IEEE International Conference on Software Testing
ICST 2021

Some challenges and pitfalls in engineering contemporary software systems under the plague of defects

Guilherme Horta Travassos

PESC/COPPE

Federal University of Rio de Janeiro

CNPq researcher, ISERN member

CNPq

ght@cos.ufrj.br

orcid: 0000-0002-4258-0424

COPPE UFRJ

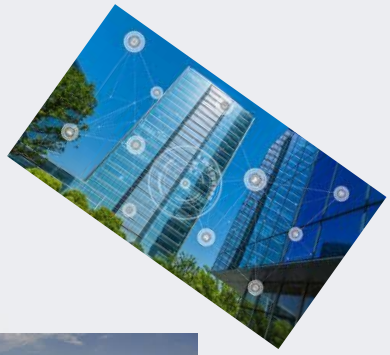
Overview

Modern software systems (context-aware, Internet of Things, Industry 4.0, cyber-physical systems) have dominated the technological landscape and have become increasingly influential in society's activities. Some concepts, methods, tools, and standards have been proposed to support their development. However, despite all the efforts, the plague of defects persists. Some challenges, such as context-awareness, jeopardize their identification. Therefore, the risks associated with the use of modern software systems that have been made available to society daily are still high, with some systems causing human losses.

This talk intends to present some of these challenges and draw community attention to contemporary software systems engineering.

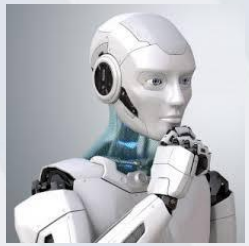
What do they use in common?

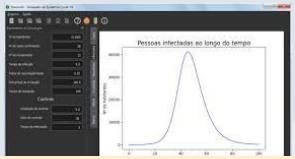


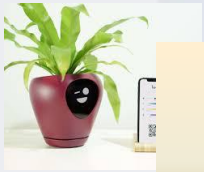



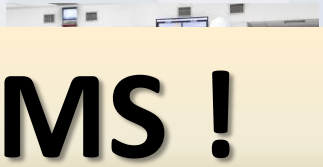

















SOFTWARE SYSTEMS !

Software Systems Evolution

1st stage
Custom Software
Standalone
Batch
Multi-user
Real-time
Database
Product Software
Distributed Systems
Embedded "intelligence"
Low cost hardware
Consumer Impact

2nd stage
Powerful desk-top systems
Object-oriented technologies
Expert systems
Artificial neural networks
Parallel computing
Network computers
Multi-skilled, geographically distributed development
Componentry (reuse and recycling)

3rd stage
mobile apps
e-science with intensive use of e-infrastructure
Ubiquitous Systems
Systems of systems

4th stage: Contemporary Stage
Digital Transformation!!!

70's
One computer, many users

05's
One computer, one user

10's
Many computers for one user

20's
Many computers for many users


SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

Development and evolution models, including biological analogies
Interdependence among design, business, and evaluation
Agile software manufacture
Empowering the domain expert (vs. maintaining integrity)
Non-scripting development languages

Some Software Systems Characteristics

Some Software Systems Characteristics

Construction costs are concerned with its engineering.

X

Hardware

X

Software

The software doesn't "wear out," but it deteriorates.

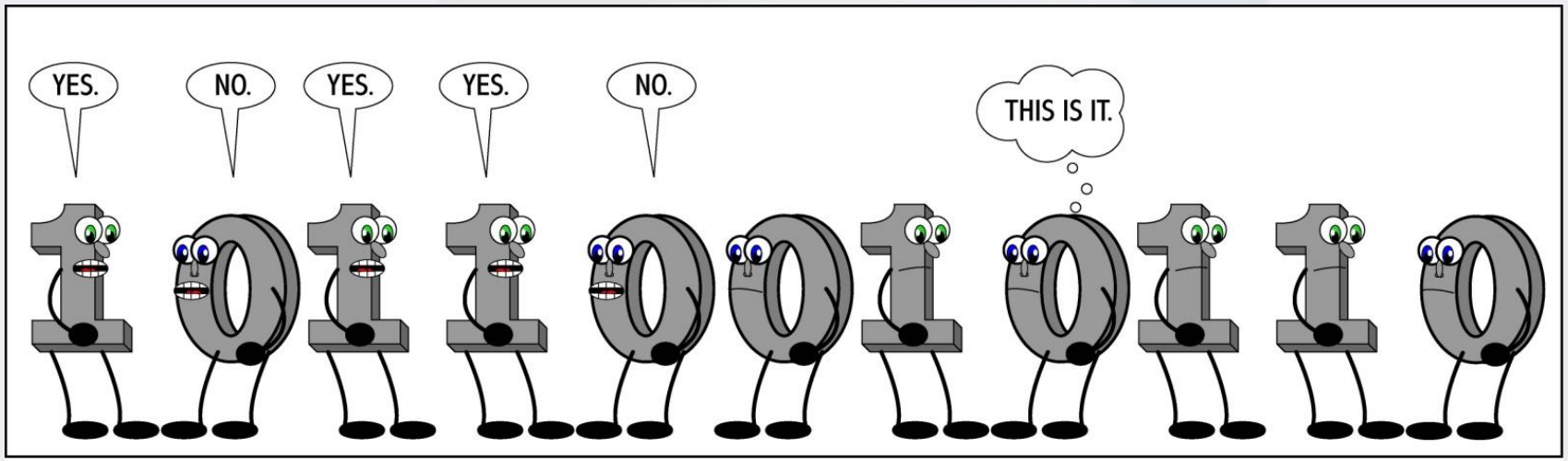
Usually not assembled from existing (high quality) components yet.

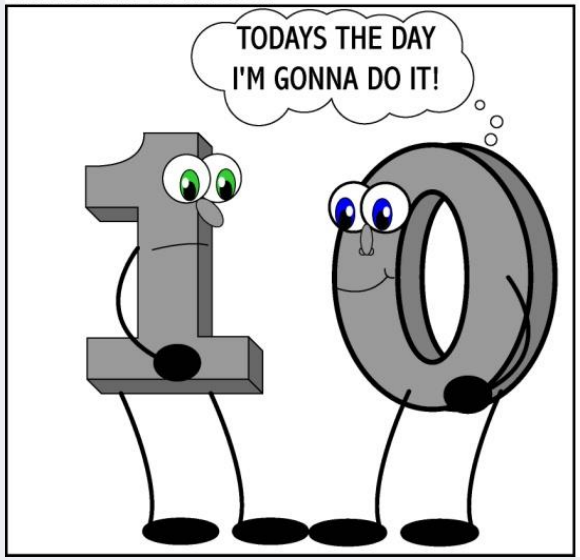
Customized

X

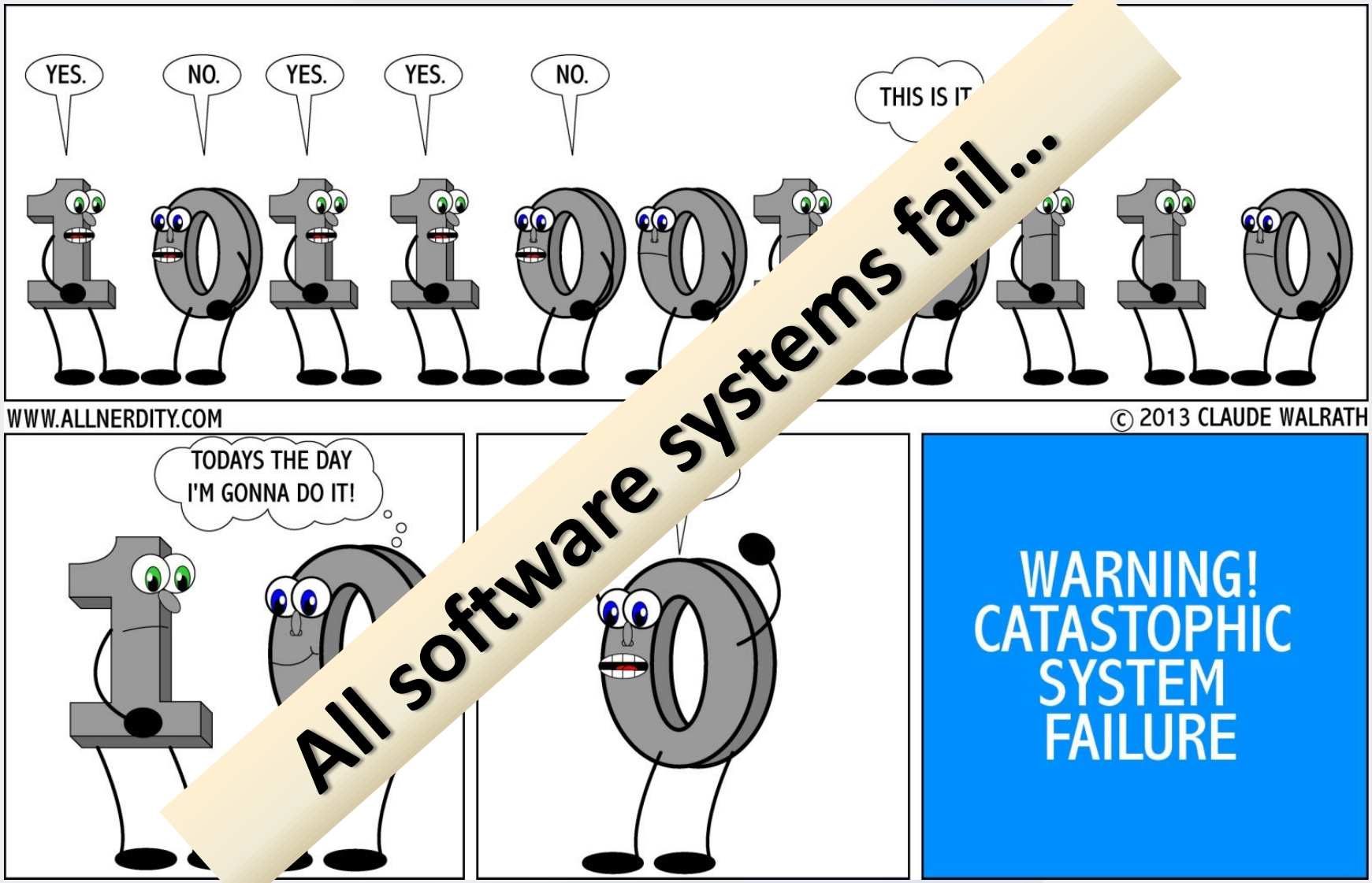
Componentized

Some Software Systems Characteristics



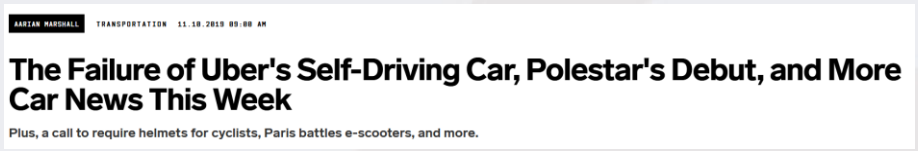


Some Software Systems Characteristics



All software systems fail...

Some Software Systems Characteristics


AARIAN MARSHALL TRANSPORTATION 11.18.2019 09:00 AM

The Failure of Uber's Self-Driving Car, Polestar's Debut, and More Car News This Week

Plus, a call to require helmets for cyclists, Paris battles e-scooters, and more.

https://www.wired.com/story/uber-self-driving-crash-volvo-polestar-1-roundup/


NASA delays Mars helicopter Ingenuity's 1st flight to April 14

By Meghan Bartels 3 days ago

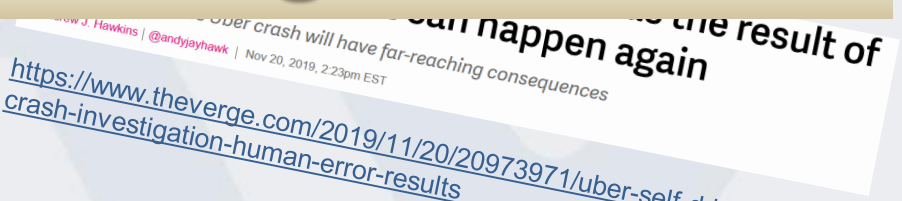
The little chopper was grounded after a test ended early.

https://www.space.com/nasa-mars-helicopter-flight-delay


Tesla's 'shatter… metaph…

…media/teslas-shatterproof-window-a-metaphor-for-…-driving-tech-industry/

What could be the failing reasons?


…Hawkins | @andyjayhawk | Nov 20, 2019, 2:23pm EST

…uber crash will have far-reaching consequences

…the result of …can happen again

https://www.theverge.com/2019/11/20/20973971/uber-self-driving-car-crash-investigation-human-error-results


The first Boeing 737 Max crash was 2 years ago today. Here's the complete history of the plane that's been grounded since 2 crashes killed 346 people 5 months apart.

David Slotnick Oct 29, 2020, 2:55 PM

https://www.businessinsider.com/boeing-737-max-timeline-history-full-details-2019-9


Microsoft: 30% of IoT projects fail in the proof-of-concept stage

Kyle Wiggers @Kyle_L_Wiggers July 30, 2019 8:00 AM Cloud

https://venturebeat.com/2019/07/30/microsoft-30-of-iot-projects-fail-in-the-proof-of-concept-stage/

Some Software Systems Characteristics

Lack of Quality*, mostly a side effect of the plague of *software defects

External Quality

FAILURES (observed)

Internal Quality

TECHNICAL DEBT (perceiveid)

Software Defects

Error: *a human action that produces an incorrect result*.

Fault: *a manifestation of an error in software*.

Failure: ***(a)*** *termination of the ability of a product to perform a required function or its inability to perform within previously specified limits; or* ***(b)*** *an event in which a system or system component does not perform a required function within specified limits*.

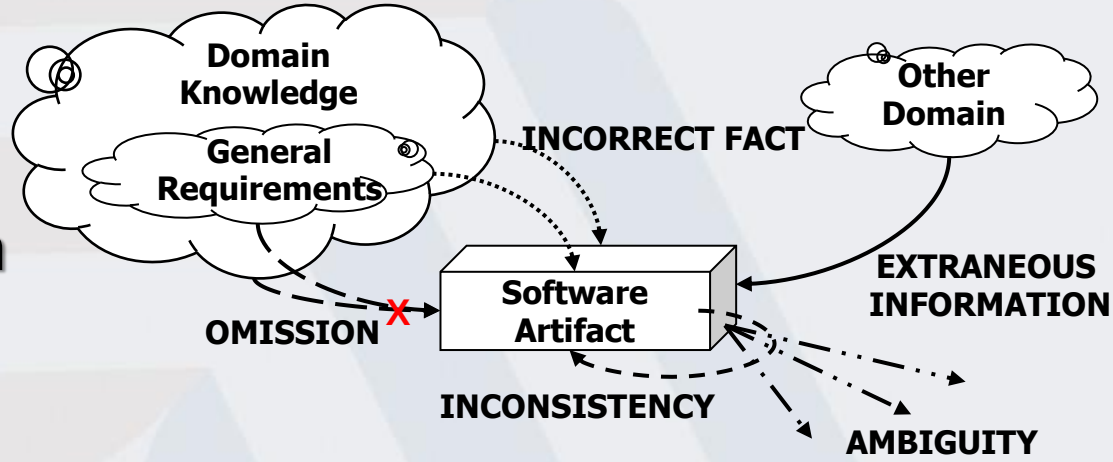
Defect:

an imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced.

It is a fault when detected during the execution of software

Software Defects

From where defects come from?



Most of them results from human based activities!

What types of defects we can find?

Defect	***General Description***
Omission	Necessary information about the system has been omitted from the software artifact.
Incorrect Fact	Some information in the software artifact contradicts information in the requirements document or the general domain knowledge.
Inconsistency	Information within one part of the software artifact is inconsistent with other information in the software artifact.
Ambiguity	Information within the software artifact is ambiguous, i.e. any of a number of interpretations may be derived that should not be the prerogative of the developer doing the implementation.
Extraneous Information	Information is provided that is not needed or used.

Software Defects

They are introduced due to communication or information transformation issues.

FORMAL

Scalene Triangle:

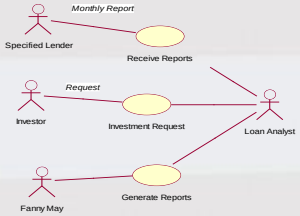
{<x,y,z>: (x != y) ^ (x != z) ^ (y != z)}

REQUIREMENTS

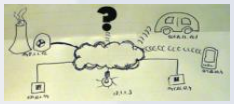
Loan-Arranger Requirements Specification – Jan. 8, 1999

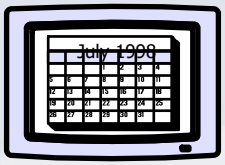
Background

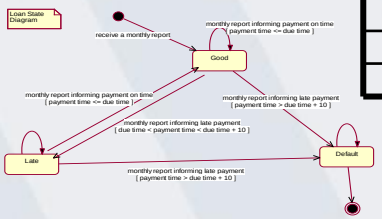
Banks generate income in many ways, often by borrowing money from their depositors at a low interest rate, and then lending that same money at a higher interest rate in the form of bank loans. However, property loans, such as mortgages, typically have terms of 15, 25 or even 30 years. For example, suppose that you purchase a $150,000 house with a $50,000 down payment and borrow a $100,000 mortgage from National Bank for thirty years at 5% interest. That means that National Bank gives you $100,000 to pay the balance on your house, and you pay National Bank back at a rate of 5% per year over a period of thirty years. You must pay back both principal and interest. That is, the initial principal, $100,000, is paid back in 360 installments (once a month for 30 years), with interest on the unpaid balance. In this case the monthly payment is $536.82. Although the income from interest on these loans is lucrative, the loans tie up money for a long time, preventing the banks from using their money for other transactions. Consequently, the banks often sell their loans to consolidating organizations such as Fannie Mae and Freddie Mac, taking less long-term profit in exchange for freeing the capital for use in other ways.



TEST CASES			
CLASS	X	Y	Z
Scalene	3	4	5
Isosceles	5	5	8
Isosceles	3	4	3
Isosceles	4	7	7
Eqüiláteral	2	2	2
No-triangle	1	2	3
No-triangle	5	1	4
No-triangle	3	5	2



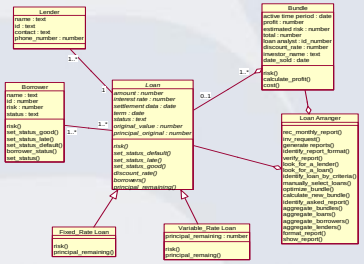




Tacit requirements

AD-HOC

Problem Domain



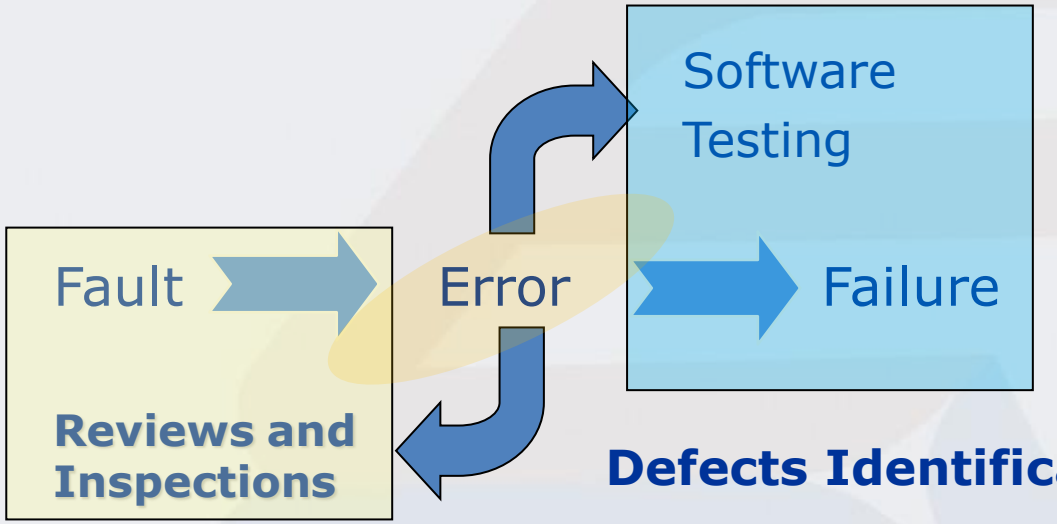
Solution Domain

Computer Domain

SOURCE CODE

Travassos, G.H. (2014). Software Defects: Stay Away from Them. Do Inspections!. In: 9th International Conference on the Quality of Information and Communications Technology (QUATIC) . Guimarães, Portugal. http://dx.doi.org/10.1109/QUATIC.2014.8

Defeating the Plague of Defects

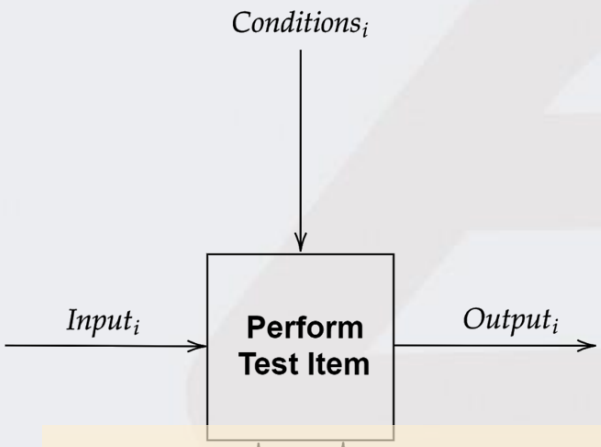


Defects Identification Techniques Efficiency

Technique	Efficiency
Software Reviews	25% a 40%
Software Inspections	45% a 65%
Code Reviews	20% a 35%
Code Inspections	45% a 70%
Unit Testing	15% a 50%
Integration Testing	25% a 40%
System Testing	25% a 55%
Beta Test (< 10 customers)	24% a 40%
Beta Test (> 1000 customers)	60% a 85%

Adapted from Capers Jones, Software defect-removal efficiency, IEEE Computer, April 1996, https://doi.org/10.1109/2.488361

Revealing Failures in Software Systems



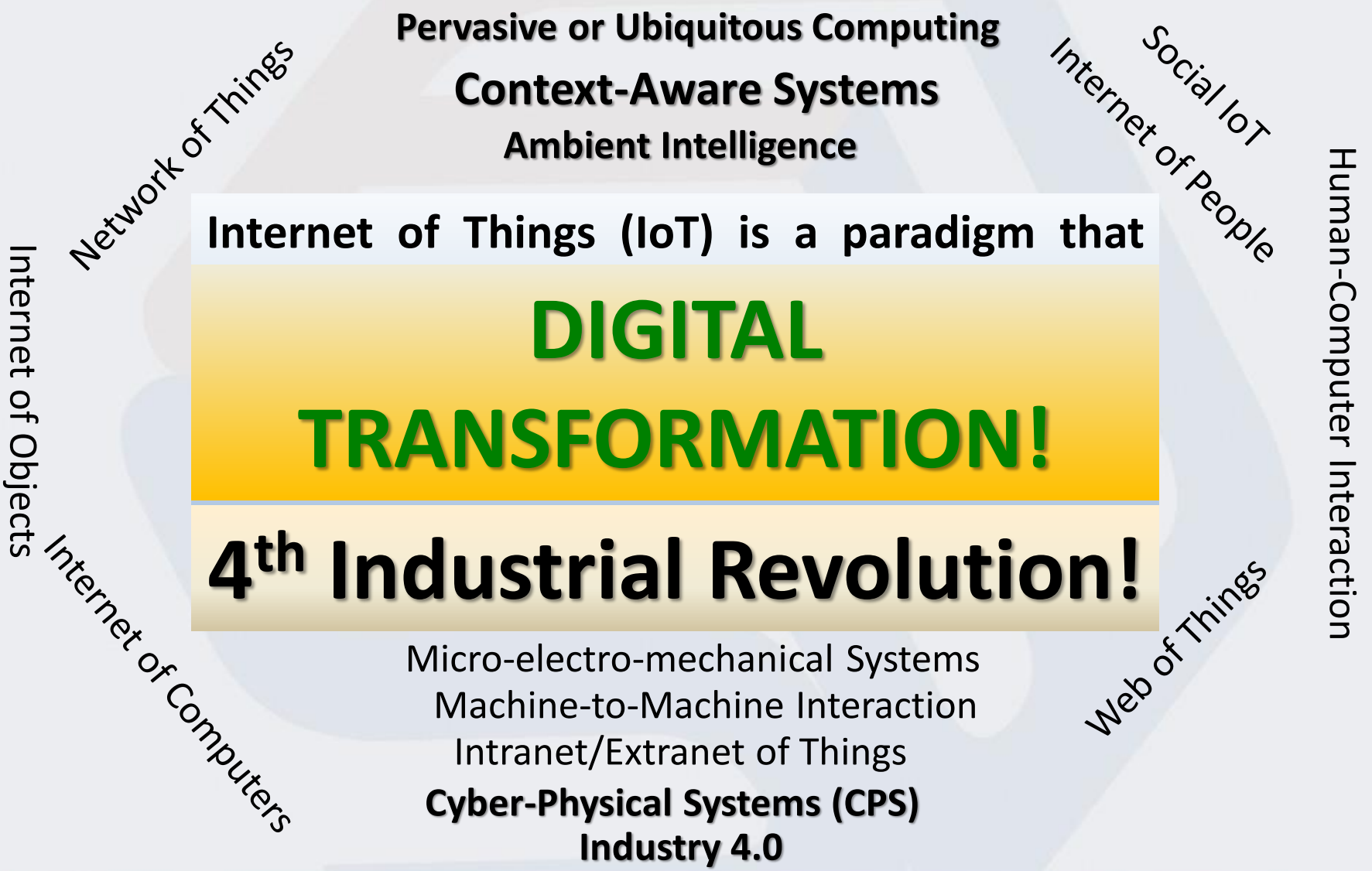
Usual Software Testing Perspective

So far, we accomplish it very well with first, second, and third stages' software systems.

$$TC = \{(I, C, E) : I \subset D_I, C \subset D_C, E \subset D_E\}$$

D_I : Input Domain	I: Input
D_C : Conditions Domain	C: Condition
D_E : Expected Result Domain	E: Expected Result

Contemporary Software Systems (CSS)



Contemporary Software Systems

Under the IoT paradigm perspective

Context-awareness

Security

Heterogeneity

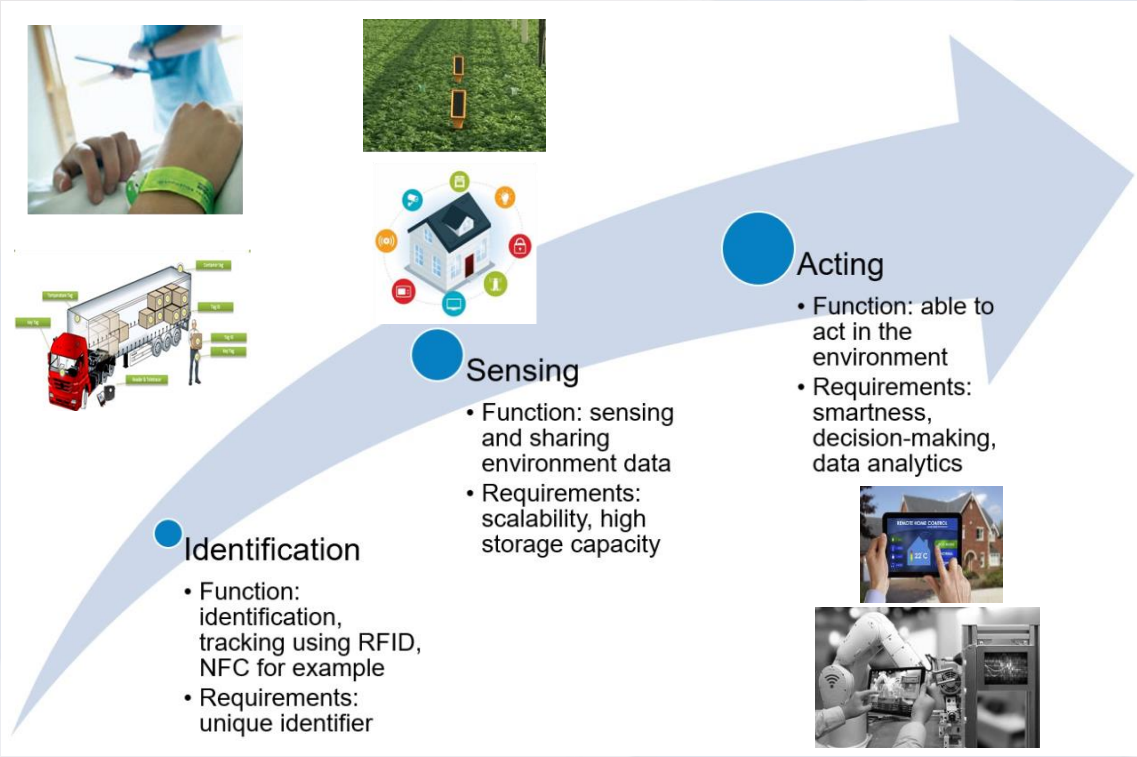
Interoperability

Mobility

Addressability

Unique ID

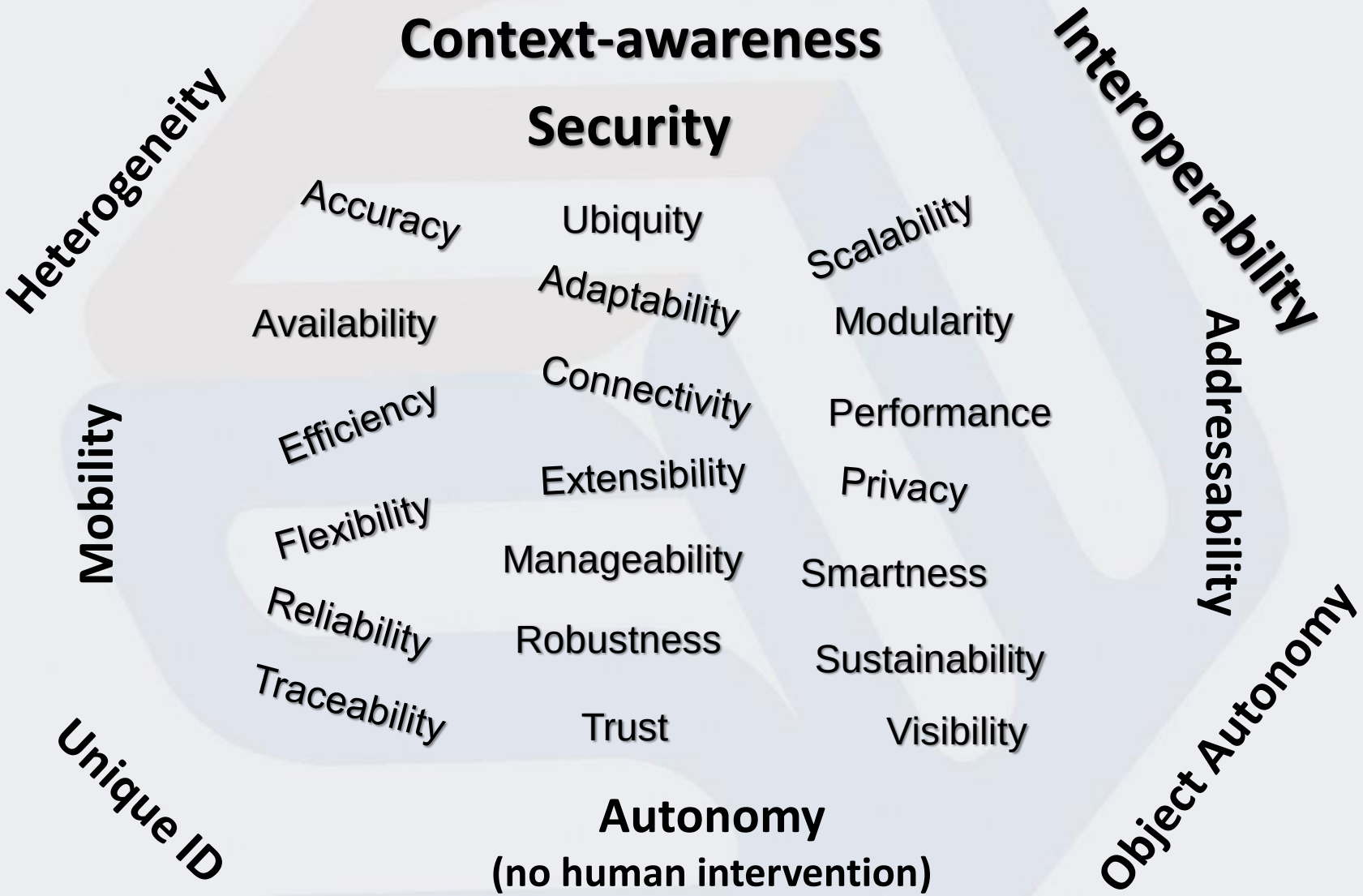
Object Autonomy



Autonomy

(no human intervention)

Challenges in CSS Engineering



Challenges in CSS Engineering

IOT FACETS

Problem Domain

It represents an area of expertise or application that should be examined to solve a problem. IoT software systems are developed to reach a goal for a specific purpose.

Things

It exists in the physical realm, such as sensors, actuators, or any objects equipped with identifying, sensing, or acting behaviors and processing capabilities that can communicate and cooperate to reach a goal, varying according to the systems requirements.

Connectivity

It is necessary to have available a medium by which *things* can connect to materialize the IoT. The idea is not to limit Internet-only connectivity but to represent different forms of connections.

Interactivity

It refers to the involvement of actors in the interaction to exchange information with things. The actors engaged with IoT applications are not limited to humans. Therefore, beyond the sociotechnical human-thing interaction, it should consider non-human actors and the thing-thing interaction.

Smartness

It refers to orchestration associated with things and to what level of intelligence with technology it evolves, allowing things to acquire a higher or lower degree of smartness. A smart system needs a set of actions, for example, treating data, making decisions, and acting through software.

Data

It regards the activities and technologies necessary to treat the data captured from the environment and other devices, such as data analysis and processing, to give meaning and achieve the system's goal.

Behavior

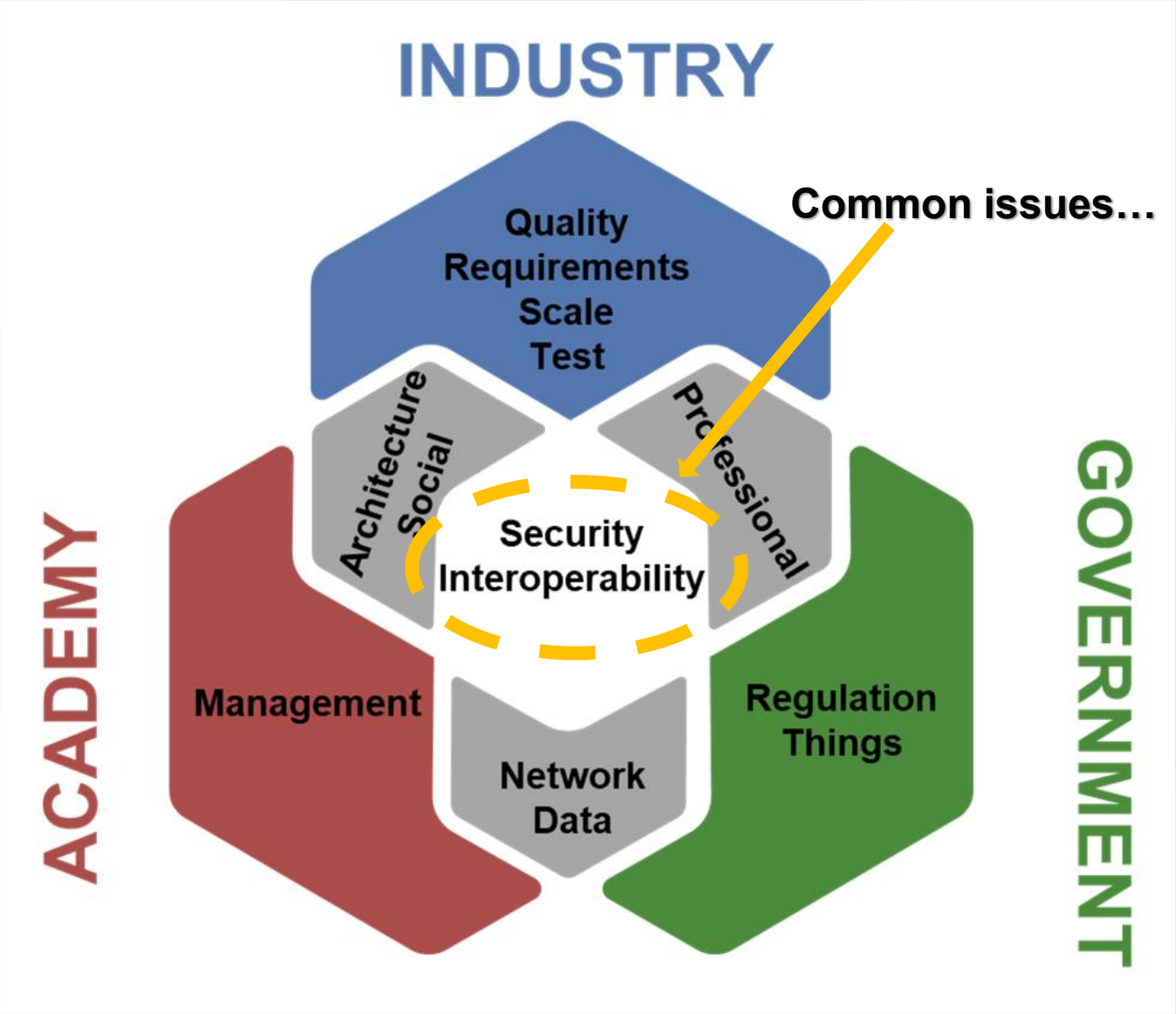
It provides the chance of enhancements in the things, extending their original behaviors. It relates to functions that enable Identification, Sensing, and Actuation behaviors, for example.

Environment

It is the place holding *things*, actions, events, and people. IoT systems provide smart services to adapt to users' needs and behavior according to the context of a given environment.

IoT CONCERNS

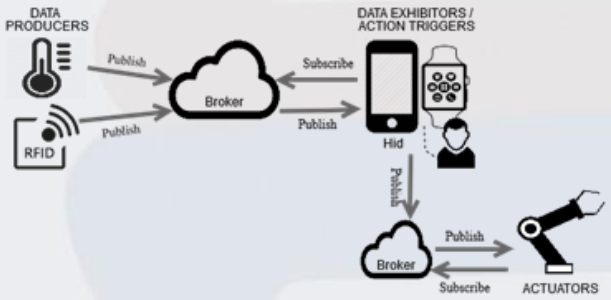
Challenges in CSS Engineering



Challenges in CSS Engineering

REQUIREMENTS

Scenarl$_{oT}$ provides nine information Catalogs to support the description of scenarios in the development of IoT-based software systems



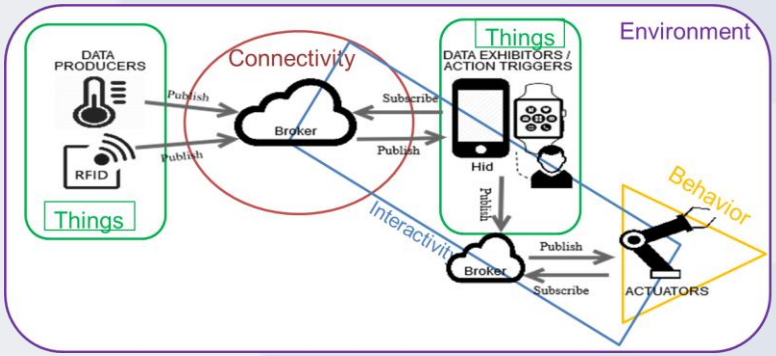
Catalog for the - IIA-4: Actuation triggered by an individual, based on IoT data.

Catalog – Actuation triggered by an individual, based on IoT data	
Entity	**Related Information**
Data producers	1. Who collects data {Sensors, Tag readers}
	2. What data is collected (temperature, humidity, among others)
	3. Source of data (rooms, a cup of coffee, refrigerator, ground, among others)
Data Exhibitor (Hid)	1. What exhibits data (Ex. Devices running user applications)
	2. Data format
Data consumer and Action trigger (human)	1. Who accesses data (Ex. Person, Persona, Profile, Role, among others).
	2. Data semantics (the meaning of data according to who visualizes it)
Action Performers	1. What performs action {IoT actuator}
	2. Type of action (Ex. Circular motion, Straight-line motion, On/Off circuit, among others)

Challenges in CSS Engineering

REQUIREMENTS

SCENARI$_{OT}$CHECK is a checklist-based inspection technique to support the verification of IoT scenarios produced with **SCENARI$_{oT}$**



General Questions	01	Has the overall application domain been established? (Health, leisure, traffic)
	02	Is the specific purpose of the system correctly described? (Data visualization, visualization, decision making, and actuation only)
	03	Is the type of data collected specified? (Temperature, humidity, pollution)
	04	Is it possible to identify who or what collects the data? (Sensors, QR code readers)
	05	Is it possible to identify who or what manages the data collected? (Administrator, decision-maker, users)
	06	Is it possible to identify who or what accesses the data collected? (Things, software systems, users)
	07	Is the user interface device that displays the data described? (da
	08	Is it possible t (Things, softw
	09	Is it possible t provided? (Ch
	10	Are the roles i software syste
	11	Is there any de scenarios?
	12	Is it possible t
	13	Has each actio clearly and co
	14	Is there any se comprehensio
Specific Questions	24	Is it possible to identify the specific context in which the system is embedded? (Smart room, smart greenhouse, autonomous vehicle, healthcare)
	25	Are the limitations of the environment described? (e.g., lack of connectivity structure, lack of hardware structure, inadequate infrastructure)
	26	Are the technologies associated with system objects described? (smartphones, smartwatches, wearables)
	27	Are the events that the system has identified? (e.g., on/off an object, sending data)
	28	What kind of communication technology does the system use in the scenarios? (Bluetooth, intranet, internet ...)
	29	Does the proposed communication technology meet the geographic/physical specifications of the system? (Large, medium or small scale)
	30	Is it possible to identify how the system will react according to changes in the environment?
	31	Are the interactions between the system and the environment represented in the scenarios?
	32	Is it possible to identify the interaction between actors?

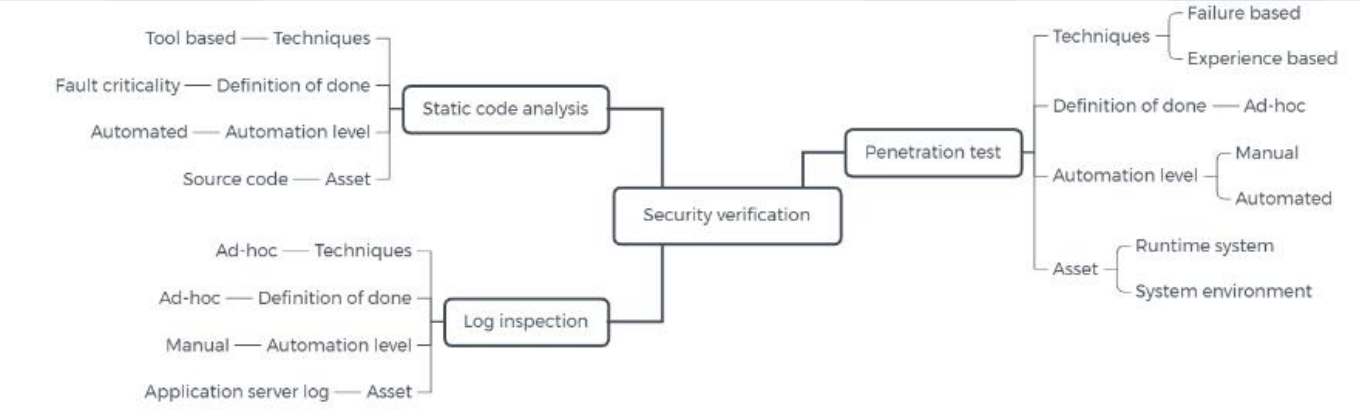
De Souza, B. P.; Motta, R. C.; Costa, D. O.; Travassos, G. H. (2019) An IoT-based Scenario Description Inspection Technique. In: SBQS 2019: ACM Press, 2019. p. 20. http://dx.doi.org/10.1145/3266237.3266263

Challenges in CSS Engineering

SECURITY

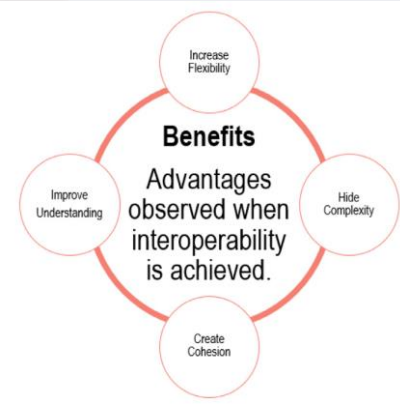
NFR	Sub-NFR	Description
Security	System capability to protects data or resources from people or other systems that do not have access permission, providing correct access level from people or other systems that have access permission. It means that the system must have the capability to continue providing essential services even under attack.	
	Confidentiality	System capability to allow only users with an appropriate access level to access resources (data, print, services, etc.), including data traffic.
	Auditability	The availability of auditing capability of service invocations that can be traced to specific users for logging and repudiation.
	Vulnerability	System capability to prevent attacks. It
		means that the system must have the capability to continue providing essential services even under attack.

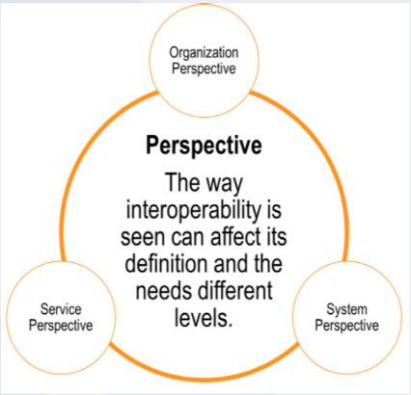
1. PROMOTE ORGANIZATIONAL AWARENESS OF THE IMPORTANCE OF SECURITY AND PERFORMANCE
2. KEEP A CROSS-FUNCTIONAL TEAM
3. PRODUCE CLEAR REQUIREMENTS
4. SELECT SUITABLE SUPPORT TOOLS
5. CONFIGURE AN ADEQUATE VERIFICATION ENVIRONMENT
6. USE A SYSTEMATIC VERIFICATION METHODOLOGY
7. PLAN SECURITY AND PERFORMANCE VERIFICATION ACTIVITIES
8. ENCOURAGE REUSE PRACTICES



Challenges in CSS Engineering

INTEROPERABILITY





Challenges in CSS Engineering

CONTEXT AWARENESS

Context is any piece of information that may be used to characterize an entity's situation (logical and physical objects present in the system's environment) and the relations relevant to the actor-computer interaction between actors and computers.

G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a Better Understanding of Context and Context-Awareness, in: H.-W. Gellersen (Ed.), Handheld and Ubiquitous Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999: pp. 304–307. https://doi.org/10.1007/3-540-48157-5_29 .

Context-awareness is a dynamic property of a software system that can evolutionarily affect its overall behavior in the interaction between actors and computers.

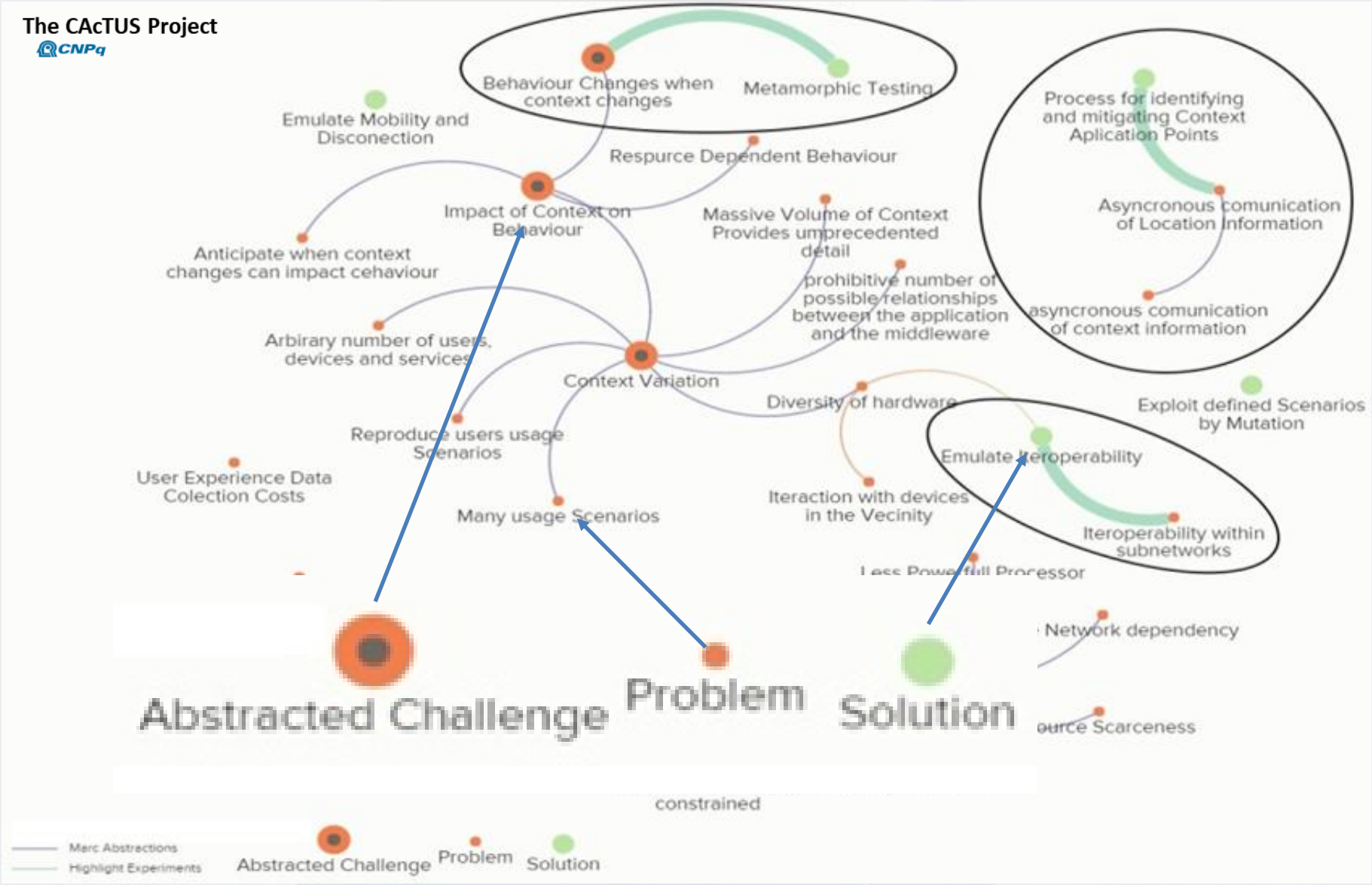
How to test CSS?

Cont[illegible] can iden[illegible]ment (i.e., context) and adapt their behavior to provide better service to the actor.

What we can see in research?

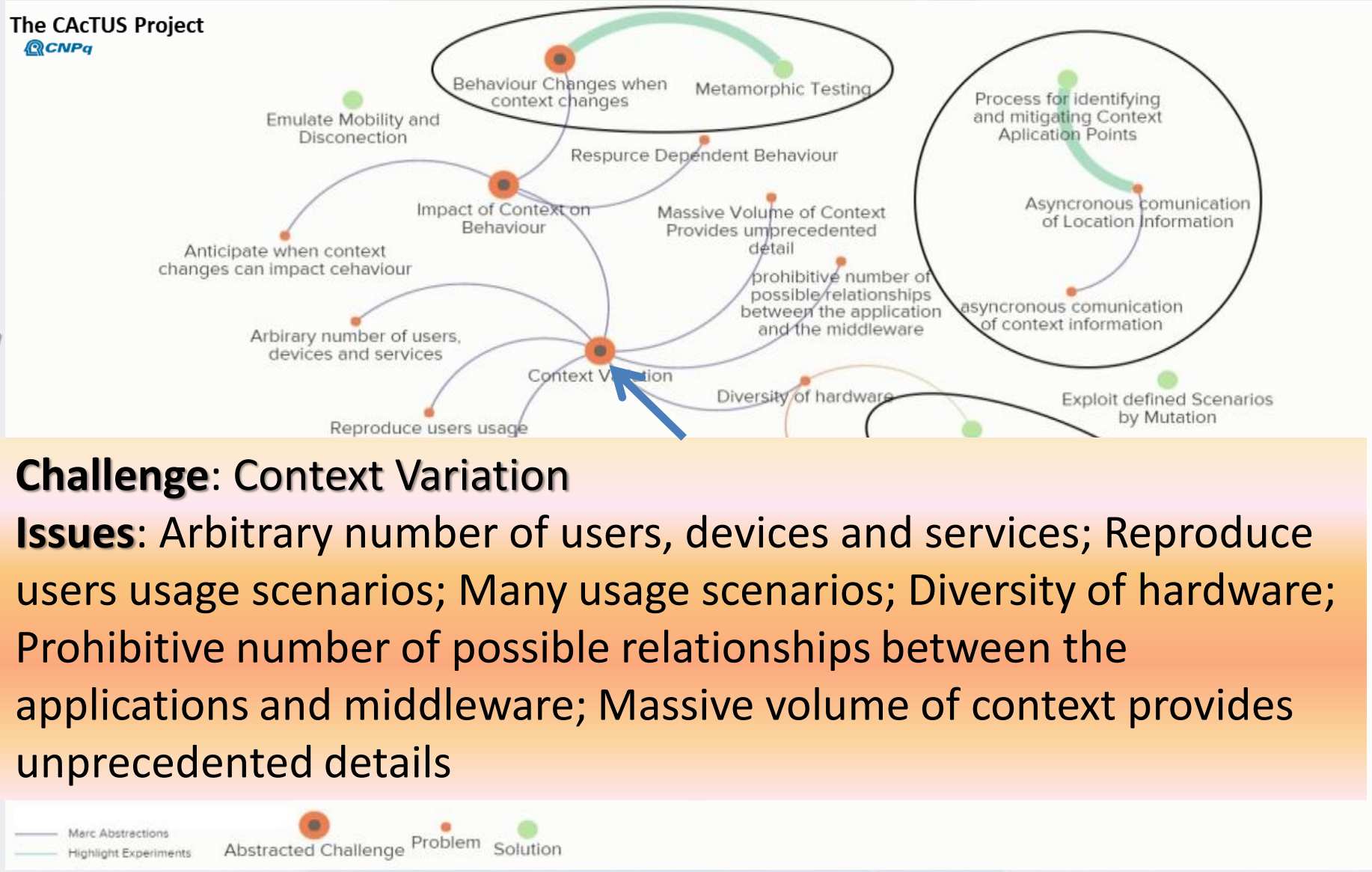
Challenges in Testing CSS

CONTEXT AWARE TESTING



Challenges in Testing CSS

CONTEXT AWARE TESTING



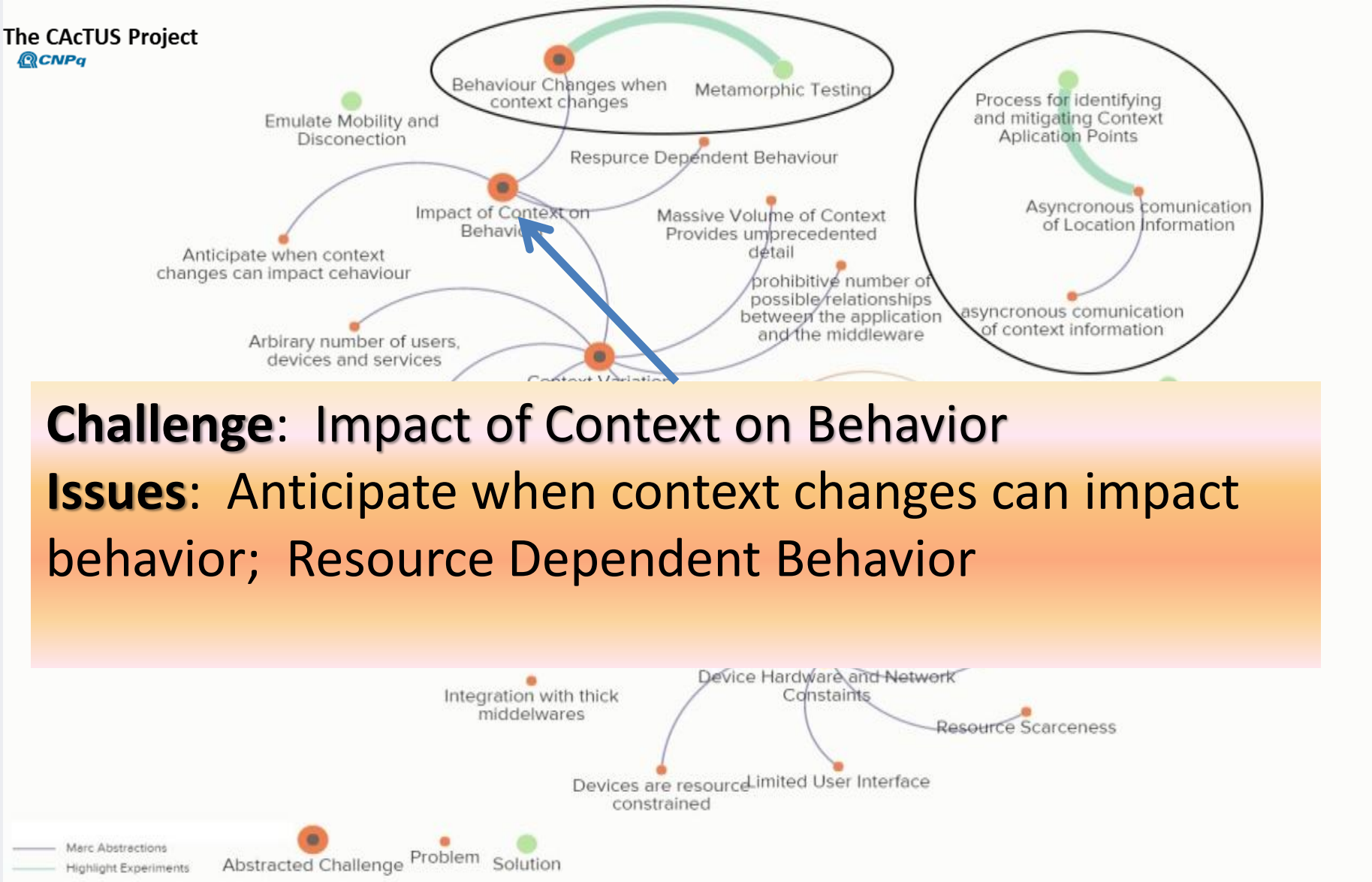
Challenge: Context Variation

Issues: Arbitrary number of users, devices and services; Reproduce users usage scenarios; Many usage scenarios; Diversity of hardware; Prohibitive number of possible relationships between the applications and middleware; Massive volume of context provides unprecedented details

Challenges in Testing CSS

CONTEXT AWARE TESTING

The CAcTUS Project
CNPq



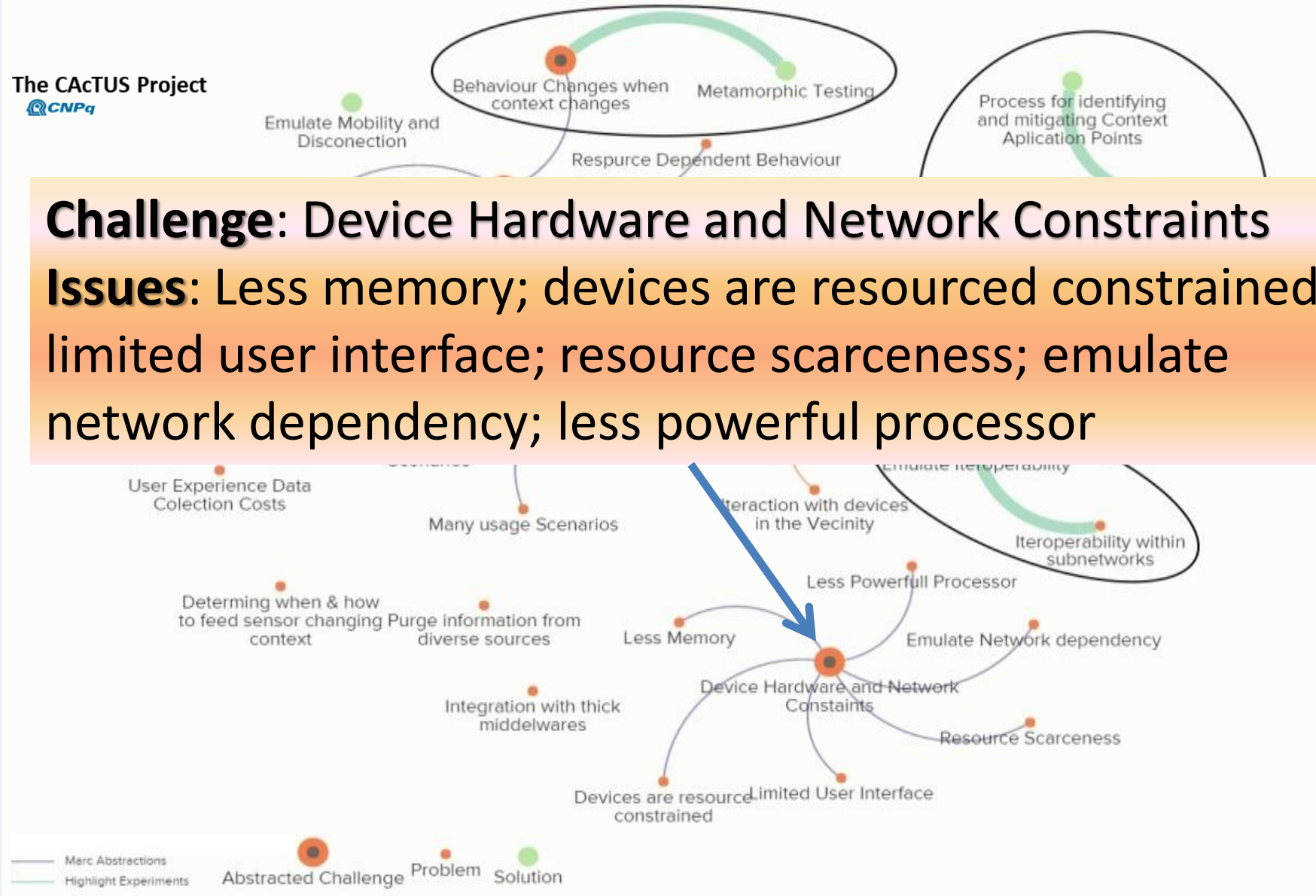
Challenge: Impact of Context on Behavior

Issues: Anticipate when context changes can impact behavior; Resource Dependent Behavior

Challenges in Testing CSS

CONTEXT AWARE TESTING

The CAcTUS Project
CNPq

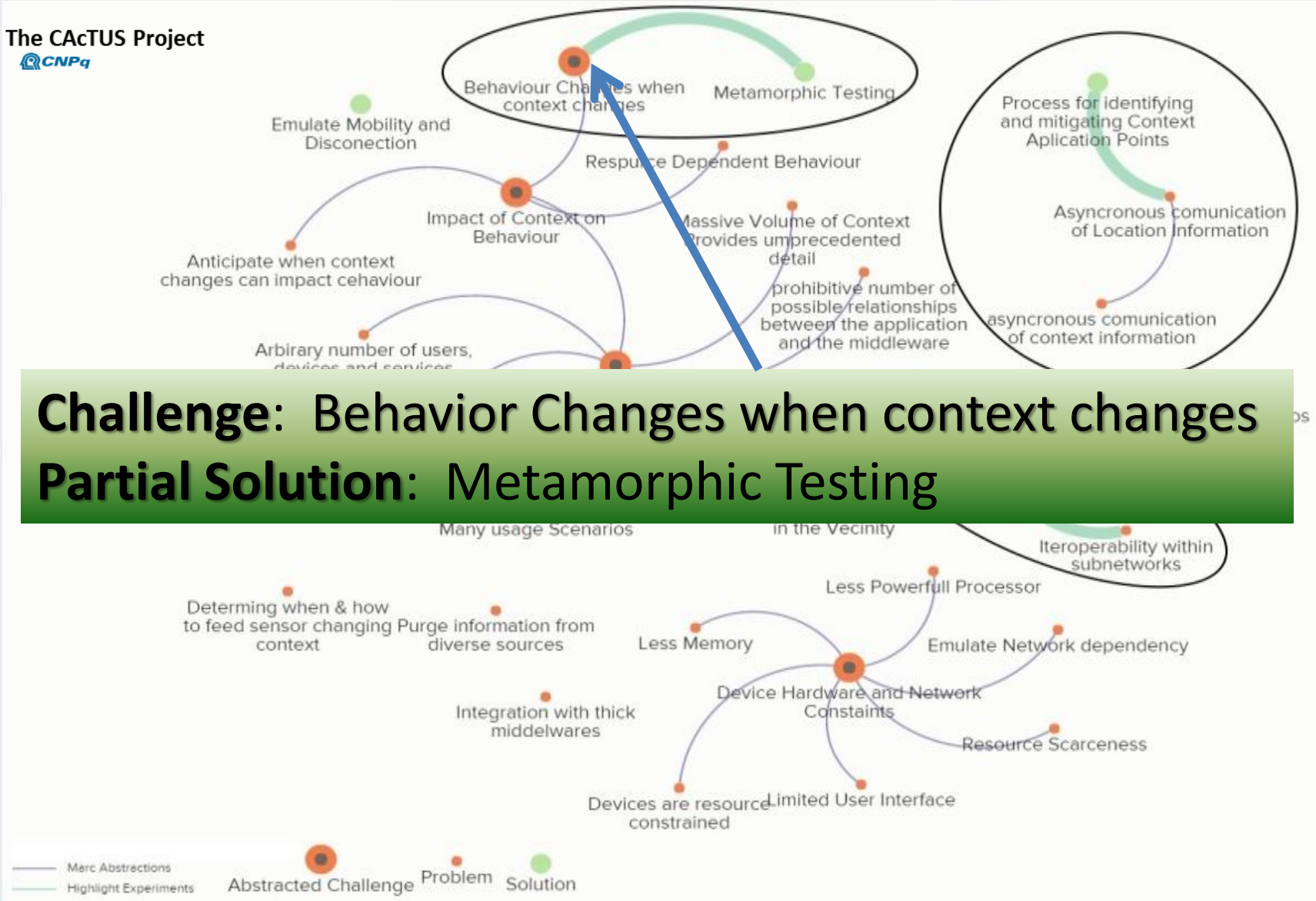


Challenge: Device Hardware and Network Constraints

Issues: Less memory; devices are resourced constrained; limited user interface; resource scarceness; emulate network dependency; less powerful processor

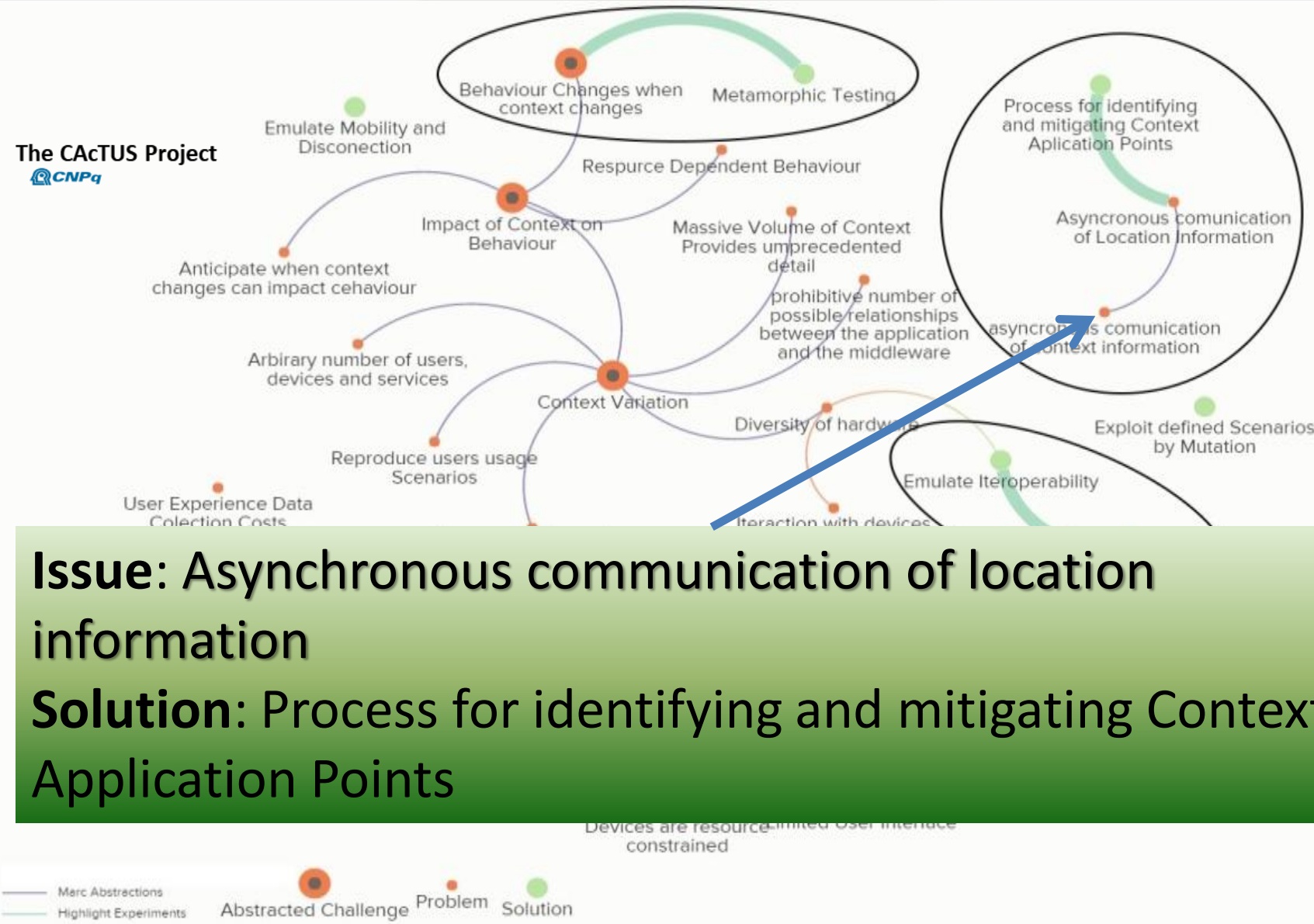
Challenges in Testing CSS

CONTEXT AWARE TESTING



Challenges in Testing CSS

CONTEXT AWARE TESTING



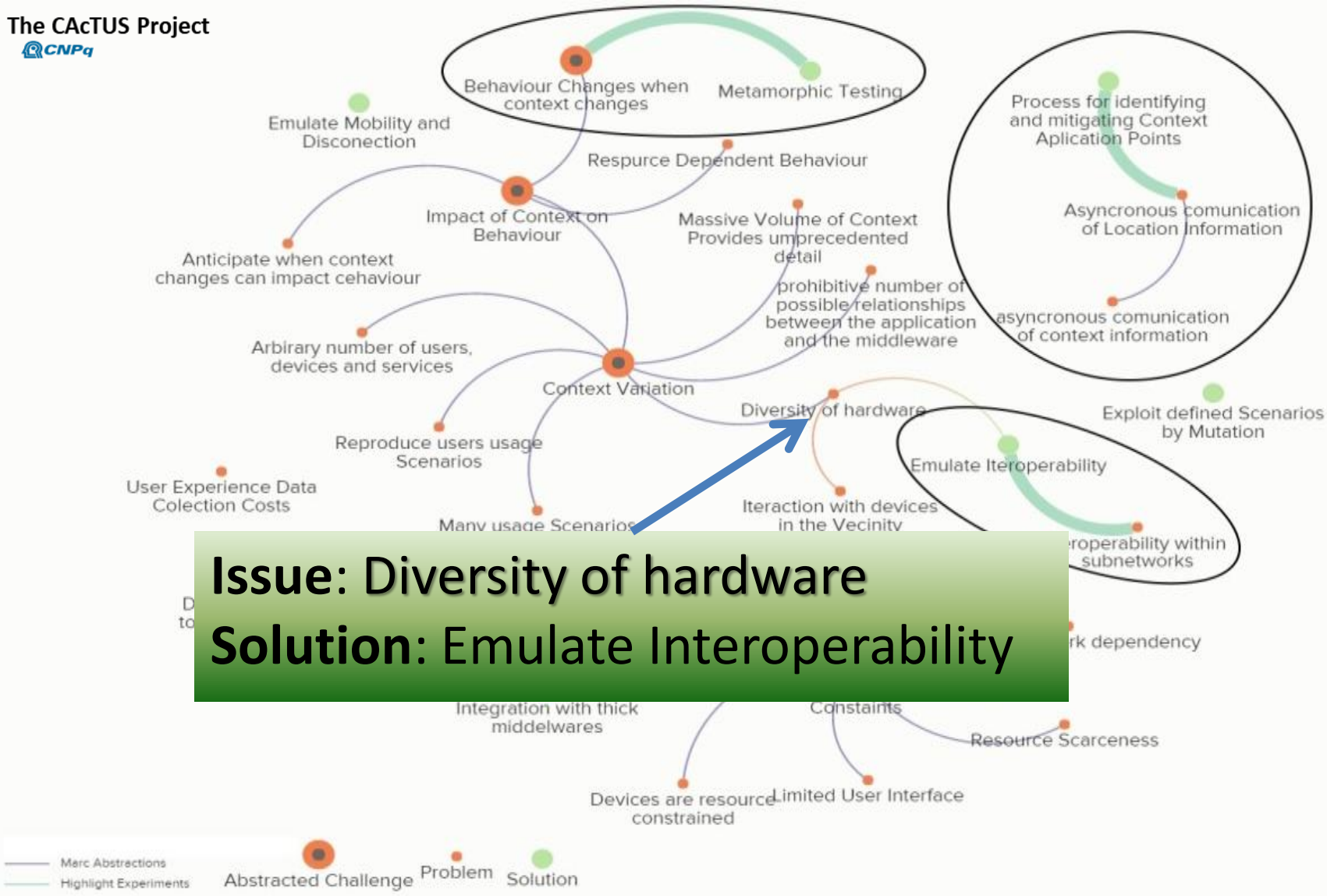
Issue: Asynchronous communication of location information

Solution: Process for identifying and mitigating Context Application Points

Challenges in Testing CSS

CONTEXT AWARE TESTING

The CAcTUS Project
CNPq



Issue: Diversity of hardware
Solution: Emulate Interoperability

Challenges in Testing CSS

CONTEXT AWARE TESTING

"A few test case design techniques (TCDT) partially support context-aware software systems testing (CASS). However, there has not been observed evidence on any TCDT supporting the truly context-aware testing, which can adapt the expected output based on the context variation (dynamic perspective) during the test execution. It is an open issue deserving researchers' greater attention to increase the testing coverage and ensure users' confidence in CASS."

The CAcTUS Project
CNPq

Challenges in CSS Engineering

CONTEXT AWARENESS

Context is any piece of information that may be used to characterize an entity's situation (logical and physical objects present in the system's environment) and the relations relevant to the actor-computer interaction between actors and computers.

G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a Better Understanding of Context and Context-Awareness, in: H.-W. Gellersen (Ed.), Handheld and Ubiquitous Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999: pp. 304–307. https://doi.org/10.1007/3-540-48157-5_29 .

Context-awareness is a dynamic property of a software system that can evolutionarily affect its overall behavior in the

How to test CSS?

Context-aware contemporary software systems can ide

What we can see in practice?

ent (i.e., context) and adapt their behavior to provide better service to the actor.

Challenges in Testing CSS

CONTEXT AWARENESS

- Current Software Testing Technologies (in general):
  - They cannot completely cover all the test input space of a context-aware application, which limits the possible coverage of the resulting test cases.
  - it is unlikely that a test oracle can be defined for all possible values (and/or a combination of) values that can stimulate the context-aware test item.
  - it is not feasible to define a test oracle for each possible combination of context variable values.

Challenges in Testing CSS

CONTEXT AWARENESS

- What jeopardizes our evolution in the practice:
  - Testing context-awareness features of contemporary software systems require a model capable of modeling the system's dynamic behavior.
  - Lack of technologies for developing CSS models.
  - Testing CSS is cost-intensive and requires the exploitation of computational resources.
  - Requirements for driving the variation of context in Test Environments vary according to the Software Development Life Cycle.

Challenges in Testing CSS

CONTEXT AWARE TESTING

While such software testing technologies are not available :

a) **Accept the nature of context and differentiate that the test item is subjected to different input types - *The test input and the input from the context***

b) **Start with a dynamic system model**

c) **Assure functional correctness before turning to test context-aware requirements**

d) **Design test cases to target context variables**

e) **Take advantage of automatic testing tools as much as possible**

f) **Manage the context - and the exposure of the test item to the context!**

Challenges in CSS Engineering

CONTEXT AWARENESS

Context is any piece of information that may be used to characterize an entity's situation (logical and physical objects present in the system's environment) and the relations relevant to the actor-computer interaction between actors and computers.

G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a Better Understanding of Context and Context-Awareness, in: H.-W. Gellersen (Ed.), Handheld and Ubiquitous Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999: pp. 304–307. https://doi.org/10.1007/3-540-48157-5_29 .

Context-awareness is a dynamic property of a software system that can evolutionarily affect its overall behavior in the

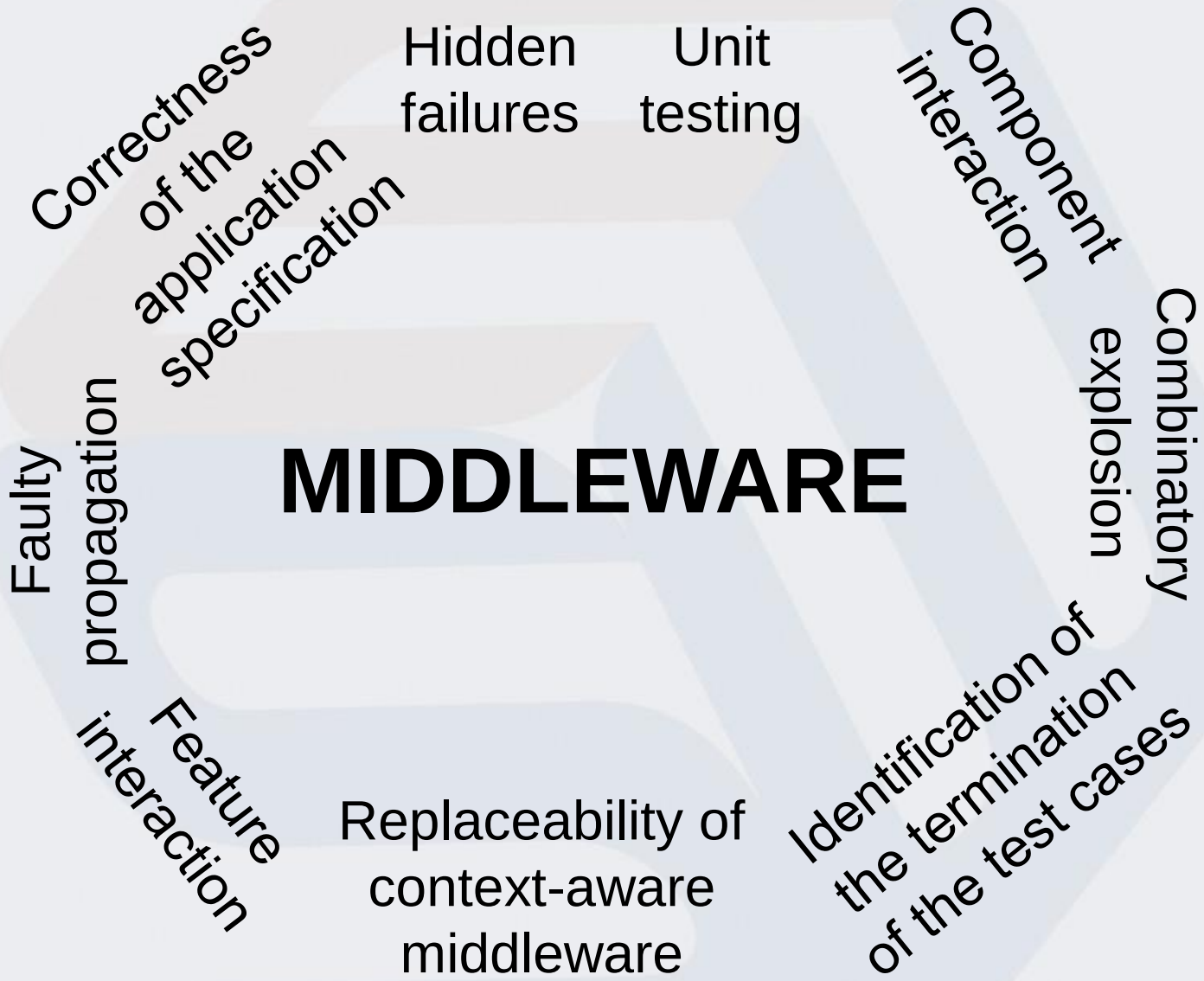
How to test CSS?

Context-aware contemporary software systems can i t (i.e., context) and adapt their behavior to provide better service to the actor.

What could make a difference?

Challenges in Testing CSS

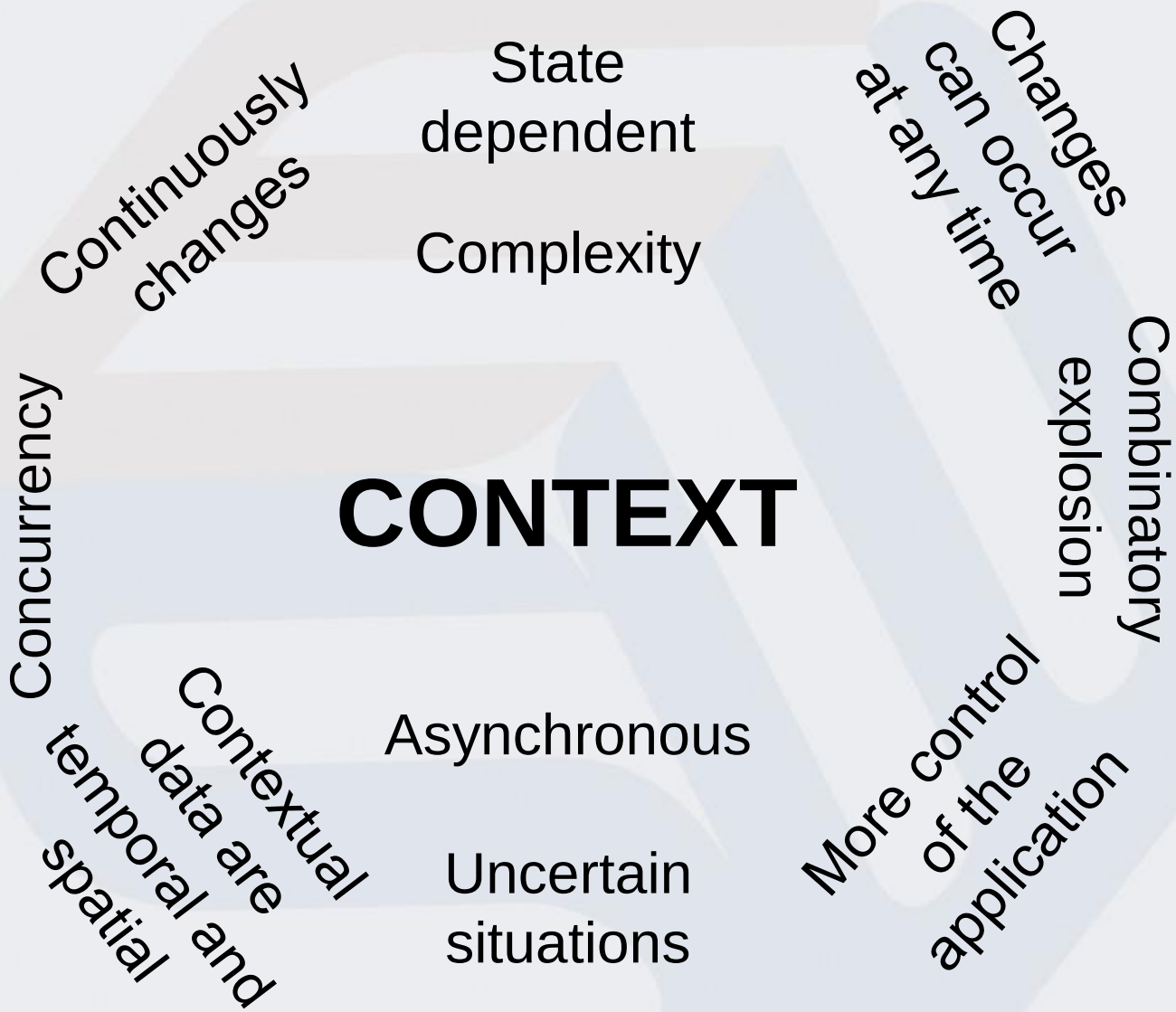
CONTEXT AWARE TESTING



Santos, I.S.; Andrade, R.M.C; Rocha, L.S.; Matalonga, S.; Oliveira, K. M.; Travassos, G.H. (2017). Test case design for context-aware applications: Are we there yet?. Information and Software Technology. V. 88, pp. 1-16. https://doi.org/10.1016/j.infsof.2017.03.008

Challenges in Testing CSS

CONTEXT AWARE TESTING



Challenges in Testing CSS

CONTEXT AWARE TESTING

To use the context information as a test coverage item

To define the test context

To design test cases for dealing with uncertainty

Test design

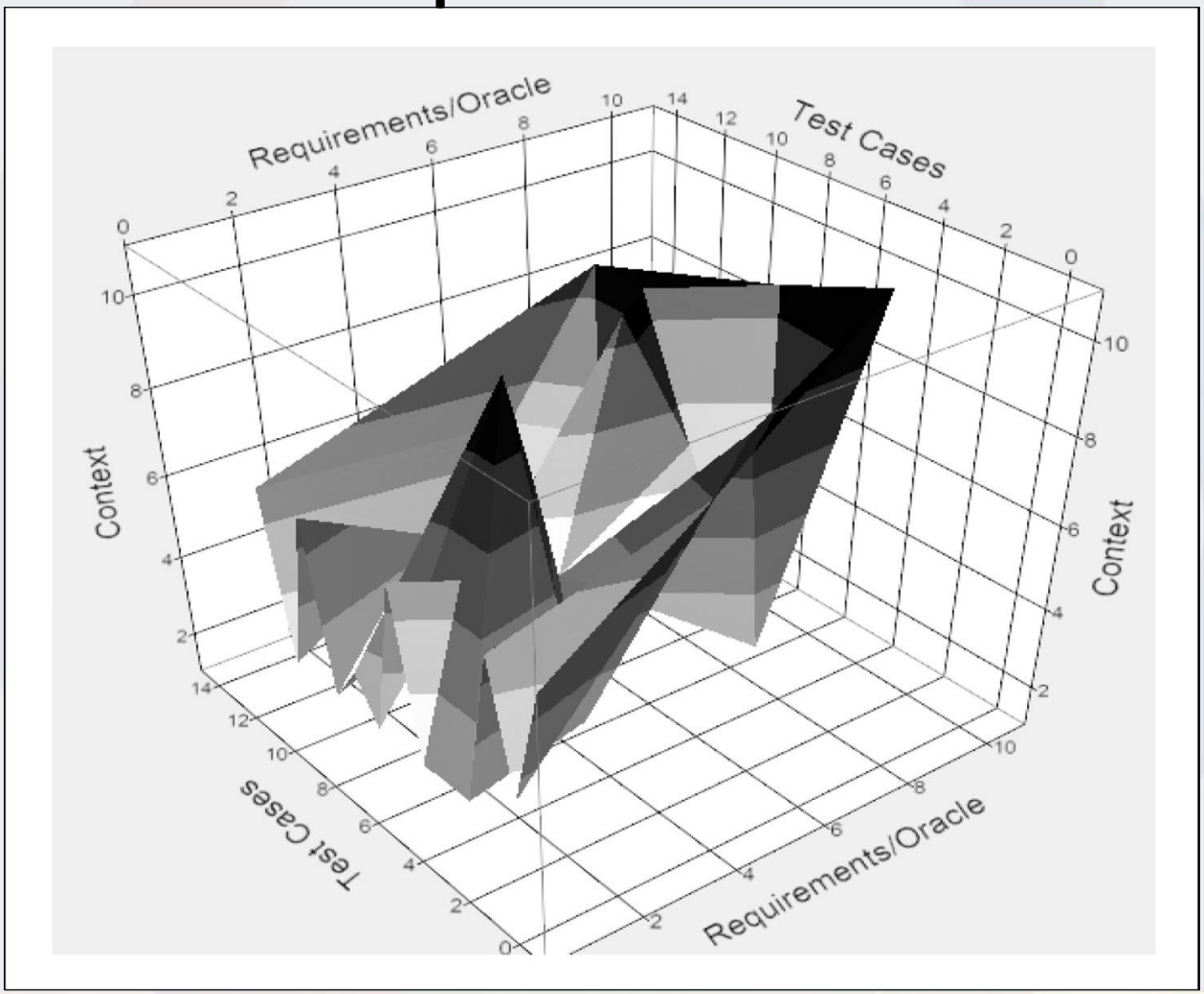
To reflect the context-awareness in the test cases

To specify when the expected output should be assessed

CONTEXT AWARE TESTING

Challenges in Testing CSS

"The context should freely vary during test execution as it does in production environments"

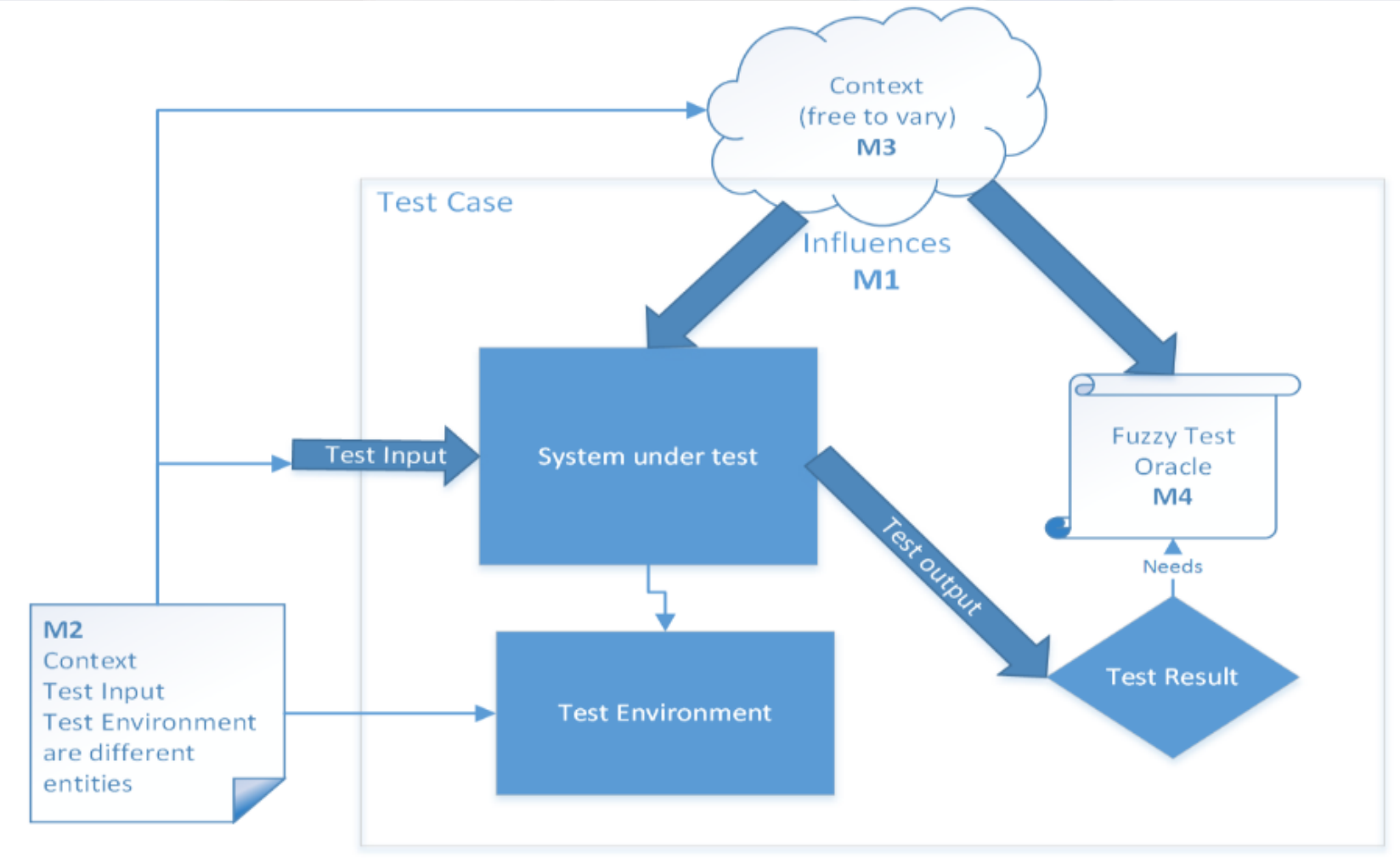


Matalonga, S.; Rodrigues, F.; Travassos, G. H. Characterizing testing methods for context-aware software systems: Results from a quasi -systematic literature review. The Journal of Systems and Software 131 (2017) 1–21. doi: http://dx.doi.org/10.1016/j.jss.2017.05.048

CONTEXT AWARE TESTING

Challenges in Testing CSS

"The context should freely vary during test execution as it does in production environments"

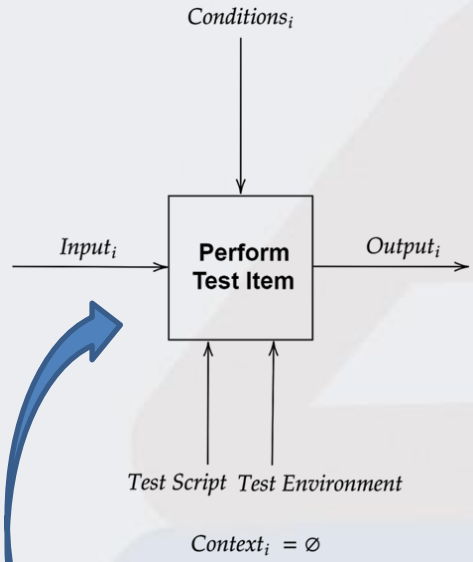


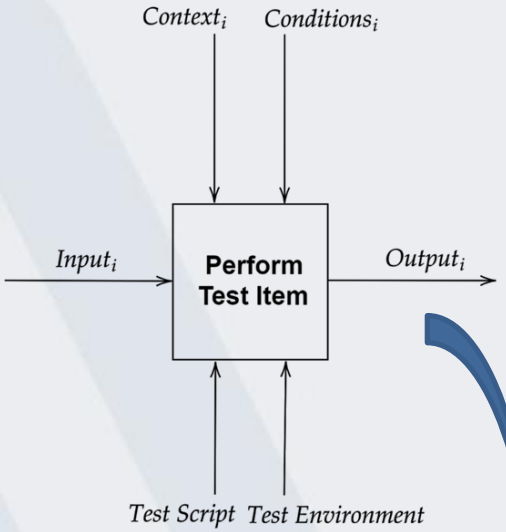
Matalonga, S. ; Travassos, G. H. (2017). Testing context-aware software systems: Unchain the context, set it free! In: SBES'17. p. 250-254. http://dx.doi.org/10.1145/3131151.3131190

Challenges in Testing CSS

CONTEXT AWARE TESTING

Test Case Perspectives

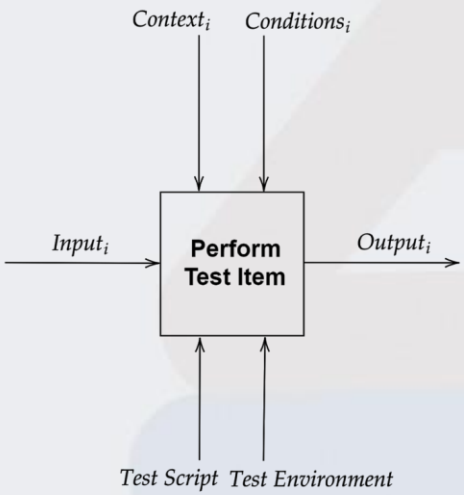




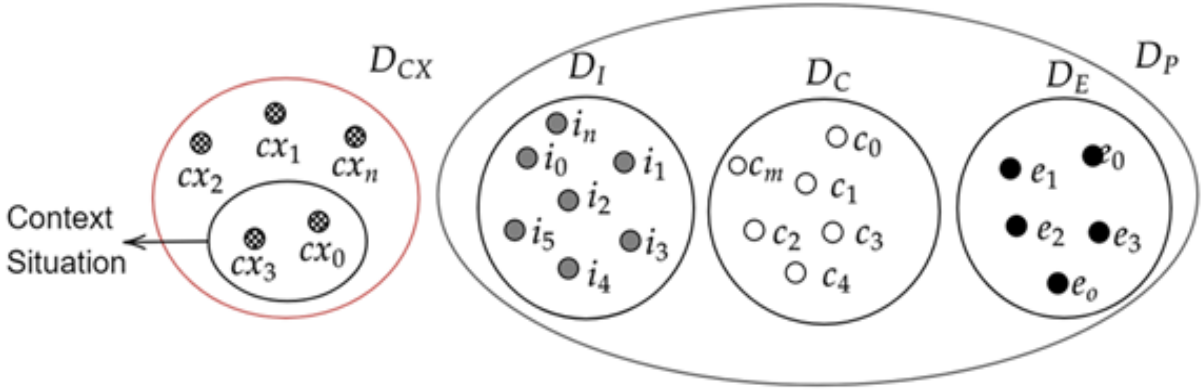
Test Case Model	Input (I)	Condition (C)	Expected Result (E)
Usual Test Case	Static Value	Static Value	Static Value
CSS Model A	Dynamic Value	Static Value	Dynamic Value
CSS Model B	Static Value	Dynamic Value	Dynamic Value
CSS Model C	Dynamic Value	Dynamic Value	Dynamic Value

Challenges in Testing CSS

CONTEXT AWARE TESTING



CATS#: A CSS Test Cases Model



$$TC = \{I_{cx}, C_{cx}, E_{cx}\}$$

$$I_{cx} = \{(I, S_i) : I \in D_I, S_i \in D_{CX}\}$$
$$C_{cx} = \{(C, S_i) : C \in D_C, S_i \in D_{CX}\}$$
$$E_{cx} = \{(E, S_i) : E \in D_E, S_i \in D_{CX}\}$$

$$I = \{i_i : i_i \in D_I\}$$
$$C = \{c_i, : c_i \in D_C\}$$
$$E = \{e_i : e_i \in D_E\}$$
$$S = \{cx_i : cx_i \in D_{CX}\}$$

a. No Context	b. Context influences the Input
$I_{cx} = \{i_i : i_i \in D_I\}$ $C_{cx} = \{c_i : c_i \in D_C\}$ $E_{cx} = \{e_i : e_i \in D_E\}$	$I_{cx} = \{(I, S_i)\}$ $C_{cx} = \{C\}$ $E_{cx} = \{(E, S_i)\}$
c. Context influences the Conditions	**d. Context influences the Conditions and the Input**
$I_{cx} = \{I\}$ $C_{cx} = \{(C, S_i)\}$ $E_{cx} = \{(E, S_i)\}$	$I_{cx} = \{(I, S_i)\}$ $C_{cx} = \{(C, S_i)\}$ $E_{cx} = \{(E, S_i)\}$

Challenges in Testing CSS

CONTEXT AWARENESS

- **Future Software Testing Research Suggestions:**
  - Evaluate the efficacy of real world models and their capacity to represent the production environments.
  - Measure the coverage of the Test Suites.
  - ...ting
  - **The Engineering of CSS offers Challenges and Pitfalls**
  - Manage the CSS test activities, since practices and procedures should evolve to manage the context and test suites in different environments. The CSS must be dynamically tested!
  - Use artificial intelligence approaches for dealing with the context complexity and reducing the costs of dynamic testing processes for CSS.

The Experimental Software Engineering Group at COPPE/UFRJ

(ese.cos.ufrj.br)

- **Experimental Software Engineering (ESE)** is one of the research topics of Software Engineering in the Department of Systems Engineering and Computer Science (PESC) of COPPE/UFRJ (www.coppe.ufrj.br).

- It aims to evolve SE Knowledge through experimentation and using truly engineering principles and practices to build contemporary software systems.

- One of our initiatives is **DELFOS** – Observatory of the Engineering of Contemporary Software Systems, a promoter of pre-startups.

Facing the CSS Engineering Challenges and Pitfalls: Evidence-Based Software Engineering!

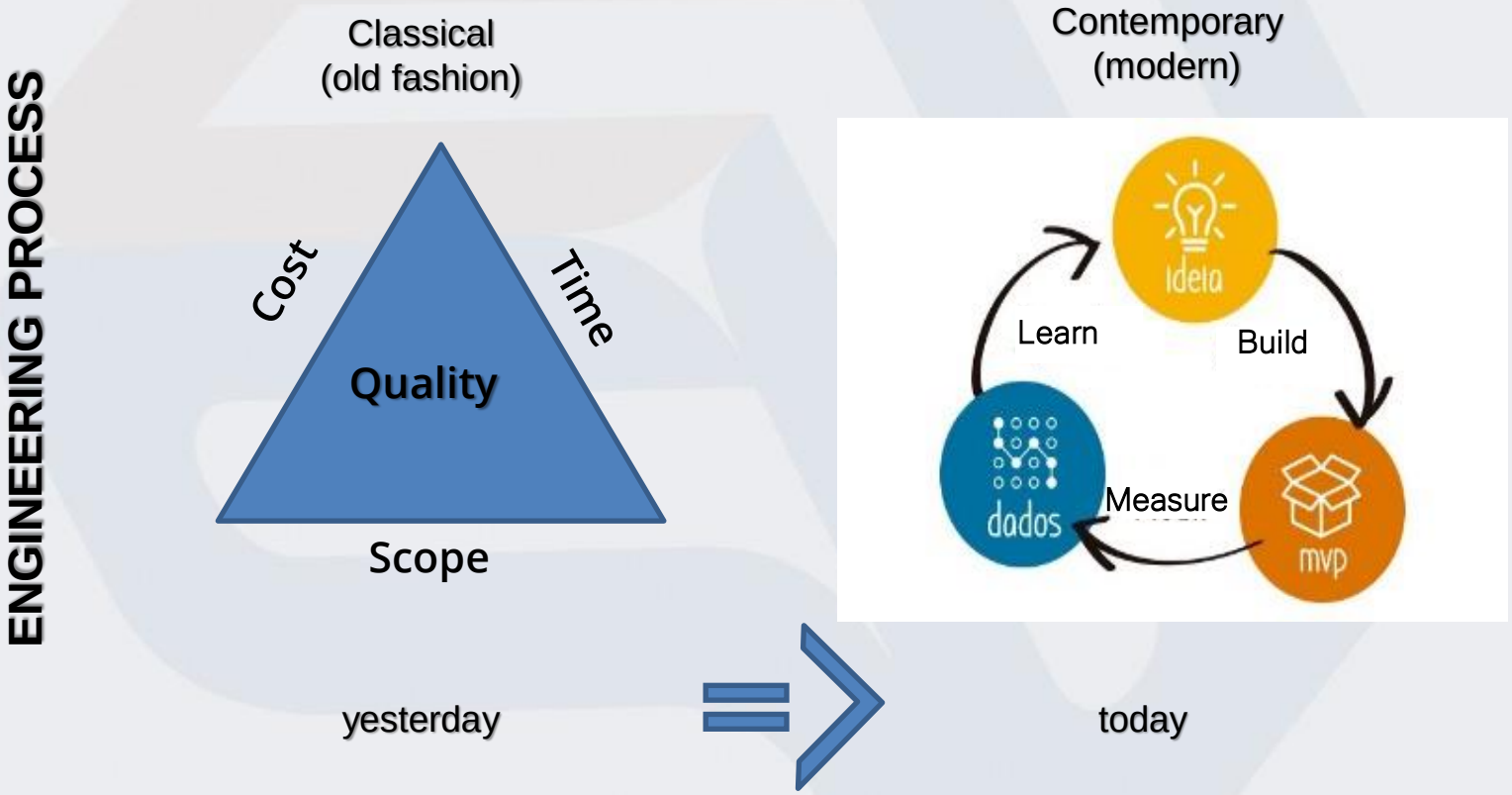


Figure adapted from Paulo Caroli, **Lean Inception: How to Align People and Build the Right Product**, 2018. https://leanpub.com/tothepoint.

Facing the CSS Engineering Challenges and Pitfalls: Evidence-Based Software Engineering!

Evolve the Engineering Process:

- Use newer programming languages
- Simple artifacts
- No working overtime

Automate the Engineering Process:

- Use contemporary platforms

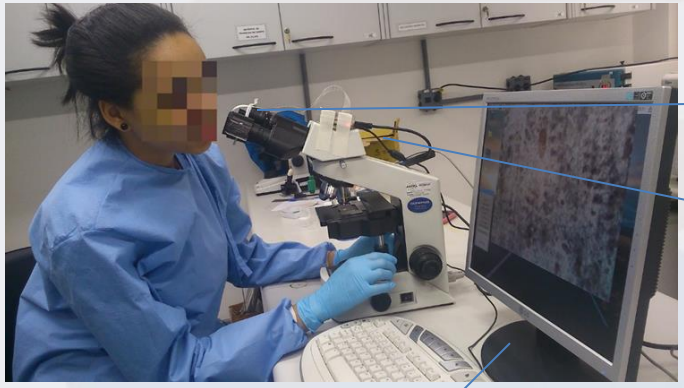
Motivation, Autonomy, Communication, and Short-term projects

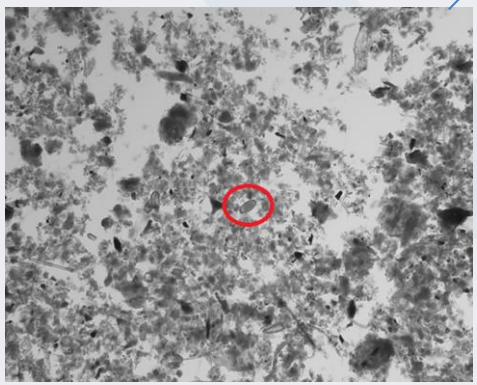
Organize adequate development teams:

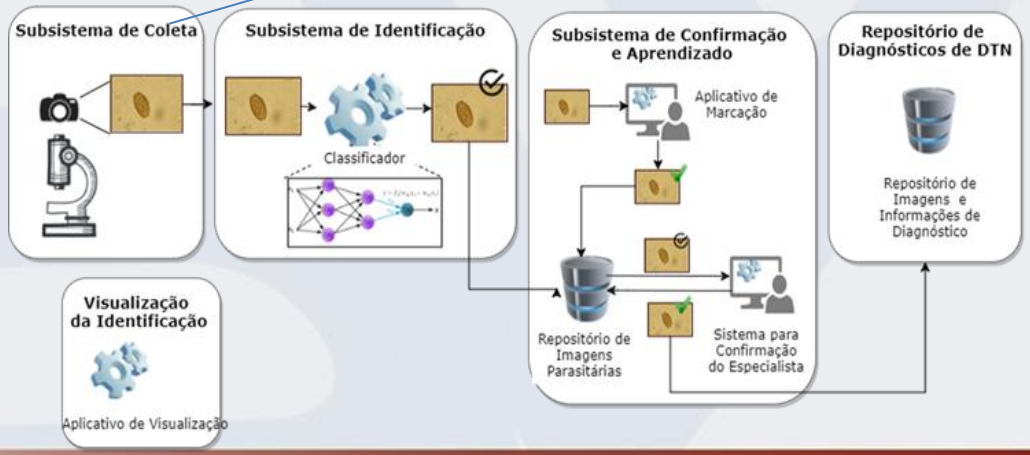
Personnel with high level capacity
Experienced in the engineering process
Domain knowledgeable
Pro-active
Communicative

Materializing CSS, so far…

Parasite Watch

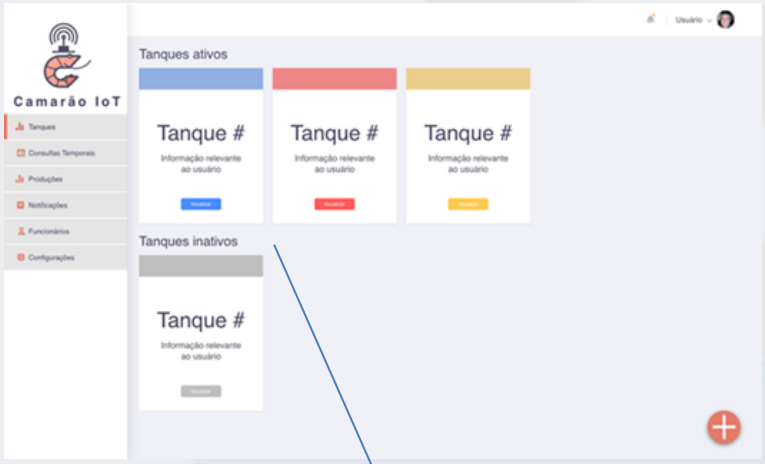


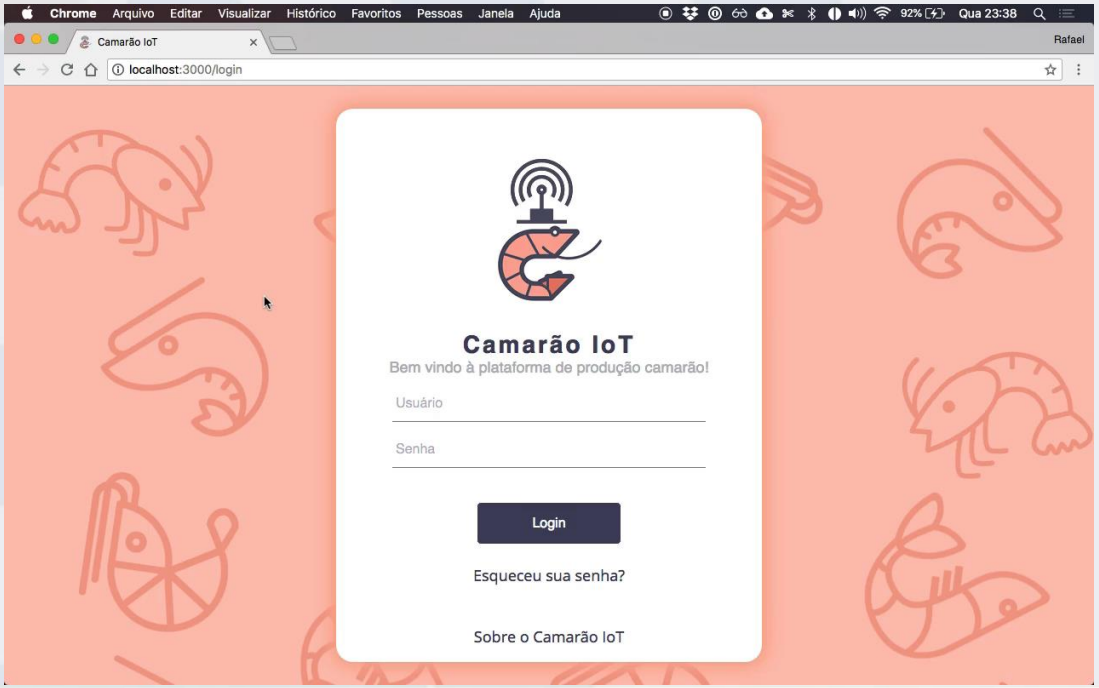




Materializing CSS, so far…

Shrimp IoT





Materializing CSS, so far...

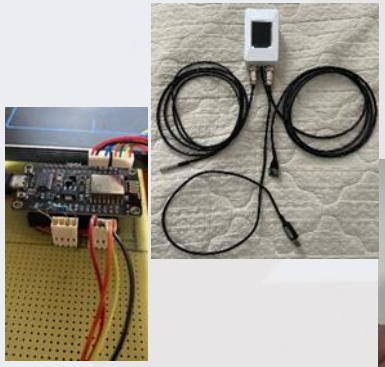
Information Systems

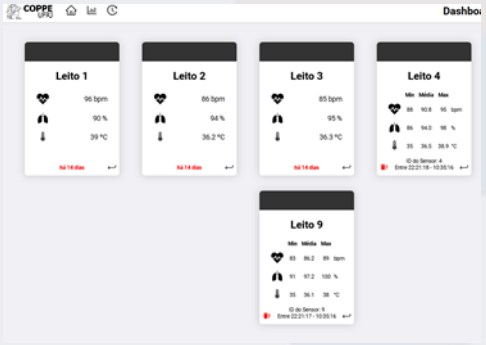
General use Dashboard https://dadoscovid19.cos.ufrj.br/

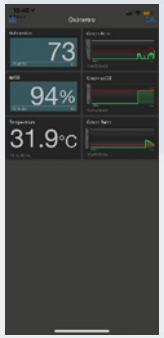
Minerva Bot (soon at coronavirus.ufrj.br)

Materializing CSS, so far…

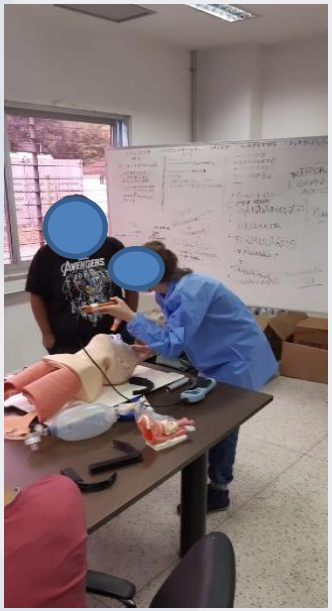
Healthcare Solutions:







Low-cost IoT Oximeter + Dashboard + App



Low-cost IoT Laryngoscope + Camera + App

Facing the CSS Engineering Challenges and pitfalls

Continuous evolution of the Engineering Process:

- Ideation
- Innovation
- Technology Probe
- User Experience
- C
- C
- M
- .

Providing CSS technologies to support development:

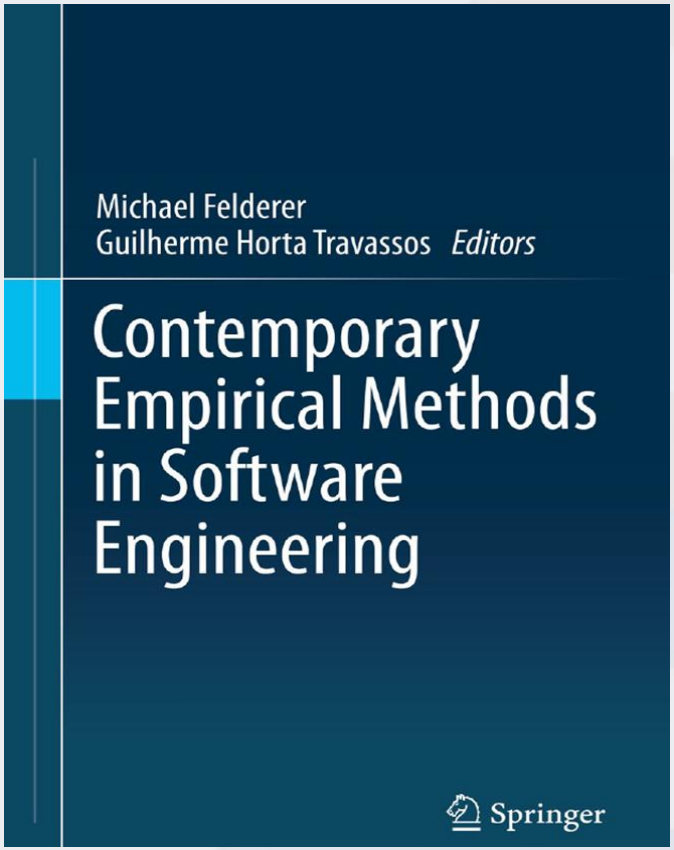
- Context-aware testing environments
- IoT based continuous development
- …

BE EVIDENCE-BASED!

USE CONTEMPORARY EMPIRICAL METHODS TO EVOLVE THE CSS ENGINEERING!

Out of the box thinking!

Experimentation for CSS engineering



Presents contemporary empirical methods in software engineering that will impact future research! Do not miss it…

https://www.springer.com/gp/book/9783030324889

IEEE International Conference on Software Testing
ICST 2021

www.ufrj.br

Thanks
Obrigado
Gracias

Guilherme Horta Travassos
PESC/COPPE
Federal University of Rio de Janeiro
CNPq researcher, ISERN member
ght@cos.ufrj.br
orcid: 0000-0002-4258-0424

COPPE UFRJ

Engenharia de Sistemas e Computação PESC
50 Anos 1970 - 2020