Querying Large SysML Models

Annelisa Sturgeon
Aleksander Przybylo
James Milstead

Presentation outline

- **Presenters' bios**
- **Overview of SLATE-FI implementation and use**
- **Migrating to SysML**
  - **What fits into SysML vs. what doesn't**
  - **Occurrence modelling**
- **Query approach trade study**
  - **Basic assumptions and requirement**
  - **The three high level options being studied**
- **Recommendation**
- **Questions?**

Authors' bios

- **Annie Sturgeon**

Joined Boeing full time in 2017 after three internships in Production Engineering. Currently works on MBSE implementation in Everett, WA.

- **Alek Przybylo**

Joined Boeing in 2010 as Process Engineer. Previous experience includes consulting for aerospace companies in the PLM and CAD fields. Currently works on MBSE development in Everett, WA.

- **Jim Milstead**

Software developer with the BR&T Product Lifecycle Modeling & Simulation team in Huntsville, AL. Associate Technical Fellow since 2010.

Overview of SLATE-FI implementation and use

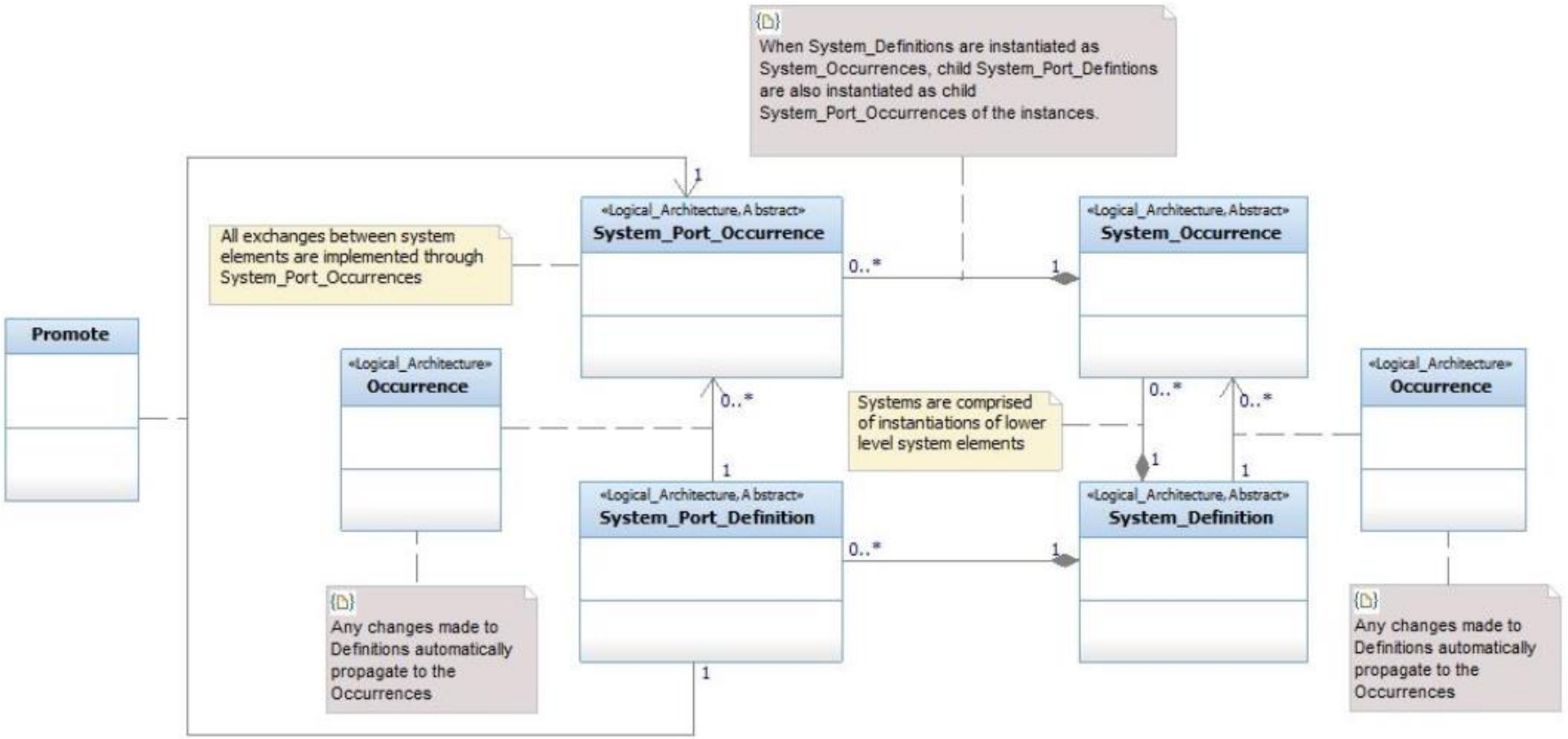
- **SLATE-FI was built from out of the box SLATE and customized by Boeing**
- **Interface Control Design tool used to model logical architectures of systems**
- **First fully functional Model Based Systems Architecting tool**
- **Complex Data Schema**
- **Legacy programs generated large data base from SLATE-FI**

SLATE-FI Occurrence Modeling

- **Used to model Logical Architecture**
- **System Objects are modeled as Definitions**
- **These Objects are instantiated on higher level System Definitions Objects as Occurrence Objects**
- **This allows for re-use of System Objects**
- **Changes made to System Definition Objects are propagated to Occurrences through Occurrence links**

Malone, R., Friedland, B., Herrold, J., & Green, G. 2017 Page 5

Overview of SLATE-FI Occurrence Modeling



Malone, R., Friedland, B., Herrold, J., & Green, G. 2017 Figure 5. Occurrence Data Model

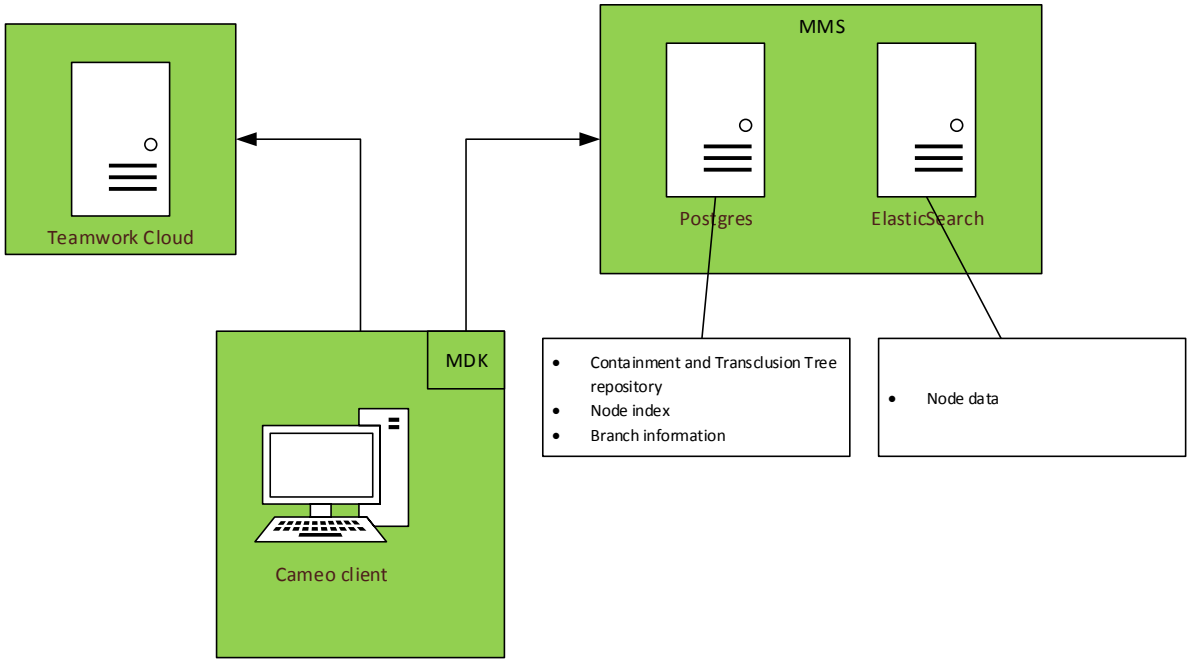
Migrating Model Data from Legacy Representations to SysML

- **Export SLATE-FI data schema to XML file**
- **Import SLATE-FI data objects with attributes into a UML Profile**
  - **Objects as Stereotypes**
  - **Attributes as Properties on corresponding Stereotypes**
- **Identify SysML representation of SLATE-FI data objects and relationships**
  - **Update Profile Elements with SysML representations**
  - **Use Profile in a Data Model to define allowed relationships between profile objects**
- **Import SLATE-FI data using Profile with SysML representations**

Basic assumptions and user requirements

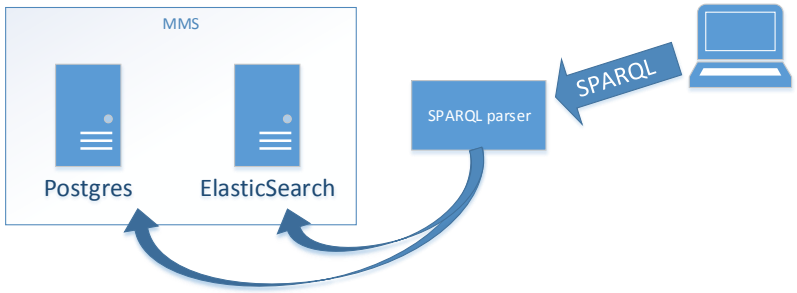
- **Queried data needs to be live (i.e. we should not rely on daily batch replications)**
- **Query response time needs to be “reasonable” for all existing use cases**
- **SPARQL language must be supported**

Current system architecture

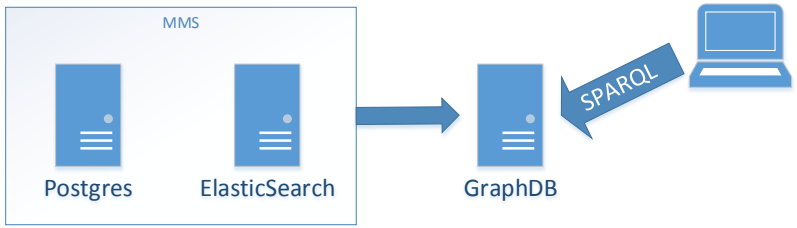


High level solution concepts

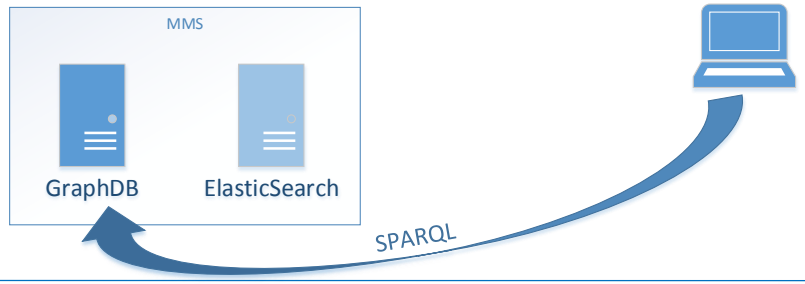
- **Option A: SPARQL parsing**



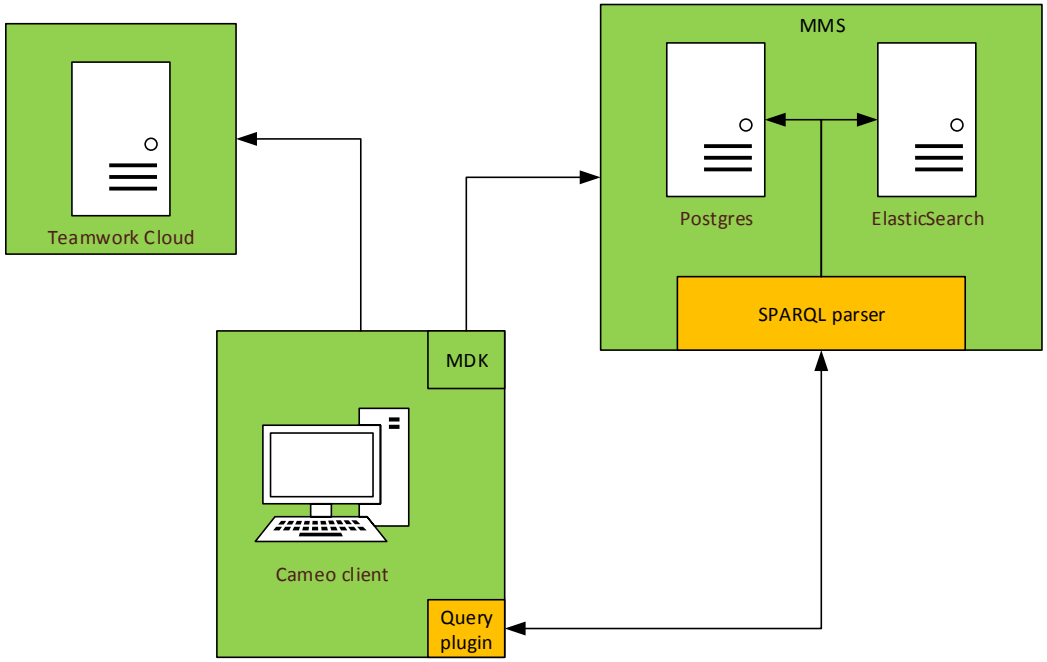
- **Option B: Additional graph repository**



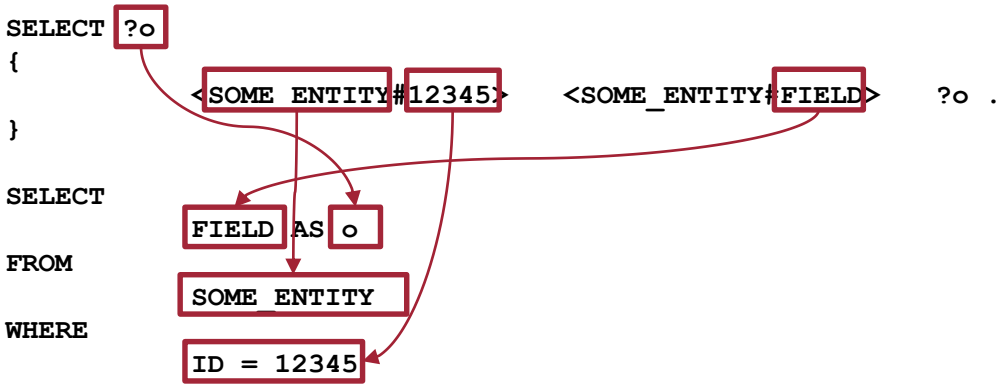
- **Option C: MMS component replacement**

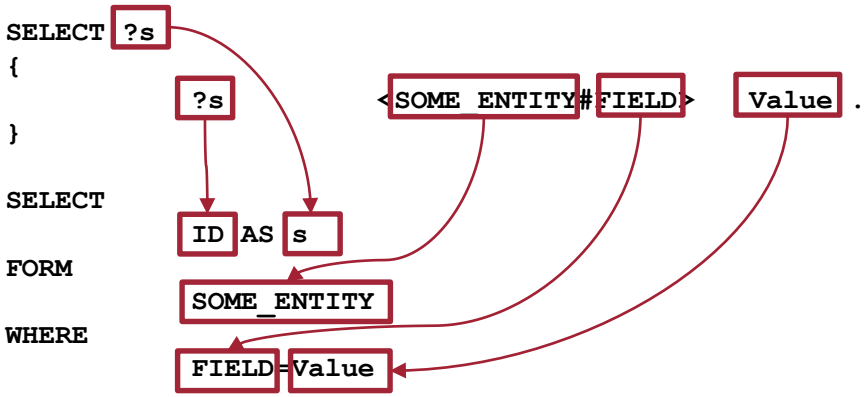


Option A – SPARQL parsing



SPARQL parsing (SPARQL -> SQL)

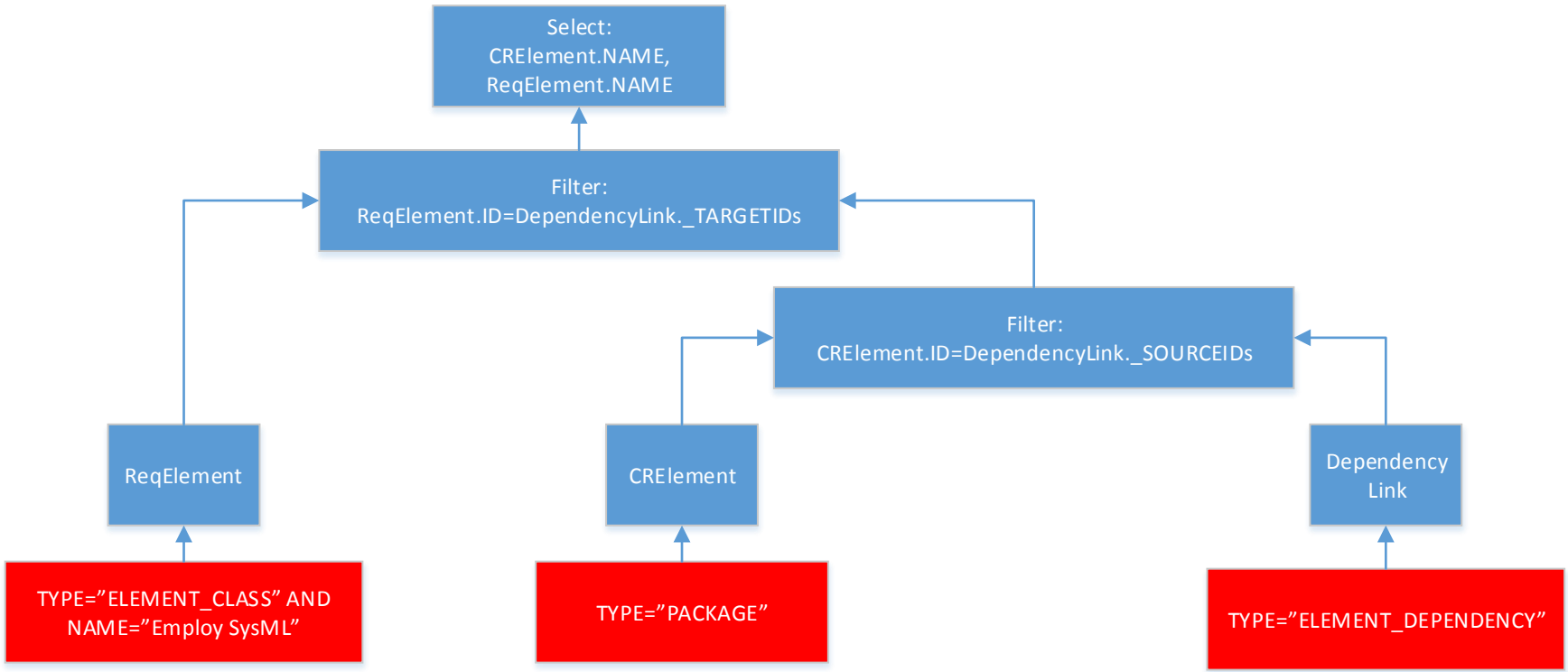




SPARQL parsing (SPARQL -> SQL)

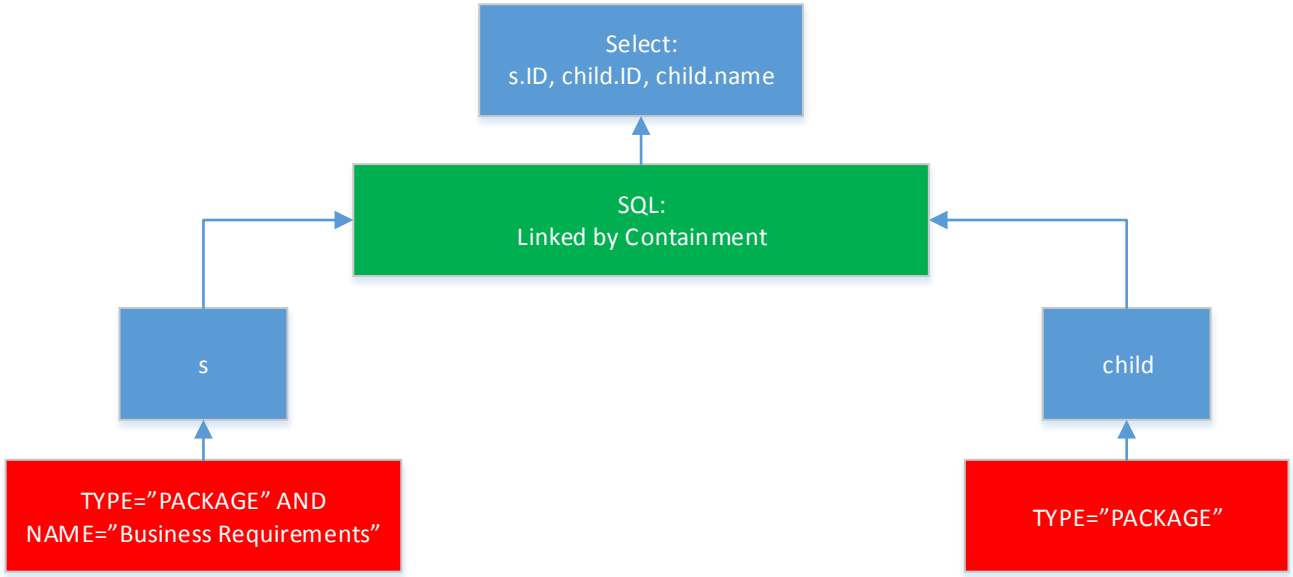

```
SELECT ?CRName ?ReqName
{
            ?CRElement <http://xml.web.boeing.com/RDF/PACKAGE.rdf#NAME> ?CRName .
            ?DependencyLink <http://xml.web.boeing.com/RDF/ELEMENT_DEPENDENCY.rdf#_SOURCEIDS>
?CRElement .
            ?DependencyLink <http://xml.web.boeing.com/RDF/ELEMENT_DEPENDENCY.rdf#_TARGETIDS>
?ReqElement .
            ?ReqElement <http://xml.web.boeing.com/RDF/ELEMENT_CLASS.rdf#NAME> "Employ SysML" .
            ?ReqElement <http://xml.web.boeing.com/RDF/ELEMENT_CLASS.rdf#NAME> ?ReqName .
}

SELECT
            PACKAGE.NAME AS CRName,
            ELEMENT_CLASS.NAME AS ReqName
FROM
            PACKAGE
            INNER JOIN ELEMENT_DEPENDENCY ON PACKAGE.ID = ELEMENT_DEPENDENCY._SOURCEIDS
            INNER JOIN ELEMENT_CLASS ON ELEMENT_CLASS.ID = ELEMENT_DEPENDENCY._TARGETIDS
WHERE
            ELEMENT_CLASS.NAME="Employ SysML"
```

SPARQL parsing (SPARQL -> SQL+ES)

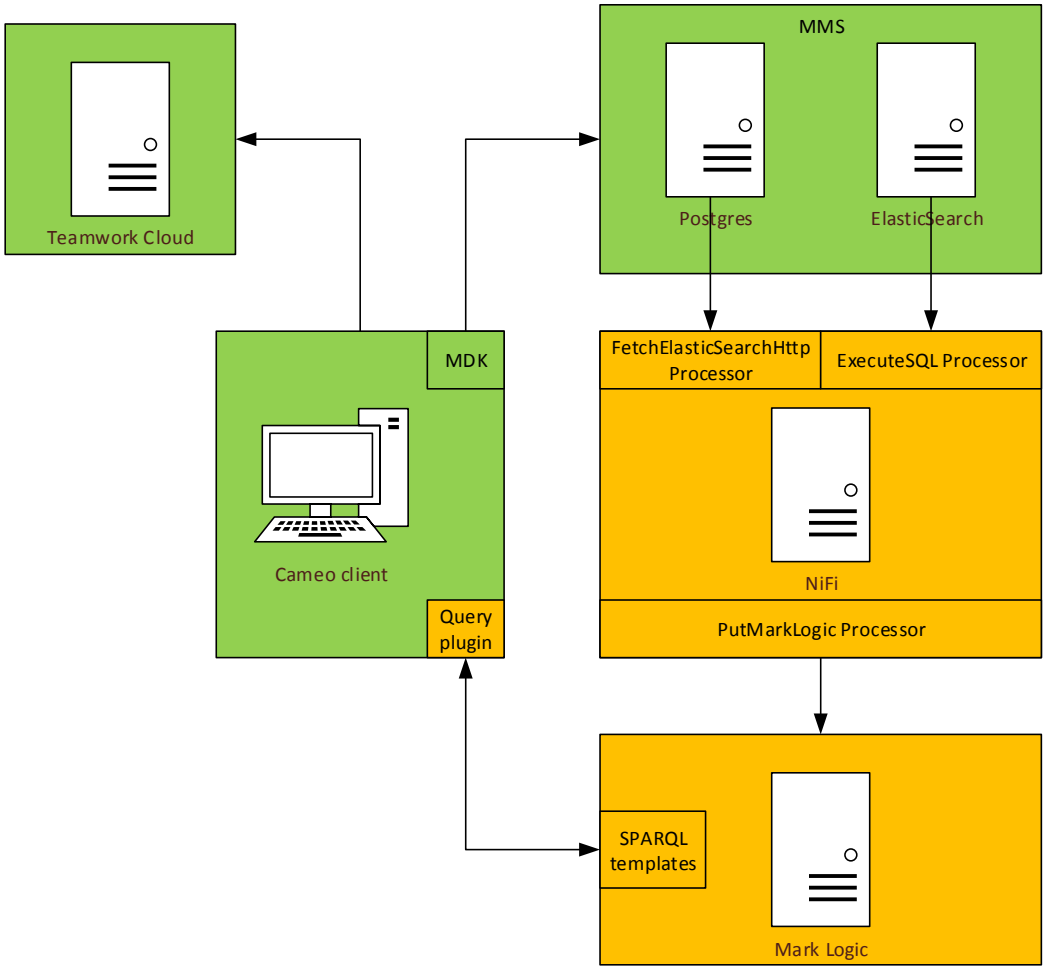


SPARQL parsing (SPARQL -> SQL+ES)

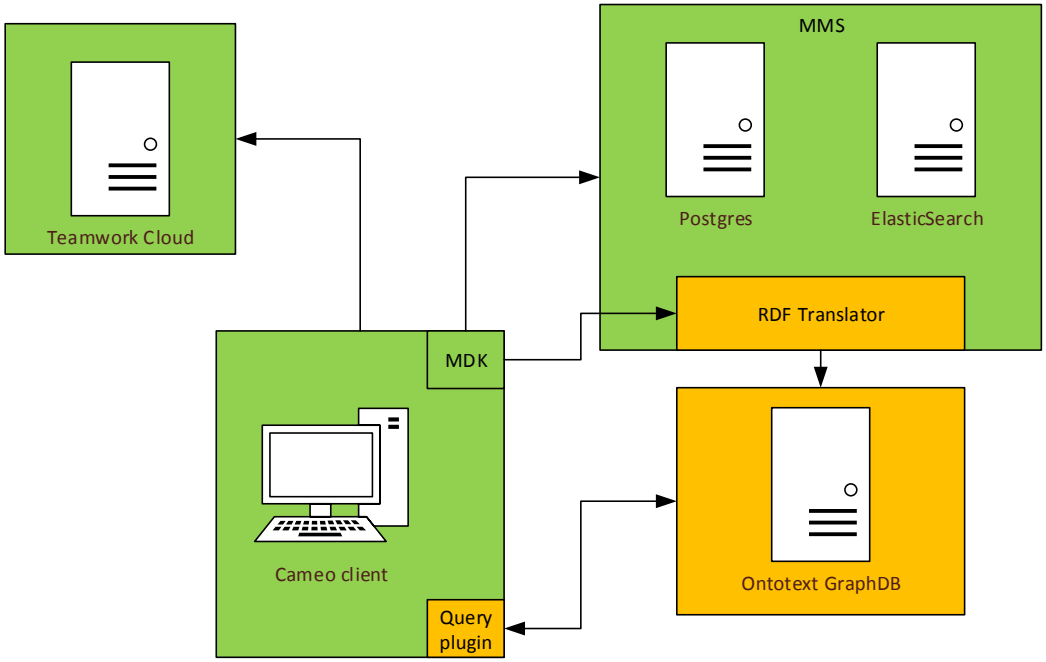

```
SELECT ?s ?child ?child_name
{
   ?s        <http://xml.web.boeing.com/RDF/NODE.rdf#CONTAINMENT_CHILD>      ?child .
   ?s        <http://xml.web.boeing.com/RDF/PACKAGE.rdf#name>                "Business Requirements" .
   ?child    <http://xml.web.boeing.com/RDF/PACKAGE.rdf#name>                ?child_name
}
```



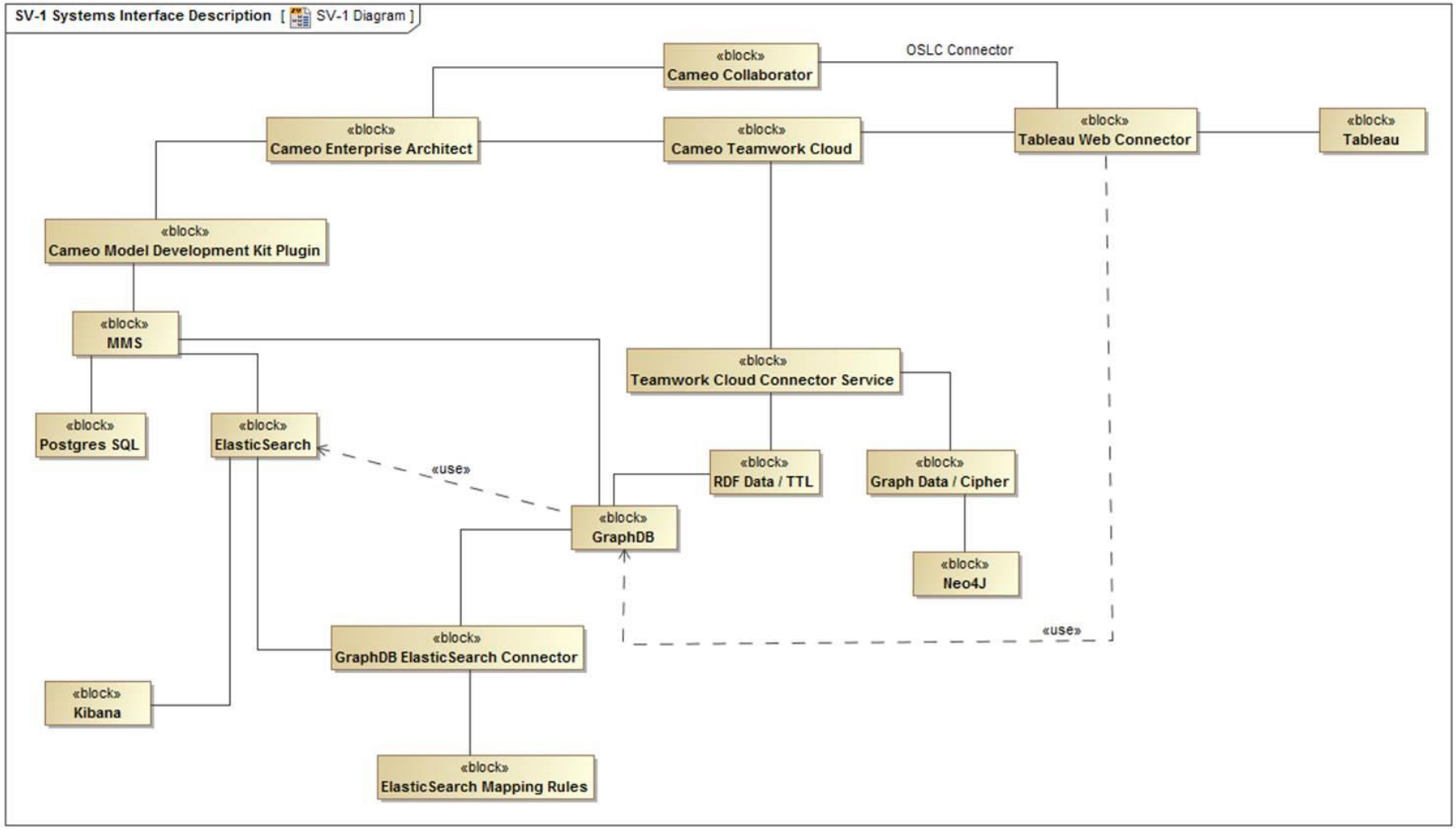
Option B – Additional graph repository (Mark Logic)



Option B – Additional graph repository (GraphDB)



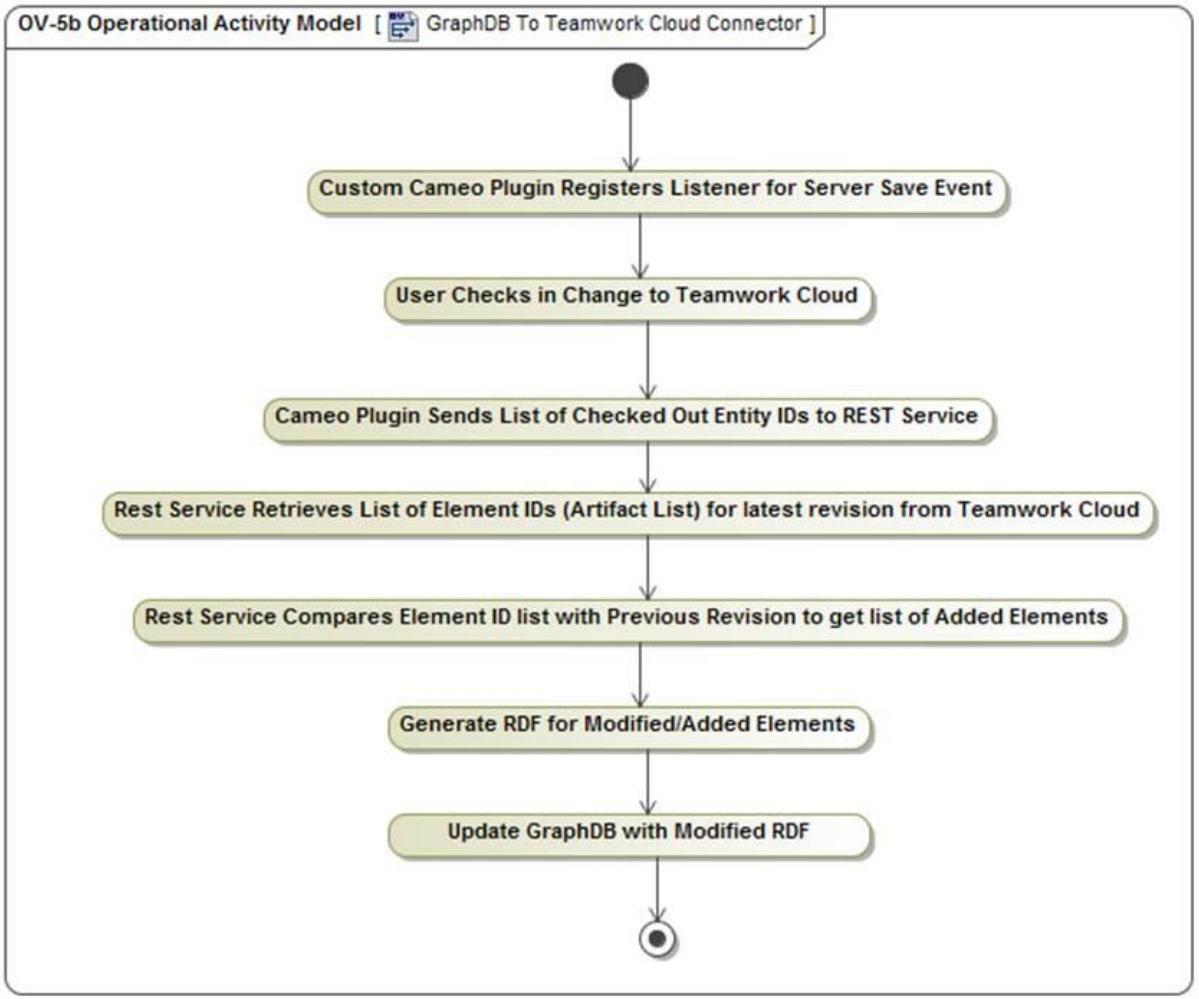
Option C – MMS component replacement (GraphDB)



Option C – MMS component replacement (Neo4j)

- **Similar approach to GraphDB**
  - **GraphAware Neo4j replicates graph into ElasticSearch**
- **Relies on proprietary graph schema and query langage (Cypher)**
- **Similar visual graph capabilities to GraphDB**

Option D – Bypass MMS

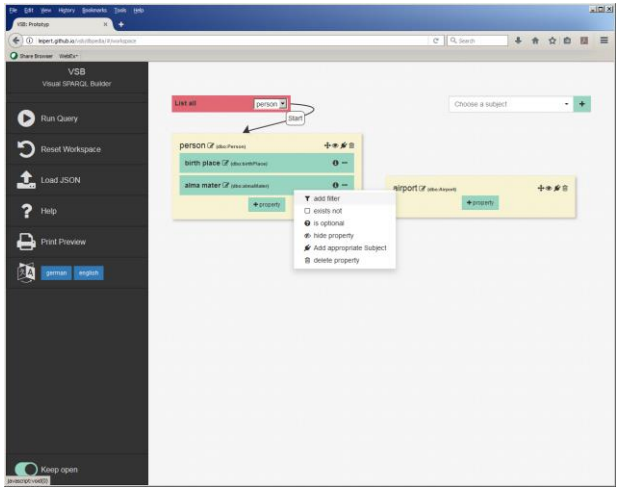


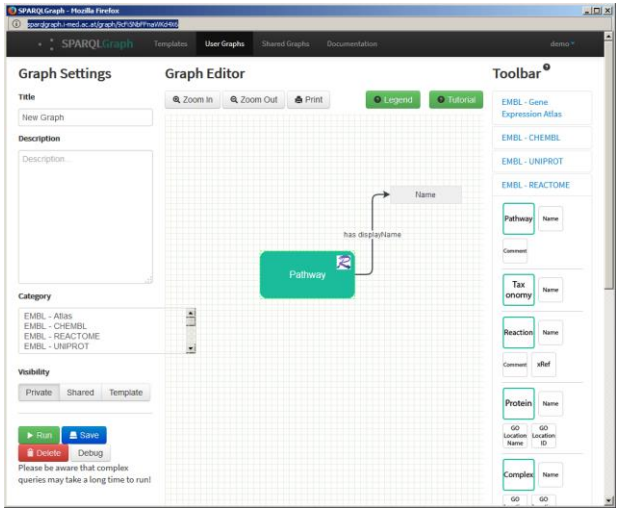
Other options (not explored)

- **Cameo Collaborator OSLC**
  - **Allows for real-time query without synchronization**
  - **Allows exposing part of model as OSLC endpoint**
- **3DX Platform – EXALEAD**
  - **White paper has few technical details**
- **Anzo**
  - **Claims ability to handle 1T triples**
  - **Nice dashboard capability – visual query builder**
- **Sparql plugin**
  - **Developed at GeorgiaTech**
  - **Appears to run a full translate to RDF for each query**

SPARQL UI

- **Visual SPARQL Builder**
  - **Code available at https://github.com/leipert/vsb**
    - **Code not touched for 3 years**
    - **MIT License (good for commercial)**
  - **Configuration needed to connect to custom endpoint**
    - **Could not find list of verified supported endpoints**
- **SPARQLGraph**
  - **Code available at https://github.com/tadKeys/sparqlgraph**
    - **Code not touched for 4 years**
    - **GPL 2.0 License (changes must also be open source)**
  - **Highly customized to medical application**
    - **Open source could be modified**

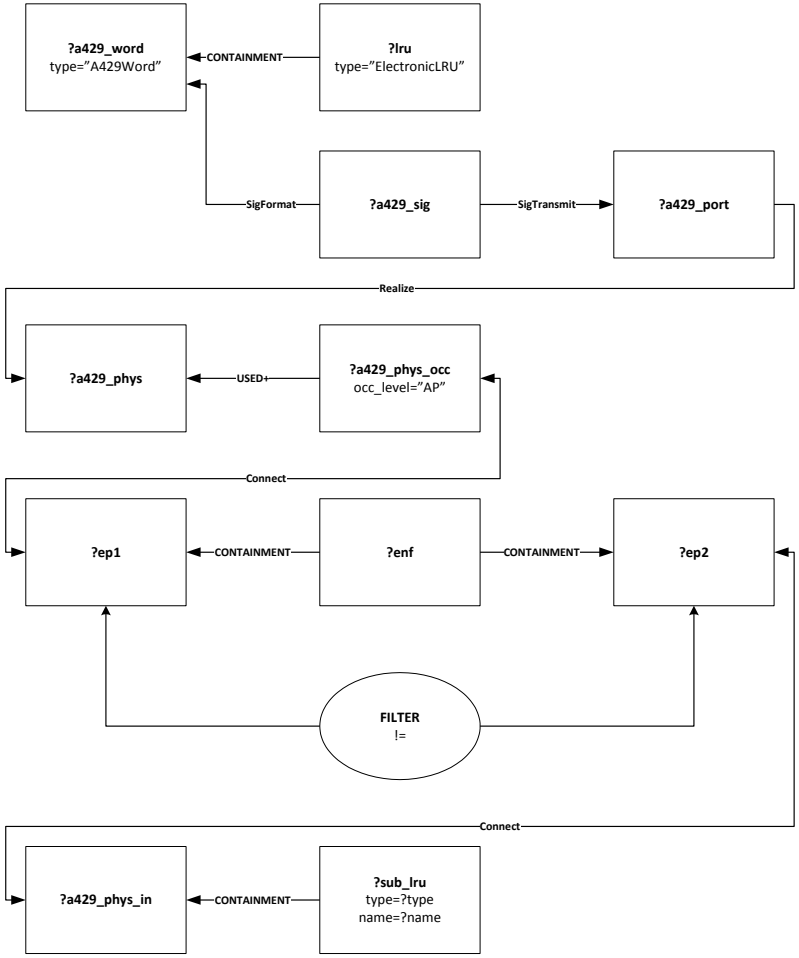




Evaluation criteria

- **Performance (<1s/1-5s/5-15s/>15s)**
- **Impact to commit performance (none/low/medium/high)**
- **Data duplication (none/low/medium/high)**
- **Data transformation (none/low/medium/high)**
- **Coding effort required (none/low/medium/high)**
- **License cost (none/low/medium/high)**
- **Future-proofness (low/medium/high)**

Benchmark query



Evaluation criteria

Option	Performance	Impact to commit	Data duplication	Data transformation	Coding effort	License cost	Future-proofness
Cameo report		None	None	None	None	None	High
Option A – Query parser		None	None	None	High	None	Medium
Option B – External RDF store							
MarkLogic		None	Medium	Low	None	High	High
GraphDB (Free)		Medium	Medium	High	Low	None (*)	Medium
Option C – MMS component replacement							
GraphDB (Enterprise) (**)		Low	Low	Medium	Medium	Medium	Low
Neo4j (Enterprise)		Low	Low	Medium	Medium	Medium	Low
Option D – Bypass MMS (GraphDB)		Medium	High	High	Medium	None (*)	Medium

(*) – has limitations on parallel query (2) => Would potentially need Enterprise version anyways
(**) – license needed for ElasticSearch connector

Recommendations

- **Two recommended approaches**
  - **Bypass MMS and integrate Teamwork Cloud directly with GraphDB (no license cost but high development cost)**
  - **Mark Logic solution (lowest development cost but high license cost)**

Questions