Towards Semantic Detection of Smells in Cloud Infrastructure Code

Indika Kumara, Zoe Vasileiou, Georgios Meditskos, Damian A. Tamburri, Willem-Jan Van Den Heuvel, Anastasios Karakostas, Stefanos Vrochidis, Ioannis Kompatsiaris

03.07.2020

WIMS 2020, June 30-July 3, 2020, Biarritz, France

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825480.

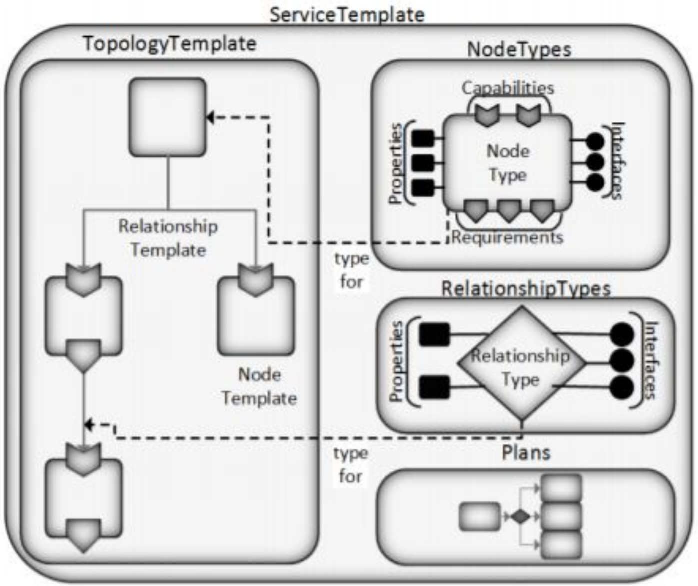
Infrastructure as Code

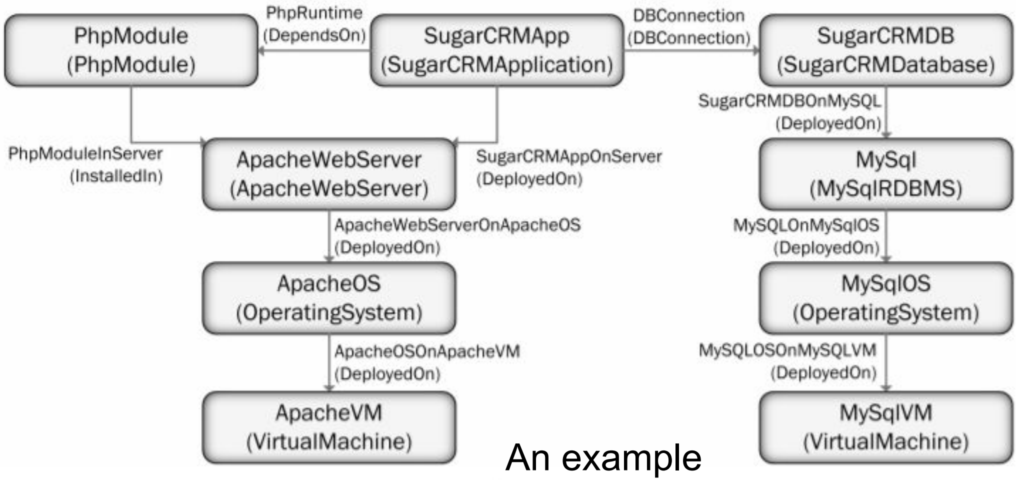
“Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.” ~ Wikipedia

Topology and Orchestration Specification for Cloud Applications

Infrastructure as Code: TOSCA

Sodalite





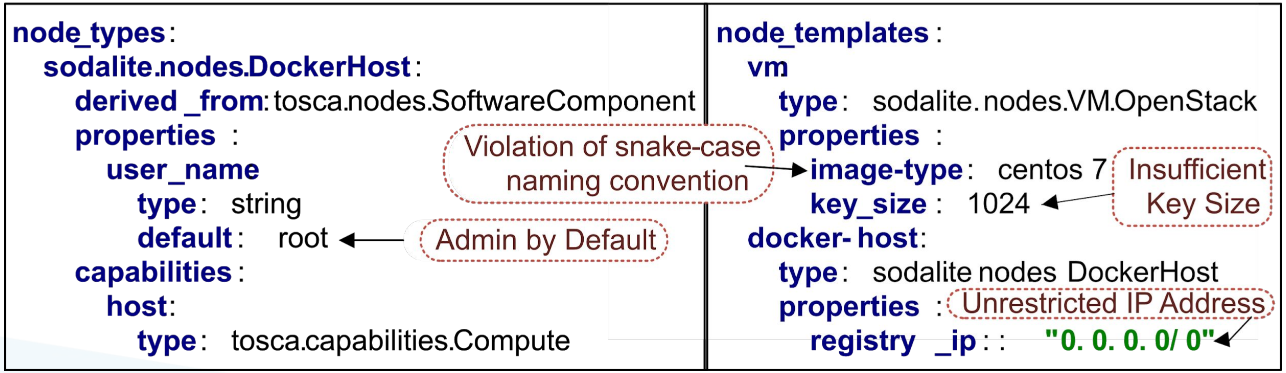
An example

TOSCA (open standard) is IaC technology agnostic, application domain agnostic, and infrastructure provider agnostic.

Problem: Smells in Infrastructure as Code

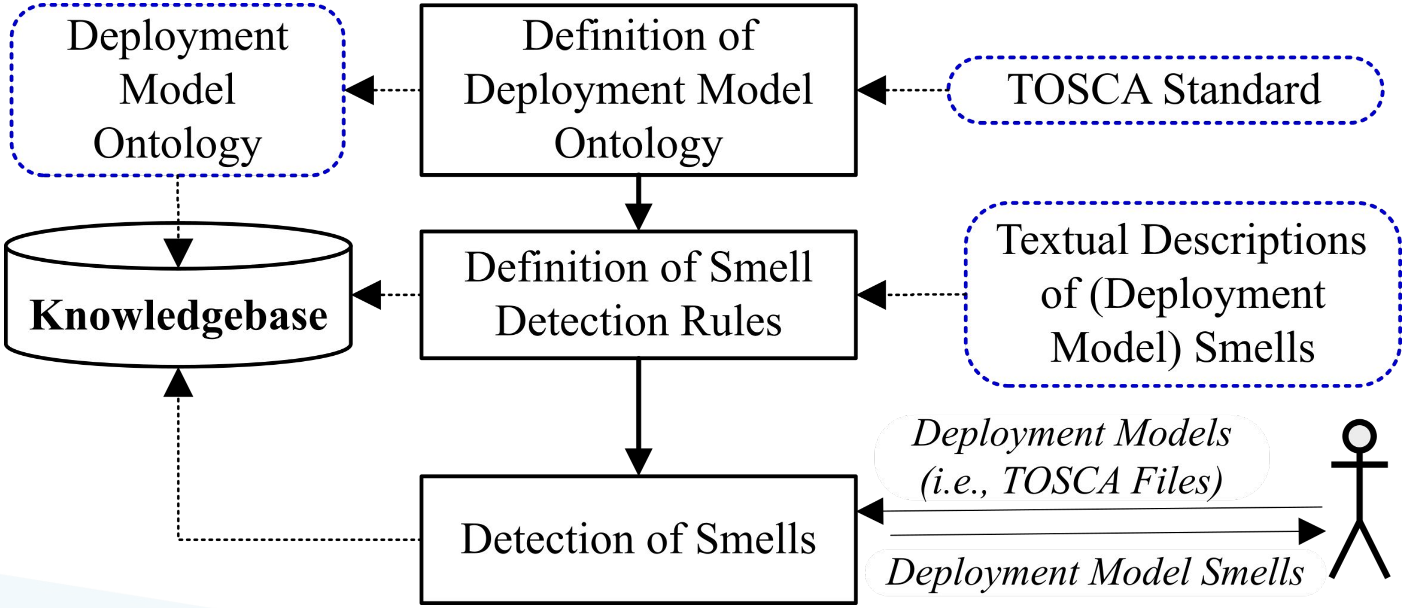
Sodalite

- A software smell is any characteristic in the artifacts of the software that possibly indicates a deeper problem or quality issue.



Our Approach to Detect Smells

Sodalite

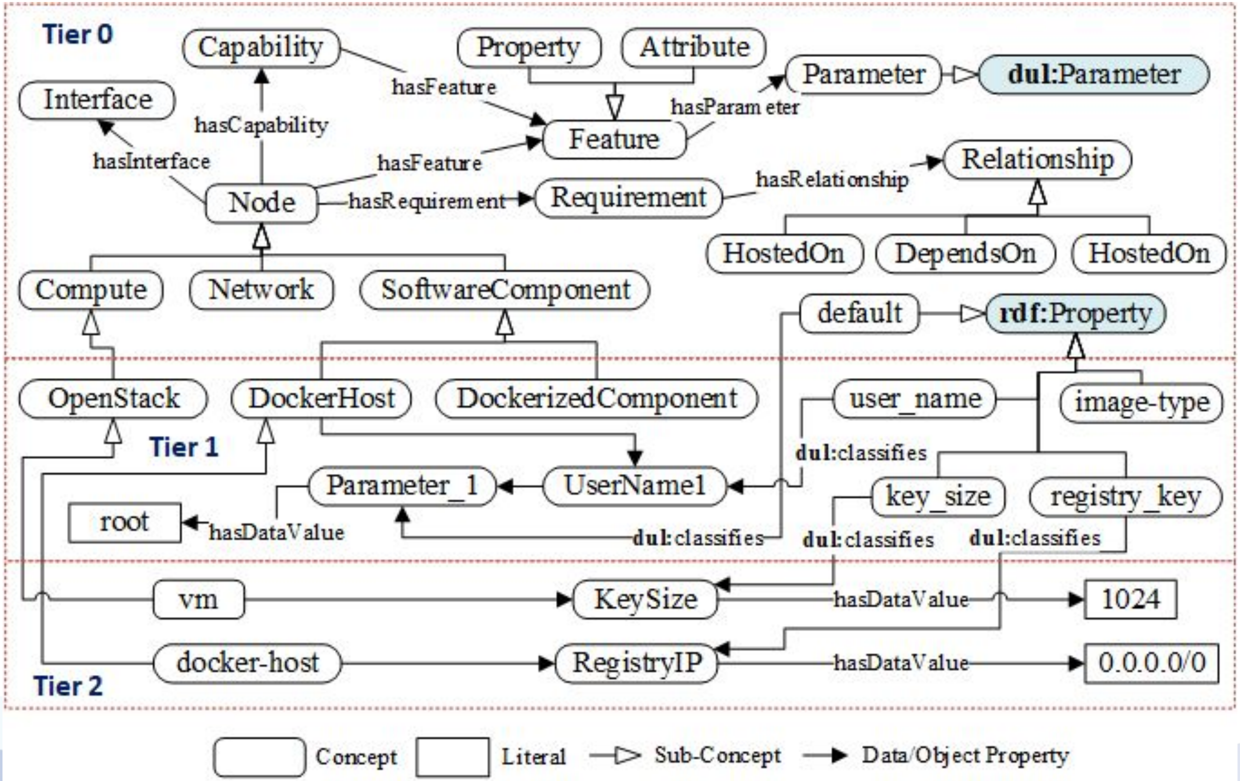


Deployment Model Ontology

Sodalite

- Align with multi-level modeling of TOSCA
- Grounded on ontology design patterns

https://github.com/SODALITE-EU/semantic-models



Smell Detection Rules

Sodalite

We primarily use SPARQL queries for specifying detection rules.

An Example: a rule for detecting admin by default.

```
select distinct ?property ?propertyDef
where {
    ?property DUL:classifies ?propertyDef.
    FILTER(regex(str(?propertyDef),"user(.+?)|(.+?)?user","i")).
    optional { # node type definitions - tier1
          ?property DUL:hasParameter ?p .
          ?p DUL:classifies tosca:default .
          ?p tosca:hasDataValue ?value.
       }.
    optional { # node template definitions - tier0
         ?property tosca:hasDataValue ?value.
       }.
    FILTER (bound(?value)).
    FILTER (str(?value) IN ('admin', 'root'))
}
```

All rules are at https://github.com/SODALITE-EU/defect-prediction

Smell Detection Rules (Cont'd)

Sodalite

Smell	Smell Description	Abstract Detection Rule
Admin by default	Default users are administrative users.	isUser (x.name) $\wedge isAdmin(x.name)$
Empty password	A password as a zero-length string.	isPassword(x.name) $\wedge(isEmpty(x.value) \vee isEmpty(x.defaultValue))$
Hard-coded secret	Secrets such as usernames and passwords are hardcoded.	(isPassword(x.name) $\vee isUser(x.name) \vee isSecKey(x.name)) \wedge$ ((~isEmpty(x.value) ∧~isVariable(x.value)) ∨~isEmpty(x.defaultValue))
Suspicious comment	A comment includes the information indicating secrets and buggy implementations.	hasComment(x) $\wedge isSuspicious(x.comment)$
Unrestricted IP address	Using "0.0.0.0" or "::" as binding IP addresses of servers	isIP(x.name) $\wedge(isInvalidBind(x.value) \vee isInvalidBind(x.defaultValue))$
Insecure communication	Using insecure communication protocols, instead of their secure counterparts	(isURL(x.value) $\wedge isInsecure(x.value)) \vee$ (isURL(x.defaultValue) $\wedge isInsecure(x.defaultValue))$
Weak crypto. algo.	Use of weak cryptography algorithms such as MD5 and SHA1	hasWeakAlgo(x.value) $\vee hasWeakAlgo(x.defaultValue)$
Insufficient key Size	The size of a key used by an encryption algorithm is less than the recommended key size, e.g., 2048 bits for RSA.	isCryptoKeySize(x.name) $\wedge(hasInsufficientKeySize(x.value)$ $\vee hasInsufficientKeySize(x.defaultValue))$
Inconsistent naming convention	The conventions used for naming nodes, properties, attributes, etc., are inconsistent.	(case=='CamelCase' $\rightarrow isCamelCase(x.name)) \vee (case ==' SnakeCase'$ $\rightarrow isSnakeCase(x.name)) \vee (case ==' DashCase' \rightarrow isDashCase(x.name))$
Invalid port ranges	TCP port values are not within the range from 0 to 65535.	isPort(x.name) $\wedge(outOfRange(x.value) \vee outOfRange(x.defaultValue))$

All SPARQL rules are at
https://github.com/SODALITE-EU/defect-prediction

Prototype and Evaluation

Sodalite

Validated with three industrial case studies of SODALITE H2020 project

1. In-silico clinical trials for spinal operations
2. Vehicle IoT Use Case
3. Water availability prediction with mountains images

Demo - https://www.youtube.com/watch?v=IThr5vlleTI

Conclusion and Future Work

Sodalite

- We presented an approach that can formally model a cloud application deployment model with ontologies, and detect the smells in the model with ontological reasoning.
- Ongoing works
  - Extend the semantic models to specify smells, their causes, and their fixes.
  - Extend the rule-base to cover all smells identified by a systematic literature review on infrastructure code smells.

Conclusion and Future Work

Sodalite

- Ongoing works
  - Build a unified framework to detect smells across heterogeneous deployment and infrastructure code specifications by utilizing semantic Web techniques such as ontology mapping, alignment, and query rewriting

Questions

Sodalite

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 825480.