Design and Development of OWL-DL Ontologies for Satellite Launch Vehicle Missions

26-Feb-2020

S S Uma Sankari
Quality Assurance Mission Division
VSSC/ ISRO

Outline

- **Introduction**
  - **Knowledge Engineering and Semantic technology**
  - **Ontology and its details**
  - **Launch vehicle and Mission simulation**
- **Design and Development of Ontologies**
  - **Design Guidelines and Steps involved**
  - **Hands-on experience : Launch vehicle domain ontology**
- **Key points from experience**
- **Summary**

Introduction

- **Knowledge Engineering: a subfield of Artificial Intelligence**
  - Build knowledge-based systems
  - Tries to mimic **cognitive abilities** of human beings
  - Knowledge Modelling & Reasoning
- **Semantics Technology**
  - Categorizing and processing data based on **meaning**
- **Semantics structures: Ontology**
  - Description of knowledge in
    - Machine-processable syntax
    - Formally defined meaning
  - Intelligent agent: interpret the context automatically
- **Feasibility**
  - Modelling the knowledge of a multidisciplinary system
    - **Knowledge modelling: using ontology & semantic rules**
    - **Reasoning: - Assertive & Inferred**

Aids in automating & enhancing quality of time consuming, expert-dependent processes

Ontology- an introduction

- Domain of interest - can be modeled as
  - a set of concepts & relationships among them

Element	Details	Example
Instances (individual)	most basic component; represent fundamental objects in the domain	ram, sharma
Classes (concept)	description of a set of elements in a domain; a set of instances	PhDStudent, Student, Professor
Properties (roles)	represent binary relations; stand-alone elements	hadGuide, isStudentOf
Axioms	expressions; state the required constrains to be satisfied by other elements in ontology	1. PhD student is a Student who has guide 2. ram is PhD student 1& 2 : assertive → ram has guide : Inferred

- Ontology representation using – RDF, RDFS and OWL

Ontology – possible applications

Ontology is used

1. as a common vocabulary for communication among distributed agents
2. as a conceptual schema of a relational DB
3. as a backbone information for a user of KB
4. for answering competence questions
5. for standardization of terminology, meaning of concepts and tasks
6. for semantic analysis
7. for reusing knowledge of a KB
8. for reorganizing a KB

Global Scenario

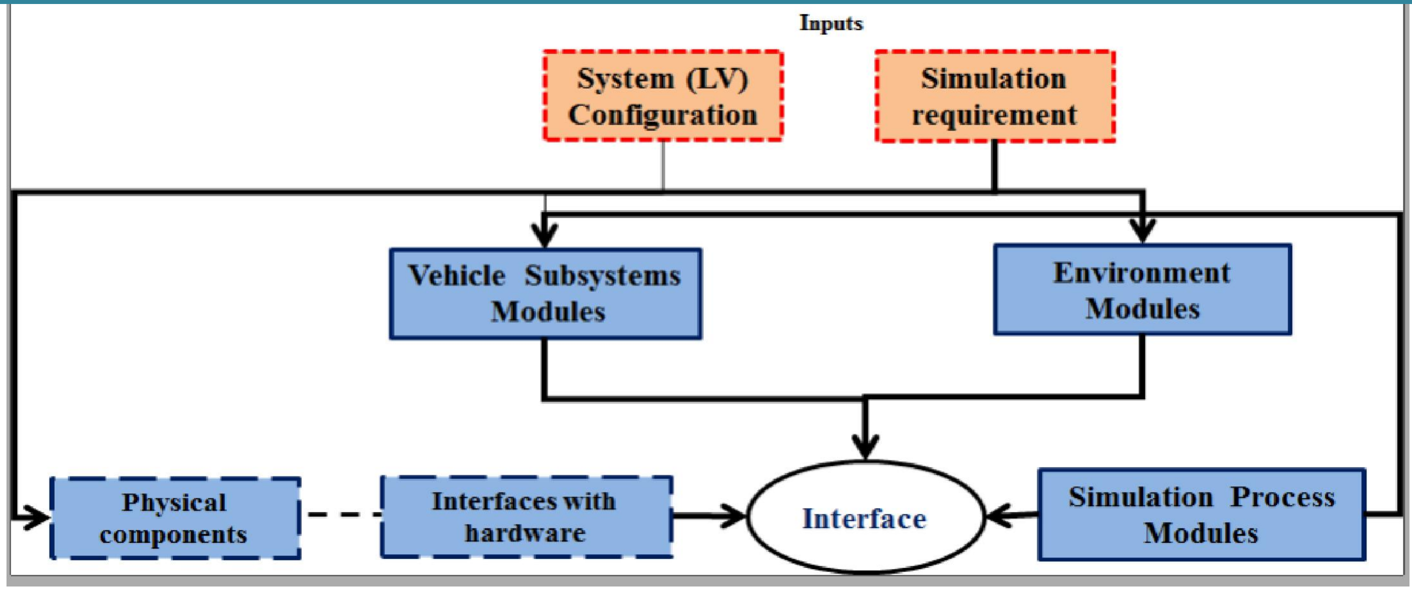
- **Applications of Ontology**
  - Widely applied in various domains –Medical, biology, Education, Finance etc
    - Healthcare: as workflow assistance- from keeping medical records to conducting surgery
    - Finance : for pattern prediction and decision making
  - **Limited studies in Aerospace and Atomic energy**
- **Ontology in Aerospace domain**
  - Accident investigation ontology
  - NASA taxonomy
  - Semantic organizer

Demonstrating potential of knowledge engineering application

Ontology enabled automated generation of Mission simulation software

Mission Simulation Software (MSS)

- **Mission simulations -** assess the behavior and performance by solving numerical models/algs
- Mission simulations- carried out using **MSS**



- Any error in the software may **cause** clearance of
  - Unworthy LV subsystem/ system; Erroneous flight software; Incorrect flight data
- **Effect** – Lead to loss of/ destruction of /damage to
  - Vital Equipment; Human Lives; Environment

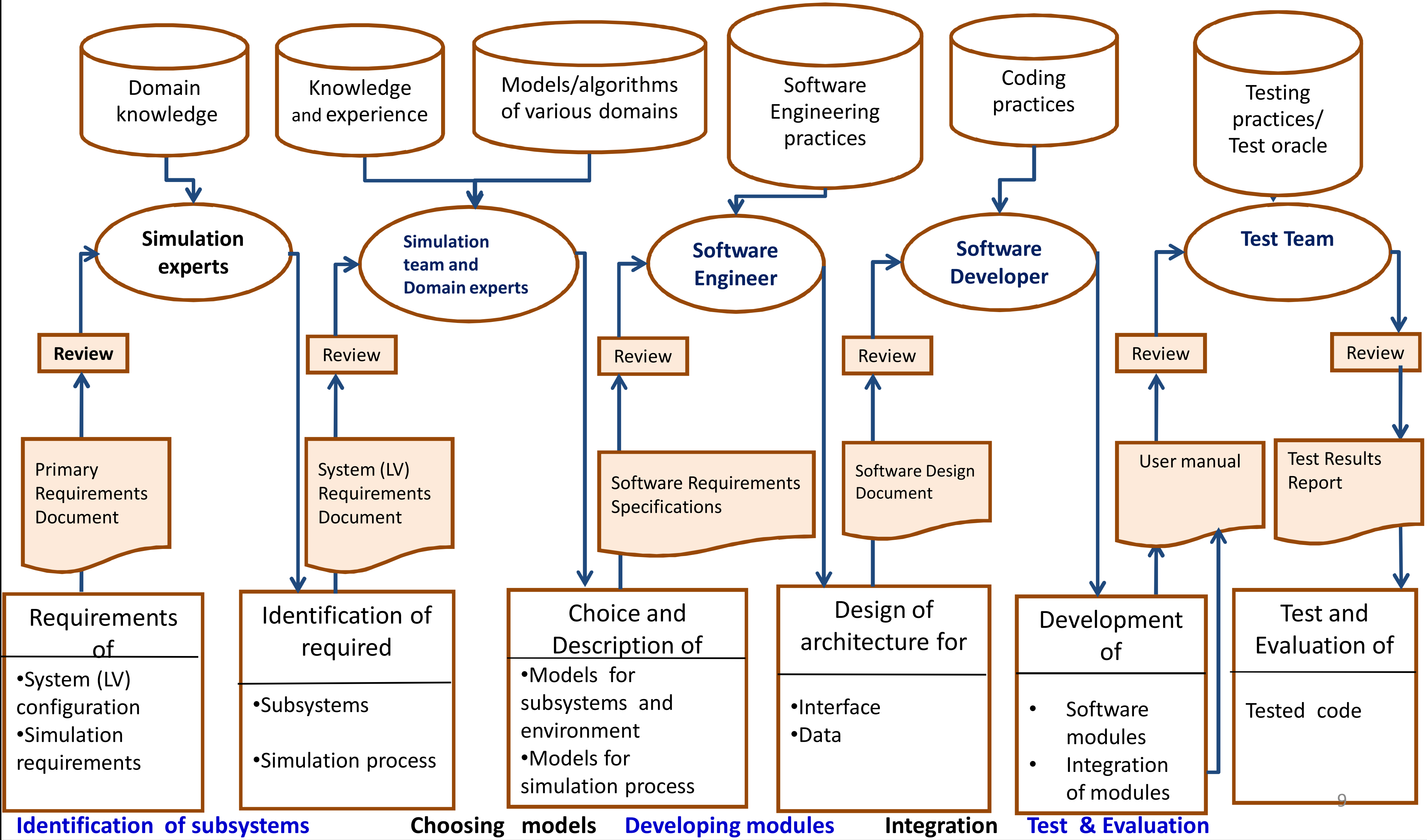
Error-free software: must be developed with high level of quality and Reliability

Importance of Mission Simulation Software (MSS)

- Simulations in aerospace industry -throughout the life cycle phases of LV
  - by various domain-experts
  - at different levels
  - for various objectives
- **Conceptualization phase to vehicle-in-loop simulation**

- **Development of simulation software**
  - a new mission requirement /change in requirement
  - a new vehicle configuration /change in configuration
  - studies with alternate designs of subsystems models/modules
- **Every group – develops : own customized software**
- **Each development/modification**
  - Goes through - life-cycle process of software development.

Mission Simulation Software Development – a Typical Life-cycle process



Addressing the Limitations in Existing practice

Limitations	Reason	Solution
Dependency on Experts	Identification of subsystems, choice on models	Ontology and Semantic Rules
Consumes time and effort	Sequential completion of development activities – human involvement	
Non Zero probability of Error	Human error	Ontology enabled Automation
Duplication of efforts	Similar efforts spent by various groups of experts	Archiving validated modules and Reuse

Demonstrating potential of Knowledge Engineering application

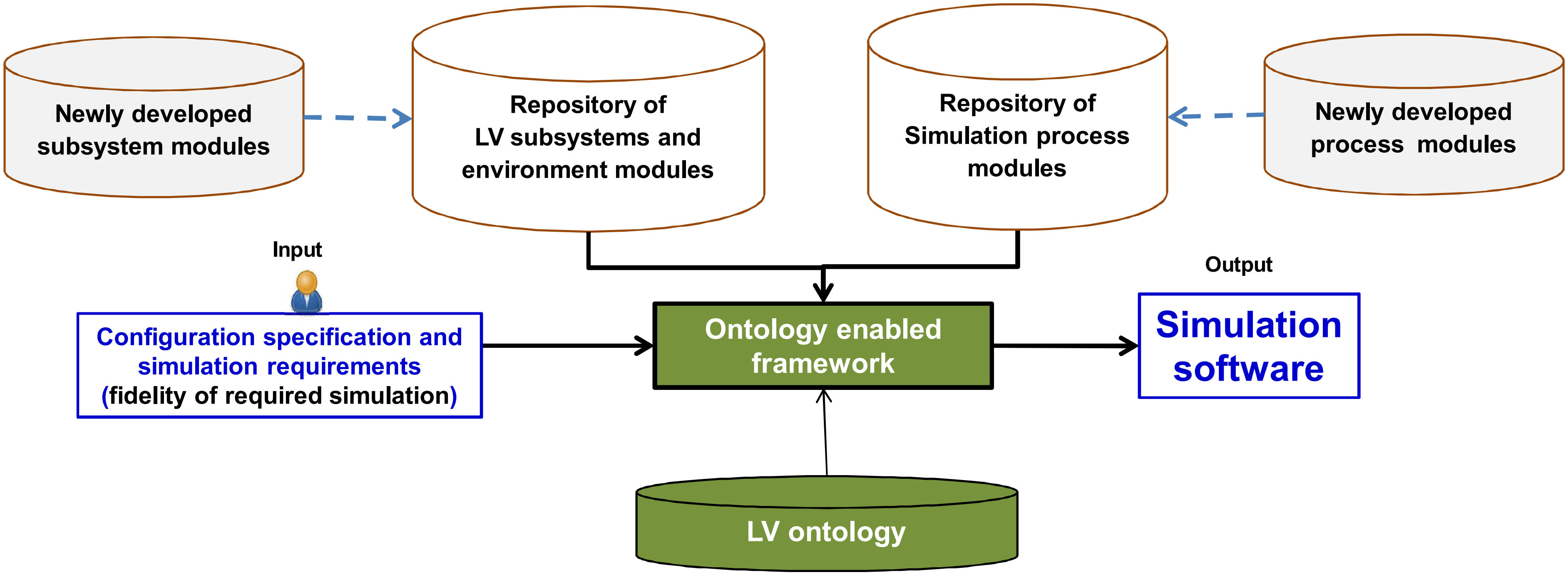
Need

Experts–dependency and development time are to be reduced without compromising the quality

Proposal

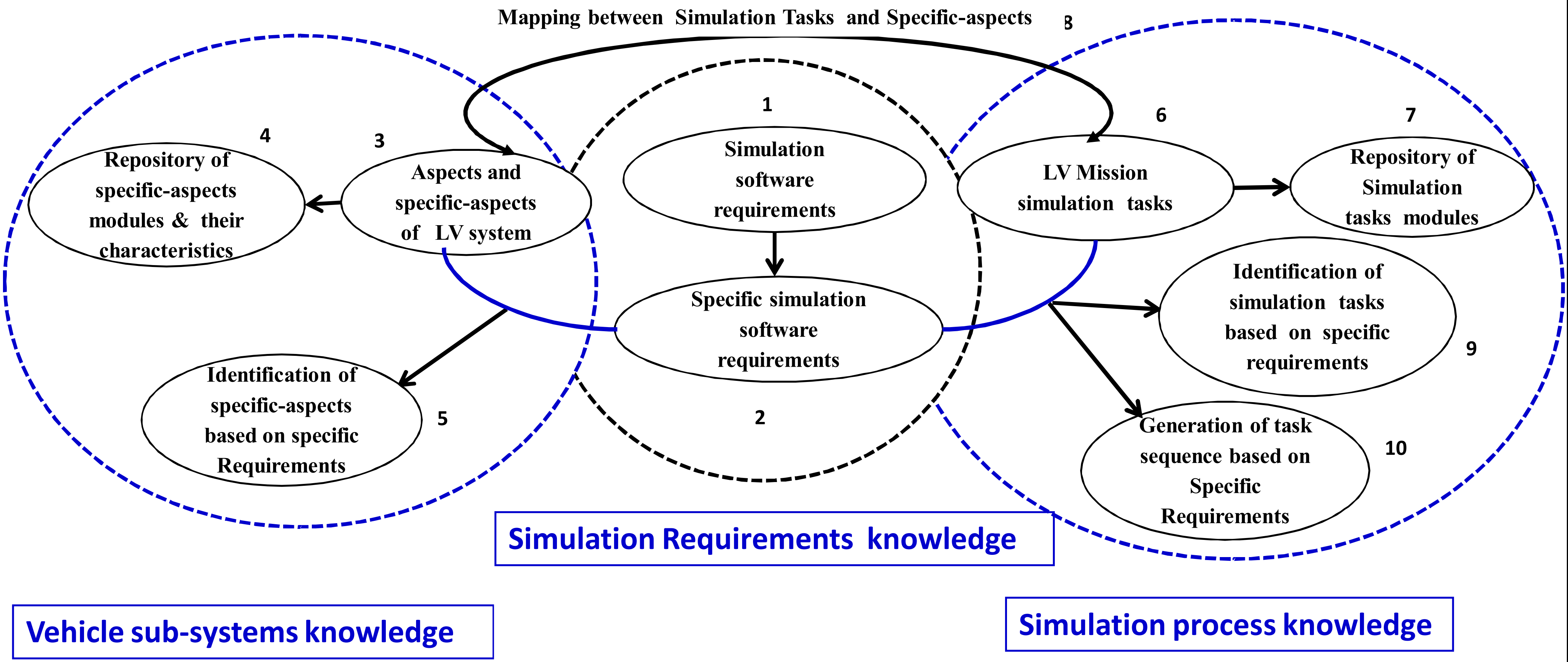
- **Model the multidisciplinary knowledge associated with MSS development**
- **Envisage a strategy to use it to enhance the quality of existing process**

Envisaged Strategy



1. Design and Development of Ontology
2. Development of Ontology enabled framework

Mission Simulation Software Knowledge(MSSK)



MSSK is a hybrid knowledge:

1,2,3,4,6 &7 : Assertive knowledge: Can be formalized as ontologies

5, 8 , 9 &10 : Inferred Knowledge: as Semantic Rules

Ontology Design : Guidelines

- The design is to be useful for a wide range of applications
- Design should be amenable for extension : for a wider scope
- Design: Flexible, easy-to use and maintain

Ontology Development Process

1. **Consolidation** of knowledge from various experts
2. **Conceptualization** - Weighing various options
3. **Formalization**
    - Using OWL and different variants of DL
    - Addressing OWL assumptions
4. **Implementation**
5. **Maintenance**

Description Logics (DL)

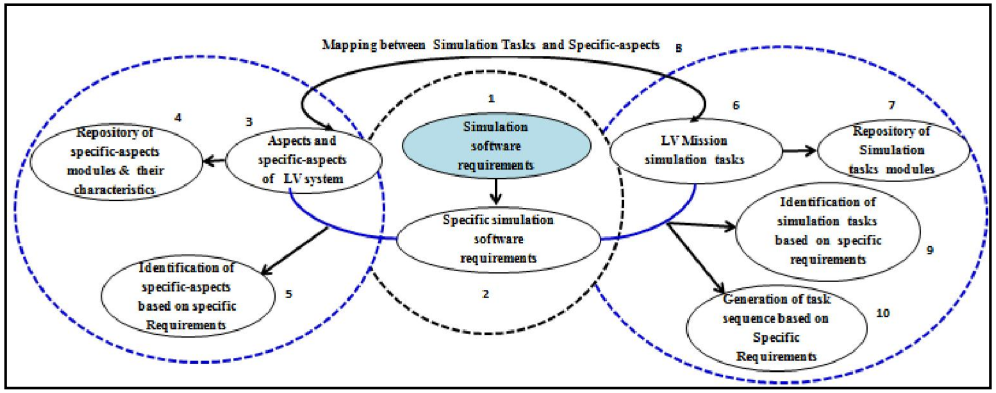
- Formal semantics of OWL is based on DL
- Computational <u>completeness</u> and retaining <u>decidability</u>
- Basic elements: concepts, roles and individuals
- Complex concept from basic concepts

Constructor Name	Symbol	Syntax
Top, Bottom, Negation, Conjunction, Disjunction, Existential restriction, Value restrictions ($\mathcal{ALC}$ extended with transitive roles)	$\mathcal{S}$	⊤, ⊥, ¬C, C ⊓ D, C ⊔ D, ∃R.C, ∀R.C
Role Hierarchy	$\mathcal{H}$	R ⊑ S
Nominal(Enumerated classes of object value restrictions)	$\mathcal{O}$	{a}
Inverse Role	$\mathcal{I}$	R^-
Qualified number restrictions (have filler value other than Top concept)	$\mathcal{Q}$	≤n R.C
Concrete domains	$(\mathcal{D})$	data values/ data type

DL : Varying levels of expressiveness of semantics

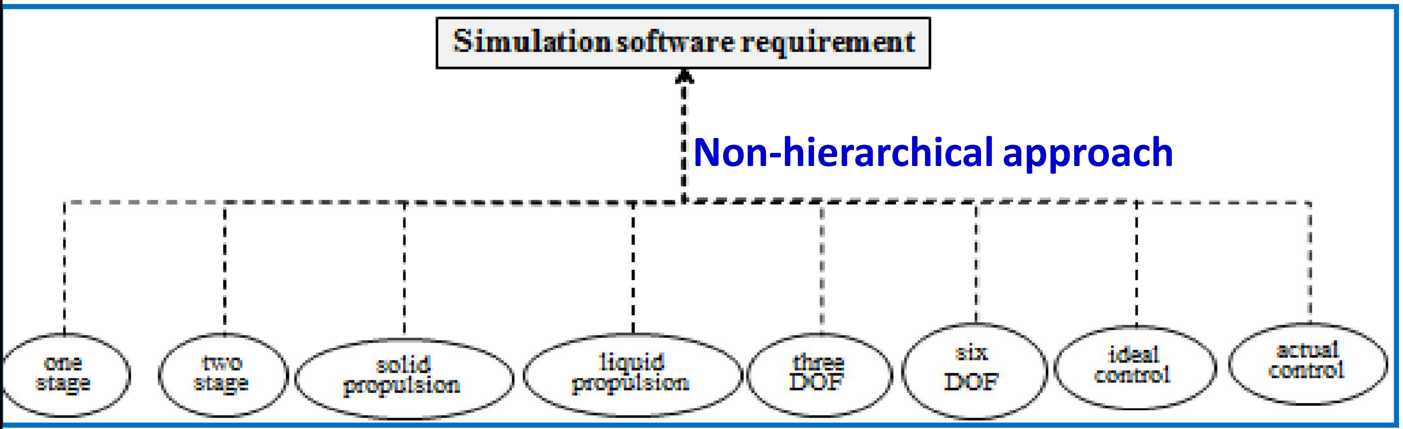
Simulation Requirements Knowledge

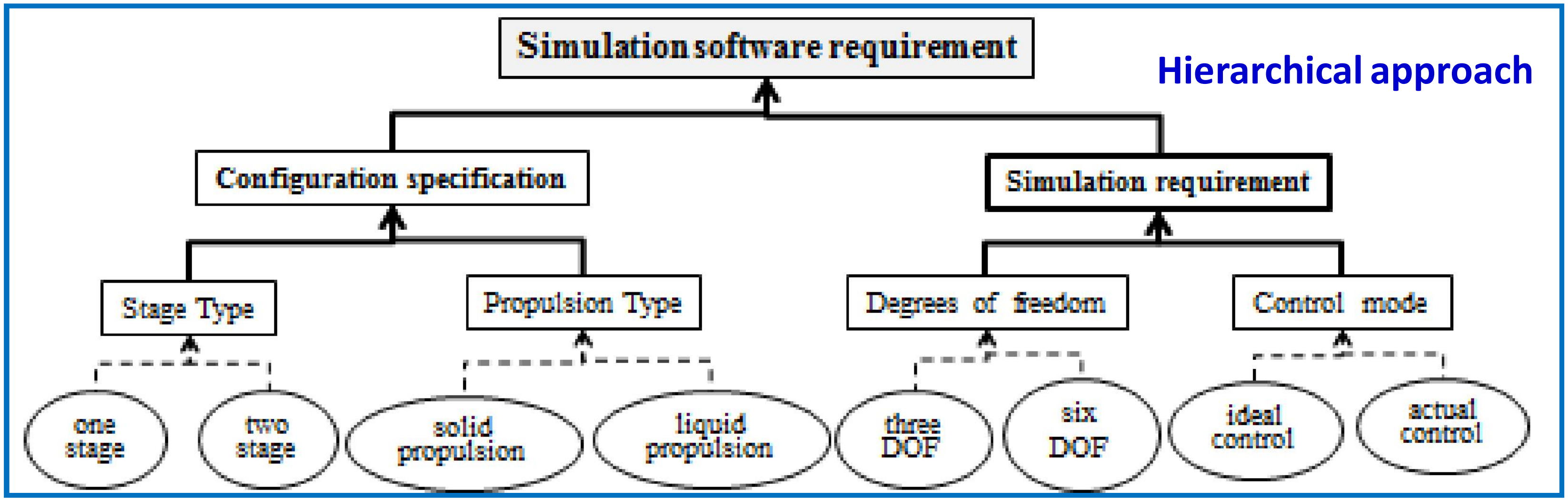
- **Configuration Specification : Sub-concept**
  - Describes the types of stage and propulsion
  - Depend on the objective of the simulation study
  - **Stage Type :** one stage or two stage
  - **Propulsion Type :** solid fuel or liquid fuel
- **Simulation Requirement**
  - **Degrees Of Freedom(DOF) :** 3DOF or 6 DOF
  - **Control Mode:** ideal or actual
- **Combination of concepts**
  - Certain combinations are incompatible



S.no	Stage type	Propulsion type	DOF	Control mode
1.	one stage	solid	3DOF	ideal
2.	**one stage**	**solid**	**3DOF**	**actual**
3.	one stage	solid	6DOF	ideal
4.	one stage	solid	6DOF	actual
5.	one stage	liquid	3DOF	ideal
6.	**one stage**	**liquid**	**3DOF**	**actual**
7.	one stage	liquid	6DOF	ideal
8.	one stage	liquid	6DOF	actual

Conceptualization of Simulation Requirements Knowledge



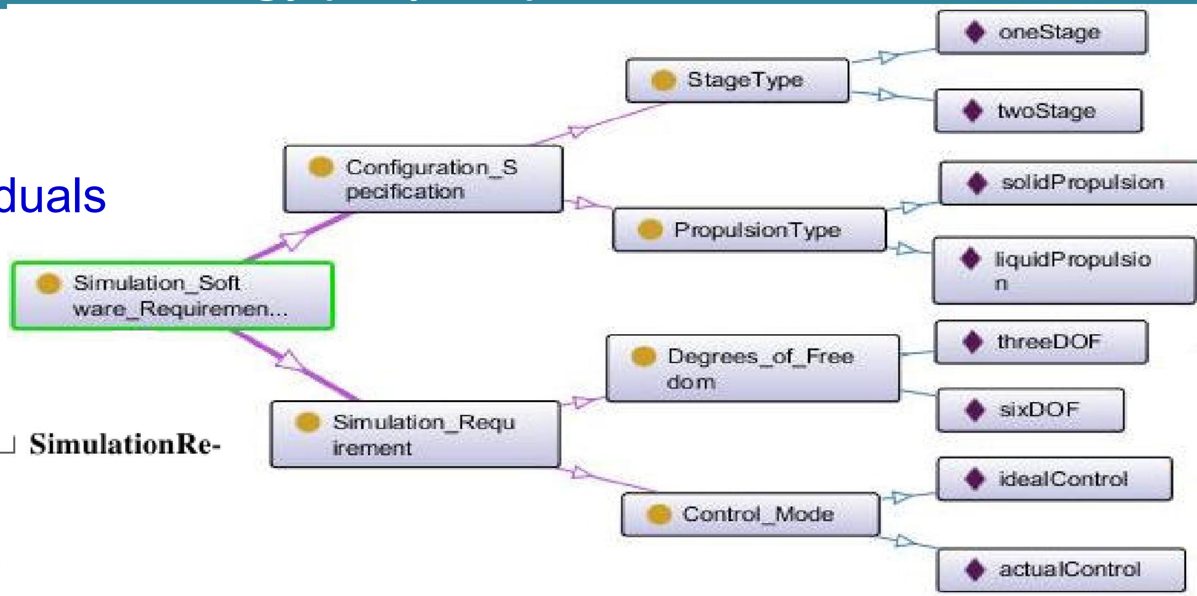


Hierarchical approach is chosen for formalization

1. Describes the complete domain concept
2. Amenable for other applications

Requirements Ontology (Reqonto)

- Major concepts - as Classes
- Specific values of concepts- as individuals



$\mathbf{SimulationSoftwareRequirement} \equiv \mathbf{ConfigurationSpecification} \sqcup \mathbf{SimulationRequirement}$

$\mathbf{ConfigurationSpecification} \sqsubseteq \mathbf{SimulationSoftwareRequirement}$

$\mathbf{SimulationRequirement} \sqsubseteq \mathbf{SimulationSoftwareRequirement}$

$\mathbf{ConfigurationSpecification} \equiv \mathbf{PropulsionType} \sqcup \mathbf{StageType}$

$\mathbf{StageType} \sqsubseteq \mathbf{ConfigurationSpecification}$

$\mathbf{StageType(oneStage)}$

$\mathbf{StageType(twoStage)}$

$\mathbf{oneStage} \neq \mathbf{twoStage}$

$\mathbf{PropulsionType} \sqsubseteq \mathbf{ConfigurationSpecification}$

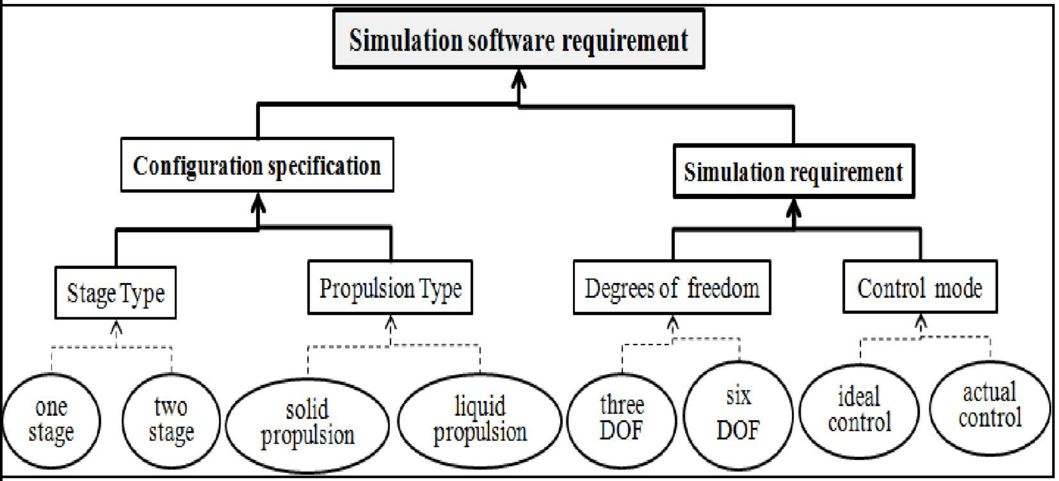
$\mathbf{PropulsionType(solidPropulsion)}$

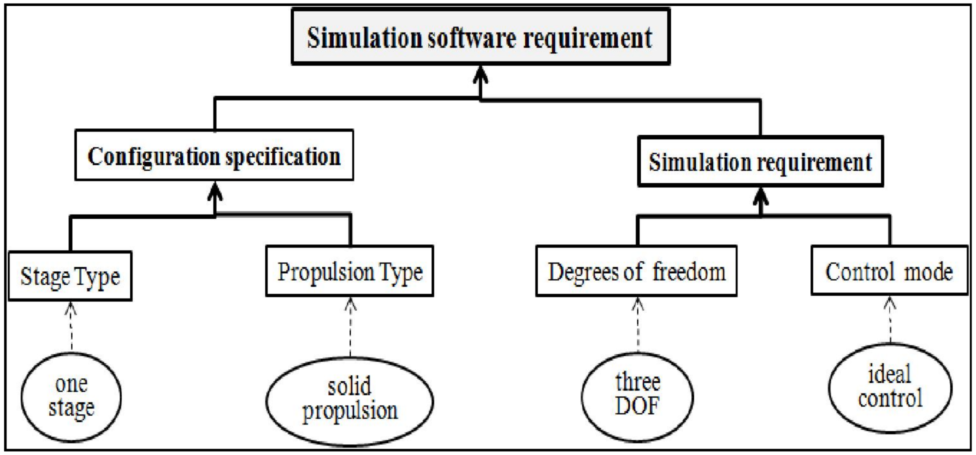
$\mathbf{PropulsionType(liquidPropulsion)}$

- No Unique Name Assumption: ***owl:allDifferent***
- Disjoint subclasses : no common individual

Ontology Name	No of Logical axioms	No of Individuals	No of Classes	OWL-DL Variant
ReqOnto	**44**	**9**	**7**	$\mathcal{ALC}$

Conceptualization of Specific Requirements Knowledge



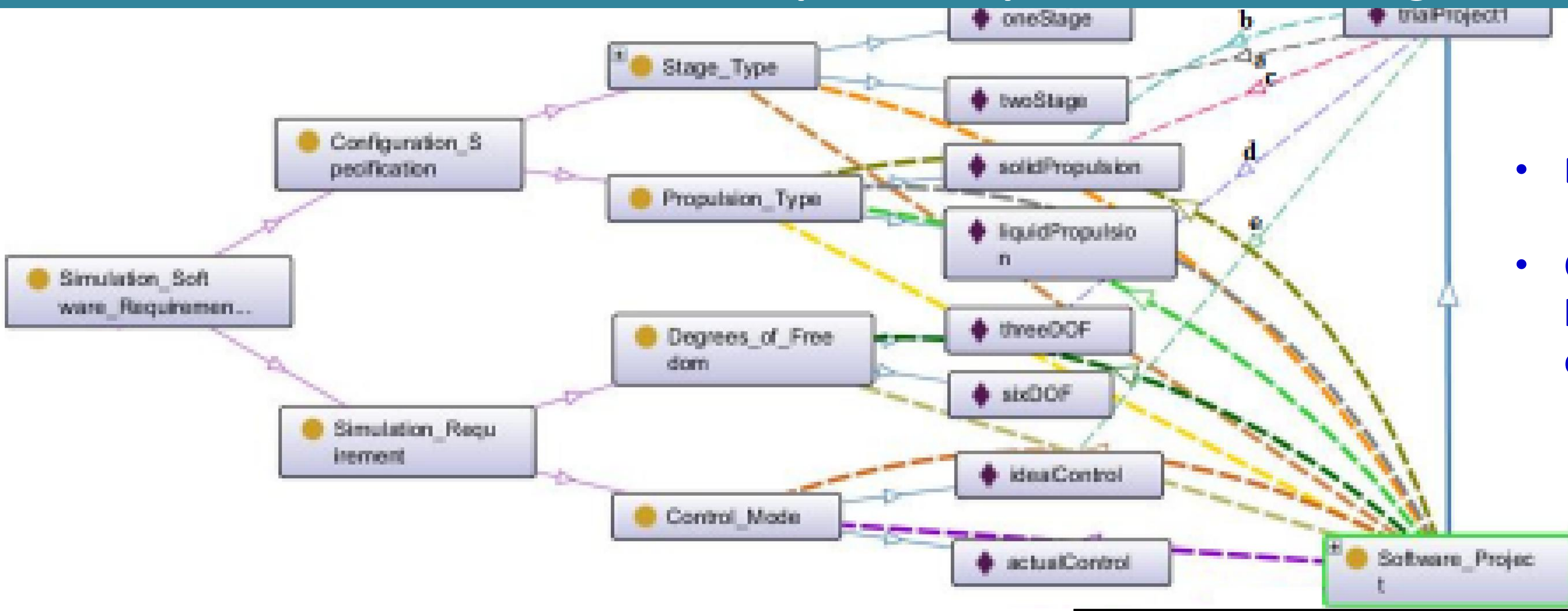


Specific requirements- Deduced from Generic requirement

Property restrictions

- ✓ **Universal restriction**
- ✓ **Existential restriction**
- ✓ **Number restriction**

Formalization of Specific Requirements Knowledge

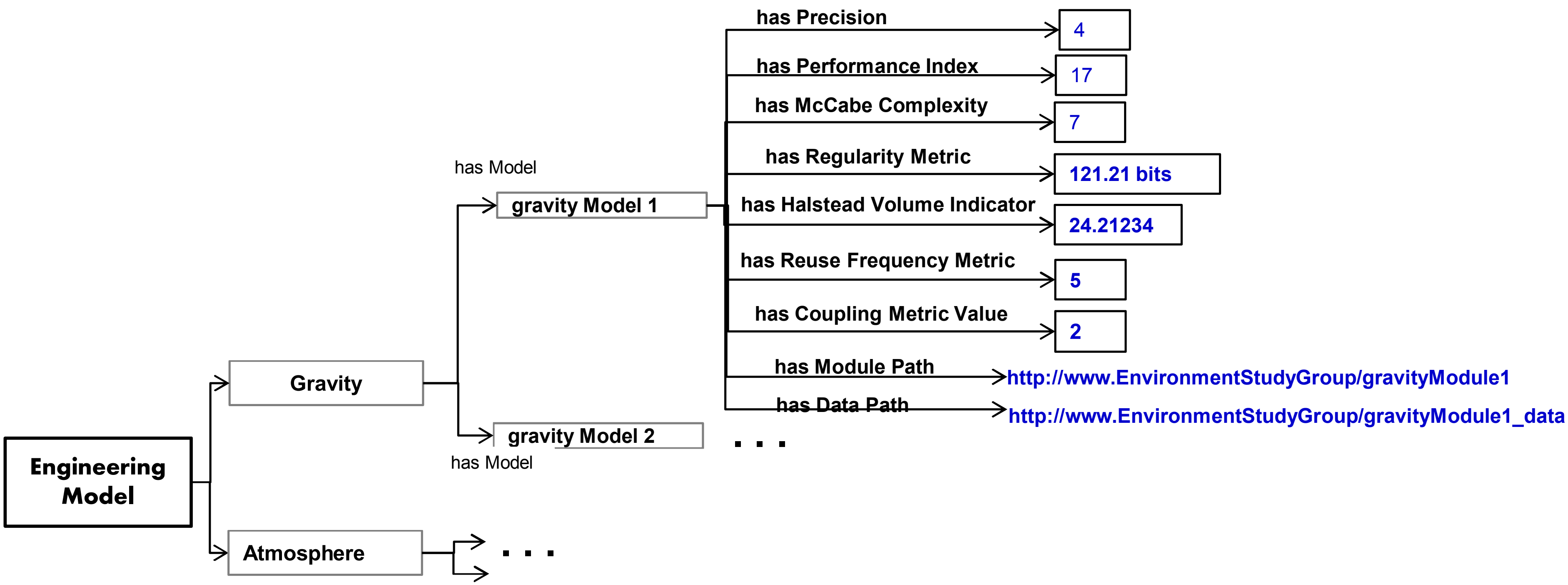


- **Domain & Range**
- **Compatibility between derived classes**

OWL-DL variant: $\mathcal{SHOIQ}$

$$\begin{aligned}
&Software_Project \sqsubseteq \forall\, hasStage1Type.Propulsion_Type\\
&Software_Project \sqsubseteq \exists\, hasStage1Type.Propulsion_Type\\
&Software_Project \sqsubseteq\ \leq 1\ hasStage1Type\\
&\\
&Software_Project \sqsubseteq \forall\, hasStage2Type.Propulsion_Type\\
&Software_Project \sqsubseteq \exists\, hasStage2Type.Propulsion_Type\\
&Software_Project \sqsubseteq\ \leq 1\ hasStage2Type\\
&\\
&Software_Project \sqsubseteq \forall\, hasNumberOfDOF.Degrees_of_Freedom\\
&Software_Project \sqsubseteq \exists\, hasNumberOfDOF.Degrees_of_Freedom\\
&Software_Project \sqsubseteq\ \leq 1\ hasNumberOfDOF\\
&\\
&Software_Project \sqsubseteq \forall\, hasControlMode.Control_Mode\\
&Software_Project \sqsubseteq \exists\, hasControlMode.Control_Mode\\
&Software_Project \sqsubseteq\ \leq 1\ hasControlMode
\end{aligned}$$

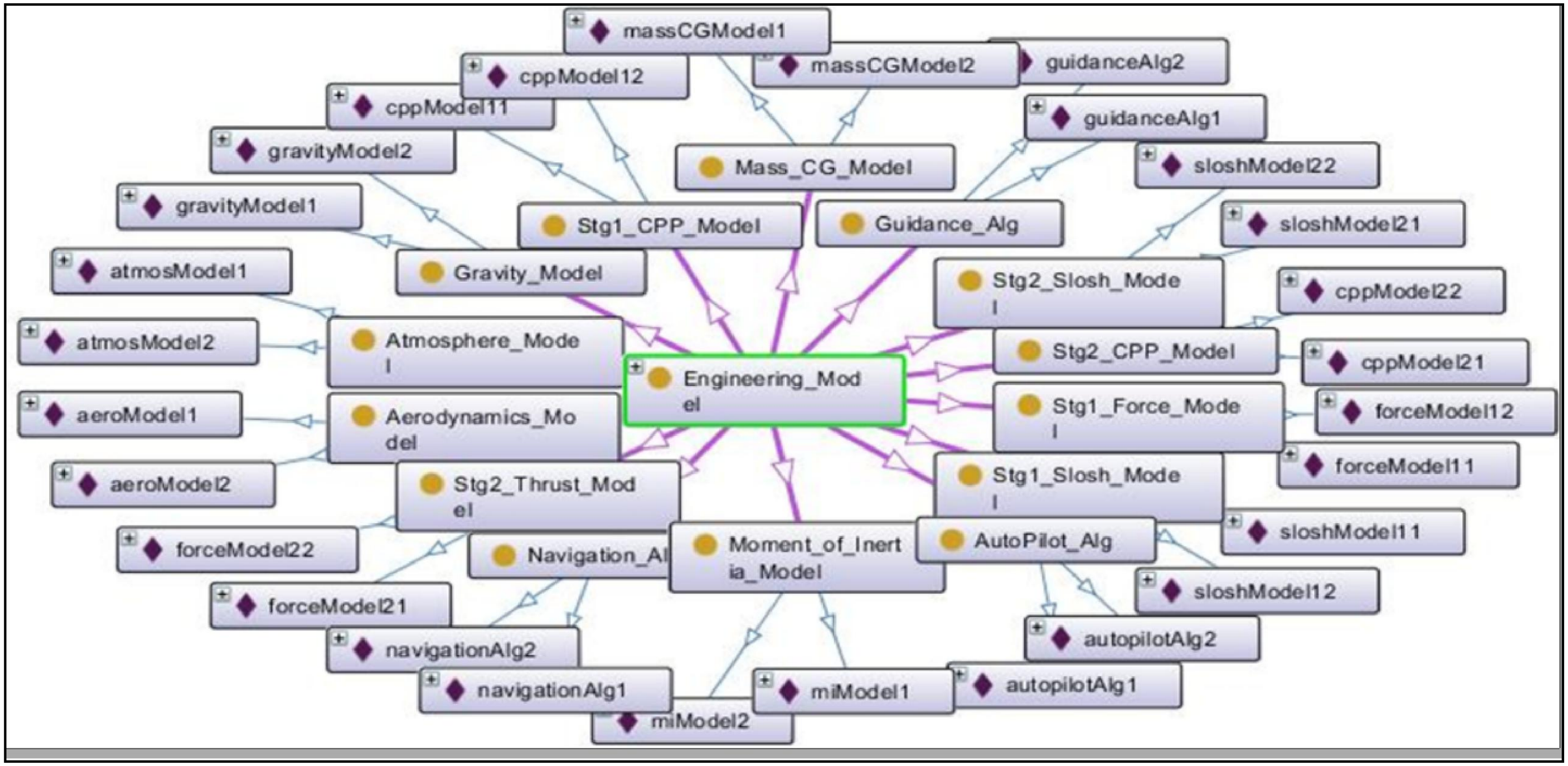
Conceptualization of Engineering models



Fidelity of Engineering model

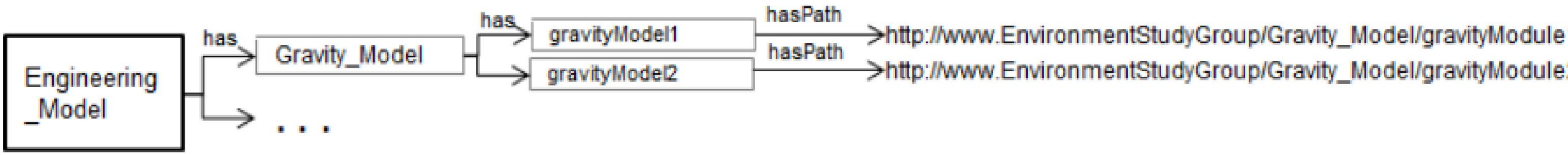
```
hasReuseFrequencyMetric(gravityModel1,5)
hasPrecision(gravityModel1,4)
hasPerformanceIndex(gravityModel1,12)
hasMaCabeComplexity(gravityModel1,7)
hasRegularityMetric(gravityModel1,121.21bits)
hasCouplingMetricValue(gravityModel1,2)
halsteadVolumeIndicator(gravityModel1, 12.2134)
```

Formalization of Engineering Models Knowledge

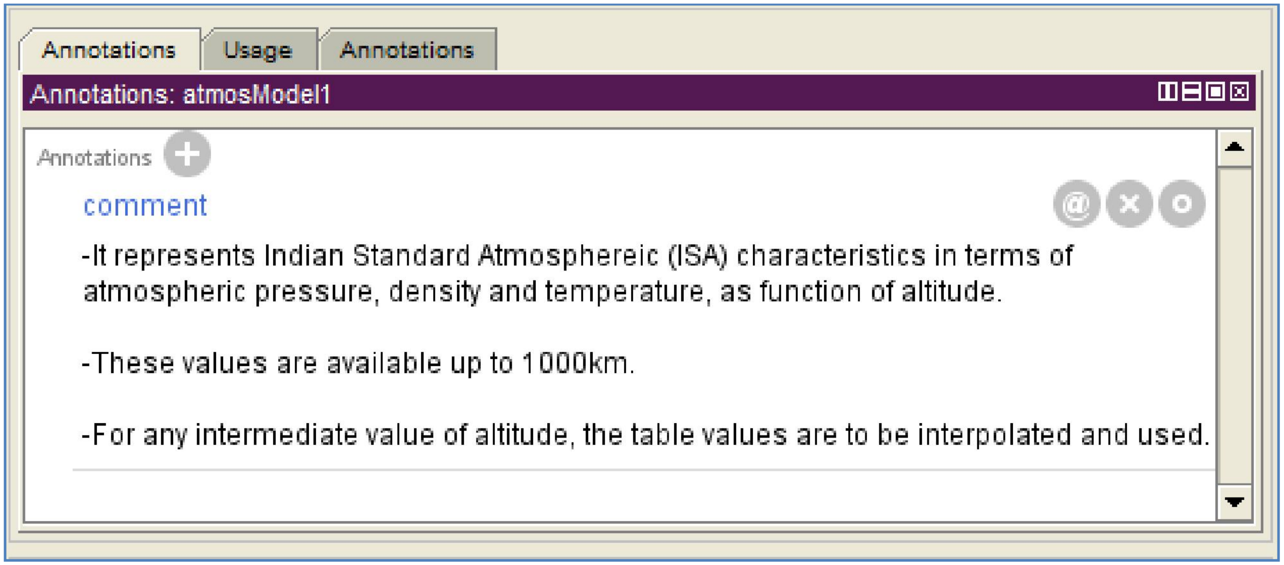


- ✓ **Inverse property**
- ✓ **Singular/plural**

Model and Module



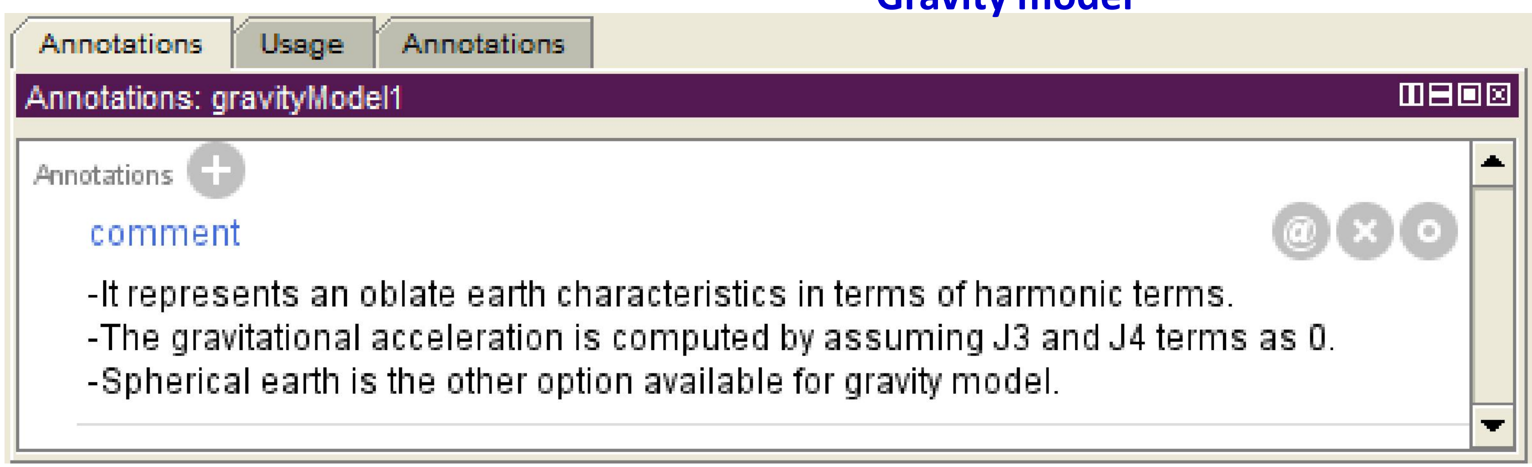
Annotations on Engineering models



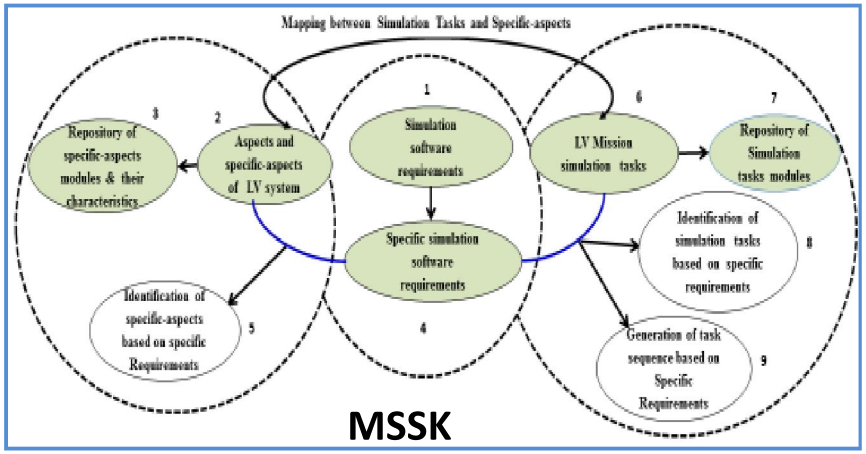
Atmosphere model

RDFS: comment

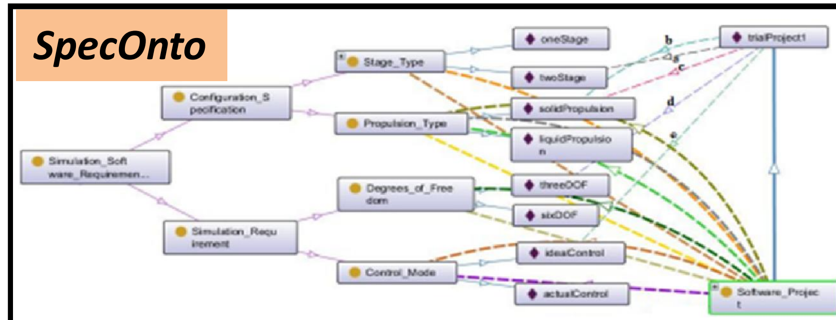
Gravity model

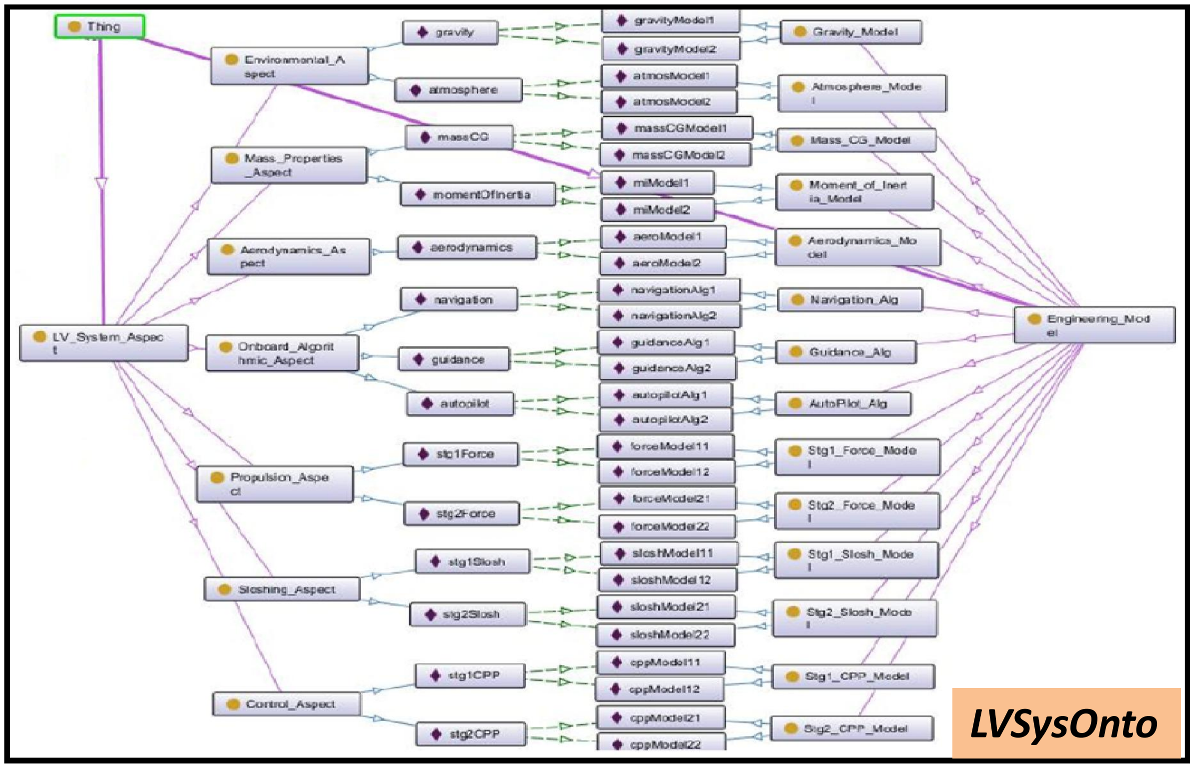


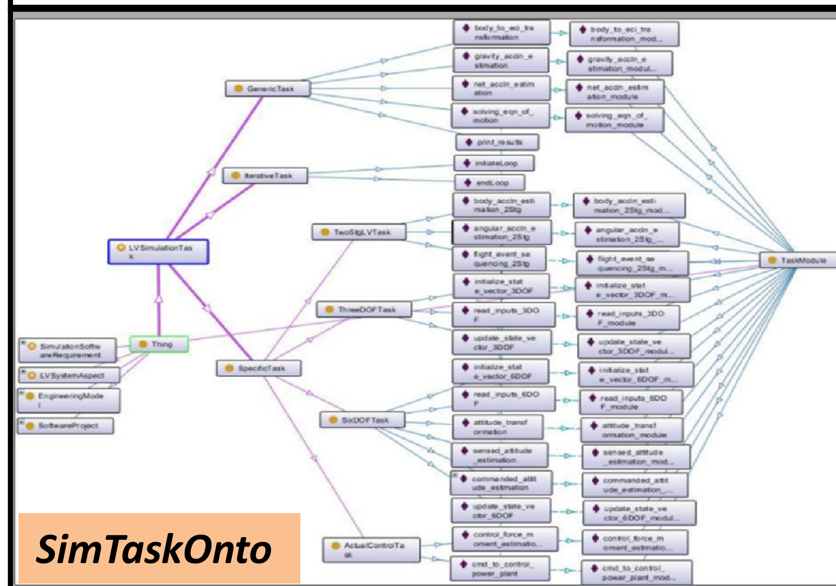
LV system ontology and its components



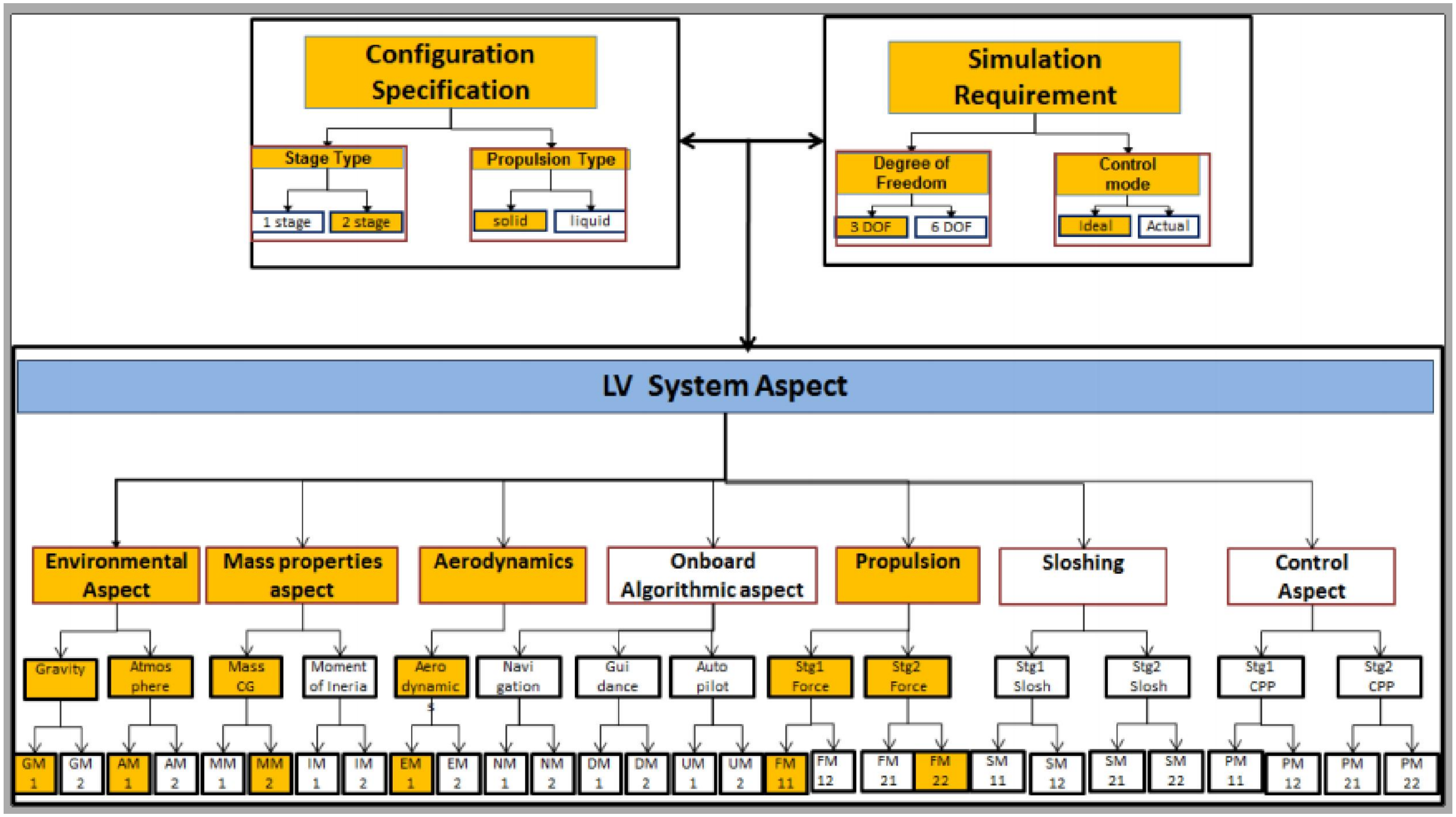
Ontology Name	No of Logical axioms	No of Individuals	No of Properties	No of Classes	OWL-DL Variant
SpecOnto	55	11	7	9	$\mathcal{ALCOQ}$
LVSysOnto	361	51	13	26	$\mathcal{ALCOI(D)}$
SimTaskOnto	117	46	12	10	$\mathcal{ALUHI(D)}$







Knowledge on Identification of specific-aspects



Key Challenge

Interrelationships between various subsystems –**play a crucial role**

- in identifying *specific-aspects*

Semantic Web Rule Language (SWRL)

- Specific-aspects required – to be identified based on the predefined relations
- Relation: Combination of specific values of requirements are to be mapped to combination of specific values of aspects.
- OWL : **Calls for additional expressive power** – for free mixing of properties and classes

- SWRL: a rule-based knowledge representation language
  - Express relations between combinations of individuals referenced by properties
  - Rule: antecedent (body) → consequent (head)
  - allow writing of rules in terms of OWL concepts
  - free mixing of properties and class expressions directly
  - **layers well on top of OWL**

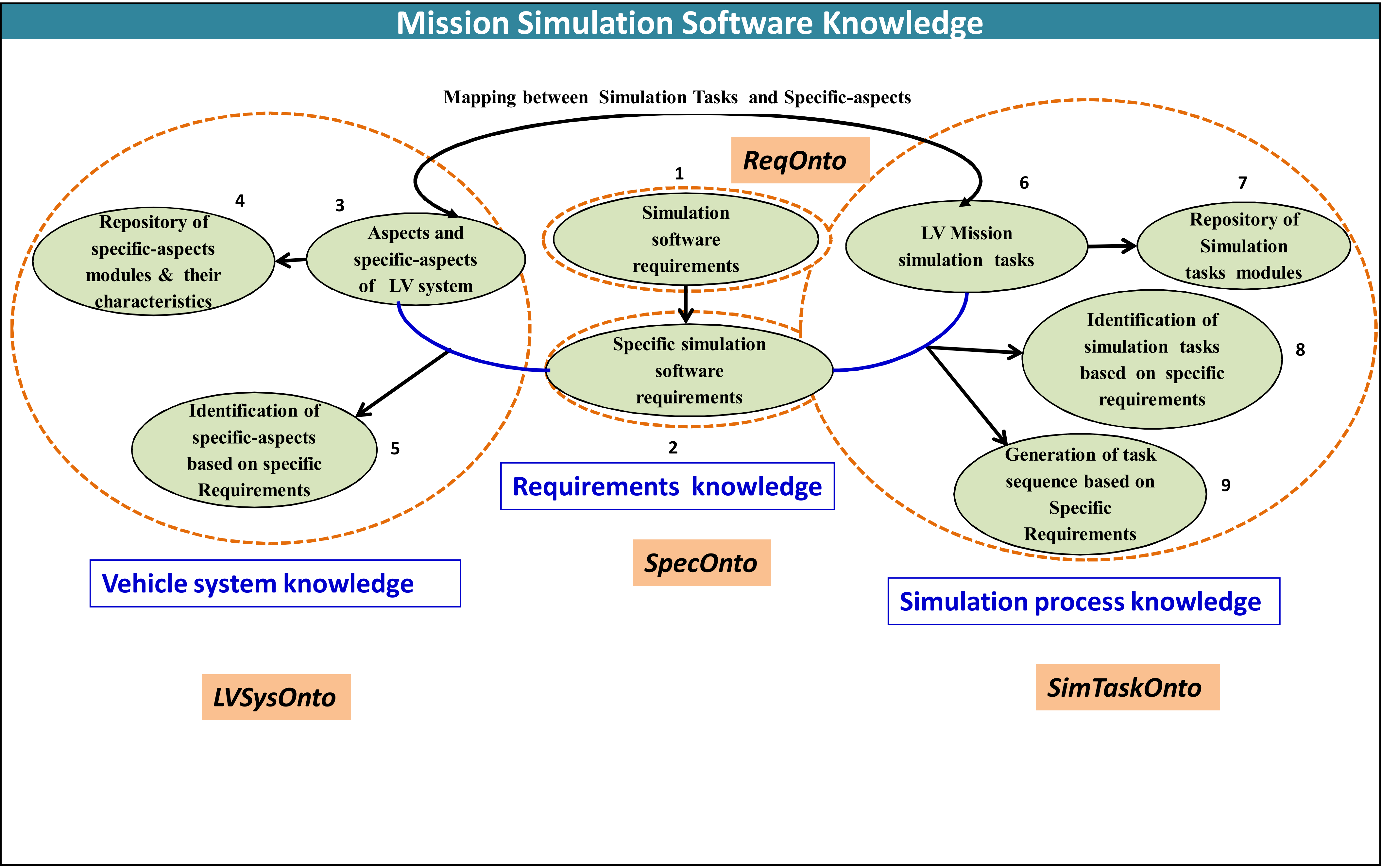
SWRL- suitable to formalize inference knowledge

Conceptualization of Knowledge from domain experts

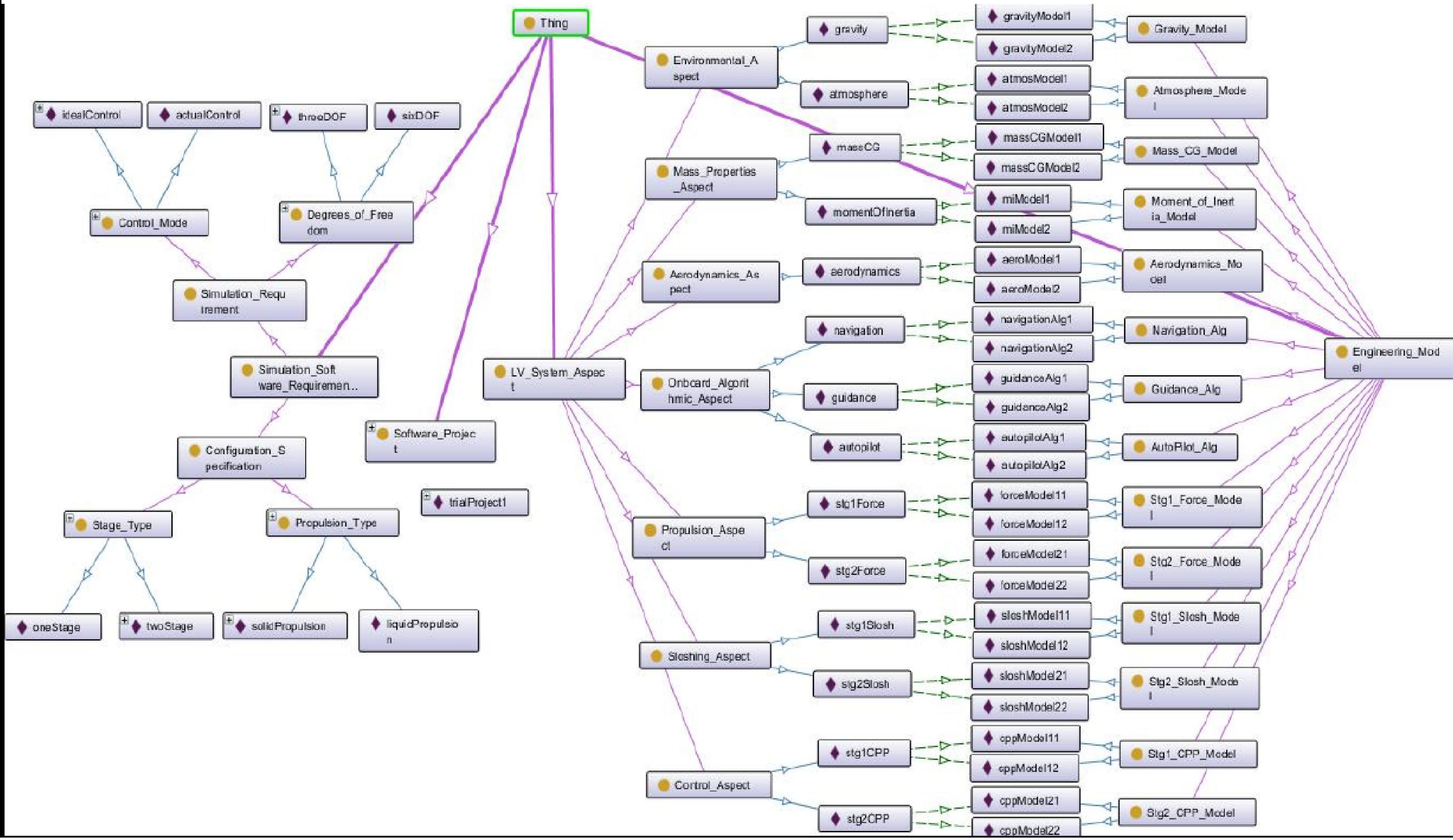
Expert Involved	Key pieces of Knowledge
System Expert	• Gravity and atmosphere aspects need to be included in all projects as they are essential for all LV simulations • Projects only with actual control and six DOF will require the aspect of Control Power Plant (CPP) for simulation • The control mode - actual control - can be specified for no project that uses only three DOF • **Taking care of six DOF requires more specific aspects to be simulated compared to that for three DOF**
Inertial System Expert	• Projects only with actual control and six DOF will require the aspect of Navigation for simulation
Guidance and Control Expert	• Projects only with actual control and six DOF will require the aspect of Guidance and Autopilot aspects for simulation.
Vehicle Engineering Team	• Whatever may be the requirements for a project, mass and aerodynamics aspects need to be simulated • Projects only with actual control and six DOF will require the aspect of Moment of Inertia for simulation
Propulsion Expert	• Force aspect of the respective stage needs to be simulated for all simulations as per the stage type specified for a project • **Whenever a rocket stage uses liquid stage, the sloshing aspect needs to be simulated.**

Semantic Rules

R1	SoftwareProject (?x), EnvironmentalAspect (?y) →hasSpecificAspect (?x,?y)
R2	SoftwareProject (?x)→ hasSpecificAspect (?x, massCG)
R3	SoftwareProject (?x), AerodynamicsAspect(?y) → hasSpecificAspect(?x, ?y)
R4	SoftwareProject (?x) → hasSpecificAspect(?x, stg1Force)
R5	SoftwareProject (?x), hasNumberOfStages (?x, twoStage) → hasSpecificAspect (?x, stg2Force)
R6	SoftwareProject(?x), hasNumberOfDOF(?x, sixDOF), OnboardAlgorithmicAspect(?y) → hasSpecificAspect(?x, ?y)
R7	SoftwareProject (?x), hasNumberOfDOF(?x, sixDOF), hasControlMode(?x, actualControl) → hasSpecificAspect (?x, stg1CPP)
R8	SoftwareProject(?x), hasNumberOfStages(?x,twoStage), hasNumberOfDOF(?x, sixDOF), hasControlMode(?x, actualControl) → hasSpecificAspect(?x, stg2CPP)
R9	SoftwareProject (?x), hasNumberOfDOF (?x, sixDOF) → hasSpecificAspect(?x, momentOfInertia)
R10	SoftwareProject (?x), hasStage1Type (?x, liquidPropulsion) → hasSpecificAspect(?x, stg1Slosh)
R11	SoftwareProject (?x), hasNumberOfStages(?x, twoStage), hasStage2Type(?x, liquidPropulsion) → hasSpecificAspect (?x, stg2Slosh)
R12	SoftwareProject (?x), hasNumberOfDOF(?x, threeDOF), hasControlMode(?x, actualControl)→ hasIncompatiableRequirement(?x, actualControl) ▪ **To formalize incompatible requirements**


Mission Simulation Software Knowledge
Mapping between Simulation Tasks and Specific-aspects
ReqOnto
1
Simulation software requirements
2
Specific simulation software requirements
3
Aspects and specific-aspects of LV system
4
Repository of specific-aspects modules & their characteristics
5
Identification of specific-aspects based on specific Requirements
6
LV Mission simulation tasks
7
Repository of Simulation tasks modules
8
Identification of simulation tasks based on specific requirements
9
Generation of task sequence based on Specific Requirements
Requirements knowledge
Vehicle system knowledge
SpecOnto
Simulation process knowledge
LVSysOnto
SimTaskOnto

Partial expansion of LV Simulation Ontology



LV Mission Simulation Ontology(LVSimOnto) : an integrated ontology

Ontology Name	No of Logical axioms	No of Individuals	No of classes	No of Properties	No of SWRL rules	OWL -DL Variant
LVSimOnto	567	108	45	32	30	$\mathcal{ALCHOIQ(D)}$

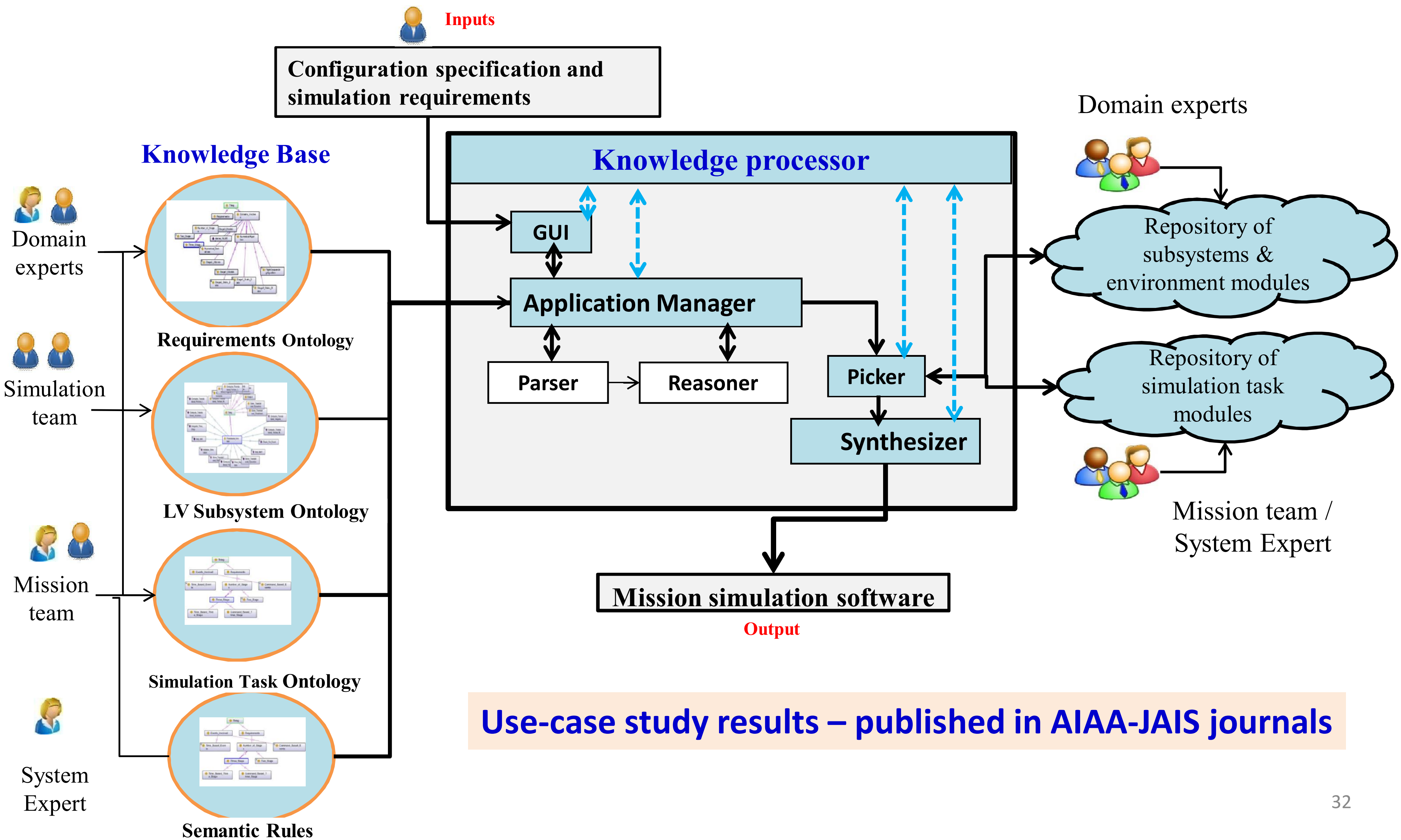
For completely automating end-to-end software generation

- used in any automation framework
- lucid interface with users
- extract as you required
- synthesize them in a seamless manner.

Designed and developed **a set of knowledge-driven software agents**

knowledge processor

Ontology enabled automated synthesis of Mission simulation software



Use-case study results – published in AIAA-JAIS journals

Querying and Reasoning

SPARQL and Result

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX onto: <http://www.semanticweb.org/vssc/ontologies/2015/0/LVSimOnto#>
PREFIX sim:  <http://www.semanticweb.org/vssc/ontologies/2015/0/untitled-ontology-66#>
SELECT ?subject ?object
        WHERE { ?subject sim:hasMcCabeComplexity ?object filter(?object< 10)}
```

subject	object
gravityModel1	"7"^^<http://www.w3.org/2001/XMLSchema#int>
atmosModel1	"9"^^<http://www.w3.org/2001/XMLSchema#int>
guidanceAlg2	"9"^^<http://www.w3.org/2001/XMLSchema#int>
aeroModel2	"8"^^<http://www.w3.org/2001/XMLSchema#int>
massCGModel2	"9"^^<http://www.w3.org/2001/XMLSchema#int>

Execute

DL Query and Result

DL query:

Query (class expression)

(hasStage1Type some PropulsionType) and (hasStage2Type some PropulsionType) and (hasStage3Type some PropulsionType)

Execute | Add to ontology

Query results

Sub classes (1)

SoftwareProject

Instances (1)

testProject

- [ ] Super classes
- [ ] Ancestor classes
- [ ] Equivalent classes
- [x] Subclasses
- [ ] Descendant classes
- [x] Individuals

Ontology enabled approach- Salient benefits

- Automatic transformation of knowledge to design & code
- Explanation Generation
- Users-Need not have the knowledge about semantic technology
- Sophisticated way to accumulate knowledge
- Automatic generation of simulation requirements-ontological way
- Allowed and non-allowed entities thro domain & range
- Aids in selecting suitable model (fidelity) – for type of simulations
- Sharing and reusing : for other applications
- Improving quality of existing practices in LV domain

Summary

- Key points

    1. **Think before you leap**
    2. **No crowbar to hit a cockroach**
    3. **Don’t go for the best; go for the right**
    4. **Don’t forget the past**
    5. **The proof is in the pudding**

Summary

The AI Canvas

Decision making tool: Harvard Business review

Use it to think through how AI could help with business decisions.

PREDICTION
What do you need to know to make the decision?

JUDGMENT
How do you value different outcomes and errors?

ACTION
What are you trying to do?

OUTCOME
What are your metrics for task success?

INPUT
What data do you need to run the predictive algorithm?

TRAINING
What data do you need to train the predictive algorithm?

FEEDBACK
How can you use the outcomes to improve the algorithm?

SOURCE AJAY AGRAWAL ET AL.

References

- To explore
  - **Pizza ontology**
  - **GENE ontology**
  - **Plant ontology**
  - **Yahoo ontology**
  - **LVSimOnto**

"Success of any organization depends more than anything on its knowledge: on what it knows - and how it uses what it knows"

Thank You