

Autonomic Computing Design Framework

by Youssif Al Nashif

We are currently developing a general design methodology for autonomic computing systems and applications. Our approach is based on attaching a control and management software to each hardware resource or software component such that this resource/software behaves as an autonomic component; that means it can be dynamically and automatically re-configured (self-configuration), seamlessly tolerate any component failure (Self-healing), automatically detect attacks and protect against them (Self-protection), and automatically change its configuration parameters to improve performance once it deteriorates beyond certain performance threshold (Self-optimization).

Figure 1 shows our approach to autonomize any software module or resource by adding two software modules: Observer and Controller. Each autonomic component has its own Controller that is delegated to manage its execution and enforcing the operational policies. The Observer is used for sensing and analyzing the monitored data of the component whenever is required. In fact, the Observer and Controller pair provides a unified management interface to support self-optimizing, self-healing, self-protecting and self-configuring management services by continuously monitoring and analyzing the execution of the component in order to dynamically plan the appropriate corrective actions and then execute them to change the execution environment to meet the component requirements at runtime.

By using Autonomia, self-management, self-healing, self-optimizing, self-protecting, can be provided to any cyber-space component. The Observer monitors and gathers data about the component and then analyzes them to form the knowledge required by the controller to enforce the policies. When a component behavior triggers a certain policy, then the controller plans and executes the set of actions associated with that policy.

Network self-healing; for example, will be achieved by continuously monitoring and analyzing the behavior of resources and their interactions in the cyber-space. Any deviation from the normal behaviors base line will be detected by the Observer and based on that a set of policies will be triggered, and the required actions will be taken to bring the network back to its operational state.

Observer

The Observer monitors the component and its context by using sensing interface and analyzes that information to determine more information about the component such as the state and it stores that set of information into its knowledgebase. Each observer is defined by a configuration file. The observer's configuration file is composed of three main sections:

- Configuration Section: This section defines the configuration attributes that are required for the observer to function correctly and the list of controllers who are allowed to access this observer. One example is the amount of memory needed for the observer to operate correctly.
- Sensors Section: This section defines how can the observer collects the monitored data, what is the type of that data, and how often the monitored data is collected.

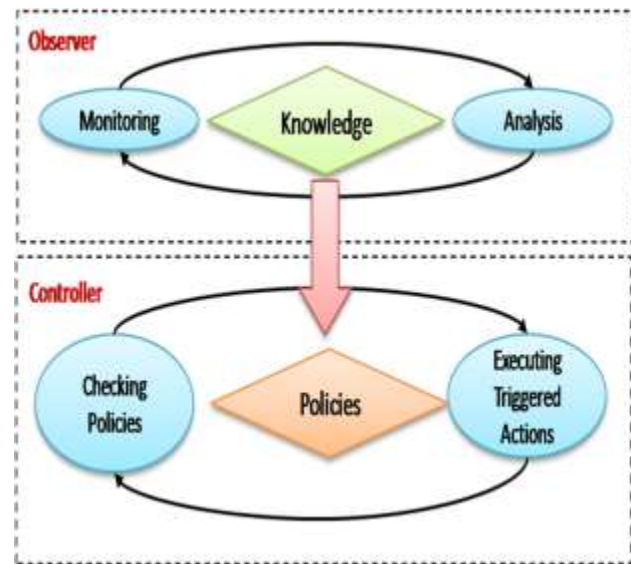


Figure 1: Autonomic Component Model

- Analyzers section: This section defines what analyzers are associated with this observer, where is the data that is used by each analyzer, what is the type of each analyzer, etc.

Controller

The controller checks to see if any policy (the *condition in the 'IF condition THEN action*) is triggered with gathered knowledge about the managed component. If a policy is triggered, the controller picks up the *action(s)* corresponding to that *policy* (condition) and executes that action(s) on the autonomic component using its actuators (control APIs). These lists of policies are stored in the policies database in the controller. Note a policy can be assigned a priority level. Each controller is defined by a configuration file.

The controller's configuration file is made of the following two main sections:

- Configuration Section: just like in the observer, this section defines the configuration attributes that are required for the controller to function correctly.
- Policies Section: This section defines the rules and the limits which the managed system must obey, and the actions that should be triggered to guarantee the desired operation of the managed system.

Figure 2 shows the planned deployment of Autonomia, and how it can be used to convert the managed system (domain) to be autonomic. It also shows how to convert a whole environment to operate as an autonomic environment by using local controller (domain controller) combined with a global controller (environment controller).

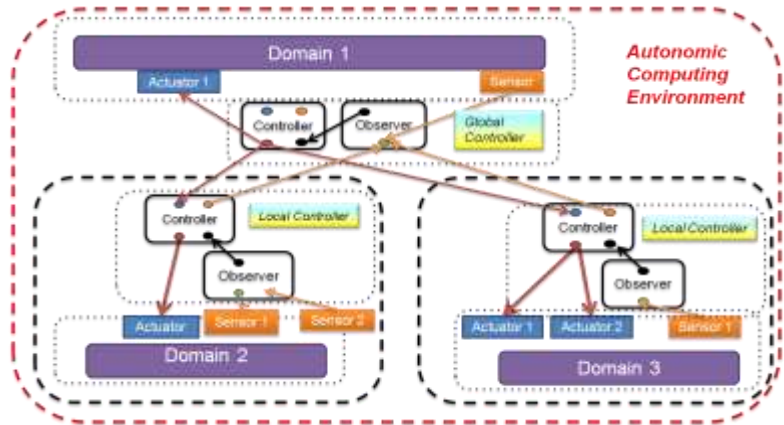


Figure 2: Autonomic Component Environment