Continuous Certification of Non-Functional Properties Across System Changes

Marco Anisetti **Claudio A. Ardagna** Nicola Bena

firstname.lastname@unimi.it
https://homes.di.unimi.it/*lastname*/

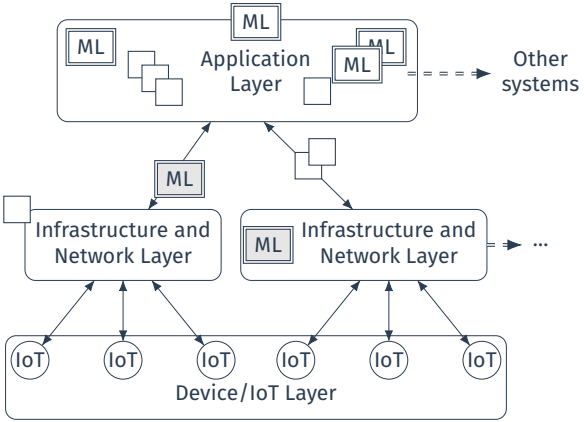
Department of Computer Science, Università degli Studi di Milano, Milan, Italy

The work was partially supported by the projects *i)* MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037); *ii)* SERICS (PE00000014) under the NRRP MUR program funded by the EU – NextGenerationEU.

Scenario

Modern distributed systems

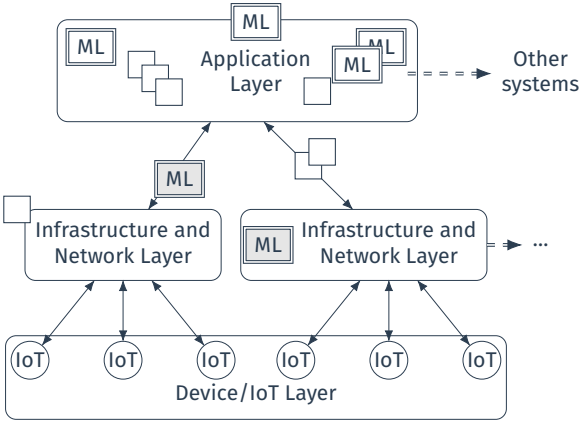
- confluence of cloud-edge-IoT
- multi-layer structure
- ML-based services and infrastructure
- dynamic, non-deterministic, and unpredictable behavior



Scenario

Modern distributed systems

- impact of AI by 2030: $13 trillion (McKinsey)
- ≈15bln of connected devices in 2023: estimated to double by 2030 (Statista)
- economic impact of cloud-edge-IoT by 2025: $2.7–6.2 trillion (McKinsey)

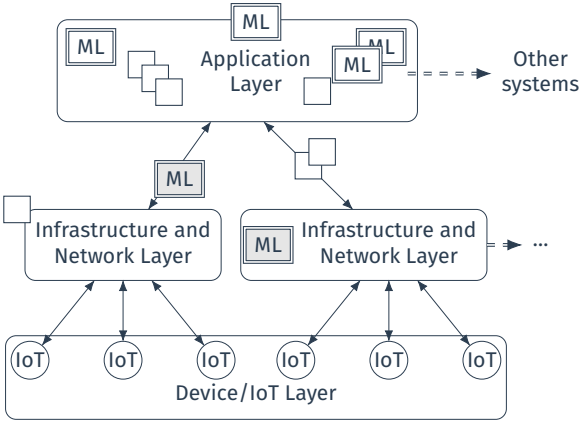


Scenario

Modern distributed systems

- increasing pervasiveness
- increasing risk for security, safety, and privacy
- lack of trustworthiness

$\Longrightarrow$ assurance based-certification to the rescue



Certification

Certification is an assurance procedure by which a third party gives written evidence that a product, process or service shows compliance with certain standards

- Certification techniques provide enough evidence that a system holds some non-functional properties and behaves correctly
- Certification has become widespread in the last 20 years and is also becoming important in today cloud environments
- A priori and run-time techniques

Certification History

Software Certification	Service Certification	Cloud Certification	Modern Systems Certification
• Fully manual • Testing • Focus on run-time target behavior • Continuous (preliminary, later)	• Semi-automatic • Testing • Focus on run-time target behavior • Continuous (preliminary) • Certification-based system life cycle management	• Semi-automatic • Multiple evidence • Focus on run-time target behavior • Continuous • Certification-based system life cycle management • Multi-layer	• Semi-automatic • Multiple evidence • Focus on run-time target behavior • Continuous • Certification-based system life cycle management • Multi-layer • Guidelines for ML

Certification in Europe

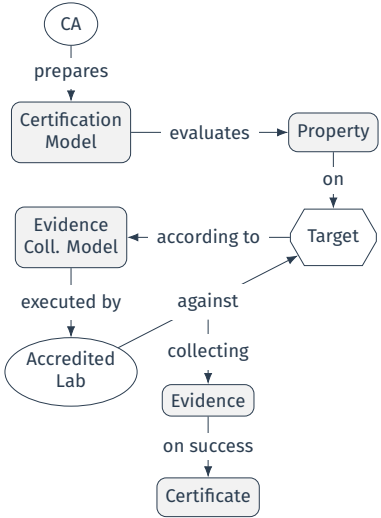
EU Cybersecurity Act

- new mandate for ENISA
  - EU Agency for Cybersecurity with permanent mandate, increased responsibilities and resources
  - setting up and maintain EU cybersecurity certification framework
  - operational cooperation at EU level for management of cyber incidents and large-scale attacks and crises
- European Cybersecurity Certification Framework
  - governance and rules for EU-wide certification of ICT products, processes, services
  - multiple schemes for different domains (*EUCC*, *EUCS*, *EU5G*)

Certification

Certification scheme details the certification process, according to

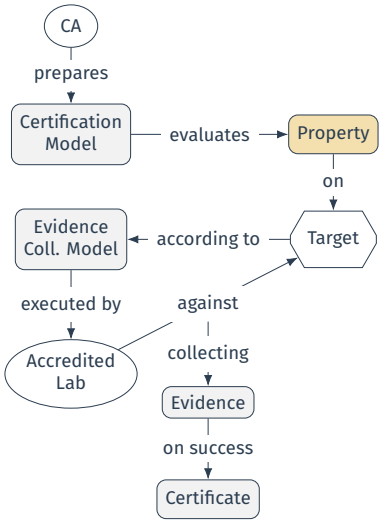
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

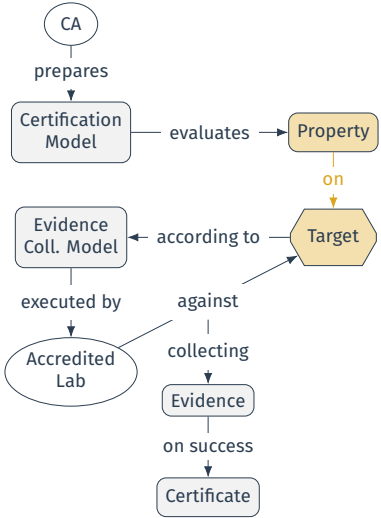
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

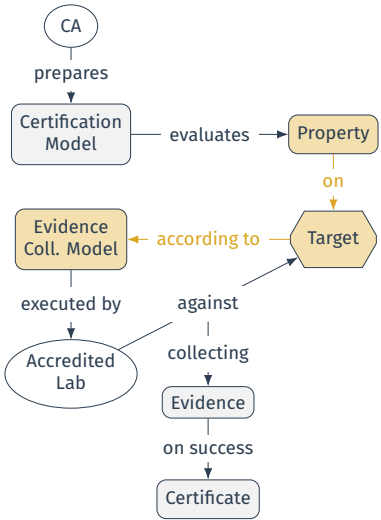
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

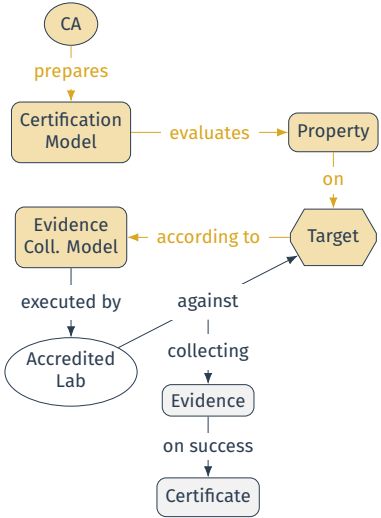
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

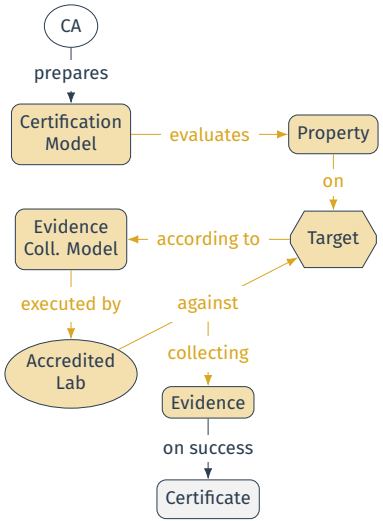
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

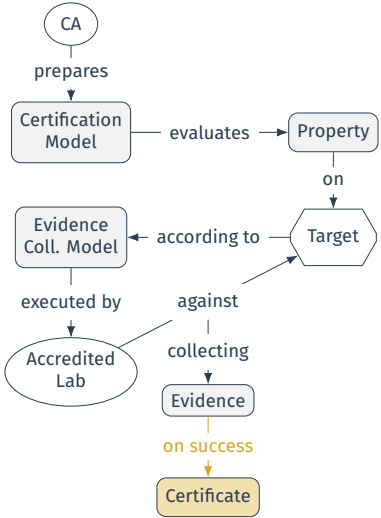
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification

Certification scheme details the certification process, according to

- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certificate Validity (1)

What makes a certificate *valid*?

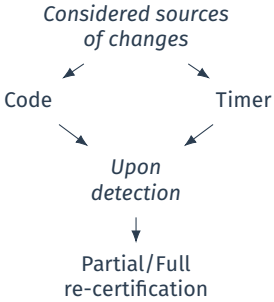
A1) trust: certification model $\mathcal{CM}$ is created by a trusted CA, binding certificates to a chain of trust rooted at it

A2) correctness: $\mathcal{CM}.ToC$ correctly represents the target system components; $\mathcal{CM}.p$ correctly represents the property held by the system

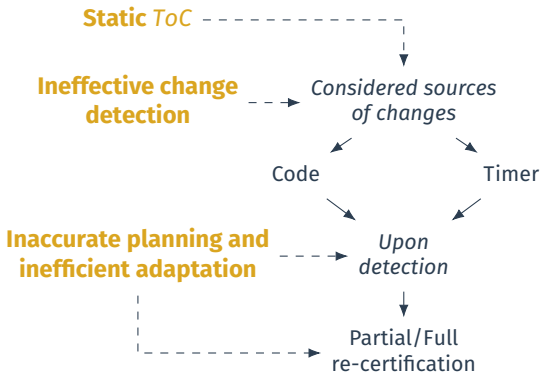
A3) soundness: when evidence is successfully collected according to $\mathcal{CM}$, a certificate $\mathcal{C}$ is awarded proving that $\mathcal{CM}.ToC$ supports $\mathcal{CM}.p$

Can we do this continuously, ensuring that the certificate remains valid across system changes without full re-certification at any changes?

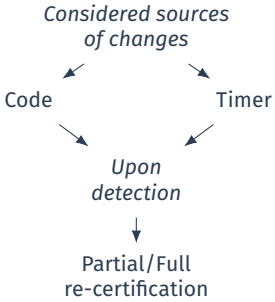
Certificate Validity: State of the Art



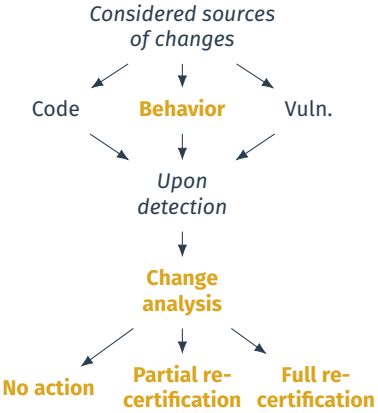
Certificate Validity: State of the Art



Certificate Validity: Our Approach



State of the Art



Our Approach

Our Approach (1)

Certification Model

A certificate model $\mathcal{CM}_t$ at time t is a tuple of the form $\langle p, ToC, \{\{tc_i\}_{c_j}\}, \mathcal{CM}_{t-1}\rangle$ where

- p is the property
- ToC is the target
- $\{\{tc_i\}_{c_j}\}$ is the evidence collection model, with each test case tc_i insisting on a component $c_j \in ToC$
- $\mathcal{CM}_{t-1}$ is a reference to the certification model at time $t-1$ (if any)

Our Approach (2)

Certificate

A certificate $\mathcal{C}_t$ at time t is a tuple of the form $\langle \mathcal{CM}_t, \{ev\}_t, \mathcal{B}, \mathcal{C}_{t-1}, st\rangle$ where

- $\mathcal{CM}_t$ is the certification model
- $\{ev\}_t$ is the collected evidence, including
  - the new evidence $\{\overline{ev}\}$ collected at time t
  - the subset of evidence $\{ev\}_{t-1}$ in certificate $\mathcal{C}_{t-1}$ not superseded by evidence in $\{\overline{ev}\}$
- $\mathcal{C}_{t-1}$ is a reference to the certificate at time $t-1$ (if any)
- st is the certificate status retrieved using function $\texttt{state}\colon \mathcal{C}_t \to \{\text{Valid}, \text{Suspended}, \text{Superseded}, \text{Revoked}\}$

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
  $\implies$ $\mathcal{CM}$ and $\mathcal{C}$ still valid
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
  $\implies$ $\mathcal{CM}$ still represents the system
  $\implies$ new evidence to be collected to ensure $\mathcal{C}$ is still valid
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*

  $\Longrightarrow$ some portions of $\mathcal{CM}$ no longer represent the system

  $\Longrightarrow$ $\mathcal{CM}$ needs to be updated and new evidence re-collected
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

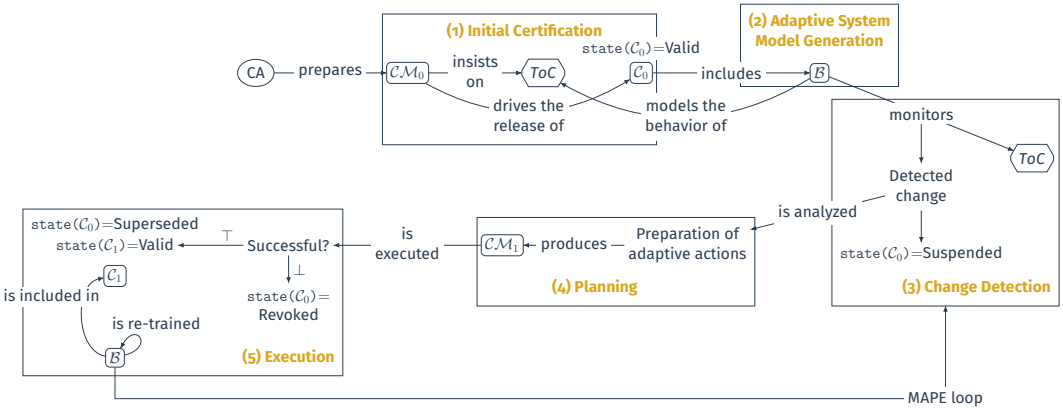
Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

  $\Longrightarrow$ $\mathcal{CM}$ no longer represent the system

  $\Longrightarrow$ a new $\mathcal{CM}$ needs to be defined, and new evidence collected

Our Approach (4)



Reference Example

Example

Microservice managing applicable discounts in an e-commerce application

- service modeled as a set of components: $ToC = \{c_{\text{db}}, c_{\text{api}}\}$
- property *performance*: $p_p = (\hat{p}_p, \{(lang, \text{Java}), (max\text{-}time, 10\text{ms}), (tolerance, 1\text{ms})\})$, with $\hat{p}_p = Performance$
- evidence collection model as a set of test cases insisting on components: $\{\{tc_1\}_{c_{\text{api}}}\}$, where $\{tc_1\}_{c_{\text{api}}}$ consists of one test case sending concurrent requests to the microservice and checking if the response time is compatible with p_p

Reference Example

Example

Microservice managing applicable discounts in an e-commerce application

At time t_0:

1. certification model $\mathcal{CM}_{t_0}$ is created: $\mathcal{CM}_0 = \langle p_p, ToC, \{\{tc_1\}_{c_{\text{api}}}\}, - \rangle$
2. evidence is collected and certificate $\mathcal{C}_{t_0}$ is released
3. $\mathcal{B}$ is created with initial data (ML training using logs, traces, metrics, ...)
4. the continuous certification process starts

Phase Change Detection

Phase change detection continuously monitors *ToC* to retrieve changes for later analysis. Considered changes:

- Behavioral change: anomalies in the system behavior
  - observability data are fed to $\mathcal{B}$ to detect anomalies
- Code change: changes in the code base upon any releases
- Vulnerability change: new vulnerability

When at least one change at time t is detected: $\texttt{state}(\mathcal{C}_t) = \text{Suspended}$

Phase Change Detection: Example

Phase change detection continuously monitors *ToC* to retrieve changes for later analysis

Example

At time t_1, a new version of the service is released, component c_{db} updated to improve its efficiency

- behavior of c_{db} changed
- behavior of c_{api} changed as cascading effect

Output: $\langle \Delta_{\text{b}}{=}(\top, \{c_{\text{api}}, c_{\text{db}}\}), \Delta_{\text{c}}{=}(\top, \{c_{\text{db}}\}, \textit{maj.}), \Delta_{\text{v}}{=}(\bot, \emptyset) \rangle$

- change detected $\Longrightarrow$ $\texttt{state}(\mathcal{C}_{t-1})$=Suspended

Phase Planning (1)

Phase planning analyzes the changes detected at time $t-1$ to retrieve the applicable scenario and the adaptive actions needed to obtain a valid certificate at time t

Input: changes retrieved a time $t-1$ $\langle \Delta_{\mathsf{b}}, \Delta_{\mathsf{c}}, \Delta_{\mathsf{v}} \rangle$

Output: adaptive action $(\mathcal{CM}_t, \mathcal{T})$, where

- $\mathcal{CM}_t$ is a certification model correctly representing the system after changes at time t
- $\mathcal{T} \in \mathcal{CM}_t$ is a subset of test cases in $\mathcal{CM}_t$ to be later executed

Phase Planning (2)

The first set of conditions evaluated to $\top$ determine the applicable scenario

S#	Conditions	Output $(\mathcal{CM}_t, \{tc\})$
S0	– minor code change in non-critical, existing comp.	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1}$ – $\mathcal{T}$=$\emptyset$
S1	– behavioral change (environmental) – behavioral change (minor code change) – major code change without impact on behavior	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1}$ – $\mathcal{T}$=test cases on affected comp.
S2	– vulnerability discovered – behavioral change (major code change) in non-critical, existing comp.	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1} \cup \{tc\}_{\text{new}}$ – $\mathcal{T}$=test cases on affected comp.
S3	– behavioral change (environmental) in critical, existing comp. – major/minor code change in critical, existing comp. – code change adding a new comp.	– new $\mathcal{CM}_t$ – all test cases in $\mathcal{CM}_t$

Phase Planning: Example

Example

Retrieved change:
$\langle \Delta_b = (\top, \{c_{api}, c_{db}\}),$
$\Delta_c = (\top, \{c_{db}\}, \{1\}),$
$\Delta_v = (\bot, \emptyset)\rangle$

Output:

- novel $\mathcal{CM}_{t_1}$ with p_p updated (maximum response time reduced to 7ms)
- $\mathcal{T} = \mathcal{CM}_{t_1}.\{tc_i\}$

S3: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired

Conditions:

- behavioral change in critical, existing components
- code change in critical, existing components (with eventual impact on the behavior)
- code change adding a new component

Output:

- new $\mathcal{CM}_t$
- all test cases in $\mathcal{CM}_t$

Phase Execution

Phase execution executes the adaptive actions $\mathcal{CM}_t, \mathcal{T}$ to award a valid certificate $\mathcal{C}_t$

1. execution of test cases $\mathcal{T}$ to collect new evidence
2. system model re-training
   - re-train the ML model with new data related to the change
   - $\mathcal{B}$ can thus detect behavioral changes in the new version of *ToC*
3. release of certificate $\mathcal{C}_t$
   - new certificate is valid: $\texttt{state}(\mathcal{C}_t) = \text{Valid}$
   - previous certificate is superseded: $\texttt{state}(\mathcal{C}_{t-1}) = \text{Superseded}$
   - previous certificate is revoked, otherwise: $\texttt{state}(\mathcal{C}_{t-1}) = \text{Revoked}$

Phase Execution

Phase execution executes the adaptive actions $\mathcal{CM}_t, \mathcal{T}$ to award a valid certificate $\mathcal{C}_t$

Example

Evidence is successfully collected from *ToC* according to $\mathcal{T}$

- new certificate $\mathcal{C}_{t_1}$ is released and is valid: $\texttt{state}(\mathcal{C}_{t_1})$=Valid, it contains
  - previous evidence in $\mathcal{C}_{t_0}$ not superseded by the new evidence
  - new evidence
- previous certificate is superseded: $\texttt{state}(\mathcal{C}_{t_0}) =$ Superseded

Experiments

We evaluated the quality of our scheme compared to a representative state-of-the-art-continuous certification scheme (SOTA)

- $\mathcal{B}$ is implemented as a set of *isolation forest models*, each isolation forest is trained and detect anomalies on given system component
- more results in our online supplement: `https://doi.org/10.13130/RD_UNIMI/9WXZRC`

Target Systems

We used three distributed systems in literature

- *MS*, 38 microservices for streaming and reviewing movies
- *SN*, 36 microservices for a social network application
- *TT*, 41 microservices for train tickets management

For each distributed system we have the response time of the microservices in normal and anomalous conditions; we mapped

- each distributed system to a *ToC*
- each microservice to a component $c_i \in ToC$
- D_i denotes the dataset of distributed system i

Comparative Evaluation

D_{MS}, D_{SN}, D_{TT} provide information on behavioral changes only

- starting from them, we probabilistically generated datasets D'_{MS}, D'_{SN}, D'_{TT} including additional sources of changes
- we applied our scheme and SOTA on the datasets and measured their quality

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.**–*P4.**

- *P1.**: uniform probabilities for Δ_b, Δ_c, Δ_c with cascading

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P2.**: larger probability for environmental change Δ_b

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	**0.5**	**0.25**	**0.25**	0.75	0.25	0.25	0.25	0.25
P2.2	**0.5**	**0.25**	**0.25**	0.5	0.5	0.5	0.5	0.5
P2.3	**0.5**	**0.25**	**0.25**	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.**–*P4.**

- *P3.**: larger probability for code change Δ_c

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	**0.25**	**0.5**	**0.25**	0.75	0.25	0.25	0.25	0.25
P3.2	**0.25**	**0.5**	**0.25**	0.5	0.5	0.5	0.5	0.5
P3.3	**0.25**	**0.5**	**0.25**	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P4.**: larger probability for code change with impact on behavior Δ_c with cascading

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	**0.25**	**0.25**	**0.5**	0.75	0.25	0.25	0.25	0.25
P4.2	**0.25**	**0.25**	**0.5**	0.5	0.5	0.5	0.5	0.5
P4.3	**0.25**	**0.25**	**0.5**	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P*.1*: major changes with critical impact (i.e., with impact on critical components) on a low number of components

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	**0.75**	0.25	**0.25**	**0.25**	**0.25**
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	**0.75**	0.25	**0.25**	**0.25**	**0.25**
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	**0.75**	0.25	**0.25**	**0.25**	**0.25**
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	**0.75**	0.25	**0.25**	**0.25**	**0.25**
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P*.2*: average scenario balancing minor and major changes, critical and non-critical impact, on a medium number of components

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation: Settings

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P*.3*: minor changes with non-critical impact on a high number of components

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	**0.75**	**0.75**	**0.75**	**0.75**
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	**0.75**	**0.75**	**0.75**	**0.75**
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	**0.75**	**0.75**	**0.75**	**0.75**
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	**0.75**	**0.75**	**0.75**	**0.75**

Comparative Evaluation: Process

Experimental evaluation

1. for each initial dataset in D_{MS}, D_{SN}, D_{TT}, we generated 10 different annotated datasets D'_{MS}, D'_{SN}, D'_{TT}
2. we executed our scheme, detecting behavioral changes according to the isolation forest models
3. we executed a representative state-of-the-art continuous certification scheme
   - partial re-certification at minor code changes
   - full re-certification at major code changes, or any code changes affecting critical components
   - once the change is determined, it retrieves the applicable scenario

Comparative Evaluation: Process

Experimental evaluation

4. we then measured, for each scheme:
   - the ability of detecting all changes ($REC(changes)$)
   - the ability to filter out scenarios where no adaptative action is required ($PREC(no\ action)$)
   - the ability to detect all components involved ($REC(comp)$)
   - the accuracy in retrieving the correct scenario ($ACC(S0)$–$ACC(S3)$)

Comparative Evaluation: Results

Name	REC(changes)		PREC(no action)		REC(comp)		ACC(scenarios)			
	Our	SOTA	Our	SOTA	Our	SOTA	ACC(S0)	ACC(S1)	ACC(S2)	ACC(S3)
P1.1	0.9948	0.6693	0.993	0.8179	0.8622	0.5727	0.8464	0.8716	0.9996	0.8622
P1.2	0.9894	0.6892	0.9882	0.8289	0.8769	0.5205	0.8397	0.8789	1	0.826
P1.3	0.9858	0.6741	0.9846	0.8242	0.8808	0.4215	0.8397	0.8456	0.9995	0.8402
P2.1	0.9958	0.6751	0.9946	0.8153	0.8633	0.5664	0.8445	0.8744	0.9998	0.8579
P2.2	0.9933	0.6738	0.9927	0.8203	0.8718	0.4968	0.844	0.8761	0.9998	0.8133
P2.3	0.9835	0.6645	0.9822	0.8218	0.8857	0.3874	0.835	0.8577	1	0.8072
P3.1	0.9948	0.6733	0.994	0.821	0.8676	0.5838	0.8469	0.9059	0.9998	0.8229
P3.2	0.9912	0.6761	0.9893	0.8184	0.8724	0.5129	0.8434	0.8621	1	0.8355
P3.3	0.9856	0.6726	0.9844	0.8157	0.8808	0.4037	0.8402	0.8549	0.9998	0.8172
P4.1	0.9949	0.6822	0.9947	0.8266	0.8707	0.5914	0.8439	0.8834	0.9998	0.8684
P4.2	0.9911	0.6821	0.9911	0.8373	0.8723	0.5325	0.8373	0.8404	0.9998	0.8802
P4.3	0.9875	0.6943	0.9858	0.8246	0.883	0.4238	0.8434	0.8494	0.9995	0.8391
AVG	**0.9906**	**0.6772**	**0.9895**	**0.8227**	**0.874**	**0.5011**	**0.842**	**0.8667**	**0.9998**	**0.8392**

Conclusions

We proposed a continuous certification scheme supporting certificate validity over time and across system changes

- dynamic certificate lifecycle management built on the ML-based modeling of the system behavior
- fine-grained planning of adaptive actions reducing unnecessary recertification

Future work

- impact of ML techniques for system behavior modeling on continuous certification
- taxonomy of code metrics for continuous certification
- continuous certification of composite service-based systems