TAPAS

IST-2001-34069

Trusted and QoS-Aware Provision of Application Services

TAPAS Final Report

Report Version: Deliverable D20

Report Delivery Date: 31 March 2005

Classification: Public Circulation

Contract Start Date: 1 April 2002 **Duration:** 36m

Project Co-ordinator: Newcastle University

Partners: Adesso, Dortmund – Germany; University College London – UK; University of Bologna – Italy; University of Cambridge – UK

Project funded by the European Community under the "Information Society Technology" Programme (1998-2002)

TAPAS Final Report

Santosh Shrivastava
School of Computing Science, University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, England

Table of Contents

Summary

The main objective of the TAPAS project was to develop novel methods, tools, algorithms and protocols that support the construction and provisioning of Internet application services. The project planned to achieve this objective by developing QoS enabled middleware services capable of meeting Service Level Agreements (SLAs) between application services.

The project identified the following three key requirements for application service provisioning.

1. Enhancing the application hosting middleware platform to be QoS aware. This way, hosting platform will be better equipped to meet the requirements of the hosting applications. In the absence of such a feature, the only alternative available to an ASP is *over provisioning*, which is not particularly desirable.

2. Ability to ensure that all inter-organisation interactions are strictly according to the terms and conditions contracts in force. 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.

3. Ability to demonstrate that hosted applications are meeting the various QoS requirements of SLAs.

These three requirements underpin the design of the TAPAS architecture that contains three new systems (layers).

The QoS management, monitoring and adaptation layer is intended to make the underlying application server QoS enabled (requirement 1). 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.

All cross-organisational interactions performed by applications are policed by the Inter-Organisation Interaction regulation subsystem (requirement 2). Techniques were developed enable relevant aspects of terms and condition 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. Techniques were developed to enhanced middleware to incorporate non-repudiable service interactions providing audit trails of service interactions.

It is necessary to be able to demonstrate that a hosted application actually meets the QoS requirements (e.g., availability, performance) stated in the hosting contract SLAs (requirement 3). For this reason, we developed an application level QoS monitoring service, which must also measure various application level QoS parameters, calculate QoS levels and report any violations. In TAPAS, QoS requirements in SLAs are specified using the SLAng language.

An important feature of TAPAS architecture is that the three subsystems can be deployed independent of each other. For example, an ASP might decide to use a ‘standard’ application server, without the need for QoS management features, because in a given scenario, over provisioning might be acceptable. The ASP still might need one or both of inter-organisation interaction regulation and QoS monitoring and violation detection subsystems. Another important feature of the TAPAS architecture is that the inter-organisation interaction regulation subsystem, as well as the QoS monitoring and violation detection subsystem could be provided by the ASP or one or more trusted third parties, thereby providing extreme flexibility in deployment.

1. Introduction

It is well known that organisations are increasingly focusing on their core businesses and streamlining their operations by 'outsourcing' non-core businesses to external organisations. In particular, many organisations find it cost effective to outsource their IT applications to *Application Service Providers* (ASPs). An ASP typically uses middleware and component technologies for deploying, hosting and managing applications of an organization from a centrally managed facility. However, as organisations become global and distributed, such centrally managed hosting solutions will need to be replaced by multi-site, distributed hosting solutions.

The TAPAS project was 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 first year deliverable report D5 [1], essentially this means that an ASP should be capable of hosting Virtual Organisations (VOs), meaning, it should be capable of providing facilities for forming and managing VOs. We define a Virtual Organisation (VO) as a strategic alliance among a group of cooperating organisations that share services electronically – say using Web/Internet technology – for the accomplishment of a set of mutually beneficial business goals; these arrangements being made such that each organisation continues to maintain its own autonomy, except for the mutually agreed undertakings of the alliance.

A central requirement of VO operational management is to enable organisations to regulate access to their service resources in a manner, which honours their individual resource sharing policies both securely and with integrity. Regulating such access is made difficult since each potentially accessible organisation might not unguardedly trust the others. Accordingly, all organisations within a VO will require their interactions to be strictly controlled and policed. There will therefore be a need for all business process relationships to be underpinned by guarded trust management procedures.

In the TAPAS project, we have taken the view that to form and automatically manage partnerships within a VO underpinned by guarded trust management procedures, it will be necessary to have electronic representations of contracts that can be used to mediate 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.

With the above observations in mind, the overall objective of the TAPAS project was to develop novel methods, tools, algorithms and protocols that support the construction and provisioning of Internet application services.

2. Motivating Example

The auction demonstrator application developed within the project [2] is a good example, illustrating various contracts involved. The application is a *sealed reverse auction* that is used by a car manufacturer (Toyota, Ford, etc.) for buying car parts (e.g. tyres, radiators, mirrors, etc.) from car part suppliers. Figure 1 shows the parties that participate in our demonstration scenario. We will discuss the roles played by each party first and latter on we will discuss their contractual business relationships, which are represented by double-headed arrows in the figure.

- **The auctioneer** is the representative of a car manufacturing company that at a given time runs one or several instances of the auction with the purpose of buying car parts from car part suppliers; for example, he might run an instance of the auction for buying seats and another one for buying batteries. We assume that he does not want to be disturbed with computer-related issues, thus, he relies on somebody else to provide the infrastructure to run the auction; the business related activities which the auctioneer performs include selecting and sending invitations to potential bidders, opening and closing of bid rounds, declaring winners and so on.

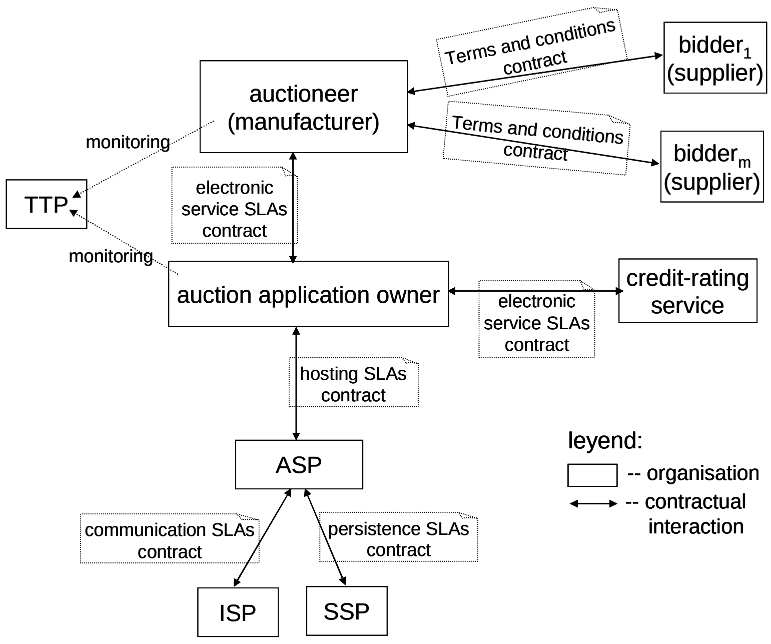

Terms and conditions contract
bidder$_1$ (supplier)
auctioneer (manufacturer)
Terms and conditions contract
bidder$_m$ (supplier)
monitoring
electronic service SLAs contract
TTP
monitoring
auction application owner
electronic service SLAs contract
credit-rating service
hosting SLAs contract
leyend:
-- organisation
-- contractual interaction
ASP
communication SLAs contract
persistence SLAs contract
ISP
SSP

Fig 1. Auction scenario with its participants and their contractual relationships.

- **The bidders** are car part suppliers interested in selling their products to the auctioneer. In our scenario we consider that the number of bidders can be in the order of several hundreds.

- **The auction application owner** is the enterprise that offers the auctioneer the auction application ready to use; we can conceive it as an enterprise that owns a source code of the auction software and builds on-demand customized and ready to use copies of it to different auctioneers. The auction application owner is not involved in business related activities but in technical ones only; its responsibilities include running and tuning the auction software. We assume that the auction application owner does not have the ancillary resources to run his auction software, thus he relies on somebody else to host it.

- **The Application Service Provider (ASP)** is an enterprise that offers hosting services to the auction application owner. It provides all the necessary ancillary resources (CPU, databases, ISP, human, etc.) to host the auction software. The assumption is that at a given time the ASP might be hosting several instances of the reverse sealed auction that belong to the one or several auctioneers as well as other applications of arbitrary nature.

- **The Internet Service Provider (ISP)** is an enterprise that offers Internet connectivity to the APS so that the auction can be reached by other interested parties of the auction scenario.

- **The Storage Service Provider (SSP)** is an enterprise that offers disk space to the ASP.

- **The business service** is an enterprise that offers credit-rating services, or any other ancillary service, such as billing, to the auction application owner.

- **The Trusted Third Party (TTP)** is an enterprise with enough credentials to act as a trusted third party. It measures the performance of a party, assesses it and determines whether the party is honouring its contractual obligations with respect to another party. For the sake of simplicity, Fig. 1 shows the TTP monitoring the contractual obligation only between the auctioneer and the auction application owner; however, a TTP can and should be deployed between any pair of business partners such as the auctioneer and bidder$_1$, and the auction application owner and the ASP.

It is time now to discuss the meaning of the double-headed arrows that join the participants of our demonstration scenario shown in Fig. 1. A crucial assumption in our scenario is that the participants are autonomous and independent organizations that besides being mutually suspicious still want to conduct business together; because of the existence of this degree of mutual mistrust each pair of business partners needs to rely, as in conventional business, on legal business contracts to regulate their business interactions.

If in conventional business there are contracts for different commercial agreements (contracts for the rent of a house, contracts for loan of machinery, etc.) in our demonstration scenario, we identify five different types of contracts, namely, terms and conditions contracts, electronic service level agreement (SLA) contracts, hosting SLA contracts, communication SLA contracts and persistence SLA contracts. From a structural view, the five contract types are rather similar: all of them contain a header (signatories' names, addresses, signatures, start and end date, etc.) and clauses that stipulate the rights and obligations of each signatory party, however, from the point of view of the content of their clauses they are different:

- **Terms and conditions contracts** describe the relationship between the application owner and the bidders, i.e. the suppliers, specifying business action interactions. They specify 'business conversations' constrained by permissions, obligations, prohibitions, actors (agents), time constraints, and message type checking. We define a *conversation* as a small business activity executed between two or more business partners to perform a well defined task, such as submit a bid, issue a purchase order, process payment, refund money, cancel purchase order, etc.

- **Electronic service SLAs contracts** stipulate the QoS expected from the interaction between the auctioneer/auction application owner pair and the auction application owner/credit rating service pair. For the first pair, the electronic service SLA will contain clauses such as “the auction application owner shall guarantee that even during peak periods the invocation of the place_bid operation is successfully completed within two seconds when there are less than 100 users logged in”; for the second pair the electronic service SLA will contain clauses such as “the auction application owner shall never place more that 50 request per second”.

- **Hosting SLAs contracts** are used to specify the QoS between the ASP and the application owner. They contain the objectives of the electronic service SLA because the application owner will be eager to delegate the objectives to other providers. Due to the more technically oriented relationship between application owner and ASP, the hosting SLA contains as well technical objectives such as memory space and utilisation regulations for technical services such as user management, maintenance windows etc.

- **Communication SLAs contracts** specify QoS objectives for the relationship between ASP and ISP.

- **Persistence SLAs contracts** are used to specify QoS objectives for data storage service offered by an SSP.

The distinction of different types of contracts is relevant because we have learnt that the concepts and technology needed to represent, monitor and enforce a contract varies depending of the contract type.

3. Description of consortium and roles of research teams

Work within the project was structured into four technical workpackages (WPs).

- WP1: Application Service Requirements and Specification.
- WP2: Design of QoS-aware Infrastructure for Application Hosting
- WP3: Implementation of QoS-aware Core Services.
- WP4: Case Studies and Evaluation.

Basically, within WP1 we worked towards acquiring an understanding of the requirements and then developed SLA specification and its QoS analysis tools and techniques; WP2 was devoted to the development of trusted and QoS-aware middleware architecture based on the requirements generated from WP1. WP3 was about implementation of the architecture developed in WP2, and within WP4 we performed assessment of the architecture and its implementation through demonstrator application building exercises.

The TAPAS consortium brought together significant expertise from the application hosting, distributed computing and middleware, fault tolerance, software engineering, computer security and computer networking communities. Below we describe their principal responsibilities.

The School of Computing Science at the University of Newcastle: The coordinating contractor for TAPAS, the Distributed Systems group conducts research on concepts, tools and techniques for constructing distributed fault-tolerant systems that make use of standard, commodity hardware and software components. Current work is focused on dependable workflow management for cross-organisation workflows, information sharing in virtual enterprise and wide area group communication systems. The group has built a number of major distributed software systems as CORBA middleware services. Newcastle led WP2 and played active role in the development of the TAPAS architecture, tasks on QoS enabled group communication, trust management, trusted coordination and auction application development.

Adesso: Adesso AG is a German mid-range company offering IT consulting, software development and application hosting. Main clients are insurance companies and banks, for which the company analyses, designs and implements enterprise relevant applications based on component technologies such as Enterprise JavaBeans. The variety of hosted applications include applications in fields of telecommunication (B2B) and Internet portals for online communities and banks. Hosting of applications in areas of insurance companies and banks in mainly performed by the clients themselves, though the requirements and concepts are defined by the consultants and developers of Adesso AG. Adesso led WP 4 and provided case studies, hosting facilities and undertook auction application development and evaluation work. Adesso also played an active role in WP1 by providing requirements of application hosting.

The Dept. of Computer Science, University of Bologna: The research group is currently investigating a number research issues in the design of QoS-preserving, distributed middleware platforms. Specifically, these issues include: i) strategies for providing World Wide Web service users with adequate QoS. This activity involves investigating the design of middleware services that can meet effectively application-level (i.e., end-to-end) QoS requirements of Internet-based, latency-sensitive multimedia applications; and investigation on the use of group communication mechanisms to support replication in database systems. Bologna led WP3 and contributed to all the activities concerned with QoS, in particular, load balancing and QoS aware application server.

The Dept. of Computer Science, University College London (UCL): The Software Systems Engineering Group is concerned with the development of large and complex software intensive systems. It focuses on: the real-world goals for, services provided by, and constraints on such systems; the precise specification of system structure and behaviour, and the implementation of these specifications. The three key technologies where the group contributes to the state-of-the-art are: databases, distributed objects (particularly middleware and mobile agent technologies), web infrastructure (particularly XML and related technologies). UCL led WP1 and contributed to the development of SLA specification and analysis tool, architecture development.

Computer Laboratory, University of Cambridge: The Systems Research Group at the Computer Laboratory, University of Cambridge, has been one of the premier research forces in communications, distributed systems and operating systems since the founding of the lab, the oldest computer science teaching department in the world. Past projects include Universe (which delivered one of the earliest high speed distributed systems), nemesis, a novel operating system with excellent multimedia scheduling properties, as well as Home Area Networks, Xenos (an accountable peer-to-peer distributed architecture). Networks and Operating Systems related work is focusing on Disk QoS Enforcing Quality of Service in Storage Systems , Efficient Network Routeing, Next Generation Inter-AS Routeing. Cambridge worked on QoS networking requirements (WP1), Network Control Architectures (WP2) and QoS enabled group communication (WP3).

4. Project Results

4.1. Objectives

According to the description of work, "the overall objective of the TAPAS project is to develop novel methods, tools, algorithms and protocols that support the construction and provisioning of Internet application services. The project will achieve the overall objective by developing QoS enabled middleware services capable of meeting Service Level Agreements (SLAs) between application services and will enhance component based middleware technologies such that components can be deployed and interact across organisational boundaries. The project will develop notations for expressing SLAs to enable specification of QoS, such as the availability as well as trust relationships. SLA trust specifications will be used for deriving service invocation primitives enriched with authentication, non-repudiation mechanisms, with or without the involvement of trusted third parties."

The key scientific results and achievements of the projects have been classified as follows:

- TAPAS Architecture
- SLA Specification and analysis
- Terms and conditions contract specification, monitoring and enforcement
- QoS monitoring and violation detection
- QoS aware middleware
- Integration and evaluation

4.1. TAPAS Architecture

We identified the following three key requirements for application service provisioning.

1. Enhancing the application hosting middleware platform to be QoS aware. This way, hosting platform will be better equipped to meet the requirements of the hosting applications. In the absence of such a feature, the only alternative available to an ASP is *over provisioning*, which is not particularly desirable.

2. Ability to ensure that all inter-organisation interactions are strictly according to the terms and conditions contracts in force. 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.

3. Ability to demonstrate that hosted applications are meeting the various QoS requirements of SLAs.

These three requirements underpin the design of the TAPAS architecture. Figure 2 shows its main features. If we ignore the three shaded/patterned entities (these are TAPAS specific components), then we have a fairly 'standard' application hosting environment: an application server constructed using component middleware (e.g., CORBA, J2EE). It is the inclusion of the shaded/patterned entities that makes all the difference.

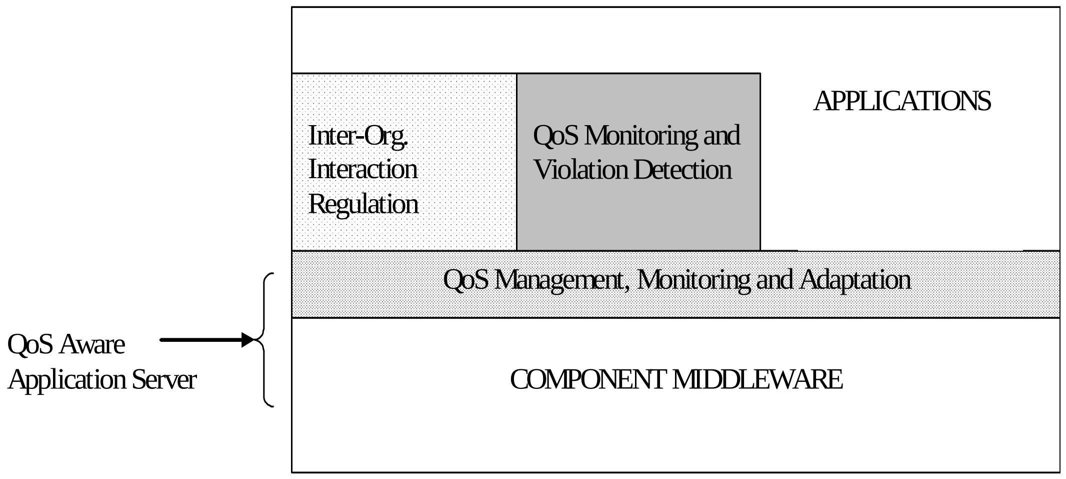

APPLICATIONS
Inter-Org. Interaction Regulation
QoS Monitoring and Violation Detection
QoS Management, Monitoring and Adaptation
COMPONENT MIDDLEWARE
QoS Aware Application Server

Fig. 2. TAPAS Architecture

The QoS management, monitoring and adaptation layer is intended to make the underlying application server QoS enabled (requirement 1). 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. Final year deliverable report D11 [3] describes in detail the design and implementation of QoS aware application server.

All cross-organisational interactions performed by applications are policed by the Inter-Organisation Interaction regulation subsystem (requirement 2). First year deliverable D5 [1] described how relevant aspects of terms and condition 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. Second year deliverable report D9 [4] described how component middleware can be enhanced to incorporate non-repudiable service interactions providing audit trails of service interactions. This subsystem could be provided by the ASP or one or more trusted third parties.

It is necessary to be able to demonstrate that a hosted application actually meets the QoS requirements (e.g., availability, performance) stated in the hosting contract SLAs (requirement 3). 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. This subsystem could be provided by the ASP or one or more trusted third parties. Second year deliverable report, D10 [5] described the design and implementation this subsystem. In TAPAS, QoS requirements in SLAs are specified using the SLAng language described in deliverable report D2 [6].

The overall features of the TAPAS architecture summarised above are described in detail in the deliverable report D6 [7].

An important feature of TAPAS architecture is that the three subsystems can be deployed independent of each other. For example, an ASP might decide to use a 'standard' application server, without the need for QoS management features, because in a given scenario, over provisioning might be acceptable. The ASP still might need one or both of inter-organisation interaction regulation and QoS monitoring and violation detection subsystems. Another important feature of the TAPAS architecture is that the inter-organisation interaction regulation subsystem, as well as the QoS monitoring and violation detection subsystem could be provided by the ASP or one or more trusted third parties, thereby providing extreme flexibility in deployment.

4.2. SLA Specification and analysis

4.2.1. Stakeholders

In order to understand Service Level Agreements between the participants in an ASP scenario we focused on the parties involved in ASP and the relationships between them. The ASP model described below (described in detail in deliverable D1 [8]) is an abstraction of typical industrial ASP situations. Figure below shows the stakeholders and their SLA-based relationships in an abstract ASP model.

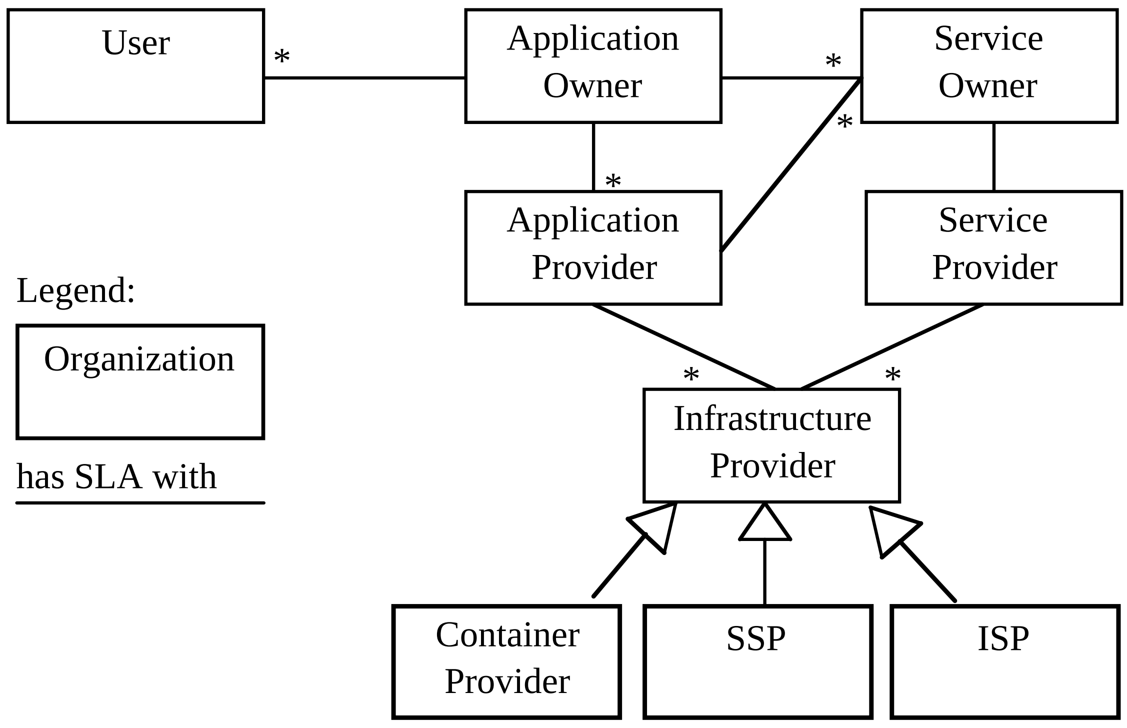


Fig. 3. Stakeholders

4.2.2. SLAng Specification language and analysis

SLAng is an XML language for defining service level agreements, the part of a contract between the client and provider of an Internet service that describes the quality attributes that the service is required to possess [6]. We define the semantics of SLAng precisely by modelling the syntax of the language in UML, then embedding the language model in an environmental model that describes the structure and behaviour of services. Fig. 4. illustrates the notations used in SLAng.

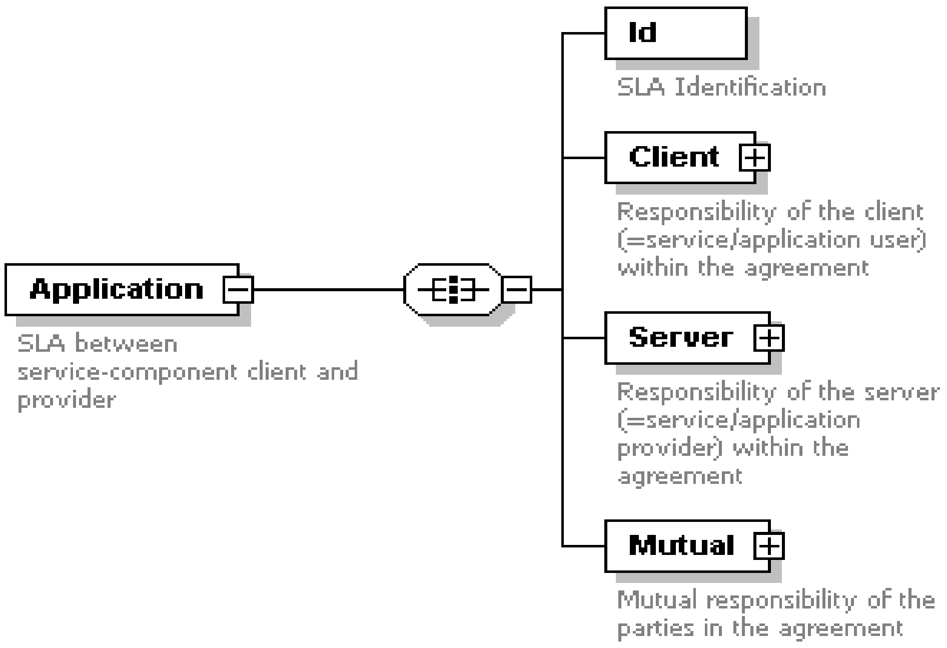

Id
SLA Identification
Client
Responsibility of the client (=service/application user) within the agreement
Application
SLA between service-component client and provider
Server
Responsibility of the server (=service/application provider) within the agreement
Mutual
Mutual responsibility of the parties in the agreement

Fig. 4. SLAng language

Work on the SLAng Service-Level Agreement (SLA) language as part of the TAPAS project has led both to a better understanding of the requirements for SLA languages and novel solutions to those requirements.

Our initial requirements analysis focussed on the problem of providing end-to-end Quality of Service (QoS) in federated distributed systems. We found that previous SLA languages had tended to target horizontal relationships between services; that is: relationships between services of a similar type, or at a uniform level of abstraction. We argued that because QoS factors such as performance and reliability were intrinsically dependent not only on the coordination of services but upon the infrastructure upon which they were deployed, that a more architectural approach was required. Accordingly we based the early design of SLAng on a classical layered model of application service provision. SLAng was able to represent agreements relating not only to electronic services, but also to component hosting, component replication, storage service provision and internet service provision, allowing service providers to use SLAng to make all of the agreements required to guarantee end-to-end QoS.

In the early design of SLAng we identified the need to protect the service provider from the actions of the service consumer. Because many QoS properties vary according to the load on a service, it is essential that the obligations on the client be described if the service provider is to be capable of fulfilling their own obligations.

We next considered the architecture of the language itself. SLAs are associated with financial obligations on the part of the signatories, and their main purpose is to mitigate the financial risk to each party of the other failing to fulfil their obligations. We therefore considered precision of expression to be a primary requirement for this language. Previous SLA languages provided either informal definitions of their terms, or made use of ontologies written in natural language to describe their semantics. We wished to develop a more rigorous semantic definition for the language. However, the relatively high level of abstraction at which the language must be described, and the dependence on the real-world details of services of particular types, did not lend itself naturally to a purely mathematical semantic description. We overcame this issue by adopting the model-denotational style of semantic definition employed by several specifications in the Model Driven Architecture technology space.

The principle underlying the model-denotation style of semantic definitions is that object-oriented models of the syntax of a language can be associated with an object-oriented model of the semantic domain of the language. The use of object-oriented models, whose semantics is partly defined by the interpretation of names in the models, allows fine control over the level of abstraction of the models, leading to very unambiguous descriptions of language semantics despite the use of real-world terms. We achieved a highly unambiguous description of the SLAng language semantics for electronic services by this means. The syntactic model of that part of SLAng is associated with a model of electronic services, including a model of events pertinent to the QoS of such services. Logical constraints within the model ensure that SLAs are only associated with services whose performance is consistent with the quantities specified in the SLA. The semantic model provides an explicit reference to the domain of application of the SLA, and the logical constraints provide an unambiguous description of the conditions implied by the SLA, leading to a high level of precision overall.

We considered the benefits that our semantic description might confer to a software engineering process. We looked at two scenarios of service composition: in the first, a client has a need for a service and a particular set of service requirements. They must choose a suitable service from a range of options, each offering non-negotiable ‘commodity’ SLAs. If the client expresses their requirements in SLA terms, then the SLAng semantics provide a strong standard of comparison between the requirements and the offered SLAs, which we call ‘compatibility’. An SLA A is compatible with another SLA B if the range of behaviours acceptable to A is always acceptable to B. Assessing compatibility is an open problem as in general the set of behaviours acceptable to an SLA may be infinite. However, the concept of compatibility is superior to previous standards of comparison proposed for SLAs, which rely on ordered metric spaces and may lead to unsafe choices of SLA.

The second scenario of service composition is to predict the QoS behaviour of a service based on the qualities offered by any services that it composes and its expected load. Since the emergent QoS is also dependent on the implementation of the service, this amounts to performing a traditional performance analysis, which is a broad area of ongoing research. We contributed to the large body of work on deriving performance models from UML designs by demonstrating that logical constraints could be used as contracts for such derivations. We also showed how SLA information could be incorporated into models by defining a profile for SLAng electronic service SLAs.

Finally we considered the implementation of monitoring solutions based on the SLAng language. The SLAng semantics unambiguously define the conditions that SLAng SLAs impose, but in practice these conditions will be monitored electronically to detect violations. Such a monitoring solution must respect the semantics of the language, and so neither ignore violations implied by the terms of the agreement, nor report incorrect violations. We observed that the highly formal and machine-readable nature of the models specifying the language rendered them appropriate input to MDA code generation tools. We therefore implemented a contract checker by employing a code generator to produce a data repository from the specification of the language. This repository can contain SLAs, and also data related to the execution of services, conforming to the types described in the semantic part of the language specification. The monitor is completed by a constraint evaluator that assesses whether the data in the repository is free from constraint violations. Since we employed constraints to describe the conditions required for a service to be performing adequately in terms of a related SLA, this process amounts to detecting SLA violations.

Automatically generating the SLA checker provides benefits in terms of both correctness and ease of implementation. Because the generation process and constraint violation checking processes are defined independent of the application domain (checking SLAs), we do not expect the generation process to introduce the same kind of errors of interpretation that might arise were a contract checker implemented by hand. Moreover, as the language definition changes, we can regenerate the checker at no additional cost. We have recently generalised our description of the approach to describe how it can be used to implement run-time requirements monitors for software systems, in which some of the requirements are specified at run-time.

We evaluated our contract checker implementation by integrating it into an industry standard application server and employing it to monitor the operation of the TAPAS auction-house example application. We discovered scalability issues in the implementation of the logical constraint checker, which we were able to report as research results. However, the experience validated the feasibility of our approach.

4.3. Terms and conditions contract specification, monitoring and enforcement

All cross-organisational interactions performed by applications are policed by the Inter-Organisation Interaction regulation subsystem. This subsystem could be provided by the ASP or one or more trusted third parties. 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 mechanisms for authenticated access. In the deliverable report D5, we described how contract clauses can be converted into finite state machines (FSMs).

Inter-Organisation Interaction regulation subsystem has two main layers (see figure 5). The contract monitoring and enforcement layer makes use of the services of the underlying layer that provides trusted coordination.

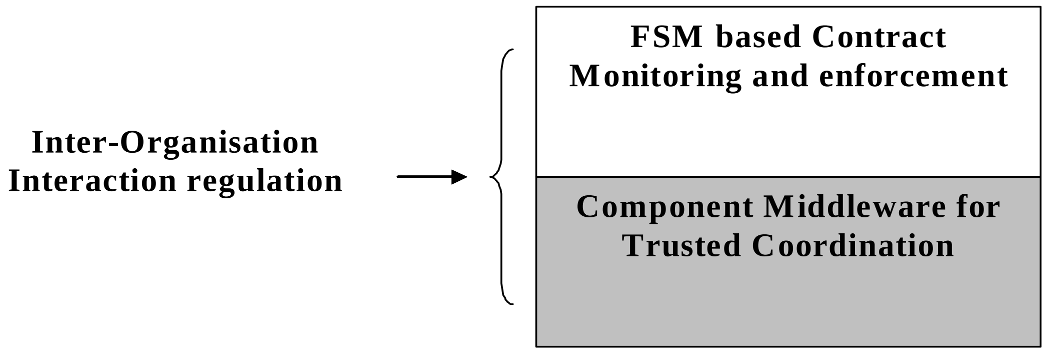

Inter-Organisation Interaction regulation
FSM based Contract Monitoring and enforcement
Component Middleware for Trusted Coordination

Fig.5. Inter-Organisation Interaction regulation

To regulate the interactions involved, a given action must be attributable to the party who performed the action and commitments made must be attributable to the committing party. For example, it should not be possible for a client to subsequently disavow the request and/or consumption of a service. Similarly, it should not be possible for the service provider to subsequently deny having delivered the service. If information is shared then the parties sharing the information should be able to validate a proposed update, the update should be attributable to its proposer and the validation decisions with respect to the update attributable to the other parties. That is, to regulate an interaction we require attribution, validation and audit of the actions of the parties involved. Non-repudiable attribution binds an action to the party performing the action. Validation determines the legality of an action with respect to interaction agreements. Audit ensures that evidence is available in case of dispute and to inform subsequent interactions.

4.3.1. Contract representation

It is necessary to have electronic representations of contracts that can be used to mediate the rights and obligations that each interacting entity promises to honour. This requirement implies that the original natural language contract that is drawn by lawyers and other non-technical people has to undergo a conversion process from its original format into a piece of executable code or *executable contract* (*x-contract* for short) that works as a mediator of the business conversations. This conversion process involves the creation, with the help of a formal notation, of one or more computational models of the contract with different levels of details.

We developed a general method of representing business interactions as FSMs using a widely used modelling language Promela [9] and showed how it can be used to represent permissions, obligations, prohibitions, actors (agents), time constraints, and message type checking; that is, all the basic parameters that compose most typical business contracts. Our motivations for using Promela here is that such a representation can be validated with the help of the accompanying Spin model-checker tool [10].

We developed two levels of contract representation. (i) *Implementation neutral*: free of technical details related to technology-related interactions; in other words, specifying only business action interactions (for example, issue a purchase order, send payment, etc.). Such a description can be model checked and used for improving the original natural language contract to be free from various forms of inconsistencies as discussed in [11]. (ii)

Implementation specific: a representation (also amenable to model checking) that is a refinement of the former to include technical details such as acknowledgements and synchronization messages that form an important part of any implementation; the details will vary depending upon the implementation techniques and standards that are selected (e.g., Rosettanet [12]). Such a representation can be used for implementing the actual x-contract for business conversation mediator.

4.3.2. Deployment models for contract enforcement

Conceptually speaking an x-contract is placed in between the two business partners so that it can observe their business interactions. Deployment can be either *centralized* or *distributed*. Further, the x-contract could be *reactive* or *proactive,* giving us four deployment models discussed below, where for illustration purposes we assume an interaction from buyer to seller:

(i) *Reactive Central*: The contract is deployed in a trusted third party (TTP), see Fig. 6(a). The job of the x-contract here is to intercept (1) and analyze (2) the messages exchanged between the two business partners; correct messages are forwarded (3) to their final destination whereas incorrect ones are dropped (3’).

(ii) *Proactive Central*: The contract is deployed in a TTP, see Fig. 6(b); the contract is proactive in that it coordinates the conversational interactions between organisations by invitation only. It sends (1) an invitation message to the business partner; the response is received (2) by the x-contract and analyzed (3); correct messages are forwarded (4) to the seller, whereas incorrect ones are dropped (4’).

(iii) *Reactive Distributed*: Distributed version of reactive central: the contract is split and deployed in two TTPs, see Fig. 6(c).

(iv) *Proactive Distributed*: Distributed version of proactive central.

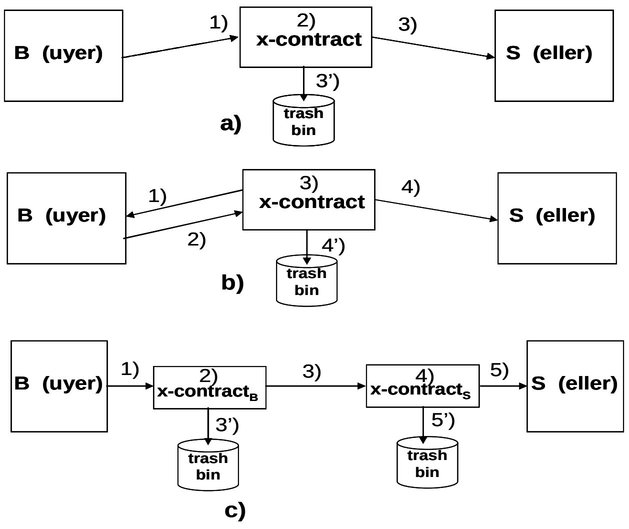


Fig.6. Deployment models (a) reactive central (b) proactive central and c) reactive distributed.

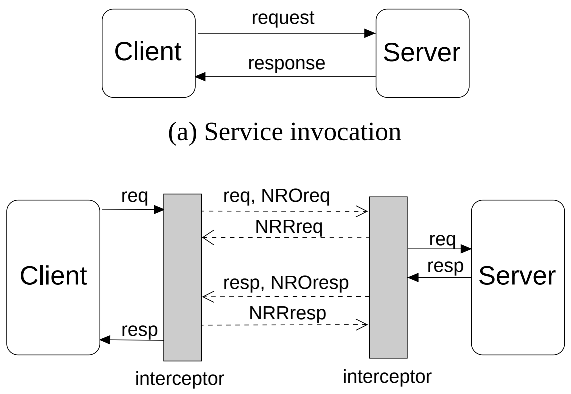
Which particular model is suitable in a given VO setting is a very interesting research problem, worthy of further investigation. We note that distributed deployments face the difficult challenge of keeping contract state information synchronised at both ends. For example, a valid message forwarded by the buyer's x-contract could be dropped at the seller's end because intervening communication delays render the message untimely (and therefore invalid) at the seller side. State synchronisation is necessary to ensure that both the parties either agree to treat the message as valid or invalid. Non-repudiable information sharing protocol designed and implemented by us and discussed in deliverables D5 and D9 provides such synchronisation (see also the next section).

4.3.2. Component middleware for trusted coordination

We developed two novel building blocks for regulated interaction between organisations: non-repudiable service invocation (NR-Invocation) and non-repudiable information sharing (NR-Sharing). Design of these building blocks has been described in the TAPAS deliverable report D9 [4], where an implementation using the JBoss application server is also presented. Here we summarise that design. We also describe how the ideas can be extended to the world of Web services.

4.3.2.1. Non-repudiable service invocation (NR-Invocation)

We developed the abstraction of *trusted interceptors* that mediate inter-organisational interaction and then model non-repudiable service invocation and non-repudiable information sharing in terms of this abstraction. The trusted interceptor abstraction is sufficiently general to apply to a variety of interaction scenarios. For example, it is not bound to any particular non-repudiation protocols but can be seen as a flexible framework in which protocols can be deployed as appropriate to the regulatory regime governing an interaction or to the trust relationships between the parties to an interaction.



(a) Service invocation

(b) Non-repudiable service invocation

Fig. 7. NR-Invocation through trusted interceptors

Figure 7(a) shows a typical two-party, client-server interaction. The client invokes a service by sending a request to the server who issues a response. Non-repudiable service invocation provides the following assurances to the client:

1. that following an attempt to submit a request to a server, either: (a) the submission failed and the server did not receive the request; or (b) the submission succeeded and there is proof that the request is available to the server; and:

2. that if a response is received, there is proof that the server produced the response.

For the server, the corresponding assurances are:

1. that if a request is received, there is proof identifying the client who submitted the request; and:

2. that following an attempt to deliver a response to the client, either: (a) the delivery failed and the client did not receive the response; or (b) delivery succeeded and there is proof that the response is available to the client.

To provide the above assurances, trusted interceptors execute a non-repudiation protocol that ensures the following:

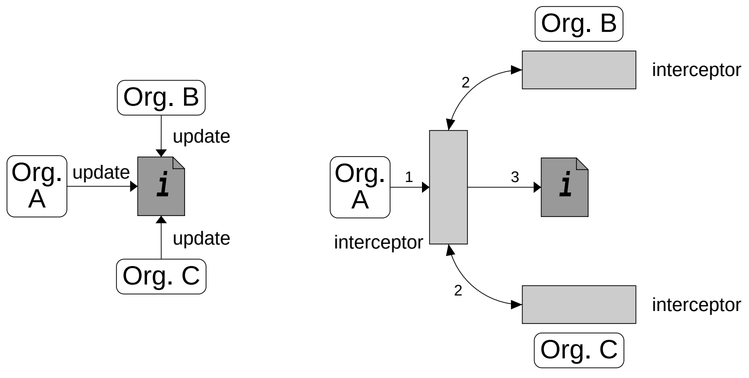
1. a request is passed to a server if, and only if, the client (or its interceptor) provides non-repudiation evidence of the origin of the request (NROreq) and the server (or its interceptor) provides non-repudiation evidence of receipt of the request (NRRreq)

2. the response is passed to the client if, and only if, the server (or its interceptor) provides non-repudiation evidence of the origin of the result (NROresp) and the client (or its interceptor) provides non-repudiation evidence of receipt of the response (NRRresp).

Non-repudiation tokens include a unique request identifier, to distinguish between protocol runs and to bind protocol steps to a run, and a signature on a secure hash of the evidence generated. Figure 7(b) models the exchange of evidence achieved by the execution of an appropriate non-repudiation protocol between interceptors acting on behalf of client and server. The client initiates a request for some service. The client's interceptor generates an NROreq token and then sends both the request and the token to the server's interceptor. The server's interceptor generates an NRRreq token and returns it to the client's interceptor. The server's interceptor then passes the request to the server to generate a response. On receipt of the response, the server's interceptor generates an NROresp token and sends both the response and the token to the client's interceptor. The interceptors ensure that irrefutable evidence of the exchange is both generated and stored.

4.3.2.2 Non-repudiable information sharing (NR-Sharing)

Figure 8(a) shows three organisations (A, B and C) accessing and updating shared information. If, for example, A wishes to update the information, then they must reach agreement with B and C on the validity of the proposed update. For the agreement to be non-repudiable: (i) B and C require evidence that the update originated at A; and (ii) A, B and C require evidence that, after reaching a decision on the update, all parties have a consistent view of the agreed state of the shared information. The latter condition implies that there must be evidence that all parties received the update and all parties know whether there was unanimous agreement to it being applied to the information. Figure 8(b) shows A proposing an update to the information shared by A, B and C. Interceptors are used to mediate each organisation's access to the information. In step 1, A attempts an update to the information. A's interceptor intercepts the update and, in step 2, executes a non-repudiable state coordination protocol with B and C to achieve the following:



(a) Information sharing (b) Non-repudiable information sharing

Fig. 8. NR-Sharing through trusted interceptors

1. That A's update is irrefutably attributable to A and proposed to B and C.

2. That B and C independently validate A's proposed update, using a locally determined and application-specific process, and their respective decisions are made available to A and are irrefutably attributable to B and C.

3. That the collective decision on the validity of the update (in this case, responses from B and C to A) are made available to all parties (A, B and C).

If the resolution of the protocol executed at step 2 represents agreement to the update then the shared information is updated in step 3. Otherwise, the information remains in the state prior to A's proposed update. Non-repudiable connect and disconnect protocols govern changes to the membership of the group of organisations sharing the information

The use of interceptor's allows us to abstract away the details of state coordination and insulate the application from protocol specifics. From the application viewpoint, the update to shared information is an atomic action that succeeds or fails dependent on the agreement of the parties sharing the information. Thus the interceptors may execute any protocol that achieves non-repudiable agreement on: the origin and state of a proposed update; the state of the shared information after application of an update; and the membership of the group that agreed to, or vetoed, the update.

4.3.2.3 Non-Repudiable Interactions with Web Services

The ideas presented in the previous two sections have been incorporated in the J2EE middleware (JBoss application server), see deliverable D9 [4]. Despite the fact that Web services are increasingly used for enabling B2B interactions, there is currently no systematic support for non-repudiation. We have therefore extended our approach to Web Services: we assume the typical pattern of XMLbased business messages that should be exchanged between partners to execute some function (such as order processing). Our design and implementation is based on a third party delivery agent (DA, a trusted third party) that takes on most of the responsibilities of evidence verification and storage and, thereby, simplifies the tasks for end users. Details are in deliverable D6 [7].

4.4. QoS monitoring and violation detection

As the name suggests, monitoring of contractual SLAs is about collecting statistical metrics about the performance of a service to evaluate whether the service provider complies with the level of QoS that the consumer expects. Such monitoring is frequently required to be carried out with the help of third parties to ensure that the results are trusted both by the provider and consumer. The state of art in the monitoring of SLAs by third parties is not yet well advanced: current contracts frequently leave SLAs open to multiple interpretations because they either contain ambiguous specifications of SLAs or no specification at all; likewise, they often do not unambiguously specify how the QoS attributes are to be monitored and evaluated.

The TAPAS QoS monitoring and violation detection subsystem (fig. 2) overcomes these shortcomings. This subsystem could be provided by the ASP or one or more trusted third parties. Second year deliverable report, D10 [5] described the design and implementation this subsystem. In TAPAS, QoS requirements in SLAs are specified precisely using the SLAng language described in deliverable report D2 [6]. Month 30 deliverable report D15 [2], chapter 3, describes how QoS monitoring of the auction application was performed.

The architecture that we developed for monitoring the level of QoS delivered by a provider to a given service consumer$_i$ at a given service point of presence ISP$_i$, is shown in Fig. 9. In the figure we assume that the interaction between the provider and the service consumer is regulated by a signed contract. The goal of monitoring is to watch what a business partner is doing, to ensure that it is honouring its obligations. We assume that monitoring is to be carried out with the help of third parties to ensure that the results are trusted both by the provider and consumer.

Notice that for the sake of simplicity only one point of presence and one service consumer is shown in the figure. However, in a general scenario, the provider would have one or more points of presence; each of them with an arbitrary number of service consumers.

To keep the figure and our discussion simple and without loosing generality we assume that the provision of the service is unilateral, that is, only the provider provides a service. Because of this, only the performance of the provider needs to be measured and evaluated. In practice, it is quite possible to find applications with bilateral service provision, where the contracting parties deliver something to each other and applications where the performance of the consumer affects the performance of the provider. We will show the generalisation of our architecture later. Though it is not shown in the figure, the assumption here is that the business between the provider and each of its service consumers (service consumer$_i$ for instance) is regulated by a signed contract. The contract clearly stipulates the SLAs at the service point of presence. Similarly the contract stipulates metrics that are to be measured and with which frequencies, to asses the performance of the provider.   With these observations in mind, it makes sense to think that a provider will have several instances of the scheme shown in the figure, that is, one instance for each of its service consumers.

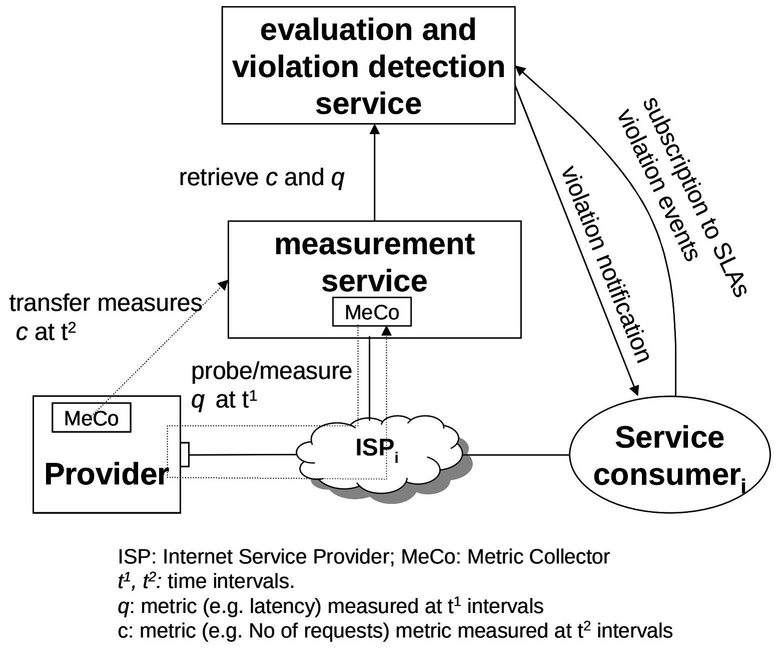

evaluation and violation detection service
retrieve c and q
measurement service
MeCo
transfer measures c at t2
probe/measure q at t1
MeCo
Provider
ISPi
Service consumeri
violation notification
subscription to SLAs violation events
ISP: Internet Service Provider; MeCo: Metric Collector
t1, t2: time intervals.
q: metric (e.g. latency) measured at t1 intervals
c: metric (e.g. No of requests) metric measured at t2 intervals

Fig. 9. Architecture for unilateral monitoring of QoS.

Two third party services are required:

- **Measurement service:** an enterprise trusted by the provider and the service consumer and with expertise in measuring a given list of metrics at specifies intervals and storing the collected results in its databases.

- **Evaluation and detection violation service**: an enterprise trusted by the provider and the service consumer. It is there to retrieve metrics from the databases of the measurement service, perform computation on them, compare the results of the computation against high or low watermarks and send notifications of violations to the service consumer when violations of SLAs are detected.

Notice that, for the sake of simplicity, in the figure we show single enterprises performing the functions of the measurement, and the evaluation and detection violation services. In practice, the measurement service can be performed by several enterprises that compensate their functionality with each other or replicate them to provide more reliability. Naturally, the evaluation and detection violation service can be realised in a similar way.

Notifications of violations are represented as events. We envisage an event notification system offering the service consumer the possibility to subscribe to events in which it is interested. It is not difficult to imagine that the service consumer can dynamically subscribe and unsubscribe to different events, perhaps in accordance with the momentary needs of the applications that it is running. To simplify the figure, notifications of violations are sent only to the service consumer; however, these notifications can be sent to other parties (for example, the provider) who express interest by means of subscriptions.

4.5. QoS aware middleware

As we have indicated earlier, practical networked systems are under increasing obligations to provide certain levels of Quality of Service (QoS) to end users. Approaches to distributed system designs have thus far assumed two broad classes of computational and communication models: in the synchronous model, processing and communication delays are considered to be uniformly distributed in a known range; in the asynchronous model on the other hand, delays are finite but without any assumption on the ability to deduce delay bounds or delay distribution. So, any bound on delays, deduced however judiciously, is subject to being violated.

Basing the design of a system on the synchronous model will require careful provisioning of system resources combined with a complete prior knowledge of the user environment. This approach is only suited to a restricted set of applications. A system design based on the asynchronous model can only guarantee eventual correctness, leaving QoS considerations as an after-thought. Experience has shown that QoS provisioning, like many non-functional system properties, cannot be achieved as an add-on feature, but rather should be a core objective in the design process.

We developed a generic system model called the *probabilistic asynchronous* model which we claim characterises the context in which many practical and the Internet-based applications are built. Specifically, our model regards that basic services and system components (e.g., network services) do meet their performance requirements most of the time, and occasionally they may not; only when they don't, they adhere to the classical asynchronous model. Our design approach draws from, and combines probabilistic design techniques and asynchronous ones. Its objective is to render systems that adaptively meet QoS obligations to the end users when system components meet their QoS guarantees or violate them only marginally; eventual correctness is never compromised when components fail in their QoS obligations.

There are several factors that can perturb a system's ability to maintain the QoS level desired by an end user. For example, a new user may request services with some specified QoS or an existing user may dynamically request for an enhanced level of QoS for the on-going service provisioning. In these occasions, the system has to be able to evaluate if the request can be met without jeopardising the QoS commitments already in force. Thus, a QoS enabled system should essentially be able to evaluate the feasibility of QoS provisioning and, where and if possible, to adapt itself when QoS perturbs are encountered. The system in essence needs to be *QoS adaptive* in nature. The adaptation can range from adjusting the operational parameters (e.g., reducing the level of redundancy) to, at the extreme end, deploying additional resources such as computational capacity, bandwidth and storage.

In TAPAS we developed two QoS aware middleware systems/services: group communication and application server.

4.5.1. QoS-Enabled Group Communication

In TAPAS, we focused on the development of QoS enabled multi-party communication to support applications that require information dissemination to many processes (e.g., in distributed games). For such applications we concentrated on QoS properties of fault tolerance, availability, and timeliness. Currently, strict separation exists between the middleware that executes application services and the network, so providing services that go beyond ‘the best effort’ is difficult.

The designed system, then, offers a so-called *QoS-Enabled Reliable Multicast Service*, and is capable of providing probabilistic guarantees on latency bounds as well as probability of success. Guarantees rely on a previous QoS negotiation with the system user, with the system capable of either accepting or refusing the service request based on an analytical approximation of the network behaviour in the near future.

4.5.1.1. Architecture

The group communication (GC) system designed for the TAPAS project is composed by three main components, described in deliverable report D8 [13].

The *Negotiation Component* (*NC*) is in charge of negotiating the QoS request with the system’s client. It relies on an analytical approximation of the network behaviour, and provides the *Reliable Multicast Service Component* with a series of parameters fine-tuned so as to fulfil the agreed service level.

The *Reliable Multicast Service Component* (*RMC*) realizes the Reliable Multicast logic. Its algorithm is based on the originator broadcasting ρ redundant copies of the same message, where ρ is the *level of redundancy*, to the other group members. Each broadcast is separated by a η interval time, chosen to be as small as possible but big enough to guarantee failure independence. Both η and ρ are dynamically generated from the NC after (successful) negotiations, and passed afterwards to RMC. On the receiving side, upon reception each receiver sets a timeout within which it expects to receive next copy of the message. Reception of such copy in time means that the protocol is progressing well. If the timeout expires, the receiver pessimistically assumes the originator to have crashed, and tries to appoint itself as new broadcaster. To avoid multiple receivers to act this way at the same moment, drastically increasing the message overhead, a selection procedure guarantees that only one receiver will be selected as new broadcaster. The protocol features, moreover, two types of adaptation, to reduce message overhead and to adapt to unforeseen environmental (i.e. network) changes. They both rely on relaxation/stretching of some system timeouts.

The *Network Monitoring Component* (*NMC*) monitors the network to detect a set of parameters useful to the NC to approximate network’s behaviour. Parameters monitored are average packet delay and loss, average jitter and an approximation of the packet delay distribution curve into a well known statistical distribution. The NMC is based on a mechanism that gathers all needed information by means of an RTT-based technique, whose data is after processed and averaged over fixed amounts of time so as to obtain desired data. Once data is calculated, the set of four parameters is passed to the NC that uses them for the approximation.

The idea behind this organization is that the RMC offers a service that is based on certain guarantees, provided by the NC and validated by the NMC.

4.5.1.2. Components Interoperation

The architecture is assumed to be instantiated on each member, and components are assumed to interoperate in a local context: each process using the system will have all three components locally instantiated. When a user comes and asks for a service, it provides the system with a required delay and probability of success. The first component to be instantiated is the NC that, in turn, instantiates and starts the NMC. The NC takes then the user-requested parameters and match them against the ones obtained by means of approximations (evaluated on the base of network metrics provided by the NMC). Approximation generation involves production of a set of parameters that determine RMC properties, which are fine-tuned so as to put RMC in a situation of being able to fulfil the requested service. Once negotiation is over and successful, the RMC is instantiated and parameters are passed to start the protocol. Even after the protocol has started, both the NC and the NMC continue their own execution: the NMC monitors the network on a constant basis, and it constantly passes network metrics to the NC that, in turn, re-approximates the network behaviour and eventually updates values on behalf of the RMC so as to keep its execution consistent with the expected network behaviour.

4.5.1.3. Integration

The Application Server platform used for the TAPAS project is *JBoss* [14]. Group Communication is used, in JBoss, to implement the transport layer of the clustering technique. This whole transport layer is based on *JGroups* [15], whose architectural core is represented by a stack of `Protocol` objects that the user spans in the client application. Each `Protocol` realizes a specific task, and, taking care of eventual dependencies, juxtaposition of multiple objects offer a more complete service. Our system has been converted into JGroups format and included in the set of available `Protocols` with the name `RMCAST`.

4.5.2. QoS-aware application server

We have designed and developed a family of middleware services that extend current, J2EE-based, open-source application server technology so as to enable it to meet QoS application requirements, such as timeliness, availability, and throughput. We have termed QoS-aware application server an application hosting environment designed to honour the hosting Service Level Agreements (SLAs); i.e., the SLAs that bind that environment to the applications it hosts.

Current J2EE-based application server technologies (e.g., JBoss [14], JOnAS [16], WebLogic [17], WebSphere [18]) can meet only partially QoS requirements such as availability, timeliness, security, and trust of the applications they host, as these technologies are not fully instrumented for meeting those requirements (i.e., they are not designed to be *QoS-aware*).

In order to construct such an environment, the hosting SLAs are to be enforced, and monitored at run-time. This entails that possible deviations of the QoS delivered by a hosting environment from that expected by the application, and specified in the relative hosting SLA, must be detected, and corrective actions taken, before this SLA be violated.

In order to carry out SLA enforcement and monitoring within an application service environment, we have developed two principal middleware services, namely a Configuration Service (CS) and a Monitoring Service (MS), which can be incorporated in the current application server technology.

In addition, as advanced application server technology such as JBoss, enables hosting of distributed, component-based applications within a cluster of application servers, the middleware services mentioned above have been designed so as to exercise SLA enforcement and monitoring over clustered application servers.

Typically, clustered servers can be used in order to provide the applications with a highly available, fault tolerant, and scalable hosting environment. For both performance and fault tolerance purposes, load balancing is to be deployed within a cluster of application servers so as to distribute appropriately the computational load amongst those servers.

Current open source, J2EE technology (e.g., JOnAS, JBoss) offers load balancing services which implement simple, non adaptive load distribution strategies. However, static load distribution may affect the QoS delivered by an application server cluster. In order to overcome this shortcoming, we have developed a Load Balancing Service that incorporates an adaptive load balancing strategy. This strategy can cope effectively with run time variations of both the clustered servers computational load, and the cluster configuration.

The principal responsibilities of the Configuration, Monitoring, and Load Balancing Services introduced above are summarised below (for details, see deliverable D11 [3]).

The CS is responsible for configuring an application hosting environment (be this a single application server, or a cluster of servers) so that it meets effectively the hosting SLA of a customer application. Thus, in essence, the CS takes in input a customer application hosting SLA, and discovers the available system resources that can honour that SLA. Provided that those resources be sufficient to meet that input SLA, the CS reserves those resources, generates a so-called “resource plan” (i.e., the QoS levels the application expects from the hosting environment resources) for the hosted application, and sets up the QoS-aware application hosting environment for that application. In case the CS discovers that there are not sufficient resources to host that application, it returns an exception. (Typically, one such an exception can be handled either by rejecting the application hosting request, or by offering a reduced service, for example, depending on the policy implemented by the Application Service Provider owning the hosting environment.)

The MS is responsible for monitoring the hosting environment at application run time, so as to detect possible violations of the hosting SLA. In order to prevent those violations, the MS takes appropriate actions if it discovers that the QoS delivered by the hosting environment reaches a predefined *warning* point (i.e., a level of QoS beyond which SLA violation may occur). Thus, for example, the MS can make use of a predefined “overload” warning point in order to detect dangerous load conditions that may lead to server overloading. In case that warning point is reached, the MS invokes the CS, and requires that the application hosting environment be reconfigured appropriately, so that it can adapt to the new load conditions, and continue to honour the application SLA.

It is worth observing that the CS and the MS exercise their activity over both the internal resources of each application server instance in the application hosting environment, and the set of clustered server instances that form this environment; i.e., they are responsible for configuring and monitoring both a single application server and a cluster of servers.

Owing to this observation, the CS and the MS can be conveniently thought of as operating at two distinct levels of abstraction, that we have termed the *micro-resource* and the *macro-resource* levels, respectively. The former level consists of resources, such as server queues, thread and connection pools, internal to each individual application server; the latter level consists of such resources as the group of clustered application servers, and their IP addresses.

Thus, for example, the CS at the micro-resource level is responsible for sizing appropriately an application server request queue, in order to enable that server to deal with an (anticipated) maximum number of concurrent requests, and to maintain its responsiveness. In contrast, in order to meet possible load balancing and responsiveness requirements, the CS at the macro-resource level may have to modify the cluster configuration at application run time, e.g., by enabling one (or more) new application server instance(s), or by replacing a crashed application server instance with an operational one.

The MS at the micro-resource level monitors the QoS (e.g., throughput, response time) delivered by the single application server, and requires the *server reconfiguration* when the delivered QoS reaches a predefined warning threshold. At the macro-resource level, instead, the MS monitors the QoS delivered by the clustered application servers, and requires *cluster reconfiguration* in case the cluster delivered QoS reaches a predefined warning threshold.

The interface between the micro and the macro resource levels can be thought of as consisting of primitive operations and data objects that enable the macro-resource level both to obtain micro-resource QoS data (e.g., server throughput, server response time, JVM free memory) from the micro-resource level, and to provide this level with QoS requirements derived from the SLA.

The Load Balancing Service for clustered application servers has been designed so as to support both “request-based” (or “per-request”) load balancing, and a “session-based” (or “per-session”) load balancing.

In “request-based” load balancing, each individual client request is intercepted by the Load Balancing Service, and dispatched to an application server for processing, according to some specific load distribution policy. Thus, two consecutive requests from the same client may be dispatched to two different servers.

In contrast, in “session-based” load balancing, a specific client session (i.e., a sequence of client requests) is created in one of the clustered application server, at the time a client program requires access to the application hosted by that server; every future request from that client will be processed by that server (these client-server sessions are termed “sticky sessions”). Thus, the Load Balancing Service intercepts each client request and, depending on the sticky session the request belongs to, dispatches it to the appropriate server.

Fig. 10 shows the main features of our load balancer. User requests are being sent to a host of the cluster whereby a HTTP load balancer is working as a reverse proxy. Hence, this load balancer is responsible for (i) intercepting all the requests coming from the clients, (ii) choosing the nodes to forward the requests in order to balance the load (dashed lines in Fig. 10). This is carried out by the load balancer scheduler which chooses the target node according to an adaptive load balancing strategy), (iii) receiving the response back from the chosen hosts and finally (iv) giving back the response to the client.

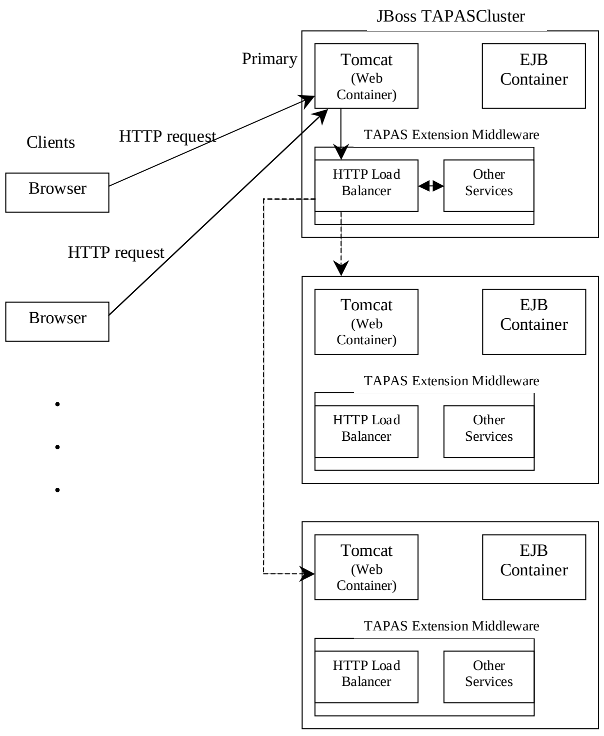


Fig. 10. Load balancing

Finally, we have carried out an extensive experimental evaluation of an implementation of our services, integrated in a cluster of JBoss application servers, and compared and contrasted our implementation with a cluster of standard JBoss node (i.e., not including our additional services). The results of this experimental evaluation indicate the adequacy of our approach as, in summary, our services i) add negligible overheads to the standard JBoss, yet providing a robust and reconfigurable environment, and ii) enable a hosting environment that can respond effectively the hosting SLA without the need for resources over-provision. In contrast, standard application server technology, such as JBoss, may well be able to meet a hosting SLA, indeed; however, this is to be done at the cost of resource over-provision.

4.6. Integration and Evaluation

An instance of the TAPAS architecture (fig. 2) has been implemented. We have also implemented the auction application (fig. 1) to run on the TAPAS platform. The auction application runs on a cluster of application servers(assumed to belong to an ASP) that has been enriched with TAPAS features. A load generator has been used to exercise the application.

The auction application scenario was chosen, because it provides a variety of interacting parties, each of which is interested in different aspects of QoS. However, during the project runtime discussion led to the basic question, if an application shall or shall not be aware of QoS-monitoring and related services. If is obviously beneficial to be able to deploy a QoS-unaware application to a QoS-platform, thus gaining QoS-monitoring and even SLA-aware reconfiguration of resources. Hence the auction application could be realized indeed without explicitly using TAPAS technology. On the other hand it should be noted, that real-world applications tend to follow J2EE concepts only to a certain point, because in some cases there currently still are better solutions outside J2EE, e.g. when accessing large sets of data.

The integration, demonstration and evaluation exercise has been described in deliverable reports D15 [2] and D14 [19].

4.6.1. SLA monitoring

A third party service is able to monitor the electronic service SLA (specified in SLang) reporting any violations.

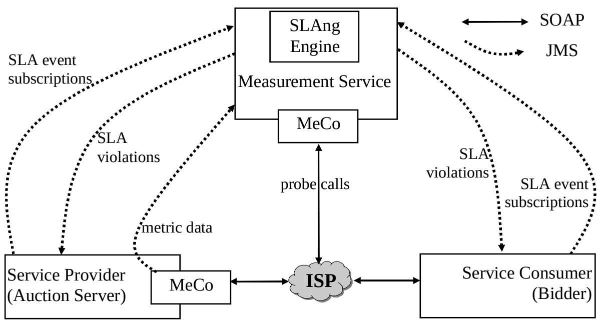


Fig. 11. SLA monitoring architecture

Different mechanisms are provided by TAPAS to suite the different types of stakeholders. An ASP can of course read the data that has been produced by the TAPAS middleware. The same data can be used by an external party, be it the client himself or a trusted third party to check it against SLA regulations. When considering the stakeholders it should be mentioned that the TAPAS middleware allows to monitor the QoS of External Services that are used by a hosted application. They can easily be monitored regarding their SLA fulfilment by using the TAPAS technology by wrapping the otherwise TAPAS-unaware external service into a dedicated component, be it a single Session Bean, a JCA adaptor or even an own J2EE application. For existing applications this of course would lead to the necessity of restructuring the conventional service invocation to use such a session bean, for instance, if the application architecture does not provide own wrapping elements.

4.6.2. Terms and conditions contract monitoring and enforcement

The inter-organisation interaction regulation system of the TAPAS architecture ensures that only valid interactions (as defined in the contract) take place and non-repudiation information is automatically generated. It was relatively easy to construct FSMs for representing contractual conversations.

In normal operation, an auctioneer and at least two bidders will submit valid requests to which the middleware will attach well-formed non-repudiation evidence. The requests will follow the full lifecycle of an auction from bidder registration to auction closure. The evidence for correct operation of the middleware will be that:

1. for each correctly behaving client, there is a full set of entries in both client-side and service-side non-repudiation logs. The logs will in effect be a non-repudiable trace of the execution of the application as represented by the client requests. Each NRRreq entry in a log will include a decision attesting to the validity of the related request.

2. from the point of view of each correctly behaving client, the auction will progress normally to completion, and the application state will change in response to their requests.

If a bidder uses an invalid key to sign request data then:

1. the bidder’s client-side non-repudiation log will contain an NROreq entry but, since the request was mal-formed and failed verification, there will be no corresponding entry in the service-side log.

2. no NRRreq will be returned by the service. Instead, an error will be propagated to the client indicating failure of the request.

3. the request will not be passed to the application and application state will not be changed as a result of the request

Contract violation: in this case one or more clients will generate a request that violates contract terms and conditions. For example, in an auction round, a bidder will attempt to place a bid after a previous bid from them has been accepted in the given round. In this case, the following happens:

1. both bidder and service-side logs contain NROreq and NRRreq evidence. However, the NRRreq evidence will attest that the related request was invalid with respect to contract.

2. an error will be propagated to the client indicating failure of the request.

3. the request will not be passed to the application and application state will not be changed as a result of the request.

4.6.3. QoS enabled application server

We have carried out an extensive experimental evaluation of our platform, and compared and contrasted our implementation with a cluster of standard JBoss nodes (i.e., not including our additional services). These are described in detail in D11 [3]. For our initial tests we have used a cluster configuration consisting of three application servers running on three Linux machines interconnected by a 100Mb Ethernet LAN. Each machine was 2.4 GHz Pentium 4, with 512MB of RAM. The three machines were dedicated to our experiments.

The test discussed here was intended to show the effectiveness of the TAPAS clustering mechanism in order to prevent SLA violations. Specifically, we have considered the following case. We have assumed that a possible ASP policy dictates that application deployment be carried out using the minimal set of resources which are required to run the application; hence, for example, an application is deployed on a single application server, if possible. A single application server may reach the response time and throughput breaching points, thus violating the hosting SLA, if it is not instrumented to reconfigure dynamically, prior to the SLA violation.

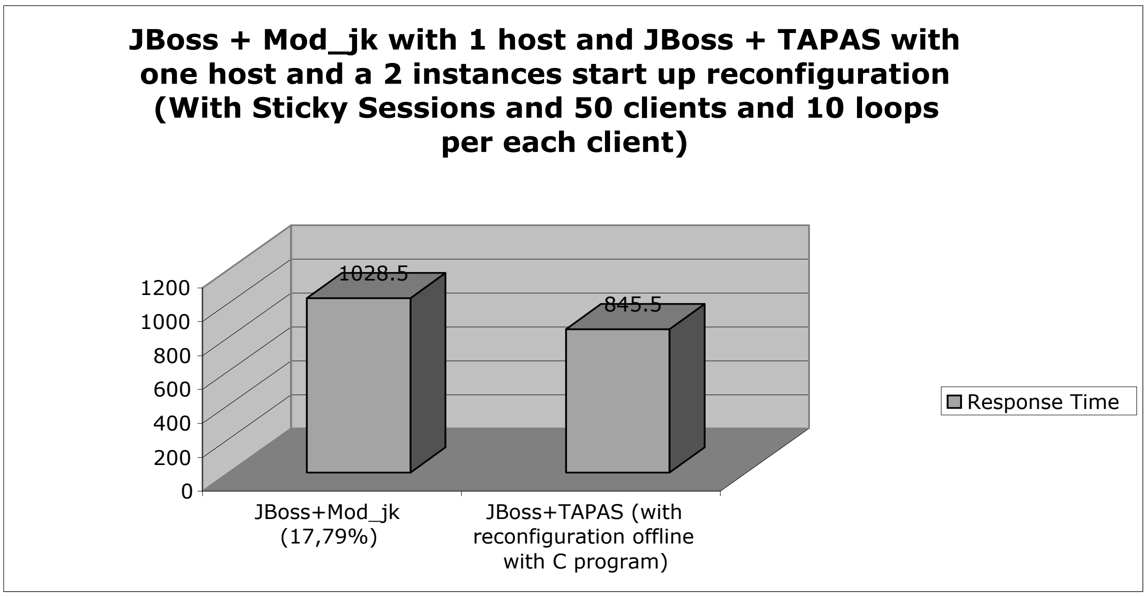


Figure 12: JBoss vs. TAPAS JBoss: response time

Thus, we have compared and contrasted the deployment of our test application on a standard JBoss application server, equipped with mod-jk, with that of the same application on a TAPAS extended JBoss server, with per-session load balancing enabled, and two spare application servers for use for dynamic reconfiguration purposes. In this test, 50 clients were concurrently accessing the hosted application. The results of this test are depicted in Figures 12 and 13.

Figure 12 shows that the TAPAS middleware, with dynamic reconfiguration and load balancing, allows the application server to maintain the average response time below 850 ms. The standard JBoss, instead, provides an 18% slower response time than the TAPAS extended JBoss, approximately, as it is unable to apply reconfiguration, when necessary.

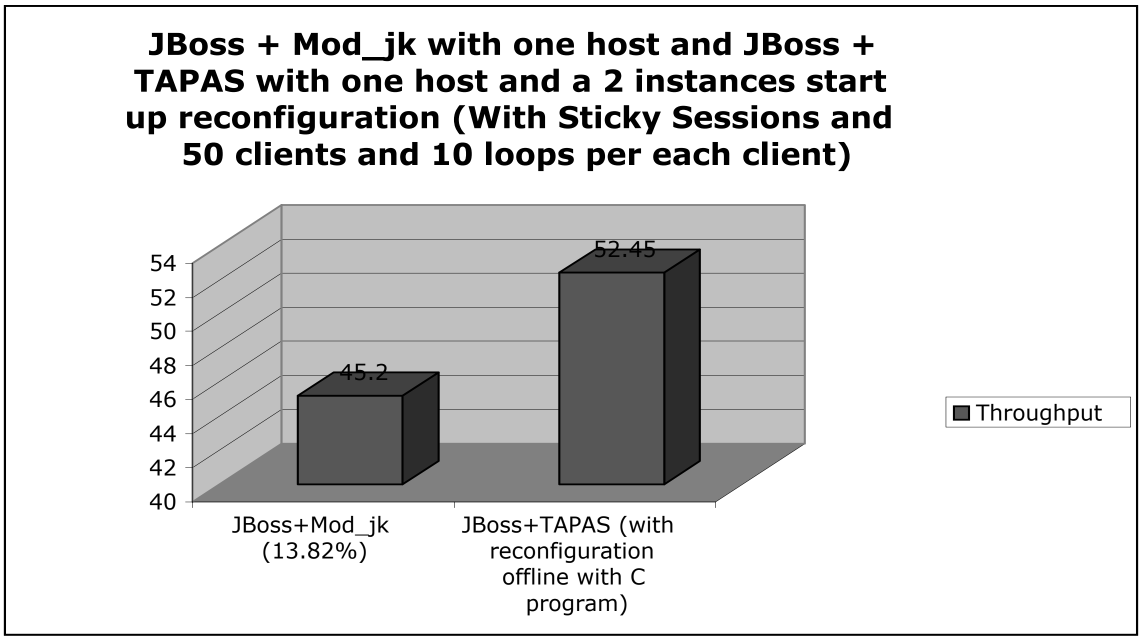


Figure 13: JBoss vs. TAPAS JBoss: throughput

Similarly, Figure 13 shows that the standard JBoss throughput is approximately 14% lower than that generated by the JBoss application server extended with the TAPAS middleware (owing to the same motivation as above).

4. 6.4. QoS enabled group communication

The auction application is not ideal for illustrating the features of the group communication system. We have therefore developed a separate application: a distributed collision detection system as required in distributed games/virtual reality applications.

A real-time collision detection service that supports QoS guarantees suitable for satisfying the collision detection requirements of graphically represented 3D virtual worlds. Our approach is a distributed one, allowing our real-time collision detection service to scale to support complex virtual worlds via the use of clustered servers. We base our service on QoS enabled group communications middleware to enable a deterministic approach to satisfying the QoS requirements of virtual worlds. Our approach is adaptive in that we always guarantee real-time requirements of a virtual world via the adaptation of the accuracy of collision detection in runtime to offset varying processing availability or message latency. Our experiments, described in deliverable D15, show that in such a system, QoS enabled group communication system is invaluable.

5. Deliverables

Project results are documented in the following deliverables.

5.1. Deliverable Reports

D1: Application Hosting and Networking Requirements

D2: Specification Language for SLAs

D3: Method for Service Composition and Analysis

D4: Service Composition & Analysis Tool

D5: TAPAS Architecture: Concepts and Protocols

D5Supplement: An Overview of the TAPAS Architecture

D6: TAPAS Architecture

D7: TAPAS Architecture: QoS Enabled Application Servers

D8: QoS adaptive Group Communication

D9: Component middleware for Trusted Coordination

D10: QoS Monitoring of Service Level Agreements

D11: QoS-aware Application Server: Design, Implementation, and Experimental Evaluation

D12: First year Evaluation and Assessment Report

D13: Second Year Evaluation and Assessment Report

D14: Third Year Evaluation and Assessment Report

D15: TAPAS QoS-aware Platform: technology and demonstration

D16: Dissemination and Use Plan

D17: Updated Dissemination and Use Plan

D18: Technological Implementation Plan

D19: Project Presentation

D20: Final Report

PP1-3: Periodic Progress Reports

PM1-3: Periodic Management Reports

5.2. Software

TAPAS software developed under deliverables D4 (service composition and analysis), D8 (QoS aware group communication), D9 (trusted coordination), D10 (QoS monitoring) and D11 (QoS aware application server) has been made available in open source form at sourceforge.net (http://tapas.sourceforge.net). Given the use of JBoss application server within the project, it seems natural to seek closer collaboration with the JBoss organisation that develops the open source application server. Accordingly, the project team has enlisted itself as academic partners with them (there is a link to TAPAS on the main JBoss site)

6. Dissemination and Exploitation

6.1. TAPAS Industrial Advisory Board

To help assess the progress we are making in the project, we formed an Industrial Advisory Board. The role of the advisory board was (i) to guide and validate our research; and (ii) to provide a means of dissemination of our results. The Board met at the end of each year. As a matter of fact, the members of the advisory board acted as external evaluators of the progress of our work. The appendix describes the membership of the Board.

6.2. Workshops and Conferences

Project results were presented at a number of international conferences and workshops. The full list appears in the deliverable report D14. We would like to highlight here one workshop that was organised by TAPAS partners.

The Workshop on Quality of Service for Application Servers 2004 (QoSAS'04), in conjunction with IEEE 23rd Symposium on Reliable Distributed Systems(SRDS 2004), Jurerê Beach Village, Santa Catarina Island, Brazil, 17 October2004 was organised by TAPAS members to provide a forum for researchers, application designers and users to review, discuss and learn about new approaches and concepts in application server QoS development. The topics of the workshop reflected the work carried out in the TAPAS project. As such, the workshop provided an ideal opportunity to disseminate the ongoing results from the TAPAS project to an audience of academics and industrial professionals. In addition, work from others (non TAPAS members) were also presented at the workshop, providing an excellent opportunity for members of the TAPAS project to learn from other academics/industrialists carrying out work in similar areas.

All the papers submitted to the workshop were reviewed by at least three members of the Program Committee (constituted from industrial/academic experts in distributed systems research). Seven papers were selected as regular papers. These paper presentations were complemented by an invited talk and a panel session.

The workshop was organised in five sessions. The first session was an invited talk given by Dr. Graeme Dixon from IBM on recent developments of IBM's application server (WebSphere). The focus of the second session (adaptability) was on how application servers may be configured during run-time to make best use of processing resources while still satisfying QoS guarantees. The third session (scheduling) presented work aimed at satisfying the QoS requirements of real-time systems. The fourth session (web services) presented research associate to satisfying QoS requirements of service based architectures. Finally, the fifth session was a panel discussion entitled "research challenges for application server developers". The workshop was a success, attracting paper submissions from over ten different countries and delegates from both academia and industry.

6.3. Research papers and articles

The project has been particularly successful in producing a number of refereed research publications. These are listed here.

Year 2002

W. Emmerich:

Distributed Component Technologies and their Software Engineering Implications

Proc. of the 24th Int. Conference on Software Engineering, Orlando, Florida. pp. 537-546. ACM Press. 2002

G. Piccinilli, W. Emmerich, C. Zirpins and K. Schuett:

Web Services Interfaces for Inter-organizational Business Processes: An Infrastructure for Automated Reconciliation

In Proc. of the 6th IEEE Int. Conference on Enterprise Distributed Object Computing, Lausanne, IEEE Computer Society Press. pp. 285-292. 2002

W. Emmerich and N. Kaveh:

Component Technologies: Java Beans, COM, CORBA, RMI, EJB and the CORBA Component Model

Proc. of the 24th Int. Conference on Software Engineering, Orlando, Florida. pp. 691-692. ACM Press. 2002

N. Cook, S.K. Shrivastava and S.M. Wheater:

Distributed Object Middleware to Support Dependable Information Sharing between Organisations

IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2002), June 2002, Washington DC

S.K. Shrivastava:

Middleware for supporting inter-organisational interactions

Proceedings of Workshop on Future Directions in Distributed Computing (FuDiCo), Bertinoro, Italy, June 02

E. Turrini and F. Panzieri:

Using P2P Techniques for Content Distribution Internetworking: A Research Proposal

in proceedings of the 2nd IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, 5-7 Sept. 2002

G. Lodi:

End-to-end QoS-aware Middleware Services

7th Cabernet Radical Workshop, Bertinoro (FC), Italy, 13-16 Oct. 2002

E. Turrini

A Platform for Request Routing in Content Distribution Internetworks

7th Cabernet Radical Workshop, Bertinoro (FC), Italy, 13-16 Oct. 2002

N. Mezzetti and F. Panzieri:

The Data Grid: Security and Privacy Issues

Proc. 4th European Dependable Computing Conference, Toulouse (F), 22-25 Oct. 2002

A. Aldini, M. Bernardo, R. Gorrieri and M. Roccetti:

QoS Evaluation of IP Telephony Services: A Specification Language Based Simulation Software Tool

Systems Analysis Modelling Simulation, Taylor and Francis Group Pub., accepted for publication, December 2002

Year 2003

A. Di Ferdinando, P. McKee and A. Amoroso:

A Policy Based Approach for Automated Topology Management of Peer To Peer Networks and a Prototype Implementation

G.N. Rodrigues, G. Roberts, W. Emmerich and J. Skene:

Reliability Support for the Model Driven Architecture

In Proceedings of the ICSE Workshop on Software Architecture for Dependable Systems 2003 (RRES03: Reliability Support), ICSE 2003

D.D. Lamanna, J. Skene and W. Emmerich:

SLAng: A Language for Service Level Agreements

In Proceedings of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems (LSE03: SLAng), 2003, (pages 100-106) IEEE Computer Society Press.

D.D. Lamanna, J. Skene and W. Emmerich:

SLAng: A Language for Defining Service Level Agreements

Accepted for Poster presentation, Middleware 2003, Rio de Janeiro, Brazil

N. Kaveh and W. Emmerich:

Validating Distributed Object and Component Designs

in Formal Methods for Software Architecture, Springer Verlag, Lecture Notes in Computer Science, vol. 2804, 2003, pages 63-91, Edited by M. Bernardo and P. Inverardi KE03: Validating)

J. Skene and W. Emmerich:

Model Driven Performance Analysis of Enterprise Information Systems

In Proc. of International Workshop on Test and Analysis of Component Based Systems, Warsaw, April 13th, 2003 in conjunction with European Joint Conferences on Theory and Practice of Software (ETAPS) 2003. And in Electronic Notes in Theoretical Computer Science, April 2003, vol. 82, number 6 (SE03: Performance)

C. Molina-Jimenez, S.K. Shrivastava, E. Solaiman and J. Warne:

Contract Representation for Run-time Monitoring and Enforcement

IEEE Conference on Electronic Commerce (CEC'03), Newport Beach, CA, June 2003, pp. 103-110

A. Amoroso and F. Panzieri:

A scalable architecture for responsive auction services over the Internet

TR UBLCS-2003-09, Dept. of Computer Science, University of Bologna, June 2003

E.Turrini:

Dependability Issues in Content Distribution Internet-working

in Proc. of the International Conference on Dependable Systems and Networks, Student Forum, June 2003

G. Lodi and F. Panzieri:

JBoss vs. JOnAS

TAPAS Project Internal Report, June 2003

W. Beckmann and M. Koßmann:

An Answer to the JBoss vs. JOnAS Comparison

adesso AG, 30 June 2003

J. Crowcroft, S. Hand, R. Mortier, T. Roscoe and A. Warfield:

QoS`s Downfall: At the bottom, or not at all!

In Proceedings of the ACM Workshop on Revisitng IP Quality of Service (RIPQoS), pp. 109-114, August 2003, Karlsruhe, Germany

P. Gevros:

Internet Service Differentiation using Transport Options: the case for policy-aware congestion control

In Proceedings of the ACM Workshop on Revisitng IP Quality of Service (RIPQoS), pp. 151-157, August 2003, Karlsruhe, Germany

P.D. Ezhilchelvan and S.K. Shrivastava:

Systematic Development of a Family of Fair Exchange Protocols

Seventeenth Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Estes Park, Colorado, August 2003

S. Ferretti and M. Roccetti:

On Designing an Event Delivery Service for Multiplayer Networked Games: An Approach based on Obsolescence

Proc. 7th International Conference on Internet, Multimedia Systems and Applications (IMSA 2003), Honolulu, (HI), August 2003

M. Roccetti and P. Salomoni:

The Design and Performance of a Wireless Internet Application for Supporting Multimedia City Guides

Proc. IEEE International Conference on Information Technology: Research and Education (ITRE 2003), Newark (NJ), August 2003

N. Mezzetti:

Towards a Model for Trust Relationships in Virtual Enterprises

In Proceedings of 14th Database and Expert Systems Applications (DEXA'03) Workshop, 1 - 5 September 2003, Prague (Czech Republic)

J. Skene, G. Piccinelli and M. Stearns:

Modelling Electronic Service Systems Using UML

in Workshop on Service Based Software Engineering, FM2003-SBSE, Pisa, Italy, 2003, September, "Technische Universität München", pages15—30, url: http://www.cs.ucl.ac.uk/staff/J.Skene/phd/sbse2.pdf (SPS03: Modelling)

A.I. Kistijantoro, G. Morgan, S.K. Shrivastava and M.C. Little:

Component Replication in Distributed Systems: a Case study using Enterprise Java Beans

22nd IEEE/IFIP Symposium on Reliable Distributed Systems (SRDS2003), Florence, October 2003, pp. 89-98, ISBN: 0-7695-1955-5

J.Skene and W. Emmerich:

A Model Driven Architecture Approach to Analysis of Non-Functional Properties of Software Architectures

In Proceedings of the 18th IEEE Conference on Automated Software Engineering (SE03), October 2003, Montreal, Canada (pages 236-239), 2003. IEEE Computer Society Press

E. Turrini:

A Protocol for exchanging performance data in Content Distribution Internetworks

8th CaberNet Radicals Workshop, Ajaccio, Corsica, 5 - 8 October 2003

N. Cook, S.K. Shrivastava and S. Wheater:

Middleware Support for Non-repudiable Transactional Information Sharing between Enterprises

4th IFIP International Conf. on Distributed Applications and Interoperable Systems, DAIS 03, November 2003, Paris

E. Solaiman, C. Molina-Jimenez and S.K. Shrivastava:

Model Checking Correctness Properties of Electronic Contracts

International Conference on Service Oriented Computing, Trento, November, 2003. Lecture Notes in Computer Science Vol. 2910, Springer (2003).

E. Turrini:

An architecture for Content Delivery Networks federation

CaberNet Plenary Workshop, 5-7 November 2003, Porto Santo, Portugal

E. Turrini:

Analyzing web response time

CaberNet Plenary Workshop, 5-7 November 2003, Porto Santo, Portugal

Year 2004

C. Molina-Jimenez, S.K. Shrivastava, E. Solaiman and J. Warne:

Run-time Monitoring and Enforcement of Electronic Contracts

Electronic Commerce Research and Applications (ECRA), Elsevier, Vol. 3, No. 2, 2004

G. Denaro, A. Polini and W. Emmerich:

Early Performance Testing of Distributed Software Applications

in Proceedings of the 4th Int. Workshop on Software and Performance, San Francisco, January 2004 (ACM Press)

E. Turrini and V. Ghini:

A Protocol for exchanging performance data in Content Distribution Internetworks

3rd International Conference on Networking (ICN'04), February 29 - March 4, 2004 – Creole Beach Hotel, Gosier, Guadeloupe, French Caribbean

N. Cook, P. Robinson and S.K. Shrivastava:

Component Middleware to Support Non-repudiable Service Interactions

IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2004), Florence, June 2004, pp. 605-614.

C. Molina-Jimenez, S.K. Shrivastava, J. Crowcroft and P. Gevros:

On the Monitoring of Contractual Service Level Agreements

The First IEEE International Workshop on Electronic Contracting (WEC), July 2004, San Diego

J. Skene, D. Lamanna and W. Emmerich:

Precise Service Level Agreements

In Proc. of the 26th Int. Conference on Software Engineering, Edinburgh, UK, Sept. 2004. ACM Press

L. Capra. "Engineering Human Trust in Mobile System Collaborations". In Proc. of the 12th International Symposium on the Foundations of Software Engineering (SIGSOFT 2004/FSE-12). November 2004.

L. Capra. "Towards a Human Trust Model for Mobile Ad-hoc Networks". In Proc. of 2nd UK-UbiNet Workshop. May 2004, London, United Kingdom

N. Mezzetti, ``Enabling Trust-Awareness in Naming Services", *Proceedings of the 1st International Conference on Trust and Privacy in Digital Business*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3184, pp. 20-29, 2004.

N. Mezzetti:

A Socially Inspired Reputation Model, Lecture Notes in Computer Science, Springer-Verlag GmbH, Volume 3093 / 2004: Public Key Infrastructure: First European PKI Workshop: Research and Applications, Samos Island, Greece, June 25-26, 2004, Editors: Sokratis K. Katsikas, Stefanos Gritzalis, Javier Lopez

Giorgia Lodi, Fabio Panzieri:

"QoS-aware Clustering of Application Servers", Proc. 1st IEEE International Workshop on Quality of Service in Application Servers (QoSAS 2004), in conjunction with 23rd Symposium on Reliable Distributed Systems (SRDS 2004), Jurerê Beach Village, Santa Catarina Island, Brazil, 17 October 2004.

Davide Rossi, Elisa Turrini:

"Testing J2EE clustering performance and what we found there", Proc. 1st IEEE International Workshop on Quality of Service in Application Servers (QoSAS 2004), in conjunction with 23rd Symposium on Reliable Distributed Systems (SRDS 2004), Jurerê Beach Village, Santa Catarina Island, Brazil, 17 October 2004.

Di Ferdinando A., Ezhilchelvan P.D., Mitrani I.:

"Design and Evaluation of a QoS-Adaptive System for Reliable Multicasting" . In Proceedings of the 23rd Symposium on Reliable Distributed Systems (SRDS04), Florianópolis, Brazil, October 2004.

G. Ferrari, S. Shrivastava, P. Ezhilchelvan:

„An Approach to Adaptive Performance Tuning of Application Servers“. In Proc. 1st IEEE International Workshop on Quality of Service in Application Servers QoSAS 2004), in conjunction with 23rd Symposium on Reliable Distributed Systems (SRDS 2004),Santa Catarina Island, Brazil, 17 October 2004.

Year 2005

Davide Rossi, Elisa Turrini:

Analyzing Performance Data Exchange in Content Delivery Networks, Proc. International Conference on

Networking (ICN '05), Reunion Island, April 17-21, 2005 .

N. Mezzetti:

Design and Evaluation of a Trust-Aware Naming Service, To appear in Computer Systems Science and Engineering Journal, 2005.

<u>To appear:</u>

Wolfgang Emmerich, James Skene:

Engineering Runtime Requirements-Monitoring Systems using MDA Technologies, Symposium on Trustworthy Global Computing (part of ETAPS in Edinburgh, April 2005), to be published in LNCS by Springer Verlag

Carlos Molina-Jimenez, Jim Pruyne and Aad van Moorsel:

Software Architectures for Service Level Agreements and Contracts, To appear in "Architecting Dependable Systems III", to be published in the LNCS series by Springer in 2005, Editors: Rogerio de Lemos, Cristina Gacek and Sascha Romanovsky

Articles

EU-Forschungsprojekt fördert ASP-Markt (EU research project encourages ASP market)

Industriemanagement (Industrial management), pp 77, GITO mbH Verlag,1/2003, Berlin, Germany

Marktbelebung durch mehr Sicherheit und Qualität (Market upturn by more security and quality)

Versicherungswirtschaft (Insurance economy) pp 67, Verlag Versicherungswirtschaft GmbH, 1/2003, Karlsruhe, Germany

Die zweite ASP-Welle ist auf dem Weg (The second wave of ASP is on it's way) Computerwoche, pp 34-35, IDG Business Verlag GmbH, 26/2003, München, Germany

TAPAS macht Appetit auf ASP (TAPAS whets one's appetite for ASP) eCommerce Magazin 06-07 /2003, IWT Magazin Verlags-GmbH, Vaterstetten, Germany.

6.4. Exploitation

In today's ASP market ASP companies typically will try to cover as much of the value chain as they can, thus extending their business to a maximum profit. On the other hand it is currently still difficult for small and medium companies, such as Adesso to act as an application owner, because they cannot really prove that they deliver the promised service quality, while companies like IBM can easily act as full service providers. TAPAS has produced the notions of electronic service SLAs and hosting SLAs. With the circulation of TAPAS technology and concepts it will be easier for entrepreneurs to start companies dedicated to services such as an application owner, who does not actually host the application but relies on an ASP. In fact, during the preparation for the auction application it turned out that DaimlerChrysler has given the mandate to run a procurement auction platform to a mid-size company. This company in turn relies on an ASP/ISP company to actually run the systems. Besides the QoS guarantee and monitoring provided by TAPAS it can be observed, that monitoring QoS is still an issue, because in today's ASP business the ASP will monitor the SLA fulfilment. Only in rare cases ASP clients will monitor the fulfilment themselves. However, even with TAPAS technology somebody will have to evaluate the monitoring results. Considering other industries it seems a fair assumption that it makes sense for ASP clients to outsource monitoring issues to a third party.

Though TAPAS results are already available it is quite difficult to reason about future business types. Asides from SLA-related services in consultancy, software development and hosting, the availability of formal SLAs can foster a completely different type of business. When building web portals most companies are eager to integrate foreign services, depending on the portals target group. While Internet portals will typically integrate information and shopping services such as weather data, stock exchange rates or dedicated offerings for members, portals for employees tend to integrate internal and external services such as time sheets, travel booking etc. Focussing on external services it will be easier for start up companies to offer data services, because they can prove their SLA fulfilment and thus gain reliability. However, as markets and economies are changing rapidly reliable predictions will be rather difficult to construct.

Though TAPAS does not address all aspects of the ASP scenario, it is obvious that open and clear interfaces together with proven QoS will enable outsourcing of currently integrated services. During the last years it could be observed in the market that for instance Storage Area Networks (SAN) became quite popular, which resulted in companies offering even storage via Internet TCP/IP connections. In contrast to this outsourcing trend costs for disk space and memory have fallen to a level, where it does not pay to outsource the storage any more. It seems that human work is the more expensive factor, so that currently manual process steps such as software development and support are outsourced to foreign countries. However, based on the clear separation of concerns in TAPAS it is fair to say that business partners can find suitable division of work, thus allowing to outsource parts into new business types.

9. Conclusions

The main objective of the TAPAS project was to develop novel methods, tools, algorithms and protocols that support the construction and provisioning of Internet application services. The project planned to achieve this objective by developing QoS enabled middleware services capable of meeting Service Level Agreements (SLAs) between application services.

The project has achieved the main objective. We identified the following three key requirements for application service provisioning.

1. Enhancing the application hosting middleware platform to be QoS aware. This way, hosting platform will be better equipped to meet the requirements of the hosting applications. In the absence of such a feature, the only alternative available to an ASP is *over provisioning*, which is not particularly desirable.

2. Ability to ensure that all inter-organisation interactions are strictly according to the terms and conditions contracts in force. 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.

3. Ability to demonstrate that hosted applications are meeting the various QoS requirements of SLAs.

These three requirements underpin the design of the TAPAS architecture. Figure 2 shows its main features. If we ignore the three shaded/patterned entities (these are TAPAS specific components), then we have a fairly ‘standard’ application hosting environment: an application server constructed using component middleware (e.g., CORBA, J2EE). It is the inclusion of the shaded/patterned entities that makes all the difference.

The QoS management, monitoring and adaptation layer is intended to make the underlying application server QoS enabled (requirement 1). 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.

All cross-organisational interactions performed by applications are policed by the Inter-Organisation Interaction regulation subsystem (requirement 2). Techniques were developed enable relevant aspects of terms and condition 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. Techniques were developed to enhanced middleware to incorporate non-repudiable service interactions providing audit trails of service interactions.

It is necessary to be able to demonstrate that a hosted application actually meets the QoS requirements (e.g., availability, performance) stated in the hosting contract SLAs (requirement 3). For this reason, we developed an application level QoS monitoring service, which must also measure various application level QoS parameters, calculate QoS levels and report any violations. In TAPAS, QoS requirements in SLAs are specified using the SLAng language.

An important feature of TAPAS architecture is that the three subsystems can be deployed independent of each other. For example, an ASP might decide to use a ‘standard’ application server, without the need for QoS management features, because in a given scenario, over provisioning might be acceptable. The ASP still might need one or both of inter-organisation interaction regulation and QoS monitoring and violation detection subsystems. Another important feature of the TAPAS architecture is that the inter-organisation interaction regulation subsystem, as well as the QoS monitoring and violation detection subsystem could be provided by the ASP or one or more trusted third parties, thereby providing extreme flexibility in deployment.

In the ASP scenario there are quite a few business stakeholders for which QoS related technology is beneficial. First of all there are ASP clients, who are currently not or only quite rarely in the position to monitor the fulfilment of SLAs. It is quite obvious that the availability to monitor such services is beneficial to them instantly for existing ASP situations. For a future ASP client it is even more beneficial because the client is not only in the position to ask for an SLA but as well for monitoring access. For the duration of an ASP contract, clients will be even be able to identify differences and subsequently claim financial penalties. In order to achieve this goal, the ASP must use a TAPAS platform, providing data to externals, be it the client or a third party. The ASP is now able to find out the resource requirements before he has to enter a costly general SLA, i.e. the prediction preciseness is much better than today. Typically, ASPs will need to run load testing to configure the parameters appropriately, resulting not only in more precise SLAs but as well in a better resource usage in terms of used machines in a cluster node.   The benefit of the resource usage especially lies in the likeliness of the predicted load. If the average expected load results in usage of two machines (or, nodes) in a cluster while the more unlikely higher loads will require four machines in the cluster, the resource usage can be optimised. However, the main benefit can be achieved by providing a unique infrastructure that will host multiple applications of perhaps many clients.

References

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

[2] TAPAS Deliverable report D15, “TAPAS QoS-aware Platform: technology and demonstration”, September 2004.

[3] TAPAS Deliverable report D11, “QoS-aware Application Server: Design, Implementation, and Experimental Evaluation”, March 2005.

[4] TAPAS Deliverable report D9, “Component middleware for Trusted Coordination”, March 2004.

[5] TAPAS Deliverable report D10, “QoS Monitoring of Service Level Agreements”, May 2004.

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

[7] TAPAS Deliverable report D6, “TAPAS Architecture”, March 2005.

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

[9] Gerard J. Holzmann: Design and Validation of Computer Protocols. Prentice Hall, (1991).

[10] Gerard J. Holzmann: The SPIN model checker, Primer and reference manual. Addison-Wesley, (2004).

[11] Ellis Solaiman, Carlos Molina-Jimenez, Santosh Shrivastava: Model Checking Correctness Properties of Electronic Contracts. In Proc. of the Int. Conference on Service Oriented Computing (ICSOC03).Trento, Italy, Dec. 2003. Lecture Notes in Computer Science Vol. 2910, Springer (2003).

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

[13] TAPAS Deliverable report D8, “QoS adaptive Group Communication”, May 2004.

[14] *The JBoss project*, http://www.jboss.org

[15] *JGroups – A Toolkit for Reliable Multicast Communication*, http://www.jgroups.org

[6] http://www.objectweb.org

[17] BEA, ``BEA WebLogic Server 8.1 Overview: The Foundation for Enterprise Application Infrastructure", White Paper, August 2003.

[18] http://www-306.ibm.com/software/webserver/appserv

[19] TAPAS Deliverable report D14, “Third Year Evaluation and Assessment Report”, March 2004.

Appendix

TAPAS Industrial Advisory Board

The project will form an Industrial Advisory Board, whose membership will represent a cross-section of technology providers, end-users and middleware standards bodies. Regular meetings with the Board will help us in revising, where necessary, the objectives of the project. The membership of the Board includes:

Paul McKee (*BT exact Technologies*): is a team leader in the Distributed Computing and Information Systems research group at BT exact Technologies. He currently manages projects including collaboration with a number of Universities. His research is focused on large-scale distributed systems, particularly policy-based management and high performance event-based architectures for capturing and processing management information. Paul joined BT in 1989 and initially worked on high-resolution optical devices before moving to a distributed systems group where he worked on autonomous replication and low overhead consistency protocols. He has published over 40 papers and is a member of the IEEE Computer Society.

Andrew Watson (*Technical Director of the OMG*): graduated from the University of Cambridge in Computer Science and Engineering and spent two years at Hewlett-Packard's Bristol Research Centre, working on one of the first X.400 implementation. In 1989 Andrew joined the ANSA core team, working initially on the of the ANSA Computational Model and DPL, a language realising that model. Andrew then joined the Object Management Group (OMG) and chaired the ORB2 Task Force. Andrew is now Technical Director of the OMG and is responsible for the OMG's technology adoption process. Andrew also chairs the OMG's Architecture Board, a group of distinguished technical contributors from OMG member organizations. It was during Andrew's technical directorship that the OMG adopted the Unified Modelling Language (UML), the Common Object Request Broker Architecture (CORBA) and the CORBA Component Model.

Prof. Dr. Rudolf K. Keller (*Zühlke Engineering AG*): is the leader of the business unit Java Computing at Zühlke Engineering AG in Schlieren (Zürich), Switzerland. He is was an Associate Professor in the Software Engineering Group (GÉLO) at the Department of Computer Science and Operations Research at University of Montreal (UdeM). Before joining the faculty at UdeM in 1994, he was for several years a researcher at Montreal's CRIM research institute. Rudolf has taught at the School of Computer Science at McGill University and at University of California at Irvine, where he was a postdoctoral fellow from 1989 to 1991. He received a M.Sc. degree in mathematics from the Swiss Federal Institute of Technology (ETH) at Zürich in 1983, and a Ph.D. degree in computer science from University of Zürich in 1989. Rudolf's current interests are in object-oriented analysis and design, reverse engineering, design components and patterns, software quality, user interface engineering, business process modelling, and technologies for electronic marketplaces.

Dr. Marko Boger (*CEO of Gentleware AG*): is founder and CEO of Gentleware AG, a German company building UML-CASE-tools. He holds a PhD from the University of Hamburg where he worked as researcher on topics like UML, distributed systems development and e-Commerce for several years. He is author of the book 'Java in Distributed Systems', originally published in German (dpunkt-verlag) and later translated to English and published by Wiley. Marko was a key contributor to Argo/UML developer, which has now been developed by Gentleware into the Poseidon Toolsuite that is becoming part of Sun's Forte for Java development environment. Marko is a regular speaker at conferences, member of the program committee of the UML conference series and actively engaged in the standardisation of UML at the OMG.

Dr. Mark Little (*HP Arjuna Labs*): is a Distinguished Engineer/Architect, within HP Arjuna Labs., Newcastle upon Tyne, England, where he leads the Transactions team. He joined HP via a series of company acquisitions: Bluestone Software, Arjuna Solutions, which he was one of the founders. Before joining Arjuna Solutions he was for over 10 years a member of the Arjuna Distributed Computing team within the University of Newcastle upon Tyne (where he continues to have a Visiting Fellowship). His research within the Arjuna team included replication and transactions support, which include the construction of an OTS/JTS compliant transaction processing system.

Dr. Stuart Wheater (*HP Arjuna Labs*): is a Distinguished Engineer/Architect, within HP Arjuna Labs., Newcastle upon Tyne, England. He joined HP via a series of company acquisitions: Bluestone Software, Arjuna Solutions, which he was one of the founders. Before joining Arjuna Solutions he was for over 10 years a member of the Arjuna Distributed Computing team within the University of Newcastle upon Tyne (where he continues to have a Visiting Fellowship). His research within the Arjuna team included transactions and long-lived process support, which include the construction of a CORBA based transactional workflow system.

Dr. Tobias C. Kiefer (*Head of eTransaction Banking, Commerz NetBusiness AG/ Commerzbank Group*): Since April 2001 Head of eTransaction Banking at Commerz NetBusiness AG. Responsible for business development concerning epayments, mpayments, electronic bill presentment and payment, internet trust services and innovative transaction technologies and methods. Author of numerous publications and conference presentations concering the topic of services based on PKI, eBusiness strategies, banking strategies as well as speaker and moderator of specialized conferences with regard to strategies in e-commerce and etransaction banking. Main expertise in strategies, innovation management and business development.