An Architectural Approach to Fault Treatment in Critical Infrastructures

José Luiz Fiadeiro

ATX Software S.A. and LabMOL–University of Lisbon
Alameda António Sérgio 7 – 1 C, 2795-023 Linda-a-Velha, Portugal
jose@fiadeiro.org

1. On the challenges raised by critical infrastructures

Critical infrastructures, as information-intensive systems that support vital functions of our modern society (telecommunications, financial services, transports, energy supplies, etc). are becoming particularly vulnerable to failure. They are often built over unreliable networks of heterogeneous, fragile platforms. They perform critical missions that make them vulnerable to attacks. These are also systems that, individually, are becoming ever more complex and, globally, ever more interdependent.

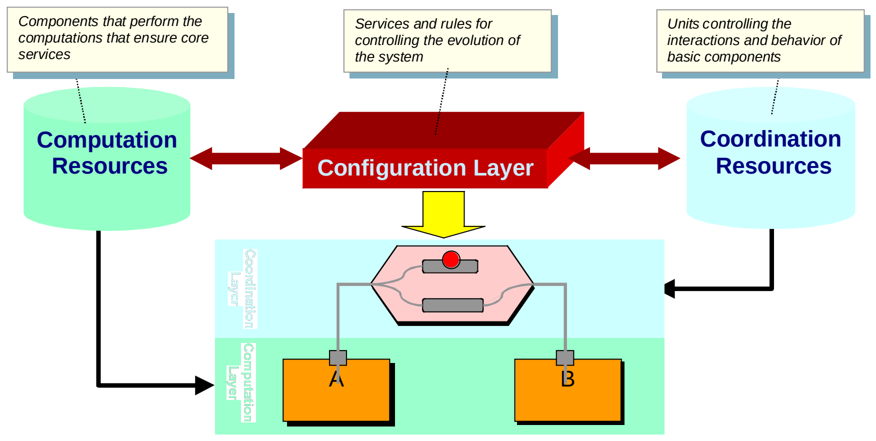
Fault treatment for this kind of systems requires new levels of flexibility and responsiveness. Due to the critical nature of the services that they support, such infrastructures cannot stop their operation when a fault arises. They can tolerate a certain degree of downgraded service for a certain period, but they need to ensure a minimal set of properties while the original services are not fully restored. For this purpose, they need to be able to react automatically to the occurrence of faults and reconfigure themselves to adapt to the new situation in which they need to operate, making use of the available resources.

2. Separating computation, coordination and configuration

The need to operate, in "real-time", with "surgical" precision for limiting the impact of the treatment, in contexts of increasing interdependency, requires a clear separation of concerns to be enforced in the way we model and manage such systems. We propose a three-layered architecture that separates what we consider to be the key concerns involved in this problem: computation, coordination and configuration.

The first separation, that between computation and coordination, is enforced by modelling explicitly the interactions that exist in the system as first-class entities – architectural connectors that we call coordination contracts. These connectors coordinate the way the components that reside in the computation layer interact. The latter correspond to "core" entities of the domain that provide basic services that, usually, cannot be "repaired" because they are performed by "black-boxes". By externalising all interactions as connectors, it becomes possible to circumscribe treatment of faults occurring at the level of a component to the connectors through which it interacts with the rest of the system. Basically, because it is often impossible to find a component that performs "equivalent" services, we see fault treatment as consisting of searching, within the available resources, components that offer alternative services, even if in a downgraded mode, and establishing the connectors that can adapt them to the expectations of the components with which they are required to interact.

This model supports the means for fault treatment to be performed through dynamic reconfiguration, in run-time, without interruption of service. For this process of reconfiguration to be able to be programmed, leading to self-adaptive and self-healing, we propose a third architectural layer consisting of entities that can react to events and act on the configuration, which are treated, again, as first-class citizens.


Components that perform the computations that ensure core services
Services and rules for controlling the evolution of the system
Units controlling the interactions and behavior of basic components
Computation Resources
Configuration Layer
Coordination Resources
Coordination Layer
Computation Layer
A
B

3. Agent and objects have been promising the same…

Interaction in most agent-based models is, like for object-oriented systems, based on *identities* in the sense that, through clientship, objects interact by invoking specific methods of specific objects (instances) to get something specific done. As a result, systems become too rigid to support the levels of agility that are required for fault treatment in critical infrastructures: any change on the collaborations that an object maintains with other objects needs to be performed at the level of the code that implements that object and, possibly, of the objects with which the new collaborations are established. On the contrary, being based on external connectors, our proposal models interactions in a service-oriented approach, i.e. interconnections are established on the basis of the description of what is required, thus decoupling the "what one wants to be done" from the "who does it". This is why faults can be treated in ways that are not intrusive on the rest of the system, thus ensuring increased levels of agility and responsiveness.

This does not mean that our architectural approach cannot be deployed over object-oriented platforms: it can, and we have even provided a proof of concept over Java. What is important is that the methodological approach and corresponding conceptual model is based on a service-oriented architecture as described. More information on this architectural approach can be found in www.atxsoftware.com.