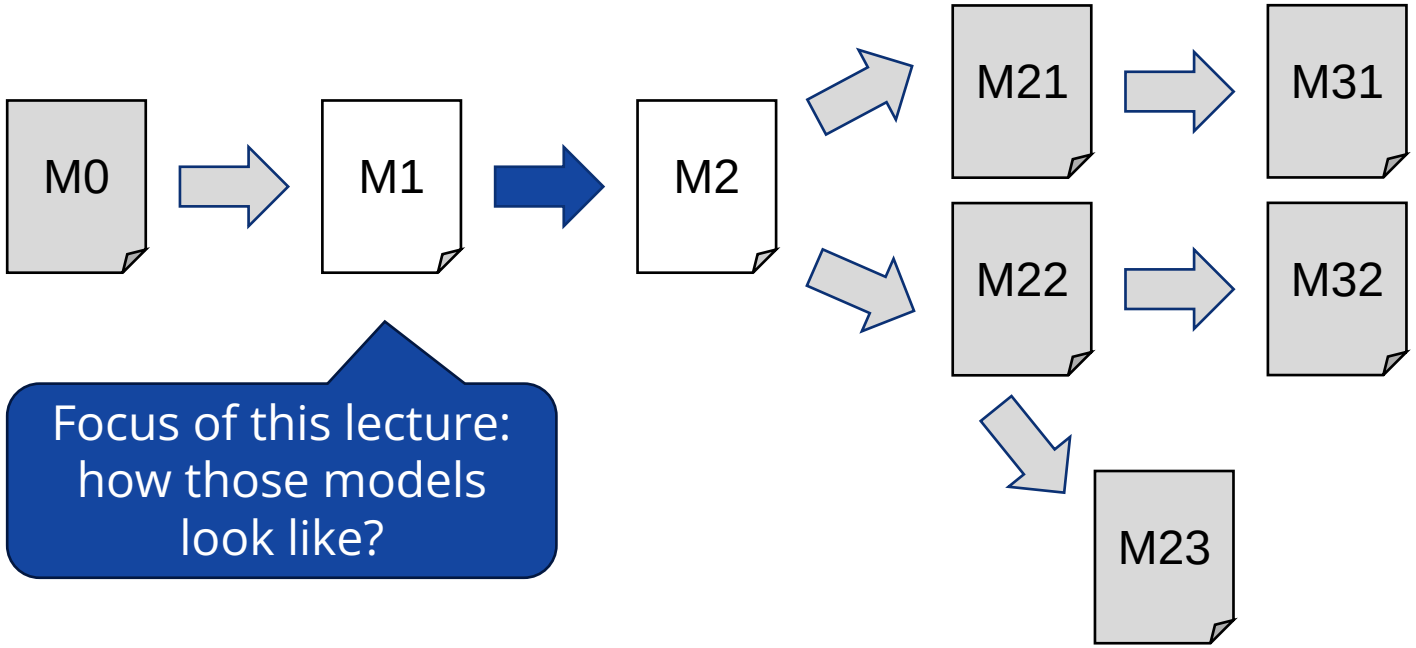
Graph-Based Modeling

Oszkár Semeráth, Kristóf Marussy

Critical Systems Research Group

Reminder: development processes

- We create models and automate their processing
- Software is constructed by a chain of transformations



Structure of models

Documents in development

- What kind of documents are used during development?

Documents in development

- What kind of documents are used during development?
- Textual documents

Source Code

- General purpose language
- Available Compilers / Interpreters

Main.java

```
public class Main {
 public static void
  main(String[] args) {
   System.out.println(
    "Hello World");
  }
}
```

- Typically, the main development artifacts.
- Low-code / No-code: replace coding with modeling
- Model-Driven Software Development: models are the main artifacts. Code / Documentation is generated.

Documents in development

- What kind of documents are used during development?
- Textual documents

Source Code

- General purpose language
- Available Compilers / Interpreters

Main.java

```java
public class Main {
 public static void
  main(String[] args) {
   System.out.println(
    "Hello World");
  }
}
```

Textual modeling languages

- General / Domain-Specific
- Custom editors
- Transformations

SysML v2 example

```
part def HubAssembly;
part def Tire;
part def Vehicle;

part def Chassis {
    part hubAssembly : HubAssembly[3..4];
    part tire : Tire[3..4];
}

connection def WheelConnection {
    end : HubAssembly[1];
    end : Tire[1];
}
```

- Custom editors and transformations can introduce
  - Advanced code generators
  - Mathematical algorithms
- SysML v2 is developed in ftsrg

Documents in development

- What kind of documents are used during development?
- Textual documents

Source Code

- General purpose language
- Available Compilers / Interpreters

Main.java

```java
public class Main {
 public static void
  main(String[] args) {
   System.out.println(
    "Hello World");
  }
}
```

Textual modeling languages

- General / Domain-Specific
- Custom editors
- Transformations

SysML v2 example

```
part def HubAssembly;
part def Tire;
part def Vehicle;

part def Chassis {
    part hubAssembly : HubAssembly[3..4];
    part tire : Tire[3..4];
}

connection def WheelConnection {
    end : HubAssembly[1];
    end : Tire[1];
}
```

Structured textual data formats

- Standard formats and tools to describe data
- E.g., xml, json, csv, …

note.xml

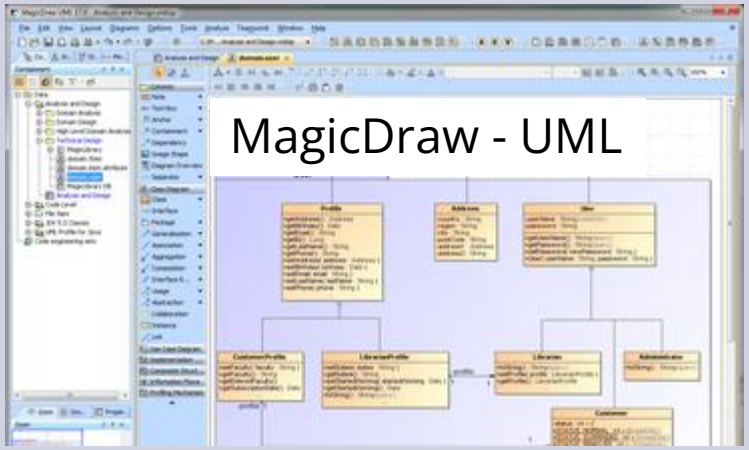
```xml
<?xml version="1.0" encoding=
"UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this
weekend!</body>
</note>
```

Documents in development

- What kind of documents are used during development?
- Graphical documents

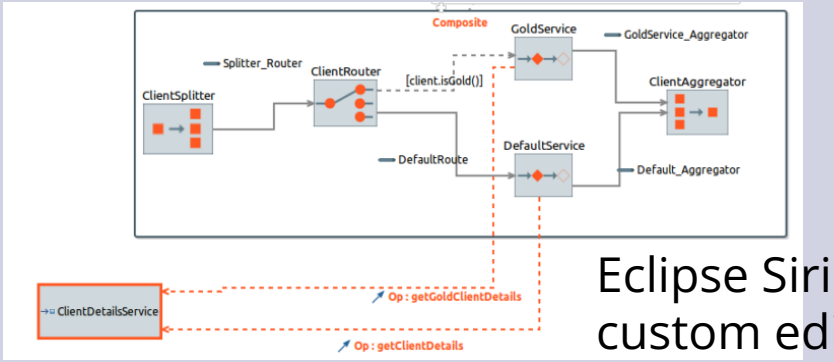
General Purpose Modeling tools

- Covers wide array of problems
- Non-specialized tools



Textual modeling languages

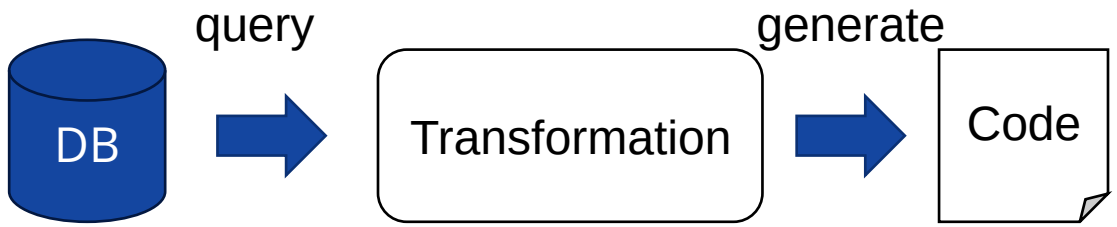
- General / Domain-Specific
- Custom editors
- Transformations

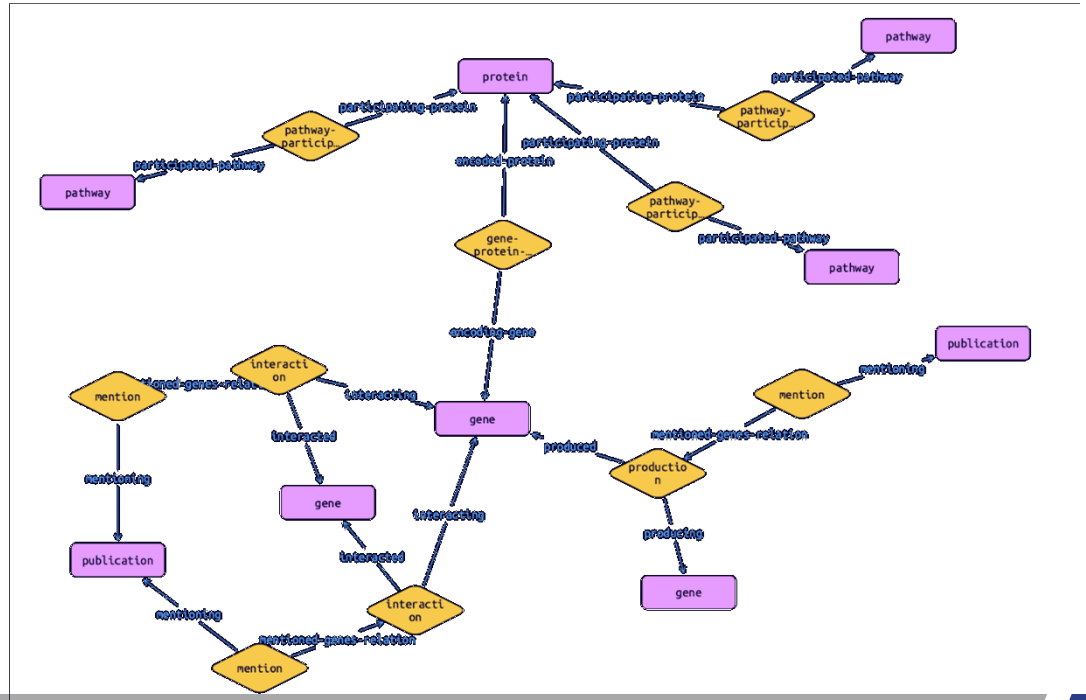


Eclipse Sirius – custom editors

Documents in development

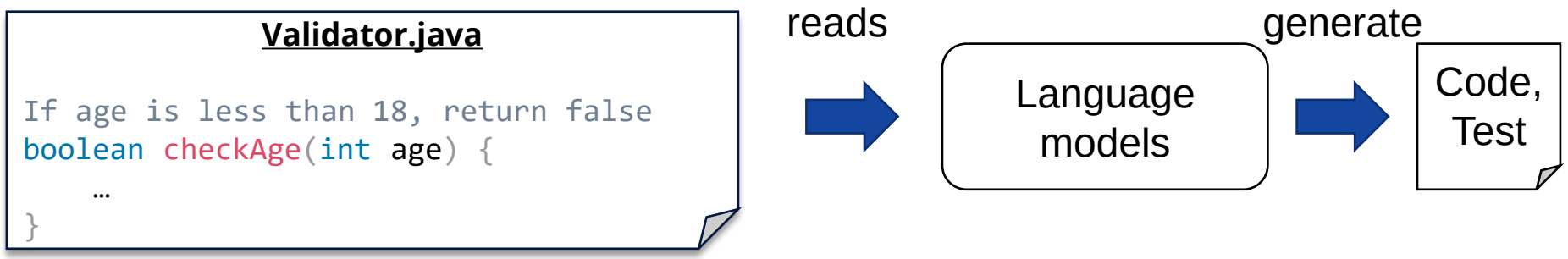
- What kind of documents are used during development?
- Model queries
  - Database (relational or no-sql)
  - Knowledge base or Ontologies
- example: TypeDB
- More and more frequent





Documents in development

- What kind of documents are used during development?
- Natural language, new opportunities



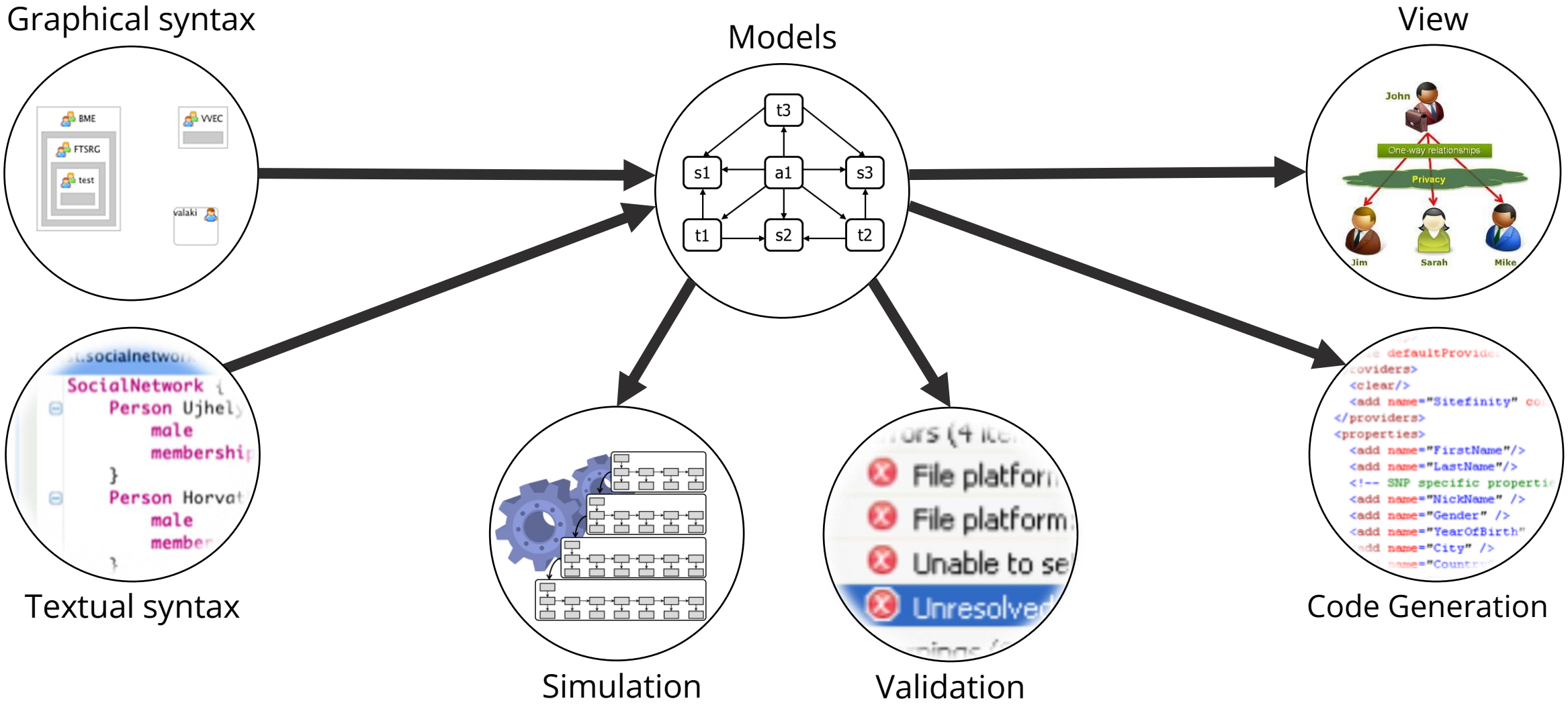
Documents in development

- What kind of documents are used during development?
- Textual documents
  - source code
  - textual modeling languages
  - structured textual documents: csv, xml, json
- Graphical documents
  - General Purpose modeling languages (e.g., UML)
  - or Domain-Specific
- Model queries
  - Database (relational or no-sql)
  - Knowledge base or Ontologies
- Natural language

How to **capture** all those information?

Logic Structures + **Graphs**

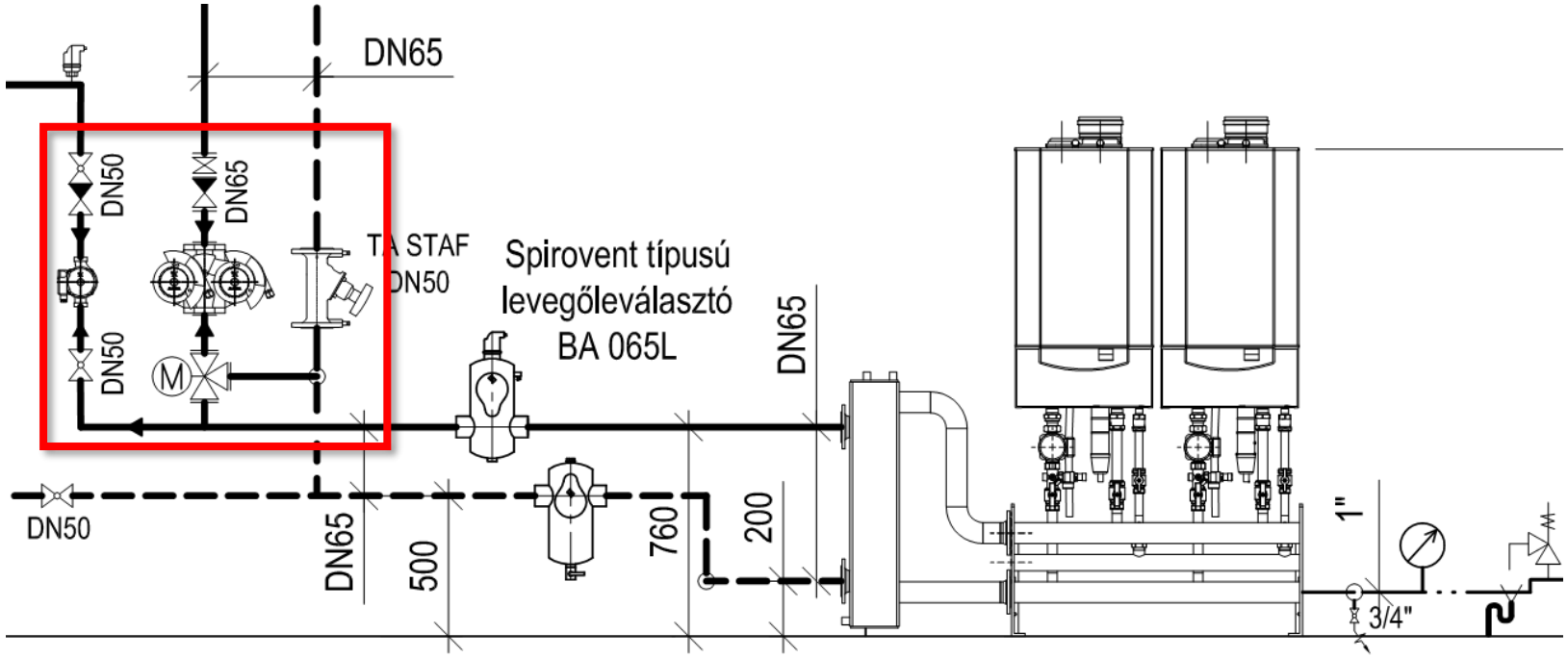
Common understanding of models



Concrete & Abstract syntax

Instance model, concrete syntax

- How models look like?



Honeywell keverőcsap DN50 K_{VS} 40

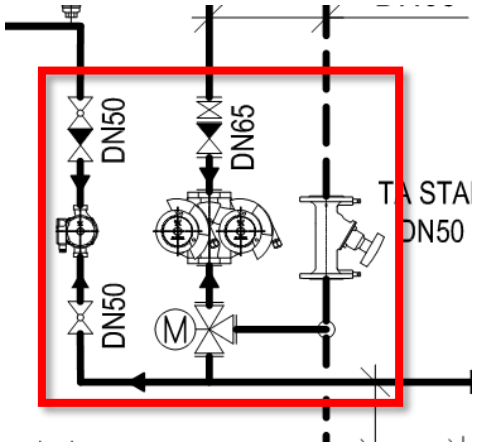
Spirovent típusú iszapleválasztó BE 065L

Remeha Quinta kaszkád rendszer hidrauliku

Concrete syntax

Instance model, abstract syntax

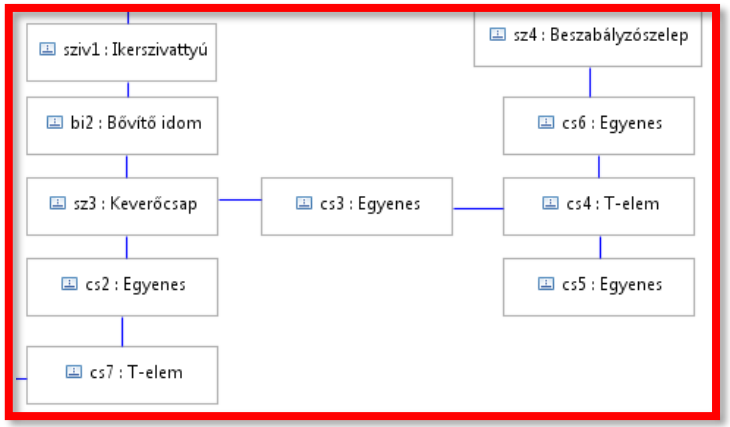
- How to represent models in a computer?

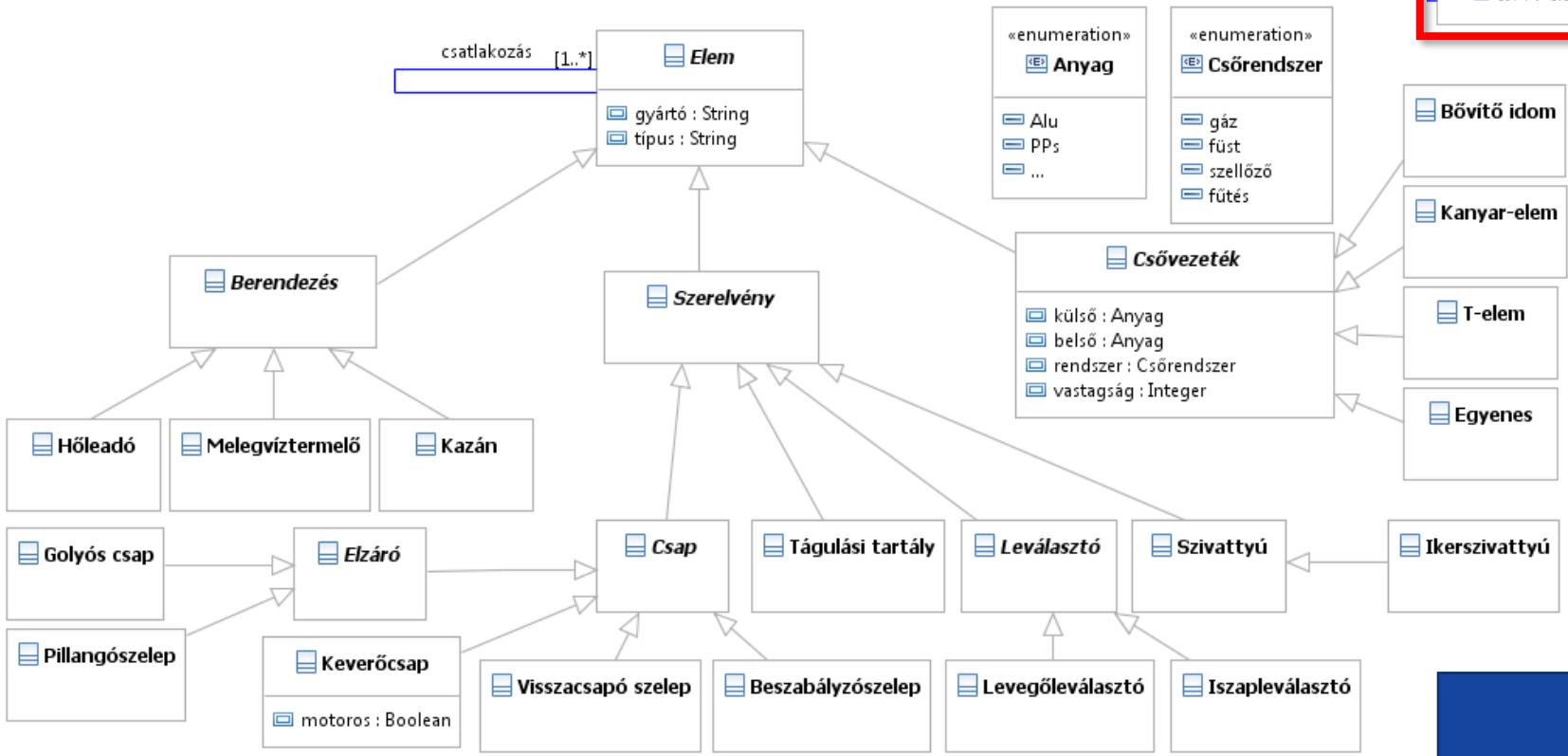


Abstract syntax

Example metamodel

- How to describe the data structure?





Metamodel

Definitions

- Separate models to **representation-independent** and **representation-specific** parts.

Definition

Abstract syntax is the (abstract) data structure of a model, which excludes representation-specific details.

Definition

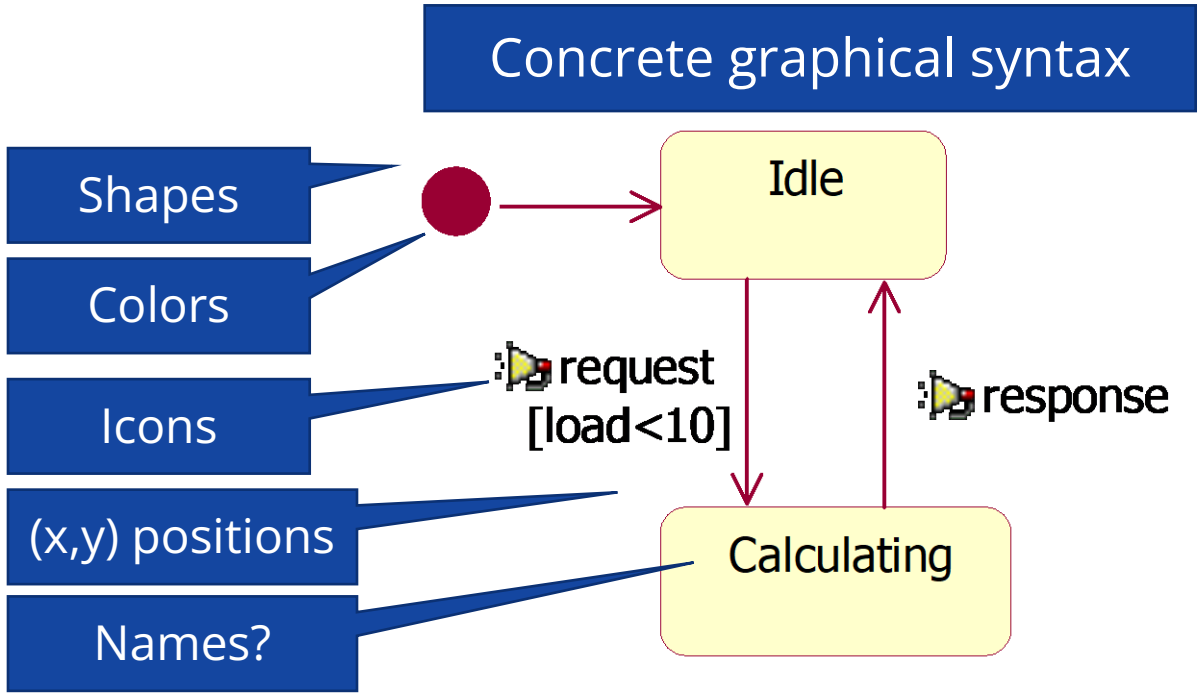
Concrete syntax is the complete representation of a model (which includes representation-specific details).

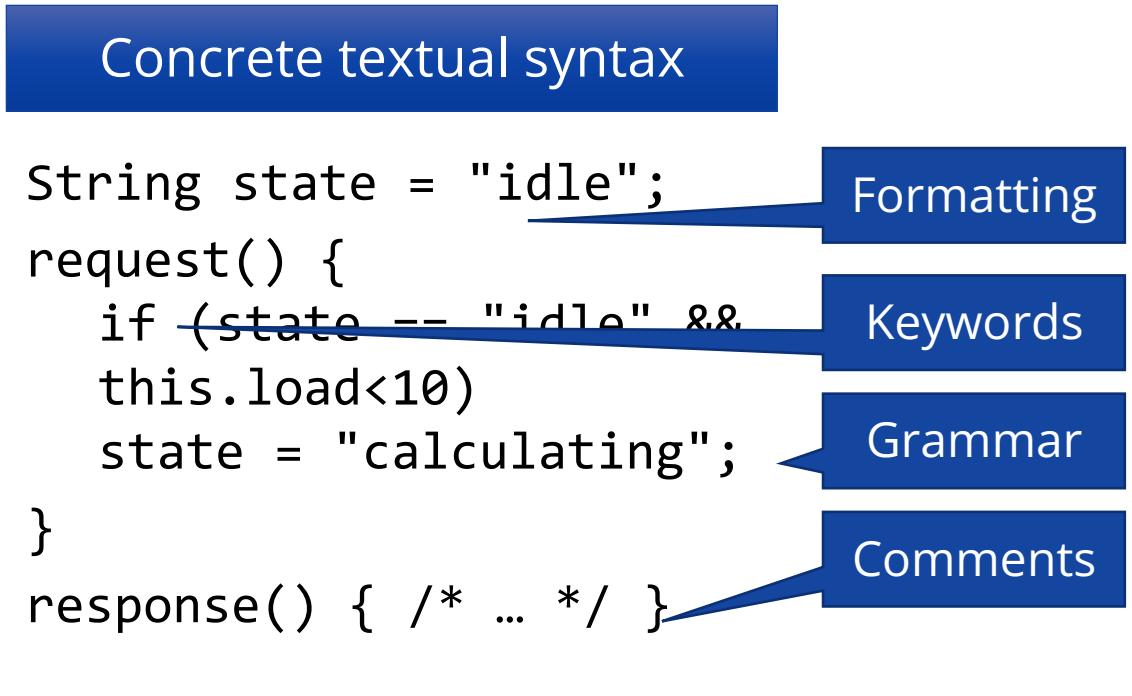
- Abstract and concrete syntax may also denote the representation technique of a modeling language.

“concrete syntax of UML” →
The way that UML represents models in its editor.

Concrete syntax example

- What is part of the concrete syntax, but not part of the abstract?





Textual vs. Visual

Textual notation

+ **Easy to write:**
  Able to capture complex expressions

- **Difficult to read:**
  Difficult to comprehend and manage after certain complexity (e.g what refers me?)

Visual notation

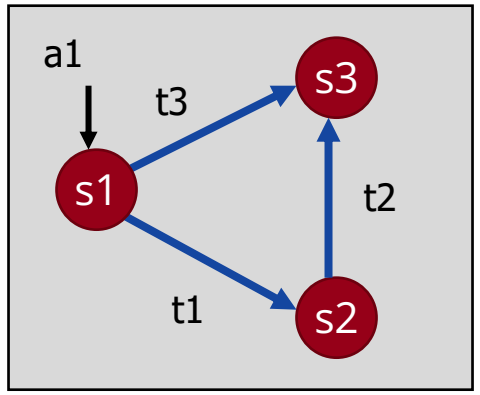
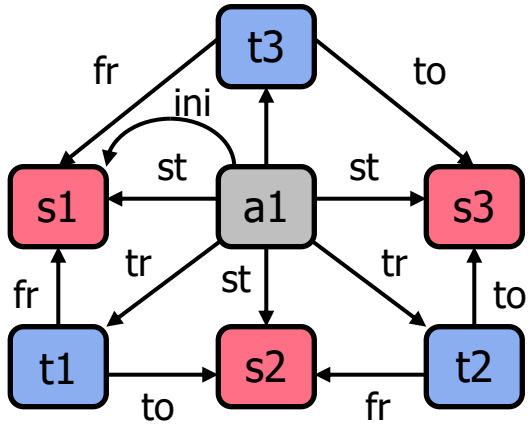
+ **Easy to read:**
  Able to express (selected / subset of) details in an intuitive, understandable form

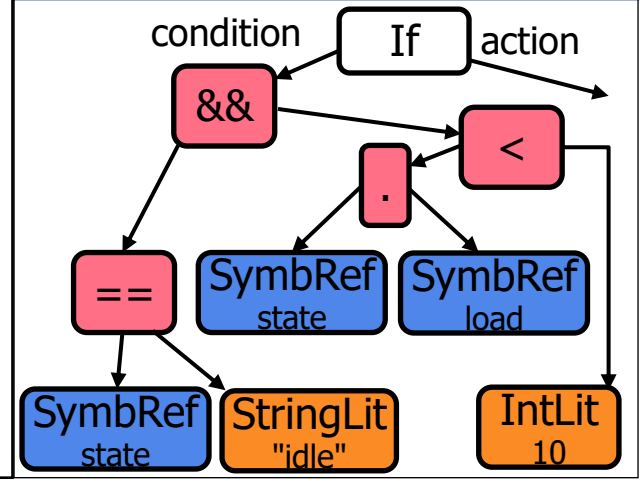
+ **Safe to write:** Able to construct syntactically correct models

- **Difficult to write:** graphical editing is slower

Abstract syntax example

- How to capture the abstract syntax of a model?
- What to include to abstract syntax?
  (semantics, version control, abstract syntax to concrete syntax traf)

Concrete syntax	Abstract syntax
	

Concrete syntax	Abstract syntax
`if ( state == "idle" && this.load<10) …`	

- Abstract syntax: typically a graph-based structure.

Multiplicity of Notations

- 1 abstract syntax → many textual and visual notations
  - Human-readable-writable textual or visual syntax
  - Textual syntax for exchange or storage (typically XML)
  - In case of UML, each diagram is only a partial view
- 1 abstract model → many concrete forms in 1 syntax!
  - Whitespace, diagram layout
  - Comments
  - Syntactic sugar
- 1 semantic interpretation → many abstract models
  - e.g. UML2 Attribute vs. one-way Association

Metamodels and Instance Models

Metamodels

- How to describe the range of valid models?

Definition

A **metamodel** defines the most important concepts and relations of a modeling language and defines the basic structure of the models.

Goal: to define…

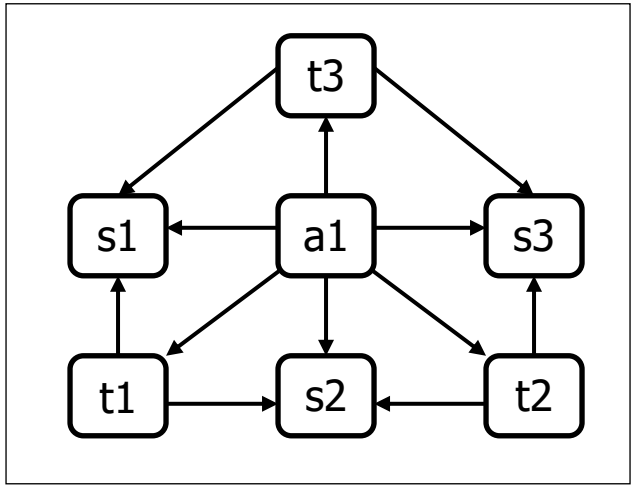
- Basic concepts
- Relations between concepts
- Attributes of concepts
- Abstraction / refinement (Taxonomy, Ontology) between model elements
- Aggregation
- Multiplicity restrictions
- …

How to model with graphs?

- Directed graphs: $G = \langle V_G, E_G \rangle$, where:
  - V_G: finite set of **nodes** (of G)
  - $E_G: V_G \times V_G \rightarrow \{true, false\}$: **edges** between the nodes (in G)
  - Loop edges supported, parallel edges to the same direction not

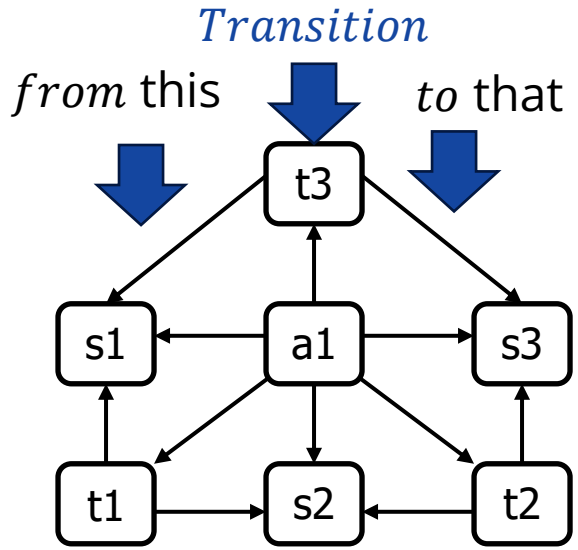
- Example: statechart model S
  - $V_S = \{a_1, s_1, s_2, s_3, t_1, t_2, t_3\}$
  - $E_S(a_1, s_1) = true$, but $E_S(s_1, a_1) = false$

- Question: How to add labels?



Problem: too simple

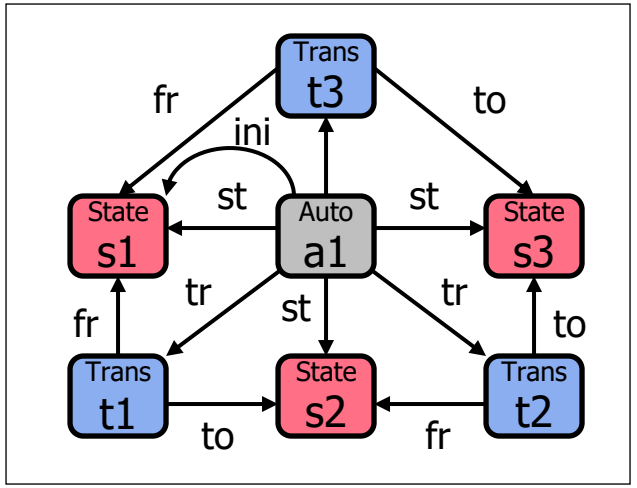
- We want to add some more information
- Add labels to the nodes and the edges!
- How to define the set of labels?



How to model with labelled graphs?

Dictionary: $\langle \Sigma, \alpha \rangle$

- Σ: Set of labels
  - For example, if we want to write the words "$Automaton$", "$State$", "$Transition$", … on the graph, then:
    $\Sigma = \{Automaton, State, Transition, states, transitions, from, to, init\}$
- Node and edge labels are separate by the Arity
- Arity $\alpha: \Sigma \to \mathbb{N}$ specifies the role of the labels
  - Node labels $\alpha: Automaton, State, Transition \mapsto 1$
  - Edge labels: $\alpha: states, transitions, from, to, init \mapsto 2$
  - Other arity is rare.



How to model with labelled graphs?

Labelled graphs: $G = \langle O_G, I_G \rangle$, where:

- O_G: finite set of **nodes** or **objects** (of G)
- $I_G(s): V_G^{\alpha(s)} \to \{true, false\}$: **interpretation function** for each $s \in \Sigma$

  *I_G tells whether a node **label** or **type** is true for a node or there is an **edge** or **link** between two nodes with a label.*

Example: statechart model S

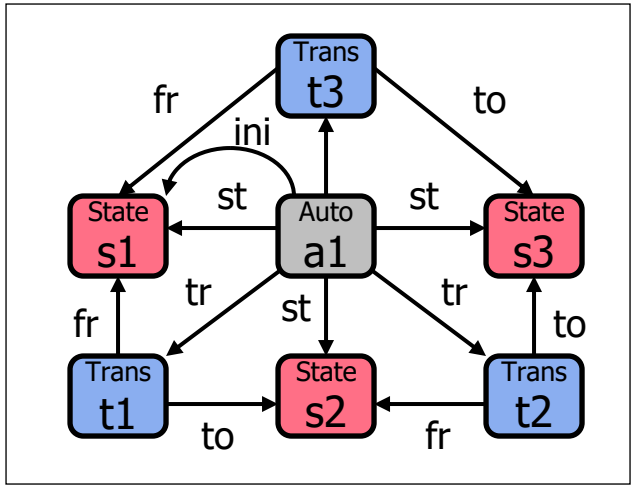
$$O_S = \{a_1, s_1, s_2, s_3, t_1, t_2, t_3\}$$

$I_S(State)(s_1) = true$, or simply $State(s_1) = true$, $Transition(t_1) = true$

$State(t_1) = false$, $Transition(s_1) = false$

Similarly: $from(t_1, s_1) = true, to(t_1, s_2) = true$, but $from(s_1, s_1) = false$

This will be very simple after LAB!

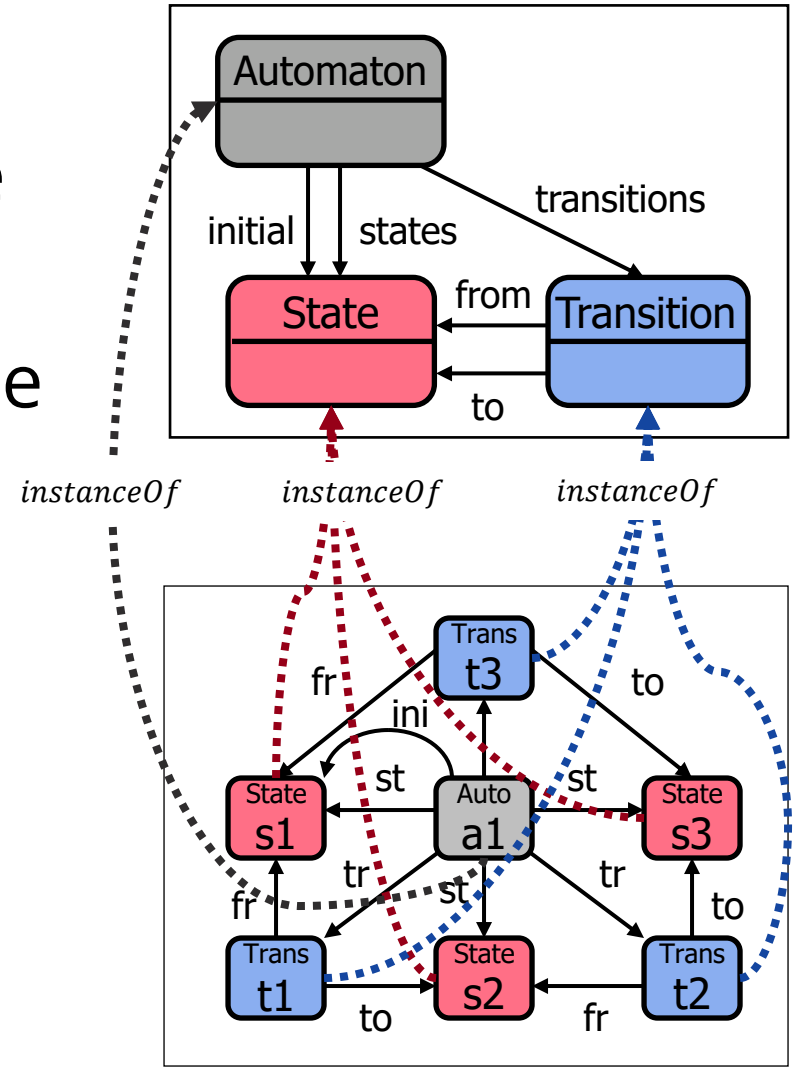


Basic Structure

- Labels used as types

Basic Structure

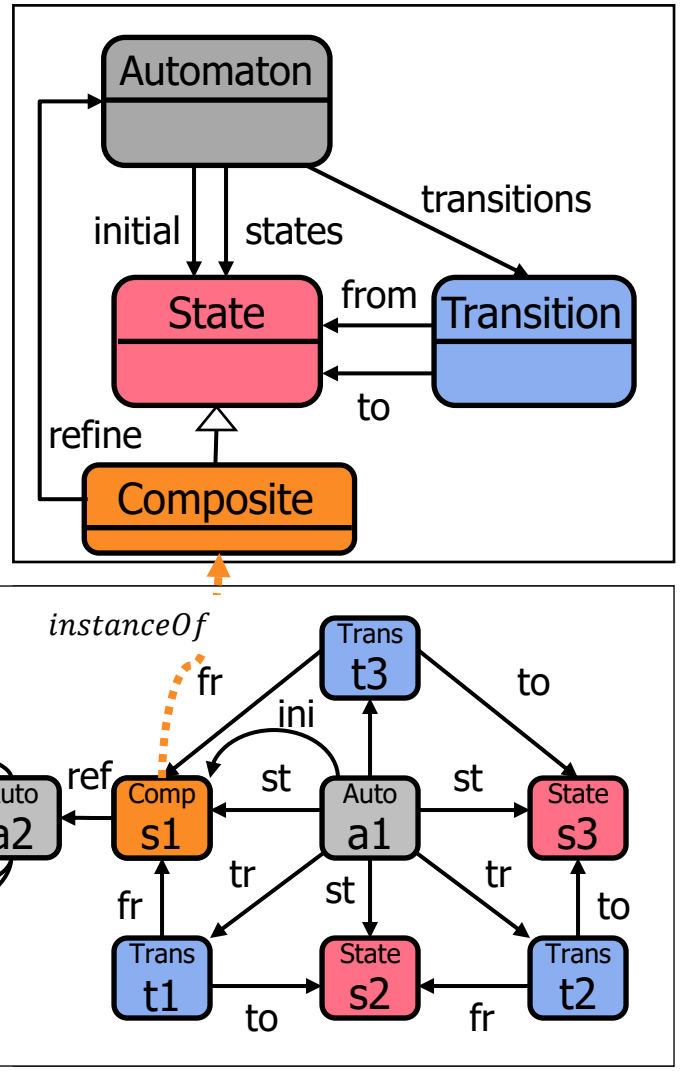
- We want to assign labels more clearly
- We can use a “class diagram” to define the valid labeling.
- Edges are allowed if they are allowed in the metamodel



Supertypes and subtypes

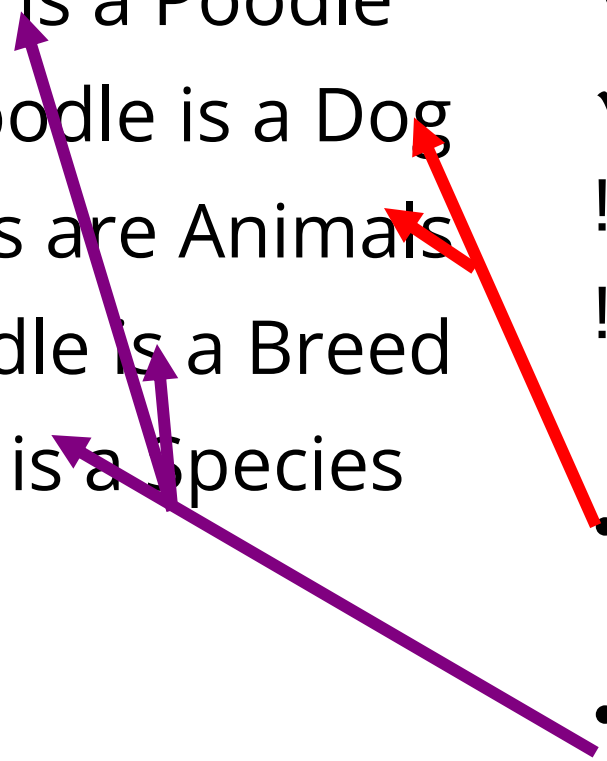
- Create more complex types: $supertype$ relation (inheritance)

- Type system can be extended with subtypes. A graph G is type conformant, if:
  - every node in G has a type in T defined by $instanceOf$,
  - for each every edge with label S between s and t if there is an edge with S between a **supertype of** $instanceOf(s)$ and a **supertype of** $instanceOf(t)$ in T.
- **Direct type:** $instanceOf$
  Indirect type: $instanceOf + supertype*$
- **Abstract type:** cannot be direct type



Exercise: Classification vs. Generalization

1. Fido is a Poodle
2. A Poodle is a Dog
3. Dogs are Animals
4. Poodle is a Breed
5. Dog is a Species



✓1+2 = Fido is a Dog

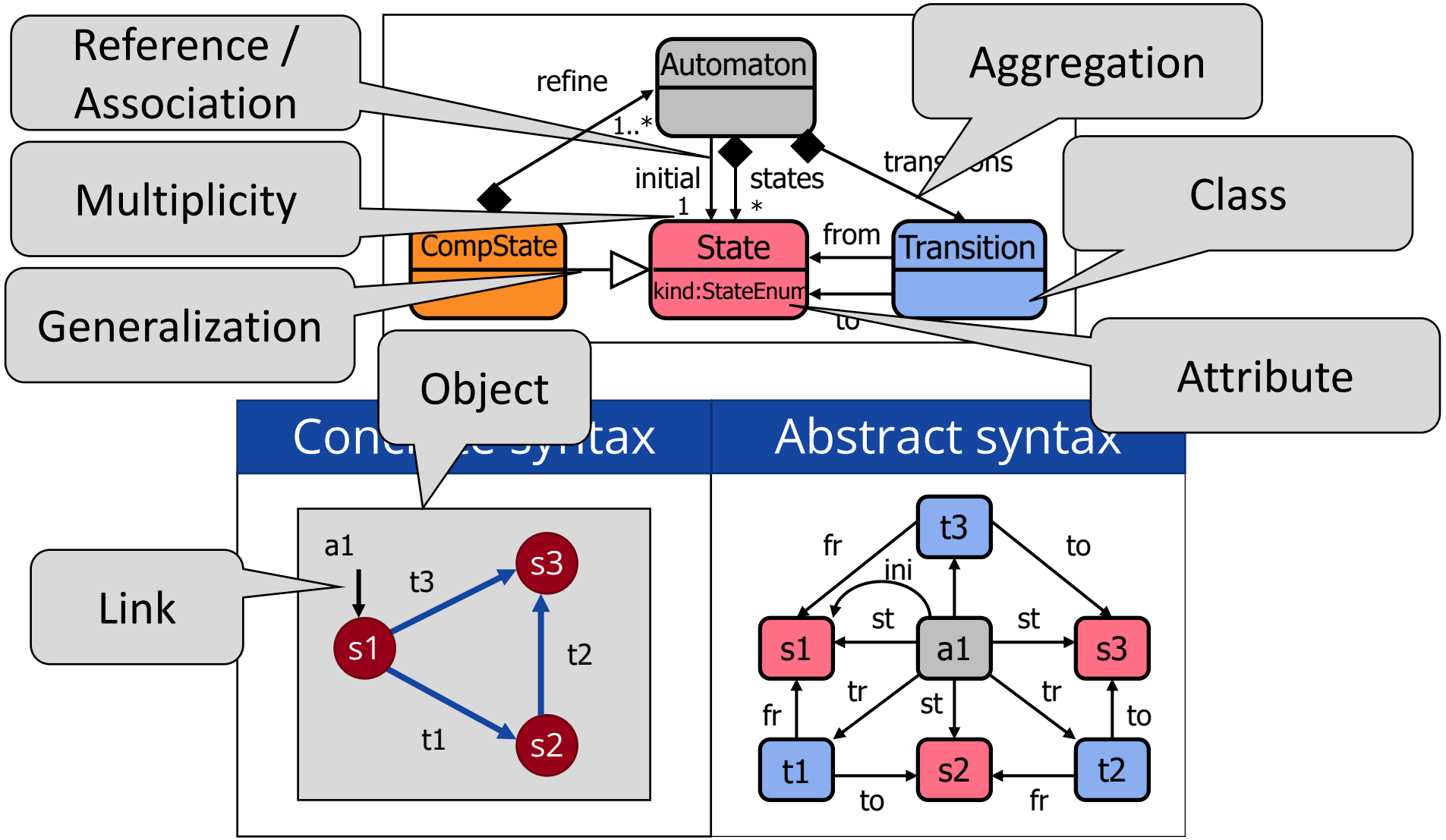
✓1+2+3 = Fido is an Animal

! 1+4 = Fido is a Breed

! 2+5 = A Poodle is a Species

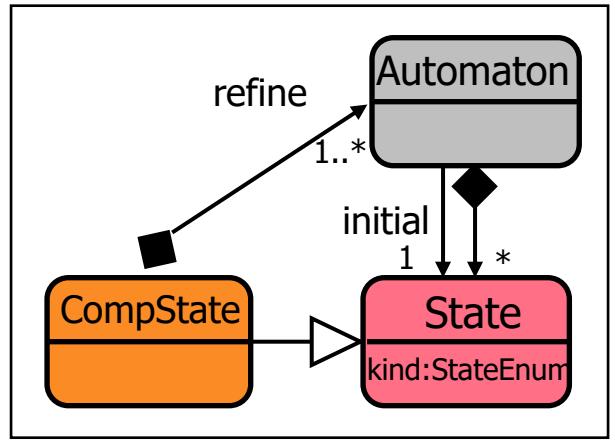
- Generalization (SupertypeOf) is transitive
- Classification (InstanceOf) is NOT transitive

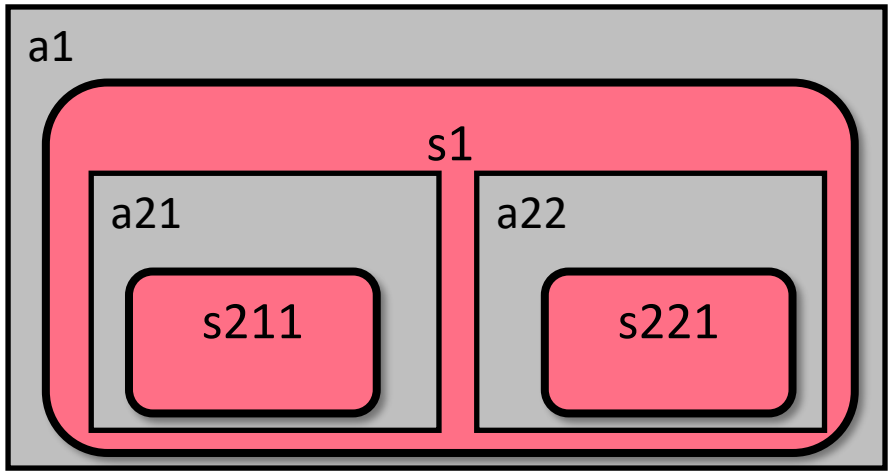
Additional structural elements

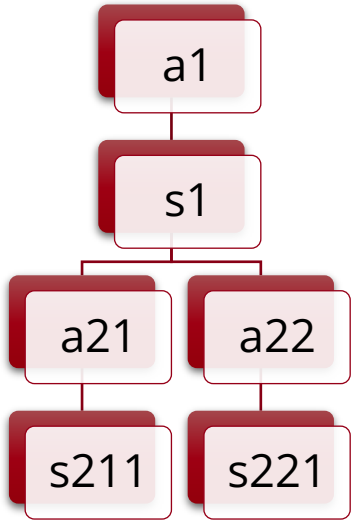


Containment

- Each model element has a unique parent
  - N children → 1 parent
  - Single root element
- Aggregation as relationship:
  - Defined in the metamodel along reference edges
  - Provides restriction for instance models
- Circularity
  - No circular containment (in the model)
  - Aggregation relations in the metamodel may be circular (hierarchy)

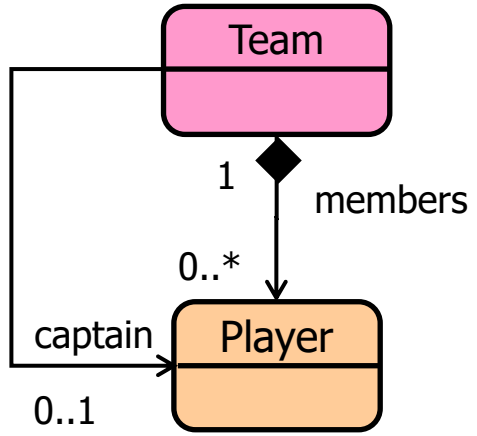






Multiplicity

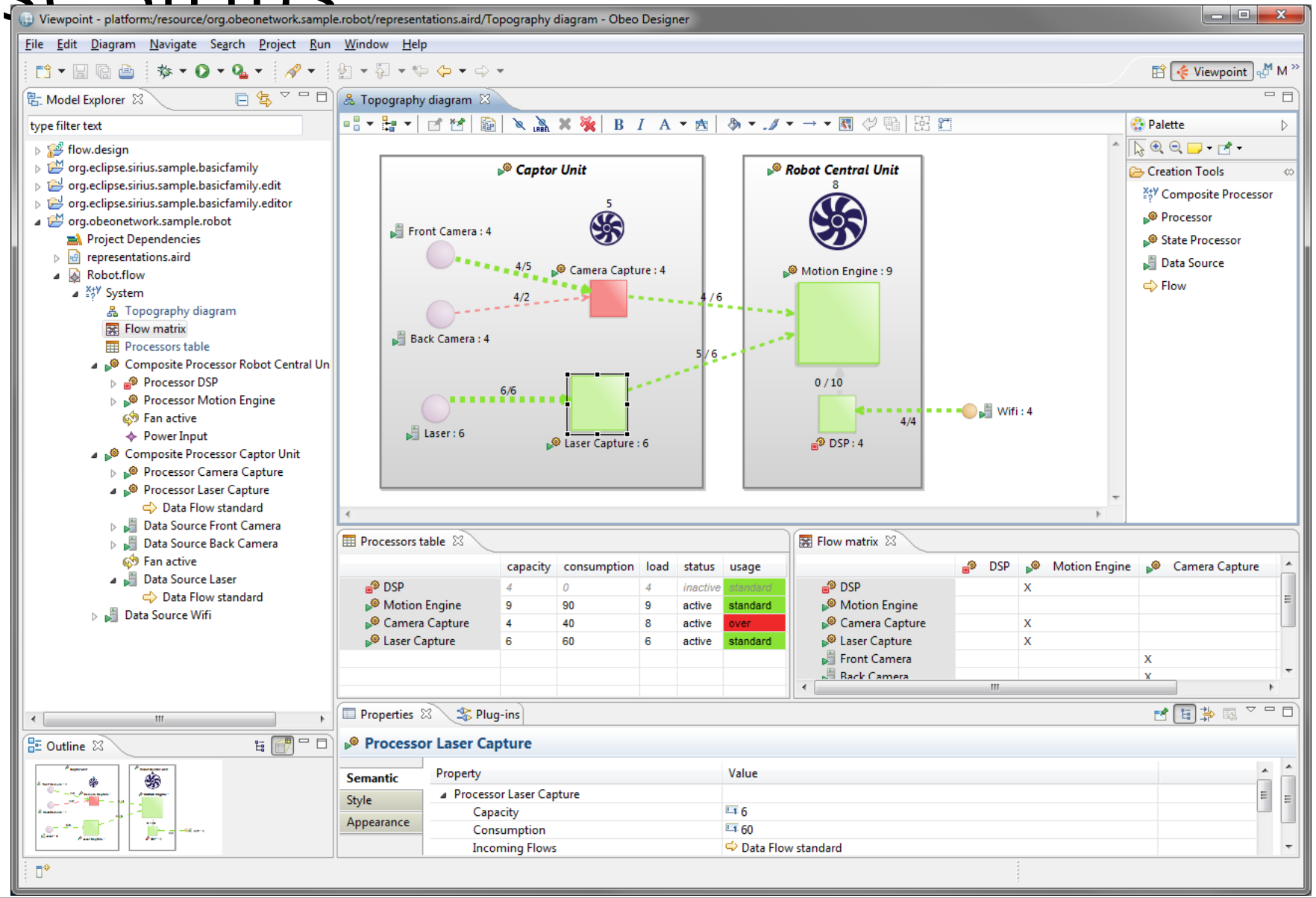
- Definition: Lower bound .. Upper bound
  - Lower bound: 0, 1, (non-negative integer)
  - Upper bound: 1, 2, … * (positive integer + any)
- Scope:
  - References: allowed number of links between objects of specific types
  - Attributes: e.g. arrays of strings (built-in values



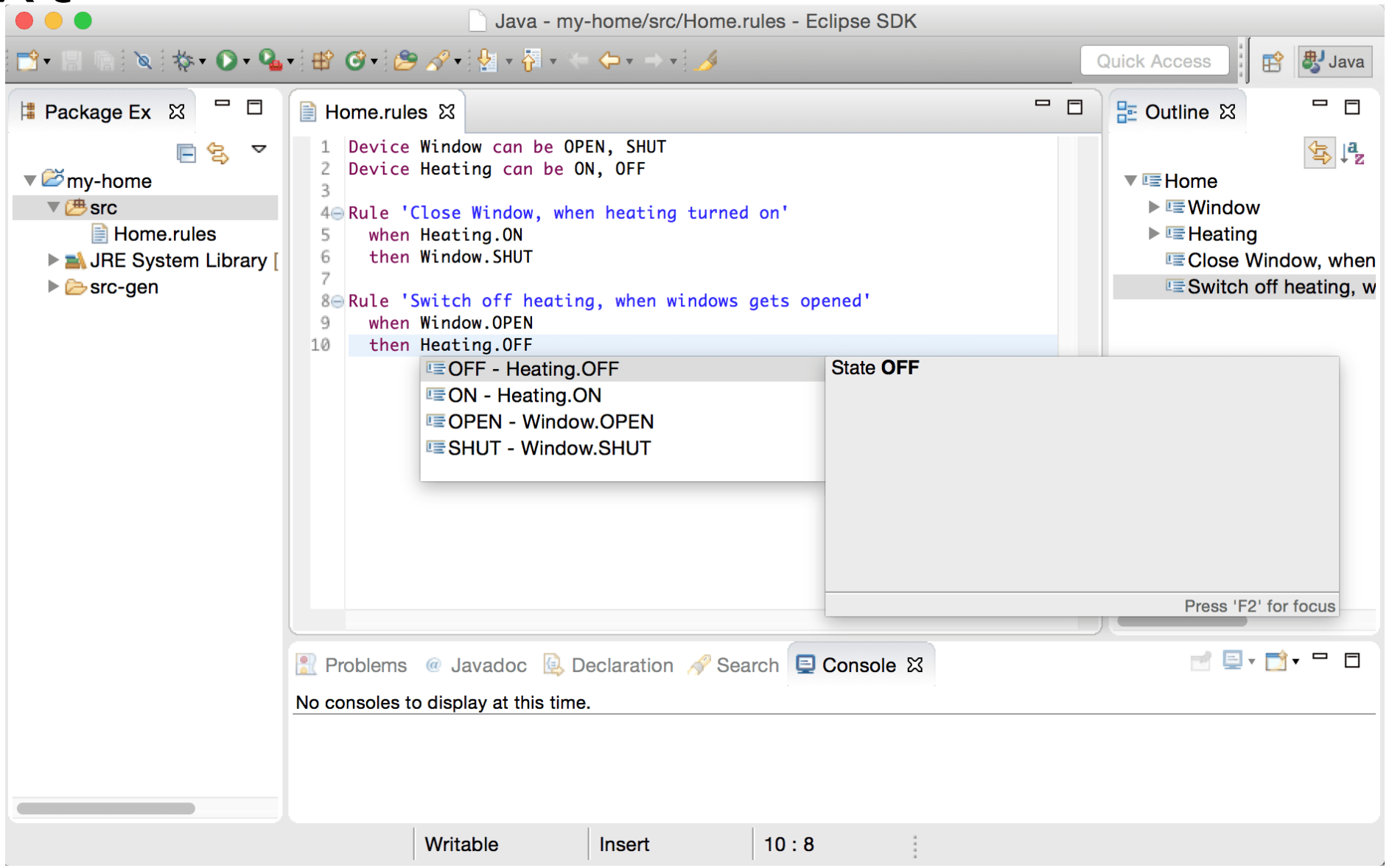
Which are the most common multiplicity definitions in practice?

Modeling tools

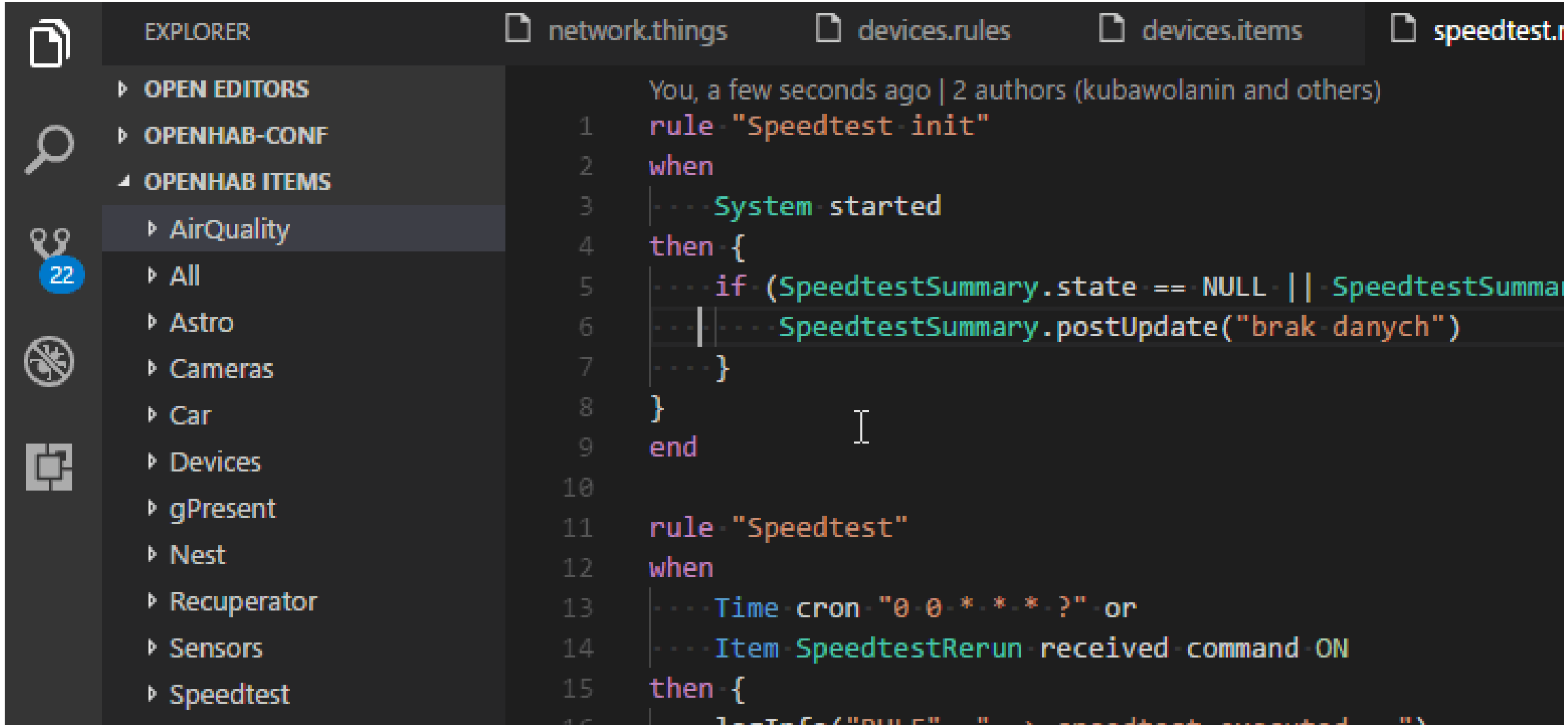
Eclipse Sirius



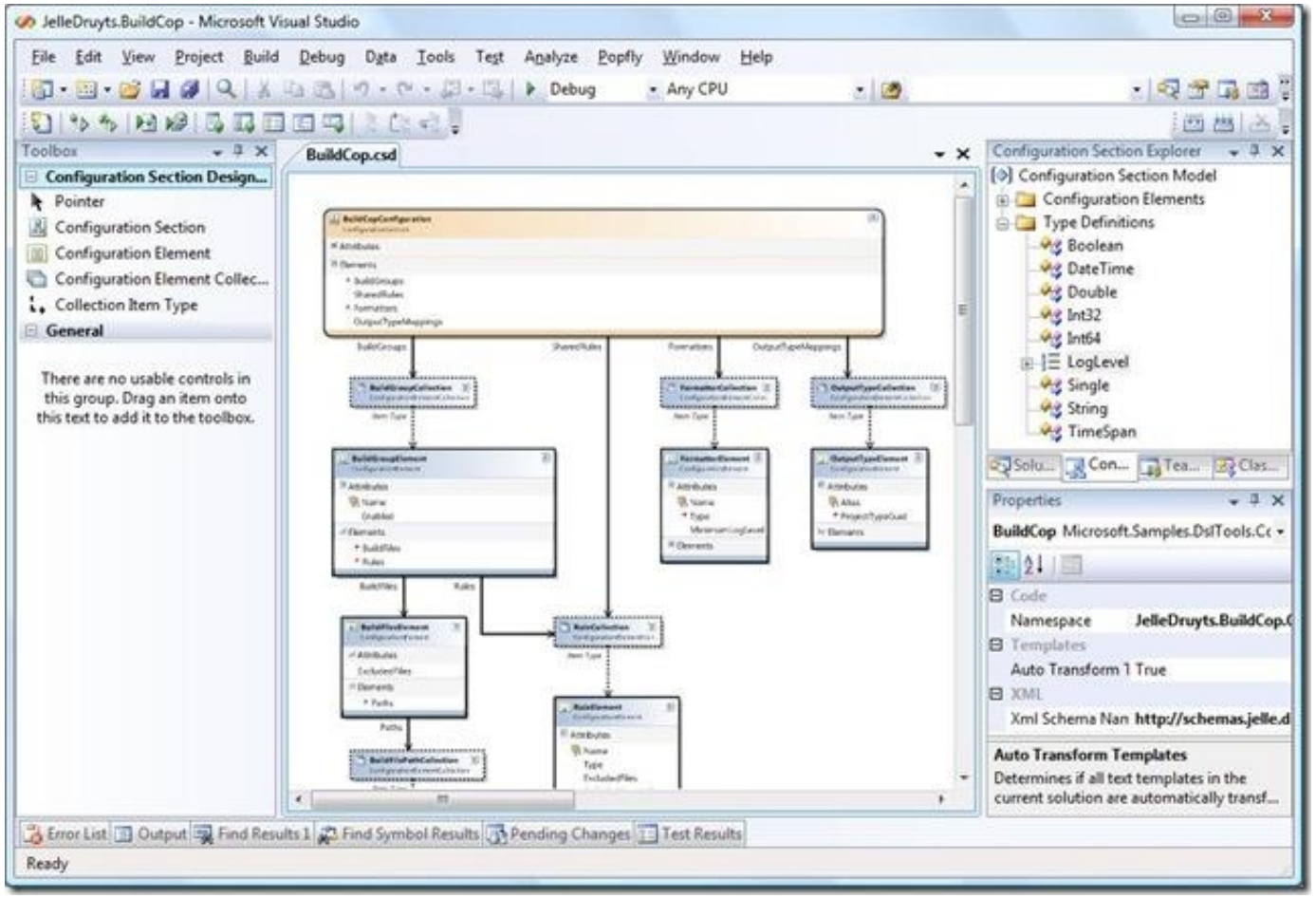
Xtext


Java - my-home/src/Home.rules - Eclipse SDK
Quick Access
Java
Package Ex
my-home
src
Home.rules
JRE System Library [
src-gen
Home.rules
1 Device Window can be OPEN, SHUT
2 Device Heating can be ON, OFF
3
4 Rule 'Close Window, when heating turned on'
5 when Heating.ON
6 then Window.SHUT
7
8 Rule 'Switch off heating, when windows gets opened'
9 when Window.OPEN
10 then Heating.OFF
OFF - Heating.OFF
ON - Heating.ON
OPEN - Window.OPEN
SHUT - Window.SHUT
State OFF
Press 'F2' for focus
Outline
Home
Window
Heating
Close Window, when
Switch off heating, w
Problems
Javadoc
Declaration
Search
Console
No consoles to display at this time.
Writable
Insert
10 : 8

Xtext - Theia


EXPLORER
network.things
devices.rules
devices.items
OPEN EDITORS
OPENHAB-CONF
OPENHAB ITEMS
AirQuality
All
Astro
Cameras
Car
Devices
gPresent
Nest
Recuperator
Sensors
Speedtest
22
You, a few seconds ago | 2 authors (kubawolanin and others)
rule "Speedtest init"
when
System started
then {
if (SpeedtestSummary.state == NULL ||
SpeedtestSummary.postUpdate("brak danych")
}
}
end
rule "Speedtest"
when
Time cron "0 0 * * * ?" or
Item SpeedtestRerun received command ON
then {

Microsoft DSL Tools



MPS / mbeddr

```
HelperFunctions
model   test.ex.core.externalOFile
constr
import

void foo() {

} foo (function)
```

```
module ADemoModule imports nothing {

  enum MODE { FAIL; AUTO; MANUAL; }

  statemachine Counter (initial = start) {
    in start() <no binding>
       step(int[0..10] size) <no binding> trace R2
    out started() <no binding>
        resetted() <no binding> {resettable}
        incremented(int[0..10] newVal) <no binding>
    vars int[0..10] currentVal = 0
         int[0..10] LIMIT = 10
    state start {
      on start [ ] -> countState { send started(); }
    }
      start ^inEvents (cdesignpaper.screenshot.ADemoModule)
      step  ^inEvents (cdesignpaper.screenshot.ADemoModule)
    state
      on step [currentVal + size > LIMIT] -> start { send resetted(); }
      on step [currentVal + size <= LIMIT] -> countState {
Error: wrong number of arguments
                                    + size;
        send incremented();
      }
      on start [ ] -> start { send resetted(); } {resettable}
  }}

  System.out.println(String.valueOf((∑ [[1 k 0] [0 1.0 0] [0 0 1]])));

  System.out.println(exp(a + i * b) - exp(a) * (cos(b) + i * sin(b)));

  matrix<Double> s = [[3.0] [sin(1)] [1] [1]
                      [3^2] [1]      [3 + 1.0/2] [2]
                      [0]   [7 - 1.0/2 + 1] [exp(1)] [3]
                      [4]   [2]      [0]   [0]];
```

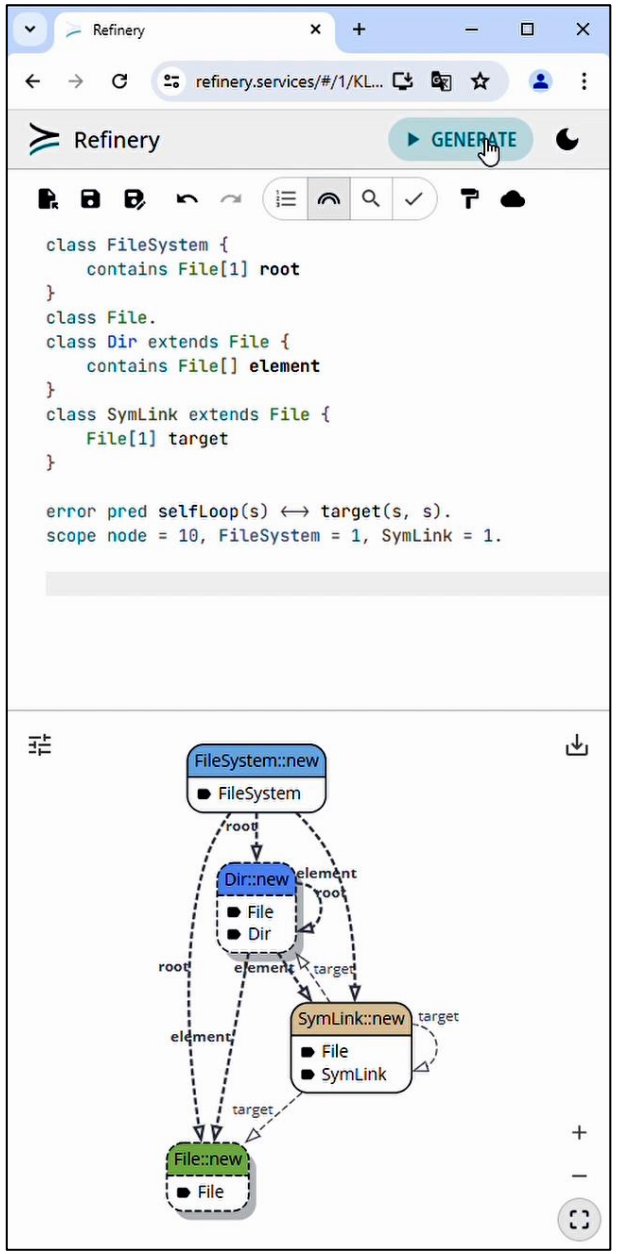
https://www.youtube.com/watch?v=8BjganpKT6U

Modern tools from ftsrg

- We are experts in **tool development**
- Interactive state-of-the-art **web-based editors**
- Generation algorithms **deployed on server**
- Various domains, e.g.,

  blockchain architecture | chemical reaction railway topology | modeling environment, satellite network | video game maps

- **Goal:** support **engineers** and **experts** solving complex problems



Recent Results

- Refinery
  Continuously deployed:
  https://refinery.tools/
- Amazon Research Award
  First in the region

amazon | science

How to encode those documents to graphs?

- Textual documents
  - source code
  - textual modeling languages

  Next lecture
  - structured textual documents: csv, xml, json

  Inherent graph structure, tools
- Graphical documents
  - General Purpose modeling languages (e.g., UML)
  - or Domain-Specific

  Typically uses xml
- Model queries
  - Database (relational or no-sql)
  - Knowledge base or Ontologies

  We are using a relational logic structure
- Natural language

  Later lecture