OntoDSL

An Ontology-Based Framework for Domain-Specific Languages

Tobias Walter
Fernando Silva Parreiras
Steffen Staab

08.10.2009
Objectives

Motivation

- Scenario
- Requirements

Description Logics

Ontology-based DSL Frameworks

- Design DSLs
- Use DSLs

Conclusion
Scenario (Roles)

Framework Developer

DSL Designer

DSL User

Metamodelling Language

DSL Metamodel

Domain Model

Constraints based on defined services

Guidance and services

uses

uses

specifies

specifies

builds

requires

Framework Developer

DSL User

DSL Designer

DSL Metamodel

Domain Model

Metamodelling Language

Guidance and services

Constraints based on defined services

uses

uses

specifies

specifies

builds

requires

Scenario (Roles)
Scenario

- Modeling physical devices, e.g. Cisco network devices

**Cisco 7603:**

**Domain Model:**

- **Device**
  - Slot
  - HotSwappableOSM
  - SupervisorEngine
  - Slot

Restrictions modeling a Cisco7603 device:

- Every *Cisco7603* has at least 1 *Configuration7603*
- Every *Configuration* has at least 1 *Slot* in which a *SupervisorEngine* card is plugged in
- A *Configuration7603* has exactly 3 *Slots* in which either a *HotSwappableOSM* or *SPAInterface* card is plugged in.
Scenario (DSL User)

• Domain Model: (inconsistent)

• Requirements of DSL User:
  • Consistency Checking
  • Debugging of domain models
Scenario (DSL User)

- Domain Model: (consistent)

- Requirements of DSL User:
  - Consistency Checking
    - Debugging of domain models
  - Validate incomplete models
    - Guidance and explanations how to complete the model
Scenario (DSL User)

- Domain Model: (inconsistent)

- Requirements of DSL User:
  - Consistency Checking
    - Debugging of domain models
  - Validate incomplete models
    - Guidance and explanations how to complete the model

Explanation:
Configuration hasSlot some Slot and hasCard some SupervisorEngine
Scenario (DSL User)

• Domain Model: (consistent)

• Requirements of DSL User:
  • Consistency Checking
    • Debugging of domain models
  • Validate incomplete models
    • Guidance and explanations how to complete the model
  • Suggestions of suitable domain concepts
  • Use of services without any extra effort
Metamodel of *Physical Device DSL* (PDDSL) (M2 layer)
• Implemented using KM3 (a Java-like syntax)
• Simple to use and understandable
• *But:* Not effectual to define configurations with valid cards and slots

```java
class Device {
    reference hasConfiguration [1-*]: Configuration;
}
class Cisco7603 extends Device{
}
class Configuration {
    reference hasSlot [1-*]: Slot;
}
class Configuration7603 extends Configuration{
}
class Slot {
    reference hasCard [1-*]: Card;
}
class Card {
}
```
Description Logics (DLs) are logics designed to represent and reason on structured knowledge.

The domain of interest is structured into (TBox):

- concepts, which correspond to classes, and denote sets of individuals
- roles, which correspond to associations, and denote binary relations on individuals

The knowledge is asserted through so-called assertions (ABox).

They provide formal semantics.

DLs provide the foundations for standard ontology languages, like OWL2.
## Description Logic (Example)

**TBox:**

- \(Cisco7603 \sqsubseteq Device\) (1)
- \(Cisco7603 \equiv \exists \text{hasConfiguration} . Configuration7603\) (2)
- \(Configuration \sqsubseteq \exists \geq 1 \text{hasSlot}.Slot \sqcap \exists \text{hasSlot}.(\exists \text{hasCard}.\text{Supervisor Engine})\) (3)
- \(Configuration7603 \equiv \exists = 3 \text{hasSlot}.Slot \sqcap \exists \text{hasSlot}.(\exists \text{hasCard}.(\text{HotSwappableOSM} \sqcup \text{SPAInterface} ))\) (4)
- \(Slot \sqsubseteq \exists \text{hasCard}.Card\) (5)
- \(Card \equiv \text{HotSwappableOSM} \sqcup \text{SPAInterface} \sqcup \text{Supervisor Engine}\) (6)

**ABox:**

- \(Cisco7603(cisco)\) \(\sqsubseteq Device(cisco)\) (1)
- \(Configuration7603(conf)\) \(\sqsubseteq Configuration(conf)\) (2)
- \(Slot(cslot1), Slot(cslot2), Slot(cslot3)\) (3)
- \(\text{Supervisor Engine}(\text{supervisor720}), \text{SPAInterface}(\text{spa360})\) (4)
- \(\text{hasConfiguration}(cisco, conf)\) (5)
- \(\text{hasSlot}(conf, cslot1), \text{hasSlot}(conf, cslot2), \text{hasSlot}(conf, cslot3)\) (6)
- \(\text{hasCard}(cslot1, \text{supervisor720}), \text{hasCard}(cslot2, \text{spa360})\) (7)
Description Logics

• DLs allow for joint as well as for separate sound and complete reasoning at the model and at the instance level
• DLs allow for tractable reasoning and efficient query answering
• DLs provide more expressiveness than usual metamodeling languages

• But: DLs are not designed to act as a metamodel for defining DSLs
• DSL Designer has no experience modeling with DLs and ontologies
• DSL Designer requires to use the languages he is familiar with as much as he can (standard metamodeling language like KM3)
• DSL Designer wants to define constraints
Proposed Solution

Ontology-based framework for domain-specific languages

Integrate KM3 with ontology language OWL2
- Provide a metamodeling language to specify further DSLs

Design Domain Specific Languages
- Develop new DSL with integrated constraints and axioms

Use domain-specific languages
- Builds domain models and uses services
Model-based Integration Architecture

- Framework Developer
  - Provide framework for designing and using DSLs
- DSL Designer
  - Defines abstract Syntax, concrete Syntax, semantics
- DSL User
  - Builds domain models

Development Environment

Integrated Metametamodel

\[ \text{Integrated Metametamodel} \]

\[ \text{KM3 Metametamodel} \]

\[ \text{OWL Metamodel} \]

\[ \text{:M3} \]

\[ \text{instanceOf} \]

Domain Definition

\[ \text{Domain Definition Metamodel} \]

\[ \text{(Concrete Syntax)} \]

\[ \text{Visualizaton} \]

\[ \text{Integration Model} \]

\[ \text{(Abstract Syntax)} \]

\[ \text{Transform} \]

\[ \text{Transform} \]

\[ \text{Ontology Definition} \]

\[ \text{Reasoning Service} \]

TBox

ABox

Development Environment

ISWeb - Information Systems & Semantic Web

Tobias Walter

walter@uni-koblenz.de

MoDELS 2009

14 of 18
Integrated Modeling

• Metamodel of PDDSL

class Device {
    reference hasConfiguration [1-*]: Configuration;
}

class Cisco7603 extends Device {
    equivalentWith restrictionOn hasConfiguration
    with min 1 Configuration7603 {
    }
}

class Configuration {
    equivalentWith
    IntersectionOf(restrictionOn hasSlot with min 1 slot,
    restrictionOn hasSlot some
    restrictionOn hasCard some SupervisorEngine) {
        reference hasSlot : Slot;
    }
}

class Configuration7603 extends Configuration {
    equivalentWith IntersectionOf(restrictionOn hasSlot with exactly 3 slot,
    restrictionOn hasSlot with some
    restrictionOn hasCard with some
    UnionOf(HotSwappableOSM, SPAInterface) {
    }
}

class Slot {
    reference hasCard [1-*]: Card;
}
Benefits of DL in Domain Modeling

Open World Assumption
- assumes incomplete information by default
- guidance and validation of incomplete models

Joint semantic definitions at 2 layers
- M1- and M2 layer affect each other
- simultaneously reasoning at M1- and M2 layer

Debugging and reasoning explanation
- identifying debugging-relevant facts (e.g. model elements) which lead to inconsistency with regard to the metamodel
- explanations of errors in domain models
Conclusion

Framework Developer
• Integration of KM3 and OWL at the M3 layer
• Provide metamodeling language that allows designing metamodels with embedded OWL Constructs

DSL Designer
• Specifies new DSLs with additional, integrated constraints

DSL User
• Builds domain models
• Gets services and guidance for free
Thanks for your attention …

… Questions?

http://ontodsl.semanticsoftware.eu

Supported by EU STReP-216691 MOST