ensure that the Service Request Manager, which acts as the adaptation to the ws smoothly, as unforeseen Service Providers Manager, which is used Service Allocation Manager, which is in charge of fail in providing runtime Application, which sends

Abstract—Designing the Qualities of Services (QoS) and verifying their ensuring at runtime are key issues addressed by today information systems. In this paper, we present our solution to these issues through a case study related to the provisioning of media streaming services. Our approach is based on the main concepts of the Service Oriented Architecture and exploits the Service Level Agreements to express QoS and adaptive mechanisms to monitor and change QoS at runtime.

Keywords-Quality of Service; Service Oriented Architecture; Service Level Agreement; adaptivity.

I. INTRODUCTION

Designing a Service Oriented Architecture (SOA) [3, 7, 9] implies addressing critical issues related to the quality aspects of the provided and requested services. Since a low level of quality at provisioning time can hinder the execution workflow of the service requester architecture, it is important to find a way to monitor and enforce those levels of quality directly and dynamically.

The first step of this process is to define properly the Quality of Service (QoS) [1, 2, 12]. When QoS are explicitly and clearly outlined, Service Level Agreements (SLA) [4] can be negotiated between the requester and the provider of a service. SLA are contracts that hold the description of the requested service along with the expected quality levels and the means for monitoring them.

Since SLA alone are not enough to ensure that the provisioning of a service flows smoothly, as unforeseen problems may arise at provisioning time, there is a need for a mean to further enforce the quality levels expected by the requester. Runtime adaptivity [8] provides support to achieve this goal by enabling the architecture to change a service provider dynamically, whenever quality levels negotiated through the SLA are not maintained.

In this paper we present an actual case study related to a streaming service provisioning. The case study exploits a SOA-based model which defines and manages SLA to describe and manipulate QoS and, implements adaptive strategies to monitor and change QoS at runtime.

The rest of the paper is organized as follows. Section 2 introduces briefly the architectural model. The case study is described in Section 3. Section 4 addresses the related work. Conclusions and further work are dealt with in Section 5.

II. ARCHITECTURE OVERVIEW

The goal of our solution is to design a SOA able to manage services characterized not only by their functional but also by their non-functional qualities, which in our proposed model are granted and maintained by SLA exploitation and runtime adaptivity.

SLA are used in our framework and stipulated between service consumers and service providers. These contracts include the description of the requested service, both in terms of functionality and of desired quality and other non-functional aspects. Quality at this level is described by a monitorable property and a precise indication on the minimum level required for that property.

SLA are particularly adapted to describe the issues related to the QoS, while they fail in providing runtime mechanisms to address possible service delivery problems. Therefore, our framework implements such mechanisms which enable it to adapt to the changed operating environment by modifying the SLA in order to assure future quality levels or, by changing a provider at runtime when one or more non-functional aspects cannot be guaranteed by the current contractor. SLA fit for this task since they can also describe penalties and adaptation strategies to overcome provisioning shortfalls.

Considering all the previously mentioned issues, we designed an architectural model divided in four main components as shown in Figure 1:

- Application, which sends the requests to the underlying architecture through the Service Request Agent (SRA);
- Service Request Manager, which acts as the framework’s input component, from the application’s point of view, and it is in charge of handling requests coming from the SRA;
- Service Allocation Manager, which is in charge of handling service provisioning by following the directions and information available in a document containing the workflow execution plan;
- Service Providers Manager, which is used whenever it is necessary to discover and select providers for a given service.

For more details on the architectural model see [6, 10].
III. MEDIA STREAMING SERVICES

The case study presented in this paper regards the provisioning of streaming services which focus on the delivery of content large in size and long in duration. The provisioning quality is fundamental and determinant for the satisfaction of the users’ expectations. Furthermore, media may be incompatible with the user’s playback capabilities, thus requiring the streaming service to transcode its contents in real-time.

In this context, customers have access to different Media Content Providers (MCP), each one characterized by its media library, media format and streaming qualities, granted by a network of available streaming service providers. Additionally, a number of transcoding services are present and characterized by the input and output formats and streaming qualities. The streaming of media contents involves a time-critical approach to its provisioning: a customer has to receive enough data to play the media content in real-time. Thus, tuning the stream properties such as video resolution, audio fidelity or bit-rate, is essential to determine the quality of the service provisioning.

Since QoS are important and have to be assured, SLA come to address this task by describing exactly the user’s needs and the provider’s grants. When QoS levels fall below the agreed minimum values, an adaptive architecture has the ability to overcome the problem by adjusting those required levels and negotiating a modified SLA. In cases where adjusting QoS levels is not enough to assure a sufficient quality, switching from one provider to another at runtime is possible by effectively canceling the SLA and by negotiating a new one with a new provider.

A. Details on the Workflow of the Streaming Services

In our case study we define a media-oriented streaming service as a composite type, since we require it to be able to provide more than just a stream for an already known content. We want it to be able to present Media Content Providers libraries and let us choose what to play based on the required QoS. We want the ability to schedule the service provisioning and we also want the ability to play content with formats not supported by our system.

A workflow diagram for the composite service is shown in Figure 2 and the description of each service node is the following:

- Data Collection – this service collects information on the user profile and playback environment.
- Media Content Selection – this is executed as a multiple service, thus contacting every Media Content Provider which is present in the Service Providers Pool.
- Sign-On Service – when the selected content requires user authentication on the owning Media Content Provider, this service is contacted and the information needed is identified.
- Payment Service – the selected content may not be free of charge and may require a form of payment: this service is therefore used and data is sent, if present.
- Media Content Provision Scheduling – when content has been selected and the required information is sent in case sign-on or payment were required, this service enables the architecture to negotiate an SLA for the content provisioning.
- Streaming Service – this is the service which actually provides the requested content, streaming it to the user application.
- Transcoding Service – since the Media Ticket holds the details on both audio and video formats and Case Packet Variables contain playback system details, such as supported audio and video...
codec which were retrieved with the Data Collection service, it is possible for the architecture to understand if the requested media will be played correctly on the user’s system. If this is not possible, a Transcoding Service is used in place of the Streaming Service.

B. Usage Scenario

A client uses the application on his laptop and requests the media streaming service execution. As he is interested in watching a movie after dinner he asks the application only to list media coming from providers which allow scheduling. The user selects his own Home Theater system as playback platform, thus requesting for contents suitable for a High Definition system. Video QoS are set to “HD Ready” and audio to “digital 5.1”. He used this service before and therefore, he has a preference for previously contacted providers which granted the requested quality.

The request is received by the framework and the Service Request Manager recognizes the composite service. It uses the Service Composition Manager to parse and decompose that request and its non-functional aspects, then the Workflow Execution Planner creates the document used for execution and forwards it to the Service Allocation Manager. Execution starts, information is collected and the user is prompted to select a media he prefers. When the selection is done the execution continues by authenticating the user with the data already contained in the document, as previously known providers were to be used, and payment is done using credits associated with the user account on the Media Content Provider.

The Media Ticket is finally created and negotiation starts with the Media Content Provision Scheduling, assuring that all the requested non-functional aspects will be granted. When this task is done the booking is accepted and the Media Ticket is updated with all the necessary information. The execution stops and the architecture waits for the booking time to advance, then it uses the playback system information and the Media Ticket to decide whether transcoding is necessary. Since the media is encoded with an unsupported codec, the execution proceeds to the Transcoding Service, sending the Media Ticket to it along with the system information.

Figure 2. The Media Streaming Workflow Diagram
The Transcoding Service uses the ticket and contacts one of the available Streaming Service providers, which were listed by the Media Content Provision Scheduling service, and receives the stream. Then it starts transcoding and sends the result to the architecture, which keeps incoming video content in a buffer. We suppose that a problem occurs in service provisioning since the current provider does not send the minimum data per second, as agreed in the SLA. Our architecture parses the adaptation rules and chooses to cancel the SLA and change the provider for Transcoding Service. The Service Providers Pool is prompted to send a new provider and the SLA Manager negotiates a new contract. As soon as the process is finished and required data is sent, provisioning resumes without the user experiencing any interruption in the video playback, due to a buffer that held enough content to be played during the switch. Streaming goes on smoothly and service provisioning ends successfully.

IV. RELATED WORK

In [2] authors propose a solution to manage QoS in multimedia Web Services. They introduce a new entity, called QoS Broker, which is responsible to collect QoS information, to select the most suitable service in response of a client request, and to analyze and keep statistics of the QoS during and after service executions. Conceptually, the QoS broker is a separate entity which resides between the clients and providers. From the implementation point of view, it can be part of a client, of a UDDI register, or even an independent service. The service selection considers not only service capabilities, but also the actual situation of servers, such as number and type of clients. The service selection is divided into two steps. The first step regards the discovery of services: the broker searches in its database services which can accomplish the client’s request. The second step, called service confirmation, is a negotiation between the service provider and the broker in order to assure that QoS can be actually provided to the client. After the start of the service provisioning, if clients notice that QoS is far behind what promised, the broker can cancel the service execution and switch immediately to another provider which can accomplish the initial QoS requirements. Furthermore, clients collect QoS information about the service execution and send them to the QoS Broker, which keeps statistics over them.

This solution is the one closer to our approach also because of the multimedia application domain. However, it is the QoS broker which manages the QoS during the execution of services. If it fails to find another service provider, clients cannot claim any penalties, as in our approach. Moreover, the booking of services is not addressed in this work.

V. CONCLUSIONS AND FUTURE WORK

Non-functional aspects such as Quality of Service play a determinant role in a Service Oriented Architecture and software engineers should understand and address them when modeling a similar solution [5, 12].

This paper has proposed a solution which considers not only the specification of QoS, but also the ensuring of their proper delivery at runtime through adaptive mechanisms which address quality and scheduling issues. Furthermore, services may be required both for an immediate execution or can be booked [11].

A limitation of the current system is the inability to compose services automatically. We use a fixed approach based on libraries of known and pre-established services to be able to plan their execution. Automatic service composition is the object of many current research projects, thus being not available in a well-established commercial platform. Semantics represent a promising approach to solve this issue, since requests may be of new, instead of known services.

REFERENCES