The Role of Agreements in IT Management Software

Carlos Molina-Jimenez[1], Jim Pruyne[2], and Aad van Moorsel[1]

[1] University of Newcastle Upon Tyne, School of Computing Science, Newcastle upon Tyne, NE1 7RU, United Kingdom
{carlos.molina, aad.vanmoorsel}@ncl.ac.uk
[2] Hewlett-Packard Laboratories, 1501 Page Mill Rd., Palo Alto, CA 94304
jim.pruyne@hp.com

Abstract. Various forms of agreements naturally arise in the service provider model as well as in multi-party computing models such as business-to-business, utility and grid computing. The role of these agreements is twofold: they stipulate obligations and expectations of the involved parties, and they represent the goals to be met by the infrastructure. As a consequence of this latter point, in order to automate run-time adaptation and management of systems and services, agreements should be encoded and integrated in management software platforms. In this paper, we review the state of the art in software support for various forms of agreements, for all stages of their life-cycle. We also review emerging platforms and technologies in standard bodies, industries and academia.

1 Introduction

We will argue and illustrate in this paper that distributed computing infrastructures must incorporate *agreements* as first-class software building blocks to support automated adaptation and management in the presence of multiple (possibly competing) interests. These agreements represent expectations and obligations of various partners about the functionality and performance of systems and services. Additionally, these agreements are means to set the objectives for automated decision-making in system adaptation and management. It can therefore be useful for an IT operator to formulate objectives in the form of agreements even if no other parties are exposed to this information.

Modern-day and emerging computing infrastructures are increasingly flexible in their support of computational and business models, as witnessed by the developments of adaptive and on-demand computing solutions advocated by HP, IBM, Oracle, SUN and others. These solutions typically envision a service provider model for various aspects of computing, such as CPU use, network use, application hosting, etc. As software platform, such solutions are often tied to the grid [20], which supports resource sharing across multiple parties using open software. This software virtualises resources and applications, thus shielding the customer from the complexities of the underlying infrastructure and providing the operator with tools to adapt the system gracefully at run-time.

R. de Lemos et al. (Eds.): Architecting Dependable Systems III, LNCS 3549, pp. 36–58, 2005.

To enable the mentioned multi-party computing models the supporting software infrastructure should make use and keep track of the agreements established between parties. Therefore, it should embody these agreements in software. Such *run-time agreements* can come in various shapes or forms (at times we will use the adjective 'run-time' to stress that we discuss software embodiments of agreements, used at run-time to manage the execution of services). For instance, a run-time agreement can be defined by an operator to represent aspects of a hard-copy contract signed between provider and customers. Alternatively, the run-time agreement represents agreed-upon service levels automatically negotiated by software agents of the provider and customer. Irrespective, the information in the agreement can be used throughout the platform as needed, e.g., to adapt the system to meet service levels while optimising profits. Agreements thus naturally fit the service provider model, but also provide the necessary information to allow for automated decision-making by management software and self-managing components and services.

Since this article has been prepared for the series of books on architectures for dependable systems, it is opportune to address the relation of agreements with both dependability and architectures. The notion of architecture used in this paper relates to the structure of software platforms (middleware). These architectures will be heavily influenced by the emergence of the service provider computing model (including utility or on-demand computing), and by an increased pressure to automate system operation and hence save operational cost. We foresee a prominent role for agreements, which will be integrated in such architectures and will be represented as objects, services or other software components. Once these are in place, they define the objectives to be used in the algorithms that adapt systems and services.

With respect to dependability, run-time agreements play a double role, as indicated above. On the one hand, agreements must be available in system operations software to determine how to adapt the system, also in response to failures. This entails further automation of the processes traditionally involved in fault management. On the other hand, agreements are a way of providing trust in the system, by allowing customers to express their interests, and by providing them with information about whether the agreements are met (possibly through a trusted third party).

There are many open issues in the technologies required to support run-time agreements. The emphasis in this paper is on a survey of existing and ongoing work related to software architectures, with pointers to remaining research issues. The survey is extensive, but arguably not exhaustive because of the vastness of the area we want to cover. In Section 3 we discuss technologies in (1) specification, (2) provision, (3) monitoring, (4) adaptation and (5) resolution, roughly following the life cycle of typical agreements. Table 1 summarizes our findings. In Section 4 we then discuss representative solutions proposed by standards bodies (WS-Agreement in the Global Grid Forum), industries (Hewlett Packard and IBM enterprise IT) and academia (TAPAS, an EU research project). To set the stage, we first discuss terminology used in this paper and in the literature.