An Overview of the TAPAS Architecture

Santosh Shrivastava

School of Computing Science

University of Newcastle upon Tyne

1. Introduction

This report has been produced in response to the First Year Review requirement that "D5 should be extended by an attachment which describes the overall architecture of the TAPAS platform including the actual software tools and runtime support envisaged to exist at the end of the project".

In the TAPAS project, we are particularly interested in developing solutions to the problem faced by Application Service Providers (ASPs) when called upon to host distributed applications that make use of a wide variety of Internet services provided by different organisations. This naturally leads to the ASP acting as an intermediary for interactions for information sharing that cross organisational boundaries. As explained in the main report D5 [1], essentially this means that an ASP should be capable of hosting Virtual Enterprises[1] (VEs): meaning, it should be capable of providing facilities for forming and managing VEs. The main problem in VE management is how enterprises can regulate access to their resources by other enterprises in a way to ensure that their individual policies for information sharing are honoured. Regulating access to resources by other enterprises is made difficult, since each potentially accessible enterprise might not unguardedly trust the others. Enterprises within a VE will therefore require their interactions with one another other to be strictly controlled and policed. And in this context, there will be a clear need among all parties to embark upon their business relationships underpinned by guarded trust management procedures. How can this be achieved?

It is argued in D5 that to form and manage VEs, we need to emulate electronic equivalents of the contract based business management practices; this means: (i) relationships between organisations for information access and sharing will need to be regulated by *electronic contracts*, defined in terms a variety of *service level agreements* (SLAs), and then enforced and monitored; and (ii) organisations will need to establish appropriate *trust relationships* with each other and implement corresponding access control policies before permitting interactions to take place. It is on this basis that we intend to develop TAPAS platform, tools and services.

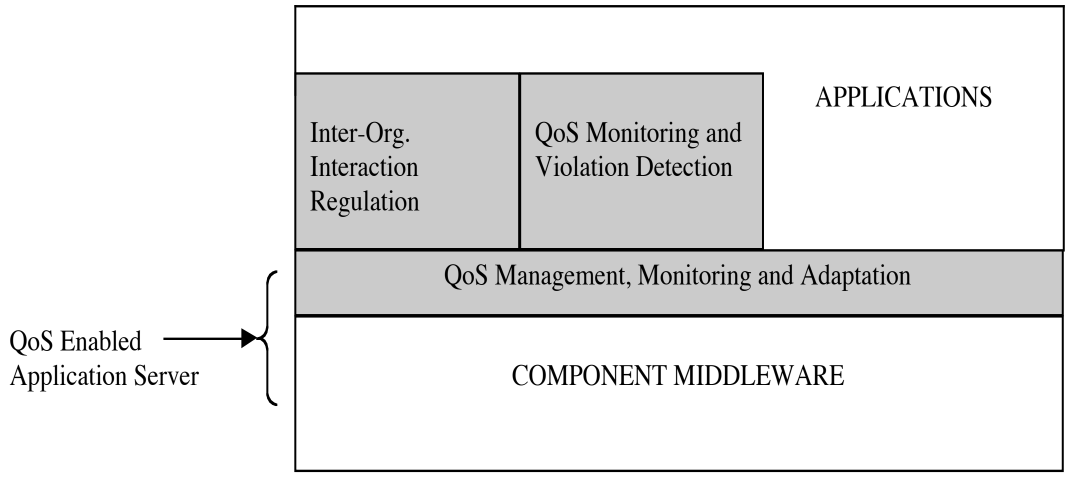
---

[1] A Virtual Enterprise comprises *n* independently existing and possibly mutually suspicious enterprises each of which wishes to establish a close business relationship for an agreed period of time without loosing its independence (autonomy).

2. TAPAS platform, tools and services

2.1. Architecture

The figure shows the main features of the TAPAS architecture. If we ignore the three shaded entities (these are TAPAS specific components), then we have a fairly 'standard' application hosting environment: an application server constructed using component middleware (e.g., J2EE). It is the inclusion of the shaded entities that makes all the difference.


APPLICATIONS
Inter-Org. Interaction Regulation
QoS Monitoring and Violation Detection
QoS Management, Monitoring and Adaptation
COMPONENT MIDDLEWARE
QoS Enabled Application Server

TAPAS Architecture

The QoS management, monitoring and adaptation layer is intended to make the underlying application server QoS enabled. It is responsible for reserving the underlying resources necessary to meet the QoS requirements of applications hosted by that application server, and monitoring the reserved resources, and possibly adapting resource usage (e.g., reserving some more) in case the QoS delivered by these resources deviates from that required by the applications. Specific architectural details are given in deliverable report D7 [2], and further discussed in section 2.2.

All cross-organisational interactions performed by applications are policed by the Inter-Organisation Interaction regulation subsystem. Deliverable D5 describes how relevant aspects of contracts can be converted into electronic contracts (x-contracts) and represented using state machines and role based access control (RBAC) mechanisms for run time monitoring and policing; it is further discussed in section 2.3.

It is necessary to ensure that a hosted application actually meets the QoS requirements (e.g., availability, performance) stated in the hosting contract SLAs. For this reason, we need an application level QoS monitoring service, which must also measure various application level QoS parameters, calculate QoS levels and report any violations. That is the function of the third subsystem shown in the figure. Design of such a subsystem is part of second year work.

We can see that QoS monitoring is occurring at two distinct levels: within an application server for controlling use of application server resources and at higher level for controlling application level QoS requirements. In TAPAS, QoS requirements will be specified using the SLAng language described in deliverable report D2 [3].

2.2. QoS Enabled Application Server

QoS control in application server will be implemented, as discussed in D7, by two principal middleware services, named *Configuration Service* and *Controller Service*, respectively, that can be used to extend an application server. The former service is responsible for discovering, negotiating, and reserving the resources necessary to meet the QoS requirements of a particular application component, hosted by that application server; the latter service is responsible for monitoring the reserved resources, and possibly adapting the component execution in case the QoS delivered by these resources deviates from that required by the component itself.

QoS requirements of hosted applications as stated in SLAs have been discussed in TAPAS deliverable report D1 [4]. These requirements will be specified using SLAng. Such a SLAng specification will be used for two purposes:

(i) *Design Time*: Given the QoS requirements, a designer needs to know what should be the configuration of the application server(s). TAPAS will define the semantics of SLAs using stochastic process algebras, and then use model checking capabilities developed for stochastic process algebras to support reasoning about the performance and scalability characteristics of components and their composition. This will support the application service provider in assessing the qualities of service perceived by end-users prior to developing and deploying an application service. A service composition and analysis tool will be developed (deliverable D4).

(ii) *Application Deployment Time*: SLAng specification will be used for initialising various QoS related configuration information required by the configuration service of the application server, through the deployment descriptors (e.g., for specifying component replication).

There is one more use of SLAng specification: for configuring application level QoS monitoring. This aspect is discussed in section 2.4.

Three services/subsystems will be developed, each addressing a specific QoS property (availability, timeliness and security respectively):

(i) *Component Replication*: how replication for availability can be supported by component containers so that components that are transparently using persistence and transactions can also be made highly available. Initial ideas concerning this work have been presented as an appendix to the deliverable report D7.

(ii) *QoS enabled group communication*: how reliable multicast with timeliness delivery guarantees can be supported. TAPAS project is particularly interested in hosting applications that perform multiparty communication (e.g., an auction application). This work will be reported in the second year deliverable report D8.

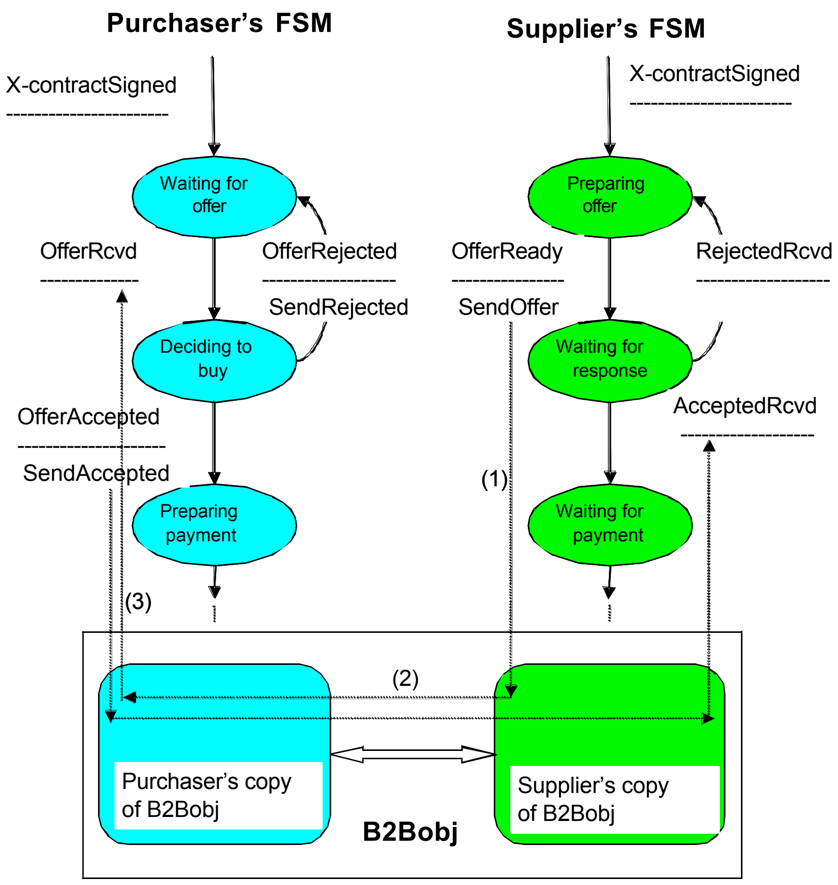
(iii) *Trusted coordination*: what generic support is required to support inter-organisational interactions between organisations that need not trust each other. A specific mechanism for information sharing, called B2Bobject has been developed, and described in the appendix of deliverable report D5. It is assumed that each organisation has a local set of policies for information sharing that is consistent with the overall information sharing agreement between the organisations (this agreement can be viewed as a business contract between organisations). The safety property of our system ensures that local policies of an organisation are not compromised despite failures and/or misbehaviour by other parties; whilst the liveness property ensures that if all the parties are correct (not misbehaving), then agreed interactions would take place despite a bounded number of temporary network and computer related failures. The system automatically maintains non-repudiable evidence of all inter-organisation interactions. Work on integrating B2Bobjects into component middleware has begun. Additional coordination mechanisms will be investigated, if necessary. This work will be reported in the second year deliverable report D9.

2.3. Inter-Organisation Interaction Regulation

In our business model enterprises that engage in contractual relationships are autonomous and wish to remain autonomous after signing a contract. We assume that interacting entities cannot simply rely on the trust they have in one another. To be of practical use, such trust relationships must be managed and observed. Because of this, interacting parties must resort to mechanisms that guarantee the rights and obligations that each interacting entity promises to honour. In the worst case, violations of agreed interactions are detected and notified to all interested parties (for this, an audit trail of all interactions will need to be maintained). Each enterprise expects access to other's services. An operation on a service is allowed only if it is permitted by the rules of the contract and then only if it is invoked by a legitimate role player of a participating enterprise. Thus, a contract is a mechanism that is conceptually located in the middle of the interacting enterprises to intercept all the contractual operations that the parties try to perform. Intercepted operations are accepted or rejected in accordance with the contract clauses and role players' authentication.

Our approach is to represent service interactions as finite state machines and make use of role based access control (RBAC) mechanisms for authenticated access. Use of finite state machines for representing service interactions has been proposed for Web services (Web service conversation language, WSCL [5]). We note that inter-organisation business interactions, PIPs (partner interaction processes) as specified in Rosettanet industrial consortium [6] can also be represented as finite state machines. In the deliverable report D5, we describe how contract clauses can be converted into finite state machines.

D5 also describes how B2Bobject coordination mechanism can be used for supporting a distributed version of regulated contractual interactions encoded as finite state machines. The main diagram from that report is reproduced below (it depicts purchaser-supplier interactions).



Regulating Inter-organisation interactions

2.4. QoS Monitoring and Violation Detection

Several of the rights and obligations in SLAs in a contract refer to the quality of service (e.g., service availability, performance guarantees). We assume that interacting entities cannot simply rely on the trust they have in one another and assume that QoS levels are being honoured. To be of practical use, a service provider must be able to demonstrate that the offered service meets the QoS levels promised to service users; hence the need for the QoS monitoring and violation detection subsystem. With the agreement of the parties involved, the hosted applications and services need to be instrumented with appropriate sensors for measuring QoS parameters. The parties involved must also agree on the QoS evaluation techniques (calculation procedures to be used). These details need to be encoded in some way in the SLAng specification language. Then the SLAng specification can be used for configuring the monitoring and violation detection service. Design of such a subsystem, including extending SLAng as indicated here is part of second year work.

3. Concluding Remarks

We have presented an overview of the TAPAS architecture. Referring to the figure in section 2.1, we have identified three subsystems that the project will develop. For each of these subsystem, we have identified tools and techniques the project will develop. We have also described how the technical details presented in the three deliverable reports, D2, D5 and D7 relate with each other.

References

[1] TAPAS Deliverable report D5, “TAPAS Architecture: Concepts and Protocols”, March 2003.

[2] TAPAS Deliverable report D7, “TAPAS Architecture: QoS Enabled Application Servers”, March 2003.

[3] TAPAS Deliverable report D2, “Specification Language for Service Level Agreements”, March 2003.

[4] TAPAS Deliverable report D1, “Application Hosting and Networking Requirements”, September 2002.

[5] Web Service Conversation Language (WSCL) 1.0 (http://www.w3.org/TR/wscl10/)

[6] Rosettanet implementation framework: core specification, V2, Jan 2000. http://rosettanet.org