SELMAS 2003 Panel: Dependability and QoS in Large-Scale MASs

Moderator:
Rogério de Lemos (University of Kent, UK)
Panellists:
Paolo Bresciani (ITC-irst, Trento, Italy)
Tom Maibaum (King's College London, UK)
Alexander Romanovsky (University of Newcastle upon Tyne, UK)
Arndt von Staa (PUC-Rio, Brazil)

The moderator, Rogério de Lemos, in order to promote the discussion, started the panel by defining key terms associated with its topic. Computing systems, in general, can be characterized by five fundamental properties: functionality, usability, performance, cost, and dependability. While, quality of services (QoS) refers to the non-functional properties of systems, dependability, in particular, refers to the ability of a computer system to deliver service that can justifiably be trusted. Since SELMAS 2003 had as a theme the interplay between the notions of agents and objects, the key question to be asked, from a software engineering perspective, was how an agent-based approach may improve the dependability and the QoS of software systems? Between the four basic means to achieve dependability, fault tolerance may be the most relevant one from the perspective of building agent based systems, since it deals with the continue provision of services despite the presence of faults. The basis for designing and building any fault tolerant systems is the specification of the failure assumptions of its components. However, if we consider that autonomy is a key property of agent systems, it is not clear how the failure assumptions can be associated with agents since their complete behaviour is not known beforehand. Even if we assume that autonomy can be restricted and worst class of failures can be associated with agents, still it is not clear whether a feasible implementation can be found without making strong assumptions about the communication between agents. The ability that agents have in adapting to environmental changes may also introduce some challenges, since the uncertainties associated with the outcome of their learning process might lead to unpredictability, which might impair the ability for evaluating the dependability attributes of multi-agent systems. Moreover, if autonomy and adaptability are essential features of these systems, what type of stability criteria should be associated with these systems for avoiding divergent behaviours among agents? Another property of an agent is its mental state, and if the notion of this 'mental state' is more anthropomorphic than the more conventional notion of a 'state', then can this mental state be observed by other system entities? If this state cannot be observed then it cannot be controlled, making it difficult the provision of fault tolerance. Based on the issues raised above, the question to be asked is what then can be the role of agents in large-scale systems? There are several ways in which agents could be employed: firstly, as basic building blocks, like functions, objects and components, secondly, as means for integrating legacy systems, and finally, in a lesser role, as a means for the provision of specific services, such as, an additional system layer to collect information, inform third parties, and control the allocation of resources. On the other hand, instead of restricting their applicability, another alternative would be to impose restrictions on the features offered by agents depending on the role taken by them when building systems. Likewise, when for the sake of predictability, inheritance is discarded when designing real-time systems using object-oriented technology, or a safe subset of Ada (SparkAda) is employed when building safety-critical systems. The moderator concluded by raising three questions to the panellist that should provide the basis for the panel: how do MASs features (e.g. autonomy, self-adaptation, intelligence, mobility, emergent behaviour...) make dependability and QoS easier or more difficult than in object-oriented systems, from the software engineering viewpoint? To what extent conventional and object-oriented dependability and QoS techniques can be used in the context of MASs engineering? Which are the challenges facing the promotion of dependability and QoS in MASs development?

Also within the realm of dependability, Paolo Bresciani discussed security in the context of agent-oriented software engineering. He started his talk by emphasizing that security analysis should be considered early in the lifecycle, instead of after the design of the system. In order to show the feasibility of this claim, Bresciani

presented an approach for dealing with security requirements using the Tropos agent-oriented software engineering methodology. The security process in Tropos consisted in analysing the security needs of the stakeholders and the system in terms of security constraints imposed on the stakeholders and the system, identifying secure entities that guarantee the satisfaction of the security constraints, and assigning capabilities to the system to help towards the satisfaction of the secure entities. Bresciani concluded that it was not easy to consider security issues in a simple way at the requirements level, hence the reason why they are usually considered at the late stages of development.

In examining the agent literature, Tom Maibaum has found two essential new concerns. One, a qualitative difference with software engineering concerns in the past, namely, agents combine a large number of underlying technologies, and are hence very complex artefacts. The second is a real difference in relation to past software engineering concerns, i.e., the use of introspection in the implementation of an agent. The use of anthropomorphisms in agent technology may be dangerous because it might lead to an illusion of understanding of concepts that does not actually exist. For example, Maibaum reiterated that he had never found a proper definition of the concept of autonomy, at least one that can be used for the design and analysis of agents. So, for him the most interesting research challenge in relation to agent technology is the management of the complexity of multiple technologies referred to above. The use of coordination technologies might be an aid to deal with this complexity.

Alexander Romanovsky in his talk considered the issue of fault tolerance and exception handling in large-scale MASs. He started by stressing that due to types of faults in MASs what is actually needed is software fault tolerance at the application level, rather than software based mechanisms for tolerating hardware faults (ACID transactions, replications, atomic broadcasts, etc.). Moreover, instead of recovery techniques based on rollback, alternative solutions based on forward error recovery should be sought. One of these solutions is exception handling, and the challenge is to develop novel exception handling techniques suitable for large-scale MASs. There are several advantages of using exception handling as a means to achieve fault tolerance: separation of code and flow in terms of normal and abnormal behaviour, and the recursive structuring by multiple level exception handling as a way to limit the scope of recovery. However, due to the characteristics of MASs new problems arise. Agents are autonomous, thus they cannot report exceptions to higher level. Agents are interactive, thus they need the concept of scope or exception context. Agents are mobile, thus requiring special exception handling techniques since agents can leave the location and move to another location, or the execution environment and the resources available can change on the fly. The communication between agents is asynchronous, thus requiring the decoupling of producers and consumers, and anonymous communication using event-based interactions. In order to deal with some of these limitations, several solutions are possible. Exception handling contexts should include all the agents involved in a particular exception or its respective handling, and this could be implemented by defining all the cooperating agents in a particular location or dynamically by mutual agreement. Agents should include in their implementations additional information about the sort of services they provide, and assumptions about their environment and other agents; this additional information could also be store in some sort of registry.

Arndt von Staa focused his talk on agents and dependability, and how trust can be obtained when using agents for building critical systems. In his initial statement he said that the forty plus years he has dealt with computers has taught him to be sceptic, and to support this statement he enumerated several cases in which either computers or software were held responsible for system failures. In order to motivate the usage of agents in critical applications, von Staa described a hypothetical scenario in which individual aeroplanes were agents in a MASs. The collection of aeroplanes forms a society of agents in which aeroplanes and obstacles enter and leave the space of interest of the society or of a specific aeroplane. In this society, each aeroplane would be able to sense the proximity of obstacles and other aeroplanes, to determine the possibility of collision in the next n time units, and to take evasive action to avoid collision. The question raised by von Staa was whether anyone in audience would allow themselves to fly in such system. In the second part of his talk, von Staa's raised several issues related with the validation of MASs. For example, can we prove the correctness of MASs? More specifically, do we really know how to specify them or are able to obtain adequate models? In terms of the development process of MASs, he questioned whether we correctly understand all interface problems in agents? Whether we can trust separately (incrementally) developed of agents and their integration? His final query was how one can test MASs? He concluded his talk by stating

that we must still be very humble when proposing agent-based solutions, and that high risk applications should not be based on MASs, at least for the time being.