

Autonomic Computing: The Next Era to Designing Self* Systems and Applications

by Salim Hariri



Advances in computing and communication technologies and software tools have resulted in an explosive growth in networked-applications and information services that cover all aspects of our life. These services and applications are inherently complex, dynamic and heterogeneous. Similarly, the underlying information infrastructures such as the Internet are complex, heterogeneous and dynamic. This combination exacerbates complexities related to application development, configuration and management, and makes current computing paradigms brittle and inefficient. As a result, applications, programming environments and information infrastructures are rapidly becoming unmanageable, insecure and inefficient when handling runtime changes. This has led researchers to consider alternative programming paradigms and management techniques that are based on strategies used by biological systems to deal with complexity, dynamism, heterogeneity and uncertainty.

programming paradigms and management techniques that are based on strategies used by biological systems to deal with complexity, dynamism, heterogeneity and uncertainty.

Autonomic computing is inspired by the human autonomic nervous system that handles complexity and uncertainties, and aims at realizing computing systems and applications capable of managing themselves with minimum human intervention. In this paper we first give an overview of the architecture of the autonomic nervous system and use it to motivate our approach to develop the autonomic computing paradigm. We then illustrate how this paradigm can be used to control and manage complex applications.

The Need for Integration and Automation

The control and management of computing systems have evolved from an environment in which a single process running on a computer system to a large, complex and dynamic environment in which multiple processes running on geographically dispersed heterogeneous computers that could span several continents (e.g., Grid). The techniques to design computing systems and services that meet their requirements have been mainly ad hoc. Initially, designers focused on developing efficient parallel processing and high performance architecture to improve system and application performance. As the deployment of computing systems and applications spread to many areas, especially those where failures can be catastrophic and life threatening, the reliability and availability of such systems and applications become a major concern. This requirement has driven separate research activities that have focused on reliability and fault tolerance computing. In a similar manner, the research in computing security has mainly addressed the needs to protect the integrity and the confidentiality of computing systems and their services without consideration to other important system attributes such as performance, reliability, and configuration. Consequently, this has led to the development of specialized and isolated computing systems and applications that can efficiently optimize a few of the system attributes or functionalities, but not all of them.

However, next generation computing systems and applications need to run fast, reliably, securely and cost-effectively. The ad-hoc integration (Figure 1) of the techniques that have been developed to manage performance, security, and fault-tolerance results in systems that are costly, insecure and unmanageable, as actions performed by the security engines, for example, might cancel actions taken by high performance computing engines. Furthermore, the performance, security and fault tolerance requirements of such systems and applications might change continuously at runtime. Hence, it is essentially critical for next generation computing systems and/or software architectures to be holistic in addressing systems and applications requirements, which needs to be done in

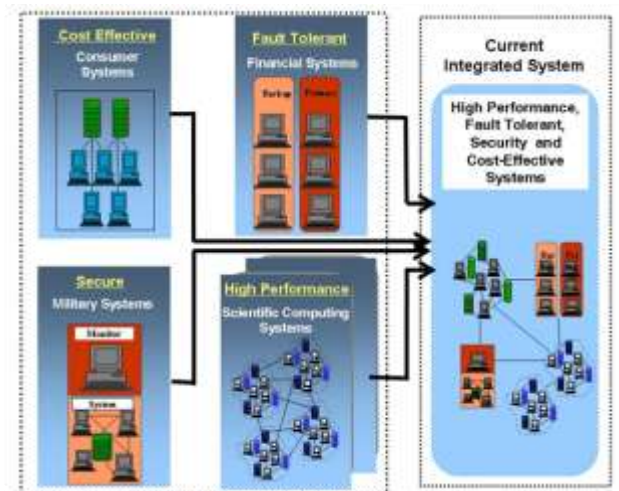


Figure 1: Integration of isolated solutions is inefficient, costly, insecure and unmanageable.

integrated and timely manners. The complexity, heterogeneity and dynamism of networked systems and their applications have resulted into systems whose control and timely management exceeds human ability.

Autonomic Components and Systems

This has led researchers to consider alternative design paradigms and management techniques that are based on strategies used by biological systems to deal with complexity, dynamism, heterogeneity and uncertainty – a vision that has been referred to as autonomic computing. Autonomic computing is inspired by the human autonomic nervous system and aims at realizing computing systems and applications that are capable of managing themselves with minimum human intervention. There have been several efforts to characterize the main features that make a computing system or an application autonomic. However, most of these techniques agree that an autonomic system must at least support the following four features:

1. **Self-Protecting:** Be able to detect attacks and protect its resources from both internal and external attacks.
2. **Self-Optimizing:** Be able to detect sub-optimal behaviors and intelligently perform self-optimization functions.
3. **Self-Healing:** Be able to detect hardware and/or software failures and should have the ability to reconfigure itself to continue its operations in spite of failures.
4. **Self-Configuring:** Be able to dynamically change the configuration of its resources in order to maintain overall system and application requirements.

Large scale autonomic computing systems can be dynamically composed from smaller Autonomic Components (AComs) where each component supports in a seamless manner any combination of the four properties mentioned above. That means, each ACom can be dynamically and automatically configured, seamlessly tolerate any component failure, automatically

detect component attacks and protect against them, and automatically change its configuration parameters to improve performance once it deteriorates beyond certain performance threshold. Once such autonomic components become available, we can dynamically build autonomic computing systems (see Figure 2) to meet any static requirements and runtime changes.

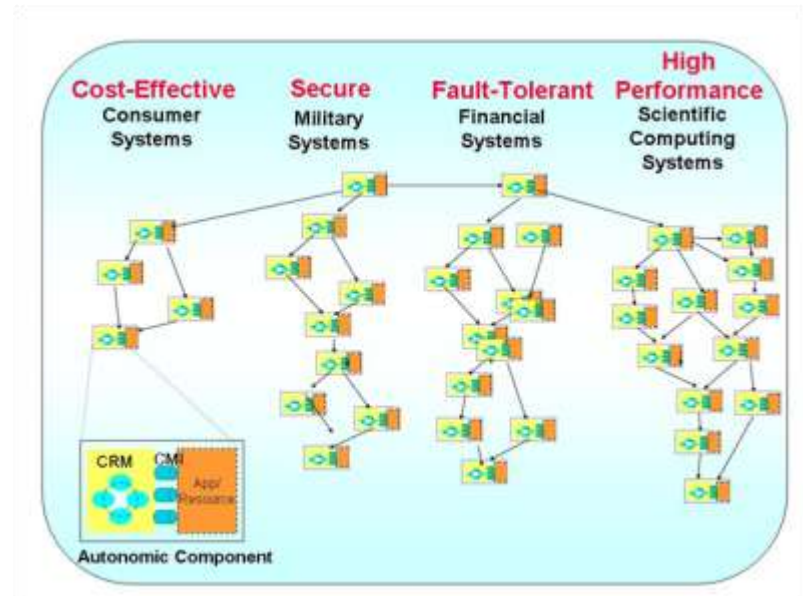


Figure 2: Holistic Approach: Autonomic Computing System