“Knowlang” is designed as part of the ASCENS project in an attempt to represent the knowledge in the domain of service component ensembles in a formal and declarative way.

Challenging task: the representation model has to incorporate both a common vocabulary of concepts and the means to support an autonomic, self-aware and adaptive behavior.

My introduction to modeling, knowledge representation and language specification: useful for the RELATE project.
Based on

1. “Requirements and Initial Model for Knowlang – A Language for Knowledge Representation in Autonomic Service-Component Ensembles”
   • Emil Vassev (Lero, U of Limerick), Mike Hinchey (Lero, U of Limerick), Benoit Gaudin (Lero, U of Limerick), Paddy Nixon (U of Tasmania)
   • May 2011: 4th Inter. C* Conference on Computer Science and Software Engineering

   • Emil Vassev (Lero, U of Limerick), Mike Hinchey (Lero, U of Limerick)
   • October 2011: 3rd International Conference on Data Mining and Intelligent Information Technology Applications

3. “Representing Knowledge in Robotic Systems with Knowlang”
   • Emil Vassev (Lero, U of Limerick), Mike Hinchey (Lero, U of Limerick)
   • October 2011: 1st International ISoLA Workshop on Software Aspects of Robotic Systems
Awareness as a starting point

Awareness as a product of:
• knowledge representation
• knowledge processing
• monitoring

Kinds of awareness:
• self-awareness (own states, capacity, capabilities, connections, etc.)
• context-awareness (interaction with environment, other components)
Knowledge requirements indicate four different knowledge domains (or corpuses):

- **SC knowledge** – internal configuration, resource usage, communication ports, services, goals, events, etc.
- **SCE knowledge** – architecture topology, structure, system-level goals and services, public interfaces, etc.
- **Environment knowledge** – parameter and properties of the operational environment e.g. external systems, communication interfaces, integration mechanisms, etc.
- **Situational knowledge** – specific situations, involving one or more SCs and eventually the environment
• Different representation techniques for different kinds of knowledge. Popular ones:
  • Rules
  • Frames
  • Semantic Networks
  • Concept Diagrams
  • Ontologies

• *Logic* is used to give a knowledge representation approach precise (formal) semantics: important for computational purposes.

• One have to pick up or create the technique which best fits their needs (in every domain of interest), meaning that it provides the necessary expressiveness, preciseness and comprehensiveness.
Ontologies

• Explicit representation of a domain’s concepts and their relations
• Basic idea: formal and well-formed terminologies, taxonomies and vocabularies
• The represented knowledge encompasses types of entities, attributes and properties, relations and functions, and also various constraints/restrictions
• Each ontology maps to a Description Logic (DL) variant
• Can be regarded as a Terminological Box (TBox) in DLs
• Popular, standardized ontology language is the Web Ontology Language (OWL)
ASCENS knowledge corpuses

Each of the four identified ASCENS knowledge domains maps to a ASCENS Knowledge Corpus, consisting of:

- Domain ontology
- Contexts
- Logical framework
ASCENS knowledge corpuses

Each of the four ASCENS knowledge domains is going to be represented by a top-level domain-specific ontology.

- General concepts, broad semantic interoperability between the low-level ontologies for each ASCENS case study
Knowlang specification model
Every concept tree has a root concept

Explicit concepts must be presented in the knowledge representation of an ASCENS system
Ontology layer II

- autonomic self-adapting behavior provided by *policies*, *events*, *actions*, *situations*, and relations between policies and situations
- cardinality for the policy-situation relationship: *many-to-many*
- *Binary Relations* considered only
Extract the relevant knowledge from the ontology by emphasizing the key concepts -> help the inference mechanism narrow the domain of knowledge by exploring the concept trees down to the emphasized concepts only.

“Clean” knowledge -> more efficient reasoning
Logical Framework

**Facts:** define true statements in the knowledge domains that can be used to discover situations \( \{ \text{can be regarded as an Assertional Box (ABox) in DLs} \)\)

**Rules:** hypothesis, conclusions

**Constraints:** used to validate knowledge, i.e. check its consistency

\[
L_f := \{F_a, R_l, C_t\} \\
F_a := \{f_{a_0}, f_{a_1}, \ldots, f_{a_n}\} \\
f_a := b_f(O) \rightarrow T \\
R_l := \{r_{l_0}, r_{l_1}, \ldots, r_{l_n}\} \\
r_l := \text{if } f_{a_1} \text{ then } f_{a_2} \text{ [else } f_{a_3} \} \\
C_t := \{c_{t_0}, c_{t_1}, \ldots, c_{t_n}\} \\
c_t := \langle \text{if } f_{a_1} \text{ then MUST } f_{a_2} \rangle \mid \langle \text{if } f_{a_1} \text{ then MUST } \neg f_{a_2} \rangle \\
f_{a_1}, f_{a_2} \in F_a
\]
Knowlang specification model

- ASCENS Knowledge Base
  - Knowledge Corporuses
    - Domain Ontology
      - Metaconcepts
        - Concepts
          - Policies
          - Events
          - Actions
          - Situations
          - Groups
        - Concept Trees
        - Explicit Concepts
        - Object Trees
        - Objects
        - Relations
      - Ambient Trees
      - Domain Facts
      - Domain Rules
      - Domain Constraints
    - Contexts
    - Logical Framework
      - Tell
      - Ask
    - KB Operators
      - Inter-ontology Operators
  - Inference Primitives
Knowledge Base Operators

- knowledge-operating mechanism providing for storing, updating and retrieval/querying:
  - Ask operators: retrieve knowledge from the knowledge corpus
  - Tell operators: update the knowledge corpus
  - Inter-ontology operators: merging, mapping, alignment, etc. of one or more ontologies
- **Physical storage** through the KCLAIM language tuple-space mechanism
- Knowledge Base Operators may imply the use of *inference primitives.*
Knowlang specification model
Inference Primitives

- Algorithms for reasoning and knowledge inference
- based on First Order Logic (FOL), First Order Probabilistic Logic (FOPL) and Description Logic (DL) reasoning
- FOPL allows us to make assertions over “likely” features
- Intend to support:
  - induction (FOL) - induct new general knowledge from specific examples
    
    *Example:* Every robot I know has grippers. $\rightarrow$ Robots have grippers.
  - deduction (FOL) – deduct new specific knowledge from more general one
  - abduction (FOPL) – conclude new knowledge based on shared attributes
    
    *Example:* The object was pulled by a robot.
    
    MarXbot has a gripper. $\rightarrow$ MarXbot pulled the object.
  - subsumption (DL) – the act of subsuming a concept by another concept
  - classification (DL) – assessing to which category a given object belongs to
  - recognition (DL) – recognizing an object in the environment
Case study: Robots

Robot SC Ontology: Robot Concept Tree

- They inherit the concepts “Phenomenon”, “Virtual Entity” and “Thing”

Figure taken from Vassev, Hinchey: “Representing Knowledge in Robotic Systems with Knowlang”
Case study: Robots

Robot SC Ontology: Robot Object Tree

- Shows the object properties of the marXbots Robot object

Figure taken from Vassev, Hinchey: “Representing Knowledge in Robotic Systems with Knowlang”
Case study: Robots

Robot Environment Ontology: Environment Concept Tree

- Presents parameters and properties of the robot’s operational environment: external systems, obstacles, measurements etc.

- Apart from the SC Ontology and the Environment Ontology we need to develop the SCE Ontology and the Situational Ontology
Case study: Robots

Robot Contexts and Logical Framework example

- Context applied at runtime to narrow the scope of knowledge
- “Electronics”, “Software” and “Property” generalize all their descendant concepts in this specific context
- “robot cannot move” situation -> possible actions determined by policies or (here) rules

Figure taken from Vassev, Hinchey: “Representing Knowledge in Robotic Systems with Knowlang”
Future work

- Specify the knowledge representation for *all three* ASCENS case studies using Knowlang
- Include toolset for formal validation
Thank you!