Basic Multi-Level Modeling in RDF and SHACL

— Tutorial and Task Assignment —

Sebastian Gruber, Bernd Neumayr

JOHANNES KEPLER UNIVERSITY LINZ

salzburgresearch

LUDWIG BOLTZMANN INSTITUTE Digital Health and Prevention

Contents

Use Case: Just-in-Time Adaptive Intervention (JITAI)

- The just-in-time adaptive intervention (JITAI) is an intervention design to support health behavior changes for, e.g., physical inactivity, …

(Nahum-Shani et al., 2018)

"At 2021-07-15 12:10, Jane is available for intervention. She has been sedentary for the last 30 minutes and the weather is fine. The system suggests to take a walk to the nearby lake to have lunch at the lakeside restaurant. Jane follows the suggestion and makes 3000 steps in the next hour." (Gruber et al., 2021)

- This intervention design aims to
  - "provide the right type/amount of support",
  - "at the right time",
  - "by adapting to an individual's changing internal and contextual state".

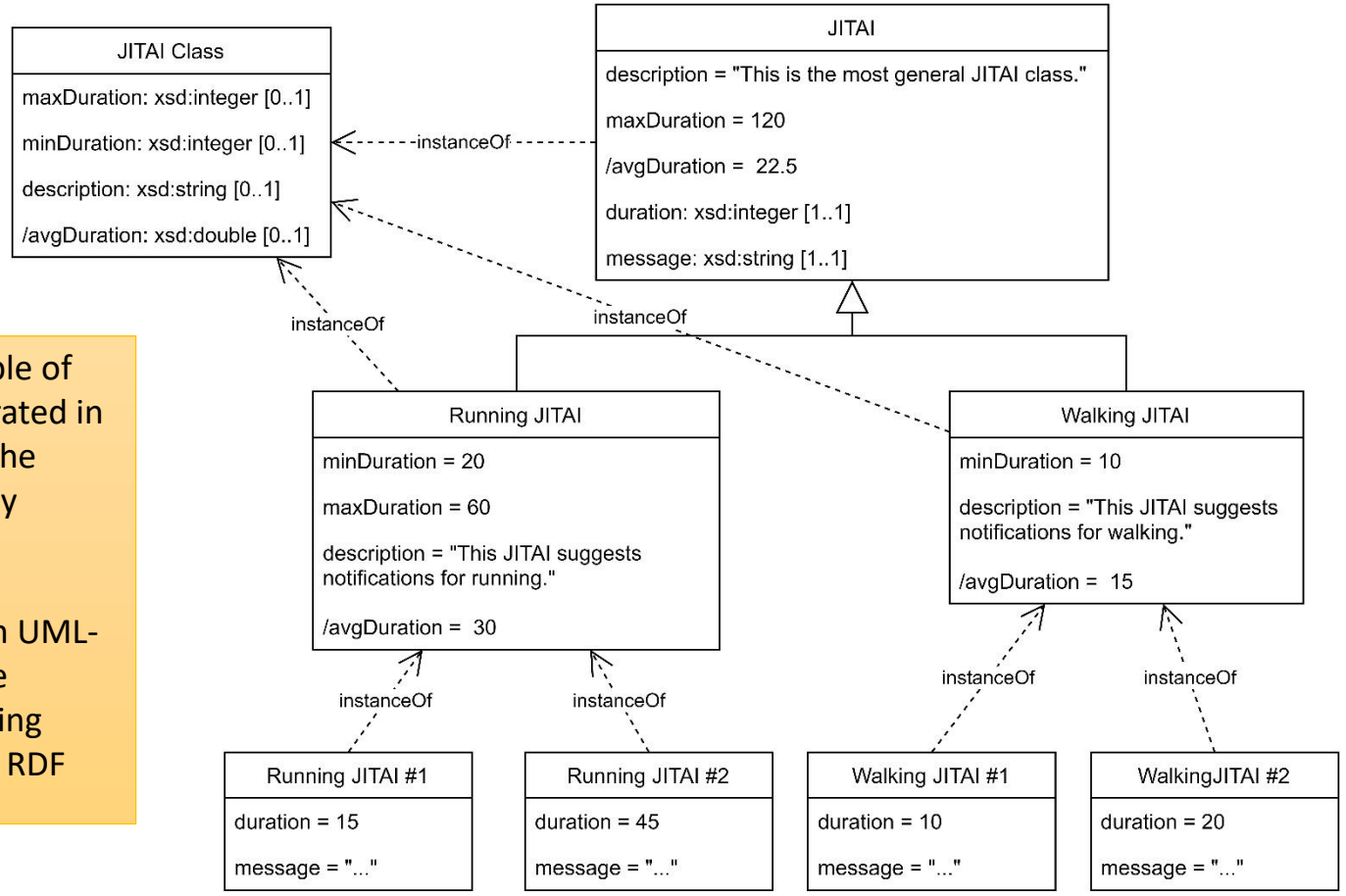
(Nahum-Shani et al., 2018)

Basic Multi-Level Modeling in RDF/SHACL

- The purpose of this tutorial is to provide students a first contact with basics of multi-level modeling (MLM) in RDF and SHACL.
- We assume that students have a solid knowledge of RDF, SPARQL and SHACL.

- We only consider 3 classification levels (individuals, classes, meta classes) and focus on the use of powertypes for metamodeling in SHACL.

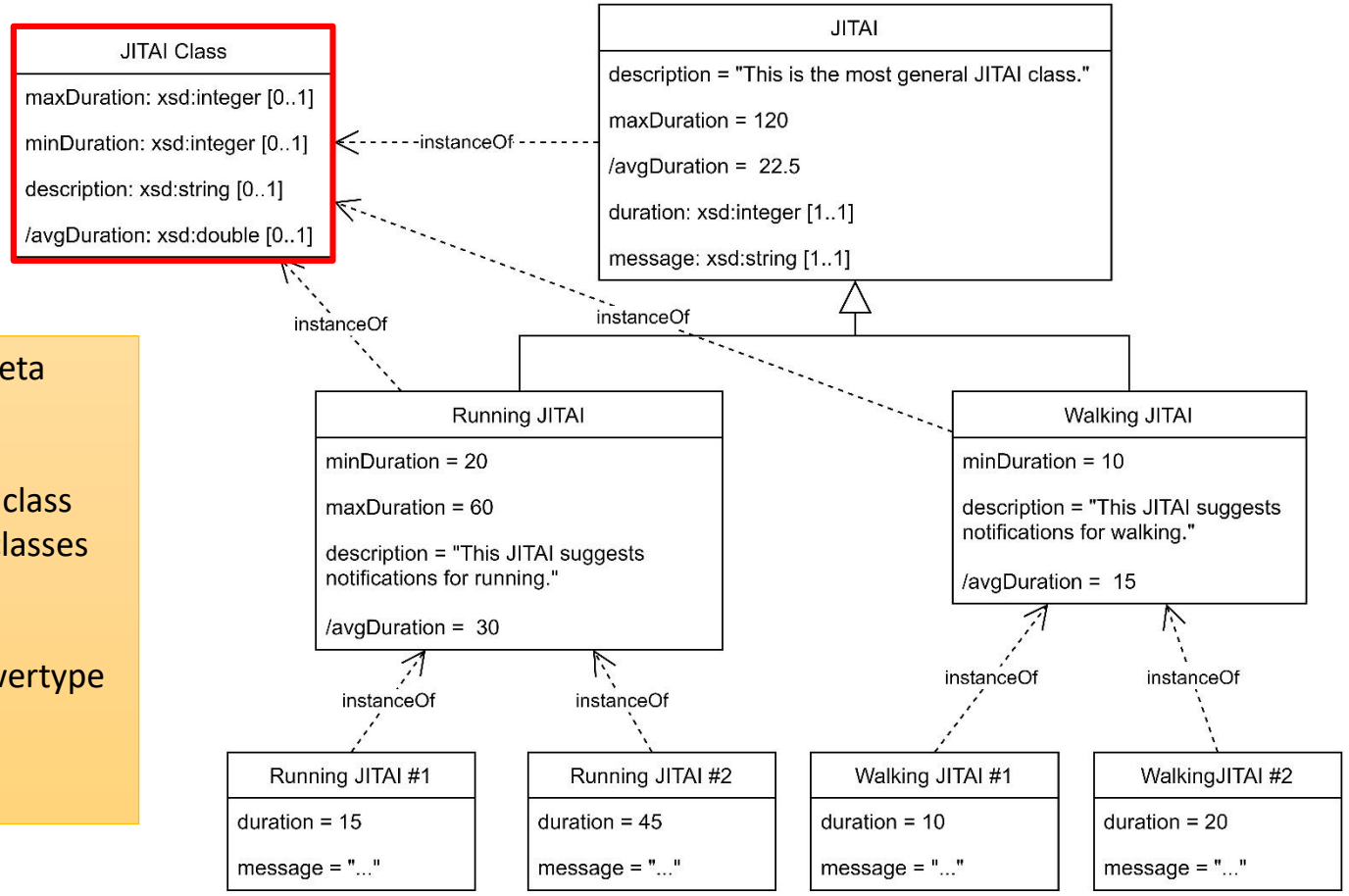
Basic **Multi-Level Modeling** in RDF/SHACL



This is a part of our running example of MLM of JITAIs which will be elaborated in the rest of the tutorial and forms the basis of the tasks to be prepared by students.

For illustration purposes we use an UML-like representation here, but in the remainder of the tutorial the running example will be represented using RDF and SHACL.

Basic **Multi-Level Modeling** in RDF/SHACL



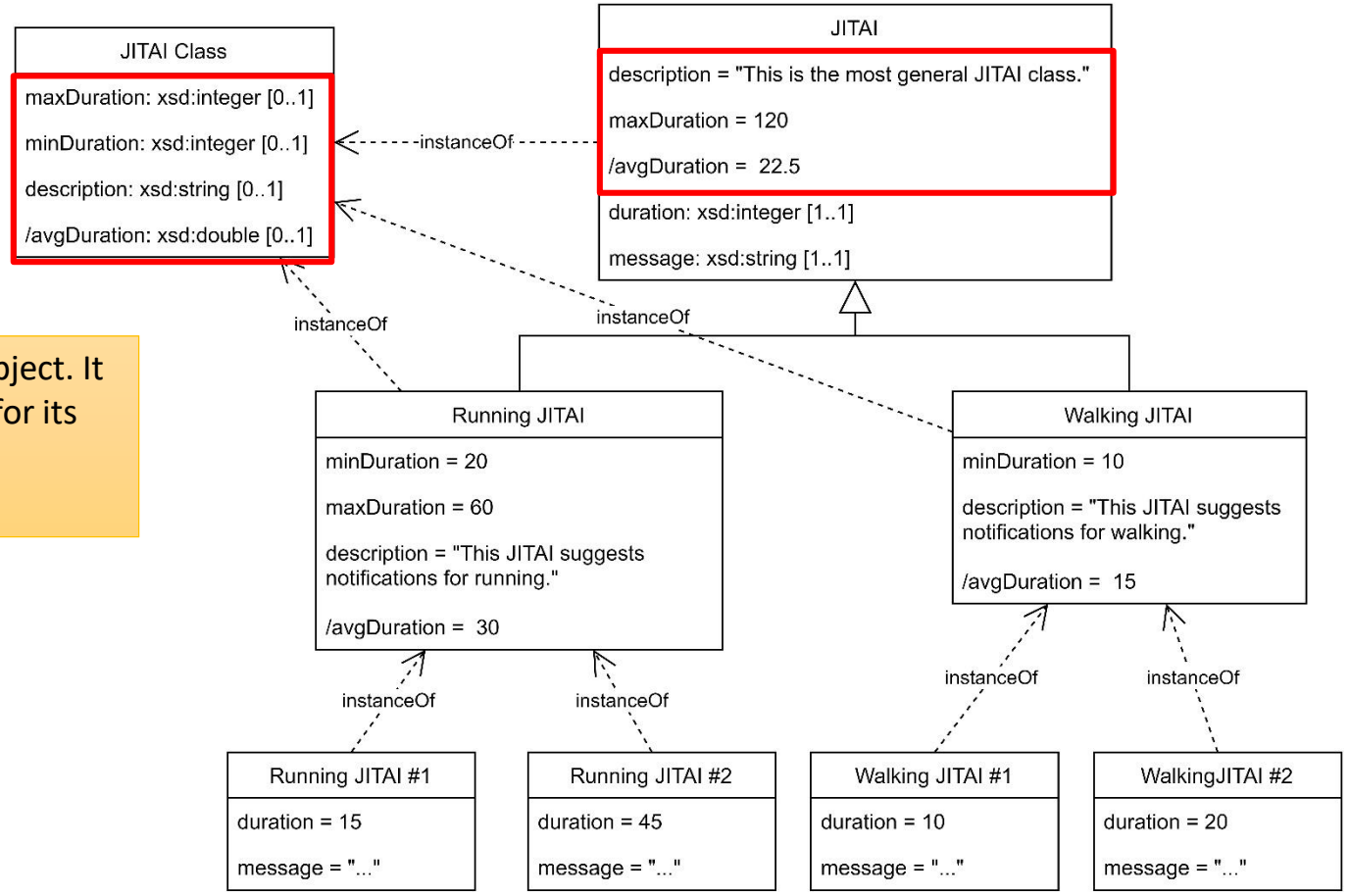
In MLM classes are instances of meta classes.

The **powertype** of a class has that class and all its direct and indirect sub classes as instances. (Carvalho et al., 2018)

For example, **JITAI Class** is the powertype of class JITAI and has JITAI and its subclasses as instances.

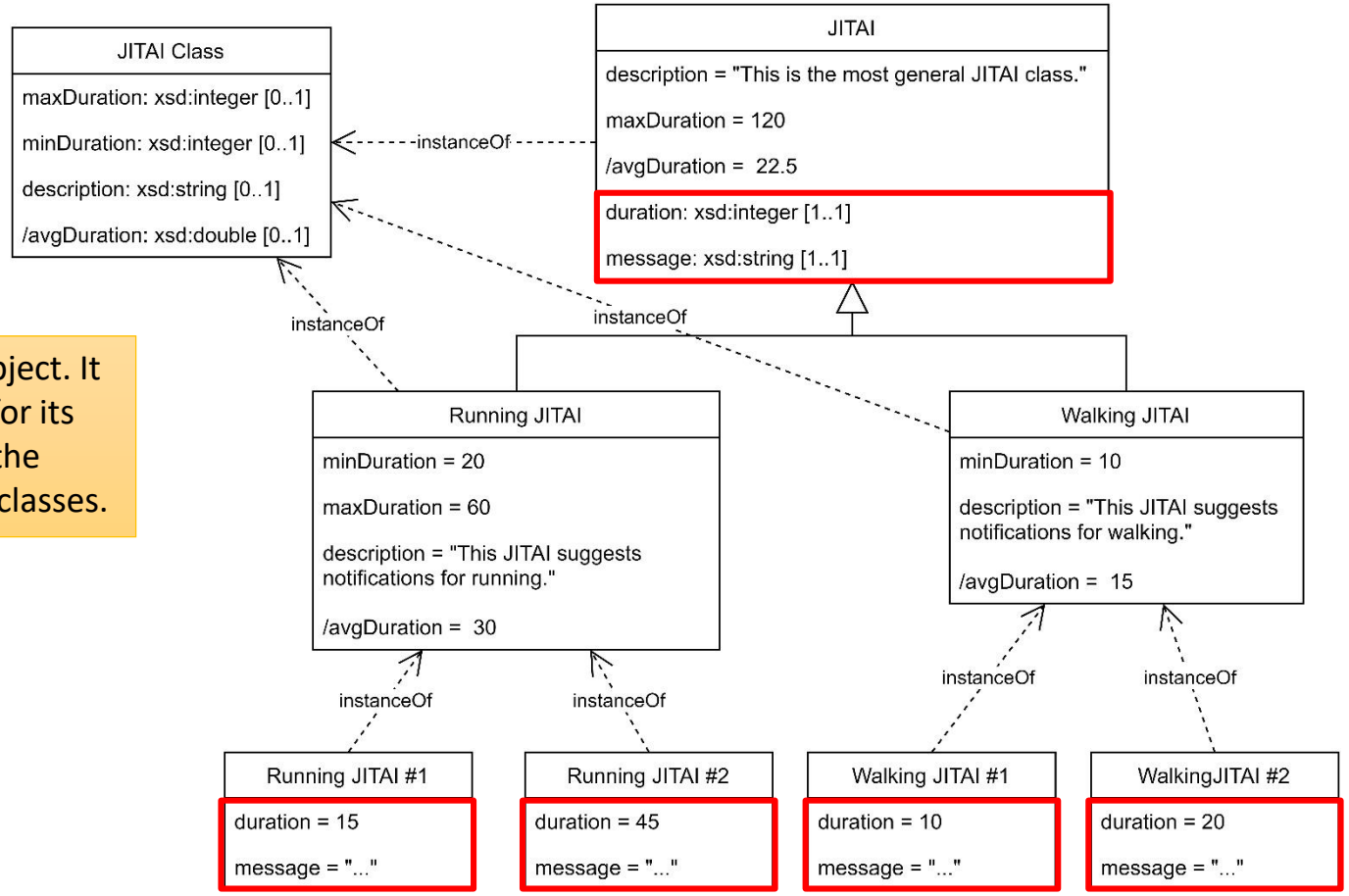
Basic **Multi-Level Modeling** in RDF/SHACL

A **clabject** is both a class and an object. It defines the schema of properties for its instances …



Basic **Multi-Level Modeling** in RDF/SHACL

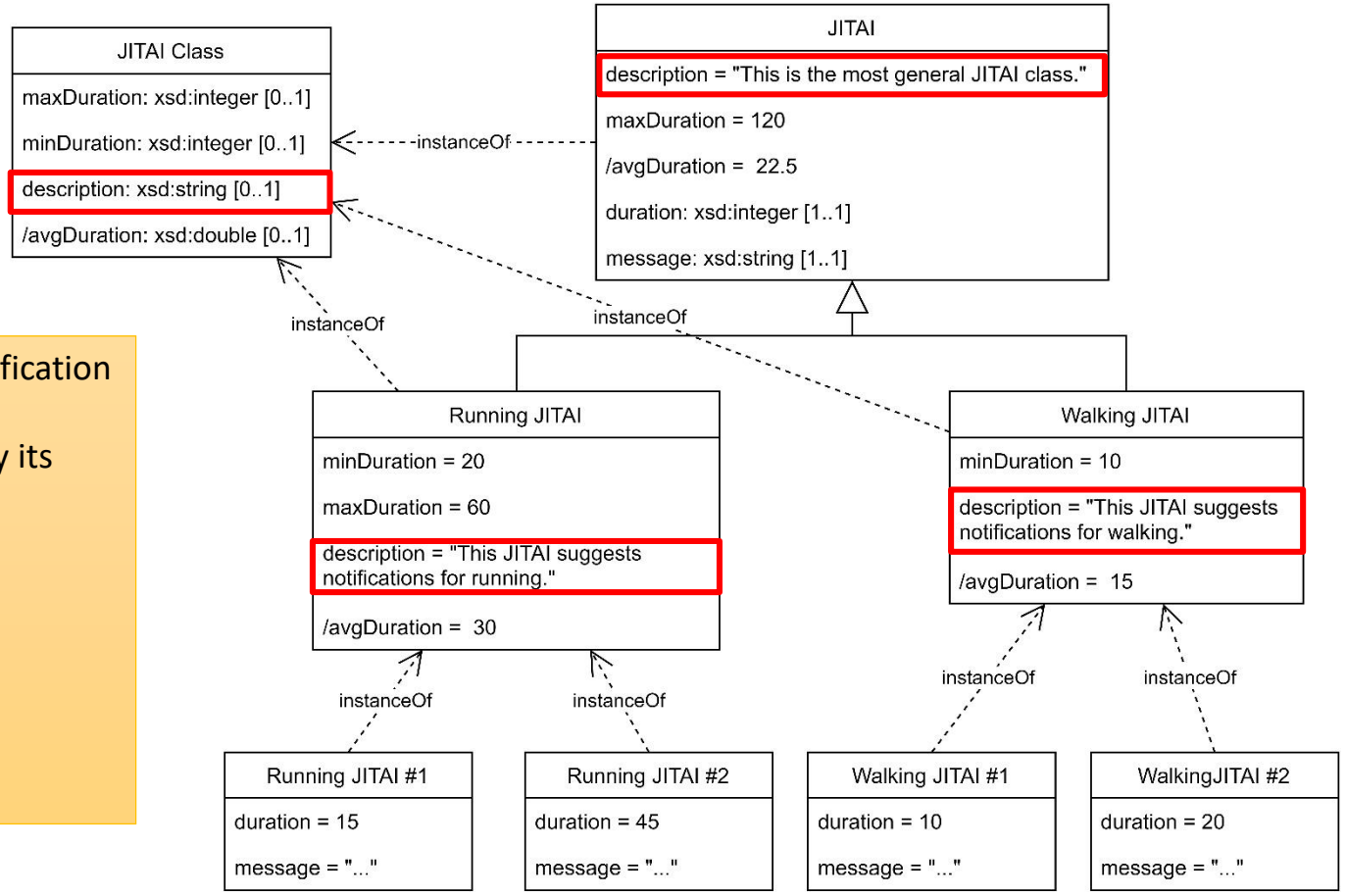
A **clabject** is both a class and an object. It defines the schema of properties for its instances and specifies values for the properties introduced by its meta classes.



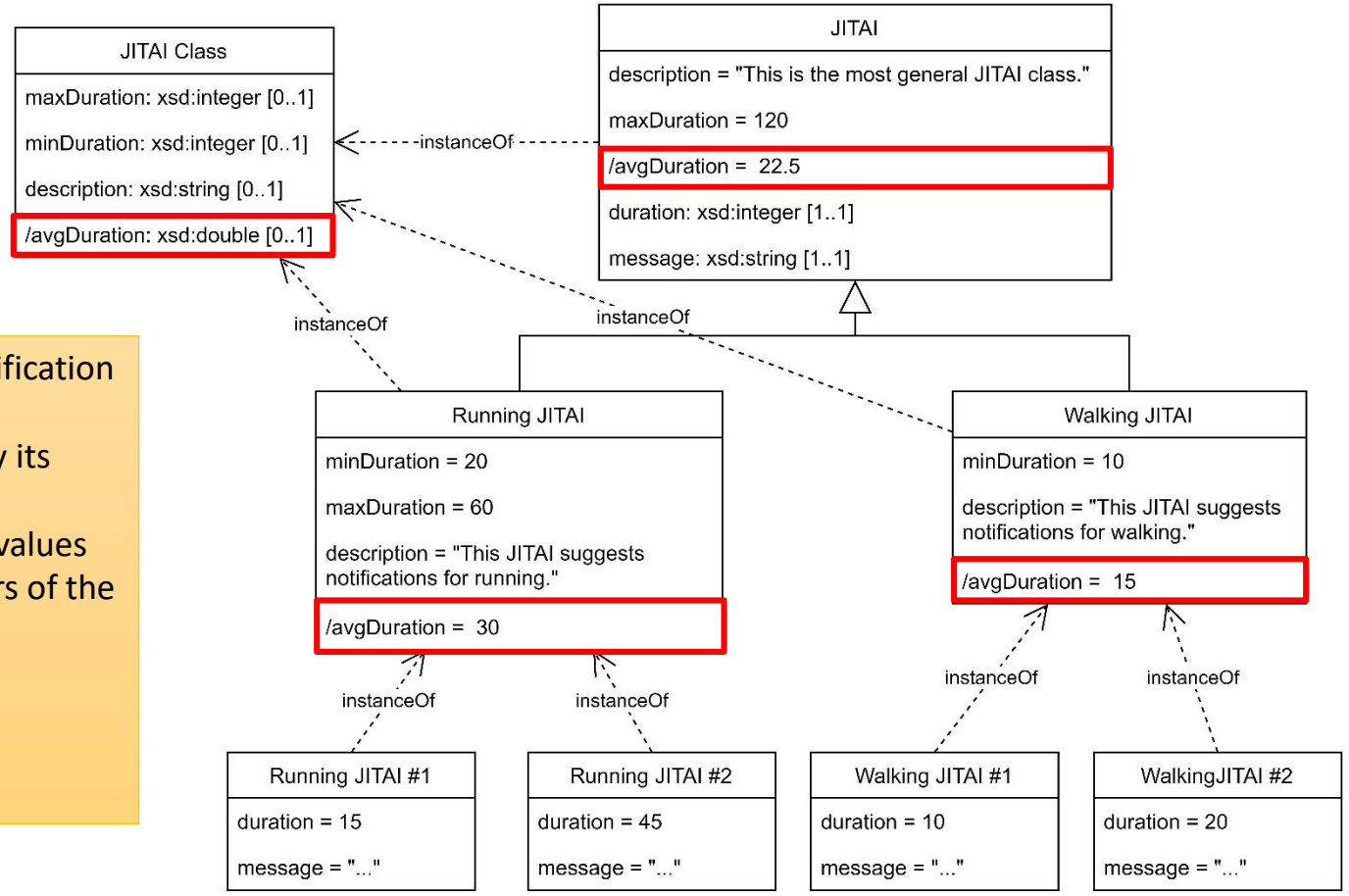
Basic **Multi-Level Modeling** in RDF/SHACL

Meta classes can be used for specification of

- **direct properties** instantiated by its member classes,



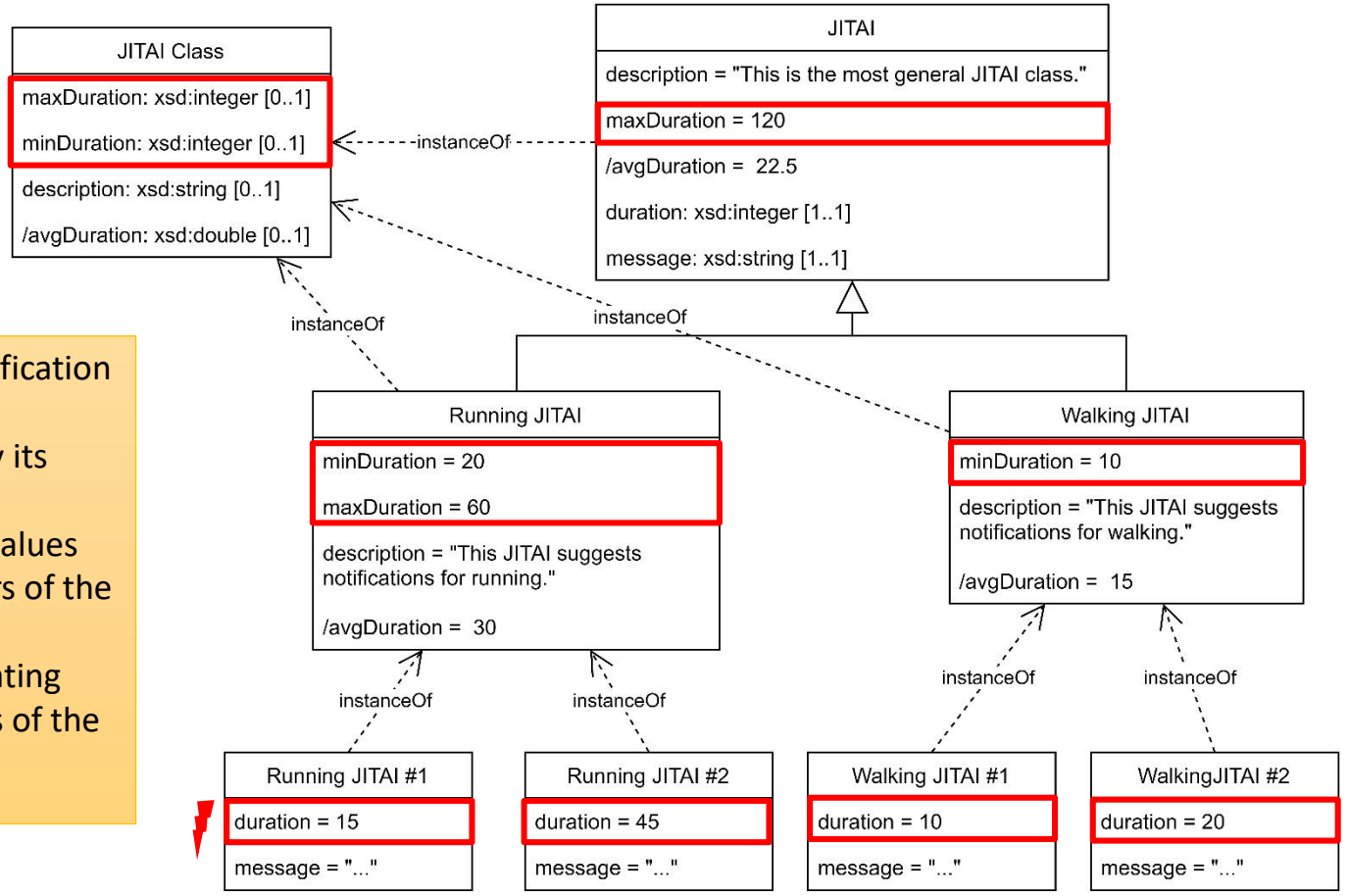
Basic **Multi-Level Modeling** in RDF/SHACL



Meta classes can be used for specification of

- direct properties instantiated by its member classes,
- **resultant properties** with their values being derived from the members of the member classes

Basic **Multi-Level Modeling** in RDF/SHACL



Meta classes can be used for specification of

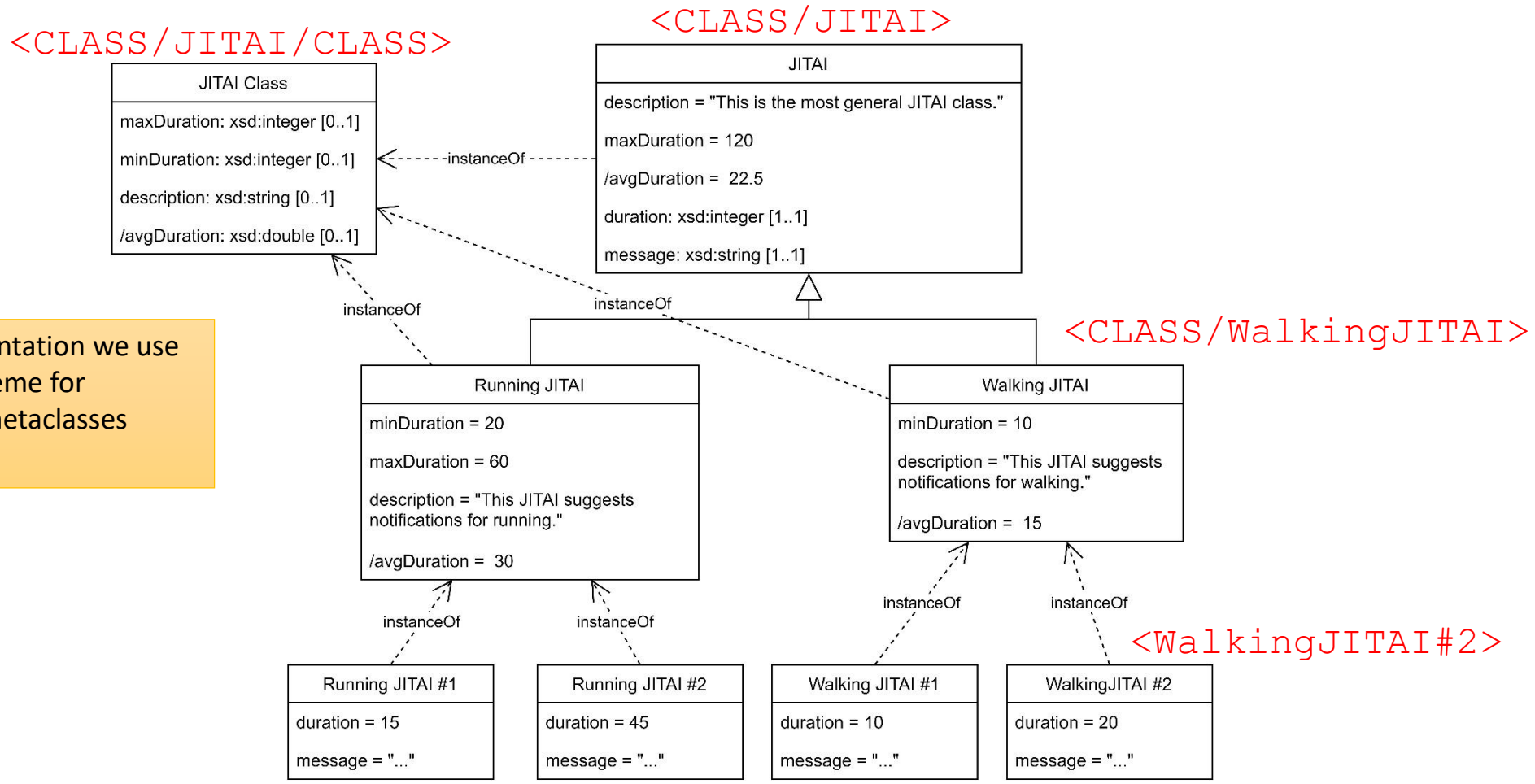
- direct properties instantiated by its member classes,
- resultant properties with their values being derived from the members of the member classes,
- and **regularity properties** regulating property values of the members of the member classes.

(Almeida et al., 2021)

Basic Multi-Level Modeling **in RDF/SHACL**

- The Resource Description Framework (RDF) is a W3C standard for graph-based description of resources.
  - https://www.w3.org/TR/rdf11-primer/
- The Shapes Constraint Language (SHACL) is a W3C standard for validation of RDF graphs and beyond.
  - https://www.w3.org/TR/shacl/
  - https://www.w3.org/TR/shacl-af/
- SHACL builds on SPARQL, the query language for RDF, which is also a W3C standard
  - https://www.w3.org/TR/sparql11-query/

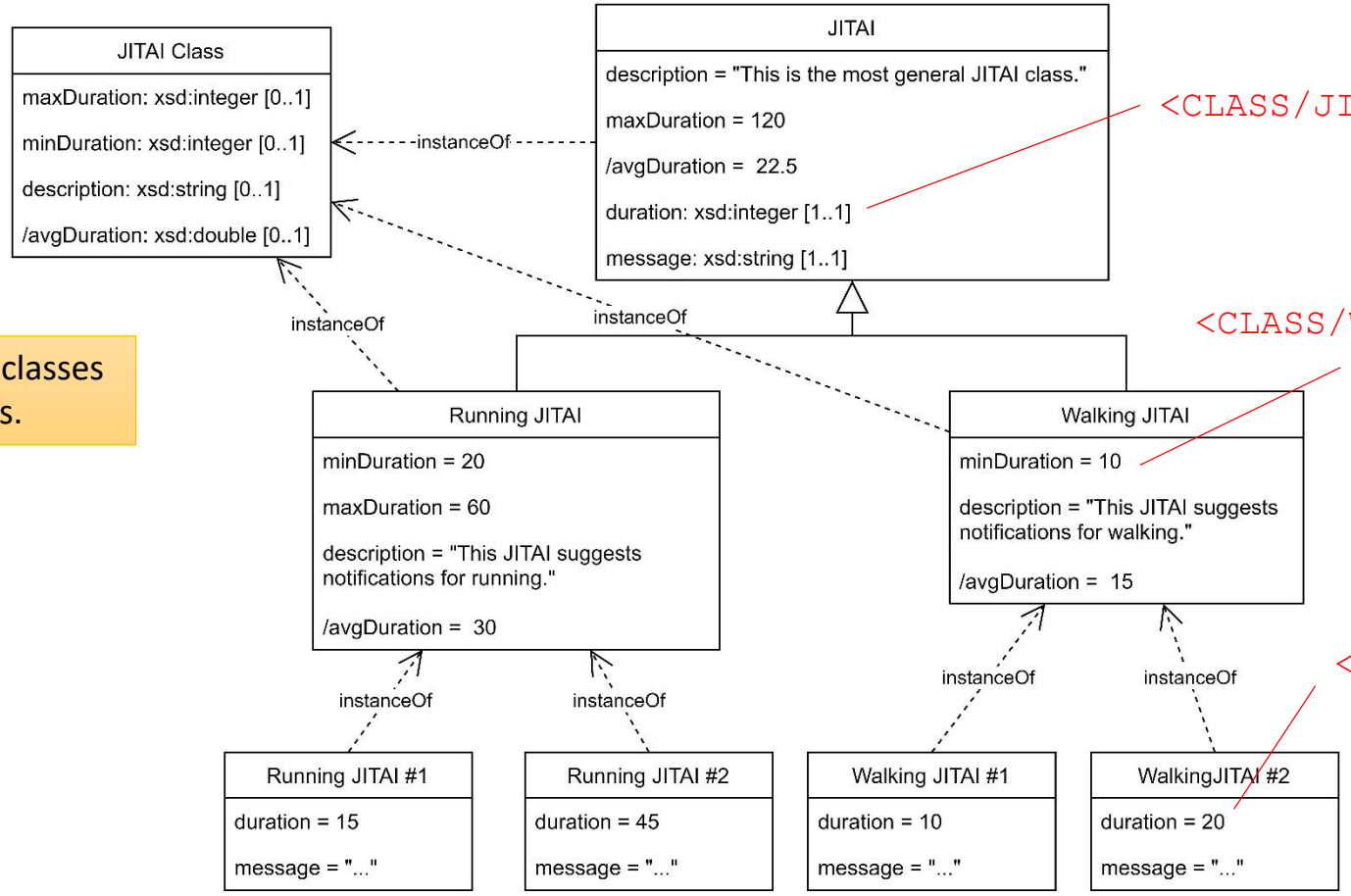
Basic Multi-Level Modeling **in RDF/SHACL**



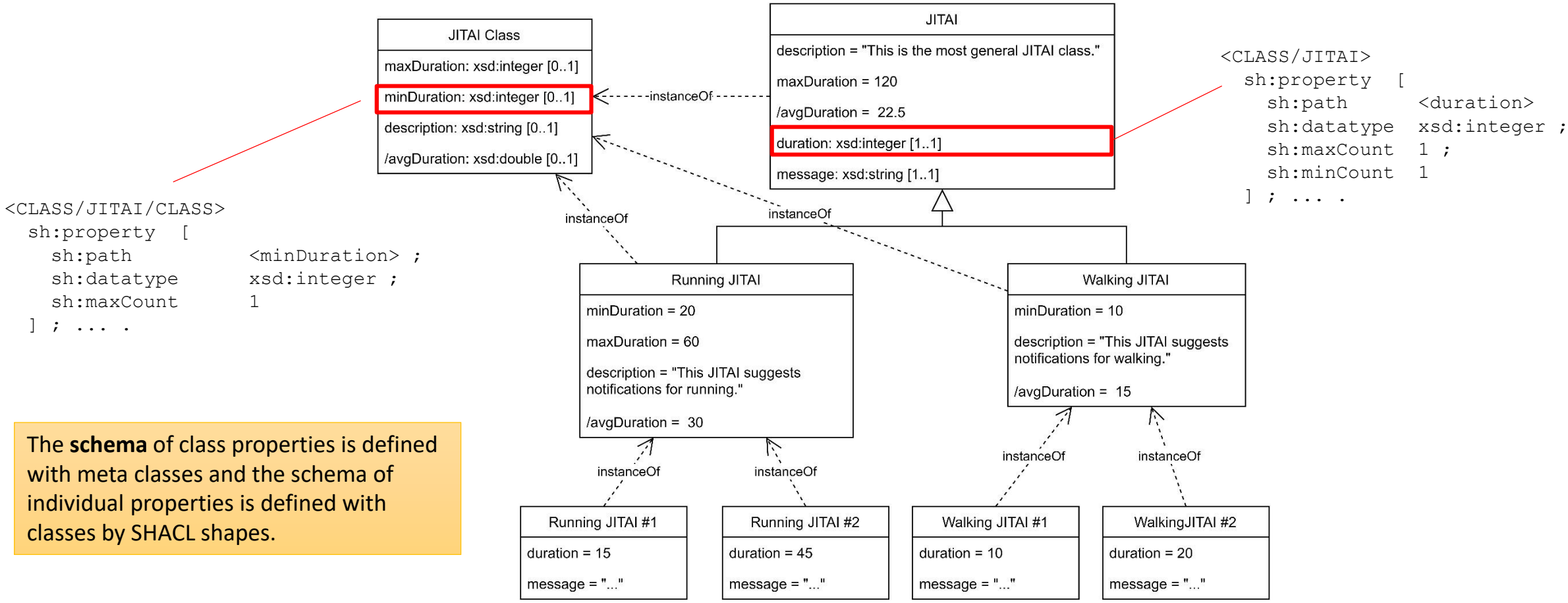
In the RDF/SHACL representation we use a specific URI naming scheme for individuals, classes, and metaclasses illustrated here.

Basic Multi-Level Modeling **in RDF/SHACL**

Property values of individuals and classes are represented by RDF statements.



Basic Multi-Level Modeling **in RDF/SHACL**




```
<CLASS/JITAI/CLASS>
  sh:property  [
    sh:path          <minDuration> ;
    sh:datatype      xsd:integer ;
    sh:maxCount      1
  ] ; ... .
```

```
<CLASS/JITAI>
  sh:property  [
    sh:path      <duration>
    sh:datatype  xsd:integer ;
    sh:maxCount  1 ;
    sh:minCount  1
  ] ; ... .
```

The **schema** of class properties is defined with meta classes and the schema of individual properties is defined with classes by SHACL shapes.

Basic Multi-Level Modeling **in RDF/SHACL**

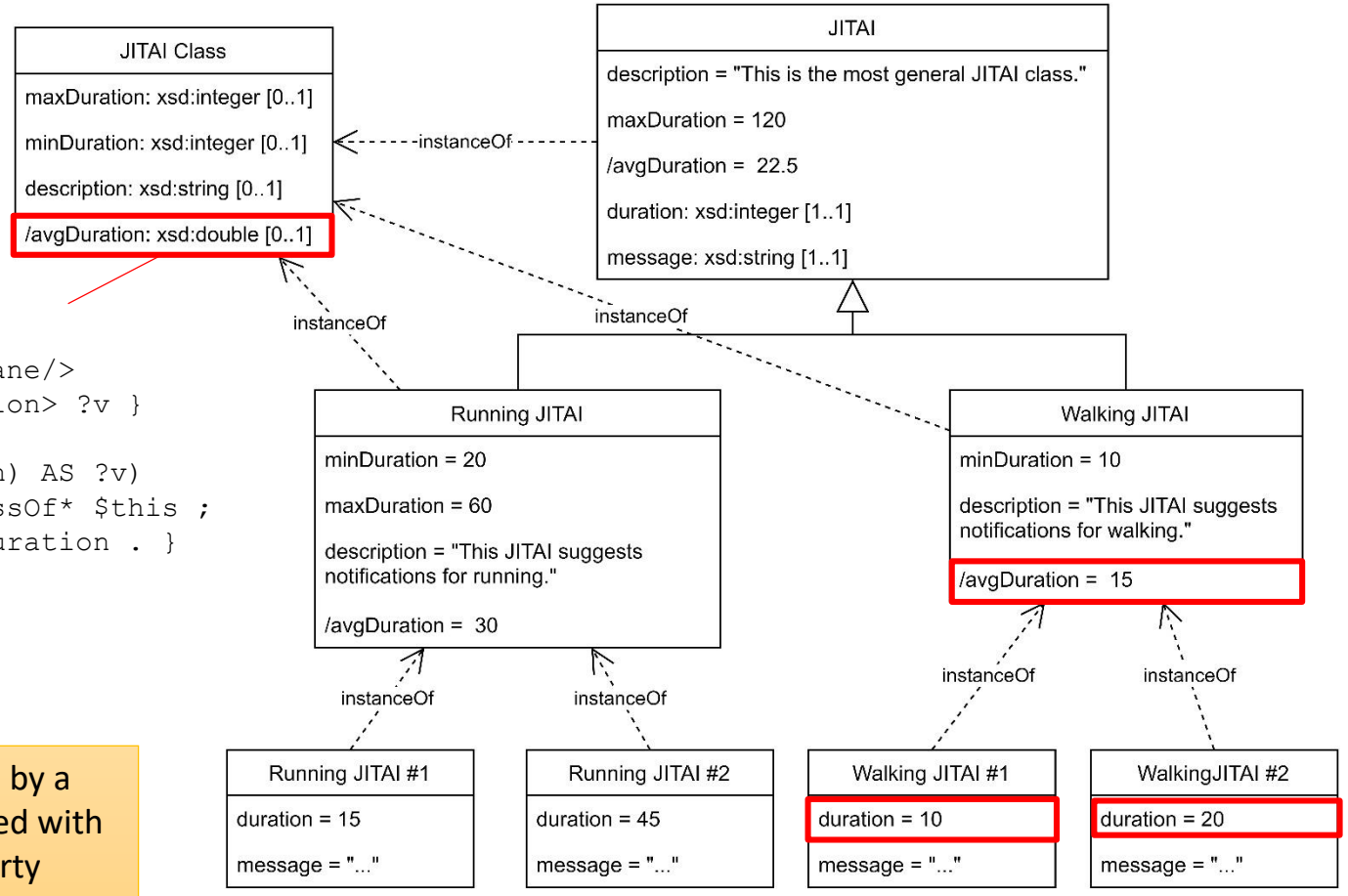
```
<CLASS/JITAI/CLASS> sh:rule [
  rdf:type  sh:SPARQLRule ;
  sh:construct  "
   BASE <https://example.org/Jane/>
   CONSTRUCT {
    $this sh:property ?shape .
    ?shape sh:path <duration> ;
    sh:minInclusive ?minDuration . }
   WHERE {
    SELECT $this ?minDuration ?shape
    WHERE {
     $this <minDuration> ?minDuration
     BIND(IRI(CONCAT(STR($this),
         \"#\",\"duration\")) AS ?shape) } }" ;
] ;
```


```
<CLASS/WalkingJITAI#duration>
  sh:path          <duration> ;
  sh:minInclusive  10 .
<CLASS/WalkingJITAI>
  <minDuration>  10 ;
  sh:property <CLASS/WalkingJITAI#duration> .
```

Each **regularity property** comes with a SPARQL-based SHACL rule (specified with the meta class) that derives SHACL constraints for classes from regularity property values.

Basic Multi-Level Modeling **in RDF/SHACL**

```
<CLASS/JITAI/CLASS>
 sh:rule [
  rdf:type sh:SPARQLRule ;
  sh:construct  """
   BASE <https://example.org/Jane/>
   CONSTRUCT { $this <avgDuration> ?v }
   WHERE {
    SELECT $this (AVG(?duration) AS ?v)
    WHERE { ?obj a/rdfs:subClassOf* $this ;
                <duration> ?duration . }
    GROUP BY $this }
    """
   ]
```



Each **resultant property** is defined by a SPARQL-based SHACL rule (specified with the meta class) that derives property values for classes from individuals.

Prototype

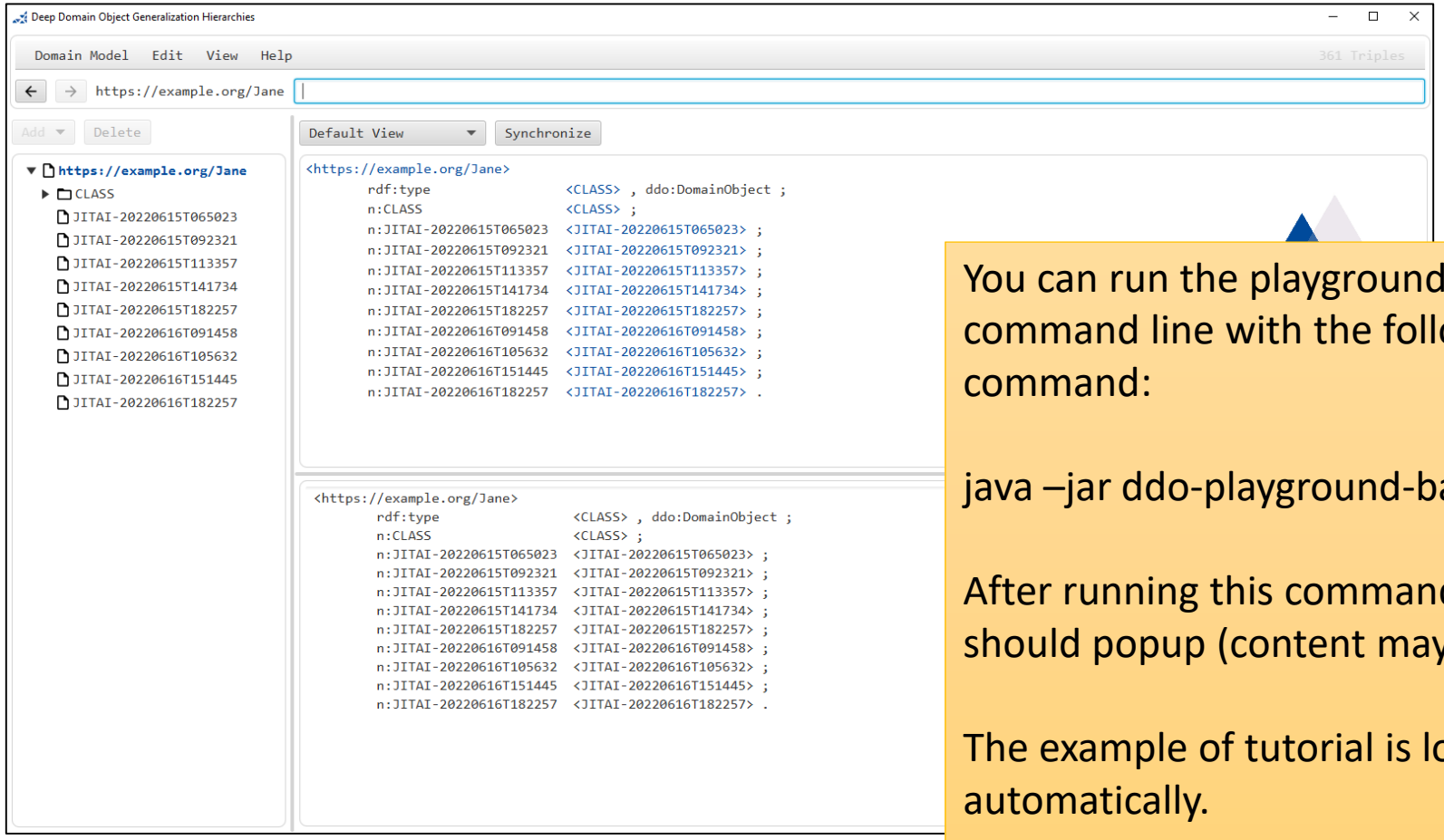
DDO Playground

- The prototype (DDO-Playground) used in the tutorial implements an extension of the Deep Domain Object Hierarchies (DDO) approach (Neumayr & Schrefl, 2022).
- For the purposes of this tutorial, we use the DDO-Playground, but severely limit its functionality and do not consider DDO-specific language constucts and composition hierarchies.
- You can download the limited version of the DDO-Playground here:
  - https://resources.lbi-dhp.at/mlm4jitais/ddo-playground-basic.jar

Note: You may have to install Oracle JDK 18.
(You may run into troubles using Open JDK)

DDO-Playground

Getting Started



You can run the playground from command line with the following command:

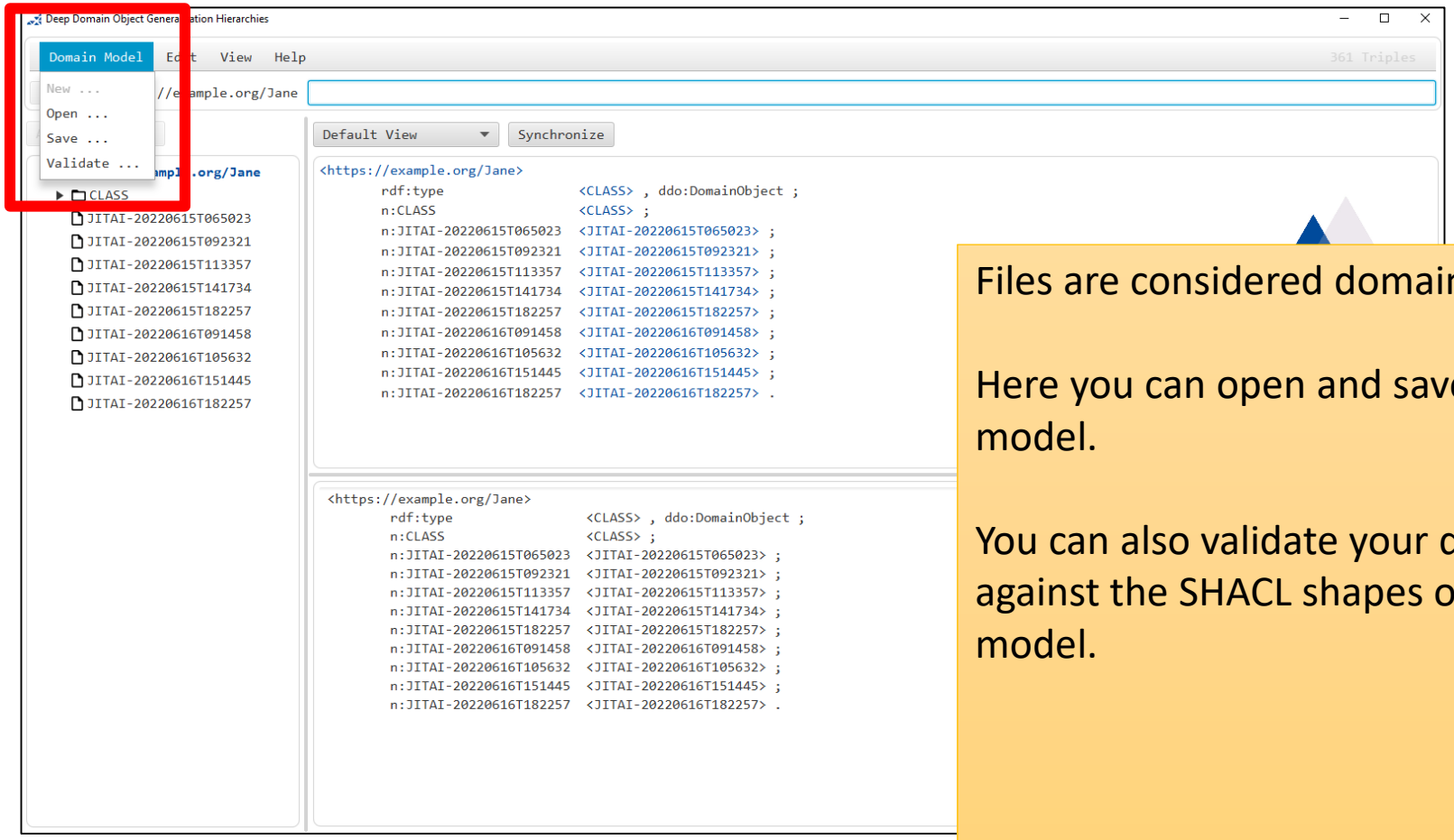
java –jar ddo-playground-basic.jar

After running this command, this window should popup (content may differ).

The example of tutorial is loaded automatically.

DDO-Playground

Domain Model



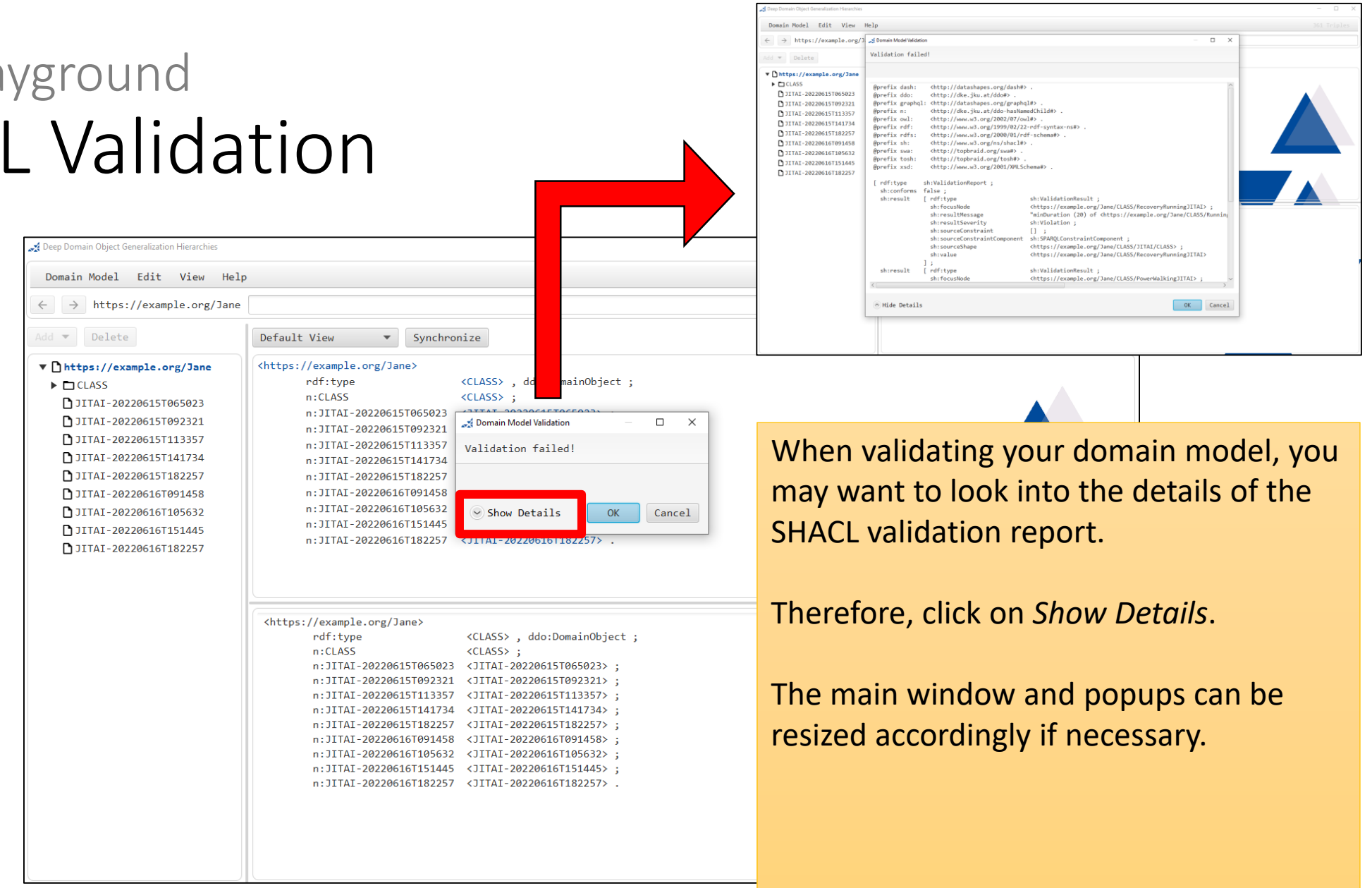
Files are considered domain models.

Here you can open and save a domain model.

You can also validate your domain model against the SHACL shapes of the current model.

DDO-Playground

SHACL Validation



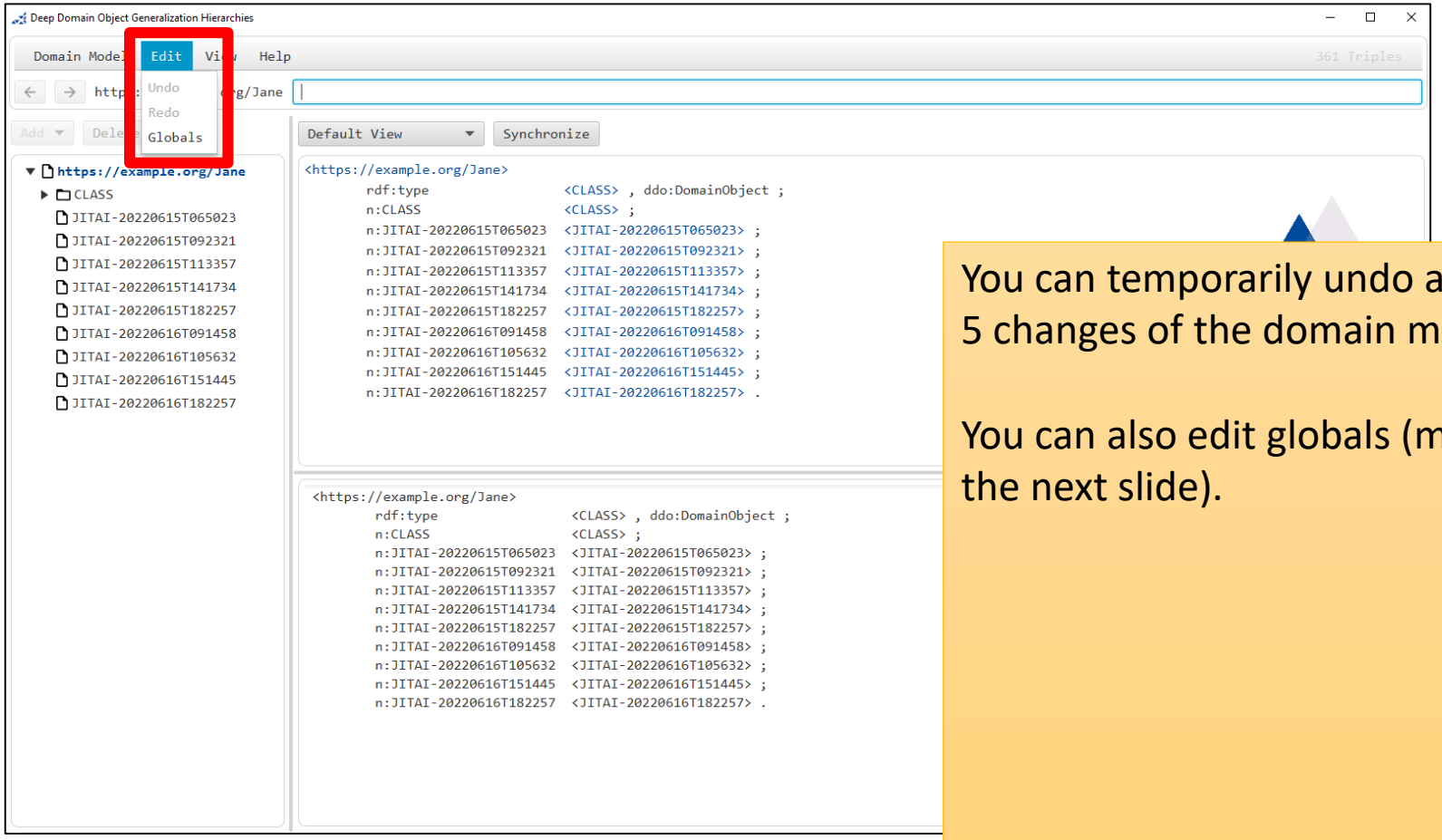
When validating your domain model, you may want to look into the details of the SHACL validation report.

Therefore, click on *Show Details*.

The main window and popups can be resized accordingly if necessary.

DDO-Playground

Edit

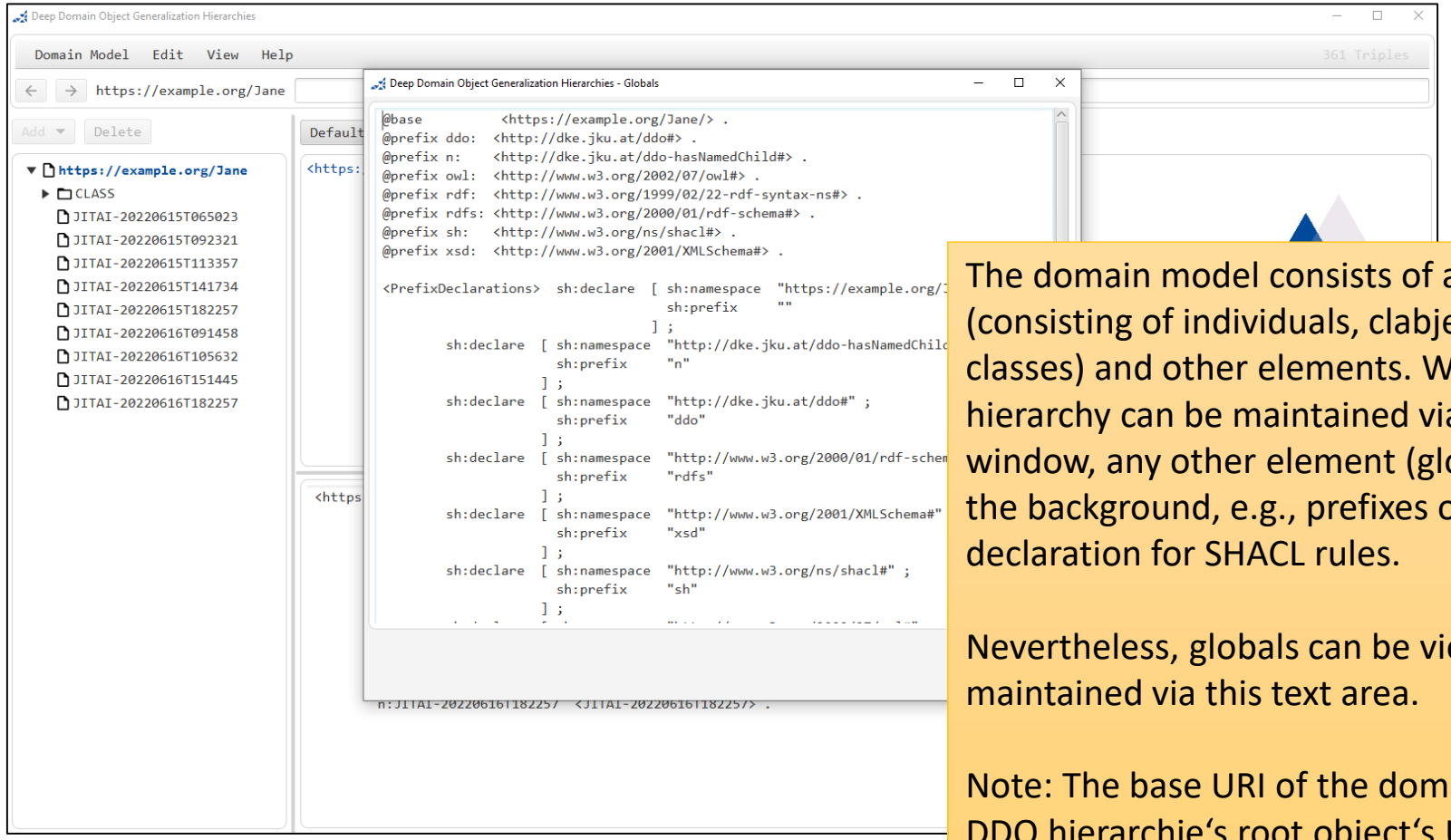


You can temporarily undo and redo up to 5 changes of the domain model.

You can also edit globals (more details on the next slide).

DDO-Playground

Globals



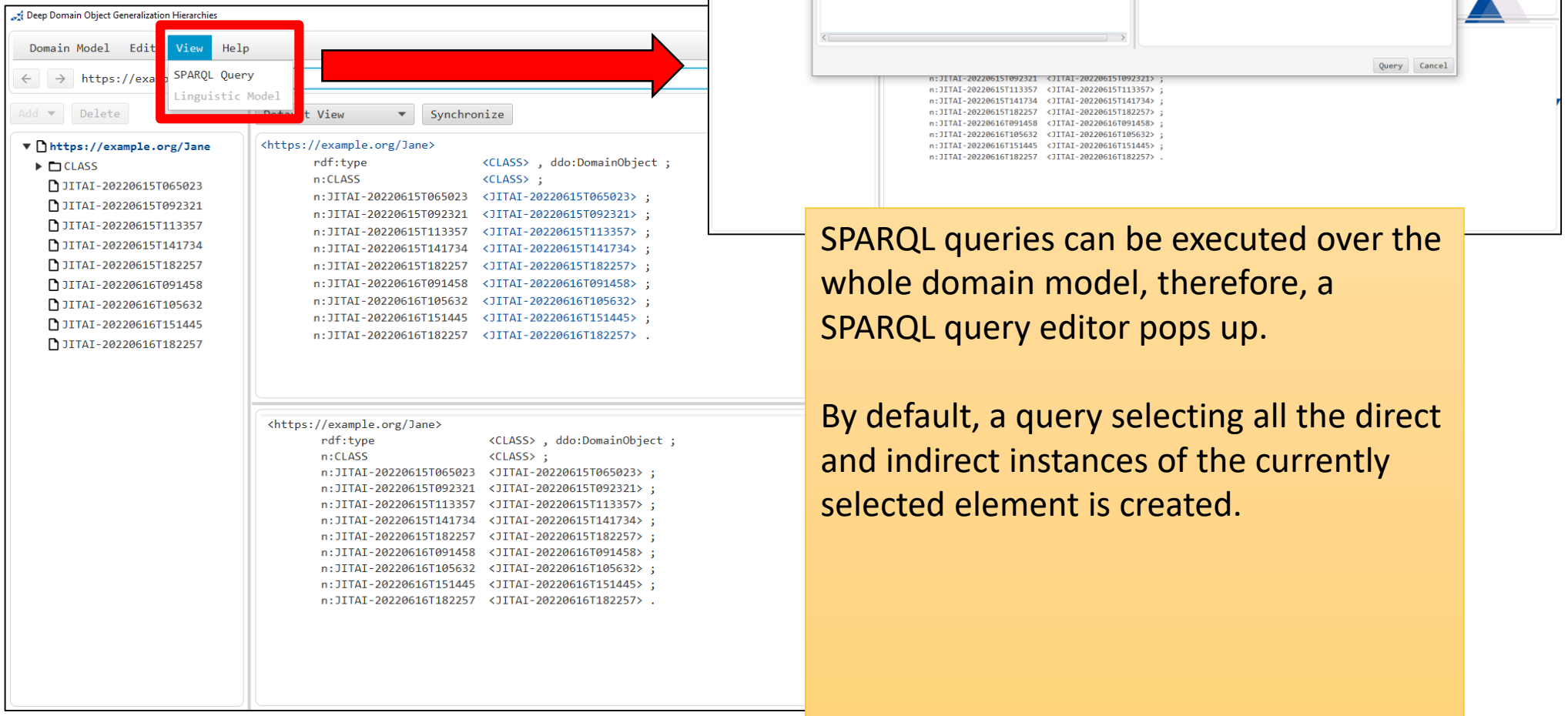
The domain model consists of a DDO hierarchy (consisting of individuals, clabjects and meta classes) and other elements. While the DDO hierarchy can be maintained via the main window, any other element (globals) remain in the background, e.g., prefixes or prefix declaration for SHACL rules.

Nevertheless, globals can be viewed or maintained via this text area.

Note: The base URI of the domain model is the DDO hierarchie‘s root object‘s URI + /.

DDO-Playground

SPARQL Queries

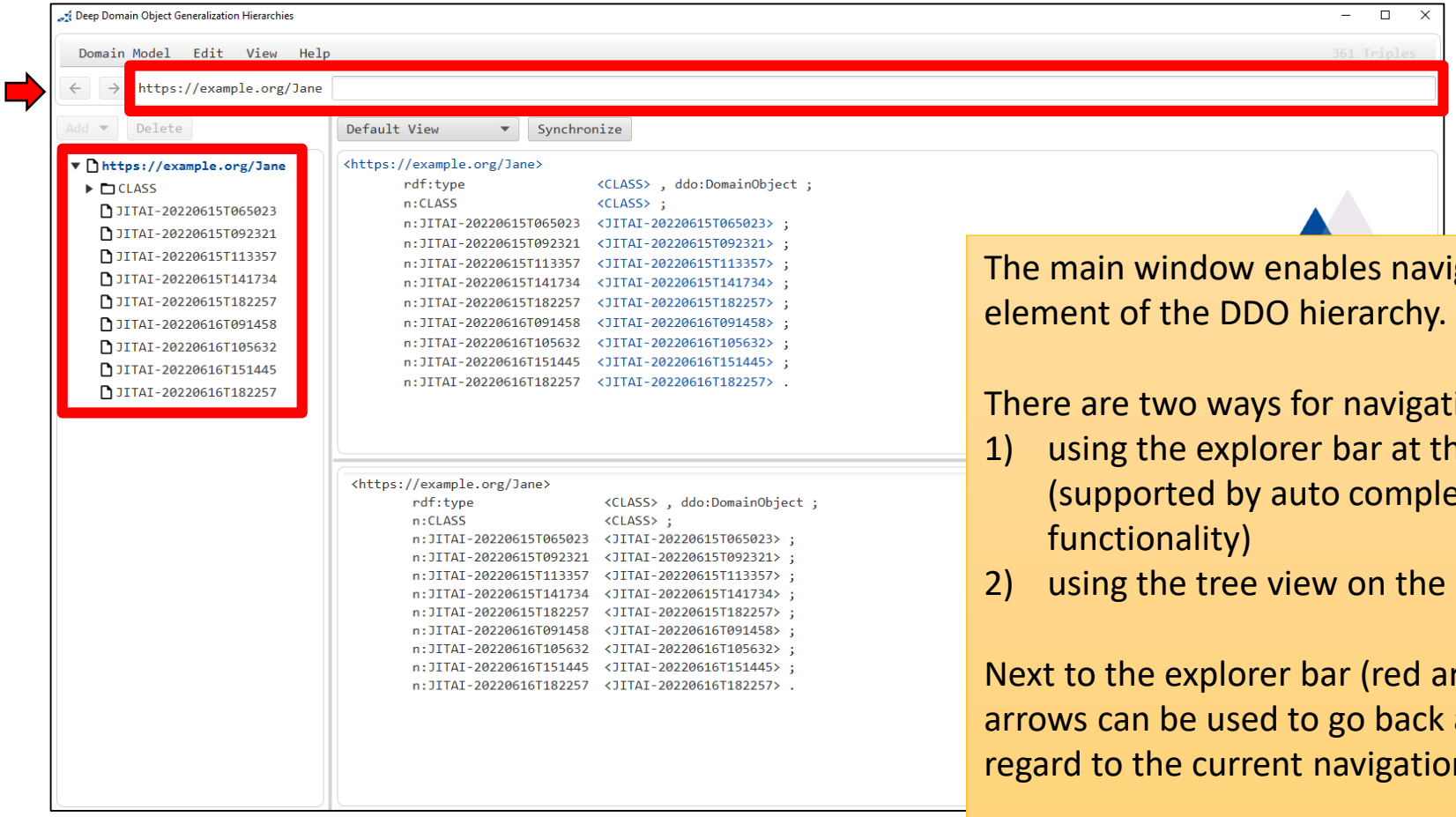


SPARQL queries can be executed over the whole domain model, therefore, a SPARQL query editor pops up.

By default, a query selecting all the direct and indirect instances of the currently selected element is created.

DDO-Playground

Navigation



The main window enables navigation to any element of the DDO hierarchy.

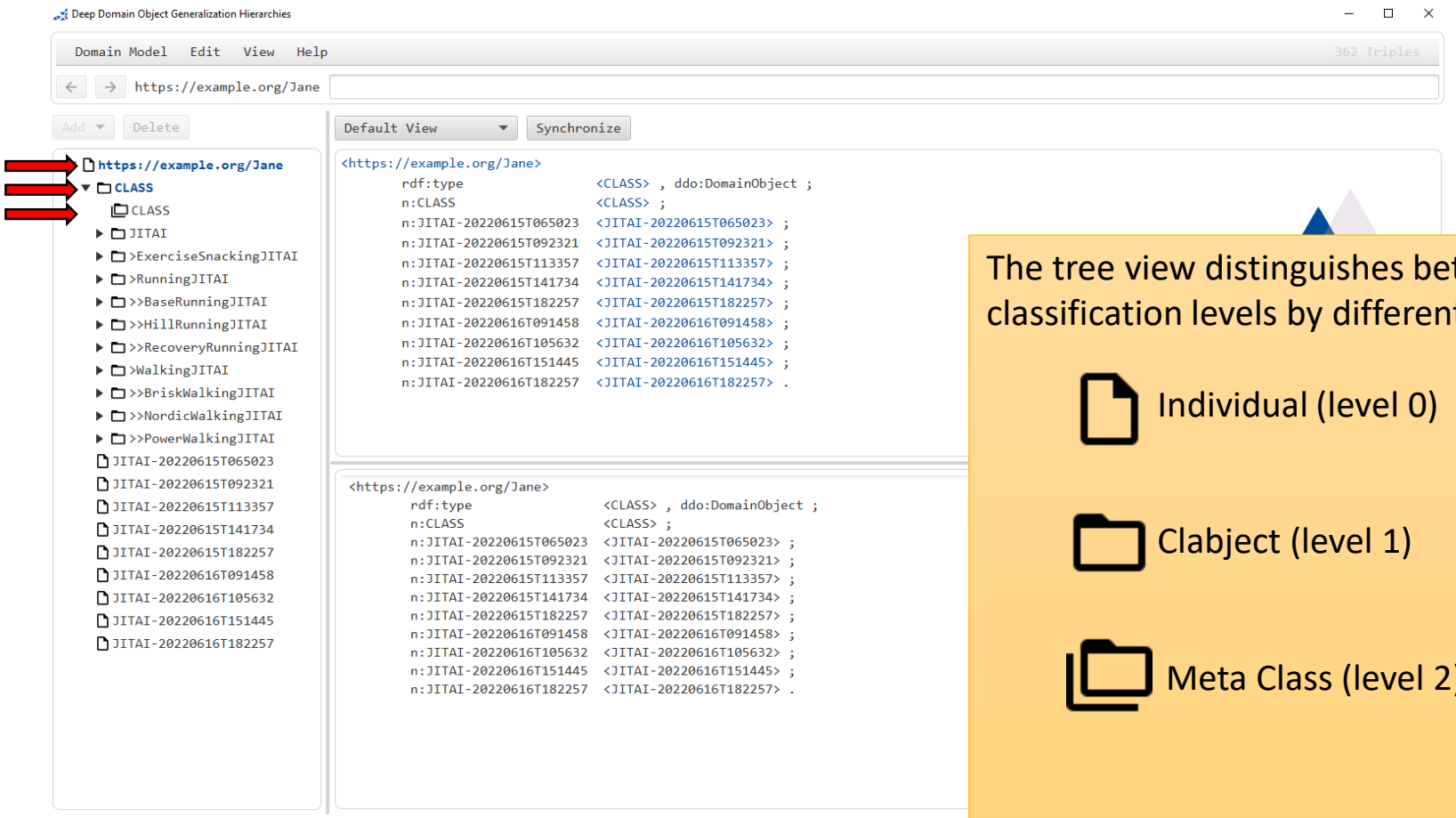
There are two ways for navigation:

1) using the explorer bar at the top (supported by auto complete functionality)
2) using the tree view on the left hand side

Next to the explorer bar (red arrow), the arrows can be used to go back and forth with regard to the current navigation history.

DDO-Playground

Tree View Symbols

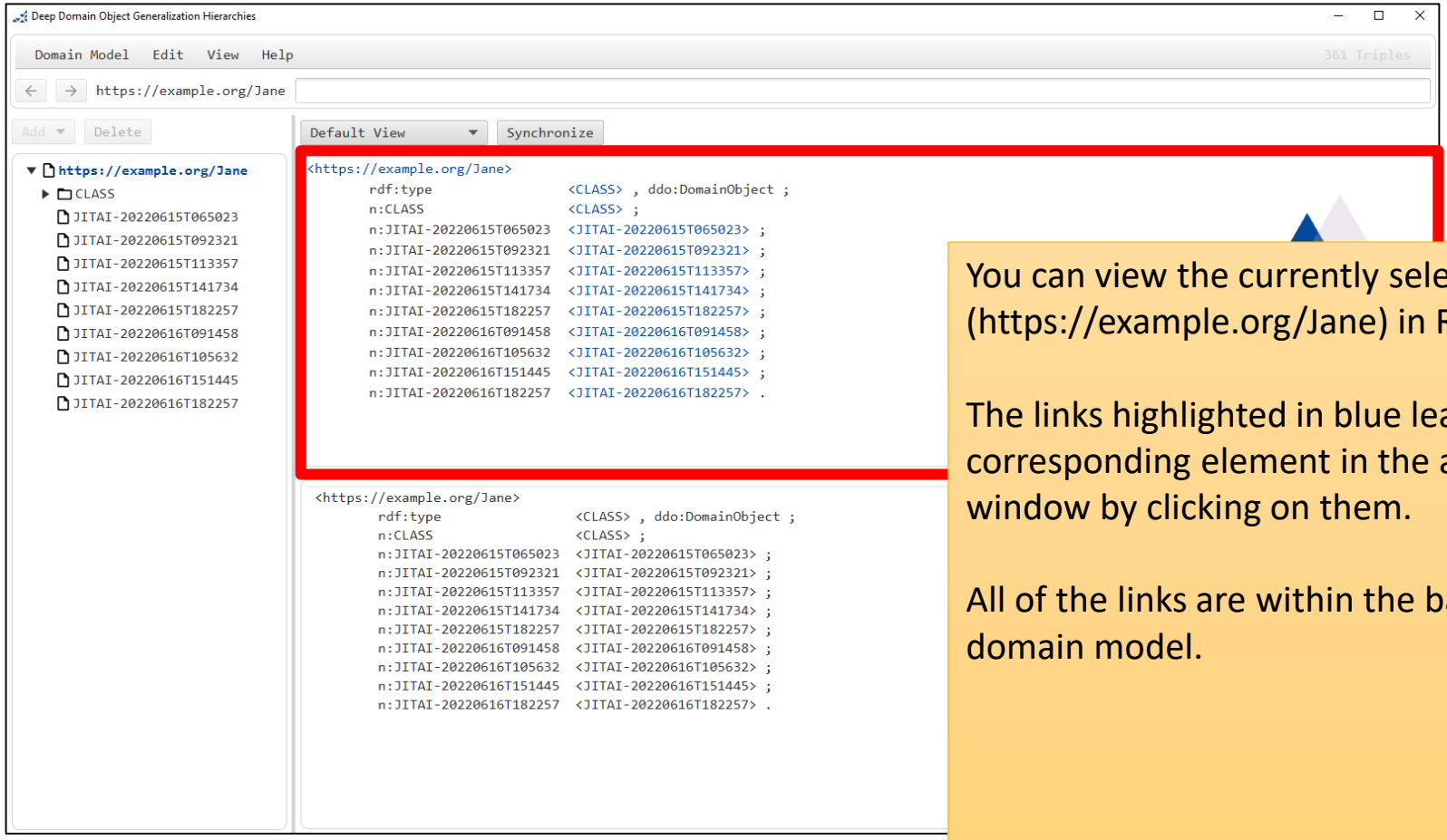


The tree view distinguishes between classification levels by different symbols:

- Individual (level 0)
- Clabject (level 1)
- Meta Class (level 2)

DDO-Playground

Element View



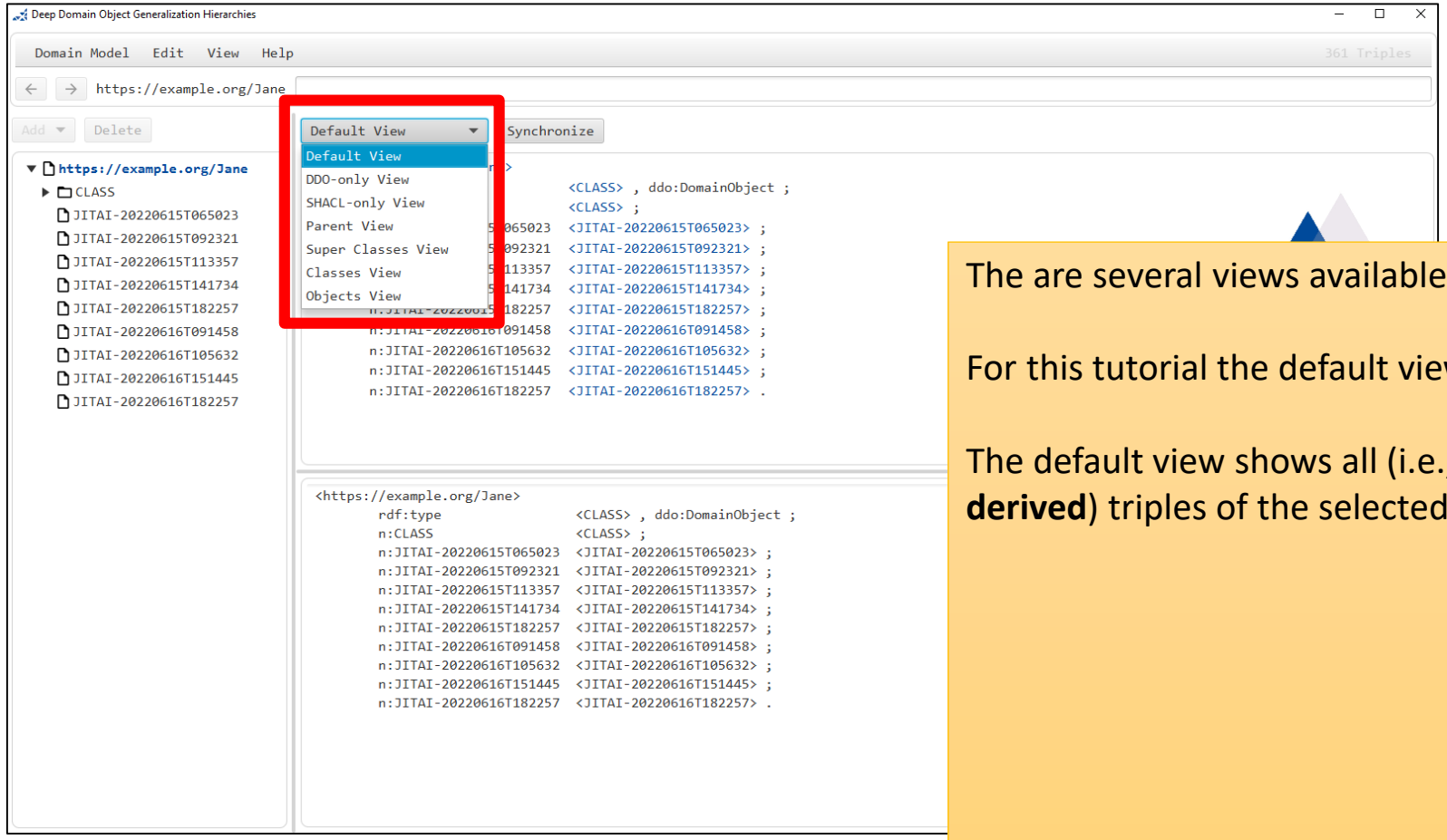
You can view the currently selected element (https://example.org/Jane) in RDF/Turtle.

The links highlighted in blue lead to the corresponding element in the application window by clicking on them.

All of the links are within the base URI of the domain model.

DDO-Playground

Views



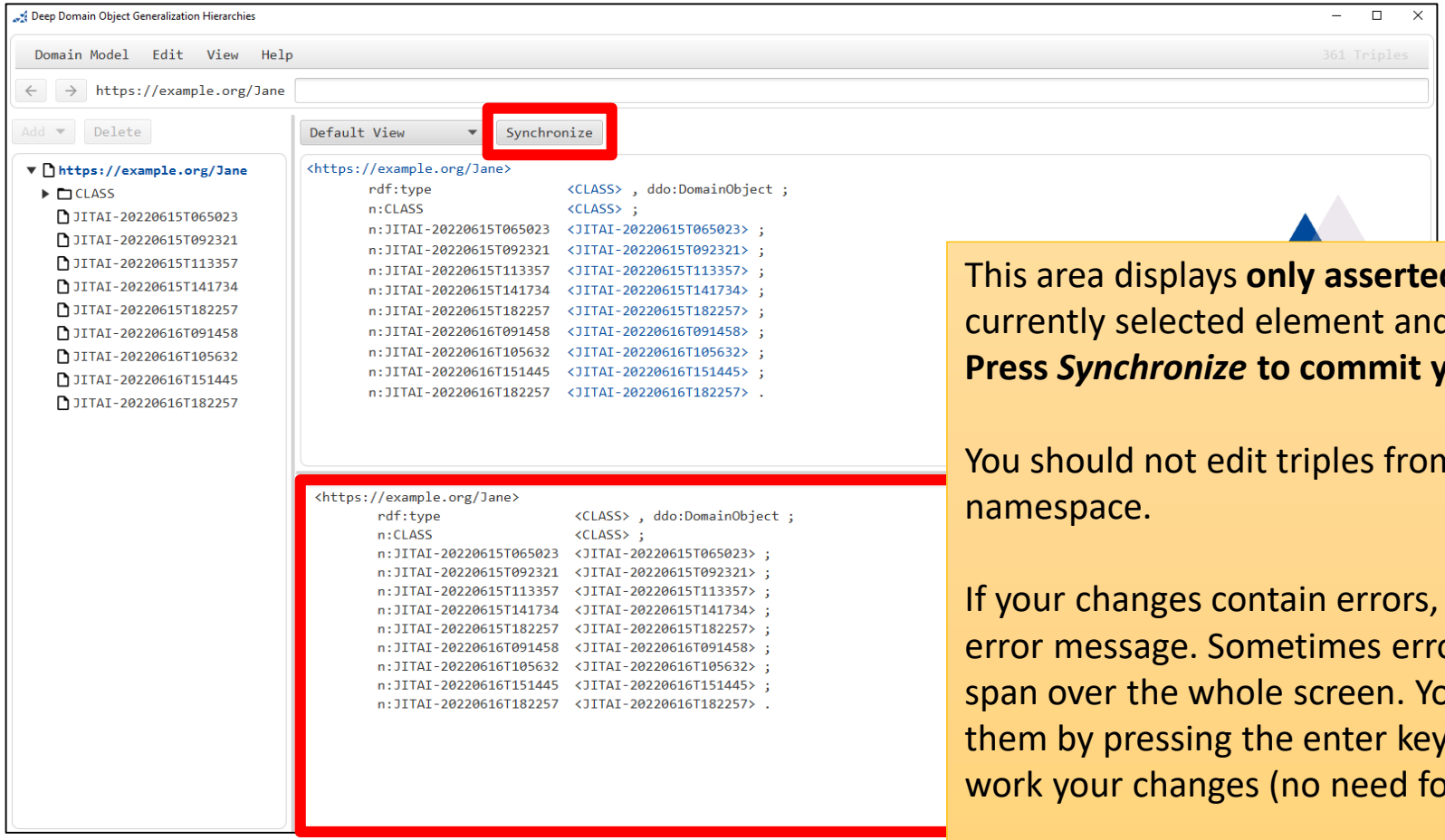
The are several views available.

For this tutorial the default view is sufficient.

The default view shows all (i.e., **asserted and derived**) triples of the selected element.

DDO-Playground

Editor

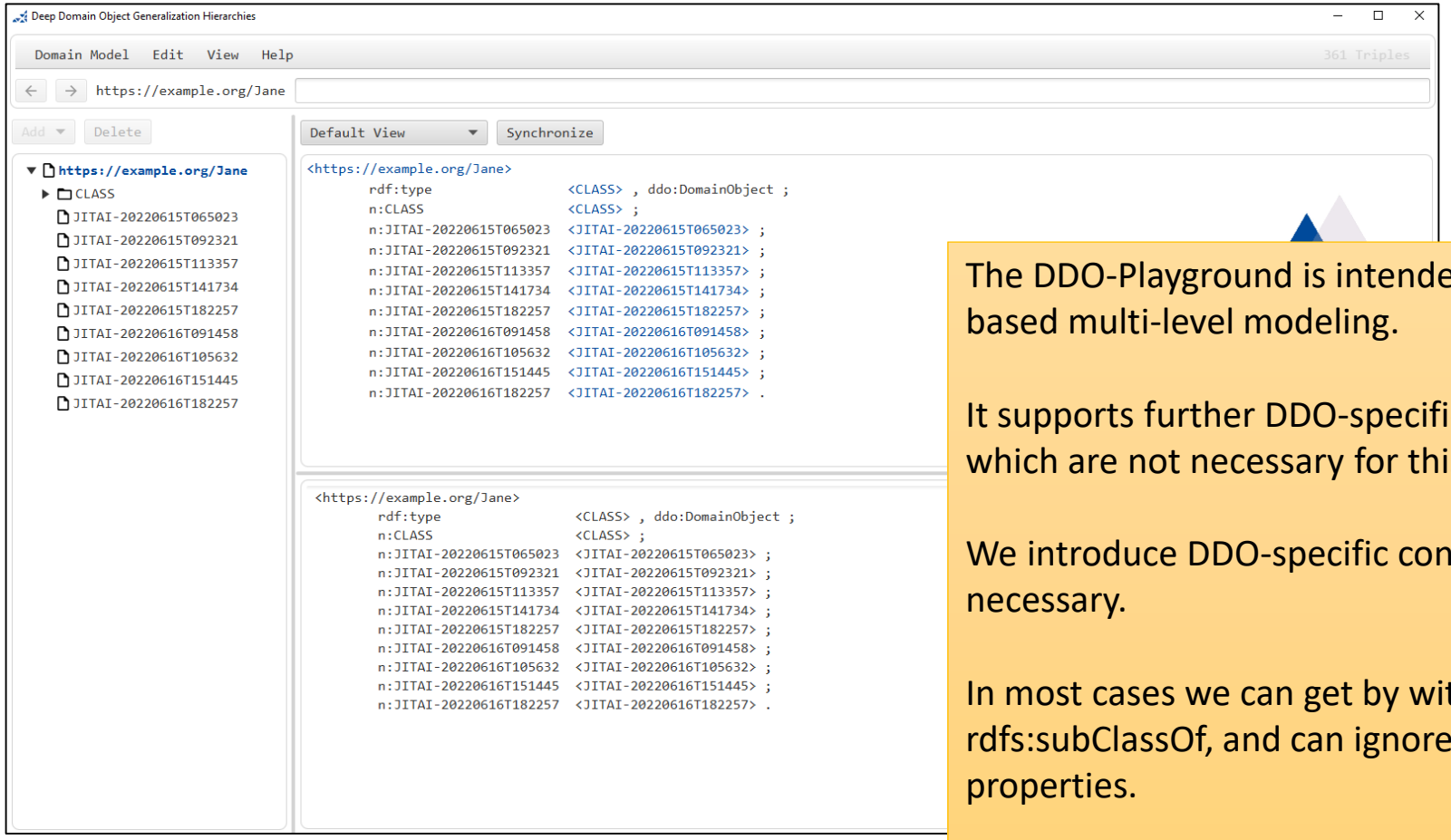


This area displays **only asserted** triples of the currently selected element and can be **edited**. **Press *Synchronize* to commit your changes.**

You should not edit triples from the ddo: or n: namespace.

If your changes contain errors, you will face an error message. Sometimes error messages span over the whole screen. You can close them by pressing the enter key. You can re-work your changes (no need for an undo).

DDO-specific Constructs



The DDO-Playground is intended for DDO-based multi-level modeling.

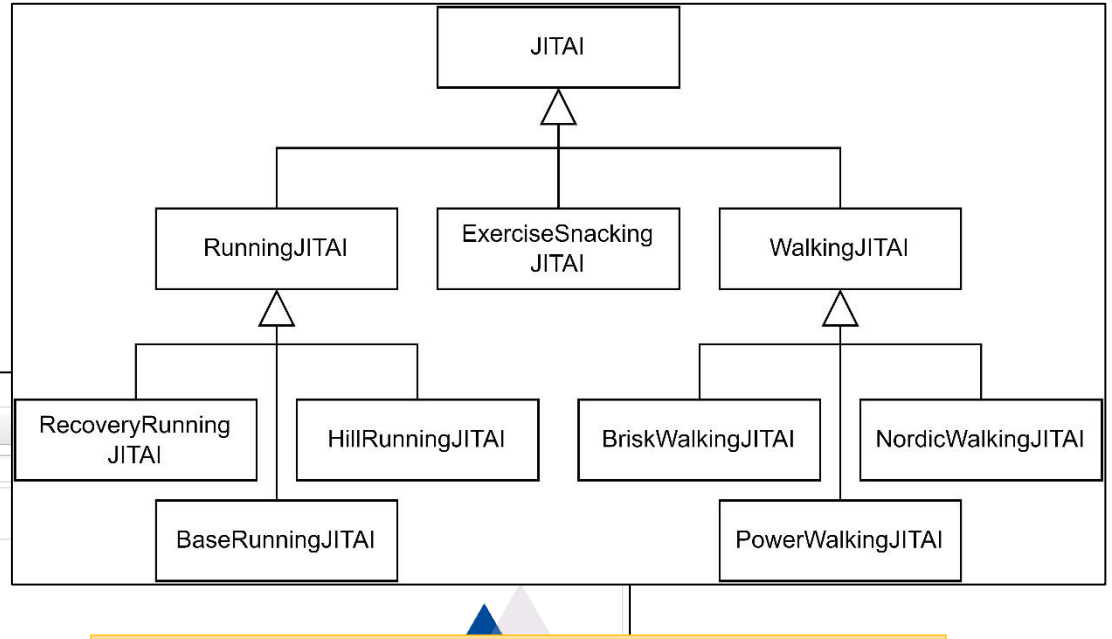
It supports further DDO-specific constructs which are not necessary for this tutorial.

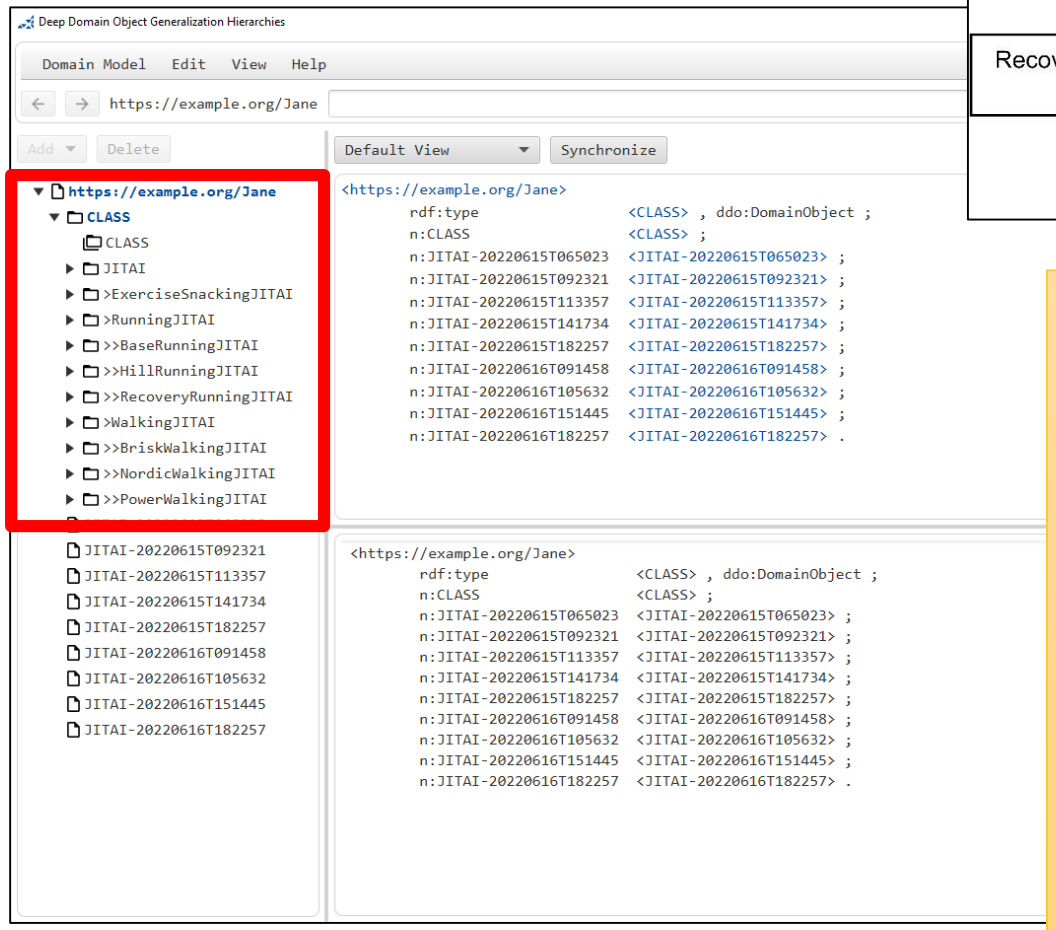
We introduce DDO-specific construct only if necessary.

In most cases we can get by with rdf:type and rdfs:subClassOf, and can ignore other properties.

Use Case

JITAI Clabjects





In this example, we were given access to Jane's JITAI data for the purpose of analysis and personalization.

The resource <https://example.org/Jane> is a representation of Jane.

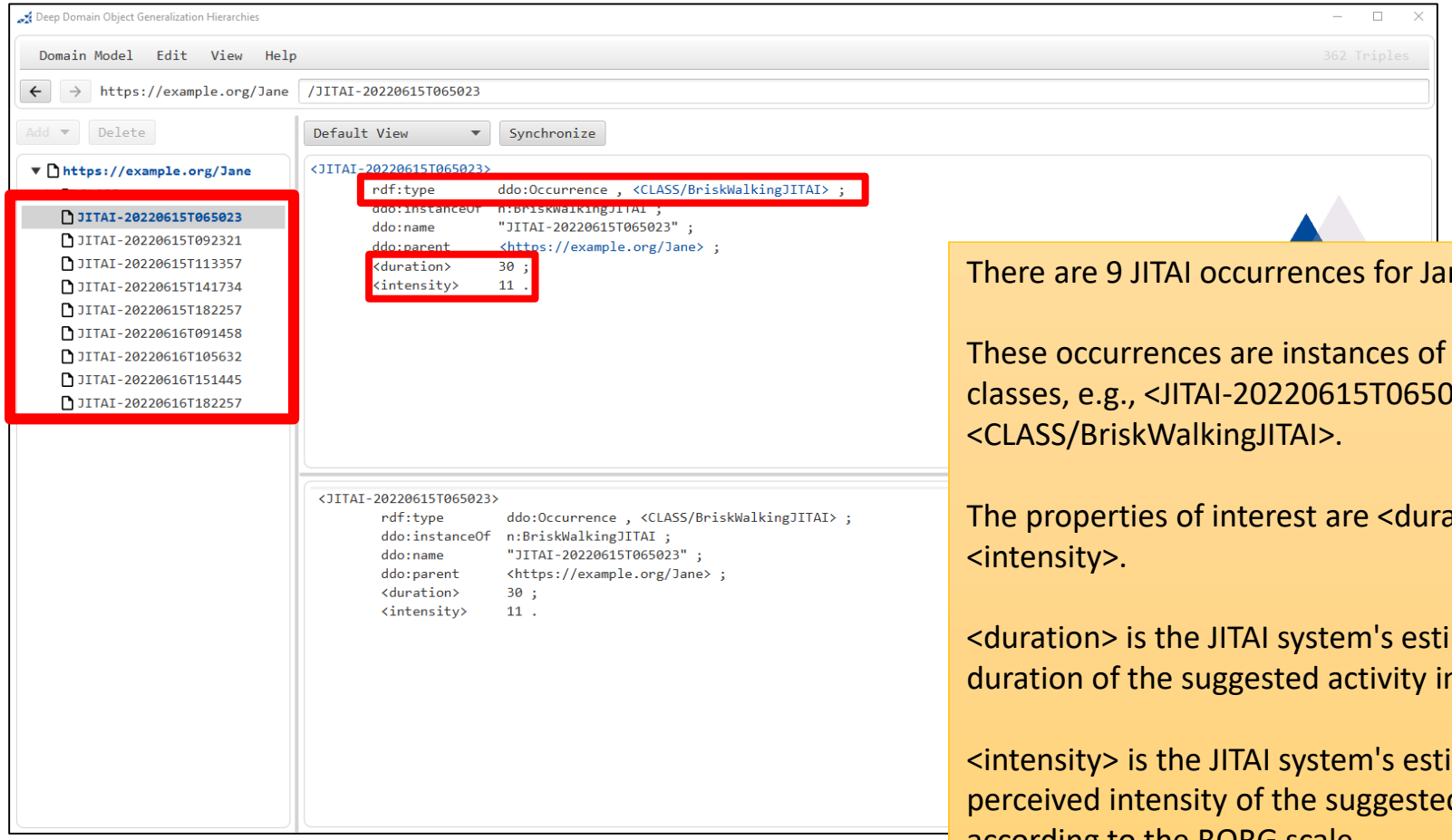
Jane‘s singleton class (<CLASS>) introduces a class hierarchy for different JITAI classes which is represented by the class hierarchy above.

The class hierarchy is encoded using rdfs:subClassOf properties.

Note: The URI of an element is related to its arrangement in the tree view.

Use Case

JITAI Occurrences



There are 9 JITAI occurrences for Jane within 2 days.

These occurrences are instances of different JITAI classes, e.g., <JITAI-20220615T065023> is instance of <CLASS/BriskWalkingJITAI>.

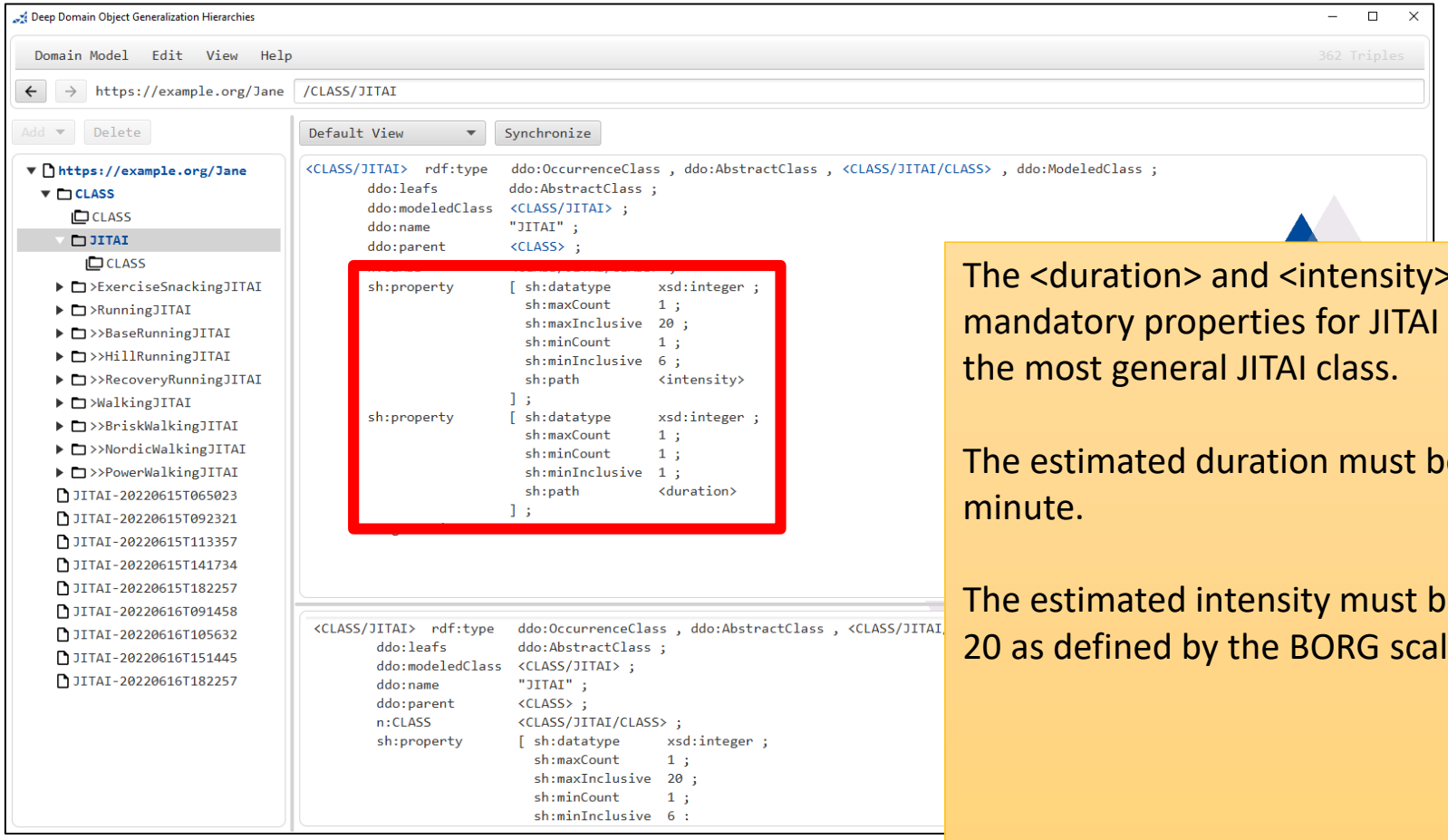
The properties of interest are <duration> and <intensity>.

<duration> is the JITAI system's estimate of the duration of the suggested activity in minutes.

<intensity> is the JITAI system's estimate of the perceived intensity of the suggested activity according to the BORG scale.

Use Case

Duration & Intensity



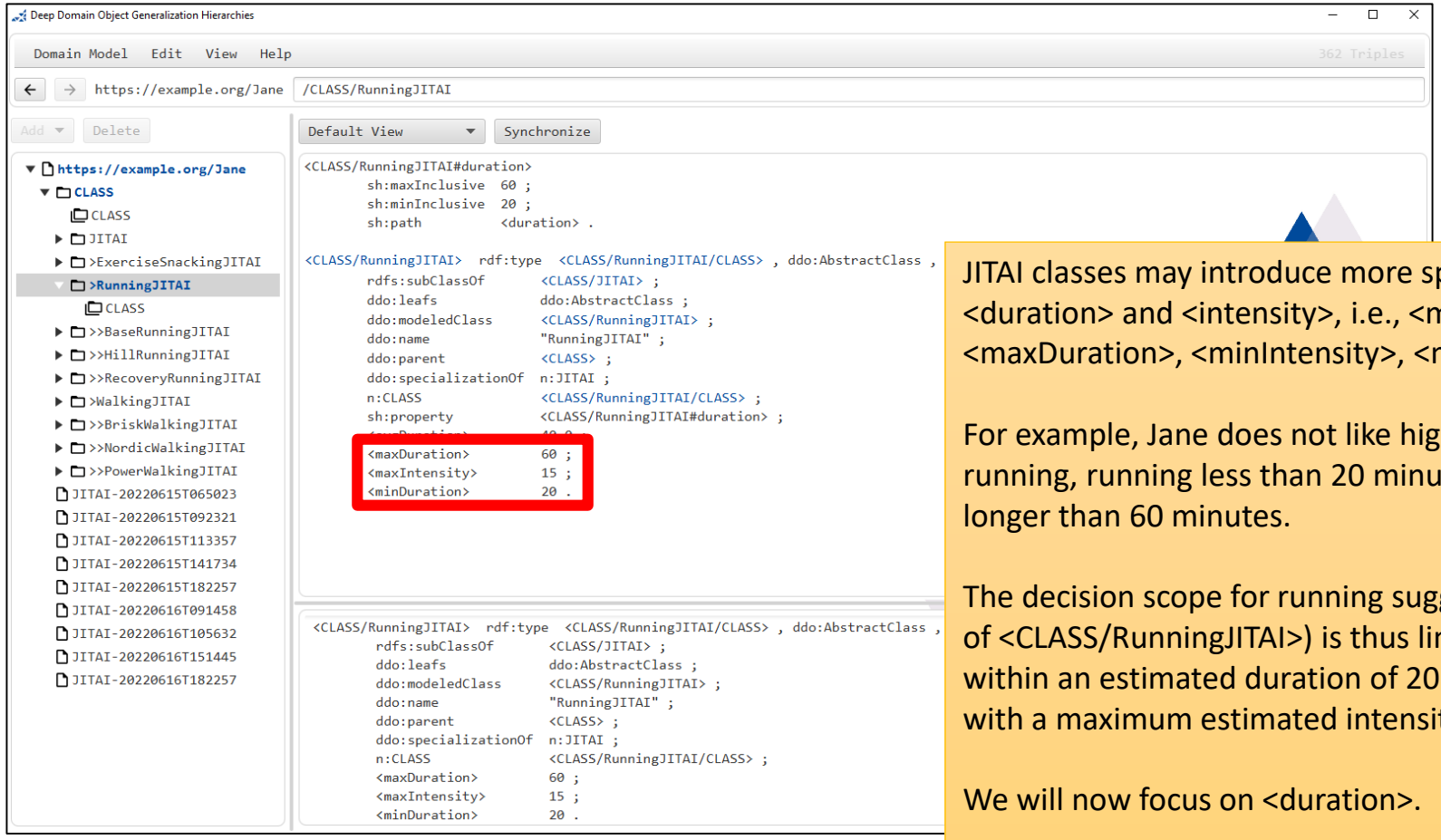
The <duration> and <intensity> are defined as mandatory properties for JITAI occurrences in the most general JITAI class.

The estimated duration must be more than 1 minute.

The estimated intensity must be within 6 and 20 as defined by the BORG scale.

Multi-level Modeling

JITAI Clabjects revisited



JITAI classes may introduce more specific limits for <duration> and <intensity>, i.e., <minDuration>, <maxDuration>, <minIntensity>, <maxIntensity>.

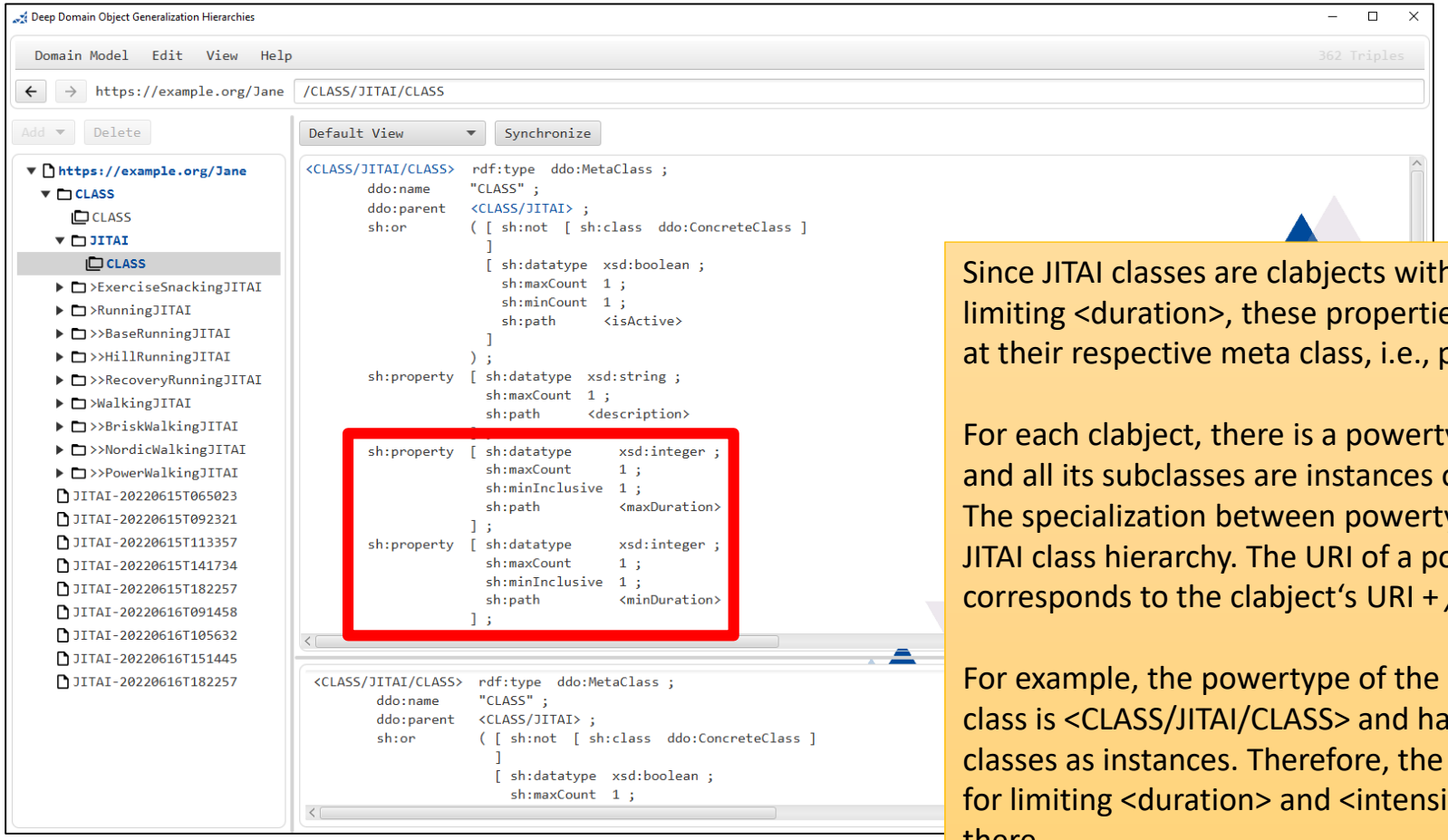
For example, Jane does not like high intensity running, running less than 20 minutes and running longer than 60 minutes.

The decision scope for running suggestions (instances of <CLASS/RunningJITAI>) is thus limited to activities within an estimated duration of 20 to 60 minutes and with a maximum estimated intensity of 15.

We will now focus on <duration>.

Multi-level Modeling

Powertype (1)

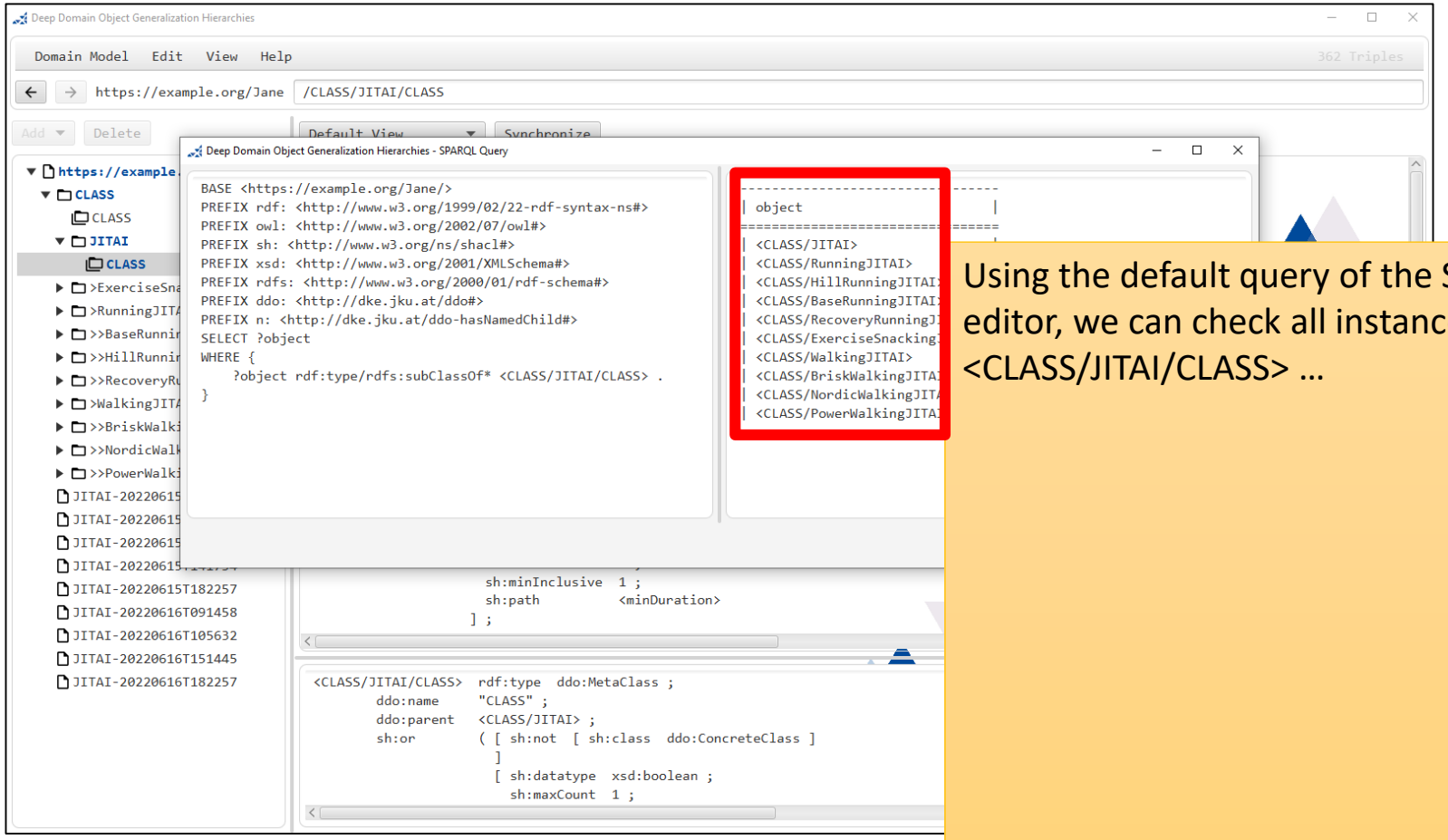


Since JITAI classes are clabjects with values for limiting <duration>, these properties are introduced at their respective meta class, i.e., powertype.

For each clabject, there is a powertype. The clabject and all its subclasses are instances of the powertype. The specialization between powertypes mirrors the JITAI class hierarchy. The URI of a powertype corresponds to the clabject‘s URI + */CLASS*.

For example, the powertype of the most general JITAI class is <CLASS/JITAI/CLASS> and has all the JITAI classes as instances. Therefore, the property shapes for limiting <duration> and <intensity> are introduced there.

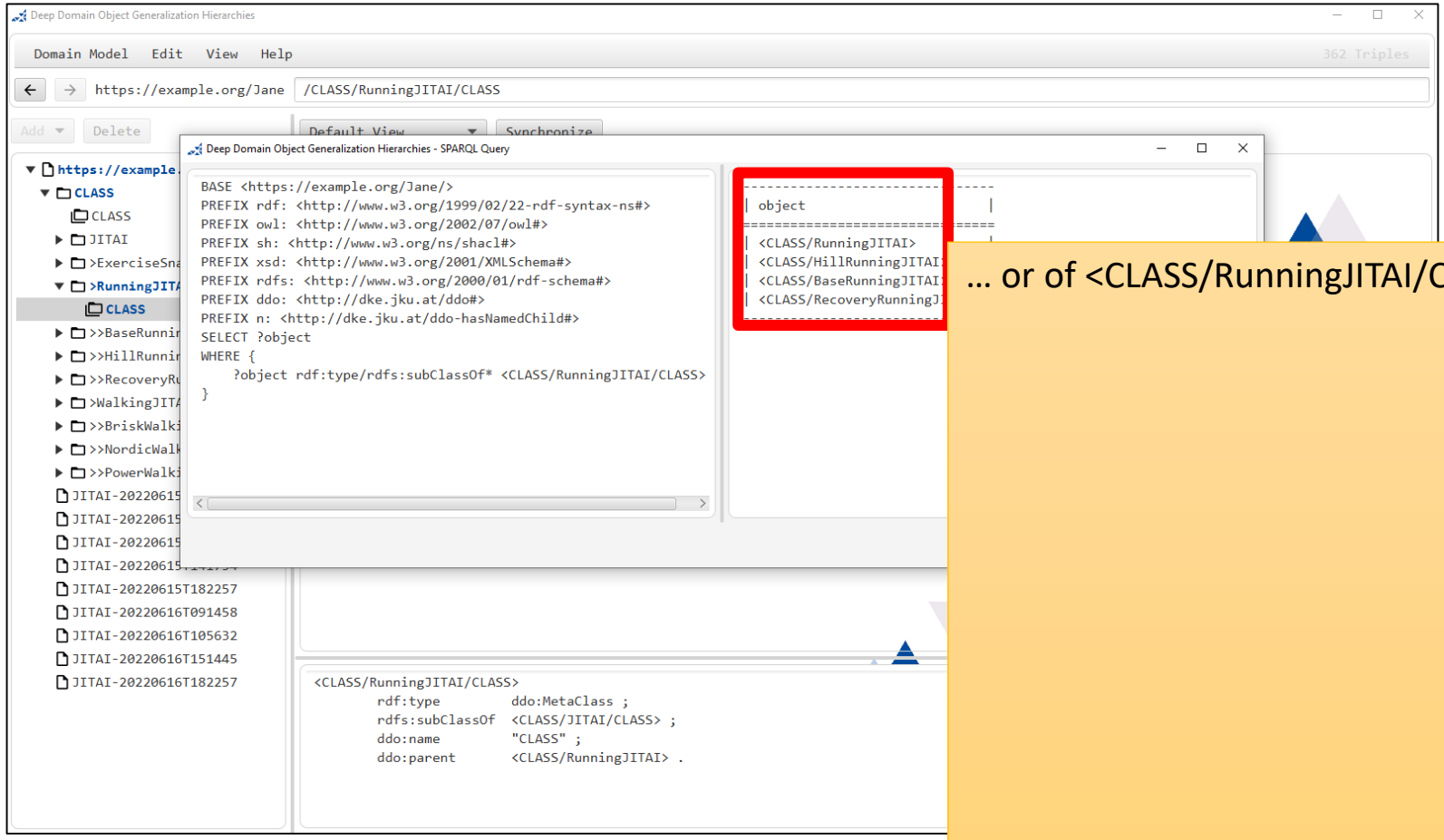
Multi-Level Modeling
Powertype (2)



Using the default query of the SPARQL query editor, we can check all instances of the <CLASS/JITAI/CLASS> …

Multi-Level Modeling

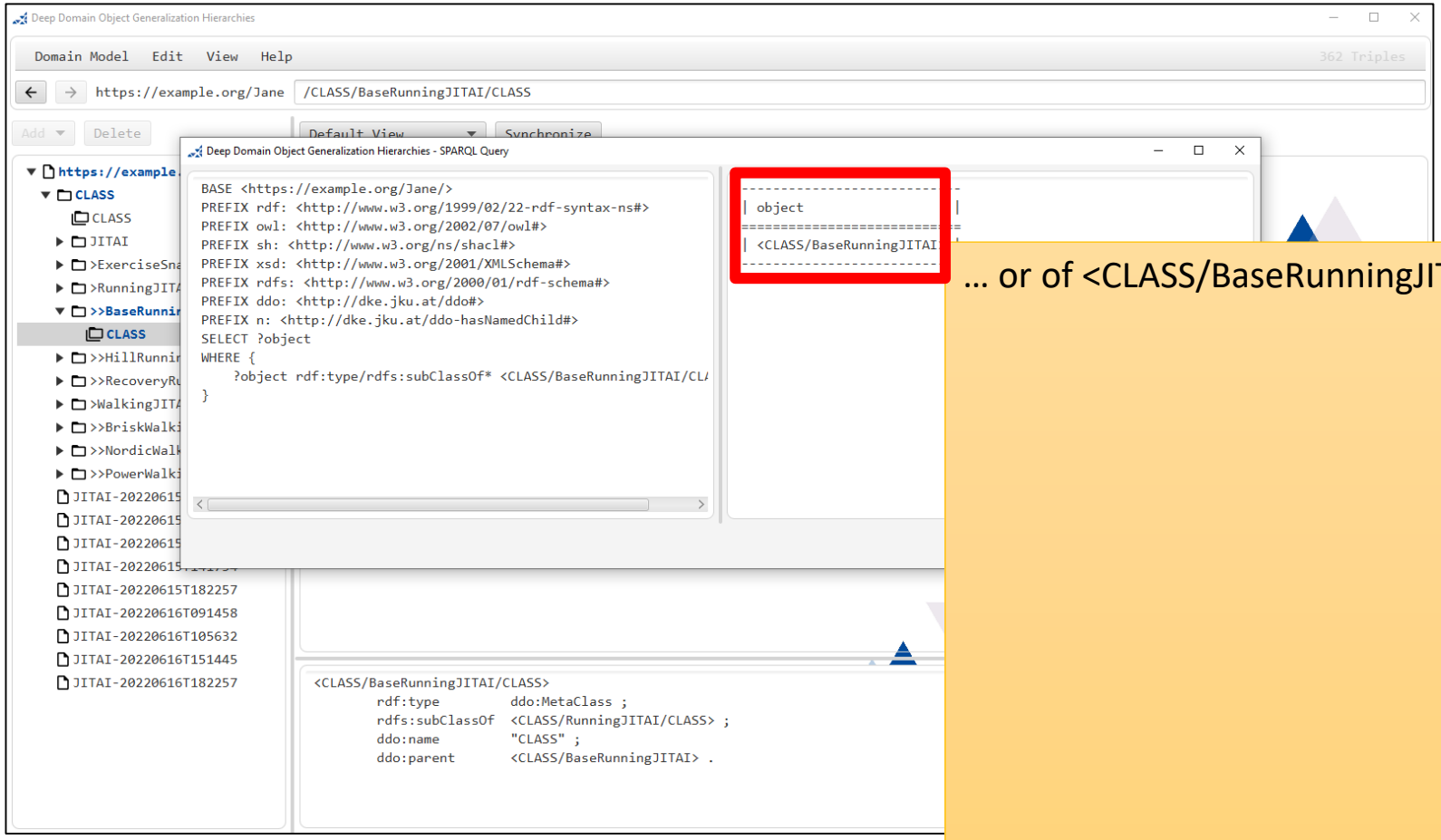
Powertype (3)



… or of <CLASS/RunningJITAI/CLASS>.

Multi-Level Modeling

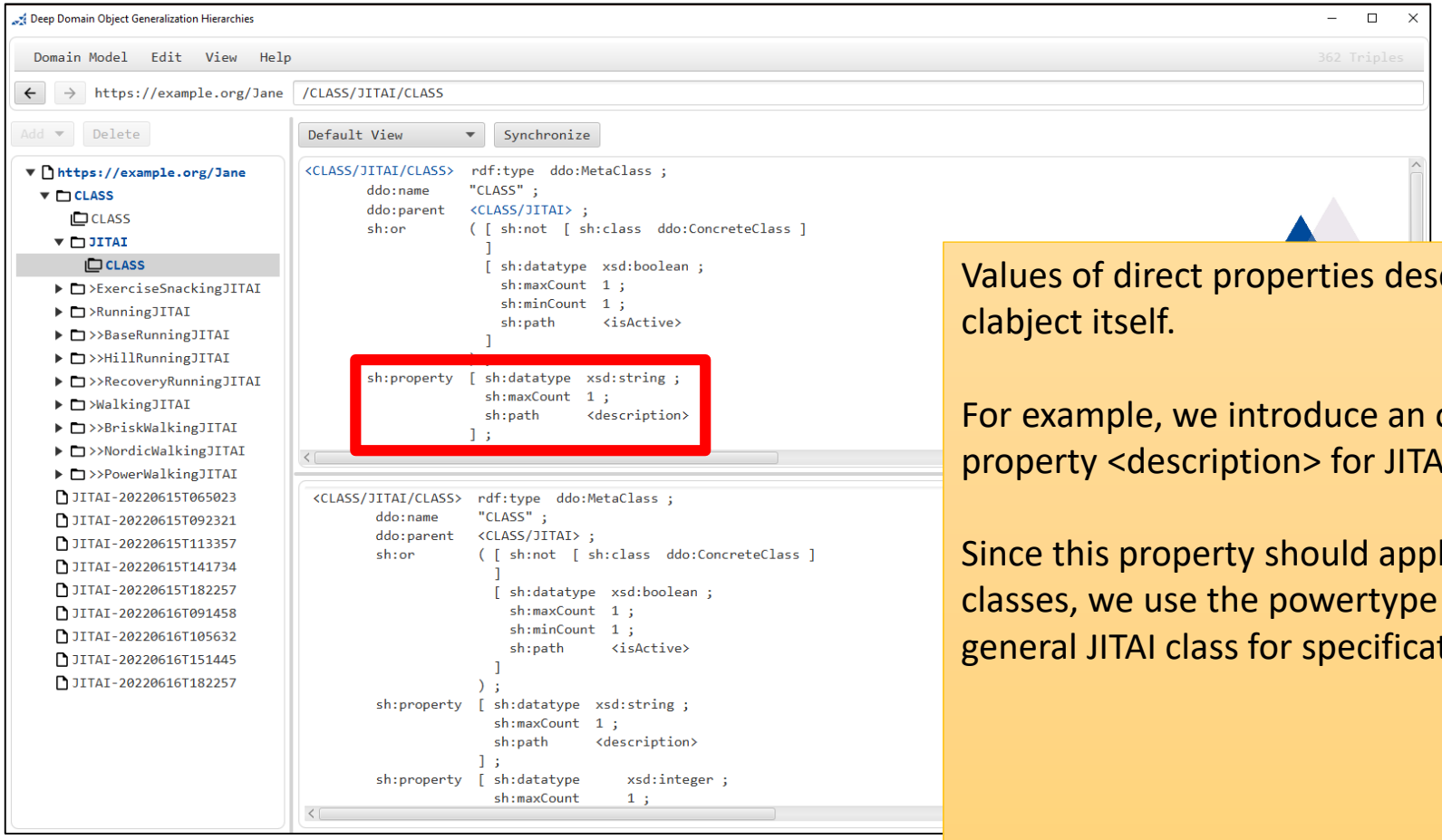
Powertype (4)



… or of <CLASS/BaseRunningJITAI/CLASS>.

Multi-Level Modeling

Direct Properties (1)



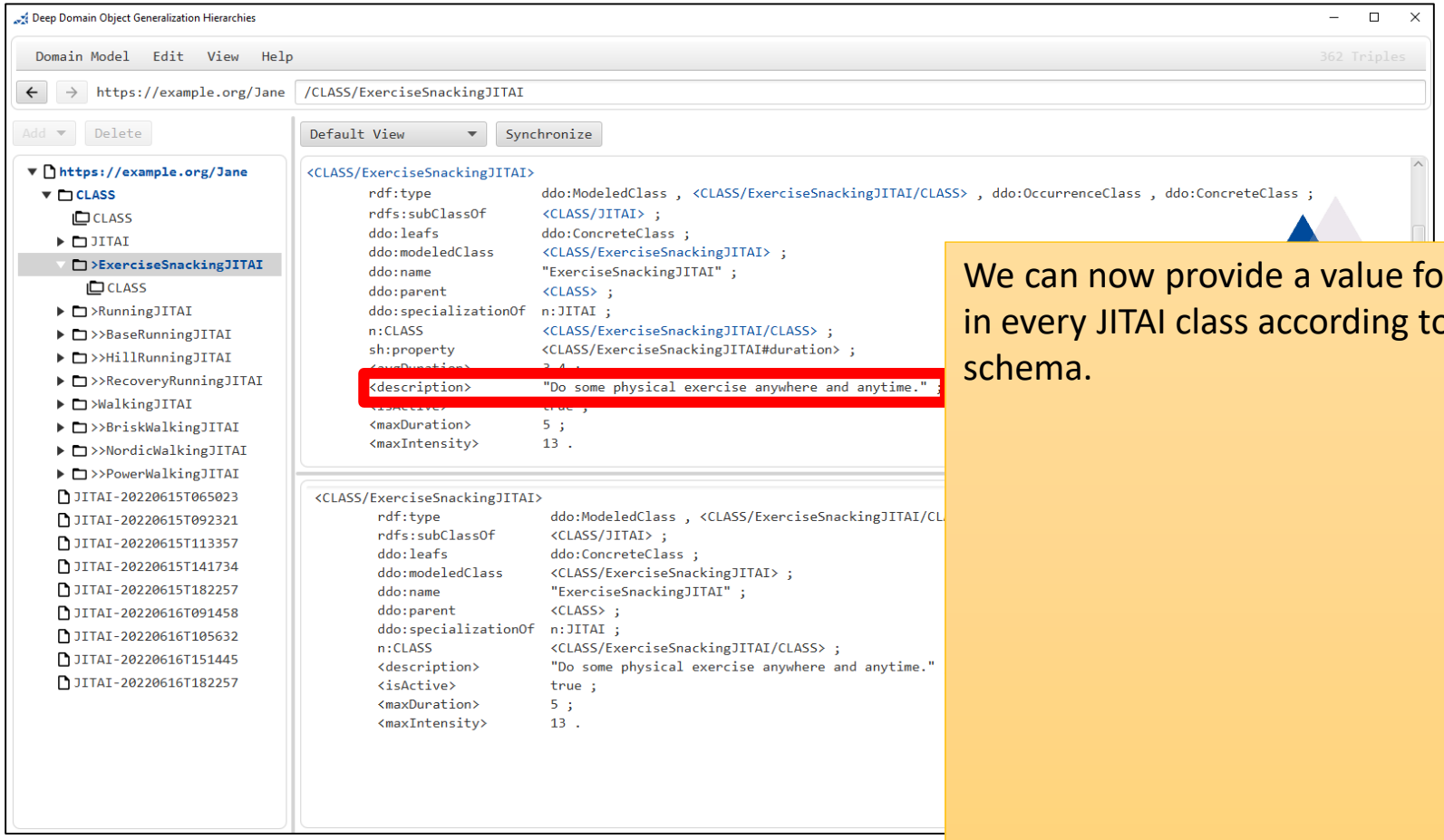
Values of direct properties describe the clabject itself.

For example, we introduce an optional direct property <description> for JITAI classes.

Since this property should apply to all JITAI classes, we use the powertype of the most general JITAI class for specification.

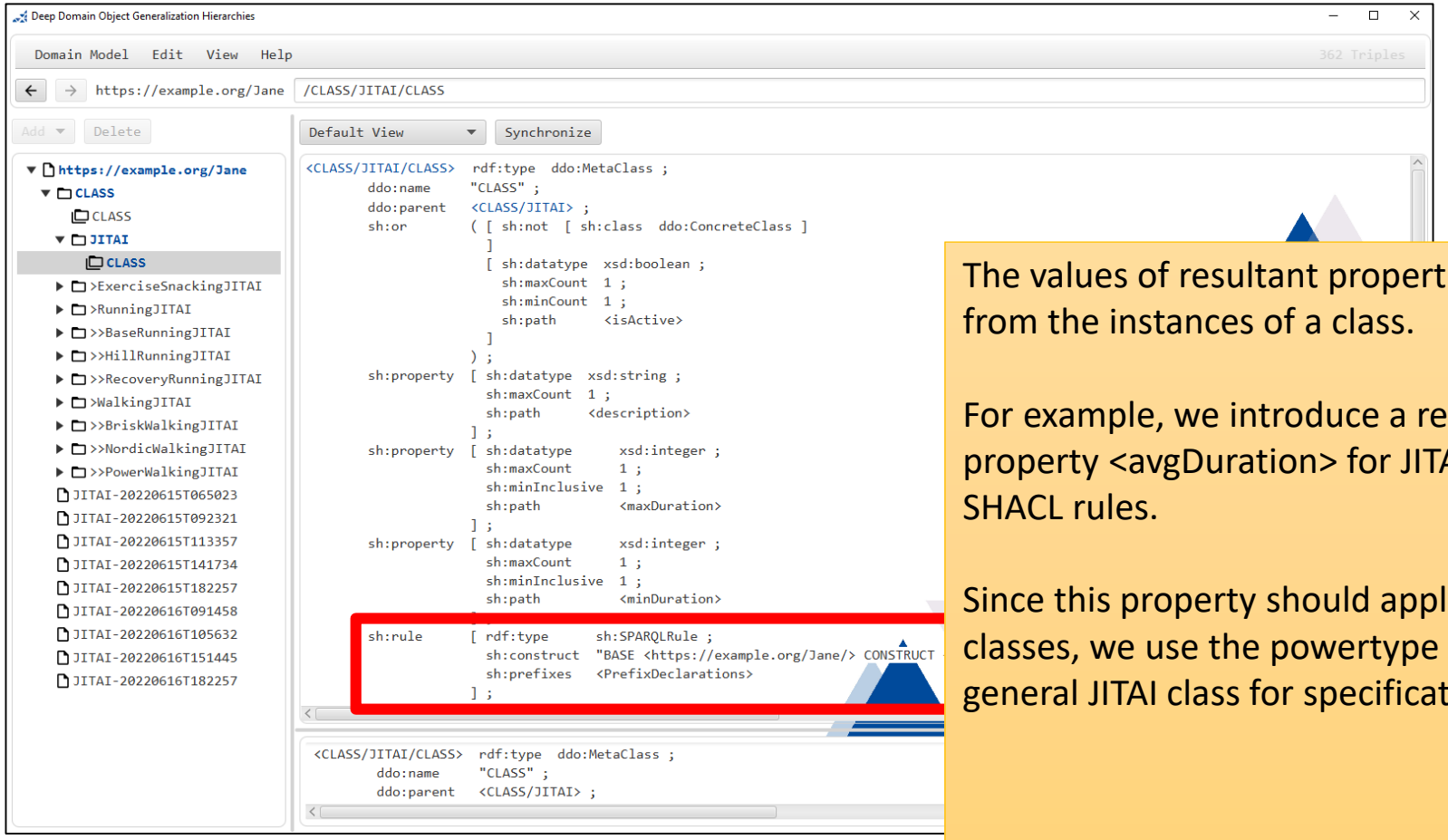
Multi-Level Modeling

Direct Properties (2)



We can now provide a value for this property in every JITAI class according to the specified schema.

Resultant Properties (1)



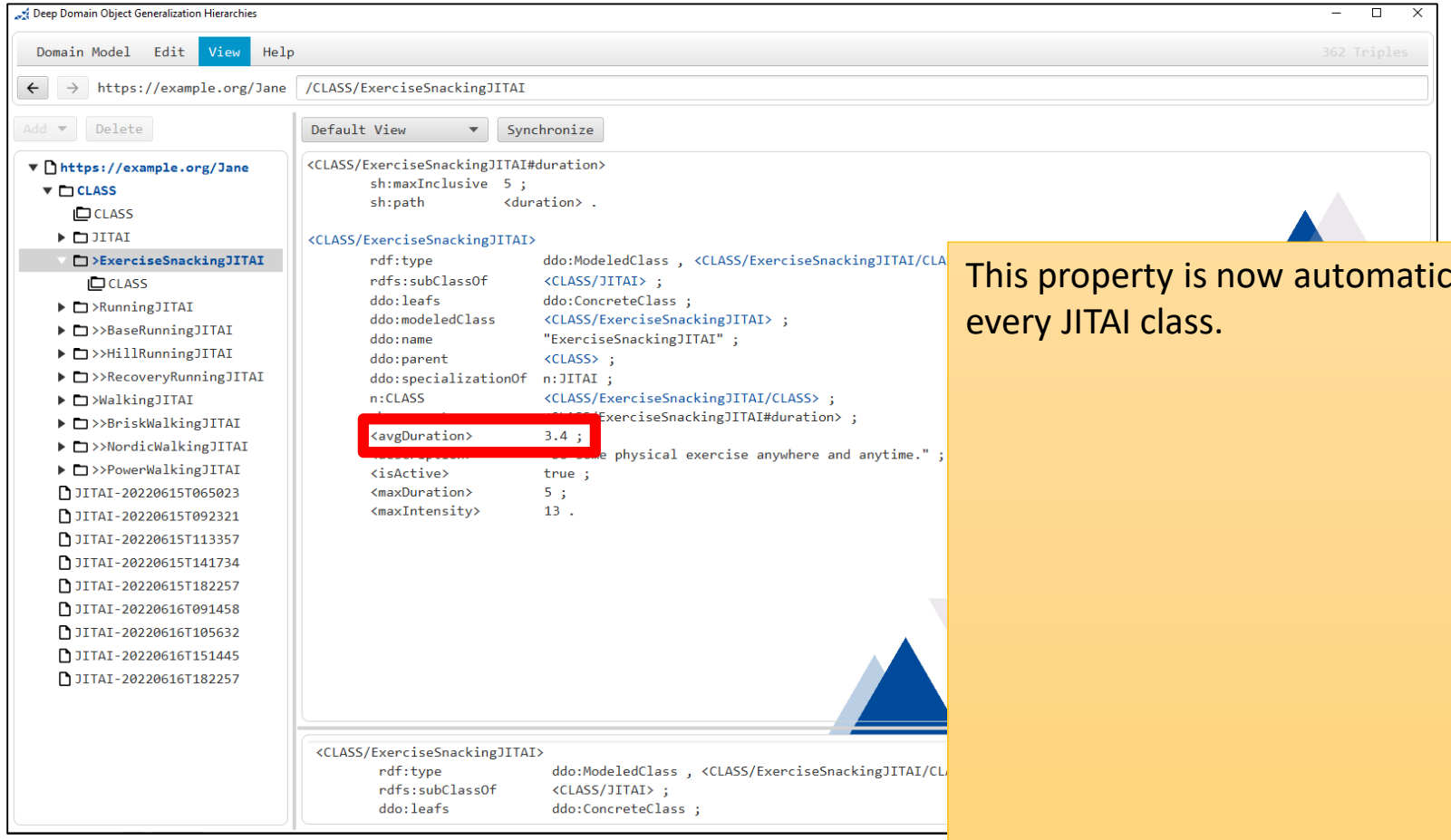
The values of resultant properties are derived from the instances of a class.

For example, we introduce a resultant property <avgDuration> for JITAI classes using SHACL rules.

Since this property should apply to all JITAI classes, we use the powertype of the most general JITAI class for specification.

Multi-Level Modeling

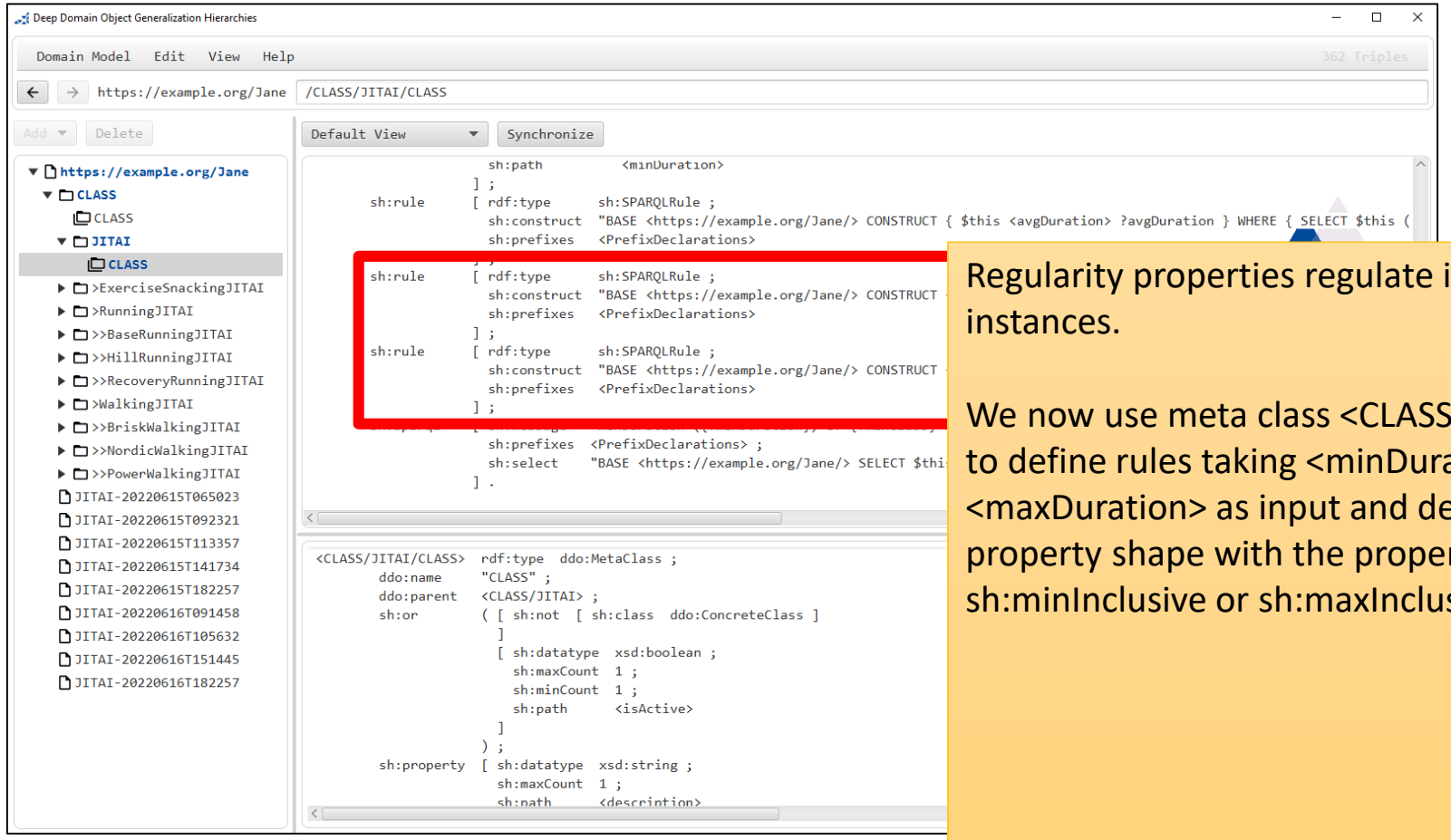
Resultant Properties (2)



This property is now automatically derived in every JITAI class.

Multi-Level Modeling

Regularity Properties (1)

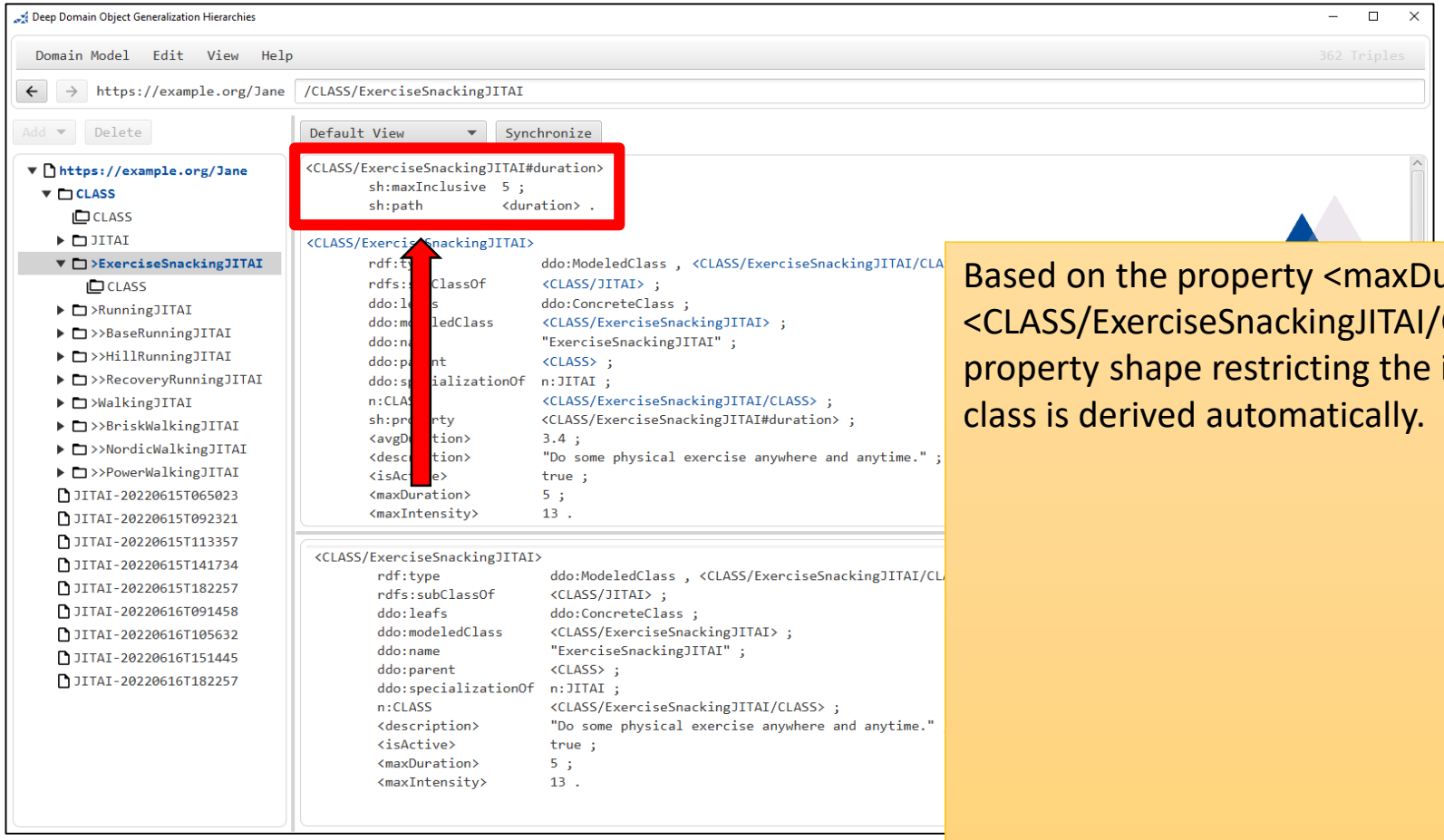


Regularity properties regulate instances of the instances.

We now use meta class <CLASS/JITAI/CLASS> to define rules taking <minDuration> and <maxDuration> as input and deriving a property shape with the property‘s value as sh:minInclusive or sh:maxInclusive.

Multi-Level Modeling

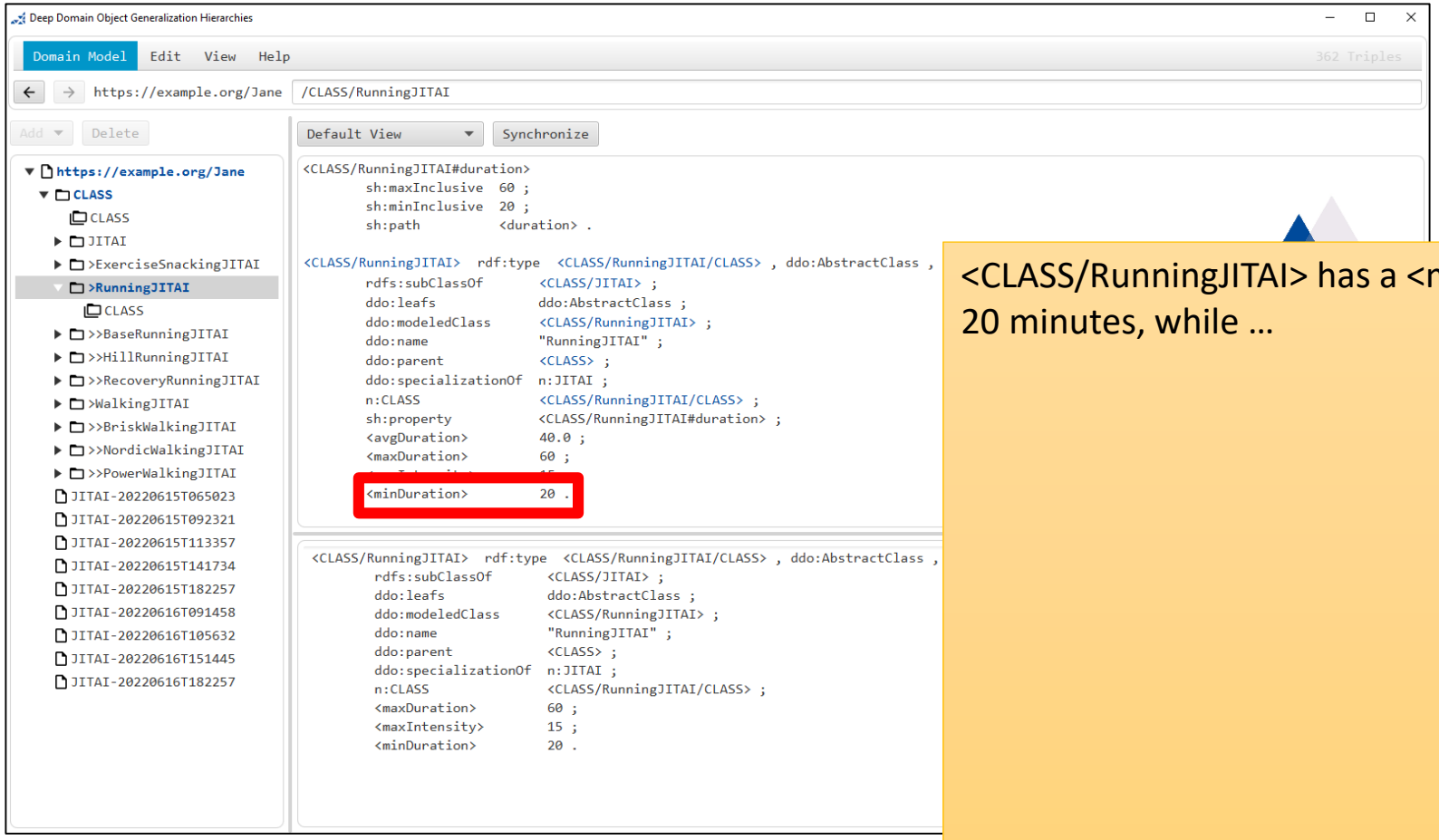
Regularity Properties (2)



Based on the property <maxDuration> in <CLASS/ExerciseSnackingJITAI/CLASS> a property shape restricting the instances of this class is derived automatically.

Multi-Level Modeling

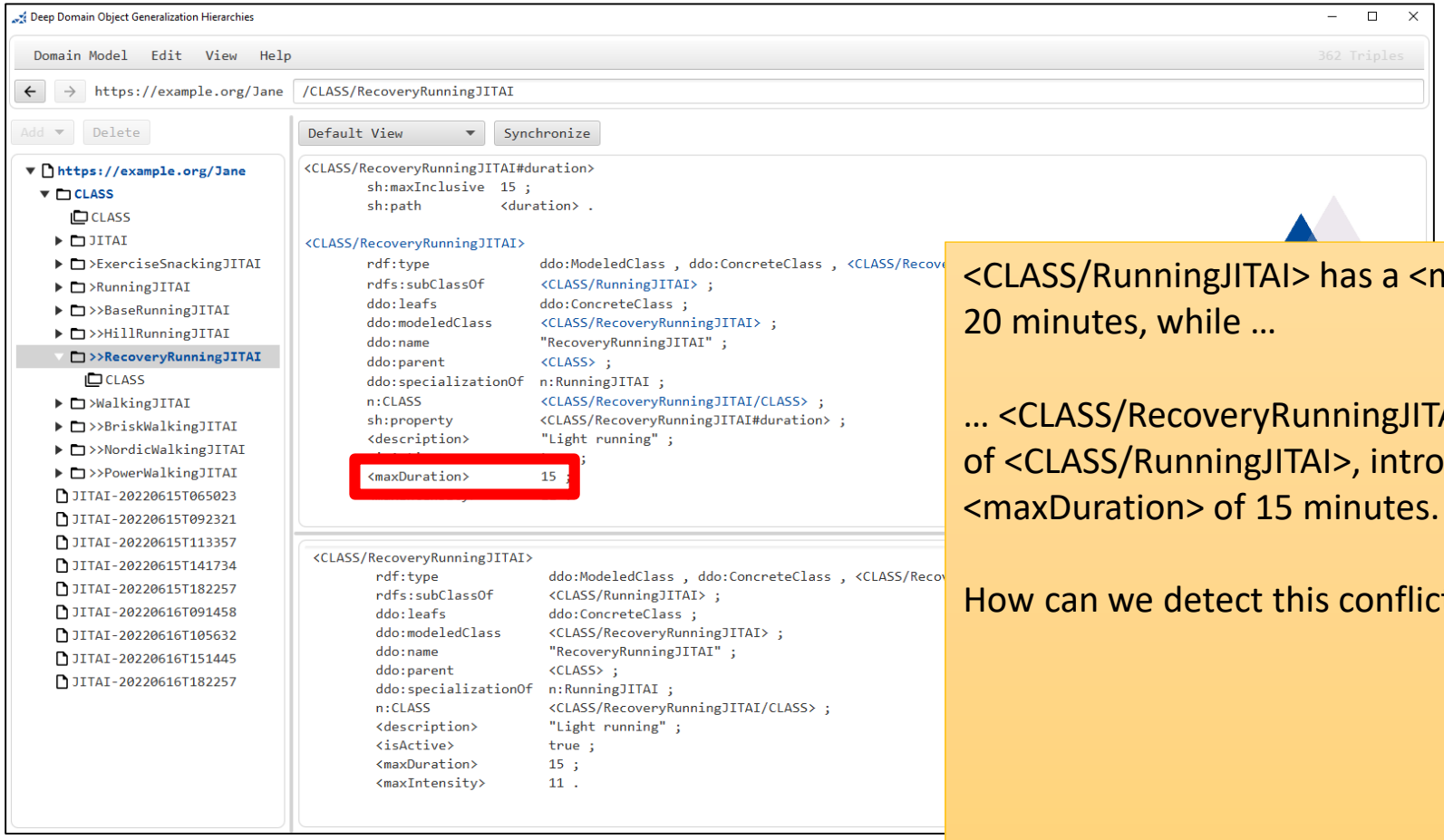
Conflict Detection (1)



<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while …

Multi-Level Modeling

Conflict Detection (2)



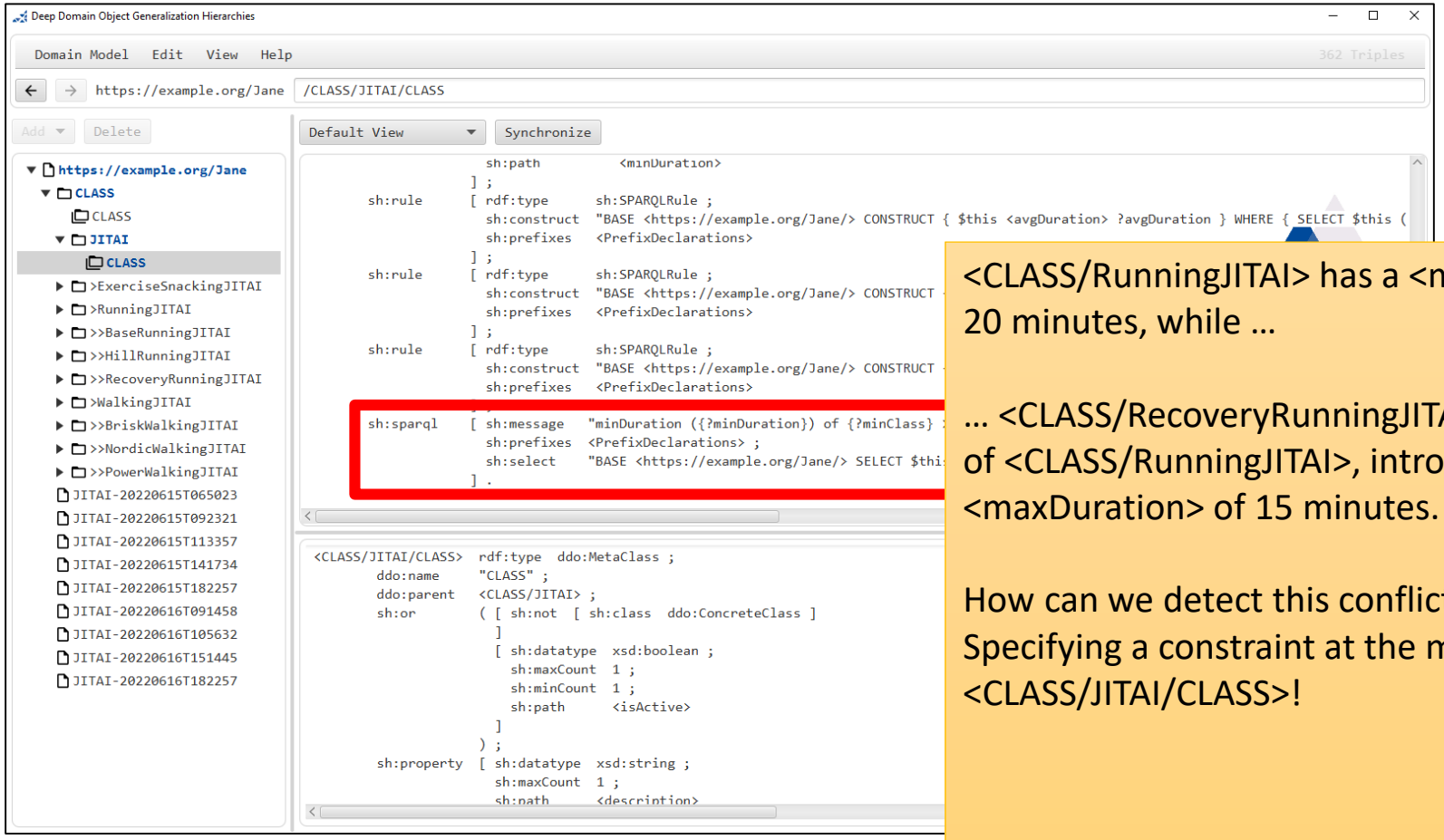
<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while …

… <CLASS/RecoveryRunningJITAI>, a subclass of <CLASS/RunningJITAI>, introduces a <maxDuration> of 15 minutes.

How can we detect this conflict?

Multi-Level Modeling

Conflict Detection (3)



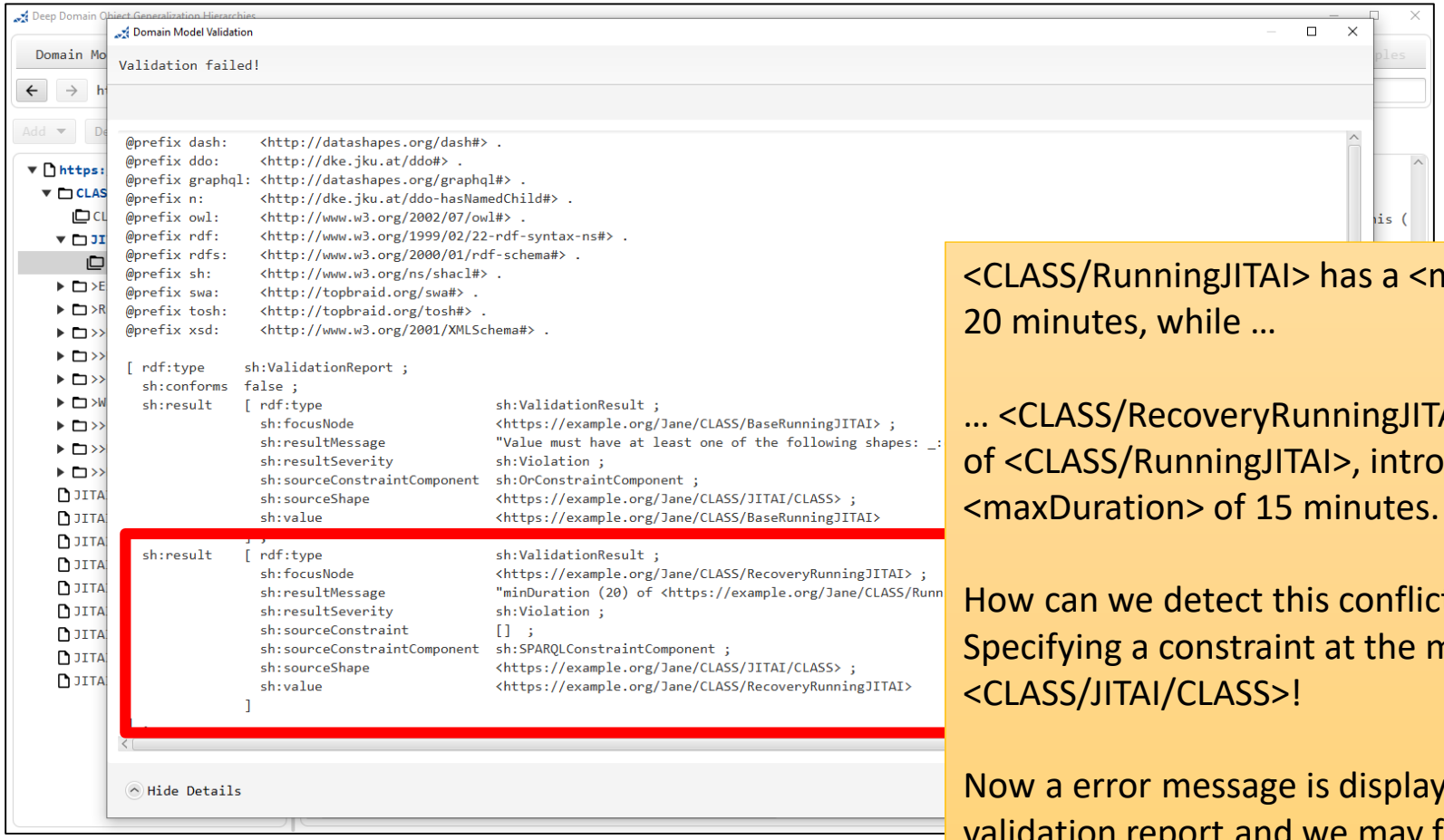
<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while …

… <CLASS/RecoveryRunningJITAI>, a subclass of <CLASS/RunningJITAI>, introduces a <maxDuration> of 15 minutes.

How can we detect this conflict?
Specifying a constraint at the meta class <CLASS/JITAI/CLASS>!

Multi-Level Modeling

Conflict Detection (4)



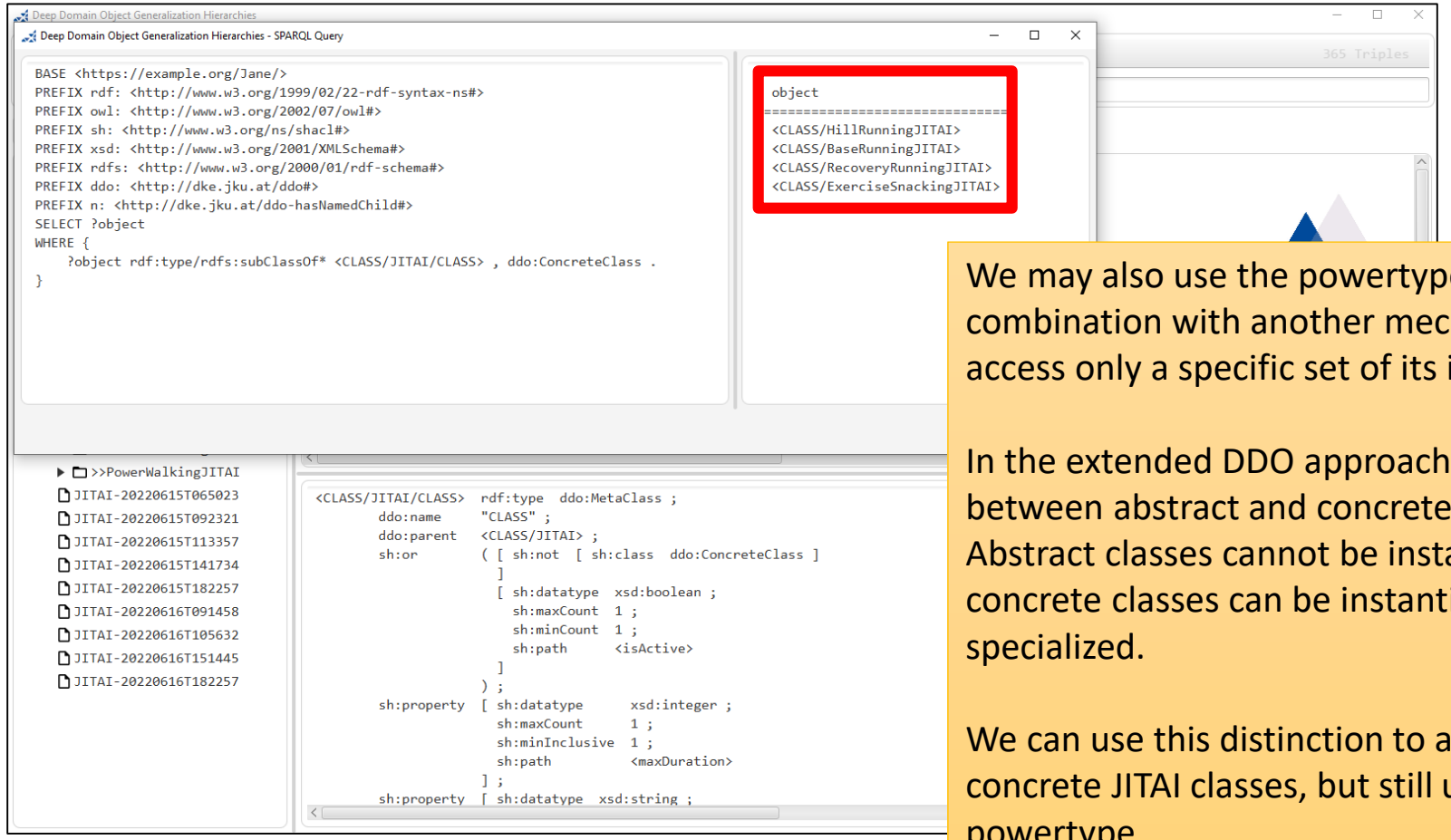
<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while …

… <CLASS/RecoveryRunningJITAI>, a subclass of <CLASS/RunningJITAI>, introduces a <maxDuration> of 15 minutes.

How can we detect this conflict?
Specifying a constraint at the meta class <CLASS/JITAI/CLASS>!

Now a error message is displayed in the validation report and we may fix the problem.

Meta Class Combinations (1)

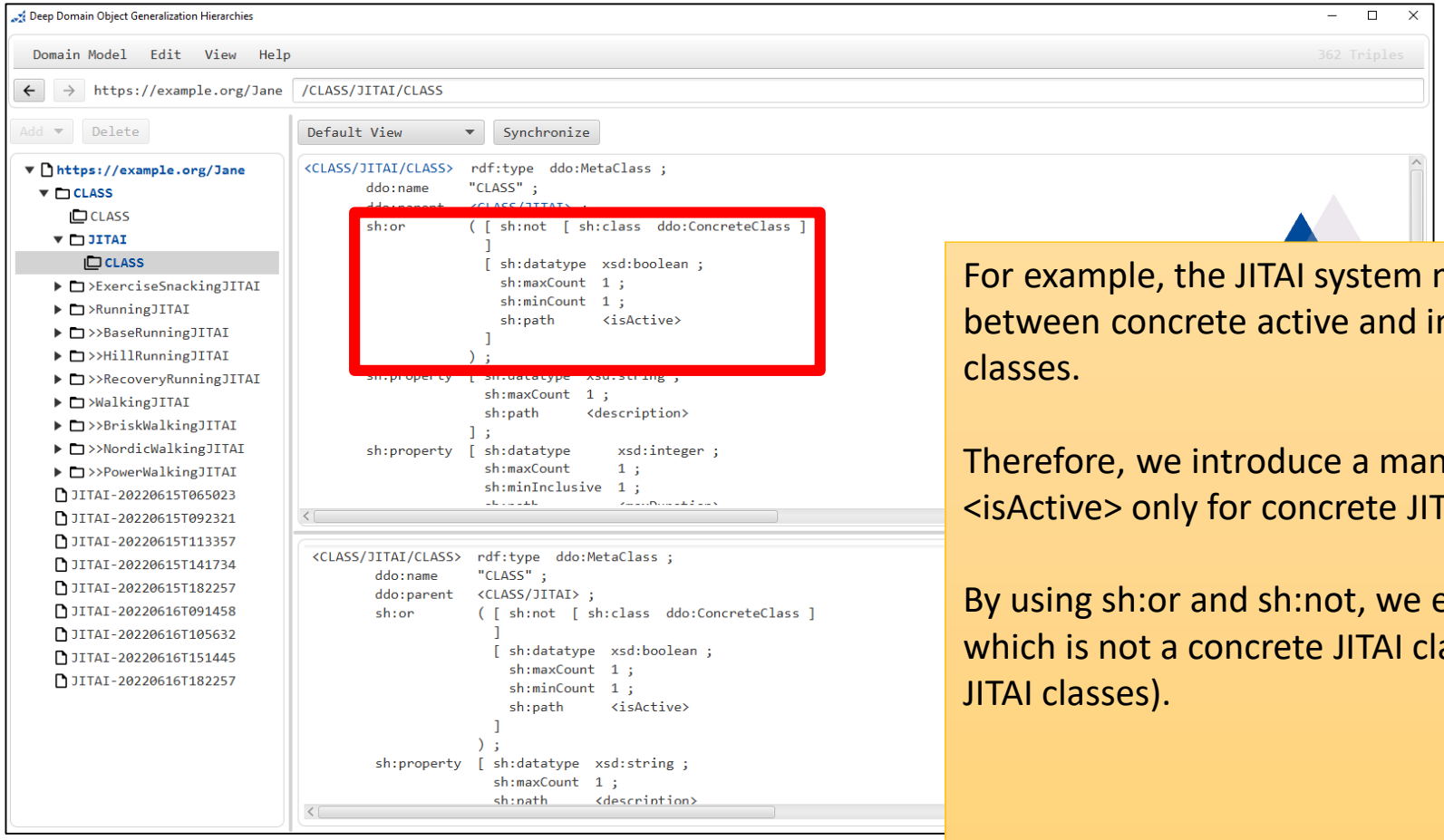


We may also use the powertype in combination with another mechanism to access only a specific set of its instances.

In the extended DDO approach, we distinguish between abstract and concrete classes. Abstract classes cannot be instantiated, whilst concrete classes can be instantiated but not specialized.

We can use this distinction to access only concrete JITAI classes, but still use the powertype.

Meta Class Combinations (2)

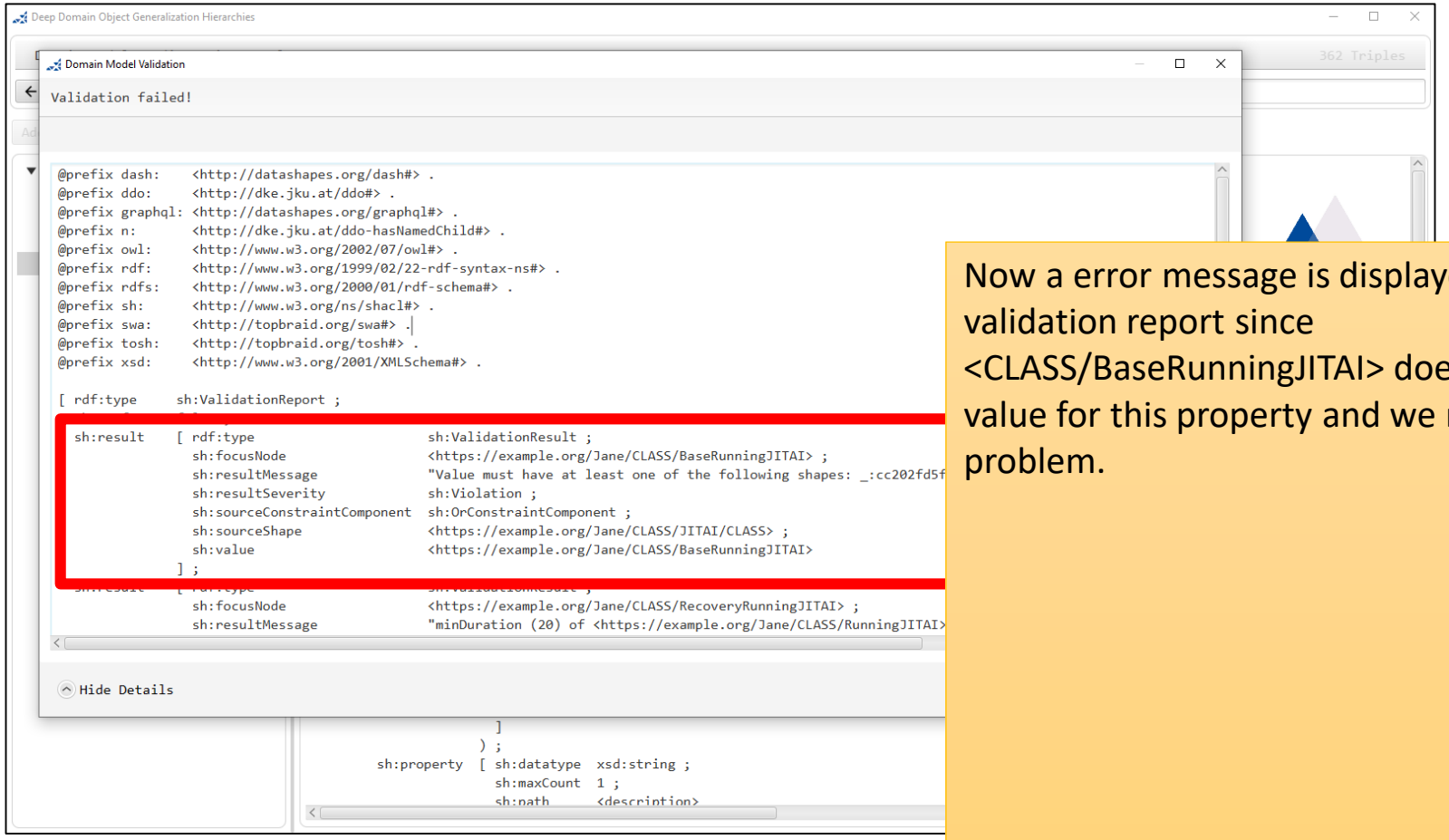


For example, the JITAI system must distinguish between concrete active and inactive JITAI classes.

Therefore, we introduce a mandatory property <isActive> only for concrete JITAI classes.

By using sh:or and sh:not, we exclude any class which is not a concrete JITAI class (i.e., abstract JITAI classes).

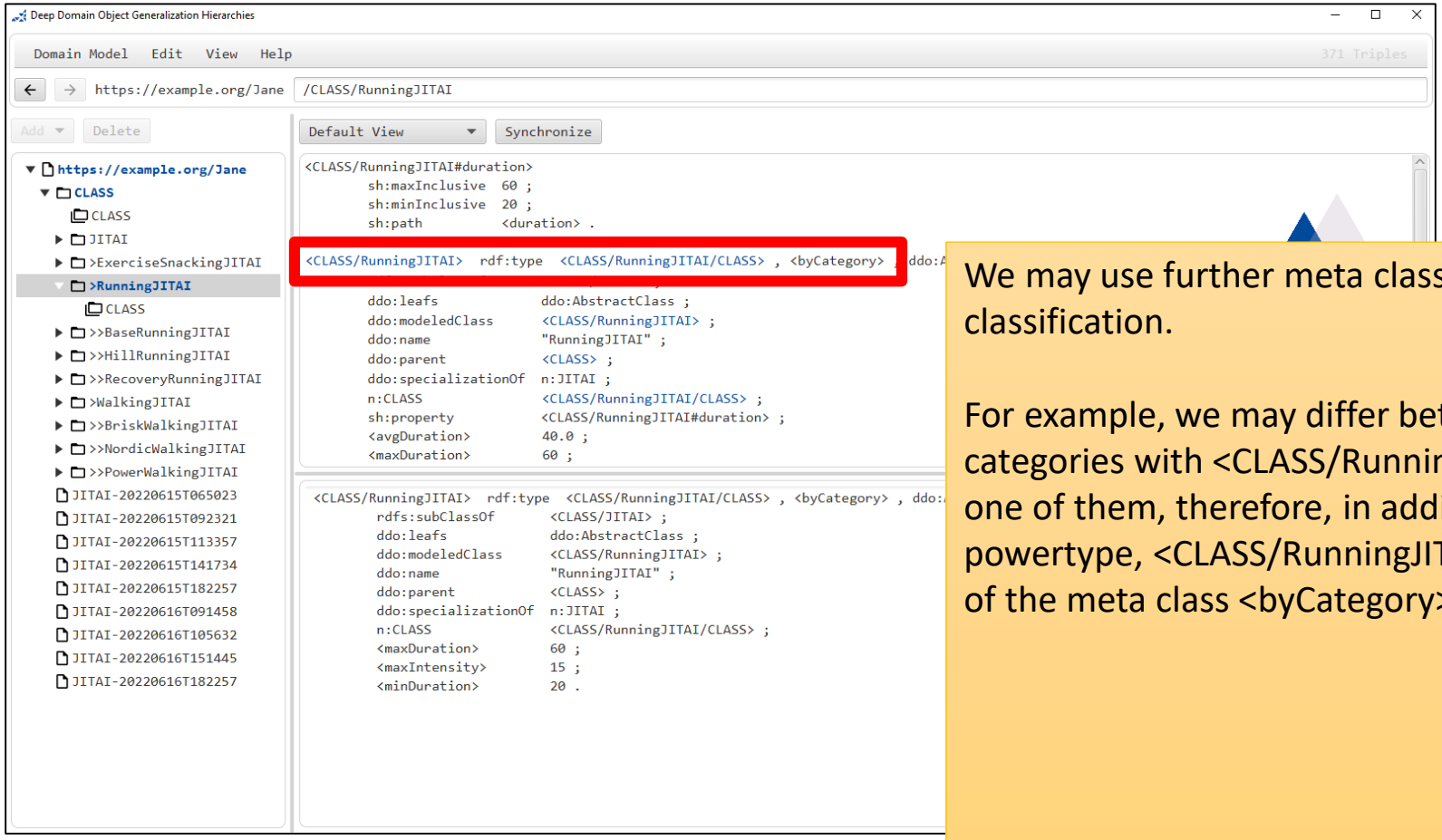
Meta Class Combinations (3)



Now a error message is displayed in the validation report since <CLASS/BaseRunningJITAI> does not provide a value for this property and we may fix the problem.

Multi-Level Modeling

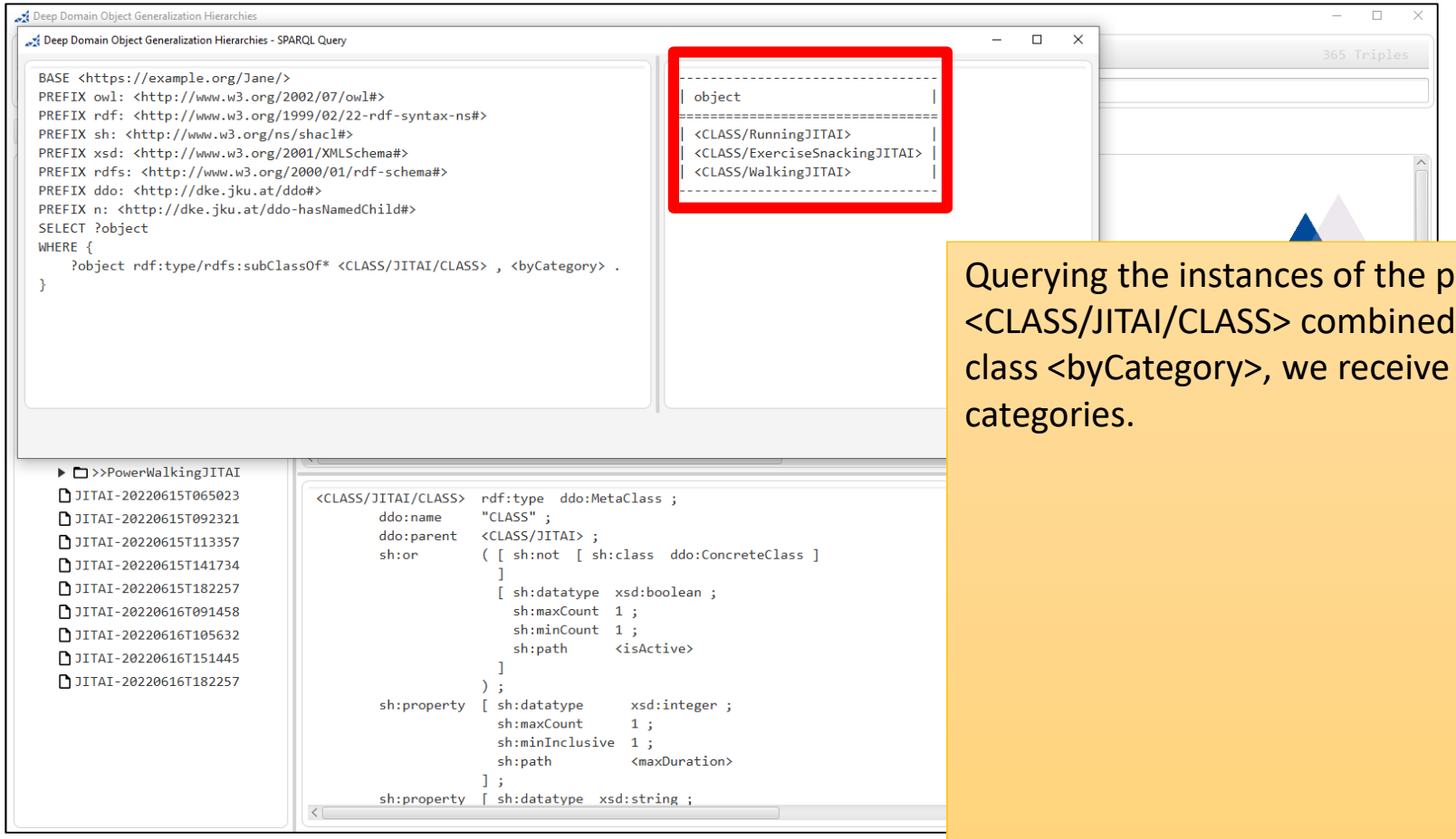
Meta Class Combinations (4)



We may use further meta classes for clabject classification.

For example, we may differ between 3 JITAI categories with <CLASS/RunningJITAI> being one of them, therefore, in addition to its powertype, <CLASS/RunningJITAI> is instance of the meta class <byCategory>.

Meta Class Combinations (5)



Querying the instances of the powertype <CLASS/JITAI/CLASS> combined with the meta class <byCategory>, we receive all JITAI categories.

Task

Introduction

- Before heading over to the task, take a moment, browse the model and especially consider rdf:type and rdfs:subClassOf relationships between elements.

Task

Subtask 1

- Define a SPARQL query that queries the direct and indirect instances of metaclass <CLASS/JITAI/CLASS> together with their descriptions and average duration.
- Expected result

class	description	avgDuration
<CLASS/JITAI>		16.89
<CLASS/RunningJITAI>		40.0
<CLASS/HillRunningJITAI>	"Vigorous running."	
<CLASS/BaseRunningJITAI>	"Moderate running."	40.0
<CLASS/RecoveryRunningJITAI>	"Light running"	
<CLASS/ExerciseSnackingJITAI>	"Do some physical exercise anywhere and anytime."	3.4
<CLASS/WalkingJITAI>		31.67
<CLASS/NordicWalkingJITAI>	"Walking with sticks."	
<CLASS/BriskWalkingJITAI>	"Moderate walking."	31.67
<CLASS/PowerWalkingJITAI>	"Vigorous walking."	

Task

Subtask 2

- Define a SPARQL query that counts for each JITAI class the number of individual JITAI occurrences (considering direct and indirect instances and also considering JITAI classes that have no instances).
- Use the appropriate meta class in your query.
- Expected result

class	numberOfJITAIs
<CLASS/NordicWalkingJITAI>	0
<CLASS/HillRunningJITAI>	0
<CLASS/BriskWalkingJITAI>	3
<CLASS/JITAI>	9
<CLASS/PowerWalkingJITAI>	0
<CLASS/BaseRunningJITAI>	1
<CLASS/RecoveryRunningJITAI>	0
<CLASS/WalkingJITAI>	3
<CLASS/RunningJITAI>	1
<CLASS/ExerciseSnackingJITAI>	5

Task

Subtask 3

- Define a SPARQL query that selects WalkingJITAI classes (use the appropriate meta class) that are also instances of meta class <byIntensity>.
- Expected result

```
------------------------------
| class                      |
==============================
| <CLASS/NordicWalkingJITAI> |
| <CLASS/BriskWalkingJITAI>  |
| <CLASS/PowerWalkingJITAI>  |
------------------------------
```

Task

Subtask 4

- Define a SPARQL query that identifies for each JITAI class the limitations on <duration> made by SHACL property shapes using sh:minInclusive and sh:maxInclusive.
- Expected result

class	minInclusive	maxInclusive
<CLASS/JITAI>	1	
<CLASS/RunningJITAI>	20	60
<CLASS/HillRunningJITAI>		
<CLASS/BaseRunningJITAI>		
<CLASS/RecoveryRunningJITAI>		15
<CLASS/ExerciseSnackingJITAI>		5
<CLASS/WalkingJITAI>	10	60
<CLASS/NordicWalkingJITAI>		
<CLASS/BriskWalkingJITAI>		
<CLASS/PowerWalkingJITAI>		

Task

Subtask 5

- We have introduced various concepts of MLM using <duration>. Now we focus on regularity properties for <intensity>.
- Add SHACL property shapes for <minIntensity> and for <maxIntensity> at the appropriate powertype, specifying that <minIntensity> and <maxIntensity> have multiplicity 0..1 and datatype integer.
- You can evaluate these constraints by manually violating them.

Task

Subtask 6

- Add SHACL rules for transforming a JITAI class‘ value for <minIntensity> and <maxIntensity> into property shapes with sh:minInclusive and sh:maxInclusive.
- You may use the rules that transform <minDuration> and <maxDuration> into property shapes as a template.
- Expected result, e.g., for <CLASS/RunningJITAI>

Task

Subtask 7

- Compute the average intensity <avgIntensity> as resultant property for each JITAI class by defining a SHACL rule at the appropriate meta class.
- You may use the rule that computes <avgDuration> as a template.
- Expected result, e.g., for <CLASS/ExerciseSnackingJITAI>

Task

Subtask 8

- Detect conflicts between <minIntensity> and <maxIntensity> within the class hierarchy using SHACL SPARQL-based constraints at the appropriate meta class.
- You may use the shape that detects conflicts between <minDuration> and <maxDuration> as a template.
- Expected result (additional validation result, Domain Model > Validate …)

Evaluation

- There is a voluntary and anonymized survey, you are kindly invited to participate after completing the tutorial and working through the task.
- Even if you could not successfully solve all subtasks, any feedback is highly important and appreciated.
- Here is the link (< 5min):
  - https://survey.jku.at/jku2018/index.php/365873?lang=en
- **Thank you!**

Contact

- If you have questions about the survey, encounter bugs or have other technical problems, you can contact
  - sebastian.gruber@dhp.lbg.ac.at
  - sebastian.gruber@salzburgresearch.at

References

Almeida, J. P. A., Carvalho, V. A., Fonseca, C. M., & Guizzardi, G. (2021, October). A Note on Properties in Multi-Level Modeling. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 497-501). IEEE.

Carvalho, V. A., & Almeida, J. P. A. (2018). Toward a well-founded theory for multi-level conceptual modeling. *Software & Systems Modeling*, *17*(1), 205-231.

Gruber, S., Neumayr, B., Schrefl, M., & Niebauer, J. (2021, October). Towards Multi-level Modeling of Just-in-Time Adaptive Interventions (JITAIs) in Mobile Health. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 541-545). IEEE.

Hardeman, W., Houghton, J., Lane, K., Jones, A., & Naughton, F. (2019). A systematic review of just-in-time adaptive interventions (JITAIs) to promote physical activity. *International Journal of Behavioral Nutrition and Physical Activity*, *16*(1), 1-21.

Nahum-Shani, I., Smith, S. N., Spring, B. J., Collins, L. M., Witkiewitz, K., Tewari, A., & Murphy, S. A. (2018). Just-in-time adaptive interventions (JITAIs) in mobile health: key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, *52*(6), 446-462.

Neumayr, B., & Schrefl, M. (2022). Domain object hierarchies inducing multi-level models. *Software and Systems Modeling*, *21*(2), 587-621.