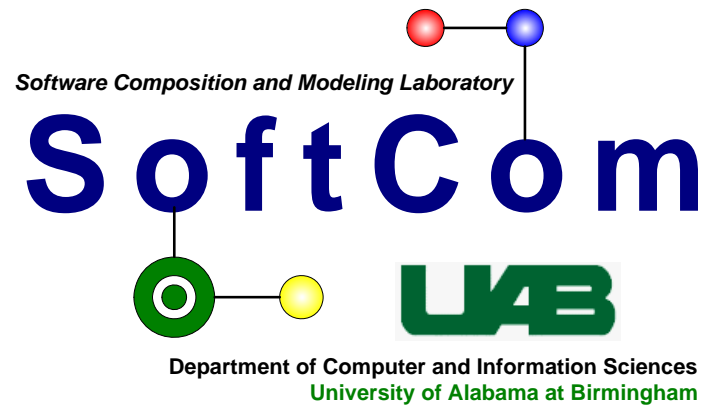


Using Ontologies in the Domain Analysis of Domain-Specific Languages

Robert Tairas, Marjan Mernik, Jeff Gray



Workshop on Transformation and Weaving Ontologies in Model-Driven Engineering (TWOMDE)
International Conference on Model-Driven Engineering, Languages, and Systems (MoDELS)
Toulouse, France, September 28, 2008

Outline

- DSLs and Domain Analysis
- Ontologies
- Case Study
- Summary
- Future work

DSLs and Domain Analysis

- A DSL is a computer language dedicated to a particular **domain**. It provides appropriate built-in abstractions and notations*.
- Programs in DSLs explicitly specify only part of the behavior because a significant portion of the behavior is implicit and fixed.

* Mernik, M., Heering, J., Sloane, A.: When and How to Develop

DSLs and Domain Analysis

- DSL development phases:
 - decision
 - **analysis**
 - **design**
 - implementation
 - deployment
 - maintenance

DSLs and Domain Analysis

- The beginning phases of DSL development are less well understood. Some open issues:
 - **How should results from domain analysis drive the language design process?**
 - Should a DSL be designed by a domain expert, GPL designer or software language engineer?
 - **How much domain analysis and language design is actually needed?**
 - Can we build tools which support us in earlier phases of DSL development?

DSLs and Domain Analysis

- Domain Analysis
 - To build the domain model (an explicit representation of the common and the variable properties of a domain, the semantics of the properties, the dependencies between the properties).
- Some typical domain analysis activities are analysis of similarities, analysis of variations, and analysis of combinations.

DSLs and Domain Analysis

- Many domain analysis methods exist, such as:
 - Feature-Oriented Domain Analysis (FODA)
 - Draco
 - Domain Analysis and Reuse Environment (DARE)
 - Family-Oriented Abstraction, Specification, and Translation (FAST)
 - ...
- But, they are rarely used in DSL development and domain analysis is usually done informally and in an incomplete manner.

DSLs and Domain Analysis

- Why?
- The first observation might be that information gathered during domain analysis cannot be automatically used in the language design process.
- Another reason might be that complete domain analysis is too complex and outside of a software engineer's capabilities.

DSLs and Domain Analysis

- We strongly believe that good domain analysis will pay off for even a medium size DSL community and medium DSL life span.
- Possible alternatives:
 - Grammatical inference
 - API2DSL
 - **Ontologies**
 - ...

Ontologies

Ontology

- Representative vocabulary
 - Objects, concepts, and other entities of a domain
- Body of knowledge
 - Relationships of entities from vocabulary

=

DSL Domain Model

- A domain defining the scope of the domain
- The domain terminology
- Description of domain concepts

- Commonalities and variabilities of domain concepts for domain-specific language

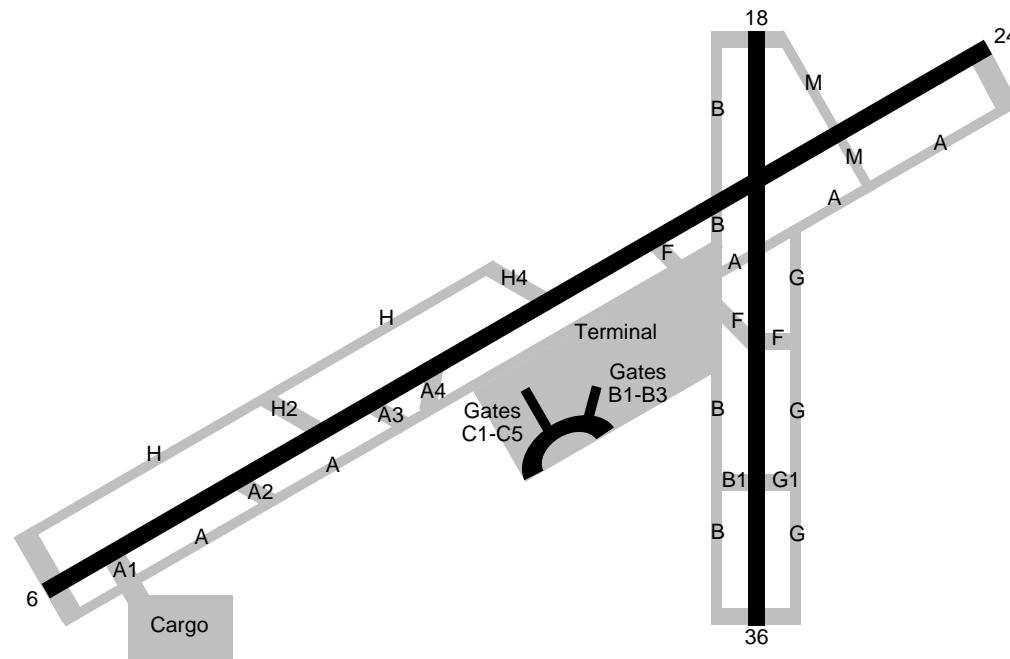
?

Ontologies

- Ontology competency questions to achieve the goals of domain analysis
 - *What are the concepts of the domain and the interdependencies of these concepts?*
 - *What are the commonalities and variabilities of the domain?*

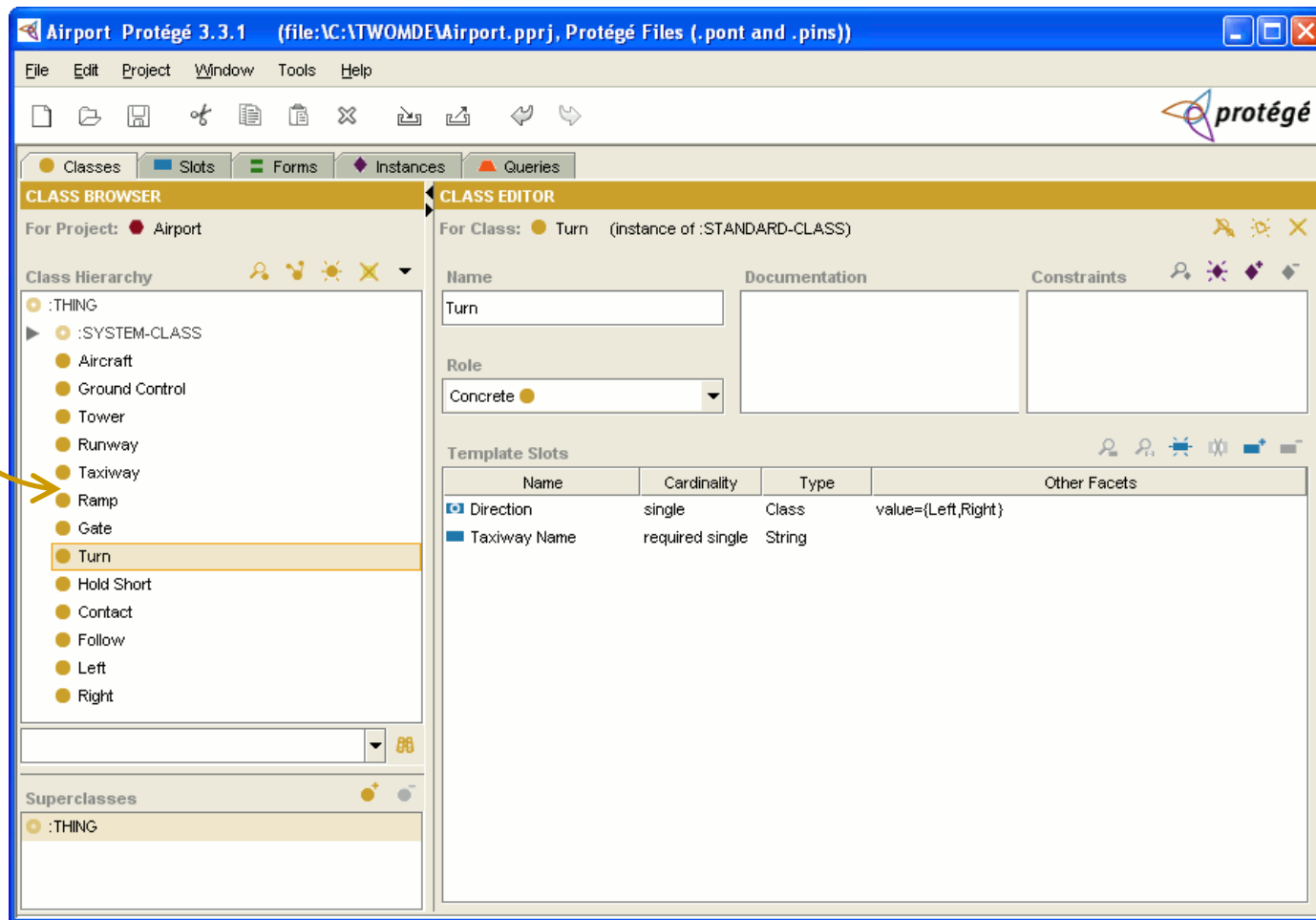
Case Study

- Air Traffic Communication (Video)
- BHM used in case study



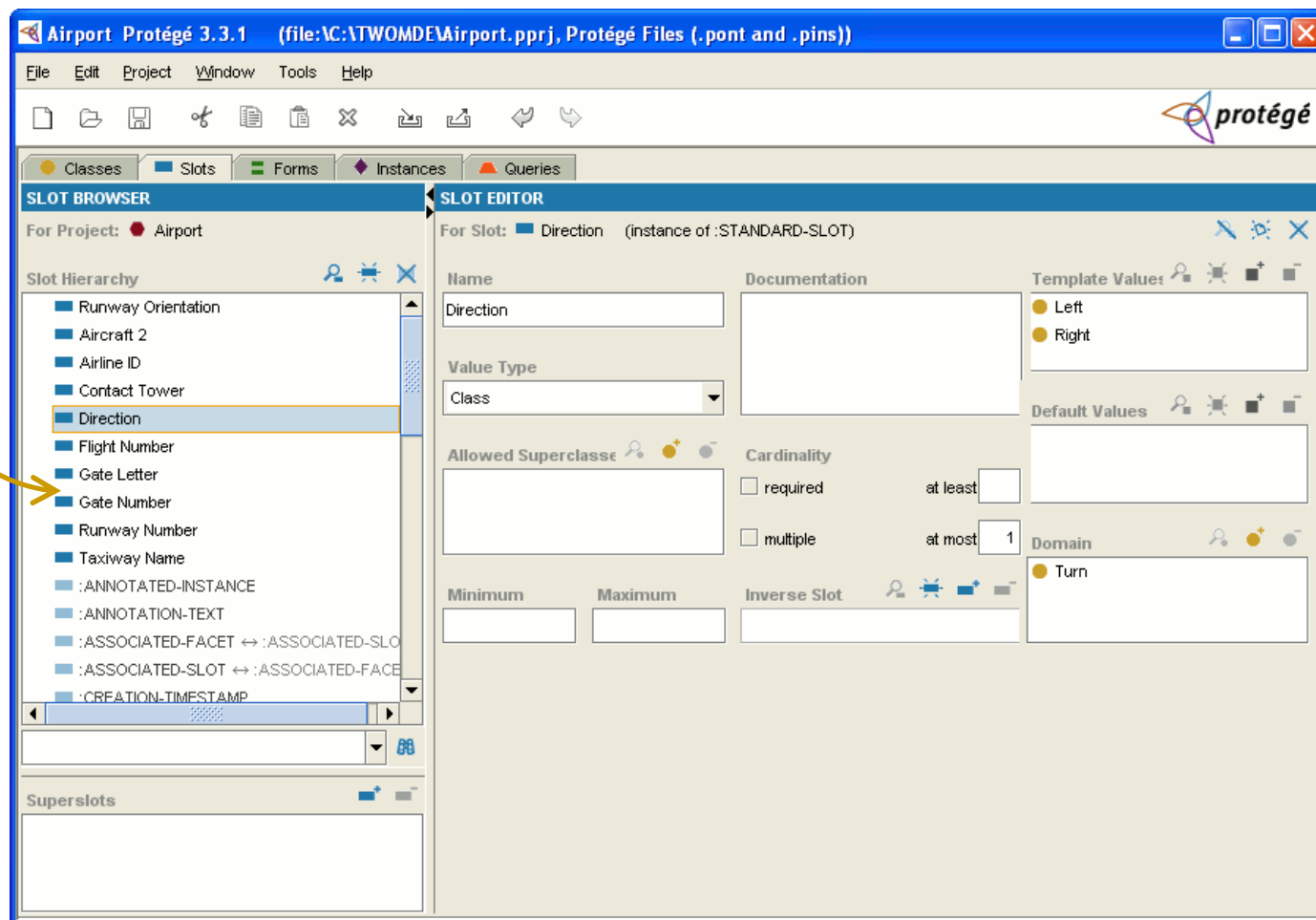
Protégé (Classes)

Classes: terms related to the domain



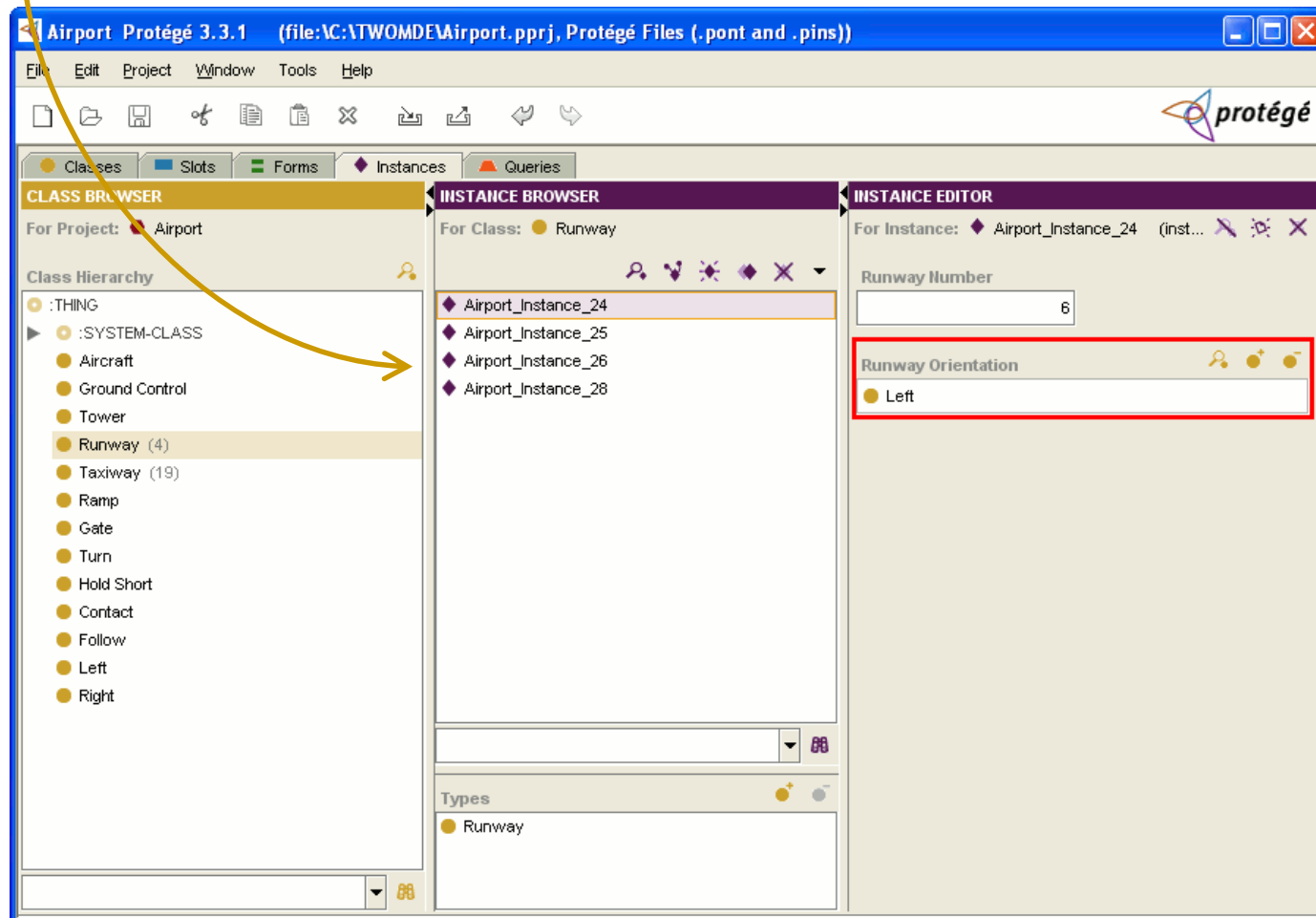
Protégé (Slots)

Slots: properties and constraints of the classes



Protégé (Instances)

Instances: represents specific real world instance

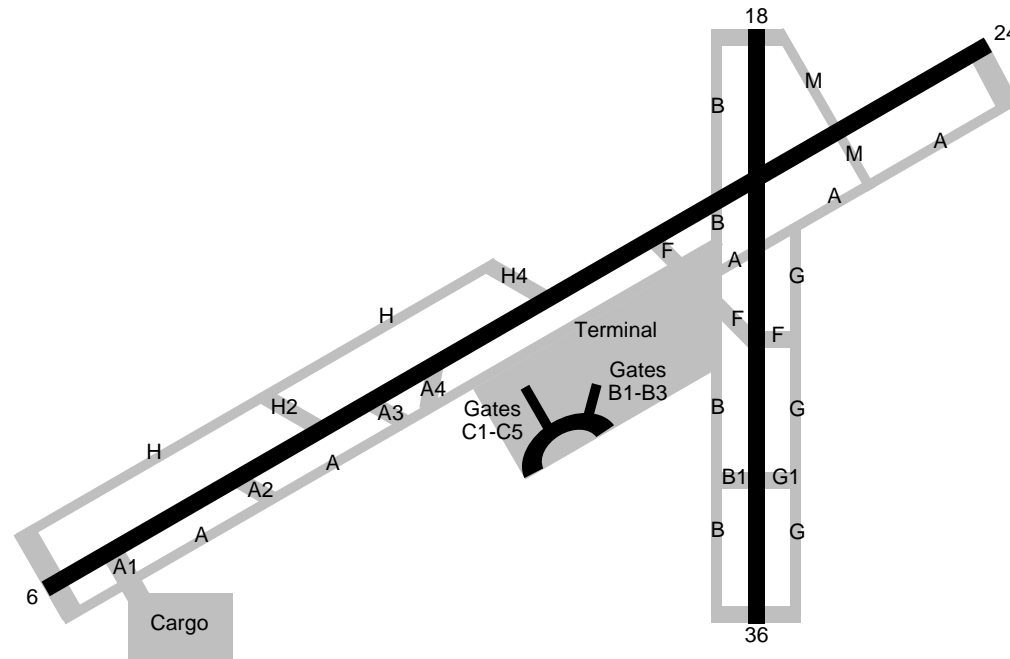


Ontology from Protégé

Listing of classes and associated slots

Class	Description	Slots		
		Name	Description	Values
Aircraft	Arriving or departing aircraft	Airline ID	Name of the airline	Two letters
		Flight Number	Flight Identification	Integer
GroundControl	Controller in charge of airport ground traffic			
Tower	Controller in charge of take-offs and landings			
Runway	Available take-off and landing locations	Runway Number	Runway Identification	1 – 36 (i.e., runway heading 10° – 360°)
		Runway Orientation	To distinguish parallel runways	Class Left or Right
Taxi way	Paths connecting runways to ramps	Taxi way Name	Taxiway Identification	One or two letters (digits)
Ramp	Aircraft parking area	Ramp Name	Ramp Identification	String
Gate	Passenger embarkation and disembarkation	Gate Letter	Gate Identification	One letter
		Gate Number	Gate Identification	Integer
Turn	Command to turn	Direction	Turning direction	Class Left or Right
		Taxi way	Taxiway Identification	Class Taxi way
HoldShort	Command to hold short of a runway or taxiway	Runway	Runway Identification	Class Runway
		Taxi way	Taxiway Identification	Class Taxi way
Contact	Command to contact a separate controller	ATC	Controller to contact	Class Tower or GroundControl
Follow	Command to follow behind another aircraft	Aircraft	Aircraft Identification	Class Aircraft

Instances for BHM



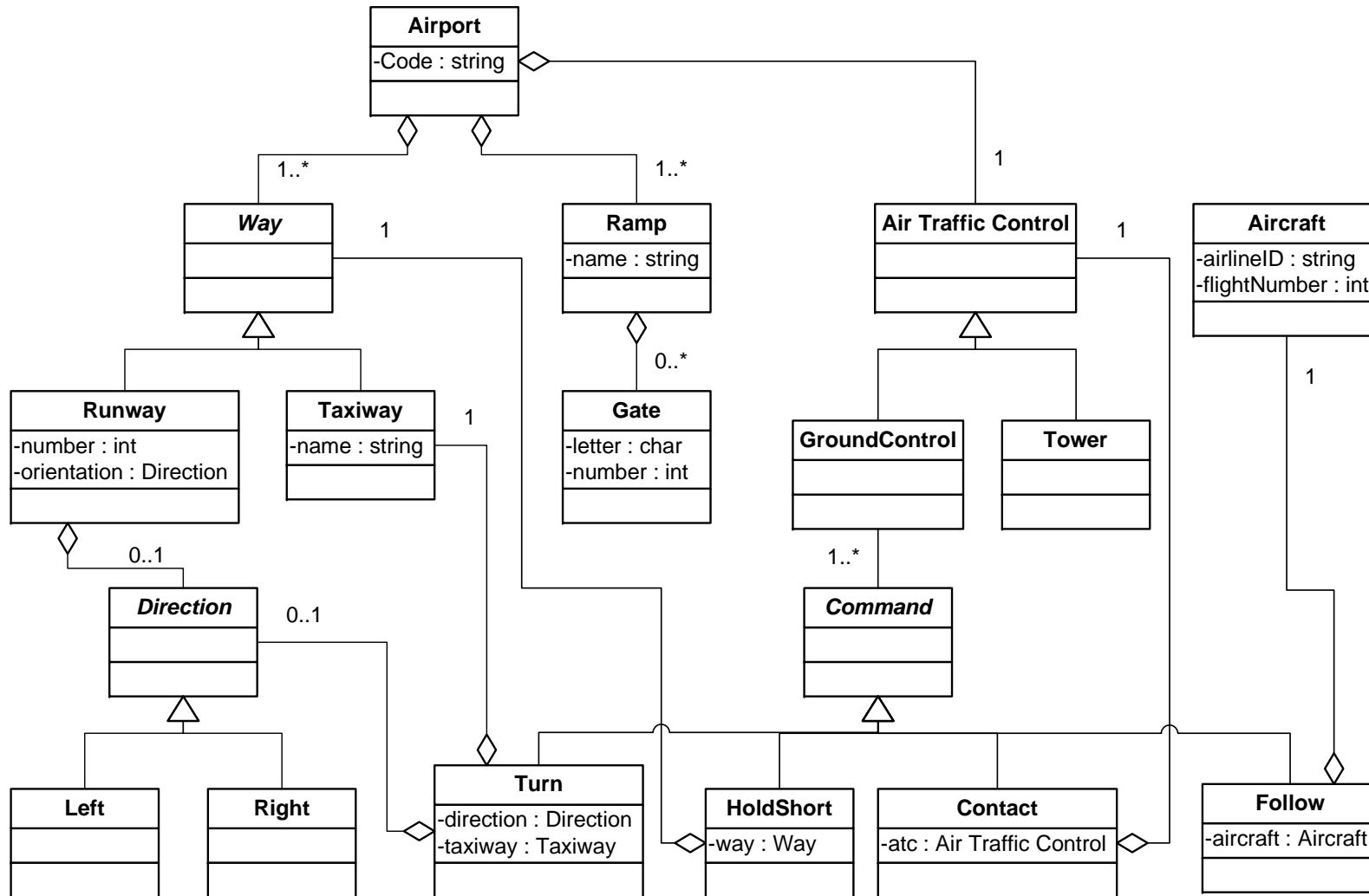
Instances of runways, taxiways, ramps, and gates

Class	Instances
Runway	6, 24, 18, 36
Taxi way	A, B, F, G, H, M, A1, A2, A3, A4, B1, G1, H2, H4
Ramp	Cargo, Terminal
Gate	B1, B2, B3, C1, C2, C3, C4, C5

Revisiting Competency Questions




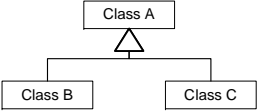
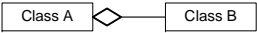

- *What are the concepts of the domain and the interdependencies of these concepts?*
 - Identified in classes and slots
- *What are the commonalities and variabilities of the domain?*
 - Observe the instances of classes

Class Diagram from Ontology



Class Diagram to Context-Free Grammar

Transformations

Association	Class Diagram Element	Grammar
Class		Class (non-terminal) Attribute (terminal)
Association		$A ::= B$
Navigability		$A ::= B$
Generalization		$A ::= B \mid C$
Aggregation		$A ::= B$
Composition		$A ::= B$

AIRPORT	$::= \text{WAYS RAMPs ATC}$
WAYS	$::= \text{WAYS WAY} \mid \text{WAY}$
WAY	$::= \text{runway RUNWAY} \mid \text{taxiway TAXIWAY}$
RUNWAY	$::= \text{number DIRECTION}$
TAXIWAY	$::= \text{name}$
RAMPs	$::= \text{RAMPs RAMP} \mid \text{RAMP}$
RAMP	$::= \text{ramp name GATES}$
GATES	$::= \text{GATES GATE} \mid \epsilon$
GATE	$::= \text{gate letter number}$
ATC	$::= \text{GROUNDCONTROL} \mid \text{TOWER}$
GROUNDCONTROL	$::= \text{COMMANDs}$
COMMANDs	$::= \text{COMMANDs COMMAND} \mid \text{COMMAND}$
COMMAND	$::= \text{CONTACT} \mid \text{FOLLOW} \mid \text{HOLDSHORT} \mid \text{TURN}$
CONTACT	$::= \text{contact ATC}$
FOLLOW	$::= \text{follow AIRCRAFT}$
HOLDSHORT	$::= \text{hold short WAY}$
TURN	$::= \text{turn DIRECTION on TAXIWAY}$
DIRECTION	$::= \text{left} \mid \text{right} \mid \epsilon$
AIRCRAFT	$::= \text{airlineID flightNumber}$
TOWER	$::= \text{tower}$

Example program

```
// description of BHM ai rport
runway 6 runway 24 runway 18 runway 36
taxi way A taxi way A1 taxi way A2 taxi way A3 taxi way A4 taxi way B taxi way B1
taxi way F taxi way G taxi way G1 taxi way H taxi way H2 taxi way H4
ramp Cargo
ramp Terminal gate B1 gate B2 gate B3 gate C1 gate C2 gate C3 gate C4 gate C5

// commands from Ground Control
turn right on A
turn left on M
hold short runway 18
contact tower
```

Summary

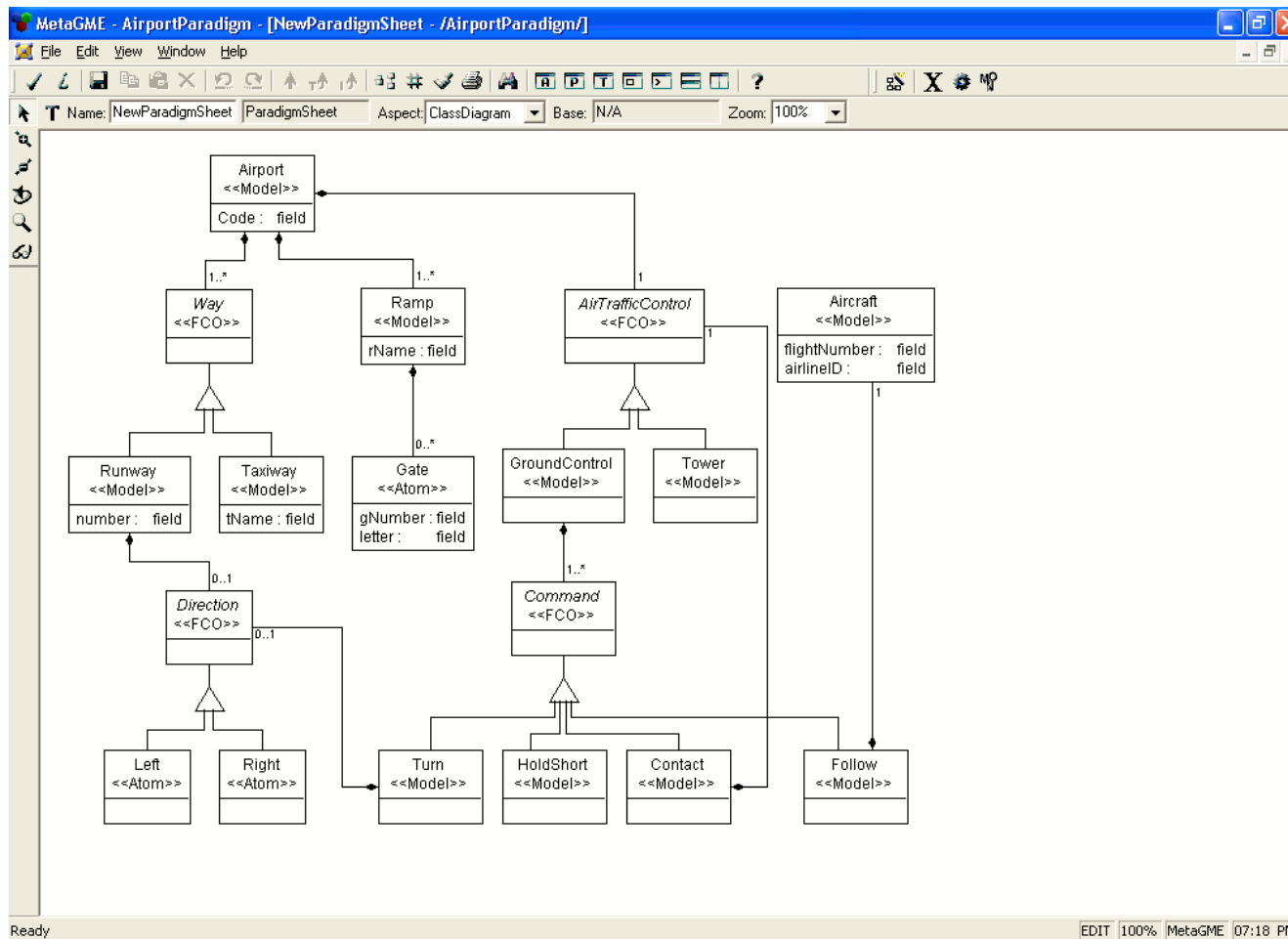
- In domain analysis for DSL development, the ontology assisted in identifying
 - Concepts and interdependencies
 - Commonalities and variabilities
- Inclusion in developmental process, the ontology can
 - Provide a head start in the language development
 - Act as an input or alternative to more formal techniques
 - Assist in generating a class diagram and subsequent context free grammar

Future Work

- Using Model-Driven Engineering
 - Generating a metamodel from class diagram
- Ontology transformed manually to class diagram
 - Possibility of transforming representation of ontology (i.e., OWL) directly to BNF

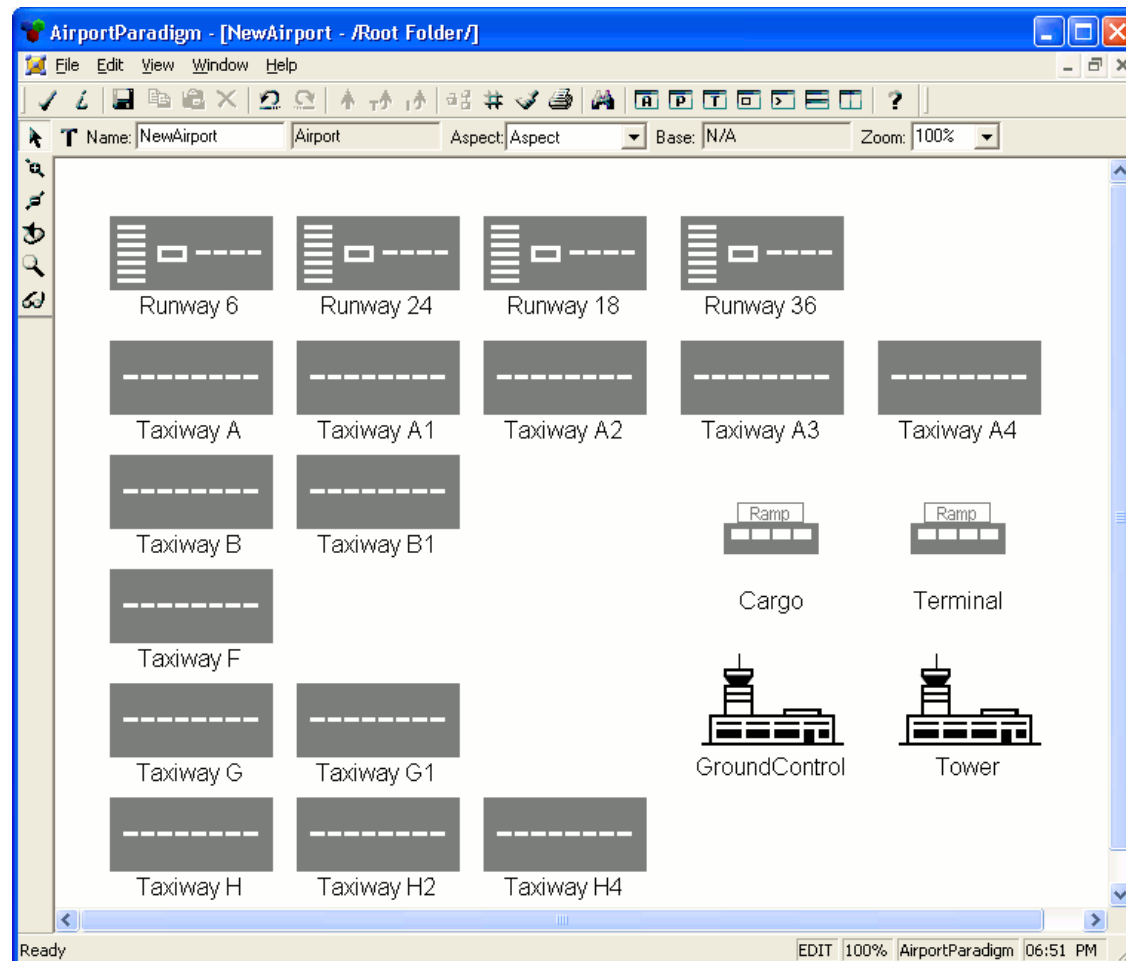
Generic Modeling Environment

Metamodel in GME



Generic Modeling Environment

Model for BHM



Generic Modeling Environment

Commands contained in "GroundControl" element

