 43(1), pages 273–284. ACM, 2008.

[6] Jolie. Programming Language. `http://www.jolie-lang.org/`.

[7] K. M. Lyng, T. T. Hildebrandt, and R. R. Mukkamala. From paper based clinical practice guidelines to declarative workflow management. In *Business Process Management Workshops*, pages 336–347, 2008.

[8] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, Sept. 1992.

[9] F. Montesi and M. Carbone. Programming services with correlation sets. In *ICSOC*, pages 125–141, 2011. Available online at `http://www.itu.dk/people/fabr/papers/icsoc2011`.

[10] F. Montesi, C. Guidi, and G. Zavattaro. *Web Services Foundations*, chapter Service-oriented Programming with Jolie. Springer-Verlag, 2014. Available online at `http://www.itu.dk/people/fabr/papers/soc_jolie`.

[11] V. T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.

[12] W3C WS-CDL Working Group. Web services choreography description language version 1.0. http://www.w3.org/TR/ws-cdl-10/, 2004.