· U C ·

MSc in Informatics Engineering

Entermediate Report

Evaluate the robustness of Cloud

Gonçalo Silva Pereira

Supervisor	Raul Barbosa
Co-Supervisor	Henrique Madeira

Department of Informatics Engineering
University of Coimbra

March 23, 2015

Dedication

Acknowledgements

I would like to thank to —— and to professors Raul Barbosa and Henrique Madeira, who are role models, by their support and help to take good decisions.

Thank my girlfriend for their support, understanding and the fellowship along this path. At my friends and colleagues of Department of Informatics Engineering the patience, support and for all times they have given me.

Last but certainly not least, I would like to thank my family for encouragement, love and all the unconditional and constant support that led me to fulfill this dream. Obrigado!

Gonçalo Silva Pereira

“ Bridges are normally built on-time, on-budget, and do not fall down. On the other hand, software never comes in on-time or on-budget. In addition, it always breaks down.

Alfred Z. Spector, Google Research ”

“ "I have no special talents. I am only passionately curious."

Albert Einstein ”

Contents

List of Figures

List of Tables

Abbreviations

DDOS	Distributed Denial of Service
IaaS	Infrastructure-as-a-Service
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service

Abstract

Inject faults in code of applications, compile them and evaluate the results.
The theme of the dissertation is “Evaluate the robustness of Cloud”.

This thesis/dissertation presents an ????
Keywords: Faults, Errors, Failures, Vulnerabilities, Fault Injection, Fault Tolerance, Security, Robustness.

1 Introduction

1.1 Contextualization

The present dissertation describes the work developed in scope of MSc in Informatics Engineering. It is focused in “Evaluate the robusteness of Cloud” and this is one subject very important nowadays, because of the increase usage of these services. This services are characterized by the placement of data and software on remote infrastructure. Despite of the numerous benefits, the reliability of these platforms has not kept the needs, and users trust their applications to systems outside of personal control.

In this context, naturally arises the problem of confidence in the entity that manages the platform where applications have been executed. Any organization that put an application in the cloud (for example, Microsoft Azure or Amazon EC2) will have to accept the assurances given by the service provider.

This internship deals with the challenge of assessing the robustness of cloud platforms. The computing service provider uses virtualization to manage and allocate computing power to meet actual needs of the application. Although, there are solid virtualization platforms, fault tolerance is still a research problem.

1.2 The project

This project is based essentially in inject software faults. It was decided to inject software faults, since there are already other people involved in the part of hardware faults.

1.3 Objectives

The main objective of this work is to build a tool to inject software faults in code of some programs before the compilation.

But this main objective is divided in some other goals:

- Generate derivations of main code of selected programs;
- Verify and analyze the effect of produced faults;
- Compile the programs with injected faults, by using make file.

1.4 Document Structure

This document is ...

In the second section ...

In the third section ...
Finally, in last section ...
dasdasdsadsdsda

1.5 Management

In this section is described the planning of work developed in stage.

1.5.1 Meetings

In relation a meetings, the supervisor Raul Barbosa and me agreed that meet weekly was the best option. And the meetings were going on, with one or another change of schedule to reconcile with the other activities of both. In addition, I went to several general meeting of the project. Where could discuss concepts and the direction of the project with colleagues and teachers, among them: Raul Barbosa (supervisor), Henrique Madeira (co-supervisor), João Durães and João André Ferro.

1.5.2 Risks

The main-risks of execution of this project are:

- Equipment Failure
- Data lost
- Publication of similar research
- Personal issues interfere with progress
- Student loses interest
- Dispute between student and supervisor
- Supervisor takes excessive time to check final drafts
- Student wants to submit thesis without supervisor approval

The preventative measures and recovery measures can be seen at Appendix C.

1.5.3 Planning and Tracking

In Appendix A, is presented the gantt with the planning tasks to first and to second semester.

I have prioritized the tasks using the nomenclature in Table 1:

Classification	**Mean**
Must	Must be implement at project finish, and his implementation is priority.
Nice	May be part of the functionality implemented at the end of project, and his implementation is optional.
Wishful	It's specified but his implementation is not expected until the end of project.

Table 1: Classification of requirements

About the development of this project, I have used an *Agile Life Cycle* based in a *Incremental Model*.

What is the requirements of this project???

2 State of the Art

Nowadays, people use lot's of services based in cloud and lot's of companies choose to use them too. Using it, companies reduce the costs of IT infrastructure and people don't buy "physical storage" and don't care where are the data. The cloud service provide that the data is secure. But, like any system, the cloud have problems such as another computer systems, software and hardware faults. And the resilience of the cloud is an important characteristic.

The increased use of cloud is related with a low usage of many dedicated servers, lower voltage levels, reduce noise margins and increase clock rates [1].

The cloud providers offers resources ready to deliver [1].

With this work, I want to inject software faults and analyze how the system react to them.

A lot of studies show that the software faults it's the main cause of computer failures.

In this work deliberate how

I had access to the application of Robert Natella, called SAFE, that inject software faults, as I also have to do.

[2] [3]

About 44% of the software faults cannot be emulated [4].

2.1 SAFE - Robert Natella

3 Research objectives and approach method

3.1 GCC Parser vs CDT Parser vs Bison

3.2 Cloud Computing

Three levels of Cloud Computing:

- Infrastructure-as-a-Service (IaaS);
- Platform-as-a-Service (PaaS);
- Software-as-a-Service (SaaS).

The cloud computing isn't free of external disturbances[1], the most importants are:

- Security attacks;
- Accidents;
- Power surges;
- Workload faults;
- Malfunction;
- Worms
- Distributed Denial of Service (DDOS) attacks.

3.3 Applications to inject faults

4 Current work and preliminary results

5 Work plan and implications

Built three separated modules:

- Generate the derivations of main code of selected programs;
- Verify and analyze the effect of produced faults;
- Compile the programs with injected faults, by using make file.

5.1 Generate derivations

I chose to use the most representative faults [2], divided into missing, wrong and extraneous, specified individually further down:

5.2 Contraints

The contraints defined below was specified by João Durães in ... .

Constraint	Description
C01	Return value of the function must not being used
C02	Call must not be the only statement in the block
C03	Variable must be inside stack frame
C04	Must be the first assignment for that variable in the module
C05	Assignment must not be inside a loop
C06	Assignment must not be part of a for construct
C07	Must not be the first assignment for that variable in the module
C08	The if construct must not be associated to an else construct
C09	Statements must not include more than five statemens and not include loops
C10	Statements are in the same block, do not include more than 5 stats. nor loops
C11	There must be at leat two variables in this module

5.2.1 Fault Types - Missing:

- **MIFS** - if construct plus statements

  This operator is based in the remotion of one conditional if. To do that, i need to verify the constraints **??**

- **MLAC** - AND sub-expr in expression used as branch condition

- **MFC** - function call

- **MIA** - if construct around statements
- **MLOC** - OR sub-expr in expression used as branch condition
- **MLPA** - small and localized part of the algorithm
- **MVAE** - variable assignment using an expression
- **MFCT** - functionality
- **MVAV** - variable assignment using an value
- **MIEB** - if construct plus statements plus else before statements
- **MVIV** - variable initialization

5.2.2 Fault Types - Wrong:

- **WLEC** - logical expression used as branch condition
- **WALL** - algorithm - large modifications
- **WVAV** - value assigned to variable
- **WAEP** - arithmetic expression in parameter of function call
- **WSUT** - data types or conversion used
- **WPFV** - variable used in parameter of function call

5.2.3 Fault Types - Extraneous:

- **EVAV** - variable assignment using another variable

5.3 Analyze the effects

The fault injected results is equal to the real software faults?

5.4 Compile programs

Select five to ten programs to test.

6 Conclusion

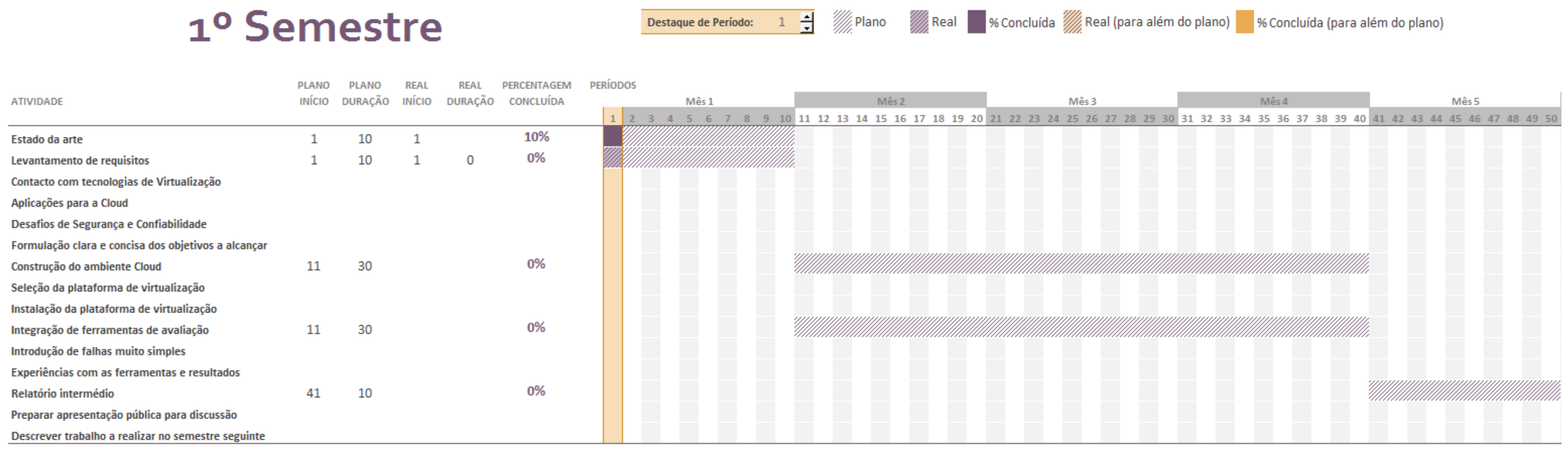
6.1 Global Vision

Global Vision

6.2 Future Work

Future Work

A Appendix A - Gantt diagrams



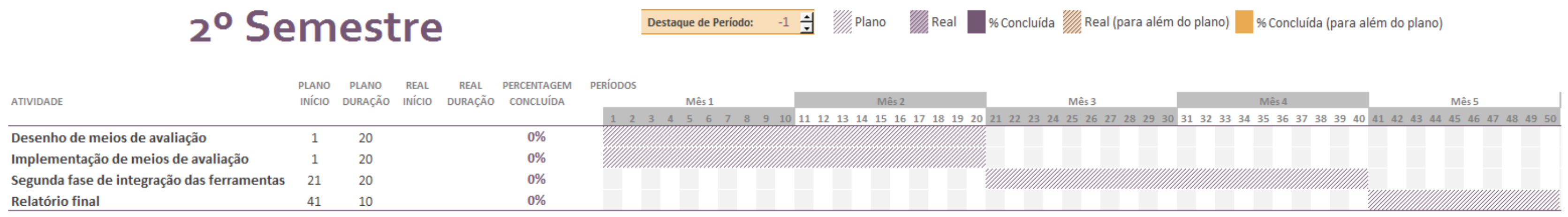
1º Semestre

Destaque de Período: 1

▨ Plano ▨ Real ■ % Concluída ▨ Real (para além do plano) ■ % Concluída (para além do plano)

ATIVIDADE	PLANO INÍCIO	PLANO DURAÇÃO	REAL INÍCIO	REAL DURAÇÃO	PERCENTAGEM CONCLUÍDA
Estado da arte	1	10	1		10%
Levantamento de requisitos	1	10	1	0	0%
Contacto com tecnologias de Virtualização					
Aplicações para a Cloud					
Desafios de Segurança e Confiabilidade					
Formulação clara e concisa dos objetivos a alcançar					
Construção do ambiente Cloud	11	30			0%
Seleção da plataforma de virtualização					
Instalação da plataforma de virtualização					
Integração de ferramentas de avaliação	11	30			0%
Introdução de falhas muito simples					
Experiências com as ferramentas e resultados					
Relatório intermédio	41	10			0%
Preparar apresentação pública para discussão					
Descrever trabalho a realizar no semestre seguinte					

Figure 1: *First semester gantt.*



2º Semestre

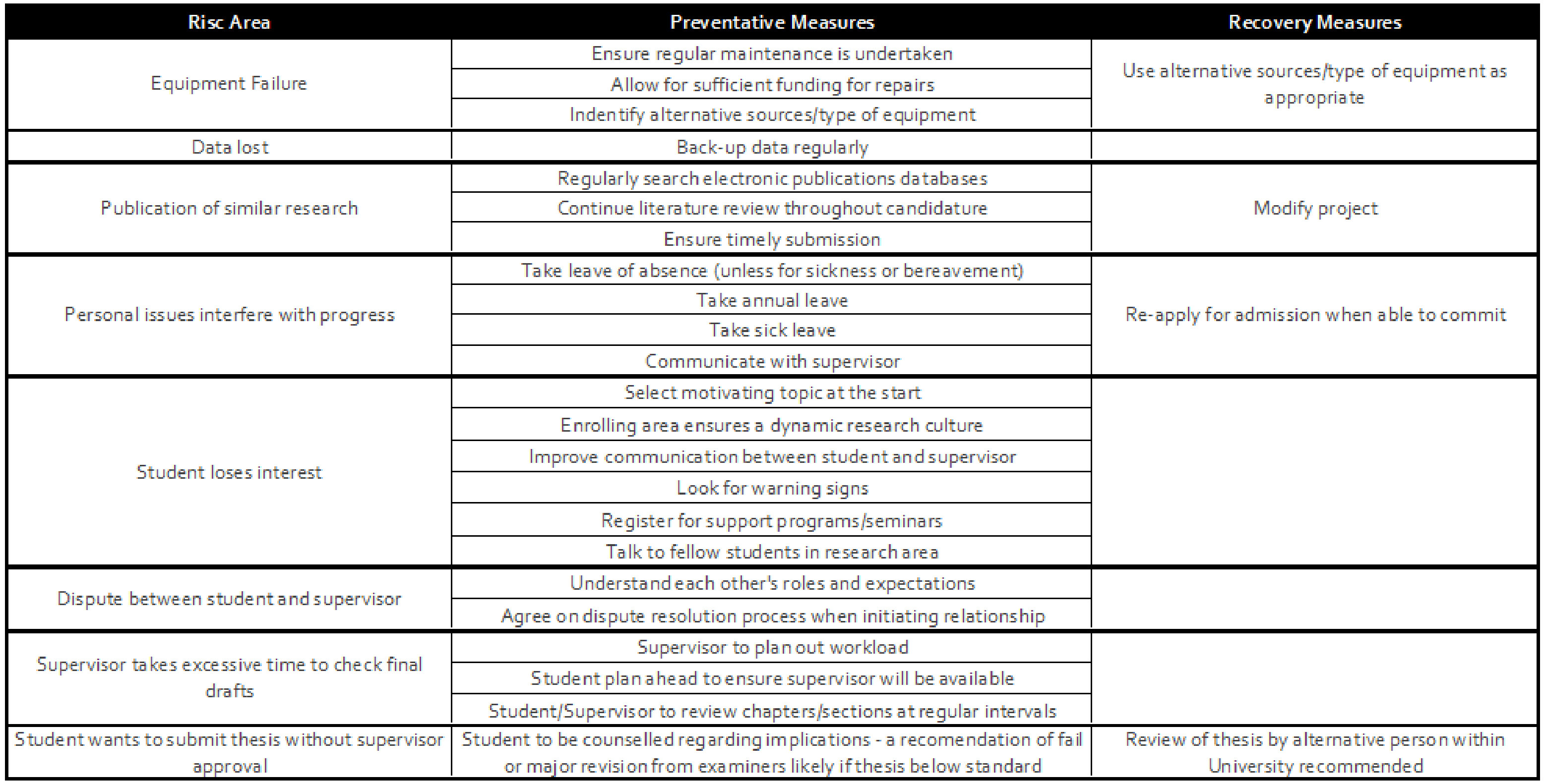
Destaque de Período: -1

▨ Plano ▨ Real ■ % Concluída ▨ Real (para além do plano) ■ % Concluída (para além do plano)

ATIVIDADE	PLANO INÍCIO	PLANO DURAÇÃO	REAL INÍCIO	REAL DURAÇÃO	PERCENTAGEM CONCLUÍDA	PERÍODOS Mês 1										Mês 2										Mês 3										Mês 4										Mês 5									
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Desenho de meios de avaliação	1	20			0%	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																														
Implementação de meios de avaliação	1	20			0%	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																														
Segunda fase de integração das ferramentas	21	20			0%																					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨										
Relatório final	41	10			0%																																									▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Figure 2: *Second semester gantt.*

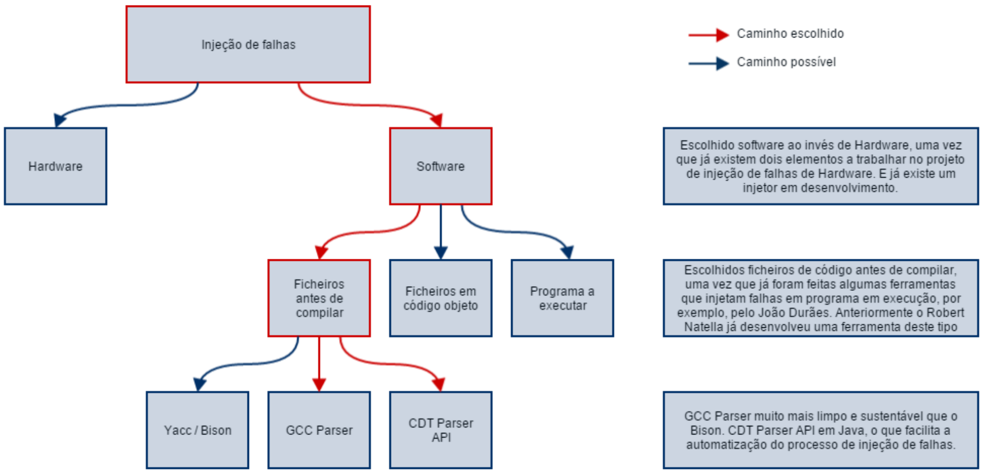
B Appendix B - Risks table



Risc Area	Preventative Measures	Recovery Measures
Equipment Failure	Ensure regular maintenance is undertaken	Use alternative sources/type of equipment as appropriate
	Allow for sufficient funding for repairs	
	Indentify alternative sources/type of equipment	
Data lost	Back-up data regularly	
Publication of similar research	Regularly search electronic publications databases	Modify project
	Continue literature review throughout candidature	
	Ensure timely submission	
Personal issues interfere with progress	Take leave of absence (unless for sickness or bereavement)	Re-apply for admission when able to commit
	Take annual leave	
	Take sick leave	
	Communicate with supervisor	
Student loses interest	Select motivating topic at the start	
	Enrolling area ensures a dynamic research culture	
	Improve communication between student and supervisor	
	Look for warning signs	
	Register for support programs/seminars	
	Talk to fellow students in research area	
Dispute between student and supervisor	Understand each other's roles and expectations	
	Agree on dispute resolution process when initiating relationship	
Supervisor takes excessive time to check final drafts	Supervisor to plan out workload	
	Student plan ahead to ensure supervisor will be available	
	Student/Supervisor to review chapters/sections at regular intervals	
Student wants to submit thesis without supervisor approval	Student to be counselled regarding implications - a recomendation of fail or major revision from examiners likely if thesis below standard	Review of thesis by alternative person within University recommended

Figure 3: *Risks.*

C Appendix C - Decision Tree



Escolhido software ao invés de Hardware, uma vez que já existem dois elementos a trabalhar no projeto de injeção de falhas de Hardware. E já existe um injetor em desenvolvimento.

Escolhidos ficheiros de código antes de compilar, uma vez que já foram feitas algumas ferramentas que injetam falhas em programa em execução, por exemplo, pelo João Durães. Anteriormente o Robert Natella já desenvolveu uma ferramenta deste tipo

GCC Parser muito mais limpo e sustentável que o Bison. CDT Parser API em Java, o que facilita a automatização do processo de injeção de falhas.

Figure 4: *Decision Tree.*

References

[1] K. Wolter, A. Avritzer, M. Vieira, and A. van Moorsel, *Resilience assessment and evaluation of computing systems.* Springer, 2012.

[2] J. A. Duraes and H. S. Madeira, "Emulation of software faults: A field data study and a practical approach," *Software Engineering, IEEE Transactions on*, vol. 32, no. 11, pp. 849–867, 2006.

[3] A. Avizzienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing."

[4] H. Madeira, D. Costa, and M. Vieira, "On the emulation of software faults by software fault injection," in *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on.* IEEE, 2000, pp. 417–426.