Autonomic Computing and Policy Based Systems

Bel G Raggad and Rahim Choudhary

Abstract

Autonomic computing and policy based systems are closely related concepts. They are examined in the light of deployment experiences with these technologies in the Government and the industry projects. The paper identifies the difficulties encountered in the practical development and deployment of these technologies, as well as possible approaches to overcome these difficulties. A solution approach is suggested that borrows capabilities from multi-agent artificial intelligence systems, as they exist in Foundation for Intelligent Physical Agents (FIPA) which is study group 11 of IEEE. This is a new approach that remediates the non availability of certain protocols and protocol implementations in autonomic computing area. The approach is illustrated via the formulation of an architecture for the policy rule-sets which are needed in all autonomic computing systems and policy based systems.

Keywords— Policy Rule-Sets, Policy Based Systems, Autonomic Computing, Policy Based Management, multi-agent, artificial intelligence, Protocols, specification, implementation, IPv6++.

1. Introduction

Autonomic Computing [1] and the Policy Based Systems [2] concepts have existed for some time. They offer a great promise for automating decisions about the management actions to be taken in a given operational context, and subject to well-defined Policy Rule-Sets [2]. Many challenging and large projects in the Government and industry have sought to implement these technologies in order to handle the operational complexities of collaborative infrastructure, automate the decision making processes, make the system response dynamic in the near real time, and to save costs in the long run. Examples of such projects include the Global Information Grid (GIG) project [3] in the Department of Defense (DoD), the Transformational Communications Architecture [4] in the National Security Space Office (NSSO), the Transformational Satellites (TSAT) project [4] at the Joint Program Management Office (JPMO), the IPv6 Transition project [5] at the Defense Information Systems Agency (DISA), Policy Management for Autonomic Computing (PMAC) initiative [6] at the IBM, and the IPv6++ initiative under the project for Exposing the features of IPv6 for building Autonomic Networks and Services (EFIPSANS) [7].

This paper uses analysis based on the experiences with the above projects, with the exception of the last. It identifies some lessons learned from those experiences: namely the practical difficulties that currently exist in the development and deployment of these technologies, and possible solution approaches to overcome these difficulties.

The solution approach is novel in the sense that it borrows those capabilities from the area of multi-agent systems in artificial intelligence that do not currently exist in the autonomic computing area. Specific instances of these capabilities are illustrated while presenting an architecture for the policy rule-sets. Such architecture does not currently exist, though it is needed in all autonomic computing systems and policy based systems.

Section 0 presents some essential concepts and definitions needed for a meaningful discussion of these technologies. Section 0 presents the practical difficulties that arise in developing these technologies, and a solution approach to these difficulties is presented in section 0. A detailed architecture for the policy rule-sets is presented in section 0 and our conclusions are given in section 0.

2. Clarification of concepts

First we define some fundamental concepts to make the discussion concrete. This is because overlapping concepts do exist in this arena and misunderstandings can arise among the researchers in the field.

Policy Rule-Sets: The term policy rule-sets applies to what the national security agency calls the Digital Policies [8]. They are derived from the analysis of the executive policies promulgated by the people or institutions of authority. Digital policies use policy rule-sets for their formulation. Policy rule-sets are specified in a machine interpretable language, such as the eXtensible Markup Language (XML), that a computer can evaluate to arrive at *unique* and *unambiguous* decisions with respect to the operational action to be taken.

Policy Based Systems (PBS): The term policy based systems is used for any software system that is based on the policy rule-sets and conforms to some notional architectural aspects [9]. Some examples of the application areas for the PBS technology are the decision support systems, management systems, financial systems, situation awareness systems, common operating picture determination systems, quality assurance systems, and access control systems. A PBS achieves automation by replacing man in the middle of the operations by a software module that implements policy rule-sets and operates autonomously. However, it is an error to think that PBS is just automation. To definitively distinguish PBS from automation we give the following two criteria for a PBS: (1) a PBS will achieve automation through the deployment of policy rule-sets; and (2) a PBS will dynamically adjust to the changes in the operational conditions in the near-term and to the evolution in the operational requirements over the long-term. A PBS can accomplish this by changes in the policy rule-sets without any changes to the software code for the system. A poor analogy is the DATA STATEMENT extension that IBM introduced into FORTRAN, which allowed to change the behavior of the software via changes only to the values in the DATA STATEMENT. However, the advantages offered by the policy rule-sets are very large compared to the above example.

Policy Based Management (PBM): A policy based management system is a special type of PBS that focuses on management applications versus a much larger scope for the PBS applications. Even the PBM systems have widely different management capabilities depending upon what it is that the system manages. Examples of a PBM system application areas include the network management, security management, quality of service management, resource management, operations management, and enterprise management.

Autonomic Computing: The term is popularized by IBM [1]. It makes an implicit reference to the ‘autonomic nervous system’ within the human body, which governs involuntary body functions like respiration and heart rate. So the abstract goal of autonomic computing refers to autonomic computer systems with a minimum of human interference. The concept is rather amorphous because it does not specify what a ‘minimum human interference’ means, nor does it specify how the amorphous goal is to be achieved. In practice, these two questions are answered much the same way as they are answered in a PBS; namely via the use of policy rule-sets. The concept of a PBS predates that of Autonomic Computing; IETF has specified the PBM technology in detail, while a rather amorphous concept of autonomic computing is mainly promulgated by IBM.

The above definitions can clarify some potential misunderstandings. For example, a script or work-flow based approach can achieve automation but it will not achieve the second capability mentioned above for a PBS.

Similarly, a PBS that uses policy rule-sets is very similar to a rule based artificial intelligence (AI) system. The difference is in the way a rule-based system or a multi-agent AI system is architected and implemented; and that does not fulfill the second requirement given above for a PBS. Having stated this, it should not be overlooked that there are potential cross fertilizations between PBS technology and AI technology. We will elaborate this point further in section H.

Another potential confusion is between an autonomic computing system and a policy based system. Theoretically speaking, the two can have different but overlapping scopes. However, in practice autonomic computing and a PBS both deploy policy rule-sets using similar architectures; and the implementation of an autonomic computing system and a PBS is practically very similar. Both need all the capabilities with respect to the policy rule-sets and their management using a Policy Management Infrastructure (PMI) [10]. Hence, in practice one can use the two terms interchangeably.

3. Difficulties in implementing a PBS

Based on our experience with the projects that seek to use policy based systems, as was stated in the 1. Introduction, we describe below the difficulties that exist in implementing a PBS or an autonomic computing system.

A. Protocols

IETF has not pursued work in the policy based management area since many years. Thus there are no protocols for many needed capabilities such as to communicate between multiple policy decision points (PDPs), to distribute policies, to deconflict policies, and to deploy, activate, deactivate, evaluate, and execute policies. Another important capability is to audit policy performance and to forecast future policy modifications; no protocols exist for a PBS to perform policy audits.

B. Protocol Implementations

For the protocols that do exist, there is little effort spent at implementing them. For instance, IETF has specified the common open policy service (COPS) protocol and its multiple applications but no implementations for them are available. The COPS protocol is vital for PDP-PEP (policy execution point) communications and for policy provisioning.

C. PMI Products

No robust products currently exist that implement a Policy Management Infrastructure (PMI) [10]. Further, no satisfactory language exists to define policy rule-sets and to support functions that a policy object needs to support [10]. Such difficulties led the GIG project [3] to scale down expectations and to look for scripting or work-flow based solutions.

D. Vendor Support

There are many IETF specifications for the Policy Information Bases (PIBs) using the policy information models but the vendor support for them are sadly lacking.

E. Experience

There is no robust experience base in the industry to develop mature applications of PBS or autonomic computing. For example, IBM took the initiative to develop the PMAC system [6]. They circumvented the lack of COPS protocol implementation and the absence of an industry wide policy definition language by using the web services definition language and the IBM WebSphere product. While the system works, it is a barebones implementation that looks more like a prototype with place holder stubs for the policy rule-sets. The result is that, even IBM did not successfully leverage PMAC in their server offerings, or even in a more modest application, namely the network management products from their Tivoli subsidiary.

F. Policy Execution Points

The policy execution points (PEPs) constitute the capability where the rubber hits the road. The PEPs are generally implemented by the vendors of the managed elements as part of their commitment to a standard based element management system. It is similar to the vendor support for an SNMP agent. However, there is currently no vendor support for the PEPs.

There are other difficulties like the misconceptions about a PBM system capabilities, and unsubstantiated vendor claims about their products being PBM systems. To illustrate this point, we provide two examples: (1) during the TSAT program [4] there was a general attitude that regarded PBM as a 'magic capability' which could perform functions that were not even defined; (2) vendors like Cisco claim that their router management uses PBM; whereas in reality it provides merely a GUI to specify values for the router configuration parameters, with no PMI type capability provided. The result is that, if you use the GUI to specify parameter values for a router and want to use identical configuration settings for another instance of the same router, the configuration settings cannot be automatically distributed to that router instance; instead, you need to go through the configuration settings all over again using the GUI on that instance of the same router.

4. Possible Solutions

The difficulties in implementing a PBS or an Autonomic Computer system were discussed in section 0. These were based on lessons learned from the projects as stated in the 1. Introduction, as well as some additional ones. The lessons learned also suggest solutions to the difficulties. Some solution approaches are described below.

G. Gradual Implementation of the PMI

Reference [10] discusses two modes of deploying a PBM system, namely an ad-hoc mode and a PMI based mode. The difficulties associated with the lack of protocol specifications or their implementation can sometimes be circumvented by using the ad-hoc mode at initial stages, and to gradually migrate to a PMI based mode later on.

H. Borrowing from AI Capabilities

The lack of implementation for the COPS specifications and the lack of availability of an industry wide language to distribute policy rule-sets can be circumvented by using the corresponding capabilities from the AI discipline as they obtain in the IEEE FIPA (Study Group 11) model for agent based computing [11]. Examples of such capabilities include the Agent Communication Language (ACL), the Message Transport Service (MTS), and the capability of an Agent Platform to use non-agent software to support agent functions for new services such as new communication protocols, security protocols and algorithms.

This approach is rather new in the cross fertilization from the AI technology to the PBS technology. The basic idea is to define the policy rule-set structure (see section 0) using XML with well defined semantics and ontology. Such structures can then communicate among themselves using ACL, and they can be transported across platforms using MTS. Further, PBS elements like the PDP and PEP are agents in this approach; they can communicate using ACL and MTS thereby circumventing the use of COPS, an implementation for which does not exist. This approach provides concrete mechanisms to define policy rule-sets, communicate them from one agent (say a PDP agent) to another agent (say a PEP agent), and distribute them across networks (thereby circumventing the non-availability of a protocol for distribution of policy rule-sets).

Feasibility of the basic operations in this approach has been demonstrated in the context of building security in an IEEE FIPA compliant multi-agent system [12]; and a prototype PBM system was successfully developed under a project to implement the risk adaptable access control model [13].

Incidentally, this approach also partially mitigates for a lack of development experience in the PBS industry because relevant industry experience exists in multi-agent systems community. Further, this approach will remedy to some extent the problems due to a lack of vendor commitment in the PBM industry because one will now deal with the multi-agent systems vendors.

I. IPv6++

The IPv6++ approach [7] is ambitious. It attempts to use the extensibility features in IPv6 to gradually and steadily make it more and more autonomic. Examples of the IPv6 extensibility features include the introduction of new extension headers, the introduction of new options in existing extension headers, specification of new protocols, and end to end addressability. These extensibility features are two edged: they are good for future evolution of the protocol, and they are also potential sources of difficulties in areas such as security [14].

It is easy to see this point by looking at past examples in IPv6 specifications. The autoconfiguration in IPv6 is a significant step in the direction of autonomic operations. This protocol is new with respect to IPv4 but it has also given rise to numerous potential security risks [14]. Similarly the processing of the options in the extension headers and the possibility of defining new extension headers can produce addition security vulnerabilities as well as potential disruptions in operations [14].

The primary reason for IPv6 specification was the vanishing availability of IPv4 addresses. A very large number of IPv6 addresses solves this problem, and at the same time supports potentially powerful new architectures that exploit the end to end addressability in IPv6. A concrete example of such architecture is used in a new security model for IPv6 [15].

As discussed above, there are opportunities as well as unintended consequences that arise in defining new protocols and headers for IPv6. There are lessons learned while working on IPv6 transition for the department of homeland security and the defense information systems agency. They imply that it is better to focus on developing a specific application for the PBM technology versus pushing a generalized framework that might run into scope-creep problems.

J. Policy Rule-Sets

All autonomic computing systems and policy based systems use policy rule-sets. The heart of an autonomic computing system or a policy based system is therefore a collection of policy rule sets. No architecture currently exists in the literature regarding what a policy rule set looks like, how different policy rule sets are combined, and how they are communicated and distributed. These questions are an important part of any solution approach. A detailed discussion of these architectural questions is provided in section 0 below.

5. Architecture for Policy Rule-Sets

Policy rule-sets can be defined as data structures using a language like the XML. They can communicate among themselves and with a Management Agent using a communication language.

K. Policy Rules

A policy rule is expressed in the form: IF X THEN Y. Here X is a condition, referred to as a policy condition; and Y is an action, referred to as a policy action. A policy condition can be a composite condition that incorporates many conditions; and a policy action can similarly be a composite action that incorporates many actions. The order of conditions in a composite condition and the order of actions in a composite action is in general important.

As stated earlier, a policy rule can be defined as a data structure. In this architecture the structure consists of four modules discussed in the following subsections.

1) Metadata

The metadata in a policy rule constitute the credential of the policy rule itself. The metadata module contains:

- Organization ID: It signifies the organization and sub organization where the PBS incorporating the particular policy rule is currently deployed.
- Technical Point of Contact: It represents an individual within the organization who can provide further details about the particular policy rule such as its Developer ID, Deployment History, and Documentation etc.
- Authorization Point of Contact: It represents the individual who authorized the deployment of the particular policy rule and can provide further information on the continued validity of its deployment in special situations. The individual can trace the complete chain of authority under which the policy rule was deployed, modified, updated, superseded, activated, or deactivated.
- Date of Expiry: It shows the last day of validity for the particular policy rule.
- Private Extensions: These are additional data that the organization may decide to include.

Metadata need not be explicitly communicated during the operational transmissions of the policy rule. For example, when a policy rule is distributed to a PDP for deployment, this transmission may not include the metadata. All other modules of a policy rule are transmitted.

2) Status

The status module consists of a set of values that describe the current status of a policy rule. For example, it can describe whether the policy rule is deployed, activated, deactivated, expired, etc.

3) Data

The data module contains the specific parameter values that are needed to evaluate and execute the policy rule. These data must be protected against unauthorized access and change.
Data module of a policy rule contains three types of data:

1. *Condition Data*: These describe the parameters that specify a policy condition. If the condition is a compound condition, the set of all condition parameters are included in the data module.
2. *Action Data*: These describe the parameters that specify a policy action. If the action is a compound action, the set of all action parameters are included in the data module.
3. *Logic Data*: These are the parameters introduced by the policy rule logic. These are the programming parameters introduced in the logic module (see below) of a policy rule. An example of a logic data is the parameter n discussed in the example below.

Consider a simple example. The policy rule may assert that an account be locked after n unsuccessful login attempts. A specific value of the parameter n will be included in the data module. The value can be specified using common methods such as the tag-length-value (TLV) format.

It should be noted that the behavior of a policy rule can be changed by merely changing the parameter values in the data module. Further, some values may signify triggers for special behaviors of the policy rule. For example a value of n=1000 may indicate that no account will be locked after unsuccessful attempts are made at login, i.e. the policy rule is effectively deactivated.

4) Logic

This module contains the executable instructions that contain the decision logic for the evaluation of the policy rule. In the above mentioned example, the module would contain logic that compares the number of unsuccessful login attempts with a specified value. The logic will define the parameter n and acceptable values for this parameter; it will also access its data module to retrieve the currently specified value for the parameter. Needless to say that policy logic must be protected against unauthorized access and change.

This module can incorporate any degree of complexity in the logic that evaluates the policy rule. Further, the logic is not limited to algorithmic logic. Special pattern recognition techniques, the neural network techniques, communities of mobile intelligent agents, or other techniques can be freely used as needed.

The logic module will also check the validity of the policy rule, such as verification of its security credentials, checking for security hashes for integrity during transmission, and checking the expiry date. However, this validation logic need not be part of every policy rule. Instead, a general validation policy rule may be developed that is automatically invoked in conjunction with the evaluation of all policy rules.

L. Policy Rule-Sets

As the terminology suggests, a policy rule-set is a set of policy rules. Policy rules are nested together to make a policy rule-set. The order of policy rules in a policy rule-set is in general important. It is analogous to a programming function that calls upon other previously defined functions.

A policy rule-set has its own metadata, status, data, and logic modules. The logic module for the policy rule-set specifies the process of including the policy rules within the policy rule-set, as well as the logic of combining the included policy rules.

It is a matter of implementation as to whether the policy rules included in a policy rule-set are incorporated via the inclusion of the body of the policy rules or via appropriate logical references to those rules. It would appear that the use of reference pointers would be more efficient from the perspective of storage, bandwidth, and synchronization.

A data value may be allocated in the data module for the included policy rule as well as in the data module for the policy rule-set. The situation can offer an advantage. If it is specified that the value in the data module for the policy rule-set takes precedence, it provides a mechanism to override the earlier values for the included policy rules. For this to be workable, such data values provided in the data module for the policy rule-set must be specified using a fully qualified path that unambiguously points to the corresponding policy rule for which the data value is being overridden.

M. Compound Policy Rule-Sets

Like the Chinese boxes, policy rule-sets can be nested together to make a compound policy rule-set. A compound policy rule-set will have its own metadata, status, data, and logic modules, as discussed earlier. The compounding process and the compounding logic is part of the logic module for the compound policy rule-set. The policy rule-sets incorporated in a compound policy rule-set will be automatically retrieved during the evaluation of a compound policy rule-set.

Let us illustrate the use of a compound policy rule-set through a simple example. Let us assume a Security Condition (SecCon) that takes Green, Yellow, Orange and Red values. Let us assume that a separate policy rule-set is defined for each one of these security conditions. The logic module in the compound policy rule-set can simply consist of reading the current setting for the SecCon parameter from its Data module and sending the operational pointer to one of the four previously defined policy rule-sets.

The logic module of a compound policy rule-set implicitly specifies what policy rule-sets are incorporated in it, and need to be retrieved.

The level of nesting can be arbitrarily deep as a compound policy rule set can be incorporated into another compound policy rule-set. However, a very deep nesting can give rise to unintended behavior from policy rules that are incorporated during the compounding or nesting process.

N. Meta Policy Rule-Sets

The SecCon example of a compound policy rule-set that was discussed above is a rather special case. It is special because ultimately the logic of the compound policy rule-set invoked only one policy rule-set for one of the four SecCon values. Thus only one policy action was obtained.

In general, compounding can give rise to the execution of more than one policy rule-sets. This can result in multiple policy actions that the compound policy rule-set may have to arbitrate.

Let us illustrate the situation using an example. Let us assume that the compound policy rule-set invokes a traffic engineering policy rule-set (QA-Policy) and a security policy rule-set (IA-Policy). Let us further assume that the underlying network is multi-protocol label switched (MPLS) type with two label switched paths available to the traffic in question: Path-1 (QA-high, IA-unclassified) and Path-2 (QA-low, IA-secret). The traffic engineering policy rule-set decides to use Path-1 and the information assurance policy rule-set decides in favor of Path-2. Along which label switched path the traffic should flow?

The example of label switched paths given above demonstrates the need for Meta Policy Rule-Sets. These policy rule-sets generally arise from higher level executive decisions compared to application specific or domain specific decisions. We need a meta policy rule-set that would arbitrate between the QA-Policy and the IA-Policy in the above example. Let us assume in this case that the meta policy rule-set gives priority to the security policy rule-sets. The traffic will flow along Path-2 which does not satisfy the quality assurance requirement under traffic engineering but does satisfy the information assurance requirement under security. Such a meta policy rule-set would have to be developed and it would need to be invoked in conjunction with the compound policy rule-set in the above example.

6. CONCLUSION

This paper has presented practical considerations for the development and deployment of autonomic computing systems and policy based systems. It has presented the practical difficulties encountered in the development and deployment of these technologies, and some solution approaches to overcome these difficulties. The paper further facilitates a solution by formulating a detailed architecture for policy rule-sets that are used in all autonomic computing systems and policy based systems.

REFERENCES

[1] IBM, Autonomic Computing, http://www.research.ibm.com/autonomic/index.html

[2] John C. Strassner, “Policy-Based Network Management: Solutions for the Next Generation”, ISBN 1-55860-859-1, Morgan Kauffman Publishers, 2004.

[3] Department of Defense (DoD) Chief Information Officer (CIO), ‘Department of Defense Global Information Grid Architecture Vision – Vision for a Net-Centric, Service Oriented DoD Enterprise’, Version 1.0, June 2007.

[4] T. Meink, Office of the Assistant Secretary of Defense Networks and Information Integration, ‘Transformational Communications Systems for DoD Net-Centric Operations’, *CrossTalk: The Journal of Defense Software Engineering*, p. 23-25, July 2006.

[5] Federal Chief Information Officer (CIO) Council, Architecture and Infrastructure Committee, “IPv6 Transition Guidance”, February 2006.

[6] IBM, Policy Management for Autonomic Computing (PMAC) http://www.alphaworks.ibm.com/tech/pmac

[7] European Commission, CORDID, FP7, Information and Communication Technologies (ICT) Projects http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=85542

[8] NIST/NSA, NIST IR 7657, “A report on the Privilege (Access) Management Workshop”, March 2010.

[9] A. R. Choudhary, “Policy based network management”, Bell Labs Technical Journal Vol 9, No. 1, June 2004.

[10] A. R. Choudhary, “Digital Policies as a Tool for Transformational Communications”, Proceedings of the IASTED conference on Communications, Internet and Information Technology (CIIT), held at Cambridge MA, during Oct 31-Nov 2, 2005.

[11] Foundation for Intelligent Physical Agents (FIPA) Specifications http://fipa.org/specs/fipa00023

[12] A. R. Choudhary and J. B. Odubiyi, “Building Security into an IEEE FIPA Compliant Multi-Agent System”, Proceedings of 8th IEEE Workshop on Information Assurance and Security held at United States Military Academy, West Point, NY, during June 20-22, 2007, p. 49-55.

[13] SEGMA Technologies Inc., “Prototyping a Policy-based Management System using Societies of Collaborative Intelligent Agents for Managing Risk-Adaptive Access Control Transactions”, NSA-BAA-001-10.

[14] A. R. Choudhary, “In-Depth Analysis of IPv6 Security Posture”, Proceeding of the Trusted Collaboration Workshop at the 5th International Conference on Collaborative Computing, Crystal City, VA. Nov 11-14, 2009.

[15] A. R. Choudhary, “Securing IPv6 Network Infrastructure: A New Security Model”, Proceedings of 4th IEEE International Conference on Technologies for Homeland Security, Waltham, Massachusetts USA, Nov 8-10, 2010.