THE USE OF THE SEMANTIC WEB TECHNOLOGIES FOR PROVIDING PORTFOLIO-LEVEL END OF LIFE ANALYSIS FOR TRANSPORT INFRASTRUCTURE

END OF LIFE RULE-CHECKING MODEL: THE CASE OF NAVIGATION LOCKS

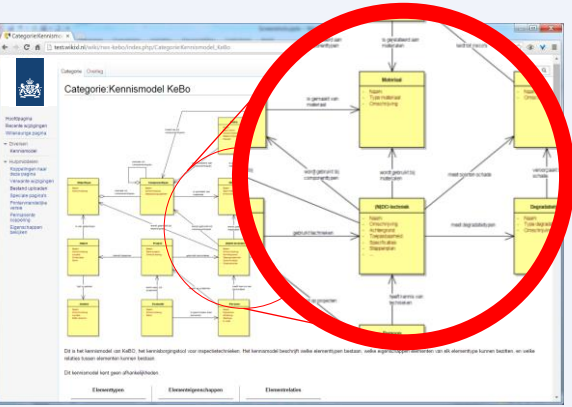
Esra Bektaş, PhD

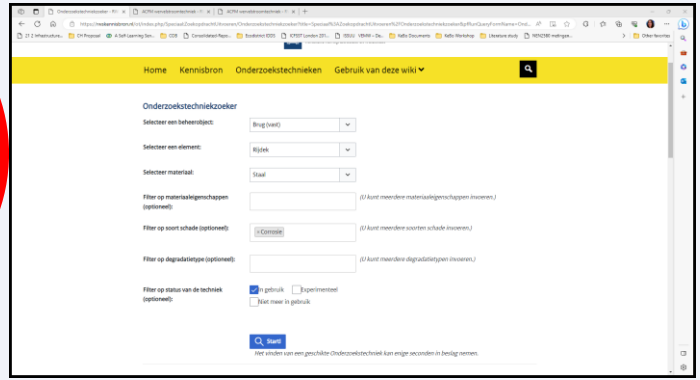
Topics to cover

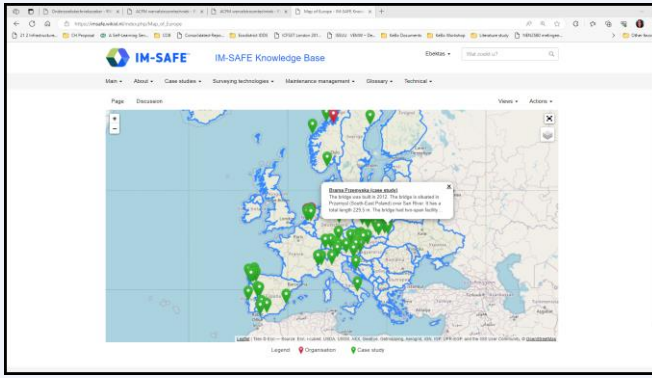
Background work	• Knowledge-bases for the Dutch Road Authority (Rijkswaterstaat) • Semantic Media Wiki Implementations	2’
Research Project as a Use Case	• Sharing The Idea & Exercising the Semantic Web Technologies • via End of Life Rule Checking Model (on-going project - PoC)	10’
Next Steps	• + Main take aways, Upscale PoC To Locks & Later New Object Types	3’

Background Work

- Knowledge-Bases At the Dutch Road Authority
  - A Semantic Wiki Implementation, always an object type and associated knowledge needs
  - Pre-determined queries, Coordinated text, images, inspection data
- End-user oriented solution together with ArchiXL





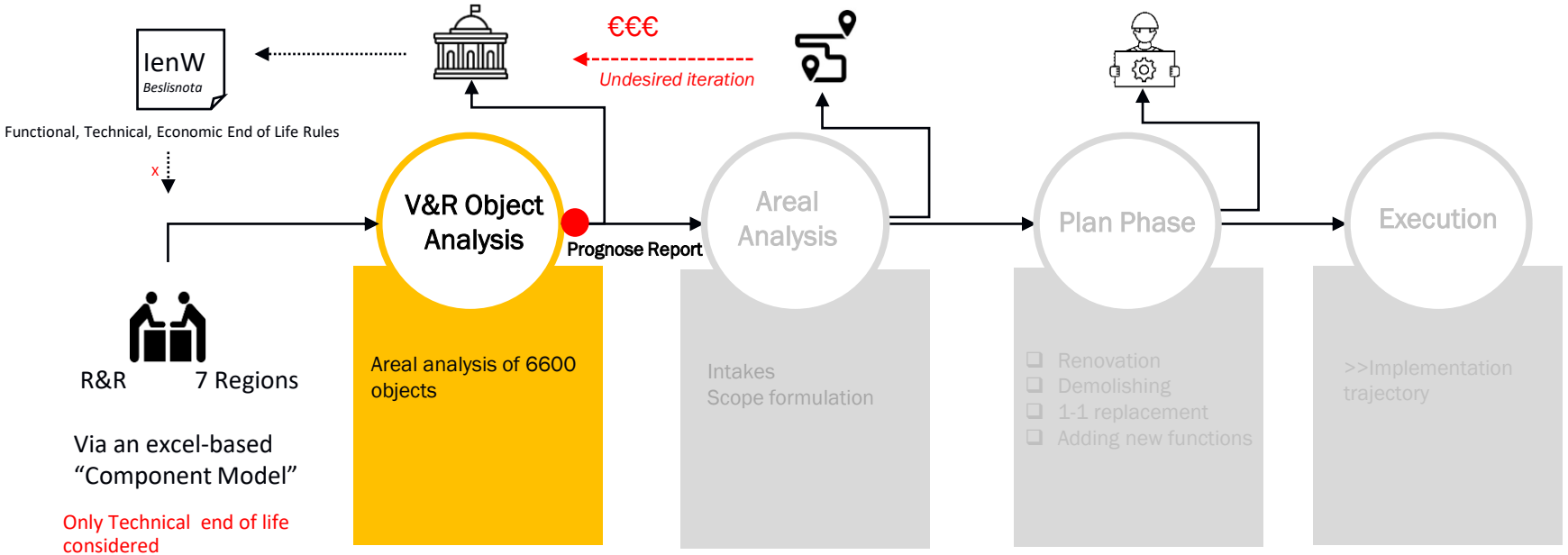


End of life rule-checking model for asset prioritization: Navigation Locks

As Knowledge Development Project

Context: End of Life Decision Moment

Rapid analysis of the asset portfolio

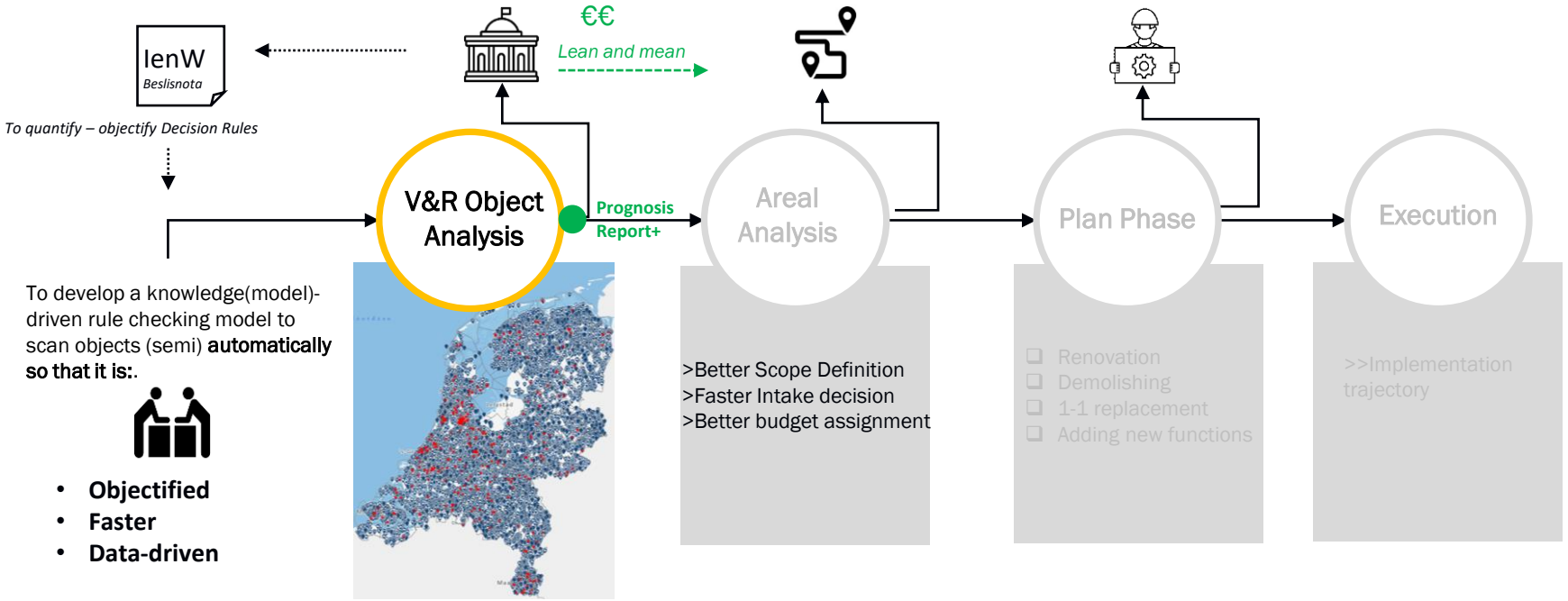


Via an excel-based "Component Model"

Only Technical end of life considered

Rule of thumb: concrete bridges < 1976, replace 10%

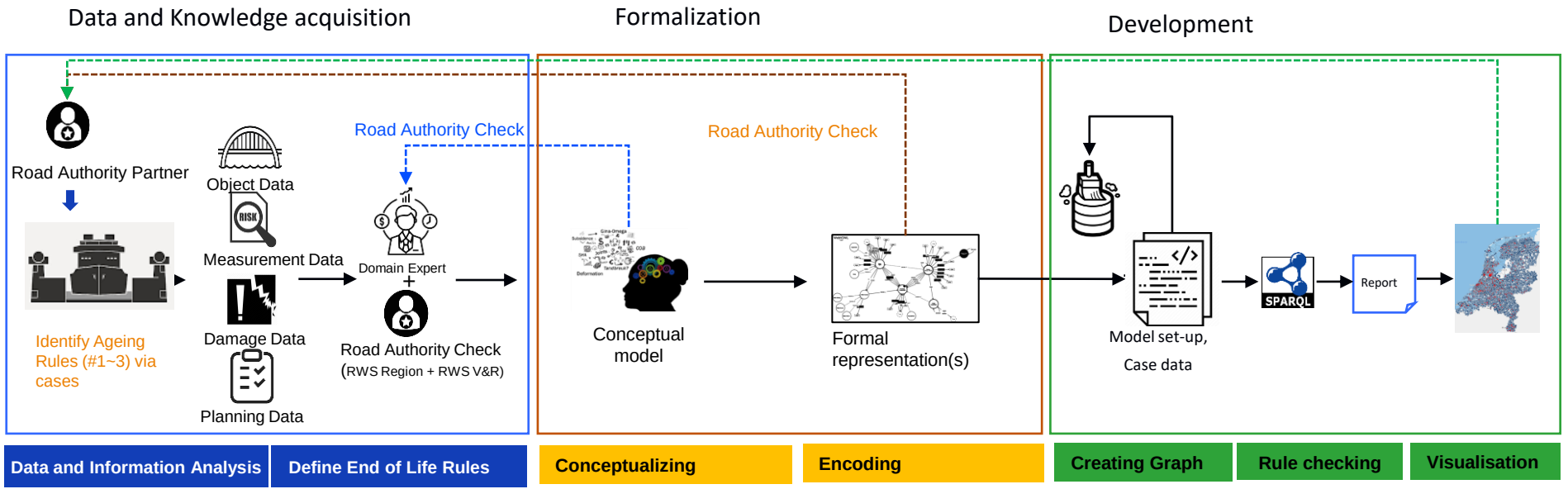
Approaching with the use of Semantic Web Technologies



To develop a knowledge(model)-driven rule checking model to scan objects (semi) **automatically so that it is:**.

- **Objectified**
- **Faster**
- **Data-driven**

Methodology, envisioned



End of Life Rules

There are formal definitions Based on I&M (2013), VONK (2012) but also Klanker et. Al (2016), Bakker et al (2016), Wessels et al. (2018)
They all interact on the level of object properties.

Technical end of life

- Structural safety can no longer be guaranteed.
- Or when a structure has serious structural defects.
- Or when a (critical) component is obsolete.

Functional end of life

- When an asset no longer fulfills its designed function due to changing environment, requirements, demand.
- When there are new functions needed from the asset itself or the location where the asset is.

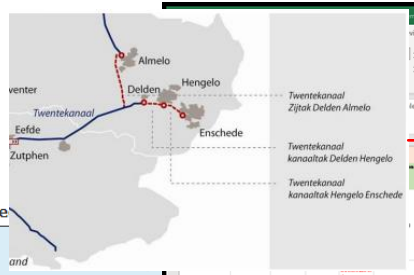
Economic end of life

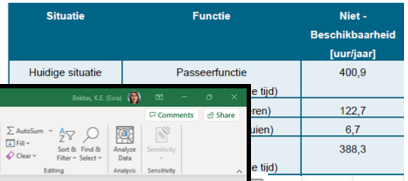
- When an asset becomes too expensive to maintain.
- When EELI indicator of Bakker et al.(2016) comes closer to 1.

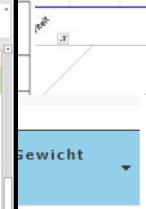
We published the findings in the IALCCE 2018.

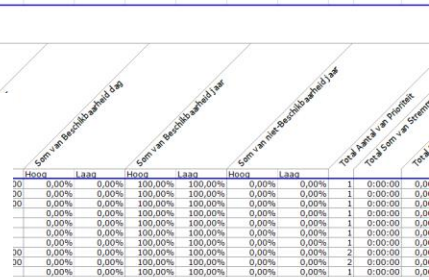
End of Life Rules for Navigation Locks

Delden, Eefde, Hengelo Locks at the Twente Canal
What makes navigation locks to be as V&R candidates?







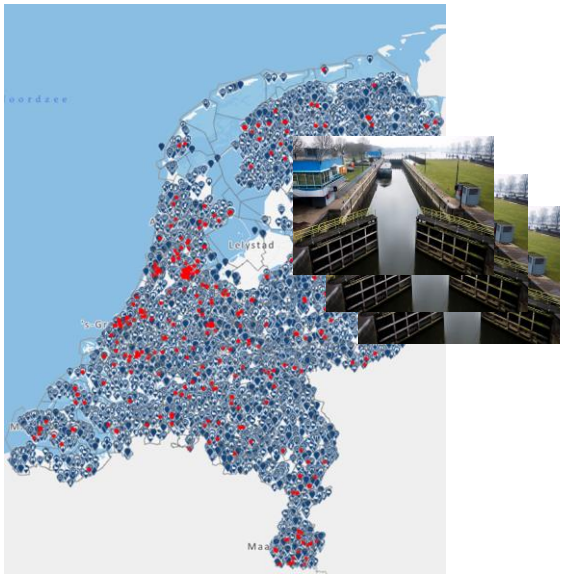


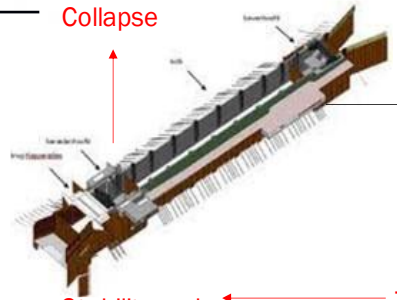
Sluis Eefde	540	3698	9518
Sluis Hagestein	450	135	5060
Sluis Hengelo	540	92	614
Sluis Weurt	504	974	20794
Spoolder sluis	540	241	3813

> Norm	P Min	P Max	P Gem	P 85	P 90	P 95	P 99
0							
0							
0							
2089	103	1079	378,11	528	582	665	859
3407	136	1436	614,22	925	1034	1179	1378
787	112	1150	354,97	478	523	615	895
6199	138	1439	741,63	1110	1198	1307	1414
923	103	985	326,80	459	510	599	786
121	145	1023	418,42	602	679,5	775	1023
6161	120	1427	459,45	667	753	888	1175
287	121	1189	319,23	436,5	500,5	620,5	919

Image Source: Arcadis (2015)

Rule 1: Technical ageing





Replace Walls ← Collapse

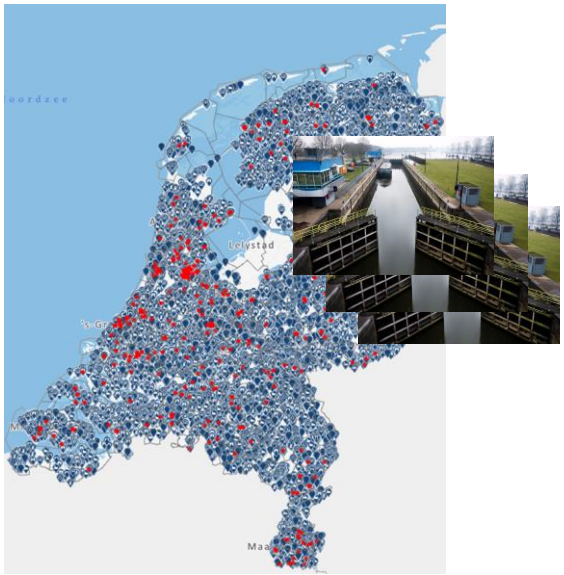
Chamber Wall → Sheet Pile

Steel Profile

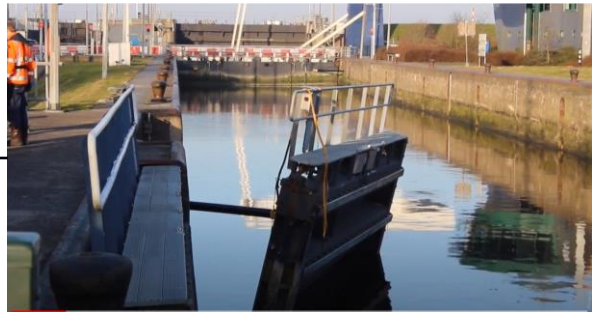
Corrosion → Thickness Reduction → Stability and Strength critical

How many locks and which one have same condition data assigned?

Rule 2: Technical ageing



Obsolete and Out of the stock



How many locks that have the gates registered as obsolete and out of the stock?

Rule: Functional ageing rule

Widen the chamber
Add new chamber

CEMT class not suitable

PREMIUM

De brug bij Macharen stond zaterdagochtend lang open voor de doorvaart van de Galactica. Hier was het aan de zijkanten passen en meten. © Gabor Heeres/Foto Mallo

Vijftien centimeter ruimte bij Hedelse spoorbrug is te weinig om de Galactica te laten passeren

How many locks that became limited to provide these dimensions?

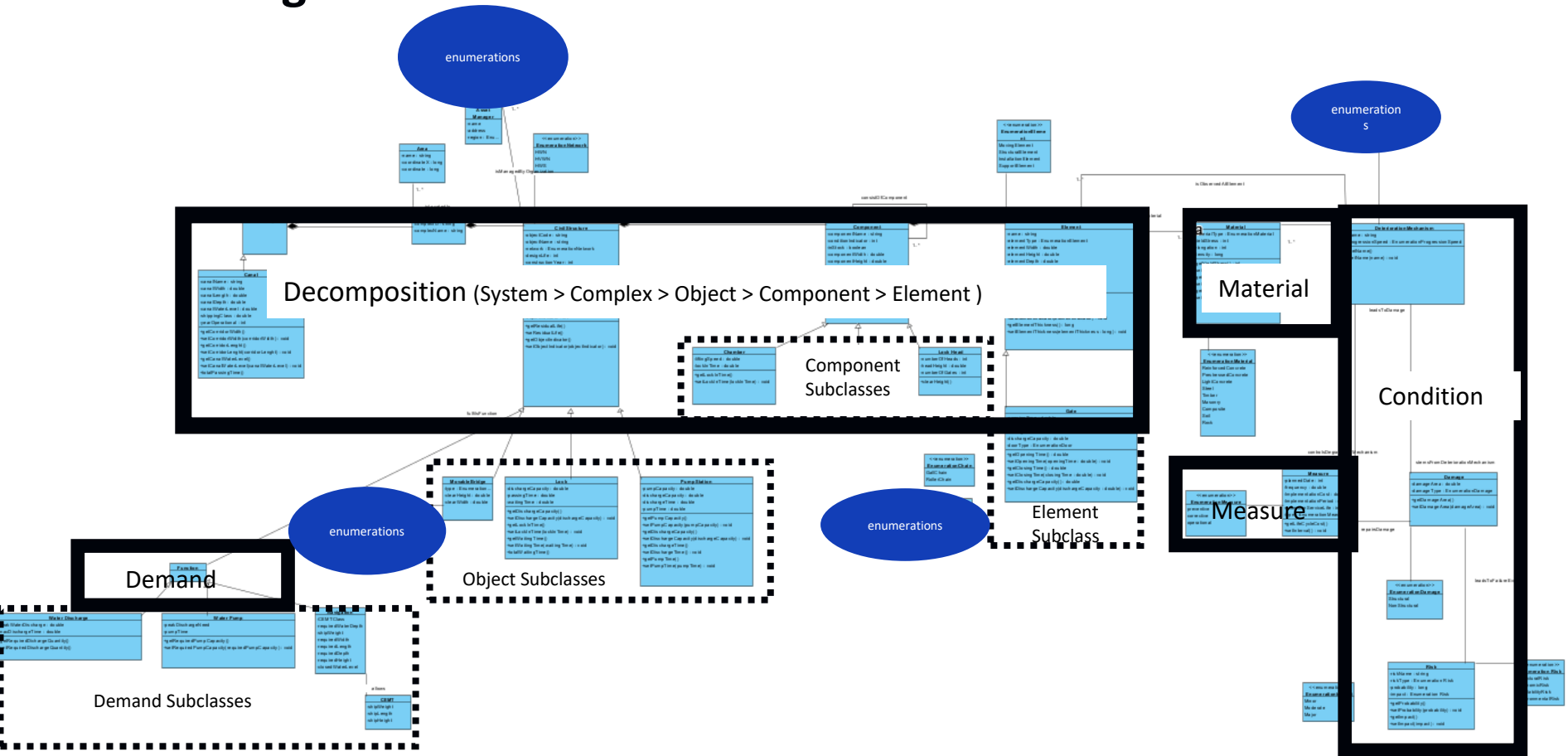
Functional requirements (Shortened)

- How many navigation locks are operational the in the Netherlands?
- What are the functional demands on navigation locks on corridor and component level?
- What are the critical metrics on navigation locks for the demand and on what level can they be quantified?
- What are the critical components of navigation locks and what are their (sub)functions?
- What are the critical information for the navigation locks from asset managers' perspective?
- What are the condition information for navigation locks and what is a minimum set of information to be assigned to locks in terms to create an overview?

Examples on detailed CQs:

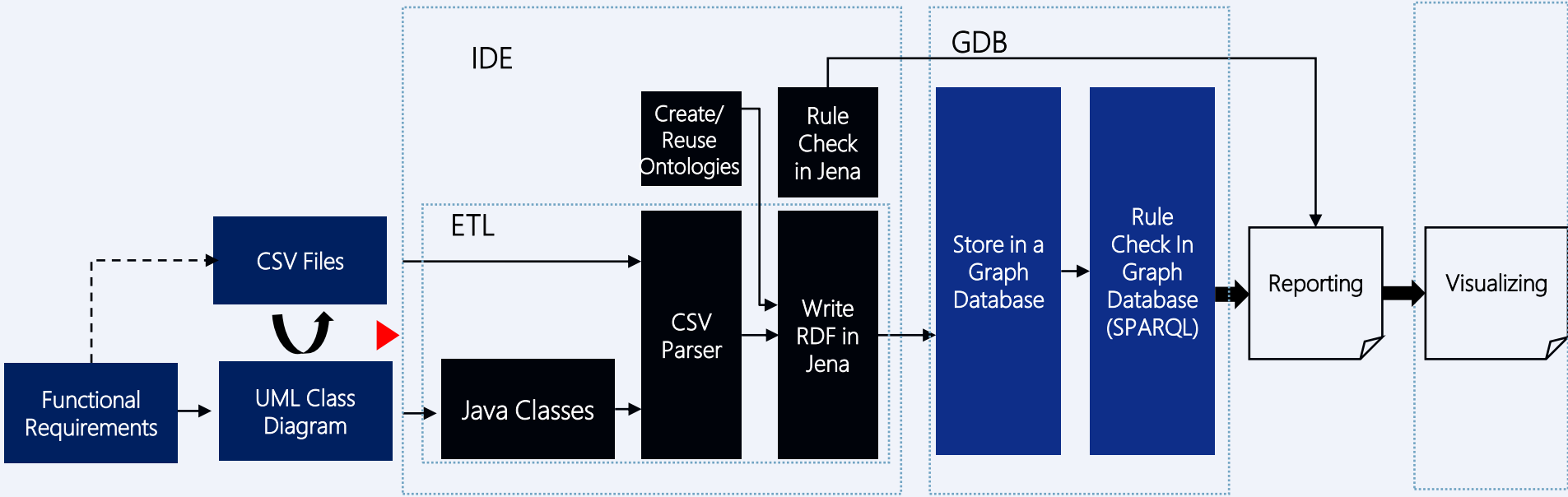
- Which navigation locks are monumental?
- How many navigation locks that have gates obsolete and not in-stock?
- What are the CEMT class types of navigation locks, what are the required dimension per CEMT class?
- What are the dimensions of the locks and also their critical components e.g. chambers, waiting area, lock-head?
- Is the CEMT class of the corridor the same with the CEMT class of the navigation locks?
- What is the waiting time for ships defined by the norm and measured at the locks?
- What are the other related objects of a complex, what are their critical properties e.g. clear height that may limit the ship dimensions that are essential for the function of lock?

UML Diagram


enumerations
enumeration
s
Decomposition (System > Complex > Object > Component > Element)
Material
Component
Subclasses
Condition
Element
Subclass
enumerations
enumerations
Measure
Demand
Object Subclasses
Demand Subclasses

The route of using the Semantic Web Technologies

Use of Apache Jena Framework

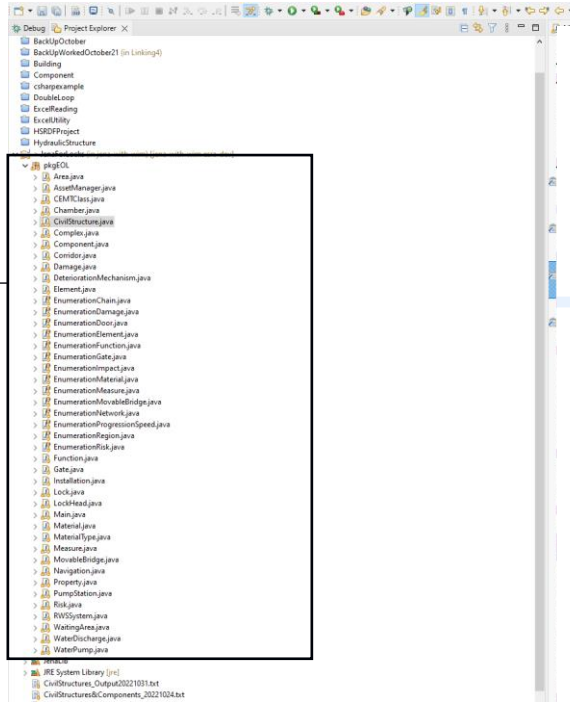


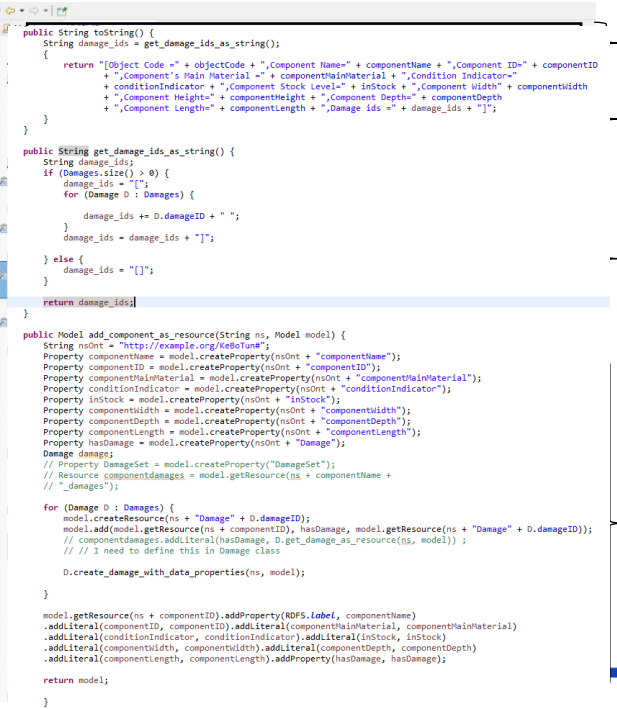
Java Classes

- Open source tooling (Jena Framework) to create model.

Corridor, Complex, Object. Component. Element, Damage, Risk, Demand etc.

15 Java Classes & Parsing data (200 locks, 1000 components…)





```
public String toString() {
    String damage_ids = get_damage_ids_as_string();
    {
        return "[Object Code =" + objectCode + ",Component Name=" + componentName + ",Component ID=" + componentID
                + ",Component's Main Material =" + componentMainMaterial + ",Condition Indicator="
                + conditionIndicator + ",Component Stock Level=" + inStock + ",Component Width" + componentWidth
                + ",Component Height=" + componentHeight + ",Component Depth=" + componentDepth
                + ",Component Length=" + componentLength + ",Damage ids =" + damage_ids + "]";
    }
}

public String get_damage_ids_as_string() {
    String damage_ids;
    if (Damages.size() > 0) {
        damage_ids = "[";
        for (Damage D : Damages) {

            damage_ids += D.damageID + " ";
        }
        damage_ids = damage_ids + "]";

    } else {
        damage_ids = "[]";
    }

    return damage_ids;
}

public Model add_component_as_resource(String ns, Model model) {
    String nsOnt = "http://example.org/KeBoTun#";
    Property componentName = model.createProperty(nsOnt + "componentName");
    Property componentID = model.createProperty(nsOnt + "componentID");
    Property componentMainMaterial = model.createProperty(nsOnt + "componentMainMaterial");
    Property conditionIndicator = model.createProperty(nsOnt + "conditionIndicator");
    Property inStock = model.createProperty(nsOnt + "inStock");
    Property componentWidth = model.createProperty(nsOnt + "componentWidth");
    Property componentDepth = model.createProperty(nsOnt + "componentDepth");
    Property componentLength = model.createProperty(nsOnt + "componentLength");
    Property hasDamage = model.createProperty(nsOnt + "Damage");
    Damage damage;
    // Property DamageSet = model.createProperty("DamageSet");
    // Resource componentdamages = model.getResource(ns + componentName +
    // "_damages");

    for (Damage D : Damages) {
        model.createResource(ns + "Damage" + D.damageID);
        model.add(model.getResource(ns + componentID), hasDamage, model.getResource(ns + "Damage" + D.damageID));
        // componentdamages.addLiteral(hasDamage, D.get_damage_as_resource(ns, model)) ;
        // // I need to define this in Damage class

        D.create_damage_with_data_properties(ns, model);

    }

    model.getResource(ns + componentID).addProperty(RDFS.label, componentName)
    .addLiteral(componentID, componentID).addLiteral(componentMainMaterial, componentMainMaterial)
    .addLiteral(conditionIndicator, conditionIndicator).addLiteral(inStock, inStock)
    .addLiteral(componentWidth, componentWidth).addLiteral(componentDepth, componentDepth)
    .addLiteral(componentLength, componentLength).addProperty(hasDamage, hasDamage);

    return model;

}
```

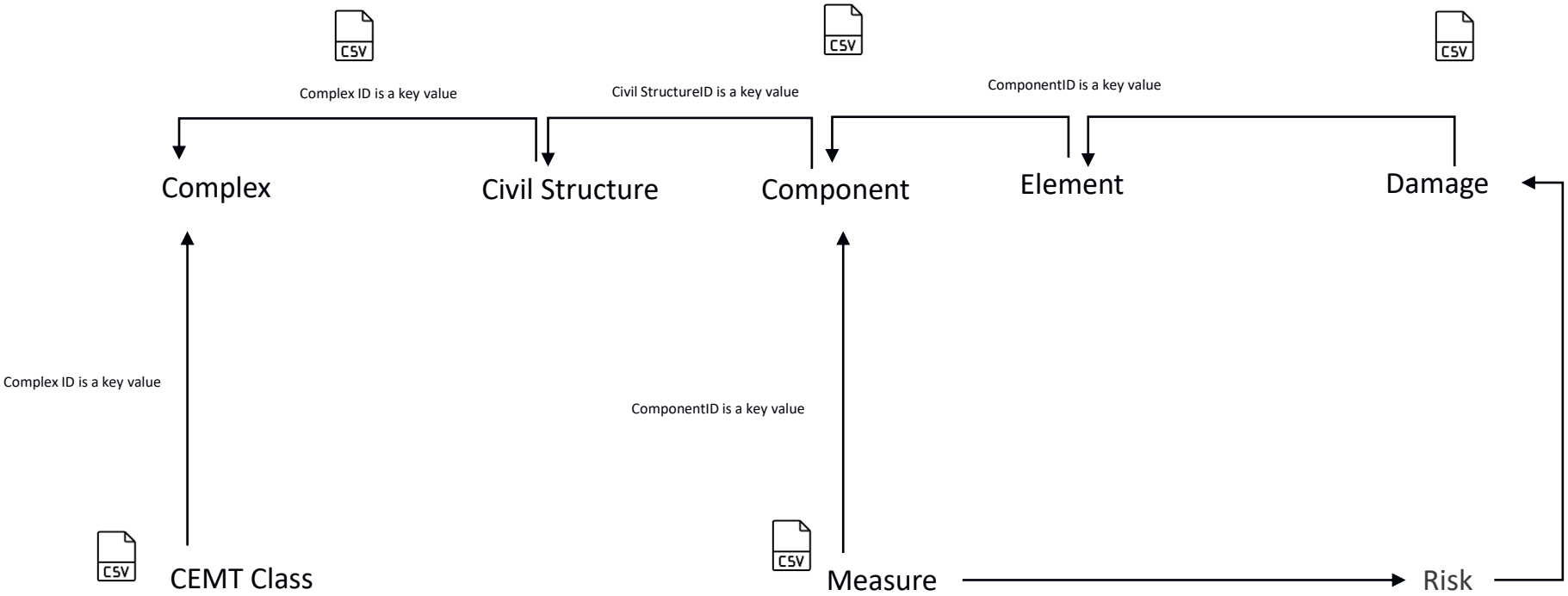
Data Properties

Object Properties

Methods

Initialization of Classes (Data read from 5 CSV files)

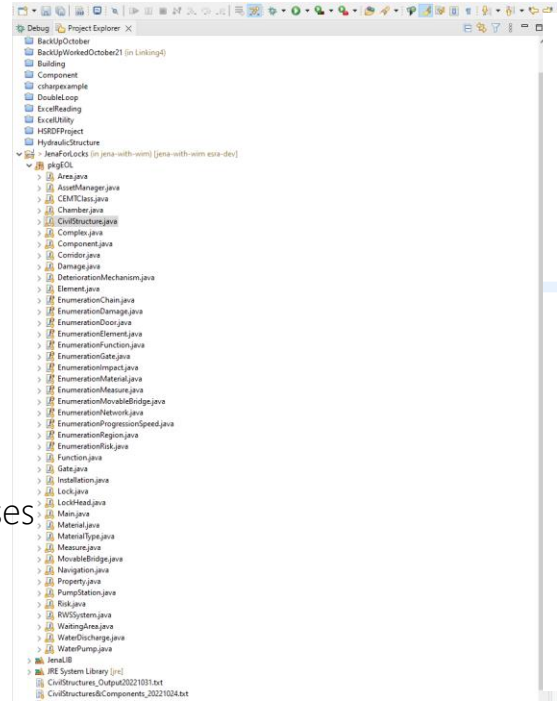
Proof of concept: 3 rules, 5 Classes



DISK, RUPS, Ultimo, Meridian (pdf!),

Using Jena to create triples

Selected Classes for PoC

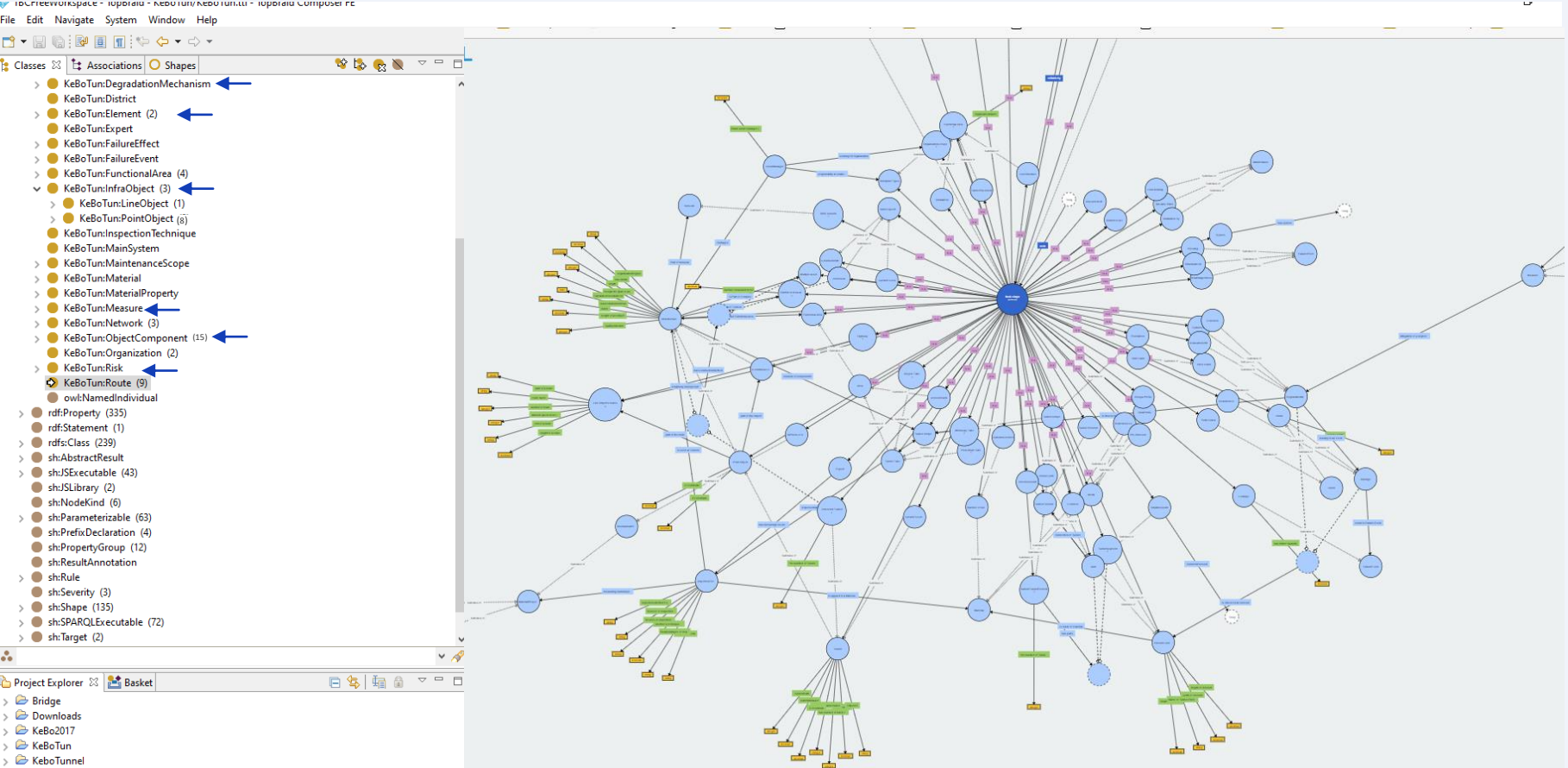


Damage added to Component

```
public String toString() {
    String damage_ids = get_damage_ids_as_string();
    {
        return "[Object Code =" + objectCode + ",Component Name=" + componentName + ",Component ID=" + componentID
                + ",Component's Main Material =" + componentMainMaterial + ",Condition Indicator="
                + conditionIndicator + ",Component Stock Level=" + inStock + ",Component Width" + componentWidth
                + ",Component Height=" + componentHeight + ",Component Depth=" + componentDepth
                + ",Component Length=" + componentLength + ",Damage ids =" + damage_ids + "]";
    }
}

public String get_damage_ids_as_string() {
    String damage_ids;
    if (Damages.size() > 0) {
        damage_ids = "[";
        for (Damage D : Damages) {

            damage_ids += D.damageID + " ";
        }
        damage_ids = damage_ids + "]";

    } else {
        damage_ids = "[]";
    }

    return damage_ids;
}

public Model add_component_as_resource(String ns, Model model) {
    String nsOnt = "http://example.org/KeBoTun#";
    Property componentName = model.createProperty(nsOnt + "componentName");
    Property componentID = model.createProperty(nsOnt + "componentID");
    Property componentMainMaterial = model.createProperty(nsOnt + "componentMainMaterial");
    Property conditionIndicator = model.createProperty(nsOnt + "conditionIndicator");
    Property inStock = model.createProperty(nsOnt + "inStock");
    Property componentWidth = model.createProperty(nsOnt + "componentWidth");
    Property componentDepth = model.createProperty(nsOnt + "componentDepth");
    Property componentLength = model.createProperty(nsOnt + "componentLength");
    Property hasDamage = model.createProperty(nsOnt + "Damage");
    Damage damage;
    // Property DamageSet = model.createProperty("DamageSet");
    // Resource componentdamages = model.getResource(ns + componentName +
    // "_damages");

    for (Damage D : Damages) {
        model.createResource(ns + "Damage" + D.damageID);
        model.add(model.getResource(ns + componentID), hasDamage, model.getResource(ns + "Damage" + D.damageID));
        // componentdamages.addLiteral(hasDamage, D.get_damage_as_resource(ns, model)) ;
        // // I need to define this in Damage class

        D.create_damage_with_data_properties(ns, model);

    }

    model.getResource(ns + componentID).addProperty(RDFS.label, componentName)
    .addLiteral(componentID, componentID).addLiteral(componentMainMaterial, componentMainMaterial)
    .addLiteral(conditionIndicator, conditionIndicator).addLiteral(inStock, inStock)
    .addLiteral(componentWidth, componentWidth).addLiteral(componentDepth, componentDepth)
    .addLiteral(componentLength, componentLength).addProperty(hasDamage, hasDamage);

    return model;

    }
```

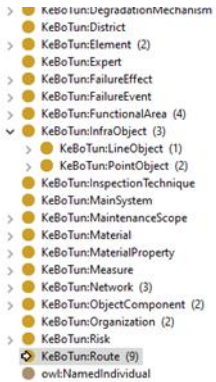
- Getting damage properties through key values (e.g. ID's)
- Add component as resource and create its properties via KeBoTun ontology
- Create damage as resources and add properties (by getting component resource)
- Get component resource and add properties

Reusing KeBoTun



Creating the RDF file (just where we are)

- First output as RDF – theoretically it should works
- Some thing need to be fixed



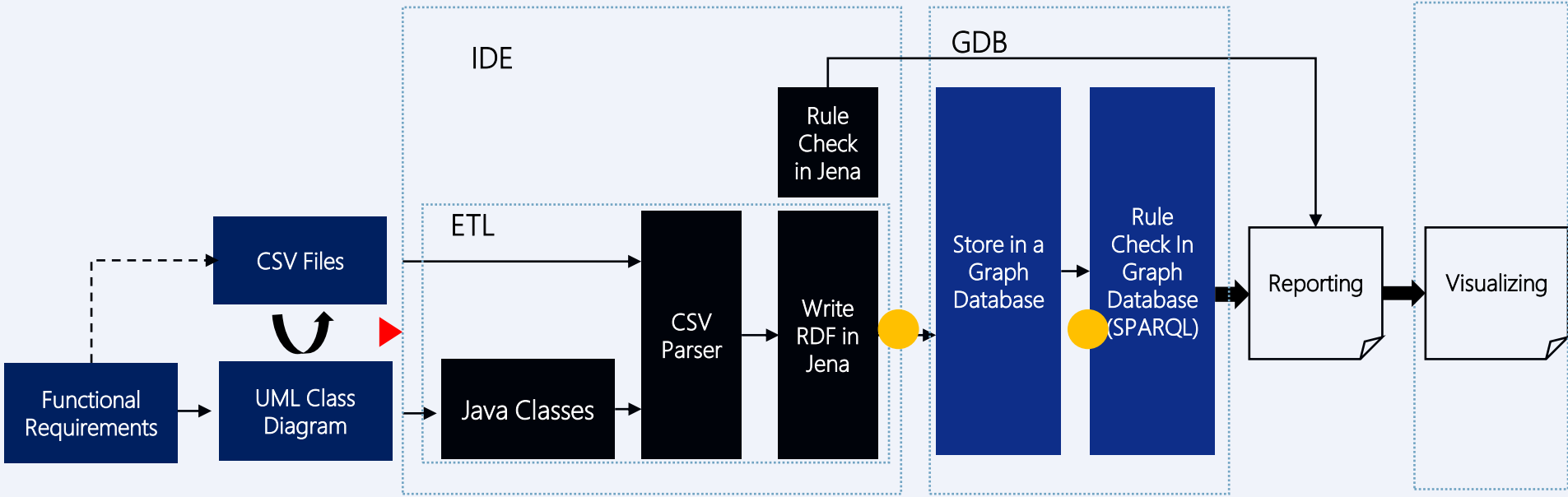
File Edit Format View Help

```
:https://www.keBo_sluices.nl#10B-001-05>
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Lorentz schutsluizen westkolk" ;
        <https://www.kebo_sluices.nl/Ontology#constructionYear>
                "1931"^^<http://www.w3.org/2001/XMLSchema#long> ;
        <https://www.kebo_sluices.nl/Ontology#damage>
                <https://www.keBo_sluices.nl#Damage113> ;
        <https://www.kebo_sluices.nl/Ontology#length>
                "1555.0"^^<http://www.w3.org/2001/XMLSchema#double> ;
        <https://www.kebo_sluices.nl/Ontology#manager>
                "MN District Noord" ;
        <https://www.kebo_sluices.nl/Ontology#network>
                "HVWN" ;
        <https://www.kebo_sluices.nl/Ontology#objectCode>
                "10B-001-05" ;
        <https://www.kebo_sluices.nl/Ontology#objectHistory>
                "Rood" ;
        <https://www.kebo_sluices.nl/Ontology#qualityIndicator>
                "5"^^<http://www.w3.org/2001/XMLSchema#long> ;
        <https://www.kebo_sluices.nl/Ontology#type>
                "Schutsluizen" ;
        <https://www.kebo_sluices.nl/Ontology#width>
                "0.0"^^<http://www.w3.org/2001/XMLSchema#double> .

:https://www.keBo_sluices.nl#Damage114>
        <hasDamageArea>  "2.0"^^<http://www.w3.org/2001/XMLSchema#double> ;
        <hasDamageType>  "Poor / insufficient functioning" .

:https://www.keBo_sluices.nl#45B-001-01>
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Sluis St. Andries" ;
        <https://www.kebo_sluices.nl/Ontology#constructionYear>
                "1934"^^<http://www.w3.org/2001/XMLSchema#long> ;
        <https://www.kebo_sluices.nl/Ontology#damage>
                <https://www.keBo_sluices.nl#Damage119> .
```

Ln 34, Col 30

TNO innovation for life

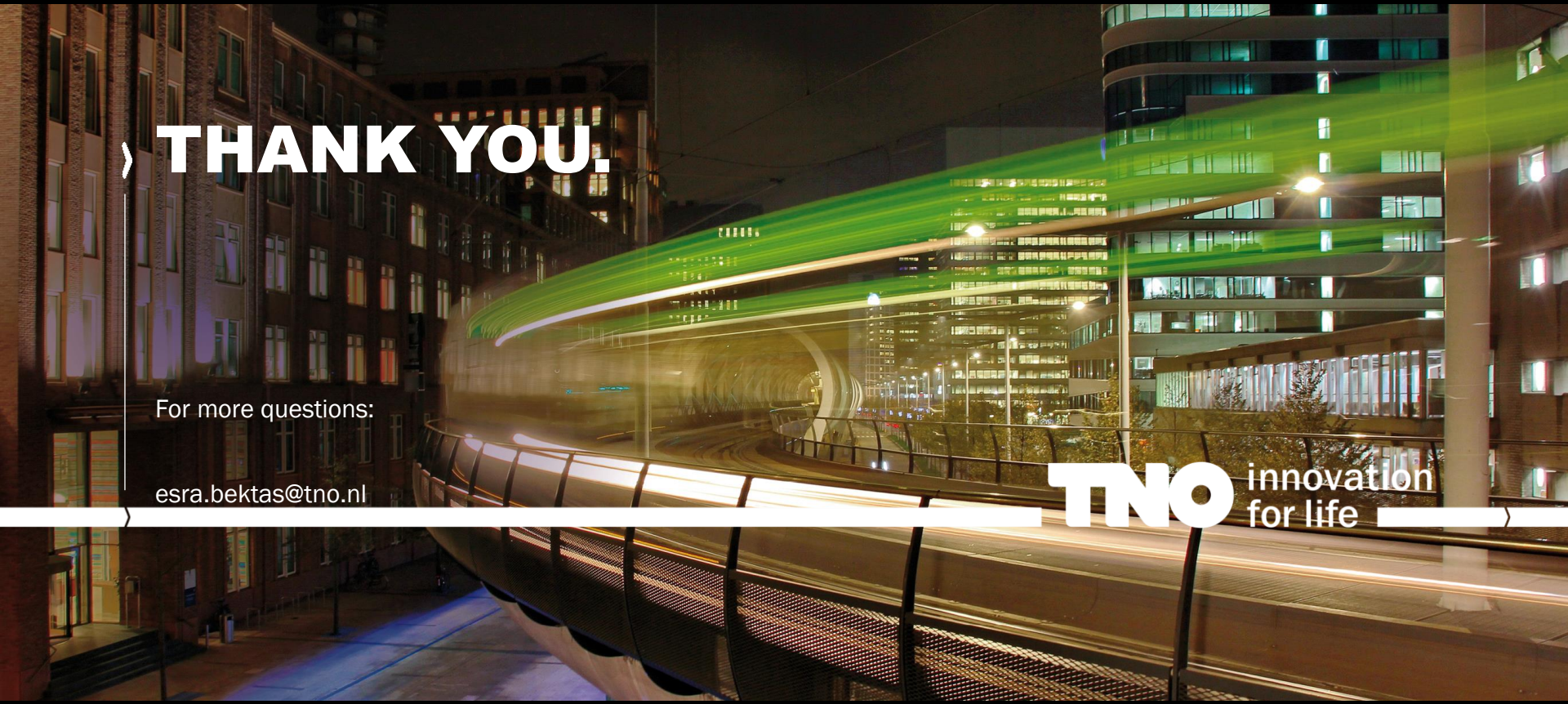
Where we are, where to move forward



Take aways until now, Moving Forward

- **This is a knowledge development project to explore the SW with no dependency for 1 specific decision moment.**
  - Accessing the data from its source is ideal, this is to prove a concept.
- **The major issue was that there were limited actual examples (except Jena documentation).**
  - Once the classes (+parsed data) were linked, Jena part was very intuitive (even though the project is not finished).
- **Most essential decision information comes from programming inspection reports/datasets. Yet that are almost always overlooked.**

- **There is always a top model, instantiated to many object types as *point-object* civil structures.**
  - The 5 main classes (and reused KeBoTun Ontology) have proven that we may need much simpler model for condition definition of the civil infrastructure.
- **There is always typical failure mechanisms associated to object types.**
- **Any additional properties can be defined via e.g. DOT Ontology.**
- **The PoC will be extendible for any civil structure as the data structures we use don't change and top-model don't change.**
  - But next step is to finish the PoC, and move to more ideal situation step by step.
- **Meanwhile we tag along the real assessor/decision maker, who became very enthusiastic (*"Inspection data-template"*)**
- **There is a lot to gain and have real impact on the society, which is shadowed by other society al trends.**


THANK YOU.
For more questions:
esra.bektas@tno.nl
TNO
innovation
for life