Knowledge Driven Robotics

Stephen Balakirsky[b], Zeid Kootbally[b,c], Thomas Kramer[a,b], Anthony Pietromartire[b], Craig Schlenoff[b]

[a]*Catholic University of America, Washington, DC, USA*
[b]*National Institute of Standards and Technology, Gaithersburg, MD USA*
[c]*University of Maryland, College Park, MD, USA*

Abstract

In this article, a newly developed knowledge architecture that was designed to support the IEEE Robotics and Automation Society's Ontologies for Robotics and Automation Working Group will be presented. This architecture allows for the creation of systems that demonstrate flexibility, agility, and the ability to be rapidly re-tasked. The architecture, as well as an implementation that combines extensions to the Unified System for Automation and Robot Simulation (USARSim) and the Robot Operating System (ROS), will be discussed in detail. This implementation extends USARSim and ROS to provide models of industrial robots and end-effectors while providing a case study in the area of robotic kit building. The architecture's potential for creating flexible, agile, and rapidly re-taskable systems will be presented, and possible areas of future work will be discussed.

Keywords:

1. Introduction

Today's state-of-the-art industrial robots are often programmed with a teach pendent. This requires that the robot cell be taken off-line for a human-led teaching period when the cell's task is altered. Many research systems depend on numerous hand-tuned parameters that must be re-tuned by operators whenever a untested environment is encountered. These requirements for human intervention cause the robotic systems to be brittle and limited in operational scope. The robotic systems of tomorrow need to be capable, flexible, and agile. These systems need to perform their duties at least as well as human counterparts, be able to be quickly re-tasked to other operations, and be able to cope with a wide variety of unexpected environmental and operational changes. In order to be successful, these systems need to combine domain expertise, knowledge of their own skills and limitations, and both semantic and geometric environmental information.

The IEEE Robotics and Automation Society's Ontologies for Robotics and Automation Working group is striving to create an overall ontology and associated methodology for knowledge representation and reasoning that will address these knowledge needs. As part of the Industrial Subgroup of the IEEE Working Group, the authors have been examining a novel architecture that allows for the combination of the static aspects of the ontology with dynamic sensor processing to allow for a robotic system that is able to cope with environmental and task changes without operator intervention.

Architectures have always been an important part of advanced robotic systems and have evolved with the robotic systems that they characterize. Early architectures such as the one developed for the Army Research Laboratory's (ARL) Demo I program tended to be hardware focused [1]. The Demo I platform was able to perform reconnaissance, surveillance, and target acquisition (RSTA) as well as tele-operated control over a compressed bandwidth radio channel. However, the lack of a software architecture limited the sharing of information between software applications and forced the human operator to fuse information returned from various applications in order to make control decisions.

The ARL Demo III program demonstrated sophisticated off-road navigation while incorporating a much more sophisticated hardware/software architecture known as 4D/RCS [2]. This architecture provide for a closer coupling of software systems and allowed for the sharing of items such as sensor data and cost maps. However, the lack of a knowledge architecture limited this system's ability to utilize a priori knowledge to reason over its environment and provide fine-grained classification of terrain and obstacles. This resulted in numerous parameters that required human tuning based on the expected environmental conditions.

In order to become accepted in industrial applications, tomorrow's systems will need to do more. They will have to demonstrate flexibility through the elimination of hand-tuned parameters. They will need to demonstrate agility through the ability to cope with rapidly changing situations and task requirements, and they will need to be able to be rapidly re-tasked in order to adapt to new tasking quickly and efficiently. The industrial subgroup of the Ontologies for Robotics and Automation Working group is striving to demonstrate that the inclusion of knowledge in the form of an ontology will allow robotic systems to progress to this next level.

The organization of the remainder of this paper is as follows: Section 2 presents an overview of the knowledge driven architecture that has been developed for this effort. Section 3 describes the domain of kit building which is a greatly simplified, but still practically useful manufacturing/assembly domain. Section 4 describes our ontology for the kitting domain. Section 5 describes the implementation of the system and Section 6 provides for conclusions and future work.

2. Architecture

The design approach described in this article is not intended to replace sound engineering of an intelligent system, but rather as an additional step that may be applied in order to provide the system with more agility, flexibility, and the ability to be rapidly re-tasked. This is accomplished by assuring that the appropriate knowledge of the correct scope and format is available to all modules of the intelligent system.

The overall knowledge model of the system may be seen in Figure 1. The figure is organized vertically by the representation that is used for the knowledge and horizontally by the classical sense-model-act paradigm of intelligent systems. On the vertical axis, knowledge begins with Domain Specific Information that captures operational knowledge that is necessary to be successful in the particular domain in which the system is designed to operate. This information is then organized into a domain independent representation (an Ontology) that allows for the encoding of an object taxonomy, object-to-object relationships, and aspects of actions, preconditions, and effects. Aspects of this knowledge are automatically extracted and encoded in a form that is easy for a planning system to utilize (the Planning Language). Once a plan has been formulated, the knowledge is once again transformed. This time into a representation that is easily used by a robotic system (the Robot Language).

It is acknowledged that sensing and action are important parts of a robotic system. However, this article focuses on knowledge representation, and thus the modeling section will be described in the most detail.

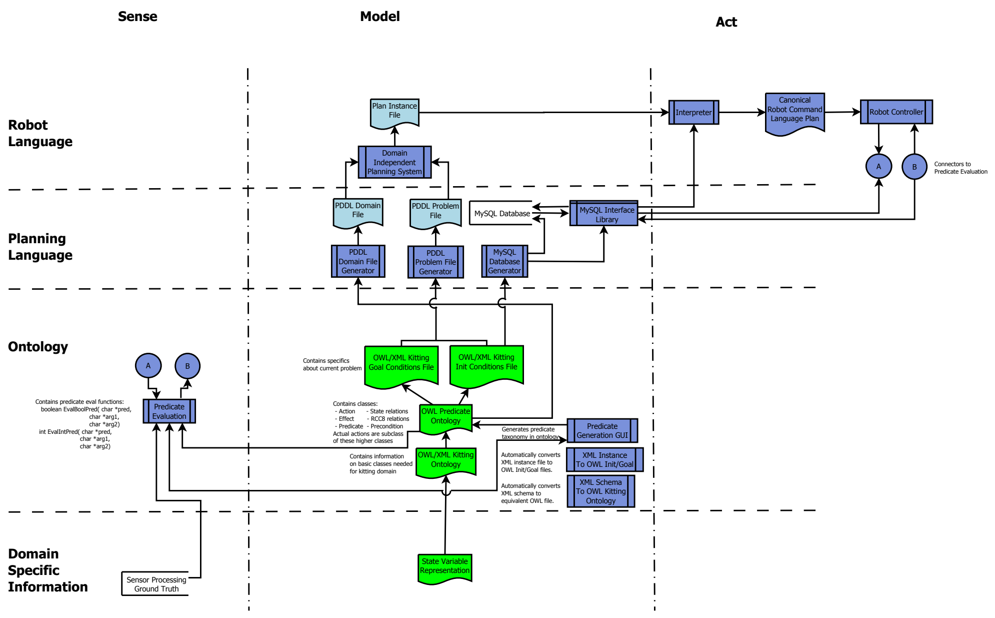
Sense
Model
Act
Robot Language
Planning Language
Ontology
Domain Specific Information
Plan Instance File
Domain Independent Planning System
Interpreter
Canonical Robot Command Language Plan
Robot Controller
A
B
Connectors to Predicate Evaluation
PDDL Domain File
PDDL Problem File
MySQL Database
MySQL Interface Library
PDDL Domain File Generator
PDDL Problem File Generator
MySQL Database Generator
Contains specifics about current problem
OWL/XML Kitting Goal Conditions File
OWL/XML Kitting Init Conditions File
A
B
Contains predicate eval functions: boolean EvalBoolPred(char *pred, char *arg1, char *arg2) int EvalIntPred(char *pred, char *arg1, char *arg2)
Predicate Evaluation
Contains classes: - Action - Effect - Predicate - State relations - RCC8 relations - Precondition Actual actions are subclass of these higher classes
OWL Predicate Ontology
Generates predicate taxonomy in ontology
Predicate Generation GUI
Contains information on basic classes needed for kitting domain
OWL/XML Kitting Ontology
Automatically converts XML instance file to OWL Init/Goal files.
XML Instance To OWL Init/Goal
Automatically converts XML schema to equivalent OWL file.
XML Schema To OWL Kitting Ontology
Sensor Processing Ground Truth
State Variable Representation

Figure 1: Knowledge Driven Design extensions- In this figure, green shaded boxes with curved bottoms represent hand generated files while light blue shaded boxes with curved bottoms represent automatically created boxes. Square boxes represent processes and libraries.

2.1. Domain Specfic Information

The most basic knowledge that must be gathered for a knowledge driven system is domain specific information (DSI). This appears along the bottom row of Figure 1. DSI includes sensors and sensor processing that are specifically tuned to operate in the target domain. Examples of sensor processing may include pose determination and object identification.

For the knowledge model, a scenario driven approach is taken where the DSI design begins with a domain expert creating one or more use cases and specific scenarios that describe the typical operation of the system. Based on these scenarios and use cases, the high-level actions that the system must be able to accomplish can be enumerated and described. An action description that includes any preconditions that must be true for an action to be valid as well as expected effects that will result from a given action is then created for each action.

put-kittray(*robot*,*kittray*,*worktable*): The *Robot robot* puts the *KitTray kittray* on the *WorkTable worktable*.

preconditions	*effects*
kittray-location-robot(*kittray*,*robot*)	¬kittray-location-robot(*kittray*,*robot*)
robot-holds-kittray(*robot*,*kittray*)	¬robot-holds-kittray(*robot*,*kittray*)
worktable-empty(*worktable*)	¬worktable-empty(*worktable*)
	kittray-location-worktable(*kittray*,*worktable*)
	robot-empty(*robot*)
	on-worktable-kittray(*worktable*,*kittray*)

Figure 2: Example action along with its preconditions and expected effects.

Based on the precondition and effect statements, objects in the environment that are relevant for system operation can be identified. For example, the action depicted in Figure 2 has a given robot place a kit tray onto a work table. Based on the preconditions and expected effects, the objects that are relevant to this action include the robot, the kit tray, and the work table. Aspects of these items

- Object information: This includes an object taxonomy and class description of objects that are relevant to the autonomous system in accomplishing its task.

- Action information: This includes information on all possible actions that

A scenario was developed that described, in detail, the types of operations that would be performed in the target application, the sequencing of steps, the parts and machines that were needed, constraints on the process such as pre- and post-conditions, etc. For this scenario, a set of concepts were extracted and defined. These concepts served as the initial requirements for the knowledge that must be encoded for the domain. This knowledge was encoded in the form of a state variable representation (SVR). In a SVR, each state is represented by a tuple of values of n state variables $\{x_1, \ldots, x_n\}$, and each action is represented by a partial function that maps this tuple into some other tuple of values of the n state variables. The concepts that were modeled in the SVR, built off of the definitions and relationships that were identified in the scenario.

The design transitions from domain expertise to knowledge modeling expertise when the SVR is used to generate an ontology which consists of a base ontology that describes the objects in the scenario, extensions that describe the actions and predicates relevant to the scenario, and instance files that describe the initial and goal states for the system. The ontology files are described in more detail in Section 4.

From this point forward, automatic planning tools have been designed to transition the knowledge into a planning framework and finally a robot specific framework.

As such, it is assumed that a system has been implemented that is capable of basic robotic actions. In our case, our knowledge driven system bottoms out in a Robot Operating System (ROS) control layer [1].

3. Industrial Kitting

Material feeding systems are an integral part of today's assembly line operations. These systems assure that parts are available where and when they are needed during the assembly operations by providing either a continuous supply of parts at the station, or a set of parts (known as a kit) that contains the required parts for one or more assembly operations. In continuous supply, a quantity of each part that may be necessary for the assembly operation is stored at the assembly station. If multiple versions of a product are being assembled (mixed-model assembly), a larger variety of parts than are used for an individual assembly may need to be stored. With this material feeding scheme, parts storage and delivery systems must be duplicated at each assembly station.

[1]Certain commercial/open source software and tools are identified in this paper in order to explain our research. Such identification does not imply recommendation or endorsement by the authors or NIST, nor does it imply that the software tools identified are necessarily the best available for the purpose.

An alternative approach to continuous supply is known as kitting. In kitting, parts are delivered to the assembly station in kits that contain the exact parts necessary for the completion of one assembly object. According to Bozer and McGinnis [3] “A kit is a specific collection of components and/or subassemblies that together (i.e., in the same container) support one or more assembly operations for a given product or shop order”. In the case of mixed-model assembly, the contents of a kit may vary from product to product. The use of kitting allows a single delivery system to feed multiple assembly stations. The individual operations of the station that builds the kits may be viewed as a specialization of the general bin-picking problem [4].

In industrial assembly of manufactured products, kitting is often performed prior to final assembly. Manufacturers utilize kitting due to its ability to provide cost savings [5] including saving manufacturing or assembly space [6], reducing assembly workers walking and searching times [7], and increasing line flexibility [3] and balance [8].

Several different techniques are used to create kits. A kitting operation where a kit box is stationary until filled at a single kitting workstation is referred to as *batch kitting*. In *zone kitting*, the kit moves while being filled and will pass through one or more zones before it is completed. This paper focuses on batch kitting processes.

In batch kitting, the kit’s component parts may be staged in containers positioned in the workstation or may arrive on a conveyor. Component parts may be fixtured, for example placed in compartments on trays, or may be in random orientations, for example placed in a large bin. In addition to the kit’s component parts, the workstation usually contains a storage area for empty kit boxes as well as completed kits.

Kitting has not yet been automated in many industries where automation may be feasible. Consequently, the cost of building kits is higher than it could be. We are addressing this problem by proposing performance methods and metrics that will allow for the unbiased comparison of various approaches to building kits in an agile manufacturing environment. The performance methods that we propose must be simple enough to be repeatable at a variety of testing locations, but must also capture the complexity inherent in variants of kit building. The test methods must address concerns such as measuring performance against variations in kit contents, kit layout, and component supply. For our test methods, we assume that a robot performs a series of pick-and-place operations in order to construct the kit. These operations include:

1. Pick up an empty kit and place it on the work table.
2. Pick up multiple component parts and place them in a kit.
3. Pick up the completed kit and place it in the full kit storage area.

Each of these may be a compound action that includes other actions such as end-of-arm tool changes, path planning, and obstacle avoidance.

It should be noted that multiple kits may be built simultaneously. Finished kits are moved to the assembly floor where components are picked from the kit for use in the assembly procedure. The kits are normally designed to facilitate component picking in the correct sequence for assembly. Component orientation may be constrained by the kit design in order to ease the pick-to-assembly process. Empty kits are returned to the kit building area for reuse.

4. Ontology

We believe that building models of the world knowledge is a necessary step towards operating an automated kitting workstation. The proposed models include representations for non-executable information about the workstation such as information about parts, kits, and trays. The description of these models includes for instance the location, orientation, and relation between components. These models are discussed in Section 4.1.

Models of executable information are also produced from the system information described in the SVR. These models include actions, actions' precondition, actions' effect, and actions' failures that consist of different spatial relations. A description of these models is given in Section 4.2.

Section 4.3 and Section 4.4 describe the init and goal files, respectively, generated from the data structure and information in the models of the world knowledge.

The rest of this section describes the two proposed different models along with their representation in the ontologies presented in Figure 1. The models were defined in OWL's functional-style syntax [9].

4.1. The OWL Kitting Ontology

4.2. The OWL SOAP Ontology

4.3. The OWL Init Conditions File

4.4. The OWL Kitting Goal Conditions File

5. Implementation

6. Conclusion

References

[1] S. Balakirsky, P. David, P. Emmerman, F. Fisher, P. Gaylord, in: Proc. Symposium of the Association for Unmanned Vehicle Systems, pp. 927–946.

[2] J. Albus, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3260–3265.

[3] Y. A. Bozer, L. F. McGinnis, International Journal of Production Economics 28 (1992) 1–19.

[4] A. Schyja, A. Hypki, B. Kuhlenkotter, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 5246 –5251.

[5] O. Carlsson, B. Hensvold, Kitting in a High Variation Assembly Line, Master's thesis, Luleå University of Technology, 2008.

[6] L. Medbo, International Journal of Production Economics Journal of Industrial Ergonomics 31 (2003) 263–281.

[7] G. Schwind, Material Handling Engineering 47 (1992) 43–45.

[8] J. Jiao, M. M. Tseng, Q. Ma, Y. Zou, Concurrent Engineering: Research and Applications 8 (2000) 297–321.

[9] W3C, in: http://www.w3.org/TR/owl2-syntax/.