An Approach To User Management Of Broadband Services

Paul Coates & Jeremy Ellman
MARI Computer Systems Ltd, England

Abstract. This paper describes an approach to the In-Call control of services as developed by the ASCOT project. Emphasis is made on the application of usability principles to the practical problems presented by real service deployment within an Integrated Service Engineering environment. The ASCOT approach to the graphical control of services in-call will be described including details of practical experiments and any conclusions drawn.

Introduction

Broadband communications offer ordinary users the possibility of virtually limitless bandwidth. This in turn will potentially support a huge number of services. These services will have far greater variety than services in use today presenting systems designers with a significant challenge: Service architecture must be extensible to new service types, and use and control of new services must be "intuitive".

Whilst this may be an impossible goal, at the very least, services must be controlled in a common way. The situation has an exact parallel with the design of the control mechanisms in a car, a typewriter keyboard, or a computer operating system. Even if the skill is difficult to acquire it is transferable. As long as services have idiosyncratic controlling mechanisms their uptake by the general public will be limited.

The problem this paper addresses is consequently an approach to the design of a generic services control mechanism that reflects telecommunications engineering needs. A key aspect of this approach is usability. The general public must be able to learn the control mechanism, and be able to apply it to a variety of services.

In this paper we shall also review a generic view on Broadband Service Engineering within the RACE program known as Integrated Service Engineering, including the implied constraints this has for end user service control. We shall then go on to the user management of services that supports ISE. Finally we shall report actual experimental trials of the approach and conclude with a discussion of emerging ideas.

Integrated Service Engineering

Integrated Service Engineering (ISE) is the preferred RACE viewpoint on services. It is an architecture that is designed to allow the development and deployment of new services and to support the open market in telecommunications which is one of the policy goals of the EU. If there were no common approach for building and managing services then every service provider would have to provide its customers with different tools to use their services.

The Open Service Architecture (OSA) model of services uses an ISE approach, and was originally developed by ROSA (R1093) and further developed by CASSIOPEIA (R2049). It describes services in terms of parties which communicate using service tasks, each service task consisting of one or more service components, operating with a specific quality [CSF C110]. Figure 1 shows the relationship between these different parts of the service description.
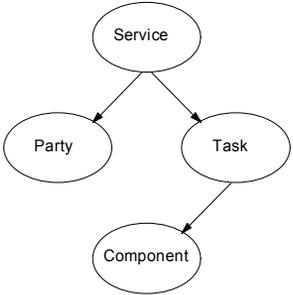

Figure 1 *The Service Model*

The architecture is made up of several levels. At the lowest level is a point to point model of the telecommunications network that routes communications between network access points. For the sake of argument, we assume that any network access point can access any other network access point. Services are therefore universally accessible.

Services are defined in a object model that collects together service components. These components contain audio, video, etc. data types, and are further characterized by the service standard they support such as MPEG-2 for video, or JPEG for picture, etc. Depending on the data type, they will also contain other parameters such as a data stream is bursty, or components are required on demand, etc.

It is important to note that service users may include automatic devices (such as video sources, or computers) in addition to people. Therefore the concept of a service party, representing both people and machines, is used rather than end users.

Using a service implies the instantiation of a service object and making the connections in the network between service parties and their service access points.

The service is then controlled using a generic signaling mechanism called service control elements [CFS C220].

Integrated Service Engineering presents a huge problem to the designer of a generic service user interface. In order to be flexible, the ISE approach explicitly includes huge quantities of information. At the very least it is unnecessary for users to see all this information, and at the worst it is would be extremely confusing for them. By taking an object oriented view of ISE, information can be selectively hidden.

Information Hiding is however a great advantage of the modern object oriented programming paradigm. This permits different views to be presented on the same information depending on who wishes to view it. Thus, an end user may only wish to see the parties in his call, whereas service engineers may wish to see the lowest level components, and their connection types.

The Ascot Approach

The ASCOT project has devised an intuitive view of services, that is easier to understand and control than a menu driven interface. To keep the interface simple non-essential information is hidden from the end-user and processed automatically. The approach also provides in-call service control that respects ISE, is generic, and is designed for usability. In ISE users control services by generating service control elements (SCEs). In ASCOT, the user has a distributed application that generates these signals.

The advantage of the ASCOT approach is that a distinction is enforced between the service and the end-users' service control interface. This entails that future broadband service users need only learn one service interface. Service provider's on the other habd will only need their services to interpret a standard set of control signals, SCEs. The actual interface to the service will be common between applications.

ASCOT represents services to end users using a graphical editor helping to provide a simple view of services by providing intuitive icons and symbols to represent the parties, tasks and components of a service. The graphical presentation allows an end-user to describe a service without the necessity of technical knowledge about a service. The end-users view of the service is also generic so a user only has to learn a single interface to be able to control all services. The graphical editor is capable of both creating service descriptions and configuring services while they are executing.

The ASCOT graphical editor was based on work by the RACE-1 usability project URM (R1077). This "User Reference Model" was designed to show what users consider important in a service. The view of the service is based on "Reference for User Services" (RUS) diagrams. RUS diagrams represent communication tasks between geographically separate entities. An alternate menu driven interface provides a more sophisticated configuration intended for network professionals.

Figure 2 shows a RUS diagram of a simple service containing three parties connected by service tasks. The service components are shows as icons on the task lines.
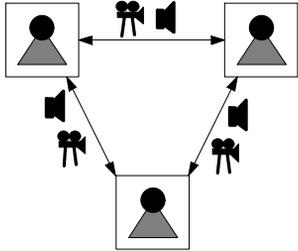

Figure 2 *A RUS diagram*

The RUS editor only requires a minimal amount of information from the user, and shows services in terms of parties, which are connected using service tasks. Each party is represented by a picture of the party and it's name. Each task is shown as a line between two parties. Service components are shown by small icons positioned on the task line. A service description can be built by adding and connecting parties, service tasks and service components.
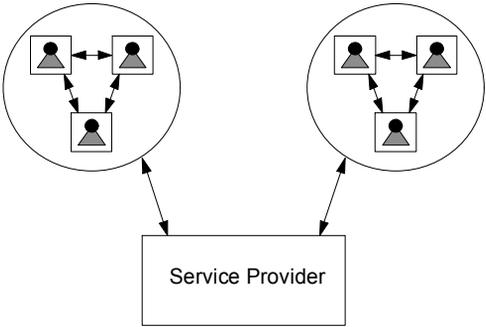

Figure 3 *The ASCOT service architecture*

The ASCOT toolset sends Service Control Element (SCE) messages to the service provider which runs the service. The SCEs are used to manipulate the service to suit the needs of the service parties. The service provider also send SCEs to the other parties in the service to maintain individual copies of the service description. Access control can be used to inform only those users that have permission to know about the change in the service. Figure 3 shows the service architecture used by the ASCOT project, the toolset for each party communicates with the service provider for the service being used. The service provider is being further developed by the DRAGON project (R2114).

The RUS editor was extended to provide in-call control of the service by manipulating the graphical service description. A service can be configured by selecting part of the diagram and either connecting or disconnect it from the service. Depending on the parts of the diagram selected the appropriate Service Control Element messages are sent to the service provider and the OSA service description is adjusted accordingly. Colour is used to indicate whether an item is connected or disconnected from the service. The quality of a service component is the only other property that can be changed using SCEs and is performed by selecting the appropriate component from the RUS diagram and using a quality slider.

Information Hiding

The ASCOT toolset provides a generic interface which can be used to control all services. An end-user only needs to learn how to use a single interface to be able to control all types of service. For some services it may be easier to use an interface specific to that service which removes any additional functionality required to handle other service types needed for a generic interface. A specific interface can also provide a simpler, more controllable interface to the user.

For a generic interface to become useful a degree of information hiding must be introduced, common interfaces should be as easy and powerful to use as possible, and information not relevant for a specific service should be hidden from the user. Information hiding is where all the technical details of a service are abstracted away from the user, e.g. an end-user should not need to know if a service component uses a bursty or continuous data stream, this information can be deduced by other means such as the type of service component being used, etc.

Many of these different approaches to information hiding are described here,

- The graphical view of a service provides a much clearer view of the entire service description presenting the majority of the service information in a single diagram which can be understood at a glance.

- Each party and service has its own icon used to represent it in the toolset. Icons are used to maintain the graphical view of services. Icons can be sent with service descriptions between toolsets and service providers to support a consistent graphical view of a service. Services can be stored with parties as part of the service description, so any icons used should be stored with the service description. Each user should be able to replace default icons with their own icons.

- Aliases are used so each user has shorter more familiar names for services and parties. Each user maintains their own set of aliases. Since only the icon and the alias are used to graphically identify each party or service, the icon and alias combination must be unique. An end-user

should never have to see a numeric identifier for a party or service as that is what the alias is for.

- Parties can be collected together into groups to perform repetitive operations in a single step. An operation on a group will be applied to each member of the group in turn. Groups also have icons and aliases to graphically represent them, similar to parties. An alternative to groups is multiple selections. When multiple items are selected in the RUS editor any operations performed will be applied to all the selected items.

- For more complex services different views of the parties, tasks and components can be shown so that only the information needed by each party is presented. This information could be selected by personal preference or by using access controls for different types of information.

- When changing the quality of a service component the user should select a particular service quality that is appropriate to the type of service component (Such as 64KHz for audio, or 30 Frames Per Second for video etc.) In ASCOT the user chooses quality from a slider that goes from high to low, and toolset automatically selects the corresponding quality for that service component type.

- More detailed features such as Mandatory or Optional service components are hidden from the user completely. They can not be influenced by the user configuration of the service and instead are infered by the toolset based on the type of service, and pre-defined default settings in the service description.

- Nested operations such as a party leaving a service. For a party to leave a service all tasks that party is using must first be disconnected. For a task to be disconnected all components in that task must first be disconnected. This single operation will therefore generate a number of different SCEs. Figure 4 shows the result of a single party leaving a service. Each SCE is propagated to the other parties in the service.
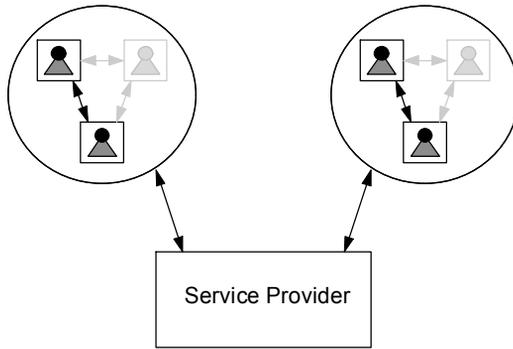
Figure 4 *In-call control of services*

- As more parties are added to the service the RUS diagram can become more complex and more difficult to read. There are two simple methods available to alleviate this problem, coordinates for parties can be stored with the service description, or automatic layout algorithms can be used to make best use of the available space. Automatic placement would work by applying an attraction between parties that are connected by a task, and applying a repulsion between parties that are not connected. After several iterations the diagram will settle down to it's best physical placement for the parties [GEM1]. This technique removes the need for users to worry over where to place party icons and how to arrange the parties to produce the clearest diagram.

- The toolset should be customizable. If you are using a generic interface you will be using it most of the time so it must be configured correctly for each individual user. Customizing colours, dialogue layout and different methods of entering data, do not provide information hiding, but other customizations do. Setting an experience level such as novice / expert can automatically adjust how much information is hidden from the user and how much control is assumed by the toolset. If certain facilities are used frequently with the same settings, for a more novice user this stage of configuration may be automatically performed for the user using this setting.

These approaches are practical applications of Information Hiding in an ISE context. They illustrate how the full power of ISE may be used to define services, but allow a much simpler view of the service. This has been conducive to the development of a generic, usable, service control interface.

For a useable system a fully featured help system should be provided. As well as providing the usual bubble and context sensitive help, the toolset help system could be oriented towards the particular service being used at any one time.

The generic interface described so far refers to the control of connections within the service, but do not describe the types of information sent down each connection. Components are described by type and sub-categorized to specify the specific type of data being transmitted, e.g. a video component in MPEG-2 format. Some component types can not be sub-categorized as they are service specific, so instead of sending information which the toolset can not interpret a user interface language should be used to describe a dialog which interfaces with the remote service. For example, a video retrieval service can be established using the RUS editor but the service requires its own interface to ask the user to select the required video signal, i.e. which piece of video do you want to watch. Several examples of a similar user interface language exist including HTML and Tcl/TK. Such a generic interface would be the next stage of development for the toolset.

Actual Implementation Trials

The ASCOT approach has been incrementally refined in actual implementation experiences in both the ASCOT and DRAGON projects.

The RUS editor and information hiding arose due to an early implementation of the toolset which used a dialog containing all information needed to describe a service. This interface was very complex and was only suitable for network / service providers. End-users found the information very confusing and unnecessary for their needs, and it was difficult to describe services in terms the toolset required. The RUS editor was then developed to provide a graphical interface which is simpler and more intuitive to use.

The first release of the ASCOT toolset could control a standalone security monitoring application involving a remote video scenario implemented over B-ISDN channels, implemented by Deutsche Telekom. The RUS editor could be used to construct simple service descriptions, but was deficient in a number of areas [DEL12]. In-call service control was performed using a separate dialogue with pairs of list boxes containing connected and unconnected, parties, tasks and components. This text based interface was difficult to use and it was decided to produce a graphical interface by incorporating it into the RUS editor.

The second release of the ASCOT toolset contains a number of improvements based on results from evaluation experiments, and requirements from the DRAGON project. The toolset is being used by the DRAGON project as a user agent to provide service control. Improvements include,

- Drag and drop control of the editor,
- Colour icons using scanned images,

- Improved integrity checks while a service description is being constructed and modified,
- Different views of the service,
- Group icons,
- Support for multicast tasks and components,
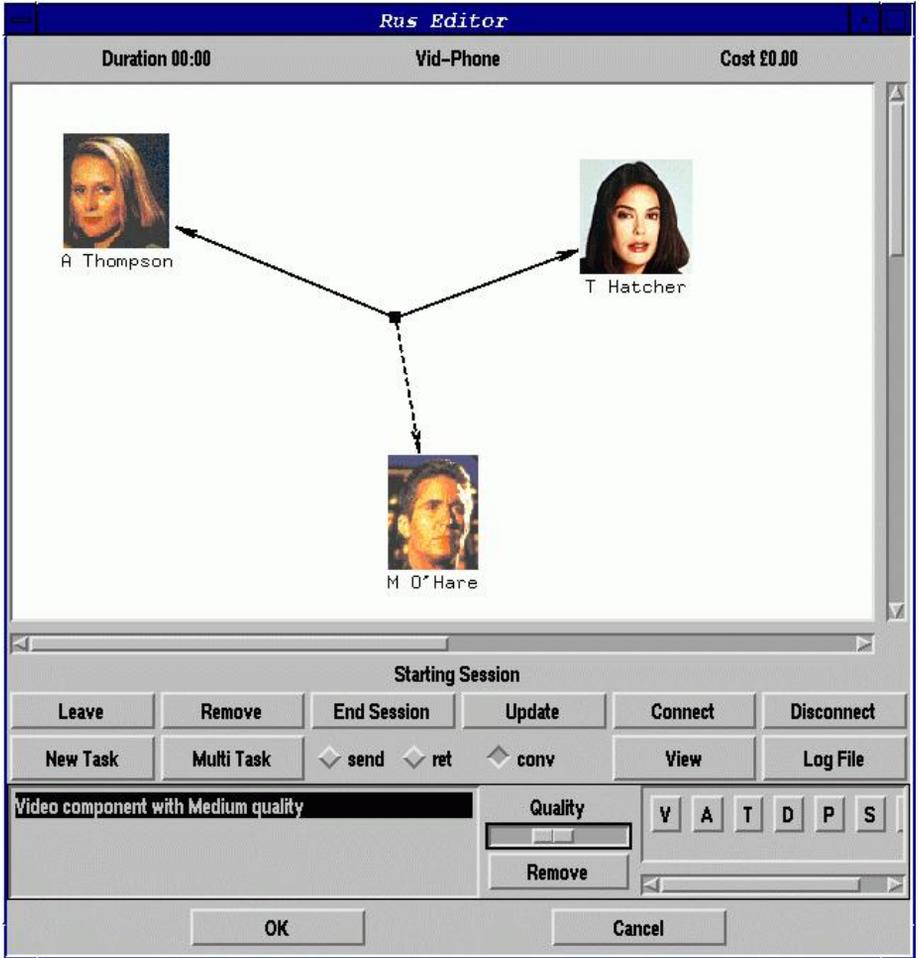- In-call service configuration incorporating the latest version of the Service Control Element messages

Figure 5 *The RUS editor from the second release of the toolset*

Service Control Elements were implemented as control message strings which were passed around the network using a CORBA based communications package.

Conclusions

ASCOT has illustrated how Integrated Service Engineering may be used without compromising either usability, or the ability of end-users to control services in-call. Furthermore, new SCE messages have been created and others modified to provide needed additional functionality.

There are very few projects that have used a Service Control Element approach. SCEs have influenced standards in CCITT and ETSI, as part of the RACE common functional specifications (CFSs). New SCE messages have been created and others modified to provide additional functionality, and a very useful graphical tool has been developed to configure a service while it is executing.

The toolset could become integrated into an end-users Personal Information Management (PIM) tool to reside along side diary and address book applications. An address book could record the Service Access Point (SAP) for a user or machine, and the diary could be used to reserve resources for services for scheduled meetings, etc.

A need for such a system is going to become more necessary when multimedia services become more widely available.

References

[CFS]		IBC Common Functional Specification, Issue D, December 1993.
	B210	"The Concept of IBC and it's Relationship to ISDN",
	B311	"Methods for the Characteristics of the Operational Requirements of IBC",
	C110	"Methods for the Specification of IBC Services",
	C220	"Service Control Elements".

[DEL12]   B.Hill, A.Whitefield, I.Denley, "Usability Evaluation". ASCOT Deliverable, R2089/UCL/ERG/DS/P/012/b1, May 1994.

[DEL17]   P.A.Coates, J.Middlemass, "Functional Specification of Improved ASCOT Toolset". ASCOT Deliverable, R2089/MAR/MCS/DS/P/017/b1, May 1995.

[GEM1]   Laszlo Szirmay-Kalos, "Dynamic Layout Algorithm to Display General Graphs". Graphic Gems IV, Edited by Paul S Heckbert, 1994, Academic Press, Inc.

[URM]   R1077 User Reference Model for IBC, 1992, Service Definition Methods