**Architecture Based Self-Adaptation**

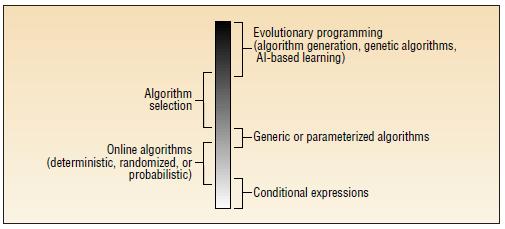
1. **INTRODUCTION:**

**1.1 Self-Adaptive software?**

Self-adaptive software modifies its own behavior in response to changes in its operating environment. By operating environment, means anything observable by the software system, such as end-user input, external hardware devices, and sensors, or program instrumentation.

**1.2 Why Architecture based Self-Adaptation?**

Initially all the software systems are often designed as open-loop systems: once a system is designed for certain function and deployed, its extra-functional quality attributes typically remain unchanged. If something goes wrong human has to intervene and also the repairs are of high costs. So the researchers tried to develop new methods to address these problems and hence several forms of self-adaptation such as exceptions in programming language features and fault tolerant protocols in algorithms have been discussed. But these are often highly specific to the application and tightly bound to the code. As a result, self-adaptation is costly to build, difficult to modify and usually provides only localized treatment of system faults. This can be seen even in the figure 1. It illustrates the software from low level to high level of abstraction. When a self-adaptation is performed in the low level abstraction the changes made are more localized. But if the adaptation is done in the architecture level it has effect on the entire system rather at a particular level. And yes, even developing self-adaptation at such high level abstraction costs more but it has lot more benefits.

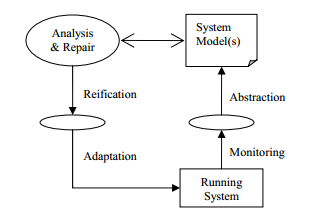


**Figure 1. A spectrum of self-adaptability. Generally, approaches near the bottom select among predetermined alternatives, support localized change, and lack separation of concerns. Approaches near the top support unprecedented changes and provide a clearer separation of Software-adaptation concerns**.

**1.3 How Architecture based Self-Adaptation is evolved?**

As, from the above we now know why we need architecture based self-adaptation rather than the other approaches. In order to implement architecture based self-adaptation an alternative approach for self –adaptation is proposed. In this approach adaptation is achieved using external software mechanisms to maintain closed loop control over the target system which in-turn allow the system to adapt dynamically and also reduced human intervention. With the adaptation mechanism as a separate entity, we can more easily modify and also we can extend it using customizable mechanisms which facilitates reuse across the systems there by reducing cost of developing new and separate self-adaptive systems for every software system.

The main principle in using external model is to determine the appropriate kind of model to use for software systems. Because, the model should not only be abstract enough to allow us to detect problems in target systems but also should provide some support to determine remedial actions to take to fix the problems. Hence we choose an architecture-based self-adaptation approach because it offers many beneﬁts as the architecture of a software system is an abstract representation of the system as a composition of computational elements and their interconnections. Most signiﬁcantly, an abstract architectural model can provide a global perspective of the system and expose important system-level properties and integrity constraints. A rich body of work on architecture trade-off analysis techniques used at system design time also facilitates runtime self-adaptation.



**Figure 2: Model based Adaptation**

The architecture based-self adaptation is illustrated in figure 2. The system is monitored to observe its run time behavior and monitored values are abstracted and related to architectural properties of the architectural model. Then the architectural model is analyzed whether is system running within the desired range or not, if not then violations are handled by using the repair system and new model is adapted, which are later propagated to the running system.

* 1. **Challenges in architecture based self-adaptation?**

Each and every phase of self-adaptation has many open issues like the artificial intelligence used in self-adaptation can be completely trusted what if suppose it takes a decision which can help in jeopardizing the entire software system. While monitoring what are the components to be monitored in the system? How to find out the problem that occurred inside the system or source of the problem? Does adaptation works only when an error occurs or it can improve performance of the system also? What are the steps to be taken while making changes to the existing system like recovery of the old running system?

In addition to these there are problems like cost of making self-adaptation systems? How to build system using reusable components? What kind of language, architecture style needs to be used while developing self-adaptation? Does the adaptation supports mobile environment? There are many more challenges still the researchers have been trying to solve.

1. **Evaluation criteria:**

**2.1**

1. **Different Approaches:**
2. **Conclusion**