Automated GUI Testing Techniques for Android Applications

Dottorato di Ricerca in Ingegneria Informatica ed Automatica

XXVIII Ciclo – Terzo Anno 2015/2016

Candidato
Nicola Amatucci

Tutor
Prof. Anna Rita Fasolino
Prof. Porfirio Tramontana

Contesto

- **Applicazioni Mobili**
  - È un mercato ancora in forte crescita
  - Attrae sviluppatori professionisti e non per le basse barriere all'ingresso
- La **qualità delle applicazioni** è un aspetto rilevante sia per i programmatori sia per gli utenti
  - Un sondaggio di SmartBear ha rivelato che il 50% degli utenti rimuoverebbe un'app in presenza di bug
- Uno studio di Kochhar et al. ha mostrato che **su 627 apps open source** pubblicate su GitHub
  - il 14% ha casi di test eseguibili
  - il 4% ha casi di test che coprono più del 40% del codice
- Gli sviluppatori di app potrebbero beneficiare della disponibilità di tecniche di testing automatiche

Attività del primo e secondo anno

- **Ricerca di strumenti e tecniche di GUI testing automatico** proposte in letteratura per applicazioni Android
  - Sperimentazione con gli strumenti individuati al fine di effettuare un confronto tra strumenti
- **Implementazione di Android Ripper**
  - Sperimentazione con Android Ripper e confronto con gli strumenti individuati
  - Individuate problematiche legate alle tecniche di testing automatiche
- **Proposta di un criterio di arresto per tecniche Random**

Attività del terzo anno

- Approfondito lo **studio della letteratura** sulle tecniche di testing automatiche nell'ambito Android
- Proposta di un **framework generico** nell'ambito del quale possono essere descritte le tecniche individuate
  - Implementazione del framework
  - Sperimentazione per il confronto tra tecniche di testing
- Proposte di ulteriori tecniche per migliorare l'efficacia e l'efficienza del testing automatico

Tecniche di Testing Automatiche

- Tecniche di **Testing Online** (Takala et al.)
  - la fase di generazione e di esecuzione dei test è contestuale
    - **Active Learning** (Choi et al.)
      - Apprendono un modello dinamicamente e a partire da questo generano casi di test
    - **Random**

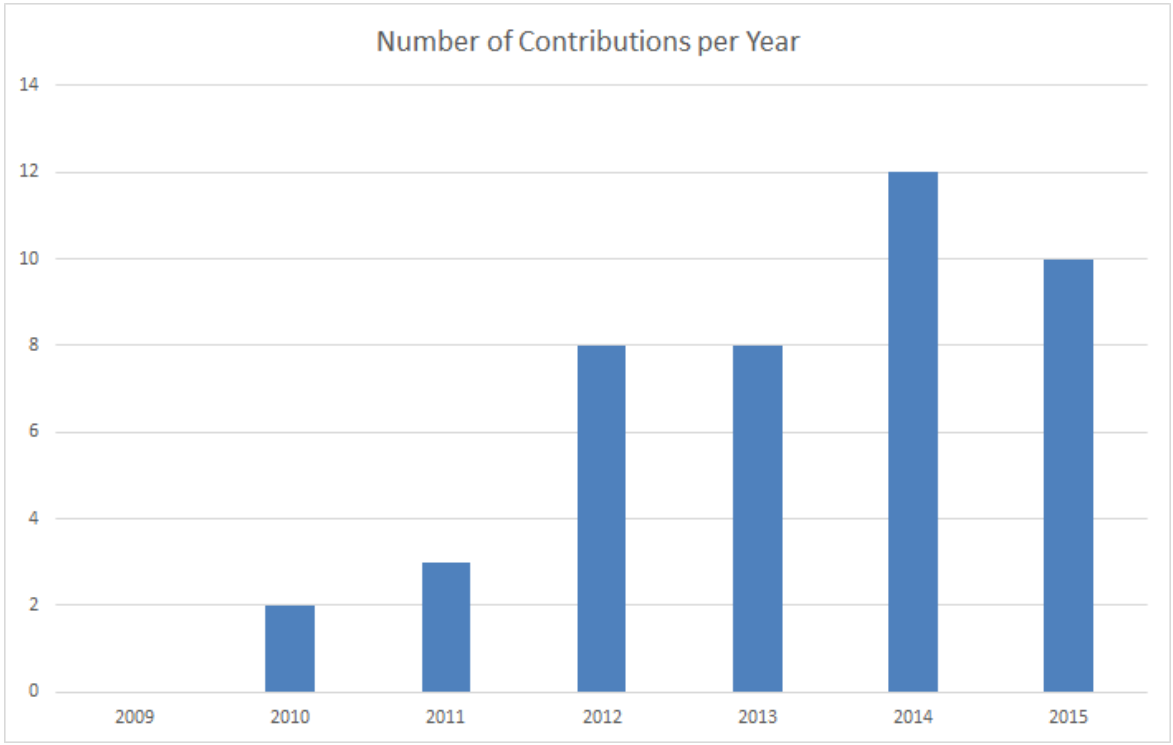
- **Ricerca Bibliografica**
  - Individuare contributi in letteratura riguardo tecniche Active Learning e Random

Ricerca Bibliografica

- **Repository**
  - ACM, IEEE, Scopus
- **Parole chiave**
  - «android testing» (4192 Risultati)
- **Filtraggio dei risultati**
  - Criteri di Inclusione (151 Risultati)
    - Risultati non duplicati, in lingua Inglese
    - Categoria: Engineering, Computer Science
    - Paper, Journal, Book Chapter
    - Titolo e Abstract
  - Lettura Approfondita dell'articolo (39 Risultati)

Ricerca Bibliografica

- **39 contributi individuati**
  - dal 2010 a Novembre 2015



Caratteristiche Comuni

- Le tecniche Active Learning e Random proposte in letteratura
  - Implementano un **processo iterativo**
  - Ottengono **informazioni sull'applicazione a tempo di esecuzione**
    - Se lo prevedono, queste informazioni vanno ad aggiornare un **modello**
  - **Schedulano** casi di test a partire dalle informazioni ottenute
  - Hanno **criteri di terminazione** simili

Algoritmo Generico

Algorithm 1 Unified Online Testing Algorithm

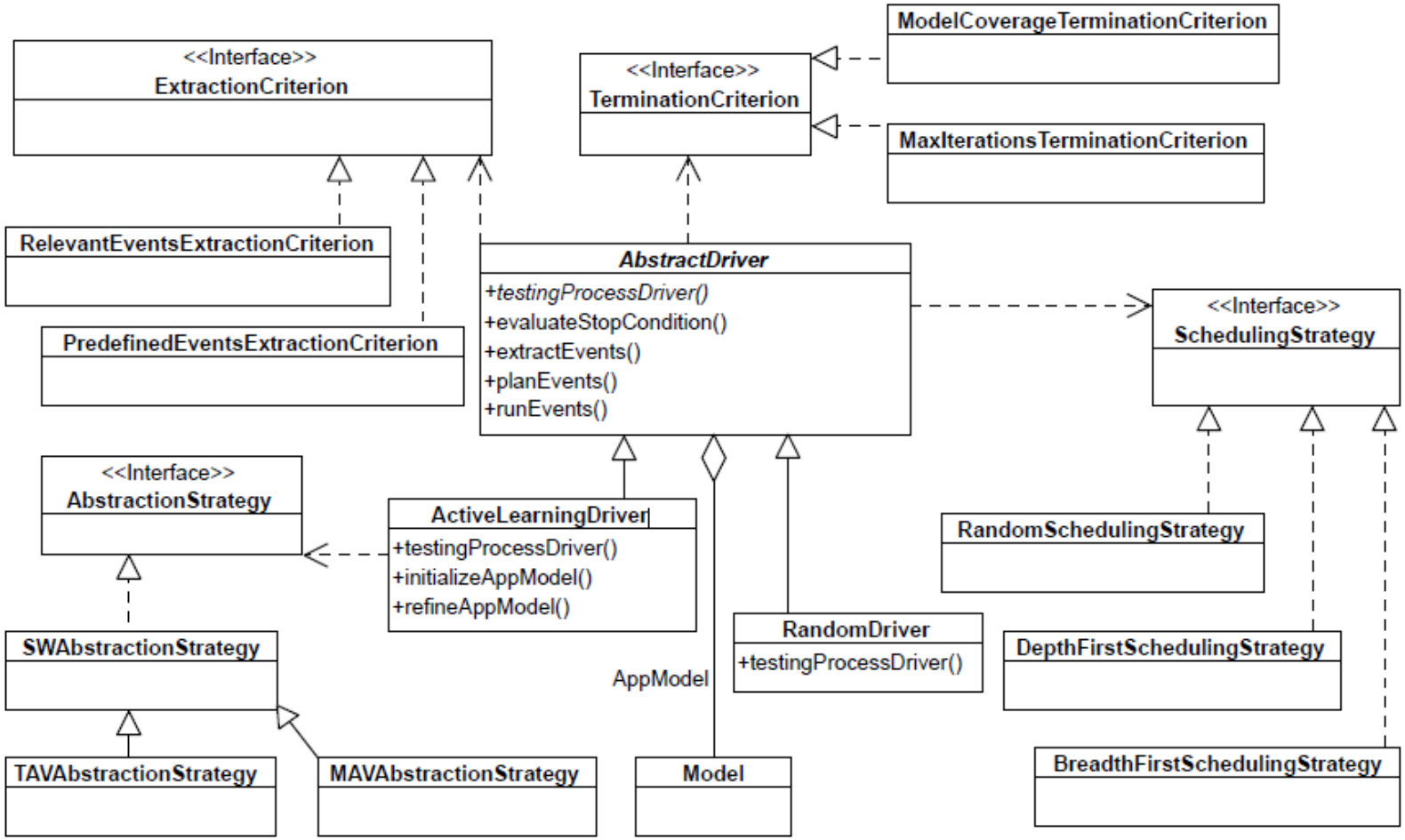
Require: TerminationCriterion, ExplorationStrategy, AbstractionStrategy, ExtractionCriterion, SchedulingStrategy

```
1: if (ExplorationStrategy == ActiveLearning) then
2:     AppModel ← initializeAppModel(AbstractionStrategy);
3: stopCondition ← EvaluateStopCondition(TerminationCriterion);
4: while (!stopCondition) do
5:     fireableEvents[] ← ExtractEvents(ExtractionCriterion)
6:     eventsSequence ← ScheduleEvents(fireableEvents[], SchedulingStrategy);
7:     RunEvents(eventsSequence);
8:     if (ExplorationStrategy == ActiveLearning) then
9:         AppModel ← RefineAppModel(AbstractionStrategy);
10:    stopCondition ← EvaluateStopCondition(TerminationCriterion);
```

Framework

Valori dei parametri associati ai contributi individuati

Implementazione di Android Ripper

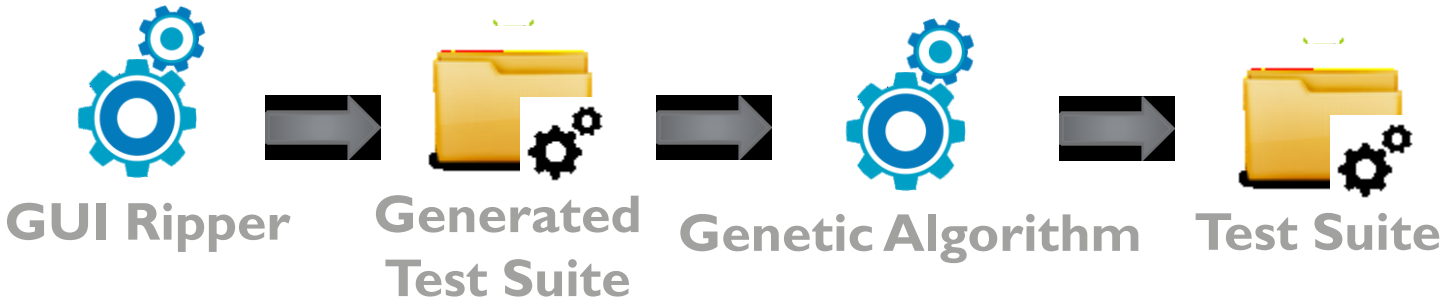


Lesson Learned

- Dalla sperimentazione con Android Ripper è emerso che tra le configurazioni considerate
  - Le tecniche Active Learning sono generalmente più efficienti delle tecniche Random
    - a meno di casi particolari in cui non si raggiunge una terminazione
  - Le tecniche Random sono sempre più efficaci delle tecniche Active Learning
    - Generano sequenze di eventi lunghe e ripetitive che possono generare particolari precondizioni
  - Il tempo di esecuzione del processo di Testing automatico in generale è elevato
- Spunti di ricerca
  - Introdurre casualità nelle tecniche Active Learning
  - Parallelizzazione del processo di Testing

AGRippin

- **A**ndroid **G**enetic **Ripping**
  - Active Learning Technique
  - Genetic Algorithm
- Lo scopo è di ottenere test suite più efficaci



Algoritmo Genetico

- **Rappresentazione**
  - Sequenze di Stati della GUI ed Eventi
- **Crossover**
  - Effettuato su Stati della GUI equivalenti (secondo un criterio definito)
  - Single-Point Crossover
- **Mutation**
  - Valori dei Widget Editabili
- **Fitness Evaluation**
  - Misura della diversità legata alla copertura del codice
- **Selection**
  - Rank-Based Selection
- **Combination Technique**
  - Esplorazione di Stati della GUI non appresi dalla tecnica Active Learning

Caso di Studio

- 5 applicazioni open source
- 6 esecuzioni di AGRippin

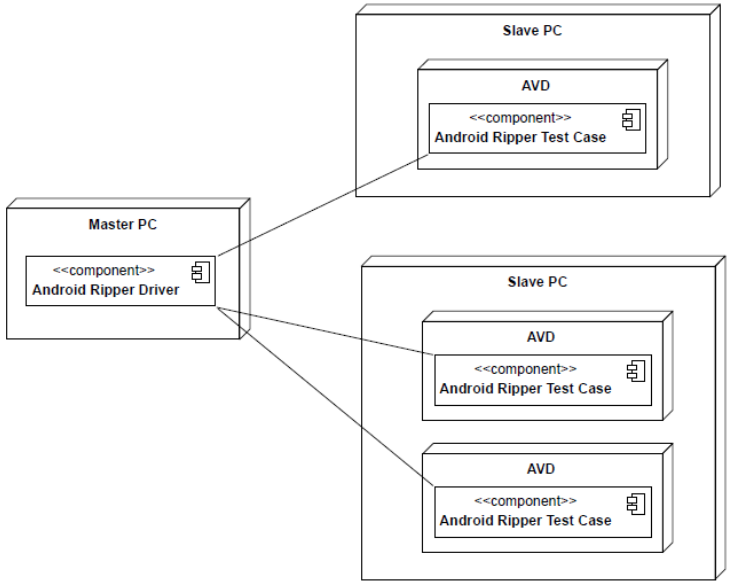
	GUI Ripper	AGRippin		
	COV	$\mu(COV)$	$\sigma(COV)$	Discovered Interfaces
AUT1	43.07%	67.10%	0.26%	1
AUT2	28.08%	32.61%	2.45%	0
AUT3	51.58%	58.31%	2.28%	0
AUT4	66.90%	68.00%	1.21%	0
AUT5	40.34%	47.22%	0.45%	1

- In ogni caso si è riscontrato un incremento dell'efficacia

Android Ripper Parallelo

- Android Ripper si basa su approccio **Master-Slave**
  - Un'implementazione dell'algoritmo generico (Master)
  - Un componente in esecuzione sul Device (Slave)
    - Si interfaccia con l'applicazione sotto test

- **Master-Multi/Slave**



Parallelizzazione delle Tecniche

- Tecniche Random
  - Più istanze del Master in esecuzione
    - Ogni Master controlla uno Slave
    - Unione dei risultati ottenuti

- Tecniche Active Learning
  - Problema
    - Aggiornamento del Modello
    - Gli accessi in lettura e scrittura al modello devono avvenire in mutua esclusione

Parallelizzazione delle Tecniche

- Tecniche Active Learning

Algorithm 1 Unified Online Testing Algorithm

Require: TerminationCriterion, ExplorationStrategy, AbstractionStrategy, ExtractionCriterion, SchedulingStrategy

```
1:  if (ExplorationStrategy == ActiveLearning) then
2:      AppModel ← initializeAppModel(AbstractionStrategy);
3:  stopCondition ← EvaluateStopCondition(TerminationCriterion);
4:  while (!stopCondition) do
5:      fireableEvents[] ← ExtractEvents(ExtractionCriterion)
6:      eventsSequence ← ScheduleEvents(fireableEvents[], SchedulingStrategy);
7:      RunEvents(eventsSequence);
8:      if (ExplorationStrategy == ActiveLearning) then
9:          AppModel ← RefineAppModel(AbstractionStrategy);
10:     stopCondition ← EvaluateStopCondition(TerminationCriterion);
```

Multi-Thread

Regione Critica

Sperimentazione Tecniche Active Learning

- Risultati ottenuti al variare di
  - Numero di Macchine (n_M)
  - Numero di Nodi Slave (n_S)

		(n_M, n_S)					
		(1,1)	(1,2)	(2,1)	(2,2)	(6,1)	(6,2)
AUT1	COV%	19	19	19	19	19	19
	t_{TOT}	00:19	00:13	00:11	00:08	00:07	00:04
AUT2	COV%	58	58	58	58	58	58
	t_{TOT}	01:02	00:45	00:43	00:32	00:19	00:14
AUT3	COV%	32	32	32	32	32	32
	t_{TOT}	02:29	01:45	01:11	00:49	00:33	00:20

Discussione dei risultati

- La copertura del codice non cambia
- I tempi di esecuzione diminuiscono
  - Passando da **n_S=1 ad n_S=2**, mantenendo n_M costante
    - si ha uno speed-up di un fattore **1.46**
  - Passando da **n_M=1 ad n_M=2**, mantenendo n_S costante
    - Si ha uno speed-up di un fattore **1.74**
  - Passando da **n_M=1 ad n_M=6**, mantenendo n_S costante
    - Si ha uno speed-up di un fattore **3.70**

Indice die capitoli della Tesi

1. Introduction
2. Background & Related Work
3. Android Ripper
4. Considering Context Events in Event-Based Testing of Mobile Applications
5. AGRippin: a novel search based testing technique for Android applications
6. Exploiting the saturation effect in automatic random testing of Android applications
7. A parallel and distributed implementation of GUI Ripping Techniques
8. Conclusions & Future Work

Bibliografia Essenziale

- P. Kochhar, F. Thung, N. Nagappan, T. Zimmermann, and D. Lo. Understanding the test automation culture of app developers. In Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on, pages 1-10, April 2015.
- T. Takala, M. Katara, and J. Harty. Experiences of system-level model-based gui testing of an android application. In Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on, pages 377-386, March 2011.
- H. Muccini, A. Di Francesco, and P. Esposito. Software testing of mobile applications: Challenges and future research directions. In Automation of Software Test (AST), 2012 7th International Workshop on, pages 29-35, June 2012.
- I. Banerjee, B. Nguyen, V. Garousi, and A. Memon. Graphical user interface (gui) testing: Systematic mapping and repository. Inf. Softw. Technol., 55(10):1679-1694, Oct. 2013.
- A. Memon, I. Banerjee, B. Nguyen, and B. Robbins. The fist decade of gui ripping: Extensions, applications, and broader impacts. In Proceedings of the 20th Working Conference on Reverse Engineering (WCRE). IEEE Press, 2013.
- H.-L. Wen, C.-H. Lin, T.-H. Hsieh, and C.-Z. Yang. Pats: A parallel gui test ing framework for android applications. In Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual, volume 2, pages 210-215, July 2015.

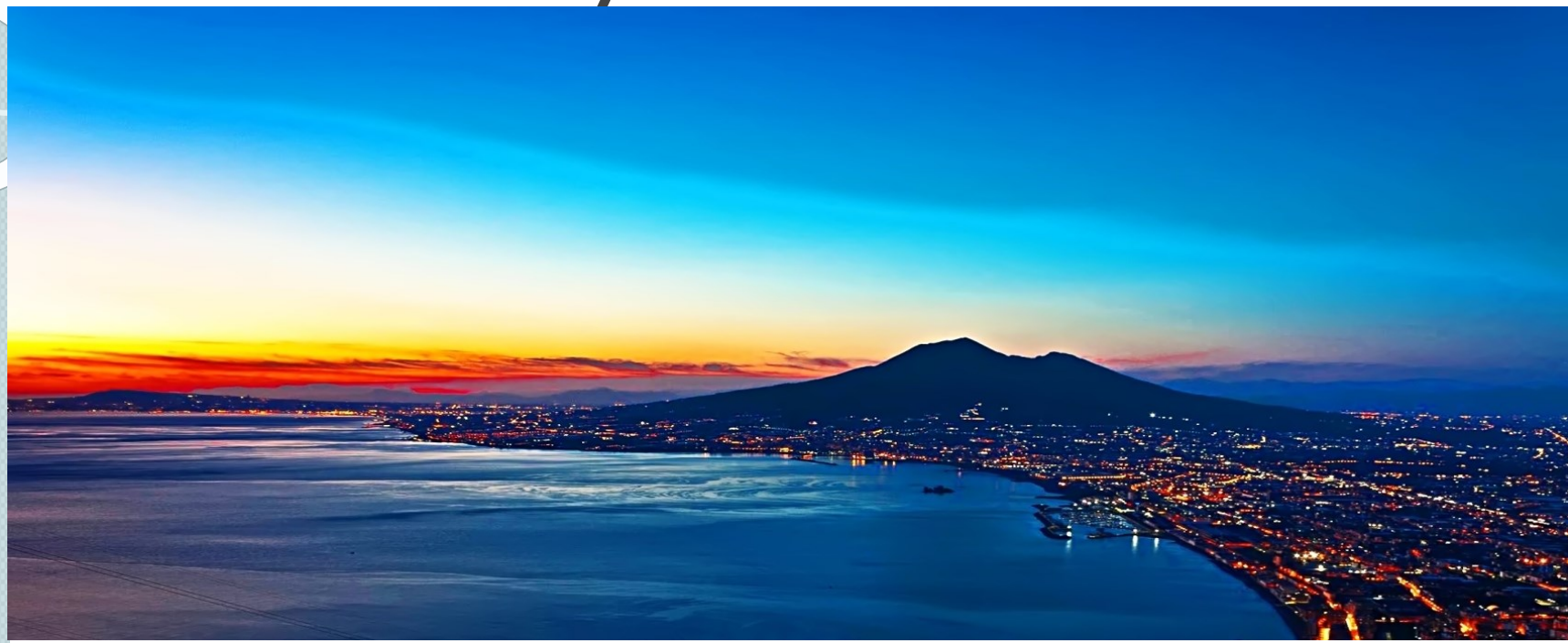
Pubblicazioni (Terzo Anno)

- "Exploiting the Saturation Effect in Automatic Random Testing of Android Applications." by Domenico Amalfitano, Nicola Amatucci, Anna Rita Fasolino, Porfirio Tramontana, Emily Kowalczyk, Atif Memon, in The Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft 2015), 2015.
- "Toward Reverse Engineering of VBA Based Excel Spreadsheets Applications.", Domenico Amalfitano, Nicola Amatucci, Vincenzo De Simone, Anna Rita Fasolino and Porfirio Tramontana, SEMS 2015
- "AGRippin: a novel search based testing technique for Android applications.", Domenico Amalfitano, Nicola Amatucci, Anna Rita Fasolino, Porfirio Tramontana, In The Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile (DeMobile 2015), 2015.
- "A conceptual framework for the comparison of fully automated gui testing techniques.", Domenico Amalfitano, Nicola Amatucci, Anna Rita Fasolino, Porfirio Tramontana, In The Proceedings of the Sixth International Workshop on Testing Techniques for Event BasED Software, 2015.

Attività di Formazione (Terzo Anno)

- Corsi di Dottorato
  - Modelli, metodi e software per l'ottimizzazione
    - Prof. Antonio Sforza - Prof. Claudio Sterle)
    - CFU Acquisiti: 4
  - The Entrepreneurial Analysis of Engineering Research Projects
    - Prof. Luca Iandoli
    - CFU Acquisiti: 3
- Seminari
  - Seminar on memory technologies for Android based systems
  - Model-Based and Pattern-Based GUI Testing
  - Verifica e Validazione di sistemi Safety Critical
  - Adversarial Testing of Protocol Implementations
- Conferenze
  - Workshop DeMobile 2015 – Bergamo
  - Presentato: "*AGRippin: a novel search based testing technique for Android applications*"

Thanks for your attention



Questions?

Further Information:

http://reverse.dieti.unina.it

@REvERSE_UNINA

nicola.amatucci@unina.it