JOT: a Protégé Scripting Environment for Creating and Managing Ontologies

Olivier Dameron

SMI - Stanford University

Protégé Conference 2004

Context / Problems

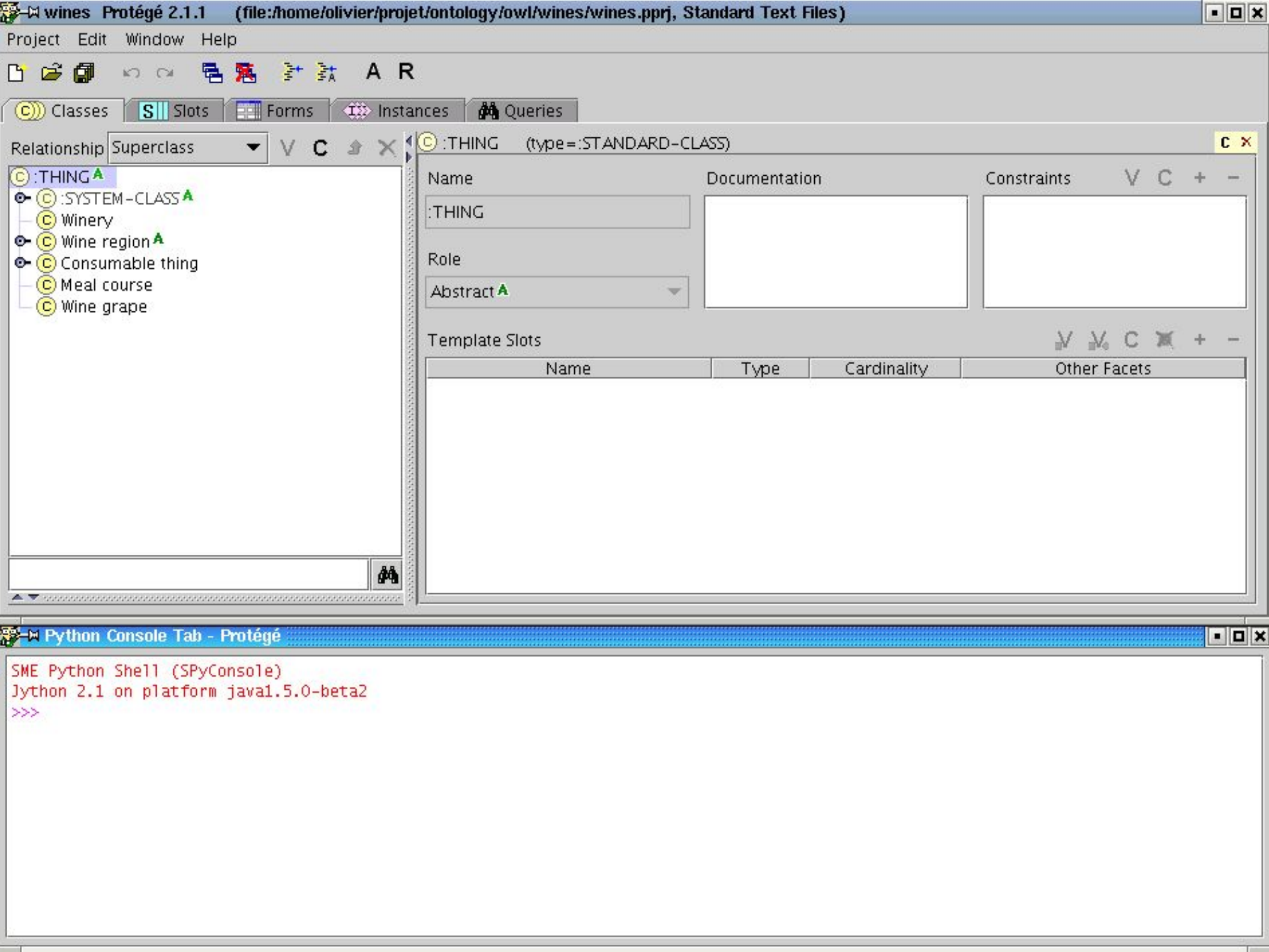
1. Repetitive tasks
   ex: creation of lateralized concepts and their relationships
2. Enumerations
   ex: Ribs
3. Dependancies between concepts or relationships
   ex: Thorax / Skin of Thorax
4. Ontology maintenance
   require adhoc detection and fixing

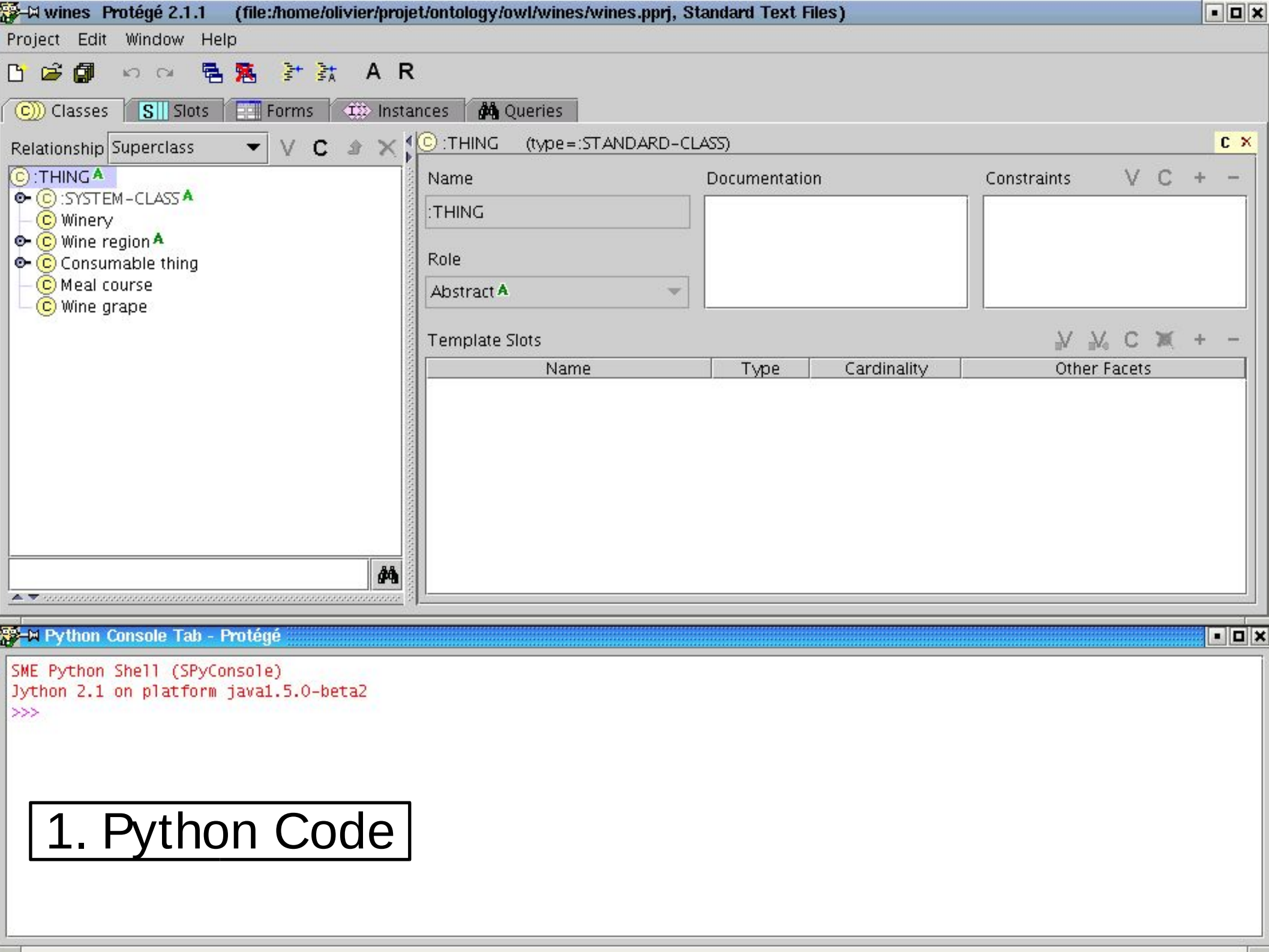
Objective:
Scripting environment for Protégé

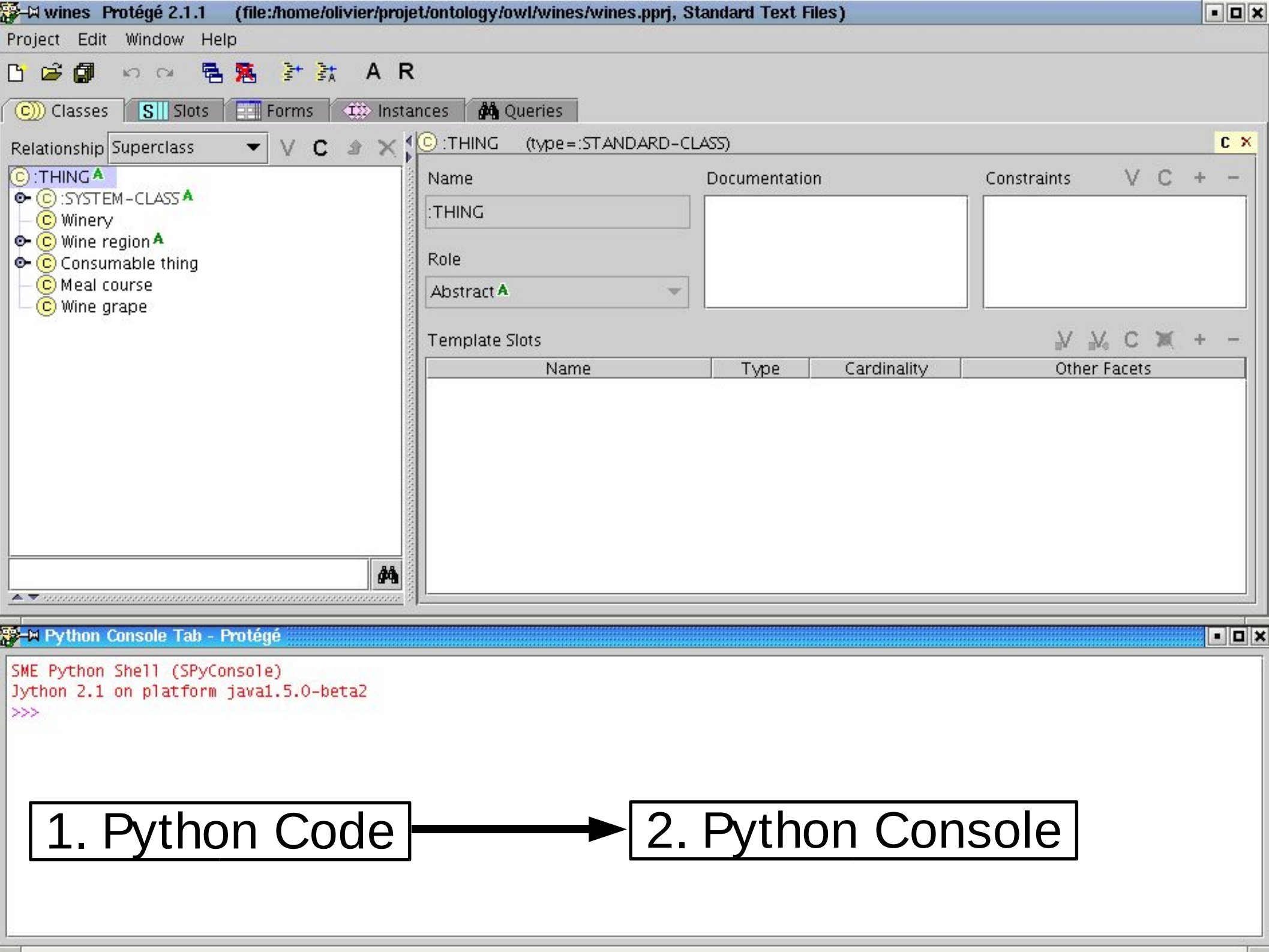
1. Create macros
    1. repetitive and error-prone tasks
    2. formalism for handling intrinsic complexity
    3. towards more abstraction
2. Code reuse
3. User-friendly and powerfull
    1. simple and intuitive syntax
    2. well formalised

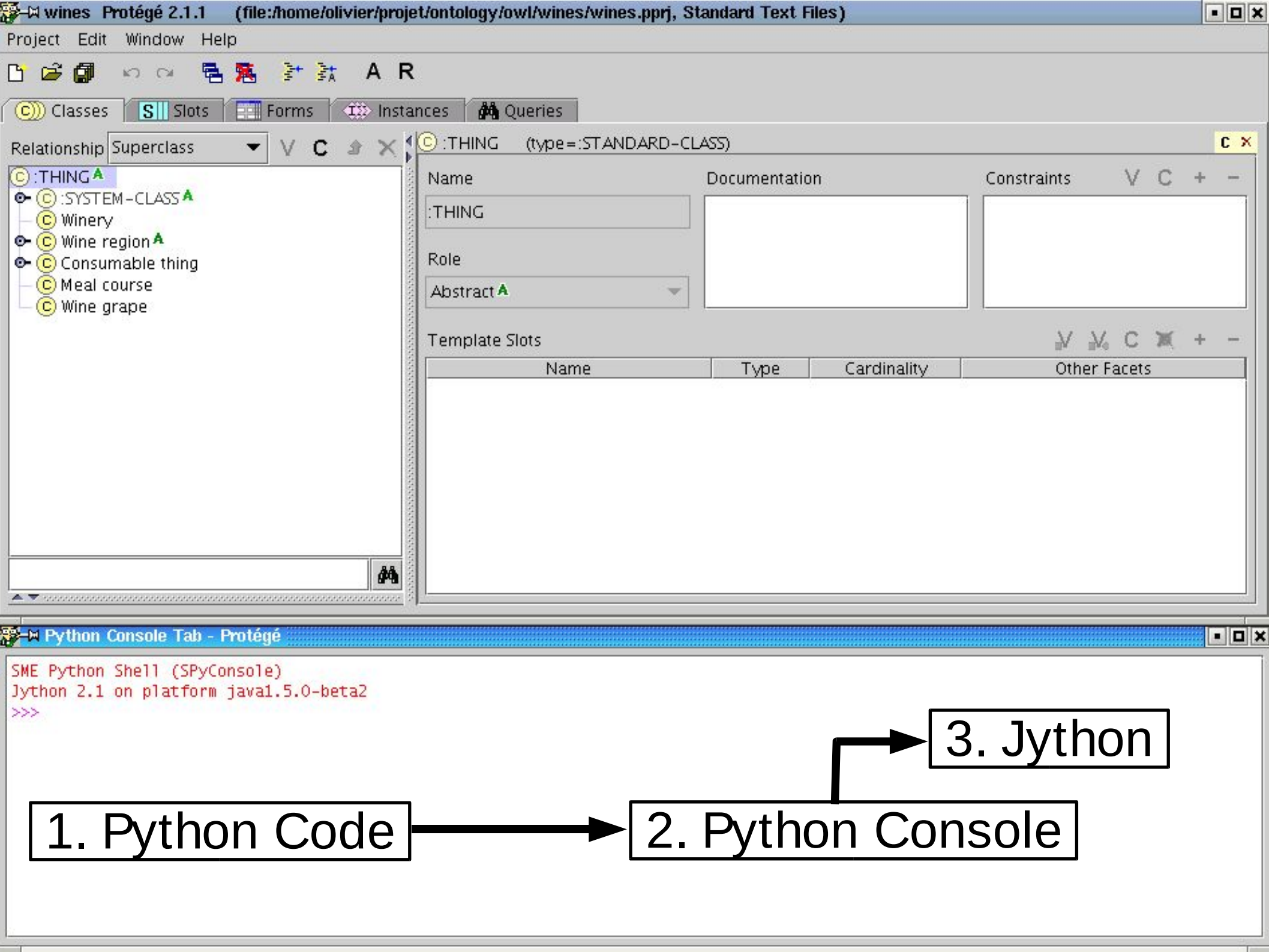
Architecture

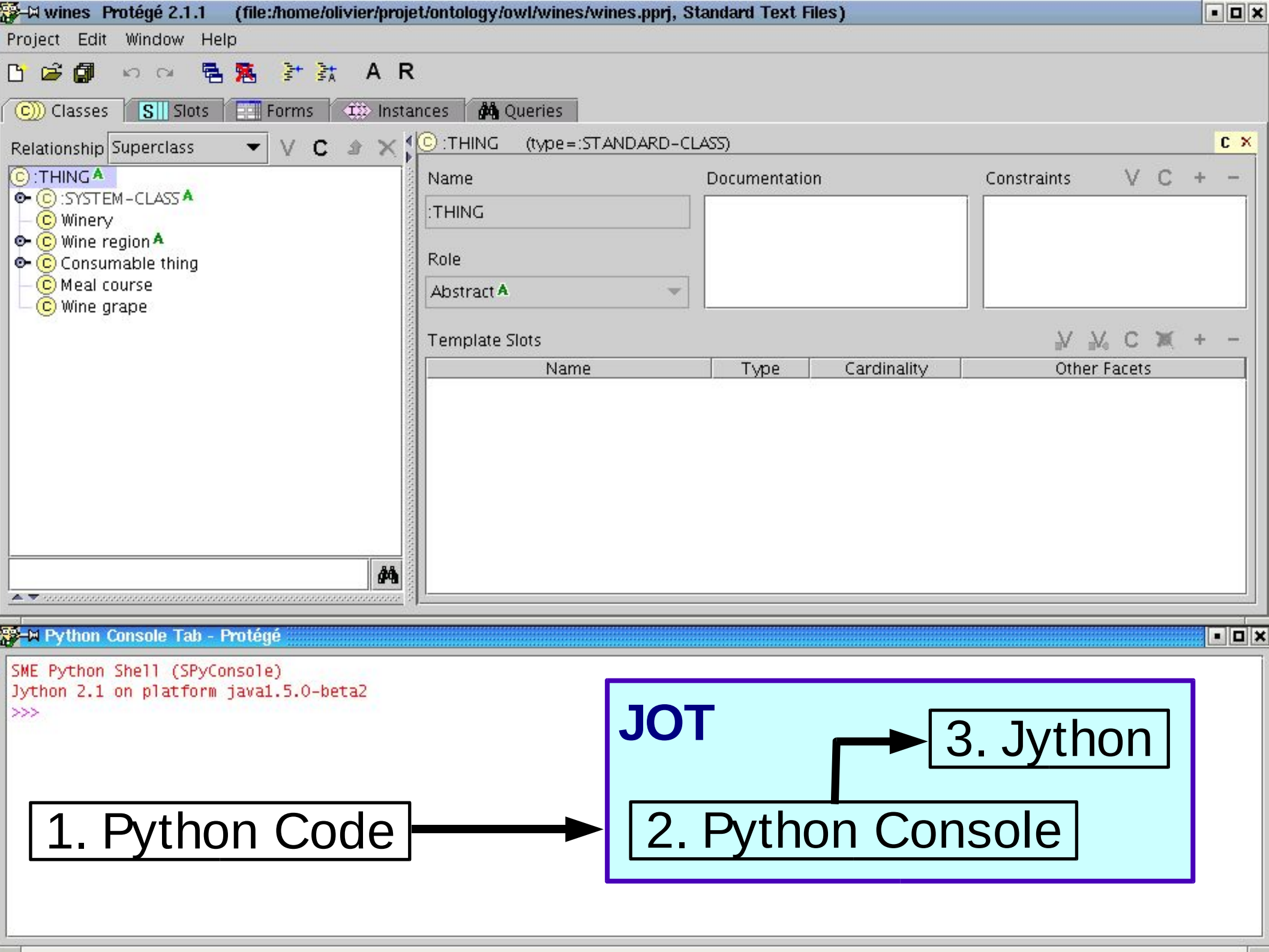
1. Principle
    1. Python interpreter in Java: Jython
    2. Thread (share address space with Protégé)
2. Shared variable: **kb**
3. Compatibility with frames and OWL
    1. instance of KnowledgeBase (Frames)
    2. instance of OWLKnowledgeBase (OWL)

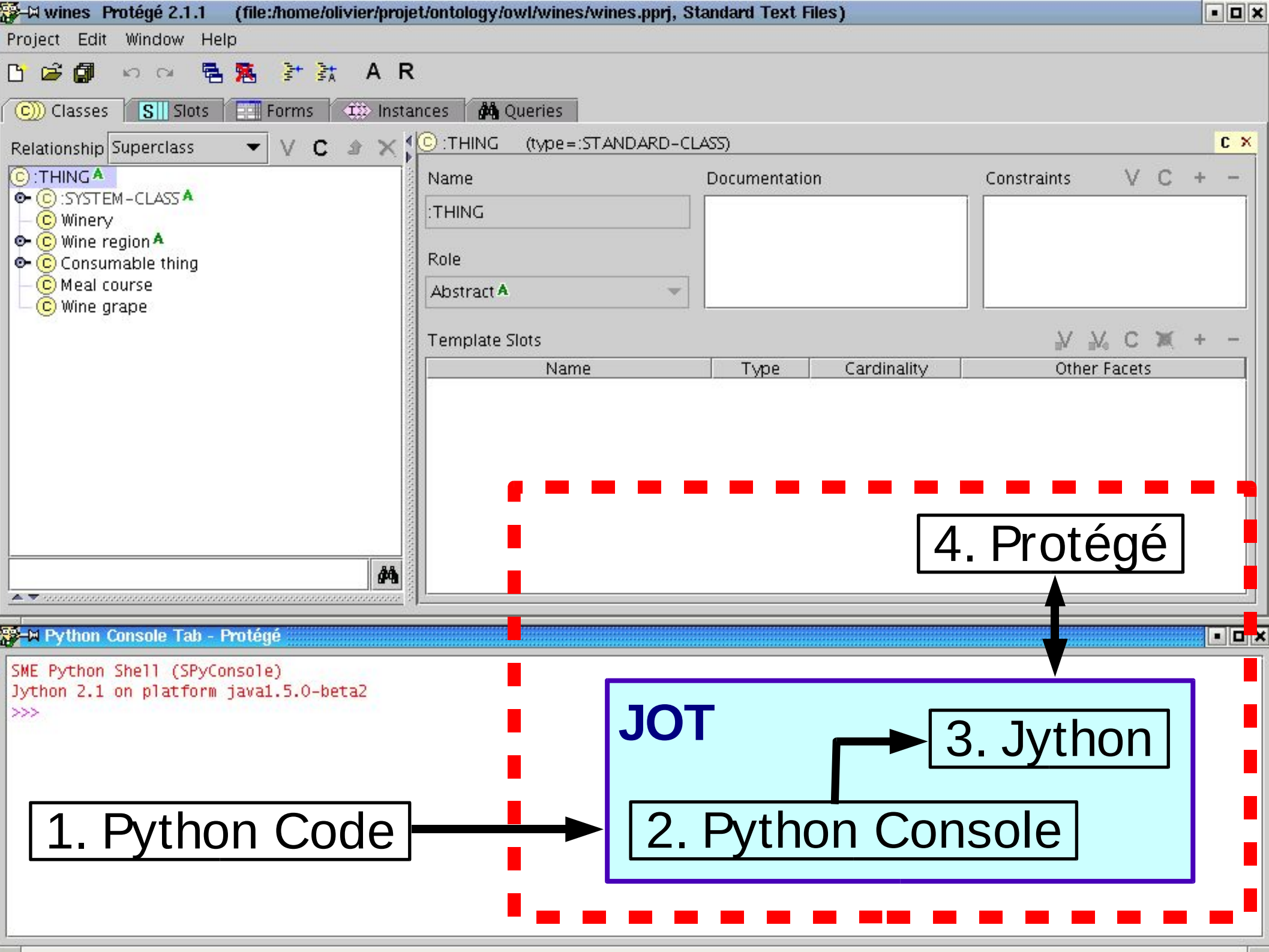

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>

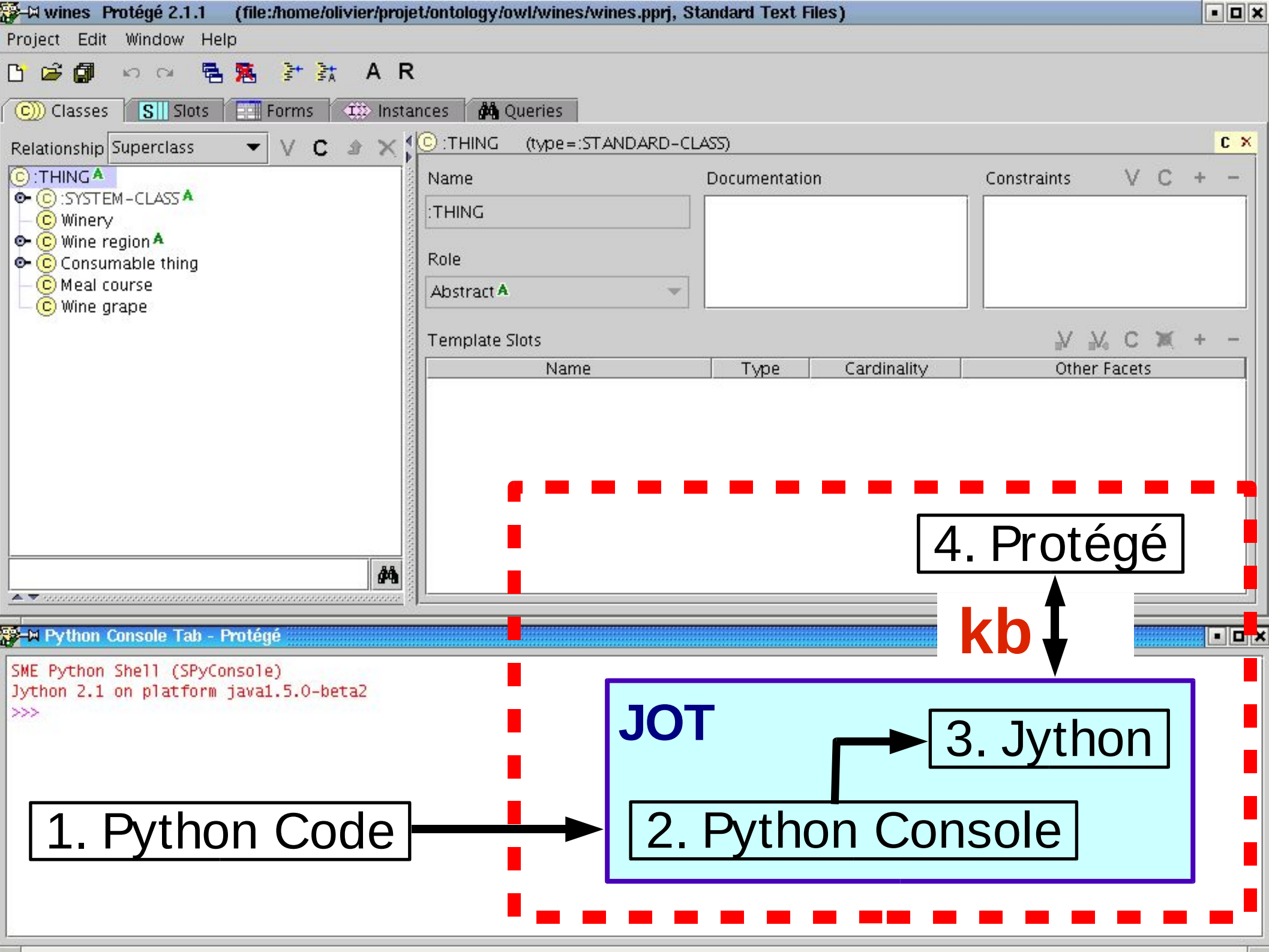

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
1. Python Code


wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
1. Python Code
2. Python Console


wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
1. Python Code
2. Python Console
3. Jython

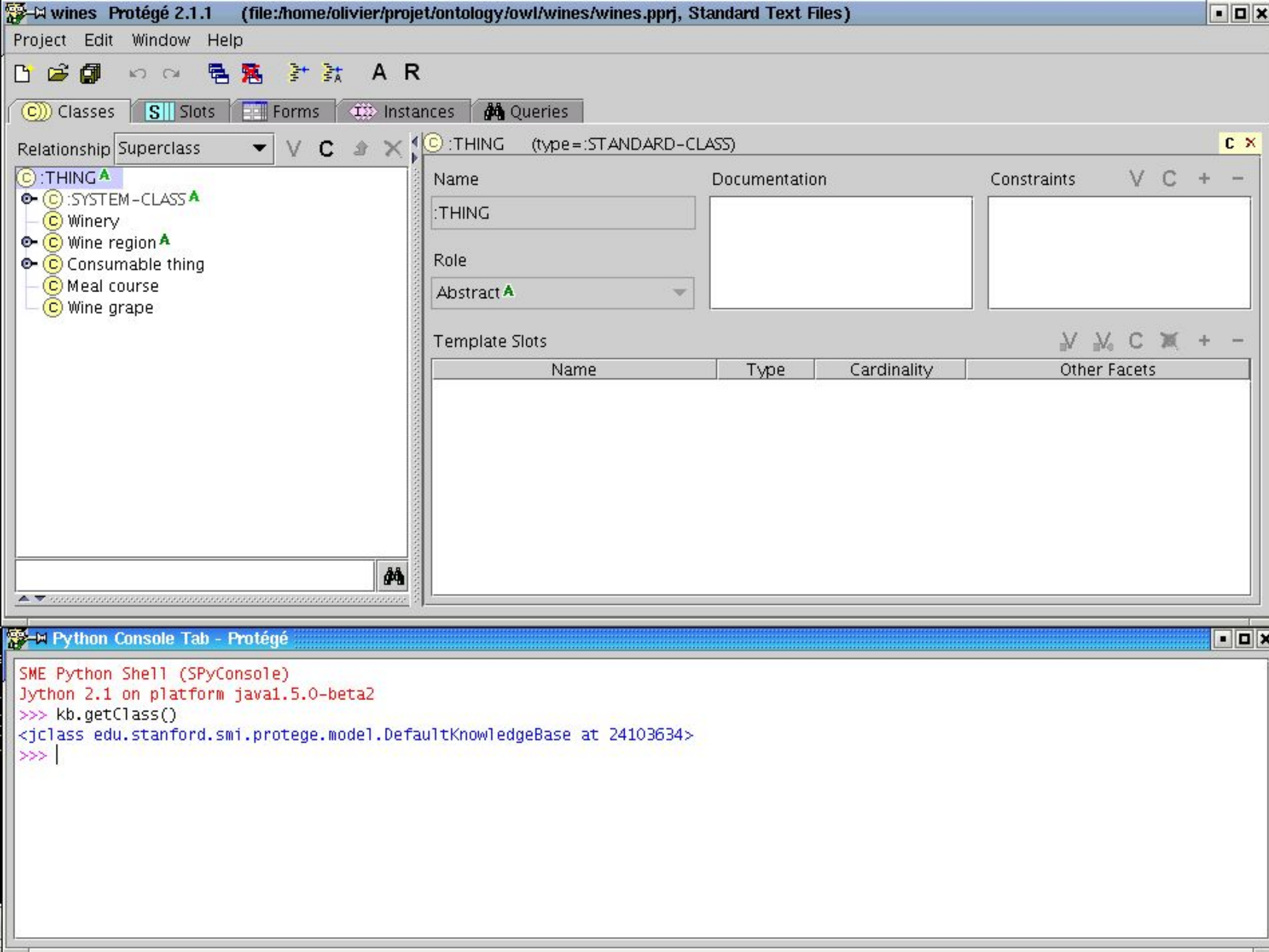

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
JOT
3. Jython
1. Python Code
2. Python Console

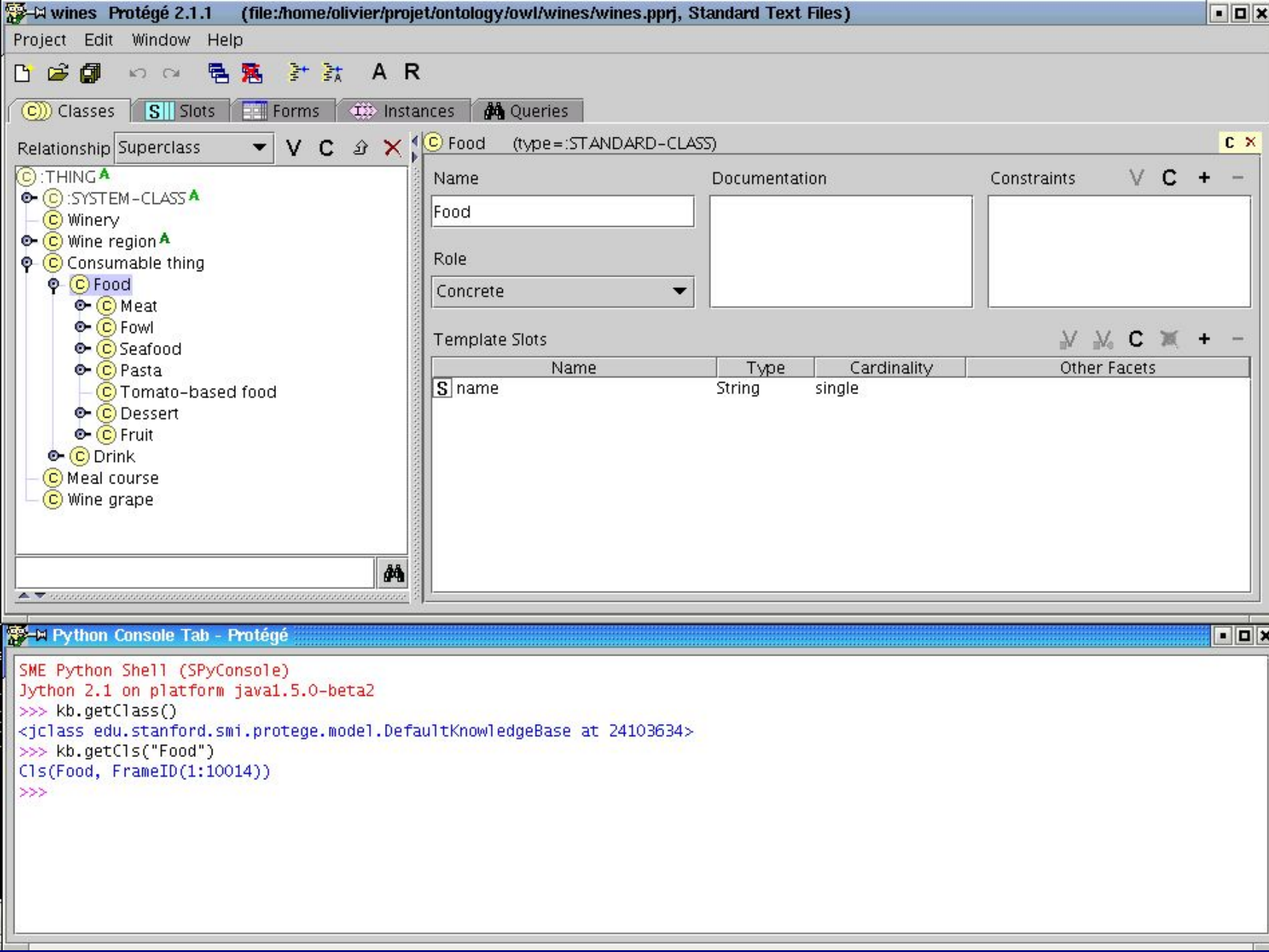

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
4. Protégé
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
JOT
3. Jython
2. Python Console
1. Python Code

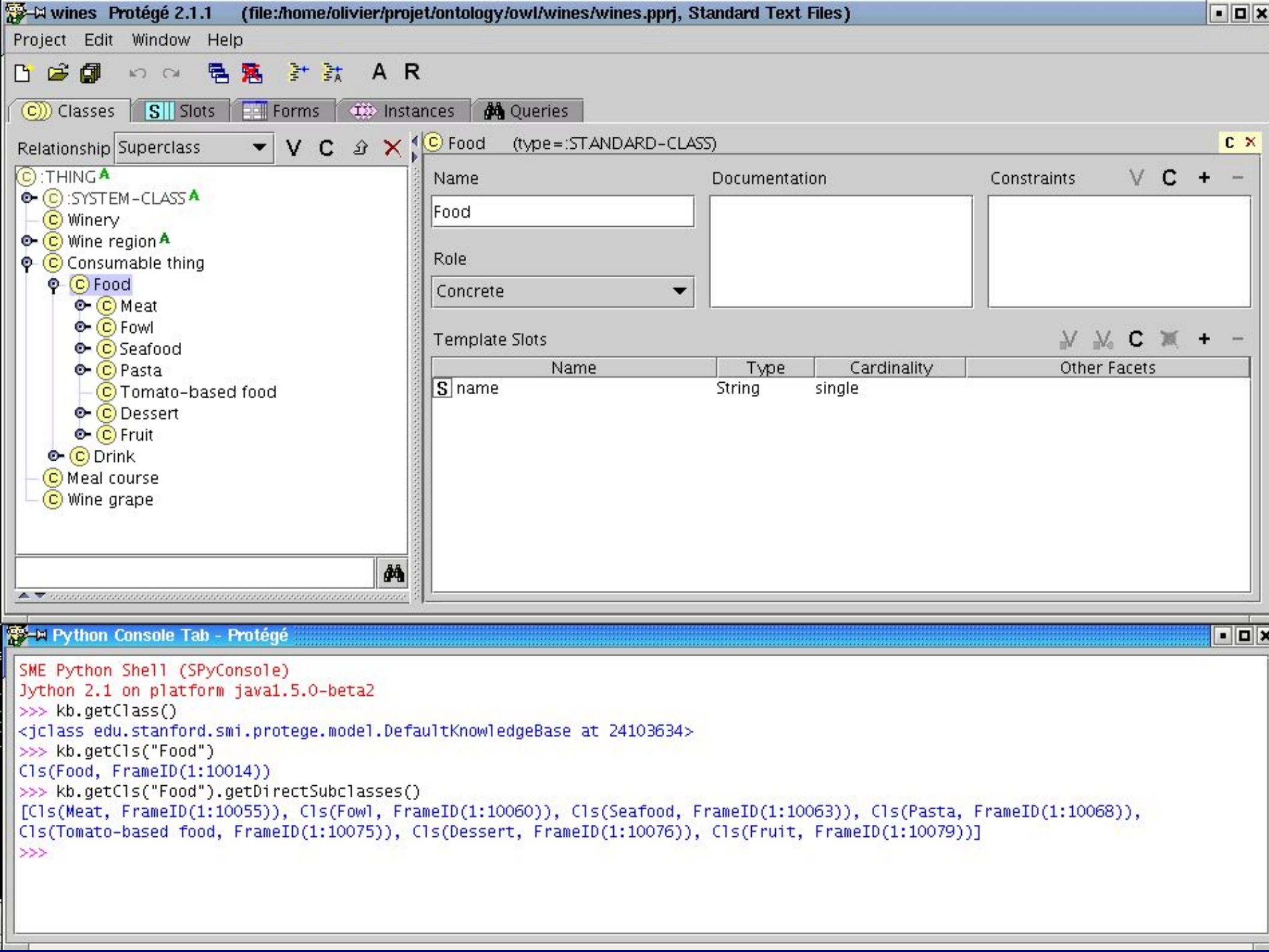

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
4. Protégé
kb
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>
JOT
3. Jython
2. Python Console
1. Python Code

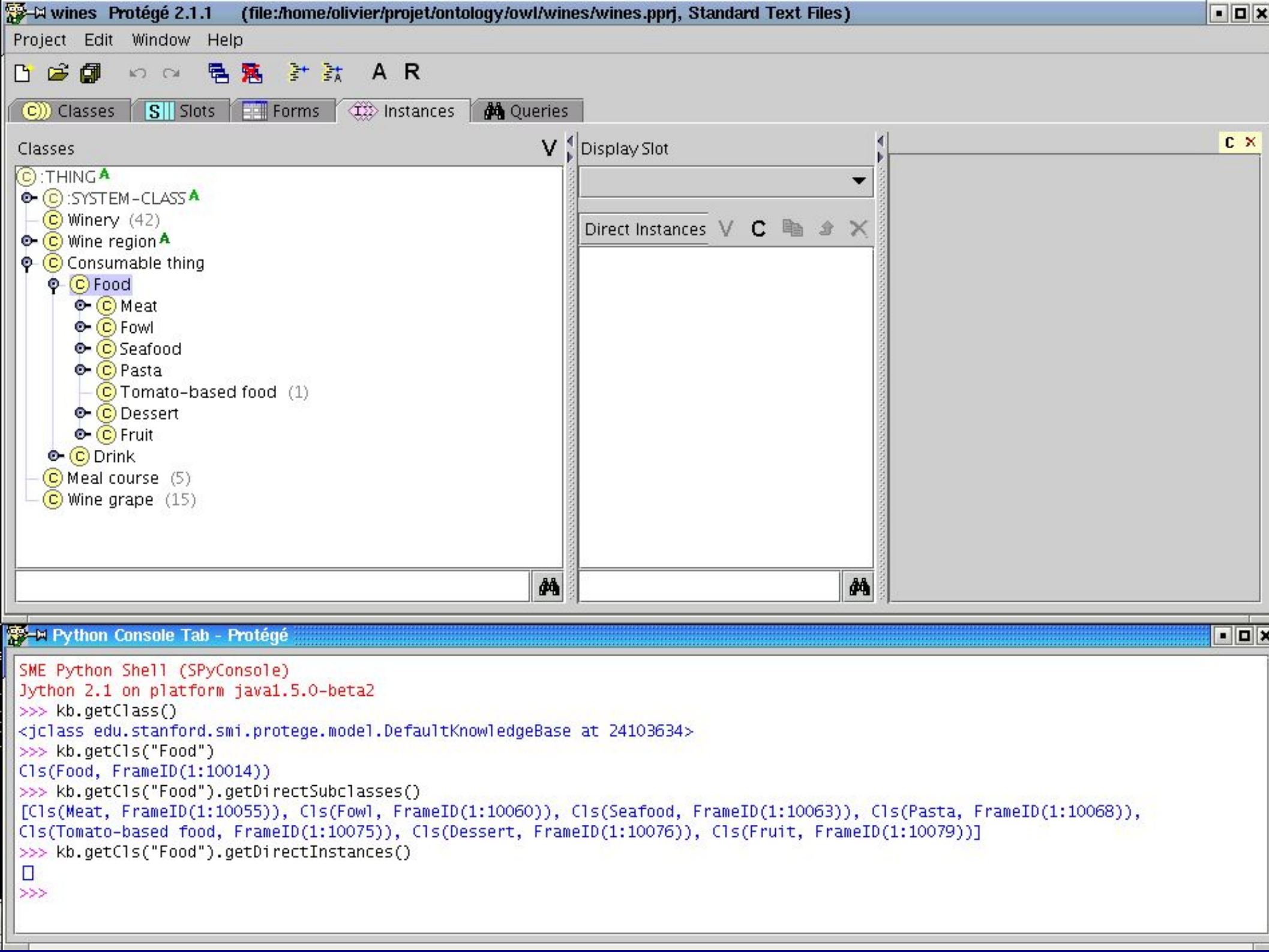
Frames

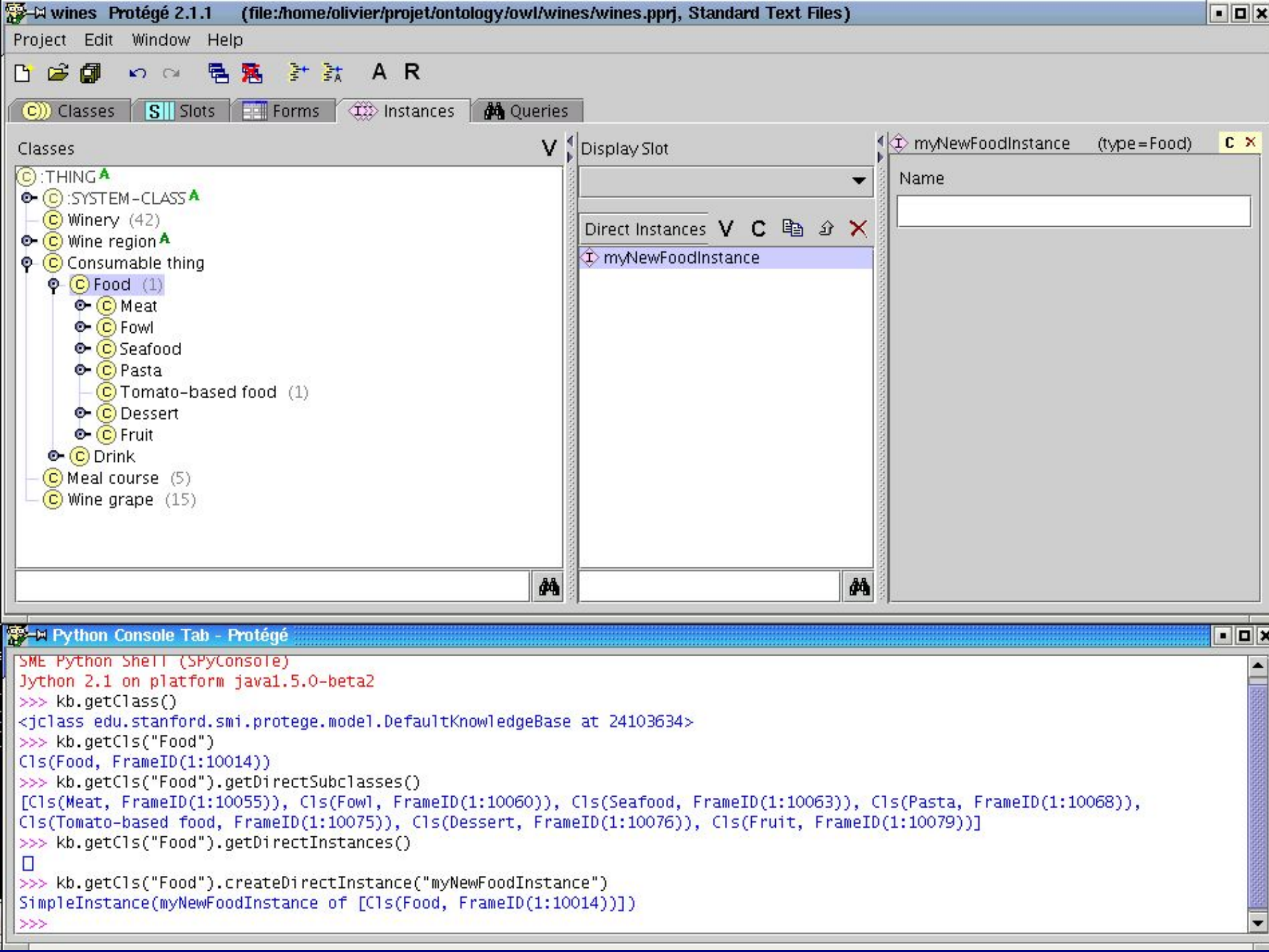
1. Get frame's attributes
2. Create frame
3. Create instances


wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Meal course
Wine grape
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Constraints
Role
Abstract
Template Slots
Name
Type
Cardinality
Other Facets
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protege.model.DefaultKnowledgeBase at 24103634>
>>>


wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Food
Meat
Fowl
Seafood
Pasta
Tomato-based food
Dessert
Fruit
Drink
Meal course
Wine grape
Food (type=:STANDARD-CLASS)
Name
Food
Documentation
Constraints
Role
Concrete
Template Slots
Name
Type
Cardinality
Other Facets
name
String
single
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protege.model.DefaultKnowledgeBase at 24103634>
>>> kb.getCls("Food")
Cls(Food, FrameID(1:10014))
>>>

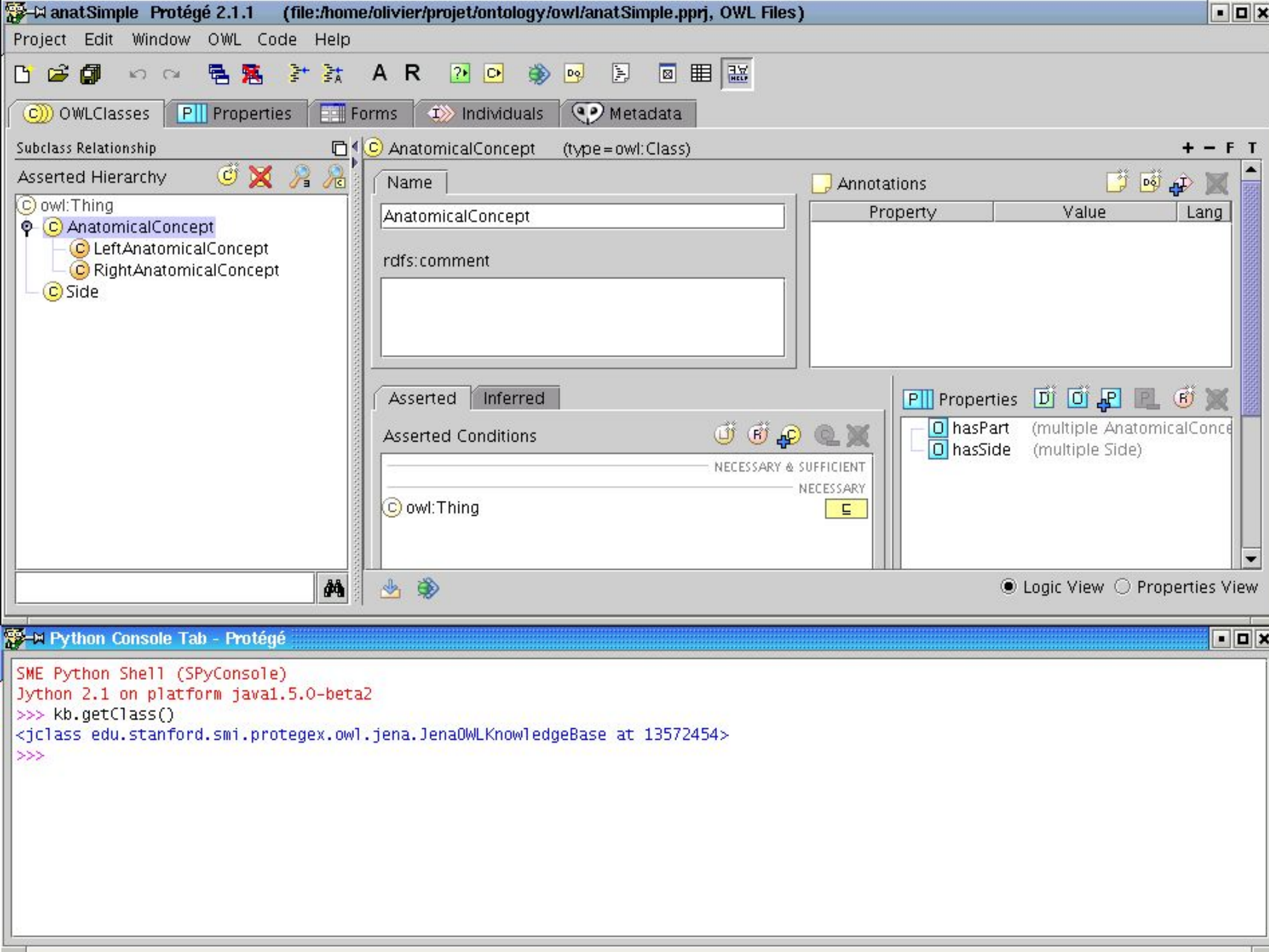

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Relationship Superclass
:THING
:SYSTEM-CLASS
Winery
Wine region
Consumable thing
Food
Meat
Fowl
Seafood
Pasta
Tomato-based food
Dessert
Fruit
Drink
Meal course
Wine grape
Food (type=:STANDARD-CLASS)
Name
Food
Documentation
Constraints
Role
Concrete
Template Slots
Name
Type
Cardinality
Other Facets
name
String
single
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protege.model.DefaultKnowledgeBase at 24103634>
>>> kb.getCls("Food")
Cls(Food, FrameID(1:10014))
>>> kb.getCls("Food").getDirectSubclasses()
[Cls(Meat, FrameID(1:10055)), Cls(Fowl, FrameID(1:10060)), Cls(Seafood, FrameID(1:10063)), Cls(Pasta, FrameID(1:10068)),
Cls(Tomato-based food, FrameID(1:10075)), Cls(Dessert, FrameID(1:10076)), Cls(Fruit, FrameID(1:10079))]
>>>

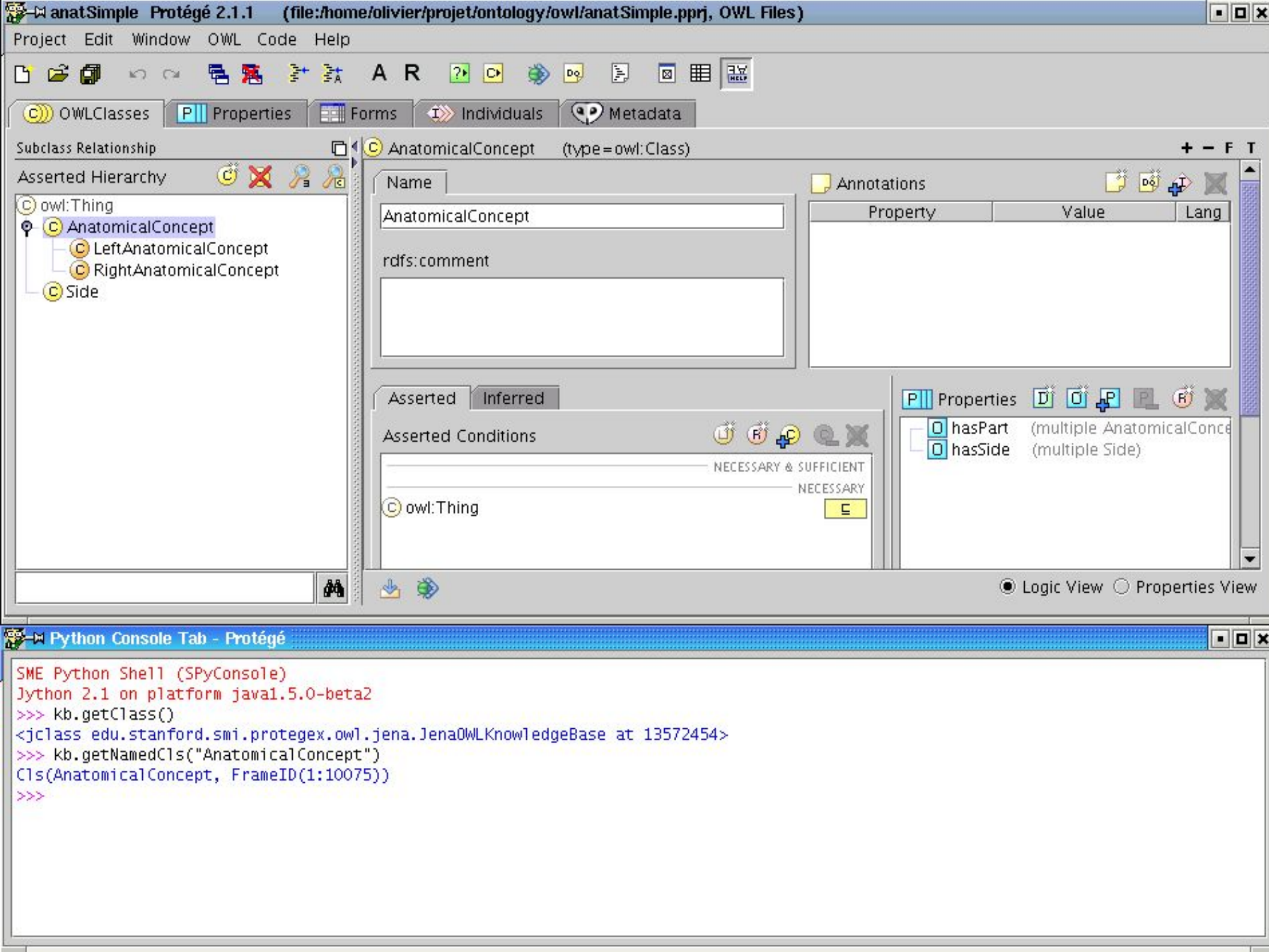

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Classes
:THING
:SYSTEM-CLASS
Winery (42)
Wine region
Consumable thing
Food
Meat
Fowl
Seafood
Pasta
Tomato-based food (1)
Dessert
Fruit
Drink
Meal course (5)
Wine grape (15)
Display Slot
Direct Instances
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protege.model.DefaultKnowledgeBase at 24103634>
>>> kb.getCls("Food")
Cls(Food, FrameID(1:10014))
>>> kb.getCls("Food").getDirectSubclasses()
[Cls(Meat, FrameID(1:10055)), Cls(Fowl, FrameID(1:10060)), Cls(Seafood, FrameID(1:10063)), Cls(Pasta, FrameID(1:10068)),
Cls(Tomato-based food, FrameID(1:10075)), Cls(Dessert, FrameID(1:10076)), Cls(Fruit, FrameID(1:10079))]
>>> kb.getCls("Food").getDirectInstances()
>>>

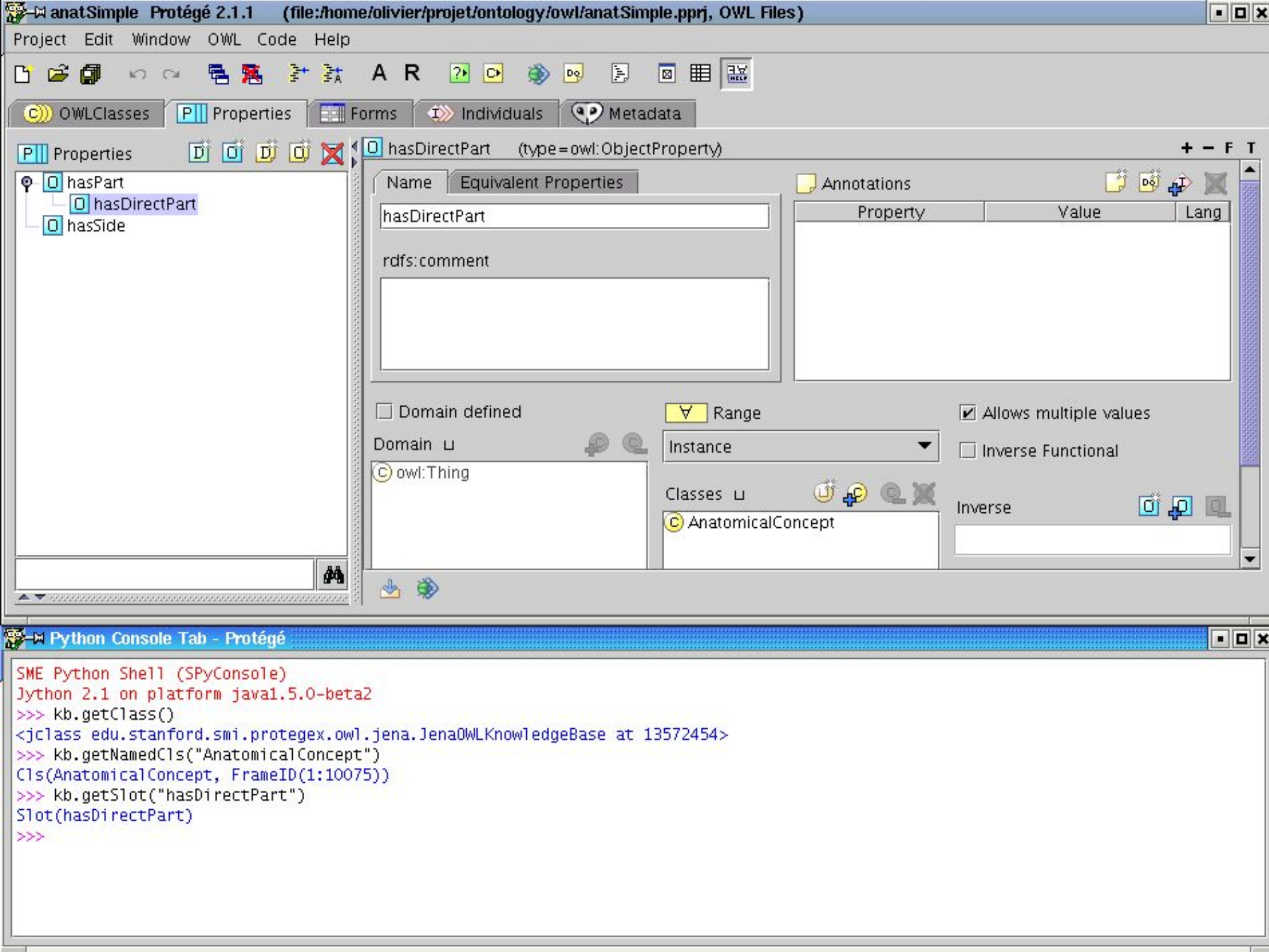

wines Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/wines/wines.pprj, Standard Text Files)
Project Edit Window Help
Classes Slots Forms Instances Queries
Classes
:THING
:SYSTEM-CLASS
Winery (42)
Wine region
Consumable thing
Food (1)
Meat
Fowl
Seafood
Pasta
Tomato-based food (1)
Dessert
Fruit
Drink
Meal course (5)
Wine grape (15)
Display Slot
Direct Instances
myNewFoodInstance
myNewFoodInstance (type=Food)
Name
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protege.model.DefaultKnowledgeBase at 24103634>
>>> kb.getCls("Food")
Cls(Food, FrameID(1:10014))
>>> kb.getCls("Food").getDirectSubclasses()
[Cls(Meat, FrameID(1:10055)), Cls(Fowl, FrameID(1:10060)), Cls(Seafood, FrameID(1:10063)), Cls(Pasta, FrameID(1:10068)),
Cls(Tomato-based food, FrameID(1:10075)), Cls(Dessert, FrameID(1:10076)), Cls(Fruit, FrameID(1:10079))]
>>> kb.getCls("Food").getDirectInstances()
>>> kb.getCls("Food").createDirectInstance("myNewFoodInstance")
SimpleInstance(myNewFoodInstance of [Cls(Food, FrameID(1:10014))])
>>>

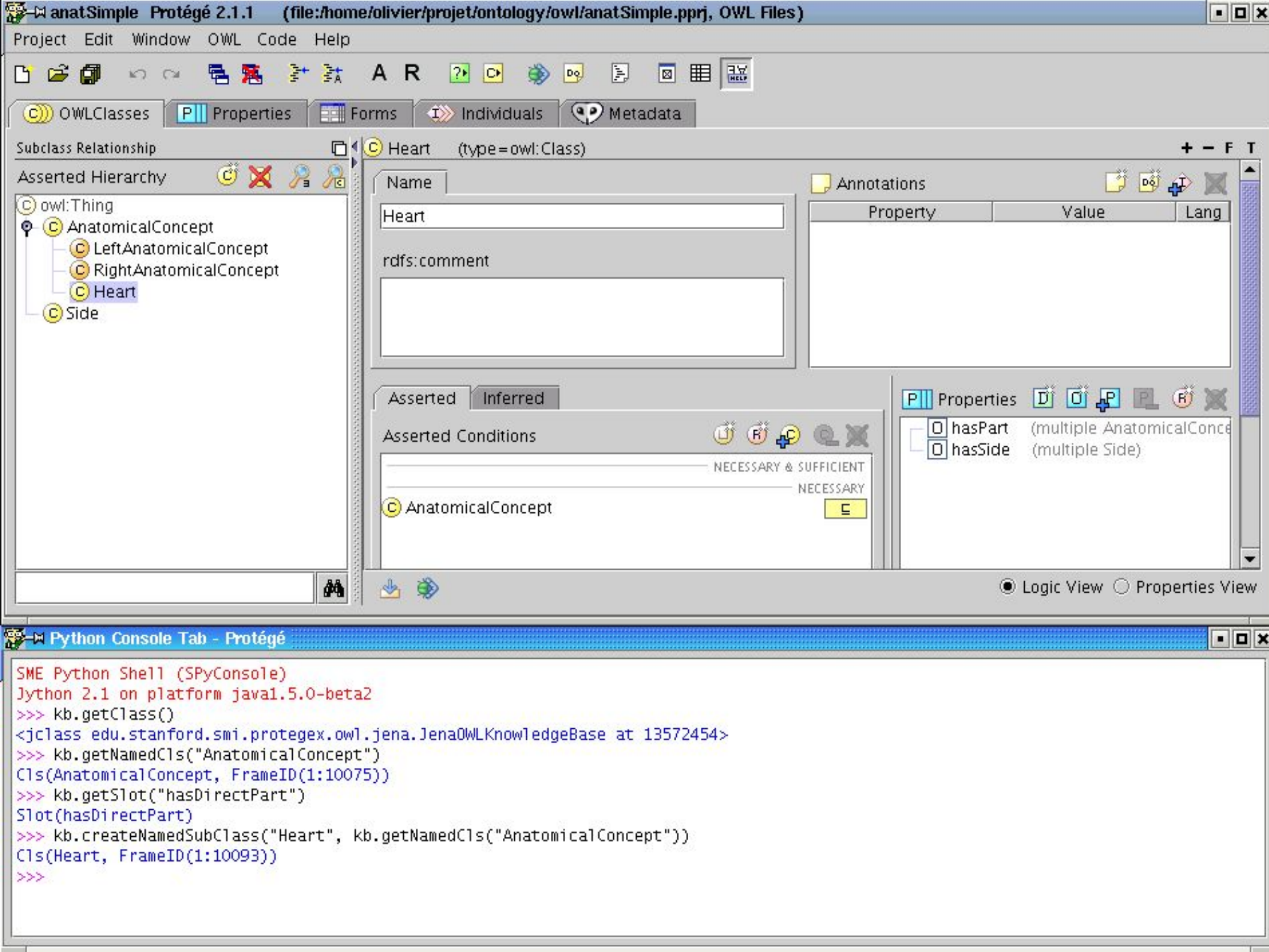
OWL

1. Get classes' attributes
2. Create class
3. Create relations


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
LeftAnatomicalConcept
RightAnatomicalConcept
Side
AnatomicalConcept (type=owl:Class)
Name
AnatomicalConcept
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
NECESSARY
owl:Thing
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protegex.owl.jena.JenaOWLKnowledgeBase at 13572454>
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
LeftAnatomicalConcept
RightAnatomicalConcept
Side
AnatomicalConcept (type=owl:Class)
Name
AnatomicalConcept
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
NECESSARY
owl:Thing
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protegex.owl.jena.JenaOWLKnowledgeBase at 13572454>
>>> kb.getNamedCls("AnatomicalConcept")
Cls(AnatomicalConcept, FrameID(1:10075))
>>>

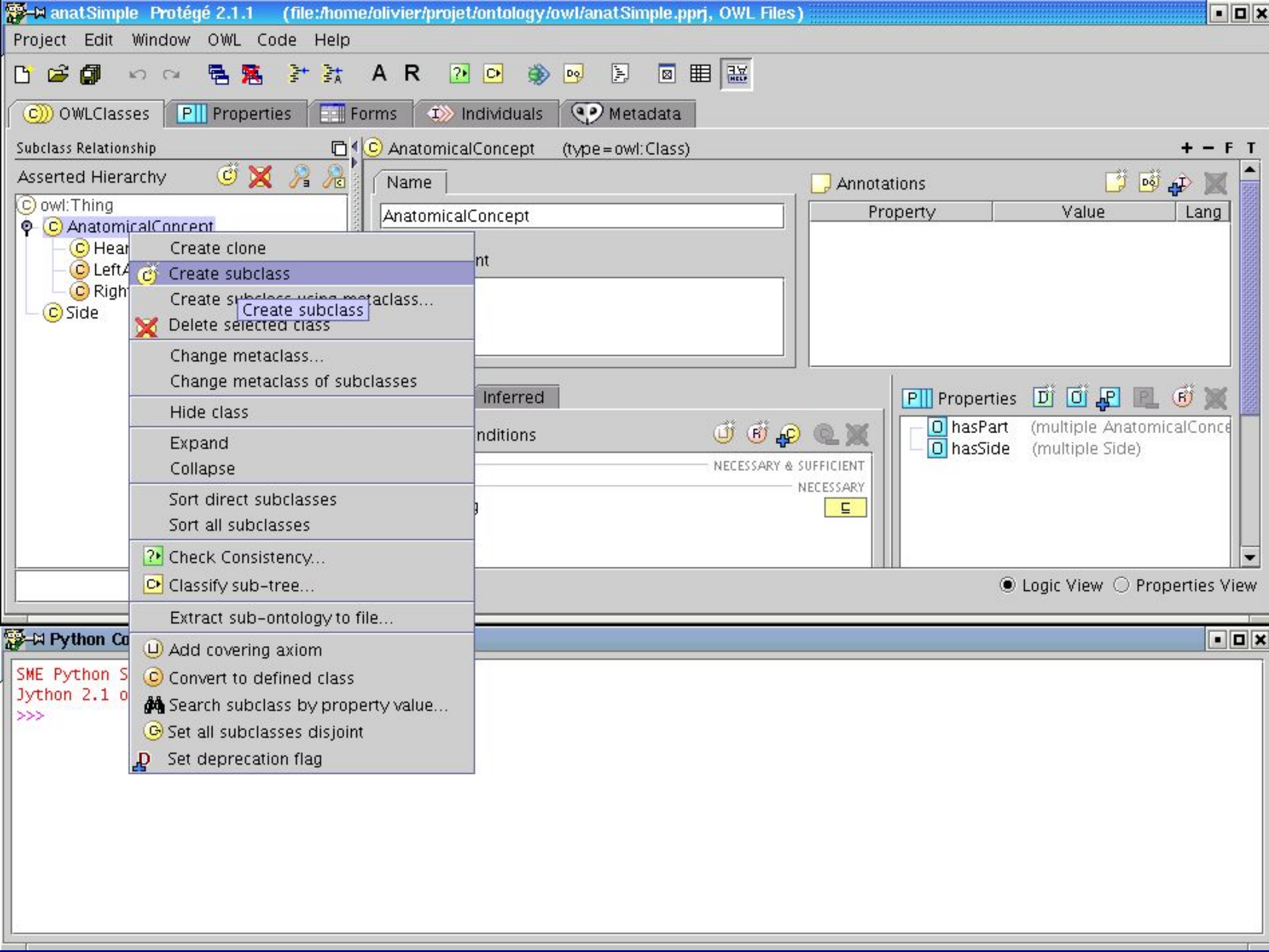

anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Properties
hasPart
hasDirectPart
hasSide
hasDirectPart (type=owl:ObjectProperty)
Name Equivalent Properties
hasDirectPart
rdfs:comment
Annotations
Property Value Lang
Domain defined
Domain
owl:Thing
Range
Instance
Classes
AnatomicalConcept
Allows multiple values
Inverse Functional
Inverse
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protegex.owl.jena.JenaOWLKnowledgeBase at 13572454>
>>> kb.getNamedCls("AnatomicalConcept")
Cls(AnatomicalConcept, FrameID(1:10075))
>>> kb.getSlot("hasDirectPart")
Slot(hasDirectPart)
>>>

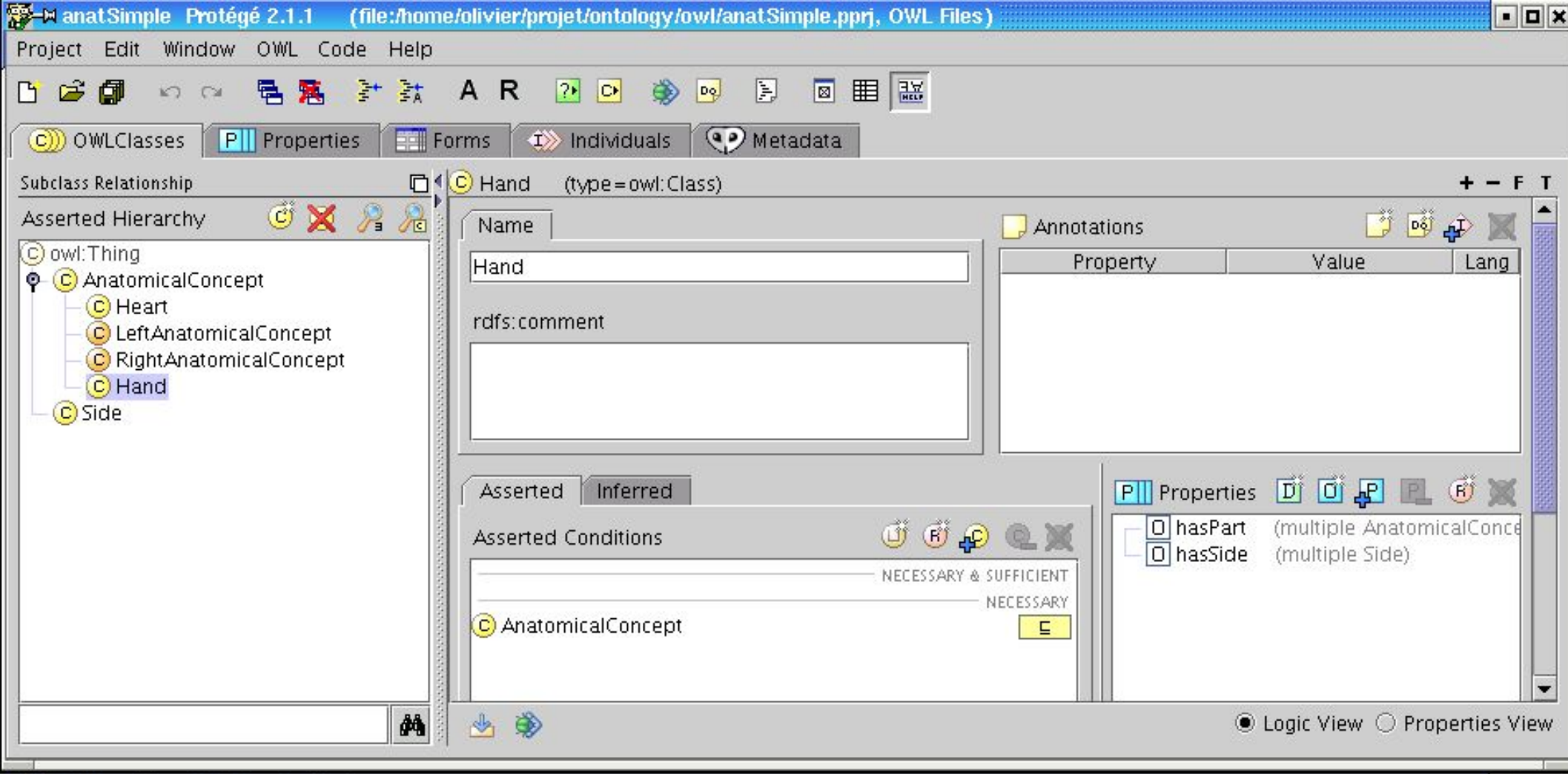

anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
LeftAnatomicalConcept
RightAnatomicalConcept
Heart
Side
Heart (type=owl:Class)
Name
Heart
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
NECESSARY
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> kb.getClass()
<jclass edu.stanford.smi.protegex.owl.jena.JenaOWLKnowledgeBase at 13572454>
>>> kb.getNamedCls("AnatomicalConcept")
Cls(AnatomicalConcept, FrameID(1:10075))
>>> kb.getSlot("hasDirectPart")
Slot(hasDirectPart)
>>> kb.createNamedSubClass("Heart", kb.getNamedCls("AnatomicalConcept"))
Cls(Heart, FrameID(1:10093))
>>>

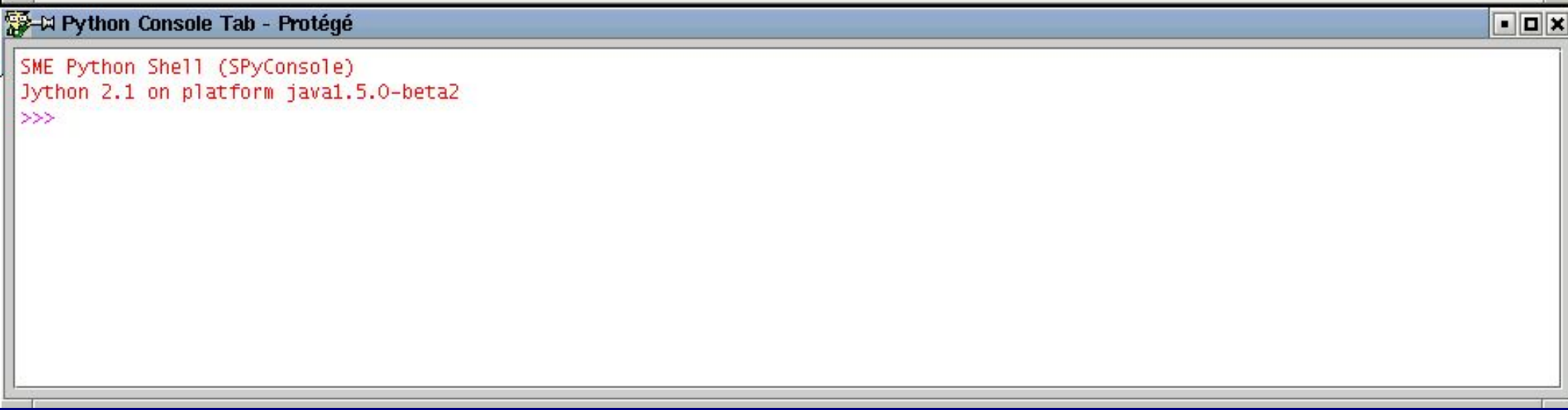
Repetitive tasks

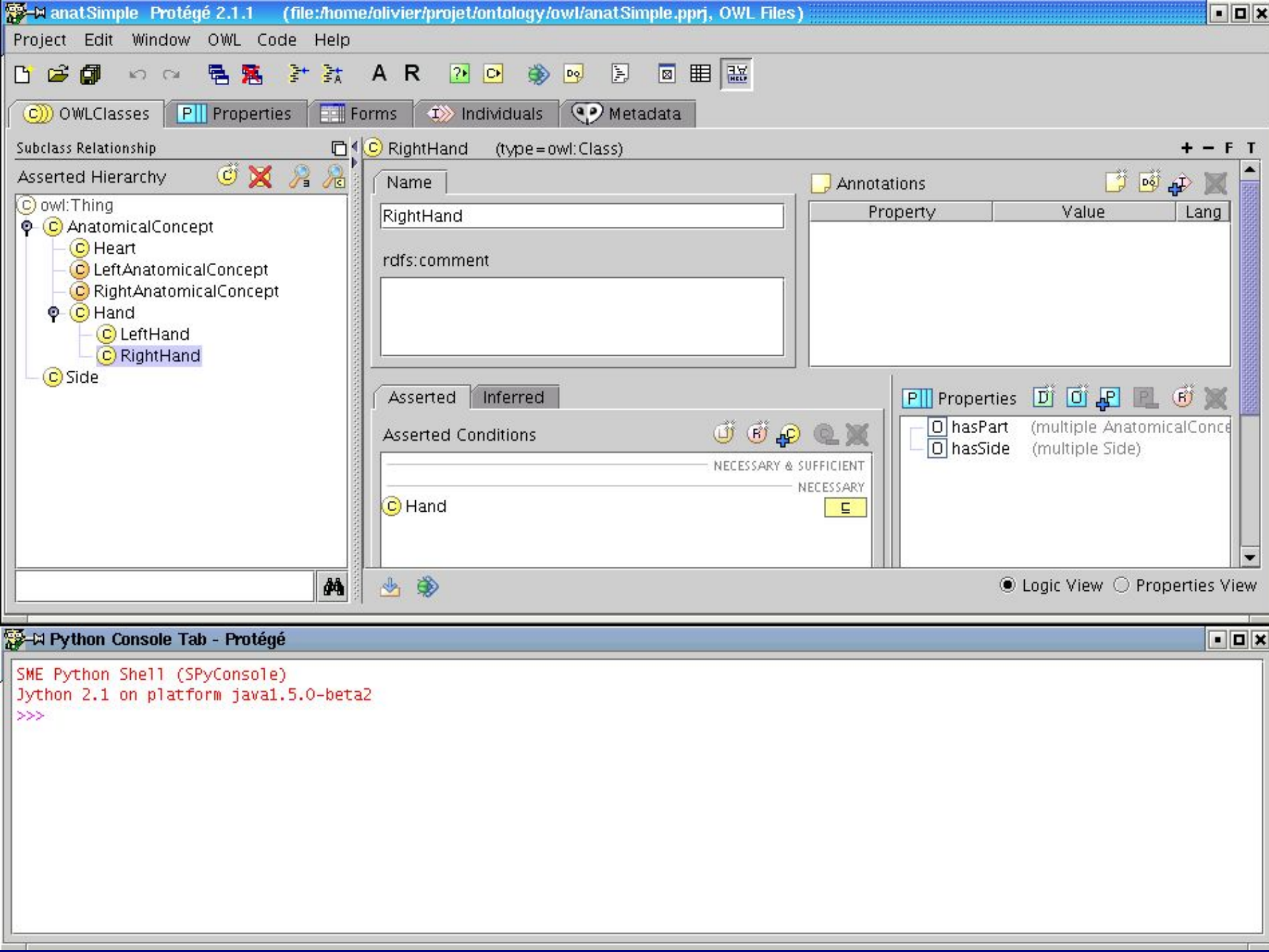
Creation of a lateralized anatomical concept: Hand

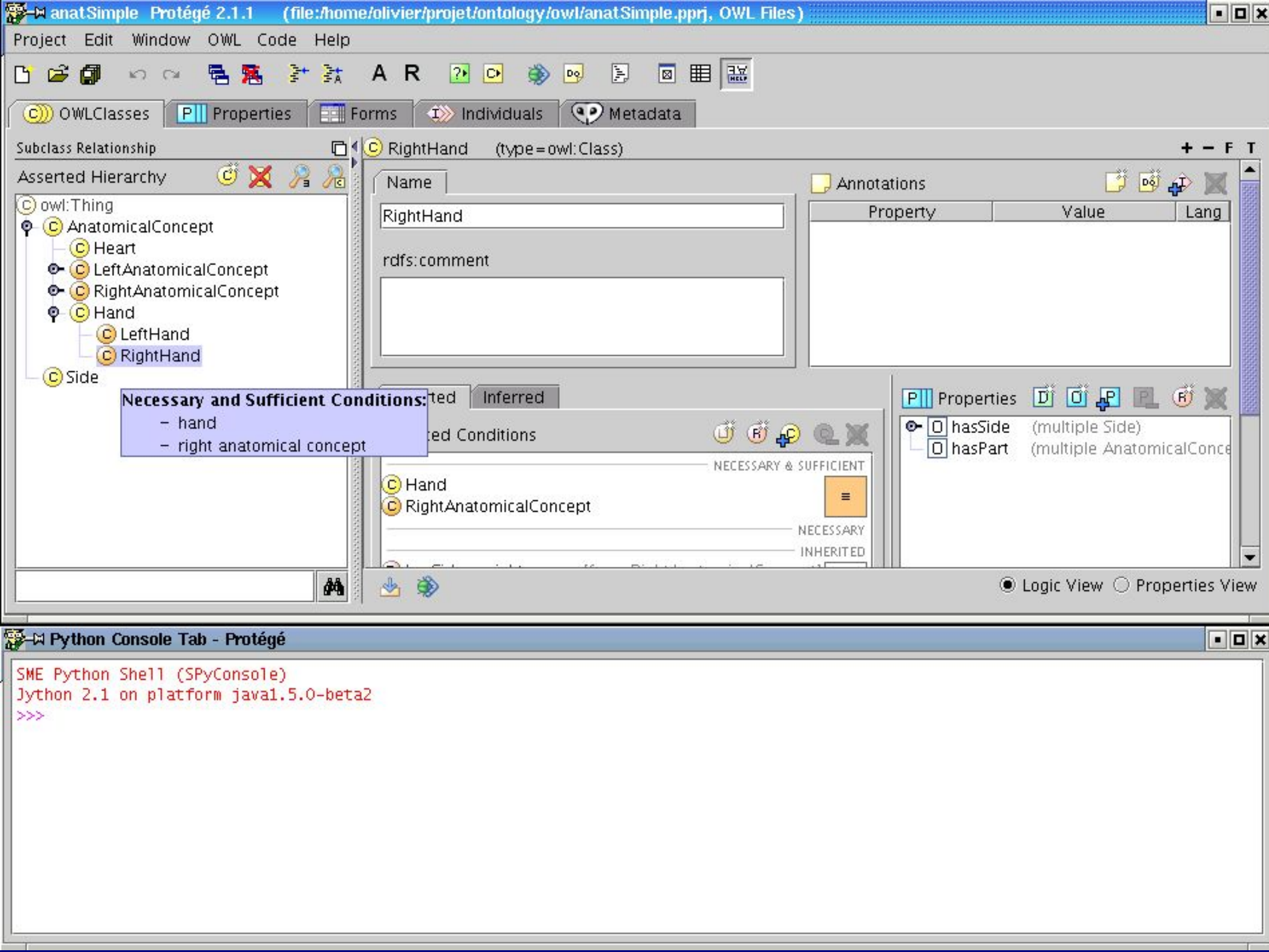
- create Hand
- create subconcepts LeftHand and RightHand
- define LeftHand = Hand on the LeftSide
- Hand: either LeftHand or RightHand
- LeftHand and RightHand are disjoint

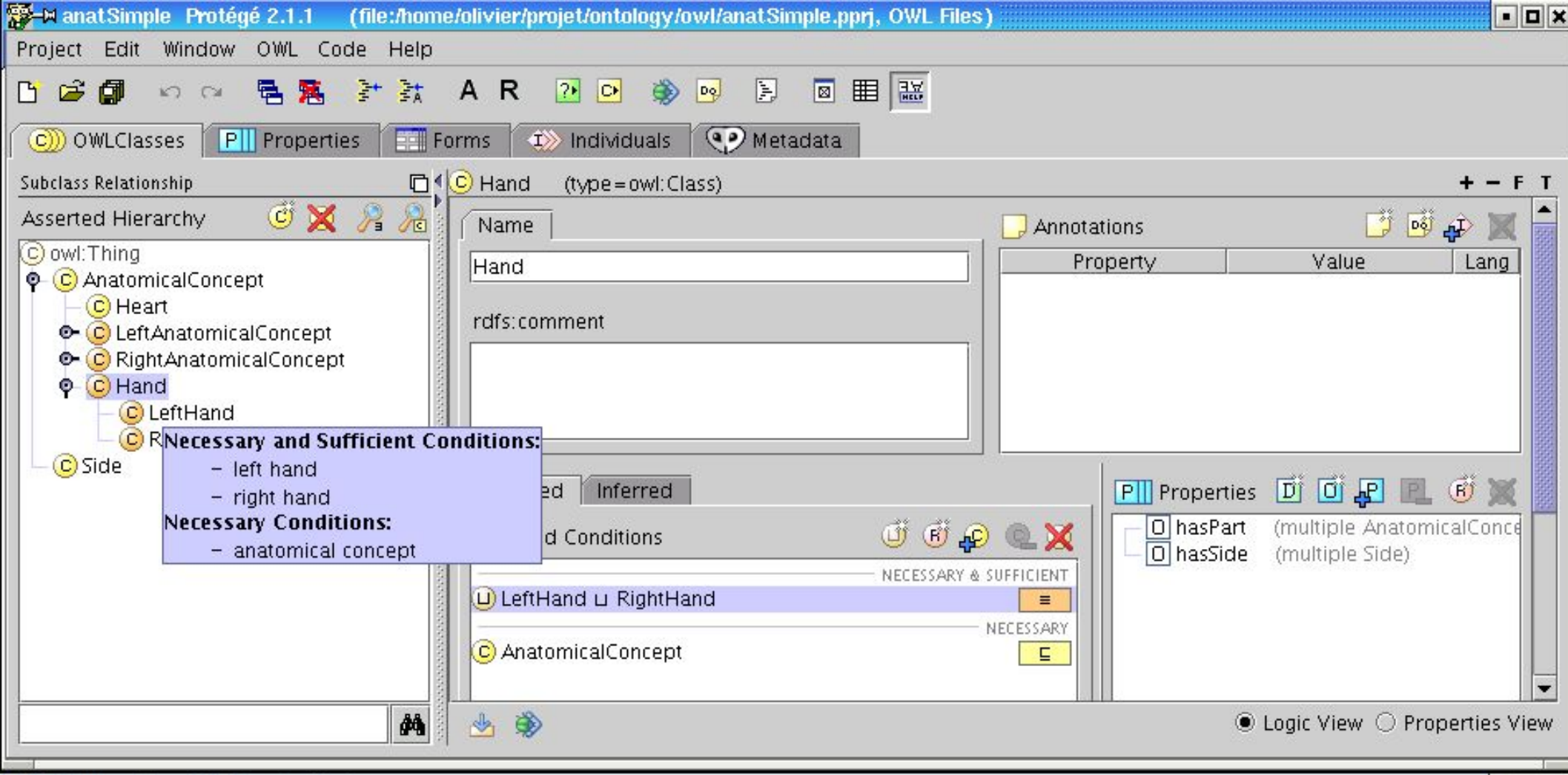

anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Side
AnatomicalConcept (type=owl:Class)
Name
AnatomicalConcept
Annotations
Property Value Lang
Inferred
NECESSARY & SUFFICIENT
NECESSARY
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View Properties View
Create clone
Create subclass
Create subclass using metaclass...
Delete selected class
Change metaclass...
Change metaclass of subclasses
Hide class
Expand
Collapse
Sort direct subclasses
Sort all subclasses
Check Consistency...
Classify sub-tree...
Extract sub-ontology to file...
Add covering axiom
Convert to defined class
Search subclass by property value...
Set all subclasses disjoint
Set deprecation flag
Python Co
SME Python S
Jython 2.1 o
>>>

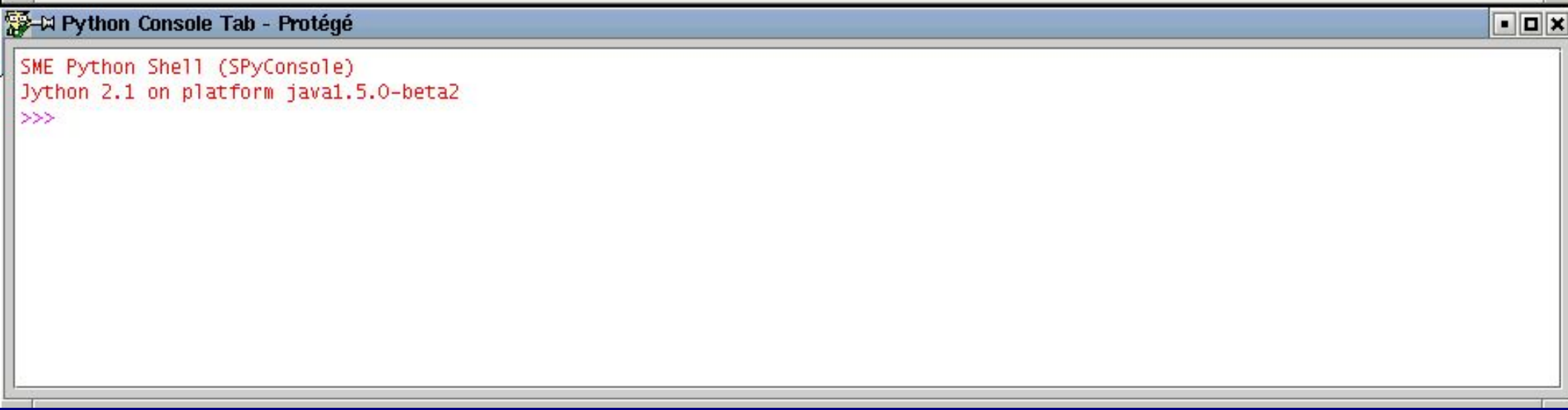

anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
Side
Hand (type=owl:Class)
Name
Hand
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
NECESSARY
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View


Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
LeftHand
RightHand
Side
RightHand (type=owl:Class)
Name
RightHand
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
NECESSARY
Hand
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
LeftHand
RightHand
Side
Necessary and Sufficient Conditions:
- hand
- right anatomical concept
RightHand (type=owl:Class)
Name
RightHand
rdfs:comment
Annotations
Property
Value
Lang
Inferred
NECESSARY & SUFFICIENT
Hand
RightAnatomicalConcept
NECESSARY
INHERITED
Properties
hasSide (multiple Side)
hasPart (multiple AnatomicalConce
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
LeftHand
Side
Necessary and Sufficient Conditions:
– left hand
– right hand
Necessary Conditions:
– anatomical concept
Hand (type=owl:Class)
Name
Hand
rdfs:comment
Annotations
Property
Value
Lang
Inferred
NECESSARY & SUFFICIENT
LeftHand ⊔ RightHand
NECESSARY
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View


Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>>

Repetitive tasks

createLateralizedConcept(“Hand”, “Anat”):

c = createConcept(“Hand”, “AnatomicalConcept”)

lc = createConcept(“LeftHand”, “Hand”)

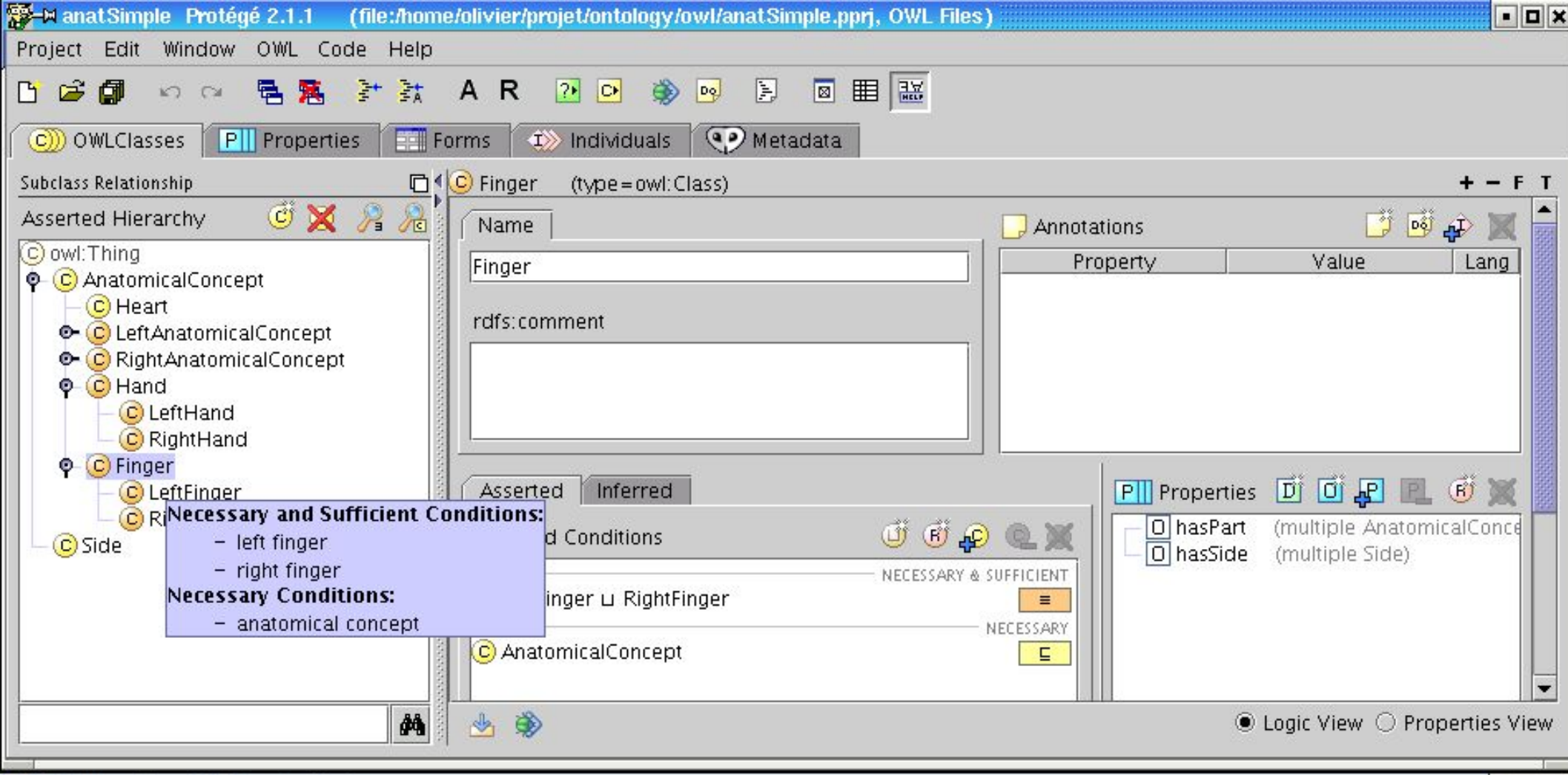
rc = createConcept(“RightHand”, “Hand”)

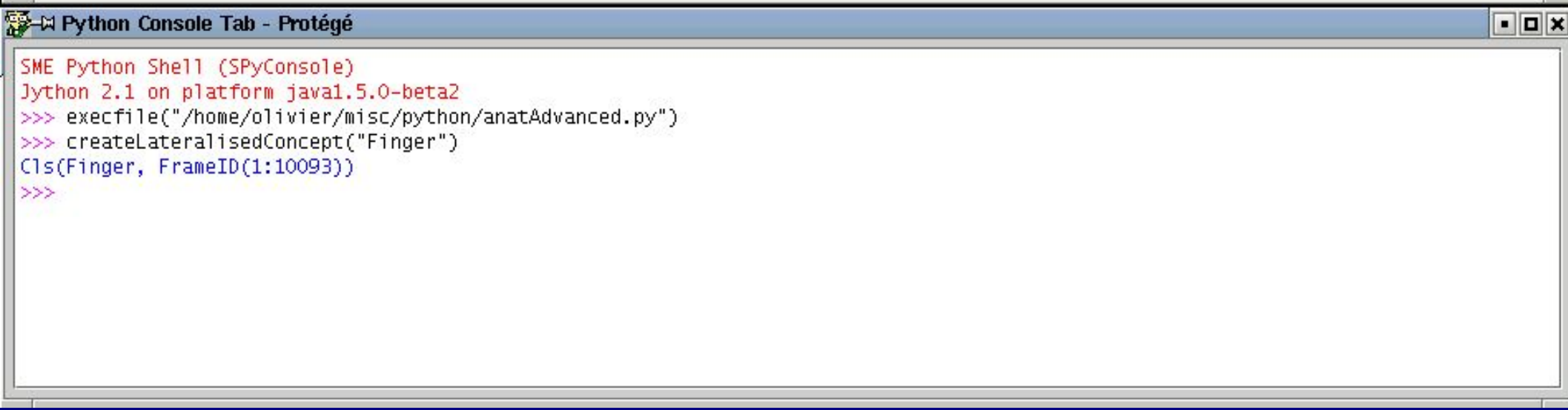
define c = lc or rc

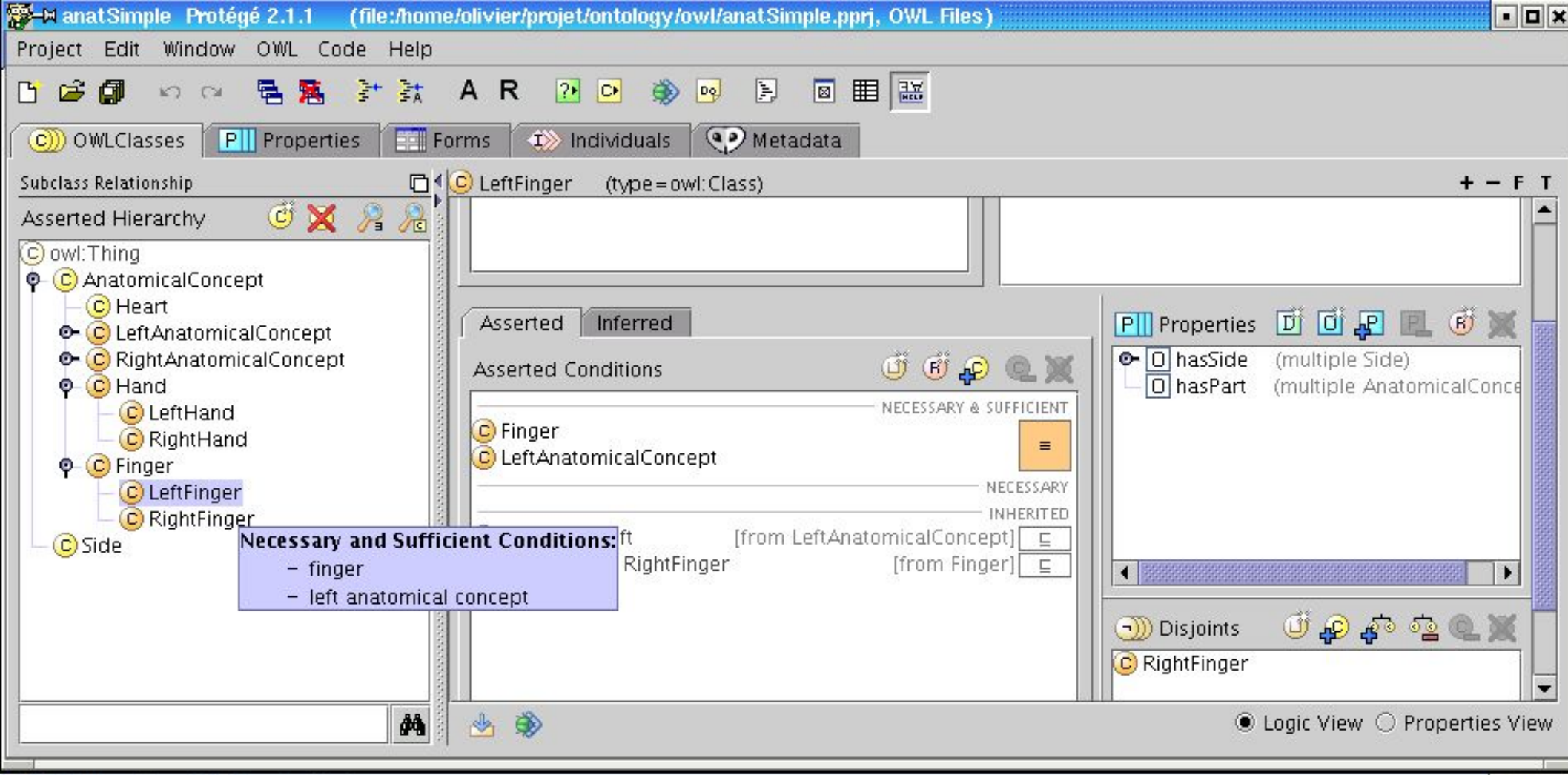
define lc = c and LeftAnatomicalConcept

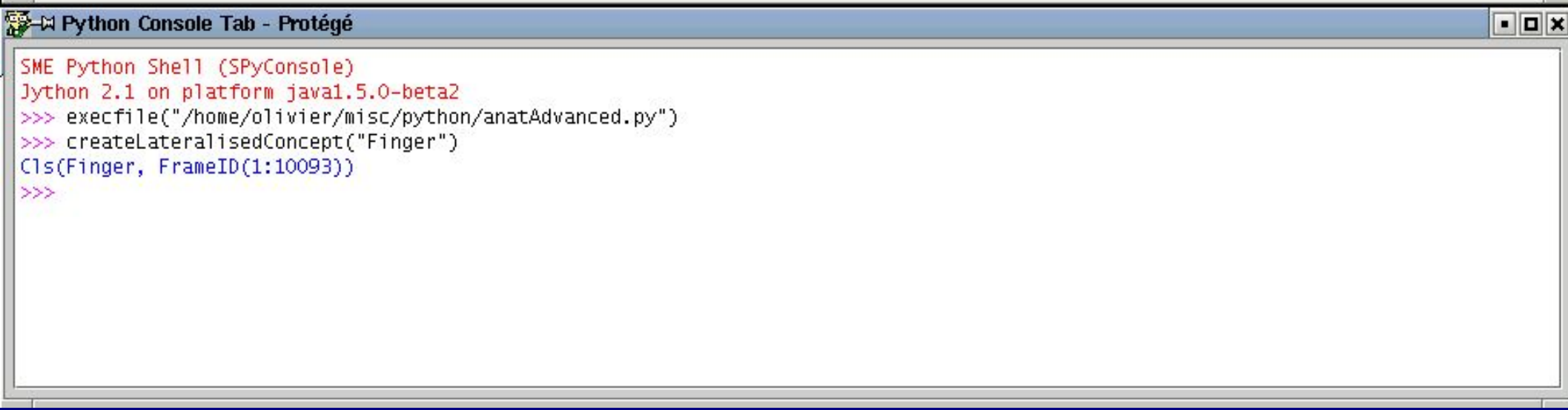
define rc = c and RightAnatomicalConcept

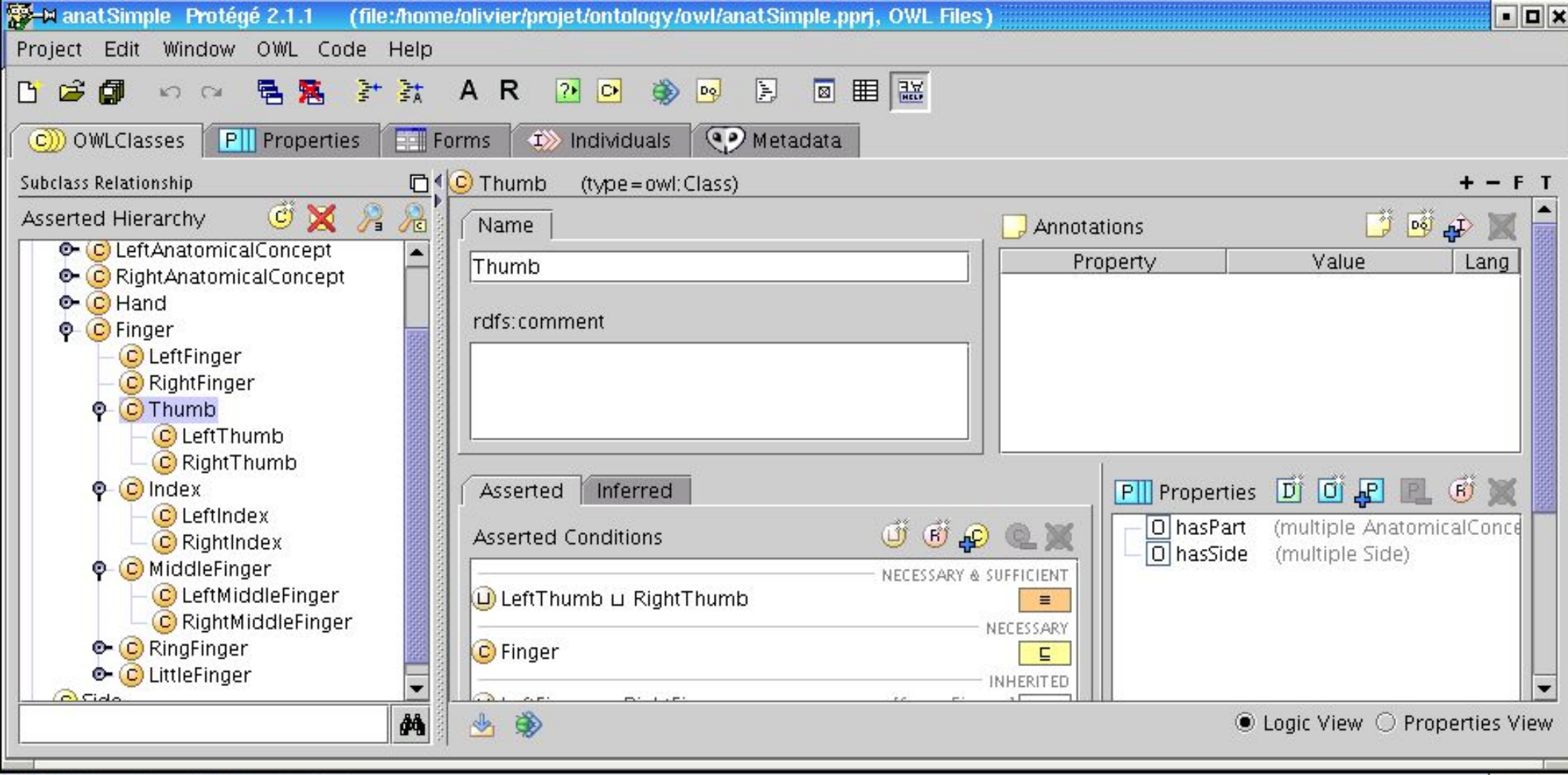
make lc and rc disjoint

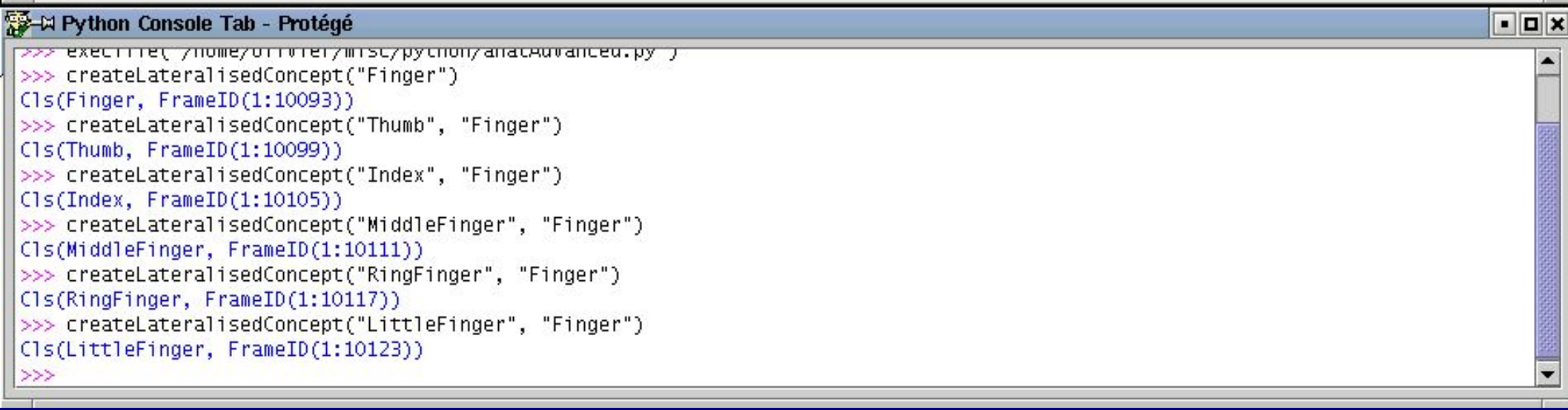

anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
LeftHand
RightHand
Finger
LeftFinger
Side
Necessary and Sufficient Conditions:
- left finger
- right finger
Necessary Conditions:
- anatomical concept
Finger (type=owl:Class)
Name
Finger
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
NECESSARY & SUFFICIENT
NECESSARY
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View

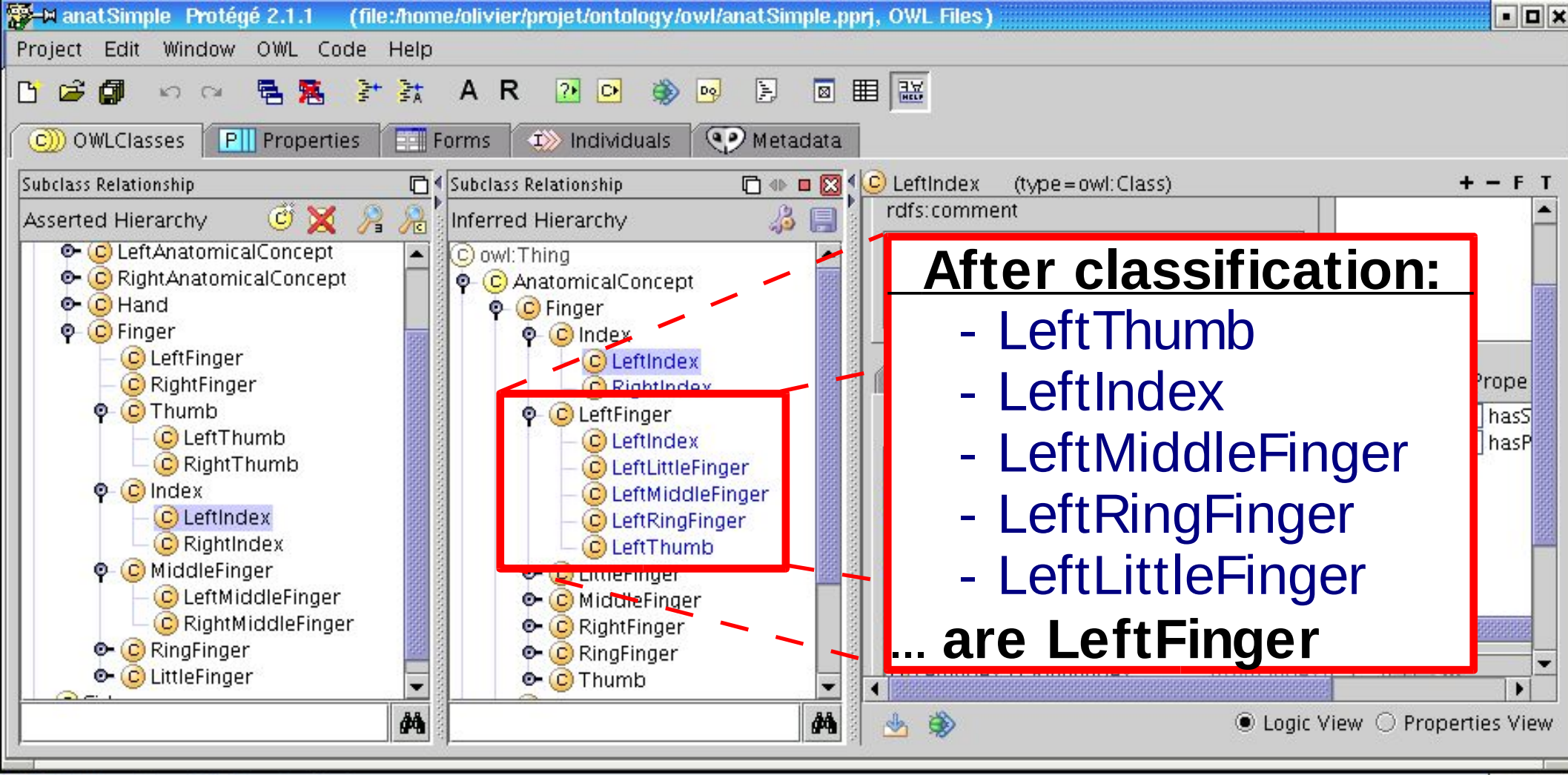

Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> execfile("/home/olivier/misc/python/anatAdvanced.py")
>>> createLateralisedConcept("Finger")
Cls(Finger, FrameID(1:10093))
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
LeftHand
RightHand
Finger
LeftFinger
RightFinger
Side
Necessary and Sufficient Conditions:
- finger
- left anatomical concept
LeftFinger (type=owl:Class)
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
Finger
LeftAnatomicalConcept
NECESSARY
INHERITED
ft [from LeftAnatomicalConcept]
RightFinger [from Finger]
Properties
hasSide (multiple Side)
hasPart (multiple AnatomicalConce
Disjoints
RightFinger
Logic View
Properties View


Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> execfile("/home/olivier/misc/python/anatAdvanced.py")
>>> createLateralisedConcept("Finger")
Cls(Finger, FrameID(1:10093))
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
LeftAnatomicalConcept
RightAnatomicalConcept
Hand
Finger
LeftFinger
RightFinger
Thumb
LeftThumb
RightThumb
Index
LeftIndex
RightIndex
MiddleFinger
LeftMiddleFinger
RightMiddleFinger
RingFinger
LittleFinger
Thumb (type=owl:Class)
Name
Thumb
rdfs:comment
Annotations
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
LeftThumb ⊔ RightThumb
NECESSARY
Finger
INHERITED
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View


Python Console Tab - Protégé
>>> createLateralisedConcept("Finger")
Cls(Finger, FrameID(1:10093))
>>> createLateralisedConcept("Thumb", "Finger")
Cls(Thumb, FrameID(1:10099))
>>> createLateralisedConcept("Index", "Finger")
Cls(Index, FrameID(1:10105))
>>> createLateralisedConcept("MiddleFinger", "Finger")
Cls(MiddleFinger, FrameID(1:10111))
>>> createLateralisedConcept("RingFinger", "Finger")
Cls(RingFinger, FrameID(1:10117))
>>> createLateralisedConcept("LittleFinger", "Finger")
Cls(LittleFinger, FrameID(1:10123))
>>>

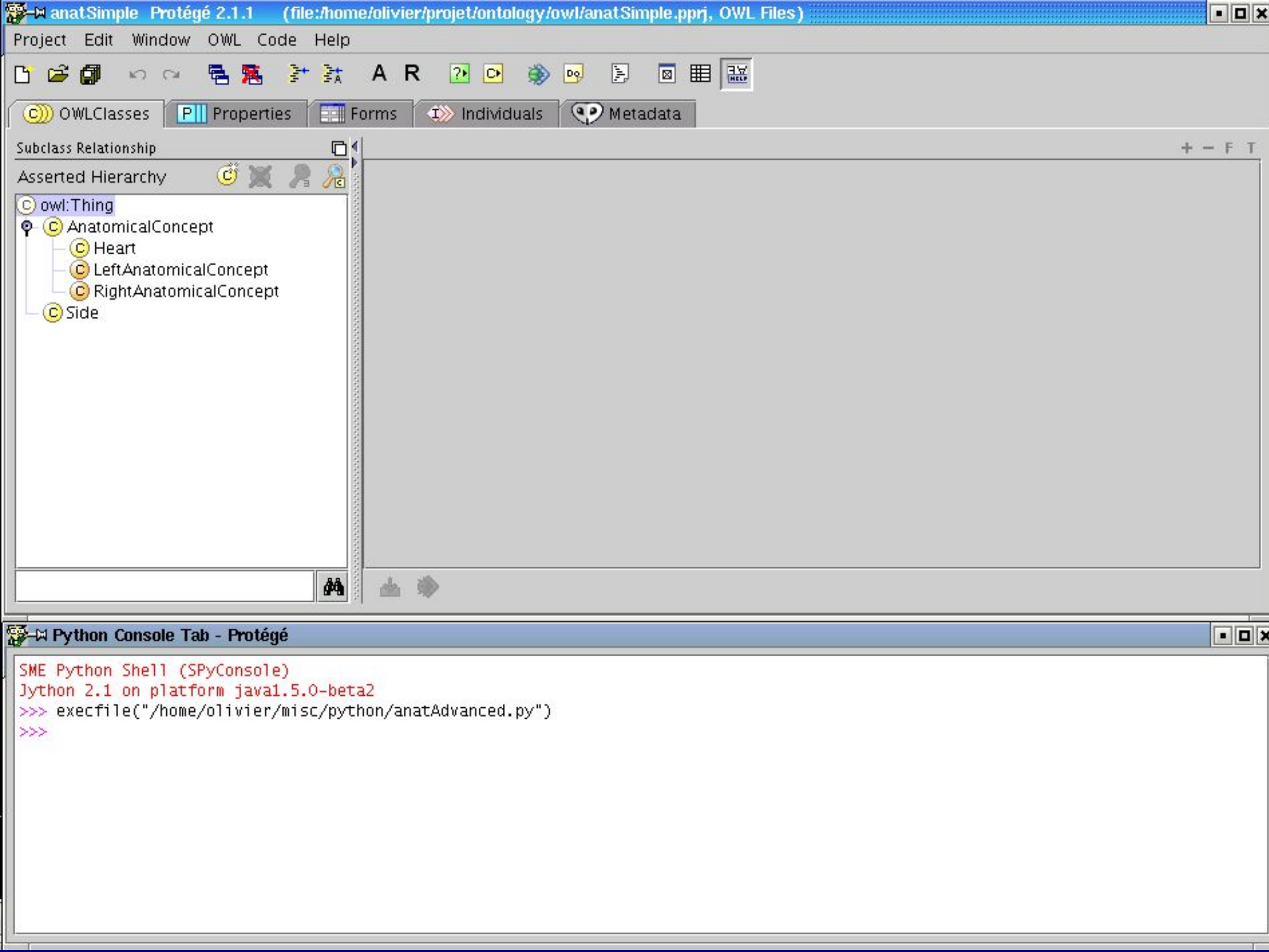


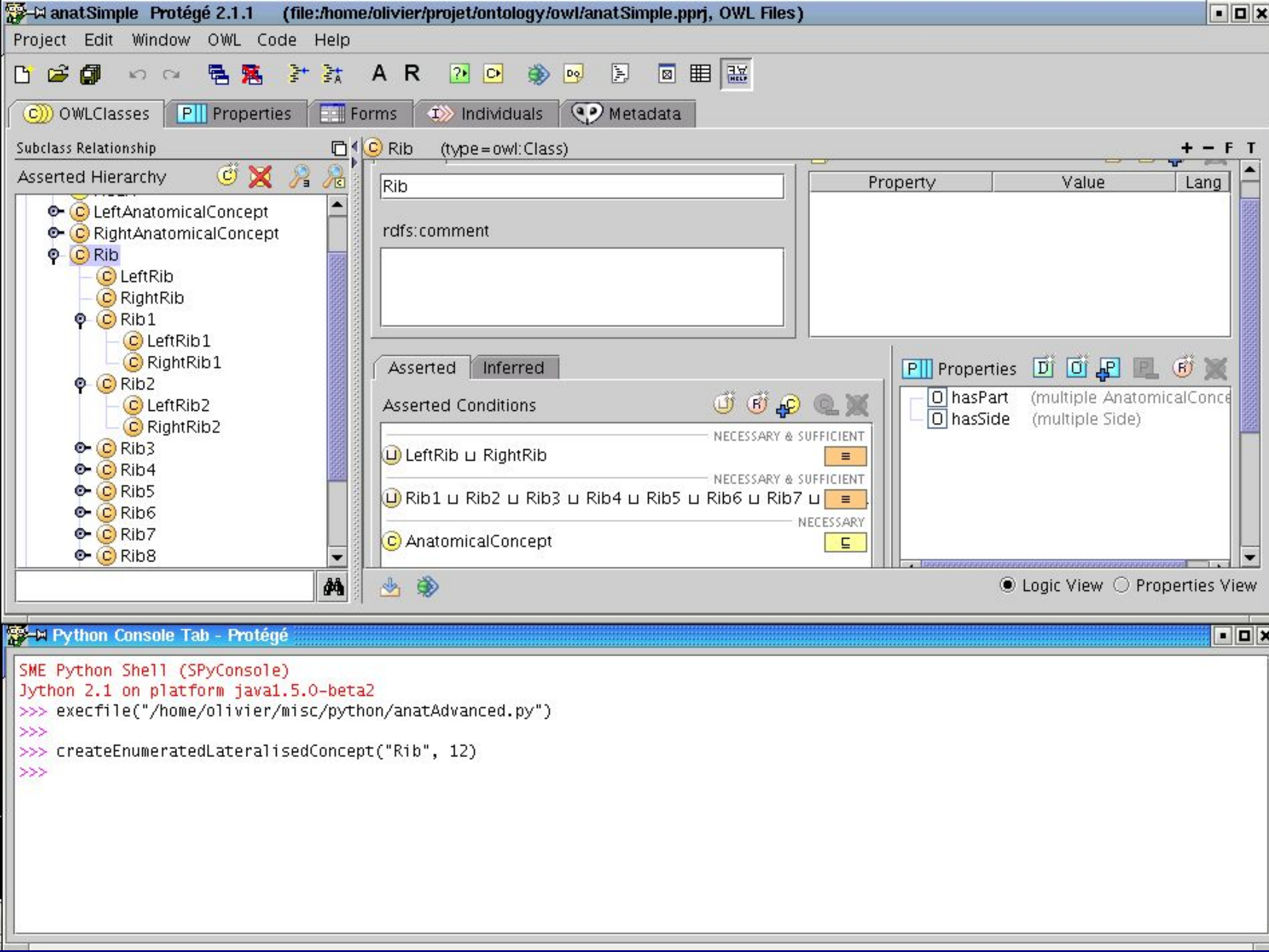
Python Console Tab - Protégé

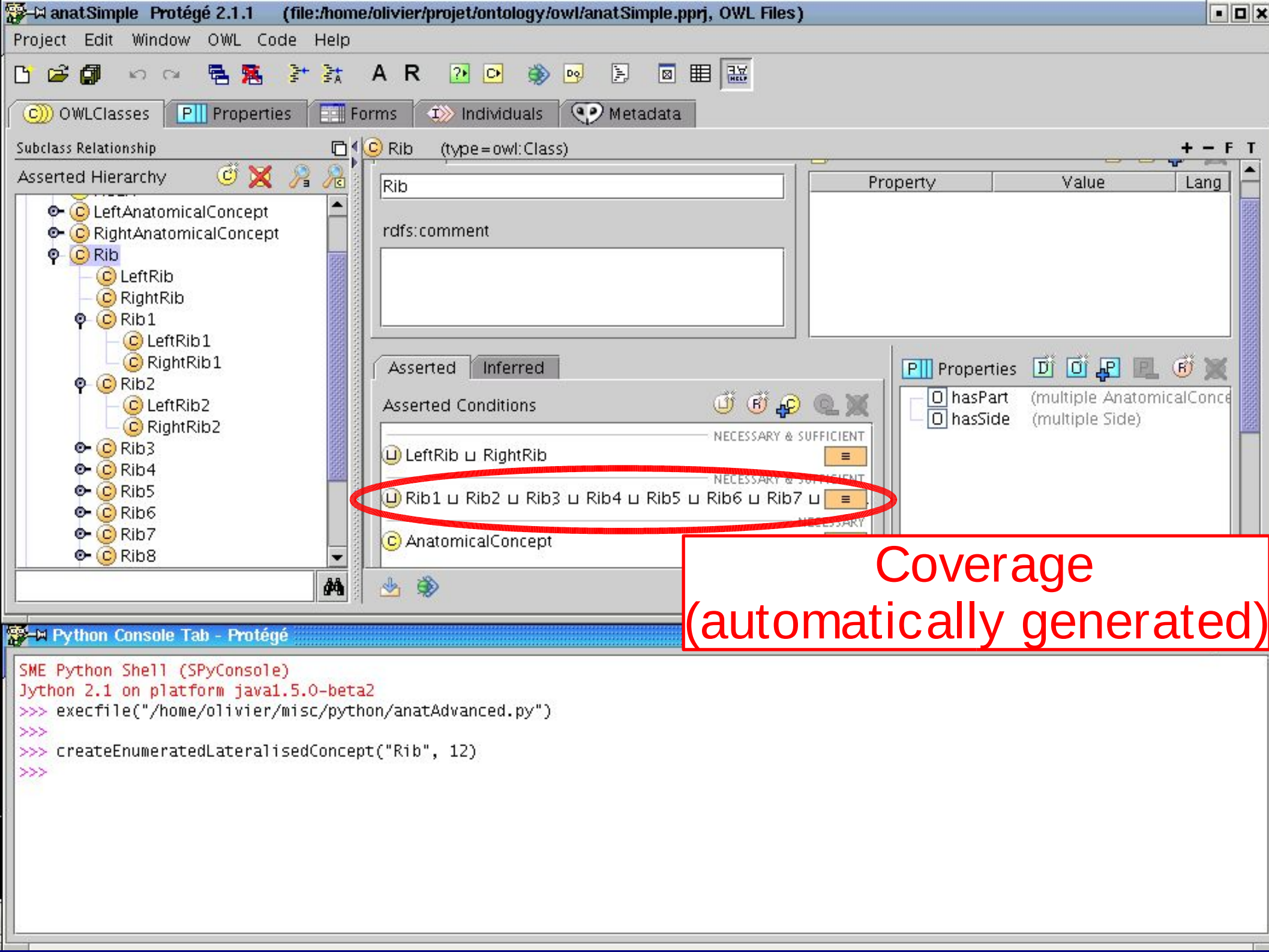
```
>>> execfile('/home/olivier/misc/python/anatAdvanced.py')
>>> createLateralisedConcept("Finger")
Cls(Finger, FrameID(1:10093))
>>> createLateralisedConcept("Thumb", "Finger")
Cls(Thumb, FrameID(1:10099))
>>> createLateralisedConcept("Index", "Finger")
Cls(Index, FrameID(1:10105))
>>> createLateralisedConcept("MiddleFinger", "Finger")
Cls(MiddleFinger, FrameID(1:10111))
>>> createLateralisedConcept("RingFinger", "Finger")
Cls(RingFinger, FrameID(1:10117))
>>> createLateralisedConcept("LittleFinger", "Finger")
Cls(LittleFinger, FrameID(1:10123))
>>>
```

Enumerations

1. Vertebrae
2. Ribs (lateralized !)
3. Muscles


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
owl:Thing
AnatomicalConcept
Heart
LeftAnatomicalConcept
RightAnatomicalConcept
Side
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> execfile("/home/olivier/misc/python/anatAdvanced.py")
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
LeftAnatomicalConcept
RightAnatomicalConcept
Rib
LeftRib
RightRib
Rib1
LeftRib1
RightRib1
Rib2
LeftRib2
RightRib2
Rib3
Rib4
Rib5
Rib6
Rib7
Rib8
Rib (type=owl:Class)
Rib
rdfs:comment
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
LeftRib ⊔ RightRib
NECESSARY & SUFFICIENT
Rib1 ⊔ Rib2 ⊔ Rib3 ⊔ Rib4 ⊔ Rib5 ⊔ Rib6 ⊔ Rib7 ⊔
NECESSARY
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Logic View
Properties View
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> execfile("/home/olivier/misc/python/anatAdvanced.py")
>>>
>>> createEnumeratedLateralisedConcept("Rib", 12)
>>>


anatSimple Protégé 2.1.1 (file:/home/olivier/projet/ontology/owl/anatSimple.pprj, OWL Files)
Project Edit Window OWL Code Help
OWLClasses Properties Forms Individuals Metadata
Subclass Relationship
Asserted Hierarchy
LeftAnatomicalConcept
RightAnatomicalConcept
Rib
LeftRib
RightRib
Rib1
LeftRib1
RightRib1
Rib2
LeftRib2
RightRib2
Rib3
Rib4
Rib5
Rib6
Rib7
Rib8
Rib (type=owl:Class)
Rib
rdfs:comment
Property
Value
Lang
Asserted
Inferred
Asserted Conditions
NECESSARY & SUFFICIENT
LeftRib ⊔ RightRib
Rib1 ⊔ Rib2 ⊔ Rib3 ⊔ Rib4 ⊔ Rib5 ⊔ Rib6 ⊔ Rib7 ⊔
AnatomicalConcept
Properties
hasPart (multiple AnatomicalConce
hasSide (multiple Side)
Coverage
(automatically generated)
Python Console Tab - Protégé
SME Python Shell (SPyConsole)
Jython 2.1 on platform java1.5.0-beta2
>>> execfile("/home/olivier/misc/python/anatAdvanced.py")
>>>
>>> createEnumeratedLateralisedConcept("Rib", 12)
>>>

Dependencies

1. Ex: Wall of Heart
   - Heart = WallOfHeart, Septum, 4 cavities
   - Heart = LeftAtrium, RightAtrium, LeftVentricle, RightVentricle
2. WallOfLeftAtrium
   - *constitutionalPartOf* LeftAtrium
   - *regionalPartOf* WallOfHeart
3. Epicardium, Myocardium, Endocardium
   - *constitutionalPartOf* WallOfLeftAtrium
   - *regionalPartOf* Epicardium

Dependencies

- 16 composed concepts
- 32 relations of direct composition

- You are lucky if you don't forget one
- If you do, enjoy the debuging
- Not scalable
  - no neighborhood relationships
  - {anterior, posterior, lateral} + {inferior, superior} parts of LeftAtrium

Ontology maintenance

1. Make specific functions on the fly
2. Reuse functions
3. Dynamically insert / remove java listeners
4. Take advantage of all the existing Java libraries (web services, ...)

Lessons learned

- JOT is usefull :-)
  - higher level functions
  - from extensional to intensional description
- Domain-independant but language-specific macros
- Domain-dependant but language independent functions
  (reuse functions from 1 according to the language)

Conclusion

1. Direct calls to the Protégé API => no limitations
2. Jython => power of Python + Java
3. Code reuse allow to hide the low-level Protégé API