Specifications for Intelligent Systems: How do they differ from those of Non-intelligent Ones?

Elena R. Messina and Hui-Min Huang
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8230

ABSTRACT

A growing number of applications are utilizing self-proclaimed intelligent systems. As for any complex system, specifications are necessary for guiding the implementation, evaluating the performance and verifying the product. We examine the question of what unique features are needed when specifying an intelligent system. We also investigate various specification techniques that may support these features.

1. INTRODUCTION

Intelligent systems have a myriad of definitions, some of which include, does the right thing, is non-linear, adaptive, goal-oriented, knowledge-based, autonomous, capable of learning, able to deal with uncertainty and complexity, and uses symbolic reasoning, Although these phrases are imprecise, they serve to indicate the flavor of intelligent systems. Clearly, they have more and different capabilities than the majority of software systems, even extremely complex ones. It is therefore especially challenging to specify what the intelligent system is supposed to do. This is a relatively unexplored area and we attempt to describe some possible means of specifying intelligent systems.

2 .SYSTEM AND SOFTWARE SPECIFICATIONS

We begin by examining specifications for software and systems in general. In this paper, we define a system as

> An assemblage of parts forming a complex or united whole that serves a useful purpose [6]

Software can be specified in the following aspects, system functions, system behavior, system communications, conceptual decomposition, component functions, component behavior, and component communications. Each of these aspects may require its own specification technique.

In the software testing world, there is a major distinction drawn between *behavioral* and *structural* testing models [5]. These models can apply to individual components and overall systems. Behavioral, also known as Black Box, evaluates the external phenomena exhibited by what is being tested. These phenomena include the system s expected functions, responses to inputs and stimuli, and the outputs. It can include temporal behavior and execution characteristics. It does not include how the system accomplishes its behavior. Therefore, specific algorithms, data structures, and other internal implementation-specific aspects are not evaluated.

Structural, also referred to as White Box testing, evaluates the internal aspects of the software. For performing white box testing, the system design and implementation must be specified. Internal aspects, such as algorithms and data structures are explicitly reviewed and tested.

We apply these two software testing models to the specification of intelligent systems. Black box specifies the behavioral aspects and white box specifies the internal aspects of systems.

3. DISTINGUISHING AN INTELLIGENT SYSTEM

We will present a simple example from the Performance Metrics for Intelligent Systems Workshop 2001 White Paper [16]. This will serve to illustrate the complexities of specifying an intelligent system (IS), versus a non-intelligent system (NIS).

An intelligent system is given a job to do (task, mission, set of commands). *The job definition for IS is expected to be less specific than in an NIS.* A system with intelligence ought to have the capability to interpret incomplete commands, understand a higher level, more abstract commands and to supplement the given command with additional information that helps to generate more specific plans internally. The IS should understand the context within which the command is given. For example, instead of telling a mobile robot to go to a specific location in world coordinates GO_TO (X, Y), the command could be Go to the window nearest to me. The robot should understand what a window is and know that it needs to find one which is the minimum distance away from me and move to that location. It also has a nominal proximity that it maintains from the goal location. Notice that the command did not determine how close the robot needs to get to the window. It is expected that the robot knows where to stop the motion in similar cases, or the distance from the window should allow for convenient performance of other, or consequent movements.

In this example, the robot has to interpret the instruction and generate appropriate actions, given a sensed model of the situation to which it has to apply *a priori* knowledge about objects like windows. These are some of the aspects of an IS that need to be specified.

4. BLACK BOX SPECIFICATION

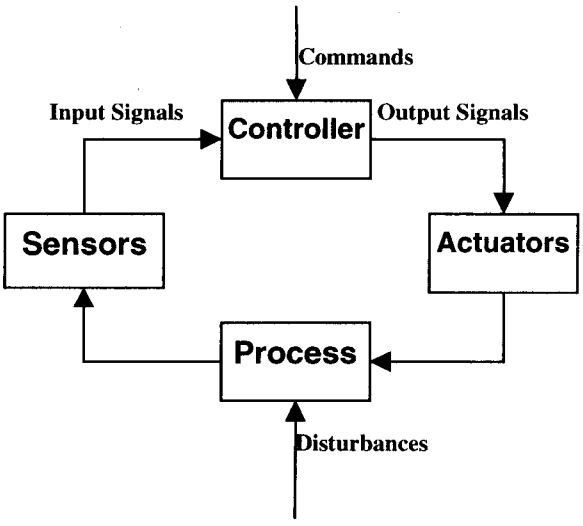

Figure 1: A Typical Process Control Model

Figure 1 shows a process model for a system. In this model, the inputs, outputs, constraints, and disturbances can be less well-defined or crisp for an IS.

Another example comes from [1], in which the black box specification for an intelligent military vehicle is given. The unmanned scout vehicle will be required to perform duties of a typical manned scout vehicle, which has a driver, commander, and lookout. The duties include locomotion, attention, and communication functions. The scout, either robotic or human, is expected to achieve a plan given by its supervisor. The plan specifies a schedule of waypoints (space and time) to a given tolerance and a definition of the mission goal, such as clearing a specific area using onboard sensors. The plan is further specified, declaring that it be performed using militaristic behaviors, such as stealthy movements, performing bounding overwatches with other vehicles, honoring corridors, etc. The scout is required to understand military maps, which may be updated with new information from its own observations, other scouts, or its superior or operator. It must be able to use its sensors to perceive the environment and avoid obstacles and hazards, use concealment and cover, and recognize enemy and friendly forces. The system must have the capability to build situational awareness by combining its perceptions with a priori knowledge. Thus, it must be able to distill and abstract the information it receives. It must further be able to project the situation into the future, in order to predict where enemy troops may be located given current trajectories. And finally, it must be able to adapt its behavior given the situational awareness it has gained.

Taking a more generic perspective, the following is a list of black box specifications for an intelligent system:

- to interpret high level, abstract, and vague commands and convert them into a series of actionable plans
- to autonomously make appropriate decisions as it is carrying out its plans
- to re-plan while executing its plans and adapt to changes in the situation
- to register sensed information with its location in the world and with a priori data
- to fuse data from multiple sensors, including resolution of conflicts

- to handle imperfect data from sensors, sensor failure or sensor inadequacy for certain circumstances
- to direct its sensors and processing algorithms at finding and identifying specific items or items within a particular class
- to schedule and focus resources according to the vehicle s mission
- to handle a wide variation in surroundings or objects with which it interacts
- to deal with a dynamic environment
- to map the environment so that it can perform its job
- to update its models of the world, both for short-term and potentially long-term using efficient formats and organization
- to understand generic concepts about the world that are relevant to its functioning and ability to apply them to specific situations
- to deal with and model symbolic and situational concepts as well as geometry and attributes
- to work with incomplete and imperfect knowledge by extrapolating, interpolating, or other means
- to be able to predict events in the future or estimate future status
- the ability to evaluate its own performance and to learn and improve

In terms of the representation of the specifications, several key aspects must be supported. They include numerical and symbolic entities, uncertainty, probability,

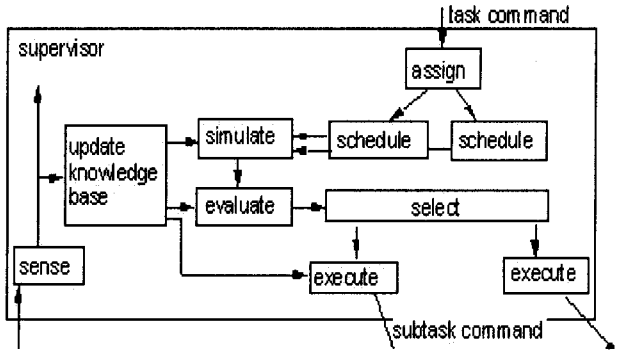

Figure 2: An RCS Node

ranges, fuzziness, temporal requirements or constraints, and spatial requirements or constraints.

5. WHITE BOX SPECIFICATION

Another approach is to specify how a system should be constructed in order to attain intelligence. In this view, the specification of a proven architecture that supports intelligence, such as 4D/RCS [2] or SOAR [18], would suffice. Figure 2 shows an example of a 4D/RCS node. A system s architecture is

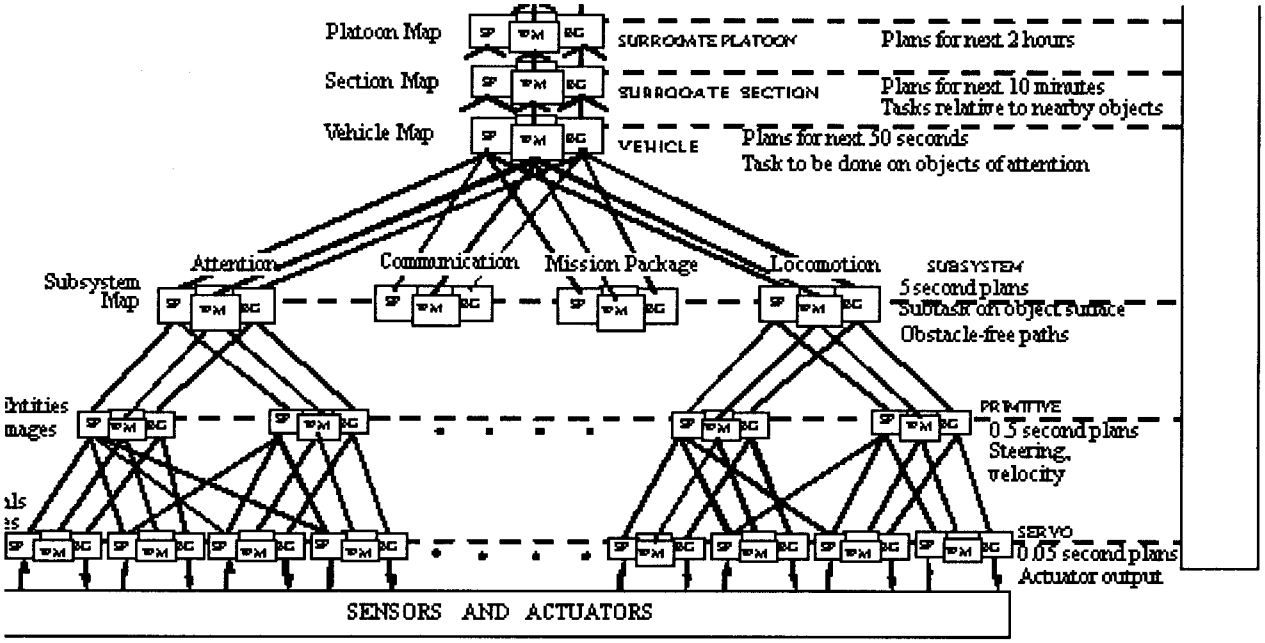

Figure 3: An Example for 4D/RCS Hierarchy

constructed from an assembly of nodes, arranged in a hierarchical fashion and according to guidelines set forth in the reference architecture. Figure 3 is a representative example of an autonomous vehicle implementation based on 4D/RCS. The node elements perform the activities deemed necessary for achieving the kinds of behaviors discussed in Section 4.

6. SPECIFICATION TECHNIQUES

We have been investigating existing representation techniques for specifying intelligent systems and believe that there are several viable alternatives. Although further research and verification is necessary, we briefly discuss some approaches for representing intelligent systems. Often, the comprehensive and higher level specification languages or methods encompass several elementary techniques. We will note some examples in this section.

For black box specification, there are several options from the formal methods or formal representation domain.

Finite state machines (FSM) have been used extensively for specifying the behavior of intelligent systems. There are limitations, however, in terms of being able to specify fuzziness, constraints, etc. Research has been done to extend FSM s capability toward this direction. Extended finite state diagram [20] utilizes variables, in addition to events, to facilitate handling continuous domain aspects. Hybrid state machine [19] uses regular expressions and combines multiple events to handle the complexity issue.

One must also recognize that FSM s cannot specify the data /knowledge and its representation.

XML has begun to play a role in software architectural specifications [21]. XML aims at improving the markup to facilitate unambiguous meanings of data that HTML lacks. Yet XML is limited in describing objects and their relationships, this gives rise to the new DARPA Agent Markup Language (DAML) [11], which is worth further investigation.

For white box specification, one can use the traditional software engineering structured techniques, such as entity relation diagrams (ERD), data flow diagrams (DFD), flow charts, and block diagrams. Wieringa provided a thorough survey on these techniques [20]. He also noted that the Yourdon System Method (YSM) uses a context diagram to specify some black box aspects for systems and uses ERD and DFD to specify some white box aspects. Another comprehensive language, Specification and Description Language (SDL), uses state diagrams and message sequence charts to specify systems external behavior black box and uses block diagrams for a part of white box specification [4].

Formal methods are also applicable. One approach, using Architectural Description Languages (ADL), was investigated in depth [10]. Particular representational requirements that were found to be necessary in order to model behavior of an intelligent system include: specifying system components, specifying connections between components, specifying system behaviors, and defining architectural style. A case study provided positive indication.

Interface Definition Language (IDL) is used to specify software component interfaces, traditionally limited to the data aspect. It has been revealed that adding behavioral specification to IDL would lead to more robust systems [8]. This is an issue of syntactic versus semantic. We believe that this work is worth further investigation as a part, but not the whole solution, of the IS specification techniques.

Craig has, in his book entitled, Formal Specification of Advanced AI Architectures, applied the Z language to formally specify two AI architectures, blackboard and CASSANDRA [9]. The achieved specifications exist in the form of shell, which can be used to encode knowledge of application systems.

The Unified Modeling Language (UML) seems to be able to support both black box and white box system specification. UML, being object oriented, is inherently consistent with the physical world in terms of specifying objects that an IS is to operate with. UML employs state transition diagrams for specifying behaviors. UML has emerged as an industrial standard language. We, therefore, take a closer look at it in the next section.

7 . UML TO SPECIFY SYSTEM INTELLIGENCE

To achieve intelligent systems, it is imperative to specify them using a comprehensive and standard language. UML satisfies these requirements by allowing:

- Specifying data requirements and the associated knowledge engineering processes that can evaluate and judge data that contain uncertainties. UML further provides customized representations such as stereotypes and profiles that can be used to specify domain specific knowledge items.

- Specifying data relationships with the UML features including generalization, aggregation, and multiple kinds of associations. These facilitate transitions from raw data to knowledge with different levels of abstraction. Figure 4 gives an example.

- Specifying complex goal structures that contain uncertainties, requiring evaluation

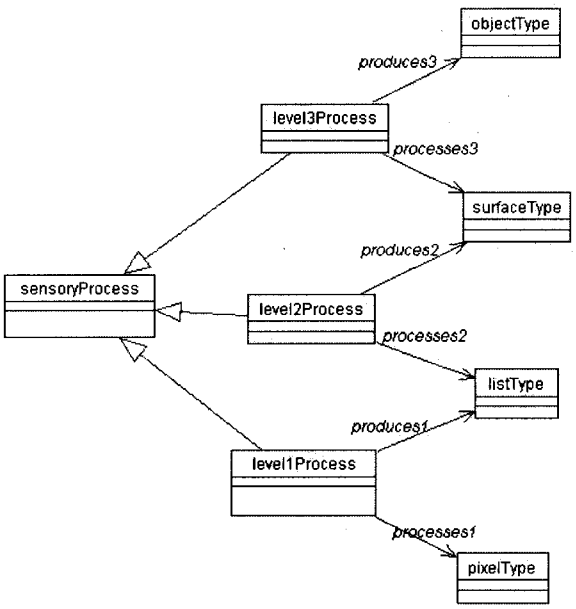

Figure 4: Specifying a Sensory Processing Function with UML

and judgement. For example, Fill up the gas tank ASAP.

- Specifying complex task structures containing either ambiguities or specific terms that require interpretation or knowledge inference. For example, to perform a search to find a trapped person named John Doe. Figure 5 illustrates such a task representation.

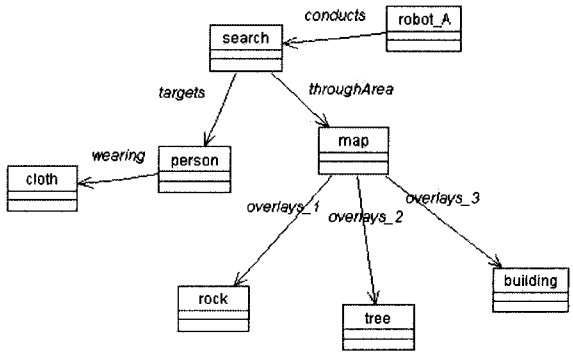

Figure 5: UML Specification of an IS Task

- Use sequence diagrams to specify system behavioral requirements. Scenario driven methods have been used to specify system requirements [13,17,7].

8. RELATED WORK

In surveying existing techniques and tools for specifying intelligent systems, we note that the agent world has been extensively defining various agents. The Foundation for Intelligent Physical Agents (FIPA) [12] has several examples on their web site. Their approach is black box:

The first characteristic assumed is that agents are communicating at a higher level of discourse, i.e. that the contents of the communication are meaningful statements about the agents environment or knowledge. This is one characteristic that differentiates agent communication from, for example, other interactions between strongly encapsulated computational entities such as method invocation in CORBA.

Core capabilities are described in terms of agent s mental attitudes: Belief, Uncertainty, and Intention.

Intelligent Networks also have evidence of similar requirements to intelligent systems and have taken the FSM approach, as well as formal specifications. SDL [4] is used extensively in this domain.

9. SUMMARY

We studied the issue of the definition of intelligent systems. We also proposed a model for system specification that contains black box and white box specification. We listed the black box specification criteria for intelligent systems and discussed the white box specification approaches for intelligent systems. We evaluated many specification techniques in an attempt to find viable ones for the specification of intelligent systems. We provided examples of how UML are used to specify certain aspects of the 4-D/RCS intelligent system architecture.

REFERENCES

1. Albus, J., "Features of Intelligence Required by Unmanned ground Vehicles," Proceedings of the 2000 Performance Metrics for Intelligent Systems Workshop, Gaithersburg, MD, 2000.
2. Albus, J., 4-D/RCS: A Reference Model Architecture for Demo III, NISTIR 5994, NIST, Gaithersburg, MD, 1997.
3. Blanchard, B., and Fabryky, W., 1990 Systems Engineering and Analysis, Prentice-Hall, Englewood Cliffs, N.J.
4. Belinda, F. and Hogrefe, D., "The CCITT-Specification and Description Language SDL," Computer Networks and ISDN Systems, Vol 16, pp. 311-341, Elsevier Science Publishers, B.V., North-Holland, 1989.
5. Beizer, Software Testing Techniques, Van Nostrand Reinhold, 1990.
6. Blanchard, B., and Fabryky, W., 1990 Systems Engineering and Analysis, Prentice-Hall, Englewood Cliffs, N.J.
7. Chen XJ, Logrippo L, "Deriving use cases for distributed systems from knowledge requirements," Annales Des Telecommunications- Annals of Telecommunications 55: (1-2) 45-57 Jan-Feb 2000, ISSN: 0003-4347 Presses Polytechniques Et Universitaires Romandes, Lausanne, Switzerland.
8. Cicalese, C.D.T. and Rotenstreich, S., "Behavioral Specification of Distributed Software Component Interfaces," IEEE Computer, July, 1999.
9. Craig, I.D., Formal Specification of Advanced AI Architectures, Ellis Horwood, New York, NY, 1991.
10. Dabrowski, C., Huang, H., Messina, E., and Horst, J., Formalizing the NIST 4-D/RCS Reference Model Architecture Using an Architectural Description Language, NISTIR 6443, Gaithersburg, MD, December, 1999.
11. http://www.daml.org/index.html
12. http://www.fipa.org/index.html
13. Huang, HM, et al., "Intelligent System Control: A Unified Approach and Applications," Book Chapter in Academic Press Volumes on "Expert Systems Techniques and Applications," C. T. Leondes, Ed., 2000.
14. N.G. Leveson, M.P.E. Heimdahl, and J.D. Reese. Designing Specification Languages: Lessons Learned and Steps to the Future. Proceedings of the Seventh ACM/SIGSOFT Symposium on the Foundations of Software Engineering, Toulouse, France, September, 1999.
15. N. G. Leveson, M. P.E. Heimdahl, H. Hildreth, and J. Reese. Requirements Specification for Process Control Systems. IEEE Transactions on Software Engineering, Vol. SE-20, No. 9, pp. 684-707 (September 1994).
16. Messina, E., Meysel, A., Reeker, L., Measuring Performance and Intelligence of Intelligent Sytems White Paper 2001, Proceedings of the 2001 Performance Metrics for Intelligent Systems Workshop, Mexico City, Mexico, 2001.
17. Mortensen KH, "Automatic code generation method based on coloured Petri net models applied on an access control system," Application and Theory of Petri Nets 2000, Proceedings, Springer-Verlag Berlin, Berlin, 2000.
18. http://ai.eecs.umich.edu/soar/
19. Sakharov, A., "A Hybrid State Machine Notation for Component Specification," ACM SIGPLAN Notices, V. 35 (4), April 2000.
20. Wierenga, R., A Survey of Structured and Object-Oriented Software Specification Methods and Techniques, ACM Computing Surveys, Vol. 30, No. 4, December, 1998.
21. http://www.isr.uci.edu/projects/xarchuci/
22. Zhang, Y. and Mackworth, A., Formal Specification of Performance Metrics for Intelligent Systems, Proceedings of the Performance Metrics for Intelligent Systems Workshop 2001, NIST SP 970.