Uncertainty-wise Engineering of IoT Cloud Systems: From System Models to Non-Functional Analyses, Deployment, and Testing

Luca Berardinelli,
Hong Linh Truong
Distributed Systems Group, TU Wien

MDE4IoT, Linz, 22/10/2017

https://www.researchgate.net/profile/Luca_Berardinelli
https://www.linkedin.com/in/lucaberardinelli
Outline

1. Introduction
2. IoT Cloud CPS, Uncertainty, and Elasticity
3. Design of IoT Cloud CPS and Uncertainty
4. Deployment of IoT Cloud CPS
5. Testing of IoT Cloud CPS
6. Conclusion and Future Work
Who We Are

Service Engineering Analytics
Research & Development

https://rdsea.github.io/

Performance and dependability

Data concerns
Modeling and Evaluation

- Performance and dependability monitoring and analysis
- Data Concerns, Data Quality Evaluation and Data Contracts for Data Services

Uncertainties
Testing and Analytics

Elasticity
Engineering and Analytics

- Uncertainty Analytics for Cyber-Physical systems, including Runtime Health Verification of CPS
- Service Elasticity Engineering and Analytics

Model-Driven Engineering/Analysis

Hong-Linh Truong
Assistant Professor
Vienna University of Technology
May 2013 – Present  •  4 yrs 3 mos

Luca Berardinelli
Postdoctoral Researcher
Vienna University of Technology
Mar 2015 – Present  •  2 yrs 5 mos
2. What are IoT Cloud CPS

- **Our Cyber-Physical Systems (CPS)**
  - Have IoT elements and cloud services in datacenter, connect via communication
  - Also called IoT Cloud CPS

- **Highly elastic:**
  - Cloud services can be provisioned and de-provisioned
  - IoT devices can be activated, de-activated
  - Communication can be changed by provisioning and de-provisioning resources in an autonomic manner
Key problems

- **Deal with Uncertainties**
  - **Data delivery functional/dependability Uncertainty**, affecting communication resources
  - **Data quality functional/dependability Uncertainty**, e.g. insufficient sampling rate from sensors
  - **Actuation functional/dependability Uncertainty**, affecting mechanisms related to routing, buffering, delivering and ordering of actuation requests

- **Deal with Elastic Execution**:  
  - Elastic tests, mapping uncertainties with elastic execution
Uncertainty Concepts for CPS from H2020 U-Test

- By *uncertainty* we mean here the lack of certainty (i.e., knowledge) about
  - the timing and nature of inputs,
  - the state of a system,
  - a future outcome,
  - as well as other relevant factors.

- If MDE then
  - BeliefStatement → ModelElement
  - BeliefAgent → MDE Tools,…
  - IndeterminacySource → ModelElement, Annotations,…
Design: Uncertainty Modeling and Evaluation (UME)

- **UME:**
  Modeling and Detecting Uncertainty @ Design Time
  Model Refactoring to support next MDE activities (e.g., MBT)

- **Tool for UME (T4UME):**
  Wizards for Modeling, Uncertainty Detection Rules (UDR), UML2JSON

---

reference:

UME: IoT Cloud Profile

...extending UML::Class / UML::InstanceSpecification
UME: Uncertainty Profile

...extending UML::Behavior, UML::StateMachine

Uncertainty Stereotypes

Uncertainty Enumerated Properties

Infrastructure Uncertainty Families
**T4UME** provides Wizards for Infrastructure Modeling *41 wizards for IoT Cloud Profile*)
- stereotype applications
- instantiation of IoT Cloud elements

```javascript
wizard InstantiateManyTimesSensor {
  guard : self.isKindOf(Class) and self.isStereotypeApplied("Sensor")
  title : "Generate Multiple Sensor instances."
  do {
    var plugin : new Native("at.ac.tuwien.dsg.t4ume");
    var ModelerAgent = plugin.getModelerAgent();
    var num_instances : Integer;
    var counter = 1;
    num_instances = UserInput.promptInteger("How many?");
    var package = self.getNearestPackage();
    while (counter <= num_instances) {
      var name = self.getName() + "," + counter;
      var instance = ModelerAgent.createInstanceSpecification(package, self, name);
      instance.attachStereotype("Sensor");
      counter = counter + 1;
    }
  }
}
```
T4UME : Wizards (via Epsilon Wizard Language)

- Extending EMF-based editors (e.g., Papyrus, Rational Software Architect) with Wizards
T4UME: Wizards (via Epsilon Wizard Language)

- Contextual menu entries to call entries on model diagram elements.
Design: Uncertainty Modeling and Evaluation (UME)

- **UME:**
  - Modeling and Detecting Uncertainty @ Design Time
  - Model Refactoring to support next MDE activities (e.g., MBT)

- **Tool for UME (T4UME):**
  - Wizards for Modeling, Uncertainty Detection Rules (UDR), UML2JSON

---

**T4UME : UDR (via Epsilon Validation Language)**

**T4UME** provides **UDRs** for uncertainty detection (U-Detection) on IoT Cloud elements:
- distinct UDR for each stereotype of applied profiles
- warnings for missing property value(s) causing potential uncertainties

```java
context U-CloudService {
    critique cloudProviderProperty {
        check {
            return not self.cloudProvider.isEmpty();
        }
        message: "Missing cloudProvider value"
    }
```

```java
fix {
    <Stereotype> CloudService
        [uml2.extensions
            <Property> cloudProvider : CloudProvider [0..*]
            <Property> serviceType : CloudServiceType [0..*]
            <Property> base_Classifier : Classifier
            <Property> dataProvider : String [0..*]
            <Property> base_InstanceSpecification : InstanceSpecification
            <Property> computingConfigs : ComputingConfiguration [0..*]
            <Property> storageConfigs : StorageConfiguration [0..*]
            <Property> communicationConfigs : CommunicationConfiguration [0..*]
            <Property> base_Class : Class
            <Property> id : Integer [0..*]
        ]
    }
    critique cloudProviderProperty {
        ...
    }
    critique serviceTypeProperty {
        ...
    }
```
T4UME in action: U-Detection by UDR

- Generate DataDeliveryUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate ActuationUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate ElasticityUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate ExecutionEnvironmentUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate StorageUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate UncertaintyFamily State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
- Generate GovernanceUncertainty State in Configurations.UC1.INFR.Units:CentralManagementSystem:CentralManagementSystem_InstanceSpecification1
Modeling and Detecting Uncertainty @ Design Time
Model Refactoring to support next MDE activities (e.g., MBT)
T4UME provides **UDRs** for uncertainty detection (U-Detection) on IoT Cloud elements
- distinct UDR for each stereotype of applied profiles
- warnings for missing property value(s) causing potential uncertainties
T4UME in action: U-Refactoring by UDR
Modeling and Detecting Uncertainty @ Design Time
Model Refactoring to support next MDE activities (e.g., MBT)
T4UME: Wizard and UDR Generation (via Epsilon Generation Language)

UME adapts to different domains (modeled as UML profiles)
T4UME automatically generates wizards and UDRs from applied profiles

```java
rule Stereotype2UDR
  transform stereo : Stereotype {
    template : "UDR_Template.egl"
    target : stereo.getName() + ".UDR.evl"
  }

context [%stereo.name%] {
  [%for (attribute in stereo.AllAttributes()) %]
  critique [%attribute.name%].Specification {
    check: not self.[%attribute.name%].isEmpty()
    message: "Missing" + [%attribute.name%] + "value"
  }
  [%}
  var uncertainty_families = stereo.getInfrastructureUProfile().getUTaxonomy();
  for (family in uncertainty_families) {
    [%}
    fix {
      title : "Generate"+ [%family.name%] + "State"
      do {
        var baseElement = UMLUtil.getBaseElement(self);
        baseElement.create [%family.name%].StateMachine();
        ...
      }
    }
  [%]}
```
Design: Uncertainty Modeling and Evaluation (UME)
**T4UME in action: UML2JSON via Java+GSON**

**Input:** IoT Cloud CPS UML Model + Profiles (IoT Cloud, MARTE,...)

**Intermediate:**
IoT Cloud CPS JAVA Main Profile-specific APIs + GSON Lib

```java
import com.google.gson.Gson;
public class VirtualSensor {
    @SerializedName("instanceof")
    private String _instanceof = new String("Undefined");
    @SerializedName("swCapabilities")
    private List<String> _swCapabilities = new ArrayList<String>();
    public String __type = new String("VirtualSensor");
```

**Output:** IoT Cloud System JSON Model

```
"ElectricitySensor1": {
    "name": "ElectricitySensor",
    "swCapabilities": ["setRate", "getRate"],
    "ownedUnits": ["HwElectricitySensor"],
    "type": "VirtualSensor"
}
```
Deployment: Work flow

- Reuse well-known tools for deployment (e.g. SALSA)
- Adapt extracted JSON for many tools → UML2JSON flexible
- Uncertainty info has to be propagated (on going work)

source
http://tuwiendsg.github.io/SALSA/
Deployment: Example of artifacts

Listing 5: Extracted simplified example of a Docker-compose deployment description

version: '3'
services:
ingest:
    build: .
volumes:
    - ./t4u
electricitysensor:
    image: "localhost:5000/t4u/mqttserver/
        realsensor:v01"
iotgateway:
    image: "localhost:5000/t4u/cloudservice/
        mqttbroker:v01"

Listing 6: Example of TOSCA deployment description

<ns2:NodeTemplate maxInstances="10"
    id="electricitysensor" type="salsa:os">
    <ns2:Properties>
        <MappingProperties>
            <MappingProperty type="salsa:os">
                <property name="provider"></property>
                <property name="instanceType"></property>
                <property name="baseImage"></property>
            </MappingProperty>
        </MappingProperties>
    </ns2:Properties>
</ns2:NodeTemplate>

Listing 3: Example of storing artifacts and metadata

$docker tag mqttsensor localhost:5000/t4u/
    clouIdservice/mqttbroker:v01
$docker push localhost:5000/t4u/cloudservice/
    mqttbroker
$t4u_metadata add localhost:5000/t4u/cloudservice/
    mqttbroker:v0

Reference:
Hong-Linh Truong, Luca Berardinelli, Ivan Pavkovic and Georgiana Copil, 
Modeling and Provisioning IoT Cloud Systems for Testing Uncertainties 
14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, 
Networking and Services (MobiQuitous 2017), November 7–10, 2017, Melbourne, 
Australia. To appear.
Testing Work flow

Problem

- MBT approaches do not consider IoT Cloud Infrastructures underlying the CPS
- Static SUT deployment

Solution:

- MBT process that deal with **dynamic configuration and elastic execution** of Cloud and IoT resources
Interwoven Testing, Provisioning and Modeling under Uncertainty
Conclusions and Future Work

Conclusions:

- We are devising methodology (UME) and tool (T4UME) for uncertainty modeling and evaluation at design-time
  - Wizards apply and instantiate IoT Cloud architectural elements
  - U-Detection to detect uncertainty caused my missing property values of stereotypes
  - U-Refactoring actions implemented ad-hoc to support MBT (test case generation from state machines)
  - UML2JSON exports UML model content into JSON via Java objects and GSON
Conclusions and Future Work

Future Work:

- Customization of UME/T4UME for different MDE tasks
  - Integration of OMG standard profiles (MARTE, SysML) thanks to Wizard and UDR generation capability (on going)
  - Performance Uncertainty caused by detected Performance Antipatterns (on going) via customized U-Detection and U-Refactoring steps
  - UDR composition algorithms
  - Mappings of stereotype properties with Uncertainty Families (e.g., no MARTE::exec_time for operations then potential Performance Uncertainty)
  - Customization for different application domains
  - Extension for non-UML based approaches (e.g., UDR from metaclasses)
Thank you! Q&A

**Service Engineering Analytics**
**Research & Development**

[https://rdsea.github.io/](https://rdsea.github.io/)

---

**Performance and dependability**

- Performance and dependability monitoring and analysis

---

**Uncertainties Testing and Analytics**

- Uncertainty Analytics for Cyber-Physical systems, including Runtime Health Verification of CPS

---

**Data concerns**

Data Modeling and Evaluation

- Data Concerns, Data Quality Evaluation and Data Contracts for Data Services

---

**Elasticity Engineering and Analytics**

- Service Elasticity Engineering and Analytics

---

Model-Driven Engineering/Analysis

---

Hong-Linh Truong

Assistant Professor
Vienna University of Technology
May 2013 – Present • 4 yrs 3 mos

Luca Berardinelli

Postdoctoral Researcher
Vienna University of Technology
Mar 2015 – Present • 2 yrs 5 mos

Austria