Knowledge is more than simply a collection of data or facts. In autonomic computing systems, knowledge is described as information that an autonomic manager can use. In this series of articles focused on the autonomic computing architecture, find out where knowledge fits in. The architecture -- the foundation on which self-managing information technology systems can be built -- defines five building blocks: autonomic managers, manual managers, touchpoints, knowledge sources, and autonomic computing usage patterns for an enterprise service bus. This column investigates knowledge sources -- and knowledge in general -- in more detail.

Introduction

The previous installment of The autonomic computing edge was the second in a series that focuses on the autonomic computing architecture. That article discussed an architectural building block called a touchpoint. This article continues the autonomic architecture series by examining the role of knowledge in self-managing autonomic systems, including the architectural building block called a knowledge source.

I'll describe what is meant by knowledge in the autonomic computing architecture, what some of the key types of knowledge are in self-managing autonomic systems, how people and systems can create knowledge, how autonomic managers use knowledge, and how a knowledge source facilitates all of this. So, let's walk right up to the edge of knowledge.

What is knowledge?

My favorite online dictionary (see Related topics for a link) contains several entries for knowledge. Two useful ones are:

- The state or fact of knowing
- The sum or range of what has been perceived, discovered, or learned

In an autonomic computing system, knowledge takes both of these forms -- knowledge captures something that is known (perhaps by humans), renders it in a standard way so that it can be used...
by autonomic managers, and that in turn enables new knowledge to be instantiated ("perceived," if you will) and, in sophisticated systems, enables new knowledge to be discovered and learned. I'll explore the details of this later in this article.

First, I need to distinguish knowledge in a self-managing autonomic system from knowledge that is well known in other computer science domains, particularly artificial intelligence. Dictionary.com has an entry specifically addressing this aspect of knowledge:

*artificial intelligence, information science* The objects, concepts and relationships that are assumed to exist in some area of interest. A collection of knowledge, represented using some knowledge representation language is known as a knowledge base and a program for extending and/or querying a knowledge base is a knowledge-based system.

Knowledge differs from data or information in that new knowledge may be created from existing knowledge using logical inference. If information is data plus meaning then knowledge is information plus processing.

A common form of knowledge, e.g. in a Prolog program, is a collection of facts and rules about some subject.

In the autonomic computing architecture, we do not use this same definition for knowledge, although there are some similarities. In self-managing autonomic systems, knowledge in general can be almost any sort of structured data or information that could be used to carry out processes, especially processes that can be automated. In this broad scope of knowledge, we include things such as the data stored in log files, state data about a touchpoint or a process, and scheduling data about when changes can be applied to a system.

However, the autonomic computing architecture -- and hence, this article -- focuses on a narrower strata of knowledge, namely the architected forms of knowledge used by autonomic managers. And the autonomic architecture goes one step further by focusing on the kinds of knowledge that can be used to express and to extend what an autonomic manager can do.

What does that mean? To answer that question, let's look at an autonomic manager. Figure 1 shows the main parts and functions of an autonomic manager.
The Autonomic Computing Architectural Blueprint (see Related topics for a link) describes an autonomic manager as:

an implementation that automates some management function and externalizes this function according to the behavior defined by management interfaces. The autonomic manager is a component that implements the control loop ... For a system component to be self-managing, it must have an automated method to collect the details it needs from the system; to analyze those details to determine if something needs to change; to create a plan, or sequence of actions, that specifies the necessary changes; and to perform those actions. When these functions can be automated, an intelligent control loop is formed.

This definition describes the Monitor-Analyze-Plan-Execute functional composition of an autonomic manager. These key functions are well known and sometimes referred to as a MAPE loop. However, Figure 1 also shows that, in addition to the MAPE functions, an autonomic manager also contains a knowledge block. Notice that the knowledge block is connected to all four of the MAPE functional blocks. This connection indicates that knowledge is used throughout the typical autonomic manager, so you should not overlook the importance of knowledge in a self-managing autonomic system. Indeed, we'll render some of this knowledge more explicit next.

**Knowledge types**

The Autonomic Computing Architectural Blueprint cites three examples of broad types of knowledge:

- Solution topology knowledge (such as that represented by installable unit descriptors)
- Policy knowledge
- Problem determination knowledge (such as monitored data and symptoms)

If you add the specific, architected types of knowledge used by autonomic managers (the knowledge that the autonomic computing architecture focuses on) to Figure 1, the result is Figure 2:
Figure 2. Autonomic manager and knowledge

Figure 2 shows policy knowledge and topology knowledge (which could be represented by an Installable Unit Deployment Descriptor) that are discussed in the Blueprint. Figure 2 also illustrates knowledge that is logically passed among the MAPE functions of an autonomic manager:

- **Symptoms**, created by the Monitor function by correlating monitored data such as Common Base Events. Symptoms indicate some condition associated with one or more managed resources.
- **Change Requests**, created by the Analyze function by scrutinizing symptoms and other data. Change Requests indicate some sort of desired alteration to one or more managed resources.
- **Change Plans**, created by the Plan function by encoding some set of steps or processes used to carry out a Change Request. Change Plans indicate how to accomplish some change for one or more resources.

Although knowledge such as symptoms, change requests, and change plans can be considered problem determination knowledge, as described in the Blueprint, the use of these knowledge types is not limited to problem determination. Symptoms, change requests, and change plans might be used in self-protecting or self-configuring scenarios, for example. See the previous installment of "The autonomic computing edge: Can you CHOP up autonomic computing" for additional background about how the knowledge types described here could be used in various ways.

The knowledge types discussed here are not an exhaustive list of the ones that are relevant in autonomic computing, but they are the knowledge types we focus on for use by autonomic managers. The next sections more fully describe how this knowledge can be created, shared and used by autonomic managers.
Knowledge source

According to the Blueprint, "A knowledge source is an implementation of a registry, dictionary, database or other repository that provides access to knowledge according to the interfaces prescribed by the architecture."

When knowledge needs to be shared, it is put into a knowledge source. The knowledge source is the place where you store the system knowledge -- policies, symptoms, and so on -- so that it can be shared among autonomic managers and manual managers, as shown in Figure 3, taken from the Blueprint.

**Figure 3. Knowledge sources in a self-managing autonomic system**

The knowledge sources hold knowledge created by humans or by automated processes. Autonomic managers and manual managers can retrieve that knowledge and use it to carry out functions that they perform, such as recognizing symptoms or enforcing policies. The use of knowledge by autonomic managers is detailed later. First, I examine how the knowledge gets into a knowledge source.

**Authoring and deploying knowledge**

Today, knowledge that is useful for systems management exists in many places. Often, much of the knowledge exists in the minds of developers, administrators, and others associated with components of the IT infrastructure. Capturing that knowledge in a standard form enables automation of some systems management tasks.
The developer of a touchpoint (see the previous installment, "The autonomic computing edge: Keeping in touch with touchpoints" for more background about touchpoints) can enhance the manageability of that touchpoint by capturing and making available knowledge about the touchpoint. Two examples of such knowledge are symptoms and installable unit deployment descriptors (IUDDs).

The touchpoint developer may know that a series of events in a particular sequence, or in a particular time period, is symptomatic of some condition in the resources that might indicate a problem (perhaps insufficient memory at run time, or inappropriate configuration values, or a broken communications link). These conditions can be encoded as Symptom knowledge, using a standard way to express the conditions that must exist for the symptom to occur, along with information about the symptom itself (what condition it represents, what resources it is associated with and so on).

The touchpoint developer also may know about certain considerations associated with the resources that need to be taken into account during installation of the touchpoint (perhaps required prerequisite or co-requisite components, or required hardware capacity). These dependencies can be encoded as solution topology knowledge in an IUDD, using a standard way to express the dependencies and checks that should be performed during installation of the touchpoint.

This and other knowledge captured and encoded in a standard form by touchpoint developers (and others) can be deployed to a knowledge source so that it can be used for autonomic self-management. After the knowledge is deployed to the knowledge source, it can later be used by autonomic managers and manual managers to manage the touchpoint. As one example, the IUDD knowledge could be used to install the touchpoint and the Symptom knowledge could be used to recognize symptoms that occur during operation of the touchpoint (so that appropriate changes can be made in response to those recognized symptoms).

**Autonomic manager use of knowledge**

So, just how does the autonomic manager use this knowledge? Figure 4 illustrates this process for Symptom knowledge, as described in the previous example.
The autonomic computing edge: The role of knowledge in autonomic systems

Figure 4. Autonomic manager use of knowledge example

The touchpoint creator first authors and deploys the Symptom knowledge to a knowledge source. Second, at some later time, an autonomic manager that has been assigned to manage this touchpoint may load this symptom knowledge from the knowledge source. Typically, the autonomic manager will not automatically begin to use this loaded knowledge; instead, its manager (represented in Figure 4 as a manual manager for step 3) must activate the knowledge in the autonomic manager. This is how the management activity associated with this knowledge is delegated from a human activity to an automated one in the autonomic manager. Finally, after the activity has been delegated, the autonomic manager can begin to use the relevant knowledge. In this example, that means that the autonomic manager can begin to recognize symptoms (shown in Figure 4 as step 4, using a correlation engine in the Monitor function) as the conditions associated with those symptoms -- first encoded by the touchpoint creator in Step 1 -- occur and are monitored.

All of this leads to two fundamental ways that knowledge is used by, and is very important to, autonomic managers:

- Autonomic managers express what they can do in terms of knowledge. In the example here, the autonomic manager could indicate that it is able to recognize symptoms, and moreover, that it can recognize particular symptoms associated with particular conditions, as encoded in the symptom knowledge.
- Autonomic managers can extend what they can do by loading new knowledge, in a standard form, from a knowledge source. In the example here, the autonomic manager could load other symptoms, generated by the touchpoint creator and all in a standard form, from the knowledge source. After activation, the autonomic manager could begin to recognize these
additional symptoms, associated with other conditions, as they occur during the resources' operation.

**Conclusion**

The autonomic computing architecture focuses on certain types of knowledge that autonomic managers can use. This knowledge offers a way to express what autonomic managers can do and extend what autonomic managers can do. Several well known types of knowledge fit this model, including symptoms, change requests, and change plans.

Knowledge can be captured in a standard form and deployed to a knowledge source, where it can then be used by autonomic managers and manual managers for autonomic self-managing functions, all related to the idea that knowledge enables autonomic managers to express and to extend what they can do.

**Future installments of The autonomic computing edge will examine other architectural building blocks and how they can be composed together to form self-managing autonomic systems.**
Related topics

• Get word definitions at Dictionary.com.
• Read more about autonomic computing self-management in the June 2005 issue of The autonomic computing edge column on developerWorks.
• Read more about touchpoints in the previous issue of The autonomic computing edge column on developerWorks.
• Don't miss the other articles in this series.

© Copyright IBM Corporation 2005
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)