Validation of a Security Metamodel for Development of Cloud Applications

Marcos Arjona - Carolina Dania - Marina Egea - Antonio Maña

14th International Workshop on OCL and Textual Modeling Applications and Case Studies (OCL 2014)

September 30, 2014

Motivation - The problem

- Development of secure applications is a challenging task due to multiple security concerns and risks that threaten any system under design.
- Development of secure applications for Cloud environments need a stronger and reliable approach to address domain-specific security requirements along with complex context aware assurance mechanisms.

Motivation - The solution I

- Proposed approaches agree in the necessity to sit a solid and affordable engineering process that can prevent, at design time, non-secure states.
- Shared assurance requirements:
  - **Dynamism**: Evolving systems and knowledge.
  - **Composition**: Both vertical (layers) or horizontal (components).
  - **Complexity**: Embedded or Cloud Systems, Cyber Physical Systems, System of Systems, etc.

Motivation - The solution II

- Our work stems in the definition and evaluation of a Model Based Secure System Engineering (MBSE) methodology for the CUMULUS and PARIS EU projects.

- The Cumulus Engineering Process(CEP) aims to orchestrate an automated sorting and processing of cloud security knowledge to make it accessible and useful for:
  - **Security Experts**: Verify, update or improve security models.
  - **Cloud Software Developers** (non security experts): Deploy trusted security mechanisms in systems.

"Security Knowledge Transfer"

The Cumulus engineering process I

The CEP addresses:

- **Composition**
  - Security and Privacy by design.
  - Local Assurance: Security Patterns to describe software realization.
- **Complexity**
  - Certification: Service Assurance Profiles to request cloud platforms for certified services.
  - TPM system attestation.
- **Dynamism**
  - Static and Dynamic evidence generation: Certification models with static or dynamic-based monitoring.
- **C & C & D**
  - Machine-processable and upgradeable artifacts.

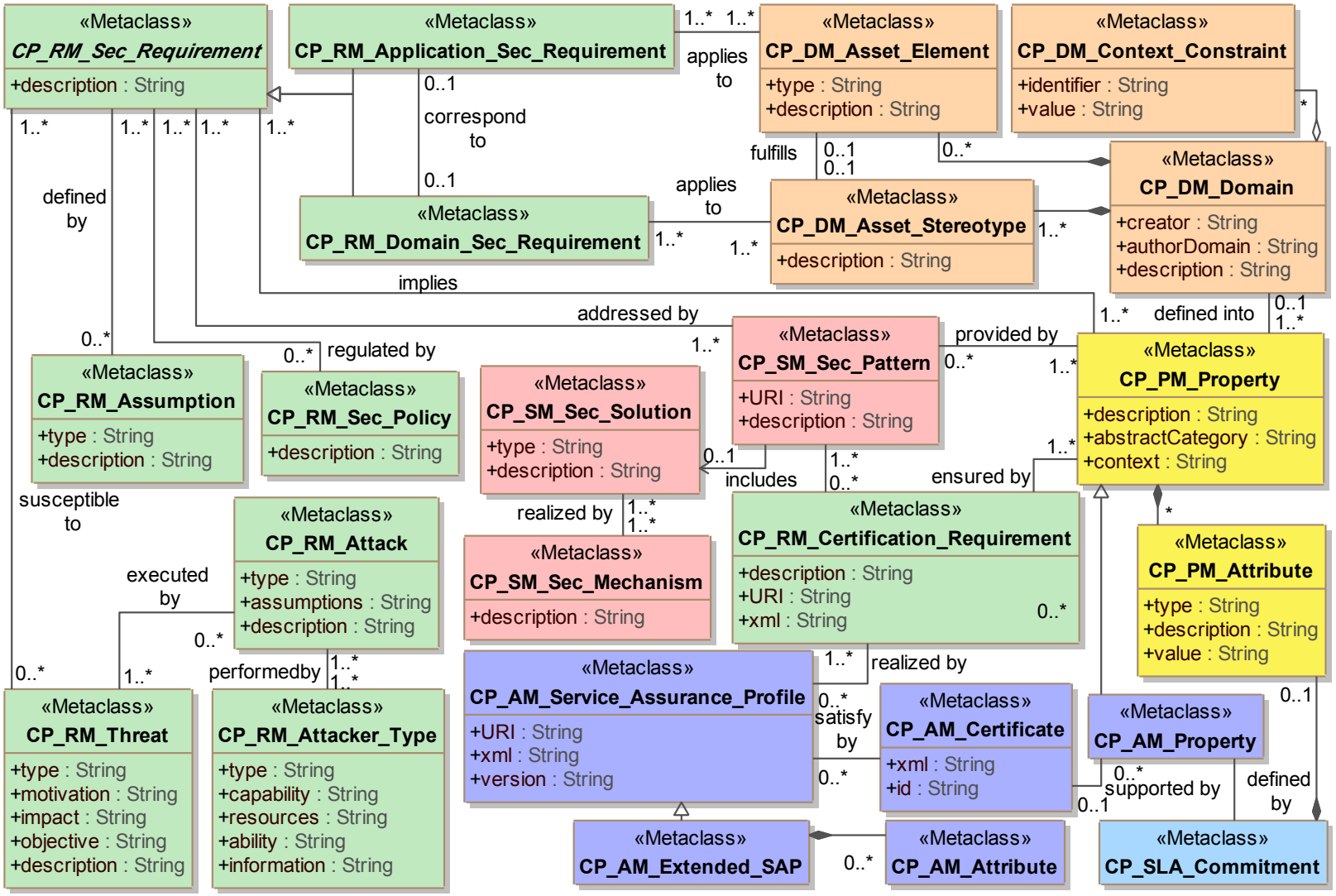
The Cumulus engineering process II

The main artifact of the CEP is the **Core Security Metamodel** (CSM), a metamodel to describe security knowledge for the development of secure cloud applications.

The CSM defines a language to drive the instantiation and express an adequate structure to represent security knowledge.

The effectiveness of this approach relies on the OCL validation system of MagicDraw that is incrementally triggered by the tool.

The Core Security Metamodel



The Core Security Metamodel (constraints)

1. A domain instance must exist and be unique.
   inv: `CP_DM_Domain.allInstances()->size() = 1`
2. A certification requirement needs to be associated with a service assurance profile.
   context: `CP_RM_Certification_Requirement`
   inv: `(not self.URI.oclIsUndefined()) implies self.service_assurance_profile->notEmpty()`
3. A certification requirement should be directly linked to a property and a security pattern for that property.
   context: `CP_RM_Certification_Requirement`
   inv: `self.property->intersection(self.sec_pattern.property)->notEmpty()`

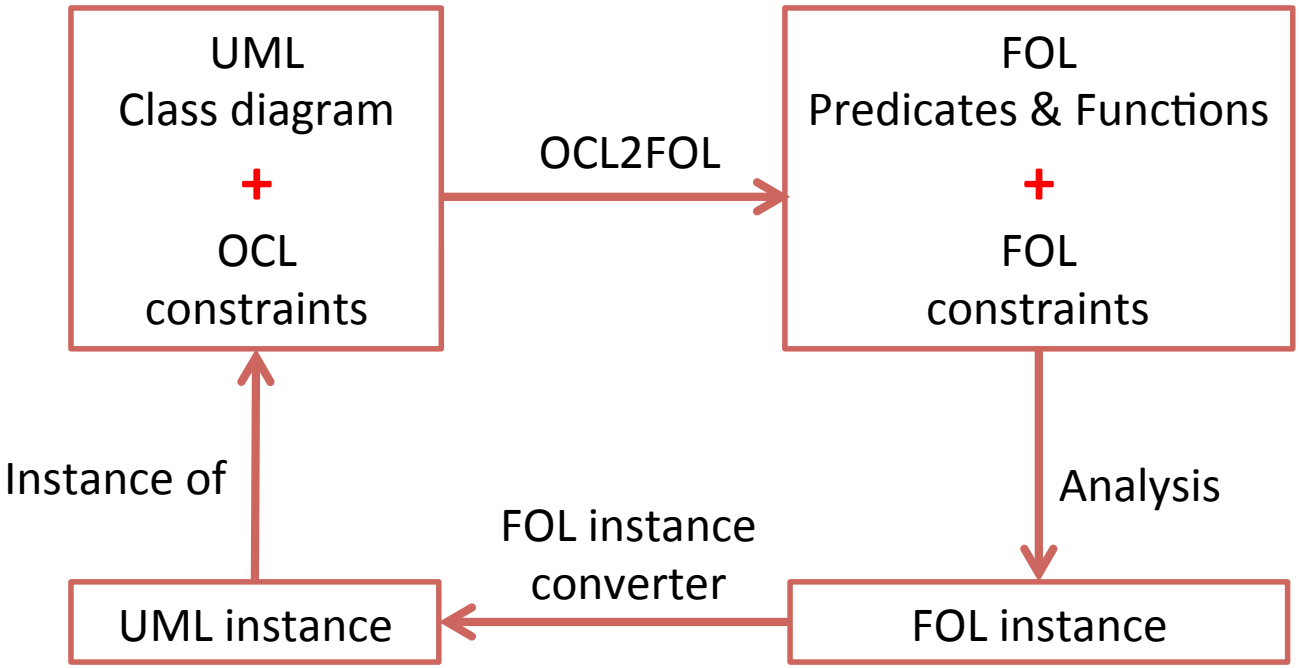
Security Modeling asissted process (supported by CASE tools)

The OCL validation system supports 3 goals in the CSM instantiation activity

1. Perform an active validation of the modeling process:
   - it raises a warning if the instance does not conform to the meta-model
   - it points out the pieces of information that are missing/wrong
2. Check that required information is present:
   - it validates whether a valid CSM instance lacks information that is required by the engineering activities. E.g., transitive association between specific components, empty attributes, etc..
3. Guide experts during the creation of the CSM instance
   - towards the next piece of information that is required and its goal in the engineering process

OCL2FOL

It is a mapping from OCL to First Order Logic which supports OCL 4 values.



CSM metamodel-formal analysis

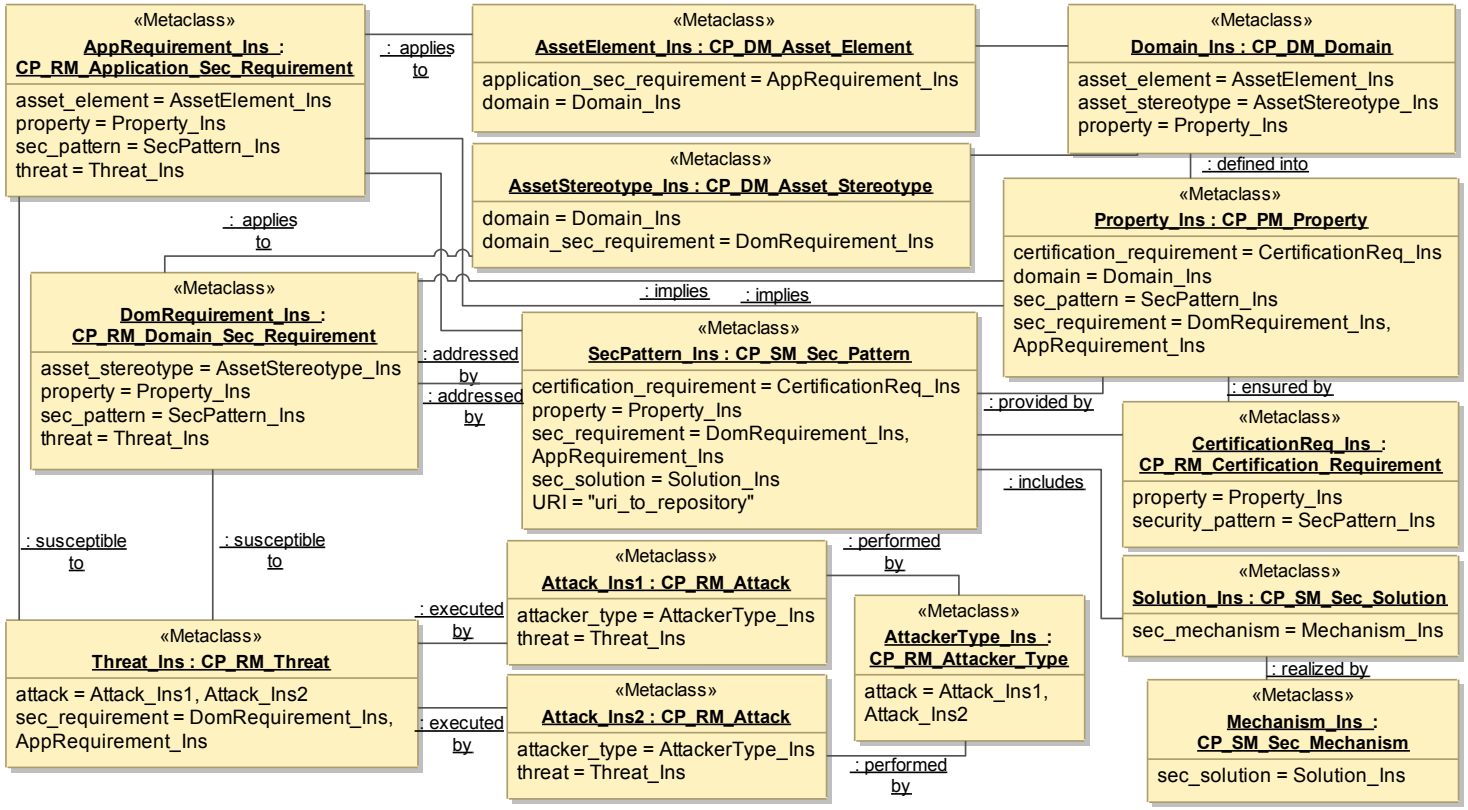
- We use CVC4 as a finite model finder to:
  1. check if there exists an instance of CSM which satisfy all invariants,
  2. and if it does, CVC4 generates automatically one of such instances: indeed, CVC4 returned sat in less than 30 seconds and a simple instance

- We also tried with Z3 and CVC4 as SMT solver, but they did not return an answer about the unsatisfiability of the spec, i.e., whether it was ‘unsat’ or ‘sat’ (incompleteness reasons?)

Instance of CSM

We required the instance to contain:

- CP_RM_Attack.allInstances()->size()=2;
- Similarly, 1 instance of CP_RM_Attack_Type, CP_SM_Sec_Solution, CP_SM_Sec_Mechanism, and CP_RM_Certification_Requirement.



Security enhanced CSM instances

As a result of the CVC4 process, the auto generated CSM instance makes an advantaged starting point for the gathering of security knowledge offering:

- **A valid model**: The current instance is valid and processable by the CEP modeling framework.
- **An optimized approach**: Security Experts receives a shorcut to avoid the instance creation and validation, reducing time and effort.
- **A customized instance**: Security Experts can enforce specific demands to the solvers, obtaining suitable instances for their desired configurations.

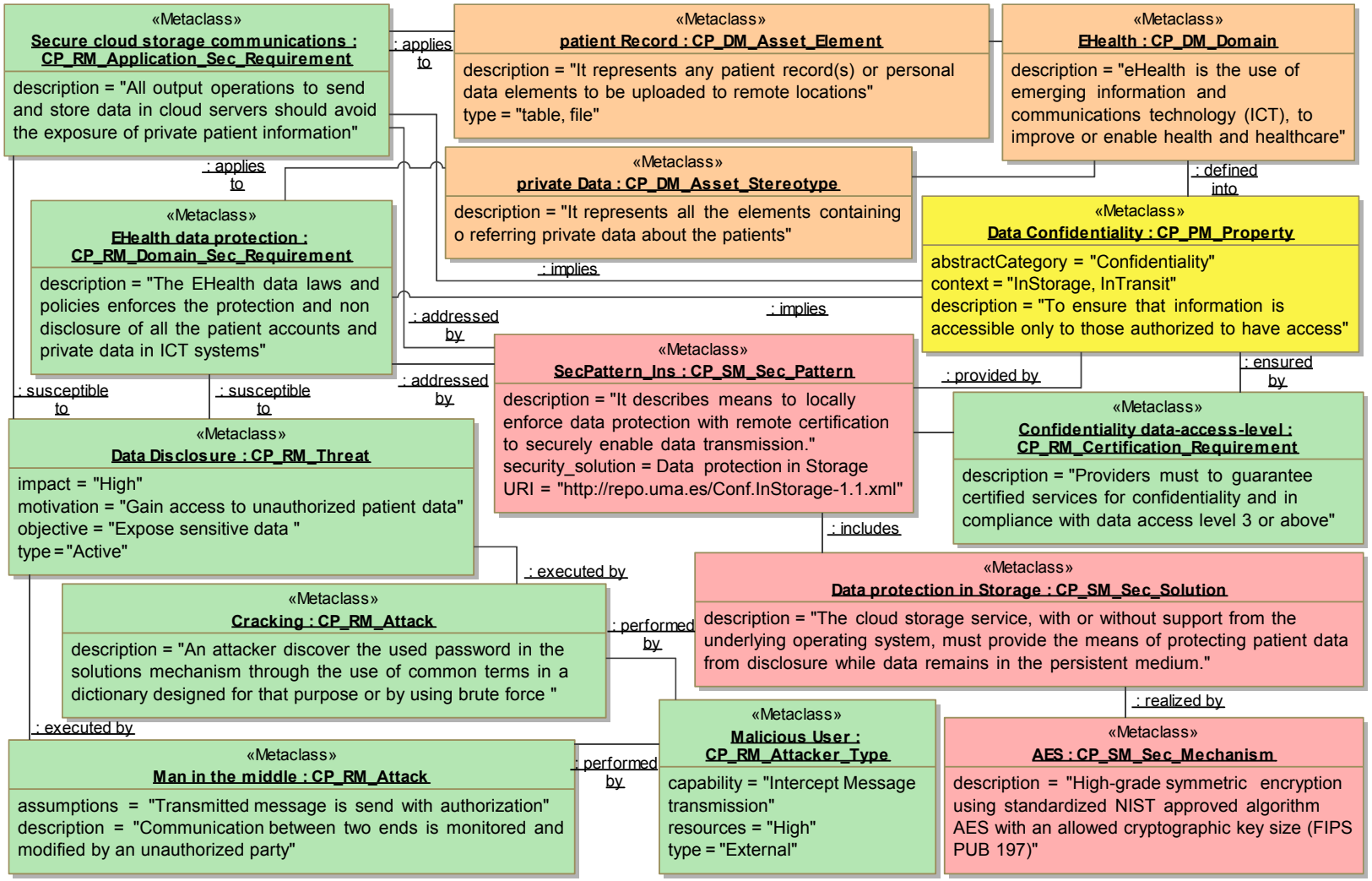
Security enhanced CSM instance

Security Experts loads the generated instance in the CUMULUS framework and starts to incorporate security knowledge.

The new CSM instance does not collide with the additional framework interactions of the CEP, which they have to be incorporated to complete all the security enhancements.

- Retrieve Security Patterns
- Select adequate Certification Requirements
- Choose most suitable Service Assurance Profiles

Security enhanced CSM instance



Conclusions

- We discussed the complexity of secure cloud applications assurance and the neccessity of high level methodologies, e.g., CEP.
- We introduced a security metamodel (CSM) which drives the engineering of secure cloud applications and formally analysed it
- We employed CVC4 as finite model finder to generate instances automatically.
- We summarized the benefits to start the representation of security knowledge at this auto-generated instance.
- Thus, we claim that formal analysis provides higher assurance of the adequacy of CSM+rules, reduces time and effort to security experts in the initial stage of the engineering process by automatic instance generation

Future Work

- Implement a converter from the CVC4 instances to a valid model input format for MagicDraw to automate the process based on instance generation.
- Expand and improve the collection of OCL rules based on discovered enhancements during the CSM analysis
- Replicate and perform a new analysis to the CSM version 2.0 recently released in the CUMULUS project.
- Yet, we note that the instance needs to be enhanced with security domain specific knowledge to trigger subsequent steps

Thank you!

See it in action!

http://proteus.lcc.uma.es/proyectos/secfutur/

Using OCL2FOL to map CSM into First Order Logic

```
CP_RM_Certification_Requirement.allInstances()
->forAll(c|not(c.URI.oclIsUndefined()) implies (not
s.service_assurance_profile->notEmpty()))
```

$$\forall(x)(\mathrm{CPRMCertificationRequirement}(x) \\ \wedge \neg(\mathrm{isNull}(\mathrm{CPRMCRurl}(x)) \vee \mathrm{isInvalid}(x)) \\ \Rightarrow \exists(y)(\mathrm{CPAMServiceAssuranceProfile}(y) \wedge \mathrm{CPRMCRrealizedby}(y, x)))$$

- classes are mapped as predicates, attributes as functions, and association-ends as binary predicates or functions (depending of multiplicity). Also, a set of constraints are added.
- there are two predicates isNull and isInvalid which evaluates true when the value is null or invalid respectively.