Utilizing Generative AI for Assisted Graph Queries in Model Based Systems Engineering

Aleksander Przybylo

Gabriel Ramirez

Note: Presentation released under **RROI 726487**

To Be Discussed:

1. Domain
2. The Issue/Idea
3. The Approach
4. Experimentation Methodology
5. Results/Observations
6. Conclusion
7. Future Experimentation

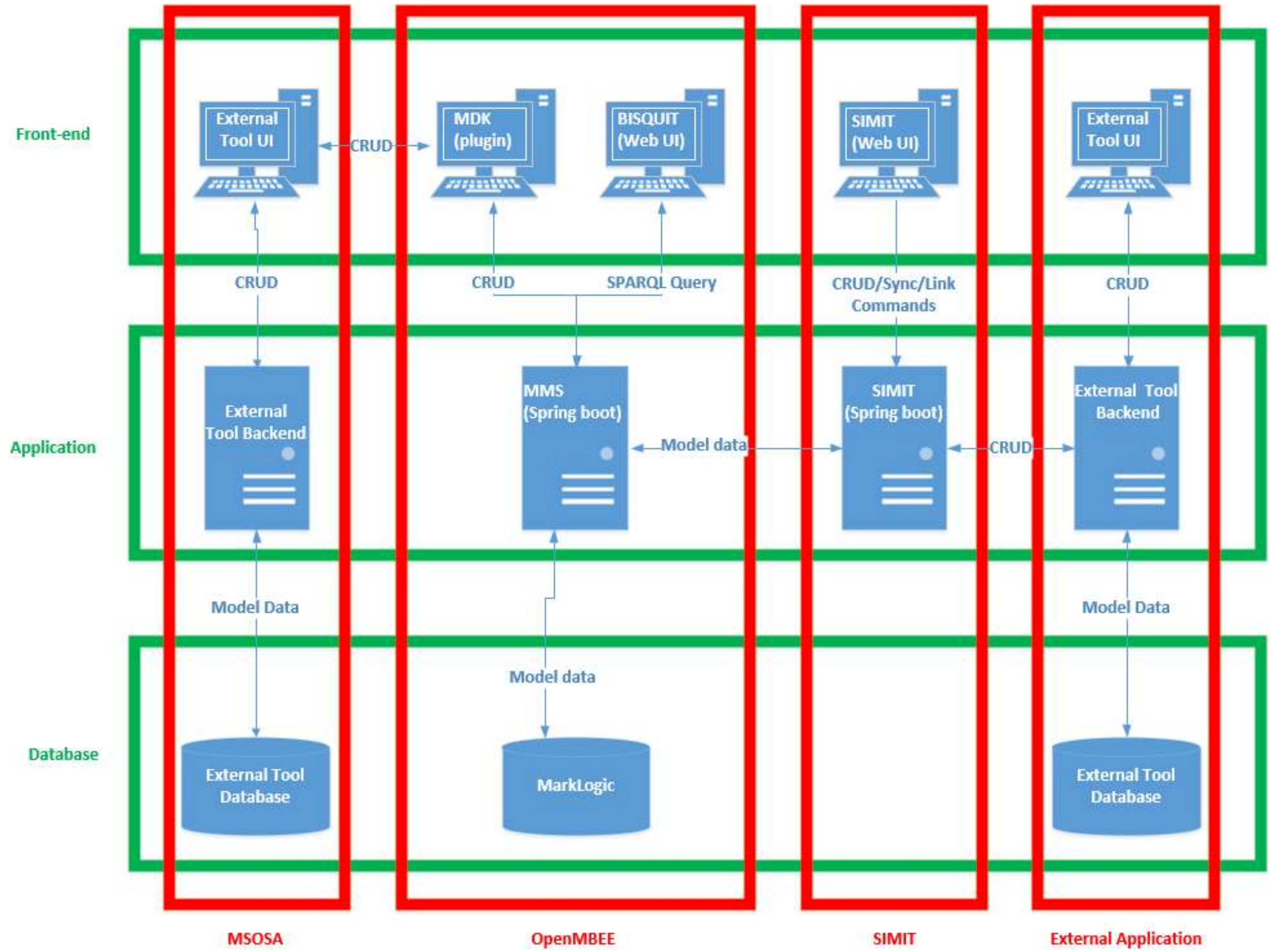
BISQUIT= Boeing Intelligent SPARQL Query User Interface Tool

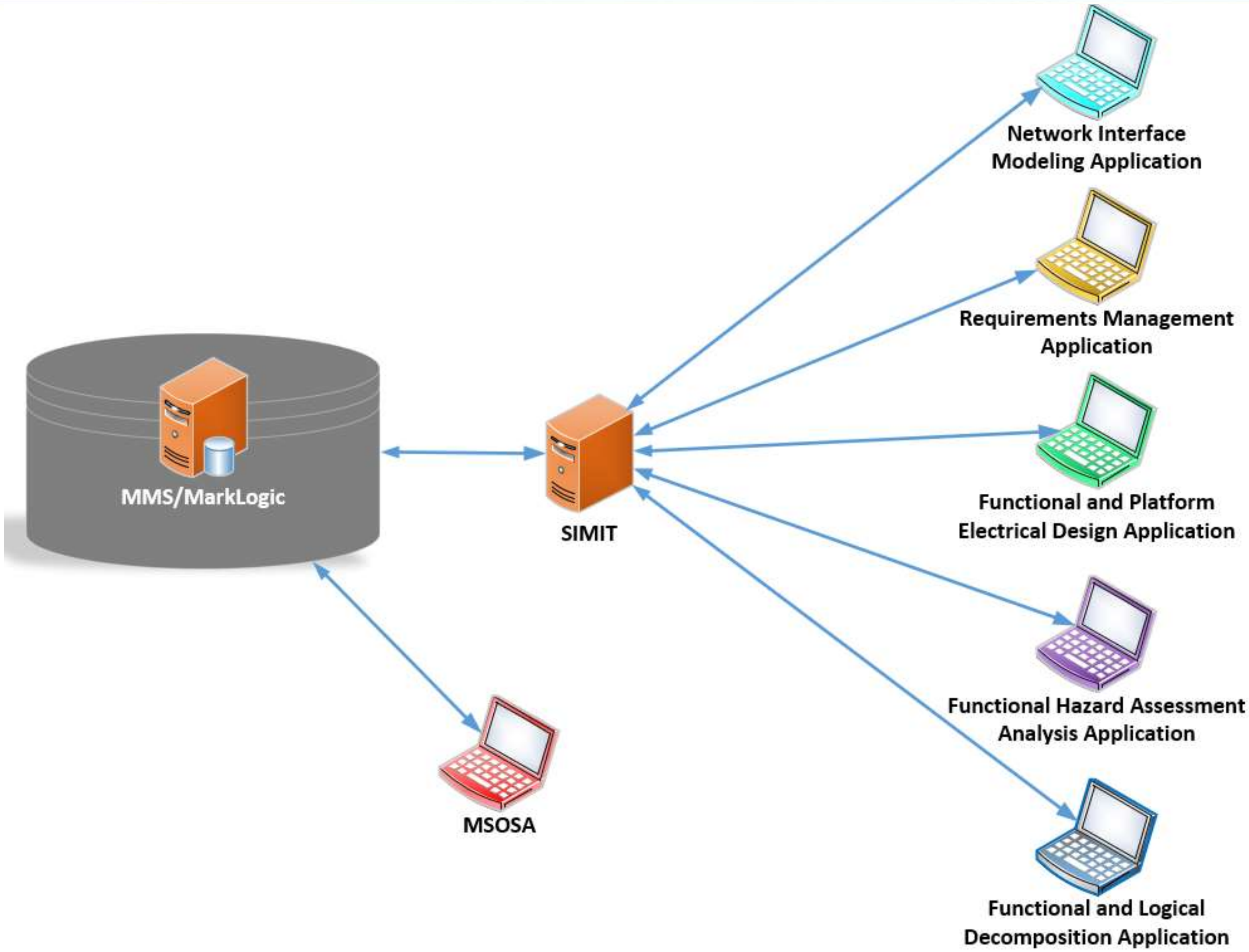
MDK= Model Development Kit Plugin

MMS= Model Management System

Open MBEE= Open Model Based Engineering Environment

SIMIT= Synchronization and Integration in MMS for Interoperability of Tools

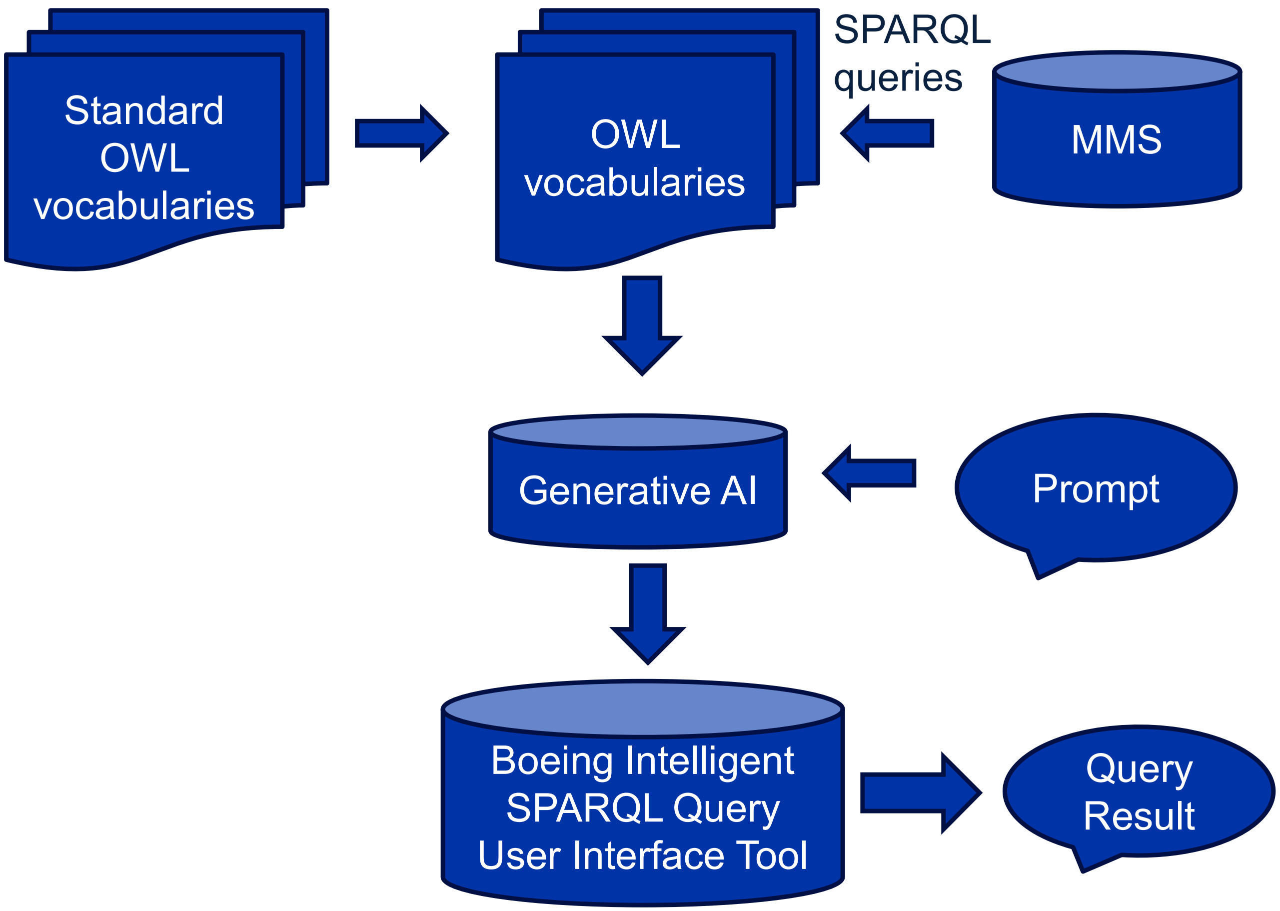



Network Interface Modeling Application
Requirements Management Application
MMS/MarkLogic
SIMIT
Functional and Platform Electrical Design Application
Functional Hazard Assessment Analysis Application
MSOSA
Functional and Logical Decomposition Application

The Issue/Idea

- Engineers with backgrounds in Safety/Systems/Electrical backgrounds utilize our tool suite to integrate their models into a digital thread.
- The SPARQL language is niche
  - The engineers who use our tool suites generally have no experience with semantic query languages
  - Relatively easy to pick up and start working with
  - Difficult to contextualize and apply in complex contexts

Can we leverage Generative AI to dramatically shorten the SPARQL learning curve and unlock powerful querying capabilities for engineers?


Standard OWL vocabularies
OWL vocabularies
SPARQL queries
MMS
MSOSA profiles
Generative AI
Prompt
Boeing Intelligent SPARQL Query User Interface Tool
Query Result

Vocabularies

```
###  https://www.omg.org/spec/SysML/20181001/SysML.xmi#Allocate
:Allocate rdf:type owl:Class ;
        rdfs:subClassOf :DirectedRelationshipPropertyPath ;
        rdfs:comment "Allocate is a dependency based on UML::abstraction. It is a mechanism for
        associating elements of different types, or in different hierarchies, at an abstract level. Allocate
        is used for assessing user model consistency and directing future design activity. It is
        expected that an allocate relationship between model elements is a precursor to a more
        concrete relationship between the elements, their properties, operations, attributes, or sub-
        classes." .

###  https://www.openmbee.org/ontologies/mbse#associates
:associates rdf:type owl:ObjectProperty ;
        rdfs:subPropertyOf owl:topObjectProperty ;
        owl:inverseOf :isDescribedBy ;
        rdfs:comment "Identifies additional information about a relationship. Implemented as an
        Association Block in SysML" .
```

Experimentation Methodology

- Our team maintains a SPARQL “Cookbook” that contains descriptions and examples of 8 simple queries and 7 complex queries that users can plug in to BISQUIT to extract information from their MSOSA models.
- For this experiment we used the descriptions provided for these queries as the input to the Generative AI, to rate the output for correctness
- Scoring: 1 point if got it right on the first try, 0.5 if one suggestion was needed, 0.25 if two suggestions were needed, 0 if more than two…

Simple queries	More complex queries
1.1 Query for ids and names of all elements of a specific metaclass (e.g. Package)	2.1 Querying for elements of all types of requirements
1.2 Query for ids and names of all elements with a specific applied stereotype (e.g. Block)	2.2 Querying for tag values (e.g. the requirement Text tag)
1.3 Counting elements with a specific applied stereotype	2.3 Querying for custom stereotype types and custom tag values
1.4 Counting non-empty packages in the model	2.4 Querying for parts recursively
1.5 Querying for parts of all blocks in the model	2.5 Querying for owned and inherited ports
1.6 Querying for parts of a specific block	2.6 Querying for owned and inherited parts
1.7 Querying for elements satisfying a requirement	
1.8 Querying for requirements refining a requirement	

1 Simple queries

1.1 Query for ids and names of all elements of a specific metaclass (e.g. Package)

```
select ?id ?name
where {
      ?id   rdf:type   uml:Package ;
            rdfs:label ?name .
}
```

```
assistant: Here is a SPARQL query that retrieves the IDs and names of all elements of type
`Package` from the provided vocabularies for SysML, UML, and MMS:

Output:
PREFIX sysml: <https://www.omg.org/spec/SysML/20181001/SysML.xmi#>
PREFIX uml: <https://www.omg.org/spec/UML/20161101/UML.xmi#>
PREFIX mms: [redacted]/ontologies/mms#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?id ?name
WHERE {
    ?package a uml:Package .
    ?package mms:id ?id .
    ?package mms:name ?name .
}
```

Output for Cookbook Query

Results

Table Response

Showing 1 to 50 of 205 entries

	id	name
1	_12_0EAPbeta_be00301_1156851270584_552173_1	MD Customization for SysML
2	_12_0EAPbeta_be00301_1157529392394_202602_1	PrimitiveTypes
3	_12_0EAPbeta_be00301_1161605926426_880859_462	diagrams
4	_12_0EAPbeta_be00301_1161869727438_806671_657	customizations
5	_12_0EAPbeta_f720368_1161678076339_464983_95	Rule Validation Suite
6	_12_0_8f90291_1163446670265_941740_378	UML correctness constraints
7	_12_0_8f90291_1163453727562_955061_538	UML completeness constraints
8	_15_5EAPbeta1_8ca0285_1207222354728_434471_815	Shape Ownership
9	_15_5EAPbeta1_8ca0285_1207222370039_532769_817	Parameters Synchronization
10	_15_5EAPbeta1_8ca0285_1207309180033_368978_408	Composition Integrity

Output for Generative AI Query

Results

Table Response

Showing 1 to 50 of 205 entries

	id	name
1	_12_0EAPbeta_be00301_1156851270584_552173_1	MD Customization for SysML
2	_12_0EAPbeta_be00301_1157529392394_202602_1	PrimitiveTypes
3	_12_0EAPbeta_be00301_1161605926426_880859_462	diagrams
4	_12_0EAPbeta_be00301_1161869727438_806671_657	customizations
5	_12_0EAPbeta_f720368_1161678076339_464983_95	Rule Validation Suite
6	_12_0_8f90291_1163446670265_941740_378	UML correctness constraints
7	_12_0_8f90291_1163453727562_955061_538	UML completeness constraints
8	_15_5EAPbeta1_8ca0285_1207222354728_434471_815	Shape Ownership
9	_15_5EAPbeta1_8ca0285_1207222370039_532769_817	Parameters Synchronization
10	_15_5EAPbeta1_8ca0285_1207309180033_368978_408	Composition Integrity

2.4 Querying for parts recursively

```
select ?blockName ?partName
where {
    ?blockId    rdf:type                   sysml:Block ;
                rdfs:label                 ?blockName ;
                mbse:hasPart+/rdfs:label   ?partName .
}
```

```
```sparql
PREFIX sysml: <https://www.omg.org/spec/SysML/20181001/SysML.xmi#>
PREFIX uml: <https://www.omg.org/spec/UML/20161101/UML.xmi#>
PREFIX mms: <[redacted]ontologies/mms#>

SELECT ?part ?partName
WHERE {
    {
        ?block a sysml:Block .
        ?block mms:id "_18_2beta_8c4028a_1433923068797_23346_13742" .
        ?block mms:ownsId ?part .
    }
    UNION
    {
        ?part a sysml:PartProperty .
        ?part mms:ownsId ?subPart .
    }
    ?part mms:name ?partName .
}
```
```

````
```sparql
PREFIX sysml: <https://www.omg.org/spec/SysML/20181001/SysML.xmi#>
PREFIX uml: <https://www.omg.org/spec/UML/20161101/UML.xmi#>
PREFIX mms: [redacted]ontologies/mms#>

SELECT ?port ?portName
WHERE {
    # Get the block with the specified ID
    ?block a sysml:Block .
    ?block mms:id "_18_2beta_8c4028a_1438610657636_56702_15142" .

    # Get owned ports
    ?block mms:ownsId ?port .
    ?port a uml:Port .
    ?port mms:name ?portName .

    # Get inherited ports (if applicable)
    OPTIONAL {
        ?inheritedPort a uml:Port .
        ?inheritedPort uml:redefinedPort ?port .
        ?inheritedPort mms:name ?portName .
    }
}
```
````

Results

- Score:
  Simple queries: 5.75/8 (72%)
  More complex queries: 4.5/7 (64%)

- The caveat here is that the Generative AI did well on the complex queries, in large part because it had learned from a lot of the mistakes made, and feedback received, when generating the simple queries.

Experiment Observations

- The Generative AI did a good job of using the comments provided in the vocabulary. It correctly used the proper predicates based on matching the prompts provided to the predicate descriptions (e.g. flowsTo).

- At first, it did not understand what the concept of inherited properties were. We explained that a property is inherited when I have a superclass that owns a property, then the classes that are generalized from it inherit the property. In the future, when I referred to inherited ports, it already understood what inherited meant in this context.

- The Generative AI often tries to replicate the complexity of the SQL language, instead of utilizing the shortcuts that SPARQL enables (such as property paths).

- It was also prone to hallucinating predicates that did not exist in the provided vocabularies instead of finding the closest match.

Conclusion

- What is generative AI useful for in this context?
  - Learning SPARQL
    - Learn by using something rather than reading SPARQL documentation
  - Speed up writing long queries (not necessarily complex ones)
  - Consuming large amounts of specialized vocabularies
  - Improving the efficiency of written queries
  - Writing simple and complex queries when detailed context is provided, and with manual review for validity.

- What is it not useful for?
  - Providing SPARQL queries to people who don't want to learn SPARQL.
  - If you are not qualified to find the answer, you're not qualified to assess its correctness.
    - I can calculate what 432*3109*614 is by hand, but I can also use a calculator to do it. If the calculator responds with something that is obviously wrong (e.g. a negative number, a number that is too high or too low, a decimal …), I will have the intuition that it's incorrect.

Future Experiment

- The Generative AI supports a Prompt Studio, which allows you to save a template prompt that it can use every time you start a new conversation.
  - It accepts:
    - Attachments (all text files, .docx, .pptx, .pdf, .md)
    - Pre-set Sample Questions (possibly to be used for calibration?)
    - A system directive that sets the context/guidelines for the model's behavior

- We can experiment with the Prompt Studio to generate a prompt that can be provided to the Generative AI every time we wish to discuss SPARQL queries with it. This prompt could be customized so that the results already consider previous lessons learned and could be improved as we continue to add context to it.
- We can then measure how much more effective the Generative AI is at generating the example queries provided in the Cookbook, when the template prompt is provided.