# An Ontological Approach to System of Systems Engineering in Product Development

**Ludvig Knöös Franzén**

Ingo Staack, Christopher Jouannet, Petter Krus

LINKÖPING UNIVERSITY

# Agenda

- Introduction
- System-of-Systems Engineering in Product Development
- Approach
- Ontology
- Method
- Testing the proposed method with a Case-Study
- Implementation
- Available Design Spaces
- Discussion
- Future work
- Conclusions

# Introduction

- Steady increment of complexity for today's aerospace applications and their development
- More interconnections with the operational environment and other systems
- The operational environment is changing throughout a product's lifecycle
- Long lead-times during development and expected lifespans -> Increased risk
- A highly intertwined problem where requirements might change

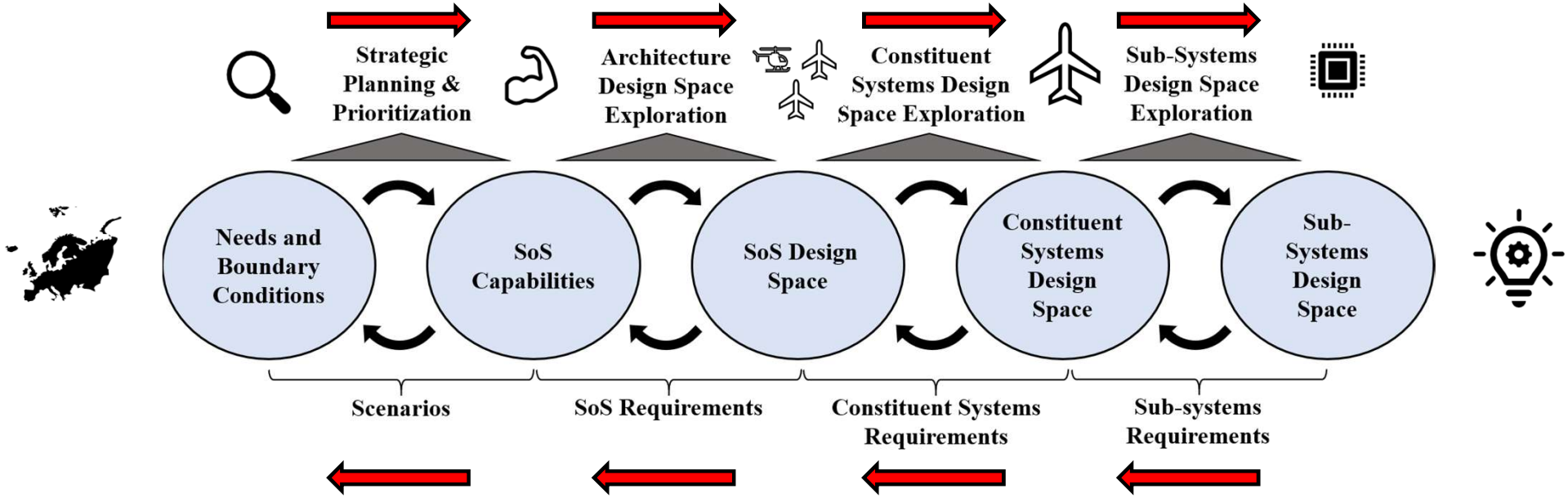
- Forecasts of an ever-changing world needs to be incorporated early in the design process.
- Predicting the future and facilitating system's survivability
- The focus shifts from fulfilling system requirements to deliver capabilities over time.

# System-of-Systems Engineering in Product Development

- Increased interest in the concept of System-of-Systems (SoS)
- Collaboration between constituent systems of a SoS –> SoS capabilities
- Capability and System-of-Systems Engineering (SoSE)

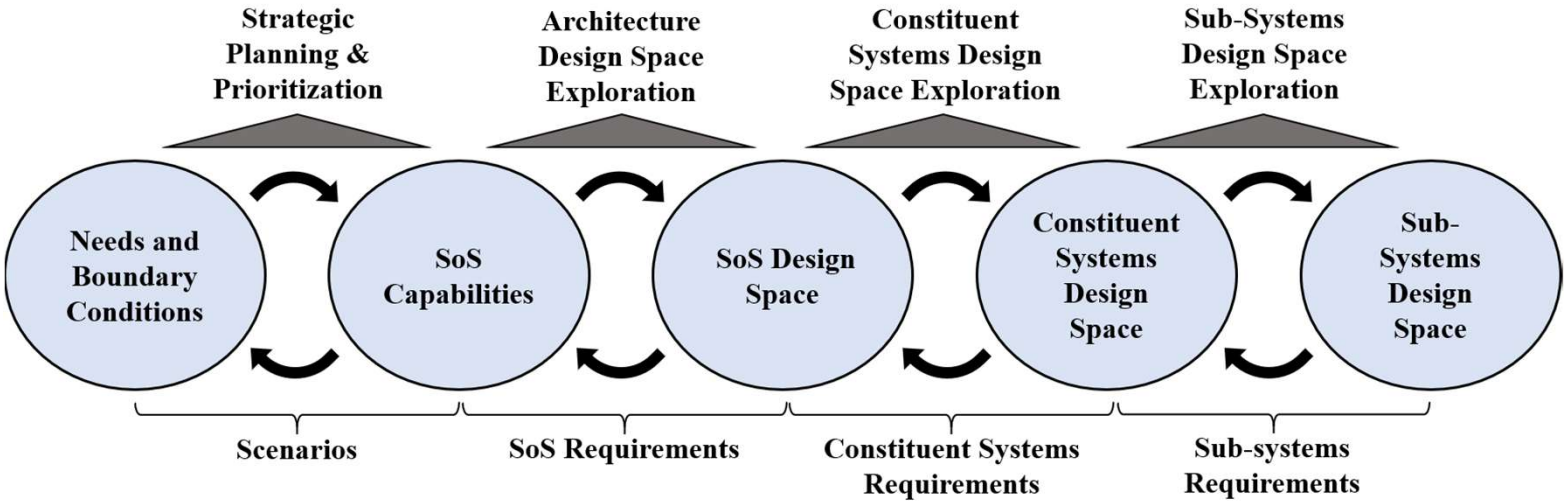
# System-of-Systems Engineering in Product Development

- A holistic approach for product development in an SoSE context
- Five intercorrelated levels of interest – SoS design space explorations



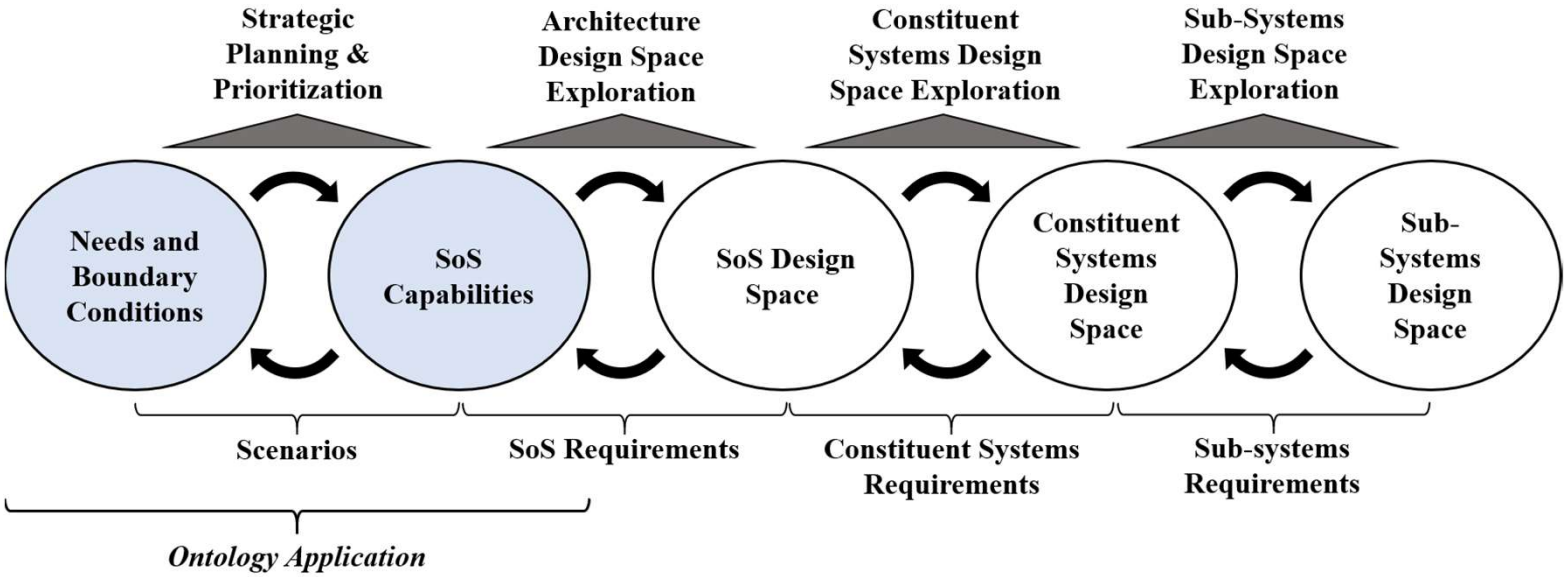
# System-of-Systems Engineering in Product Development

- How can this envisioned process be realized?



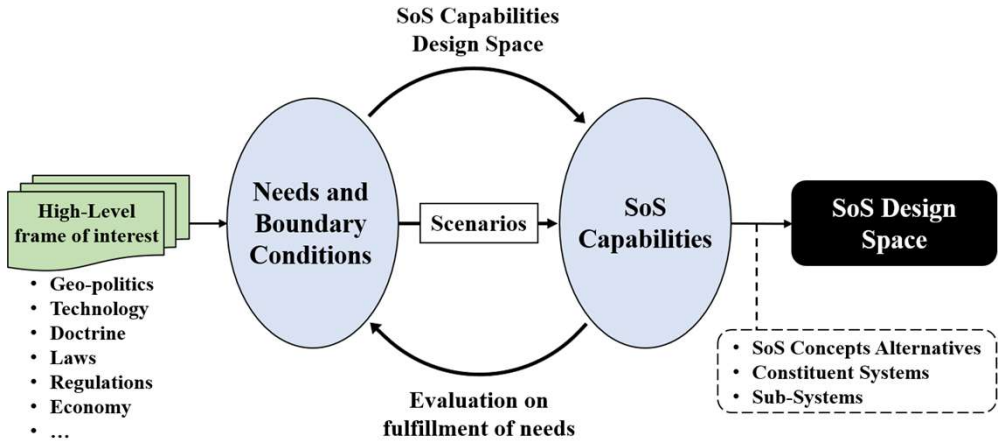
# System-of-Systems Engineering in Product Development

- This paper addresses an initial approach to tackle the first two levels of interest using Ontology



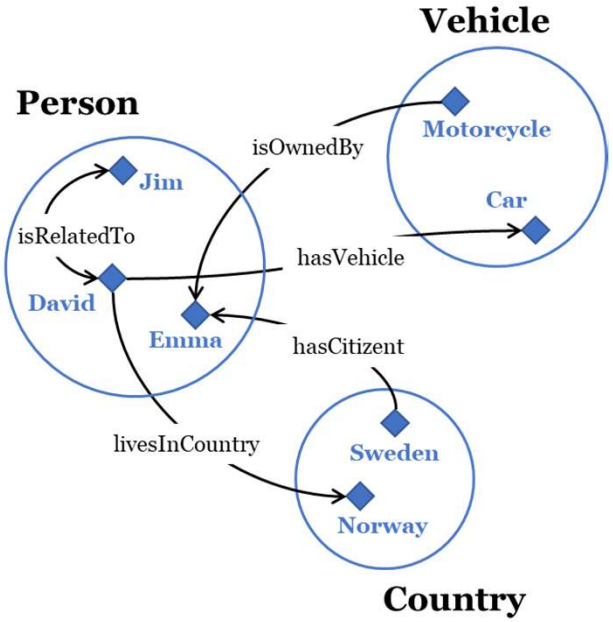
# Approach

- Approaching the first levels of interest by using Ontology
- Using Ontology to:
  - Created an environment where needs and boundary conditions can be varied
  - Define and prune a SoS Design Space
  - Explore an available Design Space



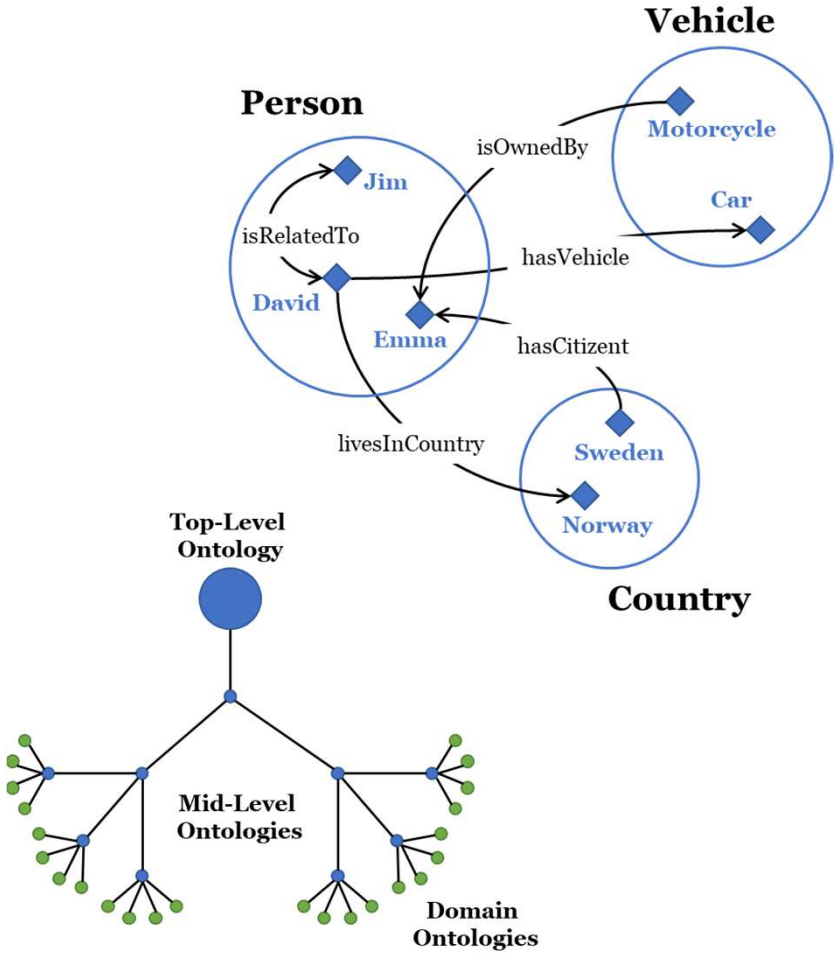
# Ontology

- What is an Ontology?
- Ontology:
  A formal and explicit representation of a given domain that involves knowledge of the involved entities and the relationships that exists between them
- Steady increase of usage in the area of Systems Engineering
- Differences between Ontology and modelling languages?
  - Increased interoperability
  - Improved scalability
  - Open world assumption
  - Supports reasoning



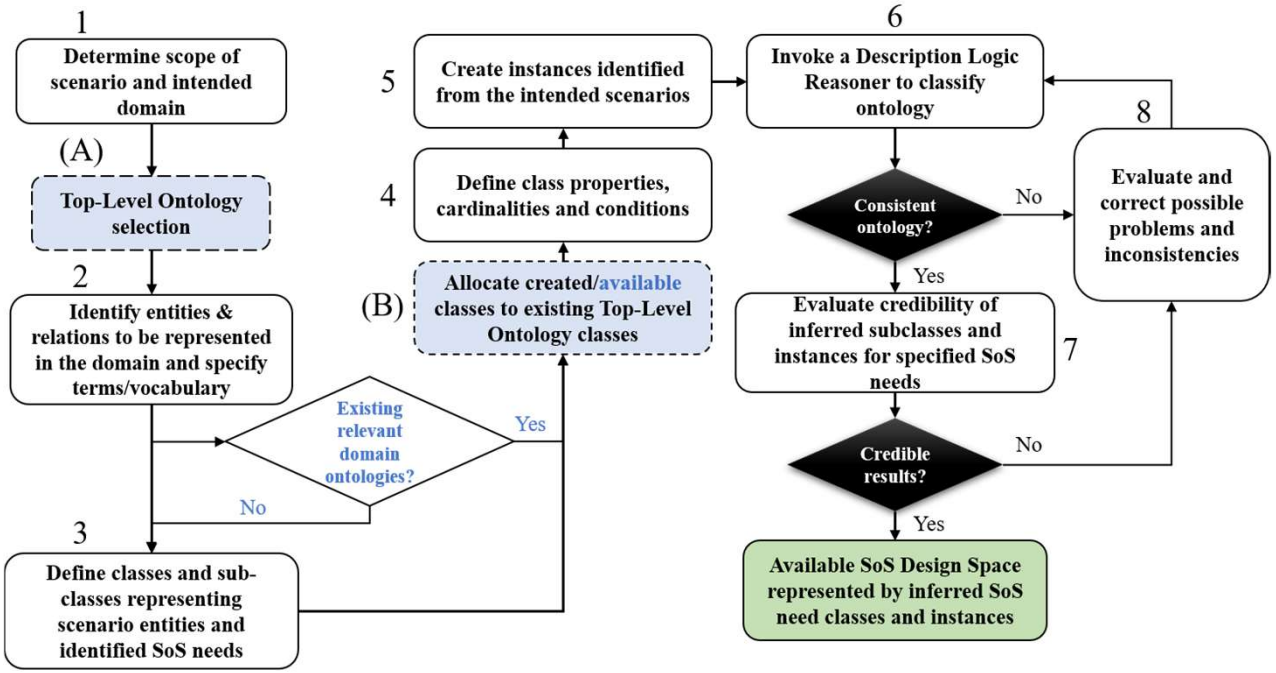
# Ontology

- Ontologies are complementary to UML and SysML
- Standard Ontology languages:
  - OWL (Web Ontology Language)
  - RDF (Resource Description Framework)
- OWL features Description Logic Reasoning (DLR)
- DLR checks the consistency of the ontology
- DLR allows automatic inference of relationships
- DLR – scalability at the cost of computational time
- Domain, mid and Top-level ontologies



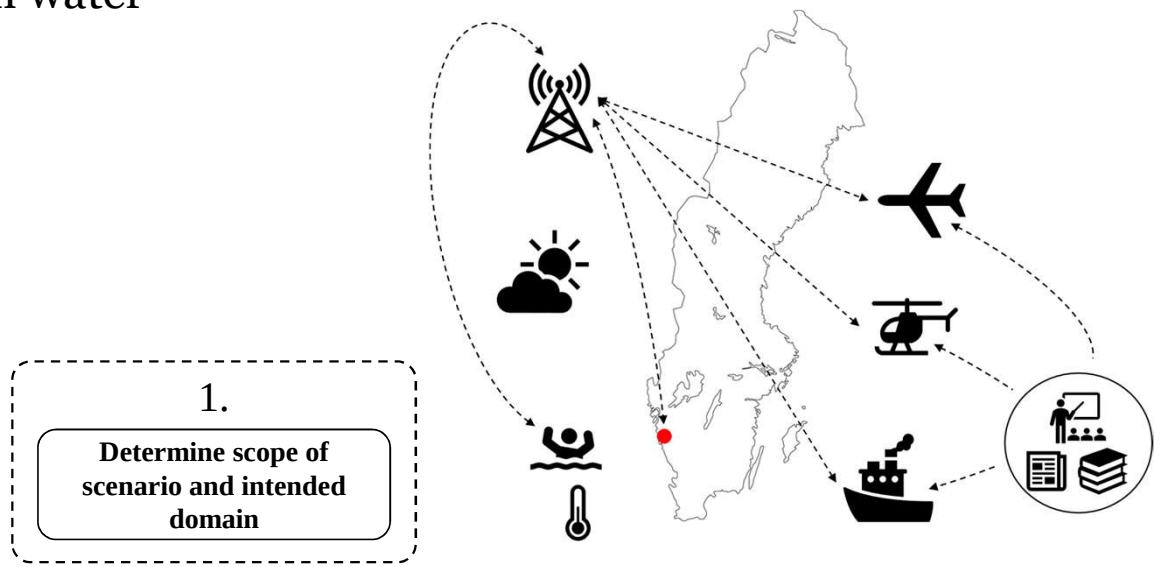
# Method

- Method of modelling an ontology intended for design space explorations on SoS:



# Testing the proposed method with a Case-Study

- A small Swedish Search and Rescue (SaR) case-study
- Based on available resources of the Swedish Maritime Administration (SMA)
- Fictitious scenario with rescue subjects in water
- Available assets
  - **AgustaWestland AW139**, Helicopter
  - **Bombardier Dash 8 Q300**, Aircraft
  - **Fictitious Fast Boat**, Sea vessel
  - **Fictitious Slow Boat**, Sea vessel

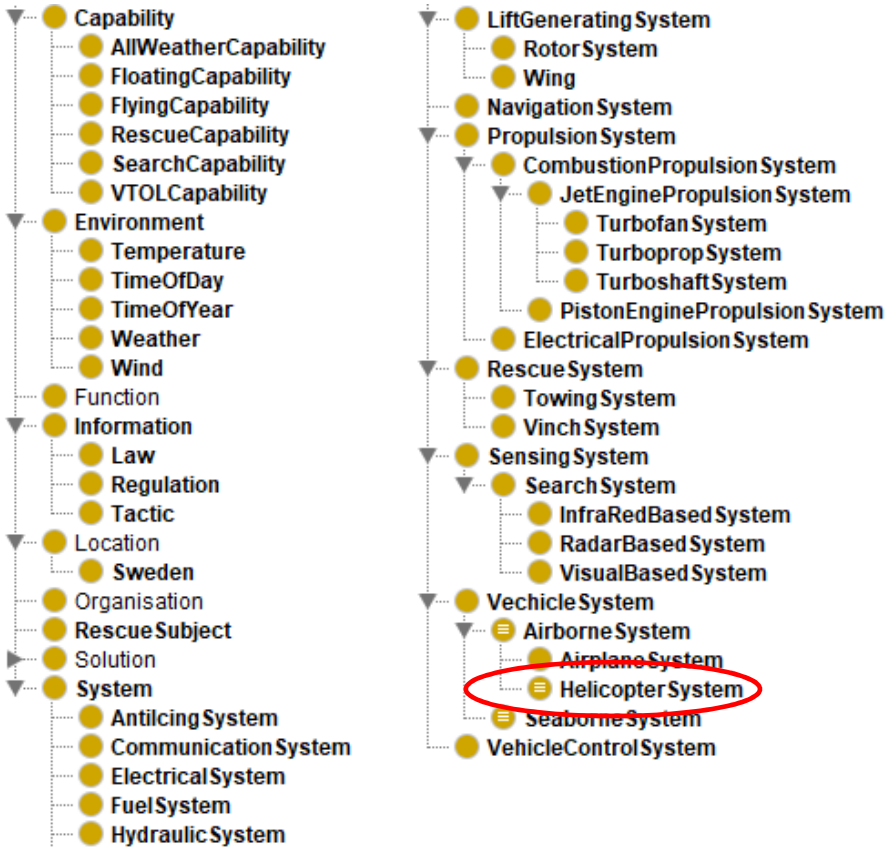


# Implementation

- Ontology editing software: Protégé 5.5.0
- No top-level ontology structure or existing ontology is used
- Ontology structure and hierarchy:
  - Subsumptive containment hierarchy, “is a”

2. **Identify entities & relations to be represented in the domain and specify terms/vocabulary**

3. **Define classes and sub-classes representing scenario entities and identified SoS needs**

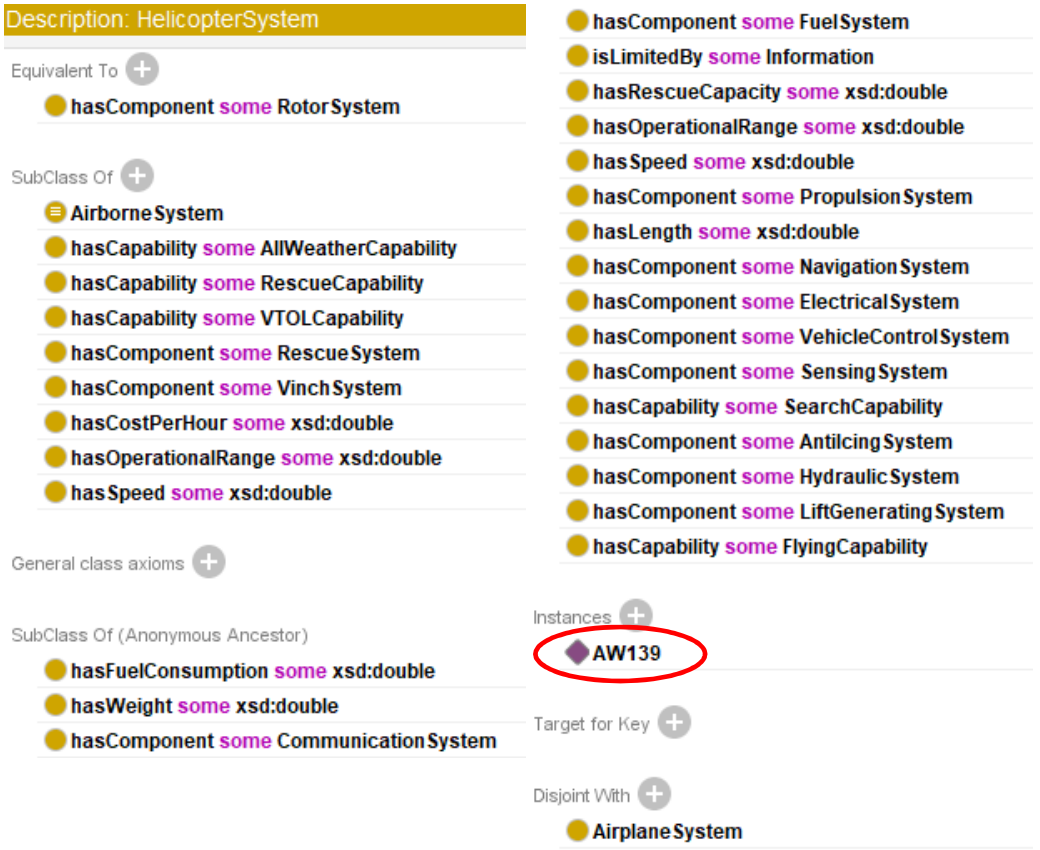


# Implementation

- Ontology editing software: Protégé 5.5.0
- No top-level ontology structure or existing ontology is used
- Ontology structure and hierarchy
  - Subsumptive containment hierarchy, “is a”
- Relationships and Data properties:

4.

**Define class properties, cardinalities and conditions**

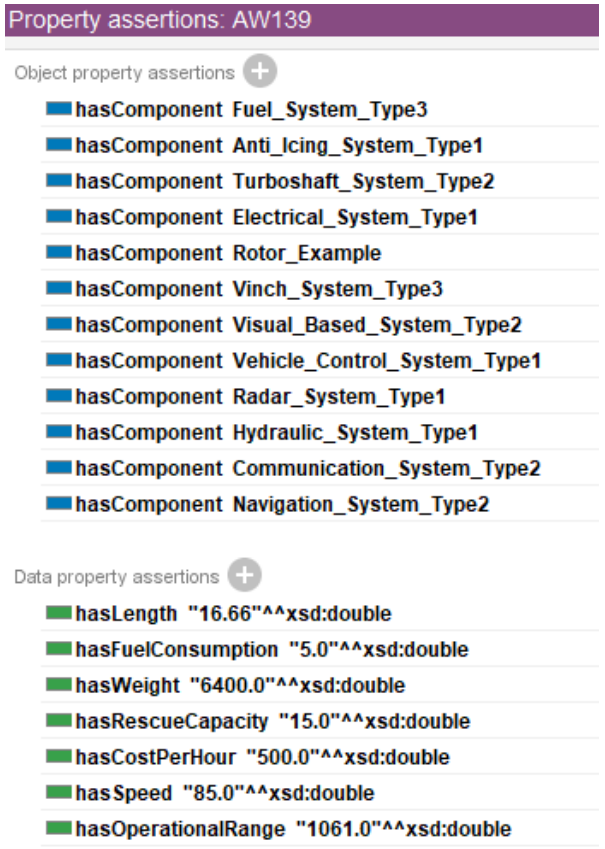


# Implementation

- Ontology editing software: Protégé 5.5.0
- No top-level ontology structure or existing ontology is used
- Ontology structure and hierarchy
  - Subsumptive containment hierarchy, “is a”
- Relationships and Data properties
- Instances:

5.

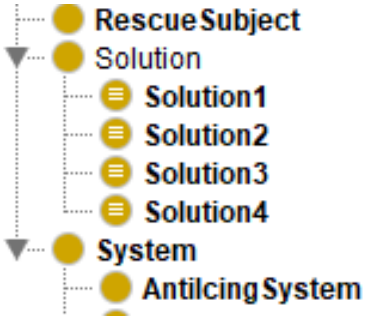
**Create instances identified from the intended scenarios**



# SoS Solution Classes

- Define the Solution Classes based on identified needs
  - *"Necessary* and *sufficient conditions"*

Asserted Ontology Hierarchy




## Solution Class 1

Description: Solution1

Equivalent To

(hasCapability some SearchCapability)
and (hasSpeed some xsd:double[>= "5.0"^^xsd:double])

## Solution Class 2

Description: Solution2

Equivalent To

Solution
and ((hasCapability some RescueCapability)
and (hasCapability some SearchCapability))

## Solution Class 3

Description: Solution3

Equivalent To

Solution
and (hasCostPerHour some xsd:double[<= "10.0"^^xsd:double])

## Solution Class 4

Description: Solution4

Equivalent To

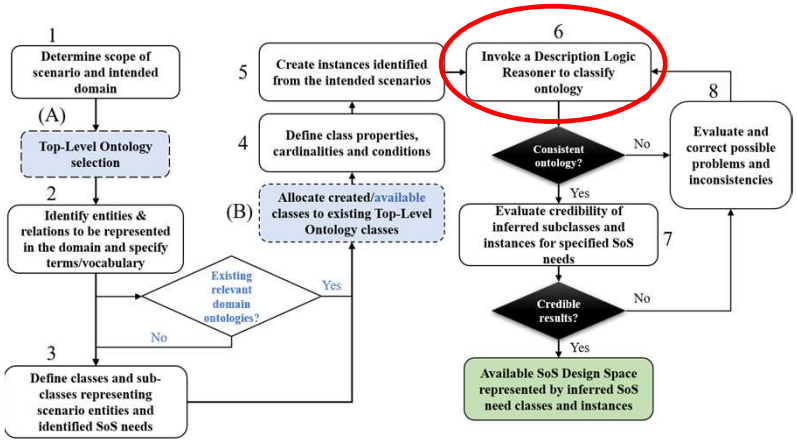
Solution
and (hasCapability some RescueCapability)
and (hasCapability some SearchCapability)
and (hasLength some xsd:double[>= "15.0"^^xsd:double])
and (hasRescueCapacity some xsd:double[>= "10.0"^^xsd:double])

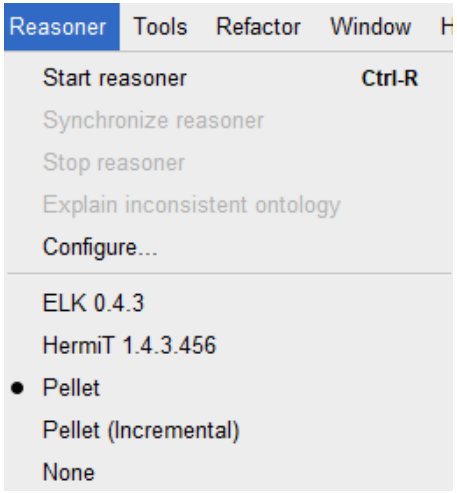
3.

**Define classes and sub-classes representing scenario entities and identified SoS needs**

# Description Logic Reasoning (DLR)

- Define the Solution Classes based on identified needs
- Invoke a reasoner
- Generate an inferred ontology and the design spaces



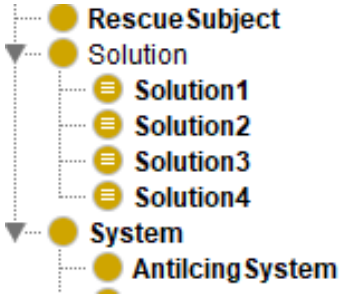

6

**Invoke a Description Logic Reasoner to classify ontology**

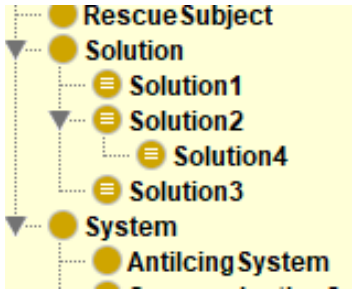
# Inferred ontology with solution classes

- The solution classes
- Available design spaces

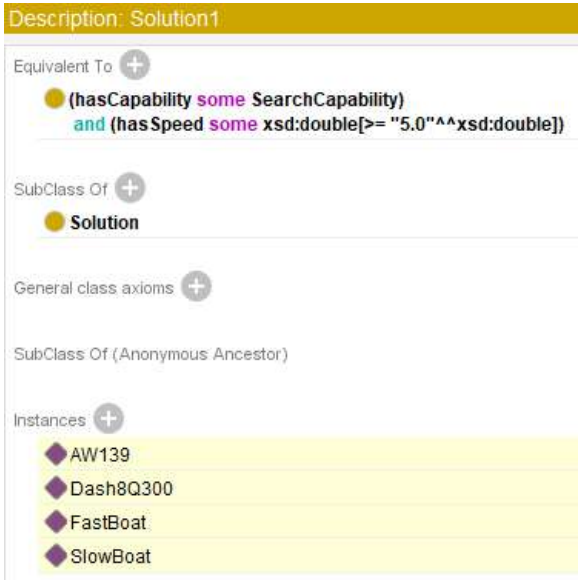
Asserted:



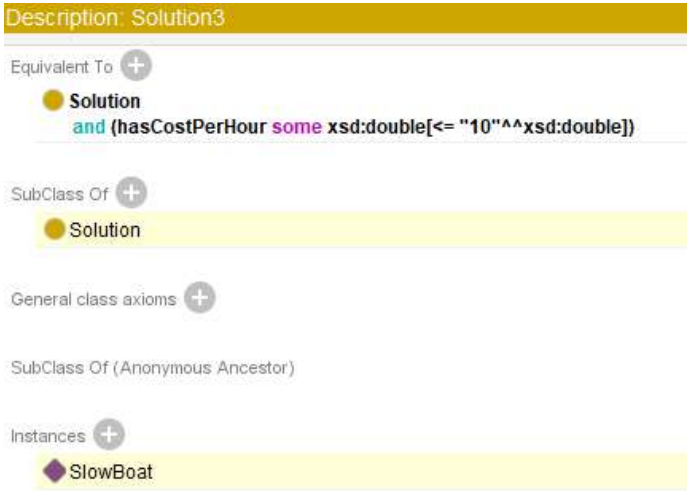

Inferred:




Solution Class 1



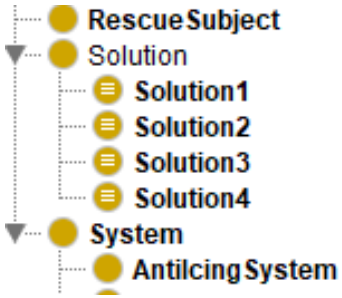

Solution Class 3



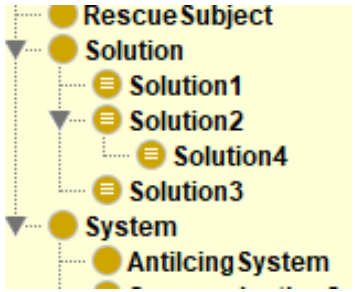
# Inferred ontology with solution classes

- The solution classes
- Available design spaces

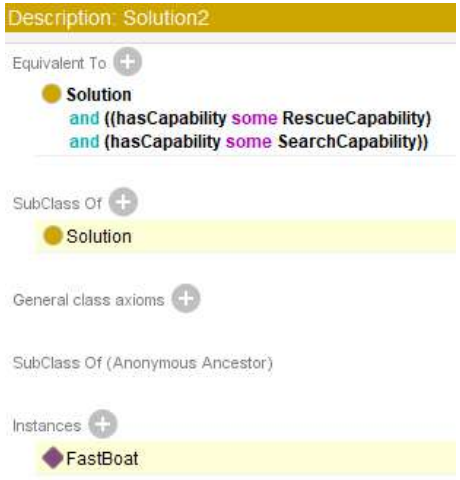
Asserted:




Inferred:




Solution Class 2

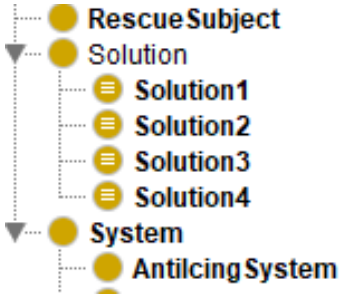



Solution Class 4

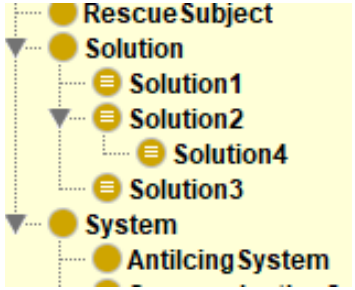
# Available Design Spaces

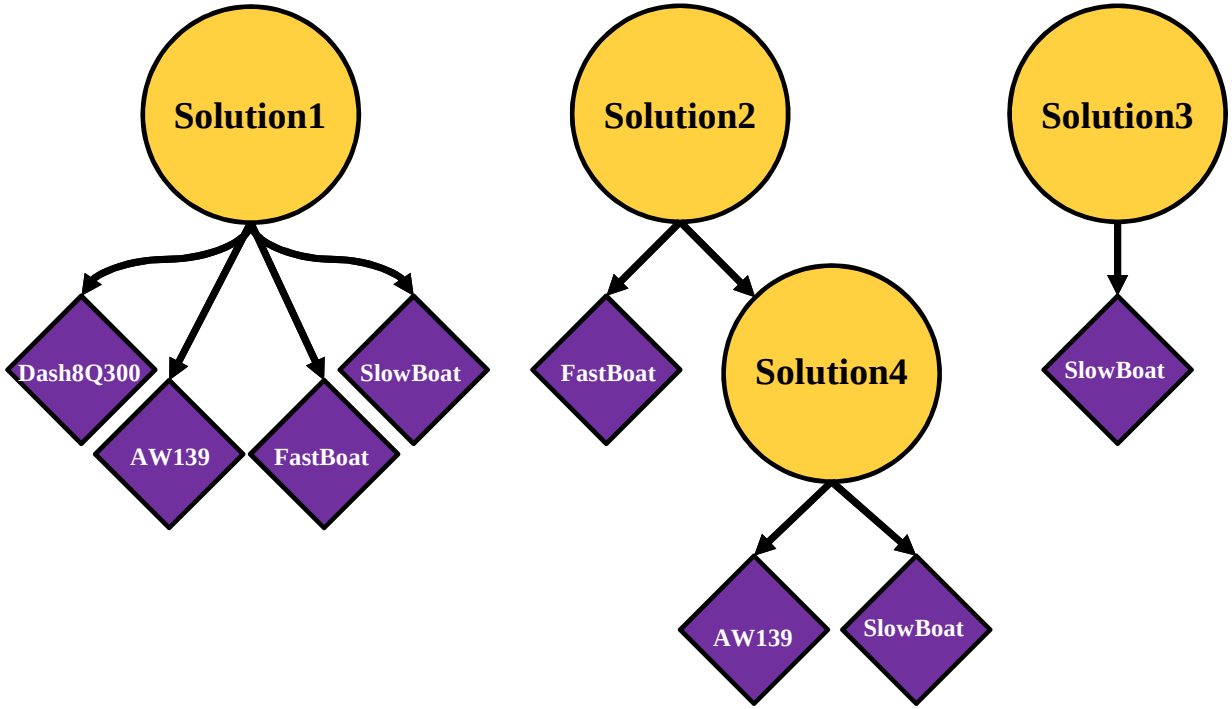
- Suitable SoS constituents

Asserted:



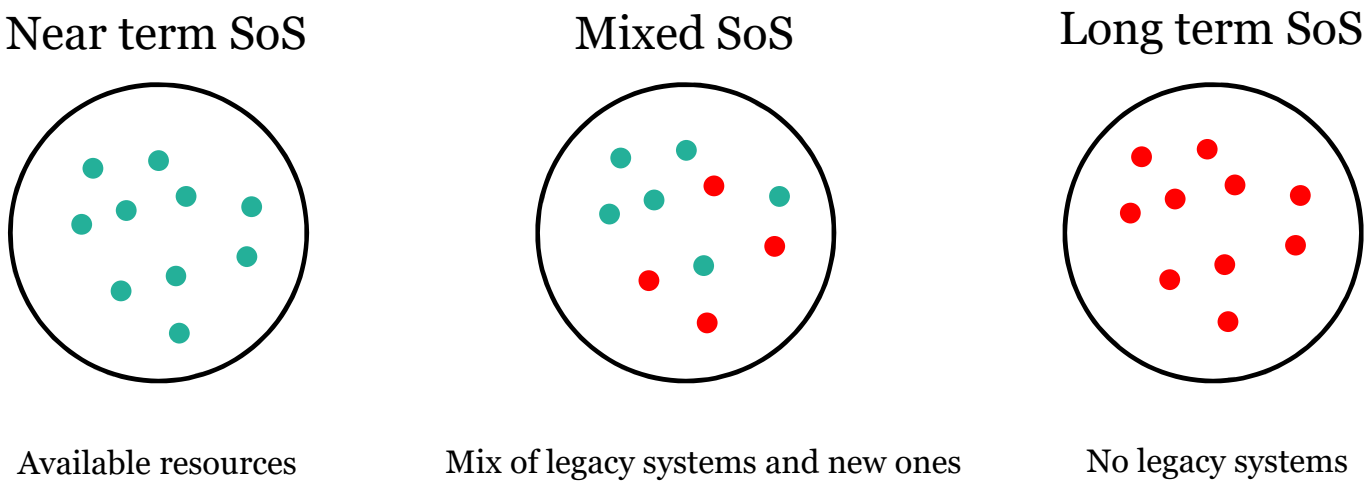

Inferred:



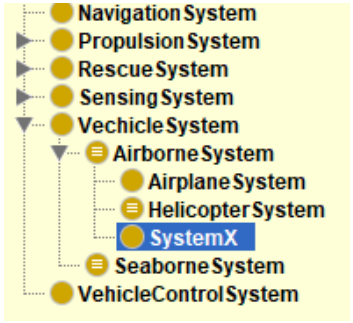
# Discussion

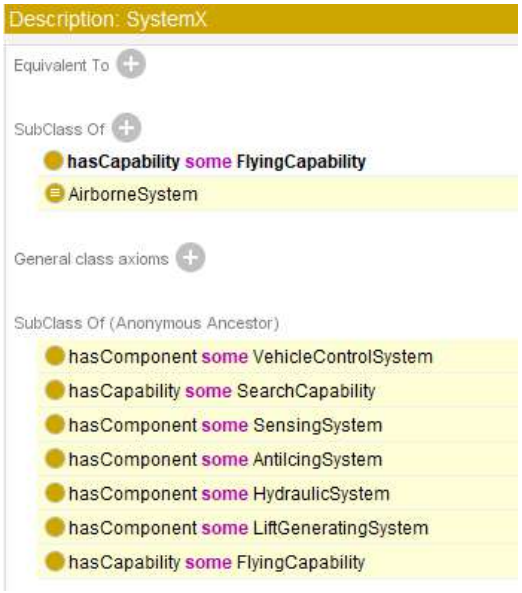
- The proposed method is intended to be used for any SoS design space creation and reduction.
- The case study corresponds to a “near term SoS-composition”

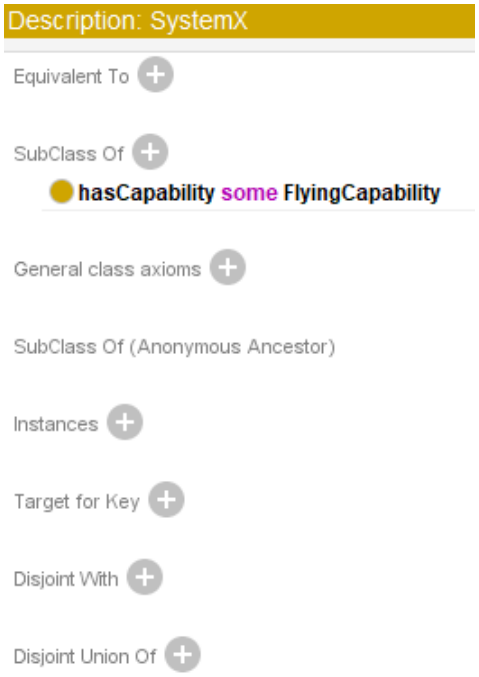


# Discussion

- Expansion of case-study and the benefits with DLR
- Scalability of the ontology

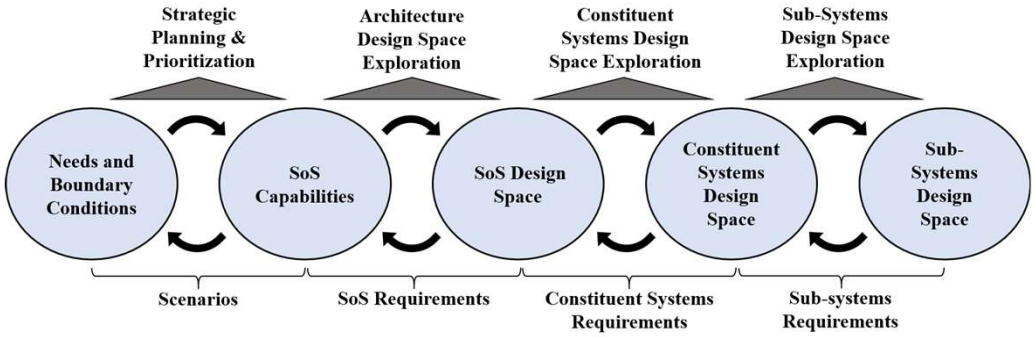


# Discussion

- Modelling SoS capabilities that are realized by system collaborations
- Needs to required capabilities to functional breakdowns and allocation
- Varying the initial conditions of the scenario to identify persistent solutions
- Simple case study, but with great potential for expansion

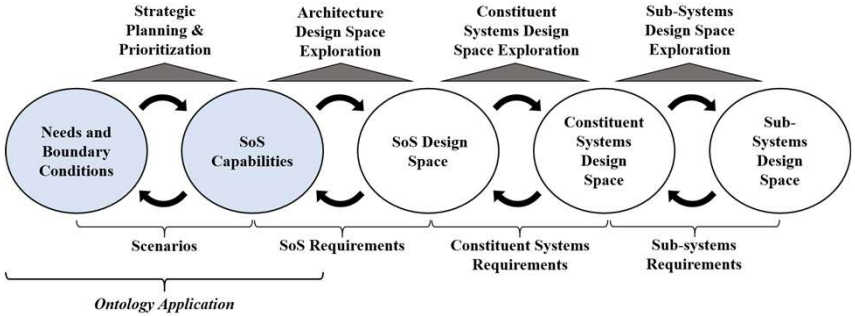


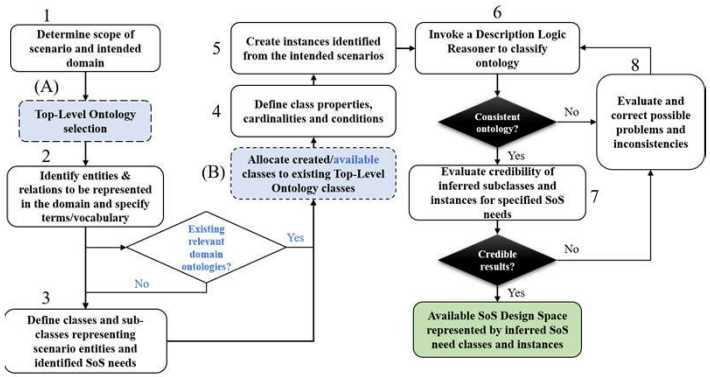
# Future work

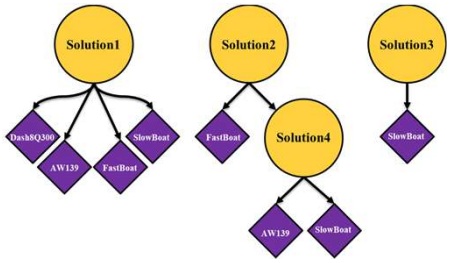
- Extracting the knowledge of the inferred ontology
- Use the outcomes of the ontology with other approaches:
  - Matrix-based -> Numerical calculations, further design space refinements, …
  - Agent-based simulations -> Measure of effectiveness, number of assets, …
  - Etc.
- Expanding the case-study
  - More entities and relationships
  - More detailed capability and functional breakdowns
  - Varying the initial conditions based on scenario and epoch analyses

# Conclusions

- Approaching the first levels of an envisioned SoS-process
- A way of generating and pruning the available design space
- Ontology provides a resilient way of exploring and generating SoS design spaces based on specified needs



# Questions?

Thank you for listening!

Ludvig.knoos.franzen@liu.se

www.liu.se

LINKÖPING UNIVERSITY