Requirements and Initial Model for KnowLang – A Language for Knowledge Representation in Autonomic Service-Component Ensembles

Emil Vassev
Lero – the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
emil.vassev@lero.ie

Mike Hinchey
Lero – the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
mike.hinchey@lero.ie

Benoit Gaudin
Lero – the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
benoit.gaudin@lero.ie

Paddy Nixon
University of Tasmania, Hobart, Tasmania, Australia
paddy.nixon@utas.edu.au

ABSTRACT

Autonomic Service-Component Ensembles (ASCENS) is a class of multi-agent systems formed as mobile, intelligent and open-ended swarms of special autonomic service components capable of local and distributed reasoning. Such components encapsulate rules, constraints and mechanisms for self-adaptation and acquire and process knowledge about themselves, other service components and their environment. ASCENS systems pose distinct challenges for knowledge representation languages. In this paper, we present requirements and an initial model for such a language called KnowLang. KnowLang is intended to provide for formal specification of distinct knowledge models each representing a different knowledge domain of an ASCENS system, such as the internal world of a service component, the world of a service-component ensemble, the surrounding external world and information of special situations related to state changes and operations of service components. KnowLang provides the necessary constructs and mechanisms for specifying such knowledge models at two main levels – an ontology level and a logic-foundations level, where the latter is formed by special facts, rules, constraints and inter-ontology operators. In this paper, we also survey one of the ASCENS case studies to derive some of the requirements for KnowLang.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – Representation languages; Predicate Logic; Representations (procedural and rule-based); D.2.1 [**Software Engineering**]: Requirements/Specifications – Languages; Methodologies; D.2.10 [**Software Engineering**]: Design – Methodologies; D.3.2 [**Programming Languages**]: Language Classifications – Very high-level languages; D.2.11 [**Software Architectures**]: Domain-specific architectures;

General Terms

Algorithms, Design, Experimentation, Languages, Performance

Keywords

knowledge representation, logic, ontology, reasoning, awareness, ASCENS

1. INTRODUCTION

"*Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?*" This famous quote of the American poet and Nobel laureate T.S. Eliot, cited in his poem "The Rock" in 1934, has become inspiration for many scientists studying knowledge at different levels of depth of meaning. One of the most challenging tasks of designing artificial intelligence is how to represent information to a machine that cannot understand a human language. Knowledge representation can be regarded as a *specification* of the system's understanding about itself and its surrounding world. Thus, a knowledge representation model has its *syntax* and *semantics* (facts presenting the knowledge meaning) where both are provided by a special language we use to write knowledge representation. To avoid ambiguity in knowledge facts, such a language should be a formal language that provides mechanisms for verifying the consistency and eventually the correctness of the knowledge representation.

In this paper, we present an overview of the requirements and an initial model for such a formal language called KnowLang, which we are currently developing at Lero - the Irish Software Engineering Research Centre, University of Limerick, Ireland. The KnowlLang language is dedicated to Autonomic Service-Component Ensembles (ASCENS) [1] systems. ASCENS is an FP7 (Seventh Framework Program) [2] project targeting the development of a coherent and integrated set of methods and tools providing a comprehensive development approach to developing *ensembles* (or swarms) of intelligent, self-aware and adaptive *service components*. One of the main scientific contributions that we expect to achieve with ASCENS is related to knowledge representation and knowledge processing for awareness in such systems. Note that it is of major importance for an ASCENS system to acquire and structure comprehensive knowledge in such

a way that it can be effectively and efficiently processed, so such a system becomes aware of itself and its environment.

In the course of this project, KnowLang shall be used to write the knowledge representation for the three ASCENS case studies [1]. The latter present three completely different application scenarios intended to prove the versatility and the expressive power of KnowLang. Those scenarios are:

- Ensemble of self-aware robots;
- Ensemble of e-Vehicles embedded in an e-infrastructure;
- Resource Ensembles as Science Clouds.

The ensemble of robots case study targets swarms of intelligent individual robots with self-awareness capabilities that help the entire swarm acquire the capacity to reason, plan and autonomously act. This shall give the robot swarm self-management capabilities and more goal-oriented and efficient use of resources. The cloud computing case study strives to optimize the use of resources in a unified effort to improve utilization and obtain a higher throughput in the cloud computing setting. By adding a notion of *dynamic self-awareness* and *aware-rich optimization*, the overall use of resources in clouds shall be significantly improved. With regards to e-Vehicle applications ASCENS aims to optimize the usage of traffic and infrastructure resources in the most efficient and flexible way while taking into account the typical e-Mobility restrictions (range limitation, battery recharge). Note that although being different in terms of application domain, the above described case studies share a class of similar features such as: dynamic behaviour (both autonomous and collective), self-awareness, resource-dependence, local intelligence, etc.

The rest of this paper is organized as follows. Section 2 presents some thoughts about awareness as a starting point of our requirements study. Section 3 presents our vision about the ASCENS Knowledge Corpuses defining the structures for knowledge representation in ASCENS systems. In Section 4, we discuss the initial KnowLang Specification Model and major requirements. Finally, Section 5 concludes the paper with a brief conclusion and future work.

2. AWARENESS AS A STARTING POINT

In the course of this research, in order to elicit the requirements for knowledge representation in ASCENS systems, we focus on awareness as the ultimate goal of knowledge representation and reasoning. Conceptually, awareness is a product of *knowledge representation*, *knowledge processing* (e.g., reasoning) and *monitoring* that build up a mechanism helping a system recognize its state changes, resource consumption, environment activities, etc. [3]. In general, we recognize two kinds of awareness:

- *self-awareness* – a system (or a system component) has detailed knowledge about its own entities, current states, capacity and capabilities, physical connections and ownership relations with other (similar) systems in its environment;
- *context-awareness* – a system (or a system component) knows how to negotiate, communicate and interact with environmental systems (or other components of a system) and how to anticipate environmental system states, situations and changes.

Our initial study of the problem domain where we emphasized the ASCENS case studies (see Section 1) concludes that awareness of a service component (SC) is about entailing the possession of a *complete* self-model that encompasses the SC's functional features, execution history, current situation, activities, abilities, services, goals, knowledge, intentions, etc. In addition, a SC should have a *good* understanding of the ensemble it is a member of in terms of common goals, ensemble states, communication mechanisms and interfaces, other SCs it works with, etc. Finally, a SC should have a rather *general* understanding of its operational environment in terms of concepts, objects, events or situations. All these awareness aspects should be empowered by structured knowledge and autonomic reasoning and goal-directed (or utility-directed) planning abilities.

3. ASCENS KNOWLEDGE CORPUSES

The initial research results [4] based on our awareness study concluded that a SC should have structured knowledge addressing the SC's structure and behaviour, the SC Ensemble (SCE) structure and behaviour, the environment entities and behaviour and situations where that SC or the entire SCE might end up in. Based on these knowledge requirements, we defined four *knowledge domains* (*corpuses of knowledge*) [4]:

- *SC Knowledge Corpus* – knowledge about internal configuration, resource usage, content, behaviour, services, goals, communication ports, actions, events, metrics, etc.;
- *SCE Knowledge Corpus* – knowledge about the whole system, e.g., architecture topology, structure, system-level goals and services, behaviour, communication links, public interfaces, system-level events, group actions, etc.;
- *Environment Knowledge Corpus* – parameters and properties of the operational environment, e.g., external systems, concepts, objects, external communication interfaces, integration with other systems, etc.;
- *Situational Knowledge Corpus* – specific situations, involving one or more SCs and eventually the environment.

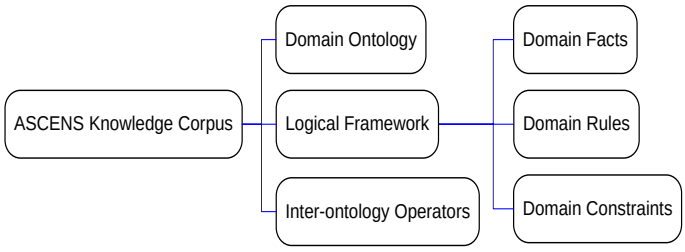

Figure 1. ASCENS knowledge corpus

Every ASCENS Knowledge Corpus is structured into a *domain-specific ontology* [5], *logical framework* and *inter-ontology operators* (see Figure 1). The domain-specific ontology gives a *formal* and *declarative representation* of the knowledge domain in terms of explicitly described domain concepts, individuals and the relationships between those concepts/individuals. The *logical framework* helps us realize the explicit representation of particular and general factual knowledge, in terms of predicates, names, connectives, quantifiers, and identity. Thus, the logical framework provides additional to the domain ontology computational structures that basically determine the logical foundations helping a SC reason and infer knowledge. As shown in Figure 1, a logical framework consists of *facts*, *rules* and *constraints* – all logically founded and built with ontology terms:

- *facts* – define true statements in the knowledge domains that can be used to discover situations;
- *rules* – express knowledge such as: 1) *if H than C*; or 2) *if H than C1 else C2*; where *H* is hypothesis of the rule and *C* is the conclusion of the rule;
- *constraints* – used to validate knowledge, i.e., to check its consistency. Can be *positive* or *negative* and express knowledge of the form: 1) *if A holds, so must B*; or 2) *if A holds B must not*.

The "Inter-ontology Operators" knowledge category defines logical operators that work on multiple ASCENS ontologies, e.g., *merging*, *mapping*, *alignment*, etc. Note that some knowledge can be shared among knowledge corpuses.

3.1 ASCENS Knowledge Base

All the four ASCENS Knowledge Corpuses - SC Knowledge Corpus, SCE Knowledge Corpus, Environment Knowledge Corpus and Situational Knowledge Corpus, form together the so-called ASCENS Knowledge Base (AKB). An AKB is hosted by a SC as a sort of *knowledge database* where knowledge is organized into ASCENS Knowledge Corpuses. In addition, an AKB must provide a *knowledge-operating mechanism* for knowledge *storing*, *updating* and *retrieval/querying*. Ideally, we can think of an AKB as a black box whose interface consists of two methods called TELL and ASK. TELL is used to add new sentences to the knowledge base and ASK can be used to query information. Both methods may involve knowledge inference and therefore, an AKB should be equipped with a special *Inference Engine* that is going to reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions, i.e., inferring new knowledge. As part of the awareness process, the Inference Engine should also imply both deterministic and probabilistic algorithms for awareness. For example in [4], we proposed a probabilistic algorithm for *awareness self-initiation* that helps a SC process its knowledge and become aware about changes in the SC ensemble or the environment. The algorithm is based on Partially Observable Markov Decision Processes (POMDP) [6].

3.2 ASCENS Ontologies

Every ASCENS domain-specific ontology is intended to give an explicit representation of the concepts and objects, together with their relationships, forming a knowledge domain. In the course of this project, we need to develop all the four knowledge corpuses for each ASCENS case study – swarm robotics, eMobility and science clouds (see Section 1). Here, we need to build a distinct ontology per knowledge corpus and per case study.

To fulfill this requirement, we are going to build four *ASCENS top-level ontologies*, where each will conceptually represent one of the four ASCENS knowledge domains. These top-level ontologies will help us describe general knowledge concepts that are the same across the three problem domains covered by the ASCENS case studies. The targeted *top-level ontologies* are intended to support very broad semantic interoperability between the *low-level ontologies* we need to further build for each one of the ASCENS case studies. Thus, the ASCENS top-level ontologies should be generic enough to cover all the lexical domains provided by these case studies. Figure 2 depicts a generic scheme for ASCENS ontologies. As shown, the ASCENS ontologies will provide the symbolic representation of conceptual classes (e.g., Resources, Entities, etc.), objects, features of classes and relationships between classes.

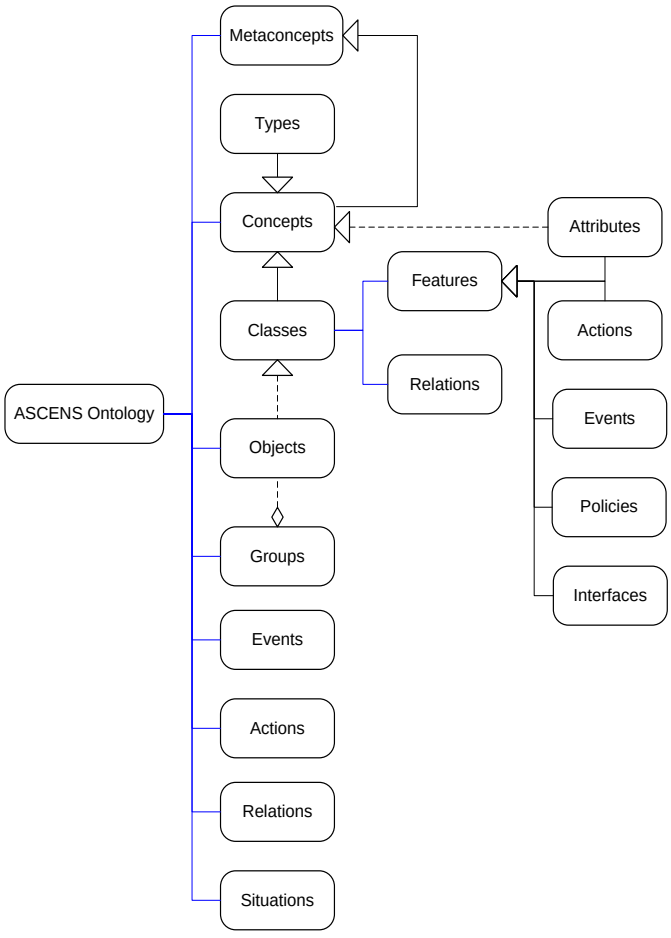

Figure 2. ASCENS ontology scheme

A major research-directing requirement is that KnowLang should incorporate all the necessary specification mechanisms and constructs needed to specify knowledge in the format imposed by the ASCENS Knowledge Corpuses shown in this section. Another major requirement is that KnowLang should be developed taking

into account the awareness principles described in Section 2. In the following Section, we develop further those two major requirements to derive more concrete requirements for KnowLang – the language for knowledge representation in ASCENS systems.

4. REQUIREMENTS FOR KNOWLANG

KnowLang is a formal language providing a comprehensive specification model that must be able to address all the aspects of an ASCENS Knowledge Corpus and eventually some of the ASCENS Knowledge Base Inference Engine.

4.1 KnowLang Specification Model

The complexity of the problem of knowledge representation necessitates the use of a specification model where knowledge can be presented at different levels of depth of meaning. Thus, KnowLang shall impose an ASSL-like multi-tier specification model [7] where we specify the ASCENS Knowledge Corpuses at different *knowledge tiers* nesting other tiers hosting the specification of a *Domain Ontology*, a *Logical Framework* and *Inter-ontology Operators*.

Figure 3 show the preliminary KnowLang Multi-tier Specification Model. As shown, to help us specify an AKB (ASCENS Knowledge Base), KnowLang should provide *constructs* not only for the specification of the ASCENS Knowledge Corpuses, but also constructs for special Knowledge Base Operators (KB Operators) and Inference Primitives. The KB Operators should provide for a means for *storing*, *updating* and *retrieval/querying* knowledge. To store physically an AKB we are going to use a special tuple-space mechanism provided by the KLAIM language [8]. Thus, the KB Operators should cope with the KLAIM mechanisms for reading and writing data from and to a KLAIM tuple space. The Inference Primitives should provide mechanisms for reasoning and knowledge inference. We intend to use inference techniques related to both First Order Logic (FOL) Reasoning [9] and Description Logic (DL) Reasoning [10]. Thus, the Inference Primitives should address the following inference techniques:

- induction (FOL) - induct new general knowledge from specific examples;

  Example: *Every robot I know has grippers.* → *Robots have grippers.*

- deduction (FOL) – deduct new specific knowledge from more general one;

  Example: *Robots can move. MarXbot is a robot.* → *MarXbot can move.*

- abduction (FOL) – conclude new knowledge based on shared attributes.

  Example: *The object was pulled by a robot.*

  MarXbot has a gripper. → *MarXbot pulled the object.*

- subsumption (DL) – the act of subsuming a concept by another concept;

  Example: *Exploit the taxonomy structure of concepts that are defined in the ontology and compute a new taxonomy for a set of concepts or derive matching statement from computed generalization/specialization relationships between task and query.*

- classification (DL) – assessing to which category a given object belongs to;

- recognition (DL) – recognizing an object in the environment.

Note that KnowLang should provide syntax and semantics rules for each tier (see Figure 3). A preliminary and incomplete KnowLan syntax is the following grammar presented in Extended Backus-Naur Form [11]. The KnowLang context-free grammar specification is obtained by the reduction of the (*AKB_Spec* → ***bof*** *Knowledge_Corpuses KB_Operators Inference_Primitives* ***eof***) rule. Therefore, an AKB (ASCENS Knowledge Base) specified with KnowLang consists of ASCENS Knowledge Corpuses (see Section 3), Knowledge Base Operators and Inference Primitives. As shown, a *domain ontology* is composed of *concepts* and optional *metaconcepts*, *objects*, *groups*, *events*, *actions*, *relations* and *situations*. The individual metaconcepts, types, classes, objects, situations, etc., are organized in sets where the individual members are distinguished by their name. Moreover, a *class* might be instantiated from a *metaconcept*, *concepts* might be *types* and classes, *objects* are instantiated from *classes* and have *implemented features* and there could be *groups* of related *objects*.

AKB_Spec → ***bof*** *Knowledge_Corpuses KB_Operators*
    Inference_Primitives ***eof***
Knowledge_Corpuses → *Domain_Ontology Logical_Framework*
    Inter_Ontology_Operators
Domain_Ontology → *Metaconcept* Concept+ Object* Group* Event**
    Action Relation* Situation**
Metaconcept → ***METACONCEPT*** *metaconcept_name* ***{*** *Feature+* ***}***
Concept → *Type* | *Class*
Type → ***TYPE*** *type_name*
Class → ***CLASS*** *class_name*
    <***INSTANCE_OF METACONCEPT***.*metaconcept_name*>?
    { *Feature+ Relation** ***}***
Feature → *Attribute* | *Action* | *Event* | *Policy* | *Interface*
Object → ***OBJECT*** *object_name* ***INSTANCE_OF CLASS***.*class_name*
    { *Impl_Feature+* ***}***
Group → ***GROUP*** *group_name* ***{*** *Object+ Relation** ***}***
Situation → ***SITUATION*** *situation_name* ***{*** *Fluent History Go_Actions* ***}***
Logical_Framework → *Domain_Fact* Domain_Rule* Domain_Constraint**
Inter_Ontology_Operators → *Inter_Ontology_Operator**
KB_Operators → *Storing_Op+* | *Updating_Op+* | *Querying_Op+*
Inference_Primitives → *Induction_Pr+* | *Deduction_Pr+* | *Abduction_Pr+* |
    Subsumption_Pr+ | *Classification_Pr+* | *Recognition_Pr+*

4.2 KnowLang Formalism

It is difficult to determine the formalism of KnowLang at this stage of the work, but an initial requirement is that it should be a declarative language. Going further, we can conclude that the underlying formalism for the ontology part should be Description Logic (DL) [10] and for the *Logical Framework* it should be First Order Logic (FOL) [9] or a derivation of the same. The Inference Primitives should be both FOL and DL based. Formal semantics and reasoning about KnowLang *ontologies* might be achieved by mapping the ontology to Description Logic and using the established and well-studied inference procedures for Description Logic to implement reasoning.

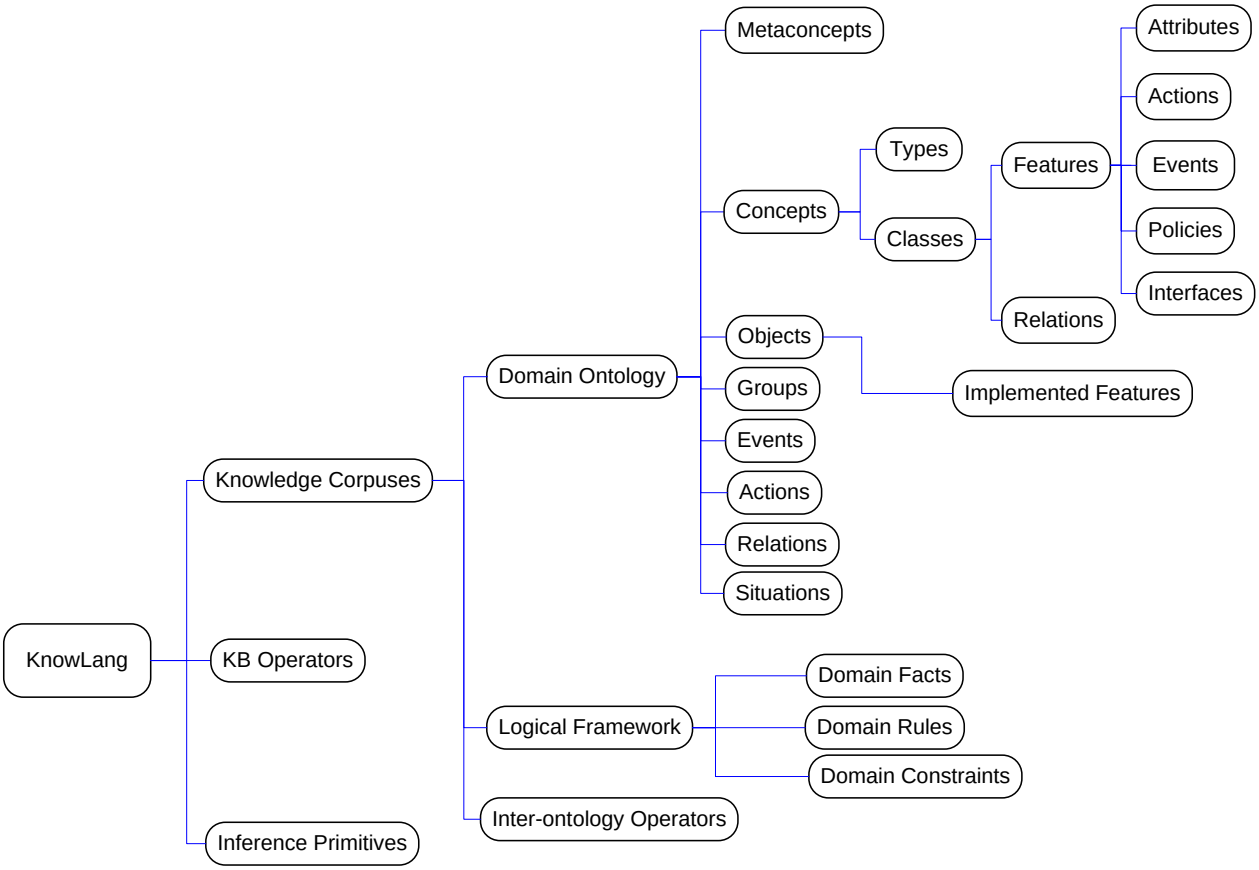

Figure 3. KnowLang multi-tier knowledge specification model

4.2.1 Ontologies

Being an ontology language, KnowLang should provide efficient constructs for encoding the ASCENS ontologies. The semantics for this part of KnowLang should be DL-based and open-world. Note that DL imposes open-world semantics, which means that if a statement can neither be proven to be true or false it is not judged as false, but as unknown.

In general, ontologies provide a form of explicit representation of domain concepts and the relationships between those concepts to form the basic structure around which knowledge can be built. The main idea is to establish standard models, taxonomies, vocabularies and domain terminology and use those to develop appropriate knowledge and reasoning models. Any ontology is a formal and declarative representation of a knowledge model of some topic or subject area. It provides concepts to be used for expressing knowledge in that subject area. This knowledge encompasses: *types of entities*, *attributes* and *properties*, *relations* and *functions*, as well as *various constraints*.

4.2.2 Distributed Reasoning

Ideally, an ASCENS system must support *distributed reasoning*, and thus, KnowLang should provide mechanisms for that. Although, the main reasoning mechanisms will be implied by the Logical Framework and the Inference Primitives, the global *actions*, *events* and *situations* are also intended to provide support for distributed reasoning at the ontology level.

4.2.3 Logical Framework

As we mentioned above, the formal semantics of Logical Framework (*facts*, *rules* and *constraints*) will be logic based - Second-Order Logic (SOL) [12] or FOL [9] where complexity issues make the application of SOL infeasible. SOL is more expressive than the FOL. The problem with FOL is that we may quantify over individuals, but not over properties, e.g., we can find the individuals of a property, but we cannot find the properties of an individual. For example, with SOL we can axiomatize the sentence "*SC1 and SC2 have at least one property in common, e.g., share at least one interface*", which cannot be done with FOL. Here, the SOL formula is:

$$\exists P \, (P(sc1) \wedge P(sc2))$$

However, the nature of some of the ASCENS Knowledge Corpuses (e.g., SC Ensemble Knowledge Corpus and Environment Knowledge Corpus) calls for a probabilistic Logical Framework [4]. Therefore, besides the resources of FOL/SOL we shall enrich the FOL/SOL formalism with generalized quantifiers (including ones like "*most*" and "*usually*") and predicate and sentence modifiers (such as "*probably"*). Moreover, for the formal semantics of some of the *Constraints*, we might adapt the

so-called Concurrent Constraint pi-Calculus [13] approach introduced as a model for concluding service-level agreements.

4.2.4 Events & Situations

An important requirement for KnowLang is the need of constructs that give some means for associating event/situation terms with event/situation-describing sentences, so that the described events and situations can be referred to, temporally modified, causally related, or otherwise qualified. The formal semantics for KnowLang Situations might be borrowed from the so-called *Situation Calculus* [14] or eventually a probabilistic extension of the same [15]. The Situation Calculus is a logic formalism based on SOL and designed for representing dynamic domains. It also, is very appropriate for various sorts of reasoning, including planning. Basically, Situation Calculus represents changing scenarios as a set of SOL formulae where the basic elements are:

- *Actions* that can be performed in the world*:*
- *Fluents* that describe the state of the world.
- *Situations* that represent a history of action occurrences*.*

The Situation Calculus has shown to be more expressive than it had been initially assumed with respect to concurrent, extended, and temporally qualified actions as well as causation [16].

Another formalism that is appropriate for KnowLang Situations is Event Calculus, which is introduced by Kowalski and Sergot [17] as a theoretical framework where it is possible to reason about events in event-driven systems. Event Calculus is defined over a set of *events* taking effect at specific time points and *fluents* representing the effects of the events. Along with the basic entities of events and fluents, it also defines a set of predicates allowing the specification of propositions about fluents, events and time points. The basic event calculus predicates are:

- Initiates (e, f, t) - fluent f is initiated be event e at time t;
- Terminates (e, f, t) - fluent f is terminated be event e at time t;
- Happens (e, t) - event e occurs at time t;

4.2.5 Policies

To specify the behaviour of a SC in important situations, KnowLang shall provide constructs for specifying special Policies (specified as part of a Domain Ontology, at the level of class's features – see Figure 3). The formal semantics for KnowLang Policies shall be borrowed from the semantics of self-managing policies specified in ASSL (Autonomic System Specification Language) [7]. The ASSL Policies specify special self-managing behaviour driven by special *fluents* and *actions*. A fluent presents a state where the system gets into when special conditions are met. Such conditions are driven by events. If a SC gets into a fluent then the policy behind that fluent is considered active and specific actions are executed.

4.3 KnowLang Base Operators

KnowLang shall provide appropriate constructs and mechanism for *writing*, *querying* and *updating* ASCENS Knowledge Corpuses.

4.3.1 Experience

KnowLang shall provide appropriate mechanisms and constructs to *store* and *query* SC's *experience* of actions' executions. This will help a SC be aware of the execution history of the actions to be executed and eventually compute the success probability for those actions. In that way, a SC may learn (infer new knowledge) not to execute actions that traditionally have low success rate.

4.3.2 Experience Abstraction

Experiences play a central role in the learning process. For example, an experience can be observed while the robot is acting and then eventually *abstracted* and stored in the ASCENS Knowledge Base. Thus, KnowLang should provide additional constructs allowing for experience abstraction.

4.4 Robot-Related Requirements

In this section, we conclude some KnowLang requirements specific to the swarm robotics ASCENS case study [1] using the marXbot robotics platform [18]. Note that similar requirements should be deducted for the other two case studies.

4.4.1 The marXbot Robot

The marXbot is a modular research robot equipped with a set of devices that help the robot interact with other robots of the swarm or the robotic environment. Figure 4 shows a marXbot robot equipped with a set of devices to interact with the environment and with other robots of the swarm [18] - a light sensor, a distance scanner, a range communication system, a gripper and wheels moving the robot.

Russel and Norvig [19] define a robot to be "a physical agent that performs tasks by manipulating the physical world and that has sensors and effectors". In this section, we analyze the requirements for knowledge representation for marXbot robots by keeping in mind this definition and examining several aspects in more detail.

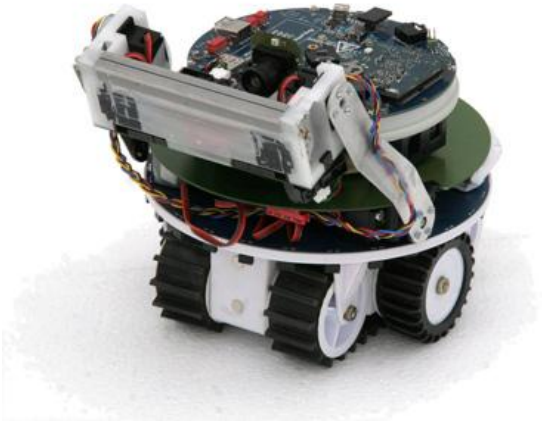

Figure 4. A marXbot robot [18]

4.4.2 PEAS Description

First we need to define the term "agent". In general, agent can be defined as *entity* that perceives its environment through *sensors* and acts upon that environment through *actuators*. A marXbot robot might be regarded as a *rational agent*. A rational agent is an

agent that “maximizes a certain performance measure given the knowledge of the world that the agent acquired by the historical percept sequence and predefined built-in knowledge” [19]. A good technique for a description of a rational agent is the so-called PEAS (*Performance*, *Environment*, *Actuators* and *Sensors*) technique, i.e., a description emphasizing performance measurement, environment, actuators and sensors of a robot. Therefore, KnowLang shall support PEAS Description at the level of Domain Ontology of the SC Knowledge Corpus.

4.4.3 Kinematic Chain

An important way to describe the structure of a robot is to describe its kinematic chain consisting of rigid links (similar to bones in human skeleton) and joints (similar to joints in human skeleton). A joint always connects two links and specifies the type of motion these two links can perform against each other. Together, links and joints provide and limit the motions that a robot is able to execute and the degrees of freedom a robot possesses. As links and joints form a hierarchical, tree like structure as each link can have only one parent link but several child links, this structure is called “kinematic chain”. The kinematic chain description of a robot can be used to compute the required pose of all links and joints in order to place the last link in the chain (e.g. the gripper) at a specific position in space (“inverse kinematics”). Or, the current position of the last link in the chain can be computed using current poses of all links and joints (“forward kinematics”). KnowLang should provide support for Kinematic Chain Description.

4.4.4 Localization

The awareness must help a robot impose the ability to determine where things are. This ability is called “localization” and it is especially challenging in dynamic environments where objects may move and change their position. Here KnowLang must have constructs for representing the localization ability of a SC and also constructs for maintaining tracking of objects.

4.4.5 Map

The ability to localize objects and to asses relatively the environment requires a map of the environment where the robot operates. Thus, KnowLang must provide constructs for describing a map of the operational environment which shall be used by a SC to localize its position and move around by navigating with this map.

4.4.6 Planning

All these high level abilities require the robot to use its sensors and effectors in a coordinated and reasonable way. Consequently, there must be a kind of controller entity that controls the single sensor and effector actions and coordinates them. This ability to “come up with a sequence of actions (where each action is a sensor or effector action) that will achieve a goal” is called “planning”. KnowLang should provide constructs for planning eventually specified with KnowLang Policies.

5. CONCLUSION

In this paper, we have presented an initial overview of KnowLang, a formal language for knowledge representation in ASCENS (Autonomic Service-Component ENSemble) systems. KnowLang provides a multi-tier specification model for specifying ASCENS Knowledge Bases comprising special ASCENS Knowledge Corpuses, Knowledge Base Operators and Inference Primitives. Further, an ASCENS Knowledge Corpus is specified at additional depth of meaning where we distinguish a Domain Ontology, a Logical Framework and Inter-ontology Operators. In addition, we have also presented major KnowLang requirements related to the underlying formalisms, knowledge base operators and the knowledge representation in swarm robotics systems, one of the ASCENS case studies. Note that this approach requires comprehensive ontological design and careful control on the ontology content.

To conclude the paper, it should be noted that an essential assumption when building knowledge models is that such cannot provide a complete picture of the domain of interest. The fundamental reasons are that domain objects often present real things that cannot be described by a finite set of symbolic structures. Moreover, such objects do not exist in isolation, but are included in unlimited sets of encompassing contexts. Therefore, *incompleteness* shall be considered when representing knowledge and also the fact that an intelligent system must rely on reasoning to infer missing knowledge. In general, *incompleteness in knowledge* can be regarded as a limitation of the meaning of the concepts of truth. Just like the concepts of "speed" and "time" are concepts of limited range, knowledge is also of limited range and an ASCENS system must deal with limits on the information that can be conveyed about the entire ensemble (e.g., current state) and surrounding environment. Clearly, a complex structure like ASCENS cannot convey all information about both itself and the rest of the world. Therefore, the system must just function with the incompleteness of its knowledge and use the reasoning mechanisms to make up for this.

Future work is mainly concerned with further development of the ASCENS knowledge models and algorithms for knowledge processing and development of KnowLang and its accompanying tools. Moreover, we shall use KnowLang to develop the knowledge base for the three ASCENS case studies and experiment with those.

6. ACKNOWLEDGMENTS

This work was supported in part by an IRCSET postdoctoral fellowship grant at University College Dublin, Ireland and by the Science Foundation Ireland grant 03/CE2/I303 1 to Lero, the Irish Software Engineering Research Centre.

7. REFERENCES

[1] ASCENS – Autonomic Service-Component Ensembles. 2010. http://www.ascens-ist.eu/.

[2] European Commission – CORDIS. *Seventh Framework Program (FP7)*. http://cordis.europa.eu/fp7/home_en.html.

[3] Vassev, E. and Hinchey, M. 2010. The Challenge of Developing Autonomic Systems. *IEEE Computer*. 43, 12 (December 2010), pp. 93–96.

[4] Vassev, E. and Hinchey. M. 2011. Knowledge Representation and Awareness in Autonomic Service-Component Ensembles – State of the Art. In *Proceedings of the 14th IEEE International Symposium on Object/Component/ Service-oriented Real-time Distributed Computing Workshops*. IEEE Computer Society. pp. 110–119.

[5] Swartout, W. and Tate, A. 1999. Ontologies. *IEEE Intelligent Systems*. 14 (1999), pp. 18–19.

[6] Littman, M. L. 1996. *Algorithms for Sequential Decision Making*. PhD Thesis, Department of Computer Science, Brown University.

[7] Vassev, E. 2008. *Towards a Framework for Specification and Code Generation of Autonomic Systems*. PhD Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada.

[8] De Nicola, R., Ferrari, G. L., Pugliese, R. 1998. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*. 24, 5 (1998), pp. 315–330.

[9] Brachman, R. J. and Levesque, H. J. 2004. *Knowledge representation and reasoning*. Elsevier, San Francisco.

[10] Baader, F. and Nutt, W. 2002. Basic Description Logics. In *The Description Logic Handbook*, F. Baader, D. Calvanese D. McGuinness, D. Nardi and P. Patel-Schneider, Ed.

[11] Knuth, D. E. 1964. Backus Normal Form vs. Backus Naur Form. *Communications of the ACM*. 7, 12 (1964), pp. 735–73.

[12] Troelstra, A. S. and Schwichtenberg, H. 2000. *Basic Proof Theory*. Cambridge University Press.

[13] Buscemi, M. G. and Montanari, U. 2008. Open Bisimulation for the Concurrent Constraint Pi-Calculus. In *Programming Languages and Systems, Lecture Notes in Computer Science*. 4960 (2008), pp. 254–268.

[14] McCarthy, J. and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence 4*. Edinburgh Univ. Press. pp. 463–502.

[15] Mateus, P., Pacheco, A., Pinto, J., Sernadas, A. and Sernadas, C. 2001. Probabilistic Situation Calculus. *Annals of Mathematics and Artificial Intelligence*. 32, 1-4 (2001), pp. 393–431.

[16] Schubert, L. K. 1990. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. *Knowledge Representation and Defeasible Reasoning*. Dortrecht: Kluwer. pp. 23–67.

[17] Kowalsky, R. and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing*. 4, 1 (1986), pp. 67–95.

[18] Bonani, M., Longchamp, V., Magnenat, S., Retornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H. and Mondada, F. 2010. The MarXbot, a Miniature Mobile Robot Opening new Perspectives for the Collective-robotic Research. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*.

[19] Russell, S.J. and Norvig, P. 2009. *Artificial intelligence: A modern approach*. Prentice Hall.