On the End-User QoS-Awareness of a Distributed Service Environment*

I. Widya[1], R.E. Stap[2], L.J. Teunissen[3], and B.F. Hopman[4]

[1]CTIT University of Twente, P.O.Box 217, Enschede, the Netherlands
widya@cs.utwente.nl
[2]TNO-FEL Twente, Enschede, the Netherlands
stap@fel.tno.nl
[3]KPN Research, Groningen, the Netherlands
l.j.teunissen@kpn.com
[4]PTS Software, Bussum, the Netherlands
frank.hopman@pts.nl

Abstract. A lot of attention has been given to network quality of service and efforts to make layers on top of the network also QoS-aware increase noticeably. This paper explores QoS-aware service provisioning at a level close to end-user's perception. It shows how end-user oriented QoS requirements have been elaborated in a high level design onto QoS support of a distributed service platform and how realized QoS can be monitored. A GameHall has been used as the application context for which validation of the explored and applied concepts and models for QoS specification and monitoring have been exercised.

1 Introduction

Along with the functionality of an on-line service, the quality of the service determines the service usability and utility, both of them influencing the likelihood of use of the service. Provisioning of QoS may therefore be a significant success factor to distributed service environments and a service provider may benefit from this provisioning to distinguish its services from other service providers.

This paper addresses some of the QoS provisioning issues described in the literature, e.g. [Me98, Au98]. In particular, it addresses the issue of specification, establishment and feedback of QoS at a level close to the user's perception, for example in accordance with a specified service level agreement [Ve99]. This paper exemplifies a procedure for this provisioning in a distributed service environment, particularly, in the context of a GameHall demonstrator that runs on a distributed service platform [Ba99].

For the specification of QoS at different layered levels of the distributed service environment and for the mapping of the specified or derived QoS parameters across those levels, this paper applies a (nested) user-provider model for QoS specification that has been elaborated in the project AMIDST [Si98, He00]. This model, in turn, is based on the ISO QoS Framework [IS97]. Complexity of QoS mappings (see e.g. [Na95, Xu00, Si00]) is mostly avoided by a heuristic approach that only uses first order relevance between parameters of the different levels. This approach is feasible

* This work has been sponsored by the Dutch Ministry of Economic Affairs within the project FRIENDS.

M.J. van Sinderen and L.J.M. Nieuwenhuis (Eds.): PROMS 2001, LNCS 2213, pp. 222-237, 2001.

within the reserved time span of the work and is sufficient for our demonstrator purposes. It simplifies QoS mapping considerably, i.e. it enables the use of mapping tables.

In the context of this work, the GameHall demonstrator is used as a vehicle to integrate the developed application oriented QoS framework into the service platform and to demonstrate the ability of this platform in differentiating resource allocations between the different types of games and end-user categories. In particular, the GameHall demonstrates the establishment and the monitoring of QoS. This monitoring is for the purpose of proactively informing the end-users on realized QoS to facilitate these users with service predictability [Bo99, Sa00] and the service accountability, which denotes the ability of the service to settle the benefit-cost factor up in accordance with realized performance.

This paper is organized as follows. Section 2 describes the business model of the demonstrator that provides the application domain context for the provisioning of QoS. Section 3 explains a procedure for QoS specification at interfaces and the mapping of QoS between these interfaces. Section 4 describes the provisioning and monitoring of QoS in the distributed service environment. Finally, Section 5 provides the conclusions of this work.

2 The Applied Business Model

This section describes the business model of the GameHall that has been used as the environment to exercise QoS provisioning, in particular the specification and the monitoring of QoS.

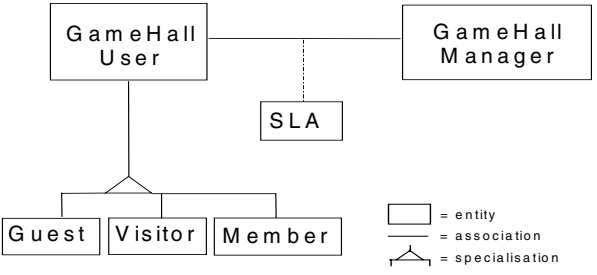

Fig. 1. The business model as a context for QoS provisioning

The GameHall offers to end-users different types of games, for example card-, chess- and car racing games. Like in the business model used in the Telecommunication Information Networking Architecture (TINA) [Ya97], the GameHall business model (Fig. 1) identifies the different stakeholders and expresses the relationship between these stakeholders. Each stakeholder has a different role in dealing with the use, control or provisioning of QoS. In the figure, the GameHall User represents a client and the GameHall manager is one of the representatives of the GameHall service provider.

In the model, users are categorized in different (sub-)roles. This yields different QoS requirements for the games in the GameHall. On one side, different requirements due to the different types and nature of the games and, on the other side, due to the

different end-user roles. The Service Level Agreement (SLA) models the kind of relationship between the users and the provider, it will accommodate users satisfaction better since it specifies what users may expect [Ve99]. Moreover, the model is also used as a mean to make end-users aware of the different QoS facilities for different end-user roles, e.g. to provide fairness of charging.

As the GameHall is used as a context to elaborate the QoS framework that spans over user-oriented to system- and network-oriented QoS, we discuss this model further.

2.1 GameHall Users

Three user roles have been introduced in the GameHall demonstrator:

- *Guest*: a guest of the GameHall may try some games. QoS support for this role is limited and the options to play games at advanced levels are also restricted;
- *Visitor*: a visitor pays for the played games and therefore gets better facilities than non paying guest;
- *Member*: a user with a subscription and who's (playing) profile is stored in the GameHall. As a regular GameHall customer, this role typically needs and expects a higher game performance than the other two roles.

To accommodate the different QoS needs at user level, QoS of games are categorized in values like "gold', "silver", or "bronze". GameHall users either are provided with a fixed setting of QoS or may choose between alternative settings, which have been pre-selected by the GameHall manager in accordance with the SLA associated to the role of the user.

To satisfy the users even better, users will receive feedback of achieved QoS from the provider. In this demonstrator, feedback is given in the form of a traffic light, which stays green as long as the agreed QoS is realized. It turns to orange or red in case the quality of service degrades or drops completely.

2.2 GameHall Service Provider

Two service provider roles have been introduced, the GameHall manager and the GameHall deployer. At the applied level of granularity, the deployer is not visible in the model shown in Fig. 1. This due to the role of the deployer that at the applied level of abstraction has no direct association to the GameHall users, but complements the role of the manager.

- **GameHall manager**

  The GameHall manager has the responsibility of issues related to the SLA, for example the implementation of the business (commercial and marketing) policy for the GameHall. The manager instantiates and fine-tunes the QoS configurations of the games that have been defined by (possibly third party) GameHall deployers.

  The GameHall manager, for example, may apply a policy that gives members a higher serving priority than others. Paying users, moreover, may get better performance of selected games or get more reliable games than non-paying guests or members that only try new games.

- **GameHall deployer**
  The responsibility of the deployer is to define the constraints for QoS profiles, QoS specification and mapping with respect to the nature of the games and, if appropriate, to the technical restrictions or capacity of the provider. For example, the deployer specifies all *invariant* aspects of games with respect to QoS.

2.3 Service Level Agreements

A service level agreement is a mean to specify the relationship between the GameHall service provider and the end-users who have different membership roles. A definition for a service level agreement is for example [Ve99]:

> *"A service level agreement (SLA) is an explicit statement of the expectations and obligations that exist in a business relationship between two organisations: the service provider and the customer[1]."*

Typically, an SLA contains [Ve99]:

- the type and nature of the service and the expected performance level of the service,
- the process of monitoring and reporting of realized service level,
- and many other aspects like management, other customer care and accounting aspects.

This paper elaborates on user-oriented QoS which settings are dependent on the role of the users and the type and nature of the games (Section 3). It also discusses a mechanism for monitoring delivered QoS (Section 4). Other customer care, management or accounting aspects are beyond the scope of this paper.

3 The Applied QoS Specification Framework

Establishment of QoS in a layered architecture includes the specification of QoS at an interface and the mapping down through the interfaces of the underlying layers until computing system or network resources can be determined for allocation. In this paper, we apply a user-provider model [Ha99, He00] to match at the interface the needed QoS by user entities to the QoS capabilities offered by the provider (Fig. 2).

The QoS interface model, shown in Fig. 2, originates from ISO QoS Framework [IS97] and has been elaborated in the project AMIDST [Si98]. User requirement in respect of QoS (abbr. as urQoS) represents the QoS of a user-to-user interaction. In this model, urQoS is in principle independent of the (middleware) provider that mediates the interaction between the peer users. The provider may however advertise its QoS capabilities in terms of QoS characteristics (denoted as midQoS in Fig. 2) and values or value ranges that it can support. Typically, a QoS characteristic is a quantifiable quality aspect of services, e.g. reliability or swiftness. In this paper, an aggregation of these characteristics and associated value ranges is called a QoS class. A provider may offer several QoS classes, each class optimised to a certain category of services, e.g., QoS classes suitable for chess up to car racing games.

[1] This paper does not distinguish between customer and end-user stakeholders.

3.1 QoS Specification Model at Interfaces

Specification of QoS at a user-provider interface therefore involves the translation of the required urQoS into a QoS class offered by a provider. To facilitate this, we use the concept QoS requirement [IS97] that specifies the qualifiers (e.g. maximum, mean) and the required values of the associated provider's characteristic. In this translation, a user entity may accept losses in accuracy of the translation or may choose between alternative QoS classes offered by one or more providers.

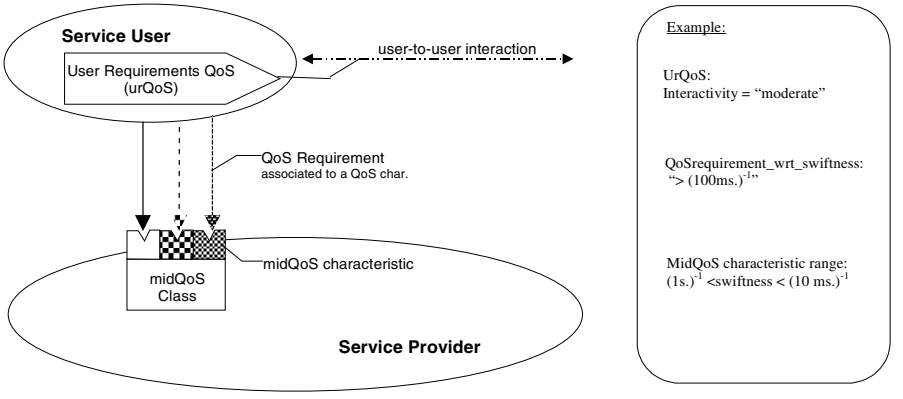

Fig. 2. User-provider model to specify QoS at interfaces

3.2 Layered QoS Model

In a layered architecture, a provider has to map the required midQoS characteristics and values mentioned earlier further downwards to QoS characteristics available at underlying interfaces (Fig. 3). In doing so, midQoS characteristics not only map "vertically" to lower level characteristics, e.g. QoS characteristics of local system components (sysQoS) and network elements (netQoS). In addition, some "horizontal" mapping may also be needed for peer capability matching, for example, to select and configure multimedia compression devices (e.g. MPEG-1 and MPEG-2) in a compatible manner. The triple headed arrow in the middleware layer in Fig. 3 models this multiparty mapping.

On the other hand, the double-headed arrow in the middleware layer models the peer-to-peer interactions that have to be QoS aware as well. These QoS needs before (multimedia) data is encoded and compressed are called media format level QoS and denoted as mfQoS in the figure. Furthermore, the QoS mapping in the layer may generate additionally the need for complementary peer-to-peer support such as retransmission protocols to increase reliability.

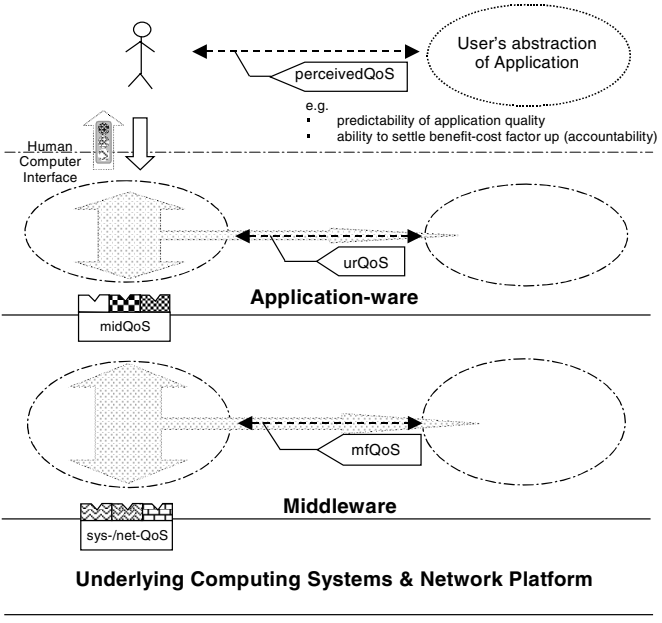

Fig. 3. Layered QoS Model for specification and monitoring

User-level QoS in the GameHall context:
As observed among others in [Bo99, Sa00, He00], QoS expected by users depends on the context of the use of the application. This so-called *task context* is characterised by the type of the application, the task of the invoked application for the user (i.e. the purpose of using the application) and the user role. In the GameHall demonstrator, the task context is characterised by the user role (i.e. guest, visitor or member), the type and nature of the game (e.g. a chess or a highly interactive car racing game) and the task (e.g. to try a game out or to play in a competition).

In our pursue to improve the attractiveness of the games in respect of QoS to the users, several articles from various disciplines have been studied:

- Models to predict the likelihood of use of distributed applications in a learning related context have been investigated in the educational science [Co00]. The so-called 3G model[2] of Kappetijn expresses this likelihood in terms of three aspects. Though not a fully appropriate translation, one of the aspects translates to benefit or pay-off. The other two aspects translate to ease of use and pleasure, attractiveness or one's engagement of use;
- In the area of stock investments, predictability of the performance of investment funds is an important aspect for investors. Investors also need an ability to settle up with these funds in accordance with the expectations and realized performance [Ba00];

[2] The 3G stand for the Dutch words "gewin", "gemak" and "genot".

- QoS studies in the human factors area among others identify feedback and also predictability as important QoS aspects for users [Bo99, Sa00].

Inspired by the previously mentioned studies, the GameHall demonstrator addresses the following task context dependent QoS aspects at user level (denoted by 'perceivedQoS' in Fig. 3):

- predictability of application quality:

  The demonstrator facilitates members to use a higher QoS setting compared to other user roles. On the other hand, paying users will be provided with more constant quality (e.g. higher reliability) than non-paying users. This GameHall policy is to better fulfill predictability of game quality as expected by end-users in respect to their task context.
- accountability that expresses the ability to settle up in accordance with realized QoS:

  In addition to the previously discussed policy, charges of used services will be adapted in case of drops in realized QoS. We believe that this ability to settle benefit-cost factor up will improve user satisfaction. However, one may argue whether this accountability is a quality characteristic or a kind of a "what-if" element of a QoS specification for the case that the commitment of the provider in respect to predictability is not met.
- task context dependent provider feedback:

  A QoS feedback mechanism will support the predictability and the ability to settle benefit-cost factor up. GameHall users will be facilitated by a monitoring component that displays a measure of realized QoS in the form of a traffic light positioned at the human computer interface HCI in Fig. 3. The setting of the traffic light depends on the task context of the user. That is, green signals for a GameHall guest and for a member generally mean different realization of QoS in respect to allocated computing system or network resources.

Applicationware QoS characteristics:
In this paper, we adopt the following urQoS dimensions introduced in [He00] and validate them for the GameHall case.

- *Availability*: UrQoS availability is the quality aspect of the peer-to-peer interaction of being present or ready for immediate use. In this paper, availability associates to both the instantiation and the use of a game, thus spanning over the whole lifetime of the game instantiation.
- *Fidelity*: UrQoS fidelity is the quality aspect of a peer-to-peer interaction of being good enough with respect to the task, such as the use of false colours to highlight important items. This urQoS dimension will not be elaborated further in the GameHall demonstrator.
- *Integrity*: UrQoS integrity is the quality aspect of the peer-to-peer interaction in maintaining the correctness of the interaction in respect to the source. In a car racing game, high integrity will preserve image resolution and synchronisation of the moving scene and the steer movements.

- *Interactivity*: UrQoS interactivity is the quality aspect of the peer-to-peer interaction of being responsive. In this paper, interactivity associates to the services (i.e. methods) of a game after instantiation of the game, and not to the responsiveness of the instantiation of the game itself (availability). It is clear that urQoS dimensions are not necessarily orthogonal.
- *Regulatory*: UrQoS regulatory is the quality aspect of the peer-to-peer interaction of being in conformance with the rules, the law, or the established usage in the application domain. In a GameHall, regulatory means the conformance to the rules of the games, for example, to ensure equal opportunity to win a game even in the case that the environment is heterogeneous.

Middleware QoS characteristics:
The QoS capabilities of a middleware provider that offers services to application components may be expressed in terms of classes of QoS characteristics, including the (ranges of) supported values. One may also specify QoS using a language for QoS that support specification of measurement units [Fr98]. The following list describes some of the midQoS characteristics introduced in [Ha99]:

- *accessibility*: MidQoS accessibility is the quality aspect of a service that represents the degree of being capable of serving a request. It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time.
- *accuracy*: MidQoS accuracy is the quality aspect of a service that represents the degree of conformity to the true value, an ICT standard or a well-accepted custom (e.g. video accuracy value "VCR" or "studio-TV", which specification may refer to the CCIR.601 standard).
- *reliability*: MidQoS reliability is the quality aspect of a service that represents the degree of being capable of maintaining the service and service quality.
- Other described midQoS characteristics are *linkage unity* (the degree of parts being linked as a complex whole, e.g. synchronisation of multimedia), *swiftness* (the degree of how immediately or quickly something is done), and *urgency* (the degree of being important for immediate attention or processing).

Platform QoS characteristics:

QoS characteristics associated to the underlying platform are denoted as comQoS, netQoS and sysQoS. They represent the QoS aspects of the communication session layer (Fig. 4), the network platform and the computing system elements, respectively.

The applied distributed service platform, upon which the demonstrator runs contains a Load Balancer component that among others supports sysQoS characteristics like priority (i.e. the scheduling priority of a process in a computing node) and workload accessibility (i.e. the chance a request to schedule a game component in a computing node be accepted). The platform also supports comQoS characteristics associated to multimedia streams and is therefore closely related to mfQoS introduced in the layered model (Fig. 3), but defined at an intermediate level interface (Fig. 4). For video streams, the platform supports width and height dimensions of a frame, frame-rate, quantisation bits/pixel, colour-depth and delay.

Audio related comQoS characteristics supported are the number of audio channels, sampling frequency, quantisation, and delay. NetQoS characteristics are delay and bit-rate.

QoS Mapping:
For the demonstrator, we use a heuristic approach to by-pass the complexity of mapping of QoS characteristics across layers (see e.g. [Na95, Xu00, Si00] for QoS mappings). At user level, the GameHall manager typically determines the urQoS values of the games in accordance with its policies and service level agreement. Further, a simple benefit functions relates realized urQoS to the earlier mentioned traffic light that provides QoS feedback to users.

At the level of the middleware service, a subset of urQoS dimensions often relates to a subset of midQoS characteristics. An urQoS value may also map to a whole range of alternative values of midQoS characteristics forming a subspace in an offered QoS class. We have similar situations in the mapping of midQoS to netQoS and sysQoS characteristics, especially if special resources such as multimedia coding devices are involved.

Though these QoS mappings are generally M:N relations, we nevertheless apply translation tables that are mainly suitable for 1:N mappings. To enable this, the most dominant relation between the QoS parameters will only be taken into account. This paper does not address horizontal capability matching aspects nor the protocols that may be needed to complement QoS mappings such as retransmission protocols.

4 Implementation of the QoS Framework into the Demonstrator

This section discusses the incorporation of the QoS framework into the distributed service environment. First, the applied QoS specification and mapping procedure will be discussed. Then, the QoS extension of the service environment will be explained in brief. Thereafter, the monitoring mechanism to provide user feedback will be briefly described.

4.1 Implementation of QoS Specifications and Mappings

As was discussed earlier, the QoS settings of games depend on the task context and are constrained by the SLA. In the case of the demonstrator, we use tables to implement QoS specification at interfaces and to map QoS across layers.

It is the responsibility of the GameHall deployer to define the templates of the QoS specification tables, the constraints between the QoS values in the table cells, and the list of values that a GameHall manager can use in a succeeding configuration step. Table 1 exemplifies the constraints of urQoS dimensions of a chess game that can be played at different skill levels and players settings. The constraints are for example $A_i \leq A_{i+1}$; C_i = {“only_once_semantic”, X_i} with movement-to-speech synchronisation tolerance $X_i \geq X_{i+1}$ for i = {1,..,3}, and Z specifying an equal QoS setting for involved players for fairness of the play.

Table 1. Chess user requirements in the deployers' perspective.

CHESS Game		Availability	Interactivity	Fidelity	Integrity	Regulatory
with computer	**Level 2**	A_1	B_1	"don't care"	C_1	"don't care"
	Level 1	A_2	B_2	"don't care"	C_2	"don't care"
other user		A_3	B_3	"don't care"	C_3	Z
simul		A_4	B_4	"don't care"	C_4	Z

A GameHall manager has the responsibility to instantiate the templates in accordance with the SLA and its policy. Table 2 exemplifies the availability of the chess game for the different user roles. It shows what a user may expect in respect of playing quality. In this example, a guest player may only play against the computer at skill level 1. The table also shows that members may select between two QoS settings.

Table 2. Example QoS availability settings per role.

CHESS		Availability		
		Guest	**Visitor**	**Member**
with computer	**Level 1**	"ch_bronze"	"ch_bronze"	"ch_bronze_premier" or "ch_silver_premier"
	Level 2		"ch_bronze"	"ch_bronze_premier" or "ch_silver_premier"
other user			"ch_silver"	"ch_silver_premier" or "ch_gold_premier"
simul				"ch_gold_premier"

The availability values in Table 2 are partially ordered string typed values that may also be expressed using a QoS language, see e.g. [Fr98, He00]. These values map further downwards to midQoS characteristics like accessibility, reliability and swiftness of invocations (Table 3). As expected, QoS values at higher levels are more specific to the type of the game and task context, values at lower levels are typically more generic, that is, the prefix "ch_" specific for chess games can be omitted in the lower level values.

Table 3. Example availability to reliability and accessibility translation table.

CHESS Availability	Accessibility	Reliability
"ch_bronze"	"moderate"	"moderate"
"ch_bronze_premier"	"high"	"moderate"
"ch_silver"	"moderate"	"high"
"ch_silver_premier"	"high"	"high"
"ch_gold_premier	"premier"	"premier"

Other QoS mappings can be elaborated in a similar way. One may indeed raise the question when mappings of string typed values ever ends or become concrete for the implementation. A downward mapping stops if it reaches values which interpretation have been specified in technical terms in a standard or are known from human factor studies (e.g. 44.1 KHz sampling rate, 16 bits/sample quantisation for uncompressed "CD" quality audio, 80 ms for a tolerable audio-video lip-synchronisation and 40 ms for a moderate lip-synchronisation [St93]).

4.2 QoS Extension of the Distributed Service Environment

This section briefly describes the high-level component architecture of the distributed service platform that has been extended with QoS. A time-sequence diagram related to the establishment of QoS for a game is used to explain the QoS extension of the platform.

As mentioned earlier, the distributed service platform is based on TINA [Ba99]. With respect to the layered QoS model (Fig. 3), the Service Session layer of the platform (Fig. 4) relates to the Application-ware layer in Fig. 3. The Communication and Connectivity Session layers relate to the Middleware layer and the Network layer (including the end-terminals) relates to the computing system and network platform in Fig. 3.

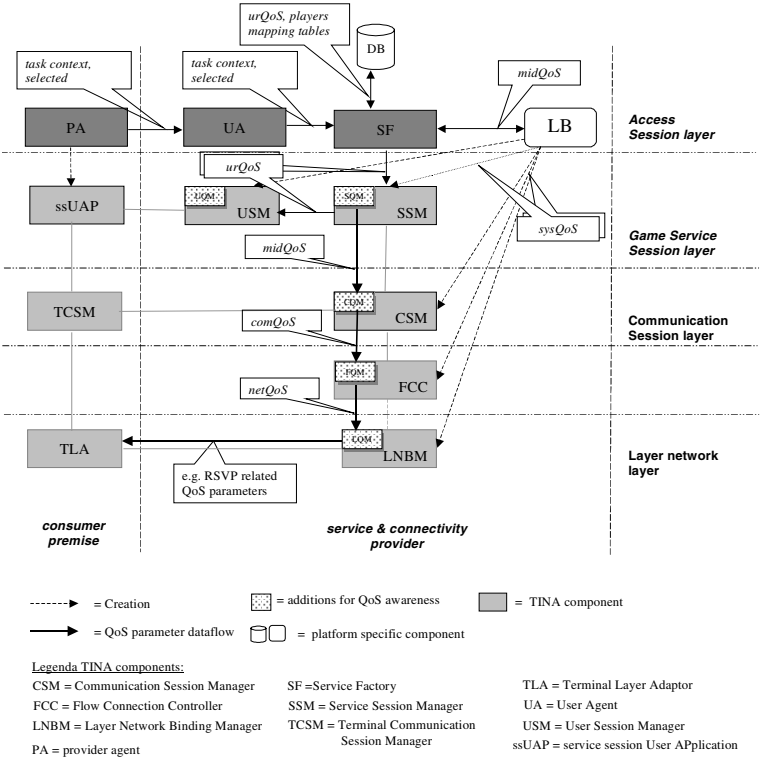

Fig. 4. Component architecture to provision a QoS service session

Some of the components in Fig. 4 are specific to the service platform [Ba99]. The database component DB contains information needed by the platform to provide the services to the users. It among others contains the user profiles and the QoS tables discussed earlier. The component LB is a load balancing component specific to the platform. This component does not only perform load balancing (including midQoS to sysQoS mappings), it also creates other TINA components, therefore, acting as a component factory. Though possibly not fully TINA complaint, these peculiarities are not significantly relevant in the context of this work that focus on QoS establishment and monitoring.

Moreover, the GameHall demonstrator is realized as one of the services of the distributed service environment, this enables accommodation of other demonstrators in the service environment. However, this means that a game is a sub-service of the GameHall service. Instantiation of a game will therefore by-pass several steps of the Access Session.

The QoS extensions are represented by different QoS Manager components (abbreviated as xQM with the prefix "x" = S, C, L, U standing for Session, Communication, Layer-network or User, respectively (Fig. 4)). These components are responsible for all QoS issues of the containing component. The call-outs that label the component interactions expose the conveyed QoS characteristic level. For clarity, the figure only depicts relevant components of a single party involved in the establishment of a multiparty service session.

Fig. 5 shows a time-sequence diagram of a game invocation after the instantiation of the GameHall service (i.e. after the user identified him/herself and received a list of games, including the settings in accordance to his/her role, and players who are also in the GameHall). This means that the service session User Application (ssUAP) and the User Agent (UA) of the GameHall service session have been instantiated.

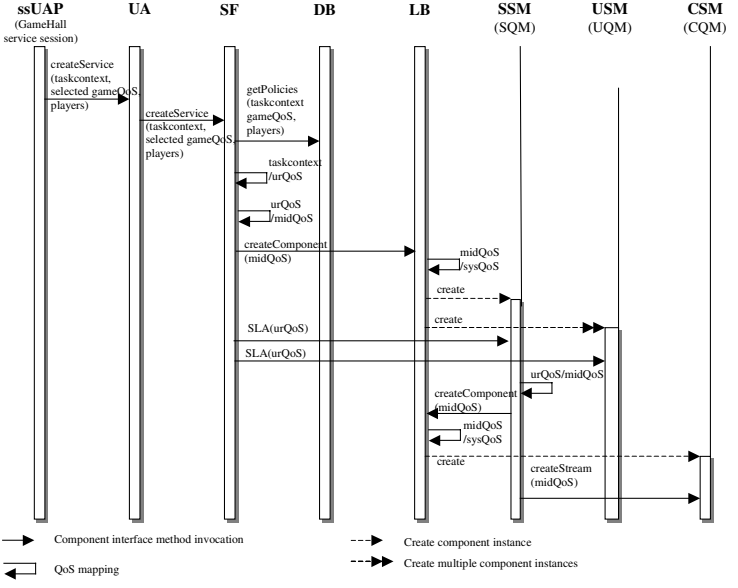

Fig. 5. Service session establishment in respect of QoS

A game session establishment procedure (see Fig. 5):

- In response to a user request for a game, the originating site ssUAP of the GameHall service session sends a createService request transparently via the Provider Agent (PA) to the UA. The request conveys the task context, the invited players (in case of a multi-party game) and the required game QoS in case a set of QoS settings are offered to the user.
- The UA forwards the request to the Service Factory (SF). The SF retrieves from the database (DB) component the urQoS dimensions and values, which are associated to the task context and the required game QoS, and the relevant urQoS/midQoS mapping tables. The SF uses these tables to compute the QoS settings of the components that will be created (Fig. 5).
- In the platform, the SF asks the Load Balancer (LB) to create the session layer components involved in the game, i.e. the Service Session Manager (SSM) and the User Session Manager (USM) of all parties. The LB component transforms the received midQoS parameters to sysQoS parameters that will be used to create the applicable components. In the user domain the PA creates the ssUAP of the game session. In the current version, creation of this ssUAP in the user premise is not yet QoS supported.
- The SF also informs (via SSM) the created USMs about the agreed urQoS to enable control of the traffic light (see also the section on QoS monitoring).
- For the establishment of QoS aware communication connections, the SSM maps the urQoS to midQoS parameters and provides these in the request to the LB to enable derivation of the sysQoS parameters and to create the CSM using these parameters. The CSM is responsible for the further processing of the midQoS parameters to a specific QoS aware network implementation, e.g. yielding an RSVP reserved stream that is supported by end-to-end QoS.

4.3 QoS Monitoring in the Distributed Service Environment

The main objective for the monitoring of the QoS performance is the implementation of the task context dependent feedback to the end-user. The following considerations have influenced the designed feedback mechanism:

- minimal monitoring-related interactions at the boundary between the user premises and the provider domain, therefore isolating provider ‘s measurement data flow within the provider’s domain;
- reuse of components that receive QoS mapping tables during QoS establishment; and
- keep the design open for extensions in which a trusted third party performs (parts of) the monitoring.

As shown by the component architecture in Fig. 6 and the time-sequence diagram in Fig. 7, realized QoS performance information flows in the reversed direction of the signalling during the QoS establishment phase, traversing the components involved in the downwards QoS mapping. Method invocation is applied to transfer the information for the reasons listed above.

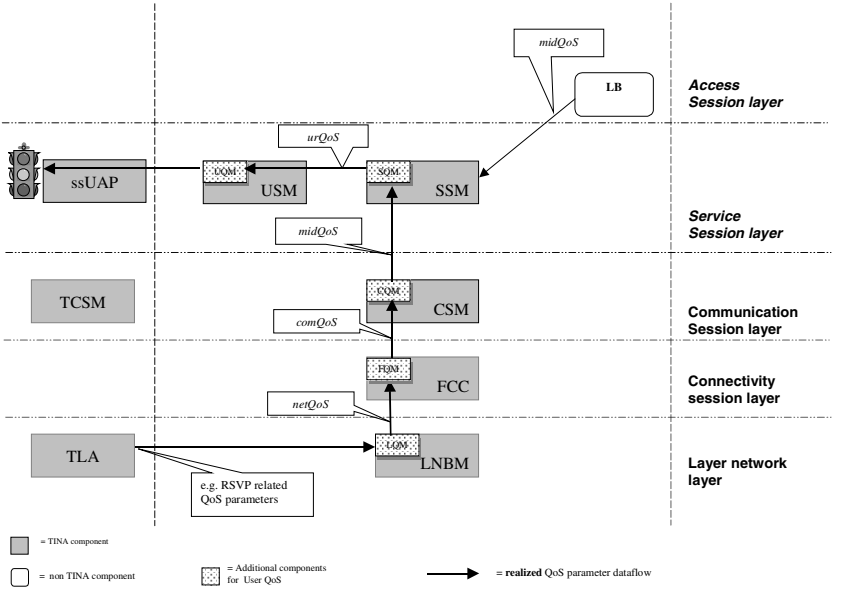

Fig. 6. Component architecture to monitor a QoS service session (with call-outs representing *realized* QoS values)

The figures also show the control of the traffic light by the UQM component of the USM. In particular, the differences in realized and required urQoS values, conditional to the type and nature of the game, the skills level selected and the user role, are equally weighted by a benefit function and then signalled to the traffic light. In case of not complying with the service level agreement, the provider may decrease a charging meter or adapt the QoS provisioning. The latter is however a challenging issue for further research.

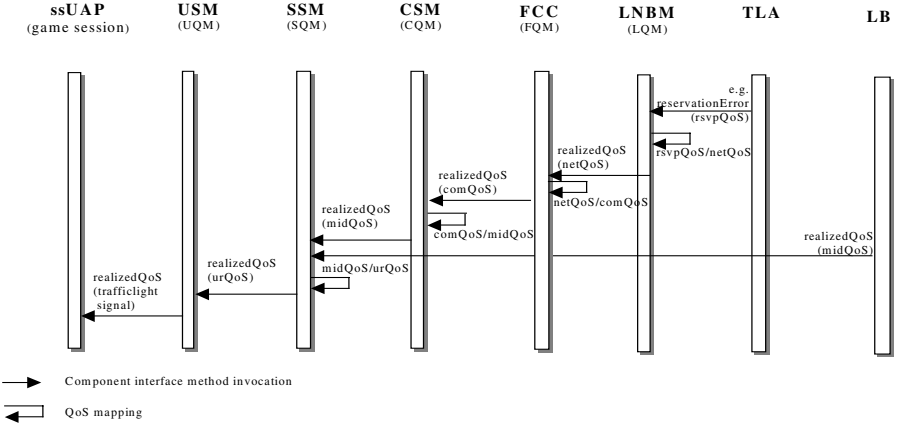

Fig. 7. Service session QoS monitor

5 Conclusions and Outlooks

In the context of a GameHall, we have exercised a QoS development trajectory starting from QoS specification at a level close to user's perception towards its high-level implementation onto a distributed service platform. To make this work feasible within the time-span of the collaboration project that develops the platform and the GameHall, we have explored and applied existing concepts and models for QoS specification and have used a heuristic design approach. This latter to avoid derivation or implementation of complicated M-to-N QoS mapping mechanisms. This work is to be considered as a validation exercise of the applied concepts and models and is a preliminary step to get experience to develop more advanced techniques for QoS specification and mechanisms for mapping of QoS.

The described model to specify QoS at interfaces has been successfully applied at several levels of the layered architecture of the service environment, at a level close to end-users' perception up to network-oriented levels. Within the layered framework, one may therefore apply similar techniques or mechanisms for QoS negotiation or mapping.

This paper works out user-oriented QoS issues which settings are dependent on the context of the users' task, determined by the role of the users, the purpose of the task and the type and nature of the games. It turns out that a lot of attention has to be put in task analysis in order to capture required QoS that should improve likelihood of service usage. To increase the satisfaction of the end-users, further (human factors) study has to be done on the relation between the required and the perceived QoS in accordance with the task context and established service level agreement.

Establishment of QoS in a layered architecture includes the specification of QoS at an interface and the mapping down through the interfaces of the underlying layers until computing system or network resources can be reserved. Several times we have experienced difficulties of QoS mappings, especially the reversed mapping of several QoS parameters to a QoS parameter one level higher, despite the simplified (table based) mapping approach applied. Further exploration of (research results of) M-to-N mappings, for example in combination with QoS specification languages, for the provisioning of end-user QoS is another interesting challenge.

This paper also discusses an elementary mechanism to monitor delivered QoS for end-users feedback. In the future, a third party that objectively appraises service performance may perform these monitoring of services. This approach also enables the benchmarking of services of different providers. This situation is anticipated, since the developed architecture distinguishes between provisioning and monitoring of services.

Other customer care issues, like accounting aspects and QoS control and management to maintain agreed QoS guarantees, are beyond the scope of this paper. Further study on the linkages between QoS policies, accounting and billing of delivered services is a necessity to enable commercial exploitation of QoS aware services.

References

[Au98] C. Aurrecoechea, A.T. Campbell and L. Haw, "A Survey of QoS Architectures", Multimedia Systems Journal, Special Issue on QoS Architectures, May 1998;

[Ba99] H.J. Batteram, J-L. Bakker, J.P.C. Verhoosel, and N.K. Diakov, "Design and Implementation of the MESH Services Platform", Proceedings of TINA'99 Conference, Oahu, Hawaii, April 1999;

[Ba00] B. Bakker, "Een kroon op het fonds: nieuwe beleggingsfondsengids 2000 van Nyfer", in newspaper attachment "Geld Telt", NRC Handelsblad, Oct. 21st 2000 (in Dutch);

[Bo99] A. Bouch and M.A. Sasse, "Network Quality of Service – An Integrated Perspective", Proc. RTAs '99, Vancouver, June 1999;

[Co00] B. Collis and N. Pals, "A Model for Predicting an Individual's Use of a Telematics Application for a Learning-Related Purpose", International Journal of Educational Telecommunications, 6(1), pp. 63 – 103, 2000;

[Fr98] S. Frølund and J. Koistinen, "Quality-of-service Specification in Distributed Object Systems", Distributed Systems Engineering, 5, 1998, pp. 179 – 202;

[Ha99] A. van Halteren et al., "QoS architecture", Amidst project deliverable D.3.1.2, 1999, http://amidst.ctit.utwente.nl/workpackages/wp3/index.html;

[He00] C. Hesselman, I. Widya, A.T. van Halteren, and L.J.M. Nieuwenhuis, "Middleware support for media-streaming establishment driven by user-oriented QoS requirements", Proceedings of the Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS2000), Enschede, the Netherlands, Oct. 2000, pp. 158 – 171, Springer Verlag LNCS 1905;

[IS97] ISO/IEC JTC1/SC21 N13236, "Information Technology – Quality of Service – Framework", Geneva, 1997;

[Me98] J. de Meer and A. Hafid, "The Enterprise of QoS", tutorial presentation at the Middleware Conference, Sep. 1998, Lake District, U.K., http://www.fokus.gmd.de /research/cc/tip/employees/jdm/private/jdmPubList1998.html;

[Na95] K. Nahrstedt and J.M. Smith, "The QoS Broker", IEEE Multimedia, 2(1), pp. 53 – 67, 1995;

[Sa00] M.A. Sasse, "User-centred quality of service: why value is everything ..", slides at QofiS 2000, Berlin, Sep. 2000, http://www.fokus.gmd.de/events/qofis2000/slides /27ix00/s012-user-and-market/sasse.pdf;

[Si00] F. Siqueira and V. Cahill, "Quartz: A QoS Architecture for Open Systems", The 20th IEEE Int. Conf. On Distributed Computing Systems, pp. 197 – 204, Taipei, Taiwan, April 2000;

[Si98] M. van Sinderen, "AMIDST Application of Middleware in Services for Telematics", http://amidst.ctit.utwente.nl, 1998;

[St93] R. Steinmetz, "Human Perception of Media Synchronization", IBM European Networking Center, IBM – Technical report no 43 9310, 1993;

[Ve99] D. Verma, "Supporting Service Level Agreements on IP networks", Macmillan Technical Publications, ISBN 1-57870-146-5, 1999;

[Xu00] D-Y. Xu, D-D. Wichadakul, and K. Nahrstedt, "Multimedia Service Configuration and Reservation in Heterogeneous Environments", The 20th IEEE Int. Conf. On Distributed Computing Systems, pp. 512 – 519, Taipei, Taiwan, April 2000;