PhD in Information Technology and Electrical Engineering

Università degli Studi di Napoli Federico II

PhD Student: Pietro Liguori

XXXIV Cycle

Training and Research Activities Report – First Year

Tutor: Domenico Cotroneo – co-Tutor: Roberto Natella

1. Information

PhD candidate: Pietro Liguori
Date of birth: 07/09/1989
Master Science title: Master's degree in Computer Engineering (cum laude), Universiy of Naples Federico II
Doctoral Cycle: XXXIV
Fellowship type: PhD student grant (Grant Type: Academic)
Tutors: Prof. Domenico Cotroneo, Prof. Roberto Natella
Year: First

I received my MS degree (cum laude) in Computer Engineering from the Univesità degli Studi di Napoli Federico II in July 2018.

My master thesis focused on anomaly detection in distributed and complex systems such as cloud computing infrastructures. To limit the massive number of false positives (due to the non-determinism of distributed systems), I developed a novel approach of anomaly detection based on a probabilistic model.

I'm currently in the first year of the Ph.D. program in Information Technology and Electrical Engineering (ITEE) at Federico II University of Naples, under the supervision of Prof. Domenico Cotroneo and Prof. Roberto Natella.

2. Study and Training activities

In this first year I attended the following courses and seminars.

Title	Type	Hours	Credits	Dates	Organizer	Certificate
Programmazione II	Master Science Course (ING-INF/05)		6	2nd semester 2018-2019	Università degli Studi di Napoli Federico II	Yes
MHD equilibrium and stability	Ad-hoc	5	1	February 4;7;8;11;12;15, 2019	Università degli Studi di Napoli Federico II	Yes
Data Science and Optimization	Ad-hoc	6	1,2	February 5;6;7, 2019	Università degli Studi di Napoli Federico II	Yes
Advanced Techniques for Software Robustness and Security Testing	Ad-hoc	15	3	January 16;30, February 14;20, March 7;19, April 2, 2019	Università degli Studi di Napoli Federico II	
Big Data	Ad-hoc	15	3	February 26;27, March 4;5;6, 2019	Università degli Studi di Napoli Federico II	
14th International School on Software Engineering	Doctoral School		3	June 17-21, 2019	Univesità degli Studi di Salerno	Yes
27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)	Conference	18	3,6	August 28-30, 2019	Software Engineering and Information Systems Group of University of Tartu's Institute of Computer	Yes

					Science	
15th European Dependable Computing Conference (Naples)	Conference	24	4,8	September 17-20, 2019	DESSERT group of University of Naples Federico II (UNINA)	Yes
Parallel and Distributed Computing with Matlab	Seminary	2	0,4	21 November, 2018	DIETI	Yes
How to publish a scientific paper	Seminary	2	0,4	26 November, 2018	DIETI	Yes
IEEEXplore Training and Authorship Workshop	Seminary	2,5	0,5	April 4, 2019	DIETI	Yes
Supervised Machine Learining	Seminary	2	0,4	May 6, 2019	DIETI	Yes
Support Vector Machines	Seminary	2	0,4	May 7, 2019	DIETI	Yes
Feature Design	Seminary	2	0,4	May 8, 2019	DIETI	Yes
Unsupervised Machine Learning	Seminary	2	0,4	May 9, 2019	DIETI	Yes
Neural Networks	Seminary	2	0,4	May 13, 2019	DIETI	Yes
From Shallow Networks to Deep Networks	Seminary	2	0,4	May 14, 2019	DIETI	Yes
Ensemble Methods	Seminary	2	0,4	May 15, 2019	DIETI	Yes
Recurrent Networks	Seminary	2	0,4	May 17, 2019	DIETI	Yes
Deep Learning in Matlab	Seminary	3	0,6	May 20, 2019	DIETI	Yes
Applying Semi-Supervised Learning to App Store Analysis	Seminary	2	0,4	July 12, 2019	DIETI	Yes
Comparison of Techniques for Dealing with Imbalance in Software Defect Prediction	Seminary	2	0,4	July 15, 2019	DIETI	Yes

	Credits year 1								Credits year 2	Credits year 3	
		1	2	3	4	5	6				
	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	Estimated	Check
Modules	25	0	2,2	6	9	3,6	4,8	**25,6**	10	0	30-70
Seminars	5	0,8	0	0,5	3,8	0,8	0	**5,9**	5	0	10-30
Research	30	9,2	7,8	3,5	0	5,6	5,2	**31,3**	45	60	80-140
	60	10	10	10	12,8	10	10	**62,8**	60	60	180

3. Research activity

Title: Anomaly Detection and Failure Mode Analysis in Cloud Computing Infrastructures

Description and Study

Cloud computing is becoming an attractive solution for running services with high-reliability requirements, such as in the telecom and healthcare domains.

However, cloud computing systems are often exposed to unpredictable failure conditions. These failures can propagate across several components or layers of the system (e.g., storage, virtual network, compute instances, etc.) in complex ways, leading to cascading effects (**failure propagation**) that make recovery actions more problematic. Therefore, identifying and analyzing failure propagation is an important activity to design more effective recovery actions.

Fault injection is a relevant approach, which emulates faults to anticipate worst-case scenarios, such as network partitions, high network latency, replica crashes, and I/O exceptions. Fault injection has reached a level of maturity that it is routinely used to reveal failures in real-world systems, including cloud computing software such as key-value data stores and distributed computing frameworks (e.g., Cassandra, ZooKeeper), entire cloud computing services (e.g., streaming services deployed by Netflix) and infrastructures (e.g., IaaS providers such as Amazon).

Nevertheless, there are still open issues for its adoption in cloud systems. Indeed, as the scale and the complexity of these systems increase, it becomes harder for developers to identify (and to analyze) failures that are triggered by fault injection. In the current approaches, the system designers write failure specifications before the experiments, and then they look for occurrences of these failures within the experimental data. For example, the most sophisticated approaches check formal specifications over events and outputs, such as finite state machines, temporal logic predicates, relational logic, and special-purpose languages.

Since these specifications are based on prior knowledge and past experience of system designers about failures, they are not meant for discovering new, unknown failure modes of a distributed system that are missed by the failure specifications. Moreover, writing failure specifications is a time-consuming and cumbersome task, which makes fault injection less applicable.

The current state of practice is to detect failures (e.g., service unavailability, performance degradation) by monitoring the quality of service during the fault injection test; more sophisticated solutions detect failures by monitoring properties expressed with formal specifications, such as finite state machines, relational logic, and special-purpose languages.

However, once a service failure has been triggered by fault injection and detected by monitoring mechanisms, a human analyst still needs to analyze the chain of events (e.g., messages) that occurred among the location where the fault/error is injected and the component that experiences the service failure. Yet, this failure analysis still relies on intuition and manual effort of the human analyst. Unfortunately, manual analysis is too difficult and time-consuming, because of:

- The **high volume of messages** generated by large distributed systems that the human analyst needs to scrutinize;

- The **non-determinism in distributed systems**, in which the timing and the order of messages can unpredictably change even if there is no failure, which introduces noise in the analysis, and increases the effort of the human analyst to pinpoint the failure (i.e., to discriminate the anomalies caused by a fault from genuine variations of the system);
- The use of **off-the-shelf software components**, either proprietary or open-source (such as application frameworks, middleware, data stores, etc.), whose events and protocols can be difficult to understand and to manually analyze.

My research proposal aims to provide automated support for analyzing failures triggered by fault injection in cloud computing systems. I aim to avoid the human analyst to manually inspect thousands of events, by automatically identifying the few relevant events that are related to the injected fault, while discarding noisy, uninteresting events. To this goal, I am working on an approach that extends fault injection, by combining it with black-box tracing and anomaly detection for failure analysis. The driving idea is to train a *probabilistic model* of the events in the distributed system under test under *fault-free* conditions, by using variable-order Markov Models for analyzing event sequences. Afterward, the system is tested with fault injection, and event traces are collected under these *faulty* conditions. The faulty event traces are analyzed with anomaly detection by using the probabilistic model, and the anomalous events are reported to the human analyst for understanding how to avoid failures.

I am also working on a **clustering** technique, built on top of the anomaly detection algorithm, for automatically identifying the classes of failure modes among the tests in a fault injection campaign.

I am experimentally evaluating the approach in the context of the OpenStack cloud management platform, which is the basis for many commercial cloud management products, and it is widespread both among public cloud infrastructure providers and private users.

4. Products

In this first year, I have produced the following products.

4.1 Publications

Conference Paper

1. Domenico Cotroneo, Luigi De Simone, Pietro Liguori, Roberto Natella, and Nematollah Bidokhti. "How Bad Can a Bug Get? An Empirical Analysis of Software Failures in the OpenStack Cloud Computing Platform". *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2019.
2. Domenico Cotroneo, Luigi De Simone, Pietro Liguori, Roberto Natella, and Nematollah Bidokhti. "FailViz: A Tool for Visualizing Fault Injection Experiments in Distributed Systems". *European Dependable Computing Conference (EDCC)*, 2019.
3. Domenico Cotroneo, Luigi De Simone, Pietro Liguori, Roberto Natella, and Nematollah Bidokhti. "Enhancing Failure Propagation Analysis in Cloud Computing Systems". *International Symposium on Software Reliability Engineering (ISSRE)*, 2019.

Student Forum

1. Pietro Liguori, Domenico Cotroneo and Roberto Natella, "Analyzing Fault Injection Data with Machine Learning". *European Dependable Computing Conference (EDCC)*, 2019

Poster Session

1. Poster Presentation at *European Dependable Computing Conference (EDCC)*, September 18, 2019.

4.2 Artifcats and Tools

1. *"OpenStack fault injection environment",* awarded with Reusable and Available badges at FSE Artifact Track. DOI: 10.6084/m9.figshare.8242877
2. *FailViz* (ongoing work), a tool for visualizing fault-injection experiments.

5. Conferences

I participated the following conference:

Conference name	Place	Dates	Number of accepted papers
27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)	Tallinn	August 28-30, 2019	74
15th European Dependable Computing Conference (EDCC)	Naples	September 17-20, 2019	24

As the author, I **presented** the following paper:

- *"How Bad Can a Bug Get? An Empirical Analysis of Software Failures in the OpenStack Cloud Computing Platform",* ESEC/FSE 2019, Tallinn
- *"Analyzing Fault Injection Data with Machine Learning",* EDCC 2019, Naples

6. Tutorship

In this first year I have been teaching assistant for the course of "Impianti di Elaborazione", ING-INF/05, a.a. 2018/2019