Tampere University

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Cloud-based Testing

Summer School on Software Evolution: From Monolithic to Cloud-Native

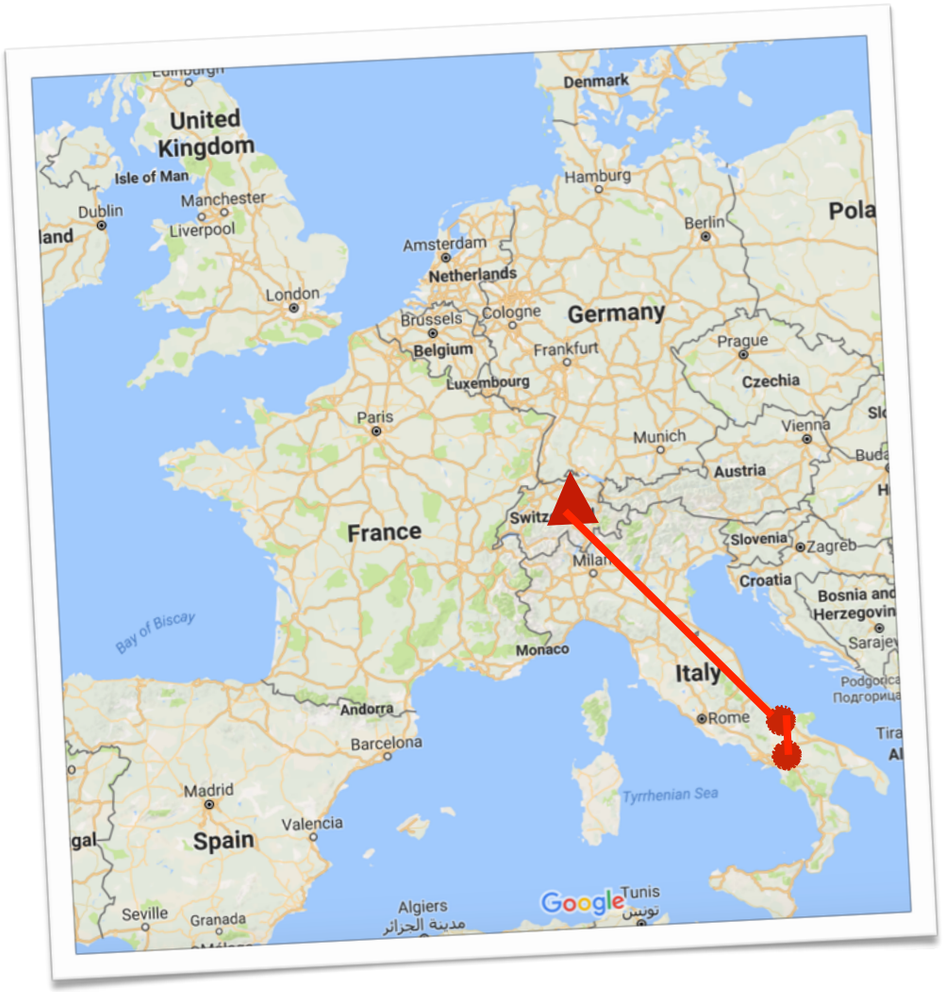
Sebastiano Panichella
Zurich University of Applied Science (ZHAW)
*Institute of Applied Information Technology (**InIT**)*

spanichella@gmail.com and panc@zhaw.ch

@spanichella

https://www.zhaw.ch/en/about-us/person/panc/ or https://spanichella.github.io/

My Background



Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

HIPPOCRATICA CIVITAS STUDIUM SALERNI

- Main topics
- Related topics
- Secondary topics

My Background

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

HIPPOCRATICA CIVITAS STUDIUM SALERNI

- Main topics
- Related topics
- Secondary topics

My Background

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

HIPPOCRATICA CIVITAS STUDIUM SALERNI

My Background

- Main topics
- Related topics
- Secondary topics

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

HIPPOCRATICA CIVITAS STUDIUM SALERNI

Program Comprehension & Maintenance (PC&M):
- Traceability Recovery
- Generation of source code documentation
- Profiling developers
- Dependencies analysis in software ecosystems

PhD thesis:
"Supporting Newcomers in Software Development Projects"

My Background

- Main topics
- Related topics
- Secondary topics

PC&M + Mobile computing (MC):

- Machine Learning & Genetic Algorithms
  - Defect Predictions
- Mobile computing
  - Mobile testing & User Feedback Analysis
    - (User Reviews + NLP analysis)
- Code Review & Continuous Delivery
  - Prioritization of Static Analysis Warning
- Summarization Techniques
  - for Code,
  - Change,
  - Testing and User Feedback

Program Comprehension & Maintenance (PC&M):

- Traceability Recovery
- Generation of source code documentation
- Profiling developers
- Dependencies analysis in software ecosystems

PhD thesis:
"Supporting Newcomers in Software Development Projects"

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zhaw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

My Background

- Main topics
- Related topics
- Secondary topics

PC&M&MC + Testing challenges:

- Test case generation and assessment
- User Feedback Analysis
- Generation of test code documentation
- Continuous Delivery
  - CD anti-patterns
  - Branch Coverage Prediction
  - Documentation defects detection

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zhaw School of Engineering

?

FNSNF ?

?

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

PC&M + Mobile computing (MC):

- Machine Learning & Genetic Algorithms
  - Defect Predictions
- Mobile computing
  - Mobile testing & User Feedback Analysis
    - (User Reviews + NLP analysis)
- Code Review & Continuous Delivery
  - Prioritization of Static Analysis Warning
- Summarization Techniques
  - for Code,
  - Change,
  - Testing and User Feedback

FNSNF

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

Program Comprehension & Maintenance (PC&M):

- Traceability Recovery
- Generation of source code documentation
- Profiling developers
- Dependencies analysis in software ecosystems

PhD thesis:
"Supporting Newcomers in Software Development Projects"

University of Salerno
Master Student
December 2010

HIPPOCRATICA CIVITAS STUDIUM SALERNI

My Background

- Main topics
- Related topics
- Secondary topics

PC&M&MC + Testing challenges:

- Test case generation and assessment
- User Feedback Analysis
- Generation of test code documentation
- Continuous Delivery
  - CD anti-patterns
  - Branch Coverage Prediction
  - Documentation defects detection

PC&M + Mobile computing (MC):

- Machine Learning & Genetic Algorithms
  - Defect Predictions
- Mobile computing
  - Mobile testing & User Feedback Analysis
    - (User Reviews + NLP analysis)
- Code Review & Continuous Delivery
  - Prioritization of Static Analysis Warning
- Summarization Techniques
  - for Code,
  - Change,
  - Testing and User Feedback

Program Comprehension & Maintenance (PC&M):

- Traceability Recovery
- Generation of source code documentation
- Profiling developers
- Dependencies analysis in software ecosystems

PhD thesis:
"Supporting Newcomers in Software Development Projects"

Zurich University of Applied Science
Senior Computer Science Researcher
Since August 2018

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

University of Zurich
Research Associate
October 2014 - August 2018

University of Zurich[UZH]

University of Sannio
PhD Student
June 2014

Università degli Studi del Sannio

University of Salerno
Master Student
December 2010

Outline

PART I

1) Research Context and Motivation

PART II

2) Literature review on:

- Testing Challenges in the Cloud
- Automated Testing
- Cloud-based testing demonstration

3) Future Work

Who Are You?

Bachelor student?

Master student?

Part-time student?

P.h.D. student?

Who Are You?

Bachelor student?

Master student?

Part-time student?

P.h.D. student?

Who Are You?

Talk a bit about your-self…

- short bio…

- why you decided to register to this summer-school?

- what do you want to achieve from it?

Outline

PART I

1) Research Context and Motivation

PART II

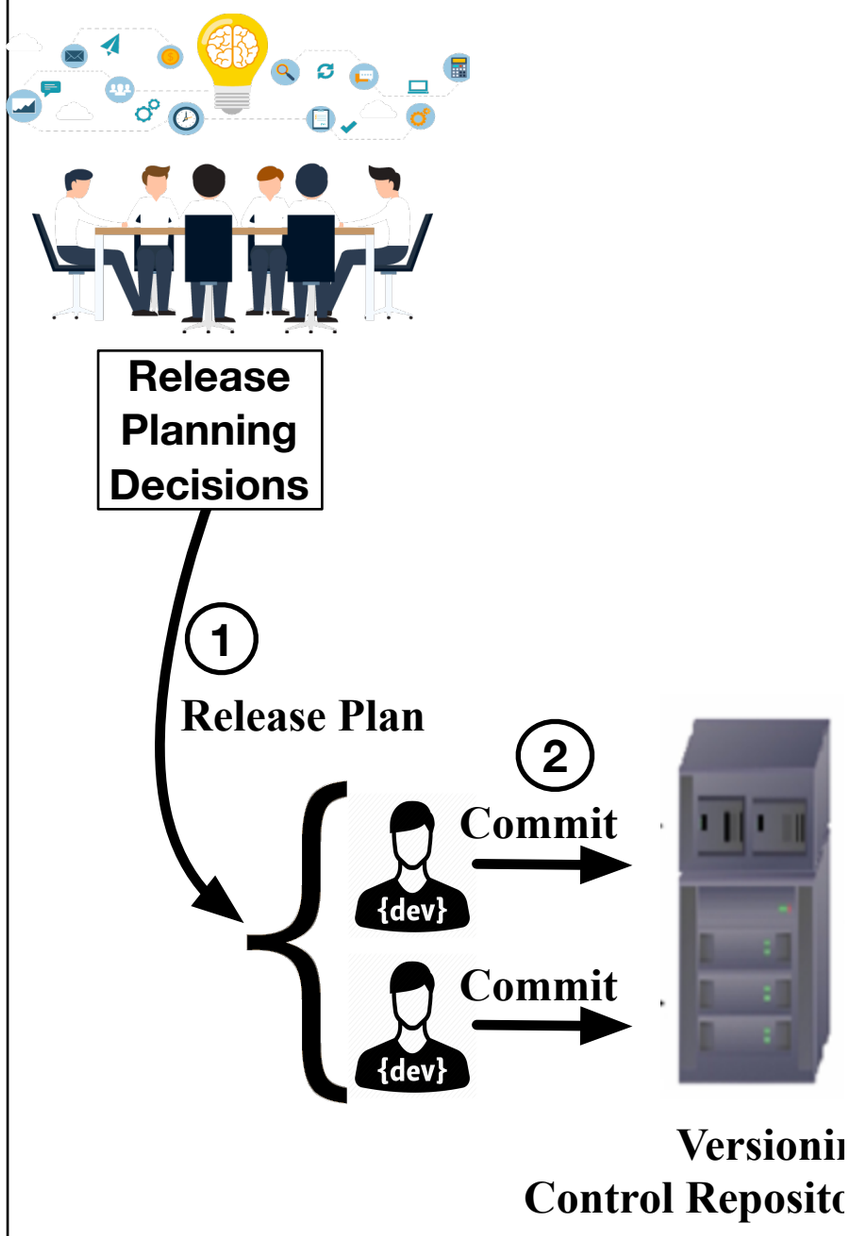
2) Literature review on:

- Testing Challenges in the Cloud
- Automated Testing
- Cloud-based testing demonstration

3) Future Work

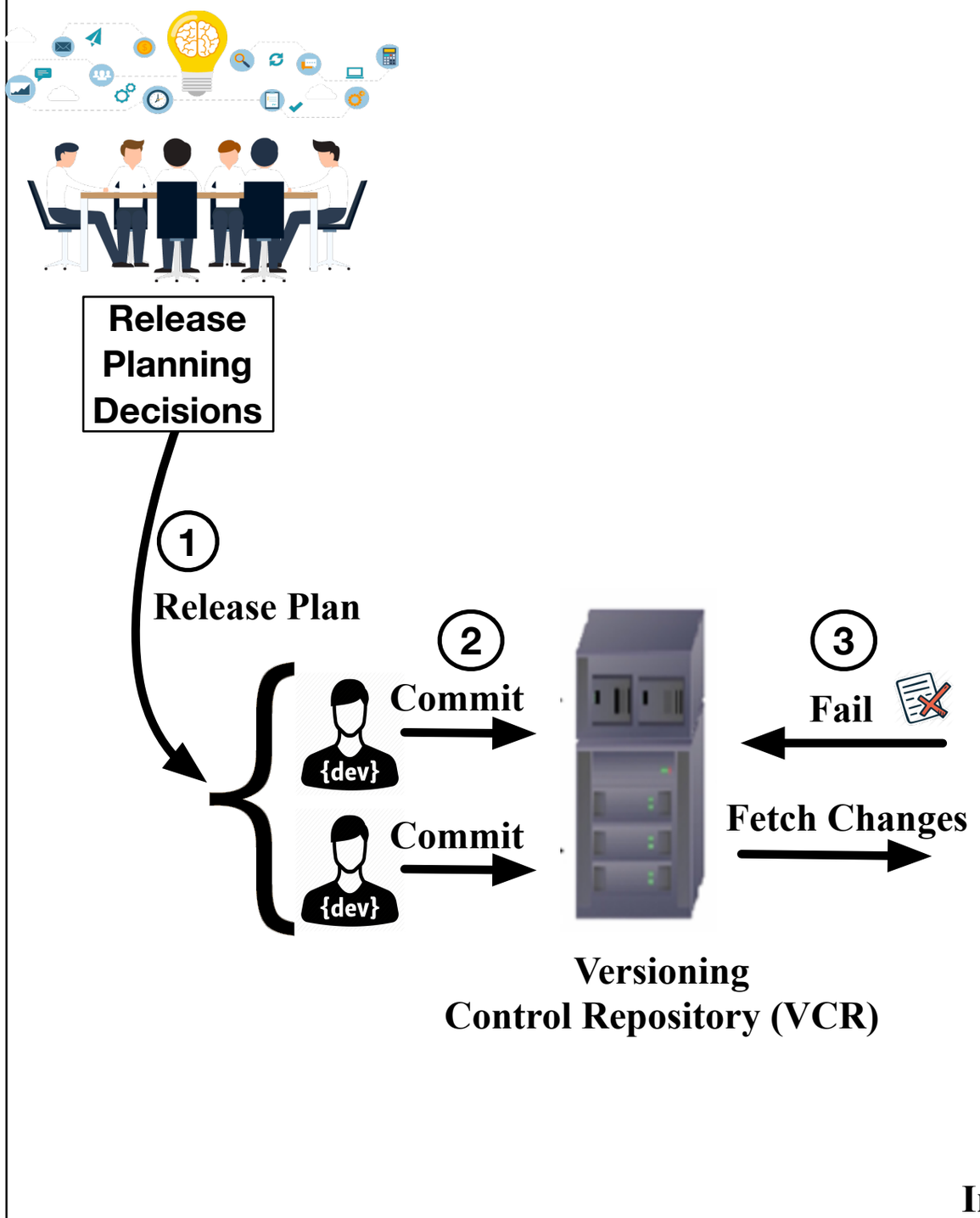
Research Context and Motivation

Simplified Development Process



Research Context and Motivation

Simplified Development Process



Research Context and Motivation

Simplified Development Process

Release Planning Decisions

1 Release Plan

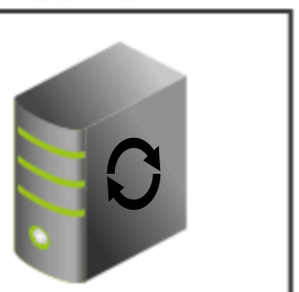
2 Commit {dev}

Commit {dev}

3 Fail

Fetch Changes

Versioning Control Repository (VCR)



Compile

Run unit tests

Run integration tests

Package

Build process

4 Success

Compile Code
Integrate DB
Running Tests
Deployment

Continuous Integration (CI) Server

Research Context and Motivation

Simplified Development Process

Release Planning Decisions

1 Release Plan

2 Commit

{dev}

Commit

{dev}

Versioning Control Repository (VCR)

3 Fail

Fetch Changes

Compile

Run unit tests

Run integration tests

Package

Build process

Continuous Integration (CI) Server

4 Success

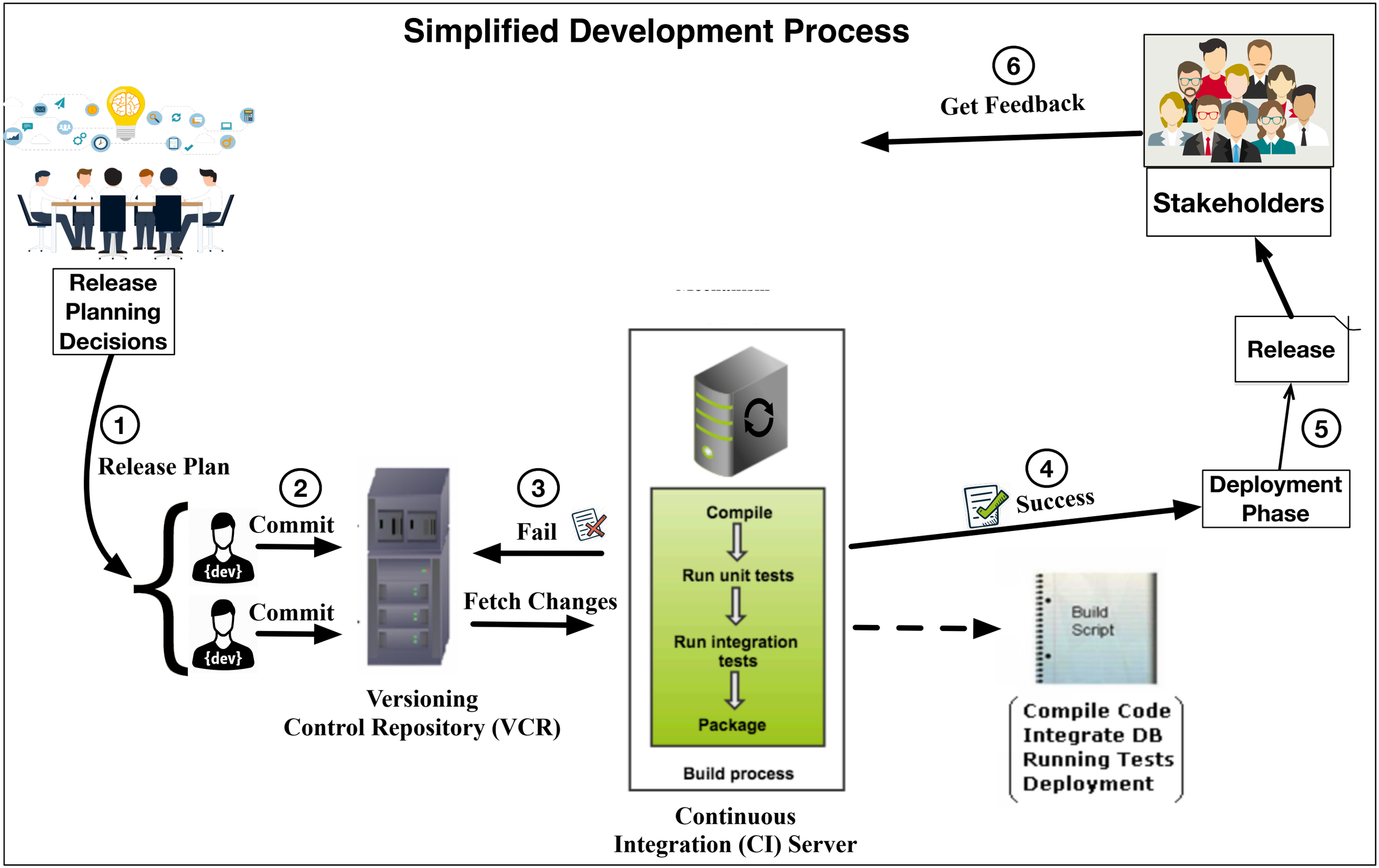
5

Release

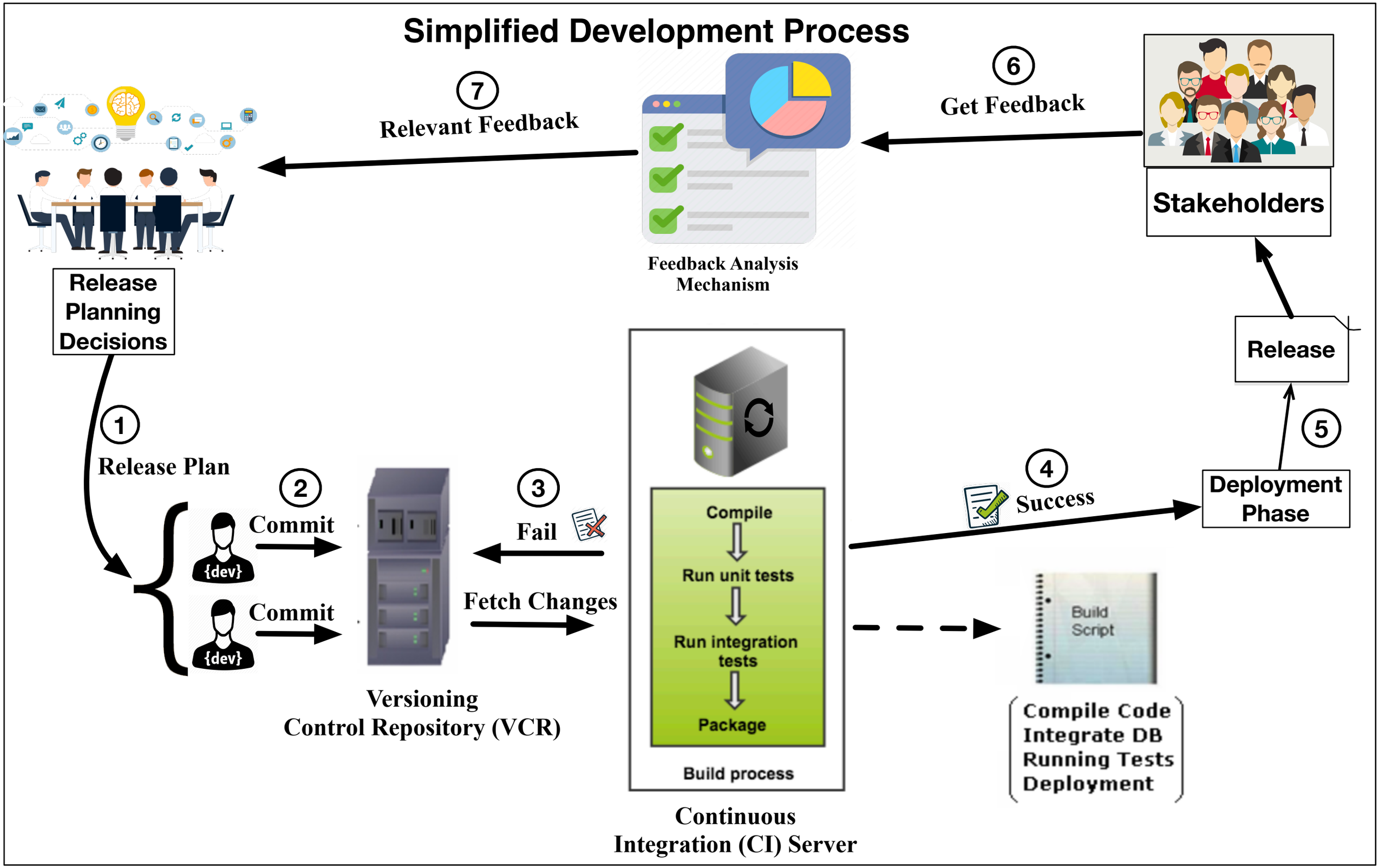
Deployment Phase

Compile Code
Integrate DB
Running Tests
Deployment

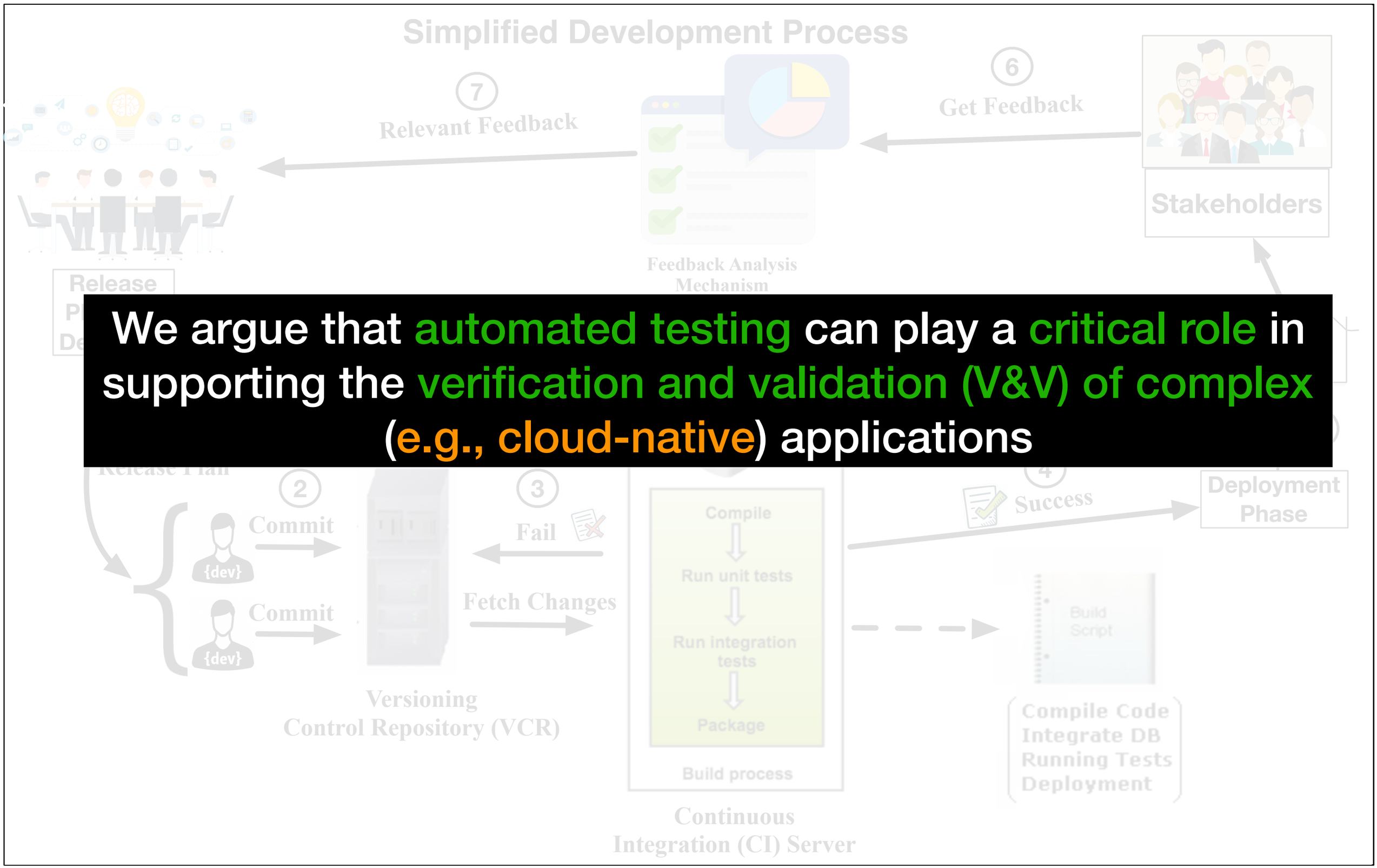
Research Context and Motivation



Research Context and Motivation



Research Context and Motivation



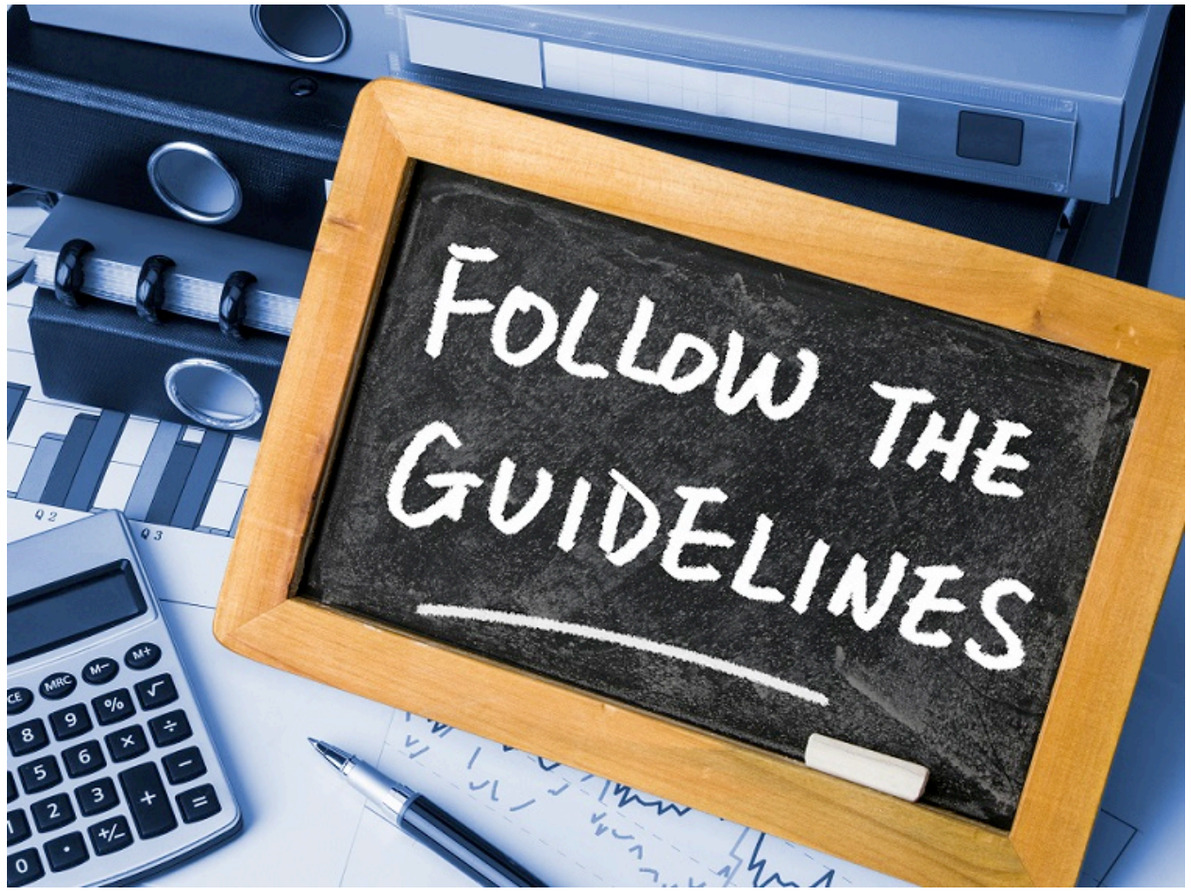
We argue that automated testing can play a critical role in supporting the verification and validation (V&V) of complex (e.g., cloud-native) applications

How to come up with a Vision?

“No Silver Bullet, but…”

How to come up with a Vision?

“..there are guidelines…”

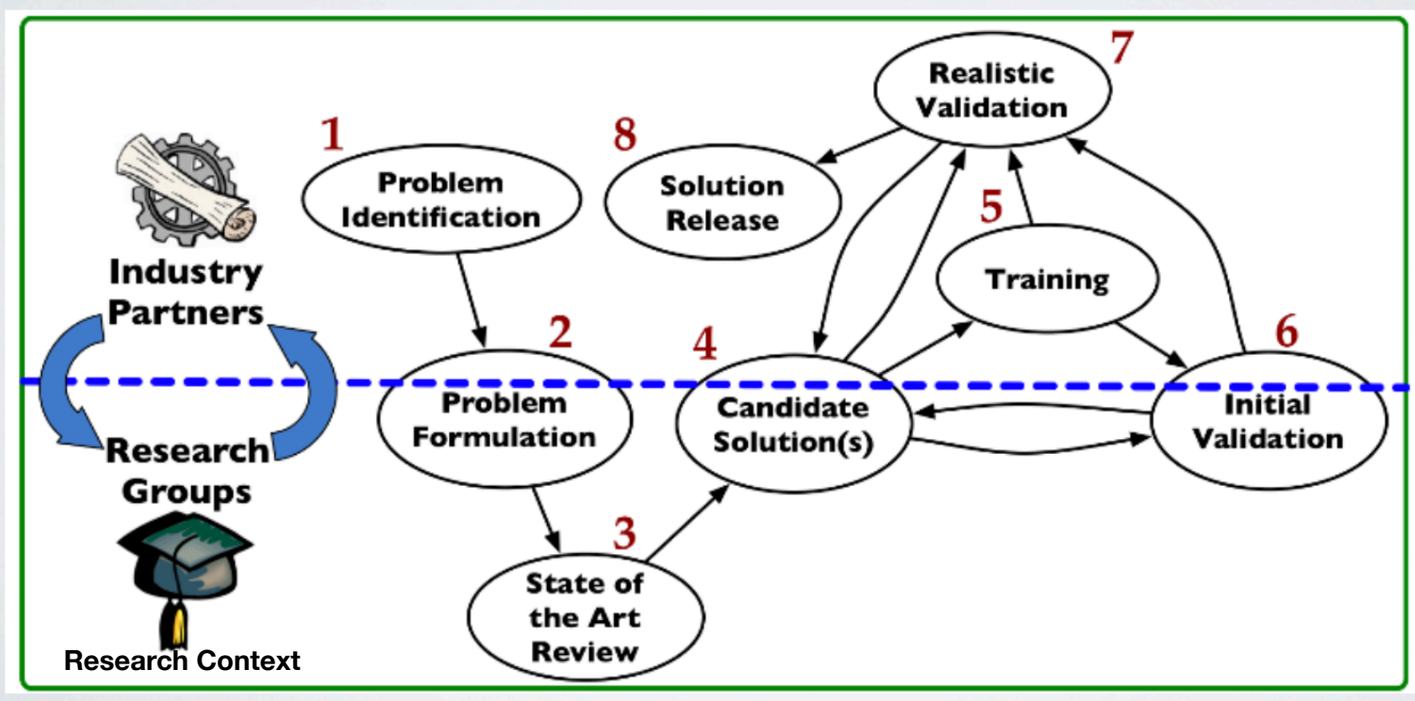


How to come up with a Vision?

"..there are guidelines..."

Mode of Collaboration

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



4

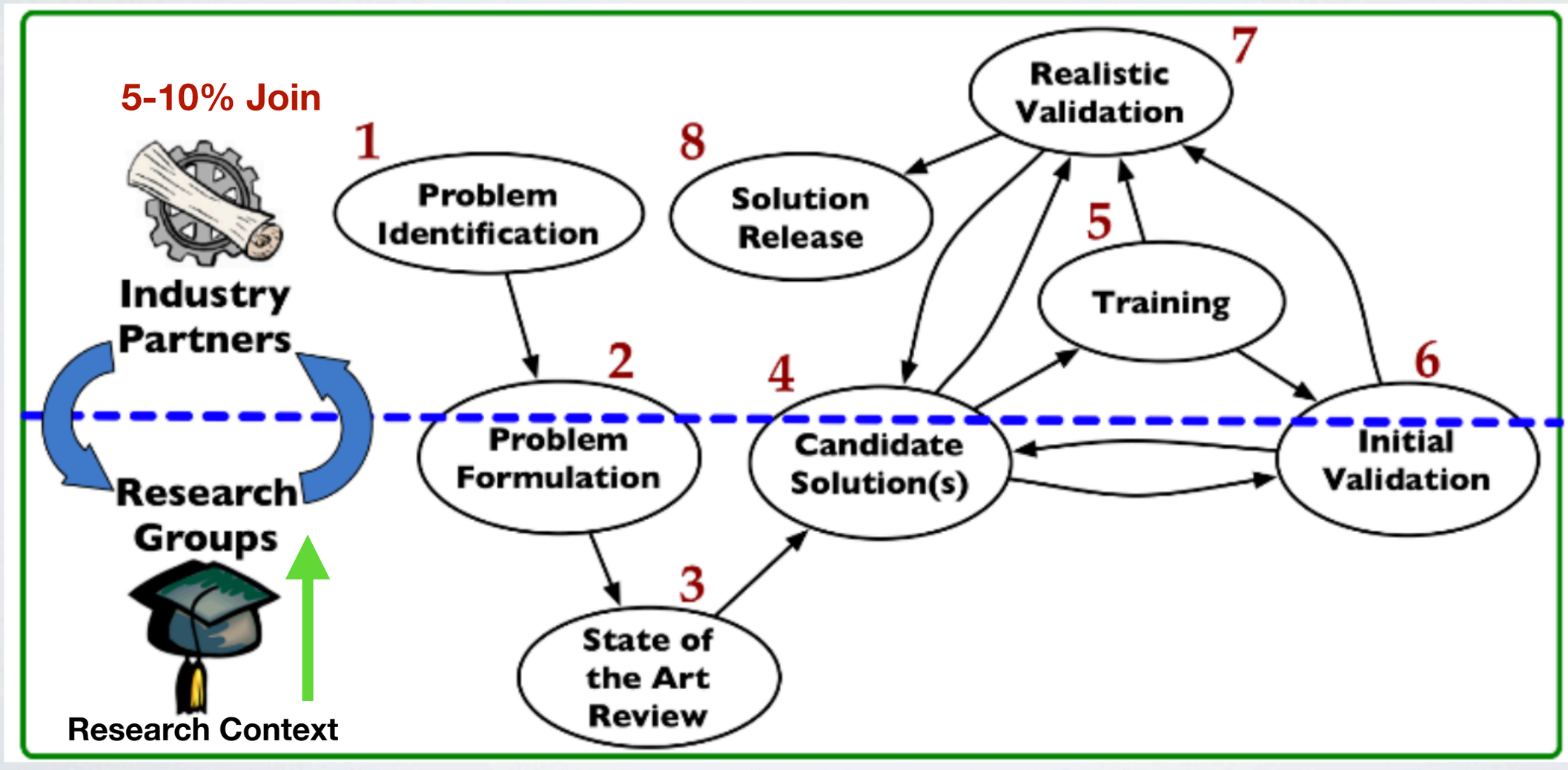
"Why and How to Get a PhD?" Prof. Briand - well know for his applied research

https://bit.ly/2LGg7nl

How to come up with a Vision?

"..there are guidelines..."

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



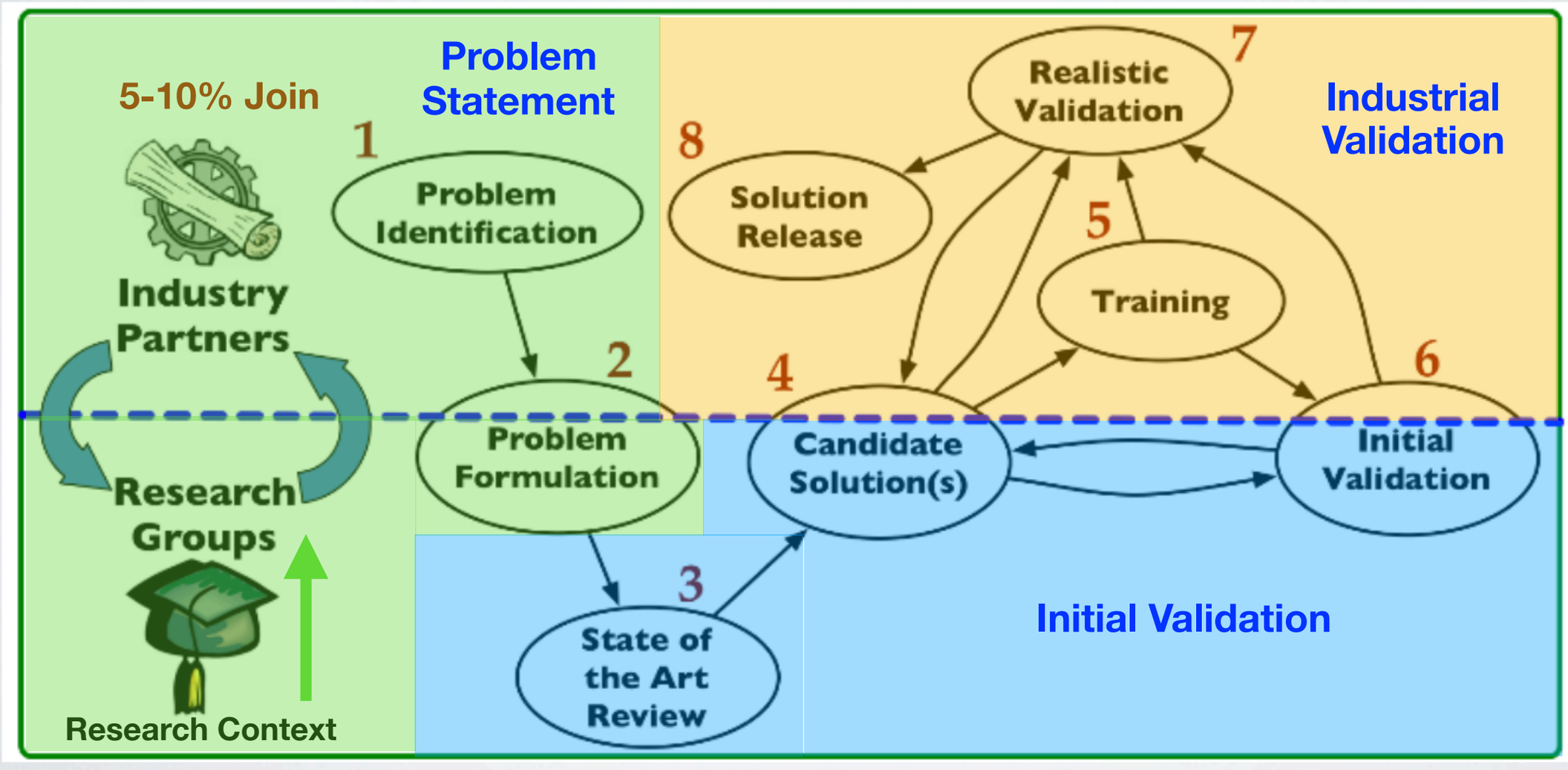
"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

How to come up with a Vision?

"..there are guidelines..."

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations

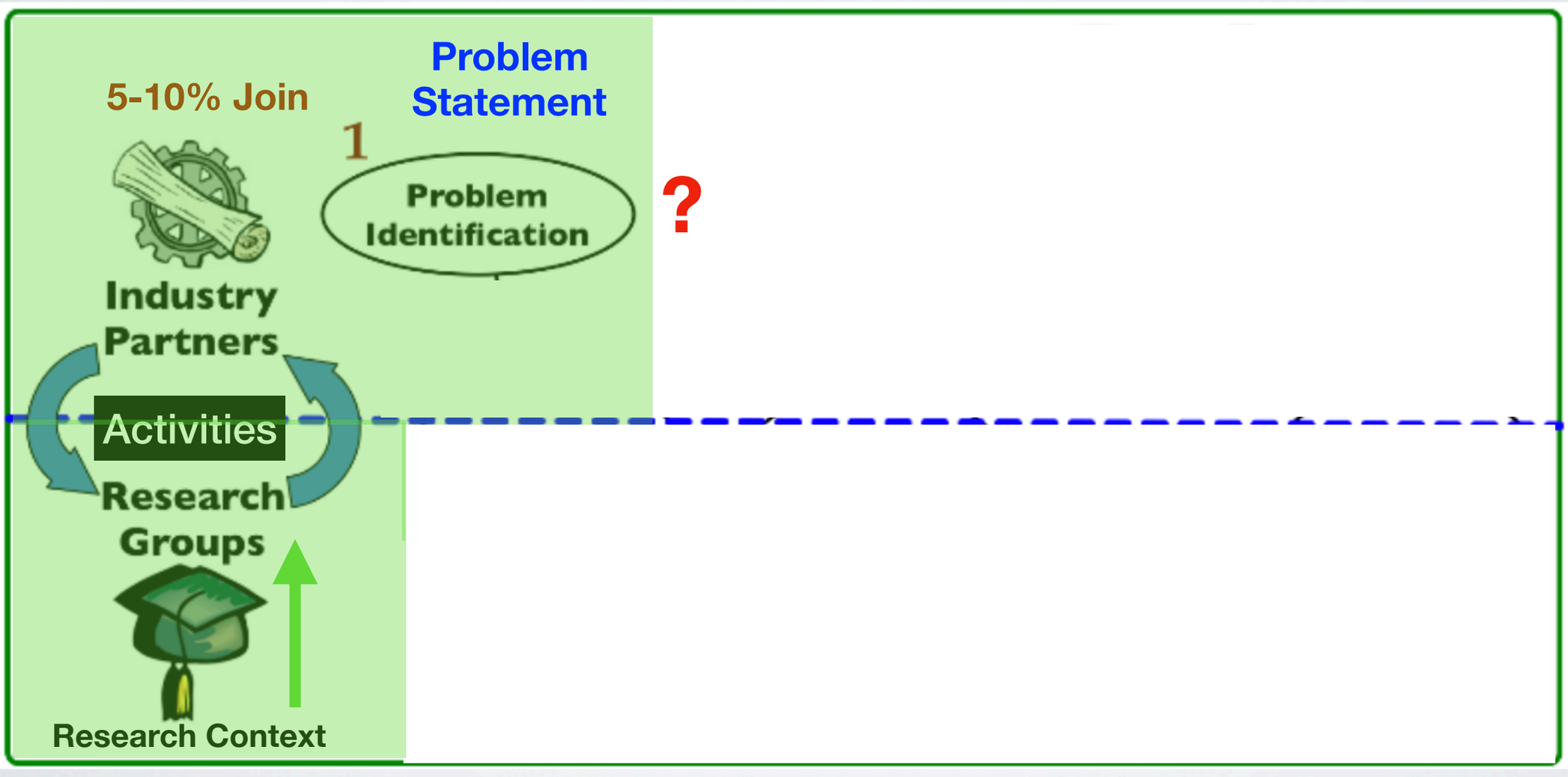


"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

Industrial Relevance of the Problem?

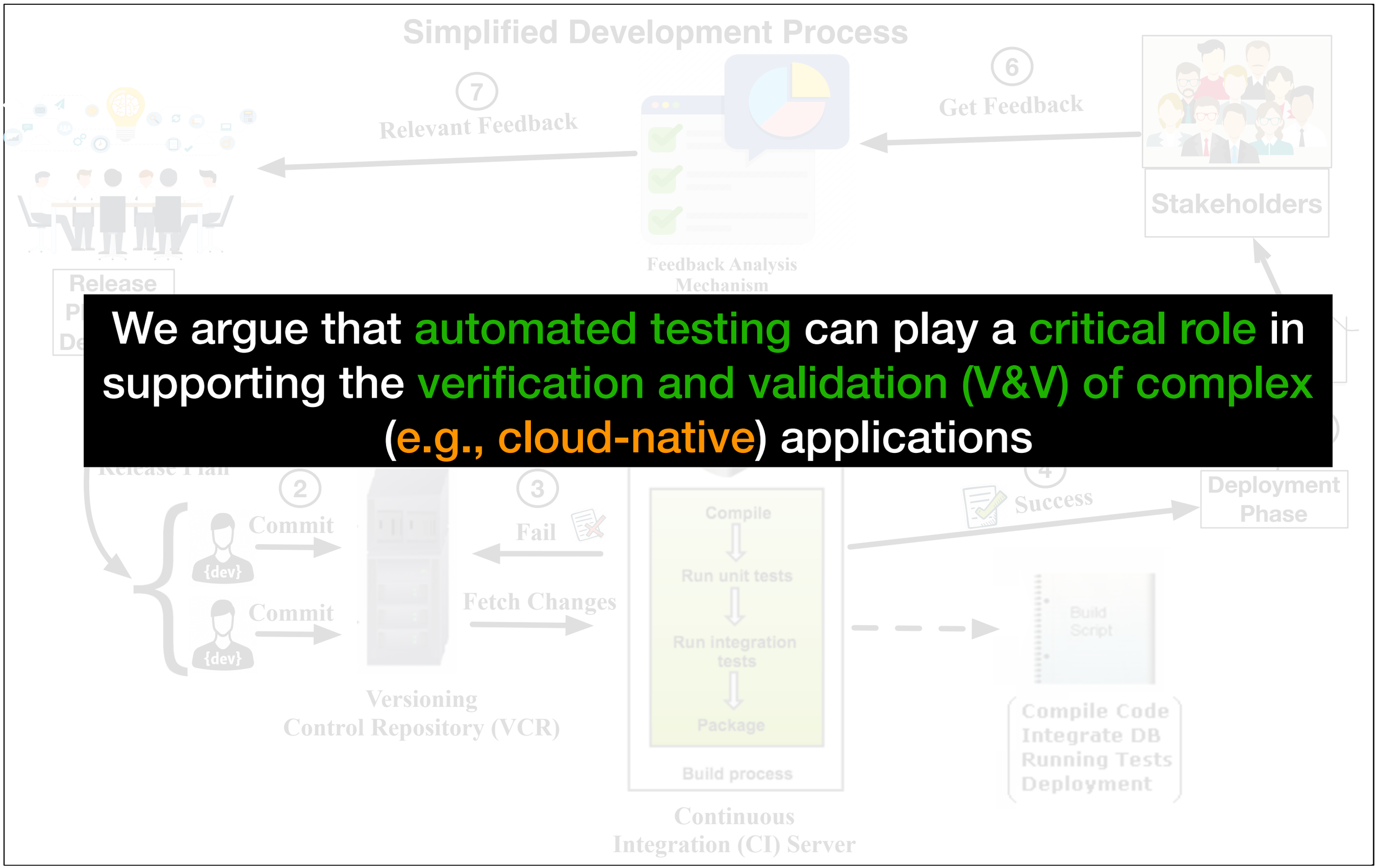
- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



"Why and How to Get a PhD?" Lionel Briand

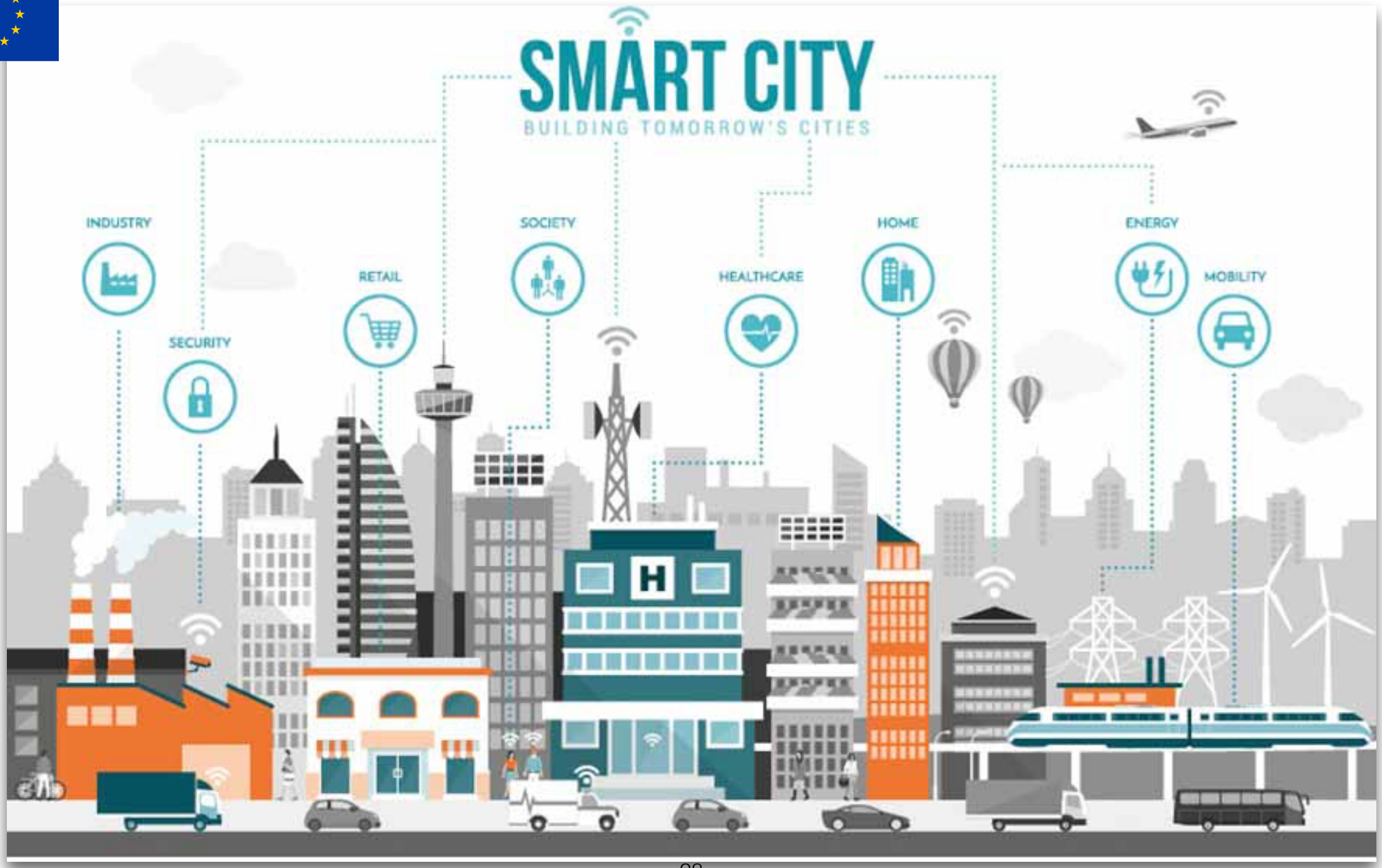
https://bit.ly/2LGg7nl

Industrial Relevance of the Problem?

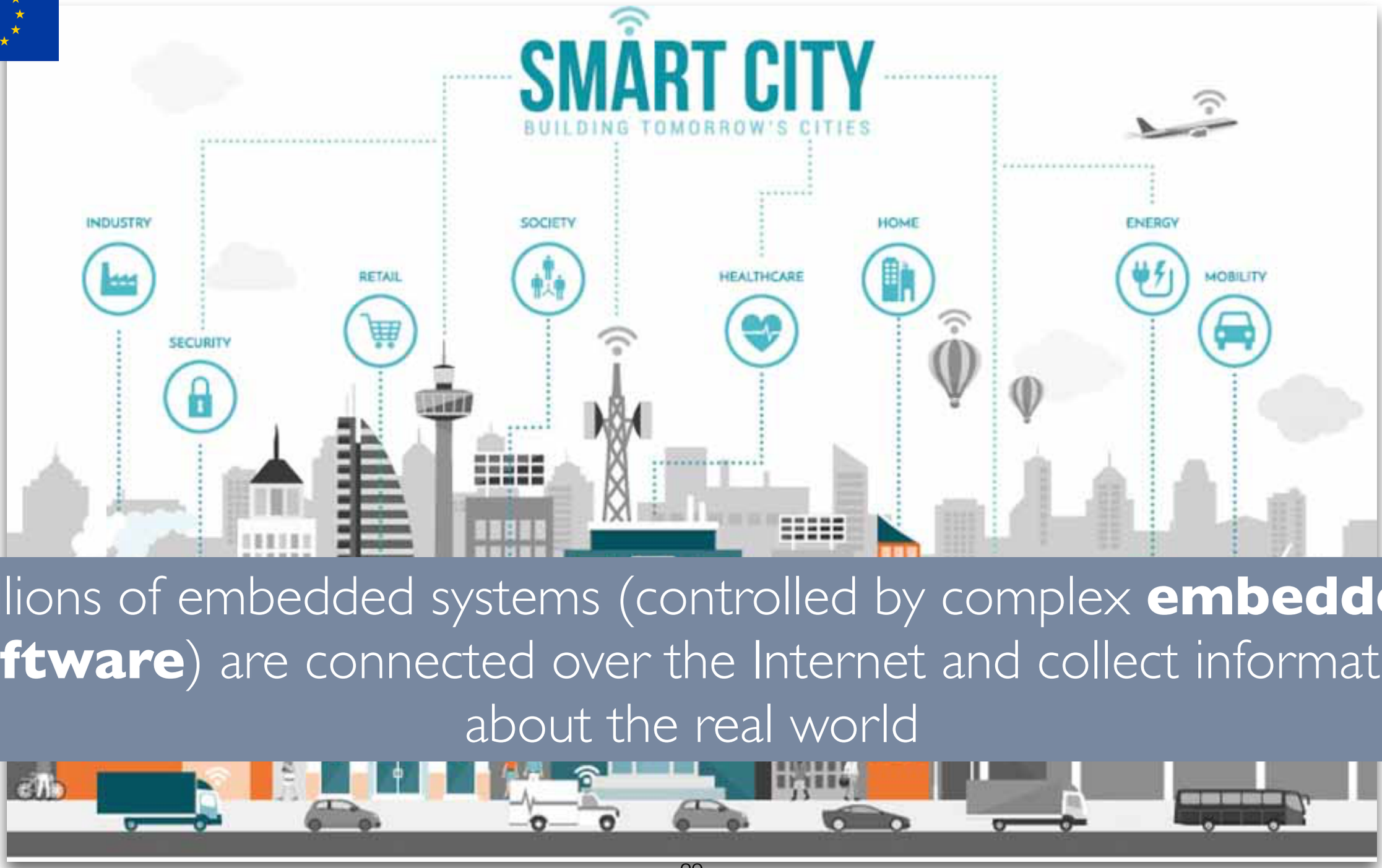


We argue that automated testing can play a critical role in supporting the verification and validation (V&V) of complex (e.g., cloud-native) applications

Research Context

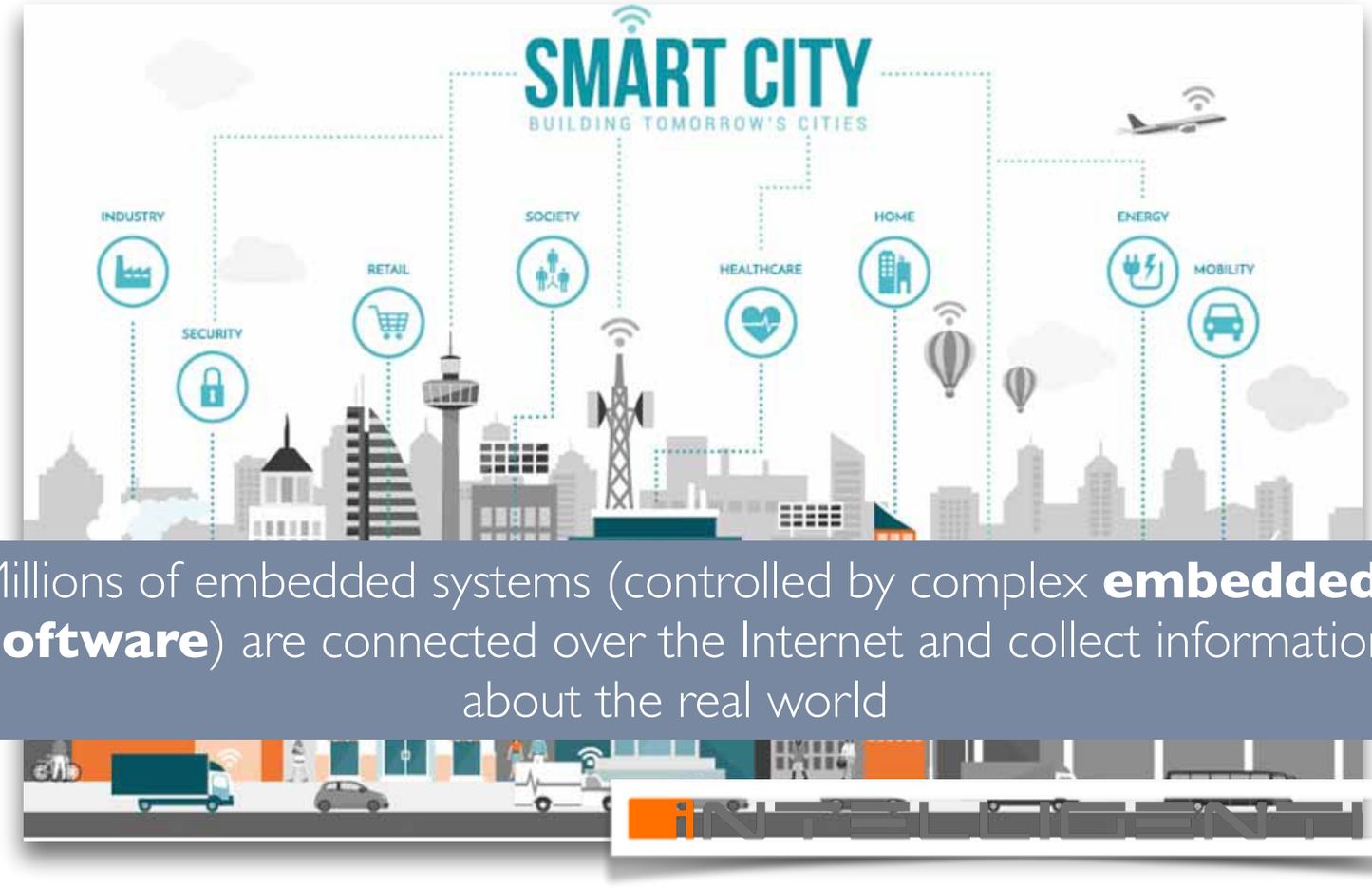


Research Context



Millions of embedded systems (controlled by complex **embedded software**) are connected over the Internet and collect information about the real world

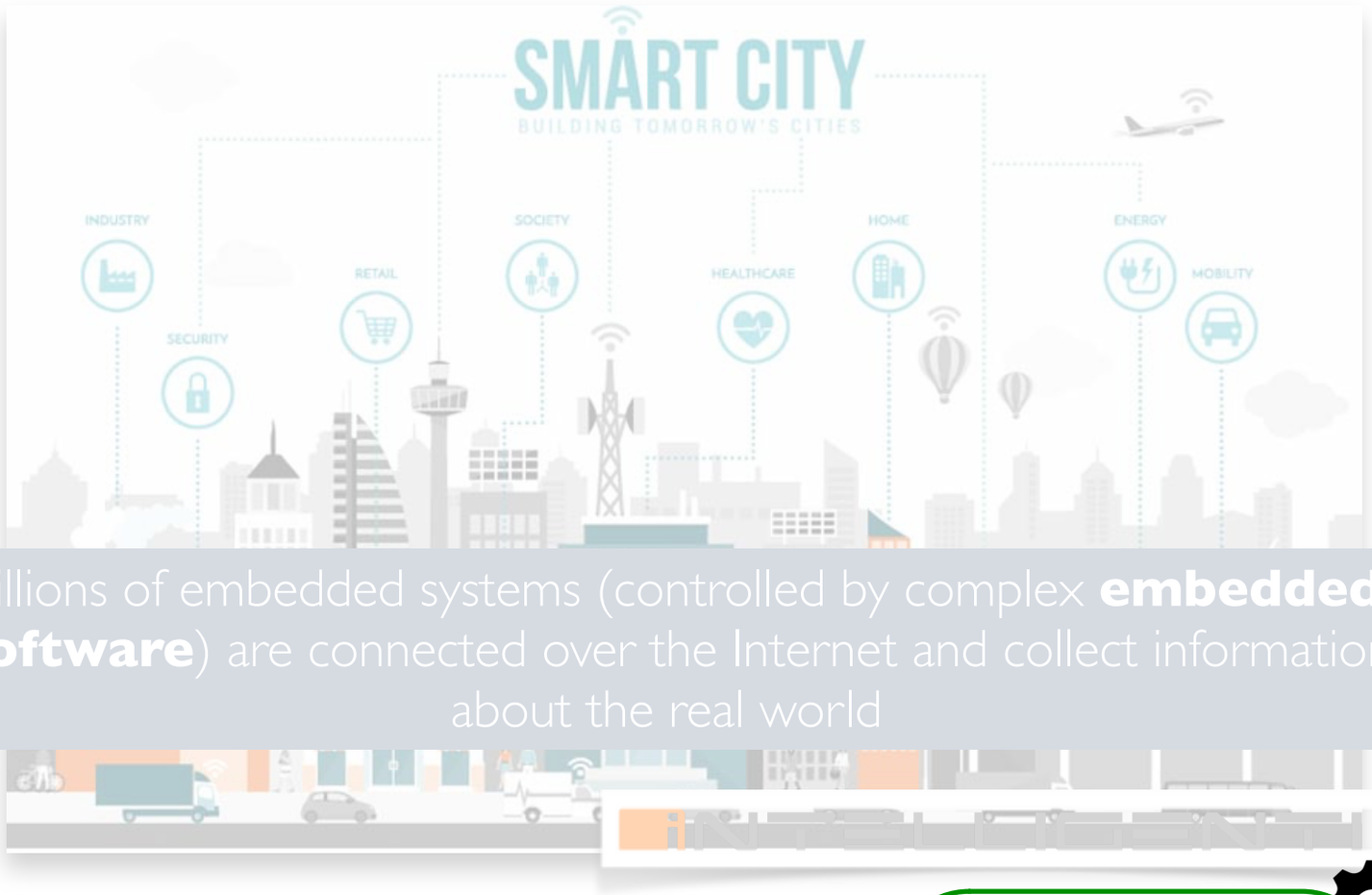
Industrial Relevance of the Problem?



Millions of embedded systems (controlled by complex **embedded software**) are connected over the Internet and collect information about the real world

Etc.

Industrial Relevance of the Problem?



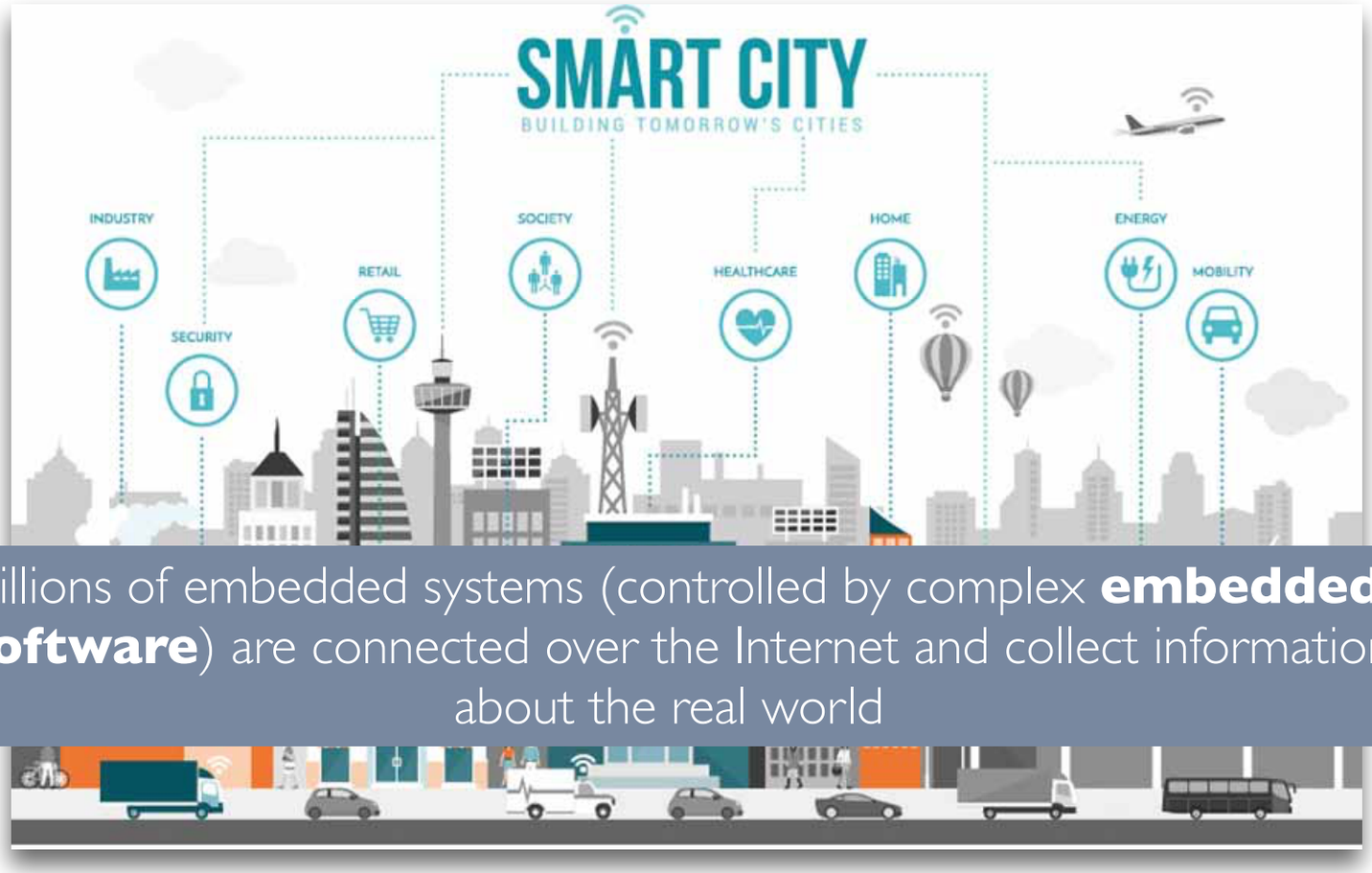
Millions of embedded systems (controlled by complex **embedded software**) are connected over the Internet and collect information about the real world

Test Suite Generation

75% of these **companies** interested to **solutions** for **debugging and testing** their **cloud-native applications**

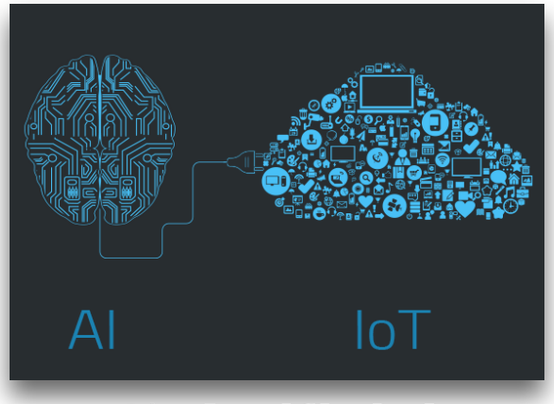
Etc.

Research Challenges and Opportunities

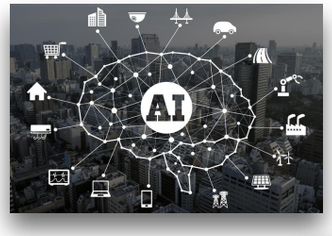


Millions of embedded systems (controlled by complex **embedded software**) are connected over the Internet and collect information about the real world

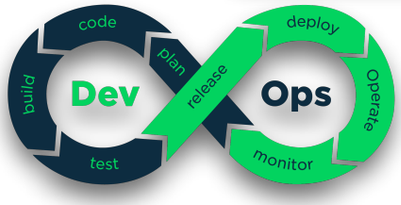
1) **Robotics**



2) **Artificial Intelligence (AI)**



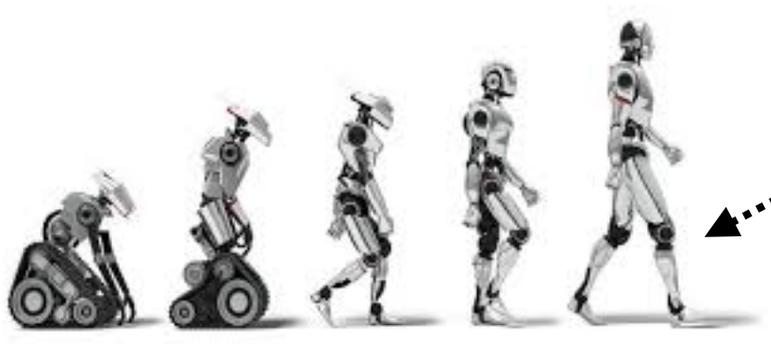
Etc.



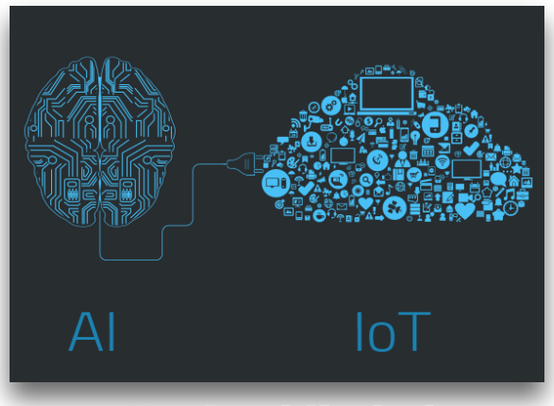
3) **DevOps, IoT, Intelligence (AI), Automated Testing (AT)**

Research Challenges and Opportunities

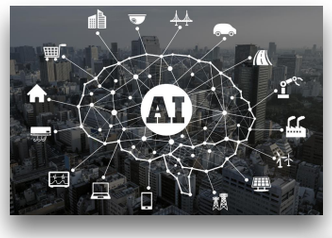
Next 10 Years

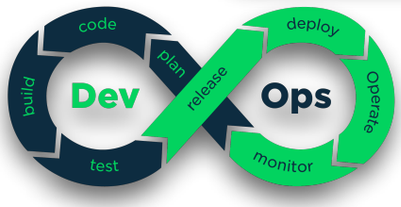


1) **Robotics**



2) **Artificial Intelligence (AI)**



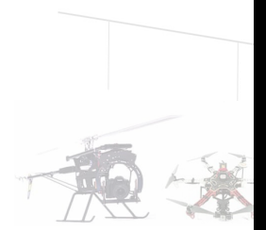


3) **DevOps, IoT, Intelligence (AI), Automated Testing (AT)**

Research Challenges and Opportunities

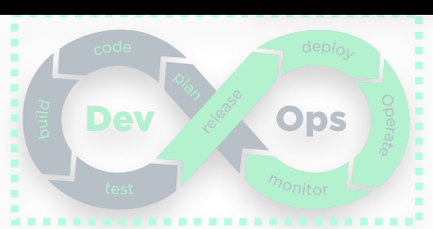
Test Suite Generation

75% of these **companies** interested to **solutions** for **debugging and testing their cloud-native applications**



1) **Robotics**

2) **Artificial Intelligence (AI)**



3) **DevOps, IoT, Intelligence (AI), Automated Testing (AT)**

Industrial Relevance of the Problem?

Cost of Developing, Maintaining and Testing IoT Systems

National Institute of Standards and Technology

Over $ 59 billion

cost to US economy in 2016

UNIVERSITY OF CAMBRIDGE

$ 312 billion

worldwide cost in 2016

50% of all time spent developing software is spent finding and fixing bugs

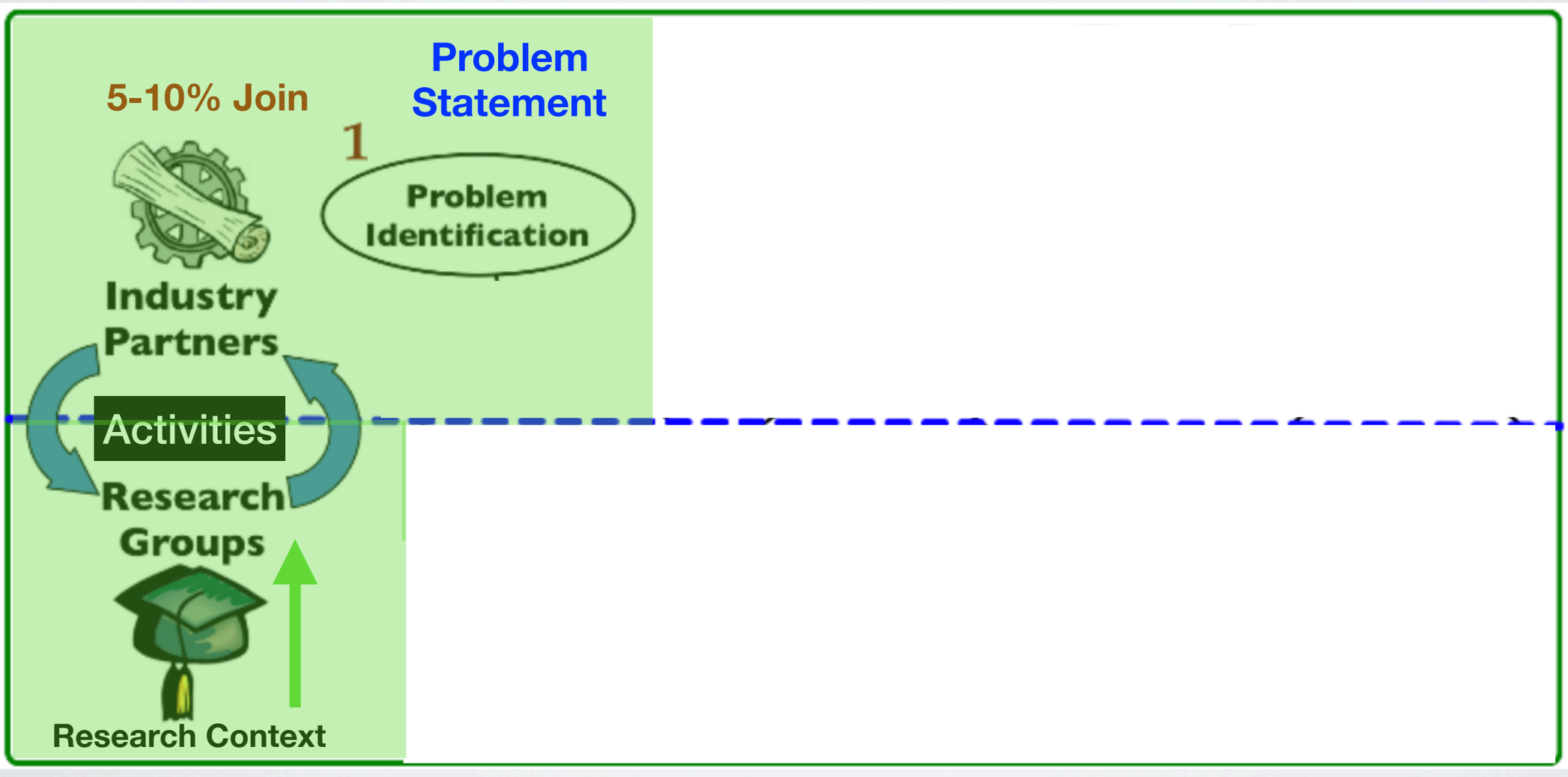
40%-50%

could be saved with proper Software Maintenance and Testing

35

Industrial Relevance of the Problem?

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

Outline

PART I

1) Research Context and Motivation

PART II

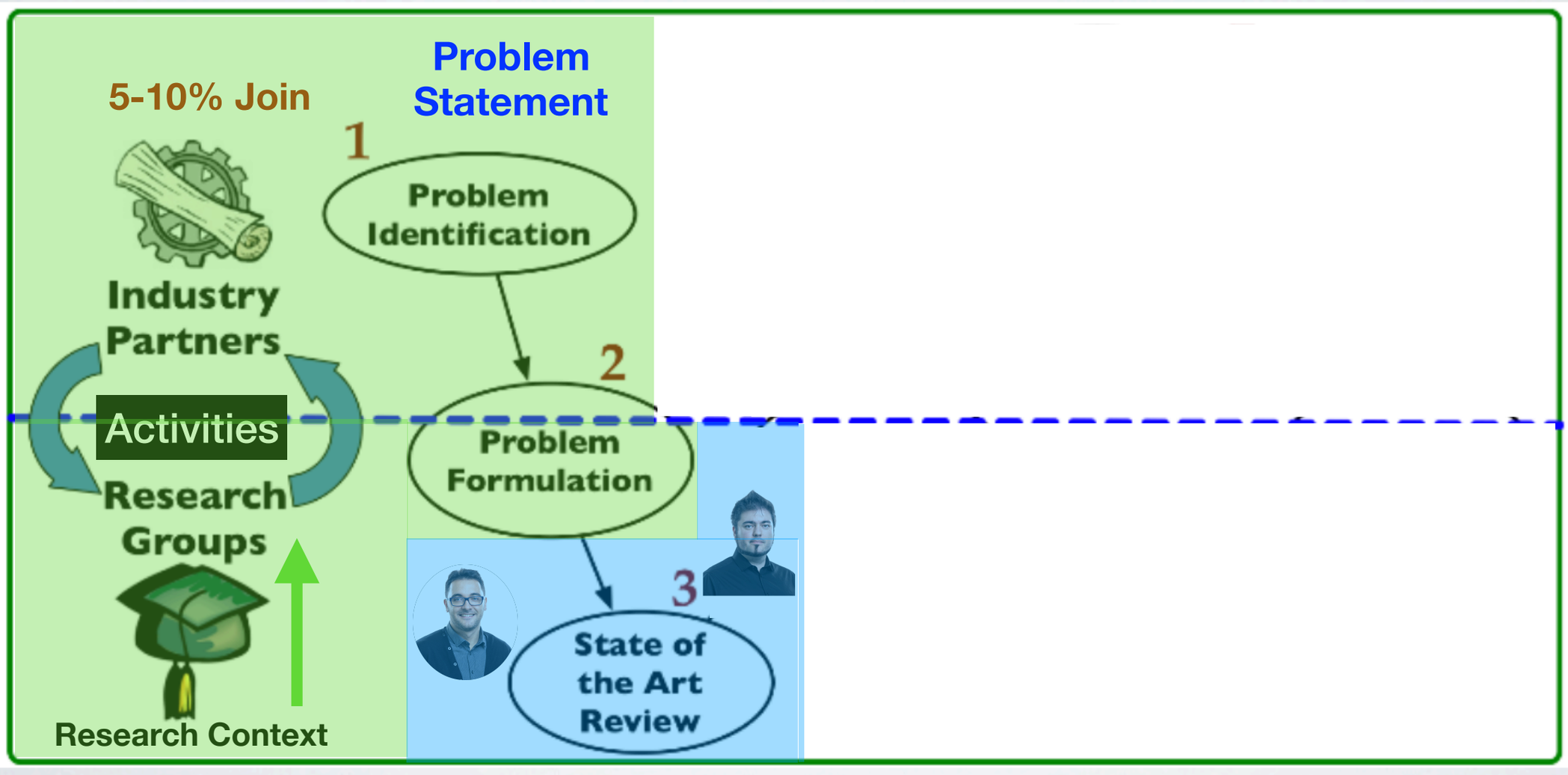
2) Literature review on:

- Testing Challenges in the Cloud
- Automated Testing
- Cloud-based testing demonstration

3) Future Work

Literature Review on Testing for the Cloud

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

Literature Review on Testing for the Cloud

The Cloudification Perspectives of Search-based Software Testing

Diego Martin
Zurich University of Applied Science (ZHAW)
Winterthur, Switzerland
marg@zhaw.ch

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)
Winterthur, Switzerland
panc@zhaw.ch

***Abstract*—To promote and sustain the future of our society, the most critical challenge of contemporary software engineering and cloud computing experts are related to the efficient integration of emerging cloudification and DevOps practices in the development and testing processes of modern systems. In this context, we argue that SBST can play a critical role in improving testing practices and automating the verification and validation (V&V) of cloudification properties of Cloud Native Applications (CNA). Hence, in this paper, we focus on the untouched side of SBST in the cloud field, by discussing (1) the testing challenges in the cloud research field and (2) summarizing the recent contributions of SBST in supporting development practices of CNA. Finally, we discuss the emerging research topics characterizing the cloudification perspectives of SBST in the cloud field.**

***Keywords*-Cloud Native Applications, Search-based Software Testing, Test Suite Generation**

I. Introduction

The first work on Search-Based Software Testing (SBST) appeared in 1976 [1] and in the last decade the SBST research field reached a high maturity, with several research work and tools [2]–[5] aimed at supporting test data generation and test suite quality assessment [6]–[9].

The most critical challenge of software engineering and cloud computing experts is related to the efficient integration of emerging cloudification [10] and DevOps practices in the development and testing processes of modern systems. This put in place the need of solutions ensuring, quantifying, and verifying the elastic scalability (i.e., adjusting their capacity by adding or removing resources) and resiliency (i.e., anticipating failures and fluctuation) [11] of the software/hardware (micro)services composing the systems.

In the cloud field, SBST strategies have been recently experimented for task scheduling [12] and service composition [13]. We argue that SBST can play a critical role in the supporting the verification and validation (V&V) of cloud native applications. Hence, in this paper, we summarize the main challenges and opportunities on the untouched side of the SBST in the cloud field.

II. Analysis and Discussion

A. Cloud Testing Literature Insights

Cloud-native applications (CNAs) are *"distributed, elastic and horizontal-scalable systems composed of (micro)services which isolates states in a minimum of stateful components"* [14]. Each self-contained deployment unit of CNAs is designed according to cloud-focused design patterns and operated on a self-service elastic platform. This means that CNAs are supposed to be ***composable***, ***decoupled***, ***elastic*** and ***resilient*** [14].

Even thought most of these *properties* have been investigated/analyzed in the literature [15], [16], there are still open problems/challenges where SBST could contribute:

1) V&V of CNA properties: Provide automated solutions for V&V of CNA properties like *composition, decoupling, elasticity and resilience.*

2) CNAs Adaptations: Provide tools for V&V of cloud migrations or migration between cloud providers.

3) Microservices Evolution: Support developers with tools providing an architectural view of microservices evolution, with V&V on the side-effects of specific changes in microservices composition and orchestration.

4) Local v.s. Global V&V: Provide automated V&V of microservices evolutions that verify the behaviour of each microservice in isolation (local level) as well as the global microservices *behaviour* and *interactions* in the system.

5) Execution Time: Execution of test suites should be efficient and effective, and should encapsulate advanced coverage criteria for microservices based architecture.

B. Literature review of SBST for the cloud

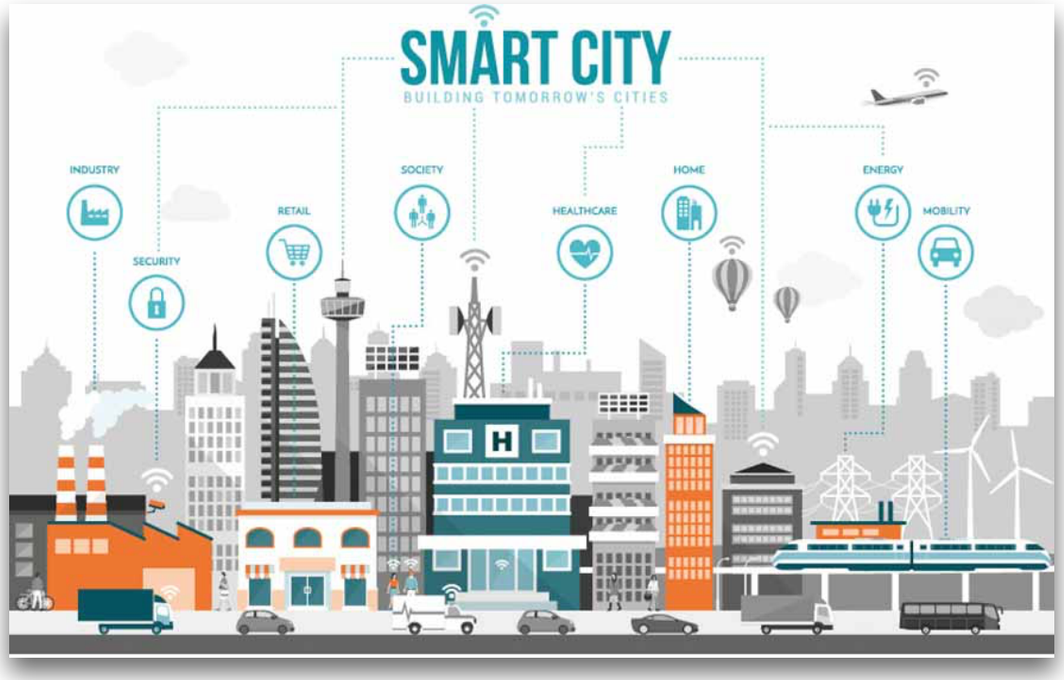
SBST literature in the cloud field focused on the optimization of testing frameworks [17], [18] and the stress of basic CNA properties (elasticity [19] and resilience [20]), with some work analyzing the challenges of cloud migrations [21], and the possibilities of using combinatorial testing [22]. However, some of the the challenges previously identified are completely untouched.

We conducted a literature review on SBST papers published in the last 5 years (period 2014-2019), focusing on the works that contributed to the cloud research field.

The literature review has been performed by using DBLP as main source of information. Specifically, in the period analyzed, the selection of papers in DBLP was performed by using specific sets of search keywords, according to the following logic formula: CLOUD ∧ (GENETIC ∨ COMBINATORIAL ∨ SIMULATED ANNEALING ∨ TABU SEARCH ∨ PROFILING ∨ SLICING ∨ COVERAGE ∨ COEVOLUTION ∨ MUTATION ∨ MORPH ∨ HEALING ∨ SELF REPAIR ∨ HILL CLIMBING ∨ SEARCH BASED).

The aforementioned filter resulted in only **6** papers relevant to SBST applications in the cloud field from the **396**

SBST 2019



"...we discuss how automated testing is a potential solution to the challenges identified in the cloud testing field"

Test Suite Generation

Literature Review on Testing for the Cloud

The Cloudification Perspectives of Search-based Software Testing

Diego Martin
Zurich University of Applied Science (ZHAW)
Winterthur, Switzerland
marg@zhaw.ch

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)
Winterthur, Switzerland
panc@zhaw.ch

***Abstract*—To promote and sustain the future of our society, the most critical challenge of contemporary software engineering and cloud computing experts are related to the efficient integration of emerging cloudification and DevOps practices in the development and testing processes of modern systems. In this context, we argue that SBST can play a critical role in improving testing practices and automating the verification and validation (V&V) of cloudification properties of Cloud Native Applications (CNA). Hence, in this paper, we focus on the untouched side of SBST in the cloud field, by discussing (1) the testing challenges in the cloud research field and (2) summarizing the recent contributions of SBST in supporting development practices of CNA. Finally, we discuss the emerging research topics characterizing the cloudification perspectives of SBST in the cloud field.**

***Keywords*-Cloud Native Applications, Search-based Software Testing, Test Suite Generation**

I. Introduction

The first work on Search-Based Software Testing (SBST) appeared in 1976 [1] and in the last decade the SBST research field reached a high maturity, with several research work and tools [2]–[5] aimed at supporting test data generation and test suite quality assessment [6]–[9].

The most critical challenge of software engineering and cloud computing experts is related to the efficient integration of emerging cloudification [10] and DevOps practices in the development and testing processes of modern systems. This put in place the need of solutions ensuring, quantifying, and verifying the elastic scalability (i.e., adjusting their capacity by adding or removing resources) and resiliency (i.e., anticipating failures and fluctuation) [11] of the software/hardware (micro)services composing the systems.

In the cloud field, SBST strategies have been recently experimented for task scheduling [12] and service composition [13]. We argue that SBST can play a critical role in the supporting the verification and validation (V&V) of cloud native applications. Hence, in this paper, we summarize the main challenges and opportunities on the untouched side of the SBST in the cloud field.

II. Analysis and Discussion

A. Cloud Testing Literature Insights

Cloud-native applications (CNAs) are *"distributed, elastic and horizontal-scalable systems composed of (micro)services which isolates states in a minimum of stateful components"* [14]. Each self-contained deployment unit of CNAs is designed according to cloud-focused design patterns and operated on a self-service elastic platform. This means that CNAs are supposed to be ***composable***, ***decoupled***, ***elastic*** and ***resilient*** [14].

Even thought most of these *properties* have been investigated/analyzed in the literature [15], [16], there are still open problems/challenges where SBST could contribute:

1) V&V of CNA properties: Provide automated solutions for V&V of CNA properties like *composition, decoupling, elasticity and resilience.*

2) CNAs Adaptations: Provide tools for V&V of cloud migrations or migration between cloud providers.

3) Microservices Evolution: Support developers with tools providing an architectural view of microservices evolution, with V&V on the side-effects of specific changes in microservices composition and orchestration.

4) Local v.s. Global V&V: Provide automated V&V of microservices evolutions that verify the behaviour of each microservice in isolation (local level) as well as the global microservices *behaviour* and *interactions* in the system.

5) Execution Time: Execution of test suites should be efficient and effective, and should encapsulate advanced coverage criteria for microservices based architecture.

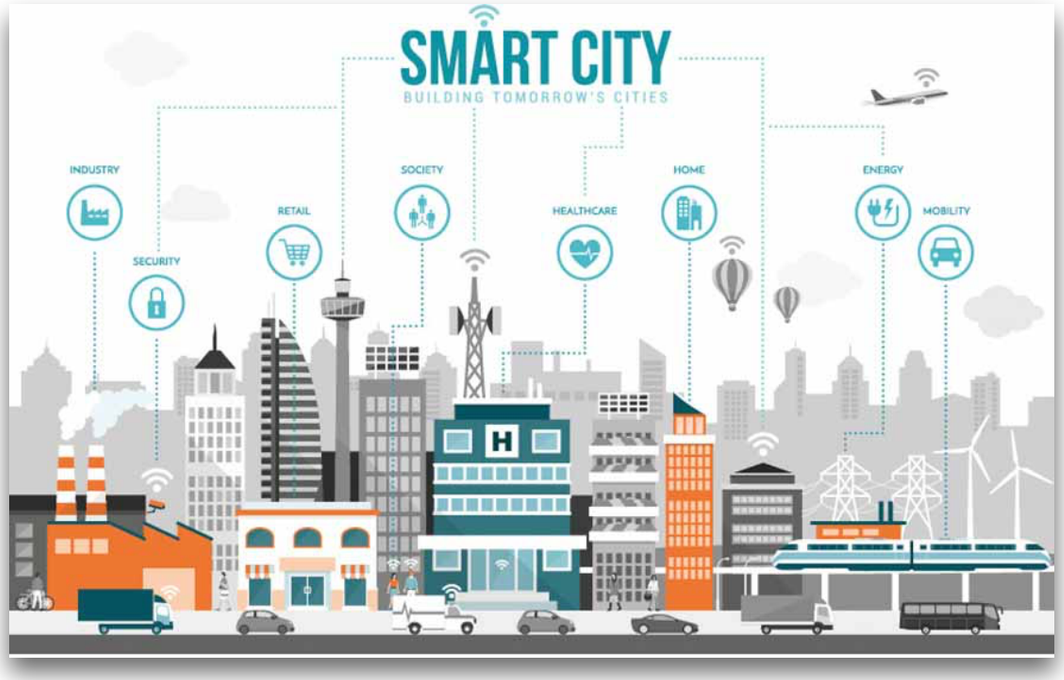
B. Literature review of SBST for the cloud

SBST literature in the cloud field focused on the optimization of testing frameworks [17], [18] and the stress of basic CNA properties (elasticity [19] and resilience [20]), with some work analyzing the challenges of cloud migrations [21], and the possibilities of using combinatorial testing [22]. However, some of the the challenges previously identified are completely untouched.

We conducted a literature review on SBST papers published in the last 5 years (period 2014-2019), focusing on the works that contributed to the cloud research field.

The literature review has been performed by using DBLP as main source of information. Specifically, in the period analyzed, the selection of papers in DBLP was performed by using specific sets of search keywords, according to the following logic formula: CLOUD ∧ (GENETIC ∨ COMBINATORIAL ∨ SIMULATED ANNEALING ∨ TABU SEARCH ∨ PROFILING ∨ SLICING ∨ COVERAGE ∨ COEVOLUTION ∨ MUTATION ∨ MORPH ∨ HEALING ∨ SELF REPAIR ∨ HILL CLIMBING ∨ SEARCH BASED).

The aforementioned filter resulted in only **6** papers relevant to SBST applications in the cloud field from the **396**



1) Cloud Testing Challenges

2) Literature review Automated Testing

3) Cloud future perspectives

Cloud Testing Challenges

ACM SIGSOFT Software Engineering Notes Page 1 May 2012 Volume 37 Number 3

Empirical Evaluation of Cloud-based Testing Techniques: A Systematic Review

Priyanka
Thapar University,India
priyankamatrix@gmail.com

Inderveer Chana
Thapar University,India
inderveer@thapar.edu

Ajay Rana
Amity University,Noida,India
ajay_rana@amity.edu

Annals of Telecommunications manuscript No.
(will be inserted by the editor)

A survey on formal active and passive testing with applications to the cloud

Ana R. Cavalli · Teruo Higashino · Manuel Núñez

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study

Nane Kratzke*, Peter-Christian Quint

Lübeck University of Applied Sciences, Center of Excellence for Communication, Systems and Applications, Mönkhofer Weg 239, 23562 Lübeck, Germany

ARTICLE INFO

Article history:
Received 23 September 2016
Revised 30 November 2016
Accepted 4 January 2017
Available online 5 January 2017

MSC:
00-01
99-00

Keywords:
Cloud-native application
CNA
Systematic mapping study
Elastic platform
Microservice
Self service
Pattern
Softwareization

ABSTRACT

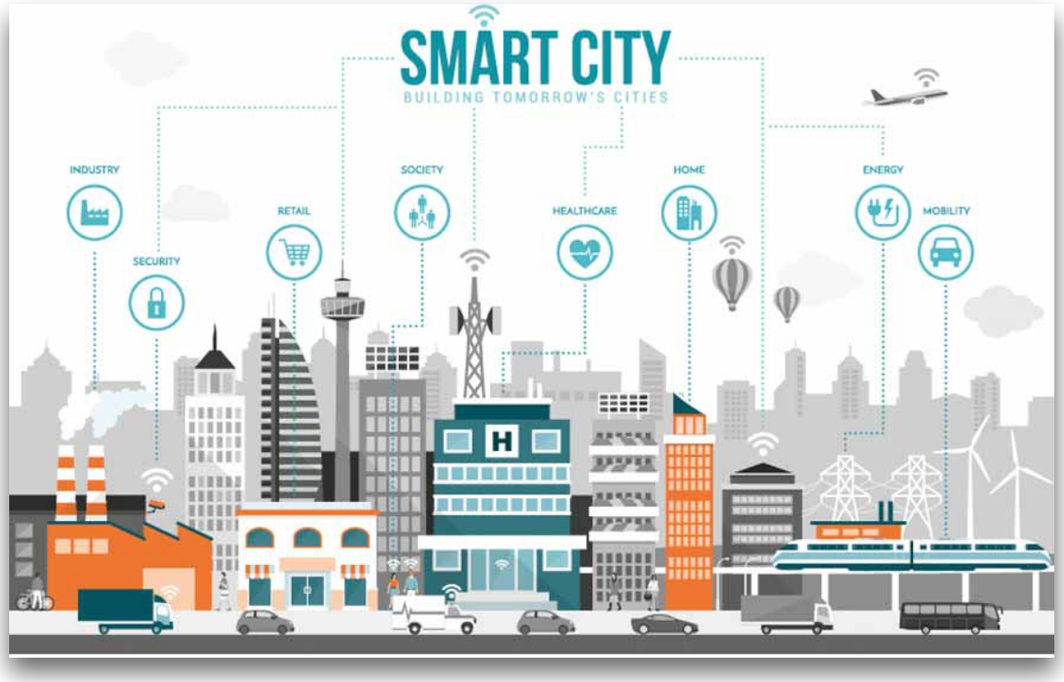
It is common sense that cloud-native applications (CNA) are intentionally designed for the cloud. Although this understanding can be broadly used it does not guide and explain what a cloud-native application exactly is. The term "cloud-native" was used quite frequently in birthday times of cloud computing (2006) which seems somehow obvious nowadays. But the term disappeared almost completely. Suddenly and in the last years the term is used again more and more frequently and shows increasing momentum. This paper summarizes the outcomes of a systematic mapping study analyzing research papers covering "cloud-native" topics, research questions and engineering methodologies. We summarize research focuses and trends dealing with cloud-native application engineering approaches. Furthermore, we provide a definition for the term "cloud-native application" which takes all findings, insights of analyzed publications and already existing and well-defined terminology into account.

1. Introduction

The birthday of the cloud can be dated into the year 2006 – the first launch of a general purpose public cloud service (Simple Storage Service, S3 at 13th March 2006[1]) by the currently most prominent public cloud service provider Amazon Web Services (AWS). Therefore, this study has not considered any papers dated before 2006. Meanwhile other providers and further services followed. All these cloud providers and their services forming nowadays our understanding of the term cloud computing. Although terms like public/private/hybrid cloud computing and acronyms like IaaS (Infrastructure as a Service), PaaS (Platform as a Service) or SaaS (Software as a Service) are frequently used, these terms are often understood in differing ways. Gladfully, the mentioned terms are defined precisely by the NIST definition of cloud computing (Mell and Grance, 2011). But there remains confusion with more specific topics in cloud computing like cloud-native applications (CNA). This contribution investigates a more precise understanding of the term "cloud-native".

According to Fig. 1 and Google trends the term "cloud-native" has an unusual diffusion rate over time. In birthday times of cloud computing – so about 10 years ago – it was used quite frequently. To label something as cloud-native at the starting days of cloud computing seems somehow obvious today. However, the usage of the term "cloud-native" decreased over following years according to Google. But since 2015, the term is more and more frequently used again and gained momentum (see Fig. 1). We have to admit that Google trends is not a very reliable source for research – it is known to be prone for "industrial buzzwords". But, our literature review will show a very similar usage of the term "cloud-native" in research studies (the reader might want to compare Figs. 1 and 5a). Furthermore, our data indicates that the current understanding of the term "cloud-native" seems to have its origin in research and not in industry. It was used and characterized since 2012 in research. And yes, it gets suddenly popular in industry, but not before 2015 (according to Google trends, see Fig. 1). We assume this sudden rise in industry has to do with the current popularity of microservice and container-based approaches. Cloud-

* Corresponding author.
E-mail addresses: nane.kratzke@fh-luebeck.de (N. Kratzke), peter-christian.

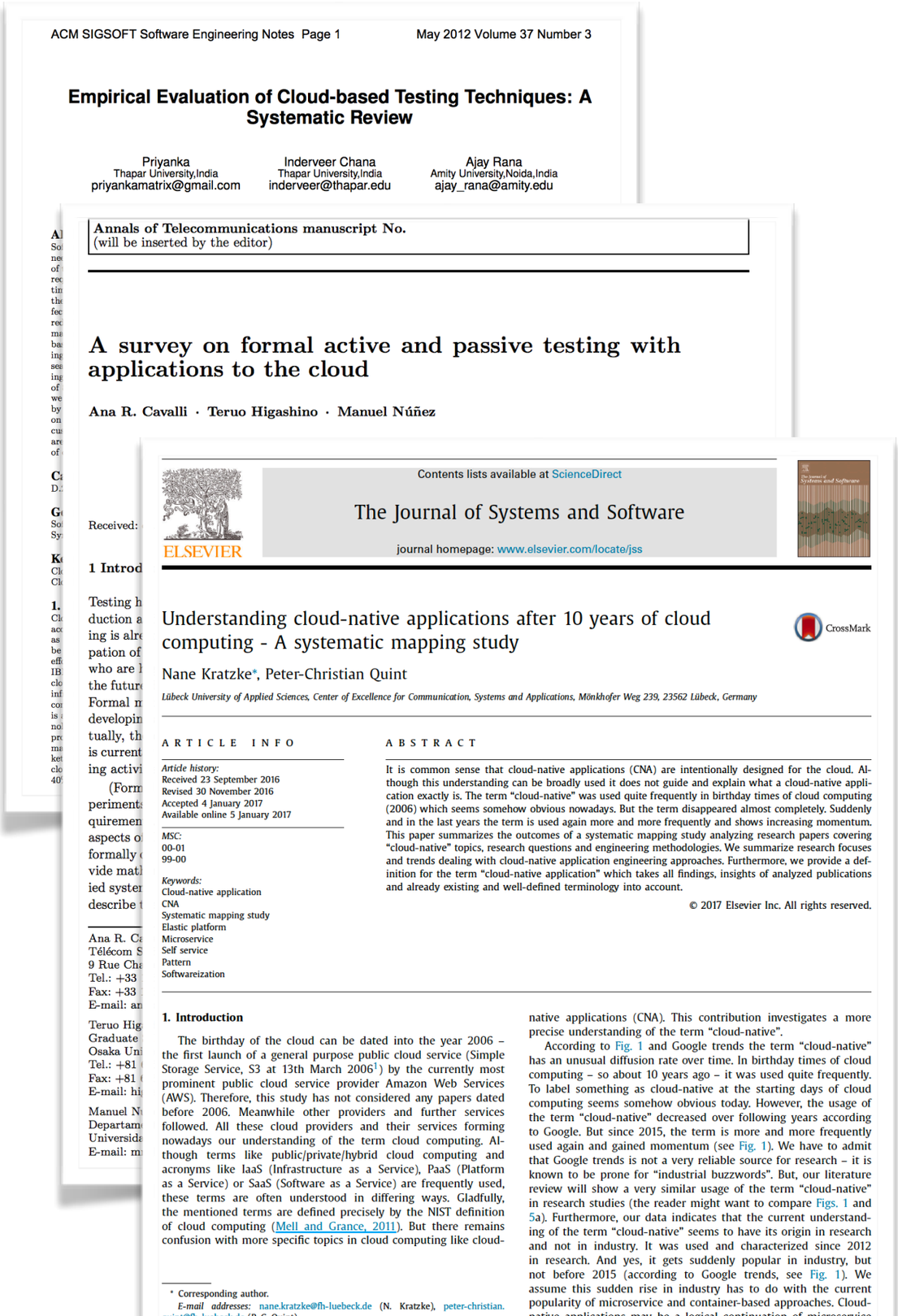


1) Cloud Testing Challenges

2) Literature review Automated Testing

3) Cloud future perspectives

Cloud Testing Challenges



ACM SIGSOFT Software Engineering Notes Page 1 May 2012 Volume 37 Number 3

Empirical Evaluation of Cloud-based Testing Techniques: A Systematic Review

Priyanka, Thapar University,India, priyankamatrix@gmail.com
Inderveer Chana, Thapar University,India, inderveer@thapar.edu
Ajay Rana, Amity University,Noida,India, ajay_rana@amity.edu

Annals of Telecommunications manuscript No.
(will be inserted by the editor)

A survey on formal active and passive testing with applications to the cloud

Ana R. Cavalli · Teruo Higashino · Manuel Núñez

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study

Nane Kratzke*, Peter-Christian Quint

Lübeck University of Applied Sciences, Center of Excellence for Communication, Systems and Applications, Mönkhofer Weg 239, 23562 Lübeck, Germany

ARTICLE INFO

Article history:
Received 23 September 2016
Revised 30 November 2016
Accepted 4 January 2017
Available online 5 January 2017

MSC:
00-01
99-00

Keywords:
Cloud-native application
CNA
Systematic mapping study
Elastic platform
Microservice
Self service
Pattern
Softwareization

ABSTRACT

It is common sense that cloud-native applications (CNA) are intentionally designed for the cloud. Although this understanding can be broadly used it does not guide and explain what a cloud-native application exactly is. The term "cloud-native" was used quite frequently in birthday times of cloud computing (2006) which seems somehow obvious nowadays. But the term disappeared almost completely. Suddenly and in the last years the term is used again more and more frequently and shows increasing momentum. This paper summarizes the outcomes of a systematic mapping study analyzing research papers covering "cloud-native" topics, research questions and engineering methodologies. We summarize research focuses and trends dealing with cloud-native application engineering approaches. Furthermore, we provide a definition for the term "cloud-native application" which takes all findings, insights of analyzed publications and already existing and well-defined terminology into account.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

The birthday of the cloud can be dated into the year 2006 – the first launch of a general purpose public cloud service (Simple Storage Service, S3 at 13th March 2006[1]) by the currently most prominent public cloud service provider Amazon Web Services (AWS). Therefore, this study has not considered any papers dated before 2006. Meanwhile other providers and further services followed. All these cloud providers and their services forming nowadays our understanding of the term cloud computing. Although terms like public/private/hybrid cloud computing and acronyms like IaaS (Infrastructure as a Service), PaaS (Platform as a Service) or SaaS (Software as a Service) are frequently used, these terms are often understood in differing ways. Gladfully, the mentioned terms are defined precisely by the NIST definition of cloud computing (Mell and Grance, 2011). But there remains confusion with more specific topics in cloud computing like cloud-native applications (CNA). This contribution investigates a more precise understanding of the term "cloud-native".

According to Fig. 1 and Google trends the term "cloud-native" has an unusual diffusion rate over time. In birthday times of cloud computing – so about 10 years ago – it was used quite frequently. To label something as cloud-native at the starting days of cloud computing seems somehow obvious today. However, the usage of the term "cloud-native" decreased over following years according to Google. But since 2015, the term is more and more frequently used again and gained momentum (see Fig. 1). We have to admit that Google trends is not a very reliable source for research – it is known to be prone for "industrial buzzwords". But, our literature review will show a very similar usage of the term "cloud-native" in research studies (the reader might want to compare Figs. 1 and 5a). Furthermore, our data indicates that the current understanding of the term "cloud-native" seems to have its origin in research and not in industry. It was used and characterized since 2012 in research. And yes, it gets suddenly popular in industry, but not before 2015 (according to Google trends, see Fig. 1). We assume this sudden rise in industry has to do with the current popularity of microservice and container-based approaches. Cloud-

* Corresponding author.
E-mail addresses: nane.kratzke@fh-luebeck.de (N. Kratzke), peter-christian.

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

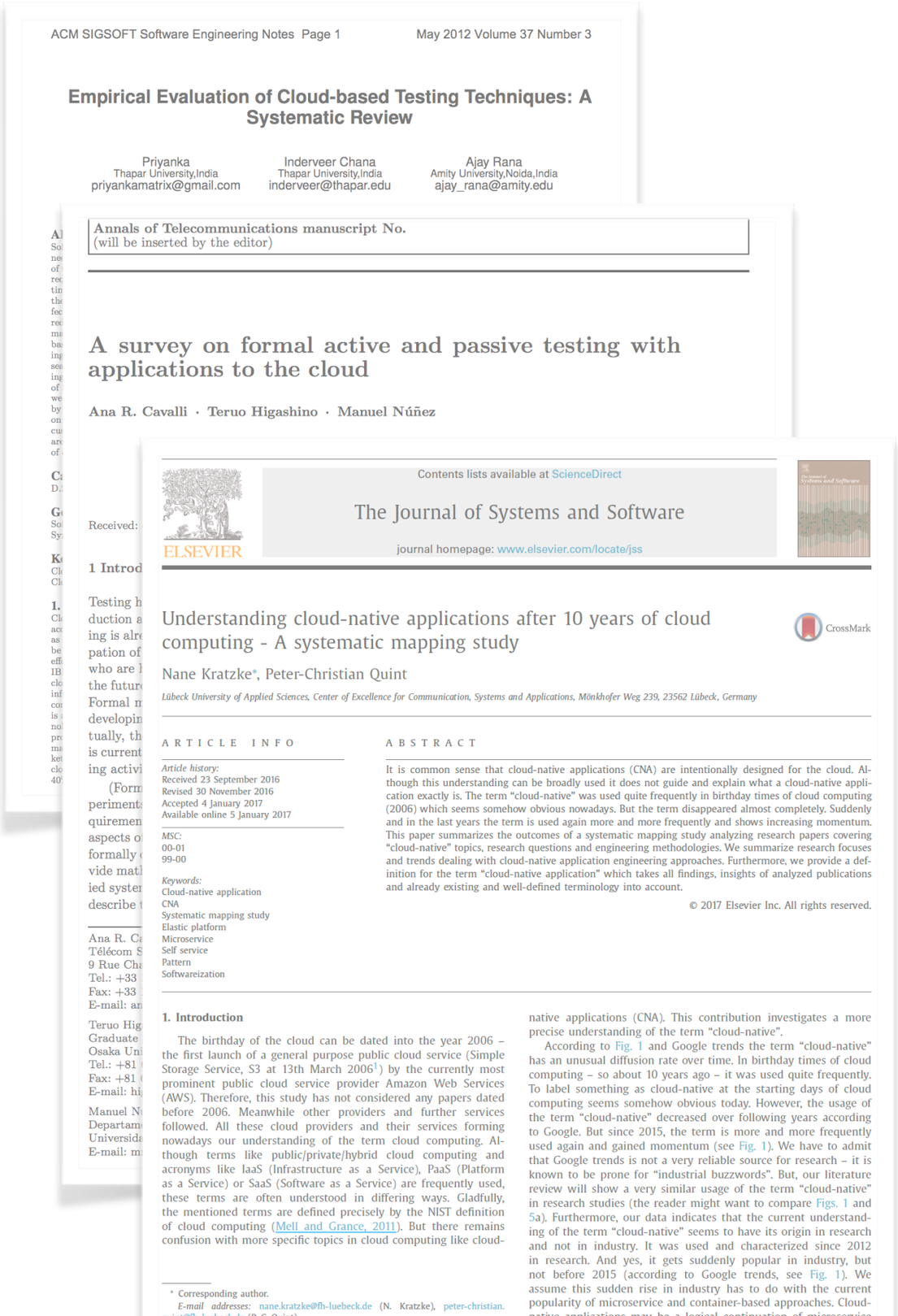
CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Cloud Testing Challenges



ACM SIGSOFT Software Engineering Notes Page 1 May 2012 Volume 37 Number 3

Empirical Evaluation of Cloud-based Testing Techniques: A Systematic Review

Priyanka, Thapar University,India, priyankamatrix@gmail.com
Inderveer Chana, Thapar University,India, inderveer@thapar.edu
Ajay Rana, Amity University,Noida,India, ajay_rana@amity.edu

Annals of Telecommunications manuscript No.
(will be inserted by the editor)

A survey on formal active and passive testing with applications to the cloud

Ana R. Cavalli · Teruo Higashino · Manuel Núñez

Received:

1 Introd

Testing h
duction a
ing is alr
pation of
who are
the futur
Formal n
developin
tually, th
is current
ing activi
(Form
periment
quiremen
aspects o
formally
vide mat
ied syste
describe

Ana R. Ca
Télécom S
9 Rue Cha
Tel.: +33
Fax: +33
E-mail: an

Teruo Hig
Graduate
Osaka Uni
Tel.: +81
Fax: +81
E-mail: hi

Manuel N
Departam
Universida
E-mail: m

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

ELSEVIER

Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study

CrossMark

Nane Kratzke*, Peter-Christian Quint

Lübeck University of Applied Sciences, Center of Excellence for Communication, Systems and Applications, Mönkhofer Weg 239, 23562 Lübeck, Germany

ARTICLE INFO

Article history:
Received 23 September 2016
Revised 30 November 2016
Accepted 4 January 2017
Available online 5 January 2017

MSC:
00-01
99-00

Keywords:
Cloud-native application
CNA
Systematic mapping study
Elastic platform
Microservice
Self service
Pattern
Softwareization

ABSTRACT

It is common sense that cloud-native applications (CNA) are intentionally designed for the cloud. Although this understanding can be broadly used it does not guide and explain what a cloud-native application exactly is. The term "cloud-native" was used quite frequently in birthday times of cloud computing (2006) which seems somehow obvious nowadays. But the term disappeared almost completely. Suddenly and in the last years the term is used again more and more frequently and shows increasing momentum. This paper summarizes the outcomes of a systematic mapping study analyzing research papers covering "cloud-native" topics, research questions and engineering methodologies. We summarize research focuses and trends dealing with cloud-native application engineering approaches. Furthermore, we provide a definition for the term "cloud-native application" which takes all findings, insights of analyzed publications and already existing and well-defined terminology into account.

1. Introduction

The birthday of the cloud can be dated into the year 2006 – the first launch of a general purpose public cloud service (Simple Storage Service, S3 at 13th March 2006[1]) by the currently most prominent public cloud service provider Amazon Web Services (AWS). Therefore, this study has not considered any papers dated before 2006. Meanwhile other providers and further services followed. All these cloud providers and their services forming nowadays our understanding of the term cloud computing. Although terms like public/private/hybrid cloud computing and acronyms like IaaS (Infrastructure as a Service), PaaS (Platform as a Service) or SaaS (Software as a Service) are frequently used, these terms are often understood in differing ways. Gladfully, the mentioned terms are defined precisely by the NIST definition of cloud computing (Mell and Grance, 2011). But there remains confusion with more specific topics in cloud computing like cloud-native applications (CNA). This contribution investigates a more precise understanding of the term "cloud-native".

According to Fig. 1 and Google trends the term "cloud-native" has an unusual diffusion rate over time. In birthday times of cloud computing – so about 10 years ago – it was used quite frequently. To label something as cloud-native at the starting days of cloud computing seems somehow obvious today. However, the usage of the term "cloud-native" decreased over following years according to Google. But since 2015, the term is more and more frequently used again and gained momentum (see Fig. 1). We have to admit that Google trends is not a very reliable source for research – it is known to be prone for "industrial buzzwords". But, our literature review will show a very similar usage of the term "cloud-native" in research studies (the reader might want to compare Figs. 1 and 5a). Furthermore, our data indicates that the current understanding of the term "cloud-native" seems to have its origin in research and not in industry. It was used and characterized since 2012 in research. And yes, it gets suddenly popular in industry, but not before 2015 (according to Google trends, see Fig. 1). We assume this sudden rise in industry has to do with the current popularity of microservice and container-based approaches. Cloud-

* Corresponding author.
E-mail addresses: nane.kratzke@fh-luebeck.de (N. Kratzke), peter-christian.

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

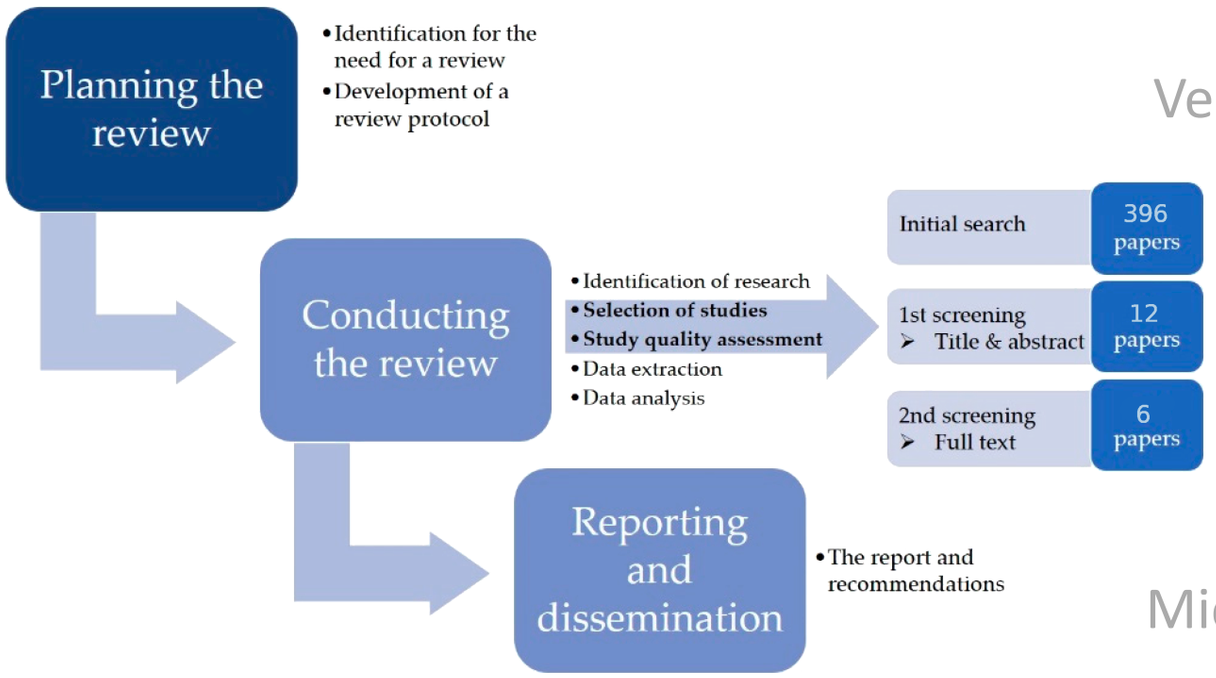
CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Literature Review on Testing for the Cloud



Search keywords formula (DBLP & Google Scholar):

CLOUD ∧
(GENETIC ∨ COMBINATORIAL ∨ SIMULATED
ANNEALING ∨ TABU SEARCH ∨ PROFILING ∨
SLICING ∨ COVERAGE ∨ COEVOLUTION ∨
MUTATION ∨ MORPH ∨ HEALING ∨ SELF REPAIR
∨ HILL CLIMBING ∨ SEARCH BASED **)**

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

CNAs Adaptations (e.g., migration)

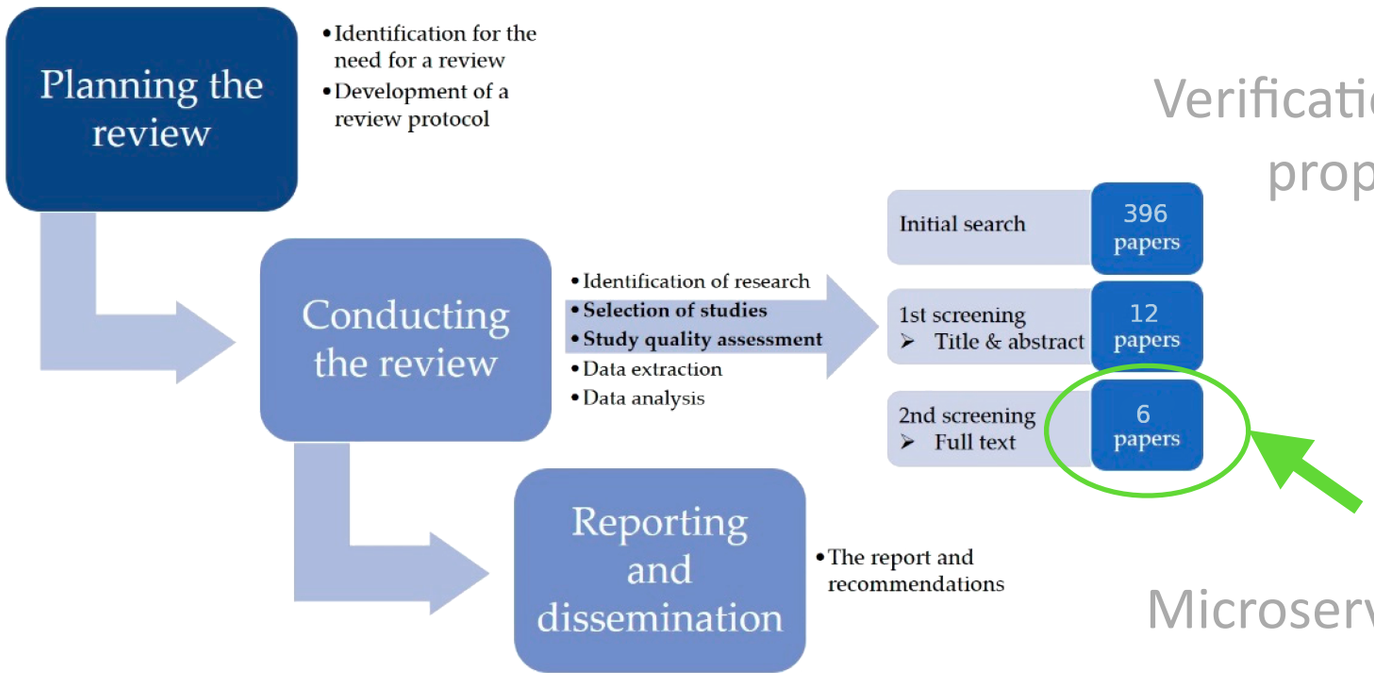
Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

https://github.com/GoDieNow/CloudSBSTPerspectives

Literature Review on Testing for the Cloud



Only 6 papers addressing Cloud Challenges

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

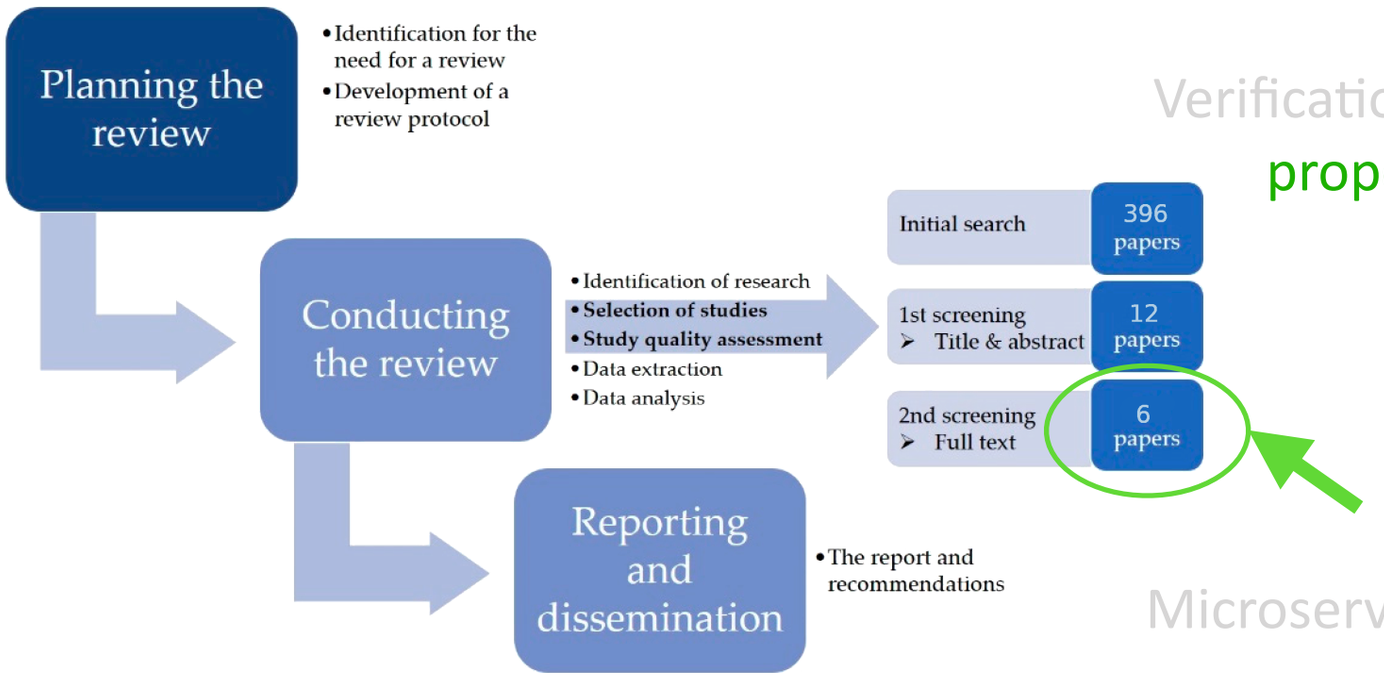
CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Literature Review on Testing for the Cloud



Only 6 papers addressing Cloud Challenges

SBST literature in the cloud field focused on the optimization of testing frameworks and the stress of basic CNA properties (elasticity and resilience)

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Future Directions

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Future Directions

Test case generation for the V&V of CNA properties is still a widely unexplored field

Metrics and tools should be designed in order to guide the generation and the quality assessment of corresponding tests.

SBST strategies, encapsulating smart mechanisms able to assess the inputs that most impact, affect, interacts with the coordination and evolution of the microservices architecture.

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

Future Directions

SBST techniques have a perfect match the assessment/verification of CNA adaptation and migration challenges

Profiling and current monitoring techniques can be combined with SBST strategies to better verify/monitor flow bottlenecks in the execution time

… helping developers achieve the typical execution-time requirements of CNA.

Cloud Challenges

Verification and Validation (V&V) of Cloud-native (CNA) properties (e.g., composition, elastic scalability)

CNAs Adaptations (e.g., migration)

Microservices Evolution (e.g., providing an architectural view of microservices evolution)

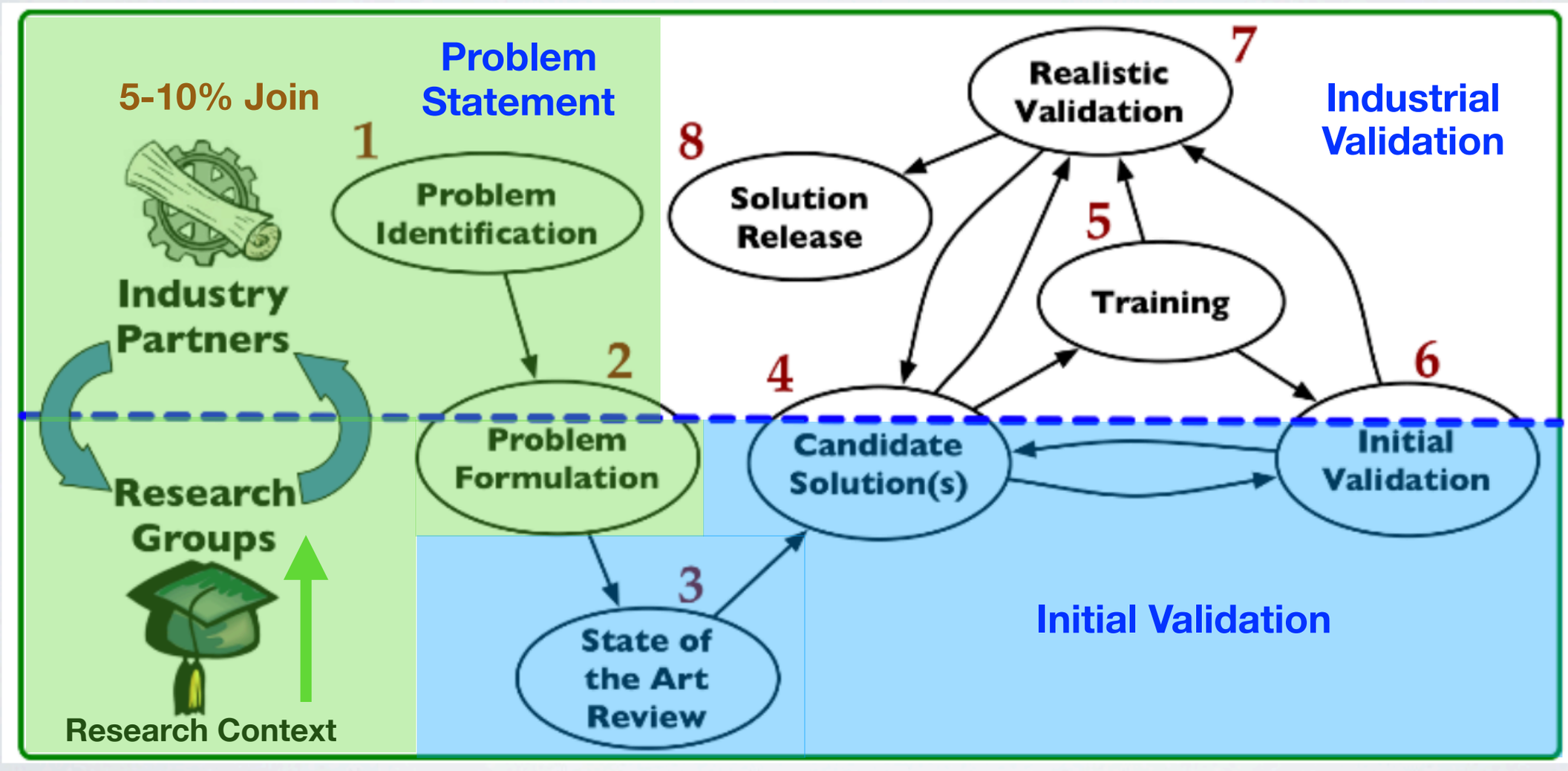
Local v.s. Global Verification and Validation (e.g., V&V local microservices v.s. all microservices)

Execution Time (Efficient and Effective)

How to come up with a Vision?

"..there are guidelines..."

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

Outline

PART I

1) Research Context and Motivation

PART II

2) Literature review on:

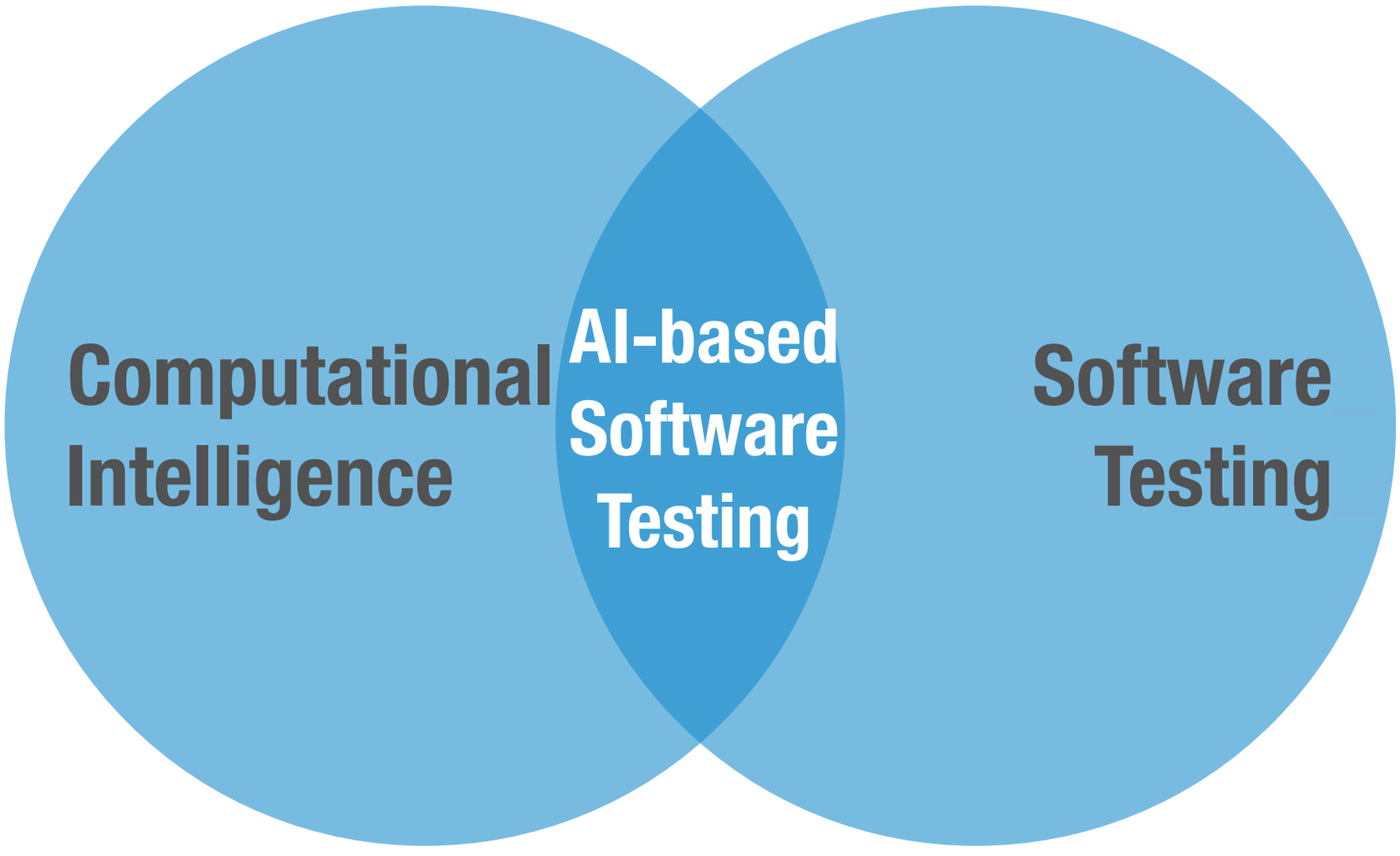
- Testing Challenges in the Cloud
- Automated Testing (**SBST**)
- Cloud-based testing demonstration

3) Future Work

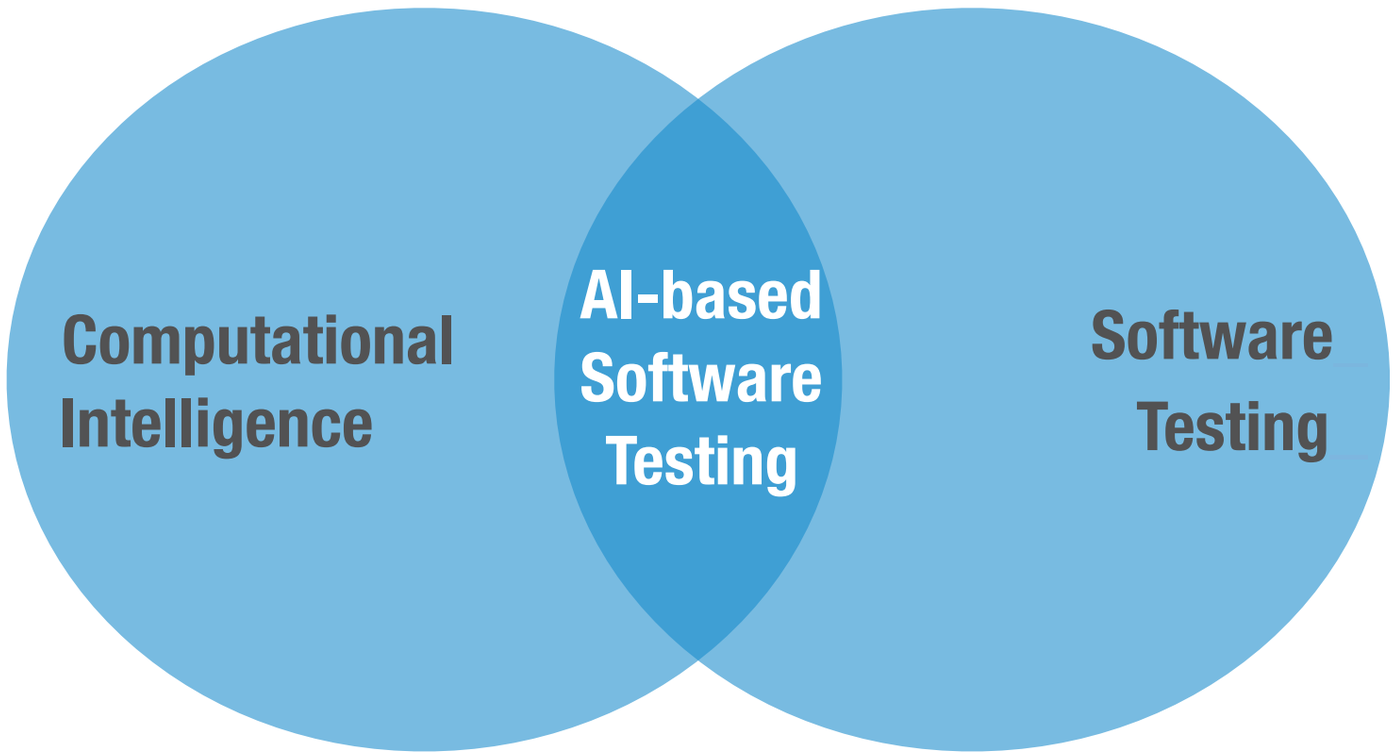
Search-Based Software Testing

=

Automated Testing



Search-Based Software Testing



[Domingos2015 “The Master Algorithm”]

Tribe	Origin	Master Algorithm
Symbolists	Logic, philosophy	Inverse deduction
Connectionists	Neuroscience	Back-Propagation
Evolutionaries	Evolutionary biology	Evolutionary Algorithms
Bayesian	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

Genetic Algorithms

Why SBST?

Project = Apache commons BCEL
Class = Pass2Verifier.java

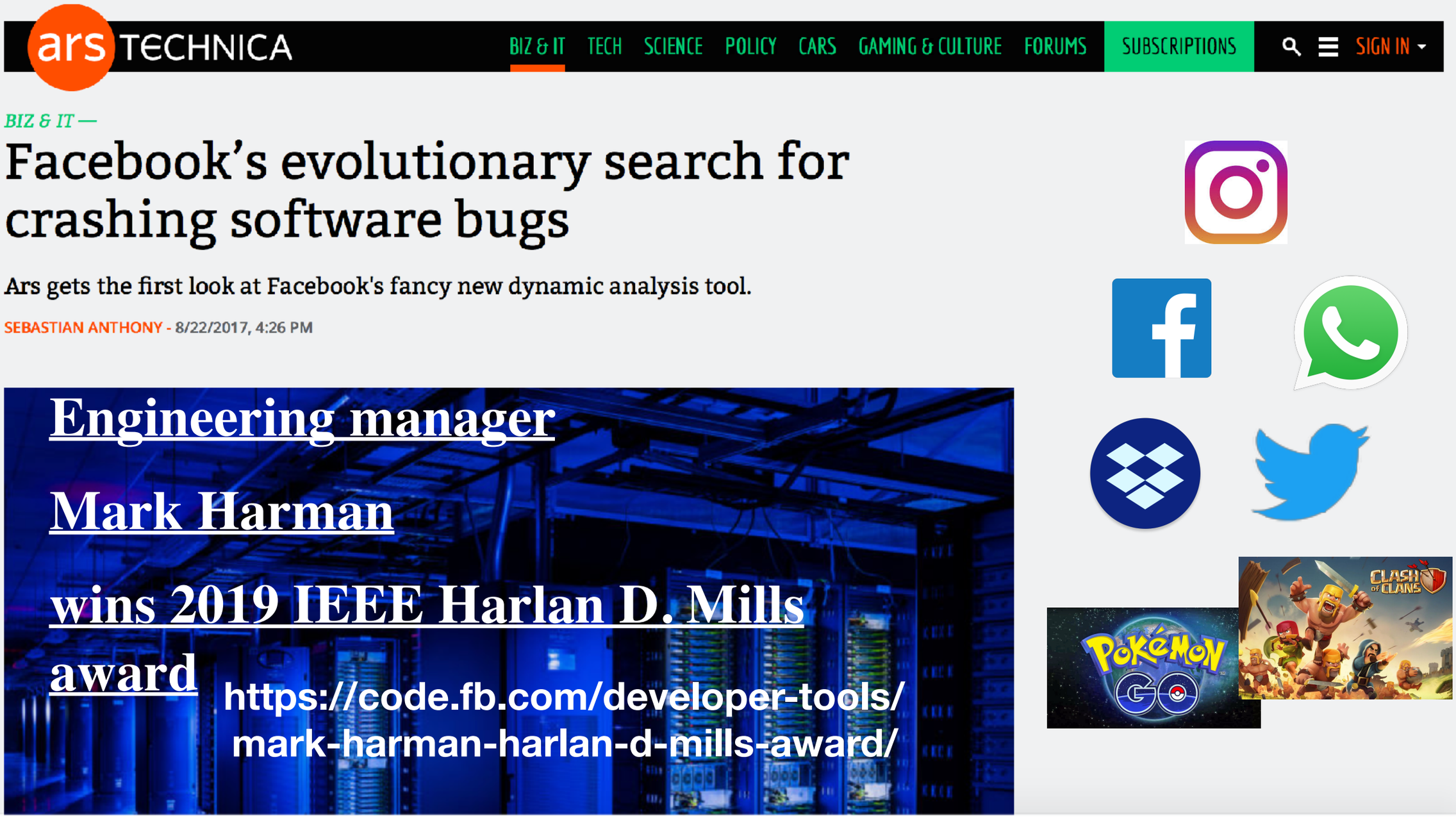
```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 */
package org.apache.bcel.verifier.statics;


import java.util.HashMap;
import java.util.HashSet;
import java.util.Locale;
import java.util.Map;
import java.util.Set;

import org.apache.bcel.Const;
import org.apache.bcel.Constants;
import org.apache.bcel.Repository;
import org.apache.bcel.classfile.Attribute;
import org.apache.bcel.classfile.ClassFormatException;
import org.apache.bcel.classfile.Code;
import org.apache.bcel.classfile.CodeException;
import org.apache.bcel.classfile.Constant;
import org.apache.bcel.classfile.ConstantClass;
import org.apache.bcel.classfile.ConstantDouble;
```

IMPACT:
The Sapienz Project at Facebook

https://arstechnica.com/information-technology/2017/08/facebook-dynamic-analysis-software-sapienz/



What You Can Do?

Class Under Test (CUT)

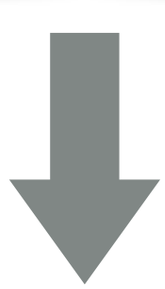
```
class Triangle {
 int a, b, c; //sides
 String type = "NOT_TRIANGLE";

 Triangle (int a, int b, int c){…}

 void computeTriangleType() {
  1.  if (a == b) {
  2.      if (b == c)
  3.          type = "EQUILATERAL";
          else
  4.          type = "ISOSCELES";
      } else {
  5.      if (a == c) {
  6.            type = "ISOSCELES";
         } else {
  7.          if (b == c)
  8.                type = “ISOSCELES”;
              else
  9.                type = “SCALENE”;
         }
      }
}
```

Test Case

```
@Test
public void test(){
  // Constructor (init)
  // Method Calls
  // Assertions  (check)
}
```

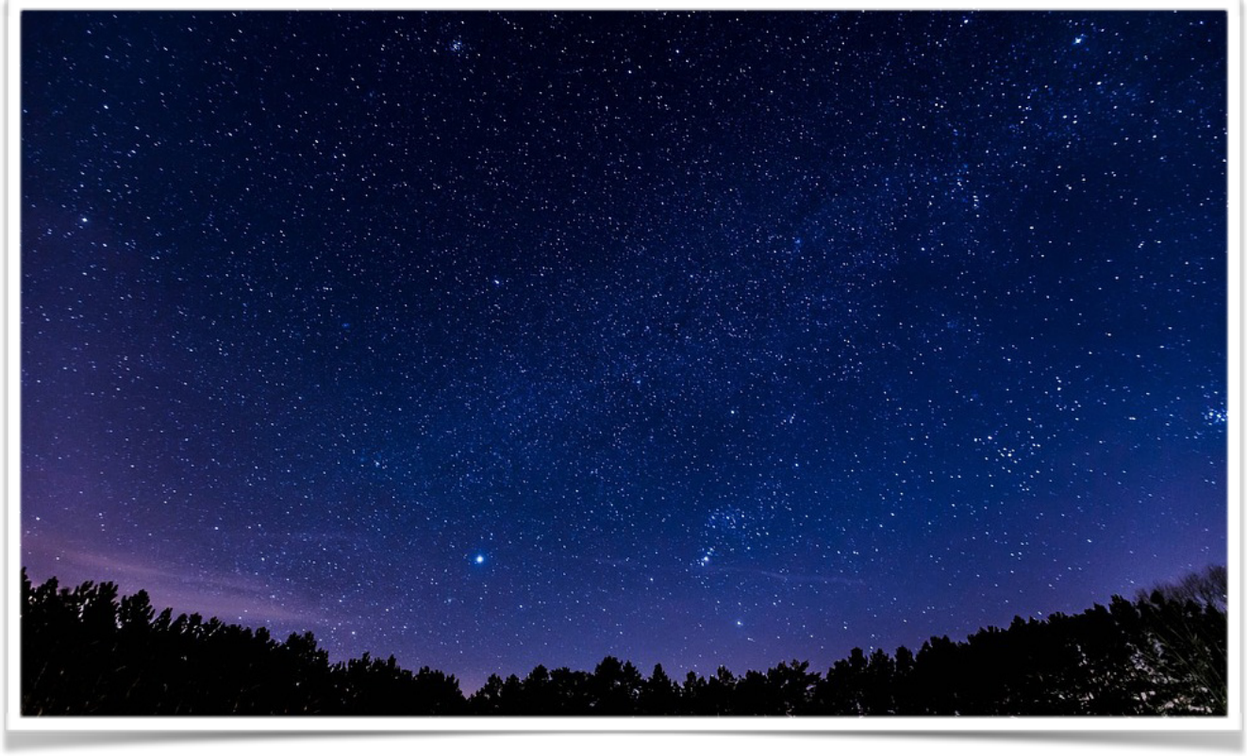


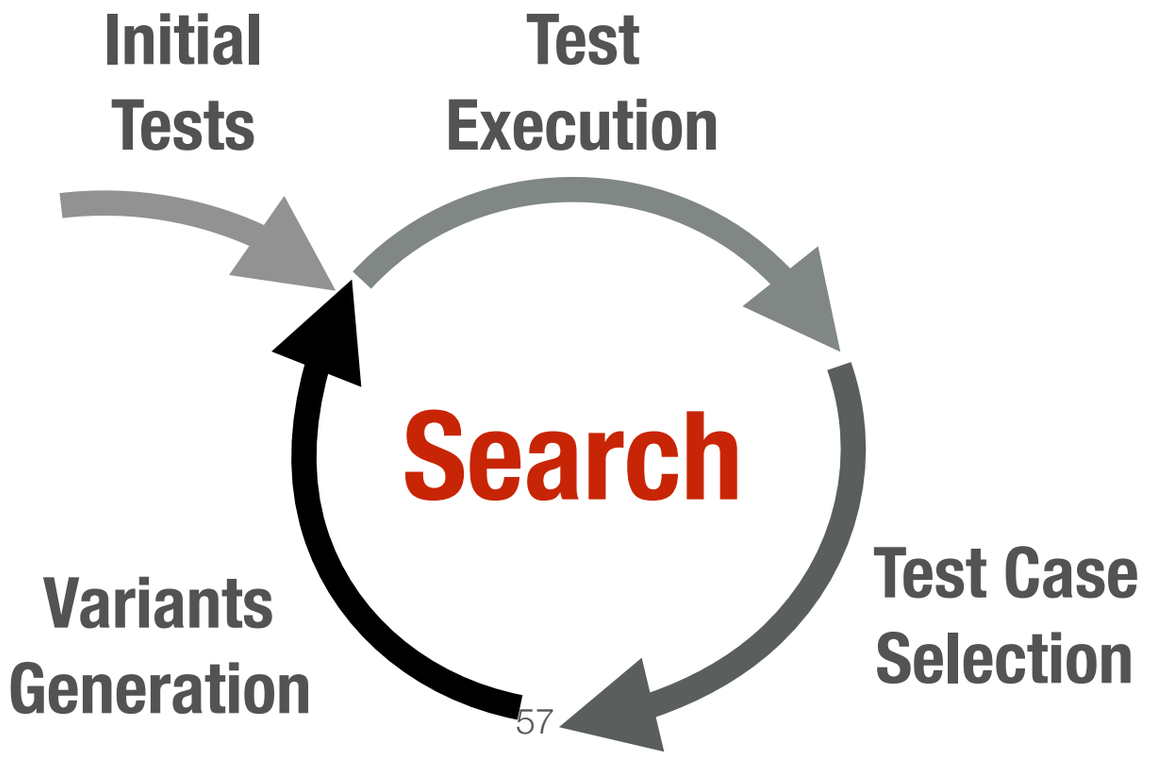
```
@Test
public void test(){
 Triangle t = new Triangle (1,2,3);
 t.computeTriangleType();
 String type = t.getType();
 assertTrue(type.equals(“SCALENE”));
}
```

http://www.evosuite.org

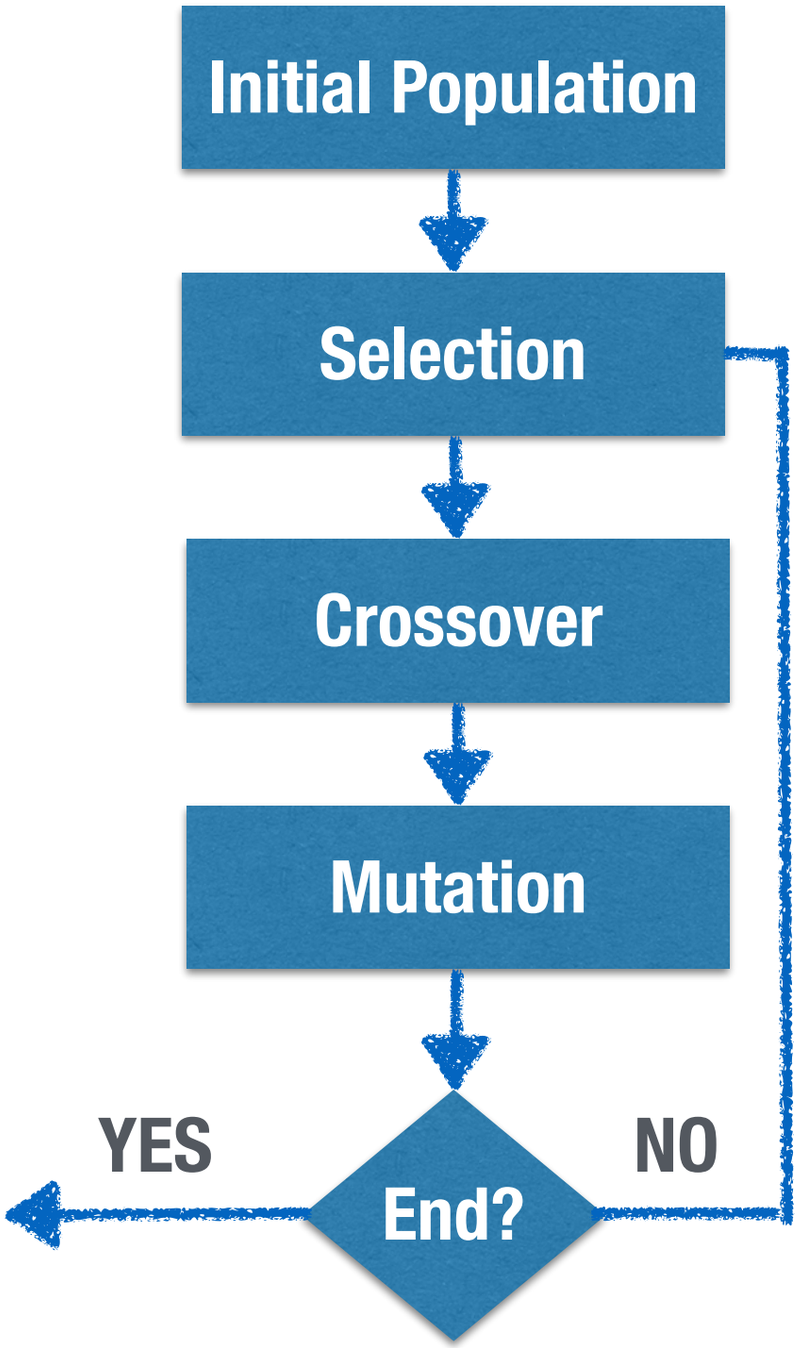
How Does It Works?

Infinite Test Space





How Does It Works?



Genetic Algorithms

Natural selection: Individuals that best fit the natural environment survive (worst die)

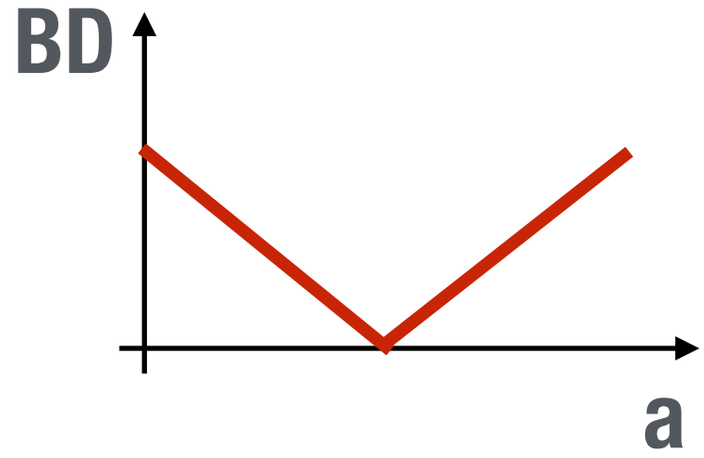
Reproduction: survived individuals generate offspring (next generation)

Mutation: offspring inherits properties of the parents (with some mutations)

Iteration: generation by generation, the new offspring fits better the environment than their parents

Branch Distance

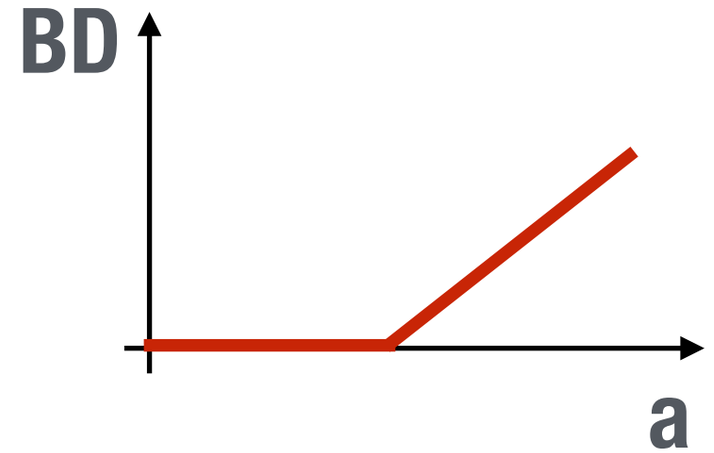
If (a == b) → abs(a-b)



If (a <= b) →

0 if a<=b

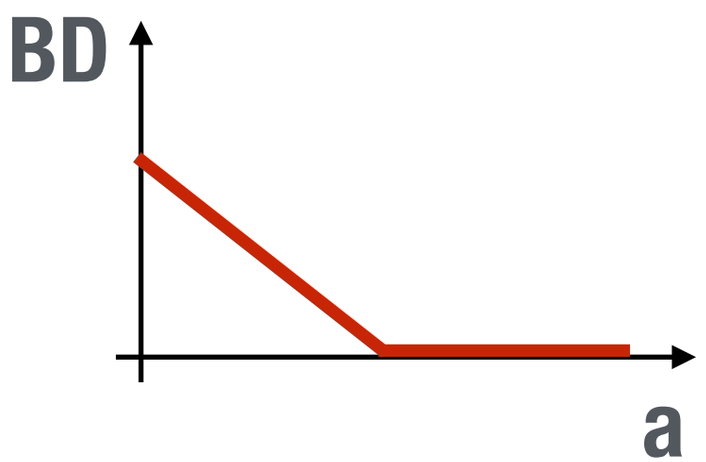
abs(a-b) otherwise



If (a => b) →

0 if >=b

abs(a-b) otherwise



SBST

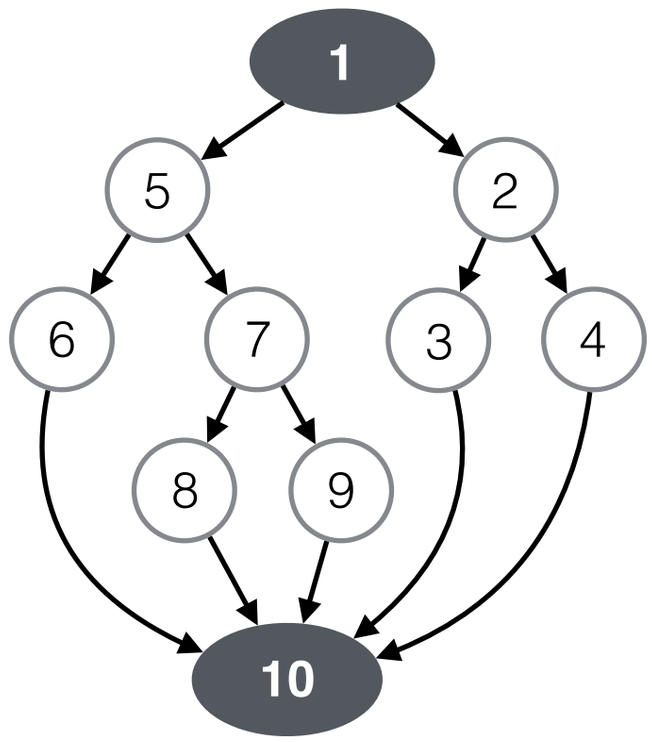
Class Under Test (CUT)

```
class Triangle {
 int a, b, c; //sides
 String type = "NOT_TRIANGLE";

 Triangle (int a, int b, int c){…}

 void computeTriangleType() {
  1.  if (a == b) {
  2.      if (b == c)
  3.          type = "EQUILATERAL";
          else
  4.          type = "ISOSCELES";
      } else {
  5.      if (a == c) {
  6.            type = "ISOSCELES";
           } else {
  7.          if (b == c)
  8.                type = “ISOSCELES”;
                else
  9.              type = “SCALENE”;
           }
      }
}
```

Control Flow Graph



SBST

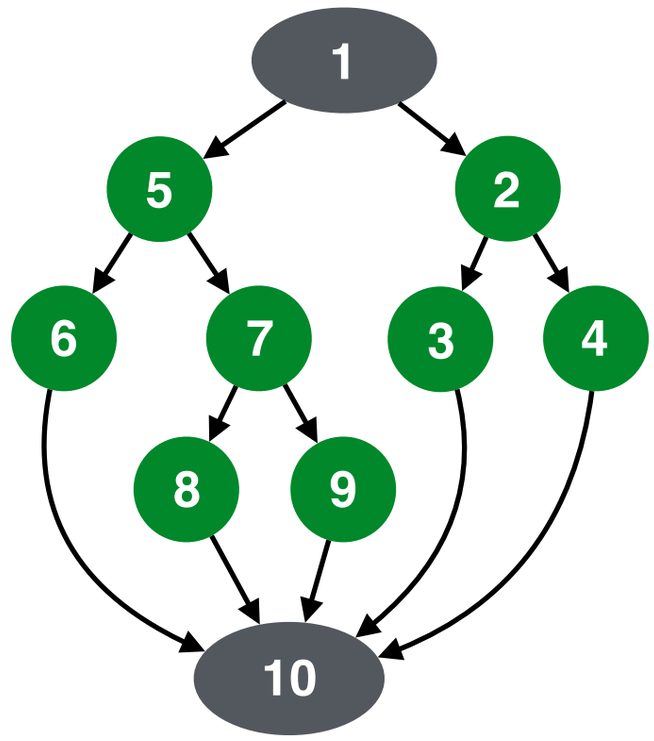
Class Under Test (CUT)

```
class Triangle {
 int a, b, c; //sides
 String type = "NOT_TRIANGLE";

 Triangle (int a, int b, int c){…}

 void computeTriangleType() {
  1.  if (a == b) {
  2.      if (b == c)
  3.          type = "EQUILATERAL";
          else
  4.          type = "ISOSCELES";
      } else {
  5.      if (a == c) {
  6.          type = "ISOSCELES";
          } else {
  7.          if (b == c)
  8.                  type = “ISOSCELES”;
                else
                  type = “SCALENE”;
          }
      }
}
```

Control Flow Graph



Goal: Covering as many code elements as possible

How Much to Test?

Statement coverage

Targets = {1, 2, 3,4 ,5 ,6, 7, 8, 9,10}

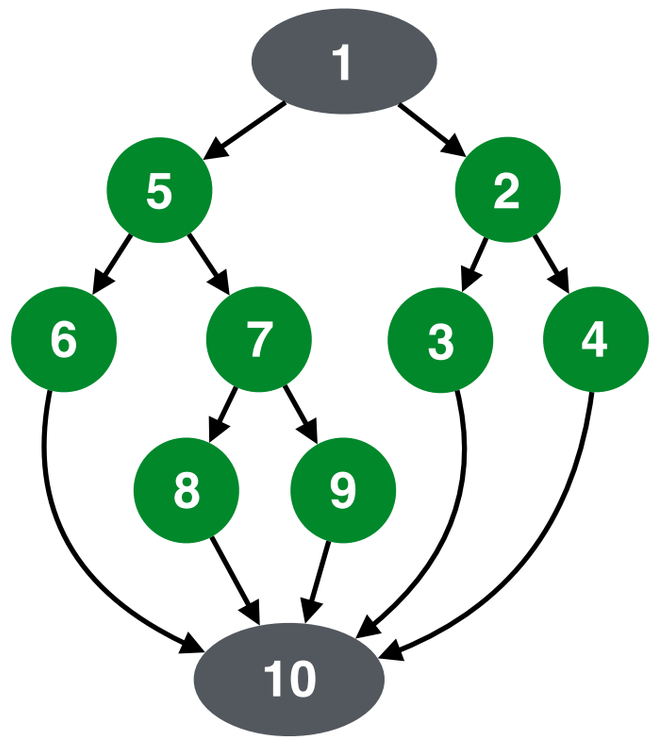
Branch coverage

Targets = {<1,5>, <1,2>,<5,6>, <5,7>,
<2,3>, <2,4>, <6,10>, <7,8>, <7,9>,
<3,10>, <4,10>, <8,10>, <9,10>}

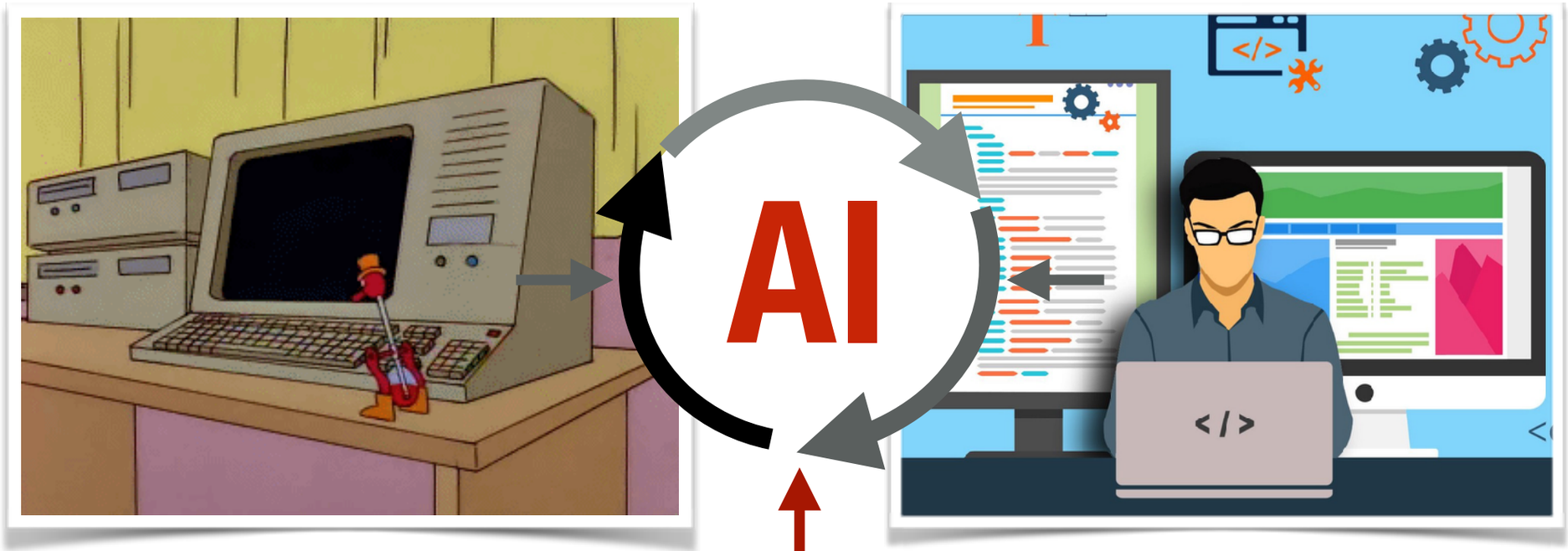
Path coverage

Targets = {<1,5,6,10>, <1,5,7,8,10>,
<1,5,7,9,10>, <1,2,3,10>, <1,2,4,10>}

Control Flow Graph



Automated Test Design and Execution



Random Fuzzer

Human Developers
(Are the tests good?)

Fitness
Function

Running Example

```
class Triangle {
    private double side1, side2, side3;
    private String type = “NOT_A_TRIANGLE”;

    public Triangle (double a, double b, double c){…}
    private void checkRightAngle() {…}
    public void computeTriangleType() {…}
    private boolean isTriangle() {…}
}
```

Running Example

```
class Triangle {
   int a, b, c; //sides
   int type = NOT_A_TRIANGLE;

   Triangle (int a, int b, int c){…}
   void checkRightAngle() {…}
   void computeTriangleType() {…}
   boolean isTriangle() {…}
   public static void main (String args[]) {…}
}
```

Class under test

Initial Test Cases

```
@Test
public void test(){
   Triangle c = new Triangle(8.0, 2.1
   c.computeTriangleType();
   assertTrue
}
```

```
@Test
public void test(){
   Triangle c = new Triangle(1
   c.computeTriangleType();
   assertTrue(c.getTriangleTyp
}
```

```
@Test
public void test(){
   Triangle c = new Triangle(1.0, 2.1, 4.2);
   c.computeTriangleType();
   assertTrue(c.getTriangleType, …);
}
```

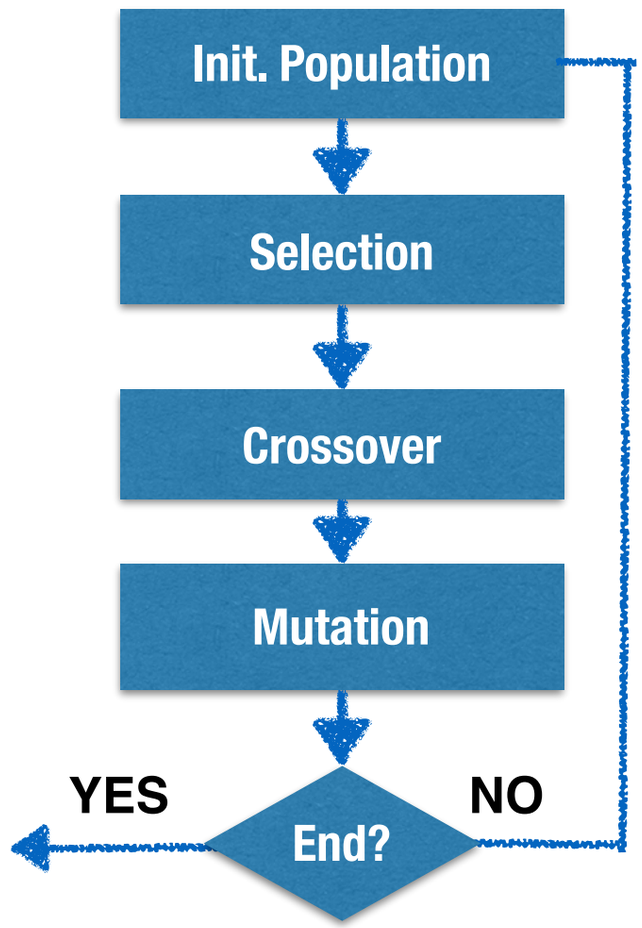
```
@Test
public void test(){
   Triangle c = new Triangle(-2.0, 12.0, 0);
   c.computeTriangleType();
   assertTrue(c.getTriangleType, …);
}
```

Iterating (generations)

```
class Triangle {

 void computeTriangleType() {
1. if (isTriangle()){
2.   if (side1 == side2) {
3.       if (side2 == side3)
4.           type = "EQUILATERAL";
         else
5.           type = "ISOSCELES";
     } else {
6.       if (side1 == side3) {
7.           type = "ISOSCELES";
         } else {
8.           if (side2 == side3)
9.               type = “ISOSCELES”;
             else
10.              checkRightAngle();
             }
         }
     }// if isTriangle()
  }}
```

Target



Branch Distance

Branch_distance(P(x), t)

Given the first control node where the execution diverges from the target t, the predicate at such node is converted to a distance (from taking the desired branch), normalised between 0 and 1.

Such a distance measures how fare the test case is from taking the desired branch. For boolean and numerical variables a, b:

Condition c = atomic predicate	Distance BD(c) = d / (d + 1)
a	d = {0 if a == true; K otherwise}
!a	d = {K if a == true; 0 otherwise}
a == b	d = {0 if a == b; abs(a - b) + K otherwise}
a != b	d = {0 if a != b; K otherwise}
a < b	d = {0 if a < b; a - b + K otherwise}
a <= b	d = {0 if a <= b; a - b + K otherwise}
a > b	d = {0 if a > b; b - a + K otherwise}
a >= b	d = {0 if a >= b; b - a + K otherwise}

Branch Distance for Strings

Branch_distance(P(x), t)

For string variables a and b, the branch distance is computed using the following rules:

Condition c = atomic predicate	Distance BD(c) = d / (d + 1)
a == b	d = {0 if a == b; edit_dist(a, b) + K otherwise}
a != b	d = {0 if a != b; K otherwise}
a < b	d = {0 if a < b; a[j] - b[j] + K otherwise}
a <= b	d = {0 if a <= b; a[j] - b[j] + K otherwise}
a > b	d = {0 if a > b; b[j] - a[j] + K otherwise}
a >= b	d = {0 if a >= b; b[j] - a[j] + K otherwise}

Example of edit distance: edit_dist("abcd", "abbb")=2

Branch Distance

Branch distance rules for composite predicate

Condition c = composite predicate	Distance BD(c) = d / (d + 1)
!p	Negation is propagated inside p
p & q	d = d(p) + d(q)
p | q	d = min(d(p), d(q))
p XOR q = p & !q | !p & q	d = min(d(p)+d(!q), d(!p)+d(q))

How to normalise the branch distance?

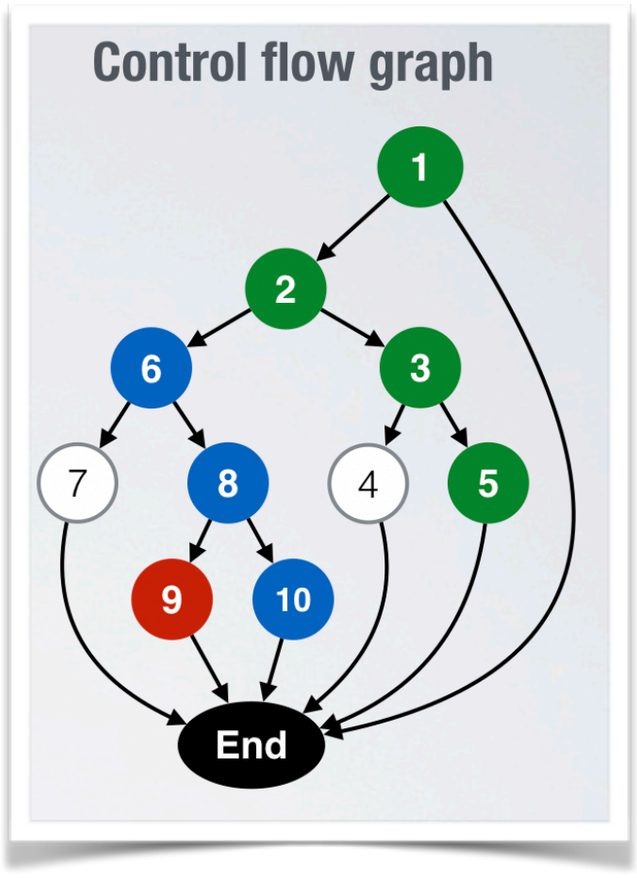
branch_distance(t) = d/(d+1)

where d is computed according to the distance rules reported in the previous tables

Fitness Function

For statement (or branch) coverage, given a specific coverage target t, a widely used fitness function (to be minimised) is:

$$f(x) = approach\_level(P(x), t) + branch\_distance(P(x),t)$$



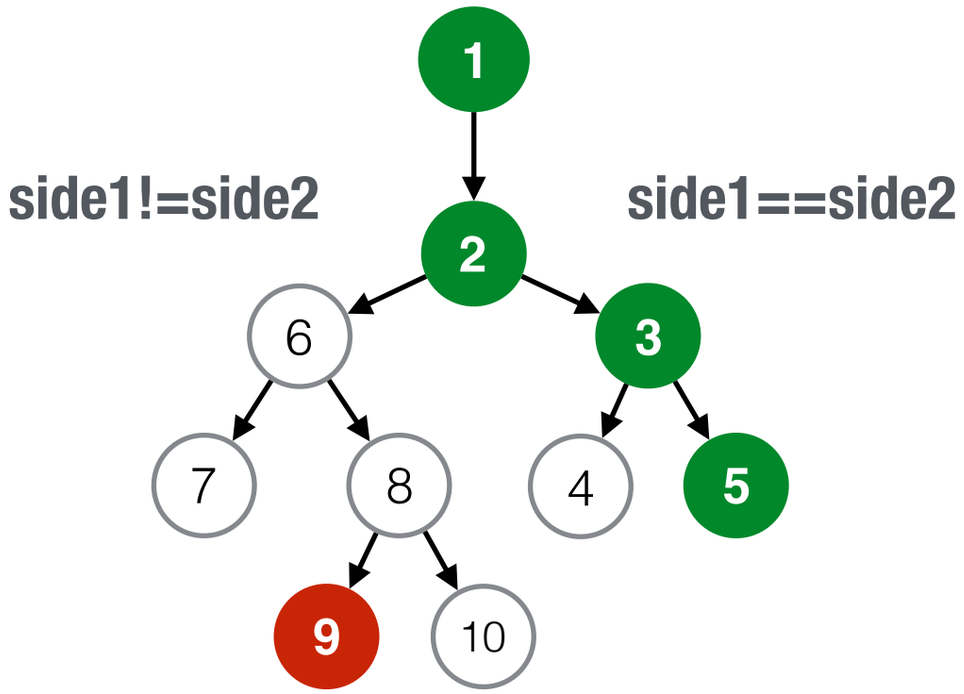
+

Fitness Function

```
class Triangle {

 void computeTriangleType() {
1. if (isTriangle()){
2.   if (side1 == side2) {
3.        if (side2 == side3)
4.            type = "EQUILATERAL";
          else
5.            type = "ISOSCELES";
      } else {
6.        if (side1 == side3) {
7.            type = "ISOSCELES";
          } else {
8.            if (side2 == side3)
9.                type = “ISOSCELES”;
              else
10.               checkRightAngle();
              }
          }
      }// if isTriangle()
  }}
```

Target



d(2 != 2) = 1
BD (2 != 2) = 1/ (1+1) = 0.5
f(x1) = 2 + 0.5 = 2.5

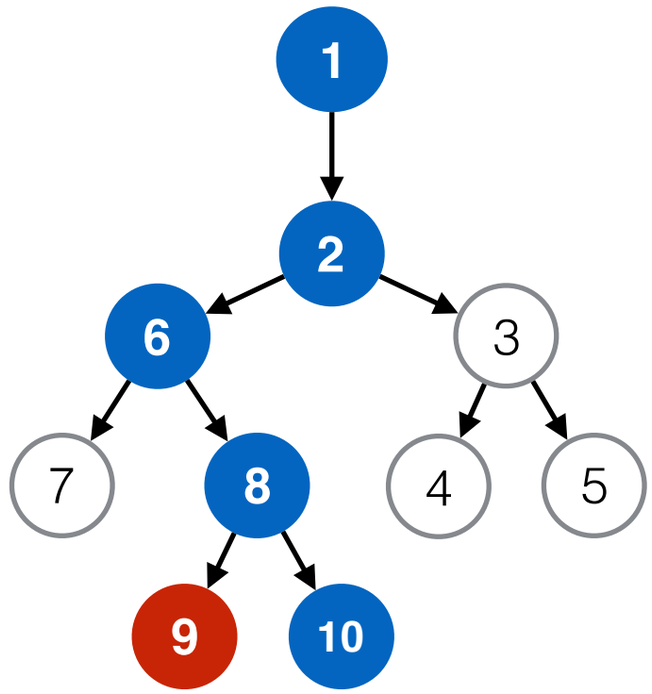
x1 = (2, 2, 3)	Path(x1) = <1, 2, 3, 5>	AL=2	f = 2.5
x2 = (2, 3, 5)	Path(x2) = <1, 2, 6, 8, 10>	AL=0	
x3 = (1, 2, 10)	Path(x3) = <1, 2, 6, 8, 10>	AL=0	

Fitness Function

```
class Triangle {

 void computeTriangleType() {
1. if (isTriangle()){
2.   if (side1 == side2) {
3.        if (side2 == side3)
4.            type = "EQUILATERAL";
          else
5.            type = "ISOSCELES";
      } else {
6.        if (side1 == side3) {
7.            type = "ISOSCELES";
          } else {
8.            if (side2 == side3)
9.                type = “ISOSCELES”;
              else
10.               checkRightAngle();
              }
          }
      }// if isTriangle()
  }}
```

Target



d(3 == 5) = abs(3-5) = 2
BD (3 == 5) = 2 / (2+1) = 0.66
f(Ch1) = 0 + 0.66 = 0.66

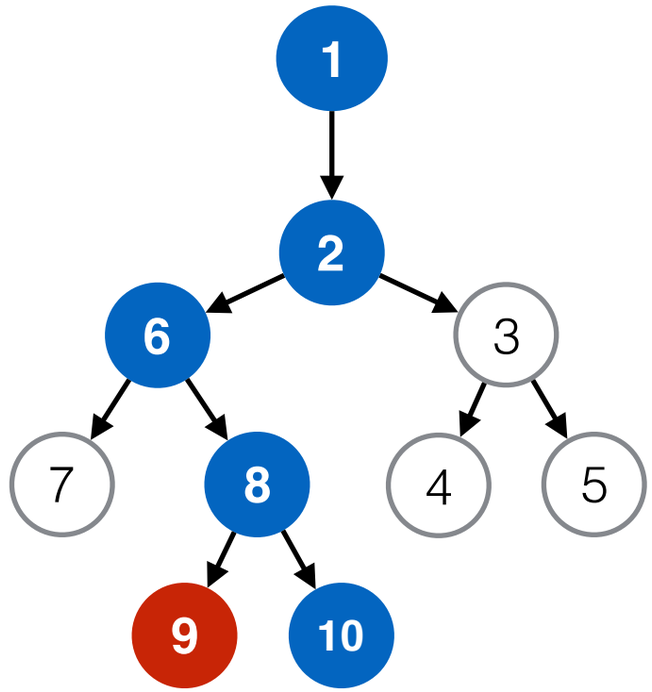
x1 = (2, 2, 3)	Path(x1) = <1, 2, 3, 5>	AL=2	f = 2.5
x2 = (2, 3, 5)	Path(x2) = <1, 2, 6, 8, 10>	AL=0	f = 0.66
x3 = (1, 2, 10)	Path(x3) = <1, 2, 6, 8, 10>	AL=0	

Fitness Function

```
class Triangle {

 void computeTriangleType() {
1. if (isTriangle()){
2.   if (side1 == side2) {
3.        if (side2 == side3)
4.            type = "EQUILATERAL";
          else
5.            type = "ISOSCELES";
      } else {
6.        if (side1 == side3) {
7.            type = "ISOSCELES";
          } else {
8.            if (side2 == side3)
9.                type = “ISOSCELES”;
              else
10.               checkRightAngle();
              }
          }
      }// if isTriangle()
  }}
```

Target



d(3 == 5) = abs(3-5) = 2
BD (3 == 5) = 2 / (2+1) = 0.66
f(Ch1) = 0 + 0.66 = 0.66

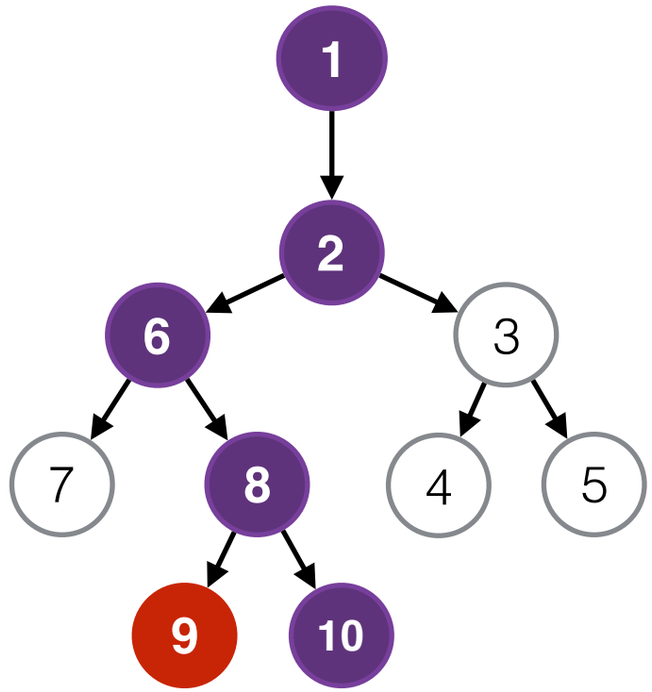
x1 = (2, 2, 3)	Path(x1) = <1, 2, 3, 5>	AL=2	f = 2.5
x2 = (2, 3, 5)	Path(x2) = <1, 2, 6, 8, 10>	AL=0	f = 0.66
x3 = (1, 2, 10)	Path(x3) = <1, 2, 6, 8, 10>	AL=0	f = ?

Fitness Function

```
class Triangle {

 void computeTriangleType() {
1. if (isTriangle()){
2.   if (side1 == side2) {
3.        if (side2 == side3)
4.            type = "EQUILATERAL";
          else
5.            type = "ISOSCELES";
      } else {
6.        if (side1 == side3) {
7.            type = "ISOSCELES";
          } else {
8.            if (side2 == side3)
9.                type = “ISOSCELES”;
              else
10.               checkRightAngle();
              }
          }
      }// if isTriangle()
  }}
```

Target



d(2 == 10) = abs(2-10) = 8
BD (2 == 10) = 8 / (8+1) = 0.89
f(x3) = 0 + 0.89 = 0.89

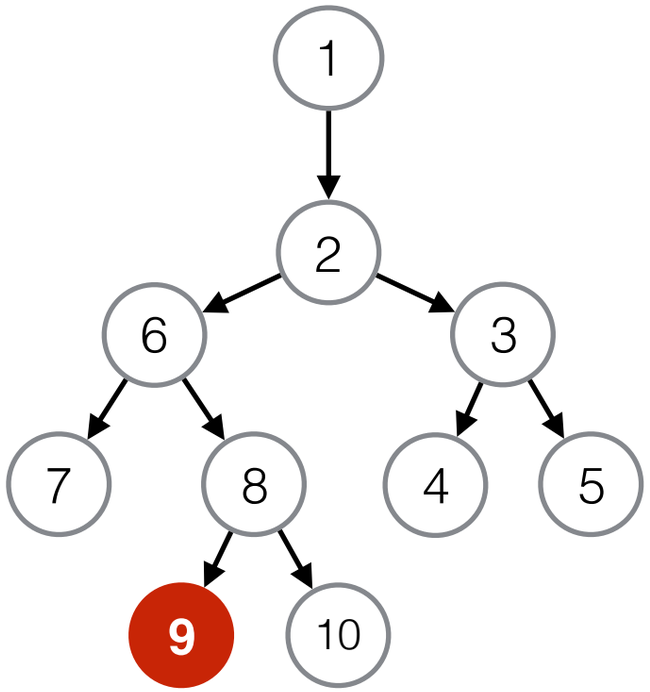
x1 = (2, 2, 3)	Path(x1) = <1, 2, 3, 5>	AL=2	f = 2.5
x2 = (2, 3, 5)	Path(x2) = <1, 2, 6, 8, 10>	AL=0	f = 0.66
x3 = (1, 2, 10)	Path(x3) = <1, 2, 6, 8, 10>	AL=0	f = 0.89

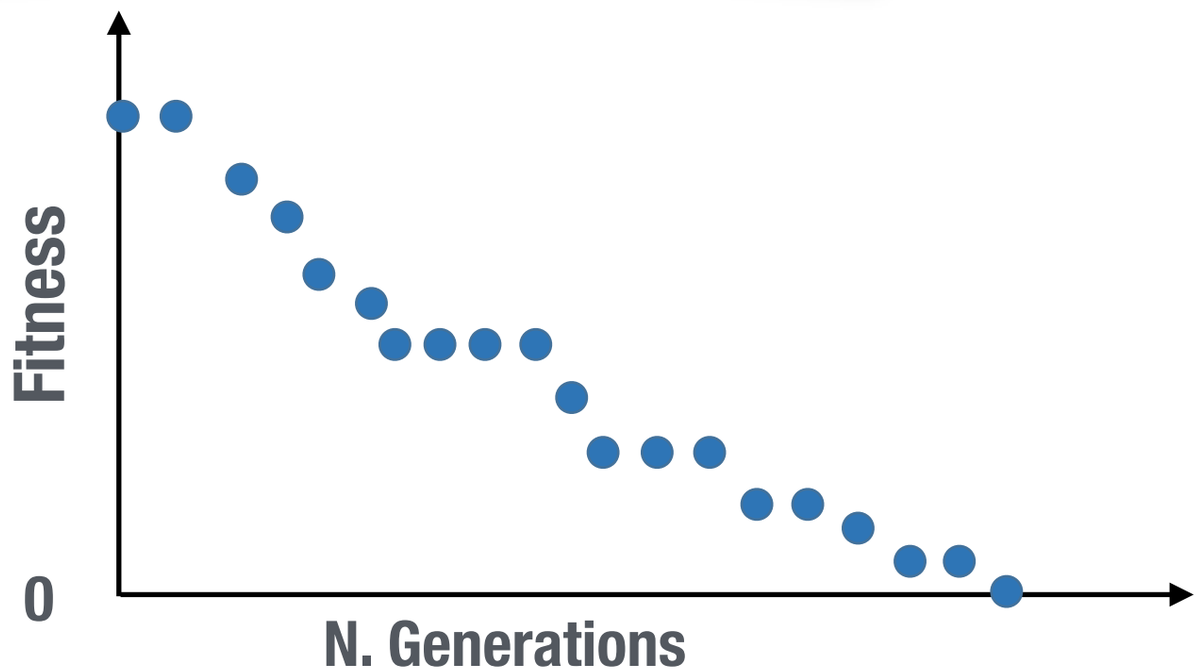
Running Example

```
class Triangle {

 void computeTriangleType() {
   if (isTriangle()){
     if (side1 == side2) {
          if (side2 == side3)
              type = "EQUILATERAL";
          else
              type = "ISOSCELES";
      } else {
          if (side1 == side3) {
              type = "ISOSCELES";
           } else {
             if (side2 == side3)
                 type = “ISOSCELES”;
             else
                checkRightAngle();
                }
          }
      }// if isTriangle()
}}
```

Target



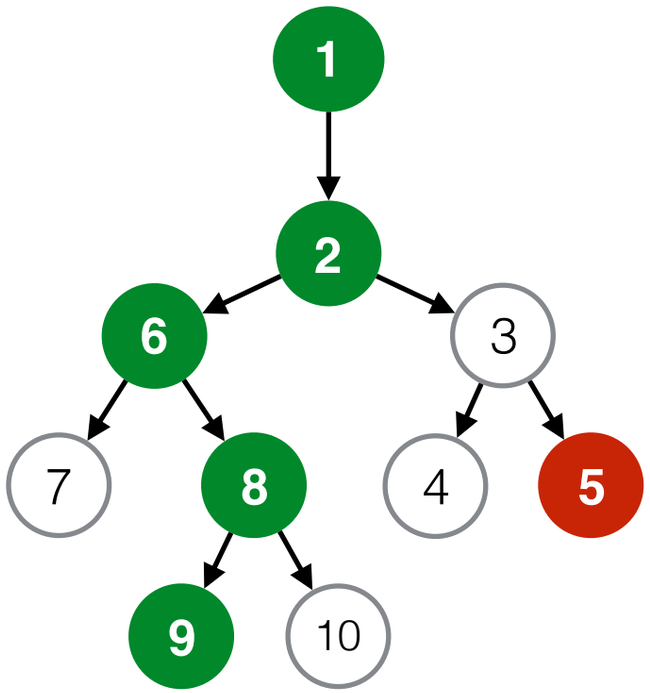


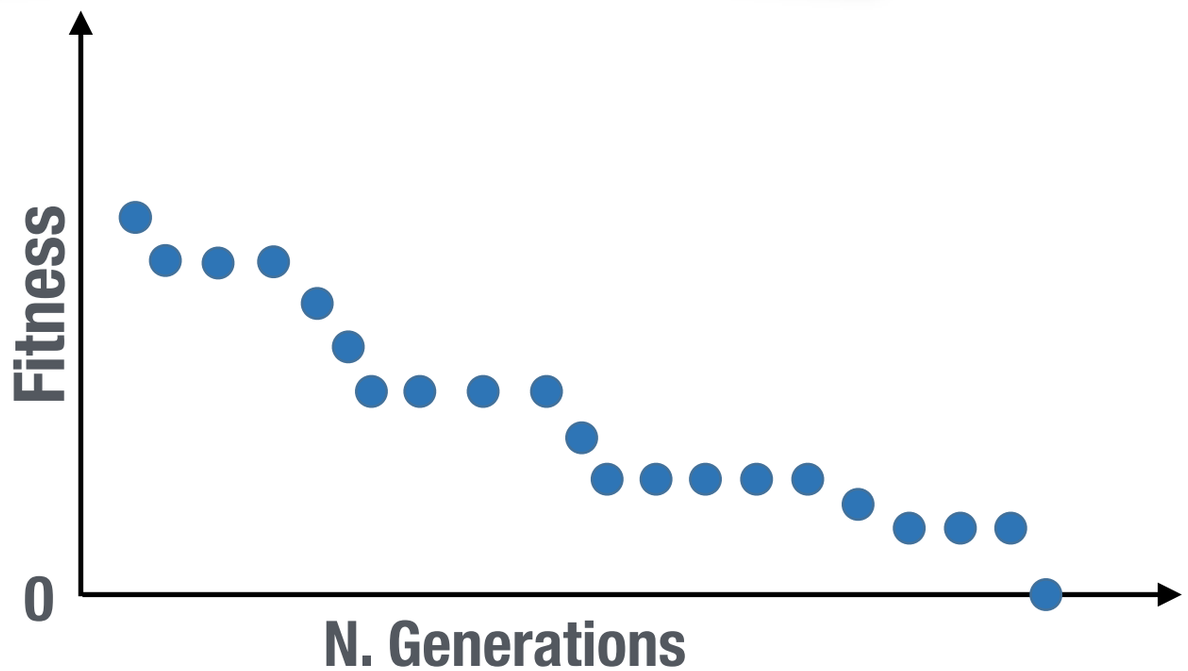
Running Example

```
class Triangle {

 void computeTriangleType() {
   if (isTriangle()){
     if (side1 == side2) {
          if (side2 == side3)
              type = "EQUILATERAL";
          else
              type = "ISOSCELES";
      } else {
          if (side1 == side3) {
              type = "ISOSCELES";
           } else {
             if (side2 == side3)
                 type = “ISOSCELES”;
             else
                checkRightAngle();
                }
          }
      }// if isTriangle()
}}
```

Target





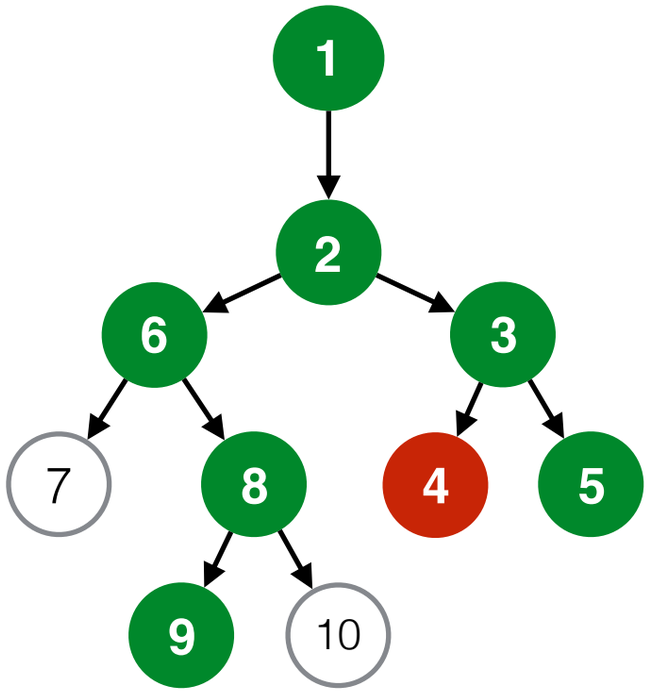
TC1 = (4,3,3)

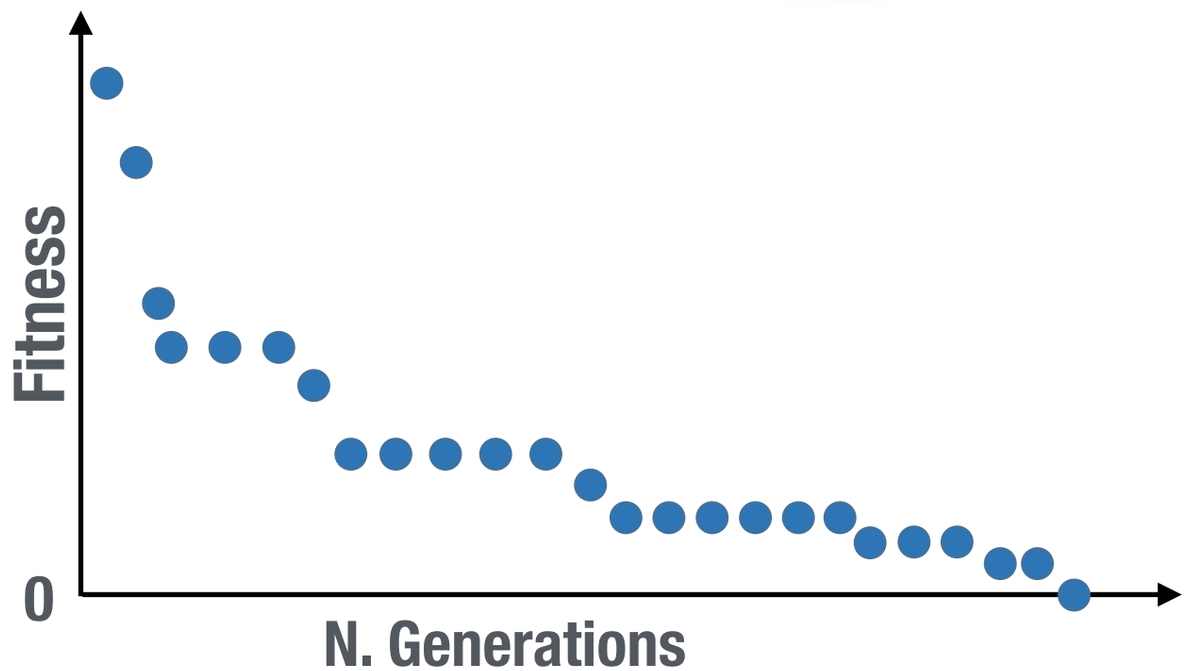
Running Example

```
class Triangle {

 void computeTriangleType() {
   if (isTriangle()){
     if (side1 == side2) {
          if (side2 == side3)
              type = "EQUILATERAL";
          else
              type = "ISOSCELES";
      } else {
          if (side1 == side3) {
              type = "ISOSCELES";
           } else {
             if (side2 == side3)
                 type = “ISOSCELES”;
             else
                checkRightAngle();
                }
          }
      }// if isTriangle()
}}
```

Target





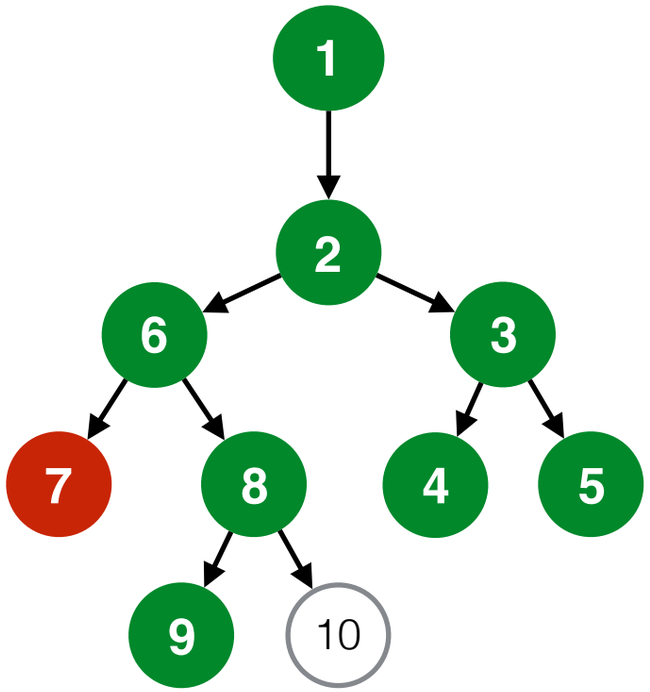
TC1 = (4,3,3)
TC2 = (2,2,4)

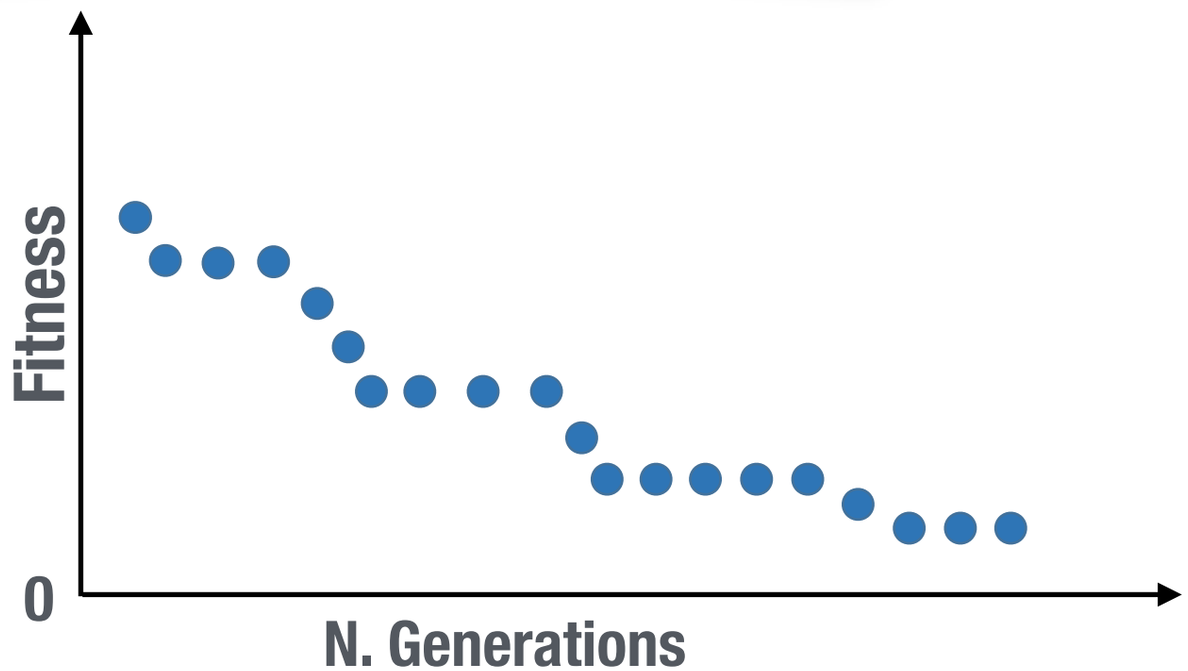
Running example

```
class Triangle {

 void computeTriangleType() {
   if (isTriangle()){
     if (side1 == side2) {
          if (side2 == side3)
              type = "EQUILATERAL";
          else
              type = "ISOSCELES";
      } else {
          if (side1 == side3) {
              type = "ISOSCELES";
           } else {
             if (side2 == side3)
                 type = “ISOSCELES”;
             else
                checkRightAngle();
                }
          }
      }// if isTriangle()
}}
```

Target



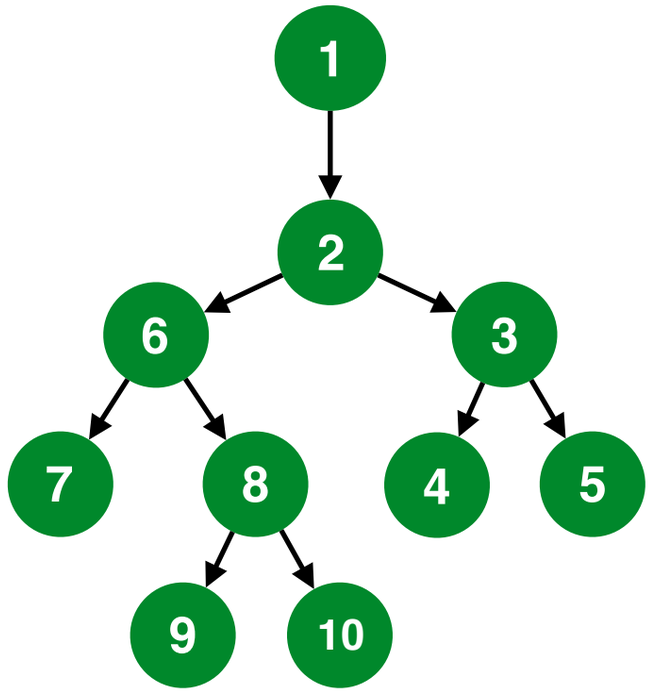


TC1 = (4,3,3)
TC2 = (2,2,4)
TC2 = (5,5,5)

Running example

```
class Triangle {

 void computeTriangleType() {
   if (isTriangle()){
     if (side1 == side2) {
          if (side2 == side3)
              type = "EQUILATERAL";
          else
              type = "ISOSCELES";
      } else {
          if (side1 == side3) {
              type = "ISOSCELES";
           } else {
             if (side2 == side3)
                type = “ISOSCELES”;
             else
                checkRightAngle();
                }
         }
     }// if isTriangle()
}}
```

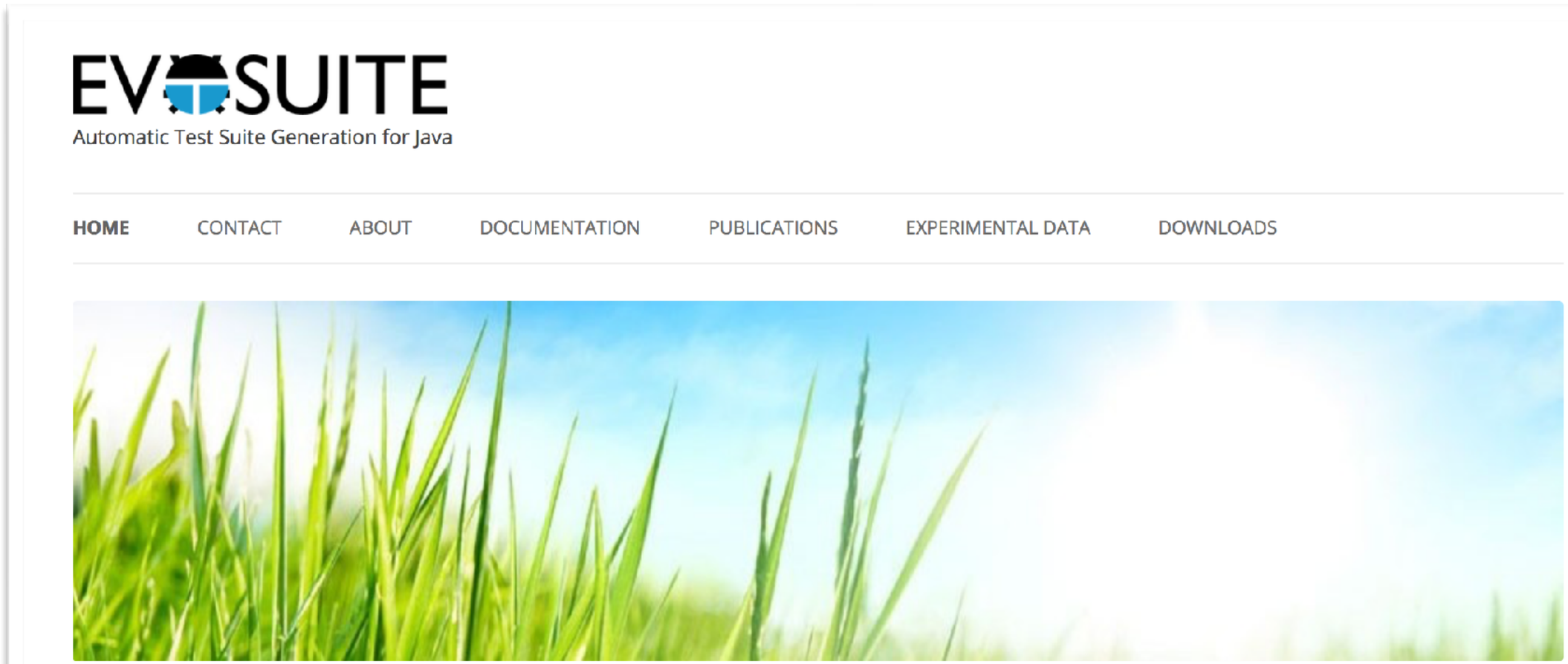


TC1 = (4,3,3)
TC2 = (2,2,4)
TC3 = (5,5,5)
TC4 = (4,3,4)
TC5 = (1,2,3)

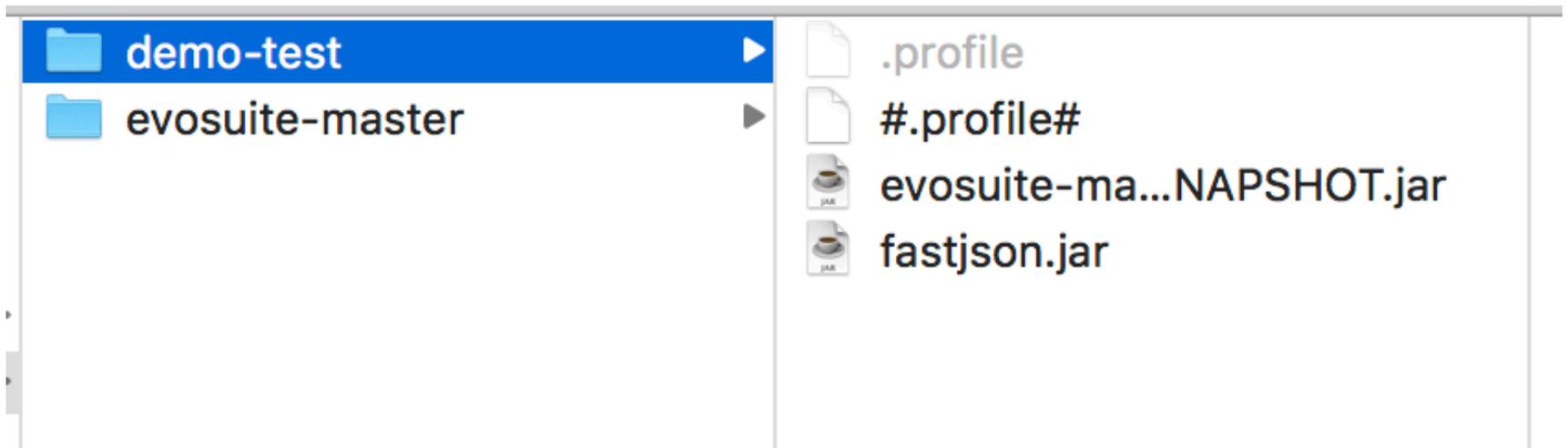
The final test suite consists of all chromosome that have been found to cover (even accidentally) one or more yet to cover statements.

EvoSuite (Unit Testing)

http://www.evosuite.org



EvoSuite Approach




```
# The execute pDynaMOSA the following command
java -jar evosuite-master-1.0.7-SNAPSHOT.jar -
generateMOSuite -projectCP fastjson.jar -class
 com.alibaba.fastjson.parser.JSONLexerBase -
Dminimize=FALSE -Dassertions=FALSE -Dsearch_budget=40
-Dconfiguration_id=Evosuite -Dtest_dir=slow-test
```

Random Testing?

Random Testing

How to generate random tests:

1) **Pick one of the available constructors (with random input)**

2) **Pick one or more public methods (with random input)**

3) **Generate the assertions by checking the final state of the object using get methods**

Random Testing

Class Under Test API

public ClassUT(…)
…
public ClassUT(…)

public method1(…)
…
public methodK(…)

Random Testing

Class Under Test API

```
public ClassUT(…)
…
public ClassUT(…)
```

Constructors

```
public method1(…)
…
public methodK(…)
```

Public Methods

Random Test

```
@Test
public void test(){
  // constructor
  // method calls
  // assertions
}
```

Random Testing

Class Under Test API

```
public ClassUT(…)
…
public ClassUT(…)

public method1(…)
…
public methodK(…)
```

Pick one of the available constructors

With random parameters input

Random Test

```
@Test
public void test(){
  ClassUT c = new ClassUT(.);
  // method calls
  // assertions
}
```

Random Testing

Class Under Test API

```
public ClassUT(…)
…
public ClassUT(…)

public method1(…)
…
public methodK(…)
```

Pick one or more public methods

With random parameters input

Random Test

```
@Test
public void test(){
  ClassUT c = new ClassUT(.);
  c.method1(…);
  c.method3(…);
  // assertions
}
```

Random Testing

Class Under Test API

```
public ClassUT(…)
…
public ClassUT(…)

public method1(…)
…
public methodK(…)
```

Use get methods to check the state of the objects after execution

Random Test

```
@Test
public void test(){
  ClassUT c = new ClassUT(.);
  c.method1(…);
  c.method3(…);
  assertTrue(me.method2());
}
```

Running Example

```
class PeakFunction {
   private double x, y;

   public PeakFunction (double pX, double pY){
      x = pX; y = pY;
   }

   public double computeZeta() {
     double z =  3*Math.pow(1-x,2);
     z *= Math.exp(- Math.pow(x,2) - Math.pow(y+1,2));
     z -= 10*(x/5 - Math.pow(x,3);
     z -= Math.pow(y,5)*Math.exp(-Math.pow(x,2)-Math.pow(x,2));
     z -= 1/3* Math.exp(-Math.pow(x+1,2) - Math.pow(y,2));
     return z;
   }

   public boolean isZero() {
   if (computeZeta() <= 0.05)
      return true;
   }
}
```

We want to cover this method

Running Example

PeakFunction API

```
// Public constructors
PeakFunction(double, double)

// Public methods
public double computeZeta()
public boolean isZero()
```

Random Test

```
@Test
public void test(){
    // constructor
    // method calls
    // assertions
}
```

Running Example

PeakFunction API

```
// Public constructors
PeakFunction(double, double)

// Public methods
public double computeZeta()
public boolean isZero()
```

Random Test

```
@Test
public void test(){
    PeakFunction pf = new PeakFunction(?,?);
    boolean b = isZero();
    assertTrue(b); or assertFalse(b);
}
```

To generate random double in [0;1], we can use the method **Math.random()**

To generate random double in [a;b], we can use the following code:
Math.random() * (a+b) - a

Running Example

PeakFunction API

```
// Public constructors
PeakFunction(double, double)

// Public methods
public double computeZeta()
public boolean isZero()
```

x = 0.8147 , y = 0.9058, z = 2.2346
x = 0.1270 , y = 0.9134, z = 2.6334
x = 0.6324 , y = 0.0975, z = 0.8979
x = 0.2785 , y = 0.5469, z = 0.1855

Random Test1

```
@Test
public void test(){
    PeakFunction pf = new PeakFunction(1.4, 2.3);
    boolean b = isZero(); // zeta = 0.4746
    assertFalse(b);
}
```

Random Test2

```
@Test
public void test(){
    PeakFunction pf = new PeakFunction(-0.2, 1.5);
    boolean b = isZero(); // zeta =7.7118
    assertFalse(b);
}
```

Running Example

```
class PeakFunction {
    ...
    public boolean isZero() {
    if (computeZeta() <= 0.05)
        return true;
    }
}
```

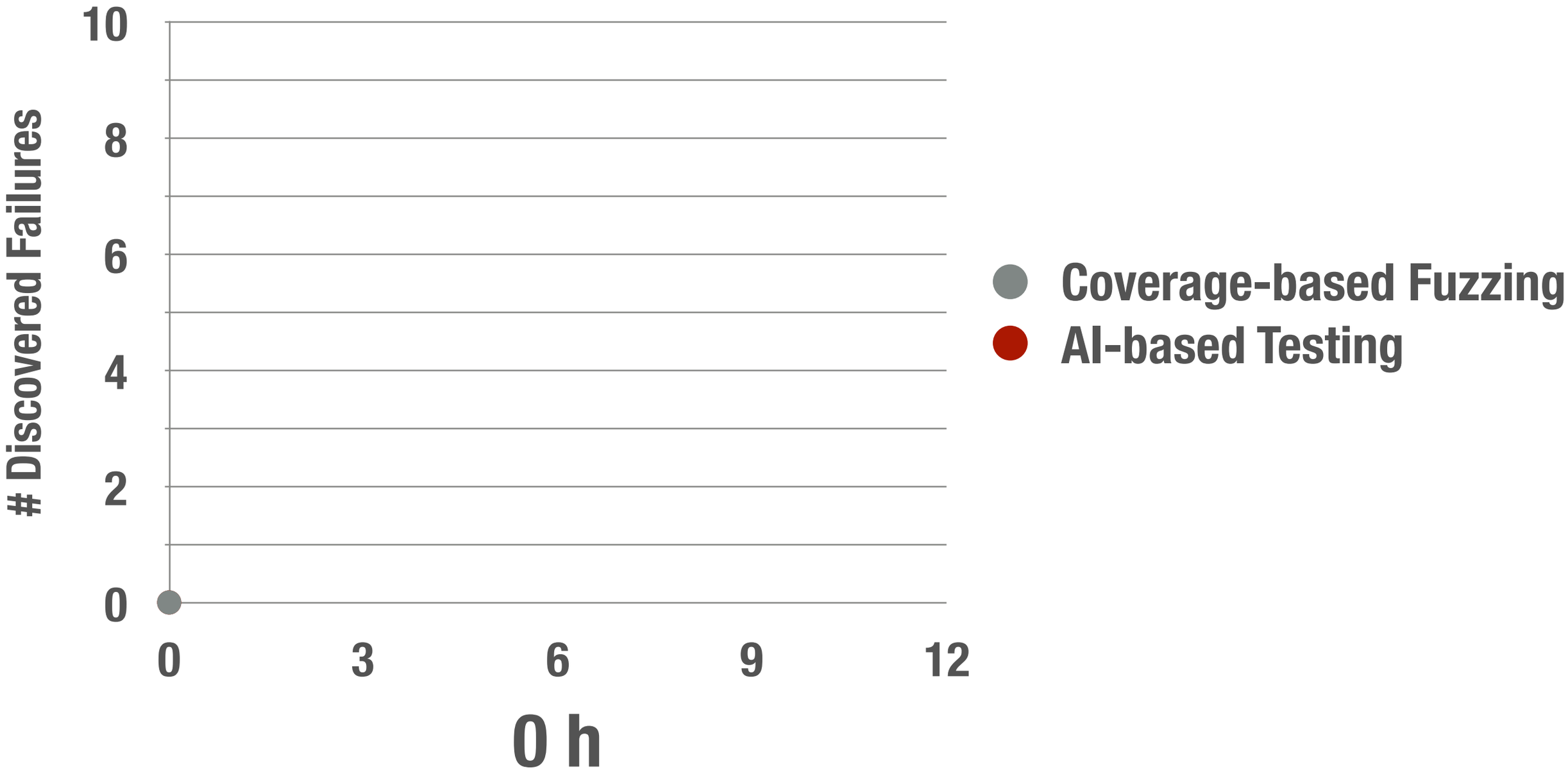
We want to cover this method

The false branch is very easy to cover with random testing

The true branch is very difficult as it require to satisfy the equation computeZeta()=0

What is the probability that a random pair of number X and Y satisfy that condition?

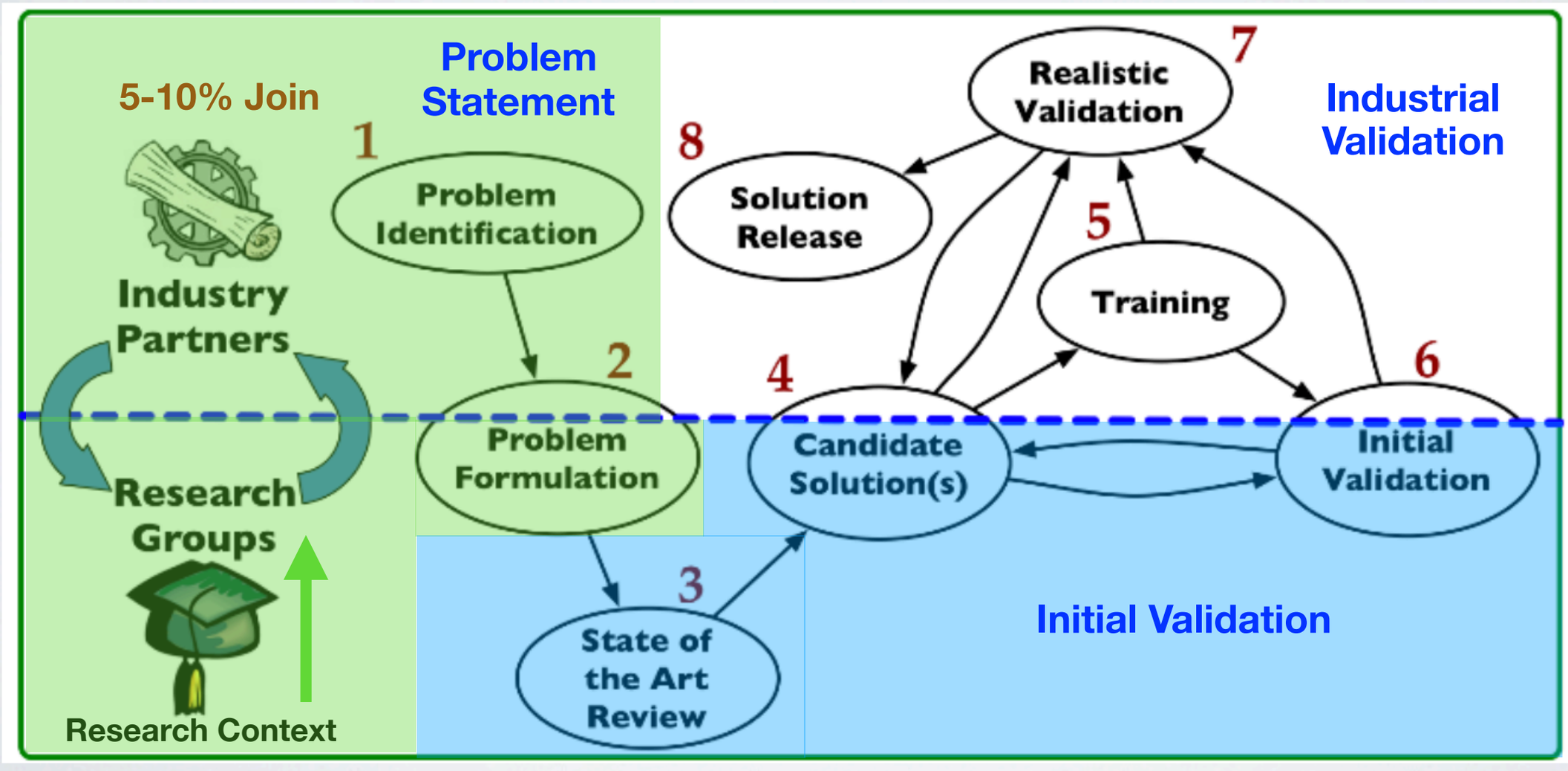
Evosuite Results



How to come up with a Vision?

"..there are guidelines..."

- Strong emphasis on applied research, driven by needs
- Tight, long-term industrial collaborations



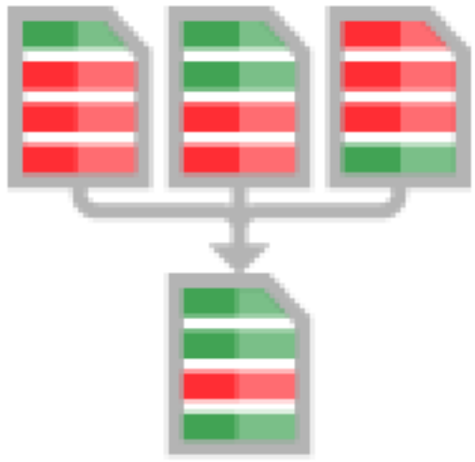
"Why and How to Get a PhD?" Lionel Briand

https://bit.ly/2LGg7nl

Research goal

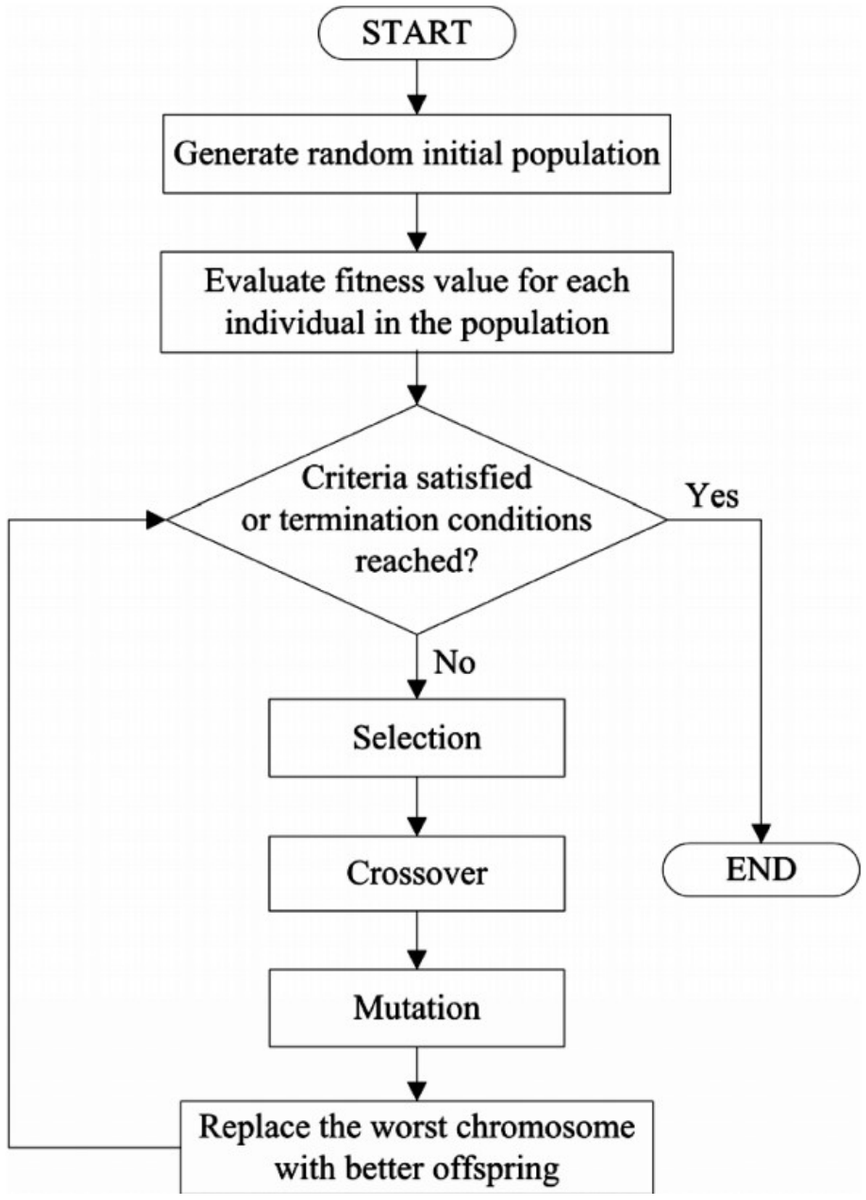
- **Automatically generating tests** with appropriate performance (CPU, memory, etc.) when deployed in different environments

It uses indicators of Test Coverage…



```
1| public class TestOption {
2|
3| @Test
4| public void test0() throws Throwable {
5|    Option option0 = new Option("", "1W|^");
6|    assertEquals("1W|^", option0.getDescription());
7|    assertEquals("", option0.getKey());
8|  }
9| }
```

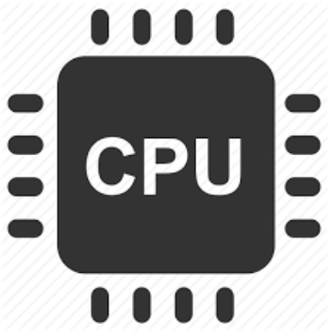
Test Case Automatically Generated by Evosuite (for the class apache.commons.Option.Java)

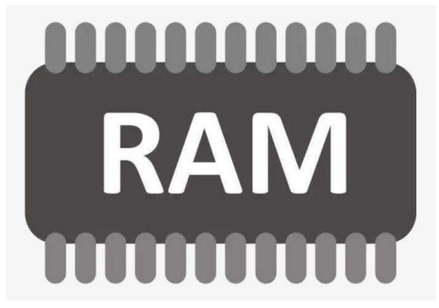


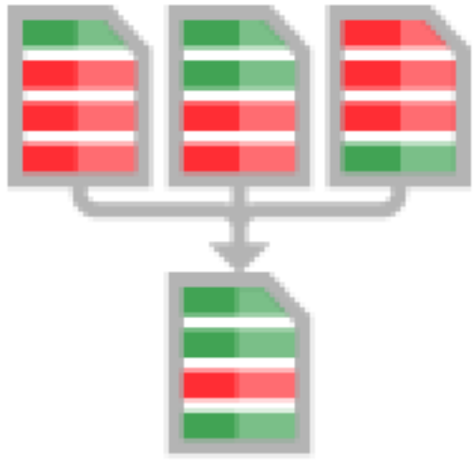
Technical solutions

- **Automatically generating tests** with appropriate performance (CPU, memory, etc.) when deployed in different environments

Further Indicators…

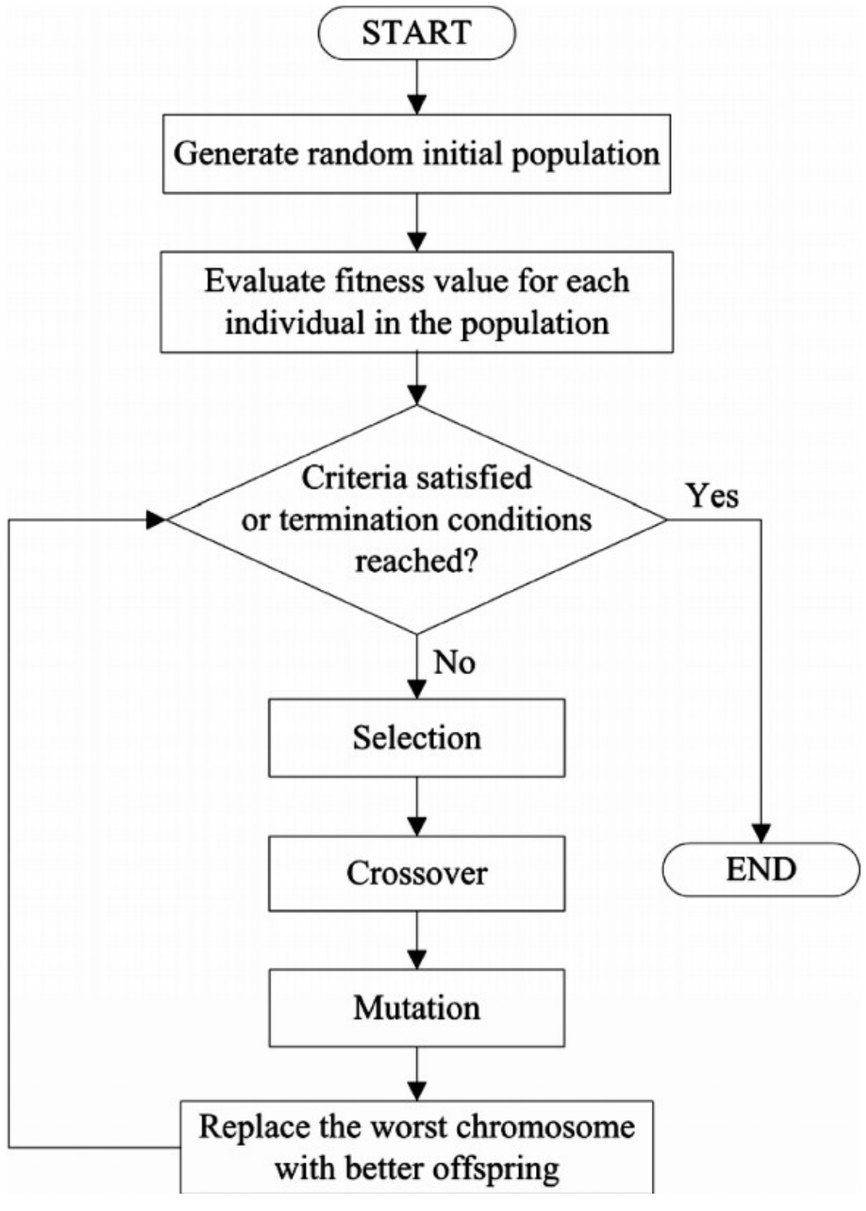




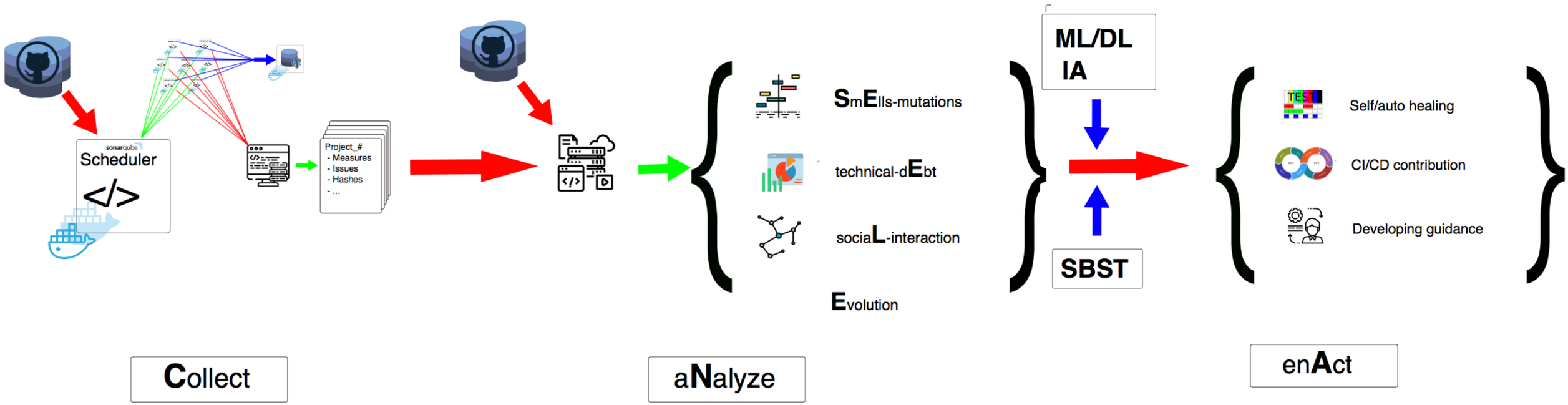


```
1| public class TestOption {
2|
3| @Test
4| public void test0() throws Throwable {
5|    Option option0 = new Option("", "1W|^");
6|    assertEquals("1W|^", option0.getDescription());
7|    assertEquals("", option0.getKey());
8|  }
9| }
```

Test Case Automatically Generated by Evosuite (for the class apache.commons.Option.Java)



CNA Research Approach



Collect

a**N**alyze

en**A**ct

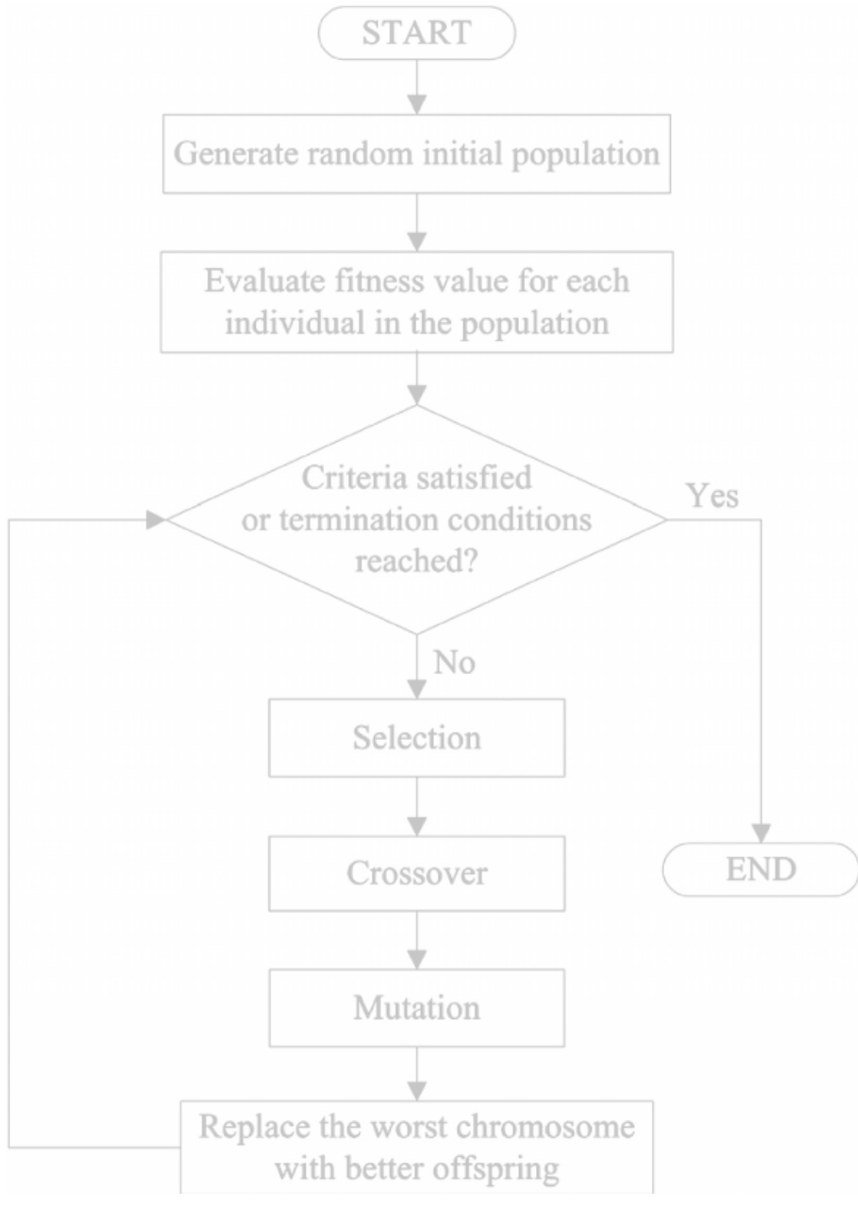
pDynaMOSA: DEMO

- **Automatically generating tests** with appropriate performance (CPU, memory, etc.) when deployed in different environments

Further Indicators…


```
1| public class TestOption {
2|
3| @Test
4| public void test0() throws Throwable {
5|    Option option0 = new Option("", "1W|^");
6|    assertEquals("1W|^", option0.getDescription());
7|    assertEquals("", option0.getKey());
8|  }
9| }
```

Test Case Automatically Generated by Evosuite (for the class apache.commons.Option.Java)



Technical Challenges

1) A **set** of **performance proxies** that provide an approximation of the test execution costs (i.e., runtime and memory usage).

G. Grano

S. Panichella

2) Leverages these proxies to **extend Evosute**, a state-of-the-art evolutionary algorithm in unit testing.

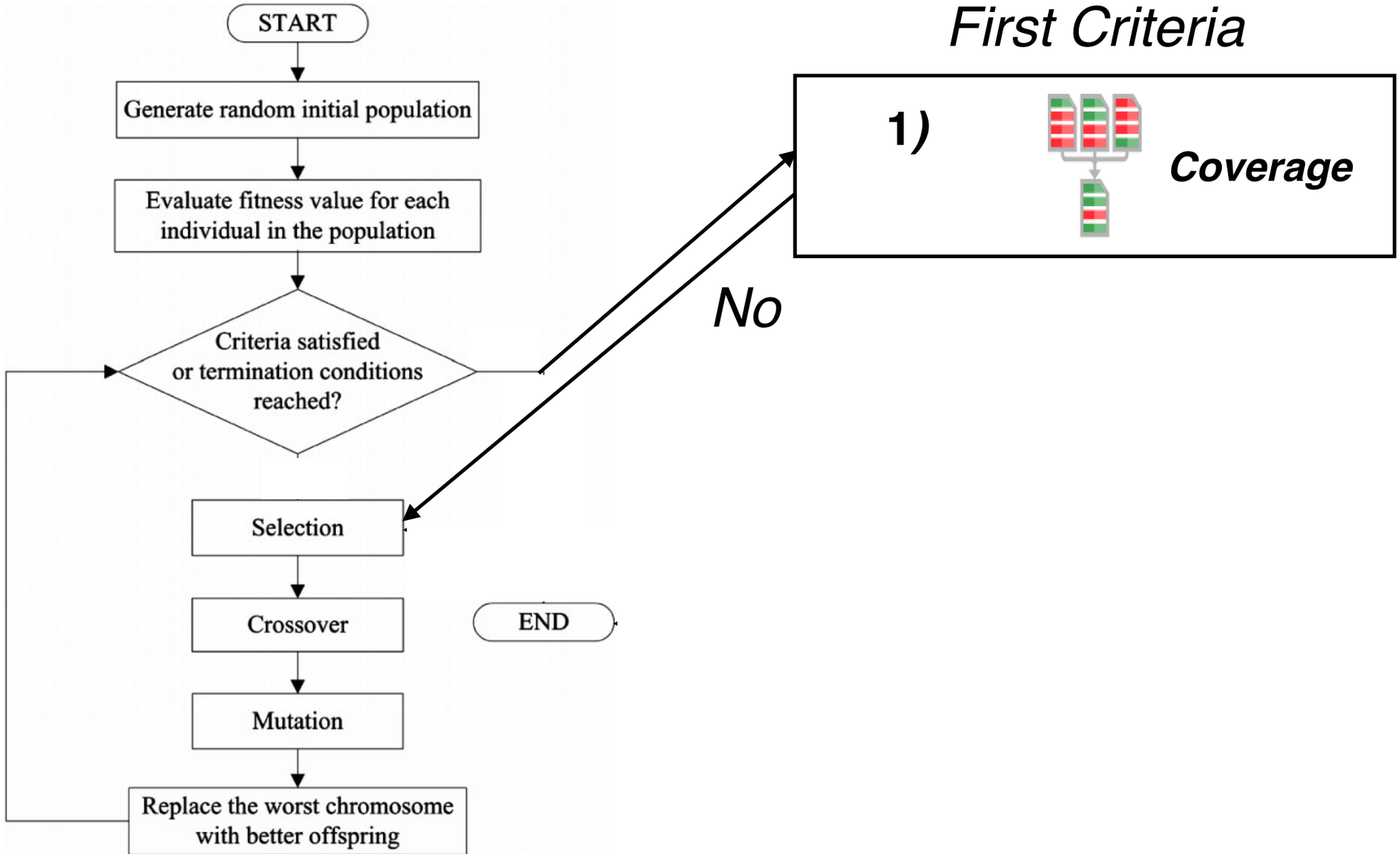
A. Panichella

G. Grano

S. Panichella

pDynaMOSA Approach

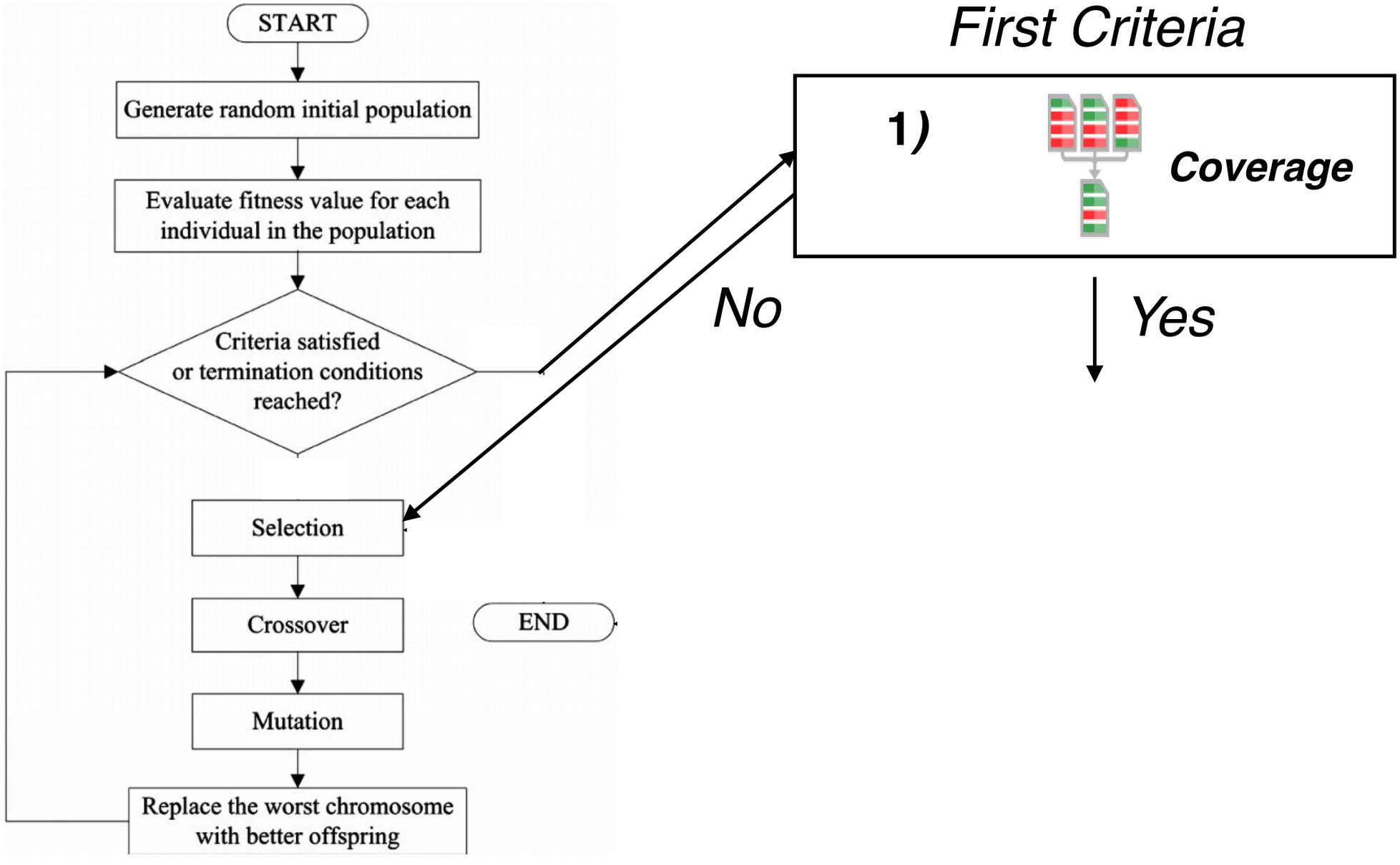
pDynaMOSA (Adaptive Performance-Aware DynaMOSA),



***pDynaMOSA* Pipeline**

pDynaMOSA Approach

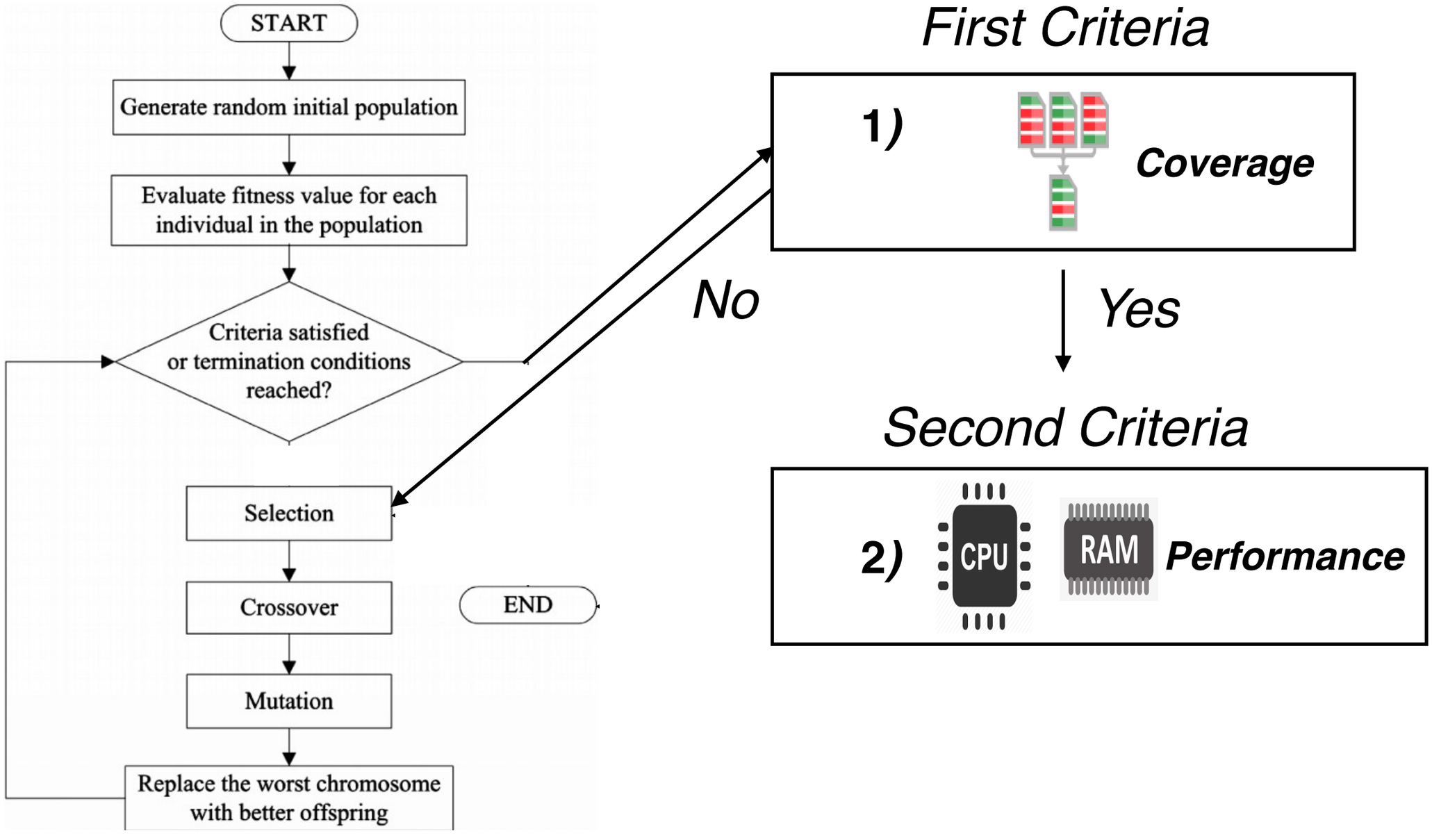
pDynaMOSA (Adaptive Performance-Aware DynaMOSA),



***pDynaMOSA* Pipeline**

pDynaMOSA Approach

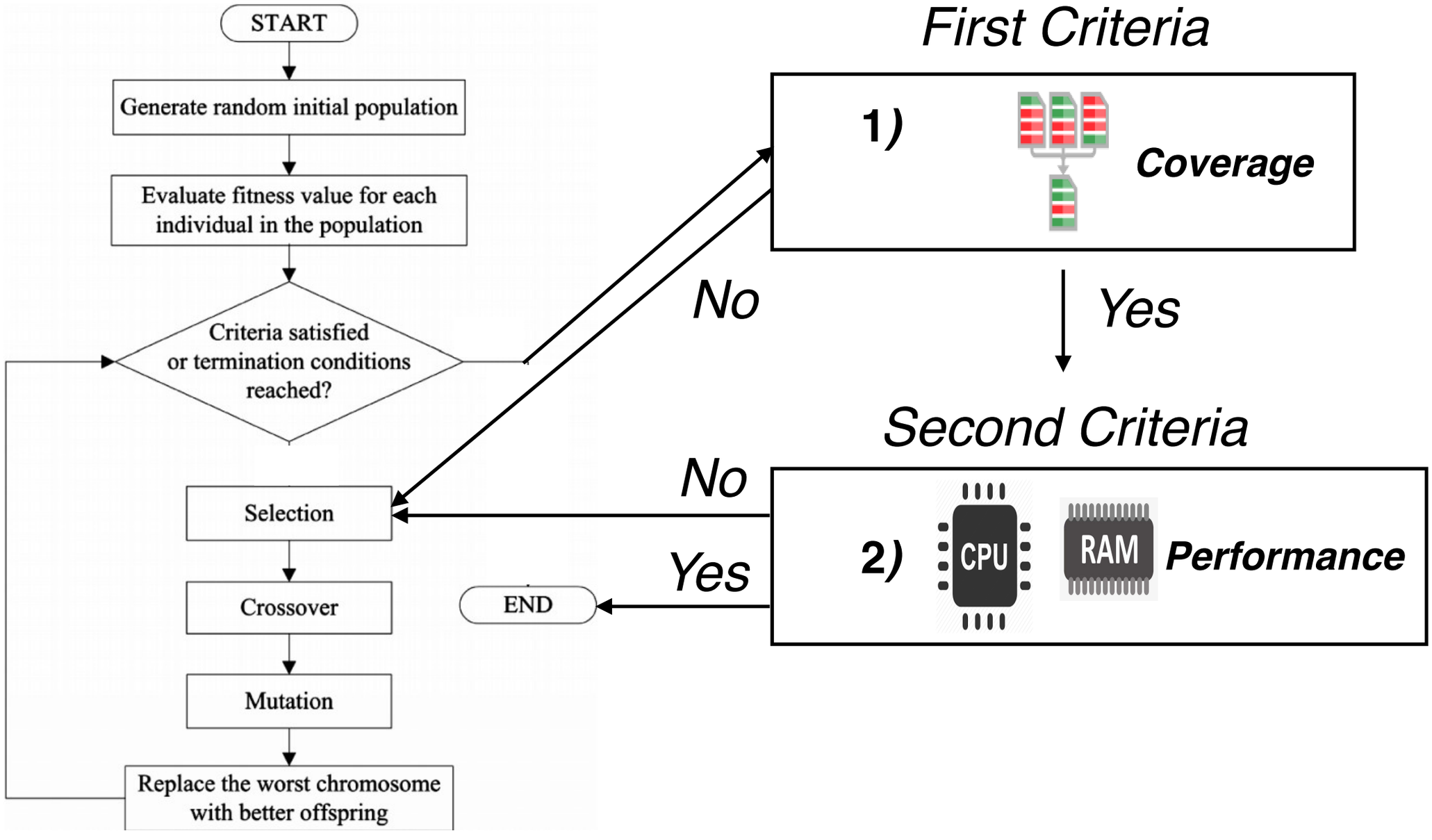
pDynaMOSA (Adaptive Performance-Aware DynaMOSA),



***pDynaMOSA* Pipeline**

pDynaMOSA Approach

pDynaMOSA (Adaptive Performance-Aware DynaMOSA),



***pDynaMOSA* Pipeline**

pDynaMOSA Approach

pDynaMOSA (Adaptive Performance-Aware DynaMOSA),

```
# The execute pDynaMOSA the following command
java -jar evosuite-master-1.0.7-SNAPSHOT.jar -
generateMOSuite -projectCP fastjson.jar -class
com.alibaba.fastjson.parser.JSONLexerBase -
Dperformance_indicators=STATEMENTS_COVERED:TEST_LENGTH:COV
ERED_METHOD_CALL:OBJECTS_INSTANTIATIONS:LOOP_COUNTER:METHO
D_CALL -Dsecondary_objectives=PERFORMANCE -
Dalgorithm=DynaMOSA  -
Dperformance_strategy=CROWDING_DISTANCE -Dminimize=FALSE -
Dassertions=FALSE -Dsearch_budget=40 -
Dconfiguration_id=DMOSA -Dtest_dir=fast-test
```

pDynaMOSA Approach

pDynaMOSA (Adaptive Performance-Aware DynaMOSA),

```
# Execute pDynaMOSA with the following command
java -jar evosuite-master-1.0.7-SNAPSHOT.jar -
generateMOSuite -projectCP fastjson.jar -class
com.alibaba.fastjson.parser.JSONLexerBase -
Dperformance_indicators=STATEMENTS_COVERED:TEST_LENGTH:COV
ERED_METHOD_CALL:OBJECTS_INSTANTIATIONS:LOOP_COUNTER:METHO
D_CALL -Dsecondary_objectives=PERFORMANCE -
Dalgorithm=DynaMOSA  -
Dperformance_strategy=CROWDING_DISTANCE -Dminimize=FALSE -
Dassertions=FALSE -Dsearch_budget=40 -
Dconfiguration_id=DMOSA -Dtest_dir=fast-test
```

Evaluation

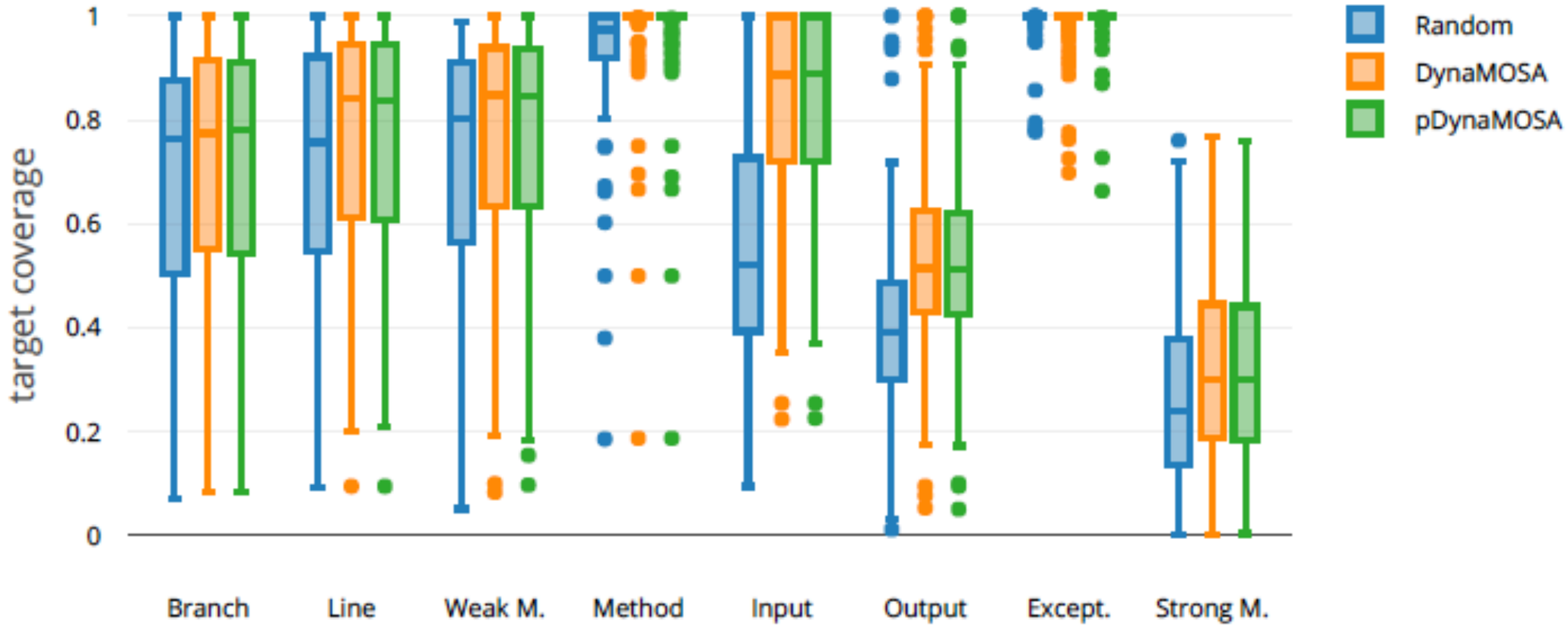
Dataset from SBST Community

Project	#	Branches			Mutants		
		Min	Max	Mean	Min	Max	Mean
a4j	2	30	124	77	15	911	463
bcel	4	52	890	475	408	1,523	1,043
byuic	1	722	722	722	2,173	2,173	2,173
fastjson	10	20	2,880	564	36	13,152	2,078
firebird	3	90	194	131	347	441	392
fixsuite	1	32	32	32	110	110	110
freehep	6	48	160	92	112	807	297
freemind	1	170	170	170	2,427	2,427	2,427
gson	4	60	660	285	126	2,870	1,212
image	7	34	274	140	214	1,676	589
javathena	1	230	230	230	752	752	752
javaviewcontrol	2	212	2,360	1,286	2,058	4,972	3,515
jdbacl	2	170	174	172	595	700	648
jiprof	1	816	816	816	6,420	6,420	6,420
jmca	2	198	1,696	947	2,436	9,669	6,052
jsecurity	1	52	52	52	165	165	165
jxpath	3	98	102	100	204	449	312
la4j	7	20	280	135	196	3,217	1,122
math	4	14	238	92	135	1,274	443
okhttp	5	64	542	194	200	2,571	846
okio	9	24	562	126	34	4,271	1,009
re2j	8	68	646	178	148	2,096	1,129
saxpath	1	458	458	458	659	659	659
shop	4	38	182	102	175	465	302
webmagic	4	10	142	84	29	337	201
weka	10	212	778	359	255	13,263	2,220
wheelwebtool	7	24	804	331	75	3,898	1,637
Total	110						

G. Fraser and A. Arcuri, “A large-scale evaluation of automated unit test generation using evosuite,” ACM Transactions on Software Engineering and Methodology (TOSEM),

Research Questions

RQ1. (Effectiveness) What is the target coverage achieved by pDynaMOSA compared to DynaMOSA?

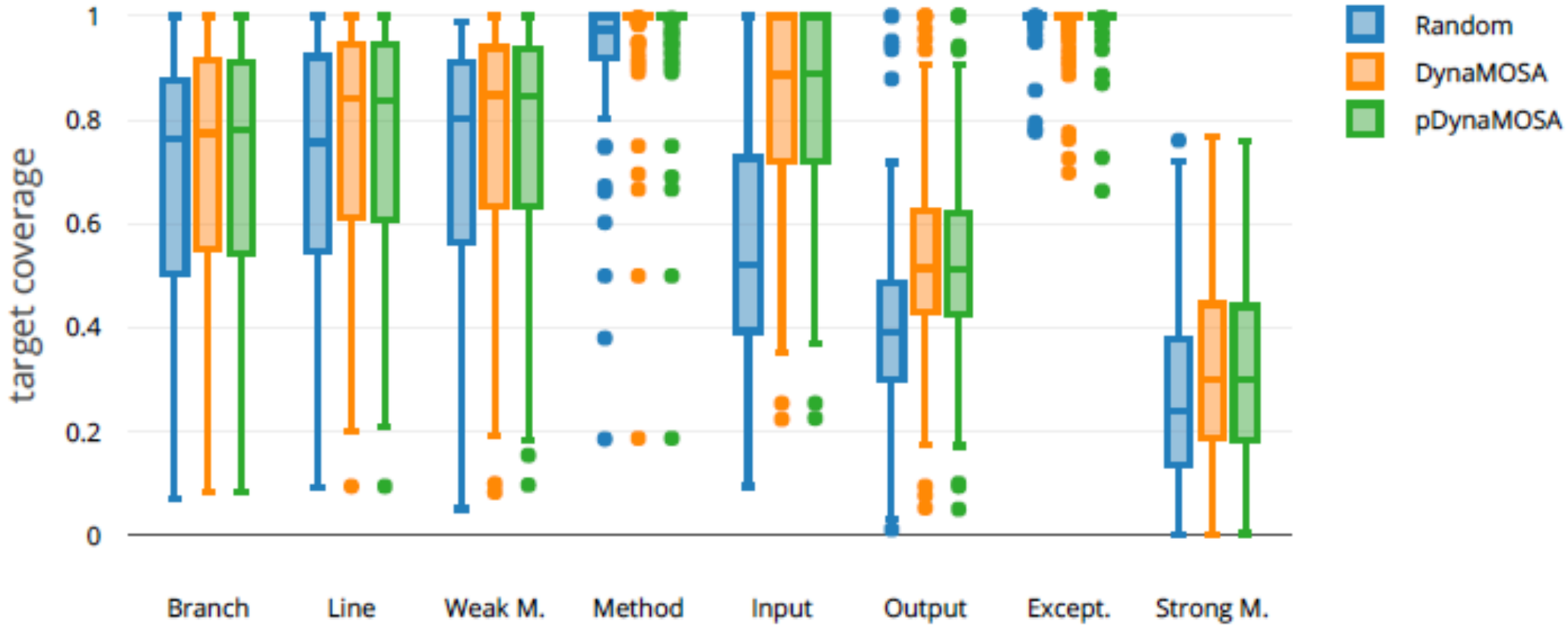


Comparison of coverage achieved by Random Search, DynaMOSA, and pDynaMOSA over 50 independent runs for the 110 studied subjects

Small gain in terms of coverage…

Research Questions

RQ2. (Fault Detection) What is the mutation score achieved by pDynaMOSA compared to DynaMOSA?



Comparison of coverage achieved by Random Search, DynaMOSA, and pDynaMOSA over 50 independent runs for the 110 studied subjects

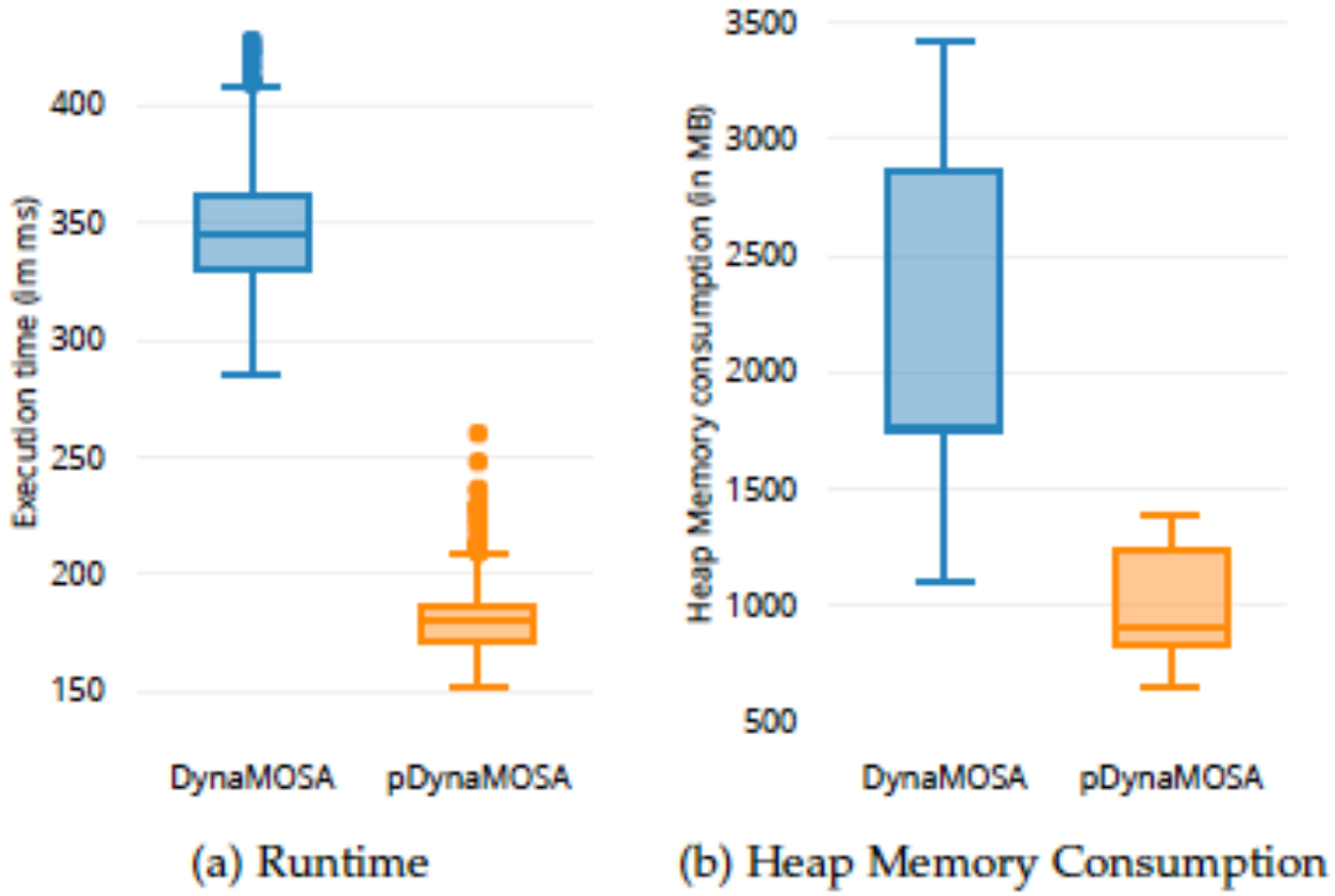
No. cases pDynaMOSA is better than Random	88 (96.8%)
No. cases pDynaMOSA is worse than Random	1 (1.1%)
No. cases pDynaMOSA is better than DynaMOSA	5 (5.5%)
No. cases pDynaMOSA is worse than DynaMOSA	16 (17.6%)

Small gain in terms of coverage…

We lose in terms of fault detection…

Research Questions

RQ3. (Performance) Does the adoption of performance proxies lead to shorter runtime and lower heap memory consumption?



(a) Runtime (b) Heap Memory Consumption

Christoph Laaber (UZH)

"Statistically rigorous java performance evaluation,"

OOPSLA '07.

Huge benefits in terms of runtime and memory consumption…

Small gain in terms of coverage…

We lose in terms of fault detection…

Outline

PART I

1) Research Context and Motivation

PART II

2) Literature review on:
   - Cloud Testing Challenges
   - Automated Testing

3) Future Directions

Future work


LOCAL DEVELOPMENT
1
Forget about **local development:** Emulation is not **comprehensive** and **brittle**
2
Write **more** tests and run them to shorten the Feedback Loop
3
Tests closer to **production** tell more
4
Tests in cloud **enforces** better **observability**
homegate.ch

Future work


LOCAL DEVELOPMENT
1
Forget about **local development:** Emulation is not **comprehensive** and **brittle**
2
Write **more** tests and run them to shorten the Feedback Loop
3
Tests closer to **production** tell more
4
Tests in cloud **enforces** better **observability**
homegate.ch

Future work


LOCAL DEVELOPMENT
1
Forget about **local development:** Emulation is not **comprehensive** and **brittle**
2
Write **more** tests and run them to shorten the Feedback Loop
3
Tests closer to **production** tell more
4
Tests in cloud **enforces** better **observability**
homegate.ch

Future work


LOCAL DEVELOPMENT
1
Forget about **local development:** Emulation is not **comprehensive** and **brittle**
2
Write **more** tests and run them to shorten the Feedback Loop
3
Tests closer to **production** tell more
4
Tests in cloud **enforces** better **observability**
homegate.ch

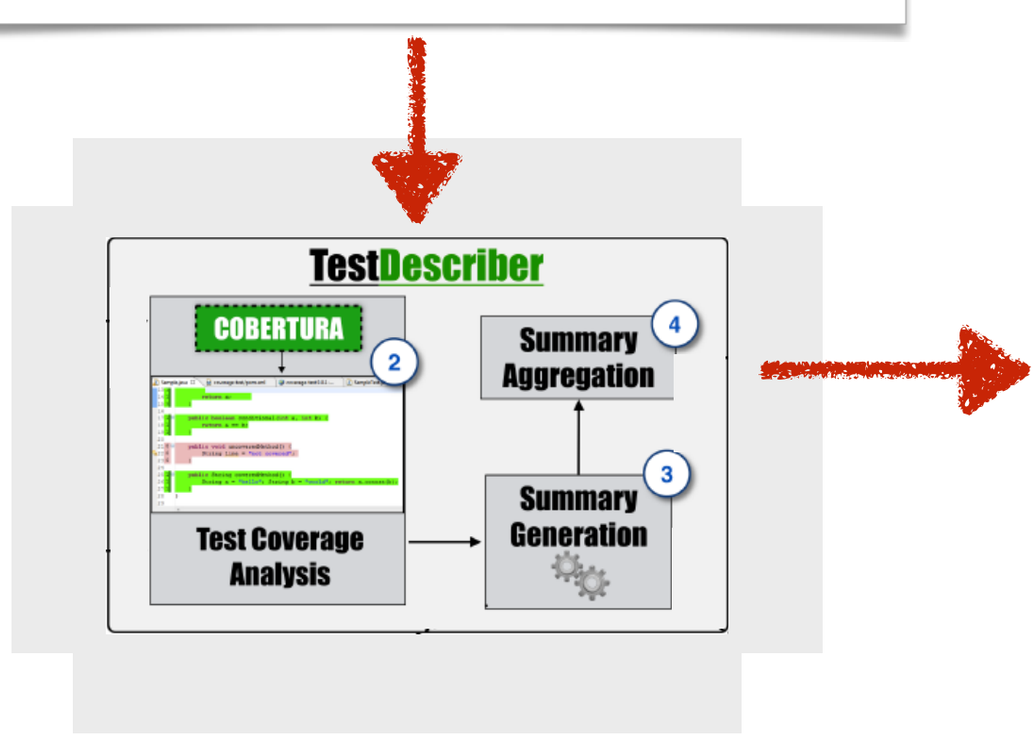
“Unit Testing Comments Generation?”

Combining Test Case Generation with NLP

Generated Unit Test

```
public class TestOption {

@Test
public void test0() throws Throwable {
   Option option0 = new Option("", "1W|^");
   assertEquals("1W|^", option0.getDescription());
   assertEquals("", option0.getKey());
 }
}
```

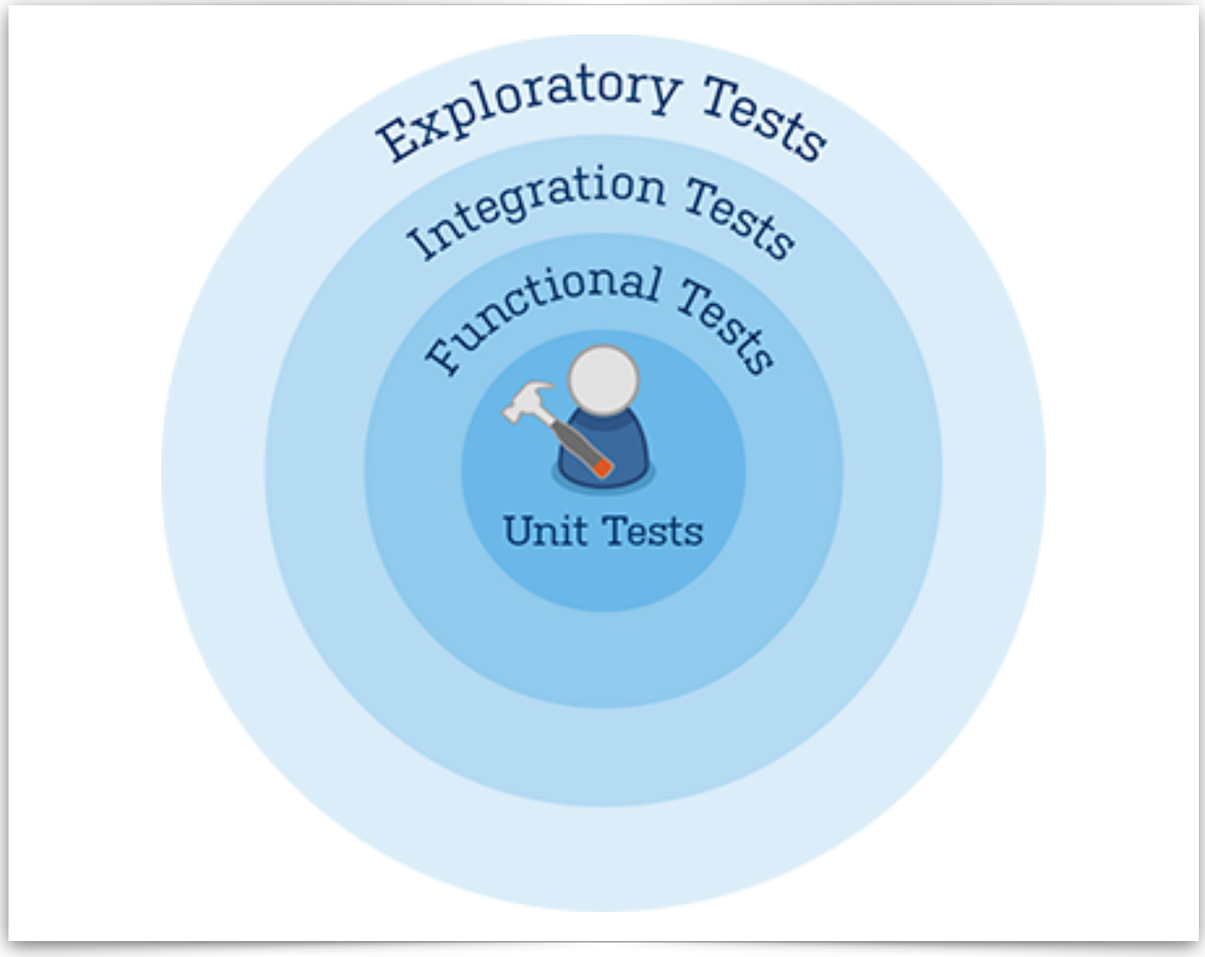


… with Descriptions

```
public class TestOption {
/** OVERVIEW: The test case "test0" covers around 3.0% 3.b
 * (low percentage) of statements in "Option" **/
@Test
public void test0() throws Throwable {
// The test case instantiates an "Option" with option 3.c
// equal to "", and description equal to "1W|^".
Option option0 = new Option("", "1W|^");
// Then, it tests:
// 1)  whether the description of option0 is equal to 3.c
// "1W|^";
 assertEquals("1W|^", option0.getDescription());
// 2) whether the key of option0 is equal to ""; 3.c
// The execution of the method call used in the assertion 3.d
// implicitly covers the following 1 conditions:
// - the condition "option equal to null" is FALSE;
 assertEquals("", option0.getKey());
 }
}
```

Panichella - “The impact of test case Summaries on Bug Fixing Performance: And Empirical Investigation”. ICSE 2016

“Beyond Unit Testing?”



Thanks for the Attention!

Sebastiano Panichella

spanichella@gmail.com or

panc@zhaw.ch

https://spanichella.github.io/

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

- Any Questions?