2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops

Knowledge Representation and Awareness in Autonomic Service-Component Ensembles – State of the Art

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

Abstract — **Knowledge is the source of intelligence and both knowledge representation and knowledge management are crucial for intelligent systems. Well employed knowledge helps such systems become aware of situations, recognize states and eventually respond to changes. This paper presents our vision of knowledge representation and awareness in mobile swarm systems formed as open-ended ensembles of special autonomic components. Such components encapsulate rules, constraints and mechanisms for self-management and acquire and process knowledge about themselves, other service components and their environment. In this paper, we present our approach to high-level model of structured knowledge and a formal model of awareness in such autonomic service-component ensembles.**

Keywords - knowledge representation; awareness; ASCENS.

I. INTRODUCTION

In general, an intelligent system is intended to possess self-awareness capabilities based on well structured knowledge and algorithms operating over the same. Knowledge representation and management is one of the important aspects of developing intelligent systems. Knowledge helps systems achieve awareness and autonomic behavior, where the more knowledgeable systems are, the closer we get to real intelligent systems. Here, the term "knowledge" is widely used in practice assuming rather vague distinctions among *data*, *information*, and *intelligence*. Along with such a context, any discussion of knowledge should refer to those categories. By its nature as the source of intelligence, knowledge allows system to recognize its states and helps to decide how to respond to situations.

Autonomic Service-Component ENSembles (ASCENS) [1] 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 self-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 this paper we survey different approaches to knowledge modeling and representation for intelligent systems and present our vision of structured knowledge and knowledge processing for awareness in ASCENS systems. We present a high-level knowledge structure and a formal algorithm for self-initiation based on knowledge processing and awareness. To illustrate the algorithm, we employ it in the ASCENS case study on swarm robotics [1].

The rest of this paper is organized as follows. Section II presents knowledge from a computer perspective and knowledge diversification for ASCENS. Section III surveys knowledge representation approaches and presents our vision of high-level knowledge models for ASCENS systems. In Section IV, we discuss awareness as a key factor to the development of advanced intelligent systems and present our awareness model for ASCENS systems. Section V, presents formally an algorithm for self-initiation of aware service components. This algorithm is illustrated with a swarm robotics cases study. Finally, Section VI concludes the paper with a brief conclusion on knowledge representation and modeling for ASCENS systems and future work.

II. KNOWLEDGE AND KINDS OF KNOWLEDGE

"*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. Such a vision is shared by Pejman Makhfi [3] who concludes that the concept of intelligence is built upon four fundamental principles: *data*, *information*, *knowledge* and *wisdom*. In this quartet, the basic compound for intelligence is *data*. In general, data takes the form of measures and representations of the internal and external worlds of a system, e.g., raw facts and numbers. Information is derived from data by assigning meaning to it relevant to domains of interest, e.g., data in a specific context. Knowledge is a specific interpretation of information, and wisdom is based on awareness, judgment rules, and principles to construct new knowledge from existing one.

A. Kinds of Knowledge

There are many kinds of knowledge that need to be considered for the development of intelligent systems, and

978-0-7695-4377-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ISORCW.2011.21

ASCENS in particular. Conceptually, knowledge can be regarded as a *large complex aggregation* [4] composed of constituent parts representing knowledge of different kind. Each kind of knowledge may be used to derive knowledge models of specific domains of interest. For example, in [4] the following kinds of knowledge are considered:

- *domain knowledge* – refers to the application domain facts, theories, and heuristics;
- *control knowledge* – describes problem-solving strategies, functional models, etc.;
- *explanatory knowledge* – defines rules and explanations of the system's reasoning process, as well as the way they are generated.
- *system knowledge* – describes data contents and structure, pointers to the implementation of useful algorithms needed to process both data and knowledge, etc. System knowledge also may define user models and strategies for communication with users.

Moreover being considered as essential system and environment information, knowledge may be classified as 1) *internal knowledge* - knowledge about the system itself; and 2) *external knowledge* - knowledge about the system environment. Another knowledge classification could consider *a priori knowledge* (knowledge initially given to a system) and *experience knowledge* (knowledge gained from analysis of tasks performed during the lifetime of a system).

B. Knowledge Models for ASCENS

Considering the problem domain addressed by ASCENS, we determined four basic kinds of knowledge in ASCENS systems, i.e., knowledge specific to:

- the individual component structure and behavior;
- the system structure and behavior;
- the environment structure and behavior;
- situations where the system might end up in.

These four kinds of knowledge helped us derive distinct *knowledge models* for ASCENS forming a *high-level knowledge structure* that is to be maintained by any *service component* (SC) member of a *service-component ensemble* (SCE):

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

By representing the knowledge in such models we shall allow the SC's control mechanisms query information about both the SC itself and the SCE, by considering the environment's parameters and properties. Moreover, this shall help SCs understand and reason about themselves and discover situations through the use of probabilistic methods working over the knowledge modeled as situational knowledge patterns.

III. KNOWLEDGE REPRESENTATION

Different knowledge representation techniques may be used to represent different kinds of knowledge. In general, to build a knowledge model we need specific knowledge elements. The latter may be primitives such as *frames*, *rules*, *logical expressions*, etc. Knowledge primitives may be combined together to represent more complex knowledge elements. A knowledge model may classify knowledge elements by type and group those of the same type into collections [4].

Different approaches to knowledge modeling and representation have been developed for intelligent systems. Note that it is important to structure the knowledge in such a way that it can be effectively processed by an intelligent system and perceived and update by humans. The following subsections present some of the most popular approaches to knowledge representation [5].

A. Rules

Rules can be easily understood by humans. By its nature, rules structure knowledge in the form of *attribute-value pairs*. In general, rules may take the following form [5]:

```
if attribute A1 has value V1
and attribute A2 has value V2
then attribute A3 has value V3
```

Shortliffe successfully applied the rule-based approach to the development of systems applying human knowledge and function at the level of a human expert [6]. In this approach, attributes may represent internal data and both system input and output (e.g., a response from the user). In such a knowledge model it is relatively easy to construct an engine that uses the set of rules in an automated reasoning system. Rules may be dynamic, i.e., they can be archived and updated as necessary.

The so-called exception systems use a similar approach to knowledge representation. Here, the rules may take the following form [5]:

```
attribute A1 has value V1
unless attribute A2 has value V2
and attribute A3 has value V3
```

B. Frames

Frames are another approach to knowledge representation understandable by humans. Frame-based knowledge models represent simple concepts via a collection of information and associated actions. An example of a simple representation of a person with the frame approach is the following [5]:

Frame: Ellery Stone
Specialization of: Frame Person
Date of Birth: 30:04:62
Sex: Male
Nationality: British
Home Town: St. Helens
Occupation: Tailor
Health: (Consult Medical system)

Frames combine information, calls to information derivation functions and output assignment. As shown in the simple frame above, some of the slots have associated values and one slot refers to another system that must be used to find a value.

C. Semantic Networks

The third approach to knowledge representation is termed as *semantic networks* [7]. Similar to the previous two approaches, semantic networks provide a knowledge representation that humans can easily cope with. A semantic network is a directed graph consisting of nodes (or vertices) connected with edges (or arcs). Here, nodes represent concepts and edges represent semantic relations between the concepts. There is no standard set of relations between concepts used in semantic networks, but the following relations are very common:

- *instance*: X is an *instance* of Y if X is a specific example of the general concept Y.
  Example: Object A is an *instance* of Class B
- *isa*: X *isa* Y if X is a subset of the more general concept Y (see Figure 1).
  Example: sparrow *isa* bird
- *haspart*: X *haspart* Y if the concept Y is a part of the concept X.
  Example: sparrow *haspart* tail

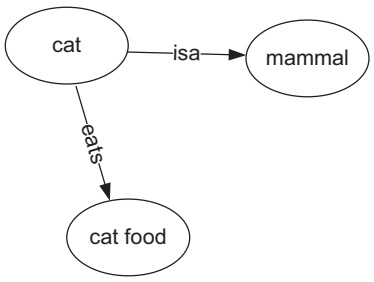

Figure 1. Semantic Network Example

Essentially, a computer-based semantic network uses metadata (data describing data) in order to represent the meaning of different information. Here, metadata helps an intelligent system understand the meaning of information. Note that systems able to recognize the meaning of information (e.g., stored in a data warehouse) become immeasurably more intelligent. Extensible Markup Language (XML) and Resource Description Framework (RDF) are content-management approaches supporting semantic networks. XML and RDF are able to sort through vast amounts of data automatically, recognizing relationships between information and presenting high-quality, relevant data to the user on demand.

A special form of a semantic network is the *semantic web*. The semantic web is a collaborative effort led by World Wide Web Consortium (W3C) that aims to transform the way we find information stored on the Internet. Rather than search for information containing specific keywords we will be able to search the semantic meaning of the content, thus helping search engines retrieve much more relevant information.

D. Concept Diagrams

Another approach to knowledge representation is the so-called *concept diagrams*. By their nature, concept diagrams are very similar to semantic networks. Hence, they also consist of nodes and arcs and the nodes and arcs have similar functions. However, concept diagrams are considered more powerful, because they can describe fairly complex concepts. The Institute for Human and Machine Cognition (IHMC), Florida, USA developed a special form of concept diagram to represent a domain of knowledge [8, 9]. The knowledge embedded in a knowledge model is structured in the form of *concept maps* [10]. Concept maps are graphs that are comprised of concepts on the nodes and the relationships among the concepts on the arcs. Concept maps are used to form knowledge models by placing them in a hierarchical organization and appending special elaborating media onto the nodes within each map. The entire knowledge model is linked together through a general, subsuming top-level map. The result is a model of expert knowledge that contains numerous of domain concepts, principles, and relations.

E. Ontologies

Our initial investigation has shown that in order to model aspects of ASCENS knowledge (see Section II.B), a more expressive and comprehensive knowledge representation approach is required. Ontologies inherit the basic concepts provided by rules (see Section III.A), frames (see Section III.B), semantic networks (see Section III.C) and concept diagrams (see Section III.D) and 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 [11]. The main idea is to establish *standard models*, *taxonomies*, *vocabularies* and *domain terminology*, and use those to develop appropriate knowledge and reasoning models. Such models may be used as reusable components for assembling knowledge-based systems, e.g., multi-agent systems. 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*.

In general, to build ontology, a special ontology language is required. For example, the Web Ontology Language (OWL) [12, 13] is such a formal language that evolved out of Description Logic and is the result of research efforts aiming at providing a knowledge representation language for the semantic web. By using OWL, we build an ontology as an explicit and formal specification of a conceptualization

that eventually can be compared with a TBox in Description Logic [14], i.e., it provides a vocabulary of concepts definitions and defines relationships among these concepts. These concepts in turn are used to represent knowledge about specific objects in the world.

Another ontology language is CycL [15], which is considered as an extension of First Order Logic. CycL has been used to build the Cyc ontology [15], which is structured into different layers and consists of terms constituting the ontology vocabulary, and assertions that represent relationships between terms. These assertions include both simple ground assertions and rules.

F. Logic

To give a knowledge representation approach (e.g., Semantic Networks, Rules, etc.) a precise semantics, it is often formalized using logic [16]. Without a precise formalisation a knowledge representation is vague and ambiguous, and thus, not appropriate for computational purposes. Moreover, logic is relevant to *reasoning*, which is about inferring new knowledge from the existing one, which in logic is relevant to logical entailment and logical deduction [16]. The most prominent logical formalism used for knowledge representation is the *First-Order Predicate Calculus* or also called *First-Order Logic* (FOL). FOL helps us describe a knowledge domain as consisting of *objects* and construct *logical formulas* around those objects. Such formulas are formed by *predicates*, *functions*, *variables*, and *logical connectives* [16]. Similar to semantic networks, statements in natural language can be expressed with logical formulas describing facts about objects with an appropriate choice of *predicate* and *function symbols*. The following example illustrates the use of FOL for knowledge representation by axiomatizing the Semantic Network example from Figure 1:

$$\forall x : (\text{Cat}(x) \rightarrow \text{Mammal}(x))$$
$$\forall x,y : (\text{eats}(x,y) \rightarrow \text{Cat}(x) \wedge \text{CatFood}(y))$$

G. Knowledge Incompleteness

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 developing knowledge models and also the fact that an intelligent system must rely on reasoning to infer missing knowledge.

H. ASCENS Ontology and Logic Foundations

In our approach, we rely on ontologies to represent the ASCENS knowledge models per case study. Considering the three different case studies to be undertaken by the ASCENS project [1]: *eMobility*, *swarm robotics* and *research clouds*, we intend to build four ASCENS *top-level ontologies*, where each conceptually represents one of the four ASCENS knowledge models (see Section II.B). This 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.

As mentioned above, to define the content of the ASCENS top-level ontologies, we are going to consider the problem domains of the three ASCENS case studies [1]: *eMobility*, *swarm robotics* and *research clouds*. Thus, the ASCENS top-level ontologies should be generic enough to cover all the lexical domains provided by these case studies. Figure 2 depicts our preliminary generic scheme for ASCENS ontologies. As shown, the ASCENS ontologies will provide the *symbolic representation* of conceptual classes (e.g., Resources, Events, etc.), objects, features of classes and relationships between classes.

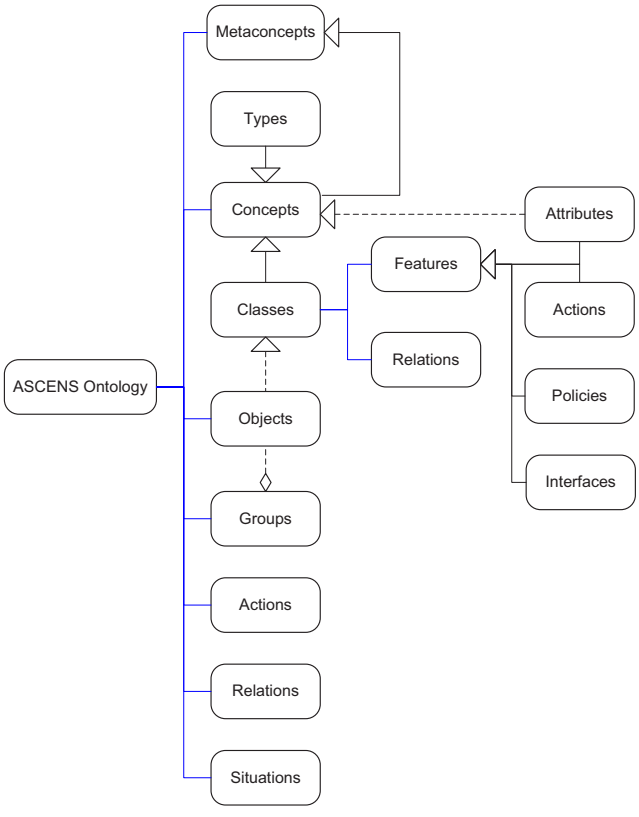

Figure 2. Generic Scheme for ASCENS Ontologies

The Policies are a SC feature that shall determine the behavior of a component in some important situations. The notion of policies is borrowed from the Autonomic System Specification Language (ASSL) [17], where the policies are addressed as self-managing behavior 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 is into a fluent then the policy behind that fluent is considered active.

To build the ASCENS top-level ontologies, in addition to applying the generic scheme shown in Figure 2, we also adopt a deductive method that is an elaborated version of the methodology presented by Gangemi et al. in [18]:

1) Select from the problem domains of the three ASCENS case studies:
   a. an initial set of domain-specific (per case study) *data entities* in terms of classes and objects;
   b. an initial set of *formal relations* among the selected data entities (neutral with respect to the domain choice), which shall play a *foundational role* in an ASCENS ontology;
2) Select and adapt from the three case studies *ground axioms* for these relations and data entities, such as:
   a. describing *instantiation relation*, *composition relation*, etc.;
   b. determining which entities are particulars (objects) and which universals (classes);
   c. determining algebraic properties.
3) Add *non-ground axioms*, which establish constraints across the formal relations.
4) Deduct properties of the data entities.
5) Define a set of properties induced by the formal relations.
6) Analyze systematically the allowed combinations of formal properties introducing a set of *concept classes*.
7) Classify the relevant kinds of domain-specific entities according to the concept classes. This classification will help us derive the minimum structure of each one of the ASCENS top-level ontologies.
8) Study the dependencies/interrelationships among concept classes, introducing *inter-concept relations.*
9) Increase the depth level of ontological analysis, by iterating this methodology within each concept class.

The *ASCENS top-level ontologies* are also considered as key elements for the construction of the so-called *ASCENS knowledge domains,* which we are going to build for each one of the four ASCENS knowledge models (see Section II.B). Thus, we consider *SC knowledge domain*, *SCE knowledge domain*, *SCE environment knowledge domain* and *SCE situational knowledge domain*. To build an *ASCENS knowledge domain*, in addition to the domain-specific ASCENS ontology, we also need to provide the computational structures determining the *logical foundations* of an ASCENS system. These structures classified as *facts*, *rules* and *constraints* (see Figure 3) shall be 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.*

As shown in Figure 3, an additional class of computational structures called "Inter-ontology Operators" can be added to an ASCENS knowledge domain to define logical operators that work on multiple ASCENS ontologies, e.g., *merging*, *mapping*, *alignment*, etc. Note that some computational structures can be shared among knowledge domains. All the ASCENS knowledge domains formed by a domain ontology, facts, rules, constraints, and inter-ontology operators will be used by the ASCENS Awareness Mechanism (see Section IV.A), which is a reasoning engine capable of knowledge inference and learning.

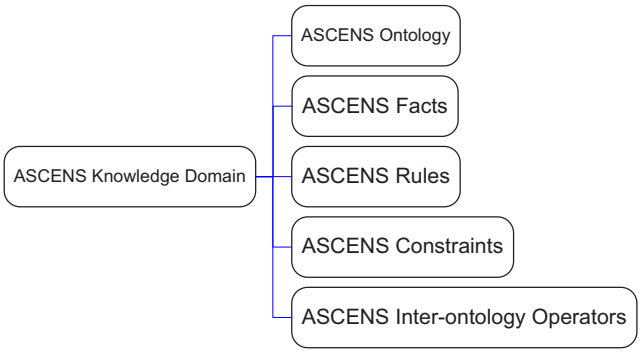

Figure 3. Generic Scheme for ASCENS Knowledge Domain

In our approach, *facts*, *rules*, *constraints* and *inter-ontology operators* will be eventually expressed in Second-Order Logic (SOL) [19] or in FOL (First-Order Logic) [16] 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:

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

Moreover, to properly address the *ASCENS knowledge domains*, we need to consider both *complexity* and *openness*. For example, single SCs are by far less complex than the SCE hosting those components. Thus, it is more reasonable to model the *SC knowledge domain* in much more detail than the *SCE knowledge domain* where more abstract concepts should be used. Also, due to issues related to the SCE architecture topology and the massive number of SCs composing an SCE, it is reasonable to assume that each SC has accurate knowledge of its own structure and states, while the states to which the SCE transits would be less certain. For these reasons, we may consider modeling the SC knowledge (e.g., attributes, actions and policies) in a deterministic fashion, which allows us to accurately

determine the current state of a SC. On the other hand, knowledge about the SCE or the environment may be encoded into probabilistic fashion (see Figure 4), because, although less accurate, this offers practical ways for a SC to determine what is the most likely current state of the SCE it belongs to. Probabilistic models are suitable for evolution and adaptation in case unplanned situations arise. This copes very well with the assumption for knowledge incompleteness (see Section III.C). The reasoning and learning capabilities of the probabilistic models will make it possible to update the *SCE situational knowledge domain* with new situations.

In our approach, to model *situations*, we intend to use the so-called *Situation Calculus* [16] or eventually a probabilistic extension of the same [20]. 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 compliance with FOL and SOL, the ability to cope with dynamic domains and the use of ASCENS ontology elements such as *actions* and *situations* make Situation Calculus a good candidate for axiomatizing ASCENS situations and consecutively the *SCE situational knowledge domain.*

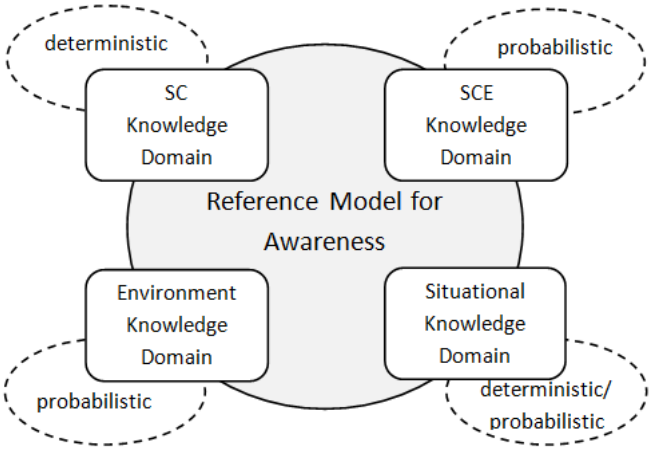

Figure 4. ASCENS Structured Knowledge

By combining the four ASCENS knowledge domains we build ASCENS Structured Knowledge that is the basis of a special reference model for awareness in ASCENS systems (see Figure 5).

IV. AWARENESS

Awareness is a concept playing a crucial role in intelligent systems. Conceptually, awareness is a product of *knowledge representation*, *knowledge processing* and *monitoring*. The Autonomic Computing paradigm [21] addresses two kinds of awareness in autonomic systems such as ASCENS:

- *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.

Lately, there has been significant research into different implementations of awareness for intelligent systems. For example, commercially-available server monitoring platforms, such as NimSoft's NimBUS [22] and Cittio's WatchTower network management application [23], offer robust, lightweight sensing and reporting capabilities across large server farms.

In another project, Kreidl and Frazier applied a special host-based Autonomic Defense System to solve the problem of awareness through model-based detection and response [24]. In this approach, special offline training of Markov models to represent different attack scenarios was applied. Forrest and Longstaff proposed methods of detecting anomalous host behavior by monitoring system call sequences of selected UNIX processes [25]. This requires an offline construction of normal pattern databases for each monitored entity.

A. Awareness in ASCENS Systems

A key success factor for an ASCENS system is to employ its knowledge in order to become an *aware system*. Such a SCE must be able to sense and analyze its SCs and the environment where it operates. A primary task should be to determine the state of each SC and the status of the global (SCE-level) and local (SC-level) *service-level objectives*. Thus, an aware SC should be able to notice change and understand the implications of that change. Thus, a monitoring system providing *self-monitoring*, *system monitoring*, and *monitoring* of the environment appears to be one of the key concepts in SCE awareness. In our approach, we target a monitoring system based on *notification events*.

Moreover, an aware ASCENS system should be able to apply both *pattern analysis* and *pattern recognition* techniques to determine normal and abnormal states based on the *situational knowledge model* (see Section II.A). Figure 5 represents a generic model for SC awareness. As shown, at the heart of the awareness model is the ASCENS Structured Knowledge (see Figure 4). This knowledge helps a SC recognize changes taking place in its internal SC world, in the SCE system and in the SCE environment (both the SCE and the environment form the *operational context* for a SC). Recall that by the nature of the ASCENS knowledge domains (see Section III.H), a SC has very comprehensive and deterministic knowledge about itself and probabilistic

knowledge (with an element of uncertainty) about the SCE system and the operational environment.

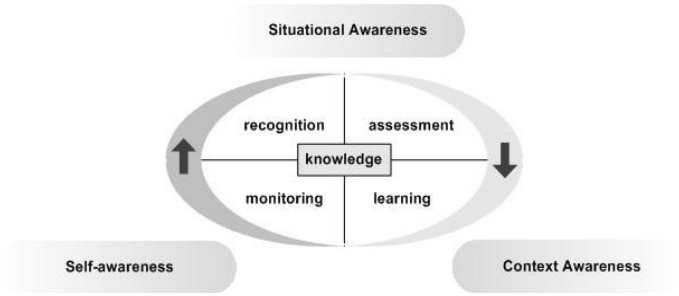

Figure 5. Generic Mechanism for Awareness in Service Components

As shown in Figure 5, a SC should also maintain *SCE situational knowledge*. Recall that the latter describes special situations that must be considered by the SC. Such situations are relevant changes in the environment, in the SCE system or in the SC itself. A situation could be a *cross-domain situation* where changes in different knowledge domains might be considered. For example, a change in the internal structure of a SC and a change in the environment could be a situation.

The model for SC awareness comprises a special *awareness control loop* that handles knowledge processing and update. This control loop comprises four distinct function steps:

- *monitoring* - collects, aggregates, filters, manages and reports internal and external details (e.g., metrics and topologies) gathered from the internal entities of a SC, the SCE system and from the operational environment;
- *recognition* - uses knowledge models to recognize changes in the SC, in the SCE system or in the SCE environment;
- *assessment* - determines entities (internal or external) of interest, generates hypotheses about situations involving these entities by processing the situational knowledge;
- *learning* - generates new situational knowledge (e.g., situational patterns) and maintains history of property changes.

The four functions forming the awareness control loop help a SC be aware of internal changes (self-awareness), external changes (context awareness – aware about both the SCE system and the SCE environment) and of situations (situational awareness).

V. AWARENESS SELF-INITIATION

In this section we present a formal algorithm for awareness based on self-initiation. The algorithm is illustrated with the ASCENS swarm robotics case study, which is using the marXbot robotics platform [26].

A. The marXbot Robot

The marXbot [26, 27] 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. The environment is defined as an arena where special cuboid-shaped obstacles are present in arbitrary positions and orientations. Moreover, the environment may contain a number of light sources, usually placed behind the goal area, which act as environmental cues used as shared reference frames among all robots.

Figure 6 shows a marXbot robot [27]. Such robot is equipped with a set of devices to interact with the environment and with other robots of the swarm [26]:

- a light sensor, that is able to perceive a noisy light gradient around the robot in the 2D plane;
- a distance scanner that is used to obtain noisy distances and angular values from the robot to other objects in the environment. Its range is 1.5 meters.
- a range and bearing communication system [28], with which a robot can communicate with other robots that are in line of sight. Its range is 4 meters.
- a gripper, that is used to physically connect to the transported object;
- two wheels independently controlled to set the speed of the robot.

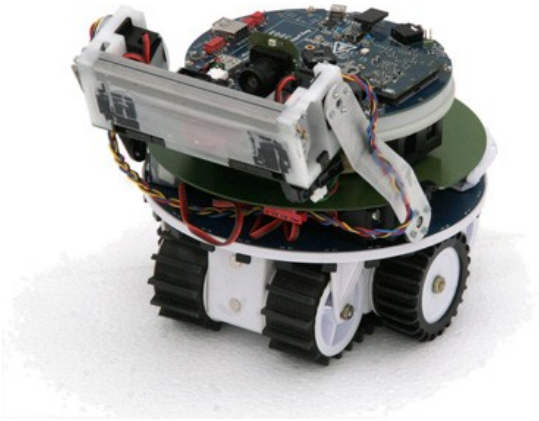

Figure 6. A marXbot Robot [27]

Currently, the marXbots robots are able to work in teams where they coordinate based on simple interactions on group tasks. For example, a group of marXbots robots may collectively move a relatively heavy object from point ***A*** to point ***B*** by using their grippers.

B. Self-initiation

Note that currently the marXbot robot does not imply complex intelligence. The latter is to be tackled by the ASCENS project and the algorithm presented here, considers next generation of intelligent marXbots which will gather information about the environment and the SCE swarm of marXbots (e.g., performance - both at the individual and the global level) and cope with changes at all levels – individual,

swarm and environment. Thus, the new generation of marXbots will imply awareness capability.

The awareness capability helps an idle robot self-initiate to react to changes in the swarm or the environment. To help a robot process its knowledge and become aware, a behavior model based on the so-called Partially Observable Markov Decision Processes (POMDP) [29] is considered. Note that this model is appropriate when there is uncertainty and lack of information needed to determine the state of the entire swarm. For example, swarm robots (e.g., marXbots) might be idle, i.e., not actively participating in the swarm's activities, because they are not certain about the current swarm state. Thus, the POMDP model helps a robot reason on the current swarm state (or that of the environment) and eventually self-initiate when an action is needed to be performed. According to our POMDP-based model, a swarm robot takes as input *observable* situations, involving other swarm robots and the environment, and generates as output actions initiating robot activity. Hence, the generated actions affect the state of the swarm. Formally, this model is a tuple $M = <S; A; T; R; Z; O>$, where:

- S is a finite set of states (formed by predefined situational patterns forming the *situational knowledge model* – see Section II.B) of the swarm that are not observable.
- An initial belief state $s_0 \in S$ is based on $p_0 (s_0; s_0 \in S)$, which is a discrete probability distribution over the set of swarm states S, representing for each state the robot's belief that is currently occupying that state. This is determined by matching the observable situations with the situational patterns.
- A is a finite set of actions that may be undertaken by the robot.
- $T: S \times A \rightarrow \Pi(S)$ is the state transition function, giving for each swarm state s and robot action a, a probability distribution over states. Here, $T (s; a; s')$ computes the probability of ending in state s', given that the start state is s and the robot takes action a, $p (s' \mid s; a)$.
- $O: A \times S \rightarrow \Pi(Z)$ is the observation function giving for each swarm state s and robot action a, a probability distribution over observations Z. For example, $O (s'; a; z)$ is the probability of observing z, in state s' after taking action a, $p (z \mid s'; a)$.
- $R : S \times A \rightarrow R$ is a reward function, giving the expected immediate reward gained by the robot for taking an action in a state s, e.g., $R (s; a)$. The reward is a scalar value in the range [0..1] determining, which action (among many possible) should be undertaken by the robot in compliance with the swarm goals.

Interpretation. To illustrate this model, let's assume that a marXbot swarm is currently occupying the state s = "*new object to be moved is discovered, but no moving team has been formed yet and still no other marXbot has self-initiated for team formation*". Let's assume there is at least one idle marXbot in the swarm ready to undertake a few actions A, including the action a = "*self-initiation for team formation*". The marXbot performs the following reasoning steps in order to self-initiate for team formation.

1. The marXbot computes its current belief state s_0 – the robot picks up the state with the highest probability p_0 and eventually $s_0 = s$.
2. The marXbot computes the probability p_1 of the swarm occupying the state s' = "*new object is discovered and a marXbot has self-initiated for team formation*" if the action a is undertaken from state s_0.
3. The marXbot computes the probability $p_2 (z \mid s'; a)$ of observation z = "*there are sufficient numbers of idle marXbots to form a new exploration team*".
4. The marXbot computes the reward $r (s_0; a)$ for taking the action a (*self-initiation for team formation*) in state s_0. If no other immediate actions should be undertaken (forced by other swarm goals), the reward r should be the highest possible, which will determine the execution of action a.

Probability Computation. The POMDP model for self-initiation requires the computation of a few probability values. In this subsection, we present a *model for assessing probability* applicable to the computation of POMDP probability values such as probability of the swarm being in a state and probability of observation. In our approach, the *probability assessment* is an indicator of the number of possible execution paths a robot may take, meaning the amount of certainty (excess entropy) in the swarm's behavior. To assess that behavior prior to the swarm implementation, it is important to understand the complex interactions among the robots in an SCE swarm. This can be achieved by modeling the behavior of individual reactive robots together with the swarm (or team) behavior as *Discrete Time Markov Chains* [30], and assessing the level of probability through calculating the probabilities of the state transitions in the corresponding models. We assume that the robot-swarm interaction is a stochastic process where the swarm events are not controlled by the robot and thus their probabilities are considered equal.

The theoretical foundation for our Probability Assessment Model is the property of Markov chains, which states that, *given the current state of the swarm, its future evolution is independent of its history*, which is also the main characteristic of a reactive autonomic robot.

Table 1. Transition Matrix P

	S_1	S_2	…	S_j	…	S_n
S_1	p_{11}	p_{12}	…	p_{1j}	…	p_{1n}
S_2	p_{21}	p_{22}	…	p_{2j}	…	p_{2n}
…	…	…	…	…	…	…
S_i	p_{i1}	p_{i2}	…	p_{ij}	…	p_{in}
…	…	…	…	…	…	…
S_n	p_{n1}	p_{n2}	…	p_{nj}	…	p_{nn}

An algebraic representation of a Markov chain is a matrix (called *transition matrix*) (see Table 1) where the rows and columns correspond to the states, and the entry p_{ij} in the i^{th} row, j^{th} column is the transition probability of being in state S_j at the stage following state S_i. The following property holds for the calculated probabilities:

$$\Sigma_j \, p_{ij} = 1 \quad (1)$$

We contend that probability should be calculated from the *steady state* of the Markov chain. A steady state (or *equilibrium state*) is one in which the probability of being in a state before and after a transition is the same as time progresses. Here, we define probability for a swarm configuration composed of k robots as the level of certainty quantified by the source excess entropy, as follows.

$$Probability\ (SCE) = \Sigma_{i=1,k} H_i - H \quad (2)$$

$$H_i = - \Sigma_j \, p_{ij} \log_2 (p_{ij}) \quad (3)$$

$$H = - \Sigma_i \, v_i \, \Sigma_j \, p_{ij} \log_2 (p_{ij}) \quad (4)$$

Here,

- H is an entropy that quantifies the level of uncertainty in the Markov chain corresponding to an SCE swarm;
- H_i is a level of uncertainty in a Markov chain corresponding to a marXbot robot;
- v is a steady state distribution vector for the corresponding Markov chain;
- p_{ij} values are the transition probabilities in the extended state machines that model the behavior of the i^{th} robot.

Note that for a transition matrix P, the steady state distribution vector v satisfies the property $v*P = v$, and the sum of its components v_i is equal to 1.

Interpretation. The level of uncertainty H is exponentially related to the number of *statistically typical paths* in the Markov chain. Having an entropy value of 0 means that there is no level of uncertainty in a Markov system for a specific robot's behavior. Here, a higher value of a probability measure implies less uncertainty in the model, and thus, a higher level of predictability.

VI. CONCLUSIONS

This paper has presented our state-of-the-art vision and initial research results on knowledge representation and awareness in special autonomic systems called Autonomic Service-Component ENSembles (ASCENS). Such systems need to be developed with initial knowledge and learning capabilities based on knowledge processing and awareness. It is very important how the system knowledge is structured and modeled to provide essence of both self-awareness and context awareness. In our approach, we consider structured knowledge based on four knowledge models such as SC knowledge, SCE knowledge, SCE environment knowledge and situational knowledge.

Among the many popular approaches for knowledge representation, ontologies are considered to be the most expressive one. In general, ontologies provide a form of explicit representation of domain concepts and relationships between those concepts. To build the ASCENS structured knowledge, we rely on ontologies to represent the four ASCENS knowledge models. For these models we consider the problem domains provided by the three different case studies of the ASCENS project [1]: *eMobility*, *swarm robotics* and *research clouds*. We use *top-level ontologies* to build the ASCENS knowledge domains each representing a high-level view of a part of the world as perceived by a SC. In addition, for each knowledge domain we construct special computational structures determining the *logical foundations* of those domains. The computational structures are classified as *facts*, *rules*, *constraints* and *inter-ontology operators*. They are intended to drive the special ASCENS Awareness Mechanism by providing for knowledge inference and learning.

Moreover, an approach to SCE awareness has been presented in this paper. At the heart of the proposed awareness model is the ASCENS structured knowledge. We further develop this awareness model, by providing a formal algorithm for awareness self-initiation. This approach helps a SC automatically determine the need of action, e.g., team formation. Our formal model for self-initiation is based on the Partially Observable Markov Decision Processes and Discrete Time Markov Chains, where we do not consider any central controller, but complex algorithms working on state-action relationships and considering a variety of probability values.

Future work is mainly concerned with further development of the ASCENS knowledge models and algorithms for knowledge processing. Those models and algorithms are going to be implemented in the three ASCENS case studies.

ACKNOWLEDGEMENT

This work was supported by ASCENS grant at Lero — the Irish Software Engineering Research Centre at University of Limerick, Ireland and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero.

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] P. Makhfi, MAKHFI - Methodic Applied Knowledge to Hyper Fictitious Intelligence, 2008, http://www.makhfi.com/.

[4] V. Devedzic and D. Radovic, "A Framework for Building Intelligent Manufacturing Systems", IEEE Transactions on Systems, Man, and Cybernetics, Part C - Applications and Reviews, Vol. 29, 1999, pp. 422–439.

[5] J.L. Gordon, “Creating knowledge maps by exploiting dependent relationships”, Journal of Knowledge-Based Systems, Elsevier Science, Vol. 13, 2000, pp. 71-79.

[6] E. H. Shortliffe, Computer Based Medical Consultations: MYCIN. Elsevier, New York, 1976.

[7] John F. Sowa, “Semantic networks”, Encyclopedia of Artificial Intelligence (ed. S. C. Shapiro), 2nd ed., Wiley, New York, 1992.

[8] A.J. Canas, J.W. Coffey, T. Reichherzer, G. Hill, N. Suri, R. Carff, T. Mitrovich, and D. Eberle, “El-Tech: A performance support system with embedded training for electronics technicians”, Proc. 11th Florida AI Research Symposium (FLAIRS '98), Sanibel Island, FL, USA, 1998.

[9] K.M. Ford, J.W. Coffey, A.J. Canas, C.W. Turner, and E.J. Andrews, “Diagnosis and explanation by a Nuclear Cardiology Expert System”, International Journal of Expert Systems, Vol. 9(4), 1996, pp. 499-506.

[10] J.W. Coffey, R.R. Hoffman, A.J. Canas, and K.M. Ford, “A Concept Map-based Knowledge Modelling Approach to Expert Knowledge Sharing”, Proc. of Information and Knowledge Sharing (IKS 2002), Virgin Islands, USA, 2002.

[11] W. Swartout and A. Tate, “Ontologies”, IEEE Intelligent Systems, Vol. 14, 1999, pp. 18-19.

[12] D.L. McGuinness and F. Van Harmelen, OWL web ontology language overview, W3C Recommendation, World Wide Web Consortium, 2004.

[13] M.K. Smith, C. Welty, and D.L. McGuinness, OWL web ontology language guide, W3C Recommendation, 2004.

[14] G. Antoniou and F. Van Harmelen, A semantic web primer, MIT Press, 2004.

[15] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, “An introduction to the syntax and content of Cyc”, Proc. AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, Citeseer, 2006, pp. 44–49.

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

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

[18] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari, “Understanding Top-Level Ontological Distinctions”, Proc. of IJCAI-01 Workshop on Ontologies and Information Sharing, 2001.

[19] A. S. Troelstra and H. Schwichtenberg, Basic proof theory, Cambridge University Press, 2000.

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

[21] J. O. Kephart and D. M. Chess, “The vision of Autonomic Computing”, IEEE Computer, Vol. 36 (1), 2003, pp. 41-50.

[22] NimSoft, Nimbus for server monitoring - solution overview paper, Tech. rep., NimSoft, 2006, http://www.nimsoft.com.

[23] B. Boardman, Cittio's watchtower 3.0, Tech. rep., Cittio (2006), http://www.networkcomputing.com/data-protection/cittios-watchtower-30.php.

[24] O. P. Kreidl and T. M. Frazier, “Feedback control applied to survivability: A host-based autonomic defense system”, IEEE Transactions on Reliability, Vol. 53 (1), 2004, pp. 148-166.

[25] S. Forrest and T. Longsta, “A sense of self for UNIX processes”, Proc. IEEE Symposium on Security and Privacy, IEEE Computer Society, 1996, pp. 120-128.

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

[27] M. Bonani, T. Baaboura, P. Retornaz, F. Vaussard, S. Magnenat, D. Burnier, V. Longchamp and F. Mondada, marXbot, Laborotoire de Systemes Robotiques (LSRO), École Polytechnique Fédérale de Lausanne, http://mobots.epfl.ch/marxbot.html.

[28] J. Roberts, T. Stirling, J.-C. Zufferey and D. Floreano, “2.5D infrared range and bearing system for collective robotics”, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), 2009, pp. 3659–3664.

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

[30] W. J. Ewens and G. R. Grant, “Stochastic processes (i): poisson processes and Markov chains”, Chapter in Statistical methods in Bioinformatics, 2nd edition, Springer, New York, 2005.